Assignment submitted by: **Adeel Abdul Sakkeer**

Reg No.: 20BCE0910

ID3 and Cart

Using a play tennis dataset

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
import pydotplus
from sklearn.tree import export_graphviz
```

In [2]:
```
DATA_PATH='./data/playtennis.csv'
```

In [3]:
```
data = pd.read_csv(DATA_PATH,sep=',')
```

In [4]:
```
data.head()
```

Out[4]:

	PlayTennis	Outlook	Temperature	Humidity	Wind
0	No	Sunny	Hot	High	Weak
1	No	Sunny	Hot	High	Strong
2	Yes	Overcast	Hot	High	Weak
3	Yes	Rain	Mild	High	Weak
4	Yes	Rain	Cool	Normal	Weak

In [5]:
```
Y = data.pop('PlayTennis')
Y
```

Out[5]:
```
0       No
1       No
2      Yes
3      Yes
4      Yes
5       No
6      Yes
7       No
8      Yes
9      Yes
10     Yes
11     Yes
12     Yes
13      No
Name: PlayTennis, dtype: object
```

In [6]:
```
X = pd.get_dummies(data[data.columns])
X
```

Out[6]:

	Outlook_Overcast	Outlook_Rain	Outlook_Sunny	Temperature_Cool	Temperature_Hot	Temperature_Mild	Hun
0	0	0	1	0	1	0	

1	0	0	1	0	1	0
2	1	0	0	0	1	0
3	0	1	0	0	0	1
4	0	1	0	1	0	0
5	0	1	0	1	0	0
6	1	0	0	1	0	0
7	0	0	1	0	0	1
8	0	0	1	1	0	0
9	0	1	0	0	0	1
10	0	0	1	0	0	1
11	1	0	0	0	0	1
12	1	0	0	0	1	0
13	0	1	0	0	0	1

ID3

In [7]:
```
ID3Tree = DecisionTreeClassifier(criterion="entropy",random_state=17)
ID3Tree.fit(X,Y)
```

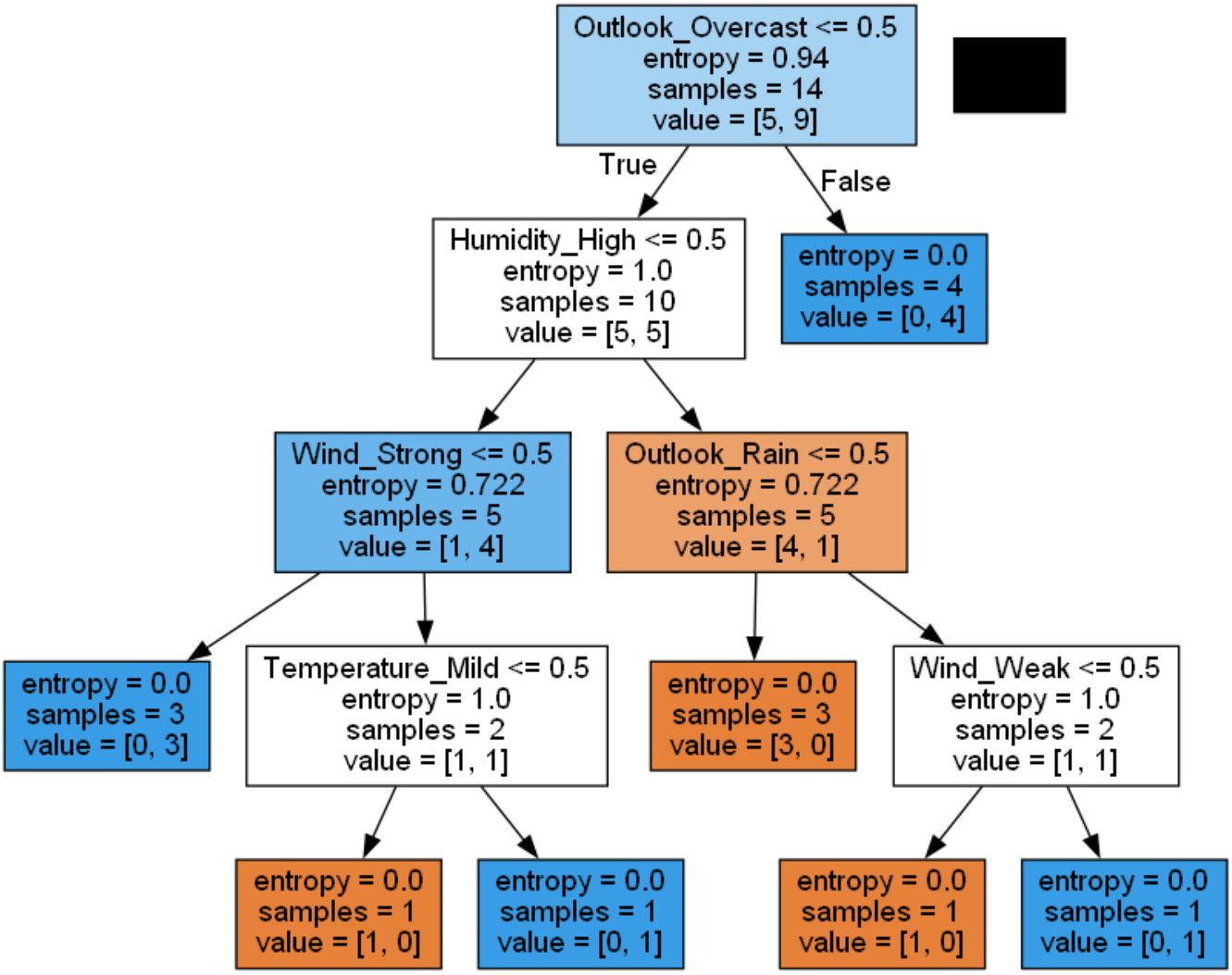
Out[7]:
```
DecisionTreeClassifier(criterion='entropy', random_state=17)
```

Exporting tree as a png

In [8]:
```
tree_str = export_graphviz(ID3Tree,feature_names=X.columns,filled=True,out_file=None)
graph = pydotplus.graph_from_dot_data(tree_str)
graph.write_png('ID3.png')
```

Out[8]:
```
True
```

Final ID3 Output



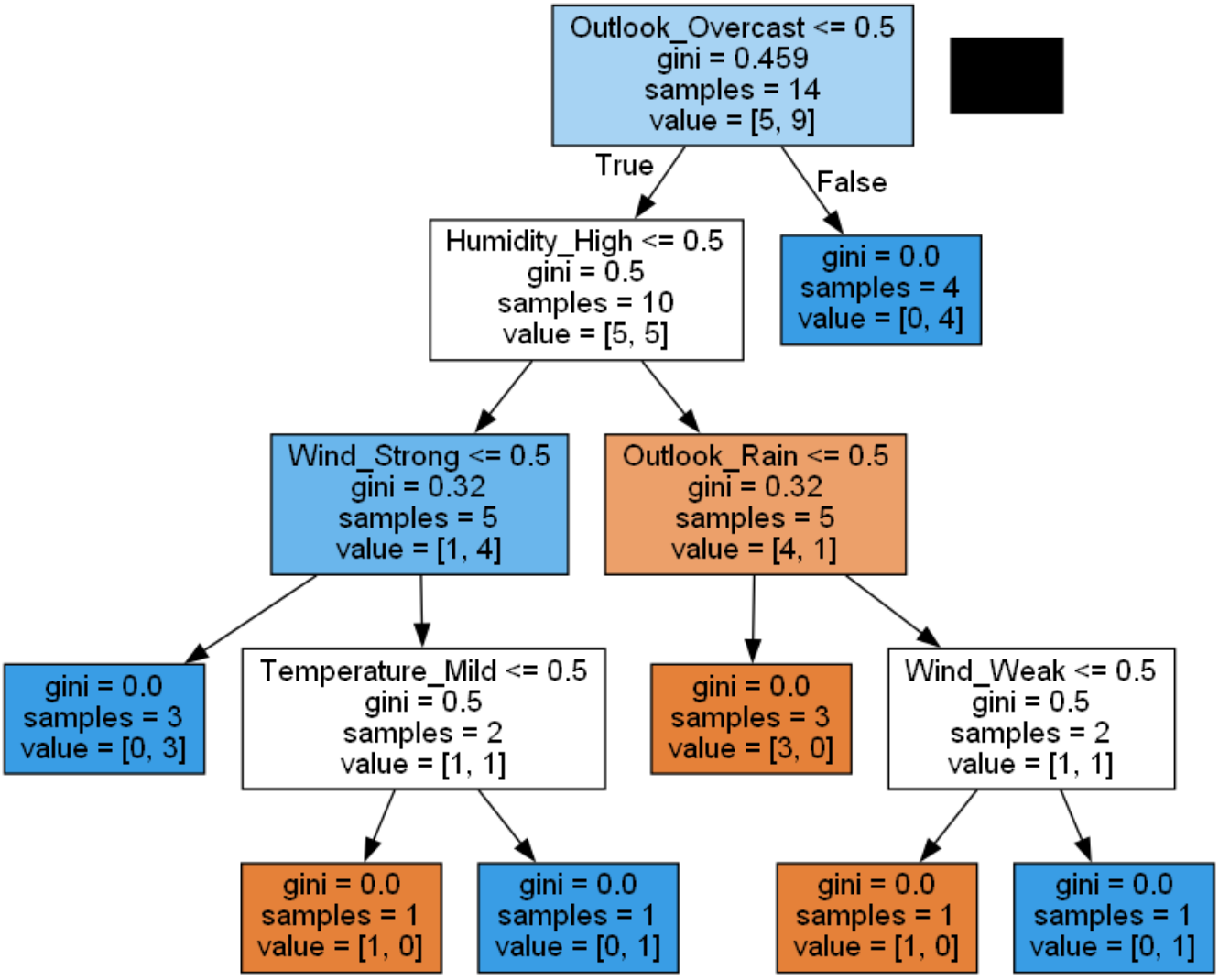
Gini

In [9]:
```
gini_tree = DecisionTreeClassifier(criterion='gini', random_state=17)
gini_tree.fit(X,Y)
```

Out[9]:
```
DecisionTreeClassifier(random_state=17)
```

In [10]:
```
tree_str = export_graphviz(gini_tree, feature_names=X.columns, filled=True, out_file=Non
graph = pydotplus.graph_from_dot_data(tree_str)
graph.write_png('gini.png')
```

Out[10]:
```
True
```


Outlook_Overcast <= 0.5
gini = 0.459
samples = 14
value = [5, 9]
True
False
Humidity_High <= 0.5
gini = 0.5
samples = 10
value = [5, 5]
gini = 0.0
samples = 4
value = [0, 4]
Wind_Strong <= 0.5
gini = 0.32
samples = 5
value = [1, 4]
Outlook_Rain <= 0.5
gini = 0.32
samples = 5
value = [4, 1]
gini = 0.0
samples = 3
value = [0, 3]
Temperature_Mild <= 0.5
gini = 0.5
samples = 2
value = [1, 1]
gini = 0.0
samples = 3
value = [3, 0]
Wind_Weak <= 0.5
gini = 0.5
samples = 2
value = [1, 1]
gini = 0.0
samples = 1
value = [1, 0]
gini = 0.0
samples = 1
value = [0, 1]
gini = 0.0
samples = 1
value = [1, 0]
gini = 0.0
samples = 1
value = [0, 1]

KNN

For this task I will use a knn to classify the MNIST dataset of digits

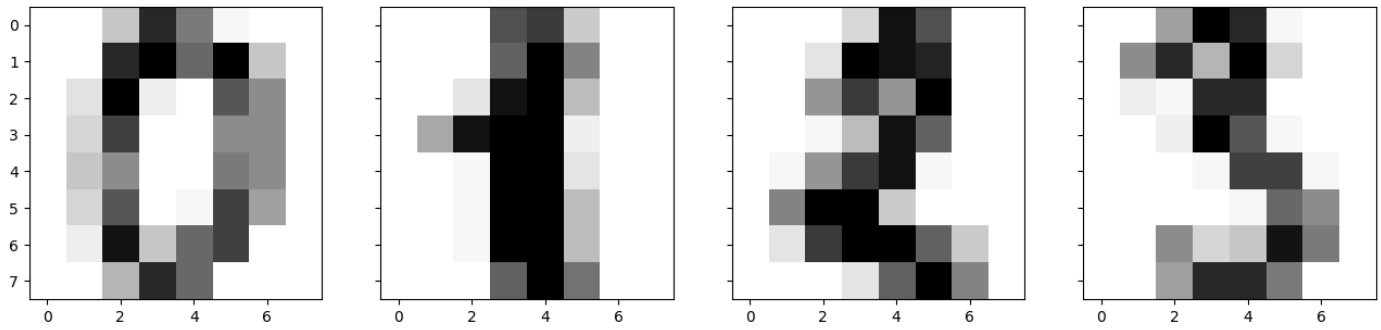
In [1]:
```
from sklearn.datasets import load_digits
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
```

In [2]:
```
data = load_digits()
X, y = data.data, data.target

X[0, :].reshape([8, 8])
```

Out[2]:
```
array([[ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.],
       [ 0.,  0., 13., 15., 10., 15.,  5.,  0.],
       [ 0.,  3., 15.,  2.,  0., 11.,  8.,  0.],
       [ 0.,  4., 12.,  0.,  0.,  8.,  8.,  0.],
       [ 0.,  5.,  8.,  0.,  0.,  9.,  8.,  0.],
       [ 0.,  4., 11.,  0.,  1., 12.,  7.,  0.],
       [ 0.,  2., 14.,  5., 10., 12.,  0.,  0.],
       [ 0.,  0.,  6., 13., 10.,  0.,  0.,  0.]])
```

In [3]:
```
f, axes = plt.subplots(1, 4, sharey=True, figsize=(16, 6))
for i in range(4):
    axes[i].imshow(X[i, :].reshape([8, 8]), cmap="Greys");
```



In [4]:
```
X_train, X_holdout, y_train, y_holdout = train_test_split(
    X, y, test_size=0.3, random_state=17
)
```

In [5]:
```
knn_pipe = Pipeline(
    [("scaler", StandardScaler()), ("knn", KNeighborsClassifier(n_neighbors=10))]
)
knn_pipe.fit(X_train, y_train);
```

In [6]:
```
knn_pred = knn_pipe.predict(X_holdout)
```

In [7]:
```
accuracy_score(y_holdout, knn_pred)
```

Out[7]:
```
0.975925925925926
```

Naive Bayes Classifier

Using a salary dataset

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.naive_bayes import GaussianNB
```

In [2]:
```
salary_train=pd.read_csv('./data/SalaryData_Train.csv')
salary_test=pd.read_csv('./data/SalaryData_Test.csv')
```

In [3]:
```
salary_train.tail()
```

Out[3]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex	capitalgai
30156	27	Private	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	
30157	40	Private	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	
30158	58	Private	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	
30159	22	Private	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	
30160	52	Self-emp-inc	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	1502

In [4]:
```
salary_test.tail()
```

Out[4]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex	capitalg
15055	33	Private	Bachelors	13	Never-married	Prof-specialty	Own-child	White	Male	
15056	39	Private	Bachelors	13	Divorced	Prof-specialty	Not-in-family	White	Female	
15057	38	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	White	Male	
15058	44	Private	Bachelors	13	Divorced	Adm-clerical	Own-child	Asian-Pac-Islander	Male	5
15059	35	Self-emp-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	

In [5]:
```
salary_train.columns, salary_test.columns
```

Out[5]:
```
(Index(['age', 'workclass', 'education', 'educationno', 'maritalstatus',
        'occupation', 'relationship', 'race', 'sex', 'capitalgain',
        'capitalloss', 'hoursperweek', 'native', 'Salary'],
       dtype='object'),
 Index(['age', 'workclass', 'education', 'educationno', 'maritalstatus',
        'occupation', 'relationship', 'race', 'sex', 'capitalgain',
```

```
       'capitalloss', 'hoursperweek', 'native', 'Salary'],
      dtype='object'))
```

In [6]:
```
string_columns=['workclass','education','maritalstatus','occupation','relationship','rac
```

In [7]:
```
label_encoder=preprocessing.LabelEncoder()
for i in string_columns:
    salary_train[i]=label_encoder.fit_transform(salary_train[i])
    salary_test[i]=label_encoder.fit_transform(salary_test[i])
```

In [8]:
```
col_names=list(salary_train.columns)
X_train=salary_train[col_names[0:13]]
Y_train=salary_train[col_names[13]]
X_test=salary_test[col_names[0:13]]
Y_test=salary_test[col_names[13]]
```

In [9]:
```
Gmodel=GaussianNB()
train_pred_gau=Gmodel.fit(X_train,Y_train).predict(X_train)
test_pred_gau=Gmodel.fit(X_train,Y_train).predict(X_test)
```

In [11]:
```
train_acc_gau=np.mean(train_pred_gau==Y_train)
test_acc_gau=np.mean(test_pred_gau==Y_test)
```

In [12]:
```
print('training accuracy: ',train_acc_gau)
print('testing accuracy: ',test_acc_gau)
```

```
training accuracy:  0.7953317197705646
testing accuracy:  0.7946879150066402
```