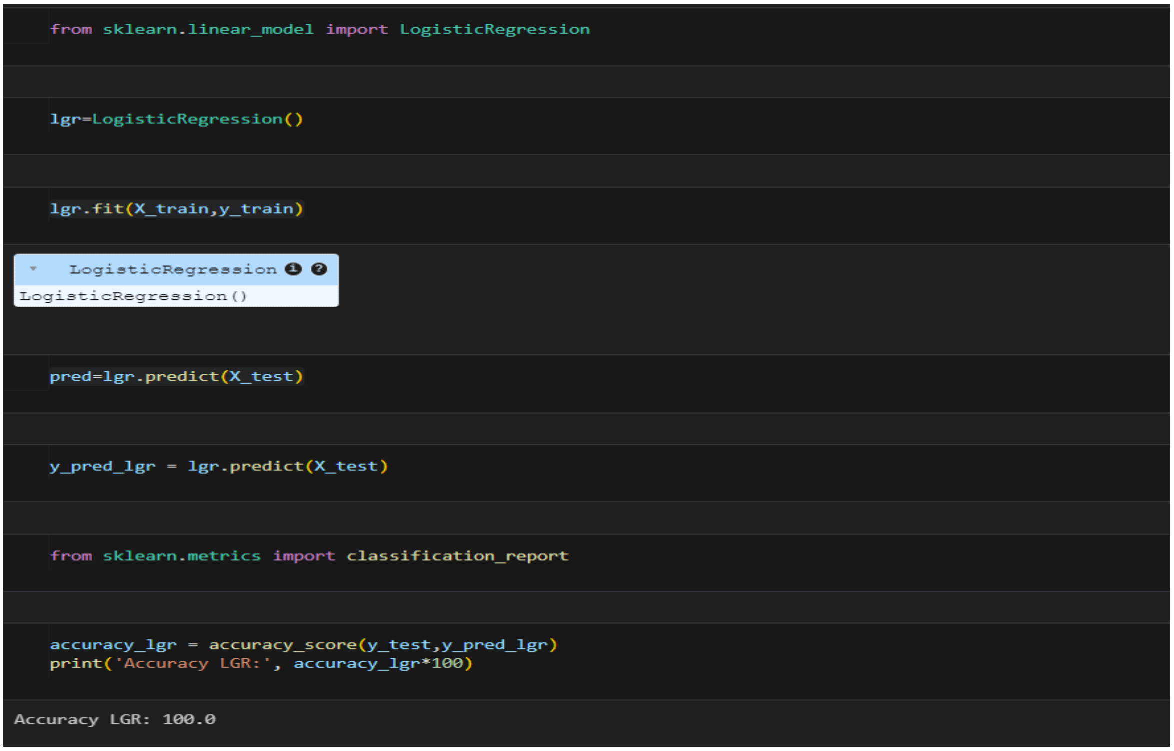
Model Development Phase Template

Date	15 July 2024
Team ID	xxxxxx
Project Title	Detection of Autistic Spectrum Disorder: Classification
Maximum Marks	4 Marks

Initial Model Training Code:

1) Logistic Regression:



```
from sklearn.linear_model import LogisticRegression
```

```
lgr=LogisticRegression()
```

```
lgr.fit(X_train,y_train)
```

```
LogisticRegression
LogisticRegression()
```

```
pred=lgr.predict(X_test)
```

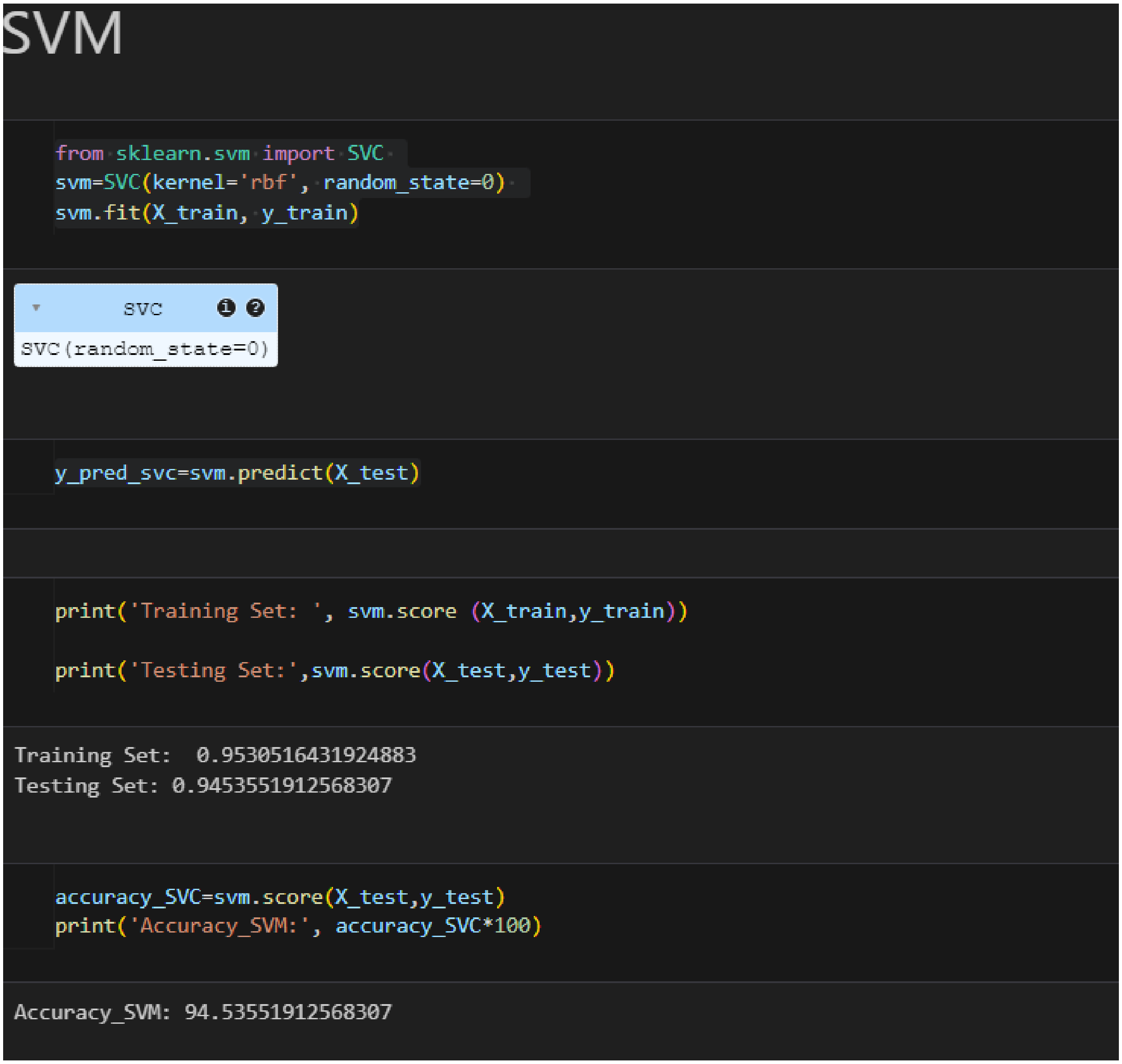
```
y_pred_lgr = lgr.predict(X_test)
```

```
from sklearn.metrics import classification_report
```

```
accuracy_lgr = accuracy_score(y_test,y_pred_lgr)
print('Accuracy LGR:', accuracy_lgr*100)
```

```
Accuracy LGR: 100.0
```

2) SVM:



SVM

```
from sklearn.svm import SVC
svm=SVC(kernel='rbf', random_state=0)
svm.fit(X_train, y_train)
```

```
SVC(random_state=0)
```

```
y_pred_svc=svm.predict(X_test)
```

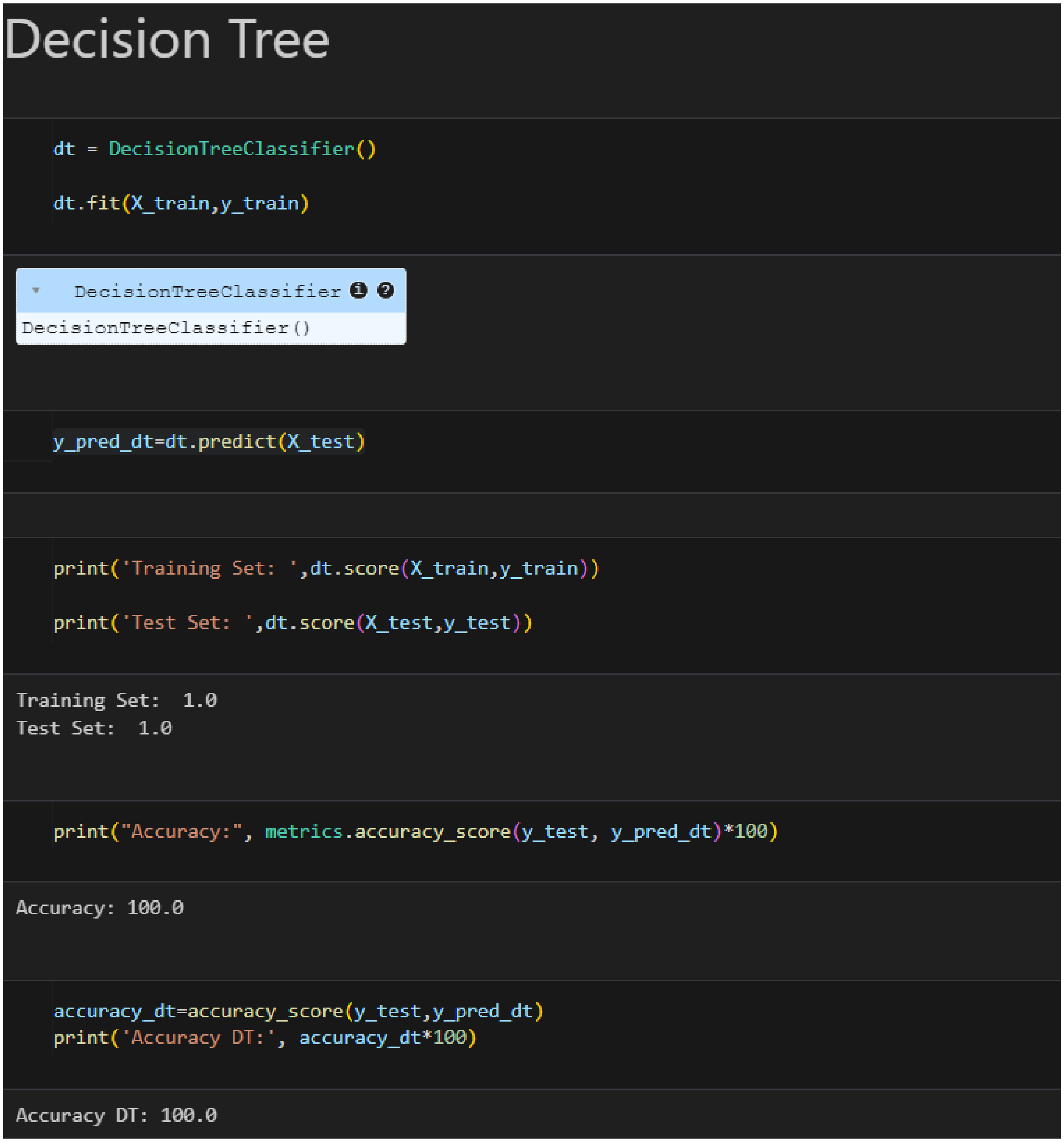
```
print('Training Set: ', svm.score (X_train,y_train))

print('Testing Set:',svm.score(X_test,y_test))
```

```
Training Set:  0.9530516431924883
Testing Set: 0.9453551912568307
```

```
accuracy_SVC=svm.score(X_test,y_test)
print('Accuracy_SVM:', accuracy_SVC*100)
```

```
Accuracy_SVM: 94.53551912568307
```

3) Decision Tree:



Decision Tree

```
dt = DecisionTreeClassifier()

dt.fit(X_train,y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

```
y_pred_dt=dt.predict(X_test)
```

```
print('Training Set: ',dt.score(X_train,y_train))

print('Test Set: ',dt.score(X_test,y_test))
```

```
Training Set:  1.0
Test Set:  1.0
```

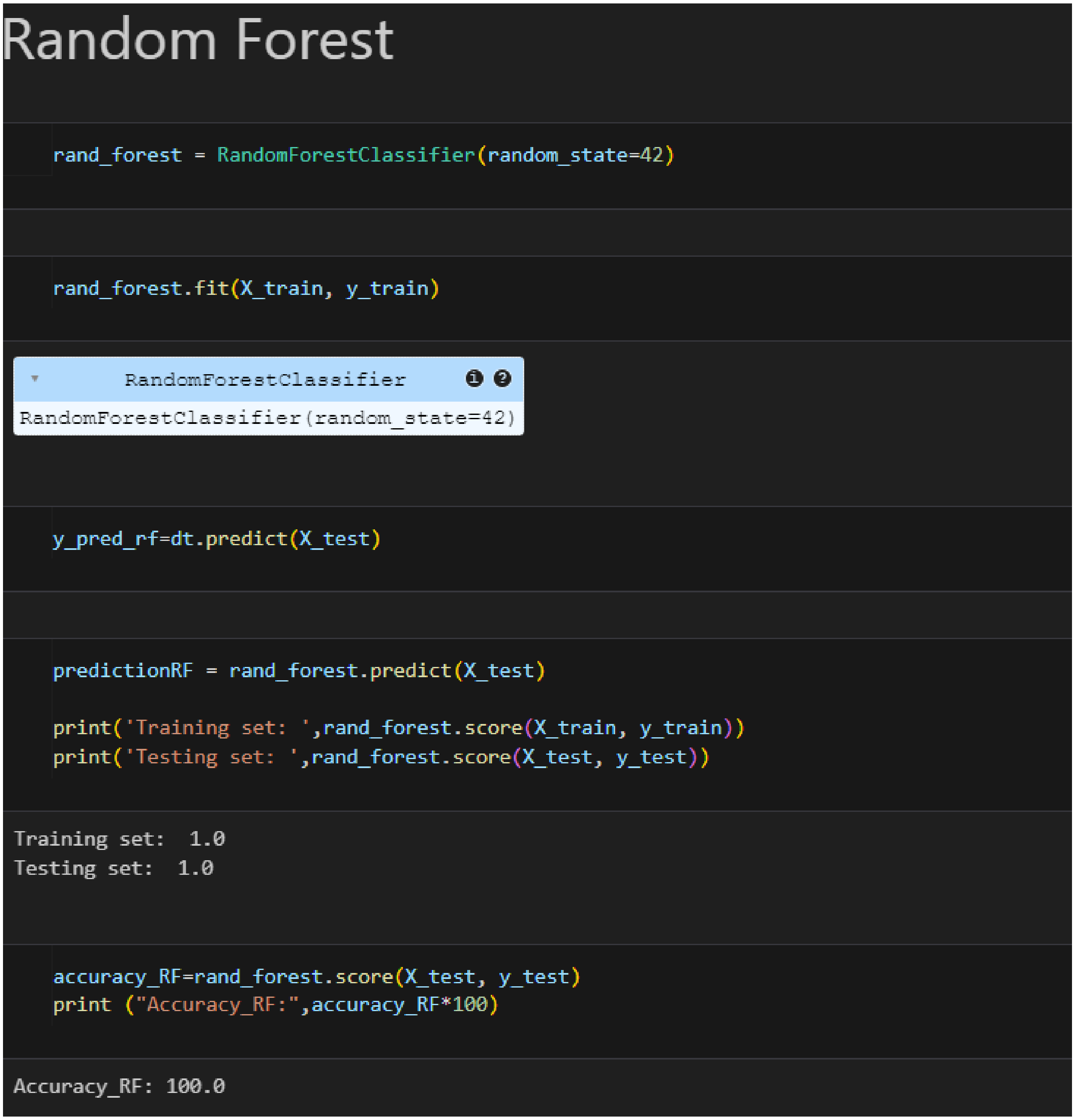
```
print("Accuracy:", metrics.accuracy_score(y_test, y_pred_dt)*100)
```

```
Accuracy: 100.0
```

```
accuracy_dt=accuracy_score(y_test,y_pred_dt)
print('Accuracy DT:', accuracy_dt*100)
```

```
Accuracy DT: 100.0
```

4) Random Forest:



Random Forest

```
rand_forest = RandomForestClassifier(random_state=42)
```

```
rand_forest.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
y_pred_rf=dt.predict(X_test)
```

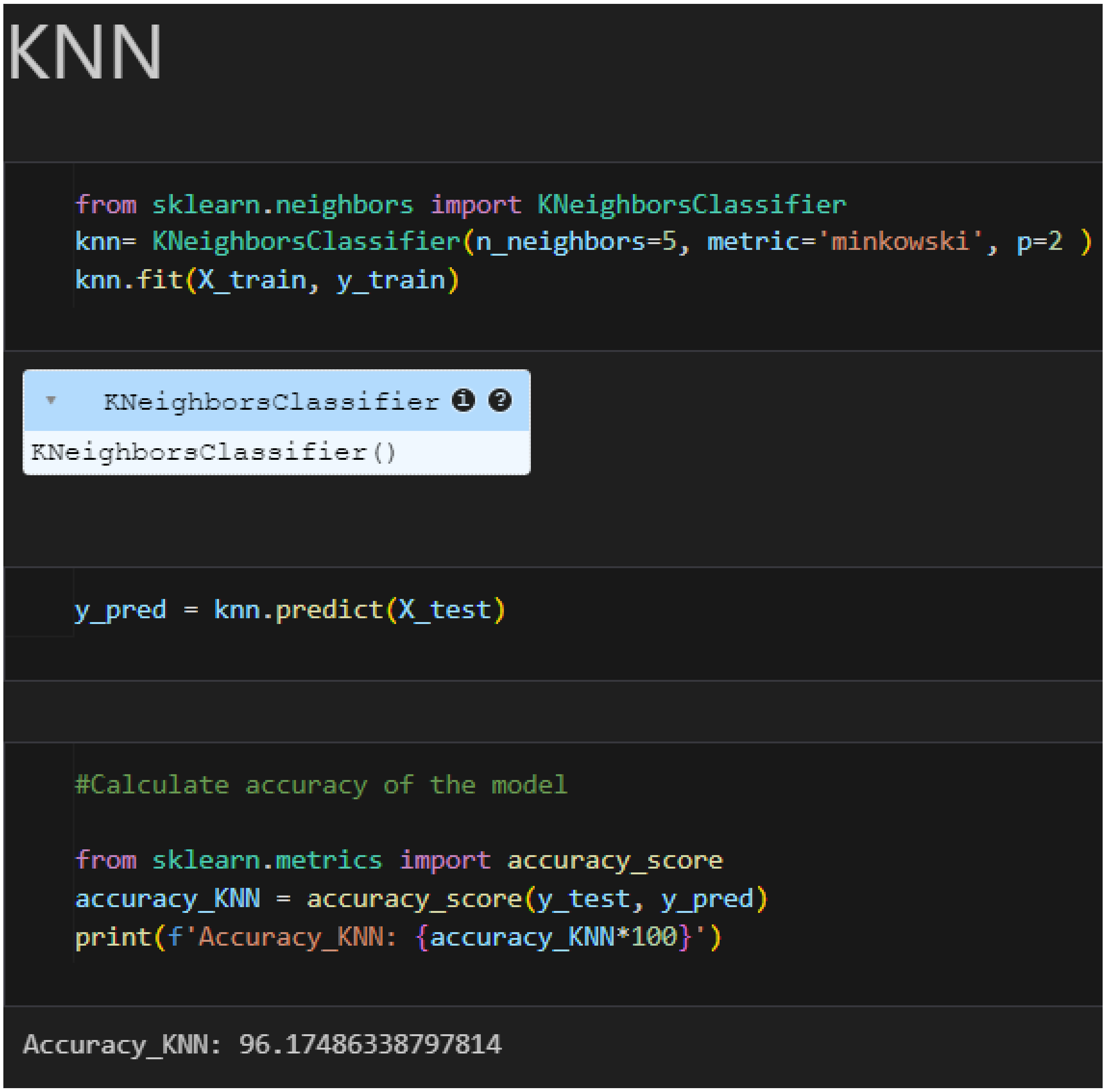
```
predictionRF = rand_forest.predict(X_test)

print('Training set: ',rand_forest.score(X_train, y_train))
print('Testing set: ',rand_forest.score(X_test, y_test))
```

```
Training set:  1.0
Testing set:  1.0
```

```
accuracy_RF=rand_forest.score(X_test, y_test)
print ("Accuracy_RF:",accuracy_RF*100)
```

```
Accuracy_RF: 100.0
```

5) KNN:



KNN

```
from sklearn.neighbors import KNeighborsClassifier
knn= KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2 )
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier()
```

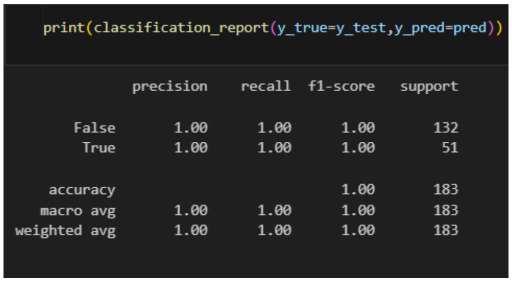
```
y_pred = knn.predict(X_test)
```

```
#Calculate accuracy of the model

from sklearn.metrics import accuracy_score
accuracy_KNN = accuracy_score(y_test, y_pred)
print(f'Accuracy_KNN: {accuracy_KNN*100}')
```

```
Accuracy_KNN: 96.17486338797814
```

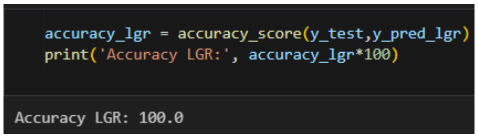
Model Validation and Evaluation Report:

<table>
<tr><th>Model</th><th>Classification Report</th><th>Accuracy</th></tr>
<tr><td>Logistic Regression</td><td>



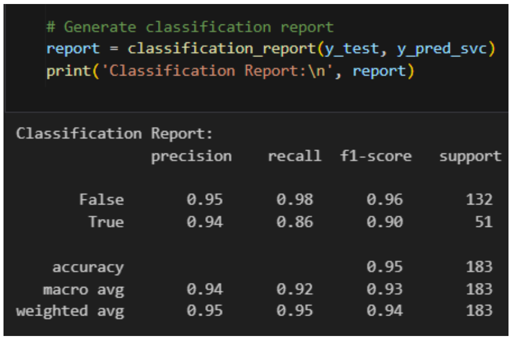
```
print(classification_report(y_true=y_test,y_pred=pred))

              precision    recall  f1-score   support

       False       1.00      1.00      1.00       132
        True       1.00      1.00      1.00        51

    accuracy                           1.00       183
   macro avg       1.00      1.00      1.00       183
weighted avg       1.00      1.00      1.00       183
```

</td><td>



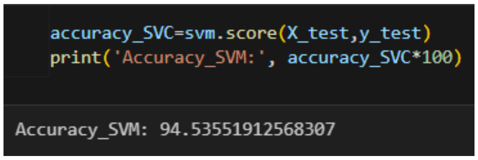
```
accuracy_lgr = accuracy_score(y_test,y_pred_lgr)
print('Accuracy LGR:', accuracy_lgr*100)

Accuracy LGR: 100.0
```

</td></tr>
<tr><td>SVM</td><td>



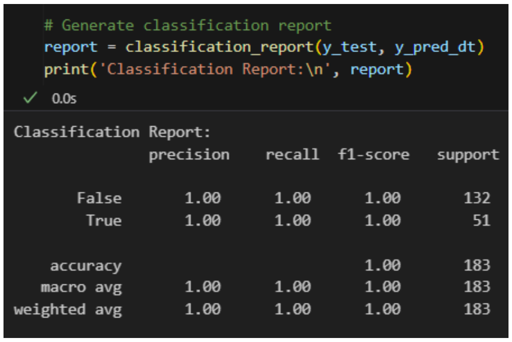
```
# Generate classification report
report = classification_report(y_test, y_pred_svc)
print('Classification Report:\n', report)

Classification Report:
              precision    recall  f1-score   support

       False       0.95      0.98      0.96       132
        True       0.94      0.86      0.90        51

    accuracy                           0.95       183
   macro avg       0.94      0.92      0.93       183
weighted avg       0.95      0.95      0.94       183
```

</td><td>



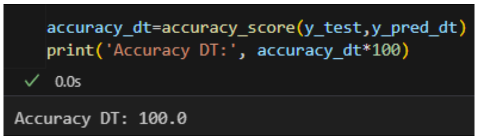
```
accuracy_SVC=svm.score(X_test,y_test)
print('Accuracy_SVM:', accuracy_SVC*100)

Accuracy_SVM: 94.53551912568307
```

</td></tr>
<tr><td>Decision Tree</td><td>



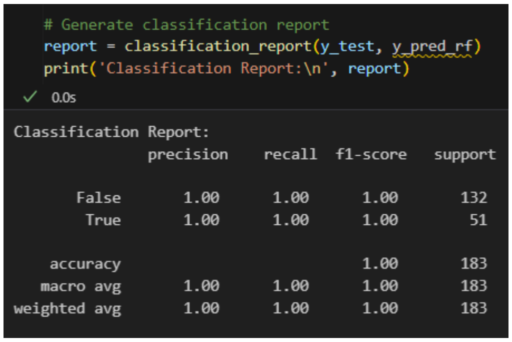
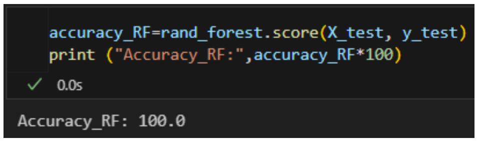
```
# Generate classification report
report = classification_report(y_test, y_pred_dt)
print('Classification Report:\n', report)
✓ 0.0s

Classification Report:
              precision    recall  f1-score   support

       False       1.00      1.00      1.00       132
        True       1.00      1.00      1.00        51

    accuracy                           1.00       183
   macro avg       1.00      1.00      1.00       183
weighted avg       1.00      1.00      1.00       183
```

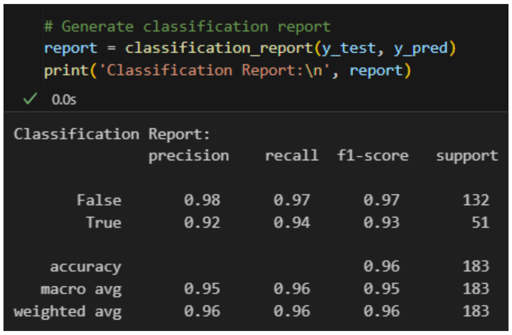
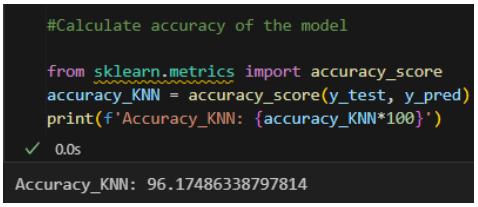
</td><td>



```
accuracy_dt=accuracy_score(y_test,y_pred_dt)
print('Accuracy DT:', accuracy_dt*100)
✓ 0.0s

Accuracy DT: 100.0
```

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

Random Forest	
	`# Generate classification report`
	`report = classification_report(y_test, y_pred_rf)`
	`print('Classification Report:\n', report)`
	✓ 0.0s
	`Classification Report:`
	`              precision    recall  f1-score   support`
	`       False       1.00      1.00      1.00       132`
	`        True       1.00      1.00      1.00        51`
	`    accuracy                           1.00       183`
	`   macro avg       1.00      1.00      1.00       183`
	`weighted avg       1.00      1.00      1.00       183`
		
		`accuracy_RF=rand_forest.score(X_test, y_test)`
		`print ("Accuracy_RF:",accuracy_RF*100)`
		✓ 0.0s
		`Accuracy_RF: 100.0`

KNN	
	`# Generate classification report`
	`report = classification_report(y_test, y_pred)`
	`print('Classification Report:\n', report)`
	✓ 0.0s
	`Classification Report:`
	`              precision    recall  f1-score   support`
	`       False       0.98      0.97      0.97       132`
	`        True       0.92      0.94      0.93        51`
	`    accuracy                           0.96       183`
	`   macro avg       0.95      0.96      0.95       183`
	`weighted avg       0.96      0.96      0.96       183`
		
		`#Calculate accuracy of the model`
		`from sklearn.metrics import accuracy_score`
		`accuracy_KNN = accuracy_score(y_test, y_pred)`
		`print(f'Accuracy_KNN: {accuracy_KNN*100}')`
		✓ 0.0s
		`Accuracy_KNN: 96.17486338797814`