```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn import model_selection
from sklearn.metrics import accuracy_score
```

```
df=pd.read_csv("Employee attrition data.csv")
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

```
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

```
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.corr()
```

```
<ipython-input-12-2f6f6606aa2c>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ve
  df.corr()
```

	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction
Age	1.000000	0.010661	-0.001686	0.208034	NaN	-0.010145	0.010146
DailyRate	0.010661	1.000000	-0.004985	-0.016806	NaN	-0.050990	0.018355
DistanceFromHome	-0.001686	-0.004985	1.000000	0.021042	NaN	0.032916	-0.016075
Education	0.208034	-0.016806	0.021042	1.000000	NaN	0.042070	-0.027128
EmployeeCount	NaN	NaN	NaN	NaN	NaN	NaN	NaN
EmployeeNumber	-0.010145	-0.050990	0.032916	0.042070	NaN	1.000000	0.017621
EnvironmentSatisfaction	0.010146	0.018355	-0.016075	-0.027128	NaN	0.017621	1.000000
HourlyRate	0.024287	0.023381	0.031131	0.016775	NaN	0.035179	-0.049857
JobInvolvement	0.029820	0.046135	0.008783	0.042438	NaN	-0.006888	-0.008278
JobLevel	0.509604	0.002966	0.005303	0.101589	NaN	-0.018519	0.001212
JobSatisfaction	-0.004892	0.030571	-0.003669	-0.011296	NaN	-0.046247	-0.006784
MonthlyIncome	0.497855	0.007707	-0.017014	0.094961	NaN	-0.014829	-0.006259
MonthlyRate	0.028051	-0.032182	0.027473	-0.026084	NaN	0.012648	0.037600
NumCompaniesWorked	0.299635	0.038153	-0.029251	0.126317	NaN	-0.001251	0.012594
PercentSalaryHike	0.003634	0.022704	0.040235	-0.011111	NaN	-0.012944	-0.031701
PerformanceRating	0.001904	0.000473	0.027110	-0.024539	NaN	-0.020359	-0.029548
RelationshipSatisfaction	0.053535	0.007846	0.006557	-0.009118	NaN	-0.069861	0.007665
StandardHours	NaN	NaN	NaN	NaN	NaN	NaN	NaN
StockOptionLevel	0.037510	0.042143	0.044872	0.018422	NaN	0.062227	0.003432
TotalWorkingYears	0.680381	0.014515	0.004628	0.148280	NaN	-0.014365	-0.002693
TrainingTimesLastYear	-0.019621	0.002453	-0.036942	-0.025100	NaN	0.023603	-0.019359
WorkLifeBalance	-0.021490	-0.037848	-0.026556	0.009819	NaN	0.010309	0.027627
YearsAtCompany	0.311309	-0.034055	0.009508	0.069114	NaN	-0.011240	0.001458
YearsInCurrentRole	0.212901	0.009932	0.018845	0.060236	NaN	-0.008416	0.018007
YearsSinceLastPromotion	0.216513	-0.033229	0.010029	0.054254	NaN	-0.009019	0.016194
YearsWithCurrManager	0.202089	-0.026363	0.014406	0.069065	NaN	-0.009197	-0.004999

26 rows × 26 columns

```
df.corr().MonthlyRate.sort_values(ascending=False)
```

```
<ipython-input-13-d4025e99262a>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ve
  df.corr().MonthlyRate.sort_values(ascending=False)
MonthlyRate                 1.000000
JobLevel                    0.039563
EnvironmentSatisfaction     0.037600
MonthlyIncome               0.034814
Age                         0.028051
DistanceFromHome            0.027473
TotalWorkingYears           0.026442
NumCompaniesWorked          0.017521
```

```
EmployeeNumber              0.012648
WorkLifeBalance             0.007963
YearsSinceLastPromotion     0.001567
TrainingTimesLastYear       0.001467
JobSatisfaction             0.000644
RelationshipSatisfaction   -0.004085
PercentSalaryHike          -0.006429
PerformanceRating          -0.009811
YearsInCurrentRole         -0.012815
HourlyRate                 -0.015297
JobInvolvement             -0.016322
YearsAtCompany             -0.023655
Education                  -0.026084
DailyRate                  -0.032182
StockOptionLevel           -0.034323
YearsWithCurrManager       -0.036746
EmployeeCount                    NaN
StandardHours                    NaN
Name: MonthlyRate, dtype: float64
```

```
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
```

```
df.OverTime.nunique()
```

```
2
```

```
df.OverTime.unique()
```

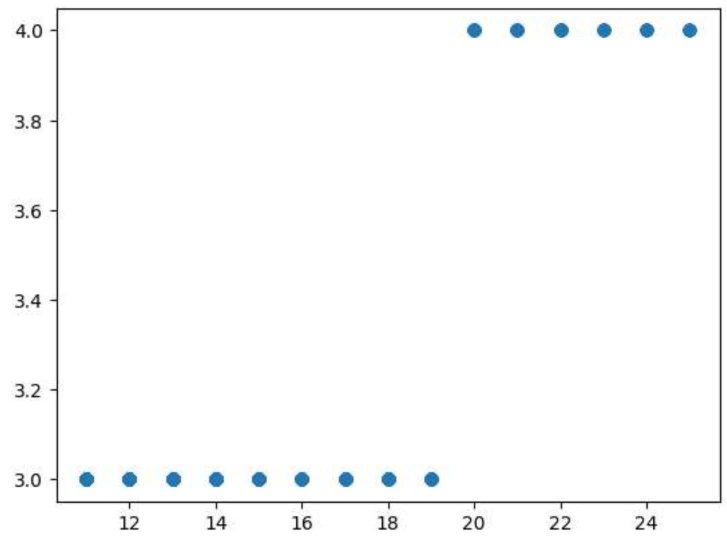
```
array(['Yes', 'No'], dtype=object)
```

```
df.OverTime.value_counts()
```

```
No     1054
Yes     416
Name: OverTime, dtype: int64
```

```
plt.scatter(df["PercentSalaryHike"],df["PerformanceRating"])
```

```
<matplotlib.collections.PathCollection at 0x7ad7238f4460>
```

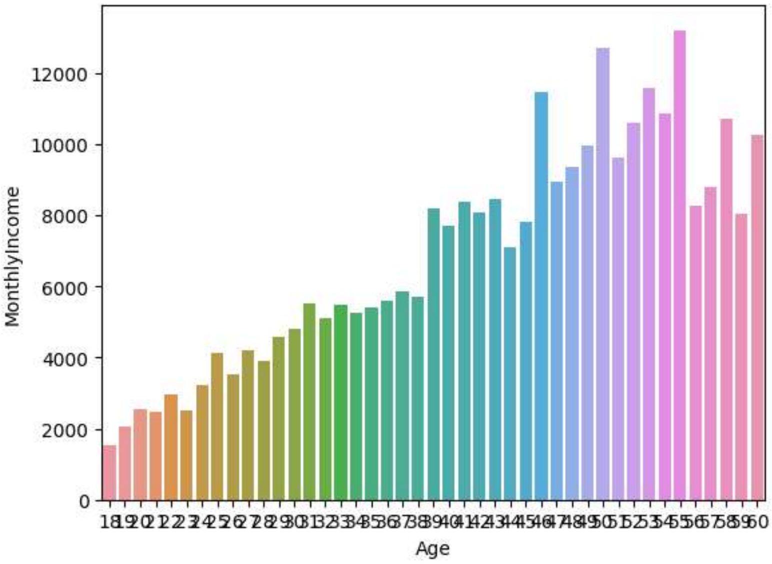



```
sns.heatmap(df.corr(),annot=True)
```

```
<ipython-input-20-8df7bcac526d>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ve
  sns.heatmap(df.corr(),annot=True)
<Axes: >
```

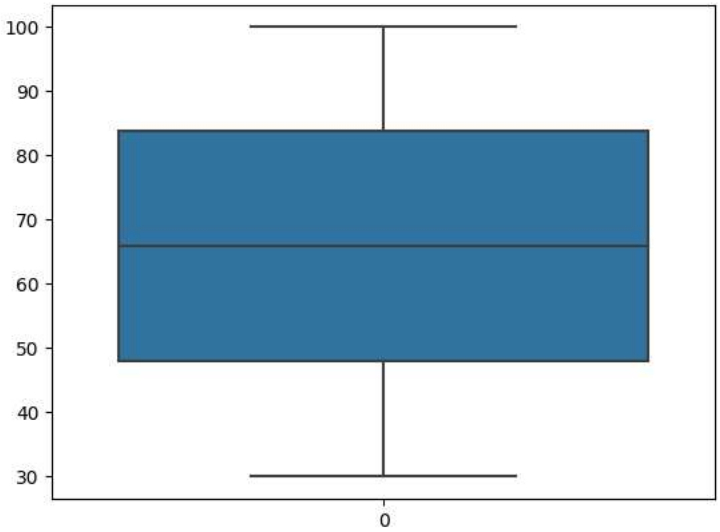

```
sns.barplot(x=df["Age"],y=df["MonthlyIncome"],ci=0)
```

```
<ipython-input-21-3cd0e332144a>:1: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=('ci', 0)` for the same effect.

  sns.barplot(x=df["Age"],y=df["MonthlyIncome"],ci=0)
<Axes: xlabel='Age', ylabel='MonthlyIncome'>
```



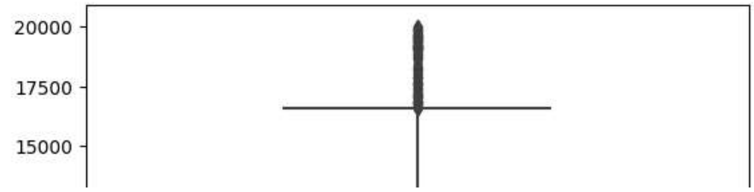

```
sns.boxplot(df["HourlyRate"])
```

```
<Axes: >
```




```
sns.boxplot(df["MonthlyIncome"])
```

```
<Axes: >
```




```
X=df.drop(columns=["PerformanceRating"],axis=1)
X.head()
```

	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	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 × 34 columns

```
X.shape
```

```
(1470, 34)
```

```
type(X)
```

```
pandas.core.frame.DataFrame
```

```
y=df["PerformanceRating"]
y.head()
```

```
0    3
1    4
2    3
3    3
4    3
Name: PerformanceRating, dtype: int64
```

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

```
X["MonthlyIncome"]=le.fit_transform(X["MonthlyIncome"])
```

```
X.head()
```

	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	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 × 34 columns

```
print(le.classes_)
```

```
[ 1009  1051  1052 ... 19943 19973 19999]
```

```
mapping=dict(zip(le.classes_,range(len(le.classes_))))
mapping
```

```
      7484: 943,
      7491: 944,
      7510: 945,
      7525: 946,
      7547: 947,
      7553: 948,
      7587: 949,
      7596: 950,
      7625: 951,
      7632: 952,
      7637: 953,
      7639: 954,
      7642: 955,
      7644: 956,
      7654: 957,
      7655: 958,
      7725: 959,
      7756: 960,
      7779: 961,
      7823: 962,
      7847: 963,
      7861: 964,
      7879: 965,
      7880: 966,
      7898: 967,
      7918: 968,
      7945: 969,
      7969: 970,
      7978: 971,
      7988: 972,
      7991: 973,
      8008: 974,
      8020: 975,
      8095: 976,
      8103: 977,
      8120: 978,
      8161: 979,
      8189: 980,
      8224: 981,
      8237: 982,
      8268: 983,
      8321: 984,
      8346: 985,
      8376: 986,
      8380: 987,
      8381: 988,
      8392: 989,
      8396: 990,
      8412: 991,
      8446: 992,
      8463: 993,
      8474: 994,
      8500: 995,
      8564: 996,
      8578: 997,
      8606: 998,
      8620: 999,
      ...}
```

```
X =df.drop(['Attrition','BusinessTravel','EducationField','OverTime'],axis=1) # Features
y =df['Attrition']
```

```
X['Department'] = preprocessing.LabelEncoder().fit_transform(X['Department'])
X['Education'] = preprocessing.LabelEncoder().fit_transform(X['Education'])
X['JobRole'] = preprocessing.LabelEncoder().fit_transform(X['JobRole'])
X['Gender'] = preprocessing.LabelEncoder().fit_transform(X['Gender'])
X['MaritalStatus'] = preprocessing.LabelEncoder().fit_transform(X['MaritalStatus'])
X['Over18'] = preprocessing.LabelEncoder().fit_transform(X['Over18'])
```

```
Scaler = StandardScaler()
X = Scaler.fit_transform(X)
```

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.2, random_state = 2)
```

```
print(X.shape)
print(X_train.shape)
print(X_test.shape)
```

```
(1470, 31)
(1176, 31)
(294, 31)
```

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
```

```
models = [] # ML Models
models.append(("Random Forest", RandomForestClassifier()))
models.append(("Logistic Regression", LogisticRegression(solver='liblinear')))
models.append(("SVM", svm.SVC(kernel='linear')))

n_folds = 5
results = []
for name, model in models:
    kfold = model_selection.KFold(n_splits=n_folds)
    print("Testing model:", name)

    # Cross Validation Score
    cv_results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold, scoring="f1_weighted", verbose=0, n_jobs=-1)

    # Fitting the Model
    model.fit(X_train,y_train)

    # Model Predictions and Finding Accuracy
    train_pred = model.predict(X_train)
    Training_score = accuracy_score(train_pred,y_train)
    test_pred = model.predict(X_test)
    Test_score = accuracy_score(test_pred,y_test)

    results.append(cv_results)

    msg = f"Cross_Val Mean: {cv_results.mean()}, Training Accuracy: {Training_score}, Testing Accuracy: {Test_score}"
    print(msg + "\n")
```

```
Testing model: Random Forest
Cross_Val Mean: 0.7974876748112971, Training Accuracy: 1.0, Testing Accuracy: 0.8469387755102041

Testing model: Logistic Regression
Cross_Val Mean: 0.8219775205615518, Training Accuracy: 0.8656462585034014, Testing Accuracy: 0.8571428571428571

Testing model: SVM
Cross_Val Mean: 0.7660141990060427, Training Accuracy: 0.8392857142857143, Testing Accuracy: 0.8367346938775511
```

```
from sklearn.metrics import accuracy_score,classification_report
```

```
accuracy_score(test_pred,y_test)
```

```
0.8367346938775511
```

```
print(classification_report(y_test,test_pred))
```

```
              precision    recall  f1-score   support

          No       0.84      1.00      0.91       246
         Yes       0.00      0.00      0.00        48

    accuracy                           0.84       294
   macro avg       0.42      0.50      0.46       294
weighted avg       0.70      0.84      0.76       294

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are
  _warn_prf(average, modifier, msg_start, len(result))
```

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

	col_0
No	246
Yes	48