CT5165 Principles of Machine Learning
Assignment 1 - Hands on data and ML packages and algorithms

Chin Zhe Jing - MSc DA - 22221970

In this assignment, we have given 2 datasets including the training and testing data to predict whether a plant is classified as “setosa” or “virginica” based on 4 independent continuous values variables which including the sepal length, sepal width, petal length and petal width.

Base on the dependent variable in the given data, the problem is to predict whether the target plant should be a setosa or virginica, hence it should be solved by a binary classification supervised learning algorithm since there is labelled data and there is only 2 classes. There are a total of 40 setosa and 40 virginica in the training set, hence it is not an imbalanced dataset.

Scikit-learn should be an ideal open-source machine learning package for this task as it comes with a lot of different algorithms such as Naïve Bayes or Decision Tree that would solve this binary classification issue. Other than providing ML algorithms it also comes with handy tools to pre-process the dataset. Besides, it could also solves unsupervised learning issues with clustering and provides fine tuning features such as dimensionality reduction to select the most related features, helps researcher to select the most suitable model with parameters tuning.

Since there is no null values in the dataset, hence there’s no need to do imputation. However, the continuous values range is sparse hence we should normalise the data to an appropriate range. We should normalize the training and testing dataset separately to avoid data leakage. After that, we should convert the label data from a nominal value to numerical value (0 or 1) to represent the binary classes.

From the ML package, we would like to choose a Naïve Bayes method, GaussianNB and an ensemble learning method, RandomForestClassifier. Naïve Bayes classifier is a type of probabilistic classifiers which predict the results with the highest confidence score and assuming all the attributes are conditionally independent while RandomForestClassifier is taking the highest vote generated by decision trees on different samples of training set as the final output. Both of the classifiers is good at solving classification issues with continuous variables. In addition, Gaussian Naïve Bayes classifier is working better on dataset that are distributed normally and it requires lower cost of time to return the results. However, the RandomForestClassifier is taking longer time as it would need to calculate all the outputs for each decision tree in order to get the majority vote.

We built a GaussianNB model with python that train the given dataset without (Fig.1) and with normalisation (Fig 2.). Both the trainings have achieved 100% of precision, accuracy and f1-score. Similarly, we built a RandomForestClassifier that achieved the same performance. Figure 3 shows the details of RFC model without normalisation while figure 4 shows the results with normalisation.

GaussianNB

W/O Normalisation

```
GNB = GaussianNB()
GNB.fit(X_train.values, y_train.values.ravel())
GNB_preds = GNB.predict(X_test.values)
print(classification_report(y_test.values, GNB_preds, target_names=target_names))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
   virginica       1.00      1.00      1.00        10

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20
```

Figure 1. shows the model construction snippet of Gaussian NB model without normalisation and its performance metrics.

W/ Normalisation

```
GNB_pipe = Pipeline([
    ('scaler', StandardScaler()),
    ('classifier', GaussianNB())
])

GNB_pipe.fit(X_train.values, y_train.values.ravel())
GNB_pipe_preds = GNB_pipe.predict(X_test.values)
print(classification_report(y_test.values, GNB_pipe_preds, target_names=target_names))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
   virginica       1.00      1.00      1.00        10

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20
```

Figure 2. shows the model construction snippet of Gaussian NB model with normalisation and its performance metrics.

RandomForestClassifier

W/O Normalisation

```
RFC = RandomForestClassifier(max_depth=2, random_state=42)
RFC.fit(X_train.values, y_train.values.ravel())
RFC_preds = RFC.predict(X_test.values)
print(classification_report(y_test.values, RFC_preds, target_names=target_names))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
   virginica       1.00      1.00      1.00        10

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20
```

Figure 3. shows the model construction snippet for RandomForestClassifier model without normalisation and its performance metrics.

W/ Normalisation

```
RFC_pipe = Pipeline([
    ('scaler', StandardScaler()),
    ('classifier', RandomForestClassifier(max_depth=2, random_state=42))
])

RFC_pipe.fit(X_train.values, y_train.values.ravel())
RFC_pipe_preds = RFC_pipe.predict(X_test.values)
print(classification_report(y_test.values, RFC_pipe_preds, target_names=target_names))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
   virginica       1.00      1.00      1.00        10

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20
```

Figure 4. shows the model construction snippet for RandomForestClassifier model with normalisation and its performance metrics.

In summary, the results of the two models are very similar, it is because this is a particularly small and clean dataset with balance data. It has minimal noise hence the models are giving similar results for both before and after normalisation.