Configuration Manual

Sumeet Kumar

X18188231

1. Introduction

The configuration manual contains all the pertinent information related to the software and hardware used in the research project. Also, it specifies the important libraries that are used and the data description is given in section 3. Moreover, it elucidates several steps that needs to be taken to reproduce the work in any machine satisfying the requirements which is covered in the following sections

1 Environment Specification

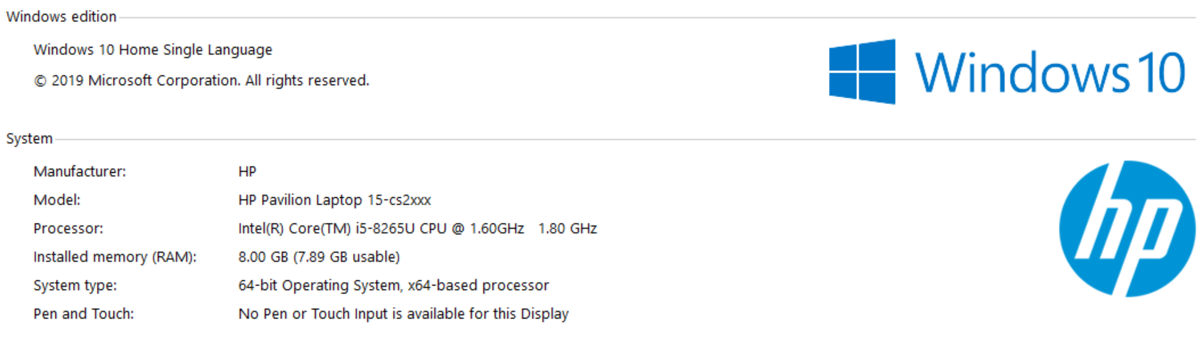
The MSc. Research project runs on a system which has certain specifications for both software and hardware that are described in the following subsections.

1.1 Hardware Specifications

This project is implemented on the hardware with the following configurations:

Hardware	Configuration
System	
Operation System	Windows 10 (64-bit Operating System)
RAM	8 GB
Hard Disk	1 TB, 256 SSD
Graphic Card	2 GB Nvidia

Table 1



Windows edition

Windows 10 Home Single Language

© 2019 Microsoft Corporation. All rights reserved.

System

Manufacturer:	HP
Model:	HP Pavilion Laptop 15-cs2xxx
Processor:	Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz
Installed memory (RAM):	8.00 GB (7.89 GB usable)
System type:	64-bit Operating System, x64-based processor
Pen and Touch:	No Pen or Touch Input is available for this Display

Figure 1

1.2 Software Specification

In this project, plethora of software are used which are represented in Table 2.

Software	Configuration

Operating System	**Windows 10 (64-bit Operating System)**
IDE	**Google Colab**
Scripting Language & Version	**Python, Python 1.7**

Table 2

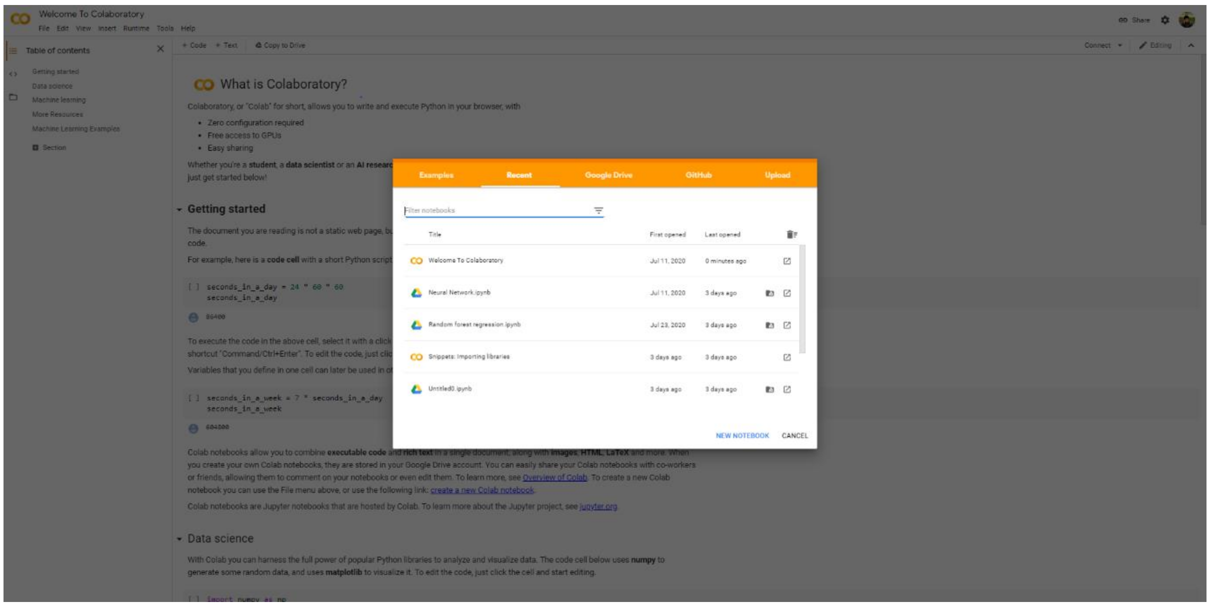
1.3 Integrated Development Environment

Google Colab

Google Colab is a cloud based Collaboratory that is created in order to help disseminate machine learning education and research. This is like a Jupiter notebook environment that doesn't need any set up and would run entirely on the cloud.
It helps to combine both executable code and rich text in one document. One can put images, HTML, LaTeX and much more to it. Figure 1 illustrates how a google colab looks after login in.

Steps to reach the Figure 1:

1- Login in Google.
2- Open Google colab and login in it using the email address.
3- Once you are on inside the Google Colab, you'll have option to load any code you have from GitHub, Google Drive or just upload. You can upload any data file you want to use for the learning method and built model. Once the data is mounted on drive one can use it by copying its path whenever needed.



What is Colaboratory?

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researc**
just get started below!

Getting started

The document you are reading is not a static web page, bu
code.

For example, here is a **code cell** with a short Python script

```
[ ] seconds_in_a_day = 24 * 60 * 60
    seconds_in_a_day
```

```
86400
```

To execute the code in the above cell, select it with a click
shortcut "Command/Ctrl+Enter". To edit the code, just clic
Variables that you define in one cell can later be used in ot

```
[ ] seconds_in_a_week = 7 * seconds_in_a_day
    seconds_in_a_week
```

```
604800
```

Title	First opened	Last opened
Welcome To Colaboratory	Jul 11, 2020	0 minutes ago
Neural Network.ipynb	Jul 11, 2020	3 days ago
Random forest regression.ipynb	Jul 23, 2020	3 days ago
Snippets: Importing libraries	3 days ago	3 days ago
Untitled0.ipynb	3 days ago	3 days ago

Colab notebooks allow you to combine **executable code** and **rich text** in a single document, along with **images**, **HTML**, **LaTeX** and more. When you create your own Colab notebooks, they are stored in your Google Drive account. You can easily share your Colab notebooks with co-workers or friends, allowing them to comment on your notebooks or even edit them. To learn more, see Overview of Colab. To create a new Colab notebook you can use the File menu above, or use the following link: create a new Colab notebook.

Colab notebooks are Jupyter notebooks that are hosted by Colab. To learn more about the Jupyter project, see jupyter.org.

Data science

With Colab you can harness the full power of popular Python libraries to analyze and visualize data. The code cell below uses **numpy** to generate some random data, and uses **matplotlib** to visualize it. To edit the code, just click the cell and start editing.

```
[ ] import numpy as np
```

Figure 2

2 Libraries

After one has started the notebook installation of vital libraries is done. Important libraries required to execute any project are installed. And if some libraries which are not there then one can use pip install (Library name).

Sklearn.ensemble	Sklearn.neighbors
sklearn.model_selection	Tensorflow.keras.models
skimage	Skimage.io
tensorflow	opencv
Numpy	Pandas
Sklearn,preprocessing	Sklearn
Tensorflow.keras.layers	Sklearn.metrics
Matplotlib	Matplotlib.pyplot
sklearn.pipeline	sklearn.model_selection
sklearn.preprocessing	keras.wrappers.scikit_learn
StandardScaler	KFold
keras.models	keras.layers

Table 3

3 Dataset

There are two different Data sets which have been taken for the research. The dataset are taken from Kaggle which is opensource.

3.1 European Soccer Data set.

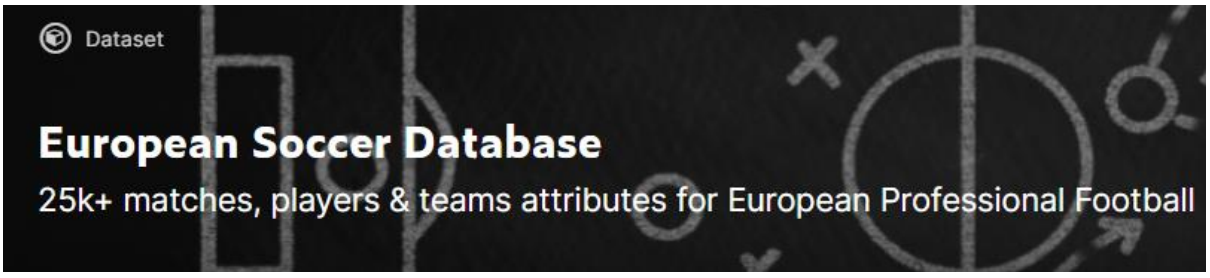

Dataset
European Soccer Database
25k+ matches, players & teams attributes for European Professional Football

Figure 3

The European Soccer Dataset contains seven different files, these files are Country, League, Match, Player, Player Attributes, Team, Team Attributes. A data contains a total of 222,796 rows and 299 rows. Every File has a set of variables which are going to be used in the research. Also, more variables were calculated.

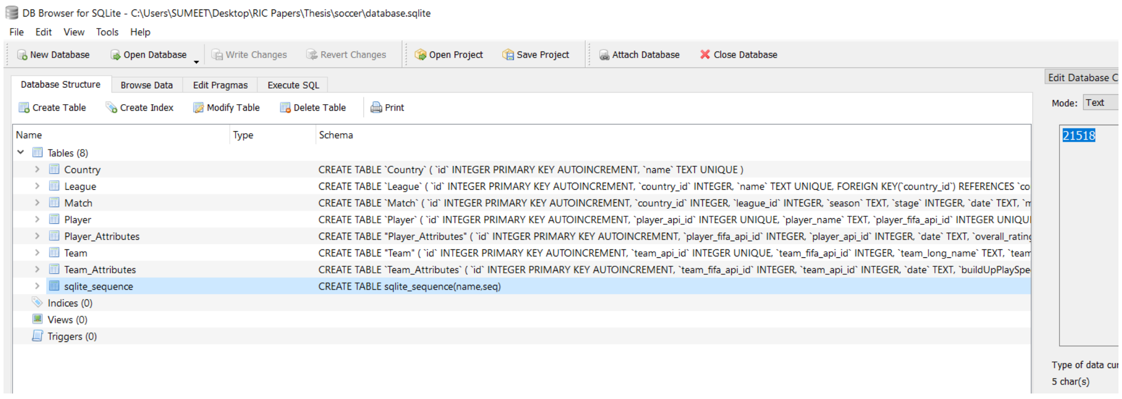
3.2 Complete Player Dataset

Dataset
FIFA 20 complete player dataset
18k+ players, 100+ attributes extracted from the latest edition of FIFA

Figure 4

The FIFA 20 complete player dataset contains all players around the world in different leagues. We have taken players who were part of FIFA for the year 2015-2016. All skills and attributes of players are available and are used in the research.

These First Dataset had files in SQLite format, these were transformed into CSV with the help of DB browser, DB browser is a free software which can be downloaded from google. After installing it, one can view the files which are present in SQ format. For our research these files were downloaded in csv format. Figure illustrates the files which are present in data and converted into csv.



Name	Type	Schema
Tables (8)		
Country		CREATE TABLE `Country` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `name` TEXT UNIQUE)
League		CREATE TABLE `League` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `country_id` INTEGER, `name` TEXT UNIQUE, FOREIGN KEY(`country_id`) REFERENCES `co
Match		CREATE TABLE `Match` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `country_id` INTEGER, `league_id` INTEGER, `season` TEXT, `stage` INTEGER, `date` TEXT, `m
Player		CREATE TABLE `Player` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `player_api_id` INTEGER UNIQUE, `player_name` TEXT, `player_fifa_api_id` INTEGER UNIQU
Player_Attributes		CREATE TABLE "Player_Attributes" (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `player_fifa_api_id` INTEGER, `player_api_id` INTEGER, `date` TEXT, `overall_ratin
Team		CREATE TABLE "Team" (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `team_api_id` INTEGER UNIQUE, `team_fifa_api_id` INTEGER, `team_long_name` TEXT, `team
Team_Attributes		CREATE TABLE `Team_Attributes` (`id` INTEGER PRIMARY KEY AUTOINCREMENT, `team_fifa_api_id` INTEGER, `team_api_id` INTEGER, `date` TEXT, `buildUpPlaySpe
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
Indices (0)		
Views (0)		
Triggers (0)		

Figure 5

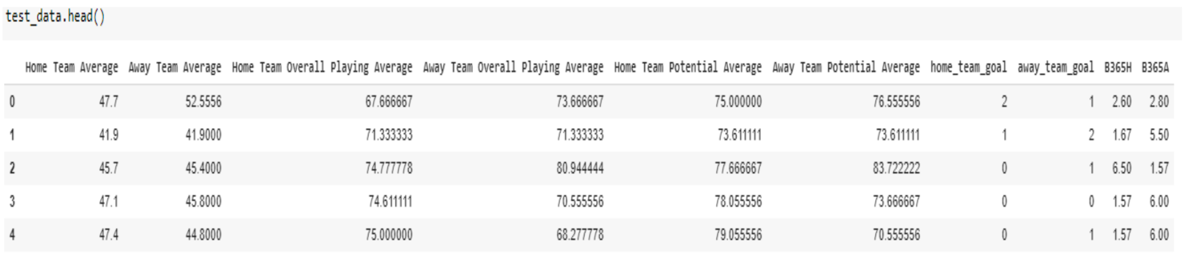
The second dataset contains player dataset from last 10 years, for this research we have simply downloaded the players who were playing FIFA in 2015.

4 Data Pre-processing

The two files downloaded contained 238,254 rows and 333 columns which in total is 14,441,635 integer value data. This data is cleaned with the help of Ms Excel and mounted to Google Colab for building model.

4.1 Data Scaling

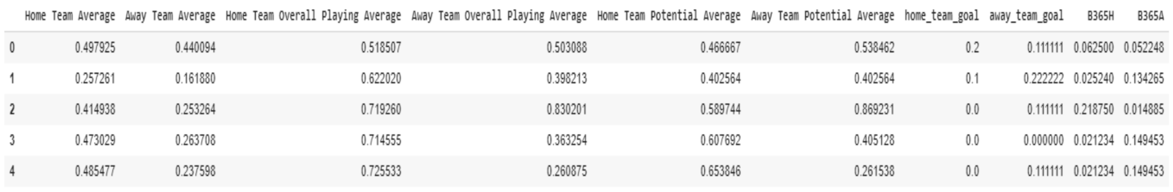
Data scaling is performed for scaling the values.



```
test_data.head()
```

	Home Team Average	Away Team Average	Home Team Overall Playing Average	Away Team Overall Playing Average	Home Team Potential Average	Away Team Potential Average	home_team_goal	away_team_goal	B365H	B365A
0	47.7	52.5556	67.666667	73.666667	75.000000	76.555556	2	1	2.60	2.80
1	41.9	41.9000	71.333333	71.333333	73.611111	73.611111	1	2	1.67	5.50
2	45.7	45.4000	74.777778	80.944444	77.666667	83.722222	0	1	6.50	1.57
3	47.1	45.8000	74.611111	70.555556	78.055556	73.666667	0	0	1.57	6.00
4	47.4	44.8000	75.000000	68.277778	79.055556	70.555556	0	1	1.57	6.00

Figure 6 Before Scaling

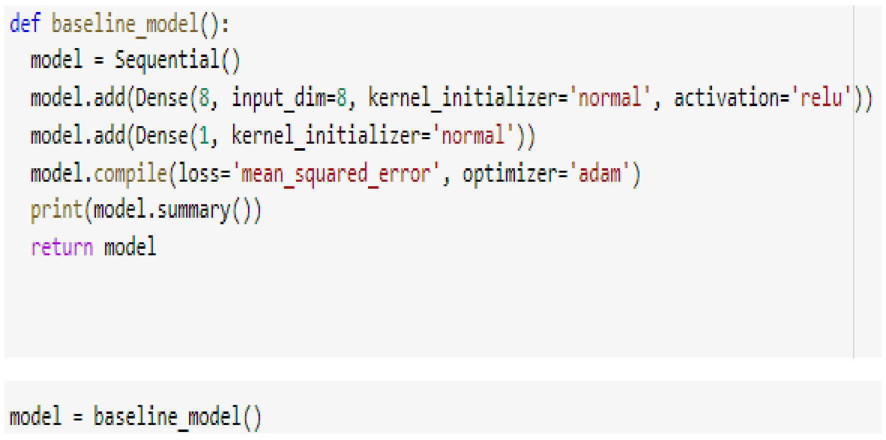


	Home Team Average	Away Team Average	Home Team Overall Playing Average	Away Team Overall Playing Average	Home Team Potential Average	Away Team Potential Average	home_team_goal	away_team_goal	B365H	B365A
0	0.497925	0.440094	0.518507	0.503088	0.466667	0.538462	0.2	0.111111	0.062500	0.052248
1	0.257261	0.161880	0.622020	0.398213	0.402564	0.402564	0.1	0.222222	0.025240	0.134265
2	0.414938	0.253264	0.719260	0.830201	0.589744	0.869231	0.0	0.111111	0.218750	0.014885
3	0.473029	0.263708	0.714555	0.363254	0.607692	0.405128	0.0	0.000000	0.021234	0.149453
4	0.485477	0.237598	0.725533	0.260875	0.653846	0.261538	0.0	0.111111	0.021234	0.149453

Figure 7 Post Scaling

1. Model --- CASE 1

a. Neural Network Model

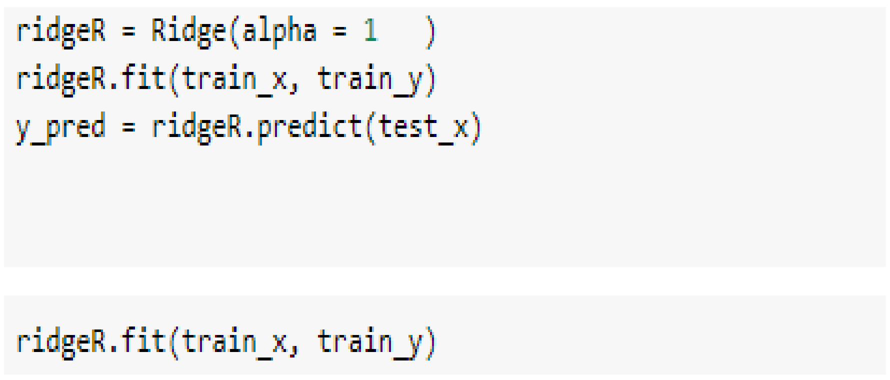


```
def baseline_model():
  model = Sequential()
  model.add(Dense(8, input_dim=8, kernel_initializer='normal', activation='relu'))
  model.add(Dense(1, kernel_initializer='normal'))
  model.compile(loss='mean_squared_error', optimizer='adam')
  print(model.summary())
  return model
```

```
model = baseline_model()
```

Figure 8

b. Ridge Regression

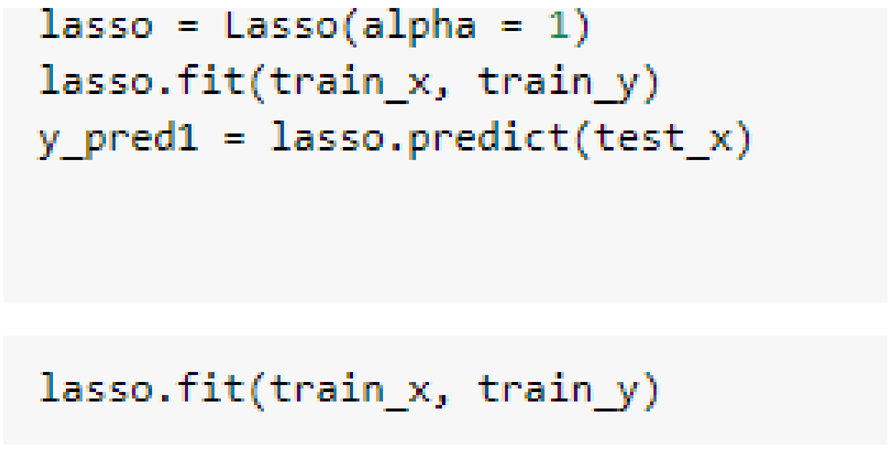


```
ridgeR = Ridge(alpha = 1    )
ridgeR.fit(train_x, train_y)
y_pred = ridgeR.predict(test_x)
```

```
ridgeR.fit(train_x, train_y)
```

Figure 9

c. Lasso Regression

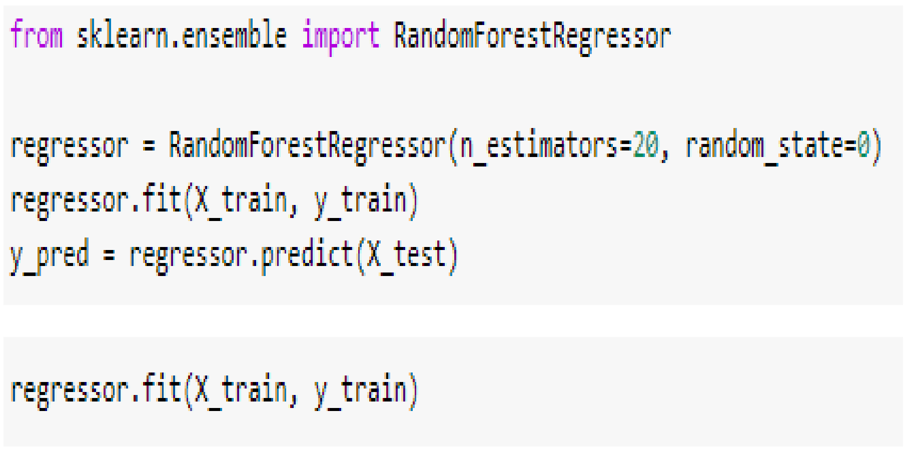


```
lasso = Lasso(alpha = 1)
lasso.fit(train_x, train_y)
y_pred1 = lasso.predict(test_x)
```

```
lasso.fit(train_x, train_y)
```

Figure 10

d. Random Forest



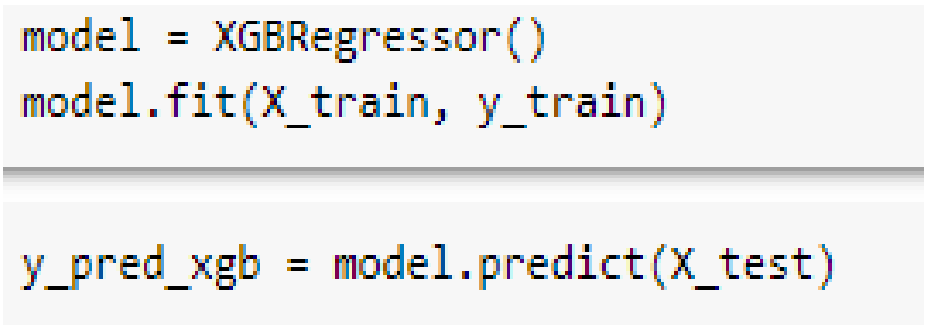
```
from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n_estimators=20, random_state=0)
regressor.fit(X_train, y_train)
y_pred = regressor.predict(X_test)
```

```
regressor.fit(X_train, y_train)
```

Figure 11

e. XGBoost



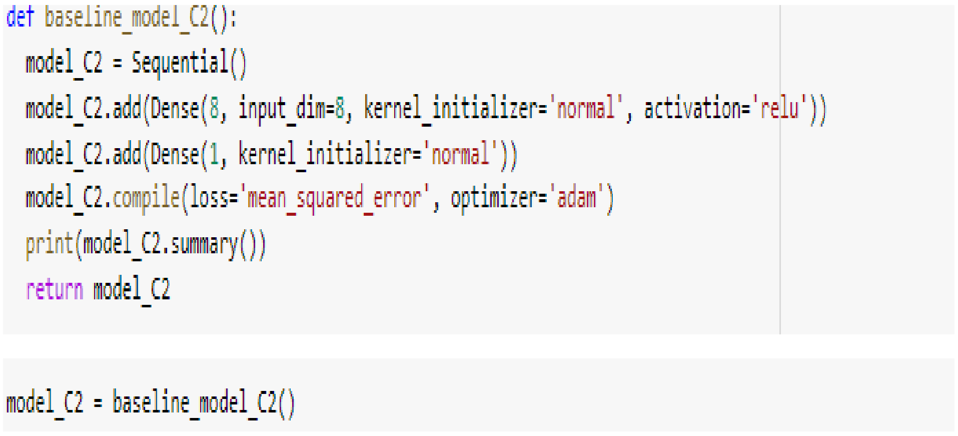
```
model = XGBRegressor()
model.fit(X_train, y_train)
```

```
y_pred_xgb = model.predict(X_test)
```

Figure 11

2. Models --- CASE 2

a. Neural Network

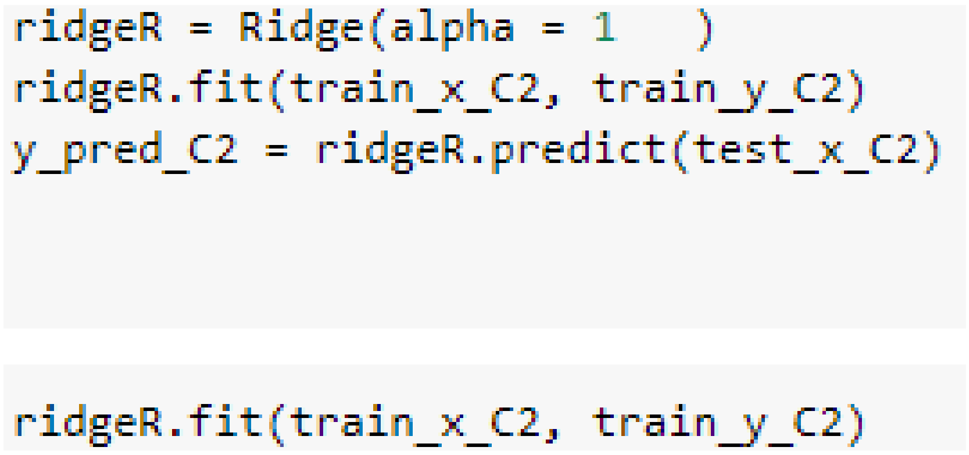


```
def baseline_model_C2():
  model_C2 = Sequential()
  model_C2.add(Dense(8, input_dim=8, kernel_initializer='normal', activation='relu'))
  model_C2.add(Dense(1, kernel_initializer='normal'))
  model_C2.compile(loss='mean_squared_error', optimizer='adam')
  print(model_C2.summary())
  return model_C2
```

```
model_C2 = baseline_model_C2()
```

Figure 12

b. Ridge Regression

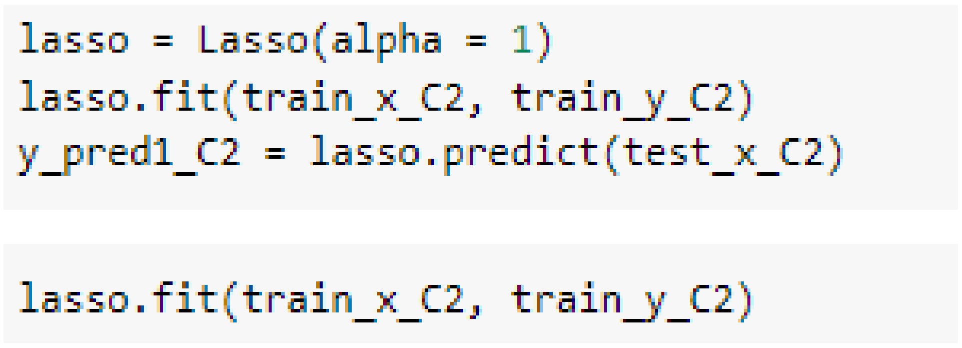


```
ridgeR = Ridge(alpha = 1    )
ridgeR.fit(train_x_C2, train_y_C2)
y_pred_C2 = ridgeR.predict(test_x_C2)
```

```
ridgeR.fit(train_x_C2, train_y_C2)
```

Figure 13

c. Lasso Regression

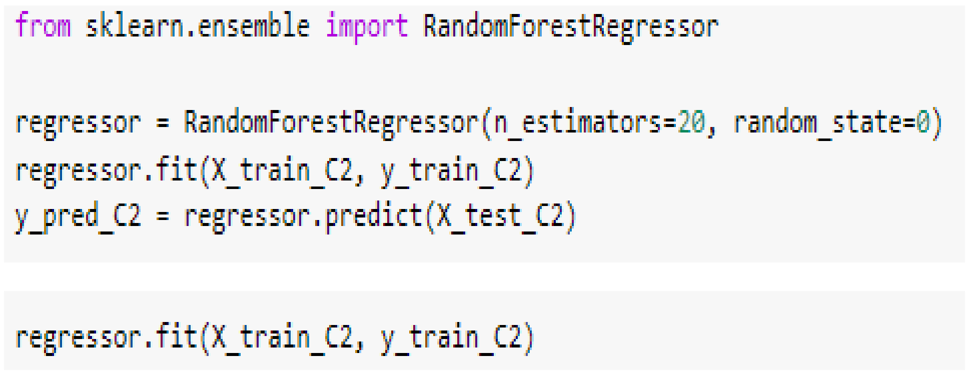


```
lasso = Lasso(alpha = 1)
lasso.fit(train_x_C2, train_y_C2)
y_pred1_C2 = lasso.predict(test_x_C2)
```

```
lasso.fit(train_x_C2, train_y_C2)
```

Figure 14

d. Random Forest



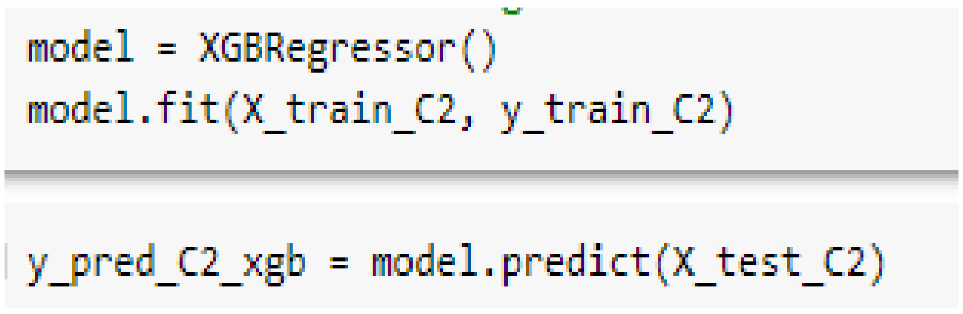
```
from sklearn.ensemble import RandomForestRegressor

regressor = RandomForestRegressor(n_estimators=20, random_state=0)
regressor.fit(X_train_C2, y_train_C2)
y_pred_C2 = regressor.predict(X_test_C2)
```

```
regressor.fit(X_train_C2, y_train_C2)
```

Figure 15

e. XGBoost



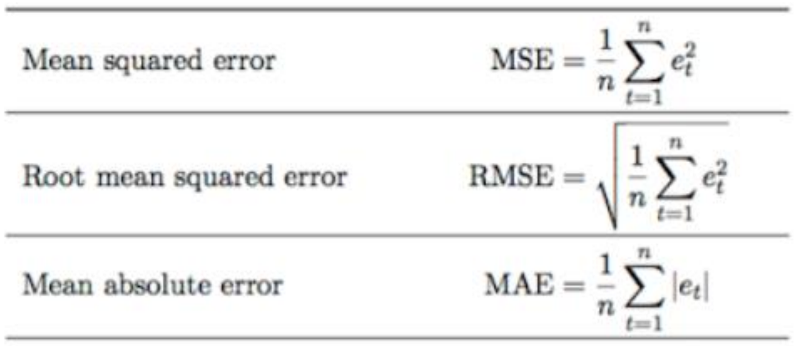
```
model = XGBRegressor()
model.fit(X_train_C2, y_train_C2)
```

```
y_pred_C2_xgb = model.predict(X_test_C2)
```

Figure 16

5 6 Evaluation

For the Evaluation three Metrics have been chosen. MSE, MAE, RMSE



Mean squared error	$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$

Figure 17

6 References

Bhattacharyya, S., 2020. *Ridge And Lasso Regression: L1 And L2 Regularization*. [online] Medium. Available at: <https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b> [Accessed 2 August 2020].

Malik, U., 2020. *Random Forest Algorithm With Python And Scikit-Learn*. [online] Stack Abuse. Available at: https://stackabuse.com/random-forest-algorithm-with-python-and-scikit-learn/ [Accessed 14 July 2020].