

HR Analytics:

Problem solving with SVM - Logistic Regression

About HR Analytics:

HR analytics is revolutionising the way human resources departments operate, leading to higher efficiency and better results overall. Human resources has been using analytics for years. However, the collection, processing and analysis of data has been largely manual, and given the nature of human resources dynamics and HR KPIs, the approach has been constraining HR. Therefore, it is surprising that HR departments woke up to the utility of machine learning so late in the game. Here is an opportunity to try predictive analytics in identifying the employees most likely to get promoted.

Client:

Can you predict which employee has a higher chance of getting promoted and help the help the organization to expedite the appraisal and promotion process?

```
In [1]: import pandas as pd
import numpy as np
```

Import Files:

Train Data & Test Data

```
In [2]: # Importing training data
train_data=pd.read_csv(r'F:\Prthon Programming\HR Analytics\HR_Analytics_train.csv',index_col=0,
                        header=0)

# Importing training data
test_data=pd.read_csv(r'F:\Prthon Programming\HR Analytics\HR_Analytics_test.csv',
                      index_col=0,header=0)
```

Head:

Train Data & Test Data

In [3]: `train_data.head()`

Out[3]:

	department	region	education	gender	recruitment_channel	no_of_trainings	age
employee_id							
65438	Sales & Marketing	region_7	Master's & above	f	sourcing	1	3
65141	Operations	region_22	Bachelor's	m	other	1	3
7513	Sales & Marketing	region_19	Bachelor's	m	sourcing	1	3
2542	Sales & Marketing	region_23	Bachelor's	m	other	2	3
48945	Technology	region_26	Bachelor's	m	other	1	4

In [4]: `test_data.head()`

Out[4]:

	department	region	education	gender	recruitment_channel	no_of_trainings	age
employee_id							
8724	Technology	region_26	Bachelor's	m	sourcing	1	2
74430	HR	region_4	Bachelor's	f	other	1	3
72255	Sales & Marketing	region_13	Bachelor's	m	other	1	3
38562	Procurement	region_2	Bachelor's	f	other	3	3
64486	Finance	region_29	Bachelor's	m	sourcing	1	3

Shape:

Train Data & Test Data

In [5]: `print(train_data.shape)`

(54808, 13)

In [6]: `print(test_data.shape)`

(23490, 12)

Finding Missing Value:

Train Data & Test Data

```
In [7]: train_data.isnull().sum()
```

```
Out[7]: department          0
region                    0
education                2409
gender                   0
recruitment_channel      0
no_of_trainings          0
age                      0
previous_year_rating    4124
length_of_service        0
KPIs_met >80%            0
awards_won?              0
avg_training_score        0
is_promoted              0
dtype: int64
```

```
In [8]: test_data.isnull().sum()
```

```
Out[8]: department          0
region                    0
education                1034
gender                   0
recruitment_channel      0
no_of_trainings          0
age                      0
previous_year_rating    1812
length_of_service        0
KPIs_met >80%            0
awards_won?              0
avg_training_score        0
dtype: int64
```

Handling The Missing Value:

Train Data & Test Data

```
In [9]: for value in ['education']:
        train_data[value].fillna(train_data[value].mode()[0],inplace=True)
```

```
In [10]: train_data['previous_year_rating'].fillna(round(train_data['previous_year_rating'].mean(),0),inplace=True)
```

```
In [11]: train_data.isnull().sum()
```

```
Out[11]: department      0
region                  0
education               0
gender                 0
recruitment_channel    0
no_of_trainings        0
age                   0
previous_year_rating    0
length_of_service      0
KPIs_met >80%          0
awards_won?            0
avg_training_score      0
is_promoted            0
dtype: int64
```

```
In [12]: for value in ['education']:
         test_data[value].fillna(test_data[value].mode()[0],inplace=True)
```

```
In [13]: test_data['previous_year_rating'].fillna(round(test_data['previous_year_rating'].mean(),0),inplace=True)
```

```
In [14]: test_data.isnull().sum()
```

```
Out[14]: department      0
region                  0
education               0
gender                 0
recruitment_channel    0
no_of_trainings        0
age                   0
previous_year_rating    0
length_of_service      0
KPIs_met >80%          0
awards_won?            0
avg_training_score      0
dtype: int64
```

Converting From Factors to Numerical:

Train Data & Test Data

```
In [15]: colname=[]
         for x in train_data.columns[:]:
             if train_data[x].dtype=='object':
                 colname.append(x)
         colname
```

```
Out[15]: ['department', 'region', 'education', 'gender', 'recruitment_channel']
```

```
In [16]: from sklearn import preprocessing

le = preprocessing.LabelEncoder()

for x in colname:
    train_data[x]=le.fit_transform(train_data[x])

train_data.head()
```

Out[16]:

	department	region	education	gender	recruitment_channel	no_of_trainings	age
employee_id							
65438	7	31	2	0	2	1	35
65141	4	14	0	1	0	1	30
7513	7	10	0	1	2	1	34
2542	7	15	0	1	0	2	39
48945	8	18	0	1	0	1	45

```
In [17]: colname=[]
for x in test_data.columns[:]:
    if test_data[x].dtype=='object':
        colname.append(x)
colname
```

Out[17]: ['department', 'region', 'education', 'gender', 'recruitment_channel']

```
In [18]: from sklearn import preprocessing

le = preprocessing.LabelEncoder()

for x in colname:
    test_data[x]=le.fit_transform(test_data[x])

test_data.head()
```

Out[18]:

	department	region	education	gender	recruitment_channel	no_of_trainings	age
employee_id							
8724	8	18	0	1	2	1	24
74430	2	28	0	0	0	1	31
72255	7	4	0	1	0	1	31
38562	5	11	0	0	0	3	31
64486	1	21	0	1	2	1	30

Defining Variable:

Train Data & Test Data

X is Independent Variable Y is Dependent Variable

```
In [19]: X_train = train_data.values[:,0:-1]
Y_train = train_data.values[:, -1]
Y_train = Y_train.astype(int)
```

```
In [20]: X_test = test_data.values[:,:]
```

```
In [21]: Y_train.shape
```

```
Out[21]: (54808,)
```

Scaler:

```
In [22]: from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
```

```
scaler.fit(X_train)
```

```
X_train = scaler.transform(X_train)
```

```
X_test = scaler.transform(X_test)
```

```
print(X_train)
```

```
[[ 0.80893285  1.77652645  1.61975831 ...  1.35687789 -0.15401776
 -1.07593145]
 [-0.38818322 -0.16303375 -0.62883817 ... -0.73698599 -0.15401776
 -0.25328242]
 [ 0.80893285 -0.61940086 -0.62883817 ... -0.73698599 -0.15401776
 -1.00114517]
 ...
 [-1.98433798 -1.76031862 -0.62883817 ...  1.35687789 -0.15401776
  1.1676568 ]
 [ 0.80893285  2.00471    -0.62883817 ... -0.73698599 -0.15401776
 -1.37507655]
 [-1.1862606  -0.16303375 -0.62883817 ... -0.73698599 -0.15401776
 -1.07593145]]
```

SVM Modeling:

Predicting Y (Dependent Variable) for Test Data as Y_pred

```
In [23]: from sklearn import svm

svc_model = svm.SVC(kernel='rbf',C=1.0,gamma=0.1)

svc_model.fit(X_train,Y_train)

Y_pred = svc_model.predict(X_test)

print(list(Y_pred))
```

8/26

9/26

10/26

11/26

12/26

13/26

14/26

15/26

16/26

17/26

18/26

19/26

20/26

21/26

22/26

23/26

```
In [24]: new_test_data=pd.read_csv(r'F:\Prthon Programming\HR Analytics\HR_Analytics_test.csv',index_col=0,
                                     header=0)

new_test_data["Y_predictions"]=Y_pred

new_test_data.head()
```

Out[24]:

	department	region	education	gender	recruitment_channel	no_of_trainings	age
employee_id							
8724	Technology	region_26	Bachelor's	m	sourcing	1	2
74430	HR	region_4	Bachelor's	f	other	1	3
72255	Sales & Marketing	region_13	Bachelor's	m	other	1	3
38562	Procurement	region_2	Bachelor's	f	other	3	3
64486	Finance	region_29	Bachelor's	m	sourcing	1	3

```
In [25]: new_test_data.to_csv(r'F:\Prthon Programming\HR Analytics\Pred_HR_Analytics_test.csv')
```

Running Model for Accuracy:

SVM (Support Vector Machine) - Cross Validation & Logistic Regression

```
In [26]: classifier=svm.SVC(kernel='rbf',C=1.0,gamma=0.1)

#performing kfold_cross_validation
from sklearn.model_selection import KFold
kfold_cv=KFold(n_splits=10)
print(kfold_cv)

from sklearn.model_selection import cross_val_score

#running the model using scoring metric as accuracy
kfold_cv_result=cross_val_score(estimator=classifier,X=X_train,
y=Y_train, cv=kfold_cv)
print(kfold_cv_result)

#finding the mean
print(kfold_cv_result.mean())

KFold(n_splits=10, random_state=None, shuffle=False)
[0.91862799 0.92081737 0.92811531 0.92921    0.92756796 0.92154716
 0.9231892  0.92081737 0.92664234 0.91879562]
0.9235330311613976
```



```
In [27]: classifier=svm.SVC(kernel='rbf',C=10.0,gamma=0.001)

#performing kfold_cross_validation
from sklearn.model_selection import KFold
kfold_cv=KFold(n_splits=10)

print(kfold_cv)

from sklearn.model_selection import cross_val_score

#running the model using scoring metric as accuracy
kfold_cv_result=cross_val_score(estimator=classifier,X=X_train,
y=Y_train, cv=kfold_cv)

print(kfold_cv_result)

#finding the mean
print(kfold_cv_result.mean())

KFold(n_splits=10, random_state=None, shuffle=False)
[0.91552636 0.91552636 0.92483124 0.9235541  0.92410144 0.91625616
 0.91954023 0.91534392 0.92153285 0.91587591]
0.9192088562079753
```

```
In [28]: from sklearn.linear_model import LogisticRegression

import warnings
warnings.filterwarnings("ignore")

classifier=(LogisticRegression())

from sklearn.model_selection import KFold
kfold_cv=KFold(n_splits=10)
print(kfold_cv)

from sklearn.model_selection import cross_val_score
#running the model using scoring metric as accuracy
kfold_cv_result=cross_val_score(estimator=classifier,X=X_train,
y=Y_train, cv=kfold_cv)
print(kfold_cv_result)
#finding the mean
print(kfold_cv_result.mean())

KFold(n_splits=10, random_state=None, shuffle=False)
[0.91351943 0.91406678 0.92373654 0.92099982 0.92154716 0.91388433
 0.91771575 0.91315453 0.9189781  0.91386861]
0.9171471053952805
```

ACCURACY RESULTS FOR H R ANALYTICS:

SVM (Support Vector Machine) - Cross Validation [92%]

Accuracy: 92%