Phase_2

Submitted to:

Dr. Mahmoud Mounir

TA: Esraa Karam

Submitted by:

Nagy Ahmed Nagy	23p0365
Abdelrhman Mohammed Mahmoud Salah	23p0370
Ahmed Mosta Gomaa	23p0375
John George Mikhael	23P0266
Peter Maged Shokry	23p0192

Table of Contents

Githup Repo: https://github.com/abdelrhmanaj/Data_Mining_Project

1.Introduction

Diabetes is one of the most common and hazardous diseases worldwide. It is a metabolic disorder that requires careful monitoring, regular medication, and lifestyle adjustments to keep it under control. Early diagnosis is crucial for preventing severe complications such as cardiovascular disease, kidney failure, and vision loss.

This project explores the **Pima Indians Diabetes dataset (or a similar dataset)**, which contains patients' medical and laboratory information, to implement and evaluate different machine learning classification models. The aim is to predict whether a patient is Non-Diabetic, Predict-Diabetic, or Diabetic based on their medical measurements.

Dataset Description:
The dataset contains the following attributes:

- Patient ID – Unique identifier
- Age – Patient's age
- Gender – Male/Female
- Sugar Level – Blood sugar measurement
- Creatinine (Cr) – Kidney function indicator
- Urea – Blood urea nitrogen
- Body Mass Index (BMI) – Weight/height^2 ratio
- Cholesterol (Chol) – Total cholesterol
- Fasting Lipid Profile – Including LDL, HDL, VLDL, Triglycerides (TG)
- HbA1c – Glycated hemoglobin (long-term sugar indicator)
- Class – Target label (Non-Diabetic, Predict-Diabetic, Diabetic)

Objective:

- Clean and preprocess the data
- Identify the most significant features
- Implement and evaluate six classification models:
    1. Decision Tree
    2. Naive Bayes
    3. K-Nearest Neighbors (KNN)
    4. Support Vector Machine (SVM)
    5. Logistic Regression
    6. Random Forest
- Evaluate models using 5-Fold Cross-Validation, GridSearchCV, accuracy, precision, recall, F1-score, and confusion matrices
- Visualize results to understand feature importance and model performance
- Provide insights into factors affecting diabetes onset (age, gender, BMI, HbA1c, etc.)

2. Data Preprocessing

2.1 Feature Selection

- Selected strong features: AGE, BMI, HbA1c
- Kept all features for comparison.

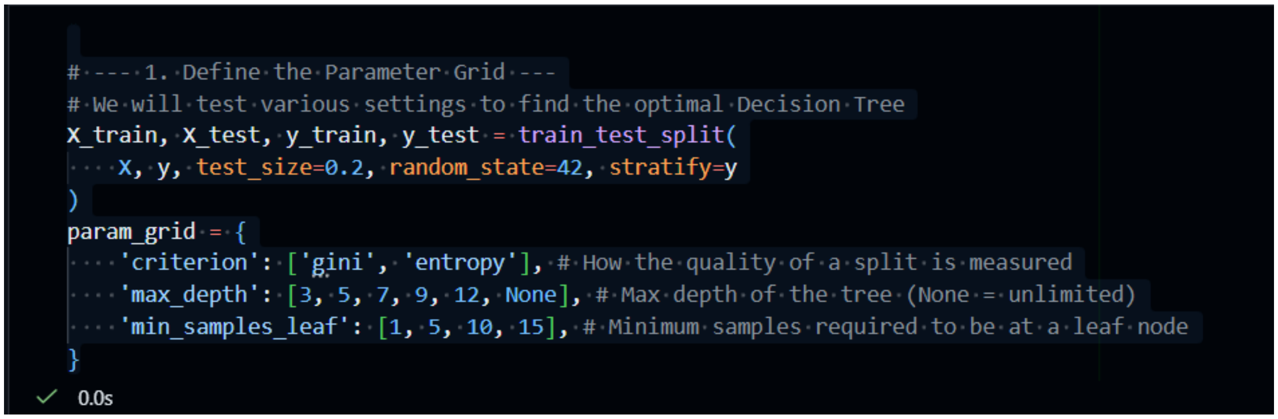
2.2 Feature Scaling & Encoding

- Used MinMaxScaler or StandardScaler where needed (especially for KNN, SVM, Logistic Regression).
- Discretization of continuous features for Naive Bayes using KBinsDiscretizer

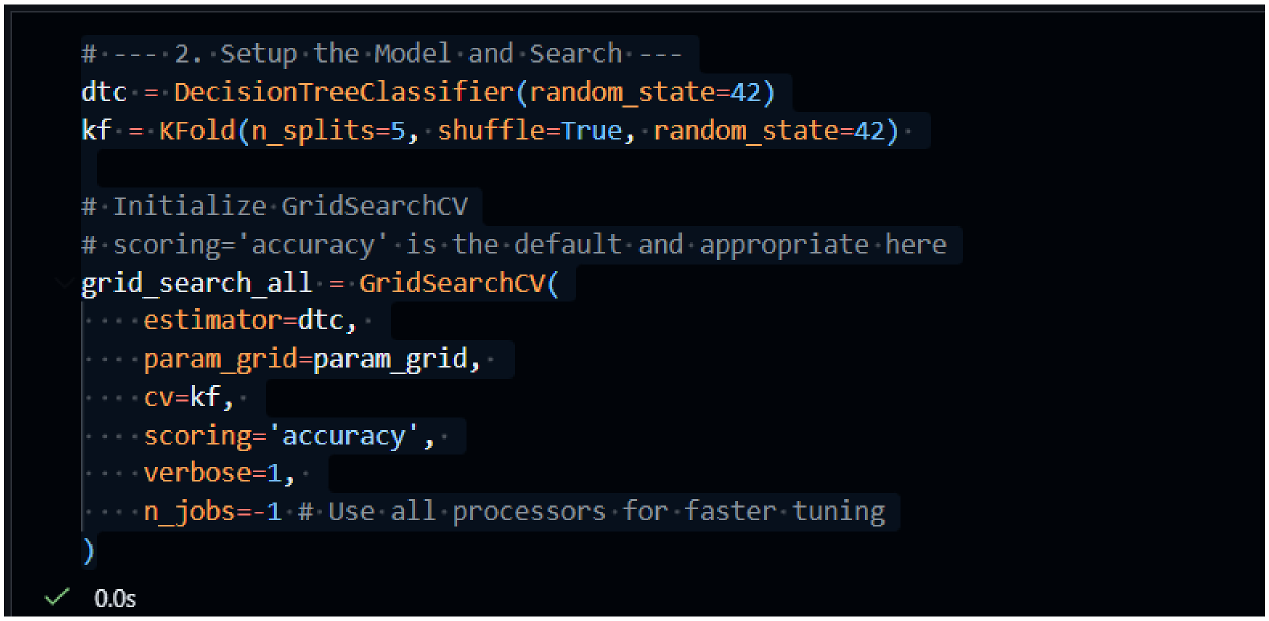
3. Model Implementation

3.1 Decision Tree

- Hyperparameter tuning using GridSearchCV and 5-Fold CV:



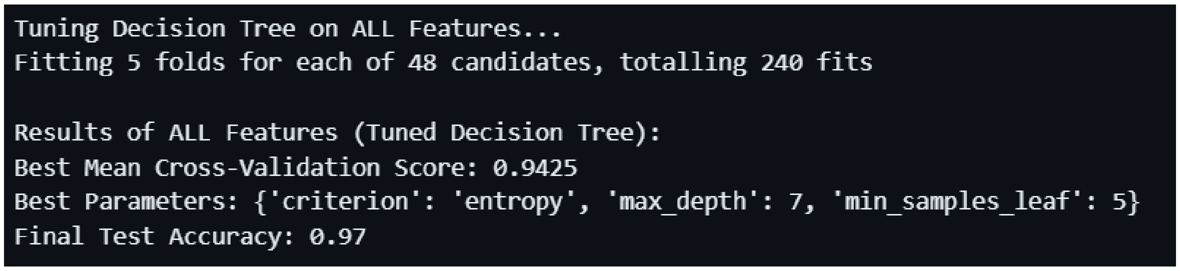
```python
# ---- 1. Define the Parameter Grid ---
# We will test various settings to find the optimal Decision Tree
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
param_grid = {
    'criterion': ['gini', 'entropy'], # How the quality of a split is measured
    'max_depth': [3, 5, 7, 9, 12, None], # Max depth of the tree (None = unlimited)
    'min_samples_leaf': [1, 5, 10, 15], # Minimum samples required to be at a leaf node
}
```
✓ 0.0s



```python
# ---- 2. Setup the Model and Search ---
dtc = DecisionTreeClassifier(random_state=42)
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Initialize GridSearchCV
# scoring='accuracy' is the default and appropriate here
grid_search_all = GridSearchCV(
    estimator=dtc,
    param_grid=param_grid,
    cv=kf,
    scoring='accuracy',
    verbose=1,
    n_jobs=-1 # Use all processors for faster tuning
)
```
✓ 0.0s

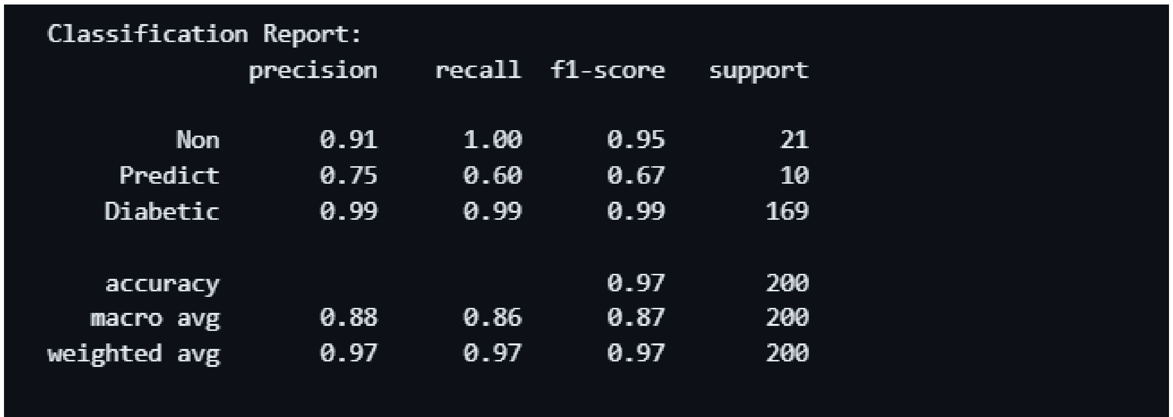
- Evaluation:

For All Features:

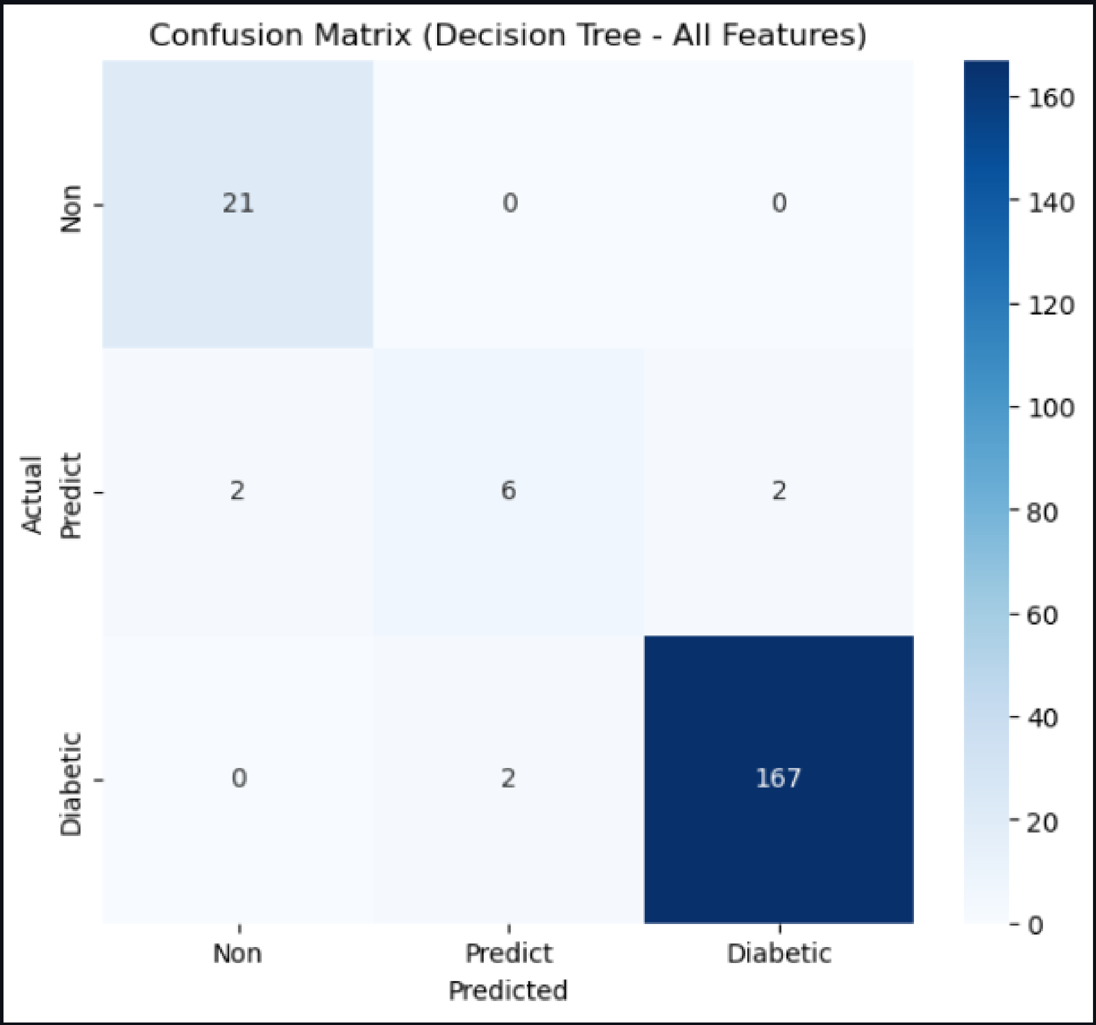


```
Tuning Decision Tree on ALL Features...
Fitting 5 folds for each of 48 candidates, totalling 240 fits

Results of ALL Features (Tuned Decision Tree):
Best Mean Cross-Validation Score: 0.9425
Best Parameters: {'criterion': 'entropy', 'max_depth': 7, 'min_samples_leaf': 5}
Final Test Accuracy: 0.97
```

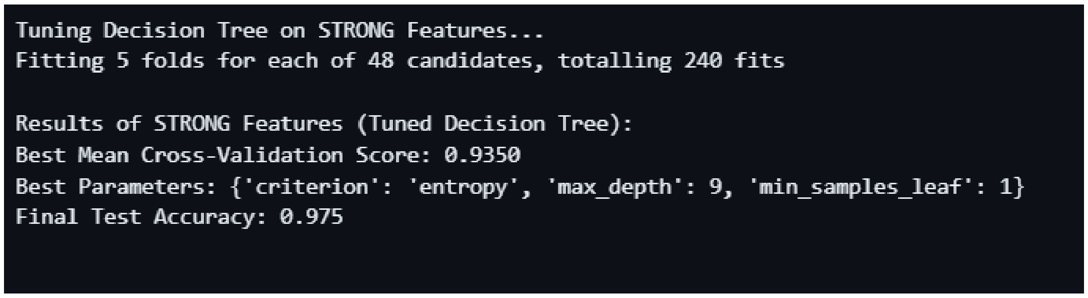
Confusion Matrix:



```
Classification Report:
              precision    recall  f1-score   support

         Non       0.91      1.00      0.95        21
     Predict       0.75      0.60      0.67        10
    Diabetic       0.99      0.99      0.99       169

    accuracy                           0.97       200
   macro avg       0.88      0.86      0.87       200
weighted avg       0.97      0.97      0.97       200
```

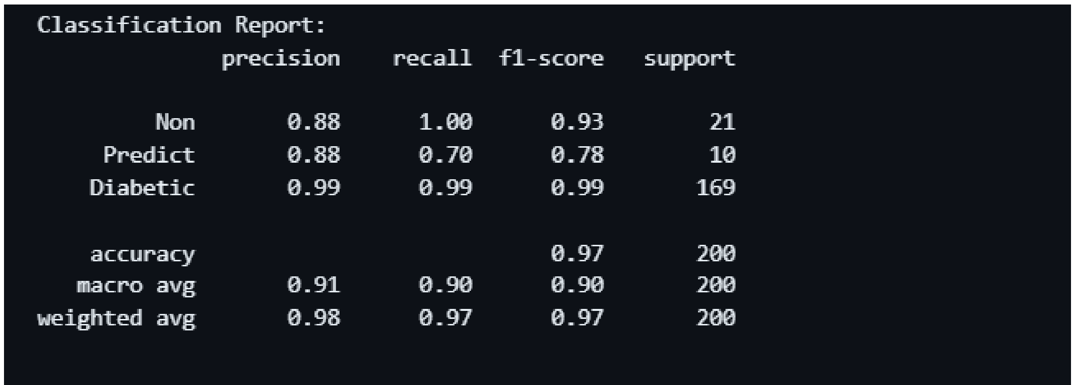


For Strong Features:

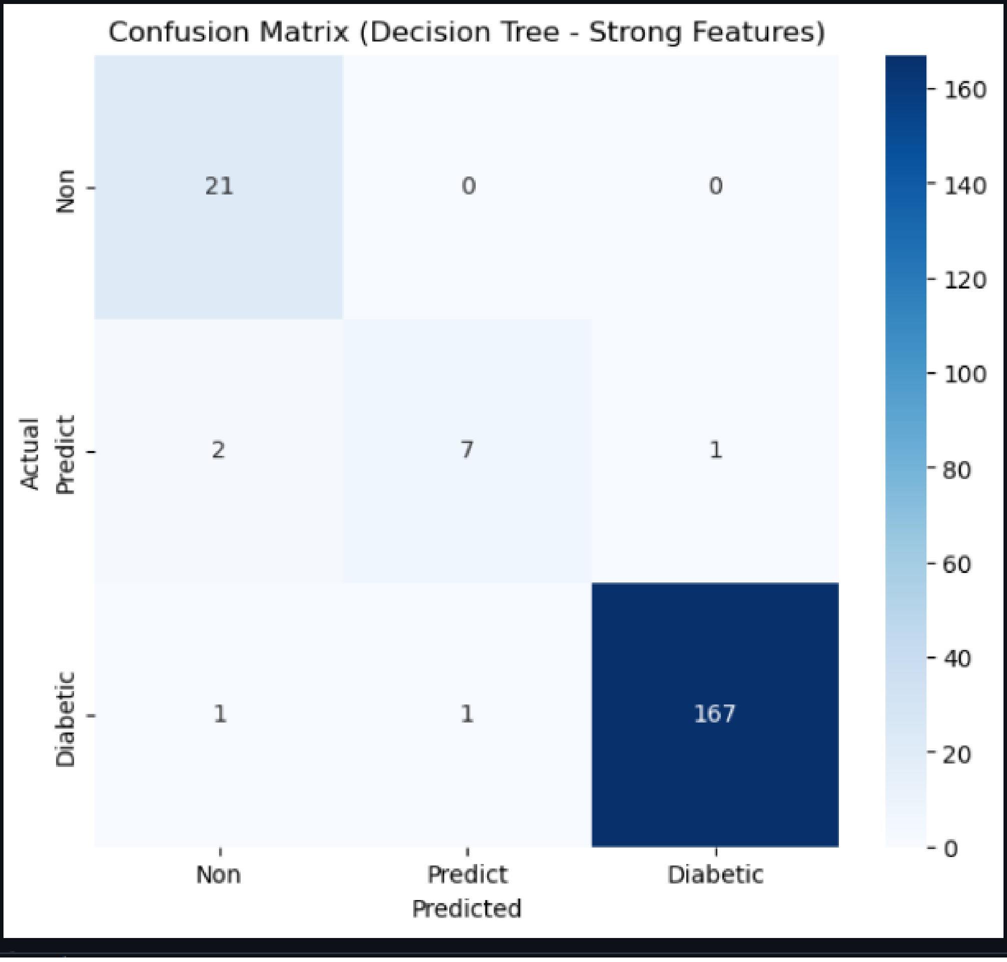


```
Tuning Decision Tree on STRONG Features...
Fitting 5 folds for each of 48 candidates, totalling 240 fits

Results of STRONG Features (Tuned Decision Tree):
Best Mean Cross-Validation Score: 0.9350
Best Parameters: {'criterion': 'entropy', 'max_depth': 9, 'min_samples_leaf': 1}
Final Test Accuracy: 0.975
```

Confusion Matrix:

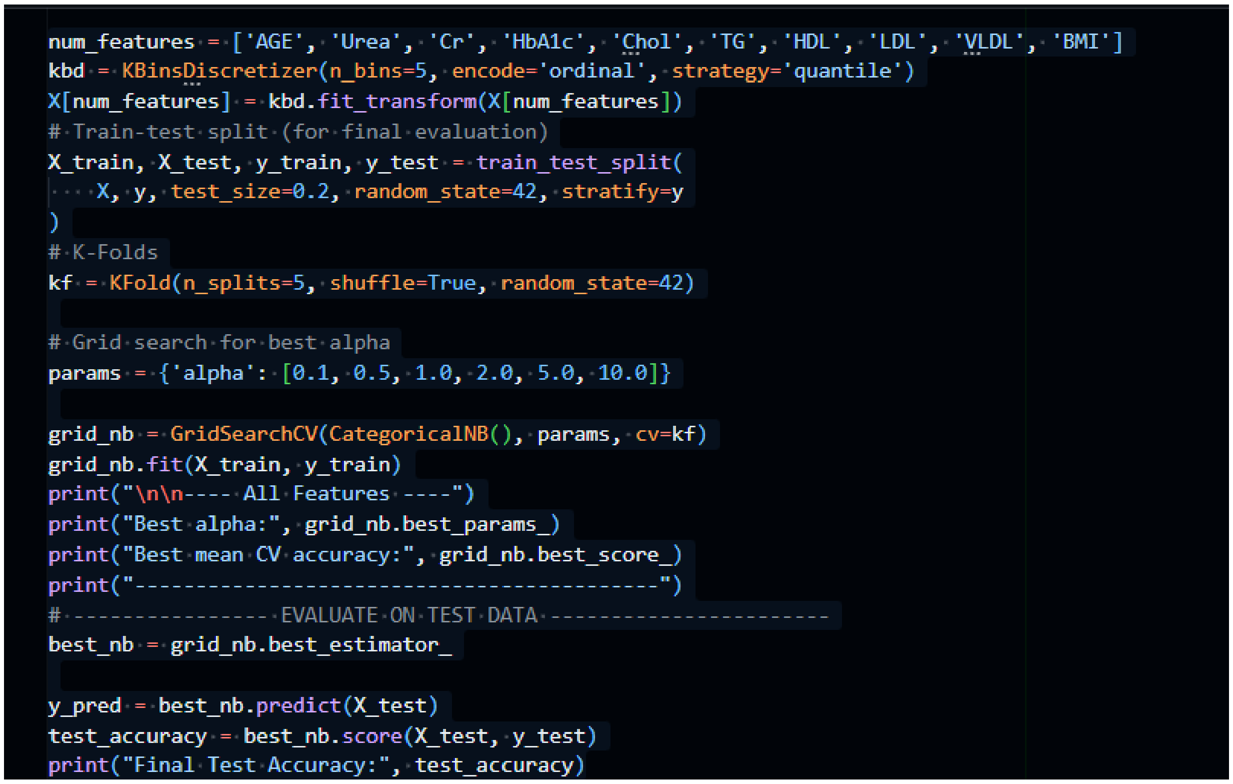


```
 Classification Report:
              precision    recall  f1-score   support

         Non       0.88      1.00      0.93        21
     Predict       0.88      0.70      0.78        10
    Diabetic       0.99      0.99      0.99       169

    accuracy                           0.97       200
   macro avg       0.91      0.90      0.90       200
weighted avg       0.98      0.97      0.97       200
```



3.2 Naive Bayes

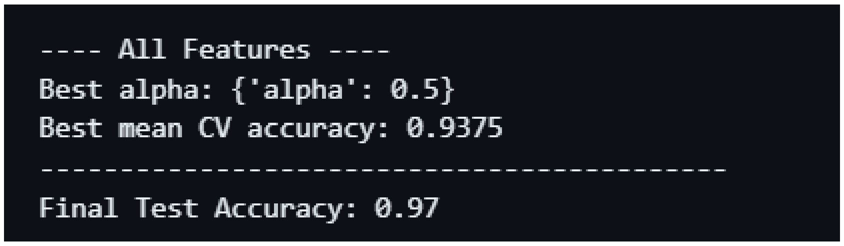
- Used **CategoricalNB** after discretization:



```
num_features = ['AGE', 'Urea', 'Cr', 'HbA1c', 'Chol', 'TG', 'HDL', 'LDL', 'VLDL', 'BMI']
kbd = KBinsDiscretizer(n_bins=5, encode='ordinal', strategy='quantile')
X[num_features] = kbd.fit_transform(X[num_features])
# Train-test split (for final evaluation)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
# K-Folds
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Grid search for best alpha
params = {'alpha': [0.1, 0.5, 1.0, 2.0, 5.0, 10.0]}

grid_nb = GridSearchCV(CategoricalNB(), params, cv=kf)
grid_nb.fit(X_train, y_train)
print("\n\n---- All Features ----")
print("Best alpha:", grid_nb.best_params_)
print("Best mean CV accuracy:", grid_nb.best_score_)
print("--------------------------------------------")
# ---------------- EVALUATE ON TEST DATA ----------------------
best_nb = grid_nb.best_estimator_

y_pred = best_nb.predict(X_test)
test_accuracy = best_nb.score(X_test, y_test)
print("Final Test Accuracy:", test_accuracy)
```

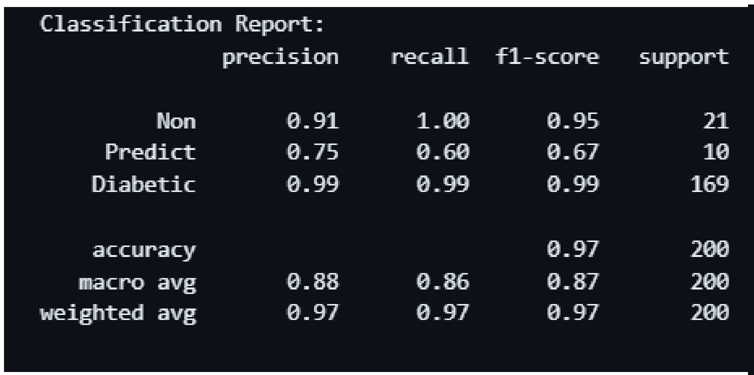
- Evaluation:

For All Features:

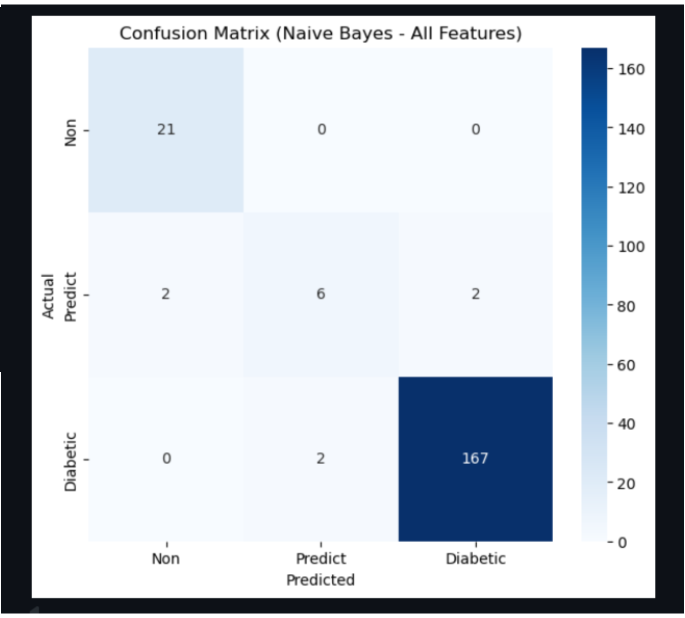


```
---- All Features ----
Best alpha: {'alpha': 0.5}
Best mean CV accuracy: 0.9375
--------------------------------------------
Final Test Accuracy: 0.97
```

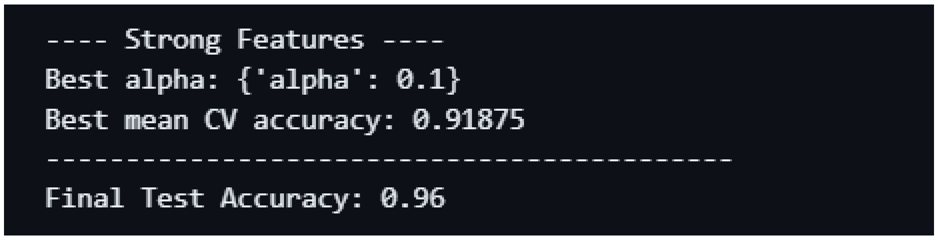
Confusion Matrix:



```
Classification Report:
              precision    recall  f1-score   support

         Non       0.91      1.00      0.95        21
     Predict       0.75      0.60      0.67        10
    Diabetic       0.99      0.99      0.99       169

    accuracy                           0.97       200
   macro avg       0.88      0.86      0.87       200
weighted avg       0.97      0.97      0.97       200
```

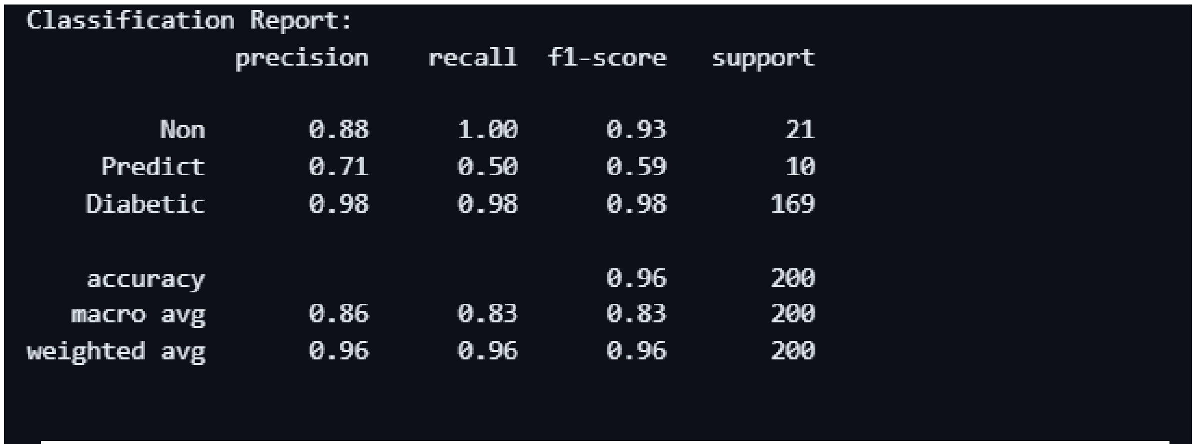


For Strong Features:

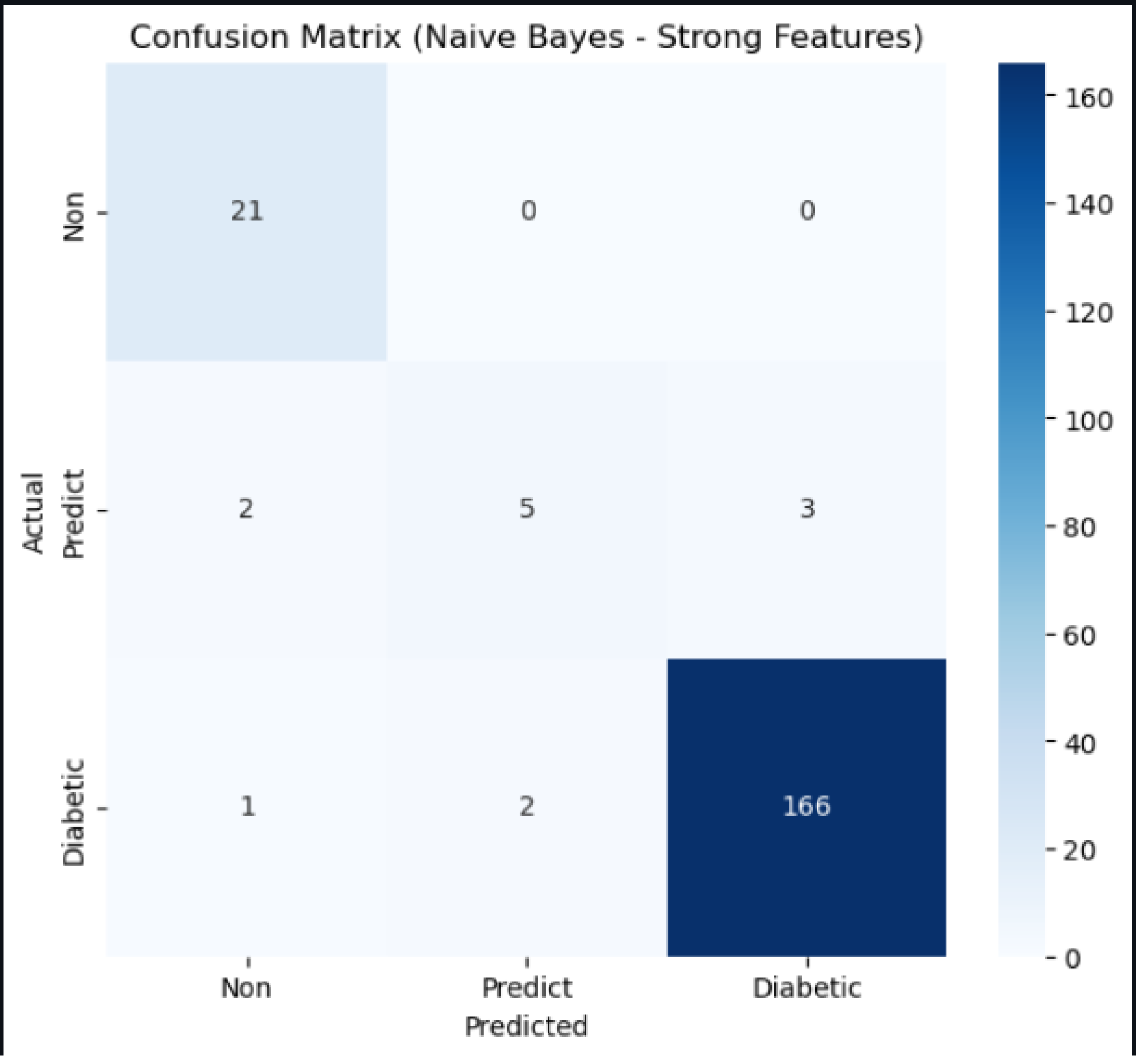


```
---- Strong Features ----
Best alpha: {'alpha': 0.1}
Best mean CV accuracy: 0.91875
------------------------------------------
Final Test Accuracy: 0.96
```

Confusion Matrix:

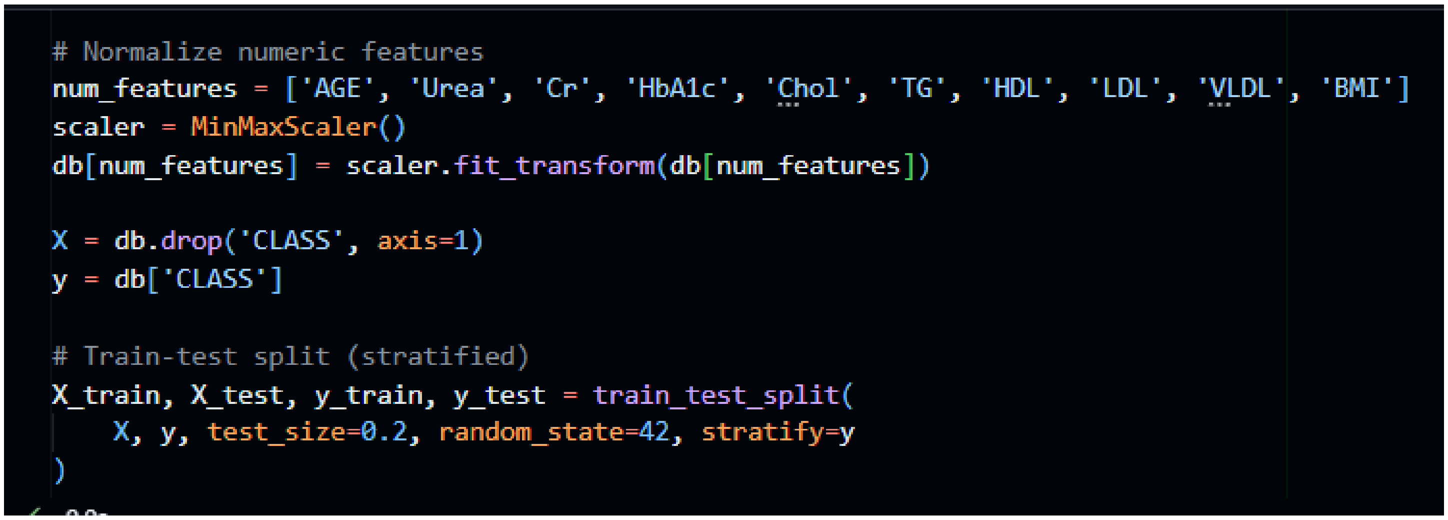


```
Classification Report:
              precision    recall  f1-score   support

         Non       0.88      1.00      0.93        21
     Predict       0.71      0.50      0.59        10
    Diabetic       0.98      0.98      0.98       169

    accuracy                           0.96       200
   macro avg       0.86      0.83      0.83       200
weighted avg       0.96      0.96      0.96       200
```



3.3 K-Nearest Neighbors (KNN)

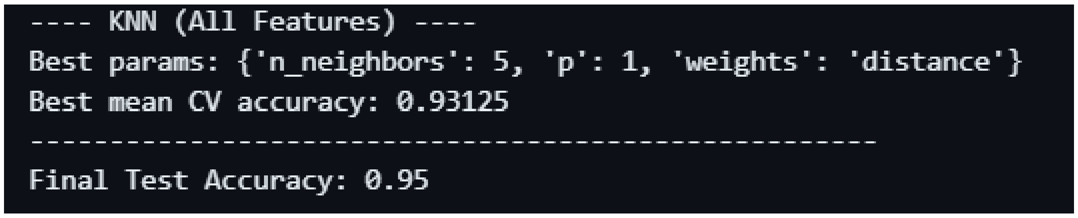
- Applied **MinMax scaling**:



```python
# Normalize numeric features
num_features = ['AGE', 'Urea', 'Cr', 'HbA1c', 'Chol', 'TG', 'HDL', 'LDL', 'VLDL', 'BMI']
scaler = MinMaxScaler()
db[num_features] = scaler.fit_transform(db[num_features])

X = db.drop('CLASS', axis=1)
y = db['CLASS']

# Train-test split (stratified)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

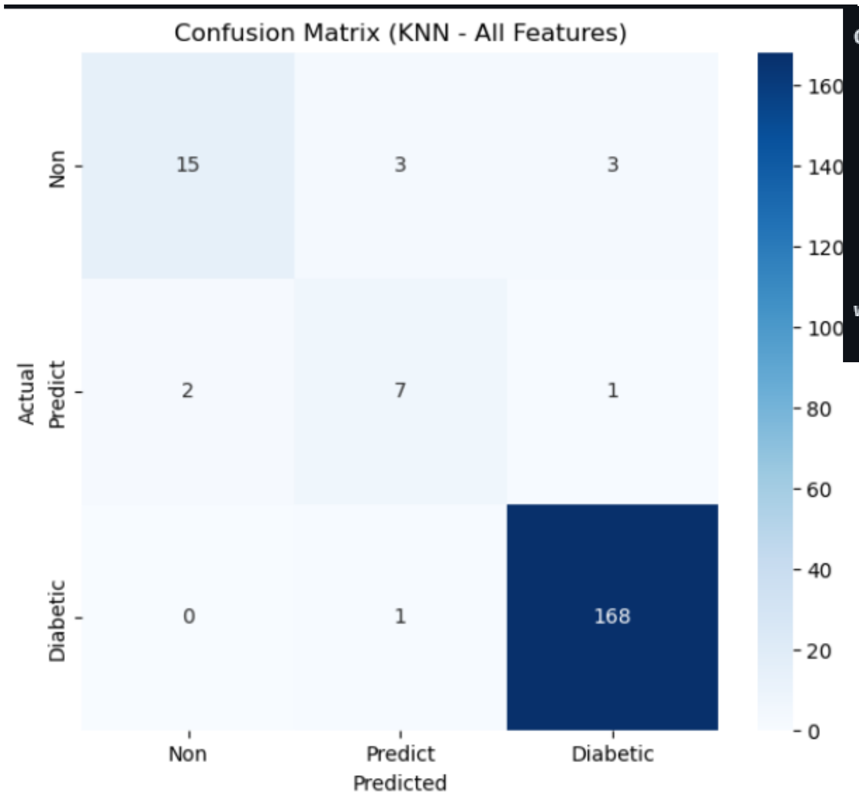
- Evaluation:

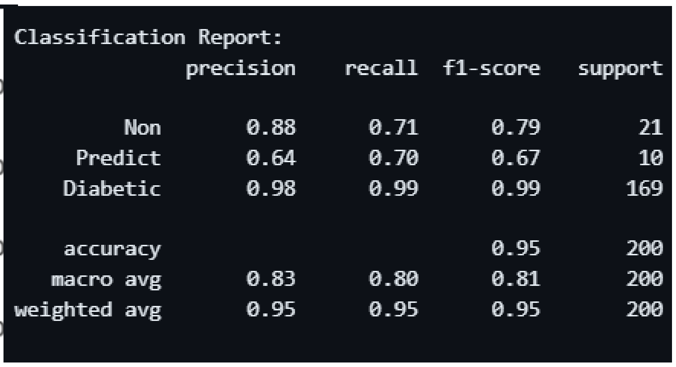
  For All Features:



```
---- KNN (All Features) ----
Best params: {'n_neighbors': 5, 'p': 1, 'weights': 'distance'}
Best mean CV accuracy: 0.93125
-------------------------------------------------------
Final Test Accuracy: 0.95
```

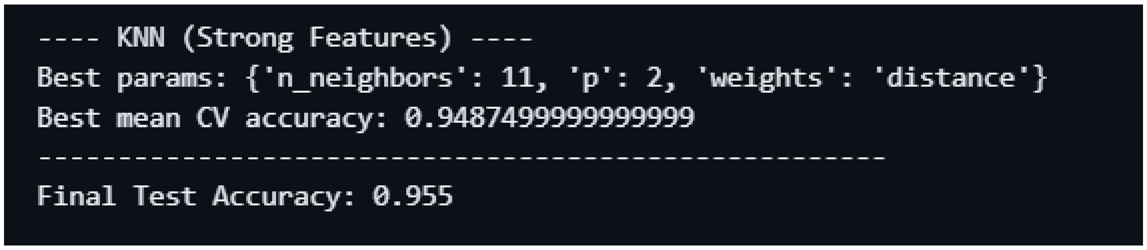
  Confusion Matrix:





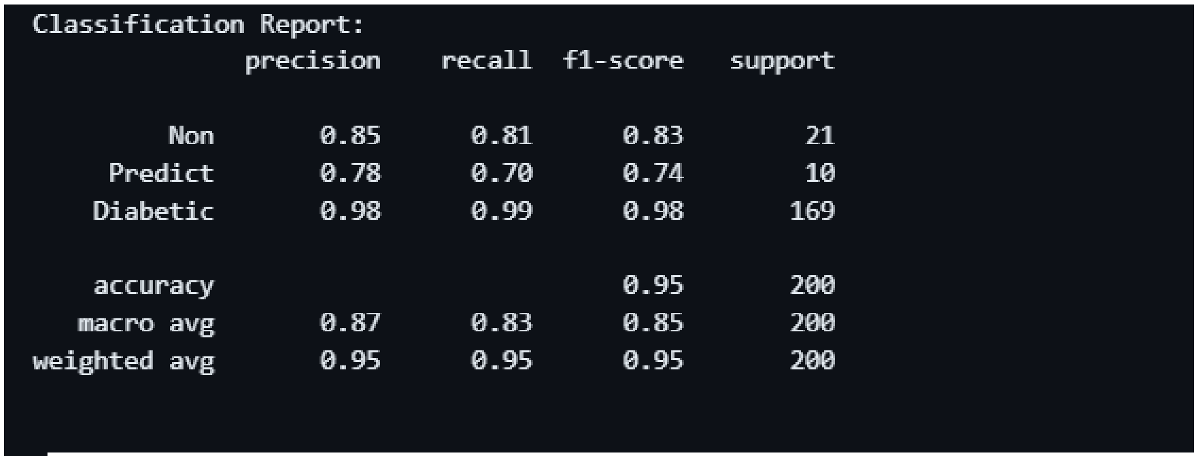
```
Classification Report:
              precision    recall  f1-score   support

         Non       0.88      0.71      0.79        21
     Predict       0.64      0.70      0.67        10
    Diabetic       0.98      0.99      0.99       169

    accuracy                           0.95       200
   macro avg       0.83      0.80      0.81       200
weighted avg       0.95      0.95      0.95       200
```

For Strong Features:

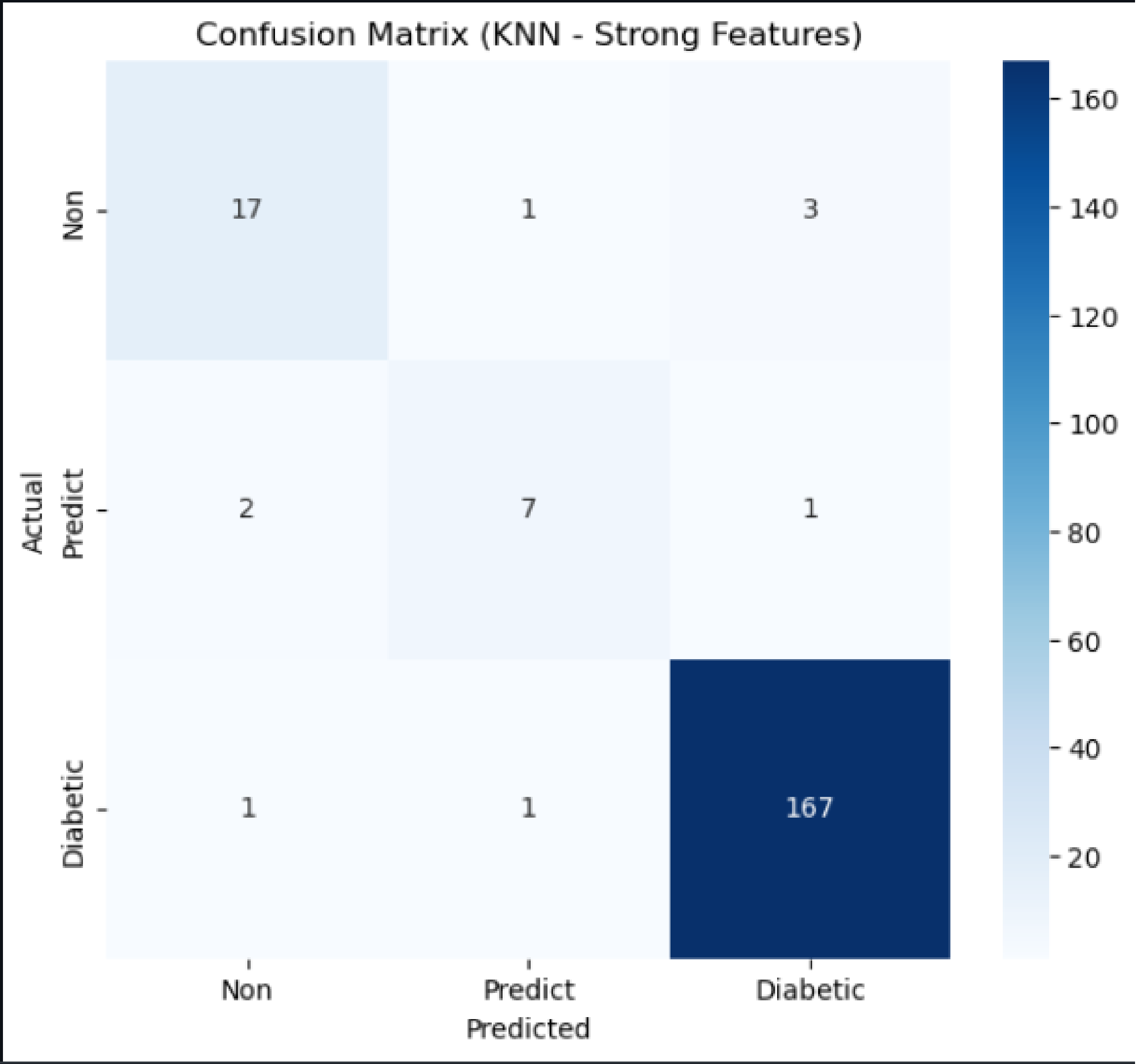


```
---- KNN (Strong Features) ----
Best params: {'n_neighbors': 11, 'p': 2, 'weights': 'distance'}
Best mean CV accuracy: 0.9487499999999999
-----------------------------------------------------
Final Test Accuracy: 0.955
```

Confusion Matrix:

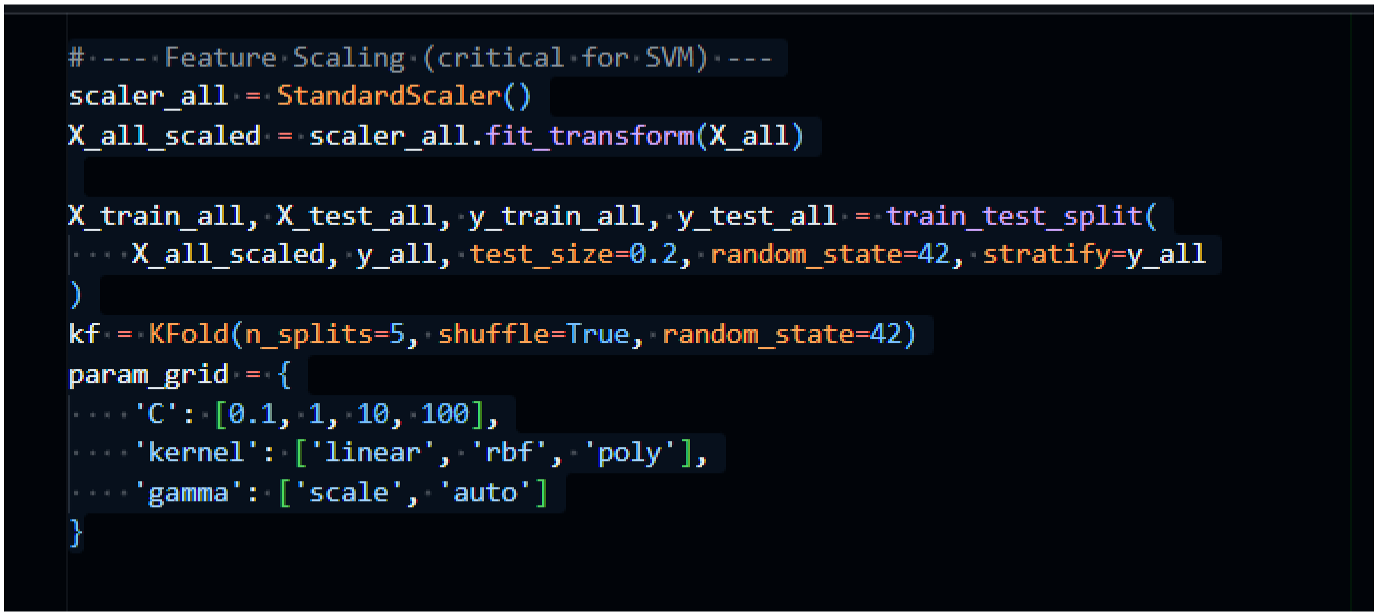


```
Classification Report:
              precision    recall  f1-score   support

         Non       0.85      0.81      0.83        21
     Predict       0.78      0.70      0.74        10
    Diabetic       0.98      0.99      0.98       169

    accuracy                           0.95       200
   macro avg       0.87      0.83      0.85       200
weighted avg       0.95      0.95      0.95       200
```



3.4 Support Vector Machine (SVM)

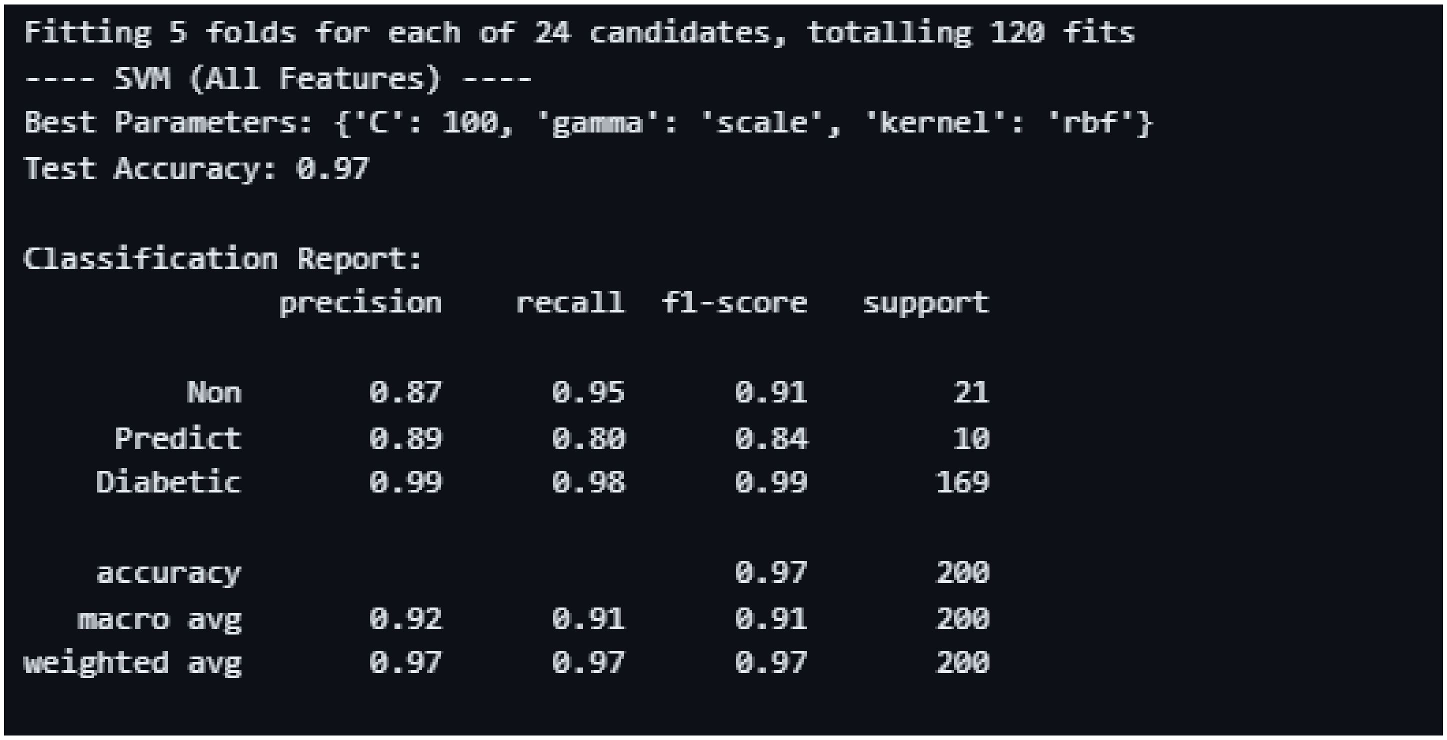
- Feature scaling is critical:



```
# --- Feature Scaling (critical for SVM) ---
scaler_all = StandardScaler()
X_all_scaled = scaler_all.fit_transform(X_all)

X_train_all, X_test_all, y_train_all, y_test_all = train_test_split(
    X_all_scaled, y_all, test_size=0.2, random_state=42, stratify=y_all
)
kf = KFold(n_splits=5, shuffle=True, random_state=42)
param_grid = {
    'C': [0.1, 1, 10, 100],
    'kernel': ['linear', 'rbf', 'poly'],
    'gamma': ['scale', 'auto']
}
```

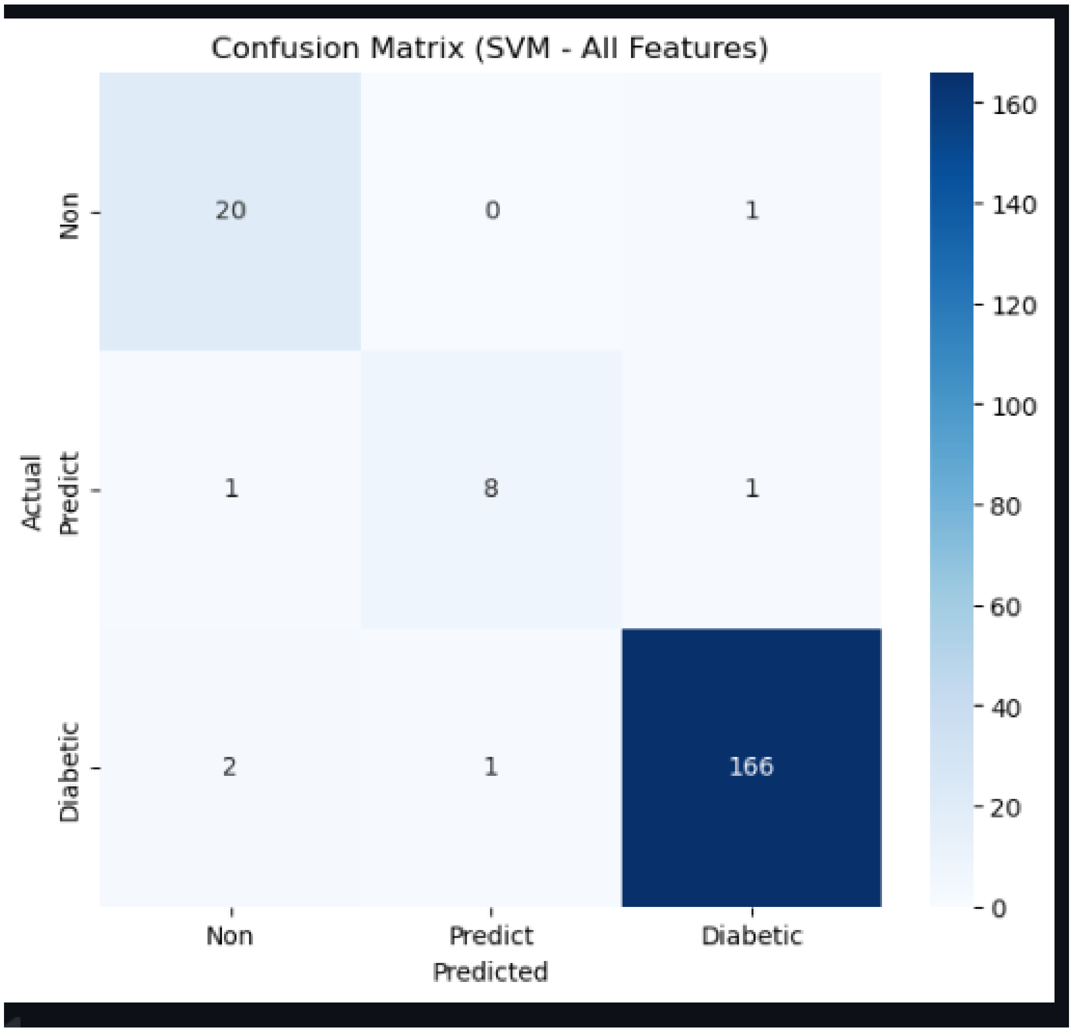
- Evaluation:

  For All Features:

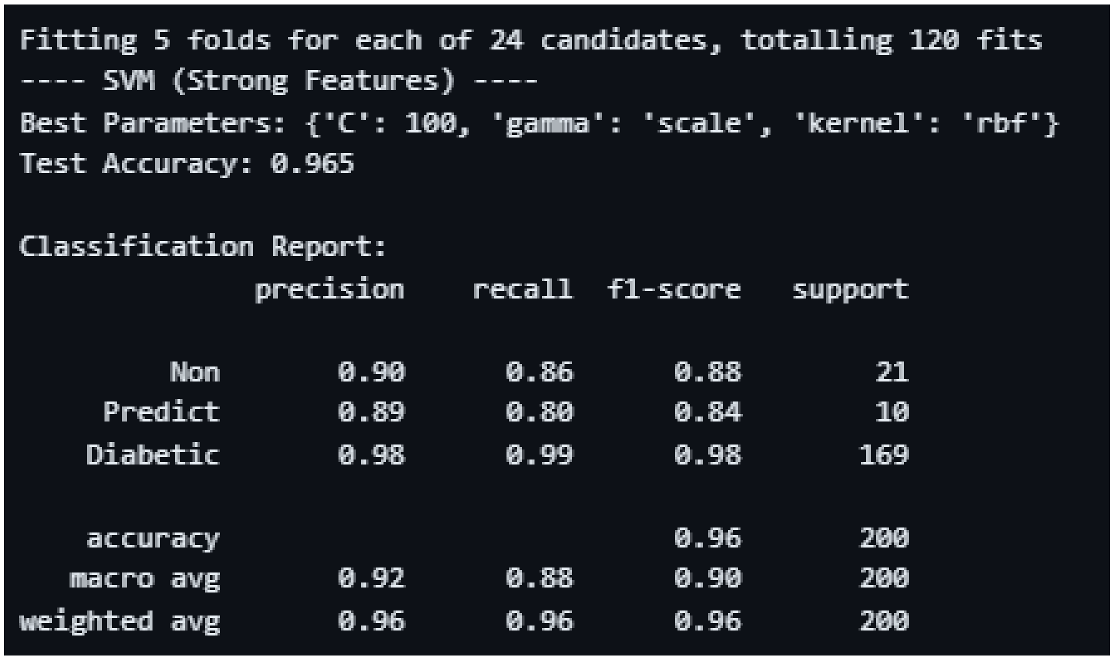


```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
---- SVM (All Features) ----
Best Parameters: {'C': 100, 'gamma': 'scale', 'kernel': 'rbf'}
Test Accuracy: 0.97

Classification Report:
              precision    recall  f1-score   support

         Non       0.87      0.95      0.91        21
     Predict       0.89      0.80      0.84        10
    Diabetic       0.99      0.98      0.99       169

    accuracy                           0.97       200
   macro avg       0.92      0.91      0.91       200
weighted avg       0.97      0.97      0.97       200
```

Confusion Matrix:

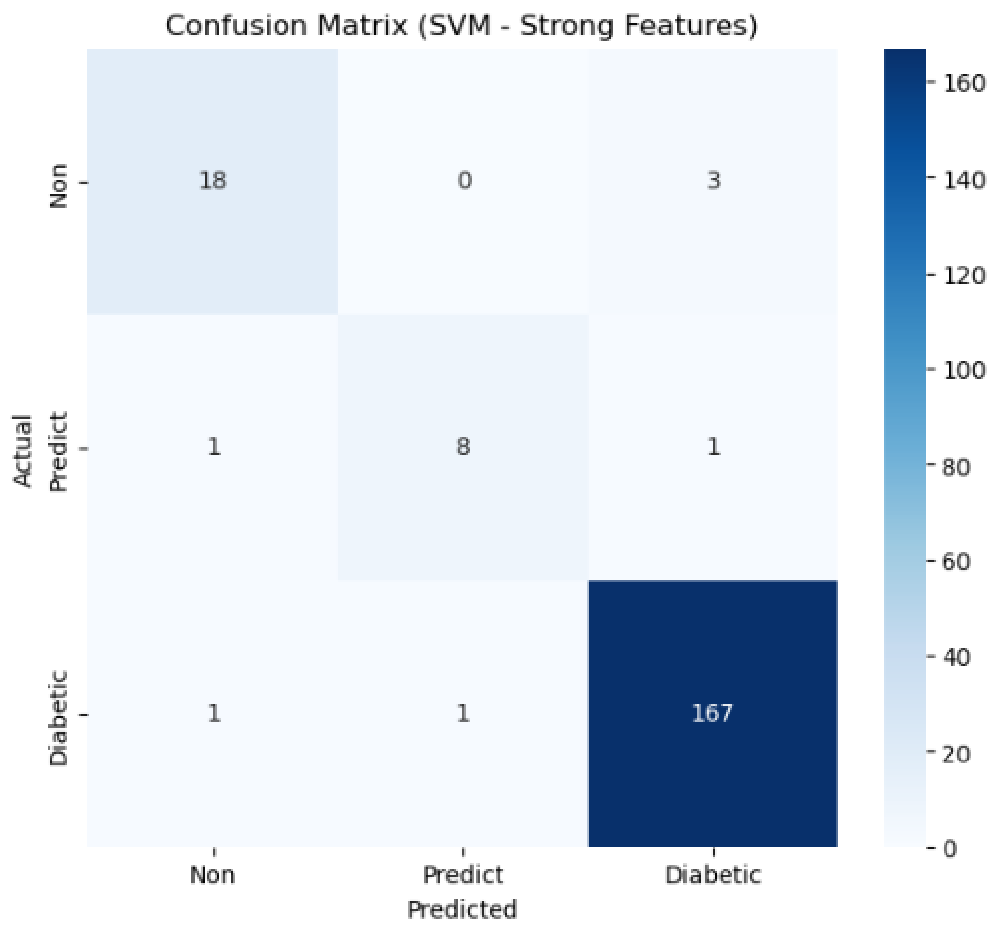


For Strong Features:



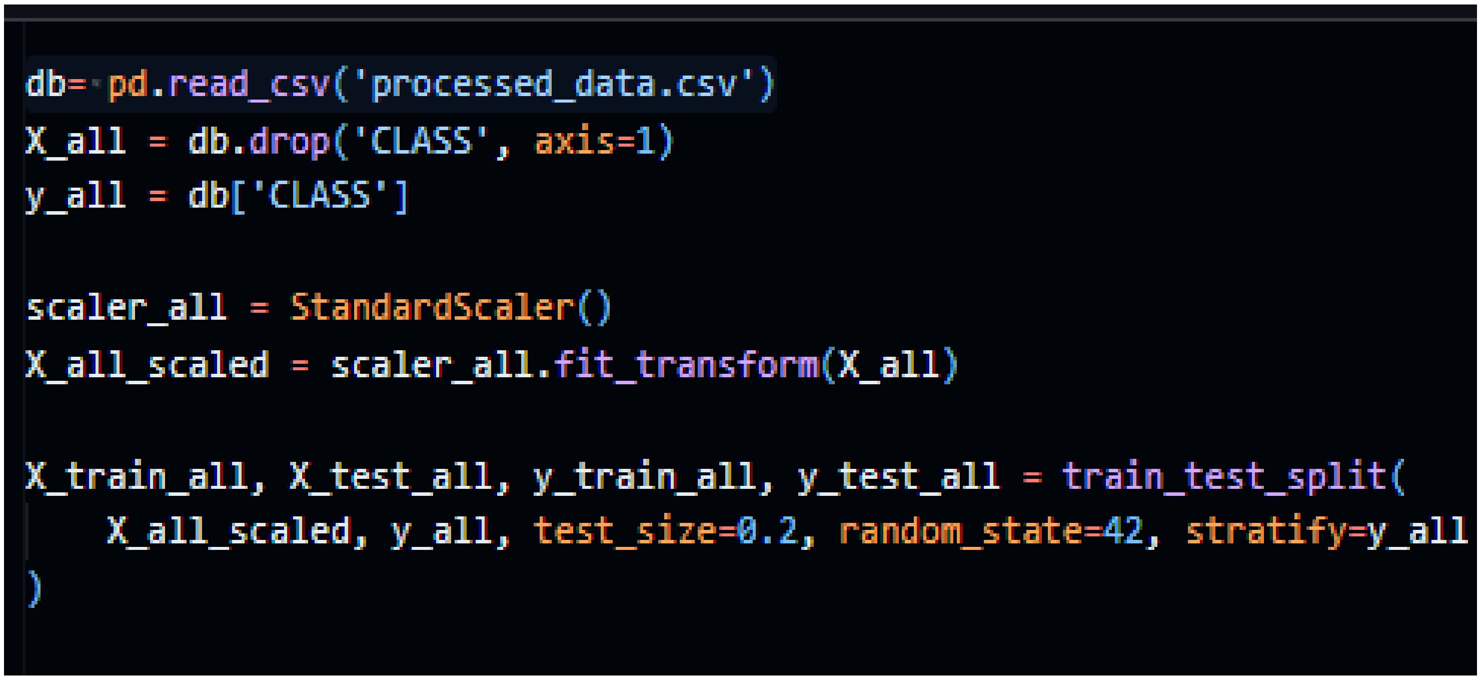
```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
---- SVM (Strong Features) ----
Best Parameters: {'C': 100, 'gamma': 'scale', 'kernel': 'rbf'}
Test Accuracy: 0.965

Classification Report:
              precision    recall  f1-score   support

         Non       0.90      0.86      0.88        21
     Predict       0.89      0.80      0.84        10
    Diabetic       0.98      0.99      0.98       169

    accuracy                           0.96       200
   macro avg       0.92      0.88      0.90       200
weighted avg       0.96      0.96      0.96       200
```

Confusion Matrix:



3.5 Logistic Regression

- Standard scaling applied:



```
db= pd.read_csv('processed_data.csv')
X_all = db.drop('CLASS', axis=1)
y_all = db['CLASS']

scaler_all = StandardScaler()
X_all_scaled = scaler_all.fit_transform(X_all)

X_train_all, X_test_all, y_train_all, y_test_all = train_test_split(
    X_all_scaled, y_all, test_size=0.2, random_state=42, stratify=y_all
)
```

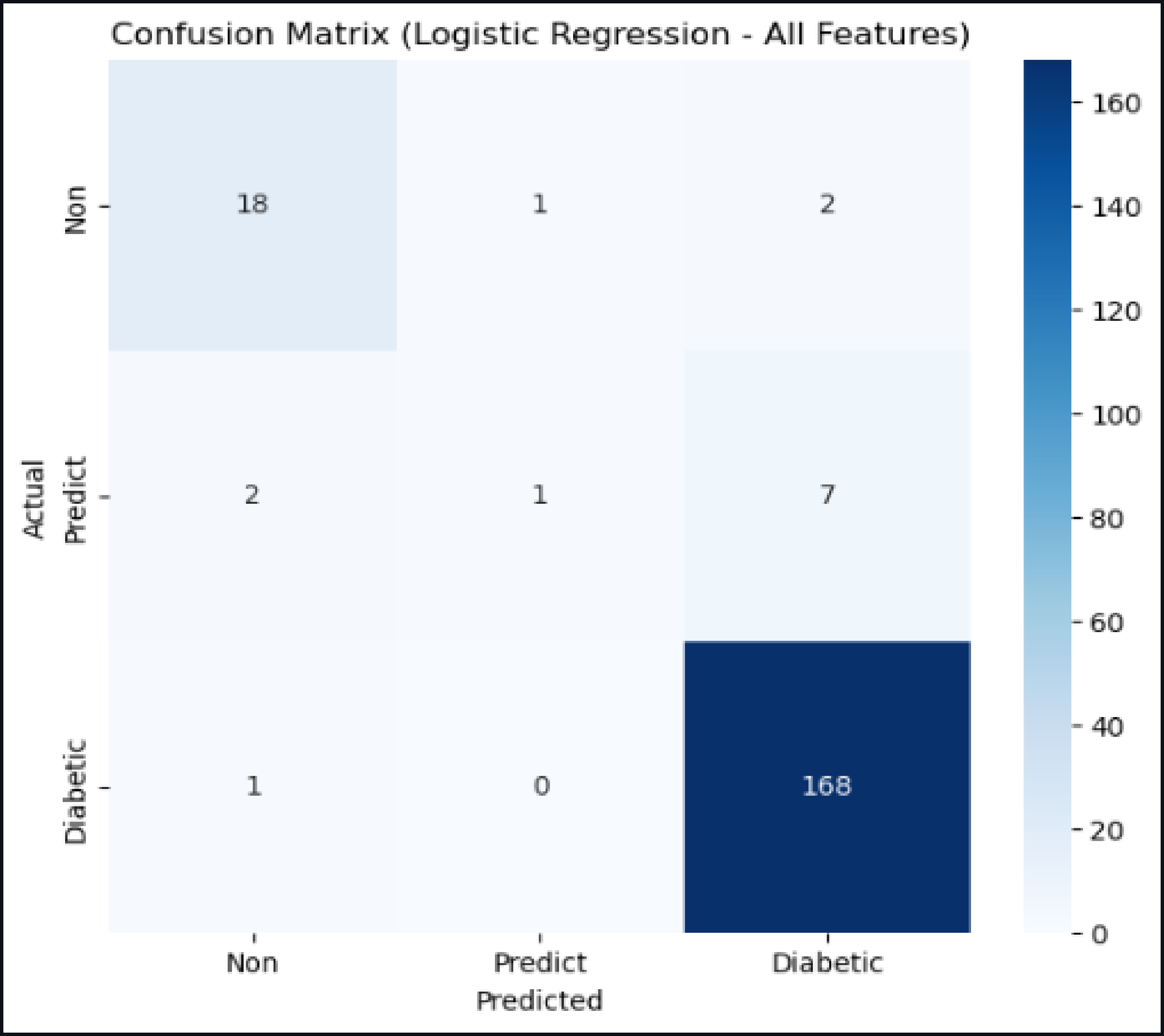
- Evaluation:

For All Features:

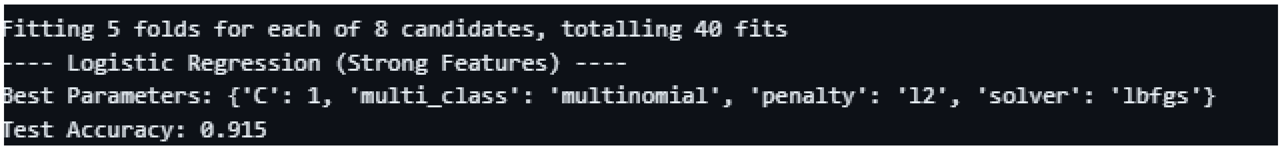
```
Fitting 5 folds for each of 8 candidates, totalling 40 fits
---- Logistic Regression (All Features) ----
Best Parameters: {'C': 0.1, 'multi_class': 'multinomial', 'penalty': 'l2', 'solver': 'lbfgs'}
Test Accuracy: 0.935
```

Confusion Matrix:

```
 Classification Report:
               precision    recall  f1-score   support

         Non       0.86      0.86      0.86        21
     Predict       0.50      0.10      0.17        10
    Diabetic       0.95      0.99      0.97       169

    accuracy                           0.94       200
   macro avg       0.77      0.65      0.66       200
weighted avg       0.92      0.94      0.92       200
```

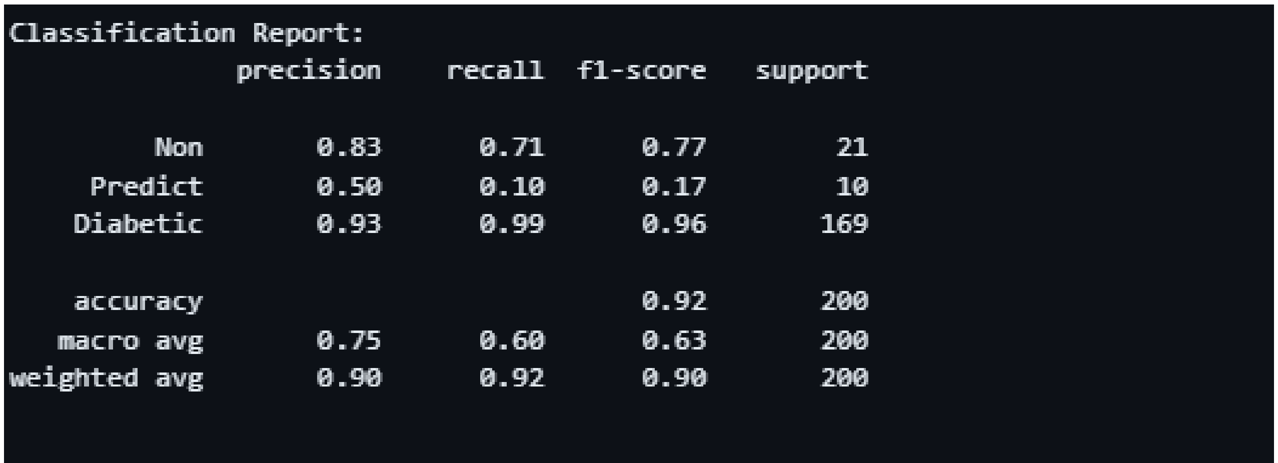


For Strong Features:

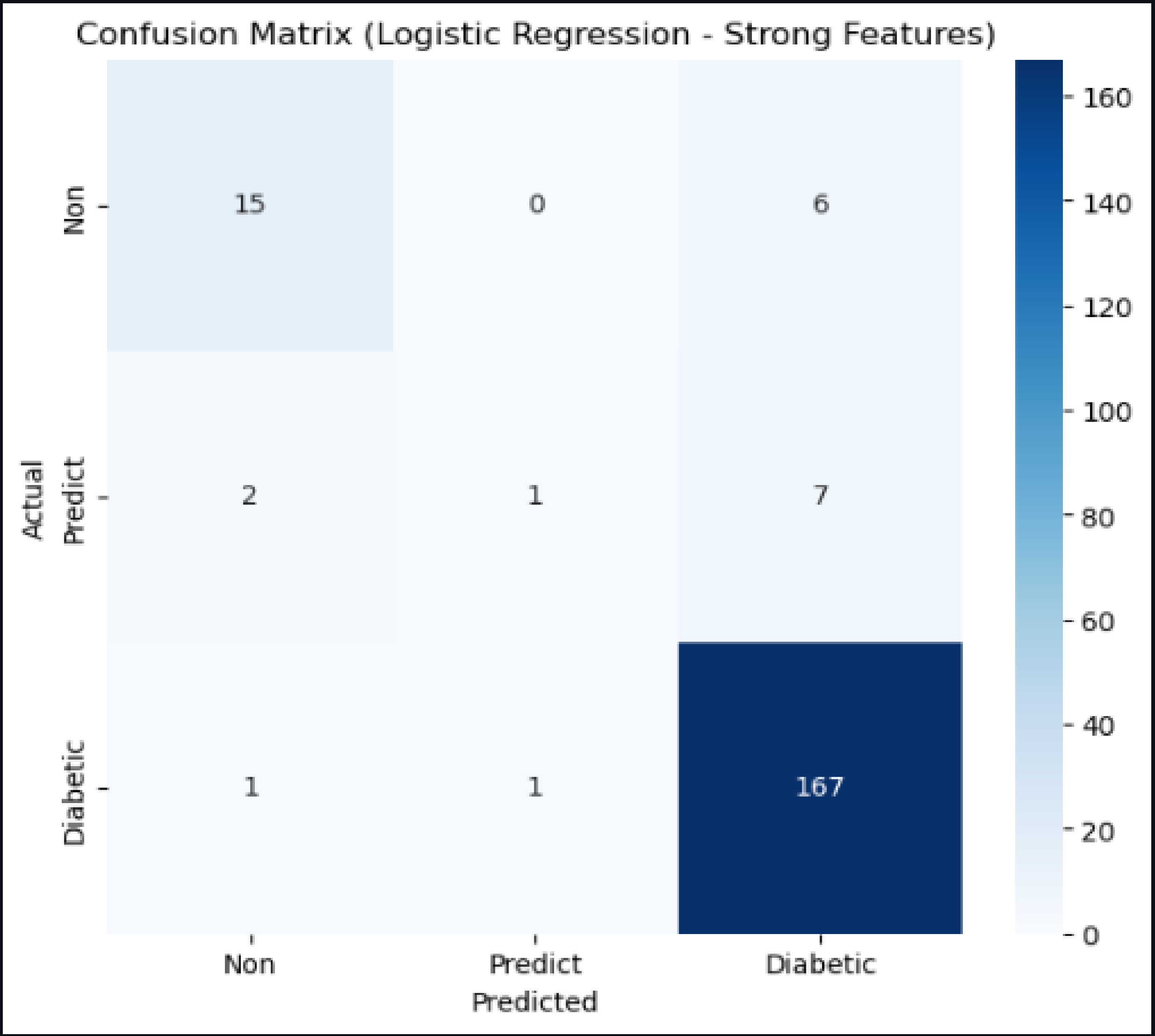


```
Fitting 5 folds for each of 8 candidates, totalling 40 fits
---- Logistic Regression (Strong Features) ----
Best Parameters: {'C': 1, 'multi_class': 'multinomial', 'penalty': 'l2', 'solver': 'lbfgs'}
Test Accuracy: 0.915
```

Confusion Matrix:

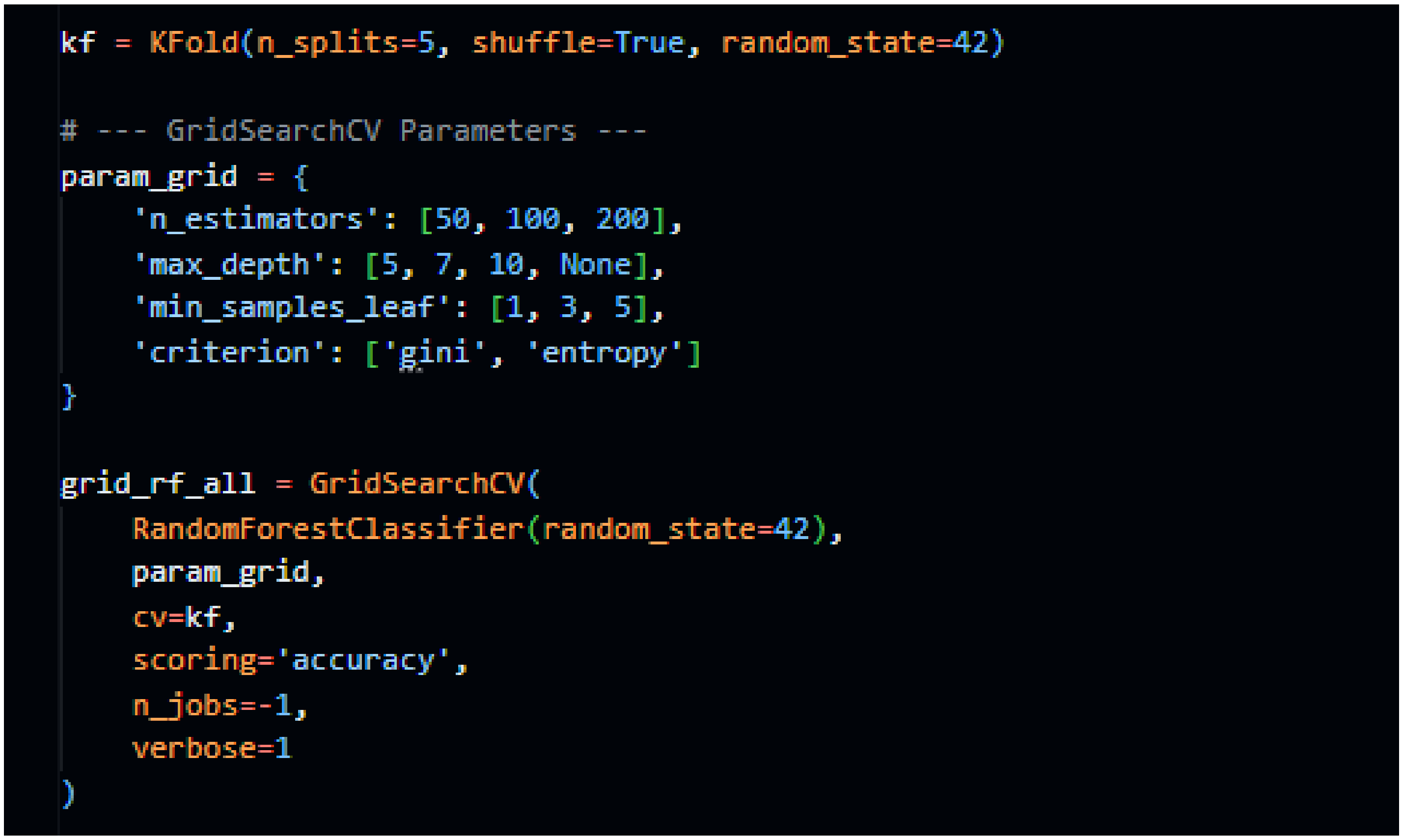


```
Classification Report:
              precision    recall  f1-score   support

         Non       0.83      0.71      0.77        21
     Predict       0.50      0.10      0.17        10
    Diabetic       0.93      0.99      0.96       169

    accuracy                           0.92       200
   macro avg       0.75      0.60      0.63       200
weighted avg       0.90      0.92      0.90       200
```



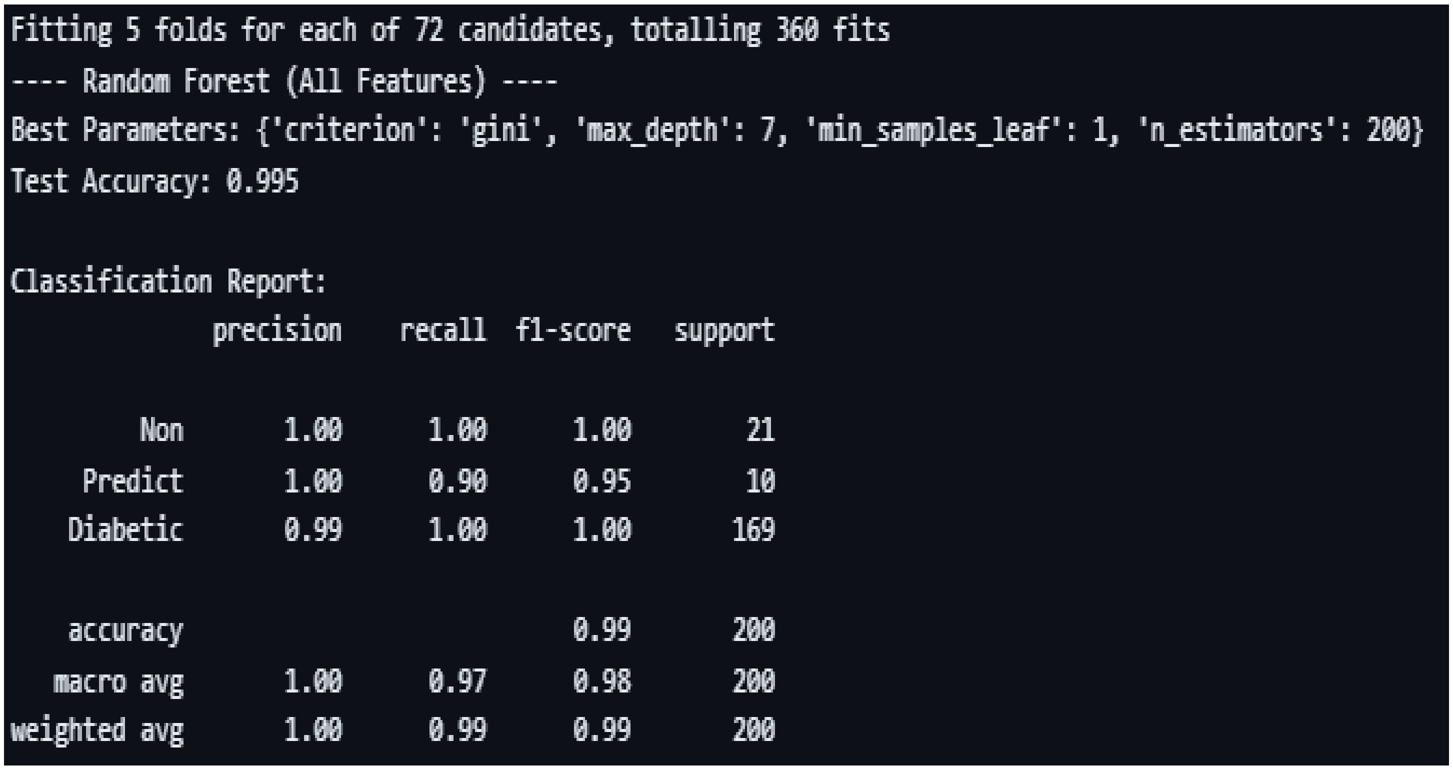
3.6 Random Forest

- Ensemble method; robust to non-linear features:



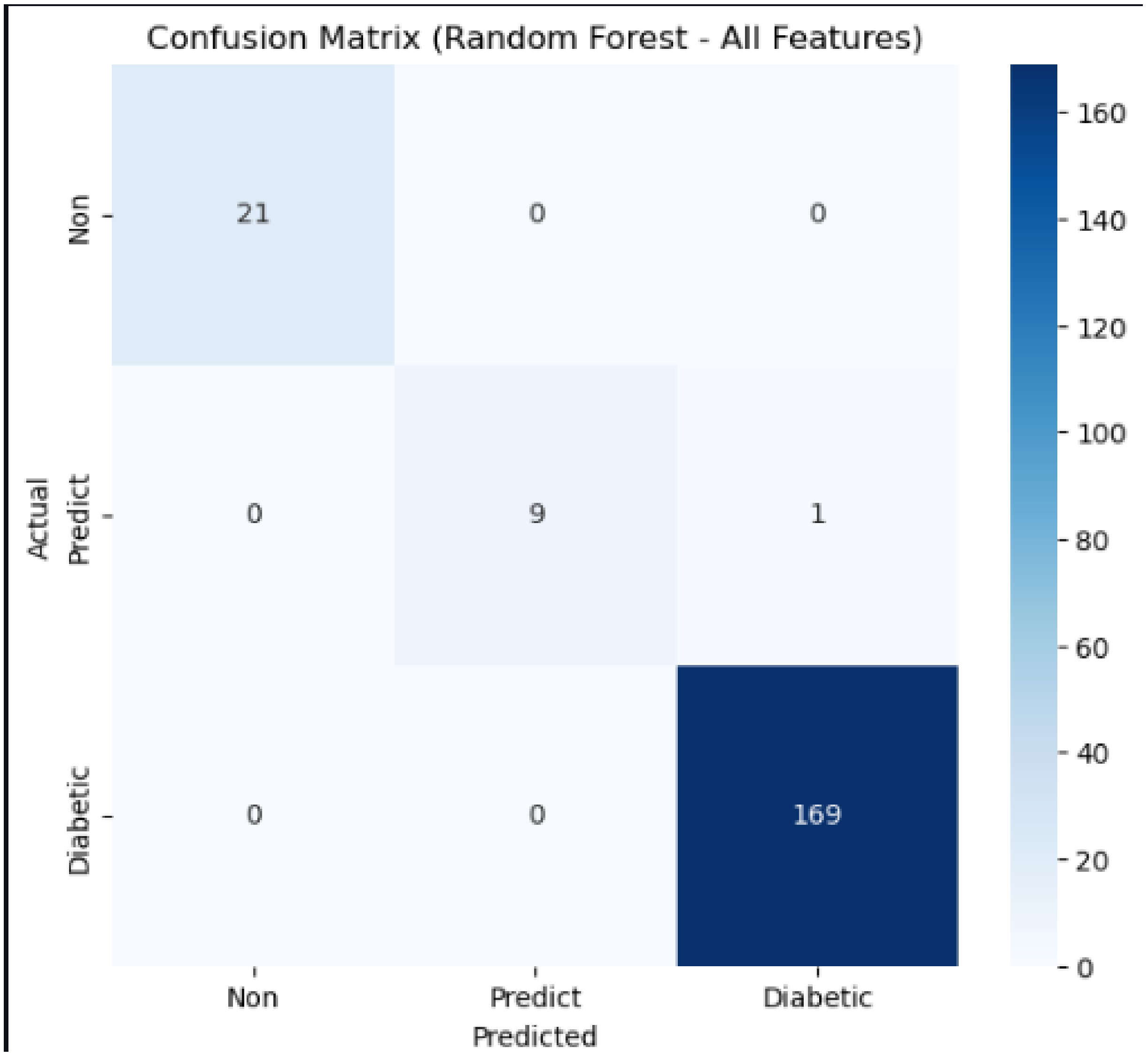
```
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# --- GridSearchCV Parameters ---
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [5, 7, 10, None],
    'min_samples_leaf': [1, 3, 5],
    'criterion': ['gini', 'entropy']
}

grid_rf_all = GridSearchCV(
    RandomForestClassifier(random_state=42),
    param_grid,
    cv=kf,
    scoring='accuracy',
    n_jobs=-1,
    verbose=1
)
```

- Evaluation:
  For All Features:

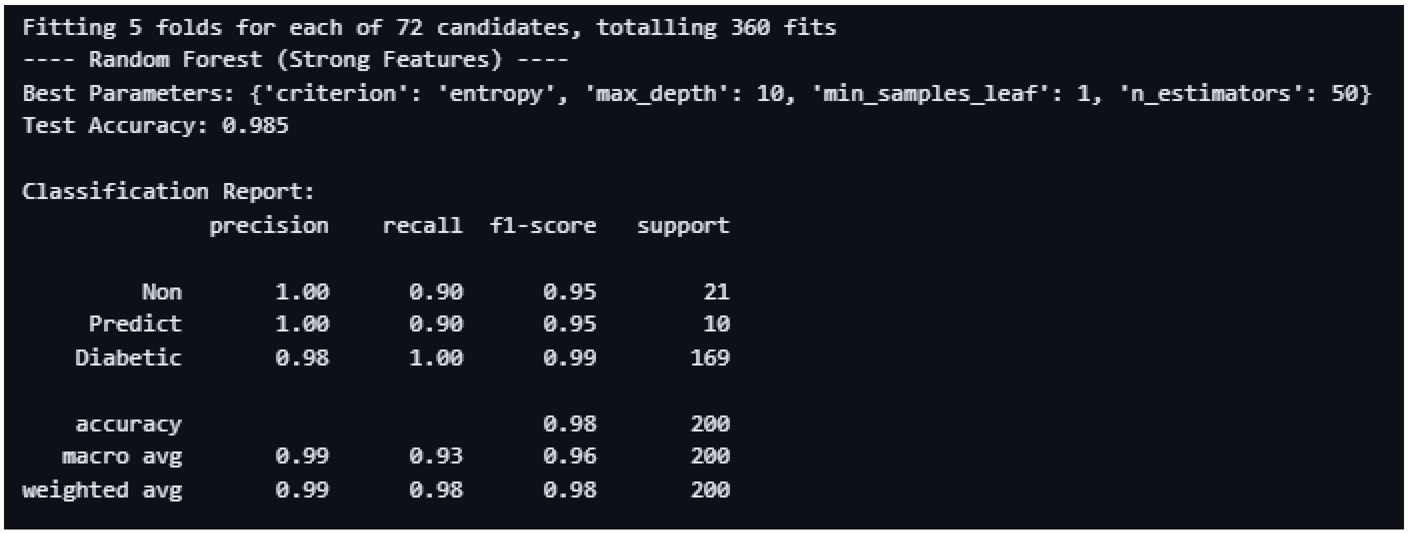


```
Fitting 5 folds for each of 72 candidates, totalling 360 fits
---- Random Forest (All Features) ----
Best Parameters: {'criterion': 'gini', 'max_depth': 7, 'min_samples_leaf': 1, 'n_estimators': 200}
Test Accuracy: 0.995

Classification Report:
              precision    recall  f1-score   support

         Non       1.00      1.00      1.00        21
     Predict       1.00      0.90      0.95        10
    Diabetic       0.99      1.00      1.00       169

    accuracy                           0.99       200
   macro avg       1.00      0.97      0.98       200
weighted avg       1.00      0.99      0.99       200
```

Confusion Matrix:

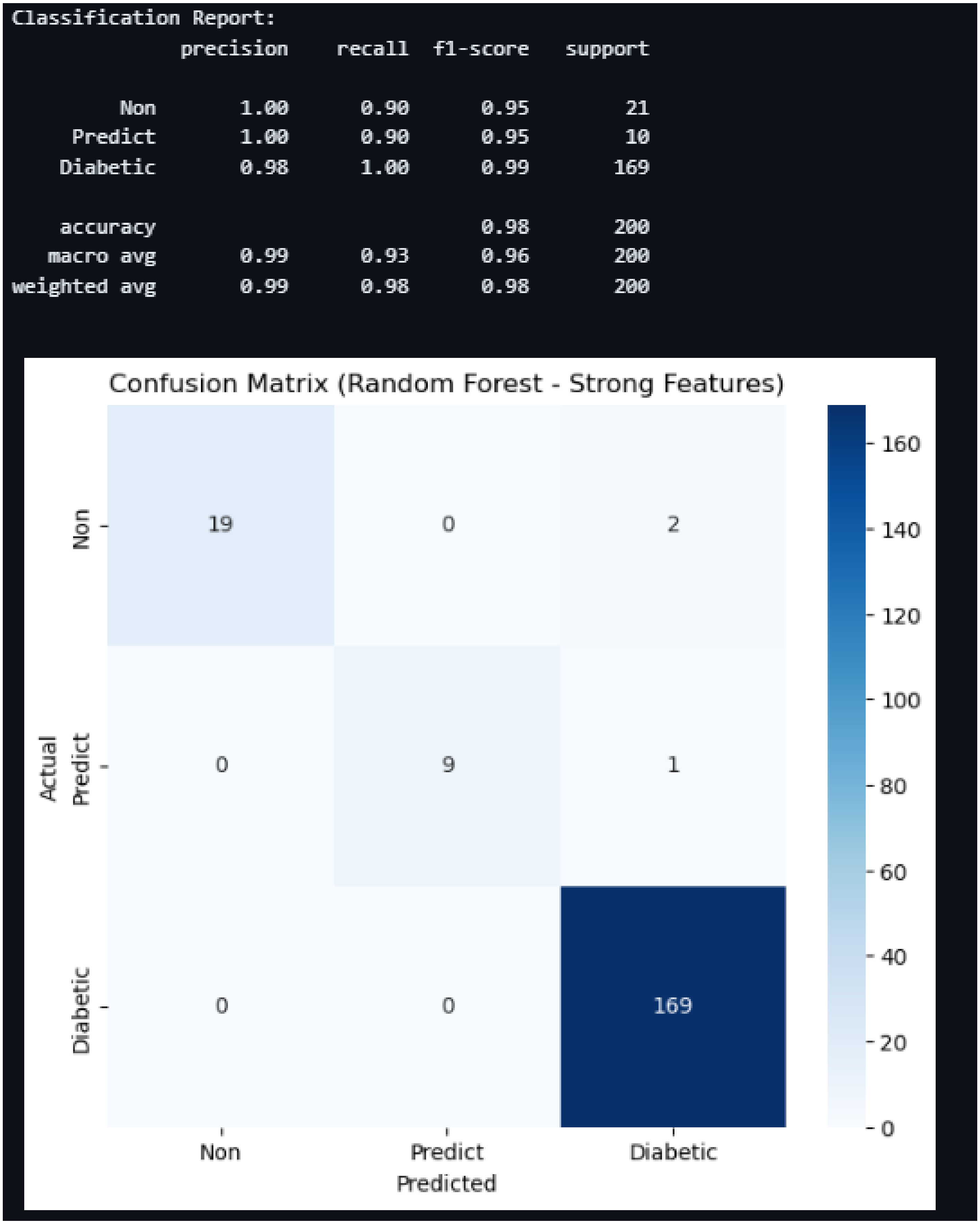


For Strong Features:



```
Fitting 5 folds for each of 72 candidates, totalling 360 fits
---- Random Forest (Strong Features) ----
Best Parameters: {'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 1, 'n_estimators': 50}
Test Accuracy: 0.985

Classification Report:
              precision    recall  f1-score   support

         Non       1.00      0.90      0.95        21
     Predict       1.00      0.90      0.95        10
    Diabetic       0.98      1.00      0.99       169

    accuracy                           0.98       200
   macro avg       0.99      0.93      0.96       200
weighted avg       0.99      0.98      0.98       200
```

Confusion Matrix:



```
Classification Report:
              precision    recall  f1-score   support

         Non       1.00      0.90      0.95        21
     Predict       1.00      0.90      0.95        10
    Diabetic       0.98      1.00      0.99       169

    accuracy                           0.98       200
   macro avg       0.99      0.93      0.96       200
weighted avg       0.99      0.98      0.98       200
```

4.Used Libraries

Model	Python Library / Module
Decision Tree	`from sklearn.tree import DecisionTreeClassifier`
Naive Bayes (Categorical)	`from sklearn.naive_bayes import CategoricalNB`
K-Nearest Neighbors (KNN)	`from sklearn.neighbors import KNeighborsClassifier`
Support Vector Machine (SVM)	`from sklearn.svm import SVC`
Logistic Regression	`from sklearn.linear_model import LogisticRegression`
Random Forest	`from sklearn.ensemble import RandomForestClassifier`

5.Comparison of Models

Model	Features	Test Accuracy	Precision	Recall	F1-score
Decision Tree	All	0.97	0.88	0.86	0.87
Decision Tree	Strong	0.975	0.91	0.90	0.90
Naive Bayes	All	0.97	0.88	0.86	0.87
Naive Bayes	Strong	0.96	0.86	0.83	0.83
KNN	All	0.95	0.83	0.80	0.81
KNN	Strong	0.955	0.87	0.83	0.85
SVM	All	0.97	0.92	0.91	0.91
SVM	Strong	0.965	0.92	0.88	0.90
Logistic Regression	All	0.935	0.77	0.65	0.66
Logistic Regression	Strong	0.915	0.75	0.60	0.63
Random Forest	All	0.995	1.0	0.97	0.98
Random Forest	Strong	0.985	0.99	0.93	0.96

6.Conclusion

Based on the results of the different machine learning models applied to the dataset, several important observations can be made:

Random Forest outperforms all other models.

- Using all features, it achieved the **highest test accuracy (0.995)**, **perfect precision (1.0)**, and very high recall (0.97) and F1-score (0.98).
- Even with the strongest features only, Random Forest still performed excellently, showing its robustness.

SVM and Decision Tree models also perform well.

- SVM with all features achieved **high precision (0.92)** and recall (0.91), making it a strong choice for balanced performance.
- Decision Tree improves slightly when using only the strongest features, achieving a **test accuracy of 0.975**.

Naive Bayes and KNN show moderate performance.

- Naive Bayes maintains consistent results with all features, but its performance drops with strong features.
- KNN shows a slight improvement when using strong features, but its overall performance is lower than Random Forest and SVM.

Logistic Regression performs the worst among all models.

- Both with all features and strongest features, it has lower precision, recall, and F1-score, indicating it may not capture the complexity of the data as well as other models.

The **Random Forest model** is the most accurate and reliable for this dataset, particularly when using all features. It consistently balances precision, recall, and F1-score better than other models. **SVM and Decision Tree** can also be considered as good alternatives, while **Logistic Regression** is the least suitable for this task.