CSE 4334: Report

Programming Assignment 2

Inara Rupani
1001534052

I *Inara Rupani* did not give or receive any assistance on this project, and the report submitted is wholly my own.

Objective:

For the Programming Assignment 2, we are tasked to classify NBA players into 5 positions on the basketball court: SG (shooting guard), PG(point guard), SF (small forward), PF(power forward), and C (center). We are asked to make these classifications based on each player's per-game performance in the 2018-2019 season.

Given:

1. Dataset file: A dataset file that includes the data of NBA players in the 2018-2019 season is given in the assignment and can also be found at the link here**.**
2. A skeleton file: The code demonstrates how to load a CSV file, how to build a classification model (specifically "k nearest neighbor" classifier), and how to evaluate it.
    2.1. The output of the file uses a few of the analysis tools such as Test Set Accuracy, Cross-Validation, and Confusion Matrix to conclude the results as such if the model is overfitting, undercutting, or is the perfect model for the case scenario.

Apply:

Based on the instructions, we are tasked to understand the working of our sample skeleton file. Understand the metrics and how the analysis is performed on the data set. We also need to understand how these metrics help us understand the functionality of each metrics and how they help us define each model.

Initial Run:

When the skeleton file runs on the first run, a few of the analysis tools prints which will help us classify if the model we are evaluating is good for our data set. The results prints measures such as test set accuracy, precision, recall, F-measure, and some tools like cross-validation and confusion matrix reports. These measures help are important in dealing with imbalanced classes just like our example. Just using "pure" accuracy on a multiple class set isn't reliable.

What could affect the classification model analysis:

1. The number of nba_feature columns/ headers from the .csv file
    1.1. There could be some features that do not affect the classification of players (classification of class attributes)
        1.1.1. Could cause skewness or biases in the results
        1.1.2. Could take longer for the classification algorithm to run and present analysis.
2. The tuning for parameters for each algorithm
    2.1. Could use trial and error method to test which values for the parameters works best for our model
    2.2. Could use hyper parameter tuning to lock in the parameters we decided from trial and error are the best one we can use for the data set.

Overall Status: Completed

Tests

Each classification algorithm we tried has many different parameters to use. Understanding each parameter allowed us to take a decision on which parameters should be used, what values affect the overall analysis, and how we can produce different and better results. Below when we discuss each algorithm these measures will be defined and explained.

A. Test 1: Decision Tree

Data Pre-Processing

1. Algorithm: Decision Tree Classification
   1.1. Modified parameters: "DecisionTreeClassifier(max_depth=7, random_state=0)"
      1.1.1. The depth of the tree will help us produce results which will in return identify if our model is overfitting or under fitting to the unseen future cases.
2. Libraries used: sklearn.tree import DecisionTreeClassifier
3. Cross Validation computation parameter increased to cv=10
   3.1. Default value used.

Before Results with feature columns with "Games Played" and "Minutes Per Game"

1. The red line identifies the Average cross-validation over how many times the data was trained.
2. The black line shows the test set accuracy which is the estimation of Test set. Accuracy can be biased based on the distribution of data in model.
3. The yellow diagonal identifies the true values and in practice those values should be higher than the upper and lower triangle.
4. Green line is the measure which is balances out recall and precision and is used as the good comparison against accuracy.

```
Cross-validation scores: [0.4        0.66666667 0.53333333 0.5        0.63333333 0.7
 0.53333333 0.63333333 0.26666667 0.6        ]
Average cross-validation score: 0.55

Test set accuracy: 0.52

Test set predictions:
['C' 'PF' 'SF' 'SG' 'PF' 'PG' 'C' 'PF' 'C' 'C' 'SF' 'C' 'SF' 'SF' 'SF'
 'PG' 'SF' 'SF' 'PG' 'PF' 'C' 'C' 'PF' 'PG' 'PF' 'SF' 'PG' 'SG' 'SG' 'SF'
 'SF' 'PG' 'SF' 'SG' 'PG' 'PG' 'PF' 'PF' 'SG' 'SG' 'PG' 'C' 'SF' 'PG' 'PF'
 'C' 'C' 'SF' 'PF' 'SF' 'C' 'SF' 'SF' 'PF' 'SG' 'SG' 'SG' 'SF' 'C' 'PF'
 'C' 'PF' 'SG' 'PF' 'SG' 'PG' 'C' 'SF' 'SF' 'PF' 'PF' 'PF' 'PG' 'SF' 'PG']

Confusion matrix:
Predicted   C  PF  PG  SF  SG  All
True
C          10   4   0   0   0   14
PF          4   6   1   5   0   16
PG          0   0  11   2   2   15
SF          0   6   0   6   3   15
SG          0   1   1   7   6   15
All        14  17  13  20  11   75

Classification report:
              precision    recall  f1-score   support

           C       0.71      0.71      0.71        14
          PF       0.35      0.38      0.36        16
          PG       0.85      0.73      0.79        15
          SF       0.30      0.40      0.34        15
          SG       0.55      0.40      0.46        15

    accuracy                           0.52        75
   macro avg       0.55      0.52      0.53        75
weighted avg       0.55      0.52      0.53        75
```

After removing those features, we see a drastic increase in cross-validation average score and test set accuracy.

```
Cross-validation scores: [0.36666667 0.6        0.63333333 0.36666667 0.73333333 0.7
 0.53333333 0.6        0.33333333 0.6        ]
Average cross-validation score: 0.55

Test set accuracy: 0.59

Test set predictions:
['C' 'SF' 'C' 'PG' 'PF' 'C' 'PF' 'PF' 'SG' 'SF' 'SG' 'PG' 'PF' 'SG' 'SF'
 'C' 'SF' 'C' 'PG' 'SF' 'C' 'C' 'SG' 'SF' 'SG' 'C' 'C' 'PF' 'PF' 'SG' 'SF'
 'PF' 'PF' 'SG' 'PF' 'C' 'SF' 'SF' 'PG' 'PG' 'SG' 'PF' 'PF' 'PF' 'SF' 'C'
 'PF' 'SF' 'C' 'SG' 'PG' 'SG' 'PF' 'PG' 'SG' 'SG' 'PG' 'PG' 'PG' 'PF' 'SF'
 'SF' 'PF' 'SG' 'SG' 'PF' 'PF' 'PG' 'SG' 'PF' 'C' 'SF' 'SF' 'PF' 'SG']

Confusion matrix:
Predicted   C  PF  PG  SF  SG  All
True
C          11   2   0   1   0   14
PF          1  10   1   4   0   16
PG          1   0  10   2   2   15
SF          0   6   0   4   5   15
SG          0   2   0   4   9   15
All        13  20  11  15  16   75

Classification report:
              precision    recall  f1-score   support

           C       0.85      0.79      0.81        14
          PF       0.50      0.62      0.56        16
          PG       0.91      0.67      0.77        15
          SF       0.27      0.27      0.27        15
          SG       0.56      0.60      0.58        15

    accuracy                           0.59        75
   macro avg       0.62      0.59      0.60        75
weighted avg       0.61      0.59      0.59        75
```

Model Evaluation:
libraries used: import pandas as pd, sklearn.model_selection import cross_val_score, sklearn.metrics import classification_report

As we compare the two data results we can see the average cross validation scores remains the same. The Test set accuracy is increased for after results which may suspect if there is a bias present in the data. But the f1-measure shows that After results is indeed better as it have a higher f-1 measures across the board for each class attribute. This made us to conclude that removing the two features help us with our classification. Our next tests will be build over this conclusion.

B. Test 2: Linear Support Vector Machine

Data Pre-Processing

1. Building model on Decision Tree classification conclusion
   1.1. cv = 10
   1.2. removed featured columns (Games Played, Minutes per Game)
2. Algorithm: Linear Vector Machine
3. Libraries used: sklearn.svm import LinearSVC
   3.1. Modified parameters: "LinearSVC(dual=False,tol=1e-4,random_state=0,max_iter=2000)"

```
Cross-validation scores: [0.5        0.63333333 0.53333333 0.53333333 0.7        0.73333333
 0.76666667 0.6        0.6        0.46666667]
Average cross-validation score: 0.61

Test set accuracy: 0.69

Test set predictions:
['SG' 'PG' 'PF' 'PF' 'C' 'SG' 'PG' 'SG' 'C' 'SF' 'SG' 'PG' 'PG' 'PG' 'C'
 'SF' 'SG' 'SG' 'C' 'PF' 'SG' 'C' 'PF' 'PF' 'SF' 'PF' 'PG' 'SG' 'C' 'PG'
 'C' 'C' 'SG' 'SF' 'SG' 'SF' 'PG' 'SF' 'SF' 'PF' 'PF' 'C' 'SF' 'PF' 'PG'
 'PG' 'C' 'PG' 'SF' 'C' 'SG' 'C' 'C' 'PF' 'SF' 'SG' 'C' 'SG' 'PG' 'C' 'SG'
 'PG' 'C' 'PF' 'PG' 'C' 'PG' 'C' 'SG' 'SG' 'PF' 'SF' 'C' 'PG' 'PF']

Confusion matrix:
Predicted   C  PF  PG  SF  SG  All
True
C          14   0   0   0   0   14
PF          4   9   0   3   0   16
PG          0   0  13   1   1   15
SF          1   3   0   6   5   15
SG          0   1   3   1  10   15
All        19  13  16  11  16   75

Classification report:
              precision    recall  f1-score   support

           C       0.74      1.00      0.85        14
          PF       0.69      0.56      0.62        16
          PG       0.81      0.87      0.84        15
          SF       0.55      0.40      0.46        15
          SG       0.62      0.67      0.65        15

    accuracy                           0.69        75
   macro avg       0.68      0.70      0.68        75
weighted avg       0.68      0.69      0.68        75
```

3.2. Modified parameters: "LinearSVC(dual=False,tol=1e-4,random_state=0,max_iter=5000)"

```
Cross-validation scores: [0.5        0.63333333 0.53333333 0.53333333 0.7        0.73333333
 0.76666667 0.6        0.6        0.46666667]
Average cross-validation score: 0.61

Test set accuracy: 0.67

Test set predictions:
['PF' 'PG' 'PF' 'C' 'C' 'SG' 'SG' 'C' 'SF' 'C' 'C' 'PG' 'PF' 'PF' 'PF'
 'PF' 'PF' 'SF' 'SG' 'PG' 'SF' 'PG' 'PF' 'PF' 'C' 'PF' 'C' 'SG' 'SG' 'PG'
 'C' 'PG' 'SG' 'C' 'SG' 'PF' 'SG' 'SF' 'PG' 'PG' 'PG' 'C' 'C' 'PF' 'SG'
 'PG' 'C' 'PG' 'PG' 'PG' 'SF' 'C' 'PG' 'SF' 'PF' 'PG' 'SG' 'C' 'C' 'C'
 'SG' 'SF' 'PG' 'PG' 'SG' 'C' 'C' 'PG' 'SF' 'C' 'SG' 'PF' 'PG' 'PG' 'SG']

Confusion matrix:
Predicted   C  PF  PG  SF  SG  All
True
C          13   1   0   0   0   14
PF          6  10   0   0   0   16
PG          0   0  14   0   1   15
SF          0   3   2   5   5   15
SG          0   0   4   3   8   15
All        19  14  20   8  14   75

Classification report:
              precision    recall  f1-score   support

           C       0.68      0.93      0.79        14
          PF       0.71      0.62      0.67        16
          PG       0.70      0.93      0.80        15
          SF       0.62      0.33      0.43        15
          SG       0.57      0.53      0.55        15

    accuracy                           0.67        75
   macro avg       0.66      0.67      0.65        75
weighted avg       0.66      0.67      0.65        75
```

3.3. Modified parameters: "LinearSVC(dual=False,tol=1e-4,random_state=0,max_iter=7000)"

```
Cross-validation scores: [0.5        0.63333333 0.53333333 0.53333333 0.7        0.73333333
 0.76666667 0.6        0.6        0.46666667]
Average cross-validation score: 0.61

Test set accuracy: 0.65

Test set predictions:
['PF' 'SF' 'C' 'PF' 'PG' 'SF' 'PG' 'PG' 'SG' 'SG' 'SG' 'SG' 'C' 'PG' 'PF'
 'SF' 'PF' 'PG' 'SF' 'C' 'C' 'PG' 'SF' 'PG' 'C' 'PG' 'SG' 'C' 'C' 'C' 'SF'
 'C' 'SF' 'PG' 'SG' 'SF' 'SF' 'C' 'SF' 'PF' 'C' 'PF' 'C' 'C' 'SG' 'SG'
 'PG' 'PG' 'SF' 'PF' 'PG' 'SF' 'PG' 'C' 'PG' 'SF' 'PF' 'PG' 'SG' 'PF' 'PG'
 'C' 'C' 'PG' 'SG' 'PF' 'SG' 'SF' 'C' 'C' 'SF' 'SG' 'SG' 'SG' 'PG']

Confusion matrix:
Predicted   C  PF  PG  SF  SG  All
True
C          14   0   0   0   0   14
PF          4   8   0   3   1   16
PG          0   0  13   0   2   15
SF          0   2   2   7   4   15
SG          0   0   3   5   7   15
All        18  10  18  15  14   75

Classification report:
              precision    recall  f1-score   support

           C       0.78      1.00      0.88        14
          PF       0.80      0.50      0.62        16
          PG       0.72      0.87      0.79        15
          SF       0.47      0.47      0.47        15
          SG       0.50      0.47      0.48        15

    accuracy                           0.65        75
   macro avg       0.65      0.66      0.65        75
```

Model Evaluation:
libraries used: import pandas as pd, sklearn.model_selection import cross_val_score, sklearn.metrics import classification_report

We performed three test using Linear support vector machine classification. we test our model on random_state=0, as it controls the pseudo random generation for shuffling the data. We use dual=False, because the number of samples in our data is greater than number of features as mentioned in scikit-learn website. The one attribute we change is max_iter=2000/5000/7000 this is the number of times the data will be trained. The more training we perform it will give us more precise and clean details thus increasing the test set accuracy and making the model not overfitted. We ran three runs to find the average of each test set for each increased iteration, the results of these averages are 66.6% , 66.3% and 67.3% respectively.

Therefore, out of three test we perform we conclude model max_iterations of 7000 is the better one. Even it might take more time than with 2000 it lowers the test accuracy and gives the balanced results in f1-measure.

C. Test 3: Naive Bayes Classifiers

Data Pre-Processing

1. Building model on Decision Tree classification conclusion
   1.1. cv = 10
   1.2. removed featured columns (Games Played, Minutes per Game)
2. Algorithm: Naive Bayes Classifier
3. Libraries used: sklearn.naive_bayes import GaussianNB
4. Model Used: GaussianNB()
   4.1. No modification to parameters

```
Cross-validation scores: [0.36666667 0.73333333 0.6        0.6        0.8        0.6
 0.5        0.53333333 0.36666667 0.46666667]
Average cross-validation score: 0.56

Test set accuracy: 0.45

Test set predictions:
['C' 'SG' 'PG' 'PG' 'C' 'SG' 'SG' 'C' 'C' 'C' 'PF' 'SG' 'PF' 'C' 'SF' 'PF'
 'C' 'SG' 'SF' 'C' 'SG' 'SG' 'PG' 'SG' 'SF' 'PG' 'SG' 'SG' 'PG' 'PG' 'C'
 'PF' 'PG' 'SF' 'PG' 'PF' 'PG' 'SF' 'SG' 'C' 'SF' 'SG' 'C' 'SF' 'SG' 'C'
 'PG' 'PF' 'SF' 'C' 'PG' 'SF' 'SG' 'PG' 'C' 'C' 'SG' 'SG' 'SG' 'SG' 'PG'
 'SF' 'PG' 'SG' 'C' 'SF' 'SF' 'SG' 'SF' 'SG' 'SG' 'SG' 'SG' 'SG' 'SF']

Confusion matrix:
Predicted   C  PF  PG  SF  SG  All
True
C          13   1   0   0   0   14
PF          3   4   0   7   2   16
PG          0   0   7   1   7   15
SF          0   1   2   3   9   15
SG          0   0   5   3   7   15
All        16   6  14  14  25   75

Classification report:
              precision    recall  f1-score   support

           C       0.81      0.93      0.87        14
          PF       0.67      0.25      0.36        16
          PG       0.50      0.47      0.48        15
          SF       0.21      0.20      0.21        15
          SG       0.28      0.47      0.35        15

    accuracy                           0.45        75
   macro avg       0.49      0.46      0.45        75
weighted avg       0.49      0.45      0.45        75
```

Model Evaluation:
ibraries used: import pandas as pd, sklearn.model_selection import cross_val_score, sklearn.metrics import classification_report

We analyzed the parameters for our classification model, there we couldn't find any important parameter could be adjusted which would change the time or the outcome of the analysis on our data set. Our tests for naive babes calculation are worse of the previous two classification model we applied. The test case accuracy of average three runs is 50%. which is lowest of Decision tree and linear support vector machine. Therefore we reject to use this model.

D. Test 4: Logistic Regression

Data Pre-Processing

1. Building model on Decision Tree classification conclusion
   1.1. cv = 10
   1.2. removed featured columns (Games Played, Minutes per Game)
2. Algorithm: Logistic Regression
3. Libraries used: sklearn.linear_model import LogisticRegression
   3.1. Modified parameters: LogisticRegression(penalty='l2',tol=1e-4,random_state=0,max_iter=2000)

```
Cross-validation scores: [0.5        0.8        0.7        0.56666667 0.73333333 0.7
 0.76666667 0.7        0.53333333 0.53333333]
Average cross-validation score: 0.65

Test set accuracy: 0.68

Test set predictions:
['C' 'C' 'C' 'PG' 'PG' 'PG' 'C' 'SG' 'SF' 'SF' 'PG' 'C' 'PF' 'PG' 'C' 'C'
 'SF' 'PG' 'PF' 'SF' 'PF' 'PF' 'PG' 'PF' 'SF' 'SF' 'C' 'C' 'SF' 'SG' 'C'
 'PG' 'PG' 'SG' 'PG' 'SF' 'PG' 'SF' 'SF' 'SF' 'SG' 'C' 'SG' 'SG' 'SF' 'PG'
 'SG' 'SG' 'PG' 'PF' 'PF' 'PG' 'PG' 'C' 'PF' 'SG' 'PG' 'C' 'SF' 'PF' 'PF'
 'PF' 'PF' 'PG' 'PF' 'SG' 'SG' 'SG' 'C' 'C' 'PG' 'SF' 'PG' 'PF' 'PF']

Confusion matrix:
Predicted   C  PF  PG  SF  SG  All
True
C          12   2   0   0   0   14
PF          3   9   0   3   1   16
PG          0   0  15   0   0   15
SF          0   3   2   7   3   15
SG          0   1   2   4   8   15
All        15  15  19  14  12   75

Classification report:
              precision    recall  f1-score   support

           C       0.80      0.86      0.83        14
          PF       0.60      0.56      0.58        16
          PG       0.79      1.00      0.88        15
          SF       0.50      0.47      0.48        15
          SG       0.67      0.53      0.59        15

    accuracy                           0.68        75
   macro avg       0.67      0.68      0.67        75
weighted avg       0.67      0.68      0.67        75
```

3.2. Modified parameters: LogisticRegression(penalty='l2',tol=1e-4,random_state=0,max_iter=5000)

```
Cross-validation scores: [0.5        0.8        0.7        0.56666667 0.73333333 0.7
 0.76666667 0.7        0.53333333 0.53333333]
Average cross-validation score: 0.65

Test set accuracy: 0.63

Test set predictions:
['PF' 'PG' 'PG' 'SF' 'C' 'SF' 'C' 'PG' 'SG' 'C' 'PG' 'SF' 'SF' 'PG' 'PF'
 'PG' 'SF' 'C' 'SG' 'PF' 'SF' 'C' 'SG' 'SF' 'SF' 'SG' 'SG' 'SG' 'C' 'PF'
 'PF' 'SG' 'PG' 'C' 'C' 'SG' 'PG' 'PF' 'PF' 'C' 'C' 'PG' 'PF' 'SG' 'PG'
 'PG' 'PG' 'PG' 'SF' 'PF' 'SF' 'SF' 'SG' 'PF' 'PG' 'SF' 'PG' 'C' 'C' 'SF'
 'PG' 'PF' 'PG' 'PF' 'PF' 'PG' 'PG' 'SF' 'PF' 'SG' 'SF' 'PG' 'SF' 'C' 'PF']

Confusion matrix:
Predicted   C  PF  PG  SF  SG  All
True
C           9   5   0   0   0   14
PF          4   8   0   4   0   16
PG          0   0  13   1   1   15
SF          0   2   2   9   2   15
SG          0   0   5   2   8   15
All        13  15  20  16  11   75

Classification report:
              precision    recall  f1-score   support

           C       0.69      0.64      0.67        14
          PF       0.53      0.50      0.52        16
          PG       0.65      0.87      0.74        15
          SF       0.56      0.60      0.58        15
          SG       0.73      0.53      0.62        15

    accuracy                           0.63        75
   macro avg       0.63      0.63      0.62        75
weighted avg       0.63      0.63      0.62        75
```

3.3. Modified parameters: LogisticRegression(penalty='l2',tol=1e-4,random_state=0,max_iter=7000)

```
Cross-validation scores: [0.5        0.8        0.7        0.56666667 0.73333333 0.7
 0.76666667 0.7        0.53333333 0.53333333]
Average cross-validation score: 0.65

Test set accuracy: 0.63

Test set predictions:
['SG' 'PG' 'SG' 'PF' 'SF' 'C' 'SG' 'C' 'SG' 'PF' 'PG' 'PG' 'PG' 'SG' 'SF'
 'SF' 'SF' 'PF' 'SG' 'PG' 'SF' 'SG' 'C' 'PF' 'C' 'SF' 'SG' 'C' 'PG' 'SG'
 'C' 'PG' 'PF' 'SF' 'PF' 'PF' 'PF' 'C' 'PF' 'PG' 'C' 'PG' 'PG' 'PF' 'PF'
 'PF' 'C' 'PG' 'SF' 'C' 'PF' 'PG' 'SG' 'PG' 'C' 'SF' 'C' 'PF' 'SG' 'C'
 'PF' 'C' 'SG' 'PF' 'PF' 'PG' 'PG' 'SF' 'PG' 'C' 'PG' 'PF' 'PF' 'SF' 'SF']

Confusion matrix:
Predicted   C  PF  PG  SF  SG  All
True
C          10   4   0   0   0   14
PF          5   9   0   2   0   16
PG          0   0  13   0   2   15
SF          0   6   2   6   1   15
SG          0   0   2   4   9   15
All        15  19  17  12  12   75

Classification report:
              precision    recall  f1-score   support

           C       0.67      0.71      0.69        14
          PF       0.47      0.56      0.51        16
          PG       0.76      0.87      0.81        15
          SF       0.50      0.40      0.44        15
          SG       0.75      0.60      0.67        15

    accuracy                           0.63        75
   macro avg       0.63      0.63      0.63        75
```

Model Evaluation:
ibraries used: import pandas as pd, sklearn.model_selection import cross_val_score, sklearn.metrics import classification_report

We performed three test using Logistic Regression classification model. As the parameters of this model were similar to Linear Support Vector Machine, we used those as our standards to unify the parameters values to check how the model responded. The main parameter for this model is "penalty" which either could be l1 or l2. On our testing l1 is failed based on how the code is structured to be used. and l2 is one we used.

```
/usr/local/lib/python3.6/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit failed.
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.

  FitFailedWarning)
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-64-dd49ab4748e7> in <module>()
     74 print("Average cross-validation score: {:.2f}".format(scores.mean()))
     75 print()
---> 76 model.fit(train_feature, train_class)
     77 print("Test set accuracy: {:.2f}".format(model.score(test_feature, test_class)))
     78 print()

                                  1 frames
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_logistic.py in _check_solver(solver, penalty, dual)
    443     if solver not in ['liblinear', 'saga'] and penalty not in ('l2', 'none'):
    444         raise ValueError("Solver %s supports only 'l2' or 'none' penalties, "
--> 445                          "got %s penalty." % (solver, penalty))
    446     if solver != 'liblinear' and dual:
    447         raise ValueError("Solver %s supports only "

ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalty.
```

We test our model on random_state=0, as it controls the pseudo random generation for shuffling the data. We use dual=False, because the number of samples in our data is greater than number of features as mentioned in scikit-learn website. The one attribute we change is max_iter=2000/5000/7000 this is the number of times the data will be trained. The more training we perform it will give us more precise and clean details thus increasing the test set accuracy and making the model not overfitted. We ran three runs to find the average of each test set for each increased iteration, the results stayed uniform and gave us average of 68% , 63% and 63% respectively. With the test set accuracy of 68% and classification measure report (Precision, recall, and f1-measure) we conclude to use model with max_iterations of 2000.

Based on further readings about the model, we conclude to reject to use this model. Logistic regression model is used for numerical class attribute values and not nominal. The one way we could have used logistical regression would be only by ranking these NBA position. Since the positions are not ranked in this way we cannot use this model.

Hyper-parameter Tuning

As we were given freedom to check and explore the library of scikit learn and apply hyper-parameter tuning to come up with the best model combination of parameters for our data set. I have done the same approach my mixing and using other combinations of parameters without the use of library. I have used trial and

error method in reading each data measurement printed for each run. And this report consists of all the different types of parameter I used and why I used.

Evaluation through Graphical Models

We were provided with sample code to use to build and analyze our models in a graphical representation as well as the measures printed out. I tried using the export_graphviz() function to build and visualize the decision tree model, but because of some Attribute Error function it kept failing.

Final Conclusion

Final comparison for choosing the best classification model for our test data set is between Decision Tree "DecisionTreeClassifier(max_depth=7, random_state=0)" and linear support vector machine "LinearSVC(dual=False,tol=1e-4,random_state=0,max_iter=7000)". Below we use a comparison chart

Measure	Decision Tree	Linear Support Vector Machine
Parameters used	max_depth=7, random_state=0	dual=False,tol=1e-4,random_state=0,max_iter=7000
Average cross validation score	0.55	0.61
Test Set Accuracy	0.59	0.65
F1-measure	0.81 @ C (Comparatively lower than LSVM) 0.56 @ PF	0.88 @ C (most values leads Decision Tree computation) 0.62 @ PF
Precision	0.85 @ C 0.50 @ PF	0.78 @ C 0.80 @ Pf

In addition, SVM may make use of inappropriate kernel (eg linear kernel for non-linear problem), poor choice of kernel and regularization of hyper parameters may cause inefficient results. In our case I wasn't able to perform code based automatic hyper parameter tuning, but as discussed I used and tried multiple combination of parameters such as dual, tol, max_iter value, libsolver. This tactic helped me select a good model out of all parameter combinations. SVM's do not take long time to train than decision tree.

Decision Tree is a recursive process which takes longer to predict the values. It utilizes the GINI index calculation to determine the split position. It runs slower but produces very detailed and precise results. The precision of Decision Tree is around higher than ones found at SVM classification, but they are not always guaranteed to be higher as seen in the table above.

Based on the above argument on which classification produces different results based on the test set and keeping our statistics from the table in mind, we conclude Linear Support vector machine is the better choice at classifying our NBA player positions. The test accuracy of this model suggest that the model is neither over fitting or under-fitting as we increased the complexity of Linear Vector machine by increasing its parameters we were able to get more detailed report but enough detailed to be able to generalized for future unseen cases. In addition of test accuracy as it be biased of some of the data in set, we can use f1-measure which balances out the precision (true values) and recall (almost true) values to tell us that Linear SVM has a lead in scores than Decision Tree. The diagonal row Linear SVM also conveys that the data is found to be correctly and more precisely identified than Decision tree.

References

- https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
- https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC
- https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html#sklearn.naive_bayes.GaussianNB
- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html?highlight=logistic%20regression#sklearn.linear_model.LogisticRegression
- https://www.dezyre.com/recipes/optimize-hyper-parameters-of-decisiontree-model-using-grid-search-in-python
- https://towardsdatascience.com/hyperparameter-optimization-with-scikit-learn-scikit-opt-and-keras-f13367f3e796