National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Vaibhav Subhash Sawant
Student ID: x19200706

School of Computing
National College of Ireland

Supervisor: Martin Alain

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Vaibhav Subhash Sawant
Student ID:	x19200706
Programme:	Msc Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Martin Alain
Submission Due Date:	16/12/2021
Project Title:	Prediction of Customer Lifetime Value and Fraud Detection in BFSI using Machine Learning
Word Count:	1106
Page Count:	18

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	31st January 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Vaibhav Subhash Sawant
x19200706

1 Introduction

1.1 Project Brief

The research was carried out to answer two research questions related to the Banking financial services and Insurance Domain, with the help of different machine learning models. The final objectives of the research were successfully implemented, and the outcomes of the models are presented in the research report.

1.2 Purpose of Configuration Manual

The document will provide the guidelines for implementing the technical part of the research. The document explained the step-by-step approach to make the end-user understand each step clearly. This approach will help the end-user recreate the same results achieved during the research using code snippets. The general information section explains all the technical requirements and platform-related information.

2 General Information

2.1 System Information

- **Operating System:** Microsoft Windows 10 Home
- **Version:** 10.0.19044 Build 19044
- **Processor:** Intel(R) Core (TM) i7-9750H CPU @ 2.60GHz, 2592 MHz, 6 Core(s), 12 Logical Processor(s)
- **Installed Physical Memory (RAM):** 16.0 GB
- **Hard Disk:** 500GB

2.2 Platforms

The programming language applied in the implementation of the project is python. Platform used for the implementation of the research project as mentioned below.

- **Python (3.8.8)**
  The version of python used in the research project is 3.8.8. The link for the python setup is given below.

  Link: `https://www.python.org/downloads/`

- **Anaconda**
  Anaconda is a platform that provides a web-based platform for the development of projects and software using python. Jupyter is one such platform available in the Anaconda environment. The link and guide to setting setup an anaconda environment is given below.

  Link: `https://docs.anaconda.com/anaconda/install/windows/`

- **Jupyter Notebook**
  It is an open-source platform developed with the help of anaconda studio to develop and execute python projects.

- **Google Colaboratory**
  Google colab is easy to use platform for python project development. The environment provides functionality to code the projects in python with the help of GPU and TPU, which helps to faster execution of the programs.

2.3 Datasets Information

Two datasets were used in research for two different research questions. The links for the datasets are provided below.

- **Dataset 01:** `https://tinyurl.com/2p99n3mn`
- **Dataset 02:** `https://tinyurl.com/2p8ejvtc`

2.4 Libraries

Different python libraries were used in the project implementation to process data, modeling, evaluation, data visualization, and hyperparameter tuning.

- **Data frame and Data Processing:** NumPy, Pandas.
- **Data Visualization:** matplotlib, seaborn.
- **Dataset Splitting (Test and Train):** Sklearn - train test split.
- **Modelling:** Sklearn - SVC, BernoulliNB, BaggingClassifier, AdaBoostClassifier.
- **Evaluation:** Sklearn - classification report, accuracy score, confusion matrix, roc auc score, roc curve.

3 Deployment or Implementation - Fraud Detection

Follow all the steps mentioned below to execute the code files successfully.

3.1 Uploading of Python Files and Datasets - Only When Using Google Colaboratory

Execute this piece of code only when you are using Google Colab

```
[ ]
    from google.colab import drive
    drive.mount('/content/gdrive')

    Mounted at /content/gdrive
```

3.2 Step 1: Importing Required Python Libraries

All the required libraries must be loaded initially before starting the execution of the code file without error.

```
[ ] import pandas as pd
    import numpy as np

    from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
    from sklearn.metrics import roc_auc_score,roc_curve
    from sklearn.model_selection import train_test_split

    from matplotlib import pyplot as plt
    import seaborn as sns
```

3.3 Step 2: Built and Run all the Pre-Created Function

All the functions are already developed to carry out the various task. We need to initialize and run those functions before loading the data. All the functions are mentioned in section 3.4 and 3.5.

3.4 Step 3: Loading the Required Data

Using load data function in first figure below, load data from .csv file to Data Frame. Be careful while providing the link for the Dataset.

```
[ ]  def Load_Data(Path):

         Df = pd.read_csv(Path)

         return Df
```

```
[ ]
   ## Importing the the Dataset into Dataframe

   Path = '/content/gdrive/My Drive/Research Project :: Msc Data Analytics/Datasets/Fraud detection/Training Data.csv'


   Df = Load_Data(Path)
```

3.5 Step 4: Data Pre-Processing

Step by step run code snippets for the below-mentioned steps.

3.5.1 Exploring Data Insights

Initially run the function Data Insights and then the piece of code in the second image.

```
[ ] def Data_Insights(Dataframe):

        print('**********************************************First Five Rows of Data**********************************************')
        print(Dataframe.head())

        print('**********************************************Last Five Rows of Data**********************************************')
        print(Dataframe.tail())

        print('**********************************************Dataframe Columns Details**********************************************')
        print(Dataframe.info())


        print('**********************************************Dataframe statistical Data**********************************************')
        print(Dataframe.describe())
```

```
[ ] Data_Insights(Df)

**********************************************First Five Rows of Data**********************************************
          Id   income  age  ...  current_job_years  current_house_years  risk_flag
155777  155778  6382731   51  ...                  5                   11          0
120399  120400  3932375   51  ...                  8                   12          0
76086    76087  4070923   23  ...                  6                   11          0
144346  144347  2111682   31  ...                  9                   14          0
137852  137853  2449147   42  ...                  3                   12          0

[5 rows x 13 columns]
**********************************************Last Five Rows of Data**********************************************
          Id   income  age  ...  current_job_years  current_house_years  risk_flag
251973  251974  1244622   35  ...                  3                   11          1
251977  251978  1330613   63  ...                 13                   12          1
251981  251982  1796713   47  ...                  2                   12          1
251982  251983  3182290   52  ...                  2                   10          1
251993  251994  8141027   60  ...                  9                   13          1
```

3.5.2 Null values Check

Run both piece of codes to check the null values.

```
[ ]  # Exploratory Data Analysis :: Exploring or Detecting the Null Values in the Dataframe

     Df.isnull().sum()

     Id                   0
     income               0
     age                  0
     experience           0
     married              0
     house_ownership      0
     car_ownership        0
     profession           0
     city                 0
     state                0
     current_job_years    0
     current_house_years  0
     risk_flag            0
     dtype: int64

[ ]  # Exploratory Data Analysis :: Exploring or Detecting the Null Values in the Dataframe

     Df.isna().sum()

     Id                   0
     income               0
     age                  0
     experience           0
     married              0
```

3.5.3 Duplicate values Check and Deleting Unwanted Columns

Here in the first image the data delete function is shown first run this function and then run snippet of codes in below images.

```
[ ]  def Data_Delete(Dataframe, columns):

         Dataframe.drop(columns, axis=1, inplace=True)
         return Dataframe
```

```
[ ] # Exploratory Data Analysis :: Exploring or Detecting the Duplicates values in the Dataframe
    duplicateRowsDf = Df[Df.duplicated()]

    print(duplicateRowsDf)
```

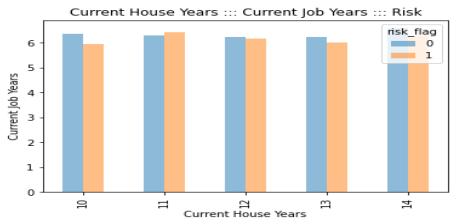
```
Empty DataFrame
Columns: [Id, income, age, experience, married, house_ownership, car_ownership, profession, city, state, current_job_years, current_house_years, risk_flag]
Index: []
```

```
[ ] # Exploratory Data Analysis :: Dropping the unwanted columns of the Dataframe

    columns =['Id','profession','city']
    Data_Delete(Df, columns)
```

3.5.4 Data Vizualization

Run all the snippets of data visualization from I to III.

```
# Data Visualization - II

df_pivot = pd.pivot_table(Df, values="current_job_years",index="current_house_years",columns="risk_flag", aggfunc=np.mean)
#Creating a grouped bar chart
ax = df_pivot.plot(kind="bar",alpha=0.5)
#Adding the aesthetics
plt.title('Current House Years ::: Current Job Years ::: Risk')
plt.xlabel('Current House Years')
plt.ylabel('Current Job Years')
# Show the plot
plt.show()
```




```
[ ]  # Data Visualization - III

     #Creating boxplots
     ax = sns.boxplot(x="age", y="house_ownership", hue="car_ownership", data=Df, palette="Set2")
     #Adding the aesthetics
     plt.title('Age ::: House ownership ::: Car ownership')
     plt.xlabel('Age')
     plt.ylabel('House ownership')
     # Show the plot
     plt.show()
```

3.5.5 Imbalanced Dataset treatment

The code is related to dealing with the imbalanced datset. Run all the code snippets steps for treatment of Imbalanced Dataset

```
[ ]  # Treatment of Imbalanced Dataset :: Step 01 :: Count of different classes present in the dependent Variable


     Df_COunt0, Df_COunt1 = Df.risk_flag.value_counts()

     Df_COunt0, Df_COunt1

     (221004, 30996)

[ ]  # Treatment of Imbalanced Dataset :: Step 02 :: classes wise creation of seperate Dataframes

     DF_RISk0 = Df[Df['risk_flag'] == 0]
     DF_RISk1 = Df[Df['risk_flag'] == 1]

[ ]  # Treatment of Imbalanced Dataset :: Step 03 :: Undersampling and forming a new Dataframe for balancing the count of classes

     DF_RISk0N = DF_RISk0.sample(Df_COunt1)

[ ]  # Treatment of Imbalanced Dataset :: Step 04 :: Adding the new Dataframe (Class 0 ) and old Dataframe (class 1) to new Dataframe

     New_Df = pd.concat([DF_RISk0N, DF_RISk1], axis=0)
     New_Df.shape

     (61992, 13)

[ ]  # Treatment of Imbalanced Dataset :: Step 05 :: calculating the values of classes in the Dataframe

     New_Df.risk_flag.value_counts()

     1    30996
     0    30996
     Name: risk_flag, dtype: int64

[ ]  # Treatment of Imbalanced Dataset :: Step 06 :: Copying new Dataframe to Original DataFrame

     Df = New_Df
```

3.5.6 Categorical Encoding

The function Categorical Encoding is shown in first figure below. Run the code snippets in both the images shown below.

```
def Categorical_Encoding(Dataframe, columns):

    Dataframe = Dataframe.replace(columns)
    Dataframe.head()

    return Dataframe
```

```
[ ] # Exploratory Data Analysis :: Categorical encoding of Categorical Data { Categorical Data conversion into numeric}


    cleanup_nums = {"married"       :     {"single": 0, "married": 1},
                    "house_ownership":    {"rented": 0, "norent_noown": 1, "owned": 2},
                    "car_ownership"  :    {"no": 0, "yes": 1},

                    "state"         :     {'Madhya_Pradesh': 1 , 'Maharashtra': 2, 'Kerala': 3, 'Odisha': 4, 'Tamil_Nadu': 5,
                                           'Gujarat': 6, 'Rajasthan': 7, 'Telangana': 8, 'Bihar': 9, 'Andhra_Pradesh': 10,
                                           'West_Bengal': 11, 'Haryana': 12, 'Puducherry': 13, 'Karnataka': 14,
                                           'Uttar_Pradesh': 15, 'Himachal_Pradesh': 16, 'Punjab': 17, 'Tripura' : 18,
                                           'Uttarakhand': 19, 'Jharkhand': 20, 'Mizoram': 21, 'Assam': 22,
                                           'Jammu_and_Kashmir': 23, 'Delhi': 24, 'Chhattisgarh': 25, 'Chandigarh': 26,
                                           'Uttar_Pradesh[5]': 27, 'Manipur': 28, 'Sikkim': 29}

    }


    Df = Categorical_Encoding( Df, cleanup_nums)
```

3.5.7 Split Data into Test and Train Sets

Here we will seperate the dataset into test and train sets. Execute the below code snippet for test and train split.

```
[ ]  # Data preperation for Modelling :: Creating a Dataframe without target variable

     Y = Df['risk_flag']
     Y

     X = Df.drop('risk_flag',axis='columns')
     X
```

	income	age	experience	married	house_ownership	car_ownership	state	current_job_years	current_house_years
155777	6382731	51	8	0	0	1	19	5	11
120399	3932375	51	19	0	0	1	15	8	12
76086	4070923	23	6	0	0	1	15	6	11
144346	2111682	31	19	0	1	0	15	9	14
137852	2449147	42	3	1	0	0	9	3	12
...	...	...	...	...	...	...	...	...	...
251973	1244622	35	15	0	0	0	7	3	11
251977	1330613	63	19	0	0	0	9	13	12
251981	1796713	47	2	0	0	0	2	2	12
251982	3182290	52	2	0	0	0	11	2	10
251993	8141027	60	10	0	0	0	2	9	13

61992 rows × 9 columns

```
[ ]  # Data preperation for Modelling :: Creating a Dataframes of Training and Test sets in 80:20 Ratio using  sklearn - train_test_split

     X_Train, X_Test, Y_Train, Y_Test = train_test_split(X,Y,test_size = 0.2, random_state=15 , stratify= Y)
```

3.6 Step 5: Baseline modelling and Evaluation

First apply baseline model in this step.

3.6.1 Model building and Evaluation

In modelbuilding evaluation function the input is gives as model and its name. The model is fitted with X and Y Train sets and then prediction is done using X Test.

```
[ ]  def Modelbuilding_Evaluation(model, Name):


          model.fit(X_Train,Y_Train)

          prediction = model.predict(X_Test)

          print('*****************Results for Model ::' + Name + '********************')
          print('*****************Accuracy Score -*********************************')
          print(accuracy_score(Y_Test, prediction ))

          print('*****************Evaluation results -*******************************')
          print(classification_report(Y_Test, prediction ))

          print('*****************Confusion Matrix -********************************')
          print(confusion_matrix(Y_Test, prediction ))


          fpr, tpr, thresholds = roc_curve(Y_Test, prediction )

          roc_auc = roc_auc_score(Y_Test, prediction )
          print("AUC of ROC Curve:", roc_auc)


          plt.plot(fpr, tpr)
          plt.title("ROC Curve")
          plt.xlabel("False Positive Rate")
          plt.ylabel("True Positive Rate")
          plt.show()
```

Execute below code snippet for Basline Model application. Follow same procedure for all the models.

```
[ ]  # Model Application :: Bagging Classifier

     from sklearn.ensemble import BaggingClassifier

     BaggingClassifier_model = BaggingClassifier()

     Modelbuilding_Evaluation(BaggingClassifier_model, 'BaggingClassifier')
```

```
     ****************Results for Model ::BaggingClassifier******************
     ****************Accuracy Score -**********************************
     0.8501492055810953
     ****************Evaluation results -*****************************
                  precision    recall  f1-score   support

               0       0.84      0.87      0.85      6200
               1       0.86      0.83      0.85      6199

        accuracy                           0.85     12399
       macro avg       0.85      0.85      0.85     12399
    weighted avg       0.85      0.85      0.85     12399
```

3.7 Step 6: Hyper tuned modelling and Evaluation

The code snippet is developed for finding the best parameters of the models. After getting the best parameter, again perform modelling stage with best parameters.

```
[ ]  # Hyper Parameter :: Model Application :: BaggingClassifier

     from sklearn.datasets import make_blobs
     from sklearn.model_selection import RepeatedStratifiedKFold
     from sklearn.model_selection import GridSearchCV
     from sklearn.ensemble import BaggingClassifier

     # define models and parameters
     model = BaggingClassifier()
     n_estimators = [10, 100]
     # define grid search
     grid = dict(n_estimators=n_estimators)
     cv = RepeatedStratifiedKFold(n_splits=2, n_repeats=3, random_state=1)
     grid_search = GridSearchCV(estimator=model, param_grid=grid, n_jobs=-1, cv=cv, scoring='accuracy',error_score=0)
     grid_result = grid_search.fit(X_Train, Y_Train)
     # summarize results
     print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
     means = grid_result.cv_results_['mean_test_score']
     stds = grid_result.cv_results_['std_test_score']
     params = grid_result.cv_results_['params']
     for mean, stdev, param in zip(means, stds, params):
         print("%f (%f) with: %r" % (mean, stdev, param))
```

```
     Best: 0.836395 using {'n_estimators': 100}
     0.835528 (0.001823) with: {'n_estimators': 10}
     0.836395 (0.000980) with: {'n_estimators': 100}
```

4 Deployment or Implementation - Customer Lifetime Value

Follow all the steps mentioned below to execute the code files successfully.

4.1 Uploading of Python Files and Datasets - Only When Using Google Colaboratory

Execute this piece of code only when you are using Google Colab

```
[ ]
      from google.colab import drive
      drive.mount('/content/gdrive')

      Mounted at /content/gdrive
```

4.2 Step 1: Importing Required Python Libraries

All the required libraries must be loaded initially before starting the execution of the code file without error.

```
[ ]  import pandas as pd
     import numpy as np

     from sklearn.metrics import classification_report, accuracy_score
     from sklearn.metrics import mean_squared_error
     from sklearn.metrics import mean_absolute_error


     import seaborn as sns
     import matplotlib.pyplot  as plt
```

4.3 Step 2: Built and Run all the Pre-Created Function

All the functions are already developed to carry out the various task. We need to initialize and run those functions before loading the data. All the functions are mentioned in section 4.4 and 4.5.

4.4 Step 3: Loading the Required Data

Using load data function in first figure below, load data from .csv file to Data Frame. Be careful while providing the link for the Dataset.

```
[ ]  def Load_Data(Path):
         Df = pd.read_csv(Path)
         return Df
```

```
[ ] ## Importing the the Dataset into Dataframe

    Path = '/content/gdrive/My Drive/Research Project :: Msc Data Analytics/Datasets/Customer Lifetime Value/data.csv'

    Df = Load_Data(Path)
```

4.5 Step 4: Data Pre-Processing

Step by step run code snippets for the below-mentioned steps.

4.5.1 Exploring Data Insights

Initially run the function Data Insights and then the piece of code in the second image.

```
[ ] def Data_Insights(Dataframe):

        print('**********************************************First Five Rows of Data**********************************************')
        print(Dataframe.head())

        print('**********************************************Last Five Rows of Data**********************************************')
        print(Dataframe.tail())

        print('**********************************************Dataframe Columns Details**********************************************')
        print(Dataframe.info())


        print('**********************************************Dataframe statistical Data**********************************************')
        print(Dataframe.describe())
```

```
[ ] Data_Insights(Df)

**********************************************First Five Rows of Data**********************************************************
           Id   income  age  ...  current_job_years current_house_years risk_flag
155777  155778  6382731   51  ...                  5                  11         0
120399  120400  3932375   51  ...                  8                  12         0
76086    76087  4070923   23  ...                  6                  11         0
144346  144347  2111682   31  ...                  9                  14         0
137852  137853  2449147   42  ...                  3                  12         0

[5 rows x 13 columns]
**********************************************Last Five Rows of Data***********************************************************
           Id   income  age  ...  current_job_years current_house_years risk_flag
251973  251974  1244622   35  ...                  3                  11         1
251977  251978  1330613   63  ...                 13                  12         1
251981  251982  1796713   47  ...                  2                  12         1
251982  251983  3182290   52  ...                  2                  10         1
251993  251994  8141027   60  ...                  9                  13         1
```

4.5.2 Null values Check

Run both piece of codes to check the null values.

```
[ ] # Exploratory Data Analysis :: Exploring or Detecting the Null Values in the Dataframe

    Df.isna().sum()
```

4.5.3 Duplicate values Check and Deleting Unwanted Columns

Here in the first image the data delete function is shown first run this function and then run snippet of codes in below images.

```
[ ]  def Data_Delete(Dataframe, columns):

         Dataframe.drop(columns, axis=1, inplace=True)
         return Dataframe
```

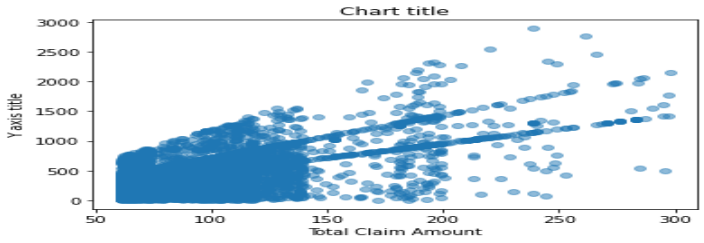
```
[ ] # Exploratory Data Analysis :: Exploring or Detecting the Duplicates values in the Dataframe

    Df.duplicated().sum()
```

```
[ ] # Exploratory Data Analysis :: Dropping the unwanted columns of the Dataframe

    columns =['Customer','Effective To Date','Policy','Renew Offer Type']
    Data_Delete(Df, columns)
```

4.5.4 Data Vizualization

Run all the snippets of data visualization from I to VI.

```
# Data Visualization - I

plt.scatter(Df['Monthly Premium Auto'],Df['Total Claim Amount'],alpha=0.5 )
#Adding the aesthetics
plt.title('Chart title')
plt.xlabel('Total Claim Amount')
plt.ylabel('Y axis title')
#Show the plot
plt.show()
```




```
[ ]
# Data Visualization - II

#Creating marginal graphs
sns.jointplot(x=Df["Policy Type"], y=Df["Monthly Premium Auto"], kind='scatter')
# Show the plot
plt.show()
```

4.5.5 Categorical Encoding

The function Categorical Encoding is shown in first figure below. Run the code snippets in both the images shown below.

```
def Categorical_Encoding(Dataframe, columns):

    Dataframe = Dataframe.replace(columns)
    Dataframe.head()

    return Dataframe
```

```
[ ] # Exploratory Data Analysis :: Categorical encoding of Categorical Data { Categorical Data conversion into numeric}


    cleanup_nums = {"State":              {"Arizona": 1, "California": 2, 'Nevada': 3, 'Oregon': 4, 'Washington': 5},
                    "Coverage":           {"Basic": 1, "Extended": 2, "Premium": 3},
                    "Response":           {"No": 0, "Yes": 1},
                    "Education":          {"Bachelor": 1, "College": 2, 'Doctor': 3, 'Master': 4, 'High School or Below': 5},
                    "EmploymentStatus":   {"Employed": 1, "Unemployed": 2, 'Medical Leave': 3, 'Disabled': 4, 'Retired': 5},
                    "Gender":             {"F": 0, "M": 1},
                    "Policy Type":        {"Corporate Auto": 1, "Personal Auto": 2, "Special Auto": 3},
                    "Sales Channel":      {"Agent": 1, "Call Center": 2, 'Web': 3, 'Branch': 4},
                    "Vehicle Class":      {"Two-Door Car": 1, "Four-Door Car": 2, 'SUV': 3, 'Luxury SUV': 4, 'Sports Car': 5, 'Luxury Car': 6},
                    "Vehicle Size":       {"Medsize": 1, "Small": 2, "Large": 3},
                    "Location Code":      {"Suburban": 1, "Rural": 2, "Urban": 3},
                    "Marital Status":     {"Married": 1, "Single": 2, "Divorced": 3}

    }

    Df = Categorical_Encoding( Df, cleanup_nums)
```

4.6 Step 5: Baseline modelling and Evaluation

Application of models with default parameters.

4.6.1 Model building and Evaluation

In modelbuilding evaluation function the input is gives as model and its name. The model is fitted with X and Y Train sets and then prediction is done using X Test.

In Evaluation results function the input is given as model, perdicted values and model name. All the evalution matrices calculation is done and displayed.

```
[ ]  def Evaluation_Results(model, prediction, Name):
       # ***************************************************************************************************
       #display adjusted R-squared
       R2 = model.score(X_Train, Y_Train)
       Adj_R2 = 1 - (1-model.score(X_Train, Y_Train))*(len(Y_Train)-1)/(len(Y_Train)-X_Train.shape[1]-1)


       # ***************************************************************************************************
       import math

       MSE = np.square(np.subtract(Y_Test,prediction)).mean()

       RMSE = math.sqrt(MSE)

       # ***************************************************************************************************
       Accuracy = model.score(X_Test,Y_Test)

       print('*****************Results for Model ::' + Name + '*******************')

       print("Accuracy Score of the Model:\n")
       print(Accuracy)

       print("R - Square Value:\n")
       print(R2)

       print("Adjusted R - Square Value:\n")
       print(Adj_R2)

       print("Mean Square Error:\n")
       print(MSE)

       print("Root Mean Square Error:\n")
       print(RMSE)
```

```
[ ]  def Modelbuilding_Evaluation(model, Name):


         model.fit(X_Train,Y_Train)

         prediction = model.predict(X_Test)

         Evaluation_Results(model, prediction, Name)
```

Execute below code snippet for Basline Model application. Follow same procedure for all the models.

```
[ ] # Modelling and Implementation :: Importing, modelling and Implementing
    from sklearn.ensemble import RandomForestRegressor
    # Instantiate model with 1000 decision trees

    rf = RandomForestRegressor(n_estimators = 100, random_state = 42)
    # Train the model on training data

    Modelbuilding_Evaluation(rf, 'RandomForestRegressor')
```

4.7 Step 6: Hyper tuned modelling and Evaluation

The code snippet is developed for finding the best parameters of the models. After getting the best parameter, again perform modelling stage with best parameters.

```
[ ] import xgboost as xgb
    from sklearn.model_selection import GridSearchCV

    params = { 'max_depth': [3,6,10],
               'learning_rate': [0.01, 0.05, 0.1],
               'n_estimators': [10, 100],
               'colsample_bytree': [0.3, 0.7]}
    xgbr = xgb.XGBRegressor(seed = 20)
    clf = GridSearchCV(estimator=xgbr,
                       param_grid=params,
                       scoring='neg_mean_squared_error',
                       verbose=1)
    clf.fit(X_Train,Y_Train)
    print("Best parameters:", clf.best_params_)
    print("Lowest RMSE: ", (-clf.best_score_)**(1/2.0))
```