NAME : SK SHAREEF

REG NO: 21BCE9384

**Data Preprocesing **

```
#Import the Libraries.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
#Importing the dataset.
df=pd.read_csv("employee.csv")
```

```
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educati
0	41	Yes	Travel_Rarely	1102	Sales	1	
1	49	No	Travel_Frequently	279	Research & Development	8	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	
3	33	No	Travel_Frequently	1392	Research & Development	3	
4	27	No	Travel_Rarely	591	Research & Development	2	

5 rows × 35 columns

```
df.shape
```

```
(1470, 35)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       1470 non-null   int64 
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64 
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64 
 6   Education                 1470 non-null   int64 
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64 
 9   EmployeeNumber            1470 non-null   int64 
 10  EnvironmentSatisfaction   1470 non-null   int64 
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64 
 13  JobInvolvement            1470 non-null   int64 
 14  JobLevel                  1470 non-null   int64 
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64 
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64 
 19  MonthlyRate               1470 non-null   int64 
 20  NumCompaniesWorked        1470 non-null   int64 
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64 
 24  PerformanceRating         1470 non-null   int64 
 25  RelationshipSatisfaction  1470 non-null   int64 
 26  StandardHours             1470 non-null   int64 
 27  StockOptionLevel          1470 non-null   int64 
 28  TotalWorkingYears         1470 non-null   int64 
 29  TrainingTimesLastYear     1470 non-null   int64 
 30  WorkLifeBalance           1470 non-null   int64 
 31  YearsAtCompany            1470 non-null   int64 
 32  YearsInCurrentRole        1470 non-null   int64 
```

```
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

```
df.describe()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyRate
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000000
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.891156
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.329428
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000000
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.000000
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000000
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750000
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000000

8 rows × 26 columns

```
#Checking for Null Values.
df.isnull().any()
```

```
Age                         False
Attrition                   False
BusinessTravel              False
DailyRate                   False
Department                  False
DistanceFromHome            False
Education                   False
EducationField              False
EmployeeCount               False
EmployeeNumber              False
EnvironmentSatisfaction     False
Gender                      False
HourlyRate                  False
JobInvolvement              False
JobLevel                    False
JobRole                     False
JobSatisfaction             False
MaritalStatus               False
MonthlyIncome               False
MonthlyRate                 False
NumCompaniesWorked          False
Over18                      False
OverTime                    False
PercentSalaryHike           False
PerformanceRating           False
RelationshipSatisfaction    False
StandardHours               False
StockOptionLevel            False
TotalWorkingYears           False
TrainingTimesLastYear       False
WorkLifeBalance             False
YearsAtCompany              False
YearsInCurrentRole          False
YearsSinceLastPromotion     False
YearsWithCurrManager        False
dtype: bool
```

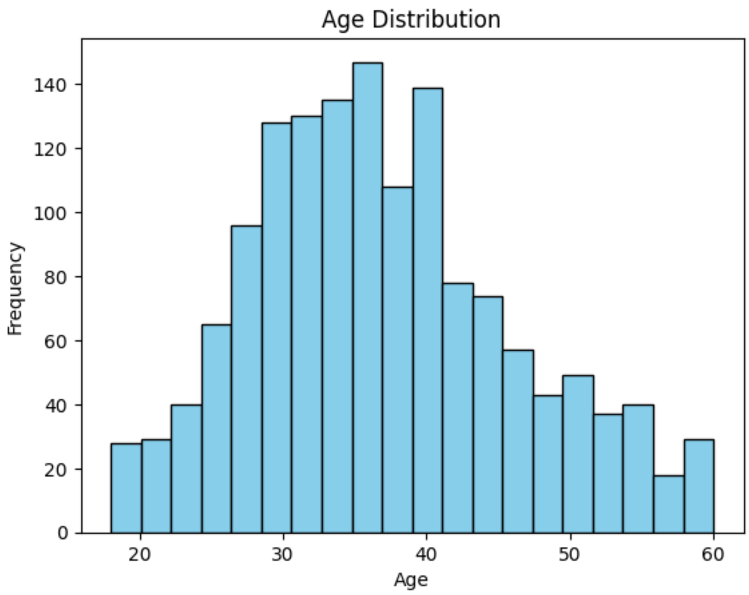
```
df.isnull().sum()
```

```
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
```

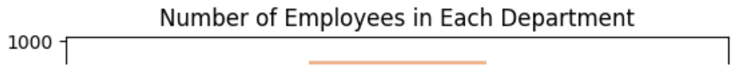
```
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

DATA VISUALISATION

```
#histogram of Age
plt.hist(df['Age'], bins=20, color='skyblue', edgecolor='black')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Age Distribution')
plt.show()
```

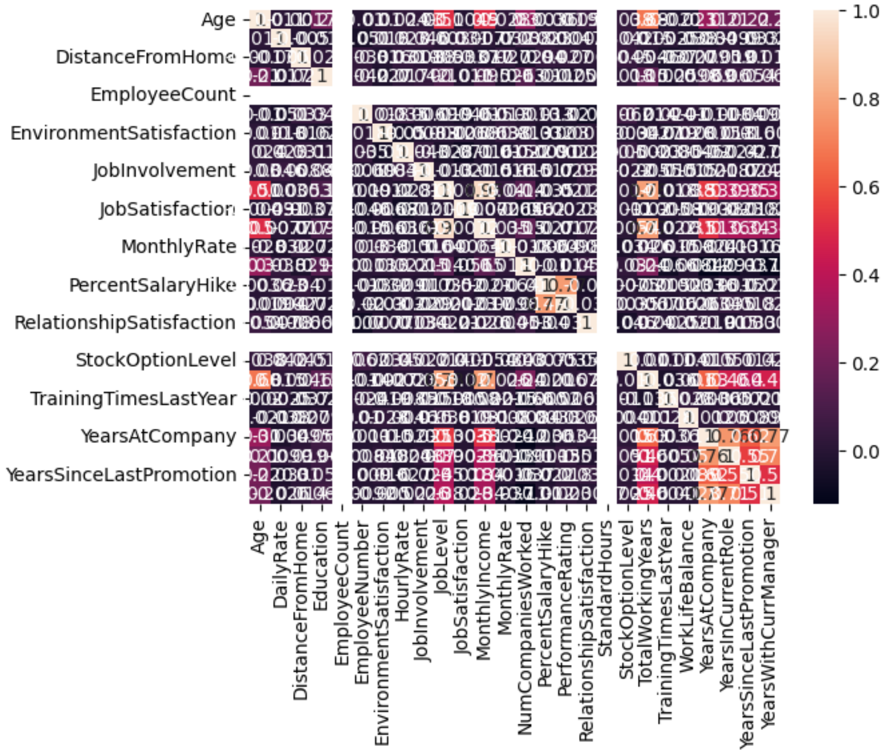



```
# countplot
sns.countplot(x='Department', data=df, palette='pastel')
plt.xlabel('Department')
plt.ylabel('Count')
plt.title('Number of Employees in Each Department')
plt.show()
```




```
sns.heatmap(df.corr(),annot=True)
```

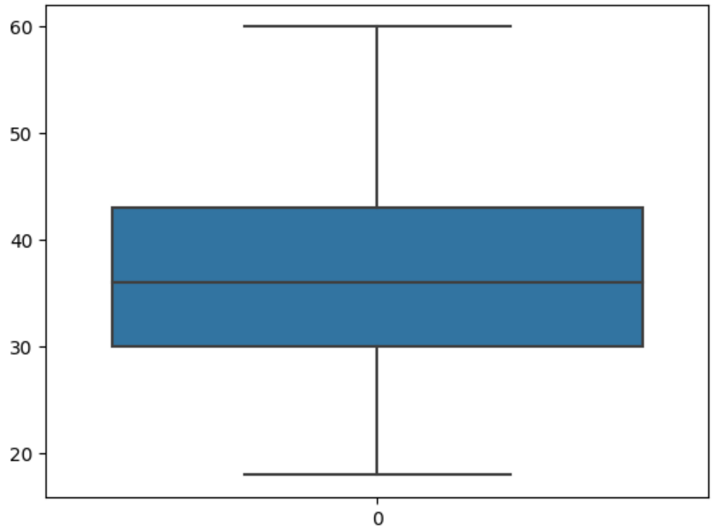
```
<ipython-input-34-8df7bcac526d>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ve
  sns.heatmap(df.corr(),annot=True)
<Axes: >
```



OUTLIERS **DETECTION**

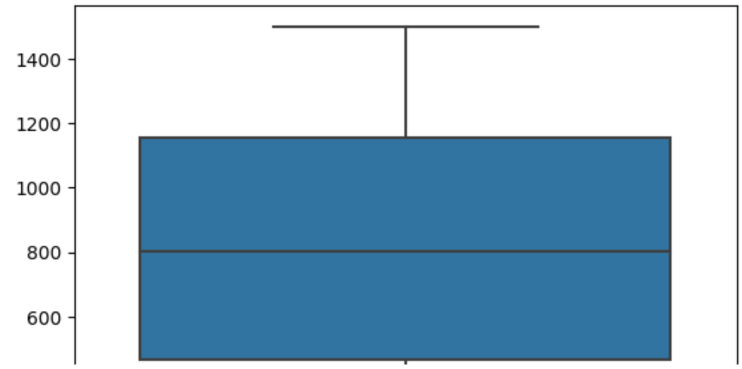
```
#checking for age column
sns.boxplot(df.Age)
```

```
<Axes: >
```



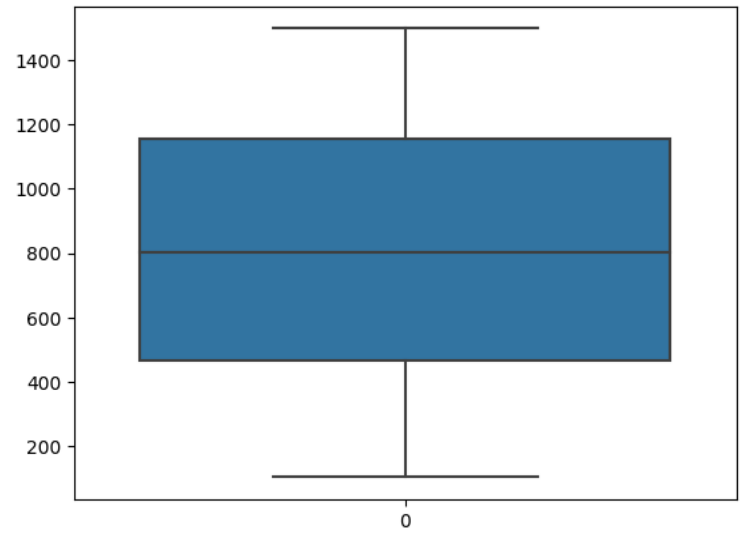

```
#Checking for Daily rate
sns.boxplot(df.DailyRate)
```

```
<Axes: >
```




```
#checking for DailyRate
sns.boxplot(df.DailyRate)
```

```
<Axes: >
```




```
df.head(3)
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumb
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	

3 rows × 35 columns

```
#checking for standardhours
sns.boxplot(df.StandardHours)
```

```
<Axes: >
```

No outliers found so we continue with next step

ENCODING

```
from sklearn.preprocessing import LabelEncoder
le  =  LabelEncoder()
```

```
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumbe
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	

5 rows × 35 columns

```
#perfomring label encoding some satisfied columns
columns = ['BusinessTravel', 'Department', 'Gender', 'Over18', 'OverTime']
df[columns]=df[columns].apply(le.fit_transform)
```

```
#performing one hot encoding on some satisfied columns
one_hot_columns = ['JobRole', 'EducationField', 'MaritalStatus']
df = pd.get_dummies(df, columns=one_hot_columns, drop_first=True)
```

```
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSa
0	41	Yes	2	1102	2	1	2	1	1	
1	49	No	1	279	1	8	1	1	2	
2	37	Yes	2	1373	1	2	2	1	4	
3	33	No	1	1392	1	3	4	1	5	
4	27	No	2	591	1	2	1	1	7	

5 rows × 47 columns

```
df.columns
```

```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EmployeeCount', 'EmployeeNumber',
       'EnvironmentSatisfaction', 'Gender', 'HourlyRate', 'JobInvolvement',
       'JobLevel', 'JobSatisfaction', 'MonthlyIncome', 'MonthlyRate',
       'NumCompaniesWorked', 'Over18', 'OverTime', 'PercentSalaryHike',
       'PerformanceRating', 'RelationshipSatisfaction', 'StandardHours',
       'StockOptionLevel', 'TotalWorkingYears', 'TrainingTimesLastYear',
       'WorkLifeBalance', 'YearsAtCompany', 'YearsInCurrentRole',
       'YearsSinceLastPromotion', 'YearsWithCurrManager',
       'JobRole_Human Resources', 'JobRole_Laboratory Technician',
       'JobRole_Manager', 'JobRole_Manufacturing Director',
       'JobRole_Research Director', 'JobRole_Research Scientist',
       'JobRole_Sales Executive', 'JobRole_Sales Representative',
       'EducationField_Life Sciences', 'EducationField_Marketing',
       'EducationField_Medical', 'EducationField_Other',
       'EducationField_Technical Degree', 'MaritalStatus_Married',
       'MaritalStatus_Single'],
      dtype='object')
```

SPLITTING IN TO DEPENDENT AND INDEPENDENT

```
# Independent variables (features)
x = df[['Age', 'DailyRate', 'DistanceFromHome', 'Education', 'EmployeeCount', 'EmployeeNumber',
        'EnvironmentSatisfaction', 'HourlyRate', 'JobInvolvement', 'JobLevel', 'JobSatisfaction',
        'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked', 'PercentSalaryHike',
        'PerformanceRating', 'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
        'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance', 'YearsAtCompany',
        'YearsInCurrentRole', 'YearsSinceLastPromotion', 'YearsWithCurrManager',
        'BusinessTravel', 'Department', 'Gender', 'Over18', 'OverTime',
        'JobRole_Human Resources', 'JobRole_Laboratory Technician', 'JobRole_Manager',
        'JobRole_Manufacturing Director', 'JobRole_Research Director', 'JobRole_Research Scientist',
        'JobRole_Sales Executive', 'JobRole_Sales Representative',
        'EducationField_Life Sciences', 'EducationField_Marketing', 'EducationField_Medical',
        'EducationField_Other', 'EducationField_Technical Degree',
        'MaritalStatus_Married', 'MaritalStatus_Single']]

# Dependent variable (target)
y = df['Attrition']
```

FEATURE SACLING

```
#feature scaling
from sklearn.preprocessing import MinMaxScaler
ms=MinMaxScaler()
x_scaled=pd.DataFrame(ms.fit_transform(x),columns=x.columns)
```

SPLITTING DATA INTO TRAIN AND SET

```
#TRAIN TEST AND SPLIT
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x_scaled,y,test_size =0.2,random_state =42)
```

```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

```
((1176, 46), (294, 46), (1176,), (294,))
```

MODEL BUILDING

o Import the model building Libraries

o Initializing the model

o Training and testing the model

o Evaluation of Model

o Save the Model

1.LOGISTIC REGRESION

2. DECISION TREE
3. RANDOM FOREST

```
#LOGISTIC REGRESSION
from sklearn.linear_model import LogisticRegression
L_model=LogisticRegression()
```

```
L_model.fit(x_train,y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
pred_L=L_model.predict(x_test)
```

EVALUATION OF THE CLASSIFICATION MODEL

```
#Accuracy score
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_auc_score,roc_curve
```

```
accuracy_score(y_test,pred)
```

```
0.8877551020408163
```

```
confusion_matrix(y_test,pred)
```

```
array([[248,   7],
       [ 26,  13]])
```

```
pd.crosstab(y_test,pred)
```

col_0	No	Yes
Attrition		
No	248	7
Yes	26	13

DECISION TREE MODEL

```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier()
```

```
dtc.fit(x_train,y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
D_pred=dtc.predict(x_test)
```

Evaluation of classification model

```
#Accuracy score
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_auc_score,roc_curve
```

```
accuracy_score(y_test,D_pred)
```

```
0.7789115646258503
```

```
confusion_matrix(y_test,D_pred)
```

```
array([[222,  33],
       [ 32,   7]])
```

```
pd.crosstab(y_test,D_pred)
```

col_0	No	Yes
Attrition		
No	222	33
Yes	32	7

```
print(classification_report(y_test,D_pred))
```

```
              precision    recall  f1-score   support

          No       0.87      0.87      0.87       255
         Yes       0.17      0.18      0.18        39

    accuracy                           0.78       294
   macro avg       0.52      0.53      0.52       294
weighted avg       0.78      0.78      0.78       294
```

HYPER PARAMETER TUNING

```
from sklearn import tree
plt.figure(figsize=(25,15))
tree.plot_tree(dtc,filled=True)
```

```
from sklearn.model_selection import GridSearchCV
parameter={
 'criterion':['gini','entropy'],
  'splitter':['best','random'],
  'max_depth':[1,2,3,4,5],
  'max_features':['auto', 'sqrt', 'log2']

}
```

```
grid_search=GridSearchCV(estimator=dtc,param_grid=parameter,cv=5,scoring="accuracy")
```

```
grid_search.fit(x_train,y_train)
```

```
grid_search.best_params_
```

```
{'criterion': 'gini',
 'max_depth': 5,
 'max_features': 'sqrt',
 'splitter': 'best'}
```

```
dtc_cv=DecisionTreeClassifier(criterion= 'gini',
 max_depth=5,
 max_features='sqrt',
 splitter='best')
dtc_cv.fit(x_train,y_train)
```

```
▾              DecisionTreeClassifier
DecisionTreeClassifier(max_depth=5, max_features='sqrt')
```

```
pred=dtc_cv.predict(x_test)
```

```
print(classification_report(y_test,D_pred))
```

```
              precision    recall  f1-score   support

          No       0.87      0.87      0.87       255
         Yes       0.17      0.18      0.18        39

    accuracy                           0.78       294
   macro avg       0.52      0.53      0.52       294
weighted avg       0.78      0.78      0.78       294
```

RANDOM FOREST

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report


# Initialize the Random Forest Classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the classifier on the training data
rf_classifier.fit(x_train, y_train)

# Predict on the test data
R_pred = rf_classifier.predict(x_test)

# Evaluate the model
accuracy = accuracy_score(y_test, R_pred)
report = classification_report(y_test, R_pred)

print(f"Accuracy: {accuracy}")
print("\nClassification Report:")
print(report)
```

```
Accuracy: 0.8775510204081632

Classification Report:
              precision    recall  f1-score   support

          No       0.88      1.00      0.93       255
         Yes       0.80      0.10      0.18        39

    accuracy                           0.88       294
   macro avg       0.84      0.55      0.56       294
```

```
weighted avg       0.87      0.88      0.83       294
```