Project Title : Customer Sentiment Analysis and Satisfaction Prediction from Support Tickets

language : Machine learning, python, SQL, Excel

Tools : VS code, Jupyter notebook, Google Colab (Collaboratory

Dataset : Dataset is available in the given link. https://drive.google.com/file/d/1DRdLKOinSNuoMwVyFGH86f3xEhkZMrz6/view

Project Overview

The primary objective of this project is to predict customer satisfaction using the historical data contained within the Customer Support Ticket Dataset. This will be achieved by applying machine learning algorithms to analyze the various factors and features present in the ticket information that have an influence on how satisfied a customer is with the support they receive. The ultimate goal is to build a predictive model capable of estimating customer satisfaction levels based on the characteristics of their support interaction.

About Dataset

The Customer Support Ticket Dataset contains records of customer inquiries related to tech products, covering issues like hardware, software bugs, network problems, account access, and data loss. It includes details about the customer (name, email - anonymized domain, age, gender), the product purchased, the purchase date, and specifics of the support ticket (type, subject, description, status, resolution, priority, channel). Crucially, for closed tickets, it also provides the first response time, time to resolution, and the customer satisfaction rating (on a scale of 1 to 5). This dataset is suitable for analyzing support trends, applying Natural Language Processing to ticket text, predicting customer satisfaction and resolution times, segmenting customers, and building recommendation systems for support resources.

Features Description:

Ticket ID: A unique identifier for each individual support ticket.

Customer Name: The name of the customer who submitted the ticket.

Customer Email: The customer's email address (domain anonymized for privacy).

Customer Age: The age of the customer.

Customer Gender: The gender of the customer.

Product Purchased: The specific tech product the customer bought.

Date of Purchase: The date when the customer purchased the product.

Ticket Type: The broad category of the support request (e.g., technical issue).

Ticket Subject: A brief summary or topic of the customer's ticket.

Ticket Description: A more detailed explanation of the customer's problem or question.

Ticket Status: The current state of the ticket (e.g., open, closed).

Resolution: The solution or action taken to close the ticket.

Ticket Priority: The assigned level of urgency for the ticket.

Ticket Channel: How the customer contacted support (e.g., email, chat).

First Response Time: The duration until the initial response was sent to the customer.

Time to Resolution: The total time taken to resolve the ticket.

Customer Satisfaction Rating: The customer's feedback on their satisfaction with the resolution (1-5 scale).

Use Cases of such dataset:

Customer Support Analysis: This allows for data-driven improvements to support operations by understanding what issues are most frequent, how efficiently they are handled, and where processes can be optimized.

Natural Language Processing (NLP): Leveraging the text data (subject and description) can lead to automation and deeper understanding of customer needs and sentiments at scale.

Customer Satisfaction Prediction: Proactively identifying customers likely to be dissatisfied enables timely interventions and can improve overall customer loyalty.

Ticket Resolution Time Prediction: Accurate predictions can help manage customer expectations, allocate resources effectively, and potentially identify tickets that might require escalation.

Customer Segmentation: Understanding different customer groups based on their support interactions allows for tailored support strategies and potentially personalized product or service offerings.

Recommender Systems: Providing relevant solutions or product suggestions within the support process can improve efficiency and customer self-service capabilities.

Data Preprocessing:

This crucial step involves cleaning and preparing the raw data for modeling.

The sample code demonstrates:

Loading the dataset using pandas.

Displaying basic information about the data (data types, non-null values).

Handling missing values by dropping rows containing any NaN values.

Encoding categorical (object) columns into numerical representations using LabelEncoder.

Exploratory Data Analysis (EDA):

While not explicitly shown in the provided code snippet, EDA is a vital step that typically precedes feature engineering and model building. It involves visualizing data distributions, identifying patterns, and understanding relationships between variables using libraries like matplotlib and seaborn. This helps in gaining insights into the data and informing subsequent steps.

Feature Engineering:

This involves selecting, transforming, and creating new features from the existing data that might improve the performance of the machine learning model. The sample code shows a basic step of defining the feature set (X) by dropping 'CustomerID' and the target variable ('Overall Satisfaction'), and defining the target variable (y). More advanced feature engineering could involve creating interaction terms, polynomial features, or extracting information from date/time columns.

Model Building:

This step involves selecting a suitable machine learning algorithm and training it on the prepared data. The sample code uses a RandomForestClassifier, a popular ensemble method known for its good performance on various classification tasks. The data is split into training and testing sets to evaluate the model's ability to generalize to unseen data.

Model Evaluation:

After training, the model's performance is evaluated on the test set using appropriate metrics. The sample code demonstrates the use of:

accuracy_score:

The proportion of correctly classified instances. classification_report: Provides precision, recall, F1-score, and support for each class.
confusion_matrix:

A table that summarizes the model's predictions against the actual values, showing true positives, true negatives, false positives, and false negatives.

Visualization:

This step involves presenting the results and insights in a visual format for better understanding and communication. The sample code shows an example of visualizing feature importance from the trained Random Forest model, highlighting the top 10 features that contributed most to the predictions.

```
from google.colab import files
uploaded = files.upload()
```

Choose Files customer_s…t_tickets.csv

- **customer_support_tickets.csv**(text/csv) - 3945533 bytes, last modified: 5/7/2025 - 100% done

```
Saving customer_support_tickets.csv to customer_support_tickets (6).csv
```

```
import pandas as pd
import io

if uploaded:
    filename = list(uploaded.keys())[0]  # Get the first uploaded filename
    print(f"Uploaded filename: {filename}")
    try:
        df = pd.read_csv(io.BytesIO(uploaded[filename]))
        print("\nDataset loaded successfully!")
        print(df.head())  # Display the first few rows
```

```
        print(df.info())  # Display dataset information
    except Exception as e:
        print(f"\nError loading the dataset: {e}")
else:
    print("\nNo file was uploaded.")
```

```
0            Product setup
1  Peripheral compatibility
2          Network problem
3           Account access
4                Data loss

                              Ticket Description  \
0  I'm having an issue with the {product_purchase...
1  I'm having an issue with the {product_purchase...
2  I'm facing a problem with my {product_purchase...
3  I'm having an issue with the {product_purchase...
4  I'm having an issue with the {product_purchase...

              Ticket Status                                      Resolution  \
0  Pending Customer Response                                            NaN
1  Pending Customer Response                                            NaN
2                    Closed   Case maybe show recently my computer follow.
3                    Closed  Try capital clearly never color toward story.
4                    Closed                    West decision evidence bit.

  Ticket Priority Ticket Channel  First Response Time   Time to Resolution  \
0       Critical   Social media  2023-06-01 12:15:36                  NaN
1       Critical           Chat  2023-06-01 16:45:38                  NaN
2            Low   Social media  2023-06-01 11:14:38  2023-06-01 18:05:38
3            Low   Social media  2023-06-01 07:29:40  2023-06-01 01:57:40
4            Low          Email  2023-06-01 00:12:42  2023-06-01 19:53:42

   Customer Satisfaction Rating
0                           NaN
1                           NaN
2                           3.0
3                           3.0
4                           1.0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8469 entries, 0 to 8468
Data columns (total 17 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Ticket ID                     8469 non-null   int64
 1   Customer Name                 8469 non-null   object
 2   Customer Email                8469 non-null   object
 3   Customer Age                  8469 non-null   int64
 4   Customer Gender               8469 non-null   object
 5   Product Purchased             8469 non-null   object
 6   Date of Purchase              8469 non-null   object
 7   Ticket Type                   8469 non-null   object
 8   Ticket Subject                8469 non-null   object
 9   Ticket Description            8469 non-null   object
 10  Ticket Status                 8469 non-null   object
 11  Resolution                    2769 non-null   object
 12  Ticket Priority               8469 non-null   object
 13  Ticket Channel                8469 non-null   object
 14  First Response Time           5650 non-null   object
 15  Time to Resolution            2769 non-null   object
 16  Customer Satisfaction Rating  2769 non-null   float64
dtypes: float64(1), int64(2), object(14)
memory usage: 1.1+ MB
None
```

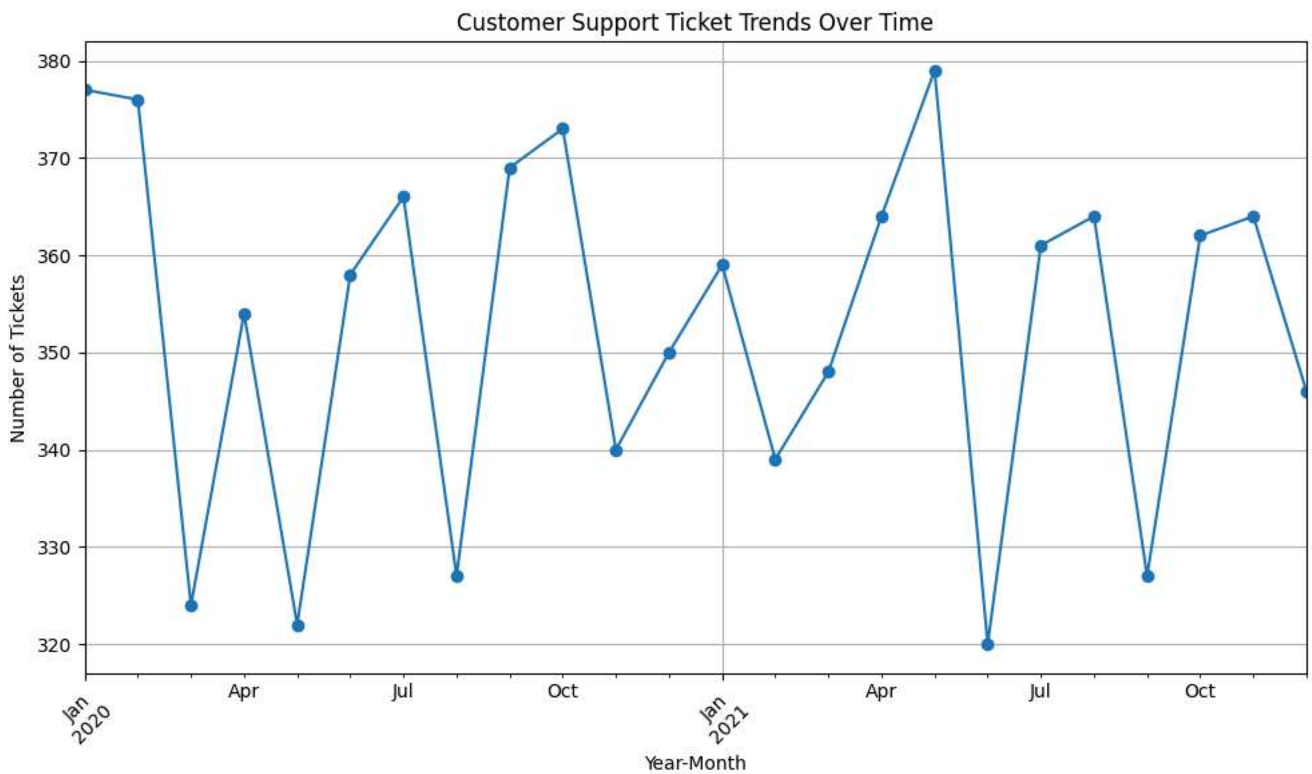
```
# Assuming your preprocessed DataFrame is named 'df_cleaned'
print(df_cleaned.columns)
```

```
Index(['Ticket ID', 'Customer Name', 'Customer Email', 'Customer Age',
       'Customer Gender', 'Product Purchased', 'Date of Purchase',
       'Ticket Type', 'Ticket Subject', 'Ticket Description', 'Ticket Status',
       'Resolution', 'Ticket Priority', 'Ticket Channel',
       'First Response Time', 'Time to Resolution',
       'Customer Satisfaction Rating'],
      dtype='object')
```

```
# Analyze customer support ticket trends
# Identify common issues
common_issues = df['Ticket Subject'].value_counts().head(10)
print("Top 10 Common Issues:")
print(common_issues)
```

```
# Plotting ticket trends over time
df['Date of Purchase'] = pd.to_datetime(df['Date of Purchase'])
df['YearMonth'] = df['Date of Purchase'].dt.to_period('M')
ticket_trends = df.groupby('YearMonth').size()
plt.figure(figsize=(10, 6))
ticket_trends.plot(kind='line', marker='o')
plt.title('Customer Support Ticket Trends Over Time')
plt.xlabel('Year-Month')
plt.ylabel('Number of Tickets')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
Top 10 Common Issues:
Ticket Subject
Refund request          576
Software bug            574
Product compatibility   567
Delivery problem        561
Hardware issue          547
Battery life            542
Network problem         539
Installation support    530
Product setup           529
Payment issue           526
Name: count, dtype: int64
```



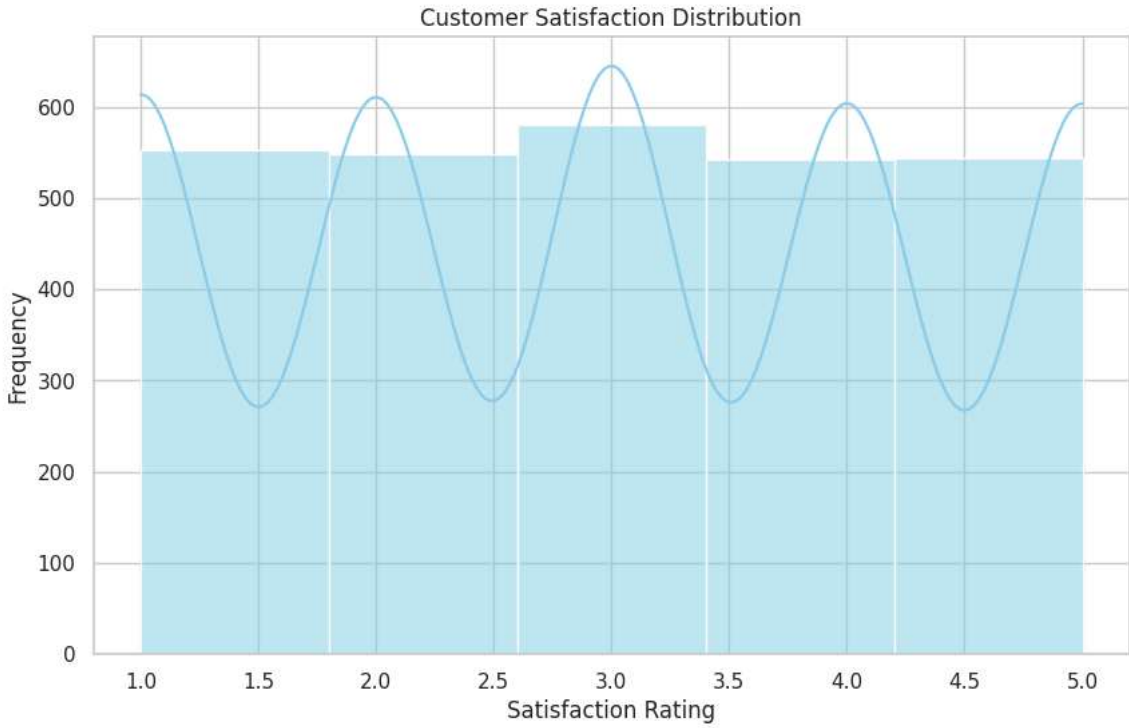

```
# Assuming your loaded DataFrame is named 'df'

# Define the target variable (Customer Satisfaction Rating)
y = df['Customer Satisfaction Rating']

# Define the features (X) - select the columns you want to use for prediction
# You'll need to choose these based on your understanding of the data and EDA.
# Here's an example using some potentially relevant columns:
features = ['Customer Age', 'Product Purchased', 'Ticket Type', 'Ticket Priority', 'Ticket Channel', 'Ticket Subject', 'Ticket Description']
X = df[features]

# If you've already done some preprocessing and your DataFrame is 'df_cleaned', use that instead:
# y = df_cleaned['Customer Satisfaction Rating']
# features = [...]
# X = df_cleaned[features]

print("Features (X) shape:", X.shape)
print("Target (y) shape:", y.shape)
```

```
Features (X) shape: (8469, 7)
Target (y) shape: (8469,)
```

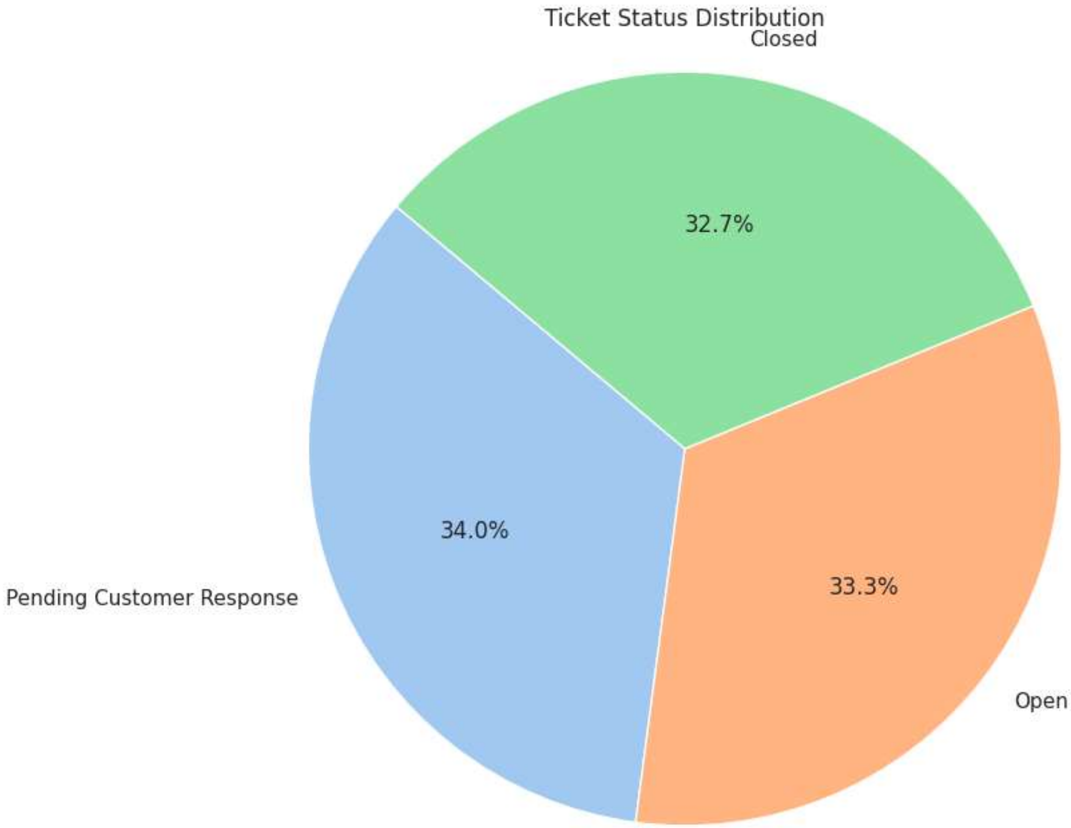
```
# Assuming your loaded DataFrame is named 'df'

# Segment based on ticket types
ticket_type_segmentation = df.groupby('Ticket Type').size()
print("\nSegmentation based on Ticket Types:")
print(ticket_type_segmentation)

# Segment based on satisfaction levels
satisfaction_segmentation = df.groupby('Customer Satisfaction Rating').size()
print("\nSegmentation based on Customer Satisfaction Levels:")
print(satisfaction_segmentation)
```

```

Segmentation based on Ticket Types:
Ticket Type
Billing inquiry         1634
Cancellation request    1695
Product inquiry         1641
Refund request          1752
Technical issue         1747
dtype: int64

Segmentation based on Customer Satisfaction Levels:
Customer Satisfaction Rating
1.0    553
2.0    549
3.0    580
4.0    543
5.0    544
dtype: int64
```

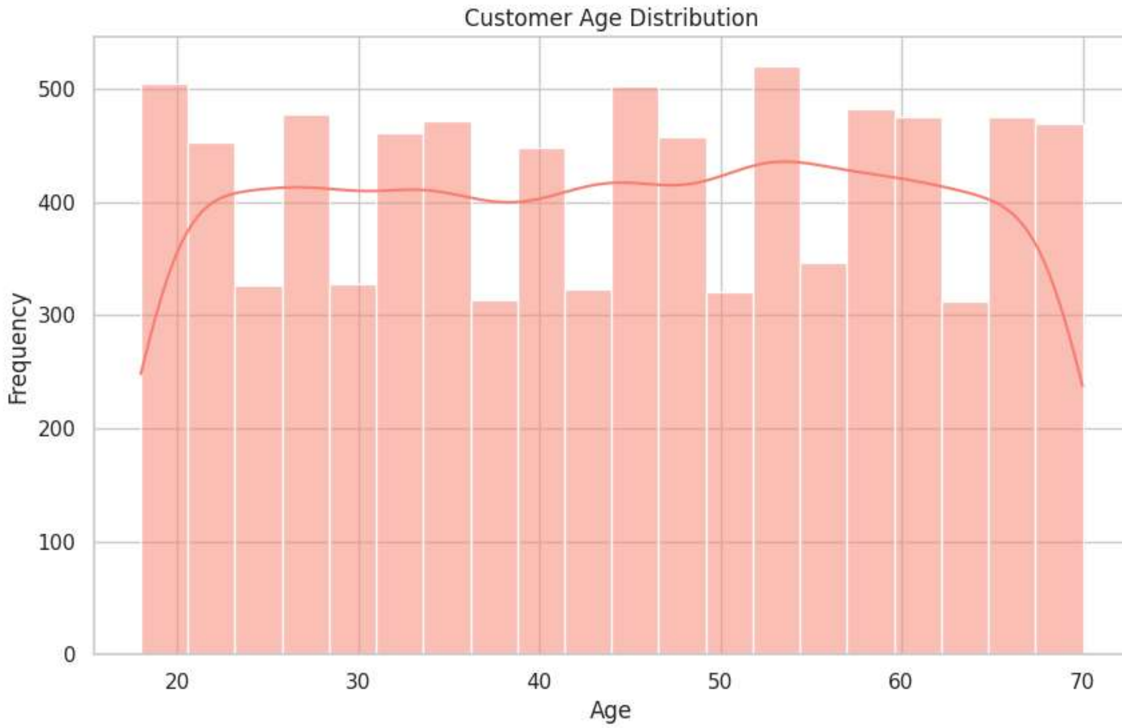
```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming your loaded DataFrame is named 'df'

# Set up the plotting aesthetics
sns.set(style="whitegrid")

# Customer Satisfaction Distribution
plt.figure(figsize=(10, 6))
sns.histplot(df['Customer Satisfaction Rating'].dropna(), bins=5, kde=True, color='skyblue')
plt.title('Customer Satisfaction Distribution')
plt.xlabel('Satisfaction Rating')
plt.ylabel('Frequency')
plt.show()
```



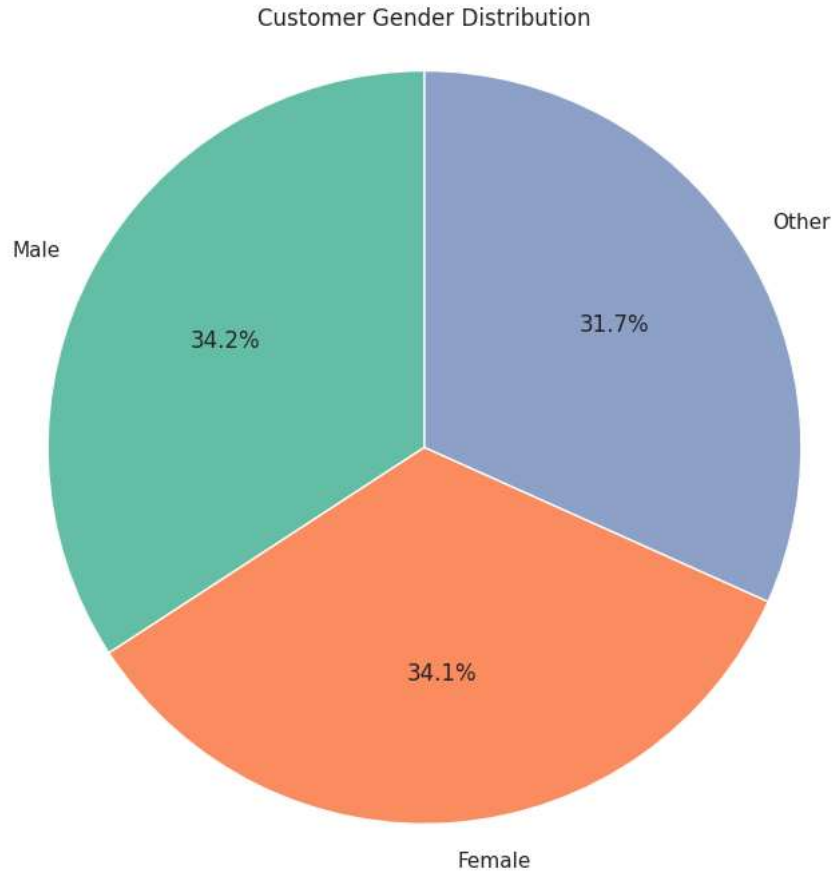

```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming your loaded DataFrame is named 'df'

# Ticket Status Distribution
ticket_status_distribution = df['Ticket Status'].value_counts()
plt.figure(figsize=(8, 8))
plt.pie(ticket_status_distribution,
        labels=ticket_status_distribution.index,
        autopct='%1.1f%%',
        colors=sns.color_palette('pastel'),
        startangle=140)
plt.title('Ticket Status Distribution')
plt.axis('equal')
plt.show()
```




```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming your loaded DataFrame is named 'df'

# Customer Age Distribution
plt.figure(figsize=(10, 6))
sns.histplot(df['Customer Age'].dropna(), bins=20, kde=True, color='salmon')
plt.title('Customer Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```

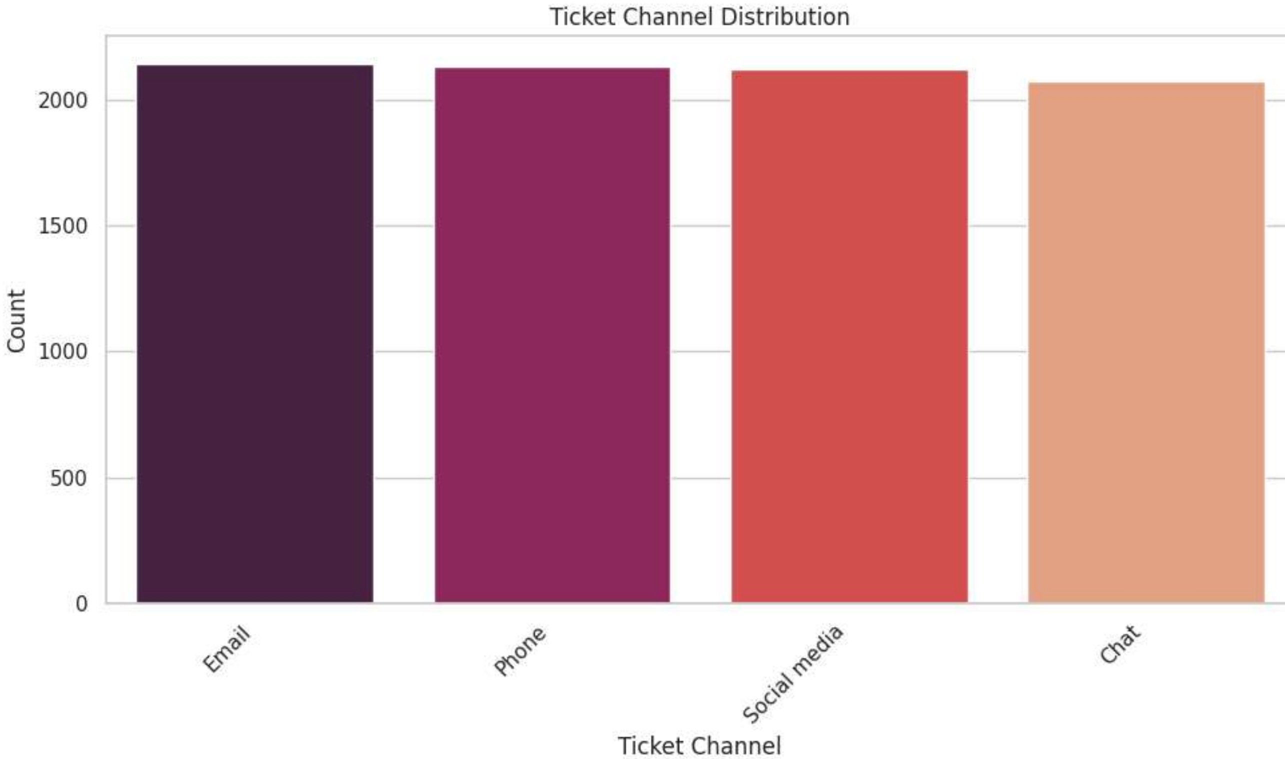



```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming your loaded DataFrame is named 'df'

# Customer Gender Distribution
customer_gender_distribution = df['Customer Gender'].value_counts()
plt.figure(figsize=(8, 8))
plt.pie(customer_gender_distribution,
        labels=customer_gender_distribution.index,
        autopct='%1.1f%%',
        colors=sns.color_palette('Set2'),
        startangle=90)
plt.title('Customer Gender Distribution')
plt.axis('equal')
plt.show()
```



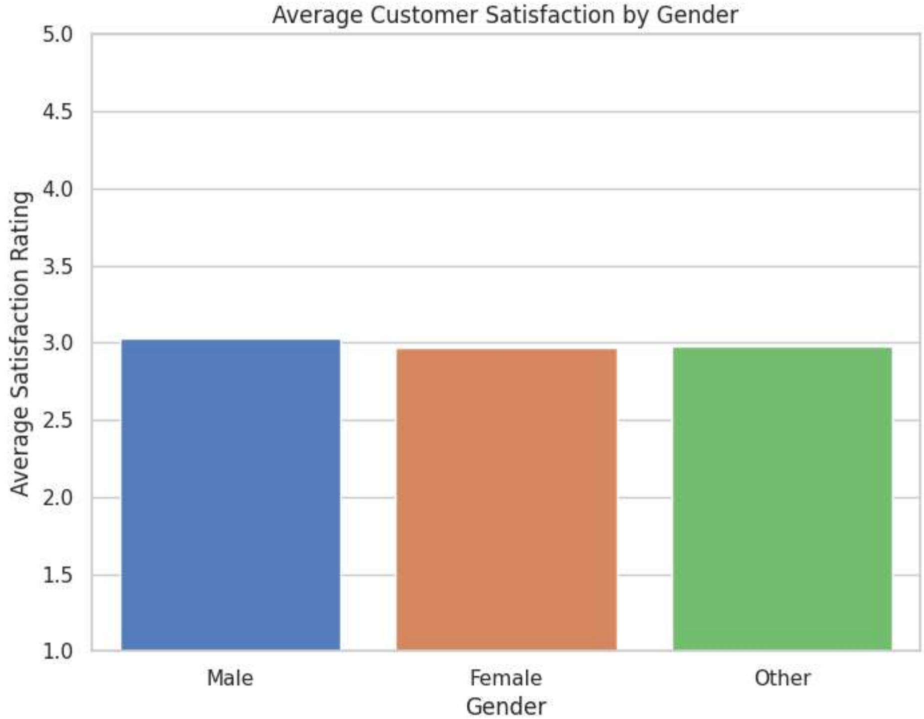

```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming your loaded DataFrame is named 'df'

# Ticket Channel Distribution
ticket_channel_distribution = df['Ticket Channel'].value_counts()
plt.figure(figsize=(10, 6))
sns.barplot(x=ticket_channel_distribution.index,
            y=ticket_channel_distribution,
            palette='rocket')
plt.title('Ticket Channel Distribution')
plt.xlabel('Ticket Channel')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right') # Added ha='right' for better label alignment
plt.tight_layout() # Added to prevent labels from overlapping
plt.show()
```

```
<ipython-input-37-bc305f982300>:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend

  sns.barplot(x=ticket_channel_distribution.index,
```



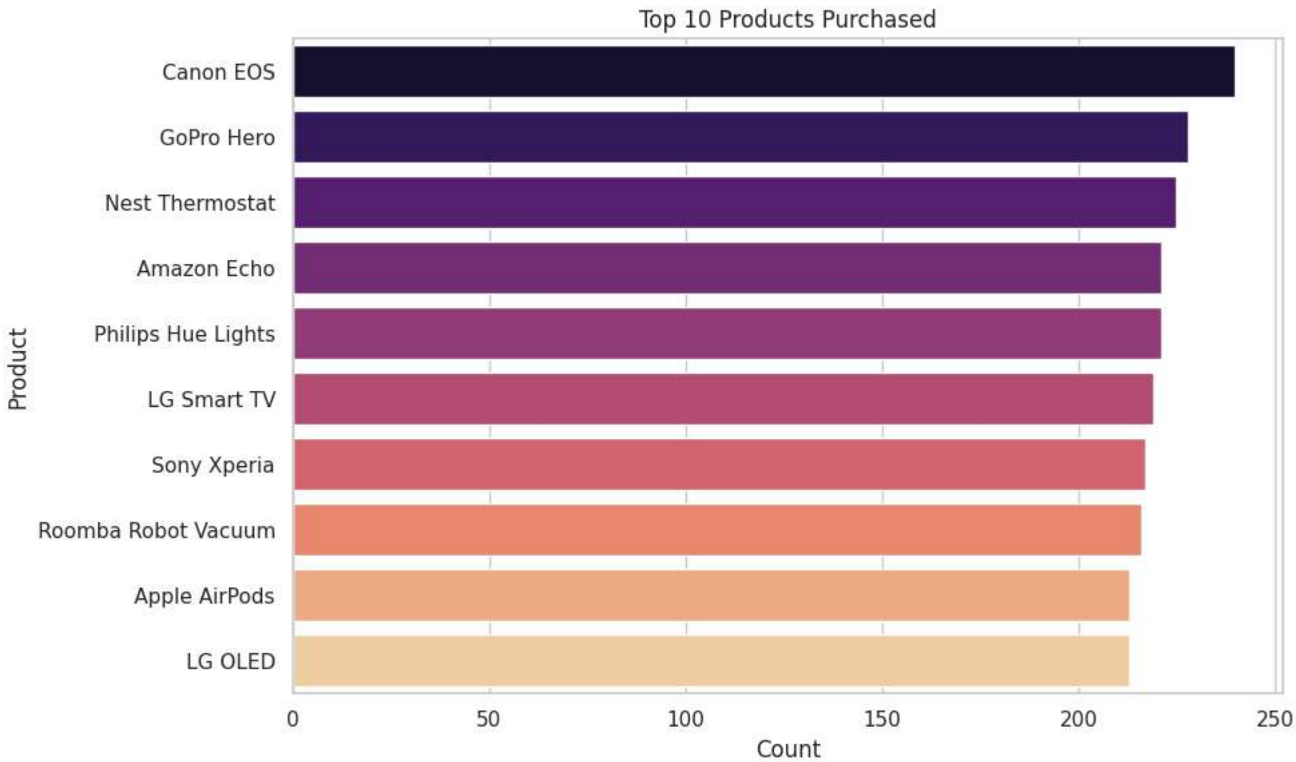

```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming your loaded DataFrame is named 'df'

# Chart 1: Average Customer Satisfaction by Gender (Bar Plot)
average_satisfaction = df.groupby('Customer Gender')['Customer Satisfaction Rating'].mean().reset_index()
plt.figure(figsize=(8, 6))
sns.barplot(x='Customer Gender', y='Customer Satisfaction Rating', data=average_satisfaction, palette='muted', order=['Male', 'Female', 'Oth
plt.title('Average Customer Satisfaction by Gender')
plt.xlabel('Gender')
plt.ylabel('Average Satisfaction Rating')
plt.ylim(1, 5) # Adjust y-axis limit if needed
plt.show()
```

```
<ipython-input-38-8fa93097f6b1>:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend

  sns.barplot(x='Customer Gender', y='Customer Satisfaction Rating', data=average_satisfaction, palette='muted', order=['Male', 'Female'
```

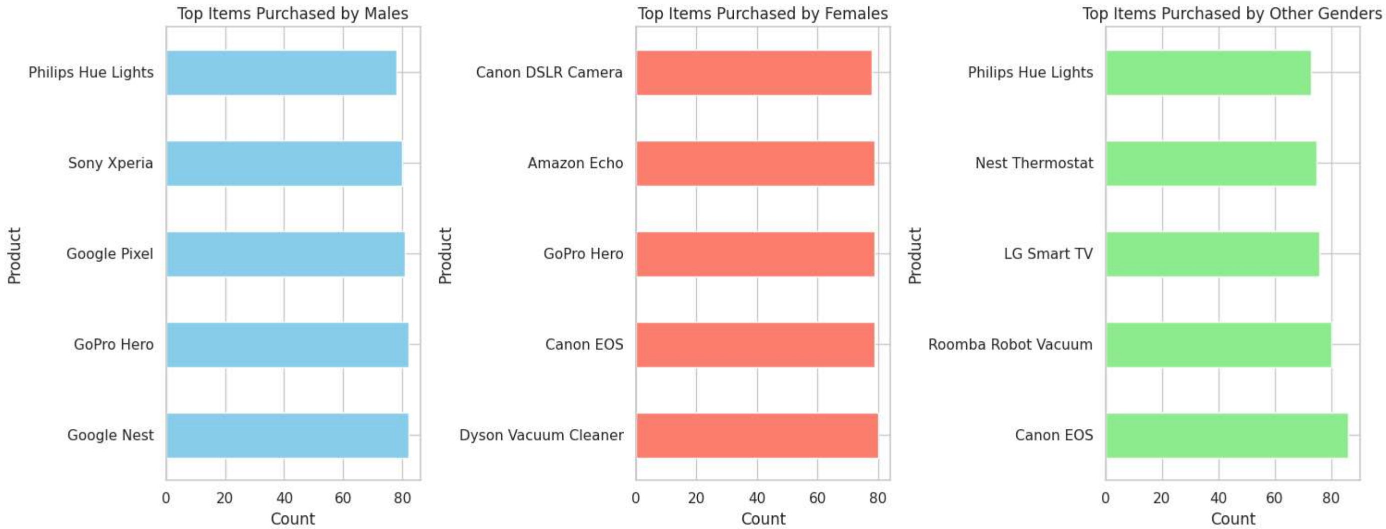



```
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming your loaded DataFrame is named 'df'

# Product Purchased Distribution
plt.figure(figsize=(10, 6))
product_purchased_distribution = df['Product Purchased'].value_counts().head(10)
sns.barplot(y=product_purchased_distribution.index,
            x=product_purchased_distribution,
            palette='magma')
plt.title('Top 10 Products Purchased')
plt.xlabel('Count')
plt.ylabel('Product')
plt.tight_layout() # Added to prevent labels from overlapping
plt.show()
```

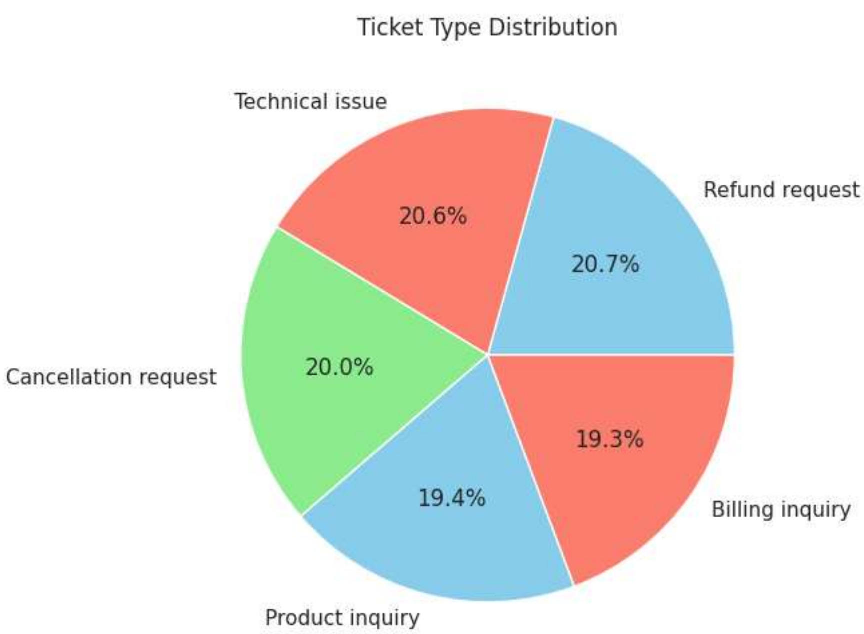
```
<ipython-input-39-134083decd9a>:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend

  sns.barplot(y=product_purchased_distribution.index,
```




```
import matplotlib.pyplot as plt

# Assuming your loaded DataFrame is named 'df'

# Chart 2: Top Items Purchased by Gender (Horizontal Bar Chart)
plt.figure(figsize=(15, 6))

# Top Items Purchased by Males
plt.subplot(1, 3, 1)
top_items_male = df[df['Customer Gender'] == 'Male']['Product Purchased'].value_counts().head(5)
top_items_male.plot(kind='barh', color='skyblue')
plt.title('Top Items Purchased by Males')
plt.xlabel('Count')
plt.ylabel('Product')

# Top Items Purchased by Females
plt.subplot(1, 3, 2)
top_items_female = df[df['Customer Gender'] == 'Female']['Product Purchased'].value_counts().head(5)
top_items_female.plot(kind='barh', color='salmon')
plt.title('Top Items Purchased by Females')
plt.xlabel('Count')
plt.ylabel('Product')

# Top Items Purchased by Other Gender
plt.subplot(1, 3, 3)
top_items_other = df[df['Customer Gender'] == 'Other']['Product Purchased'].value_counts().head(5)
top_items_other.plot(kind='barh', color='lightgreen')
plt.title('Top Items Purchased by Other Genders')
plt.xlabel('Count')
plt.ylabel('Product')

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




```
import matplotlib.pyplot as plt

# Assuming your loaded DataFrame is named 'df'

# Count ticket types
ticket_type_distribution = df['Ticket Type'].value_counts()

# Plot
plt.figure(figsize=(8, 6))
ticket_type_distribution.plot(kind='pie', autopct='%1.1f%%',
                              colors=['skyblue', 'salmon', 'lightgreen'])
plt.title('Ticket Type Distribution')
plt.ylabel('')  # Remove the default 'Ticket Type' label on the y-axis
plt.show()
```




```
import matplotlib.pyplot as plt

# Assuming your loaded DataFrame is named 'df'

# Count ticket priorities
priority_distribution = df['Ticket Priority'].value_counts()
```

```
# Plot
plt.figure(figsize=(8, 6))
priority_distribution.plot(kind='pie', autopct='%1.1f%%',
                            colors=['lightblue', 'lightgreen', 'lightsalmon', 'skyblue']
```