FML Assignment Report

Predicting the price of a football

A Course Work Report Submitted

In Partial Fulfillment of the Requirements

For the Degree of

B.Tech

In

CSC Department

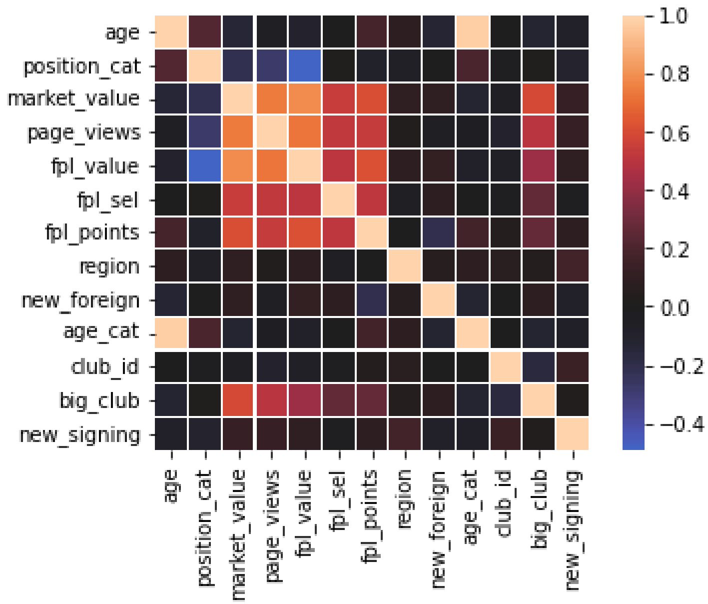
Harshit

190C2020030

BML MUNJAL UNIVERSITY™

School of Engineering and Technology BML Munjal University
Gurgaon December, 2021

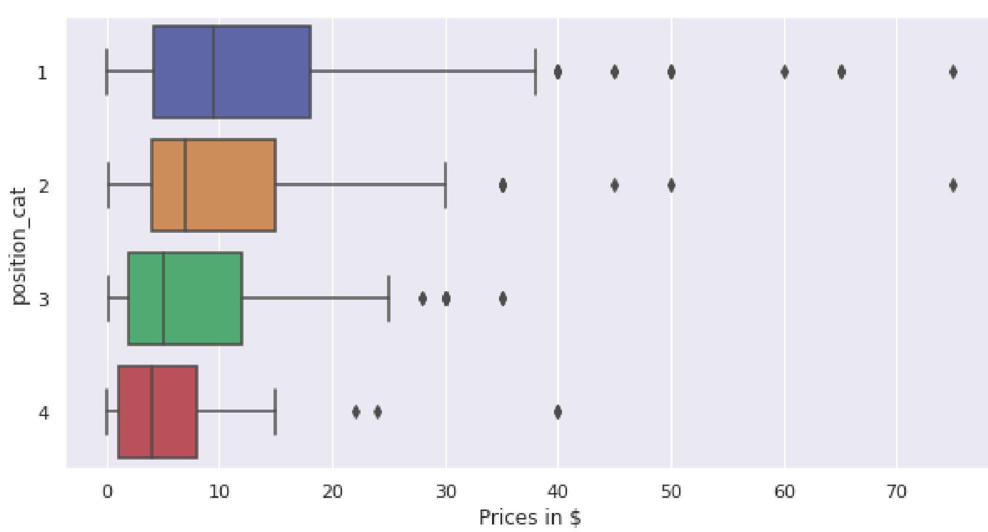
Part- 1
Heat Map for correlation of different features:



In the above heatmap, the lighter the color, the more the correlation between the features. Notice that for higher market value, page_views, fpl_value and fpl_ points matters the most because they have very bright colors.

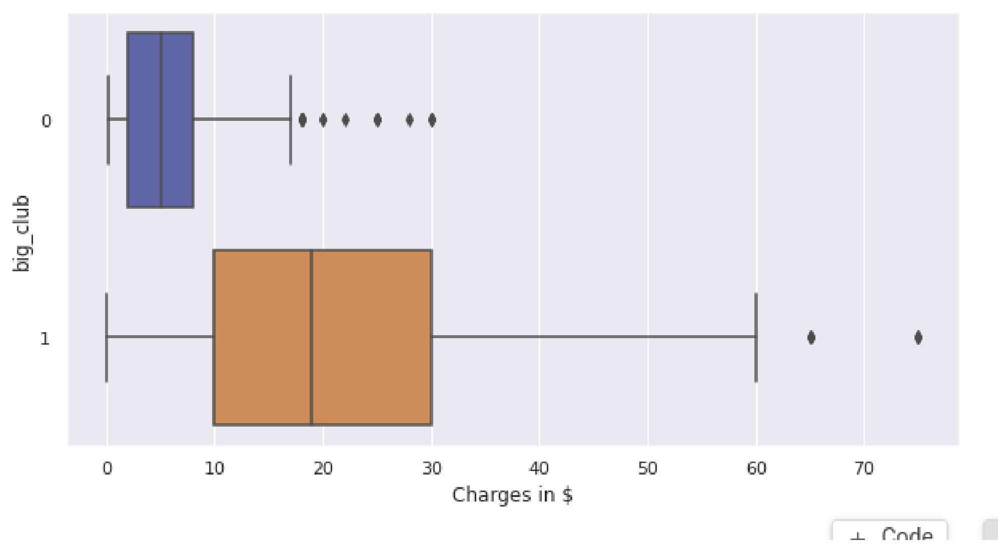
Box Plots:

Position Category vs Prices



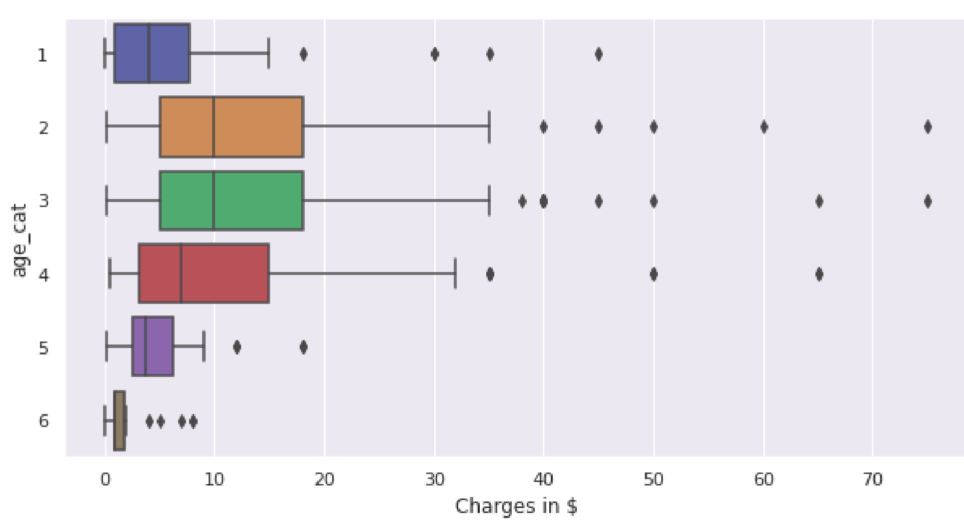
From the above box plot we can see that the players with higher salary are less in numbers and are considered as outliers. And they should be exempted while doing the predictions. If we will including these values our prediction will go wrong drastically. We can also observe that the attackers have the most number of higher charges.

2. Big club vs charges



From the above box plot we can see that the club which tops on ranking earns the most, i.e they have the highest amount of charges.

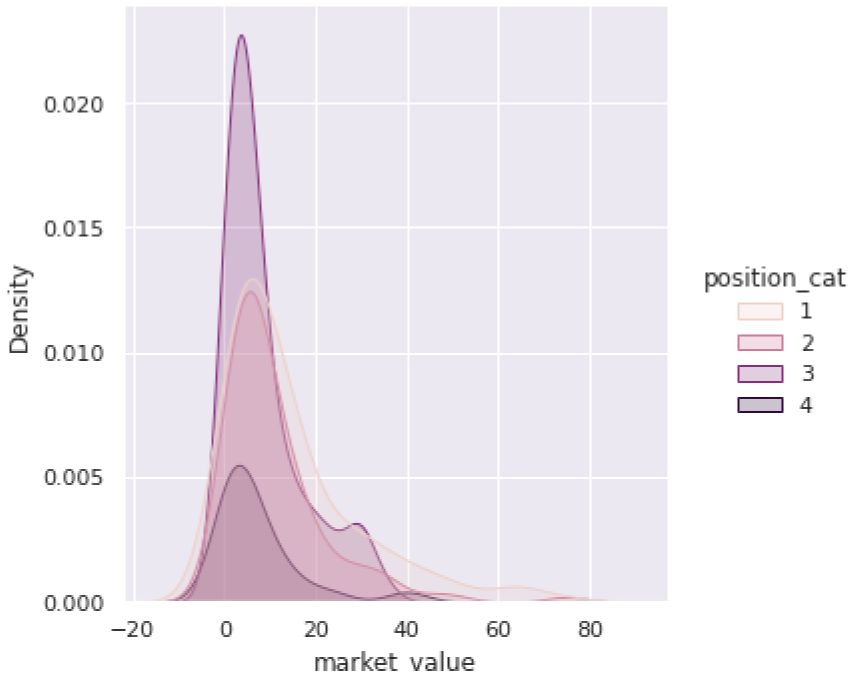
3.age_Cat vs charges



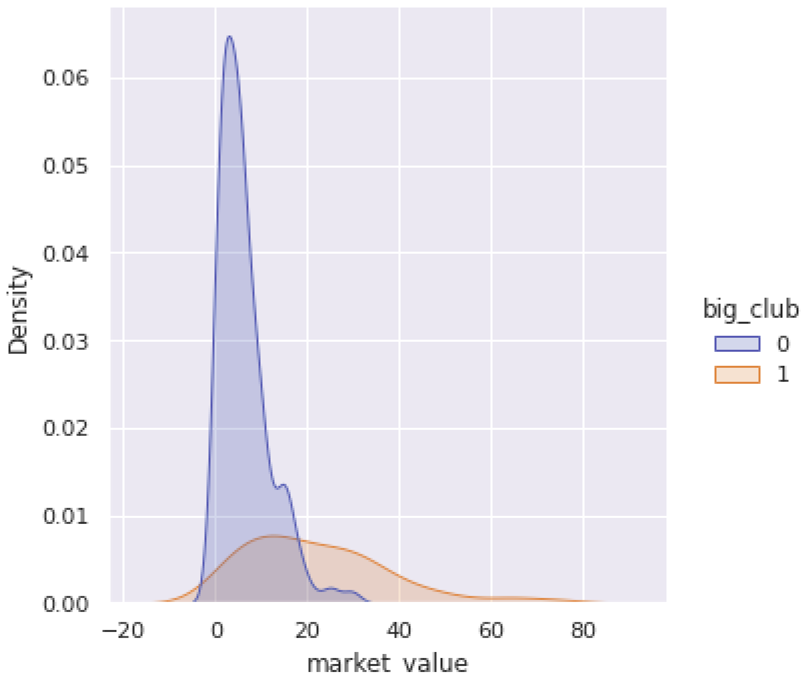
From the above box plot we can see that the charges for most of the players lies from 5-20 for players whose age lies between 22 to 30. The higher the age, the less the charges for that player.

Player who’s age is between 22 to 28 has the most higher charges i.e more no. of outliers lie in this age group.

Various other plots for visualization



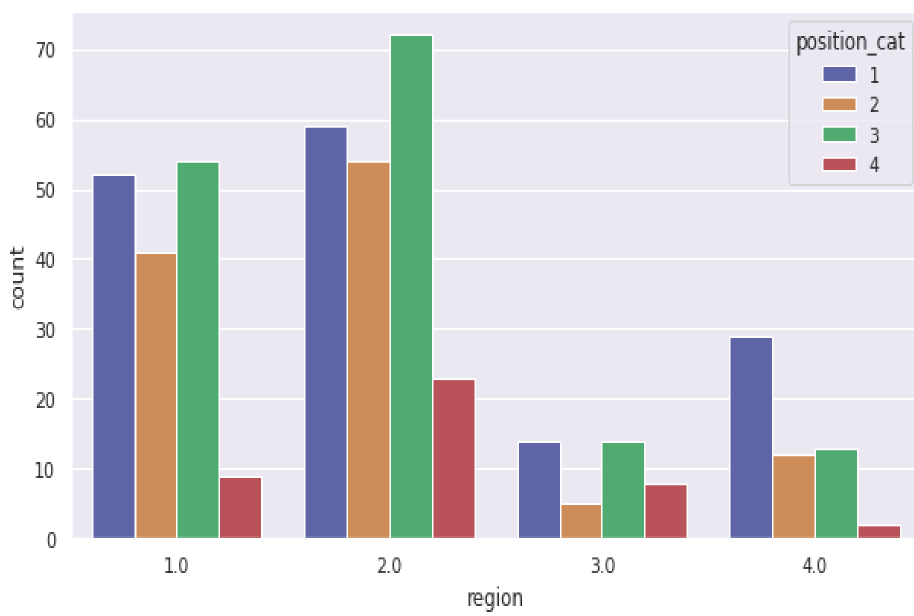
From the above displot we can see that the highest number of position_cats market value lies in 0 to 20 and secondly the defenders have the highest den



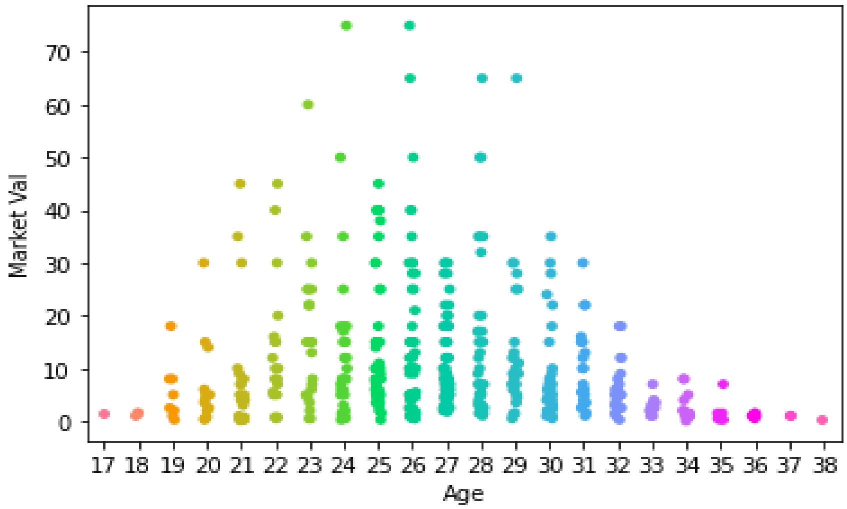
From the above displot we can see that the clubs which are not in top 6 rank have lower market_value than the clubs which are in top 6 and are concentrated in the market_value range of 0 to 20.

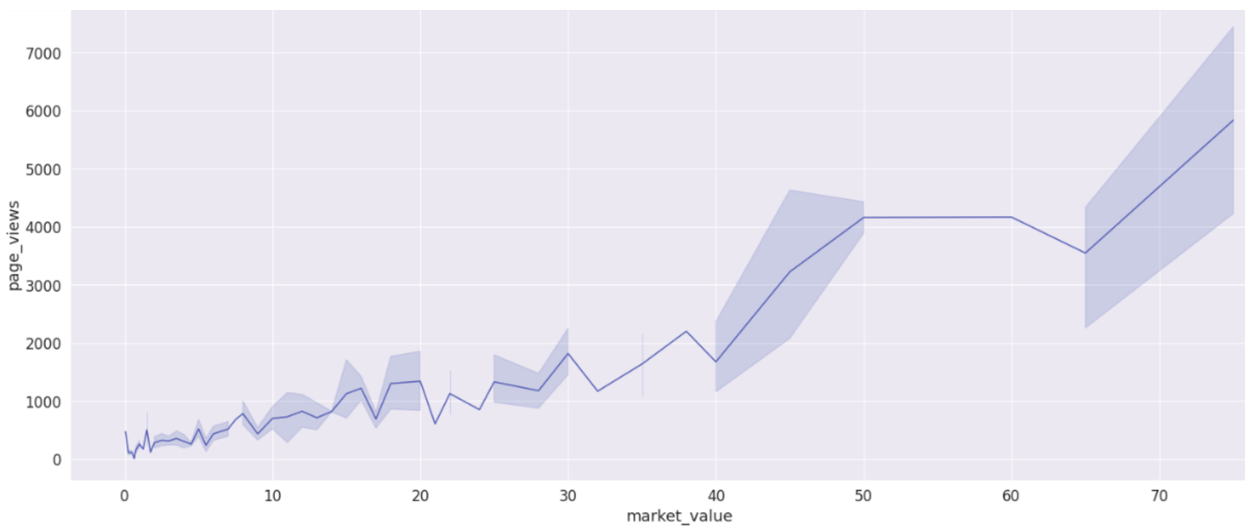
Some more plots for better visualization of data

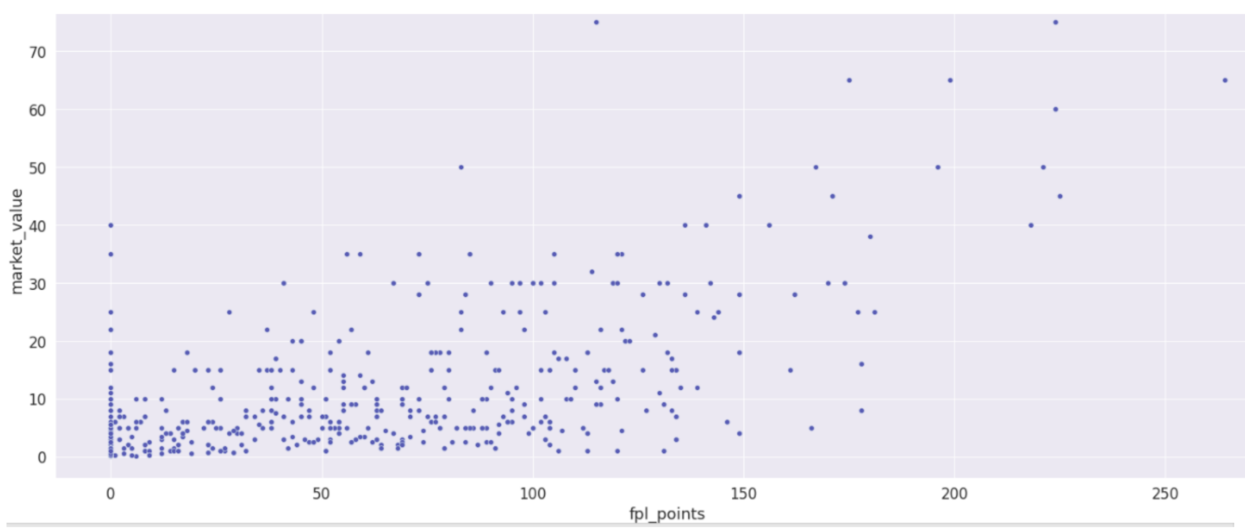
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd0866dc790>
```



```
[Text(0, 0.5, 'Market Val'), Text(0.5, 0, 'Age')]
```







Models mentioned in the questions are built using sklearn. Their execution and results are as follows:

Part- 2&3

Linear Regression

```
[62]
linear_reg = LinearRegression()
run_CrossValidation(linear_reg,X,Y)
```

```
Mean: 0.6753747991296591, Std: 0.155843202298518, Min: 0.37360151208566117, Max: 0.8148228077247442
```

Lasso Regressor

Lasso Regressor

```
[63]
lasso_params = {'alpha':[0.02, 0.03, 0.01, 0.1, 0.05, 0.5, 0.2, 0.3, 0.25], 'max_iter': [100,1000, 500, 5000, 200]}
lasso_reg = Lasso()
lasso_reg = hyperParameter(lasso_reg, lasso_params, X, Y)
```

```
Best Score:  0.6812413321438472
Best Params:  {'alpha': 0.2, 'max_iter': 100}
Cross validation on model with best params
Mean: 0.6812413321438472, Std: 0.13200194572154217, Min: 0.4340098482152186, Max: 0.8171582084239042
```

Ridge Regressor

```
[64]
ridge_params = {'alpha':[0.01, 0.1, 0.5, 1, 5, 10, 20, 25, 30, 100]}
ridge_reg = Ridge()
ridge_reg = hyperParameter(ridge_reg, ridge_params, X, Y)
```

```
Best Score:  0.6827599266185473
Best Params:  {'alpha': 20}
Cross validation on model with best params
Mean: 0.6827599266185473, Std: 0.12985710734018865, Min: 0.437735853307734, Max: 0.8137127804933832
```

KNeighborsRegressor

```
[65]
n_params = {'n_neighbors': [1,2,3,4,5,6,7,8,9,10], 'weights': ['uniform', 'distance'], 'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'], 'n_jobs': [-1]}
neigh = KNeighborsRegressor()
neigh = hyperParameter(neigh, n_params, X, Y)
```

```
Best Score:  0.47067905679853184
Best Params:  {'algorithm': 'auto', 'n_jobs': -1, 'n_neighbors': 8, 'weights': 'uniform'}
Cross validation on model with best params
Mean: 0.47067905679853184, Std: 0.19743170854894004, Min: 0.08856843158218541, Max: 0.6584860384646545
```

Support Vector Regressor

```
[66]
param = {'kernel' : ('linear', 'poly', 'rbf', 'sigmoid'),'C' : [1,5,10],'degree' : [3,8],'coef0' : [0.01,10,0.5],'gamma' : ('auto','scale'), 'max_iter': [100, 10
svr = SVR()
svr = hyperParameter(svr, param, X_mms, Y.values.ravel())
```

```
Best Score:  0.7192476506560381
Best Params:  {'C': 5, 'coef0': 10, 'degree': 3, 'gamma': 'auto', 'kernel': 'poly', 'max_iter': 2000}
Cross validation on model with best params
Mean: 0.7192476506560381, Std: 0.11864754406551849, Min: 0.4956130483257061, Max: 0.822573717695521
```

Decision Tree Regressor

```
param = {'criterion' :['mse', 'friedman_mse', 'mae'], 'splitter':['best', 'random'], 'max_depth': [1,2,3,4,5,6,7,8,9,10], 'max_features': ['auto',
tree = DecisionTreeRegressor()
tree = hyperParameter(tree, param, X_mms, Y)
```

```
Best Score:  0.6133032560386553
Best Params:  {'criterion': 'mse', 'max_depth': 7, 'max_features': 'auto', 'splitter': 'random'}
Cross validation on model with best params
Mean: 0.5677811188542468, Std: 0.14263882011586768, Min: 0.32399109844815976, Max: 0.728651034265929
```

Random Forest Regressor

```
param = {
 'max_depth': [10, 50, 90, 100],
 'max_features': ['auto', 'sqrt'],
 'n_estimators': [100,1000,2000]}
rfr = RandomForestRegressor()
rfr = hyperParameter(rfr, param, X_ss, Y.values.ravel())
```

```
Best Score:  0.7393239407974654
Best Params:  {'max_depth': 50, 'max_features': 'sqrt', 'n_estimators': 100}
Cross validation on model with best params
Mean: 0.7203383567504688, Std: 0.08593885301881521, Min: 0.578383343330517, Max: 0.8357472041181881
```

Gradient Boosting Regressor

```
param = {
'n_estimators': [500, 900, 1000],
'max_depth': [2,3, 5, 10, 15],
'max_features': ['auto', 'sqrt', 'log2']}
gbr = GradientBoostingRegressor()
gbr = hyperParameter(gbr, param, X_mms, Y.values.ravel())
```

```
Best Score:  0.7383142821634368
Best Params:  {'max_depth': 2, 'max_features': 'log2', 'n_estimators': 500}
Cross validation on model with best params
Mean: 0.7400364925053476, Std: 0.03989609746657331, Min: 0.6846820427087643, Max: 0.7796071177751461
```

Part - 4

Our best model after hyper parameter tuning is Random Forest Regressor with a score of 0.7393

Best Paramteres : {'max_depth': 50, 'max_features': 'sqrt', 'n_estimators': 100}

Part 5 :

Implement a Genetic Algorithm for learning attribute weights for the Nearest Neighbour

Algorithm. Implement at least one mechanism for maintaining Diversity within the

Population

Code :

```
import csv
import random as rand
import math
import operator


def manhattan(a, b,length):
   return sum(abs(val1-val2) for val1, val2 in zip(a,b))


def getNeighbors(trainingSet, testInstance, k):
   distances = []
   length = len(testInstance) - 1
   for x in range(len(trainingSet)):
       dist = manhattan(testInstance, trainingSet[x], length)
       distances.append((trainingSet[x], dist))
   distances.sort(key=operator.itemgetter(1))
   neighbors = []
   for x in range(min(k, len(distances))):
       neighbors.append(distances[x][0])
   return neighbors


def getResponse(neighbors):
   classVotes = {}
   for x in range(len(neighbors)):
       response = neighbors[x][-1]
       if response in classVotes:
           classVotes[response] += 1
       else:
           classVotes[response] = 1
   sortedVotes = sorted(classVotes.items(), key=operator.itemgetter(1),
reverse=True)
   return sortedVotes[0][0]


def getAccuracy(testSet, predictions):
```

```python
    correct = 0
    for x in range(len(testSet)):
        if testSet[x][-1] == predictions[x]:
            correct = correct+1
    return (correct / float(len(testSet))) * 100.0


def generateChromosome(chromosome):
    return [rand.randint(1, 100) for x in range(chromosome)]


def desimal(biner):
    return int(biner, 2)


def kNN(k, testSet, trainingSet):

    predictions = []
    for x in range(len(testSet)):
        neighbors = getNeighbors(trainingSet, testSet[x], k)
        result = getResponse(neighbors)
        predictions.append(result)
    accuracy = getAccuracy(testSet, predictions)
    return accuracy


def crossover(one, two):
    parent = [one, two]
    zero = '0'
    male = "{0:b}".format(parent[0])
    female = "{0:b}".format(parent[1])
    length = max(len(male), len(female))
    if length % 2 == 1:
        length = length + 1
    while len(male) < length:
        male = zero + male

    while len(female) < length:
        female = zero + female
```

```
    child = []
    half = int(length / 2)
    male1 = male[:half]
    male2 = male[half:]

    female1 = female[:half]
    female2 = female[half:]

    child.append(desimal(male1 + female2))
    child.append(desimal(female1 + male2))
    return child



# prepare data
trainingSet = x_train
testSet = x_test
split = 0.50

print('Train set: ' + repr(len(trainingSet)))
print('Test set: ' + repr(len(testSet)))

accResult = [[]]
chromosome = 10
population = generateChromosome(chromosome)

for x in range(len(population)):
 accResult.append([population[x], kNN(population[x], testSet,
trainingSet)])

del accResult[0]

for x in range(200):
       status_one = True
       status_zero = True
       accResult = sorted(accResult, key=lambda l: l[1], reverse=True)
       newChromosome = crossover(accResult[0][0], accResult[1][0])
       for i in accResult:
           if newChromosome[0] == i[0]:
               status_zero = False;
```

```
            if newChromosome[1] == i[0]:
                status_one = False
        if status_zero:
            accResult.append([newChromosome[0], kNN(newChromosome[0],
testSet, trainingSet)])
        if status_one:
            accResult.append([newChromosome[1], kNN(newChromosome[1],
testSet, trainingSet)])

accResult = sorted(accResult, key=lambda l: l[1],reverse=True)
print("accuracy of Genetic Algorithm:")
print(accResult[0][1])
```

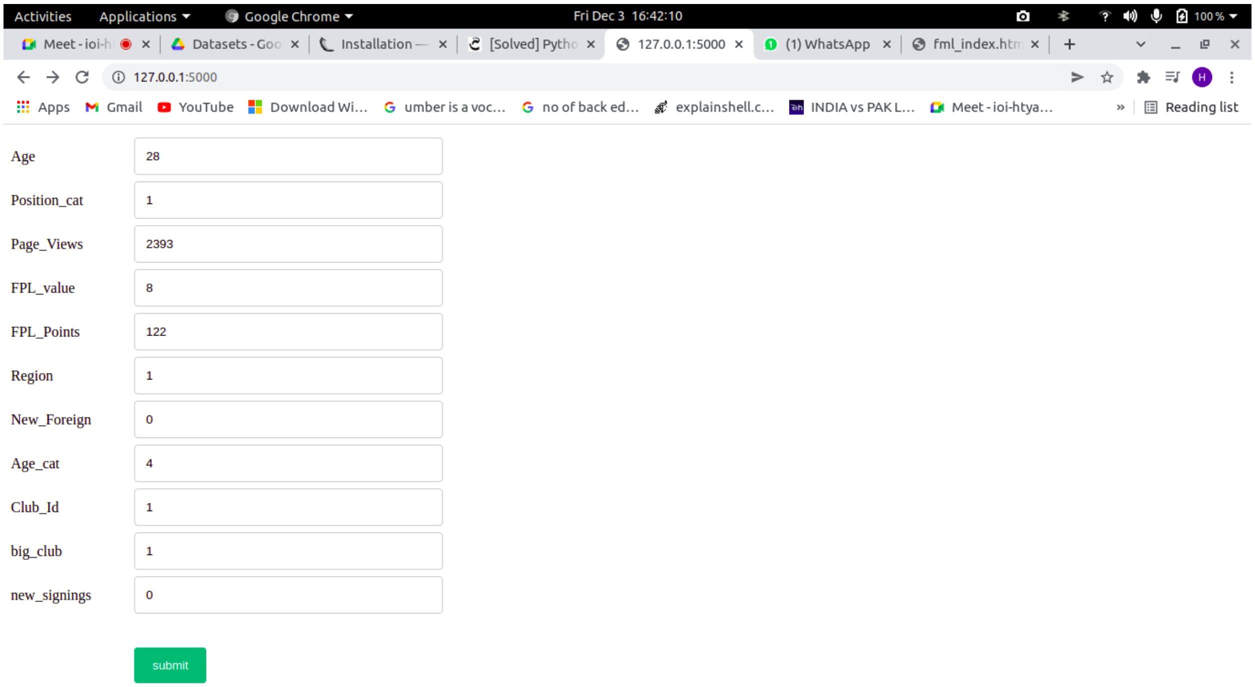
Result of Genetic Algorithm:

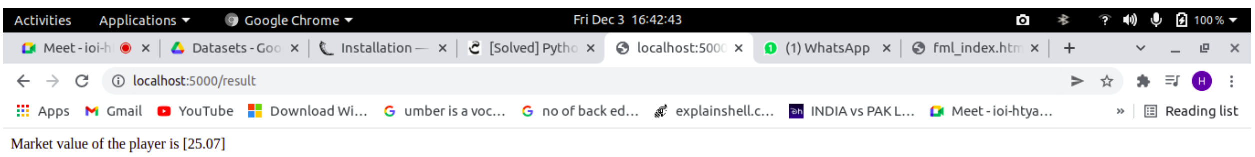
```
Train set: 322
Test set: 139
accuracy of Genetic Algorithm:
99.28057553956835
```

Part 6: Deploy your model as a RESTful Web Service

The best model that we found was RandomForestRegressor with the following parameters and R2 score.

```
Best Score:  0.7393239407974654
Best Params:  {'max_depth': 50, 'max_features': 'sqrt', 'n_estimators': 100}
Cross validation on model with best params
Mean: 0.7203383567504688, Std: 0.08593885301881521, Min: 0.578383343330517, Max: 0.8357472041181881
```


127.0.0.1:5000
Age 28
Position_cat 1
Page_Views 2393
FPL_value 8
FPL_Points 122
Region 1
New_Foreign 0
Age_cat 4
Club_Id 1
big_club 1
new_signings 0
submit


localhost:5000/result
Market value of the player is [25.07]