Data Science:

📂 Final Project Structure

```
Ecommerce-Model-Deployment/
│── data/
│   └── Realistic_E-Commerce_Dataset.csv
│
│── src/
│   ├── preprocess.py
│   ├── train_model.py
│   ├── evaluate.py
│
│── models/                # generated after training
│   ├── ecommerce_model.pkl
│   ├── scaler.pkl
│   └── feature_names.pkl
│
│── app/
│   └── streamlit_app.py     # Streamlit UI
│
│── requirements.txt
│── README.md
```

📄 1. src/preprocess.py

```
# src/preprocess.py
"""
Data Preprocessing for E-Commerce Dataset
```

```
- Removes IDs and date
- Encodes categorical variables
- Scales numerical features
- Returns feature names for deployment
"""

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder

def load_and_preprocess(filepath):
    # Load dataset
    df = pd.read_csv(filepath)

    # Drop irrelevant columns
    df = df.drop(columns=["user_id", "session_id", "date"])

    # Encode categorical features
    cat_cols = df.select_dtypes(include=["object"]).columns
    for col in cat_cols:
        le = LabelEncoder()
        df[col] = le.fit_transform(df[col])

    # Split features/target
    X = df.drop(columns=["purchase"])
    y = df["purchase"]

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

    # Scale
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)
```

```
    return X_train_scaled, X_test_scaled, y_train, y_test, scaler, X.columns.tolist
()
```

📄 2. src/train_model.py

```
# src/train_model.py
"""
Train a RandomForest model and save model, scaler, and feature names
"""

import joblib
from sklearn.ensemble import RandomForestClassifier
from preprocess import load_and_preprocess

def train_and_save_model():
    (
        X_train, X_test, y_train, y_test,
        scaler, feature_names
    ) = load_and_preprocess("data/Realistic_E-Commerce_Dataset.csv")

    # Train model
    model = RandomForestClassifier(n_estimators=200, random_state=42)
    model.fit(X_train, y_train)

    # Save model artifacts
    joblib.dump(model, "models/ecommerce_model.pkl")
    joblib.dump(scaler, "models/scaler.pkl")
    joblib.dump(feature_names, "models/feature_names.pkl")

    print("✅ Model, scaler, and feature names saved!")

if __name__ == "__main__":
```

```
    train_and_save_model()
```

📄 3. src/evaluate.py

```
# src/evaluate.py
"""
Evaluate trained ML model
"""

import joblib
from preprocess import load_and_preprocess
from sklearn.metrics import classification_report, confusion_matrix, accuracy_
score

def evaluate_model():
    (
        X_train, X_test, y_train, y_test,
        scaler, feature_names
    ) = load_and_preprocess("data/Realistic_E-Commerce_Dataset.csv")

    # Load model
    model = joblib.load("models/ecommerce_model.pkl")

    # Predictions
    y_pred = model.predict(X_test)

    # Metrics
    print("📊 Model Performance:\n")
    print(f"Accuracy: {accuracy_score(y_test, y_pred):.4f}")
    print(classification_report(y_test, y_pred))
    print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))

if __name__ == "__main__":
```

```
evaluate_model()
```

📄 4. app/streamlit_app.py

```
# app/streamlit_app.py
"""
Streamlit App for E-Commerce Purchase Prediction
Run: streamlit run app/streamlit_app.py
"""

import streamlit as st
import joblib
import pandas as pd

# ---------------------
# Load model, scaler & feature names
# ---------------------
model = joblib.load("models/ecommerce_model.pkl")
scaler = joblib.load("models/scaler.pkl")
feature_names = joblib.load("models/feature_names.pkl")

st.set_page_config(page_title="E-Commerce Purchase Prediction", layout="c
entered")
st.title("🛒 E-Commerce Purchase Prediction")
st.write("Predict whether a customer will purchase based on session details.")

# ---------------------
# Sidebar inputs for ALL features
# ---------------------
st.sidebar.header("Customer Session Details")

age = st.sidebar.slider("Age", 18, 70, 30)
gender = st.sidebar.selectbox("Gender", ["Male", "Female"])
```

```
location = st.sidebar.selectbox("Location", ["USA", "UK", "Germany", "Canad
a", "Australia"])
membership_status = st.sidebar.selectbox("Membership Status", ["Guest", "R
egistered"])
returning_customer = st.sidebar.selectbox("Returning Customer", [0, 1])
device_type = st.sidebar.selectbox("Device Type", ["Mobile", "Desktop", "Tabl
et"])
browser = st.sidebar.selectbox("Browser", ["Chrome", "Firefox", "Safari", "Edg
e"])
time_of_day = st.sidebar.selectbox("Time of Day", ["Morning", "Afternoon", "E
vening", "Night"])

time_spent = st.sidebar.slider("Time Spent (minutes)", 1, 120, 30)
pages_viewed = st.sidebar.slider("Pages Viewed", 1, 50, 10)
scroll_depth = st.sidebar.slider("Scroll Depth (%)", 0, 100, 50)
clicks = st.sidebar.slider("Clicks", 0, 100, 10)

traffic_source = st.sidebar.selectbox("Traffic Source", ["Organic", "Social", "R
eferral", "Paid"])
ad_campaign = st.sidebar.selectbox("Ad Campaign", ["Campaign_A", "Campai
gn_B", "Campaign_C"])
coupon_used = st.sidebar.selectbox("Coupon Used", [0, 1])
discount_applied = st.sidebar.selectbox("Discount Applied", [0, 1])
product_category = st.sidebar.selectbox("Product Category", ["Clothing", "Ele
ctronics", "Home", "Books", "Sports"])
wishlist_items = st.sidebar.slider("Wishlist Items", 0, 20, 2)
cart_items = st.sidebar.slider("Cart Items", 0, 10, 1)
avg_session_value = st.sidebar.slider("Average Session Value", 0, 1000, 100)
payment_method = st.sidebar.selectbox("Payment Method", ["UPI", "Debit Ca
rd", "Credit Card", "NetBanking", "COD"])

# ---------------------
# Build input dict
# ---------------------
features = {
    "age": age,
```

```
    "gender": gender,
    "location": location,
    "membership_status": membership_status,
    "returning_customer": returning_customer,
    "device_type": device_type,
    "browser": browser,
    "time_of_day": time_of_day,
    "time_spent_minutes": time_spent,
    "pages_viewed": pages_viewed,
    "scroll_depth": scroll_depth,
    "clicks": clicks,
    "traffic_source": traffic_source,
    "ad_campaign": ad_campaign,
    "coupon_used": coupon_used,
    "discount_applied": discount_applied,
    "product_category": product_category,
    "wishlist_items": wishlist_items,
    "cart_items": cart_items,
    "avg_session_value": avg_session_value,
    "payment_method": payment_method,
}

input_df = pd.DataFrame([features])

# ---------------------
# Encode categoricals
# ---------------------
cat_cols = input_df.select_dtypes(include=["object"]).columns
for col in cat_cols:
    input_df[col] = input_df[col].astype("category").cat.codes

# ✅ Reindex to match training feature order
input_df = input_df.reindex(columns=feature_names)

# ---------------------
# Predict
```

```
# ---------------------
if st.button("Predict"):
    scaled_input = scaler.transform(input_df)
    pred = model.predict(scaled_input)[0]
    prob = model.predict_proba(scaled_input)[0][1] * 100

    if pred == 1:
        st.success(f"✅ Customer is likely to purchase! (Confidence: {prob:.2
f}%)")
    else:
        st.warning(f"❌ Customer is unlikely to purchase. (Confidence: {prob:.2
f}%)")

st.markdown("---")
st.caption("Built with ❤️ using Streamlit | Model: RandomForestClassifier")
```

📄 5. requirements.txt

```
pandas
numpy
scikit-learn
streamlit
joblib
```

📄 6. README.md (for Students)

```
# 🛒 E-Commerce Purchase Prediction

This project trains a machine learning model to predict whether a customer wi
ll purchase based on their session details. The trained model is deployed usin
```

g Streamlit.

🚀 How to Run

1. Clone the repo:
   ```bash
   git clone <repo-url>
   cd Ecommerce-Model-Deployment
   ```

1. Install dependencies:

   ```
   pip install -r requirements.txt
   ```

2. Train model:

   ```
   python src/train_model.py
   ```

3. Evaluate model (optional):

   ```
   python src/evaluate.py
   ```

4. Run Streamlit app:

   ```
   streamlit run app/streamlit_app.py
   ```

📊 Model Output

- **0** → Customer will not purchase

- **1** → Customer will purchase
- Probability score (%) gives confidence in prediction