# Data Mining

Name: Piyush Zode

**Guidelines to run the file:**

1. The NBAstats.csv is stored in the same folder as the python file(p2.py)
2. To run: python P2.py
3. Classifier used: LinearSVC

**Method/Steps performed to improve accuracy:**

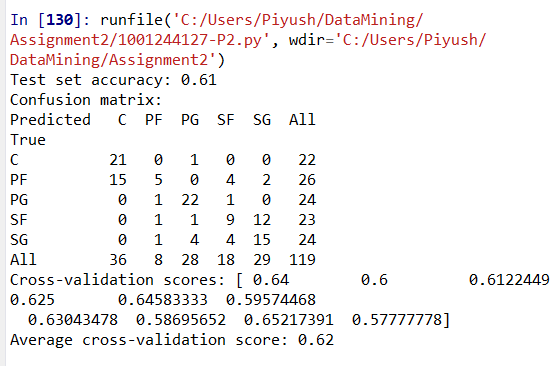
1. Initially I wrote the code for all the different classifiers (Decision Tree, SVM, KNN, Naïve Bayes) and checked the initial accuracy of these models on the NBA data set by using all the feature columns. Only SVM and Decision tree gave me a better initial accuracy so I selected these 2 and worked on these. I tried AdaBoostClassifier and ExtraTreesClassfiers. I also tried MinMaxScalar but it didn’t have any impact on the accuracy so I removed it.
2. I normalized the data using l1 normalization initially which increased the accuracy of both the models by approximately 2%.
3. Then I used l2 normalization which increased the accuracy drastically.

**A] Decision Tree:**

1. In Decision Tree, normalizing the data by l1 normalization increased the accuracy from 48% to 52%.
2. Then after using L2 normalization, the range of accuracy increased from 52-60%.
3. Then, I tried to do feature selection by removing the derived attributes by looking at the data in the csv file and the accuracy increased a bit.
4. Then I tried using the RandomForestClassifier which is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting.
5. By using the RandomForestClassifier, the range of cross-validation accuracy became 50-66%. But it was very random and the value of accuracy fluctuated a lot.
6. I tried to tweak different parameters of RandomForestClassifier and tried to prune the tree to different levels to increase the accuracy. But, it didn’t have any drastic effect on the accuracy, so I moved on to SVM.

**B] SVM:**

1. In SVM, normalizing the data by l1 normalization increased the accuracy from 50% to 53%.
2. Then after L2 normalization, the cross validation accuracy became stable and increased to 55%.
3. Then I tried do feature selection. I tried the feature selection algorithm(SelectKBest) and it didn’t work out pretty well. In fact the accuracy decreased by 2%. SO I dropped this idea.
4. Then for feature selection, I wrote a brute force code in java which gives all the possible combinations of the columns starting from 3 columns to 24 columns (the code for the same is at the below of this document). This code was just used for my analysis to get the best columns.
5. Then, I used these columns to check the cross validation accuracy and took the combination which gave the max value.
6. After selecting the columns ['MP','FG','FGA','FG%','3P','3PA','3P%','2P','2PA','2P%','FT','FTA','FT%','ORB','DRB','TRB','AST','BLK','TOV','PF','PS/G'] my accuracy increased to 60%.
7. Then, I tried to remove outliers on the training data but it decreased my accuracy by 5% and the final accuracy was 55%. So, I discarded the idea of removing outliers.
8. Then, I tried to tweak the parameters of lineaerSVC such as fit\_intercept and tried to set it to False which increased the accuracy to 61%.
9. Then by changing to fir\_intercept to true, it improved to accuracy to 62%. By setting it to True, an intercept was used in the calculations and it increased the accuracy.
10. For feature selection, the columns such as Age, G, GS, eFGT%, STL didn’t make an impact in increasing accuracy, so I removed them.
11. I was able to achieve 61% accuracy even with 4 columns ['3PA', '3P%', 'TRB','AST'] or ['3P', 'ORB', 'DRB','AST'] initially but couldn’t do much on that front after trying a lot of combinations. Also the combination of ['3P%','2P','2PA','2P%','eFG%','FT','FTA','FT%','ORB','DRB','AST','STL','BLK','TOV','PF'] gave an accuracy of 61%.
12. So, feature selection, L2 normalization and tweaks in the LinearSVC finally gave a cross validation accuracy of 62%.



**Below is the brute force code in Java for selecting the right features for analysis:**

**import** java.util.Arrays;

**public** **class** bruteForce {

**public** **static** **void** main(String[] args){

String[] columns = {"G","GS","MP","FG","FGA","FG%","3P","3PA","3P%","2P","2PA","2P%","eFG%","FT","FTA","FT%","ORB","DRB","TRB","AST","STL","BLK","TOV","PF","PS/G"};

*find\_combinations*(columns, 23, 0, **new** String[23]);

}

**static** **void** find\_combinations(String[] columns, **int** len\_of\_columns, **int** start, String[] res){

**if** (len\_of\_columns == 0){

System.***out***.println(Arrays.*toString*(res));

**return**;

}

**for** (**int** i = start; i <= columns.length-len\_of\_columns; i++){

res[res.length - len\_of\_columns] = columns[i];

*find\_combinations*(columns, len\_of\_columns-1, i+1, res);

}

}

}

**Below is the code for RandomForestClassifier which I mentioned above:**

# -\*- coding: utf-8 -\*-

"""

Created on Sun Nov 20 18:02:11 2016

@author: Piyush

"""

################################

#### Author: Piyush Zode ####

#### ID: 1001244127 ####

################################

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_val\_score

from sklearn.ensemble import RandomForestClassifier

from sklearn.datasets import load\_iris

from sklearn import preprocessing

from sklearn.preprocessing import normalize

#read from the csv file and return a Pandas DataFrame.

nba = pd.read\_csv('NBAstats.csv')

# print the column names

original\_headers = list(nba.columns.values)

# "Position (pos)" is the class attribute we are predicting.

class\_column = 'Pos'

#The dataset contains attributes such as player name and team name.

#We know that they are not useful for classification and thus do not

#include them as features.

feature\_columns = ['G', 'GS', 'MP', 'FG', 'FGA', 'FG%', '3P', '3PA', \

'3P%', '2P', '2PA', '2P%', 'eFG%', 'FT', 'FTA', 'FT%', 'ORB', 'DRB', \

'TRB', 'AST', 'STL', 'BLK', 'TOV', 'PF', 'PS/G']

#Pandas DataFrame allows you to select columns.

#We use column selection to split the data into features and class.

temp\_nba\_feature = nba[feature\_columns]

nba\_features = normalize(temp\_nba\_feature, norm='l2') ## Normalizing the data by L2 Noramization

nba\_class = nba[class\_column]

# Divide the test and train data using the normalized nba\_features

train\_feature, test\_feature, train\_class, test\_class = \

train\_test\_split(nba\_features, nba\_class, stratify=nba\_class, \

train\_size=0.75, test\_size=0.25)

clf = RandomForestClassifier(n\_estimators=16, max\_depth=None, min\_samples\_split=2, min\_samples\_leaf=1, \

min\_weight\_fraction\_leaf=0.0, max\_features='auto', max\_leaf\_nodes=None, min\_impurity\_split=1e-07, \

oob\_score=False, n\_jobs=1, verbose=0, warm\_start=False, class\_weight=None)

clf.fit(train\_feature, train\_class)

print("Test set score RandomForestClassifier: {:.3f}".format(clf.score(test\_feature, test\_class)))

### Confusion Matrix:

prediction = clf.predict(test\_feature)

print("Confusion matrix:")

print(pd.crosstab(test\_class, prediction, rownames=['True'], colnames=['Predicted'], margins=True))

## For Cross Validation:

new\_clf = RandomForestClassifier(n\_estimators=16, max\_depth=None, min\_samples\_split=2, min\_samples\_leaf=1, \

min\_weight\_fraction\_leaf=0.0, max\_features='auto', max\_leaf\_nodes=None, min\_impurity\_split=1e-07, \

oob\_score=False, n\_jobs=1, verbose=0, warm\_start=False, class\_weight=None)

scores\_random = cross\_val\_score(new\_clf, X, nba\_class, cv=10)

print("Cross-validation scores: {}".format(scores\_random))

print("Average cross-validation score RandomForestClassifier: {:.2f}".format(scores\_random.mean()))