Predict Clicked Ads Customer Classification by using Machine Learning

Created by:
Oktavian Dwi Putra
oktavian.dwiputra@gmail.com
www.linkedin.com/in/oktaviandp/

Result-oriented professional with background in SEO and a strong desire to transition into the field of Data Science. Possessing a solid foundation in statistics, machine learning, and data analysis. Eager to apply my analytical mindset, problem-solving abilities, and passion for data-driven insights to drive meaningful outcomes as a Data Scientist.

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

Overview

"A company in Indonesia wants to know the effectiveness of the advertisement they broadcast. This is important for the company to know how successful the advertisement being marketed is so that it can attract customers to see the advertisement.

By analyzing the historical advertisement data and finding insights and patterns that occur, it can help companies determine marketing targets. The focus of this case is to create a machine learning classification model that can be used to determine the right target customers."

Goals & Objectives

Goals:

1. Increase advertising effectiveness to above 90%.

2. Find out the factors that influence customers to click on ads.

Objectives:

1. Analyze historical advertisement data to find insights and patterns that occur.

2. Create a machine learning classification model to determine the right target customers.

1. Dataset Info

Using **df.info()** syntax, from the image on the right we can conclude:

- Dataset consists of **1000 rows**, **10 features** and **1 Unnamed: 0** column which is the ID customers that need to be **removed**.
- Dataset consists of **3 data types**: int64, float64, and object.
- **Timestamp** feature should be changed into **datetime** data type.
- The target variable which is **Clicked on Ad** is a categorical data and should be **converted to numerical data**.
- There are **four columns** that have **null values**.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 11 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Unnamed: 0                1000 non-null   int64
 1   Daily Time Spent on Site  987 non-null    float64
 2   Age                       1000 non-null   int64
 3   Area Income               987 non-null    float64
 4   Daily Internet Usage      989 non-null    float64
 5   Male                      997 non-null    object
 6   Timestamp                 1000 non-null   object
 7   Clicked on Ad             1000 non-null   object
 8   city                      1000 non-null   object
 9   province                  1000 non-null   object
 10  category                  1000 non-null   object
dtypes: float64(3), int64(2), object(6)
memory usage: 86.1+ KB
```

Source code: here

2. Descriptive Statistics

a. Numerical Data

	count	mean	std	min	25%	50%	75%	max	skewness
Daily Time Spent on Site	987.0	6.492952e+01	1.584470e+01	32.60	5.127000e+01	6.811000e+01	7.846000e+01	9.143000e+01	-0.369756
Age	1000.0	3.600900e+01	8.785562e+00	19.00	2.900000e+01	3.500000e+01	4.200000e+01	6.100000e+01	0.479142
Area Income	987.0	3.848647e+08	9.407999e+07	97975500.00	3.286330e+08	3.990683e+08	4.583554e+08	5.563936e+08	-0.644302
Daily Internet Usage	989.0	1.798636e+02	4.387014e+01	104.78	1.387100e+02	1.826500e+02	2.187900e+02	2.670100e+02	-0.031395

Observation:

- There are **no invalid values** among the columns used.
- Based on the mean, median, and skewness values, the **Area Income** column has **right-skewed** (positively skewed) distribution.
- Based on the min values the **Area Income** column has outliers.

Source code: here

2. Descriptive Statistics

b. Categorical Data

	count	unique	top	freq
Gender	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

Observation:

- The **Clicked on Ad** column (variable target) has **balanced class** so does not require any treatment.
- The city, province, and category columns have **high cardinality** (have many unique values) so they are likely to be **removed**.

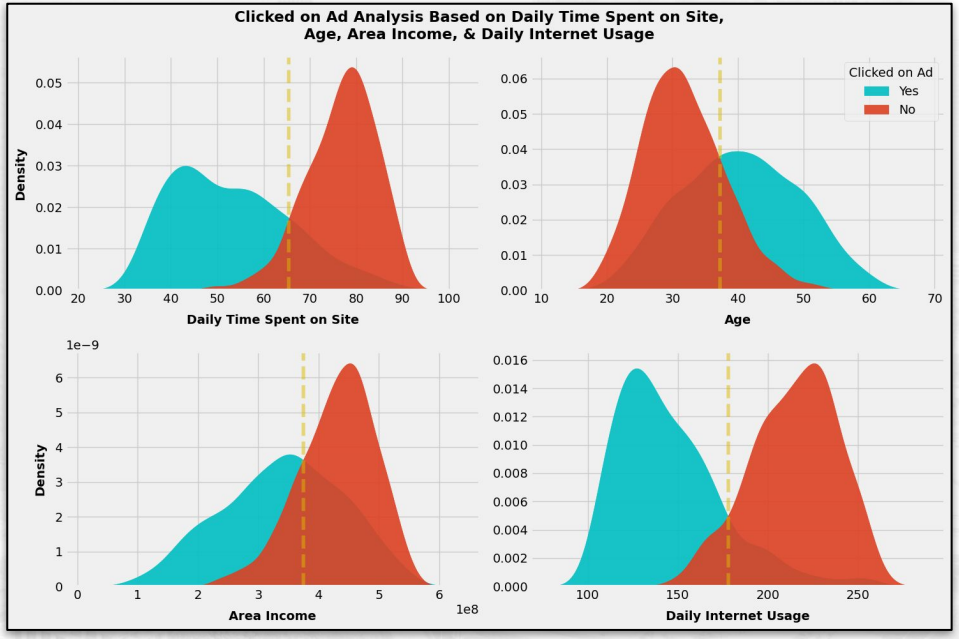
Source code: here

3. Univariate Analysis

a. Numerical Data

Observation:

- The **more time spent** on the site or the internet, the **less likely** a customer will click on an ad.
- The **older** the customer, the **more likely** a customer will click on an ad.
- The **higher** area income of customer, the **less likely** a customer will click on an ad.



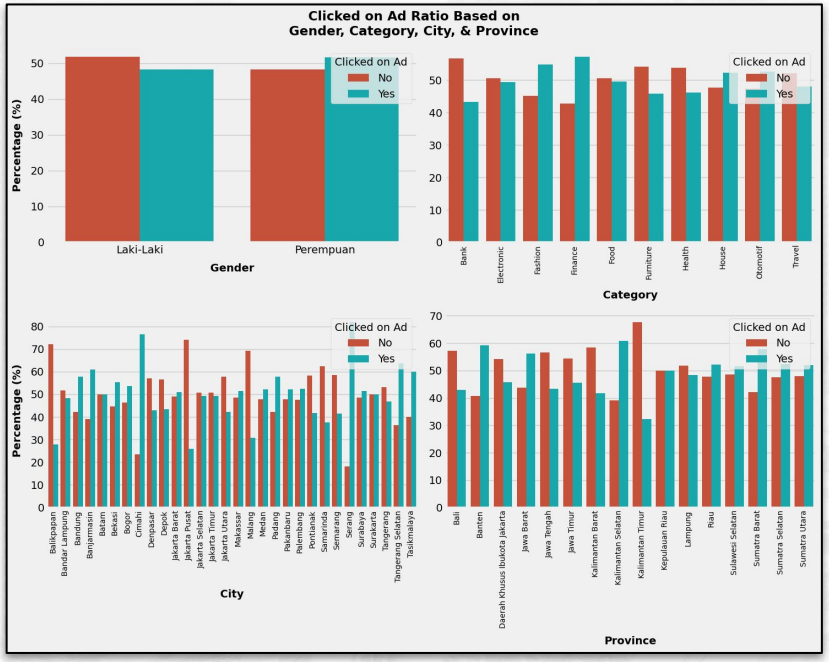
Source code: here

3. Univariate Analysis

b. Categorical Data

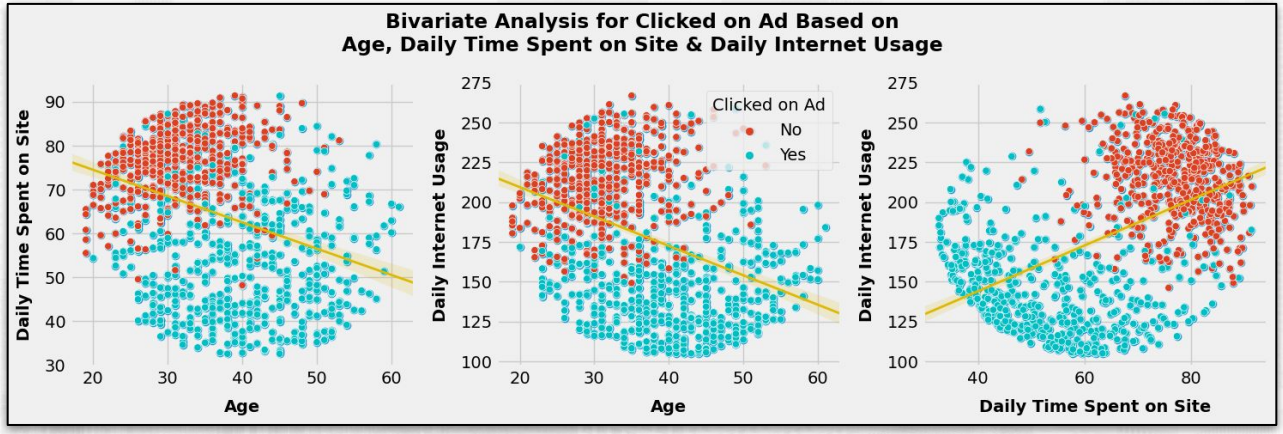
Observation:

- **Perempuan** (female) has a slightly higher probability of **clicking** on Ad than Laki-laki (male).
- Each ad category has a fairly similar click ratio with the **highest** ad category clicked being **Finance** and the **lowest** being **Bank**.
- The city with the **highest** click ratio is **Serang** and the **lowest** is **Jakarta Pusat**.
- The top 3 provinces with the **highest** click ratio are **Kalimantan Selatan**, **Banten**, **Sumatra Barat**.



Source code: here

4. Bivariate Analysis



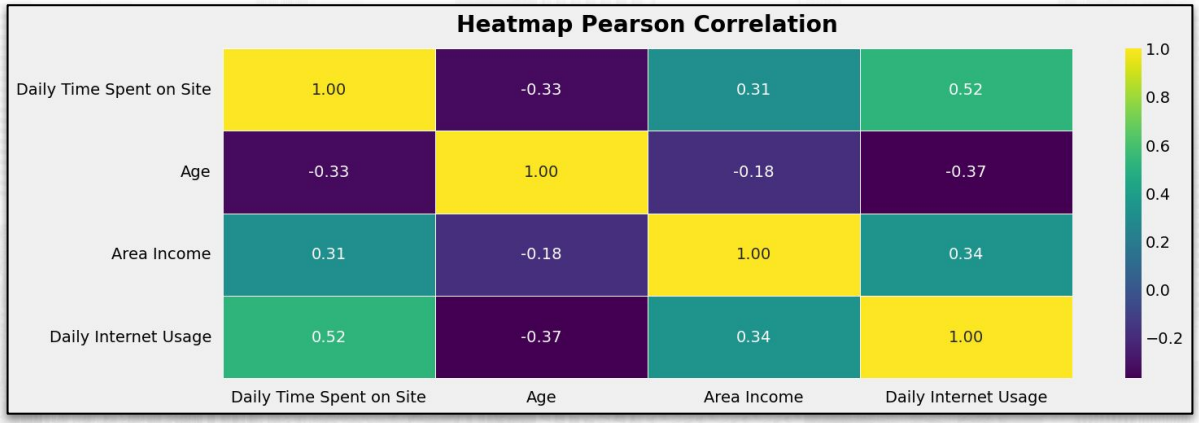
Observation:

- Age with Daily Time Spent on Site or Daily Internet Usage have a **negative correlation**. This means that the **older** the customer, the **less time** they spend on the site or the internet.
- Meanwhile, Daily Time Spent on Site and Daily Internet Usage have a **positive correlation**. This means that the **more time** spent on the internet, the **more time** will be spent on the site too.

Source code: here

5. Multivariate Analysis

a. Pearson Correlation



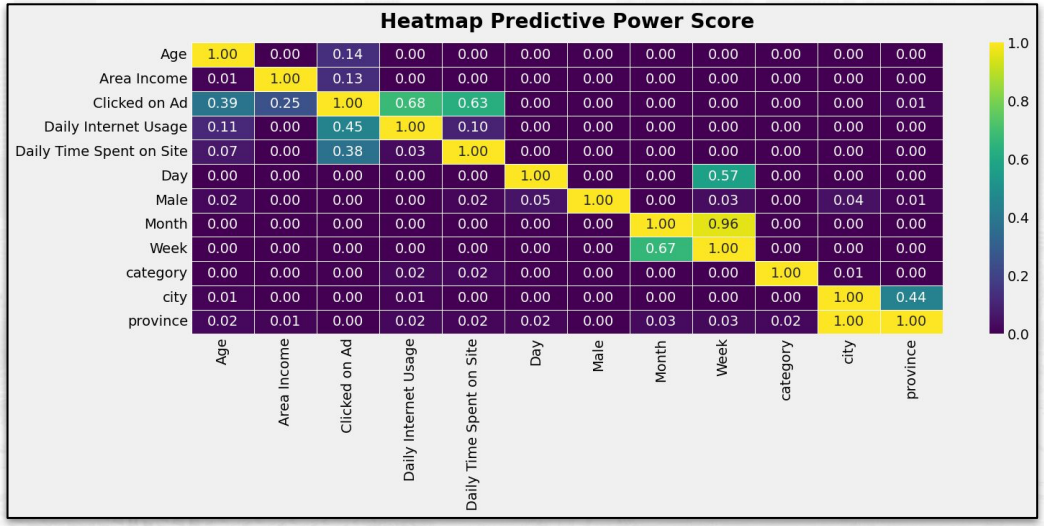
Based on the heatmap above, there are **no features** that are **redundant** or have high correlation (>= 0.7) between them. Therefore, all features can be used for modeling. However, by using Pearson correlation, we cannot determine the relationship between features and the target variable because the **target variable is categorical data**. Therefore, we will use **PPS (Predictive Power Score)** to calculate the relationship between features and the target variable.

Source code: here

5. Multivariate Analysis

b. Predictive Power Score

Based on the heatmap above, the features that are **related** to the target variable (Clicked on Ad) and will be used for modeling are **Daily Internet Usage**, **Daily Time Spent on Site**, **Age**, and **Area Income** because they have **predictive power score >= 0.05** with the target variable.



Source code: here

Data Cleaning & Preprocessing

The data preprocessing required before the data is used in the modeling process are:

1. **Impute** null values in the **Area Income** column with **median** because it has a **skewed** distribution and **Daily Internet Usage** and **Daily Time Spent on Site** columns with **mean** because they have almost **symmetric** distributions.
2. Dataset **does not have** duplicated data.
3. **Encode** the target variable (Clicked on Ad) to **numerical data**, 'No': 0 and 'Yes': 1.
4. Split the data into 70:30 proportions, **70% for training** and **30% for testing**.
5. Handle **outliers** for the **Area Income** column in the training data.
6. Conduct **normalization** process for the features used in the training and testing data.

Source code: here

Data Cleaning & Preprocessing

Rakamin Academy

1. Handle Missing Values

```
# Imputation with median
data['Area Income'].fillna(data['Area Income'].median(), inplace = True)

# Imputation with mean
data['Daily Internet Usage'].fillna(data['Daily Internet Usage'].mean(), inplace = True)
data['Daily Time Spent on Site'].fillna(data['Daily Time Spent on Site'].mean(), inplace = True)

# Show number of null values
data.isnull().sum()
```

2. Duplicated Data

```
# Show duplicated data
data.duplicated().sum()
```

```
0
```

3. Encode Categorical Column

```
# Encode the categorical column
data['Clicked on Ad'] = data['Clicked on Ad'].replace({'No': 0, 'Yes': 1})
```

Source code: here

Data Cleaning & Preprocessing

Rakamin Academy

4. Split Data

```
# Import library
from sklearn.model_selection import train_test_split

# Split data to features and target
X = data.drop(columns = 'Clicked on Ad')
y = data['Clicked on Ad']

# Split data to the data train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)

# Show the data train and test size
print(f'Data train : {X_train.shape[0]} rows')
print(f'Data test  : {X_test.shape[0]} rows')
```

```
Data train : 700 rows
Data test  : 300 rows
```

5. Handle Outliers

```
# Remove the outliers using IQR method
print("Number of rows:\n---------------")
print(f"BEFORE outliers removed: {len(X_train)}")

# Merge the data train
train = X_train.join(y_train)

# Calculate IQR
q1 = np.percentile(train['Area Income'], 25)
q3 = np.percentile(train['Area Income'], 75)
iqr = q3 - q1

# Calculate the lower and upper bounds
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr

# Filter the data
train = train[(train['Area Income'] >= lower_bound) & (train['Area Income'] <= upper_bound)]

# Split the data train
X_train = train.drop(columns = 'Clicked on Ad')
y_train = train['Clicked on Ad']

print(f"AFTER outliers removed: {len(X_train)}")
```

```
Number of rows:
---------------
BEFORE outliers removed: 700
AFTER outliers removed: 689
```

6. Normalization

```
# Import library
from sklearn.preprocessing import MinMaxScaler

# Create a Min-Max scaler
scaler = MinMaxScaler()

# Normalization process
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Source code: here

1. Experimental Results

1.1. Before Normalization

- **Tree-based** models have **far better performance** than **distance-based** models.
- The **best** performance models are **Decision Tree**, **Random Forest**, and **Extra Trees** with the highest accuracy.
- The **worst** performance models are **KNNeigbors** and **Logistic Regression** with the lowest accuracy.
- The **longest** time elapsed occured on **Random Forest**, **Extra Trees** and **XGBoost** models.

	Model	Acc	Prec	Recall	Time Elapsed
0	Decision Tree	0.953333	0.972973	0.935065	0.599740
1	Random Forest	0.953333	0.954545	0.954545	43946.556562
2	ExtraTress	0.953333	0.979452	0.928571	37221.365716
3	AdaBoost	0.943333	0.941935	0.948052	2.004584
4	XGBoost	0.940000	0.947368	0.935065	11219.605488
5	KNNeighbors	0.680000	0.750000	0.564935	1389.589961
6	Logistic Regression	0.616667	0.574713	0.974026	7.244295

Source code: here

1. Experimental Results

1.2. After Normalization

	Model	Acc (Normalized)	Prec (Normalized)	Recall (Normalized)	Time Elapsed (Normalized)
0	KNNeighbors	0.956667	0.986207	0.928571	1154.110018
1	Decision Tree	0.953333	0.972973	0.935065	0.270079
2	Random Forest	0.953333	0.954545	0.954545	42926.467355
3	ExtraTress	0.953333	0.979452	0.928571	39691.231115
4	AdaBoost	0.943333	0.941935	0.948052	1.777168
5	XGBoost	0.940000	0.947368	0.935065	10764.285743
6	Logistic Regression	0.926667	1.000000	0.857143	0.189288

- After the dataset was **normalized**, there was a **significant change** in the performance of the **distance-based** models.
- The **best** performance model changes to **KNNeighbors model**.
- The **worst** performance model still **Logistic Regression** although its performance increase significantly.
- The **longest** time elapsed still occurred on **Random Forest**, **Extra Trees** and **XGBoos**t models.

Source code: here

1. Experimental Results

1.3. Model Comparison

	Model	Acc	Acc (Normalized)	Δ Acc	Prec	Prec (Normalized)	Δ Prec	Recall	Recall (Normalized)	Δ Recall	Time Elapsed	Time Elapsed (Normalized)	Δ Time Elapsed
0	KNNeighbors	0.680000	0.956667	0.276667	0.750000	0.986207	0.236207	0.564935	0.928571	0.363636	1389.589961	1154.110018	-235.479943
1	Decision Tree	0.953333	0.953333	0.000000	0.972973	0.972973	0.000000	0.935065	0.935065	0.000000	0.599740	0.270079	-0.329662
2	Random Forest	0.953333	0.953333	0.000000	0.954545	0.954545	0.000000	0.954545	0.954545	0.000000	43946.556562	42926.467355	-1020.089206
3	ExtraTress	0.953333	0.953333	0.000000	0.979452	0.979452	0.000000	0.928571	0.928571	0.000000	37221.365716	39691.231115	2469.865398
4	AdaBoost	0.943333	0.943333	0.000000	0.941935	0.941935	0.000000	0.948052	0.948052	0.000000	2.004584	1.777168	-0.227415
5	XGBoost	0.940000	0.940000	0.000000	0.947368	0.947368	0.000000	0.935065	0.935065	0.000000	11219.605488	10764.285743	-455.319745
6	Logistic Regression	0.616667	0.926667	0.310000	0.574713	1.000000	0.425287	0.974026	0.857143	-0.116883	7.244295	0.189288	-7.055007

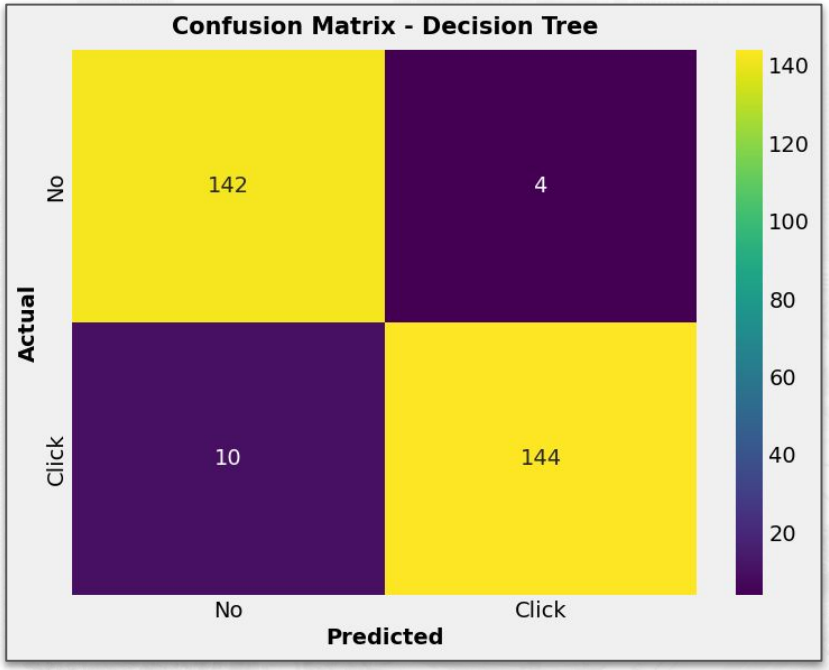
- Overall, all models **perform better** after the dataset **normalized** based on the metrics evaluation and also the time elapsed.
- The chosen model is **Decision Tree** model because it has one of the **highest accuracy** and the **fastest computation process**.

Source code: here

2. Confusion Matrix

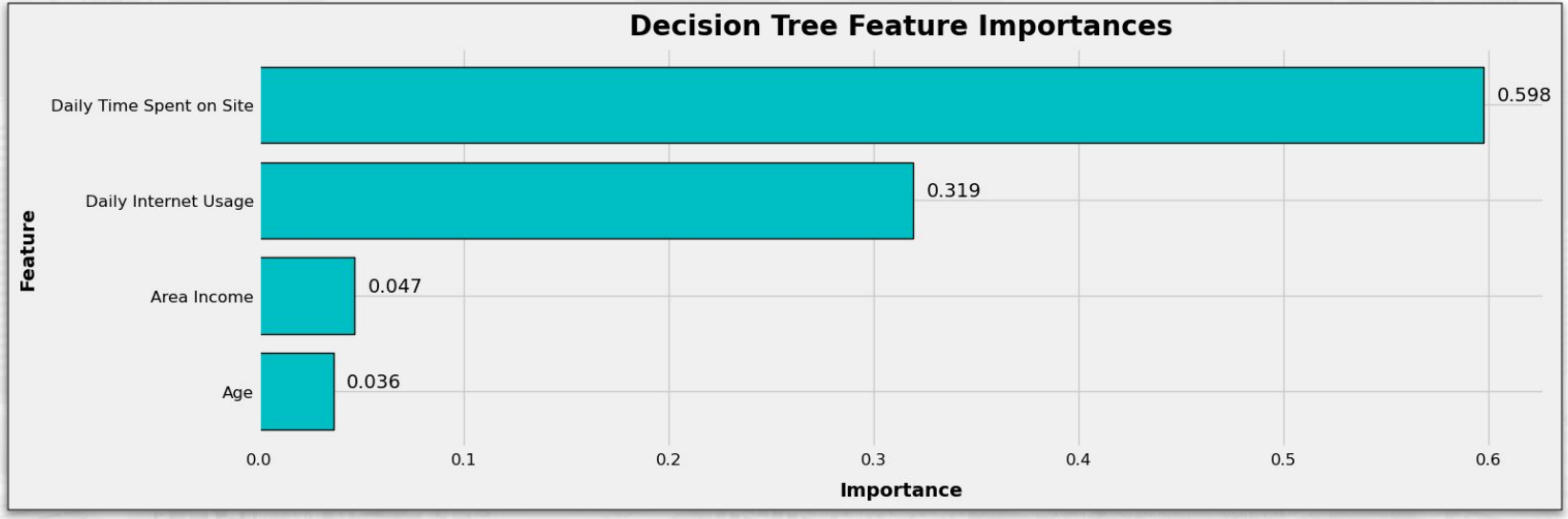
By using the results of *hyperparameter tuning* for the Decision Tree model, we train the model again to get a **confusion matrix** as shown on the right, with the following results:

- **True Positive**: Predicted to click on the ad and it turned out to be correct 144 times.
- **True Negative**: Predicted not to click on the ad and it turned out to be correct 142 times.
- **False Positive**: Predicted to click on the ad and turned out to be wrong by 4 times.
- **False Negative**: Predicted not to click on the ad and turned out to be wrong 10 times.



Source code: here

3. Feature Importances



Based on the feature importances in the image above, we can see that **Daily Time Spent on Site** is the most important feature, followed by the **Daily Internet Usage** feature in second place which determine whether **users click on ads or not**.

Source code: here

1. Business Recommendation

Based on the **insight from EDA** and **feature importances**, we can provide business recommendations such as:

1. **Content Optimization**
   Because the higher **Daily Time Spent on Site** and **Daily Internet Usage** the less likely user will click on ads, then we need create ad contents that are **engaging** and **relevant** to the target user and ensure that the messaging and visuals of the ads **align with the interests and needs** of the user.

2. **Targeted Pricing Strategies**
   Because the **lower** Area Income the **more likely** user will click on ads, we can implement targeted pricing strategies that **align with the income levels** of the target audience. This may involve creating **special pricing tiers**, **discounts**, or **bundled offerings**. Consider developing and promoting **affordable products** or **services** for the users with low area income.

3. **Age-Targeted Marketing Campaigns**
   Because the **older** the user the **more likely** user will click on ads, then we can develop targeted marketing campaigns specifically designed to resonate with **older demographics**. We can create the messages, visuals, and offers to align with the **preferences** and **interests** of older users.

Source code: here

2. Business Simulation

Assumption:

Cost per Mille (CPM) = Rp.100,000
Revenue per Ad Clicked = Rp.2,000

Conclusion:

From the results on the right, it can be seen that after we used the machine learning model, **the ad performance increased**.

Click-Through Rate (CTR) increased 45% from **50% to 95**% and **total profit** increased 100% from **Rp.900,000** to **Rp.1,800,000**.

Before Using Machine Learning Model:

Number of Users Advertised:
User = 1,000

Click-Through Rate (CTR):
500/1,000 = 0.5

Total Cost:
CPM = Rp.100,000

Total Revenue:
CTR x Number of Users Advertised x Revenue per Ad Clicked = 0.5 x 1,000 x 2,000 = Rp.1,000,000

Total Profit:
Total Revenue - Total Cost = **Rp.900,000**

After Using Machine Learning Model:

Number of Users Advertised:
User = 1,000

Click-Through Rate (CTR):
Precision = 0.95

Total Cost:
CPM = Rp.100,000

Total Revenue:
CTR x Number of Users Advertised x Revenue per Ad Clicked = 0.95 x 1,000 x 2,000 = Rp.1,900,000

Total Profit:
Total Revenue - Total Cost = **Rp.1,800,000**

Source code: here