RA2311047010040

Index

மரம் வளர்ப்போம்
மண் வளம் காப்போம்

Superior®
Note Books

Name : Vishvar S

Subject : DL

Std. : .... Sec. : .... Roll No. ....

School : ....

No.	Date	Title	Marks	Signature
1	24/7/25	Exploring the DL platforms.		
2	31/7/25	Implement a classifier using open source dataset		
3	7/8/25	Study the classifier with respect to statistical parameters.		

7/8/25

Study the classifiers with respect to Statistical parameters.

Aim:

To Study & compare the perform of Decision Tree & K-Nearest Neighbors classifiers with respect to Statistical parameters such as accuracy. precision, recall & f1 score.

Objective:

1) To implement Decision Tree and knn classifiers.

2) Train both classifiers on a dataset

3) To evaluate & compare the classifiers using statisical performance metrics

4) To determine the better performing classifier for the given dataset.

Observation:

* Dataset : Iris dataset (150 samples, 3 classes)
* classifiers used:

1) Decision Tree
2) K-Nearest Neighbors
3) SVM
4) Navi Bayes.

* Statistical parameters:

Accuracy - Correct predictions / Total predictions

precision - TP / (TP+FP)

Recall - TP / (TP+FN)

F1 Score = Harmonic mean of precision & Recall

$$= \frac{\text{Precision} \times \text{Recall}}{\text{precision} + \text{Recall}}.$$

classes :

Maligant → cancer tumor

Benign: → non-cancer tumor.

TP = True positive

FN = False Negative

TN = True Negative

FP = False positive.

Pseudocode:

```
Input: Dataset with features x & y
        label
output: Evaluation metrics for each classifier

Start processing:

1) Import libraries
2) Load the dataset
3) preprocessing dataset
     → Handling missing
     → Normalize features with
          standard scalar.
     → split dataset into trained test
4) Defined the classifier
       → Decision Tree        → SVM
       → KNN.                 → Navi Bayes
5) For each classifier:
       → Train the model
       → predict the target labels
       → Evaluate metrics.
            a) Accuracy
            b) percision
            c) Recall
            d) f1-score
6) compare the results.
END
```

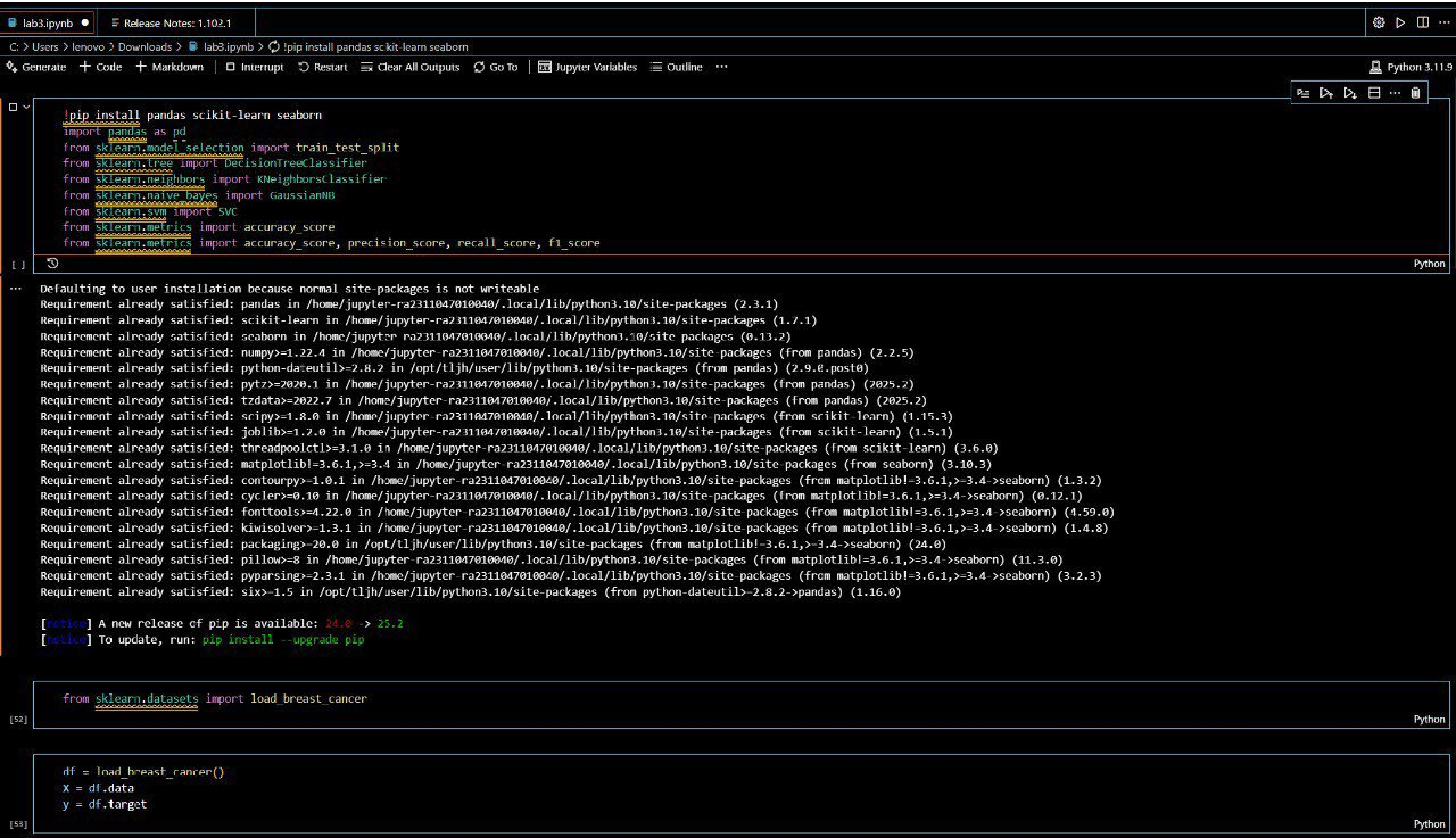
~~Decision~~ Classifier ~~Tree~~ Metrics.

Metric	Value	
	DT	KNN.
Accuracy	94.74	95.61
Precision	0.94	0.97
Recall	0.94	0.94
F1 score	0.94	0.95

Metric	value	
	NaviBayes	SVM
Accuracy	97.37	94.74
precision	0.98	0.96
Recall	0.97	0.93
F1_Score	0.97	0.94

Result:

The Iris dataset has be used to train Decision Tree SVM, NaviBayes and KNN and, the Statistical parameters has been compared & infered.



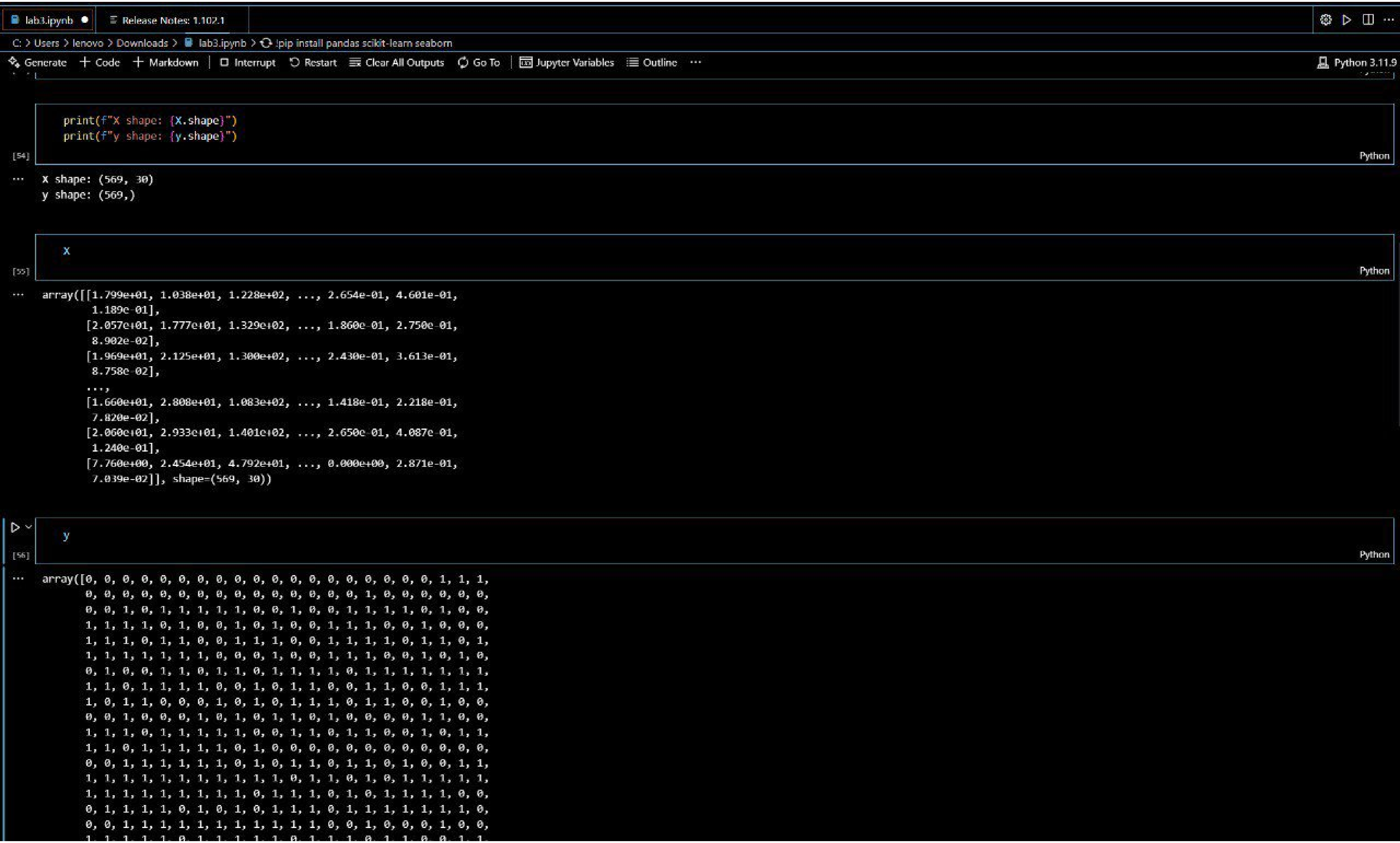
```python
!pip install pandas scikit-learn seaborn
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pandas in /home/jupyter-ra2311047010040/.local/lib/python3.10/site-packages (2.3.1)
Requirement already satisfied: scikit-learn in /home/jupyter-ra2311047010040/.local/lib/python3.10/site-packages (1.7.1)
Requirement already satisfied: seaborn in /home/jupyter-ra2311047010040/.local/lib/python3.10/site-packages (0.13.2)
Requirement already satisfied: numpy>=1.22.4 in /home/jupyter-ra2311047010040/.local/lib/python3.10/site-packages (from pandas) (2.2.5)
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/tljh/user/lib/python3.10/site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /home/jupyter-ra2311047010040/.local/lib/python3.10/site-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /home/jupyter-ra2311047010040/.local/lib/python3.10/site-packages (from pandas) (2025.2)
Requirement already satisfied: scipy>=1.8.0 in /home/jupyter-ra2311047010040/.local/lib/python3.10/site-packages (from scikit-learn) (1.15.3)
Requirement already satisfied: joblib>=1.2.0 in /home/jupyter-ra2311047010040/.local/lib/python3.10/site-packages (from scikit-learn) (1.5.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in /home/jupyter-ra2311047010040/.local/lib/python3.10/site-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /home/jupyter-ra2311047010040/.local/lib/python3.10/site-packages (from seaborn) (3.10.3)
Requirement already satisfied: contourpy>=1.0.1 in /home/jupyter-ra2311047010040/.local/lib/python3.10/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /home/jupyter-ra2311047010040/.local/lib/python3.10/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /home/jupyter-ra2311047010040/.local/lib/python3.10/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.59.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /home/jupyter-ra2311047010040/.local/lib/python3.10/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /opt/tljh/user/lib/python3.10/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (24.0)
Requirement already satisfied: pillow>=8 in /home/jupyter-ra2311047010040/.local/lib/python3.10/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /home/jupyter-ra2311047010040/.local/lib/python3.10/site-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.2.3)
Requirement already satisfied: six>=1.5 in /opt/tljh/user/lib/python3.10/site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)

[notice] A new release of pip is available: 24.0 -> 25.2
[notice] To update, run: pip install --upgrade pip
```

```python
from sklearn.datasets import load_breast_cancer
```

```python
df = load_breast_cancer()
X = df.data
y = df.target
```



```
print(f"X shape: {X.shape}")
print(f"y shape: {y.shape}")
```

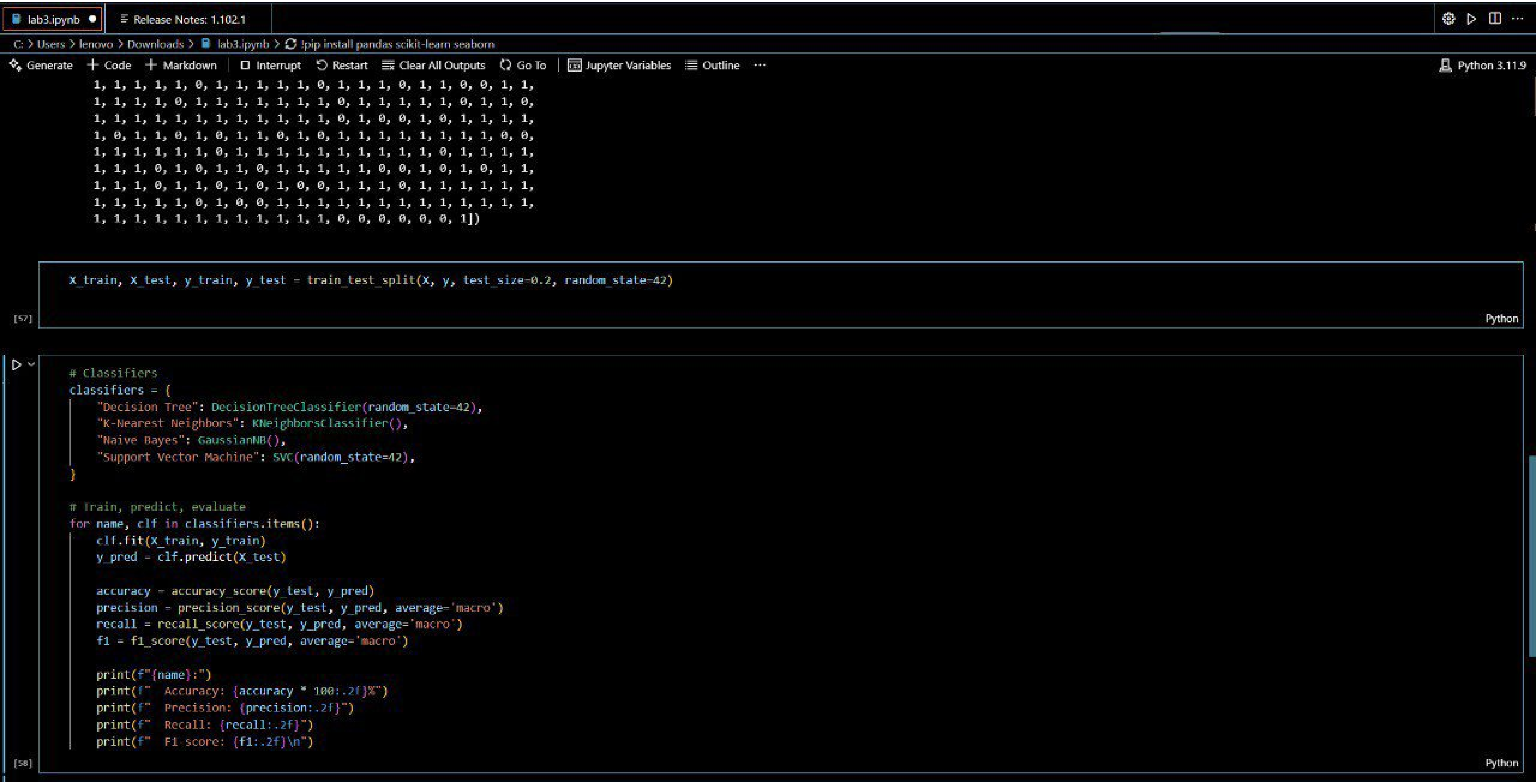
```
X shape: (569, 30)
y shape: (569,)
```

```
X
```

```
array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
         1.189e-01],
       [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
         8.902e-02],
       [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
         8.758e-02],
       ...,
       [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
         7.820e-02],
       [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
         1.240e-01],
       [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
         7.039e-02]], shape=(569, 30))
```

```
y
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
```



```
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1])
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

[57] Python

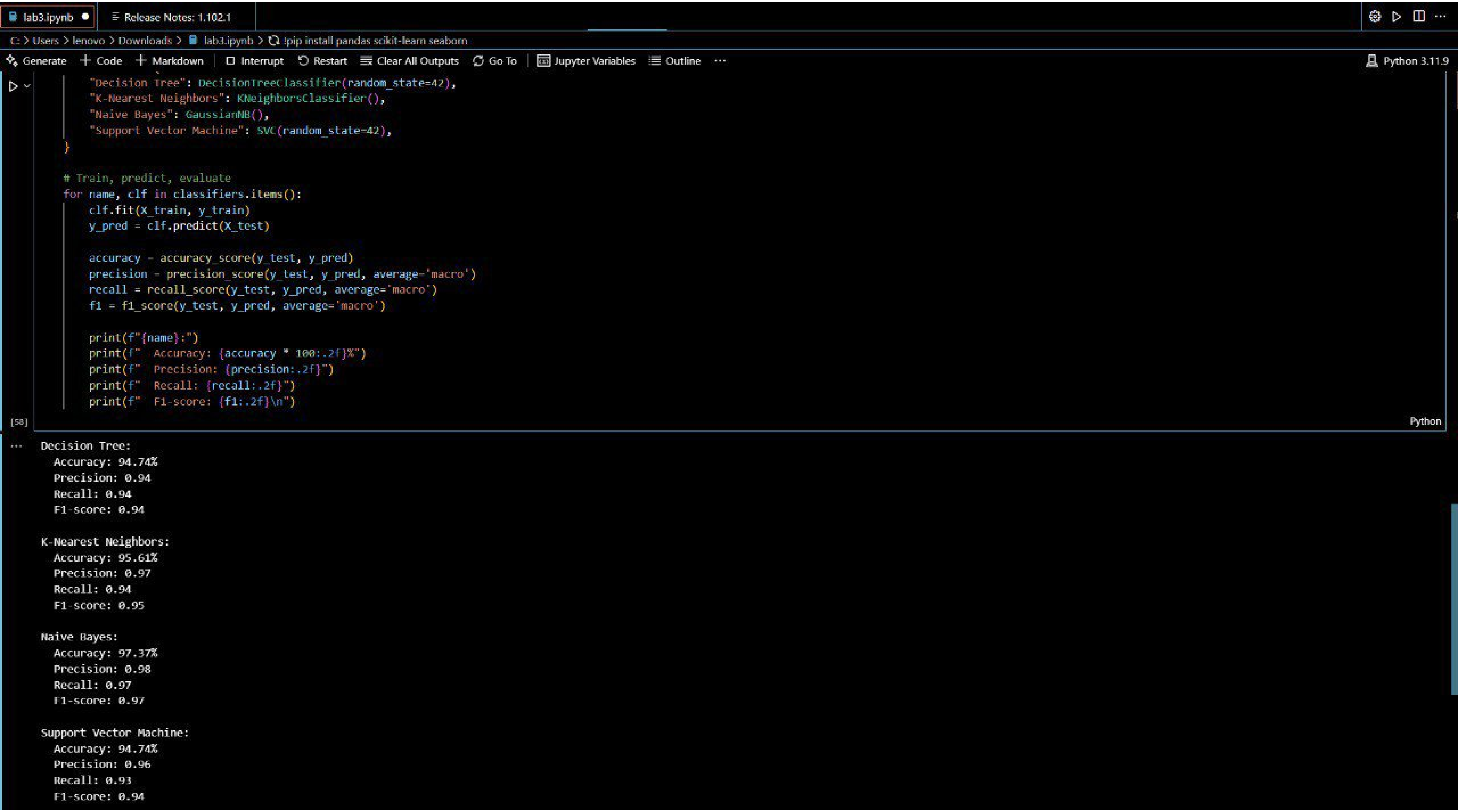
```
# Classifiers
classifiers = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "K-Nearest Neighbors": KNeighborsClassifier(),
    "Naive Bayes": GaussianNB(),
    "Support Vector Machine": SVC(random_state=42),
}

# Train, predict, evaluate
for name, clf in classifiers.items():
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='macro')
    recall = recall_score(y_test, y_pred, average='macro')
    f1 = f1_score(y_test, y_pred, average='macro')

    print(f"{name}:")
    print(f"  Accuracy: {accuracy * 100:.2f}%")
    print(f"  Precision: {precision:.2f}")
    print(f"  Recall: {recall:.2f}")
    print(f"  F1 score: {f1:.2f}\n")
```

[58] Python



```python
        "Decision Tree": DecisionTreeClassifier(random_state=42),
        "K-Nearest Neighbors": KNeighborsClassifier(),
        "Naive Bayes": GaussianNB(),
        "Support Vector Machine": SVC(random_state=42),
    }

    # Train, predict, evaluate
    for name, clf in classifiers.items():
        clf.fit(X_train, y_train)
        y_pred = clf.predict(X_test)

        accuracy = accuracy_score(y_test, y_pred)
        precision = precision_score(y_test, y_pred, average='macro')
        recall = recall_score(y_test, y_pred, average='macro')
        f1 = f1_score(y_test, y_pred, average='macro')

        print(f"{name}:")
        print(f"  Accuracy: {accuracy * 100:.2f}%")
        print(f"  Precision: {precision:.2f}")
        print(f"  Recall: {recall:.2f}")
        print(f"  F1-score: {f1:.2f}\n")
```

```
Decision Tree:
  Accuracy: 94.74%
  Precision: 0.94
  Recall: 0.94
  F1-score: 0.94

K-Nearest Neighbors:
  Accuracy: 95.61%
  Precision: 0.97
  Recall: 0.94
  F1-score: 0.95

Naive Bayes:
  Accuracy: 97.37%
  Precision: 0.98
  Recall: 0.97
  F1-score: 0.97

Support Vector Machine:
  Accuracy: 94.74%
  Precision: 0.96
  Recall: 0.93
  F1-score: 0.94
```