National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Omkar Tawade
Student ID: x19232136

School of Computing
National College of Ireland

Supervisor: Noel Cosgrave

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Omkar Tawade
Student ID:	x19232136
Programme:	Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Noel Cosgrave
Submission Due Date:	23/09/2021
Project Title:	Configuration Manual
Word Count:	XXX
Page Count:	15

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:	Omkar Tawade
Date:	23rd September 2021

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

Omkar Tawade
x19232136

1 Hardware/Software Requirements

The configuration manual outlines the steps that must be followed when running the scripts used in the research. This manual will assist you in successfully running the code. This manual also contains details on the hardware configuration of the system on which the code was run. The system's minimal needed setup is also specified.

2 System Specification

2.1 Hardware Requirements

The following are the hardware specifications for the system on which the research project is run.

- Processor: 1.1 GHz Dual-Core Intel Core i3
- RAM: 8 GB
- Storage: 256 GB SSD
- Operating System: macOS Big Sur

2.2 Software Requirements

The following programming tools were used in this research.

- Google Colaboratory (Cloud based Jupyter notebook environment)
- Python version 3.7
- Microsoft Excel
- Overleaf

3 Setting up environment

This section will guide you through the process of setting up a Google Colaboratory environment. For a better understanding, the following images are provided.

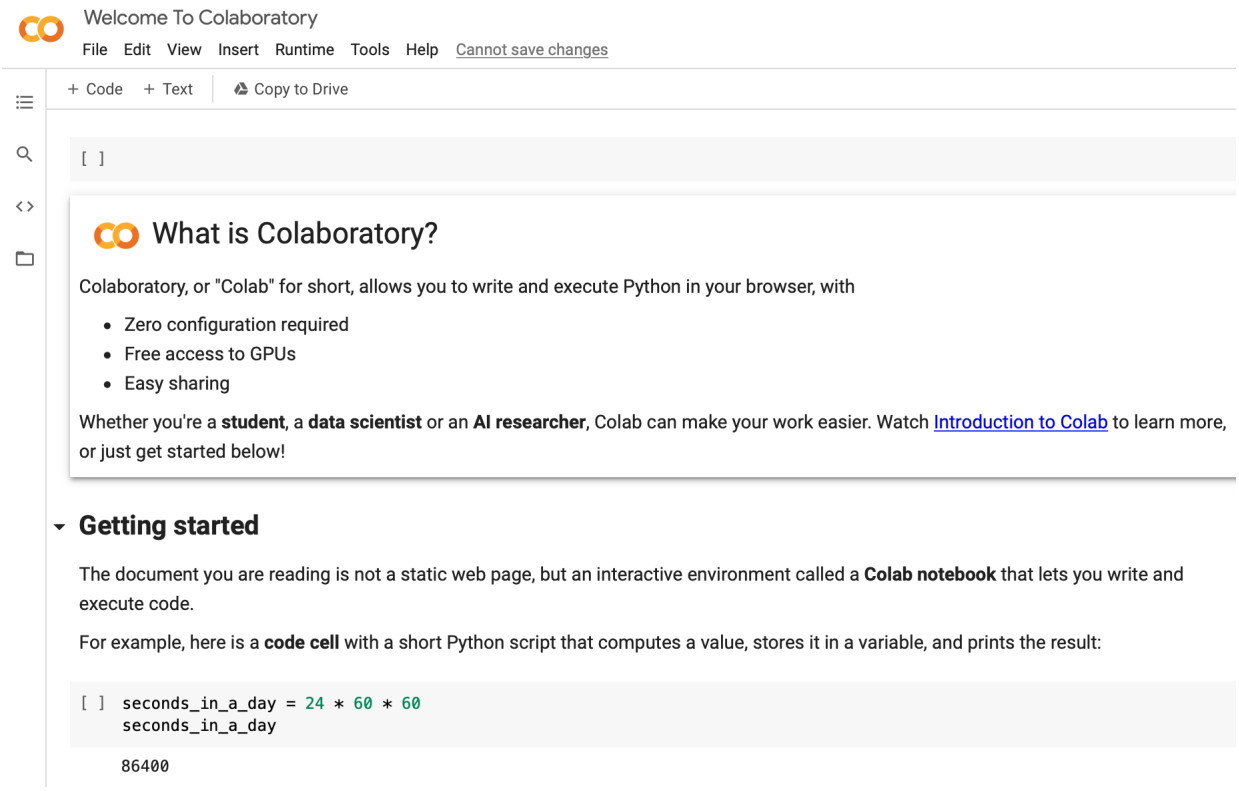


Figure 1: Google Colab Setup

4 Data Selection

The data set is downloaded from the Kaggle dataset repository. The 'IPL Complete Dataset (2008-2020)' dataset shown in Figure 2 is used for this experiment.

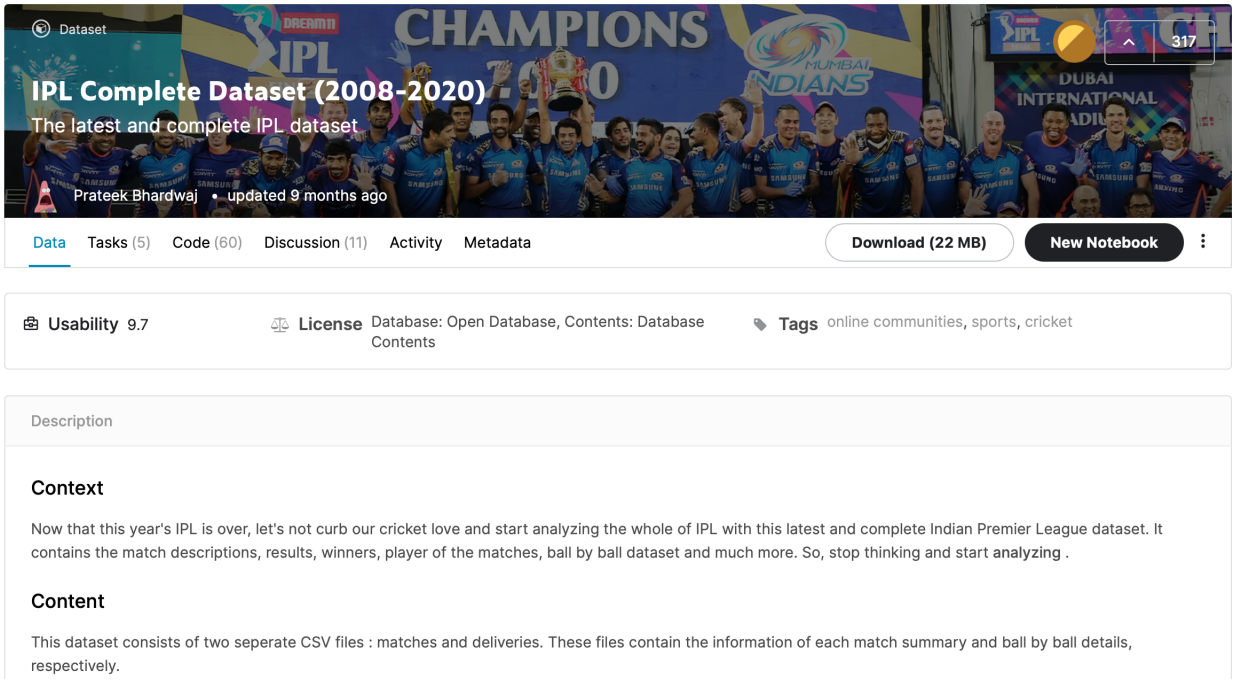


Figure 2: IPL Complete Dataset (2008-2020)

5 Implementation

The libraries needed to develop a model to predict the score of rain-interrupted matches are listed below.

- Pandas
- NumPy
- OS
- google.colab
- SKlearn
- Matplotlib

```
import pandas as pd
import numpy as np
import os
from google.colab import drive
from google.colab import files
from sklearn.model_selection import train_test_split
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import AdaBoostRegressor
import xgboost as xgb
from sklearn.metrics import mean_squared_error as MSE
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import explained_variance_score
import matplotlib.pyplot as plt
```

Listing 1: Libraries

5.1 Importing Data

Upload the downloaded dataset to google drive from a Gmail account. After mounting, the google drive to colab notebook as shown in the figure 3. Click on the URL and select the Gmail account and enter the authentication code as shown in the figure 4.

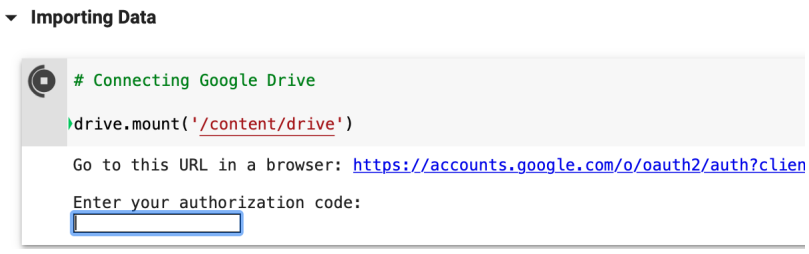


Figure 3: Connecting Google Drive

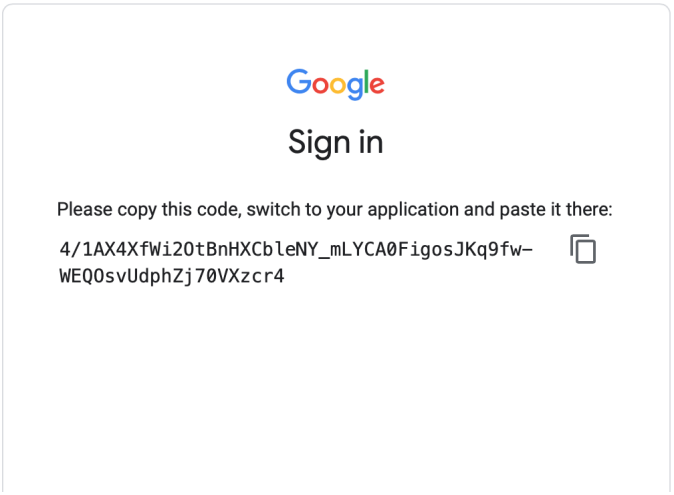


Figure 4: Authentication code

5.2 Reading Data

Pandas data frame was used to read the data as shown below.

```
# Data was downloaded from kaggle.com
# Reading IPL 2008 to 2020 data

ipl = pd.read_csv("IPL Ball-by-Ball 2008-2020.csv")

ipl.head(1)
```

Listing 2: Reading Data

5.3 Data Processing

Initially, a subset of data was selected for the model building process. In order to make data consistent, teams that played two or three-season were removed from the dataset. Mostly IPL tournament consists of eight teams. Along with this, we update the team names to their recent name as shown below.

```
# Subsetting the dataset with only required features

ipl=ipl[['id','inning','batting_team','bowling_team','over','ball','
    total_runs','player_dismissed']]

ipl.isnull().sum()

ipl.head(1)

ipl['batting_team'].unique()

# Subsetting the dataset with only consistent teams

ipl=ipl[~ipl.batting_team.str.contains("Pune")]
ipl=ipl[~ipl.batting_team.str.contains("Kochi")]
ipl=ipl[~ipl.batting_team.str.contains("Gujarat")]
ipl=ipl[~ipl.bowling_team.str.contains("Pune")]
```

```
ipl=ipl[~ipl.bowling_team.str.contains("Kochi")]
ipl=ipl[~ipl.bowling_team.str.contains("Gujarat")]

ipl.head(1)
```

Listing 3: Data Pre-processing

We renamed all teams with their initials to make data easily readable as shown below.

```
ipl['batting_team'].unique()

# renaming the team names

ipl['batting_team']=ipl['batting_team'].replace({'Sunrisers Hyderabad': '
    SRH', 'Deccan Chargers': 'SRH','Royal Challengers Bangalore': 'RCB', '
    Mumbai Indians': 'MI',
                        'Kolkata Knight Riders': 'KKR', 'Kings XI Punjab':
                            'KXIP','Delhi Daredevils': 'DC', 'Delhi Capitals
                            ': 'DC',
                        'Chennai Super Kings': 'CSK', 'Rajasthan Royals': '
                            RR'})

ipl['bowling_team']=ipl['bowling_team'].replace({'Sunrisers Hyderabad': '
    SRH', 'Deccan Chargers': 'SRH','Royal Challengers Bangalore': 'RCB', '
    Mumbai Indians': 'MI',
                        'Kolkata Knight Riders': 'KKR', 'Kings XI Punjab':
                            'KXIP','Delhi Daredevils': 'DC', 'Delhi Capitals
                            ': 'DC',
                        'Chennai Super Kings': 'CSK', 'Rajasthan Royals': '
                            RR'})

ipl['batting_team'].unique()

ipl['bowling_team'].unique()
```

Listing 4: Renaming teams

5.4 Feature Engineering

The next step involves the feature engineering process. As shown below, we created our dependent variable of the model.

```
"""##### **Feature engineering**"""

# replacing the null values with 0

ipl = ipl.replace(np.nan, 0)
ipl.reset_index(inplace=True)
ipl = ipl.drop('index', 1)

# total runs scored till current ball

ipl['total_score']=ipl.groupby(['id', 'inning'])['total_runs'].apply(lambda
     x: x.cumsum())

# total wickets fallen in the innings
```

```
ipl['player_dismissed']=np.where(ipl['player_dismissed']==0, 0, 1)

ipl['total_wickets']=ipl.groupby(['id', 'inning'])['player_dismissed'].
    apply(lambda x: x.cumsum())

ipl = ipl.replace(np.nan, 0)
ipl = ipl.replace(np.inf, 0)

#total runs scored in the innings

ipl['total']=ipl.groupby(['id','inning'])['total_runs'].transform('sum')
```

Listing 5: Generating dependent and independent variables

Further, we added our novel features to the dataset as shown below.

```
# runs scored in last 5 overs

df=ipl.groupby(['id','inning'])['total_runs'].rolling(min_periods=1, window
    =30).sum().reset_index()

ipl['prev_runs_5_overs']=df[['total_runs']]

# wickets fallen in last 5 overs

df=ipl.groupby(['id','inning'])['player_dismissed'].rolling(min_periods=1,
    window=30).sum().reset_index()

ipl['prev_wickets_5_overs']=df[['player_dismissed']]

# dot balls in last 5 overs

ipl['prev_5_overs_dot_balls']=ipl['total_runs']
ipl['prev_5_overs_dot_balls']=np.where(ipl['prev_5_overs_dot_balls']==0, 1,
     0)

df=ipl.groupby(['id','inning'])['prev_5_overs_dot_balls'].rolling(
    min_periods=1, window=30).sum().reset_index()
ipl['prev_5_overs_dot_balls']=df[['prev_5_overs_dot_balls']]

# boundaries scored in last 5 overs

ipl['prev_5_overs_boundaries']=ipl['total_runs']
ipl['prev_5_overs_boundaries']=np.where(ipl['prev_5_overs_boundaries']>3,
    1, 0)

df=ipl.groupby(['id','inning'])['prev_5_overs_boundaries'].rolling(
    min_periods=1, window=30).sum().reset_index()
ipl['prev_5_overs_boundaries']=df[['prev_5_overs_boundaries']]

ipl['overs']=ipl['over'].astype(str) + '.' + ipl['ball'].astype(str)

ipl

#Latex Output
print(ipl.iloc[:, 8:15].head(5).to_latex(index=False))
```

```
ipl.head(1)
```

Listing 6: Features related to last five overs

5.5 Data Splitting

In this step, we initially split the data into two parts based on the first innings and second innings. In each innings data, we converted batting and bowling team categorical variables into different columns. Further, using the train and split function, we split 75% of data for training and 25% of data for validations.

```
"""##### **Building model for first innings**"""

ipl_first_innings=ipl.copy()

ipl_first_innings=ipl_first_innings.loc[ipl_first_innings['inning'] == 1]

ipl_first_innings=pd.get_dummies(data=ipl_first_innings , columns=['
    batting_team', 'bowling_team'])

ipl_first_innings.columns

ipl_first_innings=ipl_first_innings[['id','batting_team_CSK', '
    batting_team_DC','batting_team_KKR', 'batting_team_KXIP', '
    batting_team_MI',
                    'batting_team_RCB', 'batting_team_RR', 'batting_team_SRH
                        ','bowling_team_CSK', 'bowling_team_DC', '
                        bowling_team_KKR',
                   'bowling_team_KXIP', 'bowling_team_MI', 'bowling_team_RCB
                      ','bowling_team_RR', 'bowling_team_SRH',
                    'overs','total_score','total_wickets','prev_runs_5_overs
                        ','prev_wickets_5_overs','prev_5_overs_dot_balls','
                        prev_5_overs_boundaries','total']]

len(ipl_first_innings)

ipl_first_innings
ipl_first_innings.to_csv('First_Innings_Data.csv')
!cp data.csv "/content/drive/MyDrive/Research_Project"

X = ipl_first_innings.drop(labels=['total','id'], axis=1)
y = ipl_first_innings['total'].values

# split into train test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
    random_state=42,stratify=y)

X_train=X_train.values
X_test=X_test.values
X_train=np.asarray(X_train).astype(np.float32)
X_test=np.asarray(X_test).astype(np.float32)

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

"""
##### **Building model for second innings**"""
```

```
ipl_second_innings=ipl.copy()

# first innings total

ipl_second_innings['1st_innings_total']=ipl_second_innings.groupby(['id'])[
    'total'].transform('first')

ipl_second_innings=ipl_second_innings.loc[ipl_second_innings['inning'] ==
    2]

ipl_second_innings=pd.get_dummies(data=ipl_second_innings, columns=['
    batting_team', 'bowling_team'])

ipl_second_innings=ipl_second_innings[['id','inning','batting_team_CSK', '
    batting_team_DC','batting_team_KKR', 'batting_team_KXIP', '
    batting_team_MI',
                    'batting_team_RCB', 'batting_team_RR', 'batting_team_SRH
                        ','bowling_team_CSK', 'bowling_team_DC', '
                        bowling_team_KKR',
                   'bowling_team_KXIP', 'bowling_team_MI', 'bowling_team_RCB
                       ','bowling_team_RR', 'bowling_team_SRH',
                    'overs','total_score','total_wickets','prev_runs_5_overs
                        ','prev_wickets_5_overs','prev_5_overs_dot_balls','
                        prev_5_overs_boundaries','1st_innings_total','total'
                        ]]

ipl_second_innings
ipl_second_innings.to_csv('Second_Innings_Data.csv')
!cp data1.csv "/content/drive/MyDrive/Research_Project"

# Splitting the data into train and test set
X = ipl_second_innings.drop(labels=['total','id'], axis=1)
y = ipl_second_innings['total'].values

# split into train test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
    random_state=42,stratify=y)

X_train=X_train.values
X_test=X_test.values
X_train=np.asarray(X_train).astype(np.float32)
X_test=np.asarray(X_test).astype(np.float32)

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

Listing 7: Data Splitting

5.6 Extreme Gradient Boosting (XGBoost)

Below code demonstrates the hyperparameter tuning for the XGBoost models. Optimum parameters are obtained by running the randomised search cross-validation for each innings data. The best parameters are used for the model initialisation.

```
"""##### **XG-Boost Method**"""
```

```
## Hyper Parameter Optimization

params={
 "learning_rate"     : [0.05, 0.10, 0.15, 0.20, 0.25, 0.30 ] ,
 "max_depth"         : [ 3, 4, 5, 6, 8, 10, 12, 15],
 "min_child_weight" : [ 1, 3, 5, 7 ],
 "gamma"             : [ 0.0, 0.1, 0.2 , 0.3, 0.4 ],
 "colsample_bytree" : [ 0.3, 0.4, 0.5 , 0.7 ]

}

#Intialising model with default papramters

ipl_xgb = xgb.XGBRegressor(objective='reg:squarederror')

#To find optimum paramters

random_search=RandomizedSearchCV(ipl_xgb,param_distributions=params,n_iter
    =5,n_jobs=-1,cv=5,verbose=3)
random_search.fit(X_train, y_train)

random_search.best_estimator_

random_search.best_params_

#Intialising model with best parameters

#First Innings
#ipl_xgb = xgb.XGBRegressor(base_score=0.5, booster='gbtree',
    colsample_bylevel=1,
#              colsample_bynode=1, colsample_bytree=0.5, gamma=0.4,
#              importance_type='gain', learning_rate=0.3, max_delta_step=0,
#              max_depth=8, min_child_weight=1, missing=None, n_estimators
    =100,
#              n_jobs=1, nthread=None, objective='reg:squarederror',
#              random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight
    =1,
#              seed=None, silent=None, subsample=1, verbosity=1)

#Second Innings
ipl_xgb = xgb.XGBRegressor(base_score=0.5, booster='gbtree',
    colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=0.5, gamma=0.2,
              importance_type='gain', learning_rate=0.15, max_delta_step=0,
              max_depth=15, min_child_weight=7, missing=None, n_estimators
                  =100,
              n_jobs=1, nthread=None, objective='reg:squarederror',
              random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
              seed=None, silent=None, subsample=1, verbosity=1)
```

Listing 8: Hyperparameter Optimisation for XGBoost Model

Below code is used to train and evaluate the model.

```
#XGBoost model

ipl_xgb.fit(X_train, y_train)
```

```
#Accuracy of the model

print("Accuracy on training set: {:.3f}".format(ipl_xgb.score(X_train,
    y_train)))
print("Accuracy on validation set: {:.3f}".format(ipl_xgb.score(X_test,
    y_test)))

#Prediction of the model

ipl_xgb_predict=ipl_xgb.predict(X_test)
ipl_xgb_predict

#Model Evaluation

print("RMSE : %f" %(np.sqrt(MSE(y_test, ipl_xgb_predict))))
print("MAE : %f" %(mean_absolute_error(y_test, ipl_xgb_predict)))
print("Variance explained : %f" %(explained_variance_score(y_test,
    ipl_xgb_predict)))

#Residual plot

residuals = y_test-ipl_xgb_predict
plt.scatter(residuals,ipl_xgb_predict)
plt.title('Residual Plot')
plt.xlabel('Residuals')
plt.ylabel('Predicted Score')
plt.show()
```

Listing 9: Training and Evaluation of XGBoost Model

5.7 Adaptive Boosting (AdaBoost)

Below code demonstrates the hyperparameter tuning for the AdaBoost models. Optimum parameters are obtained by running the randomized search cross validation for each innings data. The best parameters are used for the model initialisation.

```
"""##### **Ada Boost Method**"""

#Hyper parameter optimization

params={
 "n_estimators"     : [20,40,60,80,100] ,
 "learning_rate"    : [0.01,0.05,0.1,0.3,0.5,1]
}

#Intialising base learner model

#First Innings
#bl_xgb = xgb.XGBRegressor(base_score=0.5, booster='gbtree',
    colsample_bylevel=1,
#              colsample_bynode=1, colsample_bytree=0.5, gamma=0.4,
#              importance_type='gain', learning_rate=0.3, max_delta_step=0,
#              max_depth=8, min_child_weight=1, missing=None, n_estimators
    =100,
#              n_jobs=1, nthread=None, objective='reg:squarederror',
#              random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight
    =1,
```

```
#                seed=None, silent=None, subsample=1, verbosity=1)

#Second Innings
bl_xgb = xgb.XGBRegressor(base_score=0.5, booster='gbtree',
    colsample_bylevel=1,
                colsample_bynode=1, colsample_bytree=0.3, gamma=0.3,
                importance_type='gain', learning_rate=0.2, max_delta_step=0,
                max_depth=15, min_child_weight=1, missing=None, n_estimators
                    =100,
                n_jobs=1, nthread=None, objective='reg:squarederror',
                random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
                seed=None, silent=None, subsample=1, verbosity=1)

ipl_ada = AdaBoostRegressor(base_estimator=bl_xgb)

#To find optimum paramters

random_search=RandomizedSearchCV(ipl_ada,param_distributions=params,n_iter
    =5,scoring = 'neg_mean_absolute_error',n_jobs=-1,cv=2
                                     ,verbose=3)
random_search.fit(X_train, y_train)

random_search.best_estimator_

random_search.best_params_

#Intialising model with best parameters

#First Innings
#ipl_ada = AdaBoostRegressor(base_estimator=bl_xgb, learning_rate=0.05,
    loss='linear',
#                    n_estimators=90, random_state=None)

#Second Innings
ipl_ada = AdaBoostRegressor(base_estimator=bl_xgb, learning_rate=0.05, loss
    ='linear',
                    n_estimators=100, random_state=None)
```

Listing 10: Hyperparameter Optimisation for AdaBoost Model

Below code is used to train and evaluate the model.

```
#AdaBoost Model

ipl_ada.fit(X_train, y_train)

#Accuracy of the model

print("Accuracy on training set: {:.3f}".format(ipl_ada.score(X_train,
    y_train)))
print("Accuracy on validation set: {:.3f}".format(ipl_ada.score(X_test,
    y_test)))

#Prediction of the model

ipl_ada_predict=ipl_ada.predict(X_test)
ipl_ada_predict
```

```
#Model Evaluation

print("RMSE : %f" %(np.sqrt(MSE(y_test, ipl_ada_predict))))
print("MAE : %f" %(mean_absolute_error(y_test, ipl_ada_predict)))
print("Variance explained : %f" %(explained_variance_score(y_test,
    ipl_ada_predict)))

#Residual plot

residuals = y_test-ipl_ada_predict
plt.scatter(residuals,ipl_ada_predict)
plt.title('Residual Plot')
plt.xlabel('Residuals')
plt.ylabel('Predicted Score')
plt.show()
```

Listing 11: Training and Evaluation of AdaBoost Model

5.8 Random Forest

Below demonstrates the hyperparameter tuning for the random forest models. Optimum parameters are obtained by running the randomized search cross validation for each innings data. The best parameters are used for the model initialisation.

```
"""##### **Random Forest Method**



"""

# Hyper parameter tuning
params=[{'n_estimators':[20,30,40,60,100],
              'max_depth':  [5,10,15,20]}]

#Create the base model to tune
rf = RandomForestRegressor()

#To find optimum paramters

random_search = RandomizedSearchCV(estimator = rf, param_distributions =
    params, cv = 5, verbose=3, n_jobs = -1)
random_search.fit(X_train, y_train)

random_search.best_params_

#Intialising model with best parameters

#First Innings
#ipl_rf = RandomForestRegressor(n_estimators=60, max_depth=20, random_state
    =0)

#Second Innings
ipl_rf = RandomForestRegressor(n_estimators=100, max_depth=20, random_state
    =0)
```

Listing 12: Hyperparameter Optimisation for Random forest Model

Below code is used to train and evaluate the model.

```
#Random Forest Model

ipl_rf.fit(X_train,y_train)

#Accuracy of the model

print("Accuracy on training set: {:.3f}".format(ipl_rf.score(X_train,
    y_train)))
print("Accuracy on validation set: {:.3f}".format(ipl_rf.score(X_test,
    y_test)))

#Prediction of the model

ipl_rf_predict = ipl_rf.predict(X_test)
ipl_rf_predict

#Model Evaluation

print("RMSE : %f" %(np.sqrt(MSE(y_test, ipl_rf_predict))))
print("MAE : %f" %(mean_absolute_error(y_test, ipl_rf_predict)))
print("Variance explained : %f" %(explained_variance_score(y_test,
    ipl_rf_predict)))

#Residual plot

residuals_rf = y_test-ipl_rf_predict
plt.scatter(residuals_rf,ipl_rf_predict)
plt.title('Residual Plot')
plt.xlabel('Residuals')
plt.ylabel('Predicted Score')
plt.show()
```

Listing 13: Training and Evaluation of Random forest Model

5.9 Prediction

In the last phase of the implementation, we created a function to make test data to predict the score shown below.

```
"""##### **Machine Learning version of Duckworth Lewis method**

"""

# Function for score prediction :

def ml_dl(Bat_Team,Bowl_Team,overs,total_score,total_wickets,
    prev_runs_5_overs,prev_wickets_5_overs,prev_5_overs_dot_balls,
    prev_5_overs_boundaries,first_innings_total):

        one_hot_enc = list()

        if Bat_Team == 'CSK':
                one_hot_enc = one_hot_enc + [1,0,0,0,0,0,0,0]
        elif Bat_Team == 'DC':
```

```
            one_hot_enc = one_hot_enc + [0,1,0,0,0,0,0,0]
        elif Bat_Team == 'KKR':
            one_hot_enc = one_hot_enc + [0,0,1,0,0,0,0,0]
        elif Bat_Team == 'KXIP':
            one_hot_enc = one_hot_enc + [0,0,0,1,0,0,0,0]
        elif Bat_Team == 'MI':
            one_hot_enc = one_hot_enc + [0,0,0,0,1,0,0,0]
        elif Bat_Team == 'RCB':
            one_hot_enc = one_hot_enc + [0,0,0,0,0,1,0,0]
        elif Bat_Team == 'RR':
            one_hot_enc = one_hot_enc + [0,0,0,0,0,0,1,0]
        elif Bat_Team == 'SRH':
            one_hot_enc = one_hot_enc + [0,0,0,0,0,0,0,1]

        if Bowl_Team == 'CSK':
            one_hot_enc = one_hot_enc + [1,0,0,0,0,0,0,0]
        elif Bowl_Team == 'DC':
            one_hot_enc = one_hot_enc + [0,1,0,0,0,0,0,0]
        elif Bowl_Team == 'KKR':
            one_hot_enc = one_hot_enc + [0,0,1,0,0,0,0,0]
        elif Bowl_Team == 'KXIP':
            one_hot_enc = one_hot_enc + [0,0,0,1,0,0,0,0]
        elif Bowl_Team == 'MI':
            one_hot_enc = one_hot_enc + [0,0,0,0,1,0,0,0]
        elif Bowl_Team == 'RCB':
            one_hot_enc = one_hot_enc + [0,0,0,0,0,1,0,0]
        elif Bowl_Team == 'RR':
            one_hot_enc = one_hot_enc + [0,0,0,0,0,0,1,0]
        elif Bowl_Team == 'SRH':
            one_hot_enc = one_hot_enc + [0,0,0,0,0,0,0,1]
            ,
        one_hot_enc =[2] + one_hot_enc + [overs,total_score,total_wickets,
            prev_runs_5_overs,prev_wickets_5_overs,prev_5_overs_dot_balls,
            prev_5_overs_boundaries,first_innings_total]

        data = np.array([one_hot_enc])

        ml_dl_score = int(ipl_rf.predict(data))

        print('Machine Learning Duckworth Lewis Score',ml_dl_score)
```

Listing 14: Function to create test data

Feeding match details to test the model

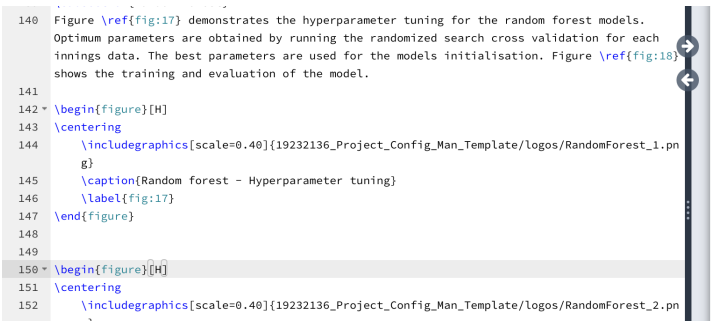
```
#Match Details before rain interruption
Bat_Team = 'SRH' # CSK,DC,KKR,KXIP,MI,RCB,RR,SRH

Bowl_Team = 'DC' # CSK,DC,KKR,KXIP,MI,RCB,RR,SRH

overs = 13.6

total_score = 104 # current score

total_wickets = 3 # current wicket

prev_runs_5_overs = 32 # runs in last 5 overs
```

```
prev_wickets_5_overs = 1 # wickets in last 5 overs

prev_5_overs_dot_balls = 10 # dots in last 5 overs

prev_5_overs_boundaries = 3 # boundaries in last 5 overs

first_innings_total = 159

#DL Method - SRH won by 7 runs
#Machine Learning DL method - SRH Lost by 3 runs
#Actual Result - Match was a tie
#Match Details - https://www.espncricinfo.com/series/ipl-2021-1249214/
    sunrisers-hyderabad-vs-delhi-capitals-20th-match-1254077/full-scorecard

#XGBoost- 156 runs
#AdaBoost - 155 runs
#Random Forest 158
```

Listing 15: Feeding actual match details to all models

6 Other Softwares

Overleaf an online latex editor was used to prepare the report for this research.



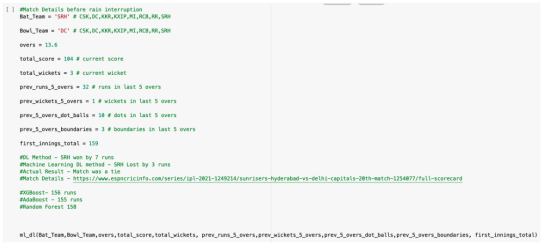


Figure 20: Prediction using a test case

Figure 5: Overleaf- Online Latex Editor

References