ASSIGNMENT-4

Name:-B.Sirisha

Reg.no:-21BCE9747

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

1.2 Logistic Regression model

[18]:
```
#Importing necessary libraries
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score,precision_score, recall_score,␣
↪f1_score,confusion_matrix,classification_report,roc_auc_score,roc_curve
```

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

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

```
C:\Users\Admin\anaconda3\lib\site-
packages\sklearn\utils\validation.py:1143: 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().
  y = column_or_1d(y, warn=True)
C:\Users\Admin\anaconda3\lib\sitepackages\sklearn\linear_model\_logis
tic.py:458: 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#logisticregression
  n_iter_i = _check_optimize_result(
```

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

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

[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
```

[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
```

[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
```

[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
```

[26]:
```
# Confusion matrix print("Confusion
matrix:\n\n",confusion_matrix(y_test,y_pred))
```

```
Confusion matrix:

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

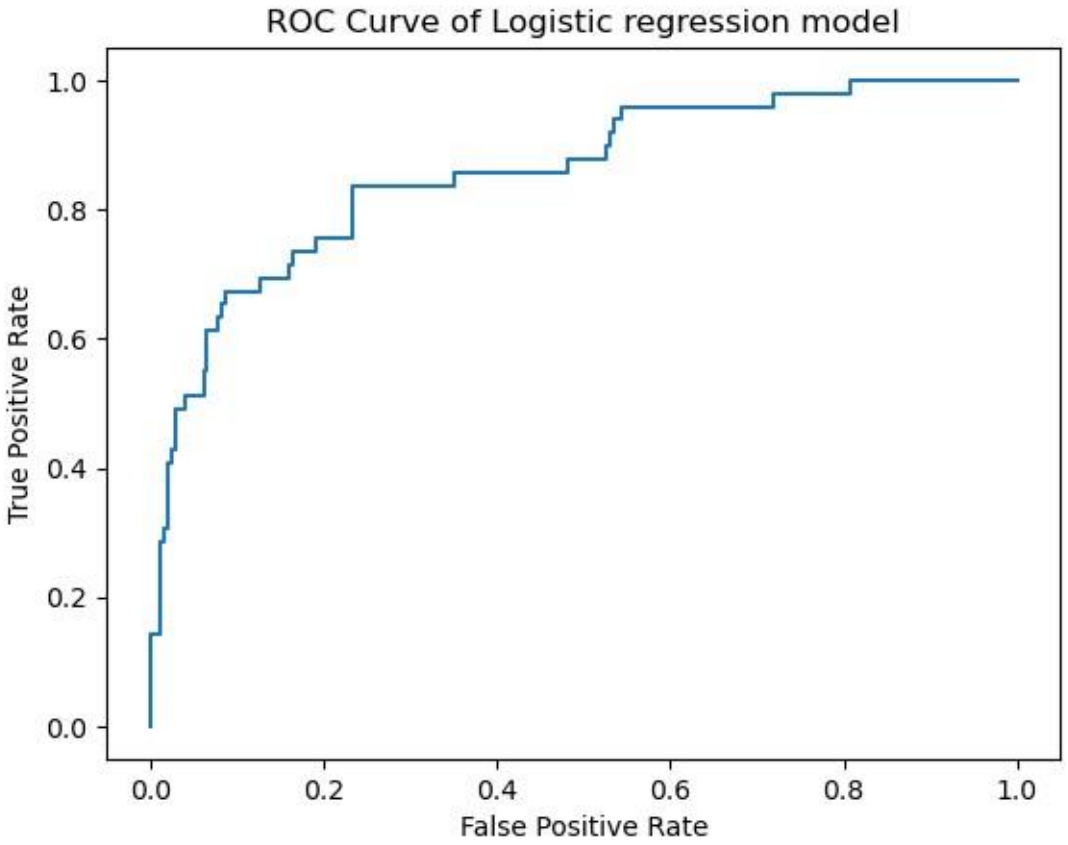
[27]:
```
# Classification Report print("Classification
report of Logistic Regression model:
 ↪\n\n",classification_report(y_test,y_pred))
```

```
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
```

[28]:
```
# ROC curve
probability = lr.predict_proba(x_test)[:,1]
fpr,tpr,threshsholds = 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()
```



1.3 Decision Tree Classifier

[29]:
```
# Importing necesary packages
from sklearn.tree import DecisionTreeClassifier
```

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

[31]:
```
# Training the model
dtc.fit(x_train, y_train)
```

[31]: DecisionTreeClassifier(random_state=30)

[32]:
```
# Testing the model
y_pred1 = dtc.predict(x_test)
```

[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
```

[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
```

[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
```

[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
```

[37]:
```
# Classification report
print("Classification report of Decision tree
model:
↪\n\n",classification_report(y_test,y_pred1))
```

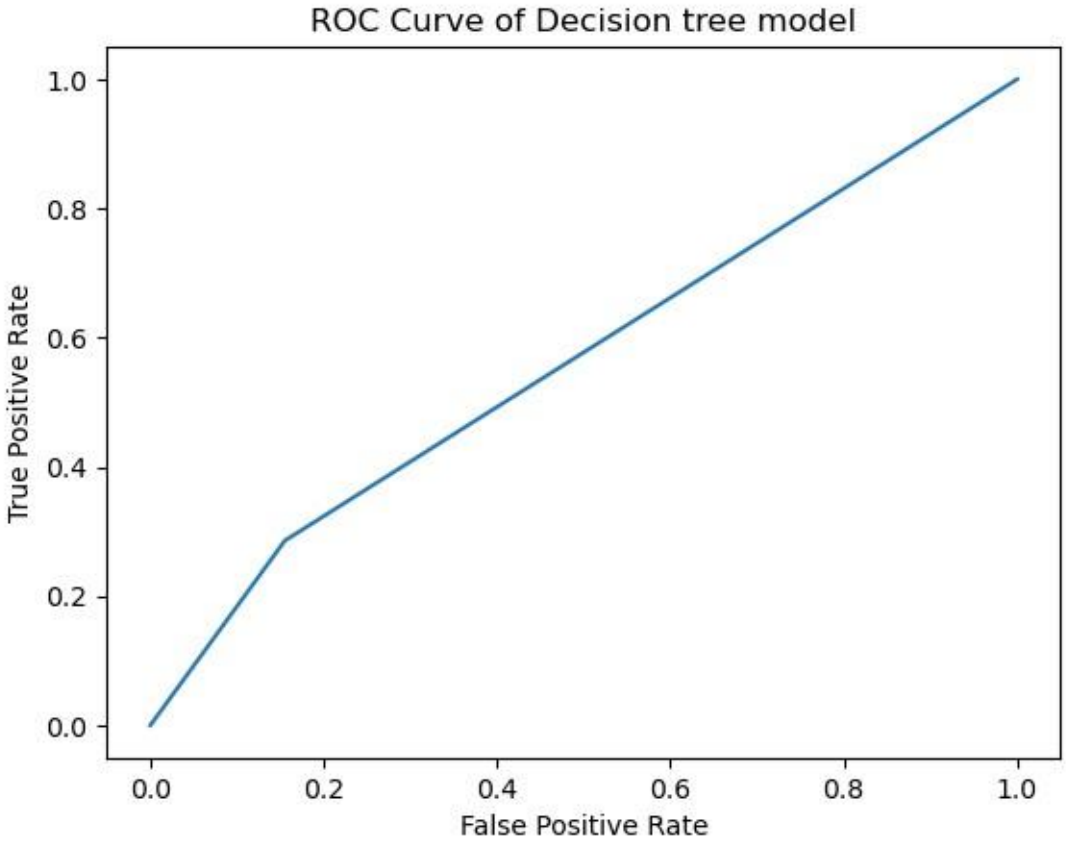
```
Classification report of Decision tree model:
              precision    recall  f1-score   support

           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
```

[38]:
```
# ROC curve probability =
dtc.predict_proba(x_test)[:,1]
fpr,tpr,threshsholds =
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()
```



1.4 Random Forest Classifier

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

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

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

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_39480\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)
```

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

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

[42]:
```
0.983843537414966
```

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

[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
```

[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
```

[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
```

[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
```

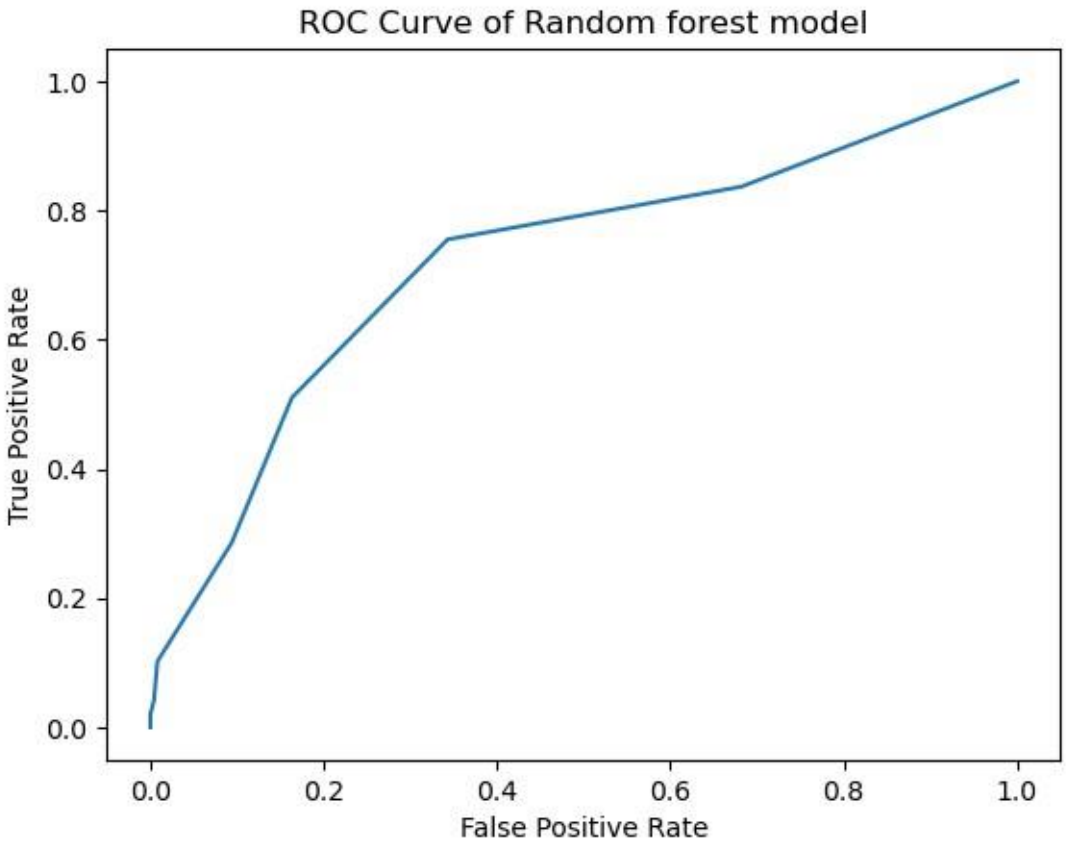
[48]:
```
# Classification Report
print("Classification report of Random Forest
model:
↪\n\n",classification_report(y_test,y_pred2))
```

```
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
```

[49]:
```
# ROC curve
probability = rf.predict_proba(x_test)[:,1]
fpr,tpr,threshsholds = 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()
```


ROC Curve of Random forest model
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0