Part 1: Introduction:

My project aims to complete a classification task using NBA player statistics to predict whether an NBA player is an MVP within a given year based on their performance across the year. The data set that I am using has already identified the top players in each year and I will add a binary column indicating whether someone was the MVP in a given year or not. My goal is to use these features to classify players as MVPs or not for each respective year. I will train and test multiple predictive models to assess which approach best predicts MVP performance, including using Logistic Regression as well as a Random Forest Regressor. I will also explore using a model that we have not learned in class yet: XGBoost (Gradient Boosting). I will separate the data set to be from 1982 to 2002 as a train period and then I will test from 2003 to 2022. This dataset is from Kaggle and was scraped from basketballreference.com. Attached is the link to where I retrieved the dataset: https://www.kaggle.com/code/robertsunderhaft/predicting-the-nba-mvp.

To summarize my findings, which will be displayed below, I found that all 3 classifiers work exceptionally well to determine whether a player is an MVP or not, with all 3 having a test score of >0.99. I found that the XGBoost Classifier and Linear Regressor typically marginally outperform the Random Forest Classifier when using my filtered data set (I will explain my filtering below as well). This is likely as a result of the simplicity of the task and the "ease" for these classifiers to take this relatively large amount of data to predict an MVP award that is typically based on these factors for voters as well. I believe the XGBoost and Linear Regressor outperform because they are inherently less likely to overfit, whereas the Random Forest is more prone to overfitting, which is why it performs relatively worse on this simple task.

Part 2: Data Description

This dataset is from Kaggle and was scraped from basketballreference.com. Attached is the link to where I retrieved the dataset: https://www.kaggle.com/code/robertsunderhaft/predicting-the-nba-mvp. The dataset consists of detailed player statistics such as points, rebounds, assists, shooting percentages (e.g. field goal percentage, three-point percentage, free throw percentage), and other relevant metrics for the top players in the NBA every year from 1982 to 2022. Additionally, the data set includes a column called “award_share” which represents the share of votes an individual

player received for the MVP award each year. The player with the highest award share in a year is the player that won the NBA MVP award. Using this information, I created an "is_mvp" column in the dataset, which tells me whether a player was MVP or not within a given year. This is the column that will be used as the target for my models.

In exploration of the data, I made charts for the number of MVPs that a player won, as well as created correlation matrices for the features in the datasets. As interesting as all of the data is, there were 50+ columns in the data set, some of which are repetitive measures of one's skill within a year (ie. multiple metrics about field goals and multiple metrics about 3 pointers.) As a result, I manually removed some of these more repetitive columns or columns that are not logical to be used. First, I made a correlation matrix with all the columns in the data set. Then, given the high correlation and near-perfect between some features as well as the logical issues with using some of the features (ie mov, mov_adj, and win_loss_pct), I am removing certain columns from the data set that are either illogical or already represented. Additionally, if there is both the made shots per game and shots attempted per game as well as the overall shot % for any given type of shot (ie. 3-pointer or free-throw), I removed the first two and used just the percentage feature for the given shot. I also used code to remove the player and team_id columns.

I then ran a linear regression to explore the absolute coefficients of each remaining feature in predicting NBA MVPs/non-MVPs. This revealed that the win shares per 48 minutes, true shooting percentage, and field goal percentages were the 3 most impactful features in the linear regression.

Part 3: Models and Methods

Having finished my exploration of the data, I ran tests using various classification models on the full dataset, which I divided into test and train sets. The code should allow for the creation of models that use the information from the test data to accurately classify every player in the dataset as either an MVP or not an MVP. The 3 models I'm using are a Logistic regressor, a RandomForest classifier, and an XGBoost classifier. A Logistic regressor is a linear model that predicts the probability of a binary outcome using the logistic (sigmoid) function and assumes the data is linearly separable. A Random Forest Classifier is an ensemble method that combines

predictions from multiple randomly built decision trees using majority voting. XGBoost is a boosting algorithm that sequentially builds decision trees to minimize a loss function using gradient descent and regularization.

To clarify my methods, I first filtered the data to only use numeric columns, and these columns are what remain after the previous filtering I did (to remove unnecessary columns). Additionally, I dropped the "award_share" column, as it reveals the voting share for each player within a year and has already been used to make a binary MVP column. Now, setting the "is_mvp" column as my target, I trained and tested all 3 of my models and plotted their confusion matrices. I also printed the test and train scores for the models.

Part 4: Results and Interpretation

1. Logistic Regression

```
Train Accuracy: 0.99848
Test Accuracy: 0.99842
Confusion Matrix:
[[8192    3]
 [  10   10]]
```

Interpretation: Logistic Regression achieved near perfect accuracy on both the test and training datasets, with only 13 errors in total (3 false positives and 10 false negatives). The test accuracy is very close to the training accuracy, which indicates that the model is not overfitting and is generalizing well to the test data.

Conclusion: This model does a great job of predicting the non-MVP class, but its performance on the MVP class is weaker (10 false negatives). The small number of MVP predictions suggests that the model is conservative in classifying players as MVPs, which may be expected since MVPs are relatively rare.

2. Random Forest

```
Train Accuracy: 1.00000
Test Accuracy: 0.99793
Confusion Matrix:
[[8195    0]
 [  17    3]]
```

Interpretation: Random Forest has a perfect train accuracy but its test accuracy is slightly lower at 99.793%. The model correctly classifies the vast majority of players (17 errors in total). The confusion matrix shows that while the model is strong at predicting non-MVPs (no false positives), it struggles slightly more with MVP classification, as indicated by 17 false negatives.

Conclusion: Random Forest is able to classify non-MVPs with high accuracy, but it struggles to predict MVPs (17 false negatives). The model is likely under-predicting the MVP class compared to non-MVPs.

3. XGBoost

```
Train Accuracy: 1.00000
Test Accuracy: 0.99842
Confusion Matrix:
[[8192    3]
 [  10   10]]
```

Interpretation: XGBoost also achieves perfect train accuracy, and its test accuracy is very high at 99.84%. It performs similarly to Logistic Regression, with just 13 errors in total (3 false positives and 10 false negatives). The confusion matrix shows that XGBoost, like Logistic Regression, has some difficulty predicting MVPs, with 10 false negatives.

Conclusion: XGBoost is also a strong model that performs similarly to Logistic Regression. Like Logistic Regression, it tends to be conservative in predicting MVPs, leading to a few false negatives. However, overall, it classifies non-MVPs very well and achieves a high test accuracy.

Overall Insights: All three models (Logistic Regression, Random Forest, and XGBoost) perform well in terms of accuracy, with test accuracies between 99.79% and 99.84%. All the models have perfect or near-perfect train accuracy, suggesting potential overfitting to the training data. However, their test performances are still very good, indicating that the task is very simple for these models.All models tend to predict the non-MVP class with very high accuracy, but they struggle somewhat with the minority MVP class, resulting in some false negatives (players who should be predicted as MVPs but are not).

Extension: I tested a different type of model to see if a logical change can make a difference in the model’s accuracy. For example,

having watched the NBA, I know that the nature of the game and what was prioritized for MVP voting in 1982 is not the same as what is likely prioritized for MVP voting today. This is as a result of the change in the way basketball is played in the NBA today, where 3-point shooting has gone up significantly. In the first year the NBA introduced the three-point line, teams averaged 2.8 3-point attempts per game. By the 2018–2019 season, that number had increased to 32.0 3-point attempts per game, which is a >1,000% increase. Currently, we are training on pre 2002 data and testing with post 2002 data, which is likely exacerbating any differences in the eras of basketball and their implications on our model. As a result, I will instead train on data for even years and test on data for odd years, which should incorporate an equal number of pre-"small ball" era basketball and post-"small ball" era basketball.

The following were some insights from those results: The test accuracy across all three models has slightly decreased in the new results compared to the previous ones:

Logistic Regression: 0.99842 → 0.99764 (slight drop)
Random Forest: 0.99793 → 0.99736 (slight drop)
XGBoost: 0.99764 → 0.99764 (no change)

Despite these small drops in test accuracy, the models still maintain very high test accuracy, showing consistent performance. Against my expectations of the models improving because they are tested and trained across eras, it is likely that the models were less accurate in making predictions now because the model was not accurate in generalizing across the two styles, and may be overfitting to specific patterns seen in the even years. Additionally, the factors that inform the differences in play styles, like 3-pt percentage, are not as heavily weighted as factors that are general across eras like win-loss percentage, true shooting percentage, win shares per 48 minutes, and field goal percentage.

Interestingly, the feature importances for the two models paint a very different picture. The models tested and trained across eras have defensive win shares and value over replacement player as their most important feature, both of which are not the most important feature in any of the 3 models tested and trained on the 1982-2002/2002-2022 split, which means that the models did change to reflect the difference in the way I have split the test and train data.

Part 5: Conclusion and Next Steps

Summary of Findings:

Logistic Regression, Random Forest, and XGBoost all performed exceptionally well at the task of classifying NBA players as MVPs or non-MVPs, with test accuracies of 99.79%–99.84%. Logistic Regression and XGBoost had similar confusion matrices, with 10 false negatives and 3 false positives, while Random Forest had no false positives but 17 false negatives. All models excelled at predicting non-MVPs but struggled with MVP predictions, reflecting the class imbalance. Testing a new train-test split (even vs. odd years) slightly reduced test accuracy but revealed changes in feature importance, such as defensive win shares and value over replacement player becoming key predictors. This suggests the models adapt to era differences but may not become more accurate if they are not able to generalize well to them.

Next Steps/Improvements:

To enhance the predictive capabilities of the models, addressing class imbalance is important. The MVPs are a minority class, and the models tend to underpredict them, which results in false negatives. I can use Synthetic Minority Oversampling Technique to create synthetic MVP samples to balance the dataset or use class weights to penalize models more heavily for misclassifying MVPs (because correctly identifying MVPs is my main goal).

Additionally, refining feature selection can significantly improve the performance of my models. Using era-specific features, such as metrics that capture the rise of three-point shooting, could help players across different periods. Combining features like true shooting percentage with points per game may also capture nonlinear relationships more effectively. Incorporating advanced features like "clutch" performance (performance in key moments of key games) or consistency in key games could align more closely with MVP selection criteria.

Changing the train-test split to incorporate era factors is an improvement I could make differently. A rolling window cross-validation approach could train and test models incrementally across overlapping time periods, helping account for changes in MVP criteria and ensuring better generalization across basketball eras. Stratified sampling could also ensure consistent class distributions

in the training and testing sets, preventing imbalances that might skew model performance.

By addressing these key areas, the models will be better equipped to accurately classify MVPs while adapting to the evolution of the game.