MACHINE LEARNING (COMP7703)

Assignment - Semester 1, 2025.

Student Name: Volter Entoma

Student ID: 44782711

1. Introduction

There are many different machine learning algorithms, and many metrics to evaluate them. In this assignment, we will focus on the K-Nearest Neighbour (K-NN) algorithm, Random Forest, and Deep Neural Networks, and the accuracy of each model. We will also looks at the different metrics to evaluate the performance of each model, and use these metrics to compare the suitability of each model for the given data.

2. Aims

- To understand the K-Nearest Neighbour (K-NN) algorithm.
- To understand the accuracy of the K-NN classifier.
- To understand the effect of data imbalance on the accuracy of the K-NN classifier.
- To understand how to use cross-validation to improve the accuracy of the K-NN classifier.

3. Data

The data used in this assignment is the `TTSWING` dataset, which is the provided dataset for this assignment. The dataset contains the 33 features, which are the 33 different measuremeents of table tennis swings. The dataset also contains the height of the player, which is the target variable. The dataset is a CSV file, and can be read into Python using the `pandas` library. The dataset is split into three categories: high, medium, and low.

3.1 Data split

The data is split into three height categories: high, medium, and low. The data is split into 60% training data, 20% testing data, and 20% validation data. The training data is used to train the each model, the testing data is used to test the accuracy of each model, and the validation data is used to compare the performance of the models.

3.2 Data imbalance

After splitting the data, there is an imbalance in the amount of training data assigned to each category in height. There are 17364 (29.73%) data points in the high, 40800 (42.03%) data points in the medium, and 27450 (28.23%) in the low category. This is a significant imbalance; this imbalance is problematic because many machine learning algorithms tend to be biased toward the majority class. This can result in poor predictive performance for the minority classes, as the model may learn to ignore them in favor of

achieving higher overall accuracy. In imbalanced datasets, accuracy becomes a misleading metric, since a model can achieve high accuracy by simply predicting the majority class most of the time, while failing to correctly classify minority class instances. This is especially concerning when the minority classes are of particular interest or importance.

The data imbalanced is addressed by the use of Synthetic Minority Over-sampling Technique (SMOTE) to generate synthetic data points for the low category. The SMOTE algorithm generates synthetic data points by interpolating between existing data points in the low category. This is done by selecting a random data point from the low category, and then selecting a random data point from the k nearest neighbours of that data point. The synthetic data point is then generated by taking a weighted average of the two data points.

After applying SMOTE, the training data is balanced, with 24552 data points in each category. All subsequent analysis is performed on the balanced training data. The testing and validation data is not balanced, and is used to test the accuracy of the models.

3.3 Data standardization

The data is standardized using the StandardScaler from the `sklearn` library. The StandardScaler standardizes the data by removing the mean and scaling to unit variance. This is done to ensure that all features have the same scale, and to improve the performance of the models.

4. Principle Component Analysis

To investigate the effect of dimensionality reduction on the accuracy of the models, we will use Principle Component Analysis (PCA) to reduce the dimensionality of the data. PCA is a linear transformation that transforms the data into a new coordinate system, where the first coordinate is the direction of maximum variance, the second coordinate is the direction of maximum variance orthogonal to the first coordinate, and so on.

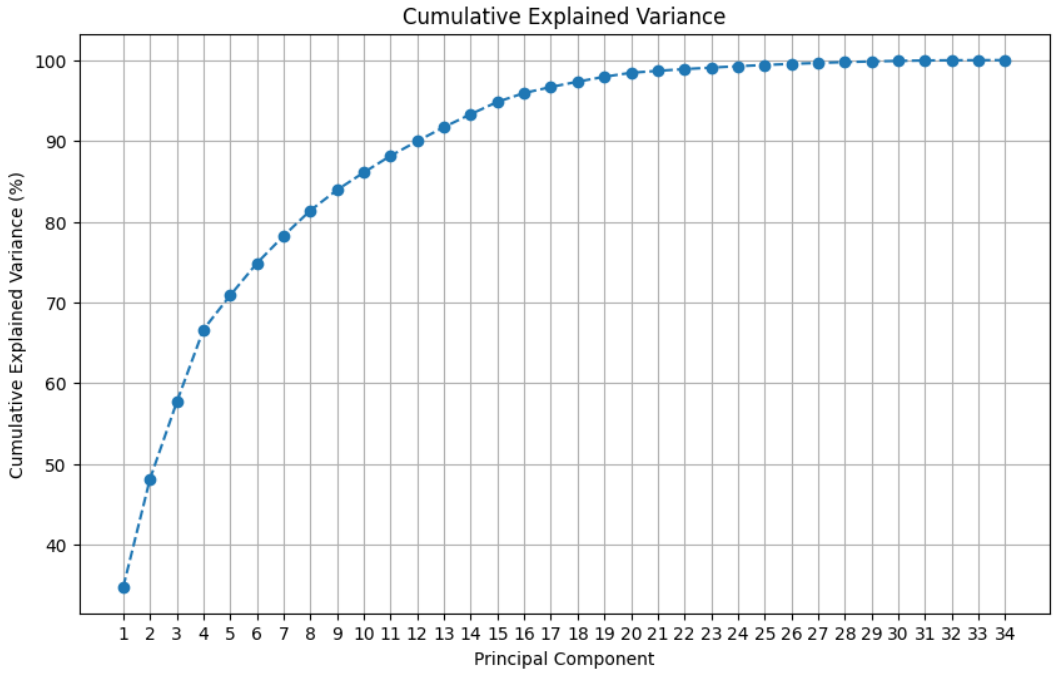

Figure 1: Cumulative variance explained by each principal component.

As shown in Figure 1, approximately 95% of the total variance is explained by the top 16 principal components. This allows us to reduce the dimensionality of our data from 33 dimensions to 16 dimensions while retaining the vast majority of the information content. This reduction offers several benefits:

- Reduced computational complexity in subsequent modeling
- Mitigation of the curse of dimensionality
- Removal of potentially noisy or redundant features
- Improved model interpretability

The effectiveness of this dimensionality reduction will be evaluated by comparing model performance on both the original and PCA-reduced datasets.

5. K-NN classifier

The K-Nearest Neighbour (K-NN) algorithm is a simple and effective machine learning algorithm that can be used for both classification and regression tasks. The K-NN algorithm works by finding the k nearest neighbours of a data point, and then predicting the class of the data point based on the classes of the k nearest neighbours. The K-NN algorithm is a non-parametric algorithm, which means that it does not make any assumptions about the distribution of the data. This makes the K-NN algorithm very flexible, and it can be used for a wide variety of tasks, including for our case of the classifying the height of a table tennis player based on their swing data.
The K-NN algorithm was applied to the data to classify the height of the player based on their swing data.

5.1 Accuracy of K-NN classifier

Using the data split outlined in Section 3.1, the K-NN classifier was trained on the training data, and then tested on the testing data.

The accuracy of the K-NN classifier is heavily dependent on the number of neighbours (k) used in the algorithm. Figure 2 shows the accuracy of the K-NN classifier reaches a maximum at 1 neighbour, and then decreases as the number of neighbours increases. This could be related to the high dimensionality of the data.
The K-NN classifier is particularily sensitive to the curse of dimensionality; as the number of dimensions increases, point within the hyperspace are more likely to be equidistant from each other. This means that the K-NN algorithm is less able to distinguish between points, and hence the accuracy of the K-NN classifier decreases. This is a common problem with high dimensional data, and is one of the reasons why dimensionality reduction techniques such as PCA are used.

We can confirm this by looking at the accuracy of the K-NN classifier on the PCA-reduced data.

The accuracy of the K-NN classifier on the PCA-reduced data is shown in Figure 3. The curve for reduced data to 2-dimensions shows the best accuracy occurs with 2^9 neighbour, which suggests that the points become more "spread-out" in 2-dimensions and the model is able to distinguish between different catgeories better as the number of neighbours increases.
However, despite the dimensionality reduction, the accuracy of the K-NN classifier is still highest at 1 neighbour, and then decreases as the number of neighbours increases. This suggests that the decrease in accuracy as neighbours increase is not solely due to the high dimensionality of the data, but also due to

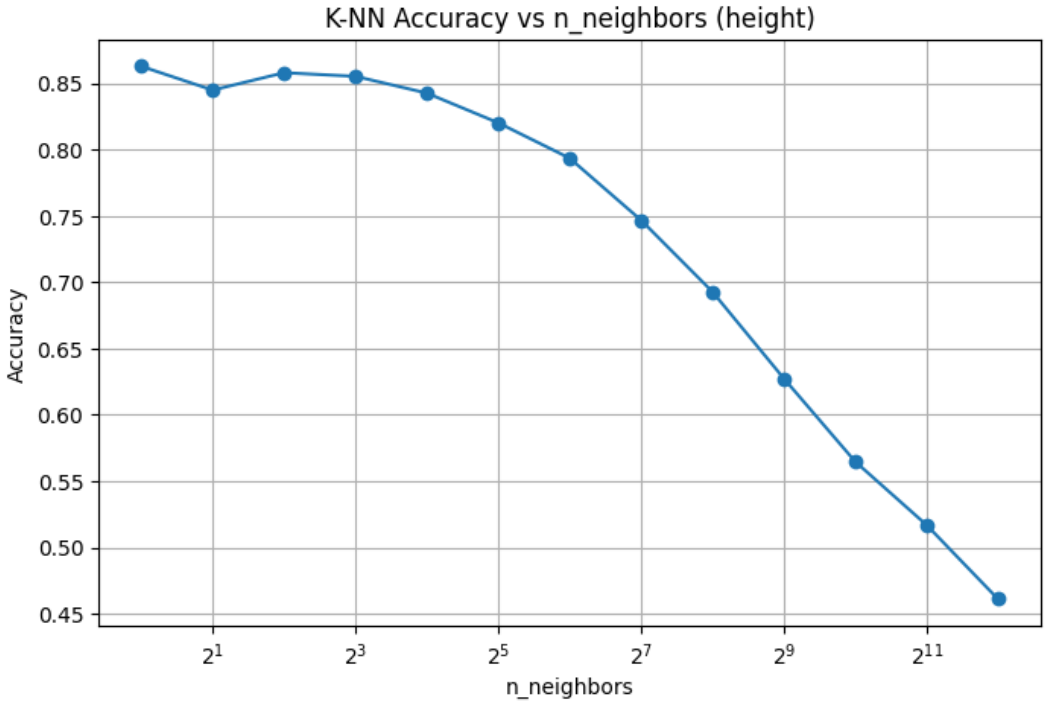

Figure 2: Accuracy of K-NN classifier on testing data.

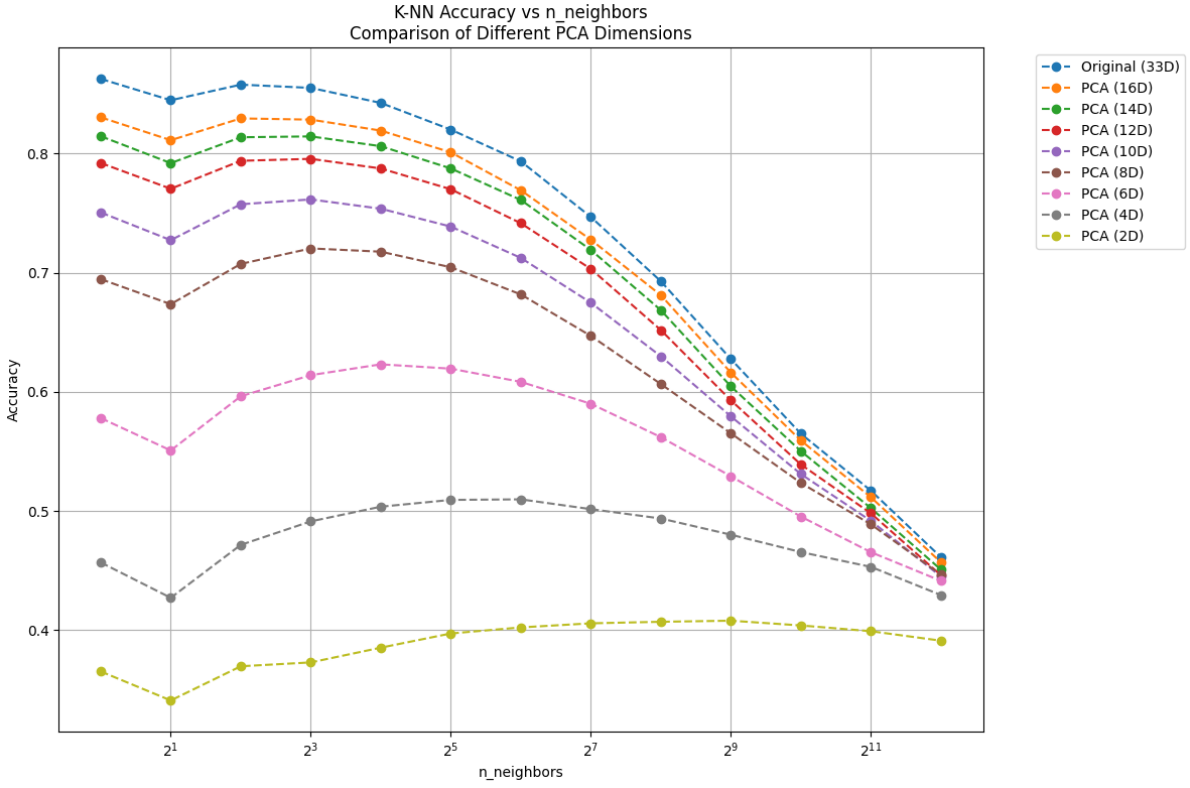

Figure 3: Accuracy of K-NN classifier on PCA-reduced data.

the nature of the K-NN algorithm on this particular dataset. This idea is supported by the fact that the

curve for the PCA-reduced data (to 16 dimensions) is similar to the curve for the original data. Additionally, the accuracy of the K-NN classifier decreases as the number of dimensions decreases. If the curse of dimensionality was a significant problem for this dataset, we would expect the accuracy to increase. This suggests that the curse of dimensionality is not a significant problem for this dataset, and that decreasing the number dimensions is not beneficial to the model, as it loses too much information.

5.1.1 Computational time after dimensionality reduction

The computational time of the K-NN classifier is also significantly reduced after dimensionality reduction. The computational time of the K-NN classifier is largest

5.2 K-Fold Cross-validation

The best accuracy of the K-NN classifier is achieved with one neighbour, and the accuracy is approximately 86.26%. However, one neighbour could be overfitting the data, therefore is important to explore different sets of training and testing data, to see if the accuracy is consistent across different sets of data. This is done by using K-Fold Cross-validation, which splits the data into K different sets, and then trains and tests the model on each set. The accuracy of the model is then averaged over all K sets.
We use 10-fold cross-validation, which means that the data is split into 10 different sets, and then the model is trained and tested on each set. Note that the choice of the number of folds was arbitrary, and was chosen to be 10 because it is a common choice in the literature.

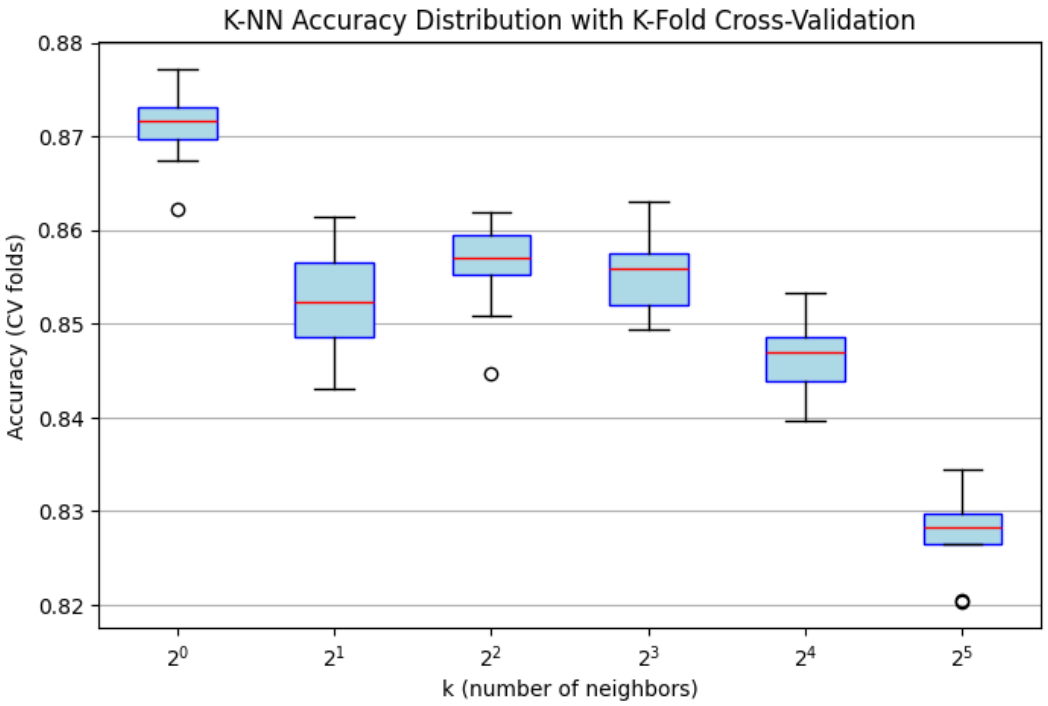

Figure 4: Accuracy of K-NN classifier with K-Fold Cross-validation.

The accuracy of the K-NN classifier with K-Fold cross-validation is shown in Figure 4. With 10-fold cross-validation, the highest average accuracy occurs with one neighbour. Notably, the minimum accuracy across the 10 folds for one neighbour is approximately equal to the maximum accuracy observed for two, four, and eight neighbours. This suggests that one neighbour is indeed the optimal choice for this dataset.

Additionally, the variance in accuracy across folds for one neighbour is comparable to the variance observed for other values of k. If one neighbour were overfitting the data, we would expect to see significantly higher variance across the folds for k=1 compared to other values of k. Since this is not the case, we can conclude that, for this dataset, the K-NN classifier with one neighbour does not appear to overfit, as evidenced by the consistently low variance.

5.3 ROC curve

We use the k=1 K-NN model with the highest accuracy in the 10-fold cross-validation to plot the ROC curve. The ROC curve is a graphical representation of the performance of a binary classifier as the discrimination threshold is varied. It plots the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The area under the ROC curve (AUC) is a single scalar value that summarizes the performance of the classifier across all thresholds.

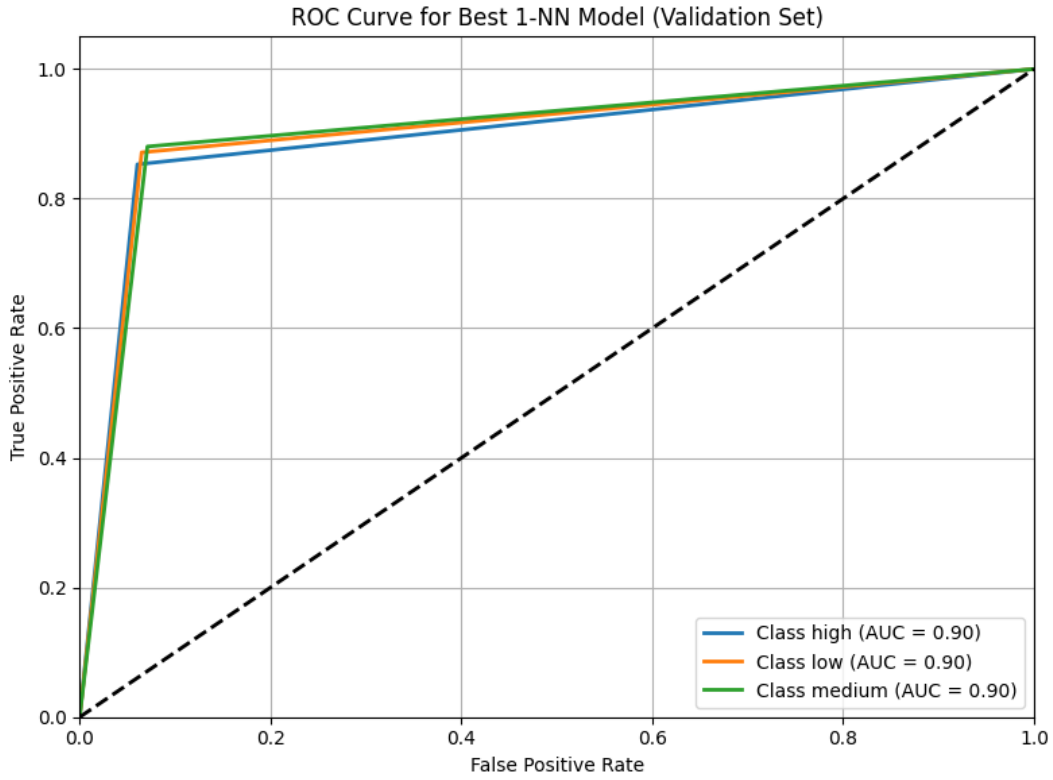

Figure 5: ROC curve for K-NN classifier with k=1.

The ROC curve for the K-NN classifier with k=1 is shown in Figure 5. The AUC for the K-NN classifier with k=1 is approximately 0.9 for all height categories. This indicates that the K-NN classifier with k=1 is a good classifier, as it has a high true positive rate and a low false positive rate.
The ROC curve itself is not very informative for one-neighbour K-NN, as it can only produce three points. The lack of points is due to the fact that the model outputs "hard" probabilityes (only 0 or 1) for each class, rather than "soft" probabilities, that would be possible for a higher number of neighbours.

5.4 Confusion matrix

We can also use the same K-NN model to plot the confusion matrix using the validation data that was not used in the training or testing of the model. The confusion matrix is a table that is often used to describe the performance of a classification model on a set of data for which the true values are known.

The confusion matrix shows the number of correct and incorrect predictions made by the model, broken down by class.

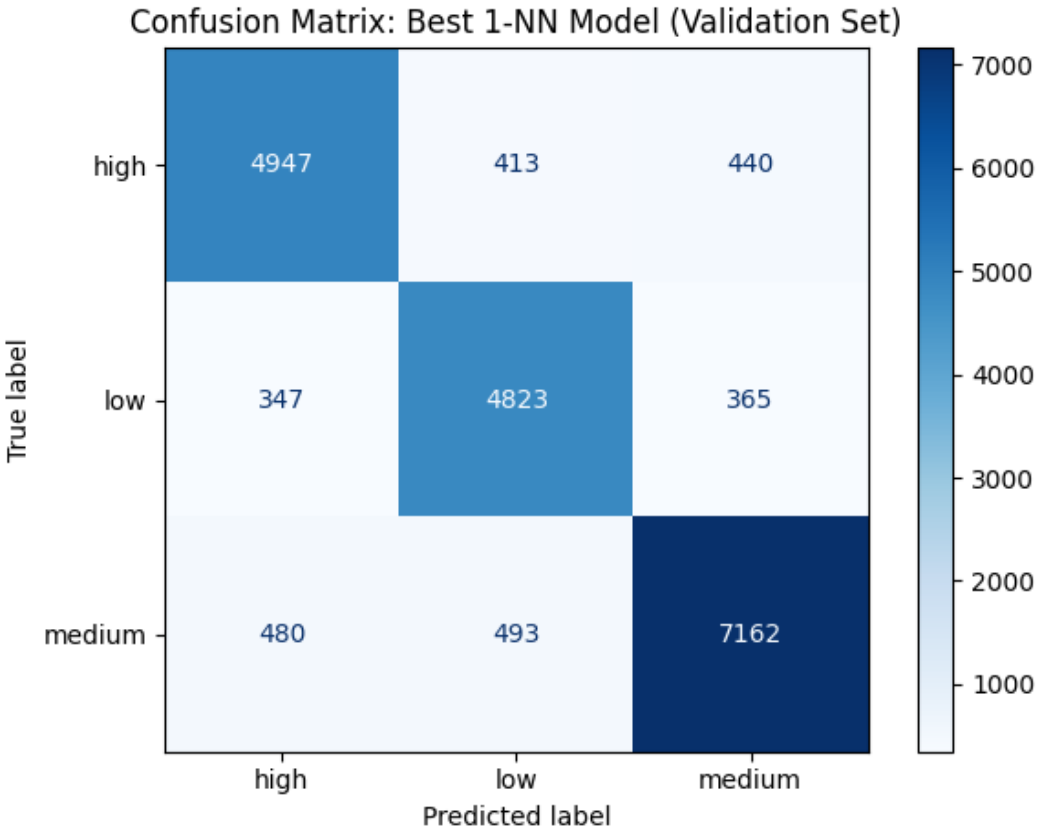

Figure 6: Confusion matrix for K-NN classifier with k=1.

The confusion matrix shows that the diagonal elements to be the largest, which indicates that the K-NN classifier with k=1 is able to correctly classify the majority of the data points. The off-diagonal elements are much smaller, which indicates that the K-NN classifier with k=1 is not making many incorrect predictions. This is a good sign, as it indicates that the K-NN classifier with k=1 is a good classifier for this dataset.

5.5 Precision, Recall, and F1 score

6. Random Forest

7. Deep Neural Networks