Data Cleaning & Preprocessing

- In terms of cleaning data, checking on ***missing values*** and ***duplicate*** rows are done
    - No duplicates detected after running the code below.
    - For missing data:
        - Daily Time Spent on Site - fill with mean
        - Daily Internet Usage - fill with mean
        - Area Income - fill with median
        - Male - fill with 'Laki-Laki' as to match the number with 'Perempuan'
    - Column 'Male' is renamed 'Sex' to avoid confusion.

```
duplicate_row = df[df.duplicated(keep=False)]
nan_rows = df[df.isna().any(axis=1)]
df['Daily Time Spent on Site'] = df['Daily Time Spent on Site'].fillna(df['Daily Time Spent on
Site'].mean())
df['Daily Internet Usage'] = df['Daily Internet Usage'].fillna(df['Daily Internet Usage'].mean())
df['Area Income'] = df['Area Income'].fillna(df['Area Income'].median())
df['Male'] = df['Male'].fillna('Laki-Laki')
df = df.rename(columns={'Male': 'Sex'})
```

For more information, see jupyter notebook here

Data Cleaning & Preprocessing

- In terms of **extracting datetime data**, new features to get, year, month, week, and day are engineered as below:

```
df['Timestamp'] = pd.to_datetime(df['Timestamp'])


df['year'] = df['Timestamp'].dt.year
df['month'] = df['Timestamp'].dt.month
df['week'] = df['Timestamp'].dt.isocalendar().week
df['day'] = df['Timestamp'].dt.day
```

- df[‘Timestamp’] then dropped for machine learning purposes as it is an object type of feature.

```
df = df.drop('Timestamp', axis=1)
```

For more information, see jupyter notebook here

Data Cleaning & Preprocessing

- Next for data preprocessing, ***feature encoding*** and ***feature standardisation*** are done.
    - Feature encoding for categorical features include:
        - Label encoding for ordinal data (Sex, Clicked on Ad)
        - One hot encoding for non-ordinal data (city, province, category)
    - Feature standardisation using StandardScaler and MinMaxScaler, features has been checked for standardised
        - 'Daily Time Spent on Site' and 'Daily Internet Usage' roughly normal but slightly skewed, standard scaling (Z-score normalization) is used.
        - 'Area Income' is rightly skewed, log transformation followed by standard scaling, which can help reduce the skewness before standardization.
        - Age' to be scaled with Min-Max scaling as it will preserve the relative differences while scaling all values to a 0-1 range.

For more information, see jupyter notebook here

Data Cleaning & Preprocessing

- Next for data preprocessing is to Split Data for train and test using the code below:

```
from sklearn.model_selection import train_test_split

X = df.drop('Clicked on Ad', axis=1)
y = df['Clicked on Ad']
# 'Clicked on Ad' value count
print("Distribution of target 'Clicked on Ad'")
print(y.value_counts())

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

For more information, see jupyter notebook here