

# Configuration Manual

Veera Avinash Chowdary Pusuluri  
Student ID: x22162402

## 1. Introduction

The documentation describes how the implementation code of this research project is to be run and configured. Specific information on the computer's hardware, as well as programs that are to be started, is included in this document. Users can generate summary summaries from research paper by following the procedures set out below.

## 2. System Specification

### 2.1 Hardware specification

Following are the hardware specifications of the system that was used to develop the project:

**Processor:** Mac OS M1-Chip

**Ram:** 6-GB

**Storage:** 128-GB

**Graphic card:** 8-GB

**Operating system:** Mac OS

### 2.2 Software Specification

The Google Collab is a web-based platform was used to train and evaluate the models and its specification was the following:

**Processor:** Mac OS

**Graphic card:** 8-GB

**RAM:** 8-GB

**Storage:** 128-GB

## 3. Software Tools

Following are the software tools that were used to implement the project:

### 3.1 Python

The project was created using the Python programming language. Python was chosen mostly because of its excellent packages for visualization and dataset preparation. Python was obtained from the official website. Figure 1 depicts the Python official website's download page.



Fig: 1 Python official website image  
Source: <https://www.python.org/downloads/>

### 3.2 Google collab

Google Collab, short for Google Colaboratory, is a Google cloud-based platform that enables users to write and execute Python code in a collaborative and interactive environment. Fig.2 illustrates google collab:

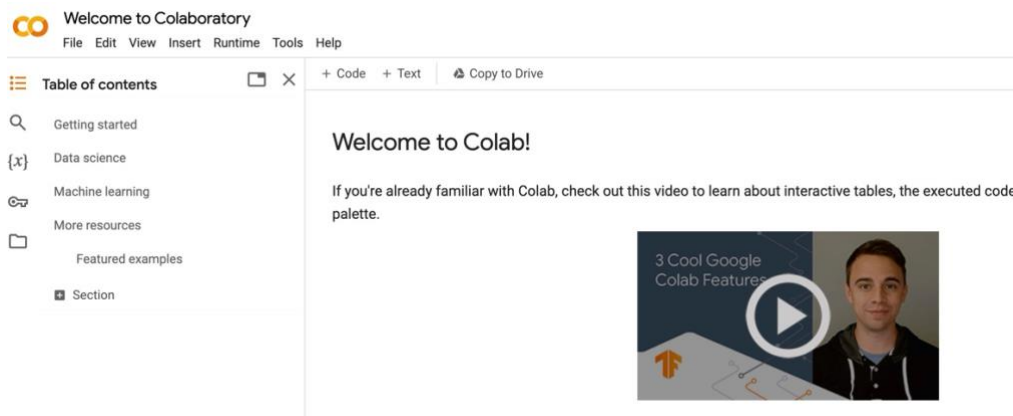



Fig: 2 It shows google collab official page.  
Source: <https://colab.research.google.com/>

## 4. Project Implementation

Following are the Python packages which were installed and used to implement the project:

- Pandas
- Numpy
- Missingno
- Plotly.express
- Datasets



```
import pandas as pd
import numpy as np
```

Fig: 3 Necessary libraries for the project  
Source: Self-Created

Pandas library was used to load and check the dataset as can be seen in Figure 4:


```
dset= pd.read_csv("PowerSupplyPosition_2012-05-02_to_2023-05-14.csv", encoding='iso-8859-1')
dset.head()
```

	Date	Energy Required (MU)	Energy Met (MU)	Energy +/- (MU)	Genco Thermal	Genco Hydel	Genco Total	CGS and Purchases	IPPS (GAS)	NCEs & Others	AP Share of TGISTS	Grand Total	Reversible Pump Consumption	Unrestricted Peak Demand (MW)	Deficit/Surplus (MW)
0	02- May- 2012	255.639	241.185	-14.454	103.643	5.276	108.919	77.106	42.752	12.408	0.0	241.185	0.0	12099	-1000.0
1	03- May- 2012	258.470	243.370	-15.100	106.255	3.748	110.003	79.273	41.374	12.720	0.0	243.370	0.0	12219	-1500.0
2	04- May- 2012	261.393	247.449	-13.944	106.153	6.527	112.680	82.753	39.385	12.631	0.0	247.449	0.0	11693	-1000.0
3	05- May- 2012	252.866	237.919	-14.947	95.295	5.334	100.629	85.987	39.256	12.047	0.0	237.919	0.0	11636	-1000.0
4	06- May- 2012	250.566	236.528	-14.038	95.862	4.494	100.356	86.762	38.017	11.393	0.0	236.528	0.0	11133	-700.0

Fig: 4 Loading and checking the dataset  
Source: Self-Created

Fig: 5 shows the information of the dataset.

 `dset.info()`

 `<class 'pandas.core.frame.DataFrame'>`  
RangeIndex: 3943 entries, 0 to 3942  
Data columns (total 15 columns):

#	Column	Non-Null Count	Dtype
0	Date	3941 non-null	object
1	Energy Required (MU)	3940 non-null	float64
2	Energy Met (MU)	3940 non-null	float64
3	Energy +/- (MU)	3940 non-null	float64
4	Genco Thermal	3940 non-null	float64
5	Genco Hydel	3940 non-null	float64
6	Genco Total	3940 non-null	float64
7	CGS and Purchases	3940 non-null	float64
8	IPPS (GAS)	3940 non-null	float64
9	NCEs & Others	3940 non-null	float64
10	AP Share of TGISTS	3940 non-null	float64
11	Grand Total	3940 non-null	float64
12	Reversible Pump Consumption	3940 non-null	float64
13	Unrestricted Peak Demand (MW)	3940 non-null	object
14	Deficit_Surplus (MW)	3939 non-null	float64

dtypes: float64(13), object(2)  
memory usage: 462.2+ KB

Fig: 5 Shows the information of the dataset.

**Source: Self-Created**

```

✓ 0s ▶ dset.isnull().sum()
🔗 Date 0
Energy Required (MU) 0
Energy Met (MU) 0
Energy +/- (MU) 0
Genco Thermal 0
Genco Hydel 0
Genco Total 0
CGS and Purchases 0
IPPS (GAS) 0
NCEs & Others 0
AP Share of TGISTS 0
Grand Total 0
Reversible Pump Consumption 0
Unrestricted Peak Demand (MW) 0
Deficit_Surplus (MW) 0
dtype: int64

```

Fig: 6 shows that dataset is cleared from null values.

Source: Self-Created

```

✓ [67] dset.describe()

```

	Energy Required (MU)	Energy Met (MU)	Energy +/- (MU)	Genco Thermal	Genco Hydel	Genco Total	CGS and Purchases	IPPS (GAS)	NCEs & Others	AP Share of TGISTS	Grand Total	Reversible Pump Consumption	Deficit_Sur plus (MW)
count	3939.000000	3939.000000	3939.000000	3939.000000	3939.000000	3939.000000	3939.000000	3939.000000	3939.000000	3939.000000	3939.000000	3939.000000	3939.00
mean	185.799824	179.544922	-6.254065	66.100691	10.181949	76.282640	53.272091	13.220817	43.748356	-6.933342	179.590562	0.045640	-277.39
std	47.048485	36.877052	15.473587	18.785745	8.154817	20.373629	30.440018	7.650374	31.453857	9.824229	36.960165	0.467394	683.80
min	0.000000	0.000000	-85.885000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	-67.202000	0.000000	0.000000	-4000.00
25%	151.784500	151.701000	0.000000	53.087500	5.746500	62.193000	31.414500	7.754500	13.162000	-14.766000	151.701000	0.000000	0.00
50%	171.603000	171.603000	0.000000	64.492000	7.710000	74.549000	42.131000	12.867000	44.227000	-1.678000	171.603000	0.000000	0.00
75%	207.976500	205.059500	0.000000	78.893500	11.626500	88.495500	71.292500	17.983500	69.859000	-0.403000	205.059500	0.000000	0.00
max	320.284000	284.778000	1.578000	109.729000	69.030000	150.161000	146.229000	45.683000	124.395000	1.873000	284.778000	9.850000	500.00

Fig: 7 Describing the dataset.

Source: Self-Created

```

✓ 0s ▶ categorical_values = []
for i in dset.columns:
    if dset[i].dtype == "object":
        categorical_values.append(i)

print("Categorical columns :", categorical_values)

🔗 Categorical columns : ['Date', 'Unrestricted Peak Demand (MW)']

```

Fig: 8 Finding the categorical columns

Source: Self-Created

```

✓ [84] #feature selection using stats Model
0s import statsmodels.api as sm
    X= sm.add_constant(X)

✓ [85] lr = sm.OLS(Y, X).fit()
0s print(lr.summary2())

```

Results: Ordinary least squares

Model:	OLS	Adj. R-squared:	0.979
Dependent Variable:	Deficit_Surplus (MW)	AIC:	47486.7240
Date:	2023-12-13 22:34	BIC:	47562.0682
No. Observations:	3939	Log-Likelihood:	-23731.
Df Model:	11	F-statistic:	1.631e+04
Df Residuals:	3927	Prob (F-statistic):	0.00
R-squared:	0.979	Scale:	10043.

	Coef.	Std.Err.	t	P> t	[0.025	0.975]

Fig: 9 Feature selection using stats model  
Source: Self-Created

```

✓ [90] from sklearn.preprocessing import MinMaxScaler
0s sc= MinMaxScaler()
    X_train= sc.fit_transform(X_train)
    X_test= sc.transform(X_test)

```

Fig: 10 Normalising the data  
Source: Self-Created

## 5. Machine learning algorithms used in this project.

### 5.1 Decision tree classifier

The decision tree classifier is a popular machine learning technique that is noted for its interpretability, adaptability, and simplicity of use. One of its key benefits is that it is

transparent and straightforward, making it accessible to people who may not have a thorough understanding of machine learning.

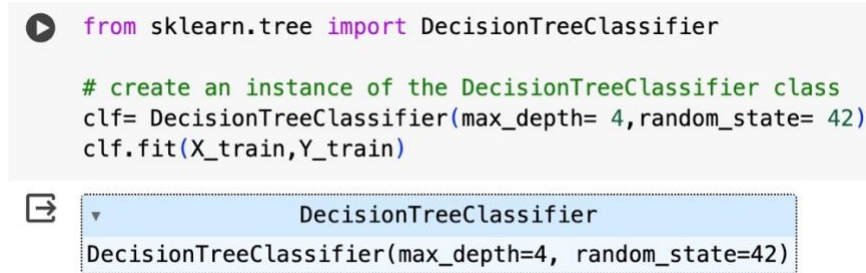


Fig: 11 Importing decision tree classifier  
Source: Self-Created

Evaluating the results of model-1 (Decision tree classifier):

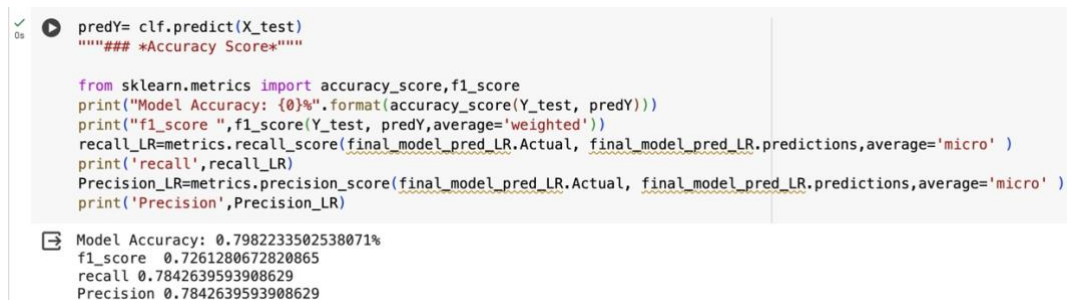


Fig: 12 It shows the results evaluated for model-1 Decision Tree Classifier  
Source: Self-Created

## 5.2 Random forest classifier

The Random Forest classifier is a well-known machine learning method that is well-known for its robust performance and versatility. It is an ensemble method that generates the mode of the classes (classification) or the average prediction (regression) of the individual trees during training.

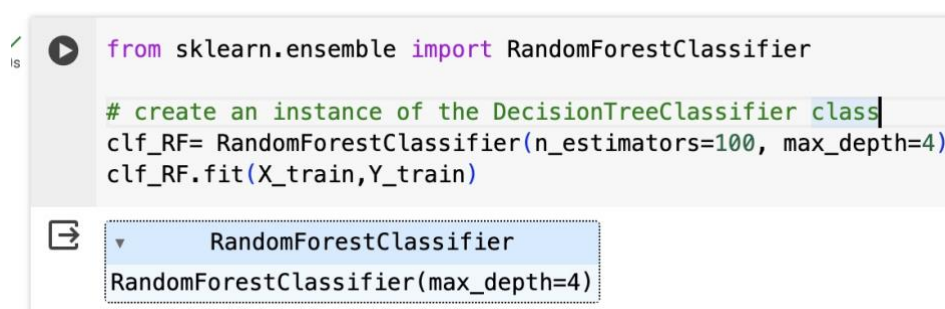




Fig: 12 Importing Random Forest classifier  
Source: Self-Created

Evaluating the results of model-2 (Random Forest Classifier):

```
[96] predY= clf_RF.predict(X_test)
      """### *Accuracy Score*"""

      from sklearn.metrics import accuracy_score,f1_score
      print("Model Accuracy: {0}%".format(accuracy_score(Y_test, predY)))
      print("f1_score ",f1_score(Y_test, predY,average='weighted'))
      recall_LR=metrics.recall_score(final_model_pred_LR.Actual, final_model_pred_LR.predictions,average='micro' )
      print('recall',recall_LR)
      Precision_LR=metrics.precision_score(final_model_pred_LR.Actual, final_model_pred_LR.predictions,average='micro' )
      print('Precision',Precision_LR)

Model Accuracy: 0.7969543147208121%
f1_score 0.7284329729465212
recall 0.7842639593908629
Precision 0.7842639593908629
```

Fig: 13 Results evaluation of a model-2 (Random Forest Classifier)  
Source: Self-Created

### 5.3 Logistic regression

Logistic regression is a popular statistical method in machine learning and statistics, especially when the outcome variable is binary or categorical. The main strength of logistic regression is its capacity to represent the likelihood of an event occurring, which makes it well-suited for tasks like binary categorization.

```
from sklearn.linear_model import LogisticRegression
logmodel= LogisticRegression()
logmodel.fit(X_train, Y_train)
```

▼ LogisticRegression  
LogisticRegression()

Fig: 13 Importing logistic regression  
Source: Self-Created

Evaluating the results of model-3 (Logistic Regression):

```
predY= clf_RF.predict(X_test)
"""### *Accuracy Score*"""

from sklearn.metrics import accuracy_score,f1_score
print("Model Accuracy: {0}%".format(accuracy_score(Y_test, predY)))
print("f1_score ",f1_score(Y_test, predY,average='weighted'))
recall_LR=metrics.recall_score(final_model_pred_LR.Actual, final_model_pred_LR.predictions,average='micro' )
print('recall',recall_LR)
Precision_LR=metrics.precision_score(final_model_pred_LR.Actual, final_model_pred_LR.predictions,average='micro' )
print('Precision',Precision_LR)

Model Accuracy: 0.7969543147208121%
f1_score 0.7284329729465212
recall 0.7842639593908629
Precision 0.7842639593908629
```



Fig: 14 Results evaluation of a model-3 (Logistic regression)  
**Source: Self-Created**