In [59]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:

```
df = pd.read_csv("HR-Employee-Attrition.csv")
df.shape
```

Out[2]:

```
(1470, 35)
```

In [3]:

```
df.head()
```

Out[3]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educ
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Lif
1	49	No	Travel_Frequently	279	Research & Development	8	1	Lif
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Lif
4	27	No	Travel_Rarely	591	Research & Development	2	1	

5 rows × 35 columns

In [4]:

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

In [5]:

```
le = LabelEncoder()
```

In [6]:

```
le_count = 0
for col in df.columns[1:]:
    if df[col].dtype == 'object':
        if len(list(df[col].unique())) <= 2:
            le.fit(df[col])
            df[col] = le.transform(df[col])
            le_count += 1
print('{} columns were label encoded.'.format(le_count))
```

```
4 columns were label encoded.
```

In [7]:

```
df.head()
```

Out[7]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educ
0	41	1	Travel_Rarely	1102	Sales	1	2	Lif
1	49	0	Travel_Frequently	279	Research & Development	8	1	Lif
2	37	1	Travel_Rarely	1373	Research & Development	2	2	
3	33	0	Travel_Frequently	1392	Research & Development	3	4	Lif
4	27	0	Travel_Rarely	591	Research & Development	2	1	

5 rows × 35 columns

In [8]:

```
df = pd.get_dummies(df, drop_first=True)
```

In [9]:

```
df.dtypes
```

Out[9]:

```
Age                                  int64
Attrition                            int64
DailyRate                            int64
DistanceFromHome                     int64
Education                            int64
EmployeeCount                        int64
EmployeeNumber                       int64
EnvironmentSatisfaction              int64
Gender                               int64
HourlyRate                           int64
JobInvolvement                       int64
JobLevel                             int64
JobSatisfaction                      int64
MonthlyIncome                        int64
MonthlyRate                          int64
NumCompaniesWorked                   int64
Over18                               int64
OverTime                             int64
PercentSalaryHike                    int64
PerformanceRating                    int64
RelationshipSatisfaction             int64
StandardHours                        int64
StockOptionLevel                     int64
TotalWorkingYears                    int64
TrainingTimesLastYear                int64
WorkLifeBalance                      int64
YearsAtCompany                       int64
YearsInCurrentRole                   int64
YearsSinceLastPromotion              int64
YearsWithCurrManager                 int64
BusinessTravel_Travel_Frequently     uint8
BusinessTravel_Travel_Rarely         uint8
Department_Research & Development    uint8
Department_Sales                     uint8
EducationField_Life Sciences         uint8
EducationField_Marketing             uint8
EducationField_Medical               uint8
EducationField_Other                 uint8
EducationField_Technical Degree      uint8
JobRole_Human Resources              uint8
JobRole_Laboratory Technician        uint8
JobRole_Manager                      uint8
JobRole_Manufacturing Director       uint8
JobRole_Research Director            uint8
JobRole_Research Scientist           uint8
JobRole_Sales Executive              uint8
JobRole_Sales Representative         uint8
MaritalStatus_Married                uint8
MaritalStatus_Single                 uint8
dtype: object
```

In [10]:

```
df.head()
```

Out[10]:

	Age	Attrition	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber
0	41	1	1102	1	2	1	1
1	49	0	279	8	1	1	2
2	37	1	1373	2	2	1	4
3	33	0	1392	3	4	1	5
4	27	0	591	2	1	1	7

5 rows × 49 columns

In [11]:

```
target = df['Attrition']
```

In [12]:

```
df = df.drop(columns=['Attrition'])
```

In [13]:

```
target.shape
```

Out[13]:

```
(1470,)
```

In [14]:

```
df.shape
```

Out[14]:

```
(1470, 48)
```

In [15]:

```
df = df.drop(columns =['EmployeeCount','EmployeeNumber','StandardHours', 'Over18'])
```

In [16]:

```
df.shape
```

Out[16]:

```
(1470, 44)
```

In [17]:

```
df.head()
```

Out[17]:

	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	Gender	HourlyRate
0	41	1102	1	2	2	0	94
1	49	279	8	1	3	1	61
2	37	1373	2	2	4	1	92
3	33	1392	3	4	4	0	56
4	27	591	2	1	1	1	40

5 rows × 44 columns

In [18]:

```
target.value_counts(normalize = True)
```

Out[18]:

```
0    0.838776
1    0.161224
Name: Attrition, dtype: float64
```

In [19]:

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

In [20]:

```
X_train, X_test, y_train, y_test = train_test_split(df, target, test_size=0.25, ranc
```

In [21]:

```
X_train.shape, y_train.shape
```

Out[21]:

```
((1102, 44), (1102,))
```

In [22]:

```
X_test.shape, y_test.shape
```

Out[22]:

```
((368, 44), (368,))
```

In [23]:

```
y_train.value_counts(normalize=True)
```

Out[23]:

```
0    0.838475
1    0.161525
Name: Attrition, dtype: float64
```

In [24]:

```
y_test.value_counts(normalize=True)
```

Out[24]:

```
0    0.839674
1    0.160326
Name: Attrition, dtype: float64
```

In [25]:

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

In [30]:

```
tree = DecisionTreeClassifier(criterion='gini')
```

In [31]:

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

Out[31]:

```
DecisionTreeClassifier()
```

In [32]:

```
tree.score(X_train, y_train)
```

Out[32]:

```
1.0
```

In [33]:

```
tree.score(X_test, y_test)
```

Out[33]:

```
0.7527173913043478
```

K-Fold Cross validation

In [34]:

```
from sklearn.model_selection import cross_val_score
```

In [36]:

```
performance = cross_val_score(tree, X_train, y_train, cv=5, scoring='accuracy')
```

In [38]:

```
performance.mean()
```

Out[38]:

```
0.7804319210201563
```

In [ ]:

```

```

Feature Importance

In [44]:

```
X_train.columns
```

Out[44]:

```
Index(['Age', 'DailyRate', 'DistanceFromHome', 'Education',
       'EnvironmentSatisfaction', 'Gender', 'HourlyRate', 'JobInvolvem
ent',
       'JobLevel', 'JobSatisfaction', 'MonthlyIncome', 'MonthlyRate',
       'NumCompaniesWorked', 'OverTime', 'PercentSalaryHike',
       'PerformanceRating', 'RelationshipSatisfaction', 'StockOptionLe
vel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalanc
e',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotio
n',
       'YearsWithCurrManager', 'BusinessTravel_Travel_Frequently',
       'BusinessTravel_Travel_Rarely', 'Department_Research & Developm
ent',
       'Department_Sales', 'EducationField_Life Sciences',
       'EducationField_Marketing', 'EducationField_Medical',
       'EducationField_Other', 'EducationField_Technical Degree',
       'JobRole_Human Resources', 'JobRole_Laboratory Technician',
       'JobRole_Manager', 'JobRole_Manufacturing Director',
       'JobRole_Research Director', 'JobRole_Research Scientist',
       'JobRole_Sales Executive', 'JobRole_Sales Representative',
       'MaritalStatus_Married', 'MaritalStatus_Single'],
      dtype='object')
```

In [47]:

```
importances = tree.feature_importances_
```

In [52]:

```
importances
```

Out[52]:

```
array([0.07126337, 0.06968948, 0.03765089, 0.01134253, 0.02608627,
       0.01134505, 0.04032249, 0.01391192, 0.        , 0.025982  ,
       0.08525609, 0.05454717, 0.03737314, 0.05721149, 0.02518967,
       0.        , 0.01769795, 0.03809493, 0.11289118, 0.03276954,
       0.01862722, 0.03412719, 0.0199655 , 0.0274056 , 0.00574305,
       0.00290343, 0.01898156, 0.00335011, 0.00446682, 0.        ,
       0.        , 0.0114716 , 0.00161808, 0.00908826, 0.        ,
       0.00446682, 0.        , 0.        , 0.00714691, 0.03035406,
       0.0089935 , 0.00368587, 0.        , 0.01897926])
```

In [53]:

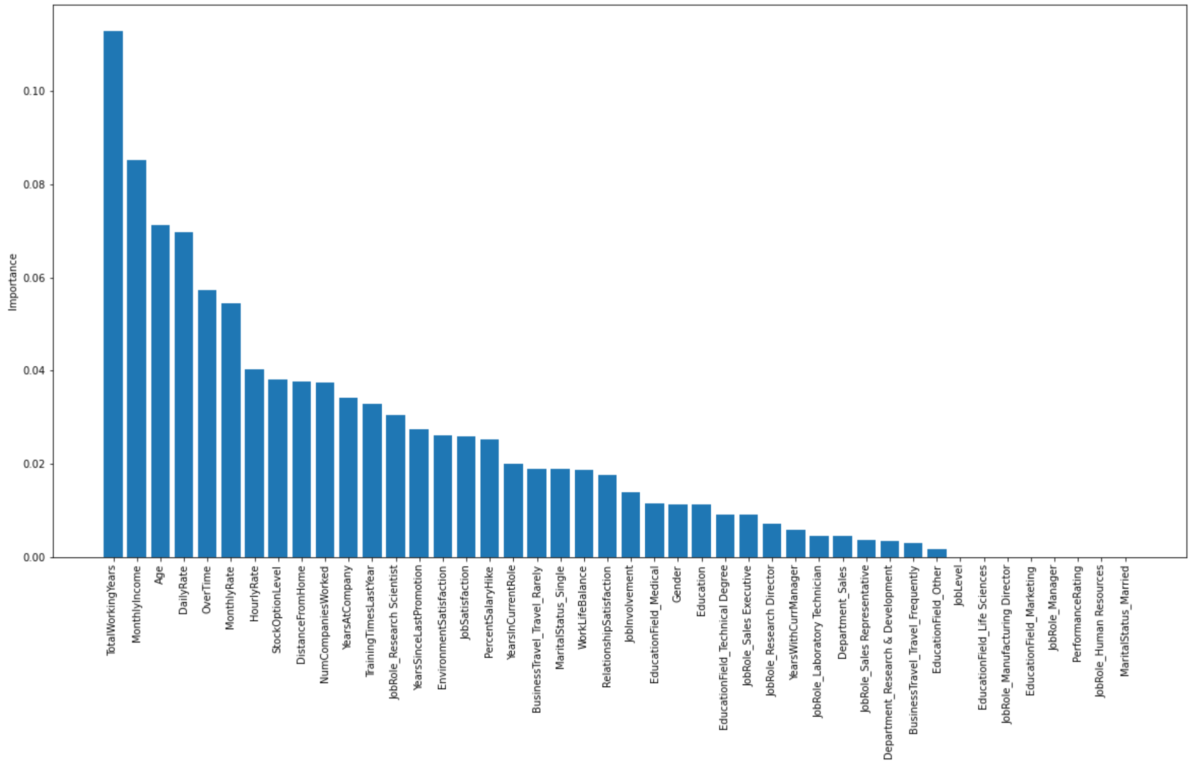
```
indices = importances.argsort()[::-1]
```

In [57]:

```
names = [X_train.columns[i] for i in indices]
```

In [69]:

```
plt.figure(figsize=(20,10))
plt.bar(range(44), importances[indices])
plt.xticks(range(44), names , rotation=90)
plt.ylabel("Importance")
plt.show()
```



In [84]:

```
features = pd.DataFrame({'feature':names,"importance":importances[indices]})
```

In [85]:

```
features.head(10)
```

Out[85]:

	feature	importance
0	TotalWorkingYears	0.112891
1	MonthlyIncome	0.085256
2	Age	0.071263
3	DailyRate	0.069689
4	OverTime	0.057211
5	MonthlyRate	0.054547
6	HourlyRate	0.040322
7	StockOptionLevel	0.038095
8	DistanceFromHome	0.037651
9	NumCompaniesWorked	0.037373

In [ ]:

```

```

Bias Variance TradeOff

max_depth

In [177]:

```
model = DecisionTreeClassifier(criterion='gini', max_depth=2)
```

In [178]:

```
model.fit(X_train, y_train)
```

Out[178]:

```
DecisionTreeClassifier(max_depth=2)
```

In [179]:

```
model.score(X_train, y_train)
```

Out[179]:

```
0.852994555353902
```

In [180]:

```
model.score(X_test, y_test)
```

Out[180]:

```
0.8369565217391305
```

Visualization of DT

In [149]:

```
!pip install graphviz
```

```
Collecting graphviz
  Downloading graphviz-0.19.1-py3-none-any.whl (46 kB)
     |████████████████████████████████| 46 kB 1.3 MB/s eta 0:00:011
Installing collected packages: graphviz
Successfully installed graphviz-0.19.1
```

In [154]:

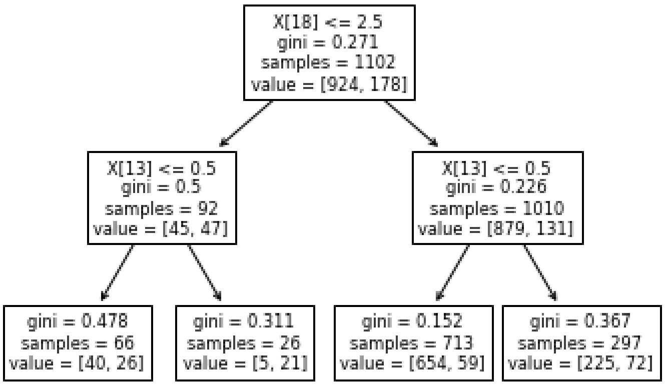
```
from sklearn import tree
```

In [158]:

```
import graphviz
```

In [181]:

```
tree.plot_tree(model)
plt.show()
```



In [192]:

```
# dot_data = tree.export_graphviz(model, out_file='abc.pdf',
#                      feature_names=X_train.columns,
#                      class_names=['0', '1'],
#                      filled=True, rounded=True,
#                      special_characters=True)
# graph = graphviz.Source(dot_data)
# graph
```

In [ ]:

```

```

In [182]:

```
model_text = tree.export_text(model)
print(model_text)
```

```
|--- feature_18 <= 2.50
|   |--- feature_13 <= 0.50
|   |   |--- class: 0
|   |--- feature_13 >  0.50
|   |   |--- class: 1
|--- feature_18 >  2.50
|   |--- feature_13 <= 0.50
|   |   |--- class: 0
|   |--- feature_13 >  0.50
|   |   |--- class: 0
```

Confusion Matrix

In [187]:

```
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
```

In [184]:

```
y_pred = model.predict(X_test)
```

In [190]:

```
confusion_matrix(y_test, y_pred)
```

Out[190]:

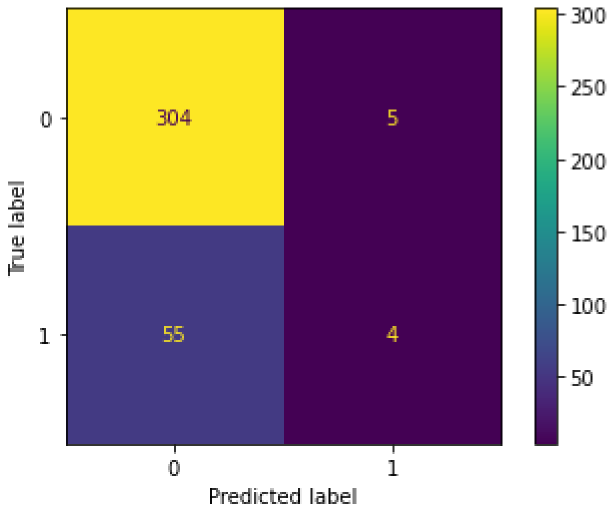
```
array([[304,   5],
       [ 55,   4]])
```

In [189]:

```
plot_confusion_matrix(model, X_test, y_test)
```

Out[189]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f
9c1b0681c0>
```



Hyper-parameter Tuning

In [195]:

```
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
```

In [203]:

```
params = {
    'criterion': ['entropy', 'gini'],
    'max_depth' : list(range(1, 11)),
    'min_samples_split' : list(range(1, 20)),
    'min_samples_leaf': list(range(1,5))
}
```

In [204]:

```
tree_clf = DecisionTreeClassifier()
```

In [208]:

```
grid_cv = GridSearchCV(tree_clf, params, scoring='accuracy', n_jobs=-1, cv=3, verbos
```

In [209]:

```
grid_cv.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 1520 candidates, totalling 4560 fits
```

```
/Users/mohit/opt/anaconda3/lib/python3.8/site-packages/sklearn/model_s
election/_search.py:918: UserWarning: One or more of the test scores a
re non-finite: [       nan 0.83756861 0.83756861 ... 0.83121816 0.8303
1236 0.83212396]
  warnings.warn(
```

Out[209]:

```
GridSearchCV(cv=3, estimator=DecisionTreeClassifier(), n_jobs=-1,
             param_grid={'criterion': ['entropy', 'gini'],
                         'max_depth': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
                         'min_samples_leaf': [1, 2, 3, 4],
                         'min_samples_split': [1, 2, 3, 4, 5, 6, 7, 8,
9, 10,
                                               11, 12, 13, 14, 15, 16,
17, 18,
                                               19]},
             scoring='accuracy', verbose=1)
```

In [210]:

```
grid_cv.best_score_
```

Out[210]:

```
0.8575356395371796
```

In [211]:

```
grid_cv.best_params_
```

Out[211]:

```
{'criterion': 'gini',
 'max_depth': 3,
 'min_samples_leaf': 1,
 'min_samples_split': 2}
```

In [ ]:

```

```

In [213]:

```
model_final = DecisionTreeClassifier(**grid_cv.best_params_)
```

In [216]:

```
model_final.fit(X_train, y_train)
```

Out[216]:

```
DecisionTreeClassifier(max_depth=3)
```

In [217]:

```
model_final.score(X_train, y_train)
```

Out[217]:

```
0.8638838475499092
```

In [218]:

```
model_final.score(X_test, y_test)
```

Out[218]:

```
0.8288043478260869
```

RandomizedSearch

In [ ]:

```
RandomizedSearchCV()
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: