Click-Through-Rate-Prediction

Mahan Madani – 99222092

1. Overview

This dataset is comprised of 10000 records of users interacting with online advertisements, containing information in 10 different columns. It includes both numerical and non-numerical columns.

The goal of this project is to perform Exploratory Data Analysis, preprocess the data, perform feature engineering, and train various classical models to predict whether or not a user would click on an online ad.

2. Dataset Exploration & Analysis

The dataset stores 10 different attributes. Each record in the dataset belongs to one specific user interaction.

Below you can find a list of all the attributes:

- **Daily Time Spent on Site:** The amount of time the user has spent on the target website
- **Age:** The age of the user
- **Area Income:** The average income of the user's area of residence
- **Daily Internet Usage:** The average amount of time the user spends online
- **Ad Topic Line:** The headline or subject line of the advertisement
- **City:** The name of the city where the user resides in
- **Gender:** The gender of the user (binary, Male/Female)
- **Country:** The name of the country where the user resides in
- **Timestamp:** The exact time and date of the record
- **Clicked on Ad:** The target variable (binary)

3. Data Preprocessing

3.1 Check for Duplicate Records:

To ensure the dataset contains no duplicate records, a combination of all features can be used. If two or more records share the same value in every single column that means they are potentially duplicates.

Even if the records aren't completely identical, they may still contradict each other and need to be handled. **After testing the dataset, it seems that it contains 341 duplicate records.** All instances of duplicate records except the first instance of each one must be dropped.

3.2 Handle Null Values:

As seen in this table, none of the dataset's attributes contain null or missing values.

Therefore, no data imputation is required for this dataset.

	Null Count
Daily Time Spent on Site	0
Age	0
Area Income	0
Daily Internet Usage	0
Ad Topic Line	0
City	0
Gender	0
Country	0
Timestamp	0
Clicked on Ad	0

3.3 Detect Outlier Values:

Using the z-score method, outlier data can be identified. Outliers can be detected from all the numerical features. A total of 122 outlier records were found which must be dropped to reduce processing errors and the chance of skewed data affecting the model.

4. Visualization

Figure 4.1: Clicked on Ad Pie Chart

This plot displays a pie chart comparing the percentage of records where the ad was clicked on versus the opposite. The two labels appear almost equally, and this means we're working with a balanced dataset.

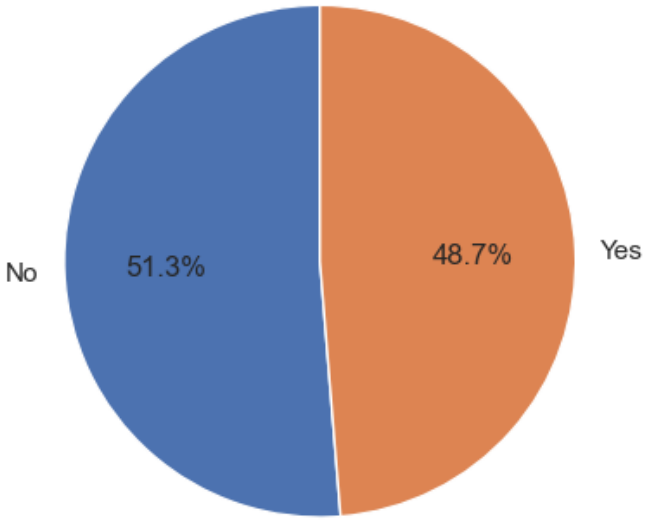


Figure 4.2: Bar plot of ads clicked on based on Gender

This plot displays the number of users clicking on ads based on gender. It is clear that female users are more likely to click on ads on average.

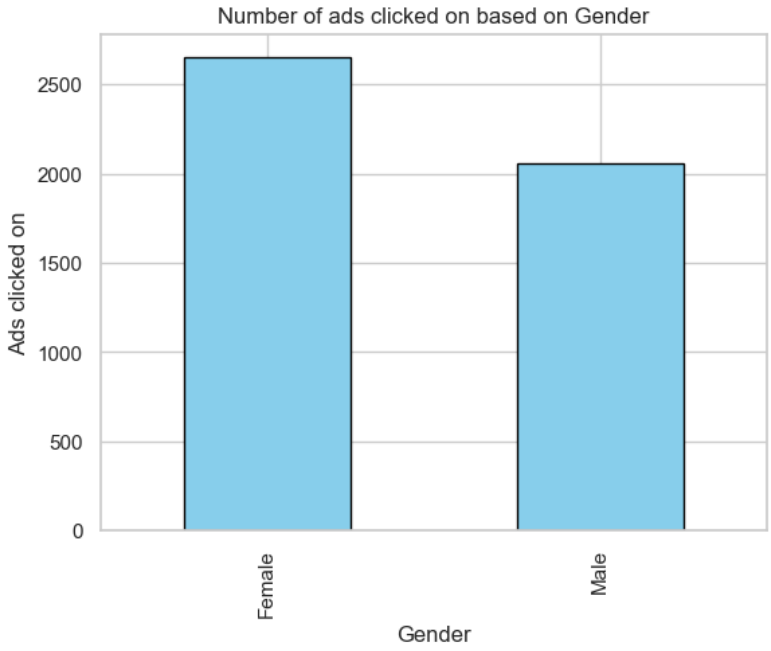


Figure 4.3: Bar plot of the top 10 countries with the highest number of clicks

As seen in the bar plot, users from certain countries are more likely to click on ads. Turkey, The Czech Republic, and Australia seem to have the highest amount of these records.

Based on this plot, we know that we must use the `Country` feature for training our models.

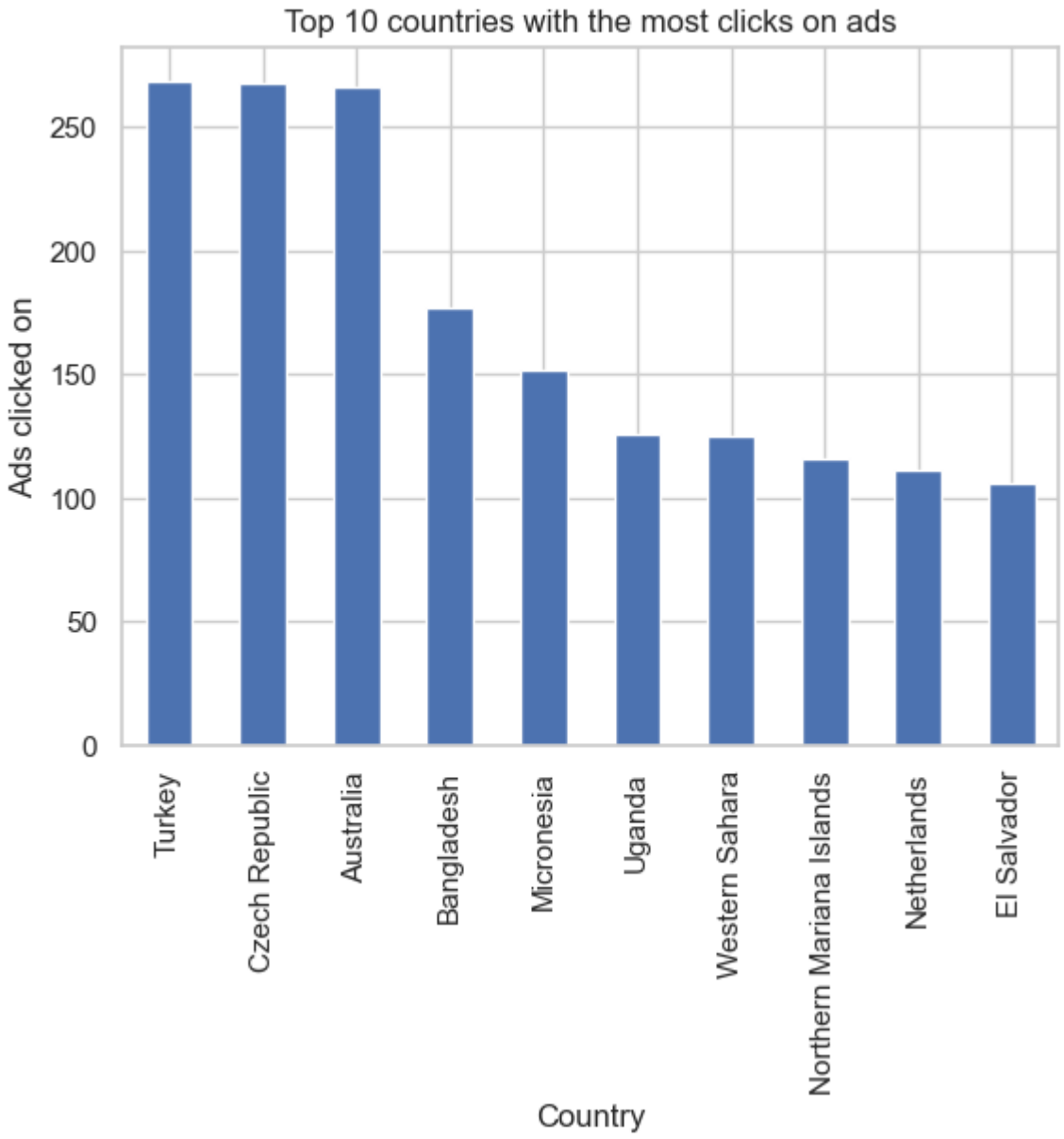


Figure 4.4: Density plot of Daily Internet Usage

While `Daily Internet Usage` doesn't seem to completely follow a normal distribution, it is rather close to it. The majority of internet users seem to stay online between 100 and 250 minutes per day.

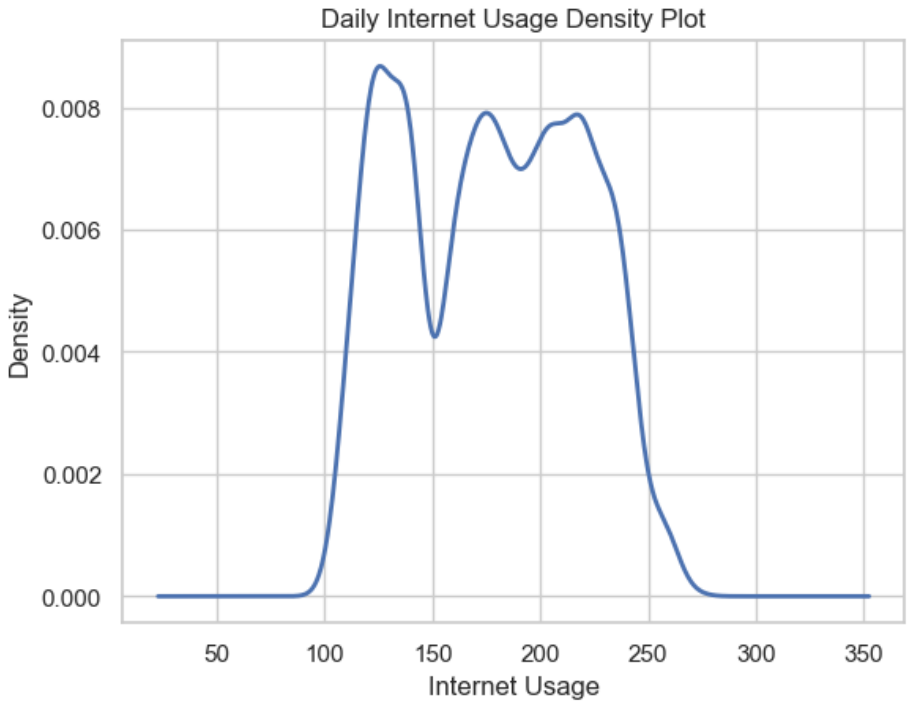


Figure 4.6: Histogram of age

Similar to the previous plot, the users' age histogram follows an almost normal distribution. The majority of users seem to be between 24 and 42 years old.

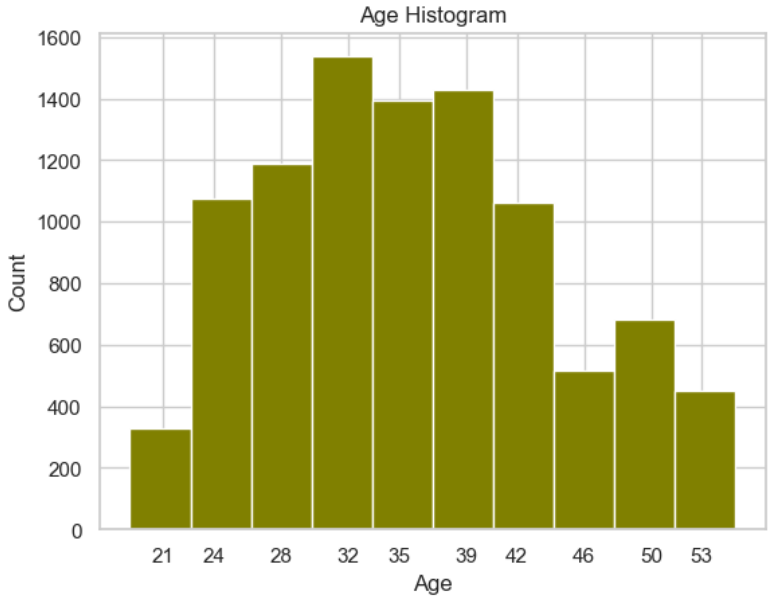


Figure 4.7: Scatter plot of daily time spent on site versus Age

This plot depicts the relations between `Age`, `Daily Time Spent on Site`, and `Clicked on Ad`. We can see that older users (>35) are more likely to click on ads. There is no significant relationship between internet usage and age, however.

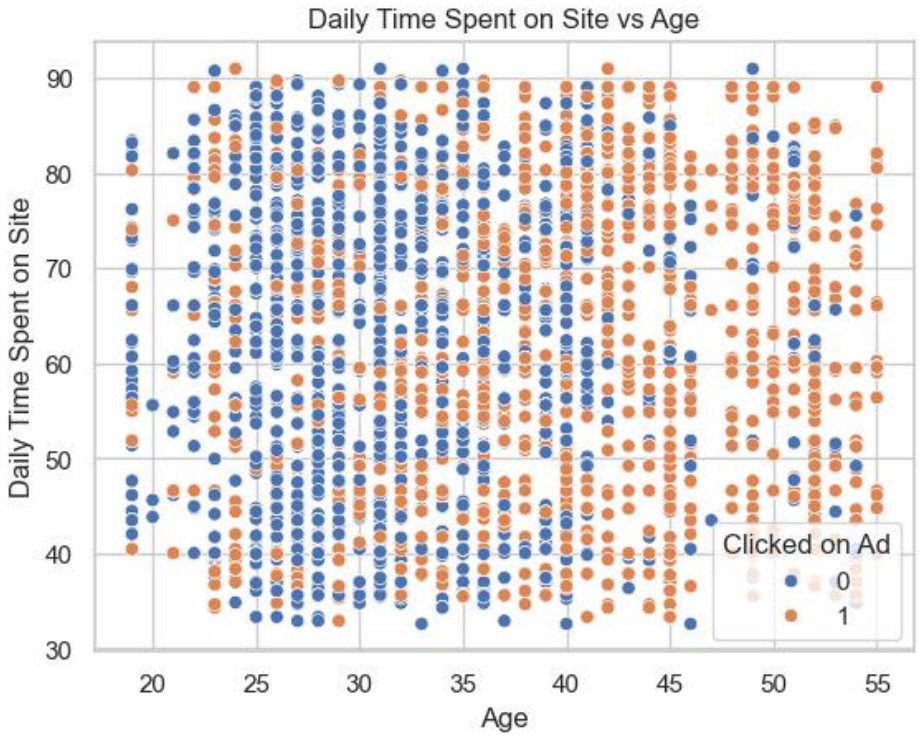
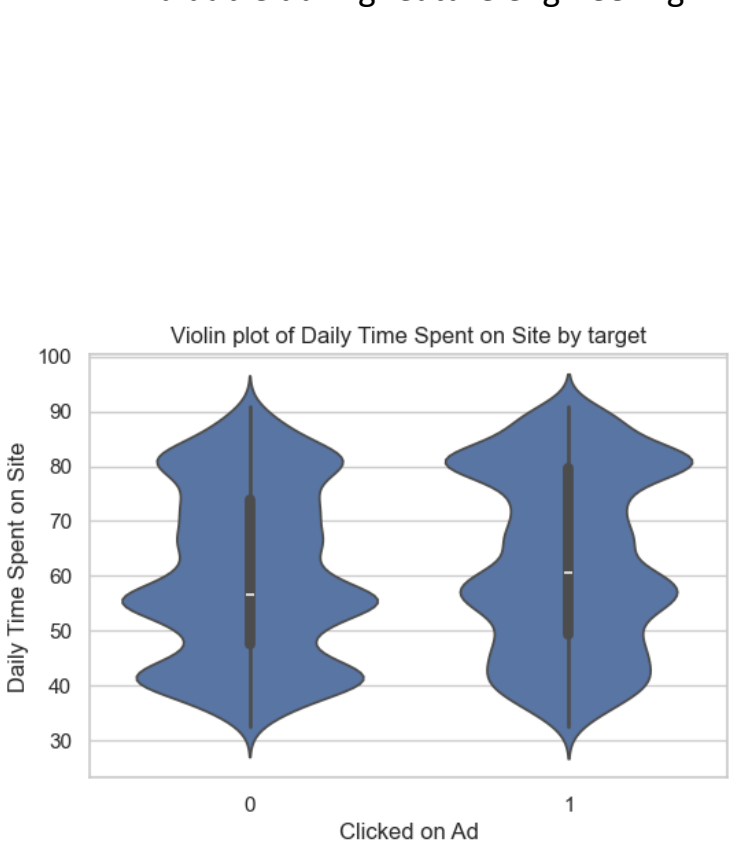


Figure 4.8: Word Cloud of Ad Topic Line

This word cloud displays the most commonly used words in the `Ad Topic Line` feature. Considering there are only 557 unique ad topic lines, this information may not be as useful as other features.

Figure 4.9: Violin Plots of numerical features

These Violin plots visualize the distribution of each numerical feature based on the target variable. Based on these plots, we can guess the correlation between the `Clicked on Ad` feature and the numerical features. This information is extremely valuable during feature engineering.



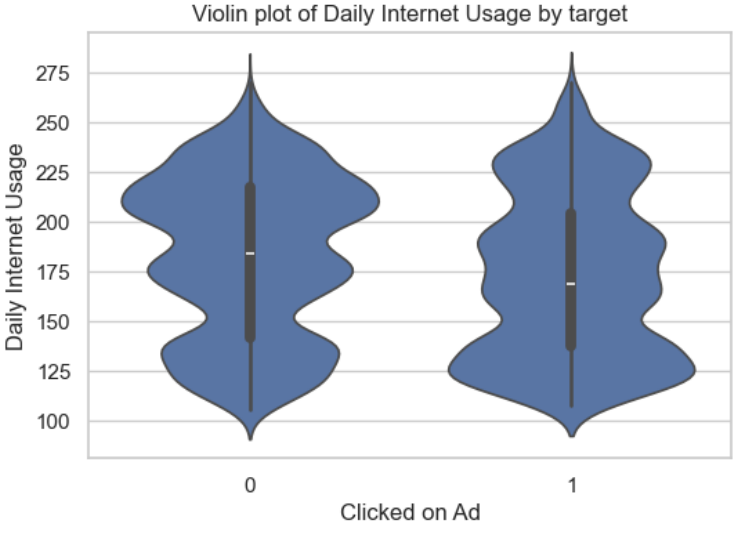
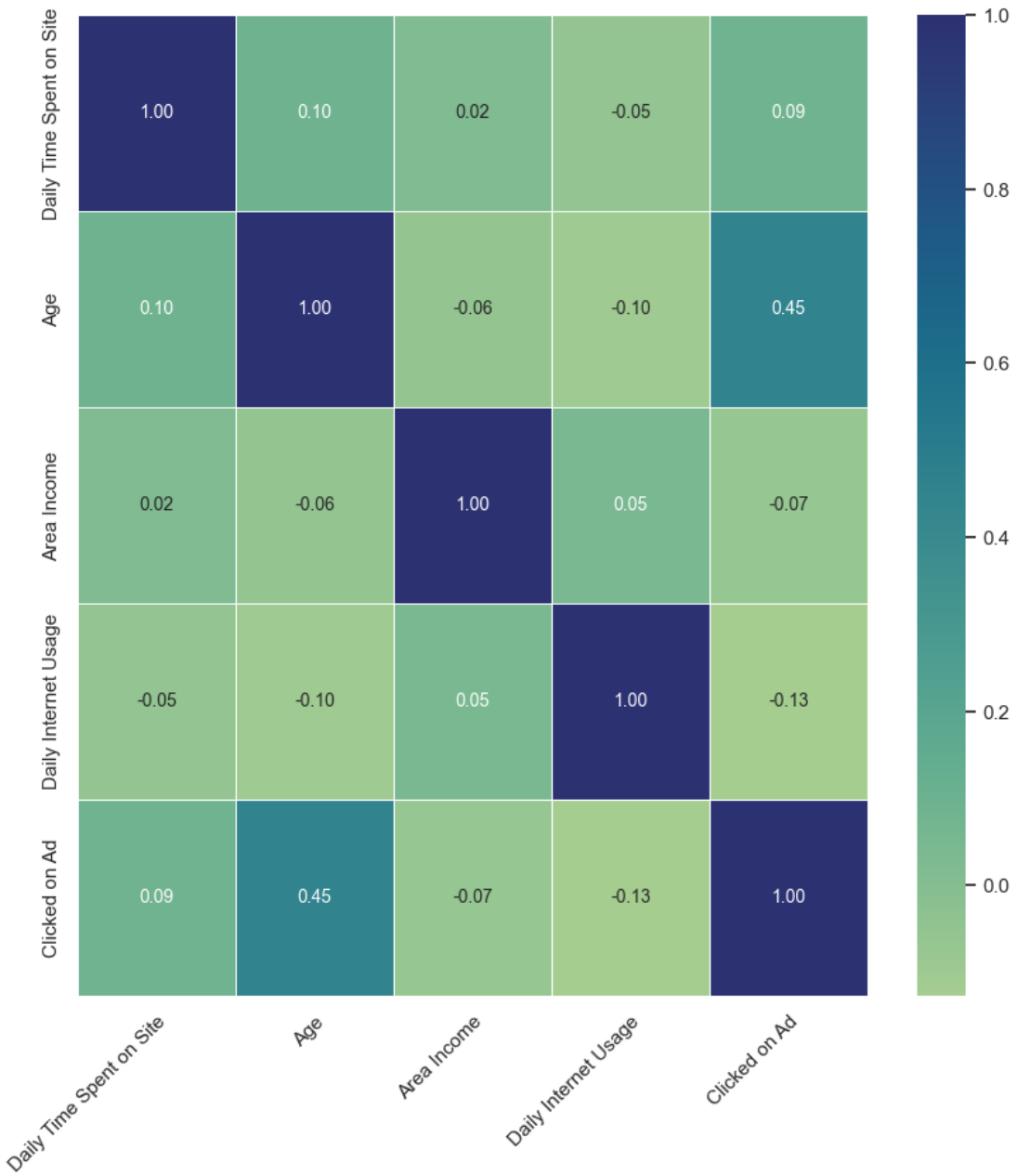


Figure 4.10: Heatmap of the Correlation Matrix

This heatmap visualizes the correlation between the various attributes of the dataset. Some strong correlations can be found here, such as the correlation between `Age `and 'Clicked on Ad'.

However, there aren't many strong correlations to be found here, making feature engineering a crucial next step. Note that many features of the dataset are strings of text that first need to be encoded.



5. Feature Engineering and Encoding

I extracted several new features from the dataset, including:

- **Percentage of Daily Time Spent on Site:** Each record contains a `Daily Time Spent on Site` feature and a `Daily Internet Usage` feature. Using both features, we can determine what percentage of a user's online activity is done on the specified website. `Percentage of Daily Time Spent on Site` is created by dividing the time spent on the site by the daily internet usage.
- **Age Over 35:** Based on the EDA performed, we can infer that `Age` is highly correlated with the target variable. A rough guess based on one of our scatter plots shows that people over the Age of 35 are more likely to click on ads. Therefore, a new feature `Age Over 35` is created.
- **DateTime Features:** The `Timestamp` feature can be used to extract various time-related data. `Day`, `Day of Week`, `Month`, and `Hour` are extracted. There is no point in adding a Year feature as all records are set in 2016, meaning that year doesn't provide any information that will help discriminate records.
- **Time Segment:** An additional feature can be created using the `Hour` feature. Dividing the 24-hour cycle to 4 equal segments, allows us to create a new feature for each `Time Segment`. I created one feature with 4 possible values (0-6, 6-12, 12-18, 18-24) which can later be one-hot encoded to 4 different features.

Feature Encoding: The dataset currently has 5 non-numerical features that should be encoded. First and foremost, `Gender` can be encoded as a simple binary feature as the dataset contains 2 possible values for gender.

The other 4 features are **one-hot encoded** to create over 1000 new features. `Time Segment` only has 4 possible values, however `Country` , `City`, and `Ad Topic Line` features are nominal and each has over 200 different possible values. Using other types of encoding is not appropriate here as there is no order or priority defined for these values.

6. Feature Scaling

Feature scaling is a crucial step in this project as some of the attributes we're working with are in completely different scales. It is imperative to avoid **data leakage** while scaling the features. Meaning the data from the validation set should not affect our scaling at all.

First, I used a train-test split to create a sample for training models. Note that the data split happened before data scaling. Not all numerical features need to be scaled. I only chose appropriate features with high variance and generally higher values (such as `Area Income`) to be scaled. One-hot vectors, for instance, do not need to be scaled.

To achieve this, I used a simple preprocessor to apply the scaling to the dataset. Only a set of specified numerical columns are scaled. There are 2 important things to note here:

- The preprocessor (scaler) is first fit on the `Train` data. Afterward, both the `Train` and `Test` data are transformed using the scaler. This is done to prevent data leakage.
- Later on, the preprocessor is used again for Cross-Validation, where each step requires a separate scaler based on the chosen `Train Folds` and `Test Folds`.

Ultimately, I took every measure to avoid data leakage. I ensured no leakage occurred during sample tests or cross-validation.

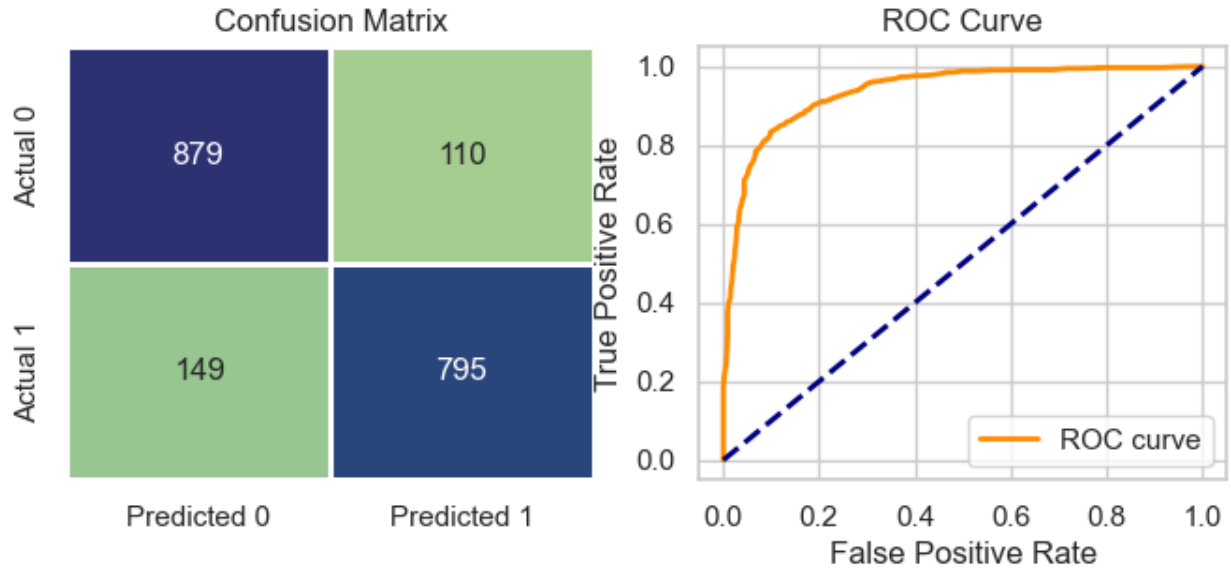
Note that for Cross-Validation, I used a Repeated Stratified K-fold Cross-Validation function. In order to be able to fit and use different scalers for each step, a **pipeline** must be passed to the function that includes a Scaling step.

According to the Scikit Learn documentation, this method of including the scaling step in a pipeline causes no data leakage and Cross-Validation results are calculated after the data has been scaled according to the chosen training data folds.

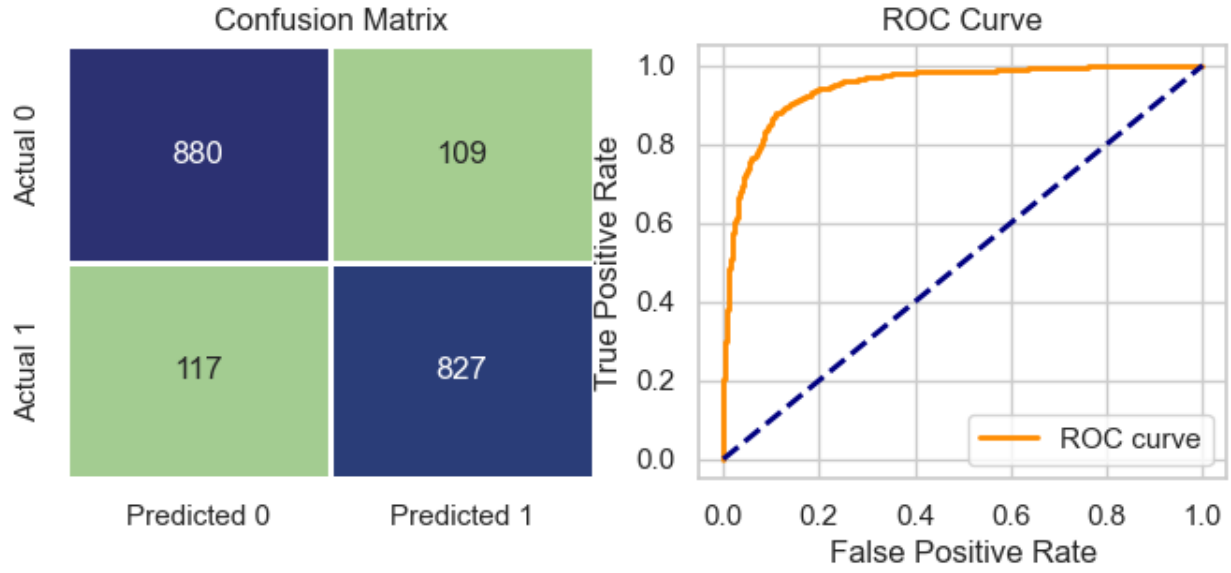
7. Training Models

For this project, I built 4 different models to predict the `Clicked on Ad` attribute. Each model is first tested with the sample `Train` and `Test` datasets, and then Cross-Validation is used.

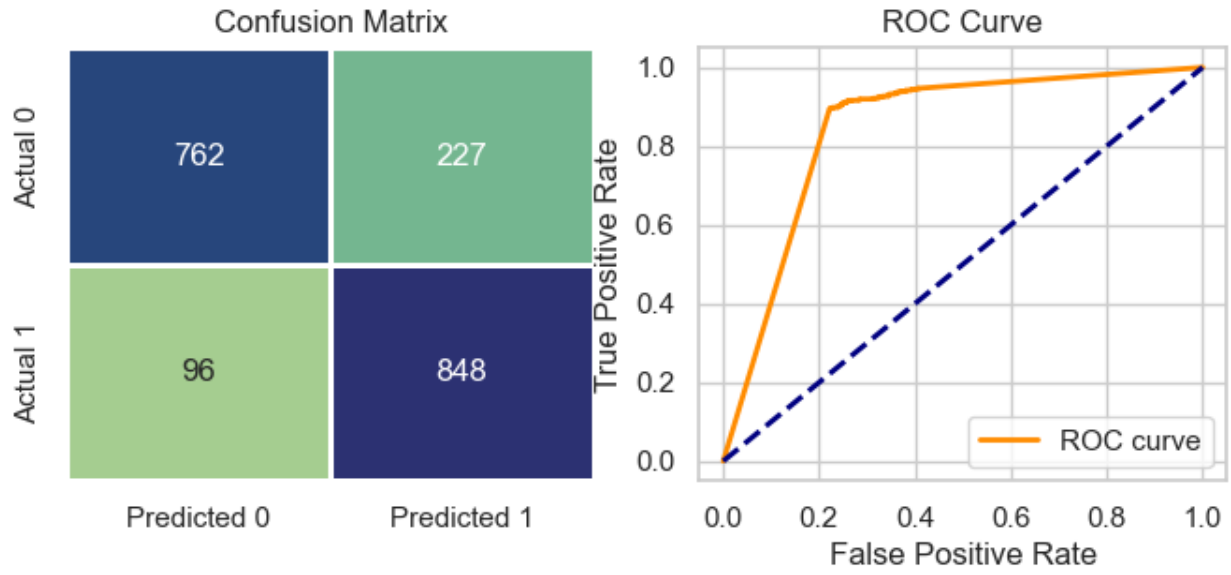
7.1: Random Forest Classifier



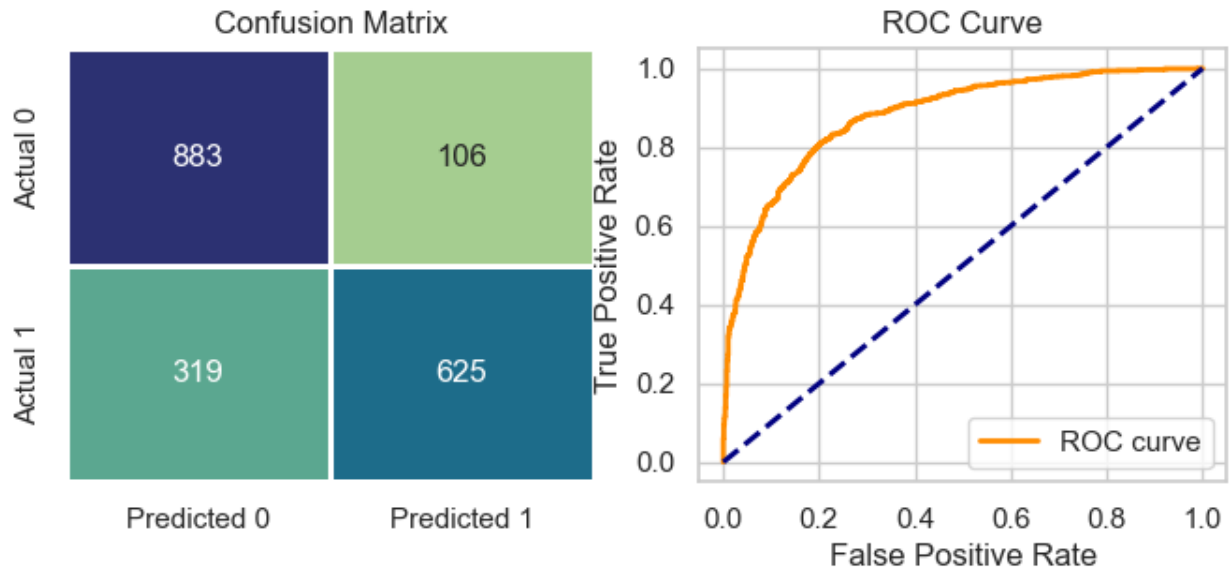
7.2: Logistic Regression



7.3: Naïve Bayes



7.4: Support Vector Classifier



(SVC results are without applying PCA)

7.5: Final Results

Below you can find the best **accuracy** results I was able to achieve for each model with cross-validation:

- Random Forest Classifier: Scaled + NO PCA
  Best CV score: **0.884**
- Logistic Regression: Scaled + NO PCA
  Best CV score: **0.893**
- Naive Bayes: Scaled + NO PCA
  Best CV score: **0.851**
- Support Vector Classifier: Polynomial kernel + Scaled + PCA applied
  Best CV score: **0.837**

I was able to achieve the best results with Logistic Regression, using the standard scaler and without applying PCA to the dataset. Scaling the data for LR is crucial, as without data scaling it reached an accuracy of 0.56.

Note that Logistic Regression and Random Forest performed similarly while SVC and Naïve Bayes performed worse. SVC in particular gained a boost in performance after applying PCA to the dataset.

In conclusion, I consider the almost 90% accuracy of Logistic Regression to be the best result for this project.