Model Development Phase Template

Date	15 April 2024
Team ID	Team-738164
Project Title	Rainfall Prediction Using Machine Learning
Maximum Marks	4 Marks

Initial Model Training Code, Model Validation and Evaluation Report

Initial Model Training Code:

Logistic Regression

Baseline

In [ ]:
```
logreg = LogisticRegression(solver='liblinear', random_state=42)
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
y_pred
```

Out[76]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [ ]:
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

def conf_matrix(model, X_test, y_test, cmap='Blues'):

    # Calculate confusion matrix
    y_pred = model.predict(X_test)
    cm = confusion_matrix(y_test, y_pred)

    # Plot confusion matrix using ConfusionMatrixDisplay
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=model.classes_)
    disp.plot(cmap='Blues')
    plt.grid()
    plt.show()

def roc_curve_custom(model, X_test, y_test):

    # Generate predicted probabilities
    y_proba = model.predict_proba(X_test)

    # Plot ROC curve using RocCurveDisplay
    disp = RocCurveDisplay.from_estimator(model, X_test, y_test)
    plt.plot([0, 1], [0, 1], color='black', linestyle='--')
    plt.show()
```

```
def evaluate(model, X_train=X_train, X_test=X_test, y_train=y_train, y_test=y_test, y_pred=y_pred):
    # Confusion Matrix
    print('Confusion Matrix')
    print('-'*53)
    conf_matrix(model, X_test, y_test)
    print('\n')

    # Classification Report
    print('Classification Report')
    print('-'*53)
    print(classification_report(y_test, y_pred))
    print('\n')

    # ROC Curve
    print('ROC Curve')
    print('-'*53)
    roc_curve_custom(model, X_test, y_test)
    print('\n')

    # Checking model fitness
    print('Checking model fitness')
    print('-'*53)
    print('Train score:', round(model.score(X_train, y_train), 4))
    print('Test score: ', round(model.score(X_test, y_test), 4))
    print('\n')

evaluate(logreg)
```

Random Forest

Baseline

In [ ]:
```
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
y_pred_rf
```

Out[102]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [ ]:
```
evaluate(rf, y_pred=y_pred_rf)
```

Decision Tree

Baseline

In [ ]:
```
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)
y_pred_tree = clf.predict(X_test)
y_pred_tree
```

Out[93]:
```
array([0, 0, 0, ..., 0, 1, 0])
```

In [ ]:
```
evaluate(clf, y_pred=y_pred_tree)
```

XGBoost

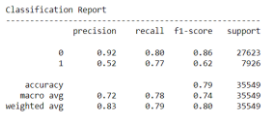
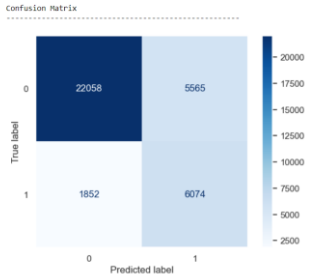
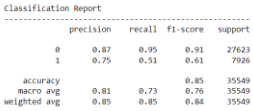
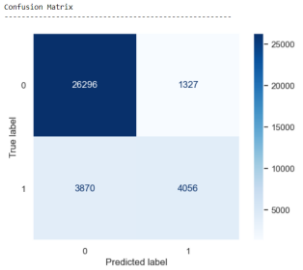
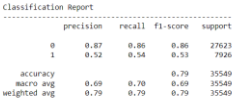
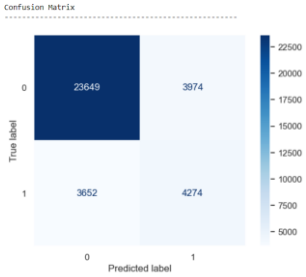
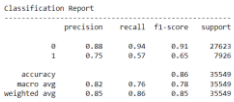
Baseline

In [ ]:
```
xgb = XGBClassifier(random_state=42)
xgb.fit(X_train, y_train)
y_pred_xgb = xgb.predict(X_test)
y_pred_xgb
```

Out[111]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [ ]:
```
evaluate(xgb, y_pred=y_pred_xgb)
```

Model Validation and Evaluation Report:

Model	Classification Report	Accuracy	Confusion Matrix
Logistic Regression (Baseline)		79%	
Random Forest (Baseline)		85%	
Decision Tree (Baseline)		79%	
XGBoost (Baseline)		86%