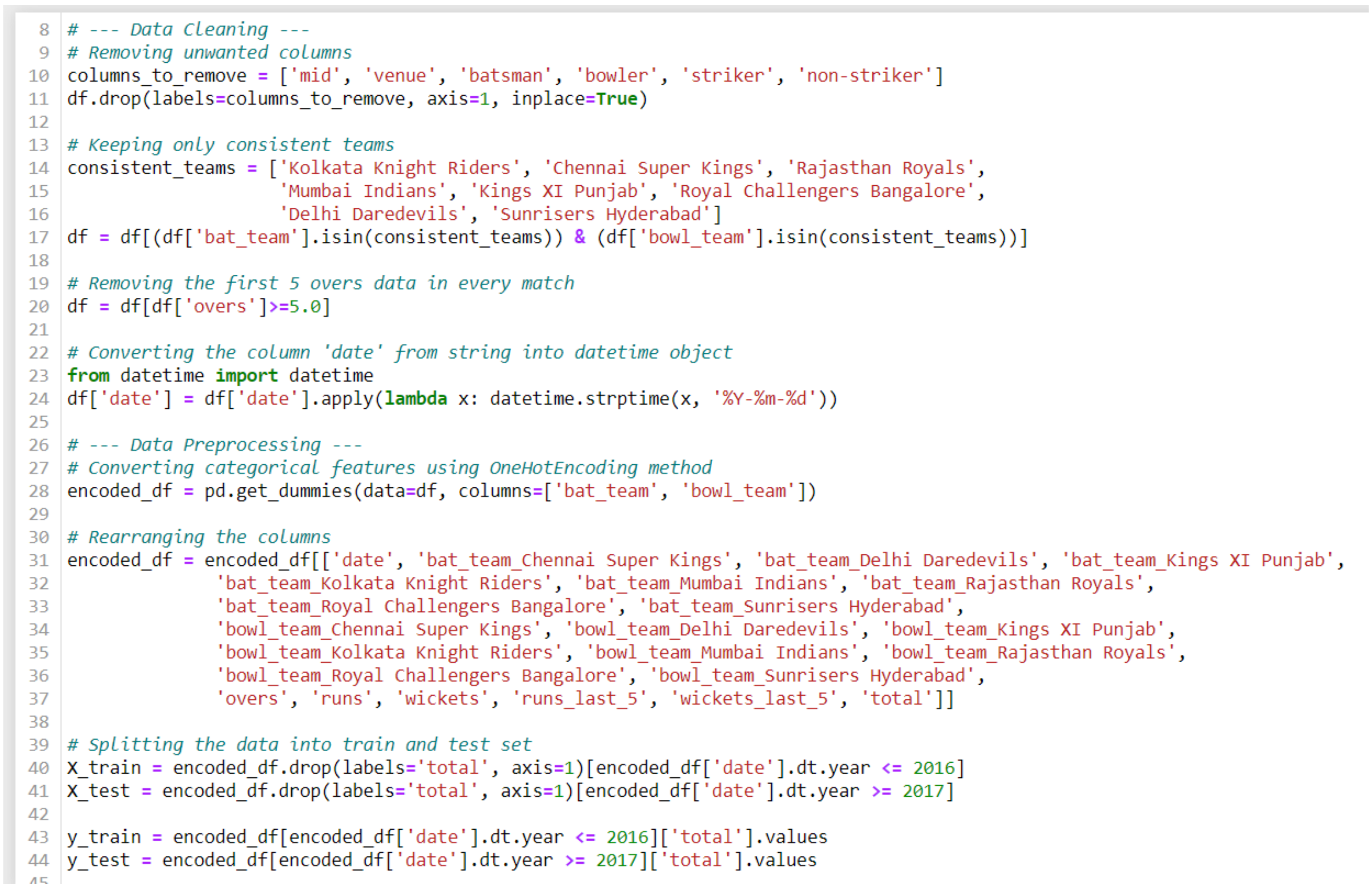
Project Name: End-To-End ML with Deployment Cricket-IPL-Score-Prediction (FileUse) – Linear Regression - Regression

Table of Contents

Demo



```python
# --- Data Cleaning ---
# Removing unwanted columns
columns_to_remove = ['mid', 'venue', 'batsman', 'bowler', 'striker', 'non-striker']
df.drop(labels=columns_to_remove, axis=1, inplace=True)

# Keeping only consistent teams
consistent_teams = ['Kolkata Knight Riders', 'Chennai Super Kings', 'Rajasthan Royals',
                    'Mumbai Indians', 'Kings XI Punjab', 'Royal Challengers Bangalore',
                    'Delhi Daredevils', 'Sunrisers Hyderabad']
df = df[(df['bat_team'].isin(consistent_teams)) & (df['bowl_team'].isin(consistent_teams))]

# Removing the first 5 overs data in every match
df = df[df['overs']>=5.0]

# Converting the column 'date' from string into datetime object
from datetime import datetime
df['date'] = df['date'].apply(lambda x: datetime.strptime(x, '%Y-%m-%d'))

# --- Data Preprocessing ---
# Converting categorical features using OneHotEncoding method
encoded_df = pd.get_dummies(data=df, columns=['bat_team', 'bowl_team'])

# Rearranging the columns
encoded_df = encoded_df[['date', 'bat_team_Chennai Super Kings', 'bat_team_Delhi Daredevils', 'bat_team_Kings XI Punjab',
              'bat_team_Kolkata Knight Riders', 'bat_team_Mumbai Indians', 'bat_team_Rajasthan Royals',
              'bat_team_Royal Challengers Bangalore', 'bat_team_Sunrisers Hyderabad',
              'bowl_team_Chennai Super Kings', 'bowl_team_Delhi Daredevils', 'bowl_team_Kings XI Punjab',
              'bowl_team_Kolkata Knight Riders', 'bowl_team_Mumbai Indians', 'bowl_team_Rajasthan Royals',
              'bowl_team_Royal Challengers Bangalore', 'bowl_team_Sunrisers Hyderabad',
              'overs', 'runs', 'wickets', 'runs_last_5', 'wickets_last_5', 'total']]

# Splitting the data into train and test set
X_train = encoded_df.drop(labels='total', axis=1)[encoded_df['date'].dt.year <= 2016]
X_test = encoded_df.drop(labels='total', axis=1)[encoded_df['date'].dt.year >= 2017]

y_train = encoded_df[encoded_df['date'].dt.year <= 2016]['total'].values
y_test = encoded_df[encoded_df['date'].dt.year >= 2017]['total'].values
```

Live Webapp Demo Link

Deployment on Heroku: http://mlcricketscoreprediction.herokuapp.com/

Abstract

The purpose of this report will be to use the IPL.csv Data to predict the score of cricket match.
This can be used to gain insight into how and why cricket score varies depending on the wickets and runs etc.
This can also be used as a model to gain a marketing advantage, by advertisement targeting those who are more likely to win cricket match and which team has more chances to lose.
Score Prediction is a regression problem, where using the match data and model building will predict what range of score a team can achieve.

This is diving into Prediction of Cricket Score through Machine Learning Concept.
End to End Project means that it is step by step process, starts with data collection, EDA, Data Preparation which includes cleaning and transforming then selecting, training and saving ML Models, Cross-validation and Hyper-Parameter Tuning and developing web service then Deployment.
This repository contains the code for Predicting Cricket Score using python's various libraries.
It used pandas, sklearn, datetime and pickle libraries.
These libraries help to perform individually one particular functionality.
Pandas objects rely heavily on Numpy objects.
Sklearn has 100 to 200 models.
"Pickling" is the process whereby a Python object hierarchy is converted into a byte stream.
Datetime module supplies classes to work with date and time.

The purpose of creating this repository is to gain insights into Complete ML Project.
These python libraries raised knowledge in discovering these libraries with practical use of it.
It leads to growth in my ML repository.
This above screenshot and Video_File Folder will help you to understand flow of output.

Motivation

The reason behind building this is, because till now I have worked on individual concepts so I wanted to combine all things that I have learnt till now and create a End to End Project that shows whole life cycle of ML Project. In addition to that, as an employee of a company, I should be able to carry out an entire process own is also essential aspect of it. Building end to end project is gave me wholesome approach to handle given data. Hence, I continue to gain knowledge while practicing the same and spread literary wings in tech-heaven.

Acknowledgement

Dataset Available: https://www.kaggle.com/yuvrajdagur/ipl-dataset-season-2008-to-2017

The Data

- Basic Statistics

Here are all the features included in the data set, and a short description of them all.

ColumnName	Description
mid	Unique match id.
date	Date on which the match was played.
venue	Stadium where match was played.
battingteam	Batting team name.
bowlingteam	Bowling team name.
batsman	Batsman who faced that particular ball.
bowler	Bowler who bowled that particular ball.
runs	Runs scored by team till that point of instance.
wickets	Number of Wickets fallen of the team till that point of instance.
overs	Number of Overs bowled till that point of instance.
runslast5	Runs scored in previous 5 overs.
wicketslast5	Number of Wickets that fell in previous 5 overs.
striker	max(runs scored by striker, runs scored by non-striker).
non-striker	min(runs scored by striker, runs scored by non-striker).
total	Total runs scored by batting team at the end of first innings.

It has 76014 total observations and 15 columns.

```
df.shape
```

```
(76014, 15)
```

This data has been cleaned of any null values.

```
df.isnull().sum()
```

```
mid               0
date              0
venue             0
bat_team          0
bowl_team         0
batsman           0
bowler            0
runs              0
wickets           0
overs             0
runs_last_5       0
wickets_last_5    0
striker           0
non-striker       0
total             0
dtype: int64
```

```
df.head()
```

	mid	date	venue	bat_team	bowl_team	batsman	bowler	runs	wickets	overs	runs_last_5	wickets_last_5	striker	non-striker	total
0	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	SC Ganguly	P Kumar	1	0	0.1	1	0	0	0	222
1	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	1	0	0.2	1	0	0	0	222
2	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	2	0	0.2	2	0	0	0	222
3	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	2	0	0.3	2	0	0	0	222
4	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	2	0	0.4	2	0	0	0	222

Analysis of the Data

Let’s start by doing a general analysis of the data as a whole, including all the features the Linear Regression algorithm will be using.

- Basic Statistics

```
df.describe()
```

```
                mid          runs       wickets         overs    runs_last_5  wickets_last_5       striker   non-striker         total
count  76014.000000  76014.000000  76014.000000  76014.000000  76014.000000    76014.000000  76014.000000  76014.000000  76014.000000
mean     308.627740     74.889349      2.415844      9.783068     33.216434        1.120307     24.962283      8.869287    160.901452
std      178.156878     48.823327      2.015207      5.772587     14.914174        1.053343     20.079752     10.795742     29.246231
min        1.000000      0.000000      0.000000      0.000000      0.000000        0.000000      0.000000      0.000000     67.000000
25%      154.000000     34.000000      1.000000      4.600000     24.000000        0.000000     10.000000      1.000000    142.000000
50%      308.000000     70.000000      2.000000      9.600000     34.000000        1.000000     20.000000      5.000000    162.000000
75%      463.000000    111.000000      4.000000     14.600000     43.000000        2.000000     35.000000     13.000000    181.000000
max      617.000000    263.000000     10.000000     19.600000    113.000000        7.000000    175.000000    109.000000    263.000000
```

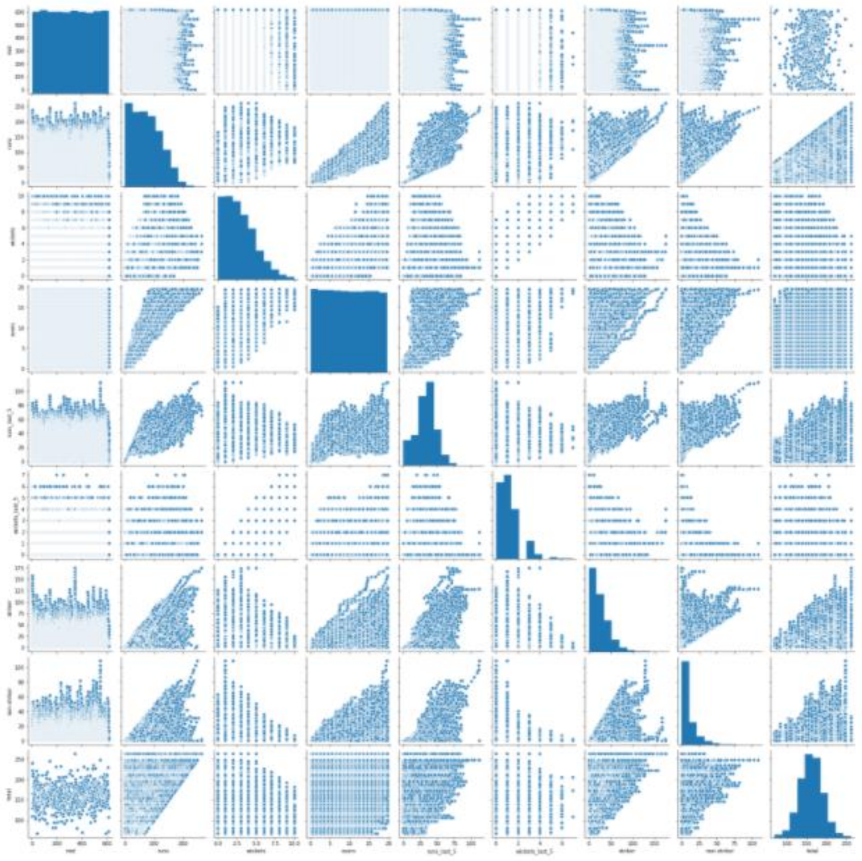
- Graphing of Features

Graph Set 1

```
In [8]: import seaborn as sns

In [9]: sns.pairplot(df)

Out[9]: <seaborn.axisgrid.PairGrid at 0x1adedb75af0>
```



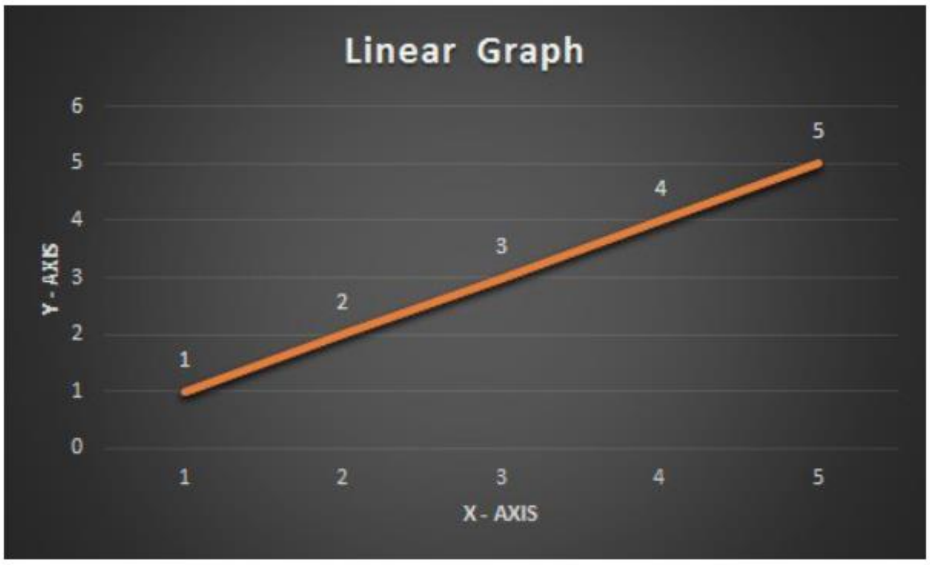
- Math behind the metrics

Linear Regression is a predictive algorithm which provides a Linear relationship between Prediction (Call it 'Y') and Input (Call is 'X').

As we know from the basic maths that if we plot an 'X','Y' graph, a linear relationship will always come up with a straight line. For example, if we plot the graph of these values.

```
(Input) X = 1,2,3,4,5
(Prediction) Y = 1,2,3,4,5
```

It will be a perfectly straight line



Linear Straight Line graph

Equation of Straight Line from 2 Points

The equation of a straight line is written using the `y = mx + b`, where `m` is the slope (Gradient) and `b` is y-intercept (where the line crosses the Y axis).

Once we get the equation of a straight line from 2 points in space in `y = mx + b` format, we can use the same equation to predict the points at different values of `x` which result in a straight line.

In this formula, `m` is the slope and `b` is y-intercept.

> *Linear regression is a way to predict the `'Y'` values for unknown values of Input `'X'` like `1.5, 0.4, 3.6, 5.7` and even for `-1, -5, 10` etc.*

Linear Regression Formula Analyses

The equation is given by:

y=a+bx

a and b can be computed by the following formulas:

b = $\frac{n\sum xy-(\sum x)(\sum y)}{n\sum x^2-(\sum x)^2}$

a = $\frac{\sum y-b(\sum x)}{n}$

Where,

x and y are the variables for which we will make regression line.

- b = Slope of the line.
- a = Y-intercept of the line.
- X = Values of the first data set.
- Y = Values of the second data set.

Q.1: Find out the linear regression equation for the following sets of data:

X	2	3	5	8
Y	3	6	5	12

Solution:

X	Y	X^2	XY
2	3	4	6
3	6	9	18
5	5	25	25
8	12	64	96
$\sum X = 18$	$\sum Y = 26$	$\sum X^2 = 102$	$\sum XY = 145$

Now we will apply the formula to get values of a and b, as follows:

$$b = \frac{n\sum xy - (\sum x)(\sum y)}{n\sum x^2 - (\sum x)^2}$$

$$b = \frac{4\times145 - 18\times26}{4\times102 - 324}$$

$$b = \frac{112}{84}$$

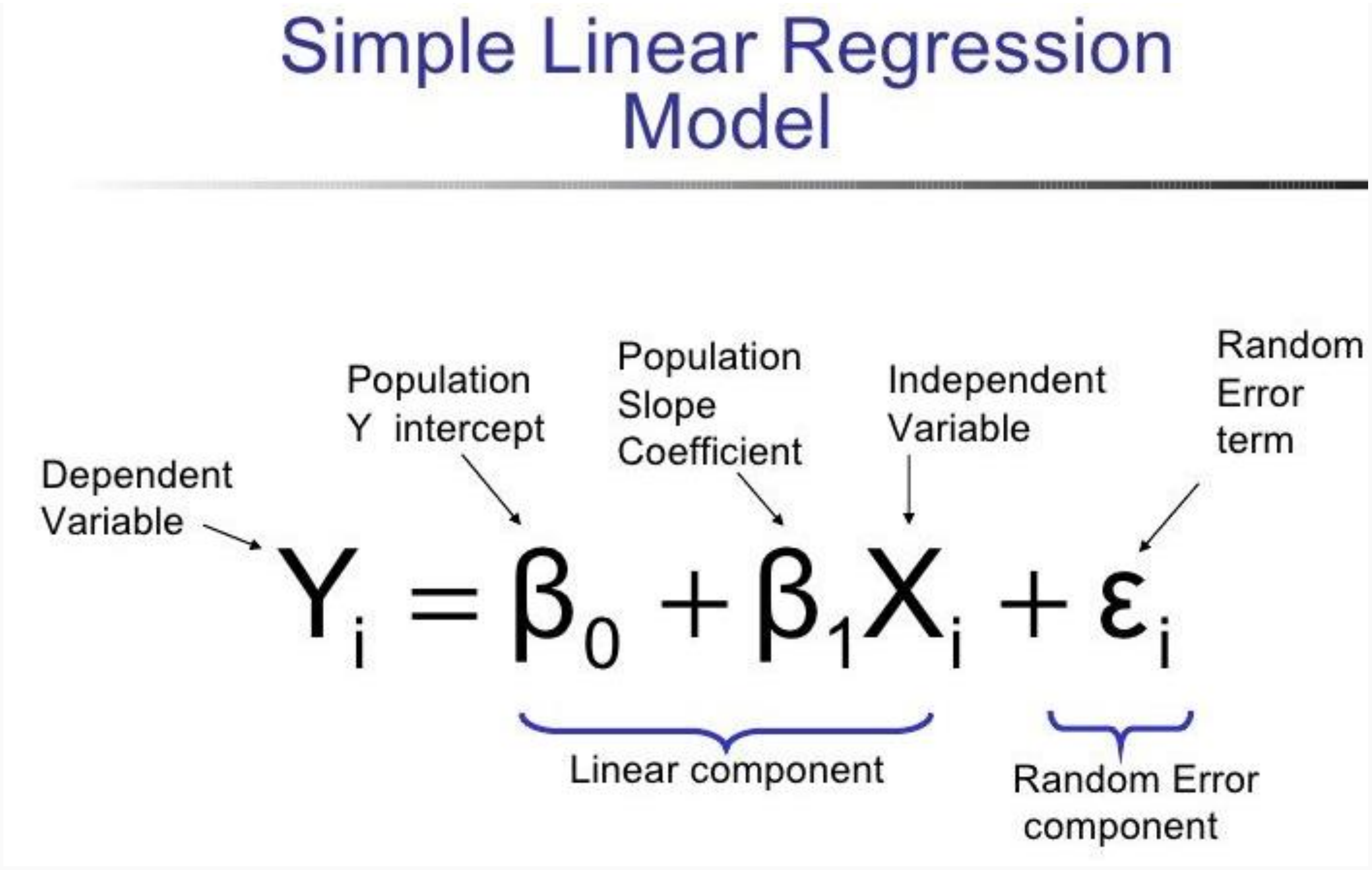
b = 1.33

$$a = \frac{\sum y - b(\sum x)}{n}$$
$$= \frac{26 - 1.33\times18}{4}$$

a = 0.515

Linear regression equation is given by:

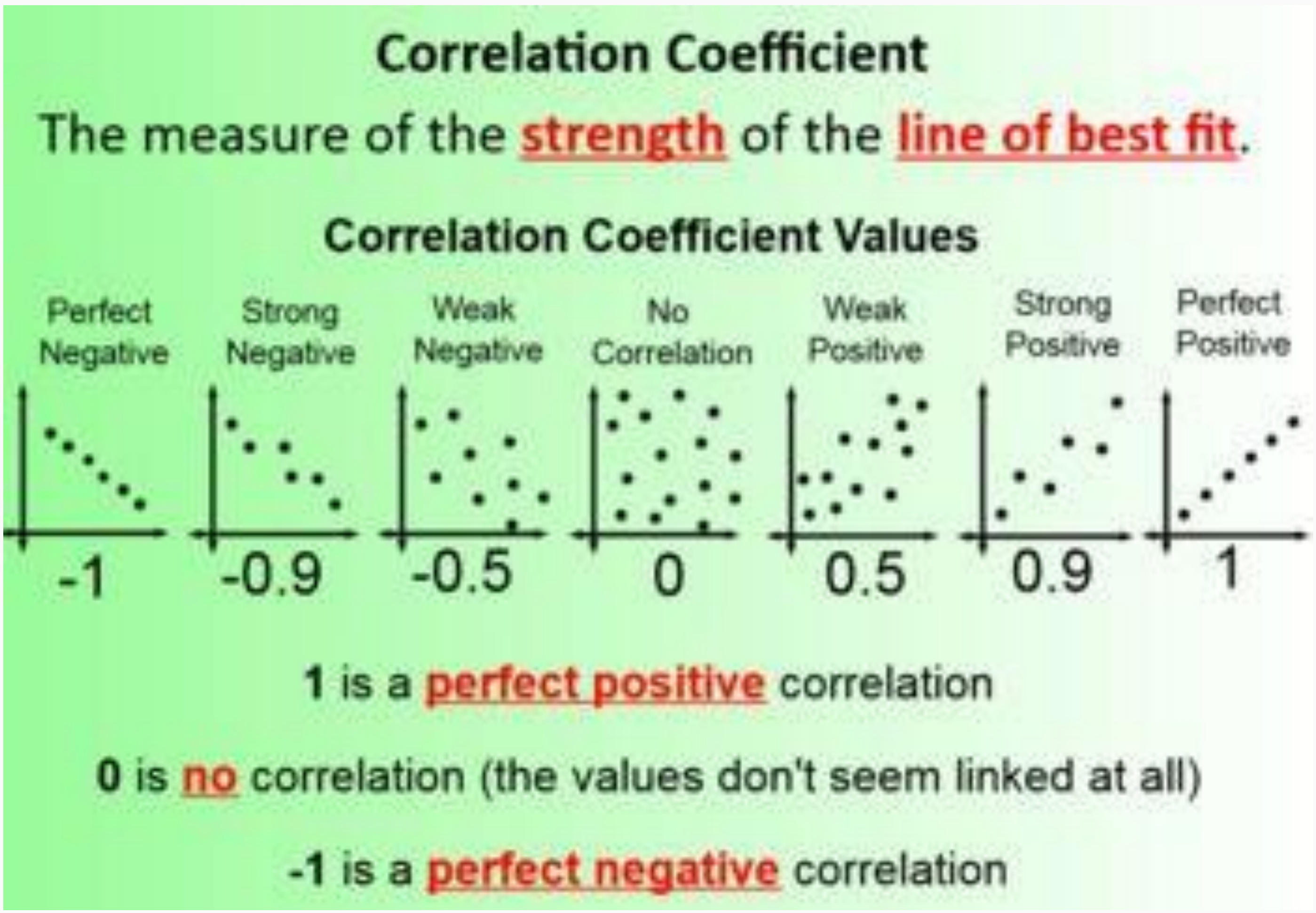
y = 0.515 + 1.33x



Simple Linear Regression Model

$$\underbrace{Y_i}_{\text{Dependent Variable}} = \underbrace{\beta_0 + \beta_1 X_i}_{\text{Linear component}} + \underbrace{\varepsilon_i}_{\text{Random Error component}}$$

- Model Architecture Process Through Visualization



Correlation Coefficient

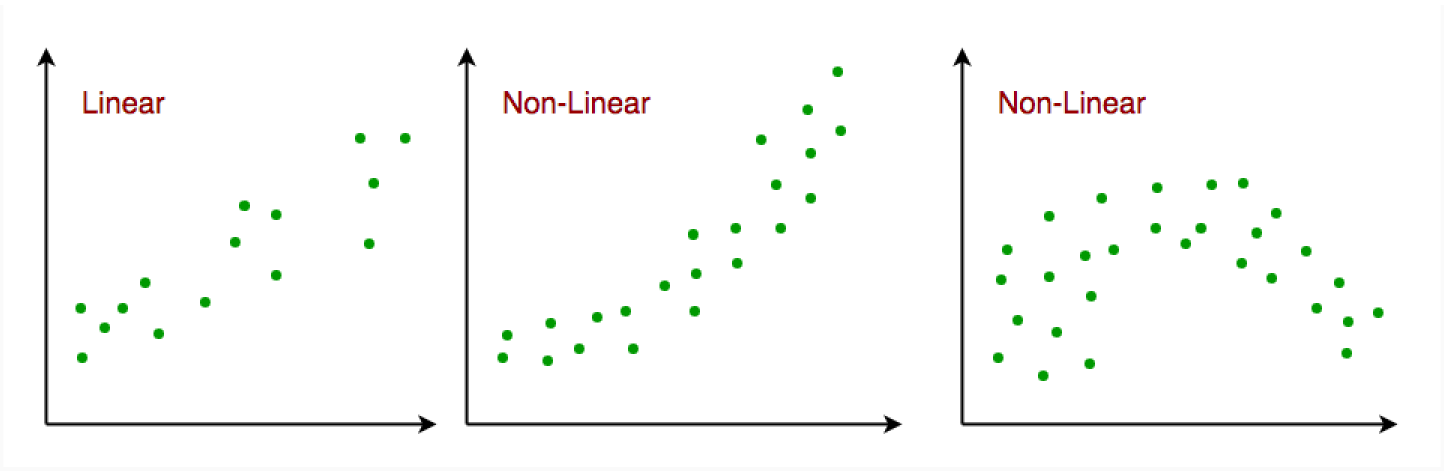
The measure of the strength of the line of best fit.

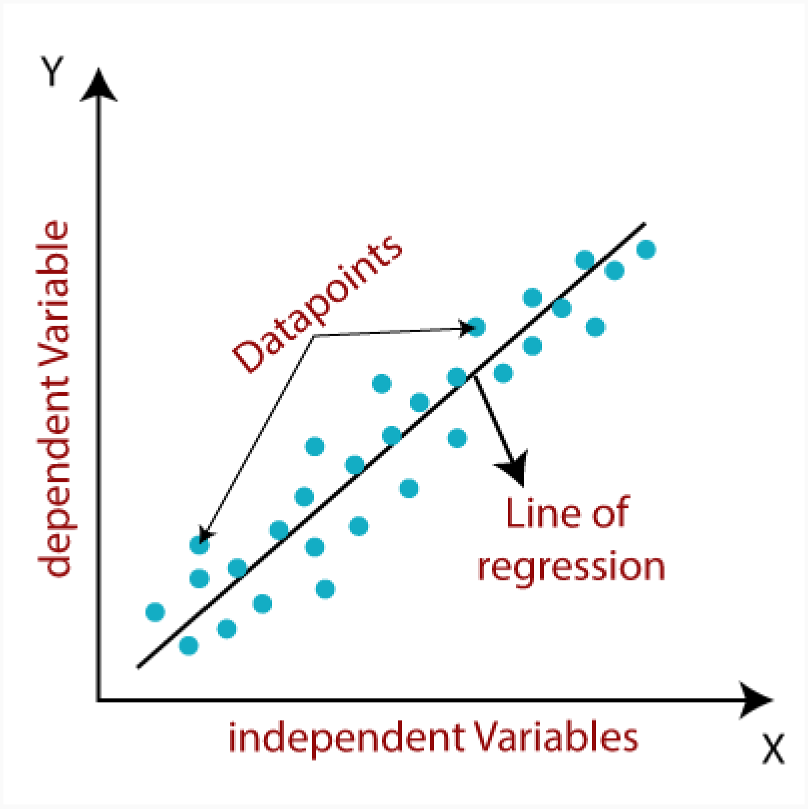
Correlation Coefficient Values

1 is a perfect positive correlation

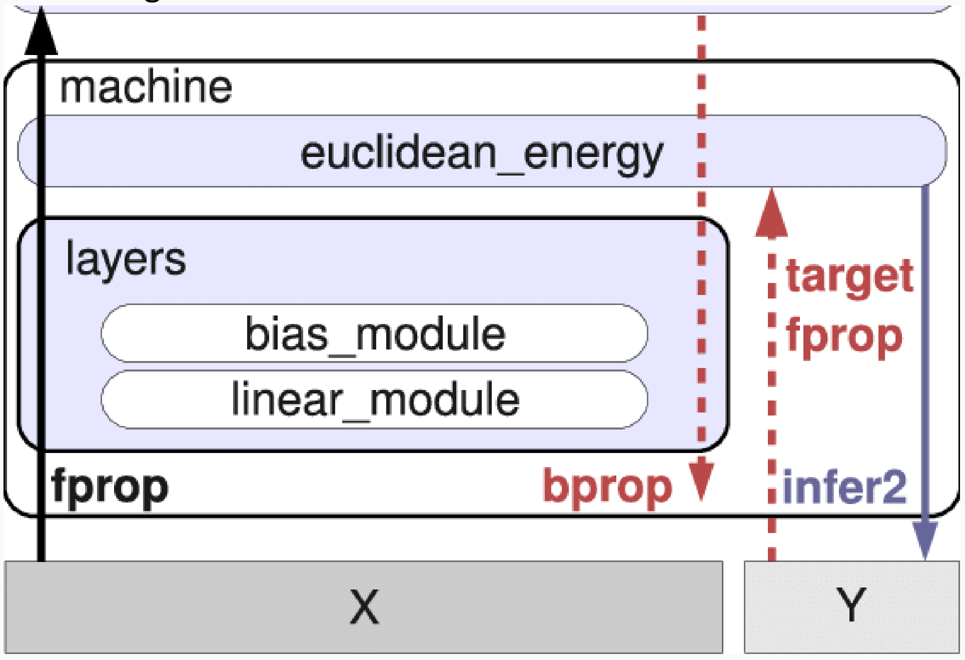
0 is no correlation (the values don't seem linked at all)

-1 is a perfect negative correlation





Linear Regression Architecture:



- Quick Notes

Step 1: Imported essential libraries.
Step 2: Loaded the dataset.
Step 3: Performed Data Cleaning.

- Removed unwanted columns.
- Kept only consistent teams.
- Removed first 5 overs data in every match.
- Converted column ‘date’ from string into datetime object.

Step 4: Performed Data Pre-processing.

- Converted categorical features using OneHotEncoding Method that is .get_dummies() function.
- Re-arranged the columns.

Step 5: Splitted the data.

- Removed ‘date’ column.

Step 6: Built the model.

- Fitted the data on Linear Regression Model.

Step 7: Saved the model as pickle file to re-use it again.
Step 8: Created Web App.

- The Model Analysis

Imported essential libraries – When we import modules, we're able to call functions that are not built into Python. Some modules are installed as part of Python, and some we will install through pip. Making use of modules allows us to make our programs more robust and powerful as we're leveraging existing code.

Loaded and read dataset – You can import tabular data from CSV files into pandas data frame by specifying a parameter value for the file.

Performed Data Cleaning - Data cleaning is the process of ensuring data is correct, consistent and usable. You can clean data by identifying errors or corruptions, correcting or deleting them, or manually processing data as needed to prevent the same errors from occurring. Data cleansing is also important because it improves your data quality and in doing so, increases overall productivity. When you clean your data, all outdated or incorrect information is gone – leaving you with the highest quality information. First, removed unwanted columns. Then, I kept only consistent teams. Third, removed first 5 overs data in every match. Then, converted column ‘date’ from string into datetime object.

Performed Data Pre-processing - Whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. Data preprocessing is an important step to prepare the data to form a ML model. To make the process easier, data preprocessing is divided into four stages: data cleaning, data integration, data reduction, and data transformation. First, converted categorical features using OneHotEncoding Method that is .get_dummies() function. A one hot encoding allows the representation of categorical data to be more expressive. One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. There are two different ways to encoding categorical variables. One-hot encoding converts it into n variables, while dummy encoding converts it into n-1 variables. If we have k categorical variables, each of which has n values. Pandas has a function named get_dummies. It will convert your categorical string values into dummy variables. pd.get_dummies create a new dataframe containing unique values as columns which consists of zeros and ones. Secondly, I re-arranged the columns.

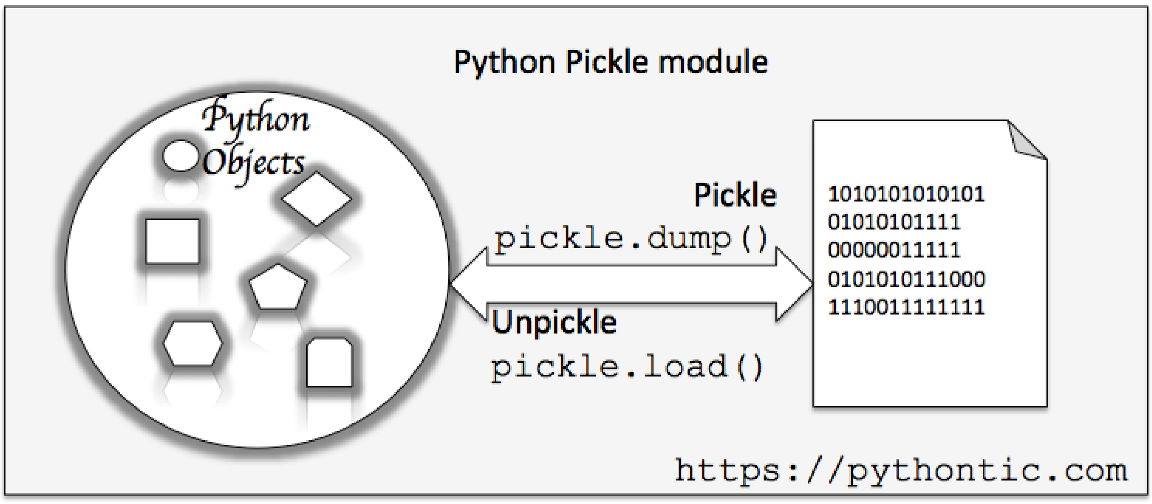
Splitted the data – Splitting data into train and test set in order to prediction w.r.t X_test. This helps in prediction after training data. Then, removed 'date' column.
Built the model - The fit() method takes the training data as arguments, which can be one array in the case of unsupervised learning, or two arrays in the case of supervised learning.
Fitted the data on Linear Regression Model.
Saved the model - as pickle file to re-use it again. Saving the created model for future reference and use so that we can directly access it rather than to go through full cycle again.
Created Web App - It is made in flask for end-users to use it.
Challenge that I faced in this project is, RF regression is time-consuming so more time in data pre-processing. In IPL, it is very difficult to predict the actual score because in a moment of time the game can completely turn upside down.

Creation of App

Here, I am creating Flask App. Loading the model that we saved then calling and calculating all the factors that are necessary in deciding score of matches.
Get the user entered value from the predict function and render respective .html page for solution.
You can create it with Streamlit also.

Technical Aspect

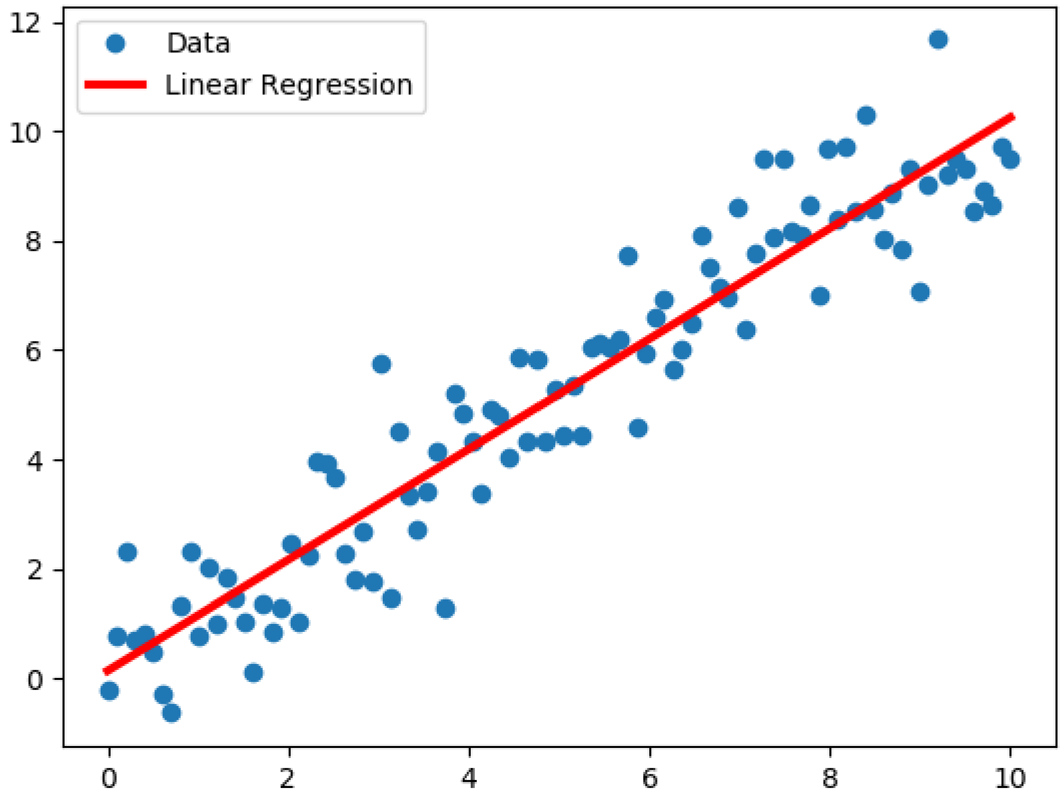
Pandas module mainly works with the tabular data. It contains Data Frame and Series. Pandas is 18 to 20 times slower than Numpy. Pandas is seriously a game changer when it comes to cleaning, transforming, manipulating and analyzing data.
Sklearn is known as scikit learn. It provides many ML libraries and algorithms for it. It provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python.
Pickle in Python is primarily used in serializing and deserializing a Python object structure. In other words, it's the process of converting a Python object into a byte stream to store it in a file/database, maintain program state across sessions, or transport data over the network.



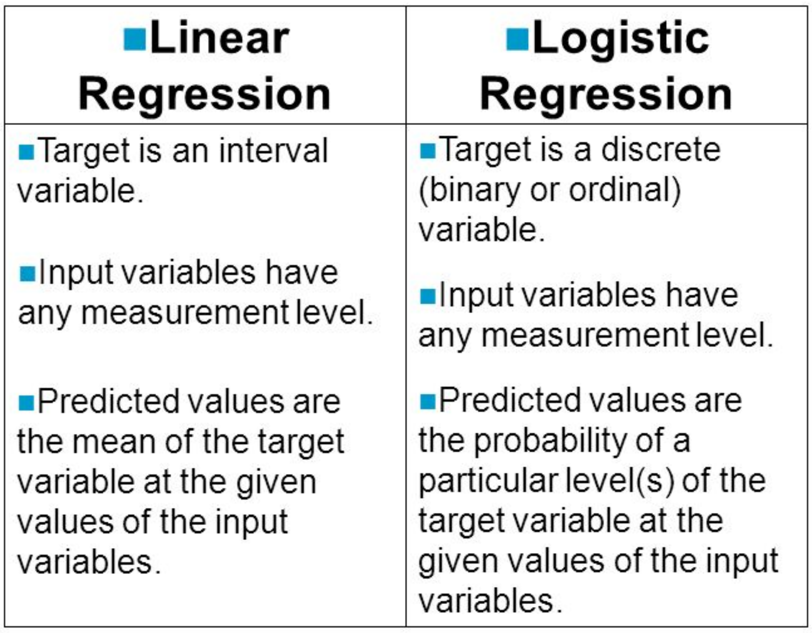
Need to train_test_split - Using the same dataset for both training and testing leaves room for miscalculations, thus increases the chances of inaccurate predictions.
The train_test_split function allows you to break a dataset with ease while pursuing an ideal model. Also, keep in mind that your model should not be overfitting or underfitting.
Date and datetime are an object in Python, so when you manipulate them, you are actually manipulating objects and not string or timestamps. Working with dates and times is one of the

biggest challenges in programming. Between dealing with time zones, daylight saving time, and different written date formats, it can be tough to keep track of which days and times you're referencing. Fortunately, the built-in Python datetime module can help you manage the complex nature of dates and times. For example, one great example of this irregularity is daylight saving time. In the United States and Canada, clocks are set forward by one hour on the second Sunday in March and set back by one hour on the first Sunday in November. However, this has only been the case since 2007. Prior to 2007, clocks were set forward on the first Sunday in April and set back on the last Sunday in October. Things get even more complicated when you consider time zones. Ideally, time zone boundaries would follow lines of longitude exactly.

Linear regression is the next step up after correlation. It is used when we want to predict the value of a variable based on the value of another variable. The variable we want to predict is called the dependent variable. Because the model is based on the equation of a straight line, y=a+bx, where a is the y-intercept (the value of y when x=0) and b is the slope (the degree to which y increases as x increases one unit). Linear regression plots a straight line through a y vs. x scatterplot. That is why it is call linear regression. Simple linear regression is a statistical method that allows us to summarize and study relationships between two continuous (quantitative) variables: One variable, denoted x, is regarded as the predictor, explanatory, or independent variable. The goal of multiple linear regression (MLR) is to model the linear relationship between the explanatory (independent) variables and response (dependent) variable. In essence, multiple regression is the extension of ordinary least-squares (OLS) regression that involves more than one explanatory variable.



Linear versus Logistic Regression



■Linear Regression	■Logistic Regression
■Target is an interval variable.	■Target is a discrete (binary or ordinal) variable.
■Input variables have any measurement level.	■Input variables have any measurement level.
■Predicted values are the mean of the target variable at the given values of the input variables.	■Predicted values are the probability of a particular level(s) of the target variable at the given values of the input variables.

Installation

Using intel core i5 9th generation with NVIDIA GFORECE GTX1650.
Windows 10 Environment Used.
Already Installed Anaconda Navigator for Python 3.x
The Code is written in Python 3.8.
If you don't have Python installed you can install Python from its official site.
If you are using a lower version of Python you can upgrade using the pip package, ensuring you have the latest version of pip, *python* -m pip install --*upgrade pip and press Enter*.

Run/How to Use/Steps

Keep your internet connection on while running or accessing files and throughout too.
Follow this when you want to perform from scratch.
Open Anaconda Prompt, Perform the following steps:

```
cd <PATH>
pip install pandas
pip install sklearn
pip install numpy
pip install scipy
pip install matplotlib
```

Note: If it shows error as ‘No Module Found’ , then install relevant module.
You can also create requirement.txt file as, pip freeze > requirements.txt
Create Virtual Environment:

```
conda create -n ipl python=3.6
y
conda activate ipl
cd <PATH-TO-FOLDER>
```

run .py files.

Paste URL to browser to check whether working locally or not.

Follow this when you want to just perform on local machine.
Download ZIP File.
Right-Click on ZIP file in download section and select Extract file option, which will unzip file.
Move unzip folder to desired folder/location be it D drive or desktop etc.
Open Anaconda Prompt, write cd <PATH> and press Enter.
eg: cd C:\Users\Monica\Desktop\Projects\Python Projects 1\ 23)End_To_End_Projects\ Project_6_ML_FileUse_EndToEnd_Cricket_IPL_FirstInningsScorePrediction\Project_ML_CricketScorePrediction

conda create -n ipl python=3.7
y
conda activate ipl
In Anconda Prompt, pip install -r requirements.txt to install all packages.
In Anconda Prompt, write python app.py and press Enter.
Paste URL to browser to check whether working locally or not.
Please be careful with spellings or numbers while typing filename and easier is just copy filename and then run it to avoid any silly errors.
Note: cd <PATH>
[Go to Folder where file is. Select the path from top and right-click and select copy option and paste it next to cd one space <path> and press enter, then you can access all files of that folder]
[cd means change directory]

Directory Tree/Structure of Project

Folder: 23)End_To_End_Projects > Project_6_ML_FileUse_EndToEnd_Cricket_IPL_FirstInningsScorePrediction > Project_ML_CricketScorePrediction

Model_Building.py

app.py

Procfile

ipl.csv

first-innings-score-lr-model.pkl

requirements.txt

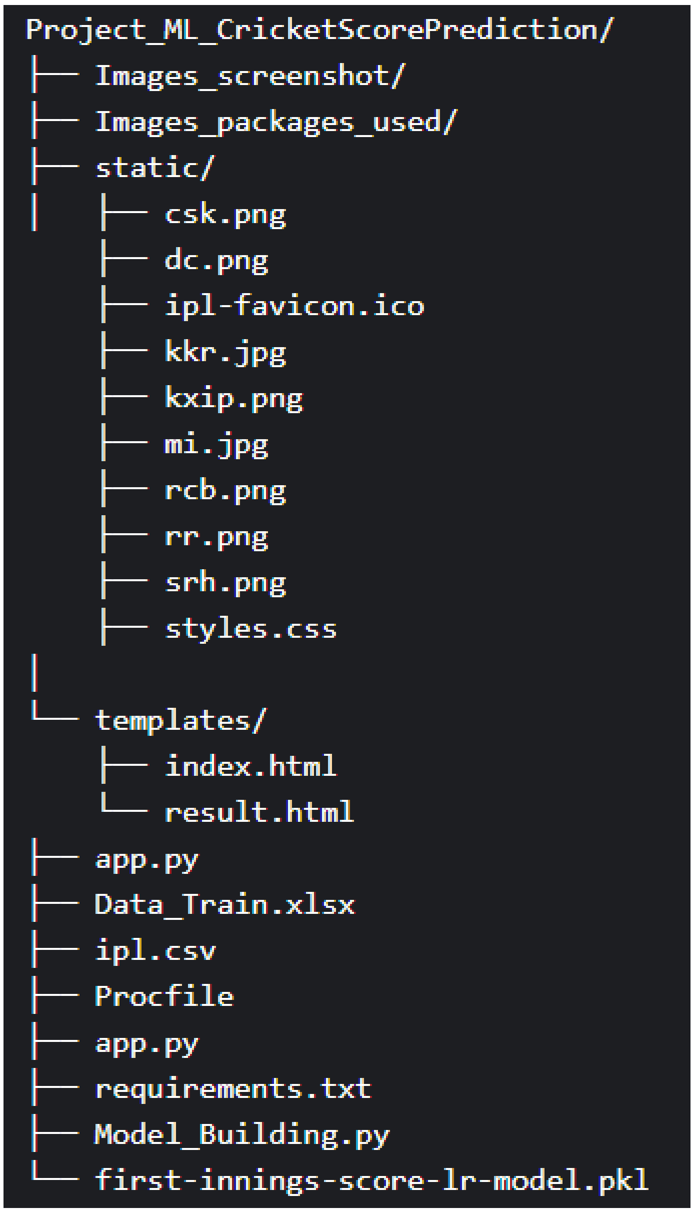
Folder: 23)End_To_End_Projects > Project_6_ML_FileUse_EndToEnd_Cricket_IPL_FirstInningsScorePrediction > Project_ML_CricketScorePrediction > templates

index.html

result.html

Folder: 23)End_To_End_Projects > Project_6_ML_FileUse_EndToEnd_Cricket_IPL_FirstInningsScorePrediction > Project_ML_CricketScorePrediction > static

styles.css, .png, .jpg files.


```
Project_ML_CricketScorePrediction/
├── Images_screenshot/
├── Images_packages_used/
├── static/
│    ├── csk.png
     ├── dc.png
     ├── ipl-favicon.ico
     ├── kkr.jpg
     ├── kxip.png
     ├── mi.jpg
     ├── rcb.png
     ├── rr.png
     ├── srh.png
     ├── styles.css
│
└── templates/
     ├── index.html
     └── result.html
├── app.py
├── Data_Train.xlsx
├── ipl.csv
├── Procfile
├── app.py
├── requirements.txt
├── Model_Building.py
└── first-innings-score-lr-model.pkl
```

To Do/Future Scope

Can create app for tennis.
Add columns in dataset of top batsmen and bowlers of all the teams.
Add columns that consists of striker and non-striker's strike rates.
Can deploy on AWS.

Technologies Used/System Requirements/Tech Stack

Download the Material

You can Download Dataset from here: https://github.com/monicadesAI-tech/Project_73/blob/main/ipl.csv

You can Download Entire Project from here: https://github.com/monicadesAI-tech/Project_73

You can Download Model Building from here: https://github.com/monicadesAI-tech/Project_73/blob/main/Model_Building.py

You can Download Web App_File from here: https://github.com/monicadesAI-tech/Project_73/blob/main/app.py

You can see Detailed Project at Website here: https://github.com/monicadesAI-tech.github.io/project/cricketipl.html

Download dependencies from here: https://github.com/monicadesAI-tech/Project_73/blob/main/requirements.txt

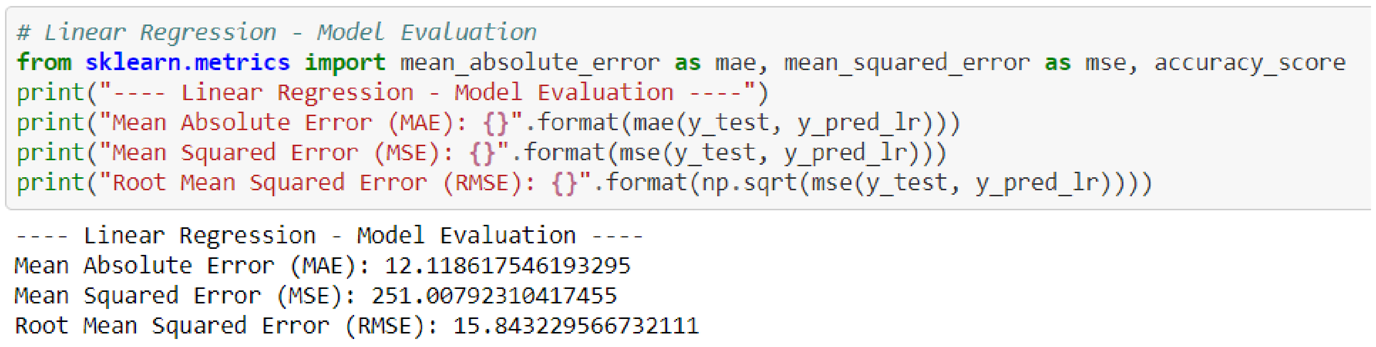
Download saved model from here: https://github.com/monicadesAI-tech/Project_73/blob/main/first-innings-score-lr-model.pkl

Conclusion

- Modelling

Using RF Regression may give better results.

- Analysis



```python
# Linear Regression - Model Evaluation
from sklearn.metrics import mean_absolute_error as mae, mean_squared_error as mse, accuracy_score
print("---- Linear Regression - Model Evaluation ----")
print("Mean Absolute Error (MAE): {}".format(mae(y_test, y_pred_lr)))
print("Mean Squared Error (MSE): {}".format(mse(y_test, y_pred_lr)))
print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(y_test, y_pred_lr))))
```

```
---- Linear Regression - Model Evaluation ----
Mean Absolute Error (MAE): 12.118617546193295
Mean Squared Error (MSE): 251.00792310417455
Root Mean Squared Error (RMSE): 15.843229566732111
```

As you can see RMSE is not extremely high however can still reduced it using RF.

Credits

Krish Naik Channel
https://machinelearningmastery.com/linear-regression-for-machine-learning/

Paper Citation

https://www.researchgate.net/publication/327904009_Predicting_Outcome_of_Indian_Premier_League_IPL_Matches_Using_Machine_Learning