```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,classification_report

import matplotlib.pyplot as plt


hrattr_data = pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition.csv")

print (hrattr_data.head())

hrattr_data['Attrition_ind'] = 0
hrattr_data.loc[hrattr_data['Attrition']=='Yes','Attrition_ind'] = 1

dummy_busnstrvl = pd.get_dummies(hrattr_data['BusinessTravel'], prefix='busns_trvl')
dummy_dept = pd.get_dummies(hrattr_data['Department'], prefix='dept')
dummy_edufield = pd.get_dummies(hrattr_data['EducationField'], prefix='edufield')
dummy_gender = pd.get_dummies(hrattr_data['Gender'], prefix='gend')
dummy_jobrole = pd.get_dummies(hrattr_data['JobRole'], prefix='jobrole')
dummy_maritstat = pd.get_dummies(hrattr_data['MaritalStatus'], prefix='maritalstat')
dummy_overtime = pd.get_dummies(hrattr_data['OverTime'], prefix='overtime')

continuous_columns = ['Age','DailyRate','DistanceFromHome','Education','EnvironmentSatisfaction',
'HourlyRate', 'JobInvolvement', 'JobLevel','JobSatisfaction','MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
'PercentSalaryHike', 'PerformanceRating', 'RelationshipSatisfaction','StockOptionLevel', 'TotalWorkingYears',
'TrainingTimesLastYear','WorkLifeBalance', 'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
'YearsWithCurrManager']

hrattr_continuous = hrattr_data[continuous_columns]

hrattr_continuous['Age'].describe()
hrattr_data['BusinessTravel'].value_counts()

hrattr_data_new = pd.concat([dummy_busnstrvl,dummy_dept,dummy_edufield,dummy_gender,dummy_jobrole,
  dummy_maritstat,dummy_overtime,hrattr_continuous,hrattr_data['Attrition_ind']],axis=1)

# Train & Test split
x_train,x_test,y_train,y_test = train_test_split(hrattr_data_new.drop(['Attrition_ind'],axis=1),
                                                 hrattr_data_new['Attrition_ind'],train_size = 0.7,random_state=42)

# Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
dt_fit = DecisionTreeClassifier(criterion="gini",max_depth=5,min_samples_split=2,min_samples_leaf=1,random_state=42)
dt_fit.fit(x_train,y_train)

print ("\nDecision Tree - Train Confusion Matrix\n\n",pd.crosstab(y_train,dt_fit.predict(x_train),rownames = ["Actuall"],colnames = ["Predict
print ("\nDecision Tree - Train accuracy:",round(accuracy_score(y_train,dt_fit.predict(x_train)),3))
print ("\nDecision Tree - Train Classification Report\n",classification_report(y_train,dt_fit.predict(x_train)))

print ("\n\nDecision Tree - Test Confusion Matrix\n\n",pd.crosstab(y_test,dt_fit.predict(x_test),rownames = ["Actuall"],colnames = ["Predicte
print ("\nDecision Tree - Test accuracy:",round(accuracy_score(y_test,dt_fit.predict(x_test)),3))
print ("\nDecision Tree - Test Classification Report\n",classification_report(y_test,dt_fit.predict(x_test)))


# Tuning class weights to analyze accuracy, precision & recall
dummyarray = np.empty((6,10))
dt_wttune = pd.DataFrame(dummyarray)

dt_wttune.columns = ["zero_wght","one_wght","tr_accuracy","tst_accuracy","prec_zero","prec_one",
                     "prec_ovll","recl_zero","recl_one","recl_ovll"]

zero_clwghts = [0.01,0.1,0.2,0.3,0.4,0.5]

for i in range(len(zero_clwghts)):
    clwght = {0:zero_clwghts[i],1:1.0-zero_clwghts[i]}
    dt_fit = DecisionTreeClassifier(criterion="gini",max_depth=5,min_samples_split=2,
                                    min_samples_leaf=1,random_state=42,class_weight = clwght)
    dt_fit.fit(x_train,y_train)
    dt_wttune.loc[i, 'zero_wght'] = clwght[0]
    dt_wttune.loc[i, 'one_wght'] = clwght[1]
    dt_wttune.loc[i, 'tr_accuracy'] = round(accuracy_score(y_train,dt_fit.predict(x_train)),3)
    dt_wttune.loc[i, 'tst_accuracy'] = round(accuracy_score(y_test,dt_fit.predict(x_test)),3)

    clf_sp = classification_report(y_test,dt_fit.predict(x_test)).split()
    dt_wttune.loc[i, 'prec_zero'] = float(clf_sp[5])
```

```
        dt_wttune.loc[i, 'prec_one'] = float(clf_sp[10])
        # Assuming clf_sp[17] might contain non-numeric values
    try:
        dt_wttune.loc[i, 'prec_ovll'] = float(clf_sp[17])
    except ValueError:
        # Handle the case where the value cannot be converted to a float
        print(f"Error: Could not convert '{clf_sp[17]}' to float for index {i}")
        dt_wttune.loc[i, 'recl_zero'] = float(clf_sp[6])
        dt_wttune.loc[i, 'recl_one'] = float(clf_sp[11])
        # Assuming clf_sp[18] might contain non-numeric values
    try:
        dt_wttune.loc[i, 'recl_ovll'] = float(clf_sp[18])
    except ValueError:
        # Handle the case where the value cannot be converted to a float
        print(f"Error: Could not convert '{clf_sp[18]}' to float for index {i}")
        print ("\nClass Weights",clwght,"Train accuracy:",round(accuracy_score(y_train,dt_fit.predict(x_train)),3),"Test accuracy:",round(accurac
        print ("Test Confusion Matrix\n\n",pd.crosstab(y_test,dt_fit.predict(x_test),rownames = ["Actuall"],colnames = ["Predicted"]))
```

```
0                  4                       0                   5
1                  7                       1                   7
2                  0                       0                   0
3                  7                       3                   0
4                  2                       2                   2

[5 rows x 35 columns]

Decision Tree - Train Confusion Matrix

 Predicted    0   1
Actuall
0          844   9
1           98  78

Decision Tree - Train accuracy: 0.896

Decision Tree - Train Classification Report
               precision    recall  f1-score   support

           0       0.90      0.99      0.94       853
           1       0.90      0.44      0.59       176

    accuracy                           0.90      1029
   macro avg       0.90      0.72      0.77      1029
weighted avg       0.90      0.90      0.88      1029



Decision Tree - Test Confusion Matrix

 Predicted    0   1
Actuall
0          361  19
1           49  12

Decision Tree - Test accuracy: 0.846

Decision Tree - Test Classification Report
               precision    recall  f1-score   support

           0       0.88      0.95      0.91       380
           1       0.39      0.20      0.26        61

    accuracy                           0.85       441
   macro avg       0.63      0.57      0.59       441
weighted avg       0.81      0.85      0.82       441

Error: Could not convert 'macro' to float for index 5
Error: Could not convert 'avg' to float for index 5

Class Weights {0: 0.5, 1: 0.5} Train accuracy: 0.896 Test accuracy: 0.846
Test Confusion Matrix

 Predicted    0   1
Actuall
0          361  19
1           49  12
```