Artificial Intelligent (Lab)

Task # 11

Submitted To: Sir Rasikh Ali

Submitted By: Shumaila Maryam

Roll no: SU92-BSDSM-F24-062

Section: BSDS-3A

Department of Software Engineering, Superior University, Lahore

Question:

Perform the following steps:

1. Train-test splitting
2. Select and apply the appropriate machine learning model
3. Perform testing/prediction on the test set
4. Display the Accuracy Score

Introduction

This task focuses on the **Model Building and Evaluation** phase of machine learning. After completing preprocessing in Task 10, the cleaned dataset is now ready for training. In this part, I performed several essential machine-learning steps such as splitting the data, training a Random Forest model, evaluating performance using multiple metrics, and finally saving the trained model for future predictions.

Machine learning cannot work properly without structured data and a trained model. Therefore, this task represents an important transition from **data preparation** to **actual predictive modelling**.

Why I Made This Part

The aim of this task is to build a working prediction model using the processed dataset.
The steps performed are important because:

- It splits the dataset to properly test the model's performance.
- It trains a machine-learning algorithm to understand patterns in the data.
- It evaluates accuracy using standard ML metrics such as R^2 Score, MAE, MSE, and RMSE.
- It saves the trained model (model_rf.pkl) so it can be used later without retraining.
- It ensures the model is efficient, accurate, and ready for deployment.

Overall, this task transforms the cleaned dataset from Task-10 into a functional machine-learning model.

How It Works

1. **Importing Required Libraries**

```python
import pandas as pd
import numpy as np
import pickle
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
```

66] ✓ 0.0s Python

2. Loading the Processed Dataset

Next, the cleaned dataset created in Task 10 (processed_data.csv) was loaded.This dataset already has:

- No missing values
- Encoded categorical features
- Correct numeric formats
- Consistent structure

Loading this file is important because only a fully pre-processed dataset can be used for training a machine-learning model effectively. Then i check the info and columns to ensure that the correct file is loaded.

```
import pandas as pd
df = pd.read_csv('processed_data.csv')
df.info()
df.columns
```

[67] ✓ 0.0s Python

3. Splitting Features and Target Variable

To train a model, the data must be divided into:

- **X (Features):** all columns except Price
- **y (Target):** the Price column

This helps the model understand the relationship between laptop specifications (features) and their price (target value).

```
X = df.drop('Price', axis=1)
y = df['Price']
```

68] ✓ 0.0s Python

4. Train-Test Split

The dataset is split into two parts:

- **Training set (80%)** → used to teach the model
- **Testing set (20%)** → used to check model accuracy

Using shuffle=False keeps the order of the rows unchanged.
This is useful when the dataset may have dependency or a pattern in order.
The test set allows us to evaluate how well the model performs on unseen data.

```
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.2, shuffle=False)
```

[69] ✓ 0.0s Python

```
print(train_X.shape, train_y.shape)
print(test_X.shape, test_y.shape)
```

[70] ✓ 0.0s Python

5. Training the Random Forest Model

A **RandomForestRegressor** model was selected because:

- It works well with large datasets
- It handles both numeric and encoded features
- It reduces overfitting by using multiple decision trees
- It provides strong and stable predictions

The model learns relationships between different laptop features and their prices.
For example:

- Higher RAM usually increases price
- SSD size affects cost
- Processor brand and generation influence pricing

This learning happens during the .fit() process.

```
model_rf = RandomForestRegressor(n_estimators=300, max_depth=10, random_state=42)
model_rf.fit(train_X, train_y)
print(model_rf)
```
[71] ✓ 1.3s Python

6. Making Predictions

The model then predicts the price of laptops in the testing dataset:

```
predictions = model_rf.predict(test_X)
model_pred_rf = predictions
print(predictions)
```
✓ 0.0s Python

These predictions allow us to compare the model's output with the actual prices and measure how accurate the model is.

7. Model Evaluation

To check how well the model performs, several evaluation metrics were used:

- **R^2 Score:** Measures how much of the price variation the model can explain
- **MAE:** Shows average difference between predicted and actual prices
- **MSE / RMSE:** Show how large the errors are (RMSE is easier to interpret)

These metrics together give a complete picture of the model's accuracy, strengths, and areas for improvement.
The R^2 score is also turned into a percentage to make interpretation clearer.

```python
r2 = r2_score(test_y, predictions)
mae = mean_absolute_error(test_y, predictions)
mse = mean_squared_error(test_y, predictions)
rmse = np.sqrt(mse)

print(f"R² Score: {r2:.3f}")
print(f"MAE: {mae:.2f}")
print(f"MSE: {mse:.2f}")
print(f"RMSE: {rmse:.2f}")
```
[73] ✓ 0.0s Python

8. Accuracy of model

For accuracy takes the R^2 score, treats it like a percentage of how much variance the model explains, and prints it neatly as a "model accuracy" value.

```python
accuracy = r2 * 100
print(f"Accuracy: {accuracy:.2f}%")
```
[76] ✓ 0.0s Python

```
Accuracy: 97.04%
```

So, my model is 97% accurate.

9. Saving the Model

Finally, the trained model is saved as model_rf.pkl using pickle.
This allows us to:

- Reuse the model later
- Make predictions instantly
- Avoid retraining the model every time

The saved model is tested by loading it again to confirm that it works properly.

```python
pickle.dump(model_rf, open('model_rf.pkl', 'wb'))
```
74] ✓ 0.0s Python

10. Loading the Saved Model

To confirm that the saved model works correctly, it was loaded again using:

```python
model_rf = pickle.load(open('model_rf.pkl', 'rb'))
```
[75] ✓ 0.0s Python

This step ensures the model can be used in future applications (like prediction systems, web apps, or further tasks) without needing to retrain it.