

Initial Model Training Code, Model Validation and Evaluation Report

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)	<pre>Classification Report ----- precision recall f1-score support 0 0.87 0.94 0.90 27623 1 0.70 0.58 0.59 7926 accuracy 0.79 0.72 0.74 35549 macro avg 0.79 0.72 0.74 35549 weighted avg 0.83 0.84 0.83 35549</pre>	84%	 <p>Confusion Matrix for Logistic Regression (Baseline):</p> <table><tr><th>True \ Predicted</th><th>0</th><th>1</th></tr><tr><th>0</th><td>25931</td><td>1692</td></tr><tr><th>1</th><td>3925</td><td>4001</td></tr></table>	True \ Predicted	0	1	0	25931	1692	1	3925	4001
True \ Predicted	0	1										
0	25931	1692										
1	3925	4001										
Random Forest (Baseline)	<pre>Classification Report ----- precision recall f1-score support 0 0.87 0.95 0.91 27623 1 0.75 0.51 0.61 7926 accuracy 0.85 0.73 0.76 35549 macro avg 0.81 0.73 0.76 35549 weighted avg 0.85 0.85 0.84 35549</pre>	85%	 <p>Confusion Matrix for Random Forest (Baseline):</p> <table><tr><th>True \ Predicted</th><th>0</th><th>1</th></tr><tr><th>0</th><td>26296</td><td>1327</td></tr><tr><th>1</th><td>3870</td><td>4056</td></tr></table>	True \ Predicted	0	1	0	26296	1327	1	3870	4056
True \ Predicted	0	1										
0	26296	1327										
1	3870	4056										
Decision Tree (Baseline)	<pre>Classification Report ----- precision recall f1-score support 0 0.87 0.86 0.86 27623 1 0.52 0.54 0.53 7926 accuracy 0.69 0.76 0.69 35549 macro avg 0.69 0.76 0.69 35549 weighted avg 0.79 0.79 0.79 35549</pre>	79%	 <p>Confusion Matrix for Decision Tree (Baseline):</p> <table><tr><th>True \ Predicted</th><th>0</th><th>1</th></tr><tr><th>0</th><td>23649</td><td>3974</td></tr><tr><th>1</th><td>3852</td><td>4274</td></tr></table>	True \ Predicted	0	1	0	23649	3974	1	3852	4274
True \ Predicted	0	1										
0	23649	3974										
1	3852	4274										
XG-Boost (Baseline)	<pre>Classification Report ----- precision recall f1-score support 0 0.88 0.94 0.91 27623 1 0.75 0.57 0.65 7926 accuracy 0.82 0.76 0.78 35549 macro avg 0.82 0.76 0.78 35549 weighted avg 0.85 0.86 0.85 35549</pre>	86%	 <p>Confusion Matrix for XG-Boost (Baseline):</p> <table><tr><th>True \ Predicted</th><th>0</th><th>1</th></tr><tr><th>0</th><td>26090</td><td>1533</td></tr><tr><th>1</th><td>3419</td><td>4507</td></tr></table>	True \ Predicted	0	1	0	26090	1533	1	3419	4507
True \ Predicted	0	1										
0	26090	1533										
1	3419	4507										