UNIVERSITY OF VAASA

FACULTY OF TECHNOLOGY

INDUSTRIAL MANAGEMENT

Marek Kwitek

A FEASIBILITY STUDY OF AZURE MACHINE LEARNING FOR SHEET METAL FABRICATION

Master's thesis in

Industrial Management

VAASA 2016

TABLE OF CONTENTS

ABBREVIATIONS

ACC: Accuracy

ANN: Artificial Neural Network

AUC: Area under a Curve

AUROC: Area under Receiver Operating Characteristics

BI: Business Intelligence

CBM: Condition Based Maintenance

CRISP-DM: Cross-Industry Standard Process for Data Mining

DAG: Directed Acyclic Graph

DDD: Data Driven Decision Making

DIKW: Data Information Knowledge Wisdom

ERP: Enterprise Resource Planning

EV: Expected Value

FN: False Negative

FNcost: False Negative Cost

FNR: False Negative Rate

FP: False Positive

FPcost: False Positive Cost

FPR: False Positive Rate

HDFS: Hadoop Distributed File System

KDD: Knowledge Discovery in Databases

MAE: Mean Absolute Error

MES: Manufacturing Execution System

ML: Machine Learning

MLaaS: Machine Learning as a Service

MMH: Maximum Margin Hyperplane

MTBF: Mean Time between Failures

MTTF: Mean Time to Failure

NN: Neural Network

PdM: Predictive Maintenance

PP: Prima Power (Company)

PR: Precision-Recall

R^2: Coefficient of Determination

RMSE: Root Mean Square Error

ROC Curves: Receiver Operating Characteristics Curves

RUL: Remaining Useful Life

SEMMA: Sample Explore Modify Model Asses

SMF: Sheet Metal Forming

SMOTE: Synthetic Minority Oversampling Technique

SPC: Specificity

SSE: Sum of Squared Errors

SSR: Sum of Squares for Regression

SST: Total Sum of Squares

SVM: Support Vector Machine

TN: True Negative

TNR: True Negative Rate

TP: True Positive

TPR: True Positive Rate

TTF: Time to Failure

LIST OF FIGURES

LIST OF TABLES

UNIVERSITY OF VAASA	
Faculty of Technology	
Author:	Marek Kwitek
Topic of the Master's Thesis:	A Feasibility Study of Azure Machine Learning for Sheet Metal Fabrication
Instructor:	Petri Helo
Degree:	Master of Science in Economics and Business Administration
Major Subject:	Industrial Management
Year of Entering the University:	2009
Year of Completing the Master's Thesis:	2016 **Pages:** 186

ABSTRACT:

The research demonstrated that sheet metal fabrication machines can utilize machine learning to gain competitive advantage. With various possible applications of machine learning, it was decided to focus on the topic of predictive maintenance. Implementation of the predictive service is accomplished with Microsoft Azure Machine Learning. The aim was to demonstrate to the stakeholders at the case company potential laying in machine learning. It was found that besides machine learning technologies being founded on sophisticated algorithms and mathematics it can still be utilized and bring benefits with moderate effort required. Significance of this study is in it demonstrating potentials of the machine learning to be used in improving operations management and especially for sheet metal fabrication machines.

KEYWORDS: Predictive Maintenance (PdM), Machine Learning (ML), Microsoft Azure Machine Learning

1. Introduction

Research presented in this document builds upon **winning and innovative ideas** that were presented by the author during the competition organized by the case company. Suggestions for improving competitive advantage of the case company were based on **utilization of the machine learning** technology and techniques. Scope of the ideas presented during the competition was very broad. Therefore, in order to make it feasible for the single authored thesis it needed to be narrowed. Following **research question** is stated:

How can the sheet metal industry use machine learning for improving its operations management?

Potential was noticed, but many questions were left open simply due to time limitation of the competition. Therefore, to further widen the acceptance of the idea and its understanding among various stakeholders at the case company following **thesis purposes** are identified:

- Explain proposed technology benefits from the business perspective.
- Provide better introduction and description of the technology aiming at less technical stakeholders.
- Empirically demonstrate that machine learning implementation can be achieved relatively easily with the Microsoft Azure Machine Learning.

Before mentioned thesis purposes are achieved by the realization of the **thesis objectives** listed below:

- Thesis document containing:

    - Introduction into Machine Learning topic for less technical stakeholders.
    - Description of the business benefits arising from the usage of the Machine Learning technology.
- Demo experiments with the Microsoft Azure Machine Learning Studio
    - Experiments implemented and deployed.
- Demo application
    - Application demonstrating in a very simple manner possible usage of the predictive service created with the Microsoft Azure ML Studio.

As with any kind of endeavour, resources are usually limited. Naturally this is also the case with the research presented here. The **thesis scope** is defined as following:

- Data collection is not in the scope. Objectives can be realized without real data. Additionally, data collection would require additional financial commitment from the case company. The goal of this thesis is to demonstrate that such commitment will pay back.
- The use of the Microsoft Azure Machine Learning came as a requirement from the case company. Some other functionalities of the Azure are utilized.
- Basic application example. Implementing anything more sophisticated would require too much of the constrained time resource and would not bring much of the benefit considering the lack of the real data.

The goal of the following two paragraphs is to give brief reasoning for the need to collect and act based on data. It relates to what can be achieved with the machine learning. Predictive maintenance is given as an example of the machine learning application for the case company. However, in the broader perspective this thesis aim at promoting the values that can be extracted from the data.

Companies that base its decisions on data ("**data driven decision making**" or DDD) prove to outperform ones that does not. Research shows that DDD is correlated with higher productivity and market value. Evidence also exists on its association with measures of profitability such as ROE or asset utilization. Therefore, DDD can be seen as intangible asset, which increases company profitability and is recognized by investors. (Brynjolfsson et al. 2011)

Similarly, the importance of data can be seen from modified version of the well-known DIKW (**data-information-knowledge-wisdom**) model. Good decisions build on data. As we move up through the pyramid from the data to the decision, the value that it represents to the business increases.

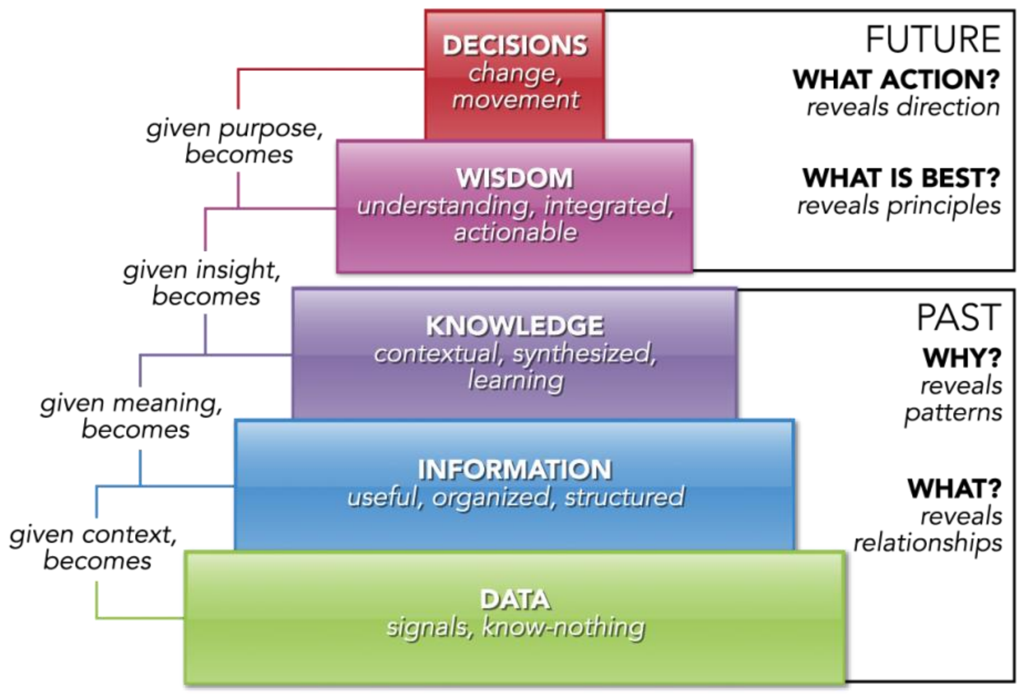


Figure 1. Modified DIKW model (Swindoll 2011)

1.1.Background Information

In order to distinguish itself on the market, companies need to gain advantage over their competitors. Data and data science capability should be seen as company's key strategic assets. Recognizing it and properly exploiting both can give a **competitive advantage**. It is important to do consideration of potential benefits, which can be derived from the data in the context of the applied strategy. Meaning that the value of those mentioned strategic assets depends on the company's strategy. (Provost & Fawcett 2013)

Looking at it from other perspective. Unrealized potential competitive advantage can become competitive disadvantage, once competitor gain it first.

In this paper we will look at the **Machine Learning** and **Predictive Maintenance** (PdM) implemented with it as a one option which can provide that kind of competitive advantage for the case company.

Predictive Maintenance implemented using Machine Learning techniques uses readings from machines' sensors over the time to learn relationships between changes of those sensors' values and historical failures. With the assumption that monitored asset has degradation pattern which could be observed from the sensors available. If assumption holds then Predictive Maintenance can do following; depending on implementation (Microsoft 2015a):

- **Predict the failure**. It could be further divided into prediction of:
    - **Remaining Useful Life** (RUL) or **Time to Failure** (TTF) for a given component of the machine or machine as the whole using **regression**.
    - Likelihood that error will occur during given **time frame** in case **binary classification** is used.
    - Asset failing in different **time windows**, e.g. probability of asset failing this week, next week or two weeks from now. It can be achieved when **multi-class classification** is used.
- **Predict type of failure**
- **Diagnose the failure – root cause analysis**
- **Detect and classify failure**
- **Provide failure mitigation** prior to its occurrence or **maintenance actions** once failure already happened

The advantage of the predictive maintenances is to reduce the maintenance cost. It is achieved by minimizing of the maintenance time and parts needed, and at the same time maximizing machine availability.

We can split maintenance costs into following groups:

- Cost occurring from the replacement materials.
- Maintenance labour cost.
- Machine not being operational, machine being idle, not used.
- Bad quality product being produced by malfunctioning machine.

With the time based maintenance, when we want to make sure that machine is always operational. Maintenance needs to be done more frequently than it is actually needed, meaning resources are wasted for non-value adding activity. Predictive Maintenance monitor condition of the machine and predicts right time for the maintenance. Those should be less frequent, therefore resulting in saved resources, meaning more profit can be made. It can also detect when something abnormal happens and therefore reduce down time, further increasing return on investment into the machine which provides predictive maintenance functionality.

The usage of the Predictive Maintenance has number of benefits, following is list of few of those:

- Cost effectively decreases asset failures. (Gulati 2012)
- Minimizes maintenance overtime and generally maintenance hours. (Gulati 2012)
- Minimizes spare parts inventory. (Gulati 2012)

- Provides more insights into the machine performance. (Maintenance Assistant 2014)
- Minimizes production hours lost. (Maintenance Assistant 2014)

Predictive Maintenance is one the applications of the machine learning. To provide that functionality machine needs to collect, store and analyse significant amount of information. As the information storage costs continues to decrease and are already very low it brings additional potential benefits that could be generated sooner or later from mining of this data.

1.2. Case Company Introduction

Prima Power operates in the Sheet Metal Forming (SMF) industry as the manufacturer of the sophisticated sheet metal forming machines. In its line of products it has machines/solutions with various level of automation available. The most sophisticated ones are fully automated and require little human interaction. Following are product lines provided by the Prima Power (Prima Power 2016):

- TheBEND – sheet metal bending.
- TheCOMBI – multifunctional systems, e.g. punching and laser cutting.
- TheLASER – sheet metal cutting with the laser with some products providing also welding and drilling capabilities.
- ThePUNCH – sheet metal punching.
- TheSYSTEM – versatile range of solutions which combines functionalities of the Prima Power machines into one automated production line. With additional functionalities such as automatic storage.

- TheSOFTWARE – number of additional software solutions, which further optimize machines operations. With Tulus® software family capable to:
    - Parts order and inventory handling
    - Work scheduling and machine capacity monitoring
    - Control and monitor machines' tasks
    - Control material storage
    - Production reports
    - Integrate with the ERP (enterprise resource planning) and act as MES (manufacturing execution system).

Prima Power products are used in many industries, listing just few as aerospace, agricultural, automotive, domestic appliances, elevators, HVAC, hospital and lab equipment, etc.

Prima Power is an innovative company, which always searches and is open for new ideas. It can be also clearly seen thru its close cooperation with the University at different levels.

2. Literature Review

In this section we focus on the current work related to the maintenance with special emphasize on the Predictive Maintenance with machine learning (ML). There are other ways in which predictive maintenance can be achieved than with ML. However, machine learning automates the process and transfers the knowledge regarding maintenance from the human to the machine. Thanks to that knowledge can be easily stored and shared between machines.

Rules and failure prediction models can be learn using several analytical approaches, listing few as correlation analysis, casual analysis, time series analysis and machine learning. Additionally to failure prediction, same techniques can be used for detecting root cause and wear rate of components which could be further used to balance between machine's maintenance time, costs and availability (Li et al. 2014). However, here we are focusing solely on the machine learning approach to the problem.

Machine learning techniques are widely used in various interdisciplinary contexts. Therefore, similar techniques and methods are labelled with different names. It is not easy to draw clear boundary between terms such as machine learning, statistical learning, predictive analytics, data mining and data science. Those all are closely related and we will not focus on differences between those but instead we will draw from all of them.

Similarly when it comes to the term Predictive Maintenance (PdM) which is closely related to the Condition Based Maintenance (CBM). Both are in oppose to the preventive maintenance or otherwise saying time based maintenance. First two, PdM and CBM, monitor equipment and trigger need for the maintenance when condition of some component requires so. Later two terms, preventive maintenance and time based maintenance, refers to regularly performed maintenances which are done at specific intervals, regardless to the condition. We can also have Corrective Maintenance approach in which maintenance activities are performed once failure occurs. (Coraddu et al. 2015)

Predictive maintenance approach described in this thesis uses before mentioned machine learning techniques to build models which can predict expected lifetime of the component. It finds patterns and relationships between various attributes in historical data which contributes to the known defects. It then uses those models to make predictions based on real-time data.

Other industries already recognized the benefits of the predictive maintenance with aerospace industry as an example. Airbus A380 which first flew in 2005 collects information on over 200 000 aspects of its every single flight. This vast amount of information let to implement predictive maintenance with machine learning. And there is much to gain, as maintenance accounts for approximately 10 percent of an airlines' operating costs and is a root cause for nearly half of accounted delays (Hollinger 2015). Those delays caused by unscheduled maintenance besides being inconvenient for the travellers, cost the air carriers estimated $10,000 for every hour of maintenance, repair and overhaul. Not to mention the significant safety hazards arising from inefficient maintenance works (Koch 2012).

There are number of identified challenges for implementation of the predictive maintenance with the machine learning which need to be addressed (Li et al. 2014).

- Measurement errors of the sensors cause some problems especially when collecting information from different machines which are not co-located. In those cases measurements can be impacted by the environmental variables.
- Big data which brings opportunities but challenges as well. Number of data that can be collected from the sensors monitoring machines can be enormous. There is much to learn and benefit from it but it also presents its own challenges on storing and processing. Taking as an example the modern aircraft which can generate data in the range of terabytes per single flight (Hollinger 2015).
- Interpretability of the rules by the human operators. Models created by the machine learning algorithms are not always easily interpretable by humans, sometimes it is even impossible. However, same techniques can be used to create simplified models which maybe sometimes will not perform as well as complex counterparts but which are easy to understand by humans. Therefore, accuracy needs to be sometimes sacrificed over interpretability. We can refer to it as interpretability-accuracy trade-off.

Microsoft has provided template for building predictive maintenance with the Microsoft Azure Machine Learning (Microsoft 2015a). It will serve as the base of the solution that is going to be developed for the Prima Power in the scope of this thesis.

Predictive maintenance is recognized also by other major players such as:

- SAP – with its "SAP® Predictive Maintenance and Service" solution available either on premise or in the cloud (SAP 2015 & Langlouis 2014).

- IBM – has its own "IBM Predictive Maintenance and Quality 2.0" solution (Negandhi 2015).
- Cisco – advocating for and providing support for interconnectivity between sensors and other elements of the system (Bellin 2014).
- Bosch – own predictive maintenance solution build on top the Bosh IoT Suite (Bosch 2014)
- Software AG (Software AG 2015)

2.1. Predictive Maintenance

Maintenance is defined here as an actions taken to assure asset productive capacity at a target level, which is not more than designed level. It includes both upkeep and repairs. It is also concern with retaining functional capabilities. (Gulati 2012)

Maintenance of the asset should be seen as an important part of the operations management. Well maintained assets should result in improved production capacity while reducing maintenance costs. It is achieved through (Gulati 2012):

- Reduced production downtime
- Increased life expectancy of the asset
- Reduced overtime cots occurring from unplanned maintenance
- Reduced cost of repairs. Often small cause creates severe damage to the asset if let alone and not fixed.
- Reduced costs occurring from poor product quality due to product rejects, reworks, scrap, etc.
- Reduced costs due to missed orders

- Identifying assets with excessive maintenance cost. Identifying the cause and taking corrective actions such as operator training, replacement or corrective maintenance of the asset.
- Improved safety and quality conditions

Several approaches to the maintenance can be identified, below are the most commonly used ones (Gulati 2012):

- Predictive Maintenance (also known as Condition Based Maintenance) – aims at assessing of the asset condition. It is achieved through periodic or continues monitoring of various asset's characteristics. The goal is to schedule proactive maintenance activities on the asset at the most optimal time. In doing so it needs to predict asset condition in the future based on what could be learn from the past. Some techniques used involve measurement of vibration, temperature, oil, noise, etc.
- Preventive Maintenance – commonly applied strategy, which schedules maintenance base on calendar or asset runtime. Given parts or components are replaced regarding to theirs condition and for some it base on theirs state. Most commonly, this kind of maintenance means changing some parts even so those could possibly last longer.
- Corrective Maintenance – sometimes called run-to-failure. Asset runs until it fails. Maintenance starts after failure is detected, equipment is then restored to the operational state or replaced with the new one. It may be sometimes correct approach. Especially for inexpensive and not critical assets.

Predictive vs. preventive maintenance. Question can arise on differences between those two approaches. However, answer is not simple. Different experts presents sometimes contradictory opinions. Following differentiation is author’s own opinion based on topic study from various sources over the time. Predictive maintenance monitoring in contrast to preventive inspection is not causing machine to be offline. Some predictive maintenance techniques require on site visit, but measurements are done without process interruption. Unlike in the case of the preventive maintenance. We would refer to preventive maintenance also when parts are replaced at a given time without regards to theirs condition. E.g. routine change of the oil.

Predictive maintenance presented in this research is achieved with machine learning technology and techniques. It should be noted that it is not the only approach available. However, utilizing machine learning techniques in the author’s own opinion seems to be the most natural evolution. Most commonly used technologies till now rely on the human inspector physical presence in close proximity to the machine. Expected step forward would be to equip machines with sensors. Then collect data and do basic data manipulation locally before sending it to the cloud where it could be further analyse. Curious reader should check also on the topic of the Industrial Internet of Things (IIoT).

Benefits of the Predictive Maintenance:

- Cost effectively decreases asset failures. (Gulati 2012)
- Minimizes maintenance overtime and generally maintenance hours. (Gulati 2012)
- Minimizes spare parts inventory. (Gulati 2012)
- Provides more insights into the machine performance. (Maintenance Assistant 2014)
- Minimizes production hours lost. (Maintenance Assistant 2014)

According to Gulati (2012), predictive maintenance can result in:

- Reduction in maintenance cost: 15-30%
- Reduction in downtime: 20-40%
- Increase in production: 15-25%

2.2. Total cost and availability consideration

Amount of effort put into maintenance activities and which aim at reaching high reliability of the asset should be considered from the perspective of the total cost.

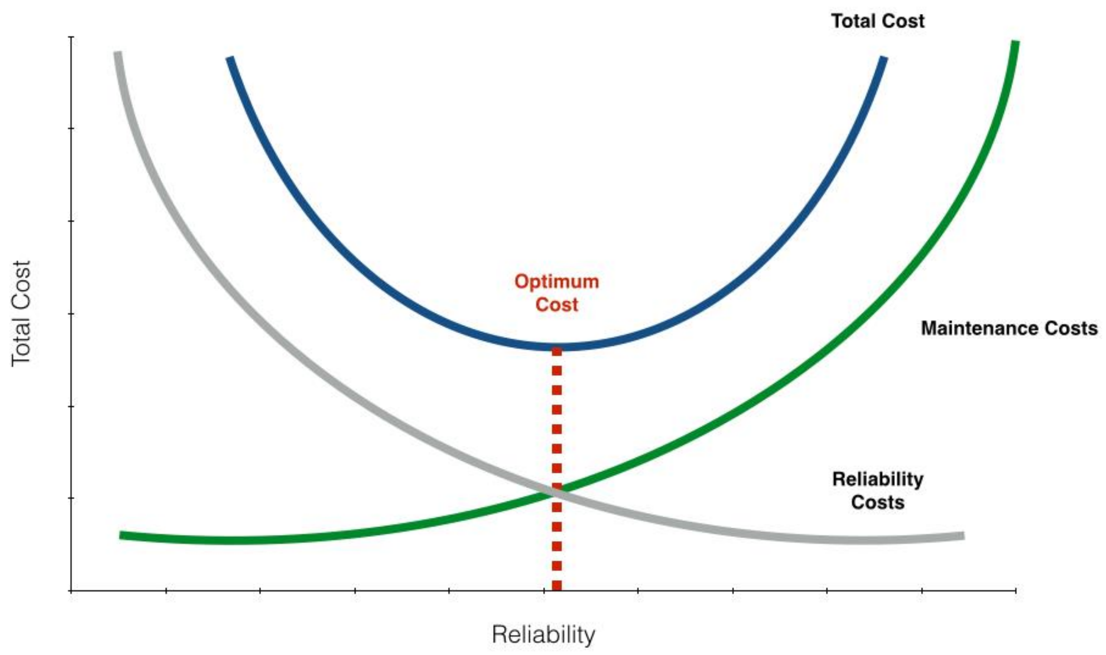


Figure 2. Total cost as a function of a reliability (O'Brien 2014)

Reliability costs relates to costs that occur due to unreliable systems. Poorly maintained machines will likely produce poor quality or defective products. Throughput is likely to be affected due to increase in cycle time and unplanned machine downtime. That in turn could mean lost important orders. Poor quality and missed orders will negatively effect on customer satisfaction what could lead to lost customers. Unreliable systems can additionally cause costs related to negative environmental impact or even occupational health and safety. (O'Brien 2014)

Maintenance costs are any costs which relate to machine maintenance. That includes maintenance work hours, direct cost of spare parts, cost of maintenance tools and cost of holding spare parts in the inventory. (O'Brien 2014)

The goal is to find the optimum reliability. It is not always necessary for the asset to have very high availability at the very high maintenance expense. Organization must find what is the optimal for them so that money are not wasted on reaching ill-stated goals. (O'Brien 2014)

One natural solution is to aim at doing maintenance more effectively. Doing so shifts the maintenance curve to the right. This shift also moves the optimum reliability point. More effective maintenance can be achieved e.g. by switching from reactive or proactive maintenances to the predictive one. The figure below shows how optimum is affected by the maintenance curve shift. (O'Brien 2014)

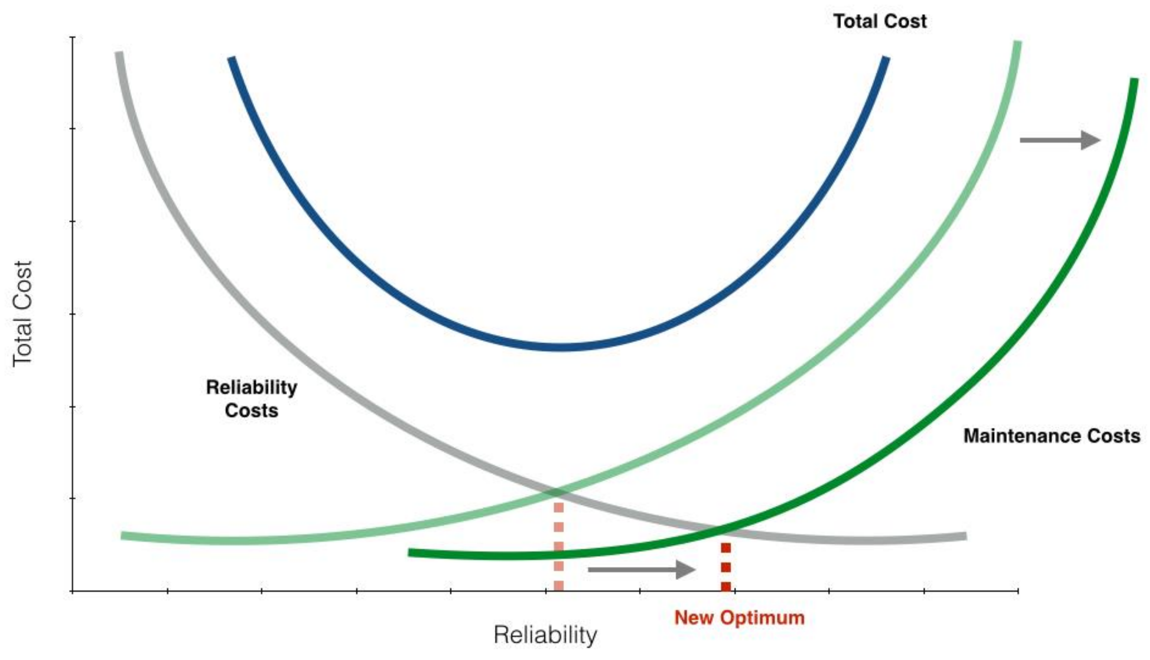


Figure 3. Maintenance Curve Shift (O'Brien 2014)

Reliability can be defined using following (O'Connor & Kleyner 2012):

- Failure Rate – the mean number of failures in a given time
- MTBF – mean time between failures, for repairable items
- MTTF – mean time to failure, for non-repairable items

Our main concern would be asset availability, which is affected by failure rate and by maintenance time. From the equation below we can see the relation between reliability expressed by the mean time between failure (MTBF) and maintainability given by the mean time to repair (MTTR). In order to increase availability of the asset one should improve either MTBF or MTTR. (O'Connor & Kleyner 2012)

$$Availability = \frac{MTBF}{MTBF + MTTR} \tag{1}$$

Predictive maintenance should have positive impact on both measures, MTBF and MTTR. With its predictive power it should eliminate unnecessary maintenance work while not allowing for errors to happened. Therefore, increasing MTBF. Additionally it provides insights and allow to plan better so that maintenance work can be done faster. That in turn means reduced MTTR. We can therefore conclude that correctly implemented predictive maintenance increases asset availability.

Unsurprisingly the high availability is expensive. Availability is directly related to reliability (MTBF). Therefore, as we know from previous discussion, optimum availability is less than 100% when total cost point of view is considered. (O'Connor & Kleyner 2012)

Contradictory relation between reliability and total cost is shown on the figure below. It based on Deming manufacturing teaching. According to him costs of preventing or correcting causes are lower than doing nothing. Therefore, according to him, total cost continues to decrease as quality/reliability is reaching perfection. His teaching are base of kaizen (continuous improvement) and founded post-war quality revolution in Japan. (O'Connor & Kleyner 2012)

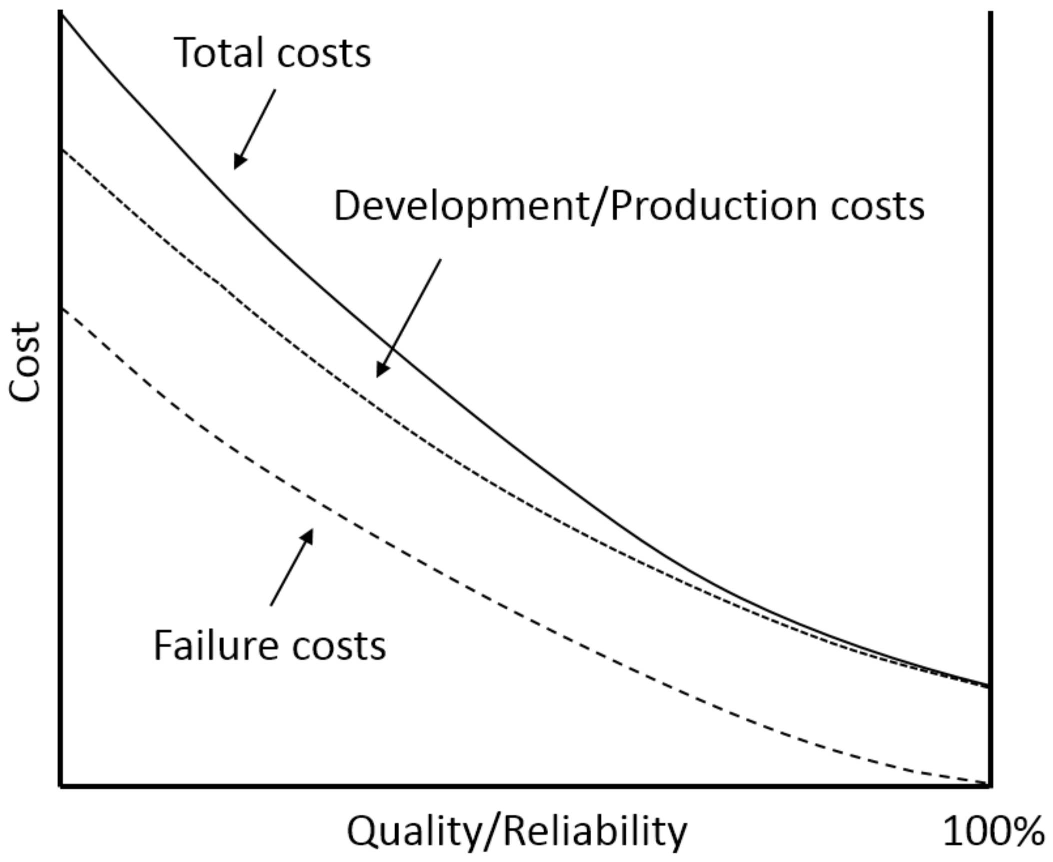


Figure 4. Life cycle cost as a function of the quality based on Deming's quality vs. cost model (O'Connor & Kleyner 2012)

In practice, Deming argumentation is hard to sell. Possibly reaching for perfection can bring benefits in the long run, but cost are occurring now and there is always time and money limitation. Research on reliability modelling by Kleyner (2010) concluded that total cost curve is highly skewed to the right, **Figure 5**. According to his research further reliability improvements needs to be done at increasing costs while returns are diminishing. (O'Connor & Kleyner 2012)

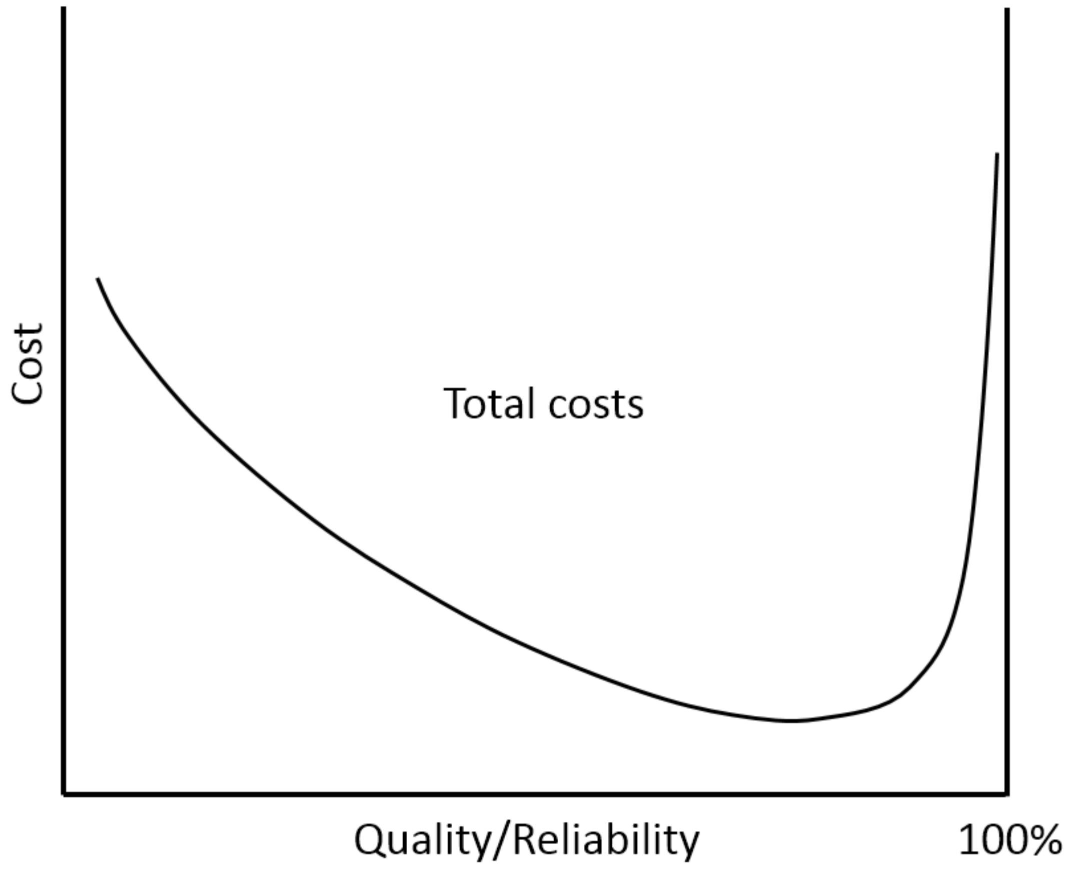


Figure 5. Life cycle cost as a function of the quality in practical applications (O'Connor & Kleyner 2012)

2.3.CRISP-DM

The CRISP-DM (Cross-Industry Standard Process for Data Mining) was used during the thesis. It is non-proprietary, neutral and freely available data mining model. It is composes from six phases: business understanding, data understanding, data preparation, modelling, evaluation and deployment. The purpose of the model is to provide industry standard that brings better understanding of the data mining process for different stakeholders involved into the project. Clear road map helps to structure otherwise unstructured data mining process which is full of exploratory approach. (Shearer 2000)

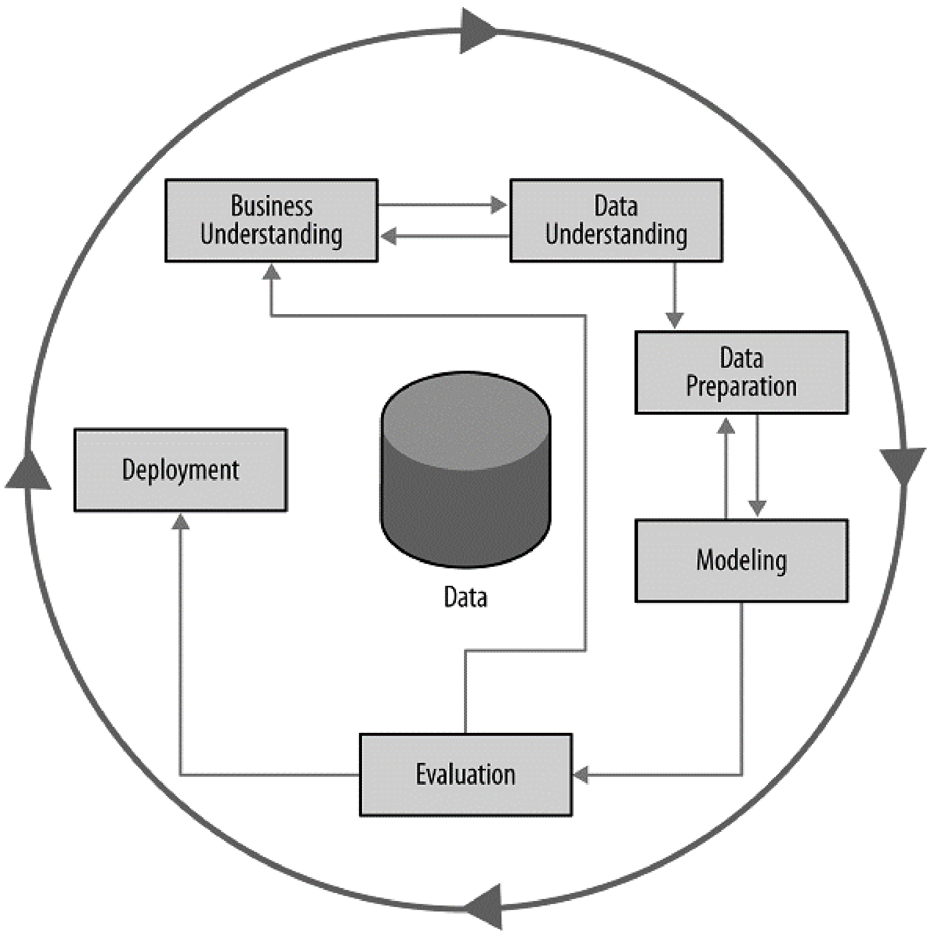
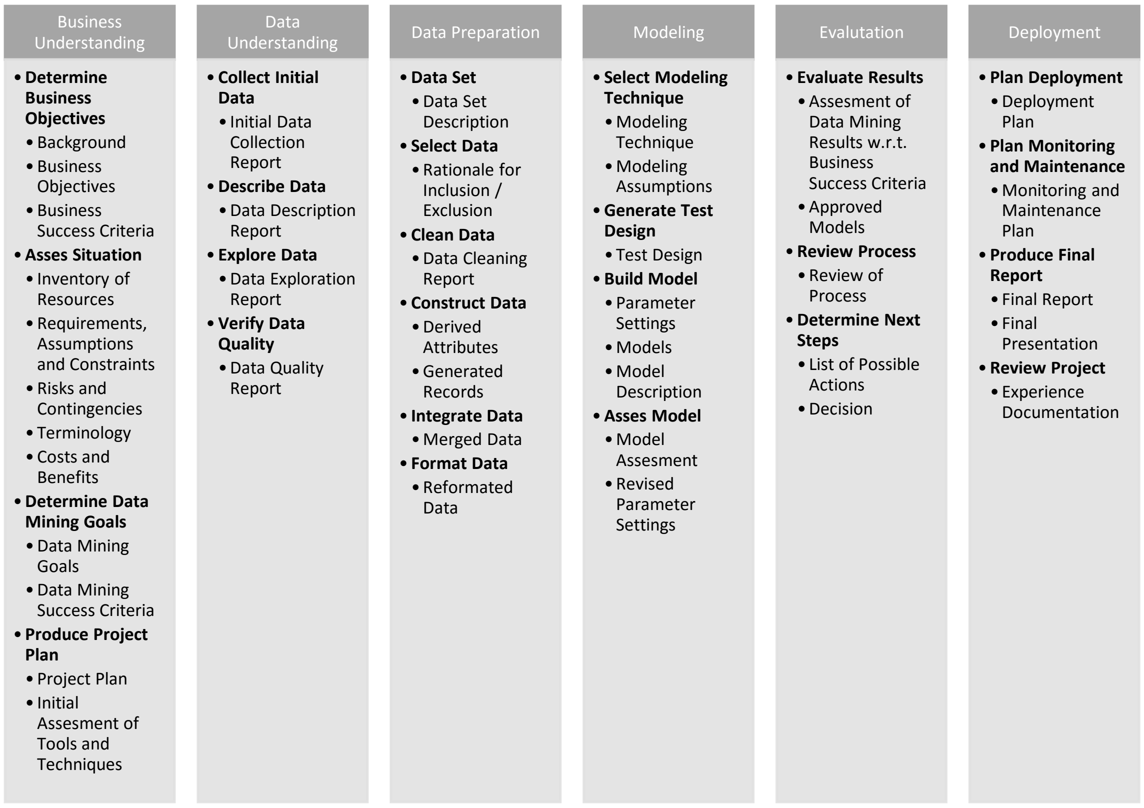


Figure 6. Phases of the CRISP-DM Reference Model (Provost & Fawcett 2013)



Business Understanding
- **Determine Business Objectives**
  - Background
  - Business Objectives
  - Business Success Criteria
- **Asses Situation**
  - Inventory of Resources
  - Requirements, Assumptions and Constraints
  - Risks and Contingencies
  - Terminology
  - Costs and Benefits
- **Determine Data Mining Goals**
  - Data Mining Goals
  - Data Mining Success Criteria
- **Produce Project Plan**
  - Project Plan
  - Initial Assesment of Tools and Techniques

Data Understanding
- **Collect Initial Data**
  - Initial Data Collection Report
- **Describe Data**
  - Data Description Report
- **Explore Data**
  - Data Exploration Report
- **Verify Data Quality**
  - Data Quality Report

Data Preparation
- **Data Set**
  - Data Set Description
- **Select Data**
  - Rationale for Inclusion / Exclusion
- **Clean Data**
  - Data Cleaning Report
- **Construct Data**
  - Derived Attributes
  - Generated Records
- **Integrate Data**
  - Merged Data
- **Format Data**
  - Reformated Data

Modeling
- **Select Modeling Technique**
  - Modeling Technique
  - Modeling Assumptions
- **Generate Test Design**
  - Test Design
- **Build Model**
  - Parameter Settings
  - Models
  - Model Description
- **Asses Model**
  - Model Assesment
  - Revised Parameter Settings

Evalutation
- **Evaluate Results**
  - Assesment of Data Mining Results w.r.t. Business Success Criteria
  - Approved Models
- **Review Process**
  - Review of Process
- **Determine Next Steps**
  - List of Possible Actions
  - Decision

Deployment
- **Plan Deployment**
  - Deployment Plan
- **Plan Monitoring and Maintenance**
  - Monitoring and Maintenance Plan
- **Produce Final Report**
  - Final Report
  - Final Presentation
- **Review Project**
  - Experience Documentation

Figure 7. Tasks and Outputs of the CRISP-DM Reference Model (Shearer 2000)

2.3.1. Business Understanding

Crucial step for any data mining project to succeed. It is important to understand the problem from the business perspective and then define it as a data mining problem. It then follow with preliminary project plan. Business understanding is further decompose into determining business objectives, assessing the situation, determining the data mining goals and producing the project plan. (Shearer 2000)

2.3.1.1. Determine the Business Objectives

Sometimes customers may not know or really understand what they want to achieve. Therefore, understanding true business problem to be solved is so crucial. Failing at that phase may result with the solution for the wrong problem. It could be paraphrased with the famous quota:

> *"An approximate answer to the right question is worth a great deal more than a precise answer to the wrong question"* John Tukey

Also at that moment measurable success goal(s) should be set. It should be achievable and related to the business objective(s). (Shearer 2000)

2.3.1.2. Assess the Situation

All project related resources are defined with special emphasize on data available. Additionally any assumptions made should be listed. Risks are identified, prioritized and actions are planned based on it. At the end cost-benefit analysis of the undertaken project is done. (Shearer 2000)

2.3.1.3. Determine Data Mining Goals

The data mining goals are stated from business perspective. If those goals cannot be easily translated into data mining ones then it should indicate that problem is maybe not well defined and may require reconsideration. (Shearer 2000)

2.3.1.4. Produce Project Plan

Finally at this last task of the first step project plan is created. It includes details on how data mining goals are planned to be achieved, also with the respect to the timeline. Identified risks are listed along with actions planned, to emphasize on probability of positive risks and to reduce probability or impact of negative ones. Likewise potential tools and techniques intended to address issues of the project should be listed here. The rule of thumb generally accepted in the industry expects that (Shearer 2000):

- Data Preparation Phase takes lion share of the time, between 50 to 70 percent of time allocated to the whole project.
- Data Understanding Phase takes between 20 to 30 percent of the time
- Modelling, Evaluation and Business Understanding Phases take in the range 10 to 20 percent
- Deployment Planning Phase is expected to take the smallest share of just 5 to 10 percent

2.3.2. Data Understanding

The main focus of this phase is to retrieve data available and to asses on its quality. Following subtasks are executed: collection of initial data, description of the data, exploration of the data and verification of the data quality. Each of those tasks is described bit more below. (Shearer 2000)

2.3.2.1. Collect Initial Data

Data is possibly collected from many sources. Process should be documented to ease replication in the future if needed. Meaning any issues encountered and related solutions should be written down. (Shearer 2000)

2.3.2.2. Describe Data

In the course of this task basic characteristics of the collected data are described. Basic properties of the data such as the format, quantity of the data, the identities of the fields, etc. are reported. The main issue to be addressed is if collected data satisfy requirements. (Shearer 2000)

2.3.2.3. Explore Data

This step builds on the previous one. Using exploratory approach data scientist should use querying, visualizations and reporting to uncover insights of the data at hand. Data exploration report is created as the outcome of this task. This report should contains details on all findings with its possible impact on the rest of the project. Initial hypothesis can be also drawn based on the findings. (Shearer 2000)

2.3.2.4. Verify Data Quality

Quality of the data is examined. Most commonly it means checking on missing values, verifying that all possible values are represented sufficiently, checking for outliers which may indicate for erroneous data but not necessary, misspellings, or looking for values that don't make sense, e.g. person height 2000 meters or age -10. (Shearer 2000)

2.3.3. Data Preparation

This is last phase at which main focus is with data. At this point final data which will serve as an input to the modelling is created based on raw data gathered. Activities of this phase include (Shearer 2000):

- Table Selection
- Record Selection
- Attribute Selection
- Transformation
- Cleaning

The sub task of this phase are data selection, data cleaning, data construction data integration and the data formatting. (Shearer 2000)

2.3.3.1. Select Data

Selection of data is done based on constraints, quality and relevance of the data to the project. As the part of the process the reasoning for inclusion and exclusion should be documented. Usually it also brings good results to reduce number of attributes and remove ones which are at some level duplicates. We may want as well reduce the level of detail if it is not relevant for our project. E.g. we may be interested to have post code but street address may be unnecessary detail for our problem. Of course all depends on the project's goals and requirements. (Shearer 2000)

2.3.3.2. Clean Data

Model need to be provided with the clean data in order to produce meaningful results. The known concept of Garbage In – Garbage Out applies very well to the data modelling. The quality of the model output is much dependent on the quality of the data at its input. Therefore at this stage all issues reported during "Verify Data Quality" step need to be addressed. Simple solution may be to drop dirty entries, e.g. ones with missing value for some of the attribute. However, it may result in modelling being performed on very small part of the original data available. It most likely will not produce best result possible. Alternative is to apply more sophisticated approach to the problem. E.g. to replace missing data with some computed value as average or median. (Shearer 2000)

2.3.3.3. Construct Data

At this stage of data preparation, derived attributes or even whole new records are created. Derived attributes are the ones created based on existing ones. It could be simple single-attribute transformation, e.g. to transform values in Fahrenheit to Celsius or age to some age group. It could be as well more complex mathematical calculation based on several other attributes or data query of some kind. (Shearer 2000)

2.3.3.4. Integrate Data

Data integration in case of tabular data means different kind of joins operations on two or more tables. Usually it means gathering pieces of information regarding same item from different tables into one. It also include aggregation, which simply refers to creation of new values for entries by the mean of summary of some kind. It can be in the form of total sum, average, median, etc. (Shearer 2000)

2.3.3.5. Format Data

Sometimes the change of the data format may be required. It could be dictated by the specific modelling tool. E.g. need to remove illegal characters or to trim text fields to maximum length. Sometimes it may involve more severe restructuring of the information. (Shearer 2000)

2.3.4. Modelling

In this phase, data mining algorithm is chosen. It is then used with data available and over several iterations optimal algorithm parameters are determined. Usually given data mining issue can be solved using number of algorithms and it is hard to determine which one will perform better. Therefore, it is common to try few of them and do selection based on performance and possibly other factors as e.g. interpretability. Some algorithms may have specific requirement regarding the input data. Consequently, stepping back to the "Data Preparation" phase is not unusual. Activities of this phase include (Shearer 2000):

- Selection of the Modelling Technique
- Test Design Generation
- Model Building
- Model Assessment

2.3.4.1. Select Modelling Technique

One or more modelling algorithm is chosen. It is often hard to say which one of the possible candidates is the best. Therefore, usual case is to verify few of them. Also it is common to prefer simple models over complicated ones, as those are easier to understand and usually generalize better. Vast amount of algorithms exist, figure below lists some of them to give better grasp on complexity related to choosing the best one for the given project. (Shearer 2000)

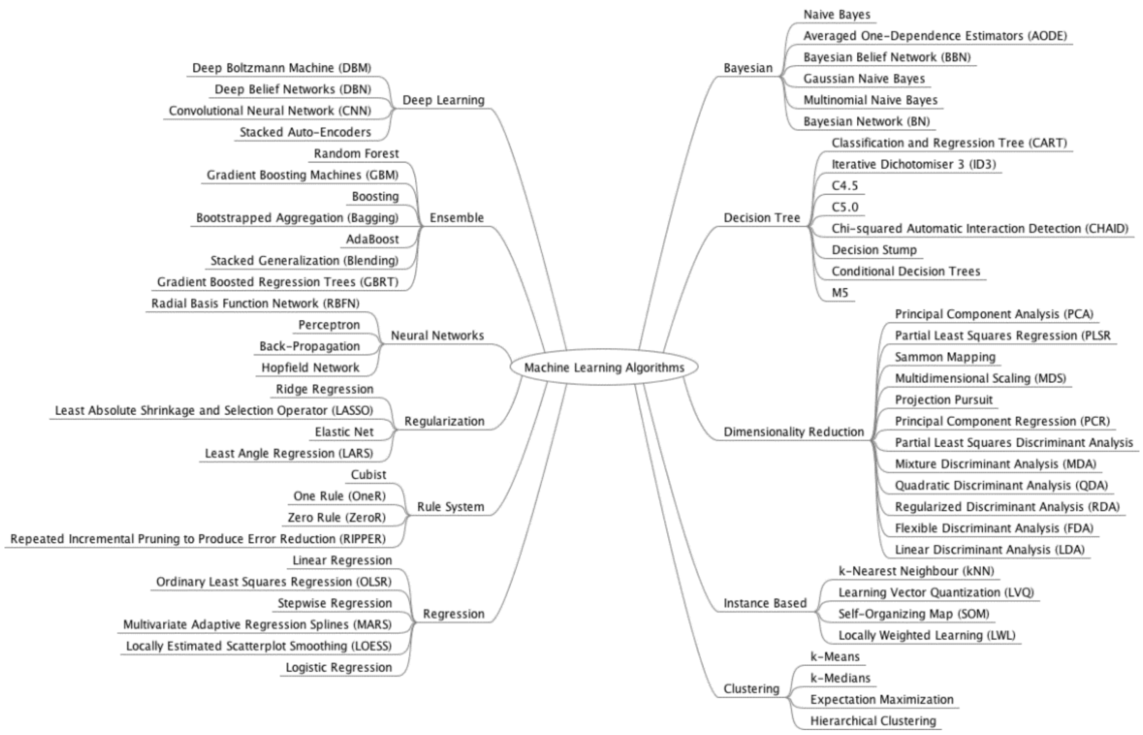

Machine Learning Algorithms
Deep Learning
Deep Boltzmann Machine (DBM)
Deep Belief Networks (DBN)
Convolutional Neural Network (CNN)
Stacked Auto-Encoders
Ensemble
Random Forest
Gradient Boosting Machines (GBM)
Boosting
Bootstrapped Aggregation (Bagging)
AdaBoost
Stacked Generalization (Blending)
Gradient Boosted Regression Trees (GBRT)
Neural Networks
Radial Basis Function Network (RBFN)
Perceptron
Back-Propagation
Hopfield Network
Regularization
Ridge Regression
Least Absolute Shrinkage and Selection Operator (LASSO)
Elastic Net
Least Angle Regression (LARS)
Rule System
Cubist
One Rule (OneR)
Zero Rule (ZeroR)
Repeated Incremental Pruning to Produce Error Reduction (RIPPER)
Regression
Linear Regression
Ordinary Least Squares Regression (OLSR)
Stepwise Regression
Multivariate Adaptive Regression Splines (MARS)
Locally Estimated Scatterplot Smoothing (LOESS)
Logistic Regression
Bayesian
Naive Bayes
Averaged One-Dependence Estimators (AODE)
Bayesian Belief Network (BBN)
Gaussian Naive Bayes
Multinomial Naive Bayes
Bayesian Network (BN)
Decision Tree
Classification and Regression Tree (CART)
Iterative Dichotomiser 3 (ID3)
C4.5
C5.0
Chi-squared Automatic Interaction Detection (CHAID)
Decision Stump
Conditional Decision Trees
M5
Dimensionality Reduction
Principal Component Analysis (PCA)
Partial Least Squares Regression (PLSR
Sammon Mapping
Multidimensional Scaling (MDS)
Projection Pursuit
Principal Component Regression (PCR)
Partial Least Squares Discriminant Analysis
Mixture Discriminant Analysis (MDA)
Quadratic Discriminant Analysis (QDA)
Regularized Discriminant Analysis (RDA)
Flexible Discriminant Analysis (FDA)
Linear Discriminant Analysis (LDA)
Instance Based
k-Nearest Neighbour (kNN)
Learning Vector Quantization (LVQ)
Self-Organizing Map (SOM)
Locally Weighted Learning (LWL)
Clustering
k-Means
k-Medians
Expectation Maximization
Hierarchical Clustering

Figure 8. Machine Learning Algorithms (Brownlee 2013)

2.3.4.2. Generate Test Design

Testing plays crucial role and need to be designed to verify how model perform and if it generalize well enough. Predictions done by model should be more accurate than those done by poor chance. It should also generalize the problem, so that it perform as well on unseen data as well as on historical data that was used for learning. (Shearer 2000)

There are various approaches to test design. The least complex one is to partition data into two, part for learning and part for testing. We refer to that technique simply as data split. However, especially when dataset is not very large other more advanced methods are preferred. Listing few most popular (Brownlee 2014):

- Bootstrap
- k-fold Cross Validation
- Repeated k-fold Cross Validation
- Leave One Out Cross Validation

2.3.4.3. Build Model

After test design phase, the part of the data that is meant for learning is used to build the model by the selected set of machine learning algorithms. (Shearer 2000)

2.3.4.4. Asses Model

Model or rather models are assessed based on domain knowledge and success criteria established earlier. It should be done from technical perspective as well as in the business context, usually with the help of the business analyst and domain experts. This is preliminary assessing as more thorough will follow. Focus is on accuracy and generality of the models. (Shearer 2000)

2.3.5. Evaluation

Even so models are already assessed in the previous step, it is vital to do it more exhaustively before final deployment. Model is tested to assure that business objectives are achieved and that all key business issues are reflected. (Shearer 2000)

2.3.5.1. Evaluate Results

As stated earlier, this is more deep evaluation than what was already done during the Modelling phase. This is final evaluation which should give an answer to the question if model is ready to be deployed. Focus is on business aspects and model is checked in order to determine if there are ones not addressed correctly or against it. If time and budget allow then model is tested on real data. Beside verification of the model feasibility for the deployment, evaluation seeks to unveil possible improvement suggestions for the next iteration of the CRISP-DM cycle. (Shearer 2000)

2.3.5.2. Review Process

In this step, review of the whole data mining process is done in order to verify that nothing important was forgotten or overlooked. It serves also as the quality assurance stage. (Shearer 2000)

2.3.5.3. Determine Next Steps

The decision point for the project leader with following possibilities (Shearer 2000):

- Move to deployment
- Initiate further iteration
- Start new data mining project
- Cancel the project – obviously something went wrong if it went that far to be cancelled.

2.3.6. Deployment

Model built does not benefit the organization much until it is deployed. Deployment usually means that model is somehow integrated into decision making process. It could make autonomous decisions or provide supportive information for decisions made by human. Deployment can be simple or more complex. At its simplest form it would be in the form of the report summarizing the findings, e.g. simple decision tree printed on paper. In the more complex form it would be the IT system taking decisions autonomously, e.g. recommendations done by Netflix or Amazon. (Shearer 2000)

2.3.6.1. Plan Deployment

In order to have smooth deployment it needs to be planned well. During this phase deployment strategy is created and documented. (Shearer 2000)

2.3.6.2. Plan Monitoring and Maintenance

However well tested before deployment it is crucial to plan and later execute monitoring and maintenance of the model. Likely new insights to the business problem which is addressed by the model will come once it is deployed. Also business environment usually changes over the time. Those and other issues require for model to be monitored and maintained in order to assure its correct usage over its life time. (Shearer 2000)

2.3.6.3. Produce Final Report

Final report is created at the end of the data mining project by the project leader and the data mining team. It content depends a bit on the deployment planned. It could be in the form of short summary or it could be a comprehensive document presenting data mining results. All previous deliverable are included into final report. Usually that phase ends with the customer meeting where results are presented and discussed. (Shearer 2000)

2.3.6.4. Review Project

The project leader should evaluate and document any failures and successes encountered during the project. Focus of this activity is to improve future projects so that same pitfalls will not reoccur. Lessons learned during this project should help with next ones, and it should be seen as an additional value added of this project. (Shearer 2000)

2.3.7. CRISP-DM vs. SEMMA vs. KDD

It was decided to use CRISP-DM methodology for the empirical part of the thesis. However, other methodologies exists and aim of this chapter is to give short comparison between CRISP-DM, SEMMA and KDD.

2.3.7.1. KDD

KDD (Knowledge Discovery in Databases) process states data mining as a one of its phases. It originates from 1989 and as such was created in a bit different context than newer models. Nevertheless, it can still be used nowadays with a bit of adaptation in some cases. KDD is the process of knowledge extraction from the database. (Azevedo et al. 2008)

KDD consist of five stages listed below and depicted on the figure (Fayyad et al. 1996):

- Selection – creating subset of the original data on which discovery will be executed.
- Pre-processing – getting data into shape for data mining algorithms to be run on. E.g. handling of missing data, removing noise.

- Transformation – reducing number of variables (dimensionality reduction) and/or transforming them.
- Data Mining – searching for patterns of interest based on project's objectives.
- Interpretation/Evaluation – interpretation and evaluation of the results produced during the data mining stage.

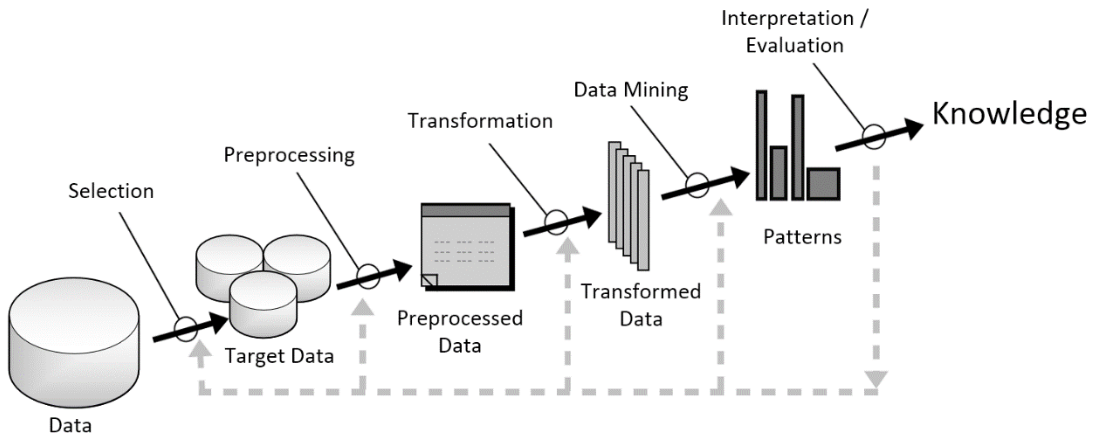


Figure 9. Steps of the KDD Process (Fayyad et al. 1996)

It is assumed that one has developed sufficient domain knowledge and good understanding of the customer needs before any of the before mentioned KDD's activities starts. Once knowledge is discovered it is also assumed that one will act based on it by incorporating it into decision making system or system of some other kind. (Fayyad et al. 1996)

2.3.7.2. SEMMA

The SEMMA is yet another methodology for directing a data mining project. It was developed by the SAS Institute. SEMMA acronym stands for Sample, Explore, Modify, Model and Asses. (Azevedo et al. 2008)

Phases of the SEMMA are listed and shortly described below (Azevedo et al. 2008):

- Sample – extract the portion of the data from the larger set. Standard purpose of the sampling is to retain information from the population inside the sample but at the same time make it smaller and more manageable to work with.
- Explore – exploring data in various way in order to gain better understanding of the data at hand.
- Modify – modify data based on domain knowledge and according to needs of data mining algorithms to be used.
- Model – run selected data mining algorithms on the data provided in order to find patterns which helps in desired outcome prediction.
- Asses – assessing of the modelling results based on its usefulness and reliability.

2.3.7.3. Comparison of methodologies

Similarities can be noticed between all three methods. It is very easy to link corresponding stages of KDD and SEMMA. It may seem that CRISP-DM covers bit more. It is true when comparing it with SEMMA. However, if we take into consideration pre and post stages of the KDD it can be noticed that those are matching to business understanding and deployment stages of the CRISP-DM methodology. It is not by surprise that SEMMA is missing those two stages when compared to remaining methodologies. It originated at SAS as a logical organization of the SAS Enterprise Miner toolset (Dean 2014). Table below summarize comparison between KDD, SEMMA and CRISP-DM methodologies. (Azevedo et al. 2008)

Table 1. Summary of the correspondences between KDD, SEMMA and CRISP-DM (Azevedo et al. 2008)

KDD	**SEMMA**	**CRISP-DM**
Pre KDD		Business understanding
Selection	Sample	Data understanding
Pre processing	Explore	
Transformation	Modify	Data preparation
Data mining	Model	Modelling
Interpretation / Evaluation	Assessment	Evaluation
Post KDD		Deployment

2.4.Machine Learning Topics

Following presents machine learning topics needed to understand subsequent chapters. Especially when differences between various machine learning algorithms are discussed. The purpose is to give short introduction to just few selected topics. Curious reader may want to find more information from other literature.

2.4.1. Number of hyperparameters

Hyperparameters are the algorithm parameters which are set prior to training phase. Those parameters of the machine learning algorithm allow to tune it to specific data and business problem. Greater number of parameters available for a given algorithm means it can be more adjusted and therefore should be capable of achieving better results. However, more parameters also means more time needed to find the sweet spot. The process of parameters fine tuning can be automated but it still going to take a time as the training time increases exponentially with the number of parameters to be adjusted. (Rohrer 2016)

Sweep Parameters is used to find optimum set of parameters to be used for training of the model. Those cannot be determined in advance as they depend on prediction task and data used. Beside basic approach (integrated train and sweep), it support also more advance cross validation mode. In that mode data is divided into number of folds and parameter sweep is executed for each of them. It usually produces better results but it is also more time consuming. (Microsoft 2015b)

2.4.2. Imbalanced Class Distribution

In the case of many predictive applications it is common for the class of interest to be in the significant minority as compared to the whole population. It is known as a class imbalance problem. Even so distribution is unbalanced it reflects the true class distribution. It is the case with the predictive maintenance as one would expect machines failures to occur unfrequently. Additionally, the whole purpose of the maintenance, including the predictive maintenance, is to reduce number of those fault events. Of course it has positive impact of the factory operation but at the same time it makes it harder to collect valuable data as no one wish to run-to-failure for the in-service asset. (Microsoft 2015a)

Other application that suffer from the same problem; just to give as an example; is in healthcare in detecting disease. Usually, probability of the given disease in the population is very small. However, consequences of not detecting one are as high as patient death. With so low probability of occurring, the model which will always give negative test result would have very high accuracy. It would never detect any disease, but in case of disease that occurs for 1 person out of 10’000 it would still be 99.99% accurate.

Similarly with the predictive maintenance and machines. Very simple model, which gives “no fault” prediction would have very high accuracy. Significantly better than a random guess. (Drummond & Holte 2000)

Traditional cost-insensitive classifiers would made following two assumptions (Provost 2000):

- The test dataset’s class distribution is same as of the training one
- The classifier’s objective is to maximize the accuracy

Class imbalance problem become meaningful when there are different costs associated with the different types of errors. In that case, usually it is more expensive to misclassify representative of the minority class as belonging to the major class than other way around. If we assume minority class as “positive”, then we can write that cost of false negative is greater than false positive, FNcost > FPcost. (Ling & Sheng 2011)

Following two are common solutions to the class imbalance problem (Ling & Sheng 2011):

- Cost sensitive learning – it aims to minimize the total cost while assigning different costs for false negatives and false positives classifications.
- Oversampling the minority class and/or undersampling the majority class in order to reduce degree of imbalance.

With the Microsoft Azure ML problem is addressed either by undersampling the majority class using custom R script or oversampling with the SMOTE module. (Microsoft 2015a)

The SMOTE module allows to use Synthetic Minority Oversampling Technique to increase number of samples and to even proportions between majority and minority classes. Using this technique increases number of rare cases in the manner better than simple duplication. It uses features of nearest neighbours combined with the target class to generate new instances. Module has two parameters. "SMOTE percentage" let to provide desired percentage increase of the minority class in a multiply of 100. "Number of nearest neighbours" parameter defines number of nearest neighbours which are taken into consideration while creating a new instance. (Microsoft 2015b)

2.4.3. Bayesian Statistics

Bayesian statistics is often portrayed as an alternative to the classical frequentist approach. Bayesian provides prior distribution which is argued by some to violate objective view point. However, it is also the reason for its superiority in some cases. In summary one may want to use Bayesian statistics when it is intended to combine domain knowledge from experts with the knowledge discovery. (Lin 2013)

2.4.4. Ensemble

Ensemble can be compared to the board of experts making a decision. Each expert can vote, but not necessary with the same voting power. Similarly with ensemble methods in machine learning, many classifiers are created and prediction is given as a weighted vote of theirs predictions. The aim is to achieve better predictive power from the group of classifier than what could be achieved with any single of them.

2.4.4.1. Boosting

Boosting is iterative, meaning that previous models performance affects the creation of the new ones by specifically resampling the dataset. It does so by enforcing new model to focus on instances which were misclassified by previous models. Model confidence for the particular prediction instance, which base on past performance, effects on its vote weight in the final voting. (Han et al. 2011)

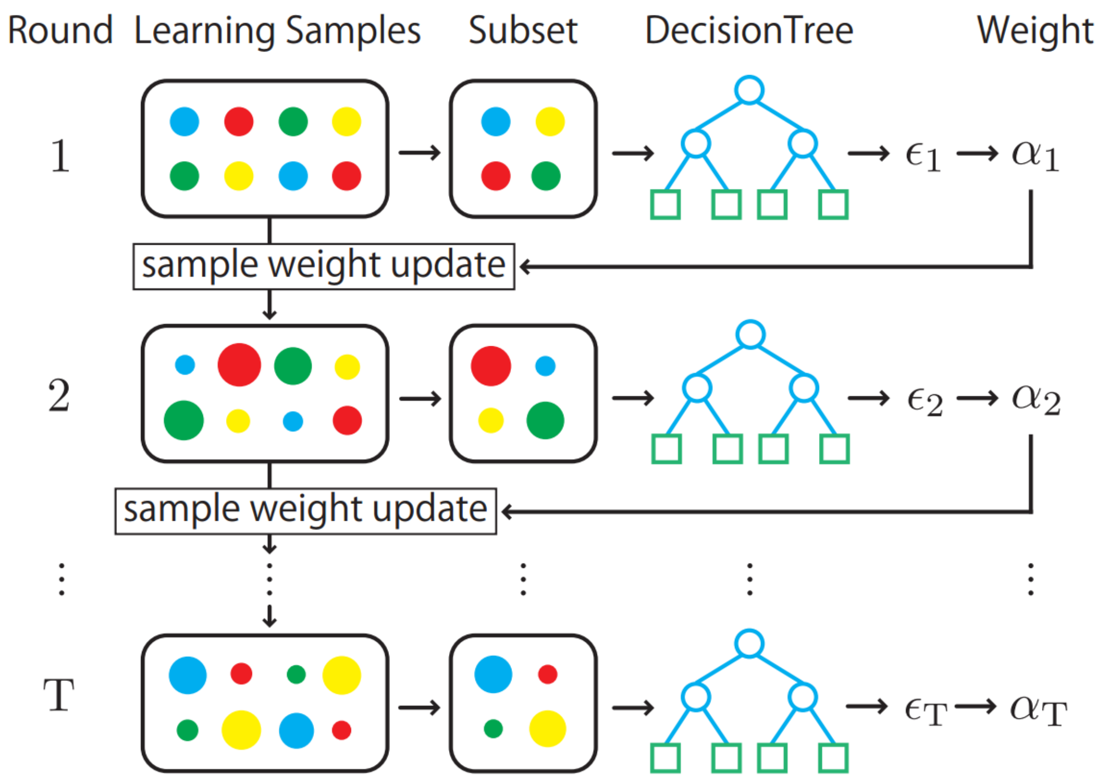

Round
Learning Samples
Subset
DecisionTree
Weight
1
2
T
sample weight update
sample weight update
$\epsilon_1 \rightarrow \alpha_1$
$\epsilon_2 \rightarrow \alpha_2$
$\epsilon_T \rightarrow \alpha_T$

Figure 10. Learning Process – Boosting (Mishina et al. 2014)

2.4.4.2. Decision Forest

Decision forest is constituted from many decision trees. Each tree differ from each other as split attributes are selected randomly. This difference plays important role. Intuitively it makes only sense to consult different models if those are diverse from each other. The idea is that in that case each model will be specialist in some part of the data and one model will complement weaknesses of others. Final decision is made in the form of voting. Trees with the higher prediction confidence get allocated higher weight to theirs votes. Aggregating decisions in that way gives final decision out of the decision tree. Decision forests can handle fairly well outliers and are good in generalization as long as there are enough trees in the forest. (Han et al. 2011)

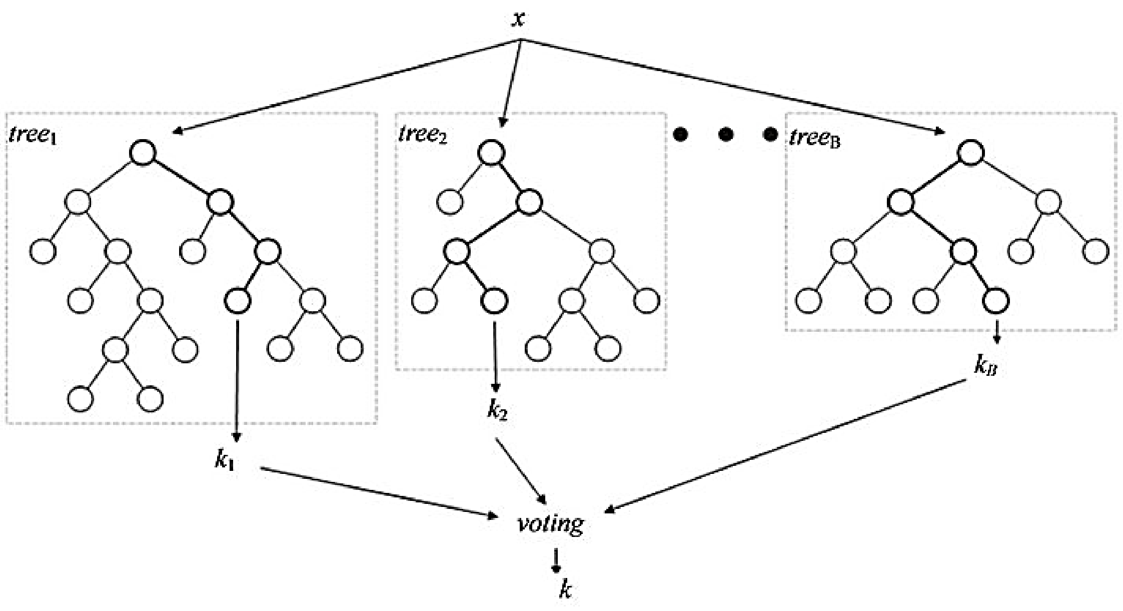


Figure 11. Decision Forest (Nguyen et al. 2013)

2.4.4.3. Decision Jungle

Decision Jungle addresses high memory consumption shortcoming of the decision trees and forests. Number of decision tree nodes grows exponentially with the depth. Therefore, some systems may not have enough resource to grow tree big enough in effort to provide best accuracy possible. Especially in case of embedded systems it may be required to artificially limit the depth. Decision jungle is an ensemble of rooted decision directed acyclic graphs (DAGs). Unlike in case of trees, DAGs allow for multiple paths to each leaf. Building DAGs takes bit more of the training time at the benefit of significantly smaller memory footprint and improved model generalization capability. (Shotton et al. 2013)

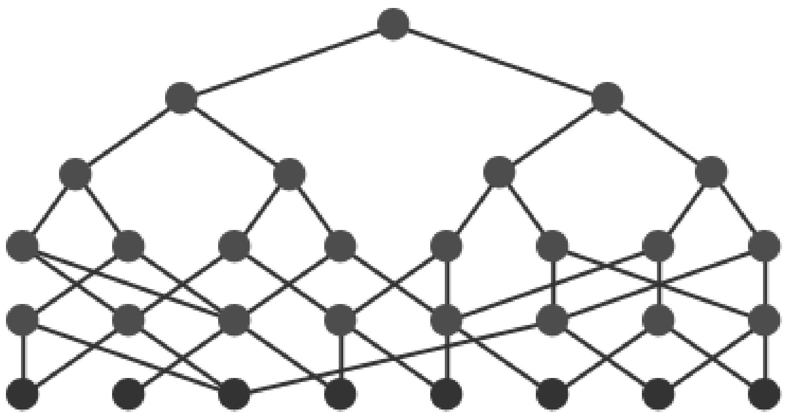


Figure 12. Decision Jungle (Pohlen 2015)

2.4.5. Artificial Neural Network

Artificial Neural Networks (ANNs) concept comes out of the inspiration with the biological neural networks of the brain. Brain uses neurons which are interconnected to solve complex problems. In the similar manner, ANN uses artificial neurons and creates connections between them in order to model relationship between input signals and an output signal. This network of interconnected neurons is the solution to the learning problem. Limitation comes in the number of neurons, with biological brains far exceeding possibilities of current state of the art ANN. Typical ANN is constituted out of few hundreds of neurons. Hard to compare to human brain which is made from approximately 85 billion neurons. There are number of practical applications of the ANNs. However, one its major problem is lack of interpretability of models created by the ANNs. Those models function as black boxes and do not provide insights into how the problem is solved. Therefore, making it virtually impossible for the human experts to evaluate. (Lantz 2015)

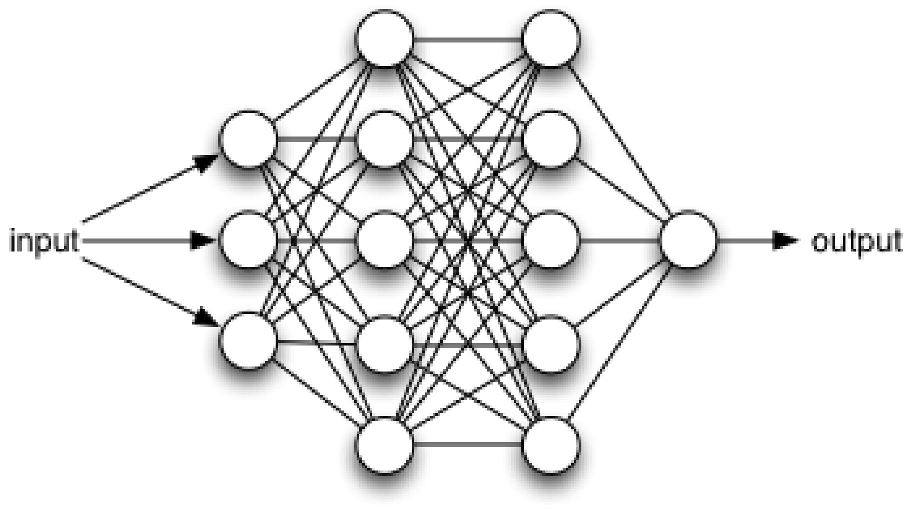


Figure 13. Artificial Neural Network (Dolhansky 2013)

2.4.6. Support Vector Machine

Support Vector Machine (SVM) uses hyperplane boundary with the goal to create homogenous partitions and maximum margin between partitions possible. Maximum Margin Hyperplane (MMH) is the one which creates highest separation amongst classes and likely generalizes best. Support vectors are the points from each class which are closest to the MMH. Each class needs to have at least one support vector and it is possible to have many of them, if all are in the same distance. The advantage of this algorithm lays in those support vectors. They allow for model to be compact as support vectors on itself are enough to define the MMH. (Lantz 2015)

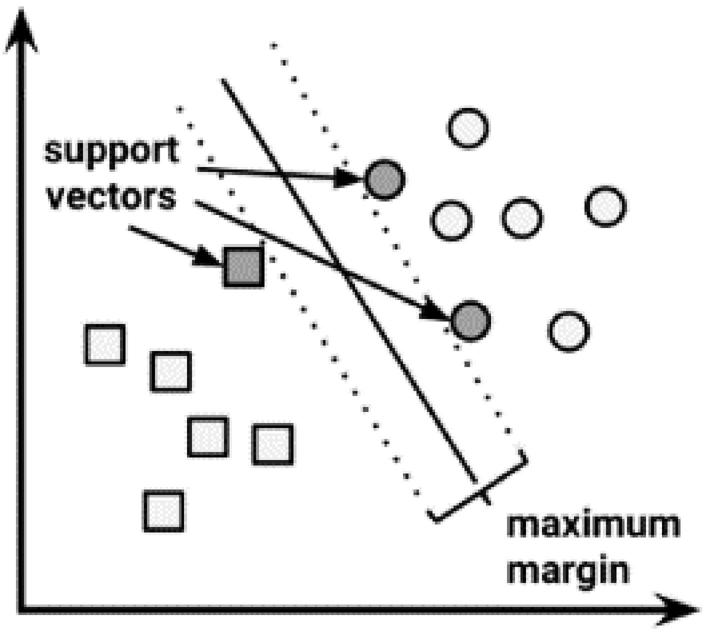


Figure 14. Support Vector Machine (Lantz 2015)

2.5. Evaluating Model Performance

It is beyond the scope of this document to provide very comprehensive description of the following concepts. Aim is support the reader with the basic information required while evaluating the results of machine learning algorithms provided later on.

First we will introduce basic concepts used in the evaluation of the machine learning projects. It is important to understand meaning of those concepts and how do they interrelate to each other. Following subchapters aim to explain basics of the following terms:

- Classification
    - Accuracy
    - Confusion Matrix

    - Unbalanced Class
    - Unequal Costs and Benefits
    - Expected Value
- Regression
    - Root Mean Squared Error (RMSE)
    - Coefficient of Determination – R^2
- Both
    - Overfitting
    - Training Time
    - Linearity

Additionally to the previous ones, following are often used to visualize the model performance. Therefore, short description for each is provided. As one need to learn to read those graphs in order to be able to come with valid conclusions regarding model's performance.

- Profit Curve
- ROC Curve and AUROC (AUC)
- Cumulative Response and Lift Curves

2.5.1. Accuracy

Accuracy is commonly used to evaluate classifiers. Its popularity is mostly due to its simplicity. However, simplicity is also its enemy and usually it cannot be used alone to give any verdict regarding the performance of the algorithm. It is defined as the following ratio between number of correct class classification and total number of classification performed. (Provost & Fawcett 2013)

$$Accuracy = \frac{Number\ of\ correct\ class\ classifiacations}{Total\ number\ of\ classifications} \qquad (2)$$

We want algorithms to produce accurate results. However, it is not necessary the ultimate goal. Sometimes less precise results can serve better. There are number of reasons for it. First, it may be more important to get result on time than accurate. E.g. we are predicting certain event which is due in few seconds. Our prediction has no value if it comes after the event is already known. Secondly, simpler models tend to generalize the population better than sophisticated ones. Overfitting is an important issue and care need to be taken to avoid it. (Rohrer 2016)

2.5.2. Confusion Matrix

Confusion matrix let to distinguish between different types of success and errors made by the classifier. Unlike accuracy which simply puts everything into single number. The confusion matrix is of n x n dimension, where n is the number of classes. As the example, let's present below confusion matrix for the binary classification.

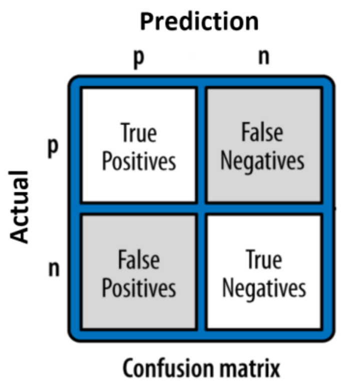


Figure 15. Confusion Matrix (Provost & Fawcett 2013)

Meaning of the confusion matrix cells is as follow:

- TP – True Positive – number of true positive classes which were also classified so by the classifier.
- TN – True Negative – number of true negative classes which were also classified so by the classifier.
- FP – False Positive – Type I Error – number of negative classes which were misclassified as positive by the classifier.
- FN – False Negative – Type II Error – number of positive classes which were misclassified as negative by the classifier.

Based on the confusion matrix we can calculate following derived quality indicators:

- True Negative Rate (TNR), Specificity (SPC)

$$TNR = \frac{TN}{TN + FP} \tag{3}$$

- True Positive Rate (TPR), Sensitivity, Recall

$$TPR = \frac{TP}{TP + FN} \tag{4}$$

- Precision

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

- F1 Score

$$F_1 = 2 \cdot \frac{Precision \ \cdot Recall}{Precision + Recall} \tag{6}$$

- FPR (False Positive Rate)

$$FPR = \frac{FP}{FP + TN} \tag{7}$$

- FNR (False Negative Rate)

$$FNR = \frac{FN}{FN + TP} \tag{8}$$

- Accuracy (ACC)

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{ALL} \tag{9}$$

2.5.3. Unbalanced Class

It is important to consider class distribution while evaluating the model. It is common task for classifiers to try to predict some rare class representing abnormality among otherwise normal entities. It is the case while predicting the fault situations, which normally are expected to occur unfrequently. In that case class distribution is skewed and we refer to it as unbalanced class. (Provost & Fawcett 2013)

Due to unbalanced classes, we cannot simply rely on the accuracy measure. Let's consider simple classifier which always returns prediction "NO FAULT" and with actual fault occurring in average every 1000 cycles. Such classifier would achieve accuracy of 99.9%. Doesn't it look amazing? We would never predict single fault, but the accuracy of the model is still very high. Therefore, for some given business problem, model with the low accuracy, e.g. 30% may do better work than one which is 99.9% accurate. Additionally, difference in accuracy may come whether it was measure on a representative sample or with artificially balanced one. It can be simply summarize that accuracy on itself says nothing about performance of the model. (Provost & Fawcett 2013)

2.5.4. Unequal Costs and Benefits

Worth of considering are the costs associated with the false positive and false negative errors. Default assumption is that both costs are the same. However, in many cases it is wrong to assume so. Classical example involves prediction of cancer. Consequences of false positive are not as big as false negative. False positive in the example means healthy patient being wrongly classified as having the cancer which would most likely result in more tests, meaning additional expense. False negative is opposite, unhealthy patient is classified as healthy one, implying no treatment and potential death. (Provost & Fawcett 2013)

In the context of the machine fault detection one need to evaluate the costs associated with early maintenance due to false positive and unexpected machine breakdown and production stoppage due to false negative.

2.5.5. Expected Value

Expected Value (EV) calculation helps to decide on threshold to be used when determining on class membership. It is calculated as a weighted average of possible outcomes, with weights being the probability of occurrence of the given outcome. The general form for the EV is given below (Provost & Fawcett 2013)

$$EV = p(o_1) \cdot v(o_1) + p(o_2) \cdot v(o_2) + \ p(o_3) \cdot v(o_3) \ldots \qquad (10)$$

Commonly used class membership threshold is 0.5. Meaning, given instance is assigned to class1 if its predicted class membership is 0.5 or more. With many real life applications this default threshold is not the best one. Especially when probabilities are very low. It may be that virtually no instance reaches probability high enough to be assigned to class1. (Provost & Fawcett 2013)

Taking machine fault prediction as an example. Let's calculated expected value from correctly predicting the fault before it occurs:

$$Expected\ benefit\ from\ fault\ prediction = p_f(x) \cdot v_f + [1 - p_f(x)] \cdot v_{nf} \qquad (11)$$

Where:

- $p_f(x)$ – Estimated probability of the fault occurrence in the near future
- v_f – Value we gain from correctly predicting the fault in advance
- v_{nf} – Costs associated with incorrect fault prediction. We predicted the fault but it did not actually occur.

Let's assume following costs. Production line stoppage we estimate at 10'000€. Cost of maintenance is 1'000€. Therefore, if we predict correctly fault in advance we save 10'000 of the potential loss minus 1'000 of the maintenance cost. Giving us $v_f = 10'000€ - 1'000€ = 9'000€$. In case our prediction was wrong we have had unnecessary maintenance for which we need to pay, $v_{nf} = -1'000€$. With those values given we can calculate probability of the fault at which we break even as follow:

$$p_f(x) \cdot 9'000€ - [1 - p_f(x)] \cdot 1'000€ > 0 \tag{12}$$

$$p_f(x) > 0.1 \tag{13}$$

Therefore, according to this simple calculation we should set class1 membership threshold to 0.1, instead of default 0.5

We can now use Expected Value to evaluate models and to determine which one has potential to bring highest benefit. Naturally there are differences between instances and we need to look at aggregated expected value in order to draw conclusions. Simply we calculate expected value based on the model results and cost/benefit matrix. Figure below illustrates the principle. (Provost & Fawcett 2013)

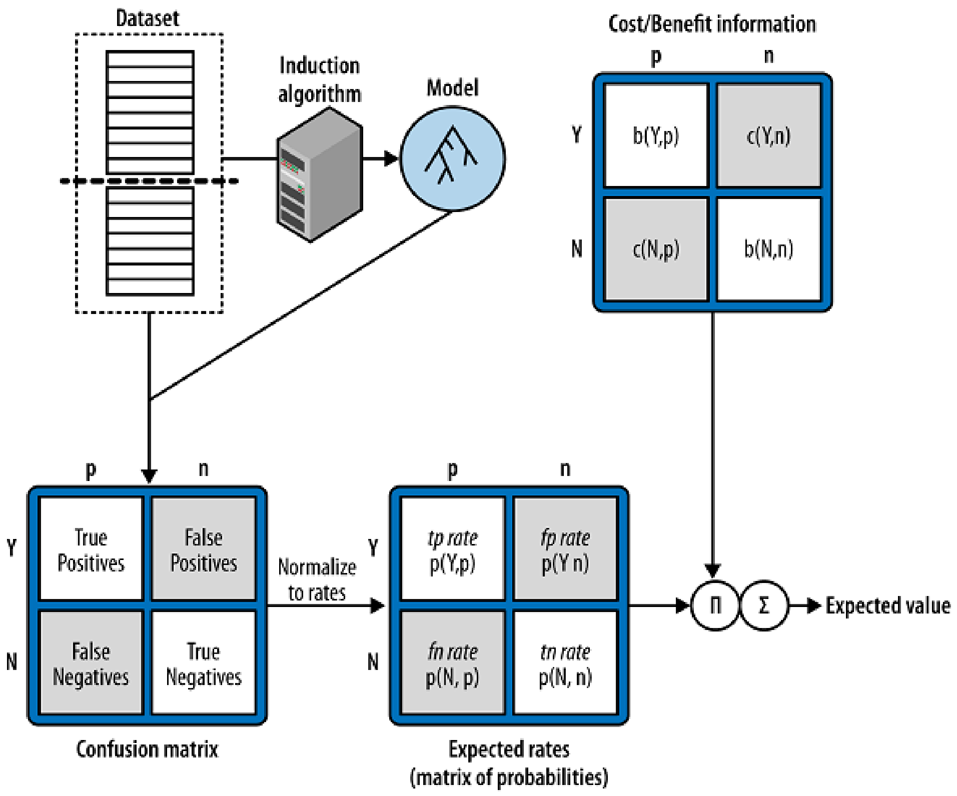


Figure 16. Calculation of the aggregated expected value (Provost & Fawcett 2013)

Same aggregated expected value represented by the formula below (Provost & Fawcett 2013):

$$Expected\ profit = p(Y,p) \cdot b(Y,p) + p(N,p) \cdot b(N,p) + p(N,n) \cdot b(N,n) + p(Y,n) \cdot b(Y,n) \tag{14}$$

2.5.6. RMSE & MAE

Both, RMSE (root mean square error) and MAE (mean absolute error), are commonly used for regression model evaluation. With many more metrics existing, there is a lack of consensus on the best one to use in the evaluation of the model errors. They emphasize different aspects of the error and none is perfect as some information is always lost when condensing big amount of data into single number. RMSE gives more weight to errors with higher absolute errors. Unlike the MAE which gives same weight to all errors. With both calculated for the same model, MAE is never bigger than RMSE. Error distribution is commonly anticipated to be Gaussian, in that case the RMSE has a leverage over the MAE to correctly represent the error distribution. Mentioned error metrics are calculated as follow (Chai & Draxler 2014):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|e_i| \tag{15}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}e_i^2} \tag{16}$$

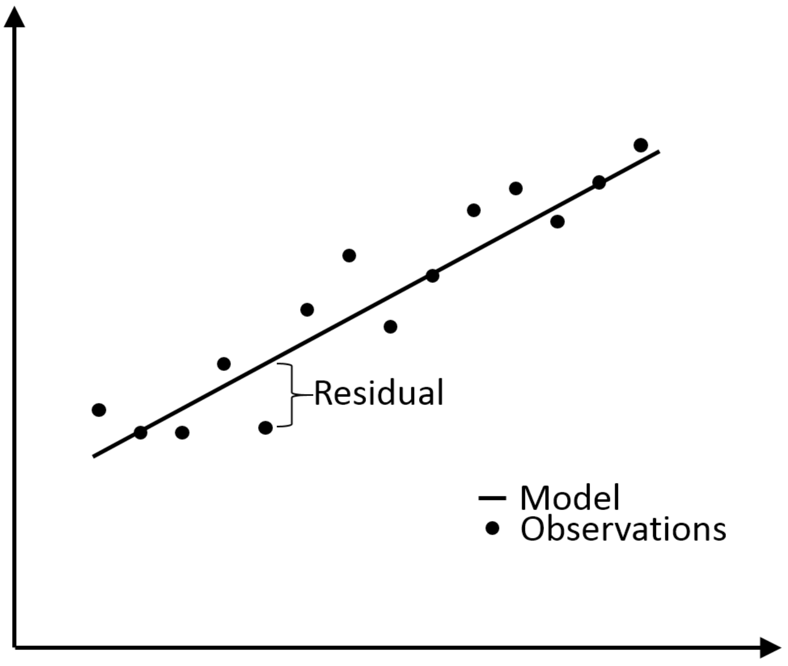


Figure 17. Residuals

2.5.7. Coefficient of Determination – R^2

Coefficient of Determination (R^2) is one of variety of goodness-of-fit statistics which describes how well data fits to the model. R^2 is a relative measure which explains how much of the response variable variation is explained by the model. Its value ranges between 0 and 1. Higher coefficient value means better is model in approximating the real data. An R^2 of 1 indicates that model fits perfectly. However, before fully trusting into value given by the coefficient of determination one should check residual plot first. It should be verified that residuals do not form any kind of pattern, as it would indicate for biased result then. (Aczel et al. 2008)

Coefficient of determination is given by the following formula:

$$R^2 = \frac{SSR}{SST} = \frac{\sum_{i=1}^{n}(\hat{y_i} - \overline{y})^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2} \qquad (17)$$

Where:

SSR – Sum of squares for regression

SST – Total sum of squares

y_i – observed value

$\overline{y}$ – mean

$\hat{y_i}$ – fitted value

Following figure illustrate few regressions with corresponding coefficients of determination.

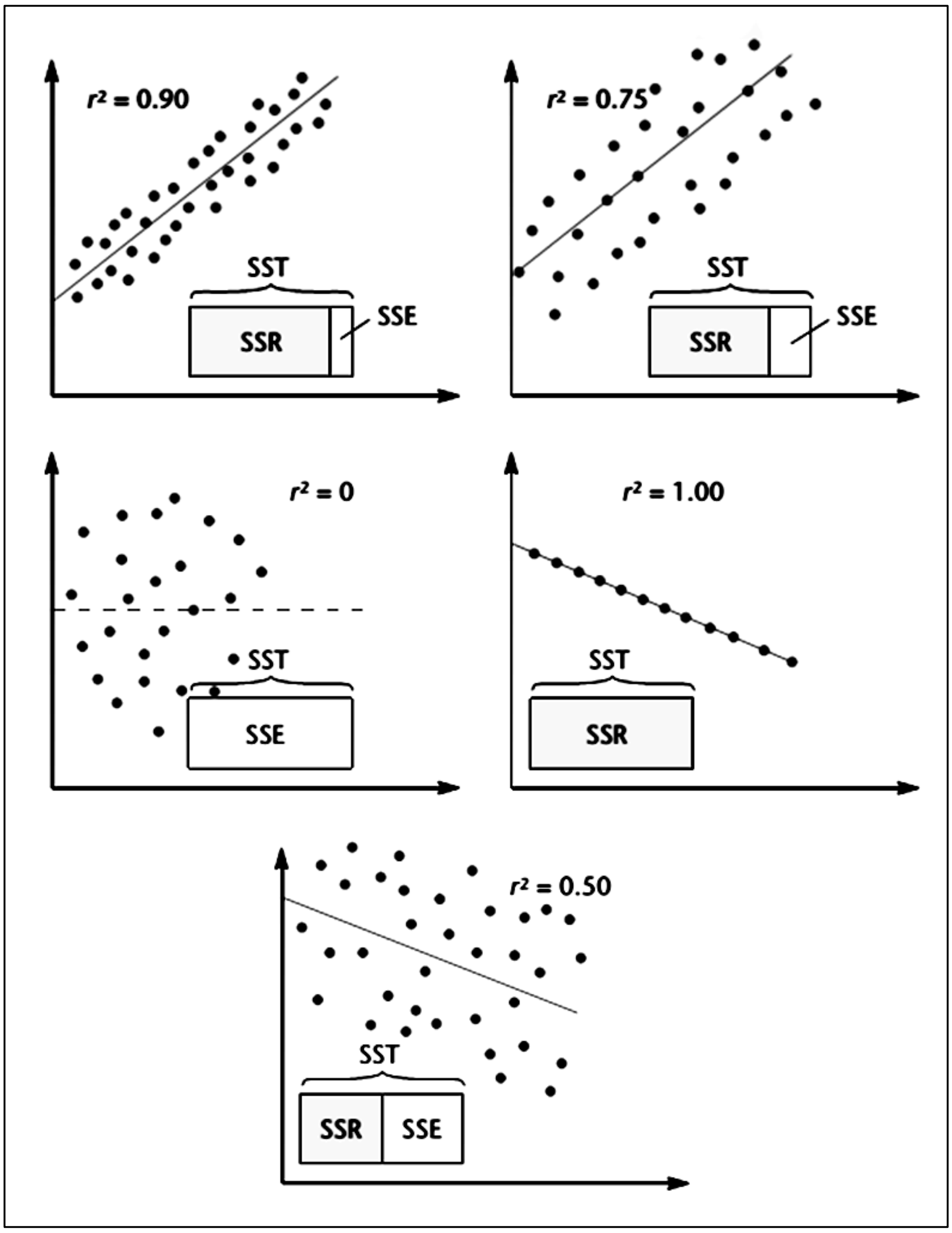


Figure 18. Value of the Coefficient of Determination in Different Regressions (Aczel et al. 2008)

As it was earlier mentioned, it is necessary to verify correctness of the model thru examination of the residuals plot. First of the following plot shows what would be expected as there is no any visible pattern. Residuals are randomly distributed. Following that are few examples of residuals plots which indicates for issues with the model.

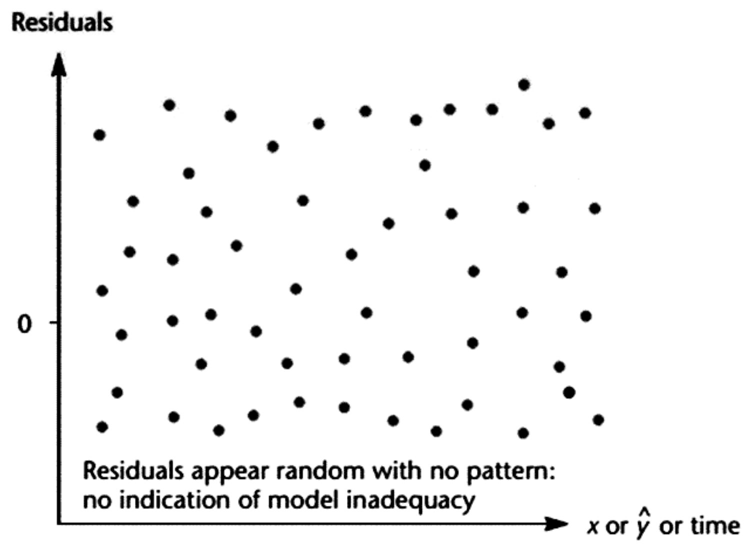


Figure 19. Residuals - Random (Aczel et al. 2008)

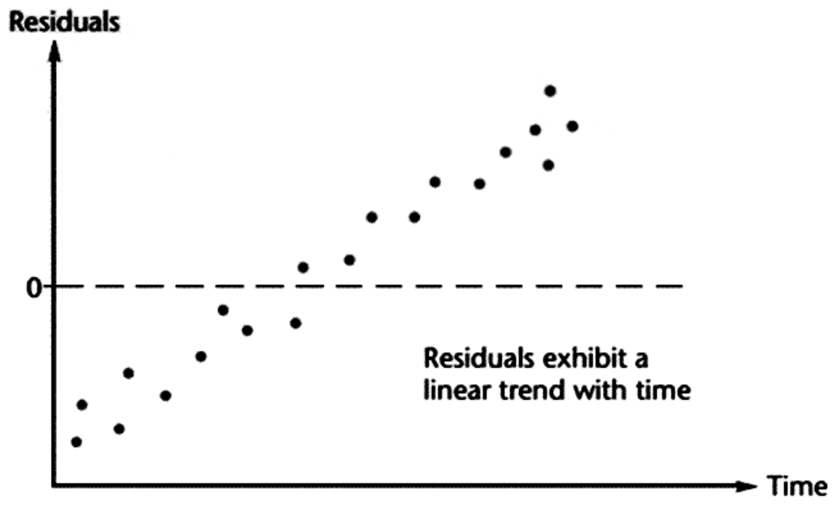


Figure 20. Residuals - Linear Trend (Aczel et al. 2008)

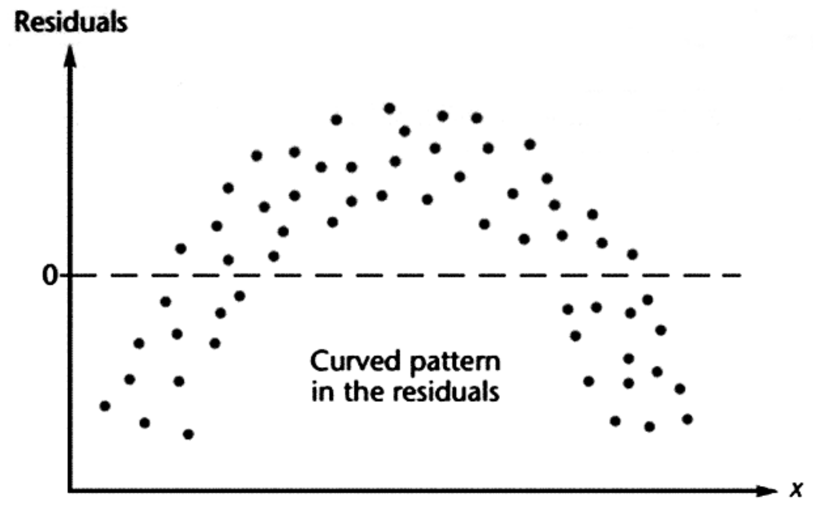


Figure 21. Residuals - Curved Pattern (Aczel et al. 2008)

2.5.8. Overfitting

Overfitting occurs when algorithm becomes to precise regarding sample and loses its ability to generalize. Such algorithm produces very high accuracy with given sample but will perform very poorly when deployed. Simple example could be the algorithm which memorizes all inputs of the sample and related outcomes. Then it executes perfectly with sample data, but it is not generalizing population at all. Meaning any input in the future which is not an exact match is classified with some default result. Of course, given example takes overfitting to the extreme and can easily be spotted as an obvious wrong doing. However, it is not so simple to spot in the real life scenario when some particular algorithm starts to overfit. Therefore, it is important to test model for overfitting. In the simplest way, one would divide data into training and testing sets. Nevertheless, it has its disadvantage, especially when data is scare, as amount of data available for learning is even more limited. Though, there are more advanced techniques to deal with that problem, such as cross-validation and bootstrapping. Figure below shows that best results are achieved when complexity is balanced well. Model should not be too simplistic neither too specific. First would fail to recognize patterns hidden in the data and second one would fail to generalize well in the population.

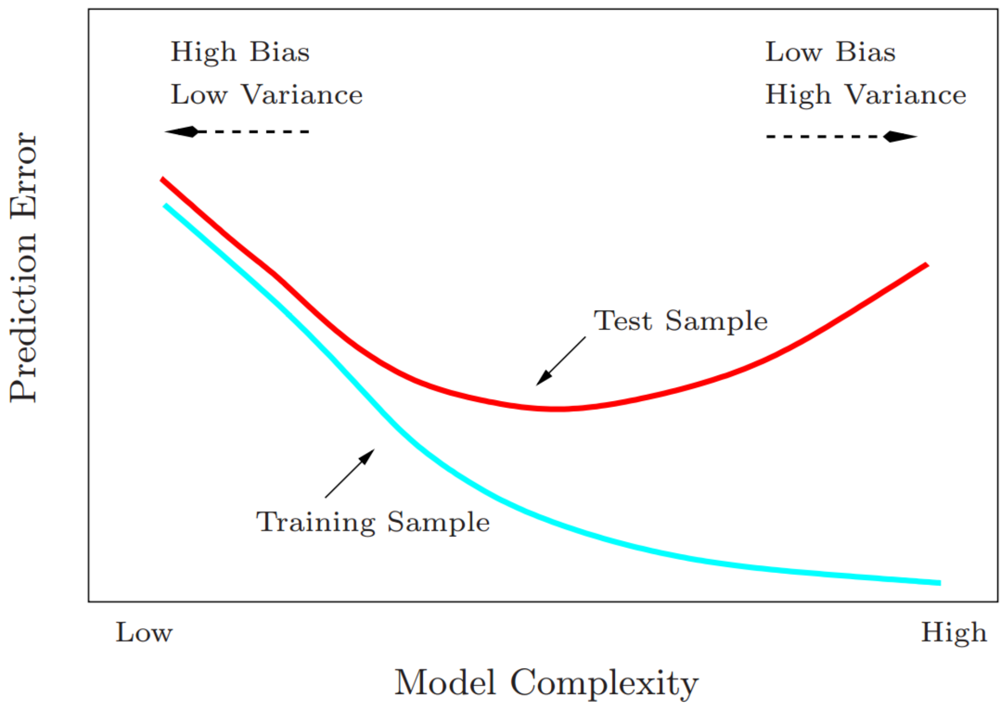


Figure 22. Model Complexity vs. Prediction Error (Hastie et al. 2009)

2.5.9. Training and Consultation Times

Two distinct times of the learning system can be identified. First is the training time. It is defined as the period during which selected algorithm is making inferences from the training data fed to the system. Second one is the consultation time. It is time used by the algorithm to come with the inference for the specific object provided. (Webb 2011)

Those two times need to be considered while choosing the best algorithm for the given data science problem. Most commonly the training time is significantly more than consultation time. Those kind of algorithms are classified as eager learning ones. However, with lazy learning it is oppose, with two common examples being instance-based learning and Lazy Bayesian Rules. (Webb 2011)

Instance based learning actually does not infer anything from the data prior to prediction request for the given instance. It produces prediction based on the similarity of the queried instances to its closest neighbour(s). (Webb 2011)

Instance based algorithms are good for the stream of constant data which is changing over the time. Inserting new data is very fast because training time is limited to the time it takes to store new data input.

Eager learning algorithms are good for less dynamic data types. For the situations in which we can train and retrain algorithms periodically but at not to frequent base. Data for which inferred rules and patterns remain valid over the longer period of the time.

Naturally there are also differences between algorithms in those groups. Usually, we can expect contrary relationship between training time and accuracy. Priorities and constraints need to be decided and taken into consideration. Differences between algorithms can be especially noticed with large data sets. (Rohrer 2016)

2.5.10. Linearity

Significant number of machine learning algorithms make linearity assumption. For classification it means that it is expected for data to be separable by the line or its higher dimension counterpart (e.g. plane). In the case of regression, same assumption means that line or higher dimension counterpart are best to proximate relation between predictor and response variables. In case mentioned assumption is incorrect it would results in accuracy less than optimal. (Rohrer 2016)

To illustrate it bit more, two examples are provided below. First is the example of non-linear classification which is addressed using the linear algorithm.

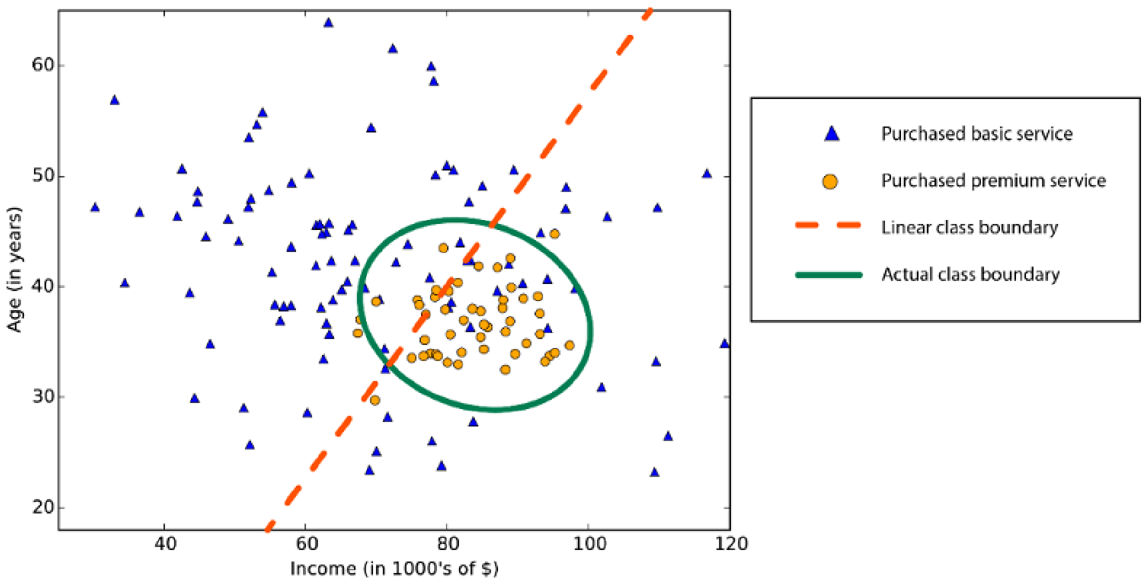


Figure 23. Linear Classification for Non-Linear Problem (Rohrer 2016)

Second example demonstrate the sub-optimal result of the linear regression which is fitted to non-linear trend.

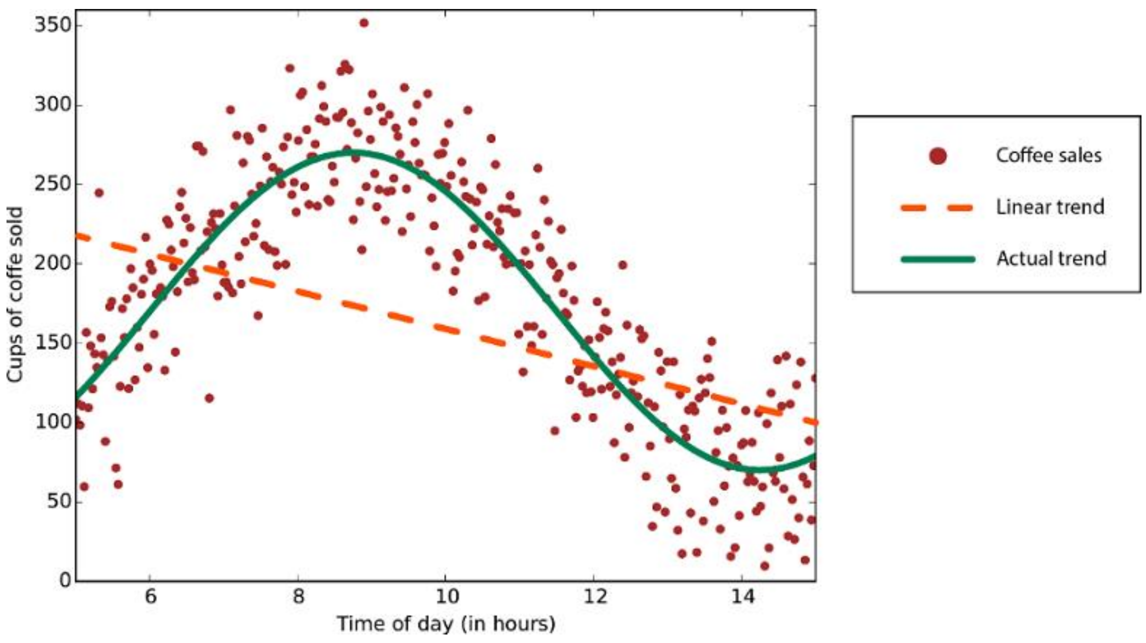


Figure 24. Data with Non-Linear Trend (Rohrer 2016)

2.5.11. Profit Curves

Profit Curve is a visual extension to the earlier mentioned concept of the expected value calculation. It is used with ranking classifier which produces list of instances ordered by the decreasing score. Practically it means taking results of any classifier and ordering them prior to further steps. Expected value is calculated and plotted as it moves along the list starting with highest ranked instances first. (Provost & Fawcett 2013)

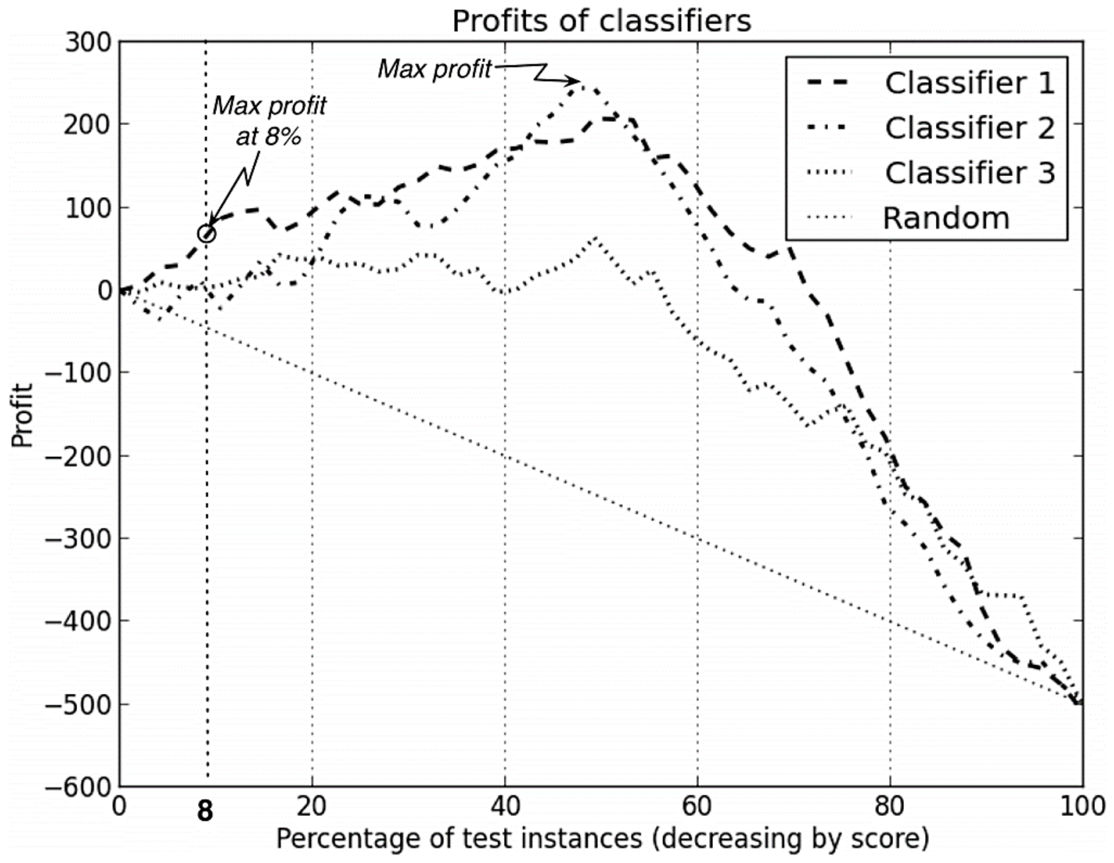


Figure 25. Profit curves of three classifiers (Provost & Fawcett 2013)

Disadvantage of the profit curve is its dependency on additional information being available and fairly stable. Specifically following two (Provost & Fawcett 2013):

- Class priors – proportion of the positive class in the population
- Cost-benefit matrix

2.5.12. ROC Curves

The Receiver Operating Characteristics (ROC) curve depicts relation between TP (true positive) and FP (false positive) rates at different binary classifier's discrimination thresholds. It is universal and does not require any additional information as it is with the profit curve. (Provost & Fawcett 2013)

The bottom left corner of the ROC space represents point at which positive classification is never assigned. It means there are no false positive errors (false alarms), but also not a single true positive (hit). On the contrary, upper right corner represent the point at which positive classification is always assigned. Line between those two represents the performance of the model which would base its prediction on a simple guess. Therefore, upper left point represents perfect classification and best models would be close to it. Generally, each model should be above the diagonal line, as otherwise it would indicate for its performance to be worse than a simple guess. (Provost & Fawcett 2013)

Classifiers which appears on the left are more conservative as those would classify instance to be positive only when strong evidence exists. More to the right means that classifiers become more permissive, positive class is assigned with less and less evidence for it. (Provost & Fawcett 2013)

While interpreting the graph, one should consider class imbalance problem. Often negative instances are in significant majority and even relatively small false alarm rate becomes faster intolerable. Therefore, usually left side of the graph is the most important. (Provost & Fawcett 2013)

Below figure presents ROC space with few different classifiers depicted on it. Classifier C is the best with the A coming second. Classifier B performs as well as simple guess, therefore it seems it is incapable of finding any relation in the data.

```
# X and Y coordinates
x <- c(0, 0.2, 0.4, 0.8)
y <- c(1, 0.6, 0.5, 0.8)
# Plot grpah
plot(x, y,
     xlim = c(0,1),
     ylim = c(0,1),
     xlab = "False Positive Rate",
     ylab = "True Positive Rate",
     pch = 19)
title("ROC")
# Draw diagonal line
lines( c(0,1), c(0,1), col = "red")
# Add labels to plotted points
names <- c("Perfect", "A", "B", "Random Guess")
text(x, y, labels = names, pos = 4)
```

Figure 26. R Script: ROC Space

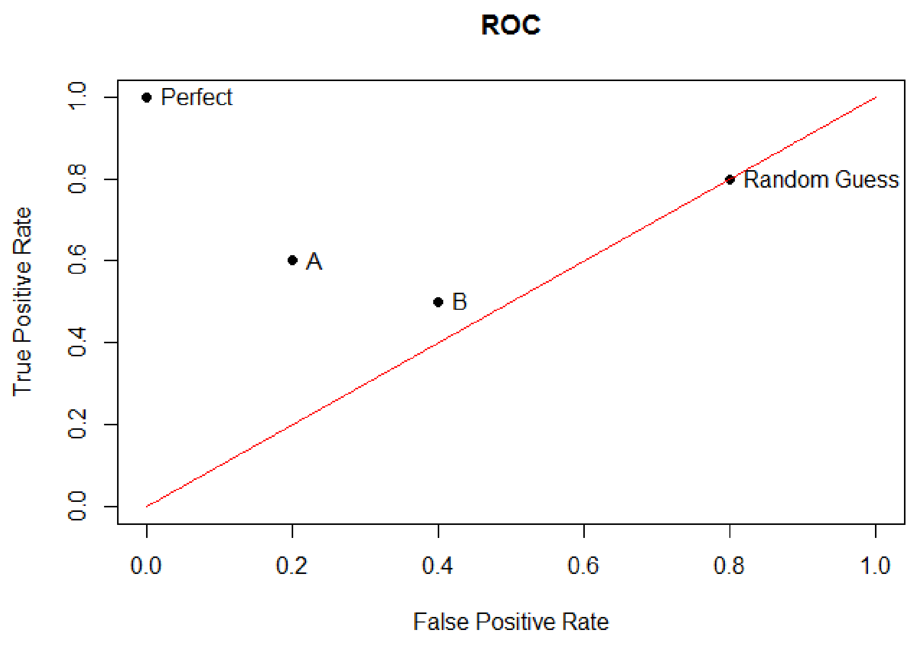
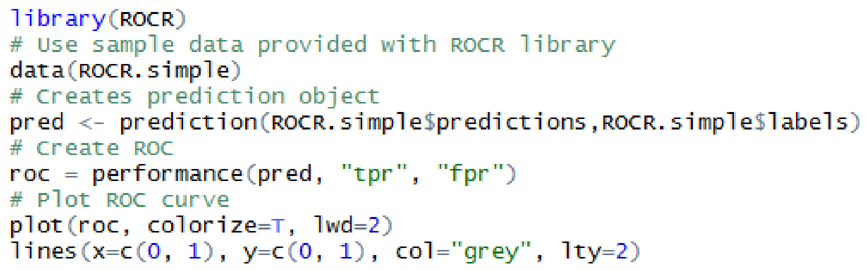


Figure 27. ROC Space

ROC curve illustrate the influence of the discrimination threshold on the TP and FP ratios as shown on the figure below.



```r
library(ROCR)
# Use sample data provided with ROCR library
data(ROCR.simple)
# Creates prediction object
pred <- prediction(ROCR.simple$predictions,ROCR.simple$labels)
# Create ROC
roc = performance(pred, "tpr", "fpr")
# Plot ROC curve
plot(roc, colorize=T, lwd=2)
lines(x=c(0, 1), y=c(0, 1), col="grey", lty=2)
```

Figure 28. R Script: ROC Curve

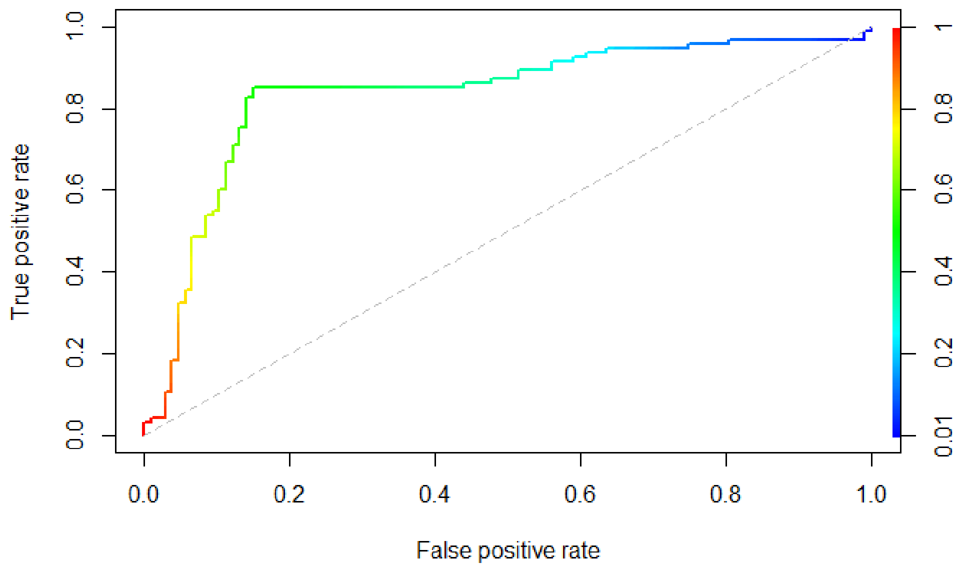
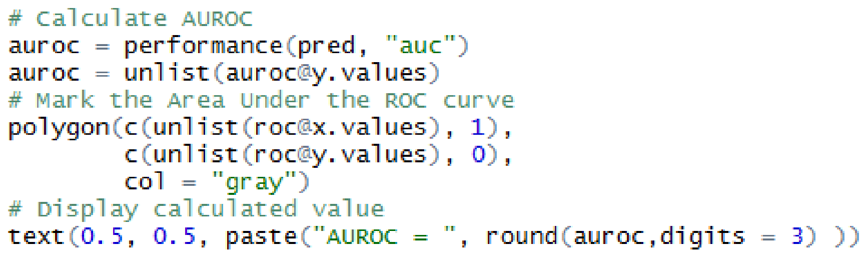


Figure 29. ROC Curve

The Area under the ROC Curve (AUROC)

The AUROC commonly accompanies the ROC curve, as it summarizes the graphical representation of the ROC into a single number. It does what it names implies. Calculate the area under the ROC curve. It ranges from zero to one. Naturally some information is lost, as single number cannot carry same amount of information as the curve. However, it is useful for comparing predictive power of different classifiers.



```
# Calculate AUROC
auroc = performance(pred, "auc")
auroc = unlist(auroc@y.values)
# Mark the Area Under the ROC curve
polygon(c(unlist(roc@x.values), 1),
        c(unlist(roc@y.values), 0),
        col = "gray")
# Display calculated value
text(0.5, 0.5, paste("AUROC = ", round(auroc,digits = 3) ))
```

Figure 30. R Script: Area under the ROC Curve

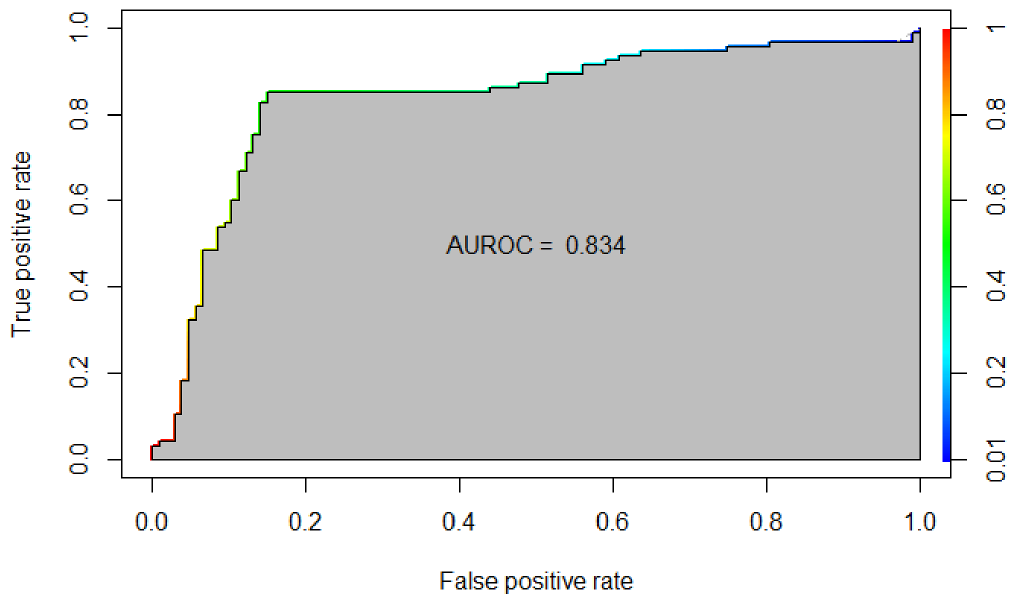


Figure 31. Area under the ROC Curve

2.5.13. Precision-Recall (PR) Curve

Precision-Recall curve is especially useful when class imbalance exists. Precision explains how much model is trustworthy with its predictions of the positive class. High precision means that almost all instances which are predicted to be of the positive class are truly positive. (Lantz 2015)

$$Precision = \frac{TP}{TP + FP} \qquad (18)$$

Recall on the other hand has bit different meaning. It depicts how well model is able to identify all positive classes in the population. It is defined as number of instances which were classified as positive over all positive ones. High recall means that classifiers is able to identify big proportion of all positive classes in the population. (Lantz 2015)

$$Recall = \frac{TP}{TP + FN} \qquad (19)$$

Figure below illustrates meaning of the terms used.

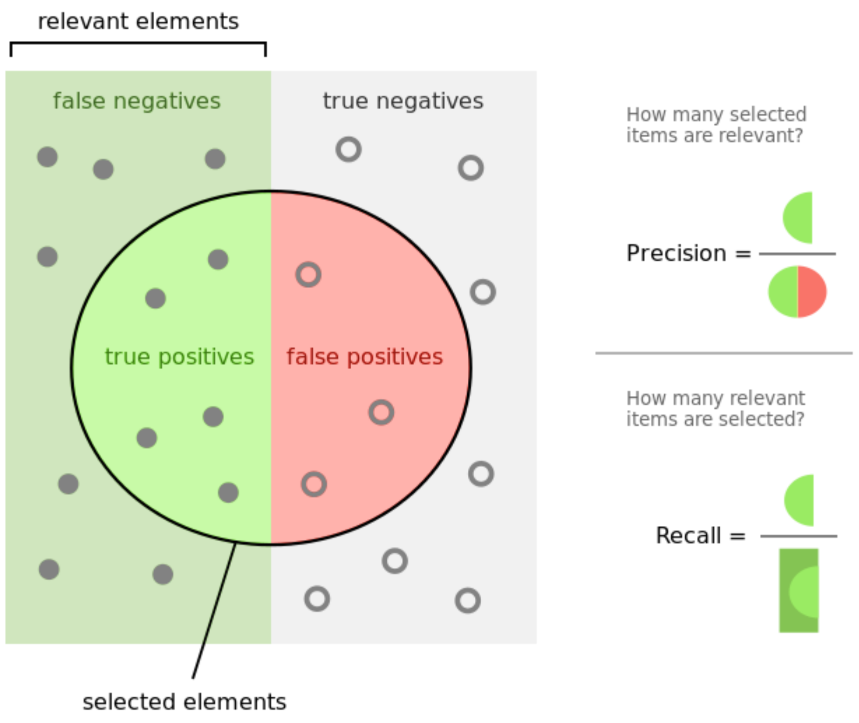
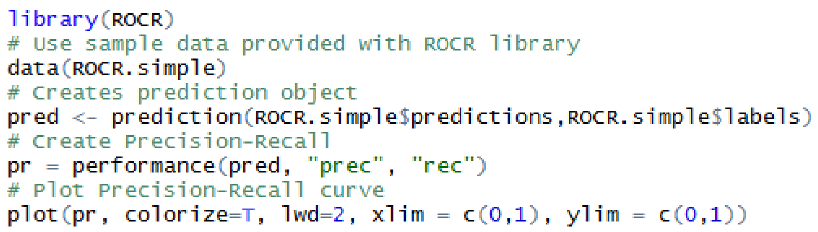


Figure 32. Precision and Recall – By Walber – Own work, CC BY-SA 4.0

Following illustrates sample Precision-Recall Curve plotted with the R.



```
library(ROCR)
# Use sample data provided with ROCR library
data(ROCR.simple)
# Creates prediction object
pred <- prediction(ROCR.simple$predictions,ROCR.simple$labels)
# Create Precision-Recall
pr = performance(pred, "prec", "rec")
# Plot Precision-Recall curve
plot(pr, colorize=T, lwd=2, xlim = c(0,1), ylim = c(0,1))
```

Figure 33. R Script: Precision-Recall Curve

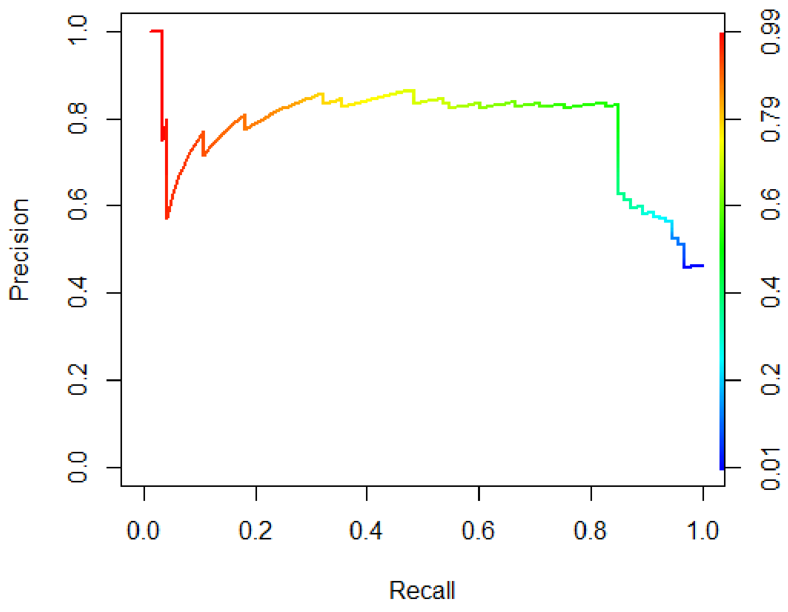
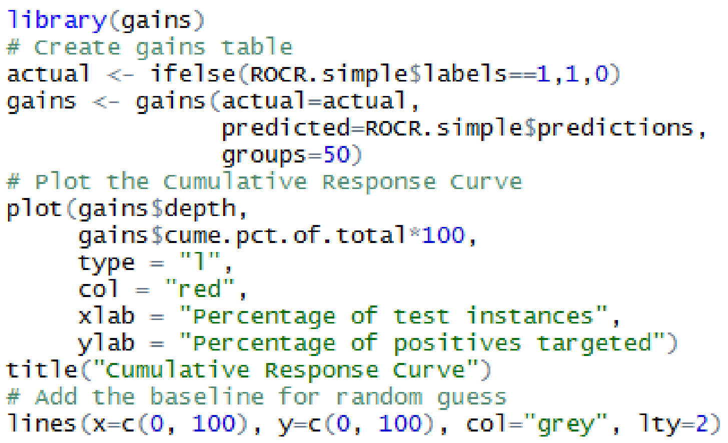


Figure 34. Precision-Recall Curve

2.5.14. Cumulative Response and Lift Curves

Cumulative response and lift curves are maybe not perfect as they miss some of ROC properties. However, they are more intuitive for number of stakeholders who are not involve into data science activities on daily basis. It should not be forgotten how important it is to communicate well with all stakeholders in the project. Therefore, cumulative response curve is often used instead of or in addition to the ROC one. It plots cumulative percentage of correctly classified positive classes as a function of the population that is targeted. Diagonal line indicates for random performance. Therefore, any good classifier should be above that line. (Provost & Fawcett 2013)

Example of the cumulative response curve in R is presented below.



```
library(gains)
# Create gains table
actual <- ifelse(ROCR.simple$labels==1,1,0)
gains <- gains(actual=actual,
               predicted=ROCR.simple$predictions,
               groups=50)
# Plot the Cumulative Response Curve
plot(gains$depth,
     gains$cume.pct.of.total*100,
     type = "l",
     col = "red",
     xlab = "Percentage of test instances",
     ylab = "Percentage of positives targeted")
title("Cumulative Response Curve")
# Add the baseline for random guess
lines(x=c(0, 100), y=c(0, 100), col="grey", lty=2)
```

Figure 35. R Script: Cumulative Response Curve

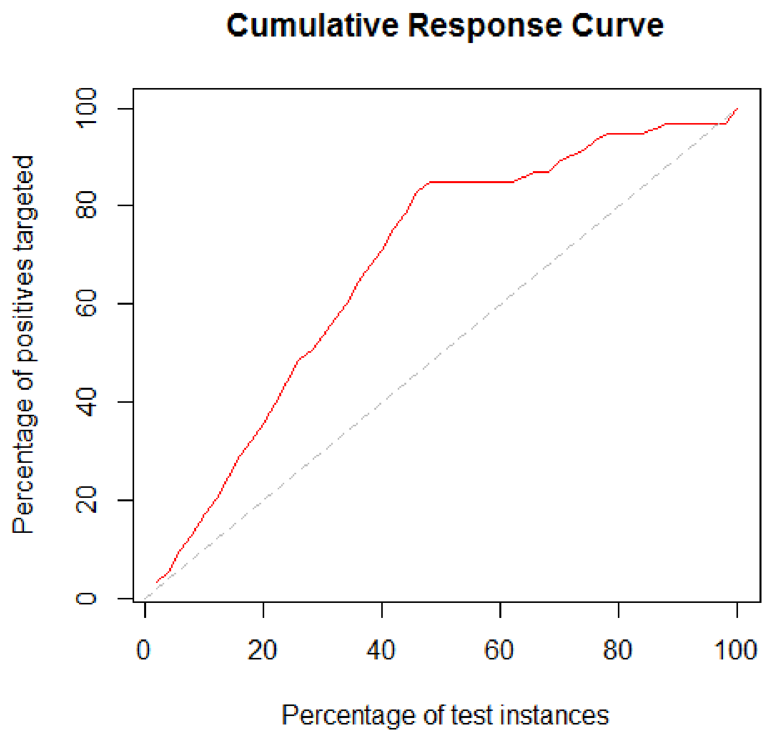


Figure 36. Cumulative Response Curve

The lift curve depicts classifier advantage over the random guess. It is used to show the degree to which it pushes up ability to identify positive classes in the population. Given the example of the population with the equal amount of the positive and negative classes. If we would classify it using random guess, ordered and then go thru half of the classifications we would get also half of the real positive classes, that gives lift of 0.5/0.5 = 1. However, if we would use perfect classifier instead, we would get all of them in the first half. Giving the lift of 1.0/0.5 = 2. (Provost & Fawcett 2013)

```
library(ROCR)
# Use sample data provided with ROCR library
data(ROCR.simple)
# Creates prediction object
pred <- prediction(ROCR.simple$predictions,ROCR.simple$labels)
# Create Lift
pr = performance(pred,"lift","rpp")
# Plot Lift Curve
plot(pr, colorize=T, lwd=2)
title("Lift Curve")
# Plot baseline
lines(x=c(0, 1), y=c(1, 1), col="grey", lty=2)
```

Figure 37. R Script: Lift Curve

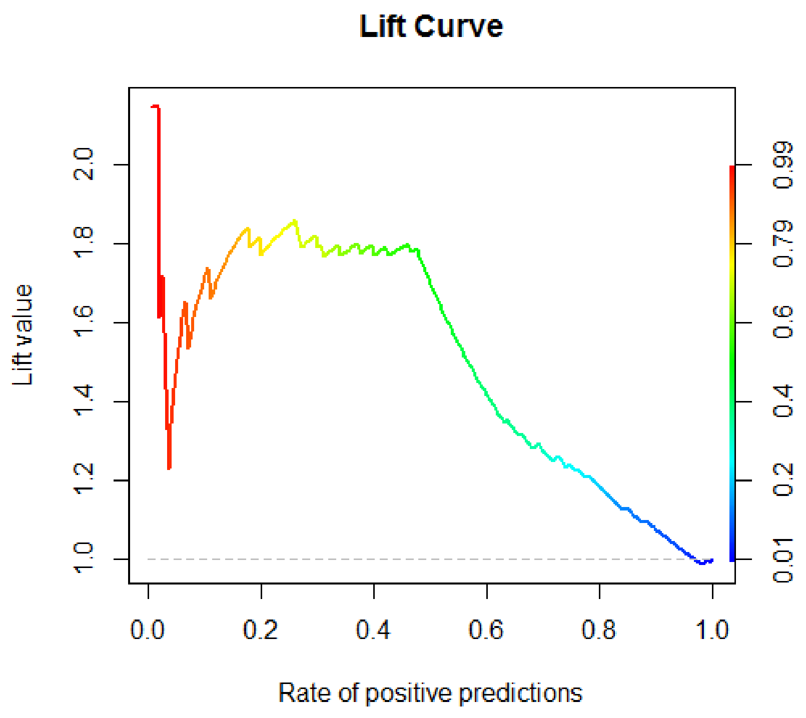


Figure 38. Lift Curve

It should be noted for the both curves that it is assumed that class priors remain same between test set and the population. (Provost & Fawcett 2013)

3. Methodology

CRISP-DM methodology is used during the research. It is constituted from six stages. Those are:

- Business Understanding
- Data Understanding
- Data Preparation
- Modelling
- Evaluation
- Deployment

It is circular model, meaning it meant to work iteratively and to improve on previous results achieved. It is the case in this research. However, only first iteration is the part of the thesis. Following iterations will follow later on.

3.1.Business Understanding

Business understanding started long before the thesis. Author got in touch with the case company through series of two competitions which were organized in cooperation between case company and the University.

Knowledge regarding the business of the case company was obtain through:

- Case company self-authored introductions.
- Factory visits.
- Interviews with managers and engineers.
- Study of publicly available materials

3.2.Data Understanding and Preparation

Due to the lack of the real data, it needed to be generated. Certainly it would be preferred to work with true machine data but it was not mandatory to realize on the research goals. However, following iteration of the project requires concentrate data to be collected.

Generated data base on modified Archard model, which is presented in more detail in the following chapter. It includes also random error factor. The algorithm which is used for data generation was consulted with the case company so that it reflects what could be measured and expected from the real machine.

3.3.Modelling

Modelling is accomplished with the Microsoft Azure Machine Learning. It does not require deep technical understanding of the algorithms and mathematics. However, one should have good understanding of properties of each algorithm in order to be able to choose the right one for a given business problem.

It is possible to provide own algorithm or to utilize one available in e.g. R. However, wide range of the machine learning algorithms is available in the Microsoft Azure Machine Learning Studio. Therefore, it was decide to limit selection to those. Ones that were used are listed below:

- Regression
    - Linear Regression
    - Bayesian Linear Regression

  - Boosted Decision Tree Regression
  - Decision Forest Regression
  - Poisson Regression
  - Neural Network Regression
- Binary Classification
  - Two-Class Averaged Perceptron
  - Two-Class Bayes Point Machine
  - Two-Class Boosted Decision Tree
  - Two-Class Decision Forest
  - Two-Class Decision Jungle
  - Two-Class Locally-Deep Support Vector Machine
  - Two-Class Logistic Regression
  - Two-Class Neural Network
  - Two-Class Support Vector Machine
- Multiclass Classification
  - Multiclass Decision Forest
  - Multiclass Decision Jungle
  - Multiclass Logistic Regression
  - Multiclass Neural Network

3.4. Evaluation

Evaluation of the models is provided in the chapter 5 "Results". It is done using standard metrics provided by the Microsoft Azure Machine Learning and in the context of the business problem. It does not however go very deeply as the real data is missing. It aims at demonstrating what should be considered while evaluating models in further iterations, especially when modelling can be done based on genuine data.

3.5.Deployment

Deployment of the selected models is accomplished in Microsoft Azure Machine Learning. Once more, its objective is to demonstrate the ease of moving from the experimentation, modelling and evaluation to ready deployed predictive service. All is accomplished without the need for much technical skill. Simple demo application is also implemented in .NET utilizing MVVM pattern. General aim is to demonstrate how easily and fast one can create predictive service and use it in a custom application.

4. Formulation and Discussions

Microsoft Azure Machine Learning is a MLaaS (Machine-Learning-as-a-Service) solution for variety of data science projects. Its advantage is ease of use, scalability and extensibility with R and Python. It provides services at every stage of the project, from raw data to deployment as a consumable web service.

Predictive maintenance implementation with Microsoft Azure ML is presented in this chapter. It base on predictive maintenance template provided by the Microsoft and is tailored to the case company needs (Microsoft 2015a). The advantage of using Microsoft Azure ML is speed and ease of development and deployment. Graphical representation of the project and data flow make it also easier to collaborate with business analysts or domain experts. In the following implementation, similarly to the template, three similar solutions are provided. They address same issue using following approaches:

- Regression – it is used to predict Remaining Useful Life (RUL), or otherwise Time to Failure (TTF). Numerical value which indicate for remaining failure free period of the time in selected units as weeks, days, hours, etc.
- Binary classification – predicts if machine is going to fail within given time frame. E.g. will machine fail within next two weeks? Binary, because the answer is only true or false. It does not predict how much time is left, just if failure occurs in a given period of time.
- Multi-class classification – it can be simply seen as an expansion of the binary classification. Instead of using single period of the time we have multiple periods. Therefore, with this approach we are able to predict e.g. if machine is going to fail this week, next week, after two weeks, etc.

> We do prediction for fixed amount of time periods. E.g. four weeks ahead with one week step. Again we are not doing prediction beyond that set period of the time.

In the examples above, we have mentioned about time period to simplify examples and to make clearer on differences between those approaches. However, it should be mentioned that it is not necessary the best approach. Probably it is more adequate to use other units. Time can be replaced with the working hours, cycles, mileage, transactions, etc.

The assumption of this solution that need to be mentioned is that asset monitored has a progressing degradation pattern which could be measured with the sensors available. Therefore, machine learning algorithm applied is able to learn relationship between sensors readings and past failures. It can then use that knowledge to predict the future failure with some degree of certainty. (Microsoft 2015a)

4.1.Data Preparation and Feature Engineering

This is the first step to be performed in the Microsoft Azure ML Studio. Previous activity of business understanding is naturally performed outside the studio environment. Vast amount of different function blocks are available in the studio to perform standard machine learning tasks. However, sometimes some custom activity needs to be implemented. It is done using "Execute R Script" function block. We use it somehow more extensively within this first step to implement our custom functionality with R. Following tasks are part of this step: data sourcing, data labelling, feature engineering and test design preparation.

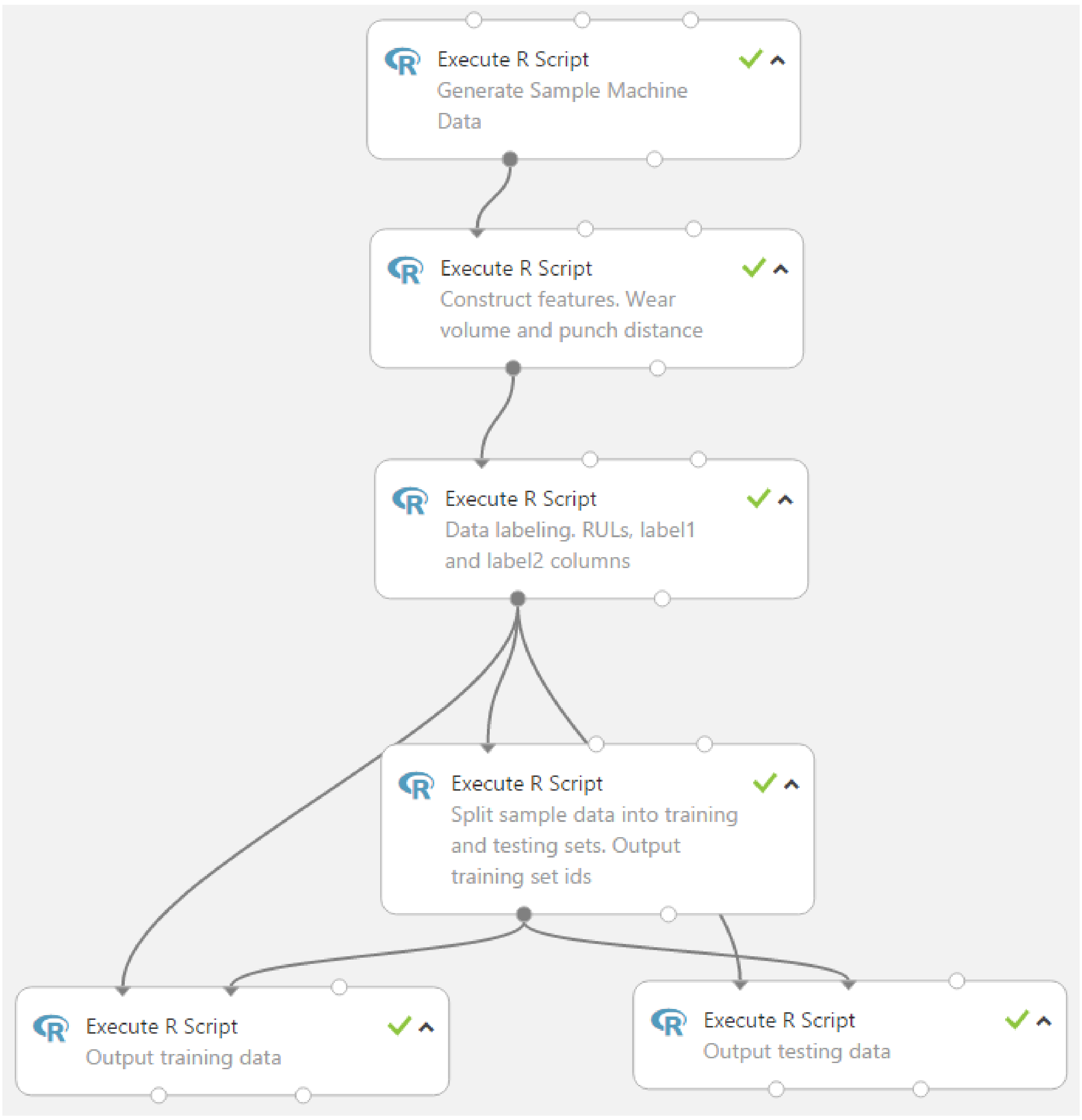

Execute R Script
Generate Sample Machine Data
Execute R Script
Construct features. Wear volume and punch distance
Execute R Script
Data labeling. RULs, label1 and label2 columns
Execute R Script
Split sample data into training and testing sets. Output training set ids
Execute R Script
Output training data
Execute R Script
Output testing data

Figure 39. Azure ML Experiment: Prepare Data

4.1.1. Source Data

Due to lack of feasible real data, source data was generated using the R. It still serves its purpose, as the main goal is to demonstrate possibilities laying in Predictive Maintenance (PdM) and Microsoft Azure Machine Learning. Once potential benefits are perceived by stakeholders, effort can be made to acquire real data for further iterations of this project.

In order for generated data to reassemble closely real one, the tool wear model was used. It is modified Archard model. Main reasoning for the modification is due to the fact that level of the tool wear is a nonlinear function over the loading duration. Basic Archard model assumes it to be on the contrary a linear as it defines variable K in the equation 20 to be constant. In the case of the modified Archard model, wear coefficient K changes over the loading duration. Wear volume W and wear coefficient K in theirs relations to the loading duration are presented on the **Figure *40***. (Ersoy-Nürnberg et al. 2008)

$$W = K\frac{PL}{H} \tag{20}$$

Where:

W is a total wear volume (mm^3)

K is a dimensional constant

P is a total normal force (N)

L is a sliding distance (mm)

H is a surface hardness (MPa)

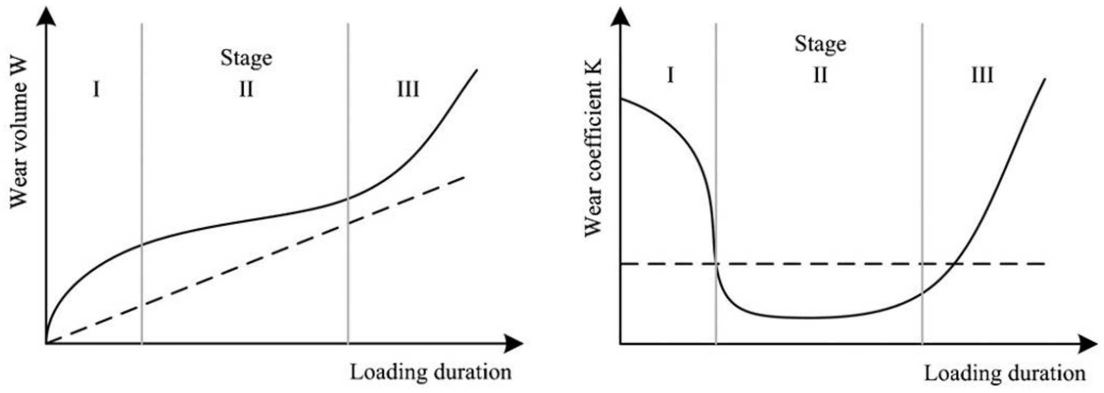


Figure 40. Wear stages for wear volume and coefficient (Ersoy-Nürnberg et al. 2008)

```
wearcoeff <- function(cycle) {
  ifelse(cycle<50,
         (450/(90+0.7^(-cycle)) + 2) / 100,
         (100/(10+1.1^(-cycle+150)) + 2) / 100 )
}
plot(wearcoeff,
      xlim = c(0,140),
      xlab="cycle",
      ylab="Wear Coefficient K")
```

Figure 41. R code to calculate and plot the wear coefficient K

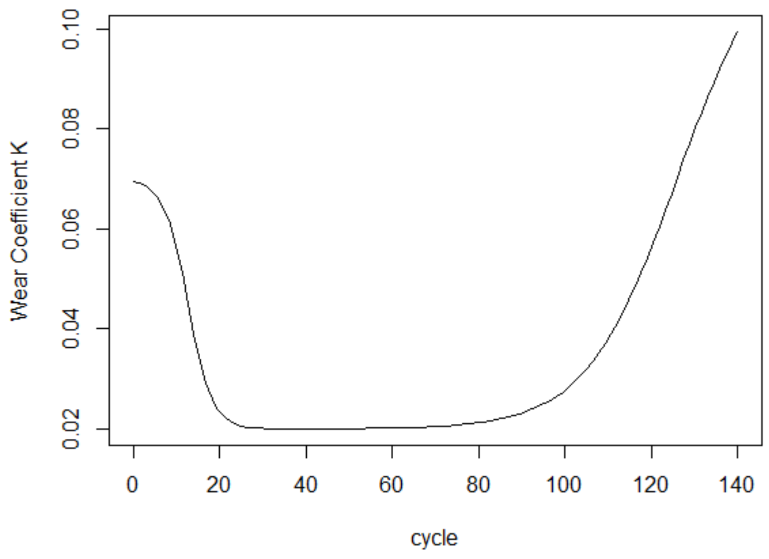


Figure 42. Wear coefficient plotted in R

```
wearvolfunc <- function(cycle, force, distance, hardness) {
  # Using Archard model W = KPL/H
  # K - coefficient as a function of the cycle
  # P - force (N)
  # L - distance (mm)
  # H - material hardness (MPa)

  W <- wearcoeff(cycle)*force*distance/hardness
}

cycles <- c(1:150)
wearvol <- lapply(cycles, wearvolfunc, 1, 10, 1000)
wearvolcum <- cumsum(wearvol)

plot(wearvolcum,
     xlim = c(0,140),
     type = "l",
     xlab="Cycle",
     ylab="Wear Volume W")
```

Figure 43. R code to calculate and plot cumulative wear volume

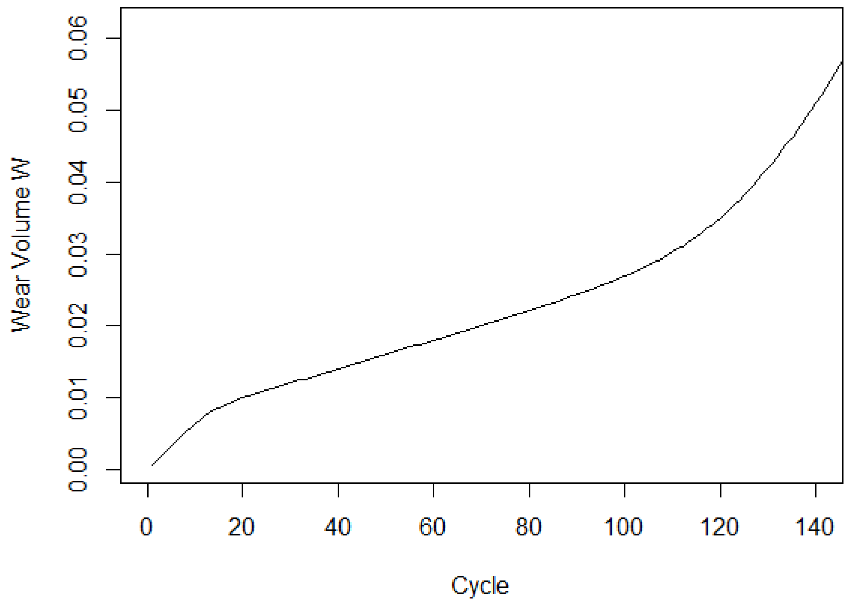


Figure 44. Cumulative wear volume W plotted in R

The data schema of the generated source data is presented in the table below. The id field uniquely identifies machine. Cycles are counted for each machine separately, starting with value of one after tool is changed for the new one. Punch force and distance are recorded for every cycle along with sheet metal type used and fault status.

Table 2. Data schema

Index	Name	Type	Description
1	Id	Integer	Machine identifier
2	Cycle	Integer	Consecutive cycle number
3	punch_force	Double	Punch force applied
4	punch_distance	Double	Punch distance
5	sheetmetal_type	Integer	Type of the sheet metal
9	fault_type	Integer	The type of fault occurred. 0 – no fault 1 – fault

The generated data is a collection of the simulated multivariate time series readings from the case company machines' sensors. Each machine has its own unique identification number "id". Machine progressive usage is measured in cycles. For each cycle, machine's sensors readings are recorded along with the information on the sheet metal type used. With this simple example, there are only two measurements. Those are force of the punch and distance at which tool had physical contact with the sheet metal. Each time series in the simulated data is assumed to start after tool change to the new one and ends when tool is determined to not to be feasible for further usage. Last cycle for a given tool is marked with the fault type being set to one. Tool starts to wear from the beginning. However, tool wear out goes through three stages of initial wear out, normal operation and end of life time. The level of wear out in each of this stage is determined by the coefficient which when plotted reassemble bath tube curve.

Table below presents selected entries from the generated data. Every series starts with cycle one and ends when fault type is set to one.

Table 3. Sample Data – Original Features

id	cycle	punch_force	punch_distance	sheetmetal_type	fault_type
1	1	1.572853	96	1	0
1	2	1.898390	98	2	0
…	…				
1	102	1.991099	58	3	0
1	103	1.068447	70	1	0
1	104	1.193310	92	3	0
1	105	1.267212	75	1	1
2	1	1.624714	83	1	0
2	2	1.405690	100	3	0
…	…				
2	99	1.538794	86	2	0
2	100	1.152788	67	2	0
2	101	1.057373	93	1	0
2	102	1.539462	56	1	1

As stated earlier, training and testing data are derived from the same input data. The difference is only in the way we use them. Obviously, we do not use fault type column while testing the model as that information is not available for prediction in a real time data. Therefore, that information is not available for prediction but we use it to verify the accuracy of predictions made by the model.

4.1.2. Feature Engineering

It is one of the crucial steps and utilizing domain knowledge at this stage can bring significant improvement in the prediction results later on. Often data scientist along with domain expert can select, extract and construct features which better reflect the problem than just simple raw data. Machine learning algorithms' quality of results highly depends on the input data.

In the example we can distinguish two types of features. Ones that are selected form the collected data and ones that are constructed. Though, it does not limit types of features to those two, but those are the ones used in the example.

Often the input data will contain too much facts and it is crucial to decide which ones should be included for further modelling. Feature selection task aims at removing unrelated features from the data. In the case presented all features are used.

Second type of features, mentioned earlier to be used in the example are added during feature construction task. This is the actual core activity of the feature engineering which makes big difference. It is difficult part of machine learning because it requires human expertise and usually a lot of manual work. Following features are constructed in the example used:

- "tool_wear_vol" – tool wear volume estimated based on the modified Archard model.
- "cum_tool_wear_vol" – cumulative tool wear volume for the lifetime of the given tool.
- "cum_punch_distance" – cumulative punch distance for the lifetime of the given tool.

Table 4. Sample Data – Constructed Features

id	cycle	tool_wear_vol	cum_tool_wear_vol	cum_punch_distance	fault_type
1	1	0.0104516	0.0104516	96	0
1	2	0.0064083	0.0168600	194	0
…	…				
1	102	0.0011296	0.1900954	7600	0
1	103	0.0022575	0.1923529	7670	0
1	104	0.0011377	0.1934906	7762	0
1	105	0.0030474	0.1965380	7837	1
2	1	0.0093342	0.0093342	83	0
2	2	0.0032280	0.0125622	183	0
…	…				
2	99	0.0017989	0.1910761	7681	0
2	100	0.0010755	0.1921516	7748	0
2	101	0.0028092	0.1949608	7841	0
2	102	0.0025298	0.1974906	7897	1

4.1.3. Data Labelling

Source data provides entries as they would be read from the machine. However, further work needs to done on this data in order to achieve goals stated. The main aim is to build the service which is able to predict next failure events on the real data prior to theirs occurrence. As it was stated earlier. Within the scope of this research three slightly different approaches to this problem are taken. It reflects also in data labelling.

Using regression we are estimating remaining useful lifetime (RUL) of the tool. Therefore, we need to add this information into historical data in order to support the learning process. We do so by adding three columns, as we further subdivide the problem into:

- Predict number of remaining cycles – "RUL_cycle"
- Predict remaining contact distance – "RUL_distance"

- Predict remaining tool wear volume – “RUL_wear”

In case of the binary classification, angle of approach is bit different to the regression one. As the name suggest, the resulting model can only provide binary result of 0 or 1. We will interpret it as it would answer to the question: Given input data, does tool require change within the given time window? Column named “label1” is used to mark window of interest for this approach.

Multiclass classification can be seen as an extension of the binary one. The result in this case can be 0 … n. Where “n” is the number of time windows. E.g. with two time windows, the results can be in range 0 … 2. Continuing with the example. We interpret it in the following manner. Zero means that there is no evidence which would support believe that error is going to occur during any of defined windows. One or two as a result, in contrary mean that model predicts fault to arise in first or second time window. Column “label2” is added with values as follow: zero for entry out of any defined window, one for first window and two for second window.

Window sizes and ultimately the model used need to be selected based on the business problem. In the example below w0 = 2 and w1 = 4 were used.

Table 5. Sample Data - Labelling

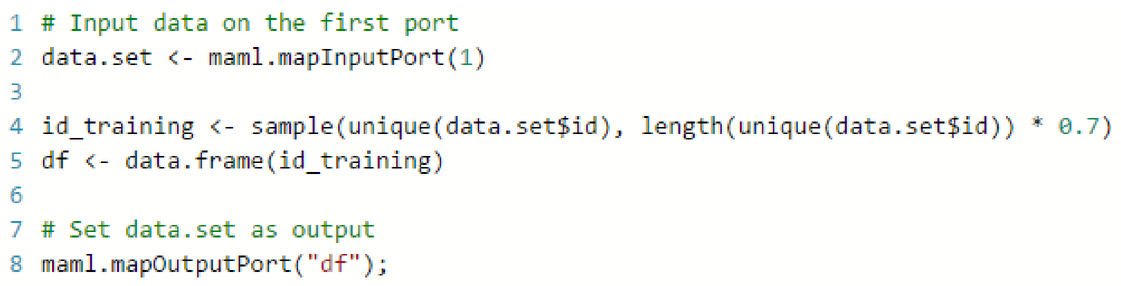
id	cycle	…	fault_type	RUL_cycle	RUL_distance	RUL_wear	label1	label2
1	1		0	104	7741	0.1860864	0	0
1	2		0	103	7643	0.1796781	0	0
1	3		0	102	7590	0.1737695	0	0
…	…							
1	98		0	7	508	0.0153153	0	0
1	99		0	6	452	0.0136418	0	0
1	100		0	5	368	0.0091285	0	0

1	101		0	4	295	0.0075722	1	1
1	102		0	3	237	0.0064426	1	1
1	103		0	2	167	0.0041851	1	2
1	104		0	1	75	0.0030474	1	2
1	105		1	0	0	0.0000000	1	2

4.1.4. Prepare the Testing Data

It is important to test models on data that was not used to train them. Therefore, training and testing sets are distinguished. Training set is used to find patterns in the data and model is build based on it. Test set is needed to verify the performance of the model, one that would be expected when model is deployed on real data. Various options for dealing with training and testing exist, e.g. cross validation. One presented here is probably the most intuitive and simple.

Data entries are randomly divided into training and testing data sets. Custom R script presented below takes as an input generated data that is described earlier and splits it so 70% of entries are allocated to training set and remaining 30% to testing set. Split is done based on machine id. Therefore, whole time series are moved.



```r
# Input data on the first port
data.set <- maml.mapInputPort(1)

id_training <- sample(unique(data.set$id), length(unique(data.set$id)) * 0.7)
df <- data.frame(id_training)

# Set data.set as output
maml.mapOutputPort("df");
```

Figure 45. R Script used to split input data into training and testing data sets

4.2. Train and Evaluate Model

We are going to build three different final models in order to present various approach paths possible. We will try to answer to the same problem in three different ways. Usually only one approach would be selected based on business problem at hand. However, as this serve as an example we will use three of them at the same time.

Following Azure ML modules are used during this step:

- Project Columns
- Metadata Editor
- Filter Based Feature Selection
- Train Model

Project Columns module is used to create subset of columns to be used in downstream operations. It does not alter the source dataset. It can be used to explicitly list columns to either include or exclude. It is used e.g. to remove unwanted columns based on decisions made during the feature selection stage. (Microsoft 2015b)

Metadata Editor module is used to alter data which describe columns in a dataset. However, it does not modify value or data type itself. It is typically used to set flags on columns, such as IsCategorical or IsFeature. Those are commonly required by other modules, e.g. some learners. (Microsoft 2015b)

Filter Based Feature Selection is used to guide or even automate feature selection process. Module determines which of the columns have the greatest predictive power. It does so by applying selected statistical test. The output columns are ordered according to theirs predictive power determined, with score available for each of the column. Following feature scoring methods are available: Pearson Correlation, Mutual information, Kendall Correlation, Spearman Correlation, Chi Squared, Fisher Score and Count Based. (Microsoft 2015b)

Train Model is one of the core modules used. As it name states, it is used to train model based on historical data provided as an input. One of the column need to be marked as a "label". The label column contains values to be predicted and is discarded as a predictor of the model. Additionally, one out of many available classification or regression algorithms is provided as the second input to the module. Algorithm specify the way in which statistical patterns are extracted from the historical data to be used later on for predictions. (Microsoft 2015b)

4.2.1. Regression Models

Following regression algorithms were evaluated:

- Linear Regression – model fits the line which best presents relation between explanatory variables and the output variable. In this case ordinal least square method was used.
- Bayesian Linear Regression – it uses Bayesian approach and intrinsically it provides prior probability distribution information in addition to linear regression. It combines prior information with a likelihood function to create parameters' estimates. (Microsoft 2015b)
- Boosted Decision Tree Regression – it uses boosting to create ensemble of regression trees.

- Decision Forest Regression – regression model that utilizes ensemble of decision trees.
- Poisson Regression – is used to predict numerical values, usually counts of independent events over a given time frame, assumed that (Microsoft 2015b):
  - Response variable follows a Poisson distribution
  - Response variable is non-negative integer
- Neural Network Regression – it can model non-linear dependence between explanatory variables and dependent variable. It also allows for customization of the network architecture using the Net# language. Therefore, this model can be considered especially when more traditional ones are not giving good results. (Microsoft 2015b)

Azure Machine Learning experiment which utilizes before mentioned algorithms is presented on the figure below.

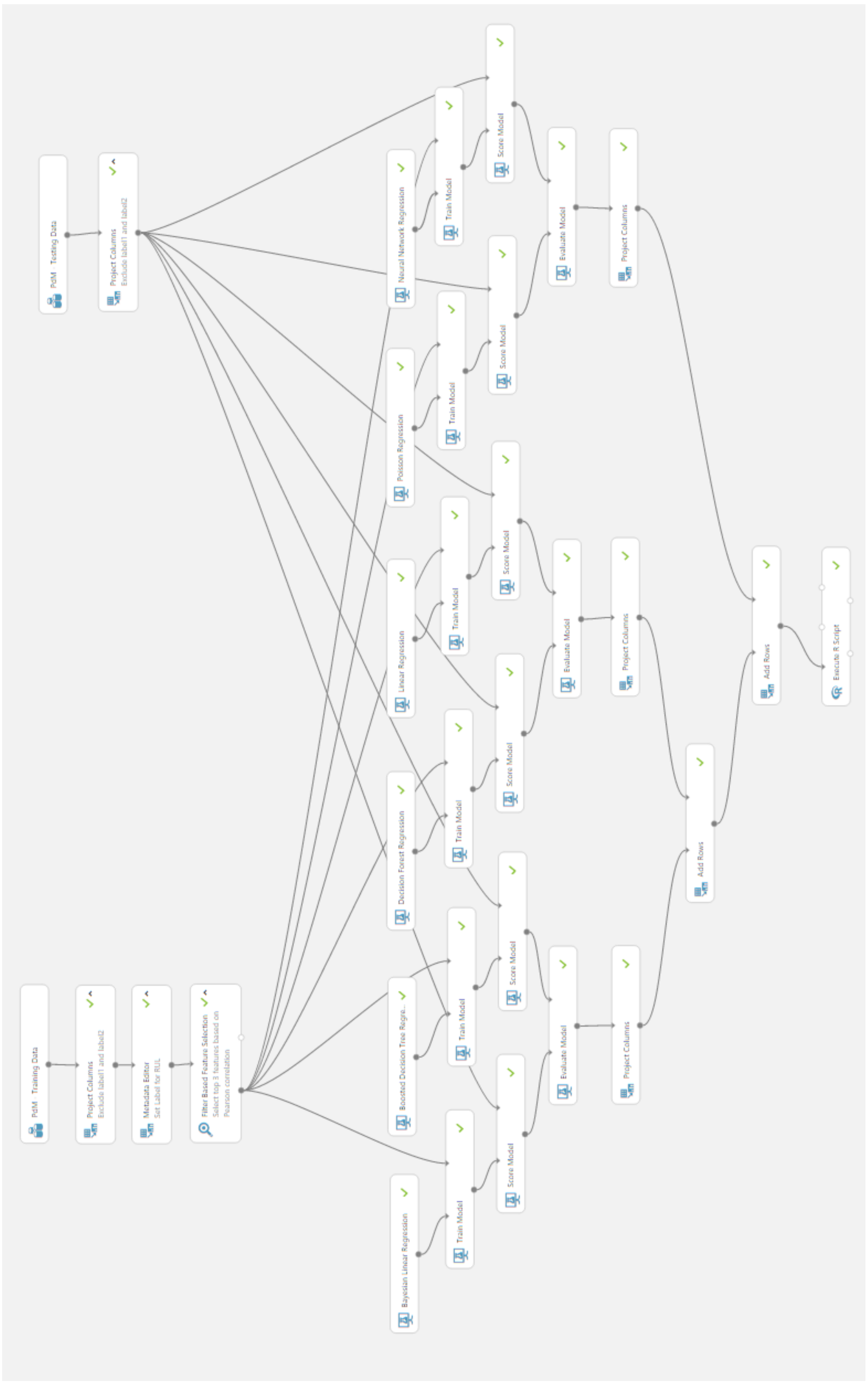

PdM - Testing Data
Project Columns
Exclude label1 and label2
PdM - Training Data
Project Columns
Exclude label1 and label2
Metadata Editor
Set Label for RUL
Filter Based Feature Selection
Select top 3 features based on Pearson correlation
Neural Network Regression
Poisson Regression
Linear Regression
Decision Forest Regression
Boosted Decision Tree Regre...
Bayesian Linear Regression
Train Model
Score Model
Evaluate Model
Project Columns
Add Rows
Execute R Script

Figure 46. Azure ML Experiment: Regression

4.2.2. Binary Classification Models

Binary classification models classify its input entry into one of two classes based on its classification rule. This kind of classification is very common for many business problems. It could be that in some cases only resulting class is given. However, it is very common that along with resulting class its probability is provided. That gives insights into model confidence regarding particular classification.

Performance of the following algorithms was evaluated:

- Two-Class Averaged Perceptron – is an online algorithm, meaning it learns one instance at the time. Therefore, it can learn continuously as new data arrives. Its updates are driven by the error. It is simpler and faster version of the neural network and suits for linearly separable classes. (Microsoft 2015b)
- Two-Class Bayes Point Machine – is a Bayesian linear classification algorithm that is suggested to be an improvement to the support vector machine one. It has its roots in Bayesian theory. The main idea behind the algorithm is to approximate optimal Bayesian average of linear classifiers. The final single "average" classifier is called Bayes Point, hence the name of the algorithm. (Microsoft 2015b)
- Two-Class Boosted Decision Tree – is a binary classification algorithm which uses ensemble of decision trees. Trees are created in such a way that next one is compensating on weaknesses of the directly preceding one. (Microsoft 2015b)
- Two-Class Decision Forest – is a binary classification algorithm which uses random decision forests algorithms during supervised ensemble

learning. Advantage of this method is its fast learning time. (Microsoft 2015b)

- Two-Class Decision Jungle – is an algorithm that builds upon the decision forests algorithm. Its key feature is decision directed acyclic graphs (DAGs) that allows for lower memory footprint at the cost of slightly longer training time. (Microsoft 2015b)
- Two-Class Locally-Deep Support Vector Machine – used to create non-linear support vector machine classifier. It is optimized for training time with slight impact on its accuracy. It can be used when a linear model performs poorly and when model size is an important factor. (Microsoft 2015b)
- Two-Class Logistic Regression – it creates well known logit model which does prediction of the categorical dependent variable by fitting of the data to a logistic function. (Microsoft 2015b)
- Two-Class Neural Network – it creates neural network module for binary classification. (Microsoft 2015b)
- Two-Class Support Vector Machine – is creates binary classification model that uses well known and widely used support vector machine. For more information refer to paragraph 2.4.6

Due to its size, experiment with all above mentioned algorithms being evaluated at the same time is not shown. Instead figure below depicts the experiment with a single binary classification algorithm being used. Worth noting is SMOTE module.

SMOTE (Synthetic Minority Oversampling Technique) is a module used to approach imbalanced datasets. Its purpose is to increase number of minority cases so that resulting dataset is more balance. Using this technique should yield better results than just simple multiplication of minority cases. (Microsoft 2015b)

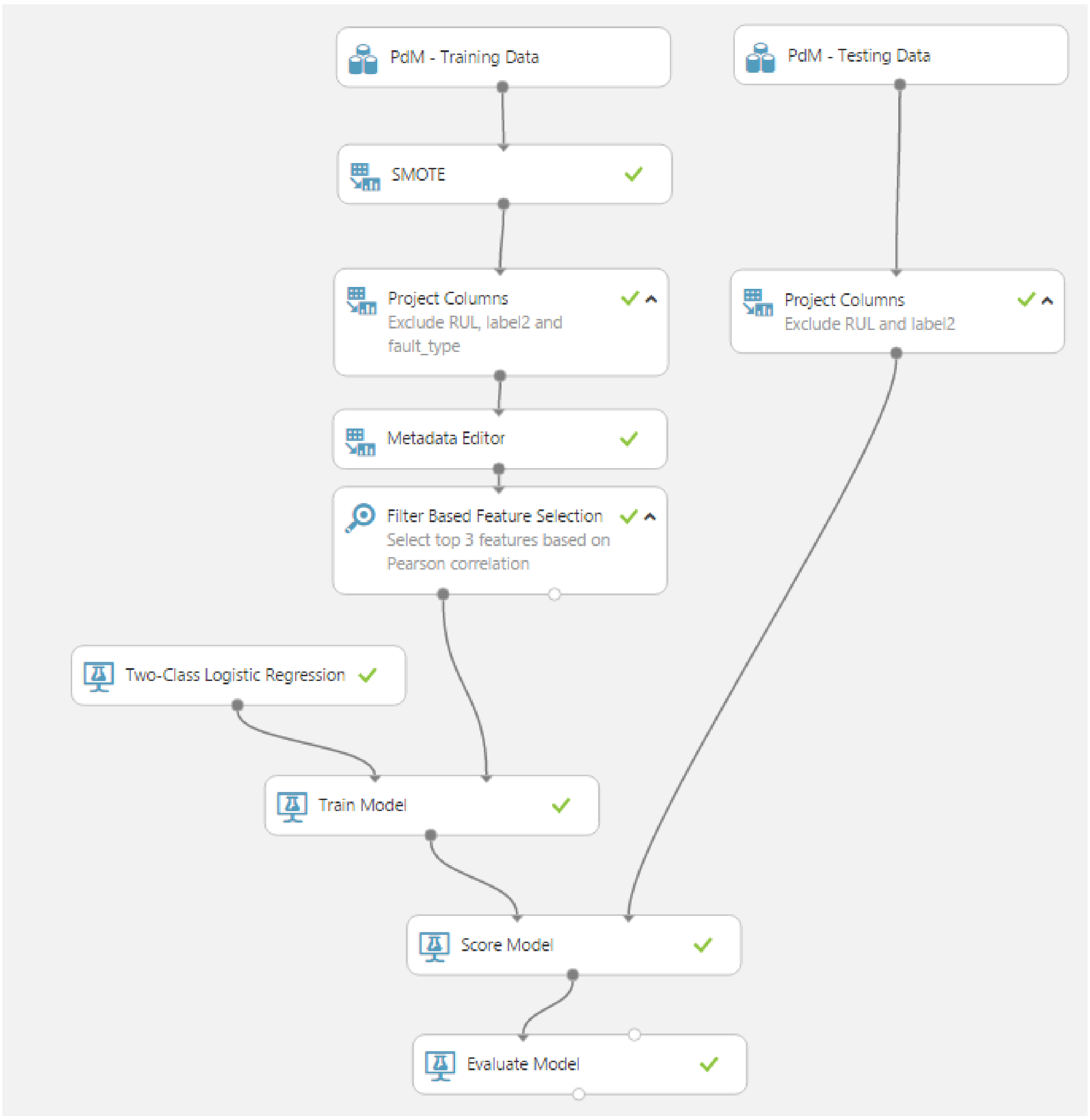


Figure 47. Azure ML Experiment: Binary Classification

4.2.3. Multiclass Classification Models

Following multiclass classifiers were evaluated:

- Multiclass Decision Forest
- Multiclass Decision Jungle
- Multiclass Logistic Regression
- Multiclass Neural Network

All of the above mentioned are closely connected to theirs two-class counterparts. Therefore, curious reader should refer to related algorithms description in the previous paragraph.

Additionally, ordinal regression was used as a multiclass classifier. Ordinal regression is used when dependent variable is ordinal. Such variable can be ordered and ranked but the distance between given values has no meaning. In the perspective of the failure prediction, "label2" is an ordinal variable. Its values are on a scale from 0 to 2, with 0 representing lowest severity and 2 highest one. It is correct to use ordinal regression because order between values of "label2" can be established. With previous mentioned multiclass classifiers this order has no meaning and it can be seen as a disadvantage in this case. (Microsoft 2015a)

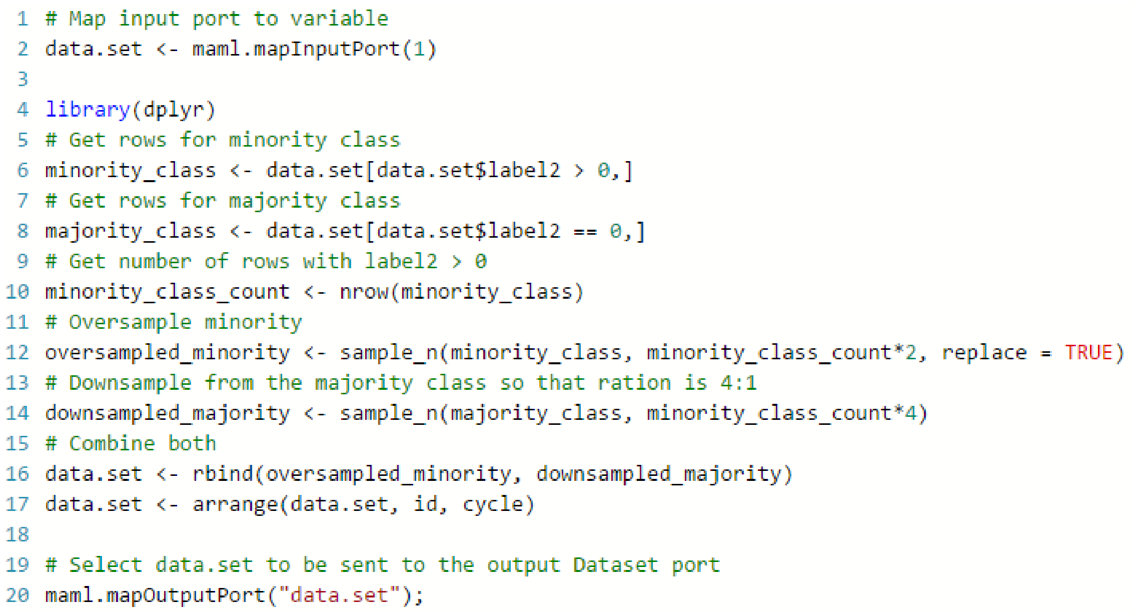
Ordinal regression takes as an input binary classification model. Following two were used:

- Two-Class Logistic Regression
- Two-Class Neural Network

Problem with the ordinal regression in the case described is with its evaluation. Metrics provided by the Azure ML Studio are listed below. Main challenge with them is that they are hardly interpretable given highly imbalanced dataset. Baccianella et al. (2009) acknowledged this issue and are suggesting macro averaged versions of those measures to be used instead.

- Mean Zero One Error
- Mean Absolute Error
- Root Mean Squared Error

The snapshot of the experiment is shown below. With binary classification it was shown how to use SMOTE module to address unbalanced class. In the case of this example, custom R script is written to do oversampling of the minority class and down sampling of the majority class.



```r
# Map input port to variable
data.set <- maml.mapInputPort(1)

library(dplyr)
# Get rows for minority class
minority_class <- data.set[data.set$label2 > 0,]
# Get rows for majority class
majority_class <- data.set[data.set$label2 == 0,]
# Get number of rows with label2 > 0
minority_class_count <- nrow(minority_class)
# Oversample minority
oversampled_minority <- sample_n(minority_class, minority_class_count*2, replace = TRUE)
# Downsample from the majority class so that ration is 4:1
downsampled_majority <- sample_n(majority_class, minority_class_count*4)
# Combine both
data.set <- rbind(oversampled_minority, downsampled_majority)
data.set <- arrange(data.set, id, cycle)

# Select data.set to be sent to the output Dataset port
maml.mapOutputPort("data.set");
```

Figure 48. R Script: Oversampling and Downsampling

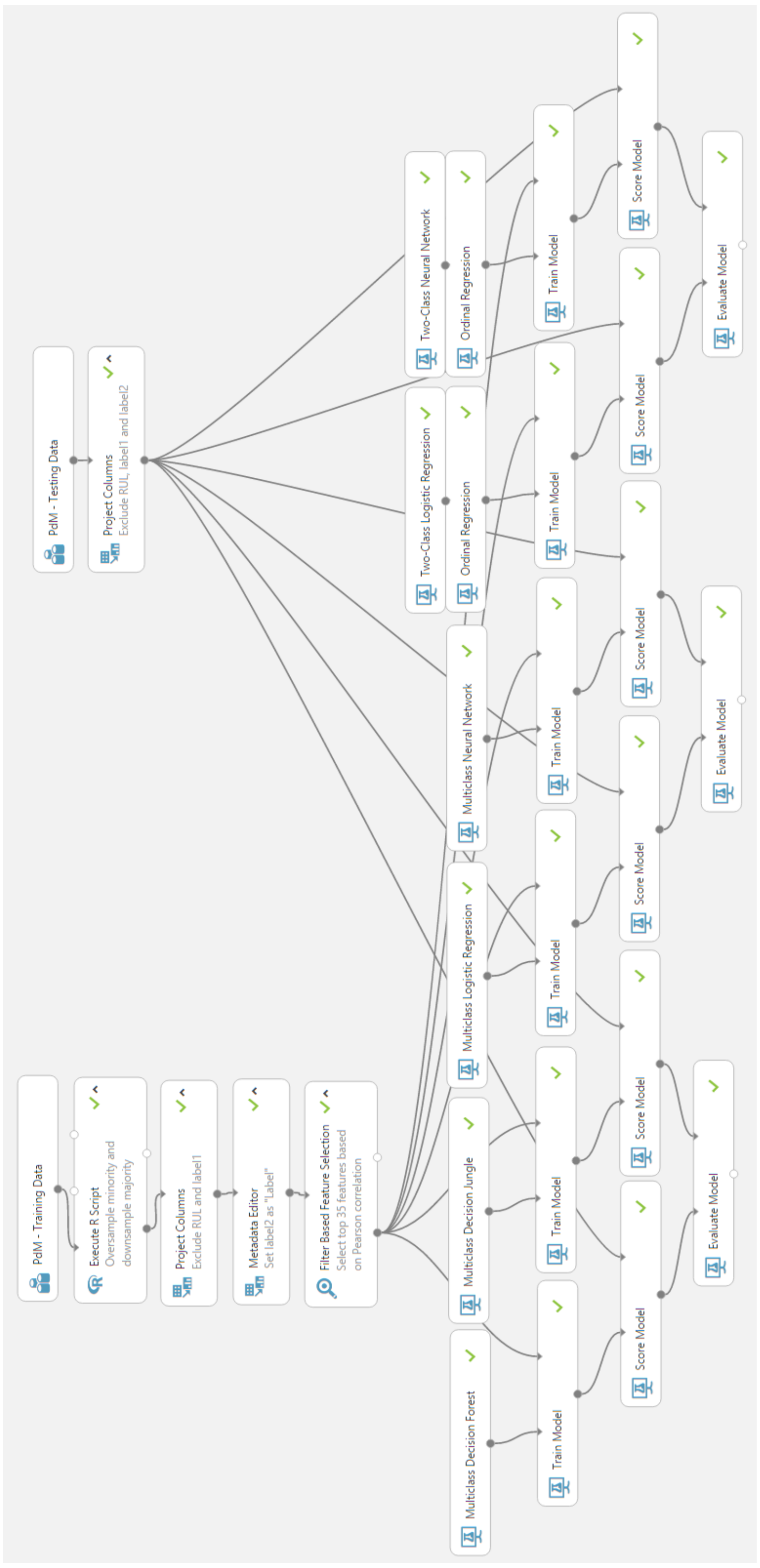

PdM - Testing Data
Project Columns
Exclude RUL, label1 and label2
PdM - Training Data
Execute R Script
Oversample minority and downsample majority
Project Columns
Exclude RUL and label1
Metadata Editor
Set label2 as "Label"
Filter Based Feature Selection
Select top 35 features based on Pearson correlation
Two-Class Neural Network
Ordinal Regression
Two-Class Logistic Regression
Ordinal Regression
Multiclass Neural Network
Multiclass Logistic Regression
Multiclass Decision Jungle
Multiclass Decision Forest
Train Model
Score Model
Evaluate Model

Figure 49. Azure ML Experiment: Multiclass Classification

4.3.Deployment as a Web Service

Microsoft Azure ML studio deploys an experiment as a RRS (Request-Response Service) which can be then consumed and used in a various ways. Standard solutions which are easy to integrate with would include Excel and Power BI dashboard. However, with a bit of additional effort, service can be used by websites or custom application. Both either desktop or mobile.

We have worked with three different models: regression, binary classification and multiclass classification. Therefore, we need to also make three separate deployments, one for each of the model used.

Deploying an experiment is a fairly easy process from the user perspective. Studio does virtually all the work required. Deployment can start once user is satisfied with the results achieved and with one requirement that latest run of the experiment must produce no errors. Deployment process starts by clicking "SET UP WEB SERVICE" and then "Predictive Web Service […]". It will generate predictive experiment based on training one.

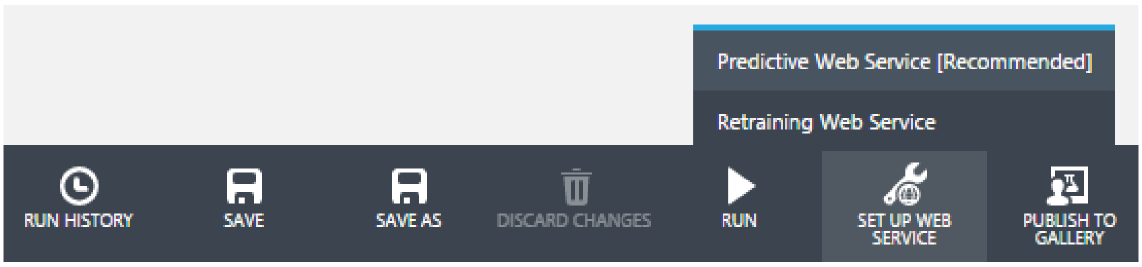


Figure 50. Azure ML Studio: Set Up Web Service

The next step is to set web service input(s). Its location effects on data that needs to be provided to the service in order for it to make a prediction. Two usual locations are:

- Before the data pre-processing steps, this location of the service input is usually used with batch scoring.
- After the data pre-processing steps, usually used to score the single row.

Once input is set, newly created prediction experiment needs to be run. Assuming no errors, all is left is to click "DEPLOY WEB SERVICE".

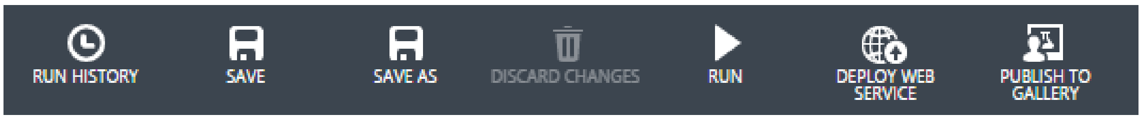


Figure 51. Azure ML Studio: Deploy Web Service

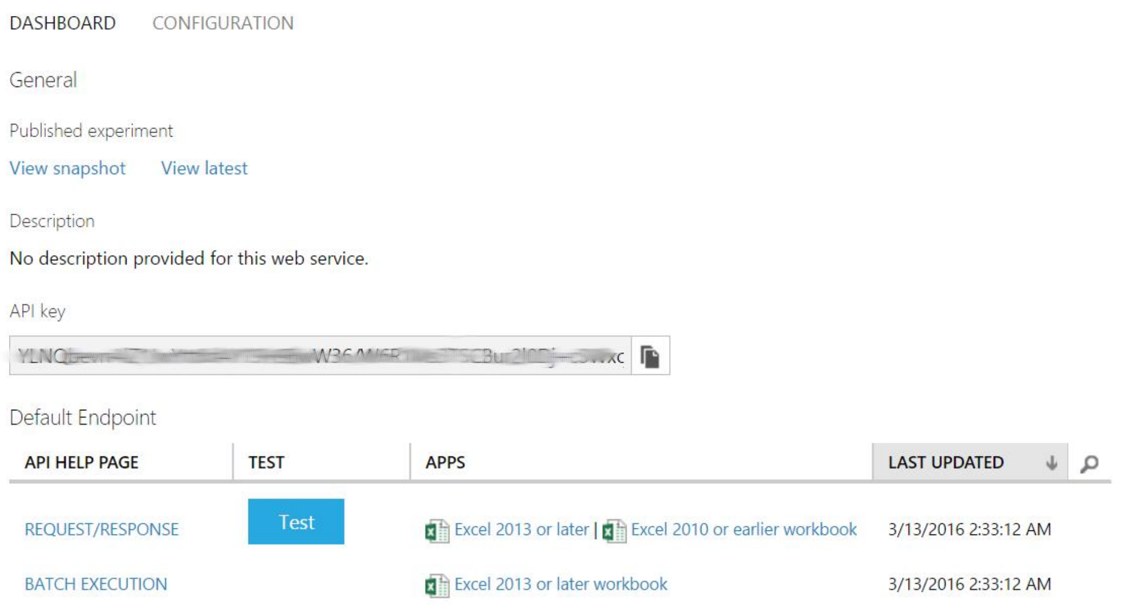
After deployment is done, studio switches view to the web service dashboard (**Figure** 52). From there one can easily test new service using:

- Build-in service test form available directly from the dashboard through the "Test" button (Figure 53).
- Download and use available Microsoft Excel files for single and batch predictions (Figure 54).

Additionally, with bit more of effort one can still relatively easy:

- Create Azure ML Request-Response Service Web App based on template available in Azure Web App Marketplace (**Figure** 55).

- Create custom application using code templates provided for C#, R and Python.



DASHBOARD CONFIGURATION

General

Published experiment

View snapshot View latest

Description

No description provided for this web service.

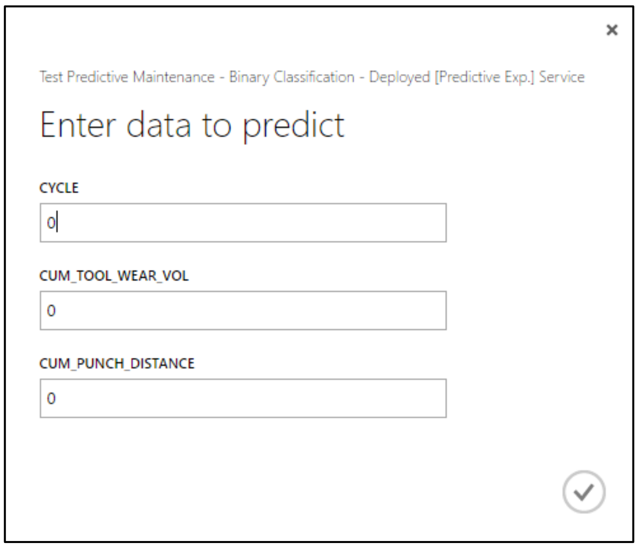
API key

[redacted]

Default Endpoint

API HELP PAGE	TEST	APPS	LAST UPDATED
REQUEST/RESPONSE	Test	Excel 2013 or later | Excel 2010 or earlier workbook	3/13/2016 2:33:12 AM
BATCH EXECUTION		Excel 2013 or later workbook	3/13/2016 2:33:12 AM

Figure 52. Azure ML Studio: Web Service Dashboard



Test Predictive Maintenance - Binary Classification - Deployed [Predictive Exp.] Service

Enter data to predict

CYCLE

0

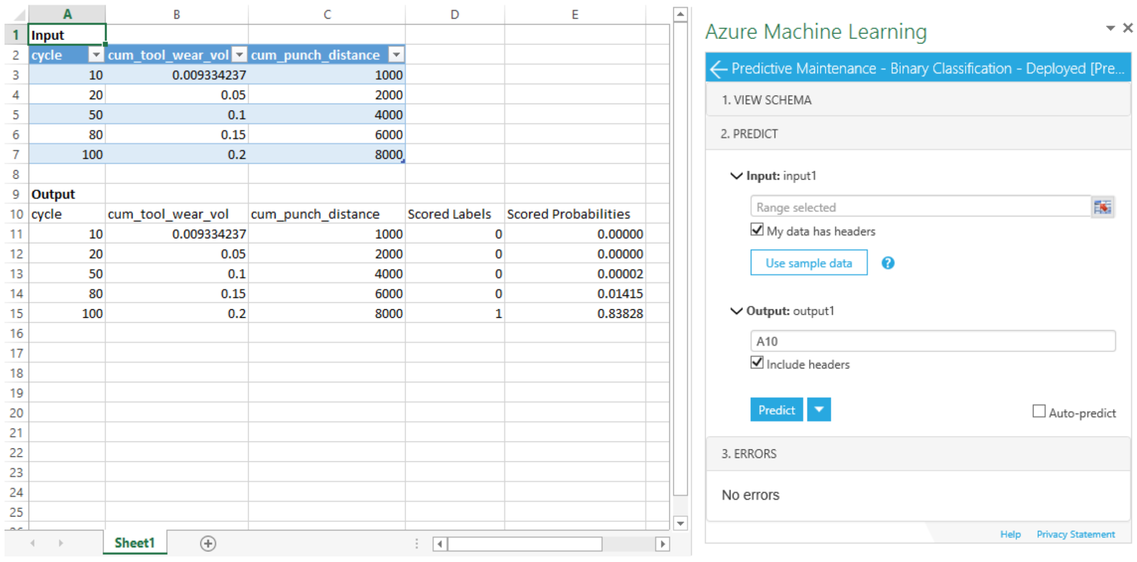
CUM_TOOL_WEAR_VOL

0

CUM_PUNCH_DISTANCE

0

Figure 53. Azure ML Studio: Build-in Service Test Form



	A	B	C	D	E
1	Input				
2	cycle	cum_tool_wear_vol	cum_punch_distance		
3	10	0.009334237	1000		
4	20	0.05	2000		
5	50	0.1	4000		
6	80	0.15	6000		
7	100	0.2	8000		
8					
9	Output				
10	cycle	cum_tool_wear_vol	cum_punch_distance	Scored Labels	Scored Probabilities
11	10	0.009334237	1000	0	0.00000
12	20	0.05	2000	0	0.00000
13	50	0.1	4000	0	0.00002
14	80	0.15	6000	0	0.01415
15	100	0.2	8000	1	0.83828

Sheet1

Azure Machine Learning

Predictive Maintenance - Binary Classification - Deployed [Pre...

1. VIEW SCHEMA

2. PREDICT

Input: input1

Range selected

☑ My data has headers

Use sample data

Output: output1

A10

☑ Include headers

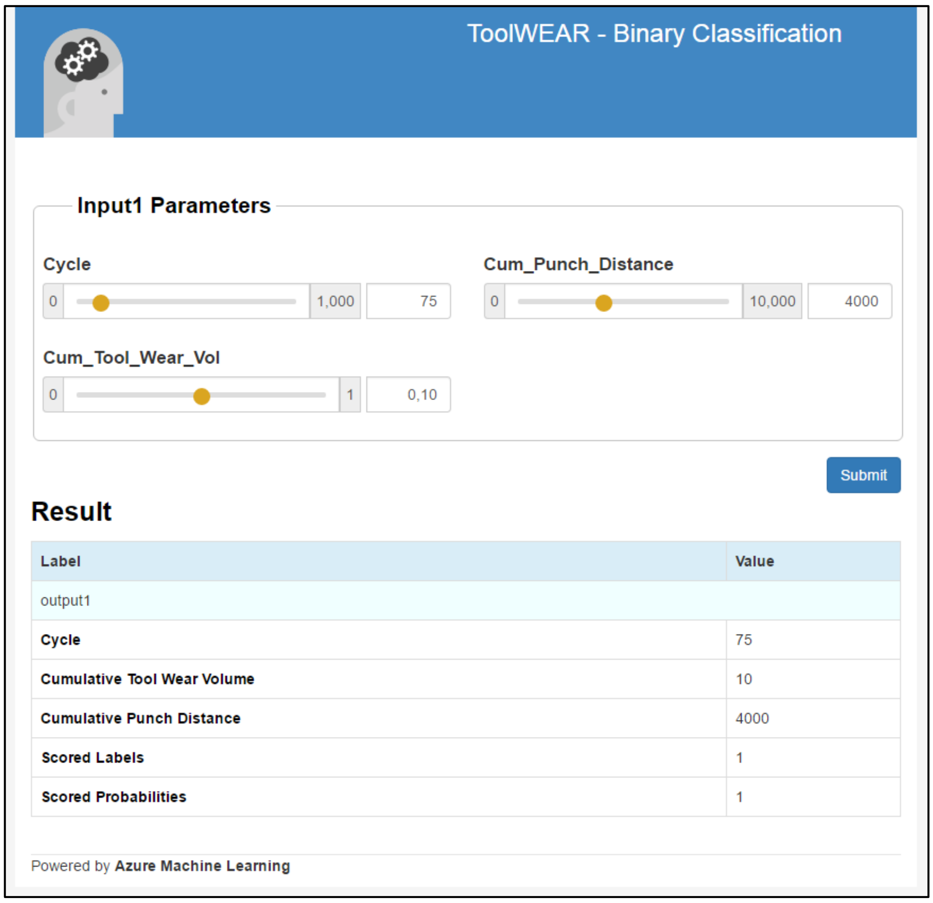
Predict ☐ Auto-predict

3. ERRORS

No errors

Help Privacy Statement

Figure 54. Microsoft Excel: Testing Deployed Predictive Web Service



ToolWEAR - Binary Classification

Input1 Parameters

Cycle: 0 – 1,000, 75

Cum_Punch_Distance: 0 – 10,000, 4000

Cum_Tool_Wear_Vol: 0 – 1, 0,10

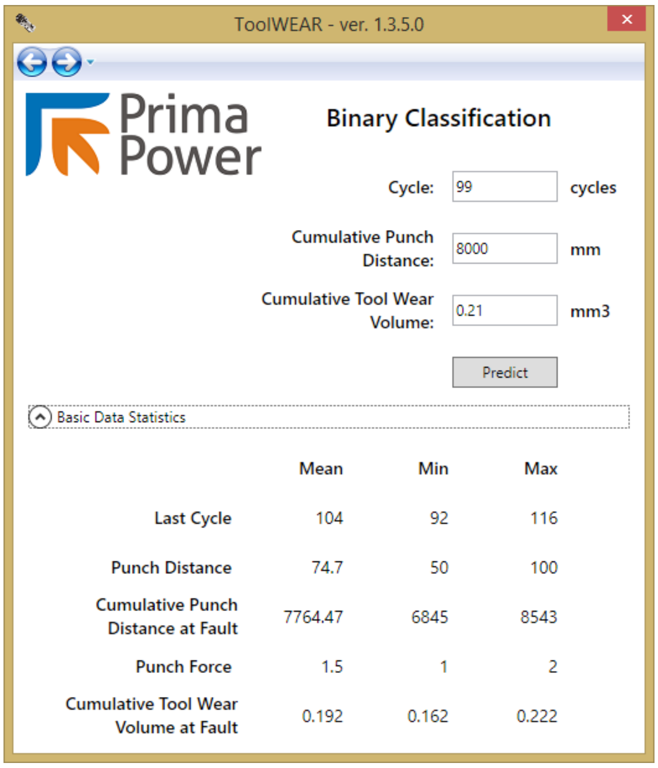
Submit

Result

Label	Value
output1	
Cycle	75
Cumulative Tool Wear Volume	10
Cumulative Punch Distance	4000
Scored Labels	1
Scored Probabilities	1

Powered by **Azure Machine Learning**

Figure 55. Web App Deployed on Azure


ToolWEAR - ver. 1.3.5.0

Binary Classification

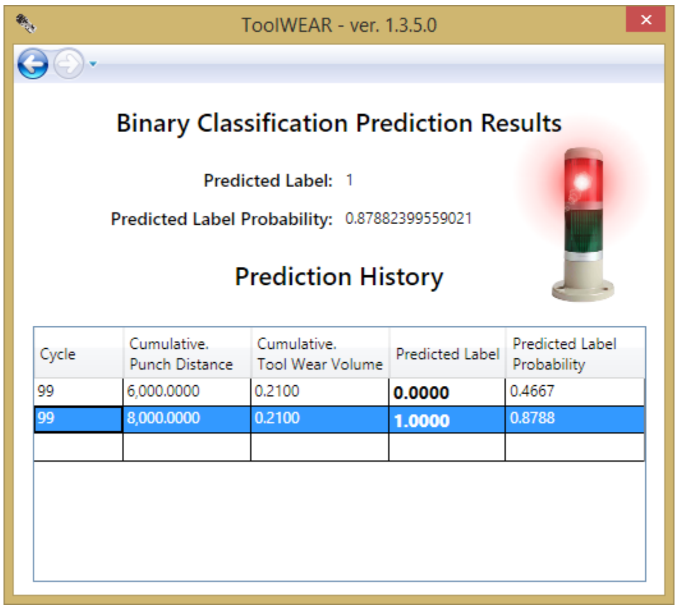
Cycle:	99	cycles
Cumulative Punch Distance:	8000	mm
Cumulative Tool Wear Volume:	0.21	mm3

Predict

Basic Data Statistics

	Mean	Min	Max
Last Cycle	104	92	116
Punch Distance	74.7	50	100
Cumulative Punch Distance at Fault	7764.47	6845	8543
Punch Force	1.5	1	2
Cumulative Tool Wear Volume at Fault	0.192	0.162	0.222

Figure 56. Custom .NET App: Data Input


ToolWEAR - ver. 1.3.5.0

Binary Classification Prediction Results

Predicted Label: 1

Predicted Label Probability: 0.87882399559021

Prediction History

Cycle	Cumulative. Punch Distance	Cumulative. Tool Wear Volume	Predicted Label	Predicted Label Probability
99	6,000.0000	0.2100	**0.0000**	0.4667
99	8,000.0000	0.2100	**1.0000**	0.8788

Figure 57. Custom .NET App: Results

4.4. Alternative Solutions Consideration

Microsoft Azure ML was used in the scope of this thesis. It came as the requirement at the beginning of the project. It is beyond the scope to make any thorough evaluation of other possibilities. However, short introduction of alternative technologies and solutions is provided here for a curious reader.

Microsoft Azure Machine Learning is not the only machine learning product to be offered as the MLaaS (Machine-Learning-as-a-Service). Some of its major large scale competitors who are also offering cloud-based machine learning solutions are:

- Google Prediction API (Google n.d.)
- Amazon Machine Learning (Amazon n.d.)
- IBM Watson Analytics (IBM n.d.)

Few smaller scale and sometimes more specialized providers of the MLaaS are listed below:

- Algorithms.io – streaming data
- BigML – focus on ease of use
- Datoin – big data text analytics
- Wise.io – customer service

Additionally, there are solutions which can be run both on premises and in the cloud. Here are mentioned two of them: Cloudera Enterprise and Microsoft R Server.

Cloudera Enterprise built on Apache Hadoop and related technologies. It uses HDFS file system to store and analyse vast amount of data using commodity hardware. Fast integration between Hadoop and other systems is achieved with Apache Sqoop, Apache Flume and Apache Kafka. First for bulk load processing and latter two for streaming. Data can be accessed and transformed using Apache Hive, Apache Pig, Map Reduce version 2 (MR2) and Apache Spark. Analysts can then work on data discovery using Apache Impala and Apache Solr. Finally machine learning tasks can be performed using Apache Spark MLlib. Picture below depicts components of the Cloudera Enterprise. (Cloudera n.d.)

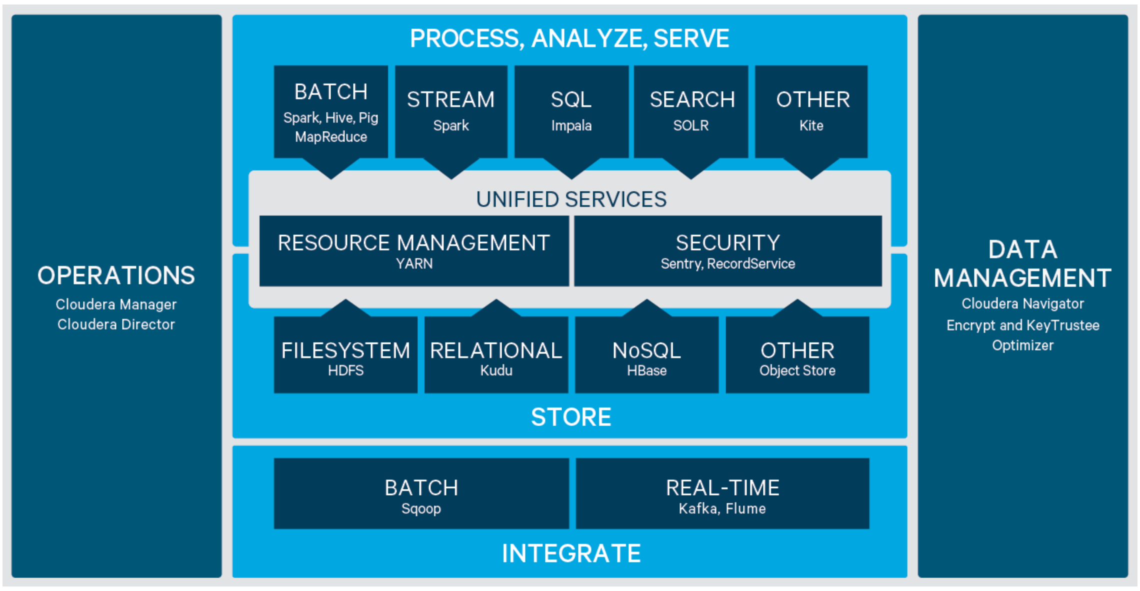


Figure 58. Cloudera Enterprise Architecture (Cloudera n.d.)

R is commonly used by data scientist to perform analytics and modelling tasks. It has vast amount of packages freely available and community of millions of users. R is powerful but its open source version is not big data ready as it is single threaded and memory bounded. Data with which open source R works needs to fit into machine's RAM. Microsoft R Server is an enterprise ready solution for the before mentioned shortcomings of the open source version. It scales and runs much faster. It is available on Windows and various Linux platforms, including Cloudera CDH. Microsoft R Server brings advanced analytics next to data, reducing the need for data movement. Microsoft R Server architecture is presented on the picture below. (Microsoft 2016)

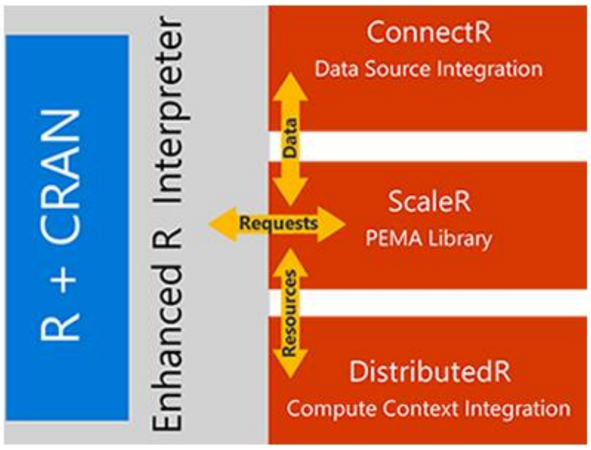


Figure 59. Microsoft R Server Architecture (Microsoft 2016)

Also worth of checking:

- DeployR
- Domino Data Lab
- OpenML
- Oracle R Enterprise
- TensorFlow
- WSO2 Machine Learner
- yhat

5. Results

In this chapter we are assessing if the outcome models produce results that are in line with the business goal of the project. Only in that perspective it can be said if results are good or not. Simply reporting statistics which are not clearly associated with the business objective is wrong. (Provost & Fawcett 2013)

Generally speaking, answering to the questions "Am I getting good results?" and "Which machine learning algorithm I should choose?" is not easy. There are no simple rules to follow and answer depends on the business objective of the problem and how one plans to utilize the answer given. We cannot just set certain accuracy threshold level to distinguish between good and bad results. Prediction with the accuracy of 99 percent is not always good. And even if the accuracy is good for the given problem, it can be that we are getting results too late. Meaning we are getting very accurate result at the moment when it does not have any value anymore. Therefore, sometimes one may decide to trade accuracy over the time. Consequently, it is virtually impossible to choose best algorithm without trial and careful results consideration. (Rohrer 2016)

5.1.Regression Models

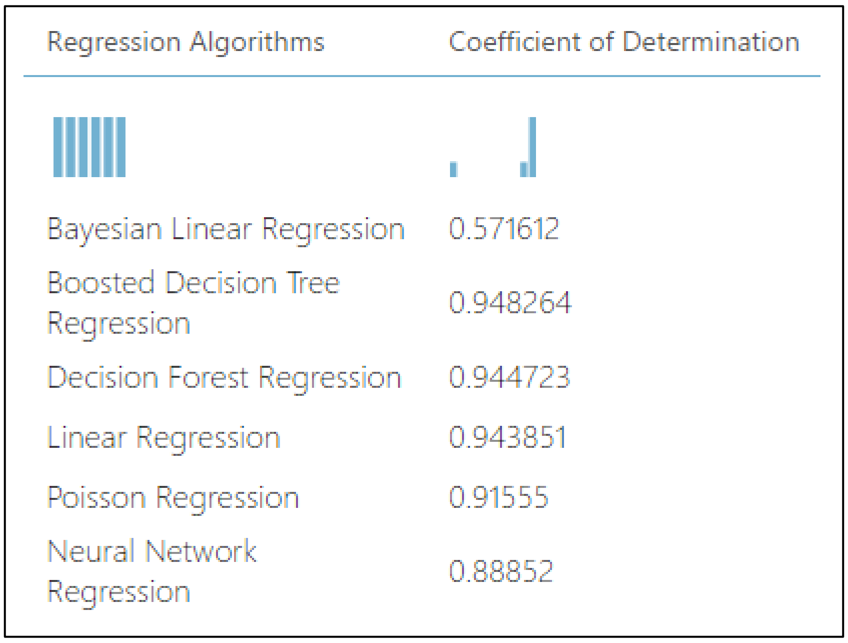
Following regressions were evaluated:

- Remaining useful lifetime measured in tool wear
- Remaining useful lifetime measured in cycles
- Remaining useful lifetime measured in distance

Figures below provide results for above mentioned regressions. For each regression six algorithms are evaluated based on theirs coefficient of determination (R^2). Simply stating, algorithm with the coefficient closest to one does best in this comparison. Acknowledging that other factors should also be taken into consideration in real life example, e.g. simplicity of the algorithm. However, we are focusing here solely on the R^2.

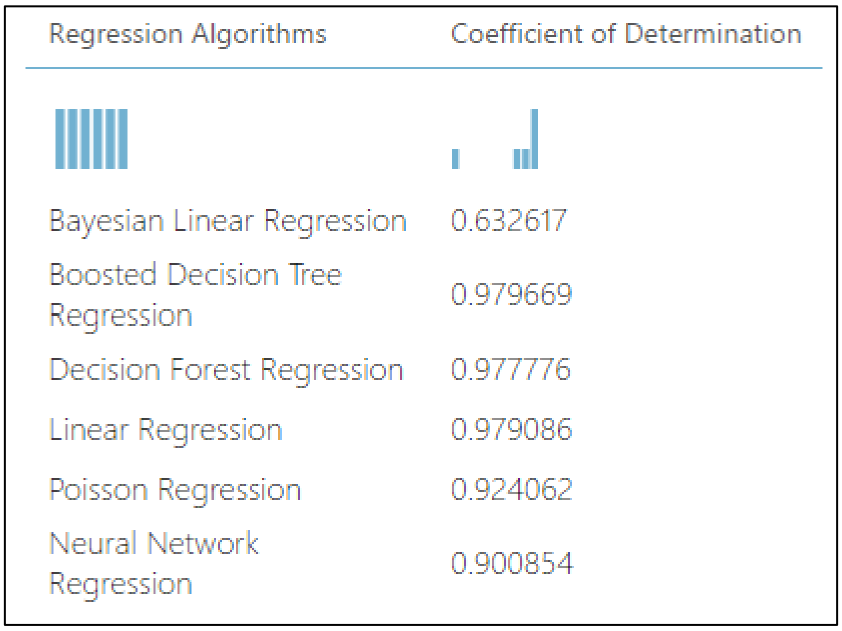
From the results presented below we can draw following conclusions:

- Performance of the Bayesian Linear and Neural Network regression algorithms is significantly worse than of remaining ones.
- In all three cases Boosted Decision Tree Regression performed best. However, difference with next contenders is not very significant. Simple Linear Regression performed almost as well.
- Neural Network Regression performed very poorly in the case of the remaining useful lifetime measured in distance. Coefficient of determination value below 0 is not common but it is correct. It means that fitted model does fit worse than horizontal straight line at mean (the null hypothesis).



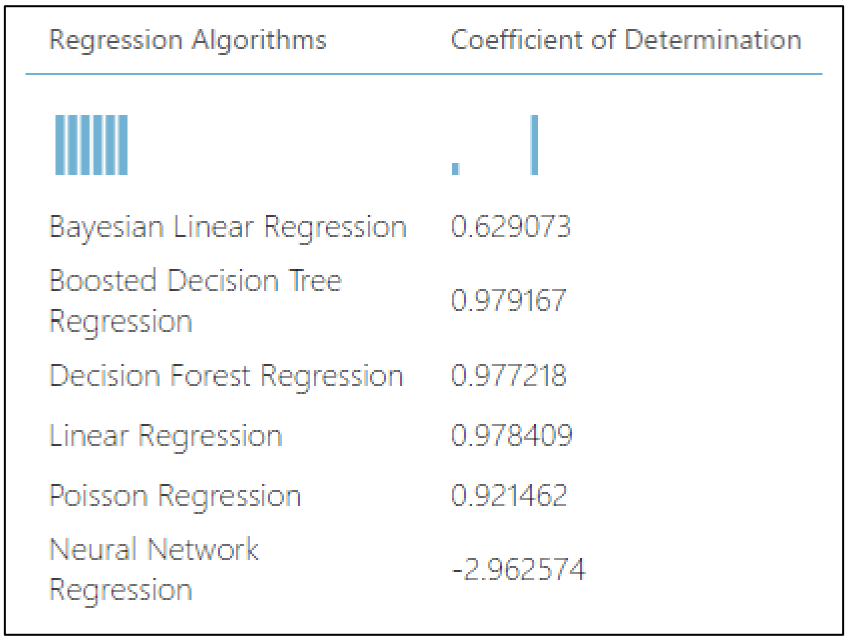
Regression Algorithms	Coefficient of Determination
Bayesian Linear Regression	0.571612
Boosted Decision Tree Regression	0.948264
Decision Forest Regression	0.944723
Linear Regression	0.943851
Poisson Regression	0.91555
Neural Network Regression	0.88852

Figure 60. Linear Regression Results (RUL_wear)



Regression Algorithms	Coefficient of Determination
Bayesian Linear Regression	0.632617
Boosted Decision Tree Regression	0.979669
Decision Forest Regression	0.977776
Linear Regression	0.979086
Poisson Regression	0.924062
Neural Network Regression	0.900854

Figure 61. Linear Regression Results (RUL_cycle)



Regression Algorithms	Coefficient of Determination
Bayesian Linear Regression	0.629073
Boosted Decision Tree Regression	0.979167
Decision Forest Regression	0.977218
Linear Regression	0.978409
Poisson Regression	0.921462
Neural Network Regression	-2.962574

Figure 62. Linear Regression Results (RUL_distance)

5.2.Binary Classification Models

Precision, recall and F1-score are used here as an evaluation metrics. All are explained in more depth earlier. Nevertheless, short explanation is provided below:

- Precision is a ratio of true positive predictions to the total number of positive predictions made (true positives + false positives). It shows how trustworthy is model when it identifies instances as a positive one. High precision means that if model label instance as a positive class then it is most likely truly is. However, it does not mean that it is good at picking positive classes from the sample. Stating otherwise, model with high precision keeps its false positives low.
- Recall is a ratio of true positive predictions to the total number of positive class instances in a given data set (true positives + false

negatives). It shows how well model is capable of picking positive classes. However, it may be at the cost of many false positives.

- F1-Score is a ratio of the product of precision and recall to the sum of both of them ($F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$). From the above descriptions of precision and recall it can be noticed that each of them puts emphasize on some aspect of prediction correctness. However, in both cases good results can be achieved with not necessary best models. Therefore, F1-Score provides single value that balance between those two.

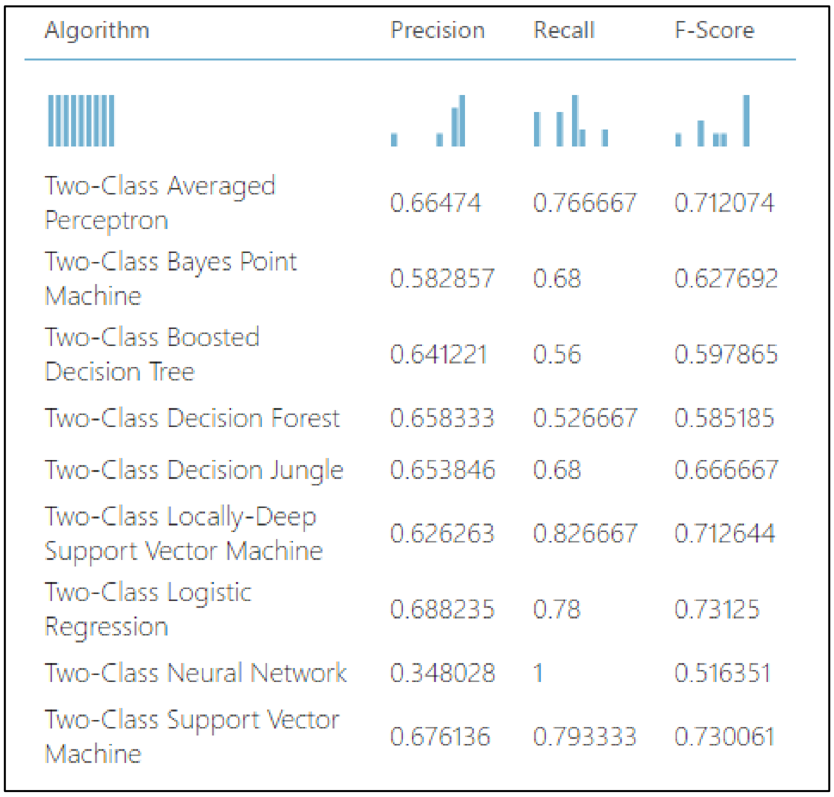
Accuracy is intentionally omitted as it doesn't work well with imbalanced classes (accuracy paradox). Consequently, relying on above mentioned metrics we can summarize results presented below (**Figure *63***) as follow:

- In the terms of precision, examined models give similar results. Except to Two-Class Neural Network which underperforms. Nevertheless, best precision is given by the Two-Class Logistic Regression.
- Differences are more noticeable with the recall. Interestingly the model with worst performance in precision is the best in the terms of the recall measure. It scored perfectly with the value of 1. Meaning that it was able to pick all positive instances of the class from the given data set.
- F1-Score suggest that Two-Class Logistic Regression or Two-Class Support Vector Machine should be selected.

Dependably on the business problem, following can be considered:

- Two-Class Neural Network can be selected if the goal is to identify as many positive class instances as possible even if it comes at the cost of significant amount of false positives.

- Two-Class Logistic Regression does best if precision and F1-score are considered. Meaning it balances well between picking as many as possible of positive class instances from the data set and keeping false positives reasonable low. Model should be selected if it is decided that Two-Class Neural Network precision is too low. Meaning costs arising from false positives are overcoming benefits of true positives.

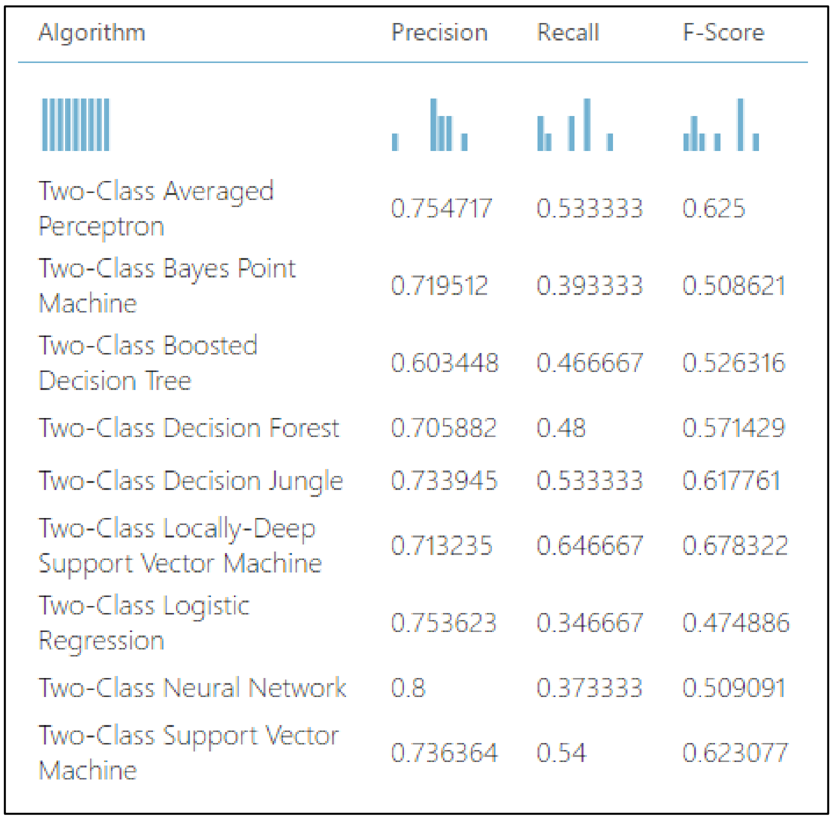


Algorithm	Precision	Recall	F-Score
Two-Class Averaged Perceptron	0.66474	0.766667	0.712074
Two-Class Bayes Point Machine	0.582857	0.68	0.627692
Two-Class Boosted Decision Tree	0.641221	0.56	0.597865
Two-Class Decision Forest	0.658333	0.526667	0.585185
Two-Class Decision Jungle	0.653846	0.68	0.666667
Two-Class Locally-Deep Support Vector Machine	0.626263	0.826667	0.712644
Two-Class Logistic Regression	0.688235	0.78	0.73125
Two-Class Neural Network	0.348028	1	0.516351
Two-Class Support Vector Machine	0.676136	0.793333	0.730061

Figure 63. Binary Classification Results with SMOTE

Previous results are realized with SMOTE module which addresses class imbalance issue. To demonstrate its impact, below results (**Figure *64***) are retrieved from the same experiment with only exception to SMOTE module not being used this time. Following difference can be noticed:

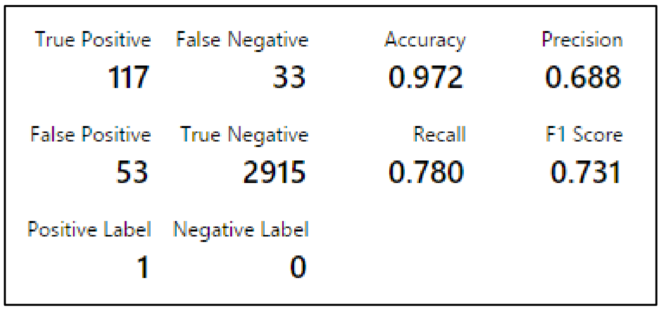
- SMOTE seems to have negative impact on the precision metric. Almost all algorithm, except to Two-Class Boosted Decision Tree, are having lower precision with SMOTE.
- It has significant positive impact on recall. It is evident for all algorithms evaluated.
- Despite its negative impact on precision, usage of SMOTE module improved F1-score values. Meaning its slight negative impact on precision is covered by positive impact on recall metric.



Algorithm	Precision	Recall	F-Score
Two-Class Averaged Perceptron	0.754717	0.533333	0.625
Two-Class Bayes Point Machine	0.719512	0.393333	0.508621
Two-Class Boosted Decision Tree	0.603448	0.466667	0.526316
Two-Class Decision Forest	0.705882	0.48	0.571429
Two-Class Decision Jungle	0.733945	0.533333	0.617761
Two-Class Locally-Deep Support Vector Machine	0.713235	0.646667	0.678322
Two-Class Logistic Regression	0.753623	0.346667	0.474886
Two-Class Neural Network	0.8	0.373333	0.509091
Two-Class Support Vector Machine	0.736364	0.54	0.623077

Figure 64. Binary Classification Results without SMOTE

Additionally to the metrics presented, it is very useful to evaluate classifier performance using the confusion matrix. **Figure 65** presents confusion matrix of the Two-Class Logistic Regression. Also examining ROC (**Figure** *66*), Precision/Recall (**Figure** *67*) and Lift (**Figure** *68*) curves can give better insight into model performance.



True Positive	False Negative	Accuracy	Precision
117	33	0.972	0.688
False Positive	**True Negative**	**Recall**	**F1 Score**
53	2915	0.780	0.731
Positive Label	**Negative Label**		
1	0		

Figure 65. Confusion Matrix: Two-Class Logistic Regression

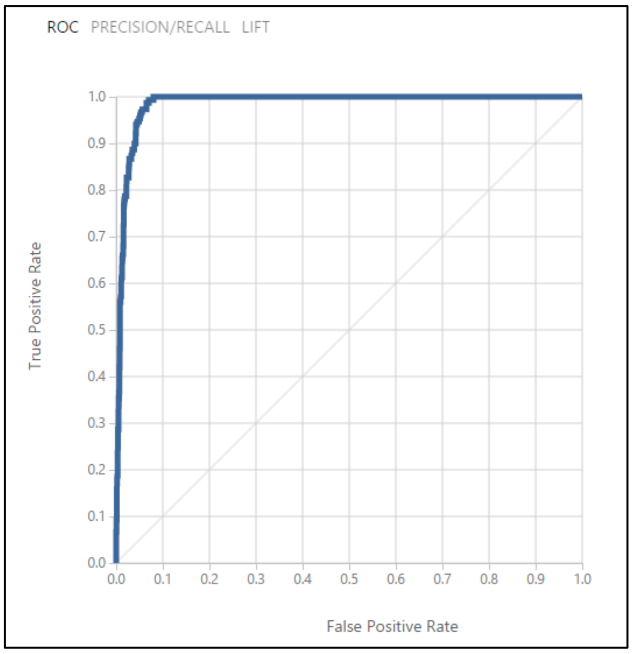


Figure 66. ROC Curve: Two-Class Logistic Regression

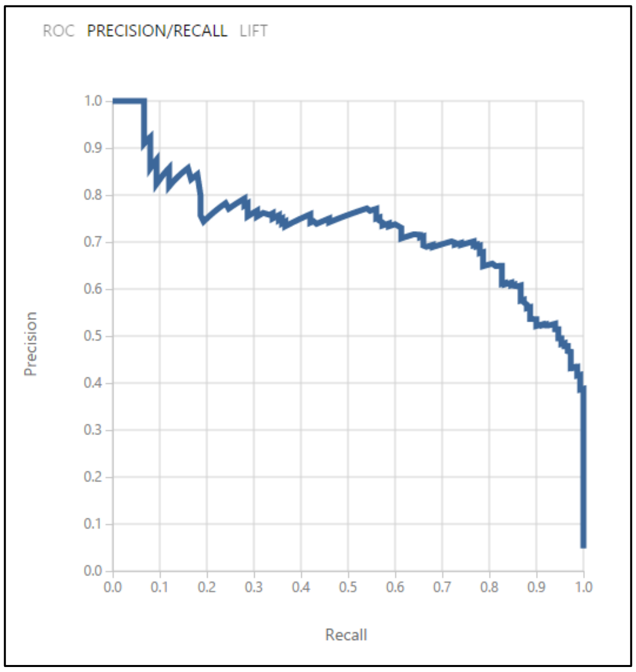


Figure 67. Precision / Recall Curve: Two-Class Logistic Regression

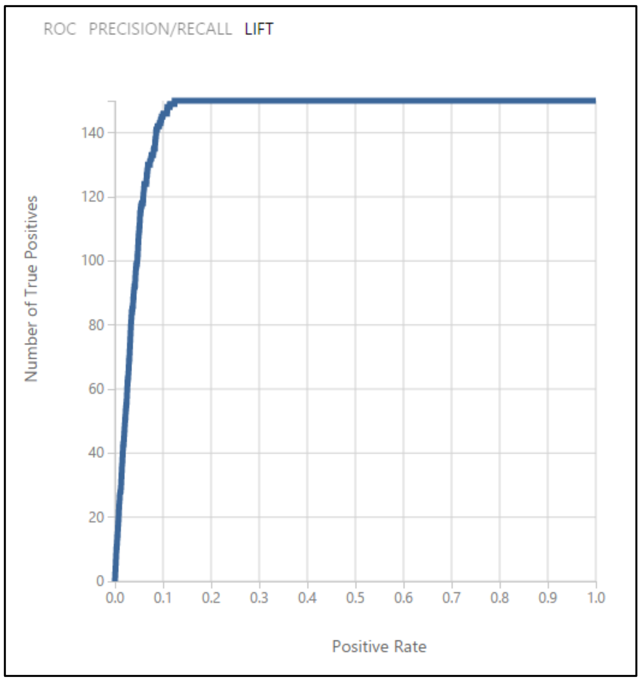


Figure 68. Lift Curve: Two-Class Logistic Regression

Following table reflect on metrics from the business perspective. It provides guideline on main metric of focus when class distribution and costs of false negatives and false positives are taken into consideration.

		Class Distribution	
		Even	Uneven
Cost	FN cost more	Recall	Recall
	Same cost	Accuracy or F1 Score	F1 Score
	FP cost more	Precision	Precision

Figure 69. Metric Selection Guideline Table (De Ruiter 2015)

5.3.Multiclass Classification Models

Evaluating results of the multiclass classifiers is bit more difficult than it was for binary ones. To evaluate each model following should be checked:

- Confusion matrix
- Class specific as well as macro-averaged precision and recall

Micro-averaged versions of the precision and recall are calculated by summing true positives, false positives and false negatives from classes and calculating precision and recall on those sums.

$$Micro - averaged\ precision = \frac{TPs}{(TPs + FPs)} \qquad (21)$$

$$Micro - averaged\ recall = \frac{TPs}{(TPs + FNs)} \qquad (22)$$

Macro-averages versions of the precision and recall are calculated as an average of precision and recall of individual classes.

$$Macro - averaged\ precision = \frac{Precision_0 + \ Precision_1 + Precision_2}{3} \tag{23}$$

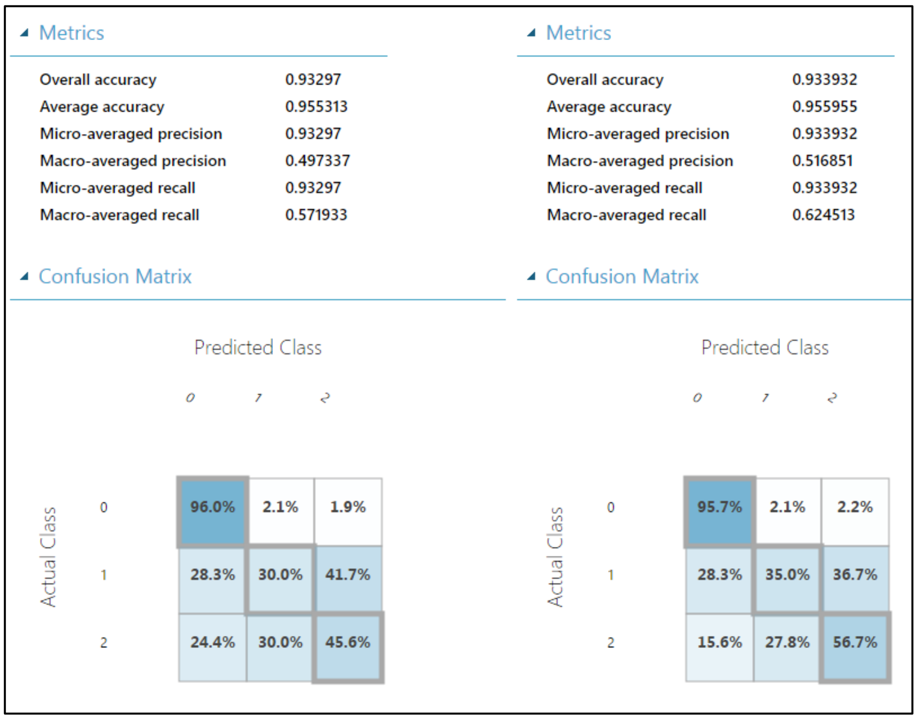
$$Macro - averaged\ recall = \frac{Recall_0 + \ Recall_1 + Recall_2}{3} \tag{24}$$

Micro-average gives equal weights to all classifications. Thus, with this method big class will over dominate the smaller ones. It is the case with the class "0". Micro-averaged values are high even so precision and recall for classes "1" and "2" are relatively low.

Macro-average in contrary gives equal weight to all classes. Consequently, it is more appropriate to use with imbalanced class distribution.

From the results we can conclude following:

- All models are quite poor in correctly predicting class "1". It can be noticed by examination of the confusion matrixes. In fact, Multiclass Logistic Regression have not predicted even single instance as belonging to that class.
- From the business perspective misclassification happening between classes "1" and "2" is not as serious as misclassifying any of them as class "0". Therefore, from that perspective Multiclass Logistic Regression does best. Only 15% of actual class "1" and 6.7% of actual class "2" instances are misclassified as class "0". That is significantly better than remaining models. It does also best in macro-averaged recall.



Metrics

Overall accuracy	0.93297
Average accuracy	0.955313
Micro-averaged precision	0.93297
Macro-averaged precision	0.497337
Micro-averaged recall	0.93297
Macro-averaged recall	0.571933

Confusion Matrix

Predicted Class

Actual Class	0	1	2
0	96.0%	2.1%	1.9%
1	28.3%	30.0%	41.7%
2	24.4%	30.0%	45.6%

Metrics

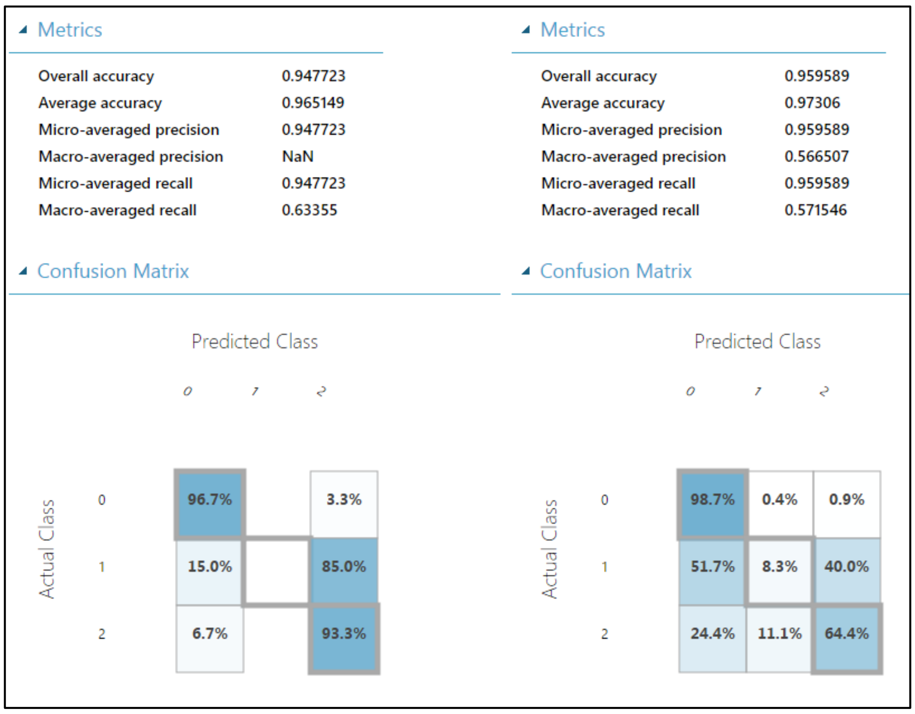
Overall accuracy	0.933932
Average accuracy	0.955955
Micro-averaged precision	0.933932
Macro-averaged precision	0.516851
Micro-averaged recall	0.933932
Macro-averaged recall	0.624513

Confusion Matrix

Predicted Class

Actual Class	0	1	2
0	95.7%	2.1%	2.2%
1	28.3%	35.0%	36.7%
2	15.6%	27.8%	56.7%

Figure 70. Evaluation Results for Multiclass Decision Forest and Multiclass Decision Jungle



Metrics

Overall accuracy	0.947723
Average accuracy	0.965149
Micro-averaged precision	0.947723
Macro-averaged precision	NaN
Micro-averaged recall	0.947723
Macro-averaged recall	0.63355

Confusion Matrix

Predicted Class

Actual Class	0	1	2
0	96.7%		3.3%
1	15.0%		85.0%
2	6.7%		93.3%

Metrics

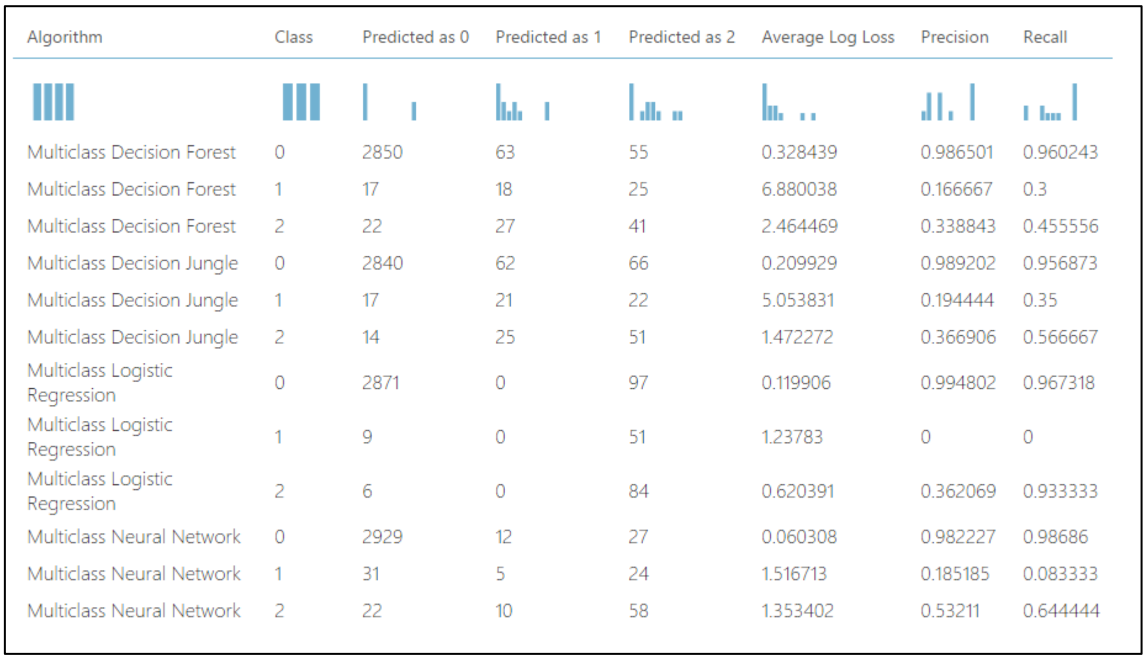
Overall accuracy	0.959589
Average accuracy	0.97306
Micro-averaged precision	0.959589
Macro-averaged precision	0.566507
Micro-averaged recall	0.959589
Macro-averaged recall	0.571546

Confusion Matrix

Predicted Class

Actual Class	0	1	2
0	98.7%	0.4%	0.9%
1	51.7%	8.3%	40.0%
2	24.4%	11.1%	64.4%

Figure 71. Evaluation Results for Multiclass Logistic Regression and Multiclass Neural Network

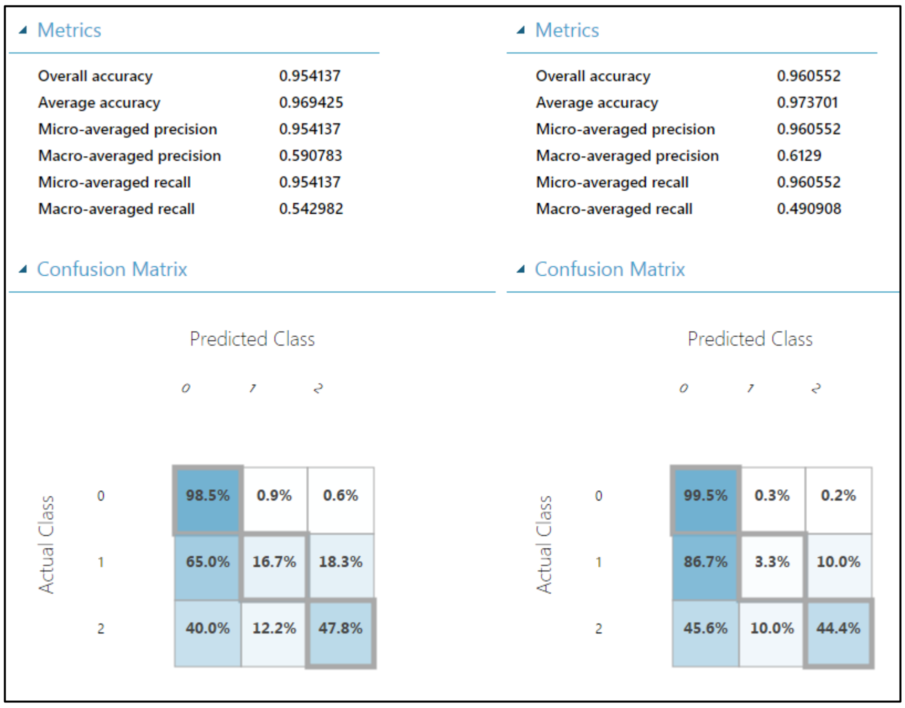


Algorithm	Class	Predicted as 0	Predicted as 1	Predicted as 2	Average Log Loss	Precision	Recall
Multiclass Decision Forest	0	2850	63	55	0.328439	0.986501	0.960243
Multiclass Decision Forest	1	17	18	25	6.880038	0.166667	0.3
Multiclass Decision Forest	2	22	27	41	2.464469	0.338843	0.455556
Multiclass Decision Jungle	0	2840	62	66	0.209929	0.989202	0.956873
Multiclass Decision Jungle	1	17	21	22	5.053831	0.194444	0.35
Multiclass Decision Jungle	2	14	25	51	1.472272	0.366906	0.566667
Multiclass Logistic Regression	0	2871	0	97	0.119906	0.994802	0.967318
Multiclass Logistic Regression	1	9	0	51	1.23783	0	0
Multiclass Logistic Regression	2	6	0	84	0.620391	0.362069	0.933333
Multiclass Neural Network	0	2929	12	27	0.060308	0.982227	0.98686
Multiclass Neural Network	1	31	5	24	1.516713	0.185185	0.083333
Multiclass Neural Network	2	22	10	58	1.353402	0.53211	0.644444

Figure 72. Confusion Matrix, Precision and Recall for Evaluated Multiclass Algorithms

For the comparison, results from the same experiment, except to oversampling and downsampling script not being executed, are shown below. It can be noticed that using the script contributed to:

- Increased true positives, especially for the class “2”.
- Significantly reduced misclassification of remaining classes as class “0”. It is evident for all models, but even more in the case of the Multiclass Logistic Regression.



Metrics

Overall accuracy	0.954137
Average accuracy	0.969425
Micro-averaged precision	0.954137
Macro-averaged precision	0.590783
Micro-averaged recall	0.954137
Macro-averaged recall	0.542982

Confusion Matrix

Actual Class \ Predicted Class	0	1	2
0	98.5%	0.9%	0.6%
1	65.0%	16.7%	18.3%
2	40.0%	12.2%	47.8%

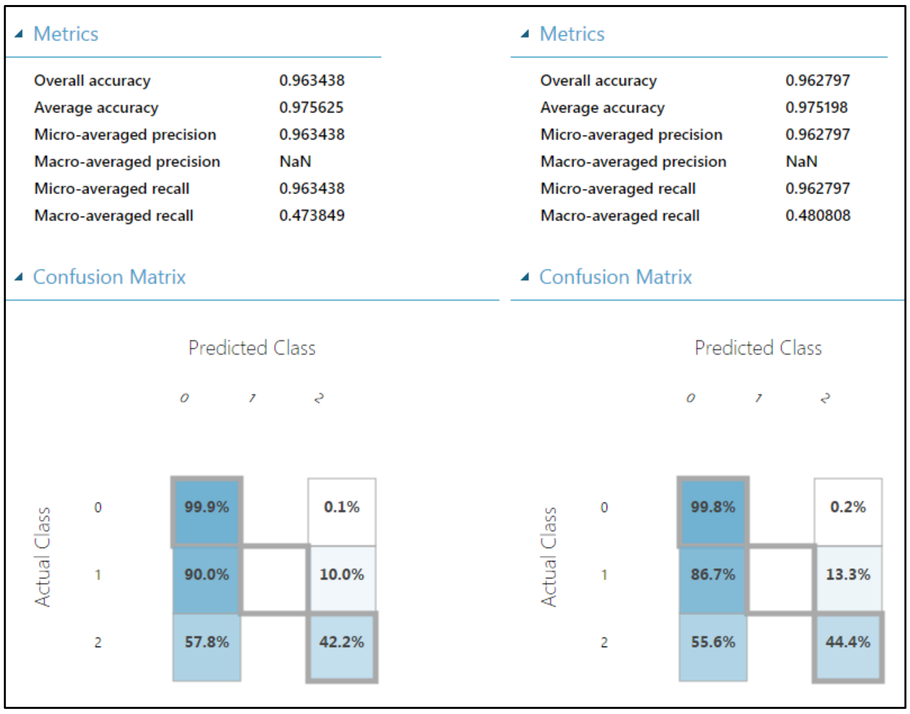
Metrics

Overall accuracy	0.960552
Average accuracy	0.973701
Micro-averaged precision	0.960552
Macro-averaged precision	0.6129
Micro-averaged recall	0.960552
Macro-averaged recall	0.490908

Confusion Matrix

Actual Class \ Predicted Class	0	1	2
0	99.5%	0.3%	0.2%
1	86.7%	3.3%	10.0%
2	45.6%	10.0%	44.4%

Figure 73. Evaluation Results for Multiclass Decision Forest and Multiclass Decision Jungle without over- and downsampling



Metrics

Overall accuracy	0.963438
Average accuracy	0.975625
Micro-averaged precision	0.963438
Macro-averaged precision	NaN
Micro-averaged recall	0.963438
Macro-averaged recall	0.473849

Confusion Matrix

Actual Class \ Predicted Class	0	1	2
0	99.9%		0.1%
1	90.0%		10.0%
2	57.8%		42.2%

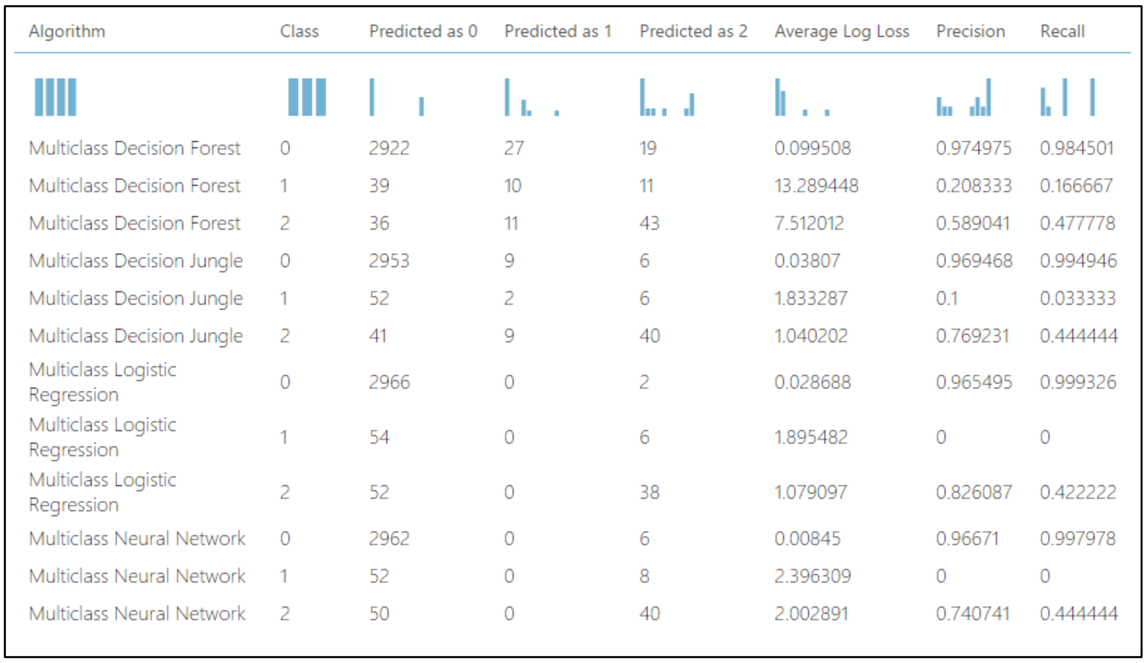
Metrics

Overall accuracy	0.962797
Average accuracy	0.975198
Micro-averaged precision	0.962797
Macro-averaged precision	NaN
Micro-averaged recall	0.962797
Macro-averaged recall	0.480808

Confusion Matrix

Actual Class \ Predicted Class	0	1	2
0	99.8%		0.2%
1	86.7%		13.3%
2	55.6%		44.4%

Figure 74. Evaluation Results for Multiclass Logistic Regression and Multiclass Neural Network without over- and downsampling



Algorithm	Class	Predicted as 0	Predicted as 1	Predicted as 2	Average Log Loss	Precision	Recall
Multiclass Decision Forest	0	2922	27	19	0.099508	0.974975	0.984501
Multiclass Decision Forest	1	39	10	11	13.289448	0.208333	0.166667
Multiclass Decision Forest	2	36	11	43	7.512012	0.589041	0.477778
Multiclass Decision Jungle	0	2953	9	6	0.03807	0.969468	0.994946
Multiclass Decision Jungle	1	52	2	6	1.833287	0.1	0.033333
Multiclass Decision Jungle	2	41	9	40	1.040202	0.769231	0.444444
Multiclass Logistic Regression	0	2966	0	2	0.028688	0.965495	0.999326
Multiclass Logistic Regression	1	54	0	6	1.895482	0	0
Multiclass Logistic Regression	2	52	0	38	1.079097	0.826087	0.422222
Multiclass Neural Network	0	2962	0	6	0.00845	0.96671	0.997978
Multiclass Neural Network	1	52	0	8	2.396309	0	0
Multiclass Neural Network	2	50	0	40	2.002891	0.740741	0.444444

Figure 75. Confusion Matrix, Precision and Recall for Evaluated Multiclass Algorithms without over- and downsampling

Additionally to the models presented above, two ordinal regression models were used as well. Following metrics are produced by the evaluate model:

- Mean Zero One Error
- Mean Absolute Error
- Root Mean Squared Error

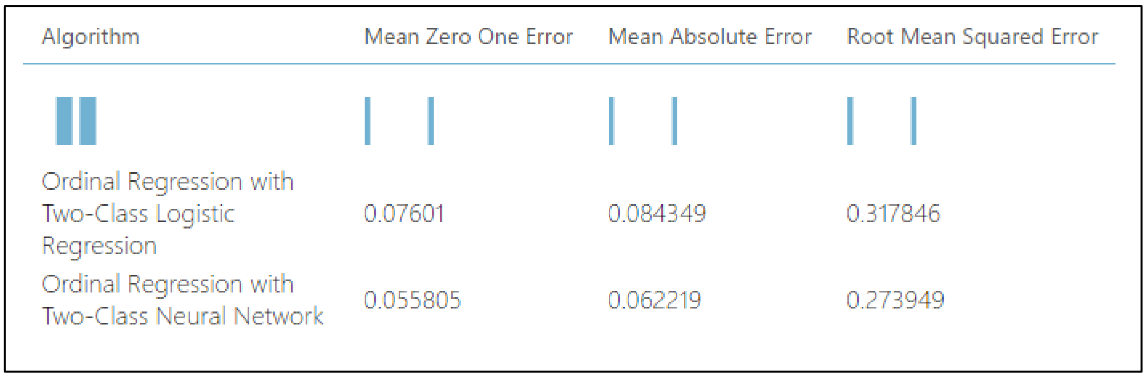
Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are well known metrics and are already explained in subchapter 2.5.6. However, Mean Zero One Error may require some additional clarification. It main difference from remaining two is that it does not measure the distance of the prediction from the ground truth. It just simply counts misclassified instances and divides it by total number of instances, according to the formula presented below:

$$Mean\ Zero\ One\ Error = \frac{1}{n}\sum_{i=1}^{n} ZeroOneLoss(\widehat{y_i}, y_i) \qquad (25)$$

Where,

$$ZeroOneLoss(\widehat{y_i}, y_i) = \begin{cases} 0\ if\ \widehat{y_i} = y_i \\ 1\ if\ \widehat{y_i} \neq y_i \end{cases} \qquad (26)$$

From the results of three metrics presented below, it can be concluded that Ordinal Regression module using the Two-Class Neural Network as the input performed better than Ordinal Regression with the Two-Class Logistic Regression being set as the input.



Algorithm	Mean Zero One Error	Mean Absolute Error	Root Mean Squared Error
Ordinal Regression with Two-Class Logistic Regression	0.07601	0.084349	0.317846
Ordinal Regression with Two-Class Neural Network	0.055805	0.062219	0.273949

Figure 76. Evaluation Results for Ordinal Regressions module using Two-Class Logistic Regression and Two-Class Neural Network.

6. Conclusions

The potential of machine learning was realized by the case company after initial technology introduction during the competition. However, many open questions were left and following research problem was established:

> *Can that bring any benefits for the sheet metal industry in a moderate time frame with a reasonable amount of effort needed?*

The research presented in this document focused on feasibility of machine learning in improving the competitive advantage of sheet metal fabrication machines. Special focus was on Microsoft Azure Machine Learning product. It was dictated by the case company's interest into this particular solution. However, despite empirical part, research was generalized and is not bounded by any specific machine learning implementation. The main research question that was driving the study was stated early as follow:

> *How can the sheet metal industry use machine learning for improving its operations management?*

Research has shown that sheet metal fabrication machines can benefit from the machine learning technology. It gave a concrete example in the form of the predictive maintenance implementation. Lack of real data was a slight drawback but it didn't prevent the research from reaching its goals.

Data science projects can be vague especially for some business stakeholders. Utilizing methodology which helped to provide structured approach to data mining project proved to bring many benefits. It guided research through what could otherwise become easily a chaotic exploration. Employing CRISP-DM helped also in communicating on project progress and status between researcher and main stakeholders.

In order to be capably derive valid models and evaluate them correctly, it was needed to recognize that data is imbalanced. That drove the need to use the SMOTE module and additionally some custom over- and undersampling scripts written in R. Without understanding of the issues arising from the imbalanced data, one could also wrongly evaluate resulting models and e.g. make his decision based solely on the accuracy. Which in this case is rather inadequate.

During the research it was also evident that models predictive power increased due to constructed features. Examination of the filter-based feature selection module results showed that there were two constructed features in top three ones. It was the case for both classifications evaluated.

It serves well as evidence to the importance of the data understanding and feature engineering stages. However, probably the most significant one is business understanding. It ensured that research wasn't just doing things right but more essentially the right thing. Good business understanding allowed also the author to make judgment calls which proved to be right but which were not obvious at the first for the case company.

Main limitations of the study were time and lack of real data. There was always drive to do more. However, time frame was set taking into consideration that thesis is given 30 ECTS credit points that equals roughly to 800 hours of work or otherwise to 1 academic semester. Real tool wear out data that would be collected from machines during operation was not available during the time of the research. This was mostly due to innovative nature of the project which was introducing new features rather than extending on already existing ones. Taking those limitations into consideration, scope and goals were set accordingly.

This research contributes to other research on reliability by examining and demonstrating how machine learning can be used for Predictive Maintenance in sheet metal industry. Empirical example gives special focus on the prediction of the tool wear out. Correct prediction allows to maximize resource utilization and reduces number of unplanned maintenance breakdowns. It demonstrates three possible means of addressing the prediction. For first, it shows how regression can be used to predict remaining useful lifetime. Binary and multiclass classifications demonstrated second and third approach respectively. Binary classification used single time frame and model was predicting probability of the fault at the current one. Multiclass classification showed that predictions can be also made for multiple time frames into the future.

The aim of the research that was successfully reached was proof of concept. Work should now continue towards its full productization. At the first, sample of the real data need to be collected. Research presented here was done based on generated data due to lack of real one. Full architecture design should follow it. It involves further feasibility studies for most appropriate data collection, storing, processing and presentation.

Another prospective research topic can focus on best business model to be used. Possibly value creation and capture should be reconsidered by the case company to best utilize on potentials of the Internet of Things and Machine Learning. Tool wear out prediction should become part of the bigger ecosystem.

The scope of this thesis focused on the Microsoft Azure Machine Learning. Evaluation of other Azure products comes as a natural continuation. Certainly the highest potential of Azure Machine Learning lies in its conceivable synergy with other Azure services. To demonstrate the versatility of Azure offerings, further research should focus on remaining Azure products, to list just few: Event Hub, Stream Analytics, Storage, Service Bus, Notification Hub, App Service, and Azure Service Fabric.

LIST OF REFERENCES

Aczel, A., Sounderpandian, J. (2008). Complete Business Statistics (7th ed.). McGraw-Hill Companies. 804 p. ISBN 978−0−39−050192−9

Amazon (n.d.). *Amazon Machine Learning* [online] Amazon Web Services: Amazon [cited 28 Mar. 2016]. Available from World Wide Web <URL: https://aws.amazon.com/machine-learning/>.

Azevedo, A., Santos, M.F. (2008, Jul. 22-27). KDD, SEMMA and CRISP-DM: A parallel overview. In H. Weghom, A.P. Abraham (Eds.). *Proceedings of the IADIS European Conference on Data Mining*. Paper presented at IADIS European Conference on Data Mining 2008, Amsterdam, The Netherlands. ISBN 978-972-8924-63-8

Bellin, D. (2014). *Predictive Maintenance: The Business Impact of IoE for Mining Companies* [online] Cisco Blogs: Cisco, 2014 [cited 21 Feb. 2016]. Available from World Wide Web <URL: http://blogs.cisco.com/digital/predictive-maintenance-the-business-impact-of-ioe-for-mining-companies>.

Bosch (2014). *Predictive Maintenance with the Service Portal. The Intelligent Way to Maximize Machine Availability* [online] Software Innovations: Bosch, 2014 [cited 22 Feb. 2016]. Available from World Wide Web <URL: https://www.bosch-si.com/media/en/bosch_software_innovations/documents/brochure/predictive_maintenance_1/brochure-predictive-maintenance-service-portal.pdf>.

Brownlee, J. (2013). *A Tour of Machine Learning Algorithms* [online] Machine Learning Mastery, 2013 [cited 1 Mar. 2016]. Available from World Wide Web <URL: http://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/>

Brownlee, J. (2014). *How To Estimate Model Accuracy in R Using The Caret Package* [online] Machine Learning Mastery, 2014 [cited 1 Mar. 2016]. Available from World Wide Web <URL: http://machinelearningmastery.com/how-to-estimate-model-accuracy-in-r-using-the-caret-package/>

Brynjolfsson, E., Hitt, L. M., & Kim, H. H. (2011). *Strength in numbers: How does data-driven decision making affect firm performance?* Tech. rep., available at SSRN: http://ssrn.com/abstract=1819486

Chai, T., Draxler, R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature. *Geoscientific Model Development*, 7(3), 1247-1250. DOI: 10.5194/gmd-7-1247-2014. Available from World Wide Web <URL: http://www.geosci-model-dev.net/7/1247/2014/>.

Cloudera (n.d.). *Cloudera Enterprise* [online] Products: Cloudera [cited 28 Mar. 2016]. Available from World Wide Web <URL: https://www.cloudera.com/products.html>.

Coraddu, A., Oneto, L., Ghio, A., Savio, S., Figari, M., Anguita, D. (2015). *Machine learning for wear forecasting of naval assets for condition-based maintenance applications*. Paper presented at International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles, Aachen, Germany. DOI: 10.1109/ESARS.2015.7101499

De Ruiter, A. (2015). *Performance measures in Azure ML: Accuracy, Precision, Recall and F1 Score* [online] Andreas De Ruiter's BI Blog: Microsoft, 2015 [cited 28 Mar. 2016]. Available from World Wide Web <URL: https://blogs.msdn.microsoft.com/andreasderuiter/2015/02/09/performance-measures-in-azure-ml-accuracy-precision-recall-and-f1-score/>.

Dean, J. (2014). *Big Data, Data Mining, and Machine Learning: Value Creation for Business Leaders and Practitioners.* John Wiley & Sons. 288 p. ISBN 978-1-118-61804-2

Dolhansky, B. (2013). Artificial Neural Networks: Mathematics of Backpropagation: Part 4 [online]. Blog: Brian Dolhansky, 2013 [cited 5 Mar. 2016]. Available from World Wide Web <URL: http://briandolhansky.com/blog/2013/9/27/artificial-neural-networks-backpropagation-part-4>.

Drummond, C., & Holte, R. C. (2000). Explicitly representing expected cost: An alternative to ROC representation. In R. Ramakrishnan, S. Stolfo, R.

Bayardo, & I. Parsa (Eds.), *Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining* (pp. 198–207). Boston. New York: ACM Press.

Ersoy-Nürnberg, K., Nürnberg, G., Golle, M., & Hoffmann, H. (2008). Simulation of wear on sheet metal forming tools—An energy approach. *Wear*, 265(11), 1801-1807.

Fayyad, U., Piatetsky-Shapiro, G., Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17(3), 37-54.

Google (n.d.). *Prediction Api: Machine Learning to Analyze Your Data and Make Predictions* [online] Google Cloud Platform: Google [cited 28 Mar. 2016]. Available from World Wide Web <URL: https://cloud.google.com/prediction/>.

Gulati, R. (2012). *Maintenance and Reliability: Best Practices* (2nd ed.). Industrial Press Inc. 400 p. ISBN 978-0831134341

Han, J., Kamber, M., Pei, J. (2011). *Data Mining: Concepts and Techniques* (3rd ed.). Morgan Kaufmann. 800 p. ISBN 978-0123814791

Hastie, T., Tibshirani, R., Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). Springer. 763 p. Available from World Wide Web <URL: http://statweb.stanford.edu/~tibs/ElemStatLearn/>.

Hollinger, P. (2015). *Smarter aircraft create a wealth of data but it remains underexploited* [online] Aerospace: Financial Times, 2015 [cited 21 Feb. 2016]. Available from World Wide Web <URL: http://www.ft.com/intl/cms/s/2/3f956a92-0943-11e5-b643-00144feabdc0.html#axzz44UFjM2B8>.

IBM (n.d.) *What is Watson Analytics?* [online] IBM Watson Analytics: IBM [cited 28 Mar. 2016]. Available from World Wide Web <URL: http://www.ibm.com/analytics/watson-analytics/us-en/>.

Kleyner, A. (2010). *Determining Optimal Reliability Targets: Engineering Analysis and Simulation of Product Validation Costs and Warranty Data*. Lambert Academic Publishing.

Koch, C. (2012). *A Real Need for Big Data: Preventing Airline Equipment Failures* [online] ForbesBrandVoice: Forbes, 2012 [cited 21 Feb. 2016]. Available from World Wide Web <URL: http://www.forbes.com/sites/sap/2012/06/27/a-real-need-for-big-data-preventing-airline-equipment-failures/#7280fde63627>.

Langlouis, A. (2014). *Launch of SAP Predictive Maintenance and Service, cloud edition* [online] SAP Solutions for the Internet of Things: SAP Community Network, 2014 [cited 21 Feb. 2016]. Available from World Wide Web <URL: http://scn.sap.com/community/internet-of-things/blog/2014/11/11/launch-of-sap-predictive-maintenance-and-service>.

Lantz, B. (2015). *Meachine Learning with R* (2nd ed.). Packt Publishing. 452 p. ISBN 978-1-78439-390-8

Li, H., Parikh, D., He, Q., Qian, B., Li, Z., Fang, D., Hampapur, A. (2014). Improving rail network velocity: A machine learning approach to predictive maintenance. *Transportation Research Part C*, 45, 17–26.

Lin, M. (2013). *Bayesian Statistics* [online] Boston University: School of Public Health, 2013 [cited 5 Mar. 2016]. Available from World Wide Web <URL: http://www.bu.edu/sph/files/2014/05/Bayesian-Statistics_final_20140416.pdf>.

Ling, C., Sheng, V. (2011). Class Imbalance Problem. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of Machine Learning* (pp. 167-168). Springer New York.

Maintenance Assistant (2014). *Predictive Maintenance (PdM)* [online] Maintenance Assistant, 2014 [cited 20 Mar. 2016]. Available from World Wide Web <URL: https://www.maintenanceassistant.com/predictive-maintenance/>.

Microsoft (2015a). *Predictive Maintenance: Step 1 of 3, Data Preparation and Feature Engineering* [online] Cortana Intelligence Gallery: Microsoft, 2015 [cited 22 Feb. 2016]. Available from World Wide Web <URL:

http://gallery.cortanaintelligence.com/Experiment/Predictive-Maintenance-Step-1-of-3-data-preparation-and-feature-engineering-2>.

Microsoft (2015b). *A-Z List of Machine Learning Studio Modules* [online] Microsoft Azure: Microsoft, 2016 [cited 31 Mar. 2016]. Available from World Wide Web <URL: https://msdn.microsoft.com/en-us/library/azure/dn906033.aspx>.

Microsoft (2016). *Bring advanced analytics to your data: Data science with Microsoft R Server* [online] Microsoft [cited 28 Mar. 2016]. Available from World Wide Web <URL: http://download.microsoft.com/download/0/9/1/091054C2-A11F-4A4C-910B-2AE6C77BE58E/Microsoft_R_Server_Advanced_Analytics_Datasheet_EN-US.pdf>.

Mishina, Y., Tsuchiya, M., Fujiyoshi, H. (2014). Boosted Random Forest. *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2, 594-598.

Negandhi, V., Sreenivasan, L., Giffen, R., Sewak, M., Rajasekharan, A. (2015). *IBM Predictive Maintenance and Quality 2.0 Technical Overview* (2nd ed.). Redbooks: IBM.

Nguyen, C. , Wang, Y. and Nguyen, H. (2013) Random forest classifier combined with feature selection for breast cancer diagnosis and prognostic. *Journal of Biomedical Science and Engineering*, 6(5), 551-560. DOI: 10.4236/jbise.2013.65070.

O'Brien, J. (2014). *Should you focus on asset availability or cost reduction?* [online] Blog: Maintenance Assistant, 2014 [cited 20 Mar. 2016]. Available from World Wide Web <URL: https://www.maintenanceassistant.com/blog/focus-availability-cost-reduction/>.

O'Connor, P., Kleyner, A. (2012). *Practical Reliability Engineering* (5th ed.). Wiley-Blackwell. 512 p. ISBN 978-0470979815

Pohlen, T. (2015). Geek Stack [online]. Geek Stack, 2015 [cited 5 Mar. 2016]. Available from World Wide Web <URL: http://geekstack.net/>.

Prima Power (2016). Products [online] Prima Power, 2016 [cited 26 Feb. 2016]. Available from World Wide Web <URL: http://www.primapower.com/en/products/>

Provost, F. (2000). Machine Learning from Imbalanced Data Sets 101. In *Proceedings of the AAAI'2000 workshop on imbalanced data.*

Provost, F., Fawcett, T. (2013). *Data Science for Business: What you need to know about data mining and data-analytic thinking*. O'Reilly Media. 414 p. ISBN 978-1449361327

Rohrer, B. (2016). *How to choose algorithms for Microsoft Azure Machine Learning* [online] Microsoft Azure: Microsoft, 2016 [cited 31 Mar. 2016]. Available from World Wide Web <URL: https://azure.microsoft.com/en-us/documentation/articles/machine-learning-algorithm-choice/>.

SAP (2015). *Leverage the Internet of Things to Transform Maintenance and Service Operations* [online] SAP Solutions for the Internet of Things: SAP, 2015 [cited 21 Feb. 2016]. Available from World Wide Web <URL: https://www.sap.com/bin/sapcom/en_us/downloadasset.2015-03-mar-19-17.leverage-the-internet-of-things-to-transform-maintenance-and-service-operations-pdf.html>.

Severtson, B. (2016). *How to choose parameters to optimize your algorithms in Azure Machine Learning* [online] Microsoft Azure: Microsoft, 2016 [cited 31 Mar. 2016]. Available from World Wide Web <URL: https://azure.microsoft.com/en-us/documentation/articles/machine-learning-algorithm-parameters-optimize/>.

Shearer, C. (2000). The CRISP-DM Model: The New Blueprint for Data Mining. *Journal of Data Warehousing*, 5(4), 13-22.

Shotton, J., Sharp, T., Kohli, P., Nowozin, S., Winn, J., Criminisi, A. (2013). Decision Jungles: Compact and Rich Models for Classification. In C. J. C. Burges & L. Bottou & M. Welling & Z. Ghahramani & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 26* (pp. 234-242). Curran Associates, Inc.

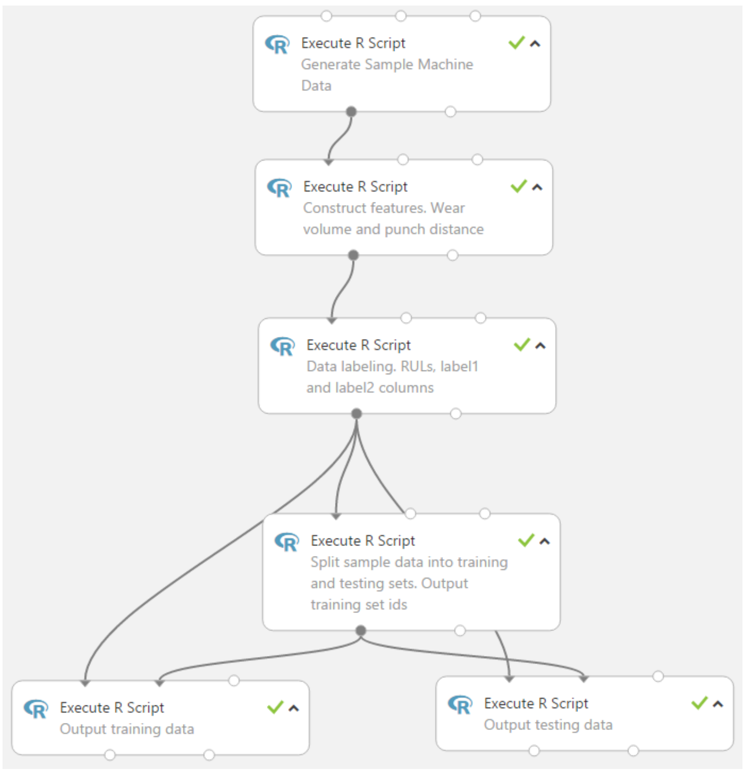
Software AG (2015). *Predictive Maintenance. Capitalize on a new revenue stream while ensuring higher service quality* [online] Solution Series: Software AG, 2015 [cited 22 Feb. 2016]. Available from World Wide Web <URL: https://www.softwareag.com/corporate/images/SAG_Predictive_Maintenance_SB_Oct15-Web_tcm16-127571.pdf>.

Swindoll, C. (2011). *Redefining Fundraising – Data* [online] Blog: Pursuant, 2011 [cited 03 Apr. 2016]. Available from World Wide Web <URL: https://www.pursuant.com/blog/redefining-fundraising-data/>.

Webb, G. I. (2011). Lazy Learning. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of Machine Learning* (pp. 571-572). Springer New York.

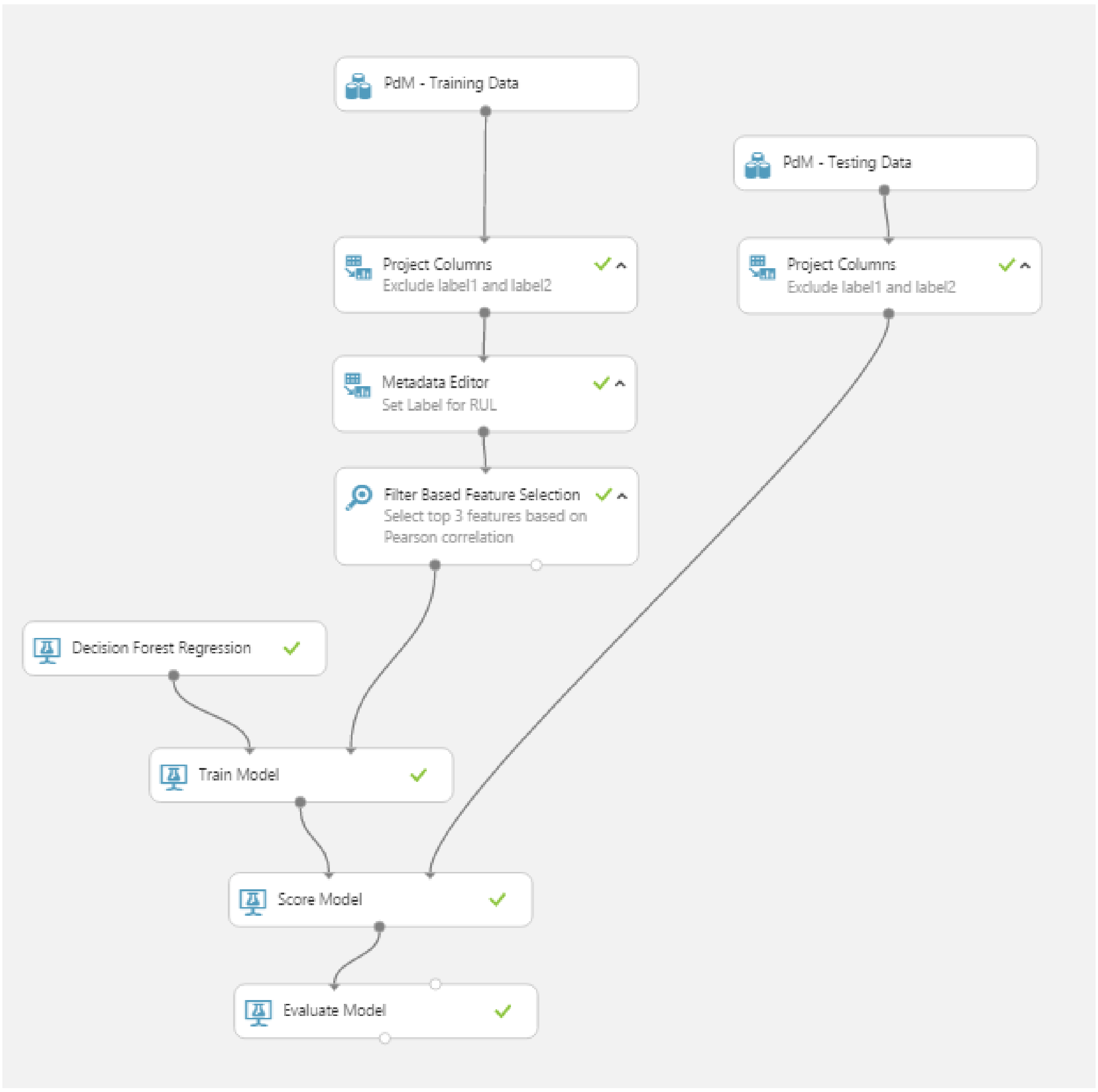
APPENDIX 1.

Prepare Data


Execute R Script
Generate Sample Machine Data
Execute R Script
Construct features. Wear volume and punch distance
Execute R Script
Data labeling. RULs, label1 and label2 columns
Execute R Script
Split sample data into training and testing sets. Output training set ids
Execute R Script
Output training data
Execute R Script
Output testing data

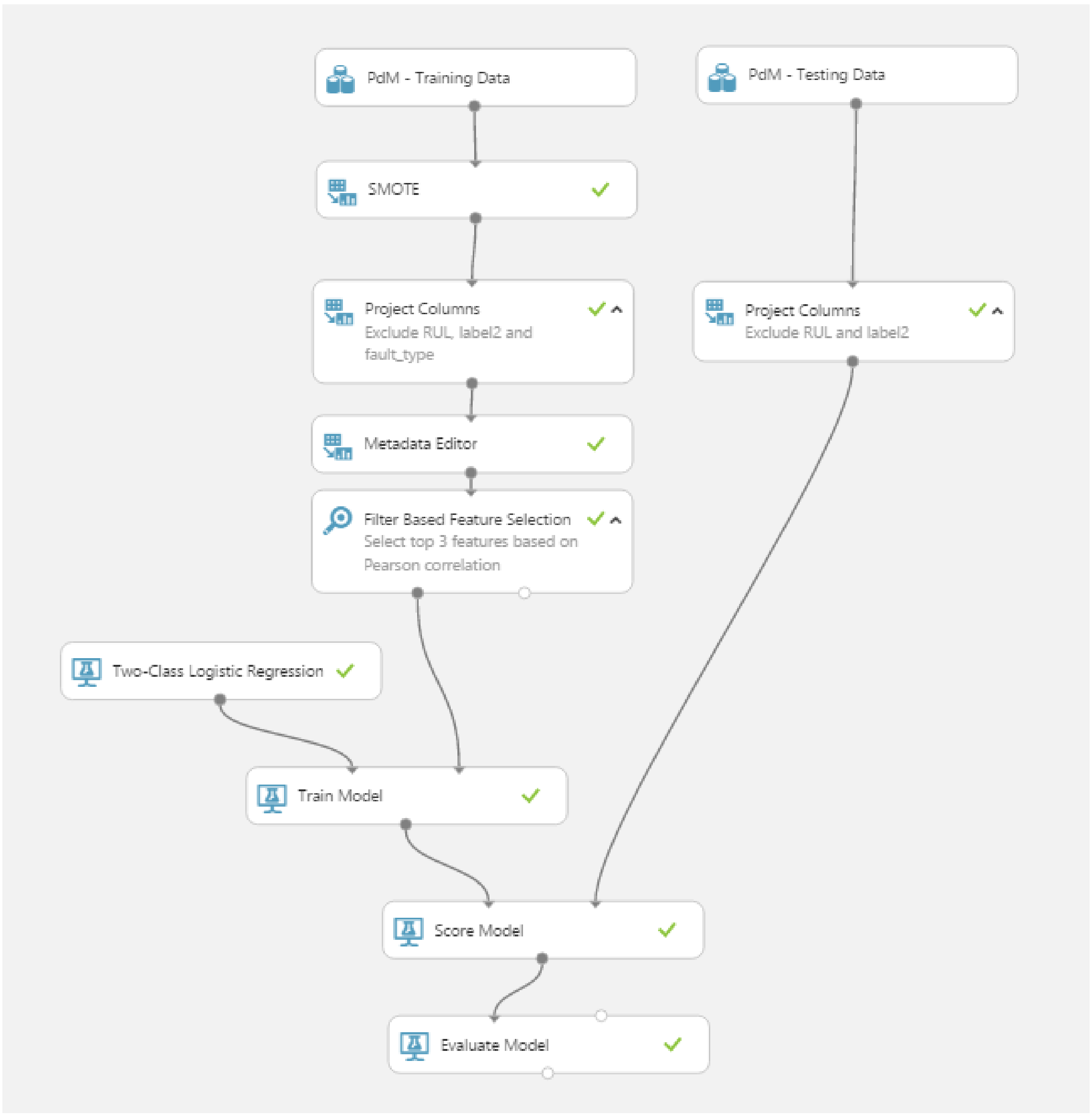
APPENDIX 2.

Regression


PdM - Training Data
PdM - Testing Data
Project Columns
Exclude label1 and label2
Project Columns
Exclude label1 and label2
Metadata Editor
Set Label for RUL
Filter Based Feature Selection
Select top 3 features based on Pearson correlation
Decision Forest Regression
Train Model
Score Model
Evaluate Model

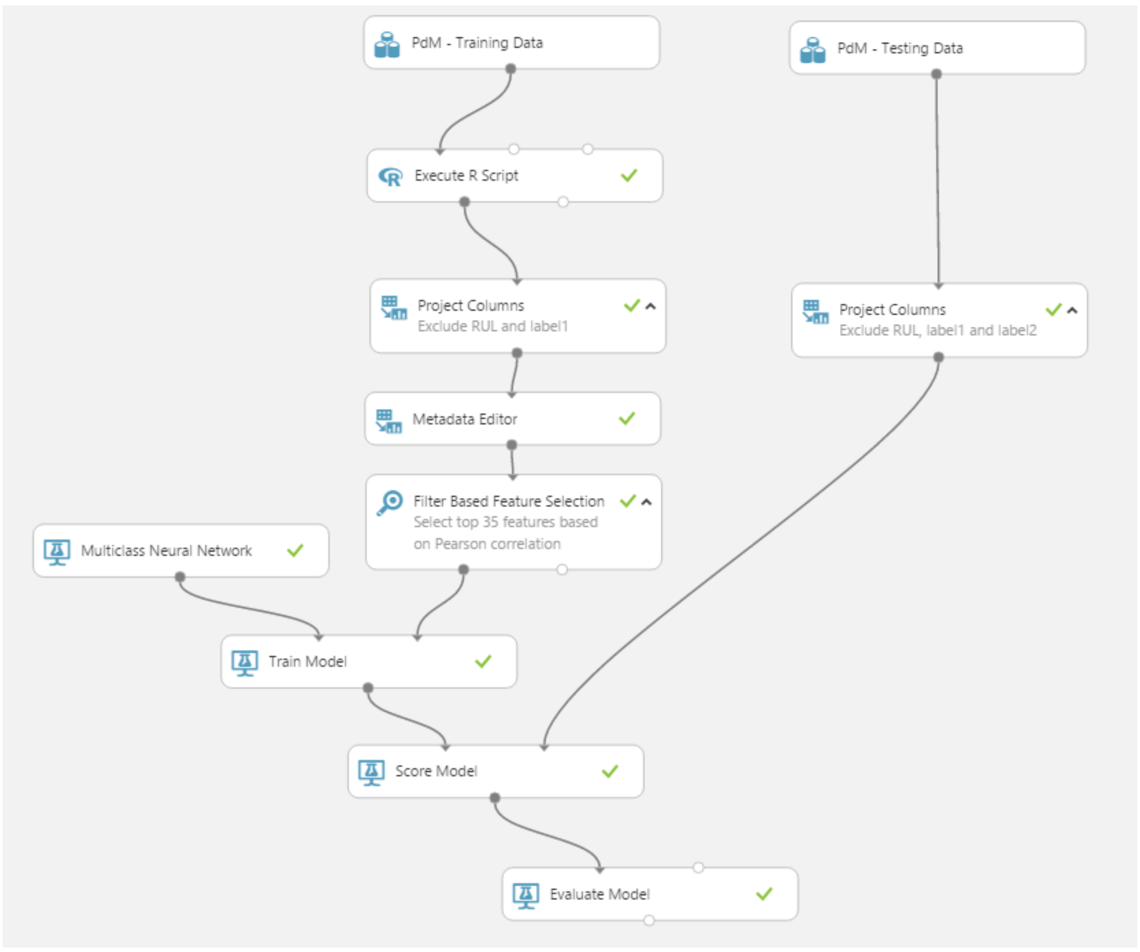
APPENDIX 3.

Binary Classification



APPENDIX 4.

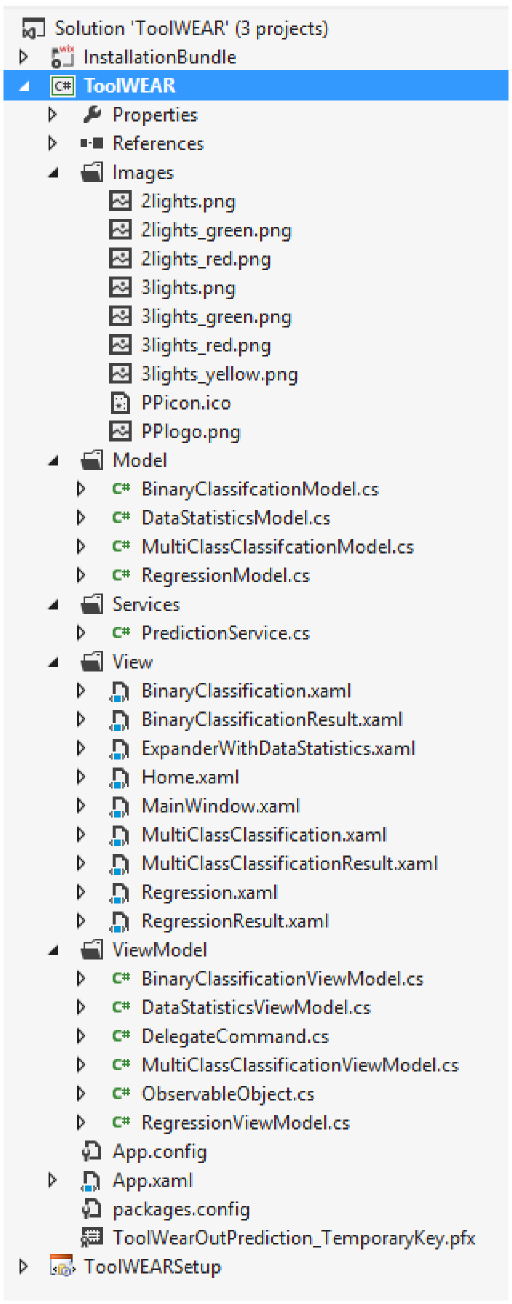
Multiclass Classification



APPENDIX 5.

Sample Application Code

Solution Explorer View



```
Solution 'ToolWEAR' (3 projects)
▷ InstallationBundle
◢ ToolWEAR
    ▷ Properties
    ▷ References
    ◢ Images
        2lights.png
        2lights_green.png
        2lights_red.png
        3lights.png
        3lights_green.png
        3lights_red.png
        3lights_yellow.png
        PPicon.ico
        PPlogo.png
    ◢ Model
        ▷ BinaryClassifcationModel.cs
        ▷ DataStatisticsModel.cs
        ▷ MultiClassClassifcationModel.cs
        ▷ RegressionModel.cs
    ◢ Services
        ▷ PredictionService.cs
    ◢ View
        ▷ BinaryClassification.xaml
        ▷ BinaryClassificationResult.xaml
        ▷ ExpanderWithDataStatistics.xaml
        ▷ Home.xaml
        ▷ MainWindow.xaml
        ▷ MultiClassClassification.xaml
        ▷ MultiClassClassificationResult.xaml
        ▷ Regression.xaml
        ▷ RegressionResult.xaml
    ◢ ViewModel
        ▷ BinaryClassificationViewModel.cs
        ▷ DataStatisticsViewModel.cs
        ▷ DelegateCommand.cs
        ▷ MultiClassClassificationViewModel.cs
        ▷ ObservableObject.cs
        ▷ RegressionViewModel.cs
      App.config
    ▷ App.xaml
      packages.config
      ToolWearOutPrediction_TemporaryKey.pfx
▷ ToolWEARSetup
```

BinaryClassifcationModel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ToolWearOutPrediction.Services;

namespace ToolWearOutPrediction.Model
{
  public class BinaryClassificationModel
  {
    public BinaryClassificationModel()
    {
    }

    public Tuple<int, double> GetPredictedToolWearOut(
        int cycle,
        double cumPunchDistance,
        double cumToolWearVol)
    {
      return PredictionService.GetBinaryClassificationPrediction(
          cycle, cumPunchDistance, cumToolWearVol);
    }
  }
}
```

DataStatisticsModel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ToolWearOutPrediction.Services;

namespace ToolWearOutPrediction.Model
{
  public class DataStatisticsModel
  {
    private static DataStatisticsModel instance;

    public event EventHandler ModelUpdated;

    #region Basic Data Statistics

    public int MeanLastCycle
    {
      get;
      set;
    }

    public int MinLastCycle
    {
      get;
      set;
    }

    public int MaxLastCycle
    {
      get;
      set;
    }

    public double MeanPunchDistance
    {
```

```
  get;
  set;
}

public double MinPunchDistance
{
  get;
  set;
}

public double MaxPunchDistance
{
  get;
  set;
}

public double MeanCumPunchDistanceAtFault
{
  get;
  set;
}

public double MinCumPunchDistanceAtFault
{
  get;
  set;
}

public double MaxCumPunchDistanceAtFault
{
  get;
  set;
}

public double MeanPunchForce
{
  get;
  set;
}

public double MinPunchForce
{
  get;
  set;
}

public double MaxPunchForce
{
  get;
  set;
}

public double MeanCumToolWearVolAtFault
{
  get;
  set;
}

public double MinCumToolWearVolAtFault
{
  get;
  set;
}

public double MaxCumToolWearVolAtFault
{
  get;
  set;
}
#endregion
```

```
private DataStatisticsModel()
{
  //PredictionService.GetDataStatistics();
  Task.Run(() => GetData());
}

void GetData()
{
  Dictionary<string, double> dataStatistics = PredictionService.GetDataStatistics();
  foreach (KeyValuePair<string, double> entry in dataStatistics)
  {
    switch (entry.Key)
    {
      case "mean_last_cycle":
        MeanLastCycle = Convert.ToInt32(entry.Value);
        break;
      case "min_last_cycle":
        MinLastCycle = Convert.ToInt32(entry.Value);
        break;
      case "max_last_cycle":
        MaxLastCycle = Convert.ToInt32(entry.Value);
        break;
      case "mean_punch_force":
        MeanPunchForce = entry.Value;
        break;
      case "min_punch_force":
        MinPunchForce = entry.Value;
        break;
      case "max_punch_force":
        MaxPunchForce = entry.Value;
        break;
      case "mean_punch_distance":
        MeanPunchDistance = entry.Value;
        break;
      case "min_punch_distance":
        MinPunchDistance = entry.Value;
        break;
      case "max_punch_distance":
        MaxPunchDistance = entry.Value;
        break;
      case "mean_last_cum_punch_distance":
        MeanCumPunchDistanceAtFault = entry.Value;
        break;
      case "min_last_cum_punch_distance":
        MinCumPunchDistanceAtFault = entry.Value;
        break;
      case "max_last_cum_punch_distance":
        MaxCumPunchDistanceAtFault = entry.Value;
        break;
      case "mean_last_cum_tool_wear":
        MeanCumToolWearVolAtFault = entry.Value;
        break;
      case "min_last_cum_tool_wear":
        MinCumToolWearVolAtFault = entry.Value;
        break;
      case "max_last_cum_tool_wear":
        MaxCumToolWearVolAtFault = entry.Value;
        break;
    }
  }

  OnModelUpdated(EventArgs.Empty);
}

protected virtual void OnModelUpdated(EventArgs e)
{
  EventHandler handler = ModelUpdated;
  if (handler != null)
```

```
      {
        handler(this, e);
      }
    }

    public static DataStatisticsModel Instance
    {
      get
      {
        if (instance == null)
        {
          instance = new DataStatisticsModel();
        }
        return instance;
      }
    }
  }
}
```

MultiClassClassifcationModel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ToolWearOutPrediction.Services;

namespace ToolWearOutPrediction.Model
{
  public class MultiClassClassificationModel
  {
    public MultiClassClassificationModel()
    {
    }

    public List<double> GetPredictedToolWearOut(
        int cycle,
        double cumPunchDistance,
        double cumToolWearVol)
    {
      return PredictionService.GetMultiClassClassificationPrediction(
        cycle, cumPunchDistance, cumToolWearVol);
    }
  }
}
```

RegressionModel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ToolWearOutPrediction.Services;

namespace ToolWearOutPrediction.Model
{
  public class RegressionModel
  {
    public RegressionModel()
    {
    }

    public Tuple<double, double> GetPredictedToolWearOut(
        int cycle,
        double punchForce,
        double punchDistance,
        double cumPunchDistance,
        int sheetMetalType,
        double cumToolWearVol)
    {
      return PredictionService.GetRegressionPrediction(
        cycle, punchForce, punchDistance, cumPunchDistance, sheetMetalType, cumToolWearVol);
    }
  }
}
```

PredictionService.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using System.IO;
using System.Net.Http;
using System.Net.Http.Formatting;
using System.Net.Http.Headers;

using System.Runtime.Serialization.Json;
using Newtonsoft.Json;

namespace ToolWearOutPrediction.Services
{
  public class ResultRootObject
  {
    public Results Results { get; set; }
    public Error error { get; set; }
  }

  public class Results
  {
    public Output1 output1 { get; set; }
  }

  public class Output1
  {
    public string type { get; set; }
    public Value value { get; set; }
  }

  public class Value
  {
    public List<string> ColumnNames { get; set; }
    public List<string> ColumnTypes { get; set; }
    public List<List<string>> Values { get; set; }
  }

  public class Error
  {
    public string code { get; set; }
    public string message { get; set; }
    public List<ErrorDetail> details { get; set; }
  }

  public class ErrorDetail
  {
    public string code { get; set; }
    public string message { get; set; }
  }
```

```
public class StringTable
{
  public string[] ColumnNames { get; set; }
  public string[,] Values { get; set; }
}

public class RequestContent
{
  public Dictionary<string, StringTable> Inputs;
  public Dictionary<string, string> GlobalParameters;

  public RequestContent()
  {
    Inputs = new Dictionary<string, StringTable>() { };
    GlobalParameters = new Dictionary<string, string>() { };
  }

  public void AddInput(string inputName, StringTable namesAndValues)
  {
    Inputs.Add(inputName, namesAndValues);
  }
}


public static class PredictionService
{
  public static Dictionary<string, double> GetDataStatistics()
  {
    const string apiKey = "...";
    Uri baseAddress = new Uri("https://...");

    RequestContent requestContent = new RequestContent();
    var reqResponseService =
      InvokeRequestResponseService(apiKey, baseAddress, requestContent);
    reqResponseService.Wait();
    string response = reqResponseService.Result;
    if (String.IsNullOrEmpty(response))
      return new Dictionary<string, double>();

    var outObject =
      JsonConvert.DeserializeObject<ResultRootObject>(response);
    if (IsErrorPresent(outObject))
      return new Dictionary<string, double>();

    Dictionary<string, double> dataStatistics =
      new Dictionary<string, double>();
    List<string> variablesNames = new List<string>() {
              "mean_last_cycle",
              "min_last_cycle",
              "max_last_cycle",
              "mean_punch_force",
              "min_punch_force",
              "max_punch_force",
              "mean_punch_distance",
              "min_punch_distance",
              "max_punch_distance",
              "mean_last_cum_punch_distance",
              "min_last_cum_punch_distance",
              "max_last_cum_punch_distance",
              "mean_last_cum_tool_wear",
              "min_last_cum_tool_wear",
              "max_last_cum_tool_wear" };
    foreach (var variableName in variablesNames)
    {
      int index =
        outObject.Results.output1.value.ColumnNames.IndexOf(variableName);
      double value =
        double.Parse(outObject.Results.output1.value.Values[0][index]);
      dataStatistics.Add(variableName, value);
    }
    return dataStatistics;
  }

  public static Tuple<double, double> GetRegressionPrediction(
      int cycle,
      double punchForce,
      double punchDistance,
      double cumPunchDistance,
      int sheetMetalType,
      double cumToolWearVol)
  {
    const string apiKey = "...";
    Uri baseAddress = new Uri("https://...");

    RequestContent requestContent = new RequestContent();
    requestContent.AddInput(
        "input1",
        new StringTable()
        {
          ColumnNames = new string[] {
            "cycle", "punch_force", "punch_distance", "sheetmetal_type",
            "tool_wear_vol", "cum_tool_wear_vol", "cum_punch_distance" },
          Values = new string[,] { {
                         cycle.ToString(),
                         punchForce.ToString(),
                         punchDistance.ToString(),
                         sheetMetalType.ToString(),
                         "0",
                         cumToolWearVol.ToString(),
                         cumPunchDistance.ToString() }, }
        });
    var reqResponseService =
      InvokeRequestResponseService(apiKey, baseAddress, requestContent);
    reqResponseService.Wait();
    string response = reqResponseService.Result;
    if (String.IsNullOrEmpty(response))
      return Tuple.Create<double, double>(-1, -1);

    var outObject = JsonConvert.DeserializeObject<ResultRootObject>(response);
    if (IsErrorPresent(outObject))
      return Tuple.Create<double, double>(-1, -1);

    int scoredLabelIndex =
      outObject.Results.output1.value.ColumnNames.IndexOf("Scored Label Mean");
    double scoredLabelValue =
      double.Parse(outObject.Results.output1.value.Values[0][scoredLabelIndex]);
    int scoredLabelSDIndex =
      outObject.Results.output1.value.ColumnNames.IndexOf("Scored Label Standard Deviation");
    double scoredLabelSDValue =
      double.Parse(outObject.Results.output1.value.Values[0][scoredLabelSDIndex]);

    return Tuple.Create<double, double>(scoredLabelValue, scoredLabelSDValue);
  }

  public static Tuple<int, double> GetBinaryClassificationPrediction(
      int cycle,
      double cumPunchDistance,
      double cumToolWearVol)
  {
    const string apiKey = "...";
    Uri baseAddress = new Uri("https://...");

    RequestContent requestContent = new RequestContent();
    requestContent.AddInput(
        "input1",
        new StringTable()
        {
          ColumnNames = new string[] {
            "cycle", "cum_tool_wear_vol",
            "cum_punch_distance" },
          Values = new string[,] { {
                      cycle.ToString(),
                      cumToolWearVol.ToString(),
                      cumPunchDistance.ToString() }, }
        });
    var reqResponseService =
      InvokeRequestResponseService(apiKey, baseAddress, requestContent);
    reqResponseService.Wait();
    string response = reqResponseService.Result;
    if (String.IsNullOrEmpty(response))
      return Tuple.Create<int, double>(-1, -1);

    var outObject =
      JsonConvert.DeserializeObject<ResultRootObject>(response);
    if (IsErrorPresent(outObject))
      return Tuple.Create<int, double>(-1, -1);

    int scoredLabelsIndex =
      outObject.Results.output1.value.ColumnNames.IndexOf("Scored Labels");
    int scoredLabelsValue =
      Int32.Parse(outObject.Results.output1.value.Values[0][scoredLabelsIndex]);
    int scoredProbabilitiesIndex =
      outObject.Results.output1.value.ColumnNames.IndexOf("Scored Probabilities");
    double scoredProbabilitiesValue =
      double.Parse(outObject.Results.output1.value.Values[0][scoredProbabilitiesIndex]);

    return Tuple.Create<int, double>(scoredLabelsValue, scoredProbabilitiesValue);
  }

  public static List<double> GetMultiClassClassificationPrediction(
      int cycle,
      double cumPunchDistance,
      double cumToolWearVol)
  {
    const string apiKey = "...";
    Uri baseAddress = new Uri("https://.....");

    RequestContent requestContent = new RequestContent();
    requestContent.AddInput(
        "input1",
        new StringTable()
        {
          ColumnNames = new string[] {
            "cycle", "cum_tool_wear_vol",
            "cum_punch_distance" },
          Values = new string[,] { {
                      cycle.ToString(),
                      cumToolWearVol.ToString(),
                      cumPunchDistance.ToString() }, }
        });
    var reqResponseService =
      InvokeRequestResponseService(apiKey, baseAddress, requestContent);
    reqResponseService.Wait();
    string response = reqResponseService.Result;
    if (String.IsNullOrEmpty(response))
      return new List<double>() { -1, -1, -1, -1 };

    var outObject =
      JsonConvert.DeserializeObject<ResultRootObject>(response);
    if (IsErrorPresent(outObject))
      return new List<double>() { -1, -1, -1, -1 };

    int scoredLabelsIndex =
      outObject.Results.output1.value.ColumnNames.IndexOf("Scored Labels");
    double scoredLabelsValue =
      double.Parse(outObject.Results.output1.value.Values[0][scoredLabelsIndex]);
    int scoredProbabilitiesClass0Index =
      outObject.Results.output1.value.ColumnNames.IndexOf(
      "Scored Probabilities for Class \"0\"");
```

```
      double scoredProbabilitiesClass0Value =
        double.Parse(outObject.Results.output1.value.Values[0][scoredProbabilitiesClass0Index]);
      int scoredProbabilitiesClass1Index =
        outObject.Results.output1.value.ColumnNames.IndexOf(
        "Scored Probabilities for Class \"1\"");
      double scoredProbabilitiesClass1Value =
        double.Parse(outObject.Results.output1.value.Values[0][scoredProbabilitiesClass1Index]);
      int scoredProbabilitiesClass2Index =
        outObject.Results.output1.value.ColumnNames.IndexOf(
        "Scored Probabilities for Class \"2\"");
      double scoredProbabilitiesClass2Value =
        double.Parse(outObject.Results.output1.value.Values[0][scoredProbabilitiesClass2Index]);

      return new List<double>() {
        scoredLabelsValue, scoredProbabilitiesClass0Value,
        scoredProbabilitiesClass1Value, scoredProbabilitiesClass2Value
      };
    }


    static async Task<string> InvokeRequestResponseService(
      string apiKey, Uri baseAddress, RequestContent requestContent)
    {
      try
      {
        using (var client = new HttpClient())
        {
          client.DefaultRequestHeaders.Authorization =
            new AuthenticationHeaderValue("Bearer", apiKey);
          client.BaseAddress = baseAddress;

          HttpResponseMessage response =
            await client.PostAsJsonAsync("", requestContent).ConfigureAwait(false);

          if (!response.IsSuccessStatusCode)
          {
            // Request Failed
            Console.WriteLine(string.Format("The request failed with status code: {0}",
              response.StatusCode));
            Console.WriteLine(response.Headers.ToString());
            string responseContent = await response.Content.ReadAsStringAsync();
            Console.WriteLine(responseContent);
          }

          return await response.Content.ReadAsStringAsync();

        }
      }
      catch (HttpRequestException e)
      {
        System.Windows.MessageBox.Show(e.Message, "Error",
          System.Windows.MessageBoxButton.OK, System.Windows.MessageBoxImage.Error
          );
        return string.Empty;
      }
    }

    private static bool IsErrorPresent(ResultRootObject result)
    {
      if (result.error == null)
        return false;

      System.Windows.MessageBox.Show(
        result.error.code + "\n" + result.error.message, "Service Responded With Error",
        System.Windows.MessageBoxButton.OK, System.Windows.MessageBoxImage.Error
        );
      return true;
    }
  }
}
```

BinaryClassificationViewModel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Collections.ObjectModel;
using System.Windows.Input;
using ToolWearOutPrediction.Model;
using System.ComponentModel;

namespace ToolWearOutPrediction.ViewModel
{
  public class BinaryClassificationPrediction
  {
    public int Cycle { get; set; }
    public double CumPunchDistance { get; set; }
    public double CumToolWearVol { get; set; }

    public int PredictedLabel { get; set; }
    public double PredictedLabelProbability { get; set; }
  }

  public class BinaryClassificationViewModel : ObservableObject, INotifyDataErrorInfo
  {
    private readonly BinaryClassificationModel _binaryClassificationModel =
      new BinaryClassificationModel();

    private readonly List<BinaryClassificationPrediction> _predictionHistory =
      new List<BinaryClassificationPrediction>();

    private readonly Dictionary<string, ICollection<string>>
    _validationErrors = new Dictionary<string, ICollection<string>>();

    private int _cycle;
    private double _cumPunchDistance;
    private double _cumToolWearVol;

    private int _predictedLabel;
    private double _predictedLabelProbability;

    public BinaryClassificationViewModel()
    {
      ErrorsChanged += RegressionViewModel_ErrorsChanged;
    }

    void RegressionViewModel_ErrorsChanged(object sender, DataErrorsChangedEventArgs e)
    {
      RaisePropertyChangedEvent("IsFormValid");
      RaisePropertyChangedEvent("HasErrors");
    }

    public int Cycle
    {
      get { return _cycle; }
      set
      {
        if (value < 0 || value > 10000)
        {
          _validationErrors["Cycle"] = new List<string>(
            new string[] { "Value out of range." });
          RaiseErrorsChanged("Cycle");
        }
        else
        {
          _validationErrors.Remove("Cycle");
          RaiseErrorsChanged("Cycle");
        }
        _cycle = value;
        RaisePropertyChangedEvent("Cycle");
      }
    }

    public double CumulativePunchDistance
    {
      get { return _cumPunchDistance; }
      set
      {
        _cumPunchDistance = value;
        RaisePropertyChangedEvent("CumulativePunchDistance");
      }
    }

    public double CumulativeToolWearVolume
    {
      get { return _cumToolWearVol; }
      set
      {
        _cumToolWearVol = value;
        RaisePropertyChangedEvent("CumulativeToolWearVolume");
      }
    }

    public List<BinaryClassificationPrediction> History
    {
      get { return _predictionHistory; }
    }

    public int PredictedLabel
    {
      get { return _predictedLabel; }
      set
      {
        _predictedLabel = value;
        RaisePropertyChangedEvent("PredictedLabel");
      }
    }

    public double PredictedLabelProbability
    {
      get { return _predictedLabelProbability; }
      set
      {
        _predictedLabelProbability = value;
        RaisePropertyChangedEvent("PredictedLabelProbability");
      }
    }

    public ICommand PredictCommand
    {
      get { return new DelegateCommand(Predict); }
    }

    private void Predict()
    {
      if (HasErrors)
        return;

      Tuple<int, double> predictionTuple = _binaryClassificationModel.GetPredictedToolWearOut(
        _cycle, _cumPunchDistance, _cumToolWearVol);
      PredictedLabel = predictionTuple.Item1;
      PredictedLabelProbability = predictionTuple.Item2;

      _predictionHistory.Add(new BinaryClassificationPrediction
      {
        Cycle = Cycle,
        CumPunchDistance = CumulativePunchDistance,
        CumToolWearVol = CumulativeToolWearVolume,
        PredictedLabel = PredictedLabel,
        PredictedLabelProbability = PredictedLabelProbability
      });
    }

    public bool IsFormValid
    {
      get { return !HasErrors; }
    }


    #region INotifyDataErrorInfo members
    public event EventHandler<DataErrorsChangedEventArgs> ErrorsChanged;
    private void RaiseErrorsChanged(string propertyName)
    {
      if (ErrorsChanged != null)
        ErrorsChanged(this, new DataErrorsChangedEventArgs(propertyName));
    }

    public System.Collections.IEnumerable GetErrors(string propertyName)
    {
      if (string.IsNullOrEmpty(propertyName)
          || !_validationErrors.ContainsKey(propertyName))
        return null;

      return _validationErrors[propertyName];
    }

    public bool HasErrors
    {
      get { return _validationErrors.Count > 0; }
    }
    #endregion
  }
}
```

DataStatisticsViewModel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ToolWearOutPrediction.Model;
using System.ComponentModel;

namespace ToolWearOutPrediction.ViewModel
{
  public class DataStatisticsViewModel : ObservableObject
  {
    private static DataStatisticsViewModel instance;

    private readonly DataStatisticsModel _dataStatisticsModel = DataStatisticsModel.Instance;

    public static DataStatisticsViewModel Instance
    {
      get
      {
        if (instance == null)
        {
          instance = new DataStatisticsViewModel();
        }
        return instance;
      }
    }

    private DataStatisticsViewModel()
```

```
{
  _dataStatisticsModel.ModelUpdated += DataStatisticsModelUpdated;
}

public DataStatisticsModel DataStatisticModelInstance
{
  get { return _dataStatisticsModel; }
}

void DataStatisticsModelUpdated(object sender, EventArgs e)
{
  MeanLastCycle = _dataStatisticsModel.MeanLastCycle;
  MinLastCycle = _dataStatisticsModel.MinLastCycle;
  MaxLastCycle = _dataStatisticsModel.MaxLastCycle;
  MeanPunchForce = _dataStatisticsModel.MeanPunchForce;
  MinPunchForce = _dataStatisticsModel.MinPunchForce;
  MaxPunchForce = _dataStatisticsModel.MaxPunchForce;
  MeanPunchDistance = _dataStatisticsModel.MeanPunchDistance;
  MinPunchDistance = _dataStatisticsModel.MinPunchDistance;
  MaxPunchDistance = _dataStatisticsModel.MaxPunchDistance;
  MeanCumPunchDistanceAtFault = _dataStatisticsModel.MeanCumPunchDistanceAtFault;
  MinCumPunchDistanceAtFault = _dataStatisticsModel.MinCumPunchDistanceAtFault;
  MaxCumPunchDistanceAtFault = _dataStatisticsModel.MaxCumPunchDistanceAtFault;
  MeanCumToolWearVolAtFault = _dataStatisticsModel.MeanCumToolWearVolAtFault;
  MinCumToolWearVolAtFault = _dataStatisticsModel.MinCumToolWearVolAtFault;
  MaxCumToolWearVolAtFault = _dataStatisticsModel.MaxCumToolWearVolAtFault;
}

#region Basic Data Statistics
private int _meanLastCycle;
private int _minLastCycle;
private int _maxLastCycle;

private double _meanPunchDistance;
private double _minPunchDistance;
private double _maxPunchDistance;

private double _meanCumPunchDistanceAtFault;
private double _minCumPunchDistanceAtFault;
private double _maxCumPunchDistanceAtFault;

private double _meanPunchForce;
private double _minPunchForce;
private double _maxPunchForce;

private double _meanCumToolWearVolAtFault;
private double _minCumToolWearVolAtFault;
private double _maxCumToolWearVolAtFault;

public int MeanLastCycle
{
  get { return _meanLastCycle; }
  set
  {
    _meanLastCycle = value;
    RaisePropertyChangedEvent("MeanLastCycle");
  }
}

public int MinLastCycle
{
  get { return _minLastCycle; }
  set
  {
    _minLastCycle = value;
    RaisePropertyChangedEvent("MinLastCycle");
  }
}

public int MaxLastCycle
```

```
{
  get { return _maxLastCycle; }
  set
  {
    _maxLastCycle = value;
    RaisePropertyChangedEvent("MaxLastCycle");
  }
}

public double MeanPunchDistance
{
  get { return Convert.ToDouble(_meanPunchDistance.ToString("N2")); }
  set
  {
    _meanPunchDistance = value;
    RaisePropertyChangedEvent("MeanPunchDistance");
  }
}

public double MinPunchDistance
{
  get { return Convert.ToDouble(_minPunchDistance.ToString("N2")); }
  set
  {
    _minPunchDistance = value;
    RaisePropertyChangedEvent("MinPunchDistance");
  }
}

public double MaxPunchDistance
{
  get { return Convert.ToDouble(_maxPunchDistance.ToString("N2")); }
  set
  {
    _maxPunchDistance = value;
    RaisePropertyChangedEvent("MaxPunchDistance");
  }
}

public double MeanCumPunchDistanceAtFault
{
  get { return Convert.ToDouble(_meanCumPunchDistanceAtFault.ToString("N2")); }
  set
  {
    _meanCumPunchDistanceAtFault = value;
    RaisePropertyChangedEvent("MeanCumPunchDistanceAtFault");
  }
}

public double MinCumPunchDistanceAtFault
{
  get { return Convert.ToDouble(_minCumPunchDistanceAtFault.ToString("N2")); }
  set
  {
    _minCumPunchDistanceAtFault = value;
    RaisePropertyChangedEvent("MinCumPunchDistanceAtFault");
  }
}

public double MaxCumPunchDistanceAtFault
{
  get { return Convert.ToDouble(_maxCumPunchDistanceAtFault.ToString("N2")); }
  set
  {
    _maxCumPunchDistanceAtFault = value;
    RaisePropertyChangedEvent("MaxCumPunchDistanceAtFault");
  }
}

public double MeanPunchForce
```

```
    {
      get { return Convert.ToDouble(_meanPunchForce.ToString("N2")); }
      set
      {
        _meanPunchForce = value;
        RaisePropertyChangedEvent("MeanPunchForce");
      }
    }

    public double MinPunchForce
    {
      get { return Convert.ToDouble(_minPunchForce.ToString("N2")); }
      set
      {
        _minPunchForce = value;
        RaisePropertyChangedEvent("MinPunchForce");
      }
    }

    public double MaxPunchForce
    {
      get { return Convert.ToDouble(_maxPunchForce.ToString("N2")); }
      set
      {
        _maxPunchForce = value;
        RaisePropertyChangedEvent("MaxPunchForce");
      }
    }

    public double MeanCumToolWearVolAtFault
    {
      get { return Convert.ToDouble(_meanCumToolWearVolAtFault.ToString("N3")); }
      set
      {
        _meanCumToolWearVolAtFault = value;
        RaisePropertyChangedEvent("MeanCumToolWearVolAtFault");
      }
    }

    public double MinCumToolWearVolAtFault
    {
      get { return Convert.ToDouble(_minCumToolWearVolAtFault.ToString("N3")); }
      set
      {
        _minCumToolWearVolAtFault = value;
        RaisePropertyChangedEvent("MinCumToolWearVolAtFault");
      }
    }

    public double MaxCumToolWearVolAtFault
    {
      get { return Convert.ToDouble(_maxCumToolWearVolAtFault.ToString("N3")); }
      set
      {
        _maxCumToolWearVolAtFault = value;
        RaisePropertyChangedEvent("MaxCumToolWearVolAtFault");
      }
    }
    #endregion
  }
}
```

DelegateCommand.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Input;

namespace ToolWearOutPrediction.ViewModel
{
  public class DelegateCommand : ICommand
  {
    private readonly Action _action;
    public event EventHandler CanExecuteChanged = null;

    public DelegateCommand(Action action)
    {
      _action = action;
    }

    public void Execute(object parameter)
    {
      _action();
    }

    public bool CanExecute(object parameter)
    {
      return true;
    }
  }
}
```

MultiClassClassificationViewModel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Collections.ObjectModel;
using System.Windows.Input;
using ToolWearOutPrediction.Model;
using System.ComponentModel;

namespace ToolWearOutPrediction.ViewModel
{
  public class MultiClassClassificationPrediction
  {
    public int Cycle { get; set; }
    public double CumPunchDistance { get; set; }
    public double CumToolWearVol { get; set; }

    public int PredictedLabel { get; set; }
    public double PredictedClass0Probability { get; set; }
    public double PredictedClass1Probability { get; set; }
    public double PredictedClass2Probability { get; set; }
  }

  public class MultiClassClassificationViewModel : ObservableObject, INotifyDataErrorInfo
  {
    private readonly MultiClassClassificationModel _multiclassClassificationModel =
      new MultiClassClassificationModel();

    private readonly List<MultiClassClassificationPrediction> _predictionHistory =
      new List<MultiClassClassificationPrediction>();
```

```
private readonly Dictionary<string, ICollection<string>>
_validationErrors = new Dictionary<string, ICollection<string>>();

private int _cycle;
private double _cumPunchDistance;
private double _cumToolWearVol;

private int _predictedLabel;
private double _predictedClass0Probability;
private double _predictedClass1Probability;
private double _predictedClass2Probability;

public MultiClassClassificationViewModel()
{
  ErrorsChanged += RegressionViewModel_ErrorsChanged;
}

void RegressionViewModel_ErrorsChanged(object sender, DataErrorsChangedEventArgs e)
{
  RaisePropertyChangedEvent("IsFormValid");
  RaisePropertyChangedEvent("HasErrors");
}

public int Cycle
{
  get { return _cycle; }
  set
  {
    if (value < 0 || value > 10000)
    {
      _validationErrors["Cycle"] = new List<string>(
        new string[] { "Value out of range." });
      RaiseErrorsChanged("Cycle");
    }
    else
    {
      _validationErrors.Remove("Cycle");
      RaiseErrorsChanged("Cycle");
    }

    _cycle = value;
    RaisePropertyChangedEvent("Cycle");
  }
}

public double CumulativePunchDistance
{
  get { return _cumPunchDistance; }
  set
  {
    _cumPunchDistance = value;
    RaisePropertyChangedEvent("CumulativePunchDistance");
  }
}

public double CumulativeToolWearVolume
{
  get { return _cumToolWearVol; }
  set
  {
    _cumToolWearVol = value;
    RaisePropertyChangedEvent("CumulativeToolWearVolume");
  }
}

public List<MultiClassClassificationPrediction> History
{
  get { return _predictionHistory; }
}
```

```
public int PredictedLabel
{
  get { return _predictedLabel; }
  set
  {
    _predictedLabel = value;
    RaisePropertyChangedEvent("PredictedLabel");
  }
}

public double PredictedClass0Probability
{
  get { return _predictedClass0Probability; }
  set
  {
    _predictedClass0Probability = value;
    RaisePropertyChangedEvent("PredictedClass0Probability");
  }
}

public double PredictedClass1Probability
{
  get { return _predictedClass1Probability; }
  set
  {
    _predictedClass1Probability = value;
    RaisePropertyChangedEvent("PredictedClass1Probability");
  }
}

public double PredictedClass2Probability
{
  get { return _predictedClass2Probability; }
  set
  {
    _predictedClass2Probability = value;
    RaisePropertyChangedEvent("PredictedClass2Probability");
  }
}

public ICommand PredictCommand
{
  get { return new DelegateCommand(Predict); }
}

private void Predict()
{
  if (HasErrors)
    return;

  List<double> predictionResults =
    _multiclassClassificationModel.GetPredictedToolWearOut(
    _cycle, _cumPunchDistance, _cumToolWearVol);
  PredictedLabel = (int)predictionResults[0];
  PredictedClass0Probability = predictionResults[1];
  PredictedClass1Probability = predictionResults[2];
  PredictedClass2Probability = predictionResults[3];

  _predictionHistory.Add(new MultiClassClassificationPrediction
  {
    Cycle = Cycle,
    CumPunchDistance = CumulativePunchDistance,
    CumToolWearVol = CumulativeToolWearVolume,
    PredictedLabel = PredictedLabel,
    PredictedClass0Probability = PredictedClass0Probability,
    PredictedClass1Probability = PredictedClass1Probability,
    PredictedClass2Probability = PredictedClass2Probability,
  });
}
```

```
    public bool IsFormValid
    {
      get { return !HasErrors; }
    }


    #region INotifyDataErrorInfo members
    public event EventHandler<DataErrorsChangedEventArgs> ErrorsChanged;
    private void RaiseErrorsChanged(string propertyName)
    {
      if (ErrorsChanged != null)
        ErrorsChanged(this, new DataErrorsChangedEventArgs(propertyName));
    }

    public System.Collections.IEnumerable GetErrors(string propertyName)
    {
      if (string.IsNullOrEmpty(propertyName)
          || !_validationErrors.ContainsKey(propertyName))
        return null;

      return _validationErrors[propertyName];
    }

    public bool HasErrors
    {
      get { return _validationErrors.Count > 0; }
    }
    #endregion
  }
}
```

ObservableObject.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.ComponentModel;

namespace ToolWearOutPrediction.ViewModel
{
  public class ObservableObject : INotifyPropertyChanged
  {
    public event PropertyChangedEventHandler PropertyChanged;


    protected virtual void RaisePropertyChangedEvent(string propertyName)
    {
      var handler = PropertyChanged;
      if (handler != null)
        handler(this, new PropertyChangedEventArgs(propertyName));
    }
  }
}
```

RegressionViewModel.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Collections.ObjectModel;
using System.Windows.Input;
using ToolWearOutPrediction.Model;
using System.ComponentModel;

namespace ToolWearOutPrediction.ViewModel
{
  public class RegressionPrediction
  {
    public int Cycle { get; set; }
    public double PunchForce { get; set; }
    public double PunchDistance { get; set; }
    public double CumPunchDistance { get; set; }
    public double CumToolWearVol { get; set; }
    public string SheetMetalType { get; set; }
    public double PredictedToolWearVolume { get; set; }
    public double PredictionStandardDeviation { get; set; }
  }


  public class RegressionViewModel : ObservableObject, INotifyDataErrorInfo
  {
    private readonly RegressionModel _regressionModel = new RegressionModel();

    private readonly Dictionary<string, ICollection<string>>
    _validationErrors = new Dictionary<string, ICollection<string>>();

    private readonly List<RegressionPrediction> _predictionHistory =
      new List<RegressionPrediction>();

    private int _cycle;
    private double _punchForce;
    private double _punchDistance;
    private double _cumPunchDistance;
    private double _cumToolWearVol;

    private string _selectedSheetMetalType;
    private ObservableCollection<string> _sheetMetalTypes;

    private double _predictedToolWearVolume;
    private double _predictionStandardDeviation;

    public RegressionViewModel()
    {
      _sheetMetalTypes = new ObservableCollection<string>();
      _sheetMetalTypes.Add("Sheet Metal 1");
      _sheetMetalTypes.Add("Sheet Metal 2");
      _sheetMetalTypes.Add("Sheet Metal 3");
      SelectedSheetMetalType = "Sheet Metal 1";


      ErrorsChanged += RegressionViewModel_ErrorsChanged;
    }

    void RegressionViewModel_ErrorsChanged(object sender, DataErrorsChangedEventArgs e)
    {
      RaisePropertyChangedEvent("IsFormValid");
      RaisePropertyChangedEvent("HasErrors");
    }

    public int Cycle
    {
      get { return _cycle; }
      set
      {
        if (value < 0 || value > 10000)
        {
          _validationErrors["Cycle"] = new List<string>(
            new string[] { "Value out of range." });
          RaiseErrorsChanged("Cycle");
        }
        else
        {
```

```
      _validationErrors.Remove("Cycle");
      RaiseErrorsChanged("Cycle");
    }

    _cycle = value;
    RaisePropertyChangedEvent("Cycle");
  }
}

public double PunchForce
{
  get { return _punchForce; }
  set
  {
    _punchForce = value;
    RaisePropertyChangedEvent("PunchForce");
  }
}

public double PunchDistance
{
  get { return _punchDistance; }
  set
  {
    _punchDistance = value;
    RaisePropertyChangedEvent("PunchDistance");
  }
}

public double CumulativePunchDistance
{
  get { return _cumPunchDistance; }
  set
  {
    _cumPunchDistance = value;
    RaisePropertyChangedEvent("CumulativePunchDistance");
  }
}

public string SelectedSheetMetalType
{
  get { return _selectedSheetMetalType; }
  set
  {
    _selectedSheetMetalType = value;
    RaisePropertyChangedEvent("SelectedSheetMetalType");
  }
}

public int SelectedSheetMetalTypeIndex
{
  get
  {
    if (_selectedSheetMetalType == "Sheet Metal 1")
      return 0;
    else if (_selectedSheetMetalType == "Sheet Metal 2")
      return 1;
    else if (_selectedSheetMetalType == "Sheet Metal 3")
      return 2;
    else
      return -1;
  }
}

public ObservableCollection<string> SheetMetalTypes
{
  get { return _sheetMetalTypes; }
  set
  {
    _sheetMetalTypes = value;
    RaisePropertyChangedEvent("SheetMetalTypes");
  }
}

public double CumulativeToolWearVolume
{
  get { return _cumToolWearVol; }
  set
  {
    _cumToolWearVol = value;
    RaisePropertyChangedEvent("CumulativeToolWearVolume");
  }
}

public List<RegressionPrediction> History
{
  get { return _predictionHistory; }
}

public double PredictedToolWearVolume
{
  get { return _predictedToolWearVolume; }
  set
  {
    _predictedToolWearVolume = value;
    RaisePropertyChangedEvent("PredictedToolWearVolume");
  }
}

public double PredictionStandardDeviation
{
  get { return _predictionStandardDeviation; }
  set
  {
    _predictionStandardDeviation = value;
    RaisePropertyChangedEvent("PredictionStandardDeviation");
  }
}

public ICommand PredictCommand
{
  get { return new DelegateCommand(Predict); }
}

private void Predict()
{
  if (HasErrors)
    return;

  Tuple<double, double> predictionTuple =
    _regressionModel.GetPredictedToolWearOut(
    _cycle, _punchForce, _punchDistance, _cumPunchDistance,
    SelectedSheetMetalTypeIndex, _cumToolWearVol);
  PredictedToolWearVolume = predictionTuple.Item1;
  PredictionStandardDeviation = predictionTuple.Item2;

  _predictionHistory.Add(new RegressionPrediction
  {
    Cycle = Cycle,
    PunchForce = PunchForce,
    PunchDistance = PunchDistance,
    CumPunchDistance = CumulativePunchDistance,
    CumToolWearVol = CumulativeToolWearVolume,
    SheetMetalType = SelectedSheetMetalType,
    PredictedToolWearVolume = PredictedToolWearVolume,
    PredictionStandardDeviation = PredictionStandardDeviation
  });
}

public bool IsFormValid
```

```
    {
      get { return !HasErrors; }
    }


    #region INotifyDataErrorInfo members
    public event EventHandler<DataErrorsChangedEventArgs> ErrorsChanged;
    private void RaiseErrorsChanged(string propertyName)
    {
      if (ErrorsChanged != null)
        ErrorsChanged(this, new DataErrorsChangedEventArgs(propertyName));
    }

    public System.Collections.IEnumerable GetErrors(string propertyName)
    {
      if (string.IsNullOrEmpty(propertyName)
          || !_validationErrors.ContainsKey(propertyName))
        return null;

      return _validationErrors[propertyName];
    }

    public bool HasErrors
    {
      get { return _validationErrors.Count > 0; }
    }
    #endregion
  }
}
```

BinaryClassification.xaml

```
<Page
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:ViewModel="clr-namespace:ToolWearOutPrediction.ViewModel"
  x:Class="ToolWearOutPrediction.BinaryClassification"
  xmlns:View="clr-namespace:ToolWearOutPrediction.View"
  mc:Ignorable="d"
  d:DesignHeight="300" d:DesignWidth="300"
        Title="Binary Classification" Margin="10">

  <Page.DataContext>
    <ViewModel:BinaryClassificationViewModel/>
  </Page.DataContext>

  <Grid>
    <Grid.RowDefinitions>
      <RowDefinition Height="50"/>
      <RowDefinition Height="50"/>
      <RowDefinition Height="50"/>
      <RowDefinition Height="50"/>
      <RowDefinition Height="50"/>
      <RowDefinition Height="Auto"></RowDefinition>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
      <ColumnDefinition/>
      <ColumnDefinition Width="100px"/>
      <ColumnDefinition Width="60px"/>
    </Grid.ColumnDefinitions>

    <Grid.Background>
      <ImageBrush ImageSource="../PPlogo.png" Stretch="None"
                  AlignmentX="Left" AlignmentY="Top"/>
    </Grid.Background>
    <Label Grid.Column="0" Grid.Row="0" Grid.ColumnSpan="2" FontSize="20"
           FontWeight="Medium" HorizontalAlignment="Right"
           Content="Binary Classification"/>

    <TextBlock Grid.Column="0" Grid.Row="1" HorizontalAlignment="Right"
               VerticalAlignment="Center"
               FontWeight="Medium" FontSize="14" TextWrapping="Wrap"
               Width="150px" Text="Cycle:" TextAlignment="Right"/>
    <TextBox x:Name="Cycle" Grid.Column="1" Grid.Row="1"
             Margin="15,0,0,0" Height="25"
             Text="{Binding Cycle}" VerticalContentAlignment="Center"/>
    <Label Grid.Column="2" Grid.Row="1" HorizontalAlignment="Left"
           VerticalAlignment="Center"
           FontWeight="Medium" FontSize="14" Margin="5,0,0,0" Content="cycles"/>

    <TextBlock Grid.Column="0" Grid.Row="2"
               HorizontalAlignment="Right" VerticalAlignment="Center"
               FontWeight="Medium" FontSize="14" TextWrapping="Wrap" Width="150px"
               Text="Cumulative Punch Distance:" TextAlignment="Right"/>
    <TextBox Grid.Column="1" Grid.Row="2" Margin="15,0,0,0" Height="25"
             Text="{Binding CumulativePunchDistance}"
             VerticalContentAlignment="Center"/>
    <Label Grid.Column="2" Grid.Row="2" HorizontalAlignment="Left"
           VerticalAlignment="Center"
           FontWeight="Medium" FontSize="14" Margin="5,0,0,0" Content="mm"/>

    <TextBlock Grid.Column="0" Grid.Row="3" HorizontalAlignment="Right"
               VerticalAlignment="Center"
               FontWeight="Medium" FontSize="14" TextWrapping="Wrap" Width="150px"
               Text="Cumulative Tool Wear Volume:" TextAlignment="Right"/>
    <TextBox Name="txtBoxCumToolWearVol" Grid.Column="1" Grid.Row="3" Margin="15,0,0,0"
             Height="25" Text="{Binding CumulativeToolWearVolume}"
             VerticalContentAlignment="Center"/>
    <Label Grid.Column="2" Grid.Row="3" HorizontalAlignment="Left"
           VerticalAlignment="Center"
           FontWeight="Medium" FontSize="14" Margin="5,0,0,0" Content="mm3"/>

    <Button Name="BtnPredict" Grid.Column="0" Grid.Row="4" Grid.ColumnSpan="2"
            Width="{Binding ElementName=txtBoxCumToolWearVol, Path=ActualWidth}"
            Height="25" HorizontalAlignment="Right"
            Command="{Binding PredictCommand}" Click="Button_Click" Content="Predict"
            IsEnabled="{Binding IsFormValid}" PreviewMouseDown="Button_MouseDown"/>

    <View:ExpanderWithDataStatistics Grid.Column="0" Grid.ColumnSpan="3" Grid.Row="5"/>
  </Grid>
</Page>
```

BinaryClassificationResult.xaml

```
<Page x:Class="ToolWearOutPrediction.BinaryClassificationResult"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      mc:Ignorable="d"
      d:DesignHeight="300" d:DesignWidth="600"
            Title="Binary Classification Result">

  <Grid>
    <Grid.ColumnDefinitions>
      <ColumnDefinition></ColumnDefinition>
      <ColumnDefinition></ColumnDefinition>
    </Grid.ColumnDefinitions>

    <Grid.RowDefinitions>
      <RowDefinition Height="60"></RowDefinition>
      <RowDefinition Height="30"></RowDefinition>
      <RowDefinition Height="30"></RowDefinition>
      <RowDefinition Height="60"></RowDefinition>
      <RowDefinition></RowDefinition>
    </Grid.RowDefinitions>

    <Image Tag="{Binding PredictedLabel}"
           Grid.Column="1" Grid.RowSpan="4"
           HorizontalAlignment="Right">
      <Image.Style>
        <Style TargetType="Image">
          <Style.Triggers>
            <DataTrigger Binding="{Binding PredictedLabel}" Value="0">
              <Setter Property="Source"
                      Value="/ToolWEAR;component/2lights_green.png"/>
            </DataTrigger>
            <DataTrigger Binding="{Binding PredictedLabel}" Value="1">
              <Setter Property="Source"
                      Value="/ToolWEAR;component/2lights_red.png"/>
            </DataTrigger>
          </Style.Triggers>
        </Style>
      </Image.Style>
    </Image>

    <Label Grid.Column="0" Grid.Row="0" Grid.ColumnSpan="2"
           HorizontalAlignment="Center"
           Content="Binary Classification Prediction Results"
           FontSize="20" FontWeight="Medium" VerticalContentAlignment="Center"/>
    <Label Grid.Column="0" Grid.Row="1"
           Content="Predicted Label:" FontSize="14" FontWeight="Medium"
           HorizontalAlignment="Right" VerticalContentAlignment="Center"/>
    <Label Grid.Column="1" Grid.Row="1"
           Content="{Binding PredictedLabel, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"
           HorizontalAlignment="Left" VerticalContentAlignment="Center" />
    <Label Grid.Column="0" Grid.Row="2"  Content="Predicted Label Probability:"
           FontSize="14" FontWeight="Medium" HorizontalAlignment="Right"
           VerticalContentAlignment="Center"/>
    <Label Grid.Column="1" Grid.Row="2" Content="{Binding PredictedLabelProbability}"
           HorizontalAlignment="Left" VerticalContentAlignment="Center" />

    <Label Grid.Column="0" Grid.Row="3" Grid.ColumnSpan="2" HorizontalAlignment="Center"
           Content="Prediction History" FontSize="20" FontWeight="Medium"
           VerticalContentAlignment="Center"/>
    <DataGrid MaxHeight="200" MinHeight="200"
              Grid.Column="0" Grid.Row="4" Grid.ColumnSpan="2"
              Width="480" Margin="5,10" RowHeaderWidth="0" ItemsSource="{Binding History}"
              AutoGenerateColumns="False" FontSize="12" VerticalAlignment="Top"
              VerticalContentAlignment="Bottom" Background="White">
      <DataGrid.Columns>
        <DataGridTextColumn Header="Cycle" Binding="{Binding Cycle}" Width="70"/>
        <DataGridTextColumn Header="Cumulative.&#x0a;Punch Distance"
                            Binding="{Binding CumPunchDistance, StringFormat={}{0:n4}}"
                            Width="100"/>
        <DataGridTextColumn Header="Cumulative.&#x0a;Tool Wear Volume"
                            Binding="{Binding CumToolWearVol, StringFormat={}{0:n4}}"/>
        <DataGridTextColumn Header="Predicted Label"
                            Binding="{Binding PredictedLabel, StringFormat={}{0:n4}}"
                            FontWeight="Bold" FontSize="14"/>
        <DataGridTextColumn Header="Predicted Label&#x0a;Probability"
                            Binding="{Binding PredictedLabelProbability, StringFormat={}{0:n4}}"
                            Width="*"/>
      </DataGrid.Columns>
    </DataGrid>
  </Grid>
</Page>
```

ExpanderWithDataStatistics.xaml

```
<UserControl
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:ViewModel="clr-namespace:ToolWearOutPrediction.ViewModel"
  x:Class="ToolWearOutPrediction.View.ExpanderWithDataStatistics"
  mc:Ignorable="d"
  d:DesignHeight="300" d:DesignWidth="500">


  <Expander ExpandDirection="Down">
    <Expander.Header>Basic Data Statistics</Expander.Header>
    <Grid>
      <Grid.RowDefinitions>
        <RowDefinition Height="10"/>
        <RowDefinition Height="40"/>
        <RowDefinition Height="40"/>
        <RowDefinition Height="40"/>
        <RowDefinition Height="40"/>
        <RowDefinition Height="40"/>
        <RowDefinition Height="40"/>
      </Grid.RowDefinitions>
      <Grid.ColumnDefinitions>
        <ColumnDefinition Width="170"/>
        <ColumnDefinition Width="90"/>
        <ColumnDefinition Width="85"/>
        <ColumnDefinition Width="88"/>
      </Grid.ColumnDefinitions>


      <Label Grid.Column="1" Grid.Row="1" HorizontalAlignment="Right"
             VerticalAlignment="Center"
             FontWeight="Medium" FontSize="14" Content="Mean"/>
      <Label Grid.Column="2" Grid.Row="1" HorizontalAlignment="Right"
             VerticalAlignment="Center"
             FontWeight="Medium" FontSize="14" Content="Min"/>
      <Label Grid.Column="3" Grid.Row="1" HorizontalAlignment="Right"
             VerticalAlignment="Center"
             FontWeight="Medium" FontSize="14" Content="Max"/>

      <Label Grid.Column="0" Grid.Row="2" HorizontalAlignment="Right"
             VerticalAlignment="Center" FontWeight="Medium" FontSize="14"
             HorizontalContentAlignment="Right" Content="Last Cycle"/>
      <Label Grid.Column="1" Grid.Row="2" HorizontalAlignment="Right"
             VerticalAlignment="Center" Margin="5,0,0,0" FontWeight="Normal"
             FontSize="14" Content="{Binding MeanLastCycle}"/>
      <Label Grid.Column="2" Grid.Row="2" HorizontalAlignment="Right"
             VerticalAlignment="Center" Margin="5,0,0,0" FontWeight="Normal"
             FontSize="14" Content="{Binding MinLastCycle}"/>
      <Label Grid.Column="3" Grid.Row="2" HorizontalAlignment="Right"
             VerticalAlignment="Center" Margin="5,0,0,0" FontWeight="Normal"
             FontSize="14" Content="{Binding MaxLastCycle}"/>

      <Label Grid.Column="0" Grid.Row="3" HorizontalAlignment="Right"
             VerticalAlignment="Center" FontWeight="Medium" FontSize="14"
             HorizontalContentAlignment="Right" Content="Punch Distance"/>
      <Label Grid.Column="1" Grid.Row="3" HorizontalAlignment="Right"
             VerticalAlignment="Center" Margin="5,0,0,0" FontWeight="Normal"
             FontSize="14" Content="{Binding MeanPunchDistance}"/>
      <Label Grid.Column="2" Grid.Row="3" HorizontalAlignment="Right"
             VerticalAlignment="Center" Margin="5,0,0,0" FontWeight="Normal"
             FontSize="14" Content="{Binding MinPunchDistance}"/>
      <Label Grid.Column="3" Grid.Row="3" HorizontalAlignment="Right"
             VerticalAlignment="Center" Margin="5,0,0,0" FontWeight="Normal"
             FontSize="14" Content="{Binding MaxPunchDistance}"/>
      <TextBlock Grid.Column="0" Grid.Row="4" HorizontalAlignment="Right"
                 VerticalAlignment="Center" FontWeight="Medium" FontSize="14"
                 TextWrapping="Wrap" TextAlignment="Right">
        <Run Text="Cumulative Punch Distance at Fault"/>
      </TextBlock>
      <Label Grid.Column="1" Grid.Row="4" HorizontalAlignment="Right"
             VerticalAlignment="Center" Margin="5,0,0,0" FontWeight="Normal"
             FontSize="14" Content="{Binding MeanCumPunchDistanceAtFault}"/>
      <Label Grid.Column="2" Grid.Row="4" HorizontalAlignment="Right"
             VerticalAlignment="Center" Margin="5,0,0,0" FontWeight="Normal"
             FontSize="14" Content="{Binding MinCumPunchDistanceAtFault}"/>
      <Label Grid.Column="3" Grid.Row="4" HorizontalAlignment="Right"
             VerticalAlignment="Center" Margin="5,0,0,0" FontWeight="Normal"
             FontSize="14" Content="{Binding MaxCumPunchDistanceAtFault}"/>

      <Label Grid.Column="0" Grid.Row="5" HorizontalAlignment="Right"
             VerticalAlignment="Center" FontWeight="Medium" FontSize="14"
             HorizontalContentAlignment="Right" Content="Punch Force"/>
      <Label Grid.Column="1" Grid.Row="5" HorizontalAlignment="Right"
             VerticalAlignment="Center" Margin="5,0,0,0" FontWeight="Normal"
             FontSize="14" Content="{Binding MeanPunchForce}"/>
      <Label Grid.Column="2" Grid.Row="5" HorizontalAlignment="Right"
             VerticalAlignment="Center" Margin="5,0,0,0" FontWeight="Normal"
             FontSize="14" Content="{Binding MinPunchForce}"/>
      <Label Grid.Column="3" Grid.Row="5" HorizontalAlignment="Right"
             VerticalAlignment="Center" Margin="5,0,0,0" FontWeight="Normal"
             FontSize="14" Content="{Binding MaxPunchForce}"/>

      <TextBlock Grid.Column="0" Grid.Row="6" HorizontalAlignment="Right"
                 VerticalAlignment="Center" FontWeight="Medium" FontSize="14"
                 TextWrapping="Wrap" TextAlignment="Right">
        <Run Text="Cumulative Tool Wear Volume at Fault"/>
      </TextBlock>
      <Label Grid.Column="1" Grid.Row="6" HorizontalAlignment="Right"
             VerticalAlignment="Center" Margin="5,0,0,0" FontWeight="Normal"
             FontSize="14" Content="{Binding MeanCumToolWearVolAtFault}"/>
      <Label Grid.Column="2" Grid.Row="6" HorizontalAlignment="Right"
             VerticalAlignment="Center" Margin="5,0,0,0" FontWeight="Normal"
             FontSize="14" Content="{Binding MinCumToolWearVolAtFault}"/>
      <Label Grid.Column="3" Grid.Row="6" HorizontalAlignment="Right"
             VerticalAlignment="Center" Margin="5,0,0,0" FontWeight="Normal"
             FontSize="14" Content="{Binding MaxCumToolWearVolAtFault}"/>

    </Grid>
  </Expander>
</UserControl>
```

Home.xaml

```
<Page x:Class="ToolWearOutPrediction.Home"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  mc:Ignorable="d"
  d:DesignHeight="300" d:DesignWidth="500"
        Title="Home">

  <Grid Margin="10">
    <Grid.ColumnDefinitions>
      <ColumnDefinition />
      <ColumnDefinition />
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
      <RowDefinition Height="40"/>
      <RowDefinition Height="Auto"/>
      <RowDefinition />
      <RowDefinition Height="Auto"/>
    </Grid.RowDefinitions>

    <Grid.Background>
      <ImageBrush ImageSource="../PPlogo.png"
                  Stretch="None" AlignmentX="Left" AlignmentY="Top"/>
    </Grid.Background>

    <Label Grid.Column="1" Grid.Row="0"
           FontSize="20" FontWeight="Medium">ToolWEAR</Label>
    <!-- People list -->
    <Border Grid.Column="1" Grid.Row="1"
            Height="35" Padding="5" Background="#4E87D4">
      <Label VerticalAlignment="Center" Foreground="White">Methods</Label>
    </Border>
    <ListBox Name="methodsListBox" Grid.Column="1" Grid.Row="2">
      <ListBoxItem>Regression</ListBoxItem>
      <ListBoxItem>Binary Classification</ListBoxItem>
      <ListBoxItem>Multi-Class Classification</ListBoxItem>
    </ListBox>

    <!-- Select method button -->
    <Button Grid.Column="1" Grid.Row="3" Margin="0,10,0,0"
            Width="125" Height="25" HorizontalAlignment="Right"
            Click="Button_Click">Select</Button>
  </Grid>
</Page>
```

MainWindow.xaml

```
<NavigationWindow x:Class="ToolWearOutPrediction.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="{Binding WindowName}" Source="Home.xaml"
  DataContext="{Binding RelativeSource={RelativeSource Self}}"
  MinHeight="350" MinWidth="525"
  SizeToContent="WidthAndHeight" ResizeMode="NoResize">

</NavigationWindow>
```

MultiClassClassification.xaml

```
<Page
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:ViewModel="clr-namespace:ToolWearOutPrediction.ViewModel"
  x:Class="ToolWearOutPrediction.MultiClassClassification"
  xmlns:View="clr-namespace:ToolWearOutPrediction.View"
  mc:Ignorable="d"
  d:DesignHeight="300" d:DesignWidth="300"
        Title="Multi-Class Classification" Margin="10">

  <Page.DataContext>
    <ViewModel:MultiClassClassificationViewModel/>
  </Page.DataContext>

  <Grid>
    <Grid.RowDefinitions>
      <RowDefinition Height="50"/>
      <RowDefinition Height="50"/>
      <RowDefinition Height="50"/>
      <RowDefinition Height="50"/>
      <RowDefinition Height="50"/>
      <RowDefinition Height="Auto"></RowDefinition>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
      <ColumnDefinition/>
      <ColumnDefinition Width="100px"/>
      <ColumnDefinition Width="60px"/>
    </Grid.ColumnDefinitions>

    <Grid.Background>
      <ImageBrush ImageSource="../PPlogo.png" Stretch="None"
                  AlignmentX="Left" AlignmentY="Top"/>
    </Grid.Background>

    <TextBlock Grid.Column="0" Grid.Row="0" Grid.ColumnSpan="2"
               FontSize="20" FontWeight="Medium" HorizontalAlignment="Right"
               TextWrapping="Wrap" Width="200px"
               Text="Multi-Class Classification"
               TextAlignment="Right"/>

    <TextBlock Grid.Column="0" Grid.Row="1"
               HorizontalAlignment="Right" VerticalAlignment="Center"
               FontWeight="Medium" FontSize="14" TextWrapping="Wrap"
               Width="150px" Text="Cycle:" TextAlignment="Right"/>
    <TextBox x:Name="Cycle" Grid.Column="1" Grid.Row="1" Margin="15,0,0,0"
             Height="25" Text="{Binding Cycle}" VerticalContentAlignment="Center"/>
    <Label Grid.Column="2" Grid.Row="1" HorizontalAlignment="Left"
           VerticalAlignment="Center" FontWeight="Medium"
           FontSize="14" Margin="5,0,0,0" Content="cycles"/>

    <TextBlock Grid.Column="0" Grid.Row="2" HorizontalAlignment="Right"
               VerticalAlignment="Center" FontWeight="Medium" FontSize="14"
               TextWrapping="Wrap" Width="150px"
               Text="Cumulative Punch Distance:" TextAlignment="Right"/>
    <TextBox Grid.Column="1" Grid.Row="2" Margin="15,0,0,0" Height="25"
             Text="{Binding CumulativePunchDistance}"
             VerticalContentAlignment="Center"/>
    <Label Grid.Column="2" Grid.Row="2" HorizontalAlignment="Left"
           VerticalAlignment="Center" FontWeight="Medium"
           FontSize="14" Margin="5,0,0,0" Content="mm"/>

    <TextBlock Grid.Column="0" Grid.Row="3" HorizontalAlignment="Right"
               VerticalAlignment="Center" FontWeight="Medium" FontSize="14"
               TextWrapping="Wrap" Width="150px"
               Text="Cumulative Tool Wear Volume:" TextAlignment="Right"/>
    <TextBox Name="txtBoxCumToolWearVol" Grid.Column="1" Grid.Row="3"
             Margin="15,0,0,0" Height="25"
             Text="{Binding CumulativeToolWearVolume}"
             VerticalContentAlignment="Center"/>
    <Label Grid.Column="2" Grid.Row="3" HorizontalAlignment="Left"
           VerticalAlignment="Center" FontWeight="Medium"
           FontSize="14" Margin="5,0,0,0" Content="mm3"/>

    <Button Name="BtnPredict" Grid.Column="0" Grid.Row="4" Grid.ColumnSpan="2"
            Width="{Binding ElementName=txtBoxCumToolWearVol, Path=ActualWidth}"
            Height="25" HorizontalAlignment="Right"
                        Command="{Binding PredictCommand}" Click="Button_Click"
            Content="Predict" IsEnabled="{Binding IsFormValid}"
            PreviewMouseDown="Button_MouseDown"/>

    <View:ExpanderWithDataStatistics Grid.Column="0" Grid.ColumnSpan="3" Grid.Row="5"/>
  </Grid>
</Page>
```

MultiClassClassificationResult.xaml

```
<Page x:Class="ToolWearOutPrediction.MultiClassClassificationResult"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  mc:Ignorable="d"
  d:DesignHeight="300" d:DesignWidth="600"
  Title="Multi-Class Classification Result">

  <Grid>
    <Grid.ColumnDefinitions>
      <ColumnDefinition></ColumnDefinition>
      <ColumnDefinition></ColumnDefinition>
    </Grid.ColumnDefinitions>

    <Grid.RowDefinitions>
      <RowDefinition Height="60"></RowDefinition>
      <RowDefinition Height="30"></RowDefinition>
      <RowDefinition Height="30"></RowDefinition>
      <RowDefinition Height="30"></RowDefinition>
      <RowDefinition Height="30"></RowDefinition>
      <RowDefinition Height="60"></RowDefinition>
      <RowDefinition></RowDefinition>
    </Grid.RowDefinitions>

    <Image Tag="{Binding PredictedLabel}" Grid.Column="1" Grid.Row="1"
           Grid.RowSpan="5" HorizontalAlignment="Right" VerticalAlignment="Bottom">
      <Image.Style>
        <Style TargetType="Image">
          <Style.Triggers>
            <DataTrigger Binding="{Binding PredictedLabel}" Value="0">
              <Setter Property="Source" Value="/ToolWEAR;component/3lights_green.png"/>
            </DataTrigger>
            <DataTrigger Binding="{Binding PredictedLabel}" Value="1">
              <Setter Property="Source" Value="/ToolWEAR;component/3lights_yellow.png"/>
            </DataTrigger>
            <DataTrigger Binding="{Binding PredictedLabel}" Value="2">
              <Setter Property="Source" Value="/ToolWEAR;component/3lights_red.png"/>
            </DataTrigger>
          </Style.Triggers>
        </Style>
      </Image.Style>
    </Image>

    <Label Grid.Column="0" Grid.Row="0" Grid.ColumnSpan="2"
           HorizontalAlignment="Center"
           Content="Multi-Class Classification Prediction Results"
           FontSize="20" FontWeight="Medium" VerticalContentAlignment="Center"/>
    <Label Grid.Column="0" Grid.Row="1"  Content="Predicted Label:"
           FontSize="14" FontWeight="Medium" HorizontalAlignment="Right"
           VerticalContentAlignment="Center"/>
    <Label Grid.Column="1" Grid.Row="1" Content="{Binding PredictedLabel}"
           HorizontalAlignment="Left" VerticalContentAlignment="Center" />
    <Label Grid.Column="0" Grid.Row="2"  Content="Class 0 Probability:"
           FontSize="14" FontWeight="Medium" HorizontalAlignment="Right"
           VerticalContentAlignment="Center"/>
    <Label Grid.Column="1" Grid.Row="2" Content="{Binding PredictedClass0Probability}"
           HorizontalAlignment="Left" VerticalContentAlignment="Center" />
    <Label Grid.Column="0" Grid.Row="3"  Content="Class 1 Probability:" FontSize="14"
           FontWeight="Medium" HorizontalAlignment="Right"
           VerticalContentAlignment="Center"/>
    <Label Grid.Column="1" Grid.Row="3" Content="{Binding PredictedClass1Probability}"
           HorizontalAlignment="Left" VerticalContentAlignment="Center" />
    <Label Grid.Column="0" Grid.Row="4"  Content="Class 2 Probability:" FontSize="14"
           FontWeight="Medium" HorizontalAlignment="Right"
           VerticalContentAlignment="Center"/>
    <Label Grid.Column="1" Grid.Row="4" Content="{Binding PredictedClass2Probability}"
```

```
          HorizontalAlignment="Left" VerticalContentAlignment="Center" />

    <Label Grid.Column="0" Grid.Row="5" Grid.ColumnSpan="2"
          HorizontalAlignment="Center"  Content="Prediction History"
          FontSize="20" FontWeight="Medium" VerticalContentAlignment="Center"/>
    <DataGrid MaxHeight="200" MinHeight="200" Grid.Column="0" Grid.Row="6"
             Grid.ColumnSpan="2" Width="480" Margin="5,10" RowHeaderWidth="0"
             ItemsSource="{Binding History}"
             AutoGenerateColumns="False" FontSize="12"
             VerticalAlignment="Top" VerticalContentAlignment="Bottom"
             Background="White">
      <DataGrid.Columns>
        <DataGridTextColumn
          Header="Cycle" Binding="{Binding Cycle}" Width="50"/>
        <DataGridTextColumn
          Header="Cumulative&#x0a;Punch Distance"
          Binding="{Binding CumPunchDistance, StringFormat={}{0:n4}}"/>
        <DataGridTextColumn
          Header="Cumulative&#x0a;Tool Wear&#x0a;Volume"
          Binding="{Binding CumToolWearVol, StringFormat={}{0:n4}}"
          Width="100"/>
        <DataGridTextColumn
          Header="Predicted&#x0a;Label"
          Binding="{Binding PredictedLabel, StringFormat={}{0:n4}}"
          FontWeight="Bold" FontSize="14"/>
        <DataGridTextColumn
          Header="Predicted&#x0a;Class0&#x0a;Prob."
          Binding="{Binding PredictedClass0Probability, StringFormat={}{0:n4}}"/>
        <DataGridTextColumn
          Header="Predicted&#x0a;Class1&#x0a;Prob."
          Binding="{Binding PredictedClass1Probability, StringFormat={}{0:n4}}"/>
        <DataGridTextColumn
          Header="Predicted&#x0a;Class2&#x0a;Prob."
          Binding="{Binding PredictedClass2Probability, StringFormat={}{0:n4}}"
          Width="*"/>
      </DataGrid.Columns>
    </DataGrid>
  </Grid>
</Page>
```

Regression.xaml

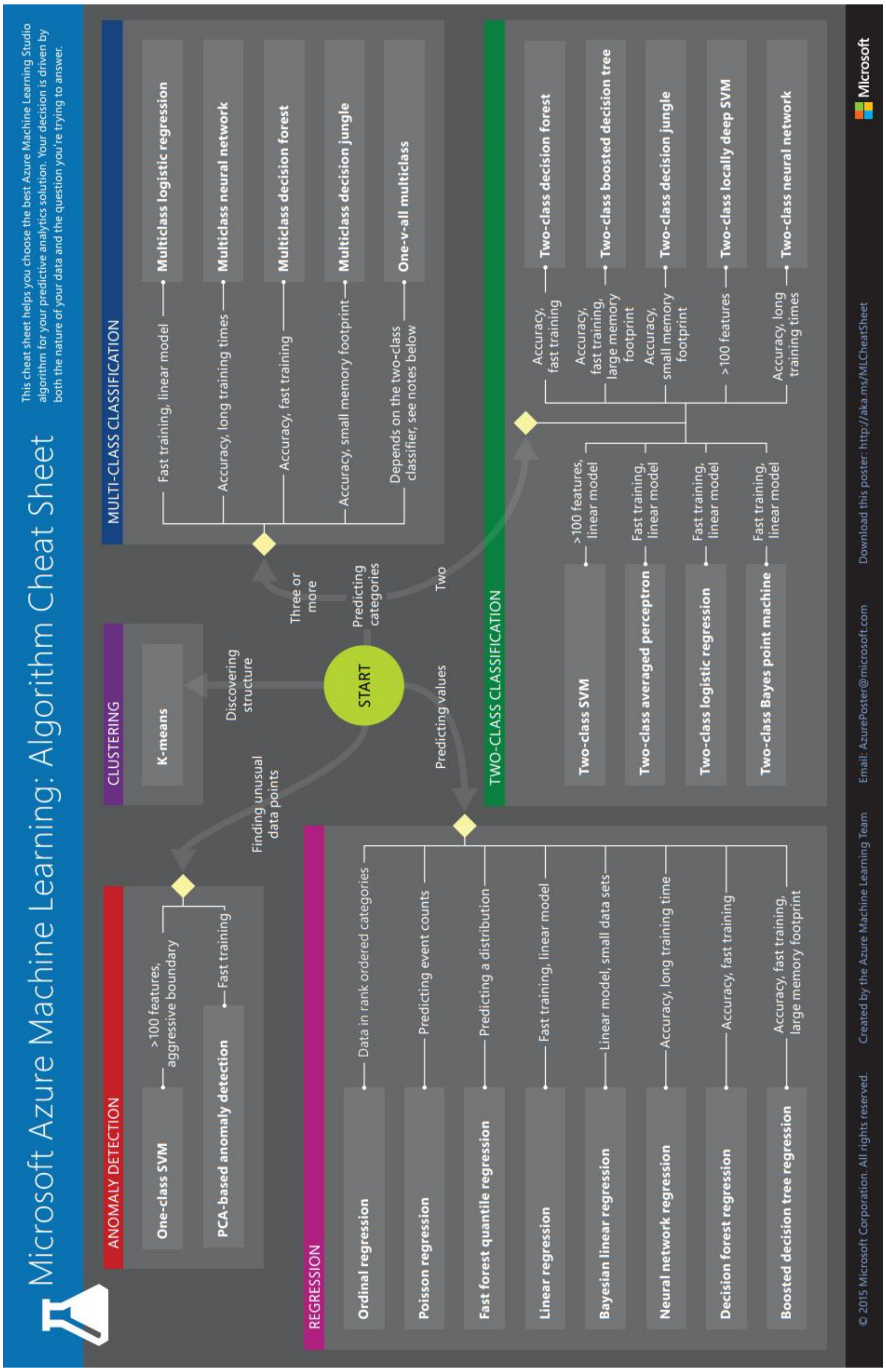
```
<Page
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:ViewModel="clr-namespace:ToolWearOutPrediction.ViewModel"
  x:Class="ToolWearOutPrediction.Regression"
  xmlns:View="clr-namespace:ToolWearOutPrediction.View"
  mc:Ignorable="d"
  d:DesignHeight="800" d:DesignWidth="600"
  Title="Regression" Margin="10"
  >

  <Page.DataContext>
    <ViewModel:RegressionViewModel/>
  </Page.DataContext>

  <Grid>
    <Grid.RowDefinitions>
      <RowDefinition Height="40"></RowDefinition>
      <RowDefinition></RowDefinition>
      <RowDefinition></RowDefinition>
      <RowDefinition></RowDefinition>
      <RowDefinition></RowDefinition>
      <RowDefinition></RowDefinition>
      <RowDefinition></RowDefinition>
      <RowDefinition></RowDefinition>
      <RowDefinition Height="Auto"></RowDefinition>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
      <ColumnDefinition Name="FirstColumn" Width="310"></ColumnDefinition>
      <ColumnDefinition Name="SecondColumn" Width="120"></ColumnDefinition>
      <ColumnDefinition Width="58"/>
    </Grid.ColumnDefinitions>

    <Grid.Background>
      <ImageBrush ImageSource="../PPlogo.png" Stretch="None"
                  AlignmentX="Left" AlignmentY="Top"/>
    </Grid.Background>

    <Label Grid.Column="0" Grid.Row="0" Grid.ColumnSpan="2"
           FontSize="20" FontWeight="Medium"
           HorizontalAlignment="Right">Regression</Label>

    <Label Grid.Column="0" Grid.Row="1" HorizontalAlignment="Right"
           VerticalAlignment="Center" FontWeight="Medium" FontSize="14">Cycle:</Label>
    <TextBox Name="Cycle" Grid.Column="1" Grid.Row="1"
             Margin="15,0,0,0" Height="25"
             Text="{Binding Cycle}" VerticalContentAlignment="Center"/>
    <Label Grid.Column="2" Grid.Row="1" HorizontalAlignment="Left"
           VerticalAlignment="Center" FontWeight="Medium"
           FontSize="14" Margin="5,0,0,0" Content="cycles"/>

    <Label Grid.Column="0" Grid.Row="2" HorizontalAlignment="Right"
           VerticalAlignment="Center" FontWeight="Medium"
           FontSize="14">Punch Force:</Label>
    <TextBox Grid.Column="1" Grid.Row="2" Margin="15,0,0,0"
             Height="25" Text="{Binding PunchForce}"
             VerticalContentAlignment="Center"/>
    <Label Grid.Column="2" Grid.Row="2" HorizontalAlignment="Left"
           VerticalAlignment="Center" FontWeight="Medium"
           FontSize="14" Margin="5,0,0,0" Content="N"/>

    <Label Grid.Column="0" Grid.Row="3" HorizontalAlignment="Right"
           VerticalAlignment="Center" FontWeight="Medium"
           FontSize="14">Punch Distance:</Label>
    <TextBox Grid.Column="1" Grid.Row="3" Margin="15,0,0,0"
             Height="25" Text="{Binding PunchDistance}"
             VerticalContentAlignment="Center"/>
    <Label Grid.Column="2" Grid.Row="3" HorizontalAlignment="Left"
           VerticalAlignment="Center" FontWeight="Medium"
           FontSize="14" Margin="5,0,0,0" Content="mm"/>

    <Label Grid.Column="0" Grid.Row="4" HorizontalAlignment="Right"
           VerticalAlignment="Center" FontWeight="Medium"
           FontSize="14">Cumulative Punch Distance:</Label>
    <TextBox Grid.Column="1" Grid.Row="4" Margin="15,0,0,0"
             Height="25" Text="{Binding CumulativePunchDistance}"
             VerticalContentAlignment="Center"/>
    <Label Grid.Column="2" Grid.Row="4" HorizontalAlignment="Left"
           VerticalAlignment="Center" FontWeight="Medium"
           FontSize="14" Margin="5,0,0,0" Content="mm"/>

    <Label Grid.Column="0" Grid.Row="5" HorizontalAlignment="Right"
           VerticalAlignment="Center" FontWeight="Medium"
           FontSize="14">Sheet Metal Type:</Label>
    <ComboBox Grid.Column="1" Grid.Row="5" Margin="15,0,0,0"
              Height="25" ItemsSource="{Binding SheetMetalTypes}"
              SelectedItem="{Binding Mode=TwoWay, Path=SelectedSheetMetalType}"
              VerticalContentAlignment="Center"/>

    <Label Grid.Column="0" Grid.Row="6" HorizontalAlignment="Right"
           VerticalAlignment="Center" FontWeight="Medium"
           FontSize="14">Cumulative Tool Wear Volume:</Label>
    <TextBox Name="txtBoxCumToolWearVol" Grid.Column="1"
```

```
             Grid.Row="6" Margin="15,0,0,0" Height="25"
             Text="{Binding CumulativeToolWearVolume}"
             VerticalContentAlignment="Center"/>
    <Label Grid.Column="2" Grid.Row="6" HorizontalAlignment="Left"
           VerticalAlignment="Center" FontWeight="Medium"
           FontSize="14" Margin="5,0,0,0" Content="mm3"/>

    <Button Name="BtnPredict" Grid.Column="0" Grid.Row="7"
            Grid.ColumnSpan="2"
            Width="{Binding ElementName=txtBoxCumToolWearVol, Path=ActualWidth}"
            Height="25" HorizontalAlignment="Right"
            Command="{Binding PredictCommand}"
            Click="Button_Click" Content="Predict"
            IsEnabled="{Binding IsFormValid}"
            Margin="0,10" PreviewMouseDown="Button_MouseDown"/>


    <View:ExpanderWithDataStatistics Grid.Column="0" Grid.ColumnSpan="3" Grid.Row="8"/>
  </Grid>
</Page>
```

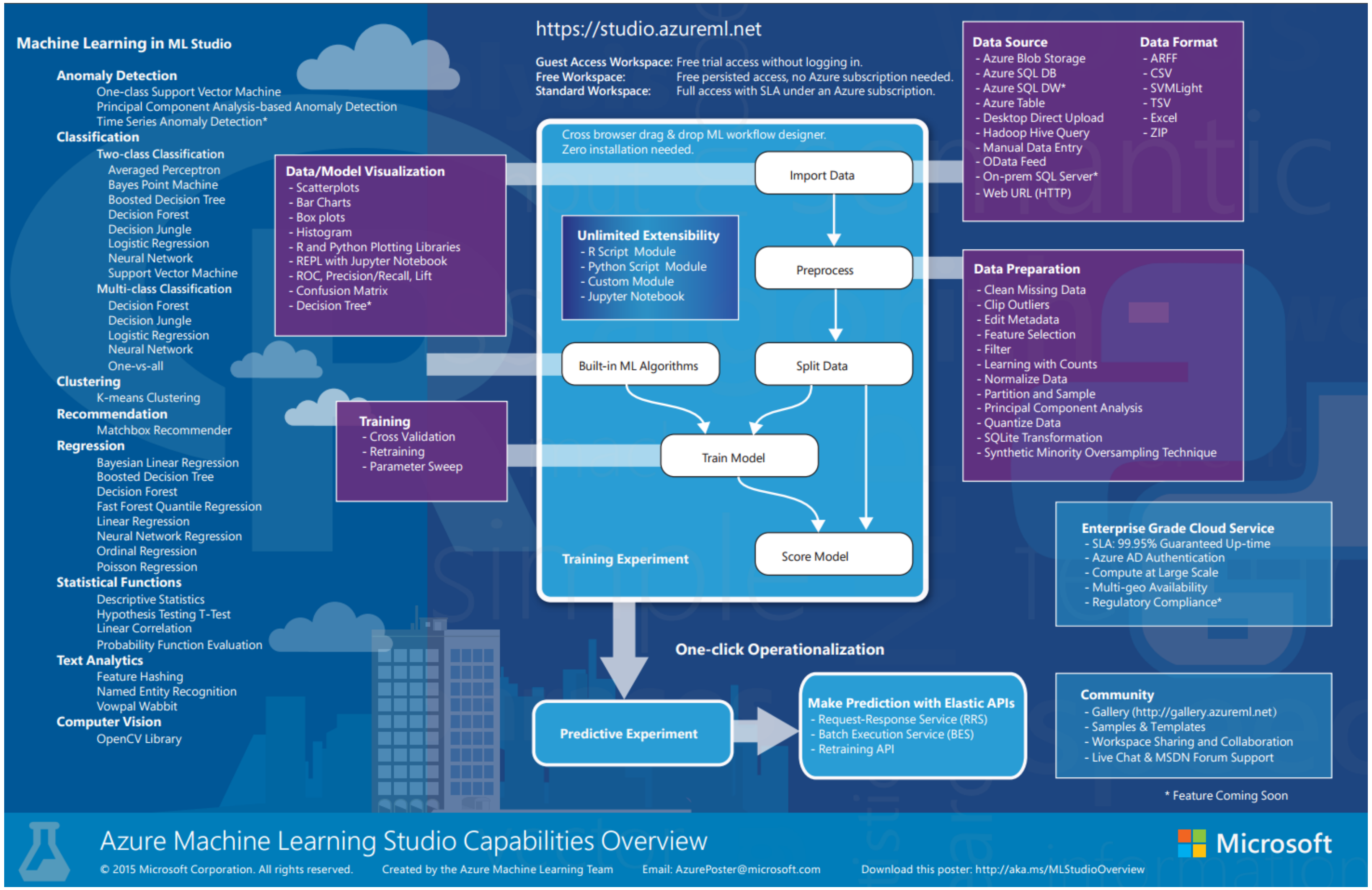
RegressionResult.xaml

```
<Page
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:ViewModel="clr-namespace:ToolWearOutPrediction.ViewModel"
  x:Class="ToolWearOutPrediction.RegressionResult"
  mc:Ignorable="d"
  d:DesignHeight="300" d:DesignWidth="600"
  Title="Regression - Result">

  <Grid Name="MainGrid">
    <Grid.ColumnDefinitions>
      <ColumnDefinition></ColumnDefinition>
      <ColumnDefinition></ColumnDefinition>
    </Grid.ColumnDefinitions>

    <Grid.RowDefinitions>
      <RowDefinition Height="60"></RowDefinition>
      <RowDefinition Height="30"></RowDefinition>
      <RowDefinition Height="30"></RowDefinition>
      <RowDefinition Height="40"></RowDefinition>
      <RowDefinition></RowDefinition>
    </Grid.RowDefinitions>

    <Label Grid.Column="0" Grid.Row="0" Grid.ColumnSpan="2" HorizontalAlignment="Center"
           Content="Regression Prediction Results" FontSize="20"
           FontWeight="Medium" VerticalContentAlignment="Center"/>
    <Label Grid.Column="0" Grid.Row="1" Content="Predicted Tool Wear Out Volume:"
           FontSize="14" FontWeight="Medium" HorizontalAlignment="Right"
           VerticalContentAlignment="Center"/>
    <Label Grid.Column="1" Grid.Row="1" Content="{Binding PredictedToolWearVolume,
      Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}" HorizontalAlignment="Left"
           VerticalContentAlignment="Center" />
    <Label Grid.Column="0" Grid.Row="2" Content="Prediction Standard Deviation:"
           FontSize="14" FontWeight="Medium" HorizontalAlignment="Right"
           VerticalContentAlignment="Center"/>
    <Label Grid.Column="1" Grid.Row="2" Content="{Binding PredictionStandardDeviation}"
           HorizontalAlignment="Left" VerticalContentAlignment="Center" />

    <Label Grid.Column="0" Grid.Row="3" Grid.ColumnSpan="2" HorizontalAlignment="Center"
           Content="Prediction History" FontSize="20" FontWeight="Medium"
           VerticalContentAlignment="Center"/>
    <DataGrid MaxHeight="200" MinHeight="200" Grid.Column="0" Grid.Row="4" Grid.ColumnSpan="2"
              Width="480" Margin="5,10" RowHeaderWidth="0" ItemsSource="{Binding History}"
              AutoGenerateColumns="False" FontSize="12" VerticalAlignment="Top"
              VerticalContentAlignment="Bottom" Background="White">
      <DataGrid.Columns>
        <DataGridTextColumn Header="Cycle" Binding="{Binding Cycle}"/>
        <DataGridTextColumn Header="Punch&#x0a;Force"
                            Binding="{Binding PunchForce, StringFormat={}{0:n4}}"/>
        <DataGridTextColumn Header="Punch&#x0a;Distance"
                            Binding="{Binding PunchDistance, StringFormat={}{0:n4}}"/>
        <DataGridTextColumn Header="Cumul.&#x0a;Punch&#x0a;Distance"
                            Binding="{Binding CumPunchDistance, StringFormat={}{0:n4}}"/>
        <DataGridTextColumn Header="Cumul.&#x0a;Tool Wear&#x0a;Volume"
                            Binding="{Binding CumToolWearVol, StringFormat={}{0:n4}}"/>
        <DataGridTextColumn Header="Sheet&#x0a;Metal&#x0a;Type"
                            Binding="{Binding SheetMetalType}"/>
        <DataGridTextColumn Header="Predicted&#x0a;Tool Wear&#x0a;Volume"
                            Binding="{Binding PredictedToolWearVolume, StringFormat={}{0:n4}}"
                            FontWeight="Bold" FontSize="14"/>
        <DataGridTextColumn Header="Prediction&#x0a;Standard&#x0a;Deviation"
                          Binding="{Binding PredictionStandardDeviation, StringFormat={}{0:n4}}"
                          Width="*"/>
      </DataGrid.Columns>
    </DataGrid>
  </Grid>
</Page>
```

APPENDIX 6.


Microsoft Azure Machine Learning: Algorithm Cheat Sheet
This cheat sheet helps you choose the best Azure Machine Learning Studio algorithm for your predictive analytics solution. Your decision is driven by both the nature of your data and the question you're trying to answer.
START
Finding unusual data points
Discovering structure
Predicting categories
Three or more
Two
Predicting values
ANOMALY DETECTION
One-class SVM
>100 features, aggressive boundary
PCA-based anomaly detection
Fast training
CLUSTERING
K-means
MULTI-CLASS CLASSIFICATION
Fast training, linear model
Multiclass logistic regression
Accuracy, long training times
Multiclass neural network
Accuracy, fast training
Multiclass decision forest
Accuracy, small memory footprint
Multiclass decision jungle
Depends on the two-class classifier, see notes below
One-v-all multiclass
REGRESSION
Ordinal regression
Data in rank ordered categories
Poisson regression
Predicting event counts
Fast forest quantile regression
Predicting a distribution
Linear regression
Fast training, linear model
Bayesian linear regression
Linear model, small data sets
Neural network regression
Accuracy, long training time
Decision forest regression
Accuracy, fast training
Boosted decision tree regression
Accuracy, fast training, large memory footprint
TWO-CLASS CLASSIFICATION
Two-class SVM
>100 features, linear model
Two-class averaged perceptron
Fast training, linear model
Two-class logistic regression
Fast training, linear model
Two-class Bayes point machine
Fast training, linear model
Accuracy, fast training
Two-class decision forest
Accuracy, fast training, large memory footprint
Two-class boosted decision tree
Accuracy, small memory footprint
Two-class decision jungle
>100 features
Two-class locally deep SVM
Accuracy, long training times
Two-class neural network
© 2015 Microsoft Corporation. All rights reserved.
Created by the Azure Machine Learning Team
Email: AzurePoster@microsoft.com
Download this poster: http://aka.ms/MLCheatSheet
Microsoft

APPENDIX 7.



Machine Learning in ML Studio

Anomaly Detection
One-class Support Vector Machine
Principal Component Analysis-based Anomaly Detection
Time Series Anomaly Detection*

Classification

Two-class Classification
Averaged Perceptron
Bayes Point Machine
Boosted Decision Tree
Decision Forest
Decision Jungle
Logistic Regression
Neural Network
Support Vector Machine

Multi-class Classification
Decision Forest
Decision Jungle
Logistic Regression
Neural Network
One-vs-all

Clustering
K-means Clustering

Recommendation
Matchbox Recommender

Regression
Bayesian Linear Regression
Boosted Decision Tree
Decision Forest
Fast Forest Quantile Regression
Linear Regression
Neural Network Regression
Ordinal Regression
Poisson Regression

Statistical Functions
Descriptive Statistics
Hypothesis Testing T-Test
Linear Correlation
Probability Function Evaluation

Text Analytics
Feature Hashing
Named Entity Recognition
Vowpal Wabbit

Computer Vision
OpenCV Library

Data/Model Visualization
- Scatterplots
- Bar Charts
- Box plots
- Histogram
- R and Python Plotting Libraries
- REPL with Jupyter Notebook
- ROC, Precision/Recall, Lift
- Confusion Matrix
- Decision Tree*

Training
- Cross Validation
- Retraining
- Parameter Sweep

https://studio.azureml.net

Guest Access Workspace: Free trial access without logging in.
Free Workspace: Free persisted access, no Azure subscription needed.
Standard Workspace: Full access with SLA under an Azure subscription.

Cross browser drag & drop ML workflow designer.
Zero installation needed.

Unlimited Extensibility
- R Script Module
- Python Script Module
- Custom Module
- Jupyter Notebook

Make Prediction with Elastic APIs
- Request-Response Service (RRS)
- Batch Execution Service (BES)
- Retraining API

Data Source
- Azure Blob Storage
- Azure SQL DB
- Azure SQL DW*
- Azure Table
- Desktop Direct Upload
- Hadoop Hive Query
- Manual Data Entry
- OData Feed
- On-prem SQL Server*
- Web URL (HTTP)

Data Format
- ARFF
- CSV
- SVMLight
- TSV
- Excel
- ZIP

Data Preparation
- Clean Missing Data
- Clip Outliers
- Edit Metadata
- Feature Selection
- Filter
- Learning with Counts
- Normalize Data
- Partition and Sample
- Principal Component Analysis
- Quantize Data
- SQLite Transformation
- Synthetic Minority Oversampling Technique

Enterprise Grade Cloud Service
- SLA: 99.95% Guaranteed Up-time
- Azure AD Authentication
- Compute at Large Scale
- Multi-geo Availability
- Regulatory Compliance*

Community
- Gallery (http://gallery.azureml.net)
- Samples & Templates
- Workspace Sharing and Collaboration
- Live Chat & MSDN Forum Support

* Feature Coming Soon

Azure Machine Learning Studio Capabilities Overview

Microsoft

© 2015 Microsoft Corporation. All rights reserved. Created by the Azure Machine Learning Team Email: AzurePoster@microsoft.com Download this poster: http://aka.ms/MLStudioOverview