comp3411 assignment 2

Abanob Tawfik
z5075490
z5075490@student.unsw.edu.au

April 2020

1 Decision trees

Consider the decision tree learning algorithm of Figure 7.7 and the data of Figure 7.1 from Poole & Mackworth [1], also presented below. Suppose, for this question, the stopping criterion is that all of the examples have the same classification. The tree of Figure 7.6 was built by selecting a feature that gives the maximum information gain. This question considers what happens when a different feature is selected.

Example	Author	Thread	Length	Where_read	User_action
e_1	known	new	long	home	skips
e_2	unknown	new	short	work	reads
e_3	unknown	followup	long	work	skips
e_4	known	followup	long	home	skips
e_5	known	new	short	home	reads
e_6	known	followup	long	work	skips
e_7	unknown	followup	short	work	skips
e_8	unknown	new	short	work	reads
e_9	known	followup	long	home	skips
e_{10}	known	new	long	work	skips
e_{11}	unknown	followup	short	home	skips
e_{12}	known	new	long	work	skips
e_{13}	known	followup	short	home	reads
e_{14}	known	new	short	work	reads
e_{15}	known	new	short	home	reads
e_{16}	known	followup	short	work	reads
e_{17}	known	new	short	home	reads
e_{18}	unknown	new	short	work	reads
e_{19}	unknown	new	long	work	?
e_{20}	unknown	followup	short	home	?

Figure 7.1: Examples of a user's preferences

Figure 7.6: Two decision trees
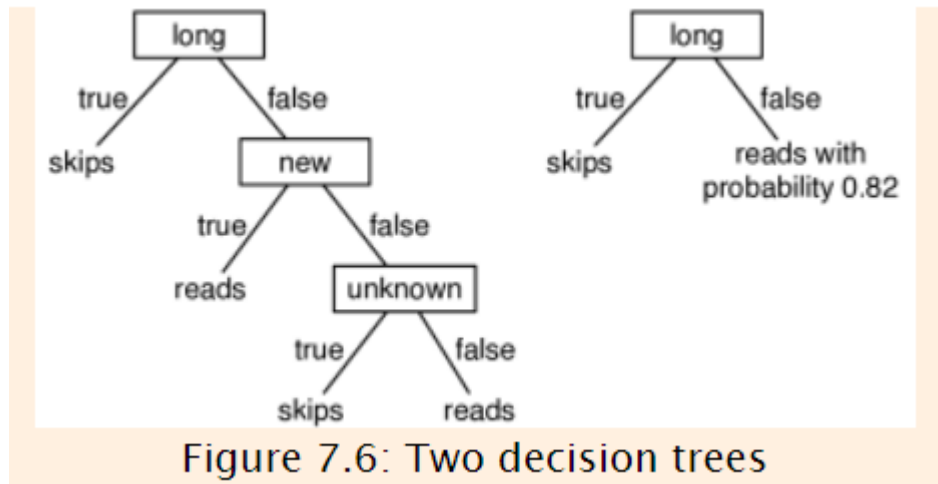
For the following Question the if then else statements representing the decision tree found with the maximum information gain split is given below.

```
if(e.Length == short){
    if(e.Thread == new){
        return "Read";
    }
    else{
        if(e.Author == known){
            return "Read";
        }
        else{
            return "Skip";
        }
    }
}
else{
    return "Skip";
}
```

(a) **Suppose you change the algorithm to always select the first element of the list of features. What tree is found when the features are in the order [*Author, Thread, Length, WhereRead*]? Does this tree represent a different function than that found with the maximum information gain split? Explain.**

Solution:

The following decision tree is made by selecting attributes in the order; Author, Thread, Length and WhereRead
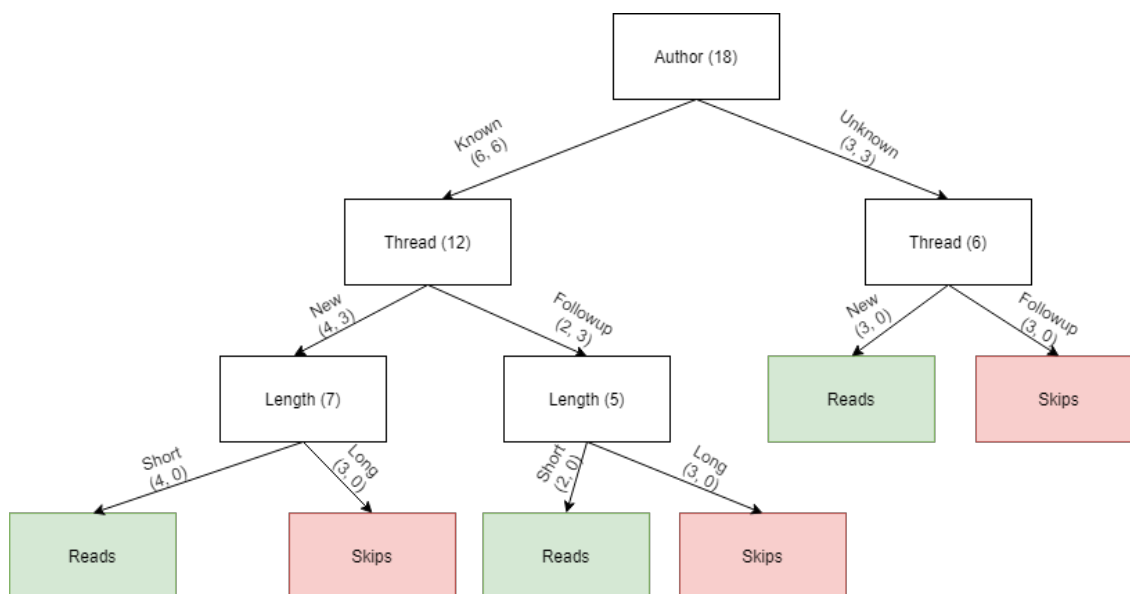
This decision tree represents a different function than the tree from Figure 7.6 with the maximum information gain splits. The following if then else statements on the next page represent the function of the tree.

```
if(e.author == known){
    if(e.Thread == new){
        if(e.Length == short){
            return "Read";
        }
        else{
            return "Skip";
        }
    }
    else{
        if(e.Length == short){
            return "Read";
        }
        else{
            return "Skip";
        }
    }
}
else{
    if(e.Thread == new){
        return "Read";
    }
    else{
        return "Skip";
    }
}
```

As can been seen from the following flow of the functions, when classifying example 19 [*unknown, new, long, reads*] the maximum information split tree gives us "Skip" as the output, as the first branch of the tree will give us "skip" since the example has length "long". With the decision tree we created example 19 gives us "read" as the output. when we follow the tree, since the author is unknown and the thread is new, the tree chooses "read". In the tree we created, the length of the article is irrelevant when the author is unknown, whereas in the maximum information split tree, length is the initial decider.

(b) **What tree is found when the features are in the order [*Wher-
eRead,Thread,Length, Author*]? Does this tree represent a differ-
ent function than that found with the maximum information gain
split or the one given for the preceding part? Explain.**
Solution:

The following decision tree is made by selecting attributes in the order;
WhereRead, Thread, Length and Author
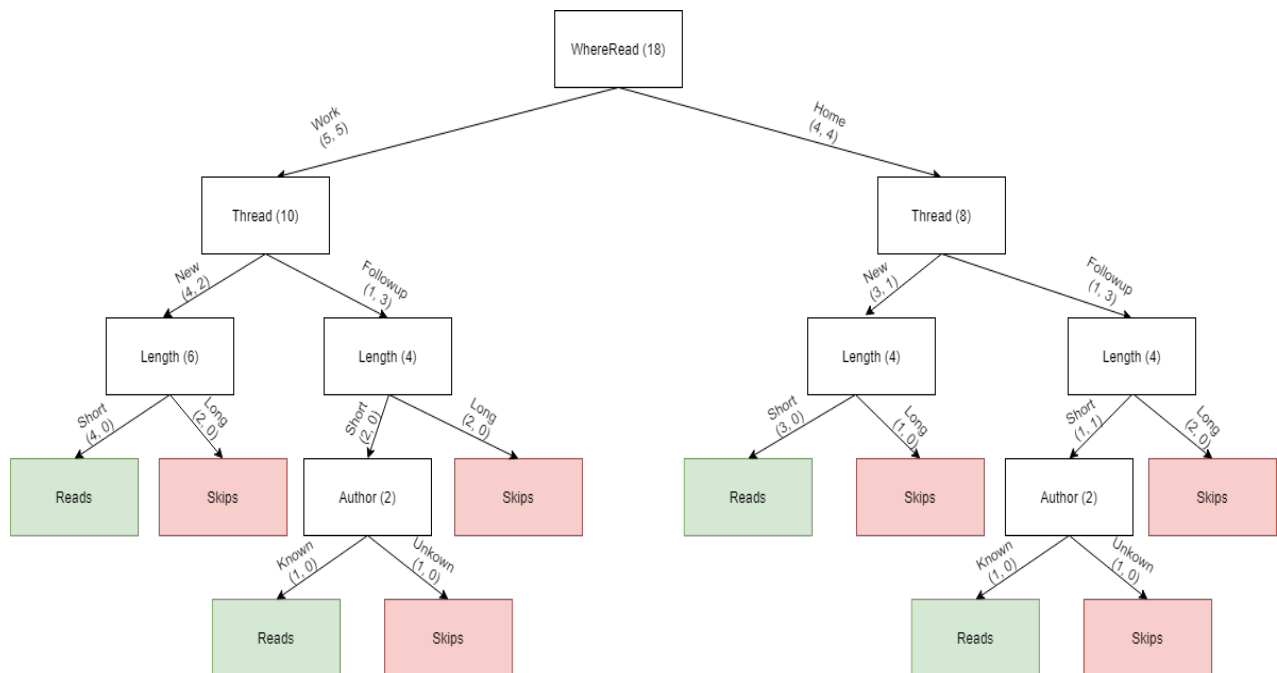
This decision tree represents the same function as the tree from Figure
7.6 with the maximum information gain splits. The following if then else
statements on the next page represent the function of the tree.

```
if(e.WhereRead == Work){
    if(e.Thread == New){
        if(e.Length == Short){
            return "Read";
        }
        else{
            return "Skip";
        }
    }
    else{
        if(e.Length == Short){
            if(e.Author == Known){
                return "Read";
            }
            else{
                return "Skip";
            }
        }
        else{
            return "Skip";
        }
    }
}
else{
    if(e.Thread == New){
        if(e.Length == Short){
            return "Read";
        }
        else{
            return "Skip";
        }
    }
    else{
        if(e.Length == Short){
            if(e.Author == Known){
                return "Read";
            }
            else{
                return "Skip";
            }
        }
        else{
            return "Skip";
        }
    }
}
```

The following tree represents the same function as the tree created with the maximum information gain split. Upon using both trees for classification on the test examples, they both had come to the same outcome, E18 was skip as on both trees, anytime length appeared as a node, the Long branch was always Skip. E19 had also turned into skip despite being short, as unknown followups had also ended up being leaves for skip. By comparing the features on both trees we can see that both trees will behave the same for all the same features.

Length: All Long articles skip, as each long transition has a leaf node for skip. All short articles are read provided either the thread is new, or the thread is a followup but the author is known. This can be seen also in the maximum information gain tree.

Thread: All follow up threads were only read provided they are short and the author is known. all new threads will be read provided the article is short. This can be seen also in the maximum information gain tree.

WhereRead: Regardless of the value of WhereRead, the output was not effected, this can be seen similair to above, where the children of WhereRead are identical, and the maximum information gain tree did not contain a node for WhereRead as it has no impact.

Author: all unknown authors are only read provided the article is short and new. All known authors are read provided the article is short. This can be seen in both trees by comparing the paths taken to the leaves.

By comparing the paths taken to reach the leaves, we can see that for each input, we will have the same output for both trees, the path on the maximum information gain tree is just shorter. Since all input produce the same output for both functions, we can say both trees represent the same function. Despite the two trees representing the same function, The maximum information gain tree is far more efficient as there is alot less branches and it is much more compact. In practice whilst both produce the same output, the tree from the maximum information gain is more desirable due to its compactness and speed in practice.

As discussed in Part (a) we learnt that the maximum information gain tree had represented a different function than that of the tree in Part (a). From this we can say that since the tree we have found represents the same function as the one from the maximum information gain tree, and the maximum information gain tree represents a different function than the tree in Part (a), we can say that the tree we have found represents a different function than the tree in Part (a).

(c) **Is there a tree that correctly classifies the training examples but represents a different function than those found by the preceding algorithms? If so, give it. If not, explain why.**
Solution:

as discussed in Part (b) the attribute WhereRead has no impact on the function of the tree. This means that the only attribute ordering which creates new trees are for the attributes [*Author, Thread, Length*]. We have already discussed the tree for picking attributes in the order [*Author, Thread, Length*], [*Length, Thread, Author*] and [*Thread, Length Author*] if we ignore the WhereRead attribute, This means there are only three other possible ways we can choose our attribute order.

(1) [*Length, Author, Thread*]

(2) [*Author, Length, Thread*]

(3) [*Thread, Author, Length*]

The tree from picking attributes in order (1) is the following tree below
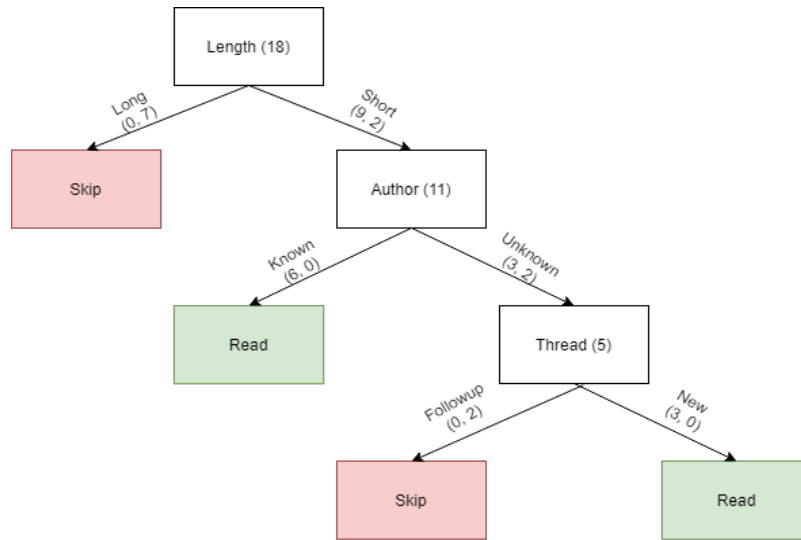
What can be observed from the tree, is that it is the exact same tree as the maximum information gain tree By comparing the impact on each feature we see that for the tree generated, all long articles are skipped and all short articles are read unless the author is unknown and the thread is a followup, This means that tree (1) represents the same function as the maximum information gain tree and the tree in Part (b).

The tree from picking attributes in order (2) is the following tree below
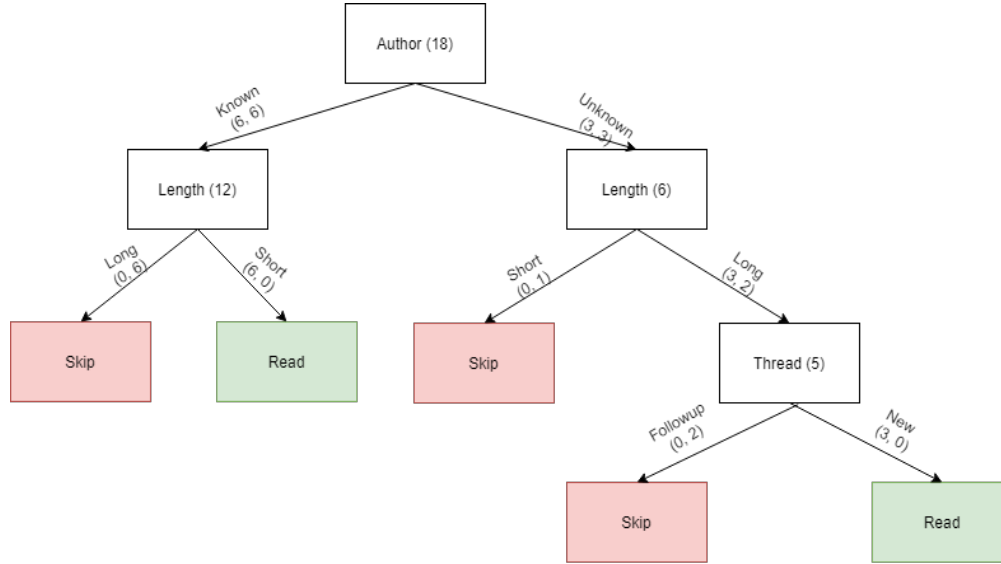
What can be observed from the tree, is that the tree represents a different function to the maximum information gain tree. What can be seen is that, short articles from unknown authors are skipped, regardless if its new or a followup, however in the maximum information gain tree, they are read. This means that the tree represents a different function than the maximum informaiton gain tree. This is also the case for the tree in part (a) as the tree in part (a) will not skip short articles from unknown authors that are new.

The tree from picking attributes in order (3) is the following tree below
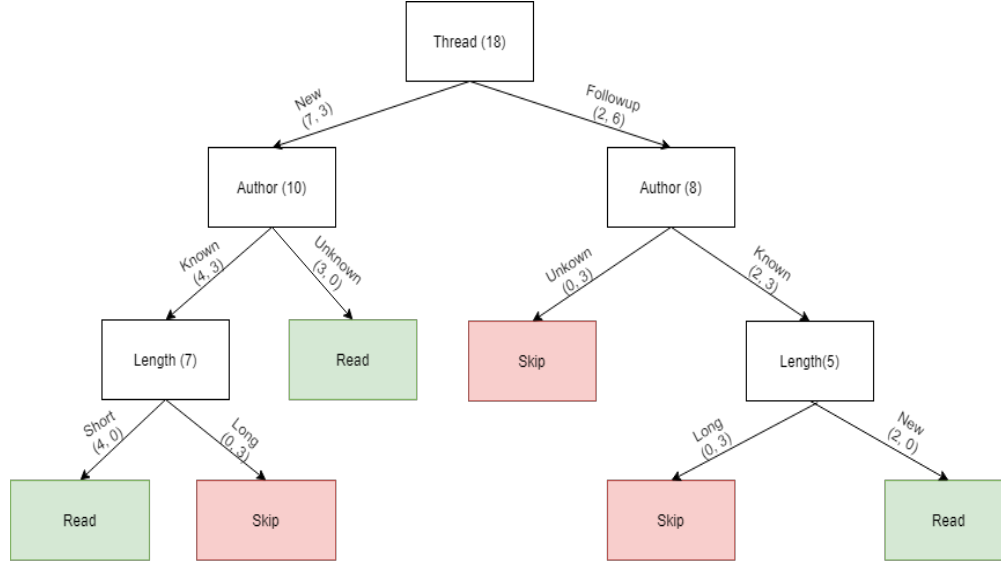
The following tree represents a different function to the maximum information gain tree, when comparing the impact on the features, any new thread from a unknown author will be read, even if it's length is long, whereas in the maximum information gain tree we skip all long articles. However what can be observed is that this tree represents the same function as the tree in Part (a). The tree itself is almost identical to the tree in Part (a) with Author and Thread swapped around, however what we can observe is that all feature input will have the same behaviour for both trees.

Length: Skip all long articles except for articles from unknown authors in a new thread. Read all short articles from known authors, and length has no impact on unknown authors. This is the same behaviour as seen from the tree in Part (a).

Thread: Read all new articles from any author provided they are not long, if the author is unknown, and the thread is new we read it otherwise we skip the article. if the author is known then Thread has no impact, as for known authors the deciding factor is the length of the article. This behaviour can be seen from the tree in Part (a) by comparing the children of known and unknown authors, and what can be observed is identical behaviour.

Author: For unknown authors, the deciding factor relied on only the thread value, for known authors the length of the article became the determining factor. This can be seen from the tree in Part (a) by comparing the children

10

of known and unknown authors, drawing the same conclusions.

As can be seen Tree (3) represents the same function as the tree discovered in Part (a), in fact they both carry the same size and will have almost the same efficiency depending on the data received.

In conclusion Since Tree (2) had represented a different function than the tree in part (a) and the maximum information gain tree, there exists a tree that correctly classifies all the examples, but represents a different function.

2   Decision trees

The goal is to take out-of-the-box models and apply them to a given dataset. The task is to analyse the data and build a model to predict whether income exceeds $50K/yr based on census data (also known as "Census Income" dataset). Use the data set Adult Data Set from the Machine Learning repository [2]. Use the supervised learning methods discussed in the lectures, Decision Trees.

Do not code these methods: instead use the implementations from scikit-learn. Read the scikit-learn documentation on Decision Trees [3], and the linked pages describing the parameters of the methods.

This question will help you master the workflow of model building. For example, you'll get to practice how to use the critical steps:

- Importing data

- Cleaning data

- Splitting it into train/test or cross-validation sets

- Pre-processing

- Transformation

- Feature engineering

Use the sklearn documentation pages for instructions. You should need the classification algorithms.

There are also available Tutorials:

- Sklearn – official tutorial for the sklearn package

- Predicting wine quality with scikit-learn – Step-by-step tutorial for training a machine learning model

The data is available here: http://archive.ics.uci.edu/ml/machine-learning-databases/adult/

From completing this question there were many things which i had learnt to building a model. from each critical step here is what i had done.

- **Importing data**
  Using the pandas library, i had imported all the data from the files into an array of objects containing one entry for each line in the data file.

- **Cleaning data**
  Some entries contained missing fields with the '?' as their value. These entries are unable to be analysed and were removed. To do this, I had first converted all the '?' values into NaN using numpy, and then i had used the pandas dropna function to remove all entries with any field containing NaN.

- **Splitting into training/test data sets**
  This was fairly simple as there were already separate files that contained a training set and a test set. I had functionised the data retrieval process and ran it on both separate files. In the case where we only had one data set, sklearn offers a test training split module which will take a percentage section for training and a percentage section for testing and provide two different data sets from one. This was however not necessary since we had both sets separately provided to us.

- **Pre-processing**
  Some attributes were strings, and this can be problematic for creating a decision tree of large scale, and the scikit library requires all input to be enumerated int32 values. To process the data to be used in an efficient manner, i had used the label encoder provided by scikit, which enumerates the string values into int32 values for usage in their classifier.

- **Transformation**
  After cleaning and pre-processing our data, our data was in a state where it could be used in an efficient manner. By removing all garbage entries and simplifying non-integer attributes, we had an efficient model which could be used inside our decision tree.

- **Feature Engineering**
  The methods chosen for selecting features is often the make or break for a machine learning project. The two methods for selecting features offered by scikit were "gini" and "entropy". I had chosen to use entropy as the lectures had discussed entropy and i was familiar with the concept. by using entropy maximum information gain to pick attributes, we had chosen the attribute with the lowest entropy value, giving us more meaningful splits. The difference between "gini" and "entropy" were very minor when it came to the accuracy of the classifier. When selecting a maximum depth for the tree in the classifier, I had observed that depths greater than

6 produced almost no increased accuracy, and depths of around 30 had lower accuracy, this can be attributed to Occham's Razor and overfitting. In order to pick a more generalised and readable tree, I had chosen my maximum feature depth as 6.

Below will show the code used in the creation of this project.

```python
from sklearn import datasets
import numpy
import pandas
from sklearn import tree
from sklearn import preprocessing
from sklearn.metrics import accuracy_score
import graphviz
from csv import reader
import os
os.environ["PATH"] += os.pathsep + r'D:/release/bin'


def process_data(filepath):
    # pull data from the adult.data file
    entries = pandas.read_csv(filepath)
    # here we will replace all ? fields with NaN for removal of row
    entries.replace(r'\?', numpy.nan, regex=True, inplace=True)
    print("before filtering list " + str(len(entries)) + " elements")
    # now we will drop any row with empty fields, as missing attributes
    # causes issues when creating our decision tree (cleaning data)
    entries.dropna(axis=0, how='any', inplace=True)
    print("after filtering list " + str(len(entries)) + " elements")

    target = entries.iloc[:, [14]]
    features = entries.iloc[:, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]]

    # pre processing data to be encoded
    label_encoder = preprocessing.LabelEncoder()
```

```
29          # enumerates the possible values for non numeric attributes on both target and features
30          target = label_encoder.fit_transform(target.values.ravel())
31
32          pandas.options.mode.chained_assignment = None  # default='warn'
33          for i in range(14):
34              features.iloc[:, i] = label_encoder.fit_transform(
35                  features.iloc[:, i].values.ravel())
36          pandas.options.mode.chained_assignment = 'warn'  # default='warn'
37          return features, target
38
39
40      training_features, training_target = process_data('Downloads/adult.data')
41      clf = tree.DecisionTreeClassifier(criterion="entropy", max_depth=6)
42      clf.fit(training_features, training_target)
43
44      feature_title = ["age", "workclass", "fnlwgt", "education", "education-num",
45                       "marital-status", "occupation", "relationship", "race", "sex",
46                       "capital-gain", "capital-loss", "hours-per-week", "native-country"]
47
48      dot_data = tree.export_graphviz(
49          clf, out_file=None, feature_names=feature_title, class_names=None, filled=True, rounded=True,
50          special_characters=True)
51      graph = graphviz.Source(dot_data)
52      graph.render("iris")
53
54      testing_features, test_expected = process_data('Downloads/adult.test')
55
56      test_results = clf.predict(testing_features)
```

```
57      accuracy = accuracy_score(test_results, test_expected)
58      print("The accuracy of the tree was " + str(accuracy*100) + "%")
59
```

Output of the code shown below

```
before filtering list 32560 elements
after filtering list 30161 elements
before filtering list 16280 elements
after filtering list 15059 elements
The accuracy of the tree was 84.3150275582708%
```

The next page shows the decision tree that was created for the classifier using the methods described previously. If the image is too difficult to read, a link is provided below to the full decision tree pdf file, you can zoom in to take a more detailed view.
https://documentcloud.adobe.com/link/track?uri=urn%3Aaaid%3Ascds%3AUS%3A613d445e-e5bb-4fe5-a9d2-94f948ea9746