SWAMY-21bcb7126-assg-4

September 29, 2023

[1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix
import matplotlib.pyplot as plt
```

[2]:
```
data = pd.read_csv("EmployeeAttrition.csv")
```

[3]:
```
data
```

[3]:
```
      Age Attrition     BusinessTravel  DailyRate              Department  \
0      41    Yes   Travel_Rarely    1102  Sales
1      49 No Travel_Frequently 279 Research & Development 2 37 Yes
Travel_Rarely 1373 Research & Development 3 33 No
Travel_Frequently 1392 Research & Development
4      27       No     Travel_Rarely     591 Research & Development
 … …       …                …       …                     …
1465   36    No Travel_Frequently  884 Research & Development
1466   39    No    Travel_Rarely    613 Research & Development
1467   27    No    Travel_Rarely    155 Research & Development
1468   49    No Travel_Frequently  1023  Sales
1469   34    No    Travel_Rarely    628 Research & Development

        DistanceFromHome Education EducationField EmployeeCount \
0                      1   2 Life Sciences  1
1                      8   1 Life Sciences  1
2                      2   2     Other 1
3                      3   4 Life Sciences  1
4                      2   1     Medical    1
      …                …        …            …            …
1465                  23  2     Medical    1
1466                   6   1     Medical    1
1467                   4   3 Life Sciences  1
```

```
1468               2   3     Medical    1

1469               8   3     Medical    1

      EmployeeNumber … RelationshipSatisfaction StandardHours \
0                1 …   1     80

1                2 …   4     80

2                4 …   2     80

3                5 …   3     80

4                7 …   4     80

     …              … …                     …           …

1465          2061 …   3     80

1466          2062 …   1     80

1467          2064 …   2     80

1468          2065 …   4     80

1469          2068 …   1     80

      StockOptionLevel TotalWorkingYears TrainingTimesLastYear \
      0                  0   8     0
      1                  1   10    3
      2                  0   7     3
      3                  0   8     3
      4                  1   6     3
      …                …               …                   …
      1465               1   17    3
      1466               1   9     5
      1467               1   6     0
      1468               0   17    3
      1469               0   6     3

          WorkLifeBalance YearsAtCompany YearsInCurrentRole \
      0                 1     6     4
      1                 3     10    7
      2                 3     0     0
      3                 3     8     7
      4                 3     2     2
      …               …             …                 …
      1465              3     5     2
      1466              3     7     7
      1467              3     6     2
```

```
1468              2     9     6
1469              4     4     3

      YearsSinceLastPromotion YearsWithCurrManager
0                           0   5
1                           1   7
2                           0   0
3                           3   0
4                           2   2
…                         …                   …
1465                        0   3
1466                        1   7
1467                        0   3
1468                        0   8
1469                        1   2

1470                        rows x 35 columns]
```

[4]:
```
data.drop(['EmployeeNumber', 'EmployeeCount', 'StandardHours', 'Over18'], axis=1, inplace=True)
```

[5]:
```
le = LabelEncoder()
categorical_columns = data.select_dtypes(include=['object']).columns
for col in categorical_columns:
    data[col] = le.fit_transform(data[col])
```

[6]:
```
X = data.drop('Attrition', axis=1)
y = data['Attrition']
```

[7]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

[8]:
```
logistic_regression_model = LogisticRegression()
logistic_regression_model.fit(X_train, y_train)
```

```
C:\Users\lenovo\anaconda3\Lib\sitepackages\sklearn\linear_model\_logi
stic.py:460: 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(
```

[8]:
```
LogisticRegression()
```

[9]:
```
logistic_regression_predictions =
logistic_regression_model.predict(X_test)
```

[10]:
```
logistic_regression_accuracy = accuracy_score(y_test,␣
↪logistic_regression_predictions)
logistic_regression_classification_report =
classification_report(y_test,␣ ↪logistic_regression_predictions)
logistic_regression_confusion_matrix = confusion_matrix(y_test,␣
↪logistic_regression_predictions)
```

[11]:
```
decision_tree_model = DecisionTreeClassifier(random_state=42)
decision_tree_model.fit(X_train, y_train)
```

[11]:
```
DecisionTreeClassifier(random_state=42)
```

[12]:
```
decision_tree_predictions = decision_tree_model.predict(X_test)
```

[13]:
```
decision_tree_accuracy = accuracy_score(y_test,
decision_tree_predictions) decision_tree_classification_report =
classification_report(y_test,␣
↪decision_tree_predictions)
decision_tree_confusion_matrix =
confusion_matrix(y_test,␣
↪decision_tree_predictions)
```

[14]:
```
print("Logistic Regression Performance Metrics:")
print(f"Accuracy: {logistic_regression_accuracy:.2f}")
print("Classification Report:\n",
logistic_regression_classification_report) print("Confusion
Matrix:\n", logistic_regression_confusion_matrix)
```

```
Logistic Regression Performance
Metrics: Accuracy: 0.86
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.99      0.92       255
           1       0.00      0.00      0.00        39

    accuracy                           0.86       294
   macro avg       0.43      0.49      0.46       294
```

```
weighted avg     0.75     0.86      0.80       294
Confusion Matrix:
 [[252   3]
 [ 39   0]]
```

[15]:
```
print("\nDecision Tree Performance Metrics:")
print(f"Accuracy: {decision_tree_accuracy:.2f}")
print("Classification Report:\n",
decision_tree_classification_report) print("Confusion
Matrix:\n", decision_tree_confusion_matrix)
```

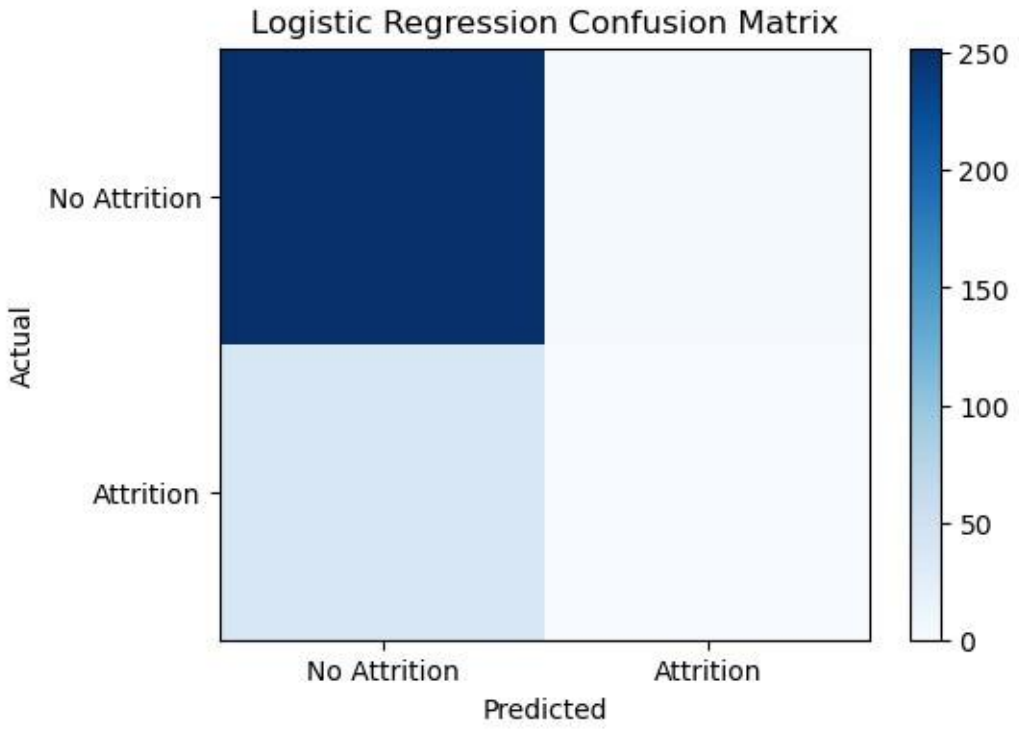
```
Decision Tree Performance Metrics:
Accuracy: 0.80
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.89      0.88       255
           1       0.24      0.23      0.23        39

    accuracy                           0.80       294
   macro avg       0.56      0.56      0.56       294
weighted avg       0.80      0.80      0.80       294
Confusion Matrix:
 [[226 29]
 [ 30   9]]
```

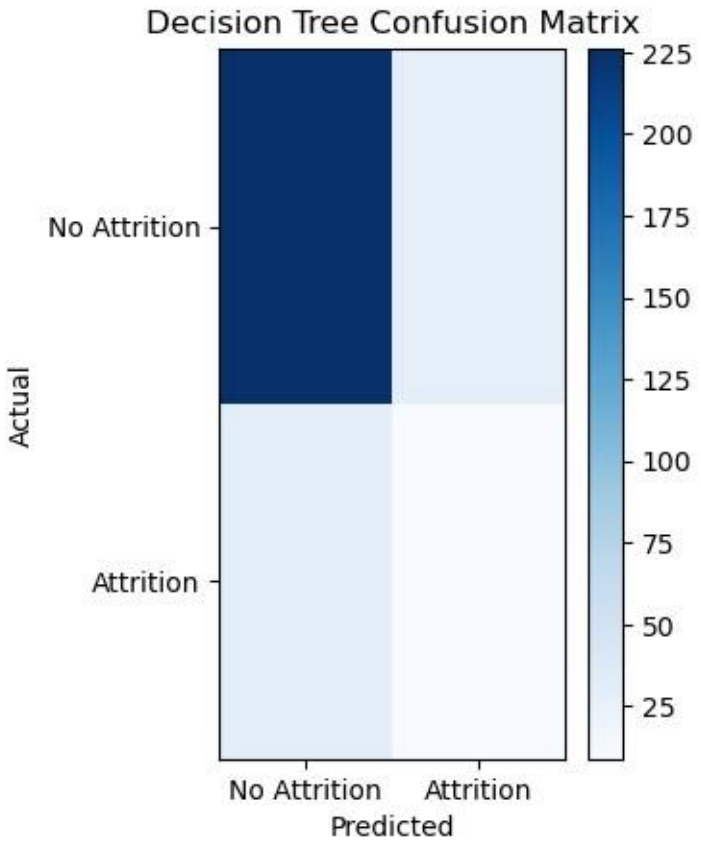
[16]:
```
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.title("Logistic Regression Confusion Matrix")
plt.imshow(logistic_regression_confusion_matrix, cmap='Blues',␣
 ↪interpolation='nearest', aspect='auto')
plt.colorbar() plt.xticks([0, 1], ['No
Attrition', 'Attrition']) plt.yticks([0,
1], ['No Attrition', 'Attrition'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
```

[16]:
```
Text(0, 0.5, 'Actual')
```



[17]:
```
plt.subplot(1, 2, 2)
plt.title("Decision Tree Confusion
Matrix")
       plt.imshow(decision_tree_confusion_matrix,
                                  cmap='Blues',␣
 ↪interpolation='nearest', aspect='auto')
plt.colorbar() plt.xticks([0, 1], ['No
Attrition', 'Attrition']) plt.yticks([0,
1], ['No Attrition', 'Attrition'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
```

[17]:
```
Text(0, 0.5, 'Actual')
```



[18]:
```
plt.tight_layout()
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```

[ ]:
```

```