0.) Import the US Perminent Visas using zip extractor

```
import pandas as pd
from google.colab import drive
import matplotlib.pyplot as plt
import numpy as np
```

```
drive.mount('/content/gdrive/', force_remount = True)
```

```
Mounted at /content/gdrive/
```

```
import zipfile

zf = zipfile.ZipFile("us_perm_visas.csv.zip")
df = pd.read_csv(zf.open('us_perm_visas.csv'))
```

```
/usr/local/lib/python3.8/dist-packages/IPython/core/interactiveshell.py:3326: DtypeWarning: Columns (0,1,2,3,4,5,6,7,10,11,16,17,20,21,
  exec(code_obj, self.user_global_ns, self.user_ns)
```

1.) US perm Visas csv from cycle using zip extractor

```
#done above
```

2.) Choose 4 features you think are important. Case_status is your target variable

```
df.head()
```

_job_title_9089	agent_city	agent_firm_name	agent_state	application_type	case_no	case_number	case_received_date	case_status	class
NaN	NaN	NaN	NaN	PERM	A-07323-97014	NaN	NaN	Certified	
NaN	NaN	NaN	NaN	PERM	A-07332-99439	NaN	NaN	Denied	
NaN	NaN	NaN	NaN	PERM	A-07333-99643	NaN	NaN	Certified	
NaN	NaN	NaN	NaN	PERM	A-07339-01930	NaN	NaN	Certified	
NaN	NaN	NaN	NaN	PERM	A-07345-03565	NaN	NaN	Certified	

ns

```
df.columns
```

```
Index(['add_these_pw_job_title_9089', 'agent_city', 'agent_firm_name',
       'agent_state', 'application_type', 'case_no', 'case_number',
       'case_received_date', 'case_status', 'class_of_admission',
       ...
       'ri_pvt_employment_firm_to', 'ri_us_workers_considered',
       'schd_a_sheepherder', 'us_economic_sector', 'wage_offer_from_9089',
       'wage_offer_to_9089', 'wage_offer_unit_of_pay_9089',
       'wage_offered_from_9089', 'wage_offered_to_9089',
       'wage_offered_unit_of_pay_9089'],
      dtype='object', length=154)
```

```
df_select = df[["class_of_admission", "us_economic_sector", "job_info_education", "case_status"]]
```

```
df_select1 = df_select.copy().drop(["case_status"], axis = 1)
```

```
y = df_select["case_status"]
y
```

```
0         Certified
1            Denied
2         Certified
3         Certified
4         Certified
            ...
374357    Withdrawn
374358    Withdrawn
374359    Withdrawn
374360    Withdrawn
374361    Withdrawn
Name: case_status, Length: 374362, dtype: object
```

3.) Clean your data for a decision tree

```
dummies = pd.get_dummies(df_select1[["job_info_education", "us_economic_sector", "class_of_admission"]])
```

```
X = dummies
```

```
X.columns
```

```
Index(['job_info_education_Associate's', 'job_info_education_Bachelor's',
       'job_info_education_Doctorate', 'job_info_education_High School',
       'job_info_education_Master's', 'job_info_education_None',
       'job_info_education_Other', 'us_economic_sector_Advanced Mfg',
       'us_economic_sector_Aerospace', 'us_economic_sector_Agribusiness',
       'us_economic_sector_Automotive', 'us_economic_sector_Biotechnology',
       'us_economic_sector_Construction',
       'us_economic_sector_Educational Services', 'us_economic_sector_Energy',
       'us_economic_sector_Finance', 'us_economic_sector_Geospatial',
       'us_economic_sector_Health Care',
       'us_economic_sector_Homeland Security',
       'us_economic_sector_Hospitality', 'us_economic_sector_IT',
       'us_economic_sector_Other Economic Sector', 'us_economic_sector_Retail',
       'us_economic_sector_Transportation', 'class_of_admission_A-3',
       'class_of_admission_A1/A2', 'class_of_admission_AOS',
       'class_of_admission_AOS/H-1B', 'class_of_admission_B-1',
       'class_of_admission_B-2', 'class_of_admission_C-1',
       'class_of_admission_C-3', 'class_of_admission_D-1',
       'class_of_admission_E-1', 'class_of_admission_E-2',
       'class_of_admission_E-3', 'class_of_admission_EWI',
       'class_of_admission_F-1', 'class_of_admission_F-2',
       'class_of_admission_G-1', 'class_of_admission_G-4',
       'class_of_admission_G-5', 'class_of_admission_H-1A',
       'class_of_admission_H-1B', 'class_of_admission_H-1B1',
       'class_of_admission_H-1C', 'class_of_admission_H-2A',
       'class_of_admission_H-2B', 'class_of_admission_H-3',
       'class_of_admission_H-4', 'class_of_admission_H1B',
       'class_of_admission_I', 'class_of_admission_J-1',
       'class_of_admission_J-2', 'class_of_admission_K-1',
       'class_of_admission_L-1', 'class_of_admission_L-2',
```

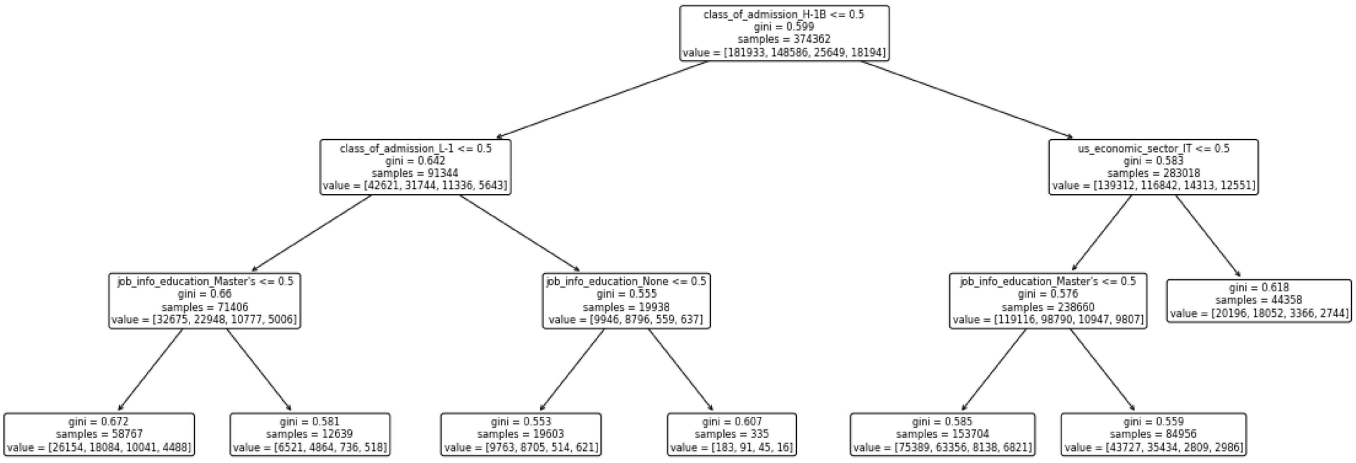
```
       'class_of_admission_M-1', 'class_of_admission_M-2',
       'class_of_admission_N', 'class_of_admission_Not in USA',
       'class_of_admission_O-1', 'class_of_admission_O-2',
       'class_of_admission_O-3', 'class_of_admission_P-1',
       'class_of_admission_P-2', 'class_of_admission_P-3',
       'class_of_admission_P-4', 'class_of_admission_Parol',
       'class_of_admission_Parolee', 'class_of_admission_Q',
       'class_of_admission_R-1', 'class_of_admission_R-2',
       'class_of_admission_T-1', 'class_of_admission_TD',
       'class_of_admission_TN', 'class_of_admission_TPS',
       'class_of_admission_U-1', 'class_of_admission_V-2',
       'class_of_admission_VWB', 'class_of_admission_VWT'],
      dtype='object')
```

4.) Fit and plot a decision tree of depth 3

```
from sklearn import tree
```

```
clf = tree.DecisionTreeClassifier(max_depth = 3)
clf = clf.fit(X,y)
plt.figure(figsize = (20,8))
tree.plot_tree(clf, max_depth = 3, rounded = True, feature_names = X.columns)
```

```
[Text(0.5769230769230769, 0.875, 'class_of_admission_H-1B <= 0.5\ngini = 0.599\nsamples = 374362\nvalue = [181933, 148586, 25649,
18194]'),
 Text(0.3076923076923077, 0.625, 'class_of_admission_L-1 <= 0.5\ngini = 0.642\nsamples = 91344\nvalue = [42621, 31744, 11336, 5643]'),
 Text(0.15384615384615385, 0.375, "job_info_education_Master's <= 0.5\ngini = 0.66\nsamples = 71406\nvalue = [32675, 22948, 10777,
5006]"),
 Text(0.07692307692307693, 0.125, 'gini = 0.672\nsamples = 58767\nvalue = [26154, 18084, 10041, 4488]'),
 Text(0.23076923076923078, 0.125, 'gini = 0.581\nsamples = 12639\nvalue = [6521, 4864, 736, 518]'),
 Text(0.46153846153846156, 0.375, 'job_info_education_None <= 0.5\ngini = 0.555\nsamples = 19938\nvalue = [9946, 8796, 559, 637]'),
 Text(0.38461538461538464, 0.125, 'gini = 0.553\nsamples = 19603\nvalue = [9763, 8705, 514, 621]'),
 Text(0.5384615384615384, 0.125, 'gini = 0.607\nsamples = 335\nvalue = [183, 91, 45, 16]'),
 Text(0.8461538461538461, 0.625, 'us_economic_sector_IT <= 0.5\ngini = 0.583\nsamples = 283018\nvalue = [139312, 116842, 14313,
12551]'),
 Text(0.7692307692307693, 0.375, "job_info_education_Master's <= 0.5\ngini = 0.576\nsamples = 238660\nvalue = [119116, 98790, 10947,
9807]"),
 Text(0.6923076923076923, 0.125, 'gini = 0.585\nsamples = 153704\nvalue = [75389, 63356, 8138, 6821]'),
 Text(0.8461538461538461, 0.125, 'gini = 0.559\nsamples = 84956\nvalue = [43727, 35434, 2809, 2986]'),
 Text(0.9230769230769231, 0.375, 'gini = 0.618\nsamples = 44358\nvalue = [20196, 18052, 3366, 2744]')]
```



5.) Write your interpretation of the largest (by sample size) leaf node

The largest by sample size leaf node had a sample size of 153704 and a gini coefficient of 0.585. The largest leaf is stemmed from nodes that make sure Class_of_admission is 0, meaning they are not admitted, to the H1_B program. If so, the largest leaf -- meaning highest proportion of positive case statuses -- has the person in IT and has a masters. Those who are in IT and have a masters have the highest modal probability to have a certified case status.

```
from sklearn.model_selection import train_test_split
```

6.) Using a for loop, make your own train-test split and determine the best max_depth for out-of sample accuracy

```
#outputs = []
from sklearn.model_selection import train_test_split
#max_depths = [1,2,3]
X_train, X_test, Y_train, Y_test = train_test_split(X,y, test_size = 0.3)
#for nd in max_depths:
  #clf = tree.DecisionTreeClassifier(X_train, Y_train, max_depth = nd)
  #outputs.append(clf.score(X_test, y_test))

from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

max_depth_range = np.arange(1,3)
dt = DecisionTreeClassifier()
grid = GridSearchCV(dt, param_grid={'max_depth': max_depth_range}, cv=5)
grid.fit(X_train,Y_train)

best_max_depth = grid.best_params_['max_depth']

dt_final = DecisionTreeClassifier(max_depth=best_max_depth)
dt_final.fit(X_train,Y_train)
```

```
DecisionTreeClassifier(max_depth=1)
```

```
#The best max depth given by the output is 1.
```

```
[]
```