**I will continue the next part**

del train\_data['player\_id']

[Now,**deleting data from the player-id from the train data]**

features = list(train\_data.columns)

features.remove('response')

X\_train, y\_train = train\_data[features].values, train\_data['response'].values

print(X\_train.shape)

print(y\_train.shape)

**[Then we make a list the train data columns into a feature. After that we remove the response from the feature.] Then X-train are testing the values of the features of train data.**

**Now, Y-train is testing the values of the response of train data. Finally, we are printing x-train and y-training and the row for x-train is 3486, and column for y-train is 42.**

**Here, the shape represents the value of x-axis and y-axis.]**

scaler = StandardScaler()

scaler.fit(X\_train)

X\_train = scaler.transform(X\_train)

print(X\_train)

**[here, we call standard scaler function and save the values into scaler. using standard scaler function, we can transform the data and find the mean and standard deviation value.**

**Then we fit x trained data and transform them.]**

model1 = LinearRegression()

model1.fit(X\_train, y\_train)

[**we call** LinearRegression() **function [**Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x).

So, this regression technique finds out a linear relationship between x (input) and y(output).**] Then we fit x trained and y-train data and transform them.]**

player\_ids = test\_data['player\_id'].values

del test\_data['player\_id']

X\_test, y\_test = test\_data[features].values, test\_data['response'].values

X\_test = scaler.transform(X\_test)

[**here, we test the values of players id. Then delete the test data of players id.** **Then we fit x-test data and transform them. Then we removing players id from test data**

**After that we remove the response from the train data.] We are testing the x-test and y-test value. Then we transform the x-test data. After that we predict to x-test value.**

def model\_to\_predictions(model):

predictions = model.predict(X\_test)

predictions = [[player\_ids[i], predictions[i], y\_test[i]] for i in range(len(player\_ids))]

predictions = pd.DataFrame(predictions, columns=['id', 'prediction', 'actual']).set\_index('id')

predictions = predictions.join(players, on='id')

return predictions

model\_to\_predictions(model1).sort\_values('prediction', ascending=False).head(40)

[**Then we call prediction function. Then in this function, we predict x-test. We play a loop to get players in the predictions. Then in the prediction, we take new columns: id, prediction, actual.**

**And set the id turn into index id. Then we join prediction into players and id. Then we sort the prediction for 40 values.]**

model1.score(X\_test, y\_test)

**[after that we can score the x-test and y-test as 0.26]**

model2 = RandomForestRegressor(n\_estimators=500, max\_depth=10, min\_samples\_split=2, min\_samples\_leaf=5)

[**for model2, we take estimators=500, max\_depth=10, min\_samples\_split=2, min\_samples\_leaf=5 for RandomForestRegressor to optimal prediction.**

model2.fit(X\_train, y\_train)

[).**] Then we fit x trained and y-train data for model2.]**

model\_to\_predictions(model2).sort\_values('prediction', ascending=False).head(40)

[**Then we sort the prediction for 40 values.]**

model2.score(X\_test, y\_test)

**after that we can the predicted score the x-test and y-test as .26**