COMP3411 Artificial Intelligence

Assignment 2 - Machine Learning

Jennifer Xu z5207930

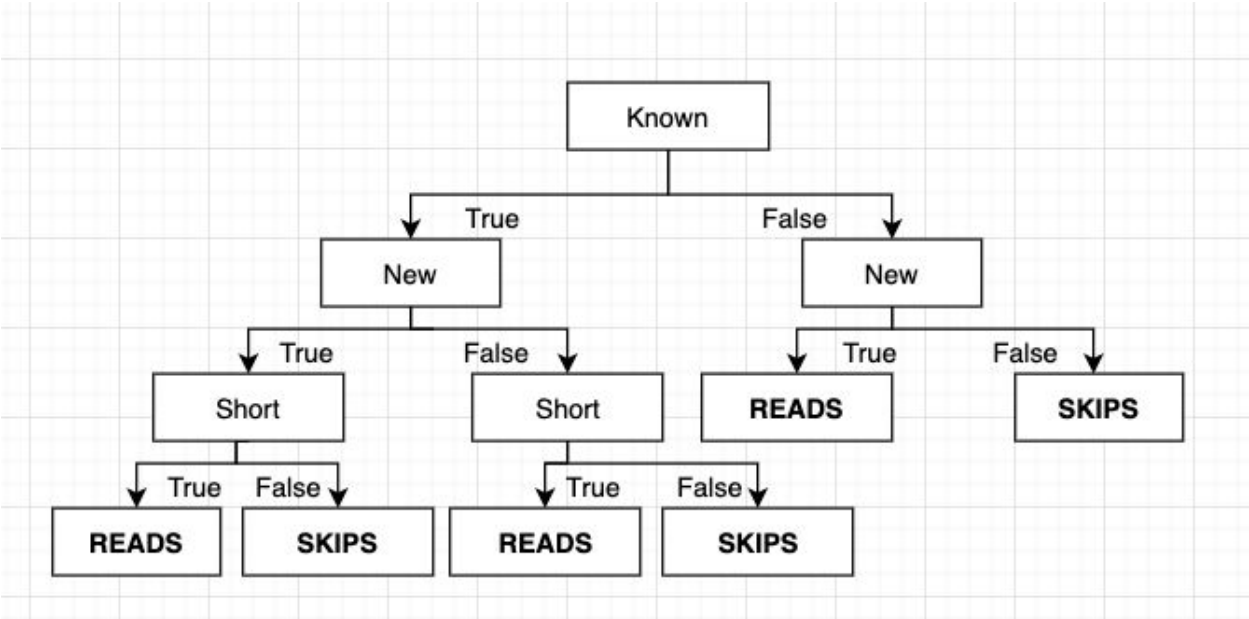

Figure 1. Decision tree in the order [*Author, Thread, Length, WhereRead]*

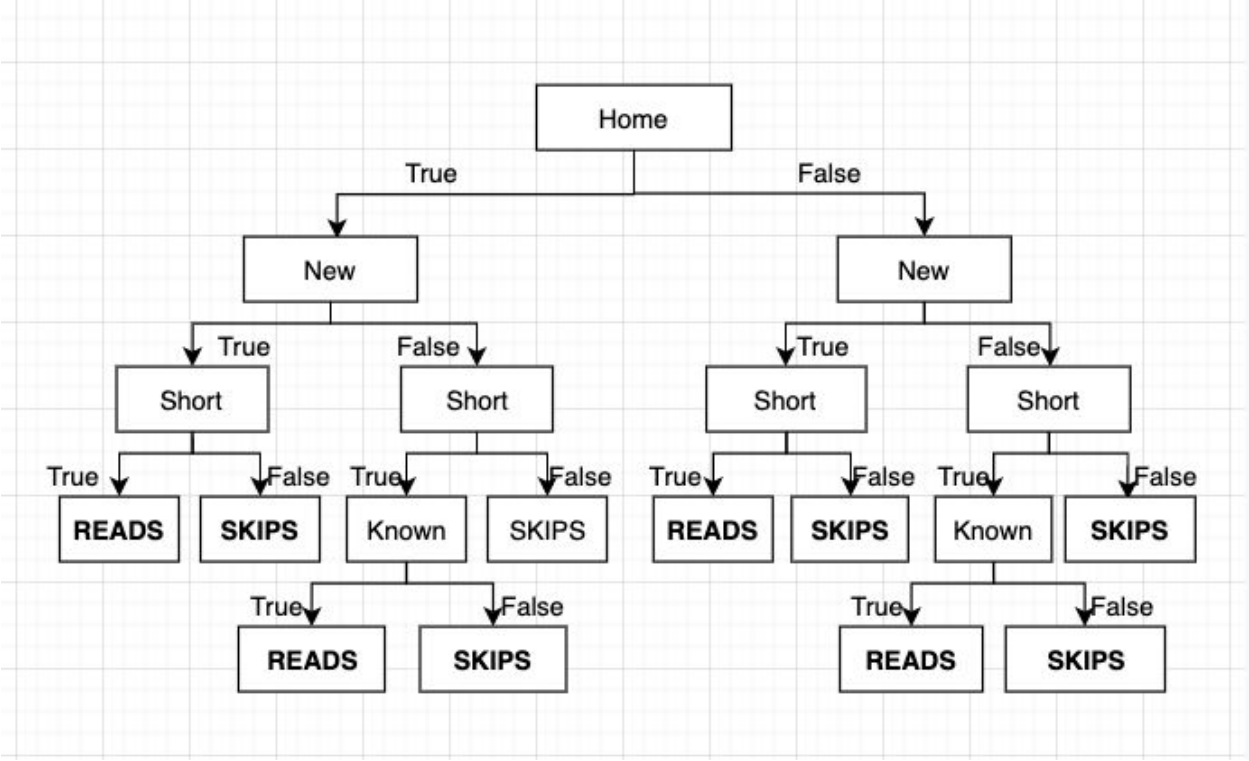

Figure 2. Decision tree in the order [*WhereRead, Thread, Length, Author]*

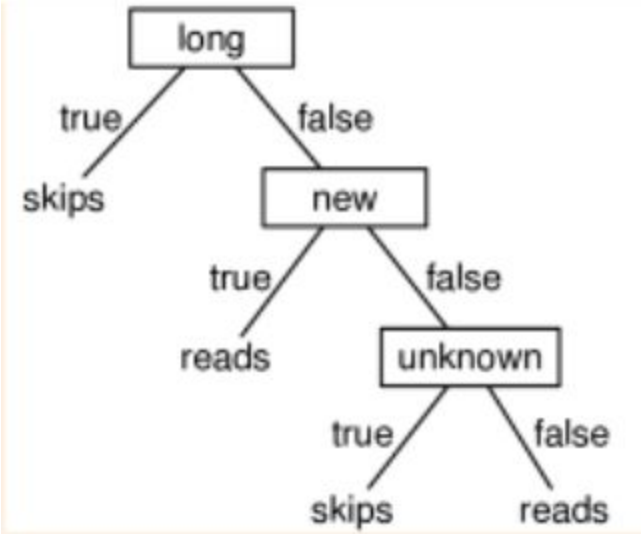

Figure 3. Decision Tree with maximum information gain split

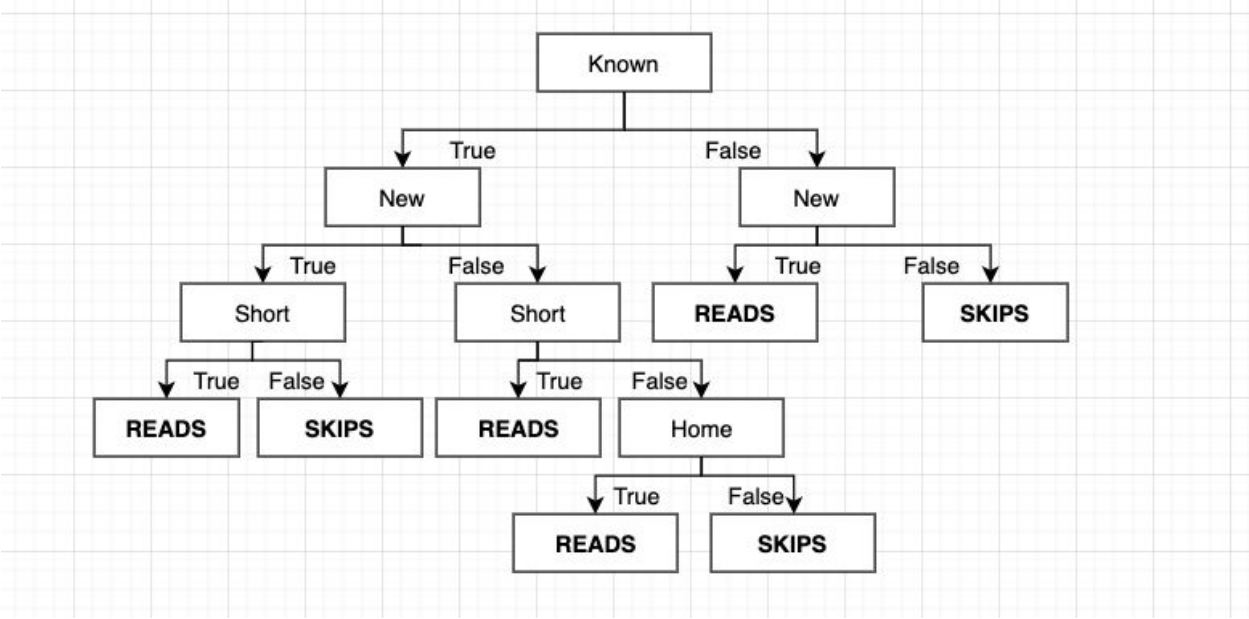

Figure 4. Decision Tree which is a different function.

Question 1

We can determine whether the two functions in Figure 1, 2 and 3 are the same by comparing the cases which have not been covered in the training data.

Case	Figure 1	Figure 2	Figure 3
1. [*Unknown, new, short, home]*	Reads	Reads	Reads
2. [*Unknown, new, long, home]*	Reads	Skips	Skips
3. [*Unknown, new, long, work]*	Reads	Skips	Skips
4. [*Unknown, followup, long, home*]	Skips	Skips	Skips

a) From the above table, we can see the decision tree in figure 1 does not have the same function as the decision tree in figure 3 - Case 3 and 4 give us different results.

b) From the above table we can see that all possible cases (those covered within the training data and those not covered) give us the same result. Therefore, the decision tree in figure 2 and 3 have the same function.

c) There exists a tree Is there a tree that correctly classifies the training examples but represents a different function than those found by the preceding algorithms. The difference occurs where we split home.
The cases:
   i) [Known, followup, long, home] -> Reads
   ii) [Known, followup, long, work] -> Skips
Gives us a different function to the previous algorithms.

Question 2

By using supervised learning, where we train an algorithm to pick an end model from a given set of training data to predict a pre-defined outcome, we created a model which achieves 85% accuracy when predicting whether the income of an individual is greater than 50K/year or not. [*Note: TARGET refers to the end result of which out model is trying to predict.]*

```
In [20]:
d = DecisionTreeClassifier(max_depth=10)
d.fit(df_train, TARGET)
d.score(df_test, TEST_TARGET)
```

```
Out[20]: 0.8557768924302789
```

This was done by using “adult.data” to train our model and “adult.test” to test the model we trained. However, we had to first clean our data by removing all rows which have an empty input and then stripping away extra spaces and lowering all the letters to ensure validity and correct sample data.

```
# Read the data, names is column name
df_train = pd.read_csv("../adult_og.data", header=None, names=names, na_values=["?", " ?"])
df_test = pd.read_csv("../adult.test", header=None, names=names, na_values=["?", " ?"])

# Drop all rows with ["?", " ?"]
df_train.dropna(inplace=True)
df_test.dropna(inplace=True)

# Lower all strings and strip extra spaces for our training data
df_str_cleaned = df_train.select_dtypes(include="object").applymap(lambda x: x.lower().strip())
df_str_cleaned = df_str_cleaned.select_dtypes(include="object").applymap(lambda x: x.strip("."))
df_train.update(df_str_cleaned)

# Lower all strings and strip extra spaces for our testing data
df_str_cleaned = df_test.select_dtypes(include="object").applymap(lambda x: x.lower().strip())
df_str_cleaned = df_str_cleaned.select_dtypes(include="object").applymap(lambda x: x.strip("."))
df_test.update(df_str_cleaned)
```

Then, we processed our data by one hot encoding *“workclass”, “martial-status”, “relationship”, “race”, “occupation”* and *“sex.”* This means that each possible answer for a category is turned into a category itself, creating more categories for the sake of simplifying our data into true and false responses.

```
# Put them together
df_together = pd.concat([df_train, df_test])
df_together.drop(labels=['id', 'education', 'salary'], inplace=True, axis=1)

# One hot code
features = ['workclass', 'status', 'native-country', 'relationship', 'race', 'occupation', 'sex']
for feature in features:
    df_onehot = pd.get_dummies(df_together[feature], prefix=feature, drop_first=True)
    df_together = pd.concat([df_together, df_onehot], axis=1)
    df_together.drop(labels=feature, inplace=True, axis=1)


# Split them apart
df_train = df_together[:len(df_train)]
df_test = df_together[len(df_train):]
```

We can visualise our model using a graph with the following code. [*Note: The max_depth was changed to 6 (still 85%) to have a smaller graph.*

```
# Visual Representaiton
data = tree.export_graphviz(d,
                            feature_names=df_together.columns,
                            class_names=["Less than 50K","Greater than 50K"],
                            filled=True, rounded=True,
                            special_characters=True,
                             out_file=None)
graph = graphviz.Source(data, format="png")
graph.render("graph")
```

[Graph is also attached.]