In [541]:

```
import pandas as pd
import numpy as np
import datetime as dt
import matplotlib.pyplot as plt
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_absolute_error
from pandas.tseries.offsets import DateOffset
import datetime as dt
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
import xgboost as xgb
from  sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.metrics import mean_absolute_error as mae
from sklearn.metrics import r2_score
import lightgbm as lgb
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.model_selection import RandomizedSearchCV
# scikit-learn classifiers
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassif
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
from sklearn.neural_network import MLPClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis, Quadra

import lightgbm as lgb
import xgboost as xgb
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f
import imblearn
from collections import Counter
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline
from collections import Counter
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
```

In [837]:

```
file=r"C:\Users\Madhujit\Downloads\WA_Fn-UseC_-HR-Employee-Attrition.csv"
df=pd.read_csv(file)
```

In [838]:

```
df.isnull().sum()
```

Out[838]:

```
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
dtype: int64
```

In [839]:

```
df.duplicated().sum()
```

Out[839]:

```
0
```

In [903]:
```
df
```

Out[903]:

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	Envi
0	41	1102	1	2	1	1	
1	49	279	8	1	1	2	
2	37	1373	2	2	1	4	
3	33	1392	3	4	1	5	
4	27	591	2	1	1	7	
...	...	...	...	...	...	...	
1465	36	884	23	2	1	2061	
1466	39	613	6	1	1	2062	
1467	27	155	4	3	1	2064	
1468	49	1023	2	3	1	2065	
1469	34	628	8	3	1	2068	

1470 rows × 27 columns

In [840]:
```
col=['BusinessTravel', 'Department', 'EducationField', 'Gender',
       'JobRole', 'MaritalStatus', 'Over18', 'OverTime']
```

In [841]:
```
one_hot_encoding=pd.get_dummies(df[col])
```

In [842]:
```
colls=[i for i in df.columns if i not in col]
```

In [843]:
```
df=df[colls]
```

In [844]:
```
df["Att"]=df["Attrition"].apply(lambda x:1 if x=="Yes" else 0)
```

In [845]:
```
df.drop(["Attrition"],axis=1,inplace=True)
```

In [857]:
```
final_df=pd.concat([df,one_hot_encoding],axis=1)
```

In [858]:
```
for i in final_df.columns:
    if final_df[i].nunique()<2:
        print(f'"{i}",')
```

```
"EmployeeCount",
"StandardHours",
"Over18_Y",
```

In [859]:
```
col2=["EmployeeCount",
"StandardHours",
"Over18_Y"]
```

In [860]:
```
final_df=final_df.drop(columns=col2)
```

In [913]:

```
for i in final_df.columns:
    print(f'{i},{final_df[i].nunique()}')
col0=["DailyRate","EmployeeNumber","MonthlyIncome","MonthlyRate"]
```

```
Age,43
DailyRate,886
DistanceFromHome,29
Education,5
EmployeeNumber,1470
EnvironmentSatisfaction,4
HourlyRate,71
JobInvolvement,4
JobLevel,5
JobSatisfaction,4
MonthlyIncome,1349
MonthlyRate,1427
NumCompaniesWorked,10
PercentSalaryHike,15
PerformanceRating,2
RelationshipSatisfaction,4
StockOptionLevel,4
TotalWorkingYears,40
TrainingTimesLastYear,7
WorkLifeBalance,4
YearsAtCompany,37
YearsInCurrentRole,19
YearsSinceLastPromotion,16
YearsWithCurrManager,18
Att,2
BusinessTravel_Non-Travel,2
BusinessTravel_Travel_Frequently,2
BusinessTravel_Travel_Rarely,2
Department_Human Resources,2
Department_Research & Development,2
Department_Sales,2
EducationField_Human Resources,2
EducationField_Life Sciences,2
EducationField_Marketing,2
EducationField_Medical,2
EducationField_Other,2
EducationField_Technical Degree,2
Gender_Female,2
Gender_Male,2
JobRole_Healthcare Representative,2
JobRole_Human Resources,2
JobRole_Laboratory Technician,2
JobRole_Manager,2
JobRole_Manufacturing Director,2
JobRole_Research Director,2
JobRole_Research Scientist,2
JobRole_Sales Executive,2
JobRole_Sales Representative,2
MaritalStatus_Divorced,2
MaritalStatus_Married,2
MaritalStatus_Single,2
OverTime_No,2
OverTime_Yes,2
```

In [914]:
```
final_df=final_df.drop(columns=col0)
```

In [920]:
```
colls=final_df.drop(["Att"],axis=1).columns
```

In [922]:
```
colls=['Age', 'DistanceFromHome', 'Education', 'EnvironmentSatisfaction',
       'HourlyRate', 'JobInvolvement', 'JobLevel', 'JobSatisfaction',
       'NumCompaniesWorked', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StockOptionLevel', 'TotalWorkingYears',
       'TrainingTimesLastYear', 'WorkLifeBalance', 'YearsAtCompany',
       'YearsInCurrentRole', 'YearsSinceLastPromotion', 'YearsWithCurrManage
       'BusinessTravel_Non-Travel', 'BusinessTravel_Travel_Frequently',
       'BusinessTravel_Travel_Rarely', 'Department_Human Resources',
       'Department_Research & Development', 'Department_Sales',
       'EducationField_Human Resources', 'EducationField_Life Sciences',
       'EducationField_Marketing', 'EducationField_Medical',
       'EducationField_Other', 'EducationField_Technical Degree',
       'Gender_Female', 'Gender_Male', 'JobRole_Healthcare Representative',
       'JobRole_Human Resources', 'JobRole_Laboratory Technician',
       'JobRole_Manager', 'JobRole_Manufacturing Director',
       'JobRole_Research Director', 'JobRole_Research Scientist',
       'JobRole_Sales Executive', 'JobRole_Sales Representative',
       'MaritalStatus_Divorced', 'MaritalStatus_Married',
       'MaritalStatus_Single', 'OverTime_No', 'OverTime_Yes']
```

In [923]:
```
one_hot_encode2=pd.get_dummies(final_df[colls],columns=colls)
```

In [924]:
```
one_hot_encode2
```

Out[924]:

	Age_18	Age_19	Age_20	Age_21	Age_22	Age_23	Age_24	Age_25	Age_26	Age_27
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	1
...	...	...	...	...	...	...	...	...	...	...
1465	0	0	0	0	0	0	0	0	0	0
1466	0	0	0	0	0	0	0	0	0	0
1467	0	0	0	0	0	0	0	0	0	1
1468	0	0	0	0	0	0	0	0	0	0
1469	0	0	0	0	0	0	0	0	0	0

1470 rows × 397 columns

In [925]:
```
final_df1=final_df.drop(columns=colls)
```

In [936]:
```
final_df2=pd.concat([final_df1,one_hot_encode2],axis=1)
```

In [990]:

```
for name ,values in final_df2.drop(["Att"],axis=1).corrwith(final_df2["Att"]
    if values>0:
        print(f'"{values}",')
```

```
"0.00035957134043653616",
"0.0009474130499167443",
"0.0009474130499167495",
"0.0016107239840919482",
"0.00164773444336447",
"0.0016477344433645023",
"0.0016477344433645214",
"0.0016477344433645394",
"0.001647734443364553",
"0.0019065831481129124",
"0.0019065831481129257",
"0.002136061030578993",
"0.0028887517110808276",
"0.0036432779351901583",
"0.004037963021174343",
"0.00583879617929592",
"0.006160065558844793",
"0.006160065558844835",
"0.006172096379851281",
```

In [960]:

```
col4=["BusinessTravel_Travel_Frequently_1",
"JobInvolvement_1",
"EnvironmentSatisfaction_1",
"JobRole_Sales Representative_1",
"YearsInCurrentRole_0",
"MaritalStatus_Single_1",
"YearsAtCompany_1",
"StockOptionLevel_0",
"YearsWithCurrManager_0",
"JobLevel_1",
"TotalWorkingYears_1",
"OverTime_No_0",
"OverTime_Yes_1","Att"]
```

In [961]:

```
final_df3=final_df2[col4]
```

In [991]:

```
x=final_df3.drop(["Att"],axis=1)
y=final_df3["Att"]
```

In [992]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
```

In [993]:

```
sc=StandardScaler()
```

In [994]:

```
x_train_s=sc.fit_transform(x_train)
x_test_s=sc.transform(x_test)
```

In [1021]:

```
reg= RandomForestClassifier()
```

In [1022]:

```
model=reg.fit(x_train_s,y_train)
```

In [1023]:

```
y_pred=reg.predict(x_test_s)
```

In [1024]:

```
accuracy_score(y_test,y_pred)
```

Out[1024]:

```
0.8559782608695652
```

In [1012]:

```

param_grid = {
    'penalty': ['l1', 'l2'],  # Penalty norm
    'C': [0.001, 0.01, 0.1, 1, 10, 100],  # Inverse of regularization streng
    'solver': ['liblinear', 'saga']  # Algorithm to use in the optimization
}


LOg_reg=LogisticRegression()
```

In [1013]:

```
randomsearch=RandomizedSearchCV(LOg_reg,param_grid,n_iter=200,cv=50,scoring=
```

In [1014]:

```
randomsearch.fit(x_train_s,y_train)
```

```
C:\Users\Madhujit\anaconda3\Lib\site-packages\sklearn\model_selection\_sear
ch.py:305: UserWarning: The total space of parameters 24 is smaller than n_
iter=200. Running 24 iterations. For exhaustive searches, use GridSearchCV.
  warnings.warn(
```

Out[1014]:

```
RandomizedSearchCV(cv=50, estimator=LogisticRegression(), n_iter=200,
                   param_distributions={'C': [0.001, 0.01, 0.1, 1, 10, 10
0],
                                        'penalty': ['l1', 'l2'],
                                        'solver': ['liblinear', 'saga']},
                   scoring='neg_mean_absolute_error')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [1015]:

```
randomsearch.best_params_
```

Out[1015]:

```
{'solver': 'liblinear', 'penalty': 'l2', 'C': 10}
```

In [1017]:

```
LOg_reg=LogisticRegression(solver= 'liblinear', penalty= 'l2', C= 10)
```

In [1018]:

```
model=LOg_reg.fit(x_train_s,y_train)
```

In [1019]:

```
y_pred=model.predict(x_test_s)
```

In [1020]:

```
accuracy_score(y_test,y_pred)
```

Out[1020]:

```
0.8777173913043478
```

In [ ]:

In [ ]: