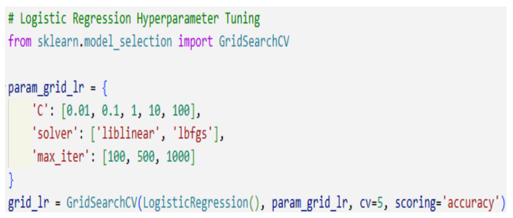
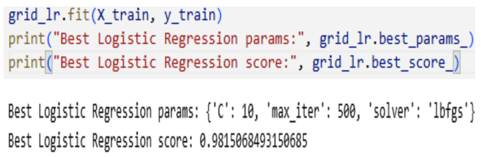
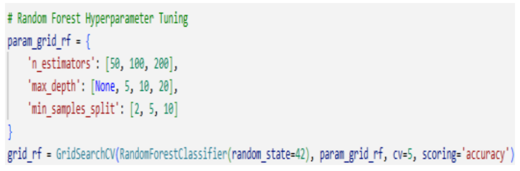
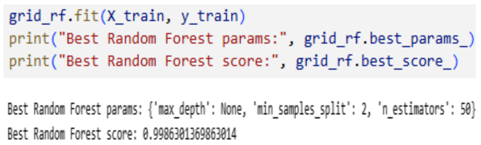
Model Optimization and Tuning Phase Report

Date	08 August 2025
Skill Wallet ID	**SWUID20250188325**
Project Title	Predictive Pulse: Harnessing Machine Learning for Blood Pressure Analysis
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

Hyperparameter Tuning Documentation (6 Marks):

Model	**Tuned Hyperparameters**	**Optimal Values**
Logistic Regression		
Random Forest		

```python
# Logistic Regression Hyperparameter Tuning
from sklearn.model_selection import GridSearchCV

param_grid_lr = {
    'C': [0.01, 0.1, 1, 10, 100],
    'solver': ['liblinear', 'lbfgs'],
    'max_iter': [100, 500, 1000]
}
grid_lr = GridSearchCV(LogisticRegression(), param_grid_lr, cv=5, scoring='accuracy')
```

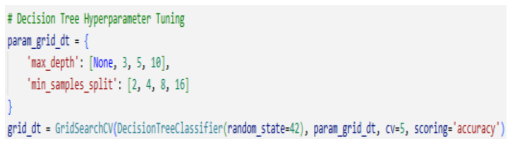
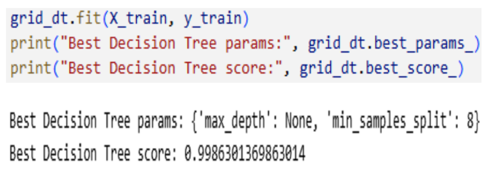
```python
grid_lr.fit(X_train, y_train)
print("Best Logistic Regression params:", grid_lr.best_params_)
print("Best Logistic Regression score:", grid_lr.best_score_)
```

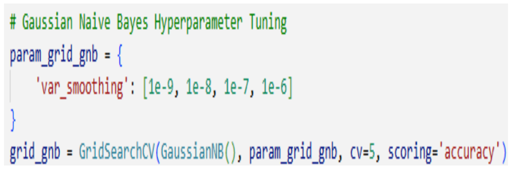
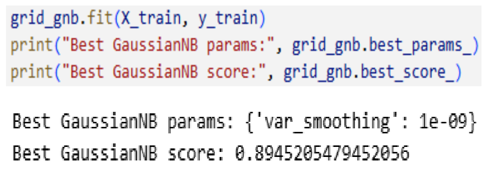
```
Best Logistic Regression params: {'C': 10, 'max_iter': 500, 'solver': 'lbfgs'}
Best Logistic Regression score: 0.9815068493150685
```

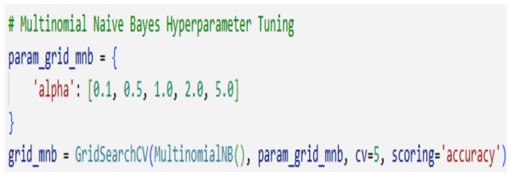
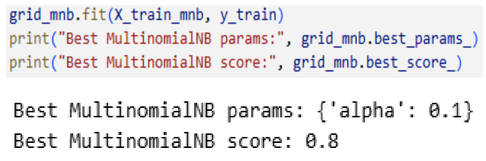
```python
# Random Forest Hyperparameter Tuning
param_grid_rf = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 5, 10, 20],
    'min_samples_split': [2, 5, 10]
}
grid_rf = GridSearchCV(RandomForestClassifier(random_state=42), param_grid_rf, cv=5, scoring='accuracy')
```

```python
grid_rf.fit(X_train, y_train)
print("Best Random Forest params:", grid_rf.best_params_)
print("Best Random Forest score:", grid_rf.best_score_)
```

```
Best Random Forest params: {'max_depth': None, 'min_samples_split': 2, 'n_estimators': 50}
Best Random Forest score: 0.9986301369863014
```

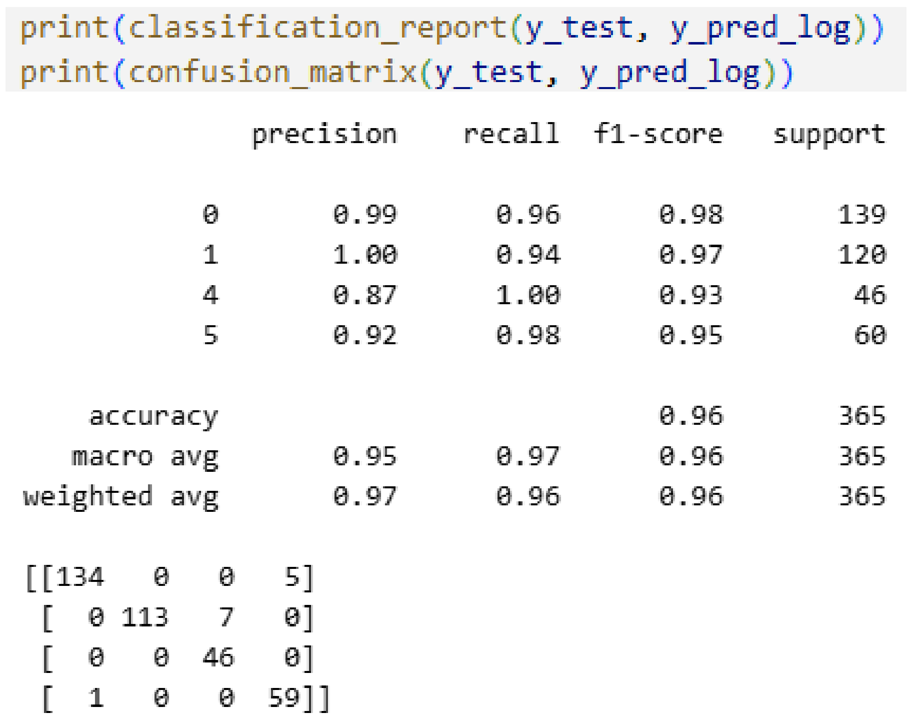
Decision Tree	 `# Decision Tree Hyperparameter Tuning`
	`param_grid_dt = {`
	`    'max_depth': [None, 3, 5, 10],`
	`    'min_samples_split': [2, 4, 8, 16]`
	`}`
	`grid_dt = GridSearchCV(DecisionTreeClassifier(random_state=42), param_grid_dt, cv=5, scoring='accuracy')`
		 `grid_dt.fit(X_train, y_train)`
		`print("Best Decision Tree params:", grid_dt.best_params_)`
		`print("Best Decision Tree score:", grid_dt.best_score_)`
		
		`Best Decision Tree params: {'max_depth': None, 'min_samples_split': 8}`
		`Best Decision Tree score: 0.9986301369863014`

Gaussian Navie Bayes	 `# Gaussian Naive Bayes Hyperparameter Tuning`
	`param_grid_gnb = {`
	`    'var_smoothing': [1e-9, 1e-8, 1e-7, 1e-6]`
	`}`
	`grid_gnb = GridSearchCV(GaussianNB(), param_grid_gnb, cv=5, scoring='accuracy')`
		 `grid_gnb.fit(X_train, y_train)`
		`print("Best GaussianNB params:", grid_gnb.best_params_)`
		`print("Best GaussianNB score:", grid_gnb.best_score_)`
		
		`Best GaussianNB params: {'var_smoothing': 1e-09}`
		`Best GaussianNB score: 0.8945205479452056`

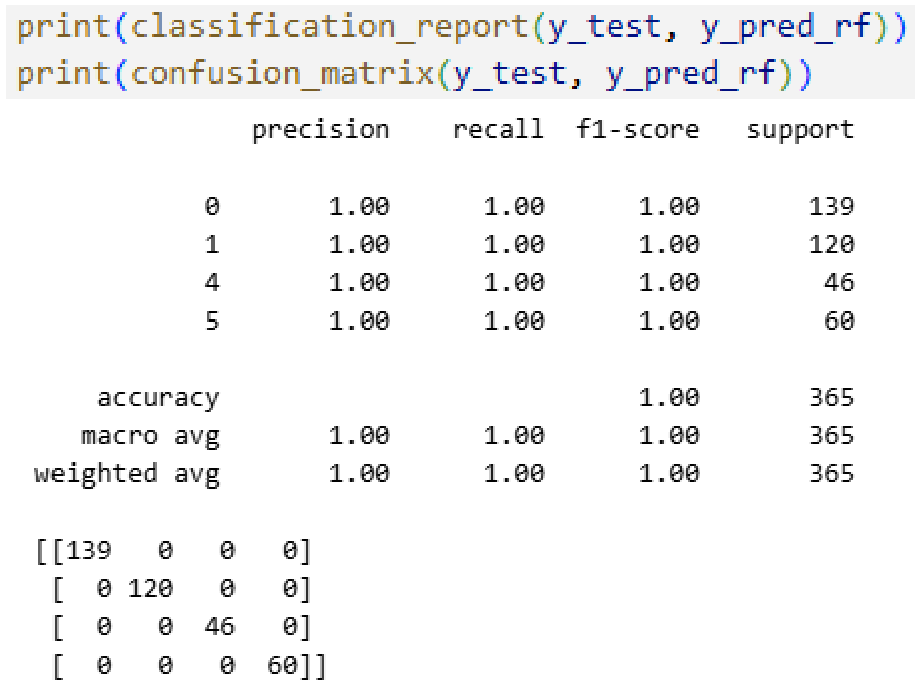
Multinomial Navie Bayes	 `# Multinomial Naive Bayes Hyperparameter Tuning`
	`param_grid_mnb = {`
	`    'alpha': [0.1, 0.5, 1.0, 2.0, 5.0]`
	`}`
	`grid_mnb = GridSearchCV(MultinomialNB(), param_grid_mnb, cv=5, scoring='accuracy')`
		 `grid_mnb.fit(X_train_mnb, y_train)`
		`print("Best MultinomialNB params:", grid_mnb.best_params_)`
		`print("Best MultinomialNB score:", grid_mnb.best_score_)`
		
		`Best MultinomialNB params: {'alpha': 0.1}`
		`Best MultinomialNB score: 0.8`

Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric

Logistic Regression	 `print(classification_report(y_test, y_pred_log))`
	`print(confusion_matrix(y_test, y_pred_log))`
	
	`              precision    recall  f1-score   support`
	
	`           0       0.99      0.96      0.98       139`
	`           1       1.00      0.94      0.97       120`
	`           4       0.87      1.00      0.93        46`
	`           5       0.92      0.98      0.95        60`
	
	`    accuracy                           0.96       365`
	`   macro avg       0.95      0.97      0.96       365`
	`weighted avg       0.97      0.96      0.96       365`
	
	`[[134   0   0   5]`
	` [  0 113   7   0]`
	` [  0   0  46   0]`
	` [  1   0   0  59]]`

Random Forest



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

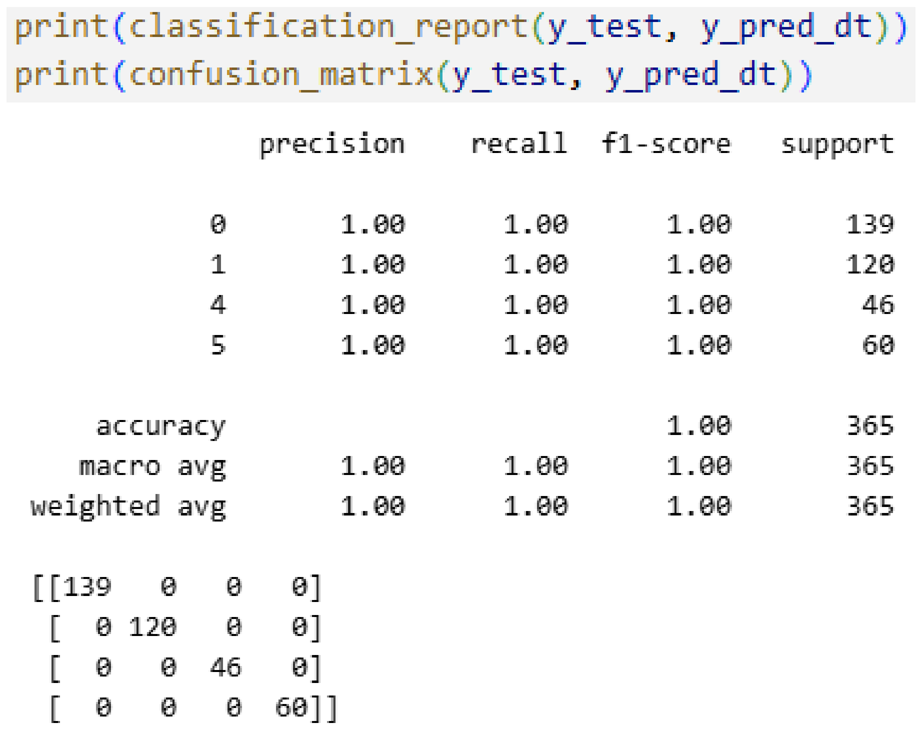
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       139
           1       1.00      1.00      1.00       120
           4       1.00      1.00      1.00        46
           5       1.00      1.00      1.00        60

    accuracy                           1.00       365
   macro avg       1.00      1.00      1.00       365
weighted avg       1.00      1.00      1.00       365

[[139   0   0   0]
 [  0 120   0   0]
 [  0   0  46   0]
 [  0   0   0  60]]
```

Decision Tree



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

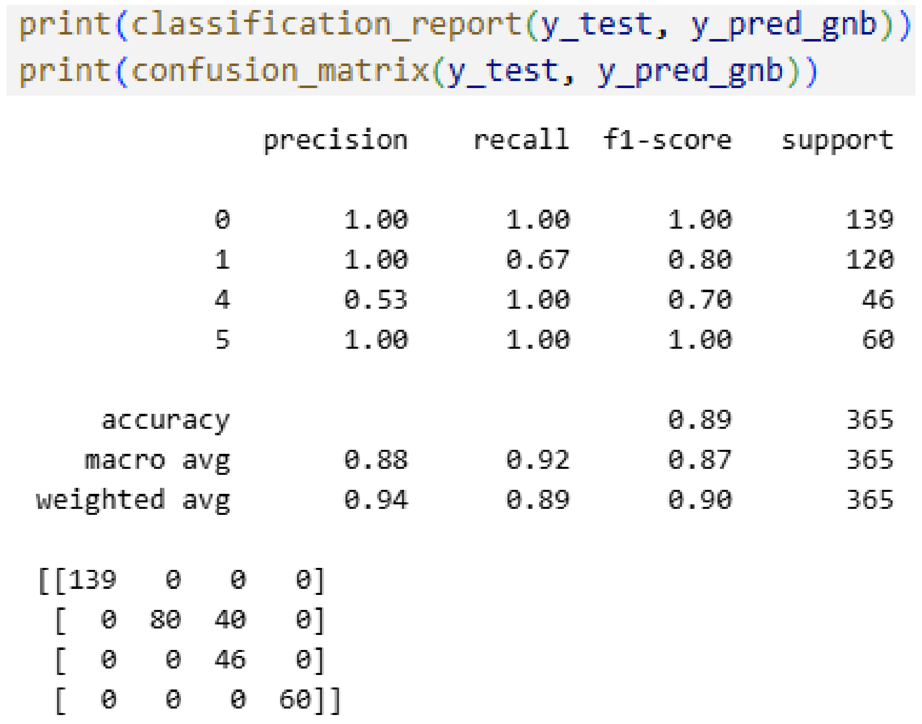
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       139
           1       1.00      1.00      1.00       120
           4       1.00      1.00      1.00        46
           5       1.00      1.00      1.00        60

    accuracy                           1.00       365
   macro avg       1.00      1.00      1.00       365
weighted avg       1.00      1.00      1.00       365

[[139   0   0   0]
 [  0 120   0   0]
 [  0   0  46   0]
 [  0   0   0  60]]
```

Gaussian Navie Bayes



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

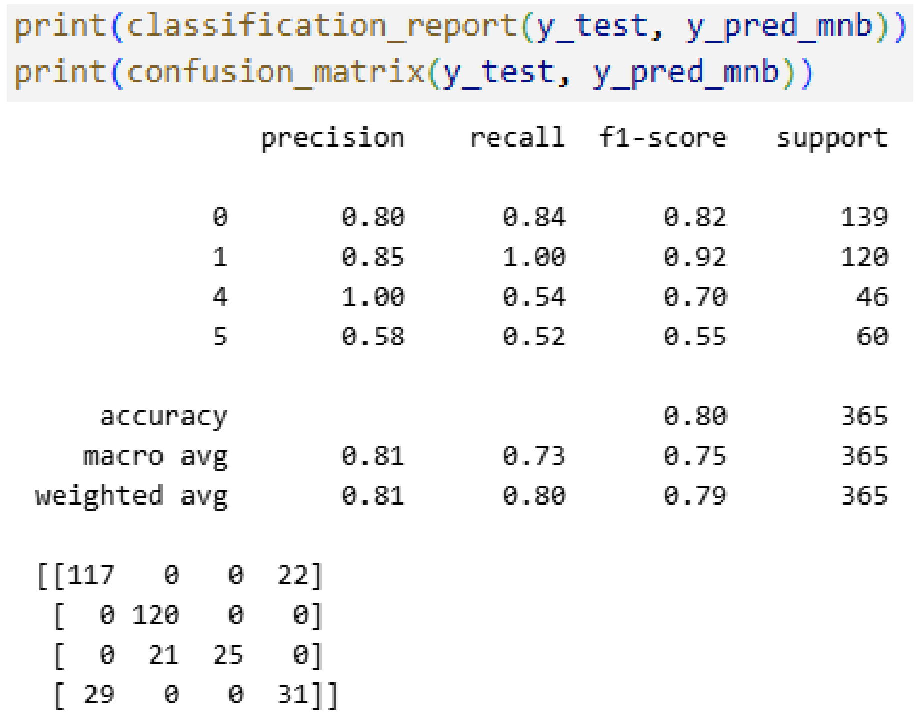
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       139
           1       1.00      0.67      0.80       120
           4       0.53      1.00      0.70        46
           5       1.00      1.00      1.00        60

    accuracy                           0.89       365
   macro avg       0.88      0.92      0.87       365
weighted avg       0.94      0.89      0.90       365

[[139   0   0   0]
 [  0  80  40   0]
 [  0   0  46   0]
 [  0   0   0  60]]
```

Multinomial Navie Bayes



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

```
              precision    recall  f1-score   support

           0       0.80      0.84      0.82       139
           1       0.85      1.00      0.92       120
           4       1.00      0.54      0.70        46
           5       0.58      0.52      0.55        60

    accuracy                           0.80       365
   macro avg       0.81      0.73      0.75       365
weighted avg       0.81      0.80      0.79       365

[[117   0   0  22]
 [  0 120   0   0]
 [  0  21  25   0]
 [ 29   0   0  31]]
```

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Random Forest (RF)	Random Forest Model is used because it consistently delivers high predictive accuracy and demonstrates strong generalization on both training and test data. Random Forest is robust to overfitting due to its ensemble approach, which averages the results of multiple decision trees. It can effectively handle complex, non-linear relationships and is less sensitive to outliers and noise in the dataset. Additionally, Random Forest provides valuable insights into feature importance, helping to interpret which variables most influence predictions. Its overall performance and reliability make it an excellent choice for this classification problem.