Amna Khan

# COMPARISON OF MACHINE LEARNING APPROACHES FOR CLASSIFICATION OF INVOICES

Faculty of Information Technology and Communication Sciences

Master of Science Thesis

December 2019

## ABSTRACT

Amna Khan: Comparison of Machine Learning Approaches for Classification of Invoices
Master of Science Thesis
Tampere University
Computational Big Data Analytics
December 2019

Machine learning has become one of the leading sciences governing modern world. Various disciplines specifically neural networks have recently gained a lot of attention due to its widespread applications. With the recent advances in the technology the resulting big data has augmented the need of bigger means of storage, analysis and henceforth utilization. This not only implies the efficient use of available techniques but suggests surge in the development of new algorithms and techniques. In this project, three different machine learning approaches were implemented utilizing the open source library of keras on TensorFlow as a proof of concept for the task of intelligent invoice automation. The performance of these approaches for improved business on data of invoices has been analysed using the data of two customers with two target attributes per customer as a dataset. The behaviour of neural network hyper-parameters using matplotlib and TensorBoard was empirically calculated and investigated. As part of the first approach, the standard way of implementing predictive algorithm using neural network was followed. Moreover, the hyper-parameters search space was fine-tuned, and the resulting model was studied by grid search on those hyper-parameters. This strategy of hyper-parameters was followed in the next two approaches as well. In the second approach, not only further possible improvement in prediction accuracy is achieved but also the dependency between the two target attributes by using multi-task learning was determined. As per the third implemented approach, the use of continual learning on invoices for postings was analysed. This investigation, that involves the comparison of varied machine learning approaches has broad significance in approving the currently available algorithms for handling such data and suggests means for improvement as well. It holds great prospects, including but not limited to future implementation of such approaches in the domain of finance towards improved customer experience, fraud detection and ease in the assessments of assets etc.

Keywords: Machine Learning, Invoice Prediction, Neural Networks, Multi-task learning, Continual Learning, Deep Learning in Finance

# Table of Contents

# 1 Introduction:

For decades, in the finance literature there had been limited application of machine learning. But as time is progressing, the financial sector has started to see a tremendous increase in the use-cases of machine learning. Today, machine learning has begun to play a major role in the domain of finance from approving loans, to managing assets, to assessing risks. Many companies are using machine learning in their daily business routines. As per the applications in various aspects the use of machine learning approaches not only saves time but also gives a high turnover in their annual budgeting and profit targets, depending on level of accuracy of the machine learning algorithm or platform they are using and its relevant domain of the problem. Recent technological advancements particularly rise of big data, processing power and the 2008 financial crisis, have justified the increased potential of machine learning approaches in various domains of finance (1).

There is a huge variety of financial assistance/services being offered by the financial institutions targeting not only individuals but businesses as well. The services and the financial institutions are explained in detail in the Chapter 2. All these financial services require some types of transactions to be carried out. With the availability of internet platforms, the number of transactions has increased multifold. Carrying out the correct transactions in minimum time is one of the key descriptors that give these businesses an edge over its competitors.

The process for a transaction usually starts when an invoice is generated. A company providing some monthly services to several customers requires sending their customer a bill/invoice that will notify the customer of the due amount to be paid. The generation of invoices from vendor to customer multiplies exponentially as the businesses scales up and customers increase. To manage this huge number of invoice generation, many companies hire people or outsource this tedious task. Institutions providing financial services are the ones who have to deal with the invoice posting.

OpusCapita is one such example of a case, which deals with posting of hundreds of invoices. Currently, this tiresome task is being done manually by teams, who determine the posting parameters on the basis of their expert opinion. We have tried to automate this wearisome task of manually determining the posting parameters by using machine learning. We have selected 2 posting parameters as the target classes for our problem.

Pertaining the data privacy concerns, we will not be using their real names but will refer to them as target variable 1 and target variable 2. We have implemented a predictive algorithm that could predict the values of target classes in invoices datasets. We also compared our algorithm with various other machine learning approaches and analysed its' behaviour in predicting the target classes.

Financial businesses have started harnessing the power of machine leaning in majority of their business domains. Based on the nature of the use cases, bankruptcy and fraud detection has remained the most significant topic and has been studied widely and consistently for the last few decades (2). The market is full of the variety of the machine learning API's with cloud storage from the leading vendors. Microsoft is one amongst such vendors. OpusCapita has begun using Microsoft Azure machine learning studio for intra company predictive analytics. This tool has proved to be relatively useful, but this usefulness comes with a certain degree of restriction on the freedom as well.

Amongst the basic information available about the working of the tool in its documentation guide, the tool still acts as a black box to the end user. This lack of awareness about the internal working of the tool doubles when a technical domain expert of machine learning has to make guesses, about the several technical aspects of the machine learning algorithm and then adjust the limited visible parameters in correspondence. To tackle this issue, OpusCapita has made its own internal classification platform however; the availability of predictive algorithm is still void.

This project begins with the discussion on the background of neural networks origin. In the first part the concept of biological neurons that led to the concept of neural networks is discussed. There on the concept of artificial neural networks upon which this whole thesis is based on has been described in the 2nd Chapter. Not only has its structure been deciphered in detail but also its parameters, hyper parameters and various types have been elucidated.

In the 3rd chapter, the approaches used previously before feed forward neural networks were implemented have been listed in details. The literature is reviewed regarding the work done in this thesis.

In the 4th chapter, the discussion moves on to our empirical work. The tools and technology that has been used is elucidated in detail and the behaviour of neural network's hyper-parameters is empirically shown.

In the 5th chapter, the behaviour of the implemented algorithm is analysed, and its performance is compared with Azure ML studio performance, for all the three approaches for both data sets.

The 6th chapter provides a conclusion to this work and determines the other strategies that can also be applied on the basis of current work.

# 2 Background:

## 2.1 Finance:

In the financial sector, several types of businesses operate providing a wide range of services. Typical bodies, providing services could be banks, insurance companies, financial firms, brokerages, credit unions etc. Some of the key services provided by the commonly known financial institutions are:

- Retail banks:
  That usually deals in products targeting individual consumers.
- Commercial Banks:
  It provides services to business for commercial use.
- Brokerage:
  It assists in trading of stocks through its stockbrokers.
- Insurance Companies:
  It aids in providing financial security against natural mishaps to individuals.
- Mortgage Companies:
  As the name indicates, the institution provides loans for mortgages to either individuals or consumers.
- Internet banking:
  In addition to the afore mentioned major financial institutions, a new way of banking has also been introduced with the name of Internet banking. It deals in same businesses as the conventional financial institutions, but the services are additionally facilitated through internet API's (application programming interface).

OpusCapita is one such organization that works in a similar manner as of internet banking but in a more constricted area. As part of its job functions it facilitates the consumers in online transactions of payments. Its customer could be a buyer, seller or payer, who has to make any transaction. On average, around millions of transactions are supported per day through its business platform.

The service model for OpusCapita at a broader level is shown in Figure 2.1:

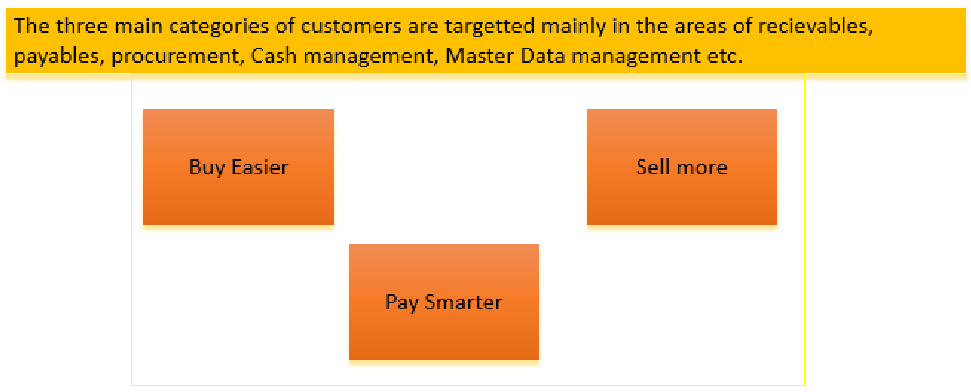
The three main categories of customers are targetted mainly in the areas of recievables, payables, procurement, Cash management, Master Data management etc.
Buy Easier
Sell more
Pay Smarter

*Figure 2.1 Categories of Customers*

The main types of transactions are explained below:

- Accounts Receivables:

This is the transaction types carried by the customer in order to pay for the services the customers have used. This type of money or the invoices are bounded by some due date that depends upon the service offerings.

- Account Payables:

These types of transactions are carried out to pay the invoices from the Companies that are offering services to its suppliers etc.

- Procurement:

This domain concerns with the supplier management and corresponding risk analysis. The workflow is usually complex enough with the suppliers ranging from selections to assessing risks and thus it has its own dedicated domain. OpusCapita provides eProcurement solution for managing all the transactions of this category.

## 2.2 Machine learning:

In this section, the background of machine learning and particularly working of neural networks has been reviewed. Recently, the use of artificial neural networks (ANN) has become very popular in various fields. Deep learning approach is progressing at a very fast pace(3).The research in the field of neural network has shown its strength in terms of pattern classification and pattern recognition (4). In various domains, artificial neural network has proved their absolute necessity by competing human performance. One of the many reasons in the increased popularity of neural networks is massive parallelism, their ability to learn and then generalize in a distributed manner are some of its major advantages (5). Unlike traditional model-based methods, ANN's are self-learning, data-oriented networks. They learn from sample data sets in order to capture the functional dependency of the predictors and target classes(4).ANNs are highly flexible in modelling the patterns in data and are highly adaptable, as they can control the learning behaviour(6).

### 2.2.1 Working of a Biological Neural Network:

Artificial neural network works on the principles of biological neurons in our brains. Artificial neural networks are technical intelligence systems which resemble the inductive power and behaviour of human brain (1). Our brain has billions of neurons that are densely packed and interconnected to each other. One such neuron is shown in Figure 2.2. A typical biological neuron has a "cell body" that receives inputs (information) from "Dendrites" which brought incoming signals and sends the information out through "Axon", a single outgoing channel(7).

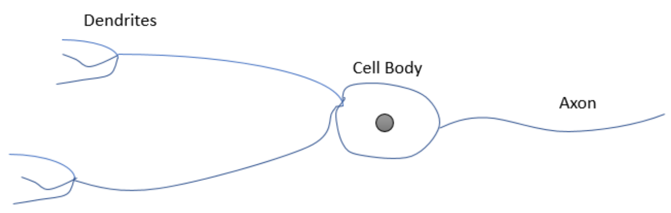

*Figure 2.2 Biological Neural Network*

More specifically, neural networks based on similar functionality are called artificial neural networks to create a distinction from biological neural networks in our brains (7). From this point onwards, we will refer to the neural network as Artificial Neural network (ANN). ANN resembles the biological neural network in the way that learning of the network is carried out from the environment and interneuron connection stores the learned knowledge(8).

### 2.2.2 Working of Artificial Neural Network (ANN):

The artificial neural network in Figure 2.3 is also sometimes termed as feed forward neural network if there are no directed cycles in the network. Analogous to biological neuron, the artificial neural network has several components. The behaviour of the neural network can be tuned by carefully setting the values for these components.

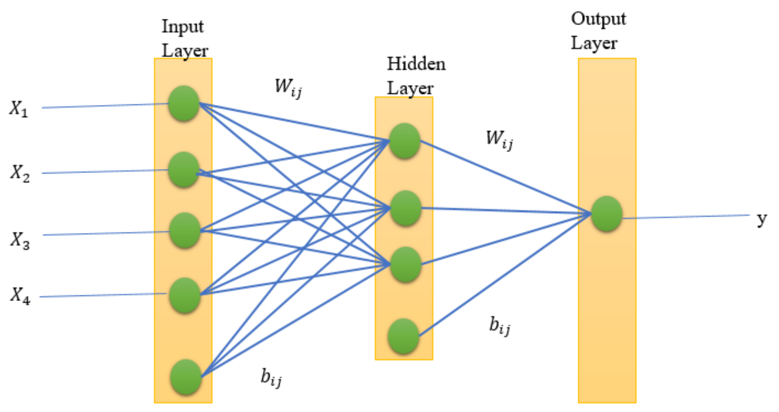

*Figure 2.3Artificial Neural Network*

### 2.2.3 Structure of Artificial Neural Network:

In the Figure 2.3, the structure of a typical neural network (multi-layer) can be seen. A neuron is represented in green colour. The structure of a neural network and its key components are discussed below:

#### 2.2.3.1 Layers:

A neural network is divided into a set of layers. Each layer consists of a set of neurons and is connected to its adjacent layers. The simplest neural network is perceptron that consists of only one layer known as the output layer. As, usually when the number of layers are counted in a neural network, the input layer is not counted. Perceptron has an input layer as well but no hidden layer. The input layer is directly connected to the output layer.

In the Figure2.3, a Multi-layer (ANN), that consists of 2 layers could be observed– Hidden layer and Output layer. The input layer is where the input data for training or testing the algorithm is being sent to the neural network. Depending upon the settings, this layer will then send its computed output to the next layer which is the hidden layer. The hidden layer will then do the computations and send its computed output to the next layer which is the output layer. Depending on the settings of the output layer, it will send its feedback on possible adjustments for the hidden layers. It must be noted that A network too small or too large can lead to poor performance. The Algorithms size is usually set using different approaches like “Constructive algorithms” or “Predictive algorithms” where the algorithm either starts from too small or too large network and then either constructed to a larger network or pruned to a smaller one(9).

#### 2.2.3.2 Neuron:

Neuron is the fundamental part of the neural network often referred to as “Unit”. In a fully connected neural network, each neuron in a layer is connected to all the neurons in the next layer. A neuron gets weighted sum of inputs from the neurons in the previous layers, if it is not the input layer. It then passes that input through an activation function. If the output of the activation function is more than the already set threshold, the neuron will fire, means it will send its calculated output to the next layer. This firing of neurons means that it can now participate in the learning process and can send its computed output to the next layer. In Figure 2.3, it could be seen that 1st layer has 5 neurons as the number of inputs. The next layer to the input layer which is called Hidden layer has 3 neurons that are not input dependent but user choice.

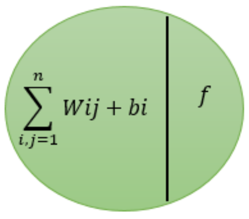

*Figure 2.4 Neuron*

### 2.2.3.3 Activation Functions

Activation functions are of great importance and acts as a fundamental part of a neuron. They aid in creating non-linearity in the neural networks. The most influential unit in the structure of neural network is the net inputs function commonly referred to as "the activation function or threshold function or transfer function", which outputs a resultant value called as "unit's activation"(10). As we explained earlier, each neuron gets a weighted sum of inputs from the neurons in the previous layer.

Depending upon the number of layers this weighted sum of inputs could grow more and more as it progresses through the layers in the network. Imagine if the network consists of large number of layers, this weighted sum would explode the network. To avoid this exploding of neural network by the computed weighted sum of inputs, activation functions are used. Depending upon the nature of the activation function you are using, the output of the neuron could be binary [0,1], in range of [-1,1] etc.

There have been different types of activation function in use. One can create his own custom defined activation functions. New strategies in defining the activation functions have been proposed in literature. For example, for a convolutional neural network, a non-static adaptive activation function has been proposed that learns the function parameters during the training time and based on those learned parameters, generates several types of activation functions at the time of testing(11). In a traditional manner, fixed activations functions are used for each neuron. Below is the list of most commonly used activation functions for non- linear cases:

1. **Rectified Linear Unit:**

This activation function is shown in Figure 2.5. It does not only introduce non-linearity in the architecture of the neural network but also has very simple logic behind its functioning. It returns the same output as the input, if the input is positive; otherwise it would return the value as zero. This type of simplicity creates an effect of scarcity in the neural network, thus making it popular in the deep learning community. But the activation function has a slight issue as well. For the neurons whose output goes into the state 0 are turned off and are no longer respondents to the gradient updates, thus playing no role in the learning process. The issue is often referred as 'Dying relu problem'.

$$RELU(x) = \begin{cases} 0 \; if \; x < 0 \\ x \; if \; x >= 0 \end{cases} \quad (1)$$

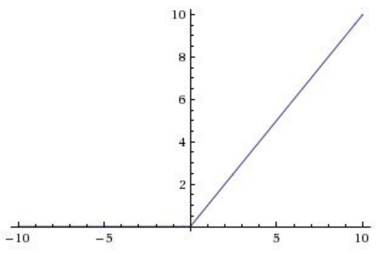

*Figure 2.5 Rectified Linear Unit. Source(12)*

2. **Hyperbolic tangent function (Tanh(x)):**
   This function is a ratio between the hyperbolic sine and the cosine functions and is shown in Figure 2.6. It is a sigmoid function that produces ‘s’ shaped curve. The output is bounded between [-1,1]. This activation function is usually preferred in the neural network due to back propagation approach which requires gradient calculations. The activation function has a little issue often termed as “Vanishing gradients”. It has a sharp slope in the centre which attracts as good distinction on predictions, but this slope almost vanishes near the edges. Thus, for any input change at those points, very little difference would occur in the output and the gradient would be small as well. Small changes in gradient would mean that the network would learn very slowly for those values.

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} \qquad (2)$$

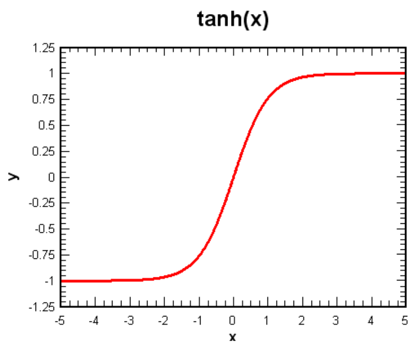

*Figure 2.6  Hyperbolic Tangent Function. Source (13)*

3. **Sigmoid:**

The function has similar properties as of hyperbolic tangent function. It also gives a steep curve in the centre, but this steepness also vanishes (not so vivid) near the edges thus effecting the properties of the gradients. Unlike the Tanh, this activation function gives output in the range (0,1) as can be seen in Figure 2.7. It is also preferred due to the back-propagation trick in the neural network. The function is a preferred choice in case of binary classification case. But it has a slight issue of squeezing the gradients.

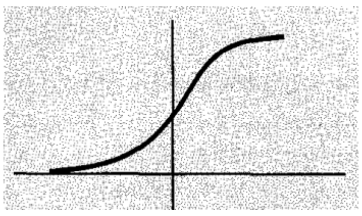

*Figure 2.7 Sigmoid Activation Function. Source (14)*

4. **Softmax:**

The function is mostly preferred in the case of multiclass classification. It gives as an output, an array of vectors (probabilities) that always sums up to 1. This function is often used to get probabilities as well. It gives categorical probability distribution, which tells the probability of any of the possible classes to be true. For calculating the probability of ith class for a data sample in a dataset having j classes, we have equation (3) mentioned below:

$$P(y = j) = \frac{e^{z_j}}{\sum_{k=1}^{C} e^{z_k}} \qquad (3)$$

where $z_j$ is the weighted sum arriving into a particular neuron of the output layer and the denominator normalizes the output layer neurons to sum to one.

### 2.2.3.4 Cost Function:

The problem of learning from inputs to generate corresponding outputs can be dealt as a process of minimizing a suitable error function(15). This error function is referred to as objective function as well, in that case the goal is to maximize the specified objective function. Optimization problems use objective functions to select the solutions that yield high values of objective function(16).The objective function is also sometimes called as cost function or error function that the network tries to reduce. There is a variety of objective functions which are being used according to the field of application of the Neural network as well as the classification cases. Most commonly used error function is the Sum of Squared error (SSE) or Mean Squared Error (MSE)(17)(18)(19). In keras, for machine learning problems that can be categorized under Multiclass classification domain, the recommended cost function is categorical cross entropy as shown in equation (4):

$$-\sum_x p(x) \log(q(x)) \qquad (4)$$

### 2.2.3.5 Gradient Descent:

Gradient descent is a computational optimization algorithm in which the error function (which is dependent on the parameters which are weights of the network)is minimized by taking the derivative of the error function and updating the parameters of the neural network in the opposite direction of the derivative (the way these derivatives are computed is called back propagation). A trade-off is made between the accuracy as well the time for parameter updating depending upon how much data you use. According to (20) , commonly used gradient descent algorithms are:

1. Batch gradient descent:
   In this strategy the gradient of the error function is computed with respect to the parameters for an error function over the entire training set in each update.
2. Stochastic gradient descent:

It performs the gradient update for each sample of training data thus it's faster in updating the gradients, than the batch gradient descent.

3. Mini batch gradient descent:
   This algorithm has been designed to take a mid-way between batch gradient descent and stochastic gradient descent. Mini batch represents the size of training set when divided into several parts.

#### 2.2.3.6 Optimizer:

There are different types of optimizers that can be used in the neural network as helpers for the gradient descent. These optimization algorithms determine how the parameters are updated based on the computed gradient. Commonly use optimizers are explained below:

1. **Momentum:**
   It basically tries to aid the objective function in getting trapped in the local minima (the minimum of the error function). Momentum works very effectively for stochastic gradient descent where the steepness is larger in one dimension than in the other for example near local minima(21).It guides SGD in the relevant direction and smooth out the oscillations(22) .

2. **Adam:**
   Instead of computing only the first derivative of the objective function, Adam calculates the second derivative as well. In other words, Adam uses the second moments of gradient in attempting to optimize the objective function(23). The authors in (23)empirically proved the convergence goal, that was defined theoretically and demonstrated how well "Adam" performs.

### 2.2.4 Types of Neural Network:

As the research in the field of neural network is advancing, the available types of neural network are also increasing. The types of network from the earliest single-layer perceptron have evolved significantly into various types depending on their use cases. Some most commonly used neural networks are explained below:

#### 2.2.4.1 Single-layer Perceptron:

It is a very simple and basic type of neuron consisting of only two layers: Input layer, Output layer. The neuron in the output layer takes a weighted combination of the input, applies some suitable activation function and passes it as a result. A typical perceptron can be seen in Figure 2.8.

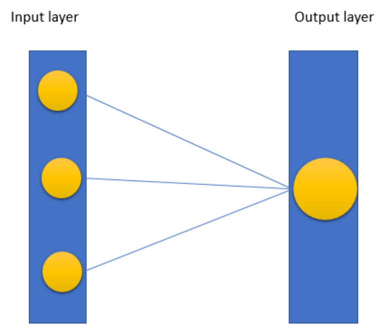

*Figure 2.8Perceptron*

### 2.2.4.2 Feed Forward Neural Network:

These networks are also very popular and have been in use in a lot of areas. In a feed forward neural network all nodes are fully connected. There is at least one hidden layer between the input layer and the output layer. As the number of hidden layers increases from 1, the network becomes deep and is often referred to as deep neural network. There is no loop in the network. Back propagation is the common way of training this neural network. A typical feed forward neural network can be seen in Figure 2.9.

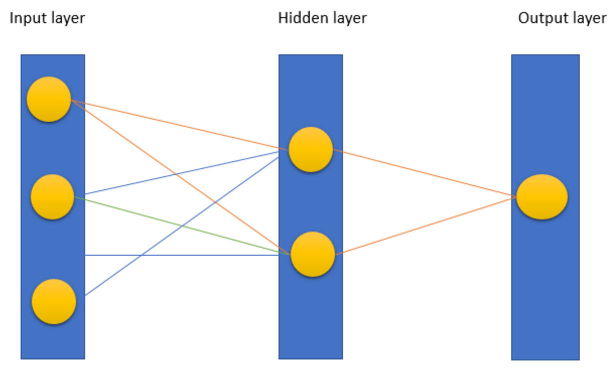

*Figure 2.9 Feed Forward Neural Network*

### 2.2.4.3 Recurrent Neural Network:

These networks are similar to the feed forward neural network with a major difference. This difference is the presence of loop in its layers. In a feed forward neural network, we assumed that all the outputs of each neuron in a particular layer can be computed independently of each other. Whereas, in some cases at some point in time the prediction of output might be dependent and could be better predicted if the previous output is also known. This kind of system could be seen in the way natural language processing occurs. This type of dependency induces a memory in these networks so that the network is able to remember about its calculations. A typical recurrent neural network can be seen in Figure 2.10.

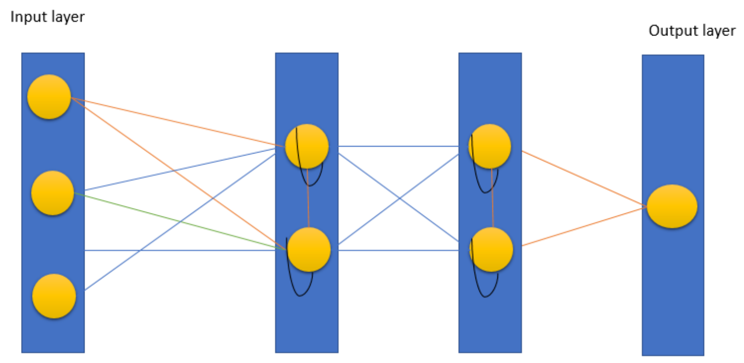

*Figure 2.10Recurrent Neural Network*

### 2.2.4.4 Autoencoders:

These types of neural networks have two parts: Encoding part, Decoding part. So, in terms of layers we can say that they have an input layer, encoding layer and decoding layer. These networks are used in unsupervised learning. They try to generate the copy of its input while learning the properties of the data. Autoencoders resemble Principal Component Analysis but it provides more flexibility and can learn both linear and non-linear representations of the data. It can also be used to search for common representations across data. There are further several types of Autoencoders depending on the usage requirements. A typical autoencoder is shown in Figure 2.11.

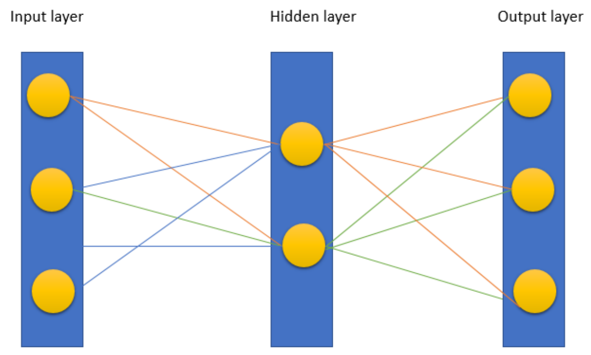

*Figure 2.11Autoencoder*

### 2.2.4.5 Extreme Learning Machine:

These neural networks are a variation of feed forward neural network and can be seen in Figure 2.12. It works faster than the feed forward neural networks. It does not use backpropagation or gradient descent for tuning its parameters. They key aspect behind its working is that the number of neurons in the hidden layer are generated randomly. This is also one of the downsides as the size of the hidden layer could randomly add up poorly chosen for the learning task.

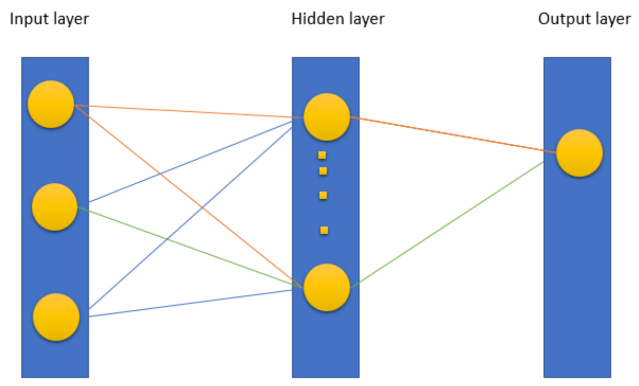

*Figure 2.12Extreme Learning Machine*

### 2.2.4.6 Radial Basis Neural Networks:

These neural networks are also a slight variation of feed forward neural networks. There neural networks use radial basis as their activation in the neurons. They perform relatively better on classification tasks. The output is determined from weighted sum of RBF neurons and number of output nodes is one per class to be classified. Ref Figure 2.13.

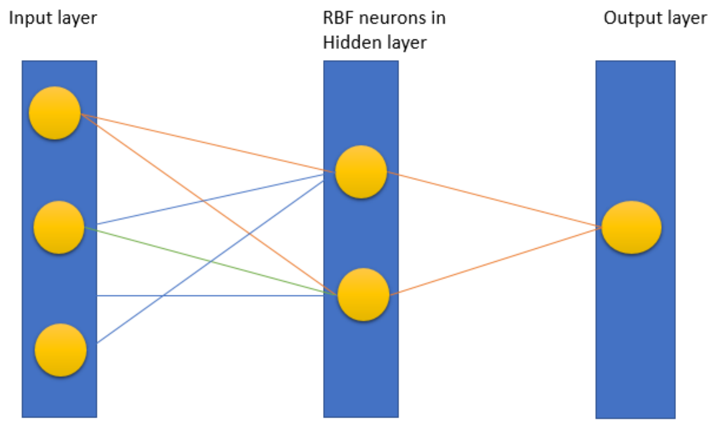

*Figure 2.13 Radial basis neural network*

Based on the availability of time and resources, only a feed forward neural network was attempted in this work.

# 3 Literature Review:

This chapter represents the literature review on the application of feed forward neural networks, multi-task learning and continual learning in finance particularly in terms of Business process automation.

## 3.1 Feed Forward Neural Networks in Financial Process Automation:

So far, neural networks or in general machine learning has been employed in the field of finance in very limited domains. The domains like fraud detection, mortgage, stock prediction, etc. that have been targeted are vital points for the financial business and so part of the research has been focussed on trying to reduce the risk factors of financial institutions rather than automating business processes.

Those areas of finance, that can give leverage to a company in terms of efficient business processes and particularly easy Invoice-to-pay processing hasn't been targeted much yet. However, under the category of Order-to-Cash (O2C) which also covers the domain of invoice automation, it has been tried to predict the customers that can potentially "default" and thus need to be given customized treatment by analysing the invoices data. For example,(24)used the invoices data as a supervised classification learning problem to predict those customers that would pay late and needed suitable actions in advance, to improve the Account Receivables but this O2C work did not employ neural networks.

(25)has worked on classifying a relatively smaller set that consists of 560 digitally scanned invoices into 68 different classes based on imaging by focussing mainly on the features to extract information from the images of invoices. (26) used a convolution neural network to address the problem of recognizing invoices and determining if the invoice is printed by machine, is it written by hand or it's just a normal receipt. This paper also focussed on feature extraction through complex deep neural network for images, however for classification they have used the commonly recommended machine learning algorithms. Surprisingly KNN (K Nearest Neighbours) surpassed the results in terms of classification. However, none has tried to automate the classification of invoices into a certain category of an attribute that is a part of invoices data.

If we look into neural networks applications in finance further, our supposition mentioned already regarding usage of neural networks in finance, seems to strengthen. During the 90's, the use of the feed forward neural network in finance increased steadily in different fields mainly, prediction of bankruptcy, forecasting stock market, credit analysis and business cycle recognition(27). Moreover, (28) also suggested, that majority of the applications of neural network in finance have targeted "default" prediction in financial decisions particularly. The literature searched varied in the domains like predicting stock, bankruptcy, fraud detection, construction contract, credit, etc. with not even a single paper trying to automate a tedious manual financial process.

Among the cited list of publications, the only article that seems to resemble our research approach was (29) that used feed forward neural network for prediction of interest rate, to support efficient investment strategy. But it differs significantly from our work in terms of the data used, methodology employed which in their case is only a feed forward neural network with a 33-3-2-1 topology and above all, the target class predicted which in our case are the categorical attributes for invoices and in their case, and  interest rate.

Some researchers used neural networks in addition to other popular machine learning algorithms, but the domain of work remained the same as explained above. For example,(30)built scientific model to combat fraud using several strategies including neural networks on transaction's data and found out that neural network performing better than the decision trees and logistic regression when compared on the basis of lift chart for prediction of "default" transactions.

We came to this conclusion that not only has the aim of the research, revolved around the subjects of predicting fraud, stock market, credit, etc. but also the topology employed for the neural networks, has remained pretty much the same i.e. majority have used only 1 hidden layer with only a few using 2 hidden layers. While few have tried to automate some financial business process as well.

Also, the use of the feed forward neural network was pretty common with some exceptions like some papers have made the use of recurrent neural networks for the targeted financial problems (31).

## 3.2 Multi-Task learning in Financial Process Automation:

Multi-task learning has been used widely in different domains giving substantially better results.(32)has used in image detection for joining the sparse sub-hyperspectral models to detect a target in high spectral images.

Yet, similar to our findings regarding feed forward neural network, it appears that the use of Multi-task learning is also limited to afore mentioned domains but comparatively there is less material available for review in terms of its financial applications. For example,(33)took Canadian stock data of upto 8 years, monthly treasury bills, applied partially soft parameter sharing for predicting future returns of stocks and concluded that partial Multi-Task learning in the form of parsimonious model performed better for the given set of targets.

(34) also applied MTL on financial forecasting but the aim was to propose the regularization in the typical neural network architecture. It is worth mentioning that both of these papers focussed on time series aspect as well which is out of the scope for this project.

However, any researches, that could suggest the use of multi-task learning to automate a financial process could not be found let alone invoice automation.

## 3.3 Continual Learning in Financial Process Automation:

Another variation of the feed forward neural network that we used is Continual Learning. Continual learning or Lifelong learning (LL) has garnered much attention in recent years. The technique mainly applies the use of models that have already learned tasks, to perform better after enhancing their learning from the tasks learned later in a continual manner (35).

However, the technique usually suffers from Catastrophic forgetting in which the learning of the new tasks interferes with the knowledge from already learned tasks (36) . (36) has also discussed in detail the ideas for rectifying the problem of catastrophic forgetting and concluded that while learning the model could validate itself repeatedly by keeping snapshots of parameters and a snapshot of original dataset. Though the technique is becoming popular, but it hasn't been tested much yet for classification of invoices.

(8) has worked on classifying the documents that consisted of Invoices using a dynamic approach towards incremental learning. It also incorporates the idea of dynamically creating a new neuron depending upon receiving data which might have a class that didn't exist in the previously learned data set. Although the approach followed can be categorized to the continual learning we performed, however the concept of dynamically evolving network of neurons base on the current data set, distinguishes their approach from ours.

We have discussed this technique further in the technical implementation section.

# 4 Technical Implementation:

## 4.1 Setup:

To begin with, we used sklearn Multi-Layer Perceptron library for our work. Due to less flexibility in the functionality of sklearn Multi-layer perceptron library and more ease of use in the open source library Keras, we shifted our work to Keras. In the table 4.1 below, are the details of the tools and technologies that we have used:

| Frameworks/ Tools | Keras on TensorFlow |
|---|---|
| Evaluation strategy | K - Fold Cross Validation where k=4 |
| Language | Python |
| Visualizations for results interpretations | TensorBoard, Matplotlib |

*Table 4.1 Tools and setups*

### 4.1.1 Keras:

Keras is an open source neural network library that is available in python and R. It acts as an application programming interface on the comparatively complex neural network libraries like TensorFlow, Theano, CNTK. Keras further provides variety of functional API and sequential models to implement a variety of Neural networks.

### 4.1.2 TensorFlow:

TensorFlow is an open source machine learning framework that provides variety of numerical computations with ease. It is used to make different machine learning models at different levels of abstractions.

### 4.1.3 Python:

Python is a scripting language and is very popular for its use in machine learning models. It supports different programming paradigms like functional programming, object-oriented programming. Machine learning and specifically neural networks supporting frameworks allows using python. Several versions of python have been released. The version used in this empirical work is: 3.6.4.

### 4.1.4 Matplotlib:

Matplotlib is a plotting library in python. It provides various graphics and visualizations. This library has been used to visualize the performance of the machine learning algorithms under investigation.

### 4.1.5 TensorBoard:

TensorBoard is a library being developed to visualize the algorithm performance and its learning. The visualization provides interactive insights into the algorithm performance on training data, validation data and test data. Also, for very deep neural networks one can visualize the network and parameters flow.

## 4.2 Methodology

In all the machine learning examples, the data pre-processing is an important and critical step in the cycle of machine learning, training and predictions. Amongst the commonly known practices, the most important is handling missing data. In the data files used in the study, missing data was marked by several notations:

["","NA","N/A","empty","NaN","Empty","nan","NAN"].

So, a common representation to represent missing data was first needed. Therefore, all the values in the afore mentioned list as were marked as “unknown” in all over the dataset. The strategy employed for the missing values imputation is “Complete Case Analysis” where the samples having missing data in the target class are deleted. Some of the softwares have this functionality to remove rows having missing data either in predictors or outcome which is called complete case analysis (37) .

- The nominal attribute in the dataset was binned manually on the basis of the expert’s knowledge into 16 bins.
- For Target Variable1, the max difference in the count for Customer1 was found around 300 for the top most occurred value and the top fifth occurred value. For Customer2, this difference was around 17,786, if the first highest occurred value count is compared with the fifth highest occurred value. Similar scenario was observed for Target Variable2 as well, in both datasets. The skewedness for data available as Target Variable1 is shown in Figure 4.1.

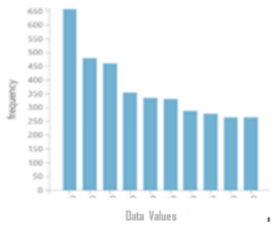

*Figure 4.1 Skewedness in Target Variable1*

- Based on data distribution and expert judgement, a sample from the available data was randomly selected so that previously top 5 most common values of the target variables will now have same number of occurrence count. The same strategy was followed on Azure ML Studio so that the results can be compared. As the nature of both target variables was categorical even though they were represented using a combination of numbers, taking mean or average of these columns would not make any sense. However, using open source python library, min-max normalization was done before feeding the data to the actual model. This type of normalization (which in our case was linear transformation) is imperative in order to get unbiased results from the model(38).

The analysis presented in the Table 4.2 below, was performed in accordance with the purpose of the algorithm which was to make a common predictive algorithm that could work on both datasets.

| ***Target Classes Properties*** | ***Customer1*** | ***Customer2*** |
|---|---|---|
| **Total Rows in Dataset** | 8890 | 67363 |
| **Unknown in Target Variable1 and Target Variable2** | 0 | 0 |
| **Top 5 occurred values row count (Target variable1)** | 658, 480, 460, 354, 334, 331, 287 | 19922,17150,3520, 2676,2136,1947,1931 |
| **After handling data Skewedness, row count of the top occurred values is:** | 334,334,334, 334,334,331,287 | 2136,2136,2136, 2136,2136,1947,1931 |
| **Total Rows after handling Data Skewedness** | 8274 | 32639 |
| **One Hot Encoding, data (rows, columns)** | (8274,2664) | (32639, 17568) |
| **Unique values in Target Variable1** | 128 | 176 |
| **Unique values in Target Variable2** | 25 | 11 |

*Table 4.2 Data preprocessing on Customer1 and Customer2*

### 4.2.1 K - fold Cross Validation:

For a large amount of data, k-fold cross validation can be used for performance evaluation of classification algorithms, because the training data is generally insufficient to evaluate algorithm performance(39). The dataset was divided into 4 folds. At a time, only first 3 folds are given as training set and the remaining 1-fold is kept as test set. This process is repeated four times. Since the divided test set in this data, was not unseen data but a portion of available data so we named it as Dev Test Set. So, we will refer the commonly used term “Test Set” as “Dev Test Set”. At the time of dividing the data into training and dev test sets, business logic was used as a constraint on splitting. According to the business logic, two or more invoices belonging to one customer should not exist both in training and test set. The primary key that is used to identify the invoice-customer relationship was checked at the time of splitting and it was made sure that a particular customer’s invoice can only be found either in Training set or DevTest Set but not both.

### 4.2.2 Validation Set

The training set is further divided into Training and Validation set. Python scikit-learn provides a library “train_test_split”, that returns the splits of the target classes proportional to a desired split proportion of the overall data set. The resulted splits were then checked to verify that the row count after splitting in each of the fold is somewhat similar. The statistics of the rows after splitting and the target class distributions are shown in the Table 4.3& Table 4.4 below:

| **Customer1** | **Fold1** | **Fold 2** | **Fold 3** | **Fold 4** |
|---|---|---|---|---|
| **Rows in Training Set** | 6384 | 6422 | 6919 | 6719 |
| **Rows in DevTest** | 1890 | 1852 | 1355 | 1555 |
| **Rows in Validation set** | 1916 | 1927 | 2076 | 2016 |
| **Remaining rows in Training set** | 4468 | 4495 | 4843 | 4703 |
| **Columns after One Hot Encoding** | 2664 | 2664 | 2664 | 2664 |
| **Total Rows** | 8274 | 8274 | 8274 | 8274 |

*Table 4.3 Customer1 - Row counts in Training set, validation set & Dev Test set*

| **Customer2** | **Fold1** | **Fold 2** | **Fold 3** | **Fold 4** |
|---|---|---|---|---|
| **Rows in Training Set** | 26,089 | 26259 | 26087 | 25930 |
| **Rows in DevTest** | 6550 | 6380 | 6552 | 6709 |
| **Rows in Validation set** | 7827 | 7878 | 7827 | 7779 |
| **Remaining rows in Training set** | 18262 | 18381 | 18260 | 18151 |
| **Columns after One Hot Encoding** | 17568 | 17568 | 17568 | 17568 |
| **Total Rows** | 32639 | 32639 | 32639 | 32639 |

*Table 4.4 Customer2 - Row counts in Training set, validation set & Dev Test set*

Before splitting, all the data was label encoded and One-hot-encoded using libraries from sklearn package.

### 4.2.3 Classification Platform

Keras provides different API's to create custom designed models for feed forward neural networks. In the current study Keras sequential model is used to build our feed forward neural network for traditional classification platform. This approach will be referred as Classical Model under investigation in the later parts of the work, for referencing while comparing it with the other machine learning approaches. In the graphs that designed for visualizing the algorithm performance, this approach will be represented as CLF. Besides CLF, the other approaches that we have tested are Multi-Task Learning and Continual Learning.

### 4.2.4 Steps involved in defining Artificial Neural Network

In order to optimize the use of the computational resources as well as minimizing the processing time, the features of the neural network for implementing a predictive algorithm were determined as below:

a. Hyperparameters are those parameters of the neural network that cannot be directly determined from the samples/cases in the given data but requires careful tuning and estimation for good performance of the predictive algorithm. In this case, a domain of hyper parameters is first defined and tested the performance of algorithm over those values. This allowed to restrict the possible values of hyper parameters for grid search to a set, so that the algorithm takes less execution time.
b. The domain of hyperparameters including the values that are being used in Azure and drew plots about the behaviour of the predictive algorithm on those domain values was then defined.
c. By drawing conclusion and careful heuristics from those graphs, the selected hyperparameters were being able to confine the domain that required to be tested.
d. The predictive algorithm was then trained using Grid search, on the selected set of Hyperparameters over the tested domain of values.

### 4.2.5 Hyper Parameters Space Exploration

As mentioned earlier, a suitable range of values was defined as the domain of the hyper parameters. The algorithm performance was then tested on those parameters and the domain values were confined further. This allowed us to spend less time in execution by avoiding the algorithm run on unnecessary parameters. Some of the most important hyper-parameters whose values can influence the algorithm directly, are explained in this chapter.

#### 4.2.5.1 Learning rate:

For learning rate, we tested the domain values [0.1, 0.001] and kept the values of the rest of the hyper parameters fixed.

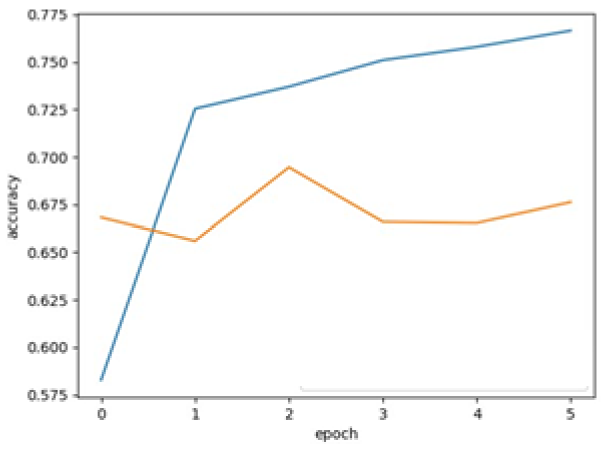

*Figure 4.2 Accuracy of the classical model on each epoch for Learning rate=0.1, BatchSize=50, Epochs=100, Prediction accuracy on training set= 76% (in blue color). Prediction accuracy on Training_Test set= 67% (in orange color). It stopped after 5 epochs*

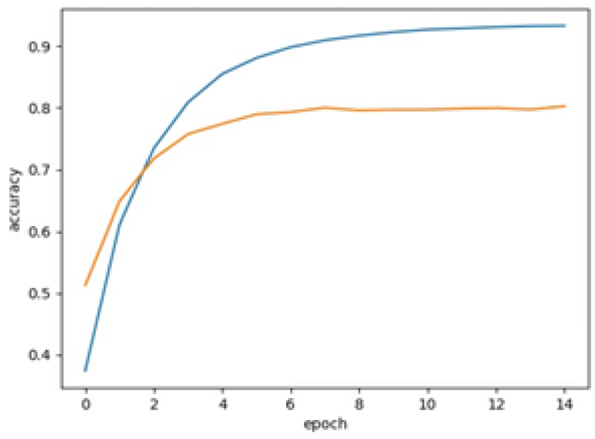

*Figure 4.3 Accuracy of the classical model on each epoch for Learning rate 0.001, BatchSize=50, Epochs=100, Prediction accuracy on training set=93% (in blue color). Prediction accuracy on Training_Test set= 80% (in orange color). It stopped after 14 epochs*

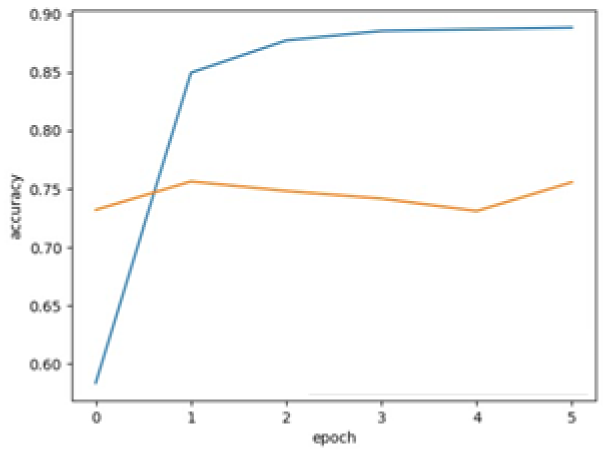

*Figure 4.4 Accuracy of the classical model on each epoch for Learning rate=0.1, BatchSize=300, Epochs=100, Prediction accuracy on training set= 88% (in blue colour). Prediction accuracy on Training_Test set= 75% (in orange colour). It stopped after 5 epochs.*

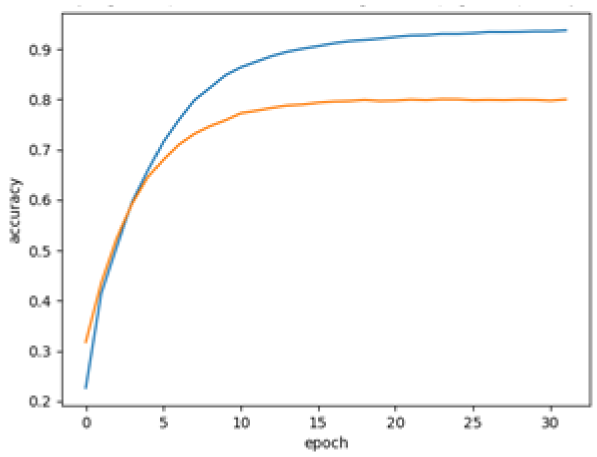

*Figure 4.5 Accuracy of the classical model on each epoch for Learning rate=0.001, BatchSize=300, Epochs=100, Activation='tanh', optimizer=Adam, DropOutRate=0.0. Prediction accuracy on training set= 93% (in blue colour). Prediction accuracy on Training_Test set= 79*

From the above displayed figure 4.2, figure 4.3, figure 4.4 and figure 4.5, we can see that, even though it takes more epochs, still the algorithm smoothly achieved high performance when the learning rate was set 0.001. When the learning rate =0.1, the classical model is performing well on the training data but not on the Dev Test Set. It takes abrupt shifts in its movement, regardless of the fact, that the number of epochs- the algorithm chose to run for, were less. From these figures, it was concluded that the learning rate could be fixed as 0.001.

#### 4.2.5.2 Batch size:

For Batch size, we tested the domain of values as: [50, 100, 200,300].

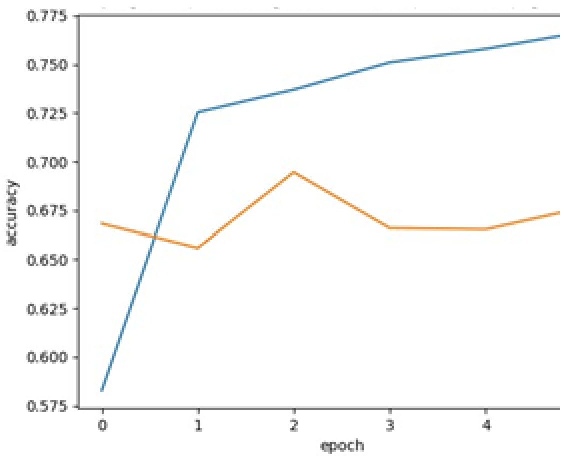

*Figure 4.6 Accuracy of the classical model on each epoch for Learning rate=0.1, BatchSize=50, Epochs=100, Activation='tanh', optimizer=Adam, DropOutRate=0.0. Max achieved accuracy on training set=76% (blue line) and on Train_Test set=67% (Orange line)*

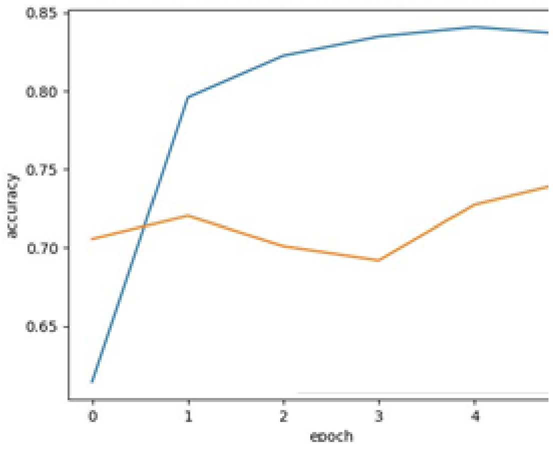

*Figure 4.7 Accuracy of the classical model on each epoch for Learning rate=0.1, BatchSize=100, Epochs=100, Activation='tanh', optimizer=Adam, DropOutRate=0.0. Max achieved accuracy on training set=83% (blue line) and on Train_Test set=74% (orange line).*

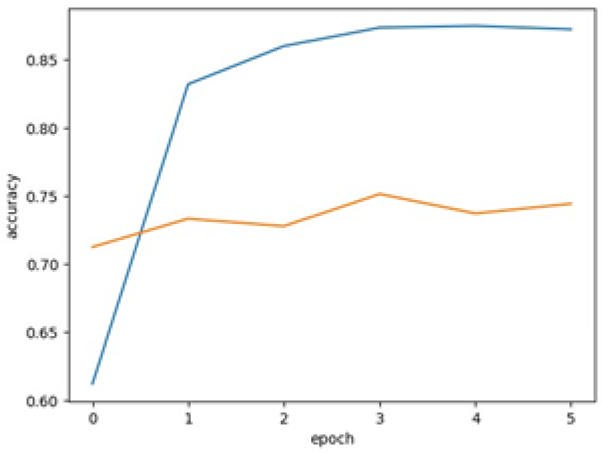

*Figure 4.8 Accuracy of the classical model on each epoch for Learning rate=0.1, BatchSize=200, Epochs=100, Activation='tanh', optimizer=Adam, DropOutRate=0.0. Max achieved accuracy on training set=87% (blue line) and on Train_Test set=74% (orange line).*

From the Figure 4.6, Figure 4.7 and Figure 4.8 it can be visible that the bigger batch size was performing better, but using all the batch sizes in the domain was preferred throughout the work, as there is no single rule for selecting a batch size.

### 4.2.5.3 Optimizer:

The performance for RMSProp, Adam and stochastic gradient descent was then tested.

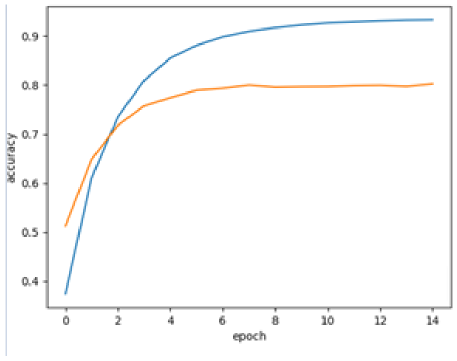

*Figure 4.9 Accuracy of the classical model on each epoch for Learning rate=0.1, BatchSize=50, Epochs=100, Activation='tanh', optimizer=Adam, DropOutRate=0.0. Max achieved accuracy on training set=93% (blue line) and on Train_Test set=80% (Orange line)*

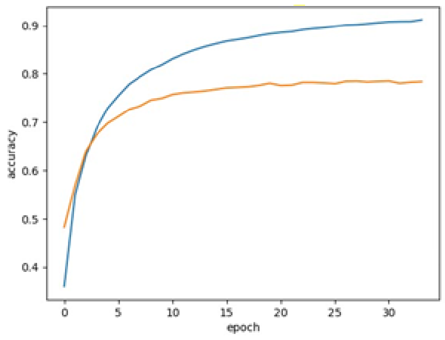

*Figure 4.10 Accuracy of the classical model on each epoch for Learning rate=0.1, BatchSize=50, Epochs=100, Activation='tanh', optimizer=Adam, DropOutRate=0.0. Max achieved accuracy on training set=93% (blue line) and on Train_Test set=80% (Orange line)*

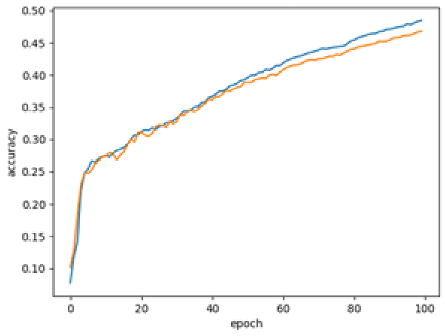

*Figure 4.11Accuracy of the classical model on each epoch for Learning rate=0.1, BatchSize=50, Epochs=100, Activation='tanh', optimizer=SGD, DropOutRate=0.0. Max achieved accuracy on training set=48% (blue line) and on Train_Test set=46% (Orange line)*

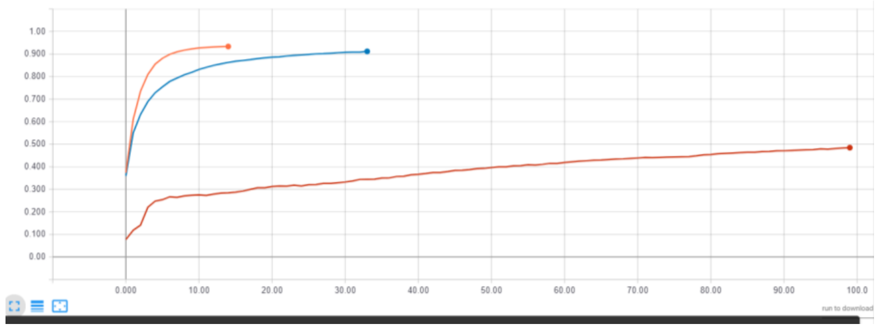

*Figure 4.12 Classical model accuracy for each epoch for three optimizers. The line at the top in orange color indicates "Adam", in Blue color is "RMS Prop" and in Red color is "SGD". 'Adam' is performing clearly better. 'SGD' need more iterations*

From the figures above, it can be seen that the performance of the optimizer "Adam" was better than the rest of the optimizers. So, the value for the optimizer was fixed as "Adam". With, stochastic gradient descent, the performance was still improving, and it would need more than 100 epochs to reach its end.

#### 4.2.5.4 Classical Model:

The classification platform that has been built using Keras sequential model consists of 3 layers- “Input layer, Hidden layer, Output layer”. The number of neurons in the input layers is the same as the number of columns in the input feature data set. This input feature dataset that we used for training the algorithm is termed as X_train. Selecting appropriate number of nodes in both the hidden output layer and by careful training of the input and output weights, SLFNs (Single Layer Feed forward neural network) turns out usually a good choice for function approximation, digital signal and image processing, modelling, adaptive control and information retrieval(40). The author discussed SLFN in reference to hyper parameters tuning, however, similar careful consideration is needed for tuning of neural networks of any size.

A suitable number of neurons were tested, varying the value around the one used in Azure ML studio. It was found that the 200 neurons seem to be the best option as after that the accuracy doesn’t vary. Similarly, the numbers of hidden layers were varied. Based on the prediction accuracies on the Dev Test Set, it was decided to keep only 1 hidden layer. For the output layer, the numbers of neurons were kept the same as the number of unique values in the output class. This led us to fix the value of activation function as ‘softmax’ in the output layer.

Since some of the hyperparameters requires careful tuning, it did not seem suitable to fix value for each of them. For example, for Batch size and DropOut rate we preferred grid search. Table 4.5 shows the values of the hyper parameters that we used in our classical model.

| | |
|---|---|
| Batch size | 50,100,200, 300 |
| DROP-OUT RATE | 0.0,0.1, 0.2 |
| ACTIVATION | Tanh |
| Based on our analysis in the process of hyper parameters space exploration, we fixed these values for the respective hyper parameters | |
| OPTIMIZER | Adam |
| LEARNING RATE | 0.001 |

*Table 4.5 Classical Model Hyper - parameters*

The Keras classical model was given freedom to select a best choice from those values of the hyperparameters after the model was trained for their all possible combinations. This strategy has been referred in this paper as “Grid Search”. Any open source library for doing this grid search was not used rather it was implemented manually. The key reason behind this was ‘hidden tuning of neural network’ that the open source library (for grid search) might be doing.

## 4.3 Models:

As discussed in chapter 3, there is so much variety in the types of models, that are available in the domain of neural networks. Based on the data size and the type of prediction required, three main types of neural network applications were chosen and experimented with the data in four ways. In all applied strategies, Multi-layer feed forward neural network has been used.

### 4.3.1 CLF:

The first strategy corresponds to the traditional multi-layer perceptron. From this section further, this classifier is represented as "CLF".

#### 4.3.1.1 General Comparison:

The performance of this model has been cross-validated first in general. This means that the performance on the data generated by the cross-validation split of our classifier "CLF" and the prediction accuracies of Azure ML studio has been compared on the data generated by cross-validation split of Azure.

#### 4.3.1.2 Exact Fold Comparison:

In order to strengthen confidence on the designed model, the performance of "CLF" was compared with the Azure ML Studio on exact same data. The training data and test data of 4-Folds was taken from Azure, and the performance of "CLF" was tested by training and testing it on those datasets.

### 4.3.2 Multi-Task Learning:

The effect of the performance of classical model was compared with another approach called Multi-Task learning. The concept of Multi-Task learning originated by (41) as mentioned in (42) and (43), however as deep networks were not prevalent at that time, this approach targeted shallow networks (44). It is a type of predictive modelling in which the learning of one task is influenced by the learning of other tasks (41). It's an inductive transfer method with the goal of achieving better generalized performance by leveraging shared information in related tasks (45). If the tasks are related, it is better to learn them simultaneously rather than following the typical way of learning one task at a time (46). In most cases, learning multiple related tasks in parallel has shown to achieve better performance (41)(47)(48)(49).

(50) has suggested that multi-task learning exhibits similarity to statistical multi-level learning and by carefully designing the prior distributions, has utilized Bayesian approach as a strategy for Multi-Task learning (MTL). (51) has exploited hidden properties of phoneme sub-categories on vocabulary speech database and then by utilizing several techniques incrementally grew larger networks without any performance degradation. Many other scientists have cited this paper as a well depicted discussion to defy the traditional strategy of "Divide and Conquer" and leading to MTL strategy. (41) has compared the effect of single task learning (STL) versus multi-task learning (MTL) on object recognition and was able to achieve 20-30% better accuracy in the latter case. (49) has used MTL for natural language preprocessing.

It could be seen that many related forecasting indicators can be predicted simultaneously in the domain of finance (52).(53)has applied multitask learning in stock price prediction by utilizing the relationship between companies and compared the performance of this model with the individual stock price prediction model and empirically demonstrated improvements in the trading profit.

The important notion is that the tasks which are going to be learnt in shared model, should somehow be related so that they can serve as a source of inductive learning (41). Keeping this assumption, MTL was used to not only determine the quality of performance but also, to know if there exists any dependency in the target business variables.

Both the prediction classes, in the Invoices of the tested datasets of two customers, were suitable to be used in Multi-task learning, as the feature set that is used for their training is common. The training process of both prediction variables was combined. This enabled to determine if any dependency in the prediction variables that could be used. It also helped to save the training time by carrying out the training of both Target Variables simultaneously.

#### 4.3.2.1 Multi-Task learning – Model architecture:

Multi-Task learning in neural networks architecture is mainly achieved through parameters sharing in the following ways:

1. Hard parameter sharing
2. Soft parameter sharing

##### 4.3.2.1.1 Hard parameter sharing:

The approach is also known as classical approach as was first introduced in (41). In the mechanism, the hidden layers are kept shared while keeping the output layers (which are task specific, thus their number would be same as the number of the tasks) unshared and independent of each other (42). This mechanism is usually preferred due to its several advantages:

1. Firstly, it avoids overfitting (54), as the added output neurons will act as regularizes for hidden representation (44).

2. Secondly, the implementation of this mechanism is easy (43). However, (55) has proposed a new architecture other than the traditional ones mentioned above, with the name as SLUICE network using recurrent neural network. The network could be replaced with Multi-layer perceptron as well and founded on the fact that the network

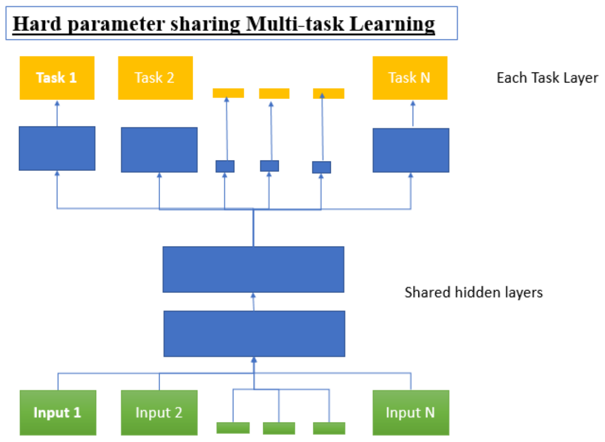

*Figure 4.13Hard parameter sharing*

can learn selective sharing of layers, subspaces. Resultingly, the model even outperformed the performance of hard parameter sharing. Below Figure 4.13 is visual description of how hard parameter sharing works.

#### 4.3.2.1.2 Soft parameter sharing:

Soft parameter sharing is also known as column-based approach (44). Every task is given its own independent model and parameters; however, the parameters are then shared across the models. (42)also suggested the similar approach by allowing the task to share kernel hyperparameter but each task can have different kernels. In Figure 4.14, a visual understanding of how soft parameter sharing works could be achieved.

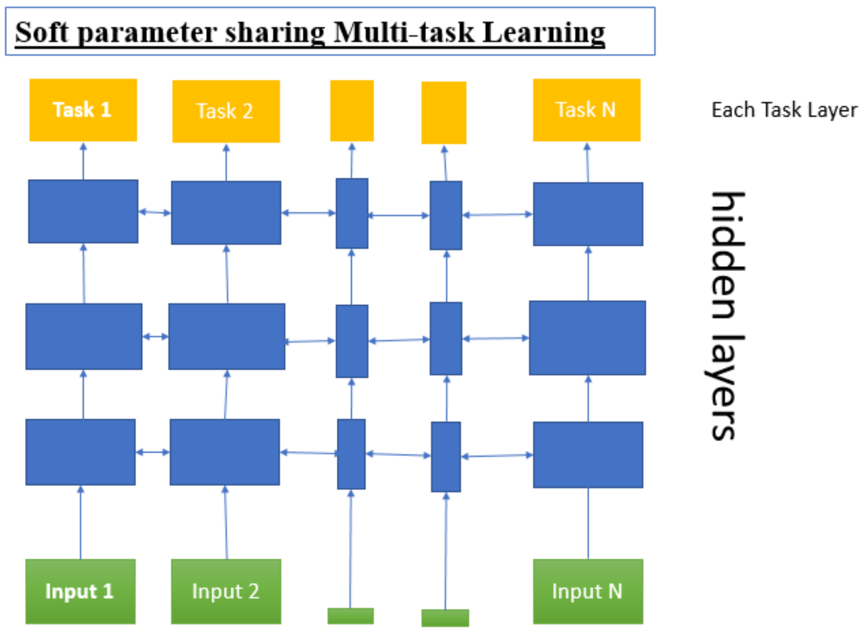

*Figure 4.14 Soft parameter sharing*

Besides, the most popular strategies mentioned above, there are several other methods proposed as variations of the already described strategies. For example, (56) has introduced Multi-task learning in the IVM algorithm where the heuristic, greedily selected the most informative examples from the tasks but it slightly impacts the computational cost. The other variation includes Regularized Multi-task learning where the tasks are assumed to be similar (have come from some particular distribution), however it puts a penalty for each task depending on how much it gets deviated from the mean(46). (57) has empirically shown the achievement of state-of-the-art results by using Multi-task learning in Web search ranking. Boosted decision trees have been used by the author along with l1-regularization to have sparsity. In the current project however, regularization wasn’t used, dropout was introduced in the network instead.

(58) has proposed another variation of Multi-task learning with the name Robust Multi-task learning. In their proposed algorithm, the tasks relationship is captured using a low rank structure and simultaneously identified the outlier tasks using a groups sparse structure which can be seen in equation (8).

$$\min_{L,S} L\left(\left(l_i + s_i\right)^T \left| x_j^i, y_j^i \right.\right) + \alpha \|L\| + \beta \, V|S|V \square_{1,2,} \tag{8}$$

Where L is low rank matrix capturing the task dependency, S is Sparse Matrix capturing the non-dependency (or as the author called – outliers). $l_i$ and $s_i$ forms the weight vector. For ith task and jth training sample, we have x representing the sample and y representing the target class.

**4.3.2.2 Technical Implementation:**

Keras provides functional API that can be used to develop a customized model. This API was used to build a neural network with one hidden layer. Input_layer is the input layer for the neural network with number of columns – 2723. This Input layer is connected to the layer "Dense_4" which is the only hidden layer in this neural network. The hidden layer has the parameters = 544800. In the Table 4.6 below, it can be seen that hidden layer is common and shared by Target Variables which are represented by the Dense_5 and Dense_6 layers respectively (can be seen in Connected to column of the table).

| Layer(type) | Output neurons | Parameters # | Connected to |
|---|---|---|---|
| Input_layer | 2723 | 0 | |
| Dense_4 | 200 | 544800 | Input_layer |
| Dense_5 | 128 | 25728 | Dense_4 |
| Dense_6 | 25 | 5025 | Dense_4 |
| Total parameters | 575,553 | | |
| Trainable parameters | 575,553 | | |

Table 4.6 Multi-Task Learning Model

The Table 4.7 shows how the numbers of parameters shown in Table 4.6 are calculated.

| **Combining input layer from the previous layer:** 2723*200 = 544,600 | **Adding bias parameters for each of the neuron in the hidden layer:** 544600+200=545,800 |
|---|---|
| 200*128= 25,600 | **Adding bias parameters for each of the neuron in the output layer:** 25,600 + 128=25,728 |

Table 4.7 Multi-Task learning number of parameters calculation

The objective function from simple Categorical Cross entropy is changed as below in Equation (9):

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} y_j^i \log \hat{y}_j^i + \left(1 - y_j^i\right)\log(1 - \hat{y}_j^i) \qquad (9)$$

In equation (9), we are summing up the individual log loss functions. Total tasks are represented by ‘k’ whereas n represents the total training samples. $y$ represents the actual output for $j$ task for sample $i$. $\hat{y}$ represents the predicted values for $j$ task for sample $i$.

### 4.3.3 Continual Learning:

Continual learning is the ability to learn based on new data while retaining the knowledge from previous learning experience(59). A typical artificial neural network forgets its previous knowledge while learning based on current information. This forgetfulness in literature has been commonly referred as catastrophic forgetting (59)(60) . Artificial neural networks are considered liable to suffer from Catastrophic forgetting which means learning on current data will interfere with the previous learning information (61)(36). However, the phenomenon of catastrophic forgetting occurs when the network is trained in sequential pattern on different tasks and the weights of the networks changes according to the objectives of the new tasks (62) .

According to (63), learning and remembering different tasks would also help in generalizing artificial intelligence. In a complex environment, it could also help in achieving full autonomy by incrementally building its competence while considering tasks in a continuously growing capacity (64).

#### 4.3.3.1 Continual learning – Technical Implementation:

Different approaches have been used to include the continual learning capability in the artificial neural networks. (65) has mentioned 3 approaches based on the available dataset for learning which are explained below:

a. “Classical fine-tuning for new tasks”: Two given datasets are disjoint and the task in the second dataset changes.
b. “Continuous learning of known classes”: The cases in the 2nd dataset have known classes that the network has learned earlier while training on the 1st dataset. One might consider this approach resembling the online learning and the datasets could be perceived as continuously growing.
c. “Continuous learning of known and new classes”: It might happen that the 2nd dataset that is available for learning might have not only the already known classes but may also include some new unseen classes as well. So, the both available datasets may not be the same.

The well-suited strategy in the case of OpusCapita was case b, the case of continuously growing datasets. The paper further suggests a network strategy for all the 3 cases. For our case, the author suggests that the network can be trained on 2nd data set without any need of modification in the network architecture. This approach was followed, and the results of artificial neural network were compared with and without the continuous learning case.

# 5 Results:

### 5.1 Accuracy Metrics

Every algorithm has some performance measures that determine the level of accuracy or the performance level of that algorithm. OpusCapita needed three business metrics to determine how well the predictive algorithm is working:

$$Accuracy\%\ on\ Dev\ Test\ Set = \frac{Number\ of\ correctly\ predicted\ rows\ in\ Dev\ Test\ set}{Number\ of\ rows\ in\ Dev\ Test\ set} * 100$$

$$Accuracy\%\ on\ Training\ Set = \frac{Number\ of\ correctly\ predicted\ rows\ in\ training\ set}{Number\ of\ rows\ in\ Training\ set} * 100$$

$$Within\ Threshold\ Invoices\ \% = \frac{Number\ of\ rows\ predicted\ with\ Confidence > Threshold}{Total\ rows\ in\ Dev\ Test\ set} * 100$$

- Business has set a confidence threshold of 89% in Azure ML studio. So, algorithm performance with Azure ML Studio was compared, first on the basis of business set threshold. Then it was varied along a range of values to get an overall performance view of our implemented algorithm and compared it with Azure ML studio.

$$Accuracy\ WithinThreshold\ Invoices\ \% = \frac{Number\ of\ rows\ predicted\ with\ Confidence > Threshold}{Total\ rows\ in\ Dev\ Test\ set} * 100$$

### 5.2 General Comparison:

In this comparison, the best results that could be received from Azure ML studio were compared and classical model was applied on the same dataset. The data in each of the folds to be tested, is not guaranteed to be same but the strategy of data splitting used in both the cases is same.

#### 5.2.1 Customer1 Target Variable1:

Below are the accuracies produced by the classical model. The data used in the folds below in Table 5.1 is based on the random sampling and splitting. The splits are not guaranteed to have same data in each fold as Azure Folds:

| **Customer1 Target Variable1:** | **Fold1** | **Fold 2** | **Fold 3** | **Fold 4** | **Avg** |
|---|---|---|---|---|---|
| Accuracy% on Training Set | 94.58 | 94.99 | 95.11 | 94.54 | 94.80 |
| Accuracy% on Dev Test Set | 74.66 | 72.30 | 71.81 | 76.14 | 73.73 |
| Invoices% predicted with confidence >89% | 54.29 | 55.83 | 55.94 | 55.31 | 55.34 |
| Accuracy% on Invoices predicted with >89% confidence | 95.81 | 96.23 | 93.14 | 94.42 | 94.90 |
| Number of invoices predicted with confidence >89% | 1026 | 1034 | 758 | 860 | 919.5 |
| Number of invoices predicted with confidence >89%, correctly | 983 | 995 | 706 | 812 | 874 |
| batch_size* | 300 | 50 | 300 | 200 | -- |
| dropout_rate* | 0.0 | 0.2 | 0.2 | 0.2 | -- |
| optimizer: | adam | adam | adam | adam | -- |

*Table 5.1 Customer1 Target Variable1 statistics*

*values that are chosen by algorithm when grid search was run.

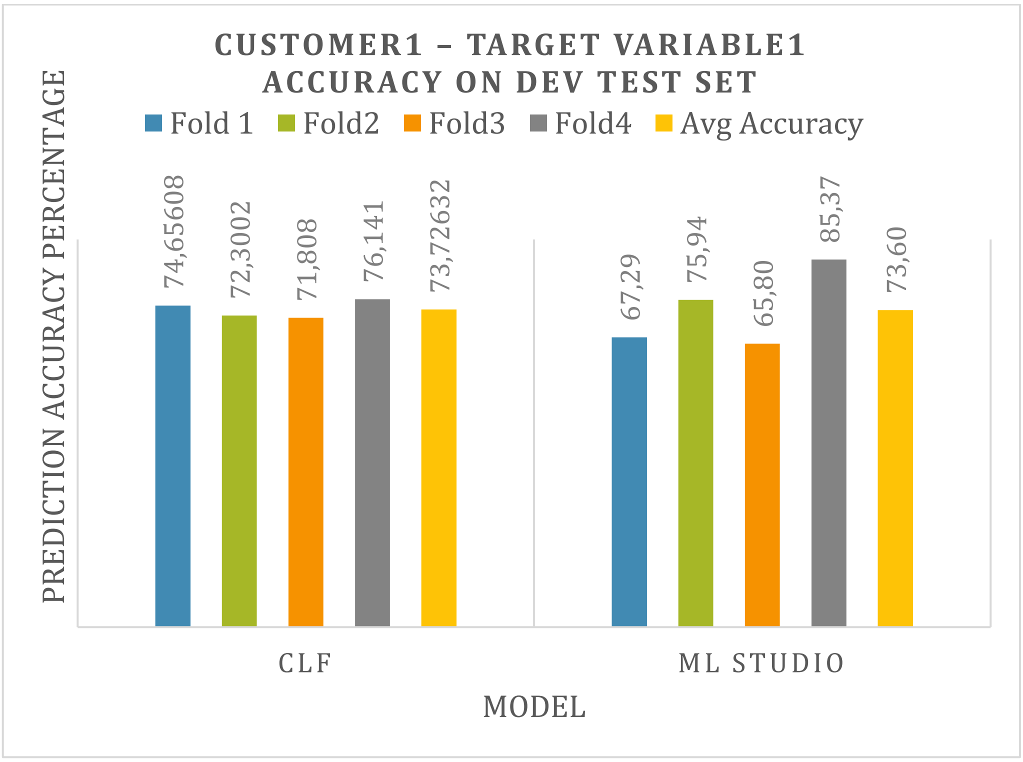

*Figure 5.1: Customer1 Target Variable1 – General Comparison Accuracy on Dev Test Set*

The Figure 5.1 shows the comparison of the investigated model and the ML Studio. On average, the model could do the predictions with 73.72% accuracy on the test sets. However, the accuracy achieved from ML studio on average is 73.60%.

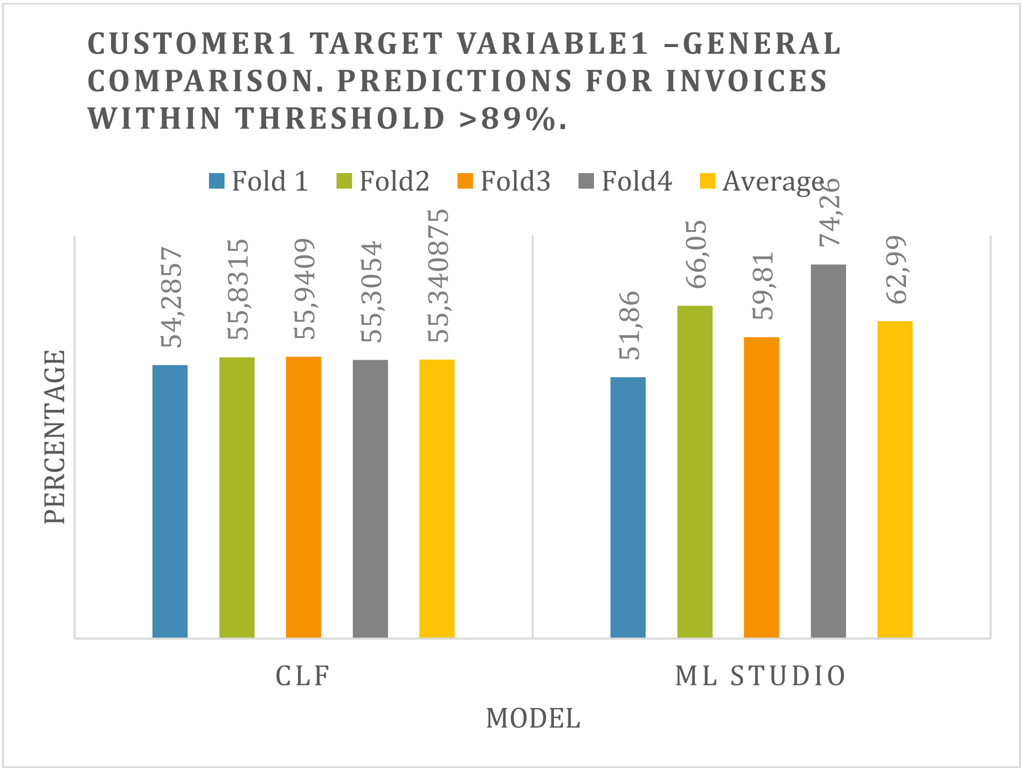

*Figure 5.2: Customer 1 Target Variable1- General Comparison*

In the Figure 5.2, the invoices that could be predicted with over 89% confidence were checked. The resulting model from the current study could predict around 55% invoices with more than 89% confidence. On the other hand, ML studio could predict 62 % invoices with more than 89% confidence.

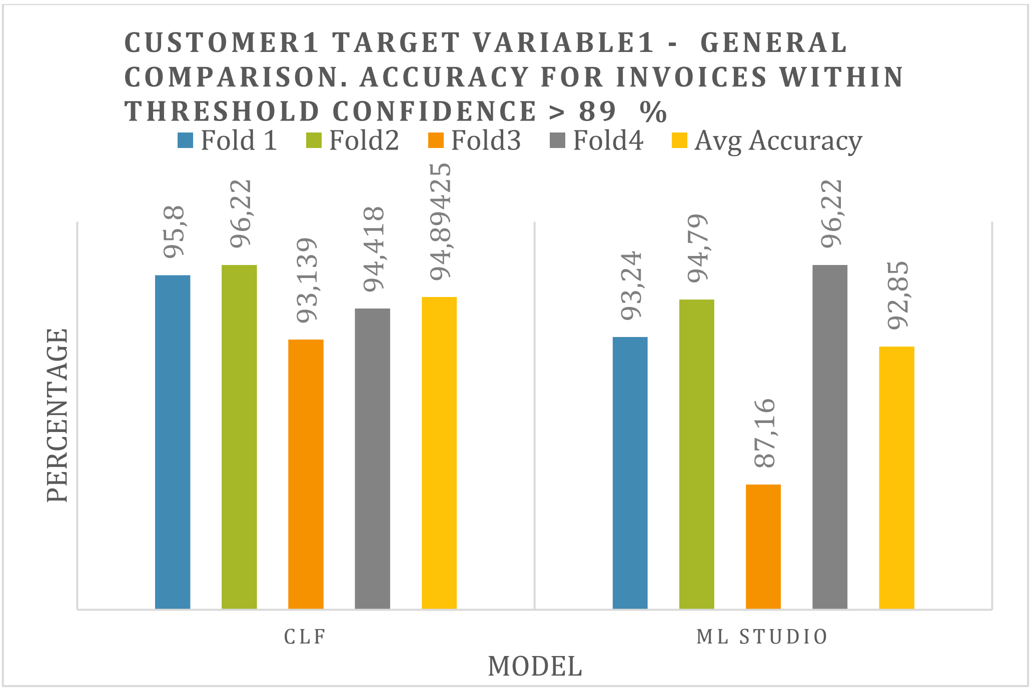

*Figure 5.3: Customer1 Target Variable1 – General Comparison Accuracy of With In Threshold invoices*

Furthermore, the accuracy of those invoices was compared, that were predicted over 89% confidence. The Figure 5.3 above shows the fold wise comparison of the prediction accuracies. The accuracy of currently investigated model was 94.89% while accuracy for ML studio was 92.85% for prediction of invoices.

### 5.2.2 Customer1 Target variable2:

Later the performance of the investigated model for the 2nd target variable was analysed which was referred as Target variable2. Table 5.2 shows the descriptive performance on accuracy.

| CLF | Fold1 | Fold2 | Fold3 | Fold4 | Average |
|---|---|---|---|---|---|
| *Accuracy% on Training Set* | 99,29 | 99,42 | 99,40 | 99,37 | 99,3693375 |
| *Accuracy% on Dev Test Set* | 97,82 | 96,02 | 95,80 | 94,47 | 96,02666 |
| *Invoices% predicted with confidence >89%* | 90,17 | 90,90 | 87,09 | 87,89 | 89,012 |
| *Accuracy% on Invoices predicted with >89% confidence* | 99,106 | 99,3743 | 98,8505 | 99,5249 | 99,213925 |
| *Number of invoices predicted with confidence >89%* | 1119 | 959 | 870 | 842 | -- |
| *Number of invoices predicted with confidence >89%, correctly* | 1118 | 953 | 860 | 838 | -- |

*Table 5.2: Customer1 Target Variable2*

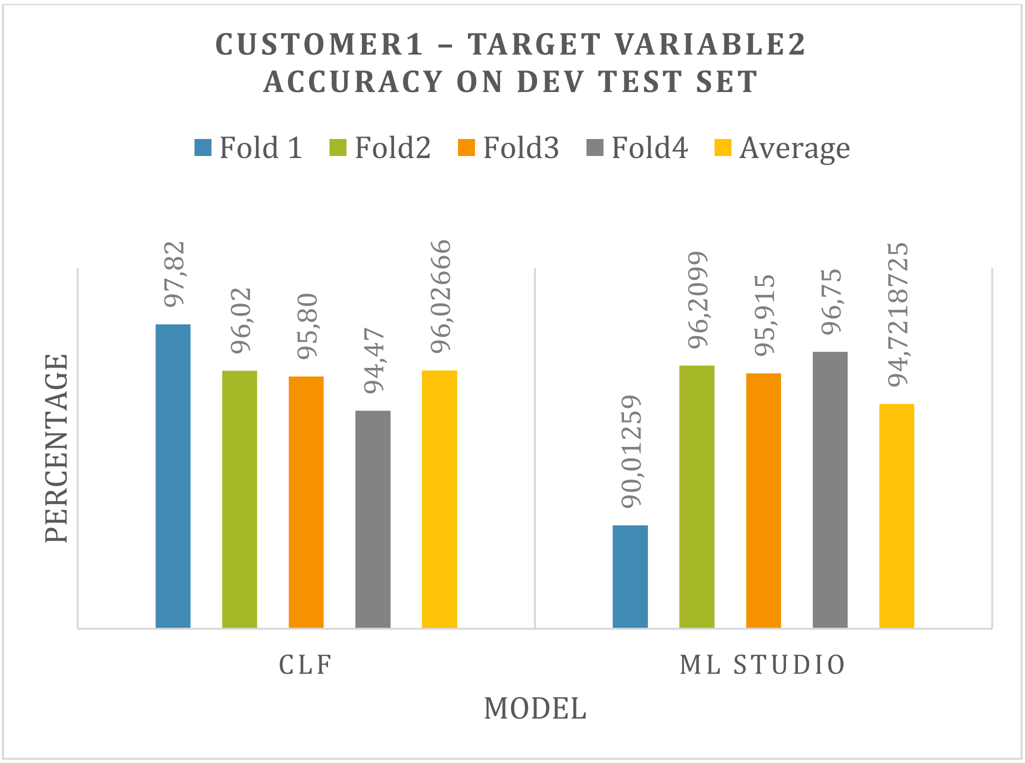

*Figure 5.4: Customer1 Target Variable2 – General Comparison*

The Figure 5.4 above shows the accuracy of the resulting model as well as the ML studio. The model investigated in the current study resulted in the predictions on the Dev Test set with 96% accuracy while we were able to get 94.7% accuracy by ML studio.

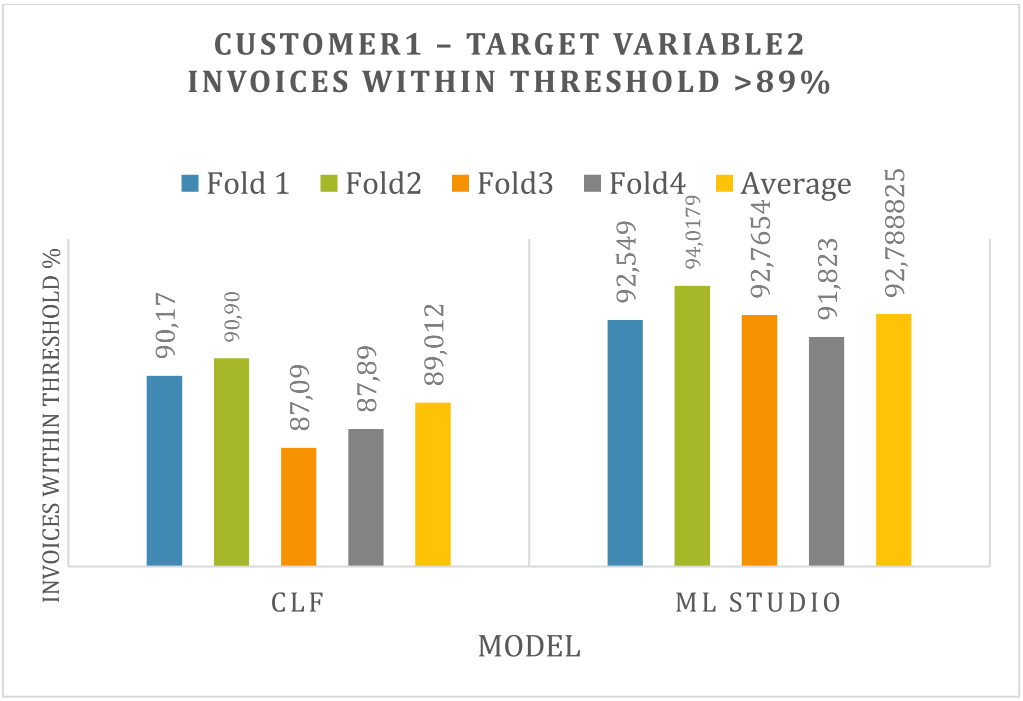

*Figure 5.5: Customer1 Target Variable2 – General Comparison*

Later on, the fold wise count of Invoices that were predicted by the algorithm were compared with a desired level of confidence of the prediction. Form Figure 5.5, it could be seen that on average, model proposed in the current study could predict 89% invoices with more than 89% confidence while ML studio could predict 92% invoices with more than 89% confidence.

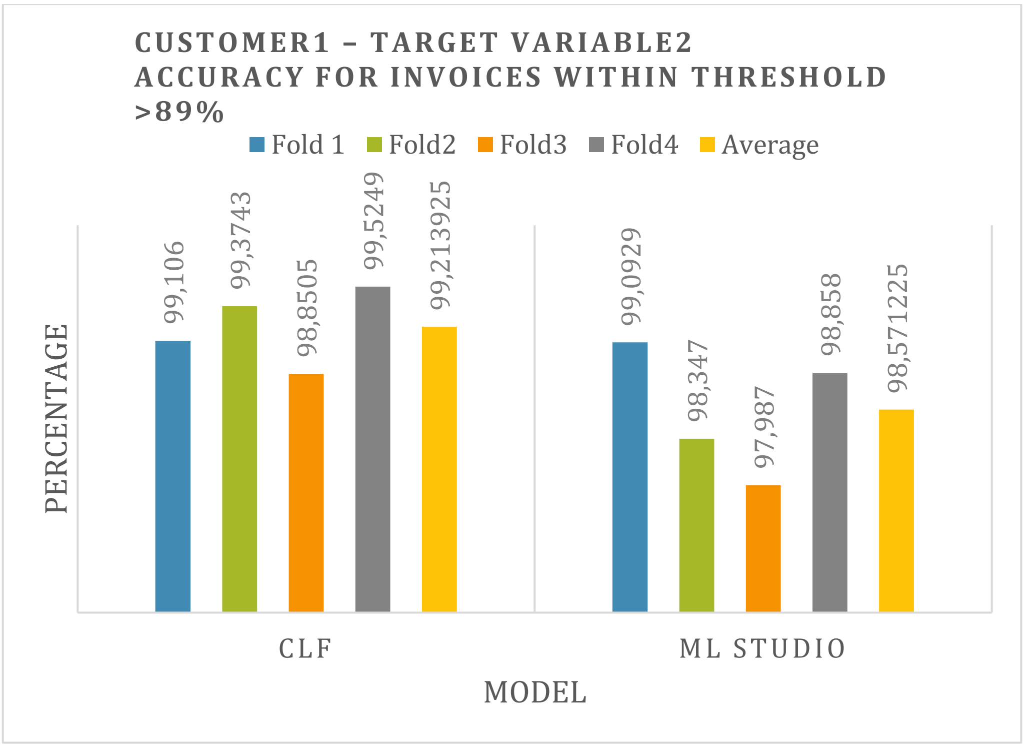

*Figure 5.6: Customer1 Target Variable2 – General Comparison*

After the performance of the algorithm with the ML studio on accuracy of those invoices that were predicted over 89% confidence were subject to comparison. This resulted in 99% accuracy on average in its prediction whereas on contrast ML studio was 98% accurate as displayed in Figure 5.6.

### 5.2.3 Customer2 Target Variable1:

Here the comparison of the performance of our algorithm with ML studio for data of Customer2 is displayed below Table 5.3 below shows the accuracy statistics.

| CLF | Fold1 | Fold2 | Fold3 | Fold4 | Average |
|---|---|---|---|---|---|
| *Accuracy% on Training Set* | 94,93 | 93,23 | 90.14 | 92,13 | 92.61 |
| *Accuracy% on Dev Test Set* | 71.59 | 74.14 | 72.63 | 73.26 | 72.90 |
| *Invoices% predicted with confidence >89%* | 52.78 | 55.77 | 52.32 | 53.03 | 53.47 |
| *Accuracy% on Invoices predicted with >89% confidence* | 91.15 | 91.51 | 91.45 | 91.68 | 91.44 |
| *Number of invoices predicted with confidence >89%* | 3457.09 | 3558,12 | 3428 | 3557,78 | 3500,24 |
| *Number of invoices predicted with confidence >89%, correctly* | 3151.13 | 3256.04 | 3134,92 | 3261.77 | 3200.96 |

*Table 5.3: Customer2 Target Variable1*

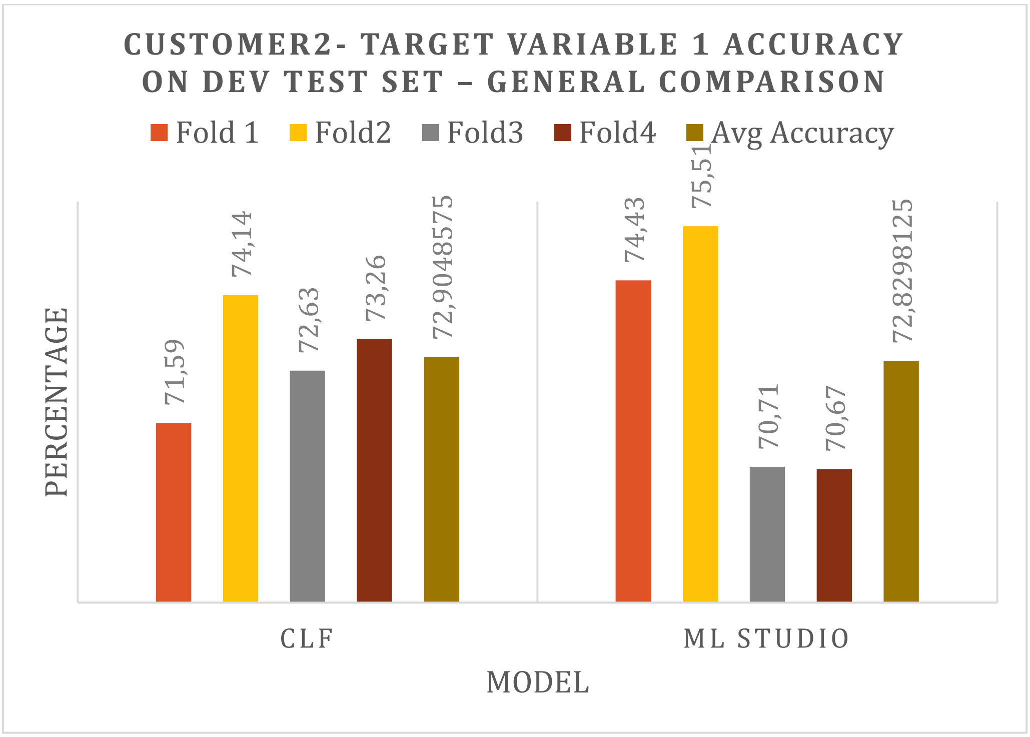

*Figure 5.7: Customer2 Target Variable1 – General Comparison– Accuracy on Dev Test Set*

The Table 5.3 and Figure 5.7 above shows the fold wise comparison of the accuracies on Dev Test set. The model investigated in the current study was able to predict with 72.9% confidence while we almost same level of accuracy could be achieved from ML studio as well.

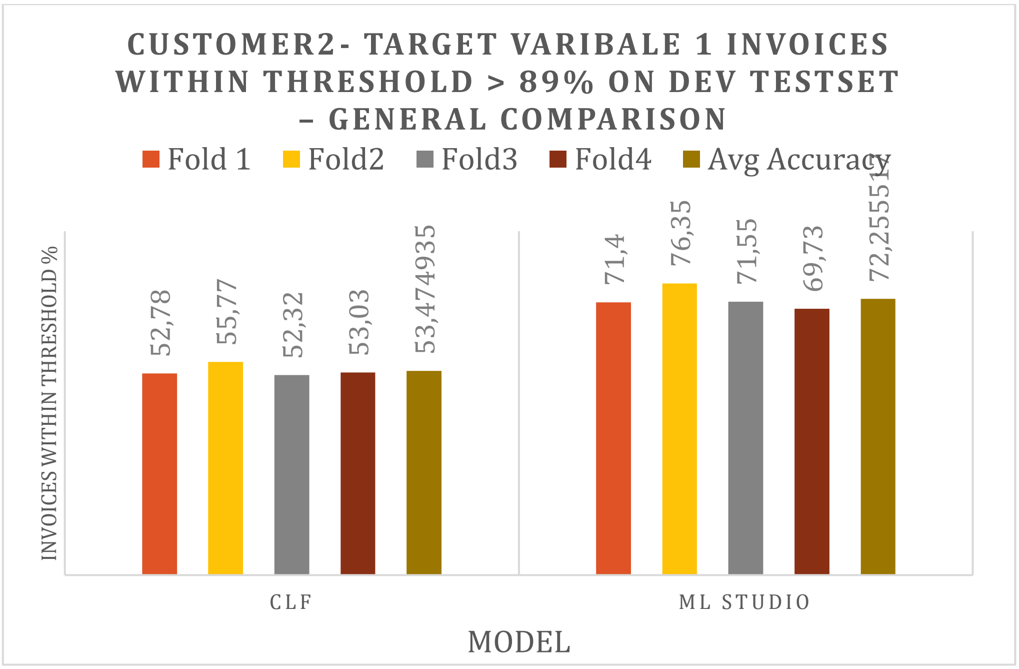

*Figure 5.8: Customer2 Target Variable1 – General Comparison*

Then the performance of the algorithm by determining how many invoices could be predicted with over 89% confidence was examined. It is worth mentioning that the model investigated in the current study could predict more than 50% invoices with more than 89% confidence. However, ML studio could predict 72% invoices with more than 89% confidence as shown in Figure 5.8 above.

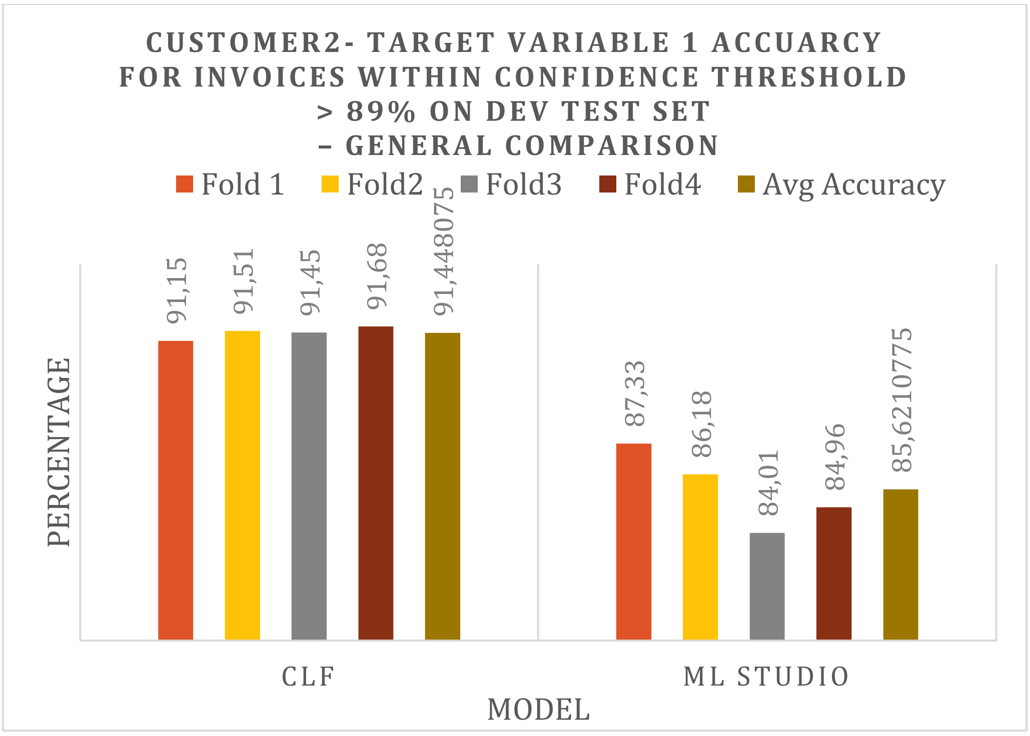

*Figure 5.9: Customer2 Target Variable1 – General Comparison*

In the Figure 5.9 above, it can be seen that the investigated model gives 91% accuracy with > 89% confidence. However, in spite of the fact that ML studio predicted 72% invoices with >89% confidence. Its accuracy is pretty low as compared to the devised model which is 85% accurate.

### 5.2.4 Customer2 Target Variable2:

The performance of our model for Customer2 for Target variable2 was compared. The key accuracy measures are shown in Table 5.4.

| | *Fold1* | *Fold2* | *Fold3* | *Fold4* | *Average* |
|---|---|---|---|---|---|
| *Accuracy% on Training Set* | 99.80 | 99.80 | 99.60 | 99.80 | 99.75 |
| *Accuracy% on Dev Test Set* | 95.63 | 95.86 | 96.56 | 93.89 | 95.4829 |
| *Invoices% predicted with confidence >89%* | 89 | 89 | 89 | 89 | 89 |
| *Accuracy% on Invoices predicted with >89% confidence* | 88.72 | 90.75 | 91.33 | 89.23 | 90.004875 |
| *Number of invoices predicted with confidence >89%* | 99.05 | 99.39 | 99.84 | 98.37 | 99.160025 |
| *Number of invoices predicted with confidence >89%, correctly* | 629 | 657 | 611 | 613 | 627.5 |
| *Accuracy% on Training Set* | 623 | 653 | 610 | 603 | 622.25 |

*Table 5.4 Customer2 Target Variable2 – General Comparison*

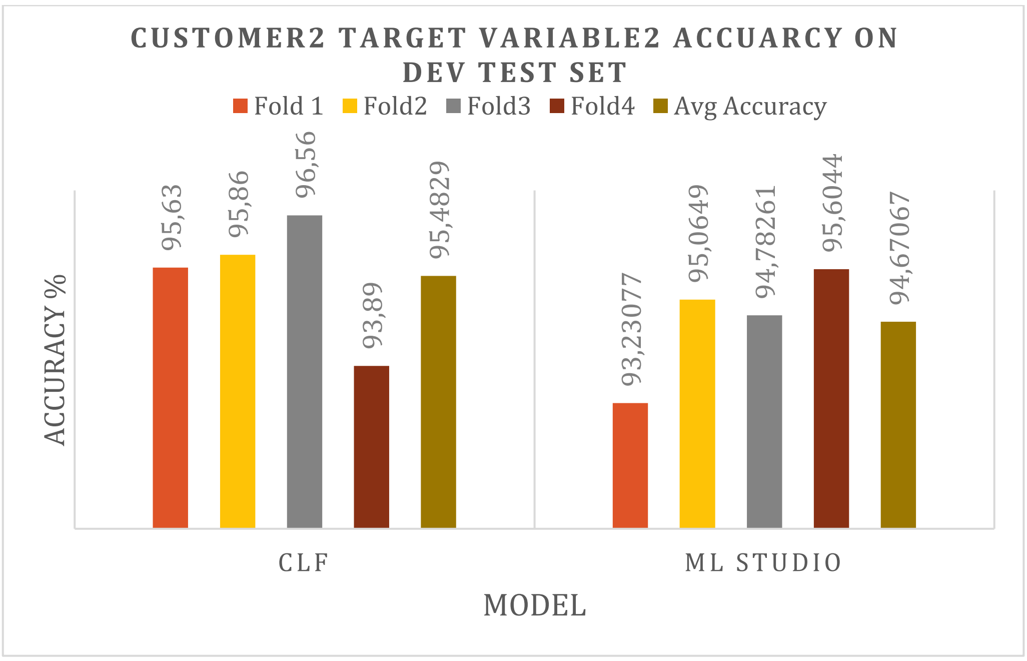

*Figure 5.10: Customer2 Target Variable2 – General comparison – Accuracy on Dev Test Set*

In the Figure 5.10 above, it can be seen that the last bar on left side is slightly higher than the last bar on the right side. The model investigated in the current study gave 95% accuracy on Dev Test set for Target Variable2 on the other hand ML studio gave 94.67% accuracy.

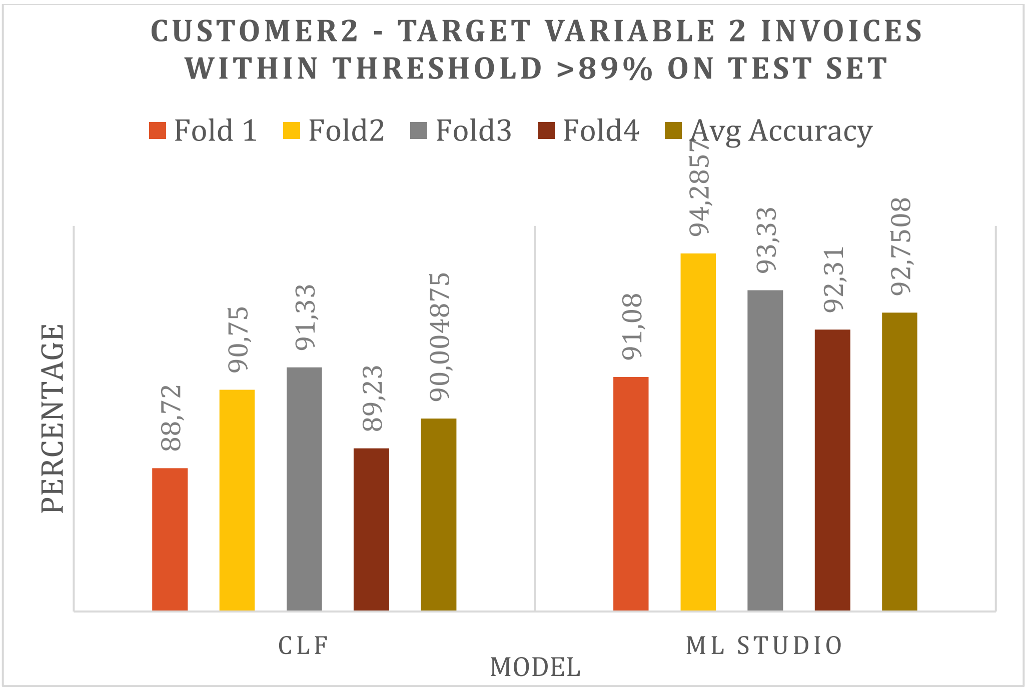

*Figure 5.11: Customer2 Target Variable2 – General Comparison – Invoices predicted with over89% confidence*

For invoices that are predicted with more than 89% confidence, The model investigated in the current study was able to predict 90% while ML studio performed slightly better resulting in 92% invoice prediction with the set level of confidence as shown in Figure 5.11.

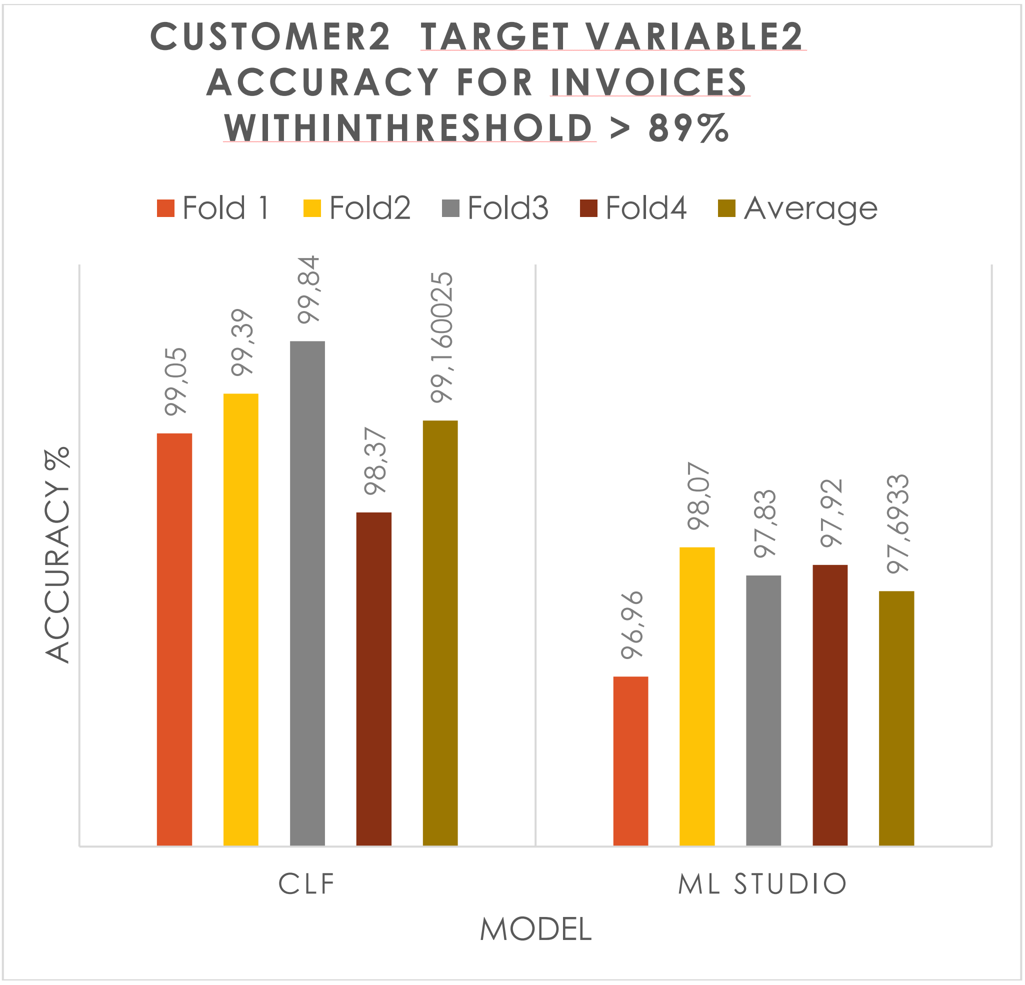

*Figure 5.12: Customer2 Target Variable2 - Accuracy for Invoices predicted with over 89% Confidence*

Now, in the Figure 5.12 above, the prediction accuracy for invoices that were predicted with more than 89% confidence could be seen. The model investigated in the current study was 99% accurate while predicting 90% invoices with > 89% confidence. While ML studio was 97% accurate in predicting 92% invoices with > 89% confidence.

## 5.3 Exactly Same Fold Comparison:

The comparison of classical algorithm and Azure ML studio mentioned above was conducted on same datasets, but the data in each fold wasn't guaranteed to be exactly the same. Therefore, the algorithm performance on exactly the same data was tested in each of the four folds that Azure ML studio was using in its 4-fold. The data for these folds was taken from Azure ML studio. Below in Table 5.5 are the results obtained from our classical platform:

### 5.3.1 Customer1 Target Variable1:

| ***Target Variable1*** | ***Fold1*** | ***Fold 2*** | ***Fold 3*** | ***Fold 4*** | ***Average*** |
|---|---|---|---|---|---|
| *Accuracy% on Training Set* | 94,30 | 94,18 | 94,66 | 94,38 | 94,38 |
| *Accuracy% on Dev Test Set* | 72,42857 | 74,11 | 66,59 | 84,5122 | 74,41 |
| *Invoices% predicted with confidence >89%* | 52,14 | 62,41 | 51,47 | 73,05 | 59,77 |
| *Accuracy% on Invoices predicted with >89% confidence* | 95,4415 | 95,352 | 90,3508 | 97,1619 | 94,58 |
| *Number of invoices predicted with confidence >89%* | 351 | 581 | 524 | 599 | 514 |
| *Number of invoices predicted with confidence >89%, correctly* | 335 | 554 | 438 | 582 | 482 |
| *batch_size** | 50 | 50 | 200 | 100 | -- |
| *dropout_rate** | 0,10 | 0,2 | 0,1 | 1,1 | -- |

*Table 5.5 : Classical Model performance on exactly same data in each fold as in Azure ML Studio*

*Indicates those parameters that are selected by Grid Search from the available options.

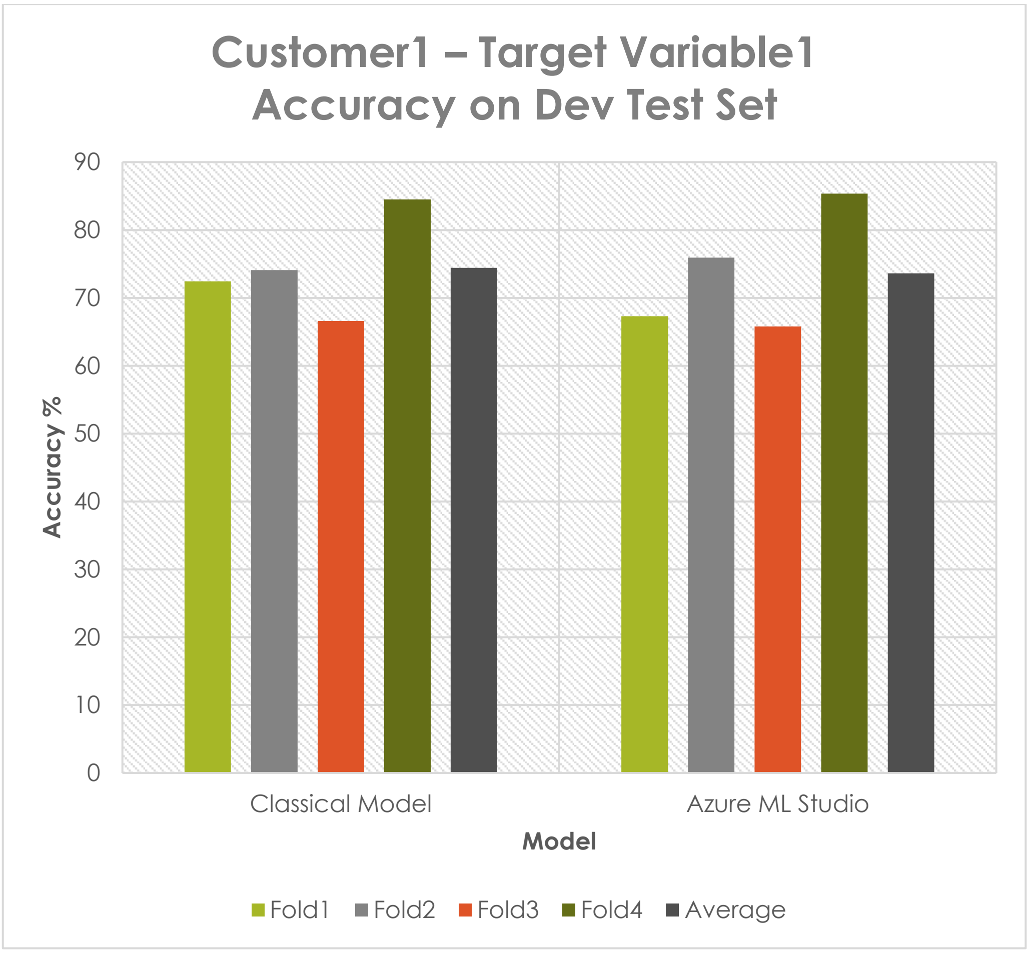

*Figure 5.13: Customer1 Target Variable1 – Accuracy on Dev Test Set*

In the Figure 5.13 above, the performance of the resulting model with the ML studio on exactly same test data was compared (Dev Test Set). The devised classical model gave the accuracy 74.41% on average which is slightly higher than the performance of the ML studio 73.6%.

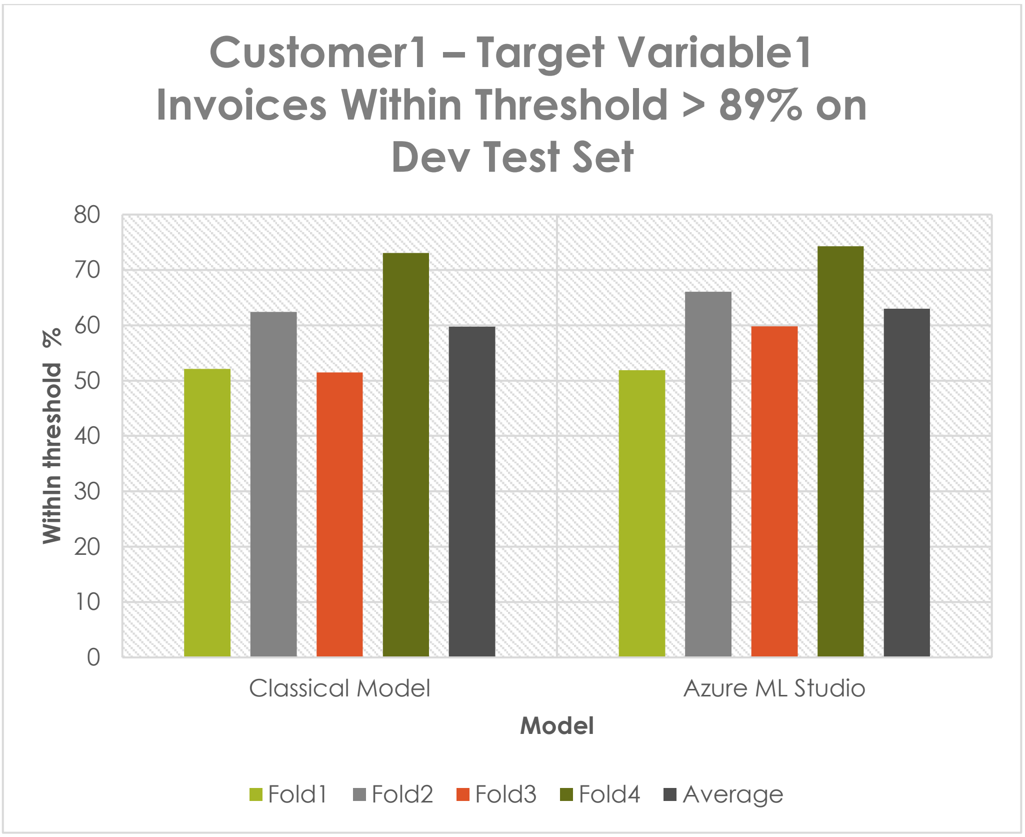

*Figure 5.14: Customer1 Target Variable1 – Invoices that are predicted with over 89% confidence*

In this Figure 5.14, the percentage of Invoices that fall within the set Threshold was compared. On average 59% invoices fall within threshold in the tested classical model which is slightly less than the 62% of ML studio.

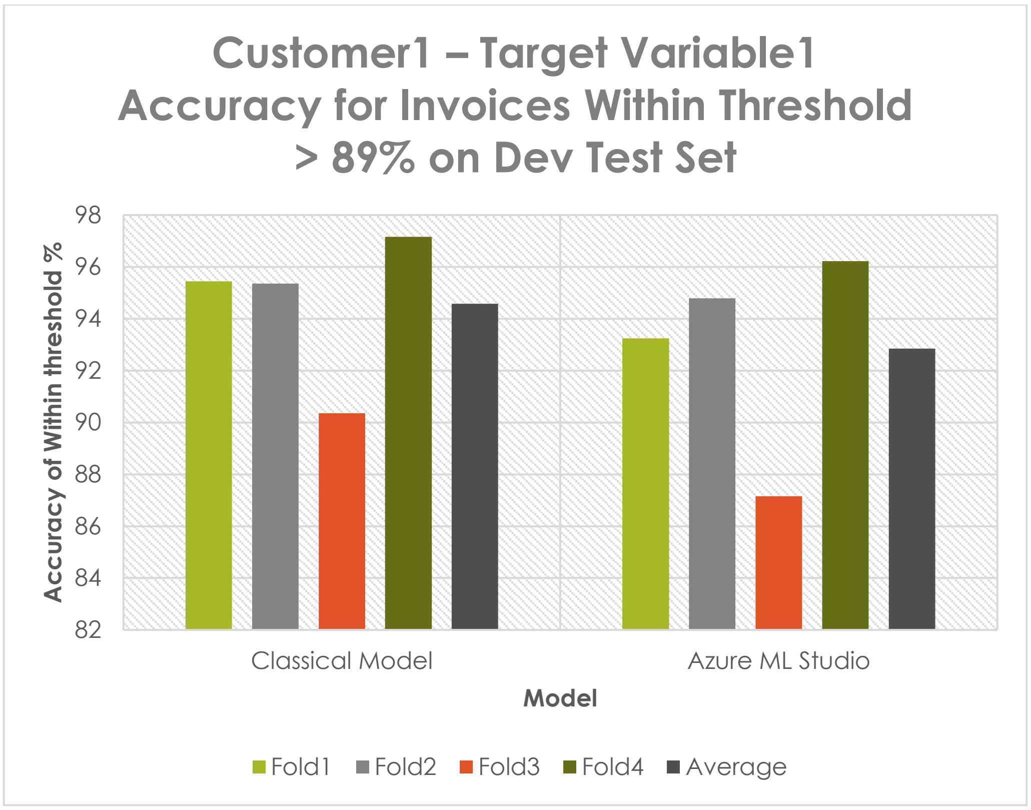

*Figure 5.15: Customer1 Target Variable1 – Accuracy for Invoices predicted with over 89% Confidence*

In the Figure 5.15 above, we compared the 3rd business Metric. The model investigated in the current study gave 94.58% accuracy for those invoices that fall within the set threshold of greater than 89% prediction confidence. This accuracy is higher than the accuracy achieved from ML Studio which is 92.85%.

In the figures below, the confidence is varied between threshold 0 and 99 to determine the count of invoices that are predicted with that confidence. It also shows the accuracy of those within confidence threshold invoices.

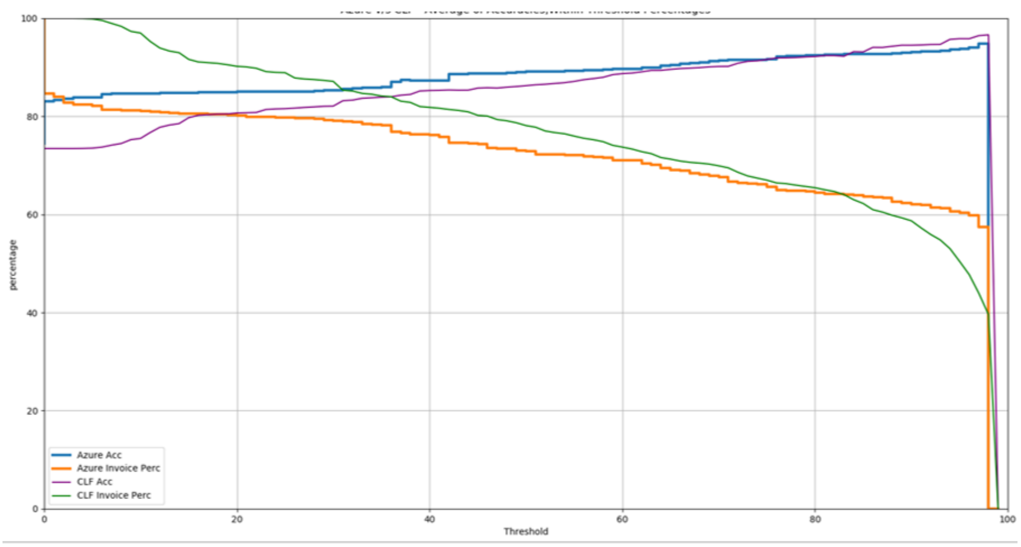

*Figure 5.16 Customer1 Target Variable1*

The Figure 5.16 above displays the average of the results of the four folds. The classical model investigated in the current study has shown better performance in terms of accuracy when the threshold was set to about 85% or higher confidence level. In terms of count of invoices that are predicted with this confidence, classical model represented as CLF predicts less invoices than Azure when the confidence level is set to about 85% or higher. From the Figure 5.16, above it can be said that the devised algorithm performed well on Business metrics for Customer1 Target Variable1.

### 5.3.2 Customer1 Target Variable2:

| | ***Fold1*** | ***Fold 2*** | ***Fold 3*** | ***Fold 4*** | ***Average*** |
|---|---|---|---|---|---|
| **Accuracy% on Training Set** | 99,4459 | 99,4025 | 99,4873 | 99.3910 | 99,431 |
| **Accuracy% on Dev Test Set** | 97,4404 | 93,2984 | 96,5708 | 94,7876 | 95,5243 |
| **Invoices% predicted with confidence >89%** | 91,350 | 85,4450 | 92,7710 | 92.0849 | 90,412 |
| **Accuracy% on Invoices predicted with >89% confidence** | 99,323 | 99,7549 | 99,100 | 98,846 | 99,159 |
| **Number of invoices predicted with confidence >89%** | 1035 | 816 | 1001 | 954 | 952 |
| **Number of invoices predicted with confidence >89%, correctly** | 1028 | 814 | 992 | 943 | 944 |
| **batch_size*** | 50 | 50 | 100 | 50 | -- |

*Table 5.6: Customer1 Target Variable2 fold wise performance on exact same data*

*represents the values that were selected by the Grid Search of the Algorithm from the given range of values.

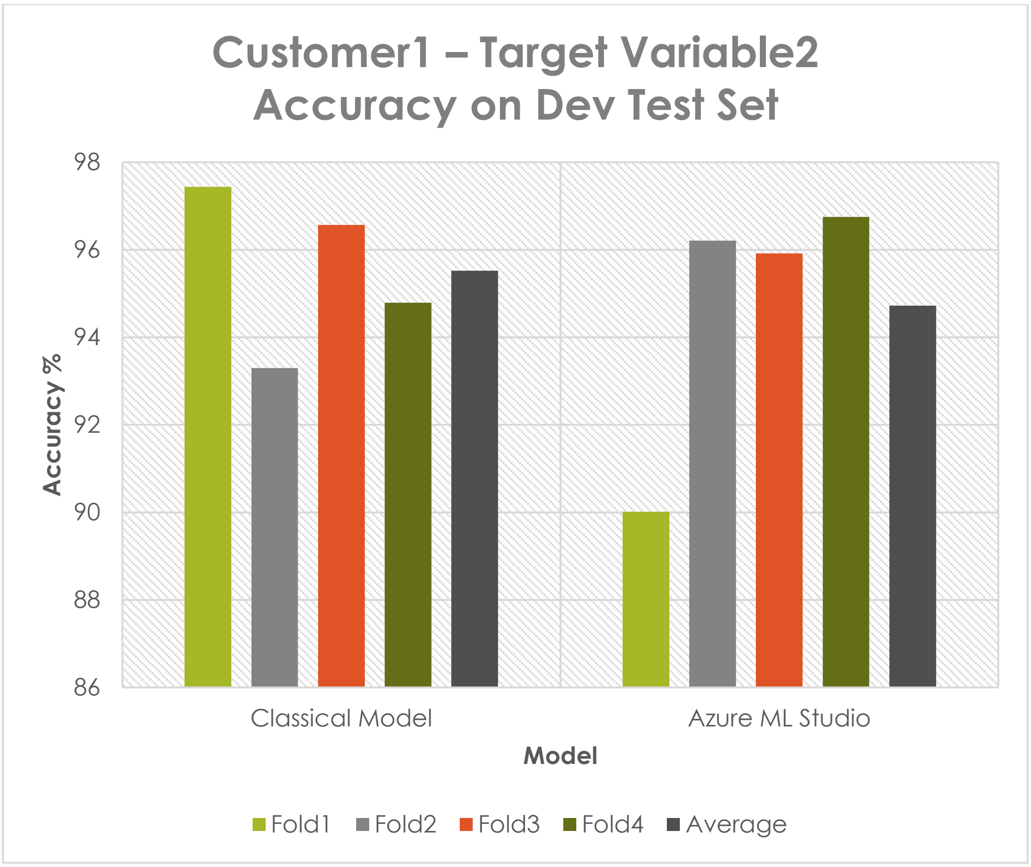

*Figure 5.17: Customer 1 Target Variable2 – Accuracy on Dev Test Set*

In the Figure 5.17 above, the 1st business metric for Target Variable2 was compared. The accuracy on Dev Test set for the investigated model on average is 95.52% while the accuracy obtained for the exact same data from ML studio was 94.72%.

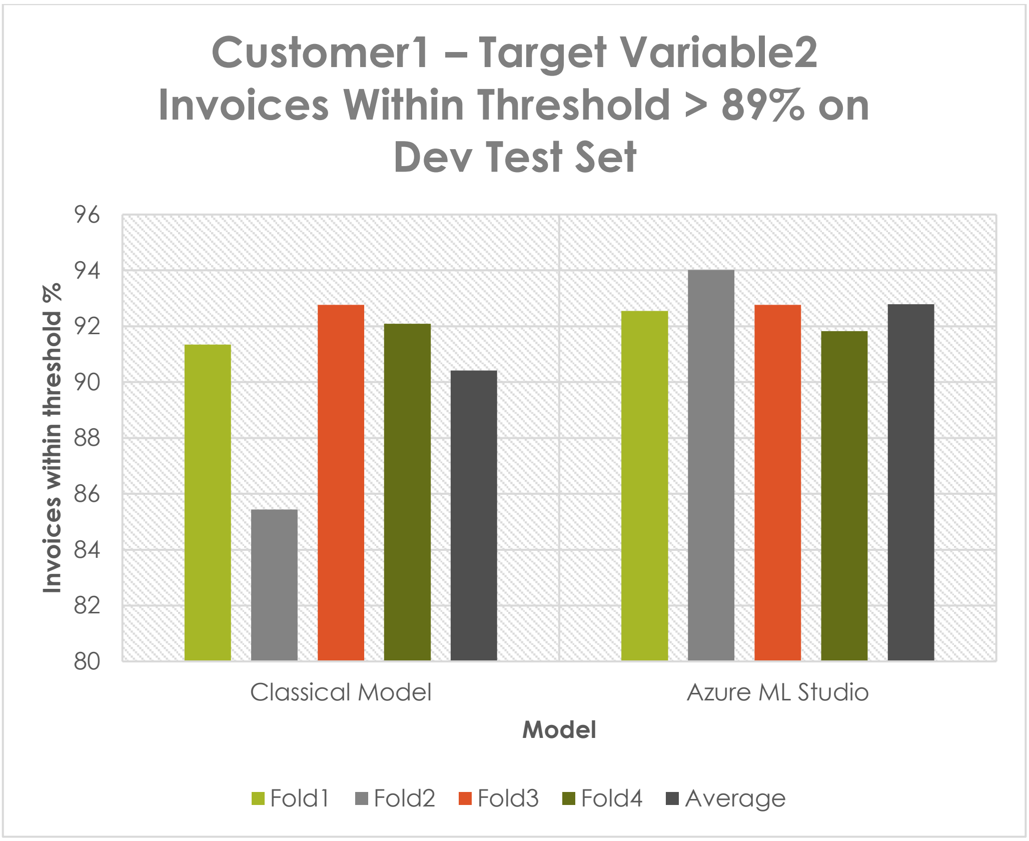

*Figure 5.18: Customer1 Target Variable2 – Invoices predicted with over 89% Confidence*

The Figure 5.18 shows the fold wise comparison for how many invoices can be predicted with more than 90% Confidence. The proposed model could predict 90.41% invoices with > 89% confidence while ML studio could predict 92% invoices with > 89% confidence.

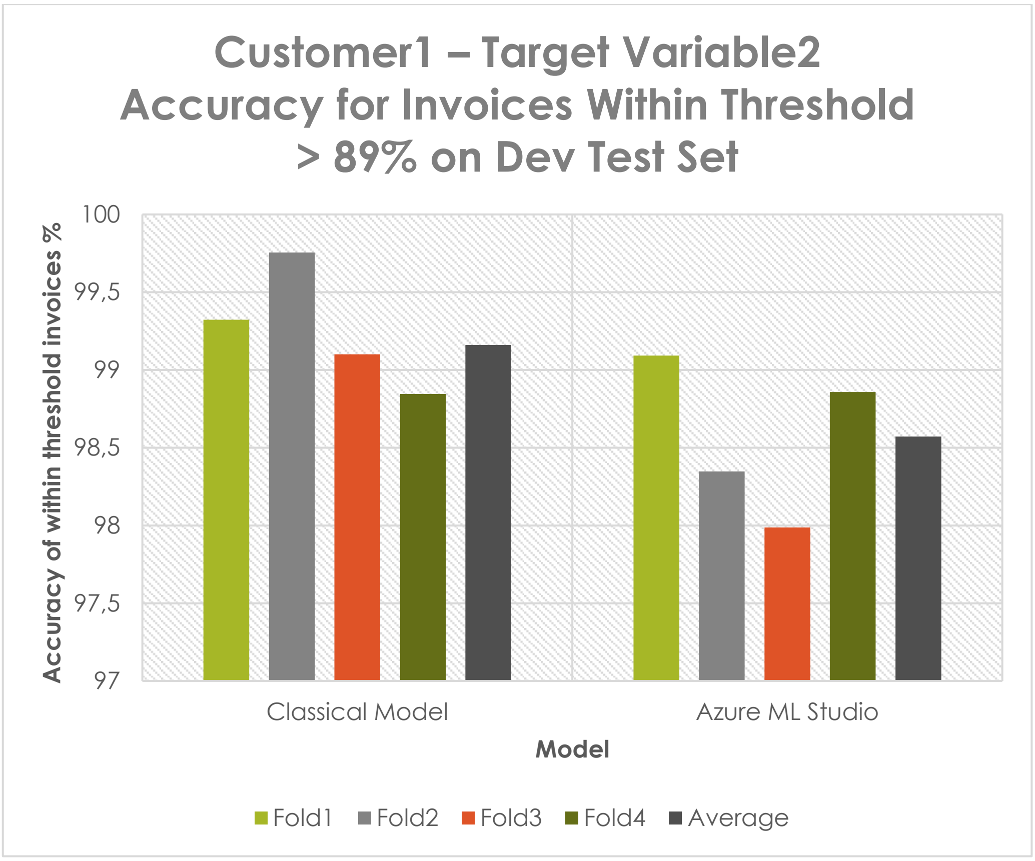

*Figure 5.19: Customer1 Target Variable2 - Accuracy for Invoices predicted with over 89% Confidence*

The Figure 5.19 displays the fold wise comparison of the accuracy of those invoices that were predicted with more than 89% confidence. The investigated model could predict invoices with over 99.15% accuracy whereas the ML studio could predict the invoices within threshold with 98.57% accuracy.

The performance of the devised classical model for Customer1 Target Variable2 was compared with the performance of Azure ML studio. The average of the results was taken as four folds and was plotted in the graph below:

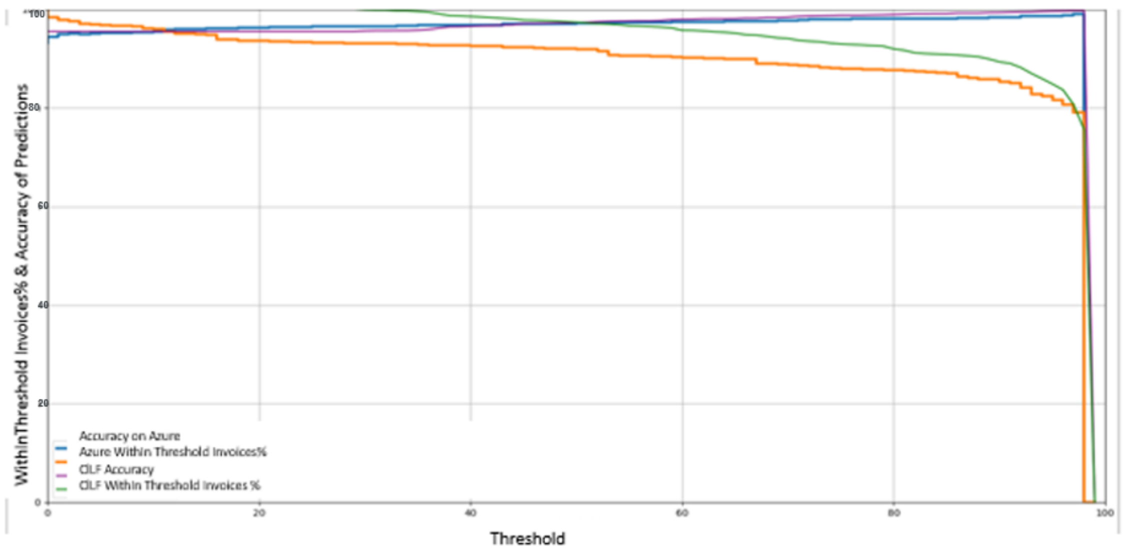

*Figure 5.20 Customer1 Target Variable2*

As depicted in the Figure 5.20 above, the threshold for confidence of prediction is varied on the horizontal X-axis, the lines, in "Purple" representing the accuracy of predictions by the Classical model and "Green" representing the accuracy of prediction of Azure ML studio, stayed very close to each other. For the percentage of Invoices that fall within the threshold of Confidence, the line in green colour stayed above the line in orange colour. This graph indicates that the investigated classical model performed initially similar however it improved later as the confidence threshold was varied. This resulted in improved confidence than Azure ML Studio for most of the threshold values.

### 5.3.3 Customer2 Target Variable1:

Next we compared the results of the Model under study with ML studio for customer 2 Target variable 1. For this comparison, the data set belongs to Customer2 and the folds used for ML studio and for the investigated model are the same.

| ***Target Variable1*** | ***Fold1*** | ***Fold2*** | ***Fold3*** | ***Fold4*** | ***Average*** |
|---|---|---|---|---|---|
| **ACCURACY% ON TRAINING SET** | 93,94 | 93,73 | 93,64 | 93,76 | 93,766 |
| **ACCURACY% ON DEV TEST SET** | 75,88 | 74,47 | 72,15 | 72,75 | 73,814 |
| **INVOICES% PREDICTED WITH CONFIDENCE >89%** | 56,44 | 57,08 | 55,66 | 52,67 | 55,462 |
| **ACCURACY% ON INVOICES PREDICTED WITH >89% CONFIDENCE** | 94,0408 | 91,0228 | 91,0204 | 91,03 | 91,778 |
| **NUMBER OF INVOICES PREDICTED WITH CONFIDENCE >89%** | 1762 | 1838 | 1715 | 1728 | 1761 |
| **NUMBER OF INVOICES PREDICTED WITH CONFIDENCE >89%, CORRECTLY** | 1657 | 1673 | 1561 | 1573 | 1616 |

Table 5.7*: Fold wise comparison of Target Variable1 for Customer2 on exact same data.*

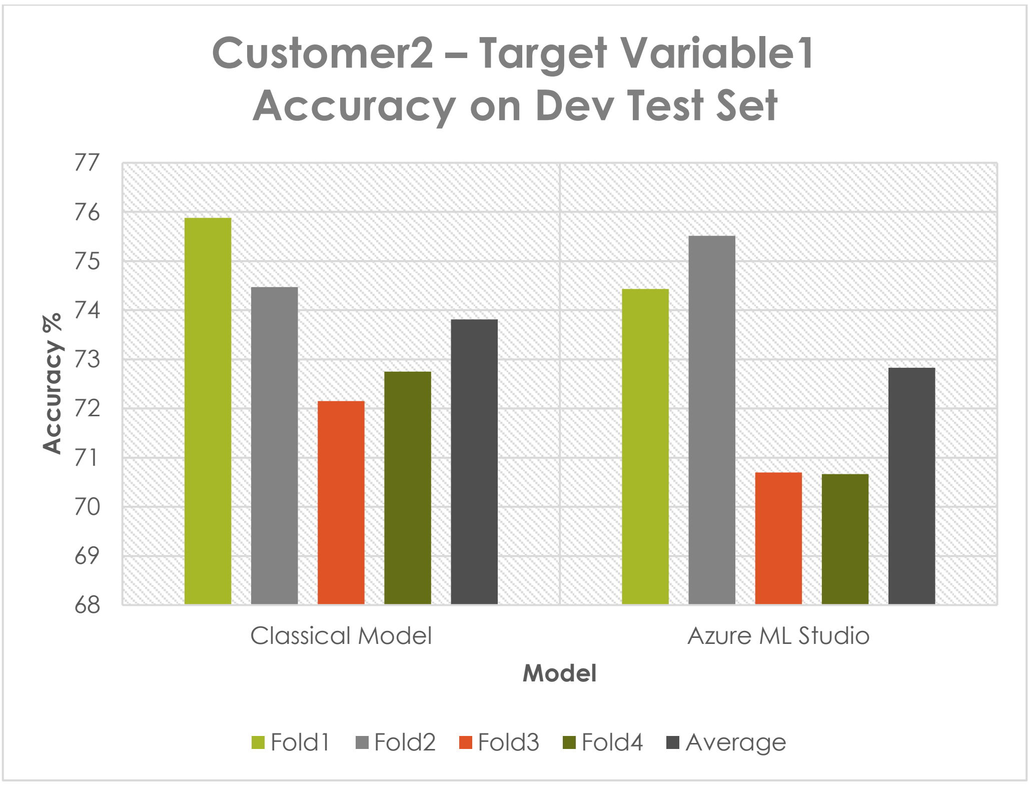

*Figure 5.21: Customer2 – Target Variable1 Accuracy on Dev Test Set*

The Figure 5.21 displays the fold wise comparison for the 4-fold cross validated accuracies of the model under study and the ML studio on exactly same data sets. The model under investigation could predict the Target Variable1 with 73.81% accuracy while on average ML studio can predict the invoices with 72.82% accuracy on the other hand.

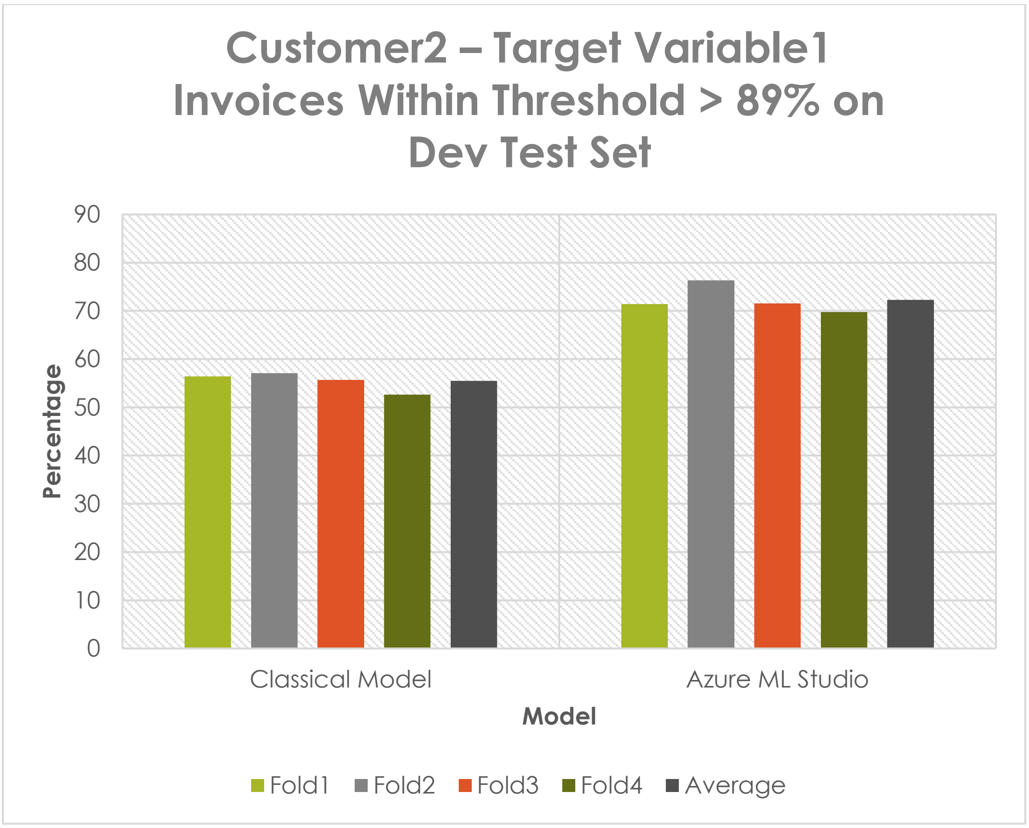

*Figure 5.22: Percentage of invoices that are predicted with over 89% confidence.*

In the Figure 5.22 above, fold wise comparison for invoices that have been predicted with more than 89% confidence can be seen. The investigated model could predict 55.46% invoices on average, while ML studio could predict 72.25% invoices with more than 89% confidence.

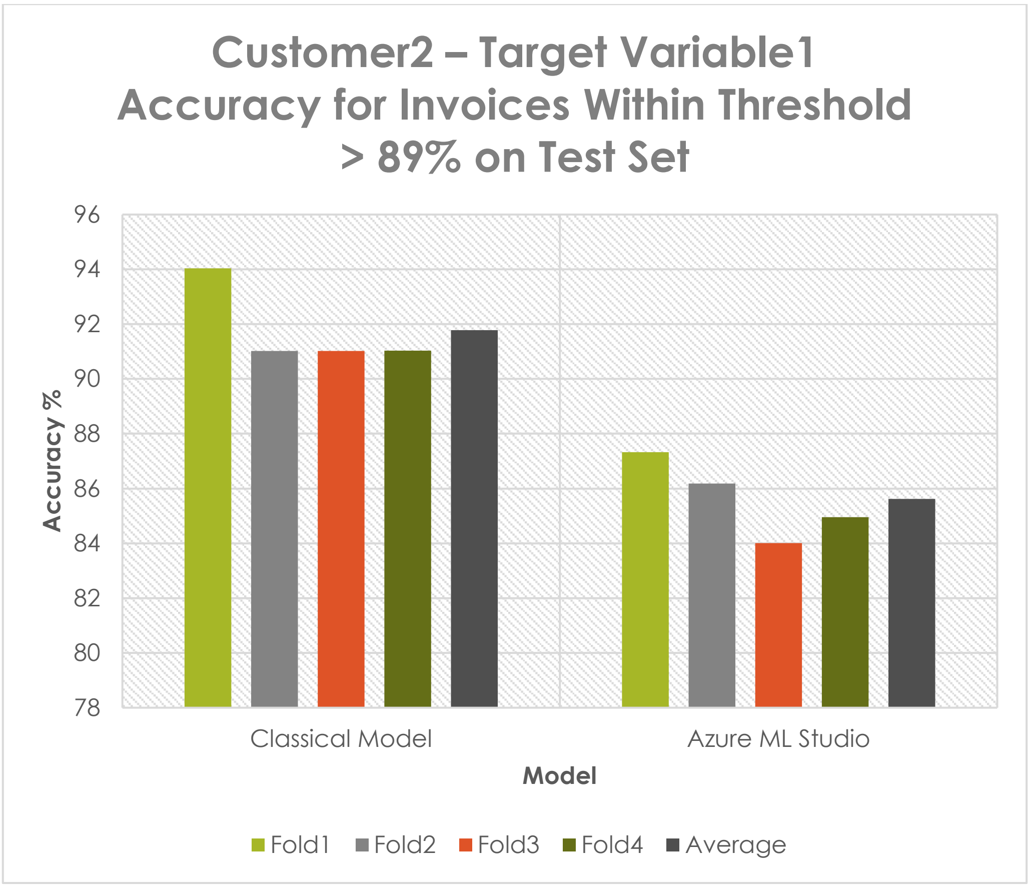

*Figure 5.23: Customer2 Target Variable1, Accuracy of invoices*

In the Figure 5.23, the fold wise comparison of accuracy for those invoices that were predicted with more than 89% confidence is displayed. The classical model under investigation on average, gave the accuracy of 91.77% while ML studio could give the accuracy of 85.62% only.

The next step was to compare the average performance of the four folds cross validation by varying threshold along the x-axis.

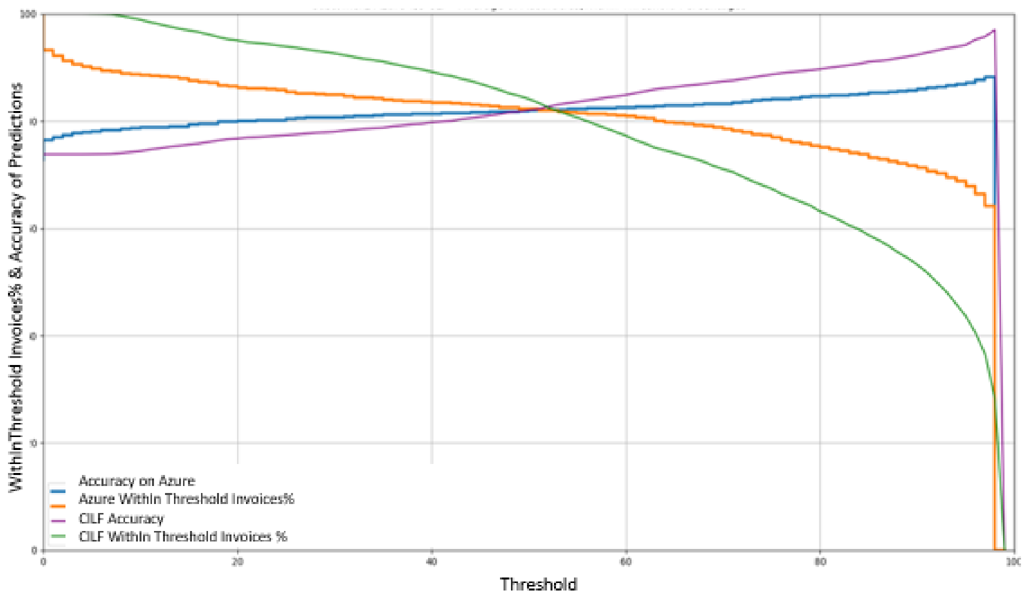

*Figure 5.24: ML Studio v/s Classical model for various Confidence thresholds.*

In the Figure 5.24 describing the trend for Customer2 above, the slopes appeared sharp as compared to the plots of Customer1 for Target variable1. The average common cut point (the point where the lines are intersecting each other) indicates a confidence threshold between 50% - 60%, where the classical model under study started predicting less invoices with better accuracy as compared to ML studio which is predicting more invoices at the expense of accuracy.

### 5.3.4 Customer2 Target Variable2:

Next of the results for fold wise comparison of Target variable2 for Customer2 were analyzed. The accuracy comparison results are shown in Table 5.8.

| **Target variable2** | ***Fold1*** | ***Fold 2*** | ***Fold 3*** | ***Fold 4*** | ***Average*** |
|---|---|---|---|---|---|
| **ACCURACY% ON TRAINING SET** | 99,60 | 99,59 | 99,50 | 99,68 | 99,593 |
| **ACCURACY% ON DEV TEST SET** | 93,54 | 92,21 | 98,26 | 94,51 | 94,628 |
| **INVOICES% PREDICTED WITH CONFIDENCE >89%** | 87,69 | 87,27 | 91,01 | 91,21 | 89,297 |
| **ACCURACY% ON INVOICES PREDICTED WITH >89% CONFIDENCE** | 97,8947 | 98,511 | 99,363 | 98,4939 | 98,565 |
| **NUMBER OF INVOICES PREDICTED WITH CONFIDENCE >89%** | 285 | 336 | 314 | 332 | 316,75 |
| **NUMBER OF INVOICES PREDICTED WITH CONFIDENCE >89%, CORRECTLY** | 279 | 331 | 312 | 327 | 312.25 |

*Table 5.8: Fold wise comparison of Target Variable2 for Customer2*

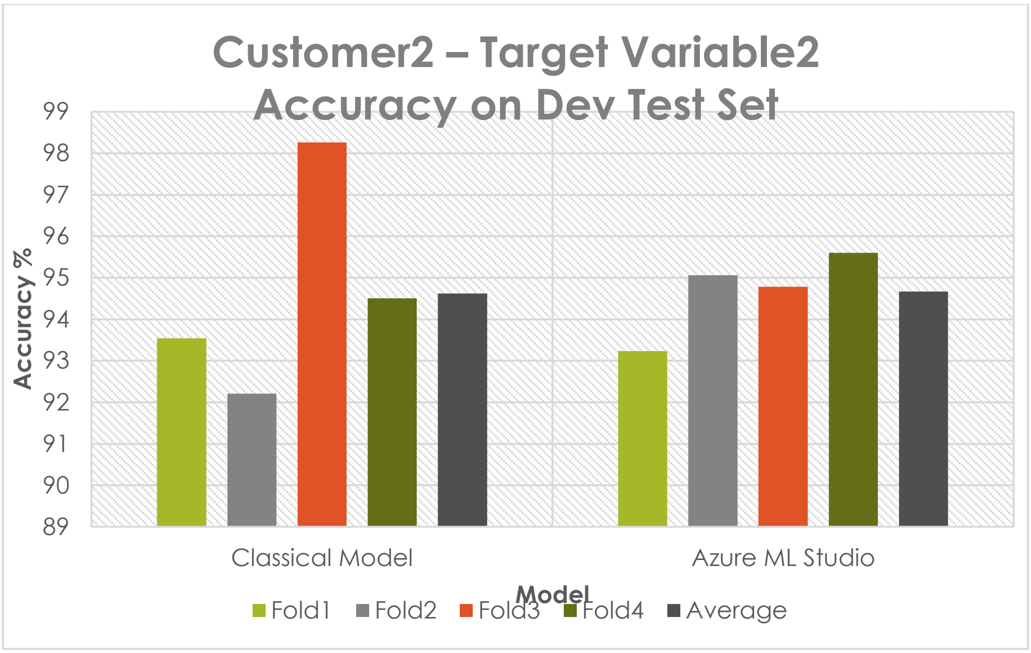

*Figure 5.25: ML Studio v/s Classical model- Customer2 Target Variable2 Accuracy on Dev Test Set*

The Figure 5.25 deciphers the fold wise comparison of the 4 folds for our model and the ML studio. On average, the investigated classical model predicted invoices with 94.62% accuracy on the other hand ML studio predicted the same data for almost 94.67% accuracy.

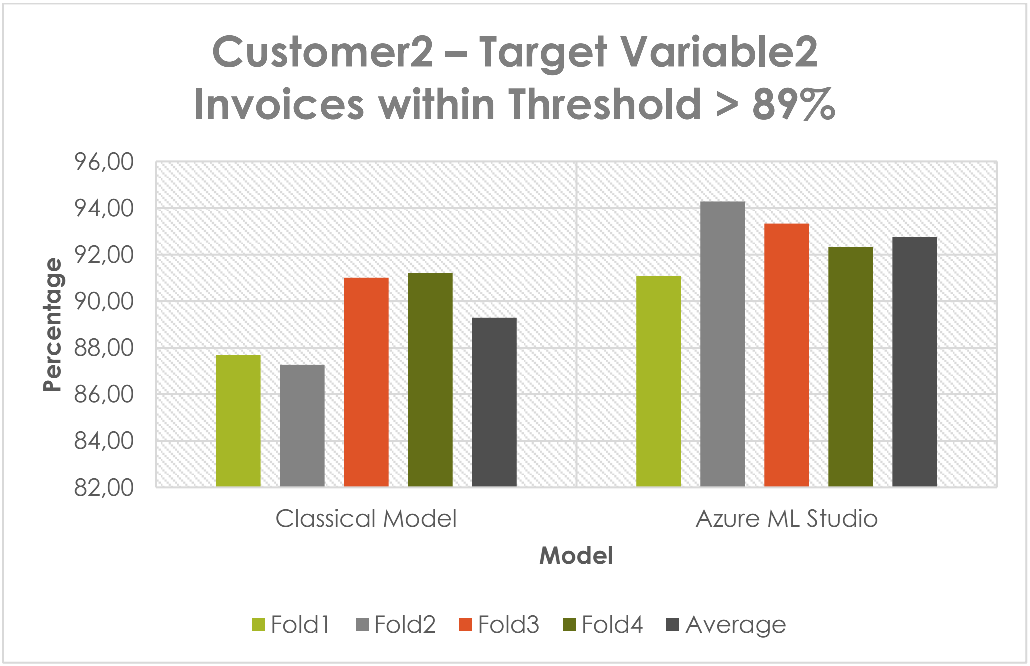

*Figure 5.26: Customer2 Target Variable2 Invoices that are predicted with over 89% Confidence*

In the Figure 5.26 above, the fold wise comparison for second metric could be observed. The devised classical model on average could predict 89.29% invoices with more than

89% confidence. Whereas, ML studio on the other hand could predict 92.75% invoices with more than 89% confidence.

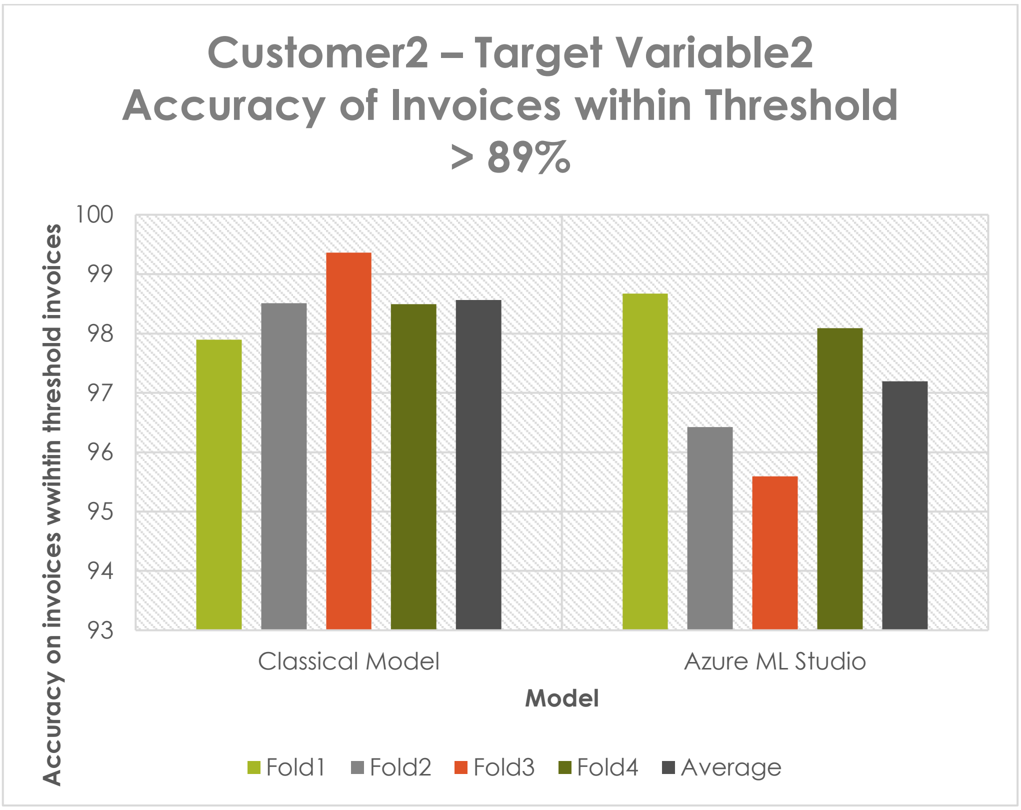

*Figure 5.27: Customer2 Target Variable2-Accuracy of Invoices that are predicted with >89 % Confidence*

Then the performance of the algorithm under study was assessed in comparison, with ML studio for 3rd Business metric. The investigated classical model could accurately predict invoices with 98.56%, however ML studio could predict invoices with 97.19% accuracy as can be seen in Figure 5.27.

Further on, the performance of our classical model with ML Studio, for a range of confidence threshold rather than the Business set threshold was analysed.

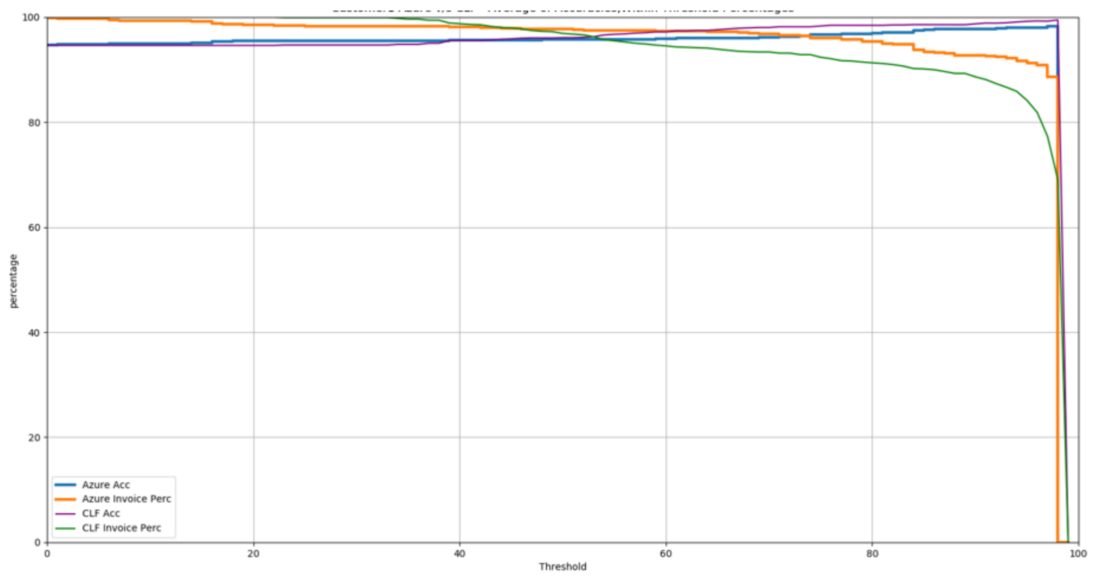

*Figure 5.28: Azure ML Studio v/s Classical model for various Confidence Thresholds.*

In the Figure 5.28, the Classical Model shown in purple and Azure ML Studio in blue can be seen. They appear very close, as the threshold of confidence is varied. It can be observed that as the confidence threshold is varied between 50%-60%, the accuracy of the algorithm under study improves. The number of invoice prediction is slightly low as compared to the ML studio when the confidence threshold is >50%.

## 5.4 Multi-Task Learning:

The accuracies concluded on the Customer 1 are deciphered below in Table 5.9, when the Multi-task learning algorithm was applied for both Target variables. (*)mark indicates that the values written were selected by the Grid Search of the Algorithm and were not hard - coded.

### 5.4.1 Customer1:

| ***Target Variable1, Target Variable2*** | | | | | |
|---|---|---|---|---|---|
| | **Fold1** | **Fold 2** | **Fold 3** | **Fold 4** | **Average** |
| **Target Variable1** | | | | | |
| *Accuracy% on Training Set* | 94,49 | 94,97 | 95,15 | 94,34 | |
| *Accuracy% on Dev Test Set* | 73,76 | 74,08 | 71,59 | 75,69 | 73,77897 |
| *Invoices% predicted with confidence >89%* | 55,34 | 54,81 | 58,89 | 57,43 | 56,6172 |
| *Accuracy% on Invoices predicted with >89% confidence* | 95,51 | 96,06 | 93,36 | 94,29 | 94,80297 |
| *Number of invoices predicted with confidence >89%* | 1046 | 1015 | 798 | 893 | 938 |
| *Number of invoices predicted with confidence >89%, correctly* | 999 | 975 | 745 | 842 | 890.25 |
| *batch_size** | 100 | 50 | 50 | 100 | --- |
| *dropout_rate** | 0.2 | 0.1 | 0.2 | 0.2 | -- |
| *optimizer:* | Adam | Adam | Adam | Adam | -- |
| *Epochs ran:** | 40-50 | 35 | 35 | 50-60 | -- |
| **Target Variable2** | | | | | |
| *Accuracy% on Training Set* | 99.574 | 99.399 | 99.5044 | 99.553 | 99.506 |
| *Accuracy% on Dev Test Set* | 95,18 | 97,52 | 95,87 | 97,17 | 96,435 |
| *Invoices% predicted with confidence >89%* | 93,39 | 94,22 | 93,14 | 92,15 | 93,2247 |
| *Accuracy% on Invoices predicted with >89% confidence* | 99,04 | 99,82 | 98,65 | 99,65 | 99,28995 |
| *Number of invoices predicted with confidence >89%* | 1745 | 1765 | 1262 | 1433 | 1551.25 |
| *Number of invoices predicted with confidence >89%, correctly* | 1742 | 1748 | 1245 | 1428 | 1540.75 |

Table 5.9: Multi-Task learning performance metrics

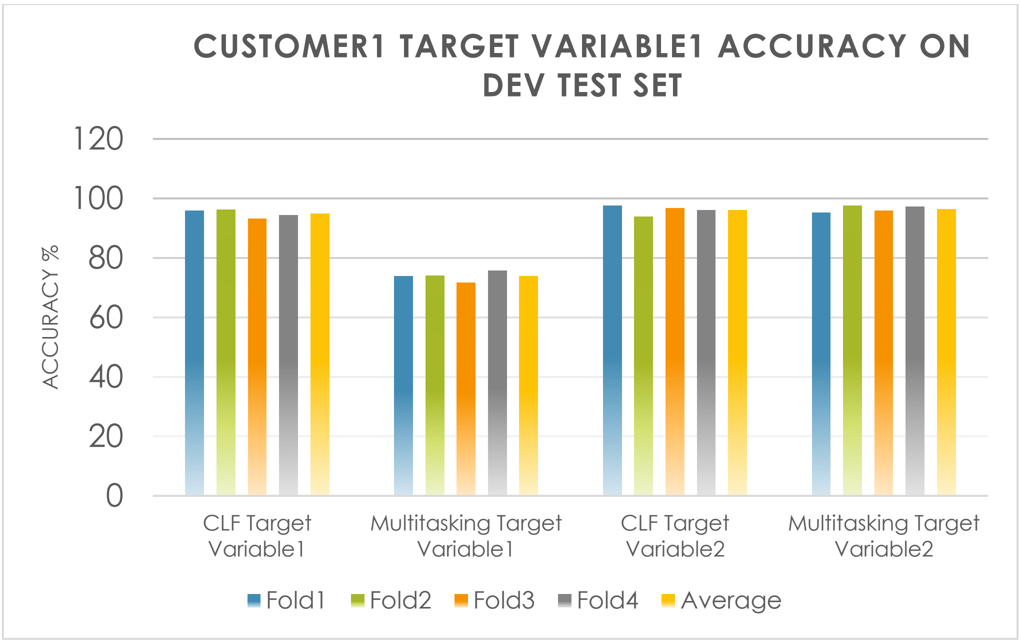

*Figure 5.29: Customer1 Target Variable1 Accuracy on Dev Test Set*

As indicated in Figure 5.29 above, the Target Variable1 accuracy didn't improve with Multi-task learning. However, the accuracy of the Target Variable 2 got slightly better in the case of multi-task learning.

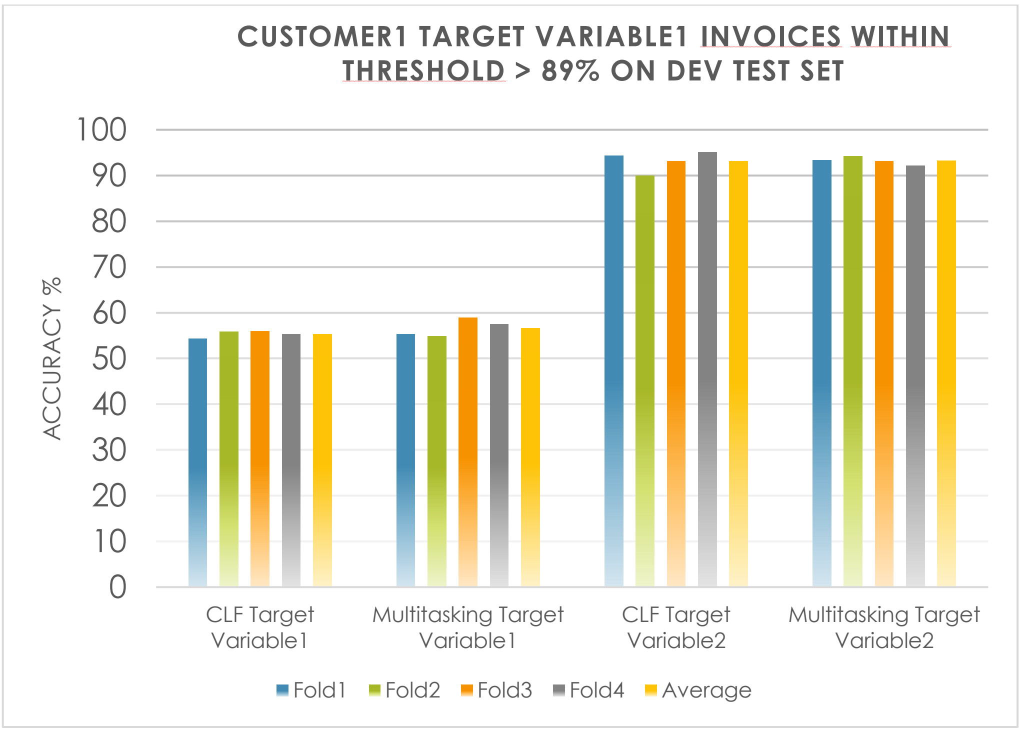

*Figure 5.30: Customer1 Target Variable1 Invoices predicted with Confidence >=90%*

In the Figure 5.30, on average 56.61% invoices of Target Variable1 and 93.22% invoices for Target Variable2 are predicted >89% Confidence. Whereas in the classical model, 55.34 % invoices for Target Variable1 and 89.02% invoices for Target Variable2 were predicted with >89% confidence. This indicates, multi-task learning approach gave improved results in this case.

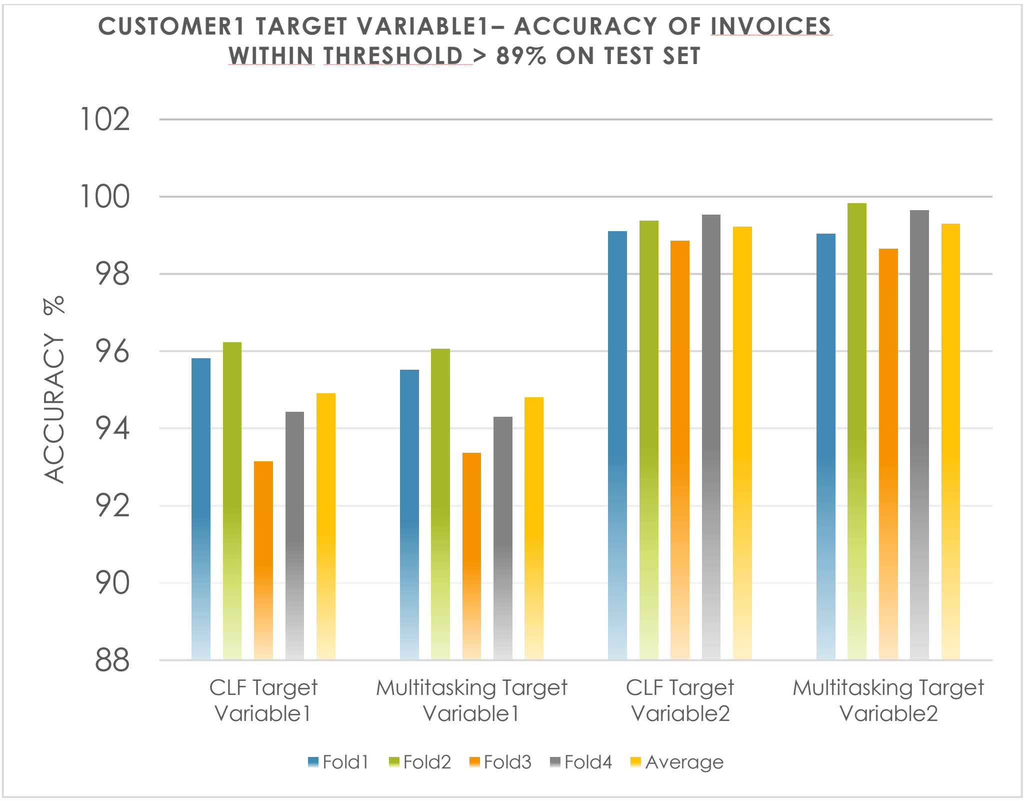

*Figure 5.31: Customer1 Target Variable1 Accuracy of Within Threshold Invoices*

The accuracy for those invoices that were predicted with over 89% confidence were compared and presented in Figure 5.31. In case of multi-task learning, on average 94.80% for Target Variable1 and 99.28% of Target Variable2 was the accuracy which is slightly equivalent to the 94.89% for Target Variable1 and 99.21% for Target Variable2 that was achieved in classical model.

### 5.4.2 Customer2:

| **CUSTOMER2 (MULTI – TASK LEARNING)<br>TARGET VARIABLE1, TARGET VARIABLE2** | | | | | |
|---|---|---|---|---|---|
| | ***Fold1*** | ***Fold 2*** | ***Fold 3*** | ***Fold 4*** | ***Average*** |
| **TARGET VARIABLE1** | | | | | |
| *ACCURACY% ON TRAINING SET* | 94,04 | 93,94 | 94,17 | 94,15 | 94.07 |
| *ACCURACY% ON DEV TEST SET* | 71,66 | 74,47 | 73,05 | 73,20 | 73,093 |
| *INVOICES% PREDICTED WITH CONFIDENCE >89%* | 53,19 | 54,42 | 54,50 | 54,29 | 54,099 |
| *ACCURACY% ON INVOICES PREDICTED WITH >89% CONFIDENCE* | 91,42 | 92,66 | 91,18 | 91,93 | 91,794 |
| *NUMBER OF INVOICES PREDICTED WITH CONFIDENCE >89%* | 3484 | 3472 | 3571 | 3642 | 3542 |
| *NUMBER OF INVOICES PREDICTED WITH CONFIDENCE >89%, CORRECTLY* | 3185 | 3217 | 3256 | 3348 | 3252 |
| **TARGET VARIABLE2** | | | | | |
| *ACCURACY% ON TRAINING SET* | 99,94 | 99,92 | 99,923 | 99,93 | 99,92 |
| *ACCURACY% ON DEV TEST SET* | 98,99 | 99,09 | 99,08 | 99,21 | 99,092 |
| *INVOICES% PREDICTED WITH CONFIDENCE >89%* | 98,45 | 98,29 | 98,59 | 98,49 | 98,456 |
| *ACCURACY% ON INVOICES PREDICTED WITH >89% CONFIDENCE* | 99,73 | 99,63 | 99,55 | 99,69 | 99,65 |
| *NUMBER OF INVOICES PREDICTED WITH CONFIDENCE >89%* | 6449 | 6271 | 6460 | 6608 | 6447 |
| *NUMBER OF INVOICES PREDICTED WITH CONFIDENCE >89%, CORRECTLY* | 6432 | 6248 | 6431 | 6588 | 6425 |

*Table 5.10: Multi-task Learning – Customer2*

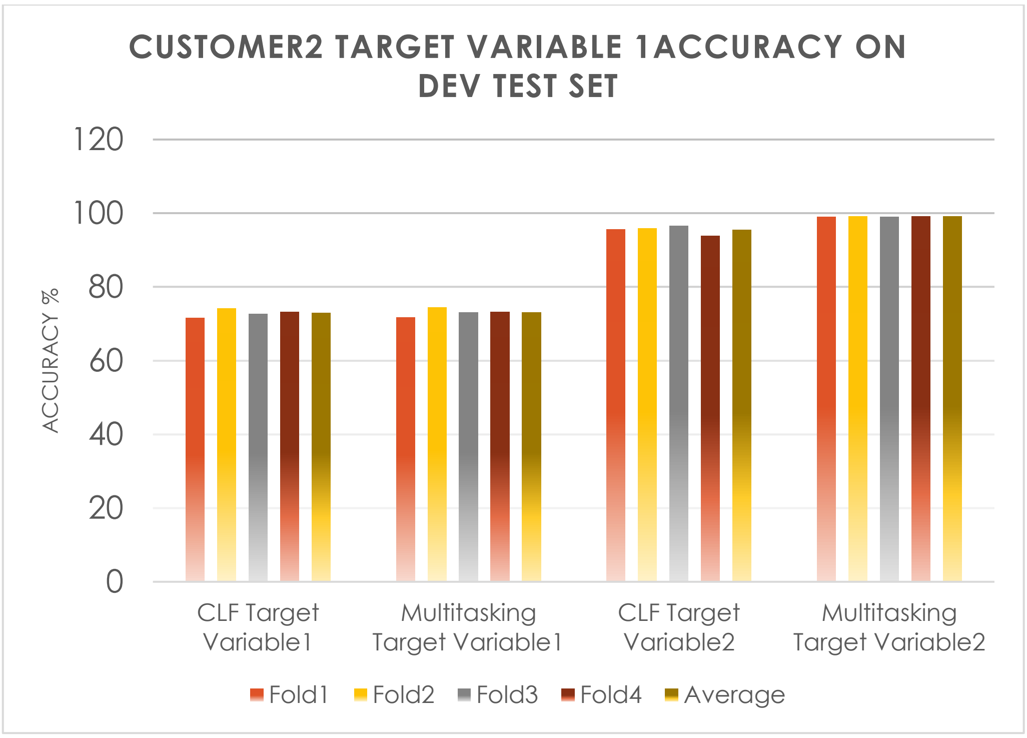

*Figure 5.32: Customer 2 Target Variable1: Accuracy on Dev Test Set*

In the Figure 5.32 above, the accuracy was tested on DevTest dataset. In case of multi-task learning, on average accuracy of 73.09% for Target Variable1 and 99.09% for Target Variable2 were achieved. These values were more than the average values, 72.90% Target Variable1 and 95.48% for Target Variable2.

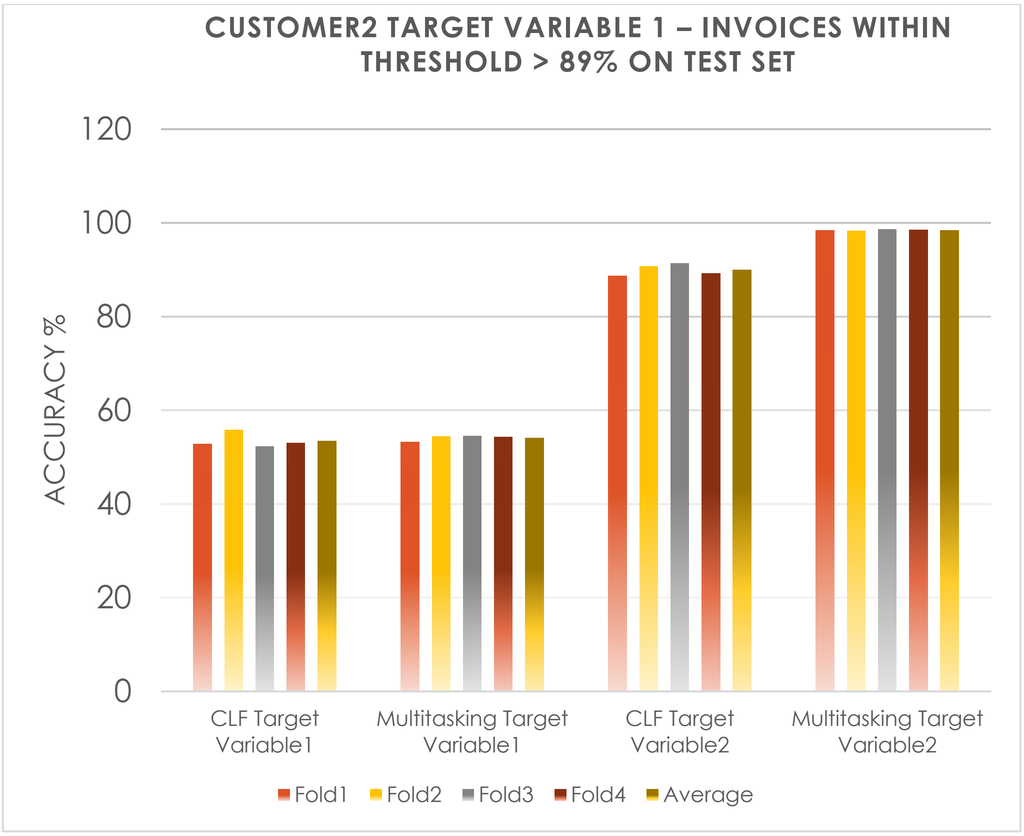

*Figure 5.33: Invoices predicted with >89% Confidence*

In this Figure 5.33, the second metric for Customer2 was analysed. By using multi-task learning, for Target Variable2 more invoices were predicted with over 89% confidence, than without multi-task learning.

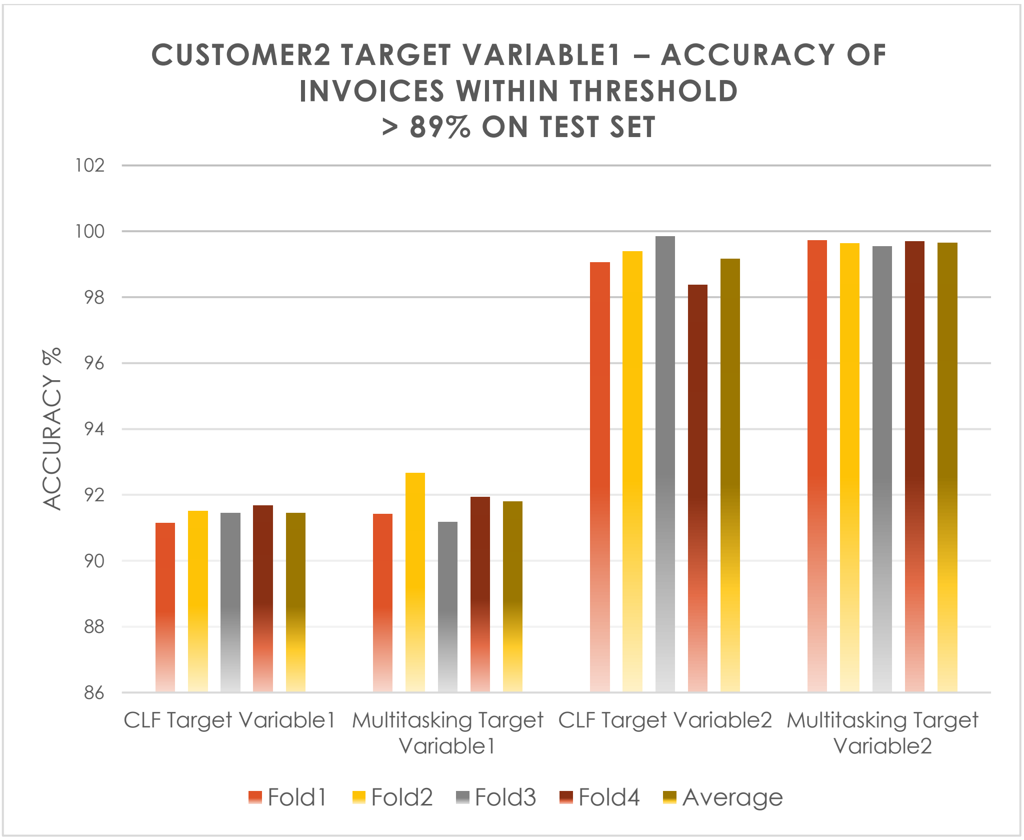

*Figure 5.34: Customer2 Target Variable1 Accuracy of Within Threshold Invoices*

In the Figure 5.34 above, the accuracy of those invoices that were predicted with over 89% confidence were analysed. For Target Variable1, 91.79% was predicted when multi-task learning was used which is slightly better than the figure 91.44% that was obtained in classical model under investigation.

For Target Variable2, 99.16% accuracy was obtained without multi-task learning and 99.65% was obtained by using multi-task learning. This multi-task learning seems to help somewhat for both variables on customer 2.

## 5.5 Continual Learning:

Data set from 3 months (October, November and December) was utilized in the study. It was divided into 2 training datasets which are October and November and model prediction accuracy was tested on the dataset of December. The accuracy metrics were kept the same since not only the business goal was changed but also the machine learning approach was varied. The strategy of Continual learning on Customer2 dataset was not involved as it was beyond the scope of this project.

The classical model under investigation was trained from scratch on the dataset of the most recent month i.e. November and its prediction accuracy were tested for the next month December. An already trained model was then trained on October dataset, on the month of November. The prediction accuracy for the second approach was compared with the prediction accuracy of the first one. The results obtained for Customer1 are shown below for Target Variable1 in Figure 5.35& Table 5.11 and for Target Variable 2 in Figure 5.36 & Table 5.12.

| ***Customer1 Target Variable1*** | ***Trained on October, tested Dec*** | ***Trained on November, tested Dec*** | ***Trained Model of October, trained on November, test Dec*** |
|---|---|---|---|
| **Accuracy% on Training Set** | **3785** | **3228** | **3228** |
| **Accuracy% on Dev Test Set** | **1877** | **1877** | **1877** |
| **Invoices% predicted with confidence >89%** | **1136** | **969** | **969** |
| **Accuracy% on Invoices predicted with >89% confidence** | **95.847489** | **94.864** | **85.081** |
| **Number of invoices predicted with confidence >89%** | **62.1204** | **69.6856** | **74.9067** |
| **Number of invoices predicted with confidence >89%, correctly** | **42.9941** | **47.94885** | **56.2599** |
| **Accuracy% on Training Set** | **91.5737** | **93.555** | **92.23484** |
| **Accuracy% on Dev Test Set** | **807** | **900** | **1056** |
| **Invoices% predicted with confidence >89%** | **739** | **842** | **974** |

*Table 5.11: Continual Learning Customer1*

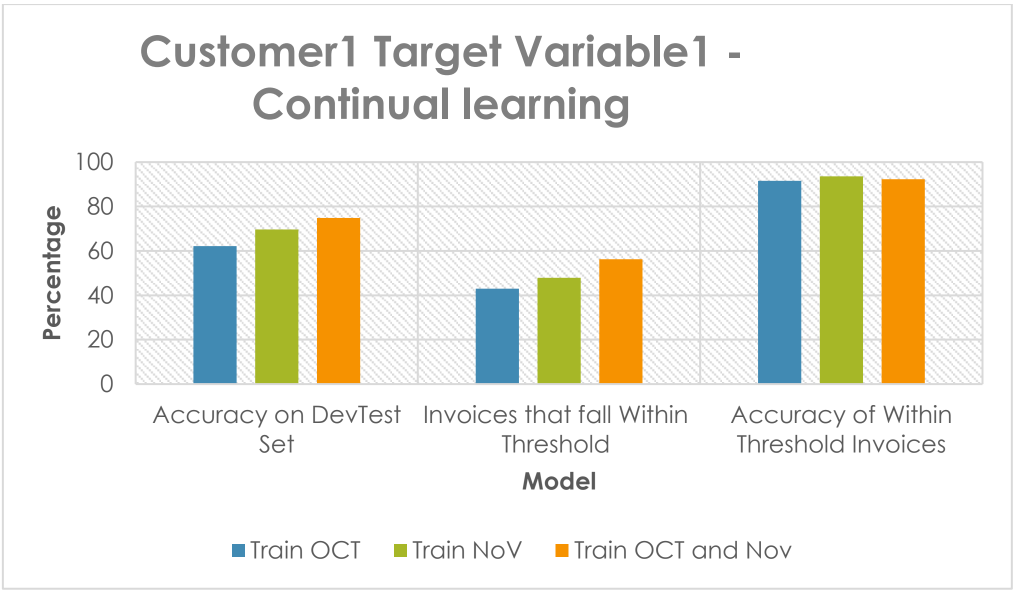

*Figure 5.3510: Customer1 Target Variable1 Continual Learning*

In Figure 5.35, the bar for resulting accuracy when training on Oct and Nov data is observed as high as compared to the bar for only training on Oct data or the bar for only Training on Nov Data. This shows that the predictive algorithm performed better when trained only on most recent (Nov) data versus training only on the earlier (Oct) data. Furthermore, the algorithm performed much better when it was trained already on known classes of data from Oct, and then a 2nd time of training is done on Nov data.

| **Customer1, Target variable2** | **Trained on October, Tested Dec** | **Trained on November, Tested Dec** | **Trained model of October, Trained again on November data set** |
|---|---|---|---|
| ROWS IN TRAINING SET | 3785 | 3228 | 3228 |
| ROWS IN DEV TEST SET | 1877 | 1877 | 1877 |
| ACCURACY% ON TRAINING SET | 93,733 | 94,599 | 95,750 |
| ACCURACY% ON DEV TEST SET | 87,320 | 90,889 | 90,942 |
| INVOICES% PREDICTED WITH CONFIDENCE >89% | 85,881 | 83,111 | 86,680 |
| ACCURACY% ON INVOICES PREDICTED WITH >89% CONFIDENCE | 96,588 | 98,589 | 99,569 |
| NUMBER OF INVOICES PREDICTED WITH CONFIDENCE >89% | 1612 | 1560 | 1627 |
| NUMBER OF INVOICES PREDICTED WITH CONFIDENCE >89%, CORRECTLY | 1557 | 1538 | 1620 |

*Table 5.12: Continual Learning Customer1 Target Variable2*

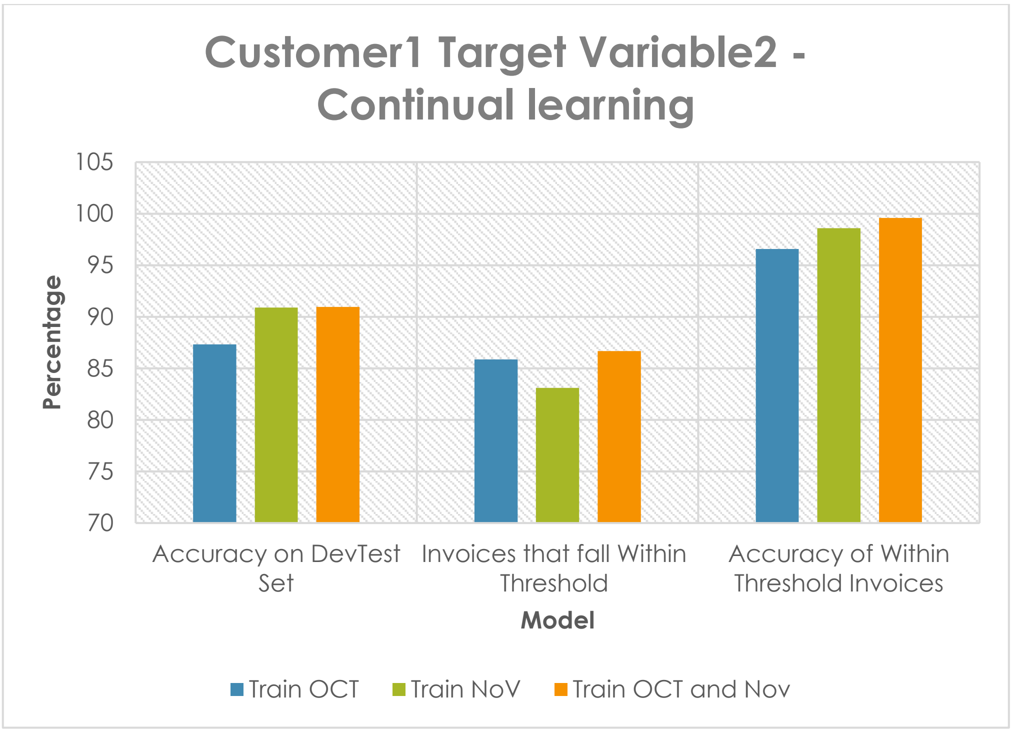

*Figure 5.3611: Continual Learning - Customer1 Target Variable2*

In Figure 5.36, the approach for Continual learning for Target Variable2 was tested. The algorithm showed the same behavior as we saw in the case of Target Variable1. All the three metrics for accuracy were high in case of Continual Learning scenario. This answered the question that Continual Learning is more effective in the case under study for Invoice postings. Due to time constraints, continual learning was not analyzed for Customer2.

Table 5.13 and Table 5.14 in the next pages depicts the average of metrics for 4 folds, concerning each of the machine learning technique used.

| | Classical model (General Comparison) | | | | Multi-Task Learning | | | | *Continual learning | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Customer1 | | Customer2 | | Customer1 | | Customer2 | | Customer1 | |
| | Target Variable 1 | Target Variable 2 | Target Variable 1 | Target Variable 2 | Target Variable 1 | Target Variable 2 | Target Variable 1 | Target Variable 2 | Target Variable 1 | Target Variable 2 |
| Accuracy% on Training Set | 94.80 | 99.36 | 92.61 | 99.75 | 94.73 | 99.506 | 94.07 | 99.92 | 85.08 | 95,75 |
| Accuracy% on Dev Test Set | 73.73 | 96.02 | 72.90 | 95.48 | 73.77 | 96.43 | 73.09 | 99.09 | 74.90 | 90.94 |
| Invoices% predicted with confidence >89% | 55.34 | 89.012 | 53.47 | 90 | 56.61 | 93.22 | 54.09 | 98.45 | 56.25 | 86.68 |
| Accuracy% on Invoices predicted | 94.90 | 99.213 | 91.44 | 99.160 | 94.80 | 99.28 | 91.79 | 99.65 | 92.23 | 99.56 |
| Number of invoices predicted with confidence >89% | 919.5 | 947.5 | 3500.24 | 627.5 | 938 | 1551.25 | 3542 | 6447 | 1056 | 1627 |
| Number of invoices predicted with confidence >89%. | 874 | 942.25 | 3200.96 | 622.5 | 890.25 | 1540.75 | 3252 | 6425 | 974 | 1620 |

*Table 5.13 Summary of averages of results based on various approaches*

| | Classical model (Exact Data per Fold Comparison) | | | | ML Studio (Exact data per fold comparison) | | | |
|---|---|---|---|---|---|---|---|---|
| | Customer1 | | Customer2 | | Customer1 | | Customer2 | |
| | Target Variable 1 | Target Variable 2 | Target Variable 1 | Target Variable 2 | Target Variable 1 | Target Variable 2 | Target Variable 1 | Target Variable 2 |
| Accuracy% on Training Set | 94.38 | 99.43 | 93.76 | 99.59 | 96.54 | 97.34 | 95.71 | 96.41 |
| Accuracy% on Dev Test Set | 74.41 | 95.52 | 73.81 | 94.62 | 73.6 | 94.72 | 72.82 | 94.67 |
| Invoices% predicted with confidence >89% | 59.77 | 90.41 | 55.46 | 89.29 | 62.99 | 92.78 | 72.25 | 92.75 |
| Accuracy% on Invoices predicted with >89% confidence | 94.58 | 99.15 | 91.77 | 98.56 | 92.85 | 98.57 | 85.62 | 97.19 |
| Number of invoices predicted with confidence >89% | 514 | 952 | 1761 | 316.75 | 541.08 | 976 | 2293 | 328.33 |
| Number of invoices predicted with confidence >89%, | 482 | 944 | 1616 | 312.25 | 502 | 962,08 | 1964 | 319.10 |

*Table 5.14 Summary of averages of results based on various approaches*

# 6 Conclusion:

In this project, possible machine learning approaches that could be applied in the domain of finance for the data of Invoices of two different Customers were carried out. Lately, the applications and henceforth the requirement for the implication of machine learning approaches is soaring in all fields of sciences. As per the field of finance, the most popular use cases explored so far are; fraud detection, loan management and insurance. The case that we targeted in this study was also from the domain of finance, however it was related to receivables. The scenario of receivables in our case of study is slightly different than the ones investigated generally. We analysed and conducted the study using the data from two (anonymous) customers and investigated the application of a classical machine learning method to determine their destination to be posted. We developed the predictive algorithm and compared its performance with Azure ML Studio already in use.

Our first approach mainly focussed on implementing a predictive algorithm that could give at least similar accuracies as achieved from Azure ML Studio. We used open source library which is Keras with TensorFlow at the backend. We then compared the performance of the algorithm with ML studio on the basis of three different business metrics. Later, the performance of our model was studied not only in the general case (where the algorithm does the splitting of dataset into training and test sets on its own) and also on the exact same data for training and testing on each fold, that was used by the Azure ML studio.

In the second approach, we implemented Multi-Task learning in our neural network. This not only helped us in identifying possible dependencies among the two target variables but also led us to possible improvement in the prediction accuracies. Based on our results, we concluded that since multi-task learning seems to improve performance, it suggests that Target Variable2 has dependency on Target Variable1. On the basis of our findings; we would suggest the business to look for case analysis that would cater the dependency of Target Variable2 on Target Variable1.

To make a clear understanding of the various approaches, we applied Continual learning as the third approach. This resulted in supporting our findings that a neural network that has learnt on the data of last two months could perform better than the model that was only trained on last month. The resulting observations seem very logical as in the former case; it utilizes the data leaned from more number of cases which is a key to better performance of a predictive algorithm.

Based on our results and prediction accuracies, we could say that overall our algorithm gave similar performance as could be achieved from Azure ML studio for our data of invoice postings. We not only evaluated the performance of our algorithm in general but also compared its performance on exactly same data that was used for training and testing in Azure in each of its 4 folds.

We successfully implemented and tested three fundamental approaches of machine learning on invoice data as discussed. However, as mentioned earlier, there are plenty of machine learning strategies available which could still be verified. This study implies that in the future, after further investing the data properties and business aspects, several other machine learning strategies can be checked for better prediction accuracies. The applications of machine learning based on utilizing neural networks in the field of finance holds tremendous opportunities. It may hold the keys to open the doors of wonders and is on the rise with the rapid advancement in technology. In addition to the current scenario of invoice postings, it can render improved customer experience, fraud detection, insurance and ease in the mode of payments etc. These widespread use cases of machine learning in sector of finance, can benefit from more variants of machine learning strategies.

## List of References:

(1) Sarlin P, Björk K. Machine learning in finance—Guest editorial. Neurocomputing 2017;264:1.

(2) Kim M, Kang D. Ensemble with neural networks for bankruptcy prediction. Expert Syst Appl 2010;37(4):3373-3379.

(3) MARKOFF J. Scientists See Promise in Deep-Learning Programs, NY Times. http://nyti.ms/sgcVec 2012.

(4) Zhang G, Patuwo BE, Hu MY. Forecasting with artificial neural networks: The state of the art. International journal of forecasting. 1998 Mar 1;14(1):35-62.

(5) Bebis G. Introduction to Artificial Neural Networks. Available at: https://www.cse.unr.edu/~bebis/MathMethods/NNs/lecture.pdf. Accessed April 22, 2018.

(6) Penpece D, Elma OE. Predicting Sales Revenue by Using Artificial Neural Network in Grocery Retailing Industry: A Case Study in Turkey. International Journal of Trade, Economics and Finance 2014;5(5):435-440.

(7) Chris W. Neural networks. Available at: https://www.explainthatstuff.com/introduction-to-neural-networks.html. Accessed April 15, 2019.

(8) Fuangkhon P. An incremental learning preprocessor for feed-forward neural network. Artificial Intelligence Review. 2014 Feb 1;41(2):183-210.

(9) Kwok TY, Yeung DY. Objective functions for training new hidden units in constructive neural networks. IEEE Transactions on neural networks. 1997 Sep;8(5):1131-48.

(10) Samatin Njikam A, Zhao H. A novel activation function for multilayer feed-forward neural networks. Appl Intell 2016 Jul;45(1):75-82.

(11) Zhou H, Li Z. Deep networks with non-static activation function. Multimedia Tools and Applications. 2019 Jan 1;78(1):197-211.

(12) Becker D. Rectified Linear Units (ReLU) in Deep Learning . 2018; Available at: https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning. Accessed April 29, 2018.

(13) Hyde D. TANH(x). Available at: http://www.dplot.com/fct_tanh.htm. Accessed April 27, 2018.

(14) Jain AK, Mao J, Mohiuddin KM. Artificial neural networks: A tutorial. Computer. 1996 Mar;29(3):31-44.

(15) Battiti R. First-and second-order methods for learning: between steepest descent and Newton's method. Neural computation. 1992 Mar;4(2):141-66.

(16) King AJ, Wallace SW. Modeling with stochastic programming. Springer Science & Business Media; 2012 Jun 19.

(17) Ferentinos KP. Biological engineering applications of feedforward neural networks designed and parameterized by genetic algorithms. Neural Networks 2005;18(7):934-950.

(18) Dorsey RE, Johnson JD, Mayer WJ. A genetic algorithm for the training of feedforward neural networks. Advances in artificial intelligence in economics, finance and management. 1994;1:93-111.

(19)Kim D, Kim H, Chung D. A modified genetic algorithm for fast training neural networks. InInternational Symposium on Neural Networks 2005 May 30 (pp. 660-665). Springer, Berlin, Heidelberg.

(20) Ruder S. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747. 2016 Sep 15.

(21) Two problems with back propagation and other steepest descent learning procedures for networks. Proceedings of the Eighth Annual Conference of the Cognitive Science Society, 1986; 1986.

(22) Mammadov M, Tas E, Omay RE. Accelerating backpropagation using effective parameters at each step and an experimental evaluation. Journal of Statistical Computation and Simulation 2008;78(11):1055-1064.

(23) Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. 2014.

(24) Hu Peiguang. Predicting and Improving Invoice-to-Cash Collection Through Machine LearningMassachusetts Institute of Technology; 2015.

(25) Bartoli A, Davanzo G, Medvet E, Sorio E. Improving features extraction for supervised invoice classification. InProceedings of the 10th IASTED International Conference 2010 (Vol. 674, No. 040, p. 401).

(26) Tarawneh AS, Hassanat AB, Chetverikov D, Lendak I, Verma C. Invoice Classification Using Deep Features and Machine Learning Techniques. In2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT) 2019 Apr 9 (pp. 855-859). IEEE.

(27) Fadlalla A, Lin CH. An analysis of the applications of neural networks in finance. Interfaces. 2001 Aug;31(4):112-22.

(28) Wong BK, Selvi Y. Neural network applications in finance: A review and analysis of literature (1990–1996). Information & Management 1998;34(3):129-139.

(29) Nikolopoulos C, Fellrath P. A hybrid expert system for investment advising. Expert Systems 1994 Nov;11(4):245-250.

(30) Shen A, Tong R, Deng Y. Application of classification models on credit card fraud detection. In2007 International conference on service systems and service management 2007 Jun 9 (pp. 1-4). IEEE.

(31) Kamijo KI, Tanigawa T. Stock price pattern recognition-a recurrent neural network approach. In1990 IJCNN International Joint Conference on Neural Networks 1990 Jun 17 (pp. 215-221). IEEE.

(32) Zhang Y, Wu K, Du B, Zhang L, Hu X. Hyperspectral Target Detection via Adaptive Joint Sparse Representation and Multi-Task Learning with Locality Information. Remote Sensing 2017 May 14,;9(5):482.

(33) Ghosn J, Bengio Y. Multi-task learning for stock selection. InAdvances in neural information processing systems 1997 (pp. 946-952).

(34) Di Persio LU, Honchar OL. Multitask machine learning for financial forecasting. International Journal of Circuits, Systems and Signal Processing. 2018;12:444-51.

(35) Awasthi A, Sarawagi S. Continual Learning with Neural Networks: A Review. InProceedings of the ACM India Joint International Conference on Data Science and Management of Data 2019 Jan 3 (pp. 362-365).

(36) McCloskey M, Cohen NJ. Catastrophic interference in connectionist networks: The sequential learning problem. InPsychology of learning and motivation 1989 Jan 1 (Vol. 24, pp. 109-165). Academic Press.

(37) Van der Heijden GJ, Donders AR, Stijnen T, Moons KG. Imputation of missing values is superior to complete case analysis and the missing-indicator method in multivariable diagnostic research: a clinical example. Journal of clinical epidemiology. 2006 Oct 1;59(10):1102-9.

(38) Patro S, Sahu KK. Normalization: A preprocessing stage. arXiv preprint arXiv:1503.06462. 2015 Mar 19.

(39) Witten IH, Frank E. Data mining: practical machine learning tools and techniques with Java implementations. Acm Sigmod Record. 2002 Mar 1;31(1):76-7.

(40) Man Z, Lee K, Wang D, Cao Z, Miao C. A new robust training algorithm for a class of single-hidden layer feedforward neural networks. Neurocomputing 2011;74(16):2491-2501.

(41) Caruana R. Multitask learning. Machine learning. 1997 Jul 1;28(1):41-75.

(42) Ruder S. An overview of multi-task learning in deep neural networks. arXiv preprint arXiv:1706.05098. 2017 Jun 15.

(43) Bingel J, Søgaard A. Identifying beneficial task relations for multi-task learning in deep neural networks. arXiv preprint arXiv:1702.08303. 2017 Feb 27.

(44) Meyerson E, Miikkulainen R. Beyond shared hierarchies: Deep multitask learning through soft layer ordering. arXiv preprint arXiv:1711.00108. 2017 Oct 31.

(45) Thrun S, Pratt L, editors. Learning to learn. Springer Science & Business Media; 2012 Dec 6.

(46) Evgeniou T, Pontil M. Regularized multi--task learning. InProceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining 2004 Aug 22 (pp. 109-117).

(47) Zhang J, Ghahramani Z, Yang Y. Learning multiple related tasks using latent independent component analysis. InAdvances in neural information processing systems 2006 (pp. 1585-1592).

(48) Evgeniou T, Micchelli CA, Pontil M. Learning multiple tasks with kernel methods. Journal of Machine Learning Research 2005;6:615-637.

(49) Collobert R, Weston J. A unified architecture for natural language processing: Deep neural networks with multitask learning. InProceedings of the 25th international conference on Machine learning 2008 Jul 5 (pp. 160-167).

(50) Bakker B, Heskes T. Task clustering and gating for bayesian multitask learning. Journal of Machine Learning Research 2004;4(1):83-99.

(51) Waibel A, Sawai H, Shikano K. Modularity and scaling in large phonemic neural networks. IEEE Transactions on Acoustics, Speech, and Signal Processing 1989;37(12):1888-1898.

(52) Allenby GM, Rossi PE. Marketing models of consumer heterogeneity. Journal of econometrics. 1998 Nov 26;89(1-2):57-78.

(53) Bitvai Z, Cohn T. Day trading profit maximization with multi-task learning and technical analysis. Machine Learning. 2015 Oct 1;101(1-3):187-209..

(54) Baxter J. A Bayesian/Information Theoretic Model of Learning to Learn via Multiple Task Sampling. Mach Learning 1997;28(1):7-39.

(55) Ruder S, Bingel J, Augenstein I, Søgaard A. Learning what to share between loosely related tasks. 2017.

(56) Lawrence ND, Platt JC. Learning to learn with the informative vector machine. InProceedings of the twenty-first international conference on Machine learning 2004 Jul 4 (p. 65).

(57) Chapelle O, Shivaswamy P, Vadrevu S, Weinberger K, Zhang Y, Tseng B. Boosted multi-task learning. Machine Learning. 2011 Oct;85(1-2):149-173

(58) Chen J, Zhou J, Ye J. Integrating low-rank and group-sparse structures for robust multi-task learning. InProceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining 2011 Aug 21 (pp. 42-50).

(59) Parisi GI, Kemker R, Part JL, Kanan C, Wermter S. Continual lifelong learning with neural networks: A review. Neural Networks. 2019 Feb 6.

(60) McClelland JL, McNaughton BL, O'Reilly RC. Why There Are Complementary Learning Systems in the Hippocampus and Neocortex: Insights From the Successes and Failures of Connectionist Models of Learning and Memory. Psychological Review 1995;102(3):419-457.

(61) McClelland JL, McNaughton BL, O'Reilly RC. Why There Are Complementary Learning Systems in the Hippocampus and Neocortex: Insights From the Successes and Failures of Connectionist Models of Learning and Memory. Psychol Rev 1995;102(3):419-457.

(62) French RM. Catastrophic forgetting in connectionist networks. Trends in Cognitive Sciences 1999;3(4):128-135

(63) Legg S, Hutter M. Universal Intelligence: A Definition of Machine Intelligence. Minds and Machines 2007;17(4):391-444.

(64) Mankowitz DJ, Žídek A, Barreto A, Horgan D, Hessel M, Quan J, Oh J, van Hasselt H, Silver D, Schaul T. Unicorn: Continual learning with a universal, off-policy agent. arXiv preprint arXiv:1802.08294. 2018 Feb 22

(65) Käding C, Rodner E, Freytag A, Denzler J. Fine-tuning deep neural networks in continuous learning scenarios. InAsian Conference on Computer Vision 2016 Nov 20 (pp. 588-605). Springer, Cham