**DATA MINING - Programming Assignment 2**

**Chinmayee Makaraju (1002091569)**

1. (30 points) Use one classification method (Decision Tree/Naive Bayes/KNN/SVM) on the dataset. You can apply any of the methods explained in this instruction notebook or any other method in scikit-learn. You can even implement your own method. You can tune your model by using any combination of parameter values. Use 75% of the data for training and the rest for testing. Print out the training and test set accuracy of the model.

For me, the SVM (Support Vector Machines) model led to better accuracy because of its ability to capture complex non-linear relationships in the data and to handle both numerical and categorical features, but anyway, we removed categorical features like Player and Tm in the features. The Decision tree algorithm recursively partitions the data into subsets based on the most informative features, resulting in a tree-like structure where each internal node represents a decision based on a feature, and each leaf node represents the predicted outcome (class label or regression value).

A black and white screen with numbers

Description automatically generated

1. (20 points) Print out the confusion matrix for the model in 1). Note that we are dealing with a multi-class (5 basketball positions) classification problem. So, the confusion matrix should be 5 x 5. (Actually, 6 x 6 since we are also printing the numbers of "All". Refer to the earlier example.)

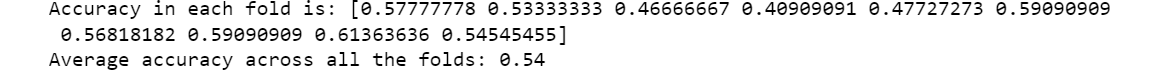
The confusion matrix would be like:

A number grid with numbers

Description automatically generated

1. (30 points) Use the same model with the same parameters you have chosen in 1). However, instead of using a 75%/25% train/test split, apply 10-fold stratified cross-validation. Print out the accuracy of each fold. Print out the average accuracy across all the folds.

Applying the 10-fold stratified cross-validation, the results would be as follows:



1. (20 points) Documentation: Explain your method that lead to better accuracy, what ideas or observations helped you achieve better accuracy on the dataset?

**Filtering by Playing Time:**

The code filters the dataset for each player position, excluding those with playing time (MP) in the bottom 10%. It calculates the 10th percentile of playing time for each position and removes players below that threshold.

**Combining Filtered Datasets:**

After filtering for each position, the code combines the resulting datasets into a single dataset, 'new\_df3', likely for further analysis or modeling.

**Feature Dropping:**

Certain columns considered redundant or unwanted for analysis are dropped. Columns such as "Player," "Tm" (Team), and various player statistics (e.g., field goals, rebounds) are removed.

**Filling Empty Values:**

The code attempts to fill the remaining missing values in the dataset with zeros.