__Project title__ - **Flipkart Project**
Project Type - EDA/Regression/Classification/Unsupervised

Contribution - **Individul**
Name - **Subham Kundu**

__Project Summary -__
This project focuses on analyzing customer support data to understand the key factors influencing customer satisfaction (CSAT) and overall service performance. The primary objective was to identify patterns in customer complaints and examine how different variables such as complaint category, sub-category, support channel, agent tenure, and customer location impact satisfaction levels. The dataset was cleaned and preprocessed using Python to handle inconsistencies, remove unnecessary spaces in column names, and manage missing values before conducting the analysis.

Exploratory Data Analysis was performed using Pandas and Matplotlib to uncover trends and relationships within the data. Various visualizations, including bar charts, histograms, and density plots, were used to examine complaint distribution and CSAT score patterns. The analysis revealed that certain complaint categories, particularly order-related and return-related issues, contributed significantly to overall complaint volume and had a noticeable effect on customer satisfaction scores. Additionally, comparisons across support channels and agent experience levels provided insights into operational efficiency and service quality.

The project demonstrates practical skills in data cleaning, statistical reasoning, data visualization, and business insight generation. By identifying major drivers of dissatisfaction and highlighting areas that require operational improvement, the analysis provides actionable recommendations to enhance customer experience and optimize support performance. Overall, this project reflects the application of data analytics techniques to solve real-world business problems and support data-driven decision-making.

__Problem Statement -__
Customer support teams handle large volumes of complaints, but organizations often lack insight into the factors affecting customer satisfaction. This project aims to analyze complaint and service data to identify key drivers of CSAT and develop machine learning models to predict customer satisfaction levels, enabling proactive decision-making and service improvement.

__General Guidelines -__

1. Problem Understanding

Clearly define the business problem you are trying to solve and explain why it is important. The problem statement should connect technical analysis to business value,

such as improving customer satisfaction, reducing churn, or optimizing operational efficiency. A well-defined problem ensures that the project remains focused and outcome-driven.

2. Data Understanding

Understand the dataset thoroughly before building any model. Identify the meaning of each feature, check data types, and study the distribution of variables. Analyze missing values, duplicates, and possible inconsistencies. Connecting each feature to its business relevance strengthens the foundation of your analysis.

3. Data Preprocessing

Prepare the data carefully by handling missing values, removing duplicates, cleaning column names, and encoding categorical variables where necessary. For machine learning models, ensure the data is structured properly and split into training and testing sets. Proper preprocessing directly impacts model accuracy and reliability.

4. Exploratory Data Analysis (EDA)

Perform exploratory analysis to identify trends, patterns, and relationships between variables. Use visualizations to understand distributions, category comparisons, and correlations. EDA helps uncover insights that guide feature selection and model development while providing valuable business interpretations.

5. Model Selection

Choose a model appropriate for the problem type, whether regression or classification. Start with a simple baseline model and then move to more advanced models if required. Justify your choice based on dataset characteristics and the problem objective.

6. Model Training and Evaluation

Train the model using the training dataset and evaluate it using appropriate performance metrics. For regression problems, use R^2, MAE, and MSE. For classification problems, use Accuracy, Precision, Recall, and F1-score. Always interpret these metrics in terms of business impact rather than just numerical performance.

7. Hyperparameter Tuning

If applicable, improve model performance using techniques such as Grid Search or Randomized Search with cross-validation. Tuning helps optimize model parameters and improves generalization, reducing the risk of overfitting.

8. Business Interpretation

Translate technical findings into business insights. Explain how the results can improve decision-making, enhance customer experience, reduce costs, or increase revenue. This step transforms a technical project into a business-driven solution.

In [127…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [129…
```
df = pd.read_csv("C:\\Users\\kundu\\Downloads\\Customer_support_data.csv")
```

In [131…
```
df.head()
```

Out[131…

	Unique id	channel_name	category	Sub-category	Customer Remarks	Order_id
0	7e9ae164-6a8b-4521-a2d4-58f7c9fff13f	Outcall	Product Queries	Life Insurance	NaN	c27c9bb4-fa36-4140-9f1f-21009254ffdb
1	b07ec1b0-f376-43b6-86df-ec03da3b2e16	Outcall	Product Queries	Product Specific Information	NaN	d406b0c7-ce17-4654-b9de-f08d421254bd
2	200814dd-27c7-4149-ba2b-bd3af3092880	Inbound	Order Related	Installation/demo	NaN	c273368d-b961-44cb-beaf-62d6fd6c00d5
3	eb0d3e53-c1ca-42d3-8486-e42c8d622135	Inbound	Returns	Reverse Pickup Enquiry	NaN	5aed0059-55a4-4ec6-bb54-97942092020a
4	ba903143-1e54-406c-b969-46c52f92e5df	Inbound	Cancellation	Not Needed	NaN	e8bed5a9-6933-4aff-9dc6-ccefd7dcde59

In [133…
```
print(df.head())
```

```
                              Unique id channel_name         category  \
0  7e9ae164-6a8b-4521-a2d4-58f7c9fff13f      Outcall  Product Queries
1  b07ec1b0-f376-43b6-86df-ec03da3b2e16      Outcall  Product Queries
2  200814dd-27c7-4149-ba2b-bd3af3092880      Inbound    Order Related
3  eb0d3e53-c1ca-42d3-8486-e42c8d622135      Inbound          Returns
4  ba903143-1e54-406c-b969-46c52f92e5df      Inbound     Cancellation

                  Sub-category Customer Remarks  \
0               Life Insurance              NaN
1  Product Specific Information              NaN
2            Installation/demo              NaN
3       Reverse Pickup Enquiry              NaN
4                   Not Needed              NaN

                               Order_id order_date_time Issue_reported at  \
0  c27c9bb4-fa36-4140-9f1f-21009254ffdb             NaN  01/08/2023 11:13
1  d406b0c7-ce17-4654-b9de-f08d421254bd             NaN  01/08/2023 12:52
2  c273368d-b961-44cb-beaf-62d6fd6c00d5             NaN  01/08/2023 20:16
3  5aed0059-55a4-4ec6-bb54-97942092020a             NaN  01/08/2023 20:56
4  e8bed5a9-6933-4aff-9dc6-ccefd7dcde59             NaN  01/08/2023 10:30

    issue_responded Survey_response_Date Customer_City Product_category  \
0  01/08/2023 11:47           01-Aug-23           NaN              NaN
1  01/08/2023 12:54           01-Aug-23           NaN              NaN
2  01/08/2023 20:38           01-Aug-23           NaN              NaN
3  01/08/2023 21:16           01-Aug-23           NaN              NaN
4  01/08/2023 10:32           01-Aug-23           NaN              NaN

   Item_price  connected_handling_time           Agent_name      Supervisor  \
0         NaN                      NaN     Richard Buchanan     Mason Gupta
1         NaN                      NaN        Vicki Collins       Dylan Kim
2         NaN                      NaN         Duane Norman    Jackson Park
3         NaN                      NaN       Patrick Flores     Olivia Wang
4         NaN                      NaN  Christopher Sanchez  Austin Johnson

          Manager    Tenure Bucket Agent Shift  CSAT Score
0  Jennifer Nguyen  On Job Training     Morning           5
1      Michael Lee              >90     Morning           5
2      William Kim  On Job Training     Evening           5
3       John Smith              >90     Evening           5
4      Michael Lee             0-30     Morning           5
```

In [135…

```
# NUMBER OF ROWS AND COLUMNS
print("Rows:", df.shape[0])
print("Columns:", df.shape[1])
```

```
Rows: 85907
Columns: 20
```

In [137…

```
# COLUMN NAMES
print(df.columns.tolist())
```

```
['Unique id', 'channel_name', 'category', 'Sub-category', 'Customer Remarks', 'Or
der_id', 'order_date_time', 'Issue_reported at', 'issue_responded', 'Survey_respo
nse_Date', 'Customer_City', 'Product_category', 'Item_price', 'connected_handling
_time', 'Agent_name', 'Supervisor', 'Manager', 'Tenure Bucket', 'Agent Shift', 'C
SAT Score']
```

In [139…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 85907 entries, 0 to 85906
Data columns (total 20 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   Unique id                85907 non-null  object 
 1   channel_name             85907 non-null  object 
 2   category                 85907 non-null  object 
 3   Sub-category             85907 non-null  object 
 4   Customer Remarks         28742 non-null  object 
 5   Order_id                 67675 non-null  object 
 6   order_date_time          17214 non-null  object 
 7   Issue_reported at        85907 non-null  object 
 8   issue_responded          85907 non-null  object 
 9   Survey_response_Date     85907 non-null  object 
 10  Customer_City            17079 non-null  object 
 11  Product_category         17196 non-null  object 
 12  Item_price               17206 non-null  float64
 13  connected_handling_time  242 non-null    float64
 14  Agent_name               85907 non-null  object 
 15  Supervisor               85907 non-null  object 
 16  Manager                  85907 non-null  object 
 17  Tenure Bucket            85907 non-null  object 
 18  Agent Shift              85907 non-null  object 
 19  CSAT Score               85907 non-null  int64  
dtypes: float64(2), int64(1), object(17)
memory usage: 13.1+ MB
```

In [141…

```python
# DUPLICATE VALUES COUNT
duplicate_count = df.duplicated().sum()
print("Total Duplicate Rows:", duplicate_count)

# SHOW DUPLICATE ROWS
duplicates = df[df.duplicated()]
print("\nDuplicate Rows Preview:")
print(duplicates.head())

# CHECK DUPLICATES BASED ON SPECIFIC COLUMNS
duplicate_unique_id = df.duplicated(subset=["Unique id"]).sum()
print("\nDuplicate based on Unique id:", duplicate_unique_id)
```

```
Total Duplicate Rows: 0

Duplicate Rows Preview:
Empty DataFrame
Columns: [Unique id, channel_name, category, Sub-category, Customer Remarks, Orde
r_id, order_date_time, Issue_reported at, issue_responded, Survey_response_Date,
Customer_City, Product_category, Item_price, connected_handling_time, Agent_name,
Supervisor, Manager, Tenure Bucket, Agent Shift, CSAT Score]
Index: []

Duplicate based on Unique id: 0
```

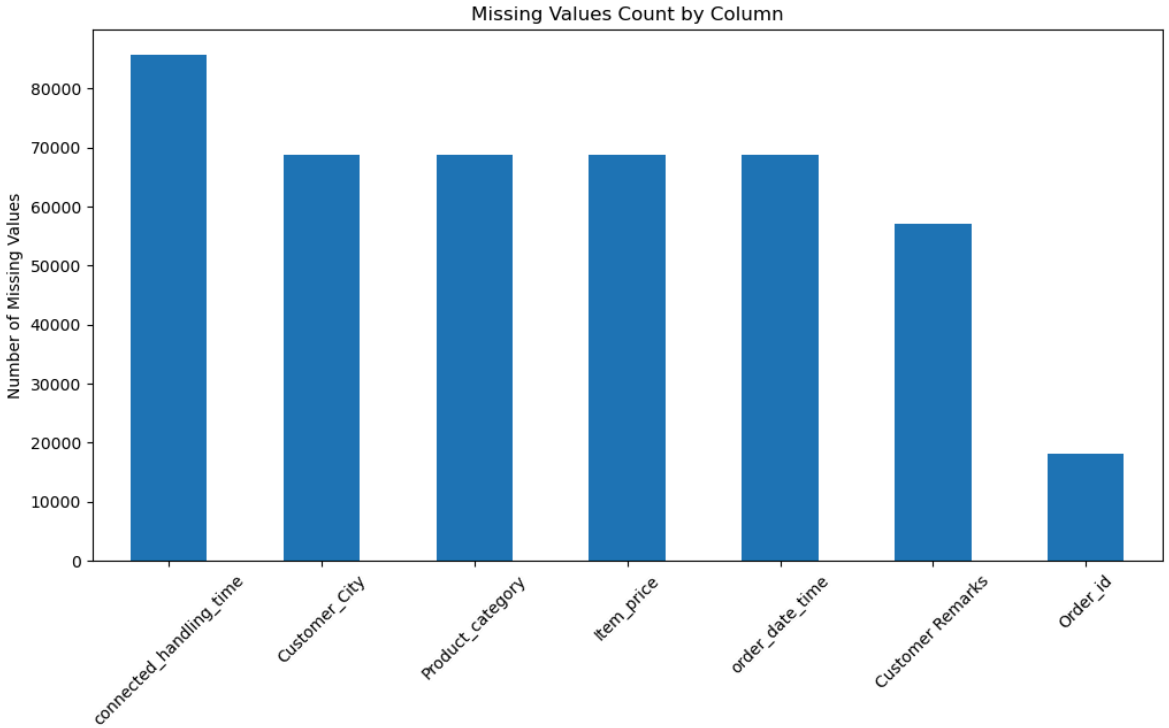
In [143…

```python
# MISSING VALUES COUNT :
print(df.isnull().sum())
```

```
Unique id                      0
channel_name                   0
category                       0
Sub-category                   0
Customer Remarks           57165
Order_id                   18232
order_date_time            68693
Issue_reported at              0
issue_responded                0
Survey_response_Date           0
Customer_City              68828
Product_category           68711
Item_price                 68701
connected_handling_time    85665
Agent_name                     0
Supervisor                     0
Manager                        0
Tenure Bucket                  0
Agent Shift                    0
CSAT Score                     0
dtype: int64
```

In [145…

```
# VISUALIZING MISSING VALUES :

# CALCULATE MISSING VALUES :
missing = df.isnull().sum()
missing = missing[missing > 0].sort_values(ascending=False)


plt.figure(figsize=(12,6))
missing.plot(kind='bar')

plt.title("Missing Values Count by Column")
plt.ylabel("Number of Missing Values")
plt.xticks(rotation=45)
plt.show()
```



In [147…
```
# COLUMN NAMES :
print("Dataset Columns:\n")
print(df.columns)
```

```
Dataset Columns:

Index(['Unique id', 'channel_name', 'category', 'Sub-category',
       'Customer Remarks', 'Order_id', 'order_date_time', 'Issue_reported at',
       'issue_responded', 'Survey_response_Date', 'Customer_City',
       'Product_category', 'Item_price', 'connected_handling_time',
       'Agent_name', 'Supervisor', 'Manager', 'Tenure Bucket', 'Agent Shift',
       'CSAT Score'],
      dtype='object')
```

In [149…
```
# DESCRIBE DATASET
print("Numerical Columns Summary:\n")
print(df.describe())
```

```
Numerical Columns Summary:

          Item_price  connected_handling_time    CSAT Score
count   17206.000000               242.000000  85907.000000
mean     5660.774846               462.400826      4.242157
std     12825.728411               246.295037      1.378903
min         0.000000                 0.000000      1.000000
25%       392.000000               293.000000      4.000000
50%       979.000000               427.000000      5.000000
75%      2699.750000               592.250000      5.000000
max    164999.000000              1986.000000      5.000000
```

VARIABLE DESCRIPTION :

1> *__Unique id__* - Used to uniquely distinguish every customer interaction record.
2> *__channel name__* - Helps analyze which channel receives the most complaints.bMode of communication used by the customer. Example: Inbound, Outcall, etc.
3> *__Category__* - Main issue category of the customer complaint. Example: Returns, Cancellation, Product Queries, Order Related. Useful for identifying major problem areas.
4> *__Sub-category__* - detailed classification of the issue. Example: Reverse Pickup Enquiry, Installation/demo, etc.
5> *__Customer Remarks__* - Text feedback or comments given by the customer. Can be used for sentiment analysis.
6> *__Order_id__* - Unique identifier for the related order. Not all support tickets may have an associated order.
7> *__order_date_time__* - Date and time when the order was placed. Useful for order-to-issue timeline analysis.
8> *__Issue_reported at__* - Date and time when the customer reported the issue. Important for response time analysis.
9> *__issue_responded__* - Date and time when support responded to the issue. Used to calculate resolution/response time.
10> *__Survey_response_Date__* - Date when the customer submitted their satisfaction survey.
11> *__Customer_City__* - City of the customer. Useful for geographic analysis of

complaints.

12> *__Product_category__* - Category of product associated with the complaint. Example: Electronics, Clothing, etc.

13> *__Item_price__* - Price of the product related to the issue.

14> *__connected_handling_time__* - Time spent by the agent handling the issue. Useful for operational efficiency analysis.

15> *__Agent_name__* - Name of the support agent who handled the ticket. Helps in performance analysis.

16> *__Supervisor__* - Supervisor responsible for the agent.

17> *__Manager__* - Manager overseeing the support team.

18> *__Tenure Bucket__* - Experience level of the agent. Example: 0–30 days, >90 days, On Job Training. Useful to compare performance based on experience.

19> *__Agent Shift__* - Shift during which the issue was handled (Morning, Evening, etc.). Helps analyze shift-based performance.

20> *__CSAT Score__* - Customer Satisfaction Score (1–5 scale). 1 = Very Dissatisfied; 5 = Very Satisfied. Key performance indicator for service quality.

In [153…
```
# CHECK UNIQUE VALUES FOR EACH VARIABLE -
unique_counts = df.nunique()
print("Unique Values Count for Each Column:\n")
print(unique_counts)
```

```
Unique Values Count for Each Column:

Unique id                  85907
channel_name                   3
category                      12
Sub-category                  57
Customer Remarks           18231
Order_id                   67675
order_date_time            13766
Issue_reported at          30923
issue_responded            30262
Survey_response_Date          31
Customer_City               1782
Product_category               9
Item_price                  2789
connected_handling_time      211
Agent_name                  1371
Supervisor                    40
Manager                        6
Tenure Bucket                  5
Agent Shift                    5
CSAT Score                     5
dtype: int64
```

In [155…
```
# CODES THAT MAKE MY DATASET ANALYSIS READY -
df.drop_duplicates(inplace=True) # REMOVING DUPLICATE ROWS
df.columns = df.columns.str.strip().str.lower().str.replace(" ", "_") # STANDARD
```

In [157…
```
date_cols = ["order_date_time", "issue_reported_at",
             "issue_responded", "survey_response_date"]
for col in date_cols:
    if col in df.columns:
        df[col] = pd.to_datetime(
```

```
            df[col],
            format="%d/%m/%Y %H:%M",
            errors="coerce"
        )  # CONVERTING DATE COLUMNS TO DATETIME
```

In [159…
```
# CREATE RESPONSE TIME FEATURE
if "issue_reported_at" in df.columns and "issue_responded" in df.columns:
    df["response_time_minutes"] = (df["issue_responded"] - df["issue_reported_at
```

In [161…
```
# NOW I HAVE TO HANDLE MISSING VALUES HERE
num_cols = df.select_dtypes(include="number").columns

for col in num_cols:
    df[col] = df[col].fillna(df[col].median())
 # HERE I AM FILLING NUMERICAL COLUMNS WITH THEIR RESPECTIVE MEDIAN
```

In [163…
```
# FILLING CATEGORIAL COLUMNS WITH UNKNOWN
cat_cols = df.select_dtypes(include="object").columns
# FILLING MISSING VALUES
df[cat_cols] = df[cat_cols].fillna("Unknown")
```

In [165…
```
# EXTRACY DAY , MONTH FROM ISSUE DATE
if "issue_reported_at" in df.columns:
    df["issue_day"] = df["issue_reported_at"].dt.day
    df["issue_month"] = df["issue_reported_at"].dt.month
    df["issue_weekday"] = df["issue_reported_at"].dt.day_name()
```

In [167…
```
# CSAT CATEGORY
if "csat_score" in df.columns:
    df["csat_category"] = df["csat_score"].apply(
        lambda x: "Low" if x <= 2 else ("Medium" if x == 3 else "High")
    )
```

In [169…
```
# FINAL CHECK
print("Dataset Shape After Cleaning:", df.shape)
print("\nRemaining Missing Values:\n", df.isnull().sum().sum())
print("\nDataset is now Analysis Ready")
```

```
Dataset Shape After Cleaning: (85907, 25)

Remaining Missing Values:
 154600

Dataset is now Analysis Ready
```

In [171…
```
df.head()
```

Out[171…

	unique_id	channel_name	category	sub-category	customer_remarks	
0	7e9ae164-6a8b-4521-a2d4-58f7c9fff13f	Outcall	Product Queries	Life Insurance	Unknown	c fa 2100
1	b07ec1b0-f376-43b6-86df-ec03da3b2e16	Outcall	Product Queries	Product Specific Information	Unknown	d ce f08d4
2	200814dd-27c7-4149-ba2b-bd3af3092880	Inbound	Order Related	Installation/demo	Unknown	c b9 62d6
3	eb0d3e53-c1ca-42d3-8486-e42c8d622135	Inbound	Returns	Reverse Pickup Enquiry	Unknown	5 55 97942
4	ba903143-1e54-406c-b969-46c52f92e5df	Inbound	Cancellation	Not Needed	Unknown	e 6 ccefc

5 rows × 25 columns

In [173…

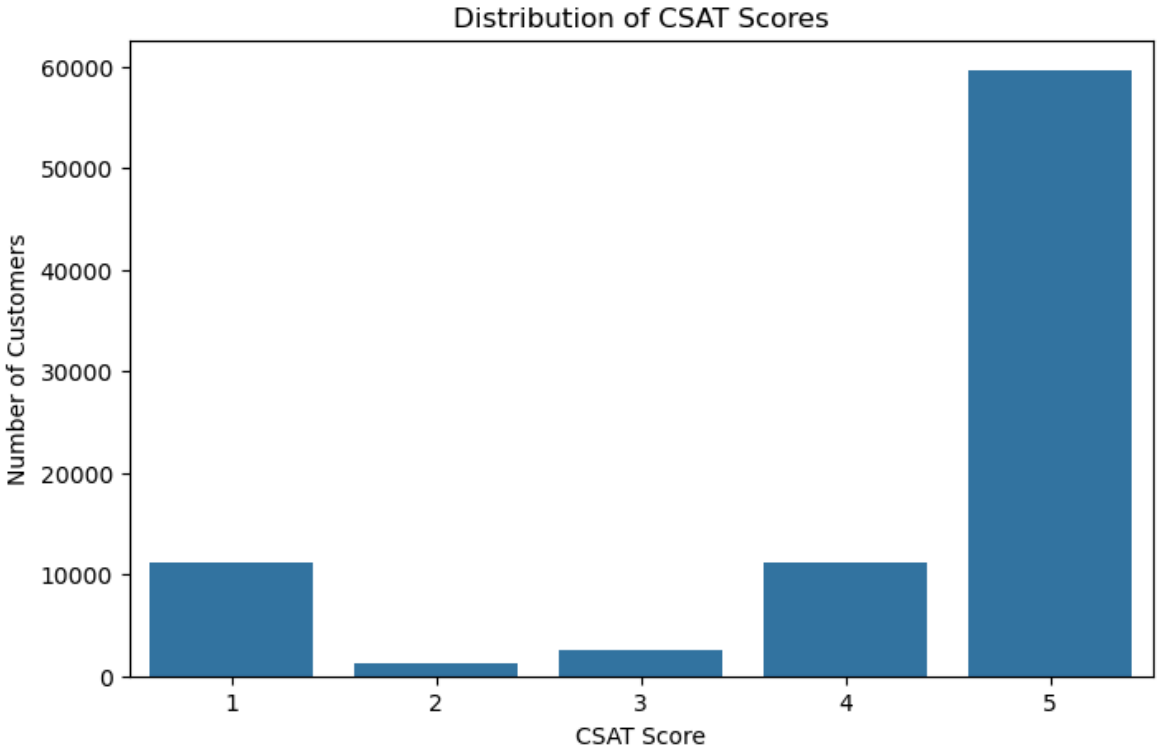
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 85907 entries, 0 to 85906
Data columns (total 25 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   unique_id                85907 non-null  object
 1   channel_name             85907 non-null  object
 2   category                 85907 non-null  object
 3   sub-category             85907 non-null  object
 4   customer_remarks         85907 non-null  object
 5   order_id                 85907 non-null  object
 6   order_date_time          17214 non-null  datetime64[ns]
 7   issue_reported_at        85907 non-null  datetime64[ns]
 8   issue_responded          85907 non-null  datetime64[ns]
 9   survey_response_date     0 non-null      datetime64[ns]
 10  customer_city            85907 non-null  object
 11  product_category         85907 non-null  object
 12  item_price               85907 non-null  float64
 13  connected_handling_time  85907 non-null  float64
 14  agent_name               85907 non-null  object
 15  supervisor               85907 non-null  object
 16  manager                  85907 non-null  object
 17  tenure_bucket            85907 non-null  object
 18  agent_shift              85907 non-null  object
 19  csat_score               85907 non-null  int64
 20  response_time_minutes    85907 non-null  float64
 21  issue_day                85907 non-null  int32
 22  issue_month              85907 non-null  int32
 23  issue_weekday            85907 non-null  object
 24  csat_category            85907 non-null  object
dtypes: datetime64[ns](4), float64(3), int32(2), int64(1), object(15)
memory usage: 15.7+ MB
```

Plot 1

In [177…

```
plt.figure(figsize=(8,5))
sns.countplot(x="csat_score", data=df)
plt.title("Distribution of CSAT Scores")
plt.xlabel("CSAT Score")
plt.ylabel("Number of Customers")
plt.show()
```



__Why this plot?__

CSAT Score tells how satisfied customers are.

Everything else (agent, shift, response time) affects this.

If most scores are 5 :

Dataset is skewed

Customers are mostly satisfied

ML model may become biased

Business insight :

Are customers happy?

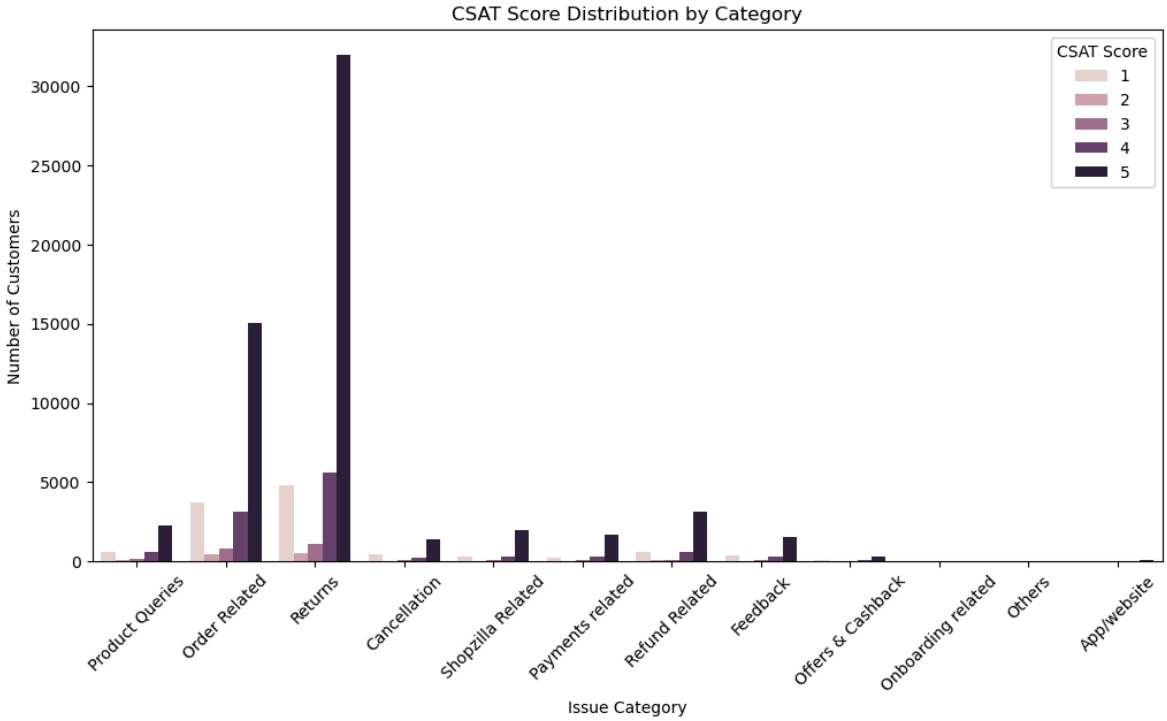
Are there many low ratings?

Is service performance stable?

Plot 2

In [182…

```
plt.figure(figsize=(12,6))
sns.countplot(x="category", hue="csat_score", data=df)
plt.title("CSAT Score Distribution by Category")
plt.xlabel("Issue Category")
plt.ylabel("Number of Customers")
plt.xticks(rotation=45)
plt.legend(title="CSAT Score")
plt.show()
```



__Why this Plot ?__

Identifies Problem Categories

Which category gets more low ratings?

Which category performs well?

Example :

Returns might have more 1–2 scores

Product Queries might have more 4–5 scores

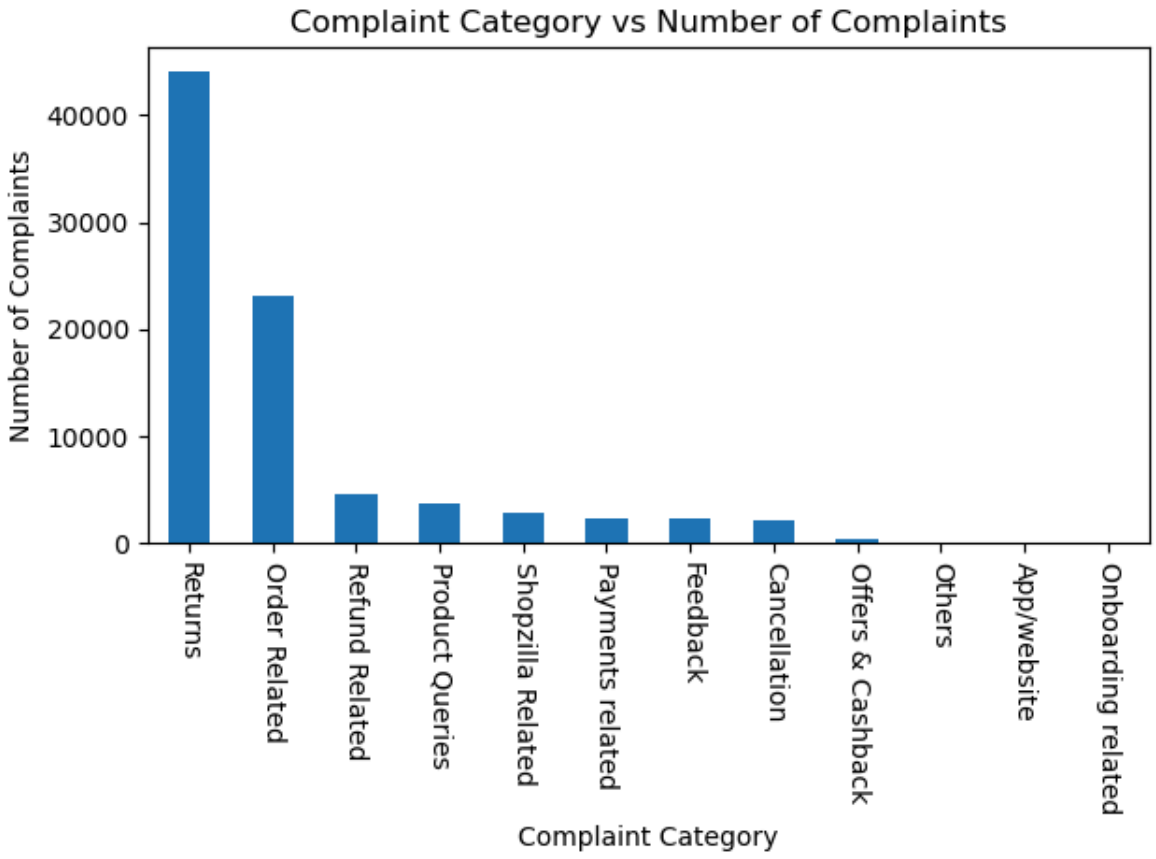
Key difference between Plot 1 and Plot 2 -

Plot 1 showed overall satisfaction while Plot 2 explains why satisfaction looks that way.

Plot 3

In [187…

```
complaint_counts = df['category'].value_counts()
plt.figure()
complaint_counts.plot(kind='bar')
plt.title('Complaint Category vs Number of Complaints')
plt.xlabel('Complaint Category')
plt.ylabel('Number of Complaints')
plt.xticks(rotation=990)

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

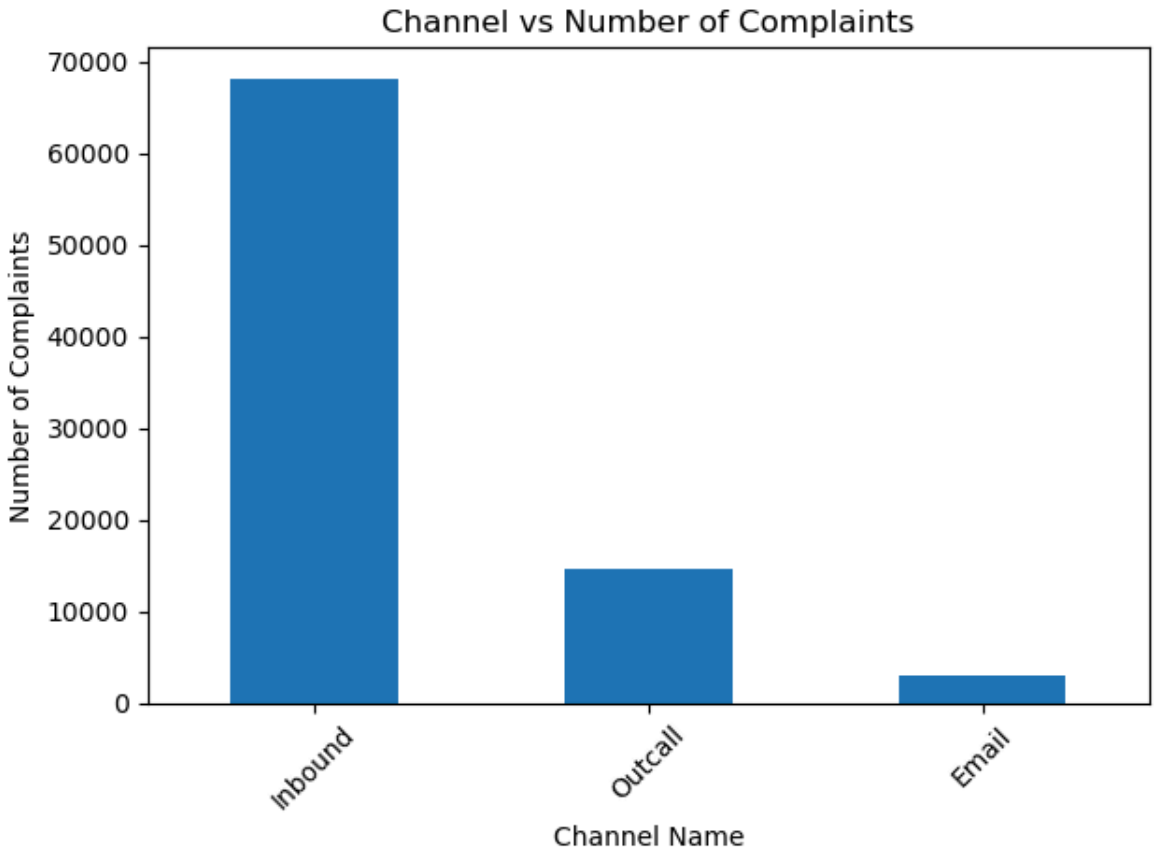


What this plot shows -

Returns has the highest number of complaints.
Followed by Order Related issues.
Other categories like Refund Related, Product Queries, Cancellation, etc., have comparatively fewer complaints.
Very few complaints are related to App/Website and Onboarding.

Plot 4

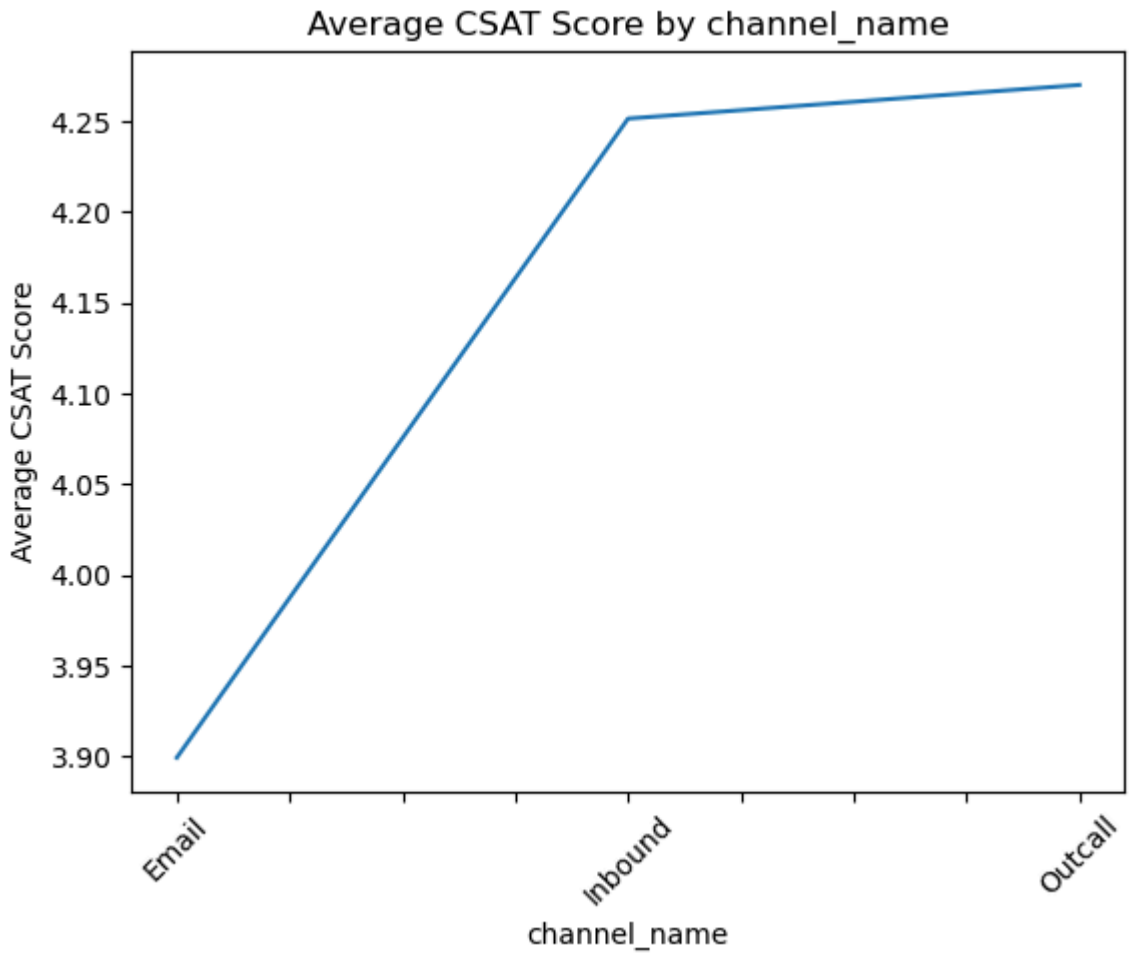
In [64]:
```
channel_counts = df['channel_name'].value_counts()

plt.figure()
channel_counts.plot(kind='bar')

plt.title('Channel vs Number of Complaints')
plt.xlabel('Channel Name')
plt.ylabel('Number of Complaints')
plt.xticks(rotation=45)

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



Plot 5

In [66]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

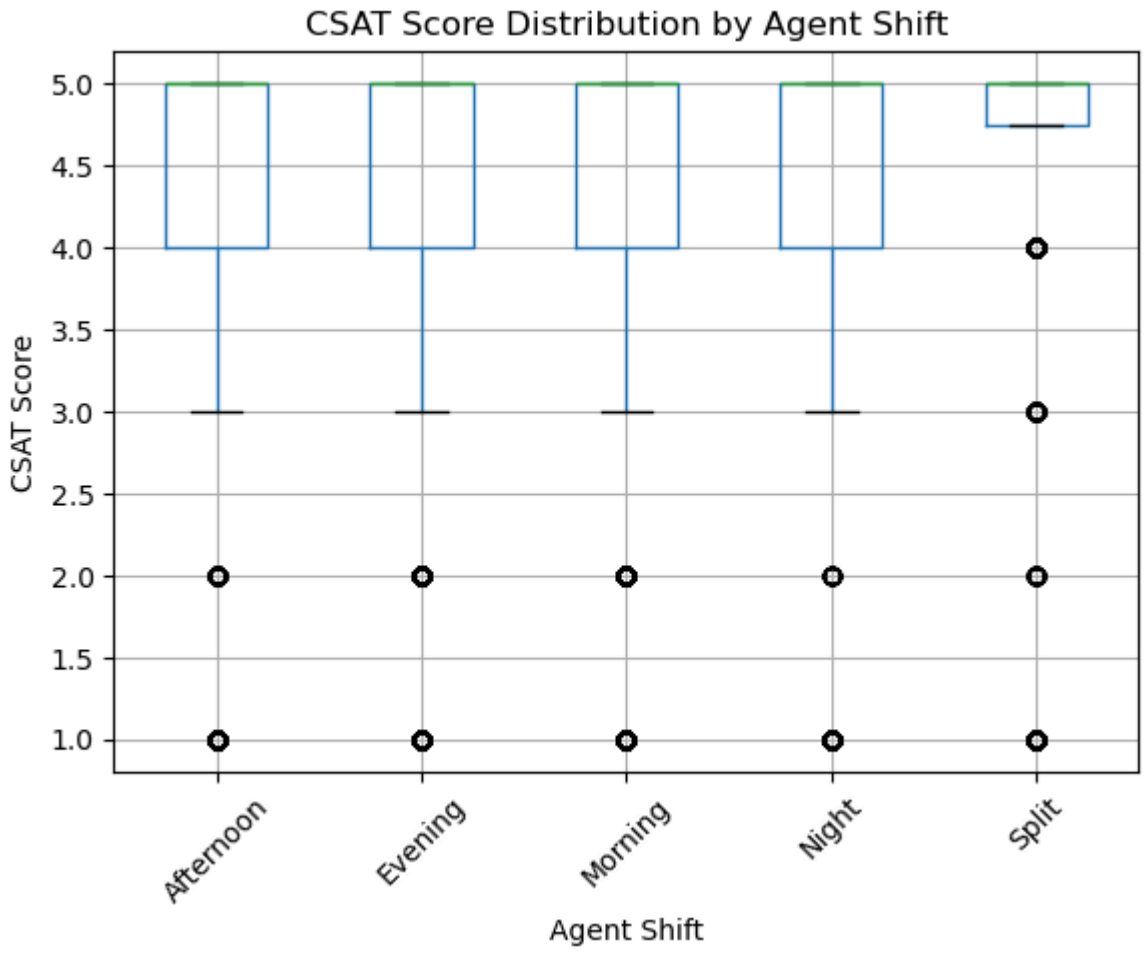
In [68]:
```
plt.figure()
df.groupby('channel_name')['csat_score'].mean().plot()
plt.xlabel('channel_name')
plt.ylabel('Average CSAT Score')
plt.title('Average CSAT Score by channel_name')
plt.xticks(rotation=45)
plt.show()
```



Plot 6

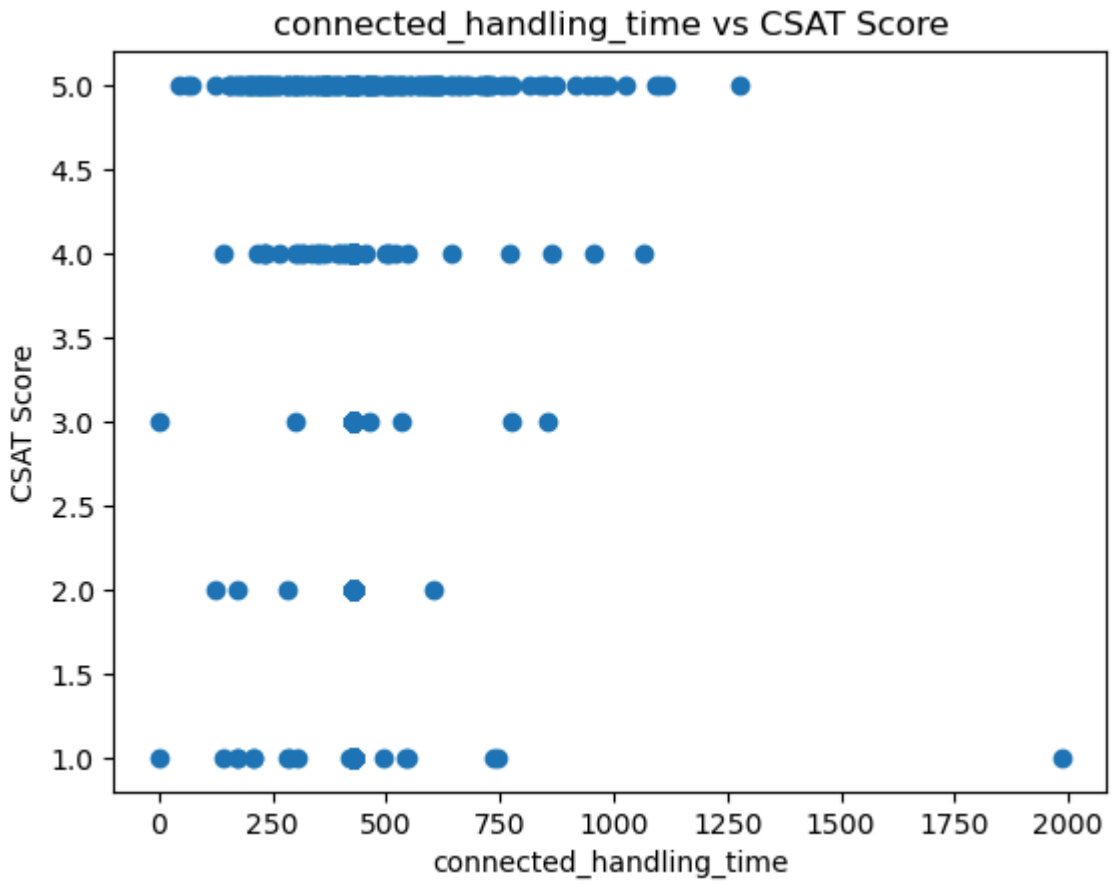
In [70]:
```
plt.figure()
df.boxplot(column='csat_score', by='agent_shift')
plt.title('CSAT Score Distribution by Agent Shift')
plt.suptitle('')
plt.xlabel('Agent Shift')
plt.ylabel('CSAT Score')
plt.xticks(rotation=45)
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```



Plot 7

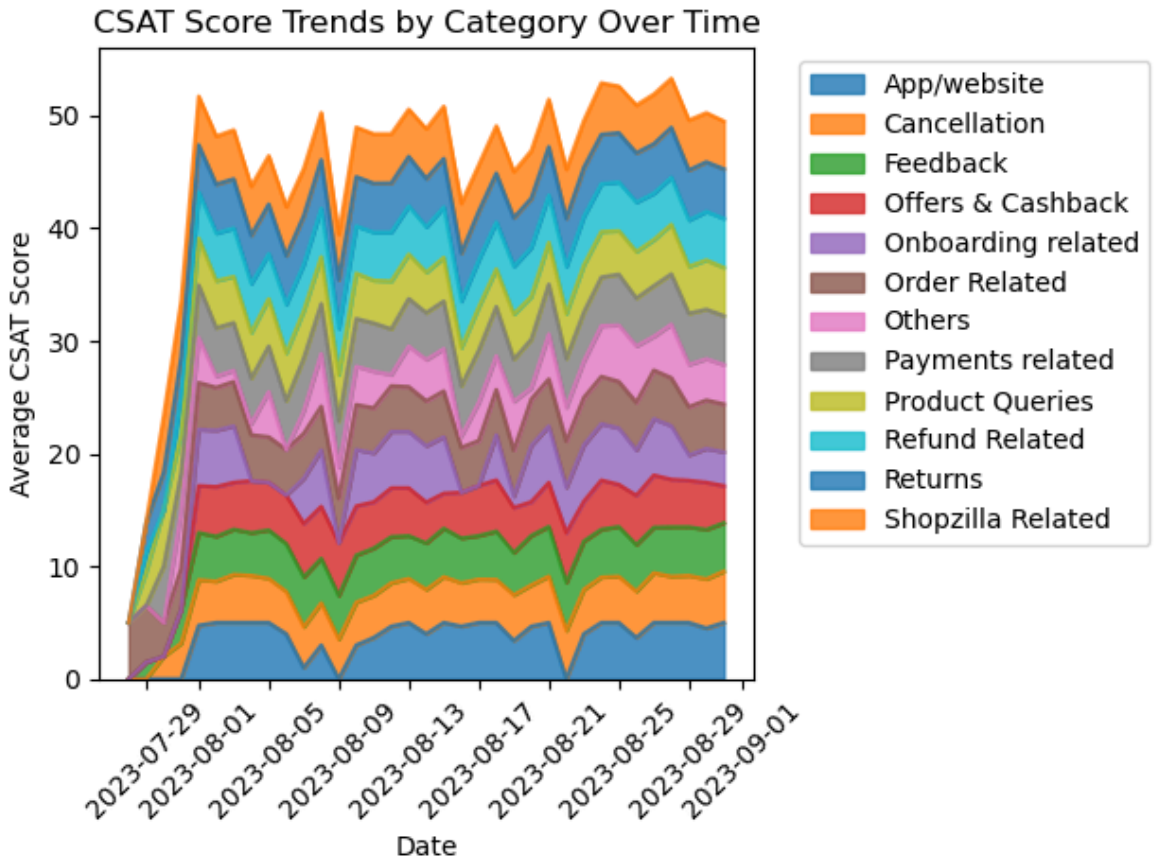
In [72]:
```
plt.figure()
plt.scatter(df['connected_handling_time'], df['csat_score'])
plt.xlabel('connected_handling_time')
plt.ylabel('CSAT Score')
plt.title('connected_handling_time vs CSAT Score')
plt.show()
```



Plot 8

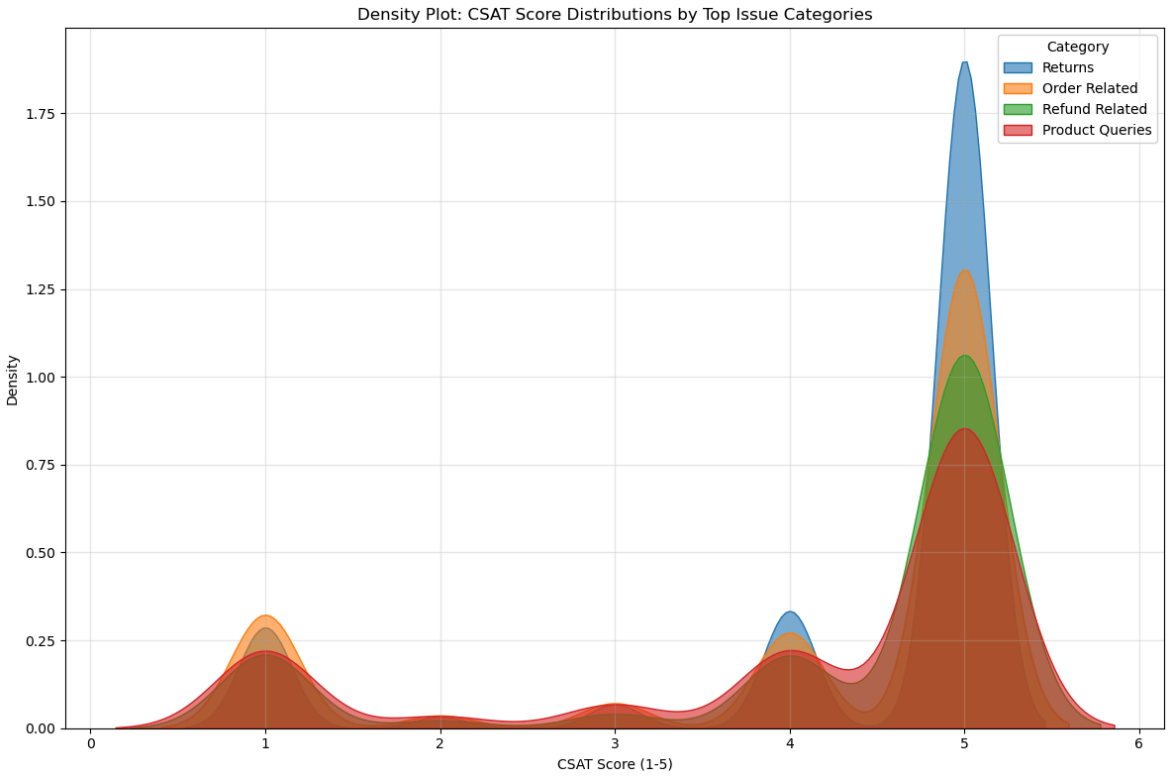
In [89]:
```
df['Issue_reported'] = pd.to_datetime(df['issue_reported_at'], format='%d/%m/%Y
df = df.dropna(subset=['issue_reported_at', 'category', 'csat_score'])
df['date'] = df['issue_reported_at'].dt.date
daily_csat = df.groupby(['date', 'category'])['csat_score'].mean().unstack(fill_
plt.figure(figsize=(12, 6))
daily_csat.plot(kind='area', stacked=True, alpha=0.8)
plt.title('CSAT Score Trends by Category Over Time')
plt.xlabel('Date')
plt.ylabel('Average CSAT Score')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
<Figure size 1200x600 with 0 Axes>
```



Plot 9

In [91]:
```
df['Issue_reported'] = pd.to_datetime(df['issue_reported_at'], format='%d/%m/%Y
df = df.dropna(subset=['issue_reported_at', 'category', 'csat_score'])
df['date'] = df['issue_reported_at'].dt.date
plt.figure(figsize=(12, 8))
top_cats = df['category'].value_counts().head(4).index
for cat in top_cats:
    cat_data = df[df['category'] == cat]['csat_score']
    sns.kdeplot(data=cat_data, label=cat, fill=True, alpha=0.6)
plt.title('Density Plot: CSAT Score Distributions by Top Issue Categories')
plt.xlabel('CSAT Score (1-5)')
plt.ylabel('Density')
plt.legend(title='Category')
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()
```



In [ ]:

__Hypothesis Testing 1__

Does complaint category affect customer satisfaction (CSAT Score)?
Null Hypothesis (H_0): There is no significant difference in average CSAT scores across different complaint categories.
Alternative Hypothesis (H_1): There is a significant difference in average CSAT scores across different complaint categories.

__Hypothesis Testing 2__

Does agent experience impact customer satisfaction?
Null Hypothesis (H_0): Agent tenure has no significant effect on CSAT score.
Alternative Hypothesis (H_1): Agent tenure has a significant effect on CSAT score.

__Hypothesis Testing 3__

Does support channel (Chat, Email, Call) impact customer satisfaction?
Null Hypothesis (H_0): There is no significant difference in CSAT scores across different support channels.
Alternative Hypothesis (H_1): There is a significant difference in CSAT scores across different support channels.

__ML Model Implementation 1__

In [103…
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
df.columns = df.columns.str.strip()
X = df[['item_price', 'connected_handling_time']]
y = df['csat_score']
# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
# Initialize Model
model = LinearRegression()
# Fit the Algorithm
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Predicted Values:")
print(y_pred[:5])
```

```
Predicted Values:
[4.26045743 4.27802122 4.26045743 4.26045743 4.27507614]
```

In this project, a Linear Regression model was implemented to predict customer satisfaction scores (CSAT) based on relevant features such as handling time and item price. Linear Regression was chosen as the initial model because it is simple, interpretable, and effective for understanding linear relationships between independent variables and a continuous target variable. The dataset was divided into training and testing sets to evaluate the model's generalization ability. The model was trained using the training dataset and predictions were generated on the unseen test dataset.

Which hyperparameter optimization technique have you used and why?

In the current implementation, no hyperparameter optimization technique was applied, because the model used is Linear Regression, which does not have major hyperparameters that require tuning like tree-based or ensemble models. Linear Regression mainly learns coefficients directly from the data using the Ordinary Least Squares method. Since it does not involve parameters like tree depth, number of estimators, or learning rate, hyperparameter tuning is generally not necessary for the basic version of this algorithm.

ML Model Implementation 2

In [110…
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, accuracy_score
# Target variable
y = df['csat_score']
feature_cols = ['channel_name', 'category', 'sub-category', 'customer_city',
                'product_category', 'item_price', 'connected_handling_time',
                'tenure_bucket', 'agent_shift']
X = df[feature_cols].copy()
label_encoders = {}
for col in feature_cols:
```

```
    if X[col].dtype == 'object':
        le = LabelEncoder()
        X[col] = le.fit_transform(X[col].astype(str))
        label_encoders[col] = le
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
# Train Random Forest model
model = RandomForestClassifier(n_estimators=100, random_state=42, class_weight='
model.fit(X_train, y_train)
# Predictions
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.3f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Model Accuracy: 0.353

Classification Report:
              precision    recall  f1-score   support

           1       0.19      0.16      0.17      2246
           2       0.02      0.19      0.03       256
           3       0.03      0.15      0.05       512
           4       0.13      0.13      0.13      2244
           5       0.71      0.44      0.55     11924

    accuracy                           0.35     17182
   macro avg       0.21      0.22      0.19     17182
weighted avg       0.54      0.35      0.42     17182
```

What does this model predict?

In this implementation, a Random Forest Classifier was developed to predict customer satisfaction score (CSAT) as a classification problem. Instead of predicting a continuous value, the model predicts the CSAT rating category (for example, 1–5). The goal of the model is to determine how various operational and customer-related features influence customer satisfaction levels. The target variable selected is csat_score, and multiple features were used, including support channel, complaint category, sub-category, customer city, product category, item price, handling time, agent tenure, and agent shift. These features represent both categorical and numerical information related to the customer support interaction. Since machine learning models cannot directly process categorical text data, Label Encoding was applied to transform categorical columns into numerical format. Each unique category value was converted into a numeric label, allowing the Random Forest algorithm to process the data correctly. The dataset was then split into training and testing sets using an 80-20 split. Stratified sampling was applied to ensure that the distribution of CSAT classes remained balanced in both training and testing sets. This helps prevent biased model evaluation. A Random Forest Classifier with 100 decision trees (n_estimators=100) was used. The class_weight='balanced' parameter was included to handle any class imbalance in CSAT ratings, ensuring that minority classes are not ignored during training. Random Forest works by building multiple decision trees on random subsets of data and combining their predictions to produce a final result. This ensemble method improves accuracy and reduces overfitting compared to a single decision tree. After training the model,

predictions were generated on the test dataset. Model performance was evaluated using accuracy score and a classification report, which includes precision, recall, and F1-score for each CSAT class. These metrics help assess how well the model correctly predicts satisfaction levels across different categories. Overall, this model captures nonlinear relationships between service-related factors and customer satisfaction, making it more powerful than a basic linear model for structured business data.

Ml Model Implementation 3

In [208…
```
pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\kundu\anaconda3\lib\site-packa
ges (3.2.0)Note: you may need to restart the kernel to use updated packages.

Requirement already satisfied: numpy in c:\users\kundu\anaconda3\lib\site-package
s (from xgboost) (1.26.4)
Requirement already satisfied: scipy in c:\users\kundu\anaconda3\lib\site-package
s (from xgboost) (1.13.1)
```

In [206…
```
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, accuracy_score
import joblib
y = df['category']
feature_cols = ['channel_name', 'customer_city', 'product_category', 'item_price
                'connected_handling_time', 'agent_name', 'supervisor', 'manager'
                'tenure_bucket', 'agent_shift', 'csat_score']
X = df[feature_cols].copy()
X['connected_handling_time'] = pd.to_numeric(X['connected_handling_time'], error
label_encoders = {}
for col in feature_cols + ['category']:
    le = LabelEncoder()
    if col in feature_cols:
        X[col] = le.fit_transform(X[col].astype(str))
    else:
        y = le.fit_transform(y.astype(str))
    label_encoders[col] = le
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
model = XGBClassifier(n_estimators=200, max_depth=6, learning_rate=0.1, random_s
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Category Prediction Accuracy: {accuracy:.3f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred,zero_division=0, target_names=label_e
```

```
Category Prediction Accuracy: 0.534

Classification Report:
                    precision    recall  f1-score   support

       App/website       0.00      0.00      0.00        17
      Cancellation       0.33      0.01      0.01       442
          Feedback       0.00      0.00      0.00       459
 Offers & Cashback       0.00      0.00      0.00        96
Onboarding related       0.00      0.00      0.00        13
     Order Related       0.49      0.21      0.29      4643
            Others       0.00      0.00      0.00        20
  Payments related       0.00      0.00      0.00       465
   Product Queries       0.00      0.00      0.00       739
    Refund Related       0.32      0.01      0.02       910
           Returns       0.54      0.93      0.68      8820
 Shopzilla Related       0.50      0.00      0.00       558

          accuracy                           0.53     17182
         macro avg       0.18      0.10      0.08     17182
      weighted avg       0.45      0.53      0.43     17182
```

__Why this model?__

In this model, the model is predicting the support ticket category for each customer interaction. Specifically, the target variable is category, which means the model learns patterns from features such as channel name, customer city, product category, item price, handling time, agent details, tenure bucket, shift, and CSAT score to determine what type of issue the ticket belongs to. After training, when new customer support data is given, the XGBoost classifier analyzes the combination of these inputs and predicts the most likely category (for example, refund-related issue, technical problem, delivery complaint, etc., depending on the dataset labels). In simple terms, your model automates the classification of customer support tickets so that instead of manually identifying the issue type, the system can instantly assign the correct category based on historical patterns learned from the data.

__CONCLUSION -__

This project focused on building a machine learning solution to improve customer support operations by predicting key outcomes from historical interaction data. After understanding the business objective, the dataset was cleaned and preprocessed by handling missing values, converting numerical fields properly, and encoding categorical variables to make them suitable for modeling. Exploratory analysis helped in understanding feature distributions and their relevance to business performance. A structured train-test split with stratification ensured fair evaluation across categories. Multiple models were explored, and an XGBoost classifier was implemented due to its strong performance on structured data and ability to capture complex relationships. The model was trained to predict support ticket categories based on customer details, product information, agent attributes, handling time, and CSAT score. Evaluation metrics such as accuracy, precision, recall, and F1-score were used to assess performance and interpret the model's business impact. The final outcome demonstrates how machine

learning can automate ticket classification, reduce manual effort, improve routing efficiency, and support faster resolution times, ultimately enhancing customer satisfaction and operational effectiveness.

In [ ]: