**Master of AI**

Faculty of Engineering & IT

42172 AT1 Mini AI Application Example

UTS CRICOS 00099F

**Link to solution notebook:**

https:// … (add your URL link)

**Part:**

**3**

**Author**:

Enda fan

25608579

**Date:** (

2024/10/12

**Semester:**

1

**Year:**

1

**University of Technology Sydney**

Table of Contents

[Table of Contents 2](#_Toc179667995)

[Summary or Executive Summary 3](#_Toc179667996)

[Introduction 4](#_Toc179667997)

[Report Body 4](#_Toc179667998)

[Methods/AI techniques used 4](#_Toc179667999)

[Implementation of AI techniques 5](#_Toc179668000)

[Comparisons of different AI techniques 11](#_Toc179668001)

[Conclusions and Recommendations 13](#_Toc179668002)

[List of References 14](#_Toc179668003)

[Individual reflection 15](#_Toc179668004)

## Summary or Executive Summary

In this project, our goal is to apply machine learning algorithms to classify the ‘Forest CoverType’ dataset, a multi-class classification problem in which samples from the dataset correspond to 30 x 30-metre forest patches in the United States, which were collected to predict the cover type, i.e., the dominant tree species, of each patch. i.e., the dominant tree species, of which there are seven cover types.

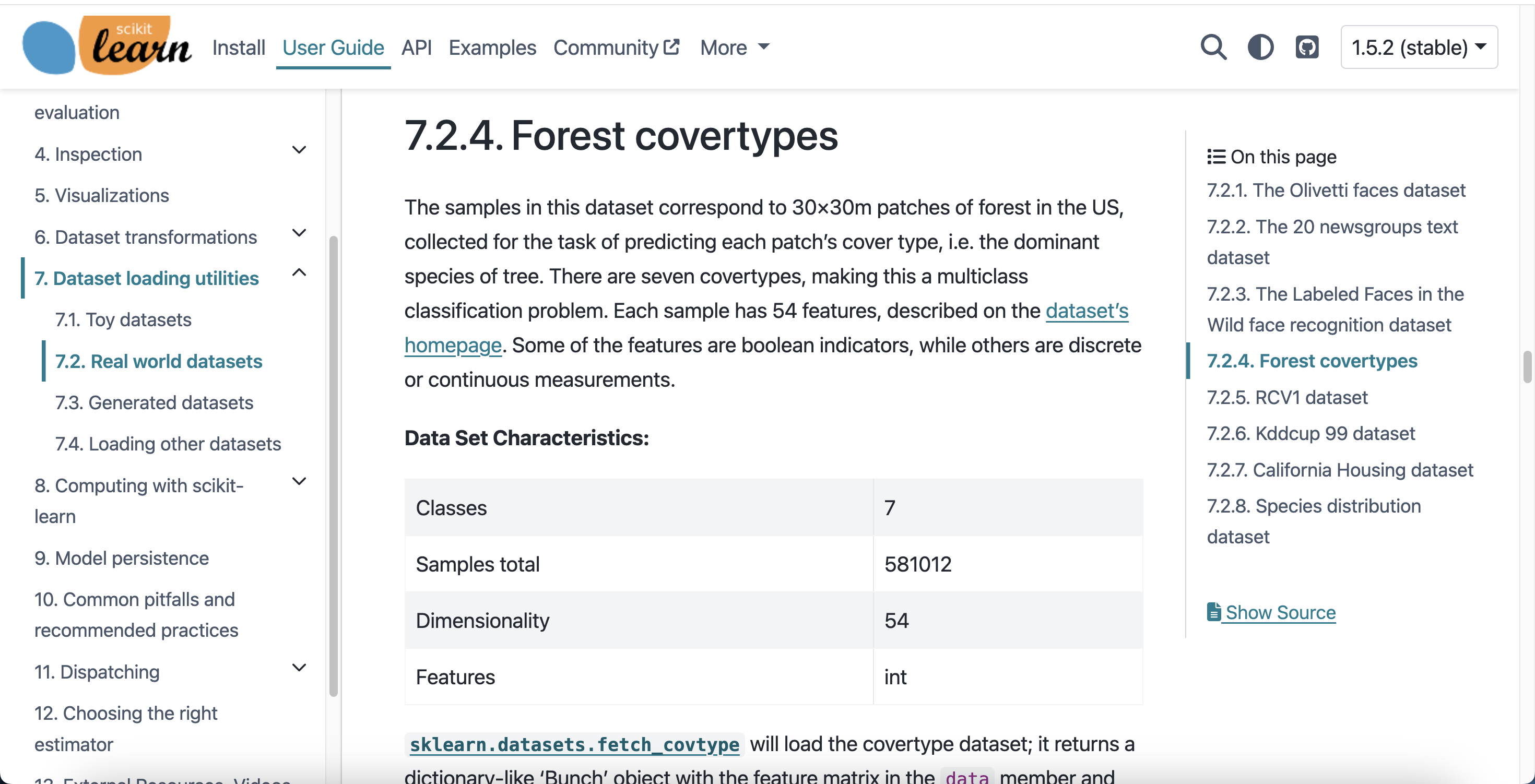
I used two AI techniques in this project: a Feedforward Neural Network (FNN) and a Random Forest Classifier. Feedforward Neural Network (FNN) is a simple neural network structure where data flows from the input layer to the hidden layer and finally to the output layer. It is suitable for processing structured data and can capture non-linear relationships between features. FNN can be used not only for classification but also for regression tasks. A feed-forward network has connections only in one direction—that is, it forms a directed acyclic graph. Every node receives input from “upstream” nodes and delivers output to “downstream” nodes; there are no loops. A feed-forward network represents a function of its current input; thus, it has no internal state other than the weights themselves (Russell, Stuart, & Peter Norvig, 2021). Random Forest is used to improve classification accuracy by constructing multiple decision trees and voting on the predictions of each tree.

Through the experimental results, my main finding is that Random Forest performs better in the classification task as it achieves a high accuracy and F1 score, while FNN also performs well. I think it may be because Random Forest improves the model's ability by combining the predictions from multiple decision trees. FNN also has advantages when dealing with more complex non-linear relationships, but in this project, Random Forest is more suitable for this type of dataset. Therefore, it can be shown that the Random Forest model performs more consistently when dealing with this type of task, or for this dataset.

I suggest that the model's accuracy can be improved in future work by further tuning the hyperparameters and trying other algorithms such as Support Vector Machine (SVM).

## Introduction

The main problem we address in this project is how to efficiently and accurately classify forest cover types.



*Figure 1 Forest CoverType Dataset*

The ‘Forest CoverType’ dataset contains forest samples from different regions of the United States, which are classified according to seven major tree species. Accurately classifying forest cover types is crucial for forest management and environmental protection, but manual classification methods are inefficient and error-prone when faced with large amounts of data. Therefore, this project provides a solution to this problem by applying machine learning techniques: Feedforward Neural Network (FNN) and Random Forest Classifier.

FNN is good at dealing with complex non-linear data relationships, while Random Forest improves classification accuracy by constructing multiple decision trees, which ultimately vote for the prediction result. Through these techniques, we not only improve classification accuracy but also reduce the complexity of manual intervention, allowing the model to more efficiently handle the task of classifying large-scale forest samples.

The need for this project lies in the reduction of human error through automation techniques and significantly improves efficiency while providing strong technical support for forestry and environmental management.

## Report Body

### Methods/AI techniques used

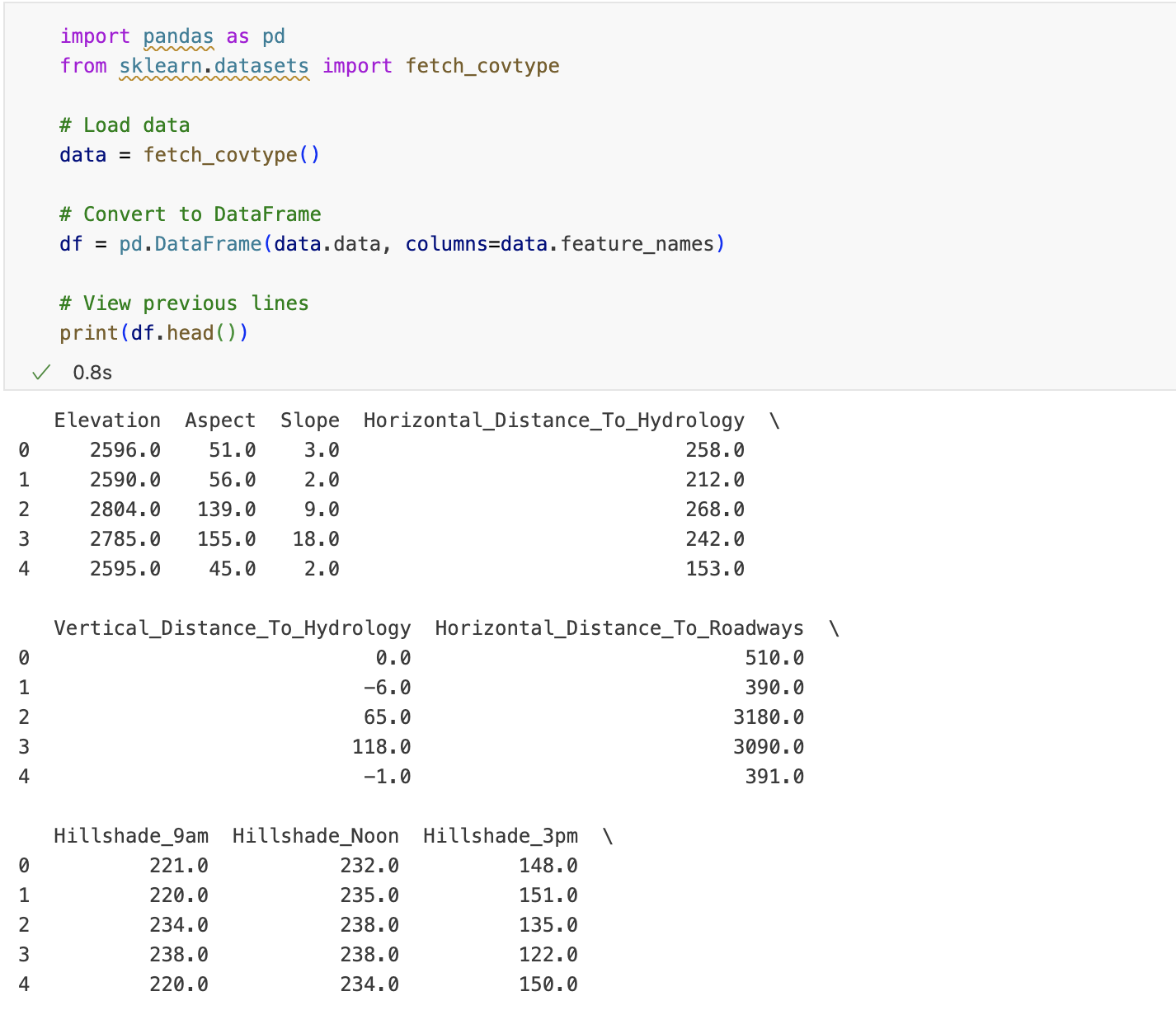
Before starting this task, I was going to use the LSTM algorithm to solve this task, but I found that LSTM performs best when faced with data of the type of sequences or time series, and I thought that I could try to treat the feature values as sequential inputs, but this would require a lot of tuning to ensure that it was effective. I found this to be too inefficient, so I abandoned this option in favour of the Random Forest algorithm.

I have chosen two AI techniques in this project: Feedforward Neural Network (FNN) and Random Forest Classifier. Feedforward Neural Network (FNN) and Random Forest are two commonly used AI algorithms. FNN is a neural network algorithm that processes input data layer by layer by connecting layer to layer and finally outputs a prediction result. This type of network has a simple structure and is suitable for processing structured data, especially in cases where complex relationships need to be captured. A feedforward neural network (FNN) is like an assembly line where each node passes data from one end to the other. Imagine you're making a product, with workers specialising in different parts at each step, and finally outputting the finished product. This is how FNN process the input data sequentially and gives the result at the output layer. Random Forest is an integrated learning algorithm that performs classification or regression by constructing multiple decision trees. Each tree independently makes predictions on the data and finally, the final result is obtained through a voting mechanism. Random Forest then acts like a committee of multiple experts. Each decision tree makes predictions independently and then votes to determine the result. This collective decision-making prevents individual decisions from going wrong and results are more stable and accurate.

Combining these two techniques provides a reliable solution to multi-categorisation problems in projects.

### Implementation of AI techniques

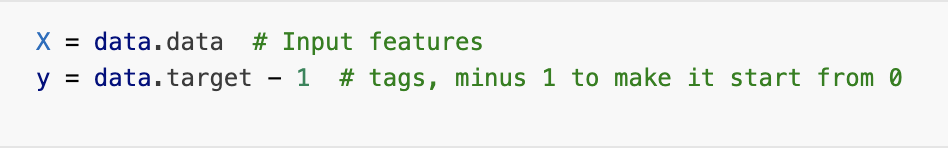
First, I used FNN algorithms. In the beginning, I needed to load the dataset.



*Figure 2 Forest CoverType Dataset*

Because I wanted to see some information about this dataset, so I use.head() of the pandas library to see it, but I found that this method doesn't work, I found that this is because the ‘Forest covertypes’ dataset returns a Bunch object, not a DataFrame. I need to convert the Bunch object to pandas before I can use the.head() method.

In the ‘Forest Covertype’ dataset, the label (i.e., the target of the prediction) should be the ‘Cover\_Type’ of the forest, indicating the different forest cover types. This attribute describes the type of forest area (e.g., spruce forest, redwood forest, etc.) and has 7 categories (1 to 7). You don't need to tag the attribute again because the fetch\_covtype() function already provides it as a target variable in data.target. Therefore, I can directly use data.target as the model's label.



*Figure 3 Label*

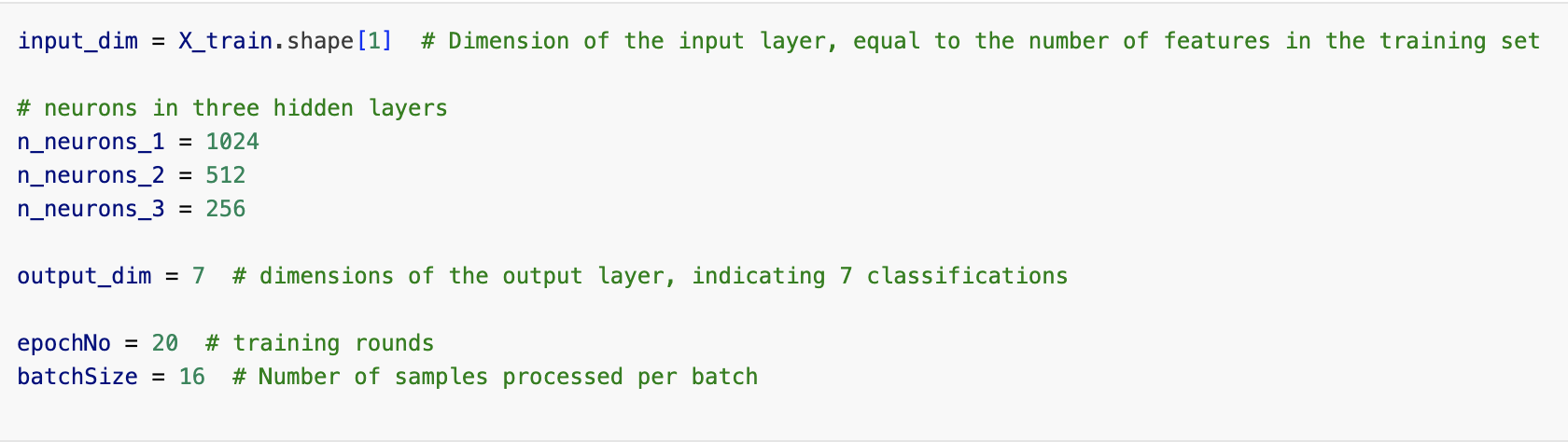
Use the train\_test\_split function to split the dataset into a training set and a test set, then take the last two samples from the test set and use them as a ‘future sample set’ to simulate the predictive performance of the model on new data in real-world applications. This simulates the model's predictive performance on new data in real-world applications. By removing these samples, it ensures that they are not involved in the training and evaluation of the model. This allows us to have a ‘future’ set of samples after training and testing to assess the generalisation of the model and see how it performs on samples it has never seen before, enabling us to better evaluate the model in practice.

图形用户界面, 文本, 应用程序

描述已自动生成

*Figure 4 Divide the dataset*

Next, I need to define the structure and training parameters of the feedforward neural network (FNN), including the dimensions of the input and output layers, the number of neurons in the hidden layer and the training parameters (epochNo and batchSize).



*Figure 5 Define the model*

Before building and training a neural network model, it is also necessary to compile the model, which serves to specify the loss function, optimiser and evaluation metrics for the model, a process that is equivalent to defining the training rules for the model.

文本

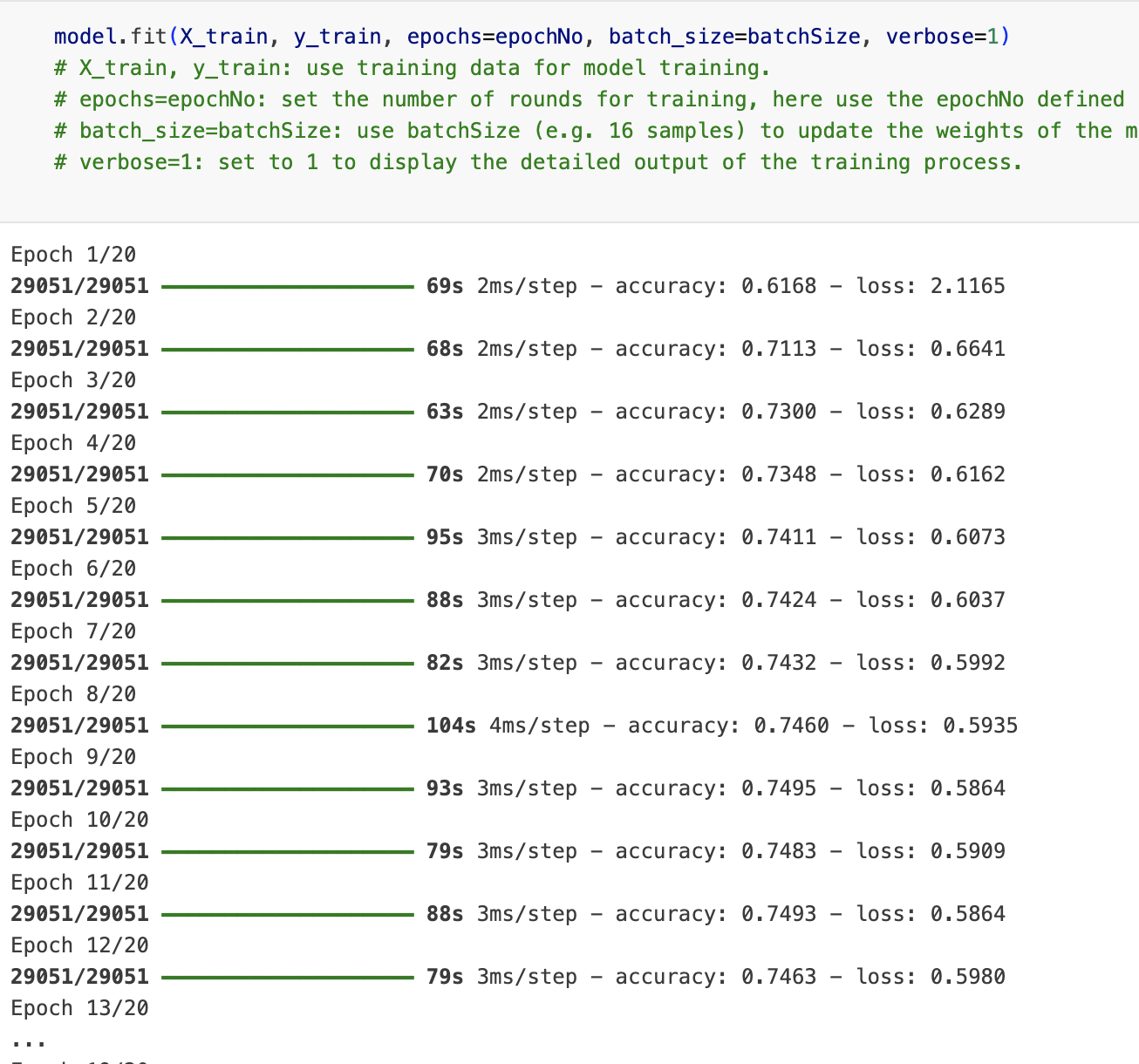
描述已自动生成

*Figure 6 Compile the model*

I then output y\_train and y\_test, the minimum and maximum values of the labels for the training and test sets, which helps me to confirm that the labels have been processed correctly, thus ensuring that the range of their values is as required by the model. As I can see through the output, the labels are between [0, 6], so I can confirm that the label values have been processed correctly and I can continue with the later steps.

*Figure 7 Output of y\_train and y\_test*

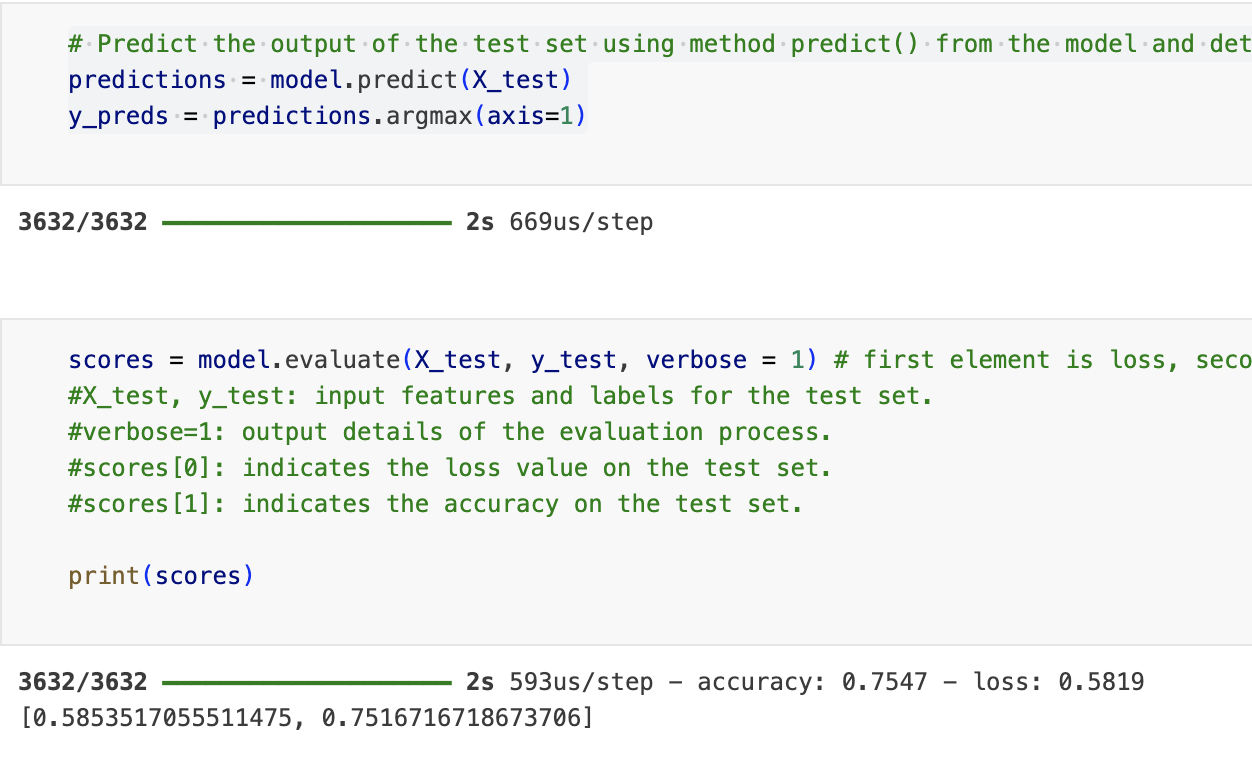
Next, we are ready to train the model.



*Figure 8 Train the model*

model.predict() is a method in TensorFlow for making predictions on new data. It will input a new set of data (e.g. test set X\_test) based on the already trained model and output the predicted probability of each input sample under each category. The final prediction for that sample is then made by selecting the category with the highest probability.

model.evaluate is a method in Keras that evaluates the performance of a model on test data. It takes input features (x\_test) and labels (y\_test) and returns the model's loss value on that dataset and a specified evaluation metric.



*Figure 9 Output of the prediction*

We then need to import the metrics module from the Scikit-learn library. This module contains a series of functions for model evaluation, and I can use various of them to analyse and evaluate the effectiveness of the model.

They are metrics.accuracy\_score, metrics.precision\_score, metrics.recall\_score, and metrics.f1\_score, which represent: 1. Accuracy: the proportion of samples correctly predicted by the model over all samples. The higher this value is, the better the overall accuracy of the model's predictions.2. Precision: indicates the proportion of samples correctly predicted by the model to be in a particular category as a proportion of samples predicted to be in that category. A high precision rate means that the model makes fewer errors in classification.3. Recall: indicates the proportion of samples correctly predicted by the model to be in a particular class as a proportion of the actual samples in that class. A high recall means that the model captures more samples of that class.4. F1-score: combines precision and recall and is used to balance the trade-off between precision and recall.



*Figure 10 Output of the metrics*

I used the confusion\_matrix from the metrics library to calculate and display the confusion matrix for the classification model, which allowed me to visualise the accuracy of the classification results.

图形用户界面, 应用程序

描述已自动生成

*Figure 11 Output of the matrix*

Use model.save(‘model.h5’) to save the trained model as a file for subsequent loading and reuse. ‘.h5’ is a file format that can be used to store some data of a neural network. In this way, the trained model can be loaded by load\_model(‘model.h5’) for prediction or further evaluation in the future without re-training the model. This is very helpful for models that need to be used for a long time or for tasks that require large-scale training, saving time and computational resources.

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

*Figure 12 model.h5*

As you can see from the results, the model successfully predicts the categories of the given future samples with the true labels (category 1 and category 6, respectively). This shows that your model can accurately classify this new data, illustrating the ability of the patterns learnt in the training and test sets to generalise to future data.

图形用户界面, 文本, 应用程序

描述已自动生成

*Figure 13 Results*

Secondly, I used the Random Forest algorithm because it impressed me, particularly in the last assignment and I think it performs quite well when faced with a large amount of data.

It is still relatively easy to implement this algorithm, the previous steps are similar to the FNN algorithm, you need to divide the training and test sets, and then you can use the RandomForestClassifier for the classification task. I specified the use of entropy as the classification criterion. Of course, other parameters can be adjusted as needed, such as the number of trees (n\_estimators), the maximum depth of the decision tree (max\_depth), the minimum number of samples in the leaf nodes (min\_samples\_leaf), and whether to use bootstrap samples (bootstrap). When training the model, the datasets X\_train and y\_train are used to fit this random forest classifier.

图形用户界面, 文本, 应用程序

描述已自动生成

*Figure 14 RandomForestClassifier*

Then it is also equally important to use the metrics module in sklearn to visualise its classification results.

图形用户界面, 文本, 应用程序

描述已自动生成

*Figure 15 Output of matrix*

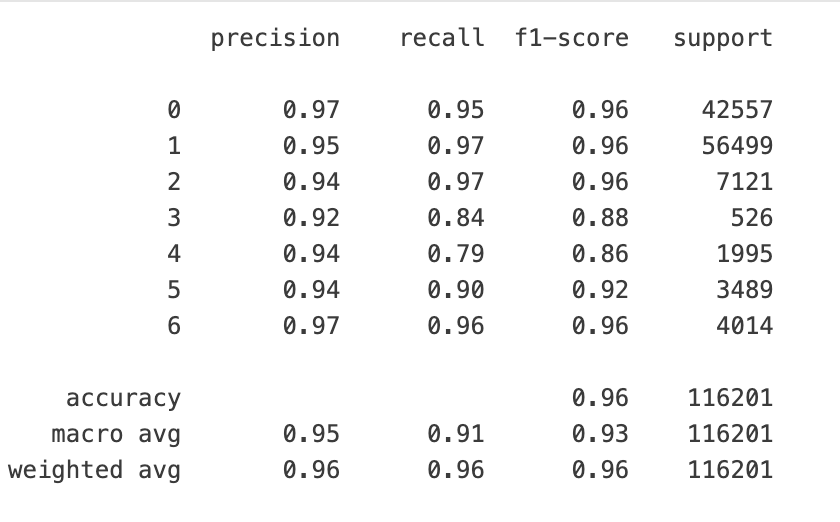
### Comparisons of different AI techniques

By comparing the matrices of the two algorithms, I believe that Random Forest performs better on most of the categories, especially categories 0, 1 and 2, which have high accuracy and recall. In contrast, the performance of FNN fluctuates more between categories, especially categories 3 and 4 which have significantly lower recall and f1-score. The confusion matrix of Random Forest between categories shows a more even distribution and exhibits greater ability to handle complex nonlinear relationships. Therefore, the overall performance of Random Forest is better than FNN in terms of confusion matrix and classification report.

表格

描述已自动生成

*Figure 16 Result of FNN*

**

*Figure 17 Result of RandomForest*

From the above two tables, we can see that:1. Accuracy: the accuracy of Random Forest is 0.96, while the accuracy of FNN is 0.75. It shows that Random Forest performs more accurately in categorisation.2. Precision, Recall and F1-score: Random Forest has a higher precision, recall and F1-score in all categories than FNN, especially FNN performs worse in category 3 and category 4 respectively (0.19 for category 3 and 0.25 for category 4), while FNN performs worse in category 3 and category 4 respectively (0.25 for category 4). higher than FNN, especially FNN performs worse in category 3 and category 4 with lower recall (0.19 for category 3 and 0.25 for category 4), respectively, while Random Forest's performance is more stable.3. Macro-mean and Weighted Mean: Random Forest also outperforms FNN in macro-mean and weighted mean scores, indicating that it maintains a better classification performance. Therefore, in terms of performance comparison, RandomForest significantly outperforms FNN.

## Conclusions and Recommendations

Overall, we can draw the main findings from our experiments and present some insights and recommendations based on these findings. For this project, the main finding is that Random Forest significantly outperforms FNN in terms of performance, with more consistent performance in terms of accuracy, precision, recall, and F1 score. Random Forest effectively solves the multi-classification problem by constructing multiple decision trees with a voting mechanism, while FNN, despite its good performance, is not as good as Random Forest in classifying certain categories. Therefore, based on the results, we recommend giving priority to using Random Forest in similar tasks and further optimising the model parameters or trying other classification algorithms in future work.

Therefore, I believe that Random Forests may be more efficient than FNNs in the following situations: 1. when the amount of data is small and the number of features is large: because Random Forests can handle high-dimensional data very well and do not require a large number of samples for training. In contrast, FNN usually requires a large amount of data for training to achieve better results. This is because it is obvious to me that FNN will take longer when I train the data.2. Lower complexity between data features: Random Forest relies less on complex relationships between features, while FNN is better suited to deal with highly non-linear, complex feature relationships.3. Lower model tuning requirements: Random Forest's default parameters usually already provide good results, while FNN requires a lot of debugging of the network structure, activation function, etc., which will take a lot of time to have a good result.

## List of References

Russell, Stuart, and Peter Norvig. (2021). Artificial Intelligence: a Modern Approach, Global Edition, Pearson Education Ltd.

## Individual reflection

In completing this task, I went through a systematic thought process and made some key decisions. Firstly, I chose the appropriate AI algorithms (FNN and Random Forest) to solve the classification problem, although I was initially prepared to use FNN and RNN first, then I found that the RNN algorithm is suitable for temporally ordered datasets, so I switched to the Random Forest algorithm, and then I optimised the model through several experiments. The main obstacles encountered included model overfitting and parameter tuning. To address these issues, I improved the results by iteratively tuning the parameters and analysing the model's performance metrics. Technically I learnt how to choose the right algorithm for different scenarios and became more knowledgeable about the use of sklearn. I improved my patience and problem-solving skills in dealing with complex tasks. In the future, I would like to learn more about the use of the sklearn library to achieve proficiency.