Celebal Assignment Week – 6

1. Objective:

The primary goal of this project is to train multiple machine learning models on a medical dataset (tumor diagnosis), evaluate their performance using common classification metrics, and select the best-performing model. Additionally, hyperparameter tuning techniques such as GridSearchCV and RandomizedSearchCV are used to optimize the model parameters for better results.

2. Dataset Overview:

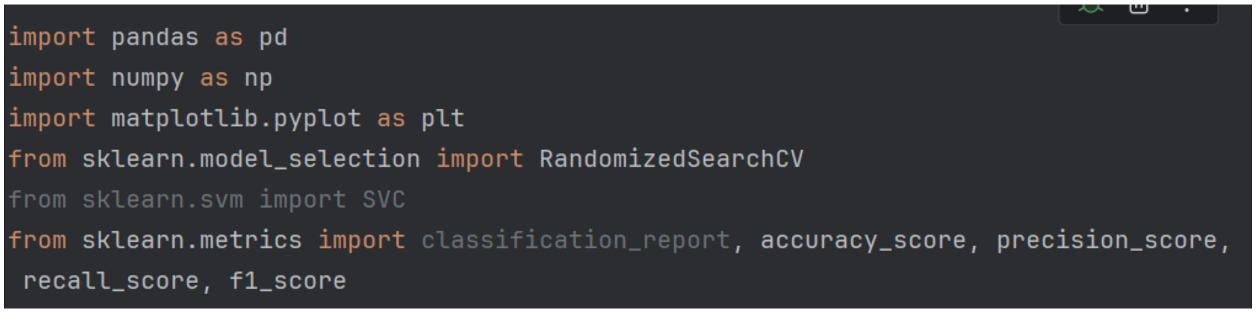
The dataset used for this project consists of various medical measurements related to tumor cells, including radius, texture, smoothness, and more.

The target variable is diagnosis, where:

- M (Malignant) → Cancerous
- B (Benign) → Non-cancerous

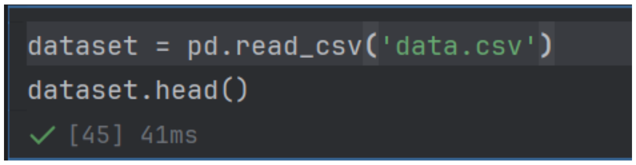
3. Preprocessing Steps:

- **Importing Libraries**: Essential Python libraries like pandas, numpy, matplotlib, sklearn were imported.



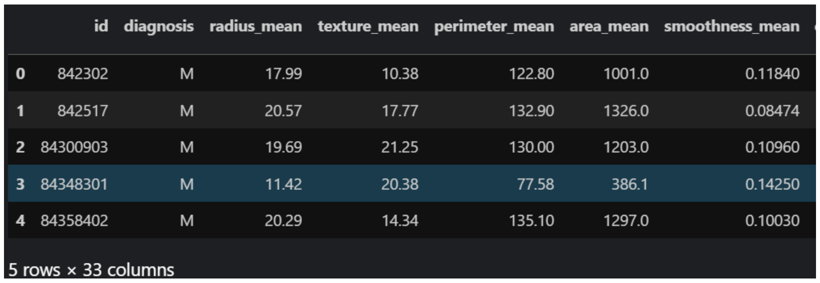
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import RandomizedSearchCV
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score, precision_score,
 recall_score, f1_score
```

- **Loading Data**: The dataset was read using pd.read_csv().



```python
dataset = pd.read_csv('data.csv')
dataset.head()
```

```
✓ [45] 41ms
```



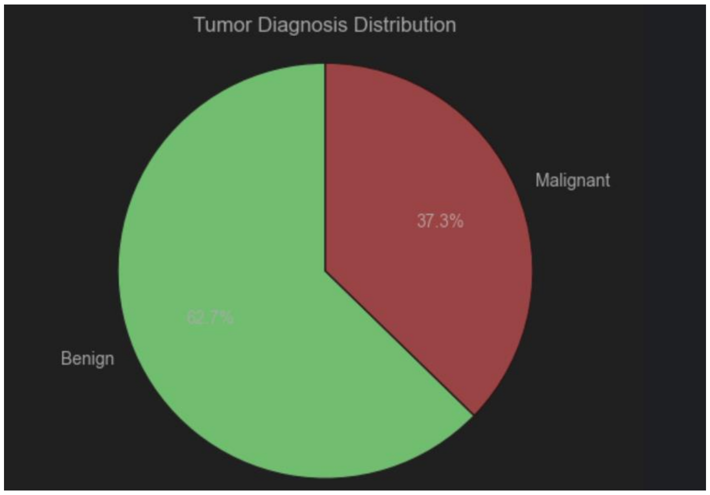
	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840
1	842517	M	20.57	17.77	132.90	1326.0	0.08474
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960
3	84348301	M	11.42	20.38	77.58	386.1	0.14250
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030

5 rows × 33 columns

- **Label Encoding**: The categorical diagnosis column was mapped to numerical values (M → 1, B → 0).
- **Feature Selection**: Irrelevant columns like id and unnamed columns were dropped.
- **Splitting Data**: The dataset was split into X (features) and y (labels).

4. Visualization:

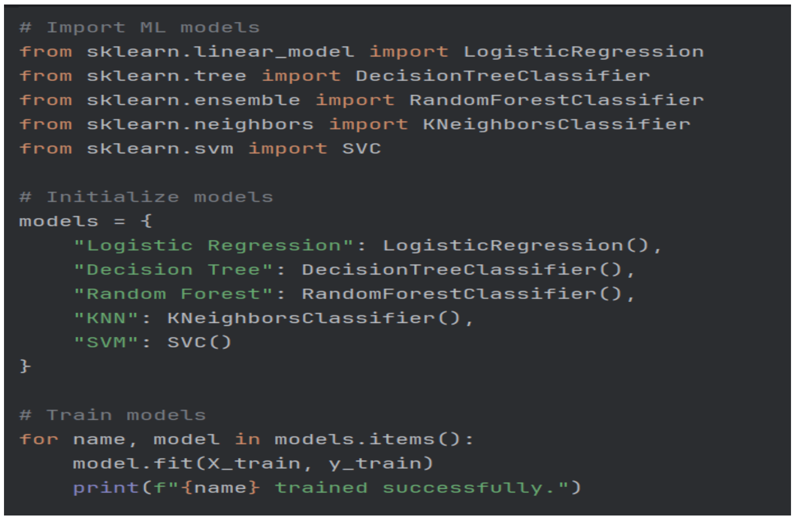
A pie chart was used to visualize the proportion of benign vs malignant tumors. This helps in understanding class imbalance, if any.



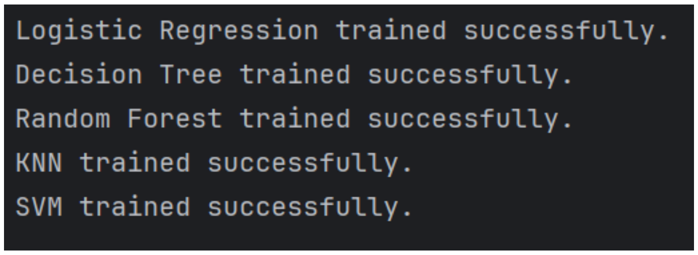
5. Model Training:

The following models were trained:

- Logistic Regression
- Decision Tree Classifier
- Random Forest Classifier
- Support Vector Machine (SVM)
- K-Nearest Neighbors (KNN)



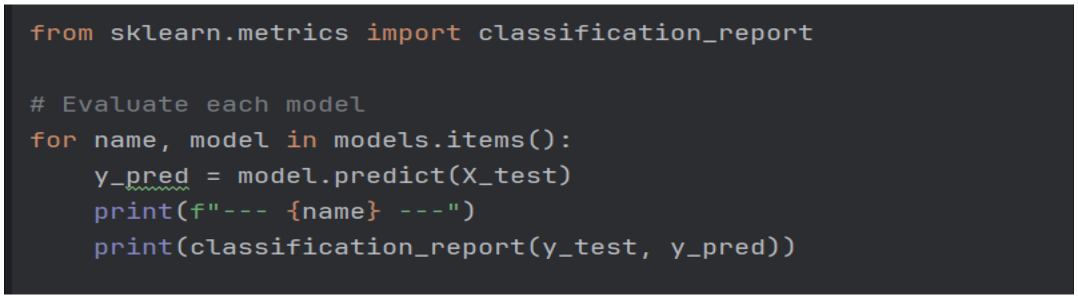
```
# Import ML models
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC

# Initialize models
models = {
    "Logistic Regression": LogisticRegression(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "KNN": KNeighborsClassifier(),
    "SVM": SVC()
}

# Train models
for name, model in models.items():
    model.fit(X_train, y_train)
    print(f"{name} trained successfully.")
```



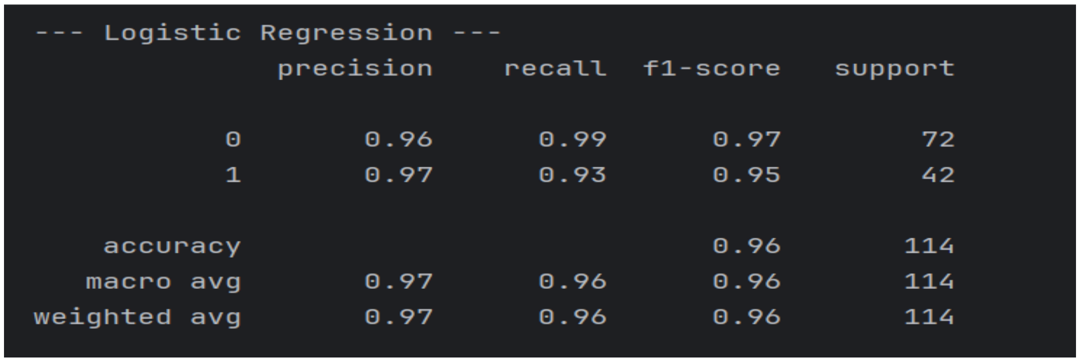
```
Logistic Regression trained successfully.
Decision Tree trained successfully.
Random Forest trained successfully.
KNN trained successfully.
SVM trained successfully.
```

Each model was evaluated on the test set using the following metrics:

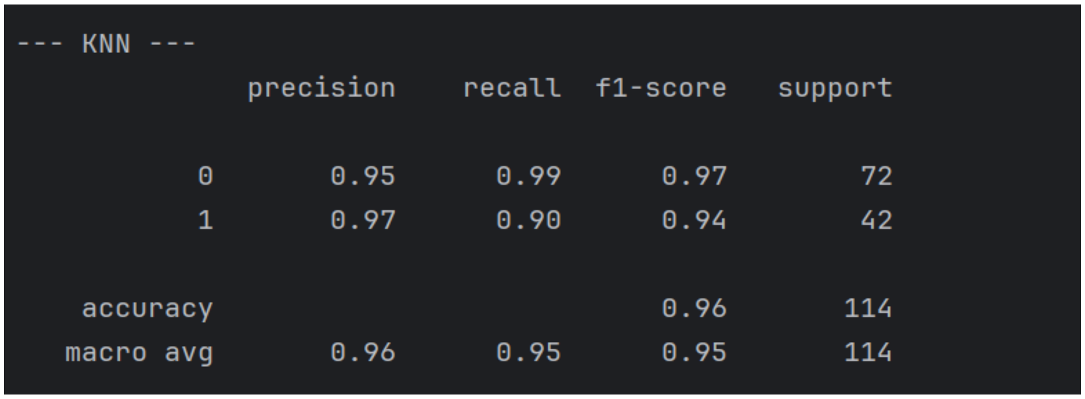
- **Accuracy**: Overall correctness of the model.
- **Precision**: How many of the predicted positives are actually positive.
- **Recall**: How many of the actual positives the model was able to capture.
- **F1-Score**: Harmonic mean of precision and recall.



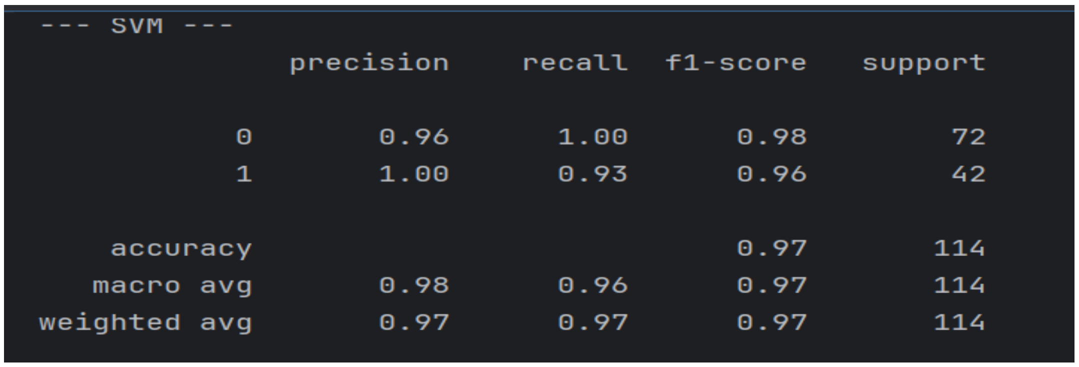
```
from sklearn.metrics import classification_report

# Evaluate each model
for name, model in models.items():
    y_pred = model.predict(X_test)
    print(f"--- {name} ---")
    print(classification_report(y_test, y_pred))
```



```
--- Logistic Regression ---
              precision    recall  f1-score   support

           0       0.96      0.99      0.97        72
           1       0.97      0.93      0.95        42

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114
```



```
--- KNN ---
              precision    recall  f1-score   support

           0       0.95      0.99      0.97        72
           1       0.97      0.90      0.94        42

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
```



```
--- SVM ---
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        72
           1       1.00      0.93      0.96        42

    accuracy                           0.97       114
   macro avg       0.98      0.96      0.97       114
weighted avg       0.97      0.97      0.97       114
```

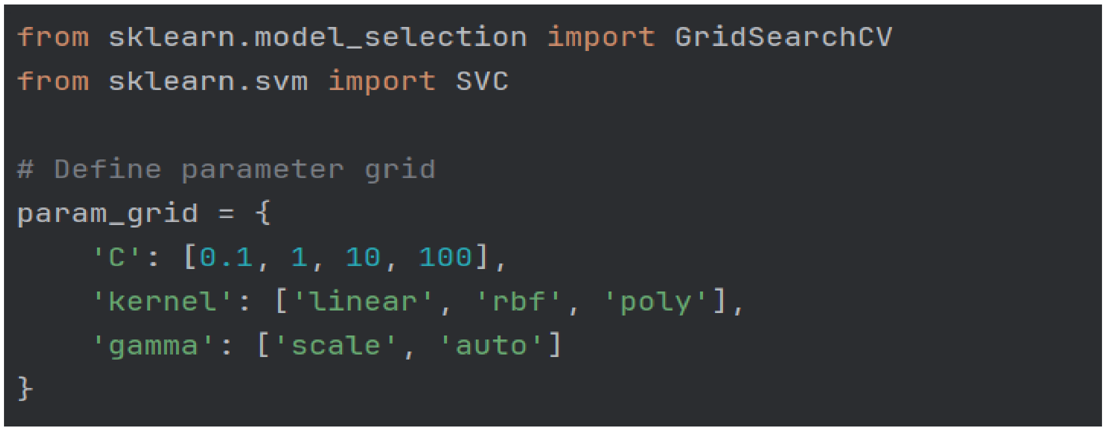
Scores of different models are predicted

6. Hyperparameter Tuning:

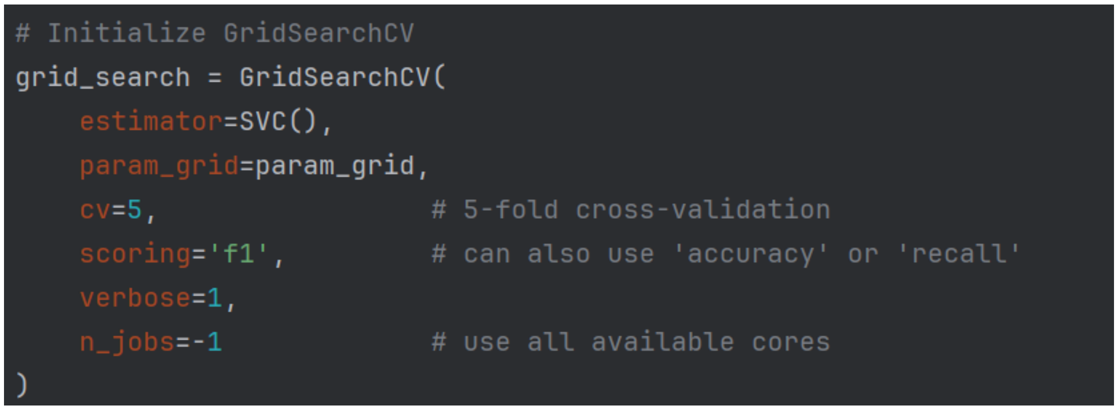
Two tuning techniques were applied:

GridSearchCV:

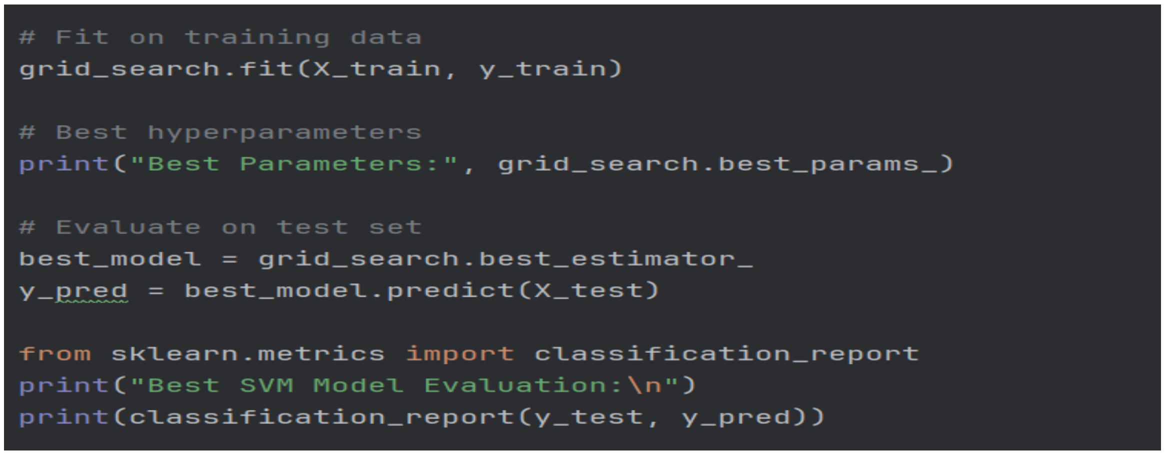
- Explores all combinations of parameters provided in a grid.
- Exhaustive and time-consuming but thorough.
- Used primarily for small datasets or fewer parameters.



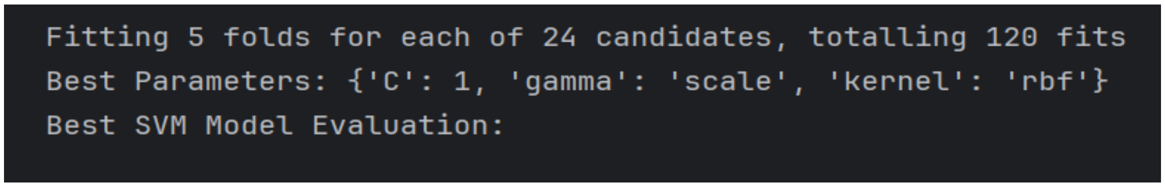
```python
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC

# Define parameter grid
param_grid = {
    'C': [0.1, 1, 10, 100],
    'kernel': ['linear', 'rbf', 'poly'],
    'gamma': ['scale', 'auto']
}
```



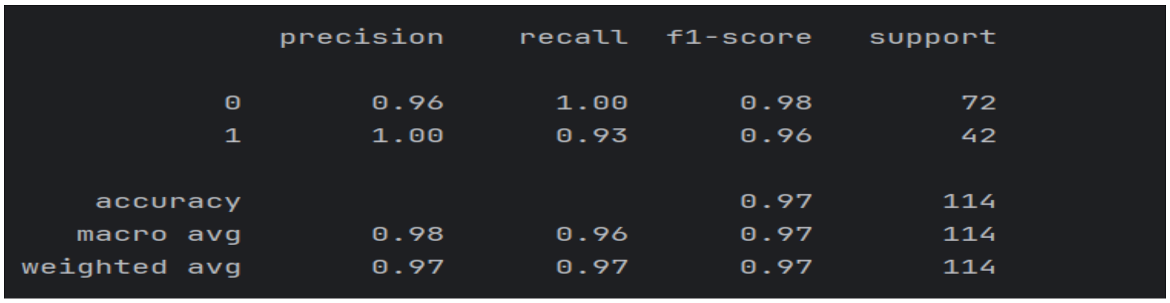
```python
# Initialize GridSearchCV
grid_search = GridSearchCV(
    estimator=SVC(),
    param_grid=param_grid,
    cv=5,                   # 5-fold cross-validation
    scoring='f1',           # can also use 'accuracy' or 'recall'
    verbose=1,
    n_jobs=-1               # use all available cores
)
```



```python
# Fit on training data
grid_search.fit(X_train, y_train)

# Best hyperparameters
print("Best Parameters:", grid_search.best_params_)

# Evaluate on test set
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

from sklearn.metrics import classification_report
print("Best SVM Model Evaluation:\n")
print(classification_report(y_test, y_pred))
```



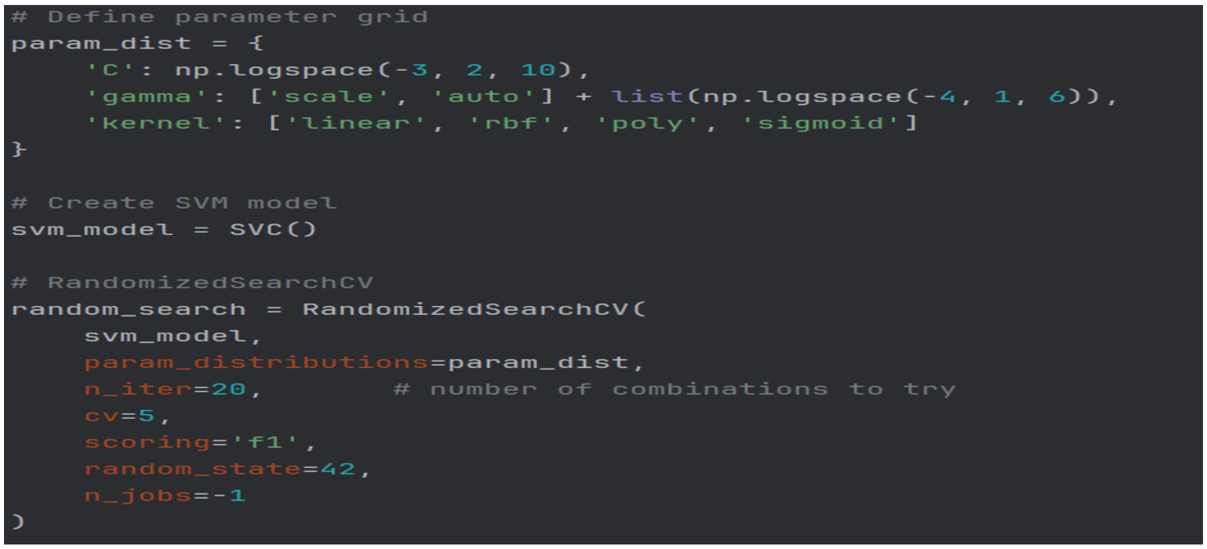
```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
Best Parameters: {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}
Best SVM Model Evaluation:
```



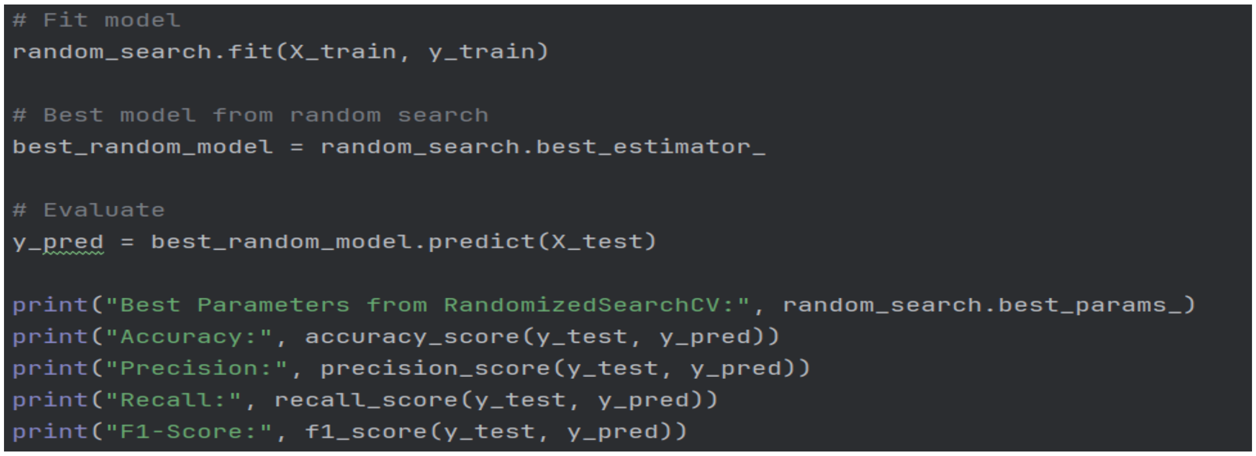
```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        72
           1       1.00      0.93      0.96        42

    accuracy                           0.97       114
   macro avg       0.98      0.96      0.97       114
weighted avg       0.97      0.97      0.97       114
```

RandomizedSearchCV:

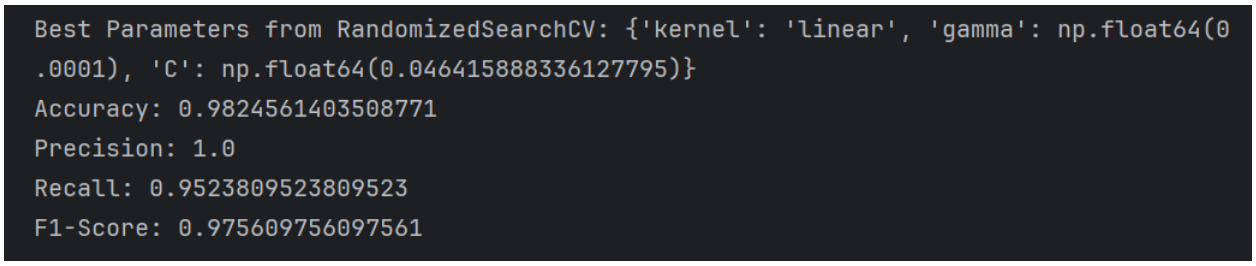
- Randomly selects combinations from a parameter distribution.
- Faster and efficient for large search spaces



```python
# Define parameter grid
param_dist = {
    'C': np.logspace(-3, 2, 10),
    'gamma': ['scale', 'auto'] + list(np.logspace(-4, 1, 6)),
    'kernel': ['linear', 'rbf', 'poly', 'sigmoid']
}

# Create SVM model
svm_model = SVC()

# RandomizedSearchCV
random_search = RandomizedSearchCV(
    svm_model,
    param_distributions=param_dist,
    n_iter=20,        # number of combinations to try
    cv=5,
    scoring='f1',
    random_state=42,
    n_jobs=-1
)
```



```
# Fit model
random_search.fit(X_train, y_train)

# Best model from random search
best_random_model = random_search.best_estimator_

# Evaluate
y_pred = best_random_model.predict(X_test)

print("Best Parameters from RandomizedSearchCV:", random_search.best_params_)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
print("F1-Score:", f1_score(y_test, y_pred))
```

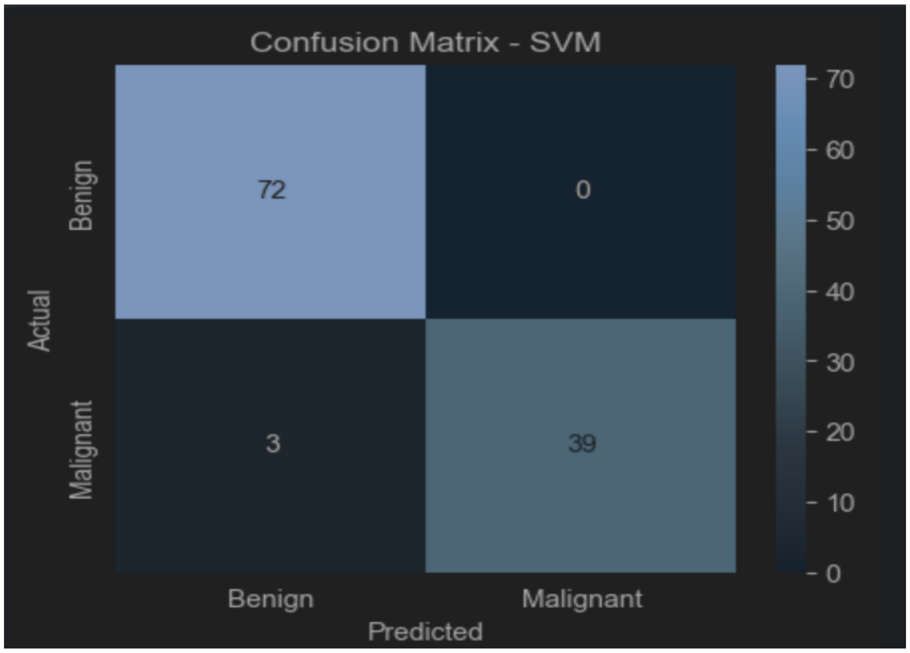


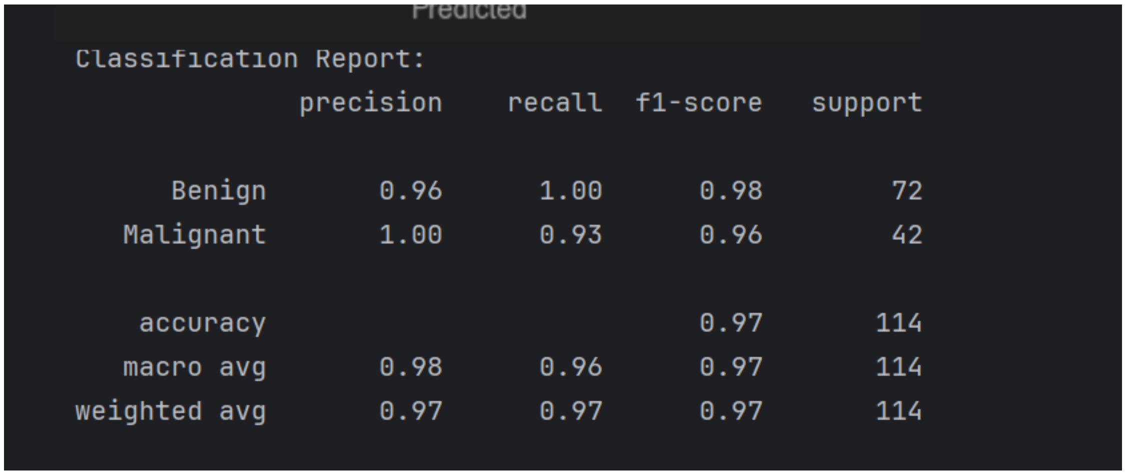
```
Best Parameters from RandomizedSearchCV: {'kernel': 'linear', 'gamma': np.float64(0
.0001), 'C': np.float64(0.046415888336127795)}
Accuracy: 0.9824561403508771
Precision: 1.0
Recall: 0.9523809523809523
F1-Score: 0.975609756097561
```

Hyperparameter tuning was applied particularly to the SVM and Random Forest models to find optimal parameters like:

- C and kernel for SVM
- n_estimators and max_depth for Random Forest

Confusion Matrix of SVM: As it is the most suited model for the prediction





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

          Benign       0.96      1.00      0.98        72
       Malignant       1.00      0.93      0.96        42

        accuracy                           0.97       114
       macro avg       0.98      0.96      0.97       114
    weighted avg       0.97      0.97      0.97       114
```

- ✓ Based on the evaluation metrics, the **SVM** was found to be the best-performing model due to its high precision, recall, and balanced F1-score. The tuned hyperparameters further boosted its performance.

7. **Conclusion:** This project demonstrates the importance of training and evaluating multiple machine learning models and applying hyperparameter tuning to optimize their performance. Such models can assist in medical diagnosis and decision support systems with a high degree of accuracy.