WestVirginiaUniversity
THE RESEARCH REPOSITORY @ WVU

Graduate Theses, Dissertations, and Problem Reports

2023

AI-Driven Security Constrained Unit Commitment Using Predictive Modeling And Eigen Decomposition

Talha Iqbal
West Virginia University, ti0001@mix.wvu.edu

Follow this and additional works at: https://researchrepository.wvu.edu/etd

Part of the Electrical and Electronics Commons, and the Power and Energy Commons

Recommended Citation
Iqbal, Talha, "AI-Driven Security Constrained Unit Commitment Using Predictive Modeling And Eigen Decomposition" (2023). *Graduate Theses, Dissertations, and Problem Reports*. 12109.
https://researchrepository.wvu.edu/etd/12109

This Dissertation is protected by copyright and/or related rights. It has been brought to you by the The Research Repository @ WVU with permission from the rights-holder(s). You are free to use this Dissertation in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you must obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/ or on the work itself. This Dissertation has been accepted for inclusion in WVU Graduate Theses, Dissertations, and Problem Reports collection by an authorized administrator of The Research Repository @ WVU. For more information, please contact researchrepository@mail.wvu.edu.

AI Driven Security Constrained Unit Commitment Using Predictive Modeling and Eigen Decomposition

Talha Iqbal

Dissertation submitted
to the Benjamin M. Statler College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in
Electrical Engineering

Ali Feliachi, Ph.D., Chair
Muhammad Choudhry, Ph.D.
Natalia Schmid, Ph.D.
Vinod Kulathumani, Ph.D.
Hong Jian Lai, Ph.D.

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia

2023

Keywords: Predictive Modeling, Data Analysis, Unit Commitment, Generation Scheduling, Electricity Market, Optimization, Eigen Decomposition, Generation Shift Factors

Copyright 2023 Talha Iqbal

ABSTRACT

AI Driven Security Constrained Unit Commitment Using Predictive Modeling and Eigen Decomposition

Talha Iqbal

Security Constrained Unit Commitment (SC-UC) is a complex large scale mix integer constrained optimization problem solved by Independent System Operators (ISOs) in the daily planning of the electricity markets. After receiving offers and bids, ISOs have only few hours to clear the day-ahead electricity market. It requires a lot of computational effort and a reasonable time to solve a large-scale SC-UC problem. However, exploiting the fact that a UC problem is solved several times a day with only minor changes in the system data, the computational effort can be reduced by learning from the historical data and identifying the patterns in the historical data using data mining techniques.

In this thesis, two data driven approaches based on predictive modeling techniques are proposed to solve a SC-UC problem in a day ahead electricity market which can be used as alternative backup methods for solving a SC-UC problem. In the first approach (*Algo-1*), the SC-UC is partially modeled using predictive modeling techniques to enhance the computational speed of the problem, while in the second approach (*Algo-2*), the optimization problem is completely replaced by data driven predictive models to further enhance the computational efficiency, however, at the cost of some optimality loss. The proposed approaches are validated through numerical simulations on different IEEE case studies to demonstrate and study the effectiveness of the developed approaches. The results obtained from the proposed approaches are compared with those obtained using commercial optimization solvers e.g., IBM CPLEX MIQP and GUROBI MIQP solvers.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER

INTRODUCTION

1

1.1 Unit Commitment

In an electric power system, daily demand patterns show large variations between peak and off-peak hours of a day. If enough generating units are kept online throughout a day to meet the peak demand of the day, then during off-peak hours, most of the unnecessary units will be operating at their minimum generation limits. It will result in an avoidable high cost of energy production. Hence, it is the job of a grid operator to decide which units to keep online and which units are to be taken offline and for how long. Scheduling the on/off status and output power of electric power generating units is defined as a Unit Commitment (UC) problem. It consists of two stages; scheduling the on/off status of a unit is called the *commitment decision* while planning how much energy a committed unit will produce at any time is called the *production decision*.

The UC problem is a mathematical optimization problem where the goal is to coordinate the production of a set of generators to match the forecasted energy demand at minimum cost/maximum profit, as depicted in *Figure 1.1*. It is one of the most fundamental constrained optimization problems in the planning and operations of a wholesale electricity market.

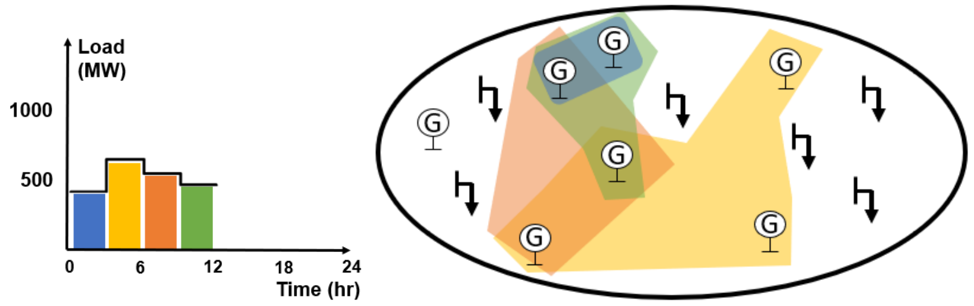


Figure 1.1 Unit Commitment Problem

1.2 Wholesale Energy Markets

Beginning in the 1990s, many US states decided to deregulate their electricity systems to create competition and lower costs. This transition, known as restructuring, required electric utilities to sell their generating assets and led to the creation of independent energy suppliers (IESs) that owned generators. Because each new IES could not cost-effectively create its own power line infrastructure, electric utilities held onto these assets and became transmission and distribution utilities. The biggest impacts resulting from deregulation were changes to retail and wholesale electricity sales, with the creation of retail customer choice and wholesale energy markets. Energy markets are auctions that are used to coordinate the production of electricity on a day-to-day basis. In a wholesale energy market, IESs offer to sell the electricity that their power plants generate for a particular bid price, while load-serving entities bid for that electricity to meet their customers' energy demand [1].

Following deregulation, Regional Transmission Organizations (RTOs) or Independent System Operators (ISOs) or Transmission System Operators (TSOs) replaced utilities as grid operators which are different types of organizations that operate the grid but do not own the resources and infrastructure within it. *Figure 1.2* illustrates the areas operated by RTOs in the United States. The U.S. has 7 RTOs i.e., California ISO (CAISO), Electric Reliability Council of Texas (ERCOT), Southwest Power Pool (SPP), Midcontinent ISO (MISO), PJM Interconnection, New York ISO (NYISO) and ISO New England (ISO-NE). RTOs are regulated by the Federal Energy Regulatory Commission (FERC) because they engage in interstate commerce (except ERCOT because it is entirely contained within Texas). Traditional wholesale electricity markets exist in the Southeast, Southwest and Northwest where vertically integrated utilities (they own the generation, transmission, and distribution systems) are responsible for system operations and management. They are regulated by state Public Utility Commissions (PUCs or PSCs).

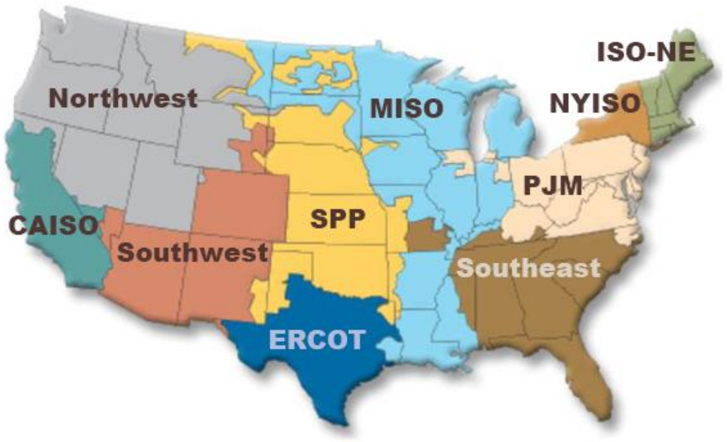


Figure 1.2 Electric Power Markets in USA [2]

While major sections of the country operate under more traditional market structures, two-thirds of the nation’s electricity load is served in RTO regions [2]. RTOs use energy markets to decide which units to dispatch or run based on the solutions obtained from the UC problem. RTOs typically run two bid-based markets to determine economic dispatch: the Day Ahead Market

(DAM) and the Real Time Market (RTM). The DAM, which represents about 95 percent of energy transactions, is based on forecasted load for the next day and typically occurs the prior morning to allow generators time to prepare for operation. The remaining energy market transactions take place in the RTM, which is typically run once every hour and once every five minutes to account for real-time load changes.

1.3 Problem Statement

Security constrained unit commitment (SC-UC) problems are solved multiple times a day by ISOs under different possible operating scenarios to schedule 4 billion mega-watthours (MWh) of annual energy production and clear a $400 billion electricity market annually in the United States [3]. The objective of a SC-UC problem is to obtain the most economical schedule for power generating resources under many operational, economical, and physical constraints. The intention is to obtain not only the most cost-efficient schedules which guarantee enough power production to satisfy the total power demand in each hour, but also make sure that they do not contravene power system security constraints e.g., maximum power carrying capacity of transmission lines, maximum and minimum power production of generating units, ramp-up and ramp-down rates, minimum on/off times, to name a few. Involvement of these security constraints makes it a very complex large-scale energy optimization problem.

UC decisions are made for a finite number of time instants, ahead in time, usually one or two days. These time instants are usually hours or half-hours. One of the practical issues in solving such large-scale SC-UC problems multiple times a day, is the computational time required to obtain economical and feasible production schedules due to the short market clearing windows in most electricity markets, after receiving all offers and bids. System operators only have few hours (typically 3 to 4 hours) to clear the DAM and produce low-cost and secure generation schedules for the upcoming day. In this short clearing window, the SC-UC problem is solved multiple times under many different probable scenarios. A typical state-of-the-art software can obtain a feasible generation schedule for a large-scale SC-UC within 0.1% optimality in about 20 minutes [4]. Improving this computational performance can allow market operators to implement several enhancements which can not only improve the market efficiency, but also bring many economic benefits e.g., implement higher resolution cost curves, sub-hourly dispatch, obtain schedules for several realistic operating scenarios in less time [5].

By exploiting the fact that ISOs solve the SC-UC problem several times a day with only minor changes in the system data, the required computational effort can be reduced by identifying the patterns in the historical data using data mining techniques. For this purpose, predictive modeling techniques can be employed to train the machine learning models offline using synthetic or historic data and then predict the commitment and production decisions of the generating resources online. Therefore, in this dissertation, we will investigate these data mining techniques to develop fast SC-UC algorithms which can either be used to provide a good warm start for a regular full MIP problem or they can also be used as alternative backup SC-UC solving algorithms.

1.4 Proposed Approaches

This thesis proposes two data driven approaches for solving a SC-UC problem. The first approach is a two-stage hybrid approach in which a mix-integer optimization problem is decomposed and solved in two stages as shown in *Figure 1.3*. In the first stage, a data based predictive classification model predicts the commitment status of each of the generators during each hour of the day using the generator cost coefficients, load requirements, and other system conditions. Once an optimal set of generators is predicted, the values of the binary variables are fixed and the UC problem becomes an optimal dispatch problem, however, with some security constraints. Hence, a complex non-convex mix-integer SC-UC problem is converted to a comparatively easier to solve convex optimization problem with reduced decision variables and less constraints. Hence, in stage-2 of the algorithm, a security constrained economic dispatch (SC-ED) problem is solved to obtain the production decisions of the committed generators.

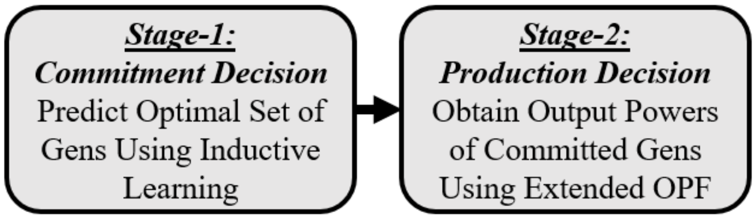

Stage-1:
Commitment Decision
Predict Optimal Set of Gens Using Inductive Learning
Stage-2:
Production Decision
Obtain Output Powers of Committed Gens Using Extended OPF

Figure 1.3 Two-Stage Proposed Algorithm (Algo-1)

In the second algorithm, commitment and production decisions are solely determined using data based predictive models as shown in *Figure 1.4*. The first stage is the same as in *Algo-1*, however, in the second stage, another predictive model is employed to determine the production decisions of the committed generators for each hour. Since both stages are based on inductive models trained on provided historical/synthetic data, there is always some uncertainty in the obtained solutions and a possibility of violating security constraints. Hence, a third and the most important stage is required to polish the solutions obtained from the first two stages. In the third stage, a post processing algorithm is used to mitigate any imbalances in the total committed generation and total power demand and eliminate any possible security constraints violations.

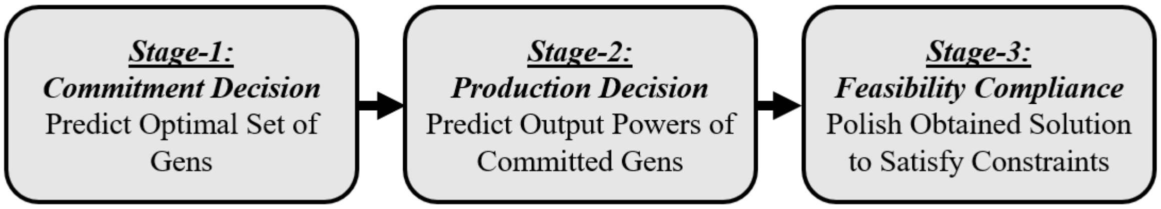

Stage-1:
Commitment Decision
Predict Optimal Set of Gens
Stage-2:
Production Decision
Predict Output Powers of Committed Gens
Stage-3:
Feasibility Compliance
Polish Obtained Solution to Satisfy Constraints

Figure 1.4 Three-Stage Proposed Algorithm (Algo-2)

The contribution of this research work is as follows:

- To improve the computational time of a conventional MIP solver for solving a security constrained unit commitment problem, a two-stage data driven hybrid algorithm (*Algo-1*) is proposed.
- To further enhance the computational speed, a three-stage data driven approach (*Algo-2*) is proposed in which the conventional MIP solver is completely replaced by data driven models to obtain solutions for binary and continuous variables in a SC-UC problem.
- To check and ensure feasibility conformance of the predicted solution obtained using the algorithm *Algo-2*, a third post processing *feasibility compliance* stage is developed.
- To fulfil transmission capacity constraints in the *feasibility compliance* stage, a generation rescheduling approach is proposed using eigen decomposition and generation shift factors.
- To achieve distributed rescheduling of generators in the *feasibility compliance* stage, a network zoning algorithm is developed using *fast community detection algorithm* to decompose a power network into communities.
- Develop a framework to implement and solve a *distributed security constrained unit commitment* (DSC-UC) in a decentralized manner using the proposed approaches (future work).

- Incorporating *an Aggregator* (a grid scale electric vehicles charging station with bi-directional power flow capability i.e., G2V and V2G) in the problem formulation of a SC-UC problem and investigate the performance of the proposed approaches to solve the problem (future work).

1.5 Overview

The dissertation consists of the following chapters:

Chapter 1. Introduction

This chapter describes the electricity market infrastructure in the USA and expresses the significance of the problem statement.

Chapter 2. Literature Survey

This chapter presents different SC-UC algorithms and techniques discussed in the literature and being used in the industry.

Chapter 3. Predictive Modeling

This chapter briefly expresses the model building process for developing predictive models using data mining techniques.

Chapter 4. Problem Formulation

This chapter formulates the presented problem and describes different components of a SC-UC problem. It also explains in detail the proposed approaches for solving a SC-UC problem.

Chapter 5. Case Studies and Simulation Results

Case studies and their obtained results for the so far completed work are presented in this chapter. It also presents the performance comparison of the proposed approaches with the IBM CPLEX MIQP solver (MATLAB CPLEX Toolbox [6]) and GUROBI MIQP solver.

Chapter 6. Conclusion and Future Work

This chapter narrates the completed work and outlines the future work which can be done as an extension to this research work.

CHAPTER

2

LITERATURE SURVEY

Due to the presence of binary variables (i.e., commitment decisions) as well as continuous variables (i.e., production decisions), UC problem is considered as one of the most complex optimization problems. Furthermore, it becomes more and more complicated with the size of the power system because of the exponentially growing possible combinations of binary variables. In literature, a lot of techniques have been discussed to solve a UC problem as given below. Some of the most discussed optimization techniques are discussed in detail here.

- Exhaustive Enumeration (EE)
- Priority List (PL)
- Dynamic Programming (DP)
- Lagrangian Relaxation (LR)
- Branch and Bound (BB)
- Fuzzy Logic (FL)
- Evolutionary Programming (EP)
- Expert Systems (ES)
- Particle Swarm Optimization (PSO)
- Tabu Search (TS)

- Mix Integer Programming (MIP)
- Genetic Algorithm (GA)
- Simulated Annealing (SA)
- Neural Network (NNET)
- Ant Colony Optimization (ACO)
- …etc.

2.1 Exhaustive Enumeration (EE)

In EE techniques, UC problems are solved by enumerating all possible combinations of the binary variables representing on/off status of the generating units. The combinations that result in least operational cost are selected as the optimal solutions for the commitment decisions. Earlier solutions of UC problems were obtained using EE techniques. Hara [7] and Kerr [8] solved the UC problems by using the exhaustive enumeration methods. Although, EE provides an accurate optimal solution, however, it is not suitable for large power systems due to exponential increase in possible combinations of the on/off status of the generating units as the system size increases which results in excessive computational time i.e., for N units and T time intervals, the total no. of possible combinations in each time interval T are

$$C_N^1 + C_N^2 + \cdots + C_N^N = 2^N - 1$$

Here $C_N^i = \frac{N!}{i!(N-i)!}$ is the combination of N units taken i units at a time. For T intervals,

$$the\ total\ no.of\ possible\ combinations = (2^N - 1)^T$$

Table 2.1 below gives a glimpse of this dimensionality curse of the solution space for 24 one-hour intervals [9]. A flowchart of a typical EE based UC algorithm is shown in *Figure 2.1*.

.

Table 2.1 Exponential Increase in No. of Combinations with No. of Gens

N	$(\mathbf{2}^{N} - \mathbf{1})^{\mathbf{24}}$
5	$6.2\text{x}10^{35}$
10	$1.73\text{x}10^{72}$
20	$3.12\text{x}10^{144}$
:	:

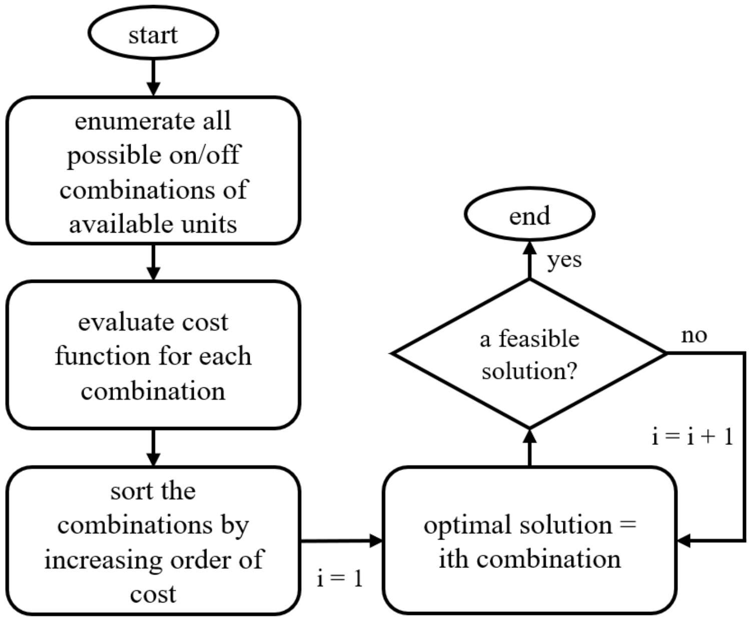

start
enumerate all possible on/off combinations of available units
evaluate cost function for each combination
sort the combinations by increasing order of cost
i = 1
optimal solution = ith combination
a feasible solution?
yes
no
i = i + 1
end

Figure 2.1 Exhaustive Enumeration Algorithm for UC Problem

2.2 Priority List (PL)

In PL method, the generating units are sorted based on lowest operational cost characteristics. This prespecified order is then used to solve the UC problem such that the system power demand is satisfied. Lee *et al.* [10]-[13] and Burns [15] solved the UC problem using PL based approaches. In [16] Shoults *et al.* presented a PL based approach to solve a UC problem which also includes import/export constraints. Lee *et al.* [13]-[14] solved a single-area and a multi-area UC problem using PL based on a classical index. The advantage of the PL based methods is the fast computational time, however, it suffers from producing generation schedules which have high operational cost. A flowchart of a basic PL algorithm is shown in *Figure 2.2* which is repeated for every hour [17].

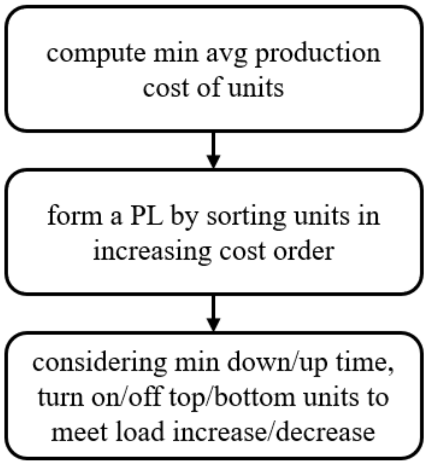

compute min avg production cost of units
form a PL by sorting units in increasing cost order
considering min down/up time, turn on/off top/bottom units to meet load increase/decrease

Figure 2.2 Priority List Algorithm for solving UC problem for each hour

2.3 Dynamic Programming (DP)

The number of possible combinations for a system having N units are 2^N-1 which increases exponentially with the system size. In this sense, DP has an advantage over enumeration method, that it comparatively reduces the dimensionality of the problem. There are two types of DP algorithms i.e., forward DP and backward DP. For UC problems, forward DP algorithm, which runs forward in time, is often used because of the time dependent start-up cost of a unit [17]. The recursive algorithm is used to compute the minimum cost in hour t with feasible state I [9] i.e.,

$$F_{cost}(K,I) = \min_{\{L\}}[P_{cost}(K,I) + S_{cost}(K-1,L:K,I) + F_{cost}(K-1,L)]$$

Here

$F_{cost}(K,I)$: total cost to arrive at state (K,I)

$P_{cost}(K,I)$: production cost for state (K,I)

$S_{cost}(K-1,L:K,I)$: transition cost from state (K − 1, L) to state (K, I)

$\{L\}$: set of feasible states in hour K – 1

$State\ (K,I)$: Ith combination in hour K

DP is one of the earliest and most widely used optimization method to be applied to the UC problem [18]. The advantage of DP based approaches is that it can be easily modified to model characteristics of specific utilities and it is relatively easier to add constraints that affect operations at an hour. However, the drawback is that it is more difficult to include constraints that are time dependent and its suboptimal treatment of minimum up and downtime constraints and time-dependent startup costs [19]. Moreover, solving a large-scale UC problem using DP can be time consuming. In [20], Lowery *et al.* studied the feasibility of using DP for solving UC problems. They implemented dynamic programming to solve UC problem for a 14-machine power system. Sen [21] solved a multi-area UC problem using truncated dynamic programming and applied to Indian Power System with two different areas having different operating characteristics. In [22], Singhal *et al.* compared the performance of conventional-DP, Sequential-DP, and Truncated-DP on a 10-machine power system. A flowchart of a forward DP algorithm is shown in *Figure 2.3* [9].

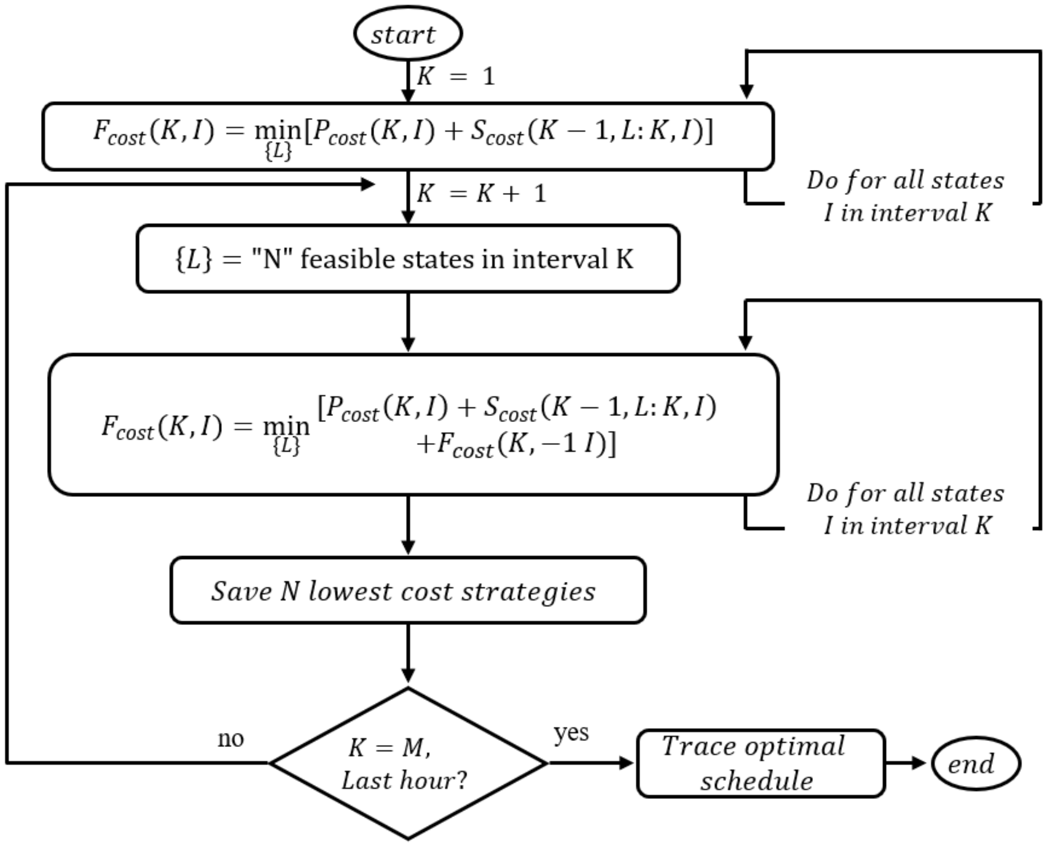

start
$K = 1$
$F_{cost}(K,I) = \min_{\{L\}}[P_{cost}(K,I) + S_{cost}(K-1, L:K,I)]$
Do for all states I in interval K
$K = K + 1$
$\{L\}$ = "N" feasible states in interval K
$F_{cost}(K,I) = \min_{\{L\}} [P_{cost}(K,I) + S_{cost}(K-1, L:K,I) + F_{cost}(K,-1\ I)]$
Do for all states I in interval K
Save N lowest cost strategies
$K = M$, Last hour?
no
yes
Trace optimal schedule
end

Figure 2.3 A Flowchart for Solving Unit Commitment Using Forward DP

2.4 Lagrangian Relaxation (LR)

For large power systems with a large no. of generators, DP algorithm has many drawbacks due to the necessity of forcing the DP method to search over a small number of commitment states to reduce total no. of combinations to be checked in each time interval. These disadvantages can be avoided using LR approach. LR method is another widely used optimization technique used by the utilities for UC problem. Its use in the utilities for UC problems is more recent than the DP method. Like DP, it can also be easily modified to model characteristics of specific utilities. It is comparatively more beneficial than DP for large systems as the degree of suboptimality goes to zero as the number of units increases. But it also has a disadvantage of having inherent suboptimality. The LR method solves the UC problem by "relaxing" or temporarily ignoring the coupling constraints and solving the problem as if they did not exist. This is done through the dual optimization procedure. The dual procedure attempts to reach the constrained optimum by maximizing the Lagrangian with respect to the Lagrange multipliers while minimizing with respect to the other variables i.e.,

$$q^*(\lambda) = \max_{\lambda^t} q(\lambda)$$

Here

$$q(\lambda) = \min_{P_i^t,\ U_i^t} L(P, U, \lambda)$$

Here, L is the Lagrangian, $\lambda^t$are the Lagrange multipliers for interval t and U_i^t and P_i^t are the i^{th} unit's on/off status and dispatch in time interval t respectively.

In [23], Merlin *et al.* proposed a new technique for UC problem using LR method and tested it at Electricite De France. Zhuang [24] proposed a new three-stage LR algorithm for UC. In the first stage, the Lagrangian dual of the UC is maximized with standard sub-gradient technique, the second stage finds a reserve feasible dual solution, and economic dispatch is done in the third stage. In [25], Takriti et al. proposed a refined LR method using integer programming. There is a lot of research been done on using LR methods, and its refinements using hybrid approaches, for solving UC problems [26]-[29]. A flowchart of a LR method is shown in *Figure 2.4* [9]. Here the primal value J^* is the total generation cost summed over all hours obtained from economic dispatch.

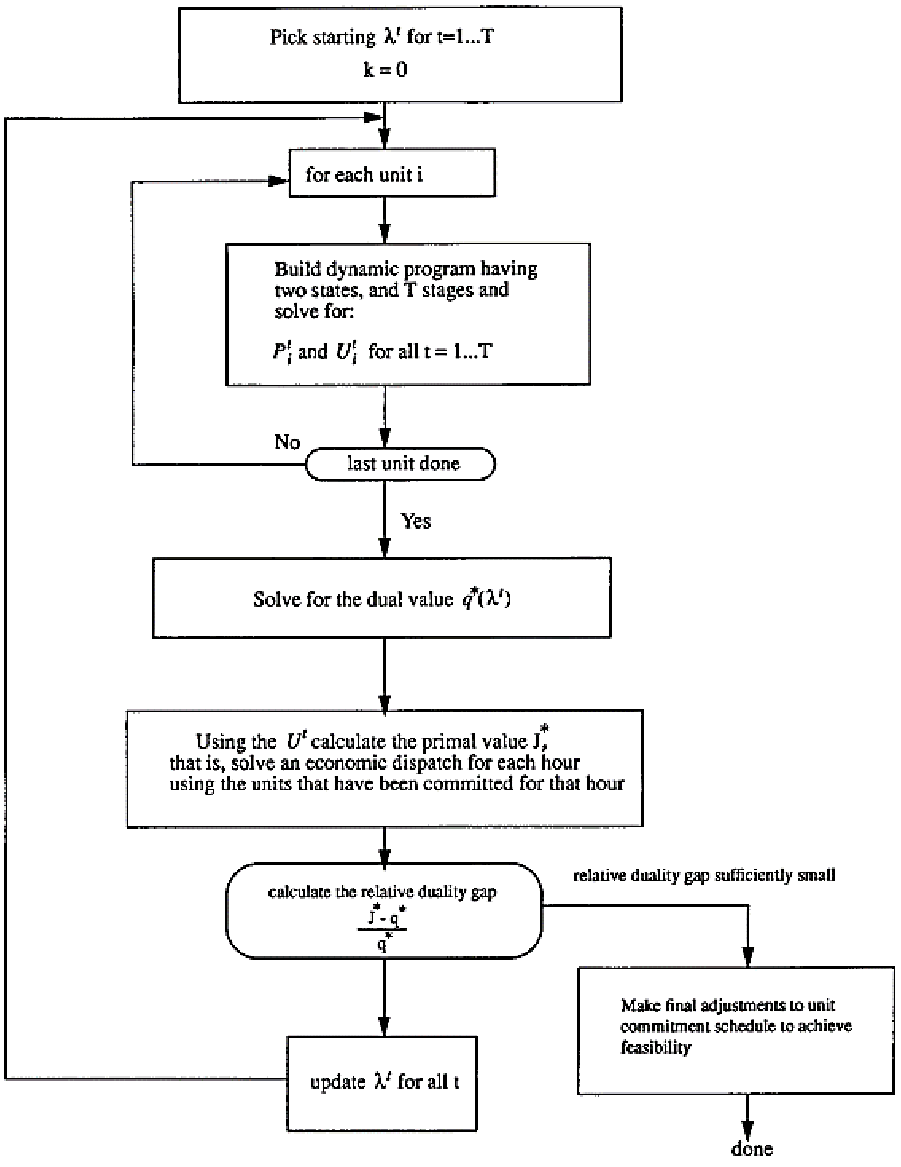

Pick starting λ^t for t=1...T
k = 0
for each unit i
Build dynamic program having two states, and T stages and solve for:
P_i^t and U_i^t for all t = 1...T
No
last unit done
Yes
Solve for the dual value $q^*(\lambda^t)$
Using the U^t calculate the primal value J^*, that is, solve an economic dispatch for each hour using the units that have been committed for that hour
calculate the relative duality gap $\frac{J^* - q^*}{q^*}$
relative duality gap sufficiently small
Make final adjustments to unit commitment schedule to achieve feasibility
update λ^t for all t
done

Figure 2.4 A Flowchart for Solving Unit Commitment Using LR Method

2.5 Branch and Bound (BB)

The BB method includes repeated application of the following steps. First, a portion of the solution space in which the optimal solution is known to exist is partitioned into subsets. Second, a subset is eliminated from further consideration if all the elements in the subset violate the constraints of the minimization problem. Third, an upper bound on the minimum value of the objective function is computed. Finally, lower bounds are computed on the value of the objective function when the decision variables are constrained to lie in each subset still under consideration [19]. A subset is then eliminated if its lower bound exceeds the upper bound of the minimization problem, since the optimal decision variable cannot lie in that subset. Convergence takes place when only one subset of decision variables remains, and the upper and lower bounds are equal for that subset [18]. Cohen [30] and Ohuch [31] presented a new approach for solving UC problem based on BB method. BB method incorporates all time-dependent constraints and does not require a priority ordering of units. In [32], Huang *et al.* proposed a constraint logic programming along with the BB technique to provide an efficient and flexible approach to the UC problem. The drawback of the BB method is large storage capacity and high execution time especially for large power systems.

2.6 Mix Integer Programming (MIP)

Unit commitment problem is a very large mixed-integer constrained optimization problem due to the presence of both integer and continuous variables. Dillon *et al.* [33]-[34] developed an integer programming method for practical size scheduling problem based on the extension and modification of the branch-and-bound method. The UC problem can be partitioned into a nonlinear economic dispatch problem and a pure integer nonlinear UC problem based on benders approach. Whereas the mixed integer programming (MIP) approach solves the UC problem by reducing the solution search space through rejecting infeasible subsets. A linear programming (LP) UC problem can be solved either by decomposing the whole problem into subproblems with help of Dantzig–Wolfe decomposition principle and then each subproblem is solved using linear programming or the problem can be solved directly by revised simplex technique [35].

2.7 Evolutionary Programming

During the last few decades, there has been a growing interest in problem-solving systems based on the principles of evolution. Such systems maintain a population of potential solutions based on fitness of individuals and some “genetic” operators. One such system is a class of evolution strategies (i.e., algorithms which imitate the principles of natural evolution for parameter optimization problems). Sheble *et al.* [36], applied the genetic algorithm (GA) to the UC problem for one to seven days. The feasibility of genetic algorithms application for UC problems has been examined for both small- and large-size problems [37]. Maifeld *et al.* [38] presented a new UC scheduling algorithm using genetic algorithm with domain-specific mutation operators. The robustness of the proposed algorithm is demonstrated by comparison to a Lagrangian Relaxation UC algorithm on different utilities. Swarup *et al.* [39] employed a new strategy for representing chromosomes and encoding the problem search space, which is efficient and can handle large-scale UC.

In the 90s and early 2000s, researchers experimented to solve UC problem using neural networks (NNET), but it proved very challenging to obtain high quality solutions due to processing limitations [40]-[41]. In recent years, there has been interest in using machine learning (ML) within optimization community, however, their main objective has been to improve the performance of the mix-integer-programming solvers by reducing the no. of constraints instead of completing replacing them with the data driven techniques [42]-[43].

CHAPTER 3

PREDICTIVE MODELING

Predictive modeling is the process of developing a mathematical model that generates an accurate prediction. Geisser [44] defines predictive modeling as the process by which a model is created or chosen to try to best predict the probability of an outcome. We can easily find a lot of examples of predictive models in our daily life e.g., Banks use predictive models for fraud detection, spam filters use them to trash spam emails, Netflix and YouTube use these models to recommend videos to their viewers, etc. Predictive models have even been used in the US elections e.g., MIT Technology Review [45] reports that it was the Obama campaign's effective use of data mining that helped President Obama win the 2012 presidential election over Mitt Romney. The process of developing predictive models has evolved throughout several fields such as chemistry, computer science, physics, and statistics and is called by several names such as *machine learning*, *artificial intelligence*, *pattern recognition*, *data mining*, *predictive analytics*, *knowledge discovery*, however, the ultimate objective is the same: to make an accurate prediction and they all belong to a broader class of predictive/inductive modeling [46].

One of the key ingredients of a predictive model is *relevant data*. Vast amount of data is being collected every moment in our daily life. The McKinsey Global Institute (MGI) [47] reports that most American companies with more than 1000 employees have an average of at least 200 terabytes of stored data. MGI projects that the amount of data generated worldwide will increase by 40% annually. The task of a predictive model is to leverage this data and extract knowledge from these large datasets to be used for prediction [48].

3.1 Model Building Process:

Building a predictive model is an iterative process. There are several steps involved in the process of building a predictive model. It is described with slight differences in different fields, however, the core concept behind is the same. For example, Kuhn [46] describes the model building process as shown in *Figure 3.1*. The received data is initially pre-processed for transformation from the raw format into an appropriate format for subsequent analysis. There are several steps involved within the pre-processing step itself e.g., normalization, dimensionality reduction etc. Next, the data is split for training and evaluation. After that, different suitable models are trained and, based on some performance metric e.g., accuracy, F1-score, AUC, MSE. MAE etc., the best model is selected for future prediction. Similar processes are discussed in some other literature works as shown in *Figure 3.2* through *Figure 3.5* [48]-[51].

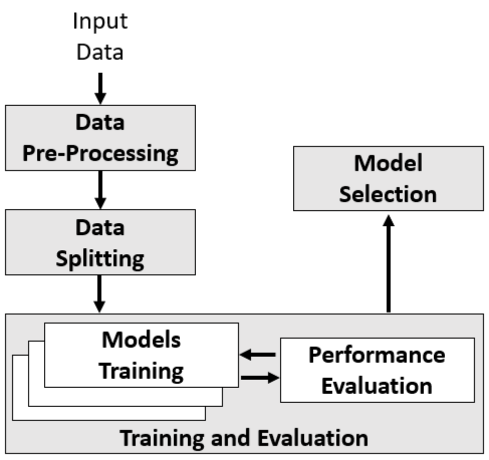


Figure 3.1 Predictive Model Building Process [46]

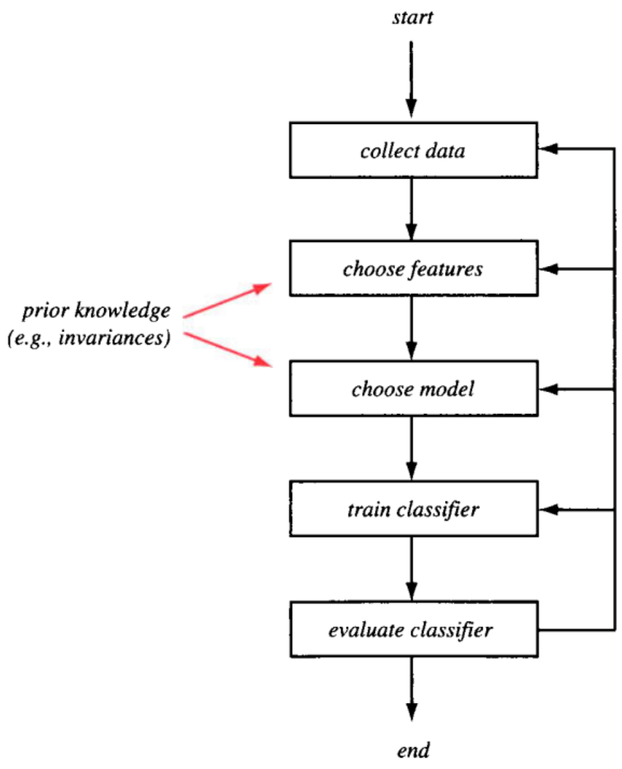


Figure 3.2 Design Cycle of a Pattern Recognition System [49]

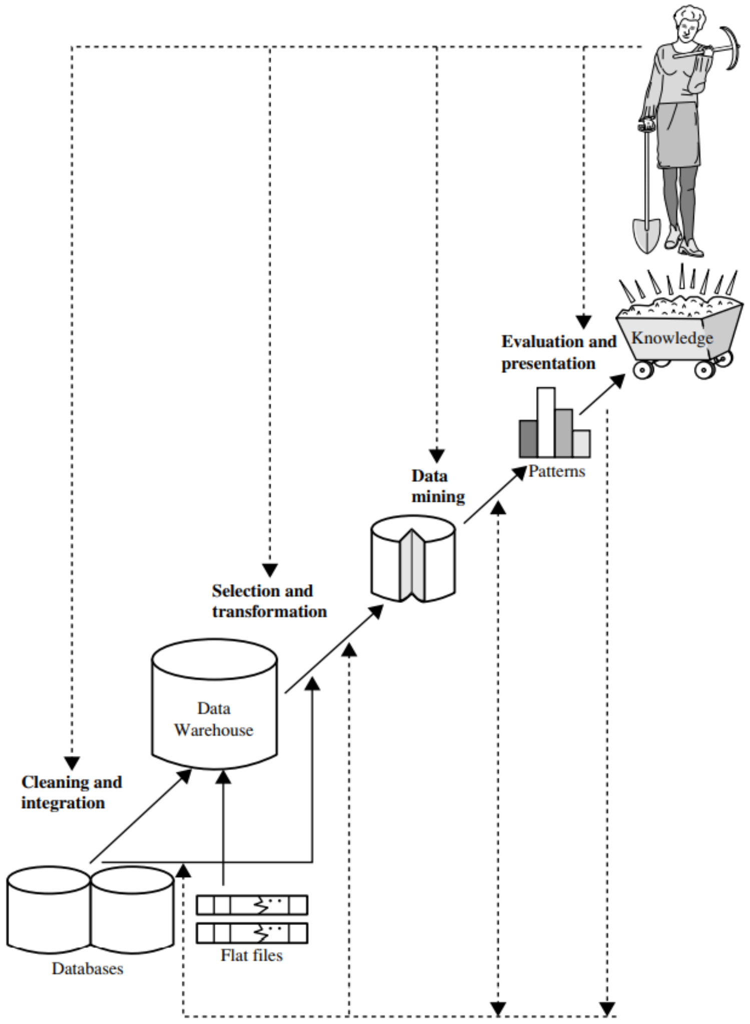


Figure 3.3 Process of Data Mining [50]

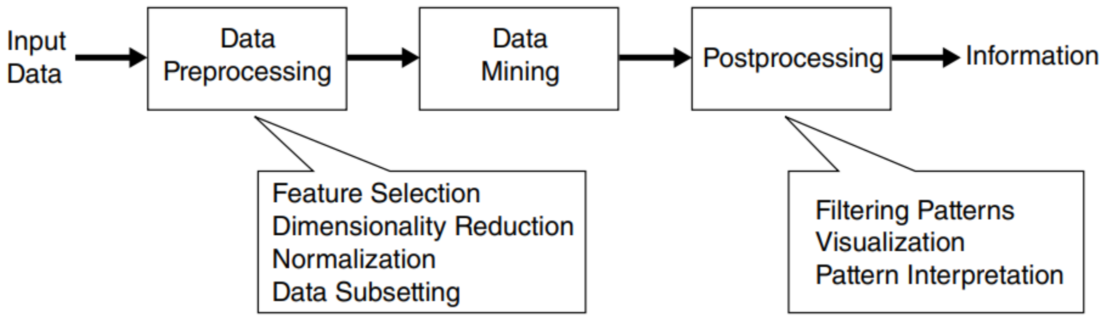


Figure 3.4 Knowledge Discovery Process in Databases [51]

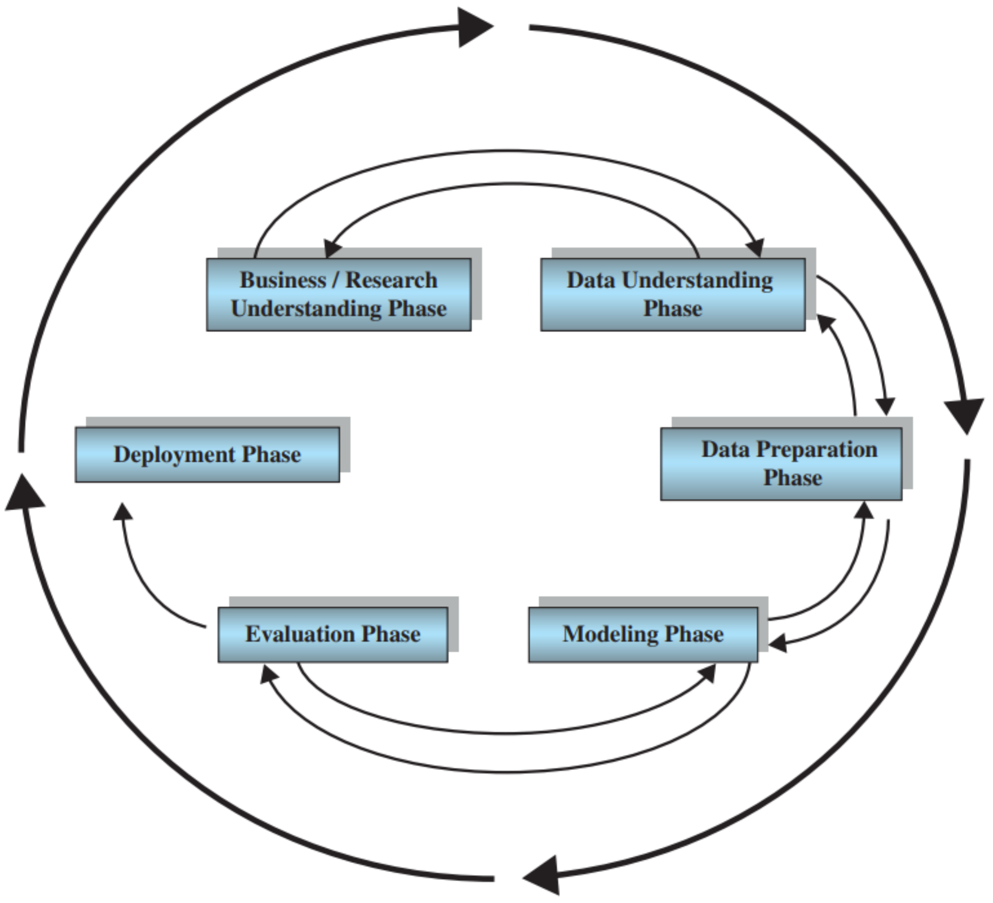


Figure 3.5 Cross-Industry Standard Process for Data Mining (CRISP-DM) [48]

3.1.1 Data Exploration:

After collecting the raw data, the first step in model building process is to understand and getting familiar with the data. Usually, we want to know the following: What are the types of attributes or features that make up the data? What kind of values does each attribute have? Which attributes are discrete, and which are continuous-valued? What do the data look like? How are the values distributed? Are there ways we can visualize the data to get a better sense of it all? Can we spot any outliers? Can we measure the similarity of some data objects with respect to others? Gaining such insight into the data will help with the subsequent analysis. Datasets differ in several ways. For example, the attributes used to describe data objects can be of different types—

quantitative or qualitative—and datasets may have special characteristics, e.g., some datasets contain time series or objects with explicit relationships to one another. The type of data determines which tools and techniques can be used to analyze the data.

While most data mining techniques can tolerate some level of imperfection in the data, a focus on understanding and improving data quality typically improves the quality of the resulting analysis. Data quality issues that often need to be addressed include the presence of noise and outliers, missing, inconsistent, or duplicate data, and data that is biased or, in some other way, unrepresentative of the phenomenon or population that the data is supposed to describe. Hence some kind of exploratory data analysis (EDA) e.g., summary statistics, visualization, multi-dimensional data analysis etc., is needed to discover initial insights and evaluate the quality of the data. Finally, if desired, some interesting feature/sample subsets are selected that may contain actionable patterns.

3.1.2 Data Preparation:

The next step in the model building process is to preprocess the data and prepare it for modeling. Real-world data tend to be dirty, incomplete, and inconsistent. Data preprocessing techniques can improve data quality, thereby helping to improve the accuracy and efficiency of the subsequent modeling process. Data preprocessing is an important step in the predictive modeling process because quality decisions must be based on quality data. Detecting data anomalies, rectifying them early, and reducing the data to be analyzed can lead to huge payoffs for decision making. There are several steps involved in data preprocessing. *Figure 3.6* summarizes typical steps involved in preprocessing.

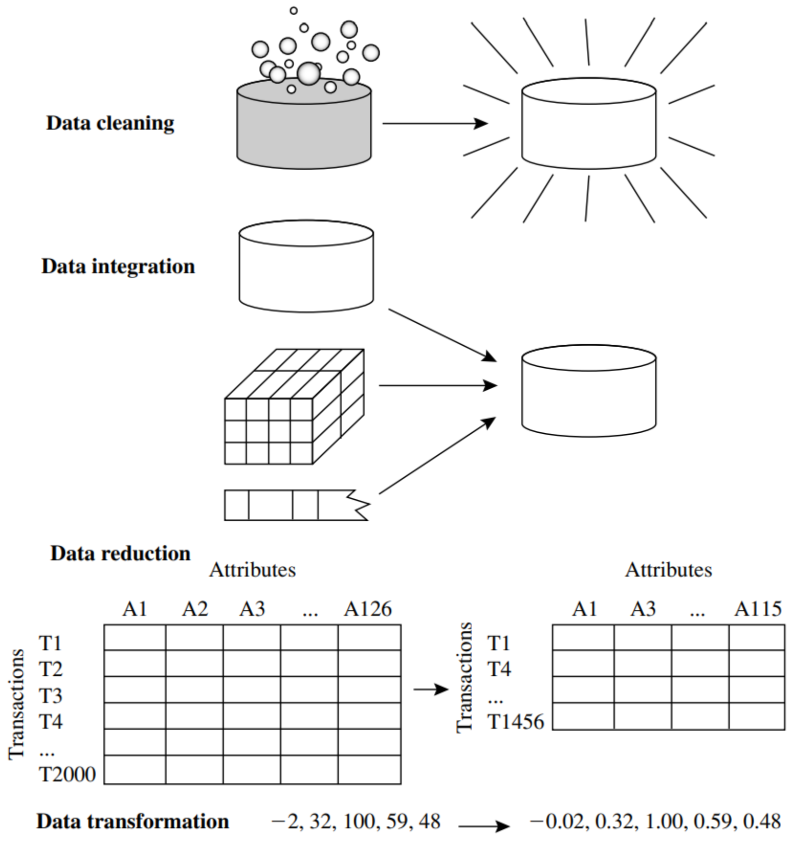


Figure 3.6 Steps Involved in Data Preparation [50]

Data cleaning routines work to "clean" the data by filling in missing values, smoothing noisy data, identifying, or removing outliers, and resolving inconsistencies. If users believe that the data are dirty, they are unlikely to trust the results of any data mining that has been applied. Furthermore, dirty data can cause confusion for the modeling procedure, resulting in unreliable output. Although most mining routines have some procedures for dealing with incomplete or noisy data, they are not always robust. Instead, they may concentrate on avoiding overfitting the data to the function being modeled. Therefore, a useful preprocessing step is to run your data through some data cleaning routines.

Collecting data from multiple sources involve integrating multiple databases, data cubes, or files. This is called **data integration**. This may cause inconsistencies and redundancies. For example, some attributes may have different names in different databases e.g., the attribute for customer identification may be referred to as *customer_id* in one dataset and *cust_id* in another. Naming inconsistencies may also occur for attribute values. For example, the same first name could be registered as “Bill” in one database, “William” in another, and “B.” in a third. Furthermore, having a large amount of redundant data may slow down the learning and/or confuse the knowledge discovery process. Typically, data cleaning and data integration are performed as a single step in data preparation because additional data cleaning may be needed to detect and remove redundancies that may have resulted from data integration.

Data reduction obtains a reduced representation of the dataset that is much smaller in volume yet produces the same (or almost the same) analytical results. Data reduction strategies include dimensionality reduction and numerosity reduction. In dimensionality reduction, data encoding schemes are applied to obtain a reduced or “compressed” representation of the original data. Examples include data compression techniques (e.g., wavelet transforms and principal components analysis), attribute subset selection (e.g., removing irrelevant attributes), and attribute construction (e.g., where a small set of more useful attributes is derived from the original set). In numerosity reduction, the data are replaced by alternative, smaller representations using parametric models (e.g., regression or log-linear models) or nonparametric models (e.g., histograms, clusters, sampling, or data aggregation).

Distance based modeling algorithms, such as neural networks, nearest-neighbor classifiers, or clustering provide better results if the data to be analyzed is normalized, that is, scaled to a smaller range such as [0.0, 1.0]. Customer data, for example, contains the attributes *age* and *annual salary*. The annual salary attribute usually takes much larger values than age. Therefore, if the attributes are left unnormalized, the distance measurements taken on annual salary will generally outweigh distance measurements taken on age. Discretization and concept hierarchy generation can also be useful, where raw data values for attributes are replaced by ranges or higher conceptual levels. For example, raw values for age may be replaced by higher-level concepts, such as youth, adult, or senior. Normalization, data discretization, and concept hierarchy generation are forms of **data transformation**.

3.1.3 Data Splitting:

One of the first decisions to make when modeling is to decide which samples will be used to evaluate performance. Ideally, the model should be evaluated on samples that were not used to build or fine-tune the model, so that they provide an unbiased estimate of model performance. For comparison of different methods of splitting data, consult reference [52].

For **large datasets** (i.e., large number of samples), a set of samples can be set aside to evaluate the final model. The *training* dataset is the general term for the samples used to create the model, while the *test* or *validation* dataset is used to evaluate performance. Several methods for splitting the samples are used. Nonrandom sampling methods to split the data are sometimes appropriate. However, in most cases, there is the desire to make the training and test sets as homogeneous as possible. Random sampling methods can be used to create homogeneous datasets. The simplest way is to take a *simple random sample*. This does not control for any of the data attributes, such as the percentage of data in the classes. When one class has a disproportionately small frequency compared to the others, there is a chance that the distribution of the outcomes may be substantially different between the training and test sets. To account for the outcome when splitting the data, *stratified random sampling* applies random sampling within subgroups (such as the classes). In this way, there is a higher likelihood that the outcome distributions will match.

When the outcome is a number, a similar strategy can be used i.e., the numeric values are broken into similar groups (e.g., low, medium, and high) and the randomization is executed within these groups. Alternatively, the data can be split based on the predictor values e.g., data splitting based on *maximum dissimilarity sampling*. Dissimilarity between two samples can be measured in several ways. The simplest method is to use the distance between the predictor values for two samples. Larger distances between points are indicative of dissimilarity. *Figure 3.7* shows an example of maximum dissimilarity sampling.

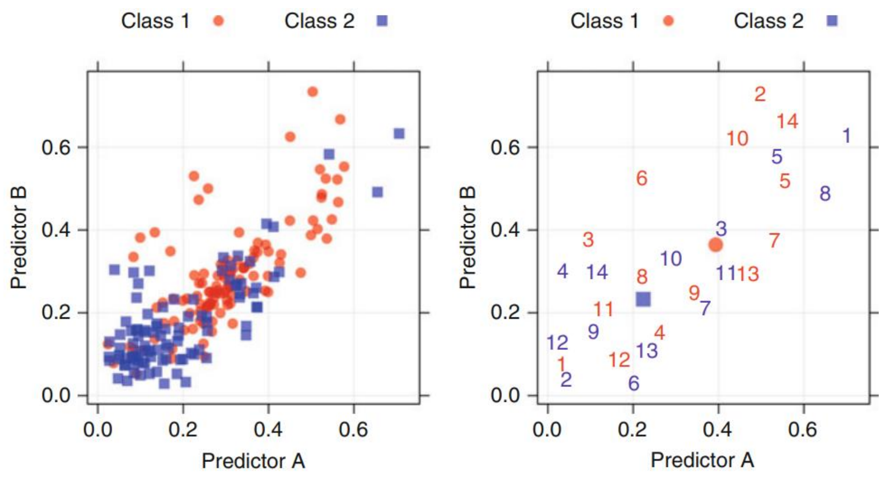


Figure 3.7 An example of maximum dissimilarity sampling to create a test dataset (right) from original dataset (left) [46]

For **small datasets**, a separate test dataset is usually avoided because every sample may be needed for model building. Additionally, the size of the test set may not have sufficient power or precision to make reasonable judgements. Therefore, validation using a single test set can be a poor choice. Resampling methods are usually used in such scenarios for estimating model performance and they often produce performance estimates superior to a single test set because they evaluate many alternate versions of the data. There are two main methods used for resampling i.e., *cross-validation* and *bootstrap*. There are different versions of cross validation. In *k-fold cross* *validation*, the samples are randomly partitioned into k sets of roughly equal size. A model is fit using all the samples except the first subset (called the first fold). The held-out samples are predicted by this model and used to estimate performance measures. The first subset is returned to the training set and procedure repeats with the second subset held out, and so on. The k resampled estimates of performance are summarized (usually with the mean and standard error) and used to estimate the model performance. The 3-fold cross-validation process is shown in *Figure 3.8*.

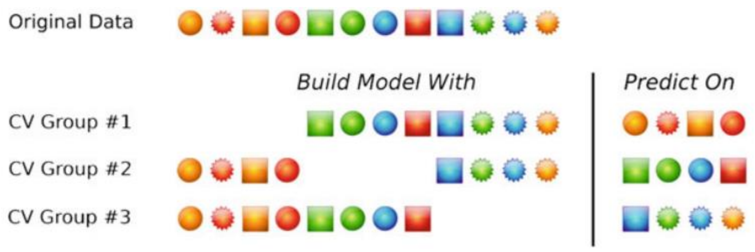


Figure 3.8 A schematic of 3-fold cross validation [46]

A slight variant of this method is *stratified k-fold cross validation* which makes the folds balanced with respect to the outcome. Additionally, *repeated k-fold cross validation* replicates the procedure in *Figure 3.8* multiple times. *Leave-one-out cross validation* (LOOCV) is the special case where only one sample is held-out at a time, the final performance is calculated from the k individual held-out predictions where k is the number of samples. In *Leave-group-out cross validation* (LGOCV) or *Monte-Carlo cross validation*, repeated training/test splits are performed as shown in *Figure 3.9*.

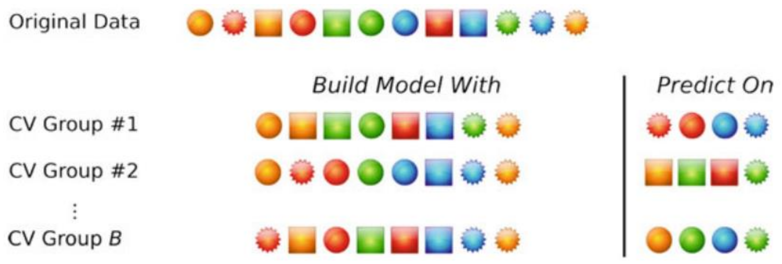


Figure 3.9 A schematic of B repeated Monte-Carlo cross validation [46]

The *bootstrap* sample is a random sample of the data taken with replacement. This means that, after a data point is selected for the subset, it is still available for further selection. The bootstrap sample is the same size as the original data set. As a result, some samples will be represented multiple times in the bootstrap sample while others will not be selected at all. The samples not selected are referred to as the "out-of-bag" samples. For a given iteration of bootstrap resampling, a model is built on the selected samples and evaluated using the out-of-bag samples (*Figure 3.10*). Like cross validation, there are some versions of the bootstrap method e.g., *632 method*, *632+ method*.

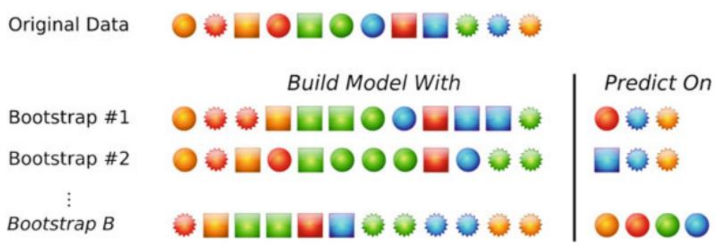


Figure 3.10 A schematic of bootstrap resampling [46]

3.1.4 Model Fitting:

Machine learning algorithms can be categorized into two main categories i.e., *supervised*, and *unsupervised* as shown in *Figure 3.11*. In unsupervised methods, no target variable is identified. Instead, the learning algorithm searches for patterns and structure among all the variables. The most common unsupervised method is clustering. It is used for exploratory data analysis to find hidden patterns or groupings in data. Most learning algorithms, however, are supervised methods. In supervised methods, a particular prespecified target variable is available so that the algorithm may learn which values of the target variable are associated with which values of the predictor variables. Examples of supervised methods are classification and regression [53].

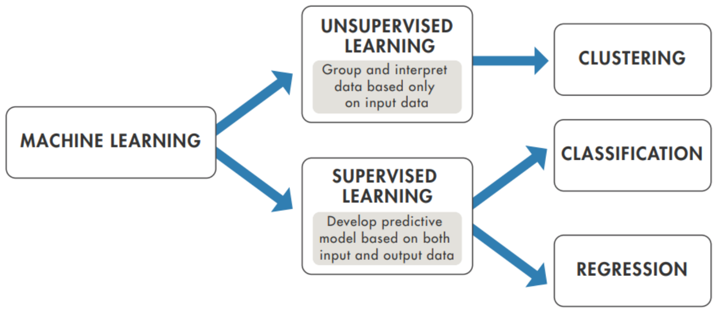

MACHINE LEARNING
UNSUPERVISED LEARNING
Group and interpret data based only on input data
SUPERVISED LEARNING
Develop predictive model based on both input and output data
CLUSTERING
CLASSIFICATION
REGRESSION

Figure 3.11 Machine learning algorithms [53]

Supervised vs Unsupervised:

The decision to choose which method to use for predictive modeling depends mainly on: What kind of data are we working with? What insights do we want to get from it? How and where will those insights be applied? Answers to these questions help us decide whether to use supervised or unsupervised learning. Supervised learning is chosen if we need to train a model to make a prediction e.g., predicting temperature or a stock price, or identify makes of cars from webcam video footage. We choose unsupervised learning if we need to explore data and want to train a model to find a good internal representation, such as splitting data up into clusters.

Overfitting:

Many modern algorithms are highly adaptable and capable of modeling complex relationships. However, they can very easily over-emphasize patterns that are not reproducible. This type of model is said to be *over-fit* and will usually have poor accuracy when predicting a new sample. *Figure 3.12* shows example class boundaries from two distinct binary classifiers. The lines envelop the area where each model predicts the data to be the second class (blue squares). The left-hand panel (Model #1) shows a boundary that is complex and attempts to encircle every possible data point. This pattern is not likely to generalize to new data. The right-hand panel shows an alternative model fit where the boundary is smooth and does not overextend itself to correctly classify every data point in the training set.

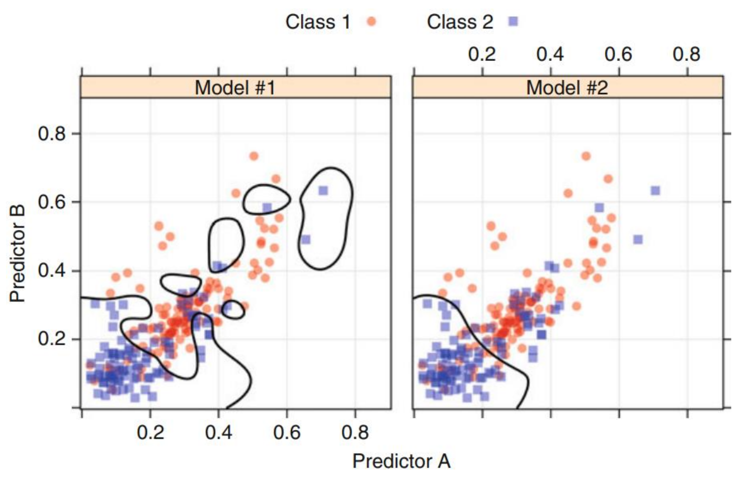


Figure 3.12 An example of overfitting [46]

In 2D, it is not difficult to visualize that one model is over-fitting, but most modeling problems are in much higher dimensions. In these situations, it is very important to have a tool for characterizing how much a model is over-fitting the training data. . There is an eternal tension in model building between model complexity (resulting in high accuracy on the training set) and generalizability to the test and validation sets. Increasing the complexity of the model to increase the accuracy on the training set eventually and inevitably leads to a degradation in the generalizability to the test set, as shown in *Figure 3.13*. The optimal level of model complexity is at the minimum error rate on the test set.

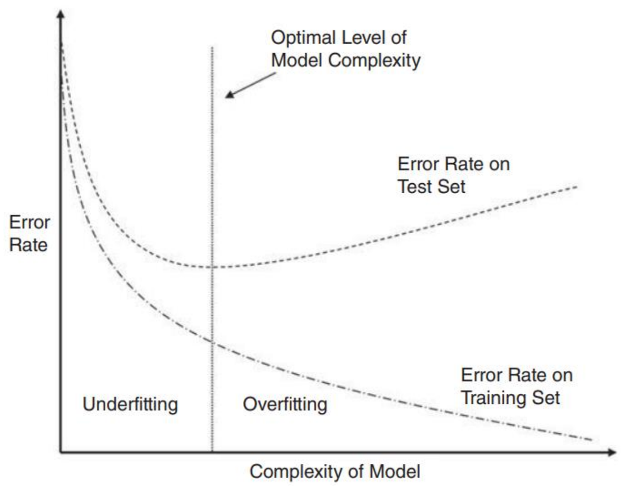


Figure 3.13 Error rate vs model complexity [48]

Class Imbalance:

In classification models, the relative frequencies of the classes can have a significant impact on the effectiveness of the model. An imbalance occurs when one or more classes have very low proportions in the training data as compared to the other classes. There are several ways for overcoming class imbalances e.g., model tuning can be used to increase the sensitivity of the minority class, alternative probability cutoffs can be derived from the data to improve the error rate on the minority class, modifications of case weights and prior probabilities, balancing the training dataset using sampling methods to mitigate the class imbalance prior to model training and cost-sensitive training to emphasize the accuracy of the model for the less frequent class(es).

Hyperparameter Tuning:

Many models have important parameters which cannot be directly estimated from the data, and they affect the performance of the models. For example, in the k-nearest neighbor (KNN) classification model, a new sample is predicted based on the k-closest data points in the training set. A choice of too few neighbors may over-fit the individual points of the training set while too many neighbors may not be sensitive enough to yield reasonable performance. These types of model parameters are referred to as a *hyperparameters* and there is no analytical formula available to calculate their appropriate values. There are different approaches used for *hyperparameter tuning* to obtain their optimized values. A general heuristic approach (called *random/grid search*) that can be applied to almost any model is to randomly define a set of candidate values, train the model, obtain performance metrics, and then choose the parameter values which give best performance. A flowchart of this process for hyperparameter tuning is shown in *Figure 3.14*.

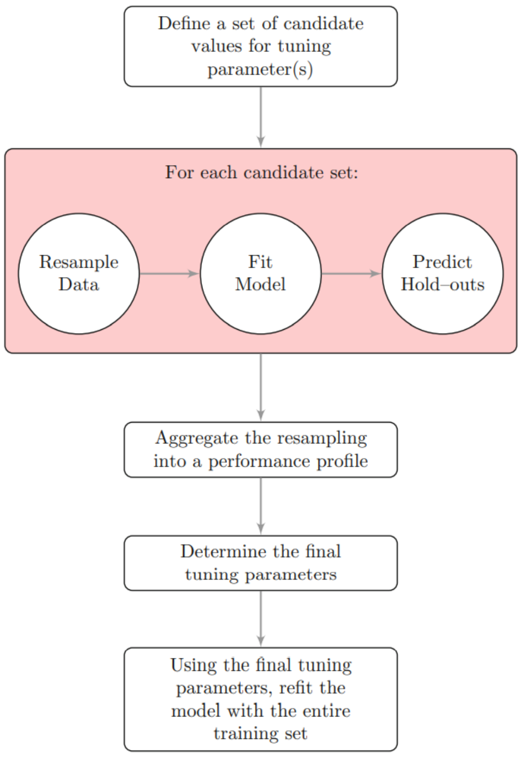

Define a set of candidate values for tuning parameter(s)
For each candidate set:
Resample Data
Fit Model
Predict Hold–outs
Aggregate the resampling into a performance profile
Determine the final tuning parameters
Using the final tuning parameters, refit the model with the entire training set

Figure 3.14 A schematic of the hyperparameter tuning process [46]

Other approaches such as genetic algorithms or simplex search methods can also find optimal tuning parameters. These procedures algorithmically determine appropriate values for tuning parameters and iterate until they arrive at parameter settings with optimal performance. These techniques tend to evaluate many candidate models and can be superior to the random search procedure presented above.

Performance Evaluation:

There are several different metrics and techniques to evaluate the performance of a predictive model. Different performance metrics are used for classification and regression models. A common method for describing the performance of a classification model is based on the counts of correctly and incorrectly predicted data samples. These counts are tabulated in the form of a table called *confusion matrix/contingency table*. *Figure 3.15* shows confusion matrix for a binary classification model. Diagonal cells denote cases where the classes are correctly predicted while the off diagonals illustrate the number of errors for each class.

		Predicted Class		**Total**
		1	**0**	
Actual Class	**1**	True Positives (TP)	False Negatives (FN)	Total Actually Positives (P)
	0	False Positives (FP)	True Negatives (TN)	Total Actually Negatives (N)
Total		Total Predicted Positives (P’)	Total Predicted Negatives (N’)	Grand Total (P+N)

Figure 3.15 Confusion matrix for a binary classifier

There are four terms which act as building blocks in computing many performance measures i.e., *True positives* (TP) and *true negatives* (TN) refer to the positive and negative samples respectively that are correctly predicted by the classifier. *False positives* (FP) and *false negatives* (FN) are the negative and positive samples that are incorrectly predicted as positives and negatives respectively. Some of the confusion-matrix-based measures and other performance measures which are used for accessing and comparing the performance of classification models are discussed here.

Accuracy: The accuracy (aka recognition rate) of a classifier on a given dataset is the percentage of samples that are correctly classified by the classifier. The disadvantage of using this simple statistic is that it makes no distinction about type of errors.

$$accuracy = \frac{no.\,of\ correct\,predictions}{total\,no.\,of\ predictions} = \frac{TP + TN}{P + N}$$

Kappa: The Kappa statistic is to assess the agreement between the observed accuracy (OA) and expected accuracy (EA). The statistic can take on values between −1 and 1. A value of 0 means there is no agreement between the observed and predicted classes, while a value of 1 indicates perfect concordance of the model prediction and the observed classes. Negative values indicate that the prediction is in the opposite direction of the truth.

$$Kappa = \frac{OA - EA}{1 - EA}$$

Error Rate: The error rate (aka misclassification rate) is the percentage of samples that are incorrectly classified.

$$error\ rate = 1 - accuracy = \frac{no.of\ incorrect\ predictions}{total\ no.of\ predictions} = \frac{FP + FN}{P + N}$$

Sensitivity (Recall): Sensitivity/Recall (aka true positive rate) is the proportion of positive samples that are correctly identified. It can be thought of as a measure of completeness.

$$Sensitivity\ (Recall) = \frac{no.of\ correct\ positive\ predictions}{total\ no.of\ positive\ samples} = \frac{TP}{P}$$

Specificity: Specificity (aka true negative rate) is the proportion of negative samples that are correctly classified.

$$Specificity = \frac{no.of\ correct\ negative\ predictions}{total\ no.of\ negative\ samples} = \frac{TN}{N}$$

It can be noticed that accuracy is a function of sensitivity and specificity.

$$Accuracy = Sensitivity\frac{P}{(P + N)} + Specificity\frac{N}{(P + N)}$$

Youden's J-index: It is the proportions of correctly predicted samples for both positive and negative samples. In some contexts, this may be an appropriate method for summarizing the magnitude of both types of errors.

$$J = Sensitivity + Specificity - 1$$

Precision: It is the percentage of total positive predicted samples that are correctly identified as positive samples. It can be thought of as a measure of exactness.

$$Precision = \frac{no.of\ correct\ positive\ predictions}{total\ no.of\ positive\ predictions} = \frac{TP}{P'}$$

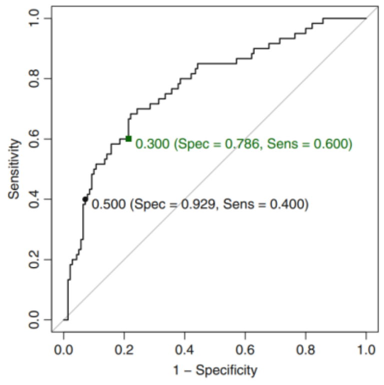
F$_\beta$-measure: It is a measure which combines precision and recall into a single measure. It is a weighted measure of precision and recall. Here β is a non-negative real number. And its value is chosen such that recall is considered β times as important as precision.

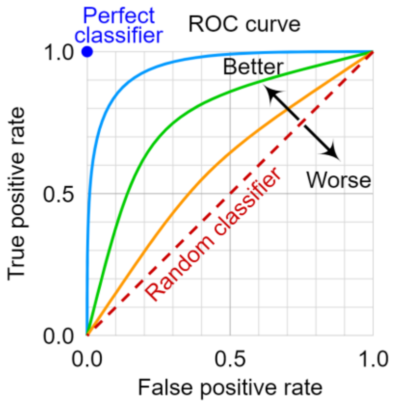
$$F_\beta = \frac{(1+\beta^2) \times precision \times recall}{(\beta^2 \times precision) + recall}$$

Commonly used F$_\beta$-measure is F$_1$-measure (aka F$_1$-score), which gives the harmonic mean of precision and recall.

$$F_1 = \frac{precision \times recall}{precision + recall}$$

ROC curve: The ROC curve is created by evaluating the class probabilities for the model across a continuum of thresholds. For each candidate threshold, the resulting true-positive rate (the sensitivity) and the false-positive rate (one minus the specificity) are plotted against each other. This plot is a helpful tool for choosing a threshold that appropriately maximizes the trade-off between sensitivity and specificity as shown in *Figure 3.16(a)* for an example classifier with 30% (green) and 50% (black) threshold values highlighted on the ROC curve. The ROC curve can also be used for a quantitative assessment of the model. A perfect binary classifier that completely separates the two classes would have 100% sensitivity and specificity. Graphically, the ROC curve would be a single step between (0, 0) and (0, 1) and remain constant from (0, 1) to (1, 1) as shown in *Figure 3.16(b)*. The area under the ROC curve (AUC) for such a model would be 1. A completely ineffective model would result in an ROC curve that closely follows the 45◦ diagonal line and would have AUC of approximately 0.50. To visually compare different models, their ROC curves can be superimposed on the same graph. Comparing ROC curves can be useful in contrasting two or more models with different predictor sets (for the same model), different tuning parameters (within model comparisons), or completely different classifiers (between-model comparison).





(a) (b)

Figure 3.16 (a) ROC Curve for an example classifier [46], (b) ROC curve

Lift and Gain Charts: For classification models, *lift* is a concept, originally from the marketing field, which seeks to compare the response rates with and without using the classification model. Lift charts and gains charts are graphical evaluative methods for assessing and comparing the usefulness of classification models. Lift is defined as the proportion of positive hits in the set of the model's positive classifications, divided by the proportion of positive hits in the data set overall.

$$Lift = \frac{proportion\ of\ positive\ hits\ in\ set\ of\ positive\ classification}{proportion\ of\ positive\ hits\ in\ the\ dataset\ as\ a\ whole}$$

Lift charts are often presented in their cumulative form, where they are denoted as *cumulative lift charts*, or *gains charts*. Lift charts and gains charts can also be used to compare model performance. *Figure 3.17* shows the combined lift chart for two example models 1 (red) and 2 (blue). The figure shows that when it comes to model selection, a particular model may not be uniformly preferable. For example, up to about the 6th percentile, there appears to be no apparent difference in model lift. Then, up to approximately the 17th percentile, model 2 is preferable, providing slightly higher lift. Thereafter, model 1 is preferable. Gain charts for two example models is shown in *Figure 3.18*. The diagonal on the gains chart is analogous to the horizontal axis at lift=1 on the lift chart. It is desired to see gains chart to rise steeply from left to right and then gradually flatten out.

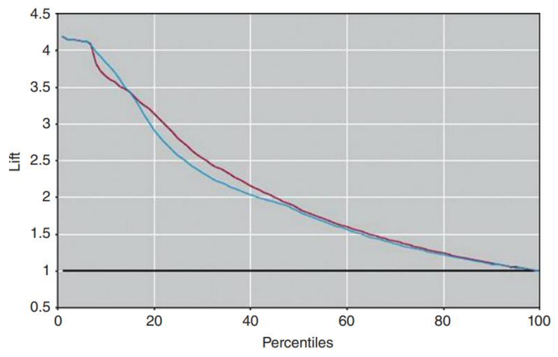


Figure 3.17 Combined lift charts for models 1 (red) and 2 (blue) [48]

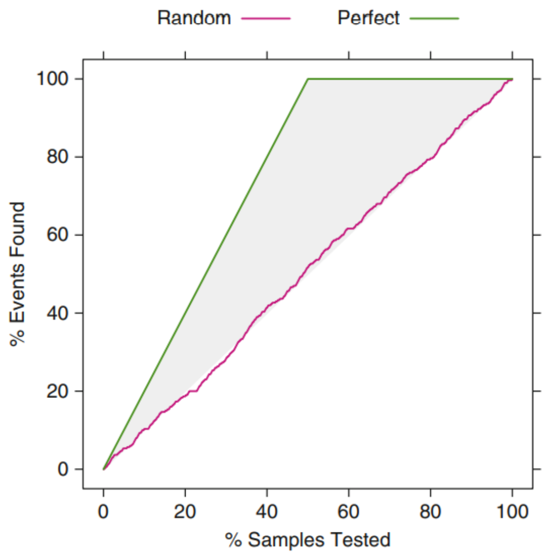
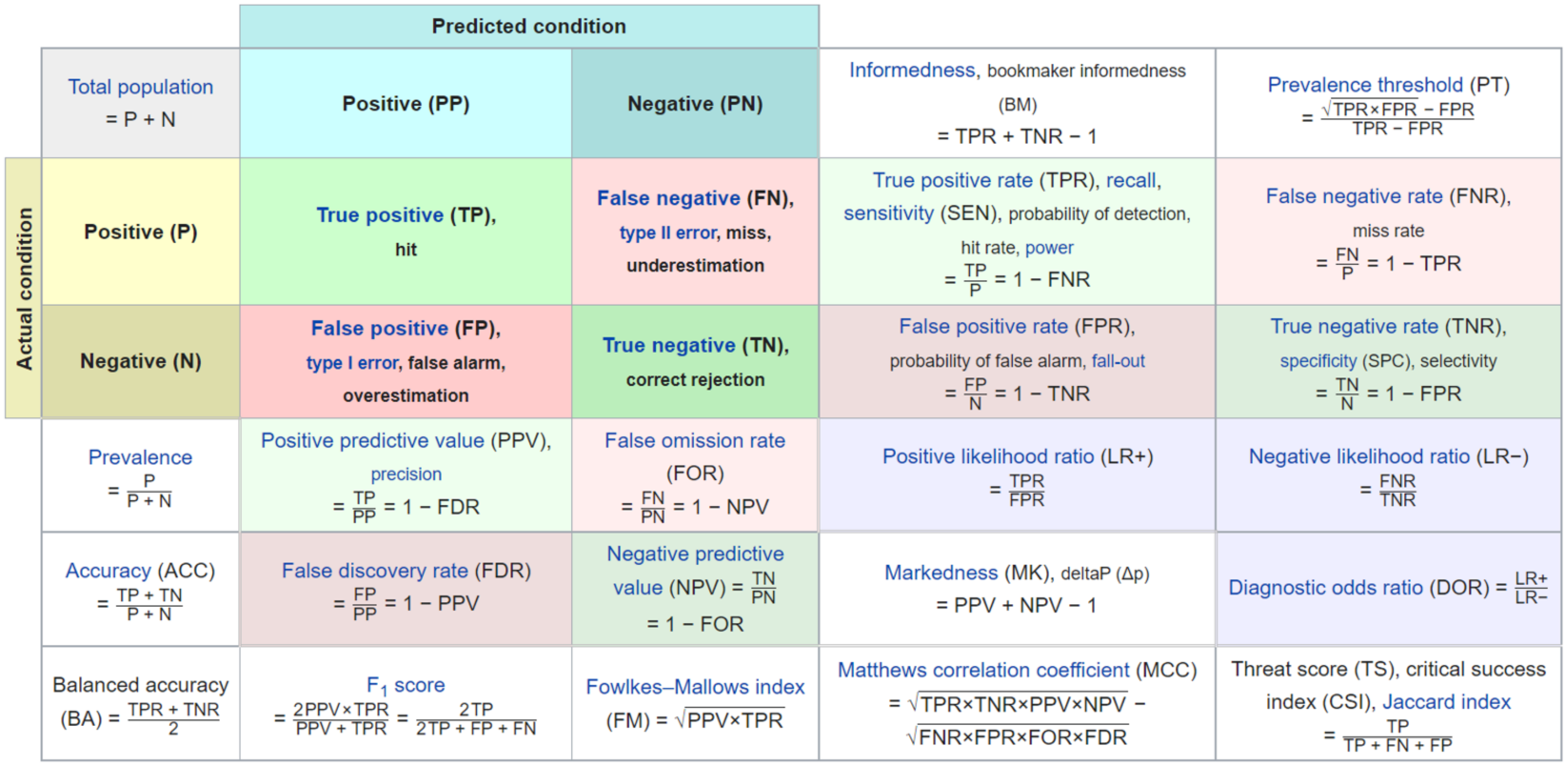


Figure 3.18 Gain charts for two example models: one that perfectly separates two classes and another that is completely non-informative [46]

There are a lot of quantitative and graphical performance measures e.g., calibration plots, profit charts etc. discussed in literature that we cannot explain all of them here, however, a summary table of several commonly used measures is shown in *Figure 3.19* [54]-[55]. In addition to the performance measures mentioned earlier, classifiers can also be compared with respect to the following additional aspects: **Speed:** This refers to the computational costs involved in generating and using the given classifier. **Robustness:** This is the ability of the classifier to make correct predictions given noisy data or data with missing values. Robustness is typically assessed with a series of synthetic data sets representing increasing degrees of noise and missing values. **Scalability:** This refers to the ability to construct the classifier efficiently given large amounts of data. Scalability is typically assessed with a series of data sets of increasing size. **Interpretability:** This refers to the level of understanding and insight that is provided by the classifier or predictor. Interpretability is subjective and therefore more difficult to assess. Decision trees and classification rules can be easy to interpret, yet their interpretability may diminish as their complexity increases.



		Predicted condition			
	Total population $= P + N$	**Positive (PP)**	**Negative (PN)**	Informedness, bookmaker informedness (BM) $= TPR + TNR - 1$	Prevalence threshold (PT) $= \frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
Actual condition	**Positive (P)**	**True positive (TP),** **hit**	**False negative (FN),** **type II error, miss,** **underestimation**	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{TP}{P} = 1 - FNR$	False negative rate (FNR), miss rate $= \frac{FN}{P} = 1 - TPR$
	Negative (N)	**False positive (FP),** **type I error, false alarm,** **overestimation**	**True negative (TN),** **correct rejection**	False positive rate (FPR), probability of false alarm, fall-out $= \frac{FP}{N} = 1 - TNR$	True negative rate (TNR), specificity (SPC), selectivity $= \frac{TN}{N} = 1 - FPR$
	Prevalence $= \frac{P}{P + N}$	Positive predictive value (PPV), precision $= \frac{TP}{PP} = 1 - FDR$	False omission rate (FOR) $= \frac{FN}{PN} = 1 - NPV$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$
	Accuracy (ACC) $= \frac{TP + TN}{P + N}$	False discovery rate (FDR) $= \frac{FP}{PP} = 1 - PPV$	Negative predictive value (NPV) $= \frac{TN}{PN}$ $= 1 - FOR$	Markedness (MK), deltaP (Δp) $= PPV + NPV - 1$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
	Balanced accuracy (BA) $= \frac{TPR + TNR}{2}$	F_1 score $= \frac{2 PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$	Fowlkes–Mallows index (FM) $= \sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) $= \sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index $= \frac{TP}{TP + FN + FP}$

Figure 3.19 Performance measures for a binary classifier [54]

Like classification models, there are different ways to measure the performance of regression models where the predicted outcome is numeric. The most used quantitative measures are mean absolute error (MAE), means square error (MSE), root mean square error (RMSE) and coefficient of determination (R^2).

MSE: The mean squared error (MSE) is calculated by squaring the residuals, summing them up and dividing by the number of samples (n). Here y_i is the actual output while $\widehat{y_i}$ is the predicted output.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y_i})^2$$

RMSE: It is calculated by taking square root of MSE.

$$RMSE = \sqrt{MSE}$$

MAE: The mean absolute error (MAE) is the mean value of the absolute values of the residuals.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y_i}|$$

R^2: The coefficient of determination (R^2) is the proportion of the information in the data that is explained by the model. For example, an R^2 value of 0.75 implies that the model can explain three-quarters of the variation in the outcome. R^2 is a measure of correlation not accuracy. There are different definitions for calculating R^2 in literature but the most common definition of R^2 is the ratio between sum of squared residuals (SSR) and total sum of squares (SST).

$$R^2 = \frac{SSR}{SST} = \frac{\sum_{i=1}^{n}(y_i - \widehat{y_i})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Here, $\bar{y}$ is the mean of the actual output i.e.,

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

Visualizations of the model fit e.g., *residual plots*, is also critical in understanding whether a model is a good fit.

Feature Selection/Extraction:

For a given set of features $F = \{f_1, \ldots, f_d\}$, feature selection problem is to find a subset $F' \subseteq F$ that maximizes the performance. While feature problem is to map F to some feature set $F"$ that maximizes the performance as shown in *Figure 3.20*. Determining which features/predictors to use in building a predictive model is very critical especially with high-dimensional (large number of features) dataset. A model with less predictors may be more interpretable, less costly, and more efficient. Some models are naturally resistant to non-informative predictors e.g., tree-based models. Feature selection methods can be categorized as supervised or unsupervised methods. In *unsupervised* methods, the outcome is ignored during the elimination of predictors e.g., removing predictors that have high correlations with other predictors. For *supervised* methods, predictors are specifically selected for the purpose of increasing accuracy by using the outcome to quantify the importance of the predictors. The presence of non-informative variables can add uncertainty to the predictions and reduce the overall effectiveness of a model.

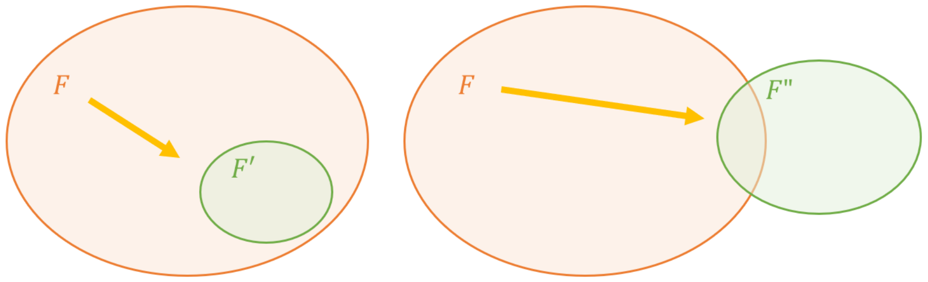


Figure 3.20 Feature Selection (left), Feature Extraction (right)

Most feature reduction techniques can be placed into two main categories. Both have their pros and cons. *Wrapper methods* evaluate multiple models using procedures by adding and/or removing predictors to find the optimal combination that maximizes model performance. *Filter* *methods* evaluate the relevance (e.g., correlation, mutual information etc.) of the features to the outcome and use only those predictors that pass some criterion. In theory the goal is to find an optimal feature-subset that maximizes the model performance. But in real world applications this is usually not possible because it is computationally intractable to search the whole space of possible feature subsets. *Figure 3.21* shows schematics for each of these method for feature subset selection approach.

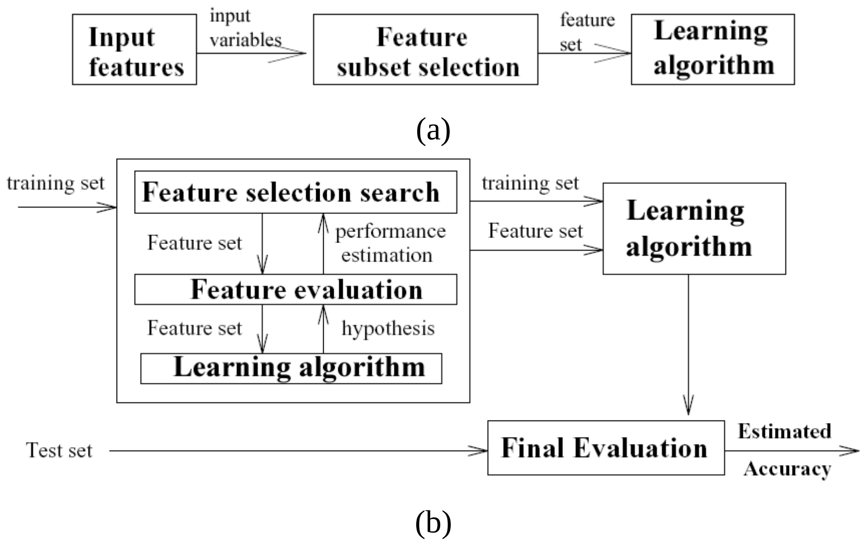


Figure 3.21 Feature Subset Selection using (a) Filter method, (b) Wrapper method

3.1.5 Model Selection:

Although, models can be compared using the performance measures discussed earlier, however, in practice, a range of models are usually equivalent in terms of performance and hence determining the best model can be difficult. In that case, following approach can be used for model selection.

1- Start with several models that are the least interpretable and most flexible, such as boosted trees or support vector machines.

2- Investigate simpler models that are less opaque (e.g., not complete black boxes), such as partial least squares, generalized additive models.
3- Consider using the simplest model that reasonably approximates the performance of the more complex methods.

Moreover, to determine if there is any statistically significant difference in the performance measures e.g., mean error rates of different models, we can employ a test of statistical significance (e.g., *t-test*). A paired t-test can be used to evaluate the hypothesis that the models have equivalent performance on average (null hypothesis) or, analogously, that the mean difference in performance is zero. In addition, this test can also provide some confidence limits for our mean error rates, for example, so that we can make statements like, “Any observed mean will not vary by ± two standard errors 95% of the time for future samples” or “One model is better than the other by a margin of error of ± 4%”.

CHAPTER 4

PROBLEM FORMULATION

The unit commitment problem (UCP) is typically formulated as a multi-period optimization problem [56]. In several other research studies, however, it is also formulated as a single-period optimization problem for simplicity [57]. The main difference being the integration of the inter-temporal constraints, such as ramping rates and minimum up/down times etc., in a multi-period UC problem. In this thesis, for the sake of simplicity and put more emphasis on the proposed approach, we will initially present results for a single-period SC-UC problem. However, the proposed approach works equally well for a multi-period SC-UC problem and results for a realistic multi-period SC-UC problem will be presented afterwards.

4.1 UCP-The Objective Function:

The main objective of a classic UC problem is to minimize the total operational cost (TOC) over a predefined time horizon (T), as given by eq. (4.1). This cost typically consists of two components i.e., production cost (PC) and startup cost (SUC) of all the generating units.

$$min \quad TOC = \sum_{t \in T} \sum_{g \in G} \left(PC_{gt} + SUC_{gt}\right) I_{gt} \qquad \forall g \in G, \forall t \in T \quad (4.1)$$

Here, g ($\in G$) represents a generator in a set of available generators (G) and t is a time interval (e.g., an hour) in a prespecified time horizon T (e.g., a day). The production cost (PC_g) of a generator is usually modeled either as a quadratic or a piecewise linear function of its output power (P_g) as follows:

Quadratic cost: $PC_g = a_g + b_g P_g + c_g P_g^2$

Piecewise linear cost: $PC_g = a_g + \sum_k b_{kg} P_{kg}$

Here a_g, b_g and c_g are cost coefficients of a generator g and b_{kg} is the cost coefficient of the k^{th} segment of its power generation cost curve. The startup cost (SUC_g) is usually modeled by two distinct values (hot-startup-cost, $HSUC_g$, and cold-startup-cost, $CSUC_g$) and is a function of down time of a generator, as follows:

$$SUC_g = \begin{cases} CSUC_g & off\ time > min.down\ time + cold\ start\ time \\ HSUC_g & otherwise \end{cases}$$

4.2 UCP-Constraints:

There are several constraints that need to be satisfied while minimizing the operational cost. Some of the commonly used constraints in formulating a SC-UC are discussed here.

<u>Nodal Power Balance</u>: At each interval t of the time horizon T, the power should be balanced at each node of the power system. While formulating a UCP as a mix-integer programming (MIP), usually the line flows are approximated by DC power flow assumptions. Hence, the nodal power balance constraints are expressed as follow:

$$\sum_{g \in G_b} P_{gt} = d_{bt} + \sum_{l \in L_b} P_{lt} \qquad \forall b \in B, \forall t \in T \ (4.1a)$$

Here P_{gt} is the power generated by generator g in interval t, d_{bt} is the forecasted net power demand at bus b in interval t, P_{lt} is the power injection from bus b into the line l in interval t. The sets G_b and L_b respectively represent the generators and lines connected with bus b. The set B contains all buses of a system.

<u>Generator Power Limits</u>: These constraints ensure that, if committed, the power generated by each generator does not exceed its maximum ($\overline{P_g}$) and minimum ($\underline{P_g}$) power generation limits. Here, a binary decision variable I_{gt} is the commitment status (i.e., 1 = on, 0 = off) of a generator g in an interval t.

$$\underline{P_g} I_{gt} \ \leq \ P_{gt} \leq \ \overline{P_g} I_{gt} \qquad \forall g \in G, \forall t \in T \ (4.1b)$$

<u>Line Capacity</u>: There is a maximum power that each transmission line can carry. So, to ensure that the line flow (P_l) of a line l does not exceed its maximum power carrying capacity ($\overline{P_l}$), we include these constraints. The set L represents all the transmission lines in the system. The line flows are approximated using DC power flow assumptions. X_l is the reactance of the line l and δ_l is the voltage angle difference of the two busses connected at the two ends of the line l.

$$-\overline{P_l} \ \leq \ P_l = \frac{\delta_l}{X_l} \leq \ \overline{P_l} \qquad \forall l \in L \ (4.1c)$$

<u>Spinning Reserve</u>: The total available power from all the committed generators which is equal to the online maximum generation capacity minus the total generated power is defined as spinning reserve. In each interval t, the spinning reserve must be higher than or equal to the minimum spinning reserve requirements (R_t). R_t is usually described as a percentage of the total demand in interval t.

$$\sum_{g\in G} R_{gt}\, I_{gt} \geq R_t \qquad \forall t \in T \ (4.1d)$$

<u>Minimum Up/Down Time</u>: A generator cannot change its status instantaneously once it is committed or decommitted. A switched off/on generator cannot be turned on/off instantaneously until elapsing some time called minimum down/up time. The minimum up/down time constraints can be represented as

$$\left(T_{g(t-1)}^{on} - T_g^{up}\right)\left(I_{g(t-1)} - I_{gt}\right) \geq 0 \qquad \forall g \in G, \forall t \in T \ (4.1e)$$

$$\left(T_{g(t-1)}^{off} - T_g^{down}\right)\left(I_{g(t-1)} - I_{gt}\right) \geq 0 \qquad \forall g \in G, \forall t \in T \ (4.1f)$$

Here T_g^{up} and T_g^{down} are minimum up and down time limits while $T_{g(t-1)}^{on}$ and $T_{g(t-1)}^{off}$ are continuous on/off times in interval $(t-1)$ of a generator g, respectively.

<u>Ramp Up/Down Limits</u>: The output power variation for consecutive periods of a committed online generator must satisfy its ramp rate constraints as follows.

$$-R^{down} \leq P_{gt} - P_{g(t-1)} \leq R^{up} \qquad \forall g \in G, \forall t \in T (4.1g)$$

<u>Binary and Non-negativity</u>: The UCP is a mix-integer non-convex optimization problem. It has both binary (I_{gt}) and continuous (P_{gt}) decision variables which are usually non-negative.

$$I_{gt} \in \{0,1\} \qquad \forall g \in G, \forall t \in T (4.1h)$$

$$P_{gt} \geq 0 \qquad \forall g \in G, \forall t \in T (4.1i)$$

4.3 Single Period SC-UC Problem:

Consider a power system having N_B number of buses, N_G number of generators and N_L number of transmission lines. Sets $B = \{1,2,3,\dots,N_B\}$, $G = \{1,2,3,\dots,N_G\}$, $L = \{1,2,3,\dots,N_L\}$

and $D = \{1,2,3,\dots,N_D\}$ represent the sets of buses, generators, transmission lines and nodal power demands respectively. For each generator $g \in G$, let a binary decision variable $I_g \in \{0,1\}$, represents the on/off status of the generator g and a continuous decision variable P_g, indicates the amount of power being generated by it. Let PC_g represents the power production cost function of a generator g. Assuming a quadratic cost function (i.e., $PC_g = a_g + b_g\, P_g + c_g\, P_g^2$) and neglecting the start-up cost, a typical single-period SC-UC problem can be formulated as given by eq. (4.2).

$$\underset{I_g, P_g, R_g, \delta_b}{minimize:} \quad \sum_{g \in G} I_g . PC_g\left(P_g\right) \tag{4.2}$$

$$subject\ to: \quad \sum_{g \in G_b} P_g = d_b + \sum_{l \in L_b} P_l \qquad \forall b \in B \quad (4.2a)$$

$$-\overline{P_l} \le P_l = \frac{\delta_l}{X_l} \le \overline{P_l} \qquad \forall l \in L \quad (4.2b)$$

$$\delta_{b_{ref}} = 0 \qquad (4.2c)$$

$$\underline{P_g} I_g \le P_g \le \overline{P_g} I_g \qquad \forall g \in G \quad (4.2d)$$

$$\sum_{g \in G} R_g \ge R \qquad (4.2e)$$

$$P_g + R_g \le \left(\overline{P_g} - \underline{P_g}\right) I_g \qquad \forall g \in G \quad (4.2f)$$

$$I_g \in \{0,1\} \qquad \forall g \in G \quad (4.2g)$$

$$P_g \ge 0 \qquad \forall g \in G \quad (4.2h)$$

Here $\delta_{b_{ref}}$ is the phase angle of the reference bus of the system. Constraints $(4.2a)$ through $(4.2e)$ are sets of constraints enforcing nodal power balance at each bus b, maximum power carrying capacity of each line l, phase angle of the reference bus, maximum and minimum power output limits of each generator g and minimum spinning reserve requirements respectively. Constraints sets $(4.2f)$ though $(4.2h)$ describe the relation between P_g and R_g variables and binary and non-negative behavior of variables I_g and P_g respectively.

4.4 Stage-1: Commitment Decisions (Algo-1 & Algo-2)

In the first stage, predictive classification models, based on supervised learning methods, are employed to predict the on/off status of each of the generators. The models are trained using either historical or synthetic data. The training dataset contains labeled examples of solved instances. A lot of supervised learning methods are available in the literature and selection of any of the methods depends on the application and performance of the method for that specific application. One method which performs well for one application may not perform well in another application. A usual practice is to try multiple different learning methods and select the one which performs best. A combination of multiple learning methods, i.e., *Ensemble*, can also be used. Each method constructs a function, based on the training dataset, which is then used to label new instances.

Initially, we employed several learning algorithms e.g., K-Nearest Neighbor (KNN), Decision Tree (DT), Random Forest (RF), Linear Discriminant Analysis (LDA), Support Vector Machine (SVM), Generalized Linear Model (GLM), Extreme Gradient Boosting (XGB) and Neural Network (NNET). Based on the performance metrics discussed in chapter 3, the model which provided the best overall classification performance was selected. Initially, we employed multiple learning algorithms for stage-1 classification e.g., Random Forest (RF), Extreme Gradient Boosting (XGB), Support Vector Machine (SVM), K-nearest Neighbor (KNN) and Neural Network (NNET). The method which provided the best classification performance, was finally used.

Performance Evaluation: Initially, we split the dataset into train/test datasets with 80/20% split. Train set was used for training and test set was used for validation and final selection. For resampling, repeated 10-fold cross-validation was used repeated 5 times. Top 3 models were initially selected based on resampling results. Then based on their performance for the test set, the best model was chosen for prediction.

Performance Optimization: For optimizing the performance of the individual models, we used hyperparameter tuning and feature selection techniques. Random grid search was used for hypermeter tuning and feature subset selection was done using a wrapper method with forward selection.

Performance Measures: The most used metric is accuracy. However, it does not provide insights into the classification errors and can be misleading, especially, for imbalanced datasets. Therefore, other widely used measures, obtained from the confusion matrix (shown in *Figure 4.1*), were also considered e.g., precision, recall/sensitivity, specificity, F1-score, area under the receiver operating characteristic (ROC) curve (AUC) etc. The selection of one or the other depends on the application and preference. Detailed explanation of these metrics can be found in chapter 3.

		Actual	
		Positive	Negative
Predicted	Positive	**True Positive (TP)**	**False Positive (FP)**
	Negative	**False Negative (FN)**	**True Negative (TN)**

Figure 4.1 Confusion Matrix for a Binary Classifier

4.5 Stage-2: Production Decisions (Algo-1)

Once the commitments decisions of all the generators are predicted in stage 1 i.e., the values of the binary decision variables $\boldsymbol{I_g}$ in eq. (4.2) are specified, the UC problem becomes an optimal dispatch problem, however, with some security constraints. Hence, a complex MIQP given in eq. (4.2) is converted to a comparatively much easier to solve QP problem with no. of decision variables reduced to half of the original variables and less constraints. Therefore, in stage-2 of the proposed approach, a SC-ED, given in eq. (4.3), is solved to obtain the production decisions $\boldsymbol{P_g}$ of the committed generators.

$$\underset{P_g}{minimize:} \quad \sum_{g\in G^c} F_g(P_g) \tag{4.3}$$

$$subject\ to: \quad \sum_{g\in G_b^c} P_g = d_b + \sum_{l\in L_b} P_l \qquad \forall b \in B \quad (4.3a)$$

$$-\overline{P_l} \le P_l = \frac{\delta_{bl}}{X_l} \le \overline{P_l} \qquad \forall l \in L \quad (4.3b)$$

$$\delta_{b_{ref}} = 0 \qquad (4.3c)$$

$$\underline{P_g} \le P_g \le \overline{P_g} \qquad \forall g \in G^c \quad (4.3d)$$

$$\sum_{g\in G^c} R_g \ge R \qquad (4.3e)$$

$$P_g + R_g \le \left(\overline{P_g} - \underline{P_g}\right) \qquad \forall g \in G^c \quad (4.3f)$$

$$P_g \ge 0 \qquad \forall g \in G^c \quad (4.3g)$$

Here, $\boldsymbol{G^c}$ is the predicted set of committed generators obtained from stage 1. A schematic diagram of the proposed algorithm-1 is shown in *Figure 4.2*.

4.6 Stage-2: Production Decisions (Algo-2)

In stage-2 of algorithm-2, predictive regression models are employed to estimate the output powers P_g of the committed generators. Hence, the MIP optimization problem formulated earlier to solve the UC, is completely replaced by predictive models. Like stage-1 classification, we employed multiple learning methods like KNN, RF, XGB, NNET, Linear and Logistics regression models and used the best model in terms of performance. The model evaluation procedures for regression models are the same as discussed earlier, however, the performance metrics are different. The most used metrics for evaluating the performance of a regression model are mean absolute error (MAE), mean square error (MSE), root mean square error (RMSE) and coefficient of determination (R-squared).

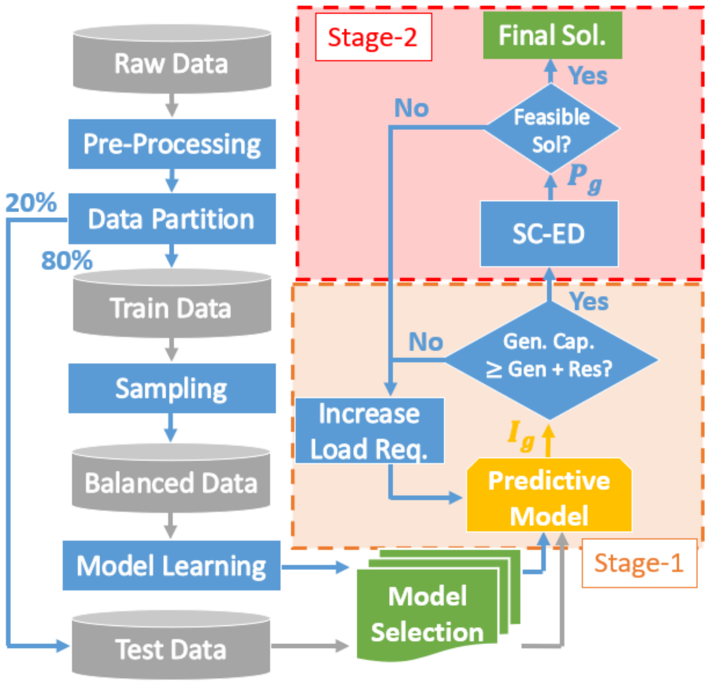


Figure 4.2 Two-Stage Proposed Algorithm for SCUC

4.7 Stage-3: Post Processing (Algo-2)

Since stage-1 and stage-2 models discussed earlier are trained on historical/synthetic data containing solved instances which satisfy system constraints, it is inherent for these data-driven approaches to predict solutions which would somewhat satisfy the system constraints as well, however, due to the embedded uncertainty and prediction error in predictive modeling techniques, it is mandatory to confirm that all system constraints are satisfied to ensure secure operation of the power system. Hence, a third stage for post processing and polishing the predicted solution is needed to make sure that all the constraints are satisfied.

Spinning Reserves: To ensure that spinning reserve constraints are satisfied, we check if the total committed generation capacity ($\overline{\mathrm{P}_{\mathrm{G}}^{\mathrm{c}}}$) is higher than total demand (D) plus spinning reserve requirement (R) as given by (4.4c). If it's less, then the total power demand is assumed to have increased, and the commitment decisions are predicted again to increase the predicted committed capacity.

$$\overline{\mathrm{P}_{\mathrm{G}}^{\mathrm{c}}} = \sum_{g \in G^c} \overline{P_g} \tag{4.4a}$$

$$D = \sum_{b \in B} d_b \tag{4.4b}$$

$$\overline{\mathrm{P}_{\mathrm{G}}^{\mathrm{c}}} \geq D + R \tag{4.4c}$$

Generator Power Limits: The production decisions obtained from stage 2 prediction are truncated by $\overline{P_g}$ and $\underline{P_g}$ to satisfy generators maximum and minimum power limits as follows:

$$P_g \in [\underline{P_g}, \overline{P_g}] \qquad \forall g \in G^c \; (4.5)$$

Power Balance: To balance the total generation (P_G) with the total demand (D), we calculate the generation load imbalance (GLI), then update the production decisions while satisfying the maximum and minimum power generation limits of the committed generators as follows:

$$GLI = P_G - D = \sum_{g \in G^c} P_g - \sum_{b \in B} d_b \tag{4.6a}$$

$$P_g' = P_g \left(1 - \frac{GLI}{P_G}\right) = P_g \left(\frac{D}{P_G}\right) \in [\underline{P_g}, \overline{P_g}] \qquad \forall g \in G^c \quad (4.6b)$$

The decision variables (P_g) which violate $\underline{P_g}$ or $\overline{P_g}$ limits are truncated at its limits and the remaining P_g variables are updated using (4.6b).

Line Capacity Limits: To ensure that none of the line capacity limits are violated, we re-dispatch the generator powers (P_g) so that all the line flows are within their capacity limits. For rescheduling the generator powers, we bring small variations in generator powers (ΔP_g) using Generation Shift Factors and Eigen Value Decomposition discussed below.

1) <u>Generation Shift Factors (GSF)</u>: Generation Shift Factors are linear sensitivity factors which describe the change in power flow on a branch for a given change in bus power injections. It gives the fraction of a change in injection at bus b that appears on a line l [58]. From the DC load flow equations, the effect of varying the bus power injections ($\Delta \boldsymbol{P}$) on the bus voltage angles ($\Delta \boldsymbol{\delta}$) is given by eq. (4.7).

$$\Delta \boldsymbol{\delta} = [B']^{-1} \Delta \boldsymbol{P} \tag{4.7}$$

B' is the DC power flow matrix. Similarly, the effect of varying the bus voltage angles ($\Delta \boldsymbol{\delta}$) on the line flows ($\Delta \boldsymbol{P_{line}}$) is given by eq. (4.8).

$$\Delta \boldsymbol{P_{line}} = (D_X A_N) \Delta \boldsymbol{\delta} \tag{4.8}$$

D_X is a diagonal square matrix in which diagonal entries are equal to the negative of the susceptance ($1/X_l$) of the corresponding transmission line and A_N is the node-arc-incidence matrix of the power network describing connections of the lines in the network. By using eqs. (4.7) and (4.8), we can find the effect of varying bus power injections on the line flows as follows.

$$\Delta \boldsymbol{P_{line}} = T \Delta \boldsymbol{P} \tag{4.9}$$

Here T is the matrix of generation shift factors given by eq. (4.10) and $\Delta \boldsymbol{\delta}$**,** $\Delta \boldsymbol{P}$ and $\Delta \boldsymbol{P_{line}}$ are vectors of variations in bus voltage angles, bus power injections and line flows respectively.

$$T = (D_X A_N)[B']^{-1} \tag{4.10}$$

$$\Delta \boldsymbol{\delta} = [\Delta \delta_1 \dots \Delta \delta_{NB}]^T \tag{4.11}$$

$$\Delta \boldsymbol{P} = [\Delta P_1 \ldots \Delta P_{NB}]^T \tag{4.12}$$

$$\Delta \boldsymbol{P_{line}} = [\Delta P_{l1} \ldots \Delta P_{lNL}]^T \tag{4.13}$$

Using eq. (4.9), we can find the change in line flows for a given change in bus power injections ($\Delta \boldsymbol{P}$) i.e., generator injections ($\Delta \boldsymbol{P_g}$).

2) Eigen Value Decomposition (EVD): When a vector $\boldsymbol{x}$ is multiplied with a matrix A as $\boldsymbol{y} = A\boldsymbol{x}$, then it changes its magnitude as well as its direction. However, when eigenvectors ($\boldsymbol{v_{\lambda i}}$) of matrix A are multiplied with A, they are only squeezed or stretched while keeping same direction [59-63], as given by eq. (4.14).

$$A\boldsymbol{v}_{\lambda_i} = \lambda_i \boldsymbol{v}_{\lambda_i} \tag{4.14}$$

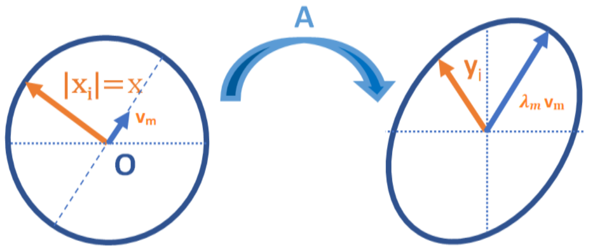


Figure 4.3 Manipulation of $\{\boldsymbol{x_i}\}$ *with* A

Manipulation of a set of all possible vectors $\{\boldsymbol{x_i}\}$, having same magnitudes, with A is shown in *Figure 4.3*. It shows that if $\boldsymbol{x}$ lies in the span of $\boldsymbol{v_m}$ (eigenvector of highest eigenvalue, λ_m), then $\boldsymbol{y}$ would be maximized i.e., if $\boldsymbol{x} = \boldsymbol{v_m}$, then $\boldsymbol{y} = A\boldsymbol{v_m} = \lambda_m \boldsymbol{v_m}$. Hence, to maximize the effect of $\Delta \boldsymbol{P}$ on a specific line l , we can use the weighted $L2$ norm of $\Delta \boldsymbol{P_l}$. Using eq. (4.9), we can write

$$\Delta P_{\boldsymbol{l}} = \Delta \boldsymbol{P}_{\boldsymbol{line}}^T W \Delta \boldsymbol{P_{line}} = \Delta \boldsymbol{P}^{\boldsymbol{T}} N \Delta \boldsymbol{P} \tag{4.15}$$

Here, $N = T^T W T$ is a real symmetric matrix and W is a weight matrix to select the line(s) whose flow(s) we want to control. Based on previous discussion, ΔP_l would be maximized if $\Delta \boldsymbol{P}$ lies in the span of the eigenvector ($\boldsymbol{v_m}$) of the highest eigenvalue (λm) of N i.e., if $\boldsymbol{\Delta P} = \alpha \boldsymbol{v_m}$, then

$\Delta P_l^{max} = \alpha^2 \lambda_m$. Scalar α is the design parameter which would give required change in corresponding line flow. It can be calculated as $\alpha = \pm\sqrt{\frac{\Delta P_{l,req}}{\lambda_m}}$. However, to keep $\sum_{\forall b} \Delta P_b = 0$, to maintain generation load balance in the obtained solution, we only use the signs of the elements of $\boldsymbol{v_m}$ to get the direction of required variations (i.e., increase/decrease) in generators' output powers which will have the maximum effect on ΔP_l, keeping generation and load balanced. The schematic diagram of proposed three stage algorithm is shown in *Figure 4.4*.

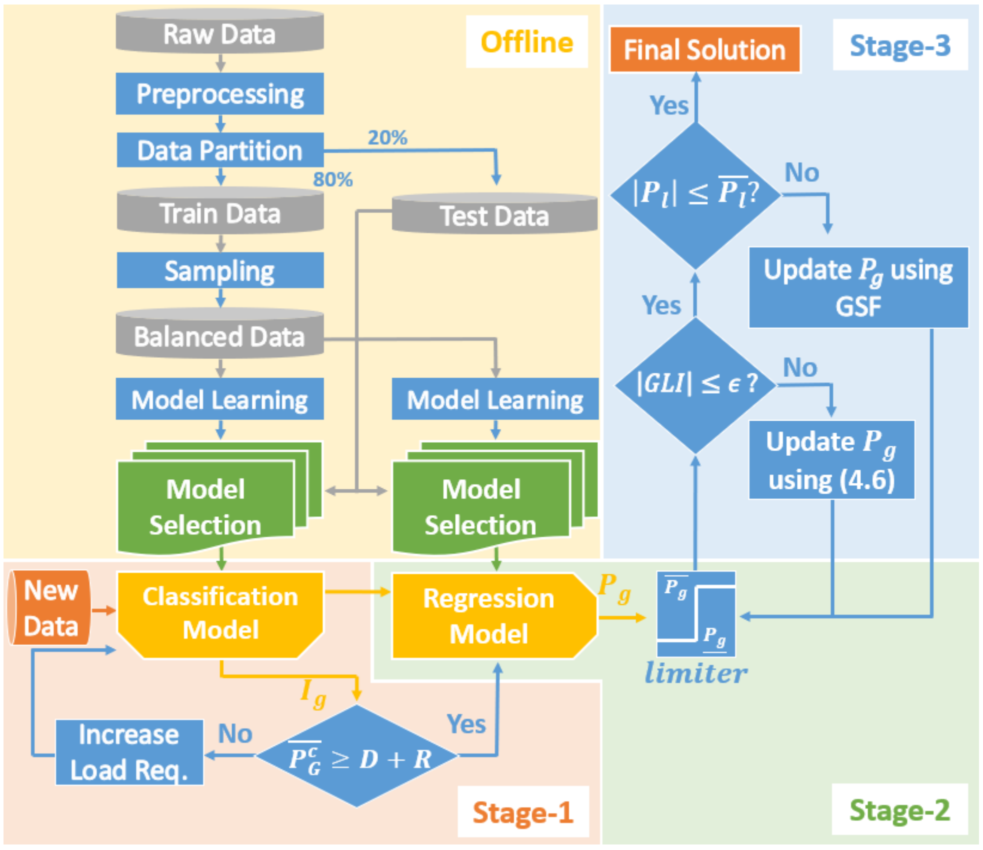


Figure 4.4 Three-Stage Proposed Algorithm

CHAPTER

5

CASE STUDIES AND SIMULATION RESULTS

To see the performance of the proposed approaches, we will study several power systems e.g., a 4-bus-3-machine system and IEEE 9-bus, 39-bus, 118-bus, and South Carolina 500-bus systems. All the simulations are performed using MATLAB and R-Studio on an AMD 16-core machine having 64GB of DDR4.

5.1 Case Study 1: 4-Bus 3-Machine System

5.1.1 System Data

This small system is used as an illustrative example to study the performance of the presented approaches. The system has four buses, three generators, two loads and five transmission lines. The single line diagram of the system is shown in *Figure 5.1*. The system data is given in *Table 5.1*.

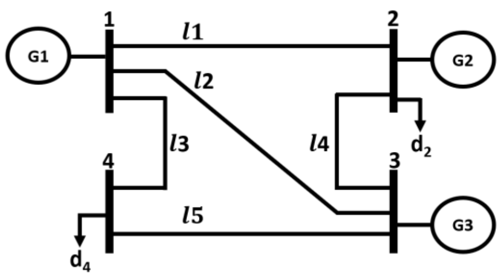


Figure 5.1 Single Line Diagram (3-Machine System)

Table 5.1 System Data (3-Machine System)

Base MVA	100
Base kV	138
Lines pu Reactances	0.25
Lines MW Capacities	200
Generators MW Limits	0 - 500
mean MW Loads (bus2/bus3)	300/200
minimum Spinning Reserve requirement (R)	30% of total load

5.1.2 MIQP Problem Formulation

Initially, the UC problem for the 4-bus system is formulated as a single-period MIQP problem assuming quadratic cost curves for the generators ($i.e.\ PC_g(P_g) = a_g + b_gP_g + c_gP_g^2$, where a_g, b_g and c_g are the cost coefficients) as presented in chapter 4. Using the system data given in *Table 5.1*, the problem is formulated as given below.

$$\underset{I_1,I_2,I_3,P_1,P_2,P_3,R_1,R_2,R_3,\delta_1,\delta_2,\delta_3,\delta_4}{minimize:} \quad : \quad I_1(a_1 + b_1P_1 + c_1P_1^2) + I_2(a_2 + b_2P_2 + c_2P_2^2)$$
$$+I_3(a_3 + b_3P_3 + c_3P_3^2)$$

$$subject\ to: \quad : \quad nodal\ power\ balance$$
$$12\delta_1 - 4\delta_2 - 4\delta_3 - 4\delta_4 = P_1$$
$$-4\delta_1 + 8\delta_2 - 4\delta_3 = P_2 - d_2$$
$$-4\delta_1 - 4\delta_2 + 12\delta_3 - 4\delta_4 = P_3$$

$$-4\delta_1 - 4\delta_3 + 8\delta_4 = -d_4$$

$$reference\ bus\ angle$$

$$\delta_1 = 0$$

$$line\ limits$$

$$-0.5 \leq \delta_1 - \delta_2 \leq 0.5$$

$$-0.5 \leq \delta_1 - \delta_3 \leq 0.5$$

$$-0.5 \leq \delta_1 - \delta_4 \leq 0.5$$

$$-0.5 \leq \delta_2 - \delta_3 \leq 0.5$$

$$-0.5 \leq \delta_3 - \delta_4 \leq 0.5$$

$$generator\ power\ limits$$

$$0 \leq P_1 \leq 5I_1$$

$$0 \leq P_2 \leq 5I_2$$

$$0 \leq P_3 \leq 5I_3$$

$$reserve\ requirement$$

$$R_1 + R_2 + R_3 \geq 0.3(d_2 + d_4)$$

$$relation\ between\ R_g and\ P_g$$

$$P_1 + R_1 \leq 5I_1$$

$$P_2 + R_2 \leq 5I_2$$

$$P_3 + R_3 \leq 5I_3$$

$$binary\ variables$$

$$I_1, I_2, I_3 \in \{0,1\}$$

Here, the angles (δ_b) are in radians while all the other non-binary variables are in per unit.

5.1.3 Data Generation

To train the predictive model, the dataset is generated by solving the above MIQP problem using IBM CPLEX toolbox for MATLAB (this solver is used as a base line method for comparing the performance of the presented approach). A data set of 1000 samples, representing different scenarios of power demands and price signals, is generated assuming normal distributions for the loads (d_2 , d_4) and the cost coefficients (a_g , b_g , c_g). *Figure 5.2* shows the histograms of the MW demands at bus 2 (i.e., d_2) and bus 4 (i.e., d_4) in the generated simulated dataset. These demands are randomly generated assuming normal distributions for power demands at the two buses with mean values of 300/200 MW and standard deviation values of 75/55 MW respectively. The distributions of the cost coefficients a_g, b_g and c_g in the generated dataset are shown in *Figure 5.3*. The spread of the cost curve resulted from these cost coefficients is shown in *Figure 5.4*.

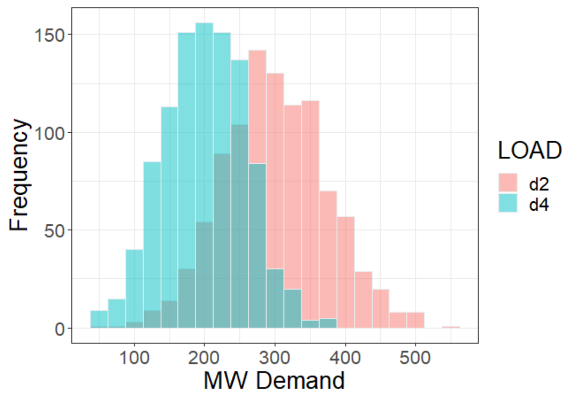


Figure 5.2 Bus Power Demands Distributions (3-Machine System)

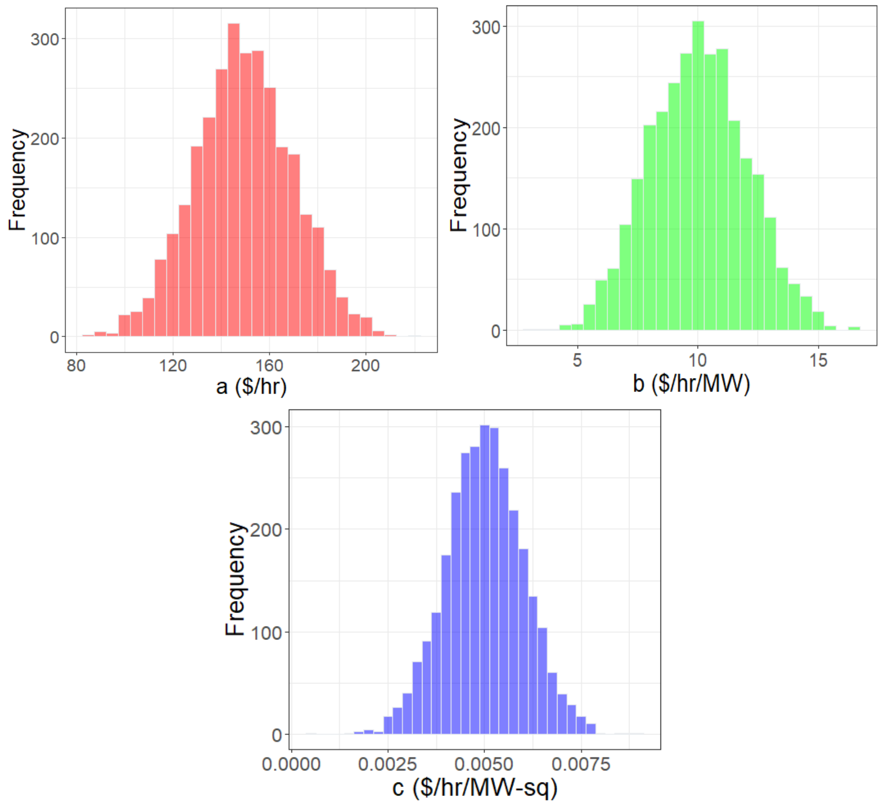


Figure 5.3 Distribution of Cost Coefficients (3-Machine System)

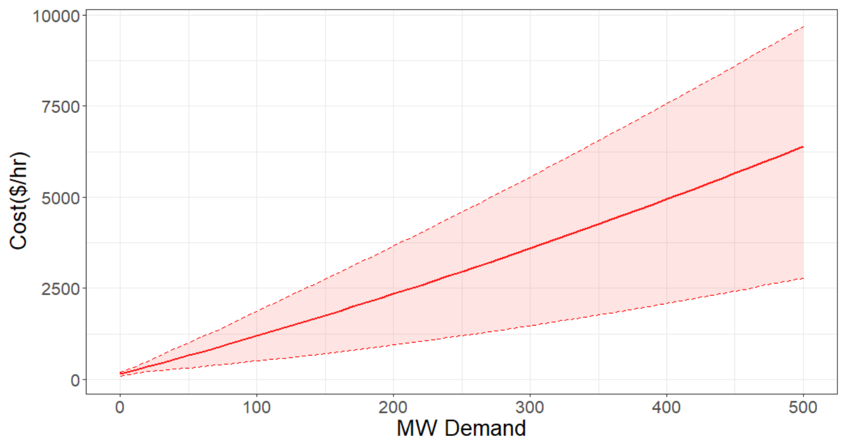


Figure 5.4 Spread of Generators' Cost Curves (3-Machine System)

Table 5.2 Location and Dispersion of the Normal Distributions (3-Machine System)

Variable	Mean	St. Dev.
d2 (MW)	300	75
d4 (MW)	200	55
a ($/hr)	150	20
b ($/hr/MW)	10	2
c ($/hr/MW-sq)	0.005	0.001

The *location* and *dispersion* values of the normal distributions used for generating these load demands, and cost coefficients are given in *Table 5.2*. To label the generated dataset, the instances are solved through simulations by solving the SC-UC problem presented earlier. MATLAB (R2021a) and IBM CPLEX toolbox for MATLAB are used to solve the MIQP problem. This data is then used for training and testing of the learning algorithms using R-Studio. *Table 5.3* shows few of the solved instances in the dataset. Here, the angles are given in degrees.

Table 5.3 A Sample of the Solved Instances (3-Machine System)

index	**cost**	**d2**	**d4**	**a1**	**a2**	**a3**
1	7034.84	357	235	132	131	150
2	5143.75	255	266	166	119	154
3	4678.61	222	226	127	139	141
b1	**b2**	**b3**	**c1**	**c2**	**c3**	**I1**
11.45	12.13	9.41	0.00278	0.00513	0.00433	1
8.5	10.31	7.55	0.00560	0.00424	0.00463	1
10.96	10.1	8.3	0.00602	0.00384	0.00382	0
I2	**I3**	**P1**	**P2**	**P3**	**del1**	**del2**
0	1	217.04	0	374.96	0	-22.74
0	1	189.35	0	331.65	0	-15.72
1	1	0	105.7	342.3	0	-2.2
del3	**del4**	**R1**	**R2**	**R3**	**I123**	
5.66	-14	282.96	0	125.04	5	
5.1	-16.5	310.65	0	168.35	5	
12.26	-10.06	0	394.3	157.7	3	

The binary variables (I_1, I_2, I_3) indicating the on/off status of the generators are converted into an equivalent decimal number (I_{123}) e.g., $(I_1, I_2, I_3) = (0,1,1) = 3$ with I_1 and I_3 being the MSB and LSB respectively. Hence, it becomes a multi-level classification problem having 7 distinct levels in the dataset as shown in *Table 5.3* above.

The objective of the predictive modeling is to predict the decision variables I_1, I_2, I_3, P_1, P_2, and P_3 using variables d_2, d_4, a_1, a_2, a_3, b_1, b_2, b_3, c_1, c_2, and c_3. Hence, variables I_1, I_2, I_3 (or equivalently I_{123}) are target variables and d_2, d_4, a_1, a_2, a_3, b_1, b_2, b_3, c_1, c_2, c_3 are predictor variables for stage 1 prediction. While for stage 2 prediction (algorithm 2), P_1, P_2, P_3 are target variables and d_2, d_4, a_1, a_2, a_3, b_1, b_2, b_3, c_1, c_2, c_3, and I_1, I_2, I_3 (obtained from stage 1 prediction) are predictor variables.

5.1.4 Data Preprocessing

As a first step, we look for any skewness in the distributions of the predictor variables. By looking at the frequency distributions of the variables in *Figure 5.2* and *Figure 5.3*, we can see that the variables are very much normally distributed (that's obvious because there were generated synthetically from normal random distributions, however, in real world data, they can be skewed), and it can also be confirmed from the near-zero skewness coefficients of the distributions of the variables given in *Table 5.4*.

Table 5.4 Skewness of Predictor Variables (3-Machine System)

Variable	Skewness	Variable	Skewness
d2	0.089	**b2**	0.039
d4	-0.006	**b3**	0.016
a1	-0.063	**c1**	-0.072
a2	0.072	**c2**	0.075
a3	0.047	**c3**	-0.04
b1	0.061		

Next, we will normalize the numeric predictors (i.e., d2 through c3 in *Table 5.3*) using min-max normalization. This will normalize all the predictor values to a common scale of [0-1]. It maps an i^{th} value x_{Pi} of a predictor P to a new value x'_{P_i} as given by eq. (5.1). It linearly transforms the original data values between 0 and 1 and preserves the relationship among the original data values. Here, min_P and max_P are the minimum and maximum values of a predictor P in the dataset.

$$x'_{P_i} = \frac{x_{P_i} - min_P}{max_P - min_P} \tag{5.1}$$

After that, we will check if there are any between-predictor correlations to avoid any instabilities in the trained models. This can be done by calculating all the pairwise correlations of the predictors. Alternatively, we can also calculate the correlation matrix and draw the correlation plot of the predictors to check for any significant dependencies between predictors. *Figure 5.5* shows all the pairwise correlations of the variables and it can be seen in the correlation plot that there are no significant between-predictor correlations.

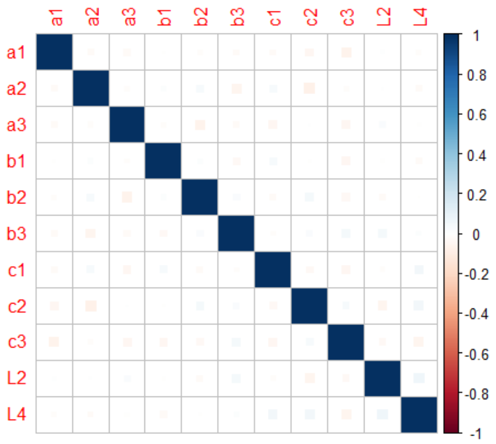


Figure 5.5 Pairwise Correlations of the Transformed Predictors (3-Machine System)

We can also confirm absence of any significant collinearity by locking at the scree plot from Principal Component Analysis (PCA) of the predictors. *Figure 5.6* shows the scree plot of the predictors. We can see that there is no sharp drop in the amount of variability explained by the PCA components.

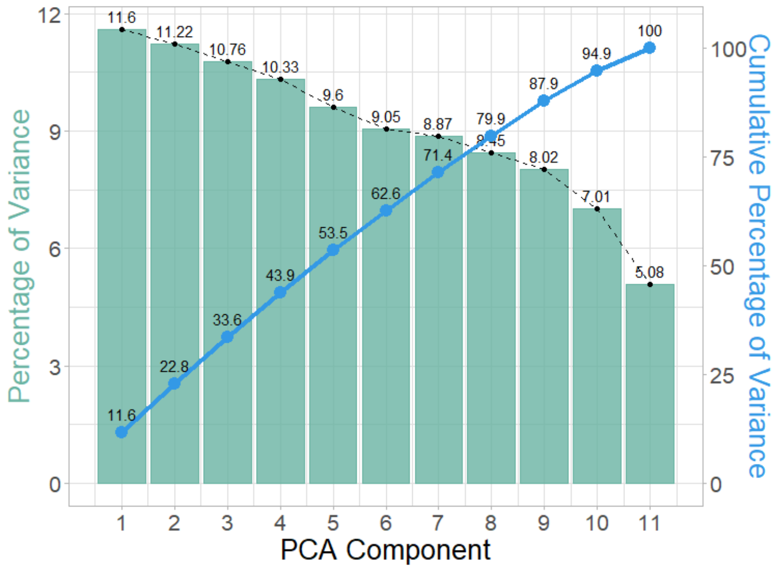


Figure 5.6 A scree plot from PCA analysis of the predictors (3-Machine System)

5.1.5 Data Splitting

As discussed in chapter 3, there are several methods which are used to evaluate the performance of a predictive model during training and testing phase e.g., holdout, cross-validation, random sampling, bootstrap etc. Here we have used a combination of holdout and cross-validation. Initially, the dataset is divided into 80/20% train/test split using stratified random sampling to preserve the distributions of the classes in the train and the test datasets. The 80% train dataset is used for training of the learning algorithms and top few models are initially selected based on their performance using 10-fold cross-validation repeated 5 times. After that, the selected models are evaluated again using the 20% test dataset to obtain their performance for the unseen data and the best model is selected for prediction as shown in *Figure 5.7*.

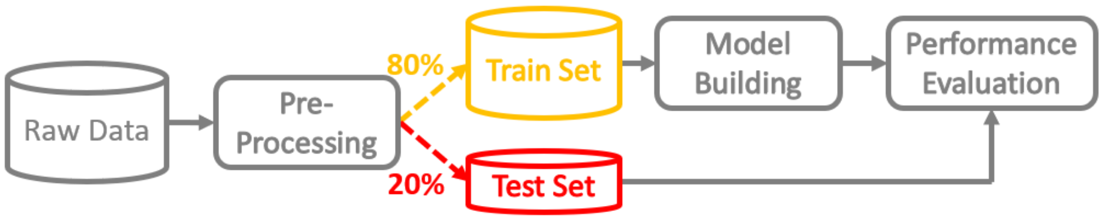


Figure 5.7 Data Splitting into Train and Test Datasets

5.1.6 Stage 1: Commitment Decisions (Algo-1 & Algo-2)

In stage-1 of the prediction, we will use several commonly used predictive models for classification including KNN, LDA, DT, RF, SVM, NNET and XGB.

Hyperparameter Tuning

There are one or more tuning parameters in all these algorithms. The hyperparameter tuning results for the trained models are shown in *Figure 5.8* through *Figure 5.13* along with the best tunned parameters values. The prediction accuracy has been used as the performance metric for model tuning.

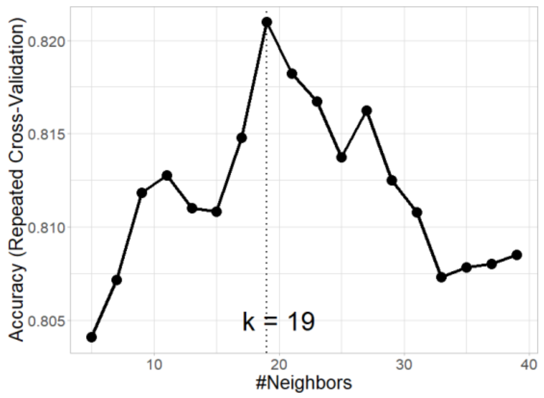


Figure 5.8 Tuning of parameter k to train KNN for predicting I_{123} (3-Machine System)

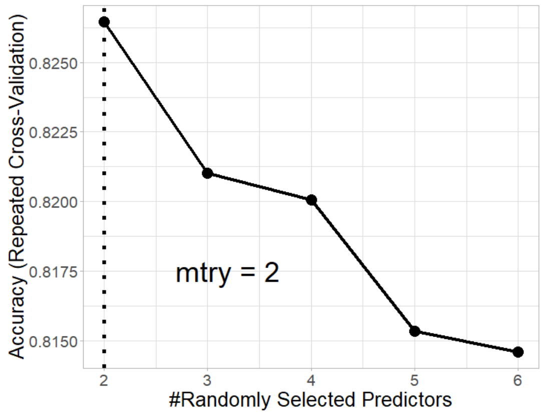


Figure 5.9 Tuning of parameter mtry to train RF for predicting I_{123} (3-Machine System)

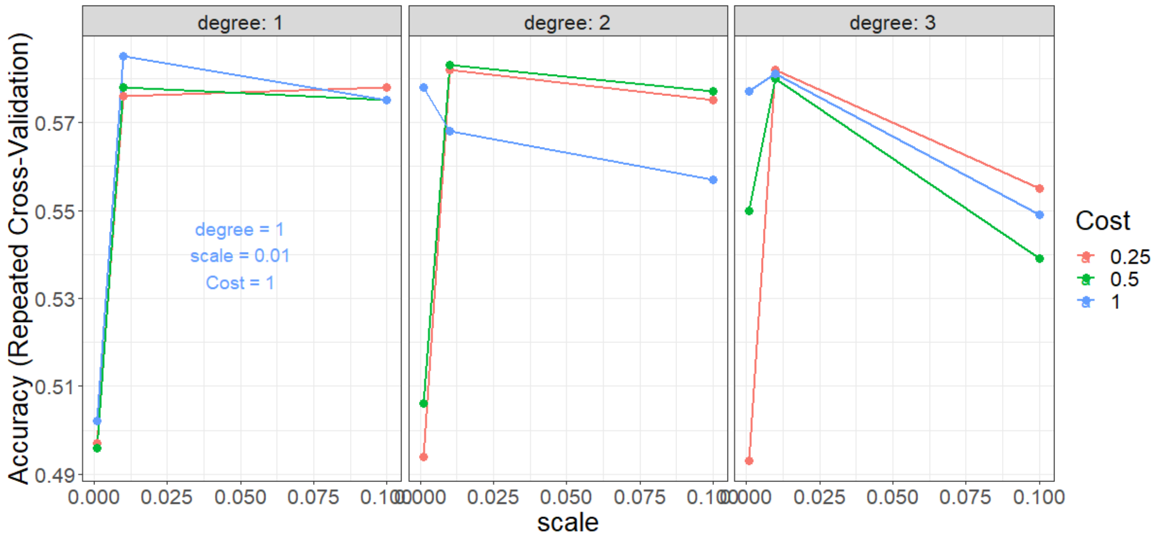


Figure 5.10 Tuning of parameters degree, scale, and cost to train SVM-Poly to predict I_{123} (3-Machine System)

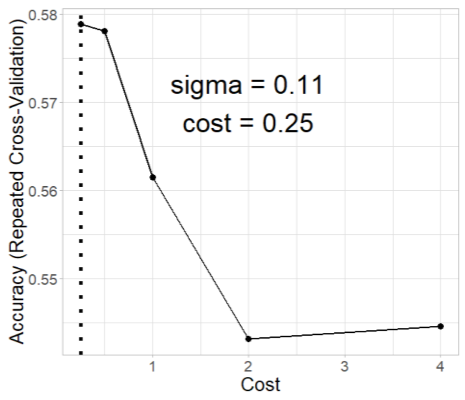


Figure 5.11 Tuning of parameter cost to train SVM-Radial for predicting I_{123} (3-Machine System)

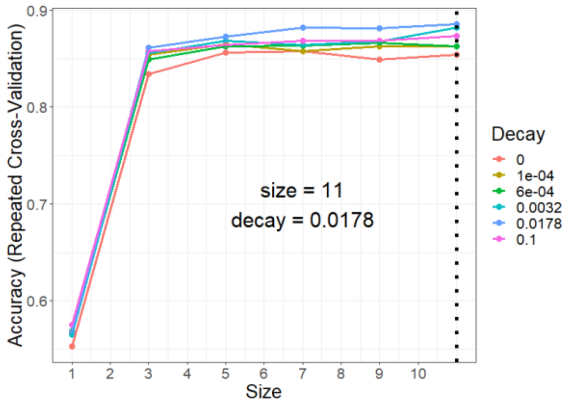


Figure 5.12 Tuning of parameters size and decay to train NNET for predicting I_{123} (3-Machine System)

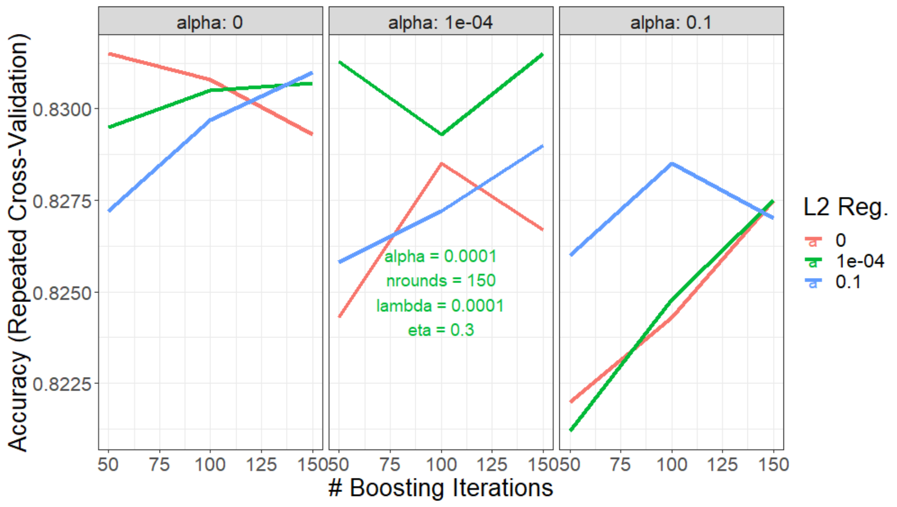


Figure 5.13 Tuning of parameters lambda (i.e., L2 Reg.), alpha and nrounds (i.e., iterations) to train XGB for predicting I_{123} (3-Machine System)

Table 5.5 Tuning Parameters Values (3-Machine System)

Model	Parameter	Value
KNN	**k**	19
RF	**mtry**	2
SVM-Poly	**degree**	1
	scale	0.01
	cost	1
SVM-Radial	**sigma**	0.11
	cost	0.25
NNET	**size**	11
	decay	0.0178
XGB	**lambda**	0.0001
	alpha	0.0001
	#rounds	150
	eta	0.3

The parameter values of the final tunned models are given in *Table 5.5*.

Model Selection

After tuning the parameters and selecting the best tunned models for each of the previously mentioned algorithms by doing within-model comparison, we will now do between-model comparison to select the best models for classification. *Table 5.6* shows the summary statistics of the performance metrics for the best tuned models.

Table 5.6 Performance Metrics for Best tunned models From Repeated CV (3-Machine System)

Model	Mean TPR	Mean TNR	Accuracy (95% CI)
LDA	0.70	0.97	**0.86 (0.84, 0.89)**
KNN	0.48	0.97	0.82 (0.79, 0.85)
DT	0.33	0.94	0.69 (0.66, 0.71)
SVM-Radial	0.55	0.92	0.58 (0.55, 0.61)
SVM-Poly	0.59	0.93	0.59 (0.55, 0.62)
RF	0.50	0.97	0.83 (0.81, 0.84)
NNET	0.76	0.98	**0.89 (0.86, 0.91)**
XGB	0.58	0.97	0.83 (0.80, 0.86)

Based on the Accuracy score, LDA and NNET models are performing best for the repeated k-fold cross-validation. *Figure 5.14* shows the boxplot for the Accuracy score to show the differences among these models.

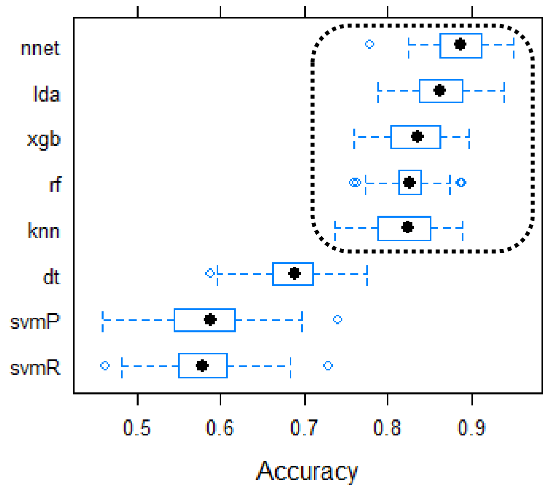


Figure 5.14 Performance Comparison for predicting I_{123} (3-Machine System)

Finally, we will check the accuracies of these models for the test dataset. The accuracies of these models for the test set are given in *Table 5.7*. Based on model performance for the test dataset, we will select NNET for I_{123} as shown in the table.

Table 5.7 Performance Metrics for Top Performing models For Test Set (3-Machine System)

Algos	**Accuracy**	
	Repeated CV	**Test Set**
LDA	0.86	0.85
KNN	0.82	0.83
DT	0.69	0.65
SVM-Radial	0.58	0.67
SVM-Poly	0.59	0.62
RF	0.83	0.83
NNET	**0.89**	**0.89**
XGB	0.83	0.83

5.1.7 Stage 2: Production Decisions (Algo-1)

In stage-2 of Algorithm-1, we will use the predicted commitment decisions obtained from stage-1 and solve the security constrained optimal dispatch problem formulated as a QP problem in eq. (4.3) of chapter 4. *Table 5.8* below shows a comparison between the solutions obtained using this approach and the actual solutions obtained by solving the original MIQP problem. We can see that the presented approach produces solutions which, on average, have a +0.2% higher cost.

Table 5.8 Final Results for the Test Dataset (3-Machine System)

	Avg. Cost [$/hour]	**Avg. Cost Difference [%]**	**Avg. GLI**	**no. of GLV**	**no. of LLV**	**Avg. Time (1000 Instances) [sec]**
Predicted (Algorithm-1)	5416	+0.2	0	0	0	~ 5
Observed (CPLEX)	5405	0	0	0	0	~ 55

5.1.8 Stage 2: Production Decisions (Algo-2)

Like stage-1, we will train several predictive methods for predicting the active power of the committed generators obtained from stage-1. These methods include GLM, KNN, RF, SVM, NNET and XGB. However, the difference here is that we will train one separate model for each generate. So, we will train three models, one for each of P_1, P_2 and P_3.

Hyperparameter Tuning

Again, we will optimize the tuning parameter values of all the predictive models being trained on the training dataset and then select the top performing models for further performance evaluation. *Figure 5.15* through *Figure 5.18* show the optimization profiles of the hyperparameters of some of the models. Mean absolute error (MAE) in the predicted output has been used as the performance metric.

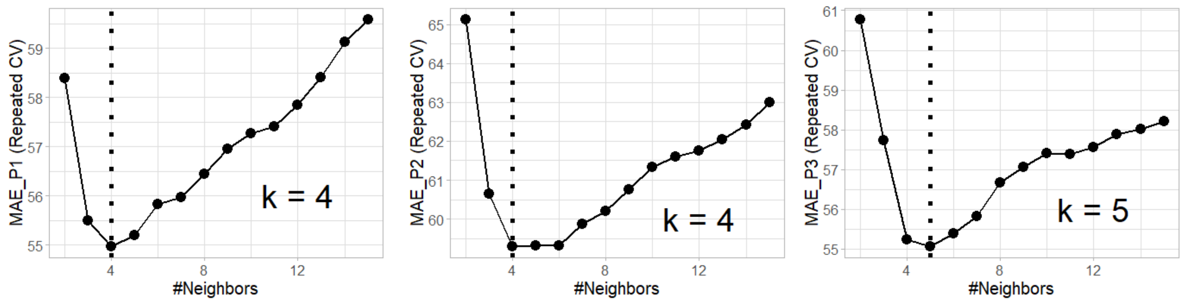


Figure 5.15 Tuning of parameter k to train KNN for predicting P_1, P_2, P_3 (3-Machine System)

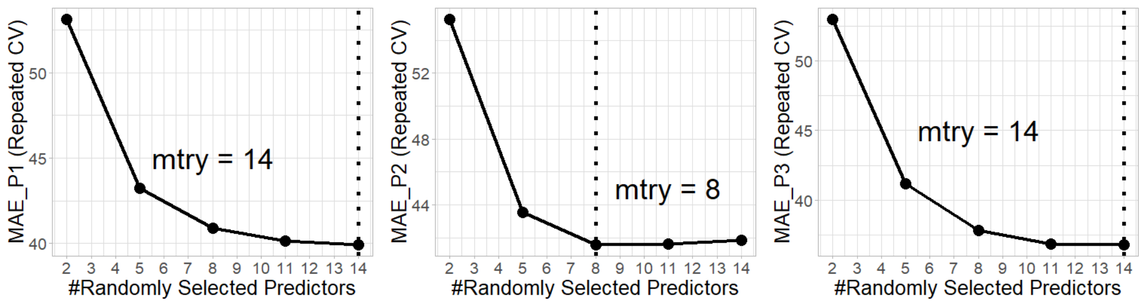


Figure 5.16 Tuning of parameter mtry to train RF for predicting P_1, P_2, P_3 (3-Machine System)

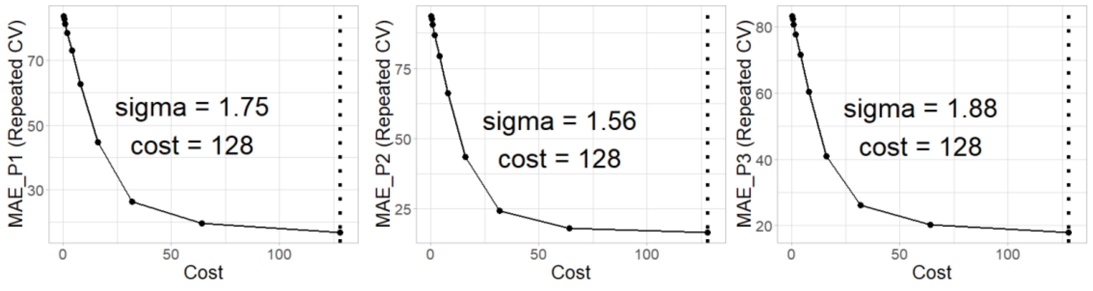


Figure 5.17 Tuning of parameter cost to train SVM-Radial for predicting P_1, P_2, P_3 (3-Machine System)

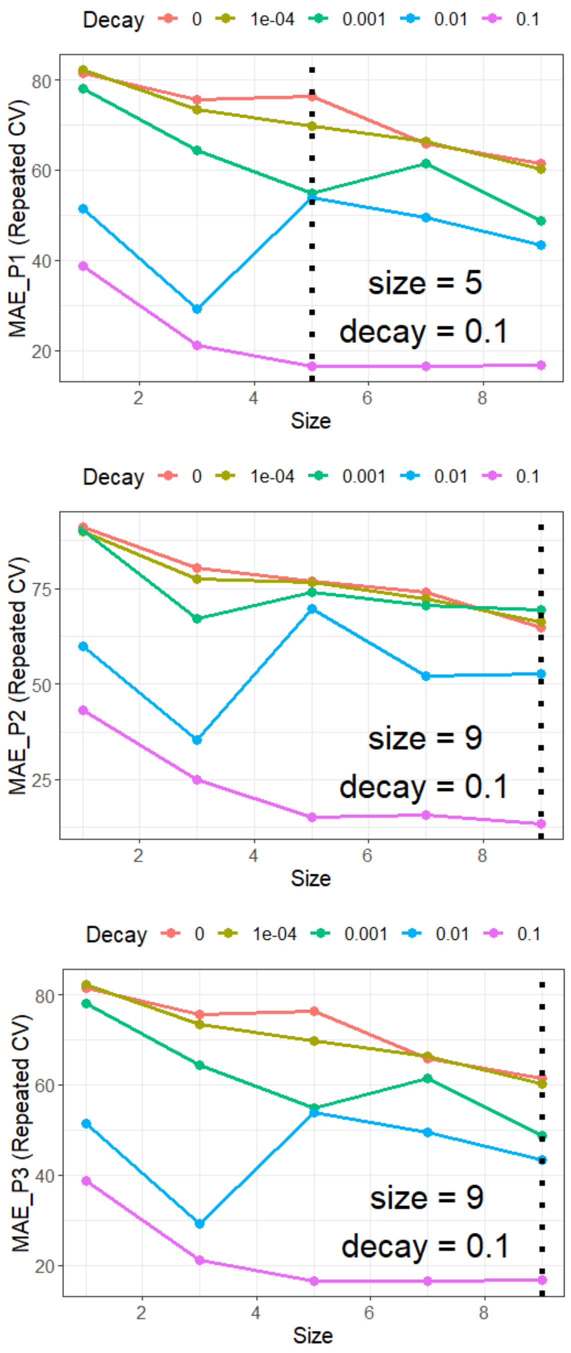


Figure 5.18 Tuning of parameters size and decay to train NNET for predicting P_1, P_2, P_3 (3-Machine System)

The parameter values of the final tunned models are given in *Table 5. 9*.

Table 5. 9 Tuning Parameters Values (3-Machine System)

Model	Parameter	Value		
		P1	P2	P3
KNN	**k**	4	4	5
RF	**mtry**	14	8	14
SVM-Poly	**degree**	3	3	3
	scale	0.1	0.1	0.1
	cost	1	1	1
SVM-Radial	**sigma**	1.75	1.56	1.88
	cost	128	128	128
NNET	**size**	5	9	9
	decay	0.1	0.1	0.1
XGB-Linear	**lambda**	0.1	0.1	0.1
	alpha	0.1	0	0.1
	#rounds	150	150	150
	eta	0.3	0.3	0.3

Model Selection

After tuning the parameters and selecting the best tunned models for each of the previously mentioned algorithms by doing within-model comparison, now, we will do between-model comparison to select the best models for prediction. *Table 5.10* shows the summary statistics of the performance metrics for the best tuned models.

Table 5.10 Performance Metrics for Best tunned models From Repeated CV (3-Machine System)

Model	P1		P2		P3		Combined	
	MAE	R^2	MAE	R^2	MAE	R^2	MAE	R^2
GLM	37.1	0.79	37.3	0.84	37.5	0.80	37.3	0.81
KNN	55.0	0.56	59.3	0.63	55.0	0.60	56.5	0.60
DT	58.8	0.50	63.4	0.56	54.7	0.57	58.9	0.54
SVM-Radial	16.9	0.94	16.7	0.96	18.0	0.94	17.2	0.95
SVM-Poly	79.8	0.35	87.2	0.40	78.8	0.43	81.9	0.40
RF	39.9	0.77	41.6	0.82	36.8	0.81	39.4	0.80
NNET	16.4	0.95	13.3	0.97	16.3	0.95	15.3	0.95
XGB-Linear	37.8	0.77	38.9	0.81	35.9	0.81	37.5	0.80

By looking at the data, we can see that the top two models are SVM-Radial and NNET. Based on the MAE and R^2 scores, neural network (NNET) predictive model is performing best for the repeated k-fold cross-validation, however, the differences between the two models seem to be negligible (i.e., less than 1% differences in mean R^2 in all three cases). When we look at the boxplots in *Figure 5.19*, we see that the distributions of R^2 values for these two models are very similar and the difference is small. To get better insights, we will perform the paired t-test, as shown in *Figure 5.20* and check if this difference is statistically significant or not.

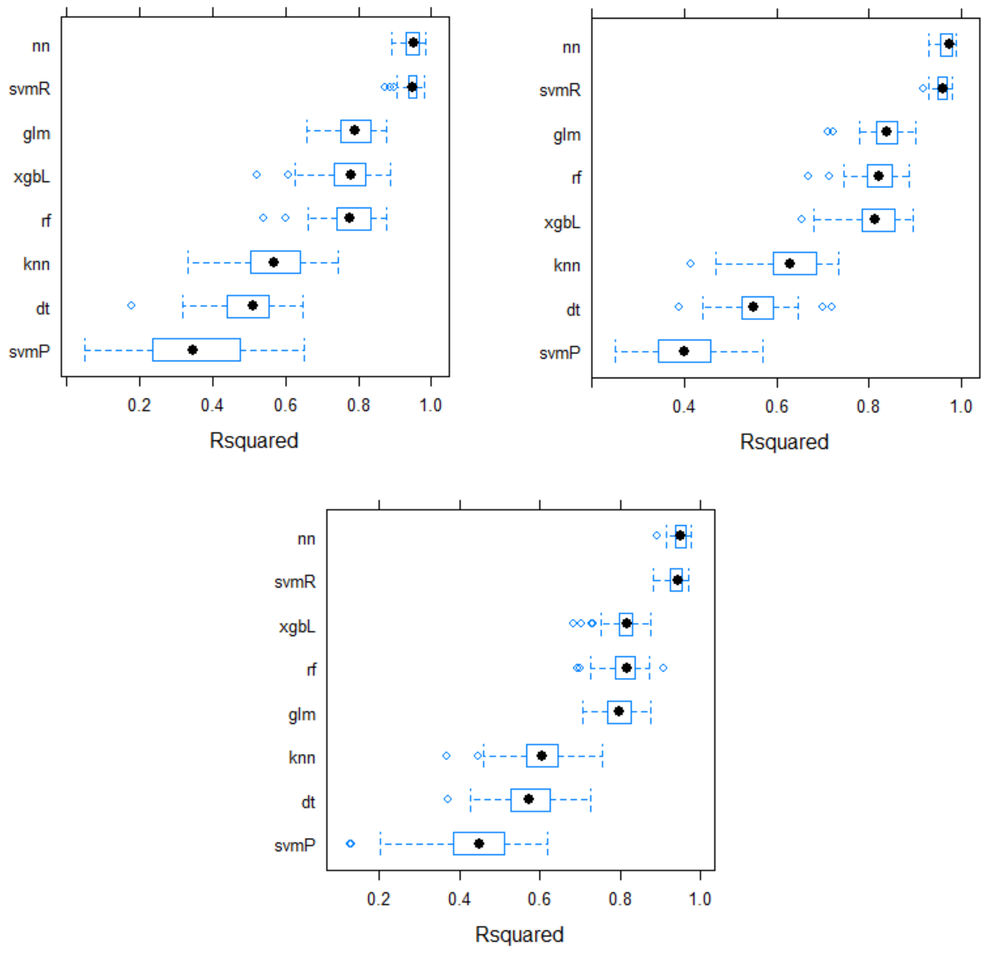
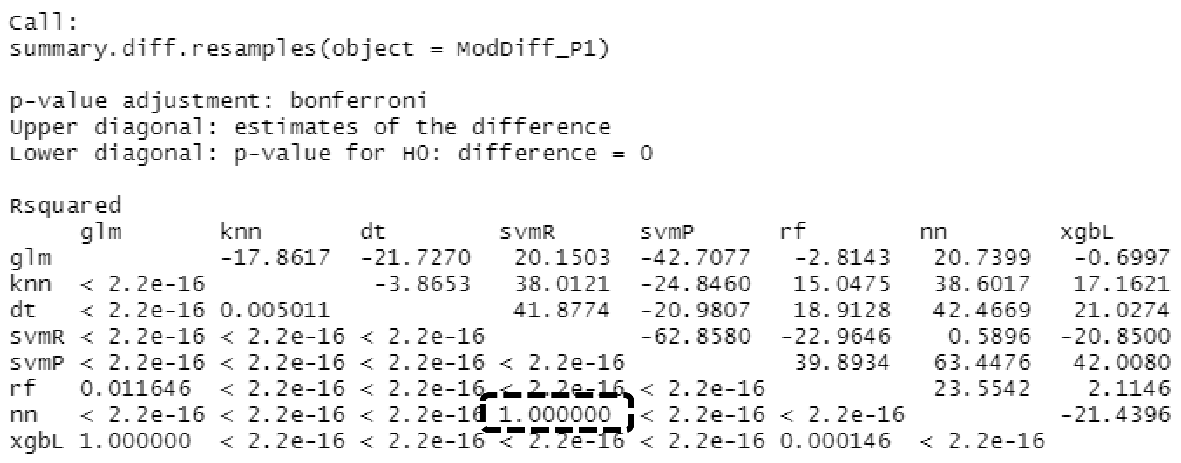
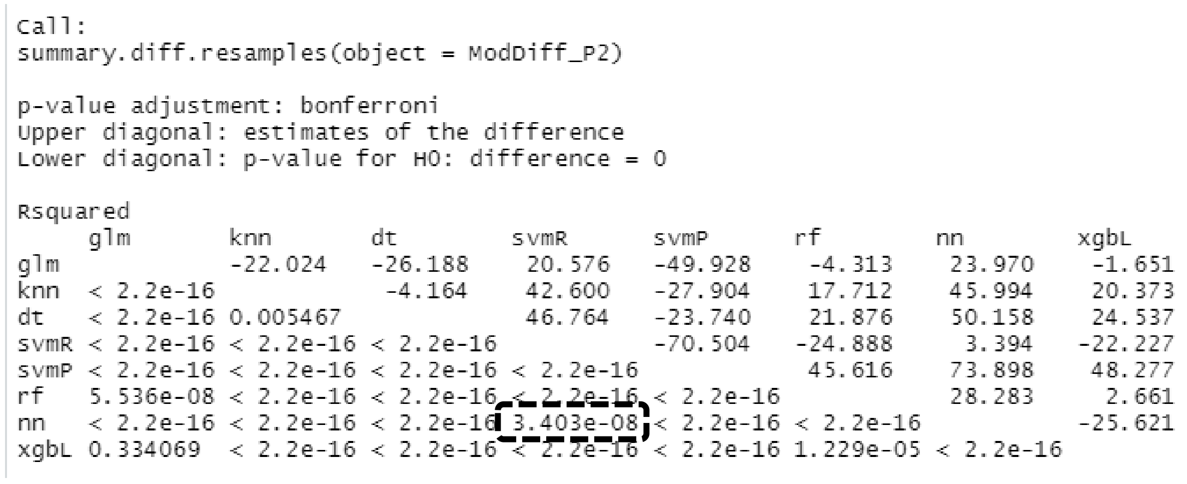


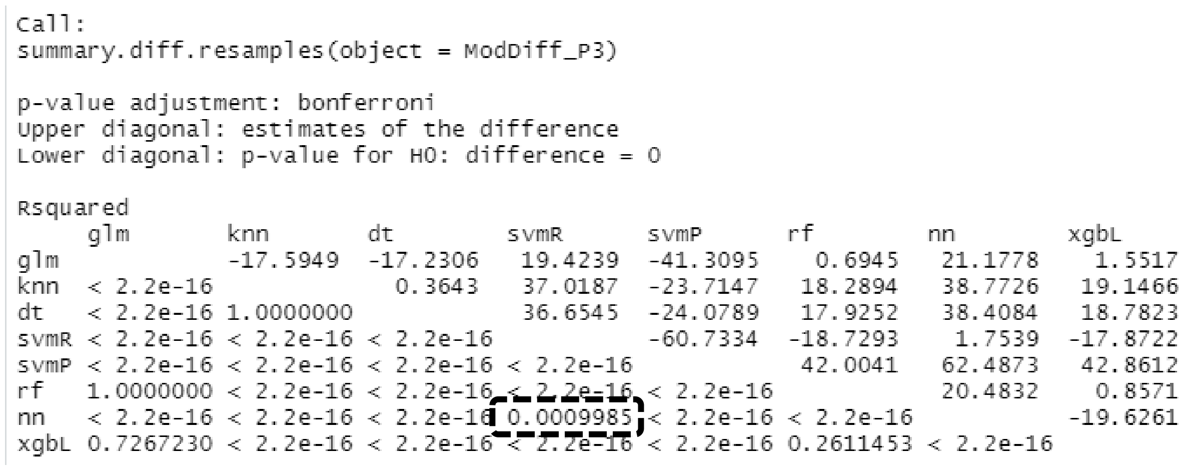
Figure 5.19 Performance Comparison for Predicting P_1, P_2 and P_3 (3-Machine System)



```
Call:
summary.diff.resamples(object = ModDiff_P1)

p-value adjustment: bonferroni
Upper diagonal: estimates of the difference
Lower diagonal: p-value for H0: difference = 0

Rsquared
     glm       knn       dt        svmR      svmP      rf        nn        xgbL
glm            -17.8617  -21.7270   20.1503  -42.7077   -2.8143   20.7399   -0.6997
knn  < 2.2e-16            -3.8653   38.0121  -24.8460   15.0475   38.6017   17.1621
dt   < 2.2e-16 0.005011             41.8774  -20.9807   18.9128   42.4669   21.0274
svmR < 2.2e-16 < 2.2e-16 < 2.2e-16           -62.8580  -22.9646    0.5896  -20.8500
svmP < 2.2e-16 < 2.2e-16 < 2.2e-16 < 2.2e-16            39.8934   63.4476   42.0080
rf   0.011646  < 2.2e-16 < 2.2e-16 < 2.2e-16 < 2.2e-16            23.5542    2.1146
nn   < 2.2e-16 < 2.2e-16 < 2.2e-16 1.000000  < 2.2e-16 < 2.2e-16           -21.4396
xgbL 1.000000  < 2.2e-16 < 2.2e-16 < 2.2e-16 < 2.2e-16 0.000146  < 2.2e-16
```



```
Call:
summary.diff.resamples(object = ModDiff_P2)

p-value adjustment: bonferroni
Upper diagonal: estimates of the difference
Lower diagonal: p-value for H0: difference = 0

Rsquared
     glm       knn       dt        svmR      svmP      rf        nn        xgbL
glm            -22.024   -26.188    20.576   -49.928    -4.313    23.970    -1.651
knn  < 2.2e-16            -4.164    42.600   -27.904    17.712    45.994    20.373
dt   < 2.2e-16 0.005467             46.764   -23.740    21.876    50.158    24.537
svmR < 2.2e-16 < 2.2e-16 < 2.2e-16           -70.504   -24.888     3.394   -22.227
svmP < 2.2e-16 < 2.2e-16 < 2.2e-16 < 2.2e-16            45.616    73.898    48.277
rf   5.536e-08 < 2.2e-16 < 2.2e-16 < 2.2e-16 < 2.2e-16            28.283     2.661
nn   < 2.2e-16 < 2.2e-16 < 2.2e-16 3.403e-08 < 2.2e-16 < 2.2e-16           -25.621
xgbL 0.334069  < 2.2e-16 < 2.2e-16 < 2.2e-16 < 2.2e-16 1.229e-05 < 2.2e-16
```



```
Call:
summary.diff.resamples(object = ModDiff_P3)

p-value adjustment: bonferroni
Upper diagonal: estimates of the difference
Lower diagonal: p-value for H0: difference = 0

Rsquared
     glm       knn       dt        svmR      svmP      rf        nn        xgbL
glm            -17.5949  -17.2306   19.4239  -41.3095    0.6945   21.1778    1.5517
knn  < 2.2e-16             0.3643   37.0187  -23.7147   18.2894   38.7726   19.1466
dt   < 2.2e-16 1.0000000            36.6545  -24.0789   17.9252   38.4084   18.7823
svmR < 2.2e-16 < 2.2e-16 < 2.2e-16           -60.7334  -18.7293    1.7539  -17.8722
svmP < 2.2e-16 < 2.2e-16 < 2.2e-16 < 2.2e-16            42.0041   62.4873   42.8612
rf   1.0000000 < 2.2e-16 < 2.2e-16 < 2.2e-16 < 2.2e-16            20.4832    0.8571
nn   < 2.2e-16 < 2.2e-16 < 2.2e-16 0.0009985 < 2.2e-16 < 2.2e-16           -19.6261
xgbL 0.7267230 < 2.2e-16 < 2.2e-16 < 2.2e-16 < 2.2e-16 0.2611453 < 2.2e-16
```

Figure 5.20 Paired t-test results of Predictive Models for Predicting P_1, P_2 and P_3

(3-Machine System)

After performing the paired t-test, we see that the t-statistic values are small (< 0.05) for NNET and SVM-Radial for P_2 and P_3 models while its value is large for P_1. It indicates that the performance difference of SVM-Radial and NNET models for predicting P2 and P3 is statistically significant, however, for predicting P_1, they are equivalent in terms of performance based on R^2 values. Furthermore, when we evaluate the performance of these two models on the test dataset, *Table 5.11*, we see that the NNET model performs slightly better than the SVM model. Hence, we will select NNET as the best predictive model.

Table 5.11 Performance Metrics for Top Performing models For Test Set (3-Machine System)

Model	**Target Variable**	**MAE**		**R^2**	
		Repeated CV	**Test Set**	**Repeated CV**	**Test Set**
SVM-Radial	**P1**	16.9	10.5	0.94	0.98
	P2	16.7	10.3	0.96	0.98
	P3	18.0	13.1	0.94	0.96
NNET	**P1**	16.4	9.9	0.95	0.98
	P2	13.3	9.0	0.97	0.98
	P3	16.3	11.5	0.95	0.98

5.1.9 Stage-3: Constraints Satisfaction (Algo-2)

In this stage, the solutions obtained from the first two stages are further processed to make sure none of the security constraints are violated. *Table 5.12* shows the instances (out of 197 total instances) in the test set where violations were detected (along with the types of observed violations), and the solutions were adjusted to fulfill the constraints. *Table 5.13* shows the performance for the final solutions for the test set obtained using the proposed 3-stage algorithm.

Table 5.12 Instances in the Test Set with Constraint Violations (3-Machine System)

Type of Violation / Number of Instances	Generator Limits Violation	Total Reserve Requirement Violation	Generation Load Imbalance (>1% of total load)	Transmission Line Limits Violation
Before Stage-3	0	1 %	76.6 %	7.6 %
After Stage-3	0	0	0	0

Table 5.13 Performance Metrics for the Test Dataset (3-Machine System)

	Variable	MAE	R^2
Before Stage-3	**P1**	15.2	0.95
	P2	15.0	0.95
	P3	17.0	0.94
	Combined	15.7	0.95
After Stage-3	**P1**	15.2	0.94
	P2	14.6	0.92
	P3	15.3	0.93
	Combined	15.0	0.93

Table 5.14 below shows the comparison between the predicted and observed solutions. The presented approach produces solutions which, on average, have a +0.3% higher cost.

Table 5.14 Final Results for the Test Dataset (3-Machine System)

	Avg. Cost [$/hour]	Avg. Cost Difference [%]	Avg. GLI	no. of GLV	no. of LLV	Avg. Time (1000 Instances) [sec]
Predicted (Algorithm-2)	5424	+0.3	0	0	0	~ 2
Observed (CPLEX)	5405	0	0	0	0	~ 55

5.2 Case Study 2: IEEE 39-Bus Power System

5.2.1 System Data

The system has 39 buses, 10 generating units, 18 loads and 46 transmission lines. The single line diagram of the system is shown in *Figure 5.21*. The system data is given in *Table 5.15*. Further details of the system can be obtained from the MATPOWER case file *case39*.

Table 5.15 System Data (10-Machine System)

Base MVA	100
Base kV	345
minimum Spinning Reserve requirement (R)	10% of total load

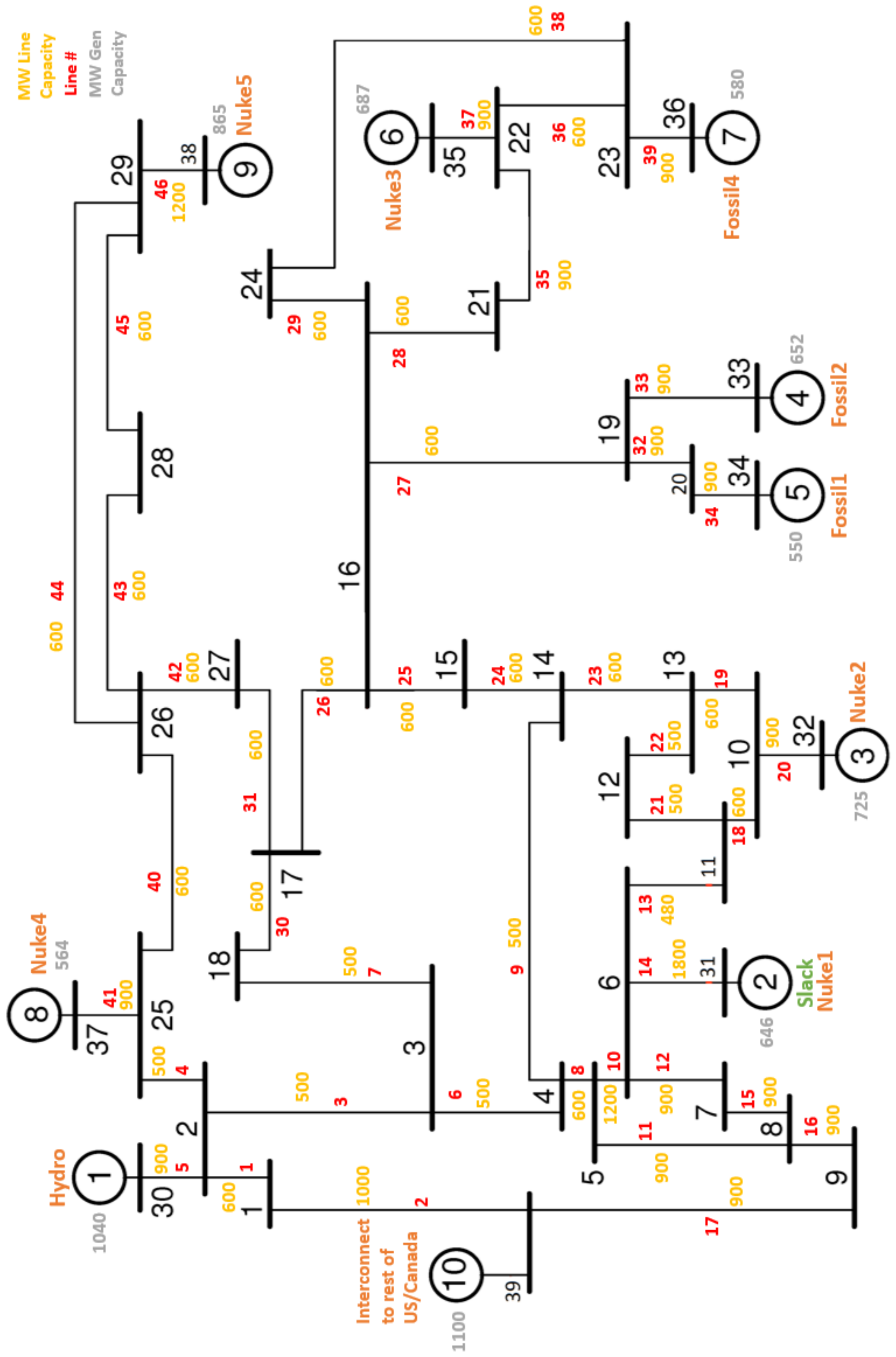


Figure 5.21 Single Line Diagram (10-Machine System)

5.2.2 MIQP Problem Formulation

From the system data, we can see that $N_B = 39$, $N_G = 10$, $N_L = 46$ and the sets $B = \{1,2,3,\dots,39\}$, $G = \{1,2,3,\dots,10\}$, $L = \{1,2,3,\dots,46\}$. Hence, the single-period MIQP problem for this system is formulated as given below. There are a total of 173 constraints (i.e., 92: line limits, 39: nodal balance, 20: generator power limits, 10: spinning reserve limits, 10: binary variable restrictions, 1: reference bus angle, 1: total spinning reserve requirement).

$$\underset{I_g, P_g, R_g, \delta_b}{minimize:} \quad \sum_{g \in G} I_g . PC_g(P_g)$$

$$subject\ to: \quad \sum_{g \in G_b} P_g = d_b + \sum_{l \in L_b} P_l \qquad \forall b \in B$$

$$-\overline{P_l} \leq P_l = \frac{\delta_l}{X_l} \leq \overline{P_l} \qquad \forall l \in L$$

$$\delta_{b_{ref}} = 0$$

$$\underline{P_g} I_g \leq P_g \leq \overline{P_g} I_g \qquad \forall g \in G$$

$$\sum_{g \in G} R_g\, I_g \geq R$$

$$P_g + R_g \leq \left(\overline{P_g} - \underline{P_g}\right) I_g \qquad \forall g \in G$$

$$I_g \in \{0,1\} \qquad \forall g \in G$$

$$P_g \geq 0 \qquad \forall g \in G$$

Here, the angles (δ_l) are in radians while all the other non-binary variables are in per unit. Bus 31 is the reference bus, d_b is the forecasted net power demand at bus b, P_l is the power injection from bus b into the line l, δ_l is the angle difference between bus b and the bus at the other end of the line l, $\overline{P_l}$ and X_l are the maximum power carrying capacity and reactance of the line l, $\delta_{b_{ref}}$ is the phase angle of the reference bus of the system, $\overline{P_g}$, $\underline{P_g}$ and R_g are the maximum power output, minimum power output, and the spinning reserve of the generator g and R is the total minimum reserve requirements of the system.

5.2.3 Data Generation

To train the predictive model, the dataset is generated by solving the above MIQP problem using IBM CPLEX toolbox for MATLAB (this solver is used as a base line method for comparing the performance of the presented approach). A dataset of 1000 random samples, representing different scenarios of power demands and price signals, is generated assuming normal distributions for the loads (d_b) and the cost coefficients (a_g , b_g , c_g). The mean and standard deviation values of the loads (d_b) are given in *Table 5.16*.

Table 5.16 Normal distributions of the loads (10-Machine System)

Bus #	**Mean (SD) in MW**	**Bus #**	**Mean (SD) in MW**
1	48.8 (5)	**21**	137 (14)
3	161 (16)	**23**	123.75 (12)
4	250 (25)	**24**	154.3 (15)
7	116.9 (12)	**25**	112 (11)
8	261 (26)	**26**	69.5 (7)
15	160 (16)	**27**	140.5 (14)
16	164.5 (17)	**28**	103 (10)
18	79 (8)	**29**	141.75 (14)
20	340 (34)	**39**	552 (55)

Figure 5.22 shows the distributions of the loads in the generated dataset. *Figure 5.23* shows the distributions of the MW demands at some of the buses (i.e., 8, 20 and 39) in the generated simulated dataset. These demands are randomly generated assuming normal distributions for power demands with the mean and standard deviation values as given in *Table 5.16*.

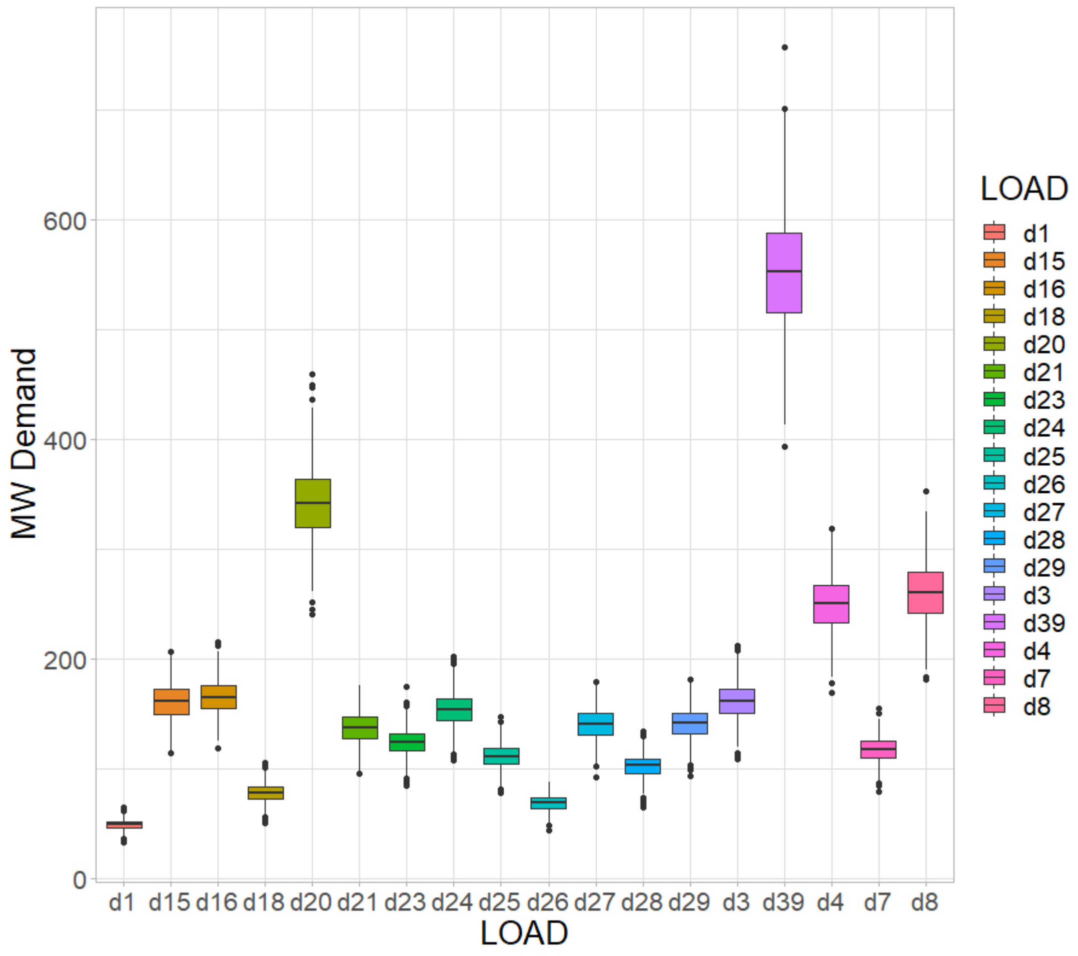


Figure 5.22 Load Distributions in the Generated Dataset (10-Machine System)

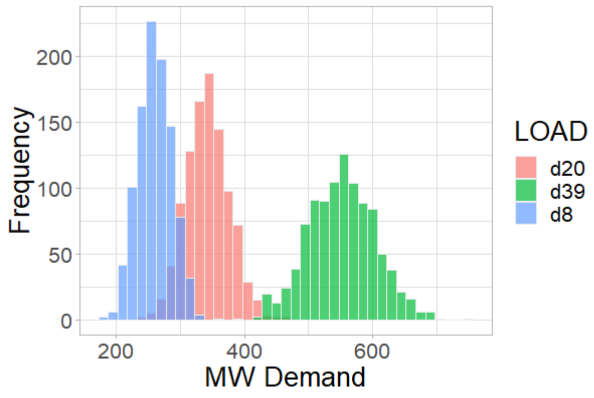


Figure 5.23 Histograms of the Bus Power Demands (10-Machine System)

There are three types of generating units in the system i.e., hydro, fossil fuel and nuclear power plants. The distributions of the cost coefficients (b_g), of one generating unit of each type, in the generated dataset are shown in *Figure 5.24*. The spread of the cost curves for one of each type of a generator resulted from these cost coefficients are shown in *Figure 5.25*.

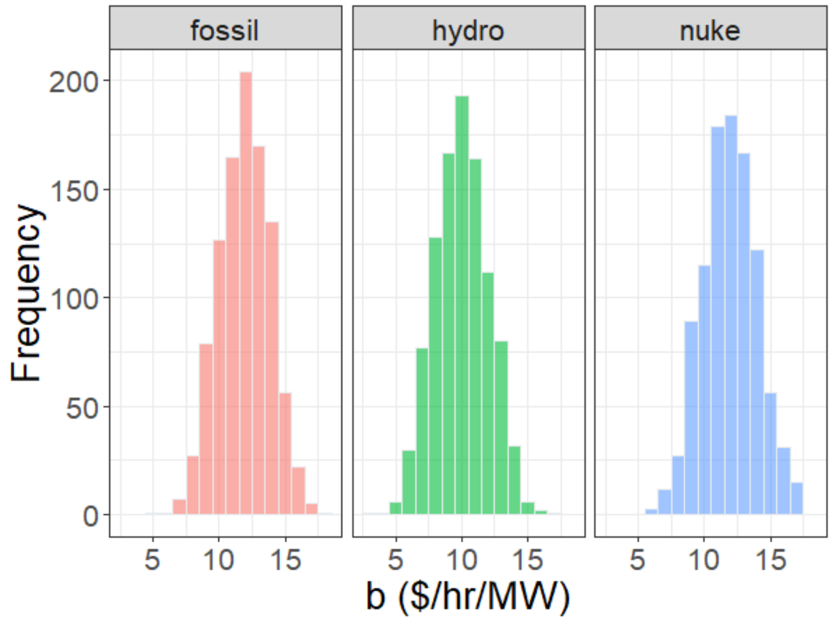


Figure 5.24 Distribution of Cost Coefficients b_g (10-Machine System)

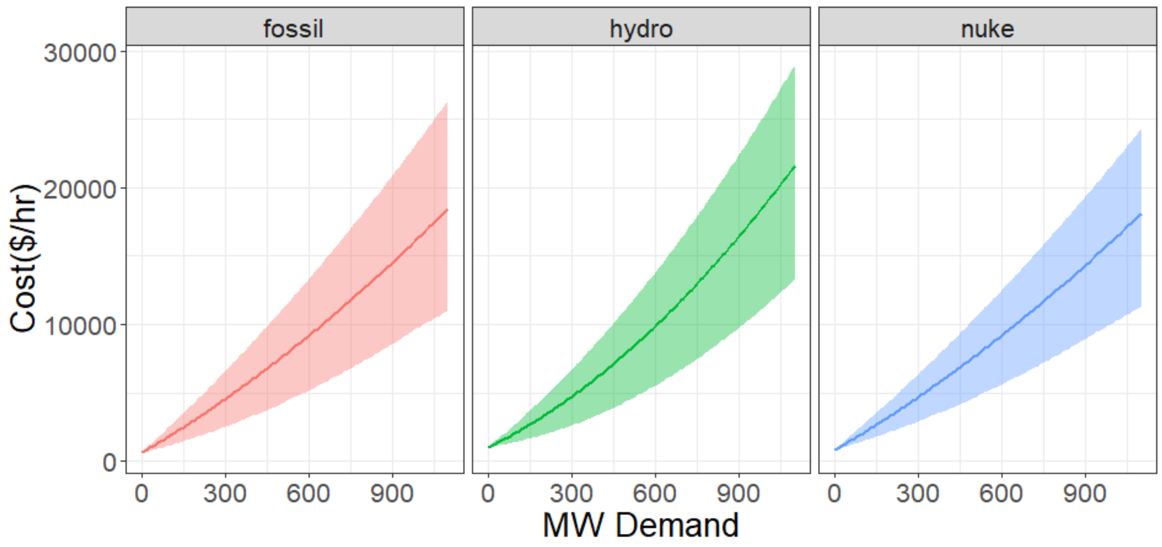


Figure 5.25 Spread of Generators' Cost Curves (10-Machine System)

To label the generated dataset, the instances are solved through simulations by solving the SC-UC problem presented earlier. MATLAB (R2021a) and CPLEX toolbox for MATLAB are used to solve the MIQP problem presented above. This data is then used for training and testing of the learning algorithms using R-Studio. *Table 5.17* shows a solved sample in the dataset. Here, the angles are in degrees.

Table 5.17 A Solved Sample from the Generated Dataset (10-Machine System)

idx	**cost**	**d1**	**d3**	**d4**	**d7**	**d8**	**d15**	**d16**	**d18**	**d20**
1	44664.6	53	163	245	113	237	160	176	94	372
d21	**d23**	**d24**	**d25**	**d26**	**d27**	**d28**	**d29**	**d39**	**a1**	**a2**
144	125	156	89	67	160	105	139	596	982	481
a3	**a4**	**a5**	**a6**	**a7**	**a8**	**a9**	**a10**	**b1**	**b2**	**b3**
600	608	608	796	791	815	790	773	11.45	13.63	11.41
b4	**b5**	**b6**	**b7**	**b8**	**b9**	**b10**	**c1**	**c2**	**c3**	**c4**
9.82	15.76	14.02	10.13	14.33	9.94	12.13	0.006	0.008	0.003	0.004
c5	**c6**	**c7**	**c8**	**c9**	**c10**	**I1**	**I2**	**I3**	**I4**	**I5**
0.003	0.004	0.003	0.004	0.004	0.003	0	0	1	1	0
I6	**I7**	**I8**	**I9**	**I10**	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**
0	1	0	1	1	0	0	635.7	652	0	0
P7	**P8**	**P9**	**P10**	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**
580	0	765.6	560.6	0	0	89.27	0	0	0	0
R8	**R9**	**R10**	**del1**	**del2**	**del3**	**del4**	**del5**	**del6**	**del7**	**del8**
0	99.35	539.4	-1.91	-0.32	-0.69	-0.58	-0.3	0	-1.1	-1.35
del9	**del10**	**del11**	**del12**	**del13**	**del14**	**del15**	**del16**	**del17**	**del18**	**del19**
-1.8	2.93	1.94	2.14	2.35	1.03	0.89	1.71	1.15	0.17	4.86
del20	**del21**	**del22**	**del23**	**del24**	**del25**	**del26**	**del27**	**del28**	**del29**	**del30**
1.73	2.54	4.57	5.96	1.87	0.58	4.28	2.11	11.08	14.16	-0.32
del31	**del32**	**del33**	**del34**	**del35**	**del36**	**del37**	**del38**	**del39**		
0	10.73	10.55	1.73	4.57	15	0.58	21.19	-2.11		

For prediction, we will train one model for each binary variable (I_g) and another model for predicting its active power output (P_g). Hence for each generator, we will train two predictive models i.e., one to predict its commitment status and another to predict its production level.

5.2.4 Data Preprocessing

As a first step, we look for any skewness in the distributions of the predictor variables i.e., variables d_1 through c_{10} in the *Table 5.17*. Since the predictor values were randomly generated from a normal distribution, the frequency distributions of all the predictor variables are very much normally distributed. It can also be confirmed from the near-zero skewness coefficients of the distributions of the variables. Skewness coefficients of some of the variables are given in *Table 5.18*.

Table 5.18 Skewness Coefficients of some of the Predictors (10-Machine System)

Variable	Skewness	Variable	Skewness
d3	0.03	**b2**	0.04
d15	0.04	**b4**	0.07
d23	0.01	**b7**	-0.08
d39	0.07	**b10**	-0.02
a1	-0.06	**c1**	-0.07
a2	0.07	**c3**	-0.04
a5	0.07	**c6**	-0.02
a9	-0.03	**c8**	-0.03

Next, we normalize the numeric predictors using min-max normalization. This will normalize all the predictor values to a common scale of 0-1. After that, we will check if there are any between-predictor (pairwise) correlations to avoid any instabilities in the trained models. For this, we will calculate the correlation matrix and draw the correlation plot of the predictors to check for any significant dependencies between the predictor variables. *Figure 5.26* shows all the pairwise correlations of the variables and it can be seen in the correlation plot that there are no significant between-predictor correlations.

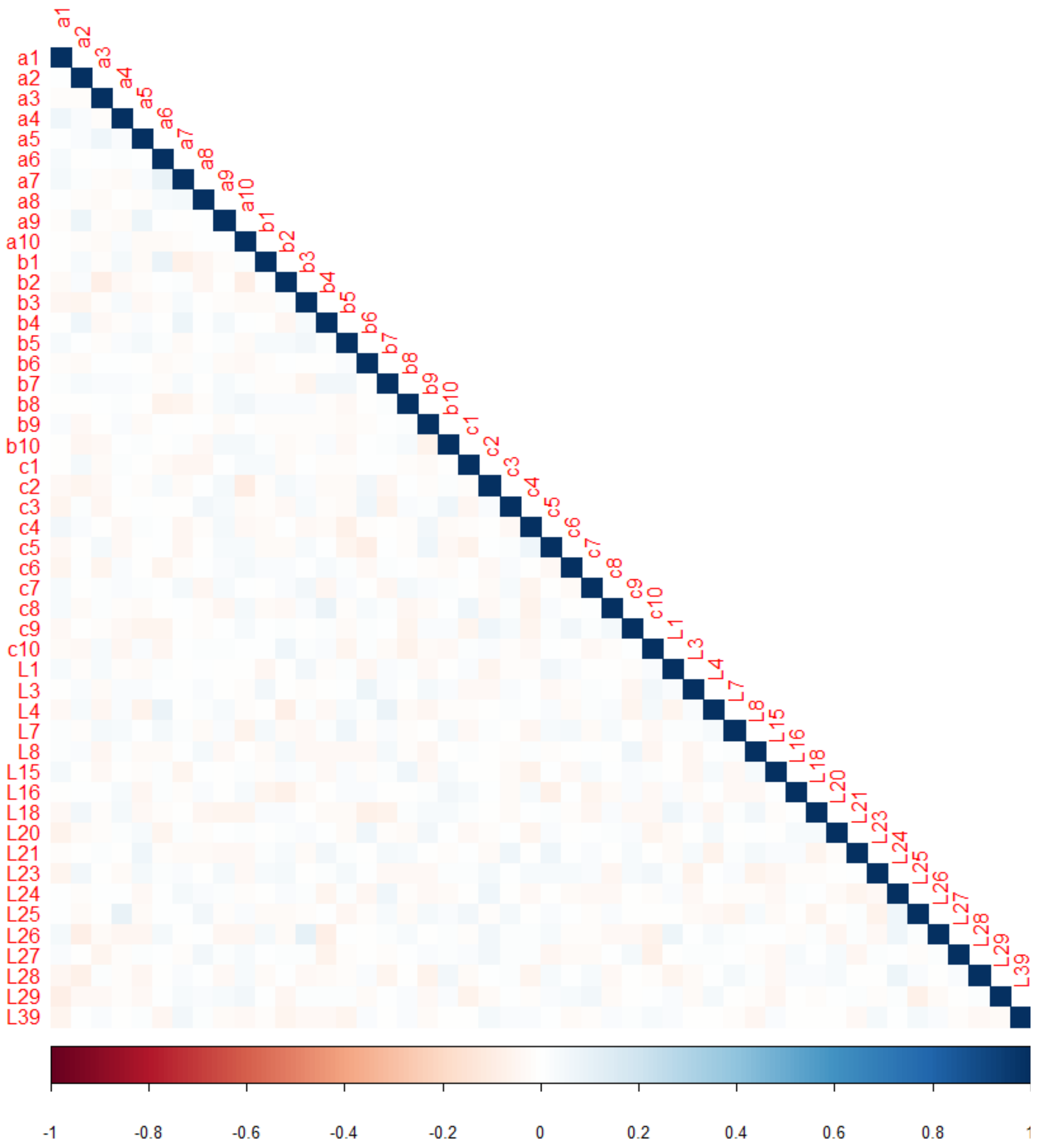


Figure 5.26 Pairwise Correlations of the Transformed Predictor Variables (10-Machine System)

We can also confirm absence of any significant collinearity by locking at the scree plot from PCA of the predictors. *Figure 5.27* shows the scree plot of the predictors. We can see that there is no sharp drop in the amount of variability summarized by the components.

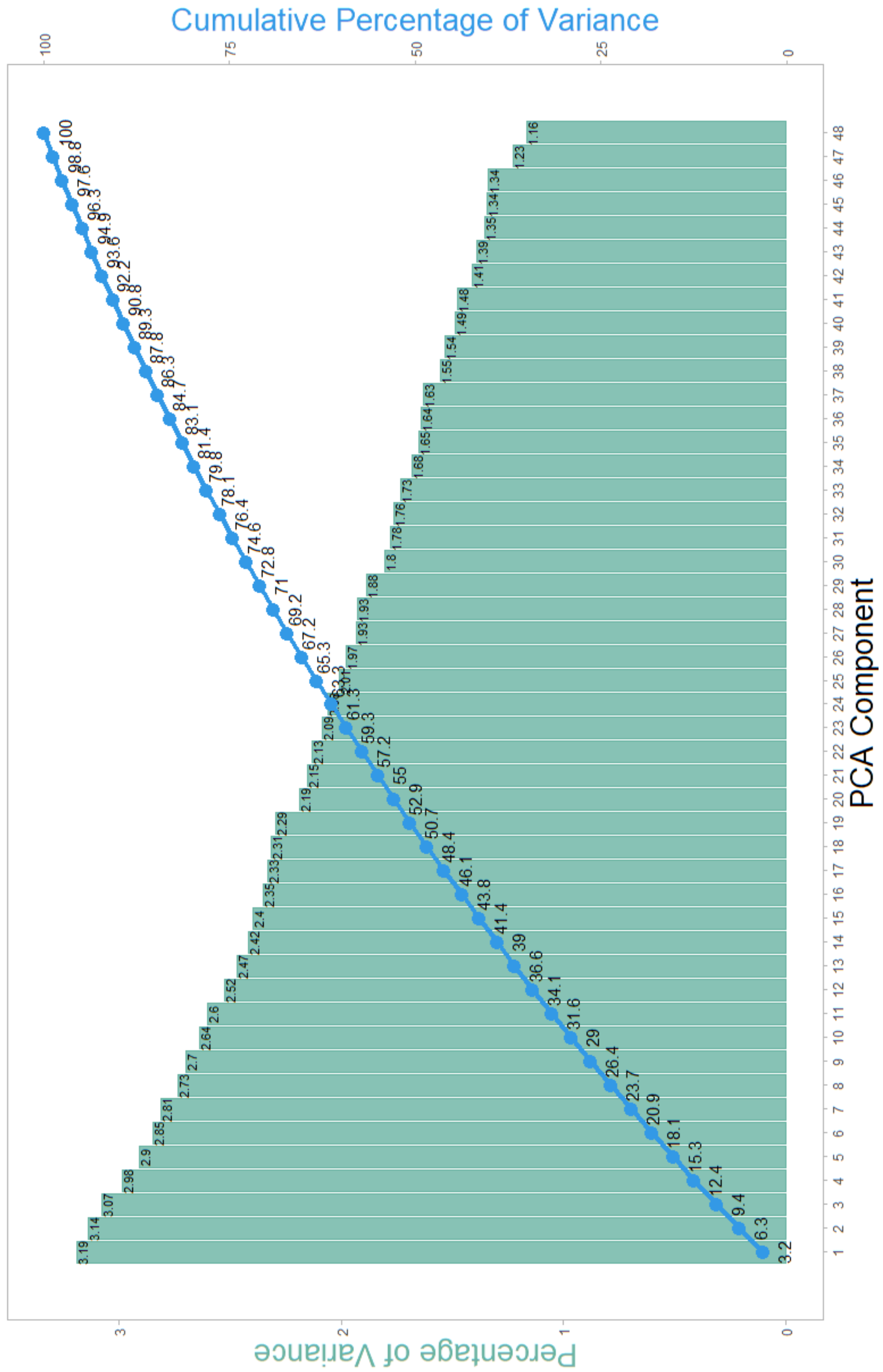


Figure 5.27 A Scree Plot from PCA Analysis of the Predictors (10-Machine System)

5.2.5 Data Splitting

Initially, the dataset is divided into 80/20% train/test split using stratified random sampling to preserve the distributions of the classes in the train and the test datasets. This 80% train dataset is used for training of the learning algorithms and top models are initially selected based on their performance using 10-fold cross-validation repeated 3 times. After that, the selected models are evaluated again using the 20% test dataset to obtain their performance for the unseen data and the best model is selected for prediction.

5.2.6 Class Imbalance

The data in the train dataset has low class imbalance. To balance the class distributions in the train dataset, we will use up-sampling. *Figure 5.28* shows the histograms of the classes before and after up-sampling in which additional samples are added to the minority classes with replacement.

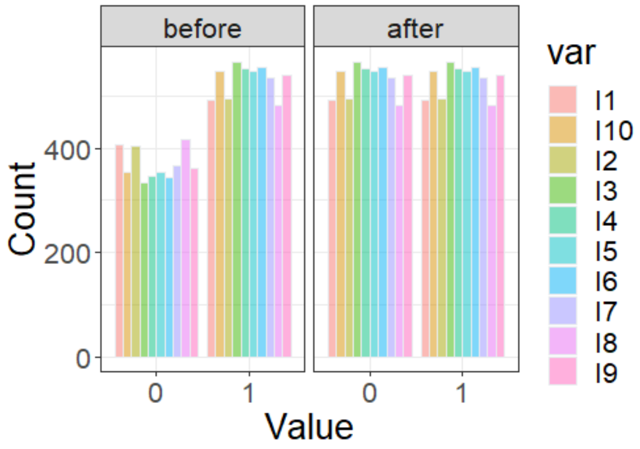


Figure 5.28 Distributions of the Classes in the Train Dataset before and after Up-Sampling (10-Machine System)

5.2.7 Stage 1: Commitment Decisions (Algo-1 & Algo-2)

In this stage, we will train several predictive models for each of the binary variables I_g. These models include GLM, LDA, KNN, RF, SVM, NNET and XGB. After that we will select top few models by comparing their performance on the training dataset. Then these models will be tested on the test dataset and the best performing model will be selected as the predictive model for that specific generator.

Hyperparameter Tuning

During the training stage of the models, the tuning parameters are optimized using grid search method. The hyperparameter tuning profiles for KNN, RF, SVM-Radial and NNET are shown in *Figure 5.29* through *Figure 5.32*. The final best tunned parameters values for all the trained models are given in *Table 5.19*. Mean absolute error (MAE) has been used as the performance metric for tuning.

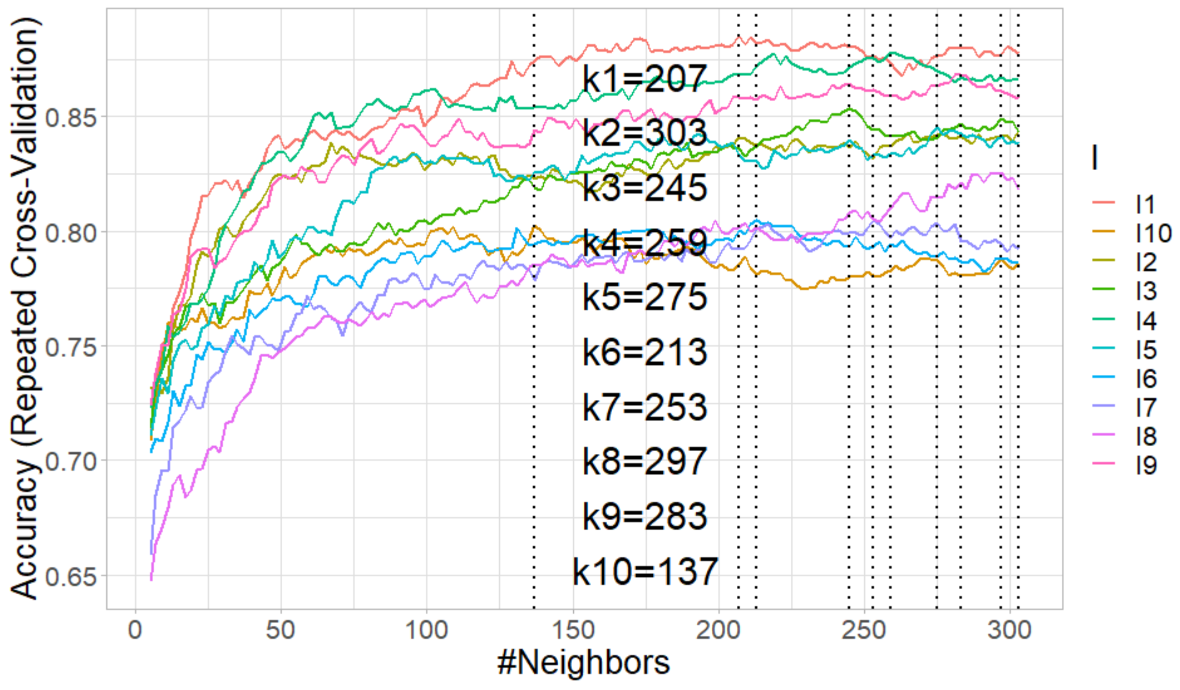


Figure 5.29 Tuning of parameter k to train KNN for predicting Ig (10-Machine System)

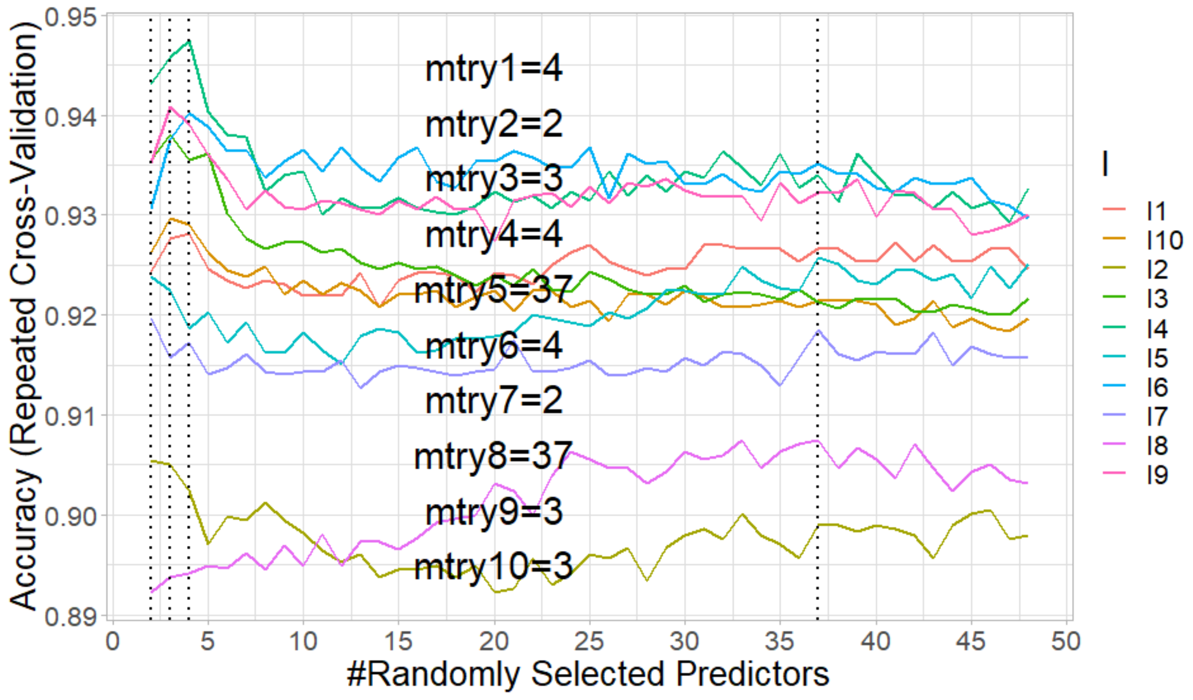


Figure 5.30 Tuning of parameter mtry to train RF for predicting Ig (10-Machine System)

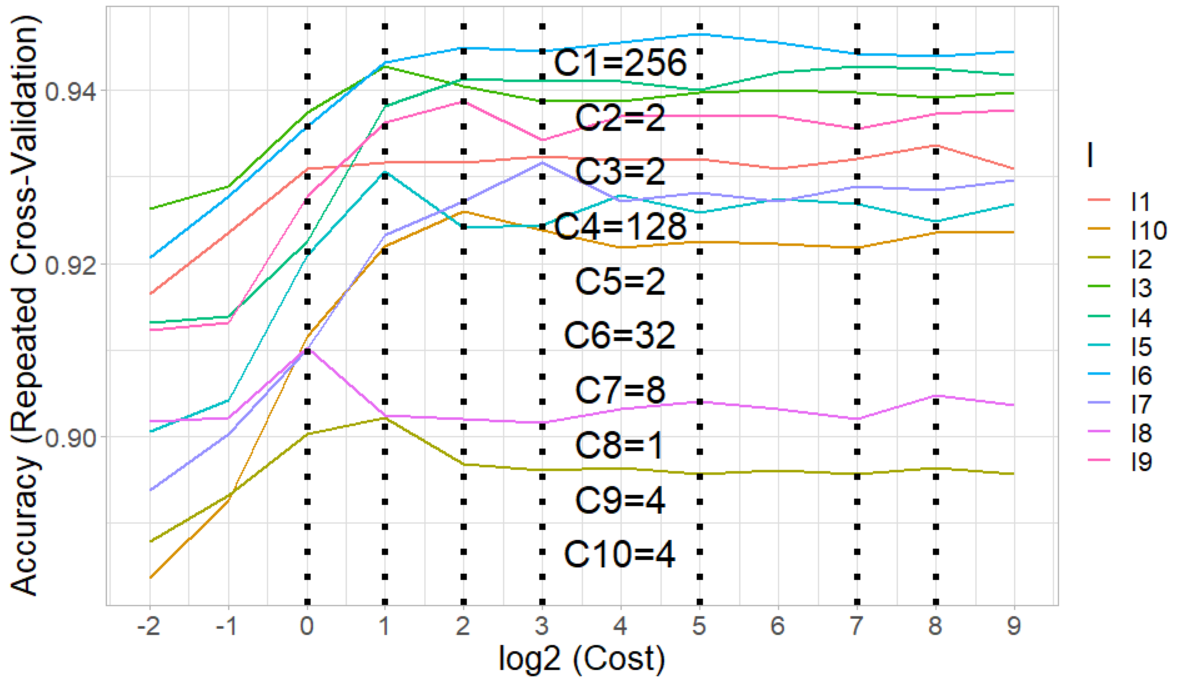


Figure 5.31 Tuning of parameter cost to train SVM-Radial for predicting Ig

(10-Machine System)

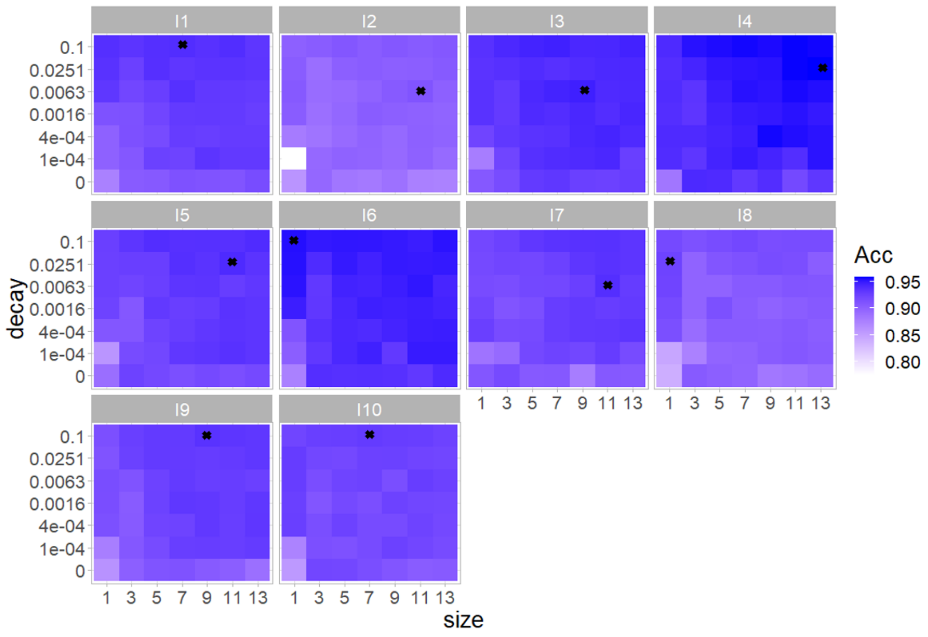


Figure 5.32 Heatmap for tuning of size and decay to train NNET for predicting Ig (10-Machine System)

Table 5.19 Tuning Parameters Values (10-Machine System)

Model	Parameter	I_1	I_2	I_3	I_4	I_5
KNN	k	207	303	245	259	275
RF	mtry	4	2	3	4	37
SVM-Poly	degree	2	1	2	3	2
	scale	0.01	10	0.01	0.01	0.01
	cost	1	2	4	2	1
SVM-Radial	sigma	0.01	0.01	0.01	0.01	0.01
	cost	256	2	2	128	2
NNET	size	7	11	9	13	11
	decay	0.1	0.006	0.006	0.025	0.025
XGB-Linear	lambda	0.003	1e-4	0.003	0.1	0.1
	alpha	1e-4	1e-4	1e-4	0.1	0.003
	#rounds	150	200	100	200	150
	eta	0.3	0.3	0.3	0.3	0.3
XGB-Tree	#rounds	841	774	1000	819	889
	eta	0.46	0.16	0.28	0.38	0.15
	max_depth	10	7	8	5	5
	gamma	1.21	1.83	1.64	1.27	0.39
	colsample_bytree	0.49	0.48	0.43	0.46	0.39
	min_child_weight	6	8	2	2	7
	subsample	0.83	0.79	0.69	0.96	0.63

Model	Parameter	I_6	I_7	I_8	I_9	I_{10}
KNN	**k**	213	253	297	283	137
RF	**mtry**	4	2	37	3	3
SVM-Poly	**degree**	1	2	1	3	2
	scale	0.1	0.01	10	0.01	0.01
	cost	4	4	2	2	4
SVM-Radial	**sigma**	0.01	0.01	0.01	0.01	0.01
	cost	32	8	1	4	4
NNET	**size**	1	11	1	9	7
	decay	0.1	0.006	0.025	0.1	0.1
XGB-Linear	**lambda**	0.1	1e-4	0.1	1e-4	0
	alpha	0.003	0	1e-4	0.003	0
	#rounds	150	200	150	200	150
	eta	0.3	0.3	0.3	0.3	0.3
XGB-Tree	**#rounds**	926	431	690	816	938
	eta	0.33	0.39	0.03	0.11	0.28
	max_depth	3	8	10	10	7
	gamma	1.20	0.79	2.79	2.1	2.23
	colsample_bytree	0.38	0.44	0.6	0.68	0.44
	min_child_weight	2	8	12	8	2
	subsample	0.61	0.83	0.79	0.85	0.56

Model Selection

After doing within-model comparison and finding the tunned values of the parameters, now, we will do between-model comparison to select the best models for classification. *Table 5.20* shows the summary statistics of the performance metrics for the best tuned models.

Table 5.20 Accuracy of tunned models from repeated CV (10-Machine System)

PM	Var	GLM	KNN	RF	SVM -P	SVM -R	NN	XGB -L	XGB -T	LDA
Accuracy	I1	0.934	0.885	0.928	0.934	0.934	0.939	0.941	0.939	0.931
	I2	0.904	0.844	0.905	0.910	0.902	0.911	0.928	0.923	0.901
	I3	0.935	0.853	0.938	0.942	0.943	0.947	0.947	0.943	0.934
	I4	0.939	0.878	0.948	0.942	0.943	0.959	0.957	0.951	0.923
	I5	0.926	0.845	0.926	0.931	0.931	0.941	0.941	0.939	0.913
	I6	0.950	0.805	0.940	0.955	0.947	0.955	0.952	0.950	0.931
	I7	0.917	0.805	0.920	0.927	0.932	0.939	0.936	0.929	0.904
	I8	0.920	0.826	0.908	0.926	0.910	0.924	0.923	0.920	0.910
	I9	0.912	0.868	0.941	0.938	0.939	0.936	0.953	0.951	0.911
	I10	0.927	0.802	0.930	0.930	0.926	0.930	0.938	0.937	0.895
	Com.	0.926	0.841	0.928	0.933	0.931	0.938	0.941	0.938	0.915

Based on these results, we can see that KNN and LDA models are performing worst in most cases and hence are dropped. Remaining models are performing similar in most cases. So, we will obtain their performances for the test dataset as given in *Table 5.21*. We see that GLM model is performing best for the test dataset with an overall accuracy score of 91.4%. But we can see that for some of the variables (I_1, I_2, I_4, I_6, I_8), the other models (highlighted in yellow) are performing better than GLM. So, we can choose those for these variables and can get a best performance with an accuracy score of 92.6%. The final selected models for predicting variables I_g are given in *Table 5.22* along with the accuracy scores.

Table 5.21 Accuracy of tunned models from test set (10-Machine System)

PM	Var	GLM	RF	SVM-P	SVM-R	NN	XGB-L	XGB-T
Accuracy	I1	0.900	0.900	0.875	0.870	0.915	0.925	0.920
	I2	0.860	0.860	0.860	0.870	0.850	0.910	0.900
	I3	0.925	0.915	0.910	0.905	0.905	0.915	0.925
	I4	0.920	0.880	0.870	0.870	0.925	0.895	0.885
	I5	0.940	0.885	0.925	0.925	0.920	0.920	0.925
	I6	0.900	0.855	0.910	0.885	0.900	0.870	0.905
	I7	0.940	0.865	0.910	0.875	0.900	0.920	0.915
	I8	0.885	0.890	0.870	0.840	0.870	0.890	0.915
	I9	0.940	0.920	0.875	0.885	0.910	0.940	0.935
	I10	0.930	0.855	0.910	0.880	0.910	0.880	0.890
	Com.	0.914	0.883	0.892	0.881	0.901	0.907	0.912

Table 5.22 Final selected models for predicting Ig (10-Machine System)

Target Variable	Predictive Model	Accuracy
I1	XGB-Linear	0.925
I2	XGB-Linear	0.91
I3	GLM	0.925
I4	NNET	0.925
I5	GLM	0.94
I6	SVM-Poly	0.91
I7	GLM	0.94
I8	XGB-Tree	0.915
I9	GLM	0.94
I10	GLM	0.93
Combined		**0.926**

5.2.8 Stage 2: Production Decisions (Algo-1)

In stage-2 of Algorithm-1, we will use the predicted commitment decisions obtained from stage-1 and solve the security constrained optimal dispatch problem formulated as a QP problem in chapter 4. *Table 5.23* below shows a comparison between the solutions obtained using this approach and the actual solutions obtained by solving the original MIQP problem. We can see that the presented approach produces solutions which, on average, have a +0.2% higher cost.

Table 5.23 Final results for the test dataset (10-Machine System)

	Avg. Cost [$/hour]	**Avg. Cost Difference [%]**	**Avg. GLI**	**no. of GLV**	**no. of LLV**	**Avg. Time (1000 Instances) [sec]**
Predicted (Algorithm-1)	44021	+0.2	0	0	0	~ 11
Observed (CPLEX)	43931	0	0	0	0	~ 114

5.2.9 Stage 2: Production Decisions (Algo-2)

In this stage, we will train predictive models for predicting the active power of the committed generators obtained from stage 1. These models include GLM, KNN, RF, SVM, NNET and XGB. Like previous stage, we will train one model for predicting each variable P_g.

Hyperparameter Tuning

The hypermeter tuning profiles for some of the used models are shown in *Figure 5.33* through *Figure 5.36*. MAE has been used as the performing metric for tuning the parameters.

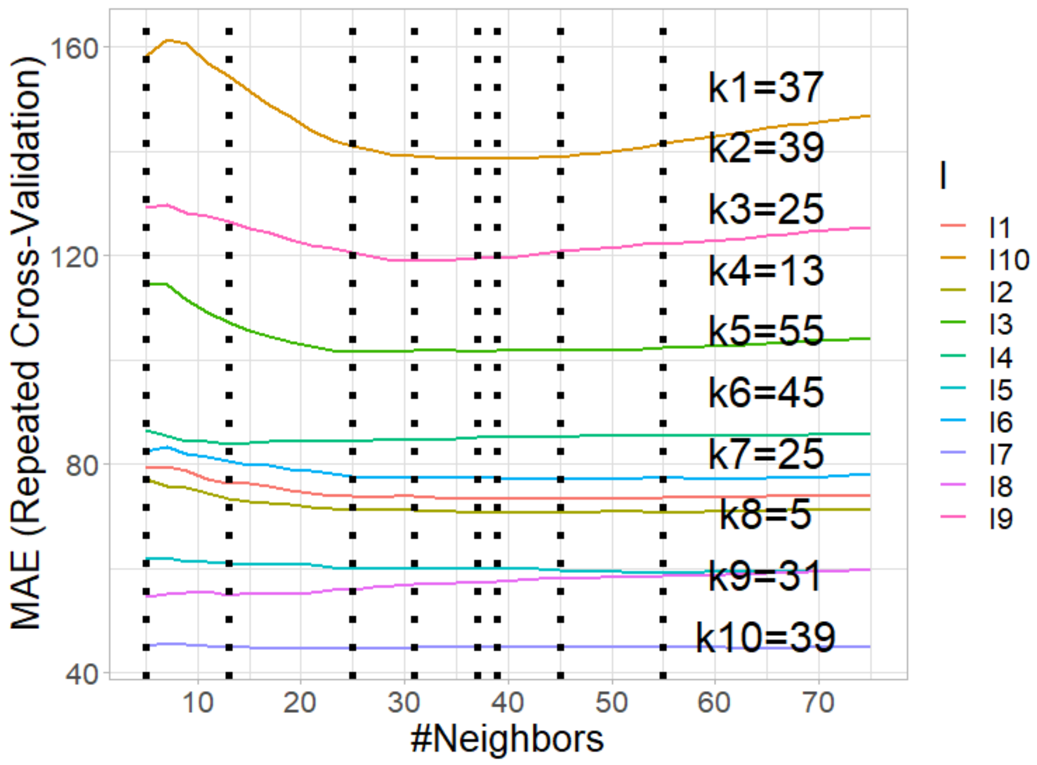


Figure 5.33 Tuning of parameter k to train KNN for predicting P_g (10-Machine System)

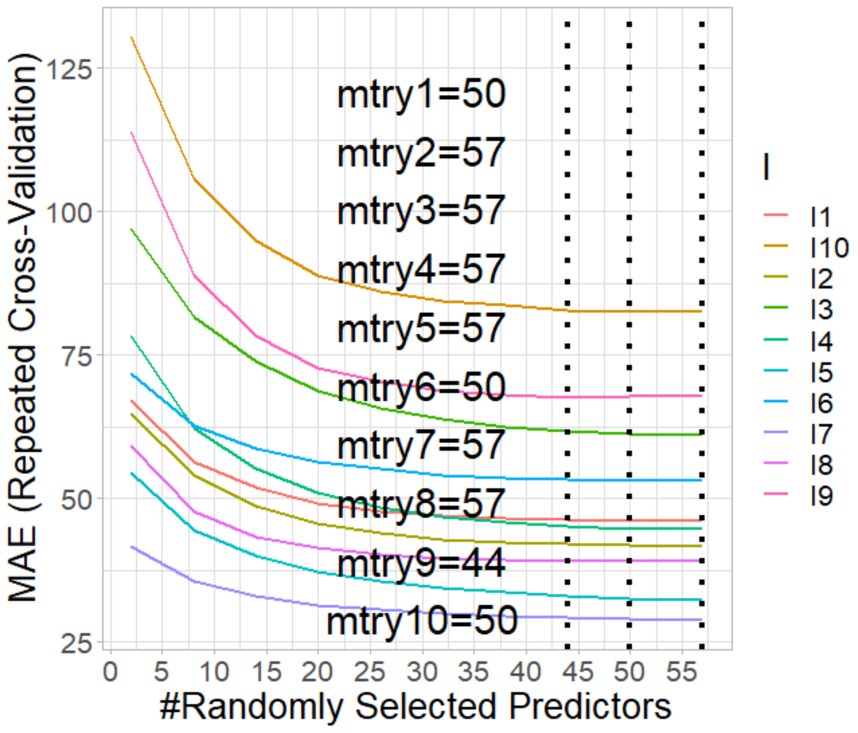


Figure 5.34 Tuning of parameter mtry to train RF for predicting P_g (10-Machine System)

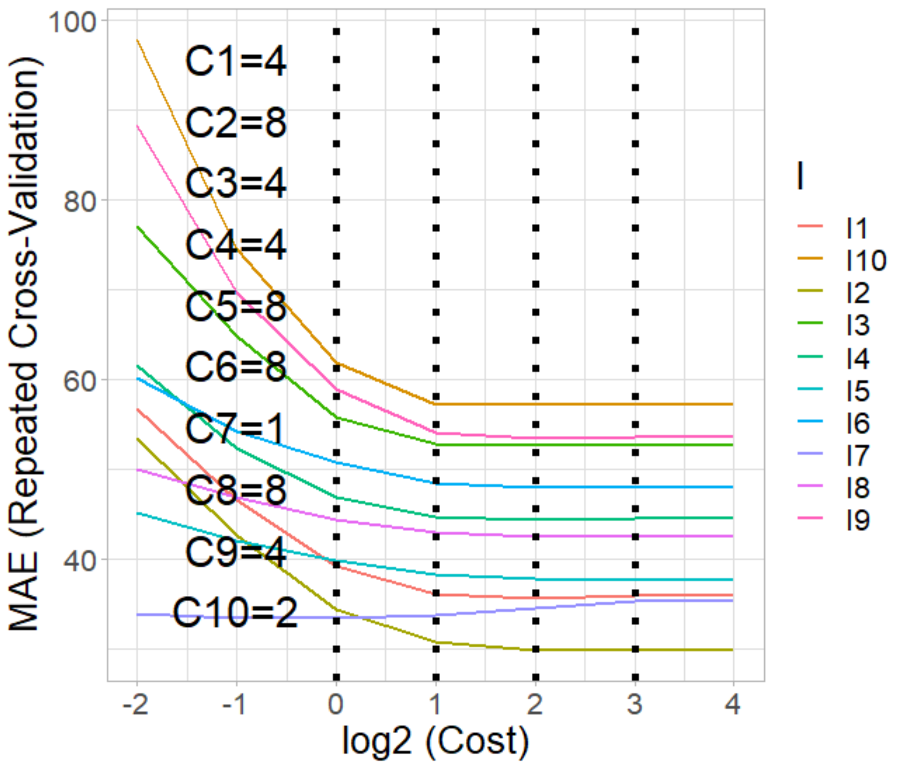


Figure 5.35 Tuning of parameter cost to train SVM-Radial for predicting P_g (10-Machine System)

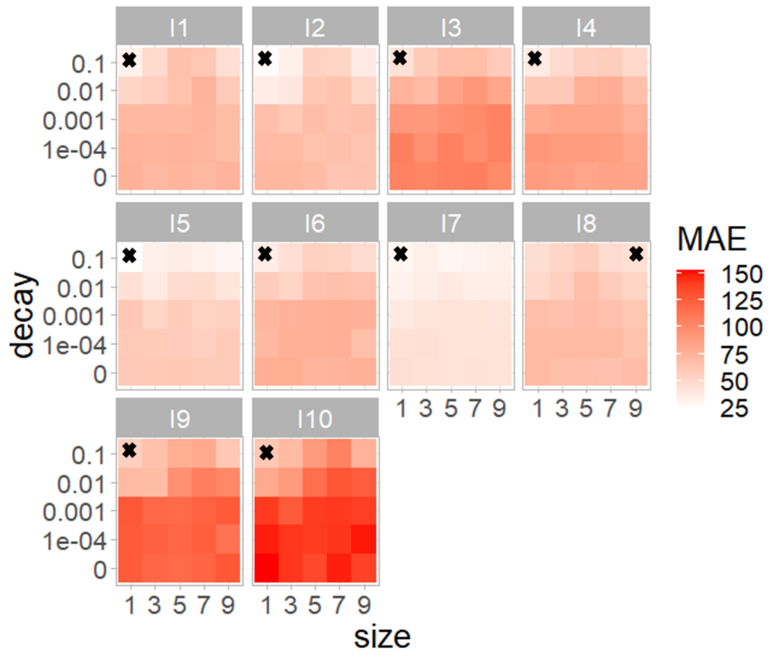


Figure 5.36 Tuning of parameters size and decay to train NNET for predicting P_g (10-Machine System)

The parameter values of the final tunned models are given in *Table 5.24*.

Table 5.24 Tuning Parameters Values (10-Machine System)

Model	Parameter	P_1	P_2	P_3	P_4	P_5
KNN	k	37	39	25	13	55
RF	mtry	50	57	57	57	57
SVM-Poly	degree	2	1	1	1	3
	scale	0.01	0.1	0.1	0.1	0.01
	cost	1	0.5	0.5	0.5	2
SVM-Radial	sigma	0.009	0.009	0.009	0.009	0.009
	cost	4	8	4	4	8
NNET	size	1	1	1	1	1
	decay	0.1	0.1	0.1	0.1	0.1
XGB-Linear	lambda	0.1	0.1	0.1	0.1	0.1
	alpha	1e-4	0.1	0	0.1	0
	#rounds	50	100	150	150	150
	eta	0.3	0.3	0.3	0.3	0.3
Model	**Parameter**	P_6	P_7	P_8	P_9	P_{10}
KNN	k	45	25	5	31	39
RF	mtry	50	57	57	44	50
SVM-Poly	degree	2	1	3	2	1
	scale	0.01	0.01	0.01	0.01	0.1
	cost	2	1	1	2	1
SVM-Radial	sigma	0.009	0.009	0.009	0.009	0.009
	cost	8	1	8	4	2
NNET	size	1	1	9	1	1
	decay	0.1	0.1	0.1	0.1	0.1
XGB-Linear	lambda	0.1	0.1	0	0.1	0.1
	alpha	0	0.1	0	0	0
	#rounds	150	100	150	100	150
	eta	0.3	0.3	0.3	0.3	0.3

Model Selection

After tuning the parameters, now, we will do between-model comparison. shows the performance metrics for the best tuned models obtained from repeated cross validation. We see that KNN is performing the worst, hence dropped, and NNET is performing the best *Table 5.25*.

Table 5.25 MAE of tunned models from repeated CV (10-Machine System)

PM	Var	GLM	KNN	RF	SVM-P	SVM-R	NN	XGB-L
MAE	P1	33.72	73.23	46.10	33.49	35.65	32.91	46.52
	P2	26.89	70.62	41.75	27.16	29.85	26.13	41.73
	P3	48.62	101.6	61.02	49.74	52.61	44.31	59.19
	P4	42.38	83.93	44.58	42.44	44.33	37.23	42.50
	P5	39.81	59.12	32.35	37.55	37.73	23.95	31.59
	P6	47.09	76.96	53.18	45.86	47.95	36.49	53.30
	P7	35.60	44.52	28.77	33.44	33.51	30.14	27.88
	P8	46.65	54.44	39.06	43.11	42.51	45.15	39.16
	P9	51.07	118.9	67.58	49.29	53.47	55.96	67.12
	P10	50.14	138.6	82.60	50.37	57.14	60.89	80.81
	Comb.	42.20	82.19	49.70	41.25	43.48	39.32	48.98

Table 5.26 MAE of tunned models from test set (10-Machine System)

PM	Var	GLM	RF	SVM-P	SVM-R	NN	XGB-L
MAE	P1	47.64	51.56	45.29	46.50	47.80	52.37
	P2	39.50	43.20	38.26	38.17	41.39	45.20
	P3	54.85	65.37	54.16	54.67	54.61	62.41
	P4	54.56	57.92	53.72	55.13	53.69	57.22
	P5	40.06	41.98	39.06	39.99	34.07	38.29
	P6	70.04	73.63	69.21	69.72	64.47	75.25
	P7	45.13	48.86	45.20	45.88	42.37	48.87
	P8	57.88	58.06	57.55	57.39	57.33	57.98
	P9	51.84	56.35	50.54	50.58	51.72	56.72
	P10	68.53	72.87	65.97	58.54	69.80	72.47
	Com.	53.00	56.98	51.90	51.66	51.73	56.68

Table 5.26 shows the MAE score of the models for the test dataset. We can see that SVM-Radial is performing better with an overall MAE score of 51.66 MW. However, for some variables, other models performing are better than SVM-Radial. So, we can further improve the MAE score to 49.87 MW if we select them for predicting P_g. *Table 5.27* shows the final selected models.

Table 5.27 Final selected models for predicting Pg (10-Machine System)

Target Variable	Predictive Model	MAE	R^2
P1	**SVM-Poly**	45.29	0.80
P2	**SVM-Radial**	38.17	0.82
P3	**SVM-Poly**	54.16	0.84
P4	**NNET**	53.69	0.81
P5	**NNET**	34.07	0.86
P6	**NNET**	64.47	0.78
P7	**NNET**	42.37	0.82
P8	**NNET**	57.33	0.72
P9	**SVM-Poly**	50.54	0.87
P10	**SVM-Radial**	58.54	0.88
Combined		**49.87**	**0.83**

5.2.10 Stage 3: Constraints Satisfaction (Algo-2)

In this stage, the solutions obtained from the first two stages are further processed to make sure none of the security constraints are violated. *Table 5.28* shows the percentage of the instances (out of 200 total instances) in the test dataset where violations were detected (along with the types of observed violations), and the solutions were adjusted to fulfill the constraints. *Table 5.29* shows the performance metrics for the final solutions for the test dataset obtained using the proposed 3-stage algorithm.

Table 5.28 Instances in the Test Set with Constraint Violations (10-Machine System)

Type of Violation / Number of Instances	Generator Limits Violation	Total Reserve Requirement Violation	Generation Load Imbalance (>1% of total load)	Transmission Line Limits Violation
Before Stage-3	0	13.5 %	90.5 %	37.5 %
After Stage-3	0	0	0	0

Table 5.29 Performance metrics for the test dataset (10-Machine System)

	Before Stage-3			After Stage-3		
Var	**MAE**	**RMSE**	**R^2**	**MAE**	**RMSE**	**R^2**
P1	45.3	101	0.80	60.3	114	0.77
P2	38.2	80	0.82	46.8	86	0.80
P3	54.1	122	0.84	61.8	119	0.85
P4	53.7	116	0.81	57.5	113	0.81
P5	34.1	86	0.86	42.2	89	0.85
P6	64.5	144	0.78	71.7	137	0.79
P7	42.4	118	0.82	50.9	121	0.80
P8	57.4	137	0.72	62.3	132	0.73
P9	50.5	122	0.87	66.3	128	0.85
P10	58.5	122	0.88	84.9	142	0.83
Comb.	49.9	116	0.83	60.5	119	0.82

Table 5.30 below shows the comparison between the predicted and observed solutions. We can see that the presented approach produces solutions which, on average, have a +0.3% higher cost. It also made sure no constraints (GLI, GLV, LLV, RR) are violated.

Table 5.30 Final results for the test dataset (10-Machine System)

	Avg. Cost [$/hour]	**Avg. Cost Difference [%]**	**Avg. GLI**	**no. of GLV**	**no. of LLV**	**Avg. Time (1000 Instances) [sec]**
Predicted (Algorithm-2)	44069	+0.3	0	0	0	~ 6
Observed (CPLEX)	43931	0	0	0	0	~ 114

5.3 Case Study 3: IEEE 118-Bus Power System

5.3.1 System Data

The system has 118 buses, 54 generating units, 99 loads and 186 transmission lines. The single line diagram of the system is shown in *Figure 5.37*. Further details of the system can be obtained from the MATPOWER case file *case118*. Bus 69 is the reference bus for the system.

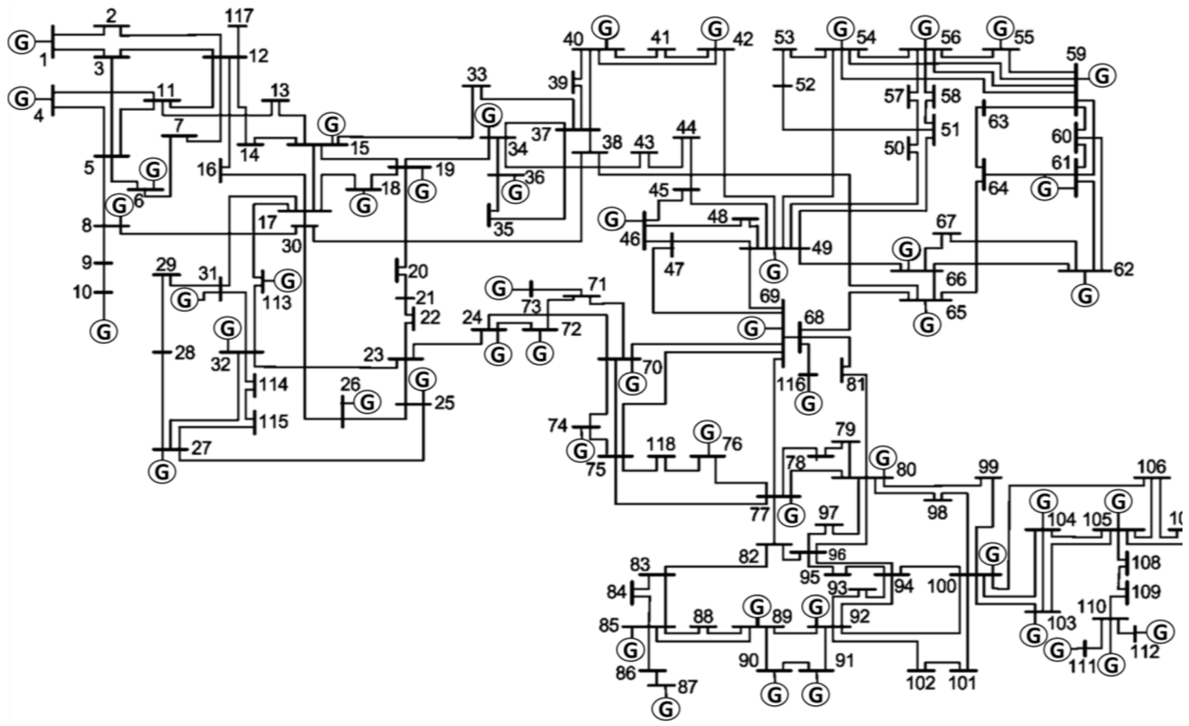


Figure 5.37 Single Line Diagram (54-Machine System)

5.3.2 MIQP Problem Formulation

From the system data, we can see that $N_B = 118$, $N_G = 54$, $N_L = 186$ and the sets $B = \{1,2,3,\ldots,118\}$, $G = \{1,2,3,\ldots,54\}$, $L = \{1,2,3,\ldots,186\}$. The single-period MIQP problem formulation for this system is similar to the one presented in section 5.2.2. There are a total of 708 constraints (i.e., 372: line limits, 118: nodal balance, 108: generator power limits, 54: spinning reserve limits, 54: binary variable restrictions, 1: reference bus angle, 1: total spinning reserve requirement).

5.3.3 Data Generation

To train the predictive model, the dataset is generated by solving the above MIQP problem using IBM CPLEX toolbox for MATLAB (this solver is used as a base line method for comparing the performance of the presented approach). A dataset of 3500 random samples, representing different scenarios of power demands and price signals, is generated assuming normal distributions for the loads (d_b) and the cost coefficients (a_g , b_g , c_g). *Figure 5.38* shows the nodal demand distributions of some of the buses.

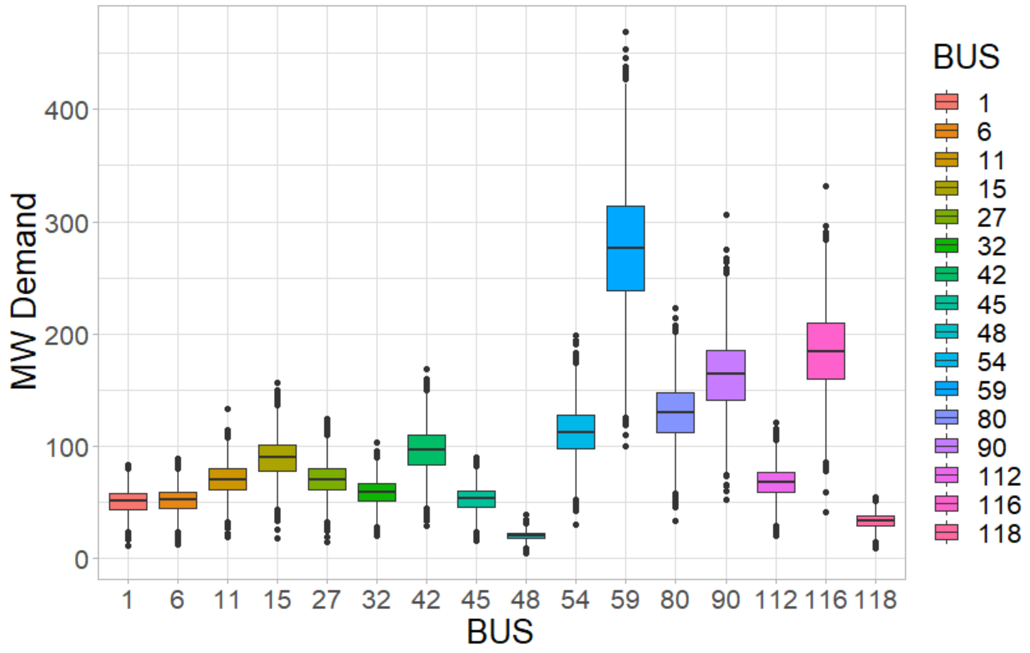


Figure 5.38 Load Distributions in the Generated Dataset (54-Machine System)

To label the generated dataset, the instances are solved through simulations by solving the SC-UC problem presented earlier. MATLAB (R2021a) and CPLEX toolbox for MATLAB are used to solve the MIQP problem presented above. This data is then used for training and testing of the learning algorithms using R-Studio.

For prediction, we will train one model for each binary variable (I_g) and another model for predicting its active power output (P_g). Hence for each generator, we will train two predictive models i.e., one to predict its commitment status and another to predict its production level.

5.3.4 Data Splitting

The dataset is divided into 80/20% train/test split using stratified random sampling. Initial model selection is done based on the train dataset performance using 10-fold repeated cross-validation. After that, the top performing model for the test dataset is selected for prediction.

5.3.5 Stage 1: Commitment Decisions (Algo-1 & Algo-2)

In this stage, we will train several predictive models for each of the binary variables I_g. These models include GLM, LDA, KNN, RF, SVM, NNET and DT. After that we will select top few models by comparing their performance on the training dataset. Then these models will be tested on the test dataset and the best performing model will be selected as the predictive model. The train set and the test set performances of the trained models for stage-1 predictions are shown in *Table 5.31*.

Table 5.31 Stage-1 Models Performance (54-Machine System)

Model	Train Accuracy	Test Accuracy
GLM	0.949	0.949
RF	0.929	0.927
SVM-Radial	0.908	0.910
LDA	0.938	0.941
KNN	0.68	0.681
NNET	0.926	0.928
DT	0.921	0.919

5.3.6 Stage 2: Production Decisions (Algo-1)

In stage-2 of Algorithm-1, we will use the predicted commitment decisions obtained from stage-1 and solve the security constrained optimal dispatch problem formulated as a QP problem in chapter 4. *Table 5.32* below shows a comparison between the solutions obtained using this approach and solutions obtained by solving the original MIQP problem using CPLEX. We can see that the presented approach produces solutions which, on average, have a +0.55% higher cost but 4 times faster in solution time.

Table 5.32 Final results for the test dataset (54-Machine System)

	Avg. Cost [$/hour]	**Avg. Cost Difference [%]**	**Avg. GLI**	**no. of GLV**	**no. of LLV**	**Avg. Time (1000 Instances) [sec]**
Predicted (Algorithm-1)	70523	+0.55	0	0	0	~ 52
Observed (CPLEX)	70133	0	0	0	0	~ 218

5.3.7 Stage 2: Production Decisions (Algo-2)

In this stage, we will train predictive models for predicting the active power of the committed generators obtained from stage 1. These models include GLM, KNN, SVM, NNET and DT. Like previous stage, we will train one model for predicting each variable P_g. *Table 5.33* shows the train set and the test set performances of the trained models for stage-2 predictions of Algorithm-2.

Table 5.33 Stage-2 Models Performance (54-Machine System)

Model	Train MAE	Test MAE
GLM	14.97	18.03
SVM-Radial	16.28	19.42
KNN	29.86	29.12
NNET	27.36	26.89
DT	29.86	29.12

5.3.8 Stage 3: Constraints Satisfaction (Algo-2)

In this stage, the solutions obtained from the first two stages are further processed to make sure none of the security constraints are violated. *Table 5.34* shows the percentage of the instances in the test dataset where violations were detected (along with the types of observed violations), and the solutions were adjusted to fulfill the constraints.

Table 5.34 Instances in the Test Set with Constraint Violations (54-Machine System)

Type of Violation / **Number of Instances**	**Generator Limits Violation**	**Total Reserve Requirement Violation**	**Generation Load Imbalance (>1% of total load)**	**Transmission Line Limits Violation**
Before Stage-3	0	25 %	83.9 %	51 %
After Stage-3	0	0	0	0

Table 5.35 below shows the comparison between the predicted and observed solutions. We can see that the presented approach produces solutions which, on average, have a +0.68% higher cost. It also made sure no constraints (GLI, GLV, LLV, RR) are violated.

Table 5.35 Final results for the test dataset (54-Machine System)

	Avg. Cost [$/hour]	**Avg. Cost Difference [%]**	**Avg. GLI**	**no. of GLV**	**no. of LLV**	**Avg. Time (1000 Instances) [sec]**
Predicted (Algorithm-2)	70613	+0.68	0	0	0	~ 41
Observed (CPLEX)	70133	0	0	0	0	~ 218

5.4 Case Study 4: South Carolina 500-Bus Power System

5.4.1 System Data

The system has 500 buses, 90 generating units, 200 loads and 584 transmission lines. This is an entirely synthetic 500 bus case, geographically situated in the northwestern part of the US state of South Carolina. The case is designed with a 345 and 138 kV transmission network to serve a load that roughly mimics the actual population of its geographic footprint. The system diagram of the system is shown in *Figure 5.39*. Further details of the system can be obtained from the MATPOWER case file *case_ACTIVSg500*. Bus 17 is the reference bus for the system.

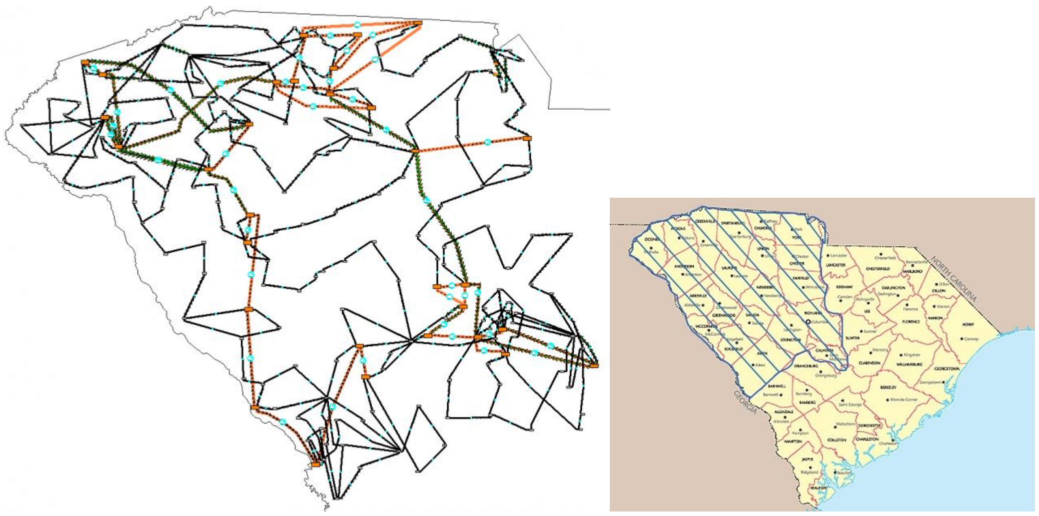


Figure 5.39 System Diagram (90-Machine System)

5.4.2 MIQP Problem Formulation

From the system data, we can see that $N_B = 500$, $N_G = 90$, $N_L = 584$ and the sets $B = \{1,2,3,\dots,500\}$, $G = \{1,2,3,\dots,90\}$, $L = \{1,2,3,\dots,584\}$. The single-period MIQP problem formulation for this system is similar to the one presented in section 5.2.2. There are a total of 2030 constraints (i.e., 1168: line limits, 500: nodal balance, 180: generator power limits, 90: spinning reserve limits, 90: binary variable restrictions, 1: reference bus angle, 1: total spinning reserve requirement).

5.4.3 Data Generation

To train the predictive model, the dataset is generated by solving the above MIQP problem using IBM CPLEX toolbox for MATLAB (this solver is used as a base line method for comparing the performance of the presented approach). A dataset of 3500 random samples, representing different scenarios of power demands and price signals, is generated assuming normal distributions for the loads (d_b) and the cost coefficients (a_g , b_g , c_g). *Figure 5.40* shows the nodal demand distributions of some of the buses.

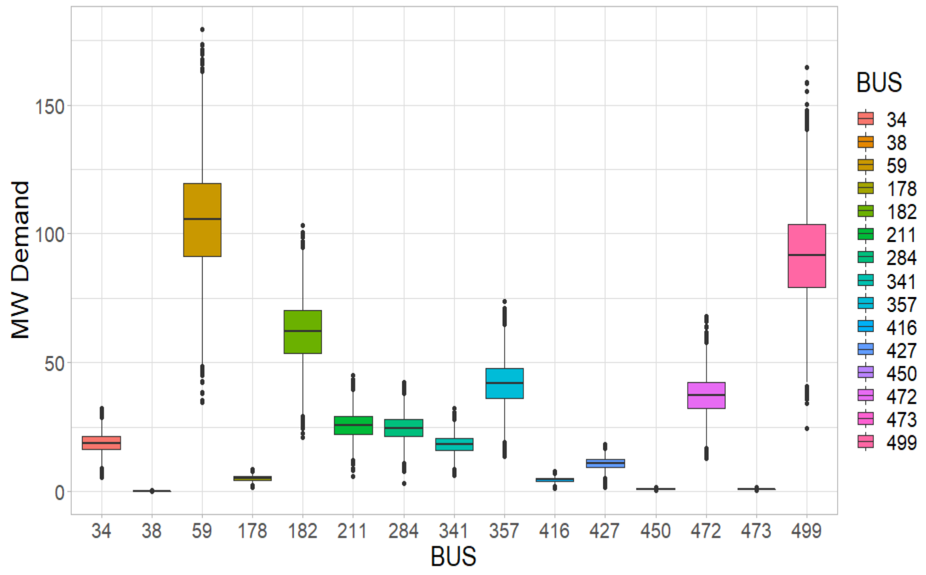


Figure 5.40 Load Distributions in the Generated Dataset (90-Machine System)

To label the generated dataset, the instances are solved through simulations by solving the SC-UC problem presented earlier. MATLAB (R2021a) and CPLEX toolbox for MATLAB are used to solve the MIQP problem presented above. This data is then used for training and testing of the learning algorithms using R-Studio.

For prediction, we will train one model for each binary variable (I_g) and another model for predicting its active power output (P_g). Hence for each generator, we will train two predictive models i.e., one to predict its commitment status and another to predict its production level.

5.4.4 Data Splitting

The dataset is divided into 80/20% train/test split using stratified random sampling. Initial model selection is done based on the train dataset performance using 10-fold repeated cross-validation. After that, the top performing model for the test dataset is selected for prediction.

5.4.5 Stage 1: Commitment Decisions (Algo-1 & Algo-2)

In this stage, we will train several predictive models for each of the binary variables I_g. These models include NB, KNN, DA, DT, RF, Ensemble-KNN, Ensemble-DA, NNET and DT. After that we will select top few models by comparing their performance on the training dataset. Then these models will be tested on the test dataset and the best performing model will be selected as the predictive model. The train set and the test set performances of the trained models for stage-1 predictions are shown in *Table 5.36*.

Table 5.36 Stage-1 Models Performance (90-Machine System)

Model	Train Accuracy	Test Accuracy
NB	0.842	0.845
KNN	0.834	0.830
DA	0.846	0.848
DT	0.843	0.843
NNET	0.848	0.848
En-KNN	0.795	0.790
En-DA	0.846	0.847
RF	0.848	0.846

5.4.6 Stage 2: Production Decisions (Algo-1)

In stage-2 of Algorithm-1, we will use the predicted commitment decisions obtained from stage-1 and solve the security constrained optimal dispatch problem formulated as a QP problem in chapter 4. *Table 5.37* below shows a comparison between the solutions obtained using this approach and solutions obtained by solving the original MIQP problem using CPLEX. We can see that the presented approach produces solutions which, on average, have a +1.6% higher cost but 7 times faster in solution time.

Table 5.37 Final results for the test dataset (90-Machine System)

	Avg. Cost [$/hour]	Avg. Cost Difference [%]	Avg. GLI	no. of GLV	no. of LLV	Avg. Time (1000 Instances) [sec]
Predicted (Algorithm-1)	7225	+ 1.6	0	0	0	~ 93
Observed (CPLEX)	7110	0	0	0	0	~ 637

5.4.7 Stage 2: Production Decisions (Algo-2)

In this stage, we will train predictive models for predicting the active power of the committed generators obtained from stage 1. These models include LM, SVM, DT, NNET, KD, and RF. Like previous stage, we will train one model for predicting each variable P_g. *Table 5.38* shows the train set and the test set performances of the trained models for stage-2 predictions of Algorithm-2. Based on their performance NNET was selected for predicting P_g variables.

Table 5.38 Stage-2 Models Performance (90-Machine System)

Model	Train Error	Test Error
LM	9.32	9.37
SVM	9.23	9.30
DT	9.10	9.04
KD	9.09	9.11
NNET	8.85	8.88
RF	8.93	8.98

5.4.8 Stage 3: Constraints Satisfaction (Algo-2)

In this stage, the solutions obtained from the first two stages are further processed to make sure none of the security constraints are violated. *Table 5.39* shows the percentage of the instances in the test dataset where violations were detected (along with the types of observed violations), and the solutions were adjusted to fulfill the constraints.

Table 5.39 Instances in the Test Set with Constraint Violations (90-Machine System)

Type of Violation / Number of Instances	Generator Limits Violation	Total Reserve Requirement Violation	Generation Load Imbalance (>1% of total load)	Transmission Line Limits Violation
Before Stage-3	0	48.6 %	64.6 %	8.2 %
After Stage-3	0	0	0	0

Table 5.40 below shows the comparison between the predicted and observed solutions. We can see that the presented approach produces solutions which, on average, have a +2% higher cost. It also made sure no constraints (GLI, GLV, LLV, RR) are violated.

Table 5.40 Final results for the test dataset (90-Machine System)

	Avg. Cost [$/hour]	**Avg. Cost Difference [%]**	**Avg. GLI**	**no. of GLV**	**no. of LLV**	**Avg. Time (1000 Instances) [sec]**
Predicted (Algorithm-2)	7252	+ 2.0	0	0	0	~ 45
Observed (CPLEX)	7110	0	0	0	0	~ 637

5.5 Case Study 5: IEEE 9-Bus Power System (12-Hour)

So far, we have implemented the proposed approaches to solve a single period UCP. Now, we will present results for solving a multi-period UCP problem using a standard IEEE power system. The system data is given below.

5.5.1 System Data

The system has 9 buses, 3 generating units, 3 loads and 9 transmission lines. The system diagram of the system is shown in *Figure 5.41*. Further details of the system can be obtained from the MATPOWER case file *case9*. Bus 1 is the reference bus for the system.

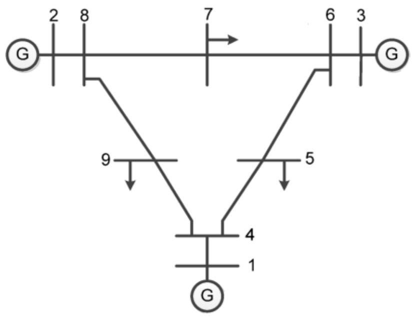


Figure 5.41 System Diagram (IEEE 9-Bus System)

5.5.2 Multi-Period MIQP Problem Formulation

We will be solving the UCP for the time horizon of 12 hours. Hence, we can see that $N_B = 9$, $N_G = 3$, $N_L = 9$, $N_T = 12$ and the sets $B = \{1,2,3,\dots,9\}$, $G = \{1,2,3\}$, $L = \{1,2,3,\dots,9\}$ and $T = \{1,2,3,\dots,12\}$. The multi-period MIQP problem for this system is formulated as given below. There are a total of 41 constraints for each hour in the time horizon (i.e., 18: line limits, 9: nodal balance, 6: generator power limits, 3: spinning reserve limits, 3: binary variable restrictions, 1: reference bus angle, 1: total spinning reserve requirement). For a period of 12-h, the total number of non-temporal constraints will be $41x12 = 492$, in addition to the temporal constraints like min up/down time and ramp up/down limits. We can see that due to the involvement of multiple time periods, the problem size increases tremendously even for a small system like the one presented here.

$$\underset{I_{gt}, P_{gt}, R_{gt}, \delta_{bt}}{minimize:} \quad \sum_{t\in T}\sum_{g\in G}\left(PC_{gt} + SUC_{gt}\right)I_{gt}$$

$$subject\ to: \quad \sum_{g\in G_b} P_{gt} = d_{bt} + \sum_{l\in L_b} P_{lt} \qquad \forall t \in T, \forall b \in B$$

$$-\overline{P_l} \le P_{lt} = \frac{\delta_{lt}}{X_l} \le \overline{P_l} \qquad \forall t \in T, \forall l \in L$$

$$\delta_{b_{ref}} = 0$$

$$\underline{P_g} I_{gt} \le P_{gt} \le \overline{P_g} I_{gt} \qquad \forall t \in T, \forall g \in G$$

$$\sum_{g\in G} R_{gt}\, I_{gt} \ge R_t \qquad \forall t \in T$$

$$\left(T^{on}_{g(t-1)} - T^{up}_{g}\right)\left(I_{g(t-1)} - I_{gt}\right) \ge 0 \qquad \forall g \in G, \forall t \in T$$

$$\left(T^{off}_{g(t-1)} - T^{down}_{g}\right)\left(I_{g(t-1)} - I_{gt}\right) \ge 0 \qquad \forall g \in G, \forall t \in T$$

$$-R^{down} \le P_{gt} - P_{g(t-1)} \le R^{up} \qquad \forall g \in G, \forall t \in T$$

$$P_{gt} + R_{gt} \le \left(\overline{P_g} - \underline{P_g}\right) I_{gt} \qquad \forall t \in T, \forall g \in G$$

$$I_{gt} \in \{0,1\} \qquad \forall t \in T, \forall g \in G$$

$$P_{gt} \geq 0 \qquad t \in T, \forall g \in G$$

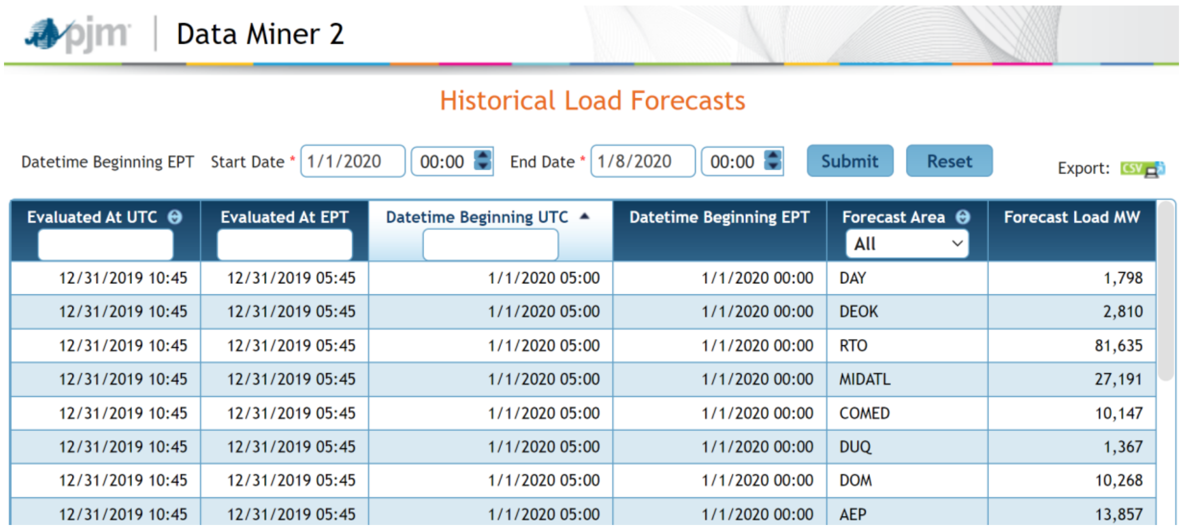
Here, the angles (δ_{lt}) are in radians while all the other non-binary variables are in per unit. d_{bt} is the forecasted net power demand at bus b in time period t, P_{lt} is the power injection from bus b into the line l in time t, δ_{lt} is the angle difference between bus b and the bus at the other end of the line l in time t, $\overline{P_l}$ and X_l are the maximum power carrying capacity and reactance of the line l, $\delta_{b_{ref}}$ is the phase angle of the reference bus of the system, $\overline{P_g}$, $\underline{P_g}$ and R_{gt} are the maximum power output, minimum power output, and the spinning reserve of the generator g in time t and R_t is the total minimum reserve requirements of the system in time period t.

5.5.3 Data Generation

To train the predictive model, the dataset is generated by solving the above MIQP problem using IBM CPLEX toolbox for MATLAB (this solver is used as a base line method for comparing the performance of the presented approach). A dataset of 1000 random samples, representing different scenarios of power demands and price signals, is generated. To generate different scenarios, we considered three types of uncertainties:

(1) Uncertainty in the System-Wide Temporal Load Profile: To generate uncertainty in the temporal profile, we analyzed the PJM historical load forecast hourly demand data for the 1st week of January 2020. The data was obtained from the PJM Data Miner 2 as shown in *Figure 5.42*. There are 10 load areas (i.e., AEP, APS, ATSI, COMED, DAY, DEOK, DOM, DUQ, EKPC and MIDATL) in the PJM RTO, similar to 10 buses in a power system, while the total RTO load is given under the area named as RTO and is the sum of demands of all the load areas. For data generation, we used normalized (to scale down w.r.t IEEE 9-bus system-wide load) 7-day average

12-hour temporal load profile of the RTO load area as our system's temporal load profile. The 24-hour temporal profiles for the PJM RTO for 7 days are shown in *Figure 5.43*.



Historical Load Forecasts

Datetime Beginning EPT Start Date * 1/1/2020 00:00 End Date * 1/8/2020 00:00

Evaluated At UTC	Evaluated At EPT	Datetime Beginning UTC	Datetime Beginning EPT	Forecast Area	Forecast Load MW
				All	
12/31/2019 10:45	12/31/2019 05:45	1/1/2020 05:00	1/1/2020 00:00	DAY	1,798
12/31/2019 10:45	12/31/2019 05:45	1/1/2020 05:00	1/1/2020 00:00	DEOK	2,810
12/31/2019 10:45	12/31/2019 05:45	1/1/2020 05:00	1/1/2020 00:00	RTO	81,635
12/31/2019 10:45	12/31/2019 05:45	1/1/2020 05:00	1/1/2020 00:00	MIDATL	27,191
12/31/2019 10:45	12/31/2019 05:45	1/1/2020 05:00	1/1/2020 00:00	COMED	10,147
12/31/2019 10:45	12/31/2019 05:45	1/1/2020 05:00	1/1/2020 00:00	DUQ	1,367
12/31/2019 10:45	12/31/2019 05:45	1/1/2020 05:00	1/1/2020 00:00	DOM	10,268
12/31/2019 10:45	12/31/2019 05:45	1/1/2020 05:00	1/1/2020 00:00	AEP	13,857

Figure 5.42 PJM Data Miner 2

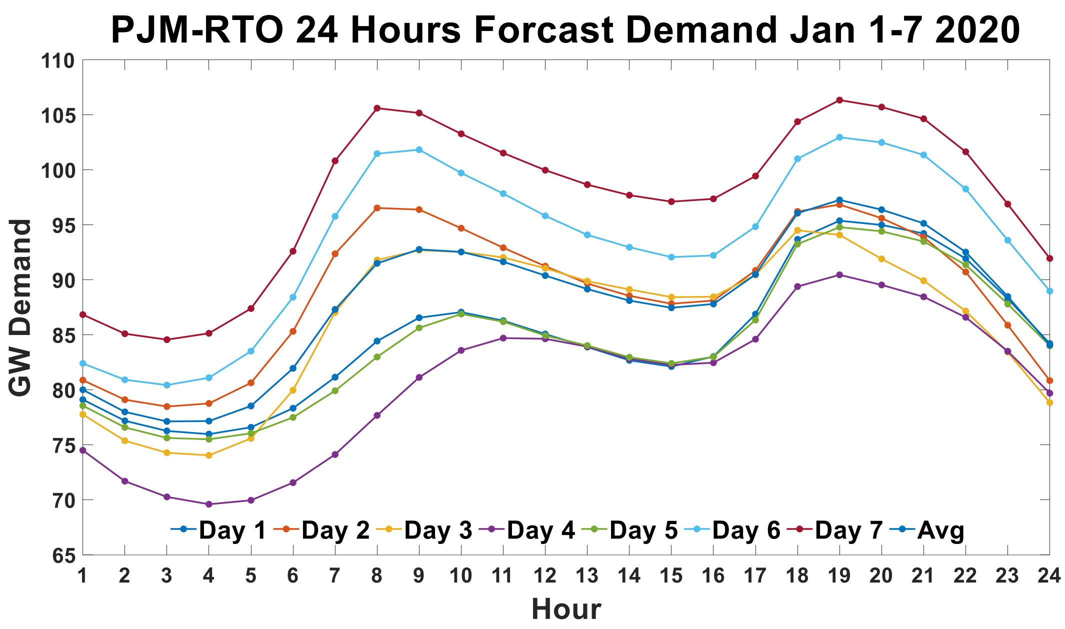


Figure 5.43 PJM RTO 7 Day Temporal Load Profiles (January 1-7, 2020)

Based on our analysis of the profiles, the system load varies within +/- 10% of the daily load average of the profile. So, to represent this uncertainty in generated dataset, we used random variables α_t drawn independently from the uniform distribution in the interval [0.9, 1.1]. The spread of the load profiles in the generated dataset is shown in *Figure 5.44*. The histogram of the nodal MW demands in the generated dataset are shown in *Figure 5.45*.

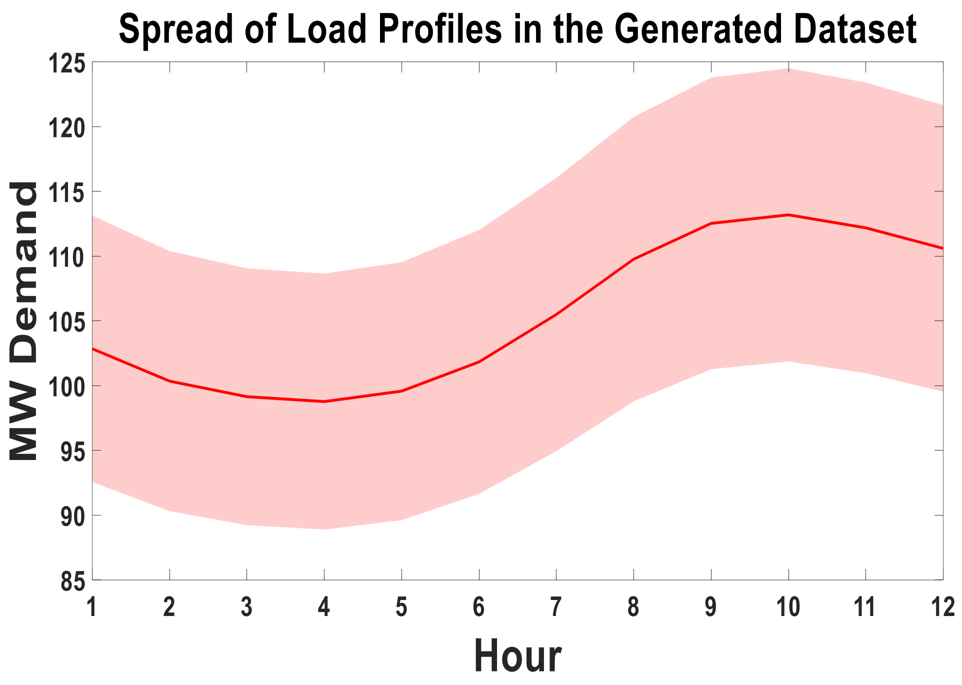


Figure 5.44 Spread of the Temporal Load Profiles in the Generated Dataset

(2) Uncertainty in the Geographical Load Distribution: In a power system, each load bus is responsible for a certain percentage of the total system load. Based on our analysis of the PJM data for 7 days, the nodal demand sharing percentages vary within +/- 2% of its average load sharing percentage. However, to generate more uncertainty in the generated dataset, we used uncertainty of +/- 10%. Hence, to generate variations in these nodal load percentages, b (no. of load buses) random numbers β_b were independently drawn from the uniform distribution in the interval [0.9, 1.1].

(3) Uncertainty in the Production Cost: Based on the PJM data analysis, the price bids of the generators taking part in the day ahead energy market, do not vary more than +/- 5 % over different days. In fact, most of the time the cost data stays same throughout a day. To generate uncertainty in the production cost coefficients a_g, b_g and c_g, we used g (no. of generators) random variables γ_g independently drawn from normal random distributions with a standard deviation of 10% of its mean value for each generator g. The histogram of the cost coefficients in the generated dataset are shown in *Figure 5.46*.

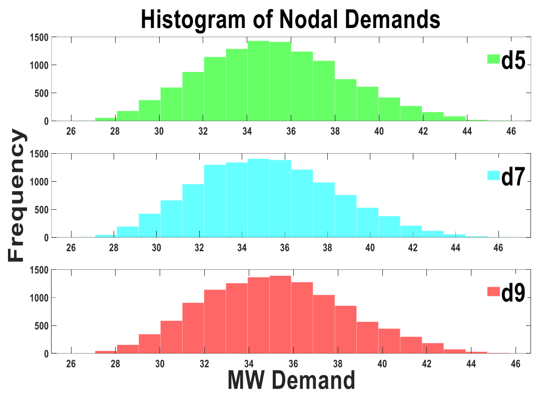


Figure 5.45 Histogram of the Nodal Demands in the Generated Dataset (IEEE 9-Bus System)

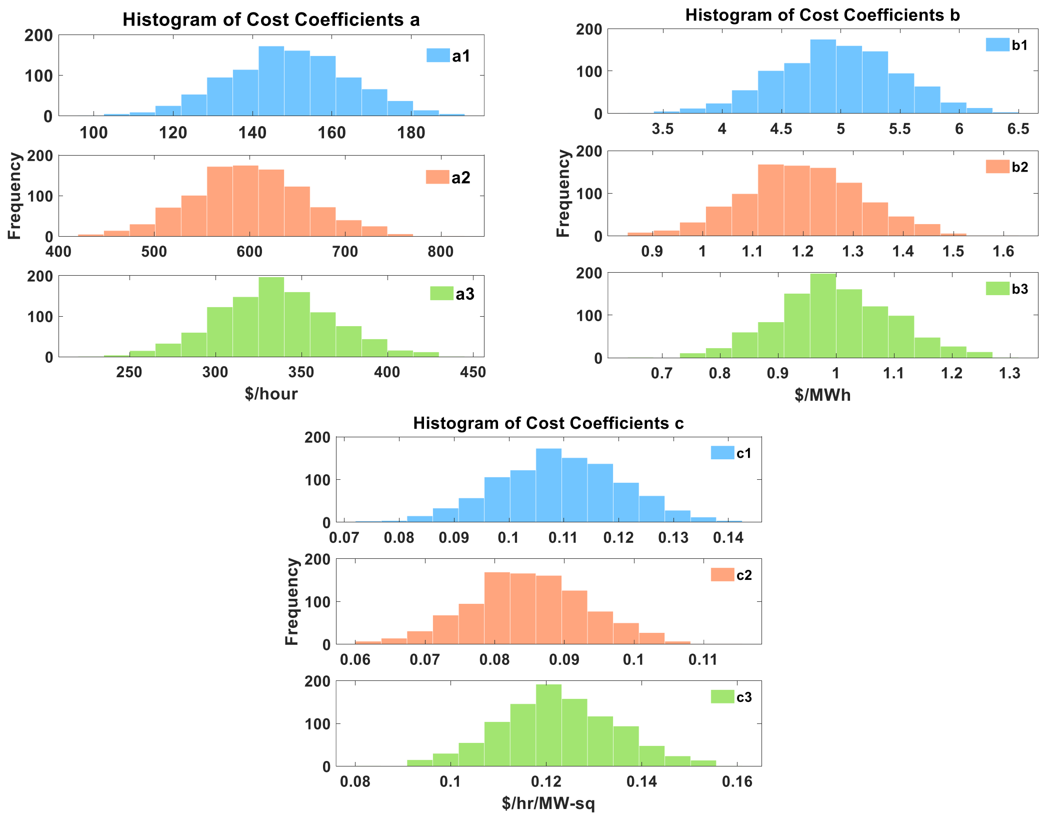


Figure 5.46 Histogram of the Cost Coefficients (IEEE 9-Bus System)

To label the generated dataset, the instances are solved through simulations by solving the SC-UC problem presented earlier. MATLAB (R2021a) and CPLEX toolbox for MATLAB are used to solve the MIQP problem presented above. This data is then used for training and testing of the learning algorithms using R-Studio.

For prediction, we will train one model for each binary variable (I_{gt}) and another model for predicting its active power output (P_{gt}). Hence for each generator, we will train two predictive models in each hour i.e., one to predict its commitment and another to predict its production level.

5.5.4 Data Splitting

The dataset is divided into 80/20% train/test split using stratified random sampling. Initial model selection is done based on the train dataset performance using 10-fold repeated cross-validation. After that, the top performing model for the test dataset is selected for prediction.

5.5.5 Stage 1: Commitment Decisions (Algo-1 & Algo-2)

In this stage, we will train several predictive models for each of the binary variables I_g. These models include NB, KNN, DA, DT, RF, Ensemble-KNN, Ensemble-DA, NNET and RF. After that we will select top few models by comparing their performance on the training dataset. Then these models will be tested on the test dataset and the best performing model will be selected as the predictive model. The train set and the test set performances of the trained models for stage-1 predictions are shown in *Table 5.41*.

Table 5.41 Stage-1 Models Performance (IEEE 9-Bus System)

Model	Train Accuracy	Test Accuracy
NB	0.912	0.940
KNN	0.906	0.901
DA	0.928	0.948
DT	0.895	0.914
NNET	0.945	0.971
En-KNN	0.690	0.632
En-DA	0.921	0.926
RF	0.933	0.939

5.5.6 Stage 2: Production Decisions (Algo-1)

In stage-2 of Algorithm-1, we will use the predicted commitment decisions obtained from stage-1 and solve the security constrained optimal dispatch problem formulated as a QP problem in chapter 4. *Table 5.42* below shows a comparison between the solutions obtained using this approach and solutions obtained by solving the original MIQP problem using CPLEX. We can see that the presented approach produces solutions which, on average, have a +0.3% higher cost but almost 2 times faster in solution time.

Table 5.42 Final results for the test dataset (IEEE 9-Bus System)

	Avg. Cost [$/hour]	Avg. Cost Difference [%]	Avg. GLI	no. of GLV	no. of LLV	Avg. Time (1000 Instances) [sec]
Predicted (Algorithm-1)	20802	+ 0.3	0	0	0	~ 127
CPLEX	20731	0	0	0	0	~ 232

5.5.7 Stage 2: Production Decisions (Algo-2)

In this stage, we will train predictive models for predicting the active power of the committed generators obtained from stage 1. These models include LM, SVM, DT, NNET, KD, and RF. Like previous stage, we will train one model for predicting each variable P_g. *Table 5.43* shows the train set and the test set performances of the trained models for stage-2 predictions of Algorithm-2.

Table 5.43 Stage-2 Models Performance (IEEE 9-Bus System)

Model	Train Error	Test Error
LM	4.24	4.20
SVM	4.18	4.22
DT	4.39	4.17
KD	4.36	4.28
NNET	3.85	3.66
RF	4.03	3.67

5.5.8 Stage 3: Constraints Satisfaction (Algo-2)

In this stage, the solutions obtained from the first two stages are further processed to make sure none of the security constraints are violated. *Table 5.44* shows the percentage of the instances in the test dataset where violations were detected (along with the types of observed violations), and the solutions were adjusted to fulfill the constraints.

Table 5.44 Instances in the Test Set with Constraint Violations (IEEE 9-Bus System)

Type of Violation / Number of Instances	Generator Limits Violation	Total Reserve Requirement Violation	Generation Load Imbalance (>1% of total load)	Transmission Line Limits Violation
Before Stage-3	0 %	0 %	65.4 %	0 %
After Stage-3	0 %	0 %	0 %	0 %

Table 5.45 below shows the comparison between the predicted and observed solutions. We can see that the presented approach produces solutions which, on average, have a +0.4% higher cost. It also made sure no constraints (GLI, GLV, LLV, RR) are violated.

Table 5.45 Final results for the test dataset (IEEE 9-Bus System)

	Avg. Cost [$/hour]	Avg. Cost Difference [%]	Avg. GLI	no. of GLV	no. of LLV	Avg. Time (1000 Instances) [sec]
Predicted (Algorithm-2)	20814	+0.4	0	0	0	~ 3
CPLEX	20731	0	0	0	0	~ 232

5.6 Performance Comparison

5.6.1 Cost Comparison

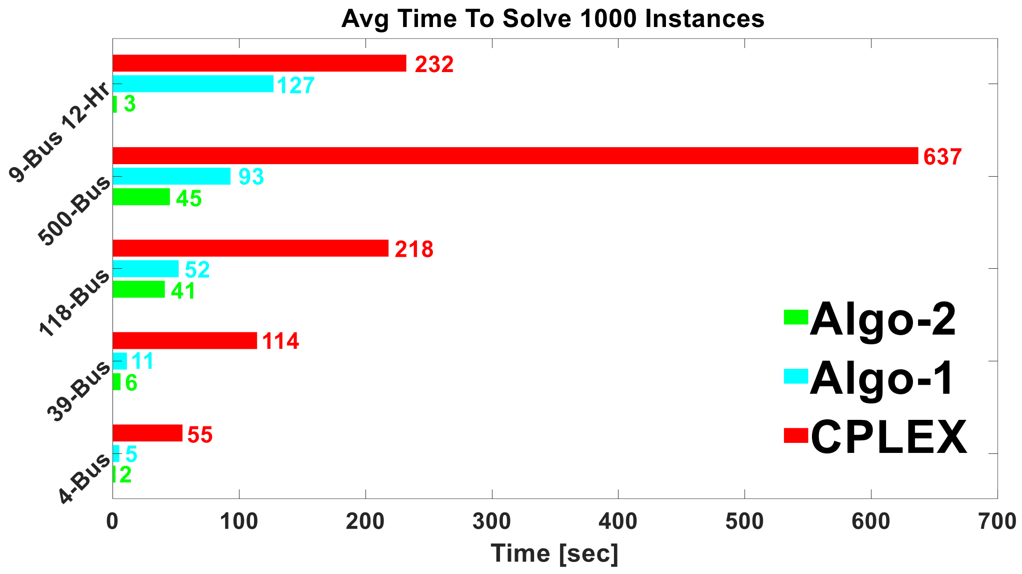
First, we will perform the cost comparison between the solutions obtained from the proposed approaches and from the conventional MIQP approach solved using IBM CPLEX MATLAB Toolbox. *Table 5.46* presents the summary of the cost comparison for the test dataset for each case study. It is clear from the table that the proposed approaches predicted the solutions while staying within 1% of the optimality gap on average.

Table 5.46 Cost Comparison with MIQP approach

Case Study	Sys	Approach	Avg Cost [$/hour]	Avg Cost Difference [$]	Avg Cost Difference [%]
1	**4 Bus**	**Algo-1**	5416	+11	+0.2
		Algo-2	5424	+19	+0.3
		CPLEX MIQP	5405	0	
2	**39 Bus**	**Algo-1**	44021	+90	+0.2
		Algo-2	44069	+138	+0.3
		CPLEX MIQP	43931	0	
3	**118 Bus**	**Algo-1**	70523	+390	+0.56
		Algo-2	70613	+480	+0.68
		CPLEX MIQP	70133	0	
4	**500 Bus**	**Algo-1**	7225	+115	+1.6
		Algo-2	7252	+142	+2.0
		CPLEX MIQP	7110	0	
5	**9 Bus (12-h)**	**Algo-1**	20802	+71	+0.3
		Algo-2	20814	+83	+0.4
		CPLEX MIQP	20731	0	

5.6.2 Computational Time Comparison

Next, we will do the computational time comparison. We will solve 100 instances from the test dataset and observe the time taken by an approach to obtain the solutions. This will be repeated several times and then we will find the average time to solve 100 instances. From there we will calculate the average time to solve 1000 instances. *Figure 5.47* shows the average time taken by the approaches to solve 1000 instances. It can be observed that the proposed data driven approaches are considerably faster than the CPLEX MIQP solver.



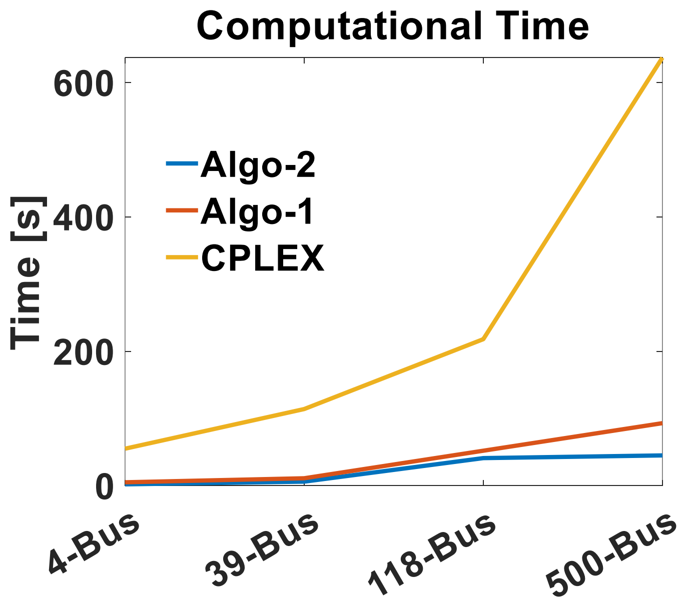


Figure 5.47 Computational Time Comparison

CHAPTER

CONCLUSION AND FUTURE WORK

6

6.1 Conclusion

In this thesis, two data driven algorithms are proposed to solve a SC-UC. The first algorithm (*Algo-1*) has two stages while the second algorithm (*Algo-2*) has three stages. In the first stage of both the algorithms, classification predictive models are trained using supervised learning methods to predict the commitment status of each of the generators. In stage-2 of *Algo-1*, a security constrained economic dispatch (SC-ED) problem is solved to obtain the production decisions of the committed generators. In stage-2 of *Algo-2*, regression predictive models are trained to predict the production status of each of the committed generators followed by the third stage of *Algo-2,* in which a post processing algorithm polishes the obtained generation schedule and performs a feasibility check of the obtained predicted solution to makes sure all security constraints are satisfied. The proposed approaches were validated through simulations on several power systems, i.e., 4-bus, 9-bus, 39-bus, 118-bus, and 500-bus systems, to study the effectiveness of the proposed approaches. The accomplishments of the completed work are outlined below:

- A modeling framework has been developed to solve a SC-UC using predictive modeling techniques. In this regard, we proposed two algorithms which can be used in parallel to an MIP solver either as alternative back-up methods or to provide a warm start for solving a full MIP SC-UC problem.
- The computational performance of the proposed approaches was validated through numerical simulations using several test systems i.e., a 4-bus 3-machine system, and IEEE 9-bus, 39-bus, 118-bus, and 500-bus power systems.
- We investigated the performance of the proposed approaches on large power systems having tens of generators and a lot of constraints to satisfy e.g., IEEE 118-bus 54-machine and South Carolina 500-bus 90-machine power systems.
- We also implemented and validated the performance of the presented approaches to solve a multi-period SC-UC problem, by incorporating the intertemporal constraints, using an IEEE 9-bus 3-machine power system, and solving the UCP for a time horizon of 12 hours.

6.2 Future Work

As discussed in chapter 5, the performance of the proposed approaches is prominent especially for a large power system. So far, we have not considered variations in the power system topology and uncertainty in the fleet size of the generators taking part in the bidding process. Moreover, we would also like to study the effectiveness of the approaches for systems having renewable energy sources along with the electric vehicles integration in the power system. Including contingencies of the transmission lines and the generating stations is another aspect that we would like to look at. These are the research areas which can be considered in the future work of this research study. Some of the research areas which can be considered in the future work of this study are given below:

- Investigate the performance of the proposed approaches to solve a multi-period UCP for a large power system having tens of generators and a lot of constraints to satisfy.
- Study the effectiveness of the approaches in the presence of renewable energy sources and dispatchable loads.
- Furthermore, in the post processing stage of *Algo-2*, we would like to implement a decentralized framework for re-scheduling the generators' production decision to fulfill transmission constraints. For this purpose, we can use *fast community detection algorithm* proposed by the authors in an earlier research work [59].
- Implementation of the proposed approaches to solve a DSC-UC in a decentralized manner.
- Implementation and investigation of the performance of the proposed approaches with the involvement of grid scale EV charging stations with bi-directional power flow.
- Include uncertainties in the power network topology by considering contingencies and availability of the generators.
- Consider AC power flow in the UCP formulation instead of DC power flow.

APPENDICES

APPENDIX A: SYSTEM DATA

4-Bus 3-Machine System Data

```
% %% MATPOWER Case Format : Version 2
% mpc.version = '2';
%
% %%-----  Power Flow Data  -----%%
% %% system MVA base
% mpc.baseMVA = 100;
%
% %% bus data
% % bus_i   type    Pd  Qd  Gs  Bs  area    Vm  Va  baseKV  zone    Vmax    Vmin
% mpc.bus = [
%   1       3       0   0   0   0   1       1   0   138     1       1.1 0.9; %ref bus
%   2       2       300 0   0   0   1       1   0   138     1       1.1 0.9;
%   3       2       0   0   0   0   1       1   0   138     1       1.1 0.9;
%   4       1       200 0   0   0   1       1   0   138     1       1.1 0.9;
% ];
%
% %% generator data
% % bus Pg      Qg  Qmax Qmin   Vg  mBase   status  Pmax    Pmin    Pc1 Pc2 Qc1min  Qc1max
Qc2min  Qc2max  ramp_agc    ramp_10 ramp_30 ramp_q  apf
% mpc.gen = [
%   1   200 0   300 -300    1   100     1       500     0   0   0   0   0   0   0   0   0   0   0
0;
%   2   100 0   300 -300    1   100     1       500     0   0   0   0   0   0   0   0   0   0   0
0;
%     3 200 0   300 -300    1   100     1       500     0   0   0   0   0   0   0   0   0   0   0
0;
% ];
%
% %% branch data
% % fbus    tbus    r   x       b   rateA   rateB   rateC   ratio   angle   status  angmin
angmax
% mpc.branch = [
%   1       2       0   0.25    0   200 200 200         0       0       1       -360    360;
%     1       3       0 0.25    0   200 200 200         0       0       1       -360    360;
%   1       4       0   0.25    0   200 200 200         0       0       1       -360    360;
%   2       3       0   0.25    0   200 200 200         0       0       1       -360    360;
%   3       4       0   0.25    0   200 200 200         0       0       1       -360    360;
% ];
%
% %%-----  OPF Data  -----%%
% %% generator cost data
% % 1   startup shutdown    n   x1  y1  ... xn  yn
% % 2   startup shutdown    n   c(n-1)  ... c0
% mpc.gencost = [
%   2   0   0   3   0.01    0.3 0.2;
%   2   0   0   3   0.01    0.3 0.2;
%   2   0   0   3   0.01    0.3 0.2;
% ];
%
% %%-----  Reserve Data  -----%%
% %% reserve zones, element i, j is 1 if gen j is in zone i, 0 otherwise
% mpc.reserves.zones = [
%   1   1   1;
% ];
```

```
%
% %% reserve requirements for each zone in MW
% mpc.reserves.req    = 1;
%
% %% reserve costs in $/MW for each gen that belongs to at least 1 zone
% %% (same order as gens, but skipping any gen that does not belong to any zone)
% mpc.reserves.cost  = [     0;   0;   0;];
% %mpc.reserves.cost   = [    6;   5;   4;   3;   2;   1    ];
%
% %% OPTIONAL max reserve quantities for each gen that belongs to at least 1 zone
% %% (same order as gens, but skipping any gen that does not belong to any zone)
% mpc.reserves.qty    = [500;500;500];
```

39-Bus 10-Machine System Data

See MATPOWER case file “case39”

118-Bus 54-Machine System Data

See MATPOWER case file “case118”

500-Bus 90-Machine System Data

See MATPOWER case file “case_ACTIVSg500”

9-Bus 3-Machine System Data

Matpower case file:

See MATPOWER case file “case9”, “ex_case3a” and “ex_case3b”

```
%CASE9    Power flow data for 9 bus, 3 generator case.
% %   Please see CASEFORMAT for details on the case file format.
% %
% %   Based on data from p. 70 of:
% %
% %   Chow, J. H., editor. Time-Scale Modeling of Dynamic Networks with
% %   Applications to Power Systems. Springer-Verlag, 1982.
% %   Part of the Lecture Notes in Control and Information Sciences book
% %   series (LNCIS, volume 46)
% %
% %   which in turn appears to come from:
% %
% %   R.P. Schulz, A.E. Turner and D.N. Ewart, "Long Term Power System
% %   Dynamics," EPRI Report 90-7-0, Palo Alto, California, 1974.
%
% %   MATPOWER
%
% %% MATPOWER Case Format : Version 2
% mpc.version = '2';
%
% %%-----  Power Flow Data  -----%%
% %% system MVA base
% mpc.baseMVA = 100;
%
% %% bus data
% %         bus_i type Pd         Qd         Gs         Bs         area Vm    Va         baseKV     zone       Vmax       Vmin
% mpc.bus = [
%       1       3   0   0       0       0       1       1       0       345     1       1.1     0.9;
%       2       2   0   0       0       0       1       1       0       345     1       1.1     0.9;
%       3       2   0   0       0       0       1       1       0       345     1       1.1     0.9;
%       4       1   0   0       0       0       1       1       0       345     1       1.1     0.9;
%       5       1   90  30      0       0       1       1       0       345     1       1.1     0.9;
%       6       1   0   0       0       0       1       1       0       345     1       1.1     0.9;
%       7       1   100 35      0       0       1       1       0       345     1       1.1     0.9;
%       8       1   0   0       0       0       1       1       0       345     1       1.1     0.9;
%       9       1   125 50      0       0       1       1       0       345     1       1.1     0.9;
% ];
% % mpc.Pd = [339,332,329,328,323,331,340,354,346,344,342,338,334,330,326,326,330,353,359,359,357,348,337,324];
% %% generator data
% %     bus     Pg    Qg    Qmax Qmin     Vg    mBase status    Pmax    Pmin    Pc1     Pc2     Qc1min  Qc1max
        Qc2min  Qc2max  ramp_agc  ramp_10  ramp_30  ramp_q  apf
% mpc.gen = [
%       1       72.3*0  27.03   300     -300    1.04    100     1       250     0       0       0
        0       0       0       0       0       250     250     0       0;
%       2       163*0   6.54    300     -300    1.025   100     1       300     0       0       0
        0       0       0       0       0       300     300     0       0;
%       3       85*0    -10.95  300     -300    1.025   100     1       270     0       0       0
        0       0       0       0       0       270     270     0       0;
% ];
%
% %% branch data
% %     fbus tbus r     x     b     rateA rateB rateC ratio   angle   status  angmin  angmax
% mpc.branch = [
%       1       4       0       0.0576          0     250     250     250     0 0     1       -360    360;
%       4       5       0.017*0   0.092       0.158*0   250     250     250     0 0     1       -360    360;
%       5       6       0.039*0   0.17        0.358*0   150     150     150     0 0     1       -360    360;
%       3       6       0       0.0586          0     300     300     300     0 0     1       -360    360;
%       6       7       0.0119*0  0.1008      0.209*0   150     150     150     0 0     1       -360    360;
%       7       8       0.0085*0  0.072       0.149*0   250     250     250     0 0     1       -360    360;
%       8       2       0       0.0625          0     250     250     250     0 0     1       -360    360;
%       8       9       0.032*0   0.161       0.306*0   250     250     250     0 0     1       -360    360;
%       9       4       0.01*0    0.085       0.176*0   250     250     250     0 0     1       -360    360;
% ];
%
% %%-----  OPF Data  -----%%
% %% generator cost data
% %     1       startup   shutdown  n       x1      y1      ...     xn      yn
% %     2       startup   shutdown  n       c(n-1)  ...     c0
% mpc.gencost = [
%       2       1500    0     3   0.11      5       150;
%       2       2000    0     3   0.085     1.2     600;
%       2       3000    0     3   0.1225    1       335;
% ];
%
%
%
% %%-----  Reserve Data  -----%%
% %% reserve zones, element i, j is 1 if gen j is in zone i, 0 otherwise
% mpc.reserves.zones = [
%       1       1       1;
% ];
%
% %% reserve requirements for each zone in MW
% mpc.reserves.req   = 1;
%
% %% reserve costs in $/MW for each gen that belongs to at least 1 zone
% %% (same order as gens, but skipping any gen that does not belong to any zone)
% mpc.reserves.cost  = [        0;      0;      0];
%
%
% %% OPTIONAL max reserve quantities for each gen that belongs to at least 1 zone
% %% (same order as gens, but skipping any gen that does not belong to any zone)
% mpc.reserves.qty   = [250;300;270];
```

```
%
% %% xgdata [MUg MDg UT0 DT0]
% mpc.xgd = [
%     2 1 0 0;
%     2 1 0 0;
%     3 1 0 0;
%     ];
```

Extra Generator Data:

```
function xgd_table = xgd_9B3G9L(mpc)
% %EX_XGD_UC  Example xGenData table for stochastic unit commitment.
%
% %   MOST
% %   Copyright (c) 2015-2016, Power Systems Engineering Research Center (PSERC)
% %   by Ray Zimmerman, PSERC Cornell
% %
% %   This file is part of MOST.
% %   Covered by the 3-clause BSD License (see LICENSE file for details).
% %   See https://github.com/MATPOWER/most for more info.
%
% %% initial xGenData
% xgd_table.colnames = {
%     'CommitKey', ...
%         'CommitSched', ...
%             'MinUp', ...
%                 'MinDown', ...
%                     'PositiveActiveReservePrice', ...
%                         'PositiveActiveReserveQuantity', ...
%                             'NegativeActiveReservePrice', ...
%                                 'NegativeActiveReserveQuantity', ...
%                                     'PositiveActiveDeltaPrice', ...
%                                         'NegativeActiveDeltaPrice', ...
%                                             'PositiveLoadFollowReservePrice', ...
%                                                 'PositiveLoadFollowReserveQuantity', ...
%                                                   'NegativeLoadFollowReservePrice', ...
%                                                       'NegativeLoadFollowReserveQuantity', ...
% };
% xgd_table.data = [
%     1  1  2  1  0   500*0.5*inf    0    500*0.5*inf    0    0    0    500*0.5*inf  0    500*0.5*inf;
%     1  1  2  1  0   600*0.5*inf    0    600*0.5*inf    0    0    0    600*0.5*inf  0    600*0.5*inf;
%     1  1  3  1  0   540*0.5*inf    0    540*0.5*inf    0    0    0    540*0.5*inf  0    540*0.5*inf;
% ];
```

Load Profile Variations Data:

```
function loadprofile = load_profile_9B3G9L
% %EX_LOAD_PROFILE  Example load profile data file for stochastic unit commitment.
%
% %   MOST
% %   Copyright (c) 2015-2016, Power Systems Engineering Research Center (PSERC)
% %   by Ray Zimmerman, PSERC Cornell
% %
% %   This file is part of MOST.
% %   Covered by the 3-clause BSD License (see LICENSE file for details).
% %   See https://github.com/MATPOWER/most for more info.
%
% %% define constants
% [CT_LABEL, CT_PROB, CT_TABLE, CT_TBUS, CT_TGEN, CT_TBRCH, CT_TAREABUS, ...
%     CT_TAREAGEN, CT_TAREABRCH, CT_ROW, CT_COL, CT_CHGTYPE, CT_REP, ...
%     CT_REL, CT_ADD, CT_NEWVAL, CT_TLOAD, CT_TAREALOAD, CT_LOAD_ALL_PQ, ...
%     CT_LOAD_FIX_PQ, CT_LOAD_DIS_PQ, CT_LOAD_ALL_P, CT_LOAD_FIX_P, ...
%     CT_LOAD_DIS_P, CT_TGENCOST, CT_TAREAGENCOST, CT_MODCOST_F, ...
%     CT_MODCOST_X] = idx_ct;
% loadprofile = struct( ...
%     'type', 'mpcData', ...
%     'table', CT_TBUS, ...
%     'rows', 1:9, ...
%    'col', 3, ...
%    'chgtype', CT_REP, ...
%    'values', [] );
% % loadprofile.values(:, 1, 1) = squeeze(Dbt(1,:,:))';
```

UC Profile Variations Data:

```
function UCprofile = uc_profile_9B3G9L
% %EX_LOAD_PROFILE  Example load profile data file for stochastic unit commitment.
%
% %   MOST
% %   Copyright (c) 2015-2016, Power Systems Engineering Research Center (PSERC)
% %   by Ray Zimmerman, PSERC Cornell
% %
% %   This file is part of MOST.
% %   Covered by the 3-clause BSD License (see LICENSE file for details).
% %   See https://github.com/MATPOWER/most for more info.
%
% %% define constants
% [CT_LABEL, CT_PROB, CT_TABLE, CT_TBUS, CT_TGEN, CT_TBRCH, CT_TAREABUS, ...
%    CT_TAREAGEN, CT_TAREABRCH, CT_ROW, CT_COL, CT_CHGTYPE, CT_REP, ...
%    CT_REL, CT_ADD, CT_NEWVAL, CT_TLOAD, CT_TAREALOAD, CT_LOAD_ALL_PQ, ...
%    CT_LOAD_FIX_PQ, CT_LOAD_DIS_PQ, CT_LOAD_ALL_P, CT_LOAD_FIX_P, ...
%    CT_LOAD_DIS_P, CT_TGENCOST, CT_TAREAGENCOST, CT_MODCOST_F, ...
%    CT_MODCOST_X] = idx_ct;
%
% UCprofile = struct( ...
%    'type', 'xGenData', ...
%    'table', 'CommitSched', ...
%    'rows', 1:3, ...
%    'col', 2, ...
%    'chgtype', CT_REP, ...
%    'values', [] );
% % loadprofile.values(:, 1, 1) = squeeze(Dbt(1,:,:))';
```

APPENDIX B: CODE FOR DATA GENERATION

4-Bus 3-Machine System (MATLAB Code)

```
% % with reserve constraints
% % added 1 more line L13 and Gen 2
% clear
% clc
%
% %% data generation
% rng('default');
% n_scenarios = 1200;
% basekV = 138;
% baseMVA = 100;
%
% % Load_max = [2 2];
% % rng(123); Load_scenarios = round(normrnd(1.50,0.25,[n_scenarios,2]),2);
% rng(123); Load_scenarios = [round(normrnd(300,75,[n_scenarios,1])) round(normrnd(200,55,[n_scenarios,1]))]/baseMVA;
%
% rng(34); a_scenarios = round(normrnd(150,20,[n_scenarios,3]));
% rng(56); b_scenarios = round(normrnd(10*baseMVA,2*baseMVA,[n_scenarios,3]));
% rng(78); c_scenarios = round(normrnd(0.005*baseMVA*baseMVA,0.001*baseMVA*baseMVA,[n_scenarios,3]));
% % rng(56); b_scenarios = [round(normrnd(11*baseMVA,2*baseMVA,[n_scenarios,1]))
round(normrnd(12*baseMVA,3*baseMVA,[n_scenarios,1])) round(normrnd(13*baseMVA,4*baseMVA,[n_scenarios,1]))];
% % rng(34); a_scenarios = round(rand(n_scenarios,2)*250);
%
```

```
% data = [Load_scenarios a_scenarios b_scenarios c_scenarios];
% data = array2table(data);
% data.Properties.VariableNames = {'L2','L4','a1','a2','a3','b1','b2','b3','c1','c2','c3'};
% writetable(data,'UC_4b3g_1200Labc.csv');
%
% %% with TL limits
% clc
% clear
%
% data_Labc = table2array(readtable('UC_4b3g_1200Labc.csv'));
% n_scenarios = size(data_Labc,1);
%
% % sys info ----------------------------------------------------------------
% sys = d_4B3G5L;
% nbus = length(sys.bus(:,1));
% basekV = 138;  % kV
% baseMVA = 100; % MVA
% Pmin = sys.gen(:,10)/baseMVA; % pu
% Pmax = sys.gen(:,9)/baseMVA; % pu
% TLcap = sys.branch(:,6)/baseMVA; % pu
% n_gen = length(Pmin);
% [Bp,~] = Bp_Bdp(sys); % pu
% ref_bus = find(sys.bus(:,2)==3); % matpower gives ref bus type-number '3'
%
% n_vars = 10; % x' = [I1 I2 I3 P1 P2 P3 d1 d2 d3 d4]; 1x10
% lb = [zeros(1,3) Pmin' -2*pi*ones(1,4)]';
% ub = [ones(1,3) Pmax' 2*pi*ones(1,4)]';
% I_ind = 1:n_gen;
% P_ind = (n_gen+1):(2*n_gen);
% d_ind = (2*n_gen+1):(2*n_gen+nbus);
% % r_ind = (2*n_gen+nbus+1):(2*n_gen+nbus+n_gen)
% gen_ind = find(sys.bus(:,2)~=1);
%
% % CPLEX MIQP -------------------------------------------------------------
% % Aeq.x=beq
% ref_eq = 1;
% nodal_eq = 2:(nbus+1);
% Aeq = zeros(nbus+1,n_vars);
% Aeq(ref_eq,d_ind(ref_bus))= 1; % ref bus angle
% Aeq(nodal_eq,d_ind) = Bp; % nodal balance eqs
% Aeq(nodal_eq(gen_ind),P_ind) = eye(n_gen)*-1;
%
% x = sys.branch(:,4).*(sys.branch(:,6)/baseMVA);
% nTL = length(sys.branch(:,1));
% TLf = sys.branch(:,1);
% TLt = sys.branch(:,2);
%
% A = zeros(nTL*2+n_gen*2+1,n_vars);
% %TL limits
% for i = 1:nTL
%     A(i,d_ind(TLf(i))) = 1; % di
%     A(i,d_ind(TLt(i))) = -1; %-dj
%
%     A(i+nTL,d_ind(TLf(i))) = -1; %dj
%     A(i+nTL,d_ind(TLt(i))) = 1; %-di
% end
% %gen limits
% for i = 1:n_gen
%     A(2*nTL+i,I_ind(i)) = Pmin(i); %gen lower
%     A(2*nTL+i,P_ind(i)) = -1;      %limits
%
%     A(2*nTL+n_gen+i,P_ind(i)) = 1; %gen upper
%     A(2*nTL+n_gen+i,I_ind(i)) = -Pmax(i); %limits
% end
%
% % spinning reserve req (R = 30% of Load_total)
% A(2*nTL+2*n_gen+1,I_ind) = -1*[Pmax-Pmin]';
% A(2*nTL+2*n_gen+1,P_ind) = 1;
%
% data = zeros(n_scenarios,28);
% no_sol = [];
% L_cols = 1:2;
% a_cols = 3:5;
% b_cols = 6:8;
% c_cols = 9:11;
% RP = 0.3; % 30% spinning reserve requirement
% SimTime = [];
% for i=1:n_scenarios
%
%     L = zeros(1,nbus); L([2,4]) = data_Labc(i,L_cols);
%     a_gen = data_Labc(i,a_cols);
%     b_gen = data_Labc(i,b_cols);
%     c_gen = data_Labc(i,c_cols);
%
%     %Aeq.x=beq
%     beq = [0 -L]';
%     %A.x<=b
%     b=[x' x' zeros(1,2*n_gen) -RP*sum(L)]';
%     % x0 = zeros(15,1);
%     f = [a_gen b_gen zeros(1,4)];
%     H = zeros(10);
%     H(P_ind,P_ind) = diag(2*c_gen);
%
%     [v,fv,ef,time]=cplexmiqp(H,f,A,b,Aeq,beq,[],[],[],lb,ub,'IIICCCCCCC');
%
%     if (ef==1)
%        SimTime = [SimTime;time.time];
%        l = L*baseMVA;
%        a = a_gen;
%        b = b_gen/baseMVA;
%        c = c_gen/(baseMVA*baseMVA);
%        I = round(v(I_ind))';
%        P = v(P_ind)'*baseMVA;
%        d = v(d_ind)';
%        Lt = sum(l);
%        Gt = sum(P);
%        Rt = sum((Pmax'*baseMVA-P).*I);
%        cost = fv;
%        data(i,:) = [l a b c I P d Lt Gt Rt cost time.time];
%     else
%        no_sol = [no_sol i];
%     end
% end
% mean(SimTime)*1000 % 55.26s for 1000 instances
%
% data1 = data;
% data1(no_sol,:) = [];
% data1 = data1(1:1000,:);
%
% %% preparing for export
% data = data1;
% del = data(:,20:23); % del cols in data
% X = sys.branch(:,4);
% LineFlows = [(del(:,1)-del(:,2))/X(1) (del(:,1)-del(:,3))/X(2) (del(:,1)-del(:,4))/X(3) (del(:,2)-del(:,3))/X(4) (del(:,3)-del(:,4))/X(5)]*baseMVA;
%
% fdata = [data(:,1:(end-5)) LineFlows data(:,(end-4):end)];
%
% % exporting
% T = array2table(fdata);
%
% x1=strcat({strseq('P',TLf)},{strseq('_',TLt)});
% x2=strcat(x1{1,1}(:,1),x1{1,1}(:,2));
```

```
%
% col_names = {strseq('L',1:nbus);strseq('a',1:n_gen);strseq('b',1:n_gen);strseq('c',1:n_gen);strseq('I',1:n_gen);strseq('P',1:n_gen);...
%     strseq('d',1:nbus);x2;'Lt';'Gt';'Rt';'cost';'time'};
%
% T.Properties.VariableNames = cat(1,col_names{:,1})';
%
% writetable(T,'UC_4b3g_1000.csv');
```

39-Bus 10-Machine System (MATLAB Code)

```
% % Matpower "case39"
%
% clear
% clc
%
% %% data generation
% rng('default');
% n_scenarios = 1500;
% basekV = 345;
% baseMVA = 100;
% sys = d_39B10G46L;
% sys.branch(:,[3,5]) = 0; % neglecting shunt flows
% nbus = length(sys.bus(:,1));
% TLcap = sys.branch(:,6);
% Pmin = sys.gen(:,10);
% Pmax = sys.gen(:,9);
% n_gen = length(Pmin);
% ng_hydro = 1;
% idx_hydro = 1;
% ng_intrcnct = 1;
% idx_intrcnct = 10;
% ng_fossil = 3;
% idx_fossil = [4,5,7];
% ng_nuke = 5;
% idx_nuke = [2,3,6,8,9];
%
% % Load Scenarios
% load_means = 0.5*sys.bus(:,3);
% Load_scenarios = zeros(nbus,n_scenarios);
% rng(123);
% for i =1:nbus
%     Load_scenarios(i,:) = round(normrnd(load_means(i),.1*load_means(i),[1,n_scenarios]),2);
% end
% Load_scenarios = Load_scenarios';
%
% % make sure no -ve prices are generated -----------------------------------
% rng(34); a_scenarios = round([normrnd(1000,100,[n_scenarios,ng_hydro]) normrnd(500,50,[n_scenarios,ng_intrcnct])
normrnd(600,60,[n_scenarios,ng_fossil]) normrnd(800,80,[n_scenarios,ng_nuke])]);
% a_scenarios(:,[idx_hydro,idx_intrcnct,idx_fossil,idx_nuke]) = a_scenarios;
% rng(56); b_scenarios = round([normrnd(10*baseMVA,2*baseMVA,[n_scenarios,ng_hydro])
normrnd(11.5*baseMVA,2*baseMVA,[n_scenarios,ng_intrcnct]) normrnd(12*baseMVA,2*baseMVA,[n_scenarios,ng_fossil])
normrnd(12*baseMVA,2*baseMVA,[n_scenarios,ng_nuke])]);
% b_scenarios(:,[idx_hydro,idx_intrcnct,idx_fossil,idx_nuke]) = b_scenarios;
% rng(78); c_scenarios = round([normrnd(0.008*baseMVA*baseMVA,0.001*baseMVA*baseMVA,[n_scenarios,ng_hydro])
normrnd(0.008*baseMVA*baseMVA,0.001*baseMVA*baseMVA,[n_scenarios,ng_intrcnct])
normrnd(0.004*baseMVA*baseMVA,0.001*baseMVA*baseMVA,[n_scenarios,ng_fossil])
normrnd(0.0035*baseMVA*baseMVA,0.0005*baseMVA*baseMVA,[n_scenarios,ng_nuke])]);
% c_scenarios(:,[idx_hydro,idx_intrcnct,idx_fossil,idx_nuke]) = c_scenarios;
%
% data = [Load_scenarios a_scenarios b_scenarios c_scenarios];
% data = array2table(data);
% col_names = {strseq('L',1:nbus);strseq('a',1:n_gen);strseq('b',1:n_gen);strseq('c',1:n_gen)};
% data.Properties.VariableNames = cat(1,col_names{:,1})';
% writetable(data,'UC_39b10g_1200Labc.csv');
%
% %% with TL limits
% clear
%
% fdata1 = table2array(readtable('UC_39b10g_1200Labc.csv'));
% n_scenarios = size(fdata1,1);
%
% % sys info ---------------------------------------------------------------
% sys = d_39B10G46L;
% sys.branch(:,[3,5]) = 0; % neglecting shunt flows
% nbus = length(sys.bus(:,1));
% basekV = 345;  % kV
% baseMVA = 100; % MVA
% Pmin = sys.gen(:,10)/baseMVA; % pu
% Pmax = sys.gen(:,9)/baseMVA; % pu
% TLcap = sys.branch(:,6)/baseMVA; % pu
% n_gen = length(Pmin);
% [Bp,~] = Bp_Bdp(sys); % pu
% ref_bus = find(sys.bus(:,2)==3); % matpower gives ref bus type-number '3'
%
% n_vars = 59; % x' = [I1...I10 P1...P10 d1...d39]; 1x59
% lb = [zeros(1,n_gen) Pmin' -2*pi*ones(1,nbus)]';
% ub = [ones(1,n_gen) Pmax' 2*pi*ones(1,nbus)]';
% I_ind = 1:n_gen;
% P_ind = (n_gen+1):(2*n_gen);
% d_ind = (2*n_gen+1):(2*n_gen+nbus);
% % r_ind = (2*n_gen+nbus+1):(2*n_gen+nbus+n_gen)
% gen_bus = find(sys.bus(:,2)~=1);
%
% % CPLEX MIQP -------------------------------------------------------------
% % Aeq.x=beq
% ref_eq = 1;
% nodal_eq = 2:(nbus+1);
% Aeq = zeros(nbus+1,n_vars);
% Aeq(ref_eq,d_ind(ref_bus))= 1; % ref bus angle
% Aeq(nodal_eq,d_ind) = Bp; % nodal balance eqs
% Aeq(nodal_eq(gen_bus),P_ind) = eye(n_gen)*-1;
%
% x = sys.branch(:,4).*(sys.branch(:,6)/baseMVA);
% nTL = length(sys.branch(:,1));
% TLf = sys.branch(:,1);
% TLt = sys.branch(:,2);
%
% A = zeros(nTL*2+n_gen*2+1,n_vars);
% %TL limits
% for i = 1:nTL
%     A(i,d_ind(TLf(i))) = 1; % di
%     A(i,d_ind(TLt(i))) = -1; %-dj
%
%     A(i+nTL,d_ind(TLf(i))) = -1; %dj
%     A(i+nTL,d_ind(TLt(i))) = 1; %-di
% end
% %gen limits
% for i = 1:n_gen
%     A(2*nTL+i,I_ind(i)) = Pmin(i); %gen lower
%     A(2*nTL+i,P_ind(i)) = -1;      %limits
%
%     A(2*nTL+n_gen+i,P_ind(i)) = 1; %gen upper
%     A(2*nTL+n_gen+i,I_ind(i)) = -Pmax(i); %limits
% end
%
% % % spinning reserve req
% A(2*nTL+2*n_gen+1,I_ind) = -1*(Pmax-Pmin)';
% A(2*nTL+2*n_gen+1,P_ind) = 1;
```

```
%
% data = zeros(n_scenarios,nbus*2+n_gen*5+4+1); %[L abc I P d Lt Gt Rt cost time]
% no_sol = [];
% L_cols = 1:39;
% a_cols = 40:49;
% b_cols = 50:59;
% c_cols = 60:69;
% RP = 0.1; % 10% spinning reserve requirement------------------------------
% SimTime = [];
% st = tic;
% for i=1:n_scenarios
%
%     L = fdata1(i,L_cols)/baseMVA;
%     a_gen = fdata1(i,a_cols);
%     b_gen = fdata1(i,b_cols);
%     c_gen = fdata1(i,c_cols);
%
%     %Aeq.x=beq
%     beq = [0 -L]';
%     %A.x<=b
%     b=[x' x' zeros(1,2*n_gen) -RP*sum(L)]';
%     % x0 = zeros(15,1);
%     f = [a_gen b_gen zeros(1,n_vars-2*n_gen)];
%     H = zeros(n_vars);
%     H(P_ind,P_ind) = diag(2*c_gen);
%
%     [v,fv,ef,time] = cplexmiqp(H,f,A,b,Aeq,beq,[],[],[],lb,ub,'IIIIIIIIIICCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC');
%
%     if (ef==1)
%         SimTime = [SimTime;time.time];
%         l = L*baseMVA;
%         a = a_gen;
%         b = b_gen/baseMVA;
%         c = c_gen/(baseMVA*baseMVA);
%         I = round(v(I_ind))';
%         P = v(P_ind)'*baseMVA;
%         d = v(d_ind)';
%         Lt = sum(l);
%         Gt = sum(P);
%         Rt = sum((Pmax'*baseMVA-P).*I);
%         cost = fv;
%         data(i,:) = [l a b c I P d Lt Gt Rt cost time.time];
%     else
%         no_sol = [no_sol i];
%     end
% end
% et = toc(st)
% mean(SimTime)*1000 % 114 seconds
%
% data1 = data;
% data1(no_sol,:) = [];
% data1 = data1(1:1000,:);
%
% %% preparing for export
% data = data1;
% del = data(:,(nbus+5*n_gen+1):(2*nbus+5*n_gen)); % del cols in data
% LineFlows = zeros(size(del));
% for i = 1:nTL
%     LineFlows(:,i) = [(del(:,TLf(i))-del(:,TLt(i)))/sys.branch(i,4)]*baseMVA;
% end
%
% fdata = [data(:,1:(end-5)) LineFlows data(:,(end-4):end)];
%
% % exporting
% T = array2table(fdata);
%
```

```
% x1=strcat({strseq('P',TLf)},{strseq('_',TLt)});
% x2=strcat(x1{1,1}(:,1),x1{1,1}(:,2));
%
% col_names = {strseq('L',1:nbus);strseq('a',1:n_gen);strseq('b',1:n_gen);strseq('c',1:n_gen);strseq('I',1:n_gen);strseq('P',1:n_gen);...
%     strseq('d',1:nbus);x2;'Lt';'Gt';'Rt';'cost';'time'};
%
% T.Properties.VariableNames = cat(1,col_names{:,1})';
%
% writetable(T,'UC_39b10g_1000new.csv');
```

REFERENCES

[1] K. Cleary, K. Palmer, “US Electricity Markets 101”, an explainer published by Recourses for the Future (RFF), March 2022. [Online] https://www.rff.org/publications/explainers/us-electricity-markets-101/.

[2] Electric Power Markets, Federal Energy Regulatory Commission (FERC). [Online] https://www.ferc.gov/electric-power-markets.

[3] Report on “Electric Power Annual 2020”, Released by U.S. Energy Information Administration (EIA), October 2021. [Online] https://www.eia.gov/electricity/annual/.

[4] Y. Chen, A. Casto, F. Wang, Q. Wang, X. Wang and J. Wan, "Improving Large Scale Day-Ahead Security Constrained Unit Commitment Performance," in IEEE Transactions on Power Systems, vol. 31, no. 6, pp. 4732-4743, Nov. 2016, doi: 10.1109/TPWRS.2016.2530811.

[5] A. S. Xavier, F. Qiu, S. Ahmed, "Learning to Solve Large-Scale Security-Constrained Unit Commitment Problems", INFORMS Journal on Computing, vol. 33, no. 2, pp. 739-756, Oct. 2020, doi: 10.1287/ijoc.2020.0976.

[6] “Getting Started with CPLEX for MATLAB,” Provided by IBM ILOG CPLEX Optimizers. [Online]. Available: https://www.ibm.com/docs/en/icos/12.7.1.0?topic=matlab-getting-started-cplex.

[7] K. Hara, M. Kimura, and N. Honda, “A method for planning economic unit commitment and maintenance of thermal power systems,” IEEE Trans. Power App. Syst., vol. PAS-85, pp. 427–436, May 1966.

[8] R. H. Kerr, J. L. Scheidt, A. J. Fontana, and J. K. Wiley, “Unit commitment,” IEEE Trans. Power App. Syst., vol. PAS-85, pp. 417–421, May 1966.

[9] A. J. Wood, B. F. Wollenberg, G. B. Sheble, Power Generation Operation and Control, 3rd ed. New Jersey: Wiley, 2014.

[10] F. N. Lee, “Short-term unit commitment—a new method,” IEEE Trans. Power Syst., vol. 3, pp. 421–428, May 1988.

[11] F. N. Lee, “A fuel-constrained unit commitment method,” IEEE Trans. Power Syst., vol. 4, pp. 1208–1218, 1989.

[12] F. N. Lee and Q. Chen, “Unit commitment risk with sequential rescheduling,” IEEE Trans. Power Syst., vol. 6, pp. 1017–1023, Aug. 1991.

[13] F. N. Lee, “The application of commitment utilization factor (CUF) to thermal unit commitment,” IEEE Trans. Power Syst., vol. 6, pp. 691–698, May 1991.

[14] F. N. Lee and Q. Feng, “Multi-area unit commitment,” IEEE Trans. Power Syst., vol. 7, pp. 591–599, May 1992.

[15] R. M. Burns and C. A. Gibson, “Optimization of priority lists for a unit commitment program,” in Proc. IEEE Power Eng. Soc. Summer Meeting, 1975.

[16] R. R. Shoults, S. K. Chang, S. Helmick, and W. M. Grady, “A practical approach to unit commitment, economic dispatch and savings allocation for multiple-area pool operation with import/export constraints,” IEEE Trans. Power App. Syst., vol. PAS-99, pp. 625–635, Mar./Apr. 1980.

[17] J. Zhu, “Unit Commitment,” in *Optimization of Power System Operation*, 2nd ed. Hoboken, NJ, USA: Wiley, 2015, ch. 7, sec. 7.2, pp. 253–255.

[18] N. P. Padhy, "Unit commitment-a bibliographical survey," in IEEE Transactions on Power Systems, vol. 19, no. 2, pp. 1196-1205, May 2004.

[19] A. M. Elsayed, A. M. Maklad and S. M. Farrag, "A new priority list unit commitment method for large-scale power systems," 2017 Nineteenth International Middle East Power Systems Conference (MEPCON), 2017, pp. 359-367.

[20] P. G. Lowery, “Generating unit commitment by dynamic programming,” IEEE Trans. Power App. Syst., vol. PAS-85, pp. 422–426, May 1966.

[21] S. Sen and D. P. Kothari, “Evaluation of benefit of inter-area energy exchange of the Indian power systems based on multi-area unit commitment approach,” Elect. Mach. Power Syst., vol. 26, pp. 801–813, 1998.

[22] P. K. Singhal and R. N. Sharma, "Dynamic Programming Approach for Large Scale Unit Commitment Problem," 2011 International Conference on Communication Systems and Network Technologies, 2011, pp. 714-717.

[23] Merlin and P. Sandrin, “A new method for unit commitment at Electricite De France,” IEEE Trans. Power App. Syst., vol. PAS-102, pp. 1218–1225, Aug. 1983.

[24] F. Zhuang and F. D. Galiana, “Toward a more rigorous and practical unit commitment by Lagrangian Relaxation,” IEEE Trans. Power Syst., vol. 3, pp. 763–773, May 1988.

[25] S. Takriti and J. R. Birge, “Using integer programming to refine Lagrangian-based unit commitment solutions,” IEEE Trans. Power Syst., vol. 15, pp. 151–156, Feb. 2000.

[26] H. Ma and S. M. Shahidehpour, “Unit commitment with transmission security and voltage constraints,” IEEE Trans. Power Syst., vol. 14, pp. 757–764, May 1999.

[27] C. P. Cheng, C. W. Liu, and C. C. Liu, “Unit commitment by Lagrangian Relaxation and genetic algorithm,” IEEE Trans. Power Syst., vol. 15, pp. 707–714, May 2000.

[28] N. Jimenez and A. Conejo, “Short-term hydro-thermal coordination by Lagrangian Relaxation: Solution to the dual problem,” in Proc. IEEE Power Eng Vol. 14, No. 1, 1999.

[29] S. J. Wang, S. M. Shahidehpour, D. S. Kirschen, S. Mokhtari and G. D. Irisarri, "Short-term generation scheduling with transmission and environmental constraints using an augmented Lagrangian relaxation," in IEEE Transactions on Power Systems, vol. 10, no. 3, pp. 1294-1301, Aug. 1995.

[30] A. I. Cohen and M. Yoshimura, “A branch- and -bound algorithm for unit commitment,” IEEE Trans. Power App. Syst., vol. PAS-102, pp. 444–451, Feb. 1983.

[31] A. Ohuch and I. Kaji, “A branch-and-bound algorithm for startup and shutdown problem of thermal generating units,” Inst. Elect. Eng. Japan, vol. 95-B, no. 10, pp. 461–468, 1975.

[32] K.-Y. Huang, H.-T. Yang, and C.-L. Huang, “A new thermal unit commitment approach using constraint logic programming,” IEEE Trans. Power Syst., vol. 13, pp. 936–945, Aug. 1998.

[33] Dillon, T.S. and Egan, G.T., "The Application of Combinatorial Methods to the Problems of Maintenance Scheduling and Unit Commitment in Large Power Systems, IFAC Symposium on Large Scale Systems, Udine, Italy, June 1976.

[34] Dillon, T.S., Edwin, K.W., Kochs, H.-D., and Taud, R. J. , "Integer Programming Approach to the Problem of Optimal Unit Commitment with Probabilistic Reserve Determination," IEEE Transactions on PAS-97, No. 6, pp. 2154-2166, Nov/Dec 1978.

[35] G. B. Sheble and G. N. Fahd, “Unit commitment literature synopsis,” IEEE Trans. Power Syst., vol. 9, pp. 128–135, Feb. 1994.

[36] G. B. Sheble et al., “Unit commitment by genetic algorithm with penalty methods and a comparison of Lagrangian search and genetic algorithm economic dispatch example,” Int. J. Elect. Power Energy Syst., vol. 18, no. 6, pp. 339–346, Feb. 1996.

[37] D. Dasgupta and D. R. McGregor, “Thermal unit commitment using genetic algorithms,” Proc. Inst. Elect. Eng., Gen. Trans. Dist., vol. 141, pp. 459–465, Sept. 1994.

[38] T. T. Maifeld and G. B. Sheble, “Genetic-based unit commitment algorithm,” IEEE Trans. Power Syst., vol. 11, pp. 1359–1370, Aug. 1996.

[39] K. S. Swarup and S. Yamashiro, “Unit commitment solution methodology using genetic algorithm,” IEEE Trans. Power Syst., vol. 17, pp. 87–91, Feb. 2002.

[40] M. P. Walsh and M. J. O'Malley, "Augmented Hopfield network for unit commitment and economic dispatch," in IEEE Transactions on Power Systems, vol. 12, no. 4, pp. 1765-1774, Nov. 1997.

[41] R. H. Liang, F. C. Kang, "Thermal generating unit commitment using an extended mean field annealing neural network", IEE Proceedings Generation, Transmission and Distribution, vol. 147, no. 3, pp. 164-170, May 2000.

[42] Y. Bengio, A. Lodi, A. Prouvost, "Machine learning for combinatorial optimization: a methodological tour d’horizon", 2018.

[43] A. S. Xavier, F. Qiu, S. Ahmed, "Learning to Solve Large Scale Security Constrained Unit Commitment", INFORMS Journal on Computing, vol. 33, no. 2, pp. 739-756, Oct. 2020.

[44] Geisser S, “Predictive Inference: An Introduction”. Chapman and Hall, 1993.

[45] S. Issenberg, “How President Obama’s campaign used big data to rally individual voters,” MIT Technology Review, December 19, 2012.

[46] M. Kuhn, K. Johnson, *Applied Predictive Modeling*, New York: Springer, 2013.

[47] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, A. H. Byers, “Big data: The next frontier for innovation, competition, and productivity,” McKinsey Global Institute, 2011.

[48] D. T. Larose, C. D. Larose, *Discovering Knowledge in Data: An Introduction to Data Mining*, 2nd ed. New Jersey: Wiley, 2014.

[49] R. Duda, P. Hart, D. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.

[50] J. Han, M. Kamber, J. Pei, *Data Mining Concepts and Techniques*, 3rd ed. Massachusetts: MK Elsevier, 2012.

[51] P. N. Tan, M. Steinbach, V. Kumar, *Introduction to Data Mining*, 2nd ed. Pearson Education India, 2016.

[52] T. Martin, P. Harten, D. Young, E. Muratov, A. Golbraikh, H. Zhu, A. Tropsha, "Does Rational Selection of Training and Test Sets Improve the Outcome of QSAR Modeling?", Journal of Chemical Information and Modeling, vol. 52, issue 10, pg. 2570–2578, October 2012.

[53] MathWorks, "Machine Learning with MATLAB", [online] https://www.mathworks.com/content/dam/mathworks/ebook/gated/machine-learning-ebook-all-chapters.pdf.

[54] A. Tharwat, "Classification assessment methods", Applied Computing and Informatics, vol. 17, Issue 1, pg. 168-192, August 2018.

[55] https://en.wikipedia.org/wiki/Template:Diagnostic_testing_diagram

[56] A. S. Xavier, F. Qiu, S. Ahmed, "Learning to Solve Large-Scale Security-Constrained Unit Commitment Problems", INFORMS Journal on Computing, vol. 33, no. 2, pp. 739-756, Oct. 2020.

[57] S. Pineda, J. M. Morales, and A. Jiménez-Cordero, "Data-Driven Screening of Network Constraints for Unit Commitment," in IEEE Transactions on Power Systems, vol. 35, no. 5, pp. 3695-3705, Sept. 2020.

[58] J. Zhu, Optimization of Power System Operation, 2nd ed. New Jersey: Wiley, 2015.

[59] T. Iqbal and A. Feliachi, "Decentralized Voltage Control Using Fast Community Detection Algorithm and Eigen Decomposition," 2019 IEEE PES-GM, pp. 1-5, 2019.

[60] H. U. Banna, T. Iqbal, A. Khan, and Z. Zahra, "Generator's coherency identification using relative correlation-based clustering," 2018 International Conference on Engineering and Emerging Technologies (ICEET), 2018, pp. 1-5, 2018.

[61] T. Iqbal, H. U. Banna and M. H. Riaz, "Cyber intrusion detection through spatio-temporal correlation in optimal power flow problem," 2018 International Conference on Engineering and Emerging Technologies (ICEET), 2018, pp. 1-5, 2018.

[62] T. Iqbal and A. Feliachi, "Discovering community structures in power system networks using voltage — Reactive power sensitivity," 2017 North American Power Symposium (NAPS), 2017, pp. 1-6, 2017.

[63] T. Iqbal, A. D. Banadaki and A. Feliachi, "Optimal voltage control using singular value decomposition of fast decoupled load flow Jacobian," 49th North American Power Symposium (NAPS), pp. 1-6, 2017.