Modeling_And_Evalution

October 11, 2024

0.0.1 Data Modeling

[3]:
```
#importing the necessary libraries
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report,␣
↪confusion_matrix, roc_auc_score, log_loss
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import LabelBinarizer
from xgboost import XGBClassifier
```

[4]:
```
#loading the preprocessed data
X_train = pd.read_csv('./data/X_train.csv')
X_test = pd.read_csv('./data/X_test.csv')
y_train = pd.read_csv('./data/y_train.csv')
y_test = pd.read_csv('./data/y_test.csv')

y_train = y_train.values.ravel()
y_test = y_test.values.ravel()
```

[5]:
```
#print the data shapes of the training and testing data
print("Data loaded successfully.")
print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_test shape: {y_test.shape}")
```

```
Data loaded successfully.
X_train shape: (299, 22)
X_test shape: (75, 22)
y_train shape: (299,)
y_test shape: (75,)
```

[6]:
```
#define model evaluation function
def evaluate_model(model, X, y, model_name):
    cv_accuracy = cross_val_score(model, X, y, cv=5, scoring='accuracy')
    cv_log_loss = cross_val_score(model, X, y, cv=5, scoring='neg_log_loss')

    print(f"\n{model_name} Cross-Validation Results:")
    print(f"Mean Accuracy: {cv_accuracy.mean():.4f} (+/- {cv_accuracy.std() * 2:
↪.4f})")
    print(f"Mean Log Loss: {-cv_log_loss.mean():.4f} (+/- {cv_log_loss.std() *␣
↪2:.4f})")
```

[7]:
```
#define model evaluation function
def train_predict_evaluate(model, X_train, X_test, y_train, y_test, model_name):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    y_pred_proba = model.predict_proba(X_test)

    accuracy = accuracy_score(y_test, y_pred)

    lb = LabelBinarizer()
    y_test_bin = lb.fit_transform(y_test)
    roc_auc = roc_auc_score(y_test_bin, y_pred_proba, multi_class='ovr',␣
↪average='macro')

    logloss = log_loss(y_test, y_pred_proba)

    print(f"\n{model_name} Test Results:")
    print(f"Accuracy: {accuracy:.4f}")
    print(f"ROC AUC: {roc_auc:.4f}")
    print(f"Log Loss: {logloss:.4f}")
    print("\nClassification Report:")
    print(classification_report(y_test, y_pred))
    print("\nConfusion Matrix:")
    print(confusion_matrix(y_test, y_pred))

    return {
        'model_name': model_name,
        'accuracy': accuracy,
        'roc_auc': roc_auc,
        'log_loss': logloss
    }
```

[22]:
```
#list of models to evaluate
models = [
    (LogisticRegression(multi_class='ovr', random_state=42), "Multinomial␣
↪Logistic Regression"),
    (DecisionTreeClassifier(random_state=42), "Decision Tree"),
```

```
    (RandomForestClassifier(n_estimators=100, random_state=42), "Random␣
↪Forest"),
    (XGBClassifier(random_state=42), "XGBoost"),
    (SVC(kernel='rbf', random_state=42, probability=True), "Support Vector␣
↪Machine"),
    (KNeighborsClassifier(n_neighbors=5), "K-Nearest Neighbors"),

]

results = []

for model, name in models:
    print(f"\n{'='*50}\nEvaluating {name}\n{'='*50}")
    evaluate_model(model, X_train, y_train, name)
    result = train_predict_evaluate(model, X_train, X_test, y_train, y_test,␣
↪name)
    results.append(result)
```

```
==================================================
Evaluating Multinomial Logistic Regression
==================================================

Multinomial Logistic Regression Cross-Validation Results:
Mean Accuracy: 0.9131 (+/- 0.0827)
Mean Log Loss: 0.4210 (+/- 0.3257)

Multinomial Logistic Regression Test Results:
Accuracy: 0.9067
ROC AUC: 0.9221
Log Loss: 0.3854

Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.88      0.82        16
           1       0.95      0.95      0.95        43
           2       0.93      0.81      0.87        16

    accuracy                           0.91        75
   macro avg       0.89      0.88      0.88        75
weighted avg       0.91      0.91      0.91        75


Confusion Matrix:
[[14  1  1]
 [ 2 41  0]
```

```
 [ 2  1 13]]

==================================================
Evaluating Decision Tree
==================================================

Decision Tree Cross-Validation Results:
Mean Accuracy: 0.8763 (+/- 0.1255)
Mean Log Loss: 3.2240 (+/- 2.6318)

Decision Tree Test Results:
Accuracy: 0.8933
ROC AUC: 0.8873
Log Loss: 3.9004

Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.81      0.79        16
           1       0.95      0.98      0.97        43
           2       0.86      0.75      0.80        16

    accuracy                           0.89        75
   macro avg       0.86      0.85      0.85        75
weighted avg       0.89      0.89      0.89        75


Confusion Matrix:
[[13  1  2]
 [ 1 42  0]
 [ 3  1 12]]

==================================================
Evaluating Random Forest
==================================================

Random Forest Cross-Validation Results:
Mean Accuracy: 0.9064 (+/- 0.0831)
Mean Log Loss: 2.0999 (+/- 3.3583)

Random Forest Test Results:
Accuracy: 0.8800
ROC AUC: 0.9208
Log Loss: 2.1108

Classification Report:
              precision    recall  f1-score   support
```

```
           0       0.72      0.81      0.76        16
           1       0.95      0.98      0.97        43
           2       0.85      0.69      0.76        16

    accuracy                           0.88        75
   macro avg       0.84      0.83      0.83        75
weighted avg       0.88      0.88      0.88        75


Confusion Matrix:
[[13  1  2]
 [ 1 42  0]
 [ 4  1 11]]

==================================================
Evaluating XGBoost
==================================================

XGBoost Cross-Validation Results:
Mean Accuracy: 0.9031 (+/- 0.0826)
Mean Log Loss: 0.6215 (+/- 0.5153)

XGBoost Test Results:
Accuracy: 0.9067
ROC AUC: 0.9011
Log Loss: 0.6244

Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.81      0.81        16
           1       0.95      0.98      0.97        43
           2       0.87      0.81      0.84        16

    accuracy                           0.91        75
   macro avg       0.88      0.87      0.87        75
weighted avg       0.91      0.91      0.91        75


Confusion Matrix:
[[13  1  2]
 [ 1 42  0]
 [ 2  1 13]]

==================================================
Evaluating Support Vector Machine
==================================================
```

```
Support Vector Machine Cross-Validation Results:
Mean Accuracy: 0.9030 (+/- 0.0929)
Mean Log Loss: 0.3505 (+/- 0.2622)

Support Vector Machine Test Results:
Accuracy: 0.8800
ROC AUC: 0.9108
Log Loss: 0.3997

Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.81      0.76        16
           1       0.95      0.98      0.97        43
           2       0.85      0.69      0.76        16

    accuracy                           0.88        75
   macro avg       0.84      0.83      0.83        75
weighted avg       0.88      0.88      0.88        75


Confusion Matrix:
[[13  1  2]
 [ 1 42  0]
 [ 4  1 11]]

==================================================
Evaluating K-Nearest Neighbors
==================================================

K-Nearest Neighbors Cross-Validation Results:
Mean Accuracy: 0.8897 (+/- 0.0883)
Mean Log Loss: 2.2124 (+/- 3.2205)

K-Nearest Neighbors Test Results:
Accuracy: 0.8800
ROC AUC: 0.9048
Log Loss: 3.0221

Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.81      0.76        16
           1       0.95      0.98      0.97        43
           2       0.85      0.69      0.76        16

    accuracy                           0.88        75
   macro avg       0.84      0.83      0.83        75
```

```
weighted avg       0.88      0.88      0.88        75


Confusion Matrix:
[[13  1  2]
 [ 1 42  0]
 [ 4  1 11]]
```

[23]:
```python
#Compare the results of the models
comparison_df = pd.DataFrame(results)
comparison_df = comparison_df.set_index('model_name')
print("\nModel Comparison:")
display(comparison_df)

best_accuracy = comparison_df['accuracy'].idxmax()
best_roc_auc = comparison_df['roc_auc'].idxmax()
best_log_loss = comparison_df['log_loss'].idxmin()

print(f"\nBest model by accuracy: {best_accuracy}")
print(f"Best model by ROC AUC: {best_roc_auc}")
print(f"Best model by log loss: {best_log_loss}")
```

```
Model Comparison:

                                 accuracy   roc_auc  log_loss
model_name
Multinomial Logistic Regression  0.906667  0.922144  0.385404
Decision Tree                    0.893333  0.887287  3.900374
Random Forest                    0.880000  0.920789  2.110767
XGBoost                          0.906667  0.901072  0.624433
Support Vector Machine           0.880000  0.910795  0.399662
K-Nearest Neighbors              0.880000  0.904770  3.022085


Best model by accuracy: Multinomial Logistic Regression
Best model by ROC AUC: Multinomial Logistic Regression
Best model by log loss: Multinomial Logistic Regression
```