Experiment Title: 2.4	
Student Name: Pritam Kumar Dutta	**UID:** 20BCS3296
Branch: CSE	**Section/Group:** 606-A
Semester: 5	**Date of Performance:** Nov. 01, 2022
Subject Name: Machine Learning Lab	**Subject Code:** 20CSP-317

❖ **Aim/Overview of the practical:** Implement Decision Tree and compare the performance with Random Forest on any data set.

❖ **Code & Output:**

jupyter decision-trees-vs-random-forest Last Checkpoint: Last Tuesday at 10:42 AM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
sns.set_style("whitegrid")
plt.style.use("fivethirtyeight")
```

In [2]:
```
df = pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition.csv")
df.head()
```

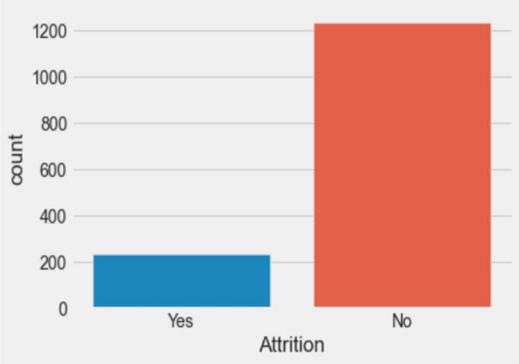
Out[2]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	RelationshipS
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	

5 rows × 35 columns

In [3]:
```
sns.countplot(x='Attrition', data=df)
```

Out[3]: <AxesSubplot:xlabel='Attrition', ylabel='count'>



In [4]:
```
df.drop(['EmployeeCount', 'EmployeeNumber', 'Over18', 'StandardHours'], axis="columns", inplace=True)

categorical_col = []
for column in df.columns:
    if df[column].dtype == object and len(df[column].unique()) <= 50:
        categorical_col.append(column)

df['Attrition'] = df.Attrition.astype("category").cat.codes
```

In [5]:
```
categorical_col.remove('Attrition')
```

In [6]:
```
# Transform categorical data into dummies
# categorical_col.remove("Attrition")
# data = pd.get_dummies(df, columns=categorical_col)
# data.info()
from sklearn.preprocessing import LabelEncoder

label = LabelEncoder()
for column in categorical_col:
    df[column] = label.fit_transform(df[column])
```

In [7]:
```
from sklearn.model_selection import train_test_split

X = df.drop('Attrition', axis=1)
y = df.Attrition

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [8]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

def print_score(clf, X_train, y_train, X_test, y_test, train=True):
    if train:
        pred = clf.predict(X_train)
        clf_report = pd.DataFrame(classification_report(y_train, pred, output_dict=True))
        print("Train Result:\n================================================")
        print(f"Accuracy Score: {accuracy_score(y_train, pred) * 100:.2f}%")
        print("_______________________________________________")
        print(f"CLASSIFICATION REPORT:\n{clf_report}")
        print("_______________________________________________")
        print(f"Confusion Matrix: \n {confusion_matrix(y_train, pred)}\n")

    elif train==False:
        pred = clf.predict(X_test)
        clf_report = pd.DataFrame(classification_report(y_test, pred, output_dict=True))
        print("Test Result:\n================================================")
        print(f"Accuracy Score: {accuracy_score(y_test, pred) * 100:.2f}%")
        print("_______________________________________________")
        print(f"CLASSIFICATION REPORT:\n{clf_report}")
        print("_______________________________________________")
        print(f"Confusion Matrix: \n {confusion_matrix(y_test, pred)}\n")
```

In [9]:
```
from sklearn.tree import DecisionTreeClassifier

tree_clf = DecisionTreeClassifier(random_state=42)
tree_clf.fit(X_train, y_train)

print_score(tree_clf, X_train, y_train, X_test, y_test, train=True)
print_score(tree_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Train Result:
================================================
Accuracy Score: 100.00%
_______________________________________________
CLASSIFICATION REPORT:
                0      1  accuracy  macro avg  weighted avg
precision     1.0    1.0       1.0        1.0           1.0
recall        1.0    1.0       1.0        1.0           1.0
f1-score      1.0    1.0       1.0        1.0           1.0
support     853.0  176.0       1.0     1029.0        1029.0
_______________________________________________
Confusion Matrix:
 [[853   0]
 [  0 176]]

Test Result:
================================================
Accuracy Score: 77.78%
_______________________________________________
CLASSIFICATION REPORT:
                     0          1  accuracy   macro avg  weighted avg
precision     0.887363   0.259740  0.777778    0.573551      0.800549
recall        0.850000   0.327869  0.777778    0.588934      0.777778
f1-score      0.868280   0.289855  0.777778    0.579067      0.788271
support     380.000000  61.000000  0.777778  441.000000    441.000000
_______________________________________________
Confusion Matrix:
 [[323  57]
 [ 41  20]]
```

In [10]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

params = {
    "criterion":("gini", "entropy"),
    "splitter":("best", "random"),
    "max_depth":(list(range(1, 20))),
    "min_samples_split":[2, 3, 4],
    "min_samples_leaf":list(range(1, 20)),
}


tree_clf = DecisionTreeClassifier(random_state=42)
tree_cv = GridSearchCV(tree_clf, params, scoring="accuracy", n_jobs=-1, verbose=1, cv=3)
tree_cv.fit(X_train, y_train)
best_params = tree_cv.best_params_
print(f"Best paramters: {best_params})")

tree_clf = DecisionTreeClassifier(**best_params)
tree_clf.fit(X_train, y_train)
print_score(tree_clf, X_train, y_train, X_test, y_test, train=True)
print_score(tree_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Fitting 3 folds for each of 4332 candidates, totalling 12996 fits
Best paramters: {'criterion': 'entropy', 'max_depth': 6, 'min_samples_leaf': 10, 'min_samples_split': 2, 'splitter': 'best'})
Train Result:
================================================
Accuracy Score: 88.82%

________________________________________________
CLASSIFICATION REPORT:
                    0           1  accuracy    macro avg  weighted avg
precision    0.906388    0.752066  0.888241     0.829227      0.879993
recall       0.964830    0.517045  0.888241     0.740938      0.888241
f1-score     0.934696    0.612795  0.888241     0.773745      0.879638
support    853.000000  176.000000  0.888241  1029.000000   1029.000000
________________________________________________
Confusion Matrix:
 [[823  30]
 [ 85  91]]

Test Result:
================================================
Accuracy Score: 85.49%

________________________________________________
CLASSIFICATION REPORT:
                    0          1  accuracy   macro avg  weighted avg
precision    0.893035   0.461538  0.854875    0.677287      0.833349
recall       0.944737   0.295082  0.854875    0.619909      0.854875
f1-score     0.918159   0.360000  0.854875    0.639079      0.840953
support    380.000000  61.000000  0.854875  441.000000    441.000000
________________________________________________
Confusion Matrix:
 [[359  21]
 [ 43  18]]
```

In [15]:
```
from IPython.display import Image
from six import StringIO
from sklearn.tree import export_graphviz

features = list(df.columns)
features.remove("Attrition")
```

In [16]:
```
from sklearn.ensemble import RandomForestClassifier

rf_clf = RandomForestClassifier(n_estimators=100)
rf_clf.fit(X_train, y_train)

print_score(rf_clf, X_train, y_train, X_test, y_test, train=True)
print_score(rf_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Train Result:
================================================
Accuracy Score: 100.00%

_______________________________________________
CLASSIFICATION REPORT:
                0      1  accuracy  macro avg  weighted avg
precision     1.0    1.0       1.0        1.0           1.0
recall        1.0    1.0       1.0        1.0           1.0
f1-score      1.0    1.0       1.0        1.0           1.0
support     853.0  176.0       1.0     1029.0        1029.0
_______________________________________________
Confusion Matrix:
 [[853   0]
 [  0 176]]

Test Result:
================================================
Accuracy Score: 87.07%

_______________________________________________
CLASSIFICATION REPORT:
                    0          1  accuracy   macro avg  weighted avg
precision    0.872979   0.750000  0.870748    0.811490      0.855968
recall       0.994737   0.098361  0.870748    0.546549      0.870748
f1-score     0.929889   0.173913  0.870748    0.551901      0.825321
support    380.000000  61.000000  0.870748  441.000000    441.000000
_______________________________________________
Confusion Matrix:
 [[378   2]
 [ 55   6]]
```

In [17]:

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV

n_estimators = [int(x) for x in np.linspace(start=200, stop=2000, num=10)]
max_features = ['auto', 'sqrt']
max_depth = [int(x) for x in np.linspace(10, 110, num=11)]
max_depth.append(None)
min_samples_split = [2, 5, 10]
min_samples_leaf = [1, 2, 4]
bootstrap = [True, False]

random_grid = {'n_estimators': n_estimators, 'max_features': max_features,
               'max_depth': max_depth, 'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf, 'bootstrap': bootstrap}

rf_clf = RandomForestClassifier(random_state=42)

rf_cv = RandomizedSearchCV(estimator=rf_clf, scoring='f1',param_distributions=random_grid, n_iter=100, cv=3,
                           verbose=2, random_state=42, n_jobs=-1)

rf_cv.fit(X_train, y_train)
rf_best_params = rf_cv.best_params_
print(f"Best paramters: {rf_best_params})")

rf_clf = RandomForestClassifier(**rf_best_params)
rf_clf.fit(X_train, y_train)

print_score(rf_clf, X_train, y_train, X_test, y_test, train=True)
print_score(rf_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits
Best paramters: {'n_estimators': 400, 'min_samples_split': 5, 'min_samples_leaf': 1, 'max_features': 'auto', 'max_depth': 90,
'bootstrap': False})
Train Result:
================================================
Accuracy Score: 100.00%
________________________________________________
CLASSIFICATION REPORT:
                0      1  accuracy  macro avg  weighted avg
precision     1.0    1.0       1.0        1.0           1.0
recall        1.0    1.0       1.0        1.0           1.0
f1-score      1.0    1.0       1.0        1.0           1.0
support     853.0  176.0       1.0     1029.0        1029.0
________________________________________________
Confusion Matrix:
 [[853   0]
 [  0 176]]

Test Result:
================================================
Accuracy Score: 86.39%
________________________________________________
CLASSIFICATION REPORT:
                    0          1  accuracy   macro avg  weighted avg
precision    0.873832   0.538462  0.863946    0.706147      0.827443
recall       0.984211   0.114754  0.863946    0.549482      0.863946
f1-score     0.925743   0.189189  0.863946    0.557466      0.823861
support    380.000000  61.000000  0.863946  441.000000    441.000000
________________________________________________
Confusion Matrix:
 [[374   6]
 [ 54   7]]
```

In [14]:

```
n_estimators = [100, 500, 1000, 1500]
max_features = ['auto', 'sqrt']
max_depth = [2, 3, 5]
max_depth.append(None)
min_samples_split = [2, 5, 10]
min_samples_leaf = [1, 2, 4, 10]
bootstrap = [True, False]

params_grid = {'n_estimators': n_estimators, 'max_features': max_features,
               'max_depth': max_depth, 'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf, 'bootstrap': bootstrap}

rf_clf = RandomForestClassifier(random_state=42)

rf_cv = GridSearchCV(rf_clf, params_grid, scoring="f1", cv=3, verbose=2, n_jobs=-1)


rf_cv.fit(X_train, y_train)
best_params = rf_cv.best_params_
print(f"Best parameters: {best_params}")

rf_clf = RandomForestClassifier(**best_params)
rf_clf.fit(X_train, y_train)

print_score(rf_clf, X_train, y_train, X_test, y_test, train=True)
print_score(rf_clf, X_train, y_train, X_test, y_test, train=False)
```

```
Fitting 3 folds for each of 768 candidates, totalling 2304 fits
Best parameters: {'bootstrap': False, 'max_depth': None, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2,
'n_estimators': 1500}
Train Result:
================================================
Accuracy Score: 100.00%
________________________________________________
CLASSIFICATION REPORT:
              0      1  accuracy  macro avg  weighted avg
precision    1.0    1.0       1.0        1.0           1.0
recall       1.0    1.0       1.0        1.0           1.0
f1-score     1.0    1.0       1.0        1.0           1.0
support    853.0  176.0       1.0     1029.0        1029.0
________________________________________________
Confusion Matrix:
 [[853   0]
 [  0 176]]

Test Result:
================================================
Accuracy Score: 85.94%
________________________________________________
CLASSIFICATION REPORT:
                    0          1  accuracy   macro avg  weighted avg
precision    0.869767   0.454545   0.85941    0.662156      0.812333
recall       0.984211   0.081967   0.85941    0.533089      0.859410
f1-score     0.923457   0.138889   0.85941    0.531173      0.814934
support    380.000000  61.000000   0.85941  441.000000    441.000000
________________________________________________
Confusion Matrix:
 [[374   6]
 [ 56   5]]
```

❖ Learning outcomes (What I have learnt):

1. We learned about data analysis and data handling in python.
2. We learned about various basic functions and libraries required for data analysis using python.
3. We learned to implement decision tree algorithm and random forest algorithm on any dataset in python.
4. We learned to compare accuracy of the decision tree and random forest algorithms.
5. We learned that random forest algorithm is a better fit than decision tree.