APA - Decision Trees

DATA 3300

Name: Chance Wiese

Assignment

Q1

Import the `funded.xlsx` dataset and the required libraries and packages, and view the headers of the df.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
from sklearn import tree #libraries and packages for training model and evaluating performance on test set

import seaborn as sns
import graphviz #libraries for producing visualizations
```

```
# replace with code to import funded.xlsx
df = pd.read_excel('/content/funded.xlsx')
# replace with code to produce df heading
df.head()
```

	id	industry	months	pfunding	patents	smember	elites	outcome
0	1	healthcare	36	2900000	0	0	5	soldoff-low
1	2	healthcare	31	5500000	0	0	0	ipo
2	3	defense	29	500000	0	0	4	bankrupt
3	4	tech	29	2400000	1	0	5	soldoff-high
4	5	healthcare	31	5300000	0	1	5	ipo

Use the `value_counts` function to view the class distribution for `Outcome`.

```
# replace with code to view class distribution of 'outcome' (hint: value counts)
df.value_counts('outcome')
```

```
outcome
soldoff-high    417
bankrupt        354
soldoff-low     260
ipo             178
dtype: int64
```

**Calculate the accuracy of an a priori (naive) prediction. Show your formula.

What would you predict the `Outcome` to be for a new observation and why?**

```
# replace with code to compute apriori prediction accuracy
apriori = 417/len(df)
print('apriori = ', apriori)
```

```
apriori =  0.34491315136476425
```

For a new outcome, I would predict that it's going to be soldoff-high, becase 34% of the time the outcome is soldoff-high, and it is the most common outcome.

Q2

Assign your IVs to an x object and your DV to a y object, then split your data into a training and test set for x and y, using a 80-20 split. Remember to perform any preprocessing necessary on your IVs (*hint: any categorical variables?)*

Then train a `DecisionTreeClassifier` model from the training set, using "gini" as the criterion, with a max depth of 10, a minimum leaf size of 8, and `min_impurity_decrease = 0.002`.

```
# replace with code to select IVs to include in analysis, assign to object called 'x'
x = df.drop(['id', 'outcome'], axis=1)
# replace with code to dummy code categorical variables
x = pd.get_dummies(data = x, drop_first=False)
# replace with code to assign your DV to an object called 'y'
y = df['outcome']
# replace with code to create 4 dataframe objects x_train, x_test, y_train, y_test. Set test_size to 0.2, set random_state to 10
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 100)
```

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(967, 9)
(242, 9)
(967,)
(242,)
```

```
# replace with code set parameters based on instructions for DecisionTreeClassifier
model = DecisionTreeClassifier(criterion = "gini", random_state=100, max_depth=10, min_samples_leaf=8, min_impurity_decrease=0.0
# replace with code to fit model to x_train and y_train
model.fit(x_train, y_train)
```

```
▾                         DecisionTreeClassifier
DecisionTreeClassifier(max_depth=10, min_impurity_decrease=0.002,
                       min_samples_leaf=8, random_state=100)
```

A

Plot your decision tree using the `export_graphviz` function from the `tree` library.

```
# replace with code to generate tree plot of model
labels = y.value_counts()
dot_data = tree.export_graphviz(model, out_file=None,
                                feature_names=x.columns,
                                class_names=labels.index.values,
                                filled=True)

# Draws graph
graph = graphviz.Source(dot_data, format="png")
graph
```

B

Using your tree plot, what would be the predicted outcome for a defense company who has no patents and 4 elites?

On how many observations would this prediction be based (i.e., how many observations are in the terminal node where the prediction was made)?

gini = 0.596 samples = 120 value = [67, 4, 17, 32] class = soldoff-high

We would predict this would be a soldoff-high, because 32 of the 120 samples that met this criteria had an outcome of soldoff-high, which was the most common outcome.

Q3

Apply your model to the test set to evaluate its accuracy. Create a confusion matrix and then respond to the prompts below.

```
# replace with code to predict onto test set using model
predictions = model.predict(x_test)
```

```
# replace with code to create new dataframe from x_test, called df_pred
df_pred = x_test
# replace with code to add predictions as a 'predicted_class' column to new dataframe
df_pred['predicted_class'] = predictions
# replace with code to add y_test as a 'actual_class' column to new dataframe
df_pred['actual_class'] = y_test
```

```
# replace with code to fit confusion matrix to 'predicted_class' and 'actual_class'
conf = pd.DataFrame(x_test, columns=['actual_class','predicted_class'])
confusion_matrix = pd.crosstab(conf['actual_class'], conf['predicted_class'], rownames=['Actual'], colnames=['Predicted'])
# replace with code to visualize confusion matrix with a heatmap
sns.heatmap(confusion_matrix, annot=True)
sns.set(rc={'figure.figsize':(12,10)})
plt.show
```

A

For each of the four outcomes, provide the number of times the model predicted it, the number of times it predicted accurately, and the % accuracy (ratio of the # of accurate predictions / # of predictions).

- Prediction: **Bankrupt**
  - Times Predicted: 114
  - Predicted Accurately: 52
  - % Accuracy: 46%
- Prediction: **Soldoff-Low**
  - Times Predicted: 31
  - Predicted Accurately: 17
  - % Accuracy: 55%
- Prediction: **Soldoff-High**
  - Times Predicted: 74
  - Predicted Accurately: 46
  - % Accuracy: 62%
- Prediction: **IPO**
  - Times Predicted: 23
  - Predicted Accurately: 17
  - % Accuracy: 74%

B

Considering *all* predictions, in what percentage of observations did the model make an accurate prediction? Show your formula.

```
# replace with code to calculate overall accuracy from values in confusion matrix
print(classification_report(y_test, predictions))
```

```
                 precision    recall  f1-score   support

        bankrupt       0.46      0.76      0.57        68
             ipo       0.74      0.47      0.58        36
```

```
soldoff-high       0.62      0.53      0.57        87
 soldoff-low       0.55      0.33      0.41        51

    accuracy                           0.55       242
   macro avg       0.59      0.52      0.53       242
weighted avg       0.58      0.55      0.54       242
```

C

Given this accuracy percentage, does it make more sense to use the *a priori* prediction approach or the decision tree model that was generated? Explain your answer.

It makes more sense to use a decision tree model as it was correct around 55% of the time, whereas the a priori method was only right around 34% of the time.

Q4

Next, compare this model's performance against three other possible decision tree models, each of which use different parameters. In a text cell placed before each new model, outline the `criterion` used (e.g., 'gini', 'entropy', or 'log_loss') , the `max_depth`, the `min_samples_leaf`, and the `min_impurity_decrease`.

Remember that the first model used the following paramters:

- `criterion` = gini
- `max_depth` = 10
- `min_samples_leaf` = 8
- `min_impurity_decrease` = 0.002

So change *at least* one of these parameters for each of your new models.

For each model generate a `classification_report` to show its overall accuracy on the test_set.

Model 1:

- `criterion` = gini
- `max_depth` = 10
- `min_samples_leaf` = 8
- `min_impurity_decrease` = 0.002

```
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

    bankrupt       0.46      0.76      0.57        68
         ipo       0.74      0.47      0.58        36
soldoff-high       0.62      0.53      0.57        87
 soldoff-low       0.55      0.33      0.41        51

    accuracy                           0.55       242
   macro avg       0.59      0.52      0.53       242
weighted avg       0.58      0.55      0.54       242
```

Model 2:

- `criterion` = entropy
- `max_depth` = 10
- `min_samples_leaf` = 8
- `min_impurity_decrease` = 0.002

```
# replace with code to specify DecisionTreeClassifier parameters
model2 = DecisionTreeClassifier(criterion = "entropy", random_state=100, max_depth=10, min_samples_leaf=8, min_impurity_decrease
# replace with code to fit model2 to training data
model2.fit(x_train, y_train)
```

```
▾                          DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=10,
                       min_impurity_decrease=0.002, min_samples_leaf=8,
                       random_state=100)
```

```
# replace with code to drop 'predicted_class' and 'actual_class' columns from x_test
x_test = x_test.drop(columns = ['predicted_class', 'actual_class'])
# replace with code to fit model2 to x_test
predictions = model2.predict(x_test)
```

```
# replace with code to print classification report
print(classification_report(y_test, predictions))
```

```
                  precision    recall  f1-score   support

        bankrupt       0.45      0.74      0.56        68
             ipo       0.56      0.50      0.53        36
    soldoff-high       0.65      0.46      0.54        87
     soldoff-low       0.57      0.41      0.48        51

        accuracy                           0.53       242
       macro avg       0.56      0.53      0.53       242
    weighted avg       0.56      0.53      0.53       242
```

Model 3:

- `criterion` = gini
- `max_depth` = 8
- `min_samples_leaf` = 6
- `min_impurity_decrease` = 0.003

```
# replace with code to specify DecisionTreeClassifier parameters
model3 = DecisionTreeClassifier(criterion = "gini", random_state=100, max_depth=8, min_samples_leaf=6, min_impurity_decrease=0.0
# replace with code to fit model to training data
model3.fit(x_train, y_train)
```

```
▾                    DecisionTreeClassifier
DecisionTreeClassifier(max_depth=8, min_impurity_decrease=0.003,
                       min_samples_leaf=6, random_state=100)
```

```
# replace with code to predict model3 onto x_test
predictions = model3.predict(x_test)
```

```
# replace with code to print classification report
print(classification_report(y_test, predictions))
```

```
                  precision    recall  f1-score   support

        bankrupt       0.53      0.75      0.62        68
             ipo       0.94      0.42      0.58        36
    soldoff-high       0.60      0.57      0.59        87
     soldoff-low       0.47      0.43      0.45        51

        accuracy                           0.57       242
       macro avg       0.63      0.54      0.56       242
    weighted avg       0.60      0.57      0.57       242
```

Model 4:

- `criterion` = log_loss
- `max_depth` = 8
- `min_samples_leaf` = 6
- `min_impurity_decrease` = 0.002

```
# replace with code to specify DecisionTreeClassifier parameters
model4 = DecisionTreeClassifier(criterion = "log_loss", random_state=100, max_depth=8, min_samples_leaf=6, min_impurity_decrease
# replace with code to fit model to training data
model4.fit(x_train, y_train)
```

```
▾                    DecisionTreeClassifier
DecisionTreeClassifier(criterion='log_loss', max_depth=8,
                       min_impurity_decrease=0.002, min_samples_leaf=6,
                       random_state=100)
```

```
# replace with code to predict model4 onto x_test
predictions = model4.predict(x_test)
```

```
# replace with code to print classification report
print(classification_report(y_test, predictions))
```

```
                  precision    recall  f1-score   support

      bankrupt       0.48      0.66      0.56        68
           ipo       0.59      0.53      0.56        36
  soldoff-high       0.60      0.60      0.60        87
   soldoff-low       0.55      0.31      0.40        51

      accuracy                           0.55       242
     macro avg       0.56      0.53      0.53       242
  weighted avg       0.55      0.55      0.54       242
```

A

What parameters led to the best-performing (highest overall accuracy) model? What was the highest accuracy achieved?

- criterion = gini
- max_depth = 8
- min_samples_leaf = 6
- min_impurity_decrease = 0.003

- This model (model3) let to the highest accuracy, of 57%

B

Plot your best performing decision tree using the `export_graphviz` function from the `tree` library.

```
# replace with code to plot tree of best performing model
labels = y.value_counts()
dot_data = tree.export_graphviz(model3, out_file=None,
                                feature_names=x.columns,
                                class_names=labels.index.values,
                                filled=True)

# Draws graph
graph = graphviz.Source(dot_data, format="png")
graph
```

Q5

Arcturus has $11.5 million to invest and is considering from among 23 applicant firms. Create a Decision Trees model using your most accurate model (i.e., whichever model from Model 1-4 has the highest accuracy percentage), then apply the model as necessary to make predictions regarding the applicant firms.

Data regarding these applicant firms can be found in the `fund-applicants.csv` data file. Note that there is no “outcome” column (the outcome hasn’t happened yet!).

However, there is an `amount` column, which indicates the requested funding amount. Arcturus wants to make as much money as possible by investing its $11.5 million, which may require this amount be spread across multiple companies. However, for any specific applicant

company, Arcturus must either invest the full requested amount or not at all (e.g., if a company has requested $2,500,000, then Arcturus must either invest $2,500,000 or make no investment in the company).

Begin by importing the `fund-applicant.csv` file, then apply your best model to this new data (after taking any required preprocessing steps).

```
# replace with code to import fund-applicants.csv dataset, name this df something different (df_2)
df_2 = pd.read_csv('/content/fund-applicants.csv')
# replace with code to produce heading
df_2.head()
```

	id	industry	months	pfunding	patents	smember	elites	amount
0	1	tech	36	6800000	1	0	4	4000000
1	2	media	23	6300000	0	1	3	1700000
2	3	healthcare	27	6100000	0	1	0	3000000
3	4	tech	18	3500000	0	0	8	2500000
4	5	tech	24	4800000	0	1	6	4000000

```
# replace with code to select IVs to include in analysis, assign to object called 'x'
x = df_2.drop(['id', 'amount'], axis=1)
# replace with code to dummy code categorical variables
x = pd.get_dummies(data = x, drop_first=False)
```

A

In addition to producing the predicted outcomes, also compute the predicted probabilities of the outcome using the `model.predict_proba()` function.

Add both the predicted outcome and the predicted probability as columns to the your dataframe.

```
# replace with code to fit final model to x, assign to object called 'predictions'
predictions = model3.predict(x)
# replace with code to produce predicted probabilities of outcome classes, assign to object called 'proba'
proba = np.max(model3.predict_proba(x), axis=1)
```

```
# replace with code to assign predictions to a new column in new df (dataset you just imported) called 'Predicted_Class'
df_2['predicted_class'] = predictions
# replace with code to assign proba to a new column in new df (dataset you just imported) called 'Predicted_Prob'
df_2['predicted_prob'] = proba
```

B

Describe what the predicted probabilities colum indicates (i.e., what do those values mean?). Are these probabilities based on the training or the test set?

The probabilities column displays how confident we are that the predicted outcome will occur. The they display the probability that the corresponding predicted outcome will occur. They are based off the training set.

C

Using the info provided from the predicted probabilities and predicted outcomes, indicate the companies' project IDs that Arcturus should invest in if, for the sake of diversification, they wanted to invest their $11.5 million in one project in each of the four industries: media, tech, healthcare, and defense.

Give the IDs of the projects and explain why you'd recommend each project.

- **Healthcare:** ID 21 because it has 100% probability that ipo will be the outcome. They have been around a while, have a good number of elites, and have experience in offering IPOs. The return is very high.

- **Defense:** Of the 2 defense companies that don't go bankrupt, ID 15 is the better investment. It's more affordable for the possible outcome, and if anything would be an investment we're limiting losses on.
- **Media:** ID 2 because we are 87% sure the predicted outcome will be soldoff-high. It already has lots of funding and return is high.
- **Tech:** ID 7. To maximize spending of the $11.5 million, this is the best tech investment. We are 59% confident it will be a soldoff-high outcome. It has a high number of elites and has patents. It's been around a while as well.

```
df_2
```