

```
# Importing Required Libraries
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

```
data=pd.read_excel("/content/drive/MyDrive/Colab Notebooks/HR_Data.xlsb",engine='pyxlsb')
```

```
data.sample(5)
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours
11668	0.62	0.73	3	
4247	0.86	0.82	4	
2405	0.18	0.75	3	
2480	0.56	0.90	3	

```
data.shape
```

```
(14999, 10)
```

```
# check for null values
data.isnull().sum()
```

```
satisfaction_level      0
last_evaluation         0
number_project          0
average_montly_hours   0
time_spend_company     0
Work_accident          0
left                    0
promotion_last_5years  0
Department             0
salary                  0
dtype: int64
```

```
data.info()
```

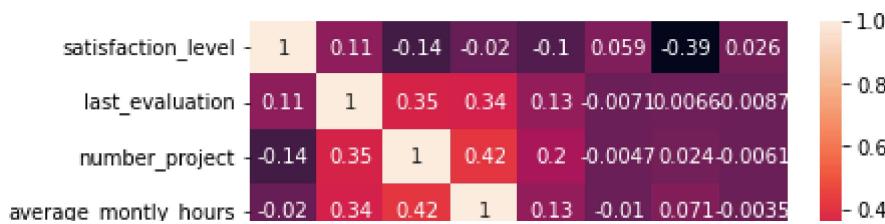
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
```

```
Data columns (total 10 columns):
 #   Column            Non-Null Count  Dtype  
 --- 
 0   satisfaction_level    14999 non-null   float64
 1   last_evaluation      14999 non-null   float64
 2   number_project       14999 non-null   int64  
 3   average_montly_hours 14999 non-null   int64  
 4   time_spend_company   14999 non-null   int64  
 5   Work_accident        14999 non-null   int64  
 6   left                 14999 non-null   int64  
 7   promotion_last_5years 14999 non-null   int64  
 8   Department           14999 non-null   object  
 9   salary                14999 non-null   object  
dtypes: float64(2), int64(6), object(2)
memory usage: 1.1+ MB
```

```
data.describe()
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_
count	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000
mean	0.612834	0.716102	3.803054	201.050337	
std	0.248631	0.171169	1.232592	49.943099	
min	0.090000	0.360000	2.000000	96.000000	
25%	0.440000	0.560000	3.000000	156.000000	
50%	0.640000	0.720000	4.000000	200.000000	
75%	0.820000	0.870000	5.000000	245.000000	
max	1.000000	1.000000	7.000000	310.000000	

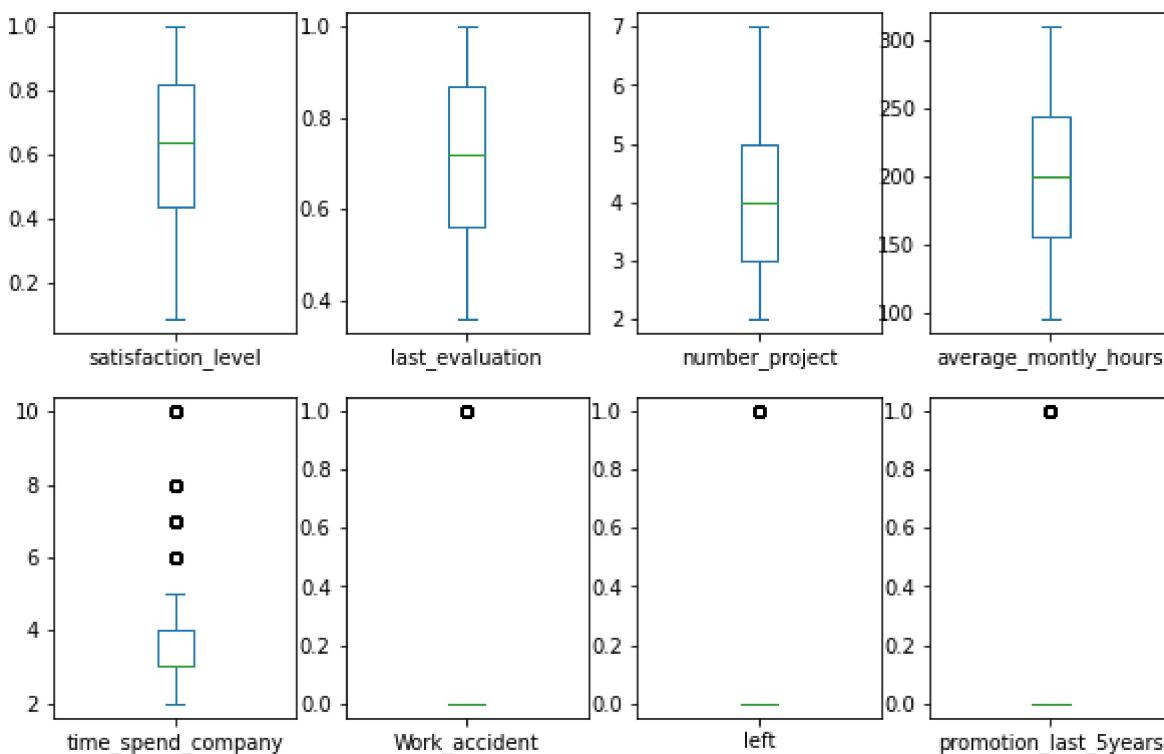
```
# correlation of independent features with dependent features
sns.heatmap(data.corr(), annot=True);
```



▼ 1. EDA

```
left -0.39 0.0066 0.024 0.071 0.14 -0.15 1 -0.062
```

```
# Handling the outliers
data.plot(kind='box', subplots=True, figsize=(10,10), layout=(3,4))
plt.show()
```



Very few outliers are detected, So there is no need to handle the outliers

```
chr_=data.select_dtypes("object")
chr_.columns

Index(['Department', 'salary'], dtype='object')

print(data["Department"].unique())
print()
print("unique_val : ",data["Department"].nunique())

['sales' 'accounting' 'hr' 'technical' 'support' 'management' 'IT'
 'product_mng' 'marketing' 'RandD']
```

```
unique_val : 10
```

```
print(data["salary"].unique())
print()
print("unique_val :",data["salary"].nunique())
```

```
['low' 'medium' 'high']
```

```
unique_val : 3
```

Handling The Categorical Data

```
data['salary']= data['salary'].map({'high':2, 'medium':1, 'low':0})
```

```
data['salary'].unique()
```

```
array([0, 1, 2])
```

```
data['Department']=data['Department'].map({'sales':0, 'accounting':1, 'hr':2,
'technical':3, 'support':4, 'management':5,'IT':6, 'product_mng':7,
'marketing':8, 'RandD':9})
```

```
data['Department'].unique()
```

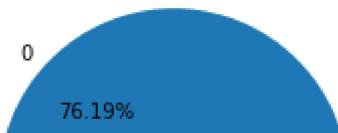
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Analysis of Dependent Feature

```
data['left'].value_counts()
```

```
0    11428
1    3571
Name: left, dtype: int64
```

```
plt.pie(data["left"].value_counts(),labels=data['left'].value_counts().index,autopct='%.1f%%')
```



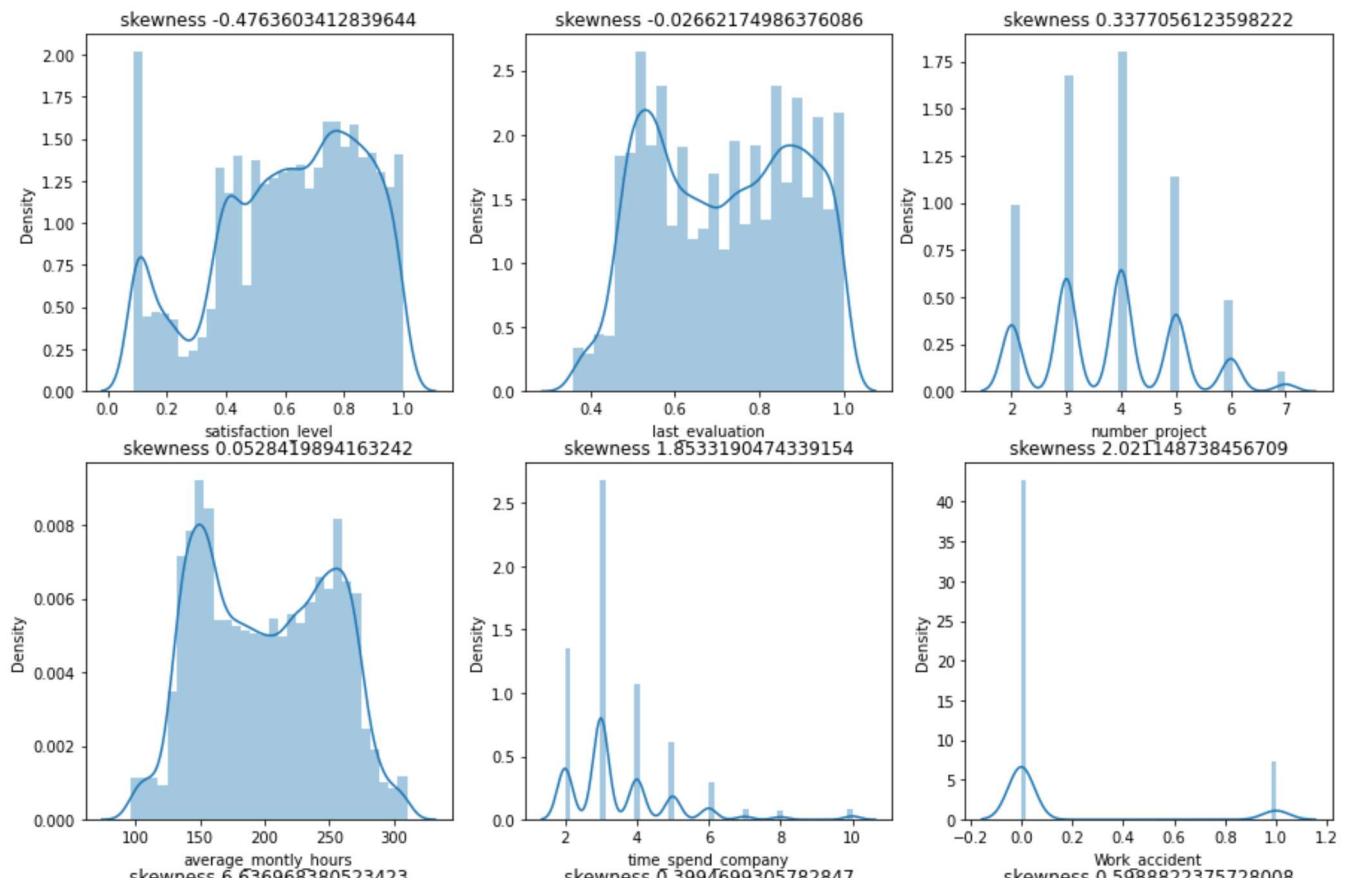
```
data.head(3)
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spen
0	0.38	0.53	2	157	
1	0.80	0.86	5	262	
2	0.11	0.88	7	272	



```
from scipy.stats import skew
```

```
k=0
plt.figure(figsize=(15,15))
for col in data.drop('left',axis=1):
    k=k+1
    plt.subplot(3,3,k)
    sns.distplot(data[col])
    plt.title("skewness"+' '+str(data[col].skew()))
```



```
# visualization of correlation of features & label
sns.pairplot(data,hue='left');
```



▼ 2. Model Building

```
X = data.drop("left", axis=1)
y = data.left
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import *
```

▼ LogisticRegression

```
from sklearn.linear_model import LogisticRegression
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=0)
lr=LogisticRegression()
lr.fit(X_train,y_train)

LogisticRegression()

y_pred=lr.predict(X_test)

lr.score(X_train,y_train)

0.7917896942565958

lr.score(X_test,y_test)

0.792
```

▼ GridSearchCV

```
from sklearn.model_selection import GridSearchCV,cross_val_score,StratifiedKFold

lr=LogisticRegression()
solvers = ['lbfgs', 'liblinear','newton-cg']
penalty = ['l2','l1"]
c_values = [100, 10, 1.0, 0.1, 0.01]

fold=StratifiedKFold(n_splits=5,shuffle=True,random_state=0)
grid = dict(solver=solvers,penalty=penalty,C=c_values)

model=GridSearchCV(estimator=lr,
                    param_grid=grid,
                    scoring='accuracy',
                    verbose=1,cv=fold)

model.fit(X_train, y_train)

print(model.best_params_)
print(model.best_score_)
print(model.best_estimator_)

Fitting 5 folds for each of 30 candidates, totalling 150 fits
{'C': 100, 'penalty': 'l2', 'solver': 'lbfgs'}
0.7969326890355952
LogisticRegression(C=100)
```

```
new_lr=LogisticRegression(C=100, solver='liblinear')
new_lr.fit(X_train,y_train)
y_pred=new_lr.predict(X_test)
```

```

lr_score=new_lr.score(X_test,y_test)
precision_score_lr=(precision_score(y_test,y_pred))
recall_score_lr=(recall_score(y_test,y_pred))
f1_score_lr=(f1_score(y_test,y_pred))
roc_auc_score_lr=(roc_auc_score(y_test,y_pred))

print("logistic_accuracy_score :",lr_score)
print("prcision :",precision_score_lr)
print("recall :",recall_score_lr)
print("f1_score :",f1_score_lr)
print("roc_auc_score :",roc_auc_score_lr)

logistic_accuracy_score : 0.7917777777777778
prcision : 0.5790297339593115
recall : 0.35645472061657035
f1_score : 0.4412641621943948
roc_auc_score : 0.639376984802797

cm = np.array(confusion_matrix(y_test, y_pred, labels=[1,0]))
confusion = pd.DataFrame(cm, index=['left', 'not left'],columns=['predicted left','predicted not left'])

```

	predicted left	predicted not left
left	370	668
not left	269	3193

```
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.83	0.92	0.87	3462
1	0.58	0.36	0.44	1038
accuracy			0.79	4500
macro avg	0.70	0.64	0.66	4500
weighted avg	0.77	0.79	0.77	4500

▼ Support Vector

```

from sklearn.svm import SVC
svc = SVC().fit(X_train,y_train)

y_pred=svc.predict(X_test)

```

```
svc_score=accuracy_score(y_test,y_pred)
svc_score
```

```
0.7902222222222223
```

▼ GridSearchCV

```
# fold=StratifiedKFold(n_splits=5,shuffle=True,random_state=0)
kernel=['linear', 'rbf']
C=[0.1,0.01,0.001,1]
grid = dict(kernel=kernel ,C=C)
gs_model=GridSearchCV(estimator=SVC(),param_grid=grid,cv=fold,verbose=1,return_train_score=True)
gs_model.fit(X_train,y_train)
print(gs_model.best_score_)
print(gs_model.best_params_)
print(gs_model.best_estimator_)
```

```
svc=SVC(C=100, kernel='rbf').fit(X_train,y_train)
y_pred=svc.predict(X_test)
svc_score=accuracy_score(y_test,y_pred)
print(svc_score)
```

```
0.9051111111111111
```

```
precision_score_svc=(precision_score(y_test,y_pred))
recall_score_svc=(recall_score(y_test,y_pred))
f1_score_svc=(f1_score(y_test,y_pred))
roc_auc_score_svc=(roc_auc_score(y_test,y_pred))
```

```
print("SVC_accuracy_score :",svc_score)
print("prcision :",precision_score_svc)
print("recall :",recall_score_svc)
print("f1_score :",f1_score_svc)
print("roc_auc_score :",roc_auc_score_svc)
```

```
SVC_accuracy_score : 0.9051111111111111
prcision : 0.8375690607734807
recall : 0.7302504816955684
f1_score : 0.7802367472979927
roc_auc_score : 0.8438947382481308
```

```
cm = np.array(confusion_matrix(y_test, y_pred, labels=[1,0]))
```

```
confusion = pd.DataFrame(cm, index=['left', 'not left'], columns=['predicted left', 'predicted not left'])
```

	predicted left	predicted not left
left	758	280
not left	147	3315

```
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.92	0.96	0.94	3462
1	0.84	0.73	0.78	1038
accuracy			0.91	4500
macro avg	0.88	0.84	0.86	4500
weighted avg	0.90	0.91	0.90	4500

▼ KNeighborsClassifier

```
from sklearn.neighbors import KNeighborsClassifier
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)

n_neighbors=[5,10,15,20,25,50,75,100]
params=dict(n_neighbors=n_neighbors)
gs_model=GridSearchCV(estimator=KNeighborsClassifier(),param_grid=params,scoring='accuracy',cv=5)
gs_model.fit(X_train,y_train)
print(gs_model.best_score_)
print(gs_model.best_params_)
print(gs_model.best_estimator_)
```

```
Fitting 5 folds for each of 8 candidates, totalling 40 fits
0.930660066694456
{'n_neighbors': 5}
KNeighborsClassifier()
```

```
knn=KNeighborsClassifier().fit(X_train,y_train)
y_pred=knn.predict(X_test)
knn_score=accuracy_score(y_test,y_pred)
knn_score
```

```
0.926
```

```
precision_score_knn=(precision_score(y_test,y_pred))
recall_score_knn=(recall_score(y_test,y_pred))
```

```
f1_score_knn=(f1_score(y_test,y_pred))
roc_auc_score_knn=(roc_auc_score(y_test,y_pred))

print("knn_accuracy_score :",knn_score)
print("precision :",precision_score_knn)
print("recall :",recall_score_knn)
print("f1_score :",f1_score_knn)
print("roc_auc_score :",roc_auc_score_knn)
```

```
knn_accuracy_score : 0.926
precision : 0.8047858942065491
recall : 0.9050991501416431
f1_score : 0.8520000000000001
roc_auc_score : 0.9187657912870377
```

```
cm = np.array(confusion_matrix(y_test, y_pred, labels=[1,0]))
confusion = pd.DataFrame(cm, index=['left', 'not left'], columns=['predicted left', 'predicted not left'])

          predicted left  predicted not left
left              639                  67
not left           155                2139
```

```
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.97	0.93	0.95	2294
1	0.80	0.91	0.85	706
accuracy			0.93	3000
macro avg	0.89	0.92	0.90	3000
weighted avg	0.93	0.93	0.93	3000

▼ NaiveBayes

```
from sklearn.naive_bayes import GaussianNB
nb=GaussianNB().fit(X_train,y_train)
y_pred_=nb.predict(X_test)
nb_score=accuracy_score(y_test,y_pred_)
nb_score

0.792

precision_score_nb=(precision_score(y_test,y_pred_))
recall_score_nb=(recall_score(y_test,y_pred_))
```

```
f1_score_nb=(f1_score(y_test,y_pred_))
roc_auc_score_nb=(roc_auc_score(y_test,y_pred_))

print("nb_accuracy_score :",nb_score)
print("precision :",precision_score_nb)
print("recall :",recall_score_nb)
print("f1_score :",f1_score_nb)
print("roc_auc_score :",roc_auc_score_nb)

nb_accuracy_score : 0.792
precision : 0.5441810344827587
recall : 0.7152974504249292
f1_score : 0.6181150550795593
roc_auc_score : 0.7654516894670418
```

```
cm = np.array(confusion_matrix(y_test, y_pred_, labels=[1,0]))
confusion = pd.DataFrame(cm, index=['left', 'not left'], columns=['predicted left','predicted not left'])

          predicted left  predicted not left
left              505                  201
not left           423                 1871
```

```
print(classification_report(y_test,y_pred_))
```

	precision	recall	f1-score	support
0	0.90	0.82	0.86	2294
1	0.54	0.72	0.62	706
accuracy			0.79	3000
macro avg	0.72	0.77	0.74	3000
weighted avg	0.82	0.79	0.80	3000

▼ DecisionTree

```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier().fit(X_train,y_train)
dt.score(X_test,y_test)
```

0.976

GridSearchCV

```

dt=DecisionTreeClassifier()
param_grid = {'max_features': ['auto', 'sqrt', 'log2'],
              'ccp_alpha': [0.1, .01, .001],
              'max_depth' : [2,4,6,8,10],
              'criterion' :['gini', 'entropy'],
              'min_samples_split':[2,4,6],
              'min_samples_leaf':[1,2,3]
            }
gs_model=GridSearchCV(estimator=dt,param_grid=param_grid,scoring='accuracy',verbose=1,cv=fold
gs_model.fit(X_train,y_train)
print(gs_model.best_score_)
print(gs_model.best_estimator_)

Fitting 5 folds for each of 810 candidates, totalling 4050 fits
0.97699732527442
DecisionTreeClassifier(ccp_alpha=0.001, criterion='entropy', max_depth=10,
                      max_features='log2', min_samples_leaf=2)

dtree=DecisionTreeClassifier(ccp_alpha=0.001, criterion='entropy', max_depth=10,max_features=
dtree.fit(X_train,y_train)
y_pred=dtree.predict(X_test)

dtree_score=accuracy_score(y_test,y_pred)
dtree_score

0.9656666666666667

precision_score_dt=(precision_score(y_test,y_pred))
recall_score_dt=(recall_score(y_test,y_pred))
f1_score_dt=(f1_score(y_test,y_pred))
roc_auc_score_dt=(roc_auc_score(y_test,y_pred))

print("dt_accuracy_score :",dtree_score)
print("prcision :",precision_score_dt)
print("recall :",recall_score_dt)
print("f1_score :",f1_score_dt)
print("roc_auc_score :",roc_auc_score_dt)

dt_accuracy_score : 0.9656666666666667
prcision : 0.886685552407932
recall : 0.886685552407932
f1_score : 0.886685552407932
roc_auc_score : 0.925905984573626

cm = np.array(confusion_matrix(y_test, y_pred, labels=[1,0]))
confusion = pd.DataFrame(cm, index=['left', 'not left'],columns=['predicted left','predicted
confusion

```

	predicted left	predicted not left
left	626	80
not left	80	2214

```
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.97	0.97	0.97	2294
1	0.89	0.89	0.89	706
accuracy			0.95	3000
macro avg	0.93	0.93	0.93	3000
weighted avg	0.95	0.95	0.95	3000

▼ RandomForest

```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier().fit(X_train,y_train)
rf.score(X_test,y_test)

0.9886666666666667
```

GridSearchCV

```
rf=RandomForestClassifier()
param_grid = {'max_features': ['auto', 'sqrt'],
              'n_estimators': [100,150],
              'max_depth' : [2,6,10],
              'criterion' :['gini', 'entropy'],
              'min_samples_split':[2,4,6],
              'min_samples_leaf':[1,2,3]
             }

gs_model=GridSearchCV(estimator=rf,param_grid=param_grid,scoring='accuracy',verbose=1,cv=fold
gs_model.fit(X_train,y_train)
print(gs_model.best_score_)
print(gs_model.best_params_)
print(gs_model.best_estimator_)

Fitting 5 folds for each of 216 candidates, totalling 1080 fits
0.9815815270251494
{'criterion': 'gini', 'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 1, 'n
RandomForestClassifier(max_depth=10)
```

```
rforest=RandomForestClassifier(criterion='gini', max_depth=10, max_features='auto',
                               min_samples_split=2, n_estimators=100).fit(X_train,y_train)
```

```
y_pred=rforest.predict(X_test)
```

```
rf_score=accuracy_score(y_test,y_pred)
rf_score
```

0.979

```
precision_score_rf=(precision_score(y_test,y_pred))
recall_score_rf=(recall_score(y_test,y_pred))
f1_score_rf=(f1_score(y_test,y_pred))
roc_auc_score_rf=(roc_auc_score(y_test,y_pred))
```

```
print("rf_accuracy_score :",rf_score)
print("precision :",precision_score_rf)
print("recall :",recall_score_rf)
print("f1_score :",f1_score_rf)
print("roc_auc_score :",roc_auc_score_rf)
```

```
rf_accuracy_score : 0.979
precision : 0.989345509893455
recall : 0.9206798866855525
f1_score : 0.9537784299339692
roc_auc_score : 0.9588142240751214
```

```
cm = np.array(confusion_matrix(y_test, y_pred, labels=[1,0]))
confusion = pd.DataFrame(cm, index=['left', 'not left'],columns=['predicted left','predicted not left'])
```

	predicted left	predicted not left
left	650	56
not left	7	2287

```
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.98	1.00	0.99	2294
1	0.99	0.92	0.95	706
accuracy			0.98	3000
macro avg	0.98	0.96	0.97	3000

weighted avg	0.98	0.98	0.98	3000
--------------	------	------	------	------

▼ AdaBoostClassifier

```
from sklearn.ensemble import AdaBoostClassifier
ada_clf = AdaBoostClassifier(RandomForestClassifier())
ada_clf.fit(X_train, y_train)

AdaBoostClassifier(base_estimator=RandomForestClassifier())

y_pred=ada_clf.predict(X_test)

ada_score=accuracy_score(y_test,y_pred)
ada_score

0.988

precision_score_ada=(precision_score(y_test,y_pred))
recall_score_ada=(recall_score(y_test,y_pred))
f1_score_ada=(f1_score(y_test,y_pred))
roc_auc_score_ada=(roc_auc_score(y_test,y_pred))

print("ada_accuracy_score :",ada_score)
print("prcision :",precision_score_ada)
print("recall :",recall_score_ada)
print("f1_score :",f1_score_ada)
print("roc_auc_score :",roc_auc_score_ada)

ada_accuracy_score : 0.988
prcision : 0.9883720930232558
recall : 0.9631728045325779
f1_score : 0.9756097560975608
roc_auc_score : 0.9798427231032549

cm = np.array(confusion_matrix(y_test, y_pred, labels=[1,0]))
confusion = pd.DataFrame(cm, index=['left', 'not left'],columns=['predicted left','predicted not left'])

predicted left predicted not left
-----
left           680            26
not left         8          2286

print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.99	1.00	0.99	2294
1	0.99	0.96	0.98	706
accuracy			0.99	3000
macro avg	0.99	0.98	0.98	3000
weighted avg	0.99	0.99	0.99	3000

▼ XGBoost

```

from xgboost import XGBClassifier
xgb_clf =XGBClassifier(max_depth=5, n_estimators=10000, learning_rate=0.3,n_jobs=-1)
xgb_clf.fit(X_train, y_train)

XGBClassifier(learning_rate=0.3, max_depth=5, n_estimators=10000, n_jobs=-1)

y_pred=xgb_clf.predict(X_test)

xgb_score=accuracy_score(y_test,y_pred)
xgb_score

0.9846666666666667

precision_score_xgb=(precision_score(y_test,y_pred))
recall_score_xgb=(recall_score(y_test,y_pred))
f1_score_xgb=(f1_score(y_test,y_pred))
roc_auc_score_xgb=(roc_auc_score(y_test,y_pred))

print("xgb_accuracy_score :",xgb_score)
print("prcision :",precision_score_xgb)
print("recall :",recall_score_xgb)
print("f1_score :",f1_score_xgb)
print("roc_auc_score :",roc_auc_score_xgb)

xgb_accuracy_score : 0.9846666666666667
prcision : 0.9714285714285714
recall : 0.9631728045325779
f1_score : 0.9672830725462305
roc_auc_score : 0.977227204358704

cm = np.array(confusion_matrix(y_test, y_pred, labels=[1,0]))
confusion = pd.DataFrame(cm, index=['left', 'not left'],columns=['predicted left','predicted
confusion

```

	predicted left	predicted not left
left	680	26
not left	20	2274

```
print(classification_report(y_test,y_pred))

          precision    recall  f1-score   support

           0       0.99      0.99      0.99     2294
           1       0.97      0.96      0.97      706

    accuracy                           0.98     3000
   macro avg       0.98      0.98      0.98     3000
weighted avg       0.98      0.98      0.98     3000
```

▼ Voting_Classifier

```
from sklearn.ensemble import VotingClassifier
log_clf = LogisticRegression(random_state=42)
rnd_clf = RandomForestClassifier(random_state=42)
svm_clf = SVC(random_state=42, probability=True)

voting_clf = VotingClassifier(
    estimators=[('lr', log_clf), ('rf', rnd_clf), ('svc', svm_clf)],
    voting='soft')

voting_clf.fit(X_train, y_train)

VotingClassifier(estimators=[('lr', LogisticRegression(random_state=42)),
                            ('rf', RandomForestClassifier(random_state=42)),
                            ('svc', SVC(probability=True, random_state=42))],
                 voting='soft')

for clf in (log_clf, rnd_clf, svm_clf, voting_clf):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    print(clf.__class__.__name__, accuracy_score(y_test, y_pred))

LogisticRegression 0.7776666666666666
RandomForestClassifier 0.988
SVC 0.7846666666666666
VotingClassifier 0.925

voting_clf.fit(X_train, y_train)
y_pred = voting_clf.predict(X_test)
```

```
vtc_score=accuracy_score(y_test, y_pred)
vtc_score
```

```
precision_score_vtc=(precision_score(y_test,y_pred))
recall_score_vtc=(recall_score(y_test,y_pred))
f1_score_vtc=(f1_score(y_test,y_pred))
roc_auc_score_vtc=(roc_auc_score(y_test,y_pred))
```

```
print("vtc_accuracy_score :",vtc_score)
print("prcision :",precision_score_vtc)
print("recall :",recall_score_vtc)
print("f1_score :",f1_score_vtc)
print("roc_auc_score :",roc_auc_score_vtc)
```

```
vtc_accuracy_score : 0.925
prcision : 0.9461966604823747
recall : 0.7223796033994334
f1_score : 0.8192771084337348
roc_auc_score : 0.8548689647337184
```

```
cm = np.array(confusion_matrix(y_test, y_pred, labels=[1,0]))
confusion = pd.DataFrame(cm, index=['left', 'not left'],columns=['predicted left','predicted not left'])
```

	predicted left	predicted not left
left	510	196
not left	29	2265

```
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.92	0.99	0.95	2294
1	0.95	0.72	0.82	706
accuracy			0.93	3000
macro avg	0.93	0.85	0.89	3000
weighted avg	0.93	0.93	0.92	3000

▼ Result Table

```
result=pd.DataFrame({
```

```
'model':[ 'LogisticRegression', 'SVC', 'KNeighborsClassifier', 'GaussianNB',
'DecisionTree', 'RandomForest', 'AdaBoostClassifier', 'XGBoost', 'Voting_Classifier'],

'Accuracy_score':[lr_score,svc_score,knn_score,nb_score,dtree_score,rf_score,ada_score,xgb_sc

'Precision_score':[precision_score_lr,precision_score_svc,precision_score_knn,precision_score_
precision_score_dt,precision_score_rf,precision_score_ada,precision_score_


'recall_score':[recall_score_lr,recall_score_svc,recall_score_knn,recall_score_nb,recall_scor
recall_score_rf,recall_score_ada,recall_score_xgb,recall_score_vtc],


'f1_score':[f1_score_lr,f1_score_svc,f1_score_knn,f1_score_nb,f1_score_dt,f1_score_rf,f1_scor

'roc_auc_score':[roc_auc_score_lr,roc_auc_score_svc,roc_auc_score_knn,roc_auc_score_nb,
roc_auc_score_dt,roc_auc_score_rf,roc_auc_score_ada,roc_auc_score_xgb,roc_auc_sco

},index=range(1,10))

result = result.round(decimals = 4)
result
```

	model	Accuracy_score	Precision_score	recall_score	f1_score	roc_auc_
1	LogisticRegression	0.7918	0.5790	0.3565	0.4413	C
2	SVC	0.9051	0.8376	0.7303	0.7802	C
3	KNeighborsClassifier	0.9260	0.8048	0.9051	0.8520	C
4	GaussianNB	0.7920	0.5442	0.7153	0.6181	C
5	DecisionTree	0.9657	0.8867	0.8867	0.8867	C
6	RandomForest	0.9790	0.9893	0.9207	0.9538	C
7	AdaBoostClassifier	0.9880	0.9884	0.9632	0.9756	C
8	XGBoost	0.9847	0.9714	0.9632	0.9673	C
9	Voting_Classifier	0.9250	0.9462	0.7224	0.8193	C

◀ ▶

```
# Maximum scores from all models
max_score =result.iloc[:,1:6].max()
max_score
```

```
Accuracy_score      0.9880
Precision_score     0.9893
recall_score        0.9632
f1_score            0.9756
roc_auc_score       0.9798
dtype: float64
```

Base on the above maximum scores,Final Best Model for these Dataset is AdaBoostClassifier

```
from sklearn.ensemble import AdaBoostClassifier
ada_clf = AdaBoostClassifier(RandomForestClassifier())
ada_clf.fit(X_train, y_train)
y_pred=ada_clf.predict(X_test)
```

▼ 3. Prediction on Test Data

```
# comparing actual values with predicted values
result=pd.DataFrame({'Actual':y_test,"Predicted":y_pred})
result.sample(10)
```

	Actual	Predicted
11346	0	0
14578	1	1
13379	0	0
4863	0	0
4921	0	0
9808	0	0
4062	0	0
1783	1	1
4192	0	0
14041	0	0

By observing the above prediction we can say that our Model is Best for the Accurate Prediction

Also, I can conclude that it gives the almost 99% correct prediction

▼ 4. Prediction on new features

```
data.head(1)
```

```
satisfaction_level  last_evaluation  number_project  average_montly_hours  time_spen
```

```
0           0.38          0.53            2             157
```

```
data.columns
```

```
↳ Index(['satisfaction_level', 'last_evaluation', 'number_project',  
         'average_montly_hours', 'time_spend_company', 'Work_accident', 'left',  
         'promotion_last_5years', 'Department', 'salary'],  
        dtype='object')
```

```
def prediction(satisfaction_level, last_evaluation, number_project, average_montly_hours, time_sp  
corr_pred=int(ada_clf.predict([[satisfaction_level, last_evaluation, number_project, average  
if corr_pred == 0:  
    print("Employee will not Left")  
else:  
    print("Employee will Left")
```

```
prediction(0.77,0.76,4,250,4,5,0,3,0)
```

```
Employee will not Left
```

```
prediction(0.10,0.1,5,250,1,10,0,0,0)
```

```
Employee will Left
```