# **Importing all the necessary libraries and models**

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.metrics **import** confusion\_matrix

**from** sklearn **import** datasets

**from** sklearn.metrics **import** accuracy\_score

**from** sklearn.metrics **import** plot\_confusion\_matrix

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.tree **import** DecisionTreeClassifier

**from** sklearn.ensemble **import** RandomForestClassifier

**from** sklearn.naive\_bayes **import** GaussianNB

**from** sklearn.neighbors **import** KNeighborsClassifier

**from** sklearn **import** svm

**from** sklearn.metrics **import** classification\_report

# **Pandas DataFrame**

df**=**pd**.**read\_csv(r"C:\Users\dushy\Downloads\WA\_Fn-UseC\_-HR-Employee-Attrition (1).csv")

df**.**head()

# **data preprocessing**

df**.**drop(0,inplace**=True**)

df**.**isnull()**.**sum()

df**.**dropna(axis**=**0,inplace**=True**)

# **Feature Engineering**

# **One hot encoding**

**def** onehot\_encode(df, column):

df **=** df**.**copy()

dummies **=** pd**.**get\_dummies(df[column], prefix**=**column)

df **=** pd**.**concat([df, dummies], axis**=**1)

df **=** df**.**drop(column, axis**=**1)

**return** df

df **=** df**.**drop(['EmployeeCount', 'EmployeeNumber', 'Over18', 'StandardHours'], axis**=**1)

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

df['OverTime'] **=** df['OverTime']**.**replace({'No': 0, 'Yes': 1})

*# Ordinal-encode the BusinessTravel column*

df['BusinessTravel'] **=** df['BusinessTravel']**.**replace({'Non-Travel': 0, 'Travel\_Rarely': 1, 'Travel\_Frequently': 2})

**for** column **in** ['Department', 'EducationField', 'JobRole', 'MaritalStatus']:

df **=** onehot\_encode(df, column**=**column)

attrition\_dict **=** df["Attrition"]**.**value\_counts()

sns**.**set\_style('darkgrid')

sns**.**countplot(x **=**'Attrition', data **=** df)

sns**.**lmplot(x **=** 'Age', y **=** 'DailyRate', hue **=** 'Attrition', data **=** df)

plt**.**figure(figsize **=**(10, 6))

sns**.**boxplot(y **=**'MonthlyIncome', x **=**'Attrition', data **=** df)

# **Models used:**

lr**=**LogisticRegression(C **=** 0.1, random\_state **=** 42, solver **=** 'liblinear')

dt**=**DecisionTreeClassifier()

rm**=**RandomForestClassifier()

gnb**=**GaussianNB()

knn **=** KNeighborsClassifier(n\_neighbors**=**3)

svm **=** svm**.**SVC(kernel**=**'linear')

# **Training And Testing Data Split**:

y **=** df['Attrition']

X **=** df**.**drop('Attrition', axis**=**1)

X\_train,X\_test,y\_train,y\_test **=** train\_test\_split(X,y,test\_size**=**0.2,random\_state**=**4)

X1**=**X\_train

X\_train**.**head()

# **Fitting Various Models With Train And Test Data**

**for** a,b **in** zip([lr,dt,knn,svm,rm,gnb],["Logistic Regression","Decision Tree","KNN","SVM","Random Forest","Naive Bayes"]):

a**.**fit(X\_train,y\_train)

prediction**=**a**.**predict(X\_train)

y\_pred**=**a**.**predict(X\_test)

score1**=**accuracy\_score(y\_train,prediction)

score**=**accuracy\_score(y\_test,y\_pred)

msg1**=**"[%s] training data accuracy is : %f" **%** (b,score1)

msg2**=**"[%s] test data accuracy is : %f" **%** (b,score)

print(msg1)

print(msg2)

[Logistic Regression] training data accuracy is : 0.876596

[Logistic Regression] test data accuracy is : 0.880952

[Decision Tree] training data accuracy is : 1.000000

[Decision Tree] test data accuracy is : 0.799320

[KNN] training data accuracy is : 0.880851

[KNN] test data accuracy is : 0.802721

[SVM] training data accuracy is : 0.859574

[SVM] test data accuracy is : 0.829932

[Random Forest] training data accuracy is : 1.000000

[Random Forest] test data accuracy is : 0.860544

[Naive Bayes] training data accuracy is : 0.787234

[Naive Bayes] test data accuracy is : 0.727891

# **Highest Accuracy in Training Data: Random Forest**

# **2nd Highest Accuracy : Logistic Regression**

# **3rd Highest Accuracy: KNN**