0.) Import the US Perminent Visas using zip extractor

In [34]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [35]:
```
df = pd.read_csv("/Users/snehilshandilya/Desktop/us_perm_visas.csv")
```

```
/var/folders/sv/s309_3dd79s_59j12prhgcd00000gn/T/ipykernel_47368/4005682600.py:1: DtypeWarning: Columns
(0,1,2,3,4,5,6,7,10,11,16,17,20,21,22,25,26,27,28,29,30,31,32,33,34,35,36,37,39,40,41,42,43,44,45,47,48
,49,50,51,52,53,55,56,57,58,59,60,61,63,64,65,66,68,69,70,71,72,73,74,77,79,80,81,82,83,84,85,86,87,88,
89,90,91,92,93,94,95,100,101,103,104,105,106,107,108,109,110,111,112,113,114,115,116,117,118,119,120,12
1,122,123,124,125,126,127,128,129,130,131,132,133,134,135,136,137,138,139,140,141,142,143,144,145,146,1
47,148,149,150,153) have mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv("/Users/snehilshandilya/Desktop/us_perm_visas.csv")
```

In [36]:
```
df
```

Out[36]:

	add_these_pw_job_title_9089	agent_city	agent_firm_name	agent_state	application_type	case_no	case_number	case_received_date
0	NaN	NaN	NaN	NaN	PERM	A-07323-97014	NaN	NaN
1	NaN	NaN	NaN	NaN	PERM	A-07332-99439	NaN	NaN
2	NaN	NaN	NaN	NaN	PERM	A-07333-99643	NaN	NaN
						A-		

3	NaN	NaN	NaN	NaN	PERM	07339-01930	NaN	NaN
4	NaN	NaN	NaN	NaN	PERM	A-07345-03565	NaN	NaN
...	...	...	...	...	...	...	...	...
374357	NaN	Buena Park	Law Offices of Yohan Lee	CA	NaN	NaN	A-16363-85407	2016-12-29
374358	NaN	Seattle	MacDonald Hoague & Bayless	WA	NaN	NaN	A-16271-56745	2016-12-30
374359	NaN	Schaumburg	International Legal and Business Services Grou...	IL	NaN	NaN	A-16354-82345	2016-12-30
374360	NaN	LOS ANGELES	LAW OFFICES OF JAMES S HONG	CA	NaN	NaN	A-16357-84250	2016-12-30
374361	NaN	Phoenix	Fragomen, Del Rey, Bernsen & Loewy, LLP	AZ	NaN	NaN	A-16279-59292	2016-12-30

374362 rows × 154 columns

In [4]:
```
#import zipfile

#df = pd.read_csv(zf.open('us_perm_visas.csv'))
```

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

In [ ]:
```
```

In [ ]:

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

In [37]:
```
df_select = df[["class_of_admission", "case_status", "country_of_citzenship","job_info_work_city","job_i
```

In [38]:
```
df_select
```

Out[38]:

	class_of_admission	case_status	country_of_citzenship	job_info_work_city	job_info_work_state
0	J-1	Certified	ARMENIA	New York	NY
1	B-2	Denied	POLAND	New York	NY
2	H-1B	Certified	INDIA	Lutherville	MD
3	B-2	Certified	SOUTH KOREA	Flushing	NY
4	L-1	Certified	CANADA	Albany	NY
...	...	...	...	...	...
374357	NaN	Withdrawn	NaN	Anaheim	CA
374358	L-1	Withdrawn	NaN	Cambridge	MA
374359	H-1B	Withdrawn	NaN	Ypsilanti	MI
374360	B-2	Withdrawn	NaN	Phoenix	AZ
374361	H-1B	Withdrawn	NaN	Fort Collins	CO

374362 rows × 5 columns

In [41]:
```
X = pd.get_dummies(df_select, prefix=[ "class_of_admission", "country_of_citzenship" ,"job_info_work_cit
X = X.drop(['case_status'], axis =1)
y = df_select["case_status"]
```

In [42]:
```
X
```

Out[42]:

	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_c
0	0	0	0	0	0	
1	0	0	0	0	0	
2	0	0	0	0	0	
3	0	0	0	0	0	
4	0	0	0	0	0	
...	...	...	...	...	...	
374357	0	0	0	0	0	
374358	0	0	0	0	0	
374359	0	0	0	0	0	
374360	0	0	0	0	0	
374361	0	0	0	0	0	

374362 rows × 9747 columns

In [43]:
```
y
```

Out[43]:
```
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
```

In [ ]:
```

```

3.) Clean your data for a decision tree

In [44]:
```
import pandas as pd

# Handle missing values
df_select.dropna()
```

Out[44]:

	class_of_admission	case_status	country_of_citzenship	job_info_work_city	job_info_work_state
0	J-1	Certified	ARMENIA	New York	NY
1	B-2	Denied	POLAND	New York	NY
2	H-1B	Certified	INDIA	Lutherville	MD
3	B-2	Certified	SOUTH KOREA	Flushing	NY
4	L-1	Certified	CANADA	Albany	NY
...	...	...	...	...	...
20571	H-2B	Certified	MEXICO	Minturn	CO
20572	EWI	Withdrawn	MEXICO	Los Angeles	CA
20573	E-2	Withdrawn	SOUTH KOREA	Everett	WA
20574	Not in USA	Denied	CHINA	Winesburg	OH
20575	B-2	Certified-Expired	UNITED KINGDOM	Chino	CA

19363 rows × 5 columns

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

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

In [45]:
```
from sklearn import tree
```

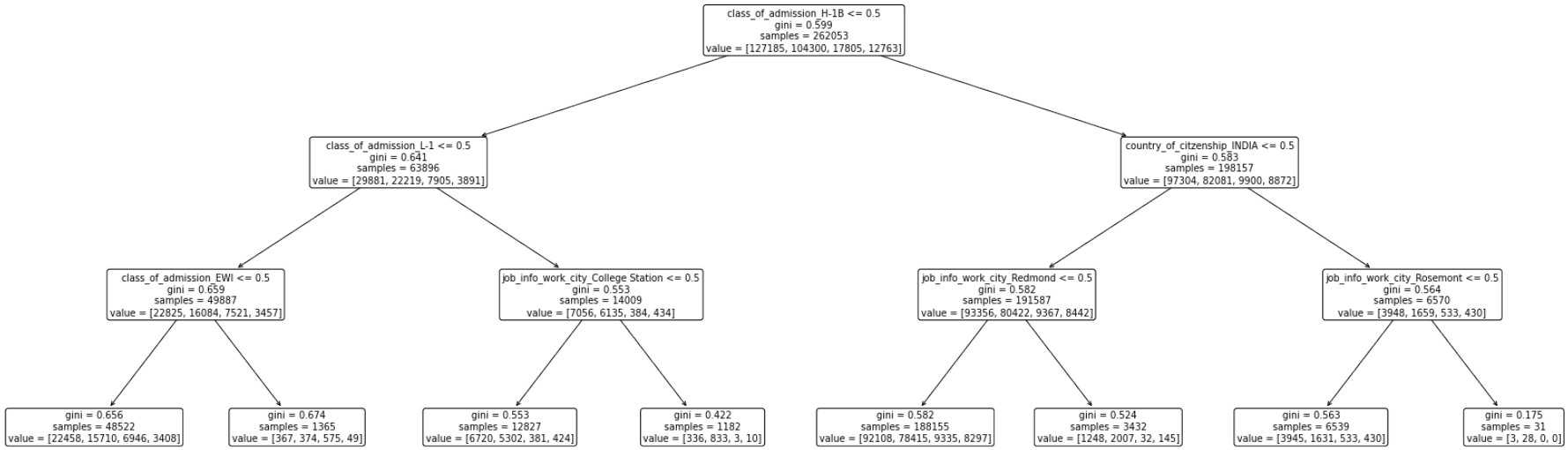
In [46]:
```
clf = tree.DecisionTreeClassifier(max_depth = 3)
clf.fit(X,y)
```

Out[46]:
```
DecisionTreeClassifier(max_depth=3)
```

In [51]:
```
plt.figure(figsize = (30,10))
tree.plot_tree(clf, max_depth = 3, rounded = True, feature_names = X.columns)
```

Out[51]:
```
[Text(0.5, 0.875, 'class_of_admission_H-1B <= 0.5\ngini = 0.599\nsamples = 262053\nvalue = [127185, 104
300, 17805, 12763]'),
 Text(0.25, 0.625, 'class_of_admission_L-1 <= 0.5\ngini = 0.641\nsamples = 63896\nvalue = [29881, 22219
, 7905, 3891]'),
 Text(0.125, 0.375, 'class_of_admission_EWI <= 0.5\ngini = 0.659\nsamples = 49887\nvalue = [22825, 1608
4, 7521, 3457]'),
 Text(0.0625, 0.125, 'gini = 0.656\nsamples = 48522\nvalue = [22458, 15710, 6946, 3408]'),
 Text(0.1875, 0.125, 'gini = 0.674\nsamples = 1365\nvalue = [367, 374, 575, 49]'),
 Text(0.375, 0.375, 'job_info_work_city_College Station <= 0.5\ngini = 0.553\nsamples = 14009\nvalue =
[7056, 6135, 384, 434]'),
 Text(0.3125, 0.125, 'gini = 0.553\nsamples = 12827\nvalue = [6720, 5302, 381, 424]'),
 Text(0.4375, 0.125, 'gini = 0.422\nsamples = 1182\nvalue = [336, 833, 3, 10]'),
 Text(0.75, 0.625, 'country_of_citzenship_INDIA <= 0.5\ngini = 0.583\nsamples = 198157\nvalue = [97304,
```

```
82081, 9900, 8872]'),
 Text(0.625, 0.375, 'job_info_work_city_Redmond <= 0.5\ngini = 0.582\nsamples = 191587\nvalue = [93356,
80422, 9367, 8442]'),
 Text(0.5625, 0.125, 'gini = 0.582\nsamples = 188155\nvalue = [92108, 78415, 9335, 8297]'),
 Text(0.6875, 0.125, 'gini = 0.524\nsamples = 3432\nvalue = [1248, 2007, 32, 145]'),
 Text(0.875, 0.375, 'job_info_work_city_Rosemont <= 0.5\ngini = 0.564\nsamples = 6570\nvalue = [3948, 1
659, 533, 430]'),
 Text(0.8125, 0.125, 'gini = 0.563\nsamples = 6539\nvalue = [3945, 1631, 533, 430]'),
 Text(0.9375, 0.125, 'gini = 0.175\nsamples = 31\nvalue = [3, 28, 0, 0]')]
```



In [53]:
```
y.unique()
```

Out[53]:
```
array(['Certified', 'Denied', 'Certified-Expired', 'Withdrawn'],
      dtype=object)
```

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

Node 1: For the first node, at the very top, we see that the leaf node which has the largest sample size is that of categorization according to the citizenship of the country INDIA. Here, we see that since it says '<= 0.5' that means it is not India. It says that there are about 198157 people who are NOT FROM India and out of them about 97000 of them got an H1B Visa.

Node 2: The node just below that, we see that the sample size is greater for the city of Redmond. We interpret this is as the one WHO ARE NOT FROM INDIA. Hence, people who reside in the city of Redmond, are not citizens of India but HAVE APPLIED an H1B visa are about 93,356.

Node 3: In the last node, we see that there are about 188155 people who DO NOT RESIDE in Redmond and ARE NOT INDIAN CITIZENS, and ones who APPLIED for an H1B visa are about 92,108, ones who were denied are about 78415, 9335 are certified-expered and 8297 have withdrawn.

In [ ]:

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

In [48]:
```
from sklearn.model_selection import train_test_split
```

In [49]:
```
from sklearn.metrics import accuracy_score

outputs = []

max_depths = [1,2,3]
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.3)

for md in max_depths:
    clf = tree.DecisionTreeClassifier(max_depth = md)
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    outputs.append(accuracy)
```

In [55]:
```
outputs
```

Out[55]:
```
[0.4874765156844064, 0.4874765156844064, 0.49377164786437416]
```

This shows that 3 level of depths shows the highest accuracy.

In [54]:
```
accuracy
```

Out[54]:
```
0.49377164786437416
```

In [ ]:

In [ ]: