Predicting Potential of players in FIFA

1. Introduction

Every year, after the end of season, the transfer window allows different clubs to buy, sell or loan players to/from other clubs

EA Sports' FIFA 19 is the latest version of their football simulation game

FIFA provides ratings of the players based on the performance in the past season, and his potential based on attributes like passing accuracy, dribbling, crossing, finishing, height, weight, etc

1.1 Problem

The aim of this project is to be able to predict the Potential score of a player based on the data present in the dataset

We also want to inspect what attributes factor into determining a soccer players Potential score

The dataset contains the details of players, their nationality, and other attributes such as dribbling, acceleration, stamina, shot accuracy, etc

1.2 Interest

Football clubs around the world want in-depth analysis before putting in a bid for the player in question

The scouting teams from different clubs' scout players extensively before recommending a player to the club

The clubs would, therefore, be very interested in predicting the potential of a player before buying

2. Data Source

The players’ data for FIFA 19 can be found on kaggle.com

The complete dataset was downloaded if form of a CSV file

This dataset contains the players’ details with attributes that would be useful in predicting the Potential score of the player

2.1 Data Cleaning

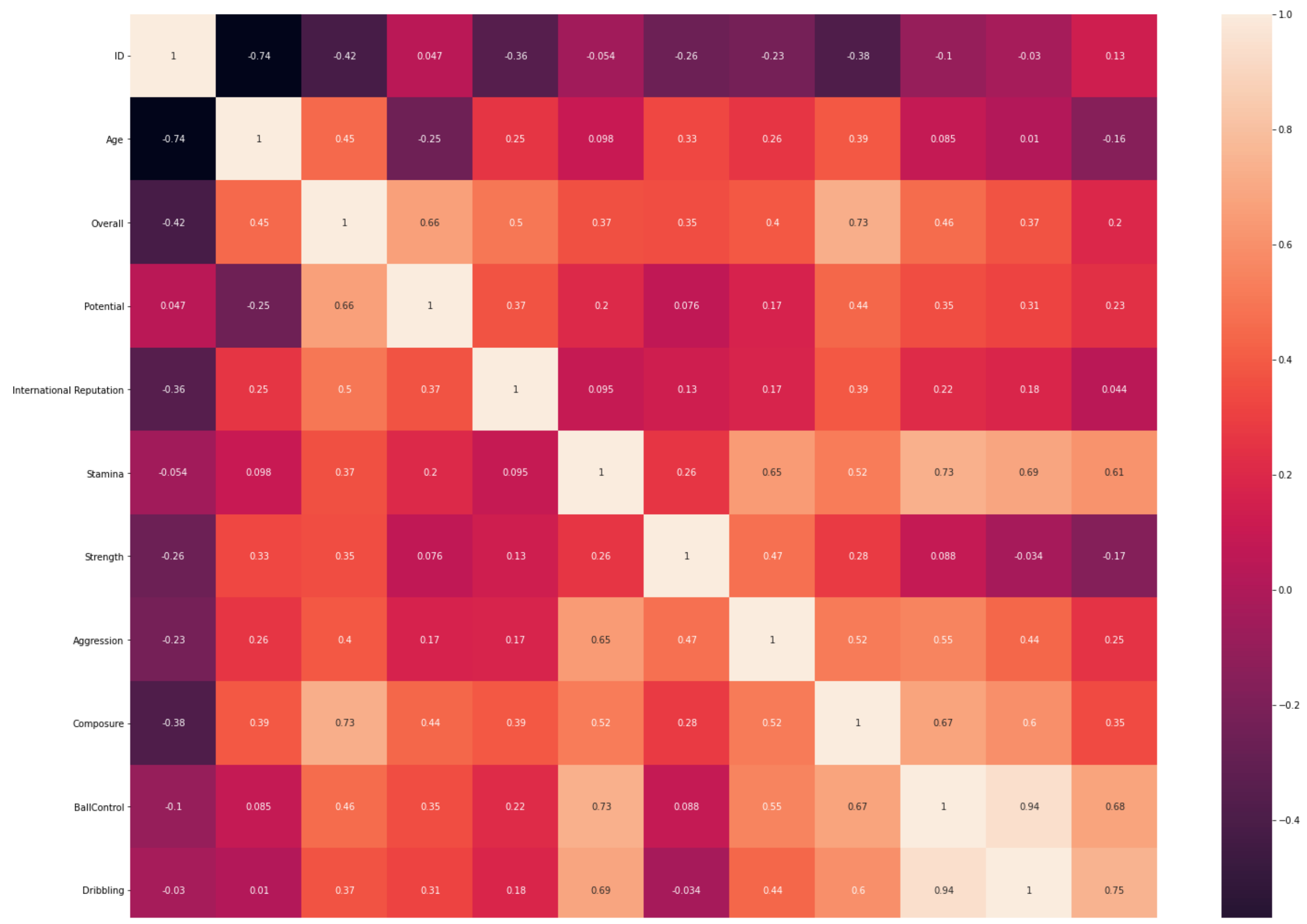
The dataset contains complete details of the players attributes such as age, preferred foot, weak foot, wages, skill moves, crossing, finishing, stamina, header accuracy, shot accuracy, etc

Some of the attributes such as stamina, strength, acceleration have a few null entries

These null entries have been replaced by the mean of the attribute to remove any discrepancy

2.3 Feature Selection

- The following features have been chosen to predict the potential based on correlation heatmap:
- Age
- International Reputation
- Stamina
- Strength
- Aggression
- Composure
- Ball Control
- Dribbling
- Acceleration



3. Exploratory Data Analysis

3.1 Target Vari1able

The potential of a player has been chosen as the target variable

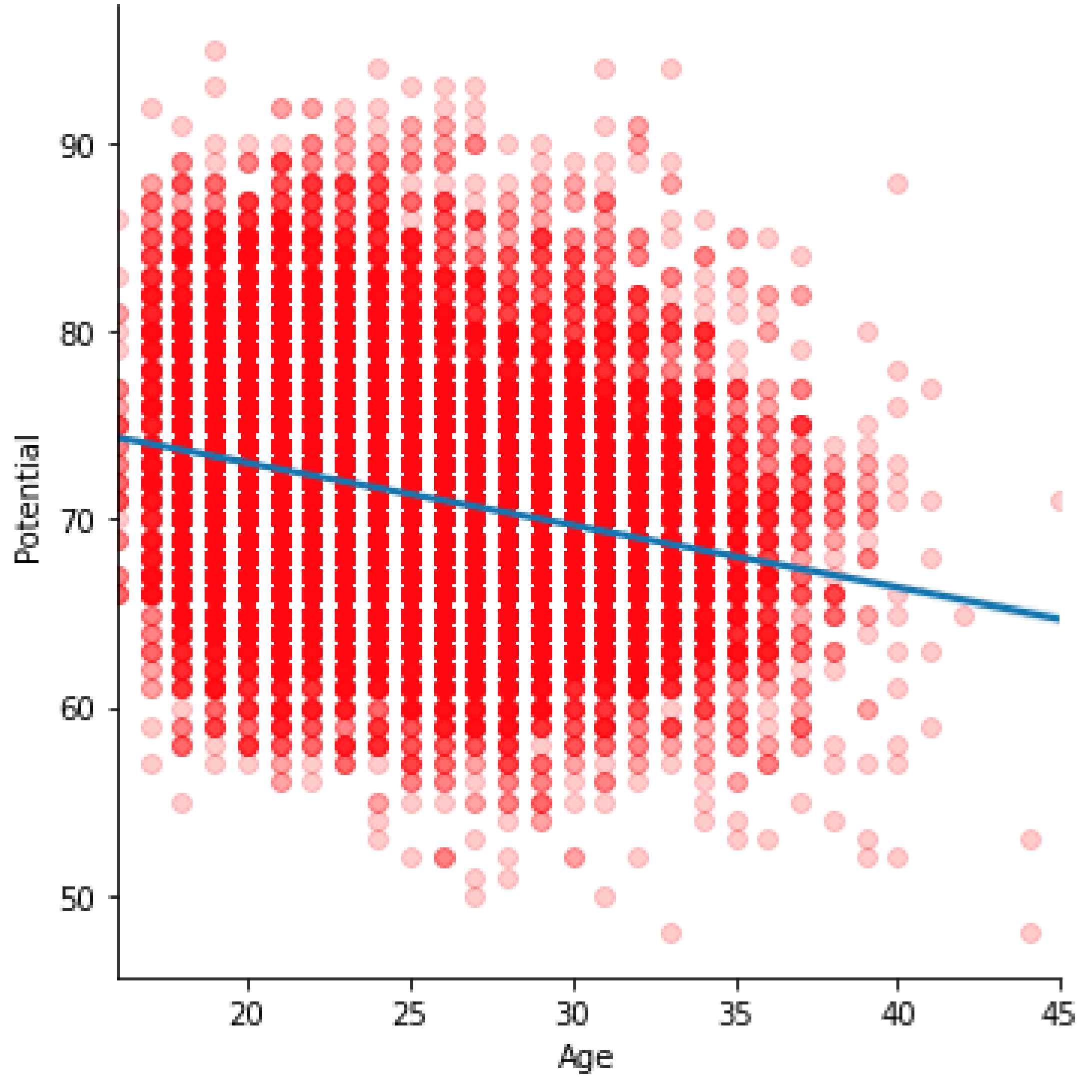
The potential of a player represents how a player would perform keeping in view that the player remains injury free for most the duration of the season

3.3 Potential & Age relationship

The following scatter plot shows the relationship between the target variable 'Potential' and 'Age' attribute

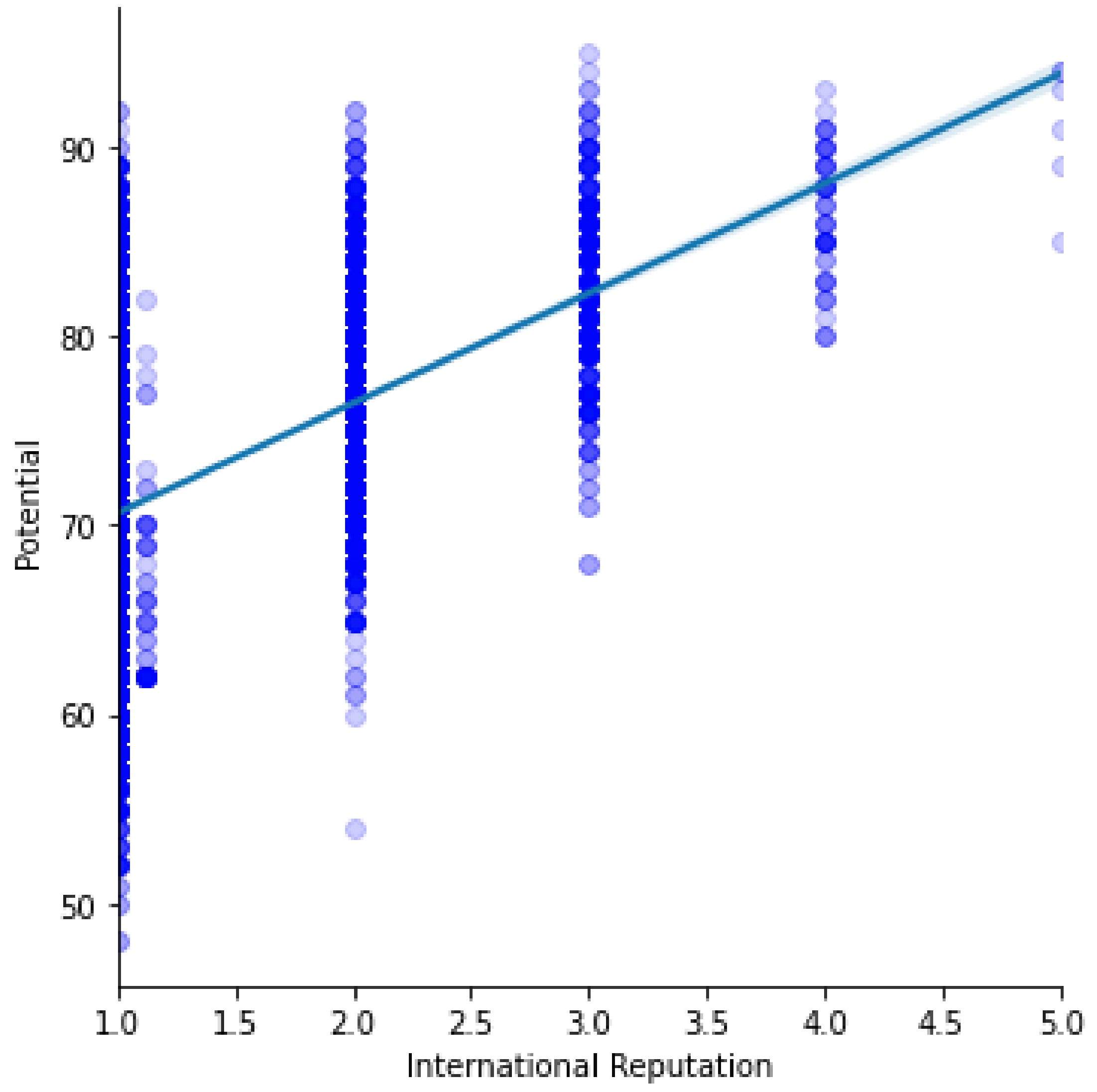
The older a player gets, the less potential they have

This logically makes sense



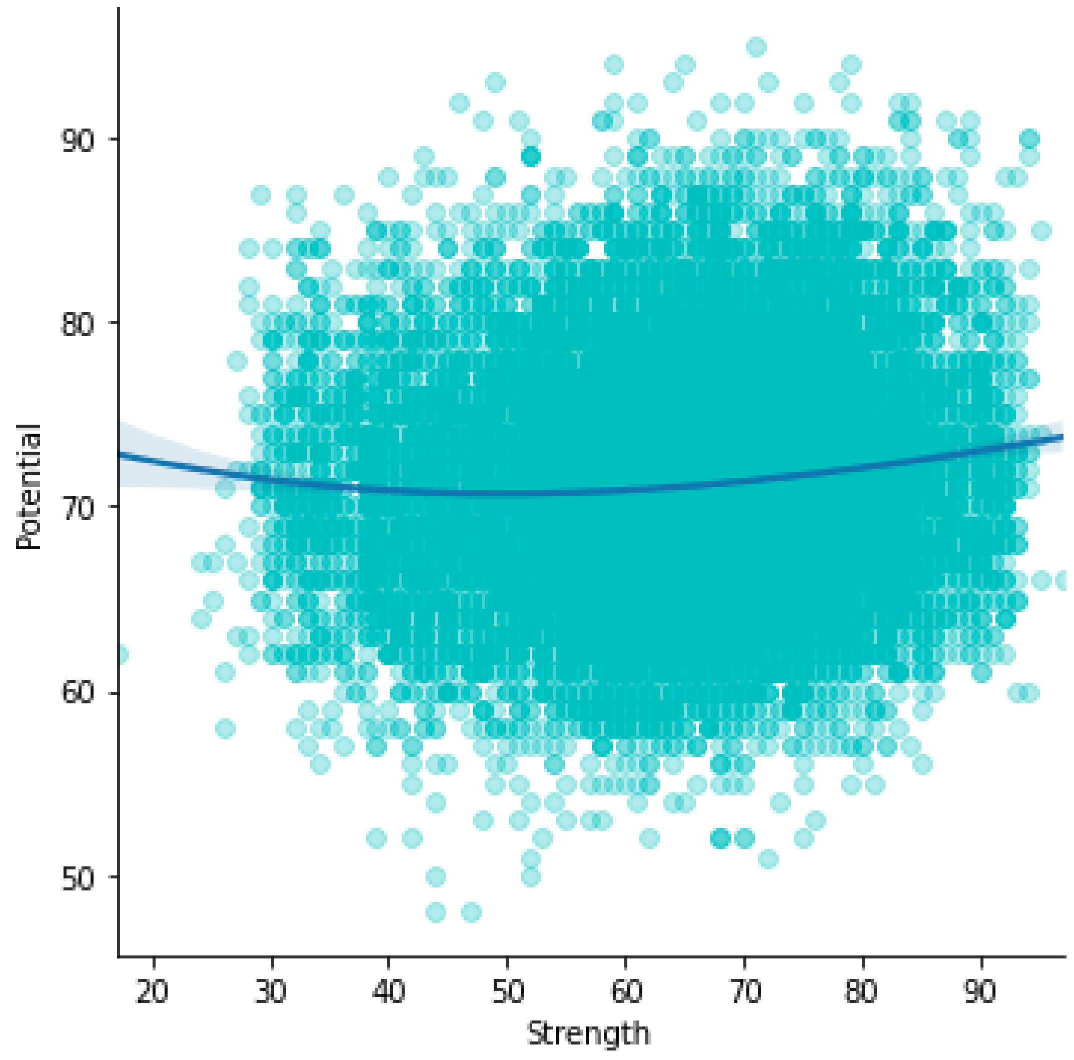
3.4 Potential & International Reputation relationship

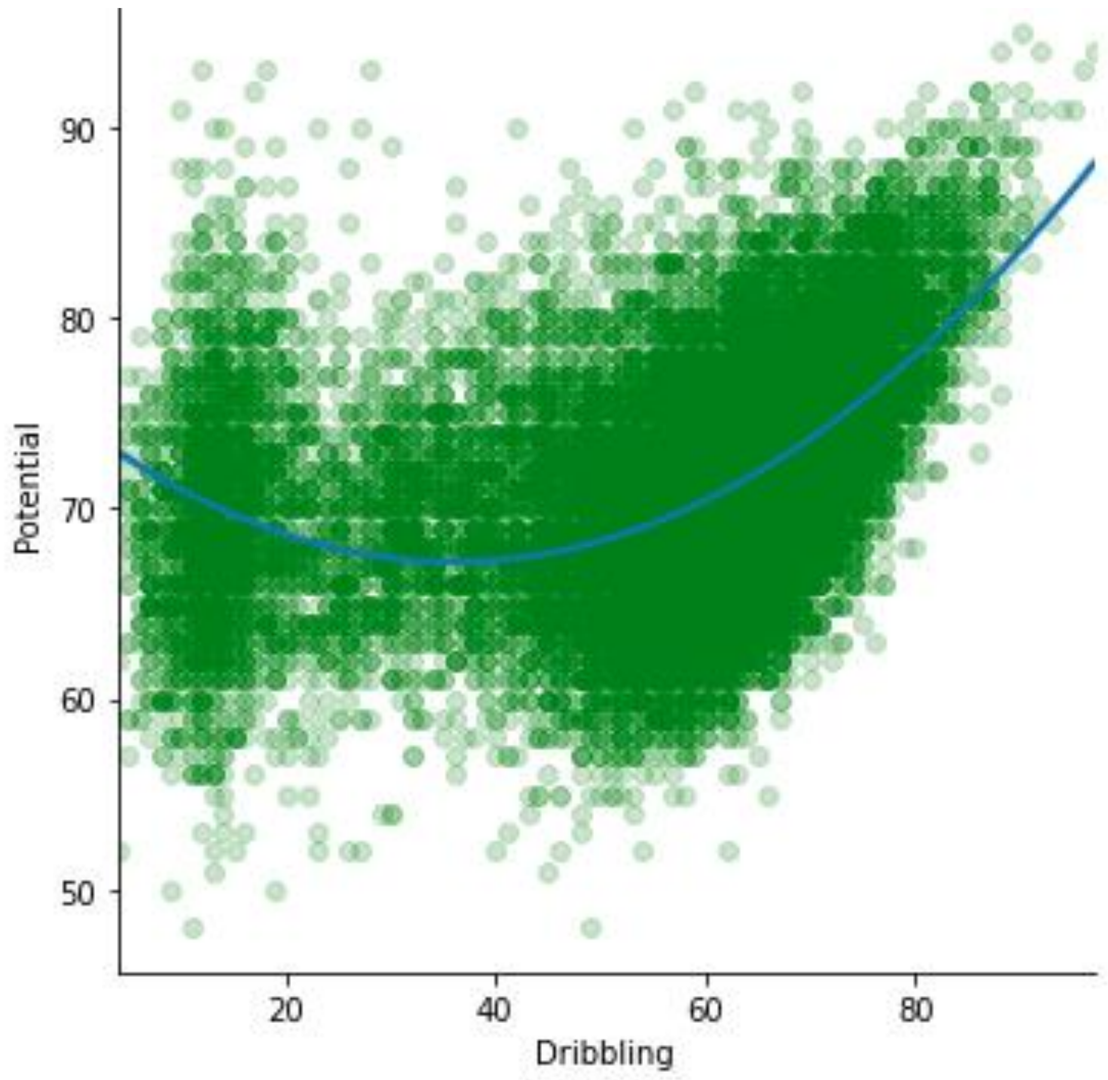
- The following scatter plot shows the relationship between the target variable ‘Potential’ and ‘International Reputation’ attribute



3.5 Potential & Strength relationship

- The following scatter plot shows the relationship between the target variable ‘Potential’ and ‘Strength’ attribute

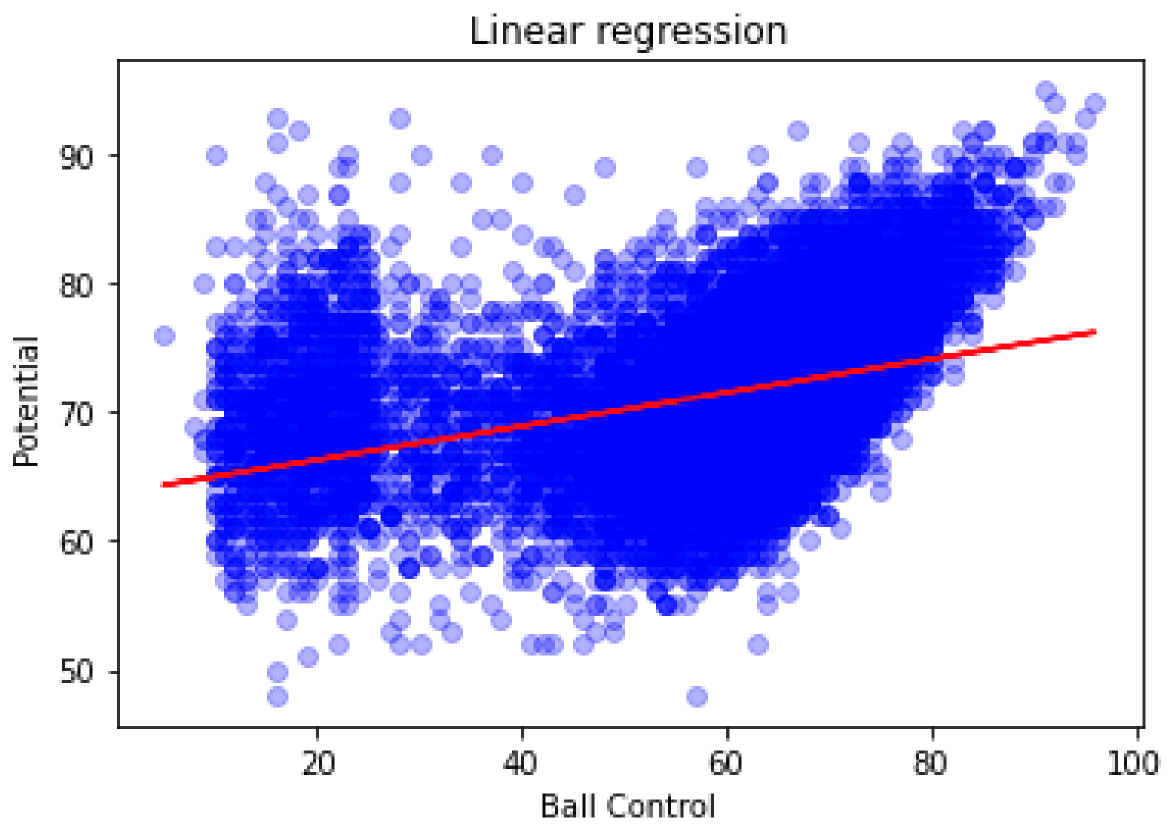




3.6 Potential & Dribbling relationship

- The following scatter plot shows the relationship between the target variable ‘Potential’ and ‘Dribbling’ attribute

3.7 Potential & Ball Control relationship



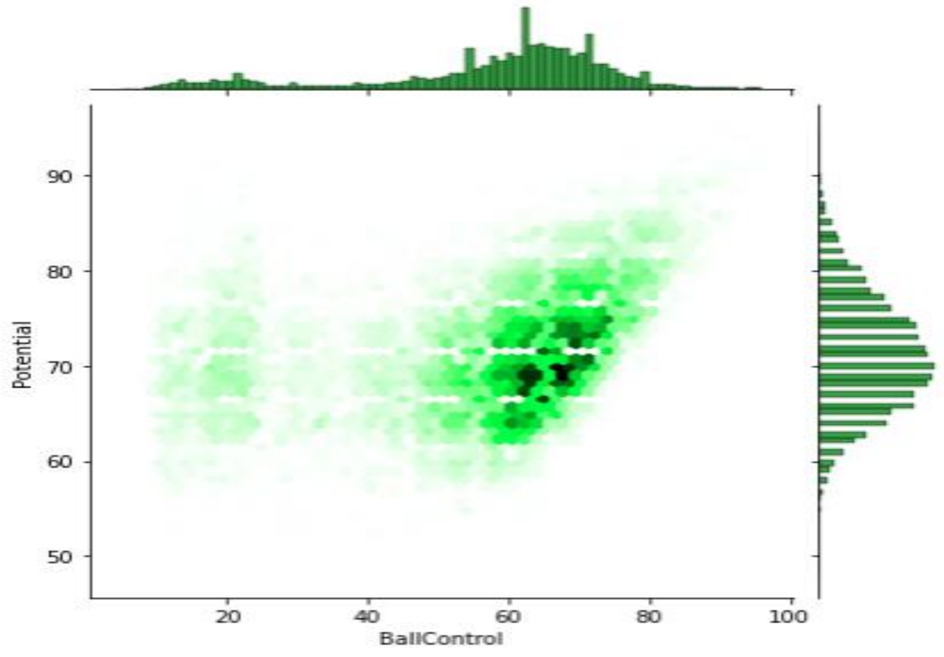
- The following scatter plot shows the relationship between the target variable ‘Potential’ and ‘Ball Control’ attribute

4. Predictive Modelling

I have used Regression models to predict the potential of a player based on other attributes present in the dataset

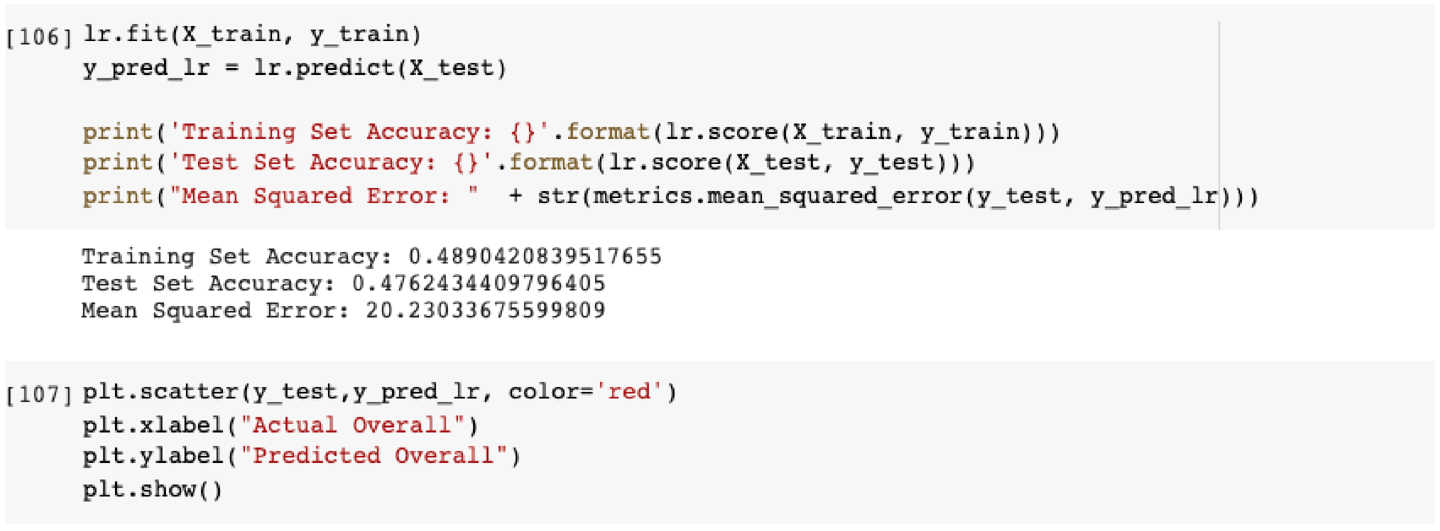
Later, I implement further regression models such as Multiple Regression, Decision Tree Regression, Random Forest, KNN, XGBoost, to predict the Potential and their accuracy has been compared.

4.1. Basic Linear Regression Model



```
Training Set Accuracy: 0.12523800798660079
Test Set Accuracy: 0.12494625557275153
Mean Squared Error: 33.176292294809365
```

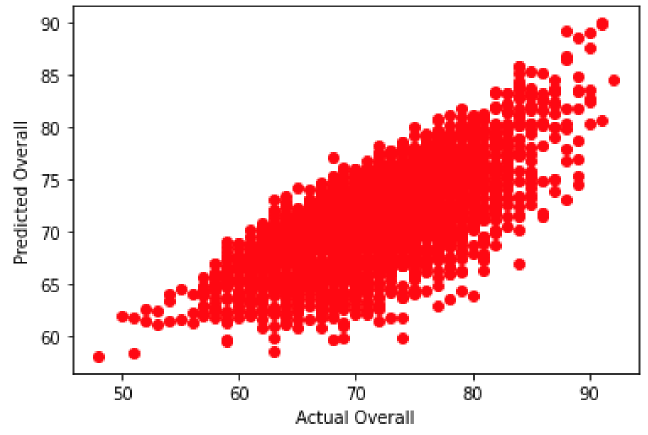
4.2. Multiple Regression Model



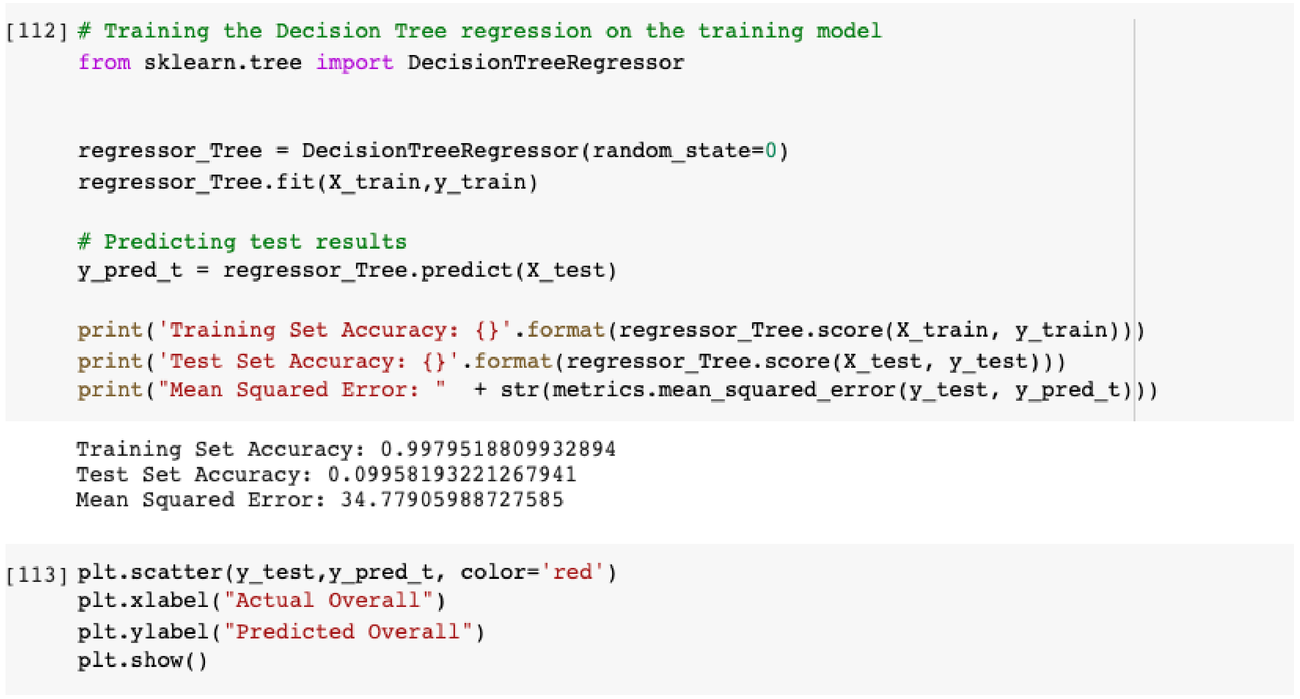
```
[106] lr.fit(X_train, y_train)
      y_pred_lr = lr.predict(X_test)

      print('Training Set Accuracy: {}'.format(lr.score(X_train, y_train)))
      print('Test Set Accuracy: {}'.format(lr.score(X_test, y_test)))
      print("Mean Squared Error: "  + str(metrics.mean_squared_error(y_test, y_pred_lr)))
```

```
Training Set Accuracy: 0.4890420839517655
Test Set Accuracy: 0.4762434409796405
Mean Squared Error: 20.23033675599809
```

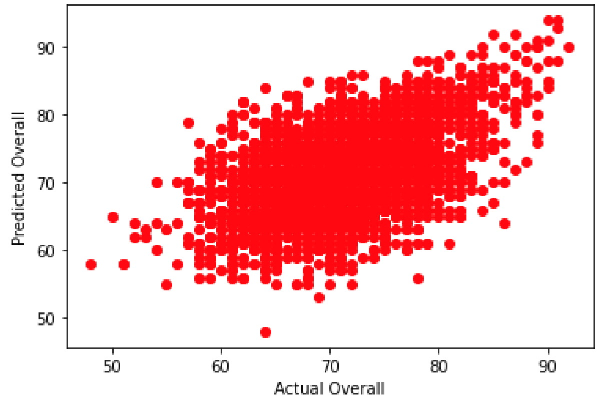
```
[107] plt.scatter(y_test,y_pred_lr, color='red')
      plt.xlabel("Actual Overall")
      plt.ylabel("Predicted Overall")
      plt.show()
```



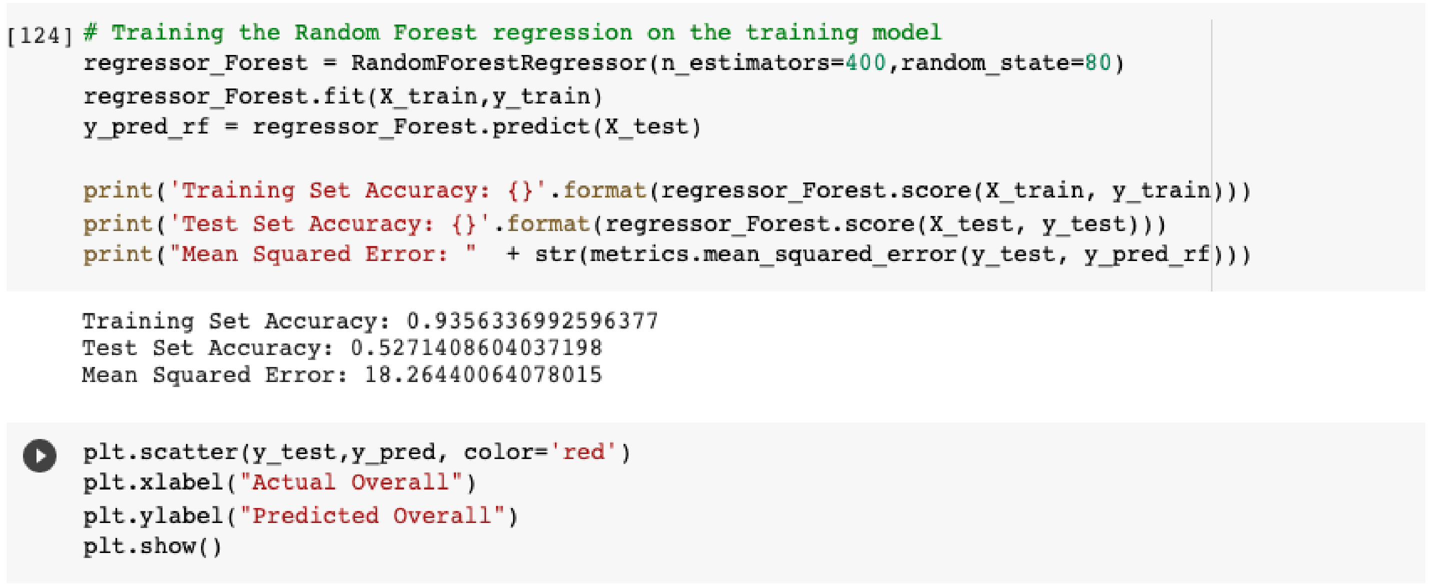
4.3. Decision Tree Regression Model



```
[112] # Training the Decision Tree regression on the training model
      from sklearn.tree import DecisionTreeRegressor


      regressor_Tree = DecisionTreeRegressor(random_state=0)
      regressor_Tree.fit(X_train,y_train)

      # Predicting test results
      y_pred_t = regressor_Tree.predict(X_test)

      print('Training Set Accuracy: {}'.format(regressor_Tree.score(X_train, y_train)))
      print('Test Set Accuracy: {}'.format(regressor_Tree.score(X_test, y_test)))
      print("Mean Squared Error: "  + str(metrics.mean_squared_error(y_test, y_pred_t)))

      Training Set Accuracy: 0.9979518809932894
      Test Set Accuracy: 0.09958193221267941
      Mean Squared Error: 34.77905988727585

[113] plt.scatter(y_test,y_pred_t, color='red')
      plt.xlabel("Actual Overall")
      plt.ylabel("Predicted Overall")
      plt.show()
```



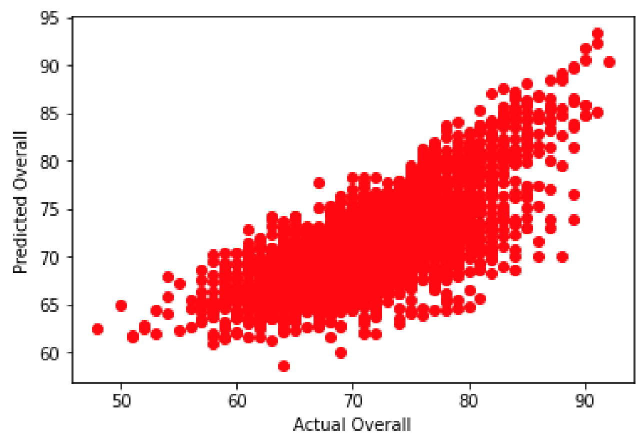
4.4. Random Forest Regression Model



```
[124] # Training the Random Forest regression on the training model
      regressor_Forest = RandomForestRegressor(n_estimators=400,random_state=80)
      regressor_Forest.fit(X_train,y_train)
      y_pred_rf = regressor_Forest.predict(X_test)

      print('Training Set Accuracy: {}'.format(regressor_Forest.score(X_train, y_train)))
      print('Test Set Accuracy: {}'.format(regressor_Forest.score(X_test, y_test)))
      print("Mean Squared Error: "  + str(metrics.mean_squared_error(y_test, y_pred_rf)))
```

```
Training Set Accuracy: 0.9356336992596377
Test Set Accuracy: 0.5271408604037198
Mean Squared Error: 18.26440064078015
```

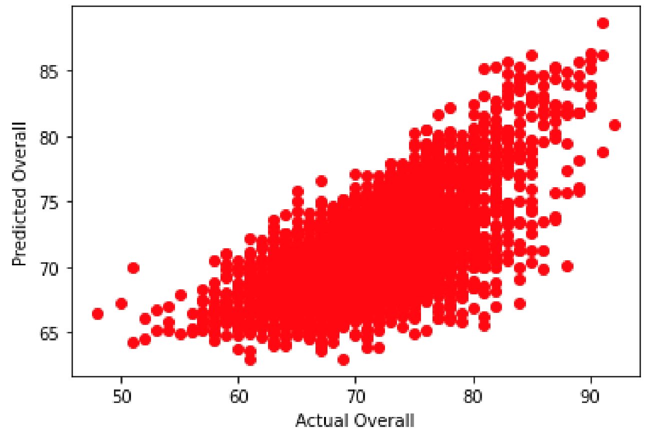
```
plt.scatter(y_test,y_pred, color='red')
plt.xlabel("Actual Overall")
plt.ylabel("Predicted Overall")
plt.show()
```



```
[118] # Training the KNN model on the training set
      regressor_knn = KNeighborsRegressor(n_neighbors=6)
      regressor_knn.fit(X_train,y_train)
      y_pred_knn = regressor_knn.predict(X_test)

      print('Training Set Accuracy: {}'.format(regressor_knn.score(X_train, y_train)))
      print('Test Set Accuracy: {}'.format(regressor_knn.score(X_test, y_test)))
      print("Mean Squared Error: "  + str(metrics.mean_squared_error(y_test, y_pred_knn)))
```

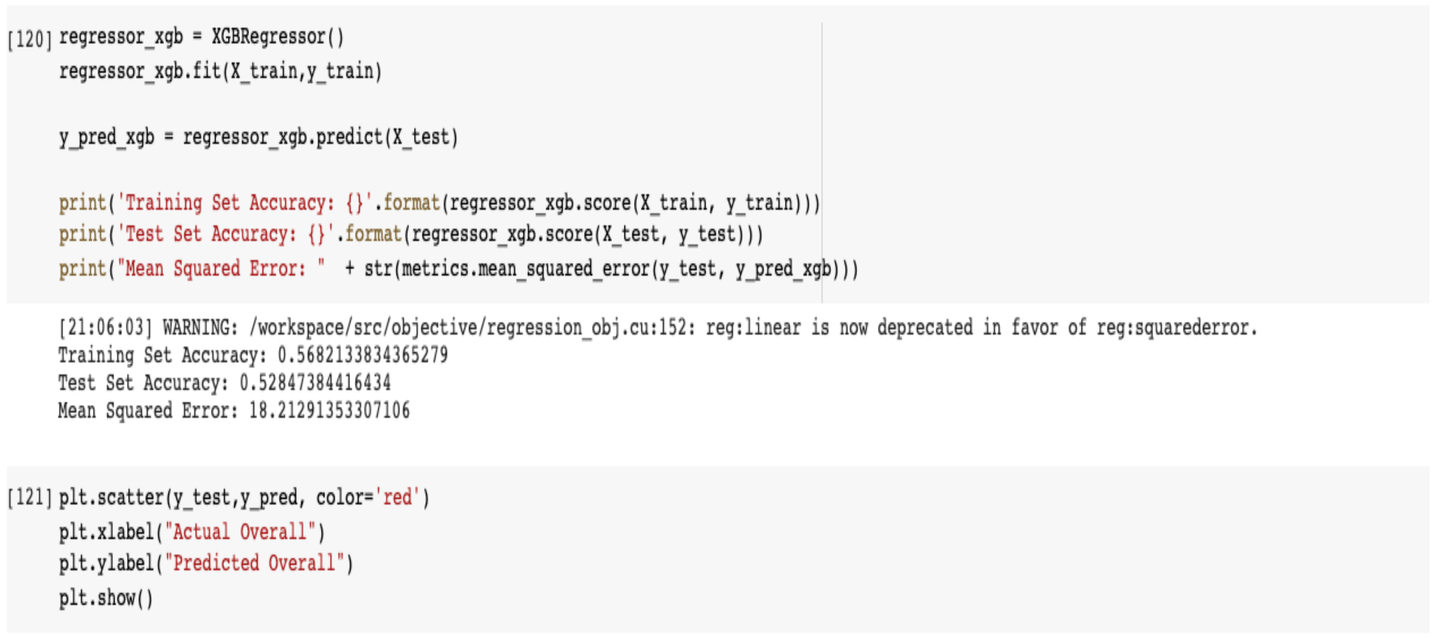
```
Training Set Accuracy: 0.5959705037057642
Test Set Accuracy: 0.41224000838327446
Mean Squared Error: 22.70249862712795
```

```
plt.scatter(y_test,y_pred, color='red')
plt.xlabel("Actual Overall")
plt.ylabel("Predicted Overall")
plt.show()
```



4.5. K-Nearest Neighbors Regression Model

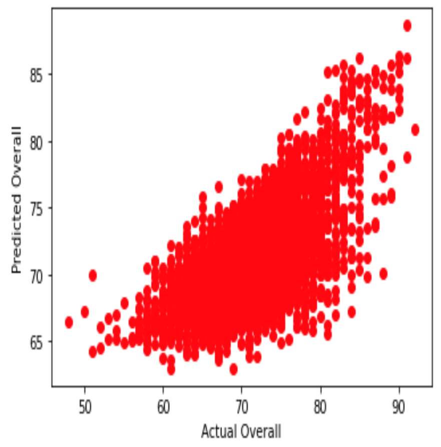
4.6. XGBoost Regression Model



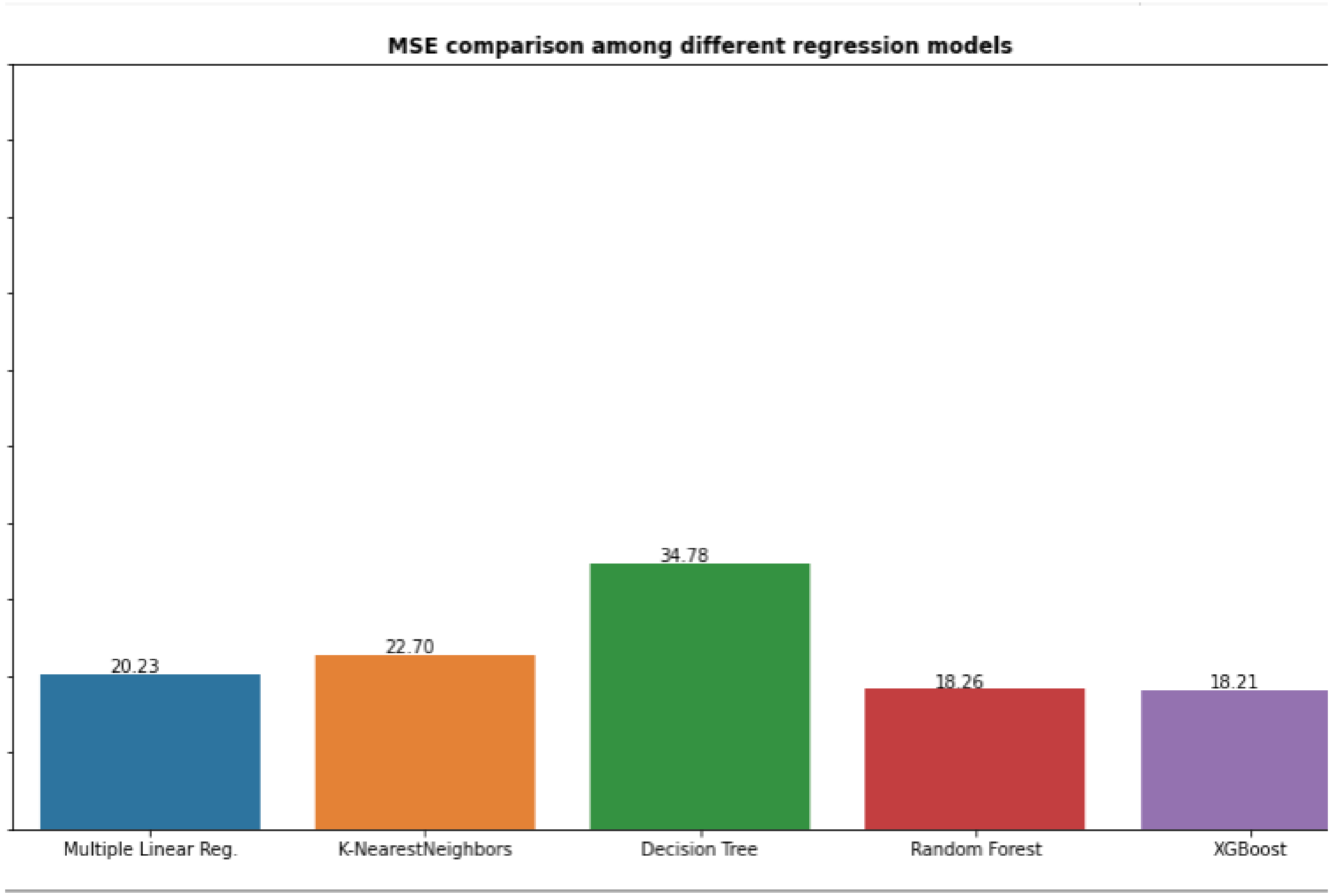
```
[120] regressor_xgb = XGBRegressor()
      regressor_xgb.fit(X_train,y_train)

      y_pred_xgb = regressor_xgb.predict(X_test)

      print('Training Set Accuracy: {}'.format(regressor_xgb.score(X_train, y_train)))
      print('Test Set Accuracy: {}'.format(regressor_xgb.score(X_test, y_test)))
      print("Mean Squared Error: "  + str(metrics.mean_squared_error(y_test, y_pred_xgb)))
```

```
      [21:06:03] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is now deprecated in favor of reg:squarederror.
      Training Set Accuracy: 0.5682133834365279
      Test Set Accuracy: 0.52847384416434
      Mean Squared Error: 18.21291353307106
```

```
[121] plt.scatter(y_test,y_pred, color='red')
      plt.xlabel("Actual Overall")
      plt.ylabel("Predicted Overall")
      plt.show()
```



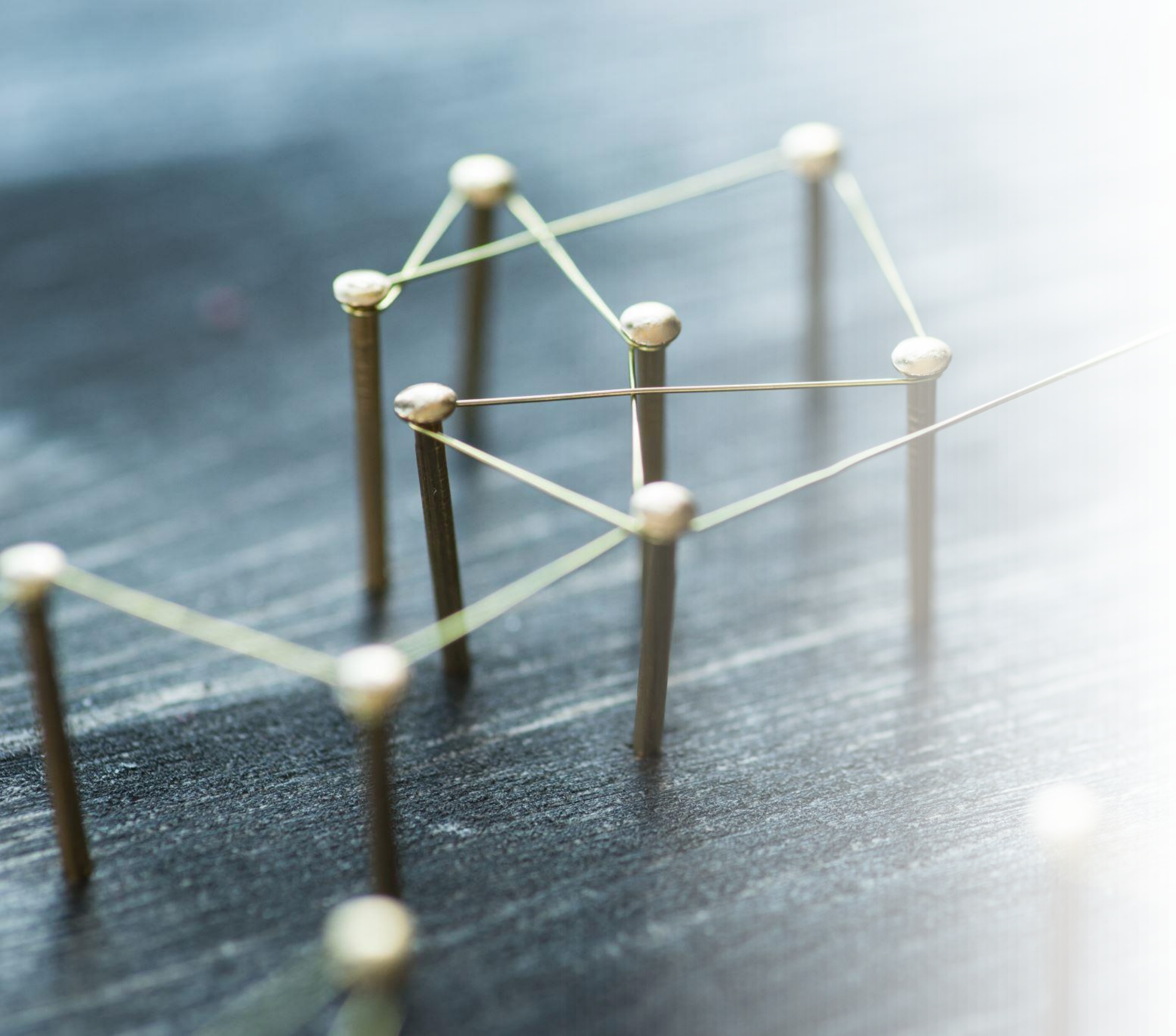
5. Conclusion



After looking at all our models, I would conclude that using either the Random Forest Regressor model or XGBoost Regressor model would be the best in predicting Player Potential

These models have the highest accuracy and lowest MSE out of all the models we tested

Even though the highest accuracy score we achieved was approximately 53%, accuracy isn't what is important here



6. Discussion

6.1 Further Modeling

- This model can be further analyzed using clustering algorithms to create clusters of players with a certain potential
- For example, players with a potential greater than 95 can be clustered into a ‘special’ category, while players with potential between 90 and 94 can be categorized as ‘exciting’, and so on
- More complicated soccer statistics are very hard to find without having membership access to them
- Being able to access those statistics might give us better insights into what other statistics go into player potential