

ASSIGNMENT - 4

Logistic regression, Decision tree and random forest classifiers on Employee Attrition dataset

NAME: M.SIDDARTHA REG.NO: 21BCE9247

Data preprocessing

```
In [1]: #Importing necessary libraries.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [2]: #Importing the dataset.
df=pd.read_csv("Employee-Attrition.csv")
```

```
In [3]: df.head()
```

```
Out[3]:
```

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

5 rows × 35 columns

```
In [4]: 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

```

In [5]: #Checking for Null Values.
        df.isnull().any()

```

```

Out[5]: 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

In [6]: `df.isnull().sum()`

Out[6]:

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	

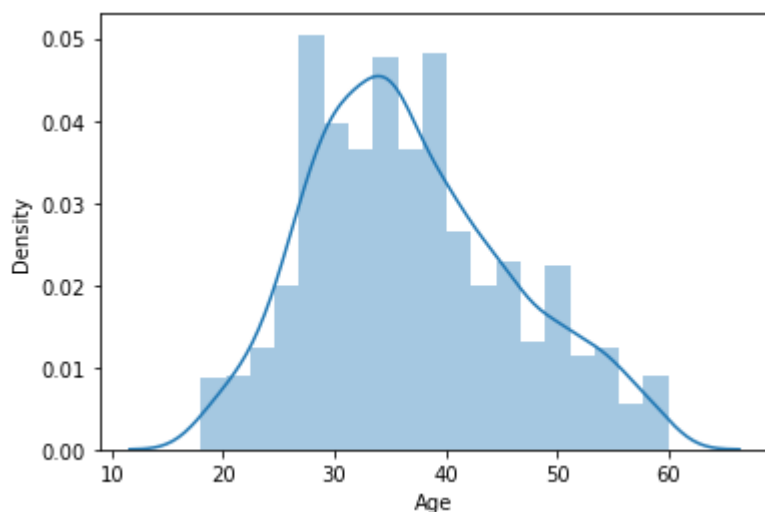
In [7]: `#Data Visualization.`
`sns.distplot(df["Age"])`

C:\Users\SIDDU\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Plea

se adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

```
warnings.warn(msg, FutureWarning)
<AxesSubplot:xlabel='Age', ylabel='Density'>
```

Out[7]:



In [8]:

```
attrition_count = pd.DataFrame(df['Attrition'].value_counts())
plt.pie(attrition_count['Attrition'], labels = ['No', 'Yes'], explode = (0.2,0))
```

Out[8]:

```
((<matplotlib.patches.Wedge at 0x20ef042ac70>,
<matplotlib.patches.Wedge at 0x20ef0439280>],
[Text(-1.136781068348268, 0.6306574368426737, 'No'),
Text(0.961891673217765, -0.5336332157899547, 'Yes')])
```



In [9]:

```
plt.figure(figsize=[20,20])
sns.heatmap(df.corr(),annot=True)
```

Out[9]:

```
<AxesSubplot:>
```

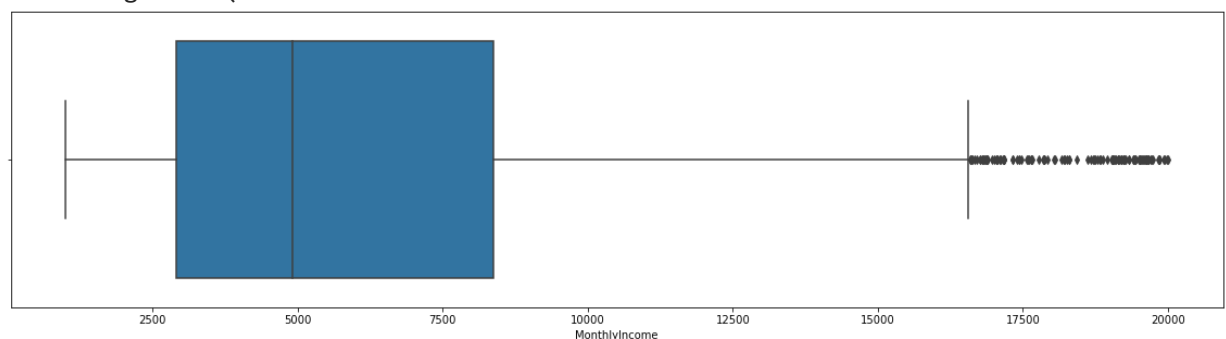


In [10]:

```
#Outlier detection
plt.figure(figsize=[20,5])
sns.boxplot(df['MonthlyIncome'],orient='h')
plt.show()
```

C:\Users\SIDDU\anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

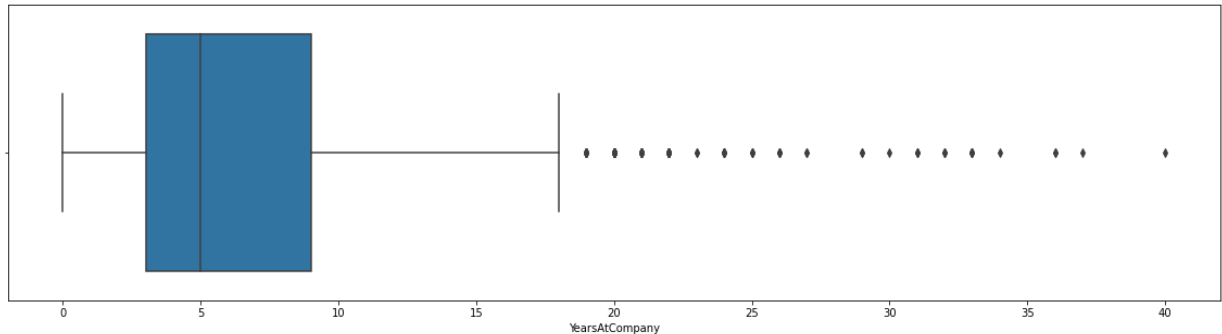
```
warnings.warn(
```



```
In [11]: plt.figure(figsize=[20,5])
sns.boxplot(df['YearsAtCompany'],orient='h')
plt.show()
```

C:\Users\SIDDU\anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```



```
In [12]: # Label Encoding
categories = ['BusinessTravel', 'Department', 'Education', 'EducationField', 'Gender', 'MaritalStatus',
              'EnvironmentSatisfaction', 'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
              'PerformanceRating', 'RelationshipSatisfaction', 'StockOptionLevel', 'TrainingLastMonth']
categorical = df[categories].astype('object')
categorical = pd.get_dummies(df[categories], drop_first = True)
```

```
In [13]: # Splitting Dependent and Independent variables
independent = ['Attrition', 'Over18', 'EmployeeCount', 'StandardHours', 'EmployeeNumber']
continuous = df.drop(columns= categories)
continuous = continuous.drop(columns= independent)
```

```
In [14]: # X - Features, Y- Target variables
X = pd.concat([categorical,continuous],axis=1)
Y = df['Attrition'].replace({'Yes': 1, 'No': 0}).values.reshape(-1,1)
```

```
In [15]: # Feature scaling
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

continuous_variables = list(continuous.columns)

X = X.reset_index()
del X['index']
X[continuous_variables] = pd.DataFrame(scaler.fit_transform(X[continuous_variables]))
```

```
In [16]: #Splitting Data into Train and Test.
         from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.2,random_state=0)
```

```
In [17]: x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

```
Out[17]: ((1176, 44), (294, 44), (1176, 1), (294, 1))
```

LOGISTIC REGRESSION MODEL

```
In [18]: #Importing necessary libraries
         from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import accuracy_score,precision_score, recall_score, f1_score,c
```

```
In [19]: #Initializing the model
         lr = LogisticRegression()
```

```
In [20]: #Training the model
         lr.fit(x_train,y_train)
```

C:\Users\SIDDU\anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().

return f(*args, **kwargs)

C:\Users\SIDDU\anaconda3\lib\site-packages\sklearn\linear_model_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):

STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:

<https://scikit-learn.org/stable/modules/preprocessing.html>

Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

n_iter_i = _check_optimize_result(

```
Out[20]: LogisticRegression()
```

```
In [21]: #Testing the model
         y_pred = lr.predict(x_test)
```

```
In [22]: # Evaluation of model
         # Accuracy score
         print("Accuracy of Logistic regression model:",accuracy_score(y_test,y_pred))
```

Accuracy of Logistic regression model: 0.8843537414965986

In [23]:

```
# Precision score
precision_yes = precision_score(y_test, y_pred, pos_label=1)
print("Precision (Yes): " + str(round(precision_yes, 2)))
precision_no = precision_score(y_test, y_pred, pos_label=0)
print("Precision (No): " + str(round(precision_no, 2)))
```

Precision (Yes): 0.76

Precision (No): 0.9

In [24]:

```
# Recall score
recall_yes = recall_score(y_test, y_pred, pos_label=1)
print("Recall (Yes): " + str(round(recall_yes, 2)))
recall_no = recall_score(y_test, y_pred, pos_label=0)
print("Recall (No): " + str(round(recall_no, 2)))
```

Recall (Yes): 0.45

Recall (No): 0.97

In [25]:

```
# F1 score
f1_score_yes = f1_score(y_test, y_pred, pos_label=1)
print("F1 Score (Yes): " + str(round(f1_score_yes, 2)))
f1_score_no = f1_score(y_test, y_pred, pos_label=0)
print("F1 Score (No): " + str(round(f1_score_no, 2)))
```

F1 Score (Yes): 0.56

F1 Score (No): 0.93

In [26]:

```
# Confusion matrix
print("Confusion matrix:\n\n", confusion_matrix(y_test, y_pred))
```

Confusion matrix:

```
[[238  7]
 [ 27 22]]
```

In [27]:

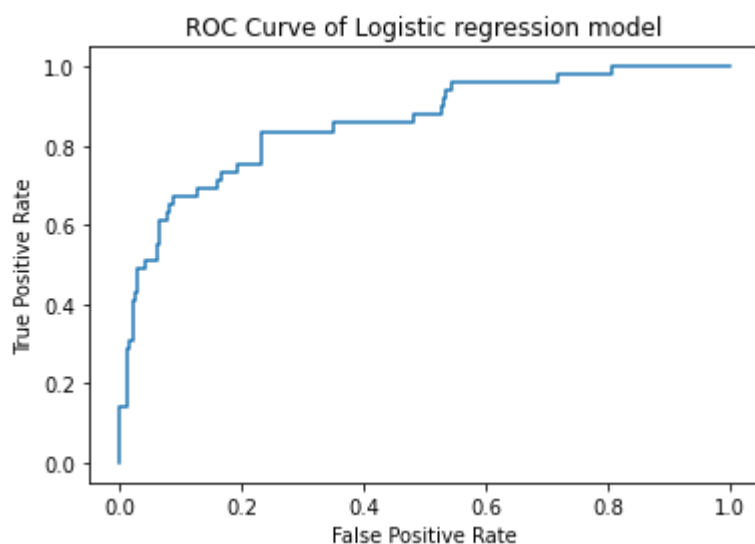
```
# Classification Report
print("Classification report of Logistic Regression model:\n\n", classification_report)
```

Classification report of Logistic Regression model:

	precision	recall	f1-score	support
0	0.90	0.97	0.93	245
1	0.76	0.45	0.56	49
accuracy			0.88	294
macro avg	0.83	0.71	0.75	294
weighted avg	0.87	0.88	0.87	294

In [28]:

```
# ROC curve
probability = lr.predict_proba(x_test)[:,-1]
fpr,tpr,thresholds = roc_curve(y_test,probability)
plt.plot(fpr,tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve of Logistic regression model')
plt.show()
```



DECISION TREE CLASSIFIER

In [29]:

```
# Importing necessary packages
from sklearn.tree import DecisionTreeClassifier
```

In [30]:

```
# Initializing the model
dtc = DecisionTreeClassifier(random_state=30)
```

In [31]:

```
# Training the model
dtc.fit(x_train, y_train)
```

Out[31]: DecisionTreeClassifier(random_state=30)

In [32]:

```
# Testing the model
y_pred1 = dtc.predict(x_test)
```

In [33]:

```
# Evaluation metrics
# Accuracy score
accuracy = accuracy_score(y_test, y_pred1)
print("Accuracy of Decision tree model: ",accuracy)
```

Accuracy of Decision tree model: 0.7517006802721088

In [34]:

```
# Precision score
precision_yes = precision_score(y_test, y_pred1, pos_label=1)
print("Precision (Yes): " , str(round(precision_yes,2)))
precision_no = precision_score(y_test, y_pred1, pos_label=0)
print("Precision (No): " + str(round(precision_no, 2)))
```

Precision (Yes): 0.27
Precision (No): 0.86

In [35]:

```
# Recall score
recall_yes = recall_score(y_test, y_pred1, pos_label=1)
print("Recall (Yes): " + str(round(recall_yes, 2)))
recall_no = recall_score(y_test, y_pred1, pos_label=0)
print("Recall (No): " + str(round(recall_no, 2)))
```

Recall (Yes): 0.29
Recall (No): 0.84

In [36]:

```
# F1 score
f1_score_yes = f1_score(y_test, y_pred1, pos_label=1)
print("F1 Score (Yes): " + str(round(f1_score_yes, 2)))
f1_score_no = f1_score(y_test, y_pred1, pos_label=0)
print("F1 Score (No): " + str(round(f1_score_no, 2)))
```

F1 Score (Yes): 0.28
F1 Score (No): 0.85

In [37]:

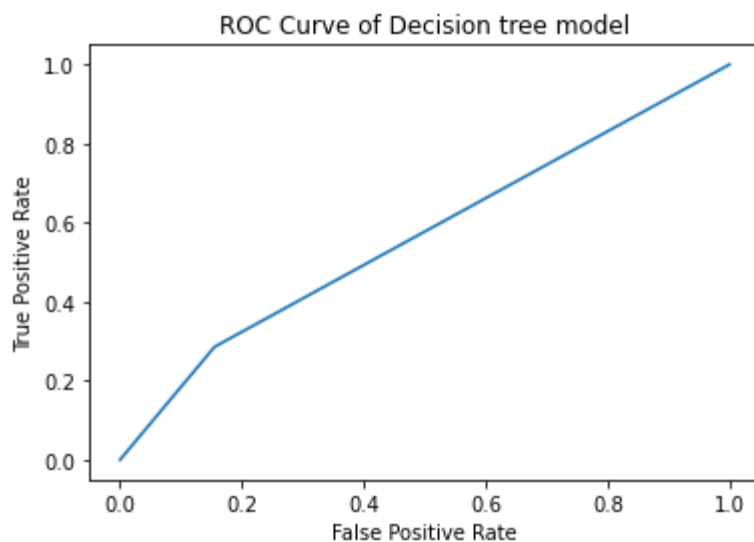
```
# Classification report
print("Classification report of Decision tree model:\n\n",classification_report(y_test,
```

Classification report of Decision tree model:

	precision	recall	f1-score	support
0	0.86	0.84	0.85	245
1	0.27	0.29	0.28	49
accuracy			0.75	294
macro avg	0.56	0.57	0.56	294
weighted avg	0.76	0.75	0.75	294

In [38]:

```
# ROC curve
probability = dtc.predict_proba(x_test)[: ,1]
fpr,tpr,thresholds = roc_curve(y_test,probability)
plt.plot(fpr,tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve of Decision tree model')
plt.show()
```



RANDOM FOREST CLASSIFIER

```
In [39]: # Importing necessary packages
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
```

```
In [40]: # Initializing the model
rf = RandomForestClassifier(n_estimators=10, criterion='entropy', random_state=30)
```

```
In [41]: # Training the model
rf.fit(x_train, y_train)
```

C:\Users\SIDDU\AppData\Local\Temp\ipykernel_20624\391630832.py:2: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().

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

```
Out[41]: RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=30)
```

```
In [42]: rf.score(x_train, y_train)
```

```
Out[42]: 0.983843537414966
```

```
In [43]: # Testing the model
y_pred2 = rf.predict(x_test)
```

```
In [44]: # Evaluation metrics
# Accuracy score
accuracy = accuracy_score(y_test, y_pred2)
print("Accuracy of Random forest model: ", accuracy)
```

Accuracy of Random forest model: 0.8435374149659864

In [45]:

```
# Precision score
precision_yes = precision_score(y_test, y_pred2, pos_label=1)
print("Precision (Yes): " , str(round(precision_yes,2)))
precision_no = precision_score(y_test, y_pred2, pos_label=0)
print("Precision (No): " + str(round(precision_no, 2)))
```

Precision (Yes): 0.71
Precision (No): 0.85

In [46]:

```
# Recall score
recall_yes = recall_score(y_test, y_pred2, pos_label=1)
print("Recall (Yes): " + str(round(recall_yes, 2)))
recall_no = recall_score(y_test, y_pred2, pos_label=0)
print("Recall (No): " + str(round(recall_no, 2)))
```

Recall (Yes): 0.1
Recall (No): 0.99

In [47]:

```
# F1 score
f1_score_yes = f1_score(y_test, y_pred2, pos_label=1)
print("F1 Score (Yes): " + str(round(f1_score_yes, 2)))
f1_score_no = f1_score(y_test, y_pred2, pos_label=0)
print("F1 Score (No): " + str(round(f1_score_no, 2)))
```

F1 Score (Yes): 0.18
F1 Score (No): 0.91

In [48]:

```
# Classification Report
print("Classification report of Random Forest model:\n\n",classification_report(y_test,
```

Classification report of Random Forest model:

	precision	recall	f1-score	support
0	0.85	0.99	0.91	245
1	0.71	0.10	0.18	49
accuracy			0.84	294
macro avg	0.78	0.55	0.55	294
weighted avg	0.82	0.84	0.79	294

In [49]:

```
# ROC curve
probability = rf.predict_proba(x_test)[:,-1]
fpr,tpr,thresholds = roc_curve(y_test,probability)
plt.plot(fpr,tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve of Random forest model')
plt.show()
```

