LAB WORK

Machine learning in Python as per Global Curriculum

Generation of Predictable Dataset

```
import pandas as pd
import numpy as np

# Set seed for reproducibility
np.random.seed(42)
# Number of samples
n_samples = 1000
# Generate features X1, X2, X3
X1 = np.random.uniform(0, 10, n_samples)
X2 = np.random.uniform(0, 10, n_samples)
X3 = np.random.uniform(0, 10, n_samples)
# Predictable rule: Y = 1 if 2*X1 + 3*X2 + X3 > 40, else 0
Y = ((2*X1 + 3*X2 + 1*X3) > 40).astype(int)

# Create DataFrame
df = pd.DataFrame({
    'X1': X1,
    'X2': X2,
    'X3': X3,
    'Y': Y
})
# Save to CSV
df.to_csv('predictable_dataset.csv', index=False)

print("Predictable dataset saved to 'predictable_dataset.csv'.")
```

Training and Testing of SVM Model

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Step 1: Load the dataset
df = pd.read_csv("predictable_dataset.csv")  # Replace with your actual file name

# Step 2: Drop missing values (optional)
df = df.dropna()

# Step 3: Split features and target
X = df.iloc[:, :-1]  # All columns except the last
y = df.iloc[:, -1]   # Only the last column

# Step 4: Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 5: Train the SVM model
model = SVC(kernel='rbf', C=1.0, gamma='scale')  # You can change kernel to 'linear', 'poly', etc.
model.fit(X_train, y_train)

# Step 6: Predict on test data
y_pred = model.predict(X_test)

# Step 7: Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.995

Confusion Matrix:
 [[166   1]
 [  0  33]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      1.00       167
           1       0.97      1.00      0.99        33

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

Training and Testing of Naïve Bayes model

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Step 1: Load CSV dataset
df = pd.read_csv("predictable_dataset.csv")  # Replace with your actual file name

# Step 2: Drop missing values (if any)
df = df.dropna()

# Step 3: Split features and target
X = df.iloc[:, :-1]  # All columns except the last
y = df.iloc[:, -1]   # Only the last column (target)

# Step 4: Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 5: Train Naive Bayes model
model = GaussianNB()
model.fit(X_train, y_train)

# Step 6: Predict on test data
y_pred = model.predict(X_test)

# Step 7: Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.97

Confusion Matrix:
 [[166   1]
 [  5  28]]

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       167
           1       0.97      0.85      0.90        33

    accuracy                           0.97       200
   macro avg       0.97      0.92      0.94       200
weighted avg       0.97      0.97      0.97       200
```

Training and Testing of Random Forest model

```
# Import required libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Step 1: Load the CSV file
df = pd.read_csv("predictable_dataset.csv")  # Replace with your actual filename

# Step 2: Check for missing values and drop if needed
df = df.dropna()

# Step 3: Split features (X) and label (y) — assuming last column is target
X = df.iloc[:, :-1]  # All columns except last
y = df.iloc[:, -1]   # Only the last column

# Step 4: Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 5: Train Random Forest model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Step 6: Make predictions
y_pred = model.predict(X_test)

# Step 7: Evaluate performance
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.975

Confusion Matrix:
 [[165   2]
 [  3  30]]

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       167
           1       0.94      0.91      0.92        33

    accuracy                           0.97       200
   macro avg       0.96      0.95      0.95       200
weighted avg       0.97      0.97      0.97       200
```

Common Code Block for Data Loading

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report

# Load and prepare dataset
df = pd.read_csv("predictable_dataset.csv")
df = df.dropna()

X = df.iloc[:, :-1]  # Features
y = df.iloc[:, -1]   # Target

# Optional scaling (good for SVM, KNN, ANN)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X = scaler.fit_transform(X)

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

Logistic Regression

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Logistic Regression Accuracy: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       167
           1       1.00      1.00      1.00        33

    accuracy                           1.00       200
   macro avg       1.00      1.00      1.00       200
weighted avg       1.00      1.00      1.00       200
```

KNN Accuracy

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=5)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("KNN Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
KNN Accuracy: 0.985
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       167
           1       0.94      0.97      0.96        33

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

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Decision Tree Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Decision Tree Accuracy: 0.955
              precision    recall  f1-score   support

           0       0.98      0.96      0.97       167
           1       0.83      0.91      0.87        33

    accuracy                           0.95       200
   macro avg       0.91      0.94      0.92       200
weighted avg       0.96      0.95      0.96       200
```

XGBoost Classifier

```
from xgboost import XGBClassifier
model = XGBClassifier(use label encoder=False, eval metric='mlogloss')
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("XGBoost Accuracy:", accuracy_score(y_test, y_pred))
print(classification report(y test, y pred))
```

```
  XGBoost Accuracy: 0.98
                precision    recall  f1-score   support

             0       0.99      0.99      0.99       167
             1       0.94      0.94      0.94        33

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

ANN (MLP Classifier)

```
from sklearn.neural_network import MLPClassifier

model = MLPClassifier(hidden_layer_sizes=(100, 50), max_iter=500,
random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("ANN (MLPClassifier) Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
ANN (MLPClassifier) Accuracy: 0.985
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       167
           1       1.00      0.91      0.95        33

    accuracy                           0.98       200
   macro avg       0.99      0.95      0.97       200
weighted avg       0.99      0.98      0.98       200
```

AdaBoost Classifier

```
from sklearn.ensemble import AdaBoostClassifier

model = AdaBoostClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

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

```
AdaBoost Accuracy: 0.99
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       167
           1       0.97      0.97      0.97        33

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

Voting Classifier

```
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

clf1 = LogisticRegression(max_iter=1000)
clf2 = RandomForestClassifier(n_estimators=100, random_state=42)
clf3 = SVC(kernel='linear', probability=True)

model = VotingClassifier(estimators=[
    ('lr', clf1), ('rf', clf2), ('svc', clf3)
], voting='soft')

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

print("Voting Classifier Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
Voting Classifier Accuracy: 0.995
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       167
           1       1.00      0.97      0.98        33

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