## horizontal line



DATA SCIENCE PROJECT

DATE : 18 - AUG - 2024

**─**

Client: **ABC Tech** | Category: **ITSM - ML**

Project Ref: PM-PR-0012

**Team Members :**

Dipanjali Patra

Sandeep Chandra Sagar

Vinay C

## **ITSM Improvement through Machine Learning: Enhancing Incident Management at ABC Tech.**

# Problem Statement

1. Predicting High Priority Tickets: To predict priority 1 & 2 tickets, so that they can take preventive measures or fix the problem before it surfaces
2. . Forecast the incident volume in different fields , quarterly and annual. So that they can be better prepared with resources and technology planning
3. . Auto tag the tickets with the right priorities and right departments so that reassigning and related delay can be reduced.
4. . Predict RFC (Request for change) and possible failure / misconfiguration of ITSM assets.

### **Business Case Description**

***ABC Tech*** is an established mid-sized organization operating in the IT-enabled business sector for over a decade. They manage a significant volume of IT incidents and tickets, averaging between 22,000 to 25,000 per year. ABC Tech follows best practices in **IT Service Management (ITSM),** including incident management, problem management, change management, and configuration management processes. These ITIL practices have matured over time, reaching a high level of process maturity.

Recently, ABC Tech conducted an audit that indicated that further improvement initiatives in their ITSM processes may not provide a sufficient return on investment (ROI). Despite their mature processes, customer feedback from recent surveys has revealed that incident management, in particular, is rated poorly, suggesting there is room for enhancement.

In response to these challenges, ABC Tech's management has decided to **explore the potential of machine learning (ML) to enhance their ITSM processes**. After attending a Machine Learning conference focused on IT Service Management (ITSM), they identified four key areas where ML can contribute to improving ITSM processes within the organization.

### **Tasks**

1. Predicting High Priority Tickets: ABC Tech aims to develop an ML model that can predict high-priority tickets, specifically those categorized as priority 1 and 2. This prediction will allow them to take proactive measures to address issues or incidents before they escalate.

2. Forecasting Incident Volume: The organization plans to use ML to forecast the incident volume in different fields on a quarterly and annual basis. This predictive capability will help them better allocate resources and plan for the required technology upgrades.

3. Auto-Tagging Tickets: ABC Tech intends to implement a text classification ML model to automatically assign correct priorities and departments to incoming tickets. This automation will reduce reassignment and related delays in ticket handling.

4. Predicting RFC and ITSM Asset Misconfigurations: The organization aims to create predictive models for Request for Change (RFC) and detect potential failures or misconfigurations in ITSM assets. Identifying these issues in advance will help in preventing disruptions and improving overall ITSM asset management.

*[ The dataset that ABC Tech plans to use for these ML initiatives comprises a total of approximately 46,000 records spanning the years 2012, 2013, and 2014. The data is stored in a MySQL database with read-only access, and the relevant connection details are provided ]*

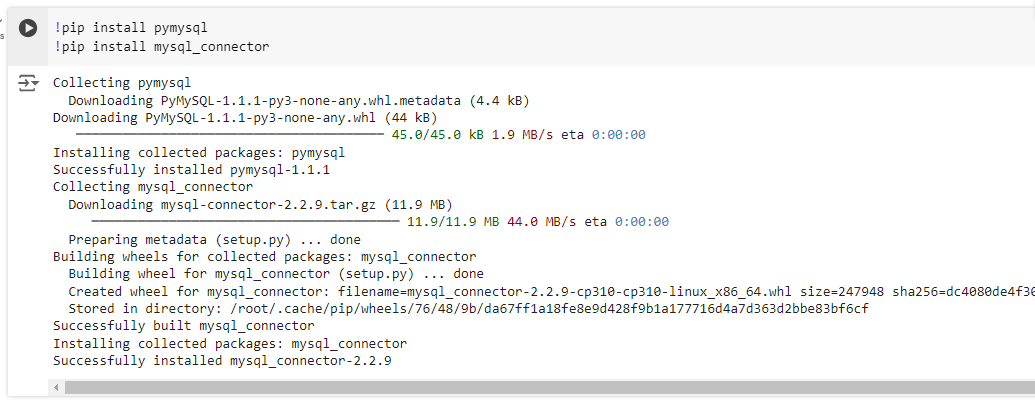
# Summary of Dataset Fields



# Priority Matrix

# Data Extraction from SQL DB

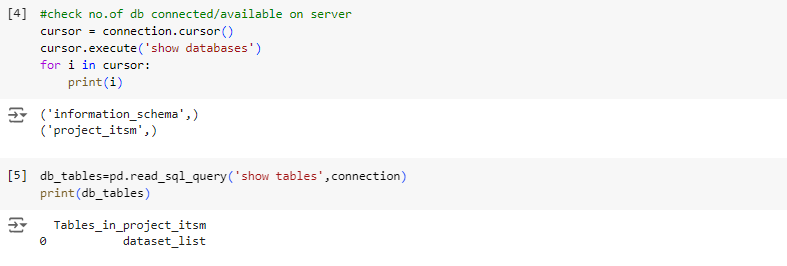
Installing pyMySQL and mysql\_connector to connect to DB



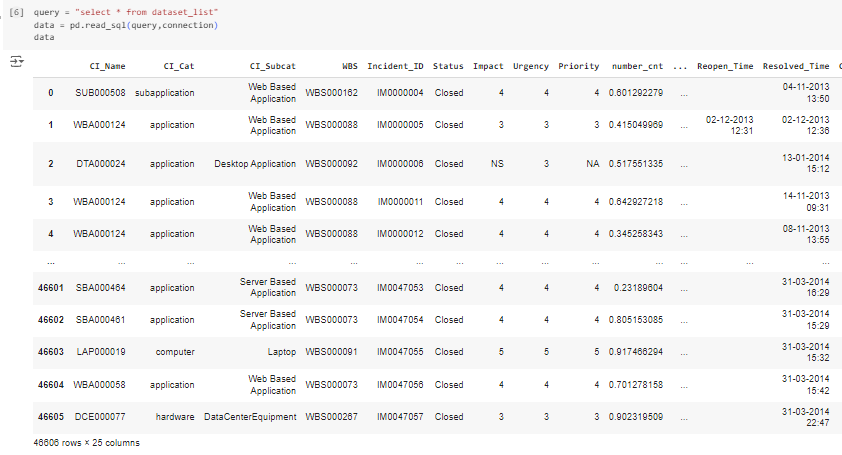
Importing necessary libraries and creating connection to server



Once connected to Server, checking the databases and tables present in it.



Projecting all columns in the table - ‘dataset\_list’



# Domain Analysis:

Certainly! Domain analysis is an essential step in any data science or machine learning project. It involves gaining a deep understanding of the domain-specific aspects of the problem you're trying to solve. In this case, we're analyzing the domain of IT Service Management (ITSM) within the context of ABC Tech's business case. Here's a domain analysis for ITSM:

**1. IT Service Management (ITSM):**

*Definition:*

IT Service Management (ITSM) refers to a set of practices and processes used by organizations to design, deliver, manage, and improve IT services for their customers and end-users.

*Importance:*

ITSM ensures that IT services are aligned with business goals, reliable, and efficiently delivered, leading to enhanced customer satisfaction and business performance.

**2. Incident Management:**

*Definition:*

Incident Management is a core ITSM process that involves identifying, categorizing, prioritizing, and resolving incidents to restore normal service operations as quickly as possible.

*Challenges:*

Common challenges in incident management include handling a high volume of incidents, determining incident priorities, minimizing response times, and reducing the impact on end-users.

**3. Priority in ITSM:**

*Definition:*

Priority is a classification system used to categorize incidents based on their severity and impact on business operations. In ITIL (IT Infrastructure Library) framework, there are typically four priority levels: Priority 1 (Critical), Priority 2 (High), Priority 3 (Medium), and Priority 4 (Low).

*Importance:*

Prioritizing incidents helps organizations allocate resources effectively and respond to critical issues promptly.

**4. ITIL Framework:**

*Definition:*

ITIL is a widely adopted framework for ITSM that provides best practices and guidelines for managing IT services, including incident management, problem management, change management, and configuration management.

*Maturity Levels:*

ITIL processes can mature over time, starting from ad-hoc practices and progressing to well-defined, controlled, and optimized processes. A mature ITIL framework leads to improved service quality and efficiency.

**5. Machine Learning in ITSM:**

*Application:*

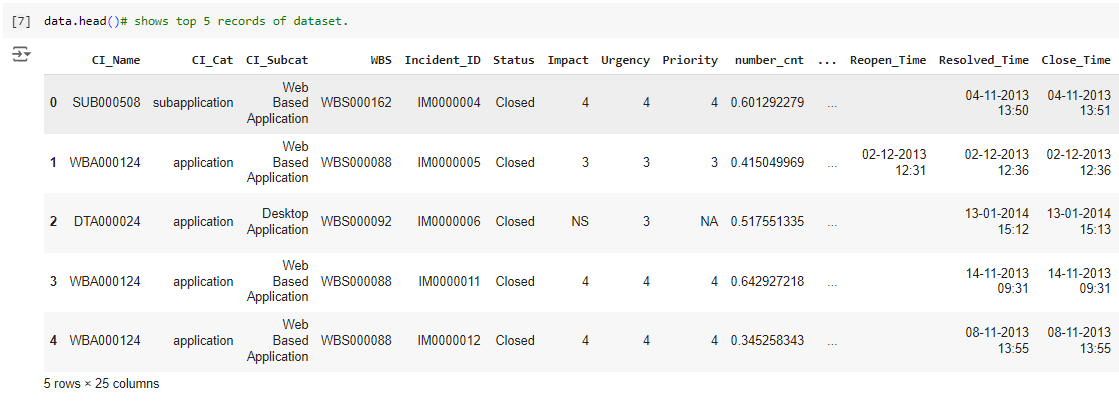
Machine learning can be applied to ITSM processes to predict incidents, automate ticket classification, forecast resource needs, and detect anomalies or misconfigurations in IT assets.

*Benefits:*

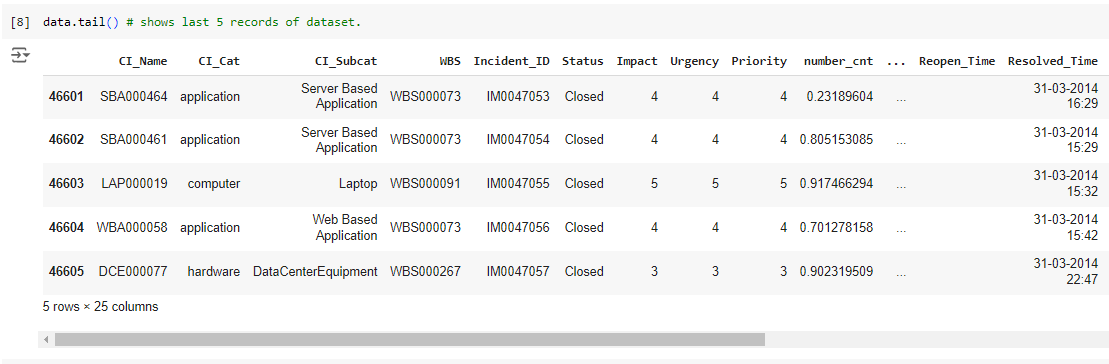
ML can enhance incident response, reduce manual workload, improve service quality, and proactively identify issues before they impact operations.

**Basic Checks**

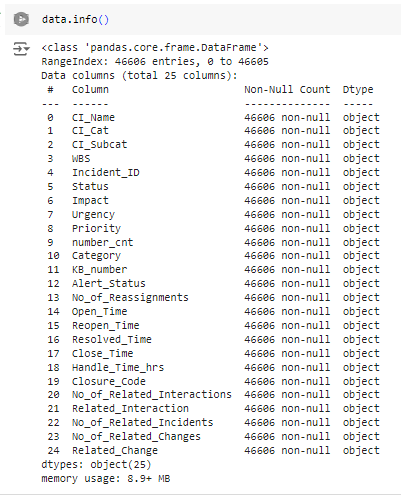
Displaying the first 5 rows of table which is stored in ‘data’



Displaying the last 5 rows of table which is stored in ‘data’



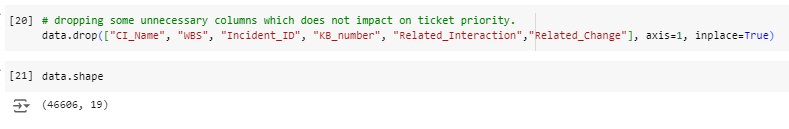
Basic information of the table



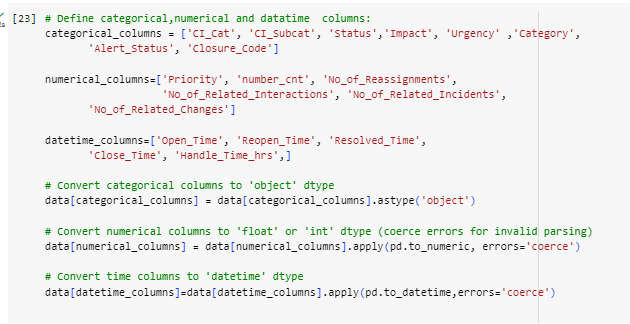
**Insights**

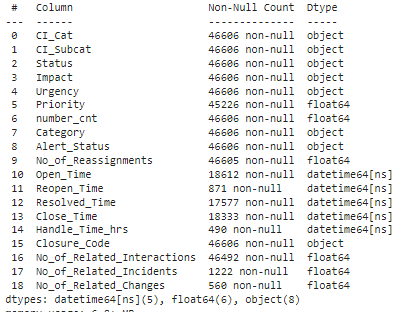
* The table contains 46606 Rows with 25 Columns. [ Can also be checked with - ‘ **data.shape()**’ ]
* All the columns are of Object type.

Removing the columns that do not impact the target variable from the dataset



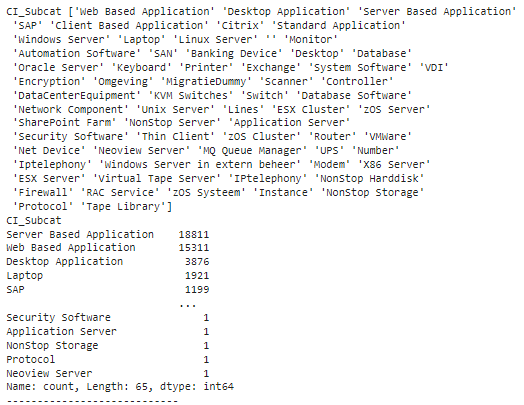
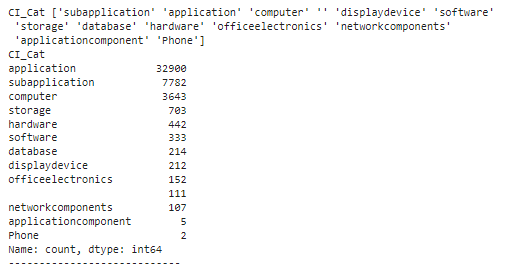
Converting the columns to corresponding Data Types

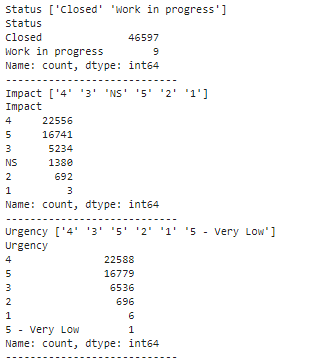


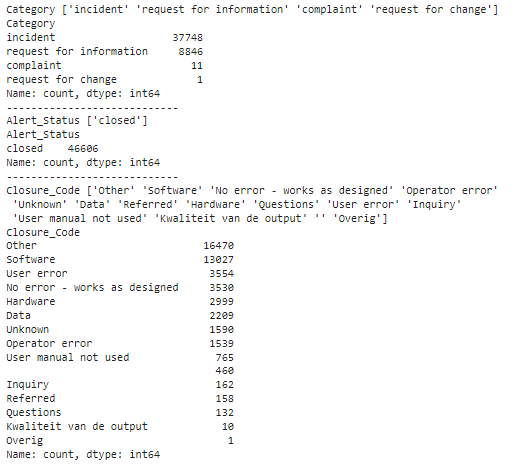


Displaying the unique values in each column

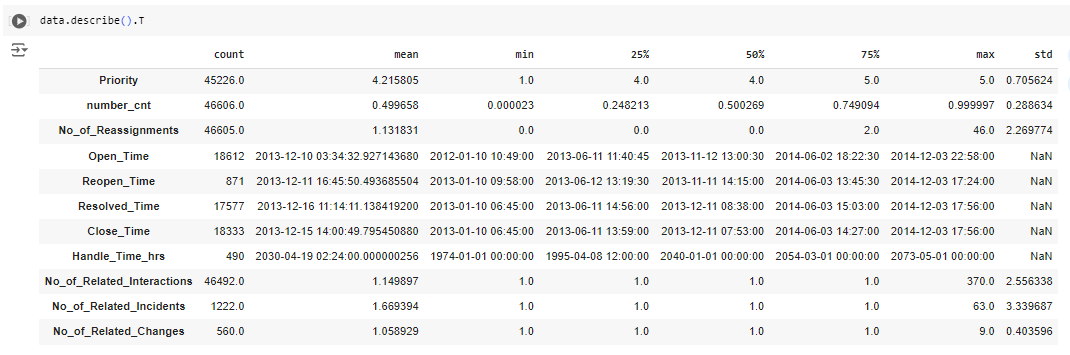








Transposing the data and describing it



Dropping the column ‘ Alert’ as its having same value throughout data



### **Basic Checks Report**

#### **Overview:**

When working with IT Service Management (ITSM) data or any dataset, it is crucial to perform basic data checks to ensure the quality and integrity of the data. These checks help identify potential issues early, ensuring that the analysis or project is based on reliable data.

#### **Data Shape:**

We examined the dimensions of the dataset using data.shape, which revealed 46,606 rows and 25 columns.

#### **Data Types:**

We verified the data types of each column using data.dtypes. All columns were identified as object types. Afterward, we separated the categorical and numerical columns for further analysis.

#### **Descriptive Statistics:**

To obtain summary statistics for numerical columns, we used data.describe().T to get the mean, minimum, maximum, and other relevant statistics. For categorical columns, we used data.describe(include='O').T to summarize their characteristics.

#### **Unique Values:**

We checked the number of unique values in categorical columns using a loop (i.e., data[i].unique()). One constant column was identified, which was subsequently dropped.

#### **Value Counts:**

We examined the distribution of values in categorical columns using a loop with data[i].value\_counts() to understand the frequency of each category.

The goal of these checks is to ensure that the data is clean, complete, and suitable for further analysis or processing.

### **Exploratory Data Analysis (EDA)**

#### **1. Univariate Analysis:**

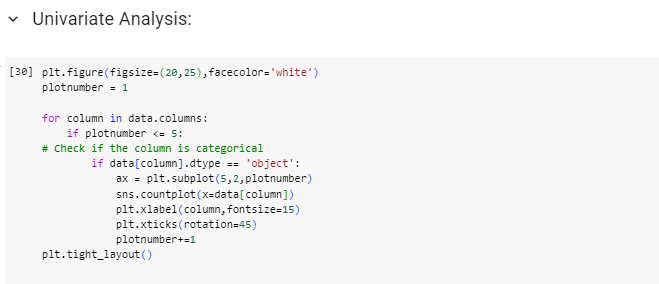
* **Overview:** Univariate analysis focuses on analyzing individual variables independently. The purpose is to understand the distribution and properties of each variable in isolation.
* **Numerical Variables:**
  + We analyze numerical columns using summary statistics (mean, median, mode, standard deviation) and visualizations like histograms, box plots, or density plots. This helps identify the spread, central tendency, and outliers in the data.
* **Categorical Variables:**
  + For categorical columns, we assess the frequency of each category using bar plots or pie charts.

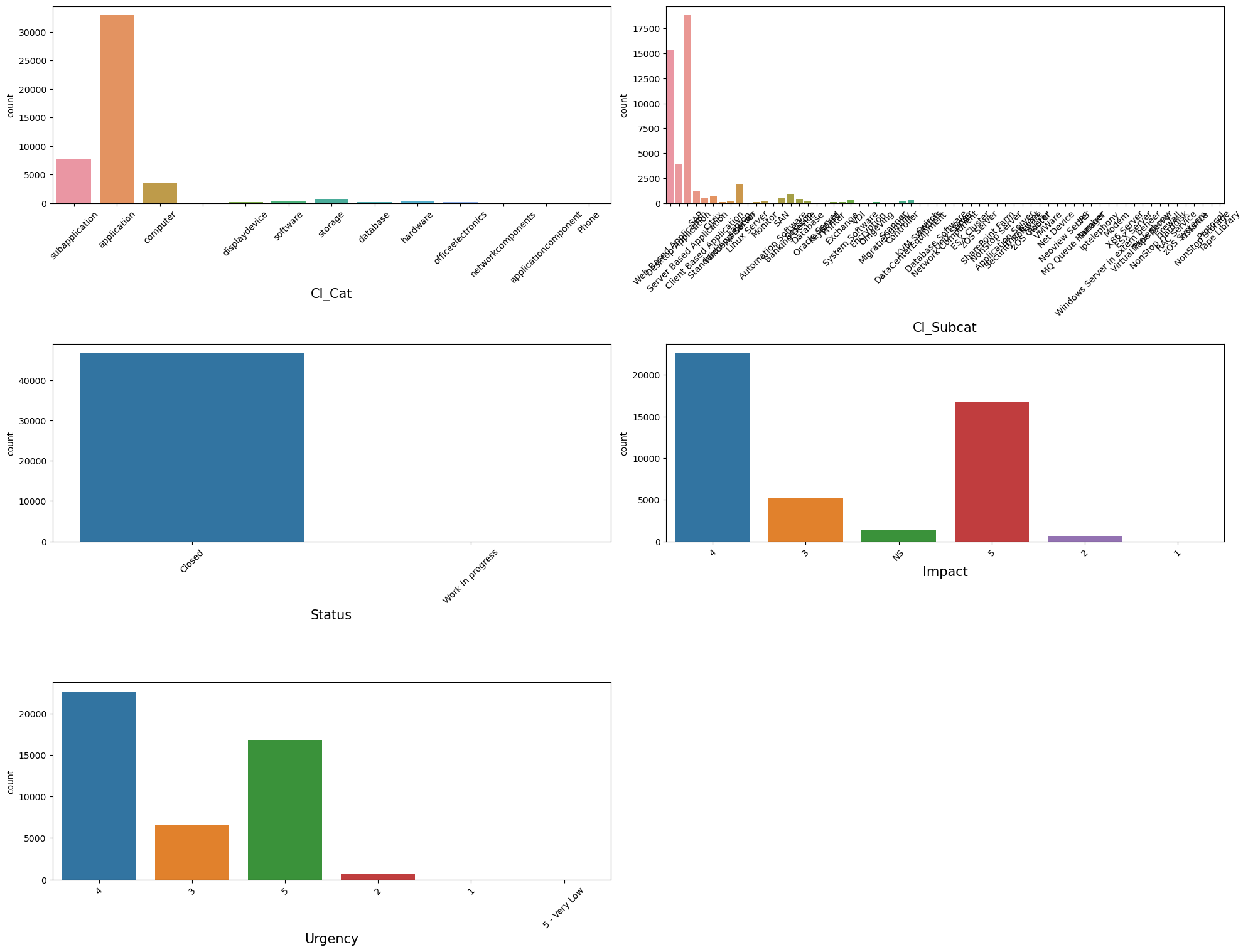
#### **2. Bivariate Analysis:**

* **Overview:** Bivariate analysis looks at the relationship between two variables to understand how they interact.
* **Numerical-Numerical Relationships:**
  + We use scatter plots, correlation matrices, or pair plots to explore the relationship between two numerical variables. This helps assess trends or correlations.
* **Categorical-Numerical Relationships:**
  + Analyzing how a categorical variable affects a numerical variable can be done using box plots, violin plots, or grouped summary statistics.
* **Categorical-Categorical Relationships:**
  + The relationship between two categorical variables is explored using crosstabs or grouped bar plots.

#### **3. Multivariate Analysis:**

* **Overview:** Multivariate analysis involves examining the relationships between three or more variables to uncover complex patterns.
* **Techniques:**
  + **Heatmaps:** A correlation heatmap is used to visualize correlations between multiple numerical variables.
  + **Pair Plots:** For exploring multiple numerical relationships, pair plots give insights into interactions between variables.

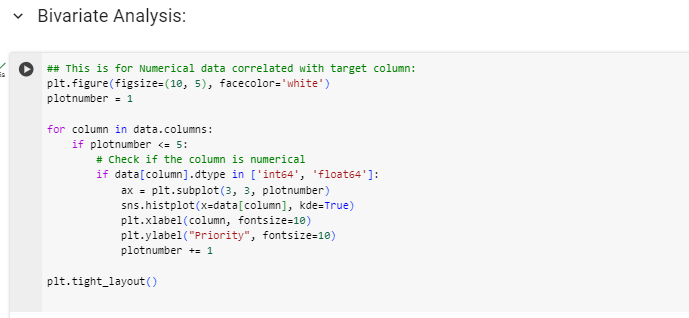


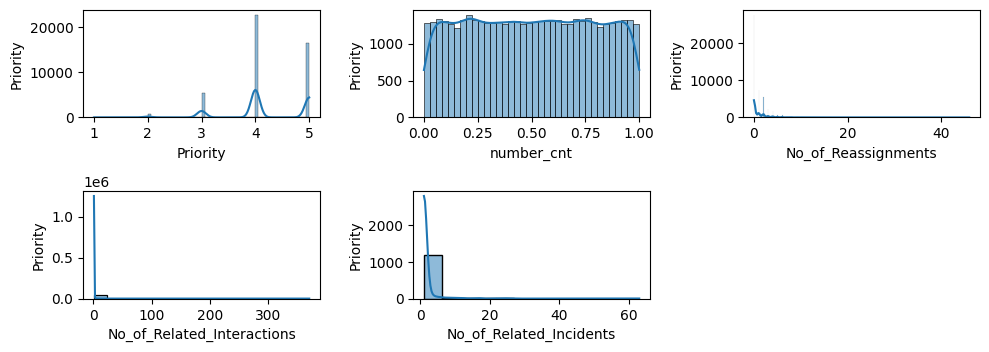


### 

### **Insights:**

1. **CI\_cat Analysis:**
   * It was observed that the *Application* category has the highest count of tickets compared to other categories within the CI\_cat column. This indicates that most tickets are related to application issues.
2. **Ticket Status:**
   * A majority of the tickets are in a *closed* state. This suggests that most of the reported issues have already been resolved.
3. **Impact and Urgency:**
   * In the impact and urgency columns, most tickets are assigned a priority of either 4 or 5. This indicates that the majority of issues are considered to have moderate to low urgency and impact.





**Insights:**

- The "No\_of\_Reassignments" column shows that most tickets are resolved on the first assignment. The number of reassignments decreases exponentially.

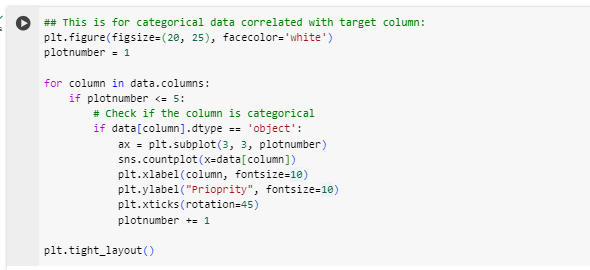
- The graph is skewed to the right, indicating that more tickets have fewer reassignments compared to those with many reassignments.

- The median number of related interactions is 1.

- 75% of tickets have 3 or fewer related interactions.

- There are a few outliers with a high number of related interactions, going up to 20.

- The distribution of the number of related incidents is skewed to the right, meaning that most incidents have fewer related incidents than those with many related incidents.

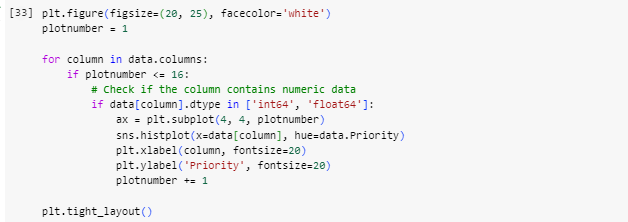


**Insights:**

- In the "CI\_cat" column, it is observed that the application category has a higher priority count compared to others.

- The status of almost all tickets is in the closed state.

- In the "Impact" and "Urgency" columns, most tickets have an impact and urgency level of either 4 or 5.

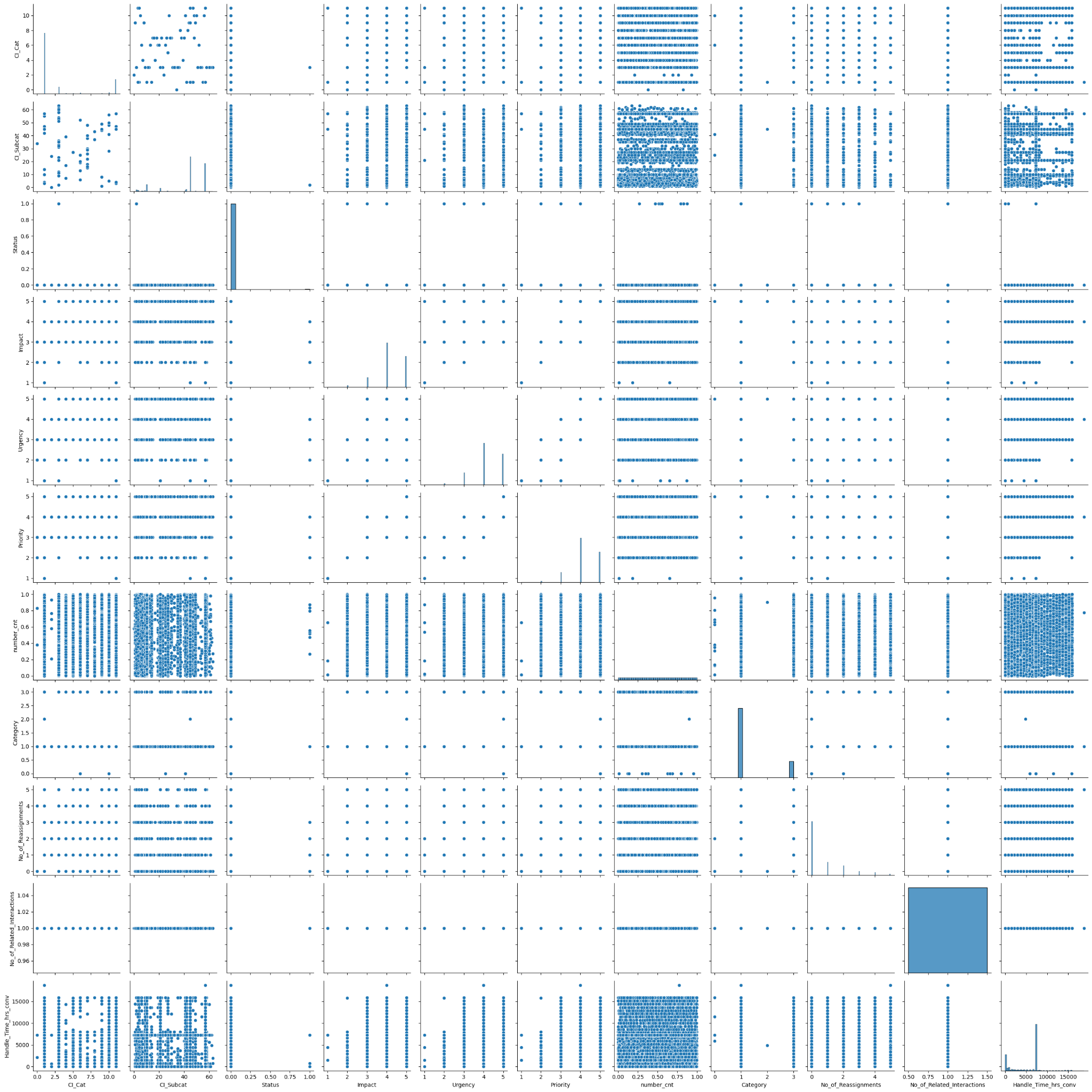
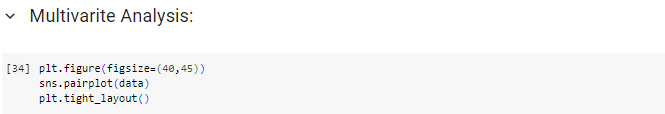


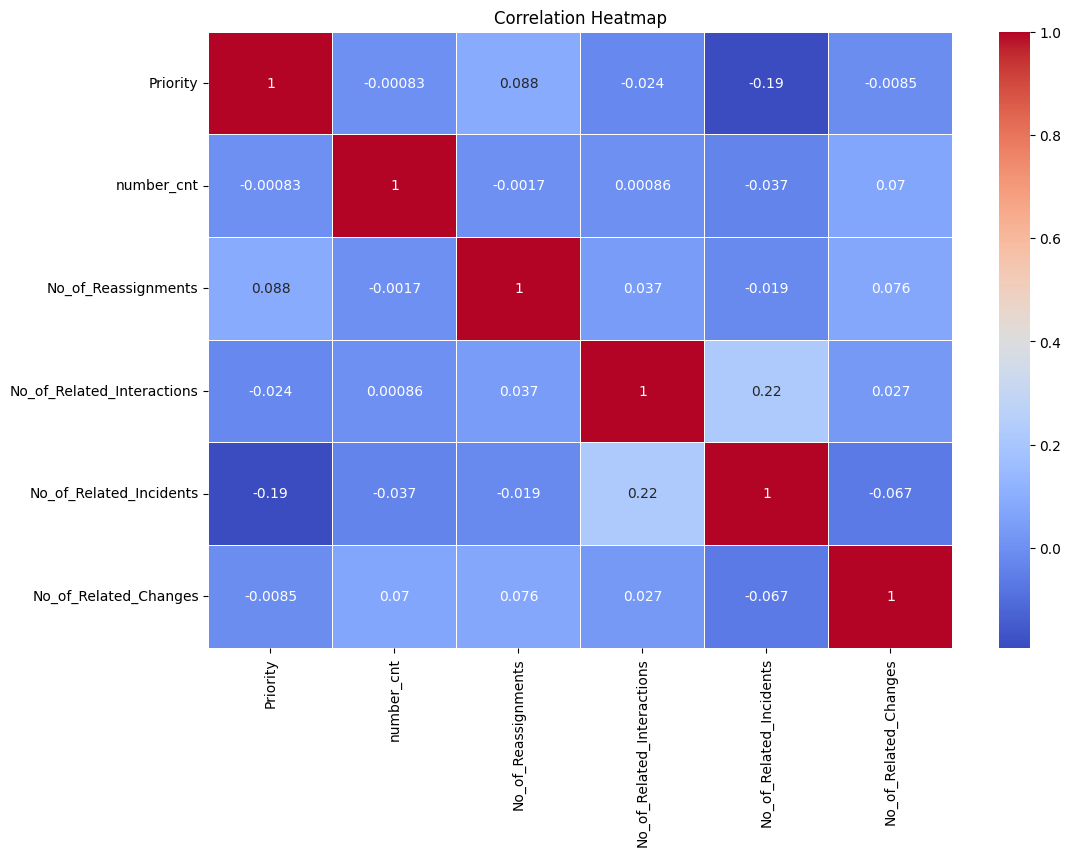
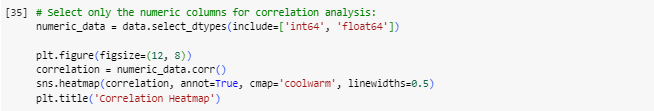
**Insights:**

- The most common number of related changes is 1.

- There are more changes with fewer related changes than those with a higher number of related changes.

- The median number of related changes is 2.

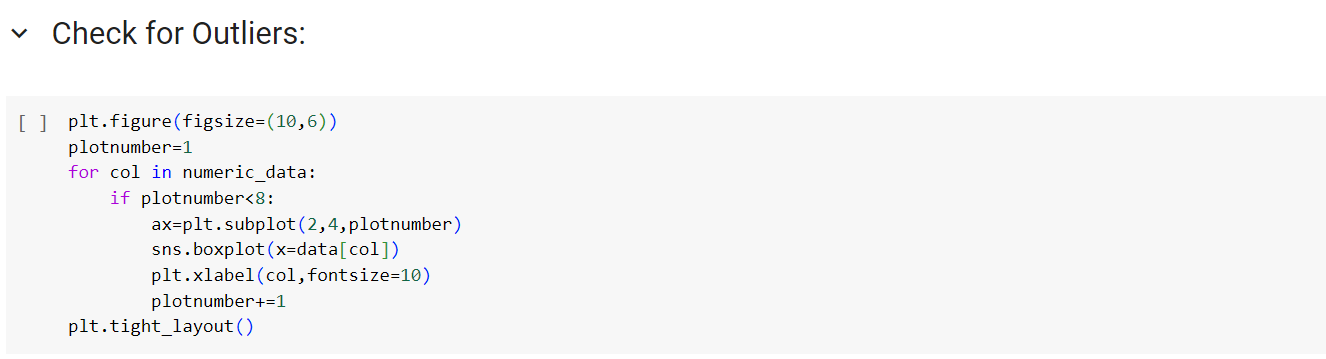


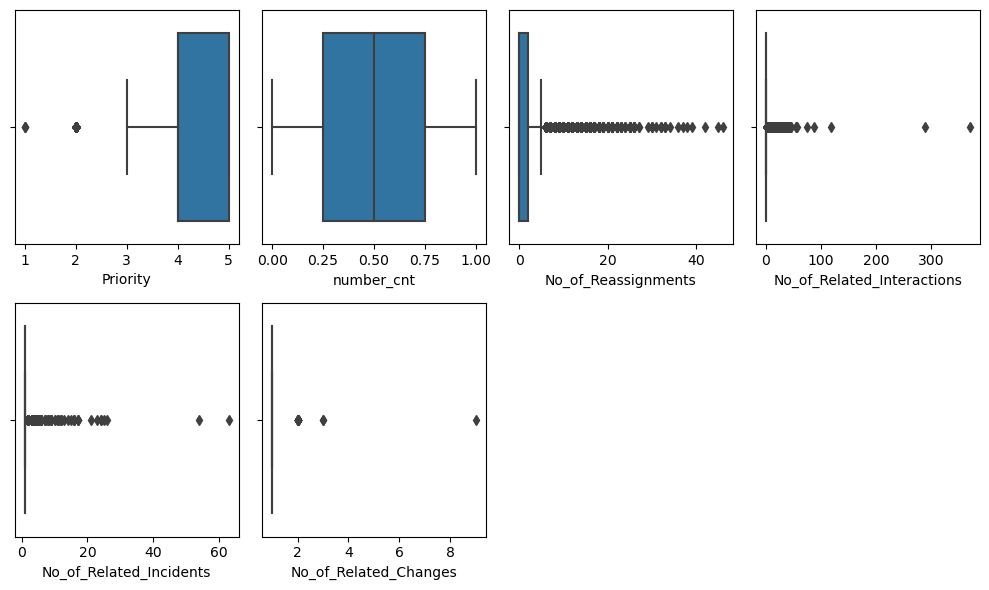
Checking correlation between columns

**Insights:**

- There is no feature highly correlated with the dependent feature (Priority), so no features are dropped.

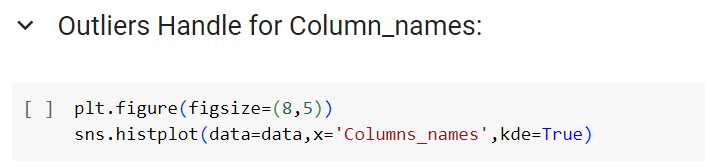
- The correlation with Priority is very low across all features.

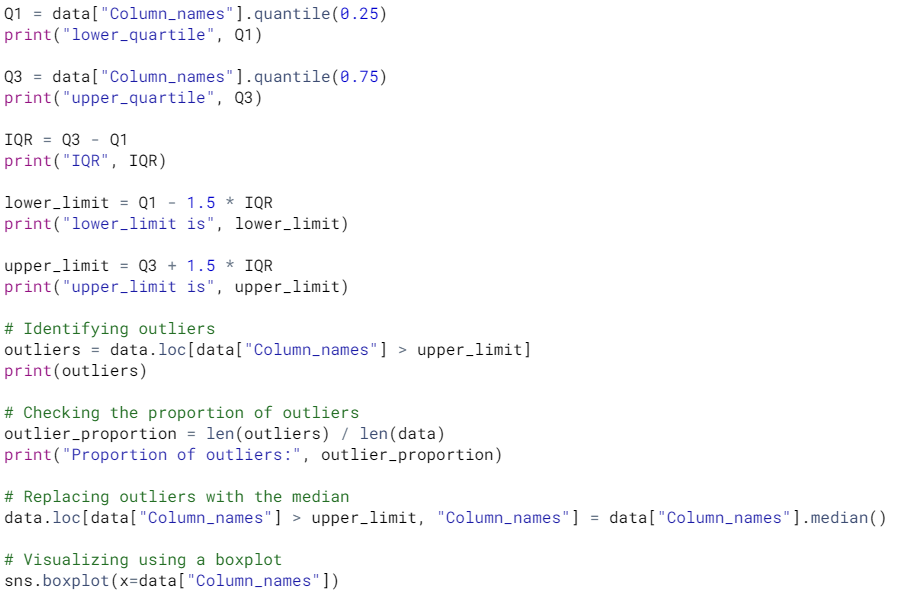




**Box Plot and Outliers Handle for :**

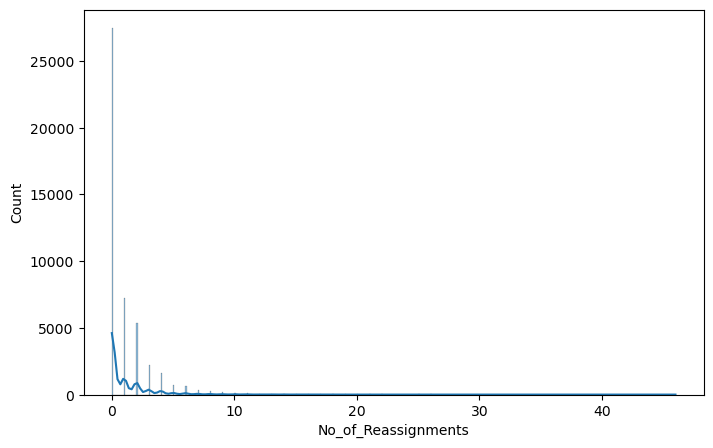
1. No\_of\_Reassignments
2. No\_of\_Related\_Interactions
3. No\_of\_Related\_Incidents
4. No\_of\_Related\_Changes







**1 No\_of\_Ressignment** 



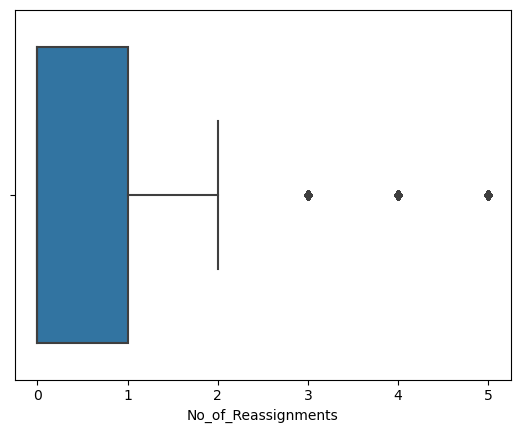
**Insights:**

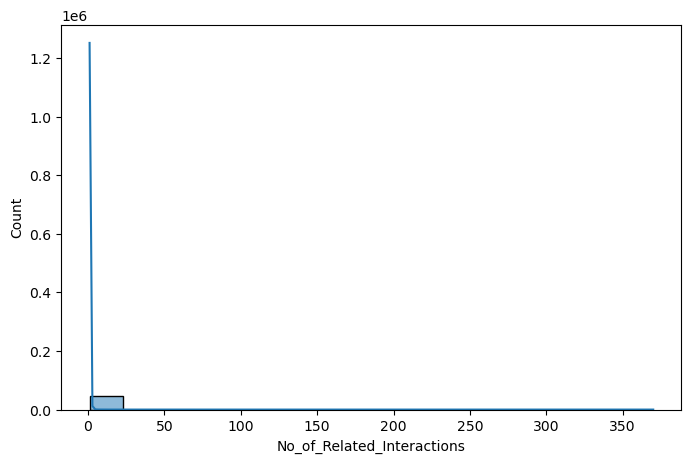
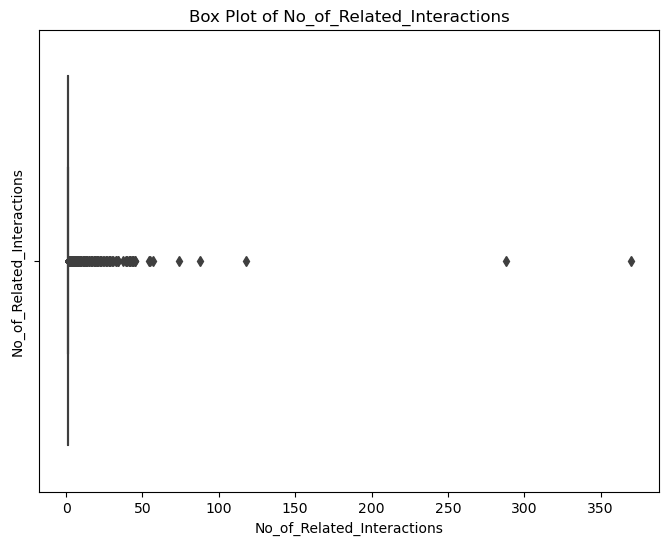
- Most tickets with 0 reassignments are resolved without being reassigned.

- The number of reassignments decreases exponentially.

- There is a slight peak at 3 reassignments.

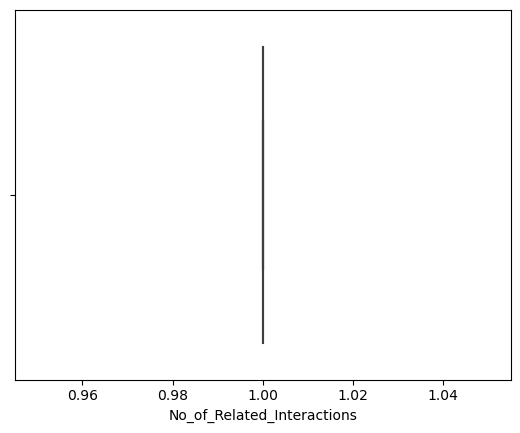
- The graph is skewed to the right, indicating that more tickets have fewer reassignments compared to those with many reassignments.



**2 No\_of\_Related\_Interactions**

**Insight:**

- Most tickets with 0 interactions are resolved without any interaction.

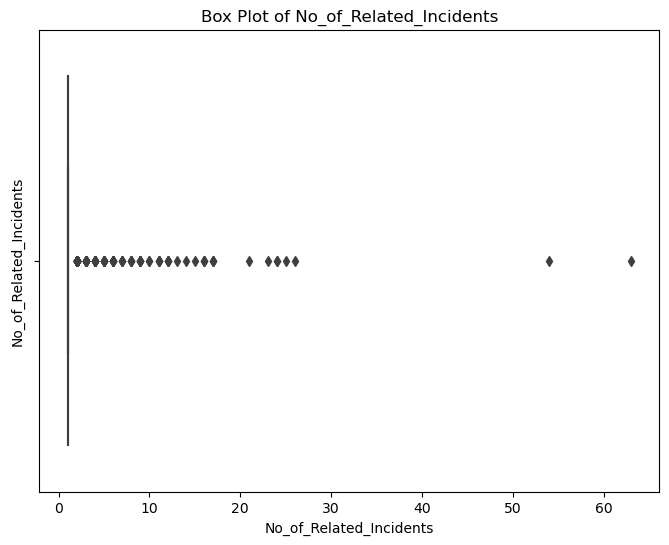


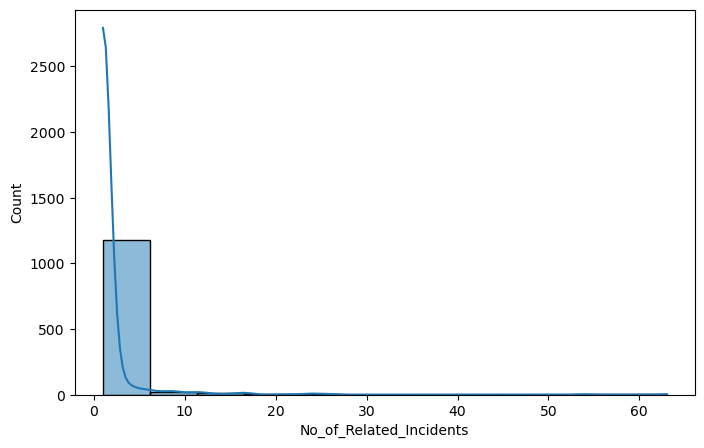
**Insights**:

- The median number of related interactions is 1.

- 75% of tickets have 3 or fewer related interactions.

- There are a few outliers with a high number of related interactions, reaching up to 20.

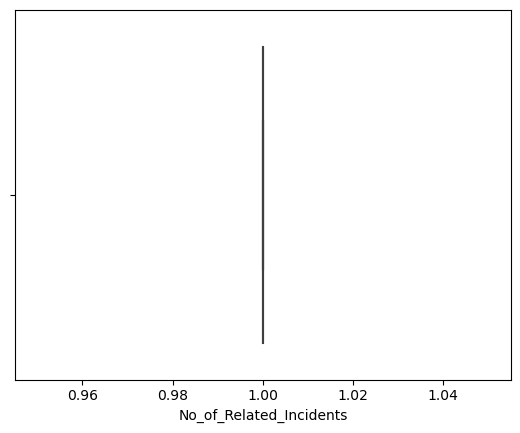
**3 No\_of\_Related\_Incidents**

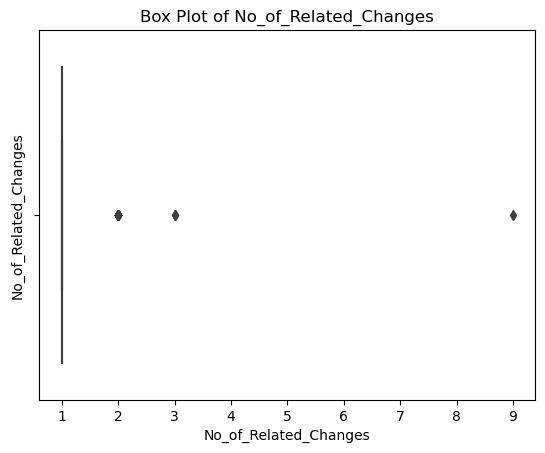
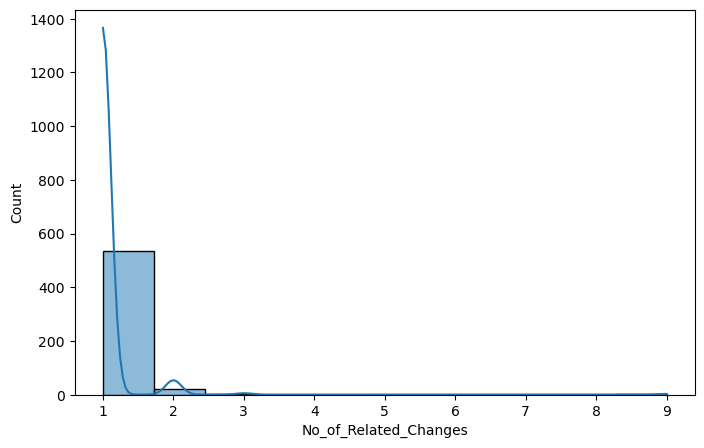


**Insights:**

- The most common number of related incidents is 1.

- The distribution of related incidents is skewed to the right, indicating that more incidents have fewer related incidents compared to those with a higher number.

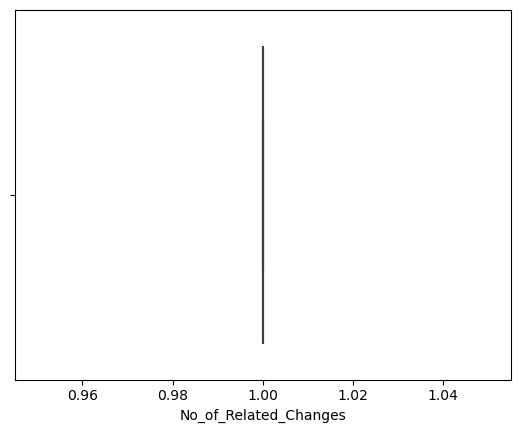
- The maximum number of related incidents is 63.

**4 No\_of\_Related\_Changes**

**Insights:**

- The most common number of related changes is 1.

- The distribution of related changes is skewed to the right, indicating that there are more changes with fewer related changes compared to those with a higher number.



# Feature Engineering

Data columns (total 18 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 CI\_Cat 46606 non-null object

1 CI\_Subcat 46606 non-null object

2 Status 46606 non-null object

3 Impact 46606 non-null object

4 Urgency 46606 non-null object

5 Priority 45226 non-null float64

6 number\_cnt 46606 non-null float64

7 Category 46606 non-null object

8 No\_of\_Reassignments 46605 non-null float64

9 Open\_Time 18612 non-null datetime64[ns]

10 Reopen\_Time 871 non-null datetime64[ns]

11 Resolved\_Time 17577 non-null datetime64[ns]

12 Close\_Time 18333 non-null datetime64[ns]

13 Handle\_Time\_hrs 490 non-null datetime64[ns]

14 Closure\_Code 46606 non-null object

15 No\_of\_Related\_Interactions 46492 non-null float64

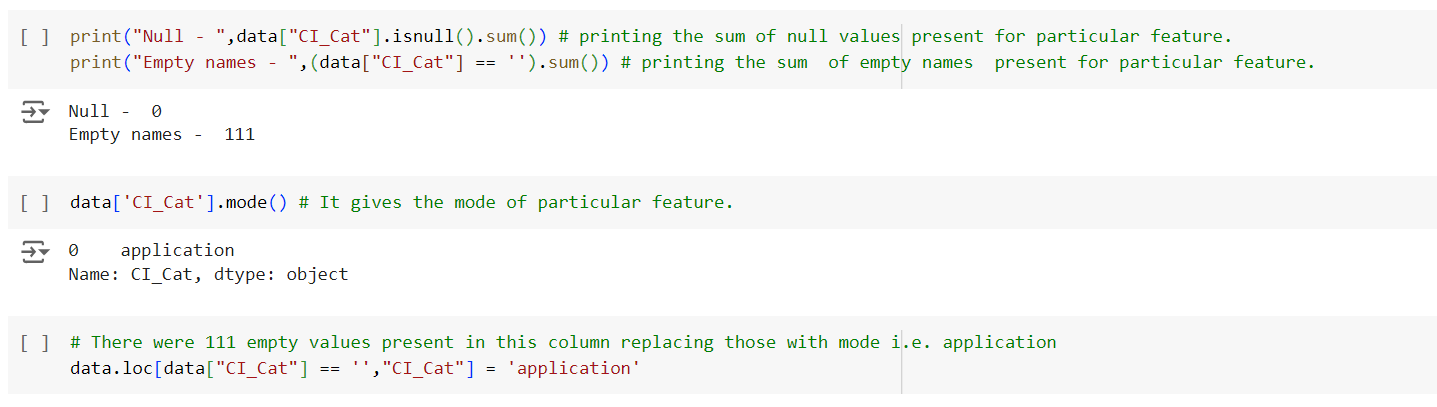
16 No\_of\_Related\_Incidents 1222 non-null float64

17 No\_of\_Related\_Changes 560 non-null float64

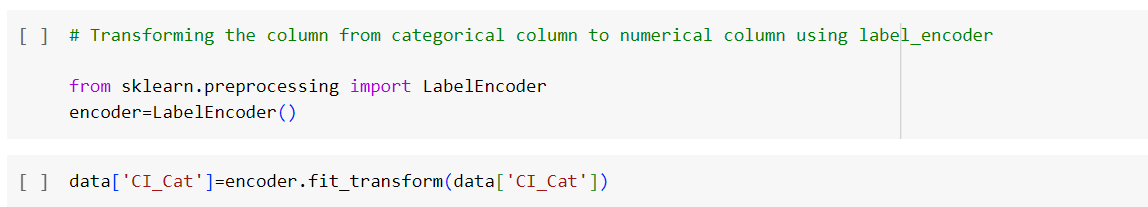
dtypes: datetime64[ns](5), float64(6), object(7)

There are 18 columns that need preprocessing:

1. Print the sum of null values and empty names for each specific feature.
2. Use the mode of each feature to fill in the null values or empty names.



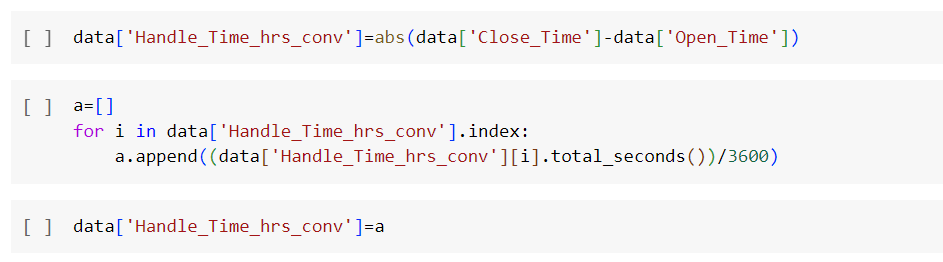
1. Convert the categorical columns to numerical format using LabelEncoder.



**\*\*\*\*\*Doing same for all the columns in table\*\*\*\*\***

The column ***data['Handle\_Time\_hrs']***does not contain meaningful information, so we can manually create ***Handle\_Time\_hrs\_conv*** and drop the original column.

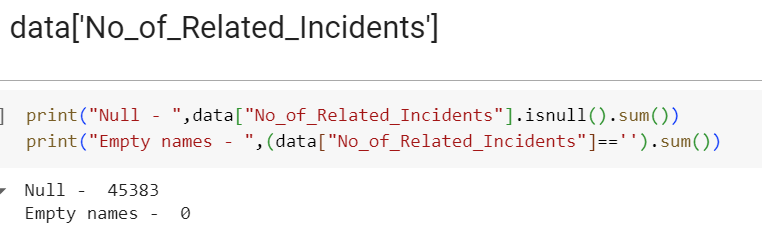
To do this, we will calculate the difference between the ***open\_time* and *close\_time*** columns in days and convert it to hours.

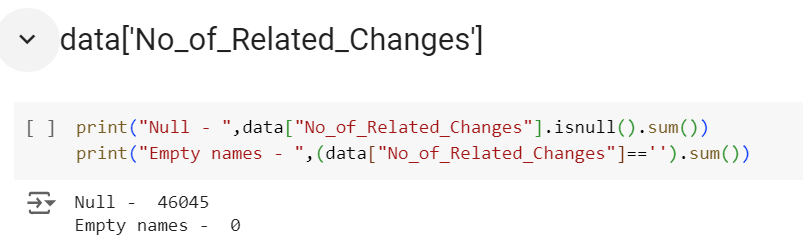
data.drop("Handle\_Time\_hrs", axis=1, inplace=True)

Since the closure code does not determine the ticket priority and its importance is only considered at a later stage of ticket resolution, we can drop the column.

data.drop('Closure\_Code', axis=1, inplace=True)

We are dropping the columns No\_of\_Related\_Changes and No\_of\_Related\_Incidents because they contain more than 50% null values.





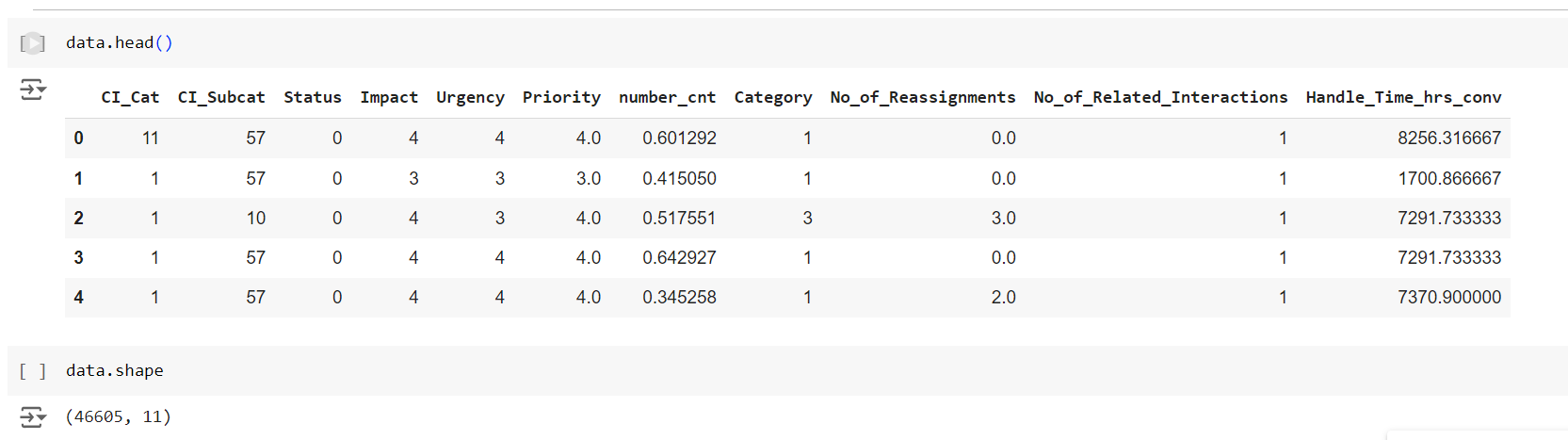
data.drop('No\_of\_Related\_Changes',axis=1,inplace=True)

data.drop('No\_of\_Related\_Incidents',axis=1,inplace=True)

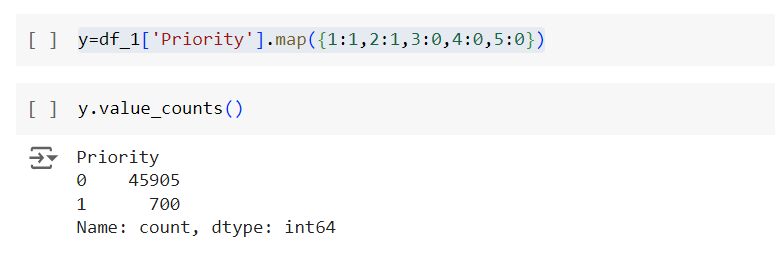
Since we have already used the columns Close\_Time, Open\_Time, and Resolved\_Time to create handle\_time\_hrs\_conv, we can drop these columns.

data.drop(['Open\_Time','Resolved\_Time','Close\_Time',],axis=1,inplace=True)

**Preprocessed Dataset for ML**



**Task 1: Predicting High Priority Tickets**  
The goal is to predict priority 1 and 2 tickets so that preventive measures can be taken or the problem can be fixed before it arises.

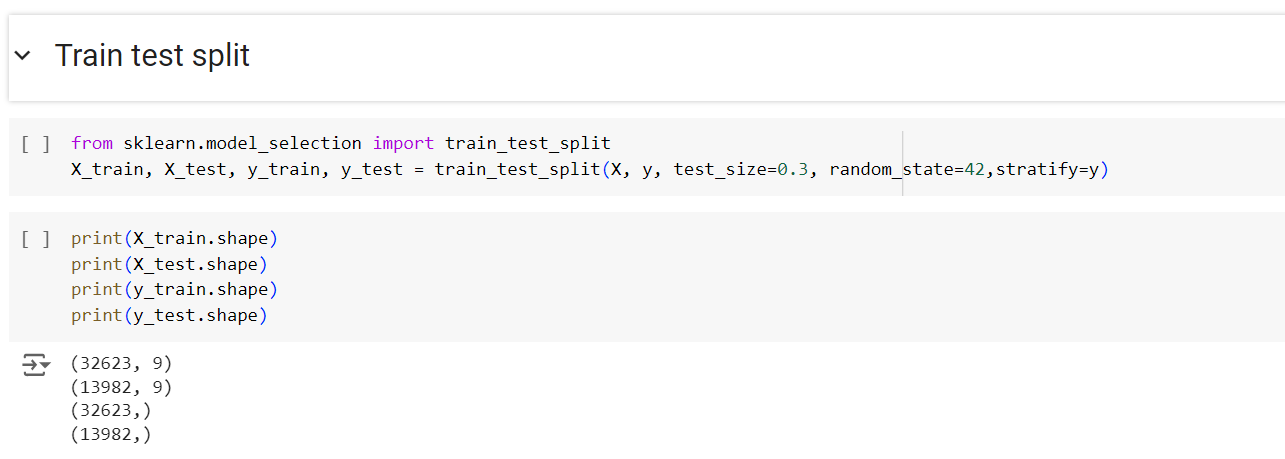


Mapping the 1-5 level priorities to 1 and 0 as per Task 1.

Priority 1 and 2 will be mapped to 1 (high priority), while priorities 3, 4, and 5 will be mapped to 0 (low priority).

**Train-Test Split**:

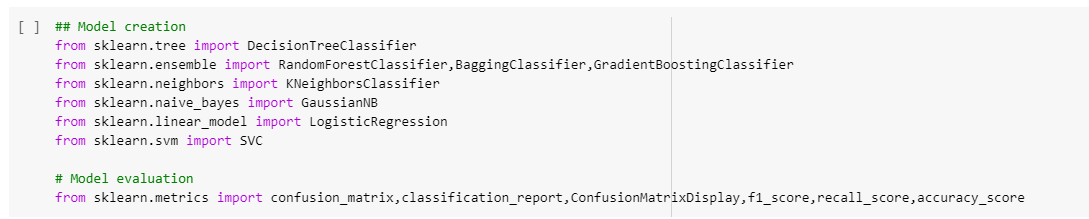
This process involves dividing a dataset into two subsets: one for training a model and the other for testing its performance. Typically, the data is split into a training set (used to train the model) and a test set (used to evaluate the model's accuracy and generalization ability). This helps ensure that the model is tested on data it hasn't seen during training, providing a more realistic assessment of its performance.



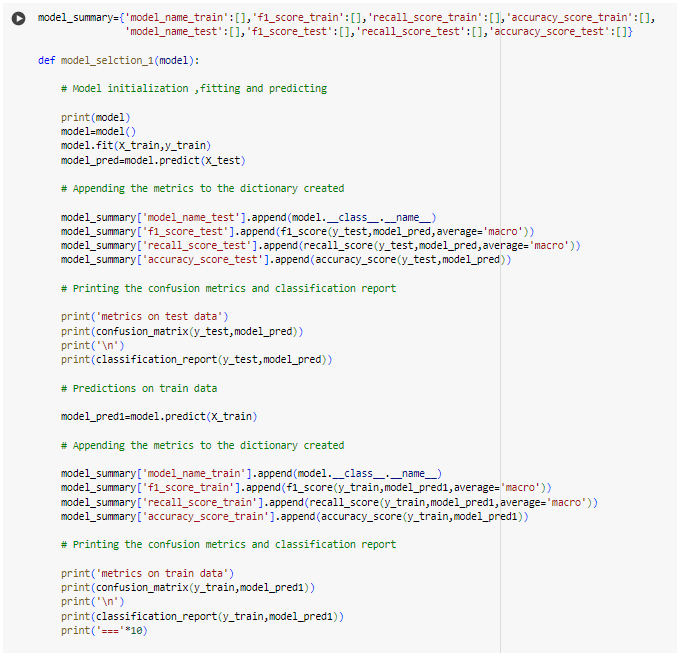
**Logic Behind the Function**

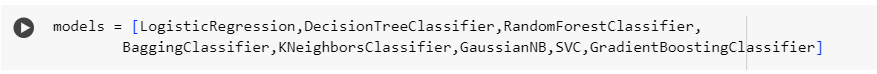
1. Create a dictionary named model\_summary with keys and null values.
2. The function model\_selection\_1 will take a model as a parameter.
3. Inside the function, the model will be initialized and stored in a variable called model.
4. The model will be trained using x\_train and y\_train.
5. The model will make predictions on the test data.
6. After prediction, evaluation metric values will be added to the dictionary with corresponding keys.
7. The function will print the confusion matrix and classification report for the model.
8. The same steps will be repeated for the training data.

**Importing necessary libraries for Model creation and evaluation:**

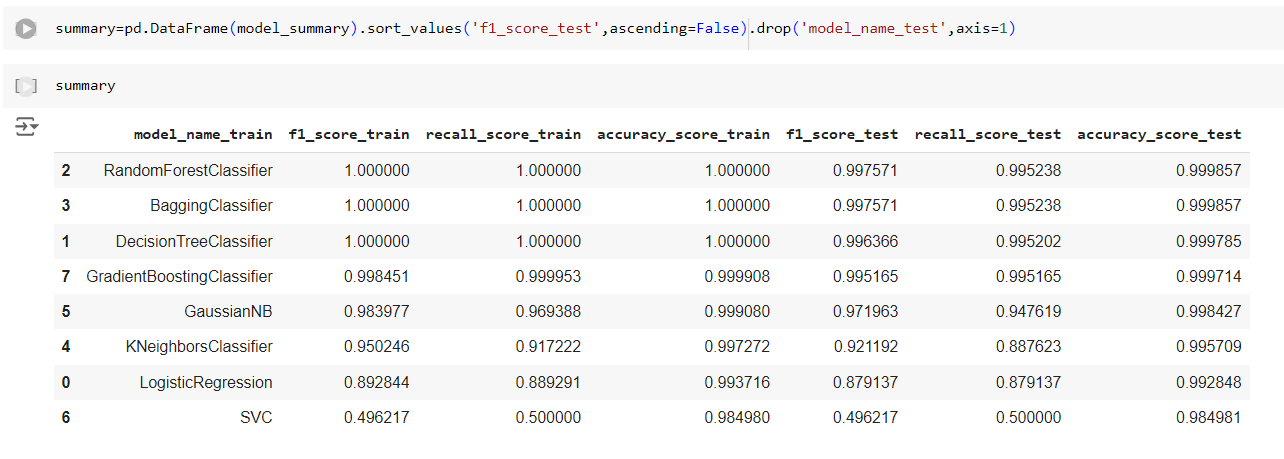


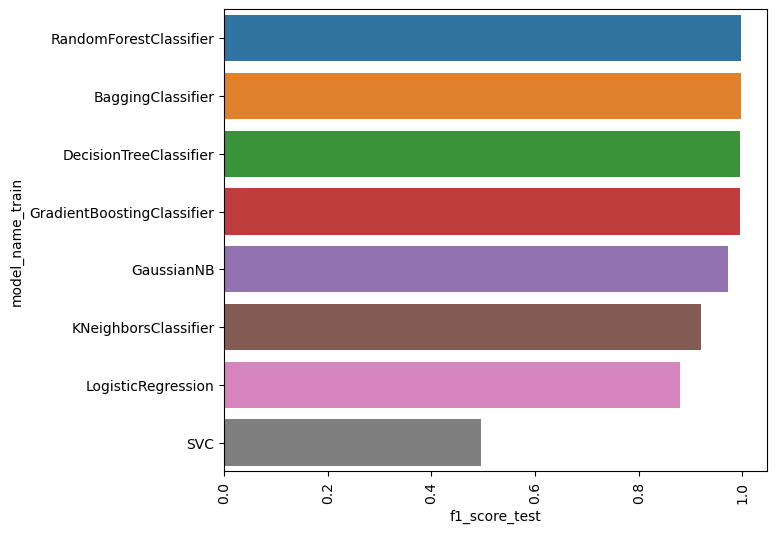
**Creating Model and Evaluating**

****

**Models are : **

**Summary of Chosen Metrics for All Models**

****

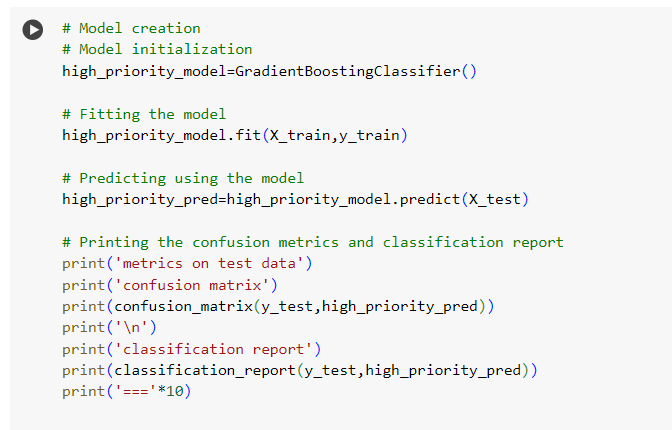
****

**Model Selection for Task 1**

From the graph, it is observed that the RandomForestClassifier, BaggingClassifier, DecisionTreeClassifier, and GradientBoostingClassifier are performing well compared to other algorithms, achieving performance above 95%. Therefore, additional optimization techniques are not required separately.

We are choosing the GradientBoostingClassifier over the other models—RandomForestClassifier, BaggingClassifier, and DecisionTreeClassifier—as it has demonstrated better performance more consistently.

We will proceed with creating the GradientBoostingClassifier model for further use.



**Accuracy**: Measures the proportion of correctly predicted instances out of the total instances. It provides a general idea of the model's performance.

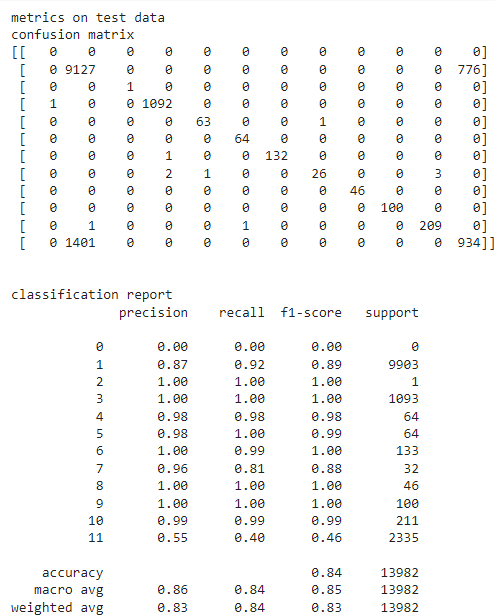
**Precision**: Indicates the proportion of true positive predictions among all positive predictions made by the model. It reflects how many of the predicted positives are actually positive.

**Recall (Sensitivity)**: Shows the proportion of true positive predictions among all actual positive instances. It reflects how well the model captures the actual positives.

**F1 Score**: The harmonic mean of precision and recall. It balances the two metrics and provides a single score to evaluate the model's performance, especially when dealing with imbalanced classes.

**Confusion Matrix**: A table that shows the counts of true positives, true negatives, false positives, and false negatives. It helps in understanding the model's classification performance.

**Classification Report**: A detailed report that includes precision, recall, F1 score, and support for each class. It provides a comprehensive view of the model's performance in each class.



**Task 2: Forecasting**

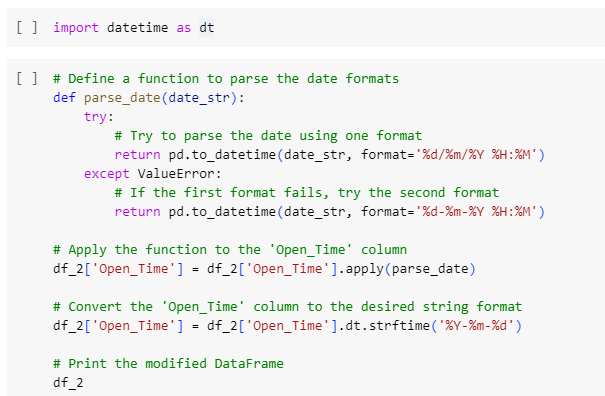
Forecast the incident volume across different fields on a quarterly and annual basis. This will help in better resource allocation and technology planning.

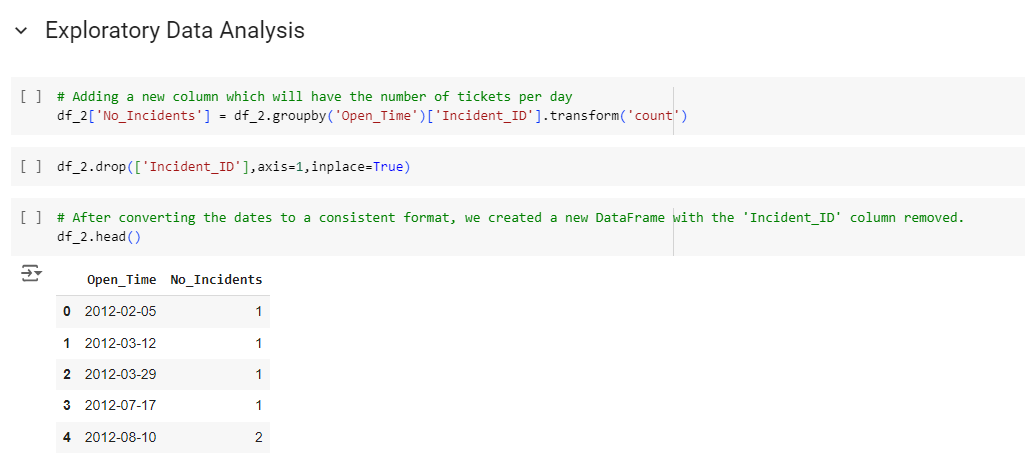
**# Imporing the necessary columns**

**# Importing the necessary columns from dataset, which include 'Incident\_ID' and 'Open\_Time'.**

**df\_2 = df.loc[:,['Incident\_ID','Open\_Time']]**

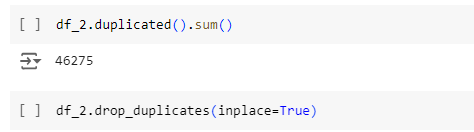
**Parsing the date to one format [%Y-%m-%d]**

****

****

**Data Cleaning and Indexing**

1. Calculated the number of incidents per day and identified duplicate values in the dataset.
2. After removing the duplicates, set the Open\_Time column as the index.
3. Checked the date range to ensure the data is complete and accurately represents the incidents over time.

****



**Set the Open\_Time column as the index of the DataFrame df\_2 and convert it to datetime format:**

df\_2 = df\_2.set\_index('Open\_Time')

df\_2.index = pd.to\_datetime(df\_2.index)

**Check the range of dates:**

print(df\_2.index.min(), 'to', df\_2.index.max())

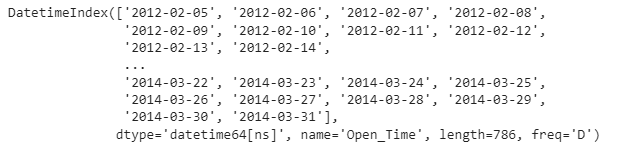
**Create a new Series with a daily frequency:**

data1 = df\_2['No\_Incidents']

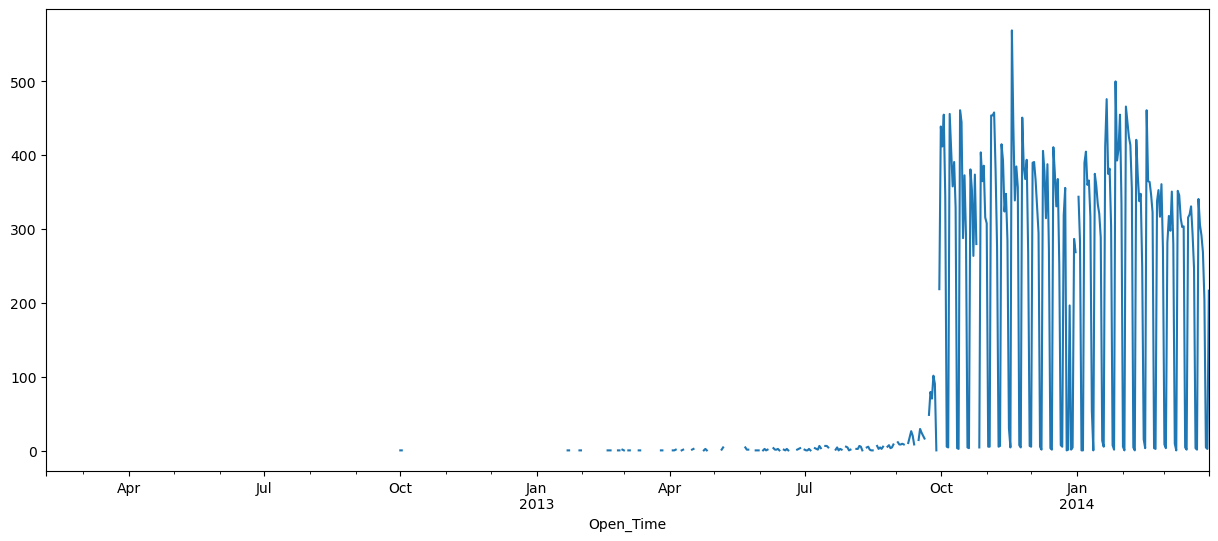
data1 = data1.asfreq('D')

**Display the new Series index to verify:**

data1.index



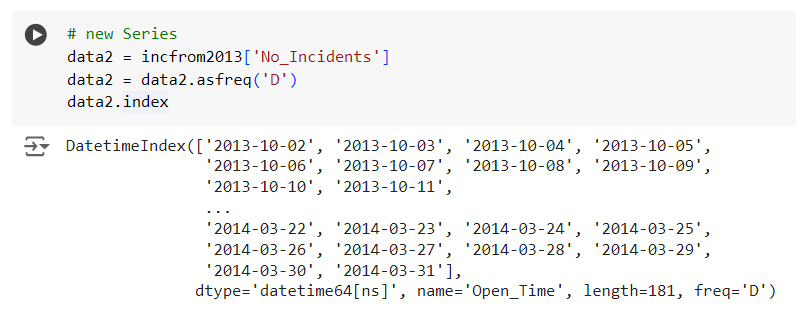
**Data Analysis and Visualization**



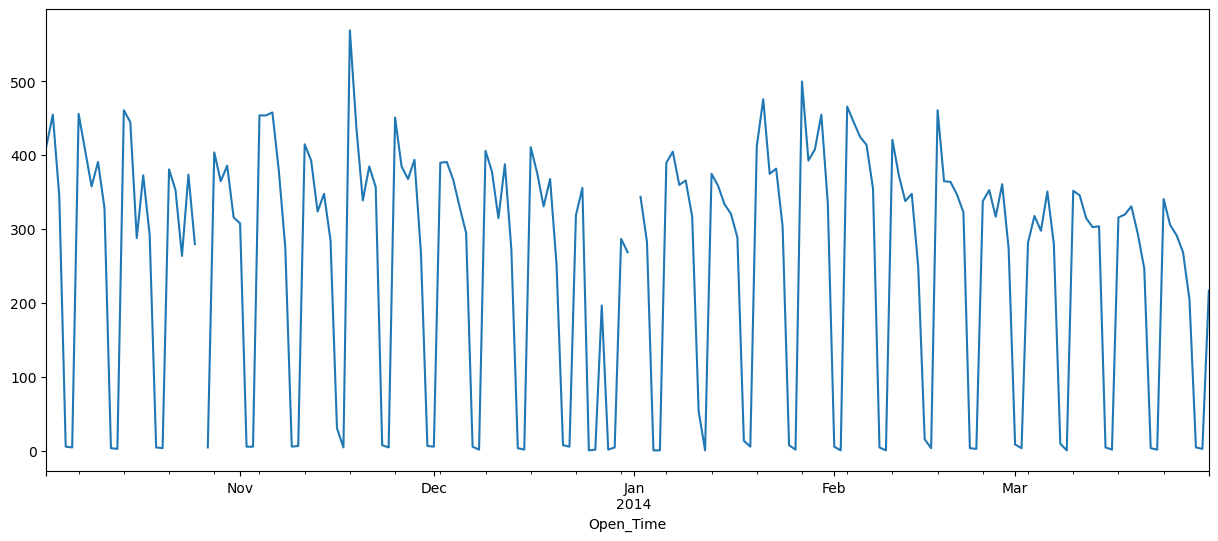
Analyzing Incident Volume

1. Created a time series plot of the number of tickets per day to better understand the incident volume.
2. Observed a significant increase in incidents after October 2013 from the plot.
3. Note that there were fewer tickets before October 2013, so we will focus on the data from after this date.





Plotting Number of Tickets Per Day After October 2013

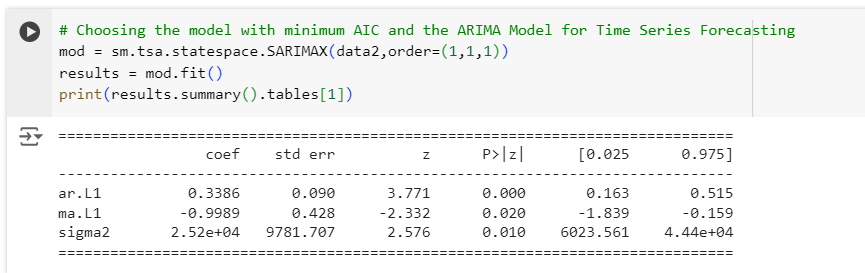




Time series forecasting involves predicting future values based on previously observed values

**Choosing the ARIMA Model with Minimum AIC for Time Series Forecasting**

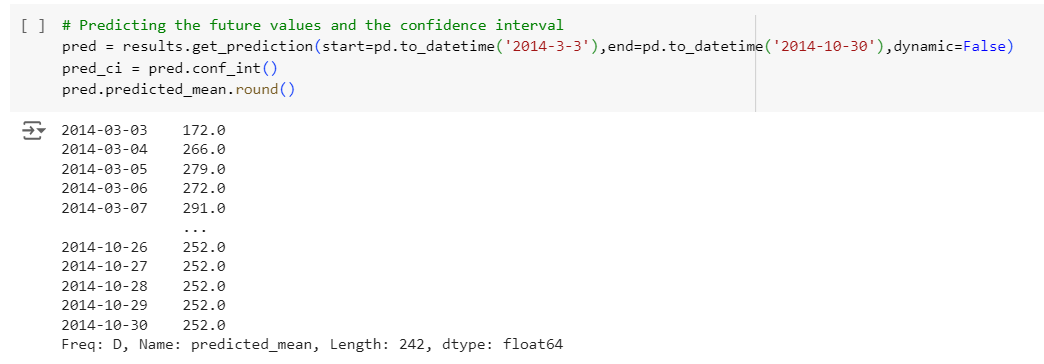
1. **Model Selection Using AIC**:
   * Evaluate different ARIMA models by calculating their Akaike Information Criterion (AIC) values. The AIC helps in selecting the model that best balances fit and complexity.
   * Select the ARIMA model with the lowest AIC value as it indicates the best trade-off between model accuracy and complexity.
2. **Fitting the Selected ARIMA Model**:
   * Once the model with the minimum AIC is identified, fit the ARIMA model to the time series data using the chosen parameters.
3. **Forecasting**:
   * Use the fitted ARIMA model to generate forecasts for future values based on the historical data.



**Summary of the Selected ARIMA Model (1,1,1)**

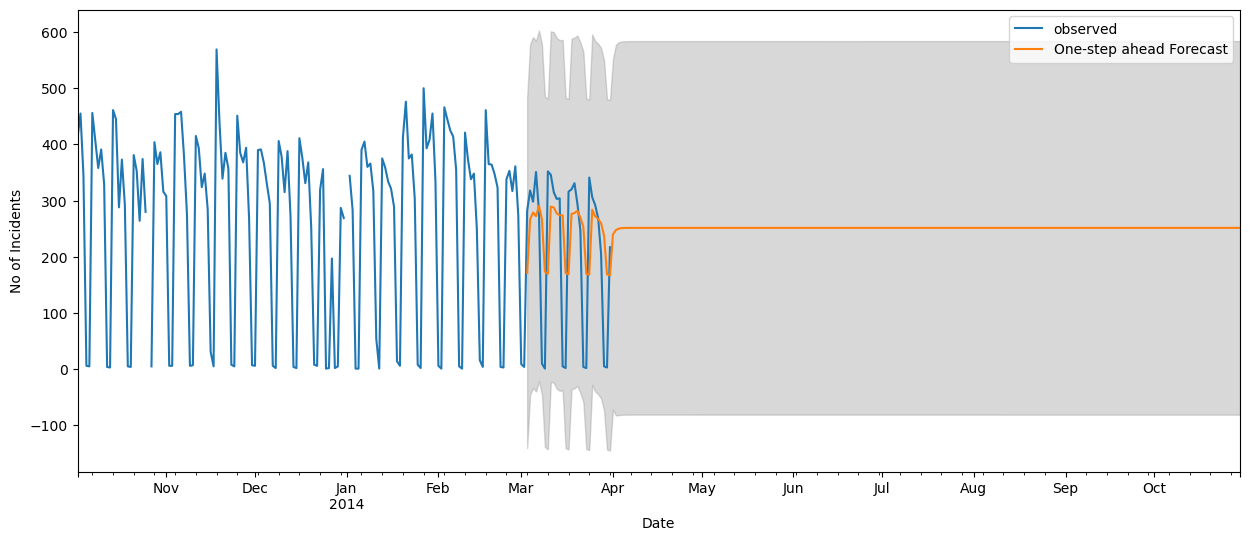
The summary of the selected ARIMA model with parameters (1,1,1) includes:

1. **Coefficients**:
   * **Autoregressive (AR) Term**: Coefficient value indicating the influence of past values on the current value.
   * **Moving Average (MA) Term**: Coefficient value reflecting the influence of past forecast errors on the current value.
2. **Sigma² Value**:
   * Represents the variance of the residuals (errors). A lower sigma² value indicates that the model's residuals are closer to zero, suggesting a better fit of the model to the data.



The selected ARIMA model was used to predict future incident volumes. The forecast was made for the period from March 3, 2014, to October 30, 2014.

**Visualization of Forecasted Incident Volumes**

The predicted incident volumes were plotted alongside the observed data for visualization. This comparison allowed us to assess the model's performance in forecasting incident volumes.

**Task 3: Tag Tickets**

Automatically tag tickets with the appropriate priorities and departments to minimize reassignment and related delays.

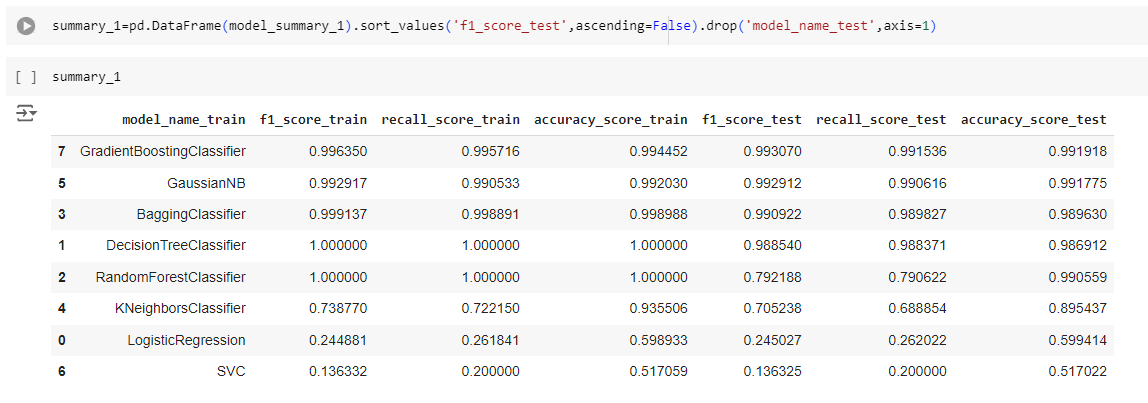
**Model Creation and Evaluation**

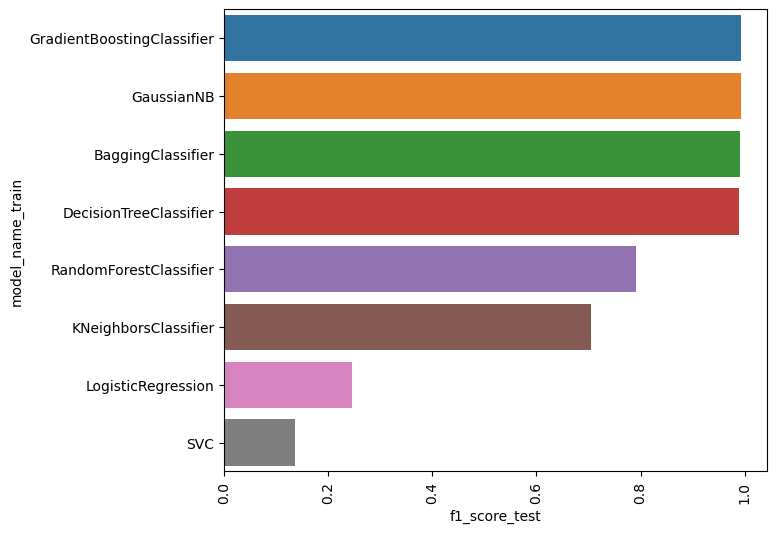


Test train split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X1, y1, test\_size=0.3, random\_state=42,stratify=y1)

**Summary of Chosen Metrics for All Models**



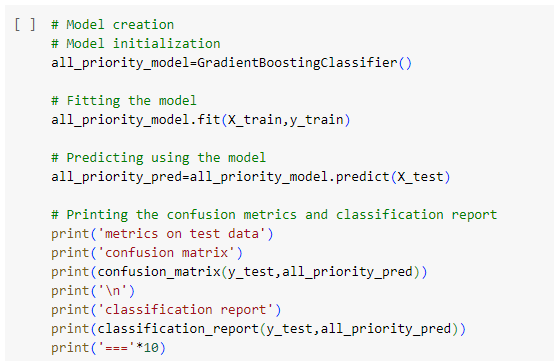


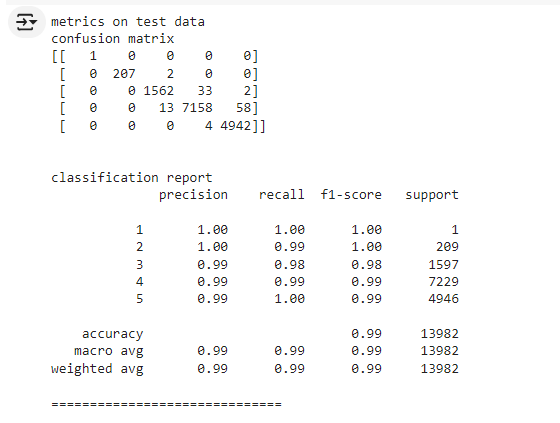
**Model Selection for Task 3: Auto-Tag Tickets with the Right Priority**

From the analysis, it is observed that the GradientBoostingClassifier, GaussianNB, BaggingClassifier, and DecisionTreeClassifier models are performing well compared to other algorithms, achieving performance above 95%. Therefore, additional optimization techniques are not required.

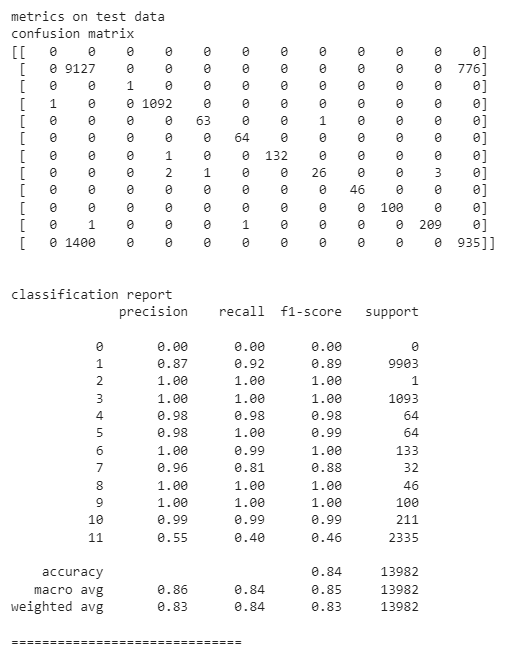
We are selecting the GradientBoostingClassifier over GaussianNB, BaggingClassifier, and DecisionTreeClassifier because it has consistently demonstrated better performance.

We will proceed with creating the GradientBoostingClassifier model for further use.





We will proceed with creating the GradientBoostingClassifier model for further use.



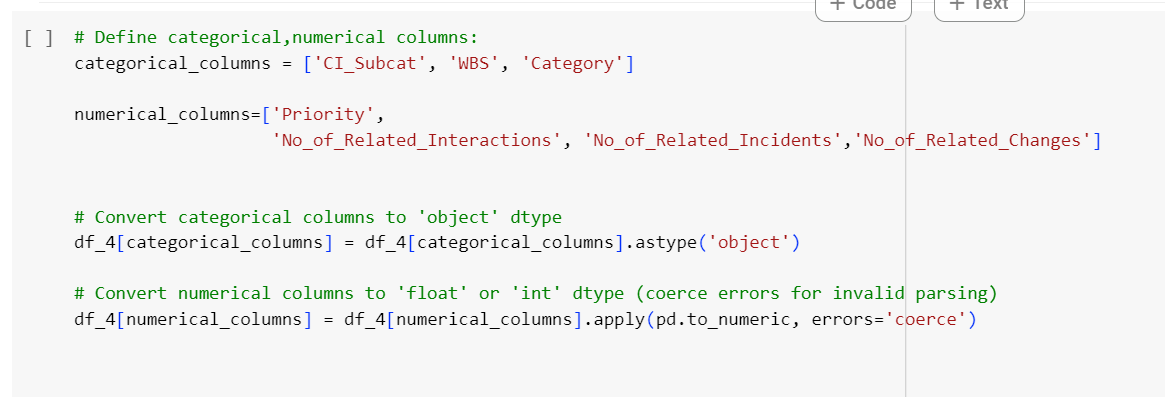
! Refer Page 36 for Metrics Definitions

**Task 4**

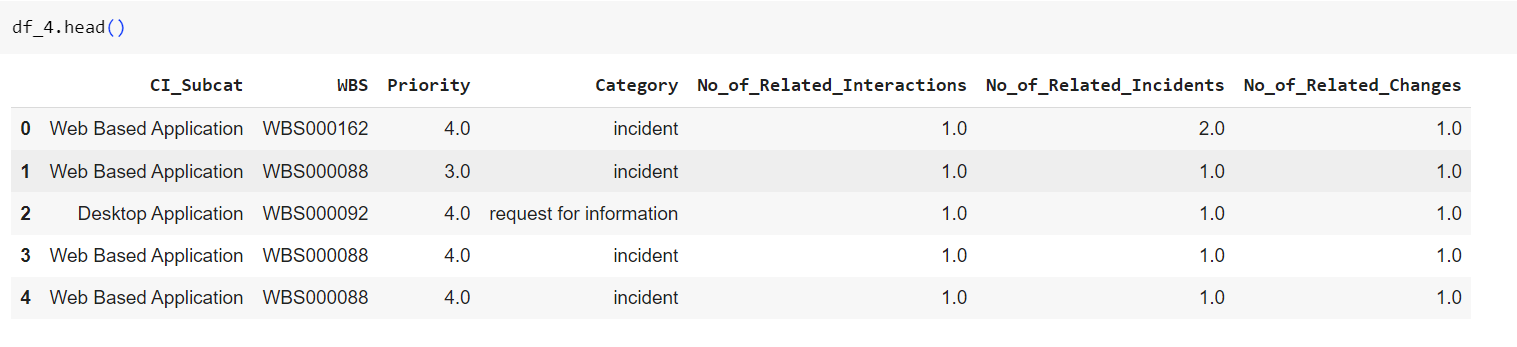
Predict Requests for Change (RFC) and potential failures or misconfigurations of ITSM (IT Service Management) assets.

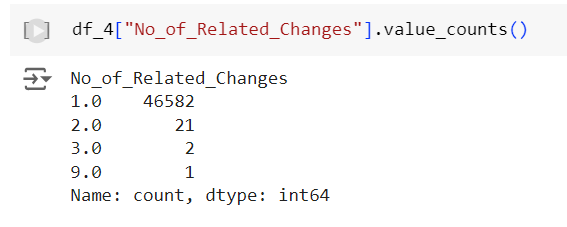
**Data Type Conversion**

1. Convert categorical columns to the 'object' data type.
2. Convert numerical columns to 'float' or 'int' data types, using coercion to handle errors for invalid parsing.



After removing replacing the Null Values with .median() of specific column

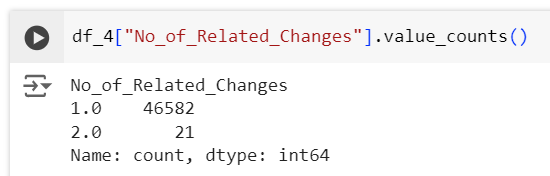




We observe that the column No\_of\_related\_changes contains values 3.0 and 9.0, each with only 2 and 1 occurrence, respectively.

df\_4.drop(df\_4.loc[df\_4['No\_of\_Related\_Changes']==3.0].index,inplace=True) # As there were 2 records in this category.

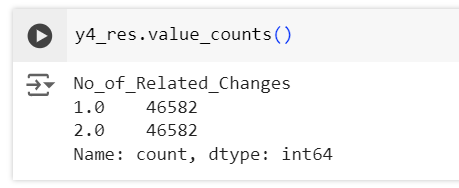
df\_4.drop(df\_4.loc[df\_4['No\_of\_Related\_Changes']==9.0].index,inplace=True) # As there was only 1 record in this category.



**Handling Imbalanced Data**

1. **Data Imbalance**: The data in our dependent feature No\_of\_Related\_Changes is highly imbalanced.
2. **Oversampling Solution**: To address this issue, we will use oversampling to balance the data.
3. **RandomOverSampler**: We will use the RandomOverSampler to handle the imbalance in the data.

This approach helps ensure that the classes in the dependent feature are balanced, improving the performance of the predictive model.

After balancing data : 

# Splitting the Data into train and test for calculating the accuracy

X4\_train, X4\_test, y4\_train, y4\_test = train\_test\_split(X4\_res,y4\_res,test\_size=0.3,random\_state=10)

# Standardization technique is used

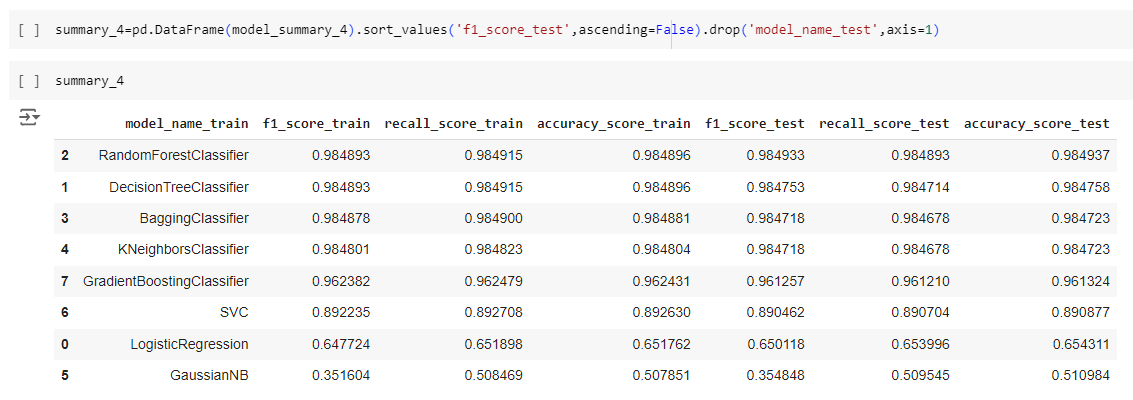
sc = StandardScaler()

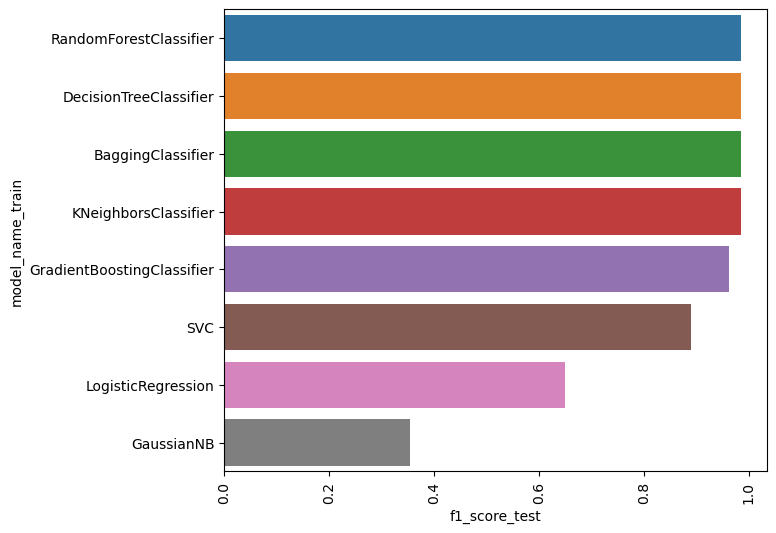
X4\_train = sc.fit\_transform(X4\_train)

X4\_test = sc.transform(X4\_test)

**Model Creation and Evaluation**



**Summary of Chosen Metrics for All Models**

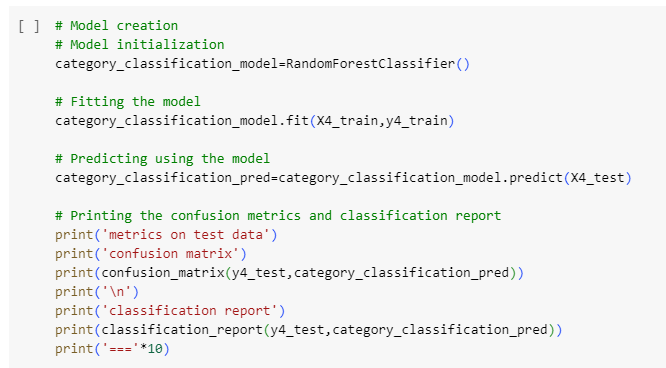
****

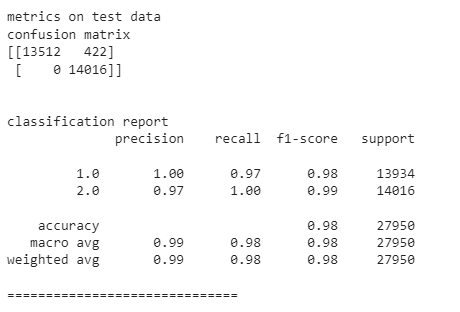
**Model Selection for Task 4**

From the analysis, it is observed that the RandomForestClassifier, BaggingClassifier, DecisionTreeClassifier, and KNeighborsClassifier models are performing well compared to other algorithms, achieving performance above 95%. Therefore, additional optimization techniques are not required.

We are selecting the RandomForestClassifier over the BaggingClassifier, DecisionTreeClassifier, and KNeighborsClassifier because it has consistently demonstrated superior performance.

We will proceed with developing the RandomForestClassifier model for further use.





**Files related to project**

1. [**PRCL-0012.pdf**](https://drive.google.com/open?id=1W54Zwl1rhIq27KT40Ra7jRZjFO8lroYz)
2. [**Project Code Notebook**](https://drive.google.com/file/d/1l1I4vU4qtpY8kQrdTgfsCYohhApsl91n/view?usp=sharing)

**Team Members Details**

## **PTID-CDS-MAR-24-1870** Team Info:

## 1. Dipanjali Patra

* (01-MAY-23-CDS-ONL-BUN-021-WDE20)
* email: [satyas.behera@gmail.com](mailto:satyas.behera@gmail.com) /[dipanjalipatra@gmail.com](mailto:dipanjalipatra@gmail.com)

## 2. Vinay C

* (09-Oct-23-CDS-BUN-021-WDE20-ONL)
* email: [vinayvinay9617@gmail.com](mailto:vinayvinay9617@gmail.com)

## 3. Sandeep Chandra Sagar R

* (09-Oct-23-CDS-BUN-021-WDE20-ONL)
* email: [sanwithdeep@gmail.com](mailto:sanwithdeep@gmail.com)

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***