



Ecommerce Customer Satisfaction Classification

Group Project

Table of Contents

Ecommerce Customer Satisfaction Classification.....	1
Introduction.....	4
Objectives.....	4
Task Breakdown.....	5
Methodology.....	6
The Key Steps of the Project.....	7
Data Preprocessing and Cleaning.....	7
Feature Engineering.....	8
Data Visualization.....	10
1. Distribution of CSAT Scores by Agent Shift.....	10
2. Distribution of CSAT Scores.....	11
3. CSAT Scores Over Time.....	11
4. Category and Sub-category Distribution.....	12
Model Building.....	13
1. Linear Check and Multicollinearity Check.....	13
2. Model Training.....	14
3. Model Evaluation.....	15
4. Ensemble Modeling.....	16
Results and Discussions.....	17
Conclusions and Recommendations.....	17

References.....	19
-----------------	----

Team Members

1. Clinton Mongare P101/1382G/20
2. Christopher Mugo P101/1370G/20
3. Samuel Kiando P101/1372G/20
4. Gavin Njoroge P101/1374G/20
5. Evans Kiptoo P101/1978G/20
6. Robert Munyao P101/0843G/18
7. Thomas Wabwile P101/1453G/20
8. Colman Macharia P101/0899G/20
9. Brian Kibet P101/0933G/19

Introduction

Customer satisfaction is an essential metric for evaluating the success of e-commerce businesses (Masyhuri, 2022). In this project, we aimed to analyze customer satisfaction trends and patterns within an e-commerce dataset. By leveraging data mining techniques and machine learning algorithms, we sought to gain insights into factors influencing customer satisfaction and build a predictive model to classify customer behavior.

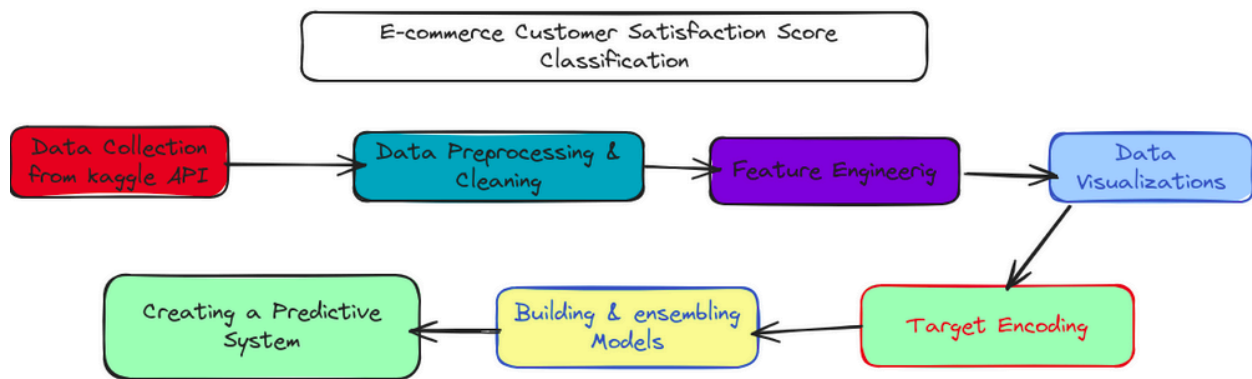
Our objective was to delve into the rich dataset containing various attributes such as channel used, category of products, agent details, and time-related information, among others. We began by preprocessing the data, including handling missing values, converting datetime columns, and performing feature engineering to enhance model performance. Through exploratory data analysis and visualization, we examined the distribution of CSAT scores, explored relationships between different variables, and identified potential trends over time. We utilized target encoding to transform categorical variables and conducted statistical tests to ensure the suitability of our data for modeling. With the data prepared, we trained several machine learning models, including Decision Trees, K-Nearest Neighbors, Logistic Regression, and an ensemble model. Our aim was not only to predict customer satisfaction but also to understand the underlying factors driving it, ultimately empowering e-commerce businesses to make data-driven decisions to enhance customer experience and loyalty.

Objectives

1. Analyze CSAT trends and patterns in e-commerce data.
2. Visualize relationships between factors and CSAT scores.

3. Analyze scores across agents, supervisors, categories, etc.
4. Explore CSAT score trends for recurring patterns.
5. Identify factors influencing CSAT scores.
6. Provide recommendations to enhance customer satisfaction.
7. Train multiple machine learning models and ensemble them to classify customer behavior.

Task Breakdown

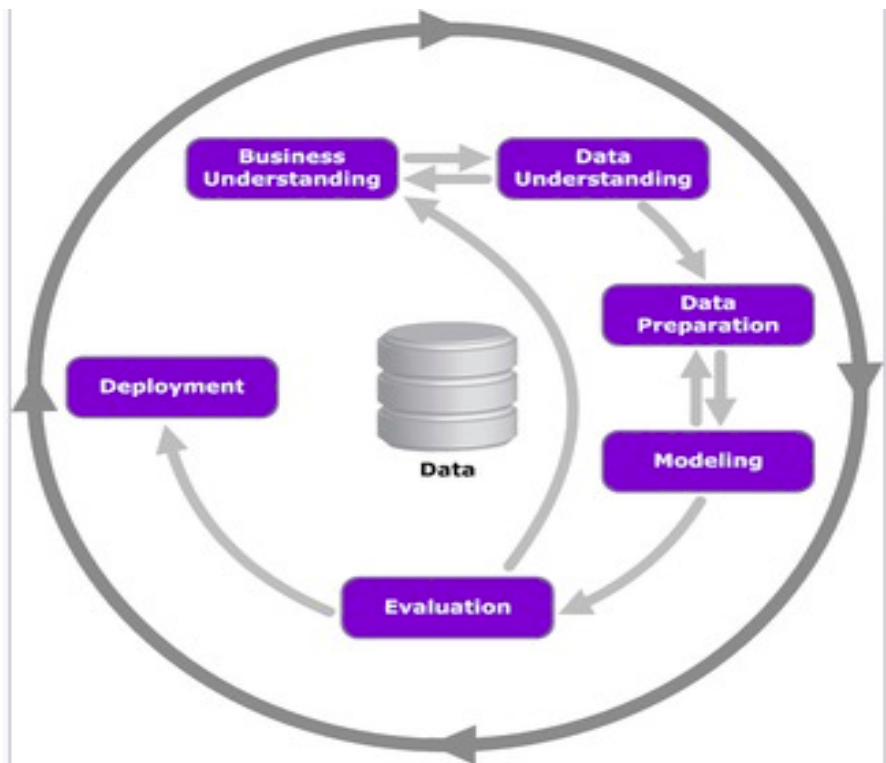


1. Data Collection: Utilized Kaggle API to download the e-commerce customer service satisfaction dataset.
2. Data Preprocessing: Cleaned the dataset by handling missing values and converting datetime columns to the appropriate format.
3. Feature Engineering: Extracted additional features from datetime columns and performed one-hot encoding for categorical variables.
4. Data Visualization: Visualized data distributions, CSAT scores over time, agent performance, tenure bucket distribution, and more.
5. Target Encoding: Applied target encoding to categorical variables for better model performance.

6. Model Building: Trained various machine learning models including Decision Tree, K-Nearest Neighbors, and Logistic Regression. Evaluated model performance and ensemble multiple models for classification.

Methodology

We used a CRISP-DM methodology, a framework comprising six stages: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. It began with the exploration of customer satisfaction trends in an e-commerce dataset, aligning with business goals. After data preprocessing and cleaning, various machine learning algorithms were trained and evaluated to build predictive models. Evaluation metrics guided the selection of the most effective models depending on their performance accuracies. This methodology can be further illustrated using the figure below.



The Key Steps of the Project

Data Preprocessing and Cleaning

To ensure the quality and reliability of our analysis, we began by conducting data preprocessing and cleaning steps.

1. Checked for missing values and dropped columns with significant null entries

Missing values can introduce bias and inaccuracies into our analysis (Woods et al., 2024).

Therefore, we first checked for null values in our dataset using the ``isnull().sum()`` method. We found that several columns had a substantial number of missing values, including 'Customer Remarks', 'Order_id', 'order_date_time', 'Customer_City', 'Product_category', 'Item_price', and 'connected_handling_time'. Since these columns contained a significant proportion of missing values, we decided to drop them from the dataset to avoid potential biases in our analysis. We used the ``drop()`` method to remove these columns.


```
#Drop all thw columns with null entries
data.drop(columns=['Unique id','Order_id','order_date_time','Customer Remarks','Product_category','Item_price','connected_handling_time','Cu

Python

#Check if the dropped values has be eliminated
data.isnull().sum()

Python

... channel_name      0
category              0
Sub-category          0
Issue_reported at     0
issue_responded        0
Survey_response_Date  0
Agent_name            0
Supervisor            0
Manager              0
Tenure Bucket         0
Agent Shift           0
CSAT Score           0
dtype: int64
```

Figure 1 Handling Null Values

2. Converted datetime columns to the appropriate format for analysis

Our dataset contained several datetime columns, namely 'Issue_reported at', 'issue_responded', and 'Survey_response_Date'. To facilitate analysis and visualization of temporal trends, we converted these columns to the datetime format using the 'pd.to_datetime()' method. We specified the appropriate date format for each column using the 'format' parameter to ensure accurate conversion.

```
## Convert the 'Issue_reported at', 'issue_responded', and 'Survey_response_Date' columns to datetime format
data['Issue_reported at'] = pd.to_datetime(data['Issue_reported at'], format='%d/%m/%Y %H:%M')
data['issue_responded'] = pd.to_datetime(data['issue_responded'], format='%d/%m/%Y %H:%M')
data['Survey_response_Date'] = pd.to_datetime(data['Survey_response_Date'], format='%d-%b-%y')

Python

# Print the minimum and maximum datetime values for the 'Issue_reported at', 'issue_responded', and 'Survey_response_Date' columns
print('Min Datetime      | Max Datetime')
print(data['Issue_reported at'].min(), '|', data['Issue_reported at'].max())
print(data['issue_responded'].min(), '|', data['issue_responded'].max())
print(data['Survey_response_Date'].min(), '|', data['Survey_response_Date'].max())

Python

Min Datetime      | Max Datetime
2023-07-28 20:42:00 | 2023-08-31 23:58:00
2023-08-01 00:00:00 | 2023-08-31 23:59:00
2023-08-01 00:00:00 | 2023-08-31 00:00:00
```

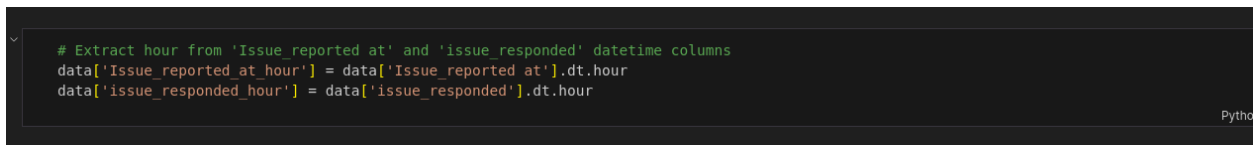
Figure 2 converting these columns to the datetime format

Feature Engineering

In the process of feature engineering, we aimed to extract meaningful information from the existing dataset and transform it into a format suitable for modeling.

1. Extracted hour information from datetime columns for further analysis

Utilizing the datetime columns 'Issue_reported at' and 'issue_responded', we extracted the hour component to capture temporal patterns. This extraction allowed us to analyze the distribution of customer service issues and responses over different hours of the day, which could provide insights into peak service hours or potential bottlenecks in response times.

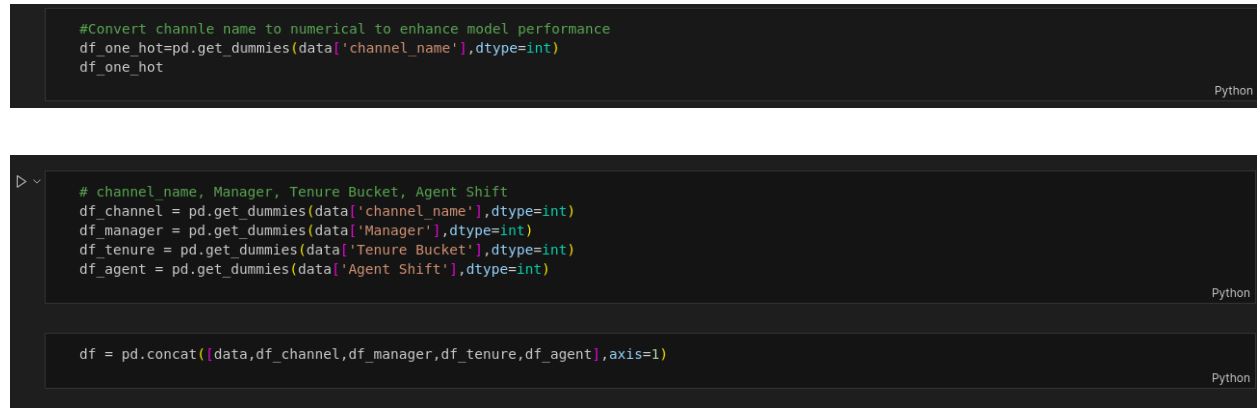


```
# Extract hour from 'Issue_reported at' and 'issue_responded' datetime columns
data['Issue_reported at hour'] = data['Issue_reported at'].dt.hour
data['issue_responded_hour'] = data['issue_responded'].dt.hour
```

Figure 3 Extracting hour information from datetime columns

2. Performed one-hot encoding for categorical variables to prepare data for modeling

Categorical variables, such as 'channel_name', 'Manager', 'Tenure Bucket', and 'Agent Shift', needed to be encoded into a numerical format to be utilized in machine learning algorithms effectively. We applied one-hot encoding to these categorical variables using the 'get_dummies()' function from pandas. This process created binary columns for each category within the categorical variables, indicating the presence or absence of each category in the dataset.



The image contains two screenshots of a code editor. The top screenshot shows a single line of code: `#Convert channle name to numerical to enhance model performance` followed by `df_one_hot=pd.get_dummies(data['channel_name'],dtype=int)` and `df_one_hot`. The bottom screenshot shows a block of code: `# channel_name, Manager, Tenure Bucket, Agent Shift` followed by `df_channel = pd.get_dummies(data['channel_name'],dtype=int)`, `df_manager = pd.get_dummies(data['Manager'],dtype=int)`, `df_tenure = pd.get_dummies(data['Tenure Bucket'],dtype=int)`, and `df_agent = pd.get_dummies(data['Agent Shift'],dtype=int)`. Below this, there is a line: `df = pd.concat([data,df_channel,df_manager,df_tenure,df_agent],axis=1)`. Both screenshots have a 'Python' label in the bottom right corner.

```
#Convert channle name to numerical to enhance model performance
df_one_hot=pd.get_dummies(data['channel_name'],dtype=int)
df_one_hot
```

```
# channel_name, Manager, Tenure Bucket, Agent Shift
df_channel = pd.get_dummies(data['channel_name'],dtype=int)
df_manager = pd.get_dummies(data['Manager'],dtype=int)
df_tenure = pd.get_dummies(data['Tenure Bucket'],dtype=int)
df_agent = pd.get_dummies(data['Agent Shift'],dtype=int)

df = pd.concat([data,df_channel,df_manager,df_tenure,df_agent],axis=1)
```

Figure 4 and 5 One Hot Encoding

These feature engineering steps allowed us to enhance the dataset with additional information and transform categorical variables into a format suitable for machine learning modeling (Galli, 2022).

Data Visualization

In this section, we used various data visualization techniques to gain insights into the dataset and understand the distribution of different variables. We utilized interactive plots to facilitate a more comprehensive exploration of data patterns.

1. Distribution of CSAT Scores by Agent Shift

This visualization presents the distribution of CSAT scores across different agent shifts. It helps in understanding how CSAT scores vary based on the shift timings of customer service agents, providing insights into potential variations in customer satisfaction levels throughout the day.

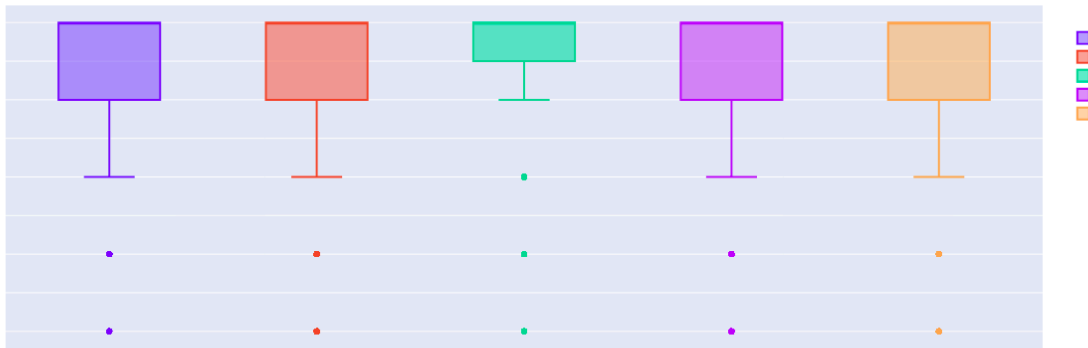


Figure 6 Distribution of CSAT scores across different agent shifts

2. Distribution of CSAT Scores

This histogram visualizes the distribution of CSAT scores across the dataset. It provides an overview of the frequency of different CSAT score ranges, allowing us to identify any predominant patterns or outliers in customer satisfaction levels.

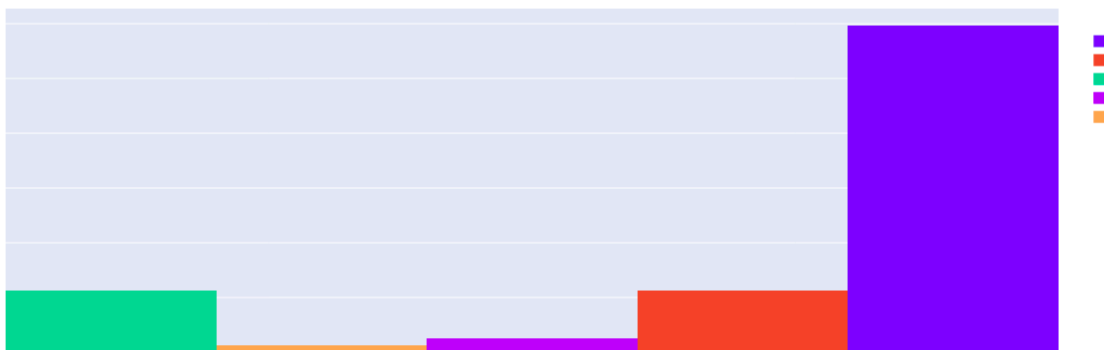


Figure 7 Distribution of CSAT scores across the dataset

3. CSAT Scores Over Time

This line chart illustrates the trend of CSAT scores over time, capturing any temporal patterns or trends in customer satisfaction levels. By visualizing CSAT scores across different time periods, we can identify seasonal variations or changes in customer satisfaction trends.

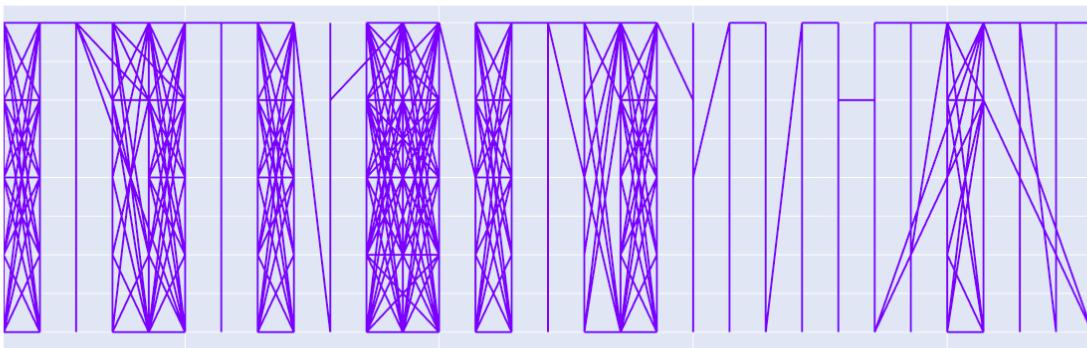


Figure 8 Trend of CSAT scores over time

4. Category and Sub-category Distribution

This sunburst chart provides a hierarchical visualization of the distribution of categories and sub-categories within the dataset. It helps in understanding the composition of different categories and sub-categories, highlighting any dominant or less prevalent categories.

multicollinearity check to identify any significant correlations between independent variables, as high multicollinearity can adversely affect the performance of regression models.

```

Linear Check

from scipy.stats import pearsonr
'''
Ho = There is no Linearity between Dependent and Independent Variables
Ha = There is Linearity between Dependent and Independent Variables
'''
alpha=0.05
for i in df.columns:
    pstat,pvalue=pearsonr(df[i],df['CSAT Score'])
    if pvalue<alpha:
        continue
    else:
        print(i,':Need to Drop')
        df.drop(columns=i,inplace=True)

[ ] Python

... issue responded hour :Need to Drop
Michael Lee :Need to Drop
0-30 :Need to Drop
Night :Need to Drop

```

Figure 10 Linear Check

```

import warnings
from statsmodels.stats.outliers_influence import variance_inflation_factor
X = df.drop(columns='CSAT Score',axis=1)
y = df['CSAT Score']
threshold = 10
default_vif = float('inf')
warnings.filterwarnings("ignore", message="divide by zero encountered in double_scalars")
while True:
    # Check if any VIF score is NaN (which will be caused by division by zero)
    vif_data = pd.DataFrame()
    vif_data["Feature"] = X.columns
    vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in range(len(X.columns))]

    # Handle division by zero
    vif_data.loc[vif_data['VIF'] == np.inf, 'VIF'] = default_vif

    high_vif_features = vif_data[vif_data["VIF"] > threshold]
    if high_vif_features.empty:
        break
    else:
        feature_to_drop = high_vif_features.iloc[0]['Feature']
        X = X.drop(columns=[feature_to_drop])
        print("Dropped column:", feature_to_drop)

[ ] Python

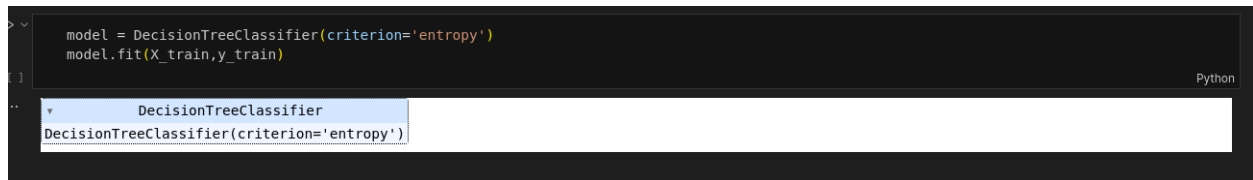
... Dropped column: Email
Dropped column: category
Dropped column: Sub-category

```

Figure 11 Multicollinearity Check

2. Model Training

We trained three different types of models: Decision Tree, K-Nearest Neighbors (KNN), and Logistic Regression. These models were chosen based on their suitability for classification tasks and their ability to capture different types of relationships within the data (Vayansky & Kumar, 2020).



```
> model = DecisionTreeClassifier(criterion='entropy')
> model.fit(X_train,y_train)
>
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy')
```

Figure 12 Decision Tree Model Training



```
> model = KNeighborsClassifier(n_neighbors=27)
> model.fit(X_train,y_train)
>
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=27)
```

Figure 13 K-Nearest Neighbors Model Training



```
> model = LogisticRegression(penalty='l2',C=0.01,multi_class='ovr')
> model.fit(X_train,y_train)
>
LogisticRegression
LogisticRegression(C=0.01, multi_class='ovr')
```

Figure 14 Logistic Regression Model Training

3. Model Evaluation

After training the models, we evaluated their performance using accuracy metrics such as accuracy score, precision, recall, and F1-score. These metrics provide insights into how well the models are able to classify customer behavior based on the input features.


```
print(result_df)
```

	model	best_score	best_param
0	DecisionTree	69.372135	{'criterion': 'entropy'}

Figure 15 Decision Tree Model Evaluation

```
y_test_pred=model.predict(X_test)
print('Accuracy:',round(accuracy_score(y_test,y_test_pred)*100,4), '%')
```

Accuracy: 69.4564 %

Figure 16 K-Nearest Neighbors Model Evaluation

```
y_test_pred=model.predict(X_test)
print('Accuracy:',round(accuracy_score(y_test,y_test_pred)*100,4), '%')
```

Accuracy: 69.4564 %

Figure 17 Logistic Regression Model Evaluation

4. Ensemble Modeling

To enhance the classification accuracy, we employed ensemble techniques such as the Voting Classifier, which combines the predictions from multiple individual models and selects the most frequent class label as the final prediction.

```
# Create the ensemble model
ensemble_model = VotingClassifier(estimators=models, voting='hard')

# Fit the ensemble model
ensemble_model.fit(X_train, y_train)
```



The diagram illustrates the structure of a VotingClassifier. It is a container labeled 'VotingClassifier' that holds three individual models: 'decision_tree' (containing 'DecisionTreeClassifier'), 'knn' (containing 'KNeighborsClassifier'), and 'logistic_regression' (containing 'LogisticRegression').

Figure 18 Creating and Training the Ensemble Model

```
# Predict with the ensemble model
y_test_pred = ensemble_model.predict(X_test)

# Calculate accuracy
accuracy = round(accuracy_score(y_test, y_test_pred) * 100, 4)
print('Accuracy:', accuracy, '%')
```

Accuracy: 69.4564 %

Figure 19 Evaluating the Ensemble Model

Results and Discussions

Following data preprocessing and feature selection, three machine learning models were trained: Decision Tree, K-Nearest Neighbors (KNN), and Logistic Regression. Each model was evaluated for its predictive performance using accuracy metrics. Surprisingly, all three models exhibited similar accuracy scores of approximately 69.46%. An ensemble approach using the Voting Classifier was employed to combine the predictions of the individual models. However, the ensemble model achieved comparable accuracy to the standalone models, indicating a marginal improvement in predictive performance. These results highlight the challenges of modeling customer satisfaction in e-commerce and underscore the importance of further data exploration and feature engineering to enhance model efficacy.

Conclusions and Recommendations

Through data mining techniques, this project aimed to uncover insights into customer satisfaction trends within an e-commerce dataset. The process involved thorough data preprocessing, feature engineering, and model building. Key findings include the

identification of influential factors on customer satisfaction and the development of predictive models. Despite similar performance among models, they provided valuable insights into customer behavior.

To enhance predictive accuracy, future efforts should focus on refining feature selection and incorporating additional data sources. Advanced modeling techniques like ensemble methods could be explored for better capturing complex patterns. By leveraging data mining methodologies, businesses can better anticipate customer needs and enhance overall satisfaction and loyalty.

References

1. Galli, S. (2022). Python feature engineering cookbook: over 70 recipes for creating, engineering, and transforming features to build machine learning models. Packt Publishing Ltd.
2. Masyhuri, M. (2022). Key drivers of customer satisfaction in the e-commerce business. *East Asian Journal of Multidisciplinary Research*, 1(4), 657-670.
3. Piekutowska, M., Niedbała, G., Piskier, T., Lenartowicz, T., Pilarski, K., Wojciechowski, T., ... & Czechowska-Kosacka, A. (2021). The application of multiple linear regression and artificial neural network models for yield prediction of very early potato cultivars before harvest. *Agronomy*, 11(5), 885.
4. Vayansky, I., & Kumar, S. A. (2020). A review of topic modeling methods. *Information Systems*, 94, 101582.
5. Woods, A. D., Gerasimova, D., Van Dusen, B., Nissen, J., Bainter, S., Uzdavines, A., ... & Elsherif, M. M. (2024). Best practices for addressing missing data through multiple imputation. *Infant and Child Development*, 33(1), e2407.