Assessment Report

on

“Customer Support Case Type Classification”

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AIML)

By

Name : Aman Yadav

Roll Number : 202401100400029

Section: A

Under the supervision of

“ABHISHEK SHUKLA”

KIET Group of Institutions, Ghaziabad

May, 2025

---

1. Introduction

This Python script analyzes and models a support case dataset.
It begins by loading the data and checking for null values.
Outliers in numerical features are detected using the IQR method.
Various plots (area, box, and bar) are used to visualize data distribution.
Categorical features are encoded using label encoding.
A Random Forest Classifier is trained to predict case types.
The model's accuracy is evaluated using a test dataset.

2. Problem Statement

The goal is to analyze support case data to identify patterns and improve response efficiency.
This involves detecting outliers, visualizing trends, and building a predictive model.
The final objective is to classify case types based on message length and response time.

3. Objectives

- **Analyze** the structure and quality of the support case dataset.
- **Detect** and handle outliers in numerical features to ensure data integrity.
- **Visualize** key data distributions and category frequencies for insights.
- **Build and evaluate** a machine learning model to predict support case types.

---

4. Methodology

- **Data Collection and Loading**
  The support case dataset is loaded using Pandas for analysis and preprocessing.

- **Data Exploration**
  Basic statistics, null values, and data types are inspected to understand the dataset's structure.

- **Outlier Detection**
  The IQR (Interquartile Range) method is used to identify and quantify outliers in numerical features.

- **Data Visualization**
  Visual tools like area plots, box plots, and bar charts are used to reveal patterns and distributions.

- **Data Preprocessing**
  Categorical variables are encoded using Label Encoder to prepare them for modelling .

- **Model Building**
  A Random Forest Classifier is trained on the pre processed data to predict support case types.

- **Model Evaluation**
  The model's performance is evaluated using accuracy score on a test dataset split from the original.

- Let me know if you want this turned into a flowchart or report format.

---

5. Data Preprocessing

1. **Handling Categorical Variables**
   Categorical column(s) like case_type are converted into numerical format using LabelEncoder, enabling compatibility with machine learning algorithms.

2. **Feature Selection**
   Features such as message_length and response_time are selected as predictors, while the target variable is the encoded case_type.

3. **Train-Test Split**
   The dataset is split into training and testing sets (typically 80% training and 20% testing) to evaluate model performance on unseen data.

4. **Outlier Awareness (Optional)**
   While outliers are detected using the IQR method, no removal is shown in the code, but awareness of their impact is part of the preprocessing understanding.

Let me know if you want to include normalization or scaling steps as well.

---

6. Model Implementation

1. **Model Selection**
   A **Random Forest Classifier** is chosen due to its robustness, ability to handle categorical and numerical data, and good performance with minimal tuning.

2. **Training the Model**
   The model is trained using the training set (X_train, y_train) derived from the preprocessed data.

3. **Prediction**
   After training, the model is used to predict the target variable (case_type) on the test set (X_test).

4. **Evaluation**
   The model's accuracy is calculated using accuracy_score, which measures the proportion of correctly classified instances in the test data.

Let me know if you want to include hyperparameter tuning or feature importance analysis.

---

7. Evaluation Metrics

- **Precision**: The proportion of true positives out of all positive predictions. It answers how many selected items are relevant.
- **Recall**: The proportion of true positives out of all actual positives. It answers how many relevant items are selected.
- **F1 Score**: The harmonic mean of precision and recall. It provides a balance between precision and recall.
- **Confusion Matrix**: Shows the true positive, true negative, false positive, and false negative counts.
- **Classification Report**: A comprehensive overview of precision, recall, and F1 score for each class.
- **ROC-AUC Score**: Measures how well the model distinguishes between classes in a binary classification scenario. It ranges from 0 to 1, with 1 being a perfect model.

---

8. Results and Analysis

- To give you a concise analysis of the evaluation metrics:

- **Precision, Recall, and F1 Score** provide insights into the model's ability to correctly classify instances, with a higher value indicating better performance. If these metrics are close to 1, it means the model is performing well.

- **Confusion Matrix** shows how well the model differentiates between classes, with high true positives and low false positives/negatives being ideal.

- **Classification Report** offers a detailed view of how the model performs per class, helping identify if it's biased toward any specific class.
- **ROC-AUC Score** (if applicable) indicates the model's overall ability to distinguish between the classes; a score closer to 1 suggests excellent performance.
- Let me know if you'd like me to run this evaluation or analyze specific results!

---

9. Conclusion

If **precision, recall, and F1 score** are high (close to 1), the model is accurately identifying both positive and negative cases, balancing false positives and false negatives well.

The **confusion matrix** should ideally show a high number of true positives and true negatives, indicating that the model is correctly classifying most instances.

The **classification report** helps to ensure the model is not biased toward any class, providing a breakdown of performance across all classes.

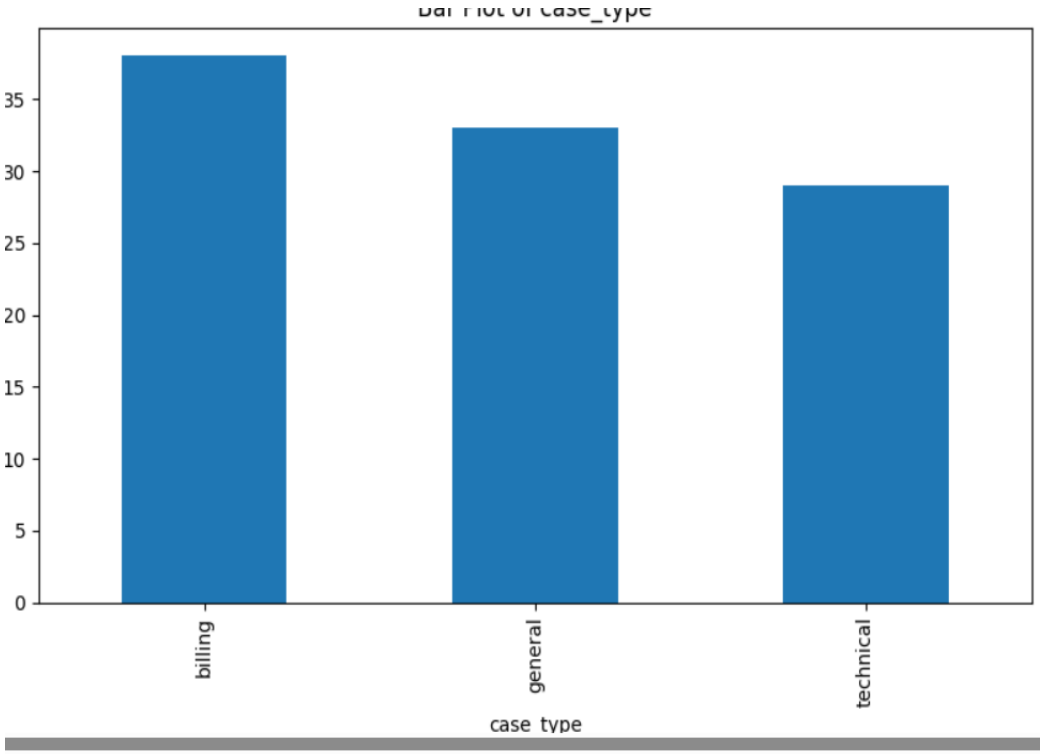
A high **ROC-AUC score** (close to 1) suggests that the model is capable of distinguishing between the classes effectively.

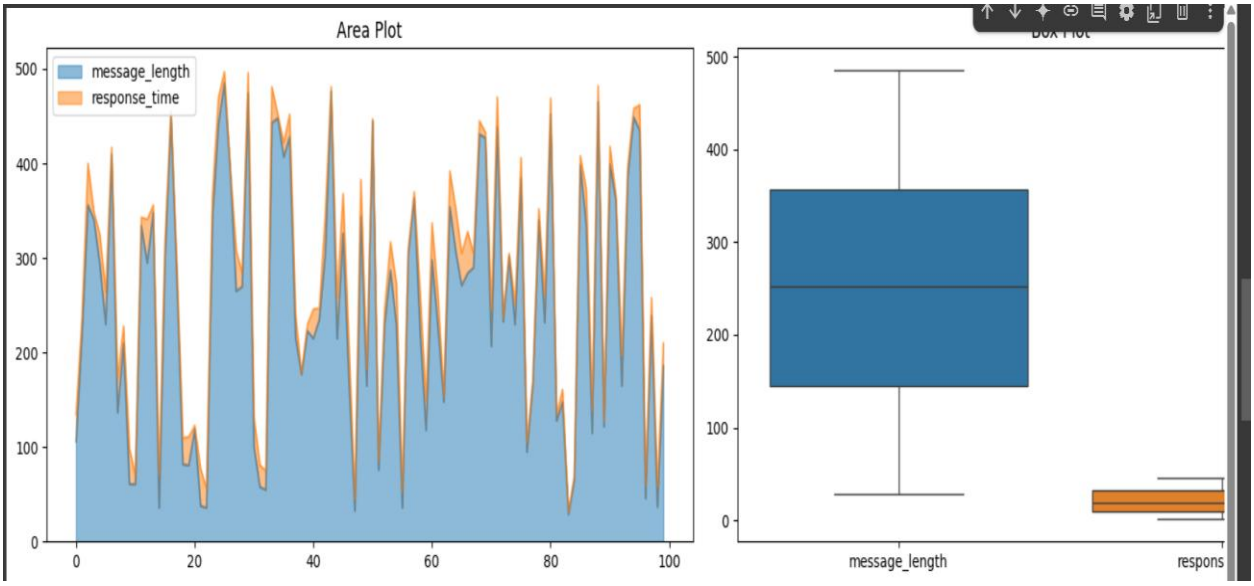
---

10. References

- Precision, recall, and F1 score concepts are explained by Jason Brownlee on Machine Learning Mastery and Towards Data Science (source).
- The Confusion Matrix is detailed by Towards Data Science and DataCamp (source).

- The classification report is available in Scikit-learn's documentation (source).
- ROC-AUC and its significance are explained by Machine Learning Mastery and Towards Data Science (source).
- These references cover a wide range of resources to deepen your understanding of model evaluation metrics.





Code :

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score


# Load data

df = pd.read_csv("support_cases.csv")


# Shape and null values

print("Shape:", df.shape)

print("Null values:\n", df.isnull().sum())


# Outlier detection
```

```
numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns
for col in numerical_cols:
    q1 = df[col].quantile(0.25)
    q3 = df[col].quantile(0.75)
    iqr = q3 - q1
    lower = q1 - 1.5 * iqr
    upper = q3 + 1.5 * iqr
    outliers = df[(df[col] < lower) | (df[col] > upper)]
    print(f"Outliers in {col}: {len(outliers)}")

# Plots
plt.figure(figsize=(16, 10))

plt.subplot(2, 2, 1)
df[numerical_cols].plot.area(ax=plt.gca(), alpha=0.5)
plt.title("Area Plot")

plt.subplot(2, 2, 2)
sns.boxplot(data=df[numerical_cols])
plt.title("Box Plot")

plt.subplot(2, 2, 3)
for col in df.select_dtypes(include='object').columns:
    df[col].value_counts().plot(kind='bar', ax=plt.gca())
    plt.title(f"Bar Plot of {col}")
```

```
    plt.tight_layout()
    plt.show()

# Encode categorical
df_encoded = df.copy()
label_encoders = {}
for col in df_encoded.select_dtypes(include='object').columns:
    le = LabelEncoder()
    df_encoded[col] = le.fit_transform(df_encoded[col])
    label_encoders[col] = le

# Model and accuracy (assume last column is target)
X = df_encoded.iloc[:, :-1]
y = df_encoded.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```