

```
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")
```

```
hrattr_data
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educati
0	41	Yes	Travel_Rarely	1102	Sales		1
1	49	No	Travel_Frequently	279	Research & Development		8
2	37	Yes	Travel_Rarely	1373	Research & Development		2
3	33	No	Travel_Frequently	1392	Research & Development		3
4	27	No	Travel_Rarely	591	Research & Development		2
...
1465	36	No	Travel_Frequently	884	Research & Development		23
1466	39	No	Travel_Rarely	613	Research & Development		6

```
print(hrattr_data)
```

1466	6	1	Medical	1
1467	4	3	Life Sciences	1
1468	2	3	Medical	1
1469	8	3	Medical	1

	EmployeeNumber	...	RelationshipSatisfaction	StandardHours	\
0	1	...	1	80	
1	2	...	4	80	
2	4	...	2	80	
3	5	...	3	80	
4	7	...	4	80	
...	
1465	2061	...	3	80	
1466	2062	...	1	80	
1467	2064	...	2	80	
1468	2065	...	4	80	
1469	2068	...	1	80	

	StockOptionLevel	TotalWorkingYears	TrainingTimesLastYear	\
--	------------------	-------------------	-----------------------	---

146/	3	6	2
1468	2	9	6
1469	4	4	3

	YearsSinceLastPromotion	YearsWithCurrManager
0	0	5
1	1	7
2	0	0
3	3	0
4	2	2
...
1465	0	3
1466	1	7
1467	0	3
1468	0	8
1469	1	2

[1470 rows x 35 columns]

```
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()
```

```
Travel_Rarely      1043
Travel_Frequently   277
Non-Travel         150
Name: BusinessTravel, dtype: int64
```

```
hrattr_data_new = pd.concat([dummy_busnstrvl,dummy_dept,dummy_edufield,dummy_gender,dummy_jobrole,dummy_maritstat,dummy_overtime,hrattr_conti
```

```
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)
```

```
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 = ["Actual"],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 = ["Actual"],colnames = ["Predict
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)))
```

Decision Tree - Train Confusion Matrix

	Predicted 0	1
Actual		
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	Predicted 1
Actual 0	361	19
Actual 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

```
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),output_dict=True)
```

```
    dt_wttune.loc[i, 'prec_zero'] = clf_sp['0']['precision']
```

```
    dt_wttune.loc[i, 'prec_one'] = clf_sp['1']['precision']
```

```
    dt_wttune.loc[i, 'prec_ovll'] = clf_sp['macro avg']['precision']
```

```
    dt_wttune.loc[i, 'recl_zero'] =clf_sp['0']['recall']
```

```
    dt_wttune.loc[i, 'recl_one'] = clf_sp['1']['recall']
```

```
    dt_wttune.loc[i, 'recl_ovll'] = clf_sp['macro avg']['recall']
```

```
    print ("\nClass Weights",clwght,"Train accuracy:",round(accuracy_score(y_train,dt_fit.predict(x_train)),3),"Test accuracy:",round(accuracy_score(y_test,dt_fit.predict(x_test)),3))
```

```
    print ("Test Confusion Matrix\n\n",pd.crosstab(y_test,dt_fit.predict(x_test),rownames = ["Actual"],colnames = ["Predicted"]))
```

Class Weights {0: 0.01, 1: 0.99} Train accuracy: 0.342 Test accuracy: 0.272

Test Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	65	315
Actual 1	6	55

Class Weights {0: 0.1, 1: 0.9} Train accuracy: 0.806 Test accuracy: 0.732

Test Confusion Matrix

	Predicted 0	Predicted 1
Actual 0	282	98
Actual 1	20	41

```
Class Weights {0: 0.2, 1: 0.8} Train accuracy: 0.871 Test accuracy: 0.83  
Test Confusion Matrix
```

```
   Predicted    0    1  
Actual1  
0           341   39  
1           36   25
```

```
Class Weights {0: 0.3, 1: 0.7} Train accuracy: 0.881 Test accuracy: 0.837  
Test Confusion Matrix
```

```
   Predicted    0    1  
Actual1  
0           345   35  
1           37   24
```

```
Class Weights {0: 0.4, 1: 0.6} Train accuracy: 0.894 Test accuracy: 0.832  
Test Confusion Matrix
```

```
   Predicted    0    1  
Actual1  
0           346   34  
1           40   21
```

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

```
   Predicted    0    1  
Actual1  
0           361   19  
1           49   12
```