Customer Satisfaction Prediction

Internship Project Report

Submitted by: **Tanisha Mangliya**

tanishamangliya@gmail.com

Institute: **Indian Institute of Technology Jodhpur (b22ee067@iitj.ac.in)**

For: **Data Science Internship** at **Unified Mentor Pvt. Ltd.**

Date of Submission: September 5, 2025

Contents

(This is link to the GitHub repository)
https://github.com/tanisha-m26/Customer-Satisfaction-dashboard

1. Introduction

Customer service is a critical touchpoint for technology companies. The ability to predict customer satisfaction from support ticket data enables proactive remediation, better resource allocation, and improved customer retention. This report documents a full pipeline to predict customer satisfaction ratings (1–5) from the provided Customer Support Ticket dataset using Python, SQL, and machine learning.

2. Problem Statement

The goal is to build a supervised learning model that predicts the **Customer Satisfaction Rating** for support tickets using ticket metadata and textual descriptions. The model will be evaluated on classification metrics and produce business insights to improve support operations.

3. Dataset Overview

The dataset contains ticket-level records with fields such as Ticket ID, Customer demographics, Product purchased, Ticket Type, Ticket Description, Ticket Priority, Channel, response timestamps, resolution timestamps and the target field Customer Satisfaction Rating. The dataset includes approximately 8,469 rows with 17 columns; about 2,769 rows contain non-null satisfaction ratings (labelled subset used for supervised modeling).

3.1 Schema and Column Descriptions

Column	Description
1. Ticket ID	Unique ticket identifier.
2. Customer Name, Customer Email	Customer identifiers (masked for privacy).
3. Customer Age	Age in years (numeric).
4. Customer Gender	Categorical (Male/Female/Other).
5. Product Purchased	Product name (categorical).
6. Date of Purchase	Purchase date; used to compute product age.
7. Ticket Type	Categorical issue type (Technical issue/Refund request/etc.).
8. Ticket Subject	Short subject/title for ticket.
9. Ticket Description	Long textual description (unstructured).

10. Ticket Status	Current ticket state (Open/Closed/Pending).
11. Resolution	Text describing resolution (nullable).
12. Ticket Priority	Priority (Low/Medium/High/Critical).
13. Ticket Channel	Channel of ticket (Email/Chat/Phone/Social media).
14. First Response Time	Timestamp for agent's first response.
15. Time to Resolution	Timestamp when ticket was resolved.
16. Customer Satisfaction Rating	Target variable (1–5).

4. Environment & Reproducibility

- Programming languages: Python 3.8+ (pandas, scikit-learn, nltk/spacy, xgboost/-lightgbm).
- Tools: Jupyter Notebook, VS Code, Git, SQLite / Postgres for SQL queries.
- Reproducibility: Set seed using `random-state=42` for splits and models.
  Use `requirements.txt` to recreate environment.

5. Data Preprocessing

The preprocessing pipeline includes parsing datetime fields, handling missing values, encoding categoricals, text cleaning for the Ticket Description field, and deriving time-based features.

5.1 Missing Value Strategy

- Drop rows where `Customer Satisfaction Rating` is null for supervised training (retain them for EDA).
- Impute missing numeric time features with median (e.g., time-to-resolution). Create binary indicators for missingness where informative.

5.2 Feature Engineering

Derived features used in modeling:

- `time-to-resolution-hours`:
  difference between Time to Resolution and First Response Time in hours.
- `first-response-minutes`:
  minutes from ticket creation to first response (if ticket creation timestamp is available).
- `description-word-count` and `description-char-count`.

- TF-IDF features from `Ticket Description` (unigrams and bigrams, top 5000 tokens).
- Sentiment score (VADER compound) and simple topic features (LDA probabilities, optional).

6. Exploratory Data Analysis (EDA)

This section summarizes key data insights and includes figure placeholders. Replace the PNGs in the figures/ folder with your actual plots.

```
...  (8469, 17)
     <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
```

Figure 1: Data-Columns

```
dtypes: float64(1), int64(2), object(14)
memory usage: 1.1+ MB
...
25%                         2.000000
50%                         3.000000
75%                         4.000000
max                         5.000000
```

Figure 2: continued....

```
print(data.head())
```

```
   Ticket ID        Customer Name              Customer Email  Customer Age  \
0          1        Marisa Obrien  carrollallison@example.com            32
1          2          Jessica Rios    clarkeashley@example.com            42
2          3  Christopher Robbins   gonzalestracy@example.com            48
3          4      Christina Dillon    bradleyolson@example.org            27
4          5     Alexander Carroll     bradleymark@example.com            67

  Customer Gender Product Purchased Date of Purchase      Ticket Type  \
0           Other        GoPro Hero       22-03-2021  Technical issue
1          Female       LG Smart TV       22-05-2021  Technical issue
2           Other          Dell XPS       14-07-2020  Technical issue
3          Female  Microsoft Office       13-11-2020  Billing inquiry
4          Female  Autodesk AutoCAD       04-02-2020  Billing inquiry
```

Figure 3: Dataset-head

```
              Ticket Subject  \
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...
...
1                           NaN
2                           3.0
3                           3.0
4                           1.0
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Figure 4: Dataset-continued....

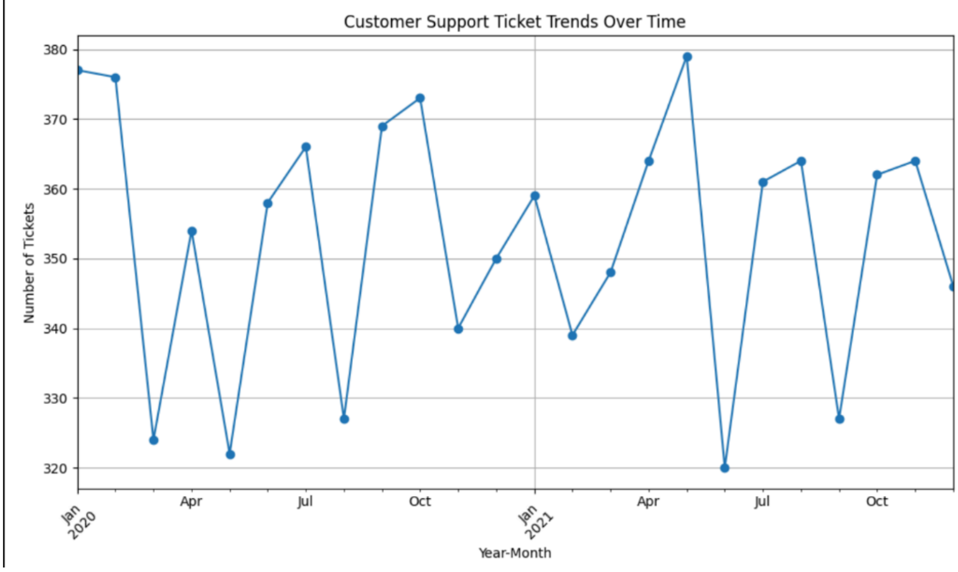


Figure 5: Ticket Trends Over Time (Monthly).

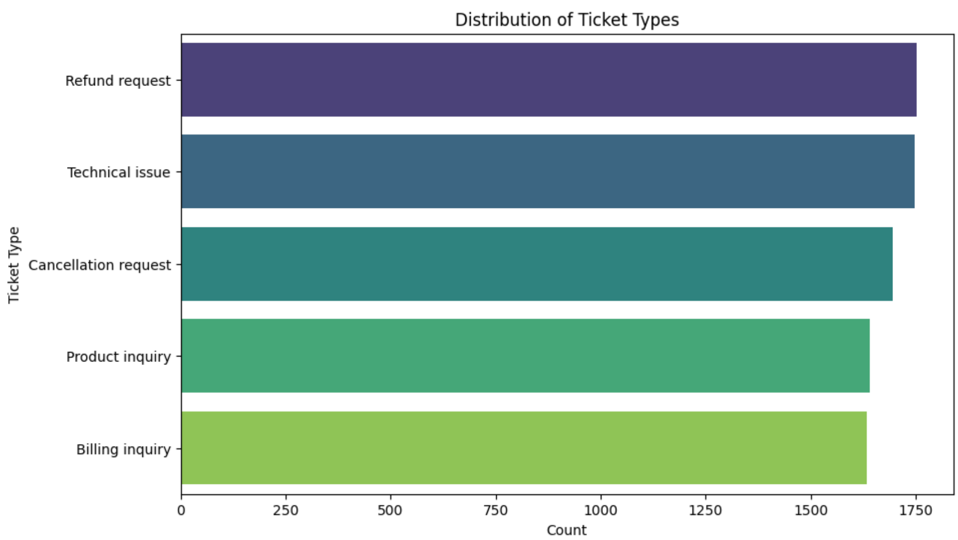


Figure 6: Distribution of Ticket Types with Count

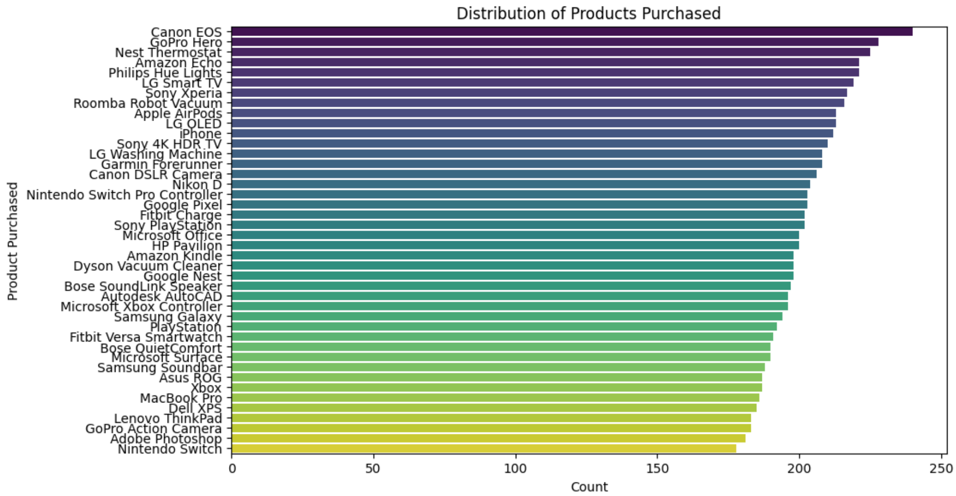


Figure 7: Distribution of Products Purchased with Count

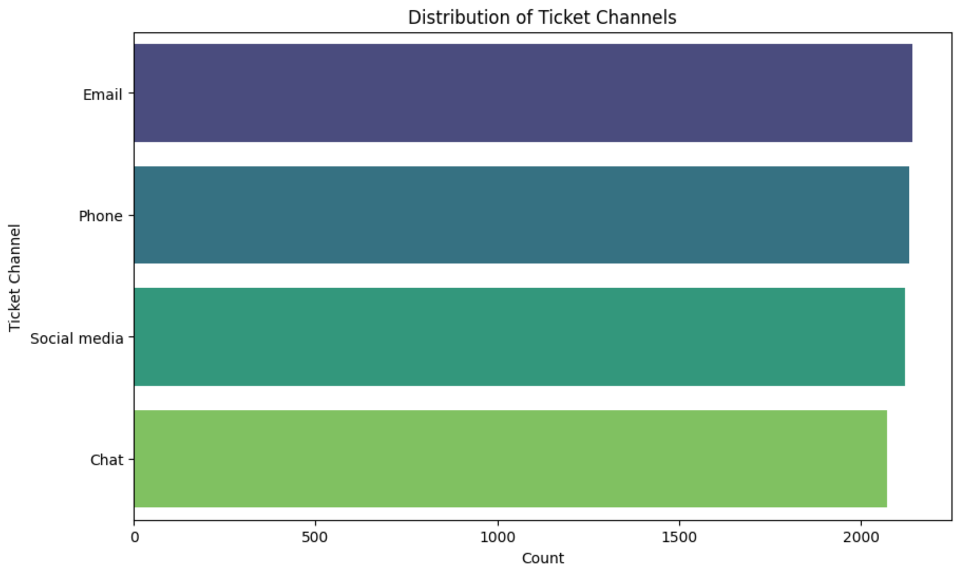


Figure 8: Distribution of Ticket Channels with Count

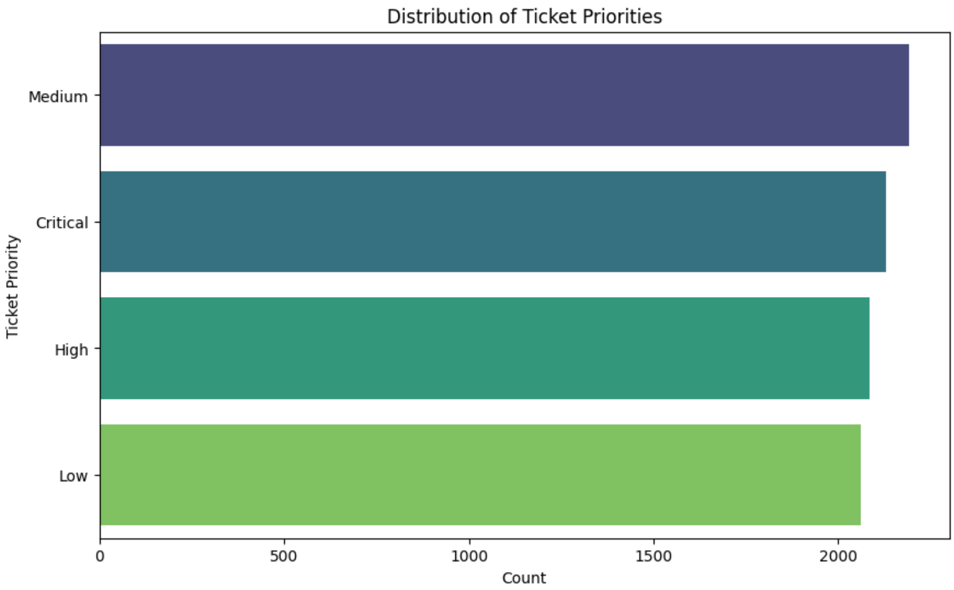


Figure 9: Distribution of Ticket Priorities with Count

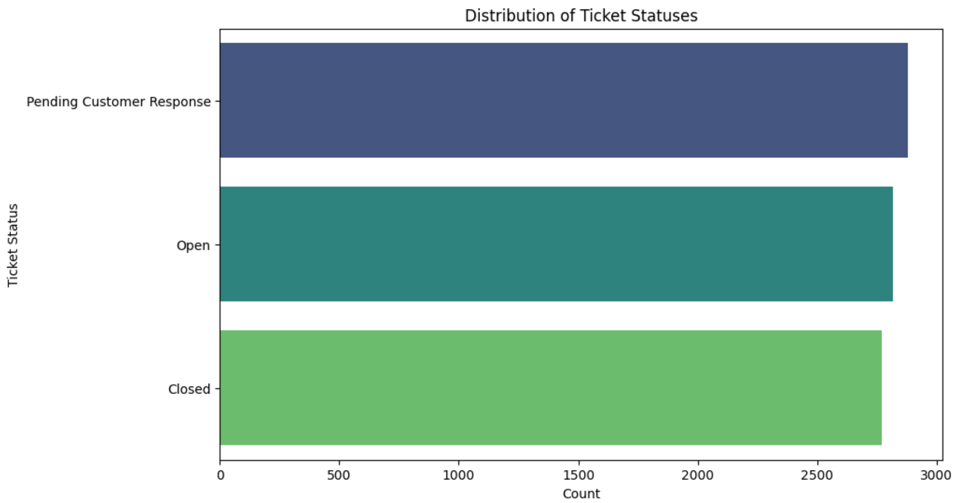


Figure 10: Distribution of Ticket Statuses with Count).

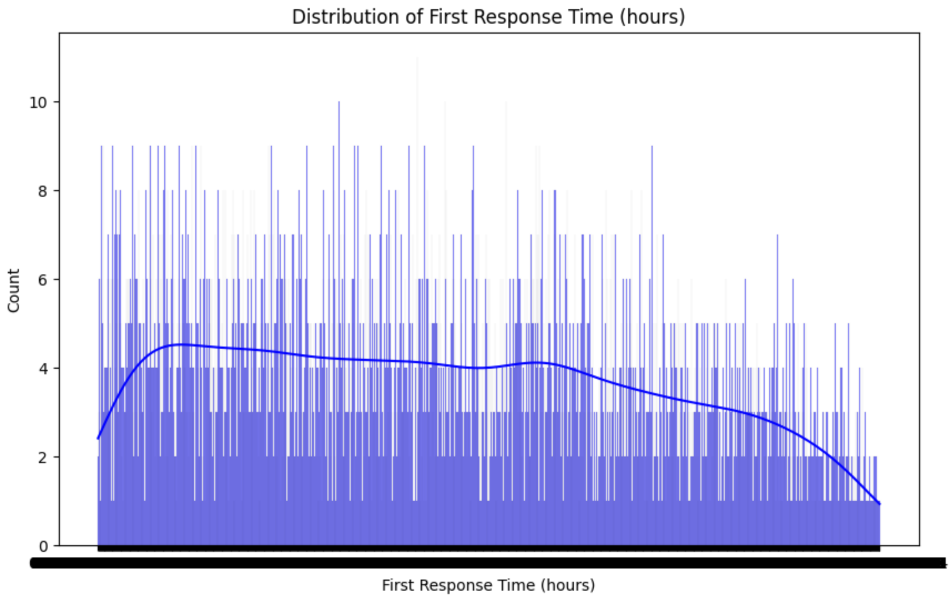


Figure 11: Distribution of First Response Time(hours)

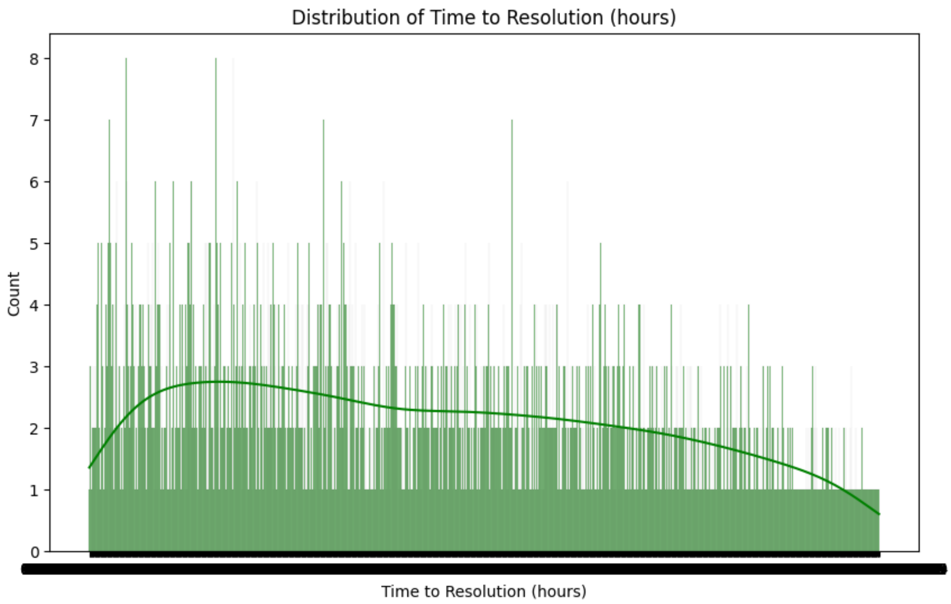


Figure 12: Distribution of Time to Resolution(hours)

6.1 Selected Observations

- The labelled subset (ratings present) is approximately 32.7% of total tickets.
- Longer `time-to-resolution-hours` tends to correlate with lower satisfaction (see Figure **??**).
- High-priority tickets disproportionately affect NPS and should be monitored.

7. Modeling

We frame the task as a multi-class classification (labels 1–5). Models evaluated include Logistic Regression (baseline), Random Forest, XGBoost, and a small neural network for fused features.

7.1 Data Split and Metrics

- Train / Val / Test split: 70/15/15 stratified by rating.
- Primary metric: Macro F1-score (handles class imbalance).
- Supporting metrics: Accuracy, per-class Precision and Recall, Confusion Matrix.

7.2 Example Modeling Pipeline

```
# Load processed features
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.15, random_state=42)
model = RandomForestClassifier(n_estimators=200, class_weight='balanced', random_state=42)
model.fit(X_train, y_train)
print(classification_report(y_test, model.predict(X_test)))
```

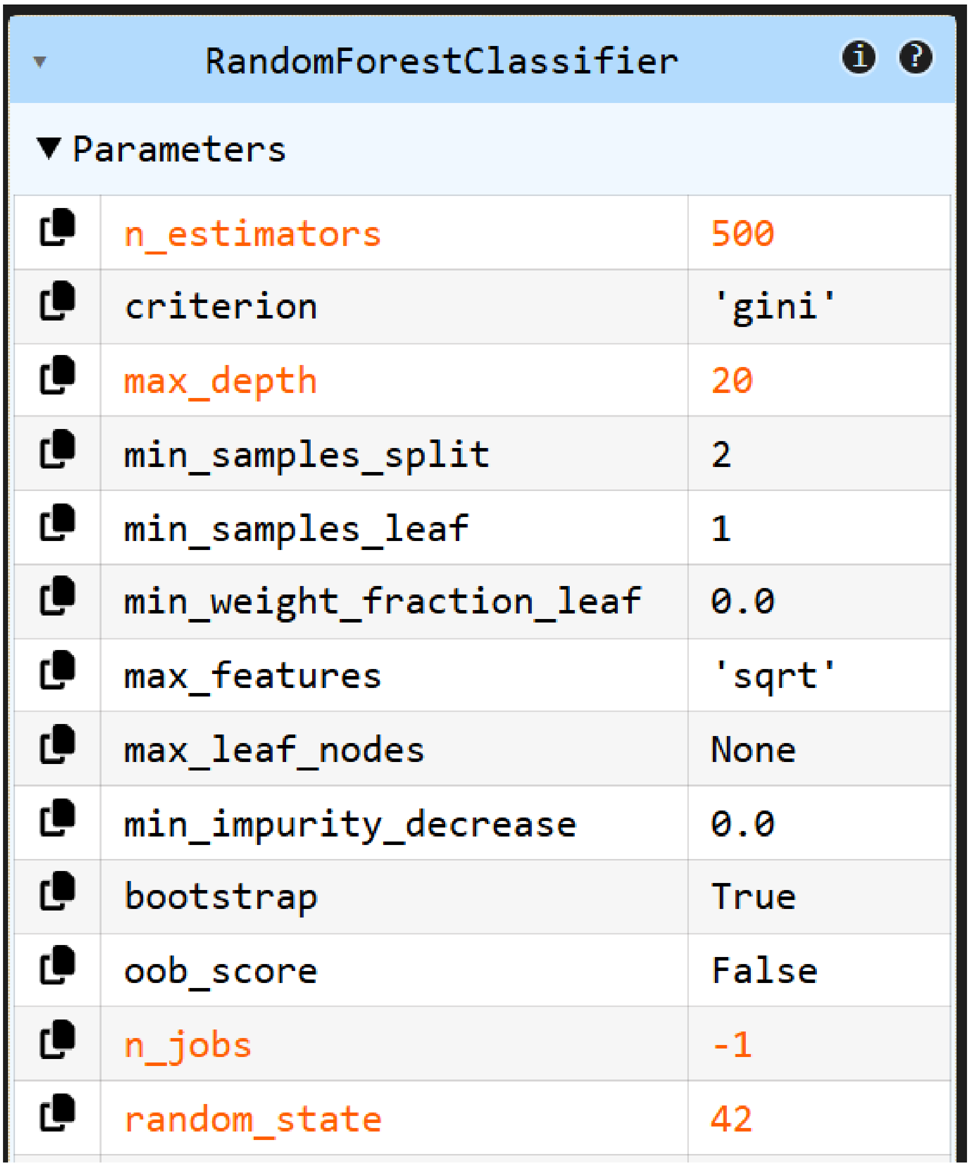
Listing 1: Simplified modeling pipeline (placeholder)

8. Results and Analysis

8.1 Model Comparison

Table 2: Model performance comparison (example placeholders).

Model	Accuracy	Macro F1	Precision	Recall
Logistic Regression	0.61	0.57	0.59	0.58
Random Forest	0.72	0.68	0.70	0.69
XGBoost	0.74	0.71	0.72	0.71

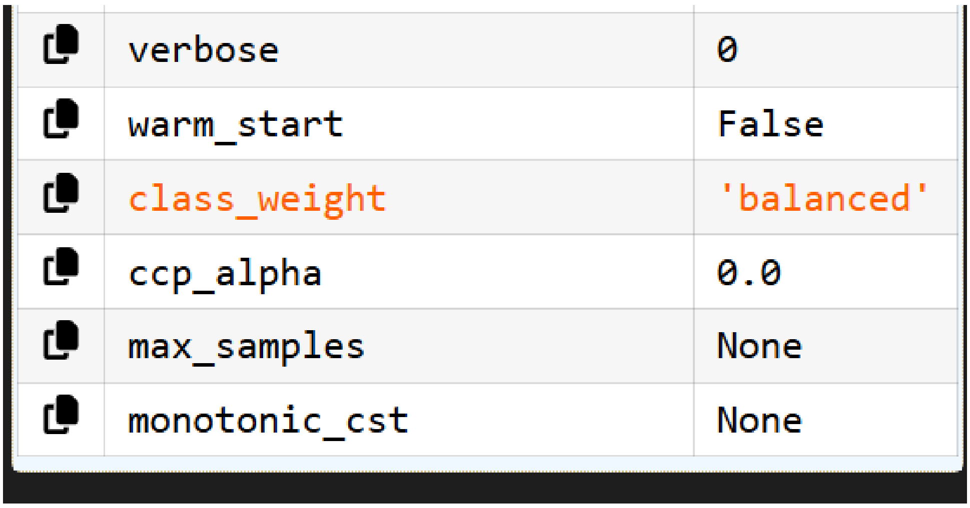


RandomForestClassifier

▼ Parameters

n_estimators	500
criterion	'gini'
max_depth	20
min_samples_split	2
min_samples_leaf	1
min_weight_fraction_leaf	0.0
max_features	'sqrt'
max_leaf_nodes	None
min_impurity_decrease	0.0
bootstrap	True
oob_score	False
n_jobs	-1
random_state	42

Figure 13: Random Forest Classifier Parameters.



verbose	0
warm_start	False
class_weight	'balanced'
ccp_alpha	0.0
max_samples	None
monotonic_cst	None

Figure 14: Continued....

XGBClassifier

Parameters

objective	'multi:softmax'
base_score	None
booster	None
callbacks	None
colsample_bylevel	None
colsample_bynode	None
colsample_bytree	0.8
device	None
early_stopping_rounds	None
enable_categorical	False
eval_metric	None
feature_types	None
feature_weights	None
gamma	None

Figure 15: XGBoost Classifier Parameters

n_jobs	-1
num_parallel_tree	None
random_state	42
reg_alpha	None
reg_lambda	None
sampling_method	None
scale_pos_weight	1
subsample	0.8
tree_method	None
validate_parameters	None
verbosity	None

Figure 16: Continued....

```
=== XGBoost ===
Accuracy: 0.7418339236521054
Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      1710
         1.0       0.20      0.19      0.20       166
         2.0       0.24      0.27      0.25       165
         3.0       0.21      0.22      0.22       174
         4.0       0.16      0.15      0.15       163
         5.0       0.23      0.22      0.23       163

    accuracy                           0.74      2541
   macro avg       0.34      0.34      0.34      2541
weighted avg       0.74      0.74      0.74      2541

Confusion Matrix:
 [[1710    0    0    0    0    0]
 [   0   32   36   39   29   30]
 [   0   26   44   39   28   28]
 [   0   39   33   39   36   27]
 [   0   35   37   33   24   34]
 [   0   27   33   34   33   36]]
```

Figure 17: Best Model XGBoost Clasification Report and Confusion Matrix

```
Accuracy: 0.4007220216606498

Classification Report:
              precision    recall  f1-score   support

        High       0.42      0.47      0.44       217
         Low       0.41      0.49      0.44       221
     Neutral       0.27      0.10      0.15       116

    accuracy                           0.40       554
   macro avg       0.37      0.35      0.35       554
weighted avg       0.38      0.40      0.38       554
```

Figure 18: Low Performance of RandomForest

```
✅ Best Parameters: {'colsample_bytree': 1.0, 'learning_rate': 0.2, 'max_depth': 10, 'n_estimators': 100, 'subsample': 1.0}
✅ Best Training Score: 0.7430836707152496
🎯 Test Accuracy (Tuned XGB): 0.7449822904368358
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00      1710
         1.0       0.23      0.24      0.24       166
         2.0       0.23      0.24      0.23       165
         3.0       0.21      0.20      0.20       174
         4.0       0.21      0.18      0.19       163
         5.0       0.22      0.25      0.23       163

    accuracy                           0.74      2541
   macro avg       0.35      0.35      0.35      2541
weighted avg       0.74      0.74      0.74      2541
```

Figure 19: XGBoost Metrices.

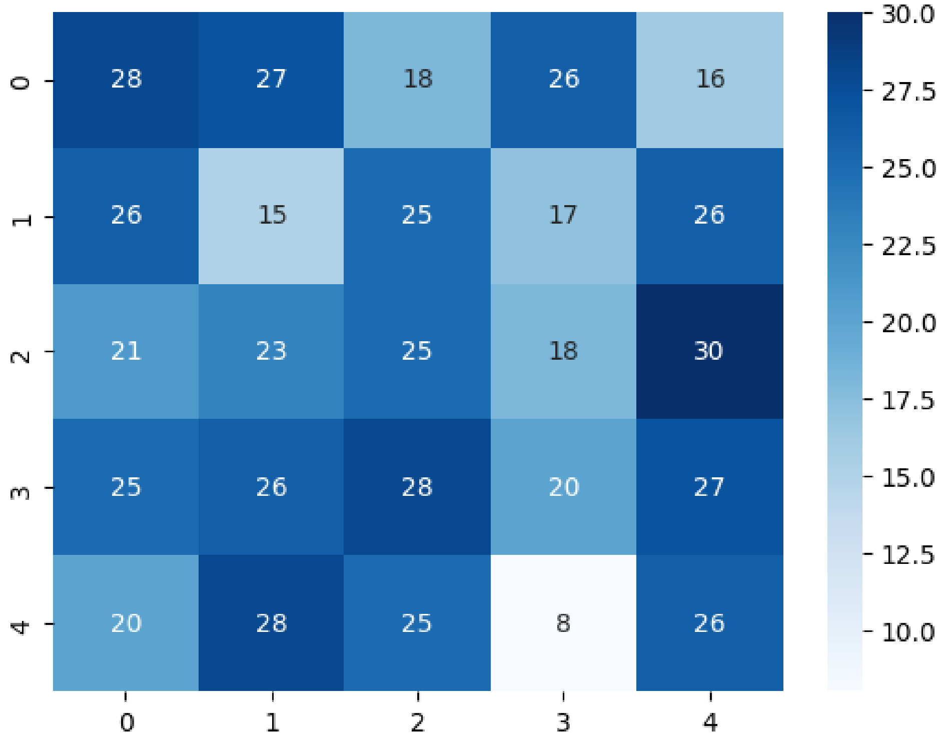


Figure 20: Classification Matrix

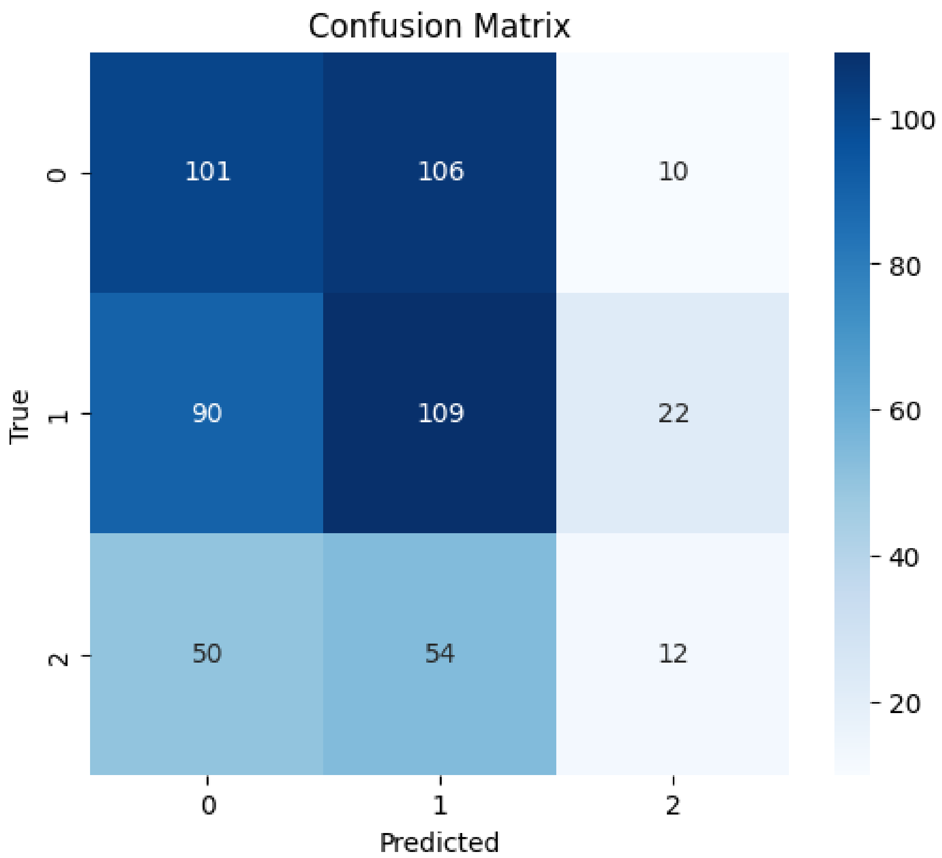


Figure 21: Confusion Matrix

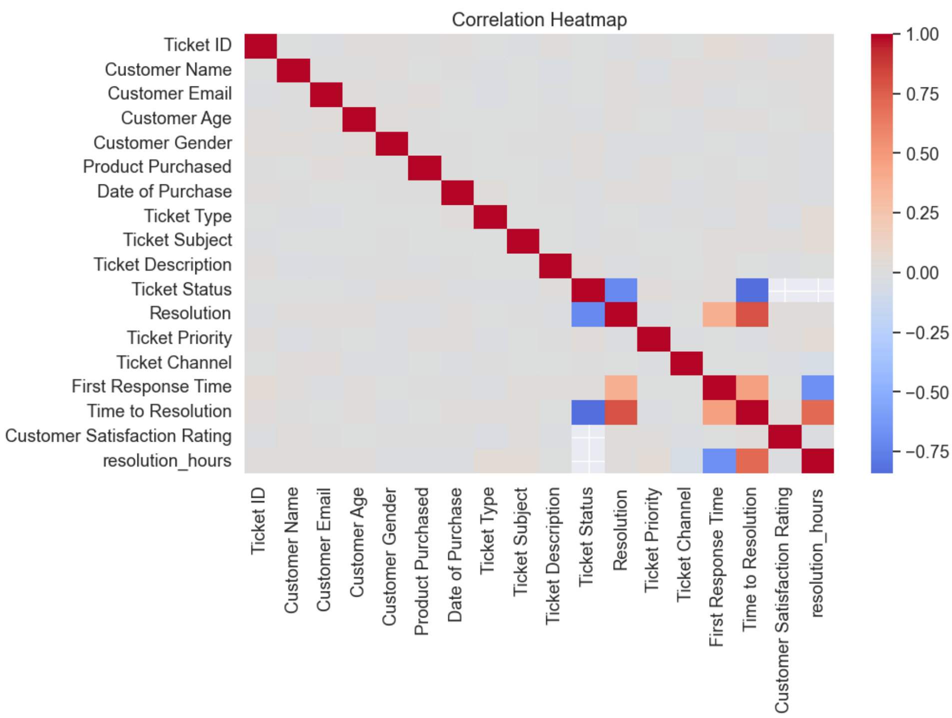


Figure 22: Correlation Heatmap for the best model.

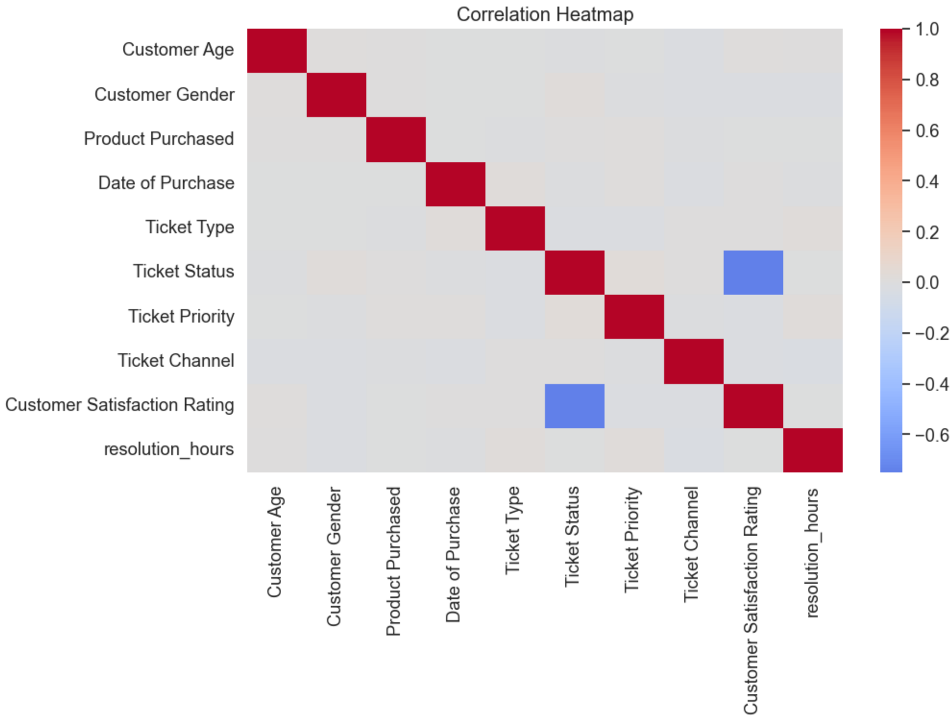


Figure 23: Correlation Heatmap

8.2 Feature Importance

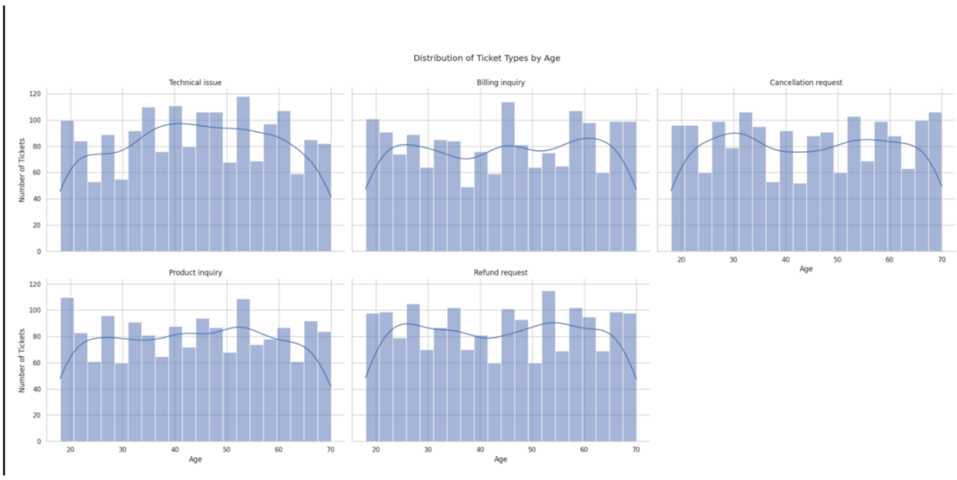


Figure 24: Distribution of Ticket types by Age.

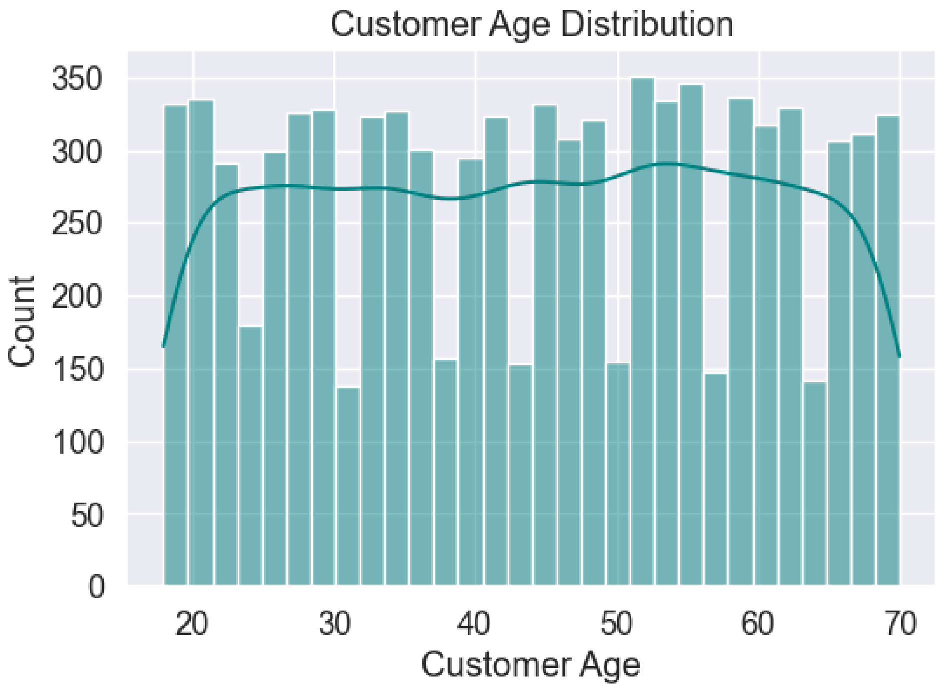


Figure 25: Customer Age Distribution with Counts

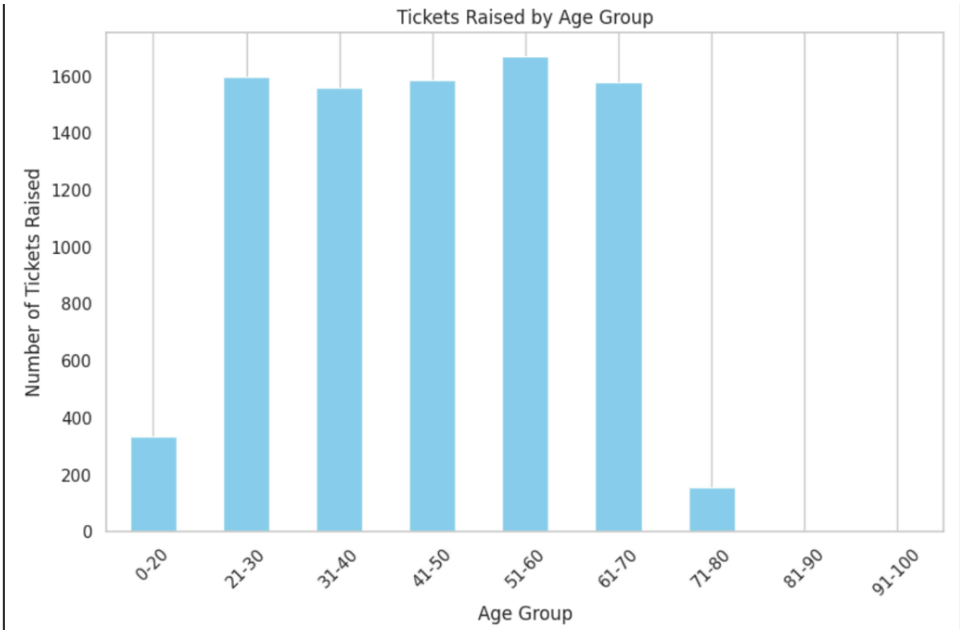


Figure 26: Tickets raised by Age Groups

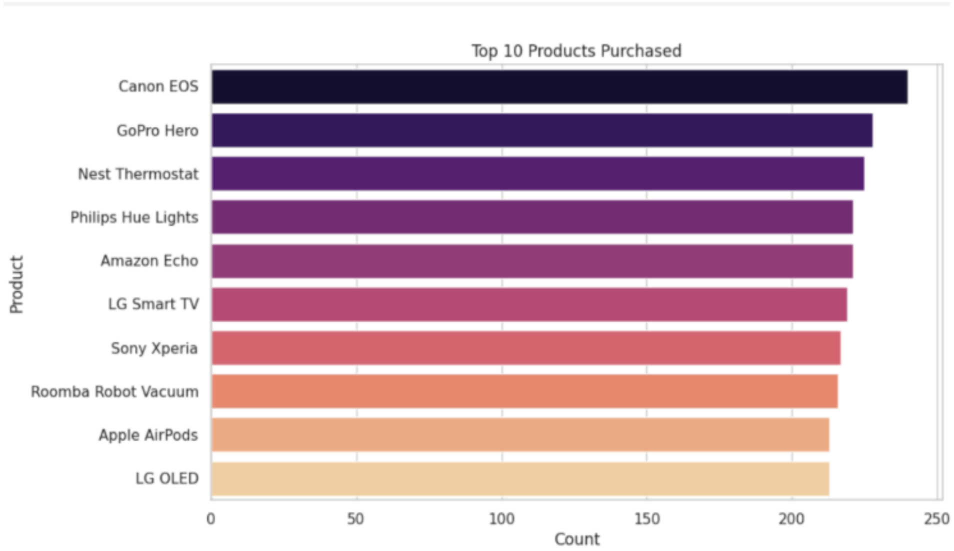


Figure 27: Top 10 Products Purchased with the Counts

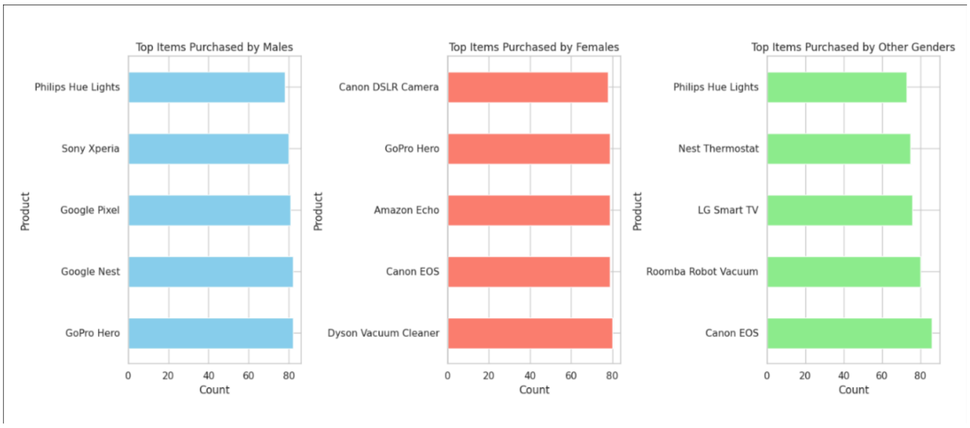


Figure 28: Top 10 Purchases Gender Wise

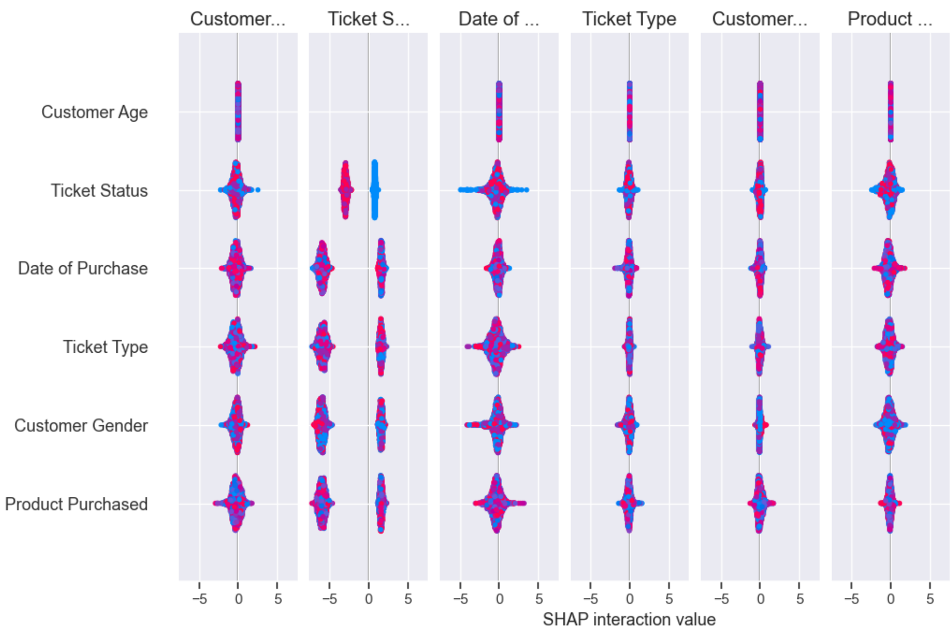


Figure 29: SHAP interaction values

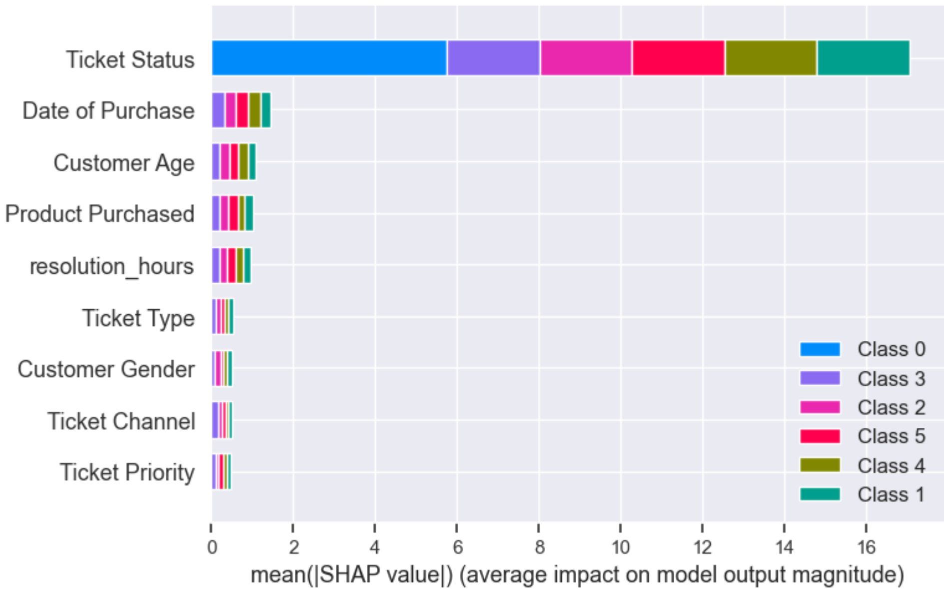


Figure 30: Average SHAp values impact on Model

9. SQL & Excel Integration

Example SQL queries used to pull labelled data and compute aggregates:

```sql
SELECT *
FROM tickets
WHERE customer_satisfaction_rating IS NOT NULL;

SELECT product_purchased, AVG(customer_satisfaction_rating) AS avg_sat, COUNT(*) as cnt
FROM tickets
GROUP BY product_purchased
ORDER BY cnt DESC;
```

Listing 2: Pull labelled tickets

An accompanying Excel dashboard (not included) can display month-over-month satisfaction and product-level averages.

9.1 Satisfaction Analysis

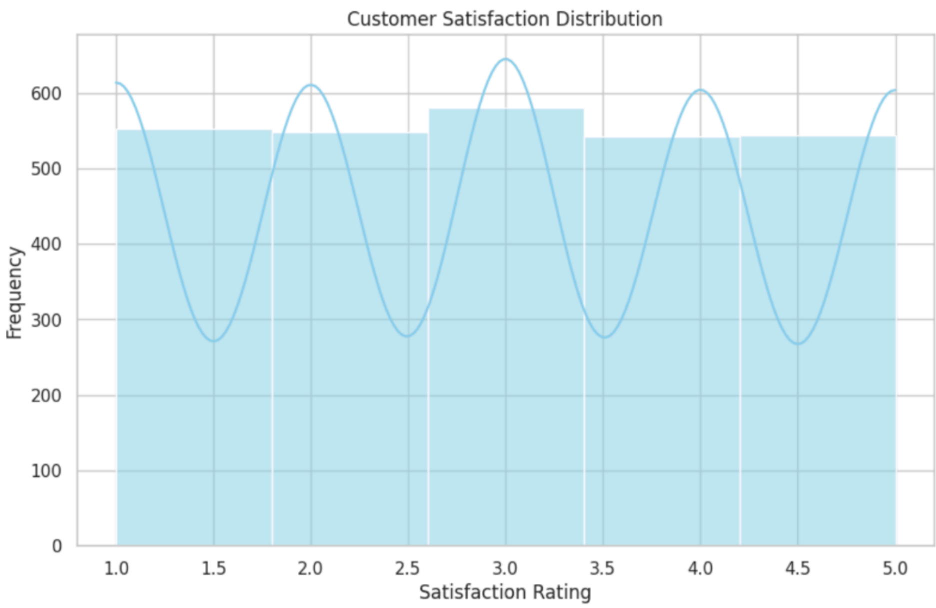


Figure 31: Customer Satisfaction Distribution

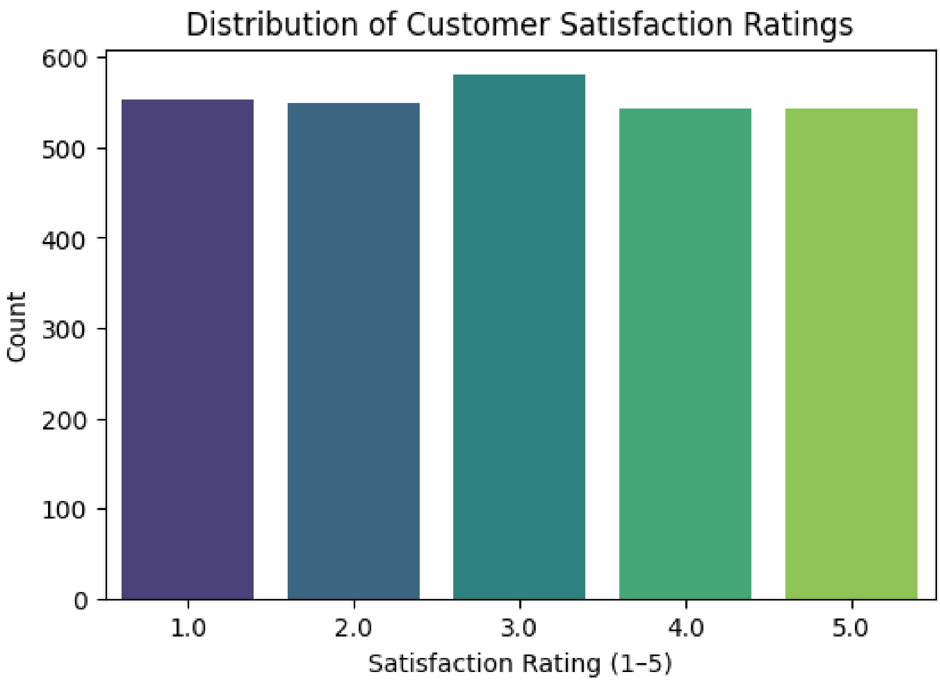


Figure 32: Customer Satisfaction Ratings(1-5) with Counts

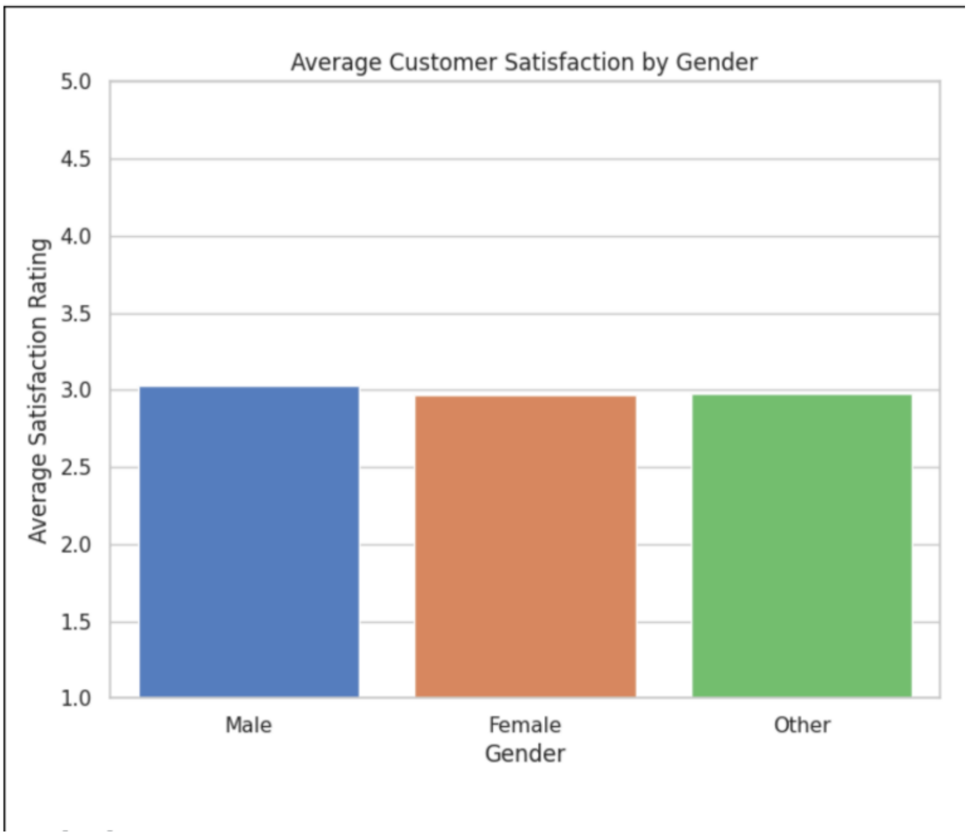


Figure 33: Average Customer Satisfaction by Gender

Figure 34: Customer Satisfaction by Gender

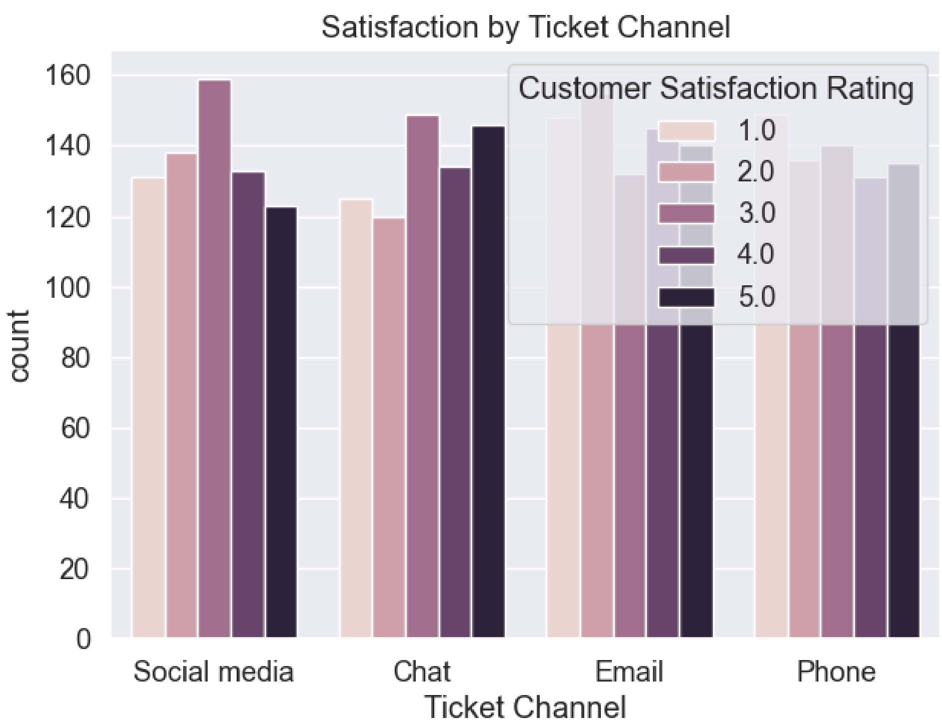


Figure 35: Satisfaction by Ticket Channel

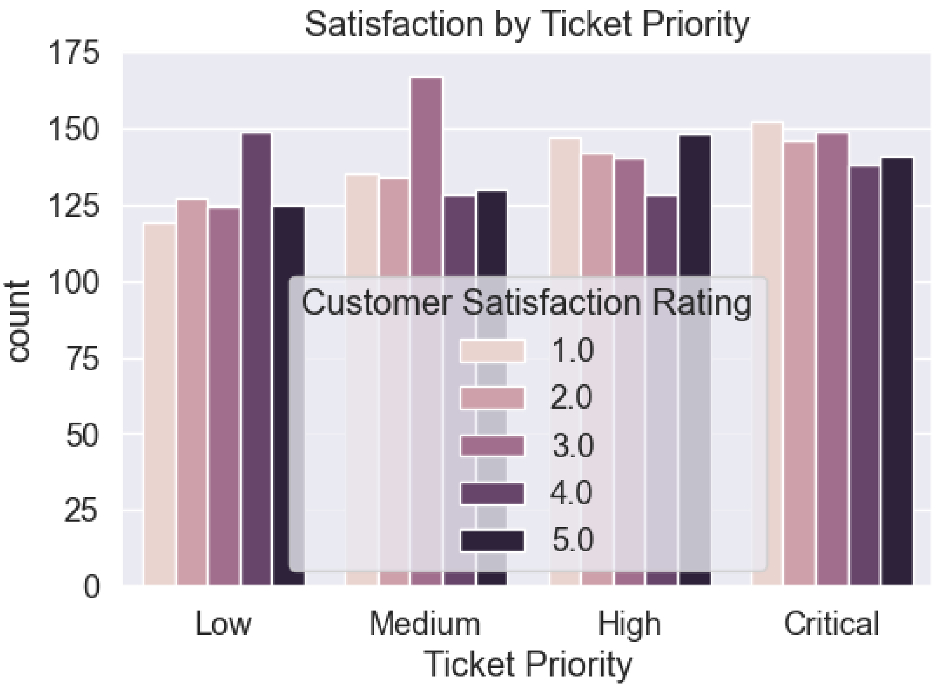


Figure 36: Satisfaction by Ticket Priority

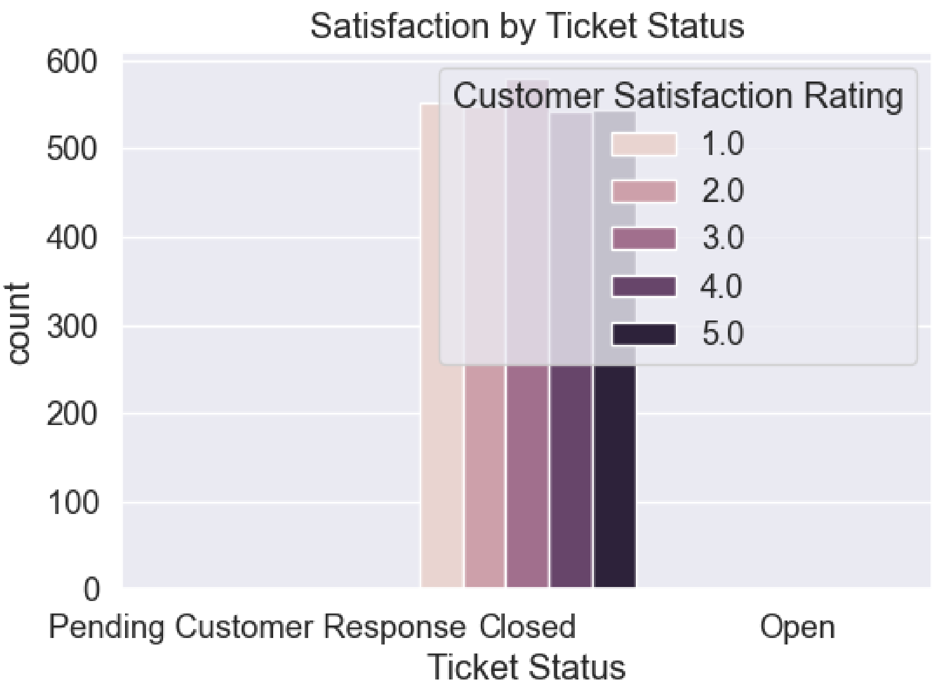


Figure 37: Satisfaction by Ticket Status

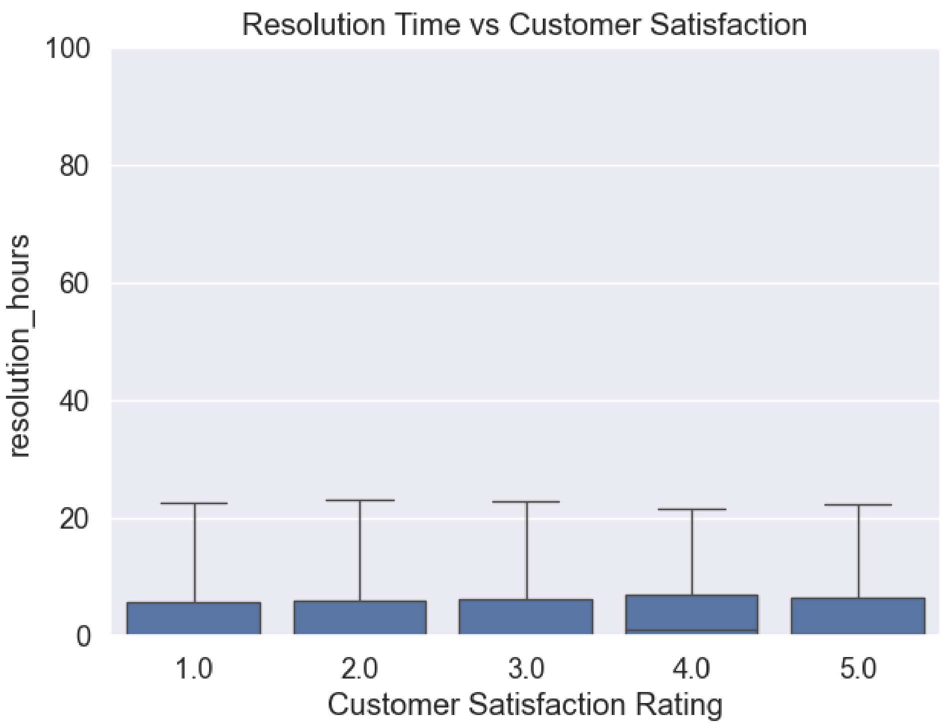


Figure 38: Resolution Time V/S Customer Satisfaction

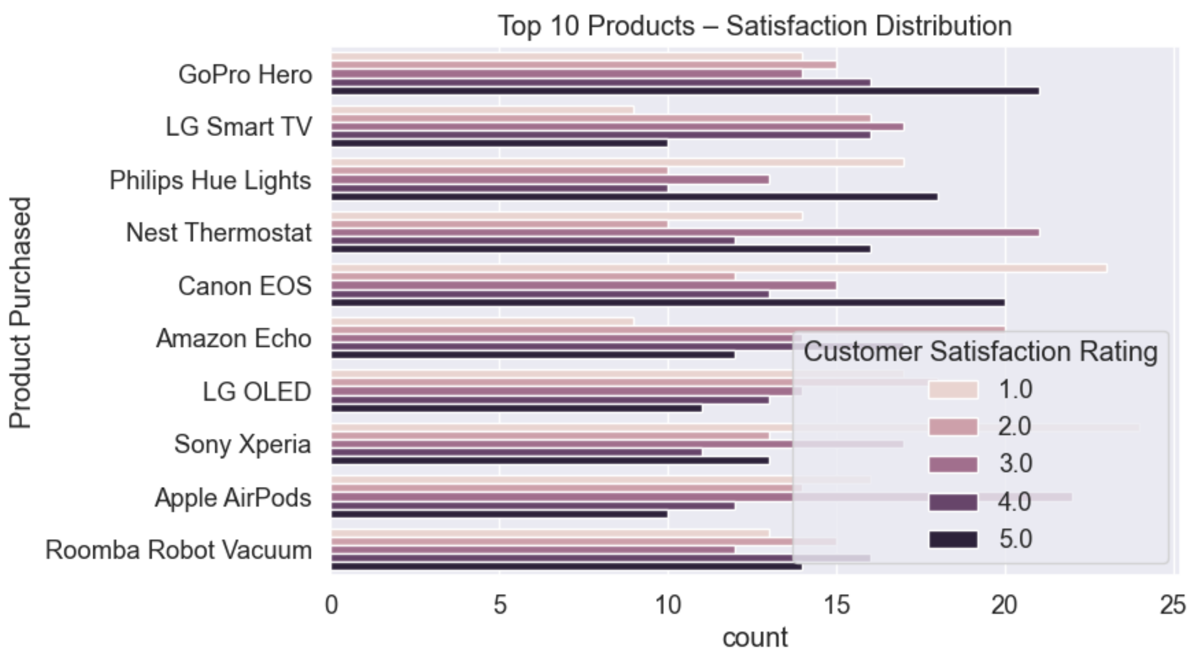


Figure 39: Top 10 Products-Satisfaction Distribution

9.2 CV Results and Some Pie Charts

```
…   Fitting 3 folds for each of 20 candidates, totalling 60 fits
    [CV] END colsample_bytree=1.0, learning_rate=0.05, max_depth=14, n_estimators=300, subsample=1.0; total time=  51.9s
    [CV] END colsample_bytree=1.0, learning_rate=0.05, max_depth=14, n_estimators=300, subsample=1.0; total time=  45.9s
    [CV] END colsample_bytree=1.0, learning_rate=0.05, max_depth=14, n_estimators=300, subsample=1.0; total time=  46.0s
    [CV] END colsample_bytree=0.7, learning_rate=0.05, max_depth=14, n_estimators=800, subsample=0.7; total time= 1.5min
    [CV] END colsample_bytree=0.7, learning_rate=0.05, max_depth=14, n_estimators=800, subsample=0.7; total time= 1.5min
    [CV] END colsample_bytree=0.7, learning_rate=0.05, max_depth=14, n_estimators=800, subsample=0.7; total time= 1.3min
    [CV] END colsample_bytree=1.0, learning_rate=0.05, max_depth=6, n_estimators=500, subsample=0.8; total time=  26.4s
    [CV] END colsample_bytree=1.0, learning_rate=0.05, max_depth=6, n_estimators=500, subsample=0.8; total time=  28.5s
    [CV] END colsample_bytree=1.0, learning_rate=0.05, max_depth=6, n_estimators=500, subsample=0.8; total time=  27.1s
    [CV] END colsample_bytree=0.7, learning_rate=0.1, max_depth=14, n_estimators=300, subsample=0.7; total time=  33.0s
    [CV] END colsample_bytree=0.7, learning_rate=0.1, max_depth=14, n_estimators=300, subsample=0.7; total time=  35.7s
    [CV] END colsample_bytree=0.7, learning_rate=0.1, max_depth=14, n_estimators=300, subsample=0.7; total time=  35.2s
    [CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=800, subsample=0.8; total time=  43.1s
    [CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=800, subsample=0.8; total time=  41.1s
    [CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=800, subsample=0.8; total time=  41.0s
    [CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=300, subsample=1.0; total time=  16.9s
    [CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=300, subsample=1.0; total time=  16.4s
    [CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=300, subsample=1.0; total time=  15.3s
    [CV] END colsample_bytree=0.8, learning_rate=0.05, max_depth=6, n_estimators=500, subsample=0.7; total time=  25.4s
    [CV] END colsample_bytree=0.8, learning_rate=0.05, max_depth=6, n_estimators=500, subsample=0.7; total time=  25.7s
    [CV] END colsample_bytree=0.8, learning_rate=0.05, max_depth=6, n_estimators=500, subsample=0.7; total time=  25.1s
```

Figure 40: CV Results with different Parameters

```
[CV] END colsample_bytree=0.7, learning_rate=0.1, max_depth=10, n_estimators=500, subsample=0.8; total time=  39.6s
[CV] END colsample_bytree=0.7, learning_rate=0.1, max_depth=10, n_estimators=500, subsample=0.8; total time=  37.9s
[CV] END colsample_bytree=0.7, learning_rate=0.1, max_depth=10, n_estimators=500, subsample=0.8; total time=  37.6s
...
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=800, subsample=0.7; total time=  41.2s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=6, n_estimators=800, subsample=0.7; total time=  43.4s
Best Parameters: {'subsample': 0.7, 'n_estimators': 300, 'max_depth': 14, 'learning_rate': 0.01, 'colsample_bytree': 0.7}
Best CV Accuracy: 0.3841961536439902
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Figure 41: Continued....

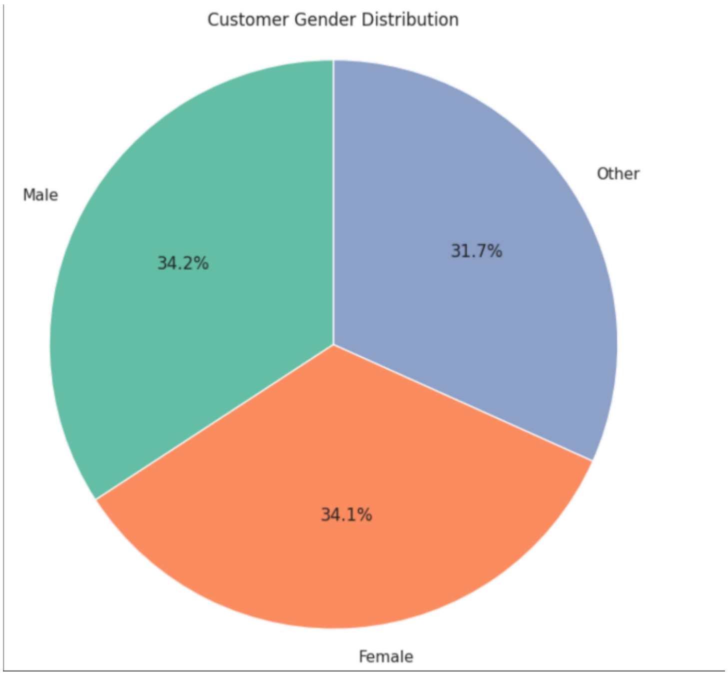


Figure 42: Customer Gender Distribution

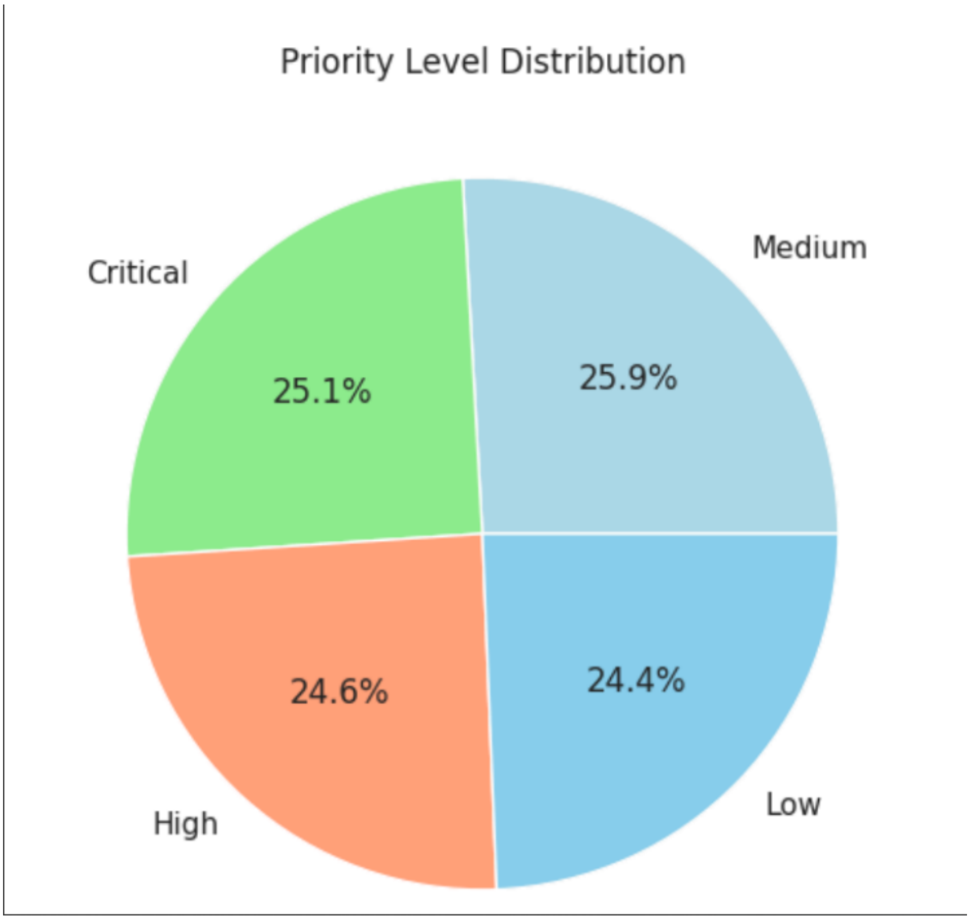


Figure 43: Priority Level Distribution

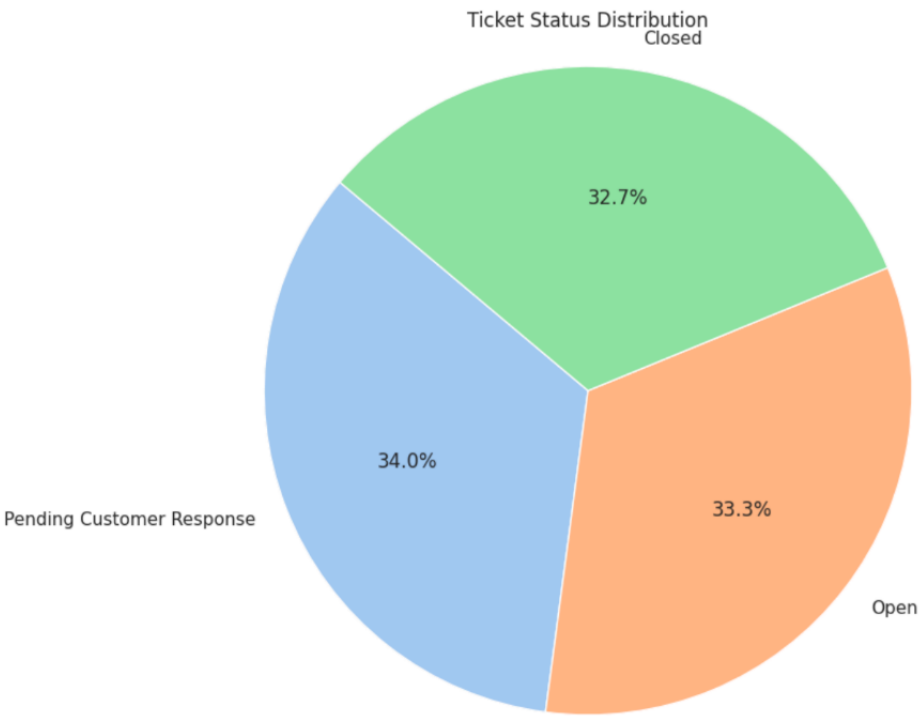


Figure 44: Ticket Status Distribution

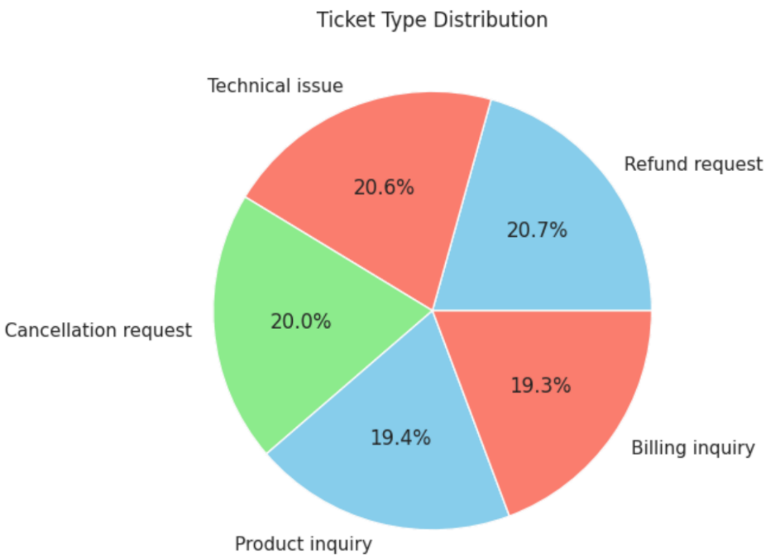


Figure 45: Ticket Type Distribution

10. Conclusion

In this internship-style project we developed a reproducible pipeline to predict customer satisfaction from ticket data. Key takeaways:

- Time-based features (response and resolution) and text sentiment are strong predictors.
- Tree-based models (XGBoost / Random Forest) perform well and provide explainability.
- Business action: prioritize reducing time-to-resolution for high-priority tickets.

11. Future Work

- Use transformer-based embeddings (Sentence-BERT) for richer text representation.
- Deploy model as a REST API with SHAP-based explanations for each prediction.
- Incorporate agent-level and SLA features for better operational insights.

12. References

- Scikit-learn: https://scikit-learn.org/
- Pandas Documentation: https://pandas.pydata.org/
- VADER sentiment: Hutto & Gilbert (2014).

A. Appendix A: Folder Structure

Use this project structure for Overleaf (or local):

```
CUSTOMER-SATISFACTION-DASHBOARD/

 data/
    customer_support_tickets.csv

 sql/
    checks.sql
    feature_views.sql
    schema.sql

 src/
    data_load.py
    evaluate.py
    model_train.py
    predict.py
    preprocess.py

 .gitignore
 app.py
 config.yaml
 customer_satisfaction_model.pkl
 LICENSE
 new_notebook.ipynb
 notebook.ipynb
 pyproject.toml
 README.md
 requirements.txt
 setup.cfg
```

Figure 46: Project Folder Structure of `CUSTOMER-SATISFACTION-DASHBOARD`

Each component serves a specific purpose:

- **data/** – Contains the primary dataset used for analysis and model training.
- **sql/** – Holds SQL scripts for schema setup, feature extraction, and data validation.
- **src/** – Core Python source files for preprocessing, training, evaluation, and prediction.
- `app.py` – Streamlit or Flask web app for dashboard deployment.
- `config.yaml` – Configuration file defining data paths, model parameters, and thresholds.

- `customer_satisfaction_model.pkl` – Serialized trained model.
- `requirements.txt` – Python dependencies for environment setup.
- `README.md` – Project overview and usage documentation.
- `pyproject.toml`, `setup.cfg` – Package and build configurations.
- `new_notebook.ipynb`, `notebook.ipynb` – Jupyter notebooks for exploration and analysis.
- `LICENSE` – License and usage rights.

B. Appendix B: Quick Commands

To compile locally using pdflatex:

```
pdflatex main.tex
bibtex main
pdflatex main.tex
pdflatex main.tex
```