```
import pandas as pd
```

1. Import the data set (1 points)

```
url = "https://raw.githubusercontent.com/jackty9/Handling_Imbalanced_Data_in_Python/master/bank-full-encoded.csv"
df = pd.read_csv(url)
```

```
df.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	du
0	40	4	1	2	0	3036	1	0	2	4	8	
1	26	9	2	1	0	945	1	0	2	4	8	
2	15	2	1	1	0	918	1	1	2	4	8	
3	29	1	1	3	0	2420	1	0	2	4	8	
4	15	11	2	3	0	917	0	0	2	4	8	

2. Print the descriptive statistics of the admission data to understand the data a little better (min, max, mean, median, 1st and 3rd quartiles).? (1 points)

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
age	45211.0	22.936055	10.618004	0.0	15.0	21.0	30.0	76.0
job	45211.0	4.339762	3.272657	0.0	1.0	4.0	7.0	11.0
marital	45211.0	1.167725	0.608230	0.0	1.0	1.0	2.0	2.0
education	45211.0	1.224813	0.747997	0.0	1.0	1.0	2.0	3.0
default	45211.0	0.018027	0.133049	0.0	0.0	0.0	0.0	1.0
balance	45211.0	1963.307469	1463.533246	0.0	988.0	1364.0	2344.0	7167.0
housing	45211.0	0.555838	0.496878	0.0	0.0	1.0	1.0	1.0
loan	45211.0	0.160226	0.366820	0.0	0.0	0.0	0.0	1.0
contact	45211.0	0.640242	0.897951	0.0	0.0	0.0	2.0	2.0
day	45211.0	14.806419	8.322476	0.0	7.0	15.0	20.0	30.0
month	45211.0	5.523014	3.006911	0.0	3.0	6.0	8.0	11.0
duration	45211.0	255.338502	239.660852	0.0	103.0	180.0	319.0	1572.0
campaign	45211.0	1.762381	3.075904	0.0	0.0	1.0	2.0	47.0
pdays	45211.0	40.154188	96.917547	0.0	0.0	0.0	0.0	558.0
previous	45211.0	0.573356	1.877700	0.0	0.0	0.0	0.0	40.0
poutcome	45211.0	2.559974	0.989059	0.0	3.0	3.0	3.0	3.0
y	45211.0	0.116985	0.321406	0.0	0.0	0.0	0.0	1.0

3. Splitting the Data-Set into Independent and Dependent Features. (1 points)

```
#check back on this
dfx = df.drop('y', axis = 1)
dfx.head(2)
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	du
0	40	4	1	2	0	3036	1	0	2	4	8	
1	26	9	2	1	0	945	1	0	2	4	8	

```
dfy = df['y']
dfy.head(2)
```

```
0    0
1    0
Name: y, dtype: int64
```

4. Convert categorical variable into numeric Using one hot encoding method. (1 points)

```
dfx["job"] = dfx["job"].astype("category")
dfx["marital"] = dfx["marital"].astype("category")
dfx["education"] = dfx["education"].astype("category")
dfx["default"] = dfx["default"].astype("category")
dfx["housing"] = dfx["housing"].astype("category")
dfx["loan"] = dfx["loan"].astype("category")
dfx["contact"] = dfx["contact"].astype("category")
dfx["month"] = dfx["month"].astype("category")
dfx["poutcome"] = dfx["poutcome"].astype("category")
```

```
dfx.dtypes
```

```
age             int64
job          category
marital      category
education    category
default      category
balance         int64
housing      category
loan         category
contact      category
day             int64
month        category
duration        int64
campaign        int64
pdays           int64
previous        int64
poutcome     category
dtype: object
```

```
dfx = pd.get_dummies(dfx)
dfx.head(2)
```

	age	balance	day	duration	campaign	pdays	previous	job_0	job_1	job_2	...	mont
0	40	3036	4	261	0	0	0	0	0	0	...	
1	26	945	4	151	0	0	0	0	0	0	...	

2 rows × 51 columns

5. Normalize the data set. (1 points)

```
dfx.describe().T
```

	count	mean	std	min	25%	50%	75%	max
age	45211.0	22.936055	10.618004	0.0	15.0	21.0	30.0	76.0
balance	45211.0	1963.307469	1463.533246	0.0	988.0	1364.0	2344.0	7167.0
day	45211.0	14.806419	8.322476	0.0	7.0	15.0	20.0	30.0
duration	45211.0	255.338502	239.660852	0.0	103.0	180.0	319.0	1572.0
campaign	45211.0	1.762381	3.075904	0.0	0.0	1.0	2.0	47.0
pdays	45211.0	40.154188	96.917547	0.0	0.0	0.0	0.0	558.0
previous	45211.0	0.573356	1.877700	0.0	0.0	0.0	0.0	40.0
job_0	45211.0	0.114375	0.318269	0.0	0.0	0.0	0.0	1.0
job_1	45211.0	0.215257	0.411005	0.0	0.0	0.0	0.0	1.0
job_2	45211.0	0.032890	0.178351	0.0	0.0	0.0	0.0	1.0
job_3	45211.0	0.027427	0.163326	0.0	0.0	0.0	0.0	1.0
job_4	45211.0	0.209197	0.406740	0.0	0.0	0.0	0.0	1.0
job_5	45211.0	0.050076	0.218105	0.0	0.0	0.0	0.0	1.0
job_6	45211.0	0.034925	0.183592	0.0	0.0	0.0	0.0	1.0
job_7	45211.0	0.091880	0.288860	0.0	0.0	0.0	0.0	1.0
job_8	45211.0	0.020747	0.142538	0.0	0.0	0.0	0.0	1.0
job_9	45211.0	0.168034	0.373901	0.0	0.0	0.0	0.0	1.0
job_10	45211.0	0.028820	0.167303	0.0	0.0	0.0	0.0	1.0
job_11	45211.0	0.006370	0.079559	0.0	0.0	0.0	0.0	1.0
marital_0	45211.0	0.115171	0.319232	0.0	0.0	0.0	0.0	1.0
marital_1	45211.0	0.601933	0.489505	0.0	0.0	1.0	1.0	1.0
marital_2	45211.0	0.282896	0.450411	0.0	0.0	0.0	1.0	1.0
education_0	45211.0	0.151534	0.358572	0.0	0.0	0.0	0.0	1.0
education_1	45211.0	0.513194	0.499831	0.0	0.0	1.0	1.0	1.0
education_2	45211.0	0.294198	0.455687	0.0	0.0	0.0	1.0	1.0
education_3	45211.0	0.041074	0.198464	0.0	0.0	0.0	0.0	1.0
default_0	45211.0	0.981973	0.133049	0.0	1.0	1.0	1.0	1.0
default_1	45211.0	0.018027	0.133049	0.0	0.0	0.0	0.0	1.0
housing_0	45211.0	0.444162	0.496878	0.0	0.0	0.0	1.0	1.0
housing_1	45211.0	0.555838	0.496878	0.0	0.0	1.0	1.0	1.0
loan_0	45211.0	0.839774	0.366820	0.0	1.0	1.0	1.0	1.0

```
dfx.head()
```

	age	balance	day	duration	campaign	pdays	previous	job_0	job_1	job_2	...	mont
0	40	3036	4	261	0	0	0	0	0	0	...	
1	26	945	4	151	0	0	0	0	0	0	...	
2	15	918	4	76	0	0	0	0	0	1	...	

```
from sklearn.preprocessing import MinMaxScaler
```

```
scaler = MinMaxScaler()
```

```
scaler.fit_transform(dfx)
```

```
array([[0.52631579, 0.4236082 , 0.13333333, ..., 0.        , 0.        ,
        1.        ],
       [0.34210526, 0.13185433, 0.13333333, ..., 0.        , 0.        ,
        1.        ],
       [0.19736842, 0.12808707, 0.13333333, ..., 0.        , 0.        ,
        1.        ],
       ...,
       [0.71052632, 0.76112739, 0.53333333, ..., 0.        , 1.        ,
        0.        ],
       [0.51315789, 0.22101298, 0.53333333, ..., 0.        , 0.        ,
        1.        ],
       [0.25      , 0.5272778 , 0.53333333, ..., 1.        , 0.        ,
        0.        ]])
```

```
pd.DataFrame(scaler.fit_transform(dfx),columns = dfx.columns)
```

	age	balance	day	duration	campaign	pdays	previous	job_0	job_1
0	0.526316	0.423608	0.133333	0.166031	0.000000	0.000000	0.000	0.0	0.0
1	0.342105	0.131854	0.133333	0.096056	0.000000	0.000000	0.000	0.0	0.0
2	0.197368	0.128087	0.133333	0.048346	0.000000	0.000000	0.000	0.0	0.0
3	0.381579	0.337659	0.133333	0.058524	0.000000	0.000000	0.000	0.0	1.0
4	0.197368	0.127948	0.133333	0.125954	0.000000	0.000000	0.000	0.0	0.0
...	...	...	...	...	...	...	...	...	...
45206	0.434211	0.242919	0.533333	0.620229	0.042553	0.000000	0.000	0.0	0.0
45207	0.697368	0.368215	0.533333	0.290076	0.021277	0.000000	0.000	0.0	0.0
45208	0.710526	0.761127	0.533333	0.709924	0.085106	0.324373	0.075	0.0	0.0
45209	0.513158	0.221013	0.533333	0.323155	0.063830	0.000000	0.000	0.0	1.0
45210	0.250000	0.527278	0.533333	0.229644	0.021277	0.331541	0.275	0.0	0.0

45211 rows × 51 columns

```
scaler_dfx = pd.DataFrame(scaler.fit_transform(dfx),columns = dfx.columns)
scaler_dfx.head(5)
```

	age	balance	day	duration	campaign	pdays	previous	job_0	job_1	job_2
0	0.526316	0.423608	0.133333	0.166031	0.0	0.0	0.0	0.0	0.0	0.0
1	0.342105	0.131854	0.133333	0.096056	0.0	0.0	0.0	0.0	0.0	0.0
2	0.197368	0.128087	0.133333	0.048346	0.0	0.0	0.0	0.0	0.0	1.0
3	0.381579	0.337659	0.133333	0.058524	0.0	0.0	0.0	0.0	1.0	0.0
4	0.197368	0.127948	0.133333	0.125954	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 51 columns

6. Divide the dataset to training and test sets. (1 points)

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

```
input_train, input_test, output_train, output_test = train_test_split(scaler_dfx, dfy, test_size=0.2, random_state=1)
```

```
input_train
```

	age	balance	day	duration	campaign	pdays	previous	job_0	job_1
22468	0.447368	0.127808	0.700000	0.147583	0.021277	0.000000	0.000	0.0	0.0
6896	0.421053	0.173852	0.900000	0.015267	0.000000	0.000000	0.000	1.0	0.0
28408	0.355263	0.169667	0.933333	0.129135	0.000000	0.462366	0.075	0.0	1.0
8481	0.263158	0.131575	0.066667	0.256997	0.063830	0.000000	0.000	0.0	0.0
28753	0.171053	0.251570	0.966667	0.271628	0.000000	0.000000	0.000	0.0	0.0
...	...	...	...	...	...	...	...	...	...
43723	0.421053	0.381471	0.433333	0.202290	0.042553	0.000000	0.000	0.0	0.0
32511	0.210526	0.201479	0.533333	0.148219	0.021277	0.000000	0.000	0.0	0.0
5192	0.394737	0.651598	0.666667	0.230916	0.106383	0.000000	0.000	0.0	0.0
12172	0.355263	0.375471	0.633333	0.006997	0.212766	0.000000	0.000	1.0	0.0
33003	0.355263	0.176503	0.533333	0.113868	0.021277	0.596774	0.050	0.0	1.0

36168 rows × 51 columns

```
input_train,input_test,output_train,output_test
```

```
(            age   balance       day  duration  campaign     pdays  previous  \
 22468  0.447368  0.127808  0.700000  0.147583  0.021277  0.000000     0.000
 6896   0.421053  0.173852  0.900000  0.015267  0.000000  0.000000     0.000
 28408  0.355263  0.169667  0.933333  0.129135  0.000000  0.462366     0.075
 8481   0.263158  0.131575  0.066667  0.256997  0.063830  0.000000     0.000
 28753  0.171053  0.251570  0.966667  0.271628  0.000000  0.000000     0.000
 ...         ...       ...       ...       ...       ...       ...       ...
 43723  0.421053  0.381471  0.433333  0.202290  0.042553  0.000000     0.000
 32511  0.210526  0.201479  0.533333  0.148219  0.021277  0.000000     0.000
 5192   0.394737  0.651598  0.666667  0.230916  0.106383  0.000000     0.000
 12172  0.355263  0.375471  0.633333  0.006997  0.212766  0.000000     0.000
 33003  0.355263  0.176503  0.533333  0.113868  0.021277  0.596774     0.050

        job_0  job_1  job_2  ...  month_6  month_7  month_8  month_9  month_10  \
 22468    0.0    0.0    0.0  ...      0.0      0.0      0.0      0.0       0.0
 6896     1.0    0.0    0.0  ...      0.0      0.0      1.0      0.0       0.0
 28408    0.0    1.0    0.0  ...      0.0      0.0      0.0      0.0       0.0
 8481     0.0    0.0    0.0  ...      1.0      0.0      0.0      0.0       0.0
 28753    0.0    0.0    0.0  ...      0.0      0.0      0.0      0.0       0.0
 ...      ...    ...    ...  ...      ...      ...      ...      ...       ...
 43723    0.0    0.0    0.0  ...      0.0      0.0      1.0      0.0       0.0
 32511    0.0    0.0    0.0  ...      0.0      0.0      0.0      0.0       0.0
 5192     0.0    0.0    0.0  ...      0.0      0.0      1.0      0.0       0.0
 12172    1.0    0.0    0.0  ...      1.0      0.0      0.0      0.0       0.0
 33003    0.0    1.0    0.0  ...      0.0      0.0      0.0      0.0       0.0

        month_11  poutcome_0  poutcome_1  poutcome_2  poutcome_3
 22468       0.0         0.0         0.0         0.0         1.0
 6896        0.0         0.0         0.0         0.0         1.0
 28408       0.0         1.0         0.0         0.0         0.0
 8481        0.0         0.0         0.0         0.0         1.0
 28753       0.0         0.0         0.0         0.0         1.0
 ...         ...         ...         ...         ...         ...
 43723       0.0         0.0         0.0         0.0         1.0
 32511       0.0         0.0         0.0         0.0         1.0
 5192        0.0         0.0         0.0         0.0         1.0
 12172       0.0         0.0         0.0         0.0         1.0
 33003       0.0         0.0         1.0         0.0         0.0

 [36168 rows x 51 columns],
            age   balance       day  duration  campaign     pdays  previous  \
 3610   0.315789  0.471746  0.466667  0.166667  0.063830  0.000000     0.000
 11677  0.250000  0.432538  0.633333  0.106234  0.021277  0.000000     0.000
 33018  0.184211  0.256593  0.533333  0.520992  0.063830  0.000000     0.000
 44323  0.460526  0.170504  0.900000  0.246819  0.042553  0.318996     0.025
 8119   0.184211  0.163667  0.033333  0.116412  0.085106  0.000000     0.000
 ...         ...       ...       ...       ...       ...       ...       ...
 22959  0.500000  0.127808  0.833333  0.064885  0.021277  0.000000     0.000
 26059  0.197368  0.626064  0.600000  0.430025  0.021277  0.000000     0.000
 18593  0.342105  0.311567  1.000000  0.282443  0.106383  0.000000     0.000
 6959   0.171053  0.170364  0.900000  0.188931  0.276596  0.000000     0.000
```

```
33795  0.131579  0.319520  0.733333  0.129135  0.063830  0.000000     0.000

       job_0  job_1  job_2  ...  month_6  month_7  month_8  month_9  month_10  \
3610     0.0    1.0    0.0  ...      0.0      0.0      1.0      0.0       0.0
11677    0.0    0.0    0.0  ...      1.0      0.0      0.0      0.0       0.0
33018    1.0    0.0    0.0  ...      0.0      0.0      0.0      0.0       0.0
44323    0.0    1.0    0.0  ...      0.0      0.0      0.0      0.0       0.0
```

```
output_train.value_counts()
```

```
0    31929
1     4239
Name: y, dtype: int64
```

7. Use the Decision Tree algorithm to predict the test set out values (1 points)

```
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
```

```
dfx.columns
```

```
Index(['age', 'balance', 'day', 'duration', 'campaign', 'pdays', 'previous',
       'job_0', 'job_1', 'job_2', 'job_3', 'job_4', 'job_5', 'job_6', 'job_7',
       'job_8', 'job_9', 'job_10', 'job_11', 'marital_0', 'marital_1',
       'marital_2', 'education_0', 'education_1', 'education_2', 'education_3',
       'default_0', 'default_1', 'housing_0', 'housing_1', 'loan_0', 'loan_1',
       'contact_0', 'contact_1', 'contact_2', 'month_0', 'month_1', 'month_2',
       'month_3', 'month_4', 'month_5', 'month_6', 'month_7', 'month_8',
       'month_9', 'month_10', 'month_11', 'poutcome_0', 'poutcome_1',
       'poutcome_2', 'poutcome_3'],
      dtype='object')
```

```
features = ['age', 'balance', 'day', 'duration', 'campaign', 'pdays', 'previous',
       'job_0', 'job_1', 'job_2', 'job_3', 'job_4', 'job_5', 'job_6', 'job_7',
       'job_8', 'job_9', 'job_10', 'job_11', 'marital_0', 'marital_1',
       'marital_2', 'education_0', 'education_1', 'education_2', 'education_3',
       'default_0', 'default_1', 'housing_0', 'housing_1', 'loan_0', 'loan_1',
       'contact_0', 'contact_1', 'contact_2', 'month_0', 'month_1', 'month_2',
       'month_3', 'month_4', 'month_5', 'month_6', 'month_7', 'month_8',
       'month_9', 'month_10', 'month_11', 'poutcome_0', 'poutcome_1',
       'poutcome_2', 'poutcome_3']
```

```
dtree = DecisionTreeClassifier(criterion="entropy", random_state=42,max_depth=4)
dtree = dtree.fit(dfx,dfy)
tree.plot_tree(dtree,feature_names=features)
```

```
[Text(0.5, 0.9, 'duration <= 410.5\nentropy = 0.521\nsamples = 45211\nvalue = [39922,
5289]'),
 Text(0.25, 0.7, 'poutcome_2 <= 0.5\nentropy = 0.357\nsamples = 37668\nvalue = [35121,
2547]'),
 Text(0.125, 0.5, 'contact_2 <= 0.5\nentropy = 0.288\nsamples = 36493\nvalue = [34655,
1838]'),
 Text(0.0625, 0.3, 'duration <= 129.5\nentropy = 0.364\nsamples = 25534\nvalue =
[23761, 1773]'),
 Text(0.03125, 0.1, 'entropy = 0.125\nsamples = 10804\nvalue = [10619, 185]'),
 Text(0.09375, 0.1, 'entropy = 0.493\nsamples = 14730\nvalue = [13142, 1588]'),
 Text(0.1875, 0.3, 'month_10 <= 0.5\nentropy = 0.052\nsamples = 10959\nvalue = [10894,
65]'),
 Text(0.15625, 0.1, 'entropy = 0.042\nsamples = 10912\nvalue = [10862, 50]'),
 Text(0.21875, 0.1, 'entropy = 0.903\nsamples = 47\nvalue = [32, 15]'),
 Text(0.375, 0.5, 'duration <= 132.5\nentropy = 0.969\nsamples = 1175\nvalue = [466,
709]'),
 Text(0.3125, 0.3, 'month_8 <= 0.5\nentropy = 0.768\nsamples = 241\nvalue = [187,
54]'),
 Text(0.28125, 0.1, 'entropy = 0.826\nsamples = 204\nvalue = [151, 53]'),
 Text(0.34375, 0.1, 'entropy = 0.179\nsamples = 37\nvalue = [36, 1]'),
 Text(0.4375, 0.3, 'duration <= 162.5\nentropy = 0.88\nsamples = 934\nvalue = [279,
655]'),
 Text(0.40625, 0.1, 'entropy = 1.0\nsamples = 119\nvalue = [60, 59]'),
```

8. Evaluate the model performance by computing Accuracy. (1 points)

```
 Text(0.625, 0.5, 'poutcome_2 <= 0.5\nentropy = 0.817\nsamples = 4351\nvalue = [3247
```

```
pred_test = dtree.predict(input_test)
pred_test
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
 Text(0.6875, 0.3, 'housing_0 <= 0.5\nentropy = 0.744\nsamples = 208\nvalue = [44
```

```
output_test
```

```
3610     0
11677    0
33018    0
44323    1
8119     0
        ..
22959    0
26059    0
18593    0
6959     0
33795    1
Name: y, Length: 9043, dtype: int64
```

```
from sklearn.metrics import accuracy_score
```

```
accuracy_score_decision = round(accuracy_score(output_test,pred_test)*100,0)
accuracy_score_decision
```

```
88.0
```