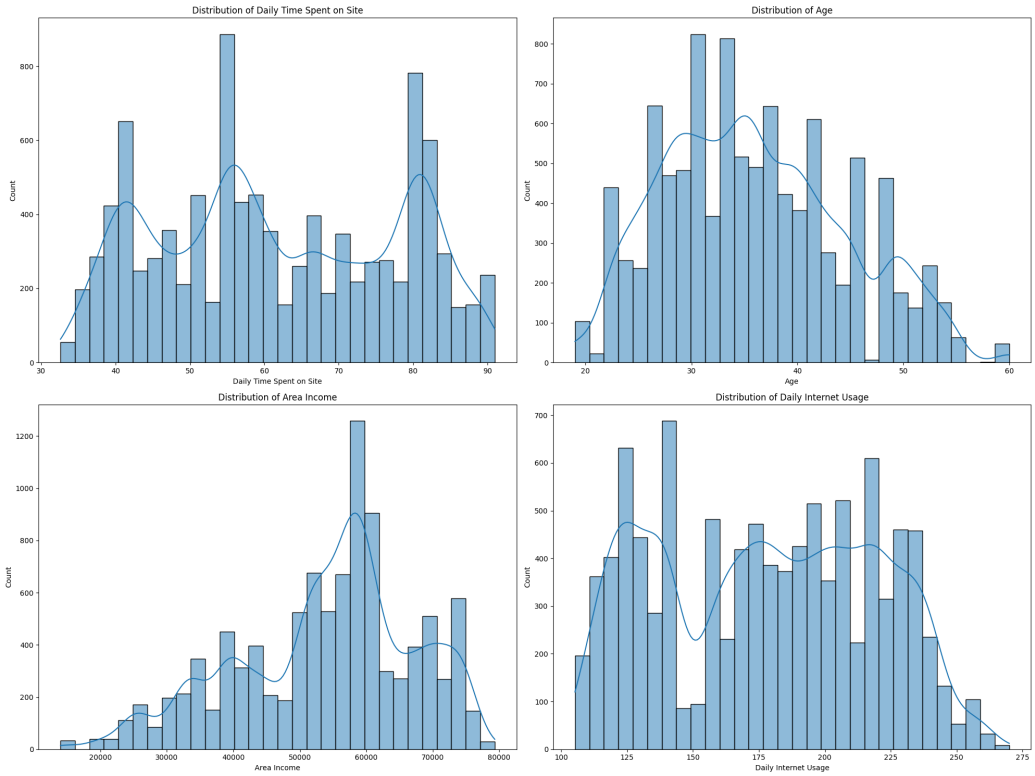
Aryan Neizehbaz – 400222112

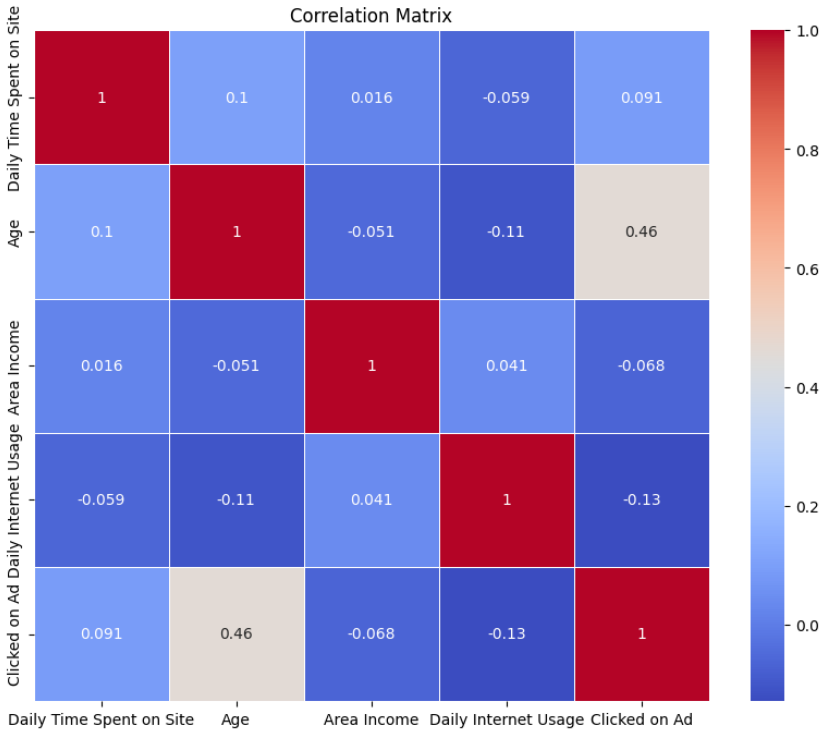
4th Assignment (Part 1)

1) EDA

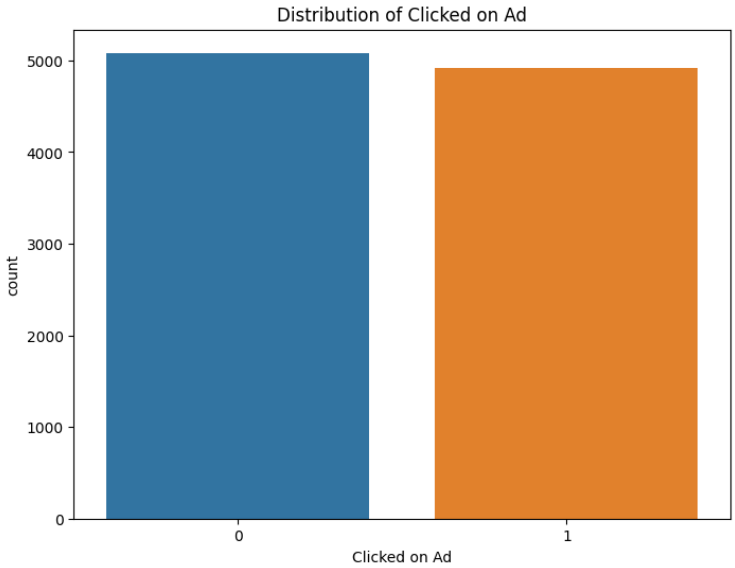
- **Daily Time Spent on Site**: The distribution is somewhat uniform with multiple peaks. There are users who spend varying amounts of time on the site.
- **Age**: The distribution is roughly normal with a peak around ages 30-35, suggesting that the majority of users fall within this age range.
- **Area Income**: The distribution is right-skewed, with a peak around the $60,000 mark, indicating that most users come from areas with moderate income levels.
- **Daily Internet Usage**: The distribution shows multiple peaks, indicating variability in internet usage among users.



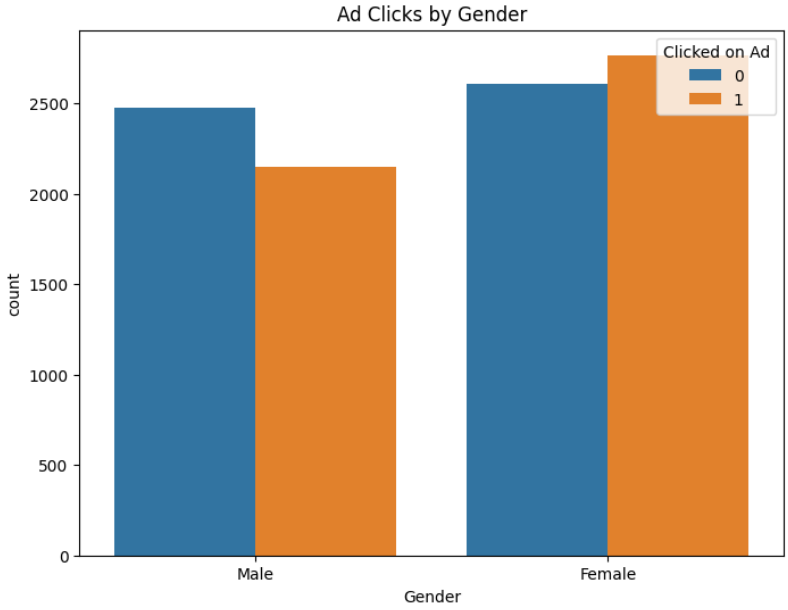
- The correlation matrix shows that `Age` has a moderate positive correlation with `Clicked on Ad` (0.46), suggesting that older users are more likely to click on ads.
- `Daily Time Spent on Site` and `Daily Internet Usage` show weak correlations with `Clicked on Ad`.
- `Area Income` has a weak negative correlation with `Clicked on Ad`.



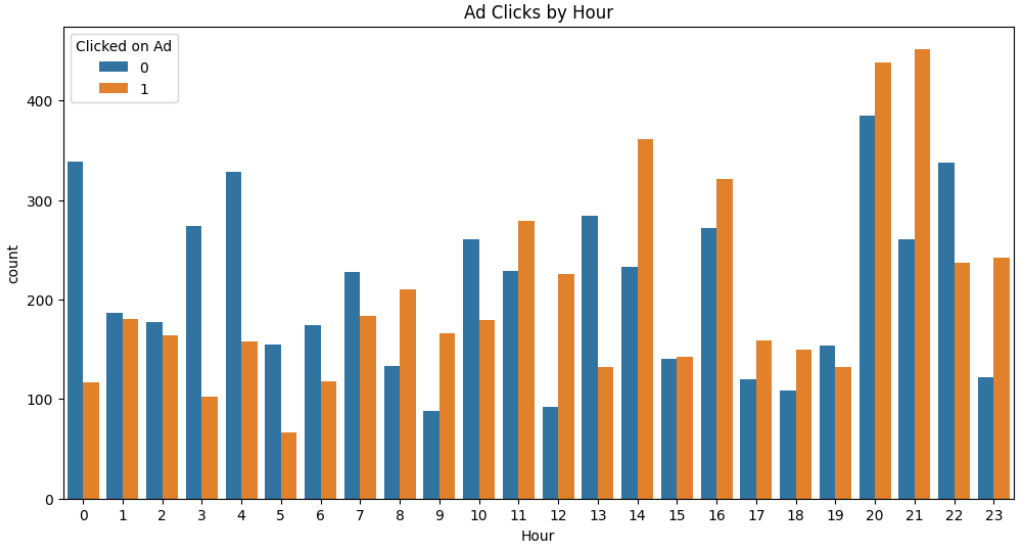
The target variable `Clicked on Ad` is fairly balanced with almost equal counts for both classes (clicked and not clicked).

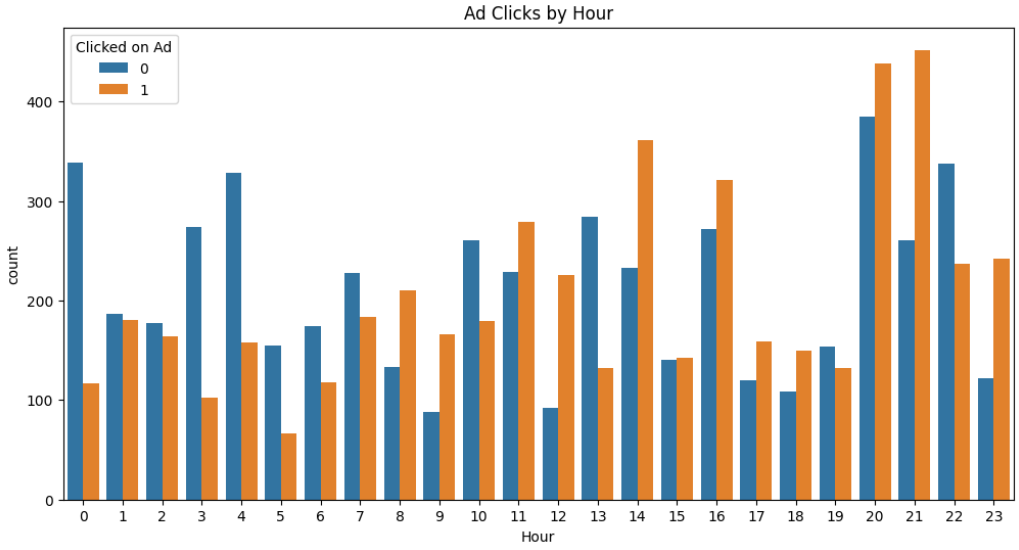


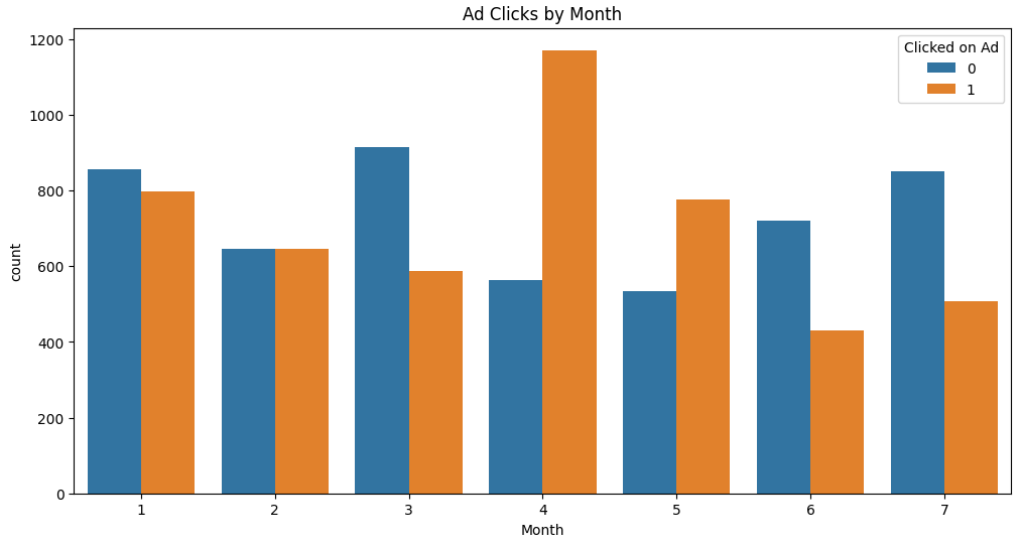
Both genders show similar patterns in terms of ad clicks. However, females seem to click on ads slightly more than males.

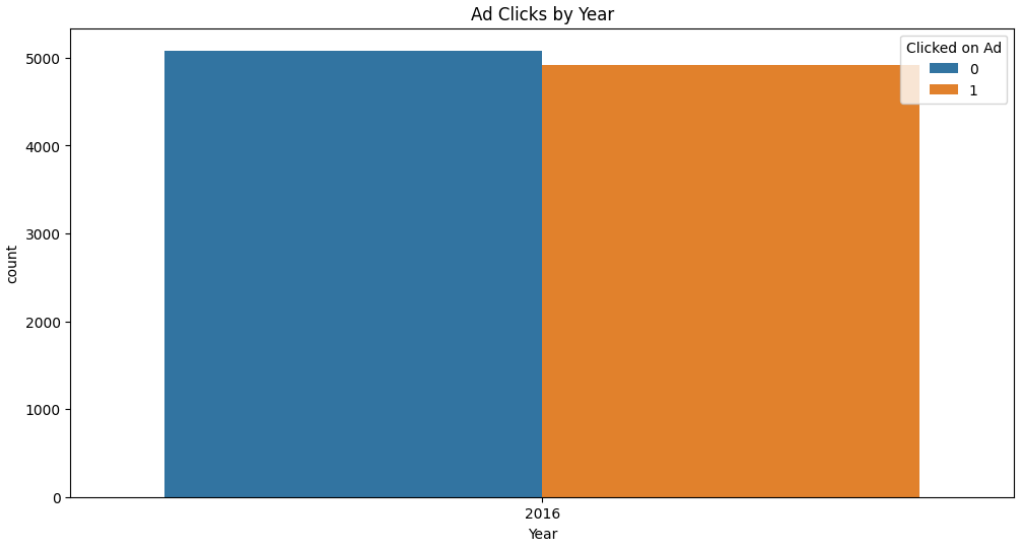


- **Ad Clicks by Hour**: There are significant variations throughout the day, with noticeable peaks around certain hours, indicating higher user activity and engagement at specific times.
- **Ad Clicks by Day**: No strong patterns are visible for ad clicks by day.
- **Ad Clicks by Month**: April shows the highest number of ad clicks, suggesting possible seasonal trends or campaign-specific factors.
- **Ad Clicks by Year**: Data is from a single year, 2016, and shows a balanced distribution of ad clicks.




Ad Clicks by Hour
Clicked on Ad
0
1
count
Hour
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
0 100 200 300 400


Ad Clicks by Month
Clicked on Ad
0
1
count
Month
1 2 3 4 5 6 7
0 200 400 600 800 1000 1200


Ad Clicks by Year
Clicked on Ad
0
1
count
Year
2016
0 1000 2000 3000 4000 5000

2) Feature Engineering

1. Timestamp Conversion

The `Timestamp` column, initially in string format, was converted to a datetime object. This conversion allows for easier extraction of temporal features such as hour, day, month, and year.

2. Extraction of Time-Based Features

After converting the `Timestamp` column to a datetime object, four new features were extracted:

- **Hour**: Represents the hour of the day when the ad interaction occurred.
- **Day**: Represents the day of the month when the ad interaction occurred.
- **Month**: Represents the month of the year when the ad interaction occurred.
- **Year**: Represents the year when the ad interaction occurred.

These time-based features can provide insights into temporal patterns in user behavior and ad interactions.

3. One-Hot Encoding of Categorical Variables

Categorical variables such as `Gender`, `Country`, and `City` were transformed using one-hot encoding. This process converts categorical values into a series of binary columns, where each unique category is represented as a separate column with 0 or 1 values. One-hot encoding helps in handling categorical data for machine learning algorithms that require numerical input.

4. Dropping Irrelevant Columns

Columns that do not contribute directly to the predictive model were dropped. These include:

- **Ad Topic Line**: The textual content of the ad topic line, which was not used in this analysis.
- **Timestamp**: The original timestamp column, as its relevant components (hour, day, month, year) were already extracted.

5. Splitting the Dataset into Features and Target Variable

The dataset was split into features (X) and the target variable (y):

- **Features (X)**: All columns except for the target variable `Clicked on Ad`.
- **Target (y)**: The column `Clicked on Ad`, which indicates whether a user clicked on the ad.

6. Train-Test Split

The dataset was divided into training and testing sets to evaluate the performance of the predictive model:

- **Training Set (80%)**: Used to train the model.
- **Testing Set (20%)**: Used to test the model's performance on unseen data.

A random state was set to ensure the reproducibility of the results.

7. Feature Scaling

Feature scaling was performed to standardize the range of the features. Standardization involves rescaling the features such that they have a mean of zero and a standard deviation of one. This step is crucial for machine learning algorithms that are sensitive to the scale of input data. The scaling was done separately for the training and testing sets to prevent data leakage.

3) Model

Feature Importance Analysis

Initially, a Random Forest Classifier was trained on the dataset to determine the importance of each feature. The top 50 most important features were selected based on their importance scores. This step reduced the dimensionality of the dataset and focused the training process on the most relevant features, enhancing model performance and reducing overfitting.

Model Training and Initial Evaluation

Three models were trained and evaluated using the reduced feature set:

- **Logistic Regression**
- **Random Forest Classifier**
- **Gradient Boosting Classifier**

The models were evaluated using accuracy, precision, recall, F1 score, and cross-validation mean accuracy. The results were as follows:

	Accuracy	Precision	Recall	F1 Score	Cross-Validation Mean Accuracy
Logistic Regression	0.8315	0.850811	0.798174	0.823653	0.820500
Random Forest	0.8600	0.868476	0.843813	0.855967	0.848750
Gradient Boosting	0.8230	0.851111	0.776876	0.812301	0.808875

The Random Forest model outperformed the other models across all metrics, indicating its suitability for this classification task.

Hyperparameter Tuning

To further improve the performance of the Random Forest model, hyperparameter tuning was conducted using RandomizedSearchCV. The following hyperparameters were tuned:

- Number of trees in the forest (`n_estimators`)
- Maximum depth of the tree (`max_depth`)
- Minimum number of samples required to split an internal node (`min_samples_split`)
- Minimum number of samples required to be at a leaf node (`min_samples_leaf`)

The best hyperparameters identified were:

- `n_estimators`: 500
- `max_depth`: None
- `min_samples_split`: 5
- `min_samples_leaf`: 1

Optimized Model Training and Evaluation

Using the optimized hyperparameters, a new Random Forest model was trained and evaluated. The performance metrics for the optimized model were as follows:

```
Optimized Model Evaluation Results
Accuracy: 0.8680
Precision: 0.8714
Recall: 0.8590
F1 Score: 0.8652
Cross-Validation Mean Accuracy: 0.8499
```

The optimized Random Forest model demonstrated a slight improvement in performance compared to the initial Random Forest model, particularly in precision and F1 score.