CT5165 Principles of Machine Learning: Assignment I

Jiarong Li 20230033

The School of Computer Science
University of Galway
j.li11@nuigalway.ie

1 Introduction

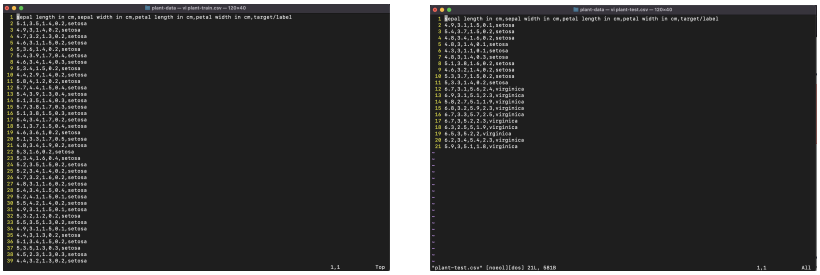
The goal of this assignment is to learn:

- the basics of understanding the dataset and use respective machine learning category,
- using a machine learning package for a given task,
- prepare a short report demonstrating how you have applied it.

The goal is to predict the dependent variable (which may be one of two classes: setosa or virginica) based on the other four independent attributes.

2 Data Processing.

2.1 Have a look at the given data



(a) Training data (b) Testing data

Figure 1: Training data and testing data.

Figure 1 shows the training and testing dataset we use in this task. Each row represents an instance, the first four columns respectively represent four independent variables (features) and

the last column represents the dependent variables (labels). In this case, there are two classes of dependent variables. Therefore, it is a classification problem and our objective is to predict the correct class for each given data.

We apply the supervised learning category to solve this problem. Classification is one of the major tasks in supervised learning. In addition, the training data we have are labelled data with a set of attributes that have known values. We can find a hypothesis that approximates the ground truth to make predictions with supervised learning techniques.

2.2 Training and testing data distribution

```
[10]: import pandas as pd
      import collections

[13]: train_data = pd.read_csv("plant-data/plant-train.csv")

      train_data = list(train_data["target/label"])

      counter_train_data = collections.Counter(train_data)

      counter_train_data

[13]: Counter({'setosa': 40, 'virginica': 40})

[14]: test_data = pd.read_csv("plant-data/plant-test.csv")

      test_data = list(test_data["target/label"])

      counter_test_data = collections.Counter(test_data)

      counter_test_data

[14]: Counter({'setosa': 10, 'virginica': 10})
```

Figure 2: The outputs of the code show that in the training dataset, there are 40 setosa and 40 virginica. In the testing dataset, there are 10 setosa and 10 virginica.

Figure 2 shows that in the training dataset and the testing dataset, the data of two classes (setosa and virginica) show the same distribution. Therefore, the dataset is balanced.

2.3 Useful packages

There are many powerful open-source packages for machine learning, such as NumPy, Scipy, Pandas, Scikit-learn, TensorFlow, Keras and PyTorch. We apply the Scikit-learn package to deal with this binary classification task.

Scikit-learn is a python library built on NumPy, Matplotlib and Scipy. The main features of the Scikit-learn package are data splitting, linear regression, classification reports, decision trees, etc.

2.4 Data preparation

We identify data preparation by transforming the raw data into a form that is suitable for our model. There are several steps to preparing data.

(a). Data cleansing [1]. We identify and correct missing data or noise by using e.g. pruning techniques.

(b. Feature Selection [1]. We identify the most relevant features as the input data by using information-based algorithms such as the decision tree algorithm.

(c). Data Transforms [1]. We change the scale or distribution of variables by using the normalisation technique so that the large income values don't overwhelm the smaller values.

(d). Feature Engineering. We can derive new variables from available data [1]. For example, image augmentation techniques in image processing tasks.

(e). Dimensionality Reduction. We create compact projections of the data [1].

2.5 Algorithms for our classification task.

I use the decision tree algorithm and naive Bayes algorithm to make the classification model respectively.

- The decision tree algorithm is an information-based algorithm used in classification tasks, etc. It predicts which category or class an item belongs to by choosing the features that can best partition the decision branches. A decision tree can be designed by calculating the importance of each feature by leveraging information gain or Gini gain techniques, etc.
- The naive Bayes algorithm has good performance on classification tasks. It is a probabilistic model that calculates the probability of the presence of each feature with the assumption that each feature is independent of other features.

The data normalisation in our case does not influence the accuracy results of the two algorithms we use. Because decision trees are non-parametric and the data normalisation does not change the ranks of the data. Therefore, the results are the same.

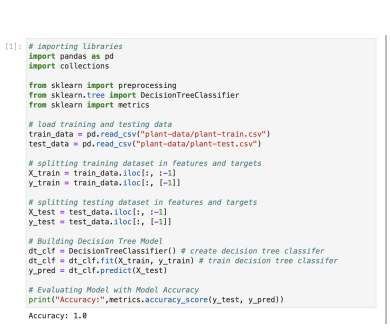
2.6 Train and test.

We choose two models (model with decision tree algorithm and model with naive Bayes algorithm) for this classification task with the Sklearn library of python.

We first pre-process the training dataset and testing dataset by leveraging the Pandas library of python. We also split the features and corresponding labels in training and testing datasets.

Our classifiers are created with the Sklearn library. Then we train the classifier with the split training dataset (independent variables as inputs and dependent variables as the ground truth). We obtain a trained classifier currently. In order to know how good our classifier is we test it by inputting the test dataset (split features as input) and comparing the predicted results with the ground truth labels of the test dataset. Finally, we calculate the prediction accuracy by using the metrics class in the Sklearn library and print out the prediction accuracy.

Based on our basic model structure, we use two classifiers (decision tree classifier and naive Bayes classifier) to build our classification model. Both of them show good performance on our training and testing data as the prediction accuracy is 100% in both models, as Figure 3 shows below:



```
[1]: # importing libraries
import pandas as pd
import collections

from sklearn import preprocessing
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics

# load training and testing data
train_data = pd.read_csv("plant-data/plant-train.csv")
test_data = pd.read_csv("plant-data/plant-test.csv")

# splitting training dataset in features and targets
X_train = train_data.iloc[:, :-1]
y_train = train_data.iloc[:, [-1]]

# splitting testing dataset in features and targets
X_test = test_data.iloc[:, :-1]
y_test = test_data.iloc[:, [-1]]

# Building Decision Tree Model
dt_clf = DecisionTreeClassifier() # create decision tree classifer
dt_clf = dt_clf.fit(X_train, y_train) # train decision tree classifer
y_pred = dt_clf.predict(X_test)

# Evaluating Model with Model Accuracy
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

Accuracy: 1.0
```

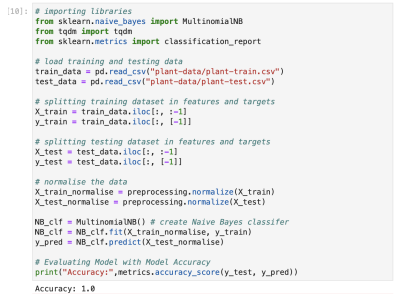
(a) Decision tree algorithm without data normalisation

```
[5]: # importing libraries
import pandas as pd
import collections

from sklearn import preprocessing
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics

# load training and testing data
train_data = pd.read_csv("plant-data/plant-train.csv")
test_data = pd.read_csv("plant-data/plant-test.csv")

# splitting training dataset in features and targets
X_train = train_data.iloc[:, :-1]
y_train = train_data.iloc[:, [-1]]

# splitting testing dataset in features and targets
X_test = test_data.iloc[:, :-1]
y_test = test_data.iloc[:, [-1]]

# normalise the data
X_train_normalise = preprocessing.normalize(X_train)
X_test_normalise = preprocessing.normalize(X_test)

dt_clf = DecisionTreeClassifier() # create decision tree classifer
dt_clf = dt_clf.fit(X_train_normalise, y_train) # train decision tree classif
y_pred = dt_clf.predict(X_test_normalise)

# Evaluating Model with Model Accuracy
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

Accuracy: 1.0
```

(b) Decision tree algorithm with data normalisation

```
[9]: # importing libraries
from sklearn.naive_bayes import MultinomialNB
from tqdm import tqdm
from sklearn.metrics import classification_report

# load training and testing data
train_data = pd.read_csv("plant-data/plant-train.csv")
test_data = pd.read_csv("plant-data/plant-test.csv")

# splitting training dataset in features and targets
X_train = train_data.iloc[:, :-1]
y_train = train_data.iloc[:, [-1]]

# splitting testing dataset in features and targets
X_test = test_data.iloc[:, :-1]
y_test = test_data.iloc[:, [-1]]

NB_clf = MultinomialNB() # create Naive Bayes classifer
NB_clf = NB_clf.fit(X_train, y_train)
y_pred = NB_clf.predict(X_test)

# Evaluating Model with Model Accuracy
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

Accuracy: 1.0
```

(c) Naive Bayes algorithm without data normalisation



```
[10]: # importing libraries
from sklearn.naive_bayes import MultinomialNB
from tqdm import tqdm
from sklearn.metrics import classification_report

# load training and testing data
train_data = pd.read_csv("plant-data/plant-train.csv")
test_data = pd.read_csv("plant-data/plant-test.csv")

# splitting training dataset in features and targets
X_train = train_data.iloc[:, :-1]
y_train = train_data.iloc[:, [-1]]

# splitting testing dataset in features and targets
X_test = test_data.iloc[:, :-1]
y_test = test_data.iloc[:, [-1]]

# normalise the data
X_train_normalise = preprocessing.normalize(X_train)
X_test_normalise = preprocessing.normalize(X_test)

NB_clf = MultinomialNB() # create Naive Bayes classifer
NB_clf = NB_clf.fit(X_train_normalise, y_train)
y_pred = NB_clf.predict(X_test_normalise)

# Evaluating Model with Model Accuracy
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

Accuracy: 1.0
```

(d) Decision tree algorithm with data normalisation

Figure 3: (a) shows the decision tree algorithm without normalising data, (b) shows the decision tree algorithm with normalising data, (c) shows the naive Bayes algorithm without normalising data, and (d) shows the naive Bayes algorithm with normalising data.

2.7 Results and reflection

The prediction of accuracy in our two models shows signification similar. One of the reasons is the size of the training dataset is 80 in total which is not sufficient enough for the training process. On the other hand, the number of features in our dataset is not large, therefore, it is respectively simple for the two models to identify the useful features.

References

[1] What is data preparation in a machine learning project. https://machinelearningmastery.com/what-is-data-preparation-in-machine-learning/. Accessed: 2020-07-17.