Midterm Project Presentation

Title Slide

Title - Ad Click Prediction

Name – Paneri Patel

**Institute – Data Science Institute**

Date of Presentation – October 25th, 2024

Github –

Intro Slide –

Problem i want to solve – I am to predict whether a user will click on an online ad based on their demographics, browsing behavior, the context of the ad's display, and the time of day.

Why is it important – Understanding a user’s behavior allows advertisers to better market their products targeting right audience.

Classification/Regression – Classification Problem, target variable click is binary.(Categorical feature)

Where i got the data- I got the data from Kaggle website, https://www.kaggle.com/datasets/marius2303/ad-click-prediction-dataset

**How the data was collected – openAI generated**

Understanding Data -

10,000 entries

9 columns

Features

Continous Feature

age: Age of the user (ranging from 18 to 64 years).

Categorical Features

id: Unique identifier for each user.

full\_name: User's name formatted as "UserX" for anonymity.

gender: The gender of the user (categorized as Male, Female, or Non-Binary).

device\_type: The type of device used by the user when viewing the ad (Mobile, Desktop, Tablet).

ad\_position: The position of the ad on the webpage (Top, Side, Bottom).

browsing\_history: The user's browsing activity prior to seeing the ad (Shopping, News, Entertainment, Education, Social Media).

time\_of\_day: The time when the user viewed the ad (Morning, Afternoon, Evening, Night).

click: The target label indicating whether the user clicked on the ad (1 for a click, 0 for no click).

EDA –

Overall, how many users have clicked on the advertisement.

Graph

Why it's important: This gives a quick snapshot of the distribution of your target variable, helping to understand the overall engagement.

Last slide EDA –

Continuous Feature(Age) vs Target Variable

Graph

Why it's important: This helps in understanding how age influences ad click behavior and whether there's a particular age group more likely to engage.

EDA –

Categorical Feature(Browsing History) vs Target Variable

Graph

Why it's important: This will help identify which types of browsing behavior are more likely to lead to ad clicks, which can be critical for targeting ads.

EDA –

Categorical Feature(Time of the day) vs Target Variable

Graph

Why it's important: This can help reveal trends in user activity and ad engagement based on the time of day, helping advertisers optimize the timing of their campaigns.

EDA –

Categorical Feature(Age) vs Categorical Feature(Browsing History)

Graph

Why it's important: This can reveal patterns in how users of different age groups engage with different types of content, which could inform ad strategies that are more specific to certain age demographics.

EDA, Summary of Key Insights: (change according to graphs)

Ad Engagement: 65% of users clicked the ad, but exploring other factors revealed deeper insights.

Age Influence: Younger users (mid-20s to 30s) are more likely to click on ads compared to older users.

Browsing History: Users engaging with shopping and social media content show the highest likelihood of clicking ads, while educational content results in lower engagement.

Time of Day: Afternoon and evening are the peak times for ad engagement, with users being less likely to engage in the morning.

Age vs. Browsing Behavior: Younger users are more focused on shopping and social media, while older users browse educational and news content.

**Calculate summary statistics?**

Splitting –

Split Non-IID Data Based on Group ID:

Your dataset is non-IID because the data points are not identically distributed—certain groups (based on age, browsing behavior, time, etc.) have different probabilities of clicking on an ad.

Reason: Your dataset likely contains some form of grouping structure (e.g., user browsing behavior, device type, time of day, etc.). If certain groups of users exhibit similar behavior, splitting based on group ID ensures that the same group is not represented in both training and test sets. This prevents data leakage and gives a more accurate estimate of your model's performance on unseen data.

Application: Use a **GroupKFold or GroupShuffleSplit** approach. For example, if you have a user ID or some group structure (e.g., users with similar browsing habits), you should ensure that the same group doesn’t appear across both the training and validation/test sets.

**Stratify**

Preprocessing -

**Do we clean/drop/mean/median null valus?**

One-Hot Encoding:

Categorical features (gender, browsing\_history, device\_type, ad\_position, time\_of\_day) are converted to numerical values using OneHotEncoder.

Scaling the age Column:

The age column is scaled between 0 and 1 using MinMaxScaler.

Handling Missing Values:

We apply SimpleImputer to fill in missing values for categorical and numerical features. Categorical missing values are filled with the most frequent category, and numerical missing values are filled with the mean.

ColumnTransformer:

This helps apply different preprocessing steps to numerical and categorical columns simultaneously.

Preprocessors Used and Why:

**categorical\_imputer = SimpleImputer(strategy='constant', fill\_value='Unknown')**

**numerical\_imputer = SimpleImputer(strategy='mean')**

**how to print features before n after handling in preprocessing?**

**Individually which type of simple imputer to use for different columns**

SimpleImputer:

Why: We used a SimpleImputer to handle missing values in both numerical and categorical features. Missing values can cause issues during model training, so it's essential to handle them.

For Numerical Features (age): We imputed missing values using the mean of the column. This is appropriate for continuous data where the average value can represent missing entries.

For Categorical Features (gender, browsing\_history, device\_type, ad\_position, time\_of\_day): We used the most frequent category to impute missing values. This is a common strategy for categorical data when you want to fill in missing values with the most common value in the feature.

OneHotEncoder:

Why: We applied OneHotEncoder to transform categorical variables into numerical representations. Machine learning models typically cannot work with categorical data directly, so one-hot encoding is used to convert categories into binary columns.

Configuration: We set handle\_unknown='ignore' to ensure that the model doesn't crash if it encounters unseen categories in the test data. We also set sparse\_output=False to get a dense output that’s easier to work with.

MinMaxScaler:

Why: We used MinMaxScaler to scale the age column, which ensures that the values are scaled between 0 and 1. This helps the model process numerical data efficiently, especially for algorithms that are sensitive to feature scaling (e.g., gradient-based models like logistic regression).

Data Before and After Preprocessing:

Before Preprocessing:

Number of Features: Initially, we had 6 features (1 numerical feature age, and 5 categorical features: gender, browsing\_history, device\_type, ad\_position, time\_of\_day).

Number of Data Points (Rows): We had 10,000 data points.

After Preprocessing:

Number of Features: After one-hot encoding and scaling, the dataset expanded to 14 features:

1 numerical feature (age) after scaling.

13 categorical features after one-hot encoding (each category was transformed into binary columns).

Number of Data Points (Rows): We still have 10,000 data points after preprocessing, as no rows were removed.

Properties of Missing Values:

Fraction of Points with Missing Values:

The dataset contains missing values in multiple columns. Here’s a breakdown:

gender: 4693 missing values (~46.93%)

browsing\_history: 4782 missing values (~47.82%)

device\_type: 2000 missing values (~20%)

ad\_position: 2000 missing values (~20%)

time\_of\_day: 2000 missing values (~20%)

Fraction of Features with Missing Values:

5 out of the 6 original features had missing values, meaning about 83% of the features had missing values.

Handling Missing Values:

We applied mean imputation for the age column and most frequent imputation for the categorical columns. These imputation methods are commonly used when there is no strong reason to believe that missing values are random or systematically biased in any specific way.