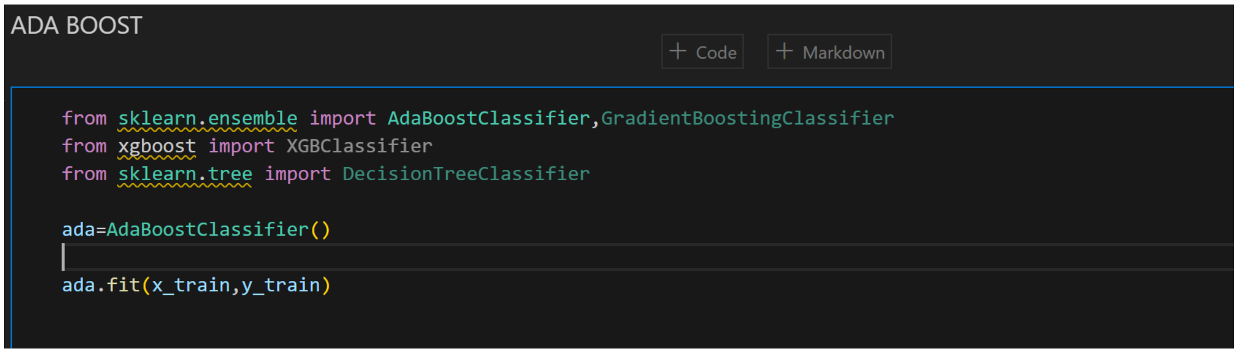
# Model Development Phase Template

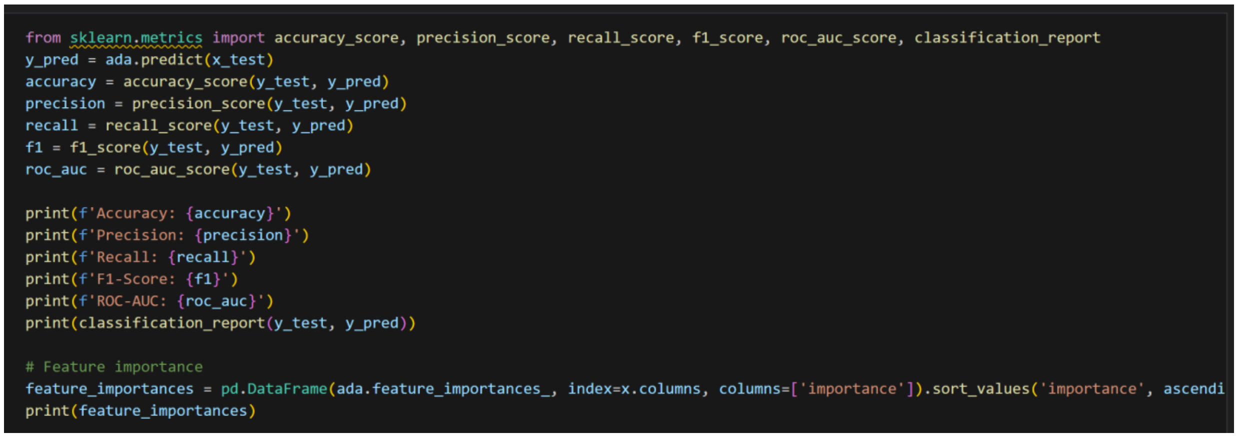
| | |
|---|---|
| Date | 15 July 2024 |
| Team ID | SWTID1720151584 |
| Project Title | Early Prediction of Chronic Kidney Disease |
| Maximum Marks | 4 Marks |

**Initial Model Training Code, Model Validation and Evaluation Report**

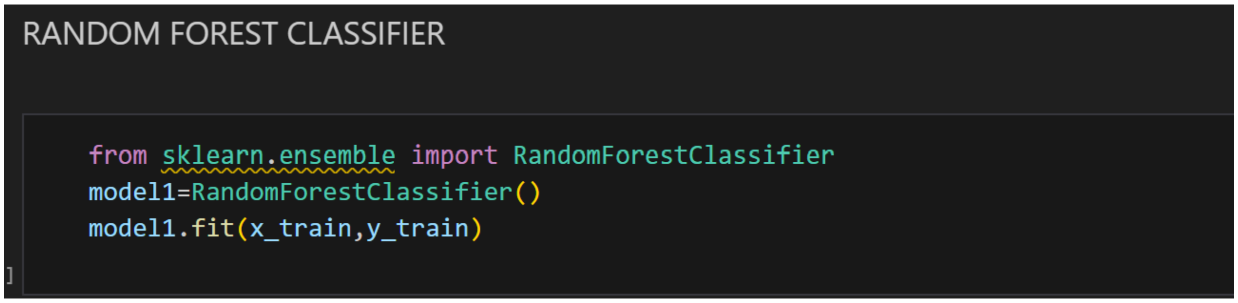
The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.



```
from sklearn.ensemble import AdaBoostClassifier,GradientBoostingClassifier
from xgboost import XGBClassifier
from sklearn.tree import DecisionTreeClassifier

ada=AdaBoostClassifier()

ada.fit(x_train,y_train)
```

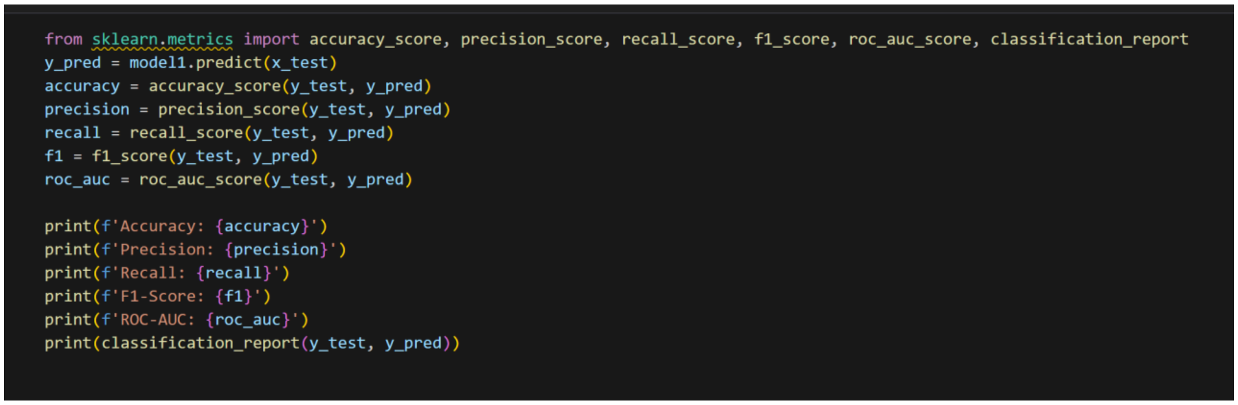


```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, classification_report
y_pred = ada.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1-Score: {f1}')
print(f'ROC-AUC: {roc_auc}')
print(classification_report(y_test, y_pred))

# Feature importance
feature_importances = pd.DataFrame(ada.feature_importances_, index=x.columns, columns=['importance']).sort_values('importance', ascendi
print(feature_importances)
```

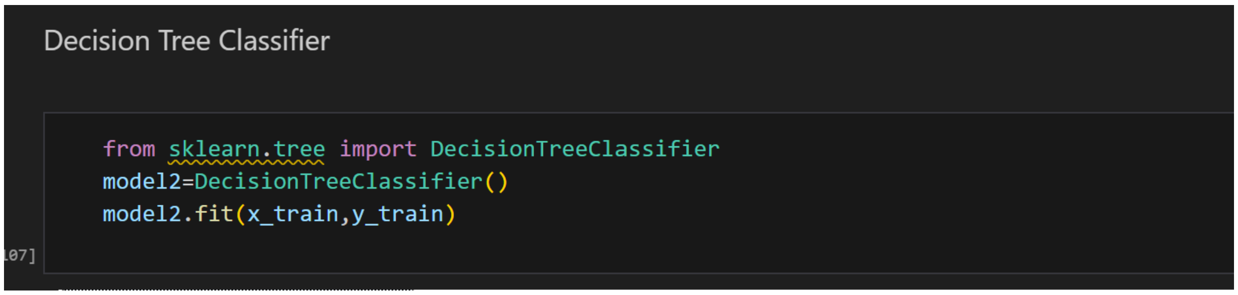


## RANDOM FOREST CLASSIFIER

```python
from sklearn.ensemble import RandomForestClassifier
model1=RandomForestClassifier()
model1.fit(x_train,y_train)
```

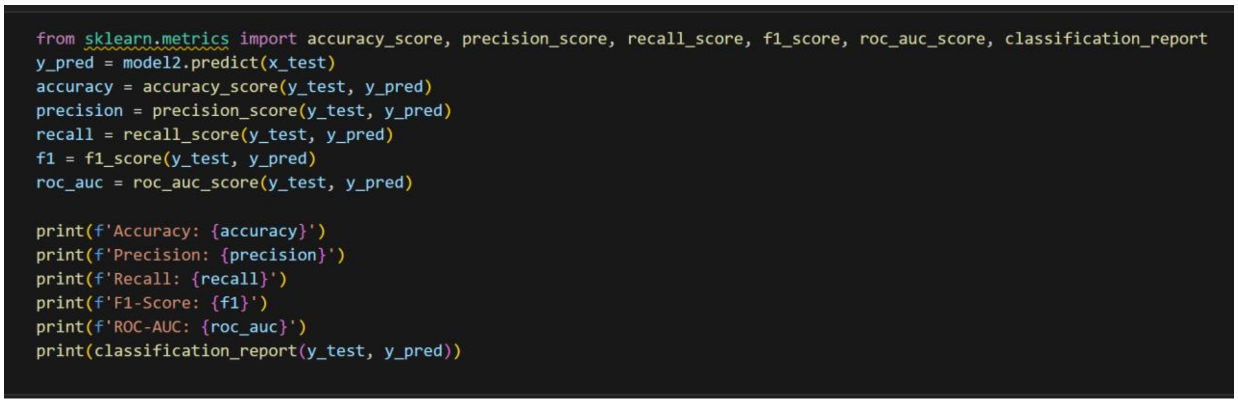


```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, classification_report
y_pred = model1.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1-Score: {f1}')
print(f'ROC-AUC: {roc_auc}')
print(classification_report(y_test, y_pred))
```

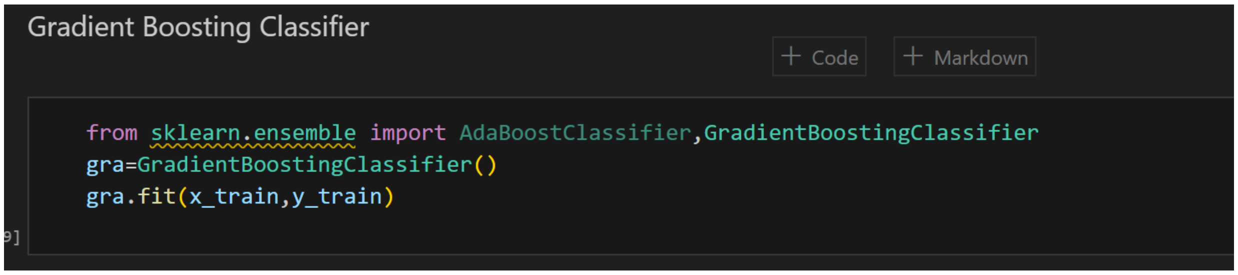


## Decision Tree Classifier

```python
from sklearn.tree import DecisionTreeClassifier
model2=DecisionTreeClassifier()
model2.fit(x_train,y_train)
```

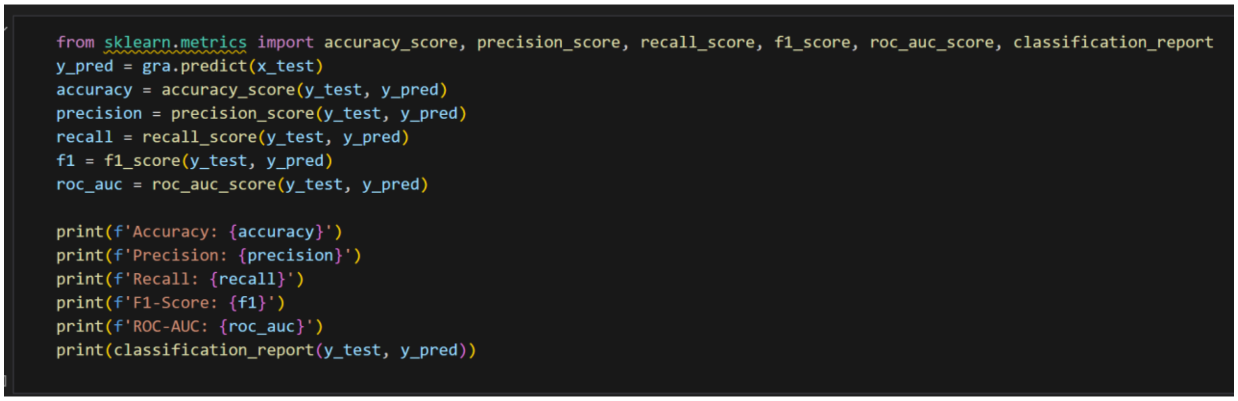


```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, classification_report
y_pred = model2.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1-Score: {f1}')
print(f'ROC-AUC: {roc_auc}')
print(classification_report(y_test, y_pred))
```



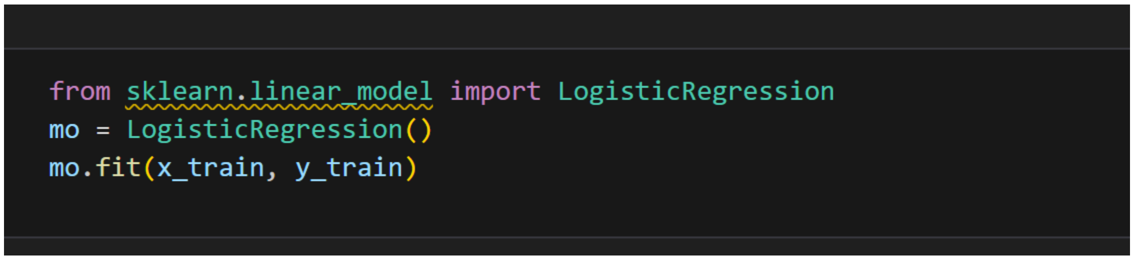
Gradient Boosting Classifier

```python
from sklearn.ensemble import AdaBoostClassifier,GradientBoostingClassifier
gra=GradientBoostingClassifier()
gra.fit(x_train,y_train)
```

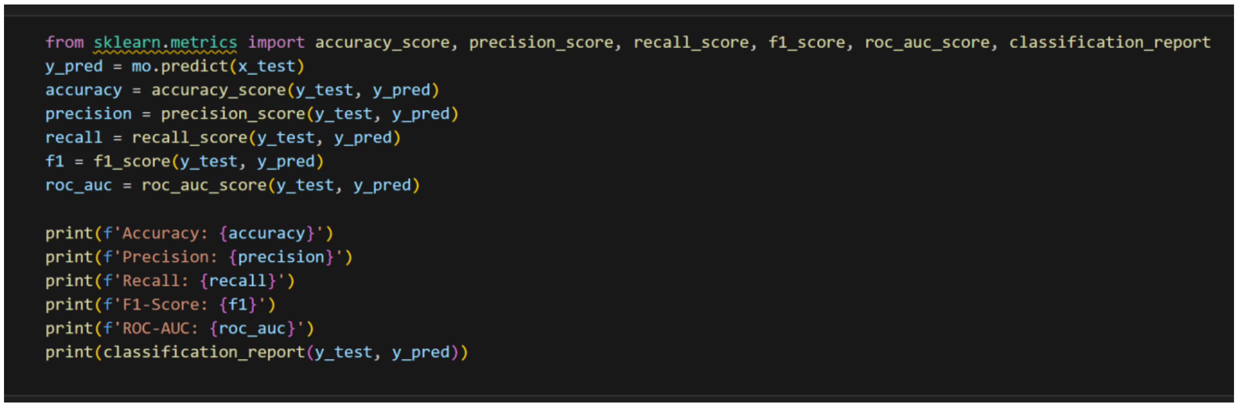


```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, classification_report
y_pred = gra.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1-Score: {f1}')
print(f'ROC-AUC: {roc_auc}')
print(classification_report(y_test, y_pred))
```

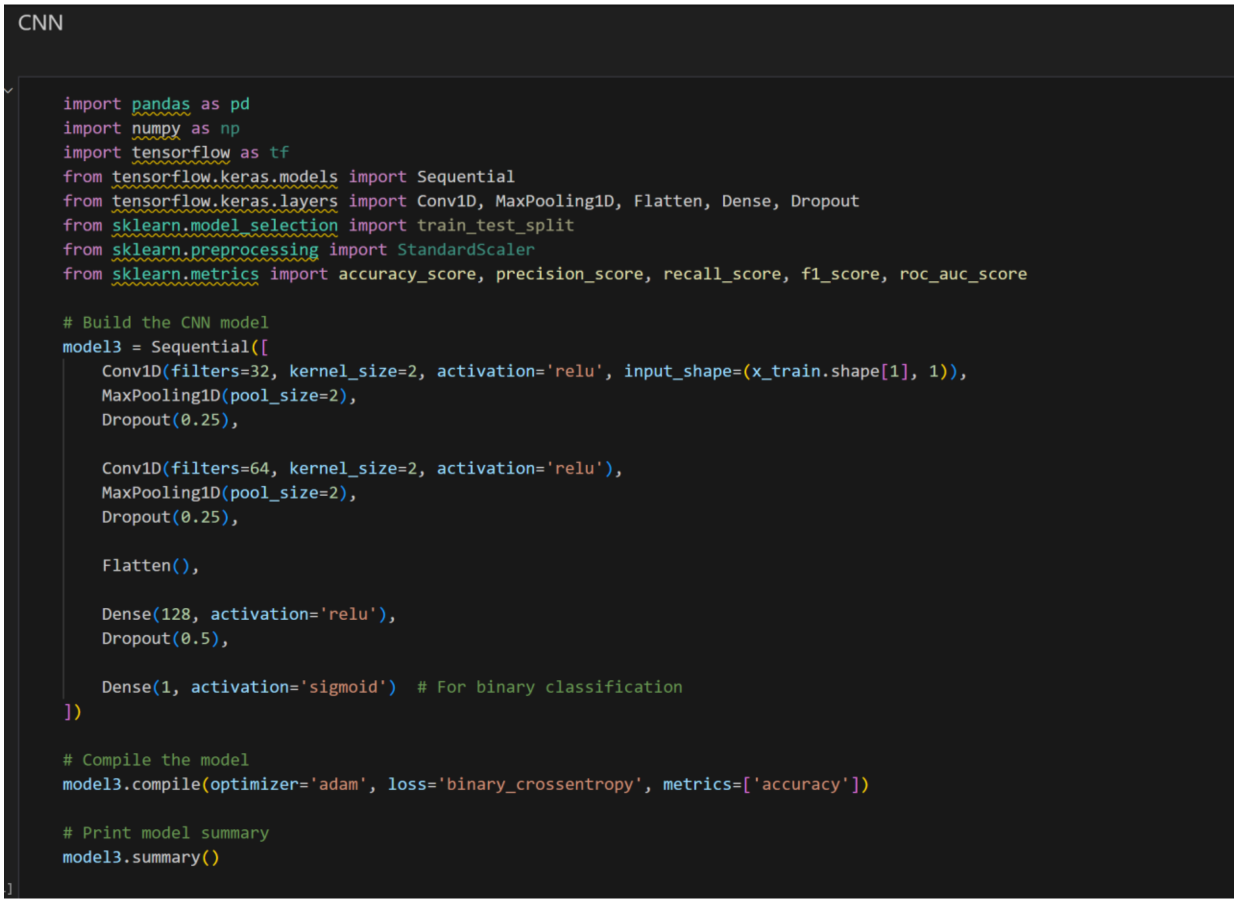
Logistic Regression



```python
from sklearn.linear_model import LogisticRegression
mo = LogisticRegression()
mo.fit(x_train, y_train)
```

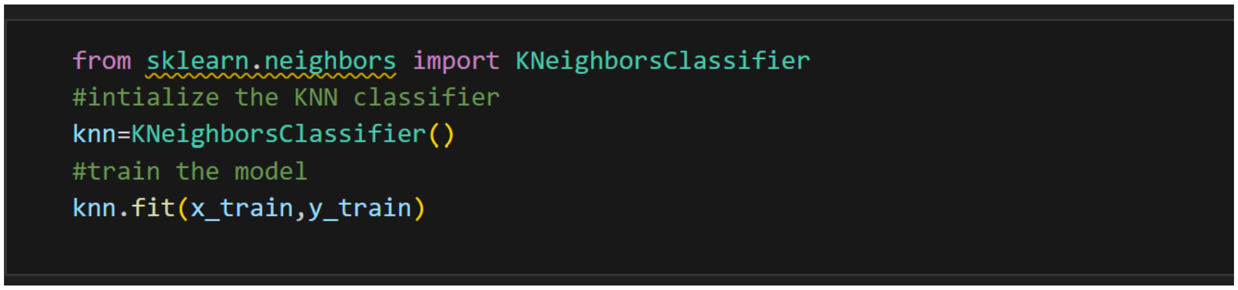


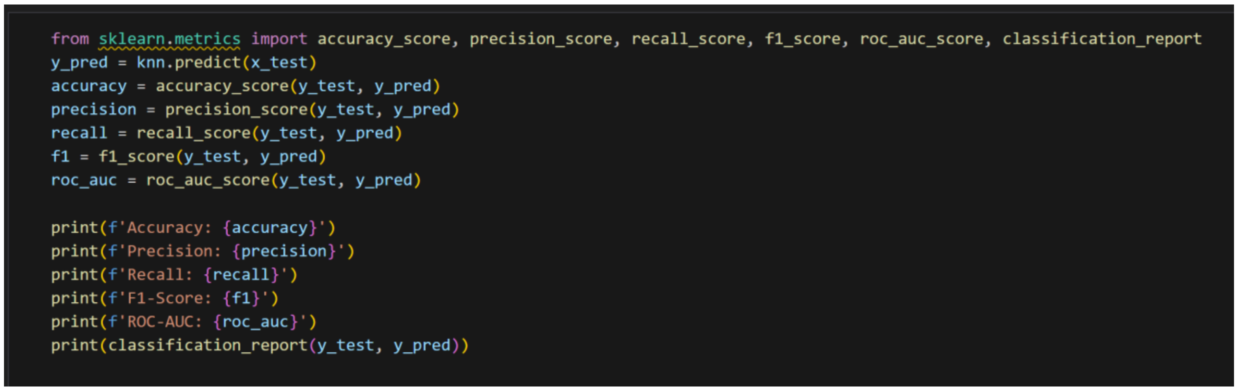
```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, classification_report
y_pred = mo.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1-Score: {f1}')
print(f'ROC-AUC: {roc_auc}')
print(classification_report(y_test, y_pred))
```


CNN

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten, Dense, Dropout
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score

# Build the CNN model
model3 = Sequential([
    Conv1D(filters=32, kernel_size=2, activation='relu', input_shape=(x_train.shape[1], 1)),
    MaxPooling1D(pool_size=2),
    Dropout(0.25),

    Conv1D(filters=64, kernel_size=2, activation='relu'),
    MaxPooling1D(pool_size=2),
    Dropout(0.25),

    Flatten(),

    Dense(128, activation='relu'),
    Dropout(0.5),

    Dense(1, activation='sigmoid')  # For binary classification
])

# Compile the model
model3.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Print model summary
model3.summary()
```

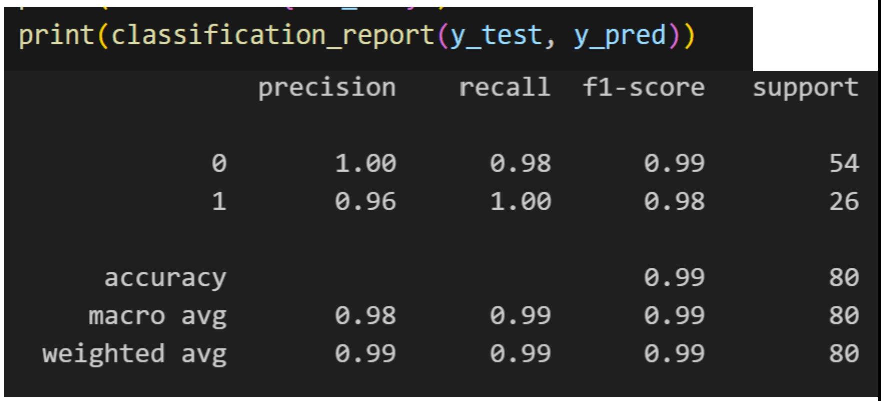
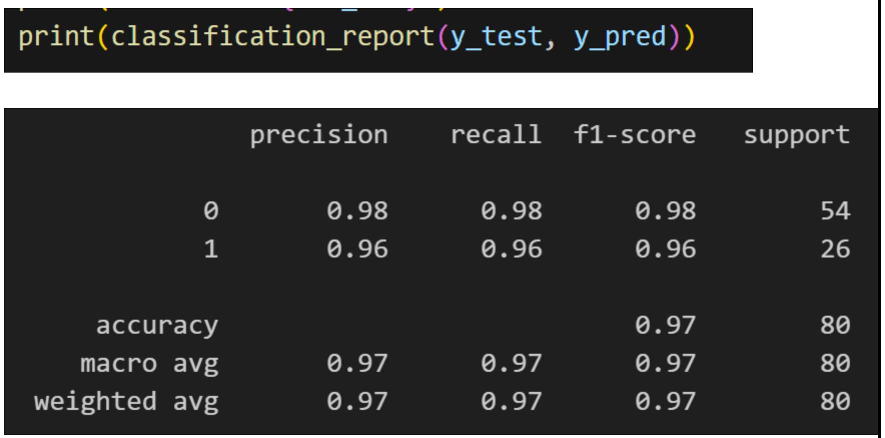
KNN


```
from sklearn.neighbors import KNeighborsClassifier
#intialize the KNN classifier
knn=KNeighborsClassifier()
#train the model
knn.fit(x_train,y_train)
```



```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, classification_report
y_pred = knn.predict(x_test)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F1-Score: {f1}')
print(f'ROC-AUC: {roc_auc}')
print(classification_report(y_test, y_pred))
```

**Model Validation and Evaluation Report:**

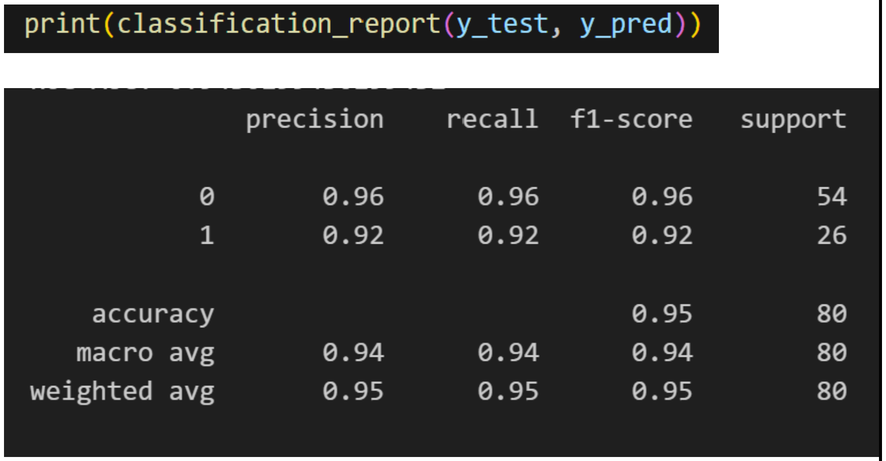
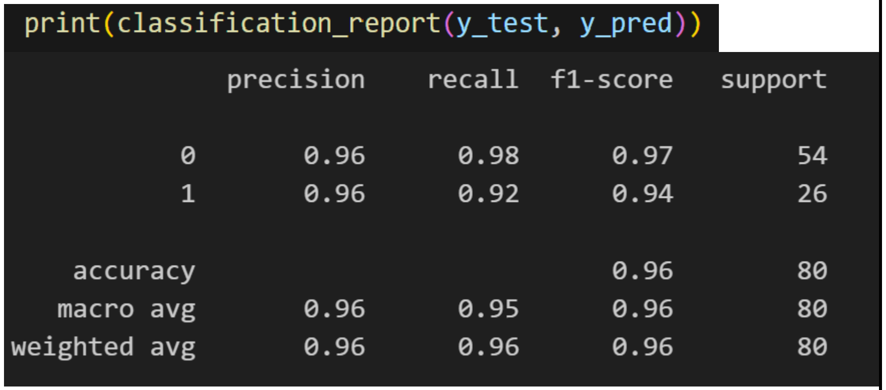
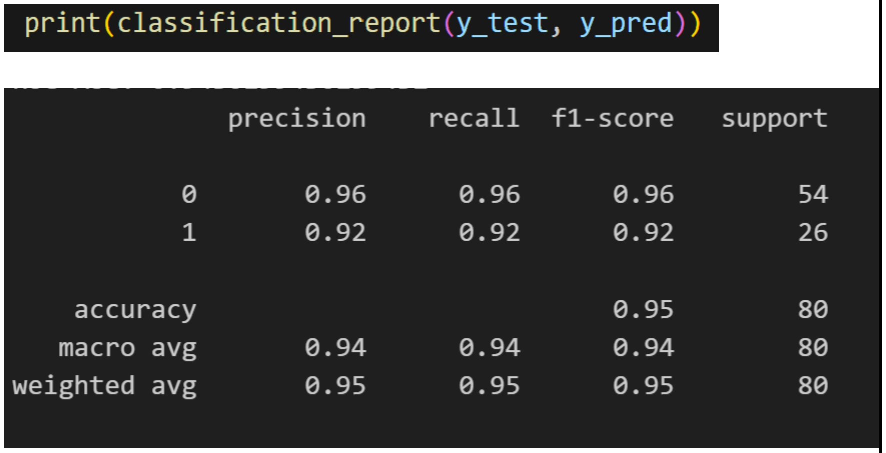
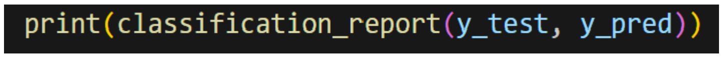
| Model | Classification Report | Accuracy | Confusion Matrix |
|---|---|---|---|
| Ada Boost Classifier |  | 98.75% | Screenshot of the confusion matrix |
| Random Forest Classifier |  | 97.5% | Screenshot of the confusion matrix |

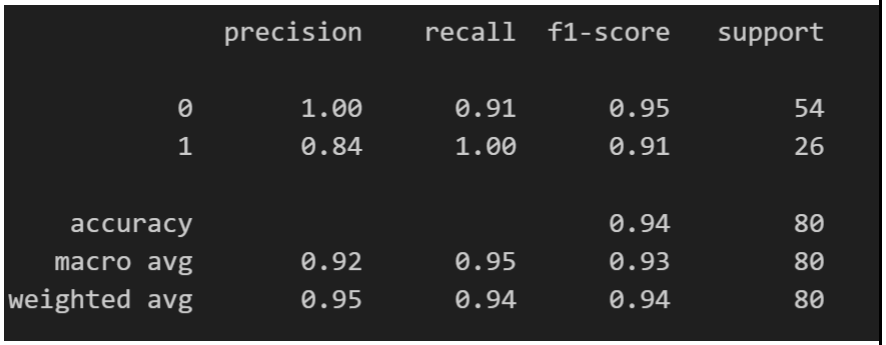
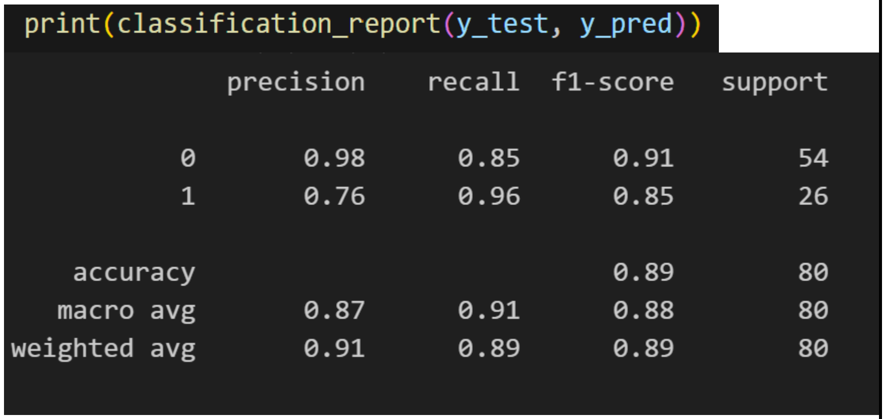
Ada Boost Classifier:

```
print(classification_report(y_test, y_pred))
              precision    recall  f1-score   support

           0       1.00      0.98      0.99        54
           1       0.96      1.00      0.98        26

    accuracy                           0.99        80
   macro avg       0.98      0.99      0.99        80
weighted avg       0.99      0.99      0.99        80
```

Random Forest Classifier:

```
print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

           0       0.98      0.98      0.98        54
           1       0.96      0.96      0.96        26

    accuracy                           0.97        80
   macro avg       0.97      0.97      0.97        80
weighted avg       0.97      0.97      0.97        80
```

| | | | |
|---|---|---|---|
| Decision Tree Classifier |  `print(classification_report(y_test, y_pred))`<br>`              precision    recall  f1-score   support`<br>`           0       0.96      0.96      0.96        54`<br>`           1       0.92      0.92      0.92        26`<br>`    accuracy                           0.95        80`<br>`   macro avg       0.94      0.94      0.94        80`<br>`weighted avg       0.95      0.95      0.95        80` | 95% | … |
| Gradient Boosting Classifier |  `print(classification_report(y_test, y_pred))`<br>`              precision    recall  f1-score   support`<br>`           0       0.96      0.98      0.97        54`<br>`           1       0.96      0.92      0.94        26`<br>`    accuracy                           0.96        80`<br>`   macro avg       0.96      0.95      0.96        80`<br>`weighted avg       0.96      0.96      0.96        80` | 96.25% | |
| XG Boost Classifier |  `print(classification_report(y_test, y_pred))`<br>`              precision    recall  f1-score   support`<br>`           0       0.96      0.96      0.96        54`<br>`           1       0.92      0.92      0.92        26`<br>`    accuracy                           0.95        80`<br>`   macro avg       0.94      0.94      0.94        80`<br>`weighted avg       0.95      0.95      0.95        80` | 95% | |
| Logistic Regression |  `print(classification_report(y_test, y_pred))` | 93.75% | |

| | | | |
|---|---|---|---|
| |  | | |
| K Neighbors Classifier |  | 88.75% | |

```
              precision    recall  f1-score   support

           0       1.00      0.91      0.95        54
           1       0.84      1.00      0.91        26

    accuracy                           0.94        80
   macro avg       0.92      0.95      0.93        80
weighted avg       0.95      0.94      0.94        80
```

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.85      0.91        54
           1       0.76      0.96      0.85        26

    accuracy                           0.89        80
   macro avg       0.87      0.91      0.88        80
weighted avg       0.91      0.89      0.89        80
```