ML_Assignment

November 8, 2024

ML_Assignment: Model For Classification of Drug Type

Done By : Gopika and Prateek Kumar

[ ]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder

# Load the dataset
df = pd.read_csv('drug_type_classification.csv')

# Encode the target variable 'Drug' (if it's categorical)
label_encoder = LabelEncoder()
df['Drug'] = label_encoder.fit_transform(df['Drug'])

# Separate features and target variable
X = df.drop('Drug', axis=1)
y = df['Drug']

# Convert categorical features to numerical using one-hot encoding
X = pd.get_dummies(X, drop_first=True)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Train a Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Evaluate the model
y_pred = model.predict(X_test)
print("Classification Report:")
print(classification_report(y_test, y_pred))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

[7]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score
from sklearn.preprocessing import LabelEncoder

# Load the dataset
df = pd.read_csv('drug_type_classification.csv')

# Encode the target variable 'Drug' (if it's categorical)
label_encoder = LabelEncoder()
df['Drug'] = label_encoder.fit_transform(df['Drug'])

# Separate features and target variable
X = df.drop('Drug', axis=1)
y = df['Drug']

# Convert categorical features to numerical using one-hot encoding
X = pd.get_dummies(X, drop_first=True)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Train a Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Evaluate the model
y_pred = model.predict(X_test)
print("Classification Report:")
print(classification_report(y_test, y_pred))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Model Accuracy:", accuracy)
```

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

           0       1.00      1.00      1.00        15
           1       1.00      1.00      1.00         6
           2       1.00      1.00      1.00         3
           3       1.00      1.00      1.00         5
           4       1.00      1.00      1.00        11

    accuracy                           1.00        40
   macro avg       1.00      1.00      1.00        40
weighted avg       1.00      1.00      1.00        40

Confusion Matrix:
[[15  0  0  0  0]
 [ 0  6  0  0  0]
 [ 0  0  3  0  0]
 [ 0  0  0  5  0]
 [ 0  0  0  0 11]]
Model Accuracy: 1.0
```

[16]:

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder

# Load the dataset
df = pd.read_csv('drug_type_classification.csv')

# Encode the target variable 'Drug' (if it's categorical)
label_encoder = LabelEncoder()
df['Drug'] = label_encoder.fit_transform(df['Drug'])

# Separate features and target variable
X = df.drop('Drug', axis=1)
y = df['Drug']

# Convert categorical features to numerical using one-hot encoding
X = pd.get_dummies(X, drop_first=True)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Train a Random Forest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Evaluate the model
y_pred = model.predict(X_test)
print("Classification Report:")
print(classification_report(y_test, y_pred))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# Print feature names to identify exact column names
print("Feature names used during training:", X.columns)

# Define a function to make manual predictions with correct feature names
def manual_test(model, label_encoder):
    # Define input data with exact feature names
    input_data = {
        'Age': [43],                           # Example age
        'Sex_M': [1],                          # 1 for male, 0 for female (encoded as␣
 ↪per get_dummies)
        'BP_LOW': [0],                         # 1 if BP is Low, else 0
        'BP_NORMAL': [0],                      # 1 if BP is Normal, else 0
        'Cholesterol_NORMAL': [0],             # 1 if Cholesterol is Normal, else 0
        'Na_to_K': [13.927]                      # Example Sodium-to-Potassium ratio
    }

    # Convert to DataFrame to match input format
    input_df = pd.DataFrame(input_data)

    # Ensure the input has all required columns in the same order as training␣
 ↪data
    input_df = input_df.reindex(columns=X.columns, fill_value=0)

    # Predict using the model
    prediction = model.predict(input_df)

    # Decode the prediction back to the original class name
    predicted_drug = label_encoder.inverse_transform(prediction)

    print("Predicted Drug Type:", predicted_drug[0])

# Run the function to test manually
manual_test(model, label_encoder)
```

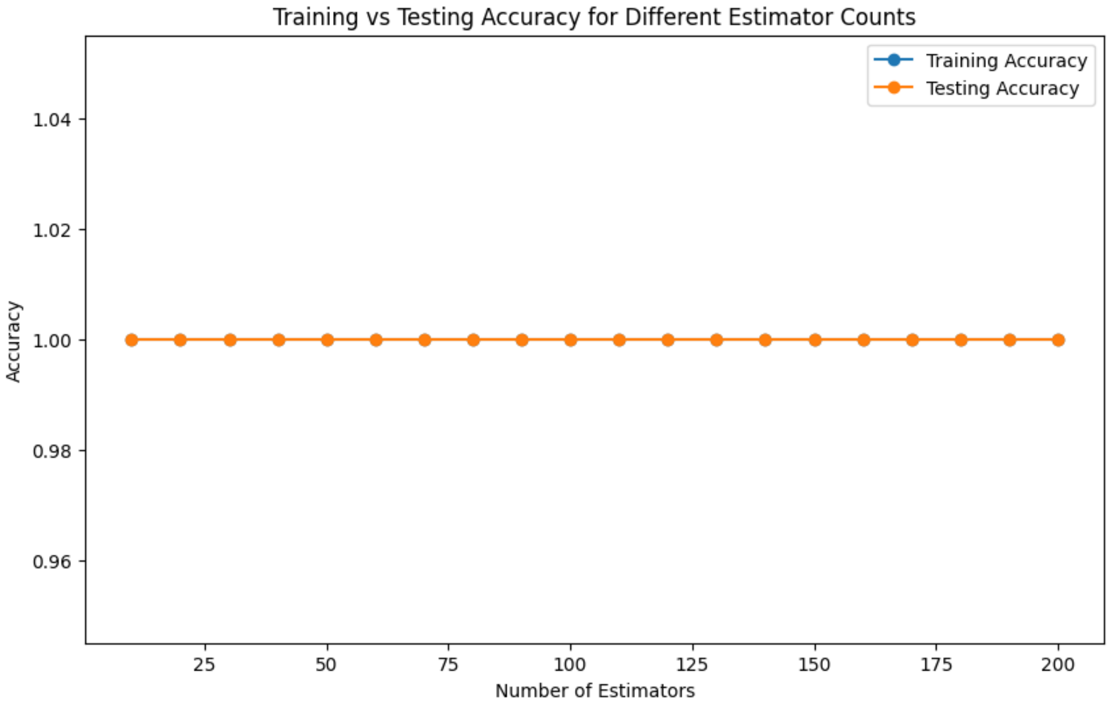
```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       1.00      1.00      1.00         6
           2       1.00      1.00      1.00         3
           3       1.00      1.00      1.00         5
           4       1.00      1.00      1.00        11
```

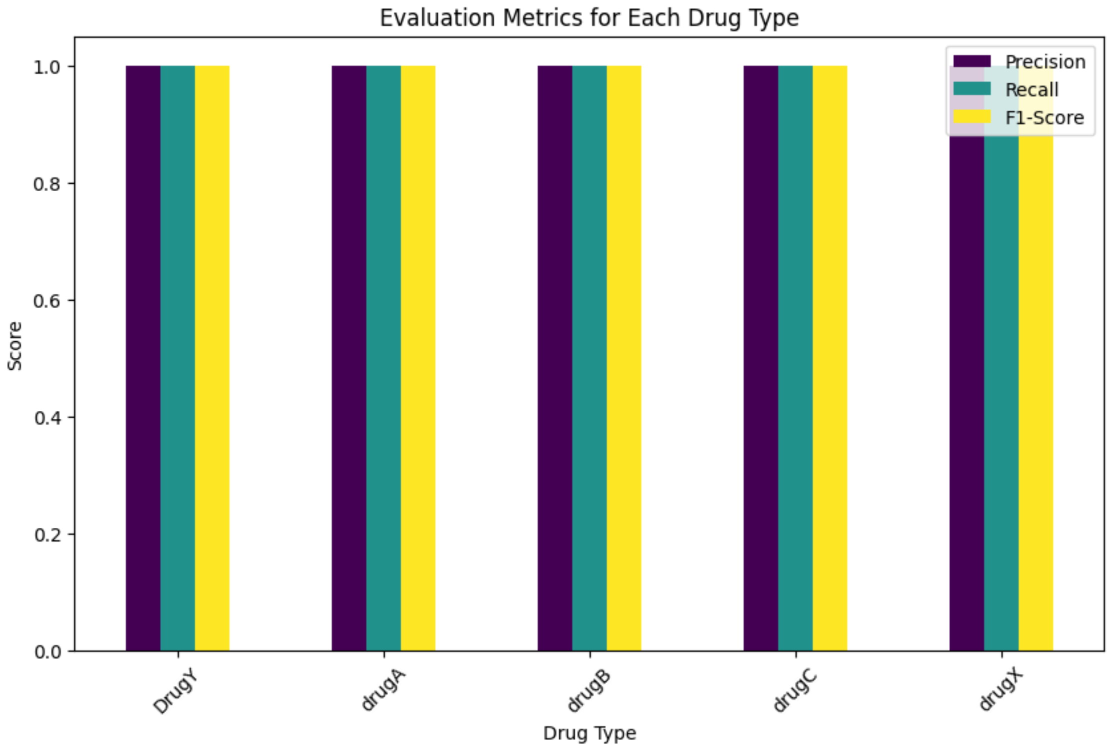
```
    accuracy                           1.00        40
   macro avg       1.00      1.00      1.00        40
weighted avg       1.00      1.00      1.00        40

Confusion Matrix:
[[15  0  0  0  0]
 [ 0  6  0  0  0]
 [ 0  0  3  0  0]
 [ 0  0  0  5  0]
 [ 0  0  0  0 11]]
Feature names used during training: Index(['Age', 'Na_to_K', 'Sex_M', 'BP_LOW',
'BP_NORMAL', 'Cholesterol_NORMAL'], dtype='object')
Predicted Drug Type: drugA
```

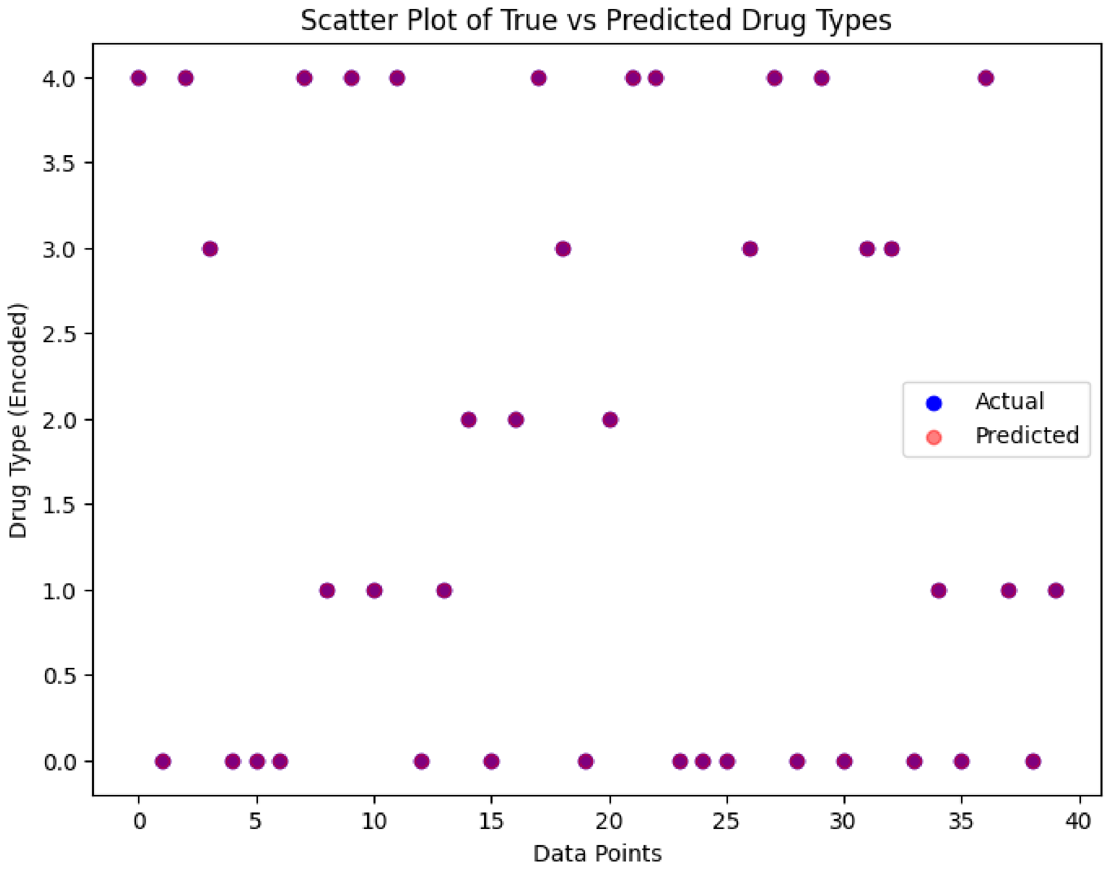
[18]:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import accuracy_score

# 1. Plot Training and Testing Accuracy at Different Numbers of Estimators
train_accuracies = []
test_accuracies = []
n_estimators_range = range(10, 210, 10)

for n in n_estimators_range:
    temp_model = RandomForestClassifier(n_estimators=n, random_state=42)
    temp_model.fit(X_train, y_train)
    train_accuracies.append(accuracy_score(y_train, temp_model.
  ↪predict(X_train)))
    test_accuracies.append(accuracy_score(y_test, temp_model.predict(X_test)))

plt.figure(figsize=(10, 6))
plt.plot(n_estimators_range, train_accuracies, label="Training Accuracy",␣
  ↪marker='o')
plt.plot(n_estimators_range, test_accuracies, label="Testing Accuracy",␣
  ↪marker='o')
plt.xlabel("Number of Estimators")
plt.ylabel("Accuracy")
plt.title("Training vs Testing Accuracy for Different Estimator Counts")
plt.legend()
plt.show()

# 2. Evaluation Metrics Plot (Precision, Recall, F1-score for each class)
from sklearn.metrics import precision_recall_fscore_support

# Get precision, recall, f1-score for each class
precision, recall, f1_score, _ = precision_recall_fscore_support(y_test, y_pred)
```

```
metrics_df = pd.DataFrame({'Precision': precision, 'Recall': recall, 'F1-Score':
 ↪ f1_score},
                          index=label_encoder.inverse_transform(np.unique(y)))

metrics_df.plot(kind='bar', figsize=(10, 6), colormap='viridis')
plt.title("Evaluation Metrics for Each Drug Type")
plt.xlabel("Drug Type")
plt.ylabel("Score")
plt.xticks(rotation=45)
plt.show()

# 3. Scatter Plot of True vs Predicted Data Points
plt.figure(figsize=(8, 6))
plt.scatter(range(len(y_test)), y_test, color="blue", label="Actual")
plt.scatter(range(len(y_pred)), y_pred, color="red", alpha=0.5,␣
 ↪label="Predicted")
plt.xlabel("Data Points")
plt.ylabel("Drug Type (Encoded)")
plt.title("Scatter Plot of True vs Predicted Drug Types")
plt.legend()
plt.show()
```




Evaluation Metrics for Each Drug Type
Score
1.0
0.8
0.6
0.4
0.2
0.0
Precision
Recall
F1-Score
DrugY
drugA
drugB
drugC
drugX
Drug Type


Scatter Plot of True vs Predicted Drug Types
Drug Type (Encoded)
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
5
10
15
20
25
30
35
40
Data Points
● Actual
● Predicted

[ ]: