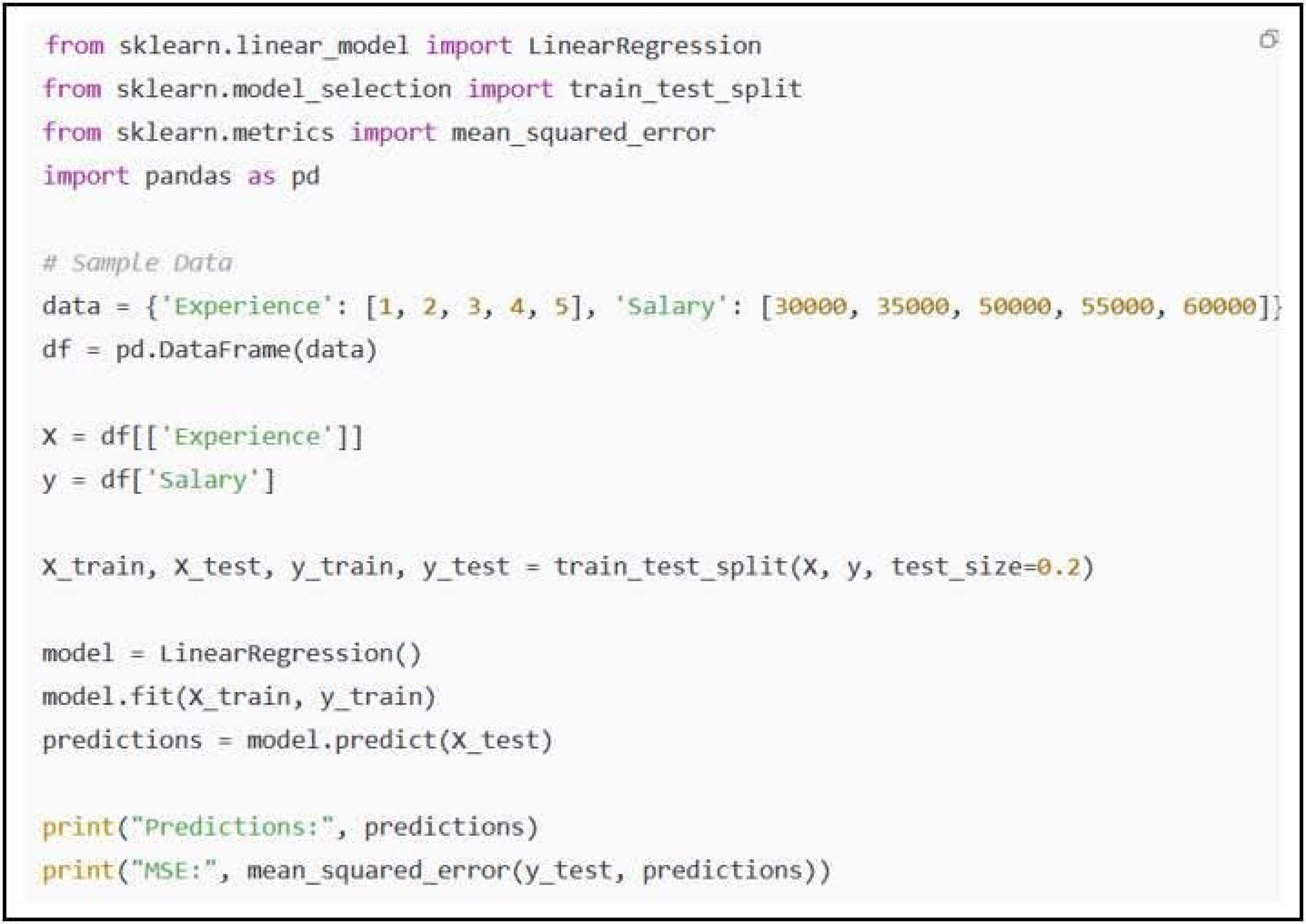
# 100 + Basic to Advance Machine Learning  Programs

Let’s start coding from today

# 100 + Basic to Advance Machine learning Programs

## Program 1

Linear Regression on a Dataset



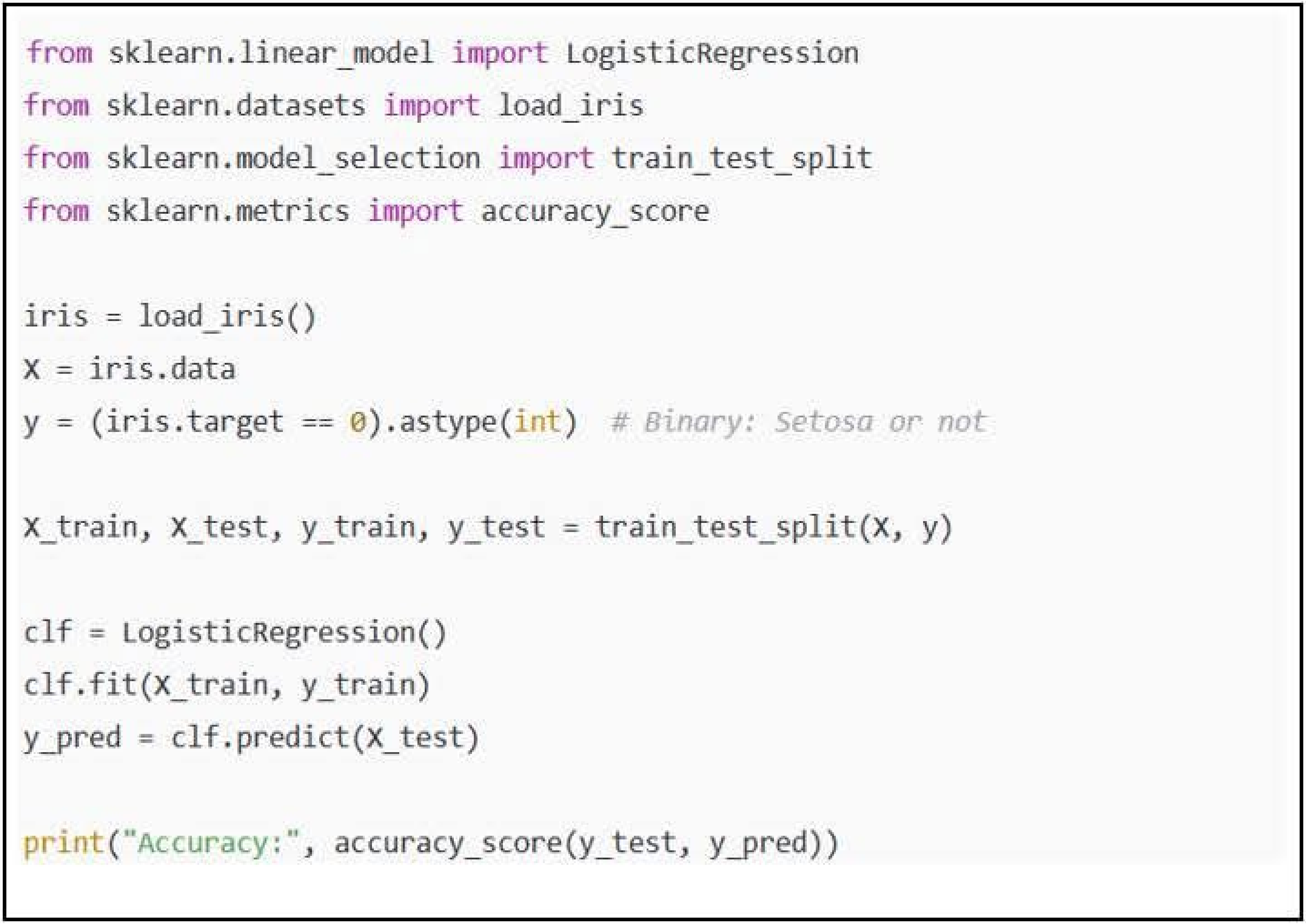
```python
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import pandas as pd

# Sample Data
data = {'Experience': [1, 2, 3, 4, 5], 'Salary': [30000, 35000, 50000, 55000, 60000]}
df = pd.DataFrame(data)

X = df[['Experience']]
y = df['Salary']

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

model = LinearRegression()
model.fit(X_train, y_train)
predictions = model.predict(X_test)

print("Predictions:", predictions)
print("MSE:", mean_squared_error(y_test, predictions))
```

## Vbnet Output

```
Predictions: [Estimated salaries]
MSE: [mean squared error value]
```

## Program 2

Logistic Regression for Binary Classification



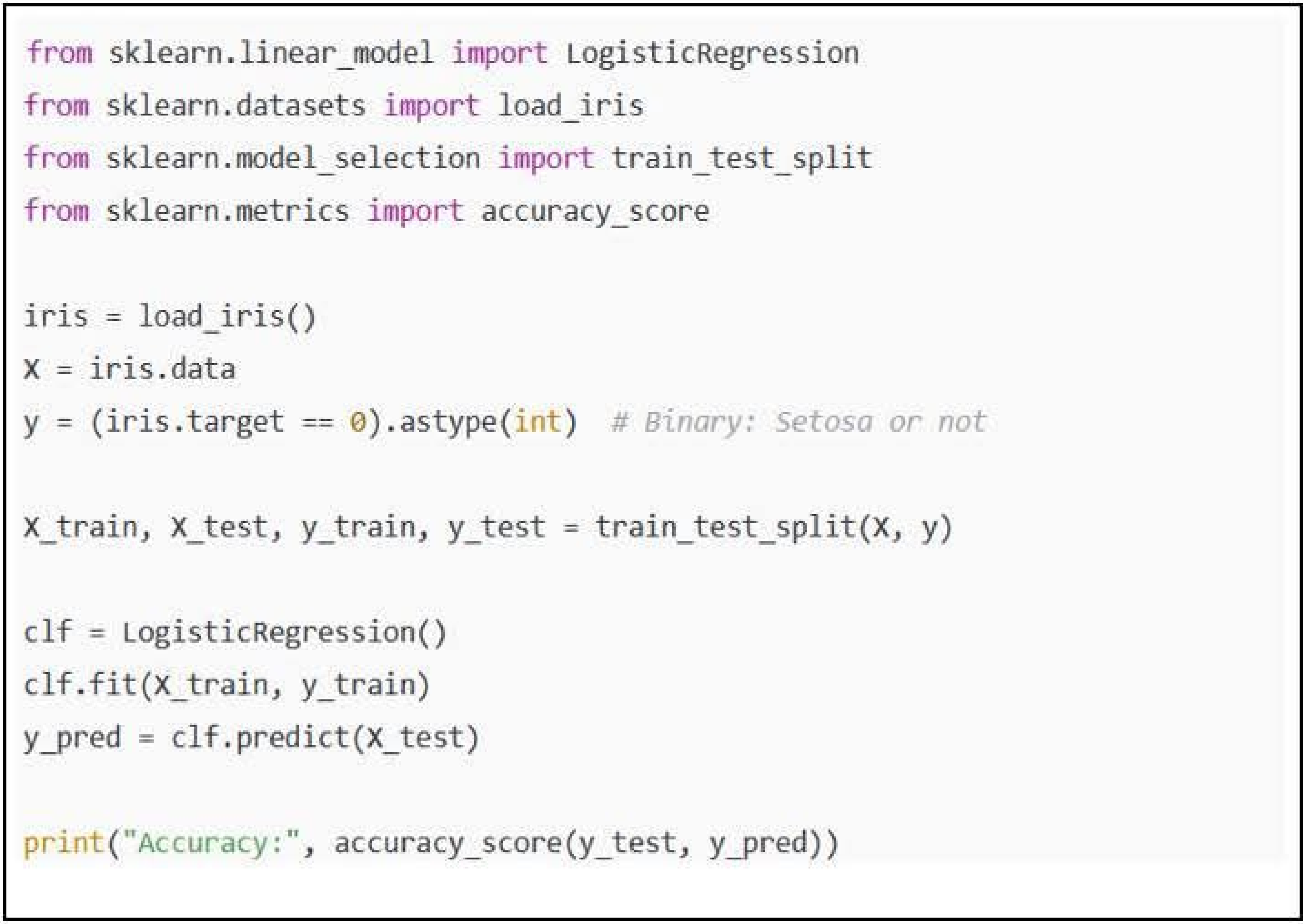
```python
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

iris = load_iris()
X = iris.data
y = (iris.target == 0).astype(int)  # Binary: Setosa or not

X_train, X_test, y_train, y_test = train_test_split(X, y)

clf = LogisticRegression()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
```

### Makefile Output

```
Accuracy: 1.0 (or similar)
```

## Program 2

Logistic Regression for Binary Classification



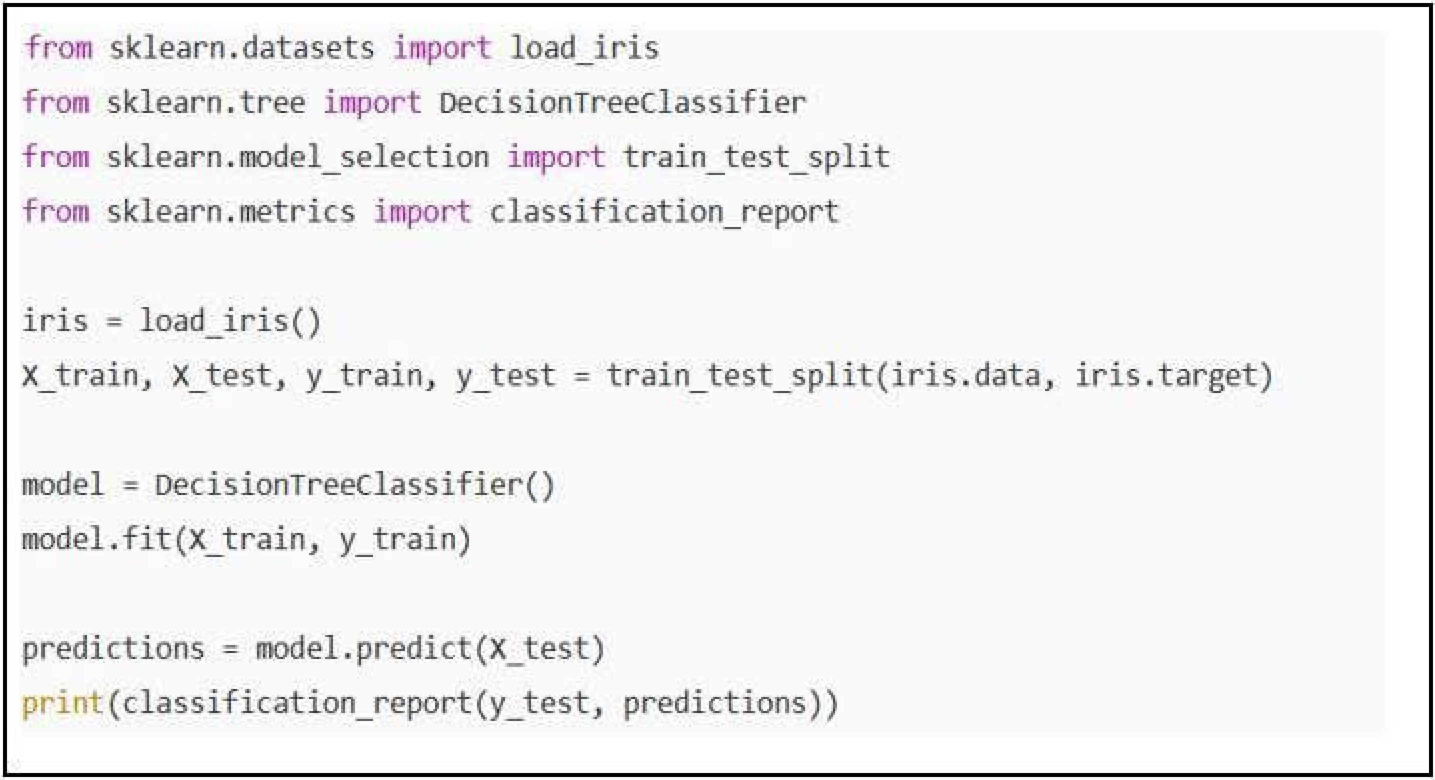
```
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

iris = load_iris()
X = iris.data
y = (iris.target == 0).astype(int)  # Binary: Setosa or not

X_train, X_test, y_train, y_test = train_test_split(X, y)

clf = LogisticRegression()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
```

### Makefile Output

Accuracy: 1.0 (or similar)

## Program 3

Decision Tree Classifier



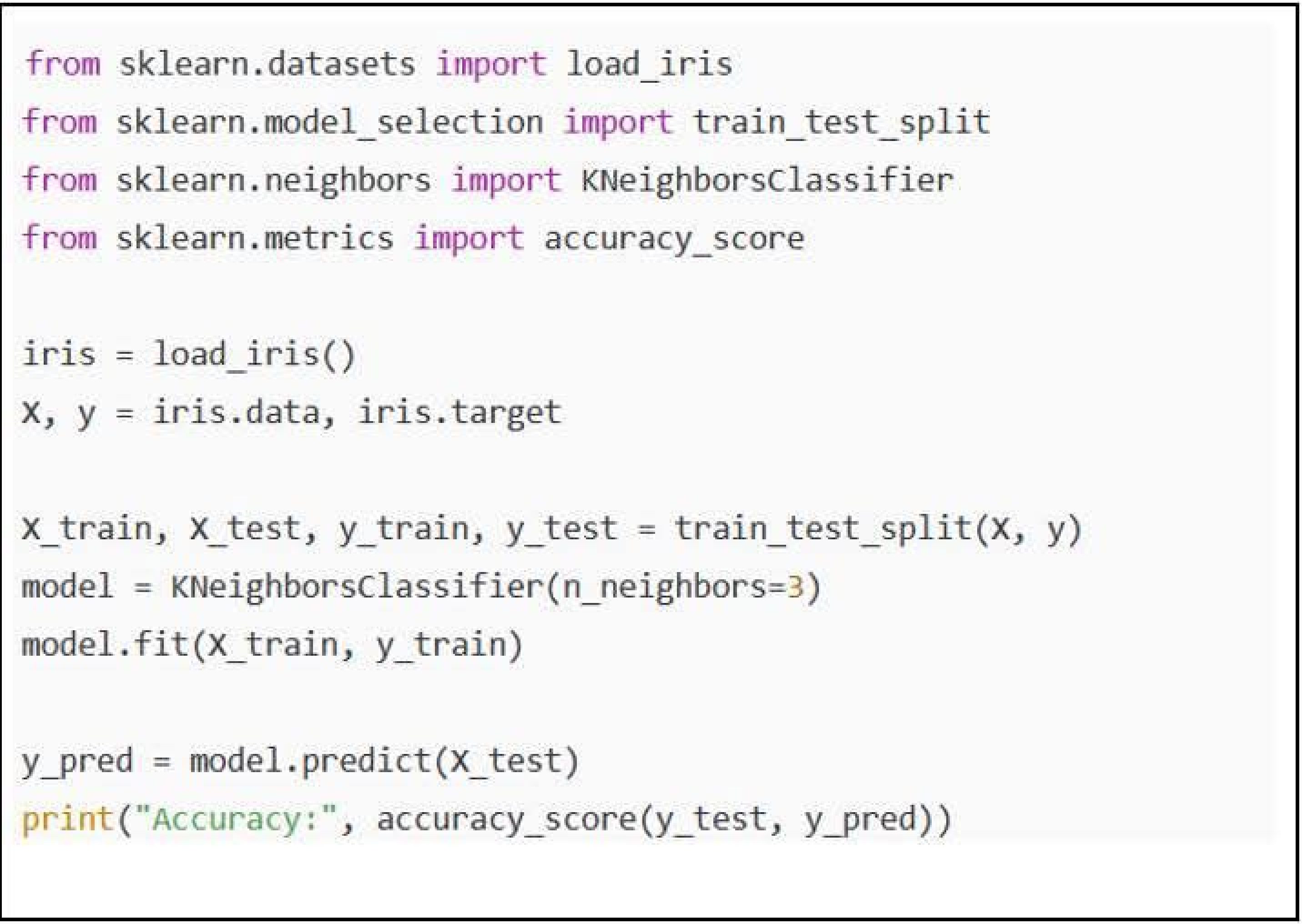
```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target)

model = DecisionTreeClassifier()
model.fit(X_train, y_train)

predictions = model.predict(X_test)
print(classification_report(y_test, predictions))
```

## SQL Output

Classification report with precision, recall,
F1-score

## Program 4

K-Nearest Neighbors (KNN)



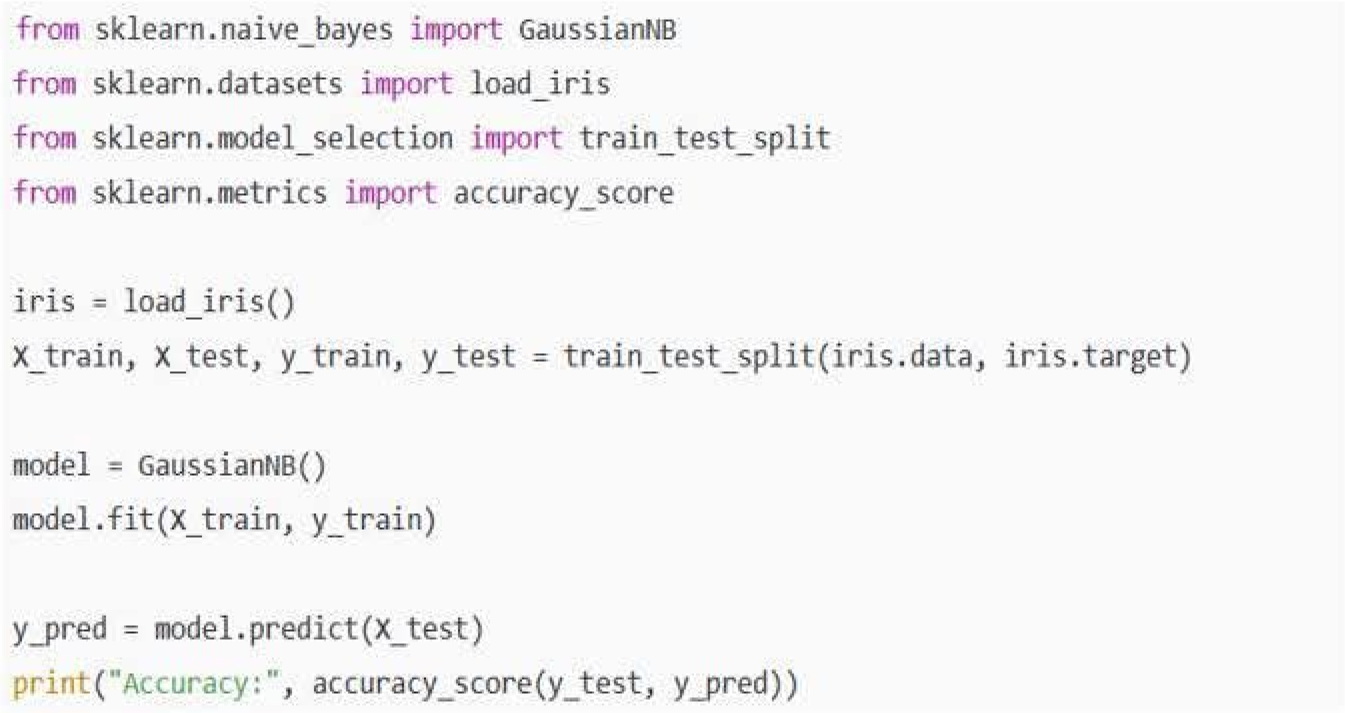
```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

iris = load_iris()
X, y = iris.data, iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y)
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

### Makefile Output

```
Accuracy: 0.97 (or similar)
```

## Program 5

Naive Bayes Classifier



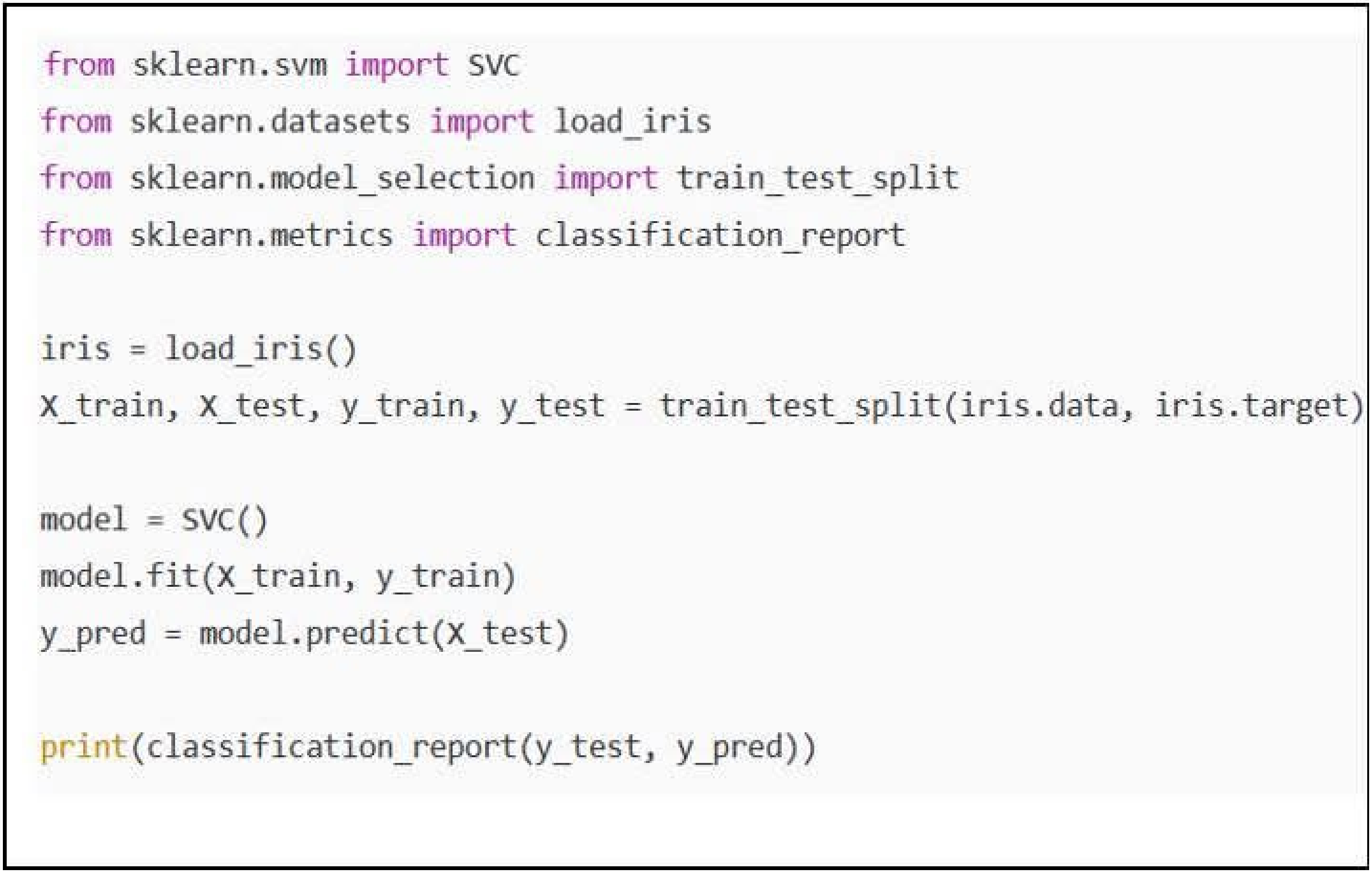
```
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target)

model = GaussianNB()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

## Makefile Output

Accuracy: 0.97 (or similar)

## Program 6

Support Vector Machine (SVM)



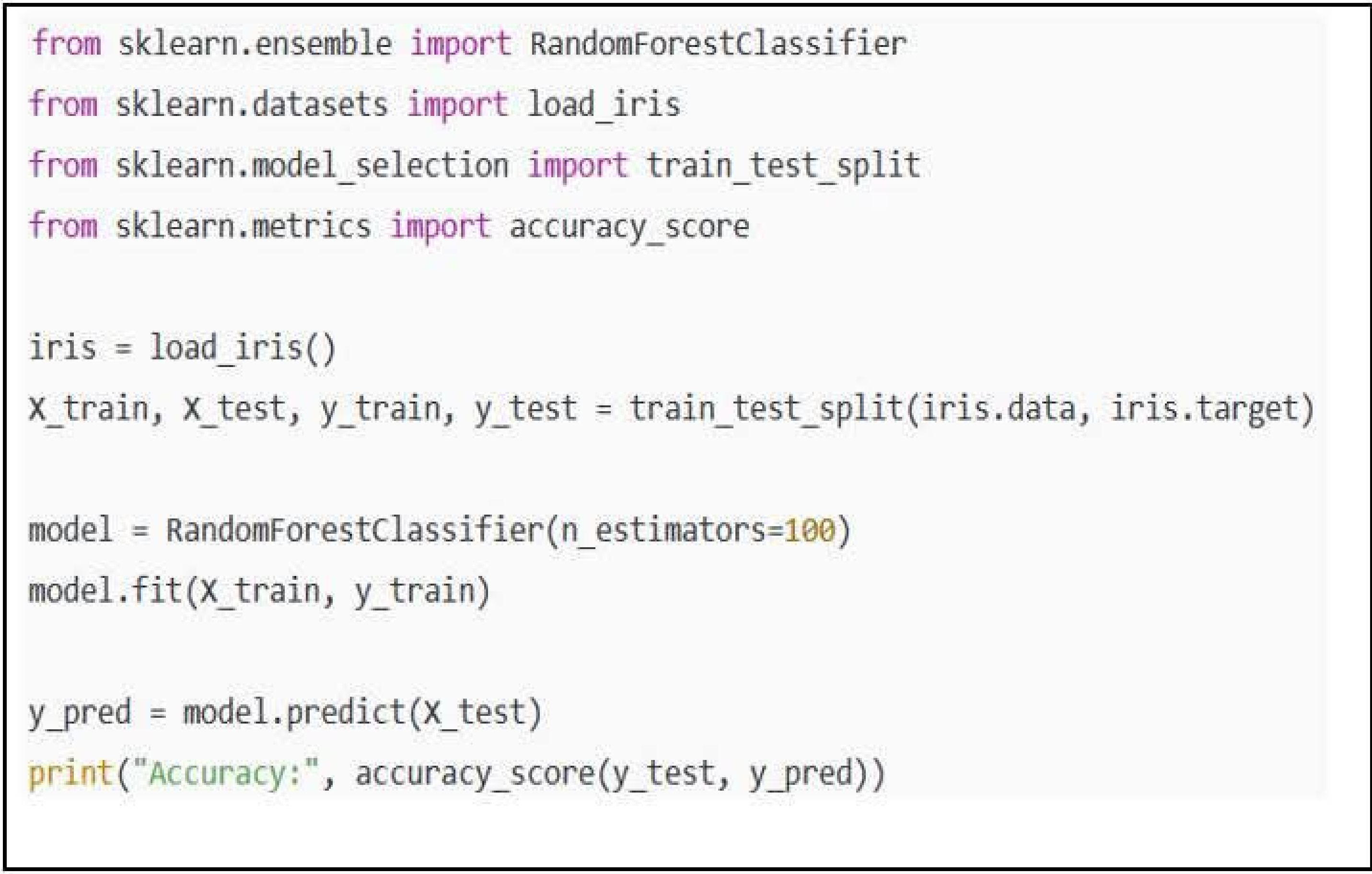
```python
from sklearn.svm import SVC
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target)

model = SVC()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

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

## SQL Output

Classification report with precision, recall, F1-score

## Program 7

Random Forest Classifier



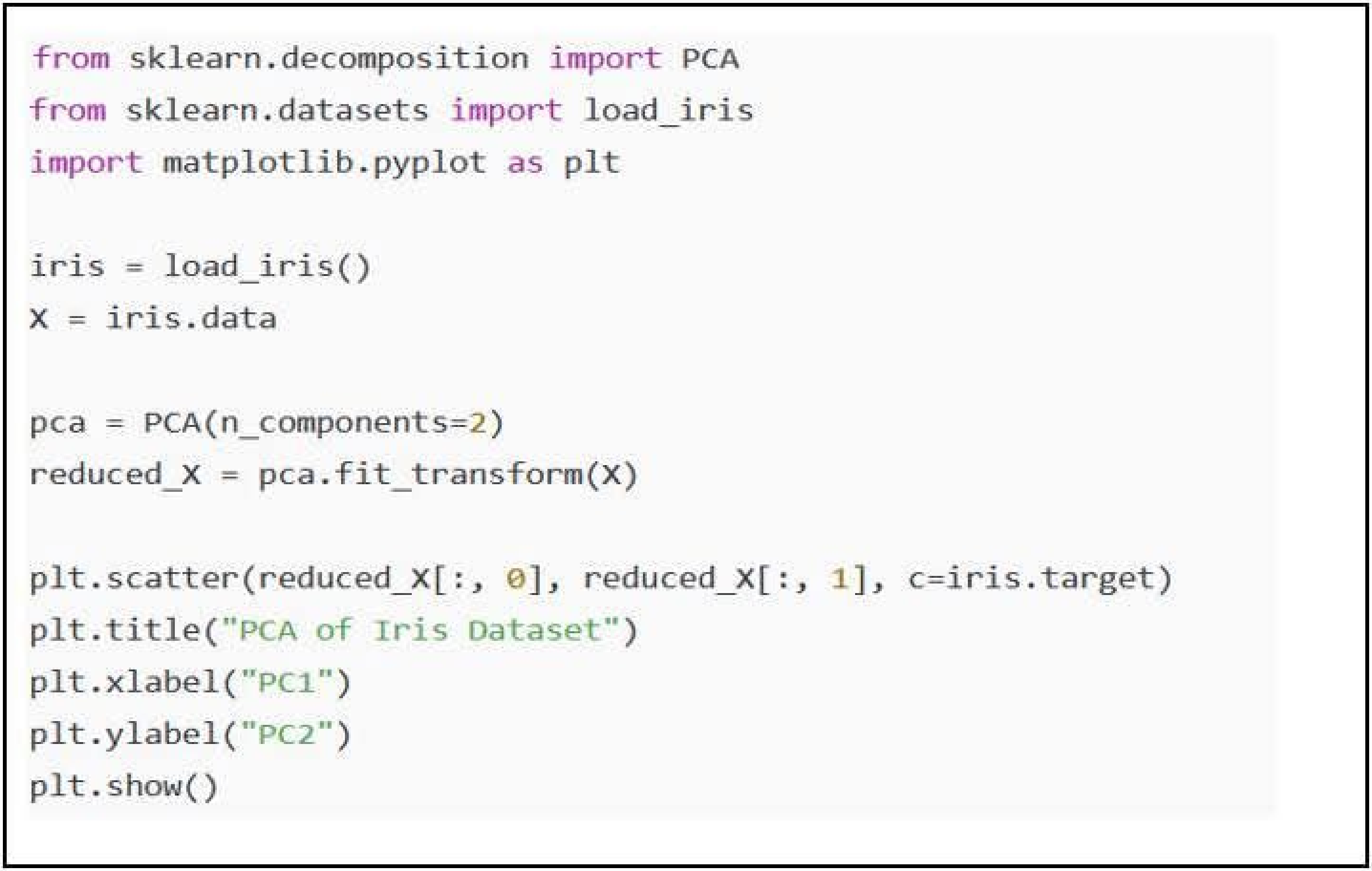
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target)

model = RandomForestClassifier(n_estimators=100)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

## Makefile Output

```
Accuracy: 0.96 (or similar)
```

## Program 8

Principal Component Analysis (PCA)



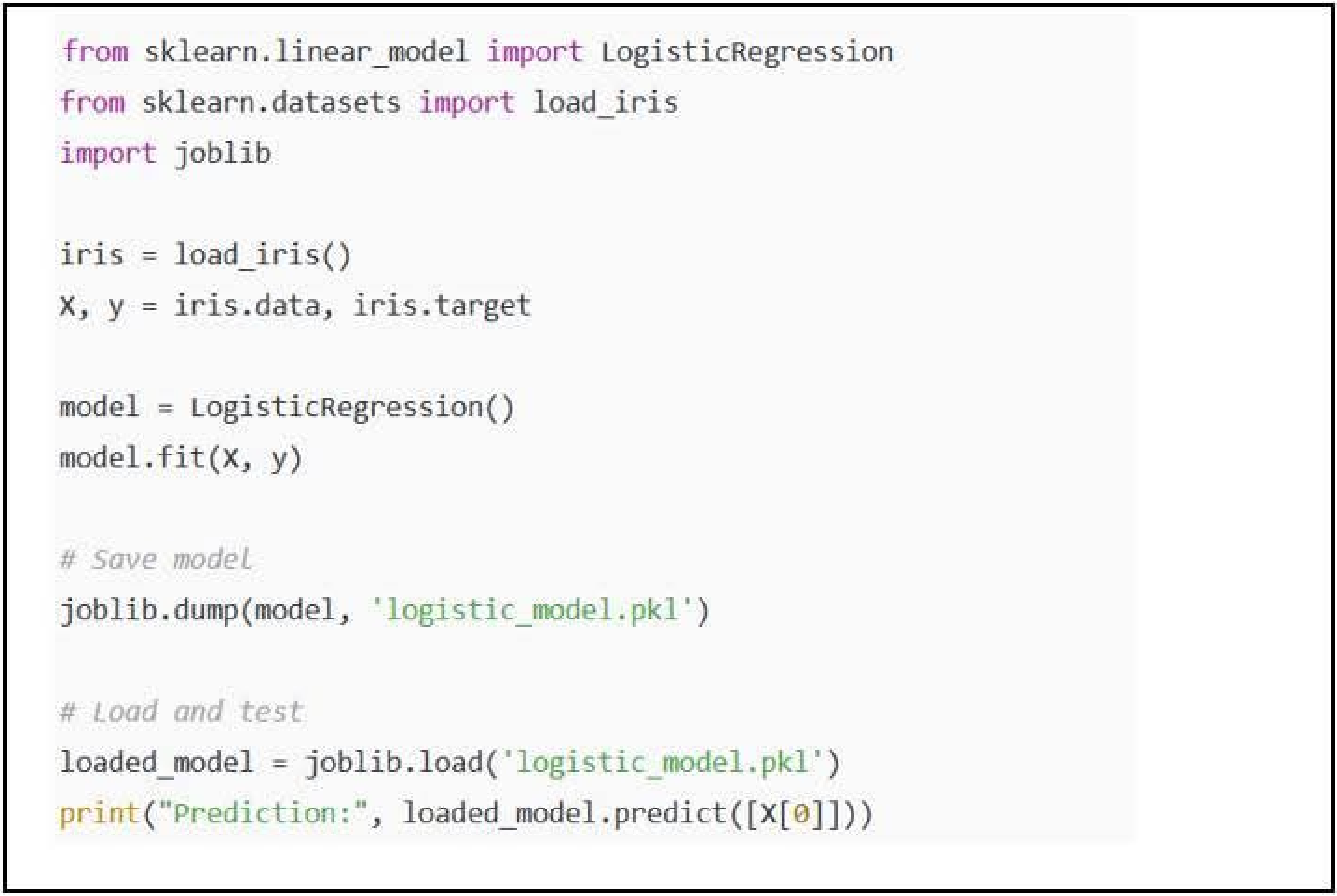
```
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt

iris = load_iris()
X = iris.data

pca = PCA(n_components=2)
reduced_X = pca.fit_transform(X)

plt.scatter(reduced_X[:, 0], reduced_X[:, 1], c=iris.target)
plt.title("PCA of Iris Dataset")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.show()
```

## Mathematic Output

PCA Scatter Plot of Iris dataset

## Program 10

Model Persistence with Joblib



```python
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris
import joblib

iris = load_iris()
X, y = iris.data, iris.target

model = LogisticRegression()
model.fit(X, y)

# Save model
joblib.dump(model, 'logistic_model.pkl')

# Load and test
loaded_model = joblib.load('logistic_model.pkl')
print("Prediction:", loaded_model.predict([X[0]]))
```

## Vbnet Output

```
Prediction: [0] (or similar depending on data)
```