

Rochester Institute of Technology

RIT Scholar Works

Theses

5-2021

Predicting the Required Demand Flow to A Customer

Thani AlMansoori
toa6305@rit.edu

Follow this and additional works at: <https://scholarworks.rit.edu/theses>

Recommended Citation

AlMansoori, Thani, "Predicting the Required Demand Flow to A Customer" (2021). Thesis. Rochester Institute of Technology. Accessed from

This Master's Project is brought to you for free and open access by RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.

Predicting the Required Demand Flow to A Customer

by

Thani AlMansoori

**A Capstone Submitted in Partial Fulfilment of the Requirements for
the Degree of Master of Science in Professional Studies:**

Data Analytics

Department of Graduate Programs & Research

Rochester Institute of Technology

RIT- Dubai

May 2021

RIT

Master of Science in Professional Studies: Data Analytics

Graduate Capstone Approval

Student Name: Thani AIMansoori

Student UID: 798005370

Graduate Capstone Title: Predicting the Required Demand Flow to A Customer

Graduate Capstone Committee:

Name: Dr. Sanjay Modak
Chair of committee

Date:

Name: Dr. Khalil Al Hussaeni
Member of committee/Mentor

Date:

Acknowledgements

Starting my thanks would begin with thanking Allah for giving me the chance to reach this stage, and complete my MS Degree in Data Analytics. As well as I extend my thanks to Dr. Sanjay and Dr. Khalil for their gaudiness and supervision. As well as all RIT staff for their cooperation and support, where they showed their capabilities during COVID-19 year.

Additionally, exclusive thanks to my parents for their effort and support during this journey, where many motivational thoughts and advices were given to complete these studies.

Nevertheless, I would like to emphasize my thanks to my company management for understanding the importance of high education degree and ensure that all requirements were given to me.

Finally, I am thankful to my friend AbdulAziz Zindaki for his support and for offering his help during the past 15 months during the classes.

Contents

Subscripts	5
Abstract.....	6
Introduction	7
1. Problem Statement.....	8
2. Background of the Problem	9
3. Project Definition and Goals	10
3.1 Project Definition	10
3.2 Aims and Objectives.....	10
3.3 Limitations of the Study	10
3.4 Project Budget.....	10
4. Literature Review	11
5. Methodology Used.....	14
6. Sources of Data	15
7. Analysis	16
7.1 Exploration Data Analysis	17
7.1.1 Getting the required data	17
7.1.2 Missing Data.....	19
7.2 Data Preprocessing	20
7.2.1 Install all required libraries	20
7.2.2 Data Cleaning Process.....	20
7.3 Data Mining.....	21
7.3.1 Exploratory Data Analysis	21
7.3.2 Training & Testing for original dataset	24
7.3.2.1 ARIMA Section	24
7.3.2.1.1 Augmented Dickey-Fuller Test.....	25
7.3.2.1.2 ARIMA Model: Average Flow	27
7.3.2.1.3 ARIMA Model: Average Power	29
7.3.2.1.4 ARIMA Model: Peak Power	31
7.3.2.2 KNN Section	33
7.3.2.2.1 KNN Model: Average Flow	34
7.3.2.2.2 KNN Model: Average Power	35
7.3.2.2.3 KNN Model: Peak Power.....	36
7.3.2.3 SVM Section	37
7.3.2.3.1 SVM Model: Average Flow.....	38
7.3.2.3.2 SVM Model: Average Power	39

7.3.2.3.3 SVM Model: Peak Power.....	40
7.3.3 Evaluating the models and finding the summary of trained model	41
7.3.3.1 Evaluating ARIMA model	41
7.3.3.2 Evaluating KNN model	42
7.3.3.3 Evaluating SVM model	42
7.3.4 Predicting the future values.....	43
7.3.4.1 ARIMA Predicted values.....	43
7.3.4.2 KNN Predicted values.....	46
7.3.4.3 SVM Predicted values	48
7.3.5 Predicted values evaluation	50
8. Analysis Results	51
9. Conclusions and Future Work	53
10. Bibliography	54
Appendix A	55

subscripts

Shortcut

Description

AIC	Akaike Information Criterion
ANN	Artificial Neural Network
ARIMA	Auto-Regressive Integrated Moving Average
BIC	Bayesian Information Criterion
COVID-19	Corona Virus Disease -19
CRISP DM	Cross Industry Standards Process for Data Mining
DCC	District Cooling Company
DCP	District Cooling Plant
EDA	Exploration Data Analytics
Eq	Equation
ETS	Energy Transfer Station
FCV	Flow Control Valve
IDEA	International District Energy Association
IQR	Inter Quartile Range (Boxplot)
KNN	K-Nearest Neighbors Algorithm
MAE	Mean Absolute Error
ML	Machine Learning
PHEX	Plate Heat Exchanger
Primary Side	DCC Side Related
SARIMAX	Seasonal Autoregressive Integrated Moving Averages with Exogenous Regressors
SCADA	Supervisory Control and Data Acquisition
Secondary Side	Customer Side Related
Std	Stander Deviation
SVM	Support Vector Machine Algorithm

Abstract

Yearly District Cooling Company (DCC) needs to upgrade the plants to meet customer requirements. Moreover, this project will help the company to find when the demand load of flow will reach above the plant's installed capacity and how much increase is expected. So, the dataset was provided by the company for previous years. By applying ML and Arima, KNN, and SVM algorithms, it was found that the best result was the KNN algorithm with an accuracy of 94%, and the maximum flow load will be reached in summer of upcoming years.

Hence, the company needs to take further action to upgrade the plant accordingly, either add a new chiller, modify the pumps, or any other suitable action.

Keywords: Algorithms, Data Analytics, KNN, SVM, ARIMA, Python, Tableau, Accuracy, Time Series Forecasting, CRISP-DM, Temperatures, District Cooling.

***Note:**

- *Due to confidentiality of the data/results by the company, some data will not be displayed clearly.*

Introduction

Most modern and big cities are moving towards applying the new generations of renewable energy to reach the requirements' load demand. UAE has the biggest malls, hotels, and towers, where cooling energy is vital during summertime, as the weather temperature is reaching around 50 °C. Hence, District Cooling System came to solve the high load demand with a friendly environment solution. Where chilled water will take place in District Cooling Plants (DCP) and will serve the customers through underground network pipes, reaching the Energy Transfer Station (ETS), where it is the focal point between the DCP (primary side) and Customer Side (Secondary side). DCC need to ensure that all requirements are met to the customer side as agreed in the contract in the initial stage. Some challenges might happen during the summer season, like flow shortage, which will impact the customer side with high supply temperature above the acceptable range. DCC spending much time and effort each year reviewing the plants' modification plans based on the requirements.

Hence, this project will focus on one critical plant which serves an important client (plant connected to a single client), where it will predict the average flow will occur and when it will occur, so DCC Management can decide to take the right action before that time, such as add additional equipment, modify the existing equipment based on previous data.

The plan for this project is to utilize data analysis technics and do data exploring using Tableau software for visualization. Then develop a Models using ARIMA Model via Python programming language and apply some algorithms for testing and training purposes like K-Nearest Neighbors (KNN) and Support Vector Machine (SVM) for better output, which will help to predict the required load demand of flow and see if other parameters can be predicted as well. Furthermore, the results will be shared with DCC management for better decision making.

1. Problem Statement

District Cooling services peak demand load is reaching and maximum during summer seasons, wherein UAE climate is reaching close to 50 °C in August/September each year. Each plant has specific equipment with limited capacity to deliver, and customers' load is growing gradually. It is crucial to think in advance for accurate decision-making, which can keep the company's business continuity plan and keep expanding and achieve UAE vision to be an innovative and clean country.

This Project will help to predict the average demand load of flow, power, and peak power of one of the customers, in different words, to forecast the demand load for short term. based on this prediction, and it will help DCC to decide if this plant is required to include in the next-year budget for upgrade/modifications where this will ensure that no interruption to the customer during this summer.

2. Background of the Problem

As Business Continuity Plan for DCC, it should keep growing and ensure that the services reach the customers with high satisfaction. If a customer asked DCC for district cooling services, DCC will review the plant capacity before connecting and approve the contract to make sure that the plant can deliver the required load at the peak.

Each plant has its specific installed capacity. During summer, the peak of cooling service reaches its maximum demand, and some customers are complaining and ask for more flow to maintain their cooling temperature in their building; some of them have their own design issue in their building. DCC serving many customers, and if this customer asked for extra flow, DCC need to ensure that the plant is capable to additional load requirement without any delay.

As it is well known that flow parameter is one of the cooling load consumption factors, the more flow, the less DT might occur, where he needs to return the flow with temperature difference around certain degree °C, due to plant equipment designed to operate with specific temperatures.

3. Project Definition and Goals

3.1 Project Definition

- This project will try to predict the flow for next years, based on historized data from previous years. Where flow, supply temperature, return temperature, temperature difference, power and energy data are available with DCC.

3.2 Aims and Objectives

- ✓ Help Management for Better Decision Making.
- ✓ Implementing new algorithms into District Cooling Sector.
- ✓ Reduce Customer Complaints.
- ✓ Achieve UAE Vision for Green Energy.
- ✓ Ensure the Company Business Continuity Plan.

3.3 Limitations of the Study

- ❖ Limited coding experience.
- ❖ A small dataset was provided.
- ❖ COVID-19 situation led to different approaches of learning.

3.4 Project Budget

- ❖ This project is not required any financial budget, where only the data is required from DCC servers.

4. Literature Review

Modern cities are famous with attractive buildings and iconic shapes, tall towers with different business types, residential, commercial, or business types. Those buildings need different energy types, either electrical energy, cooling energy, to facilitate the business.

A study was done by (HO, 2003), where he stated that: *“It has been expected that the annual mean temperature in the decade 2090-2099 will rise by 4.8 °C”*. During the summer season, the UAE climate reaching above 50 °C, under such weather, the air conditioning system is consuming high electrical energy by end-user in all buildings, reflecting more billing price and more emissions to the environment, where everyone will be affected in some stage, customers, governments and health sector as a sequence.

Using the traditional cooling system for individual users also not recommended, where electrical power plants will add more load on their generation power plant. For each building it has its own power consumption, a report was done in 2004 by the Electrical and Mechanical Service Department (Electrical and Mechanical Services Department (EMSD), 2012) in Hong Kong showed that: *“the annual electricity consumption in Hong Kong increased steadily from 134,138 TJ in 2001 to 150,895 TJ in 2010, i.e. by 12.5% over the 10-year period. In this period, the commercial sector, which is the dominant electricity consumer, had further increased its electricity consumption by 21%.”*

Also, he compared the accuracy of different building energy consumption predictions using three different ML algorithms, ANN, Support Machine Vector, and Random Forest. Where it has been studied in two types of buildings, elementary school building and commercial building, and found that *“For the elementary school buildings, the average coefficient of variance of the root mean squared error (CvRMSE) determined by ANNs was 5.4% and the average CvRMSE for commercial buildings based on the RF model was 10.9%”* (Jang & Leigh, 2017) said.

As mentioned in (Francisco, Zamora-Martínez, Romeu, & Pardo, 2013), Forecasting is the most popular technique nowadays for many business companies, which is looking for the best profits with less spending, and it more useful in terms of energy-saving and efficiency; it can help to look into the future and try to develop a proactive control system.

Auto-Regressive Integrated Moving Average Model (ARIMA model) is one of the most widely used methods for time series forecasting, where it can deliver complementary

approaches towards any forecasting problems. Where it is focused and aims to describe its autocorrelations, and the results can be considered as a reference to compare it with Artificial Neural Network (ANN) results (Escriva'-Escriva', A'lvarez-Bel, Rolda'n-Blay, & Ica'zar-Ortega, 2011) .

In general, as per (Farias, Puig, Rangel, & Flores, 2018) time series can be defined as *“a sequence of chronologically ordered observations recorded at regular time intervals. Those observations might correspond to qualitative or quantitative data.”* Where he conducted a study on how the flow demand pattern behave on hourly biases, he mentioned that: *“We assume that the chilled water flow consumption volume is recorded hourly using flowmeters. Time series present a cyclic consumption pattern, where each cycle repeats every 24 h. Observing those daily patterns, we detected different dynamic pattern behaviors that might be seen as the change of different regimes that need to be mathematically defined and validated”*. A study was made by (Quevedo & Puig, 2010), who found that water flow demand usually presents in different patterns behaviors on public holidays and weekends, and almost the same patterns on working days. Moreover, a set of algorithms related to chilled water flow demand forecasting using clustering can be found, *“Where the implementation of a daily Autoregressive Integrated Moving Average (ARIMA) model combined with hourly patterns is proposed with the objective of allowing prediction at daily and hourly scales every 24 h. The ARIMA model predicts the total day consumption while a daily pattern is selected according to a calendar for distributing the hourly consumption along the day”* (Candelieri, 2017).

Many people prefer to use ARIMA, where it easy to consider and deal with it. (Murat, Malinowska, Gos, & Krzyszczak, 2018) Showed that both (linear ARIMA and quadratic ARIMA) models had the best overall performance in making short term forecasting and predations of annual readings of the building's temperature.

Support Vector Machine (SVM) is one of the most popular supervised algorithms, where it depends on prior data to train and test; accordingly, it will understand much better for future values predictions. *“SVMs could be regarded as one generalized classifier, which was an extension of the prediction. SVM model has the ability to solve the non-linear problems even with relatively less historical samples. Accordingly, this method is not especially relying on large amount of data for training”* said by (Liu, et al., 2019).

K- Nearest Neighbor (KNN) algorithm or memory-based algorithm; its predicted values are based directly on the training examples. Where it gives a maximum likelihood estimate of the probabilities. When K is selected smaller, it will impact higher variance (less stable), and if K larger, it will reflect higher bias (less precise). The value selection of K is still not concluded globally, but most researchers agreed on K-value depends on the data: Adaptive methods (heuristics) and Cross-validation. (Cunningham & Delany, 2007) thoroughly explained the process of KNN and said: “*k-NN classification has two stages; the first is the determination of the nearest neighbors and the second is the determination of the class using those neighbors.*”

5. Methodology Used

During data analytics courses, many information and tools were explained to help data analytics users to deal with data. In this project, Python programming language will be used, where more experience was gained compared to the R-Studio programming language. The primary tool will use ARIMA to predict the required flow for a particular customer and use machine learning by applying SVM and KNN algorithms, and the training/ testing process will be applied by 70/30 of the original data set. Moreover, for visualization, the Tableau program will visualize the output values, which will be another tool used in this project.

In summary, some processes were followed from previous courses to understand the data and achieved the expected goal, which is called Cross-Industry Standards Process for Data Mining (CRISP-DM), so it can be implemented in this case also, where it is proved and provides an overall framework for planning and handling the project.

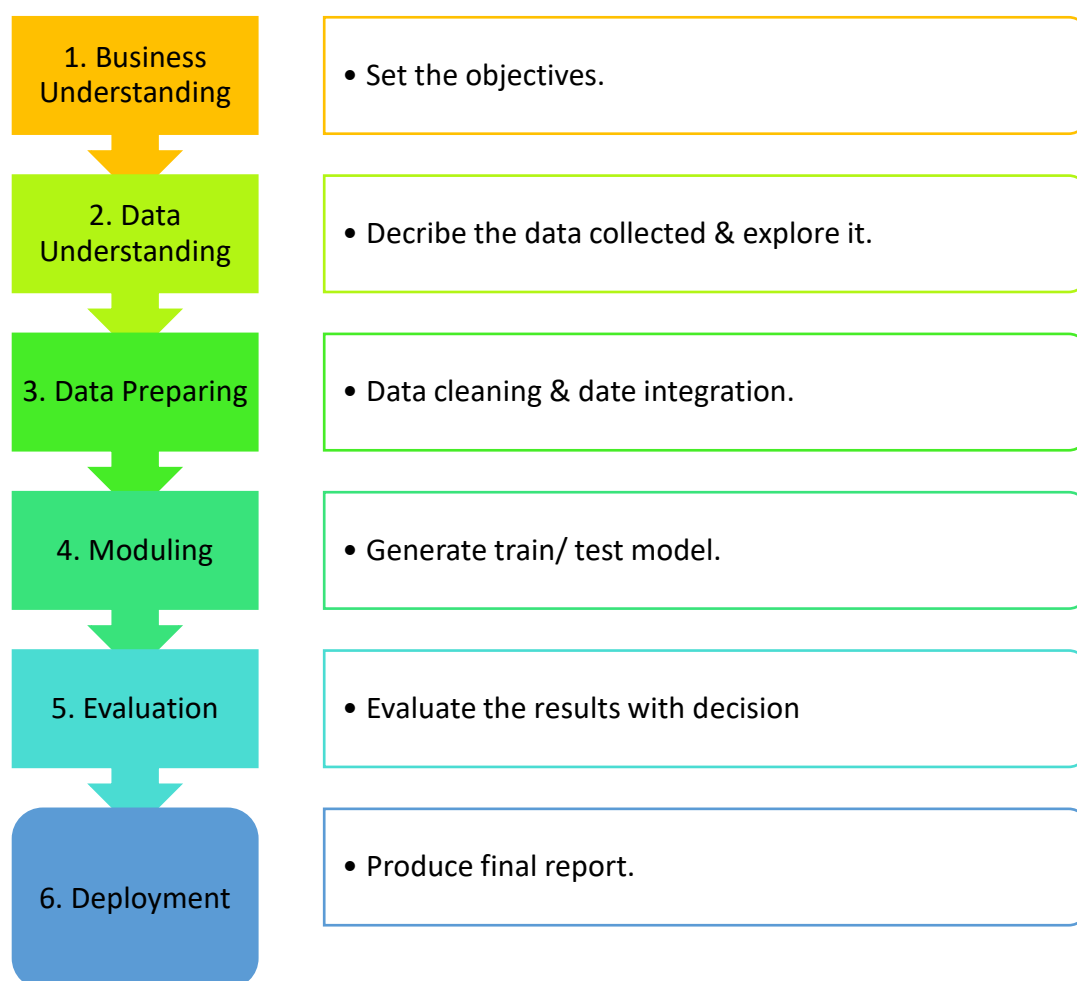


Figure 1: CRISP DM process

6. Sources of Data

For Data analytics projects, the essential thing to have the data. This project aims to solve one problem for DCC, where dataset was provided by DCC. The required data have been requested, and it was under management's approval process; within a few weeks later, data was given.

Dataset was on a daily average from previous years. It includes different parameters for that customer building, such as date (days), average supply temperature (°F), average return temperature (°F), average flow (GPM), Temperature difference (Delta T (°F)), power (TR), Peak power (TR), Energy (TRH). Furthermore, it can be considered quantitative data, which can be measured, counted, and expressed using numbers.

Checked the dataset and found that eight attributes with a total count were 8,520 readings. Moreover, it seems to be clean data and will check and confirm this statement by doing EDA. Since that dataset is qualitative, including all data for each day, it can be utilized to predict the future values under the time series forecasting group.

7. Analysis

Note:

- Python code was build using Google Colab to run it, due to required strong machine, where it is not available in my laptop.

To give brief summary of the project steps:

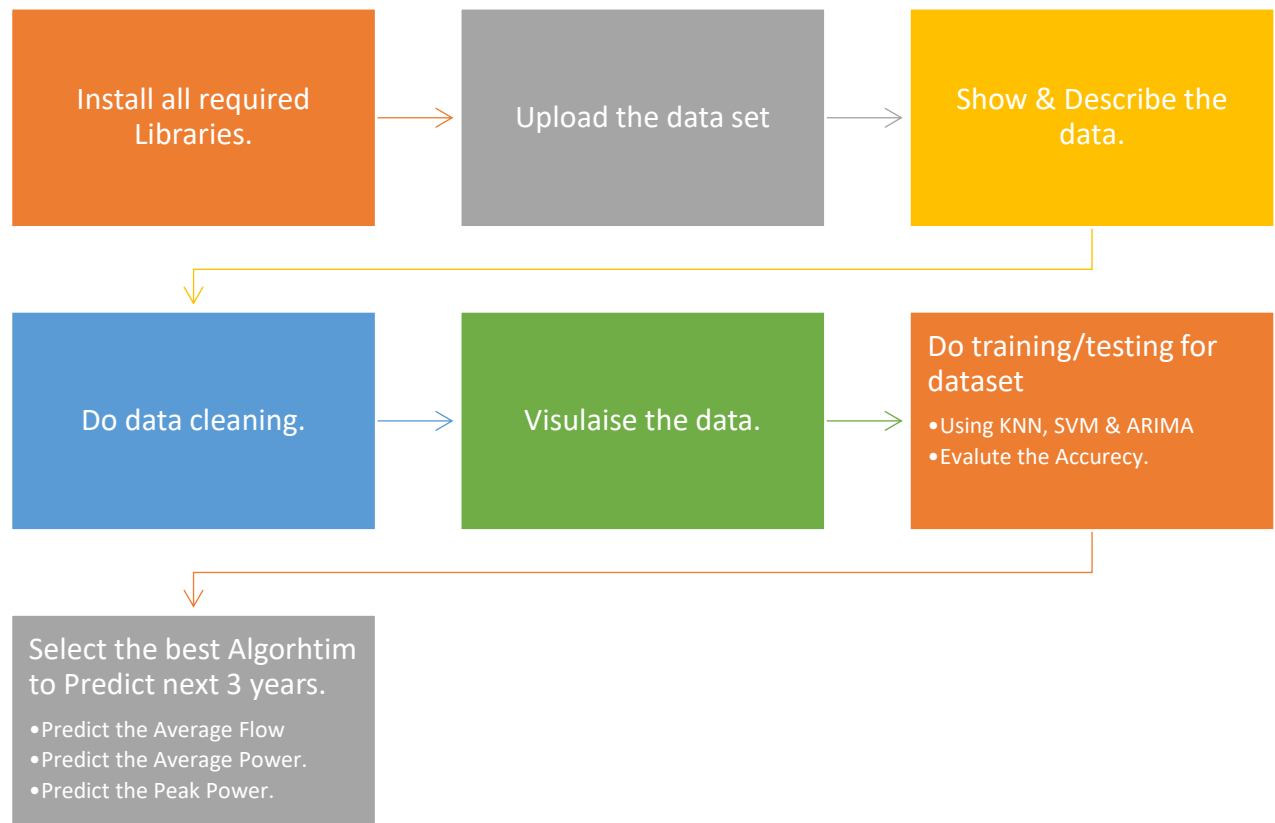
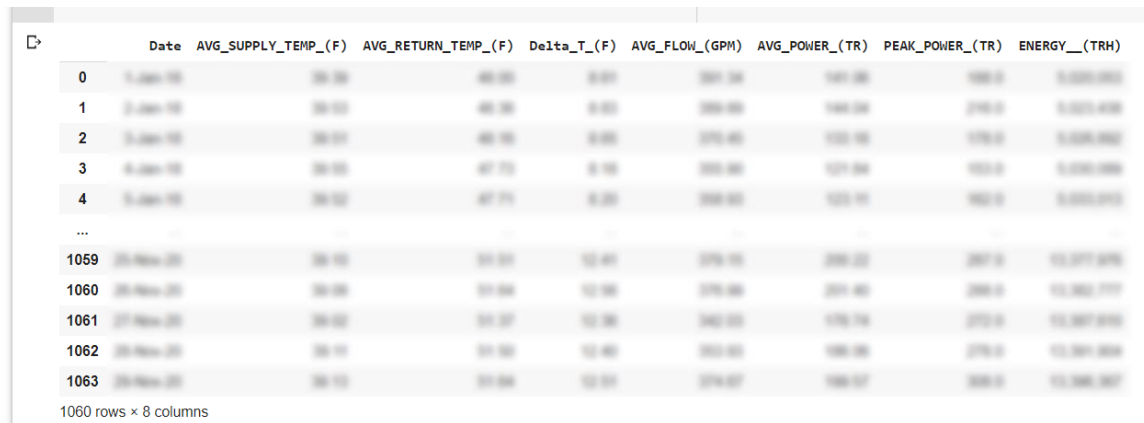


Figure 2: Project Analysis Steps used in Python

7.1 Exploration Data Analysis

7.1.1 Getting the required data

As shown below in Figure 3, the total rows available in this dataset were 1,060 with eight columns (total of 8,480 values/readings), for previous years on daily average readings. Each parameter meaning is showing in table 1 below.



	Date	AVG_SUPPLY_TEMP_(F)	AVG_RETURN_TEMP_(F)	Delta_T_(F)	AVG_FLOW_(GPM)	AVG_POWER_(TR)	PEAK_POWER_(TR)	ENERGY_(TRH)
0	1-Jan-15	59.336	48.380	9.957	2391.336	1491.385	1989.0	5.0000.0000
1	2-Jan-15	59.133	48.380	9.953	2390.450	1490.336	2190.0	5.0001.4100
2	3-Jan-15	59.133	48.380	9.953	2391.450	1491.336	1790.0	5.0000.0000
3	4-Jan-15	59.133	47.710	9.150	2391.450	1271.336	1910.0	5.0000.0000
4	5-Jan-15	59.133	47.710	9.200	2390.450	1271.336	1910.0	5.0000.0000
...
1059	25-Nov-15	59.133	51.337	12.495	2379.136	2090.336	2097.0	11.0001.0000
1060	26-Nov-15	59.133	51.337	12.495	2379.136	2091.450	2090.0	11.0001.7117
1061	27-Nov-15	59.133	51.337	12.495	2380.336	1790.336	2170.0	11.0001.0000
1062	28-Nov-15	59.133	51.337	12.495	2380.336	1990.336	2170.0	11.0001.0000
1063	29-Nov-15	59.133	51.337	12.495	2379.136	1990.336	2090.0	11.0001.0000

1060 rows x 8 columns

Figure 3: Uploading the data set and show it *.

Table 1: Data set attributes description

Attribute	Description
Date	Dates in days (daily)
AVG_SUPPLY_TEMP_(F)	Average supply temperature from DC Plant (Fahrenheit)
AVG-RETURN-TEMP_(F)	Average Return temperature from Customer side (Fahrenheit)
Delta_T_(F)	Difference Temperature (The difference between supply and return Temperature)
AVG_FLOW_(GPM)	Average Flow rate going to the customer (gallon/minute) [average of the day readings]
AVG_POWER_(TR)	Average power consumed by the customer (Ton of Refrigeration) [average of the day readings]
AVG_PEAK_POWER_(TR)	Peak (maximum) power consumed by the customer (Ton of Refrigeration) [average of the day readings]
ENERGY_(TRH)	Cumulative Readings of Power (Ton of Refrigeration)

To find the count of the data set per attribute, where it will give quick indication for next step, which data cleaning

```
[42] df.count() # counting the records of total variable.
```

Date	1060
AVG_SUPPLY_TEMP_(F)	1060
AVG_RETURN_TEMP_(F)	1060
Delta_T_(F)	1060
AVG_FLOW_(GPM)	1060
AVG_POWER_(TR)	1060
PEAK_POWER_(TR)	1060
ENERGY__(TRH)	1060
dtype: int64	

Figure 4: Dataset attributes count

Figure 5 shows that all attributes have 1,060 readings. In different words, it is 1,060 days. And it is clearly showing that all values were filled.

```
[43] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1060 entries, 0 to 1063
Data columns (total 8 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Date                                  1060 non-null   object
1   AVG_SUPPLY_TEMP_(F)                 1060 non-null   float64
2   AVG_RETURN_TEMP_(F)                 1060 non-null   float64
3   Delta_T_(F)                         1060 non-null   float64
4   AVG_FLOW_(GPM)                      1060 non-null   float64
5   AVG_POWER_(TR)                      1060 non-null   float64
6   PEAK_POWER_(TR)                     1060 non-null   float64
7   ENERGY__(TRH)                      1060 non-null   object
dtypes: float64(6), object(2)
memory usage: 74.5+ KB
```

Figure 5: Checking the attributes types

After a quick look at the above data types, two types are considered objects, Date and Energy readings, where it does not have decimals (the majority of the attributes). However, other attributes are considered float64, which means Floating point numbers. Also, this is important to the data analytics person, where need to see if it required to change any data type to another format, like Boolean, string, and integer.

7.1.2 Missing Data

From previous steps, it showed that data seems to be clean and ready to use, but it is crucial to check and evaluate the data before starting, so it will use some part of the panda library, and the total count of false/null/Nan will be counted for each attribute, and will divide that number by total values (1,060 as it was shown earlier) to get the percentage of the missing data.

```
[ ] #missing data
total = df.isnull().sum().sort_values(ascending=False)
percent = (df.isnull().sum()/df.isnull().count()).sort_values(ascending=False)
missing_df = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_df.head(6)
```

	Total	Percent
ENERGY__(TRH)	0	0.0
PEAK_POWER_(TR)	0	0.0
AVG_POWER_(TR)	0	0.0
AVG_FLOW_(GPM)	0	0.0
Delta_T_(F)	0	0.0
AVG_RETURN_TEMP_(F)	0	0.0

Figure 6: Missing data per attribute

As shown in the above figure, the data set is immaculate and ready to use. In general, it is knowing that the fewer missing values, more accurate results will reflect at later stages.

7.2 Data Preprocessing

7.2.1 Install all required libraries

As mentioned above, the model/coding language will be used Python, and Google Colab will be the area to develop/run this project.

Each project is required different libraries, which need to be installed at the beginning of the project, or later stages as per sections requirement. Some libraries are used to plot the values, and some libraries are used for data cleaning, etc.

▼ Installing and importing necessary libraries

```
[6] pip install pmdarima # installing the pmdarima packages

Requirement already satisfied: pmdarima in /usr/local/lib/python3.7/dist-packages (1.8.0)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.7/dist-packages (from pmdarima) (1.19.5)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in /usr/local/lib/python3.7/dist-packages (from pmdarima) (54.0.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from pmdarima) (1.0.1)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.7/dist-packages (from pmdarima) (1.4.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (from pmdarima) (1.24.3)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.7/dist-packages (from pmdarima) (0.22.2.post1)
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.7/dist-packages (from pmdarima) (1.1.5)
Requirement already satisfied: Cython<0.29.18,>=0.29 in /usr/local/lib/python3.7/dist-packages (from pmdarima) (0.29.17)
Requirement already satisfied: statsmodels!=0.12.0,>=0.11 in /usr/local/lib/python3.7/dist-packages (from pmdarima) (0.12.2)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.19->pmdarima) (2018.9)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.19->pmdarima) (2.8.1)
Requirement already satisfied: patsy>=0.5 in /usr/local/lib/python3.7/dist-packages (from statsmodels!=0.12.0,>=0.11->pmdarima) (0.5.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->pandas>=0.19->pmdarima) (1.15.0)

[7] import pandas as pd # library for data analysis
import numpy as np # Library for array manipulation
import matplotlib.pyplot as plt # Library for graph plotting
import sys # library for taking system permission
import sklearn # Library for ML model algorithm
from sklearn.model_selection import train_test_split # Library for nice graph visualization
import seaborn as sb # importing KNeighbors method from sklearn library
from sklearn.neighbors import KNeighborsClassifier # importing the tree algorithm from the sklearn library
from sklearn import tree # importing the confusion matrix for evaluations
from sklearn.metrics import confusion_matrix # importing ensemble for random forest model
from sklearn import ensemble # importing for KNN algorithm
from sklearn.preprocessing import StandardScaler # for finding the cost function
from sklearn.metrics import mean_squared_error
from pmdarima import auto_arima
import warnings
from statsmodels.tsa.arima_model import ARIMA # Importing ARIMA package
from math import sqrt # importing sqrt package for finding the error.
from statsmodels.tsa.stattools import adfuller # importing the adfuller method for autotrain the model.
from sklearn import svm # importing the svm library
```

Figure 7: Installed all Libraries

7.2.2 Data Cleaning Process.

As shown in section 7.1.2, the dataset is clean, and no further cleaning action is required. So, it can be used directly.

7.3 Data Mining

7.3.1 Exploratory Data Analysis

Dealing with temperatures and a limited range of data is easy, but it is essential to see the statistical information for each attribute in this data, like the average temperature supply/return, minimum & maximum. The most crucial part is knowing the readings for interquartile range (IQR), in a different way to study Boxplot, where it is used to divide the data into four quarters by 25%, where it can give people a quick understanding of value, the dispersion of the dataset, and signs of skewness.

df.describe()

	AVG_SUPPLY_TEMP_(F)	AVG_RETURN_TEMP_(F)	Delta_T_(F)	AVG_FLOW_(GPM)	AVG_POWER_(TR)	PEAK_POWER_(TR)
count	1080.000000	1080.000000	1080.000000	1080.000000	1080.000000	1080.000000
mean	39.046118	51.756387	12.710268	636.064086	329.367923	405.216888
std	9.219045	2.867186	2.507194	183.867955	143.043462	161.051784
min	36.000000	45.000000	5.000000	229.000000	114.000000	77.000000
25%	36.000000	46.867188	10.000000	503.750000	196.867188	266.750000
50%	36.625000	52.145833	12.460938	609.145833	304.000000	377.000000
75%	36.843750	54.000000	14.216250	813.250000	406.375000	501.000000
max	41.000000	57.000000	17.000000	1139.000000	641.750000	749.000000

Figure 8: Data Discription

```
import pandas as pd
import matplotlib.pyplot as plt
df = df.dropna(axis=0)
df = df.dropna(axis=0)
df['Date'] = pd.to_datetime(df['Date'])
plt.figure(figsize=(10,6))
plt.plot(df['Date'],df['AVG_FLOW_(GPM)'],'b-')
plt.ylabel('AVG_FLOW_(GPM)')
plt.grid()
plt.show()
plt.figure(figsize=(10,6))
plt.plot(df['Date'],df['AVG_POWER_(TR)'],'r-')
plt.ylabel('AVG_POWER_(TR)')
plt.grid()
plt.show()
plt.figure(figsize=(10,6))
plt.plot(df['Date'],df['PEAK_POWER_(TR)'],'k-')
plt.ylabel('PEAK_POWER_(TR)')
plt.xlabel('Date')
plt.grid()
plt.show()
```

It will presents a diverse range of utilities

Figure 9: Importing plotting library & build the code.

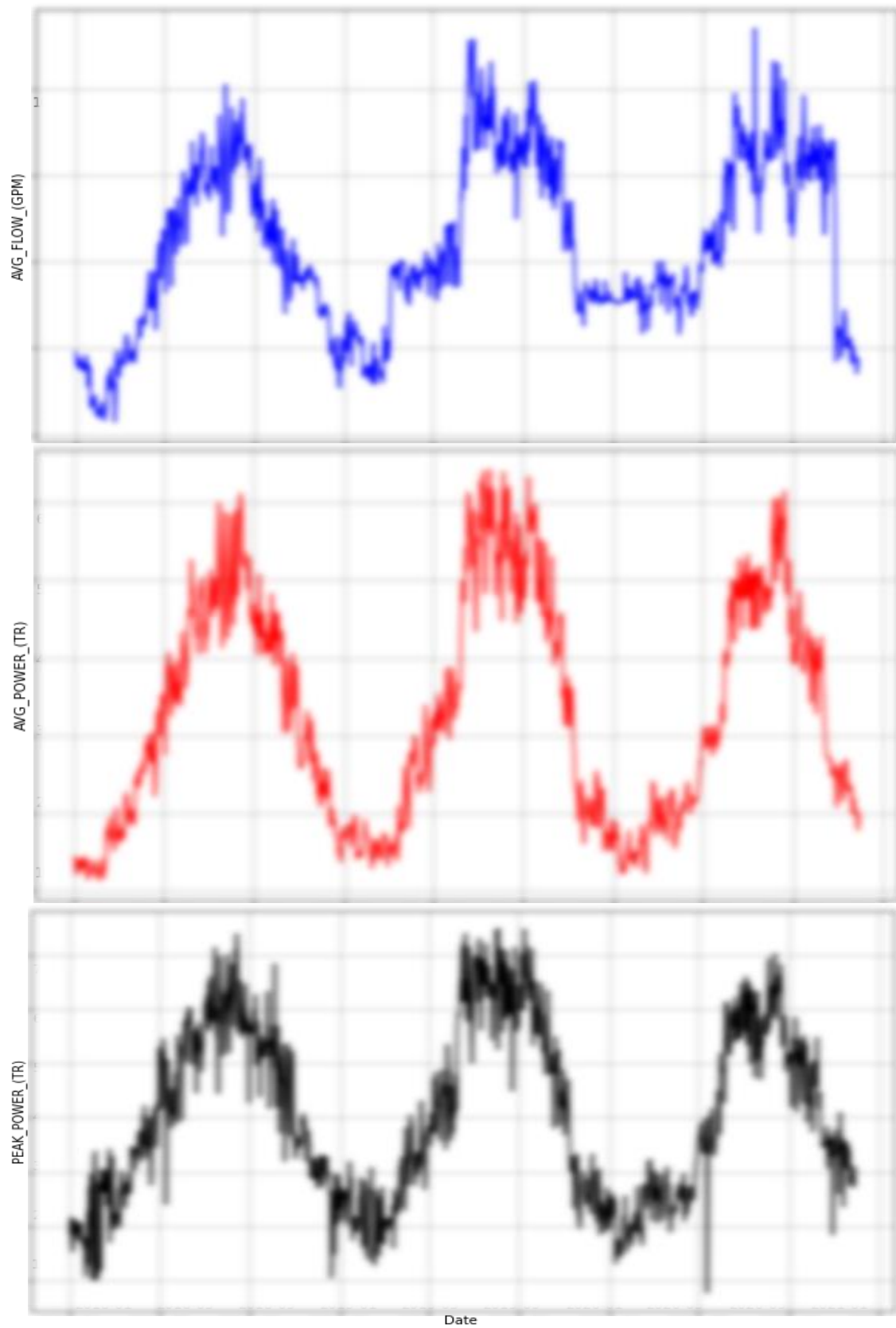


Figure 10: Average Flow, Average Power & Peak Power graphs *

From above, it is interested to plot some graphs like flow, power, and peak power, which is most related to the objectives of this project

It shows the first graph (blue line), representing the average flow, the red line showing Average Power, and the black graph showing the peak power (TR) for the same duration. Flow is a coolant liquid that needs to keep circulating in the network (dynamic system). If the flow went down, it means it is raising the temperature in the secondary side (customer side), and that what is showing during the winter season, where the secondary side does not have a high load, so the FCV is at a minimum opening set point. As subsequence, the power will go down also.

Comparing all three graphs clearly shows that it is repeated seasonal trends, where during the summer season, the values at the maximum, and lowest at winter. Also, it is observed that during one of the years the flow and power readings were a bit longer in duration, comparing to the previous year.

TR calculation can be found using below equation:

$$TR = \frac{flow (gpm) \times \Delta T (^{\circ}F)}{24 \text{ hours}} \quad (Eq.1)$$

This project will apply a few algorithms and tools that can predict the accurate values for flow, which is the objective for this paper, and will also check for power and peak power if possible. So, will use KNN & SVM algorithms along with the ARIMA tool. Those algorithms were selected because it is supervised machine learning based on the data set. Many other tools or algorithms might be used, but this depends on the user selection and his dataset applicability.

```
train = df.iloc[:300] # putting the dataset into the training variables.
test = df.iloc[300:] # putting the dataset into the testing part.

train.drop(["Date", "AVG_FLOW_GPM", "AVG_RETURN_TEMP_F", "AVG_POWER_TR", "PEAK_POWER_TR", "Delta_T(F)"], axis=1).values
train["AVG_FLOW_GPM"].values # putting the e_lfanew variable as target values.
```

Data divided into 70% training and 30% for testing; this will be used in all algorithms and tools. X is used for some attributes, and Y is used for flow, where x will be used to store the existing values, and Y will store the new values after testing.

Figure 12: Checking for x & y values *

24

number of autoregressive terms, d value represents the number of non-seasonal differences required for stationarity, d value represents the of number of slacked (lagged) estimate mistakes within the forecast condition equation.

7.3.2.1.1 Augmented Dickey-Fuller Test

This project deals with time series analysis; most statistical forecasting methods assume that the time series is approximability stationery. In a stationary time series, measurable and statistical properties such as mean, and change are steady over time. In a non-stationary series, these properties are depending on time and changing each time. So, the Augmented Dickey-Fuller Test (ad_test) is a well-known statistical test that can help determine if your time series is stationary.

```
def ad_test(dataset):                                     # defineing method for doing the adfuller test and try to analyse the method.
    dfctest=adfuller(dataset,autolag="AIC")
    print("1. ADF:", dfctest[0])
    print("2. P-value :", dfctest[1])
    print("3. Num of Lags: ", dfctest[2])
    print("4. Num Of Observations Used for ADF Regression and ADF values Calculations:",dfctest[3])
    print("5. Critical Values :")
    for key,val in dfctest[4].items():
        print("\t",key,":",val)
```

Figure 154: Augmented Data Fuller test and analysis

```
df['Date']=ad_test(df["AVG_FLOW_(GPM)"])                 # testing with respect the flow
```

```
1. ADF: -1.9999999999999999
2. P-value : 0.9999999999999999
3. Num of Lags: 10
4. Num Of Observations Used for ADF Regression and ADF values Calculations: 1000
5. Critical Values :
   ADF: -1.9999999999999999
   P-value: 0.9999999999999999
   Num of Lags: 10
```

```
df['Date']=ad_test(df["AVG_POWER_(TR)"])                 # testing with respect to power
```

```
1. ADF: -1.9999999999999999
2. P-value : 0.9999999999999999
3. Num of Lags: 10
4. Num Of Observations Used for ADF Regression and ADF values Calculations: 1000
5. Critical Values :
   ADF: -1.9999999999999999
   P-value: 0.9999999999999999
   Num of Lags: 10
```

```
df['Date']=ad_test(df["PEAK_POWER_(TR)"])                 # testing with respect to Peak_Power
```

```
1. ADF: -1.9999999999999999
2. P-value : 0.9999999999999999
3. Num of Lags: 10
4. Num Of Observations Used for ADF Regression and ADF values Calculations: 1000
5. Critical Values :
   ADF: -1.9999999999999999
   P-value: 0.9999999999999999
   Num of Lags: 10
```

Figure 145: Augmented statistical Testing with respect to Flow, Power & Peak Power *

This step is very important before doing ARIMA time series forecasting. Will apply this test on the main three attributes (Flow, Power, Peak_Power)

The Augmented Dickey-Fuller (ADF) statistic test is used to determine whether if data should be first differenced or regressed on the deterministic function of time to render the data stationary. And actual testing can either be a Null hypothesis or an Alternative hypothesis.

- Null hypothesis (H_0): its declaration about the population that either is supposed to be true (accurate) or is used to put into the open argument, unless it can be proved to be incorrect beyond a reasonable doubt.
- Alternative hypothesis (H_a): claims about the population that opposing to (H_0) and what was mentioned in (H_0) when it was rejected.
 - ADF: The more negative it is, the more grounded the dismissal of the theory that there is a unit root at a few levels of certainty and confidence.
 - P-Values: A smaller p-value means that there is stronger evidence in favor of the alternative hypothesis.
 - Num of lags: Lag is essentially delay. Just as correlation shows how much two time series are similar, autocorrelation describes how similar the time series is with itself.
 - Num of Observations used: Critical Values: Confidence values at 1%, 5% and 10% confidence interval.

Summary of testing of Augmented Dickey-Fuller Test, P-values are more than 0.05, is the probability that the null hypothesis is true, and no effect was observed.

7.3.2.1.2 ARIMA Model: Average Flow

```
model_fit = auto_arima(df["AVG_FLOW_(GPM)"], trace=True, suppress_warnings = True) # fitting the arima model
```

Performing stepwise search to minimize aic

ARIMA(2,1,2)(0,0,0)[0]	intercept	: AIC=11256.376, Time=0.70 sec
ARIMA(0,1,0)(0,0,0)[0]	intercept	: AIC=11359.763, Time=0.04 sec
ARIMA(1,1,0)(0,0,0)[0]	intercept	: AIC=11330.439, Time=0.09 sec
ARIMA(0,1,1)(0,0,0)[0]	intercept	: AIC=11307.872, Time=0.30 sec
ARIMA(0,1,0)(0,0,0)[0]		: AIC=11357.763, Time=0.04 sec
ARIMA(1,1,2)(0,0,0)[0]	intercept	: AIC=11256.403, Time=0.35 sec
ARIMA(2,1,1)(0,0,0)[0]	intercept	: AIC=11255.509, Time=0.68 sec
ARIMA(1,1,1)(0,0,0)[0]	intercept	: AIC=11260.743, Time=0.54 sec
ARIMA(2,1,0)(0,0,0)[0]	intercept	: AIC=11290.981, Time=0.16 sec
ARIMA(3,1,1)(0,0,0)[0]	intercept	: AIC=11256.749, Time=1.09 sec
ARIMA(3,1,0)(0,0,0)[0]	intercept	: AIC=11273.647, Time=0.19 sec
ARIMA(3,1,2)(0,0,0)[0]	intercept	: AIC=11256.005, Time=2.61 sec
ARIMA(2,1,1)(0,0,0)[0]		: AIC=11253.509, Time=0.30 sec
ARIMA(1,1,1)(0,0,0)[0]		: AIC=11258.743, Time=0.22 sec
ARIMA(2,1,0)(0,0,0)[0]		: AIC=11288.981, Time=0.07 sec
ARIMA(3,1,1)(0,0,0)[0]		: AIC=11254.749, Time=0.39 sec
ARIMA(2,1,2)(0,0,0)[0]		: AIC=11254.376, Time=0.33 sec
ARIMA(1,1,0)(0,0,0)[0]		: AIC=11328.439, Time=0.16 sec
ARIMA(1,1,2)(0,0,0)[0]		: AIC=11254.403, Time=0.17 sec
ARIMA(3,1,0)(0,0,0)[0]		: AIC=11271.647, Time=0.09 sec
ARIMA(3,1,2)(0,0,0)[0]		: AIC=11254.005, Time=0.95 sec

Best model: ARIMA(2,1,1)(0,0,0)[0]

Figure 16: Fitting ARIMA model with respect to average flow

From Fig. 16 (red box) the results showed that the best model for flow is 2,1,1 respectively with $p = 2$, $d = 1$, and $q = 1$

```
# fitting ARIMA to average flow with p = 2, d = 1 and q = 1
model = ARIMA(avg_flow, order=(2,1,1))
model_fit = model.fit()
print(model_fit.summary())
from pandas import DataFrame
residuals = DataFrame(model_fit.resid)
residuals.plot(legend=None)
plt.title('Residuals plot')
plt.show()
print('Residual statistics:\n', residuals.describe())
avg_flow_preds = model_fit.predict(start=0, end=len(df)-1)
avg_flow_preds.plot.line()
plt.ylabel('Predicted Average flow')
plt.show()
```

SARIMAX Results

```
=====
Dep. Variable:          avg_flow      No. Observations:          1060
Model:                ARIMA(2, 1, 1)  Log Likelihood          -5627.585
Date:                 Mon, 29 Mar 2021  AIC                    11263.170
Time:                 15:42:08         BIC                    11283.030
Sample:              01-01-2018       HQIC                    11270.697
                             - 11-29-2020
Covariance Type:      opg
=====
```

	coef	std err	z	P> z	[0.025	0.975]
ar.L1	0.4030	0.057	7.023	0.000	0.291	0.515
ar.L2	-0.1035	0.031	-3.342	0.001	-0.164	-0.043
ma.L1	-0.6564	0.058	-11.262	0.000	-0.771	-0.542
sigma2	2416.3277	53.379	45.267	0.000	2311.707	2520.949

```
=====
Ljung-Box (L1) (Q):                0.01  Jarque-Bera (JB):                1486.23
Prob(Q):                          0.94  Prob(JB):                      0.00
Heteroskedasticity (H):            1.58  Skew:                          -0.22
Prob(H) (two-sided):              0.00  Kurtosis:                      8.79
=====
```

Figure 17: ARIMA average flow details

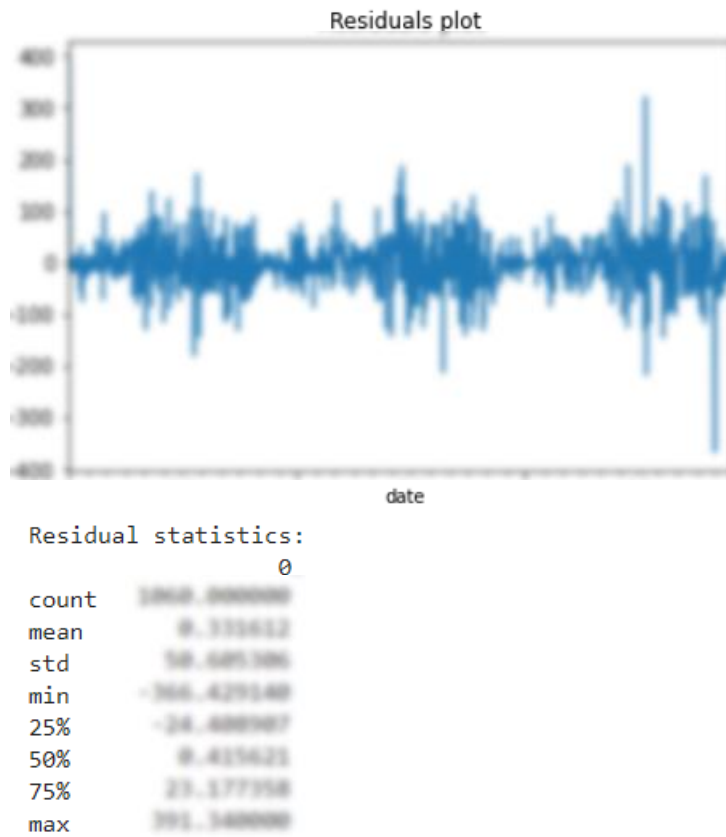


Figure 18: Residual Statistics details (for Average Flow)*

The residual means the difference between the observed values and the model's mean values predicting that observation. Also, it is observed that some negative values, which means the actual value was less than the predicted values, and the same for positive values, where the actual values were more than the predicted values. So, it is better to do plot the predicted values.

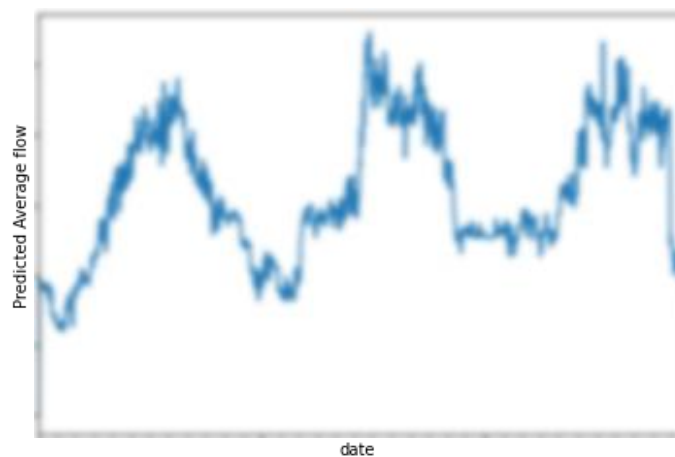


Figure 19: Testing details (for Average Flow) *

7.3.2.1.3 ARIMA Model: Average Power

```

model_fit = auto_arima(df["avg_power"], trace=True, suppress_warnings = True) # fitting the arima model

Performing stepwise search to minimize aic
ARIMA(2,1,2)(0,0,0)[0] intercept : AIC=inf, Time=1.84 sec
ARIMA(0,1,0)(0,0,0)[0] intercept : AIC=10018.911, Time=0.04 sec
ARIMA(1,1,0)(0,0,0)[0] intercept : AIC=10020.894, Time=0.10 sec
ARIMA(0,1,1)(0,0,0)[0] intercept : AIC=10020.880, Time=0.28 sec
ARIMA(0,1,0)(0,0,0)[0] intercept : AIC=10016.915, Time=0.03 sec
ARIMA(1,1,1)(0,0,0)[0] intercept : AIC=9965.645, Time=0.49 sec
ARIMA(2,1,1)(0,0,0)[0] intercept : AIC=9915.541, Time=0.67 sec
ARIMA(2,1,0)(0,0,0)[0] intercept : AIC=9964.627, Time=0.16 sec
ARIMA(3,1,1)(0,0,0)[0] intercept : AIC=9916.981, Time=1.02 sec
ARIMA(1,1,2)(0,0,0)[0] intercept : AIC=9921.804, Time=0.96 sec
ARIMA(3,1,0)(0,0,0)[0] intercept : AIC=9946.800, Time=0.20 sec
ARIMA(3,1,2)(0,0,0)[0] intercept : AIC=9914.113, Time=1.89 sec
ARIMA(4,1,2)(0,0,0)[0] intercept : AIC=9878.131, Time=2.87 sec
ARIMA(4,1,1)(0,0,0)[0] intercept : AIC=9912.225, Time=1.26 sec
ARIMA(5,1,2)(0,0,0)[0] intercept : AIC=9875.828, Time=3.35 sec
ARIMA(5,1,1)(0,0,0)[0] intercept : AIC=9900.481, Time=1.33 sec
ARIMA(5,1,3)(0,0,0)[0] intercept : AIC=9878.266, Time=3.65 sec
ARIMA(4,1,3)(0,0,0)[0] intercept : AIC=9874.100, Time=2.93 sec
ARIMA(3,1,3)(0,0,0)[0] intercept : AIC=9913.668, Time=2.75 sec
ARIMA(4,1,4)(0,0,0)[0] intercept : AIC=9853.370, Time=4.08 sec
ARIMA(3,1,4)(0,0,0)[0] intercept : AIC=9868.542, Time=3.51 sec
ARIMA(5,1,4)(0,0,0)[0] intercept : AIC=9855.945, Time=4.35 sec
ARIMA(4,1,5)(0,0,0)[0] intercept : AIC=9856.039, Time=4.86 sec
ARIMA(3,1,5)(0,0,0)[0] intercept : AIC=9872.525, Time=4.50 sec
ARIMA(5,1,5)(0,0,0)[0] intercept : AIC=9857.170, Time=5.20 sec
ARIMA(4,1,4)(0,0,0)[0] intercept : AIC=9851.455, Time=3.40 sec
ARIMA(3,1,4)(0,0,0)[0] intercept : AIC=9866.529, Time=2.00 sec
ARIMA(4,1,3)(0,0,0)[0] intercept : AIC=9871.982, Time=1.59 sec
ARIMA(5,1,4)(0,0,0)[0] intercept : AIC=9855.416, Time=2.35 sec
ARIMA(4,1,5)(0,0,0)[0] intercept : AIC=9854.554, Time=2.62 sec
ARIMA(3,1,3)(0,0,0)[0] intercept : AIC=9911.683, Time=1.07 sec
ARIMA(3,1,5)(0,0,0)[0] intercept : AIC=9870.264, Time=2.58 sec
ARIMA(5,1,3)(0,0,0)[0] intercept : AIC=9876.157, Time=1.95 sec
ARIMA(5,1,5)(0,0,0)[0] intercept : AIC=9855.253, Time=3.09 sec

Best model: ARIMA(4,1,4)(0,0,0)[0]
Total fit time: 73.027 seconds

```

Figure 20: Fitting ARIMA model with respect to average power

From Fig. 20 (red box) the results showed that the best model for flow is 4,1,4 respectively with $p = 4$, $d = 1$, and $q = 4$.

```

=====
SARIMAX Results
=====
Dep. Variable:          avg_power      No. Observations:          1060
Model:                ARIMA(4, 1, 4)  Log Likelihood             -4916.727
Date:                 Mon, 29 Mar 2021 AIC                          9851.455
Time:                 15:43:25       BIC                          9896.141
Sample:               01-01-2018     HQIC                         9868.391
- 11-29-2020
Covariance Type:      opg
=====
              coef    std err          z      P>|z|      [0.025    0.975]
-----
ar.L1         0.7178      0.040     17.946      0.000      0.639      0.796
ar.L2        -1.3056      0.028    -47.391      0.000     -1.360     -1.252
ar.L3         0.7842      0.030     25.923      0.000      0.725      0.843
ar.L4        -0.8207      0.037    -22.156      0.000     -0.893     -0.748
ma.L1        -0.7648      0.049    -15.636      0.000     -0.861     -0.669
ma.L2         1.1705      0.038     30.970      0.000      1.096      1.245
ma.L3        -0.8570      0.036    -24.048      0.000     -0.927     -0.787
ma.L4         0.6636      0.050     13.186      0.000      0.565      0.762
sigma2       656.9184     21.016     31.258      0.000     615.728     698.109
=====
Ljung-Box (L1) (Q):                0.19   Jarque-Bera (JB):                263.07
Prob(Q):                           0.67   Prob(JB):                        0.00
Heteroskedasticity (H):              0.82   Skew:                            0.04
Prob(H) (two-sided):                0.06   Kurtosis:                       5.44

```

Figure 21: ARIMA average power details

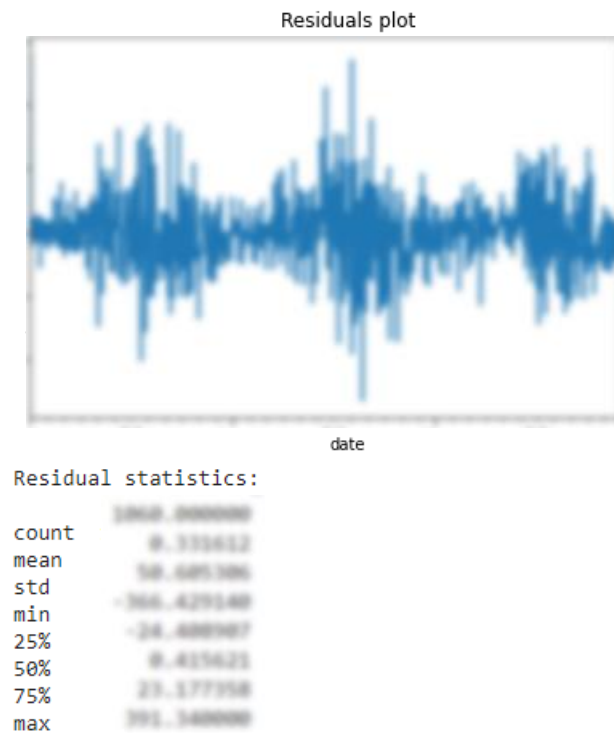


Figure 22: Residual Statistics details (for Average Power) *

Fig.22 shows many fluctuating residual values, but it is better than Average flow residual values, where many negative values were observed. Also, the std became down, and means the fluctuation is very hard here, and this might impact on the predicted values later stage.

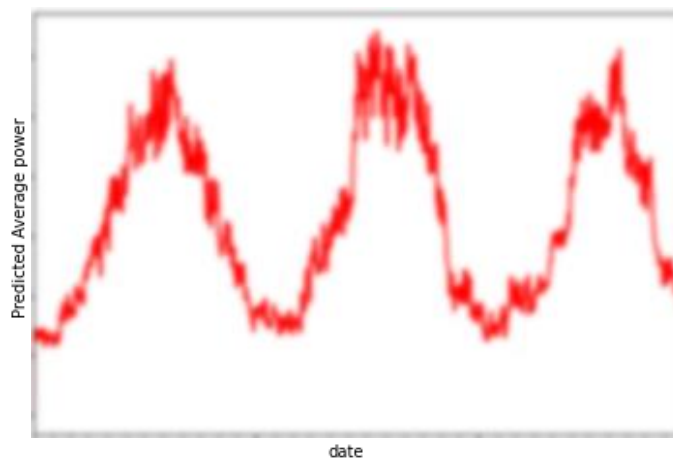


Figure 23: Testing details (for Average Power) *

The above trend shows the average power after testing, and it seems fine and normal.

7.3.2.1.4 ARIMA Model: Peak Power

```
model_fit = auto_arima(df["peak_power"], trace=True, suppress_warnings = True) # fitting the arima model

Performing stepwise search to minimize aic
ARIMA(2,1,2)(0,0,0)[0] intercept : AIC=11325.850, Time=1.63 sec
ARIMA(0,1,0)(0,0,0)[0] intercept : AIC=11586.477, Time=0.05 sec
ARIMA(1,1,0)(0,0,0)[0] intercept : AIC=11480.979, Time=0.09 sec
ARIMA(0,1,1)(0,0,0)[0] intercept : AIC=11366.318, Time=0.31 sec
ARIMA(0,1,0)(0,0,0)[0] intercept : AIC=11584.481, Time=0.03 sec
ARIMA(1,1,2)(0,0,0)[0] intercept : AIC=11323.583, Time=0.88 sec
ARIMA(0,1,2)(0,0,0)[0] intercept : AIC=11323.373, Time=0.36 sec
ARIMA(0,1,3)(0,0,0)[0] intercept : AIC=11321.916, Time=0.86 sec
ARIMA(1,1,3)(0,0,0)[0] intercept : AIC=11318.099, Time=1.70 sec
ARIMA(2,1,3)(0,0,0)[0] intercept : AIC=11253.681, Time=2.52 sec
ARIMA(3,1,3)(0,0,0)[0] intercept : AIC=11267.296, Time=3.14 sec
ARIMA(2,1,4)(0,0,0)[0] intercept : AIC=11248.679, Time=3.23 sec
ARIMA(1,1,4)(0,0,0)[0] intercept : AIC=11313.819, Time=1.61 sec
ARIMA(3,1,4)(0,0,0)[0] intercept : AIC=11256.527, Time=3.47 sec
ARIMA(2,1,5)(0,0,0)[0] intercept : AIC=11250.576, Time=3.98 sec
ARIMA(1,1,5)(0,0,0)[0] intercept : AIC=11315.813, Time=2.09 sec
ARIMA(3,1,5)(0,0,0)[0] intercept : AIC=11247.580, Time=4.52 sec
ARIMA(4,1,5)(0,0,0)[0] intercept : AIC=11231.178, Time=4.84 sec
ARIMA(4,1,4)(0,0,0)[0] intercept : AIC=11252.245, Time=4.09 sec
ARIMA(5,1,5)(0,0,0)[0] intercept : AIC=inf, Time=5.22 sec
ARIMA(5,1,4)(0,0,0)[0] intercept : AIC=11244.047, Time=4.44 sec
ARIMA(4,1,5)(0,0,0)[0] intercept : AIC=11229.480, Time=2.56 sec
ARIMA(3,1,5)(0,0,0)[0] intercept : AIC=11245.532, Time=2.47 sec
ARIMA(4,1,4)(0,0,0)[0] intercept : AIC=11250.286, Time=2.28 sec
ARIMA(5,1,5)(0,0,0)[0] intercept : AIC=inf, Time=3.73 sec
ARIMA(3,1,4)(0,0,0)[0] intercept : AIC=11254.156, Time=2.05 sec
ARIMA(5,1,4)(0,0,0)[0] intercept : AIC=11242.087, Time=2.47 sec

Best model: ARIMA(4,1,5)(0,0,0)[0]
Total fit time: 64.688 seconds
```

Figure 24: Fitting ARIMA model with respect to peak power

From Fig. 24 (red box) the results showed that the best model for flow is 4,1,5 respectively with $p = 4$, $d = 1$, and $q = 5$.

```

              SARIMAX Results
=====
Dep. Variable:      peak_power      No. Observations:      1060
Model:              ARIMA(2, 1, 1)  Log Likelihood        -5656.152
Date:              Sat, 27 Mar 2021  AIC                    11320.303
Time:              11:40:52          BIC                    11340.164
Sample:            01-01-2018        HQIC                   11327.831
                  - 11-29-2020

Covariance Type:    opg
=====
              coef      std err          z      P>|z|      [0.025      0.975]
-----
ar.L1           0.2639       0.042       6.221     0.000       0.181       0.347
ar.L2          -0.0616       0.036      -1.716     0.086      -0.132       0.009
ma.L1          -0.7665       0.034     -22.561     0.000      -0.833      -0.700
sigma2         2549.4380      79.833      31.935     0.000     2392.969     2705.907
=====
Ljung-Box (L1) (Q):              0.03  Jarque-Bera (JB):              235.80
Prob(Q):                        0.85  Prob(JB):                  0.00
Heteroskedasticity (H):          0.73  Skew:                      -0.50
Prob(H) (two-sided):            0.00  Kurtosis:                   5.09
=====

```

Figure 25: ARIMA model details for peak power

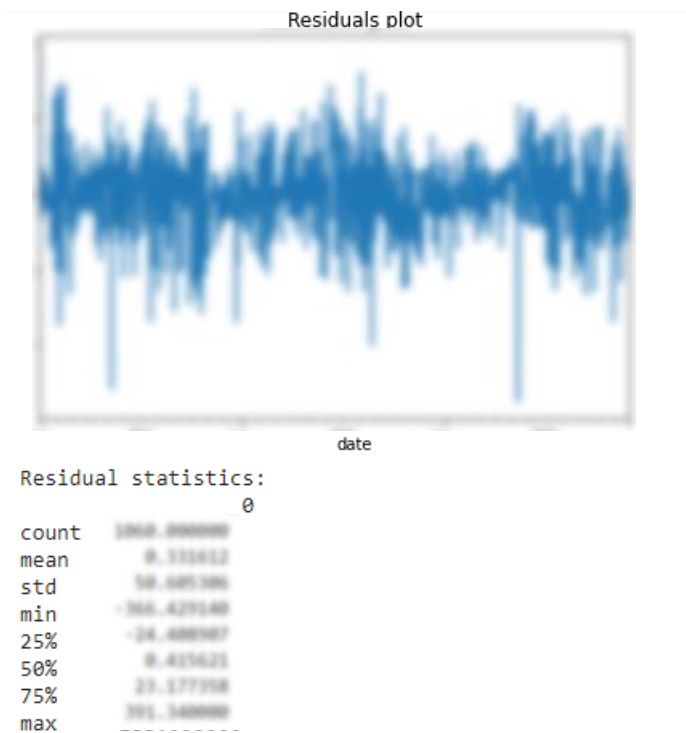


Figure 26: Residual Statistics details (for peak power)*

Still the negative values are repeated in peak power, and even it is more than positive values, and major drop like 5 times.

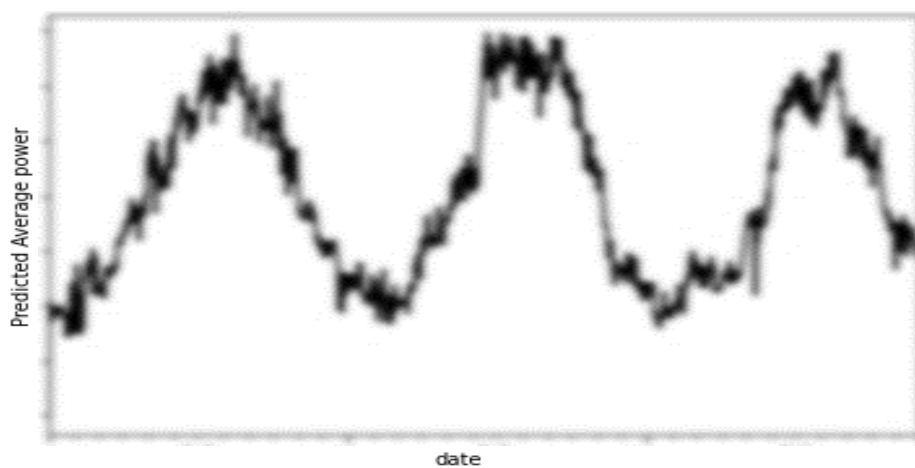


Figure 27: Testing details (for peak Power) *

Fig. 27 showing testing values for peak power, and it looks like normal dataset patterns.

7.3.2.2 KNN Section

KNN algorithm is one of the highest popular algorithms in machine learning for regression. It uses data and classifies new data points based on similarity measures. As long as the K values are increasing, the accuracy might increase; sometimes, it is about trial and error process. In this case, the best value for K was at K=2. Keep in mind that the lower values of error are better, with high accuracy.

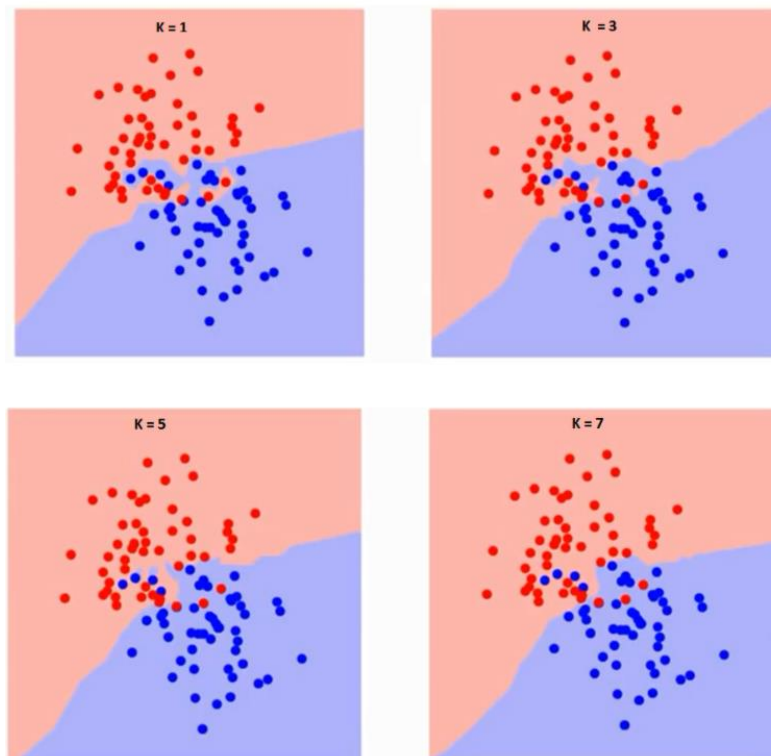


Figure 28: KNN classification

To understand the meaning of this algorithm, Fig. 28 shows that there are two classes, red and blue; when K values are smaller (top left), it is clearly showing that the grouping is much better and grouped well. However, referring to the value of k at 7 (bottom right), the classification of the red region is covering some blue values, which is not recommended, so the lower k value, the better classification. The training error rate and the validation error rate are two parameters that need to access different k values.

7.3.2.2.1 KNN Model: Average Flow

```
df = df.dropna(axis=0)
df = df.dropna(axis=0)

df['Date'] = pd.to_datetime(df['Date'])

X = np.array(df['Date']).reshape(-1,1)
y = np.array(df['AVG_FLOW_(GPM)']).reshape(-1,1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=501)
knn = KNeighborsRegressor(n_neighbors=2)
knn.fit(X_train,y_train)
ypreds = knn.predict(X_test)
plt.figure(figsize=(10,7))
plt.plot(X_test,y_test,'b.',X_test,ypreds,'r.')
plt.xlabel('Date')
plt.ylabel('AVG_FLOW_(GPM)')
plt.title('Comparison of actual and predicted data for test set of Flow using KNN')
plt.legend(['Actual','Predicted'])
plt.show()
print('KNN model Mean Absolute Error (MAE) on test set of average Flow data:',mae(y_test,ypreds))
```

Figure 30: KNN Algorithm training/ testing with respect to Average Flow

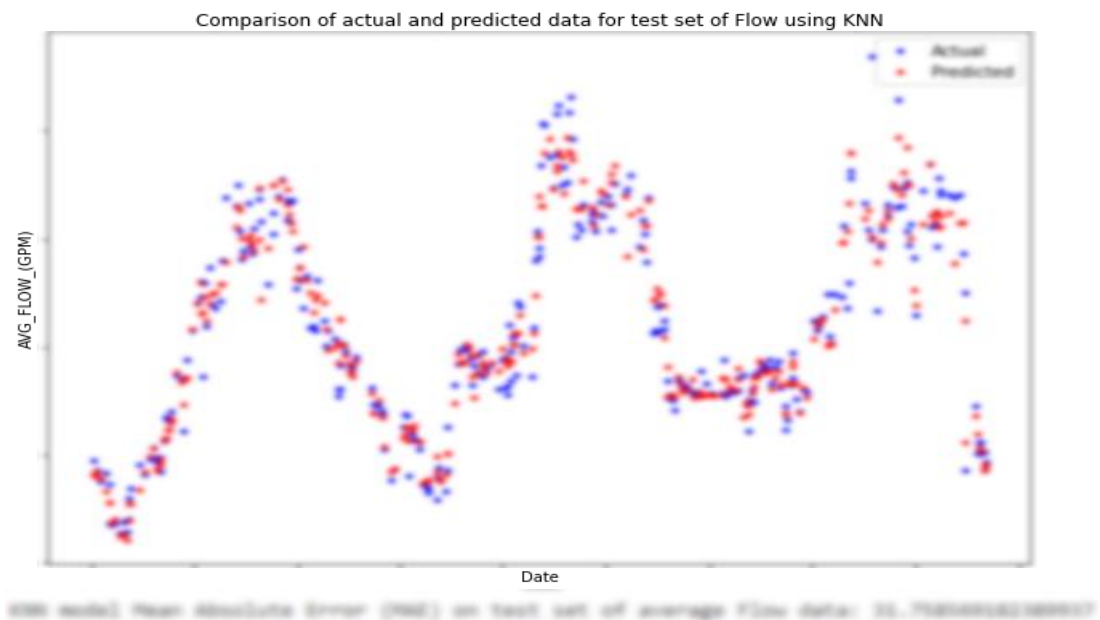


Figure 29: KNN testing/training output for Average Flow

From Fig. 26 it is showing the predicted values (red points) are close to the actual (blue points) while using $k = 2$, and the Mean Absolute Error is high, and tried to change the value of K to different numbers, but found it became less accurate, hence kept $k = 2$ with best accuracy.

7.3.2.2.2 KNN Model: Average Power

```
df = df.dropna(axis=0)
df = df.dropna(axis=0)

df['Date'] = pd.to_datetime(df['Date'])

X = np.array(df['Date']).reshape(-1,1)
y = np.array(df['AVG_POWER(TR)']).reshape(-1,1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=501)
knn = KNeighborsRegressor(n_neighbors=2)
knn.fit(X_train,y_train)
ypreds = knn.predict(X_test)
plt.figure(figsize=(10,7))
plt.plot(X_test,y_test,'b.',X_test,ypreds,'r.')
plt.xlabel('Date')
plt.ylabel('AVG_POWER_(TR)')
plt.title('Comparison of actual and predicted data for test set of Power using KNN')
plt.legend(['Actual','Predicted'])
plt.show()
print('KNN model Mean Absolute Error (MAE) on test set of average power data:',mae(y_test,ypreds))
```

Figure 31: KNN Algorithm training/ testing with respect to Average Power

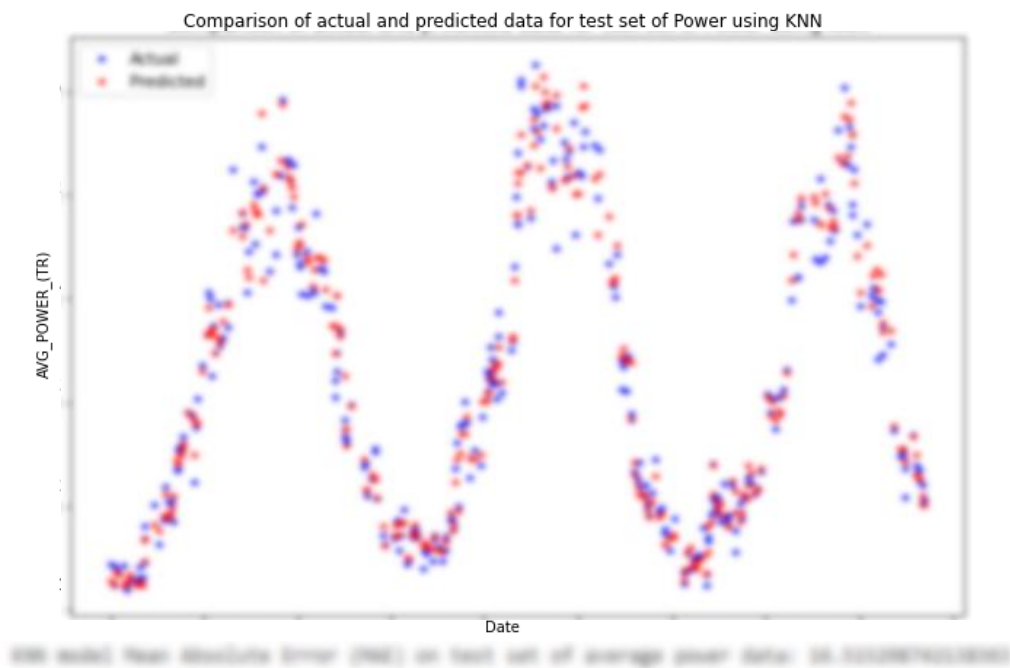


Figure 32: KNN testing/training output for Average Power

Also, for Average power, the values were regular, predicted values (red points) are close to the actual (blue points), and the best output was founded when $k=2$, and the MAE became less.

7.3.2.2.3 KNN Model: Peak Power

```
X = np.array(df['Date']).reshape(-1,1)
y = np.array(df['PEAK_POWER_(TR)']).reshape(-1,1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=501)
knn = KNeighborsRegressor(n_neighbors=2)
knn.fit(X_train,y_train)
ypreds = knn.predict(X_test)
plt.figure(figsize=(10,7))
plt.plot(X_test,y_test,'b.',X_test,ypreds,'r.')
plt.xlabel('Date')
plt.ylabel('PEAK_POWER_(TR)')
plt.title('Comparison of actual and predicted data for test set of Peak Power using KNN')
plt.legend(['Actual', 'Predicted'])
plt.show()
print('KNN model Mean Absolute Error (MAE) on test set of peak power data:',mae(y_test,ypreds))
```

Figure 34: KNN Algorithm training/ testing with respect to Peak Power

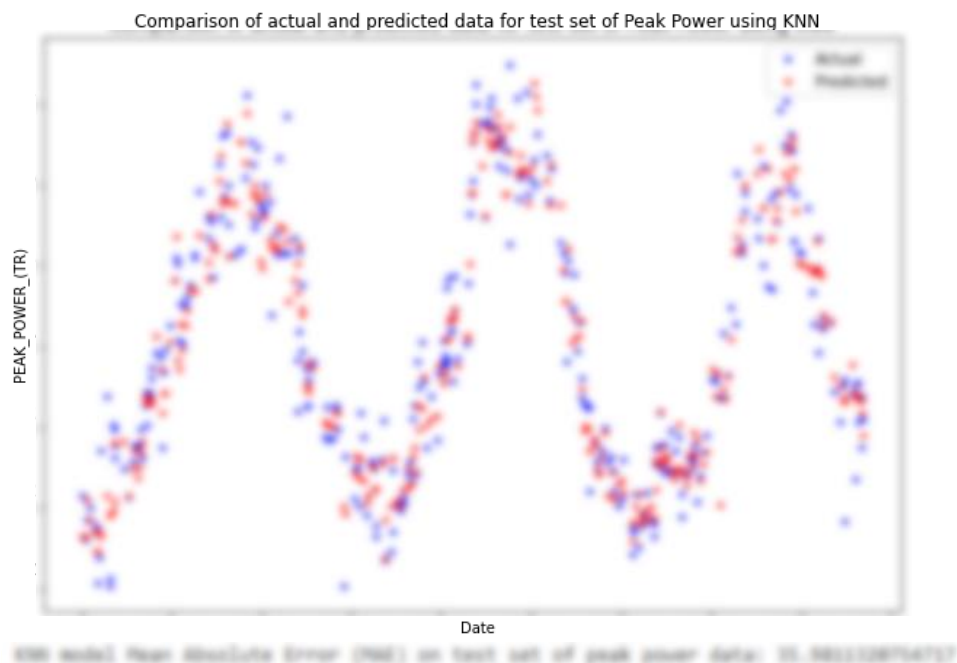


Figure 33: KNN testing/training output for Peak Power

Predicted values (red points) are close to the actual (blue points). From above trends, it is showing variance of accuracy.

7.3.2.3 SVM Section

In Support Vector Machine (SVM) algorithm, it will plot individual numbers item as a point in n-dimensional space (where n represent the number of features that dataset has), with the esteem of each point being the value of a specific coordinate. Then, it will perform classification by finding the hyper-plane that differentiates the two classes very well. The C (classification) parameter trades off the correct classification of training examples against maximization of the decision function's margin. For larger values of C, a smaller margin will be accepted if the decision function is better at classifying all training points correctly. A lower C will encourage a more significant margin, therefore a more straightforward decision function, at the cost of training accuracy. In other words, C behaves as a regularization parameter in the SVM. In this case, the C was at best at $C = 1,000,000$. This might not give an accurate output, and subsequent might impact the future predicted values. So, 70:30 train/test split is used. Fitted model on train set used to predict on the test set.

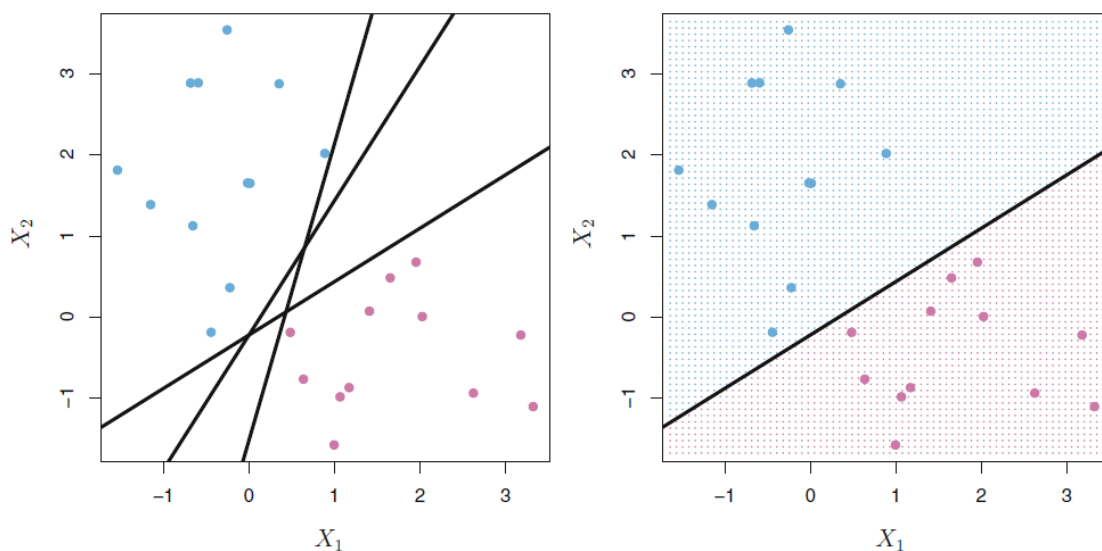


Figure 35: Support Vector Machines Classification

There are two classes of observations, showing in blue and purple, each of which has measurements on two variables. On the left graph: three separating hyperplanes (out of many) lines are showing in black. On the right graph: separating hyperplane is showing a single line in black. Both indicating the decision rule made by the classifier based on the separated lines, which means whatever is falling in the blue region will consider as a blue class, and whatever is falling in the purple side will consider as a purple class.

7.3.2.3.1 SVM Model: Average Flow

```
regr = make_pipeline(StandardScaler(), SVR(C=1000000, epsilon=0.3))
X = np.array(df['Date']).reshape(-1,1)
y = np.array(df['AVG_FLOW_(GPM)'])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=301)
regr.fit(X_train, y_train)
ypreds = regr.predict(X_test)
plt.figure(figsize=(10,7))
plt.plot(X_test,y_test,'b.',X_test,ypreds,'r.')
plt.xlabel('Date')
plt.ylabel('AVG_FLOW_(GPM)')
plt.title('Comparison of actual and predicted data for test set')
plt.legend(['Actual','Predicted'])
plt.show()
print('SVM model mean absolute error on test set of average flow data:',mae(y_test,ypreds))
```

Figure 36: SVM Algorithm training/ testing with respect to Average Flow

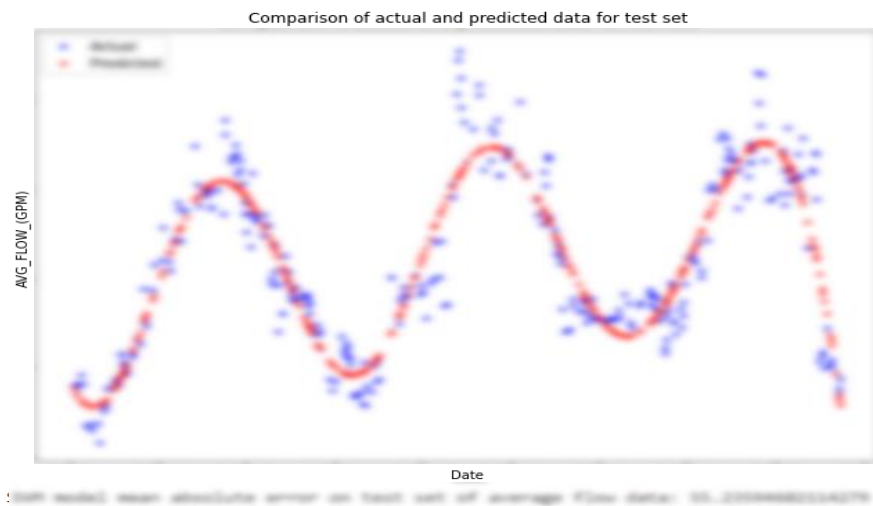


Figure 37: SVM testing/training output for Average Flow

From Fig. 37, the predicted values (red points) are not that much close to actual values (blue points), even when C values changed up to 1 Million, and many trial and error was made to reach to this shape.

7.3.2.3.2 SVM Model: Average Power

```
regr = make_pipeline(StandardScaler(), SVR(C=1000000, epsilon=0.3))
X = np.array(df['Date']).reshape(-1,1)
y = np.array(df['AVG_POWER_(TR)'])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=301)
regr.fit(X_train, y_train)
ypreds = regr.predict(X_test)
plt.figure(figsize=(10,7))
plt.plot(X_test, y_test, 'b.', X_test, ypreds, 'r.')
plt.xlabel('Date')
plt.ylabel('AVG_POWER_(TR)')
plt.title('Comparison of actual and predicted data for test set')
plt.legend(['Actual', 'Predicted'])
plt.show()
print('SVM model mean absolute error on test set of average power data:', mae(y_test, ypreds))
```

Figure 38: SVM Algorithm training/ testing with respect to Average Power

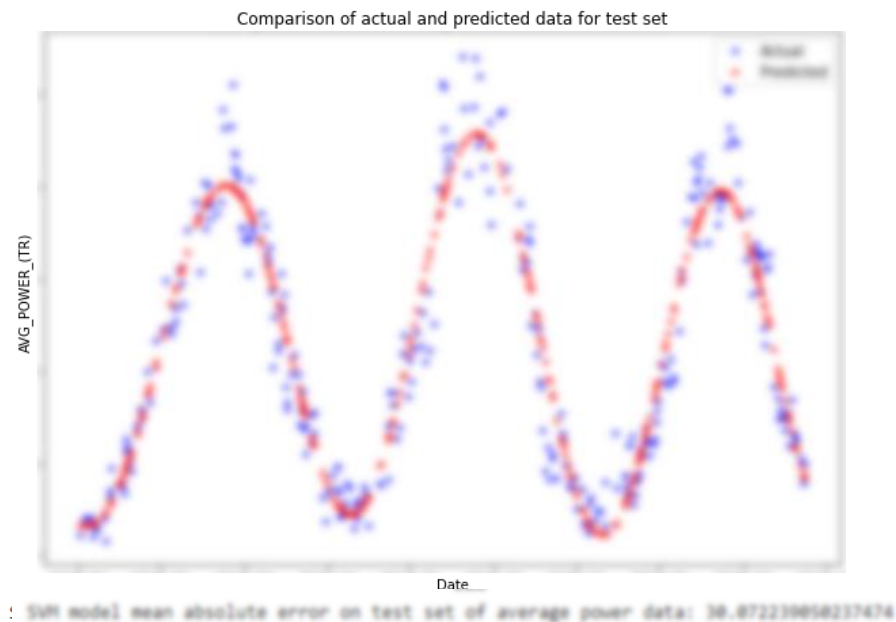


Figure 39: SVM testing/training output for Average Power

From Fig. 39, the predicted values (red points) are not that much close to actual values (blue points), even when C values changed up to 1 Million, and many trial and error was made to reach to this shape.

7.3.2.3.3 SVM Model: Peak Power

```
regr = make_pipeline(StandardScaler(), SVR(C=1000000, epsilon=0.3))
X = np.array(df['Date']).reshape(-1,1)
y = np.array(df['PEAK_POWER_(TR)'])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=301)
regr.fit(X_train, y_train)
ypreds = regr.predict(X_test)
plt.figure(figsize=(10,7))
plt.plot(X_test, y_test, 'b.', X_test, ypreds, 'r.')
plt.xlabel('Date')
plt.ylabel('PEAK_POWER_(TR)')
plt.title('Comparison of actual and predicted data for test set')
plt.legend(['Actual', 'Predicted'])
plt.show()
print('SVM model mean absolute error on test set of average flow data:', mae(y_test, ypreds))
```

Figure 40: SVM Algorithm training/ testing with respect to Peak Power

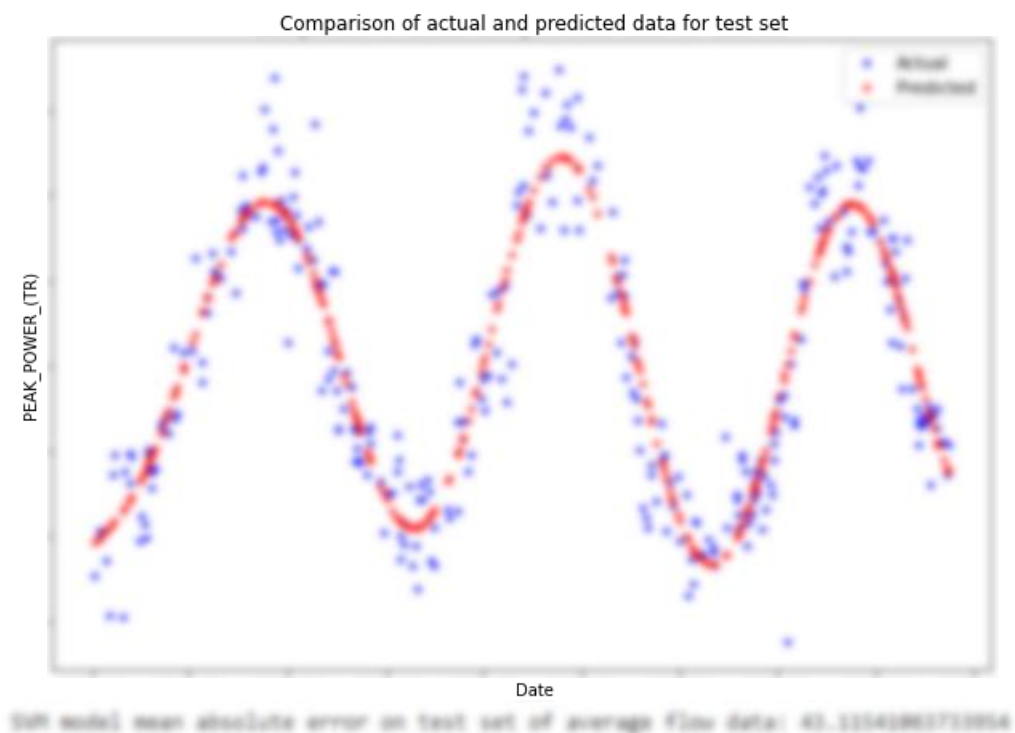


Figure 41: SVM testing/training output for Peak Power

From Fig. 41, the predicted values (red points) are not that much close to actual values (blue points), even when C values changed up to 1 Million, and many trial and error was made to reach to this shape.

7.3.3 Evaluating the models and finding the summary of trained model

In the above section covered under 7.3.2, it was found that the accuracy is not stable and not close to each other. This proves how each algorithm is dealing with the dataset. For data analytics people, it is crucial to evaluate the outcomes and proceed with the most suitable method to achieve the expected results with less risks, especially while dealing with decision-makers people.

7.3.3.1 Evaluating ARIMA model

```
test["AVG_FLOW_(GPM)"].mean() # calculating the mean values
```

213.5176518032434

```
rmse = sqrt(mean_squared_error(model_pred1,test["AVG_FLOW_(GPM)"])) # finding the errors
print(rmse)
```

213.5176518032434

```
test["AVG_POWER_(TR)"].mean() # checking the mean values of model
```

334.66499999999999

```
rmse = sqrt(mean_squared_error(model2_pred,test["AVG_POWER_(TR)"])) # find the erros of model
print(rmse)
```

190.1877094647779

```
test["PEAK_POWER_(TR)"].mean()
```

407.49666666666667

```
rmse = sqrt(mean_squared_error(model3_pred,test["PEAK_POWER_(TR)"])) # find the erros of model
print(rmse)
```

206.95469529097284

Figure 42: Checking MSE for flow, power & peak power *

The Mean Squared Error (MSE) is essential in this case, telling how close the regression line is to a set point. Usually, it is used to check how close forecasted readings are close to the actual values. The lower the MSE, the accurate values to the actual, but this dataset shows a massive difference for average flow, average power, and peak power. In other words, this will lead to incorrect values for forecasting the future. Also, the Squared function was used to remove the negative, which were observed in residual parts.

7.3.3.2 Evaluating KNN model

```
df = pd.read_csv('/content/sample_data/____.csv')
new_data={"data":np.array(df[["AVG_SUPPLY_TEMP_(F)","AVG_RETURN_TEMP_(F)","Delta_T_(F)"]],ndmin=2),
          "target":np.array(df[["AVG_FLOW_(GPM)"]])}
X_train,X_test,Y_train,Y_test=train_test_split(new_data["data"],new_data["target"],random_state=0)
kn=KNeighborsClassifier(n_neighbors=2)
kn.fit(X_train.astype(int),Y_train.astype(int))
print(" KNN Accuracy is : {:.2f} %".format(1-kn.score(X_train.astype(int),Y_train.astype(int))))
```

KNN Accuracy is : 94 %

Figure 43: KNN Accuracy

KNN algorithm showing a good score, compared to ARIMA, and it is showing 94% accuracy which help to predict the future values with high accuracy.

7.3.3.3 Evaluating SVM model

```
SVmodel = svm.SVC(kernel = "linear",gamma="auto",C=1000000)
SV_Model = SVmodel.fit(X_train.astype(int),Y_train.astype(int))
SV_Model
```

```
SVC(C=3, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
print("SVM Accuracy is : {:.2f} %".format(SVmodel.score(X_train.astype(int),Y_train.astype(int))))
```

SVM Accuracy is : 5 %

Figure 44: SVM accuracy

As mentioned above, SVM was not much accurate even if the C values changed up to 1 million; still the accuracy of the trained values to tested is 5% only, which is not accepted, and clearly, in Fig. 37,39, and 41 it indicated that the outliers were ignored during testing and training process.

(Note: even though, will prove the accuracy by predicting the future values and will try plot the outcome)

7.3.4 Predicting the future values

This section will forecast the reading for flow, power, and peak power for the ARIMA model, KNN, and SVM. Prediction values trends shows the maximum /minimum values to decide when the time for action is required by DCC.

7.3.4.1 ARIMA Predicted values

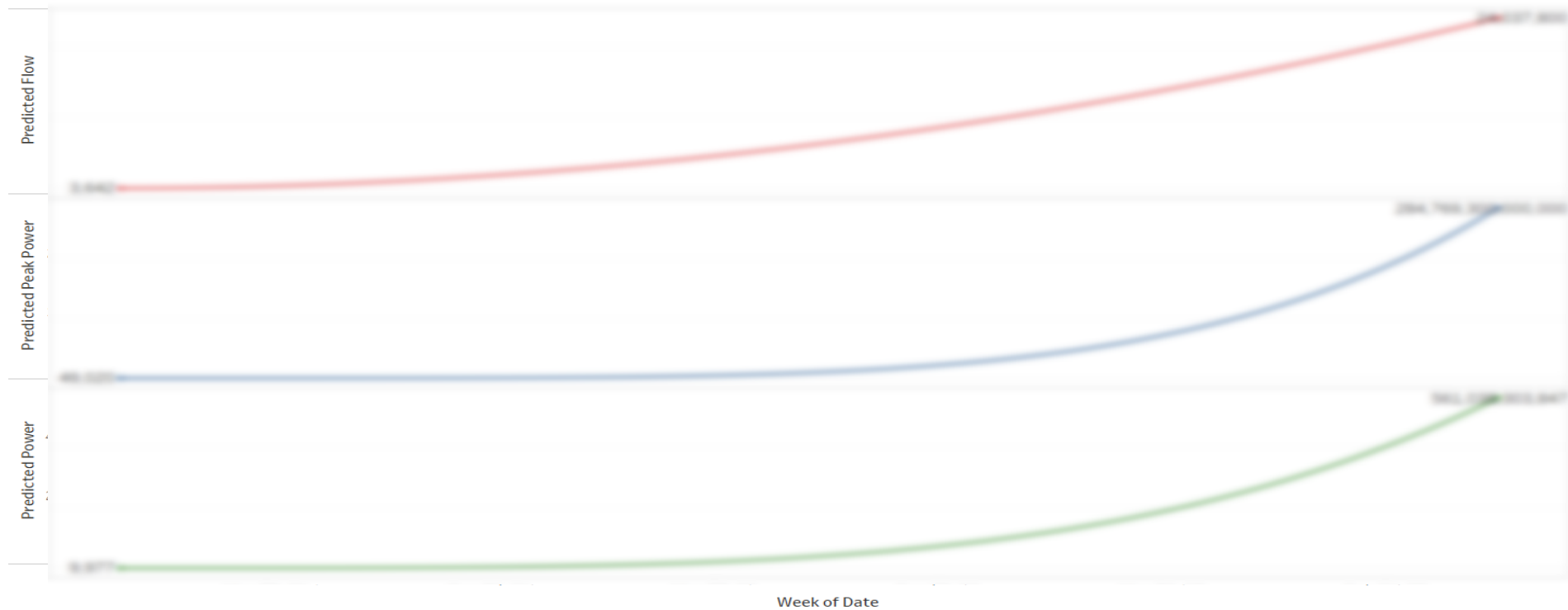


Figure 45: ARIMA Predicted Values graphs using tableau

The predicted values show some values that inconsistent in daily readings and not behaving similarly to original data. Hence, it extracted the data from python to CSV format, and plot it into Tableau for visualization.

The above figure is showing cumulative values, which should not be in this case, due to that the values are changing based on the season and demand load, but here it is showing the condition is the same, even if the data set trained and tested, the result not met the expectations at this level, but from MAE, it was indicating that the error is high. This might be a result of the dataset length and not sufficient for this tool. it noticed that also, some scaling issue reflected, in peak power, it is showing numbers with 15 digits, which is incorrect.

Trend Lines Model

A linear trend model is computed for sum of Predicted Flow given Date Week. The model may be significant at $p \leq 0.05$.

Model formula: (Week of Date + intercept)
Number of modeled observations: 161
Number of filtered observations: 0
Model degrees of freedom: 2
Residual degrees of freedom (DF): 159
SSE (sum squared error): 5.13028e+14
MSE (mean squared error): 3.22659e+12
R-Squared: 0.93863
Standard error: 1.79627e+06
p-value (significance): < 0.0001

A linear trend model is computed for sum of Predicted Peak Power given Date Week. The model may be significant at $p \leq 0.05$.

Model formula: (Week of Date + intercept)
Number of modeled observations: 161
Number of filtered observations: 0
Model degrees of freedom: 2
Residual degrees of freedom (DF): 159
SSE (sum squared error): 2.75786e+29
MSE (mean squared error): 1.7345e+27
R-Squared: 0.676947
Standard error: 4.16473e+13
p-value (significance): < 0.0001

A linear trend model is computed for sum of Predicted Power given Date Week. The model may be significant at $p \leq 0.05$.

Model formula: (Week of Date + intercept)
Number of modeled observations: 161
Number of filtered observations: 0
Model degrees of freedom: 2
Residual degrees of freedom (DF): 159
SSE (sum squared error): 9.13293e+23
MSE (mean squared error): 5.74398e+21
R-Squared: 0.753624
Standard error: 7.5789e+10
p-value (significance): < 0.0001

Figure 46: ARIMA Trends Line Model description from Tableau software

Moreover, the values of b , d & q are affecting the results. In this model, the Augmented Dickey-Fuller Test was recommending using order as $(p = 2, d = 1, q = 1)$, where p means the number of auto-regressive terms, d is the number of non-seasonal difference needed for stationary, and q is the number of lagged forecast errors in the prediction equation. The Dickey-Fuller test is checks the dataset intercept to see if there is any relationship between the parameters selected for testing, and accordingly, it finds the best order for p , d , and q values. Another factor is affecting this model is that the AIC, P-value, and number of observations and number of the lags.

7.3.4.2 KNN Predicted values

From the KNN evaluation section, it was found that the accuracy of this algorithm is about 94%, which is the best result compared to other tools and algorithms. To find the behavior of the predicted data, it is extracted from python to CSV format and plotted in Tableau software for comparison and visualization.

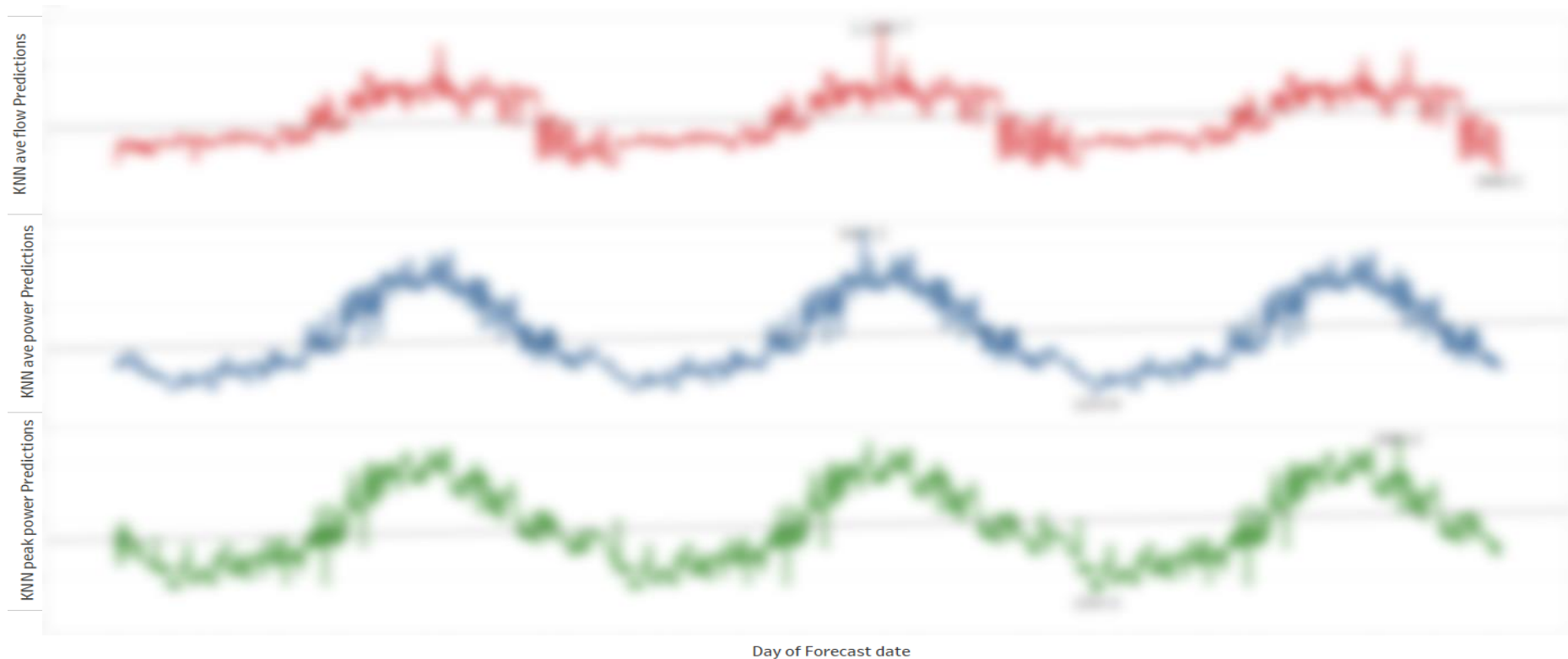


Figure 47: KNN predicted Values graphs using Tableau *

Comparing the original dataset trend to these trends are showing similar patterns, and from the Tableau, the retrieved data is showing the models degrees of freedom is 2, and MSE is 24334 with a standers error of 155.9, and P-value for these trends is 0.0001 (less than 0.05) which indicate it is significance.

Trend Lines Model

A linear trend model is computed for sum of KNN ave flow Predictions given Forecast date Day. The model may be significant at $p \leq 0.05$.

Model formula: (Day of Forecast date + intercept)
Number of modeled observations: 1089
Number of filtered observations: 0
Model degrees of freedom: 2
Residual degrees of freedom (DF): 1087
SSE (sum squared error): 2.64515e+07
MSE (mean squared error): 24334.4
R-Squared: 0.04171
Standard error: 155.995
p-value (significance): < 0.0001

A linear trend model is computed for sum of KNN ave power Predictions given Forecast date Day. The model may be significant at $p \leq 0.05$.

Model formula: (Day of Forecast date + intercept)
Number of modeled observations: 1089
Number of filtered observations: 0
Model degrees of freedom: 2
Residual degrees of freedom (DF): 1087
SSE (sum squared error): 1.62412e+07
MSE (mean squared error): 14941.3
R-Squared: 0.040292
Standard error: 122.235
p-value (significance): < 0.0001

A linear trend model is computed for sum of KNN peak power Predictions given Forecast date Day. The model may be significant at $p \leq 0.05$.

Model formula: (Day of Forecast date + intercept)
Number of modeled observations: 1089
Number of filtered observations: 0
Model degrees of freedom: 2
Residual degrees of freedom (DF): 1087
SSE (sum squared error): 1.90228e+07
MSE (mean squared error): 17500.3
R-Squared: 0.0437003
Standard error: 132.289
p-value (significance): < 0.0001

Figure 48: KNN Trends Line Model description from Tableau software

7.3.4.3 SVM Predicted values

The SVM evaluation section showed that this model's accuracy is 5%, which is not recommended to be taken in decision considerations. But, to get more understanding, will be plotting the graphs of these model.

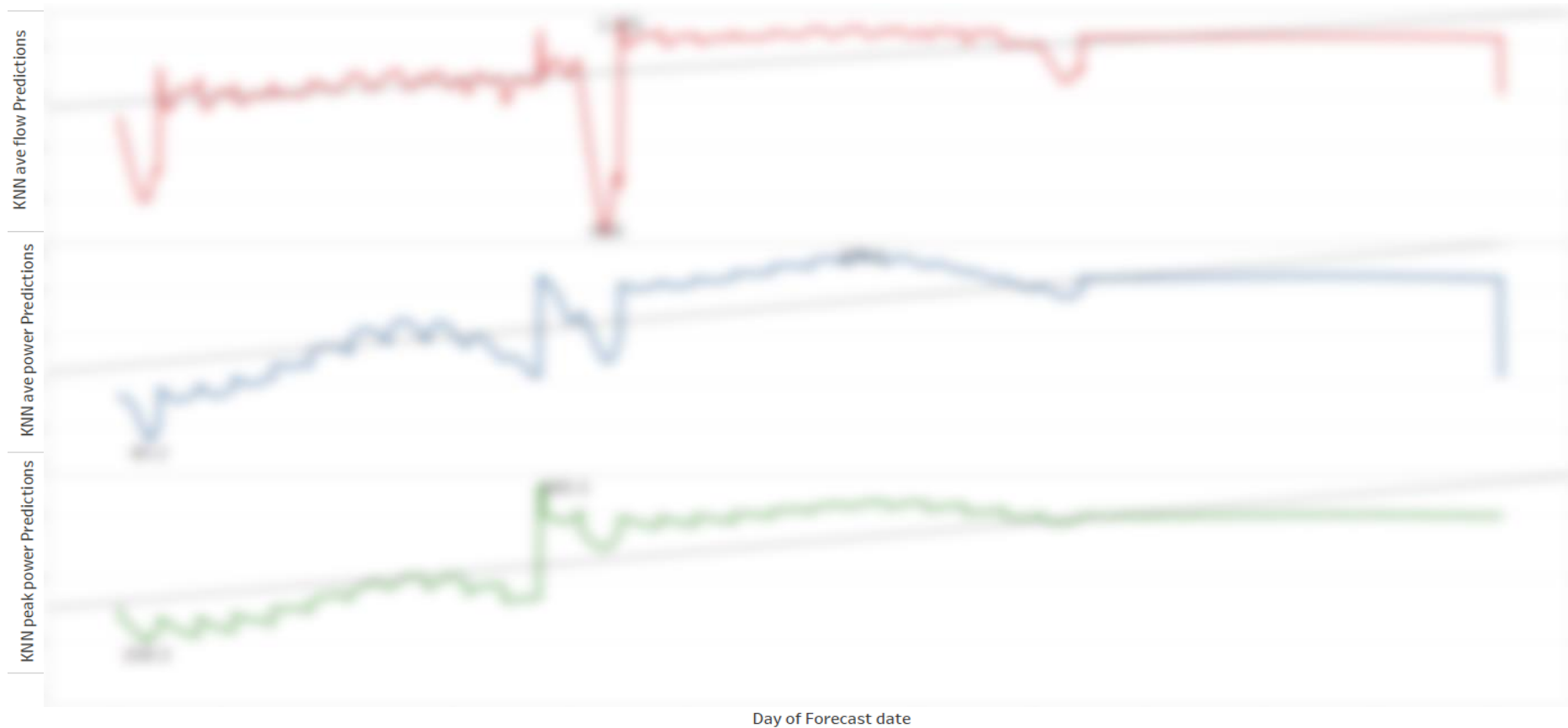


Figure 49: SVM predicted Values graphs using Tableau

Trend Lines Model

A linear trend model is computed for sum of KNN ave flow Predictions given Forecast date Day. The model may be significant at $p \leq 0.05$.

Model formula: (Day of Forecast date + intercept)
Number of modeled observations: 1095
Number of filtered observations: 0
Model degrees of freedom: 2
Residual degrees of freedom (DF): 1093
SSE (sum squared error): 8.47615e+07
MSE (mean squared error): 77549.4
R-Squared: 0.440729
Standard error: 278.477
p-value (significance): < 0.0001

A linear trend model is computed for sum of KNN ave power Predictions given Forecast date Day. The model may be significant at $p \leq 0.05$.

Model formula: (Day of Forecast date + intercept)
Number of modeled observations: 1095
Number of filtered observations: 0
Model degrees of freedom: 2
Residual degrees of freedom (DF): 1093
SSE (sum squared error): 3.78613e+06
MSE (mean squared error): 3463.98
R-Squared: 0.630203
Standard error: 58.8556
p-value (significance): < 0.0001

A linear trend model is computed for sum of KNN peak power Predictions given Forecast date Day. The model may be significant at $p \leq 0.05$.

Model formula: (Day of Forecast date + intercept)
Number of modeled observations: 1095
Number of filtered observations: 0
Model degrees of freedom: 2
Residual degrees of freedom (DF): 1093
SSE (sum squared error): 6.55577e+06
MSE (mean squared error): 5997.96
R-Squared: 0.663505
Standard error: 77.4465
p-value (significance): < 0.0001

Figure 50: SVM Trends Line Model description from Tableau software

In Figure 47, the trend of flow shows negative values, which is totally incorrect, and it is not looking like seasonal data. Overall, the SVM algorithm seems to be illogical, as it is showing stable values starting from certain time onwards. This might be due to the weakness of the training process in this algorithm. Also, the MSE, as shown in the earlier stage, it is very high and impacted in the predicted values, and it is proved that this algorithm can be used and give better results if it is used as a classifier based on the dataset type.

7.3.5 Predicted values evaluation

Based on the above section, three prediction types were used, and different values were predicted based. Each algorithm has its way of dealing with data, and this also depended on the data length and data type. In this report, ARIMA, KNN, and SVM were selected due to supervised learning and regression analysis, where this can help identify which variables have an impact/intercept on a specific interest. During the process of performing a regression, it helps the user to determine which factors matter most and which factors can be ignored.

The ARIMA model showed that testing values were typically compared to the original dataset, it could not find seasonal patterns, so it predicts the mean values (straight line). In other words, the data set is not stationary, and it varies each day; even though that the module predicted the best order for p , d & q , the model still not functioning well. To improve the model, either to work to improve coefficient estimation like maximum estimation likelihood, least-square. Alternatively, work to improve model & Residual Diagnostic such as AIC, BIC.

KNN showed the best accuracy during the testing/ training stage and found that the accuracy was 94%, which much close to the original data. Even predicted values showed that the trends were a similar pattern, indicating the confidence of the values. The reason behind this is the mechanism of this algorithm and how it is looking for the link between each point. To improve this model's accuracy can be by doing some tuning hyperparameters, which means these parameters control the learning process and efficiency during the training stage of machine learning. It basically determines how the algorithm is going to different learning approaches in different steps of its process. Additionally, Robust scaling can enhance the algorithm's performance, which means using scaling to overcome the presence of outliers in the dataset, which rescales the feature using the median and quartile range.

SVM was also used in this project, and it was challenging to select the C values; trial and error were used to help the training dataset to improve. In general, SVM takes data points to draw a straight line between two classes, all the data points that fall on one side will be labeled as one class, and other points laid on the other side will be considered as second class. It is usually a better idea to apply the ensemble method to improve the model's accuracy by using boosting or bagging. Boosting means a technique used to adjust the weight of an observation based on the loss classification, which will decrease the bias error and develop strong predictive models.

8. Analysis Results

Three different models were developed and found that the accuracy percentage not matching to each other. The reasons might be due to the mechanism of the algorithm and its functionality. So, it decided to ignore the ARIMA model and SVM due to incorrect prediction values and will consider the KNN model as a validation reference.

For comparison and check how accurate predicted values to actual values, requested additional data for 1 month (actual) and compared it to predicted Flow, Power, and Peak Power values. Figure 51 shows that the pattern is close to both readings, with little difference. For the Flow, it is shown that almost all the readings are the same on some days. However, in Power and Peak Power not that close, but within the same range tolerance.

This project aims to determine and predict the load demand to a particular customer, and this will help DCC to serve the client in a better way to avoid any complaint regarding flow or power demand. The main goal is achieved and found that the highest flow required, and the highest average power also will occur in the same day as it shown in figure 47. This means that DCC needs to modify/upgrade the existing equipment in terms of pumps and chillers, where the above numbers exceed the plant-designed flow and refrigerant tonnage.

KNN Vs Real Values



Figure 51: Predicted Vs. Actual comparison *

9. Conclusions and Future Work

Years ago, decision-making used to be taken from experience, thoughts, lessons learned from incidents, or assumptions. The data analysis world is growing swiftly, and it will be the next decision-maker factor and will have a considerable value of money for organizations and countries. Time is money for business markets, and data analytics is a security-enhancing tool of the future, who have the data he can sustain in his business, else money will be spent more.

This project targeted the time series forecasting based on the historized dataset applied three models, where ARIMA was not successfully given the results, so went to the next model, which was KNN, and some improvement was found. Lastly, SVM was also applied to check which is the best model is the best and proved that KNN is the best algorithm for this dataset, which can be implemented in DCC decision-making system to support management during their yearly plants' upgrading meetings. Nevertheless, SVM & ARIMA can be improved more by applying the techniques mentioned in section 7.3.5.

The next target for this project is to look to the possibility to configure this model or any other similar model into DCC system, where live data are coming into control room like SCADA system, this will enhance the behavior of the model, and more training and testing will take place. If such a system is implemented, it is assured that DCC is the first organization who implement this technology in the district cooling sector.

Proudly, to be the first person in DCC how worked in such project and data analytics, and more confident to share the findings with DCC management for their approval for implementation, and it was planned earlier to participate in IDEA (International District Energy Association) conferences and to share the value of the data and how it can help your business for decision making. Due to confidentiality and DCC policy, it is not allowed to share the results with any parties without official approval from DCC side.

10. Bibliography

- Baru, C., Bhandarkar, M., & Nambiar, R. (2018). A Predictive Map Task Scheduler for Optimizing Data Locality in MapReduce Clusters. *Int. J. Grid High Perform*, 10(4), 122-138.
- Boussaada, Z., Curea, O., Remaci, A., Camblong, H., & Bellaaj, N. (2018, 11). A Nonlinear Autoregressive Exogenous. *Neural Network Model for the Prediction of the Daily Direct Solar Radiation*, 620-629.
- Candelieri, A. (2017). *Clustering and Support Vector Regression for water demand forecasting and anomaly* (Vol. 9). New York.
- Cunningham, P., & Delany, S. J. (2007). *k-Nearest neighbour classifiers*. University College Dublin. Dublin: Dublin Institute of Technology.
- Electrical and Mechanical Services Department (EMSD). (2012). *Implementation of district cooling system at SEKD*. Hong Kong: Legislative Council Panel on Environmental Affairs.
- Escriva'-Escriva', G., A'lvarez-Bel, C., Rolda'n-Blay, C., & Ica'zar-Ortega, M. (2011). New artificial neural network prediction method for electrical consumption forecasting based on building end-uses. *Energy Build*, 43 - 45.
- Farias, R. L., Puig, V., Rangel, H. R., & Flores, J. J. (2018, March 15). Multi-Model Prediction for Demand Forecast in Water Distribution Networks. *Energies*, 4-6.
- Francisco, Zamora-Mart'inez, Romeu, P., & Pardo, J. (2013). Towards Energy Efficiency: Forecasting Indoor Temperature via Multivariate Analysis. *Snergies*, 18 - 21.
- HO, P. Y. (2003). *Weathering the Storm: Hong Kong Observatory and Social*. Hong Kong: Hong Kong University Press,.
- Jain, A., & Ormsbee, L. (2002). *Short-term flow demand forecast modeling techniques - conventional methods versus AI*. Oakland: Hydrologic Engineering.
- Jang, J., & Leigh, S. (2017, Apr). A Prediction of Optimal Heating Timing based on Artificial Neural Network by. 37, 563-564.
- Liu, Z., Wu, D., Liu, Y., Han, Z., Lun, L., Gao, J., . . . Cao, G. (2019, January 28). Accuracy analyses and model comparison of machine learning adopted in building energy consumption prediction. *Energy Exploration & Exploitation*, 37(4). doi:10.1177/0144598718822400
- Murat, M., Malinowska, I., Gos, M., & Krzyszczak, J. (2018). *Forecasting daily meteorological time series using ARIMA and regression models*. 2Institute of Agrophysics, Polish Academy of Sciences. Lublin, Poland: Internaltional Agrophysics. doi:10.1515/intag-2017-0007
- Quevedo, V., & Puig, J. (2010). Validation and reconstruction of flow meter data in the Barcelona water distribution network. *Control Engineering*, 640-651.

Appendix A

Table 1: Data set attributes description	17
Figure 1: CRISP DM process	14
Figure 2: Project Analysis Steps used in Python	16
Figure 3: Uploading the data set and show it *.	17
Figure 4: Dataset attributes count.....	18
Figure 5: Checking the attributes types	18
Figure 6: Missing data per attribute	19
Figure 7: Installed all Libraries	20
Figure 8: Data Discription	21
Figure 9: Importing plotting library & build the code.....	21
Figure 10: Average Flow, Average Power & Peak Power graphs *	22
Figure 11: Dividing the dataset into training and testing	24
Figure 12: Checking for x & y values *	24
Figure 13: storing the new values into their locations	25
Figure 15: Agumented statistical Testing with respect to Flow, Power & Peak Power *	25
Figure 14: Augmented Data Fuller test and analysis	25
Figure 16: Fitting ARIMA model with respect to average flow.....	27
Figure 17: ARIMA average flow details.....	27
Figure 18: Residual Statistics details (for Average Flow)*	28
Figure 19: Testing details (for Average Flow) *	28
Figure 20: Fitting ARIMA model with respect to average power	29
Figure 21: ARIMA average power details.....	29
Figure 22: Residual Statistics details (for Average Power) *	30
Figure 23: Testing details (for Average Power) *	30
Figure 24: Fitting ARIMA model with respect to peak power.....	31
Figure 25: ARIMA model details for peak power.....	31
Figure 26: Residual Statistics details (for peak power)*	32
Figure 27: Testing details (for peak Power) *	32
Figure 28: KNN classification.....	33
Figure 29: KNN testing/training output for Average Flow	34
Figure 30: KNN Algorithm training/ testing with respect to Average Flow	34
Figure 31: KNN Algorithm training/ testing with respect to Average Power	35
Figure 32: KNN testing/training output for Average Power	35
Figure 34: KNN testing/training output for Peak Power.....	36
Figure 33: KNN Algorithm training/ testing with respect to Peak Power	36
Figure 35: Support Vector Machines Classification	37
Figure 36: SVM Algorithm training/ testing with respect to Average Flow	38
Figure 37: SVM testing/training output for Average Flow.....	38
Figure 38: SVM Algorithm training/ testing with respect to Average Power	39
Figure 39: SVM testing/training output for Average Power	39
Figure 40: SVM Algorithm training/ testing with respect to Peak Power.....	40
Figure 41: SVM testing/training output for Peak Power	40
Figure 42: Checking MSE for flow, power & peak power	41
Figure 43: KNN Accuracy.....	42
Figure 44: SVM accuracy.....	42

Figure 45: ARIMA Predicted Values graphs using tableau	43
Figure 46: ARIMA Trends Line Model description from Tableau software	44
Figure 47: KNN predicted Values graphs using Tableau *	46
Figure 48: KNN Trends Line Model description from Tableau software	47
Figure 49: SVM predicted Values graphs using Tableau	48
Figure 50: SVM Trends Line Model description from Tableau software	49
Figure 51: Predicted Vs. Actual comparison *	52