Employee Turnover Analytics.

Course-end Project 4

DESCRIPTION

Project Statement:

Portobello Tech is an app innovator that has devised an intelligent way of predicting employee turnover within the company. It periodically evaluates employees' work details including the number of projects they worked upon, average monthly working hours, time spent in the company, promotions in the last 5 years, and salary level.

Data from prior evaluations show the employee's satisfaction at the workplace. The data could be used to identify patterns in work style and their interest to continue to work in the company.

The HR Department owns the data and uses it to predict employee turnover. Employee turnover refers to the total number of workers who leave a company over a certain time period.

As the ML Developer assigned to the HR Department, you have been asked to create ML Programs to

Perform data quality check by checking for missing values if any.

Understand what factors contributed most to employee turnover by EDA.

Perform clustering of Employees who left based on their satisfaction and evaluation.

Handle the left Class Imbalance using SMOTE technique.

Perform k-fold cross-validation model training and evaluate performance.

Identify the best model and justify the evaluation metrics used.

Suggest various retention strategies for targeted employees.

```
# De-facto imports
from __future__ import print_function
%matplotlib inline
import os
import warnings
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as image
import pandas as pd
import pandas_profiling
```

```
plt.style.use("ggplot")
warnings.simplefilter("ignore")
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

```
plt.rcParams['figure.figsize'] = (12,8)
```

```
emp = pd.read_csv('/content/drive/MyDrive/Projects/Employee-Turnover-
using-ML-Trees-Forests-master/employee_data.csv')
emp.head()
```

```
   satisfaction_level  last_evaluation  number_project
average_montly_hours  \
0                0.38             0.53               2
157
1                0.80             0.86               5
262
2                0.11             0.88               7
272
3                0.72             0.87               5
223
4                0.37             0.52               2
159

   time_spend_company  Work_accident  quit  promotion_last_5years
department  \
0                   3              0     1                      0
sales
1                   6              0     1                      0
sales
2                   4              0     1                      0
sales
3                   5              0     1                      0
sales
4                   3              0     1                      0
sales

   salary
0     low
1  medium
2  medium
3     low
4     low
```

```
!pip install -U pandas-profiling
```

```
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pandas-profiling in
/usr/local/lib/python3.8/dist-packages (3.6.3)
Requirement already satisfied: jinja2<3.2,>=2.11.1 in
/usr/local/lib/python3.8/dist-packages (from pandas-profiling)
(2.11.3)
Requirement already satisfied: htmlmin==0.1.12 in
/usr/local/lib/python3.8/dist-packages (from pandas-profiling)
(0.1.12)
Requirement already satisfied: multimethod<1.10,>=1.4 in
/usr/local/lib/python3.8/dist-packages (from pandas-profiling) (1.9.1)
Requirement already satisfied: statsmodels<0.14,>=0.13.2 in
/usr/local/lib/python3.8/dist-packages (from pandas-profiling)
(0.13.5)
Requirement already satisfied: matplotlib<3.7,>=3.2 in
/usr/local/lib/python3.8/dist-packages (from pandas-profiling) (3.2.2)
Requirement already satisfied: visions[type_image_path]==0.7.5 in
/usr/local/lib/python3.8/dist-packages (from pandas-profiling) (0.7.5)
Requirement already satisfied: phik<0.13,>=0.11.1 in
/usr/local/lib/python3.8/dist-packages (from pandas-profiling)
(0.12.3)
Requirement already satisfied: seaborn<0.13,>=0.10.1 in
/usr/local/lib/python3.8/dist-packages (from pandas-profiling)
(0.11.2)
Requirement already satisfied: pydantic<1.11,>=1.8.1 in
/usr/local/lib/python3.8/dist-packages (from pandas-profiling)
(1.10.4)
Requirement already satisfied: requests<2.29,>=2.24.0 in
/usr/local/lib/python3.8/dist-packages (from pandas-profiling)
(2.25.1)
Requirement already satisfied: PyYAML<6.1,>=5.0.0 in
/usr/local/lib/python3.8/dist-packages (from pandas-profiling) (6.0)
Requirement already satisfied: typeguard<2.14,>=2.13.2 in
/usr/local/lib/python3.8/dist-packages (from pandas-profiling)
(2.13.3)
Requirement already satisfied: tqdm<4.65,>=4.48.2 in
/usr/local/lib/python3.8/dist-packages (from pandas-profiling)
(4.64.1)
Requirement already satisfied: pandas!=1.4.0,<1.6,>1.1 in
/usr/local/lib/python3.8/dist-packages (from pandas-profiling) (1.3.5)
Requirement already satisfied: numpy<1.24,>=1.16.0 in
/usr/local/lib/python3.8/dist-packages (from pandas-profiling)
(1.21.6)
Requirement already satisfied: scipy<1.10,>=1.4.1 in
/usr/local/lib/python3.8/dist-packages (from pandas-profiling) (1.7.3)
Requirement already satisfied: tangled-up-in-unicode>=0.0.4 in
/usr/local/lib/python3.8/dist-packages (from
visions[type_image_path]==0.7.5->pandas-profiling) (0.2.0)
Requirement already satisfied: attrs>=19.3.0 in
```

```
/usr/local/lib/python3.8/dist-packages (from
visions[type_image_path]==0.7.5->pandas-profiling) (22.2.0)
Requirement already satisfied: networkx>=2.4 in
/usr/local/lib/python3.8/dist-packages (from
visions[type_image_path]==0.7.5->pandas-profiling) (3.0)
Requirement already satisfied: Pillow in
/usr/local/lib/python3.8/dist-packages (from
visions[type_image_path]==0.7.5->pandas-profiling) (7.1.2)
Requirement already satisfied: imagehash in
/usr/local/lib/python3.8/dist-packages (from
visions[type_image_path]==0.7.5->pandas-profiling) (4.3.1)
Requirement already satisfied: MarkupSafe>=0.23 in
/usr/local/lib/python3.8/dist-packages (from jinja2<3.2,>=2.11.1-
>pandas-profiling) (2.0.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.8/dist-packages (from matplotlib<3.7,>=3.2-
>pandas-profiling) (1.4.4)
Requirement already satisfied: python-dateutil>=2.1 in
/usr/local/lib/python3.8/dist-packages (from matplotlib<3.7,>=3.2-
>pandas-profiling) (2.8.2)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.8/dist-packages (from matplotlib<3.7,>=3.2-
>pandas-profiling) (0.11.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!
=2.1.6,>=2.0.1 in /usr/local/lib/python3.8/dist-packages (from
matplotlib<3.7,>=3.2->pandas-profiling) (3.0.9)
Requirement already satisfied: pytz>=2017.3 in
/usr/local/lib/python3.8/dist-packages (from pandas!=1.4.0,<1.6,>1.1-
>pandas-profiling) (2022.7)
Requirement already satisfied: joblib>=0.14.1 in
/usr/local/lib/python3.8/dist-packages (from phik<0.13,>=0.11.1-
>pandas-profiling) (1.2.0)
Requirement already satisfied: typing-extensions>=4.2.0 in
/usr/local/lib/python3.8/dist-packages (from pydantic<1.11,>=1.8.1-
>pandas-profiling) (4.4.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/usr/local/lib/python3.8/dist-packages (from requests<2.29,>=2.24.0-
>pandas-profiling) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in
/usr/local/lib/python3.8/dist-packages (from requests<2.29,>=2.24.0-
>pandas-profiling) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.8/dist-packages (from requests<2.29,>=2.24.0-
>pandas-profiling) (2022.12.7)
Requirement already satisfied: chardet<5,>=3.0.2 in
/usr/local/lib/python3.8/dist-packages (from requests<2.29,>=2.24.0-
>pandas-profiling) (4.0.0)
Requirement already satisfied: patsy>=0.5.2 in
/usr/local/lib/python3.8/dist-packages (from
statsmodels<0.14,>=0.13.2->pandas-profiling) (0.5.3)
```

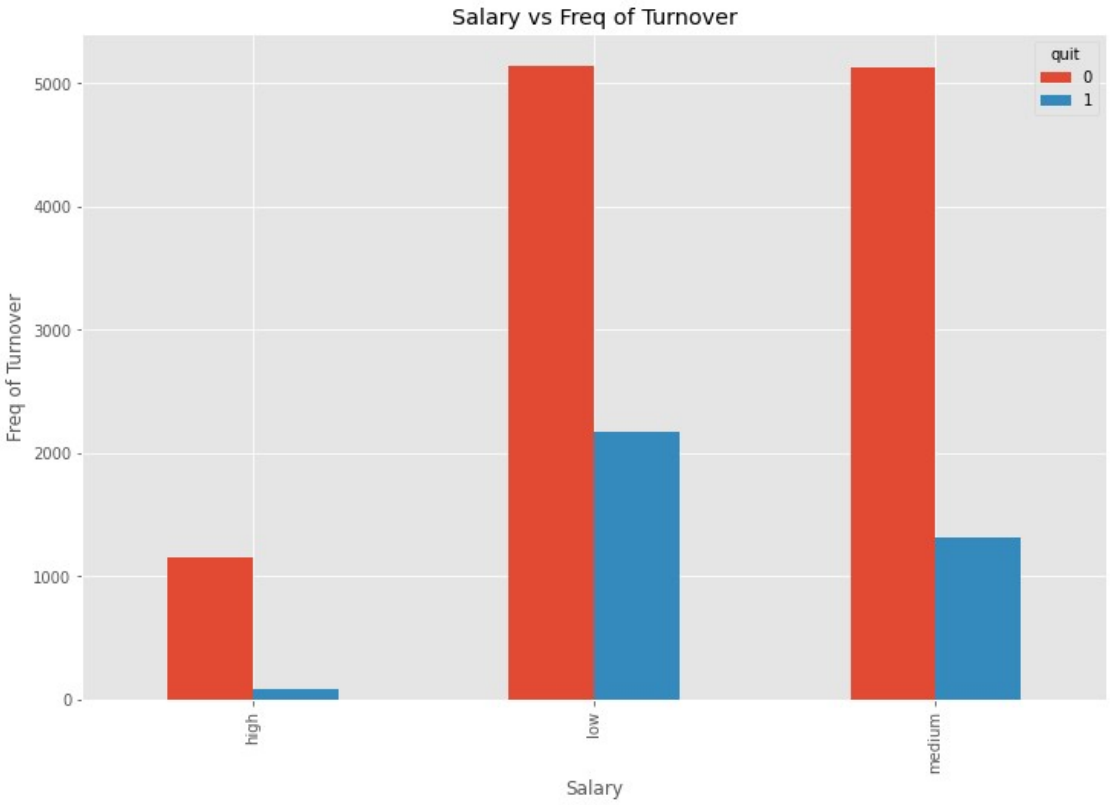
```
Requirement already satisfied: packaging>=21.3 in
/usr/local/lib/python3.8/dist-packages (from
statsmodels<0.14,>=0.13.2->pandas-profiling) (21.3)
Requirement already satisfied: six in /usr/local/lib/python3.8/dist-
packages (from patsy>=0.5.2->statsmodels<0.14,>=0.13.2->pandas-
profiling) (1.15.0)
Requirement already satisfied: PyWavelets in
/usr/local/lib/python3.8/dist-packages (from imagehash-
>visions[type_image_path]==0.7.5->pandas-profiling) (1.4.1)
```

```
import pandas_profiling

profile = pandas_profiling.ProfileReport(emp)
profile
```

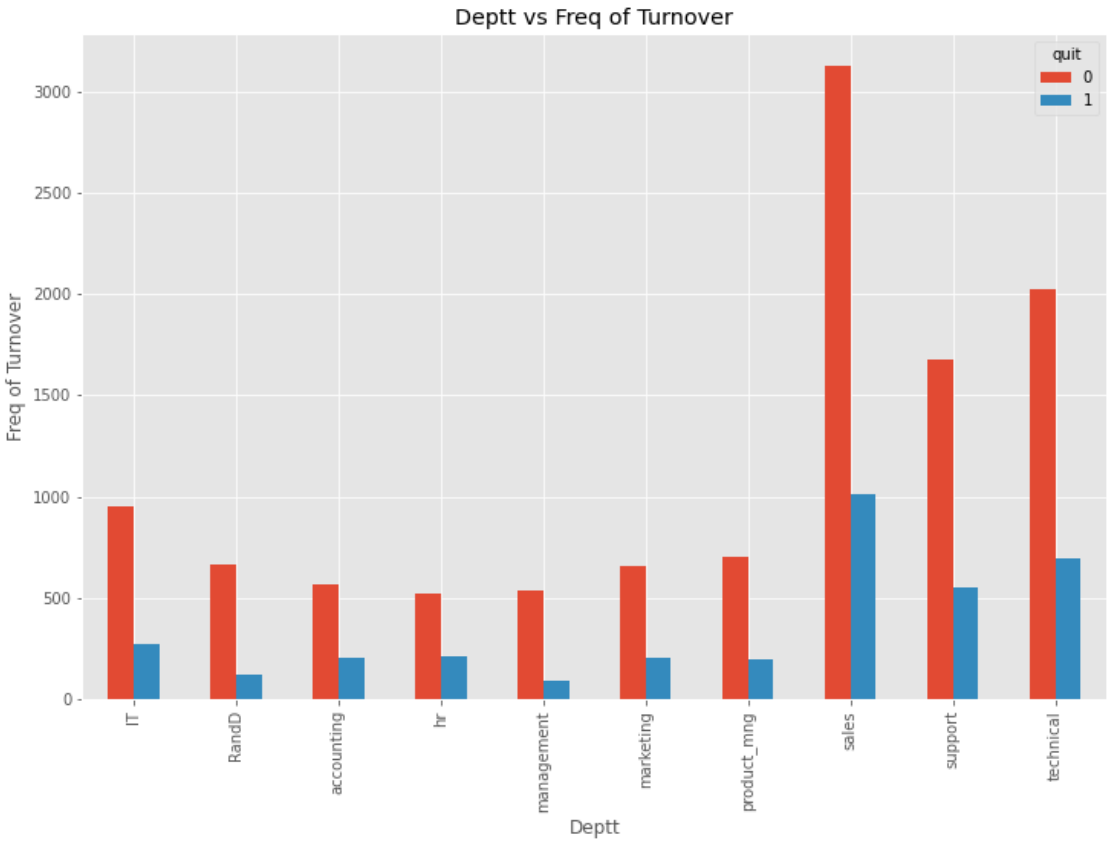

```
{"model_id":"6bcc4b938ce249b0a50c09e84bd02371","version_major":2,"vers
ion_minor":0}

{"model_id":"e839f55f4c3844aaa6c86edaaf394c38","version_major":2,"vers
ion_minor":0}

{"model_id":"b8bb46c130dd4782b99df8ef94ec1b52","version_major":2,"vers
ion_minor":0}
```


```
<IPython.core.display.HTML object>
```

```
pd.crosstab(emp.salary, emp.quit).plot(kind="bar")
plt.title("Salary vs Freq of Turnover")
plt.xlabel("Salary")
plt.ylabel("Freq of Turnover")
plt.show()
```




```
pd.crosstab(emp.department, emp.quit).plot(kind="bar")
plt.title("Deptt vs Freq of Turnover")
plt.xlabel("Deptt")
plt.ylabel("Freq of Turnover")
plt.show()
```




```
# to convert categorical variable into dummy/indicator variables
vars = ['salary', 'department']
for var in vars:
  # use prefix so that IT is names as department_IT
  ls = pd.get_dummies(emp[var],prefix=var)
  emp = emp.join(ls)

emp.head()
```

```
   satisfaction_level  last_evaluation  number_project
average_montly_hours  \
0                0.38             0.53               2
157
1                0.80             0.86               5
262
2                0.11             0.88               7
272
3                0.72             0.87               5
223
4                0.37             0.52               2
159

   time_spend_company  Work_accident  quit  promotion_last_5years
department  \
0                   3              0     1                      0
```

```
sales
1                    6              0     1                         0
sales
2                    4              0     1                         0
sales
3                    5              0     1                         0
sales
4                    3              0     1                         0
sales

   salary  ...  department_IT  department_RandD  department_accounting
\
0     low  ...              0                 0                      0

1  medium  ...              0                 0                      0

2  medium  ...              0                 0                      0

3     low  ...              0                 0                      0

4     low  ...              0                 0                      0


   department_hr  department_management  department_marketing  \
0              0                      0                      0
1              0                      0                      0
2              0                      0                      0
3              0                      0                      0
4              0                      0                      0

   department_product_mng  department_sales  department_support  \
0                       0                 1                   0
1                       0                 1                   0
2                       0                 1                   0
3                       0                 1                   0
4                       0                 1                   0

   department_technical
0                     0
1                     0
2                     0
3                     0
4                     0

[5 rows x 23 columns]
```

```
# axis=1 is for vertical
emp.drop(vars,axis=1,inplace=True)
```

```
from yellowbrick.target import ClassBalance
```

Class Imbalance

Even if all parameters fine, accuracy may be compromised if the train and test data don't come from same distribution. E.g. If in training 90% employees quit the job and in test data only 40% did, then in prediction on test data, our model is surely gonna tell >40%. Hence less accuracy.

```
# 0=stayes, 1=quit
visualizer = ClassBalance(labels=['stayed','quit']).fit(emp.quit)
visualizer.poof()
```


```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6e25a4b730>
```

```
X,y = emp.loc[:,emp.columns!='quit'], emp.quit
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=101,train_size=0.8,stratify=y)
```

Decision Trees

```
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.tree import export_graphviz # display the tree within a
Jupyter notebook
from IPython.display import SVG
```

```
from graphviz import Source
from IPython.display import display
from ipywidgets import interactive, IntSlider, FloatSlider, interact
import ipywidgets
from IPython.display import Image
from subprocess import call
import matplotlib.image as mpimg

@interact
def plot_tree(criteria=['gini','entropy'],
              split=['best','random'],
              depth=IntSlider(min=1,max=25,value=2,
continuous_update=True),
              min_split=IntSlider(min=2,max=5,value=2,
continuous_update=True),
              min_leaf=IntSlider(min=1,max=5,value=1,
continuous_update=True)):
  model = DecisionTreeClassifier(random_state=101,
criterion=criteria,splitter=split,max_depth=depth,min_samples_leaf=min
_leaf,min_samples_split=min_split)
  model.fit(X_train,y_train)
  print("Decision tree accuracy on X_test
{:.3f}".format(accuracy_score(y_test,model.predict(X_test))))
  # visualizer
  graph =
Source(tree.export_graphviz(model,filled=True,feature_names=X_train.co
lumns,class_names=['Stayed','quit']))
  display(Image(data=graph.pipe(format='png')))
  return model


{"model_id":"12791958ecd74771aad1da6ac8ef7c08","version_major":2,"vers
ion_minor":0}
```

Random Forests Decision trees tend to overfit(high variance problem). So we use forests so as to trade-off bias for variance.

- Randomization and averaging (multiple trees) leads to less variance.
- Bootstrapping is used to reduce training time & memory consumption.
- Implementation are parallelizable. So we can use multiple CPU cores simultaneously.

```
@interact
def plot_tree_rf(criteria=['gini','entropy'],
                 bootstrap=['True','False'],
                 depth=IntSlider(min=1,max=25,value=2,
continuous_update=True),

forests=IntSlider(min=1,max=150,value=75,continuous_update=True),
                 min_split=IntSlider(min=2,max=5,value=2,
continuous_update=True),
```

```
                min_leaf=IntSlider(min=1,max=5,value=1,
continuous_update=True)):
  model = RandomForestClassifier(random_state=101,
criterion=criteria,bootstrap=bootstrap,n_estimators=forests,max_depth=
depth,min_samples_split=min_split,min_samples_leaf=min_leaf,n_jobs=1,v
erbose=False)
  model.fit(X_train,y_train)
  print("Random forest accuracy on X_test
{:.3f}".format(accuracy_score(y_test,model.predict(X_test))))
  # visualizer for one of the trees
  specific_tree = model.estimators_[0]
  graph =
Source(tree.export_graphviz(specific_tree,filled=True,feature_names=X_
train.columns,class_names=['Stayed','quit']))
  display(Image(data=graph.pipe(format='png')))
  return model
```


```
{"model_id":"fdc1d1bd3ee8415180d1bbc3bcc75e83","version_major":2,"vers
ion_minor":0}
```

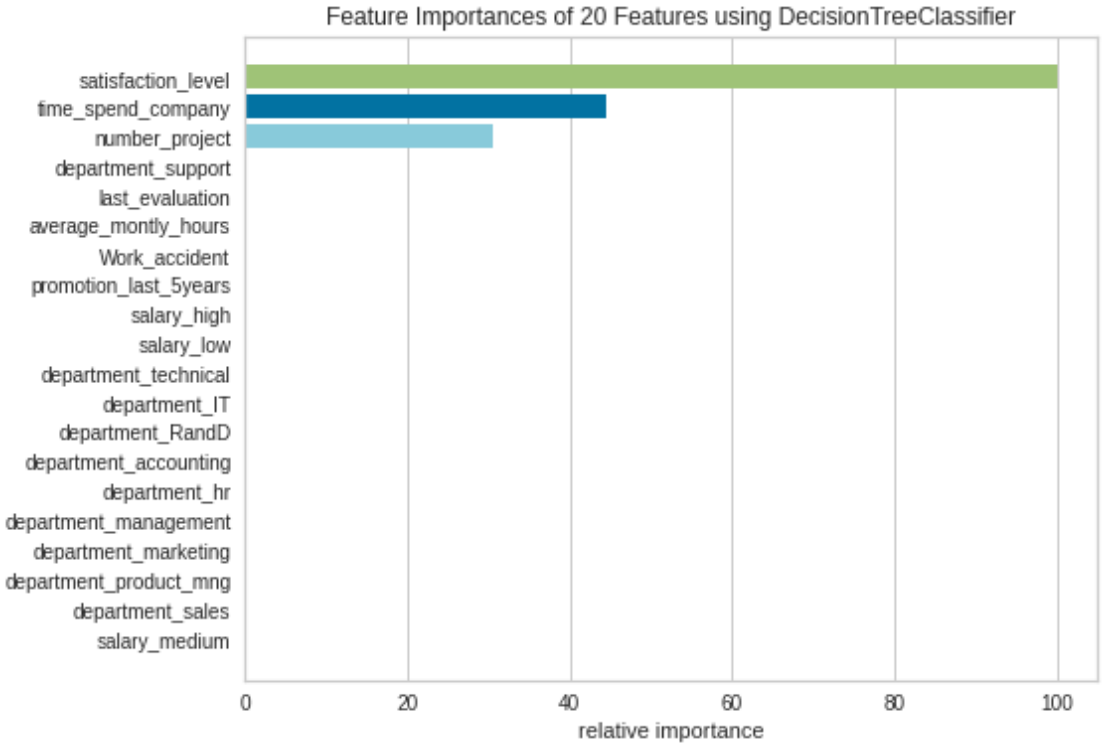

Important Features

```
!pip install -U yellowbrick
```

```
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: yellowbrick in
/usr/local/lib/python3.8/dist-packages (1.5)
Requirement already satisfied: scikit-learn>=1.0.0 in
/usr/local/lib/python3.8/dist-packages (from yellowbrick) (1.0.2)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.2 in
/usr/local/lib/python3.8/dist-packages (from yellowbrick) (3.2.2)
Requirement already satisfied: scipy>=1.0.0 in
/usr/local/lib/python3.8/dist-packages (from yellowbrick) (1.7.3)
Requirement already satisfied: numpy>=1.16.0 in
/usr/local/lib/python3.8/dist-packages (from yellowbrick) (1.21.6)
Requirement already satisfied: cycler>=0.10.0 in
/usr/local/lib/python3.8/dist-packages (from yellowbrick) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.8/dist-packages (from matplotlib!
=3.0.0,>=2.0.2->yellowbrick) (1.4.4)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!
=2.1.6,>=2.0.1 in /usr/local/lib/python3.8/dist-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (3.0.9)
Requirement already satisfied: python-dateutil>=2.1 in
/usr/local/lib/python3.8/dist-packages (from matplotlib!
=3.0.0,>=2.0.2->yellowbrick) (2.8.2)
Requirement already satisfied: joblib>=0.11 in
/usr/local/lib/python3.8/dist-packages (from scikit-learn>=1.0.0-
>yellowbrick) (1.2.0)
```

```
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.8/dist-packages (from scikit-learn>=1.0.0-
>yellowbrick) (3.1.0)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.8/dist-packages (from python-dateutil>=2.1-
>matplotlib!=3.0.0,>=2.0.2->yellowbrick) (1.15.0)
```

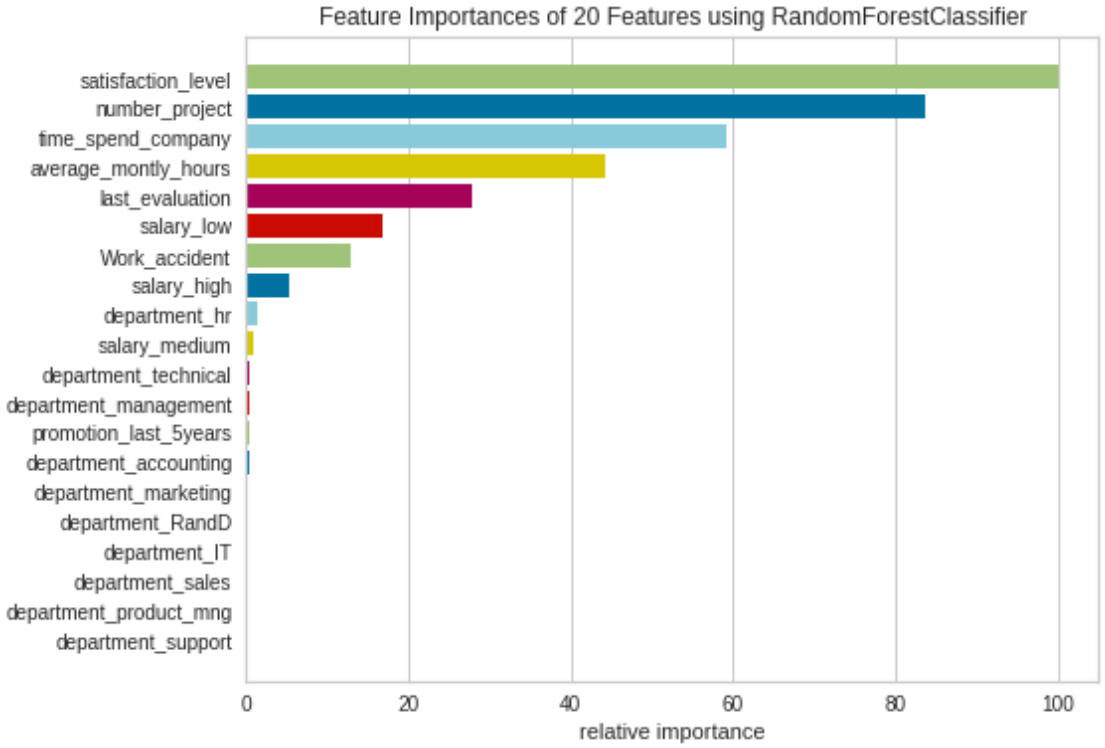
```
from yellowbrick.model_selection import FeatureImportances

dtc = DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None,
criterion='gini',
                       max_depth=2, max_features=None,
max_leaf_nodes=None,
                       min_impurity_decrease=0.0,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0,
                       random_state=101, splitter='best')
fi = FeatureImportances(dtc)
fi.fit(X_train,y_train)
fi.show()
```




```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6e284c9460>
```

```
rfc = RandomForestClassifier(bootstrap='True', ccp_alpha=0.0,
class_weight=None,
                       criterion='gini', max_depth=2,
max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
```

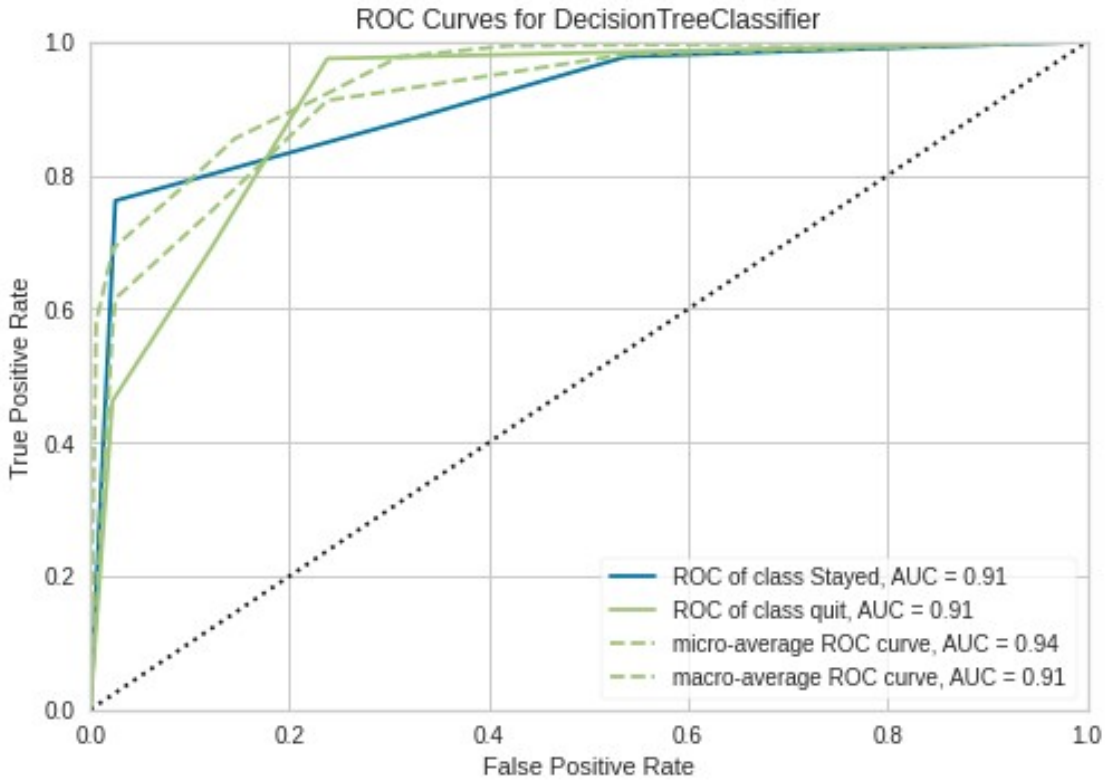
```
                       min_impurity_decrease=0.0,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=75,
n_jobs=1,
                       oob_score=False, random_state=101,
verbose=False,
                       warm_start=False)
fi = FeatureImportances(rfc)
fi.fit(X_train,y_train)
fi.show()
```




```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6e2b93b490>
```

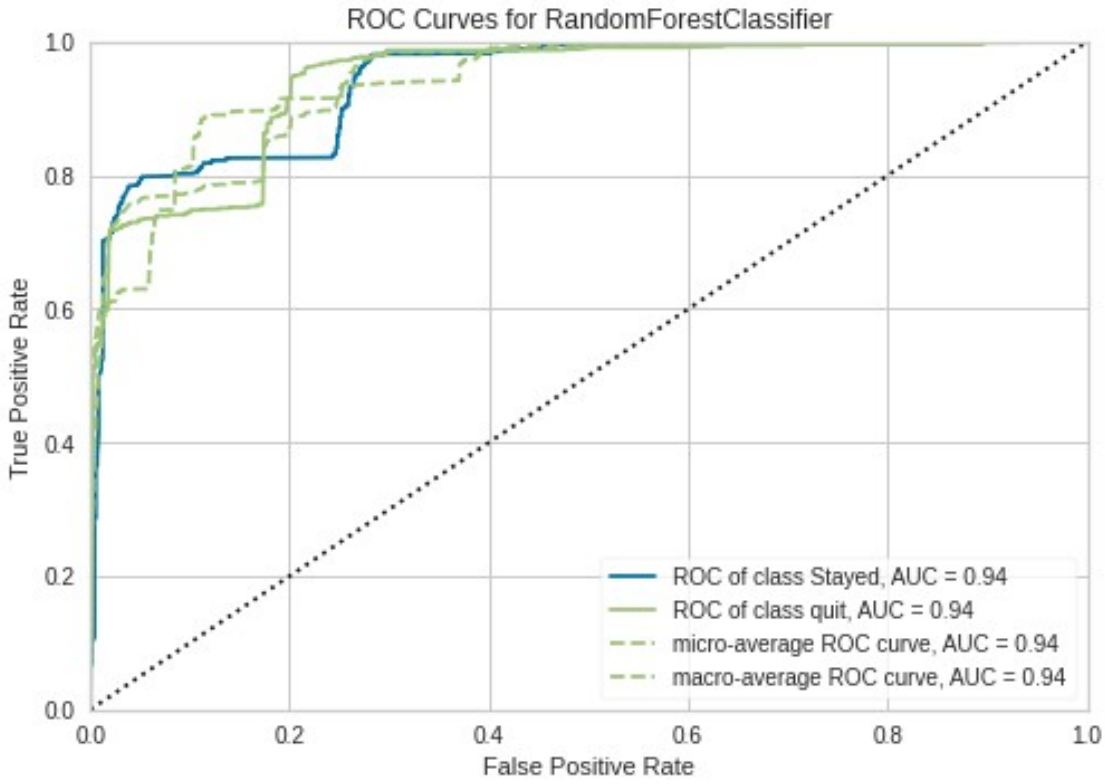
ROC AUC Curve

```
from yellowbrick.classifier import ROCAUC

model = ROCAUC(dtc, classes=['Stayed','quit'])
model.fit(X_train,y_train)
model.score(X_test,y_test)
model.poof()
```




```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6e2c463a30>
```

```
from yellowbrick.classifier import ROCAUC

model = ROCAUC(rfc, classes=['Stayed','quit'])
model.fit(X_train,y_train)
model.score(X_test,y_test)
model.poof()
```




```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6e2853f5b0>
```

Project Completed By : Santhosh TN.