Day51_Logistic_Regression_Unseen_Data_Prediction

July 24, 2025

Day 51 – Logistic Regression on Future Data

Continuing from Day 50 — Logistic Regression classification with Age and Salary.

1 Part 1: Rebuild Model from Day 50 + Predict Future Data

In Day 50, we trained a Logistic Regression model using `StandardScaler` and `random_state = 0`.

Today in **Day 51**, we're taking it one step further:

1. Rebuilding the model from scratch (same settings as Day 50)
2. Testing it on **new, unseen future data**
3. Saving the model and scaler using `pickle`
4. Creating a **real-time Streamlit app** for prediction

This simulates how models work **after deployment** — when real-world users provide new data!

Let's begin

1.1 Import Required Libraries

[1]:
```
# Step 1: Import Required Libraries
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
import pickle
```

1.2 Load Dataset (used in Day 50)

[2]:
```
# Step 2: Load Dataset (used in Day 50)
dataset = pd.read_csv(r"C:\Users\Lenovo\Downloads\logit classification.csv") #␣
 ↪Make sure it's in your working directory

# Show top rows
dataset.head()
```

[2]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

1.3 Feature Selection

[3]:
```
# Step 3: Feature Selection
X = dataset[["Age", "EstimatedSalary"]].values
y = dataset["Purchased"].values
```

1.4 Train-Test Split (Same as Day 50)

[5]:
```
# Step 4: Train-Test Split (Same as Day 50)
X_train2, X_test2, y_train2, y_test2 = train_test_split(X, y, test_size=0.25,␣
↪random_state=0)
```

1.5 Apply StandardScaler (Same as Day 50)

[6]:
```
# Step 5: Apply StandardScaler (Same as Day 50)
scaler2 = StandardScaler()
X_train2 = scaler2.fit_transform(X_train2)
X_test2 = scaler2.transform(X_test2)
```

1.6 Train the Logistic Regression Model (Same as Day 50)

[8]:
```
# Step 6: Train the Logistic Regression Model (Same as Day 50)
model2 = LogisticRegression()
model2.fit(X_train2, y_train2)
```

[8]:
```
LogisticRegression()
```

[9]:
```
# Predict on test set
y_pred2 = model2.predict(X_test2)
```

[10]:
```
# Accuracy & Confusion Matrix
print("Accuracy:", accuracy_score(y_test2, y_pred2))
print("Confusion Matrix:\n", confusion_matrix(y_test2, y_pred2))
```

```
Accuracy: 0.89
Confusion Matrix:
 [[65  3]
 [ 8 24]]
```

2 Now Predict Future Data

2.1 Upload or Create Future Data

[11]:
```
# Step 7: Create Future Data
future_data = {
    "User ID": [15724611, 15725621, 15725622, 15720611, 15588044,
                15746039, 15704887, 15746009, 15876009, 15886009],
    "Gender": ["Male", "Female", "Male", "Female", "Male",
               "Female", "Male", "Female", "Male", "Female"],
    "Age": [45, 79, 23, 34, 29, 70, 86, 46, 32, 100],
    "EstimatedSalary": [60000, 64000, 78000, 45000, 76000,
                        89000, 120000, 23000, 70000, 90000]
}

future_df = pd.DataFrame(future_data)

# or uploade
# future_df = pd.read_csv("future_test_data.csv")
```

2.2 Select Required Columns & Scale

[12]:
```
# Step 8: Select Required Columns & Scale
X_future = future_df[["Age", "EstimatedSalary"]]
X_future_scaled = scaler2.transform(X_future)
```

```
C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\utils\validation.py:2742:
UserWarning: X has feature names, but StandardScaler was fitted without feature
names
  warnings.warn(
```

2.3 Predict

[13]:
```
# Step 9: Predict
y_pred_future = model2.predict(X_future_scaled)
future_df["y_pred1"] = y_pred_future
```

2.4 Save Result to CSV

[14]:
```
# Step 10: Save Result to CSV
future_df.to_csv("Day51_Future_Predictions.csv", index=False)

# Show result
future_df
```

[14]:

	User ID	Gender	Age	EstimatedSalary	y_pred1
0	15724611	Male	45	60000	1
1	15725621	Female	79	64000	1

```
2  15725622    Male   23            78000        0
3  15720611  Female   34            45000        0
4  15588044    Male   29            76000        0
5  15746039  Female   70            89000        1
6  15704887    Male   86           120000        1
7  15746009  Female   46            23000        0
8  15876009    Male   32            70000        0
9  15886009  Female  100            90000        1
```

3 Save Model and Scaler for Streamlit App

[15]:
```python
# Save the trained model to a file
with open("model2.pkl", "wb") as f:
    pickle.dump(model2, f) # Saves logistic regression model

# Save the scaler used during training
with open("scaler2.pkl", "wb") as f:
    pickle.dump(scaler2, f) # Saves StandardScaler object
```

4 Part 2: Streamlit Real-Time Prediction App

Now let's create a Streamlit frontend to interactively predict outcomes using the saved model and scaler.

Save the following code as day51_logistic_app.py

```python
import streamlit as st
import numpy as np
import pickle

# Load the trained model and scaler
with open("model2.pkl", "rb") as f:
    model2 = pickle.load(f)

with open("scaler2.pkl", "rb") as f:
    scaler2 = pickle.load(f)

#  Updated Title
st.title(" Logistic Regression on Unseen Data")

st.markdown("###  Enter Details Below")

#  Manual Inputs
age = st.number_input("Enter Age", min_value=18, max_value=100, value=30, step=1)
salary = st.number_input("Enter Estimated Salary", min_value=10000, max_value=150000, value=50

# Predict Button
```

```
if st.button("Predict"):
    user_input = np.array([[age, salary]])
    scaled_input = scaler2.transform(user_input)
    prediction = model2.predict(scaled_input)

    if prediction[0] == 1:
        st.success("  Prediction: Will Purchase")
    else:
        st.error("  Prediction: Will NOT Purchase")
```

4.1 To Run the Streamlit App:

```
streamlit run day51_logistic_app.py
```

5 Summary

Today, we achieved an important milestone in the machine learning journey:

Rebuilt the logistic regression model from Day 50
Predicted outcomes on new future user data
Saved the trained model (`model2.pkl`) and scaler (`scaler2.pkl`)
Built a real-time prediction app using **Streamlit**

This is how machine learning is used in the real world — not just training, but testing, saving, and deploying!

Next Steps (Optional Ideas):

- Include **gender** in model using one-hot encoding
- Add **CSV upload option** in Streamlit for bulk prediction
- Try different models like **Random Forest** or **XGBoost**
- Host the Streamlit app online using **Streamlit Cloud**

Keep going strong!

Thank you !

AB

[ ]: