VIT
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Vellore-632014, Tamil Nadu, India

Artificial Intelligence

Course Code: CSE3013

(J-Component)

Score Prediction using AI

Submitted by

Kirit Kumar (19BCB0001)

Prakhar Soni (19BCB0006)

Submitted to

Dr. ANNAPURNA JONNALAGADDA

SCOPE

June 2021

1. Need for the review:

Sports is always a very attractive topic for people to discuss, and seeing whether past performance and matches between certain teams can be quantified and used to predict future matches is an interesting prospect.

2. Scope of the review:

The result of this project will be the prediction of scores of future matches, within reasonable error, as well as the collection of data we obtain from it about the sport. We also aim to provide visualizations for all the data to find unique and unthought of correlations and insights into the game of cricket.

3. Significance of the review:

This can also be extended to other sports and even business, provided the model of using past performance to predict future performance remains the same. (Gambling). We aim to use the data obtained from cricket matches to predict the scores of future matches, using Artificial Intelligence Models , depending on matches between different teams.

4. Introduction:

We will be using Machine Learning and Artificial Intelligence to train a model to recognize a pattern or trend in the scores of teams, for all their matches. We will be using T20, ODI and IPL data (three forms of Cricket matches) to train the model. Once the initial dataset is fed into the model, we will plot regression plots using this data, to try and see if there is a trend in the scores of matches, and whether it can be extended to predict future matches. In regards to this, we will check the accuracy of the prediction with an actual score to test the validity of our model.

5. Literature survey

a. Existing models:

As cricket is an extremely popular topic, it is expected that score predictors appeal to a lot of people. Falling in line with this there are various cricket score predictors available that perform similar functions to our project. They predict outcomes/scores using models based on various data points from their datasets.

b. Gaps identified/Issues in the existing models:

There are a few key differences between existing models available online and ours; we diversify our incoming data to different types of matches. For example, we use T20 data, ODI data and IPL data (three different match patterns) rather than just T20 data, furthermore, we use various models to fit each type of match in order to find which model performs best for which match type.

Some of the models used in our project are the Linear Regression model, the Random Forest Regression model, and the Decision Tree Regressor.

Strengths:

1. We use various Regression models to predict the score, for example Linear Regression, Random Forest Regression, and Decision Tree Regressor
2. We use match data from multiple types of matches to diversify the input data in order to achieve maximum accuracy in the prediction. For example, using T20 data, ODI data and IPL data (three different match patterns) rather than just T20 data.

Weakness:

1. Cannot take into account various external factors for example ball condition, pitch conditions, players mental and physical condition, and various environmental factors.
2. The complete dataset is unavailable as the data is not freely available, this does not allow us to keep our predictions completely up to date.

Opportunities:

1. Model allows us to predict scores, which can be used to further refine the team based on the past models; if a player performs better than another against a particular team, the player can be put in for that match.

Threats:

1. One of the threats posed to our project is not having enough data to operate on, All the data points are not freely available and only some are released. Furthermore, environmental factors can greatly change the outcome of a game.
2. Other projects can include better and more up to date data, as that data is not freely available.

6. Discussion on implementation aspects

a. Tools/packages/languages

Python

Pandas, numpy, seaborn, matplotlib, sklearn (models)

b. Implementation aspects in the literature

   a) We perform data visualization to understand the data better.
   b) Then, implementing seaborn to plot box plots for runs, wickets, total runs, etc.
   c) Scatter plots for wickets, overs using matplotlib.
   d) Heat maps for finding correlations between the columns.
   e) Scatter plots for various other data points.
   f) The textual data is then converted into numeric data so that those columns can be used for prediction.
   g) Using various models such as Linear Regression, Decision Tree Regressor, and Random Forest Regressor, the data is trained.
   h) For each model, we test the accuracy of the model using the R square value.
   i) We also test the accuracy of the model using a custom accuracy script that uses the test data.
   j) The predicted values based on actual training data and finding margin of error for each model are found.
   k) The total runs are predicted using random data as input to simulate new games for each model.

c. Data sets:

3 different datasets:

   a) ODI data
   b) IPL data
   c) T20 data

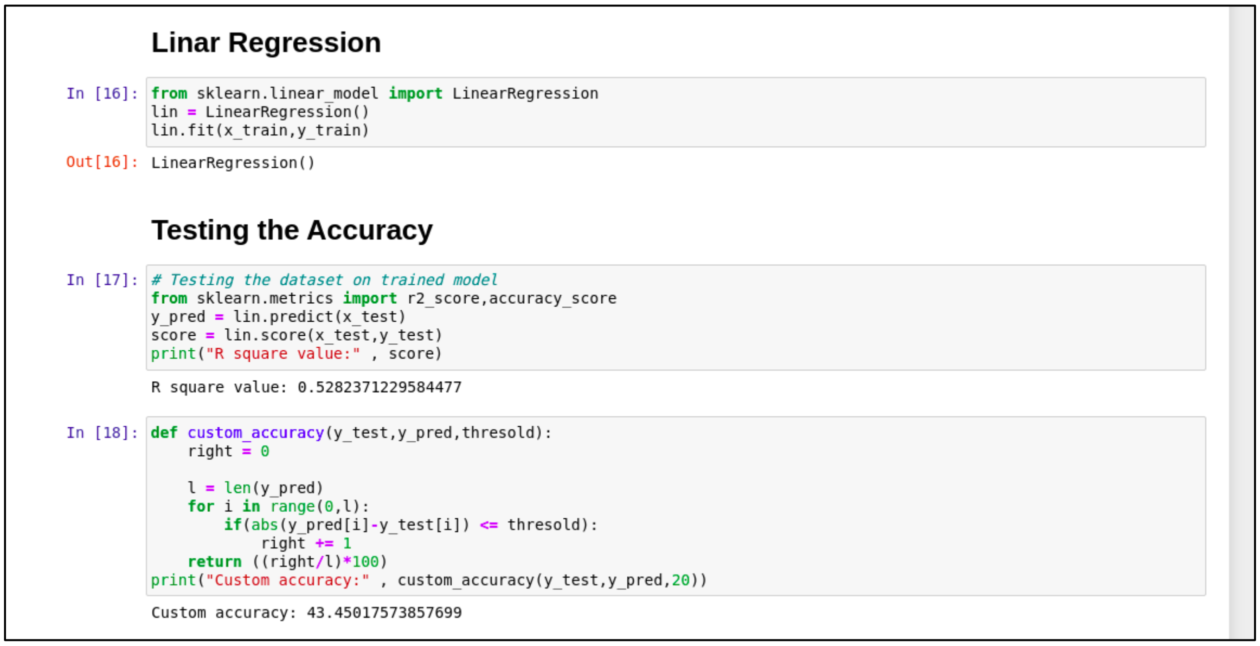
d. Metrics for evaluation

   a) Accuracy of prediction (How close is the predicted score to the actual score)
   b) Margin of error

e. Summary of your observations:

ODI Dataset:

1. Linear Regression



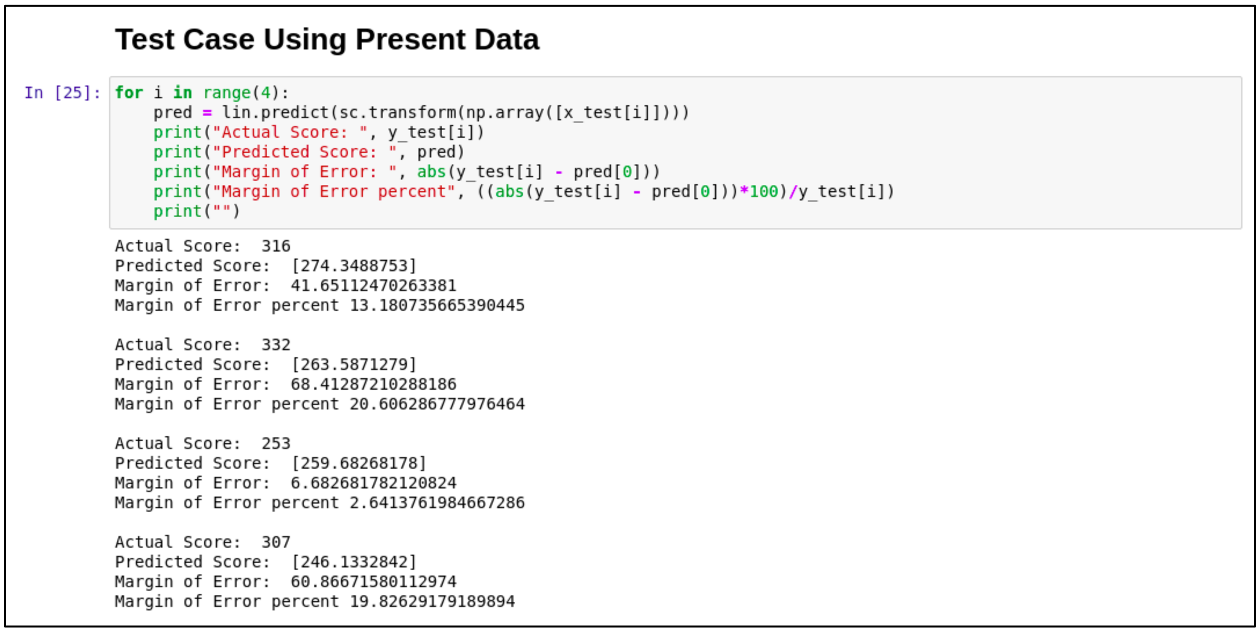
Linar Regression

In [16]:
```
from sklearn.linear_model import LinearRegression
lin = LinearRegression()
lin.fit(x_train,y_train)
```
Out[16]:
```
LinearRegression()
```

Testing the Accuracy

In [17]:
```
# Testing the dataset on trained model
from sklearn.metrics import r2_score,accuracy_score
y_pred = lin.predict(x_test)
score = lin.score(x_test,y_test)
print("R square value:" , score)
```
```
R square value: 0.5282371229584477
```

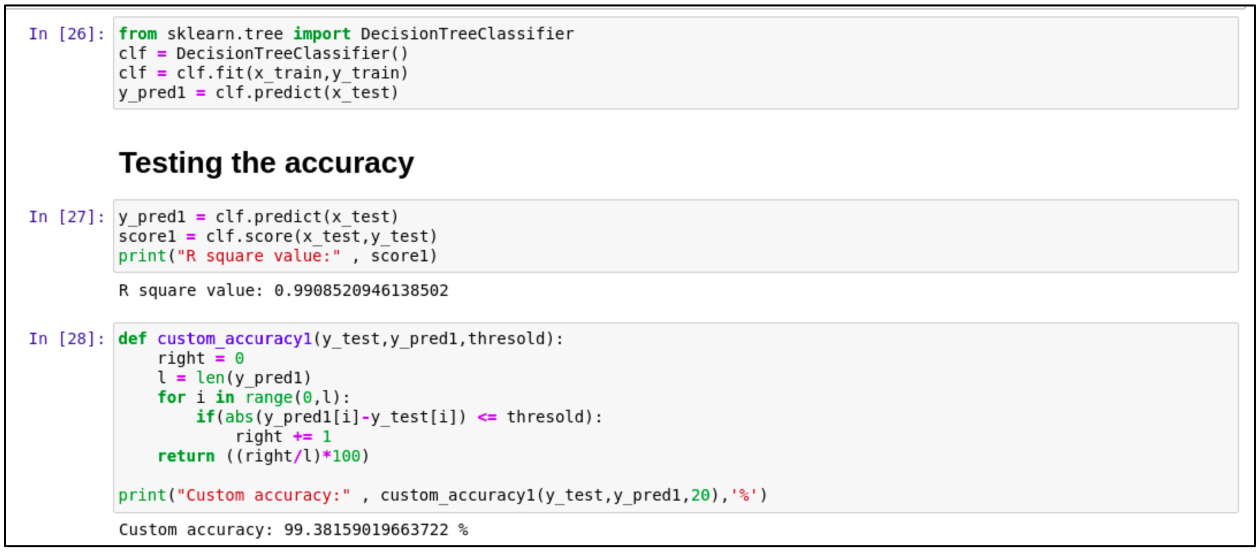
In [18]:
```
def custom_accuracy(y_test,y_pred,thresold):
    right = 0

    l = len(y_pred)
    for i in range(0,l):
        if(abs(y_pred[i]-y_test[i]) <= thresold):
            right += 1
    return ((right/l)*100)
print("Custom accuracy:" , custom_accuracy(y_test,y_pred,20))
```
```
Custom accuracy: 43.45017573857699
```



Test Case Using Present Data

In [25]:
```
for i in range(4):
    pred = lin.predict(sc.transform(np.array([x_test[i]])))
    print("Actual Score: ", y_test[i])
    print("Predicted Score: ", pred)
    print("Margin of Error: ", abs(y_test[i] - pred[0]))
    print("Margin of Error percent", ((abs(y_test[i] - pred[0]))*100)/y_test[i])
    print("")
```
```
Actual Score:  316
Predicted Score:  [274.3488753]
Margin of Error:  41.65112470263381
Margin of Error percent 13.180735665390445

Actual Score:  332
Predicted Score:  [263.5871279]
Margin of Error:  68.41287210288186
Margin of Error percent 20.606286777976464

Actual Score:  253
Predicted Score:  [259.68268178]
Margin of Error:  6.682681782120824
Margin of Error percent 2.6413761984667286

Actual Score:  307
Predicted Score:  [246.1332842]
Margin of Error:  60.86671580112974
Margin of Error percent 19.82629179189894
```

2. Decision Tree Regressor

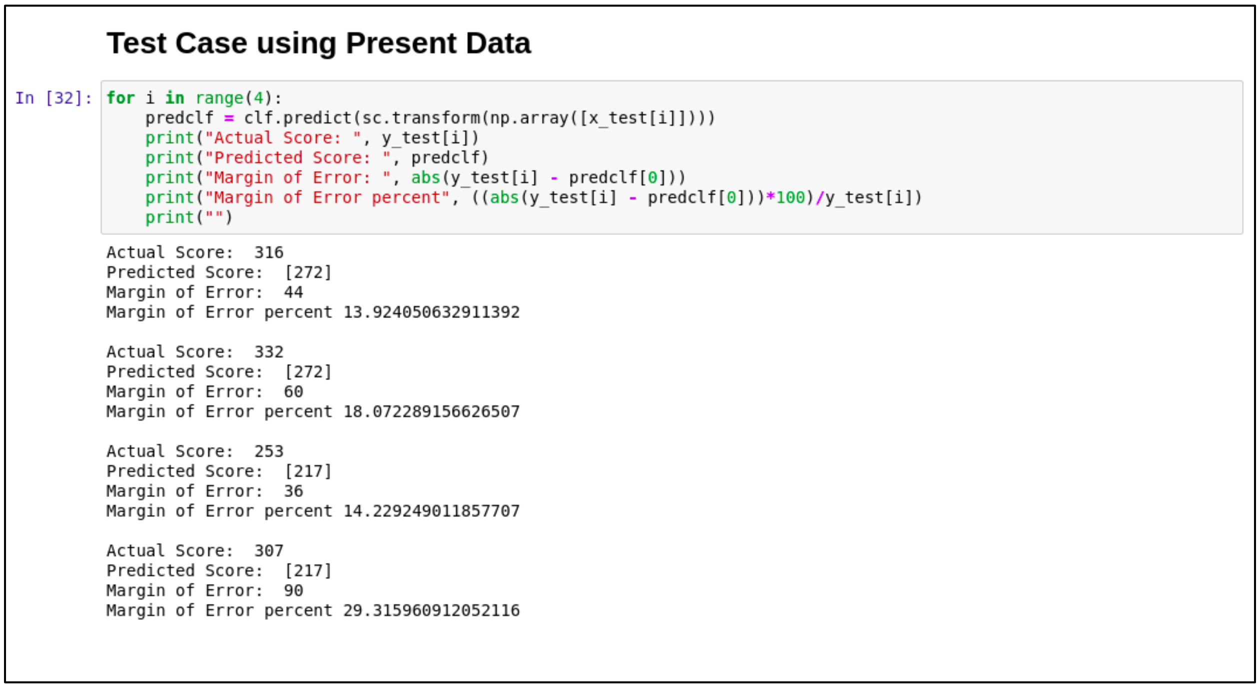


```
In [26]: from sklearn.tree import DecisionTreeClassifier
         clf = DecisionTreeClassifier()
         clf = clf.fit(x_train,y_train)
         y_pred1 = clf.predict(x_test)
```

Testing the accuracy

```
In [27]: y_pred1 = clf.predict(x_test)
         score1 = clf.score(x_test,y_test)
         print("R square value:" , score1)

         R square value: 0.9908520946138502
```

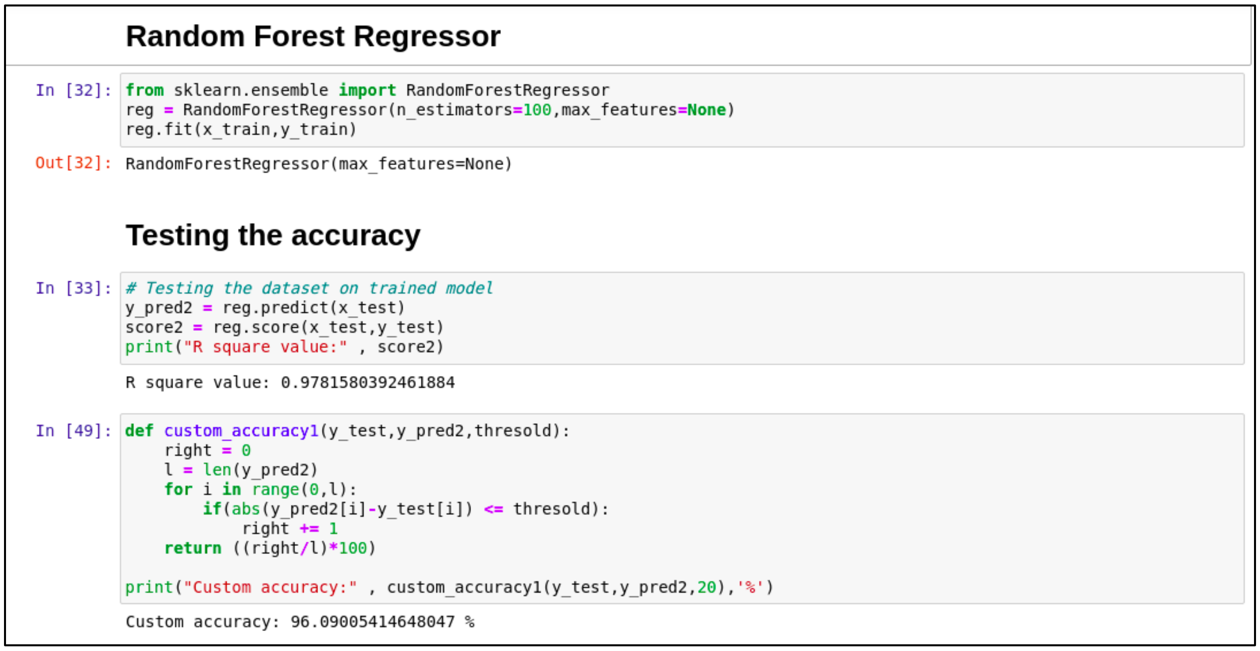
```
In [28]: def custom_accuracy1(y_test,y_pred1,thresold):
             right = 0
             l = len(y_pred1)
             for i in range(0,l):
                 if(abs(y_pred1[i]-y_test[i]) <= thresold):
                     right += 1
             return ((right/l)*100)

         print("Custom accuracy:" , custom_accuracy1(y_test,y_pred1,20),'%')

         Custom accuracy: 99.38159019663722 %
```



Test Case using Present Data

```
In [32]: for i in range(4):
             predclf = clf.predict(sc.transform(np.array([x_test[i]])))
             print("Actual Score: ", y_test[i])
             print("Predicted Score: ", predclf)
             print("Margin of Error: ", abs(y_test[i] - predclf[0]))
             print("Margin of Error percent", ((abs(y_test[i] - predclf[0]))*100)/y_test[i])
             print("")

         Actual Score:  316
         Predicted Score:  [272]
         Margin of Error:  44
         Margin of Error percent 13.924050632911392

         Actual Score:  332
         Predicted Score:  [272]
         Margin of Error:  60
         Margin of Error percent 18.072289156626507

         Actual Score:  253
         Predicted Score:  [217]
         Margin of Error:  36
         Margin of Error percent 14.229249011857707

         Actual Score:  307
         Predicted Score:  [217]
         Margin of Error:  90
         Margin of Error percent 29.315960912052116
```

3. Random Forest Regressor



Random Forest Regressor

In [32]:
```python
from sklearn.ensemble import RandomForestRegressor
reg = RandomForestRegressor(n_estimators=100,max_features=None)
reg.fit(x_train,y_train)
```

Out[32]:
```
RandomForestRegressor(max_features=None)
```

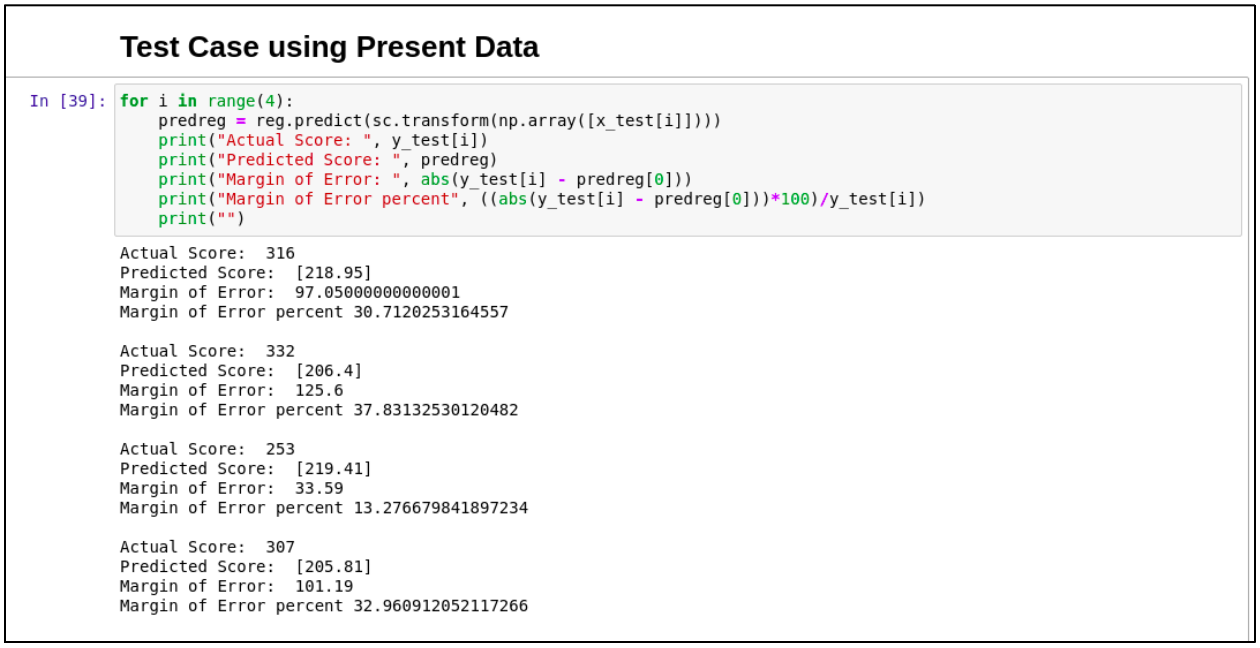
Testing the accuracy

In [33]:
```python
# Testing the dataset on trained model
y_pred2 = reg.predict(x_test)
score2 = reg.score(x_test,y_test)
print("R square value:" , score2)
```

```
R square value: 0.9781580392461884
```

In [49]:
```python
def custom_accuracy1(y_test,y_pred2,thresold):
    right = 0
    l = len(y_pred2)
    for i in range(0,l):
        if(abs(y_pred2[i]-y_test[i]) <= thresold):
            right += 1
    return ((right/l)*100)

print("Custom accuracy:" , custom_accuracy1(y_test,y_pred2,20),'%')
```

```
Custom accuracy: 96.09005414648047 %
```



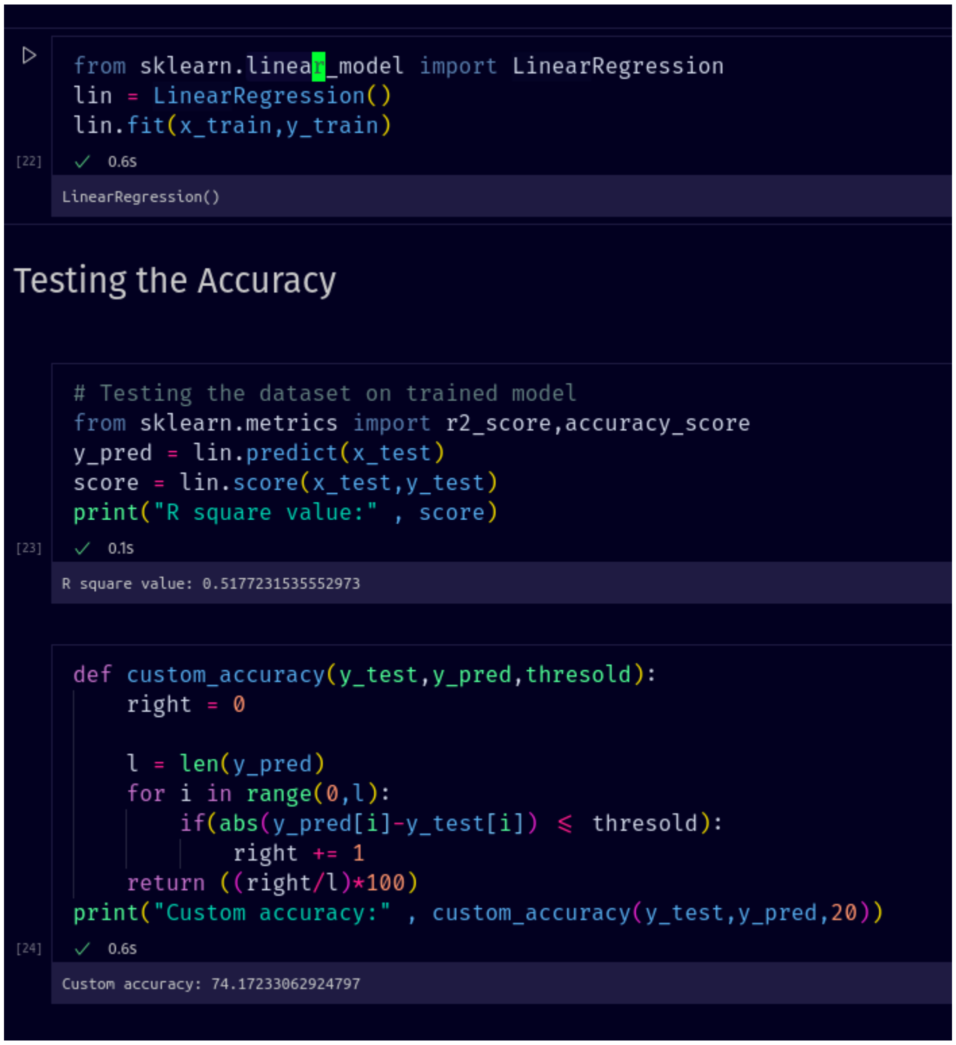
Test Case using Present Data

In [39]:
```python
for i in range(4):
    predreg = reg.predict(sc.transform(np.array([x_test[i]])))
    print("Actual Score: ", y_test[i])
    print("Predicted Score: ", predreg)
    print("Margin of Error: ", abs(y_test[i] - predreg[0]))
    print("Margin of Error percent", ((abs(y_test[i] - predreg[0]))*100)/y_test[i])
    print("")
```

```
Actual Score:  316
Predicted Score:  [218.95]
Margin of Error:  97.05000000000001
Margin of Error percent 30.7120253164557

Actual Score:  332
Predicted Score:  [206.4]
Margin of Error:  125.6
Margin of Error percent 37.83132530120482

Actual Score:  253
Predicted Score:  [219.41]
Margin of Error:  33.59
Margin of Error percent 13.276679841897234

Actual Score:  307
Predicted Score:  [205.81]
Margin of Error:  101.19
Margin of Error percent 32.960912052117266
```

IPL Dataset:

1. Linear Regression:



```
from sklearn.linear_model import LinearRegression
lin = LinearRegression()
lin.fit(x_train,y_train)
```
[22] ✓ 0.6s

```
LinearRegression()
```

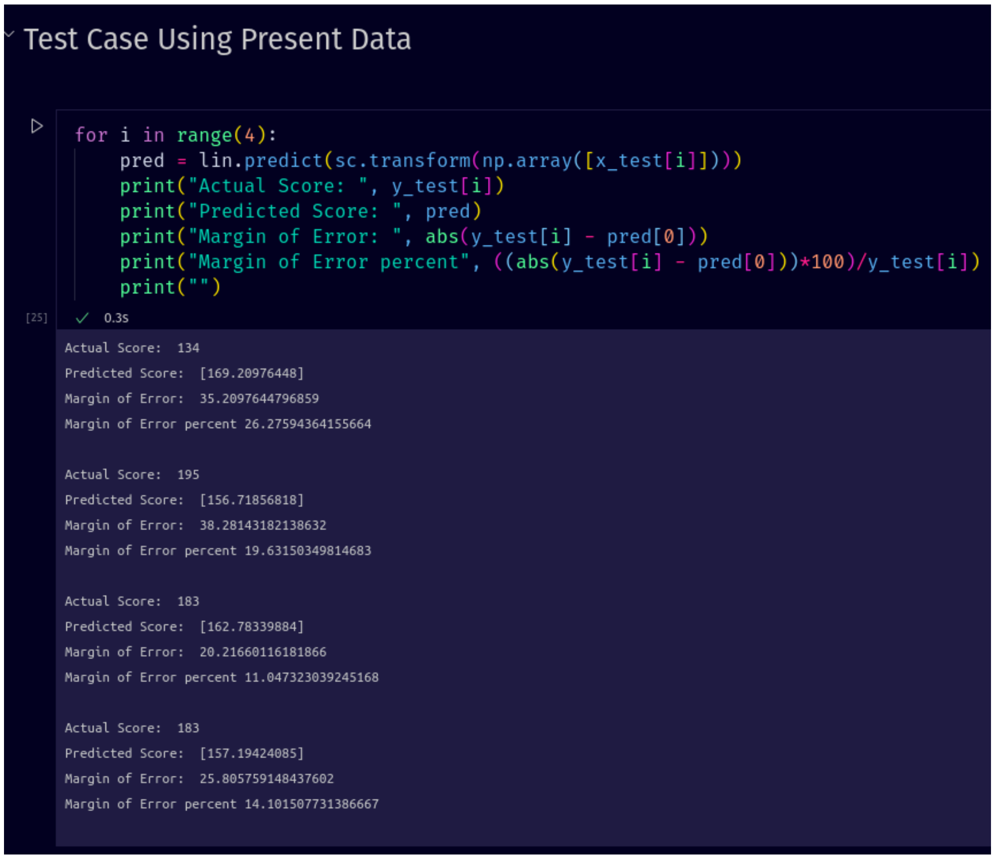
Testing the Accuracy

```
# Testing the dataset on trained model
from sklearn.metrics import r2_score,accuracy_score
y_pred = lin.predict(x_test)
score = lin.score(x_test,y_test)
print("R square value:" , score)
```
[23] ✓ 0.1s

```
R square value: 0.5177231535552973
```

```
def custom_accuracy(y_test,y_pred,thresold):
    right = 0

    l = len(y_pred)
    for i in range(0,l):
        if(abs(y_pred[i]-y_test[i]) <= thresold):
            right += 1
    return ((right/l)*100)
print("Custom accuracy:" , custom_accuracy(y_test,y_pred,20))
```
[24] ✓ 0.6s

```
Custom accuracy: 74.17233062924797
```

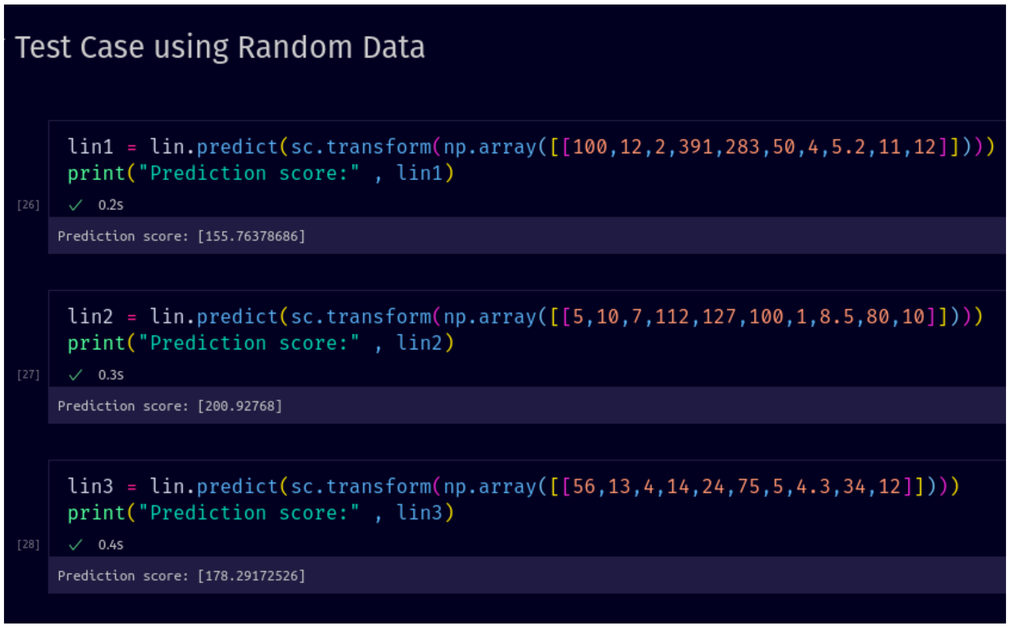


Test Case Using Present Data

```
for i in range(4):
    pred = lin.predict(sc.transform(np.array([x_test[i]])))
    print("Actual Score: ", y_test[i])
    print("Predicted Score: ", pred)
    print("Margin of Error: ", abs(y_test[i] - pred[0]))
    print("Margin of Error percent", ((abs(y_test[i] - pred[0]))*100)/y_test[i])
    print("")
```

[25] ✓ 0.3s

```
Actual Score:  134
Predicted Score:  [169.20976448]
Margin of Error:  35.2097644796859
Margin of Error percent 26.27594364155664

Actual Score:  195
Predicted Score:  [156.71856818]
Margin of Error:  38.28143182138632
Margin of Error percent 19.63150349814683

Actual Score:  183
Predicted Score:  [162.78339884]
Margin of Error:  20.21660116181866
Margin of Error percent 11.047323039245168

Actual Score:  183
Predicted Score:  [157.19424085]
Margin of Error:  25.805759148437602
Margin of Error percent 14.101507731386667
```



Test Case using Random Data

```
lin1 = lin.predict(sc.transform(np.array([[100,12,2,391,283,50,4,5.2,11,12]])))
print("Prediction score:" , lin1)
```

[26] ✓ 0.2s

```
Prediction score: [155.76378686]
```

```
lin2 = lin.predict(sc.transform(np.array([[5,10,7,112,127,100,1,8.5,80,10]])))
print("Prediction score:" , lin2)
```

[27] ✓ 0.3s

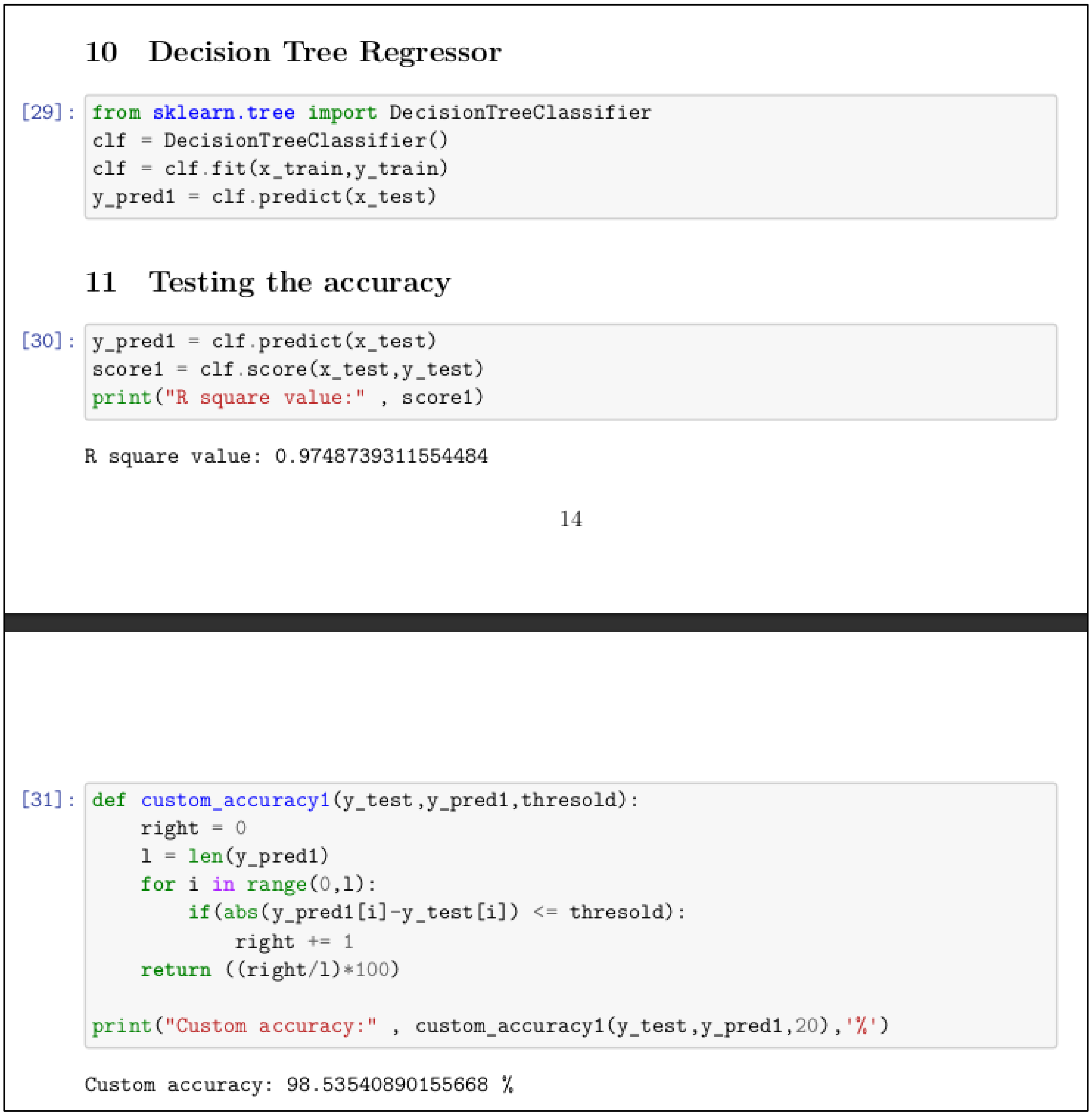
```
Prediction score: [200.92768]
```

```
lin3 = lin.predict(sc.transform(np.array([[56,13,4,14,24,75,5,4.3,34,12]])))
print("Prediction score:" , lin3)
```

[28] ✓ 0.4s

```
Prediction score: [178.29172526]
```

2. Decision Tree Regressor



10 Decision Tree Regressor

[29]:
```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
clf = clf.fit(x_train,y_train)
y_pred1 = clf.predict(x_test)
```

11 Testing the accuracy

[30]:
```
y_pred1 = clf.predict(x_test)
score1 = clf.score(x_test,y_test)
print("R square value:" , score1)
```

```
R square value: 0.9748739311554484
```

14

[31]:
```
def custom_accuracy1(y_test,y_pred1,thresold):
    right = 0
    l = len(y_pred1)
    for i in range(0,l):
        if(abs(y_pred1[i]-y_test[i]) <= thresold):
            right += 1
    return ((right/l)*100)

print("Custom accuracy:" , custom_accuracy1(y_test,y_pred1,20),'%')
```

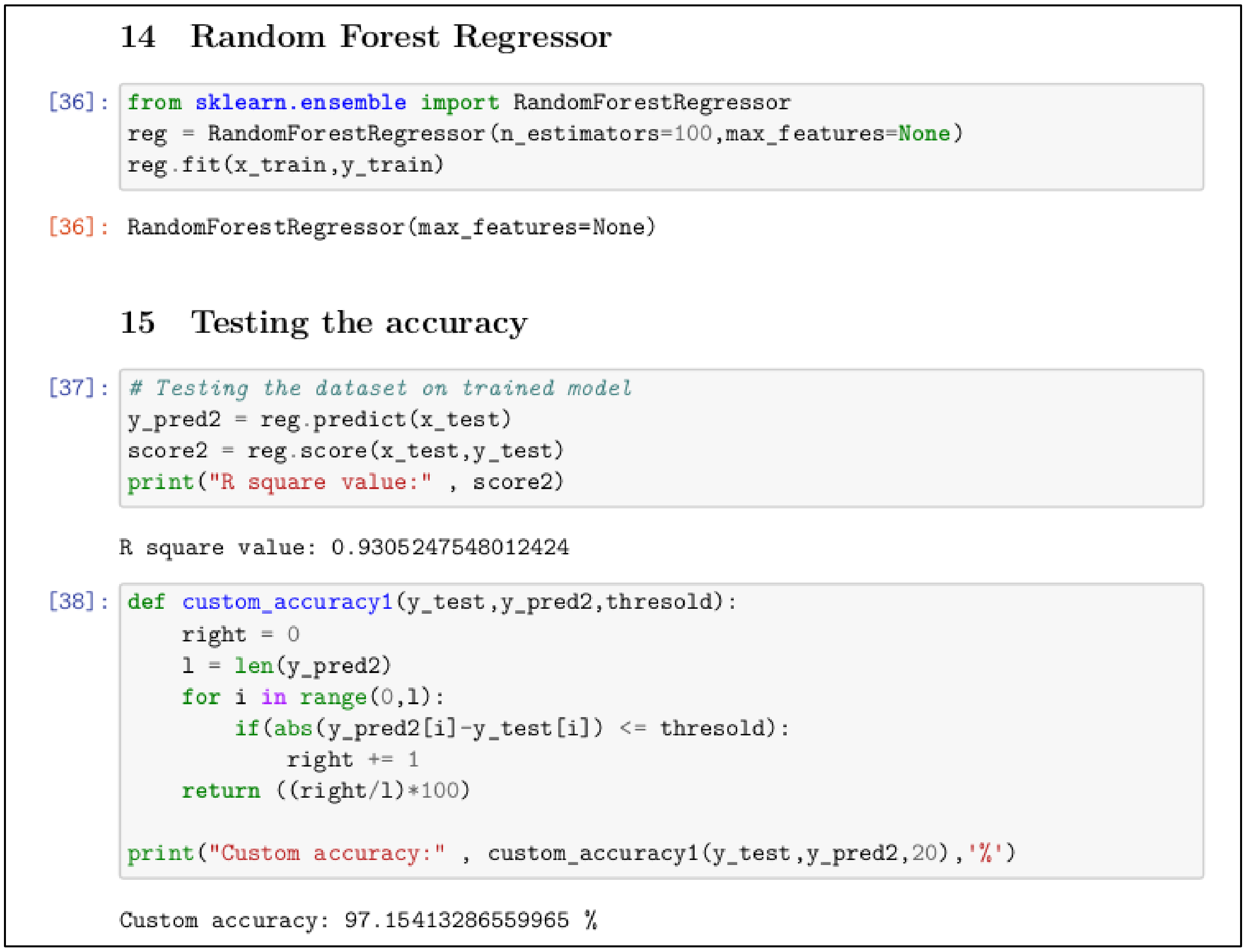
```
Custom accuracy: 98.53540890155668 %
```

12 Test Case using Present Data

```
[32]: for i in range(4):
          predclf = clf.predict(sc.transform(np.array([x_test[i]])))
          print("Actual Score: ", y_test[i])
          print("Predicted Score: ", predclf)
          print("Margin of Error: ", abs(y_test[i] - predclf[0]))
          print("Margin of Error percent", ((abs(y_test[i] - predclf[0]))*100)/
       →y_test[i])
          print("")
```

```
Actual Score:  134
Predicted Score:  [178]
Margin of Error:  44
Margin of Error percent 32.83582089552239

Actual Score:  195
Predicted Score:  [185]
Margin of Error:  10
Margin of Error percent 5.128205128205129

Actual Score:  183
Predicted Score:  [185]
Margin of Error:  2
Margin of Error percent 1.092896174863388

Actual Score:  183
Predicted Score:  [185]
Margin of Error:  2
Margin of Error percent 1.092896174863388
```

3. Random Forest Regressor

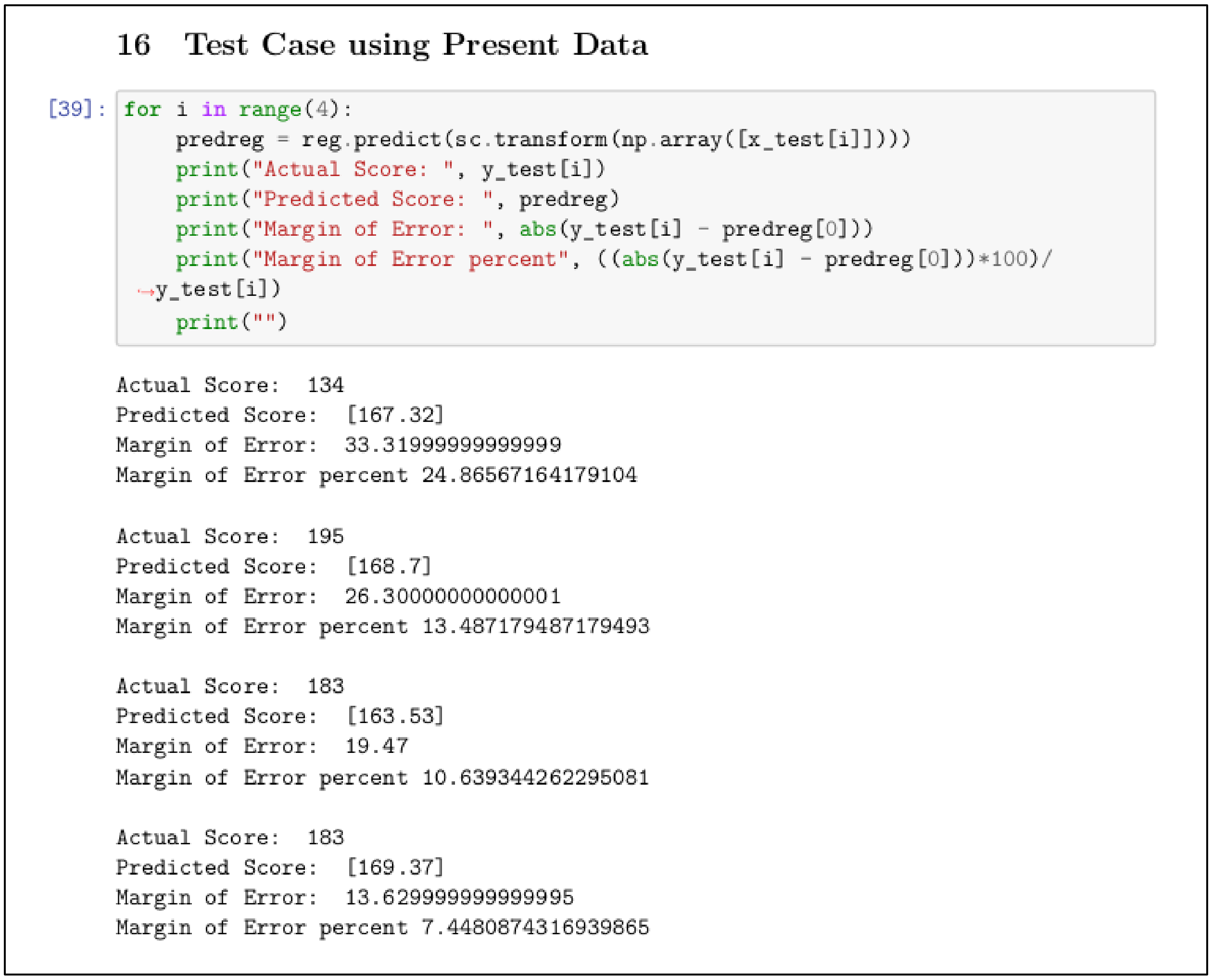


14 Random Forest Regressor

```
[36]: from sklearn.ensemble import RandomForestRegressor
      reg = RandomForestRegressor(n_estimators=100,max_features=None)
      reg.fit(x_train,y_train)

[36]: RandomForestRegressor(max_features=None)
```

15 Testing the accuracy

```
[37]: # Testing the dataset on trained model
      y_pred2 = reg.predict(x_test)
      score2 = reg.score(x_test,y_test)
      print("R square value:" , score2)

      R square value: 0.9305247548012424

[38]: def custom_accuracy1(y_test,y_pred2,thresold):
          right = 0
          l = len(y_pred2)
          for i in range(0,l):
              if(abs(y_pred2[i]-y_test[i]) <= thresold):
                  right += 1
          return ((right/l)*100)

      print("Custom accuracy:" , custom_accuracy1(y_test,y_pred2,20),'%')

      Custom accuracy: 97.15413286559965 %
```



16 Test Case using Present Data

```
[39]: for i in range(4):
          predreg = reg.predict(sc.transform(np.array([x_test[i]])))
          print("Actual Score: ", y_test[i])
          print("Predicted Score: ", predreg)
          print("Margin of Error: ", abs(y_test[i] - predreg[0]))
          print("Margin of Error percent", ((abs(y_test[i] - predreg[0]))*100)/
       ↪y_test[i])
          print("")
```

```
Actual Score:  134
Predicted Score:  [167.32]
Margin of Error:  33.31999999999999
Margin of Error percent 24.86567164179104

Actual Score:  195
Predicted Score:  [168.7]
Margin of Error:  26.30000000000001
Margin of Error percent 13.487179487179493

Actual Score:  183
Predicted Score:  [163.53]
Margin of Error:  19.47
Margin of Error percent 10.639344262295081

Actual Score:  183
Predicted Score:  [169.37]
Margin of Error:  13.629999999999995
Margin of Error percent 7.4480874316939865
```

3. T20 Results link provided in conclusion

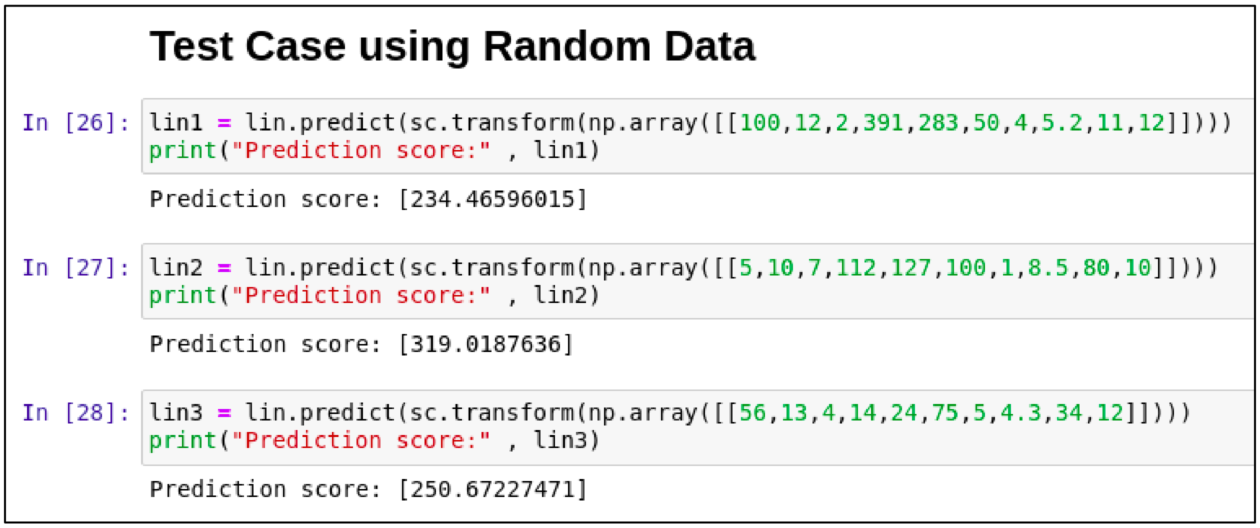
7. Conclusions and future aspects:
This project can be used as a template for a general, high accuracy score predictor that is not limited to only cricket, it can accommodate any sport provided the data is appropriate and the code is adjusted to match the dataset for the respective game.

In further projects, we would like to port this project onto some of our other favorite sports such as football, basketball and various e-sports (games which are played online).

We are able to use these models to generate predictions based on various inputted data points. One can use data from a future match to predict its outcome as shown below:

1. Using Linear Regression Model

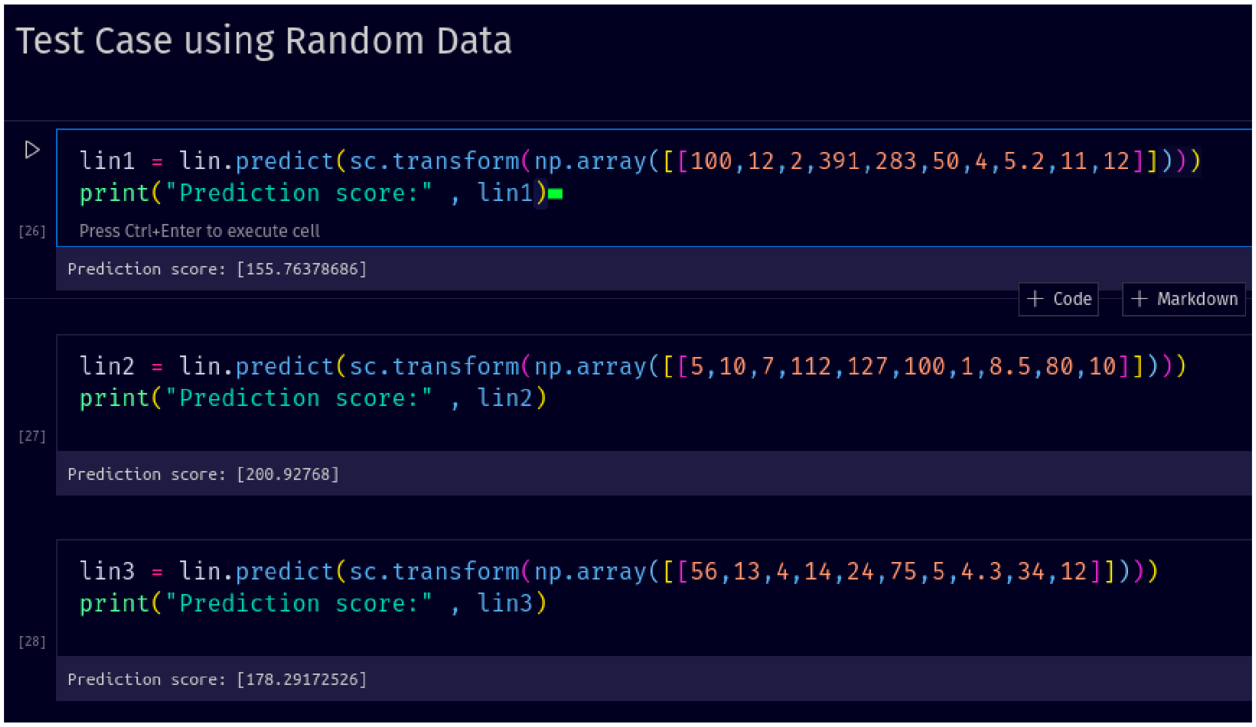
a. ODI:



Test Case using Random Data

```
In [26]: lin1 = lin.predict(sc.transform(np.array([[100,12,2,391,283,50,4,5.2,11,12]])))
         print("Prediction score:" , lin1)

         Prediction score: [234.46596015]

In [27]: lin2 = lin.predict(sc.transform(np.array([[5,10,7,112,127,100,1,8.5,80,10]])))
         print("Prediction score:" , lin2)

         Prediction score: [319.0187636]

In [28]: lin3 = lin.predict(sc.transform(np.array([[56,13,4,14,24,75,5,4.3,34,12]])))
         print("Prediction score:" , lin3)

         Prediction score: [250.67227471]
```

b. IPL:

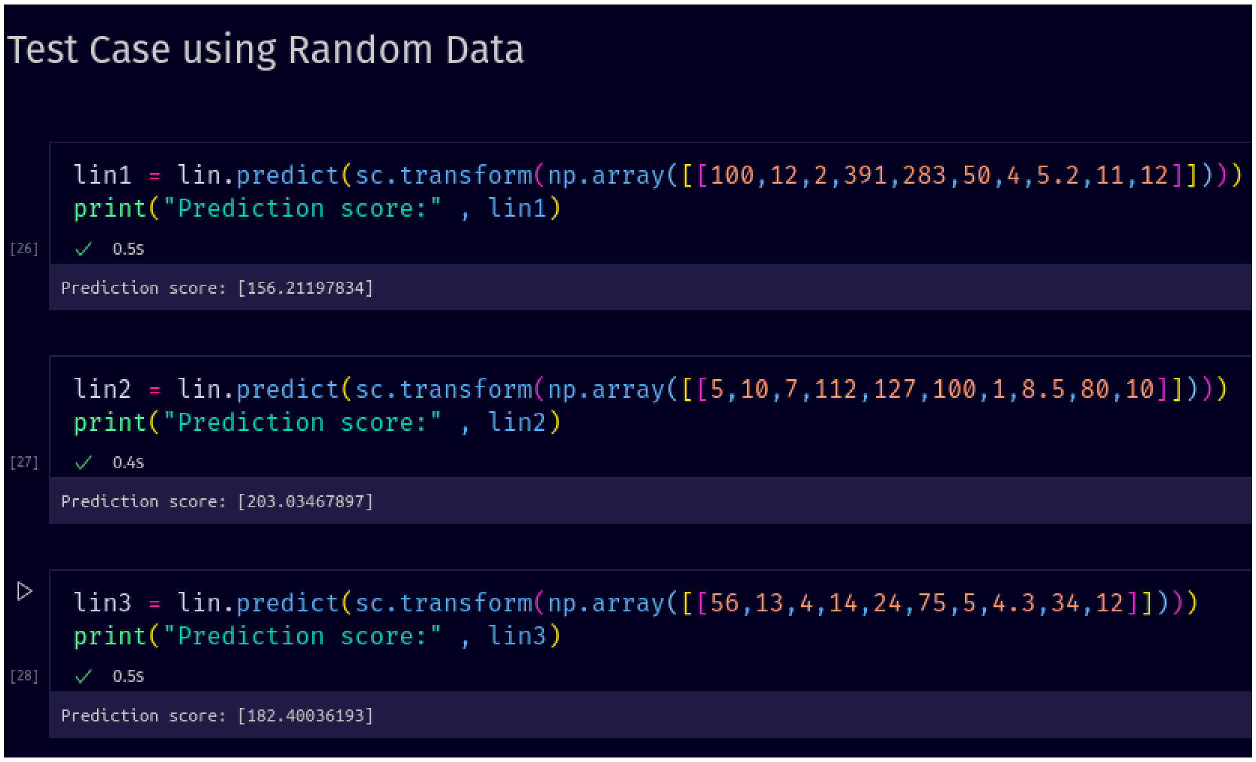


Test Case using Random Data

```
lin1 = lin.predict(sc.transform(np.array([[100,12,2,391,283,50,4,5.2,11,12]])))
print("Prediction score:" , lin1)
[26]
Press Ctrl+Enter to execute cell
Prediction score: [155.76378686]
```

```
lin2 = lin.predict(sc.transform(np.array([[5,10,7,112,127,100,1,8.5,80,10]])))
print("Prediction score:" , lin2)
[27]
Prediction score: [200.92768]

lin3 = lin.predict(sc.transform(np.array([[56,13,4,14,24,75,5,4.3,34,12]])))
print("Prediction score:" , lin3)
[28]
Prediction score: [178.29172526]
```

c. **T20:**



Test Case using Random Data

```
lin1 = lin.predict(sc.transform(np.array([[100,12,2,391,283,50,4,5.2,11,12]])))
print("Prediction score:" , lin1)
```

[26] ✓ 0.5s

```
Prediction score: [156.21197834]
```

```
lin2 = lin.predict(sc.transform(np.array([[5,10,7,112,127,100,1,8.5,80,10]])))
print("Prediction score:" , lin2)
```

[27] ✓ 0.4s

```
Prediction score: [203.03467897]
```

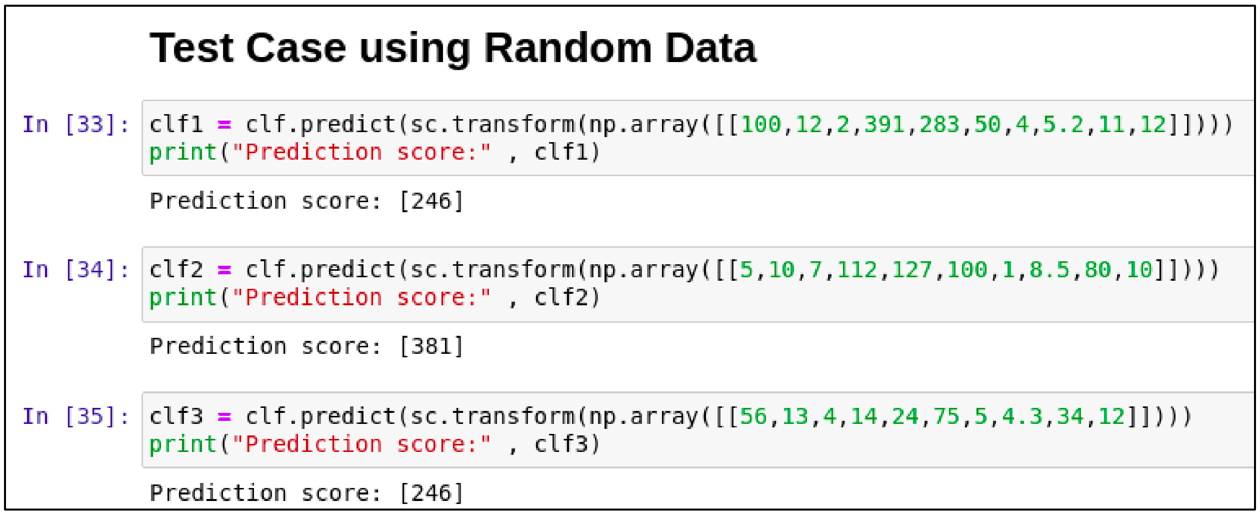
```
lin3 = lin.predict(sc.transform(np.array([[56,13,4,14,24,75,5,4.3,34,12]])))
print("Prediction score:" , lin3)
```

[28] ✓ 0.5s

```
Prediction score: [182.40036193]
```

2. Using the Decision Tree Regressor Model

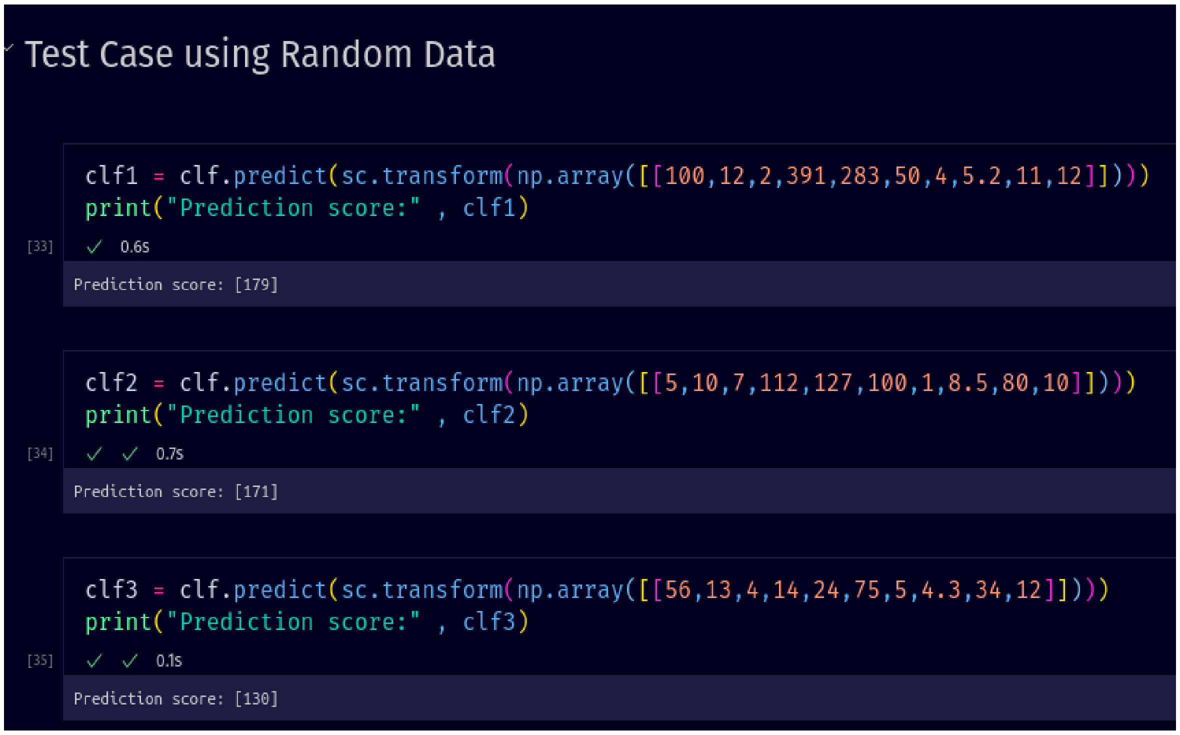
a. ODI:



Test Case using Random Data

```
In [33]: clf1 = clf.predict(sc.transform(np.array([[100,12,2,391,283,50,4,5.2,11,12]])))
         print("Prediction score:" , clf1)

         Prediction score: [246]

In [34]: clf2 = clf.predict(sc.transform(np.array([[5,10,7,112,127,100,1,8.5,80,10]])))
         print("Prediction score:" , clf2)

         Prediction score: [381]

In [35]: clf3 = clf.predict(sc.transform(np.array([[56,13,4,14,24,75,5,4.3,34,12]])))
         print("Prediction score:" , clf3)

         Prediction score: [246]
```

b. T20:



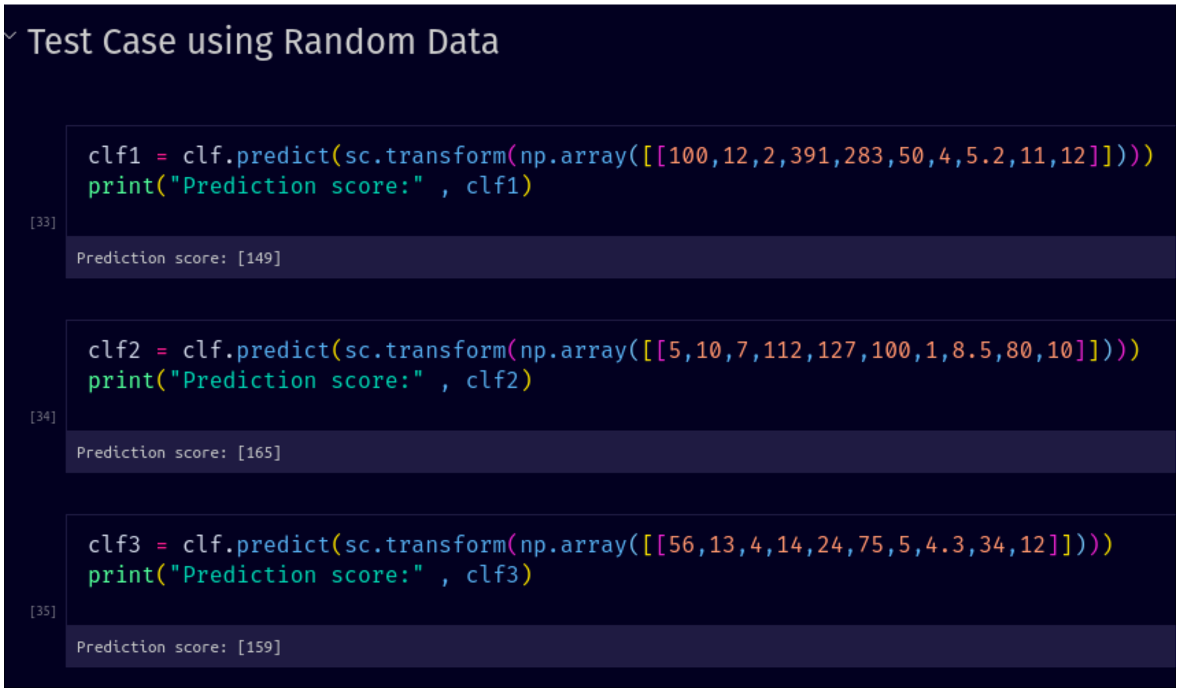
Test Case using Random Data

```
[33]
clf1 = clf.predict(sc.transform(np.array([[100,12,2,391,283,50,4,5.2,11,12]])))
print("Prediction score:" , clf1)
✓ 0.6s
Prediction score: [179]
```

```
[34]
clf2 = clf.predict(sc.transform(np.array([[5,10,7,112,127,100,1,8.5,80,10]])))
print("Prediction score:" , clf2)
✓ ✓ 0.7s
Prediction score: [171]
```

```
[35]
clf3 = clf.predict(sc.transform(np.array([[56,13,4,14,24,75,5,4.3,34,12]])))
print("Prediction score:" , clf3)
✓ ✓ 0.1s
Prediction score: [130]
```

c. IPL:



Test Case using Random Data

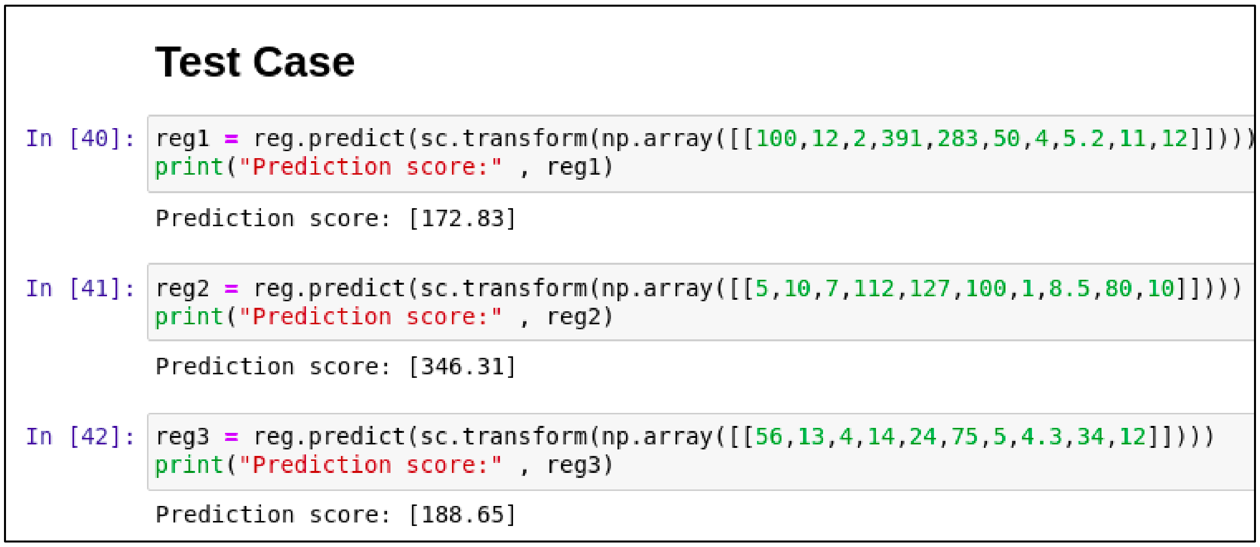
```
[33]
clf1 = clf.predict(sc.transform(np.array([[100,12,2,391,283,50,4,5.2,11,12]])))
print("Prediction score:" , clf1)
Prediction score: [149]
```

```
[34]
clf2 = clf.predict(sc.transform(np.array([[5,10,7,112,127,100,1,8.5,80,10]])))
print("Prediction score:" , clf2)
Prediction score: [165]
```

```
[35]
clf3 = clf.predict(sc.transform(np.array([[56,13,4,14,24,75,5,4.3,34,12]])))
print("Prediction score:" , clf3)
Prediction score: [159]
```

3. Using Random Forest Regressor Model

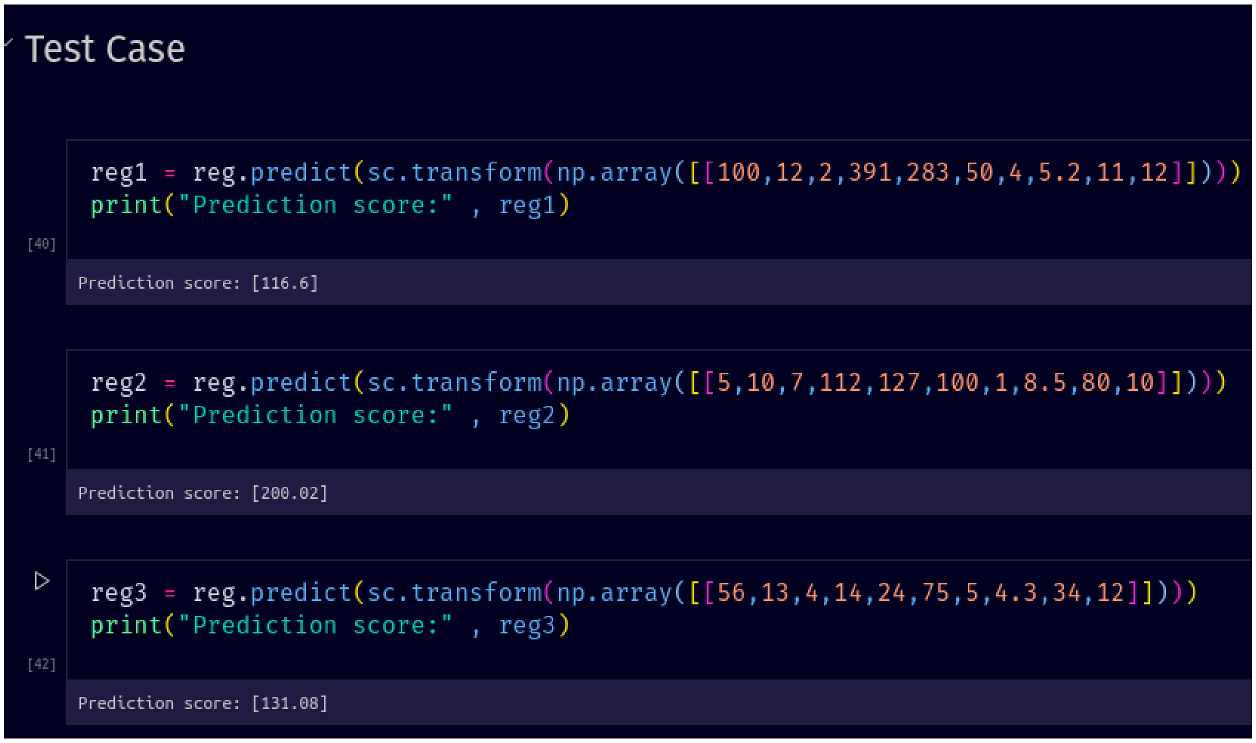
a. ODI:



Test Case

```
In [40]: reg1 = reg.predict(sc.transform(np.array([[100,12,2,391,283,50,4,5.2,11,12]])))
         print("Prediction score:" , reg1)

         Prediction score: [172.83]

In [41]: reg2 = reg.predict(sc.transform(np.array([[5,10,7,112,127,100,1,8.5,80,10]])))
         print("Prediction score:" , reg2)

         Prediction score: [346.31]

In [42]: reg3 = reg.predict(sc.transform(np.array([[56,13,4,14,24,75,5,4.3,34,12]])))
         print("Prediction score:" , reg3)

         Prediction score: [188.65]
```

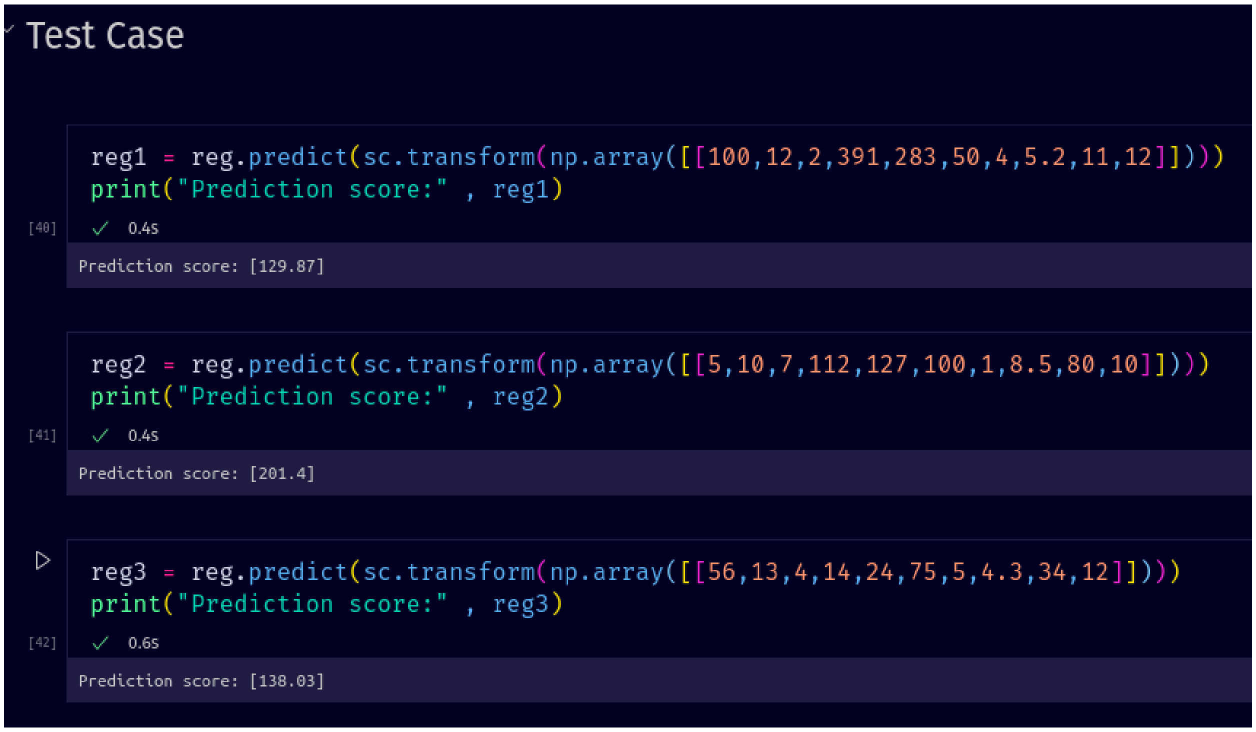
b. IPL:



Test Case

```
reg1 = reg.predict(sc.transform(np.array([[100,12,2,391,283,50,4,5.2,11,12]])))
print("Prediction score:" , reg1)
[40]
Prediction score: [116.6]

reg2 = reg.predict(sc.transform(np.array([[5,10,7,112,127,100,1,8.5,80,10]])))
print("Prediction score:" , reg2)
[41]
Prediction score: [200.02]

reg3 = reg.predict(sc.transform(np.array([[56,13,4,14,24,75,5,4.3,34,12]])))
print("Prediction score:" , reg3)
[42]
Prediction score: [131.08]
```

c. T20:



```
reg1 = reg.predict(sc.transform(np.array([[100,12,2,391,283,50,4,5.2,11,12]])))
print("Prediction score:" , reg1)
```

```
Prediction score: [129.87]
```

```
reg2 = reg.predict(sc.transform(np.array([[5,10,7,112,127,100,1,8.5,80,10]])))
print("Prediction score:" , reg2)
```

```
Prediction score: [201.4]
```

```
reg3 = reg.predict(sc.transform(np.array([[56,13,4,14,24,75,5,4.3,34,12]])))
print("Prediction score:" , reg3)
```

```
Prediction score: [138.03]
```

Our project provides a frictionless method to provide cricket score predictions, within a reasonable error margin and a good degree of accuracy, for multiple types of matches be it T20, IPL, or ODI and if external data is provided any other type of match can also be accommodated with corresponding predicted score/outcome.

PDF of Jupyter Notebooks:

1. IPL Dataset: https://github.com/prakhar0912/AI-J-Comp/blob/master/ipl.pdf
2. ODI Dataset: https://github.com/prakhar0912/AI-J-Comp/blob/master/odi.pdf
3. T20 Dataset: https://github.com/prakhar0912/AI-J-Comp/blob/master/t20.pdf

Github Repo Link with all the code: https://github.com/prakhar0912/AI-J-Comp

8. References:

1. **Numpy & Panda tutorial:** https://cloudxlab.com/blog/numpy-pandas-introduction/#:~:text=Similar%20to%20NumPy%2C%20Pandas%20is,2d%20table%20object%20called%20Dataframe
2. **Panda Tutorial:** https://www.w3schools.com/python/pandas/default.asp
3. **Iris Dataset Problem:** https://medium.com/gft-engineering/start-to-learn-machine-learning-with-the-iris-flower-classification-challenge-4859a920e5e3#:~:text=Every%20iris%20in%20the%20dataset,the%20flower%20considering%20it's%20features.
4. **IPL DataSet:** https://www.kaggle.com/nowke9/ipldata
5. **ODI DataSet:** https://www.kaggle.com/jaykay12/odi-cricket-matches-19712017?select=originalDataset.csv
6. **Unified Datasets:** https://cricsheet.org/downloads/
7. **Iris Classification:** https://www.youtube.com/watch?v=fGRAgibY5N4