proj3_report

1. The dataset that I chose, NFL_injuries_data, was from the website Kaggle and was one of a group of 15 datasets under 'NFL Player Statistics 2002 - Present'. The original 'NFL_injuries_data' dataset consists of around 80 thousand rows, with each row representing an NFL player who has been injured in some way. There were originally 16 columns as well consisting of season, game type, team, week, GSIS ID, full name, first name, last name, primary injury, etc.
   Link to original data set:
   https://www.kaggle.com/datasets/dubradave/nfl-player-statistics-2002-present/data?select=NFL_injuries_data.csv

2. The task I was aiming at accomplishing with the machine learning models was to predict whether a player would either participate in the following week's game, or not based on several features, both categorical and numerical. Some categorical variables that were used to predict a player's game status were position, team, game type, report injury, and practice status. Some numerical variables included the year the injury took place, along with the week in the season in which the injury took place. In this project, I was able to implement several machine learning models and techniques in order to predict whether a player would be active come game time or not based on injury data.

3. There were several steps that had to be taken in order to clean the original 'NFL_injuries_data' dataset. First, I began by renaming certain columns to make them more straightforward, while I also dropped specific columns that I know would not be necessary for the end goal of predicting whether a player would play or not. I then noticed that the numerical variable, year and week, were coded as floats, so I re-coded them as integers. I then made sure that the 'date_modified' column was in datetime just in case I wanted to add a deeper aspect of time of injury (which I ended up not needing to do). There was a great amount of variance when it came to the list of injuries as several of the same injuries were written in different formats so I decided to recode the injuries with more precise titles, followed by the non-injury titles, then multiple injury titles, then covid values, and finally practice status. I finished cleaning the data by dropping rows with NA values and rearranging the order of columns to my liking. I finished the file by splitting the dataframe into train, validation, and test sets and saved them to be used in the machine learning section of the project.

4. The first machine learning model implemented was logistic classification. It demonstrated a decently strong performance with consistent accuracy across all sets with a training accuracy of 92.20%, validation accuracy of 87.27%, and a test accuracy of 86.70%. The Radial Basis Function (RBF) kernel Support Vector Machine achieved high

accuracy as well, with results very close to that of the logistic regression model. It had a training accuracy of 92.14%, a validation accuracy of 87.31%, and a test accuracy of 87.11%. The best parameters were C=10 and gamma = 0.001. The Polynomial kernel Support Vector Machine showed a slightly lower test accuracy compared to the RBF kernel. It ended up having a training accuracy of 92.14%, validation accuracy of 87.11%, and a test accuracy of 86.07%. The best parameters were C=0.1 and degree = 2. The Decision Tree ended up performing with a training accuracy of 99.82%, validation accuracy of 79.76%, and a test accuracy of 79.22%. The best parameters were max_depth = 3, min_samples_leaf = 1, and min_samples_split = 2. The Random Forest performed well with a training accuracy of 92.30%, validation accuracy of 86.45%, and a test accuracy of 86.13%. The best parameters were max_depth = 10, min_samples_leaf = 1, min_samples_split = 2, and n_estimators = 100. Finally, the Neural Network (MLP) showed very high training accuracy with lower validation and test accuracies as it finished with an average training accuracy of 97.73%, average validation accuracy of 79.66%, and an average test accuracy of 75.55%.

5. Based on the resulting prediction accuracies, logistic regression and support vector machine with the RBF kernel demonstrated the most consistent performance throughout all sets, showing strong generalization with very minimal overfitting. On the other hand, the decision tree and neural network models showed significant overfitting as they had extremely high training accuracy with low validation and test accuracies. This suggests that the models did a good job of memorizing the training data, rather than learning generalizable patterns. The random forest model, along with the RBF kernel SVM and logistic regression models, showed strong performance across all sets, while the polynomial kernel SVM performed a tad bit worse than the RBF kernel. Overall, the models showed a range of convergence behavior and with tweaking to some of the hyperparameters, the models could possibly be made more accurate and thus avoid over- and underfitting.