Ad Click-Through Rate (CTR) Prediction – Project Summary

1. What is the Project About?

- **Goal:** Predict whether a user will click on an online advertisement before showing it to them.
- **Problem:** Ads are expensive to display; showing irrelevant ads wastes money. Predicting clicks helps target the right users.
- **Solution:** Train a **machine learning model** (Logistic Regression in your case) to predict the likelihood of a click based on user behavior and demographic data.

---

2. Dataset Overview

- **Source:** Example dataset with user behaviors and demographics.
- **Features:**
    1. **Daily Time Spent on Site** – Time user spends on the website.
    2. **Age** – Age of the user.
    3. **Area Income** – Average income in the user's location.
    4. **Daily Internet Usage** – Total time spent on the Internet daily.
    5. **Gender** – Male or Female.
    6. **Clicked on Ad** – Target variable (1 if clicked, 0 if not).
- **Exploration & Understanding:**
    - Checked for **missing values** and data consistency.
    - Analyzed **target distribution**: Most users didn't click (imbalanced), so model needs careful interpretation.
    - Observed **feature distributions** and relationships with clicks.

---

3. Why These Features?

- Behavioral features (time on site, internet usage) indicate user engagement.
- Demographic features (age, income, gender) help model preferences and targeting.
- **Target variable:** Clicked on Ad – Binary (Yes/No).

---

4. Model Selection & Reasoning

- **Logistic Regression:**
    - Simple and interpretable.
    - Suitable for **binary classification** problems like click/no-click.

    - Provides probability outputs (useful to rank users by likelihood to click).
- **Why not complex models?**
    - Initially, the dataset is small (~8k samples).
    - Logistic Regression gives a good baseline and is fast for prototyping.

---

5. Data Preprocessing

- **Encoding:**
    - Gender: Male = 1, Female = 0.
- **Scaling:**
    - Features normalized using a **StandardScaler** for better model performance.
- **Removed extra complexity:**
    - Initially, the code included placeholders for "country features," but these were unnecessary for a simple model.

---

6. Model Training

- Split data into **train and test sets**.
- Trained **Logistic Regression** on scaled features.
- Evaluated using:
    - **Accuracy**
    - **Confusion matrix**
    - **ROC-AUC (optional)**

---

7. Challenges Faced

1. **Unnecessary feature handling:**
    - Original code added zeroed "country" features for each input, causing confusion.
    - **Solution:** Removed these features entirely for simplicity.
2. **Scaling and shape issues:**
    - Logistic Regression requires numeric and properly scaled features.
    - **Solution:** Applied StandardScaler to input features and ensured input is 2D ([[...]]).
3. **Predicting on new data:**
    - Needed a consistent format between training data and new inputs.

- **Solution:** Converted new input into a NumPy array, scaled it, then predicted click probability.

4. **Categorical encoding:**
    - Gender needed to be numeric.
    - **Solution:** Used simple 0/1 encoding.

---

8. Prediction Workflow

Function Example:

```
def predict_ad_click(Daily_Time, Age, Area_Income, Daily_Internet_Usage, Gender):
    gender_numeric = 1 if Gender.lower() == 'male' else 0
    features = np.array([[Daily_Time, Age, Area_Income, Daily_Internet_Usage, gender_numeric]])
    features_scaled = scaler.transform(features)
    click_prob = model.predict_proba(features_scaled)[0][1] * 100
    return click_prob
```

- **Input:** User features (time, age, income, internet usage, gender)
- **Output:** Probability (%) of clicking the ad.

---

9. Streamlit App

- Created a **simple web interface** to:
    - Input user features.
    - Display **predicted click probability**.
- Used **joblib** to load the saved model and scaler.
- **Caching** applied to avoid reloading model each time (@st.cache_resource).

---

10. Key Takeaways

- Logistic Regression works well for **binary outcomes**.
- Proper **preprocessing** (encoding, scaling) is crucial.
- Keep the model **simple and interpretable** initially; extra features can confuse or break the pipeline.
- Always ensure **input shape and type** match training data.

---

11. Quick Recall Map

- **What:** Predict ad clicks before showing ads.
- **Why:** Target right users → save money → increase engagement.
- **How:** Logistic Regression → preprocess features → scale → predict probability.
- **Input:** Time on site, Age, Income, Internet Usage, Gender.
- **Output:** Click probability (0–100%).
- **Problems:** Extra features, scaling, encoding → solved by simplifying input and scaling correctly.
- **Tools:** Python, Pandas, NumPy, Scikit-learn, Streamlit, Joblib.