```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # data visualization
import seaborn as sns # statistical data visualization
%matplotlib inline
```

```
!pip
```

```
  inspect                     Inspect the python environment.
  list                        List installed packages.
  show                        Show information about installed packages.
  check                       Verify installed packages have compatible dependencies.
  config                      Manage local and global configuration.
  search                      Search PyPI for packages.
  cache                       Inspect and manage pip's wheel cache.
  index                       Inspect information available from package indexes.
  wheel                       Build wheels from your requirements.
  hash                        Compute hashes of package archives.
  completion                  A helper command used for command completion.
  debug                       Show information useful for debugging.
  help                        Show help for commands.

General Options:
  -h, --help                  Show help.
  --debug                     Let unhandled exceptions propagate outside the main subroutine,
                              instead of logging them to stderr.
  --isolated                  Run pip in an isolated mode, ignoring environment variables and us
                              configuration.
  --require-virtualenv        Allow pip to only run in a virtual environment; exit with an error
                              otherwise.
  --python <python>           Run pip with the specified Python interpreter.
  -v, --verbose               Give more output. Option is additive, and can be used up to 3 time
  -V, --version               Show version and exit.
  -q, --quiet                 Give less output. Option is additive, and can be used up to 3 time
                              (corresponding to WARNING, ERROR, and CRITICAL logging levels).
  --log <path>                Path to a verbose appending log.
  --no-input                  Disable prompting for input.
  --keyring-provider <keyring_provider>
                              Enable the credential lookup via the keyring library if user input
                              is allowed. Specify which mechanism to use [disabled, import,
                              subprocess]. (default: disabled)
  --proxy <proxy>             Specify a proxy in the form
                              scheme://[user:passwd@]proxy.server:port.
  --retries <retries>         Maximum number of retries each connection should attempt (default
                              times).
  --timeout <sec>             Set the socket timeout (default 15 seconds).
  --exists-action <action>    Default action when a path already exists: (s)witch, (i)gnore,
                              (w)ipe, (b)ackup, (a)bort.
  --trusted-host <hostname>   Mark this host or host:port pair as trusted, even though it does r
                              have valid or any HTTPS.
  --cert <path>               Path to PEM-encoded CA certificate bundle. If provided, overrides
                              the default. See 'SSL Certificate Verification' in pip documentati
                              for more information.
  --client-cert <path>        Path to SSL client certificate, a single file containing the priva
                              key and the certificate in PEM format.
  --cache-dir <dir>           Store the cache data in <dir>.
  --no-cache-dir              Disable the cache.
```

```
import os

dataset_path = r"/content/Book1.1.csv"  # Replace with your actual dataset path

for dirname, _, filenames in os.walk(dataset_path):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
import warnings

warnings.filterwarnings('ignore')
```

```
data ="/content/Book1.1.csv"

df = pd.read_csv(data, header=None)
```

```
df.shape
```

```
(4425, 13)
```

```
df.head()
```

	0	1	2	3	4	5	6	
0	Marital status	Application mode	Application order	Course	Daytime/evening attendance	Previous qualification	Nacionality	Moth qualifica
1	1	8	5	2	1	1	1	
2	1	6	1	11	1	1	1	
3	1	1	5	5	1	1	1	
4	1	8	2	15	1	1	1	

Next steps: Generate code with df View recommended plots

```
col_names = ['Marital status', 'Application mode', 'Application order', 'Course','Daytime/evening atte
```

```
df.columns = col_names
```

```
col_names
```

```
['Marital status',
 'Application mode',
 'Application order',
 'Course',
 'Daytime/evening attendance',
 'Previous qualification',
 'Nacionality',
 'Mothers qualification',
 'Fathers qualification',
 'Mothers occupation',
 'Fathers occupation',
 'Displaced',
 'Educational special needs']
```

```
# let's again preview the dataset

df.head()
```

	Marital status	Application mode	Application order	Course	Daytime/evening attendance	Previous qualification	Naciona
0	Marital status	Application mode	Application order	Course	Daytime/evening attendance	Previous qualification	Nacic
1	1	8	5	2	1	1	
2	1	6	1	11	1	1	
3	1	1	5	5	1	1	
4	1	8	2	15	1	1	

Next steps: Generate code with df View recommended plots

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4425 entries, 0 to 4424
Data columns (total 13 columns):
 #   Column                      Non-Null Count  Dtype 
---  ------                      --------------  ----- 
 0   Marital status              4425 non-null   object
 1   Application mode            4425 non-null   object
 2   Application order           4425 non-null   object
 3   Course                      4425 non-null   object
 4   Daytime/evening attendance  4425 non-null   object
 5   Previous qualification      4425 non-null   object
 6   Nacionality                 4425 non-null   object
 7   Mothers qualification       4425 non-null   object
 8   Fathers qualification       4425 non-null   object
 9   Mothers occupation          4425 non-null   object
 10  Fathers occupation          4425 non-null   object
 11  Displaced                   4425 non-null   object
 12  Educational special needs   4425 non-null   object
dtypes: object(13)
memory usage: 449.5+ KB
```

Double-click (or enter) to edit

```
','Mothers qualification','Fathers qualification','Mothers occupation','Fathers occupation','Displaced'
```

```
28                        5
30                        5
31                        4
19                        4
11                        4
23                        3
18                        3
21                        3
14                        2
25                        2
24                        2
27                        1
16                        1
17                        1
26                        1
Mother's occupation       1
Name: Mothers occupation, dtype: int64
10                     1010
8                       666
6                       516
5                       386
4                       384
9                       318
11                      266
7                       242
3                       197
2                       134
1                       128
12                       65
13                       19
44                       15
29                        8
36                        8
43                        6
35                        5
39                        4
16                        4
42                        3
31                        3
21                        3
26                        3
40                        3
17                        2
45                        2
30                        2
37                        2
20                        2
15                        2
41                        2
32                        1
38                        1
25                        1
27                        1
```

```
df['Application order'].value_counts()
```

```
1                    3026
2                     547
3                     309
4                     249
5                     154
6                     137
Application order       1
9                       1
0                       1
Name: Application order, dtype: int64
```

```
# check missing values in variables

df.isnull().sum()
```

```
Marital status                 0
Application mode               0
Application order              0
Course                         0
Daytime/evening attendance     0
Previous qualification         0
Nacionality                    0
Mothers qualification          0
Fathers qualification          0
Mothers occupation             0
Fathers occupation             0
Displaced                      0
Educational special needs      0
dtype: int64
```

```
X = df.drop(['Application order'], axis=1)

y = df['Application order']
```

```
# split X and y into training and testing sets

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 42)
```

```
# check the shape of X_train and X_test

X_train.shape, X_test.shape
```

```
((2964, 12), (1461, 12))
```

```
# check data types in X_train

X_train.dtypes
```

```
Marital status                 object
Application mode               object
Course                         object
Daytime/evening attendance     object
Previous qualification         object
Nacionality                    object
Mothers qualification          object
Fathers qualification          object
Mothers occupation             object
Fathers occupation             object
Displaced                      object
Educational special needs      object
dtype: object
```

```
X_train.head()
```

	Marital status	Application mode	Course	Daytime/evening attendance	Previous qualification	Nacionality	qua:
258	1	8	5	1	1	1	
3471	1	1	12	1	1	1	
386	1	1	10	1	1	1	
847	2	12	17	0	1	1	
4422	1	1	12	1	1	1	

Next steps: Generate code with X_train View recommended plots

```
# import category encoders
!pip install category_encoders
import category_encoders as ce
```

```
Collecting category_encoders
  Downloading category_encoders-2.6.3-py2.py3-none-any.whl (81 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 81.9/81.9 kB 1.8 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.10/dist-packages (from cate
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.10/dist-packages (fr
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from categ
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: pandas>=1.0.5 in /usr/local/lib/python3.10/dist-packages (from cate
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.10/dist-packages (from categ
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from panda
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.1->
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scik
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (fr
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from st
Installing collected packages: category_encoders
Successfully installed category_encoders-2.6.3
```

```
# encode variables with ordinal encoding


encoder = ce.OrdinalEncoder(cols=['Marital status', 'Application mode', 'Course','Daytime/evening atte




X_train = encoder.fit_transform(X_train)



X_test = encoder.transform(X_test)


X_train.head()
```

	Marital status	Application mode	Course	Daytime/evening attendance	Previous qualification	Nacionality	qua:
258	1	1	1	1	1	1	
3471	1	2	2	1	1	1	
386	1	2	3	1	1	1	
847	2	3	4	2	1	1	
4422	1	2	2	1	1	1	

Next steps: Generate code with X_train View recommended plots

```
X_test.head()
```

	Marital status	Application mode	Course	Daytime/evening attendance	Previous qualification	Nacionality	qua:
1257	1.0	1.0	3.0	1.0	1.0	1.0	
2572	1.0	2.0	2.0	1.0	1.0	1.0	
3741	1.0	10.0	7.0	1.0	1.0	1.0	
1068	1.0	2.0	10.0	1.0	1.0	1.0	
1732	1.0	2.0	15.0	1.0	1.0	1.0	

Next steps: Generate code with X_test View recommended plots

```
# import DecisionTreeClassifier

from sklearn.tree import DecisionTreeClassifier


# instantiate the DecisionTreeClassifier model with criterion gini index

clf_gini = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=0)


# fit the model
clf_gini.fit(X_train, y_train)
```

```
▾              DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3, random_state=0)
```

```
y_pred_gini = clf_gini.predict(X_test)
```

```
from sklearn.metrics import accuracy_score

print('Model accuracy score with criterion gini index: {0:0.4f}'. format(accuracy_score(y_test, y_pred
```

```
Model accuracy score with criterion gini index: 0.6872
```

```
y_pred_train_gini = clf_gini.predict(X_train)

y_pred_train_gini
```

```
array(['1', '1', '1', ..., '1', '1', '1'], dtype=object)
```

```
print('Training-set accuracy score: {0:0.4f}'. format(accuracy_score(y_train, y_pred_train_gini)))
```

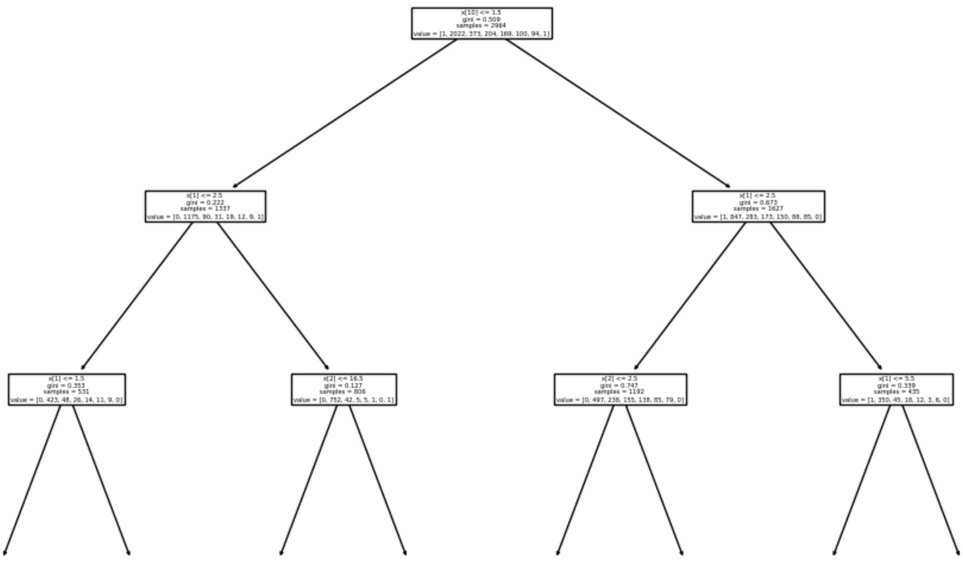
```
Training-set accuracy score: 0.6822
```

```
# print the scores on training and test set

print('Training set score: {:.4f}'.format(clf_gini.score(X_train, y_train)))

print('Test set score: {:.4f}'.format(clf_gini.score(X_test, y_test)))
```

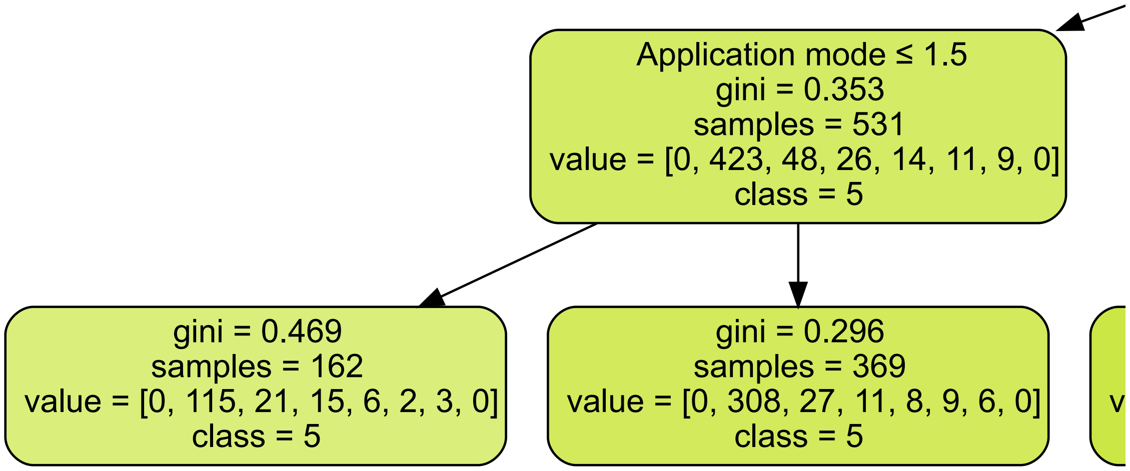
```
Training set score: 0.6822
Test set score: 0.6872
```

```
plt.figure(figsize=(12,8))

from sklearn import tree

tree.plot_tree(clf_gini.fit(X_train, y_train))
```

```
[Text(0.5, 0.875, 'x[10] <= 1.5\ngini = 0.509\nsamples = 2964\nvalue = [1, 2022,
373, 204, 169, 100, 94, 1]'),
 Text(0.25, 0.625, 'x[1] <= 2.5\ngini = 0.222\nsamples = 1337\nvalue = [0, 1175, 90,
31, 19, 12, 9, 1]'),
 Text(0.125, 0.375, 'x[1] <= 1.5\ngini = 0.353\nsamples = 531\nvalue = [0, 423, 48,
26, 14, 11, 9, 0]'),
 Text(0.0625, 0.125, 'gini = 0.469\nsamples = 162\nvalue = [0, 115, 21, 15, 6, 2, 3,
0]'),
 Text(0.1875, 0.125, 'gini = 0.296\nsamples = 369\nvalue = [0, 308, 27, 11, 8, 9, 6,
0]'),
 Text(0.375, 0.375, 'x[2] <= 16.5\ngini = 0.127\nsamples = 806\nvalue = [0, 752, 42,
5, 5, 1, 0, 1]'),
 Text(0.3125, 0.125, 'gini = 0.11\nsamples = 783\nvalue = [0, 738, 33, 5, 5, 1, 0,
1]'),
 Text(0.4375, 0.125, 'gini = 0.476\nsamples = 23\nvalue = [0, 14, 9, 0, 0, 0, 0,
0]'),
 Text(0.75, 0.625, 'x[1] <= 2.5\ngini = 0.673\nsamples = 1627\nvalue = [1, 847, 283,
173, 150, 88, 85, 0]'),
 Text(0.625, 0.375, 'x[2] <= 2.5\ngini = 0.747\nsamples = 1192\nvalue = [0, 497,
238, 155, 138, 85, 79, 0]'),
 Text(0.5625, 0.125, 'gini = 0.824\nsamples = 360\nvalue = [0, 83, 68, 64, 57, 39,
49, 0]'),
 Text(0.6875, 0.125, 'gini = 0.685\nsamples = 832\nvalue = [0, 414, 170, 91, 81, 46,
30, 0]'),
 Text(0.875, 0.375, 'x[1] <= 5.5\ngini = 0.339\nsamples = 435\nvalue = [1, 350, 45,
18, 12, 3, 6, 0]'),
 Text(0.8125, 0.125, 'gini = 0.05\nsamples = 195\nvalue = [1, 190, 4, 0, 0, 0, 0,
0]'),
 Text(0.9375, 0.125, 'gini = 0.517\nsamples = 240\nvalue = [0, 160, 41, 18, 12, 3,
6, 0]')]
```


x[10] <= 1.5
gini = 0.509
samples = 2964
value = [1, 2022, 373, 204, 169, 100, 94, 1]
x[1] <= 2.5
gini = 0.222
samples = 1337
value = [0, 1175, 90, 31, 19, 12, 9, 1]
x[1] <= 2.5
gini = 0.673
samples = 1627
value = [1, 847, 283, 173, 150, 88, 85, 0]
x[1] <= 1.5
gini = 0.353
samples = 531
value = [0, 423, 48, 26, 14, 11, 9, 0]
x[2] <= 16.5
gini = 0.127
samples = 806
value = [0, 752, 42, 5, 5, 1, 0, 1]
x[2] <= 2.5
gini = 0.747
samples = 1192
value = [0, 497, 238, 155, 138, 85, 79, 0]
x[1] <= 5.5
gini = 0.339
samples = 435
value = [1, 350, 45, 18, 12, 3, 6, 0]

```
import graphviz
dot_data = tree.export_graphviz(clf_gini, out_file=None,
                              feature_names=X_train.columns,
                              class_names=y_train,
                              filled=True, rounded=True,
                              special_characters=True)

graph = graphviz.Source(dot_data)

graph
```



```
# instantiate the DecisionTreeClassifier model with criterion entropy

clf_en = DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)


# fit the model
clf_en.fit(X_train, y_train)
```

```
▾                    DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)
```

```
y_pred_en = clf_en.predict(X_test)
```

```
from sklearn.metrics import accuracy_score

print('Model accuracy score with criterion entropy: {0:0.4f}'. format(accuracy_score(y_test, y_pred_en
```

```
Model accuracy score with criterion entropy: 0.6872
```

```
y_pred_train_en = clf_en.predict(X_train)

y_pred_train_en
```

```
array(['1', '1', '1', ..., '1', '1', '1'], dtype=object)
```

```
print('Training-set accuracy score: {0:0.4f}'. format(accuracy_score(y_train, y_pred_train_en)))
```

```
Training-set accuracy score: 0.6822
```

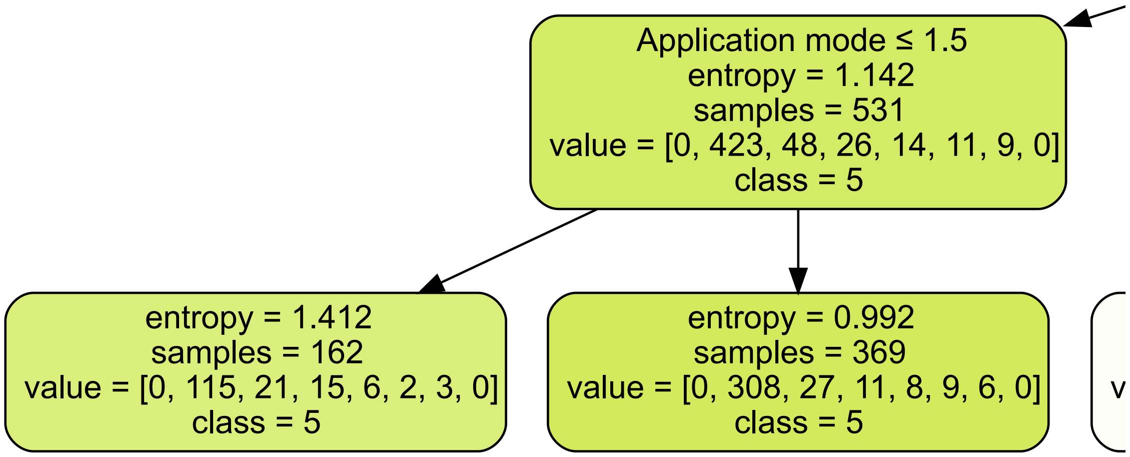
```
# print the scores on training and test set

print('Training set score: {:.4f}'.format(clf_en.score(X_train, y_train)))

print('Test set score: {:.4f}'.format(clf_en.score(X_test, y_test)))
```

```
Training set score: 0.6822
Test set score: 0.6872
```

```
plt.figure(figsize=(12,8))

from sklearn import tree

tree.plot_tree(clf_en.fit(X_train, y_train))
```

```
[Text(0.5, 0.875, 'x[1] <= 2.5\nentropy = 1.585\nsamples = 2964\nvalue = [1, 2022,
373, 204, 169, 100, 94, 1]'),
 Text(0.25, 0.625, 'x[10] <= 1.5\nentropy = 2.015\nsamples = 1723\nvalue = [0, 920,
286, 181, 152, 96, 88, 0]'),
 Text(0.125, 0.375, 'x[1] <= 1.5\nentropy = 1.142\nsamples = 531\nvalue = [0, 423,
48, 26, 14, 11, 9, 0]'),
 Text(0.0625, 0.125, 'entropy = 1.412\nsamples = 162\nvalue = [0, 115, 21, 15, 6, 2,
```

```
import graphviz
dot_data = tree.export_graphviz(clf_en, out_file=None,
                              feature_names=X_train.columns,
                              class_names=y_train,
                              filled=True, rounded=True,
                              special_characters=True)

graph = graphviz.Source(dot_data)

graph
```


Application mode ≤ 1.5
entropy = 1.142
samples = 531
value = [0, 423, 48, 26, 14, 11, 9, 0]
class = 5
entropy = 1.412
samples = 162
value = [0, 115, 21, 15, 6, 2, 3, 0]
class = 5
entropy = 0.992
samples = 369
value = [0, 308, 27, 11, 8, 9, 6, 0]
class = 5
v