CSE 4/587 - Data Intensive Computing
Project Phase 3
Waze User Churn Prediction: Insights for Improved User Experience

Jyothi Sravanthi Akkireddi (jakkired@buffalo.edu)-50487733

Sai Lakshmi Tejaswini Kallakuri (kallakur@buffalo.edu)- 50483774

Venkata Lakshmi Krishna Tejaswi Gudimetla (vgudimet@buffalo.edu)-50496378

Introduction:

The objective of this project is to develop a predictive model to identify users at risk of churning in the Waze application. Predicting user churn is crucial for Waze to enhance user retention and improve the overall user experience. By identifying users who are likely to churn, Waze can take proactive measures to retain them, reduce marketing costs, and increase user engagement. The results of this project will be valuable to Waze's data analytics team, marketing department, and other stakeholders interested in user retention strategies.

Dataset:

We are focusing on a dataset that is supplied as part of the Google Advanced Data Analytics Professional Certificate program courses on Coursera. According to Google, this dataset contains synthetic data created in partnership with Waze. We assume that the synthetic data used in this project is representative of real user behavior in the Waze application. This dataset includes various user behavior features, such as sessions, drives, total sessions, and more as features and retained, churned as labels. It has 14999 rows and 13 columns. Here is the URL for the dataset https://www.kaggle.com/datasets/juliasuzuki/waze-dataset-to-predict-user-churn.

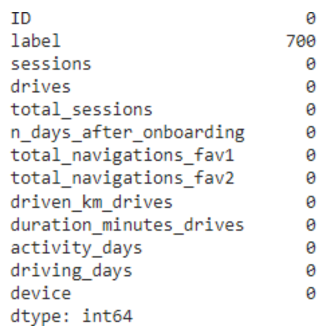
Data cleaning:

1. Removing null values from the dataset:

To ensure that your data is clean and suitable for analysis or machine learning, removing null values from a dataset is a crucial data preprocessing step. Null values can adversely affect the accuracy and reliability of your analyses.

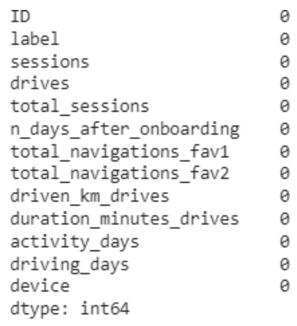
```
df.isnull().sum()
```

This function determines how many values are missing from each column in the Data Frame.



```
ID                           0
label                      700
sessions                     0
drives                       0
total_sessions               0
n_days_after_onboarding      0
total_navigations_fav1       0
total_navigations_fav2       0
driven_km_drives             0
duration_minutes_drives      0
activity_days                0
driving_days                 0
device                       0
dtype: int64
```

df.dropna(subset=['label'], inplace=True)
df.isnull().sum()
It removes rows where the 'label' column contains empty values from the Data Frame. After the drop operation, this code gives the number of missing values in each column where the 700 null values in the ‘label’ column are removed.

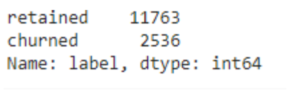


```
ID                         0
label                      0
sessions                   0
drives                     0
total_sessions             0
n_days_after_onboarding    0
total_navigations_fav1     0
total_navigations_fav2     0
driven_km_drives           0
duration_minutes_drives    0
activity_days              0
driving_days               0
device                     0
dtype: int64
```

2. Zero Value Handling (Mean Imputation):
It begins by identifying which columns in the Data Frame contain zero values, providing a list of their names as ['ID', 'sessions', 'drives', 'total_navigations_fav1', 'total_navigations_fav2', 'activity_days', 'driving_days']. Then, it calculates the mean of non-zero values for each of these columns. These means are used to replace the zero values in their respective columns. Finally, it confirms that no zero values remain in the dataframe by rechecking and printing the updated number of columns with zero values. Only ‘ID’ column is not replaced with mean values as we will drop this column in the further step.

3. Class Imbalance Handling:
Addressing a binary classification problem where the majority class is "retained" and the minority class is "churned," this code snippet seeks to handle class imbalance in a dataset. The code begins by calculating the class distribution of the 'label' column in the dataframe 'df'.

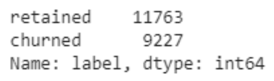
Class Distribution before balancing:



```
retained    11763
churned      2536
Name: label, dtype: int64
```

The class imbalance ratio, which is the ratio of 'retained' instances to 'churned' instances, and the imbalance ratio of 4.64 indicates a significant class imbalance, with 'retained' as the majority class. Additionally, it shuffles the newly balanced dataset. The output displays the class distribution both before and after this balancing procedure.

Class Distribution after balancing:



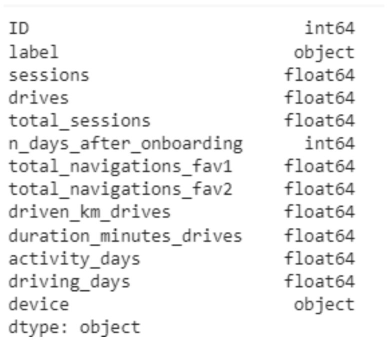
```
retained    11763
churned      9227
Name: label, dtype: int64
```

It's evident that the number of instances in the 'churned' class has increased, resulting in a more even dataset. This balancing is essential in machine learning to ensure that the model doesn't favor the majority class, improving its ability to make accurate predictions for both classes.

4. Label Encoding:

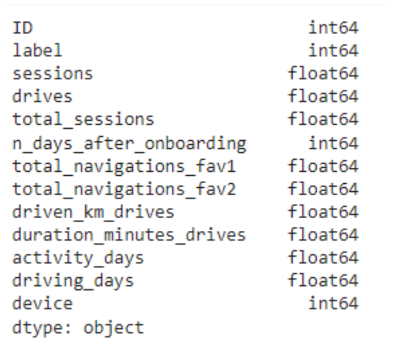
It utilizes scikit-learn library ‘LabelEncoder’ to convert the 'label' and 'device' columns within a Data Frame named ‘balanced_df’. Originally, these columns contained data of type 'object,' which typically denotes strings or categorical values. The code transforms them into integers using the LabelEncoder.

Before Encoding:



```
ID                             int64
label                         object
sessions                     float64
drives                       float64
total_sessions               float64
n_days_after_onboarding        int64
total_navigations_fav1       float64
total_navigations_fav2       float64
driven_km_drives             float64
duration_minutes_drives      float64
activity_days                float64
driving_days                 float64
device                        object
dtype: object
```

After Encoding:

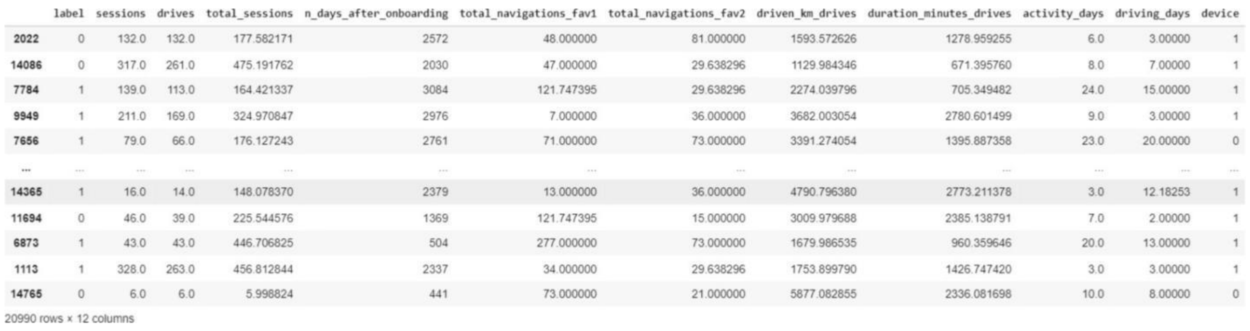


```
ID                          int64
label                       int64
sessions                  float64
drives                    float64
total_sessions            float64
n_days_after_onboarding     int64
total_navigations_fav1    float64
total_navigations_fav2    float64
driven_km_drives          float64
duration_minutes_drives   float64
activity_days             float64
driving_days              float64
device                      int64
dtype: object
```

This encoding process enables the algorithm to process these previously categorical features as numerical integers.

5. Dropping Unwanted Columns:

It is dropping a column named 'ID' from a DataFrame called 'balanced_df.' The DataFrame has 12 columns, including the 'ID' column. After executing this code,
balanced_df = balanced_df.drop('ID', axis=1) the 'ID' column will be removed from the DataFrame, resulting in a modified Data Frame with 11 columns. The balanced_df is obtained as follows:

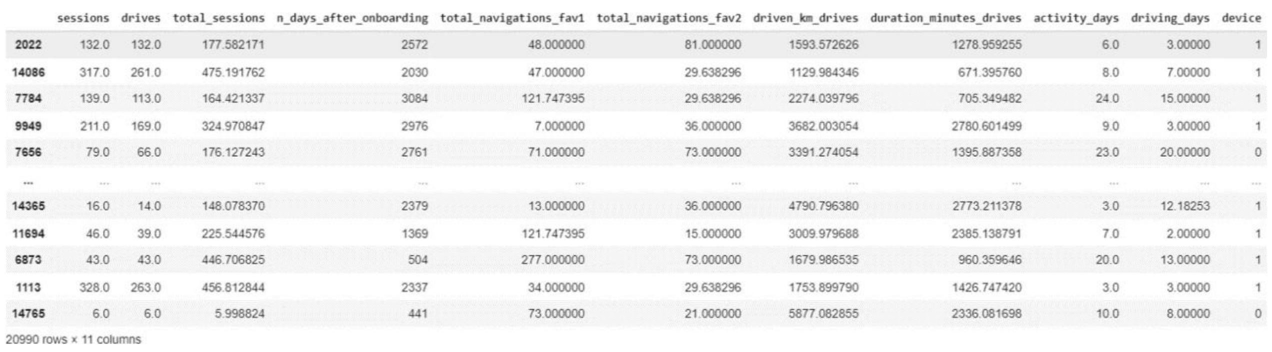


	label	sessions	drives	total_sessions	n_days_after_onboarding	total_navigations_fav1	total_navigations_fav2	driven_km_drives	duration_minutes_drives	activity_days	driving_days	device
2022	0	132.0	132.0	177.582171	2572	48.000000	81.000000	1593.572626	1278.959255	6.0	3.00000	1
14086	0	317.0	261.0	475.191762	2030	47.000000	29.638296	1129.984346	671.395760	8.0	7.00000	1
7784	1	139.0	113.0	164.421337	3084	121.747395	29.638296	2274.039796	705.349482	24.0	15.00000	1
9949	1	211.0	169.0	324.970847	2976	7.000000	36.000000	3682.003054	2780.601499	9.0	3.00000	1
7656	1	79.0	66.0	176.127243	2761	71.000000	73.000000	3391.274054	1395.887358	23.0	20.00000	0
...	...	...	...	...	...	...	...	...	...	...	...	...
14365	1	16.0	14.0	148.078370	2379	13.000000	36.000000	4790.796380	2773.211378	3.0	12.18253	1
11694	0	46.0	39.0	225.544576	1369	121.747395	15.000000	3009.979688	2385.138791	7.0	2.00000	1
6873	1	43.0	43.0	446.706825	504	277.000000	73.000000	1679.986535	960.359646	20.0	13.00000	1
1113	1	328.0	263.0	456.812844	2337	34.000000	29.638296	1753.899790	1426.747420	3.0	3.00000	1
14765	0	6.0	6.0	5.998824	441	73.000000	21.000000	5877.082855	2336.081698	10.0	8.00000	0

20990 rows × 12 columns

6. Selecting Features and Targets:

It is preparing the data for a machine learning task by separating the features (X) from the target variable (y).
X = balanced_df.drop(columns=['label'])
A new Data Frame ‘X’ excludes the 'label' column from the original 'balanced_df' Data Frame. This action is taken to distinguish the feature variables from the target variable, with the assumption that 'label' serves as the probable target variable.
y = balanced_df['label']
This line of code constructs a Series called 'y' by extracting the values found in the 'label' column of the original 'balanced_df' Data Frame. The X dataframe is given below



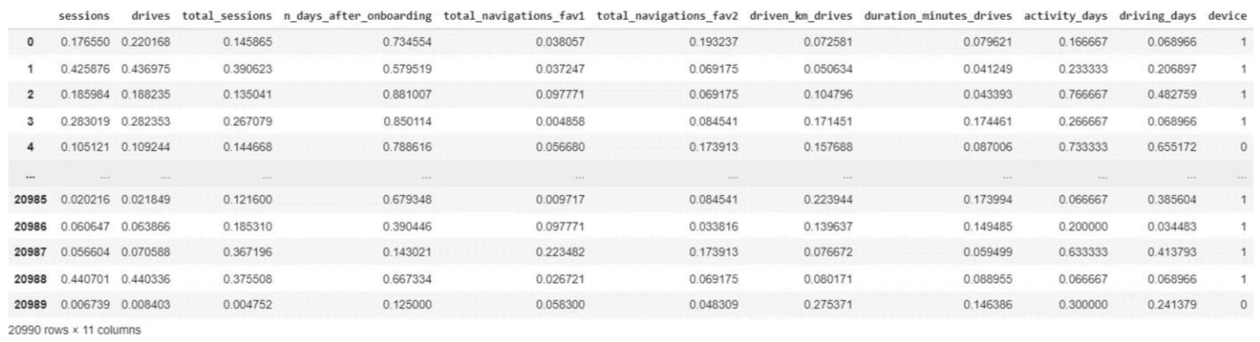
	sessions	drives	total_sessions	n_days_after_onboarding	total_navigations_fav1	total_navigations_fav2	driven_km_drives	duration_minutes_drives	activity_days	driving_days	device
2022	132.0	132.0	177.582171	2572	48.000000	81.000000	1593.572626	1278.959255	6.0	3.00000	1
14086	317.0	261.0	475.191762	2030	47.000000	29.638296	1129.984346	671.395760	8.0	7.00000	1
7784	139.0	113.0	164.421337	3084	121.747395	29.638296	2274.039796	705.349482	24.0	15.00000	1
9949	211.0	169.0	324.970847	2976	7.000000	36.000000	3682.003054	2780.601499	9.0	3.00000	1
7656	79.0	66.0	176.127243	2761	71.000000	73.000000	3391.274054	1395.887358	23.0	20.00000	0
...	...	...	...	...	...	...	...	...	...	...	...
14365	16.0	14.0	148.078370	2379	13.000000	36.000000	4790.796380	2773.211378	3.0	12.18253	1
11694	46.0	39.0	225.544576	1369	121.747395	15.000000	3009.979688	2385.138791	7.0	2.00000	1
6873	43.0	43.0	446.706825	504	277.000000	73.000000	1679.986535	960.359646	20.0	13.00000	1
1113	328.0	263.0	456.812844	2337	34.000000	29.638296	1753.899790	1426.747420	3.0	3.00000	1
14765	6.0	6.0	5.998824	441	73.000000	21.000000	5877.082855	2336.081698	10.0	8.00000	0

20990 rows × 11 columns

7. Scaling:

It scales a set of numerical features using Min-Max scaling and recombines them with a binary 'device' column. This transformation results in a Data Frame where the numerical features have been rescaled to the range [0, 1], making them suitable for various machine learning algorithms. The 'device' column remains unchanged, and the final Data Frame includes both the scaled numerical features and the original binary 'device' column, ready for further analysis or modeling.

The resultant scaled_X output is given below:

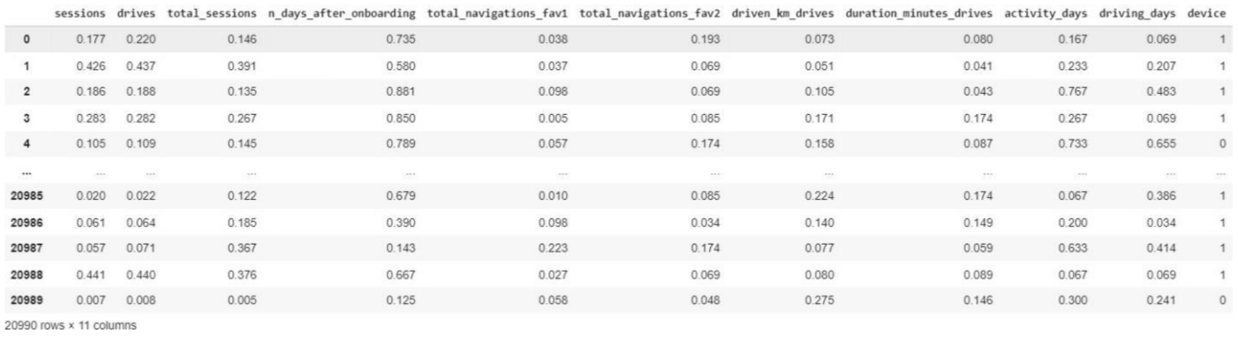


	sessions	drives	total_sessions	n_days_after_onboarding	total_navigations_fav1	total_navigations_fav2	driven_km_drives	duration_minutes_drives	activity_days	driving_days	device
0	0.176550	0.220168	0.145865	0.734554	0.038057	0.193237	0.072581	0.079621	0.166667	0.068966	1
1	0.425876	0.436975	0.390623	0.579519	0.037247	0.069175	0.050634	0.041249	0.233333	0.206897	1
2	0.185984	0.188235	0.135041	0.881007	0.097771	0.069175	0.104796	0.043393	0.766667	0.482759	1
3	0.283019	0.282353	0.267079	0.850114	0.004858	0.084541	0.171451	0.174461	0.266667	0.068966	1
4	0.105121	0.109244	0.144668	0.788616	0.056680	0.173913	0.157688	0.087006	0.733333	0.655172	0
...	...	...	...	...	...	...	...	...	...	...	...
20985	0.020216	0.021849	0.121600	0.679348	0.009717	0.084541	0.223944	0.173994	0.066667	0.385604	1
20986	0.060647	0.063866	0.185310	0.390446	0.097771	0.033816	0.139637	0.149485	0.200000	0.034483	1
20987	0.056604	0.070588	0.367196	0.143021	0.223482	0.173913	0.076672	0.059499	0.633333	0.413793	1
20988	0.440701	0.440336	0.375508	0.667334	0.026721	0.069175	0.080171	0.088955	0.066667	0.068966	1
20989	0.006739	0.008403	0.004752	0.125000	0.058300	0.048309	0.275371	0.146386	0.300000	0.241379	0

20990 rows × 11 columns

8. Setting Precision:

The "scaled_X" Data Frame initially contains columns with metrics like sessions, drives, and total sessions. After rounding the values to three decimal places, the resulting "scaled_X_rounded" Data Frame maintains the same structure but with all values rounded. This rounding can be beneficial for tasks such as improving data readability or conducting analyses that don't necessitate a high level of decimal precision. The scaled_X_rounded output is as follows:



	sessions	drives	total_sessions	n_days_after_onboarding	total_navigations_fav1	total_navigations_fav2	driven_km_drives	duration_minutes_drives	activity_days	driving_days	device
0	0.177	0.220	0.146	0.735	0.038	0.193	0.073	0.080	0.167	0.069	1
1	0.426	0.437	0.391	0.580	0.037	0.069	0.051	0.041	0.233	0.207	1
2	0.186	0.188	0.135	0.881	0.098	0.069	0.105	0.043	0.767	0.483	1
3	0.283	0.282	0.267	0.850	0.005	0.085	0.171	0.174	0.267	0.069	1
4	0.105	0.109	0.145	0.789	0.057	0.174	0.158	0.087	0.733	0.655	0
...	...	...	...	...	...	...	...	...	...	...	...
20985	0.020	0.022	0.122	0.679	0.010	0.085	0.224	0.174	0.067	0.386	1
20986	0.061	0.064	0.185	0.390	0.098	0.034	0.140	0.149	0.200	0.034	1
20987	0.057	0.071	0.367	0.143	0.223	0.174	0.077	0.059	0.633	0.414	1
20988	0.441	0.440	0.376	0.667	0.027	0.069	0.080	0.089	0.067	0.069	1
20989	0.007	0.008	0.005	0.125	0.058	0.048	0.275	0.146	0.300	0.241	0

20990 rows × 11 columns

9. Checking if there are outliers in Data Frame:

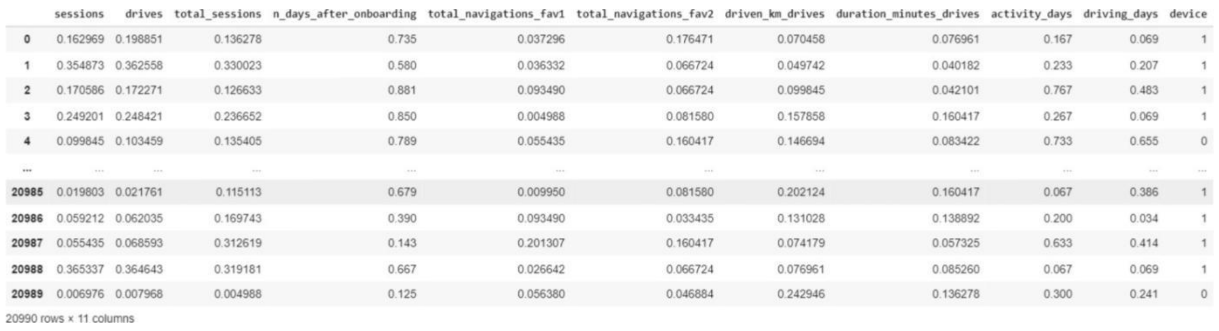
It calculates Z-scores for each value in a Data Frame (assumed to be named scaled_X_rounded) and then identifies potential outliers in the data based on a Z-score threshold of 5. The output is a Data Frame with 20,990 rows and 11 columns, where each row represents a data point, and each column represents a feature.

```
       sessions  drives  total_sessions  n_days_after_onboarding  \
0         False   False           False                    False
1         False   False           False                    False
2         False   False           False                    False
3         False   False           False                    False
4         False   False           False                    False
...         ...     ...             ...                      ...
20985     False   False           False                    False
20986     False   False           False                    False
20987     False   False           False                    False
20988     False   False           False                    False
20989     False   False           False                    False

       total_navigations_fav1  total_navigations_fav2  driven_km_drives  \
0                       False                   False             False
1                       False                   False             False
2                       False                   False             False
3                       False                   False             False
4                       False                   False             False
...                       ...                     ...               ...
20985                   False                   False             False
20986                   False                   False             False
20987                   False                   False             False
20988                   False                   False             False
20989                   False                   False             False

       duration_minutes_drives  activity_days  driving_days  device
0                        False          False         False   False
1                        False          False         False   False
2                        False          False         False   False
3                        False          False         False   False
4                        False          False         False   False
...                        ...            ...           ...     ...
20985                    False          False         False   False
20986                    False          False         False   False
20987                    False          False         False   False
20988                    False          False         False   False
20989                    False          False         False   False

[20990 rows x 11 columns]
```

10. Addressing Data Skewness:

It calculates the skewness for each column in the DataFrame and stores the results in the skewness variable. The skewness values represent the degree of asymmetry in the data distributions.
The right_skewed_columns list contains the names of specific columns that are identified as right skewed. These columns are typically positively skewed, meaning their tails extend to the right, and they have a long right tail.
For the columns listed in right_skewed_columns, the code applies the natural logarithm transformation using np.log1p. This transformation is commonly used to reduce the impact of extreme values and make the data more normally distributed. Finally, it updates the scaled_X_rounded Data Frame with the transformed values in the specified columns.

The resultant Data Frame is given below.



	sessions	drives	total_sessions	n_days_after_onboarding	total_navigations_fav1	total_navigations_fav2	driven_km_drives	duration_minutes_drives	activity_days	driving_days	device
0	0.162969	0.198851	0.136278	0.735	0.037296	0.176471	0.070458	0.076961	0.167	0.069	1
1	0.354873	0.362558	0.330023	0.580	0.036332	0.066724	0.049742	0.040182	0.233	0.207	1
2	0.170586	0.172271	0.126633	0.881	0.093490	0.066724	0.099845	0.042101	0.767	0.483	1
3	0.249201	0.248421	0.236652	0.850	0.004988	0.081580	0.157858	0.160417	0.267	0.069	1
4	0.099845	0.103459	0.135405	0.789	0.055435	0.160417	0.146694	0.083422	0.733	0.655	0
...	...	...	...	...	...	...	...	...	...	...	...
20985	0.019803	0.021761	0.115113	0.679	0.009950	0.081580	0.202124	0.160417	0.067	0.386	1
20986	0.059212	0.062035	0.169743	0.390	0.093490	0.033435	0.131028	0.138892	0.200	0.034	1
20987	0.055435	0.068593	0.312619	0.143	0.201307	0.160417	0.074179	0.057325	0.633	0.414	1
20988	0.365337	0.364643	0.319181	0.667	0.026642	0.066724	0.076961	0.085260	0.067	0.069	1
20989	0.006976	0.007968	0.004988	0.125	0.056380	0.046884	0.242946	0.136278	0.300	0.241	0

20990 rows × 11 columns

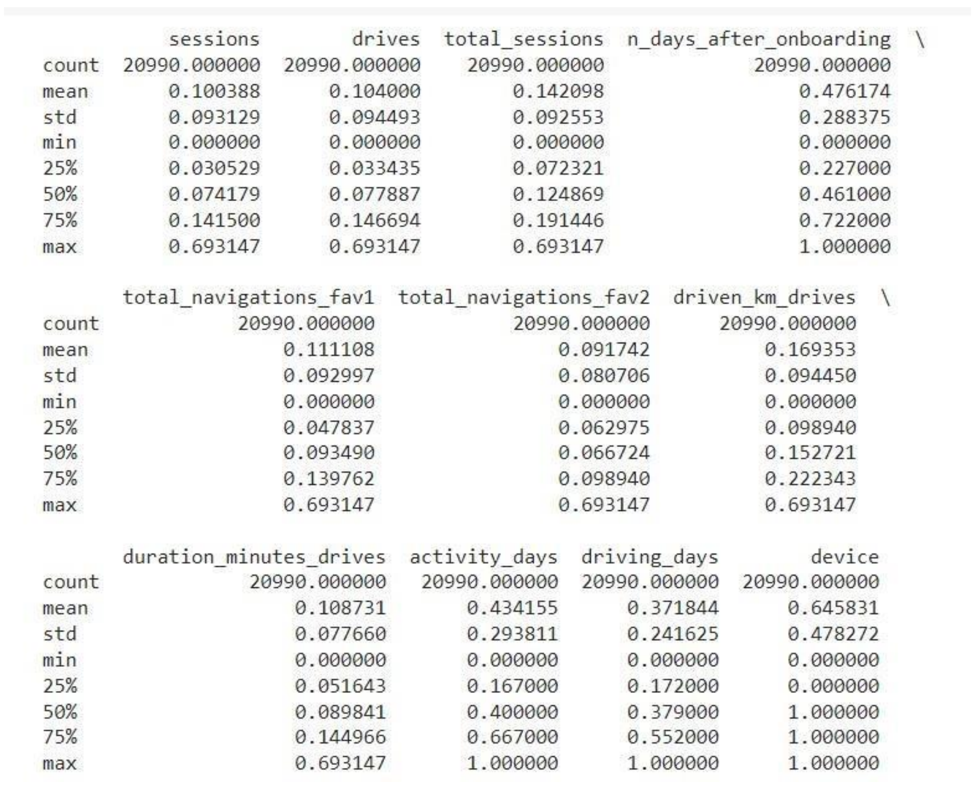
Exploratory Data Analysis (EDA)

1. **Summary Statistics:**
   The summary statistics for various features in a dataset. These statistics give insights into the distribution, central tendency, and spread of data for each feature. Let's break down the statistics for each feature:

- **Count**: This represents the number of non-missing values for each feature. For all the features listed, the count is 20990, indicating that there are no missing values and the dataset contains 20990 entries.
- **Mean**: This is the average value of the feature. For example, the mean of **sessions** is approximately 0.1038, indicating that, on average, there are about 0.1038 sessions.
- **Std (Standard Deviation)**: This measures the amount of variation or dispersion in the feature. For **sessions**, the standard deviation is approximately 0.0913, which suggests the values tend to deviate from the mean by this amount.
- **Min**: The smallest value in the feature. For all features, the minimum value seems to be 0.
- **25% (1st Quartile)**: 25% of the data falls below this value. For **sessions**, it's about 0.0305.
- **50% (Median or 2nd Quartile)**: This is the middle value of the dataset when it's sorted. For **sessions**, half of the data is below 0.0742 and half is above.
- **75% (3rd Quartile)**: 75% of the data falls below this value. For **sessions**, it's approximately 0.1415.
- **Max**: The highest value in the feature. For several features like **total_navigations_fav1**, **total_navigations_fav2**, **driven_km_drives**, and others, the maximum value is 0.693147. However, for features like **n_days_after_onboarding** and **device**, the maximum value is 1.

Based on the above statistics for each feature:

- Features like sessions, drives, total_sessions, total_navigations_fav1, total_navigations_fav2, driven_km_drives, duration_minutes_drives, activity_days, and driving_days are likely continuous variables as they have a range of values between 0 and approximately 0.693147.
- The features n_days_after_onboarding and device seem to be binary or categorical, as their maximum values are 1.
- The spread (as indicated by standard deviation) varies among the features. Some, like device, have a higher standard deviation, indicating greater variability in its values, whereas others, like total_navigations_fav1, have a lower standard deviation, suggesting less variability.
- The mean and median (50% percentile) values provide insights into the central tendency of the features. If the mean and median are close, the distribution is likely symmetrical. If they differ significantly, the distribution might be skewed.

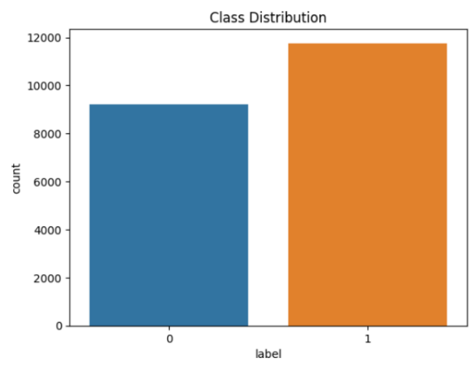


```
         sessions        drives  total_sessions  n_days_after_onboarding  \
count  20990.000000  20990.000000    20990.000000             20990.000000
mean       0.100388      0.104000        0.142098                 0.476174
std        0.093129      0.094493        0.092553                 0.288375
min        0.000000      0.000000        0.000000                 0.000000
25%        0.030529      0.033435        0.072321                 0.227000
50%        0.074179      0.077887        0.124869                 0.461000
75%        0.141500      0.146694        0.191446                 0.722000
max        0.693147      0.693147        0.693147                 1.000000

       total_navigations_fav1  total_navigations_fav2  driven_km_drives  \
count            20990.000000            20990.000000      20990.000000
mean                 0.111108                0.091742          0.169353
std                  0.092997                0.080706          0.094450
min                  0.000000                0.000000          0.000000
25%                  0.047837                0.062975          0.098940
50%                  0.093490                0.066724          0.152721
75%                  0.139762                0.098940          0.222343
max                  0.693147                0.693147          0.693147

       duration_minutes_drives  activity_days  driving_days        device
count             20990.000000   20990.000000  20990.000000  20990.000000
mean                  0.108731       0.434155      0.371844      0.645831
std                   0.077660       0.293811      0.241625      0.478272
min                   0.000000       0.000000      0.000000      0.000000
25%                   0.051643       0.167000      0.172000      0.000000
50%                   0.089841       0.400000      0.379000      1.000000
75%                   0.144966       0.667000      0.552000      1.000000
max                   0.693147       1.000000      1.000000      1.000000
```

2. **Class Distribution Bar Chart:**

The title is Class Distribution

- On the x-axis, we have "label", which has two distinct classes: **0** and **1**.
- The y-axis represents the "count", i.e., the number of instances of each class in the dataset.
- There are two bars in the chart:
- The blue bar represents class **0**. It appears to have a count somewhere just above 10,000.
- The orange bar represents class **1**. It appears to be slightly taller, suggesting its count is a bit higher than that of class **0**.

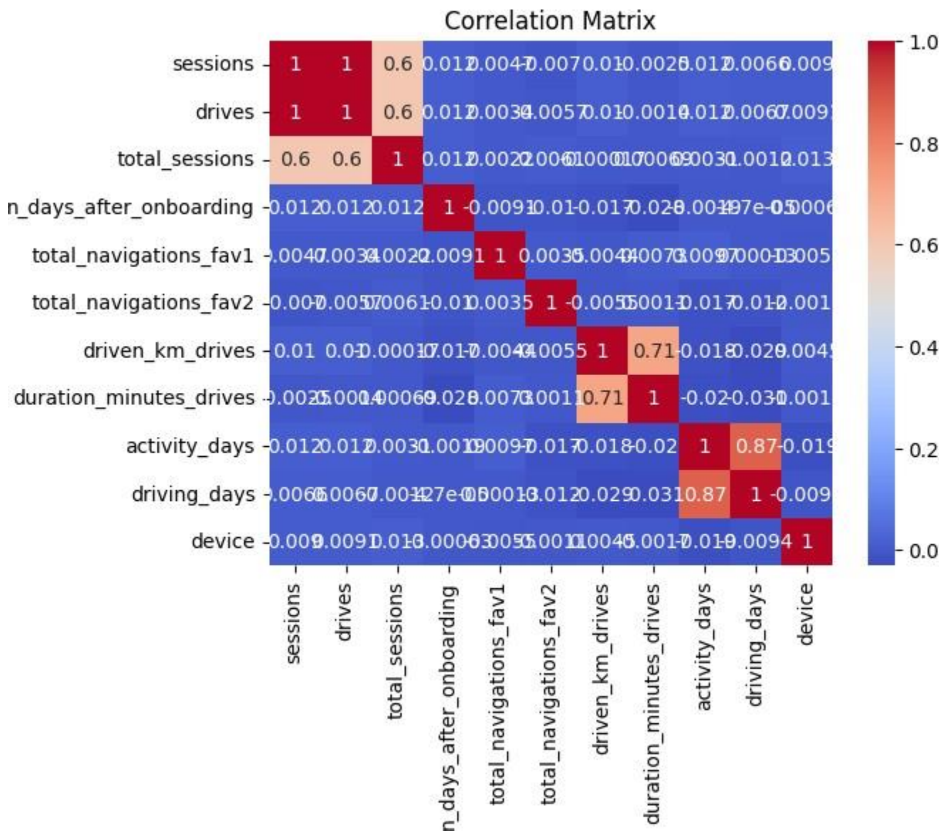


3. **Correlation Matrix:**

A correlation matrix quantifies the degree to which two variables change together. If one variable tends to go up when the other goes up, there is a positive correlation. If one variable tends to decrease when the other goes up, there's a negative correlation. A correlation of 0 indicates no relationship.

The values in a correlation matrix typically range from **-1** to **1**:

- **1** indicates a perfect positive correlation.
- **-1** indicates a perfect negative correlation.
- **0** indicates no correlation.



Analysis of the Correlation Matrix:

- **Diagonal Values**: The diagonal from the top left to the bottom right consists of **1**s. This is expected because a variable is always perfectly positively correlated with itself.
- **Sessions, Drives, and Total_Sessions**:

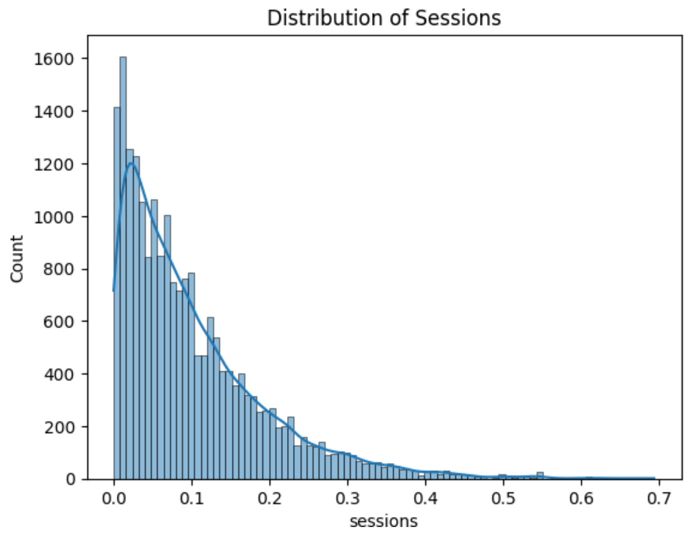
- These three features have strong positive correlations among themselves, with correlation values around **0.6**. This suggests that when the number of sessions goes up, the number of drives and total sessions tend to also increase, and vice versa.
- **Duration_minutes_drives and Driven_km_drives**:
- These two features have a strong positive correlation of **0.71**, indicating that when the duration of drives increases, the driven kilometers also tend to increase.
- **Activity_days and Driving_days**:
- These two features show a very strong positive correlation of **0.87**. Thus, as activity days increase, driving days also tend to increase proportionally.
- **No Strong Negative Correlations**:
- Most of the off-diagonal values are close to zero, indicating weak correlations. There doesn't seem to be any strong negative correlations in the provided matrix.
- **Symmetry**:
- The matrix is symmetric about the diagonal. This means that the correlation between any two variables **A** and **B** is the same as between **B** and **A**.
- **Color Gradient**:
- The matrix uses a color gradient where red represents positive correlations, blue represents negative correlations, and white indicates no correlation. The intensity of the colors corresponds to the strength of the correlation. The matrix's visual representation makes it easier to spot strong correlations at a glance.

Implications:

- Strong correlations between variables might indicate potential multicollinearity if you're planning to run regression analyses. This can affect the stability of your regression coefficients and the interpretability of your model.
- Knowing which variables are correlated can help in feature selection, as highly correlated variables might provide redundant information in predictive modeling.
- It's crucial to remember that correlation doesn't imply causation. Just because two variables are correlated doesn't mean changes in one cause changes in the other.

4. **Histogram:**

A histogram represents the distribution of data by forming bins along the range of the data and then drawing bars to show the number of observations that fall in each bin.



Description of the Histogram:

Title: The histogram is titled "Distribution of Sessions", which provides context to the viewer about what data is being visualized.

X-Axis (sessions): The x-axis represents the values of the sessions feature. The values range from 0.0 to approximately 0.7.

Y-Axis (Count): The y-axis represents the count of observations or the frequency of the sessions values. The counts range from 0 to roughly 1600.

Distribution Shape: The distribution of sessions is right-skewed, which means there's a tail on the right side of the distribution. This implies that most of the data points are concentrated on the left side of the histogram.

The peak of the distribution occurs close to the value 0.0, indicating that this is the mode (most frequent value) of the distribution.

Kernel Density Estimate (KDE): The smooth curve overlaid on the histogram is the KDE, which provides a smoothed version of the histogram. The KDE helps in understanding the probability density of the data at different values.

The KDE curve peaks where the histogram has its highest bar, further highlighting the mode of the distribution. As the values of sessions increase, the KDE curve declines, indicating decreasing density or fewer observations.

Bins: The histogram is divided into vertical bars or bins. The width of the bins seems to be uniform, suggesting equal intervals for sessions values.

Insights:

The majority of the session values lie close to 0.0. As the session value increases, the frequency of observations (or count) decreases.

The distribution being right-skewed suggests that while most of the observations have lower session values, there are a few observations with relatively higher session values.

The KDE provides a clearer idea about the shape of the distribution and confirms the observations made from the histogram bars.

Implications:

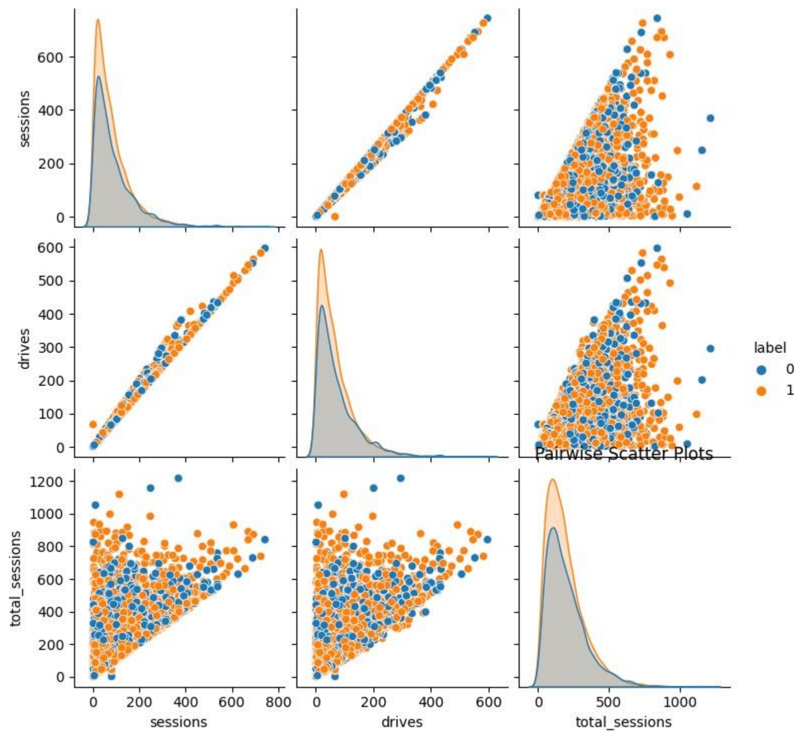
Given the skewness of the distribution, certain statistical analyses or machine learning algorithms might benefit from transformations (e.g., log transformation) to make the distribution more symmetric or "normal".

The insights from this distribution can help inform decisions regarding user engagement, marketing strategies, or any other domain-specific interpretations related to "sessions".

In conclusion, the histogram combined with the KDE provides a comprehensive view of how the sessions values are distributed in the dataset.

5. Pairwise Scatter Plot:

The pairwise scatter plots for three features (sessions, drives, and total_sessions) and their relationships with one another, differentiated by the label feature. Pairwise scatter plots are useful for understanding bivariate relationships between different features in a dataset.



Description of the Pairwise Scatter Plots:

- **Main Diagonal**:
- The diagonal running from the top left to the bottom right displays the distribution of individual features: **sessions**, **drives**, and **total_sessions**.
- These are histogram-like plots where the y-axis indicates the count, and the x-axis indicates the values of the respective feature.
- The distributions are colored by **label** (with blue for **label 0** and orange for **label 1**), allowing for insights into how each class is distributed for each feature.
- **Off-Diagonal Plots**:
- Each off-diagonal plot represents the scatter plot between two features.
- These plots help in understanding how two features relate to each other and if there's any observable pattern or trend between them.
- The points in these scatter plots are colored by **label**, which allows us to understand the distribution of different labels in the bivariate space.

Specific Insights:

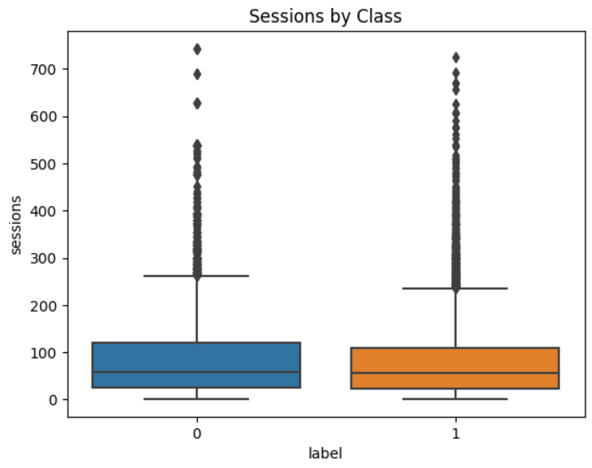
- **sessions vs. drives**:
- The scatter plot between **sessions** and **drives** shows a linear relationship. As the number of sessions increases, the number of drives also tends to increase.
- The data points are closely packed along a diagonal line, indicating a strong positive correlation between these two features.
- **sessions vs. total_sessions**:
- There's a strong positive relationship between **sessions** and **total_sessions**.
- Data points are densely packed, forming a linear trend. As the number of sessions goes up, the **total_sessions** also tends to go up.
- **drives vs. total_sessions**:
- This scatter plot also exhibits a positive correlation between **drives** and **total_sessions**.
- As the number of drives increases, the **total_sessions** also tends to increase.
- **Label Differentiation**:
- In all scatter plots, the differentiation between the two labels (0 and 1) is evident.
- In some regions of the plots, one label dominates over the other, indicating potential patterns or groupings associated with each label.
- **Feature Distributions**:
- The distribution for **sessions** and **drives** looks somewhat similar, with most data points clustered towards the lower end, and both have a right-skewed distribution.
- The distribution for **total_sessions** also appears right-skewed but has a wider spread compared to the other two features.

Conclusion:

The pairwise scatter plots provide a comprehensive view of how different features interact with each other. The strong linear relationships between the features suggest potential multicollinearity, which might need consideration if regression analyses are planned. The coloring by **label** adds another layer of understanding, helping in visualizing how different classes are distributed across various combinations of features.

6. **Box Plot:**

A box plot (also known as a whisker plot) that visualizes the distribution of the `sessions` feature, segmented by two classes, `0` and `1`, represented by the `label`.



Description of the Box Plot:

- X-Axis (`label`): Represents the two classes: `0` (depicted in blue) and `1` (depicted in orange).
- Y-Axis (`sessions`): Represents the values of the `sessions` feature, ranging from approximately `0` to '700'.

Components of the Box Plot:

- Box: The main body of the boxplot showing the interquartile range.
- Bottom Line of the Box: Represents the 1st quartile (Q1), or the 25th percentile.
- Top Line of the Box: Represents the 3rd quartile (Q3), or the 75th percentile.
- Line inside the Box: Represents the median (Q2), or the 50th percentile.
- Whiskers: These are the two lines extending from the box.
- Lower Whisker: Represents the smallest observation greater than or equal to (Q1 - 1.5*IQR), where IQR is the interquartile range (Q3-Q1).
- Upper Whisker: Represents the largest observation less than or equal to (Q3 + 1.5*IQR)
- Points above and below the Whiskers: These are potential outliers, as they fall outside of the whiskers' range.

Insights from the Box Plot:

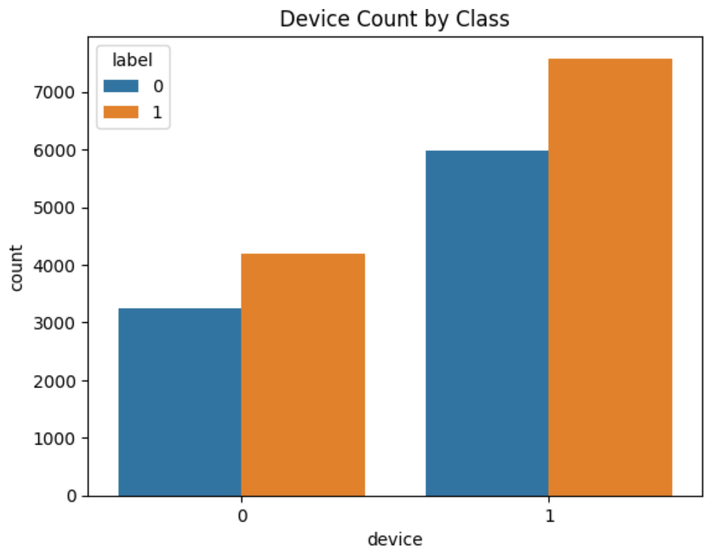
- **Median**: The median value of `sessions` for class `0` is slightly higher than that of class `1`.
- **Spread:** The interquartile range (distance between the 1st and 3rd quartiles) is slightly larger for class `0` than for class `1`, indicating a wider spread of the middle 50% of the data for class `0`.
- **Outliers:**
  - Both classes have several potential outliers, but class `1` seems to have more of them.
  - These outliers are data points that fall outside the typical range (outside the whiskers).
- **Overall Distribution:**
  - For class `0`, most session values are clustered in the lower range, with several higher session values being considered as outliers.
  - Class `1` has a similar distribution but with more outliers in the upper session values.

Conclusion:

The box plot provides a concise visual summary of the distribution of `sessions` for both classes. It allows for easy comparison between the two classes in terms of central tendency, spread, and potential outliers. The presence of outliers, especially in class `1`, suggests that there might be some unique or extreme session values that could be further investigated.

7. Count Plots:

A bar chart that illustrates the count of devices, segmented by two classes (**0** and **1**) represented by the **label**. The visualization is titled **"Device Count by Class"**.



Description of the Bar Chart:

- **Title**: The chart is titled **"Device Count by Class"**, which gives context about the visualization's main theme.
- **X-Axis (device)**: The x-axis has two categories: **0** and **1**, representing two types or categories of devices.
- **Y-Axis (count)**: Represents the number of occurrences, ranging from **0** to around **7000**.
- **Bars**:
- Each device category (**0** and **1**) has two bars:
- A blue bar represents the count for **label 0**.
- An orange bar represents the count for **label 1**.
- The height of each bar corresponds to the number of occurrences in the dataset.

Insights from the Bar Chart:

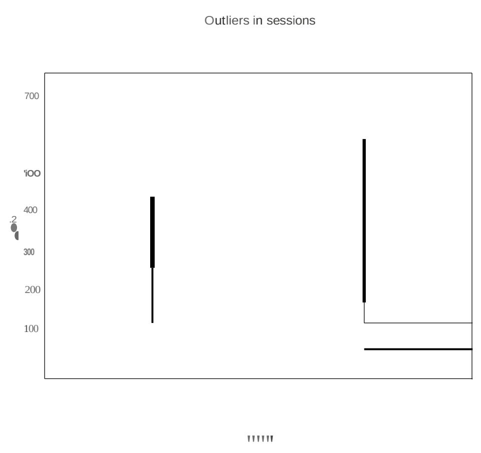
- **Device Category 0**:
- For device category **0**, the count of **label 0** is lower than the count of **label 1**.
- The difference in counts between the two labels is noticeable but not very large.
- **Device Category 1**:
- For device category **1**, the count of **label 1** is higher than the count of **label 0**.
- The difference in counts between the two labels in this category is quite significant, with **label 1** having a much higher representation.
- **Overall Observations**:
- The number of occurrences in the **label 1** category is consistently higher than **label 0** for both device categories.
- Device category **1** has a more prominent difference in the count between the two labels compared to device category **0**.

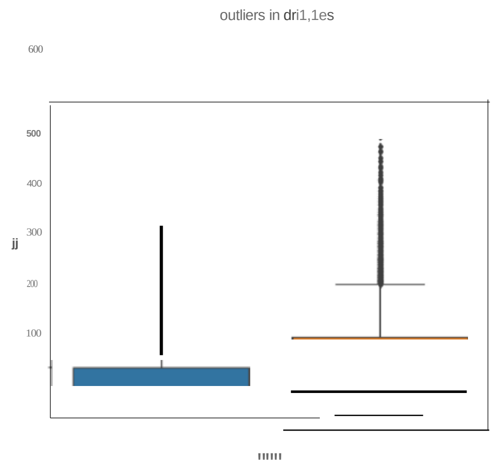
Conclusion:

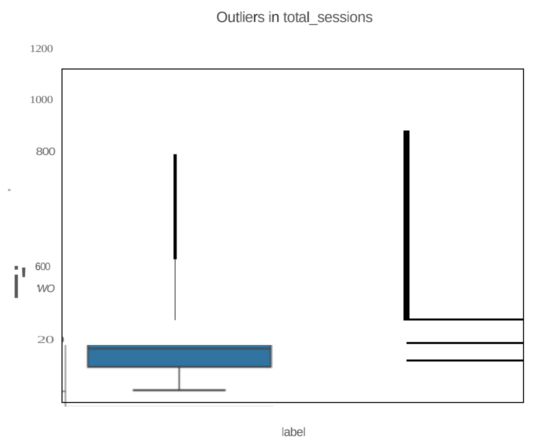
The bar chart provides a clear visual representation of the distribution of devices based on two different class labels. It reveals disparities in counts between the classes for each device category. Understanding such distributions can be crucial for tasks like targeted marketing, user segmentation, or any other domain-specific interpretations related to device types and user classifications.

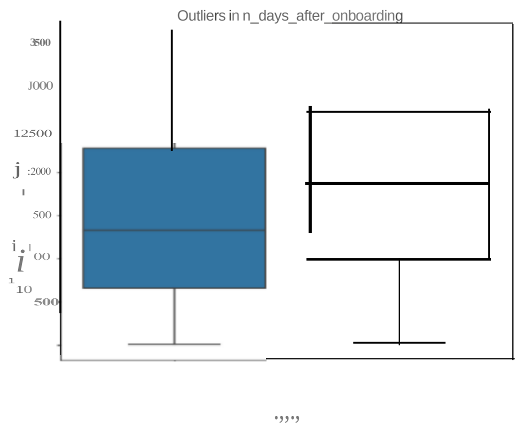
8. Outliers Detection:

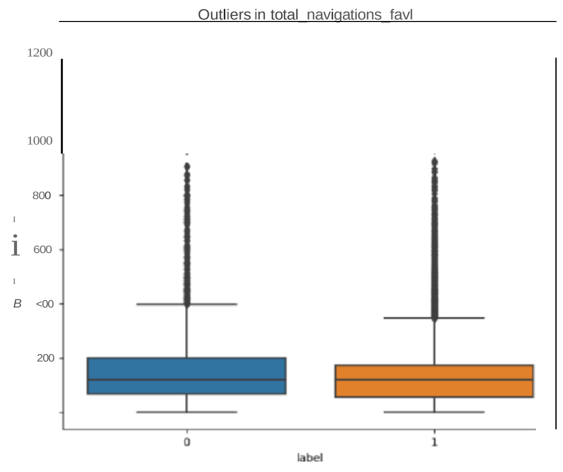
- Box plots are very effective for outlier detection as they visualize the central tendency, spread, and shape of a dataset's distribution. Points outside the whiskers of the box plot are typically considered potential outliers.

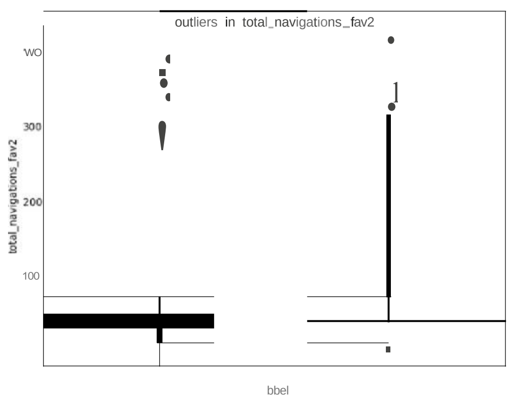

Outliers in sessions
700
100
400
300
200
100

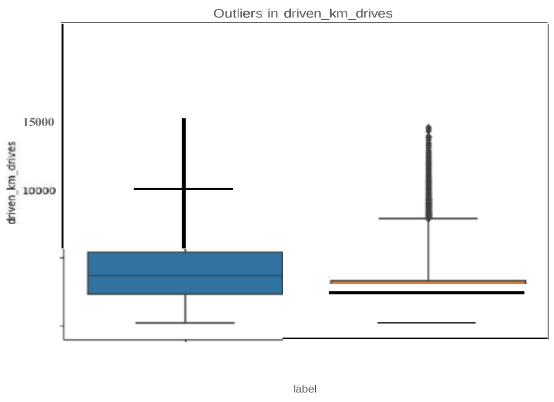

outliers in dri1,1es
600
500
400
300
200
100

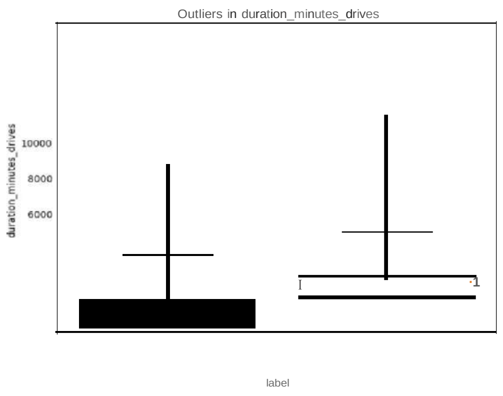

Outliers in total_sessions
1200
1000
800
600
20
label

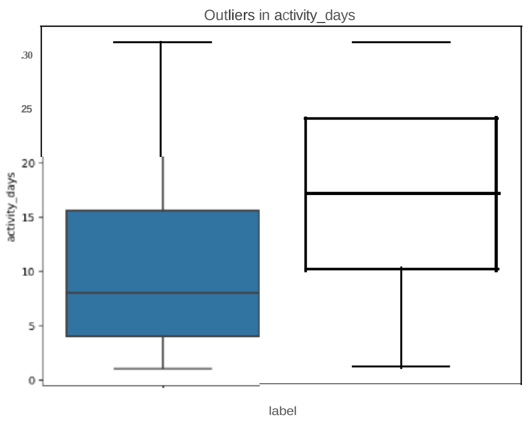

Outliers in n_days_after_onboarding
3500
500

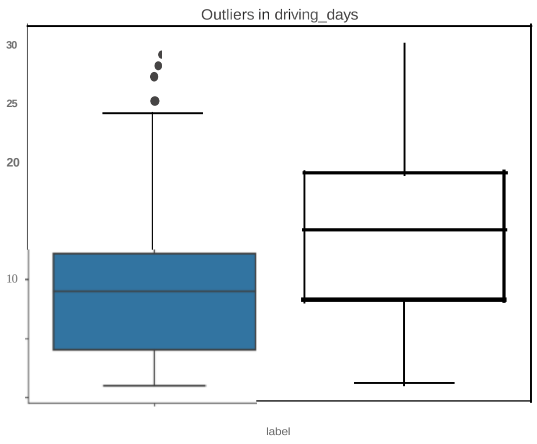

Outliers in total_navigations_favl
1200
1000
800
600
200
0
1
label

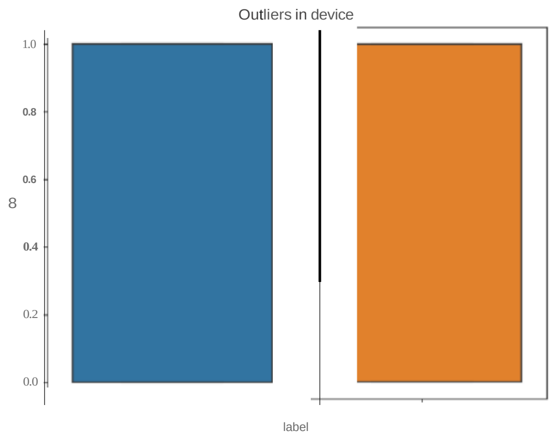

outliers in total_navigations_fav2
300
200
100
total_navigations_fav2
bbel


Outliers in driven_km_drives
15000
10000
driven_km_drives
label


Outliers in duration_minutes_drives
10000
8000
6000
duration_minutes_drives
label


Outliers in activity_days
30
25
20
15
10
5
0
activity_days
label

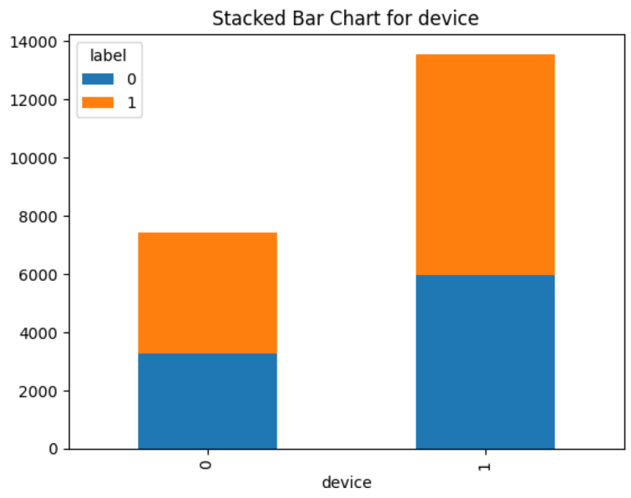

Outliers in driving_days
30
25
20
10
label


Outliers in device
1.0
0.8
0.6
0.4
0.2
0.0
label

9. **Stacked Bar Graph:**

Description of the Stacked Bar Chart:

- **Title**:
- The chart is titled **"Stacked Bar Chart for device"**.
- **X-Axis (device)**:
- The x-axis consists of two categories: **0** and **1**, signifying two types or categories of devices.
- **Y-Axis (count)**:
- Represents the number of occurrences or counts, with the scale ranging from **0** to approximately **14000**.
- **Bars**:
- Each category on the x-axis has a single bar that is divided into two segments:
- The bottom segment (in blue) represents the count for **label 0**.
- The top segment (in orange) represents the count for **label 1**.
- The height of each segment indicates the number of occurrences for each **label**.



Insights from the Stacked Bar Chart:

- **Device Category 0**:
- For device category **0**, the count for **label 0** is visibly lower than the count for **label 1**.
- **Device Category 1**:
- For device category **1**, the count for both **label 0** and **label 1** is much higher compared to device category **0**.
- Within this category, the count for **label 0** appears to be slightly higher than the count for **label 1**.

Overall Distribution:

- The chart provides an understanding of the distribution of devices for each class. It allows users to quickly gauge the proportional difference between the two labels for each device category.

Conclusion:

The stacked bar chart is an effective way to visualize the distribution of a categorical variable across different classes. It provides clarity on the composition of each category based on the classes in the dataset. In this specific chart, one can easily deduce how devices are distributed across two labels, offering valuable insights into the data's structure.

10. Grouped Histograms (Stacked):

Description of the Stacked Histogram:

Title: The graph is titled "Stacked Histograms for sessions (Balanced Dataset)".

X-Axis (sessions): Represents the range of values for the sessions feature, spanning from approximately 0 to just under 700.

Y-Axis (count): Denotes the number of occurrences, with the scale ranging from 0 to around 1000.

Histogram Bars:

There are two distinct colors for the bars:

Blue, which represents the distribution of Label 0.

Orange, which represents the distribution of Label 1.

The bars' height indicates the number of occurrences for specific value ranges of the sessions feature.

Kernel Density Estimation (KDE) Lines: Superimposed on the histogram bars are two lines that represent the Kernel Density Estimation (KDE) for both labels, aiding in understanding the distribution's shape and density.

Insights from the Histogram:

Most of the data for both labels concentrates around the lower end of the sessions value range.

There's a peak in the frequency (highest bar) at the very beginning of the distribution for both labels.

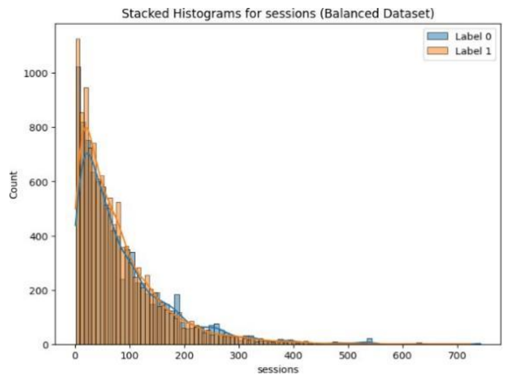
The distribution tails off gradually, with fewer occurrences as the sessions value increases.

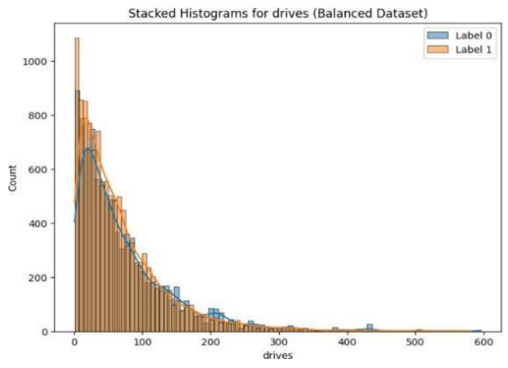
Both Label 0 and Label 1 follow a similar distribution pattern, though the actual counts differ.

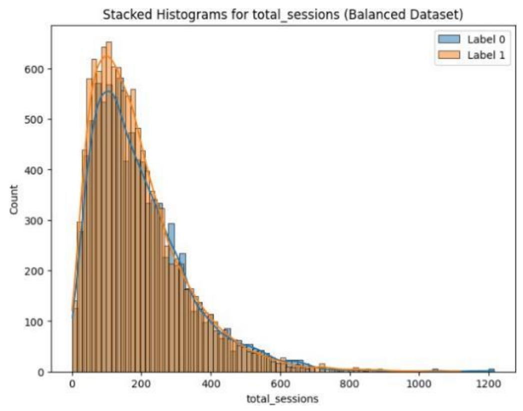
Conclusion:

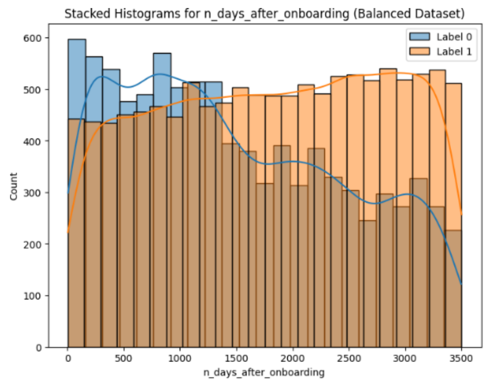
The stacked histogram provides a combined view of the distribution of the sessions feature for both labels. It aids in understanding how data is distributed across different value ranges and offers insights into similarities and differences in the distributions between the two labels.

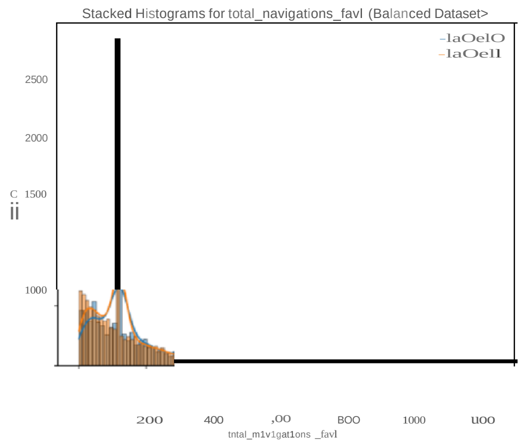
Given the code snippet, similar visualizations would be generated for all features, providing a comprehensive view of the dataset's structure.

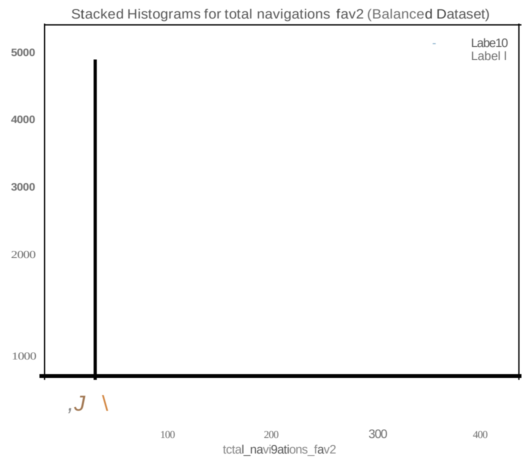


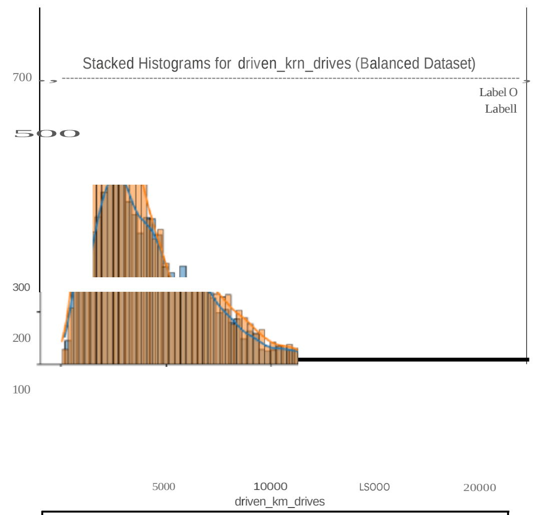


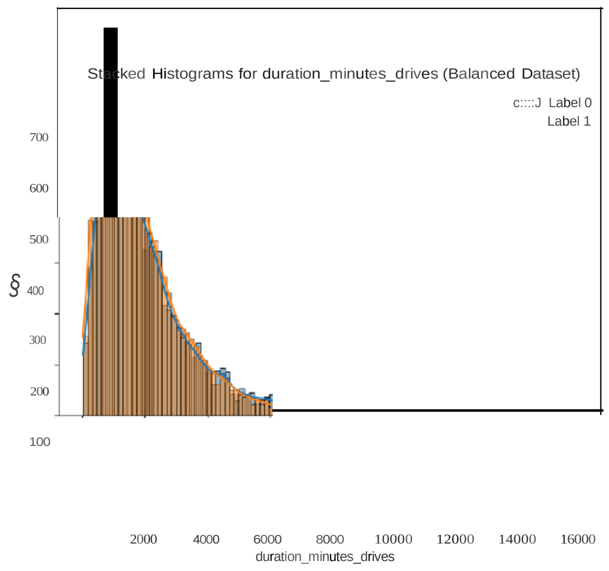


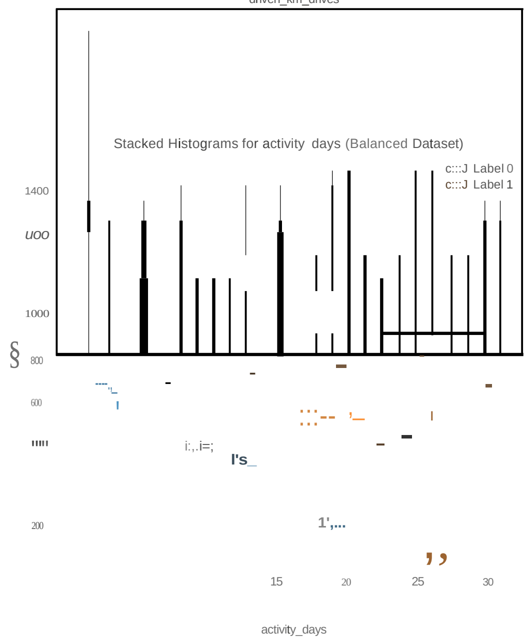


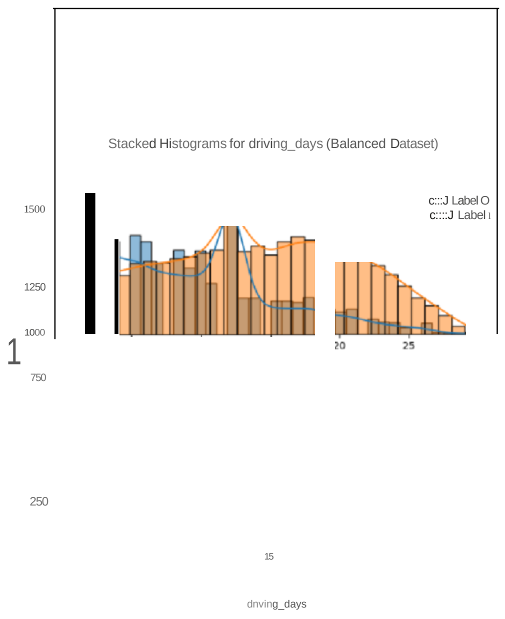

Stacked Histograms for total_navigations_favl (Balanced Dataset>
-laOelO
-laOell
2500
2000
C 1500
ii
1000
200
400
,oo
BOO
1000
uoo
tntal_m1v1gat1ons _favl

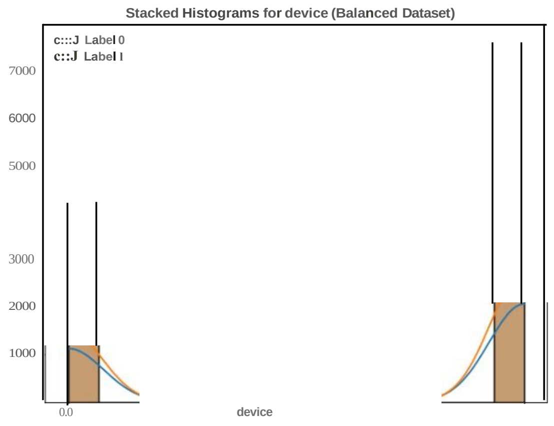

Stacked Histograms for total navigations fav2 (Balanced Dataset)
Labe10
Label l
5000
4000
3000
2000
1000
100
200
300
400
tctal_navi9ations_fav2


Stacked Histograms for driven_krn_drives (Balanced Dataset)
Label O
Labell
700
500
300
200
100
5000
10000
LS000
20000
driven_km_drives


Stacked Histograms for duration_minutes_drives (Balanced Dataset)
c::::J Label 0
Label 1
700
600
500
§ 400
300
200
100
2000
4000
6000
8000
10000
12000
14000
16000
duration_minutes_drives


Stacked Histograms for activity days (Balanced Dataset)
c:::J Label 0
c:::J Label 1
1400
UOO
1000
§ 800
600
200
15
20
25
30
activity_days


Stacked Histograms for driving_days (Balanced Dataset)
c:::J Label O
c:::J Label ı
1500
1250
1000
750
250
20
25
15
dnving_days



Phase 2 -ML Models

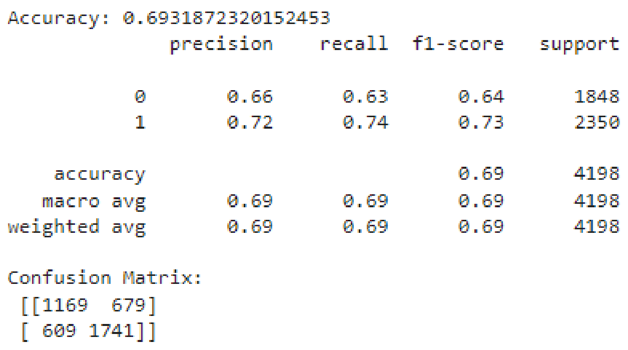
1. Logistic Regression:
In the context of predicting customer retention or churn for Waze app users, choosing logistic regression as a modeling technique can be justified by considering several aspects of the problem and the data:
• **Binary Outcome:** Logistic regression is specifically designed for binary classification tasks. Since your target variable has two states (retained or churned), logistic regression is a natural choice.
• **Interpretability:** Logistic regression models are highly interpretable, meaning it's relatively easy to understand the relationship between the input variables and the outcome. This can be particularly useful for business stakeholders who need insights to make decisions about how to improve customer retention.
• **Efficiency:** Logistic regression models are generally fast to train and can handle moderate-sized datasets well, which is suitable for agile environments or when computational resources are limited.
• **Baseline Model**: Logistic regression is often used as a baseline model. Its performance can be a benchmark for more complex models. If a simple model like logistic regression already provides sufficient predictive power, there may be no need for more complex algorithms.

Model Tuning and Training:
To train the logistic regression model, several steps would have been taken:
• Feature Selection: We might have had to select the most relevant features that contribute to customer churn. This might have involved domain knowledge, statistical tests for correlation, or feature importance techniques.
• **Data Preprocessing**: Appropriate preprocessing steps such as encoding categorical variables, handling missing values, normalizing or standardizing numerical variables, and potentially dealing with imbalanced classes would be necessary.
• **Model Selection**: We chose logistic regression parameters like the regularization strength and the type of solver. These can be determined through techniques like cross-validation and grid search.
• **Validation:** We likely implemented a validation strategy, such as k-fold cross-validation, to ensure that the model generalizes well to unseen data.

Effectiveness of the Algorithm:
• The effectiveness of the logistic regression algorithm in this case can be discussed in terms of the following metrics:
• **Accuracy:** An accuracy of approximately 69.32% is a good starting point but may not be enough depending on the business context and the baseline churn rate.
• **Precision and Recall:** These metrics are crucial in the context of churn prediction. High precision for the churned class means that when the model predicts churn, it is likely correct. High recall means that the model is good at identifying all actual churn cases.

• **F1-Score**: The F1-score for both classes gives an idea of the balance between precision and recall. In situations where the cost of false positives and false negatives is high, it's important to maximize this score.
• **Confusion Matrix:** The confusion matrix gives a detailed breakdown of the model's performance, showing how many of each class were correctly or incorrectly classified.

From the application of logistic regression to our data, we may have gained insights such as:
• **Potential for Improvement:** The model's performance can be improved by more sophisticated algorithms, additional data, or further feature engineering.
• Overall, while logistic regression might not be the most powerful algorithm available, its simplicity, efficiency, and interpretability make it a strong choice for the initial modeling of customer churn. Subsequently, we have explored more complex models like Random Forests or Gradient Boosting Machines if the predictive performance needs to be increased and we are willing to potentially sacrifice some interpretability.

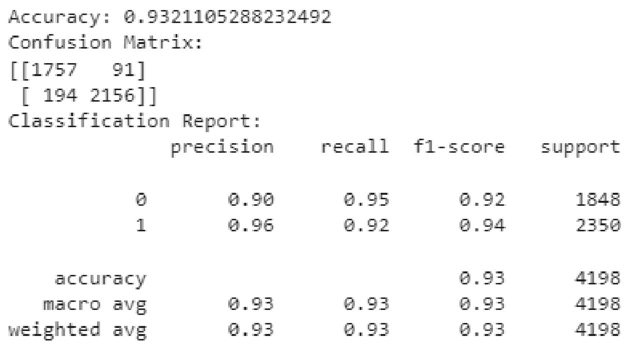


```
Accuracy: 0.6931872320152453
              precision    recall  f1-score   support

           0       0.66      0.63      0.64      1848
           1       0.72      0.74      0.73      2350

    accuracy                           0.69      4198
   macro avg       0.69      0.69      0.69      4198
weighted avg       0.69      0.69      0.69      4198

Confusion Matrix:
 [[1169  679]
 [ 609 1741]]
```

2. Random Forest:
Choosing the Random Forest algorithm for predicting customer retention or churn for Waze app users can be substantiated by several compelling reasons:
• **Handling Complexity:** Random Forest can handle a large number of features and complex, non-linear relationships between them. This is particularly useful when predicting behavior such as churn, which can be influenced by complex interactions between variables.
• **Reduced Overfitting:** The ensemble approach of Random Forest, which builds multiple decision trees and aggregates their predictions, naturally guards against overfitting compared to a single decision tree. This is important in a predictive model to ensure that it generalizes well to new, unseen data.
• **Tuning and Training the Model:** For tuning and training the Random Forest model, the following steps were likely taken:
• **Choosing the Number of Trees**: We set n_estimators=100, which means the Random Forest uses 100 trees. This is a balance between computational efficiency and model performance.
• **Handling Overfitting:** Although Random Forest is less prone to overfitting, we would still ensure that the model parameters (like maximum depth of the trees, minimum samples per leaf, etc.) are set to prevent it.
• **Cross-Validation:** Using techniques like k-fold cross-validation to tune the hyperparameters and validate the model's performance helps in ensuring the model is robust and generalizes well.
• **Feature Engineering:** Prior to fitting the model, features might have been engineered or selected based on domain knowledge, correlation with the target variable, or importance in preliminary models.

Effectiveness of the Algorithm:
• The effectiveness of your Random Forest model can be inferred from the following metrics:
• **Accuracy:** An accuracy of 93.21% is impressive and suggests the model is well-tuned for this particular task.
• **Precision and Recall**: High precision and recall across both classes indicate that the model is balanced and effective at identifying both retained and churned users accurately.

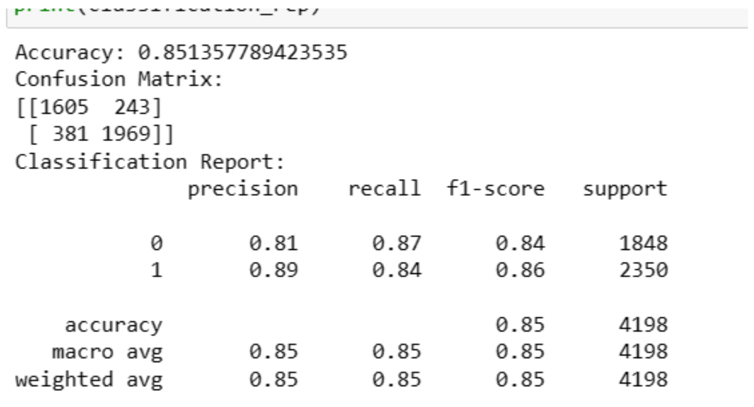
• **F1-Score**: The high F1-scores for both classes further confirm that the model has a good balance between precision and recall.
• **Confusion Matrix:** The confusion matrix shows a low number of false positives and false negatives, which is ideal in a churn prediction scenario where both types of errors can be costly.
• **Gained Intelligence:** From the Random Forest application to our data, several insights can be gained:
• **Areas for Improvement**: Any misclassifications can be analyzed to understand where the model might be falling short. This indicate areas where additional data or feature engineering could be beneficial.
• Business Strategy: The model's insights can inform strategic decisions, such as personalized retention offers or identifying the right time to engage with the customer to prevent churn. By demonstrating high predictive accuracy and providing actionable insights through feature importances, the Random Forest model has proven to be effective for the churn prediction problem in this case.


```
Accuracy: 0.9321105288232492
Confusion Matrix:
[[1757   91]
 [ 194 2156]]
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.95      0.92      1848
           1       0.96      0.92      0.94      2350

    accuracy                           0.93      4198
   macro avg       0.93      0.93      0.93      4198
weighted avg       0.93      0.93      0.93      4198
```

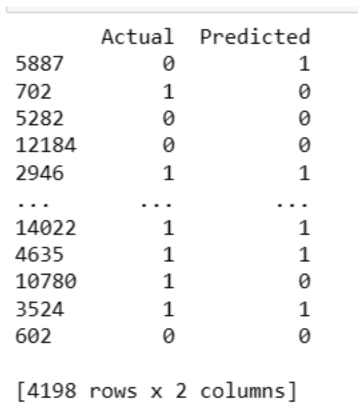
3. Extreme Gradient Boosting (XGBoost):
Using the XGBoost algorithm for the churn prediction problem for Waze app users is well-justified due to its performance and efficiency in handling various types of data. Here's the rationale and some considerations:
Justification for Choosing XGBoost:
•Performance: XGBoost is renowned for delivering high-performance models and is often a winning algorithm in machine learning competitions.
•Gradient Boosting Framework: It is an optimized gradient boosting library which can give better results than Random Forest if tuned properly.
•Regularization: XGBoost includes built-in regularization which helps to prevent overfitting, an advantage over other boosting algorithms.
•Handling Sparse Data: XGBoost can handle sparse data (like user interaction data which can be full of missing values) effectively.

Tuning and Training the Model:
•**Hyperparameter Tuning:** Parameters like n_estimators, learning rate, max depth of trees, min child weight, and gamma were likely fine-tuned to improve the model. Grid search or random search could have been used for this.
•**Cross-validation**: XGBoost has an in-built CV function that helps in assessing the robustness of the model.
•Handling Imbalanced Data: If the churned vs retained users data was imbalanced, the scale_pos_weight parameter would have been useful.
•**Early Stopping**: To prevent overfitting and optimize computation time, early stopping could have been used to stop training if the validation metric doesn't improve for a given number of rounds.

Effectiveness of the Algorithm:
The model's effectiveness is reflected in the following:
•**Accuracy:** An accuracy of approximately 85.14% is indicative of a strong model, though it is slightly lower than that achieved by the Random Forest classifier.
•**Precision and Recall:** The model shows a good balance between precision and recall for both classes indicating a robust performance.
•**F1-Score**: The F1-scores near to or above 0.85 signify a harmonic balance between precision and recall.
•**Confusion Matrix**: The matrix indicates a reasonably low number of false positives and false negatives, which is crucial for a churn prediction model.
Gained Intelligence: The insights gained from the XGBoost model are valuable for making informed business decisions:
•**Feature Importance:** The visualized feature importances can guide resource allocation by highlighting what factors contribute most to user retention or churn.
•**Model Interpretability**: While not as interpretable as simpler models, XGBoost still offers some level of interpretability, which can be enhanced with tools like SHAP values.
•**Predictive Power:** The model's strong predictive power can be harnessed for creating personalized customer experiences to prevent churn.
•Strategic Initiatives: By understanding the key drivers of churn, targeted initiatives can be designed to improve user engagement and satisfaction.
In summary, XGBoost's performance in this application demonstrates its capability to handle the complex task of churn prediction effectively, providing both high accuracy and actionable insights.



```
Accuracy: 0.851357789423535
Confusion Matrix:
[[1605  243]
 [ 381 1969]]
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.87      0.84      1848
           1       0.89      0.84      0.86      2350

    accuracy                           0.85      4198
   macro avg       0.85      0.85      0.85      4198
weighted avg       0.85      0.85      0.85      4198
```



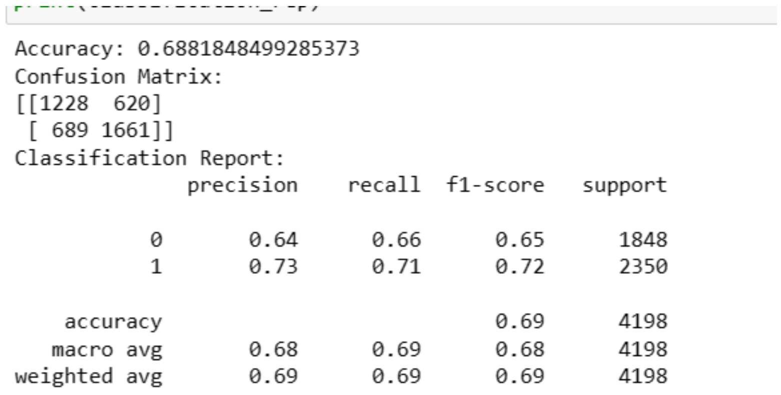
```
       Actual  Predicted
5887        0          1
702         1          0
5282        0          0
12184       0          0
2946        1          1
...       ...        ...
14022       1          1
4635        1          1
10780       1          0
3524        1          1
602         0          0

[4198 rows x 2 columns]
```

4. Support Vector Machine
The choice to apply a Support Vector Machine (SVM) algorithm for predicting customer churn for Waze app users can be justified by the nature of SVMs and their performance on certain datasets.
Justification for Choosing SVM:
1. **Efficacy on Small to Medium Datasets:** SVMs are effective on datasets of moderate size, which might be the case with the Waze users data.
2. **Feature-Rich Data:** If the data has a lot of features, SVMs can perform quite well as they can handle the high dimensionality.
3**. Linear Separability:** By choosing a linear kernel, it is presumed that the data can be linearly separated which is often the case in many binary classification problems.

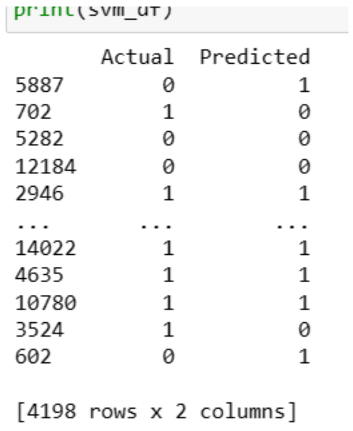
Tuning and Training the Model:
•**Regularization Parameter (C):** A C value of 1.0 indicates a balance between maximizing the margin and minimizing the classification error. The model performance has benefitted from fine-tuning this parameter.
•**Scaling Features:** SVMs are sensitive to the scale of the data, hence it's important to scale the features before training the model.
•**Cross-validation:** To ensure the model generalizes well, cross-validation has been used to optimize the hyperparameters.

Effectiveness of the Algorithm:
•**Accuracy**: The model achieves an accuracy of approximately 68.82%, which is decent but lower compared to ensemble methods like Random Forest or XGBoost.
•**Precision and Recall:** The SVM has comparable precision and recall for both classes, indicating a balanced ability to classify churned and retained users.
•**F1-Score:** The F1-scores reflect a reasonable trade-off between precision and recall, which is important for a balanced model.
•**Confusion Matrix:** There's a fair number of false positives and negatives, which could potentially be reduced with further model optimization.
Gained Intelligence:
•**Decision Boundary:** SVMs are great at finding the optimal decision boundary which can be very insightful, especially if the feature space is visualizable.
•**Support Vectors:** The model’s support vectors can give insights into the most informative data points in the prediction of churn.
•**Generalizability:** With proper regularization, SVMs can be less prone to overfitting compared to some other algorithms, leading to a model that generalizes better to unseen data.

Conclusion:
While the SVM shows a good start, its performance is less effective than the previously discussed ensemble methods. This could be due to the choice of kernel, the parameter settings, or the nature of the data. For a problem like churn prediction, where the cost of false negatives (failing to identify a customer who will churn) can be high, further tuning might be required to improve the recall for the churn class without significantly sacrificing precision. Additionally, exploring non-linear kernels or other feature transformation methods could potentially improve the model's performance.



```
Accuracy: 0.6881848499285373
Confusion Matrix:
[[1228  620]
 [ 689 1661]]
Classification Report:
              precision    recall  f1-score   support

           0       0.64      0.66      0.65      1848
           1       0.73      0.71      0.72      2350

    accuracy                           0.69      4198
   macro avg       0.68      0.69      0.68      4198
weighted avg       0.69      0.69      0.69      4198
```



```
print(svm_df)
```

```
       Actual  Predicted
5887        0          1
702         1          0
5282        0          0
12184       0          0
2946        1          1
...       ...        ...
14022       1          1
4635        1          1
10780       1          1
3524        1          0
602         0          1

[4198 rows x 2 columns]
```

5.KNN

The application of the K-Nearest Neighbors (KNN) algorithm to predict customer retention or churn for Waze app users can be evaluated based on several considerations:

Justification for Choosing KNN:

1.**Non-Parametric Nature**: KNN is a non-parametric method, meaning it makes no underlying assumptions about the distribution of data. This can be particularly beneficial if the data distribution is unknown or non-Gaussian.

2.**Simplicity and Baseline**: KNN is easy to understand and implement, which makes it a good baseline algorithm to compare with more complex models.

3.**Local Decision Making**: It makes predictions based on the labels of the nearest data points, thus capturing local data patterns that could be meaningful for user behaviors in applications like Waze.

Tuning and Training the Model:

•**Number of Neighbors (k)**: The choice of k=5 is a starting point. To optimize performance, cross-validation could be used to find the best value of k.

•**Feature Scaling**: KNN requires feature scaling because it relies on the distance between data points to determine their similarity.

•**Weighting:** Applying different weights to the contributions of neighbors (such as distance weighting) could improve performance by reducing the impact of potentially noisy neighbors.

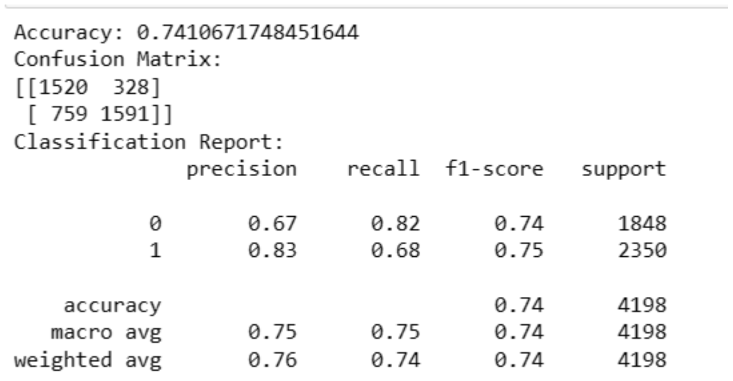
Effectiveness of the Algorithm:

•**Accuracy**: At approximately 74.11%, KNN performs better than the SVM, which suggests that the local similarity patterns are more relevant for this dataset.

•**Precision and Recall:** There is a clear trade-off, with higher precision for the positive class (retained customers) but better recall for the negative class (churned customers).

•**F1-Score:** The F1-score is a balanced metric that takes into account both precision and recall. KNN presents similar F1-scores for both classes, indicating a balanced performance.
•**Confusion Matrix:** The number of false negatives and false positives suggests that the model may be better at identifying retained users than churned users.
Gained Intelligence:
•**Local Patterns:** The KNN model may reveal local patterns that are not captured by global models, which can provide insights into user behavior.
•**Feature Relationships:** The importance of different features can be indirectly assessed by looking at the neighbors that contribute to a decision, potentially revealing feature relationships that were not previously considered.

Conclusion:
KNN has demonstrated a reasonable performance for the churn prediction problem. However, there is room for improvement, especially in terms of optimizing the number of neighbors and considering different weighting strategies. It also points out that locality and the similarity of users' behavior are significant, which can be insightful for understanding and addressing churn. Given that KNN can suffer from high computational costs with large datasets and is sensitive to irrelevant or redundant features, feature selection and dimensionality reduction techniques might also enhance its performance.


```
Accuracy: 0.7410671748451644
Confusion Matrix:
[[1520  328]
 [ 759 1591]]
Classification Report:
              precision    recall  f1-score   support

           0       0.67      0.82      0.74      1848
           1       0.83      0.68      0.75      2350

    accuracy                           0.74      4198
   macro avg       0.75      0.75      0.74      4198
weighted avg       0.76      0.74      0.74      4198
```

6. Naive Bayes
Justification for Choosing Naive Bayes:
1.Probabilistic Foundation: Naive Bayes classifiers works well for classification problems where the assumption is that the features are independent given the class label. This probabilistic foundation allows for direct estimation of the probability of churn given a set of features.
2.Performance on Large Feature Spaces: Given that Naive Bayes scales well with the number of features, it is a good candidate for datasets with a large number of predictors.
3.Fast and Efficient: Naive Bayes is known for being fast and efficient, especially in comparison to more complex models like SVM or KNN. This makes it particularly useful for baselines and when computational resources are limited.
4.Good with Categorical Data: While GaussianNB assumes features are continuous and normally distributed, Naive Bayes in general is effective when dealing with categorical data, which can be the case with user demographics or app usage categories.

Tuning and Training the Model:

•**Data Preprocessing:** As Naive Bayes assumes independence between features, removing correlated features helps the dataset contains redundant information.

Effectiveness of the Algorithm:

•**Accuracy:** At approximately 68.10%, Naive Bayes performs similarly to SVM but is outperformed by KNN. This may indicate that the independence assumption does not fully hold in this data, or that the data distribution is not entirely Gaussian.

•**Precision and Recall**: Naive Bayes shows reasonable precision and recall, though not the highest compared to other models like KNN. This could be improved with better feature selection or transformation.

•**F1-Score**: The F1-scores suggest a fair balance between precision and recall for both classes.

•**Confusion Matrix**: The number of false negatives is higher than false positives, which might indicate a tendency of the model to be more conservative in predicting churn.

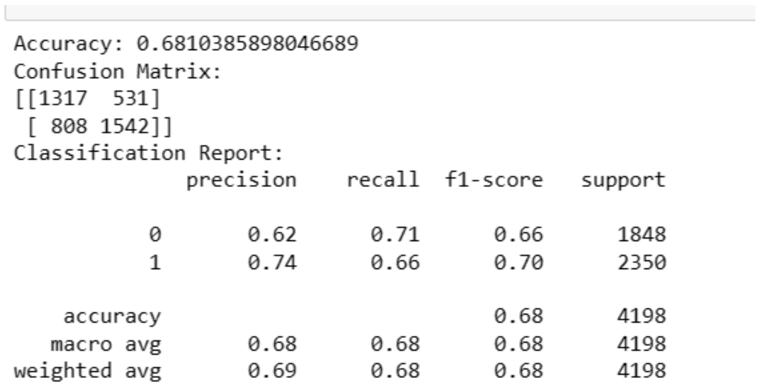
Gained Intelligence:

•**Feature Importance**: By looking at the log-probabilities in Naive Bayes, one can gain insight into the importance of different features for predicting churn.

•**Probability Thresholds**: Naive Bayes allows adjusting the threshold for classification, which can be tuned to optimize for specific business costs associated with false positives and false negatives.

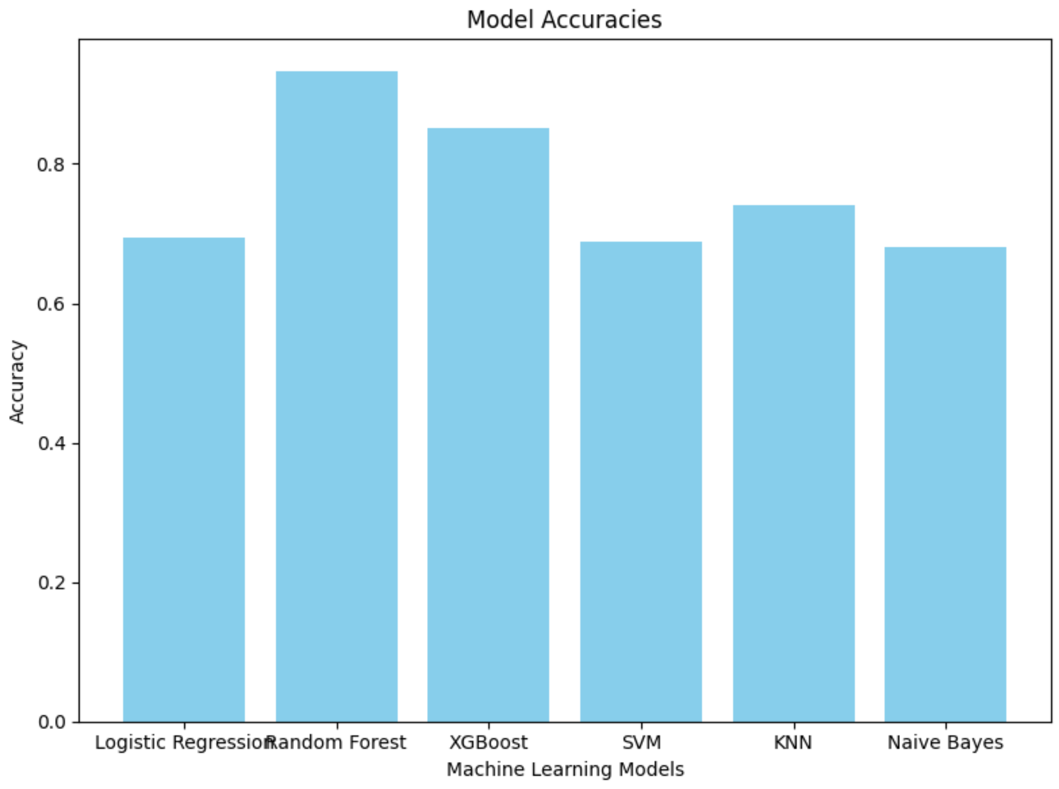
Conclusion:

Naive Bayes offers a quick and straightforward approach to modeling the churn prediction problem. Its performance is competitive but does not necessarily provide the best results. The main strengths lies in its simplicity and efficiency. Given its probabilistic nature, Naive Bayes can also be particularly useful in scenarios where probabilities are more informative than hard classifications, such as in risk assessment or decision-making processes that require understanding the uncertainty of predictions. However, in its Gaussian form, it may not be the best choice if the data doesn't align with its underlying assumptions, and other variants of Naive Bayes or different models might yield better results.



```
Accuracy: 0.6810385898046689
Confusion Matrix:
[[1317  531]
 [ 808 1542]]
Classification Report:
              precision    recall  f1-score   support

           0       0.62      0.71      0.66      1848
           1       0.74      0.66      0.70      2350

    accuracy                           0.68      4198
   macro avg       0.68      0.68      0.68      4198
weighted avg       0.69      0.68      0.68      4198
```

Graph for Comparison of Accuracies of all the models:



As seen in the graph, the accuracy of Random Forest is more than 95% so it has performed best among all the models.

PHASE 3: Building a Data Product

This phase involves building a data product to automate and parameterize the processes established in Phases 1 and 2. The data product is implemented as a web application using Flask, allowing users to interact with the predictive model.

1.a. The application we developed consists of two routes: the home route ("/") and the predict route ("/predict"). The home route renders an HTML template ('index.html') when the user accesses the root URL. The predict route is triggered when the user submits a form (HTTP POST request) with input data, and it uses a pre-trained machine learning model (loaded from "model.pkl") to predict whether a user is "Retained" or "Churned" based on the provided features. The predicted result is then displayed on a result page ('result.html'). The code includes error handling to manage exceptions during the prediction process. When executing the Flask application script, a link to launch Web Application is generated.

This includes essential files such as

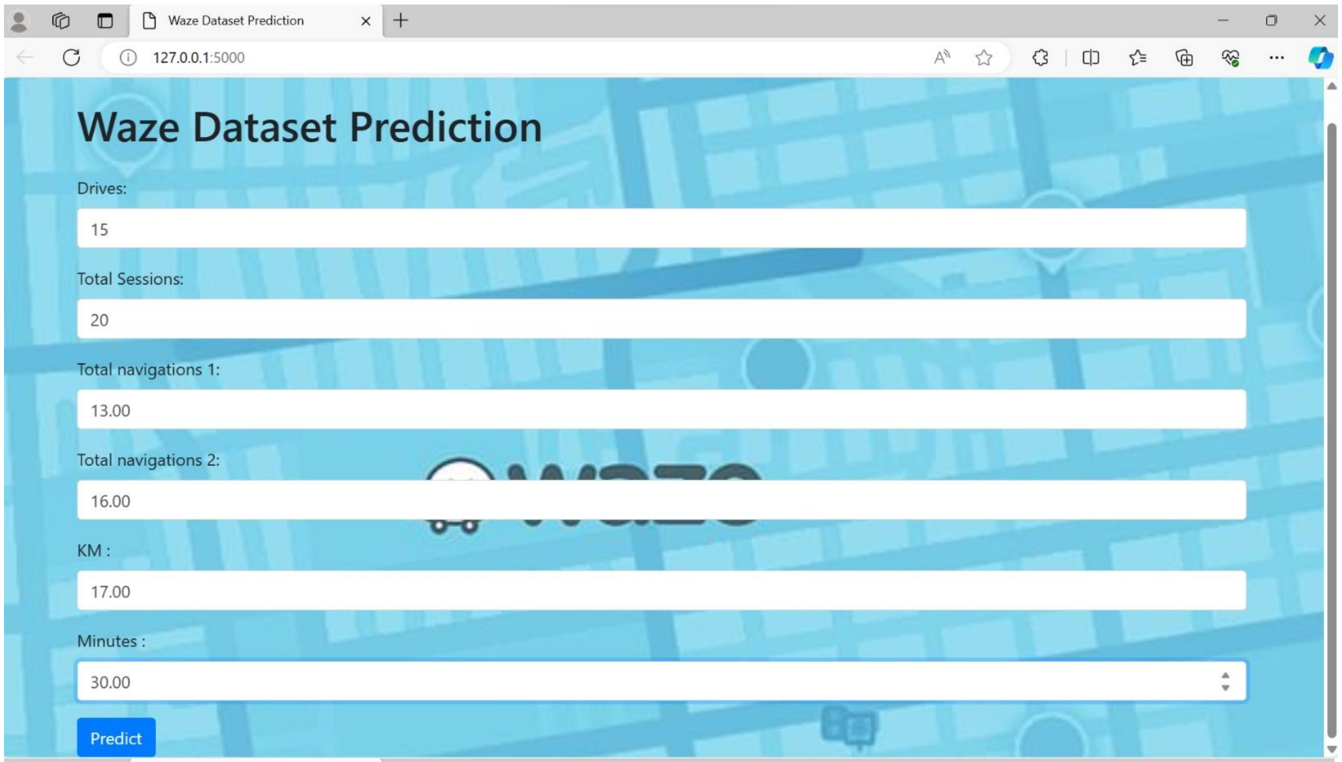
app.py : Flask Application Script
templates: HTML Templates for Web Pages
static: Static Files (e.g., Images)
Model.pkl: Trained Machine Learning model with best accuracy (RandomForest Regression model here)

Below are the steps on using Web Application after executing app.py :
1. Open the web application in a browser.
2. Fill in the input fields with relevant data.
3. Click the "Predict" button to see the churn prediction.
4. The result page will display whether the user is predicted as "Retained" or "Churned."

1.b
In Phase 2, we used 6 different models. The best performing model for our application is Random Forest since it has the highest accuracy and also because we had 11 features which had non-linear relationships between them. Random forest is particularly useful when predicting behavior such as churn, which can be influenced by complex interactions between variables. The model was instantiated with a set of parameters, including n_estimators=100 and random_state=42. The choice of these parameters was informed by considerations of achieving a balance between model complexity and generalizability. The n_estimators parameter represents the number of trees in the forest, and its value was set to 100 to ensure a sufficiently robust ensemble. The random_state parameter was fixed at 42 to maintain reproducibility.

During the training phase, the Random Forest Classifier was fitted to the training data, and subsequent predictions were made on the test data. The model's performance was assessed using key metrics, including accuracy, precision, recall, and F1-score. The obtained results indicate a high level of accuracy (93.21%) on the test set, demonstrating the effectiveness of the model in making accurate predictions.

These specific choices in model selection and tuning were guided by a thorough evaluation of the trade-offs between various hyperparameters and their impact on the model's predictive capabilities. The emphasis was on achieving a well-performing model with a focus on both precision and recall. The decision to save the trained model to a file, named 'model.pkl,' ensures seamless integration into the final product, providing users with a robust and reliable predictive tool based on the tuned Random Forest Classifier from Phase 2.
Below are the snippets of our Web Application with Churned and Retained predictions:-



Waze Dataset Prediction

127.0.0.1:5000

Waze Dataset Prediction

Drives: 15

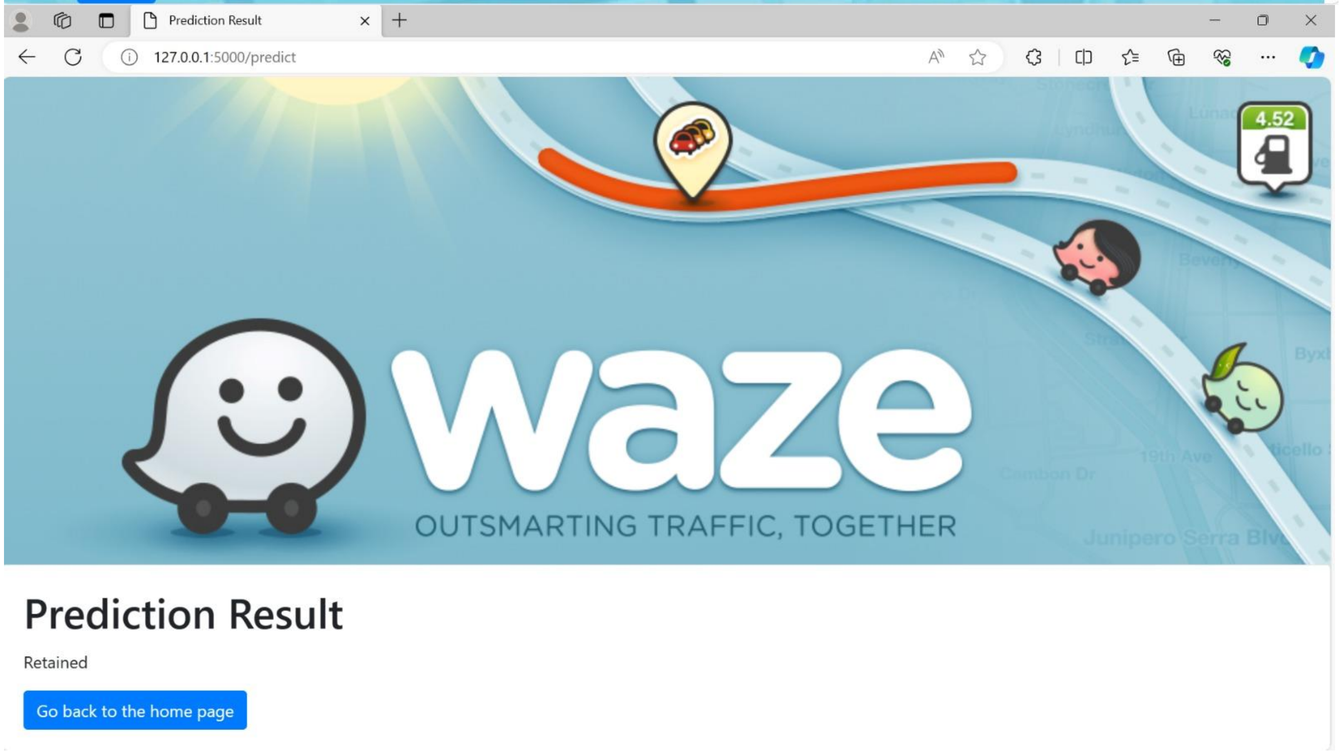
Total Sessions: 20

Total navigations 1: 13.00

Total navigations 2: 16.00

KM : 17.00

Minutes : 30.00

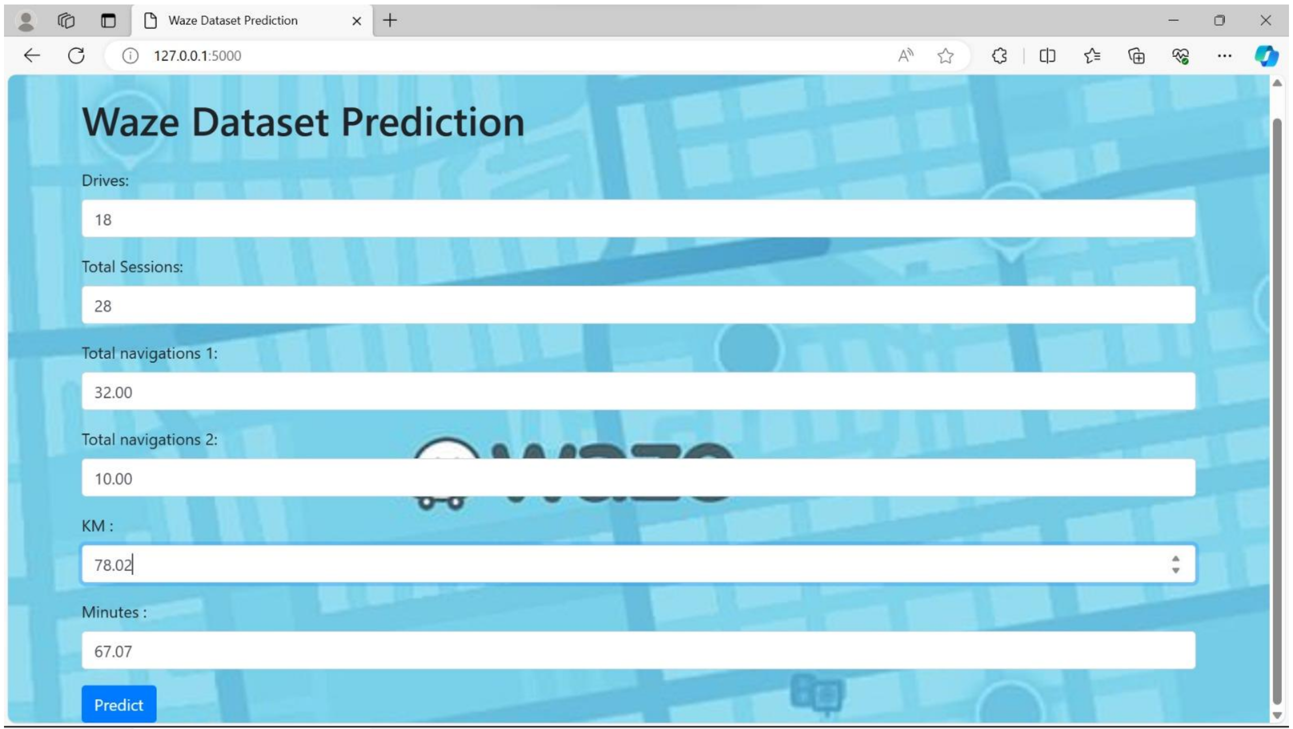


Prediction Result

127.0.0.1:5000/predict

Prediction Result

Retained



Waze Dataset Prediction

Drives:

18

Total Sessions:

28

Total navigations 1:

32.00

Total navigations 2:

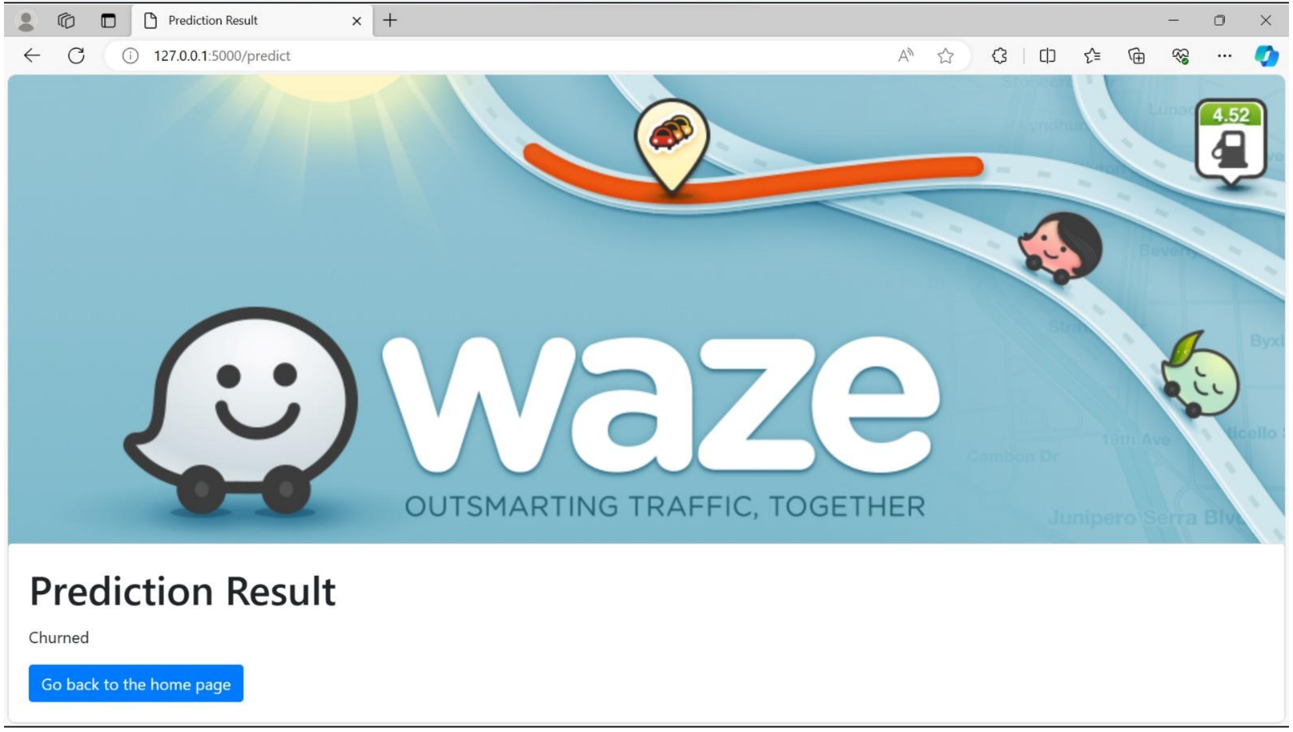
10.00

KM :

78.02

Minutes :

67.07



Prediction Result

Churned

1.c.
Our project focuses on developing a predictive model to identify users at risk of churning in the Waze application, providing valuable insights and solutions.

What Users Can Learn from Your Product:

- **Enhanced Personalization:** Users will experience a more personalized journey with Waze. They will learn that the app is adapting to their specific needs and preferences, which could include customized route recommendations, alerts, and features based on their usage patterns.
- **Community Value:** The efforts to reduce churn can highlight the importance Waze places on its user community. Users will learn that their engagement and feedback are valued and contribute to the continual improvement of the app.
- **Improved User Experience:** As the app evolves to address reasons for potential churn, users will benefit from an enhanced user interface, better functionality, and new features that are aligned with their needs and preferences.

How It Helps Solve Problems Related to the Problem Statement:

- **Proactive Engagement:** By identifying users who are at risk of churning, Waze can proactively engage with them through personalized notifications, offers, or feature recommendations, addressing any potential issues before they lead to dissatisfaction.
- **User Retention Strategies:** Insights gained from the predictive model can inform targeted user retention strategies. For example, if the model identifies that users are churning due to a lack of certain features, Waze can prioritize developing those features.
- **Enhanced Feedback Mechanism:** Integrating user feedback into the predictive model can help Waze address specific complaints or suggestions, creating a more responsive and user-centric app.

Extending the Project or Exploring Other Avenues:

- **Integration with Other Services:** Explore integrating Waze with other services (like music or podcast apps) for a more holistic driving experience, potentially reducing churn.
- **Gamification:** We may introduce gamification elements (like rewards or challenges) based on user behavior to increase engagement and decrease the likelihood of churn.
- **Predictive Analytics for New Features:** Use predictive analytics to not only identify churn risks but also to predict which new features or improvements would be most positively received by the user base.
- **Collaboration with Urban Planning Entities:** Use the data and insights gained to collaborate with city planners or transportation authorities, providing them with valuable traffic and user behavior data for urban planning.
- **Sustainability Initiatives:** Leverage the model to promote sustainability, such as recommending fuel-efficient routes or carpool options, aligning with environmental consciousness which can also enhance user retention.

By focusing on these areas, our project can effectively predict and reduce user churn, enhance the overall user experience, and provide valuable insights for continuous improvement of the Waze application.