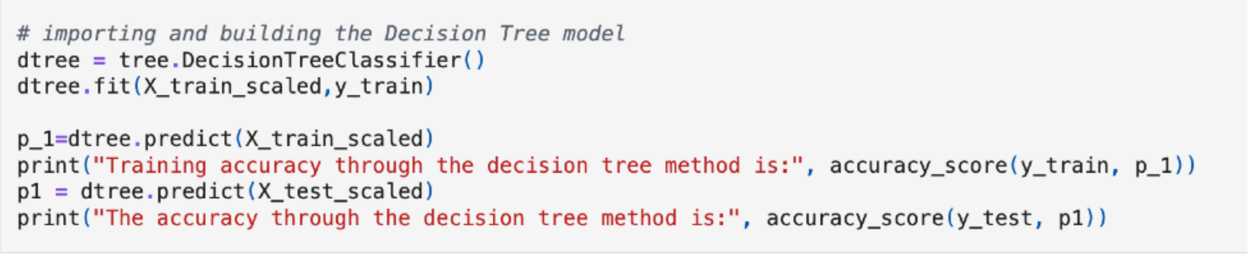
Model Development Phase Template

Date	14 June 2025
Team ID	SWTID1749618778
Project Title	Rising Waters: A Machine Learning Approach To Flood Prediction
Maximum Marks	4 Marks

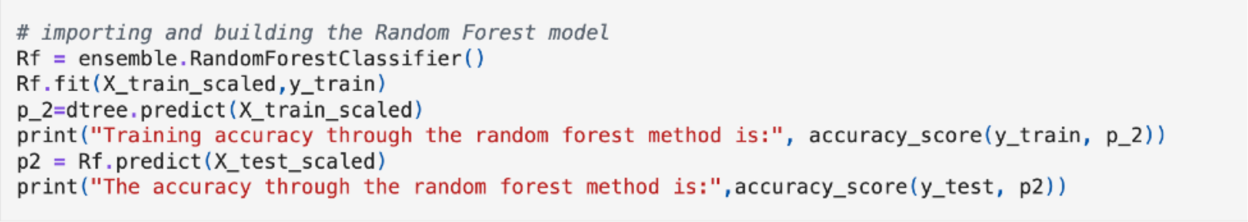
Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

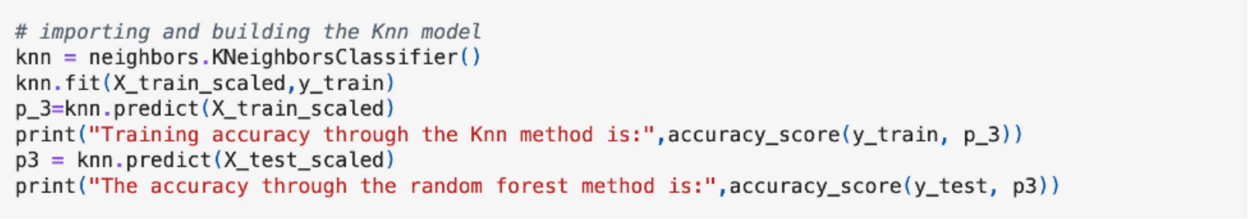
Initial Model Training Code:



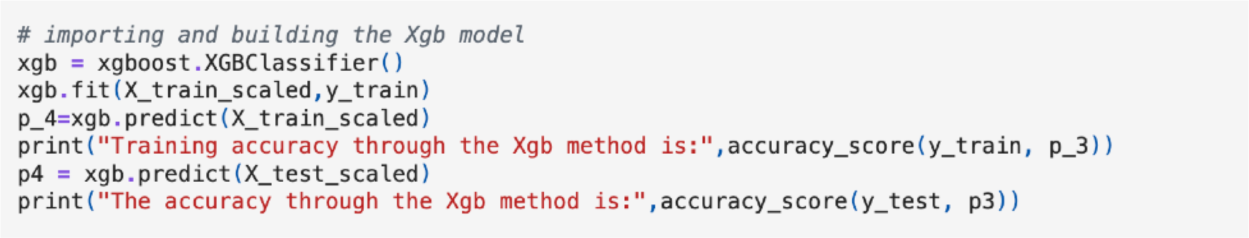
```
# importing and building the Decision Tree model
dtree = tree.DecisionTreeClassifier()
dtree.fit(X_train_scaled,y_train)

p_1=dtree.predict(X_train_scaled)
print("Training accuracy through the decision tree method is:", accuracy_score(y_train, p_1))
p1 = dtree.predict(X_test_scaled)
print("The accuracy through the decision tree method is:", accuracy_score(y_test, p1))
```



```
# importing and building the Random Forest model
Rf = ensemble.RandomForestClassifier()
Rf.fit(X_train_scaled,y_train)
p_2=dtree.predict(X_train_scaled)
print("Training accuracy through the random forest method is:", accuracy_score(y_train, p_2))
p2 = Rf.predict(X_test_scaled)
print("The accuracy through the random forest method is:",accuracy_score(y_test, p2))
```



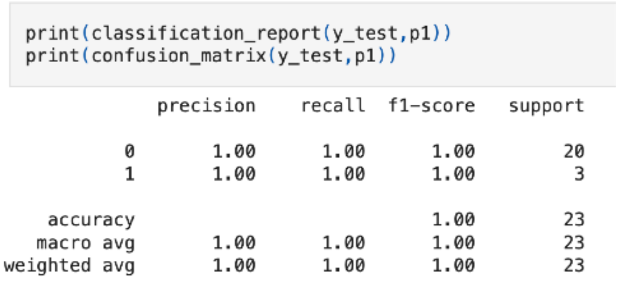
```
# importing and building the Knn model
knn = neighbors.KNeighborsClassifier()
knn.fit(X_train_scaled,y_train)
p_3=knn.predict(X_train_scaled)
print("Training accuracy through the Knn method is:",accuracy_score(y_train, p_3))
p3 = knn.predict(X_test_scaled)
print("The accuracy through the random forest method is:",accuracy_score(y_test, p3))
```



```
# importing and building the Xgb model
xgb = xgboost.XGBClassifier()
xgb.fit(X_train_scaled,y_train)
p_4=xgb.predict(X_train_scaled)
print("Training accuracy through the Xgb method is:",accuracy_score(y_train, p_3))
p4 = xgb.predict(X_test_scaled)
print("The accuracy through the Xgb method is:",accuracy_score(y_test, p3))
```

Model Validation and Evaluation Report:

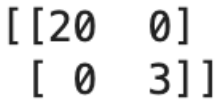
<table>
<tr><th>Model</th><th>Classification Report</th><th>Accuracy</th><th>Confusion Matrix</th></tr>
<tr><td>Decision Tree</td><td>



```
print(classification_report(y_test,p1))
print(confusion_matrix(y_test,p1))
```

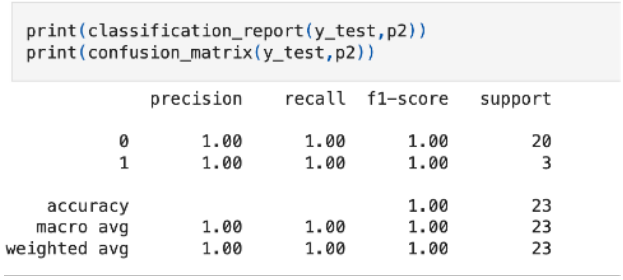
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        20
           1       1.00      1.00      1.00         3

    accuracy                           1.00        23
   macro avg       1.00      1.00      1.00        23
weighted avg       1.00      1.00      1.00        23
```

</td><td>100%</td><td>



```
[[20  0]
 [ 0  3]]
```

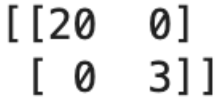
</td></tr>
<tr><td>Random Forest</td><td>



```
print(classification_report(y_test,p2))
print(confusion_matrix(y_test,p2))
```

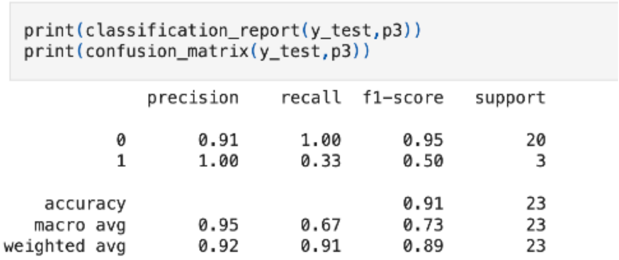
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        20
           1       1.00      1.00      1.00         3

    accuracy                           1.00        23
   macro avg       1.00      1.00      1.00        23
weighted avg       1.00      1.00      1.00        23
```

</td><td>100%</td><td>



```
[[20  0]
 [ 0  3]]
```

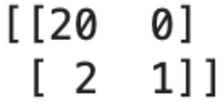
</td></tr>
<tr><td>KNN</td><td>



```
print(classification_report(y_test,p3))
print(confusion_matrix(y_test,p3))
```

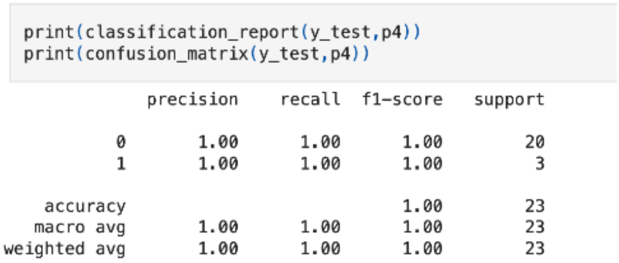
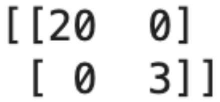
```
              precision    recall  f1-score   support

           0       0.91      1.00      0.95        20
           1       1.00      0.33      0.50         3

    accuracy                           0.91        23
   macro avg       0.95      0.67      0.73        23
weighted avg       0.92      0.91      0.89        23
```

</td><td>91.3%</td><td>



```
[[20  0]
 [ 2  1]]
```

</td></tr>
</table>

XGB	
	print(classification_report(y_test,p4))
	print(confusion_matrix(y_test,p4))
	
	              precision recall f1-score support
	
	           0 1.00 1.00 1.00 20
	           1 1.00 1.00 1.00 3
	
	    accuracy 1.00 23
	   macro avg 1.00 1.00 1.00 23
	weighted avg 1.00 1.00 1.00 23
		91.3%	
			[[20 0]
			 [0 3]]