**IPL SCORE PREDICTOR DOCUMENTATION**

BY

DEEPAK.K

Language used: Python

ML Algorithm used: Linear Regression

Dataset used: Ball-by-ball data of all IPL matches from 2008

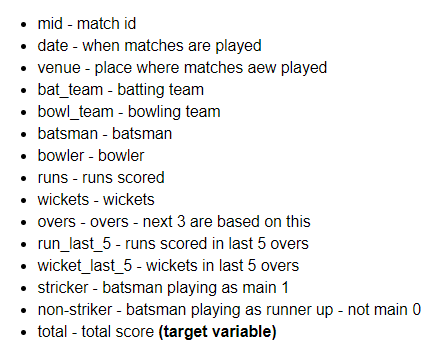
In this project, I have built an IPL score predictor in Python using numpy, pandas, pickle and sklearn and implemented the GUI in Gradio.

The project has 6 steps of execution:

* 1– Data Gathering
* 2– EDA – Exploratory Data Analysis
* 3– Data Cleaning
* 4- Data Preparation
* 5– Model Development
* 6- GUI Development

Step 1: **Data Collection/Gathering**

I am going to use the IPL Scores Dataset which has 76104 observations and 15 features as listed below:



The first thing to note here is that our dataset has certain columns which are irrelevant and increases resource consumption, so maybe we can remove them in future steps. The irrelevant columns are:

“mid,” “venue,” “batsman,” “bowler,” “striker,” and “non-striker.”

Step 2: **Exploratory Data Analysis(EDA)**

This procedure is very important and will allow us to understand the data and plan our next steps.

#### 2.1 Checking Shape

Even though we know the shape of data (observations and features) the info can be wrong. So it’s better to cross-check our data once. We use pandas for this with df=pd.read\_csv(‘ipl.csv’) and df.shape.

#### 2.2 Check Data Type

In real-world cases, the data we find are not of the desired data type, so it will be good if we check it too.

Pandas provide an info method that returns the data type of all columns present in the dataset.

#### 2.3 Check Null Values

Having understood the shape and info, we will now check for null values (fields having no data – NAN’s). It is essential for any project as null values can change the whole story depicted by data and can even contribute to making data worse for the use cases. Let’s perform the same

All we have to do is to use the is\_null method of the dataframe and then sum the outputs to get the total count for each column.

#### 2.4 Check Summary Statistics

Since there is no NaN value doesn’t mean our data is a good representation. To get the gist, we can plot summary measures which generally include:

1. Measurement of Central Tendency – These measures allow us to understand where most of our data lies and mostly include:

* **Mean**– Average of Data
* **Median**– Center for Data
* **Frequency**– No of occurrence of specific data.
* **Mode**– Highest Observation in Data

2. Measures of Dispersion – These are the measures that allow us to understand how widespread data is and mostly includes:

* **Max & Min**– Highest and Lowest Value in the dataset
* **Range**– Highest-Lowest (captures the reach of data)
* **Variance**– Captures the variation of data (how data is varying) – Usually the sum of deviation of actual data from its mean/no of samples – 1
* **Standard Deviation** – Same as standard deviation but on the same scale as data -sqrt (variance)
* **Percentiles**– Capture the spread of data for a specific value i.e how much data is above or below it. 50% – Median

Performing each check will be cumbersome so pandas pack all these in a single function called describe.

## Step 3: **Data Cleaning for IPL Score Predictor**

After understanding our data, we can now proceed to clean it for our use case. We will start by dropping a few columns so that it becomes easier to work.

#### 3.1 Removing Irrelevant Columns:

A careful inspection of our data states we have many irrelevant columns such as ‘mid’, ‘venue’, ‘batsman’, ‘bowler’, ‘striker’, ‘non-striker’.These do not contribute to data and can be removed to save memory space.

To remove the irrelevant column, all we will do is create a list of the irrelevant columns(**cols\_to\_remove**) and pass it to the panda’s **drop**method.

#### 3.2 Filtering Consistent Teams:

Next, we will filter out teams that are consistently playing. This will allow us to have a basic set of teams that are relevant to IPL’s.

**3.2.1. Find Unique Teams**

For that, we will first find the unique teams in bat\_team, create a list out of them and then perform our filtering. To find unique teams, we will just use the unique method over bat\_team.

3.2.2 **Creating List and Filtering Consistent Teams**

Now we will carve out a list with consistent teams and store it in consitent\_team. Finally, we can filter our required observations based on the condition “return only those observations for which a consistent team is present in both bat\_team and ball\_team”.

#### 3.3.3 Filtering Based On 5 Overs

**HYPOTHESIS:-** We can assume that “In most of the matches the actual game starts after elapsing 5 overs*”,*so it can serve as a good starting point for our training data.

So following our hypothesis. We will just return all the observations after 5 overs by first accessing the overs column of the dataframe(pdf) and using >= operator.

#### 3.3 Change Date Column Type

While examining the info on the features column, we found that the data column is an object data type on which no operations can be performed.

We will use the date-time library to convert the data column into a date-time object.

## Step 4: **Data Preparation for IPL Score Predictor**

Now, as evident from the info, our data takes a variability of data types including strings, data time, and numbers. But our model requires them all to be in a numeric format, so we should perform some operations to make it model-friendly.

#### 4.1 Encoding Categorical Variables:

On careful inspection, we can find that bat\_team and bowl\_team are categorical data and can be encoded as numbers(0/1). This is called ONE HOT ENCODING and can be achieved by pandas get\_dummy function. The columns have been increased and one-hot encoded. While it is taking up more space, that’s a trade-off to consider while encoding is up to individuals.

#### 4.2 Splitting Dataset:

As a general case, we split our data according to ratio train = 80% and test = 20% but here we will be learning how to adapt for a split for a problem domain. So instead of going for split size based on ratio, we will perform the split based on years. So all I did here is define the variables, drop the last column, and filter out the observation based on conditions that return only year values from the date column. For labels we only use a single column (total) which Panda assumes to be a series so we need to use values over it, else it will return an error sometimes.

#### 4.3 Dropping Date Column:

One last thing we can do before closing this section is to drop off the data column from our training and test sets, as it is of no use to us now as this will help free up some memory space for further operations using the same drop() method used earlier.

#### Step 5: Model Selection & Training:

The model architecture we choose will be a function that will find a way to map our training set to training labels thus allowing prediction of the score and given input data. Since we have seen that the data mostly have a linear relationship (as evident from the description) we will use a simple linear regression model for our use case from sklearn. First import the linear regression model from sklearn and train the model on our training data using reg.fit(X\_train, Y\_train).

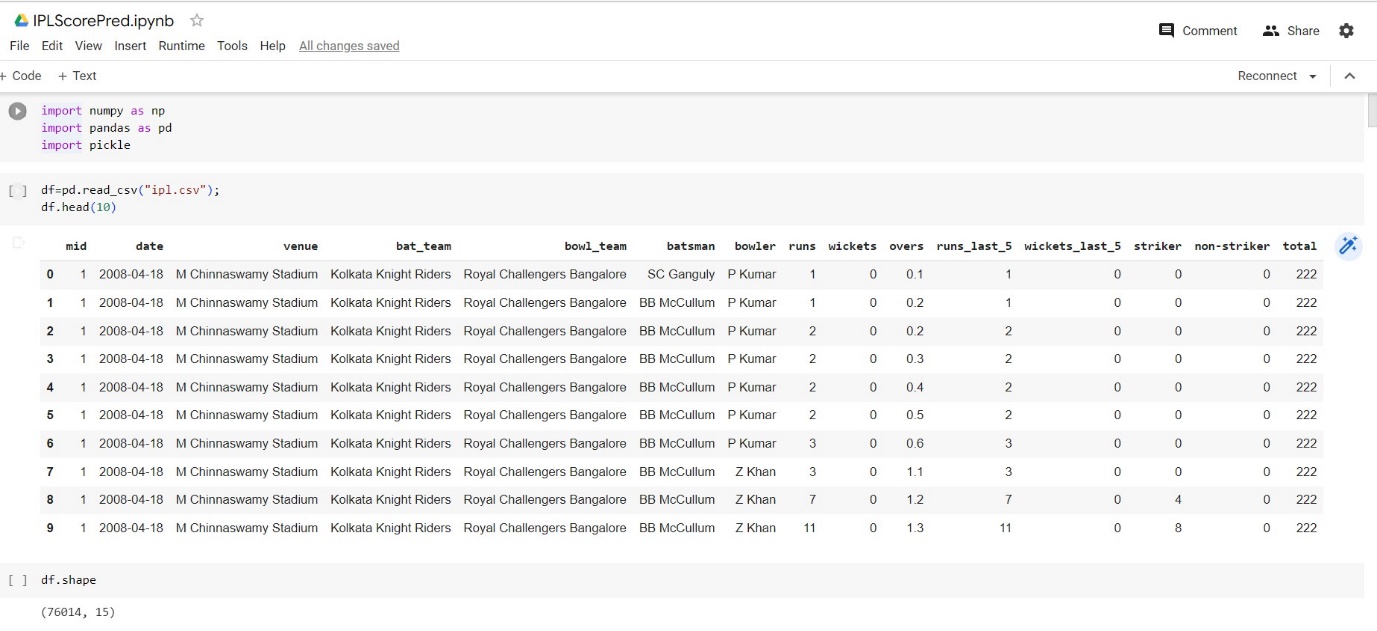
## 5.1 **Model Evaluation**

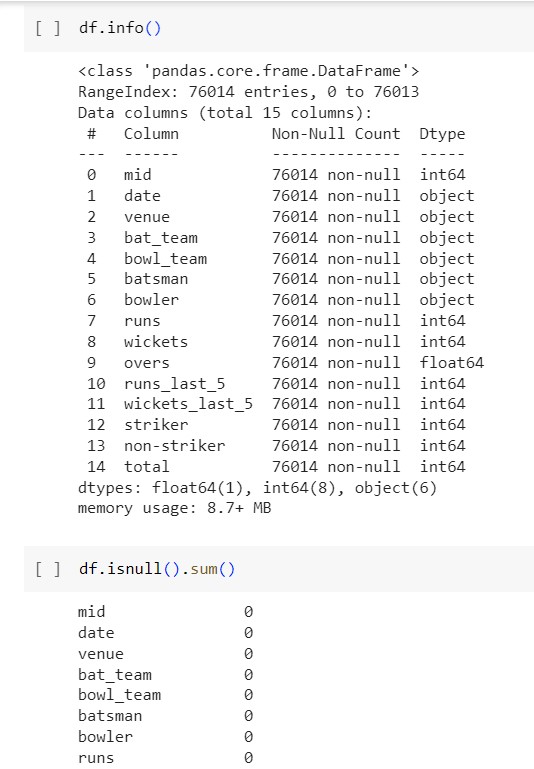
After training comes the evaluation part which tells how our model is performing on data other than the train set. If it performs well, it is good to be deployed in the wild, else you need to reiterate the entire process(or at least a few processes, boils down to how you apply logic). Let’s test our model too using the prediction method. We will be passing the X\_test to the same. The results are returned as predictions and hence it’s better to plot it out as graphs for better visualisation and also measure them using a few metrics like Mean Absolute Error, Mean Square Error and Root Mean Square Error. After evaluation, we save our model using the pickle library which creates a model .pkl file.

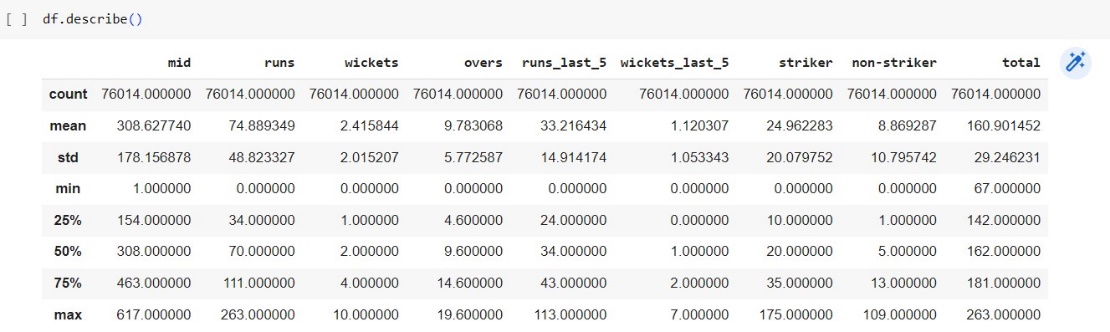
Step 6: GUI Development

For GUI, I have used Gradio which is used to demo a machine learning model with a friendly web interface. It is imported into our code with a simple pip install gradio and import gradio as gr. We can set the parameters that we want to evaluate the score. We are using batting team, bowling team, overs completed, runs scored, wickets taken, runs and wickets in previous five overs as the parameters to predict are score.

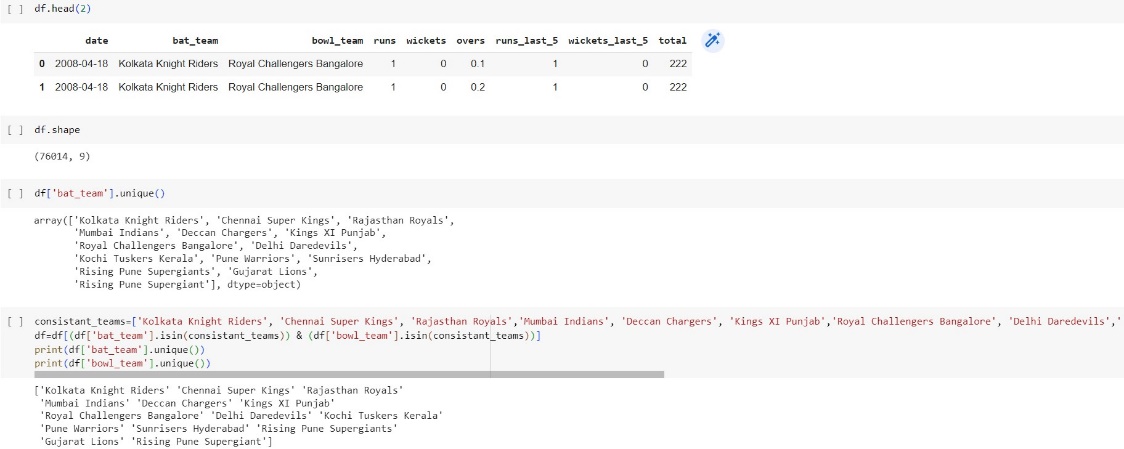
**PROJECT EXECUTION:**

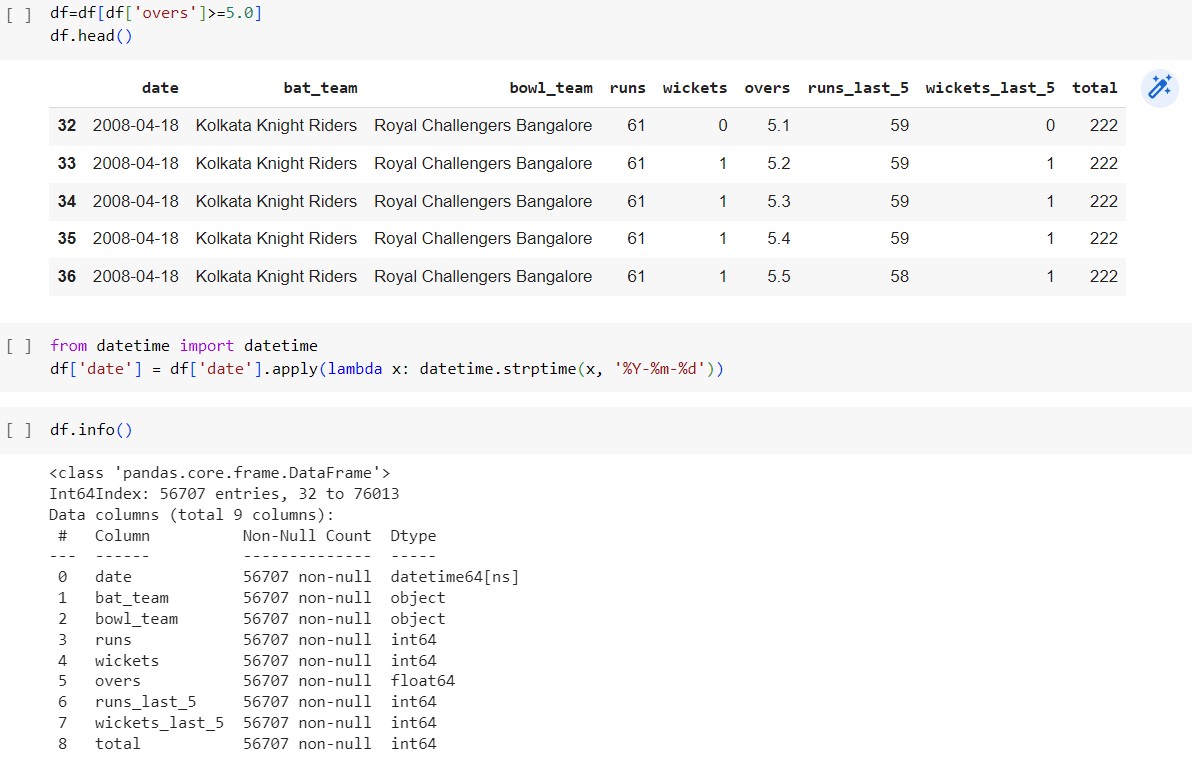


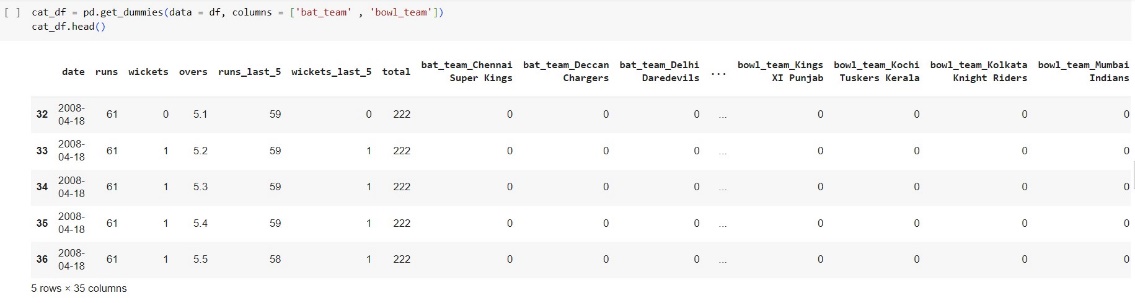


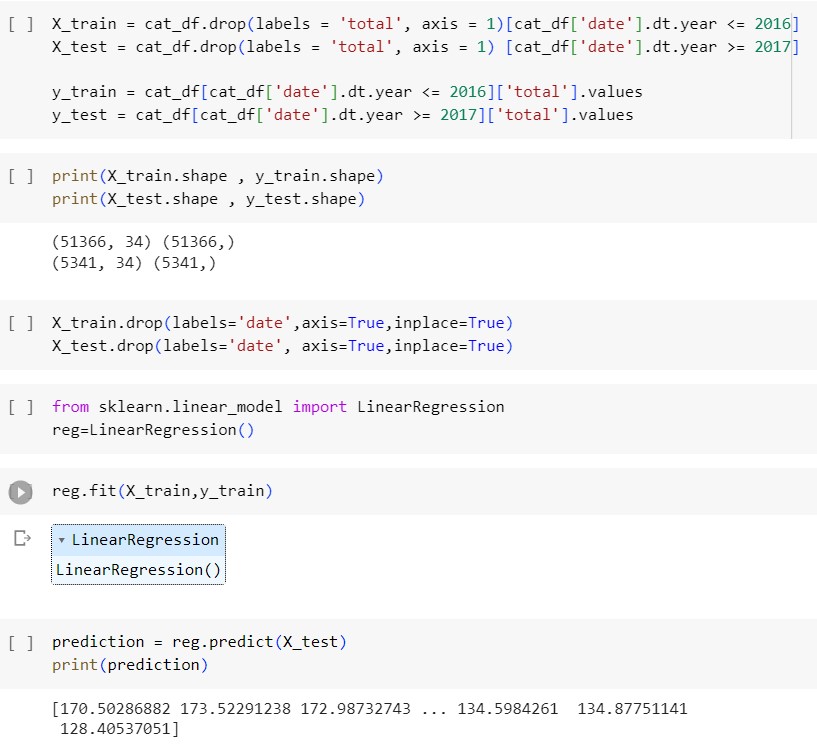


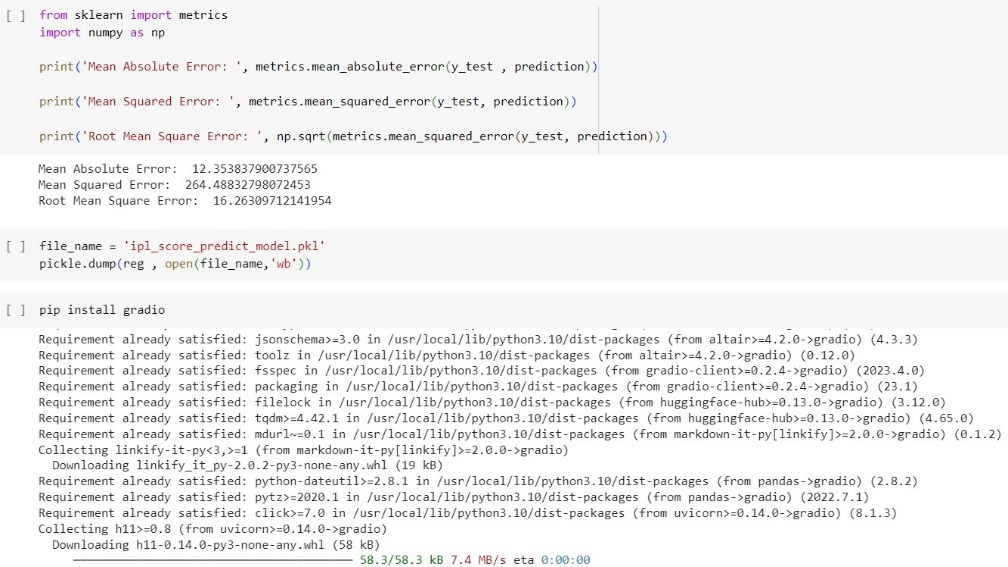


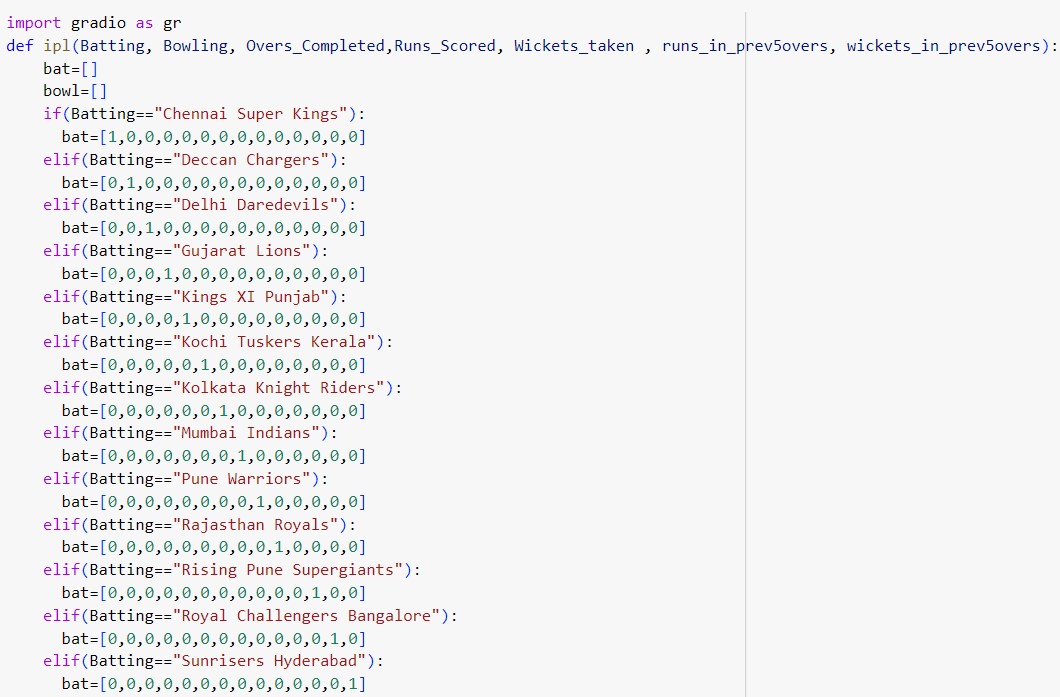






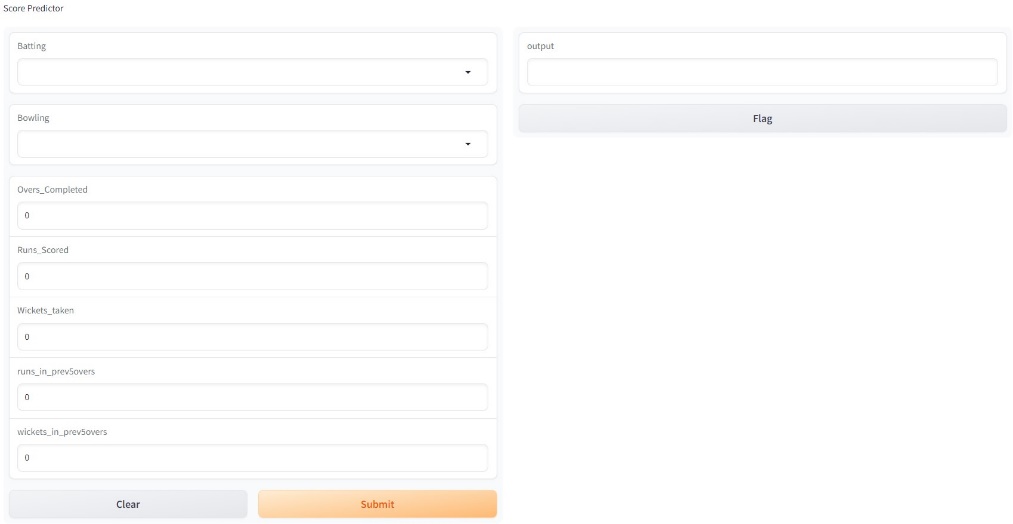




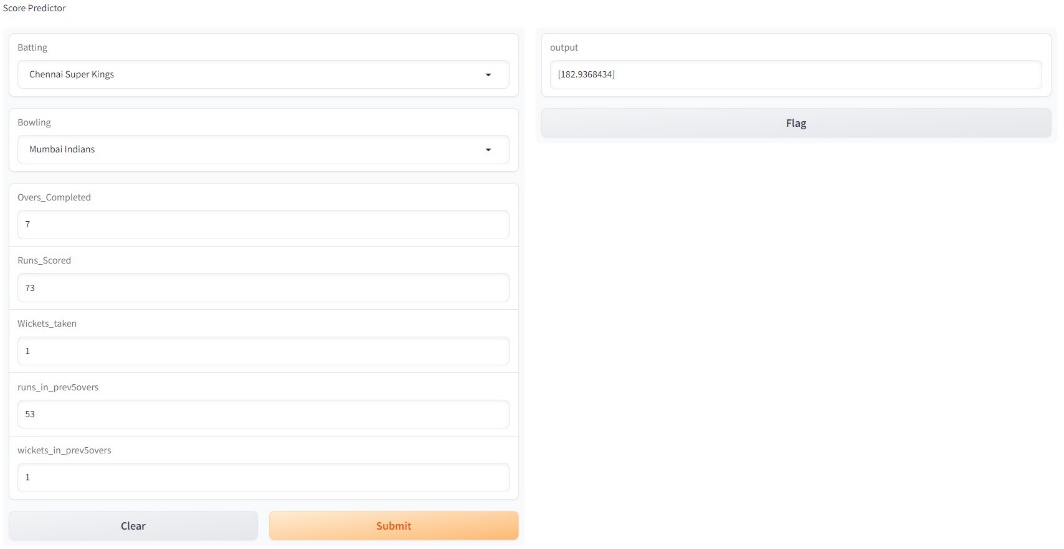


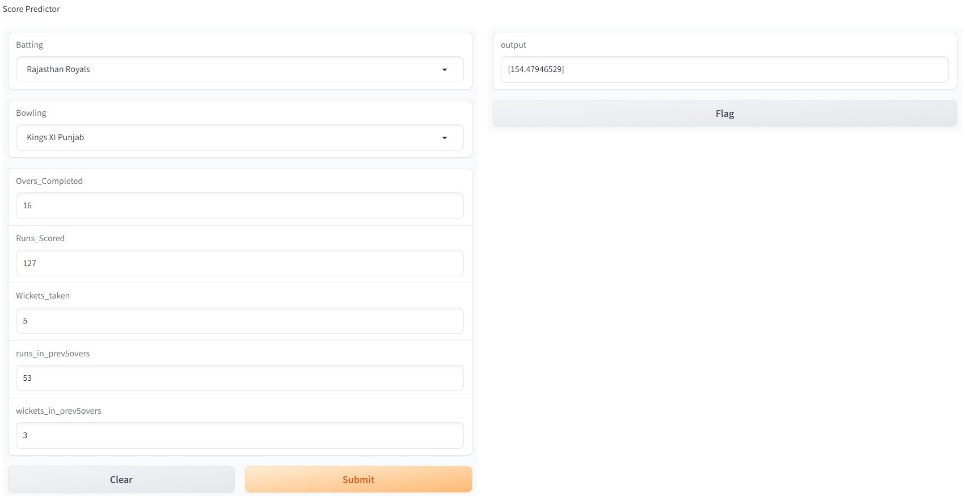
**OUTPUTS:**

**GUI:**



**TESTS:**





**RESULT:** Thus, an IPL Score Predictor was implemented using Linear Regression and executed using a GUI.