**CSE: 5334 Data Mining**

**Programming Assignment 2**

**Arjun Lakshmikanth**

**UTA ID: 1001192326**

----------------------------------------------------------------------

**Classifier:**

Multi-layer Perceptron Classifier (Average cross-validation score=0.66)

----------------------------------------------------------------------

**Changes to the Data set:**

After a thorough research on the player positions and the factors that affect each position, I came up with a reduced list of attributes which definitely affect the type of the position that the player is most fit for. For eg:

* Attributes like Player name, age and team are almost useless for classification.
* Further, attributes like Games, Games started, Minutes Played favor those players more who have a higher value for those attributes. Thus, these attributes are not a distinguishing factor for a player's position.
* Also, I removed derived attributes like Field Goal %, 3-Point Field Goal Percentage, etc which seemed to be unnecessary. Reducing the dimension of the data set helped achieve a faster and more accurate classification.

**Final features selected: ['FG', 'FGA', '3P', '3PA', '2P', '2PA', 'FT', 'FTA', 'ORB', 'DRB', 'AST', 'STL', 'BLK', 'TOV']**

----------------------------------------------------------------------

**Defining and tuning the classifier:**

The classifier I have chosen is MLPClassifier(activation='identity', alpha=2, solver='lbfgs', hidden\_layer\_sizes=200, tol=0)

Any parameters apart from the above mentioned have default values. I will now explain why I used the above parameters to improve the accuracy.

* **activation='identity'**: Among all the activation functions, identity function seemed to have the best runtime and accuracy. Using identity activation function is as good as an MLP being a single layer perceptron but with an added benefit of having hidden layers to correct the errors.
* **alpha=2**: Using L2 norm, alpha = 2 is a widely-used value for regularization term. This value gave the best accuracy for me.
* **solver='lbfgs'**: Assuming the dataset size used for training will be relatively small (less than a thousand data points), lbfgs is the most suitable for such a data set.
* **hidden\_layer\_sizes=200**: "There are some empirically-derived rules-of-thumb, of these, the most commonly relied on is 'the optimal size of the hidden layer is usually between the size of the input and size of the output layers'. Jeff Heaton, author of Introduction to Neural Networks in Java offers a few more.". Taking this into consideration 200 seemed to be a good choice for hidden layer size.
* **tol=0**: Setting the loss tolerance to 0 helps stop the learning early on and avoid further introduction of errors.