Github.com/MashClashXD, Aidan Roshan

Chen (Cherie) Ding

CPS 844

05 April 2025

Wine Quality Classification

The dataset chosen for this project is a wine quality dataset, containing various attributes of 1600 Portuguese red wine variants (Vinho Verde). These attributes include features such as pH levels, acidity, alcohol content, and other physicochemical properties. The target variable is a quality rating, which reflects the overall quality of the wine. The goal of this project is to build classifiers capable of predicting whether a wine is of high or low quality based on these attributes.

Preprocessing was conducted on the quality ratings, which originally ranged from 3 to 8. Due to the uneven distribution of instances across different quality ratings, the target variable was reclassified into a binary class, distinguishing between high and low-quality wines. This was done using equal width binning (3-5 and 6-8) and allowed for more balanced classification. Of the 1,599 wines in the dataset, 744 are now considered low quality, while 855 are now considered high quality.

Five different classification algorithms were employed to predict the wine quality: Decision Tree, Gaussian Naive Bayes (GNB), K-Nearest Neighbors (KNN), Logistic Regression,

and Random Forest. Each of these models represents a distinct approach to classification, and their performances will be compared to determine the most effective model for this problem.

DECISION TREE

A decision tree is a supervised machine learning model that classifies data by splitting it based on feature values. Each internal node in the tree represents a decision based on one feature, while each leaf node represents the class label (in this case, high or low wine quality). The Decision tree chooses how to set up its nodes based on the information gain of each choice (amount of information gained about what the prediction should be).

After creating the model using the data a cross validation test was run to gauge its performance. Cross-validation is a technique used to evaluate the performance of a model more reliably. Instead of training the model on the entire dataset and testing it on a single holdout set, cross-validation divides the data into multiple subsets, or "folds." The model is trained on some folds (training set) and tested on the remaining fold (validation set). This process is repeated for each fold, ensuring that every part of the data gets used for both training and testing. The results from each fold are averaged to give a more reliable estimate of the model's performance. Cross-validation helps reduce the risk of overfitting by providing a more generalized evaluation of the model, especially when the dataset is small or imbalanced.

This resulted in us receiving the following values:

Average Accuracy Score: 0.6422838050314466

Average Precision Score: 0.6733028969884028

Average Recall Score: 0.6512585499316006

Average F1-score: 0.6585872063075173

Accuracy is a simple metric that measures the overall proportion of correct predictions made by the model. It shows how many times the model correctly identified whether a wine was high or low quality out of all the predictions made. In this case, the accuracy score was 0.642, meaning the model correctly predicted the wine quality 64.2% of the time. While accuracy is helpful, it can be misleading when the data is imbalanced, as it doesn't account for how the model performs on each class separately (ie. it perfectly predicts all high quality wines but never low quality wines).

Precision measures the model's ability to correctly identify high-quality wines when it predicts them. It reflects the proportion of predicted high-quality wines that are truly high quality. In this case, the model's precision was 0.673, meaning that when the model predicted a wine to be high quality, it was correct 67.3% of the time. This metric is important when false positives (incorrectly classifying a wine as high quality) are costly.

Recall indicates how well the model identifies all of the actual high-quality wines. It measures the proportion of real high-quality wines that the model correctly predicted as such. The recall score was 0.651, meaning the model correctly identified 65.1% of the actual high-quality wines, but missed 34.9% of them. Recall is critical when the goal is to minimize false negatives, such as missing high-quality wines that should be identified.

F1-score is the combination of precision and recall, providing a single metric that balances the two. The F1-score of 0.659 reflects the balance between the model's ability to

correctly identify high-quality wines (precision) and the ability to capture all the high-quality wines (recall). This metric is particularly useful when dealing with imbalanced data, as it provides a more balanced evaluation by considering both the false positives and false negatives.

Together, these metrics offer a comprehensive understanding of the decision tree model's (and most model's) performance.

GAUSSIAN NAIVE BAYES

Gaussian Naive Bayes is a classification algorithm that works with continuous data, which is particularly useful for the objective and dataset. It is based on Bayes' Theorem, and assumes that features are independent of each other (the "naive" part) and that the features follow a normal distribution (the "Gaussian" part).

To classify an instance with features, the algorithm will assign it a class by computing the likelihood that the instance belongs to each class and then picking the one with the highest probability.

After running the cross validation test, the following results were produced:

Average Accuracy Score: 0.718561320754717

Average Precision Score: 0.7437682555737805

Average Recall Score: 0.7260191518467853

Average F1-score: 0.7291633297050873

K-NEAREST NEIGHBOURS

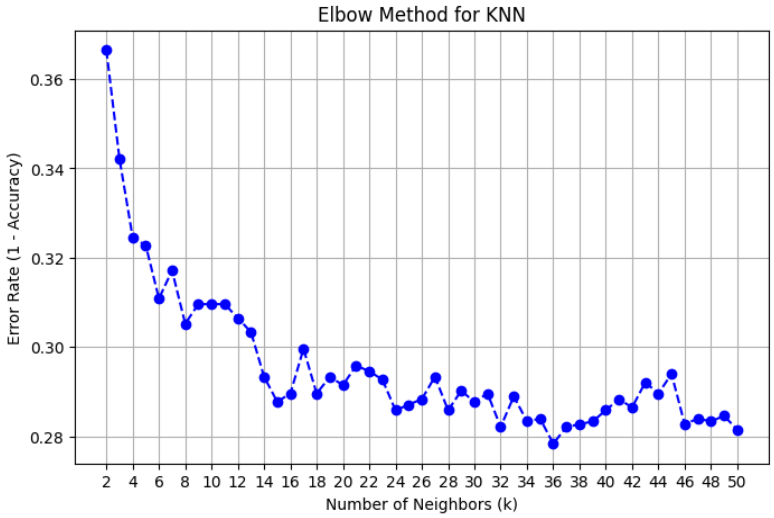
K-Nearest Neighbors (KNN) is a classification algorithm that works by comparing the input data to the training dataset and identifying the k-nearest data points (neighbors) based on a

distance metric (typically Euclidean distance) and classifying it as being a part of the majority class of these nearest neighbors.

Since KNN uses distance metrics to determine the proximity of data points, features with larger scales can disproportionately influence the distance calculation. For example, if one feature has values ranging from 0 to 1 and another ranges from 1000 to 10000, the distance will be dominated by the feature with the larger scale, skewing the results. To combat this feature scaling was used. Feature scaling, typically using normalization or standardization (standardization in our code), brings all features to a similar scale, ensuring that each feature contributes equally to the distance calculation. In this project, feature scaling was applied to ensure that all features had equal influence on the KNN algorithm and to improve the model's performance.

Finally, the optimal number of neighbors (k) for KNN must be determined, as the choice of k can significantly impact the model's performance. A small k may lead to overfitting, where the model captures noise in the data, while a large k can smooth out the decision boundary, leading to underfitting. To find the optimal k, different values of k were experimented with and the error rate (1 - accuracy score) was plotted for each k value which was collected using cross validation.

Plot showing error rate vs. number of neighbors



After analyzing the plot, the Kneed library (which is designed to find the "elbow" point) was used to find the optimal k . However, the Kneed library did not provide a clear optimal k value in this case (k = 6). A possible reason for this happening is due to the increase in error rate between k=6 and k=7 which causes the algorithm to immediately believe it found the elbow point even though visually it's apparent that the error rate drops substantially with only a couple more neighbours added. Therefore, using the chart the optimal k value was visually selected as k = 15 where the error rate appeared to stabilize.

Once the optimal k value was chosen, the KNN model was trained using this value of k and its performance was tested using cross-validation:

Average Accuracy Score: 0.7122916666666668

Average Precision Score: 0.7255055134746151

Average Recall Score: 0.7648837209302327

Average F1-score: 0.7375508877252328

LOGISTIC REGRESSION

Logistic regression is a binary classification model (although it can be used for multiclass problems). It works by computing the probability that an instance belongs to a class, then assigning that instance with the class that has the higher probability.

To do this, it first calculates weights for all the features as well as a bias during training, and this gives the equation for the model. To then classify an instance, the features are substituted in to give a value. The sigmoid function (the defining feature of logistic regression) is then applied to the value which transforms this continuous value to a probability between 0 and 1. This would give the probability that the instance belongs to class 1 (or in the case of this dataset, the probability that the wine is a high quality wine). Since this is a binary classification task and the sum of probabilities must equal 1, the probability that the instance belongs to class 0 (or a low quality wine in this case) is simply 1 minus the probability that the instance belongs to class 1.

For the dataset, features had different scales, which is why the features were first scaled to help with the model's learning process and prevent certain features from dominating.

After running the cross validation test, the following results were obtained:

Average Accuracy Score: 0.737932389937107

Average Precision Score: 0.7816803992625767

Average Recall Score: 0.7130232558139535

Average F1-score: 0.7374635847964772

RANDOM FOREST

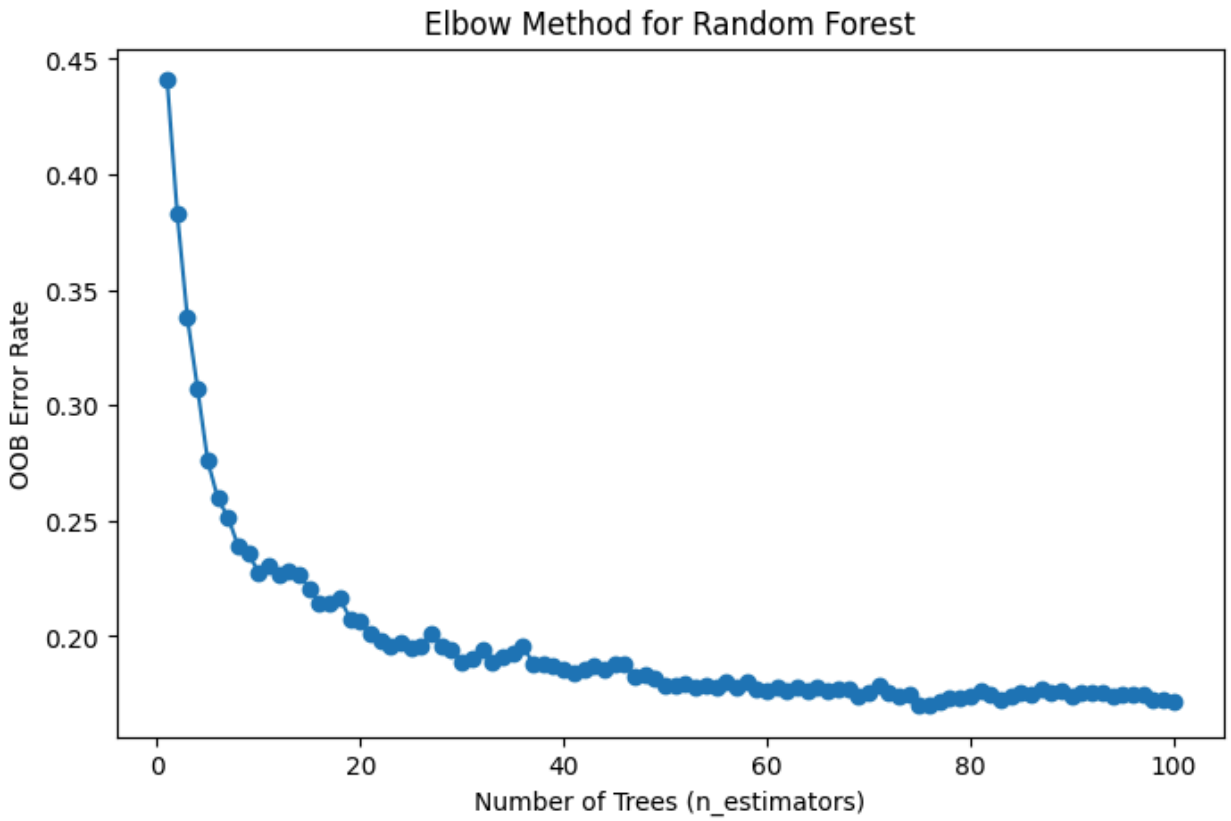
The fifth algorithm is a random forest. It creates multiple decision trees and combines their outputs to make a prediction. This makes random forests an ensemble machine learning algorithm. The "randomness" comes from 2 aspects: the way the data is sampled, and the random subset of features at each split.

The algorithm starts by using bootstrap sampling, where a random sample with replacement is taken from the original data, and this means each tree in the forest is trained on slightly different data. Next, a random subset of features is considered for each split, and the optimal one is chosen, similar to a decision tree. This process is repeated to grow the tree, until either the node is pure, the node can't be split further, or it reaches a depth that is defined by the user.

Finally, to make a prediction given an instance, bagging is used. The instance runs through every tree in the forest, and the tree gives its prediction. The results are aggregated, and in the case of this dataset and objective, if there are more trees that predict the wine is high quality than those that predict the wine is low quality, the wine is predicted to be high quality.

The first step in using a random forest for this objective was to determine the optimal number of trees in the forest. To do this, the out-of-bag (OOB) error rate was calculated for each number of trees from 1 to 100. Since bootstrapping was used, each tree is trained on 1599 data points (the size of the dataset), however, since each training dataset is sampled with replacement, this means some of the data points were not used to train that tree. These left-out data points are referred to as "out-of-bag" data. Finally, to give the forest an OOB score, each instance from the original dataset is passed through every tree in the forest where that instance was considered

out-of-bag (i.e. the tree wasn't trained on that instance). This gives an OOB score, and 1 - OOB score gives the out-of-bag error rate. The OOB error rate was saved for each forest size, and this data was plotted:



To determine the optimal number of trees (n_estimators), the kneed library was used, and it determined that the optimal number of trees for the forest is 10.

Finally, a random forest model was trained with 10 trees and the following results were produced:

Average Accuracy Score: 0.7098309748427674

Average Precision Score: 0.7558314517049827

Average Recall Score: 0.6944049247606019

Average F1-score: 0.709057076100387

COMPARISON BETWEEN MODELS

For the purpose of simplicity, the following abbreviations will be used to refer to the models:

DT → Decision Tree

GNB → Gaussian Naive Bayes

KNN → K-Nearest Neighbours

LR → Logistic Regression

RF → Random Forest

Model	**Accuracy**	**Precision**	**Recall**	**F1**
Decision Tree	0.6422838050	0.6733028969	0.6512585499	0.6585872063
Gaussian Naive Bayes	0.7185613207	0.7437682555	0.7260191518	0.7291633297
K Nearest Neighbours	0.7122916666	0.7255055134	**0.7648837209**	**0.7375508877**
Logistic Regression	**0.7379323899**	**0.7816803992**	0.7130232558	**0.7374635847**
Random Forest	0.7098309748	0.7558314517	0.6944049247	0.7090570761

DT consistently shows the lowest performance among the five models, indicating possible overfitting or difficulty in capturing the data's complexity. In contrast, GNB is a bit more well-rounded with strong accuracy, precision, and recall despite its assumption of feature independence and a Gaussian distribution.

KNN stands out for its highest recall (0.7649) and F1 (0.7376), meaning it effectively identifies high-quality wines (fewer false negatives). Its accuracy (0.7123) is still quite good compared to all the other models but slightly behind LR (0.7379). LR achieves both the highest accuracy (0.7379) and precision (0.7817), suggesting that it minimizes false positives and makes many correct predictions overall.

RF delivers solid performance, though it falls slightly short of LR and KNN in most metrics. It has good precision (0.7558), but its recall (0.6944) lags behind KNN and GNB. Ultimately, the best model depends on the objective.

If fewer false negatives are needed (ie. you are trying find as many high quality wines to try as possible), KNN's high recall is the best choice. Otherwise, for most cases it's best to use LR instead or any of the other models excluding DT as it is noticeably worse.

FEATURE SELECTION

Feature selection is the process of identifying and retaining only the most informative attributes of a dataset, which helps improve model performance, reduces overfitting, and often speeds up training. In this project, a mutual information (MI) algorithm, which calculates how much knowing the value of one variable reduces uncertainty about the target. After computing MI scores for each feature, those with an MI score of 0 were removed as they provide no value to our models and may cause unintended . This resulted in free sulfur dioxide and pH being removed. The remaining features were retrained in the five models. Hyperparameters were also optimized where necessary. For KNN, the number of neighbors was re-evaluated and for RF the optimal number of trees was re-evaluated.

DECISION TREE (WITH FEATURE SELECTION)

Another Gaussian Naive Bayes model was trained on the new set of features and the following results were produced:

Average Accuracy Score: 0.6559905660377359

Average Precision Score: 0.6781553243987701

Average Recall Score: 0.6721203830369358

Average F1-score: 0.673221608600014

GAUSSIAN NAIVE BAYES (WITH FEATURE SELECTION)

Another Gaussian Naive Bayes model was trained on the new set of features and the following results were produced:

Average Accuracy Score: 0.7210534591194968

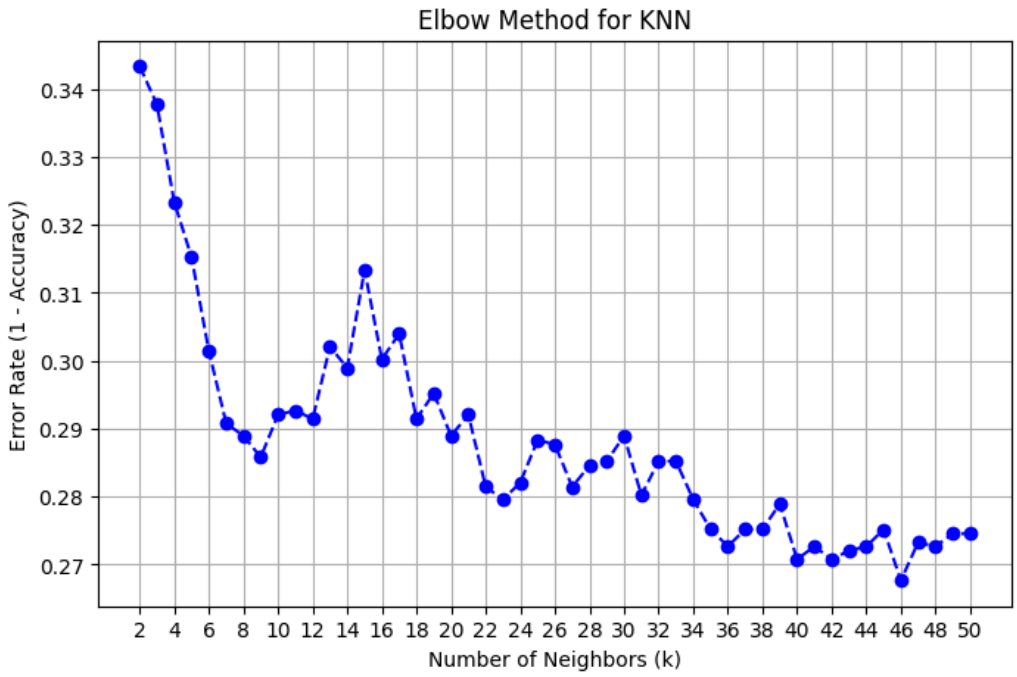
Average Precision Score: 0.7519399296966622

Average Recall Score: 0.7225034199726402

Average F1-score: 0.7295690423006366

K-NEAREST NEIGHBOURS (WITH FEATURE SELECTION)

With a new set of features, the optimal amount of neighbours to use when training the model must be re-evaluated. By utilizing the same approach as done previously with error rate the following elbow method graph was plotted:



Unlike when it was attempted to find the elbow point without feature selection, in this case the Kneed library returned an optimal k value (k = 9) that was deemed to be acceptable according to the plot.

After training the new KNN model with the new set of features and an optimal k value of 9, the following results were obtained:

Average Accuracy Score: 0.714190251572327

Average Precision Score: 0.7214142632494099

Average Recall Score: 0.7741997264021888

Average F1-score: 0.7400936504357356

LOGISTIC REGRESSION (WITH FEATURE SELECTION)

After feature selection, the new set of features were scaled again, and a new logistic regression model was trained on the new set features. The following results were obtained:

Average Accuracy Score: 0.736069182389937

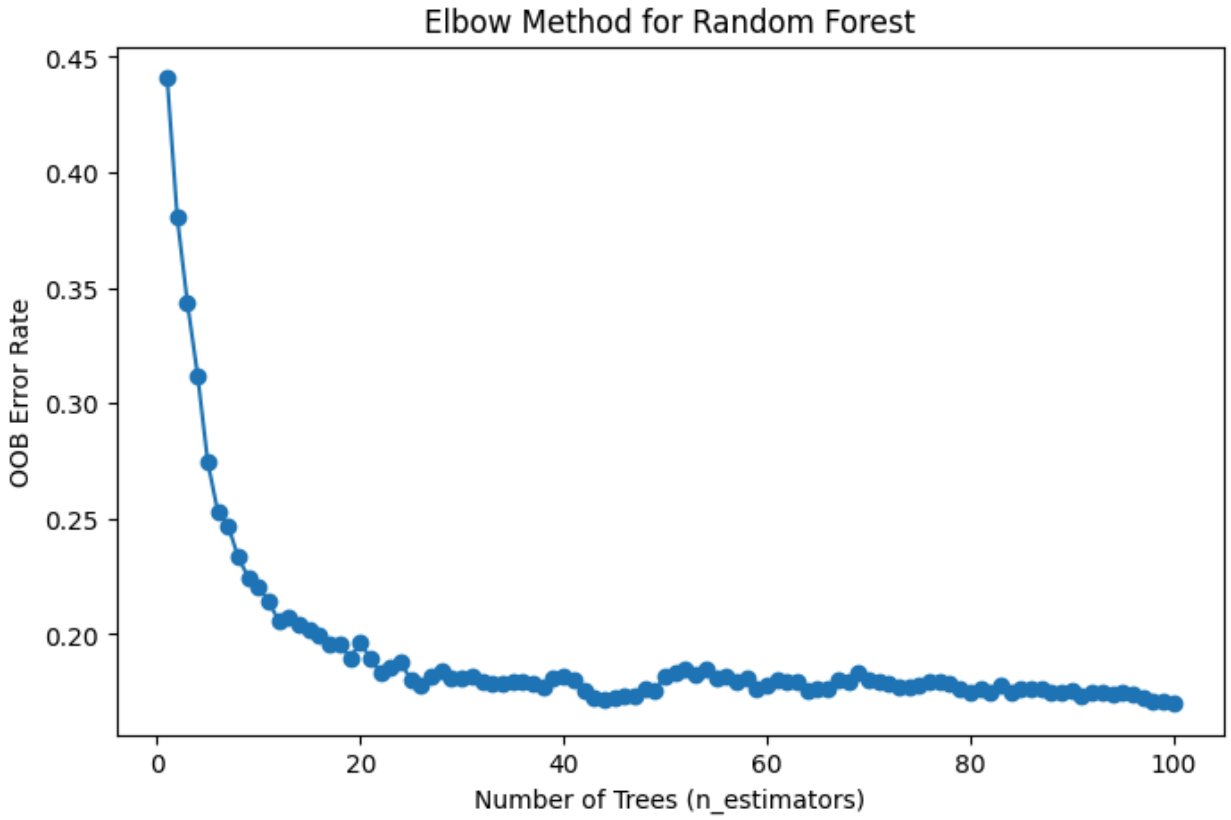
Average Precision Score: 0.7747973892233823

Average Recall Score: 0.7177154582763339

Average F1-score: 0.7383282777079325

RANDOM FOREST (WITH FEATURE SELECTION)

With a new set of features, the new optimal number of trees needed to be determined. The same process was followed, starting by plotting the OOB error rate for each number of trees from 1 to 100.



To determine the optimal number of trees, the Kneed library was used again which determined that the optimal value is 12.

A new random forest model was trained with 12 trees and using the new set of features, and the following results were obtained:

Average Accuracy Score: 0.7247916666666667

Average Precision Score: 0.7737324400410573

Average Recall Score: 0.7014090287277701

Average F1-score: 0.7250187322140318

COMPARISON BETWEEN MODELS (WITH FEATURE SELECTION)

To easily visualize the results, they have been put into a tabular format:

Model	**Accuracy**	**Precision**	**Recall**	**F1**
Decision Tree	0.655990566	0.6781553244	0.672120383	0.6732216086
Gaussian Naive Bayes	0.7210534591	0.7519399297	0.72250342	0.7295690423
K Nearest Neighbours	0.7141902516	0.7214142632	**0.7741997264**	**0.7400936504**
Logistic Regression	**0.7360691824**	**0.7747973892**	0.7177154583	0.7383282777
Random Forest	0.7247916667	**0.77373244**	0.7014090287	0.7250187322

The first thing that stands out is how poorly the decision tree did. None of the 4 metrics it was tested on reached the >= 0.7 range, while the other models did for all 4 metrics. This makes the decision tree the worst model by far. For the purpose of simplicity, it will not be compared with the remaining 4 models. In terms of accuracy, the 4 models had a similar score, but LR performed the best. This means it was able to make correct predictions more than the other

models. When it comes to precision, it again performed the best, while RF also performed very well. This means that whenever they predicted a wine to be high quality, they were correct a lot of the time, whereas GNB and KNN made many false positive predictions. Moving onto recall, surprisingly KNN did very well compared to the other models, meaning that it could identify many of the high quality wines, whereas GNB, LR, and RF made more false negative predictions. Due to this, KNN had the highest F1-score. Looking at these results, the LR model had the best predictive power, and along with RF, they both were very good at properly classifying high quality wines. This meant that the models seemed to be conservative, which makes sense considering their lower recall and worse ability to identify more high quality wines. On the opposite side, KNN seemed to be more relaxed and was more willing to classify a wine as high quality, but this led to more false positives and therefore a lower precision. Despite this, it had the best balance of precision and recall and the highest F1-score.

Determining which of the models is the best depends on the context of the problem. If the goal is to simply make correct predictions, then LR would be the best model. However, take a wine store owner who wants to decide which wines to put on the front display. They want to ensure that each wine they select for the front display is high quality, and having false positives or low quality wines on that front display when they believed they were high quality may lead to dissatisfied customers and lower brand reputation. In this case, the optimal model would be LR or RF as these models have a high precision. Now let's say someone wants to put together a database of high quality wines, and they want to have as many high quality wines in this database as possible. They want to be able to identify a large number of high quality wines, and having a low recall means there will be many false negatives and are therefore missing out on

many of the high quality wines that could have been added to the database. In this scenario, KNN is by far the best model as it is good at identifying as many of the high quality wines as possible. There are cases where a balance of precision and recall is needed, KNN would be the optimal model as it has the highest F1-score of the models.

COMPARISON BETWEEN MODELS (WITH AND WITHOUT FEATURE SELECTION)

When comparing the Decision Tree model's performance before and after feature selection, all four metrics; accuracy, precision, recall, and F1; improve slightly. These gains suggest that removing uninformative features (in this case, based on mutual information scores) helps the tree concentrate on the attributes that truly matter for determining wine quality. By excluding noise or irrelevant information, the splits the tree makes are more likely to accurately separate high and low-quality wines, improving the overall effectiveness of its classification.

For GNB, the accuracy and precision improve slightly (accuracy increases from 0.7186 to 0.7211, and precision increases from 0.7438 to 0.7519). This shows that removing the free sulfur dioxide and pH features reduced the noise and allowed the model to make more correct predictions, as Naive Bayes assumes feature independence, and removing noisy features can help. It also made the model more conservative when predicting positives, and removing misleading features helped it develop stronger indicators for which wines should actually be classified as high quality. However, the model experienced a very small decline to its recall. (from 0.7260 to 0.7225). Removing these features made it harder for the model to find all the high quality wines, meaning these features had some signals that helped the model catch edge cases, and therefore led to more false negatives. The model saw very negligible improvement to its F1-score (from 0.7292 to 0.7292). Balancing precision and recall with an improved F1-score,

as well as improved accuracy, proves that a Gaussian Naive Bayes model is likely to benefit from removing features (in this case, removing those with a 0 MI-score).

For KNN, accuracy and recall both see slight improvements (accuracy rises from 0.7123 to 0.7142, recall from 0.7649 to 0.7742), while precision dips marginally (0.7255 to 0.7214). Overall, the F1-score also rises (0.7376 to 0.7401), indicating that removing the uninformative features helps the model focus more effectively on truly relevant attributes. One of the possible reasons for these changes is that, without the zero-value MI features, KNN's distance calculations become slightly more discriminative. In other words, each remaining feature contributes more meaningfully to identifying high-quality wines, resulting in fewer missed positives (improved recall). However, with distance-based classification being sensitive to small changes in the feature set, the slightly broader classification boundaries can lead to a minor increase in false positives, which explains the small drop in precision.

For the logistic regression models, there was a very slight decrease in accuracy (0.7379 to 0.7360). The precision also worsened by a small amount (0.7817 to 0.7748). Both these decreases could mean that these removed features, despite having a zero-value MI score, actually helped with the predictive power of the model, as well as giving the model more information on how to properly classify an instance as positive. Recall saw a minute improvement (0.7130 to 0.7177), and this could mean that the model was willing to classify more instances as positive (which lead to a higher positive catch rate but also lead to a higher false positive rate). Therefore, the model with feature selection experienced a very small improvement to its F1-score (0.7375 to 0.7383). Even with a decrease in precision, the increase in recall helped improve the F1-score.

The random forest model greatly benefited from feature selection. There was an increase in accuracy (from 0.7098 to 0.7248), indicating that these features were noisy and decreased the predictive power of the model. Although random forests can handle many features, they can still benefit from feature selection. Precision also improved (0.7558 to 0.7737), which means the model was better at properly classifying a wine as high quality, leading to less false positives. Removing these misleading features may have caused cleaner splits, allowing the model to more confidently differentiate between true positives and false positives. The model saw an increase in its recall (0.6944 to 0.7014) which shows that the model was now better at catching those high quality wines and reducing false negatives. As a by-product, the model experienced an increase in its F1-score (0.7091 to 0.7250), proving that the model is much more reliable. As seen above, the individual decision tree greatly benefitted from feature selection. These misleading and noisy features hurt the predictive power of each individual tree, which in turn worsens the entire forest. Removing these features improves each individual tree, and therefore the whole forest

CONCLUSION

In this project, multiple classification algorithms such as Decision Tree (DT), Gaussian Naive Bayes (GNB), K-Nearest Neighbors (KNN), Logistic Regression (LR), and Random Forest (RF) were employed to predict the quality of Portuguese red wines. The performance of these models using accuracy, precision, recall, and F1-score, revealing each model's strengths and weaknesses. Although DT exhibited the weakest performance, GNB demonstrated consistency despite its simplifying assumptions, and KNN excelled in recall by capturing a high proportion of genuinely high-quality wines. Meanwhile, LR achieved the highest accuracy and precision, making it a strong choice for tasks when minimizing false positives. RF also benefited

the most from the removal of the least informative features using feature selection, thereby enhancing its predictive power. Ultimately, the choice of model depends on whether the primary concern is identifying as many high-quality wines as possible (favoring high recall), or ensuring that predictions of high-quality wines are highly accurate (favoring high precision). These findings highlight the importance of feature selection, as omitting uninformative or misleading attributes can greatly improve a model's effectiveness and reliability.

Works Cited

Cortez, Paulo, et al. "Wine Quality." UCI Machine Learning Repository, 2009, https://doi.org/10.24432/C56S3T.