Employee Promotion Prediction

- Using HR dataset for binary classification

In [1]:
```
#importing libraries

#for loading,eda and preprocessing of data
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas_profiling as pp
import dtale
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler

#for model implementation
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import AdaBoostClassifier

#for evaluation metrics and cross validation
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
```

In [2]:
```
#to ignore warnings

np.seterr(divide='ignore', invalid='ignore')
```

Out[2]:
```
{'divide': 'warn', 'over': 'warn', 'under': 'ignore', 'invalid': 'warn'}
```

In [3]:
```
#importing dataset - IBM HR dataset

df_train = pd.read_excel("IBM HR_train dataset.xlsx")
df_test = pd.read_excel("IBM HR_test dataset.xlsx")
```

In [4]:
```
#storing count of train dataset rows - to be used for concat purpose in later logic

train_rows=df_train.shape[0]
```

In [5]:
```
#verifying df_train

df_train
```

Out[5]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educa
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Lif

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educa
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	
...	...	...	...	...	...	...	...	
796	25	Yes	Travel_Rarely	1219	Research & Development	4	1	
797	26	Yes	Travel_Rarely	1330	Research & Development	21	3	
798	33	Yes	Travel_Rarely	1017	Research & Development	25	3	
799	42	No	Travel_Rarely	469	Research & Development	2	2	
800	28	Yes	Travel_Frequently	1009	Research & Development	1	3	

801 rows × 36 columns

EDA of train and test dataset

In [6]:

```
#viewing imported dataset

df_train
```

Out[6]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educa
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	
...	...	...	...	...	...	...	...	
796	25	Yes	Travel_Rarely	1219	Research & Development	4	1	
797	26	Yes	Travel_Rarely	1330	Research & Development	21	3	
798	33	Yes	Travel_Rarely	1017	Research & Development	25	3	

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educa
799	42	No	Travel_Rarely	469	Research & Development	2	2	
800	28	Yes	Travel_Frequently	1009	Research & Development	1	3	

801 rows × 36 columns

In [7]:

```
#checking dataset features(columns) and datapoints(rows)

print(df_train.shape)
```

```
(801, 36)
```

In [8]:

```
#checking top 10 and bottom 10 datapoints

df_train.head(10)
```

Out[8]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educatio
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life S
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life S
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life S
4	27	No	Travel_Rarely	591	Research & Development	2	1	I
5	32	No	Travel_Frequently	1005	Research & Development	2	2	Life S
6	59	No	Travel_Rarely	1324	Research & Development	3	3	I
7	30	No	Travel_Rarely	1358	Research & Development	24	1	Life S
8	38	No	Travel_Frequently	216	Research & Development	23	3	Life S
9	36	No	Travel_Rarely	1299	Research & Development	27	3	I

10 rows × 36 columns

In [9]:

```
df_train.tail(10)
```

Out[9]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educa
791	35	Yes	Travel_Rarely	1204	Sales	4	3	

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educa
792	33	Yes	Travel_Frequently	827	Research & Development	29	4	
793	28	No	Travel_Rarely	895	Research & Development	15	2	Lif
794	34	No	Travel_Frequently	618	Research & Development	3	1	Lif
795	37	No	Travel_Rarely	309	Sales	10	4	Lif
796	25	Yes	Travel_Rarely	1219	Research & Development	4	1	
797	26	Yes	Travel_Rarely	1330	Research & Development	21	3	
798	33	Yes	Travel_Rarely	1017	Research & Development	25	3	
799	42	No	Travel_Rarely	469	Research & Development	2	2	
800	28	Yes	Travel_Frequently	1009	Research & Development	1	3	

10 rows × 36 columns

In [10]:
```
#performing eda

#checking null values

df_train.isnull().sum() #no missing/nan values - hence steps for handling missing va
```

Out[10]:
```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
```

```
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
CanBePromoted               0
dtype: int64
```

In [11]:

```
#checking datatypes and other information about dataset

#out of 36 features - 26 are int type and 10 are string/object type
#depending on requirement may need to perform encoding to prepare data for model bui

df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 801 entries, 0 to 800
Data columns (total 36 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       801 non-null    int64 
 1   Attrition                 801 non-null    object
 2   BusinessTravel            801 non-null    object
 3   DailyRate                 801 non-null    int64 
 4   Department                801 non-null    object
 5   DistanceFromHome          801 non-null    int64 
 6   Education                 801 non-null    int64 
 7   EducationField            801 non-null    object
 8   EmployeeCount             801 non-null    int64 
 9   EmployeeNumber            801 non-null    int64 
 10  EnvironmentSatisfaction   801 non-null    int64 
 11  Gender                    801 non-null    object
 12  HourlyRate                801 non-null    int64 
 13  JobInvolvement            801 non-null    int64 
 14  JobLevel                  801 non-null    int64 
 15  JobRole                   801 non-null    object
 16  JobSatisfaction           801 non-null    int64 
 17  MaritalStatus             801 non-null    object
 18  MonthlyIncome             801 non-null    int64 
 19  MonthlyRate               801 non-null    int64 
 20  NumCompaniesWorked        801 non-null    int64 
 21  Over18                    801 non-null    object
 22  OverTime                  801 non-null    object
 23  PercentSalaryHike         801 non-null    int64 
 24  PerformanceRating         801 non-null    int64 
 25  RelationshipSatisfaction  801 non-null    int64 
 26  StandardHours             801 non-null    int64 
 27  StockOptionLevel          801 non-null    int64 
 28  TotalWorkingYears         801 non-null    int64 
 29  TrainingTimesLastYear     801 non-null    int64 
 30  WorkLifeBalance           801 non-null    int64 
 31  YearsAtCompany            801 non-null    int64 
 32  YearsInCurrentRole        801 non-null    int64 
 33  YearsSinceLastPromotion   801 non-null    int64 
 34  YearsWithCurrManager      801 non-null    int64 
 35  CanBePromoted             801 non-null    object
dtypes: int64(26), object(10)
memory usage: 225.4+ KB
```

In [12]:

```
#checking object data type

categorical_dtype = df_train.select_dtypes(include=['object']).columns
```

In [13]:

```
#need to peform encoding
```

```
categorical_dtype
```

Out[13]:
```
Index(['Attrition', 'BusinessTravel', 'Department', 'EducationField', 'Gender',
       'JobRole', 'MaritalStatus', 'Over18', 'OverTime', 'CanBePromoted'],
      dtype='object')
```

In [14]:
```
#checking for duplicate records

duplicate_rec = df_train.duplicated()
```

In [15]:
```
#since false, there are no duplicate records in the dataset

print(duplicate_rec.any())
```

```
False
```

In [16]:
```
#using dtale library for detailed eda

dtale.show(df_train)
```

```
2021-11-09 22:01:22,302 - INFO     - NumExpr defaulting to 4 threads.
```

Out[16]:

In [17]:
```
#using pandas profiling for eda

#df_train.profile_report()
```

In [18]:

```
#analysis of test set

df_test
```

Out[18]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educa
0	50	Yes	Travel_Frequently	959	Sales	1	4	
1	33	No	Travel_Frequently	970	Sales	7	3	Life
2	34	No	Non-Travel	697	Research & Development	3	4	Life
3	48	No	Non-Travel	1262	Research & Development	1	4	
4	45	No	Non-Travel	1050	Sales	9	4	Life
...	...	...	...	...	...	...	...	
664	36	No	Travel_Frequently	884	Research & Development	23	2	
665	39	No	Travel_Rarely	613	Research & Development	6	1	
666	27	No	Travel_Rarely	155	Research & Development	4	3	Life
667	49	No	Travel_Frequently	1023	Sales	2	3	
668	34	No	Travel_Rarely	628	Research & Development	8	3	

669 rows × 35 columns

In [19]:

```
#viewing test set datapoints summary - no null values - 26 int and 9 object/categori

df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 669 entries, 0 to 668
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       669 non-null    int64 
 1   Attrition                 669 non-null    object
 2   BusinessTravel            669 non-null    object
 3   DailyRate                 669 non-null    int64 
 4   Department                669 non-null    object
 5   DistanceFromHome          669 non-null    int64 
 6   Education                 669 non-null    int64 
 7   EducationField            669 non-null    object
 8   EmployeeCount             669 non-null    int64 
 9   EmployeeNumber            669 non-null    int64 
 10  EnvironmentSatisfaction   669 non-null    int64 
 11  Gender                    669 non-null    object
 12  HourlyRate                669 non-null    int64 
 13  JobInvolvement            669 non-null    int64 
 14  JobLevel                  669 non-null    int64 
 15  JobRole                   669 non-null    object
 16  JobSatisfaction           669 non-null    int64 
 17  MaritalStatus             669 non-null    object
 18  MonthlyIncome             669 non-null    int64 
 19  MonthlyRate               669 non-null    int64 
```

```
 20  NumCompaniesWorked        669 non-null    int64 
 21  Over18                    669 non-null    object
 22  OverTime                  669 non-null    object
 23  PercentSalaryHike         669 non-null    int64 
 24  PerformanceRating         669 non-null    int64 
 25  RelationshipSatisfaction  669 non-null    int64 
 26  StandardHours             669 non-null    int64 
 27  StockOptionLevel          669 non-null    int64 
 28  TotalWorkingYears         669 non-null    int64 
 29  TrainingTimesLastYear     669 non-null    int64 
 30  WorkLifeBalance           669 non-null    int64 
 31  YearsAtCompany            669 non-null    int64 
 32  YearsInCurrentRole        669 non-null    int64 
 33  YearsSinceLastPromotion   669 non-null    int64 
 34  YearsWithCurrManager      669 non-null    int64 
dtypes: int64(26), object(9)
memory usage: 183.1+ KB
```

Preprocessing

In [20]:

```
#data cleaning, preprocessing to get data for model implementation

df_train['CanBePromoted'] = df_train['CanBePromoted'].replace({'Yes':1,'No':0})
target_label = df_train['CanBePromoted']
print(target_label)
```

```
0      0
1      1
2      0
3      1
4      0
      ..
796    1
797    1
798    1
799    1
800    0
Name: CanBePromoted, Length: 801, dtype: int64
```

In [21]:

```
#dropping columns with constant value and less significance

df_train = df_train.drop(columns=['CanBePromoted'],axis=1)
```

In [22]:

```
#verfying dataset
df_train
```

Out[22]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educa
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	
...	...	...	...	...	...	...	...	

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educa
796	25	Yes	Travel_Rarely	1219	Research & Development	4	1	
797	26	Yes	Travel_Rarely	1330	Research & Development	21	3	
798	33	Yes	Travel_Rarely	1017	Research & Development	25	3	
799	42	No	Travel_Rarely	469	Research & Development	2	2	
800	28	Yes	Travel_Frequently	1009	Research & Development	1	3	

801 rows × 35 columns

In [23]:
```
#concat train and test data, preprocess the data - apply model - predict test set la

hr_data = pd.concat([df_train,df_test])
```

In [24]:
```
#remove columns with constant values

hr_data = hr_data.drop(columns=['EmployeeCount', 'Over18', 'StandardHours','Employee
```

In [25]:
```
#encoding for variable gender

hr_data['Gender'] = hr_data['Gender'].replace({'Female':1,'Male':0})
```

In [26]:
```
#encoding for variable Attrition

hr_data['Attrition'] = hr_data['Attrition'].replace({'Yes':1,'No':0})
```

In [27]:
```
#encoding for variable Business Travel
hr_data['BusinessTravel'] = hr_data['BusinessTravel'].replace({'Non-Travel':0,'Trave
```

In [28]:
```
#verifying the dataset

hr_data
```

Out[28]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educati
0	41	1	1	1102	Sales	1	2	Life S
1	49	0	2	279	Research & Development	8	1	Life S
2	37	1	1	1373	Research & Development	2	2	
3	33	0	2	1392	Research & Development	3	4	Life S

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	Educati
4	27	0	1	591	Research & Development	2	1	
...	...	...	...	...	...	...	...	
664	36	0	2	884	Research & Development	23	2	
665	39	0	1	613	Research & Development	6	1	
666	27	0	1	155	Research & Development	4	3	Life S
667	49	0	2	1023	Sales	2	3	
668	34	0	1	628	Research & Development	8	3	

1470 rows × 31 columns

In [29]:

```
#encoding remaining categorical data

hr_data = pd.get_dummies(hr_data)
```

In [30]:

```
#checking train data after one-hot encoding

hr_data.shape
```

Out[30]: (1470, 50)

In [31]:

```
#separate train and test

df_train = hr_data.iloc[:train_rows,:]
df_test = hr_data.iloc[train_rows:,:]
```

In [32]:

```
#verify datasets

df_train
```

Out[32]:

	Age	Attrition	BusinessTravel	DailyRate	DistanceFromHome	Education	EnvironmentSatisfacti
0	41	1	1	1102	1	2	
1	49	0	2	279	8	1	
2	37	1	1	1373	2	2	
3	33	0	2	1392	3	4	
4	27	0	1	591	2	1	
...	...	...	...	...	...	...	
796	25	1	1	1219	4	1	
797	26	1	1	1330	21	3	

	Age	Attrition	BusinessTravel	DailyRate	DistanceFromHome	Education	EnvironmentSatisfacti
798	33	1	1	1017	25	3	
799	42	0	1	469	2	2	
800	28	1	2	1009	1	3	

801 rows × 50 columns

In [33]:

```
df_test
```

Out[33]:

	Age	Attrition	BusinessTravel	DailyRate	DistanceFromHome	Education	EnvironmentSatisfacti
0	50	1	2	959	1	4	
1	33	0	2	970	7	3	
2	34	0	0	697	3	4	
3	48	0	0	1262	1	4	
4	45	0	0	1050	9	4	
...	...	...	...	...	...	...	
664	36	0	2	884	23	2	
665	39	0	1	613	6	1	
666	27	0	1	155	4	3	
667	49	0	2	1023	2	3	
668	34	0	1	628	8	3	

669 rows × 50 columns

In [34]:

```
#creating X and y variables - that is X = attributes/features y = target variable

X = df_train
y = target_label
```

In [35]:

```
#verifying X and y variables
X
```

Out[35]:

	Age	Attrition	BusinessTravel	DailyRate	DistanceFromHome	Education	EnvironmentSatisfacti
0	41	1	1	1102	1	2	
1	49	0	2	279	8	1	
2	37	1	1	1373	2	2	
3	33	0	2	1392	3	4	
4	27	0	1	591	2	1	

	Age	Attrition	BusinessTravel	DailyRate	DistanceFromHome	Education	EnvironmentSatisfacti
...	...	...	...	...	...	...	
796	25	1	1	1219	4	1	
797	26	1	1	1330	21	3	
798	33	1	1	1017	25	3	
799	42	0	1	469	2	2	
800	28	1	2	1009	1	3	

801 rows × 50 columns

In [36]:
```
y
```

Out[36]:
```
0      0
1      1
2      0
3      1
4      0
      ..
796    1
797    1
798    1
799    1
800    0
Name: CanBePromoted, Length: 801, dtype: int64
```

In [37]:
```
#split train data into train and validation sets

X_train,X_val,y_train,y_val = train_test_split(X,y,test_size=0.2,shuffle=True)
```

Model Implementation and evaluation

1)Logistic Regression

In [38]:
```
#implementing Logistic Regression

logreg_clf_1 = Pipeline([("scaler", MinMaxScaler()),("logreg_clf", LogisticRegressio
#logreg_clf_2 = Pipeline([("scaler", StandardScaler()),("logreg_clf", LogisticRegres

#calling fit method

logreg_clf_1.fit(X_train,y_train)
```

Out[38]:
```
Pipeline(steps=[('scaler', MinMaxScaler()),
                ('logreg_clf', LogisticRegression())])
```

In [39]:
```
#predict labels for validation set

y_pred_logregclf_1 = logreg_clf_1.predict(X_val)
```

Evaluating Logistic Regression Model using Validation set

In [40]:
```
#for validation set
```

```
logreg_val_acc_score = accuracy_score(y_val,y_pred_logregclf_1)
```

In [41]:

```
print(logreg_val_acc_score,"\n")
print(confusion_matrix(y_val,y_pred_logregclf_1),"\n")
print(classification_report(y_val,y_pred_logregclf_1))
```

```
0.8074534161490683

[[58 19]
 [12 72]]

              precision    recall  f1-score   support

           0       0.83      0.75      0.79        77
           1       0.79      0.86      0.82        84

    accuracy                           0.81       161
   macro avg       0.81      0.81      0.81       161
weighted avg       0.81      0.81      0.81       161
```

In [42]:

```
#cheking train accuracy to check for overfitting

logreg_train = logreg_clf_1.predict(X_train)
print("Train acc :",accuracy_score(y_train,logreg_train))
print("Validation acc : ",accuracy_score(y_val,y_pred_logregclf_1))
```

```
Train acc : 0.8359375
Validation acc :  0.8074534161490683
```

In [43]:

```
#performing cross validation

cv = StratifiedKFold(n_splits=10,shuffle=True)
logreg_scores = cross_val_score(logreg_clf_1,X,y,scoring='accuracy',cv=cv)
print(logreg_scores)
print('Accuracy: %.3f (%.3f)' % (np.mean(logreg_scores), np.std(logreg_scores)))
```

```
[0.77777778 0.825      0.825      0.775      0.7625     0.8
 0.85       0.875      0.8        0.725     ]
Accuracy: 0.802 (0.042)
```

2) Support Vector Machine

In [44]:

```
svm_clf_1 = Pipeline([("scaler", MinMaxScaler()),("svm_clf", SVC())])
#svm_clf_2 = Pipeline([("scaler", StandardScaler()),("svm_clf", SVC())])
svm_clf_1.fit(X_train,y_train)
```

Out[44]:

```
Pipeline(steps=[('scaler', MinMaxScaler()), ('svm_clf', SVC())])
```

In [45]:

```
y_pred_svm_1 = svm_clf_1.predict(X_val)
```

Evaluating SVM model

In [46]:

```
print(accuracy_score(y_val,y_pred_svm_1), "\n")
print(confusion_matrix(y_val,y_pred_svm_1), "\n")
print(classification_report(y_val,y_pred_svm_1))
```

```
0.8012422360248447
```

```
[[59 18]
 [14 70]]

              precision    recall  f1-score   support

           0       0.81      0.77      0.79        77
           1       0.80      0.83      0.81        84

    accuracy                           0.80       161
   macro avg       0.80      0.80      0.80       161
weighted avg       0.80      0.80      0.80       161
```

In [47]:
```
svm_train = svm_clf_1.predict(X_train)
print("Train acc : ", accuracy_score(y_train,svm_train))
print("Validation acc : ", accuracy_score(y_val,y_pred_svm_1))
```

```
Train acc :  0.9109375
Validation acc :  0.8012422360248447
```

In [48]:
```
svm_scores = cross_val_score(svm_clf_1,X,y,scoring='accuracy',cv=cv)
print(svm_scores)
print('Accuracy: %.3f (%.3f)' % (np.mean(svm_scores), np.std(svm_scores)))
```

```
[0.88888889 0.8375     0.7625     0.825      0.8125     0.8625
 0.7375     0.7125     0.825      0.8125    ]
Accuracy: 0.808 (0.052)
```

3)Adaboost

In [49]:
```
#implement Adaboost classifier

adaboost_clf = AdaBoostClassifier()
adaboost_clf.fit(X_train,y_train)
```

Out[49]:
```
AdaBoostClassifier()
```

In [50]:
```
y_pred_adaboostclf = adaboost_clf.predict(X_val)
```

Evaluating Adaboost model

In [51]:
```
print(accuracy_score(y_val,y_pred_adaboostclf),"\n")
print(confusion_matrix(y_val,y_pred_adaboostclf),"\n")
print(classification_report(y_val,y_pred_adaboostclf))
```

```
0.8198757763975155

[[60 17]
 [12 72]]

              precision    recall  f1-score   support

           0       0.83      0.78      0.81        77
           1       0.81      0.86      0.83        84

    accuracy                           0.82       161
   macro avg       0.82      0.82      0.82       161
weighted avg       0.82      0.82      0.82       161
```

In [52]:
```
#compare train and validation accuracy to check for overfitting
```

```
adaboost_train = adaboost_clf.predict(X_train)
print("Training acc : ", accuracy_score(y_train,adaboost_train))
print("Validation acc : ", accuracy_score(y_val,y_pred_adaboostclf))
```

```
Training acc :  0.884375
Validation acc :  0.8198757763975155
```

In [53]:

```
adaboost_scores = cross_val_score(adaboost_clf,X,y,scoring='accuracy',cv=cv)
print(adaboost_scores)
print('Accuracy: %.3f (%.3f)' % (np.mean(adaboost_scores), np.std(adaboost_scores)))
```

```
[0.80246914 0.75       0.7875     0.85       0.775      0.8375
 0.8125     0.7625     0.875      0.775     ]
Accuracy: 0.803 (0.039)
```

In [54]:

```
acc = []; prec = []; recall = []
models = ['LogisticRegression','SVM','Adaboost']
```

In [55]:

```
acc.append(accuracy_score(y_val,y_pred_logregclf_1))
prec.append(precision_score(y_val,y_pred_logregclf_1))
recall.append(recall_score(y_val,y_pred_logregclf_1))
```

In [56]:

```
acc.append(accuracy_score(y_val,y_pred_svm_1))
prec.append(precision_score(y_val,y_pred_svm_1))
recall.append(recall_score(y_val,y_pred_svm_1))
```

In [57]:

```
acc.append(accuracy_score(y_val,y_pred_adaboostclf))
prec.append(precision_score(y_val,y_pred_adaboostclf))
recall.append(recall_score(y_val,y_pred_adaboostclf))
```

In [58]:

```
compare = pd.concat([pd.Series(models),pd.Series(acc),pd.Series(prec),pd.Series(reca
                    ,axis=1)
compare.columns = ['Models','Accuracy','Precision','Recall']
compare
```

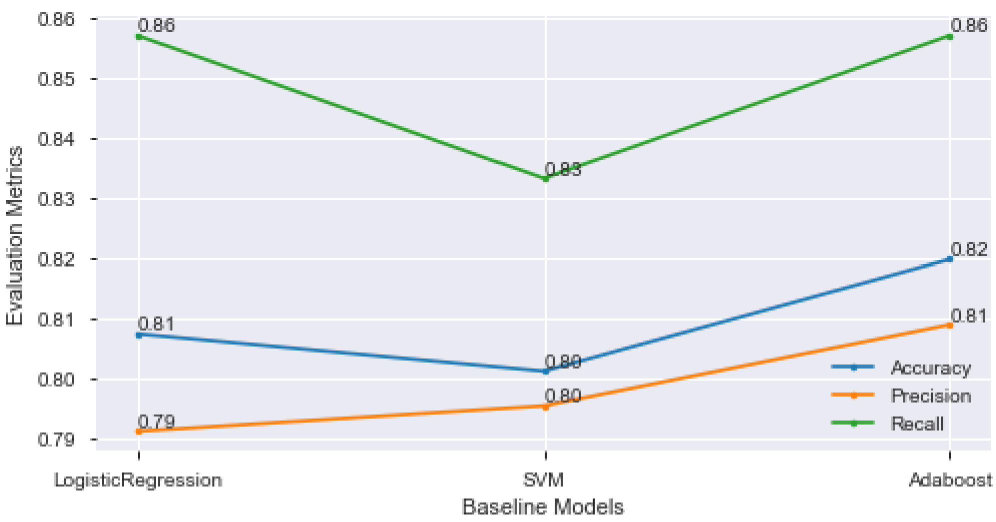
Out[58]:

	Models	Accuracy	Precision	Recall
0	LogisticRegression	0.807453	0.791209	0.857143
1	SVM	0.801242	0.795455	0.833333
2	Adaboost	0.819876	0.808989	0.857143

In [59]:

```
%matplotlib inline
pd.set_option('display.max_rows',None)
pd.set_option('display.max_columns',None)
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt

plt.subplots(figsize=(8,4))
plt.plot(compare.Models,compare.Accuracy,marker = '.')
plt.plot(compare.Models,compare.Precision,marker = '.')
plt.plot(compare.Models,compare.Recall,marker = '.')
plt.legend(('Accuracy','Precision','Recall'))
```

```
for x,y in zip(compare.Models,compare.Accuracy):

    label = "{:.2f}".format(y)

    plt.annotate(label, # this is the text
                 (x,y), # these are the coordinates to position the label
                 textcoords="offset points",
                xytext=(0,2))

for x,y in zip(compare.Models,compare.Precision):

    label = "{:.2f}".format(y)

    plt.annotate(label, # this is the text
                 (x,y), # these are the coordinates to position the label
                 textcoords="offset points",
                xytext=(0,2))

for x,y in zip(compare.Models,compare.Recall):

    label = "{:.2f}".format(y)

    plt.annotate(label, # this is the text
                 (x,y), # these are the coordinates to position the label
                 textcoords="offset points",
                xytext=(0,2))


plt.xlabel('Baseline Models')
plt.ylabel('Evaluation Metrics')
plt.show()
```



In [60]:

```
#since adaboost seems to provide better results - predicting labels using adaboost

predicted_labels = adaboost_clf.predict(df_test)
```

In [61]:

```
#converting to dataframe and change int to string

predicted_labels = pd.DataFrame(predicted_labels)
```

In [62]:

```
predicted_labels = predicted_labels.replace({0:'No', 1:'Yes'})
```

In [63]:
```
#creating file

predicted_labels.to_csv('Predicted_Promotion.csv',header='CanBePromoted',index=False
```

In [64]:
```
#comparing predicted with validation set

y_val
```

Out[64]:
```
264    0
504    0
570    0
795    0
714    1
448    1
501    1
773    1
640    1
89     0
329    0
545    1
124    0
439    0
330    0
591    1
235    1
212    1
279    0
309    1
287    0
635    0
533    1
35     1
310    1
473    0
585    0
357    0
147    0
45     1
659    0
517    1
252    0
515    0
709    0
539    1
381    0
10     0
463    0
46     1
243    0
383    1
278    0
186    0
114    0
142    1
225    0
684    0
178    1
98     1
58     1
311    0
637    0
340    1
737    1
165    0
643    0
528    0
516    0
```

```
314    1
651    0
529    1
399    1
296    0
491    1
560    0
475    1
385    1
710    1
542    0
393    0
43     1
759    0
191    1
302    0
797    1
177    0
125    1
113    0
328    1
271    1
308    1
612    0
240    1
265    0
0      0
630    0
619    0
74     0
49     0
231    1
669    1
209    1
437    0
229    0
126    1
604    1
607    0
595    1
486    0
91     1
221    1
118    0
361    1
194    1
656    1
41     0
343    0
549    1
796    1
761    0
565    0
606    1
534    1
130    1
732    0
20     1
281    1
663    0
605    1
133    1
581    1
467    0
777    0
578    1
174    0
540    1
503    1
```

```
196    1
725    1
544    1
728    1
580    0
233    0
156    0
713    1
692    0
316    0
592    1
220    1
543    1
249    1
579    1
298    1
752    1
374    1
512    1
105    1
653    1
344    1
620    1
181    1
184    0
6      0
128    0
107    1
85     0
485    1
522    0
453    0
331    0
Name: CanBePromoted, dtype: int64
```

In [65]:

```
y_pred_adaboostclf
```

Out[65]:

```
array([0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1,
       1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0,
       0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0,
       0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 0, 0, 1, 0, 0, 0], dtype=int64)
```

In [66]:

```
predicted_labels
```

Out[66]:

	0
0	No
1	Yes
2	No
3	Yes
4	Yes
5	Yes
6	No
7	No
8	No

	0
9	Yes
10	No
11	Yes
12	Yes
13	No
14	Yes
15	No
16	No
17	No
18	No
19	Yes
20	No
21	Yes
22	Yes
23	No
24	Yes
25	Yes
26	Yes
27	No
28	No
29	No
30	Yes
31	Yes
32	Yes
33	No
34	No
35	No
36	No
37	No
38	No
39	No
40	Yes
41	Yes
42	Yes
43	No
44	Yes

	0
45	No
46	Yes
47	No
48	No
49	No
50	Yes
51	No
52	No
53	Yes
54	Yes
55	No
56	Yes
57	No
58	No
59	No
60	No
61	No
62	Yes
63	No
64	Yes
65	No
66	Yes
67	Yes
68	No
69	Yes
70	No
71	No
72	Yes
73	No
74	Yes
75	Yes
76	Yes
77	No
78	No
79	Yes
80	Yes

	0
81	Yes
82	Yes
83	No
84	No
85	Yes
86	No
87	No
88	Yes
89	No
90	Yes
91	No
92	No
93	No
94	Yes
95	Yes
96	Yes
97	No
98	No
99	Yes
100	No
101	No
102	No
103	No
104	No
105	Yes
106	Yes
107	Yes
108	No
109	Yes
110	No
111	Yes
112	No
113	Yes
114	Yes
115	Yes
116	Yes

	0
117	No
118	Yes
119	Yes
120	Yes
121	Yes
122	Yes
123	Yes
124	No
125	Yes
126	Yes
127	Yes
128	No
129	No
130	No
131	No
132	No
133	Yes
134	Yes
135	No
136	Yes
137	Yes
138	No
139	No
140	No
141	No
142	Yes
143	No
144	No
145	No
146	Yes
147	Yes
148	No
149	No
150	No
151	No
152	Yes

	0
153	No
154	Yes
155	No
156	No
157	Yes
158	Yes
159	Yes
160	No
161	No
162	Yes
163	Yes
164	Yes
165	Yes
166	Yes
167	Yes
168	No
169	No
170	No
171	No
172	No
173	Yes
174	No
175	No
176	Yes
177	Yes
178	No
179	No
180	No
181	Yes
182	Yes
183	No
184	Yes
185	No
186	No
187	No
188	Yes

	0
189	No
190	No
191	Yes
192	No
193	No
194	No
195	Yes
196	Yes
197	Yes
198	Yes
199	No
200	No
201	No
202	No
203	Yes
204	No
205	No
206	No
207	Yes
208	Yes
209	Yes
210	Yes
211	No
212	No
213	No
214	Yes
215	No
216	No
217	Yes
218	No
219	No
220	No
221	No
222	Yes
223	No
224	Yes

	0
225	No
226	No
227	No
228	No
229	No
230	No
231	No
232	No
233	No
234	Yes
235	No
236	Yes
237	No
238	Yes
239	Yes
240	Yes
241	Yes
242	No
243	No
244	No
245	No
246	No
247	Yes
248	Yes
249	Yes
250	No
251	No
252	No
253	No
254	Yes
255	No
256	No
257	Yes
258	No
259	No
260	No

	0
261	No
262	Yes
263	No
264	No
265	Yes
266	Yes
267	No
268	Yes
269	Yes
270	No
271	Yes
272	No
273	No
274	Yes
275	No
276	Yes
277	Yes
278	No
279	Yes
280	Yes
281	No
282	No
283	Yes
284	No
285	Yes
286	Yes
287	No
288	No
289	Yes
290	Yes
291	Yes
292	Yes
293	Yes
294	No
295	Yes
296	No

	0
297	Yes
298	No
299	Yes
300	No
301	Yes
302	No
303	No
304	Yes
305	Yes
306	No
307	No
308	No
309	Yes
310	No
311	No
312	No
313	Yes
314	No
315	Yes
316	No
317	No
318	Yes
319	No
320	No
321	Yes
322	Yes
323	No
324	Yes
325	No
326	Yes
327	Yes
328	Yes
329	Yes
330	No
331	Yes
332	No

	0
333	Yes
334	No
335	Yes
336	No
337	Yes
338	Yes
339	Yes
340	No
341	No
342	No
343	Yes
344	No
345	No
346	Yes
347	Yes
348	No
349	Yes
350	Yes
351	Yes
352	No
353	Yes
354	Yes
355	Yes
356	Yes
357	No
358	Yes
359	No
360	Yes
361	Yes
362	No
363	Yes
364	Yes
365	Yes
366	Yes
367	No
368	No

	0
369	No
370	No
371	Yes
372	Yes
373	Yes
374	Yes
375	No
376	Yes
377	No
378	Yes
379	No
380	No
381	Yes
382	No
383	Yes
384	Yes
385	No
386	Yes
387	No
388	Yes
389	Yes
390	No
391	No
392	Yes
393	No
394	No
395	Yes
396	No
397	Yes
398	No
399	No
400	No
401	No
402	Yes
403	Yes
404	No

	0
405	Yes
406	Yes
407	Yes
408	Yes
409	Yes
410	Yes
411	No
412	Yes
413	No
414	Yes
415	No
416	Yes
417	No
418	Yes
419	Yes
420	Yes
421	No
422	Yes
423	Yes
424	Yes
425	No
426	Yes
427	No
428	No
429	No
430	Yes
431	No
432	No
433	No
434	No
435	No
436	No
437	No
438	Yes
439	Yes
440	Yes

	0
441	No
442	Yes
443	Yes
444	Yes
445	Yes
446	Yes
447	Yes
448	No
449	Yes
450	No
451	No
452	Yes
453	No
454	No
455	No
456	No
457	No
458	No
459	No
460	No
461	No
462	Yes
463	No
464	Yes
465	Yes
466	Yes
467	Yes
468	No
469	No
470	No
471	No
472	No
473	No
474	Yes
475	No
476	No

	0
477	Yes
478	Yes
479	No
480	Yes
481	Yes
482	Yes
483	Yes
484	Yes
485	Yes
486	No
487	Yes
488	Yes
489	Yes
490	No
491	Yes
492	No
493	No
494	No
495	No
496	Yes
497	Yes
498	Yes
499	Yes
500	Yes
501	No
502	Yes
503	No
504	No
505	Yes
506	Yes
507	No
508	No
509	No
510	No
511	No
512	Yes

	0
513	Yes
514	Yes
515	Yes
516	Yes
517	No
518	No
519	Yes
520	Yes
521	Yes
522	No
523	Yes
524	No
525	No
526	Yes
527	Yes
528	Yes
529	Yes
530	Yes
531	Yes
532	Yes
533	Yes
534	No
535	Yes
536	Yes
537	No
538	No
539	No
540	No
541	Yes
542	No
543	Yes
544	Yes
545	Yes
546	Yes
547	No
548	No

	0
549	No
550	No
551	Yes
552	Yes
553	Yes
554	No
555	Yes
556	No
557	Yes
558	Yes
559	No
560	No
561	Yes
562	No
563	Yes
564	No
565	Yes
566	Yes
567	No
568	No
569	Yes
570	No
571	Yes
572	Yes
573	No
574	No
575	No
576	Yes
577	No
578	No
579	Yes
580	No
581	Yes
582	Yes
583	No
584	Yes

	0
585	Yes
586	No
587	No
588	Yes
589	No
590	No
591	Yes
592	No
593	Yes
594	Yes
595	No
596	Yes
597	Yes
598	No
599	No
600	No
601	Yes
602	No
603	Yes
604	Yes
605	Yes
606	No
607	Yes
608	No
609	Yes
610	No
611	Yes
612	Yes
613	Yes
614	Yes
615	No
616	Yes
617	Yes
618	Yes
619	No
620	No

	0
621	No
622	Yes
623	No
624	No
625	Yes
626	No
627	No
628	No
629	No
630	Yes
631	No
632	Yes
633	No
634	Yes
635	Yes
636	No
637	No
638	Yes
639	No
640	No
641	No
642	No
643	Yes
644	Yes
645	Yes
646	No
647	Yes
648	Yes
649	No
650	Yes
651	Yes
652	No
653	Yes
654	Yes
655	Yes
656	Yes

	0
657	Yes
658	No
659	No
660	No
661	No
662	No
663	Yes
664	No
665	No
666	No
667	Yes
668	No

In [ ]:

```

```

In [ ]:

```

```