Predict Clicked Ads Customer Classification by using Machine Learning

Supported by:
Rakamin Academy
Career Acceleration School
www.rakamin.com

Created by:
Anik Maulia Tri Handayani
anikmaulia263@gmail.com
linkedin.com/in/anikmauliatrihandayani

I possess foundational knowledge in data analysis and a proven ability to design and develop Machine Learning solutions. I have applied it to an predict clicked ads customer classification project. I strongly believe in the pivotal role data plays in enhancing business performance. I hold relevant certifications and practical experience, which have prepared me to be a dedicated data professional. I am also adept at collaborating within multidisciplinary teams and am ready to advance my career in data analysis across various industries, including telecommunications, commerce, retail, FMCG, finance, and banking.

Overview

"A company in Indonesia wishes to assess the effectiveness of an advertisement they have broadcasted. This is crucial for the company to gauge the reach of their marketing campaign and attract customers to view the advertisement. By analyzing historical advertisement data and uncovering insights and patterns, this can aid the company in establishing marketing targets. The primary focus of this case is to develop a machine learning classification model that can identify the right target customers."

Exploratory Data Analysis (EDA): Statistical Analysis

Rakamin Academy

Categorical Summaries

```
df[cat_cols].describe().T
```

	count	unique	top	freq
Male	997	2	Perempuan	518
Clicked on Ad	1000	2	No	500
city	1000	30	Surabaya	64
province	1000	16	Daerah Khusus Ibukota Jakarta	253
category	1000	10	Otomotif	112

The majority of customers are female, they are distributed across 30 cities and spread across 16 provinces. Both the total number of customers who clicked and didn't click on the ad are equal, with the ad clicks distributed across 10 different categories.

Numerical Summaries

```
df[num_cols].describe().T
```

	count	mean	std	min	25%	50%	75%	max
Daily Time Spent on Site	987.00	64.93	15.84	32.60	51.27	68.11	78.46	91.43
Age	1000.00	36.01	8.79	19.00	29.00	35.00	42.00	61.00
Area Income	987.00	384864670.64	94079989.57	97975500.00	328632990.00	399068320.00	458355450.00	556393600.00
Daily Internet Usage	989.00	179.86	43.87	104.78	138.71	182.65	218.79	267.01

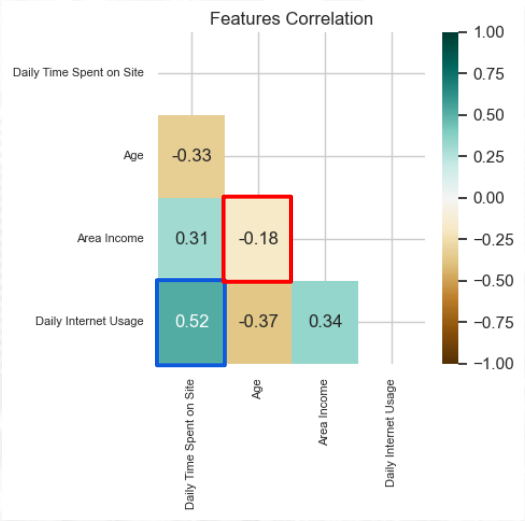
Customers' age ranges from 19 to 61 years, with an area income varying between 97,975,500 and 556,393,600 IDR. On average, customers spend 68.11 minutes daily on the site and 182.65 minutes on the internet.

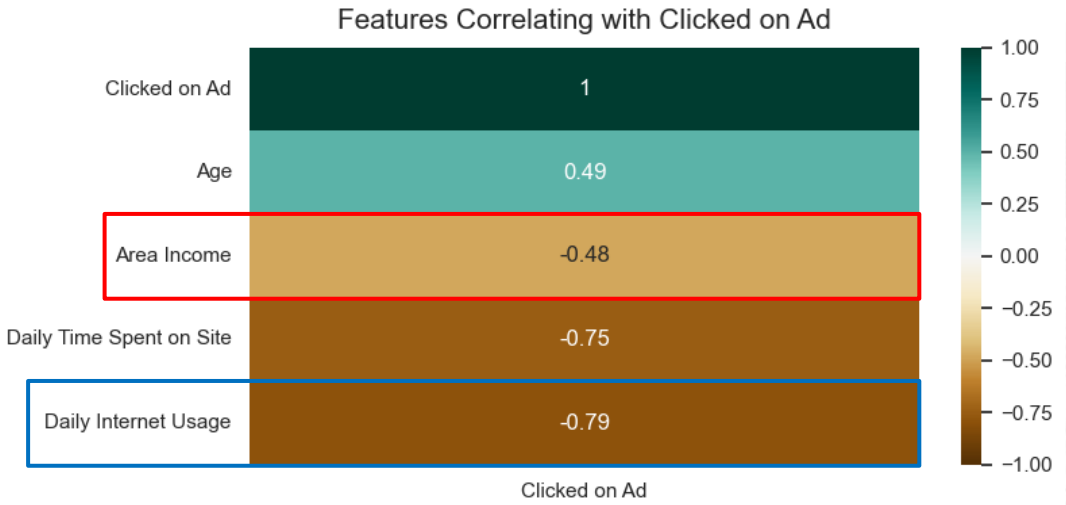
For more details, you can refer to the Jupyter notebook here

Exploratory Data Analysis (EDA): Multivariate Analysis

Rakamin Academy

Lowest Correlation Highest Correlation

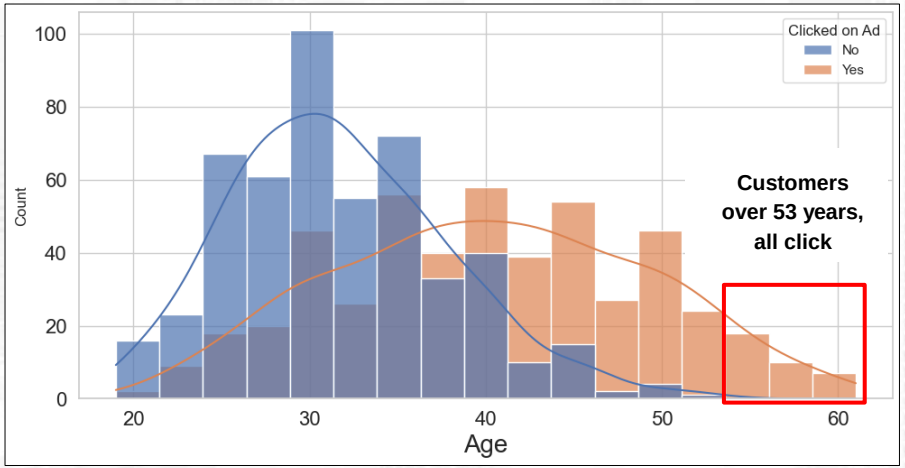




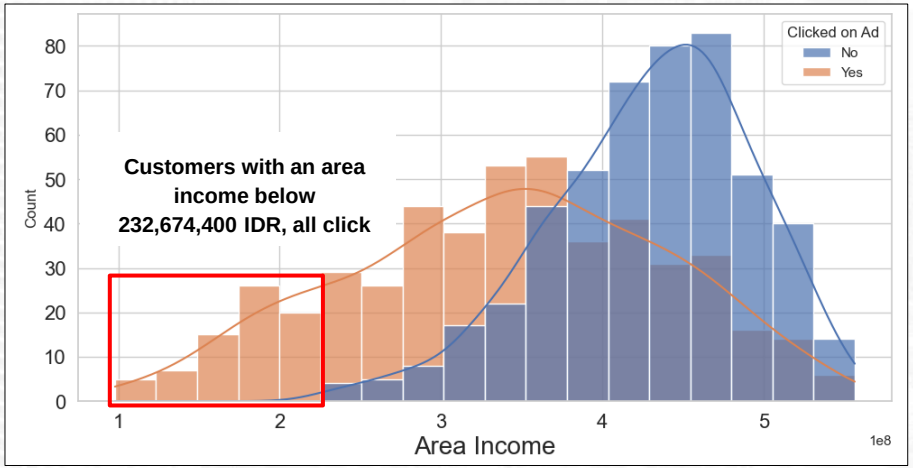
The higher the correlation between features or a feature with the target, the greater the influence, and vice versa.

Customer Type Analysis on Advertisement

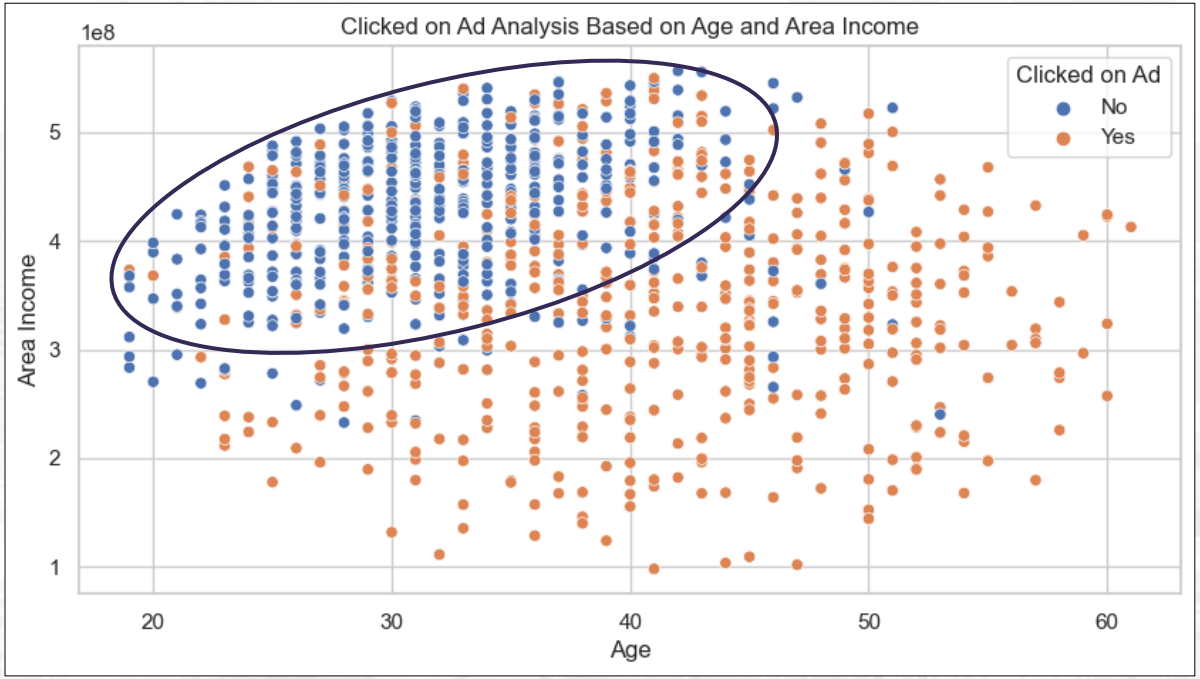
1. Customer Type Analysis on Advertisement by Age



2. Customer Type Analysis on Advertisement by Area Income



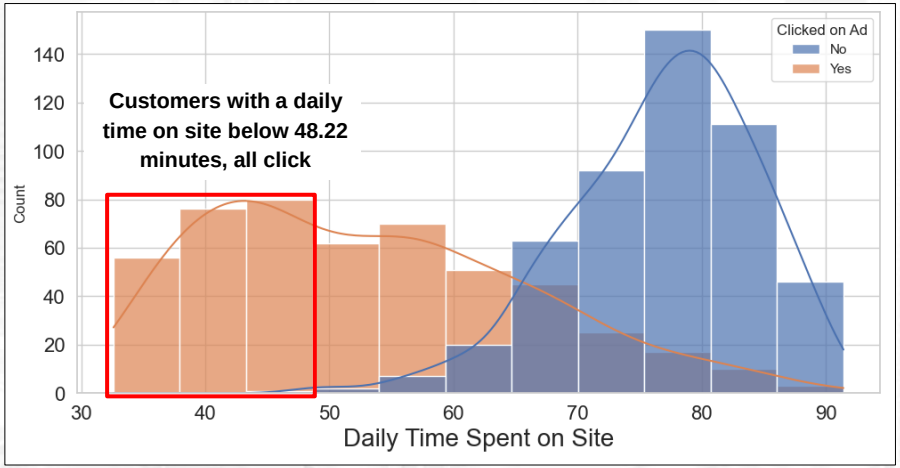
Customer Type Analysis on Advertisement



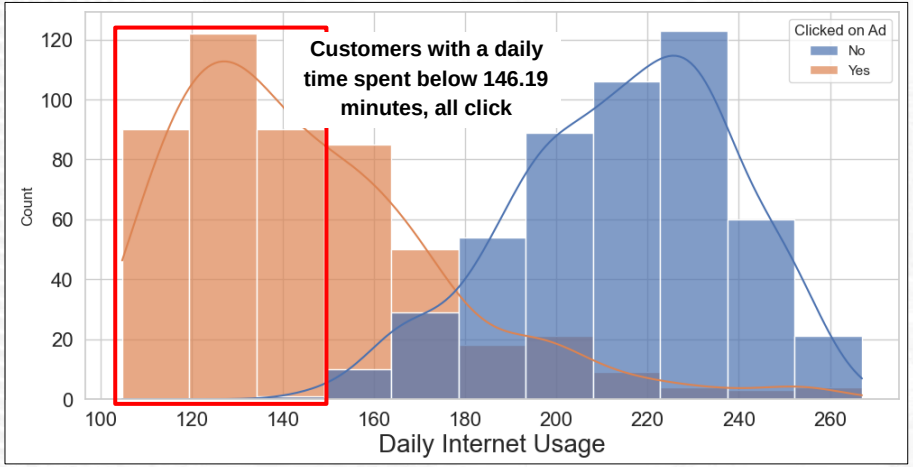
Younger customers with a higher area income tend to be less likely to click on ads. However, there are still individuals within these categories who do engage with the ads.

Customer Behaviour Analysis on Advertisement

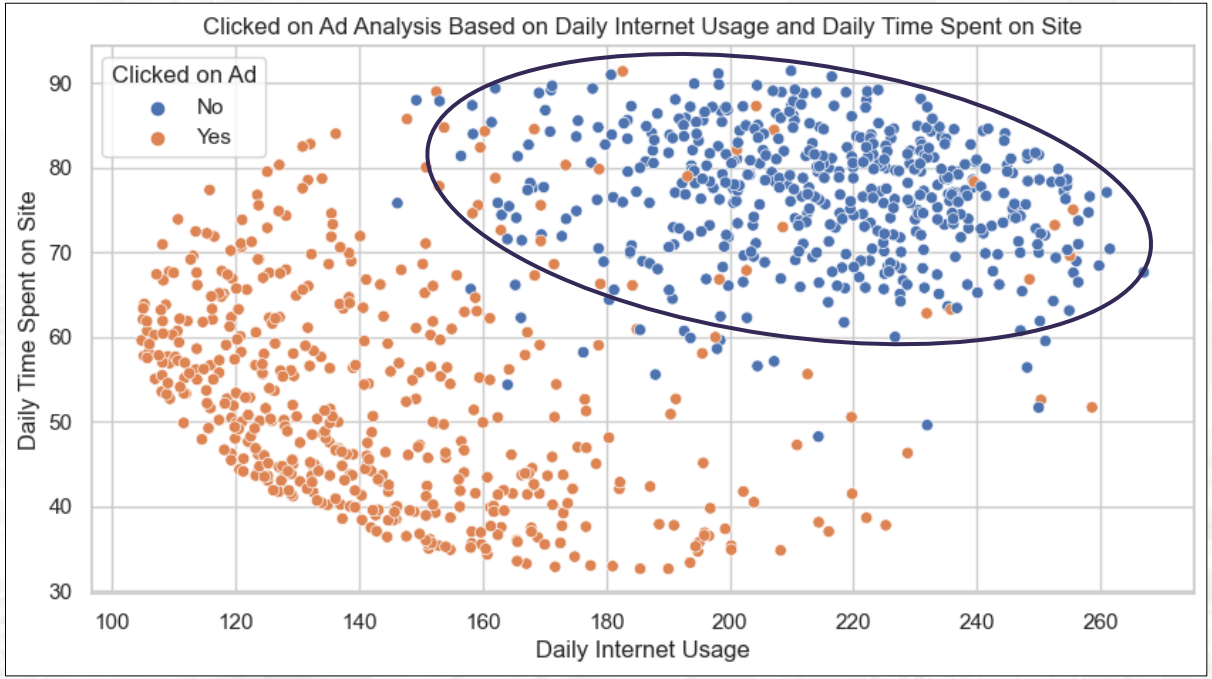
1. **Customer Behavior Analysis on Advertisement by Daily Spent Time on Site**



2. **Customer Behavior Analysis on Advertisement by Daily Internet Usage**

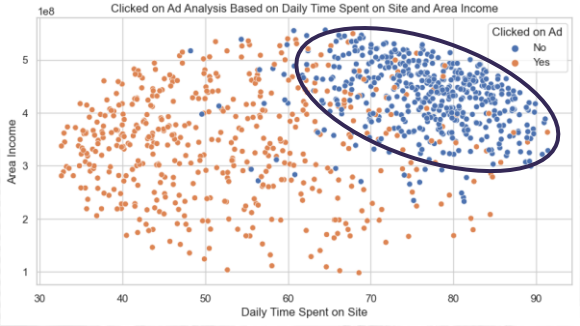


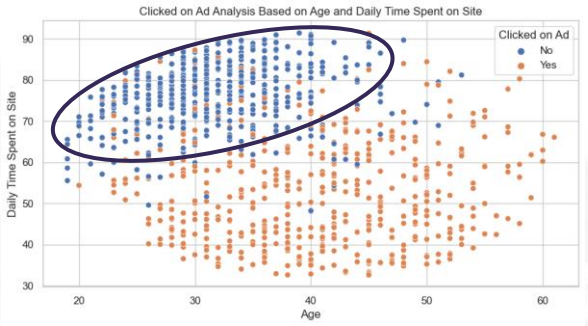
Customer Behaviour Analysis on Advertisement

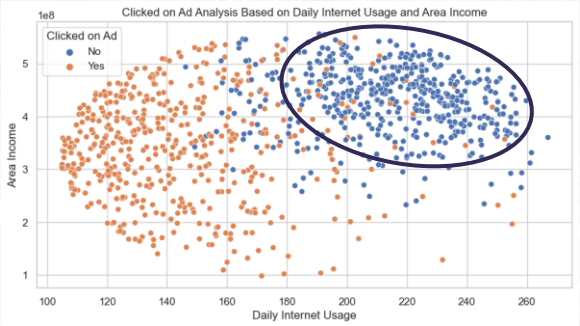


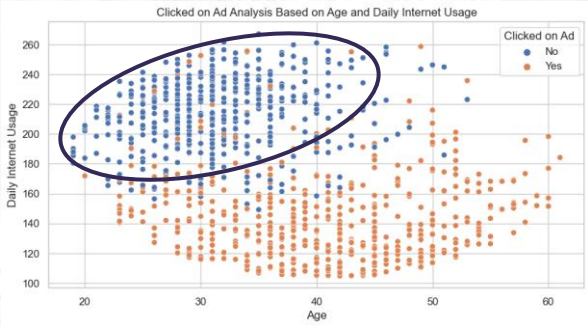
Customers with higher daily internet usage and daily time spent on site tend to be less likely to click on ads, and vice versa.

Customer Type and Behaviour Analysis on Advertisement









There is no significant correlation between the observed features and clicked on advertisement target. Not all customers with older age or higher area income refrain from clicking; some of them still engage with the ads. This could be due to other influencing factors that require further analysis.

Data Cleaning & Preprocessing

Rakamin Academy

Missing Values

```
# Check null values
df.isnull().sum()[df.isnull().sum() > 0]
```

```
Daily Time Spent on Site    13
Area Income                 13
Daily Internet Usage        11
Male                         3
```

```
# Impute using the median value
df['Daily Time Spent on Site'].fillna(df['Daily Time Spent on Site'].median(), inplace=True)
df['Daily Internet Usage'].fillna(df['Daily Internet Usage'].median(), inplace=True)
```

```
# Impute using the median value from data that shares the same city value.
df['Area Income'].fillna(df.groupby('city')['Area Income'].transform('median'), inplace=True)
```

```
# Impute using the mode value
df['Male'].fillna(df['Male'].mode()[0], inplace=True)
```

There are missing values in the **Daily Time Spent on Site**, **Area Income**, **Daily Internet Usage**, and **Male** columns, and the missing values will be handled through imputation based on the median and mode values.

Duplicated Data

```
print("Number of rows before removing duplicates:", len(df))
print("Number of rows after removing duplicates:", len(df.drop_duplicates()))
```

```
Number of rows before removing duplicates: 1000
Number of rows after removing duplicates: 1000
```

Dataset contains no duplicates

For more details, you can refer to the Jupyter notebook here

Data Cleaning & Preprocessing

Rakamin Academy

Invalid Values

There are invalid values in the **Gender**, **City**, **Province, Category,** and **Timestamp** columns data.

```
# Rename columns
df.rename(columns={'Male': 'Gender',
                   'city': 'City',
                   'province': 'Province',
                   'category': 'Category'}, inplace=True)
```

```
# Replacing values in the Gender column
df['Gender'] = df['Gender'].replace({
    'Laki-Laki': 'Male',
    'Perempuan': 'Female'
})
```

```
# Replacing values in the City column
df['City'] = df['City'].replace({
    'Jakarta Timur': 'East Jakarta',
    'Jakarta Selatan': 'South Jakarta',
    'Jakarta Barat': 'West Jakarta',
    'Jakarta Utara': 'North Jakarta',
    'Jakarta Pusat': 'Central Jakarta',
    'Tangerang Selatan': 'South Tangerang'
})
```

```
# Replacing values in the Province column
df['Province'] = df['Province'].replace({
    'Daerah Khusus Ibukota Jakarta': 'Special Capital Region of Jakarta',
    'Jawa Barat': 'West Java',
    'Jawa Timur': 'East Java',
    'Jawa Tengah': 'Central Java',
    'Sumatra Utara': 'North Sumatra',
    'Sumatra Selatan': 'South Sumatra',
    'Kepulauan Riau': 'Riau Islands',
    'Kalimantan Timur': 'East Kalimantan',
    'Sulawesi Selatan': 'South Sulawesi',
    'Kalimantan Selatan': 'South Kalimantan',
    'Sumatra Barat': 'West Sumatra',
    'Kalimantan Barat': 'West Kalimantan'
})
```

```
# Replacing values in the Category column
df['Category'] = df['Category'].replace({
    'Otomotif': 'Automotive'
})
```

```
# Change the data type of the Timestamp column to datetime
df['Timestamp'] = pd.to_datetime(df['Timestamp'])
```

Data Cleaning & Preprocessing

Rakamin Academy

Feature Extraction

```
# df['Year'] = df['Timestamp'].dt.year --> Uniform values i.e 2016; making them less likely to be an important feature
df['Month'] = df['Timestamp'].dt.month
df['Week'] = df['Timestamp'].dt.strftime('%U').astype(int) + 1
df['Day'] = df['Timestamp'].dt.day
```

There are extracted feature from **Timestamp** data; **Month**, **Week**, and **Day** columns.

Feature Encoding

```
# Mapping for Clicked on Ad column
map_clicked = {
    'No': 0,
    'Yes': 1
}
df['Clicked on Ad'] = df['Clicked on Ad'].map(map_clicked)
```

```
df = pd.get_dummies(df, columns=['Gender', 'City', 'Province', 'Category'])
```

Two techniques used for feature encoding are: Label Encoding for the **Clicked on Ad** column; One-Hot Encoding for the **Gender**, **City**, **Province**, and **Category** columns.

Feature Scaling

```
df_scaling['Area Income'] = np.log(df_scaling['Area Income'])
```

```
scalling_columns = ['Daily Time Spent on Site', 'Age', 'Daily Internet Usage', 'Month', 'Week', 'Day']
```

```
scaler = StandardScaler()
df_scaling[scalling_columns] = scaler.fit_transform(df_scaling[scalling_columns])
```

Feature scaling was performed on a copied dataframe to enable two modeling approaches, one with scaling and one without. Two techniques used for feature scaling are: Log Tranform for the **Area Income** column; Standard Scaler for the **Daily Time Spent on Site**, **Age**, **Daily Internet Usage**, **Month**, **Week**, and **Day** columns.

Data Cleaning & Preprocessing

Rakamin Academy

Feature and Target

```
# Data with scaling
X_scaling = df_scaling.drop('Clicked on Ad', axis=1)
y_scaling = df_scaling['Clicked on Ad']
```

```
# Data without scaling
X_no_scaling = df_no_scaling.drop('Clicked on Ad', axis=1)
y_no_scaling = df_no_scaling['Clicked on Ad']
```

Data Splitting

```
# Splitting the data with feature scaling
train_scaling, test_scaling = train_test_split(df_scaling, test_size=0.2, stratify=df_scaling[['Clicked on Ad']], random_state=42)
train_scaling.reset_index(drop=True, inplace=True)
test_scaling.reset_index(drop=True, inplace=True)

print(train_scaling.shape)
print(test_scaling.shape)
```

```
(800, 66)
(200, 66)
```

```
X_train_scaling = train_scaling.drop(['Clicked on Ad'], axis=1).reset_index(drop=True)
y_train_scaling = train_scaling['Clicked on Ad'].reset_index(drop=True)
print(X_train_scaling.shape, y_train_scaling.shape)

X_test_scaling = test_scaling.drop(['Clicked on Ad'], axis=1).reset_index(drop=True)
y_test_scaling = test_scaling['Clicked on Ad'].reset_index(drop=True)
print(X_test_scaling.shape, y_test_scaling.shape)
```

```
(800, 65) (800,)
(200, 65) (200,)
```

Data Cleaning & Preprocessing

Rakamin Academy

Data Splitting: Continued

```
# Splitting the data withot feature scaling
train_no_scaling, test_no_scaling = train_test_split(df_no_scaling, test_size=0.2, stratify=df_no_scaling[['Clicked on Ad']], random_state=42)
train_no_scaling.reset_index(drop=True, inplace=True)
test_no_scaling.reset_index(drop=True, inplace=True)

print(train_no_scaling.shape)
print(test_no_scaling.shape)
```

```
(800, 66)
(200, 66)
```

```
X_train_no_scaling = train_no_scaling.drop(['Clicked on Ad'], axis=1).reset_index(drop=True)
y_train_no_scaling = train_no_scaling['Clicked on Ad'].reset_index(drop=True)
print(X_train_no_scaling.shape, y_train_no_scaling.shape)

X_test_no_scaling = test_no_scaling.drop(['Clicked on Ad'], axis=1).reset_index(drop=True)
y_test_no_scaling = test_no_scaling['Clicked on Ad'].reset_index(drop=True)
print(X_test_no_scaling.shape, y_test_no_scaling.shape)
```

```
(800, 65) (800,)
(200, 65) (200,)
```

Data Modeling

Rakamin Academy

Ten machine learning models were created, and the following table presents their performance for each model

Models	Accuracy (Train)	Accuracy (Test)	Precision (Train)	Precision (Test)	Recall (Train)	Recall (Test)	F1 Score (Train)	F1 Score (Test)
Logistic Regression	0.972500	0.975000	0.982143	0.989691	0.962500	0.960000	0.972222	0.974619
Logistic Regression	0.500000	0.500000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
K-Nearest Neighbors	0.965000	0.960000	0.989474	0.979167	0.940000	0.940000	0.964103	0.959184
K-Nearest Neighbors	0.760000	0.640000	0.781081	0.632075	0.722500	0.670000	0.750649	0.650485
Decision Tree	1.000000	0.945000	1.000000	0.932039	1.000000	0.960000	1.000000	0.945813
Random Forest	1.000000	0.970000	1.000000	0.970000	1.000000	0.970000	1.000000	0.970000
Adaboost Classifier	0.987500	0.975000	0.992424	0.970297	0.982500	0.980000	0.987437	0.975124
XGBoost Classifier	1.000000	0.960000	1.000000	0.950980	1.000000	0.970000	1.000000	0.960396
Naive Bayes	0.880000	0.840000	0.863636	0.809091	0.902500	0.890000	0.882641	0.847619
Naive Bayes	0.701250	0.760000	0.763934	0.795455	0.582500	0.700000	0.660993	0.744681
Support Vector Machine	0.952500	0.970000	0.981383	1.000000	0.922500	0.940000	0.951031	0.969072
Support Vector Machine	0.705000	0.760000	0.790780	0.809524	0.557500	0.680000	0.653959	0.739130
Neural Network Classifier	0.973750	0.960000	0.982188	0.979167	0.965000	0.940000	0.973518	0.959184
Neural Network Classifier	0.500000	0.500000	0.500000	0.500000	1.000000	1.000000	0.666667	0.666667
GradientBoosting Classifier	1.000000	0.960000	1.000000	0.960000	1.000000	0.960000	1.000000	0.960000

- The first row represents models with feature scaling, while the second row represents models without scaling.
- **Decision Tree, Random Forest, AdaBoost Classifier, XGBoost Classifier,** and **Gradient Boosting Classifier** models show consistent performance whether data is scaled or not. However, for other models, performance is better with feature scaling.
- The top-performing models are **AdaBoost Classifier; Logistic Regression, K-Nearest Neighbors**, and **Neural Network Classifier** (with data scaling).

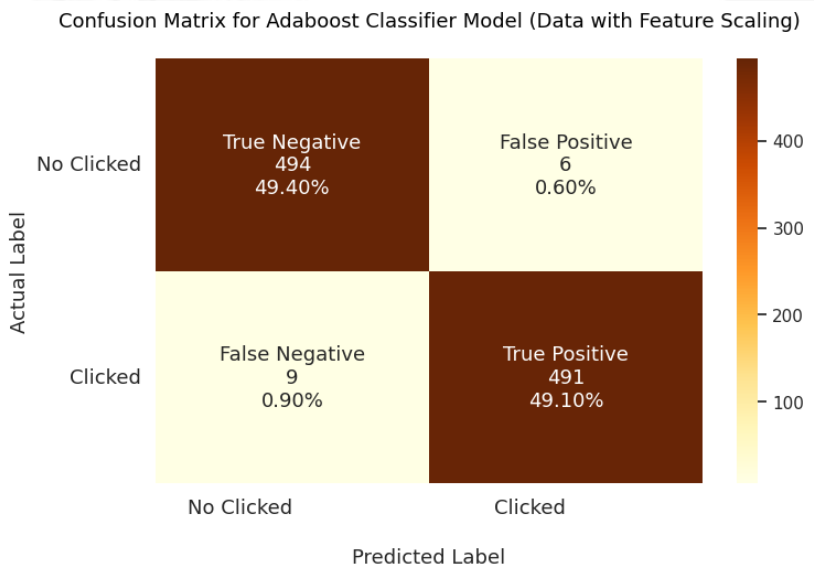
For more details, you can refer to the Jupyter notebook here

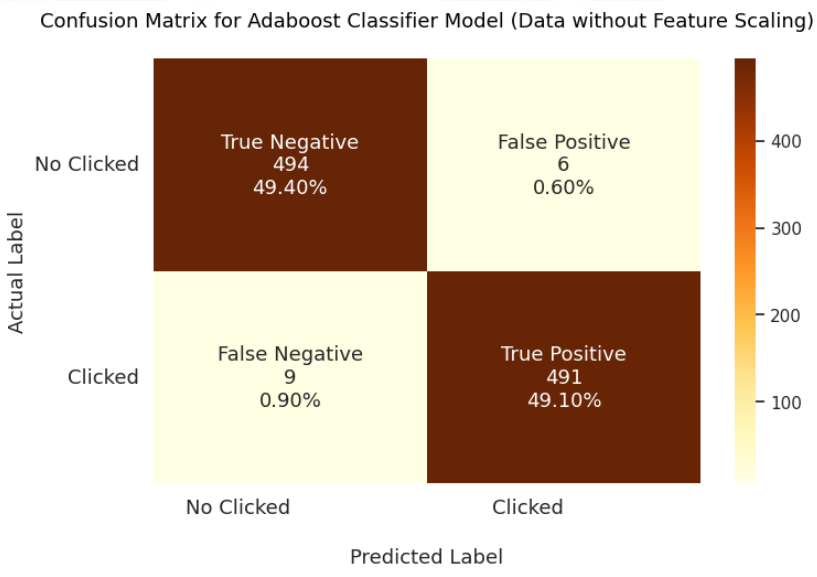
Data Modeling

Model	Before Hyperparameter Tuning						After Hyperparameter Tuning					
	Accuracy (Train)	Accuracy (Test)	Recall (Train)	Recall (Test)	Precision (Train)	Precision (Test)	Accuracy (Train)	Accuracy (Test)	Recall (Train)	Recall (Test)	Precision (Train)	Precision (Test)
Logistic Regression	0.972	0.975	0.962	0.96	0.982	0.989	0.973	0.975	0.965	0.98	0.982	0.970
K-Nearest Neighbors	0.965	0.96	0.94	0.94	0.989	0.979	0.953	0.965	0.917	0.93	0.989	1.000
Adaboost Classifier	0.987	0.975	0.982	0.98	0.992	0.970	1.000	0.96	1.000	0.97	1.000	0.95
Neural Network Classifier	0.973	0.96	0.965	0.94	0.982	0.979	1.000	0.965	1.000	0.96	1.000	0.969

The most stable model is the **AdaBoost Classifier** without hyperparameter tuning, which achieves the best recall score as the primary metric, and that is consistently close between the training and testing datasets.

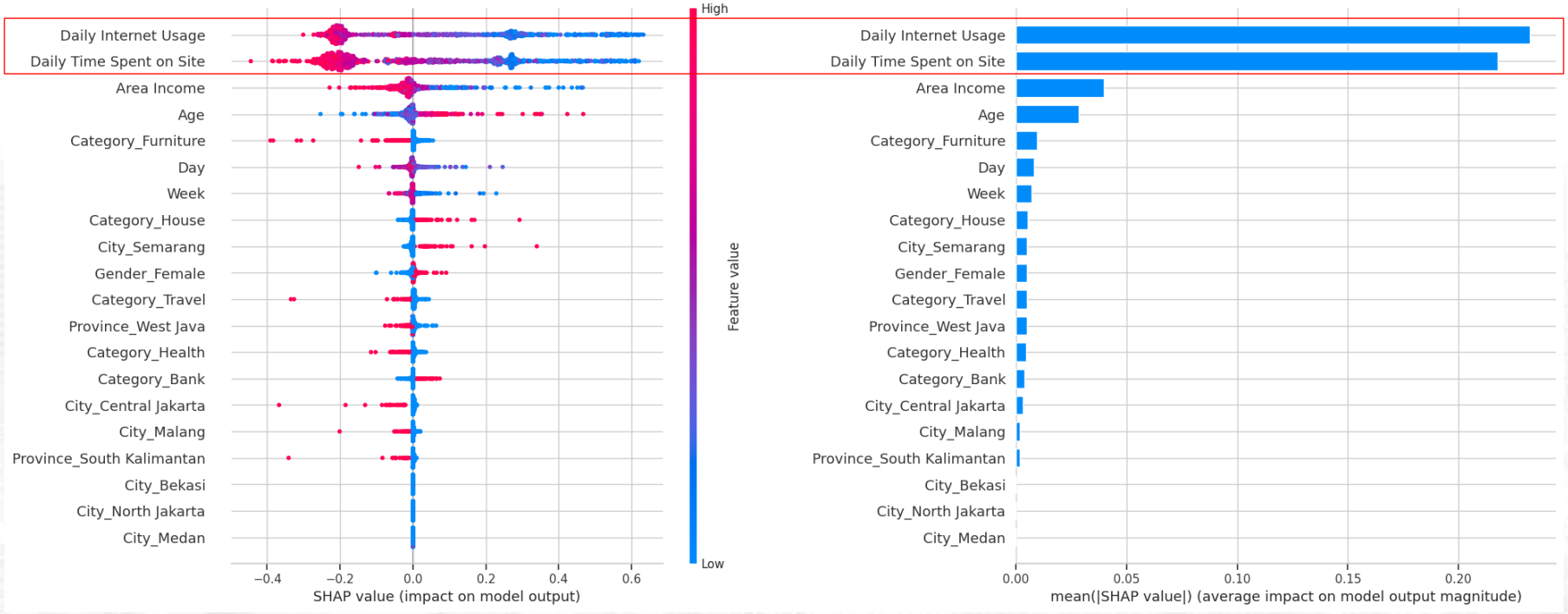
Data Modeling





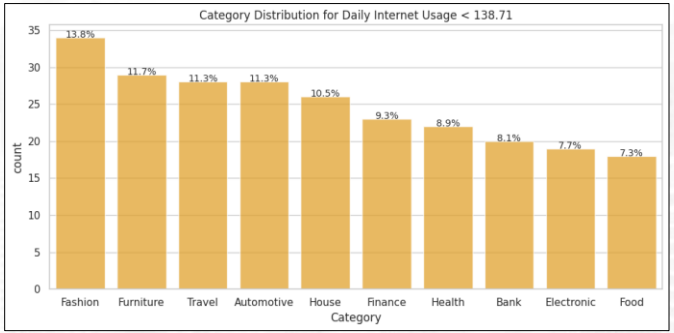
The model's performance and the confusion matrix values are identical for data with and without feature scaling, likely due to a narrow data range and a relatively small dataset. To minimize future error risks, especially with more complex data or significant scale variations, it is advisable to **choose the model with feature scaling**.

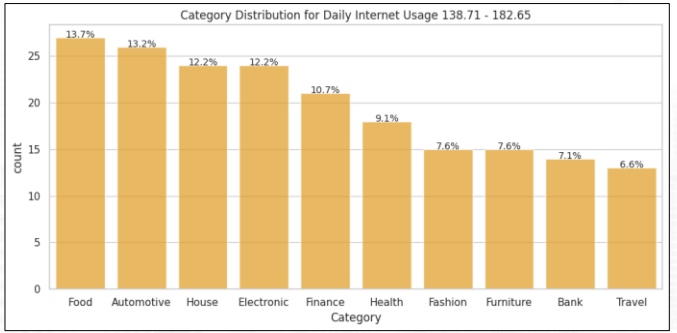
Data Modeling: Feature Importance

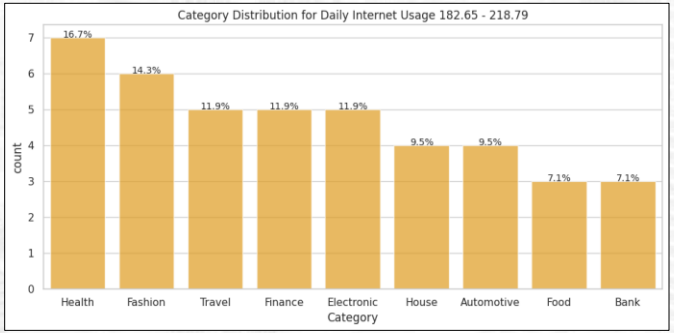


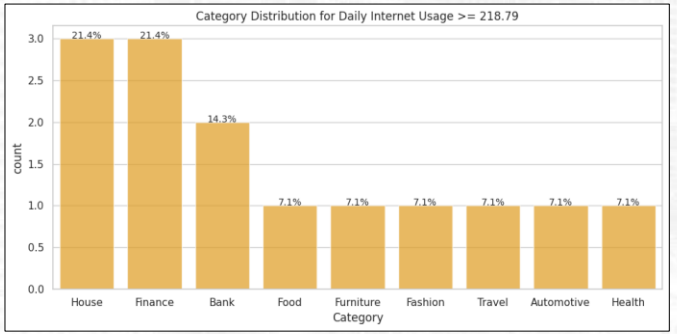
- The shorter the daily internet usage and daily time spent on the site, the higher the likelihood that a customer will click on the advertisement.

Business Insight









Across several time segments, no discernible ad category preference is evident, as all ad categories receive clicks in proportions that do not significantly vary.

For more details, you can refer to the Jupyter notebook here

Business Recommendation

- **Simplified and Compact Ad Messages**:
  For example, place the core ad message at the top of the image to enhance attractiveness and message clarity.
- **Image and Source Code Optimization:**
  Based on customer behavior, ad clicks are more likely when daily internet usage and website loading times are lower. Optimize images and source code to speed up loading and maximize ad effectiveness.
- **Time-Centric Ad Messaging:**
  Incorporate time-sensitive messages in ads, such as 'visit now' or 'valid for the next 30 minutes.' This creates a sense of urgency and encourages customer action. The 30-minute timeframe is selected based on the lowest daily customer spending time (±32 minutes).
- **Cookie System Implementation, e.g., Persistent Cookies:**
  Considering varying customer ad preferences where all categories receive clicks in proportions that do not significantly differ, implement a cookie system to track user preferences and behavior while still adhering to privacy regulations and providing options for customers to consent or reject these cookies.

Business Simulation

Rakamin Academy

Total Customers = 1.000
Total Clicked on Ads = 500

BEFORE MODEL

50%

Assuming All TP Customers Continue to Click

Click per Customer = TP / Total Target Customers

TP = 491
FP = 6
Total Target Customers (TP+FP) = 497
Total Clicked on Ads (TP) = 491

AFTER MODEL

98.8%

Business Simulation

640,000,000 IDR

BEFORE MODEL

Total Customers	1,000 Customers
Total Clicked on Ads	500
Cost (Cost per Click = 60,000 IDR)	60,000,000 IDR
Revenue (Revenue per Click = 1,400,000 IDR)	700,000,000 IDR
Profit	**640,000,000 IDR**

PROFIT
17,000,000 IDR

COST
30,180,000 IDR

657,000,000 IDR

AFTER MODEL

Total Target Customers	497 Customers
Total Clicked on Ads	491
Cost (Cost per Click = 60,000 IDR)	29,820,000 IDR
Revenue (Revenue per Click = 1,400,000 IDR)	687,400,000 IDR
Profit	**657,000,000 IDR**

Summary

Implementing business recommendations for customers predicted to click on ads can increase the click-through rate per customer by nearly 50%, shifting from the initial 50% of total customers to 98.8% of the total target customers. This also has the potential to boost profits by approximately 17,000,000 IDR, with costs amounting to 30,180,000 IDR (a decrease of over 50% from the initial cost).