Predict Clicked Ads Customer Classification by using Machine Learning

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

Created by:
Muhammad Cikal Merdeka
Email : mcikalmerdeka@gmail.com
LinkedIn : linkedin.com/in/mcikalmerdeka
Github : github.com/mcikalmerdeka

Dedicated entry-level data scientist with analytical and experimental background of Physics. My graduation 2023, a pivotal year marked by significant advancements in artificial intelligence with the introduction of GPT-4 and other generative AI models, has fueled my curiosity and excitement to delve into the field of data. I have comprehensive grasp of data science methodology from business understanding to modelling process with proficiency in **Python, SQL, Tableau, Power BI, Looker Studio and other tools** related to data analytics workflow from several coursework and bootcamps.

Data Cleaning & Preprocessing

Drop Unnecessary Columns and Rename Columns

- Unnamed: 0 columns in the dataset is initially dropped because it's just ID column and it doesn't store related information to our model, keeping them will just increase the dimension later.

- Some columns are renamed to match the format for the entire dataframe and also add more information/context to them. Columns that renamed were : (original → new name)

  - ❑ Male → Gender
  - ❑ Timestamp → Visit Time
  - ❑ city → City
  - ❑ province → Province
  - ❑ category →Category

For more details, you can view the Jupyter notebook here.

Identifying Missing and Duplicated Values

	Feature	Data Type	Null Values	Null Percentage (%)	Duplicated Values
0	Daily Time Spent on Site	float64	13	1.3	0
1	Age	int64	0	0.0	0
2	Area Income	float64	13	1.3	0
3	Daily Internet Usage	float64	11	1.1	0
4	Gender	object	3	0.3	0
5	Visit Time	object	0	0.0	0
6	City	object	0	0.0	0
7	Province	object	0	0.0	0
8	Category	object	0	0.0	0
9	Clicked on Ad	object	0	0.0	0

```python
# Impute missing values
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['Gender'] = df['Gender'].fillna(df['Gender'].mode()[0])
```

- Columns that missing values are Daily Time Spent on Site, Area Income, Daily Internet Usage, and Gender.
- We will handle them by **imputation with mean, median, and mode values** considering the distribution of the values based on task 1.
- No duplicated values in this dataset.

Rakamin Academy

Feature Engineering and Extraction

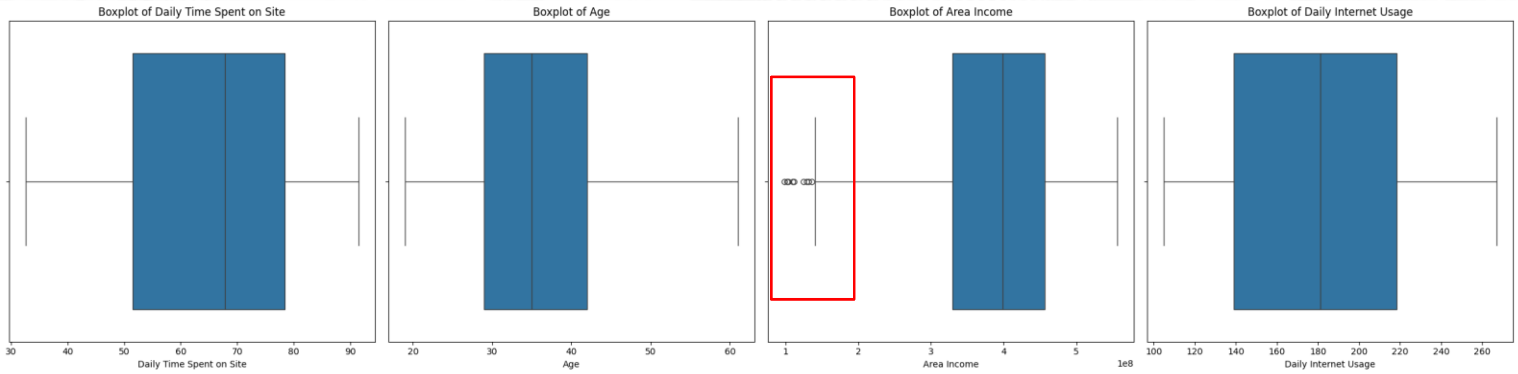
- Visit Time feature data type is corrected to datetime and some features that are extracted from the original Visit Time are Visit Month, Visit Week, Visit Day, Visit Hour, and Is Visit Day Weekend. The year component is not extracted because we knew form the EDA that the data is only 6 month period ranging from January to July.
- Some feature are also engineered from original features based on the grouping of the values range, they are Age Group and Area Income Group .

Engineered and Extracted Features

Age Group	Area Income Group	Visit Month	Visit Week	Visit Day	Visit Hour	Is Visit Day Weekend
Senior Adult	Medium-High Income	7	29	19	8	0
Middle Adult	Medium-High Income	3	12	27	2	1
Middle Adult	Medium-High Income	3	12	23	19	0
Middle Adult	High Income	3	11	15	11	0
Middle Adult	High Income	2	7	20	0	1

Outliers Handling

Handling outliers with z-score method is done only to Area Income column since it have 9 extreme positive outliers as we found on EDA (task 1) beforehand.



- Rows before removing outliers: 1000
- Rows after removing outliers: 991

Rakamin Academy

Feature Encoding

There are 6 categorical column that will be encoded based on their data type.

- Gender : Nominal type → One-hot encoding
- City : Nominal type → One-hot encoding
- Province : Nominal type → One-hot encoding
- Age Group : Ordinal type → Label encoding
- Area Income Group : Ordinal type → Label encoding

Before Encoding

	Gender	Age Group	Area Income Group
0	Perempuan	Middle Adult	Medium-High Income
1	Laki-Laki	Middle Adult	High Income
2	Perempuan	Young Adult	Medium-High Income
3	Laki-Laki	Young Adult	Medium-Low Income
4	Perempuan	Middle Adult	High Income
...	...	...	...
995	Laki-Laki	Middle Adult	High Income
996	Laki-Laki	Middle Adult	High Income
997	Laki-Laki	Senior Adult	Low Income
998	Perempuan	Young Adult	Low Income
999	Perempuan	Young Adult	Low Income

After Encoding

	Gender	Age Group	Area Income Group
0	0	1	2
1	1	1	3
2	0	0	2
3	1	0	1
4	0	1	3
...	...	...	...
995	1	1	3
996	1	1	3
997	1	2	0
998	0	0	0
999	0	0	0

* As for the one-hot encoded features are not shown because way to many unique values led to increasing in dimensions.

Those features will later be dropped in the feature selection process.

Data Cleaning & Preprocessing

Rakamin Academy

Feature Selection

- At this point we now have 70 columns which is way too many. That's why feature selection procedure need to be done. The process is through checking Spearmann correlation heatmap and analyzing feature importance (statistical test) of mutual information and chi square test with SelectKBest scikit-learn method.

- Reviewing the statistical test values of several features generated, I've decided not to use some engineered features in training the model due to multicollinearity issues with the original features. Therefore, these engineered features will only be used for analysis purposes.

- Meanwhile, some one-hot encoded features like city and province have many features with very low importance to the target, so those features will not be used.

- **Most of the features that will be used are from original features. (Check the details of the analysis on the source code, note that features stated here are not fixed yet because experiment with the model by analyzing the evaluation metrics need to be conducted later)**

- **Features that are used : Daily Time Spent on Site, Age, Area Income, Daily Internet Usage, Gender, Clicked on Ad**

Splitting Train and Test Data

- Train-test split is conducted with proportion of 70-30, which makes 693 rows go into training data while the remaining 298 rows go into test data.

```
# Split Train and Test Data
x = df_model.drop(columns=['Clicked on Ad'])
y = df_model['Clicked on Ad']

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)

#Check dimension of train and test data
print(f'Dimension of x_train data : {x_train.shape}')
print(f'Dimension of y_train data : {y_train.shape}')
print(f'Dimension of x_test data : {x_test.shape}')
print(f'Dimension of y_test data : {y_test.shape}')
```

✓ 0.0s

```
Dimension of x_train data : (693, 5)
Dimension of y_train data : (693,)
Dimension of x_test data : (298, 5)
Dimension of y_test data : (298,)
```

```
# Check distribution of target after splitting
display(y_train.value_counts())

display(y_test.value_counts())
```

✓ 0.0s

```
Clicked on Ad
1    347
0    346
Name: count, dtype: int64

Clicked on Ad
0    154
1    144
Name: count, dtype: int64
```

Feature Scaling

- After we have all the values in numerical form and conducted train-test split, we need to transform (scale) the values to ensure fair calculations, especially for features that have extreme range of values like Area Income.

- We use standardization for scaling method to ensure that all columns have mean of 0 and standard deviation of 1.

```python
# Scaling with standardization method
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()

columns_to_scale = ['Daily Time Spent on Site', 'Age', 'Area Income', 'Daily Internet Usage']
x_train[columns_to_scale] = ss.fit_transform(x_train[columns_to_scale])
x_test[columns_to_scale] = ss.transform(x_test[columns_to_scale])
```

✓ 0.0s

Before Scaling

	mean	std
Daily Time Spent on Site	-4.844610e-16	1.000722
Age	3.486068e-16	1.000722
Area Income	1.307276e-16	1.000722
Daily Internet Usage	-4.114073e-16	1.000722

After Scaling

	mean	std
Daily Time Spent on Site	6.469393e+01	1.579638e+01
Age	3.603608e+01	8.968643e+00
Area Income	3.849767e+08	9.025912e+07
Daily Internet Usage	1.794542e+02	4.393979e+01