CM3015 Mid-Terms Report

Abstract

This project explores the application of machine learning models for the classification of wine types using the wine dataset from scikit-learn. Two distinct approaches are implemented: one utilising a k nearest neighbour (knn) machine learning model developed from scratch, and the other a decision tree classifier, leveraging established machine learning libraries. The objective is to assess the accuracy of wine type classification based on selected features from the dataset and identify the most effective model for this task. Through comprehensive experimentation and evaluation, the project reveals insights into the performance and efficiency of the two models, contributing valuable findings to the field of machine learning applied to wine classification.

Introduction

Machine learning is an important tool in data analysis and classification tasks capable of extracting meaningful patterns from complex datasets and finding relationships that we cannot see. This project leverages the wine dataset from scikit-learn to apply machine learning techniques to the issue of identifying different types of wine. The main objective is to gain a deeper understanding of machine learning models through the investigation of two different approaches: creating a model from the ground up and using pre-existing machine learning libraries. By creating a machine learning model from the ground up, I want to learn more on the fundamental processes that lead to efficient classification. By using a model made using established libraries, I can use it as a benchmark to compare the performances of my model.

While the wine dataset is a dataset inherently suited for classification models, one challenge was the different diverse scales the feature values are measured in. This mirrors real world issues as well since not every dataset will have its values on a similar scale. In order to ensure that the models remain robust across a variety of datasets, this project includes ways to standardise feature scales and emphasises the importance of these issues.

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od3
173	13.71	5.65	2.45	20.5	95.0	1.68	0.61	0.52	1.06	7.7	0.64	
174	13.40	3.91	2.48	23.0	102.0	1.80	0.75	0.43	1.41	7.3	0.70	
175	13.27	4.28	2.26	20.0	120.0	1.59	0.69	0.43	1.35	10.2	0.59	
176	13.17	2.59	2.37	20.0	120.0	1.65	0.68	0.53	1.46	9.3	0.60	
177	14.13	4.10	2.74	24.5	96.0	2.05	0.76	0.56	1.35	9.2	0.61	

fig.1 : Feature values in different scales

Background

For my project I have decided to go with kNN and decision tree classifiers. kNN is a machine learning technique that can be applied to both regression and classifier tasks. It works by locating a specified number of neighbours or data points (k) nearest to the sample. Based on what the majority of the data points are, the sample is then classified as such. For instance when classifying between apples and oranges, if most of the fruit surrounding the sample are apples, the sample is classified as an apple as well.

A decision tree recursively splits a dataset based on the most significant features at each node. In classification tasks, decisions are made by splitting data into subsets based on different conditions, starting from a single box known as the root and splitting the data down the tree structure until it reaches boxes at the end, known as leaf nodes. Each node corresponds to a feature and each leaf node represents the prediction or outcome. It functions similarly to a decision based flowchart where questions are asked to guide the decision to a conclusion.

Methodology

Exploratory data analysis

After importing the wine dataset, 'df_wine.describe()' was used to view the various features included in the wine dataset.

```
df_wine.describe()
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	17
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	
mean	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.029270	0.361854	1.590899	5.058090	
std	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	0.998859	0.124453	0.572359	2.318286	
min	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.340000	0.130000	0.410000	1.280000	
25%	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.205000	0.270000	1.250000	3.220000	
50%	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.135000	0.340000	1.555000	4.690000	
75%	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	2.875000	0.437500	1.950000	6.200000	
max	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.080000	0.660000	3.580000	13.000000	

Fig 2: Results of pandas .describe() method

From the displayed results shown in fig 2, we can see that the features are in different scales. As such a scaler would be needed to scale the data to ensure accuracy. From the results we can also see that there are a total of 178 entries in the dataset and as such k-fold cross-validation can be used due to the small dataset. Next I also had to determine whether the dataset is imbalanced or not as well as the baseline accuracy. The following image shows the code used to achieve this.

```
features_selected = ['flavanoids','color_intensity','proline','hue']
X = df_wine[features_selected].values
y = wine.target
df_wine['label'] = wine.target
df_wine.label.value_counts(normalize=True).round(3)
```

```
1    0.399
0    0.331
2    0.270
Name: label, dtype: float64
```

```
#baseline caluculation
from sklearn.metrics import accuracy_score

majority_class = df_wine['label'].mode()[0]
baseline_predictions = [majority_class] * len(df_wine)
baseline_accuracy = accuracy_score(df_wine['label'], baseline_predictions)

print("Baseline accuracy:", baseline_accuracy)
```

```
Baseline accuracy: 0.398876404494382
```

Fig 3: Baseline accuracy and balance

From the image shown, we can determine that it is a relatively balanced dataset and should use accuracy as the evaluation metric. The baseline accuracy serves as a performance

benchmark to compare the machine learning model with. This will give an idea on whether the machine learning model applied is suitable for the dataset.

Next I also plotted a heatmap to show the correlation between the various features in the dataset.

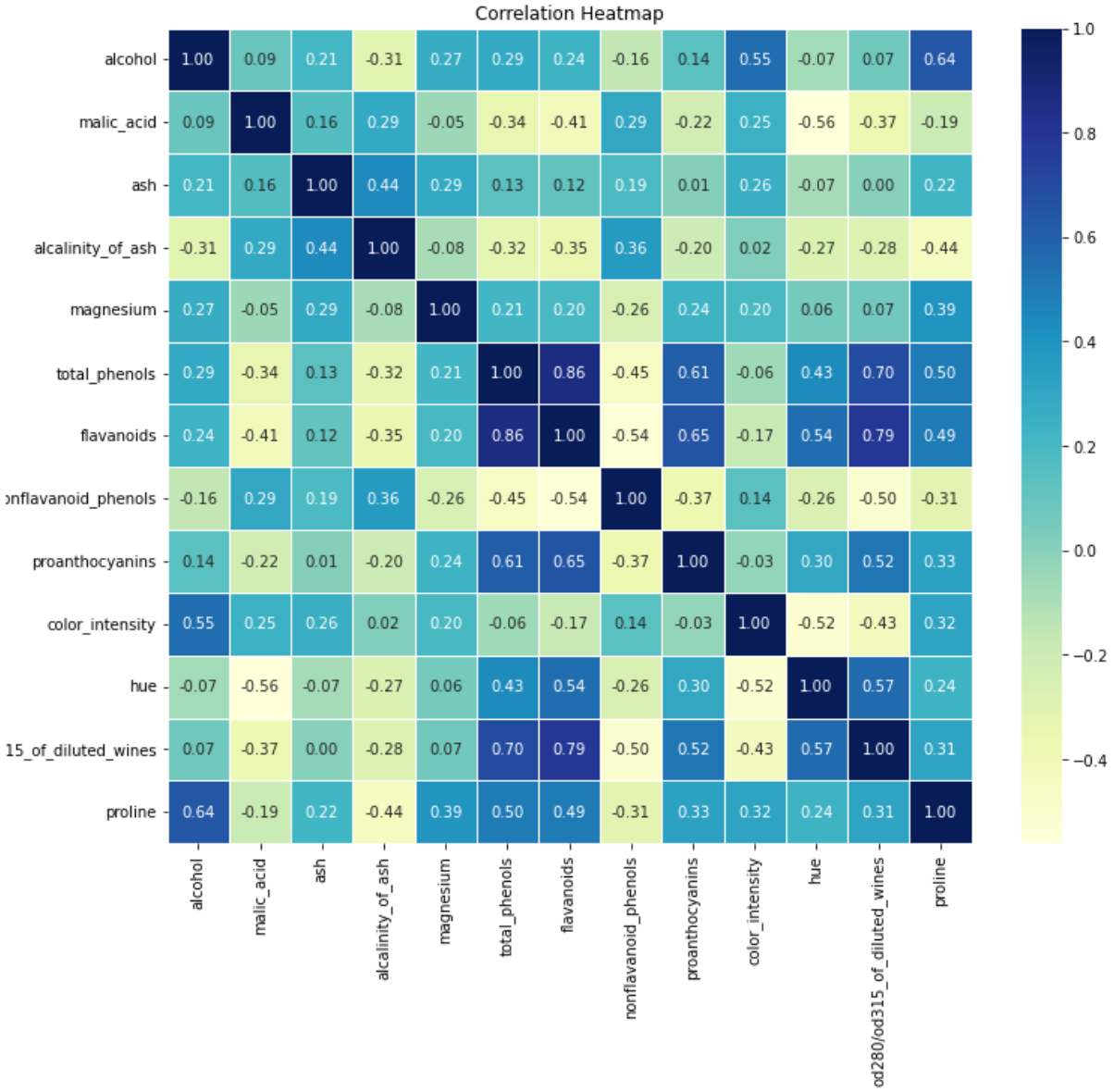

Fig 4: correlation heatmap

From the heatmap I decided to use flavonoids, colour intensity, hue and proline as features to be used in the machine learning models due to their relatively low correlations with each other. Low correlation between features is good in a machine learning model as it means each feature brings unique information to the model, which can hopefully enhance the model's performance. Following this a pairplot was made

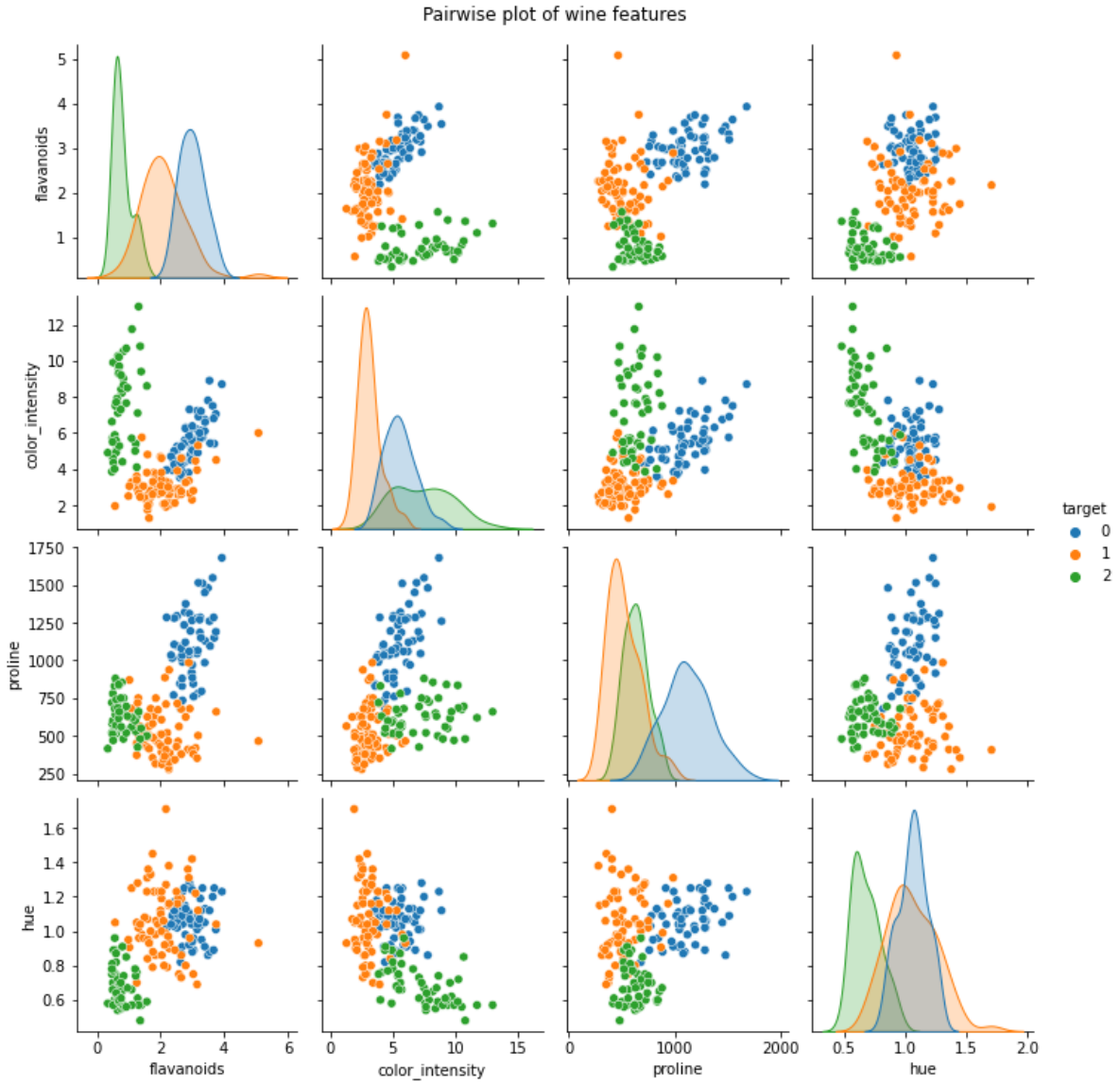

Fig 5: pairplot of selected features.

Building the models

The functions required for the kNN model are then made. The functions are ‘my_train_test_split’, ‘distance_calculator’ and ‘mykNN’. ‘My_train_test_split’ is used to shuffle and divide the dataset into training and test data to be used and ‘distance_calulator is used to calculate the distance between data points. ‘mykNN’ finds the distances between each test point and all training points using the ‘distance_calculator’ function. Next, for each test point, it finds the k training points that are closest to it. This is done by sorting the distances and selecting the top k ones. Finally, for each test point, it predicts the label by looking at the labels of the k closest training points. It uses a majority vote to decide the predicted label: the label that appears most frequently among the k neighbours is chosen as the prediction. If there’s a tie, it selects the label with the smallest value due to the behaviour of the np.argmax function. The then function returns an array of predicted labels for the test data.

A cross validation function was then implemented by hand to find the best parameters for the mykNN function. In this case, it is finding the best k values to be used for mykNN. This is necessary to provide a more robust estimate of the model's performance by averaging the model's effectiveness over several different partitions of the data. This helps to ensure that the model's performance is not overly dependent on the specific way the data was split. The function takes in the dataset 'X' , the labels 'y', the number of folds and the evaluation metric. A list of indices for the data is created and shuffled. Then, it splits these indices into the specified number of folds, ensuring that each fold is approximately the same size. Each fold is then iterated over and the kNN model is trained on the training set and makes predictions on the test set. Depending on the evaluation metric either accuracy or F1 score is calculated. The k scores are stored in a dictionary and the k with the highest score is returned. Afterwards the best k values are implemented into the function which will hopefully improve its accuracy..

For the second machine learning model, I made use of the decision tree classifier provided by sklearn. This was much simpler to implement as the functions were already defined by the library. Initially the training data was used and an arbitrary depth was used to find the accuracy of the predictions. Afterwards cross validation is implemented again to find the best max depth to use. The best max depth and test data is then used on the model. Additionally a confusion matrix was generated for each machine model to visualise the performance of each model.

Results

	Accuracy before validation	Accuracy after validation
kNN	0.962	0.981
decisionTreeClassifier	0.937	0.944

```
Predicted values:
[2 0 2 0 2 0 0 0 0 1 2 0 0 2 2 0 2 2 0 0 1 2 0 2 2 0 2 0 2 0 1 1 2 1 0 1 2
 2 0 2 1 1 1 2 1 0 1 2 2 0 0 0 1]

True values:
[2 0 2 0 2 0 1 0 0 1 2 0 0 2 2 0 2 2 0 0 1 2 1 2 2 0 2 0 2 0 1 1 2 1 0 1 2
 2 0 2 1 1 1 2 1 0 1 2 2 0 0 0 1]

Accuracy:
0.9622641509433962
```

Fig 6: Accuracy of mykNN before using parameters from cross validation

```
Predicted values:
[2 0 2 0 2 0 1 0 0 1 2 0 0 2 2 0 2 2 0 0 1 2 1 2 2 0 2 0 2 0 1 1 2 1 0 1 2
 2 0 2 1 1 1 2 2 0 1 2 2 0 0 0 1]

True values:
[2 0 2 0 2 0 1 0 0 1 2 0 0 2 2 0 2 2 0 0 1 2 1 2 2 0 2 0 2 0 1 1 2 1 0 1 2
 2 0 2 1 1 1 2 1 0 1 2 2 0 0 0 1]

Accuracy:
0.9811320754716981
```

Fig 7: Accuracy of mykNN after using parameters from cross validation

```
Best k: 1
Best accuracy: 0.7299435028248588
accuracy for each k: {1: 0.7299435028248588, 3: 0.7191148775894538, 4: 0.6797551789077213, 5: 0.6402071563088513, 6: 0.7079
4519774, 7: 0.6795668549905839}
```

Fig 8: best k result from cross validation from scratch

```
Predicted values:
[2 2 0 2 0 1 1 1 2 0 1 1 2 0 0 0 0 2 2 1 1 0 1 0 2 1 1 2 0 0 0 2 0 0 1 2 1
 0 2 1 0 2 0 1 0 1 0 0 1 1 0 2 0 1 1 0 1 1 1 2 2 0 1 2 2 0 1 0 1 2 2 1 2 1
 1 1 0 0 2 0 2 0 0 1 1 0 0 0 1 0 1 2 1 1 0 2 2 1 0 0 1 2 2 0 1 2 2 2 2 0 0
 1 0 2 0 0 1 0 0 2 1 0 2 2 0 0 2 2 2 1 1 1 1 1 0 2 0 1 1 0 1 1]

True values:
[2 2 1 2 0 1 1 1 2 0 1 1 2 0 1 0 0 2 2 1 1 0 1 0 2 1 1 2 0 0 0 2 0 0 1 2 1
 0 2 1 0 2 1 1 0 1 0 0 1 0 0 2 1 1 1 0 1 1 1 2 2 0 1 2 2 1 1 0 1 2 2 1 2 1
 1 1 0 0 2 0 2 0 0 1 1 0 0 0 1 0 1 2 1 1 1 2 2 1 0 0 1 2 2 0 1 2 2 2 2 1 0
 1 0 2 0 0 1 0 0 2 1 0 2 2 0 0 2 2 2 1 1 1 1 1 1 2 0 1 1 0 1 1]

Accuracy:
0.9366197183098591
```

Fig 9: Accuracy of decision tree before using parameters from cross validation using train data

```
Predicted values:
[0 0 2 0 1 0 1 2 1 2 1 1 0 1 0 1 1 1 0 1 0 1 1 2 2 2 1 1 1 0 0 1 2 0 0 0]

True values:
[0 0 2 0 1 0 1 2 1 2 0 2 0 1 0 1 1 1 0 1 0 1 1 2 2 2 1 1 1 0 0 1 2 0 0 0]

Accuracy:
0.9444444444444444
```

Fig 10: Accuracy of decision tree after using parameters from cross validation using test data

```
Best max_depth: 3
Best cross-validation score: 0.9362068965517242
```

Fig 11: Best max depth provided by sci kit cross validation

The table shows the accuracy values for each model before and after validation. The validation scores are also shown in the images. As stated in the methodology, each algorithm was initialised once before cross validation was done. After which the best parameters were used with the machine model to generate another accuracy score. After getting the final accuracy score we can then compare and contrast the performance of each machine learning model.

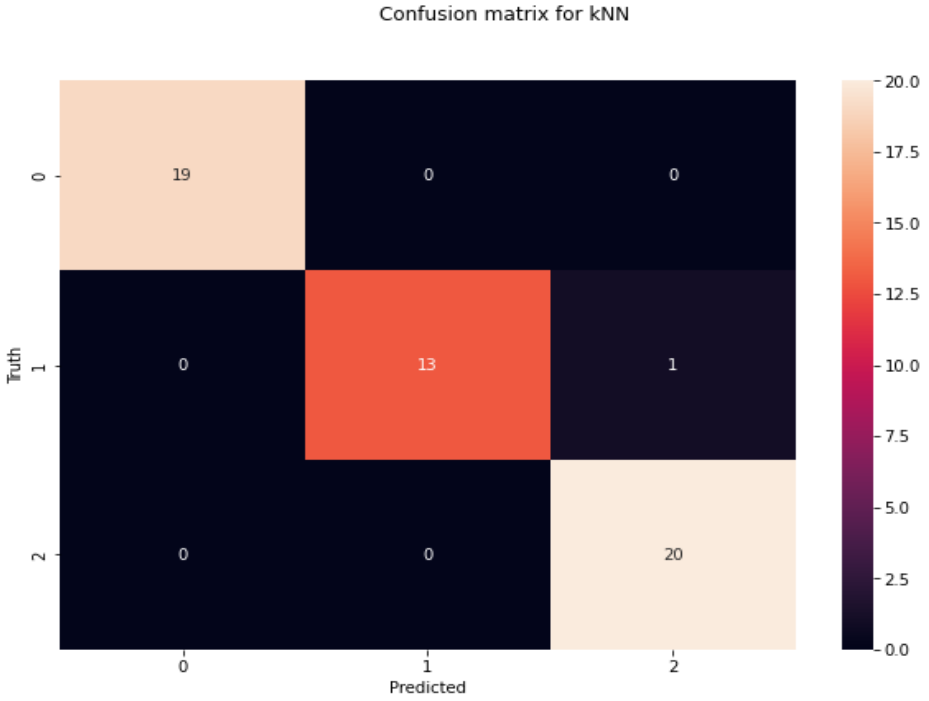

Fig 12: confusion matrix for kNN

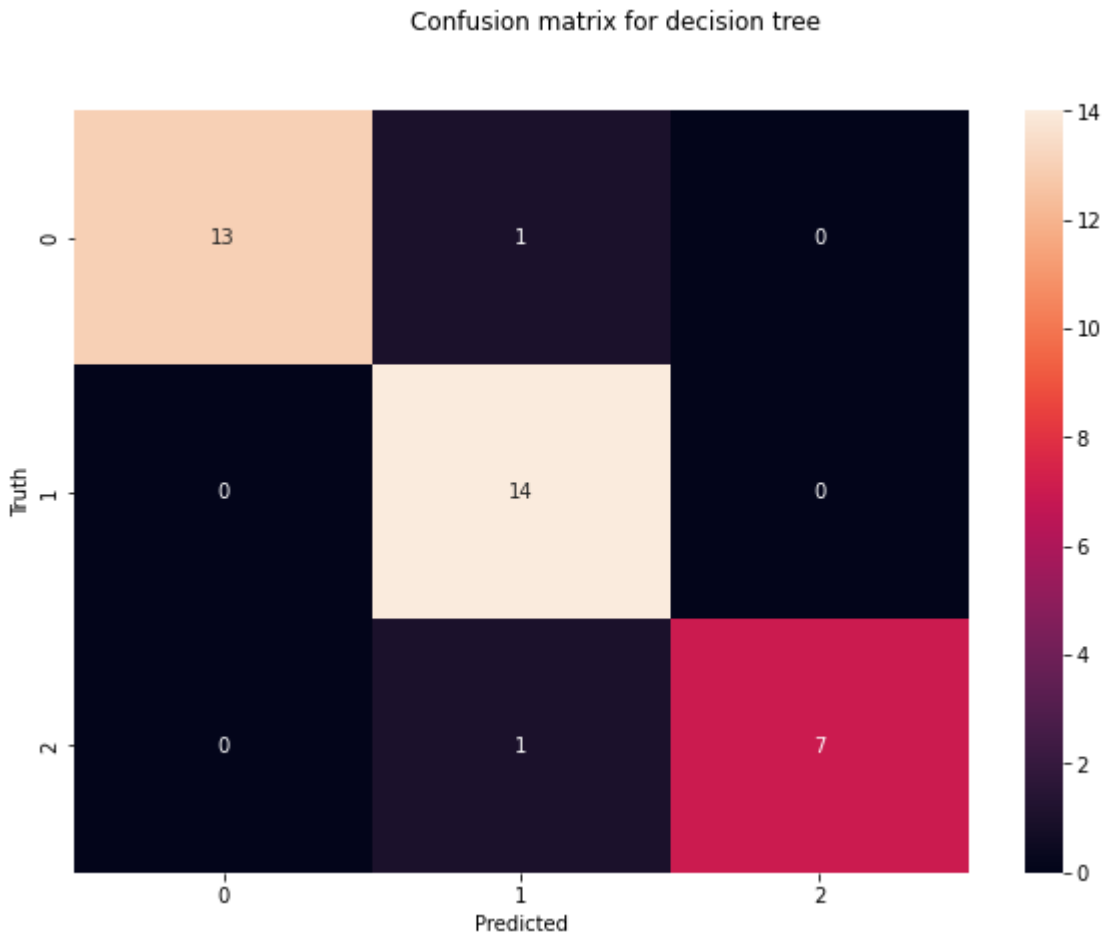

Fig 13: confusion matrix for decision tree classifier

Figure 12 and 13 shows the confusion matrix for kNN and decision tree classifier respectively. We can see that there were relatively few errors with the predictions for both models after using the best parameters from cross validation. kNN only had one incorrect prediction while the decision tree only had two incorrect predictions. From the confusion matrices we can surmise that both models perform relatively well based on the selected features. From the table summarising accuracy scores we can see that kNN performed the best at a score of 0.98 after cross validation. In the context of this project, it would seem that kNN is the best machine learning model for identifying wine types based on the selected features.

Evaluation

The machine learning models built in this project were able to correctly identify the correct wine types based on the selected features with an accuracy of above 0.9. I consider this number to be indicative of the success of this project. However the project is not without its issues. For instance in the decision tree classifier model, the performance was better on training data with an accuracy of 0.98 (fig 14) compared to the score on test data at 0.94. While it is not a significant drop and not an uncommon occurrence, there are probably ways to avoid this small decline in performance. I could have checked for fitting issues to prevent outliers and noise from interfering with the data. There should also have been checks for

outliers in the dataset to prevent the possibility of outliers affecting the model. I could also have experimented with different features of the set to test the accuracy of the models. Due to time constraints, I was not able to implement all these potential improvements to the project. Despite these small issues, I would still consider the project a general success.

```
Predicted values:
[2 2 1 2 0 1 1 1 2 0 1 1 2 0 1 0 0 2 2 1 1 0 1 0 2 1 1 2 0 0 0 2 0 0 1 2 1
 0 2 1 0 2 1 1 0 1 0 0 1 1 0 2 1 1 1 0 1 1 1 2 2 0 1 2 2 1 1 0 1 2 2 1 2 1
 1 1 0 0 2 0 2 0 0 1 1 0 0 0 1 1 1 2 1 1 1 2 2 1 0 0 1 2 2 0 1 2 2 2 2 1 0
 1 0 2 0 0 1 0 0 2 1 0 2 2 0 0 2 2 2 1 1 1 1 1 1 2 0 1 1 0 1 1]

True values:
[2 2 1 2 0 1 1 1 2 0 1 1 2 0 1 0 0 2 2 1 1 0 1 0 2 1 1 2 0 0 0 2 0 0 1 2 1
 0 2 1 0 2 1 1 0 1 0 0 1 0 0 2 1 1 1 0 1 1 1 2 2 0 1 2 2 1 1 0 1 2 2 1 2 1
 1 1 0 0 2 0 2 0 0 1 1 0 0 0 1 0 1 2 1 1 1 2 2 1 0 0 1 2 2 0 1 2 2 2 2 1 0
 1 0 2 0 0 1 0 0 2 1 0 2 2 0 0 2 2 2 1 1 1 1 1 1 2 0 1 1 0 1 1]

Accuracy:
0.9859154929577465
```

Fig 14: score of decision tree on training data.

Conclusion

This project set out to build two machine learning algorithms to determine which of them was better at identifying wine types based on a set of selected features. Two models, kNN which was done from scratch and decision tree classifiers made using machine learning libraries were utilised in this project. The two models have their accuracy compared after using their respective best parameters for cross validation and the results would indicate that kNN is a better model for identifying wine types. While there are some small issues, the project was able to compare the accuracy of both models effectively and determine the better model. In the future I hope to be able to utilise what I have learnt from this project and prevent such issues from occurring.

References

1.Harrison, O. (2019) *Machine learning basics with the K-nearest neighbors algorithm*, *Medium*. Available at: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761 (Accessed: 27 December 2023).

2. Capozzoli, A., Cerquitelli, T. and Piscitelli, M.S. (2016) *Decision Tree Classifier*, *Decision Tree Classifier - an overview | ScienceDirect Topics*. Available at: https://www.sciencedirect.com/topics/computer-science/decision-tree-classifier#:~:text=The%20decision%20tree%20classifier%20(Pang,possible%20values%20for%20that%20attribute. (Accessed: 27 December 2023).