YILDIZ TECHNICAL UNIVERSITY

FACULTY OF METALLURGICAL AND MATERIALS

DEPARTMENT OF MATHEMATICAL ENGINEERING

MULTIDISCIPLINARY DESIGN PROJECT

DATABASE INTRUSION DETECTION SYSTEM

Thesis Supervisor: Prof. Dr. Ayla ŞAYLI

19058006, ONUR TURAN

Istanbul, January 2025

1. INTRODUCTION

In this project, it is planned to develop an Anomaly-Based Intrusion Detection System (IDS) to detect SQL Injection (SQLi) attacks, which are a significant threat in the field of database security. SQLi attacks can cause malicious users to send harmful SQL queries to the database to steal data, modify data, or prevent the system from working. This project aims to secure users' data by protecting database systems from such attacks.

1.1 Aim and scope

The main objective of the project is to develop a model that effectively detects SQL Injection attacks. The project covers the following areas:

- **Algorithm Development**: A machine learning model that learns the normal and abnormal behavior of SQL queries will be created.
- **Data Analysis**: Regex and string analysis methods will be used to analyze the characteristics of database queries.
- **Testing and Evaluation**: Different datasets will be used to test the effectiveness of the developed model and evaluate the results.

1.2 Used Tools

The following tools and technologies will be used in the development of the project:

- **Programming Languages**: Python is preferred for data analysis and machine learning applications.
- **Libraries**:
    - **Pandas**: For data analysis and manipulation.
    - **NumPy**: For numerical calculations.
    - **Scikit-learn**: For implementing supervised learning algorithms.
    - **Database Management System**: MSSQL
    - **Regex**: For detecting anomaly patterns in SQL queries.
    - **Visualization Tools**: Matplotlib and Seaborn will be used for data visualization.

2 DATA DEFINITION

- The dataset to be utilized within the scope of this project will be meticulously prepared to encompass both normal and abnormal SQL queries. The richness and diversity of the dataset are paramount for the accurate training and evaluation of the anomaly-based Intrusion Detection System (IDS).

Normal queries refer to SQL queries that are widely used in real-world applications and are considered secure. These queries follow standard SQL practices and do not exhibit any malicious behavior. Examples of normal queries include:

- Data retrieval queries (e.g., SELECT * FROM users WHERE id = 1)
- Data insertion queries (e.g., INSERT INTO users (name, email) VALUES ('John Doe', 'john@example.com'))
- Data update queries (e.g., UPDATE users SET email = 'john_new@example.com' WHERE id = 1)
- Data deletion queries (e.g., DELETE FROM users WHERE id = 1)

Abnormal queries are those created by malicious users with the intent to exploit vulnerabilities in the database system. These queries often include examples of SQL Injection (SQLi) attacks, which can lead to unauthorized data access, data modification, or denial of service. Examples of abnormal queries include:

- SQL Injection attacks using tautologies (e.g., SELECT * FROM users WHERE id = 1 OR 1=1)
- Union-based SQL Injection (e.g., SELECT * FROM users WHERE id = 1 UNION SELECT username, password FROM admin)
- Blind SQL Injection (e.g., SELECT * FROM users WHERE id = 1 AND IF(1=1, SLEEP(5), 0))
- Error-based SQL Injection (e.g., SELECT * FROM users WHERE id = 1 AND (SELECT 1 FROM (SELECT COUNT(*), CONCAT((SELECT database()), 0x3a, FLOOR(RAND(0)*2))x FROM information_schema.tables GROUP BY x)a))

The dataset will be enriched with a variety of such queries to ensure the model is trained to detect a wide range of SQLi attack patterns.

Data Visualization

To provide a comprehensive understanding of the dataset, visualizations will be created to illustrate the distribution of SQL query components. The following figures demonstrate the distribution of key SQL components (SELECT, UNION, OR, AND) within the dataset.

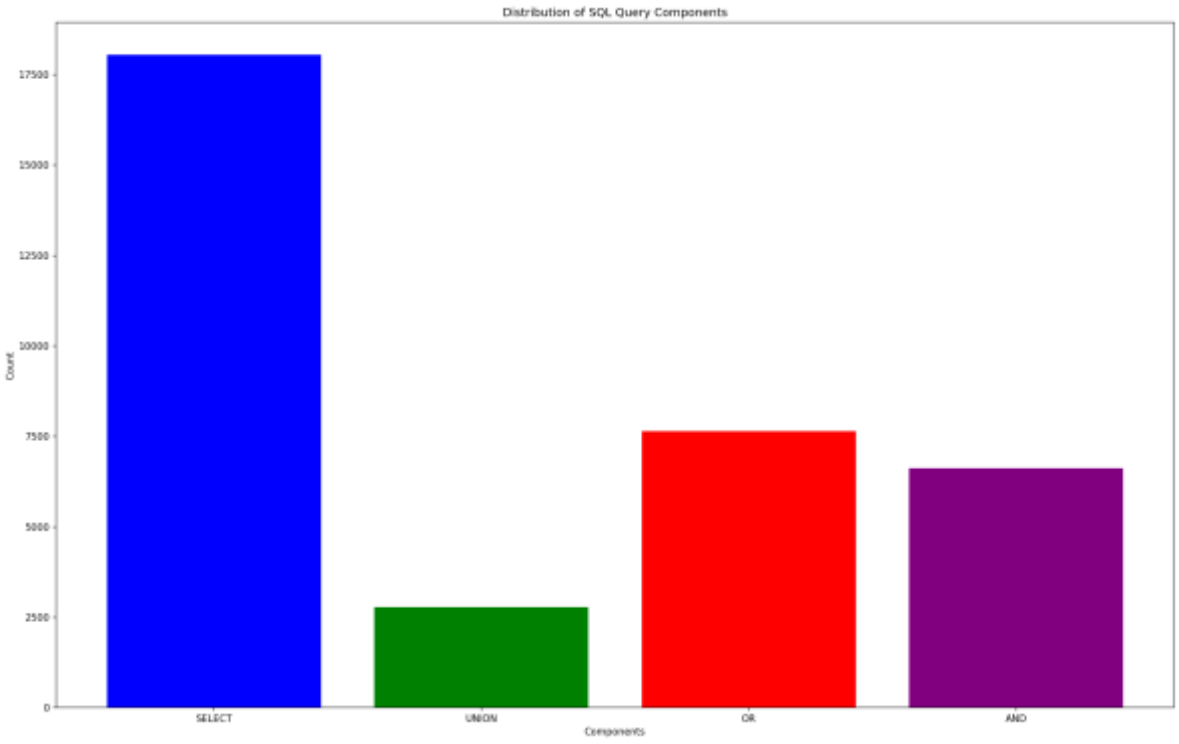
This Figure2.1 shows the count of each SQL component in the dataset, highlighting the prevalence of different query structures.

Figure 2.1

Query	Label	SELECT	UNION	OR	AND
" or pg_sleep (__TIME__) --	1	0	0	1	0
create user name identified by pass123 temporary tablespace temp default tablespace users;	1	0	0	1	0
(SELECT DISTINCT (table_name) FROM (SELECT DISTINCT (table_name) , ROWNUM AS LIMIT FROM sys.all_table	1	1	0	0	1
select * from users where id = '1' or @ @1 = 1 union select 1,version ( ) -- 1"	1	1	1	1	0
select * from users where id = 1 or 1#" (union select 1,version ( ) -- 1	1	1	1	1	0

This figure 2.2 provides a visual representation of the first five rows of the dataset, showcasing both normal and abnormal queries along with their respective labels and component presence.

Figure 2.2



3. Data Visualization Before Data Preprocessing

In this section, various visualizations of the columns in the dataset are presented before any data preprocessing steps are applied. These visualizations are crucial for understanding the general characteristics of the dataset and identifying any potential anomalies. The visualizations were created using Python libraries such as numpy, seaborn, pandas, and matplotlib.

3.1 Graph of Target Column, Column Name

This subsection focuses on the visualization of the target column, which is the `Label` column. The `Label` column indicates whether the SQL queries are malicious (1) or benign (0). Various types of plots are used to visualize the distribution and characteristics of this column.

3.1.1 Scatter Plot Graph of Label

The scatter plot graph is used to visualize the distribution of the `Label` column. This type of graph is useful for identifying how the data points are spread out and for detecting any potential anomalies.

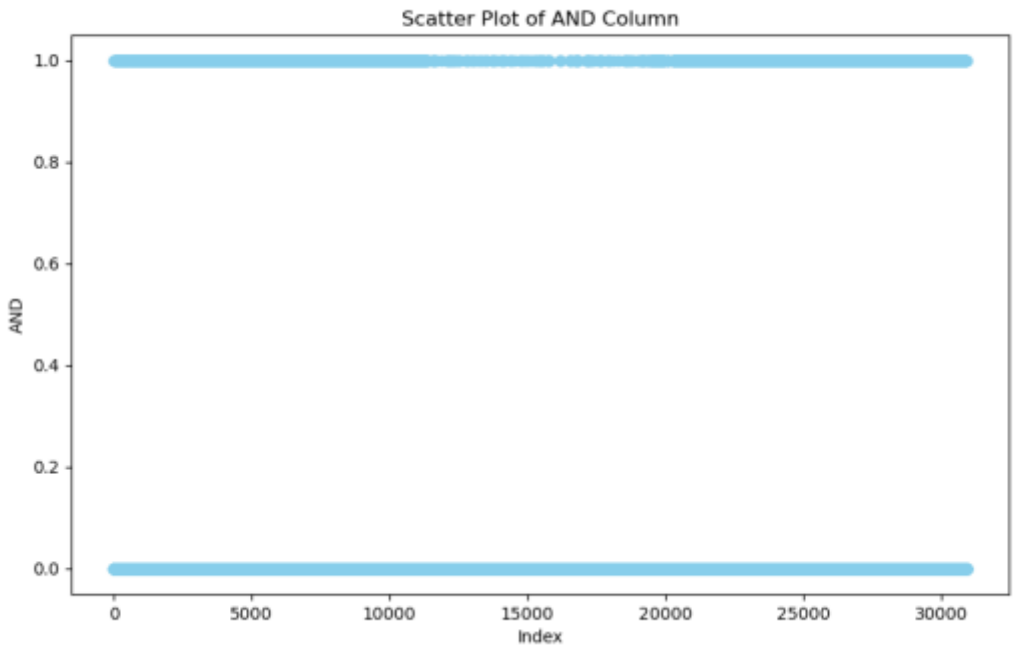
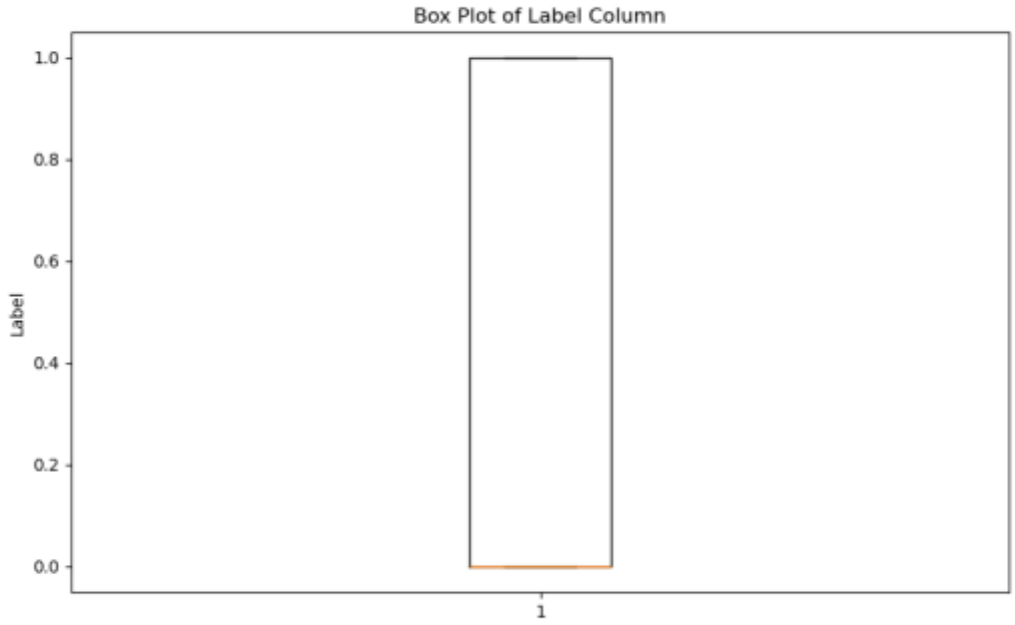


Figure 3.1.1

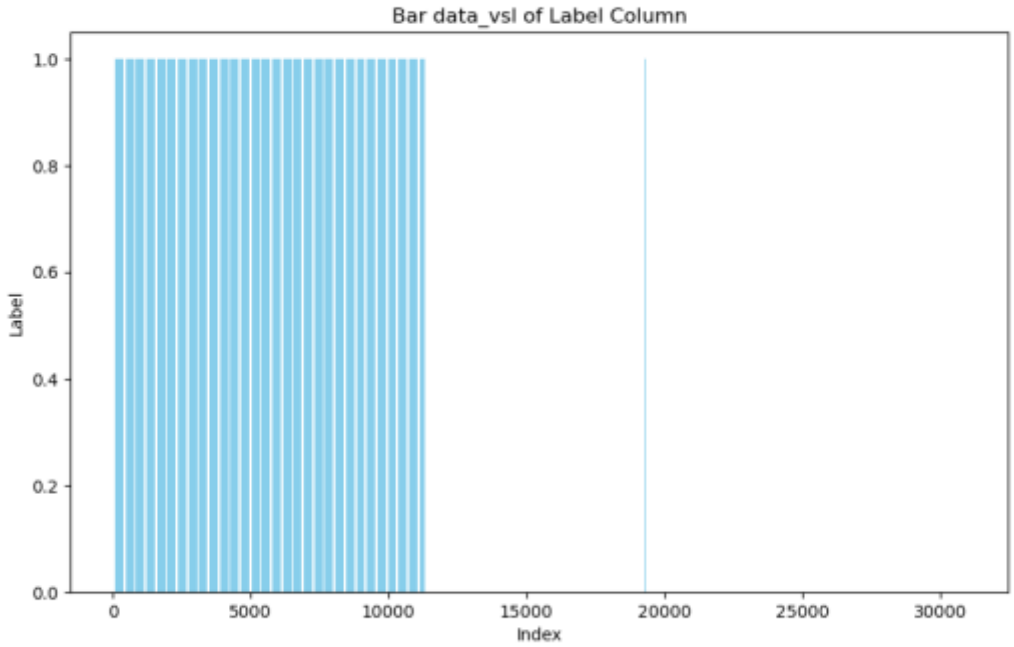
3.1.2 Box Plot Graph of Label

The box plot graph is used to visualize the distribution and potential outliers in the `Label` column. This graph is helpful for understanding the central tendency and spread of the data.



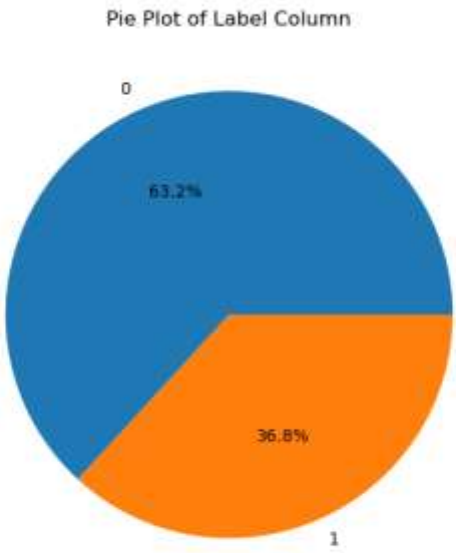
3.1.3 Bar Graph of Label

The bar graph is used to visualize the frequency distribution of the `Label` column. This graph helps in understanding which categories are more prevalent and which are less represented.



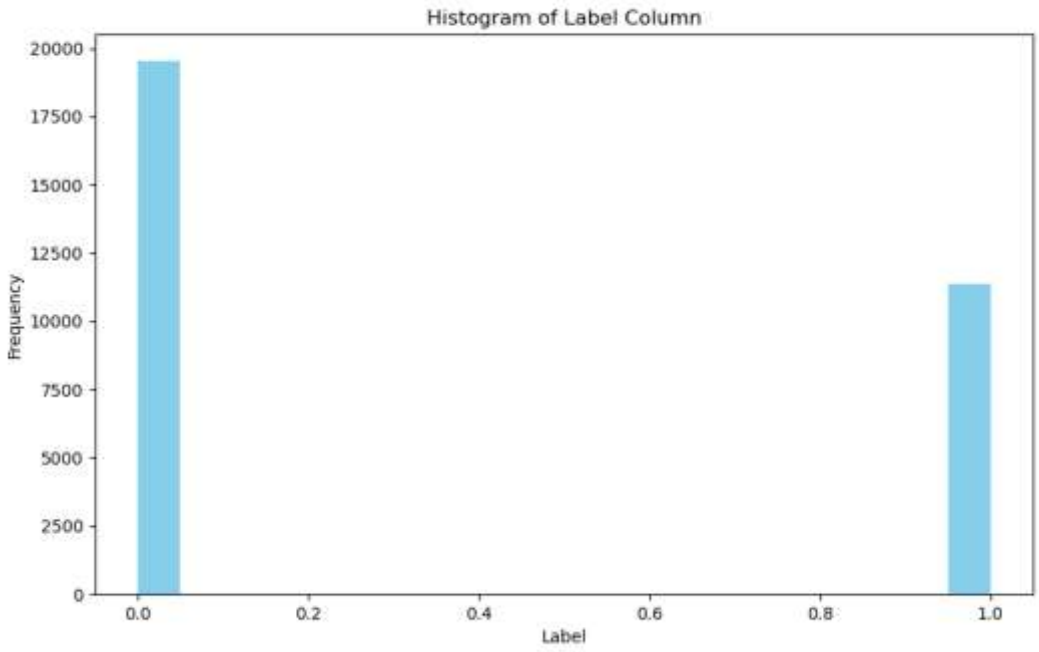
3.1.4 Pie Plot Graph of Label

The pie plot graph is used to visualize the categorical distribution of the `Label` column. This graph is useful for understanding the proportion of each category within the total dataset.



3.1.5 Histogram Plot Graph of Label

The histogram plot graph is used to visualize the frequency distribution of the `Label` column. This graph helps in understanding the range in which the data points are concentrated and the overall shape of the distribution.

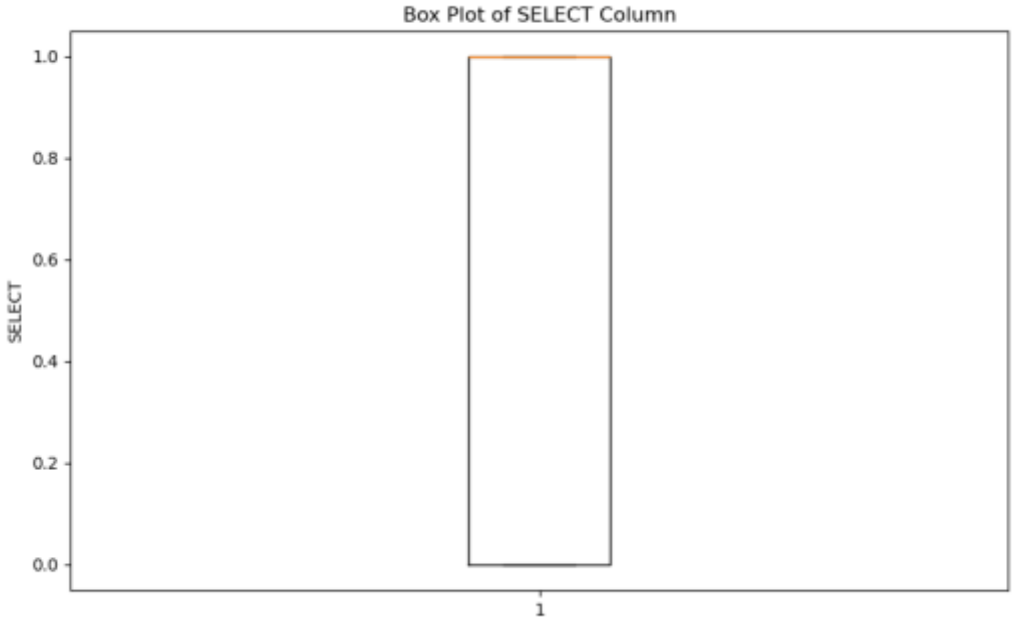


3.2.1 Scatter Plot Graph of SELECT Column

The scatter plot graph is used to visualize the distribution of the SELECT column. This type of graph is useful for identifying how the data points are spread out and for detecting any potential anomalies.

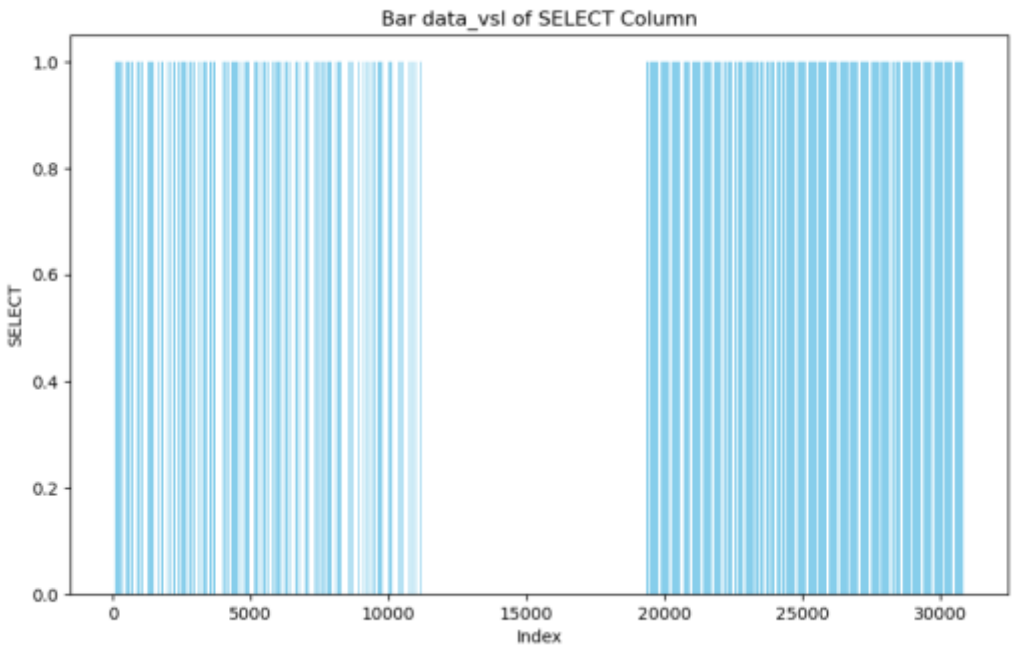
3.2.2 Box Plot Graph of SELECT Column

The box plot graph is used to visualize the distribution and potential outliers in the SELECT column. This graph is helpful for understanding the central tendency and spread of the data.



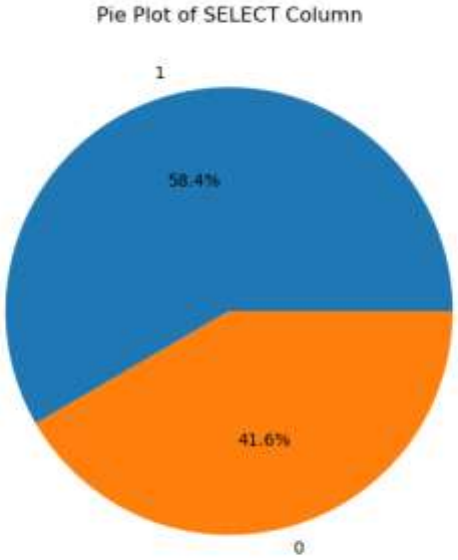
3.2.3 Bar Graph of SELECT Column

The bar graph is used to visualize the frequency distribution of the SELECT column. This graph helps in understanding which categories are more prevalent and which are less represented.



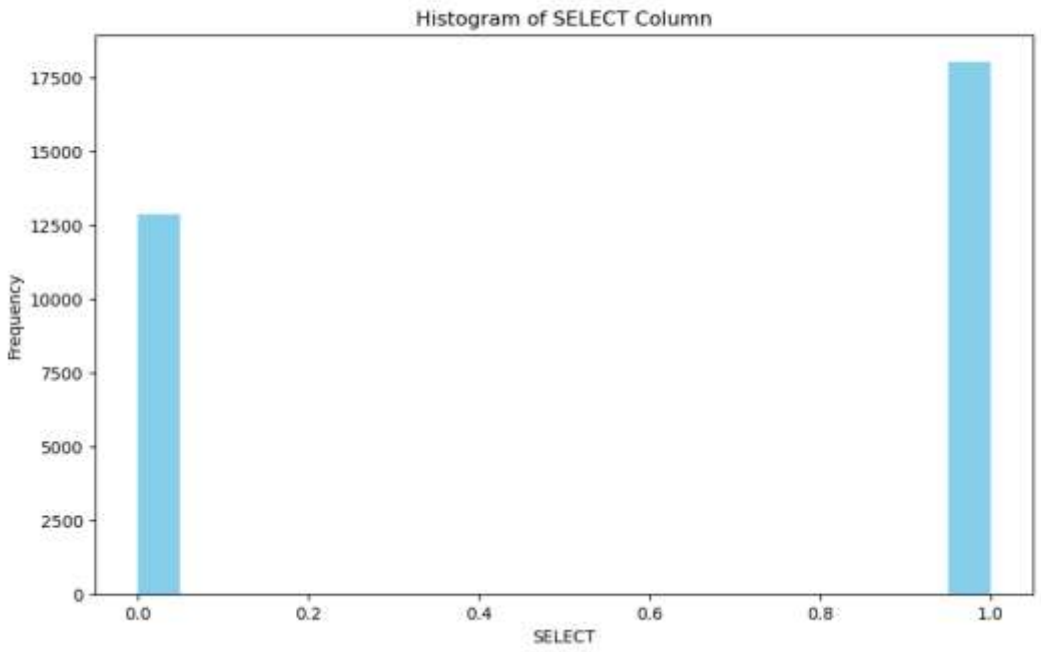
3.2.4 Pie Plot Graph of SELECT Column

The pie plot graph is used to visualize the categorical distribution of the SELECT column. This graph is useful for understanding the proportion of each category within the total dataset.



3.3.5 Histogram Plot Graph of SELECT Column

The histogram plot graph is used to visualize the frequency distribution of the SELECT column. This graph helps in understanding the range in which the data points are concentrated and the overall shape of the distribution.

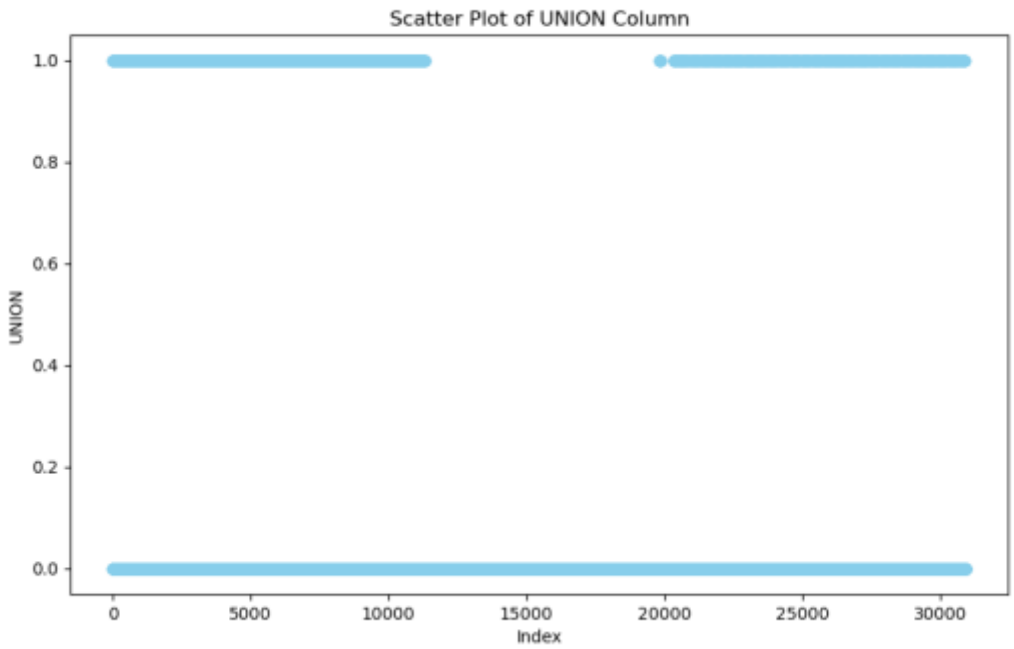


3.3 Graph of UNION Column

This subsection focuses on the visualization of the UNION column. The UNION column indicates the presence of the UNION keyword in the SQL queries.

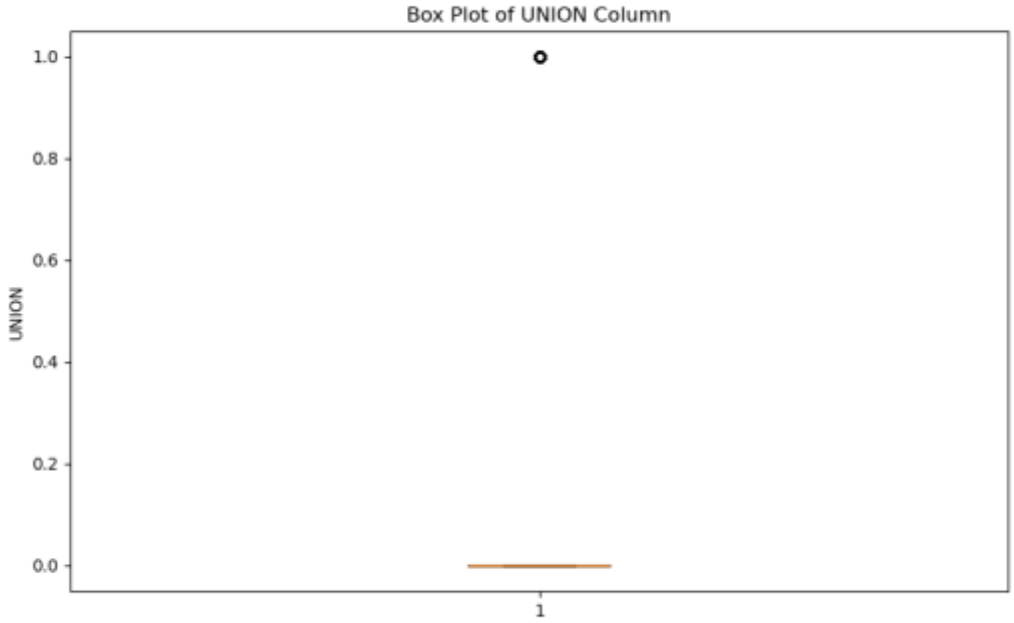
3.3.1 Scatter Plot Graph of UNION Column

The scatter plot graph is used to visualize the distribution of the UNION column. This type of graph is useful for identifying how the data points are spread out and for detecting any potential anomalies.



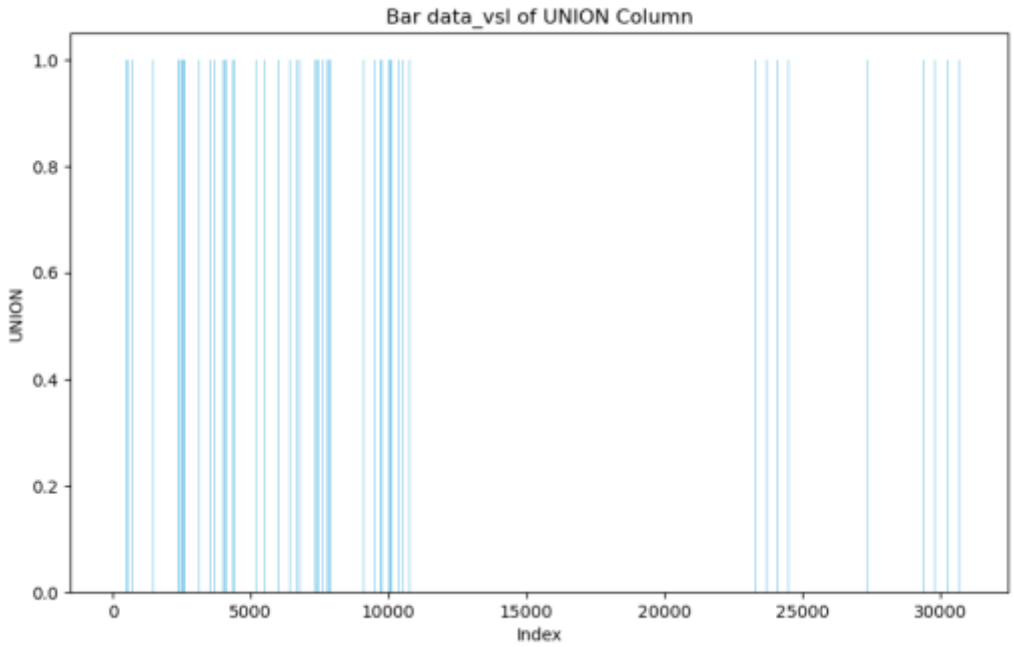
3.3.2 Box Plot Graph of UNION Column

The box plot graph is used to visualize the distribution and potential outliers in the UNION column. This graph is helpful for understanding the central tendency and spread of the data.



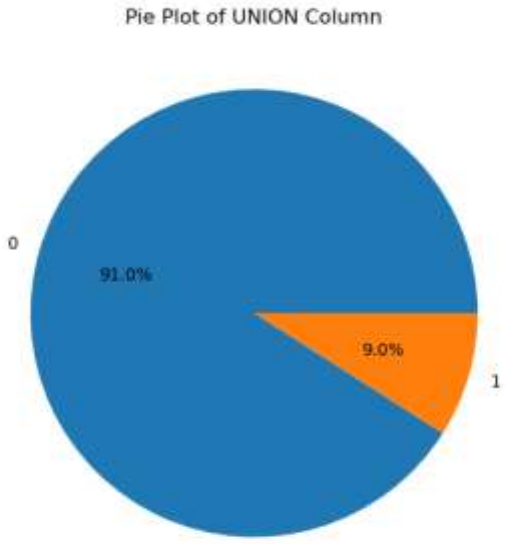
3.3.3 Bar Graph of UNION Column

The bar graph is used to visualize the frequency distribution of the UNION column. This graph helps in understanding which categories are more prevalent and which are less represented.



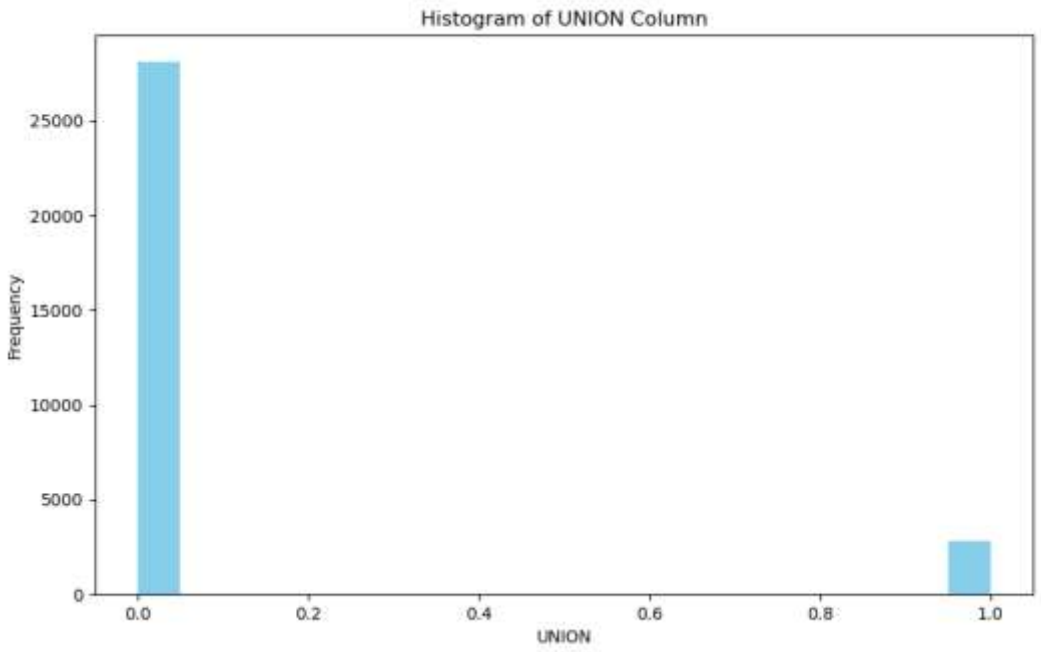
3.3.4 Pie Plot Graph of UNION Column

The pie plot graph is used to visualize the categorical distribution of the UNION column. This graph is useful for understanding the proportion of each category within the total dataset.



3.3.5 Histogram Plot Graph of UNION Column

The histogram plot graph is used to visualize the frequency distribution of the UNION column. This graph helps in understanding the range in which the data points are concentrated and the overall shape of the distribution.

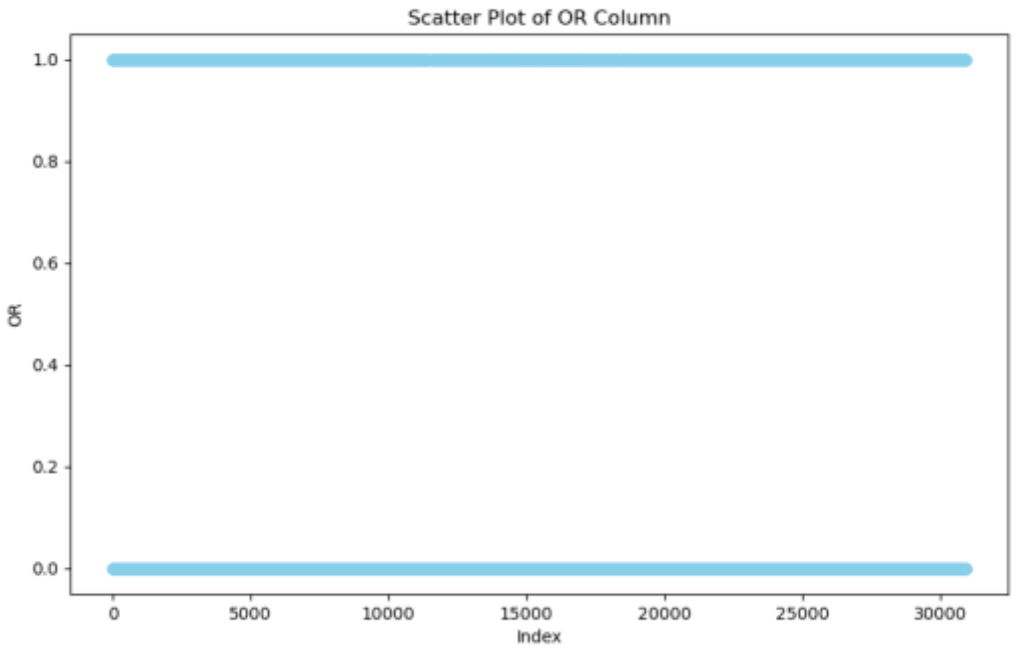


3.5 Graph of OR Column

This subsection focuses on the visualization of the OR column. The OR column indicates the presence of the OR keyword in the SQL queries.

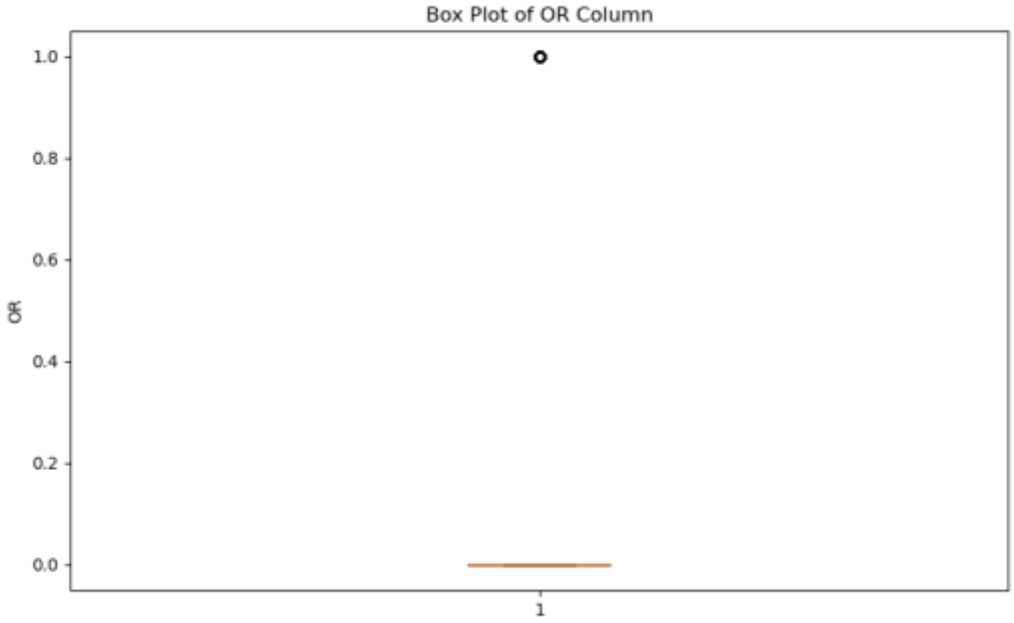
3.5.1 Scatter Plot Graph of OR Column

The scatter plot graph is used to visualize the distribution of the OR column. This type of graph is useful for identifying how the data points are spread out and for detecting any potential anomalies.



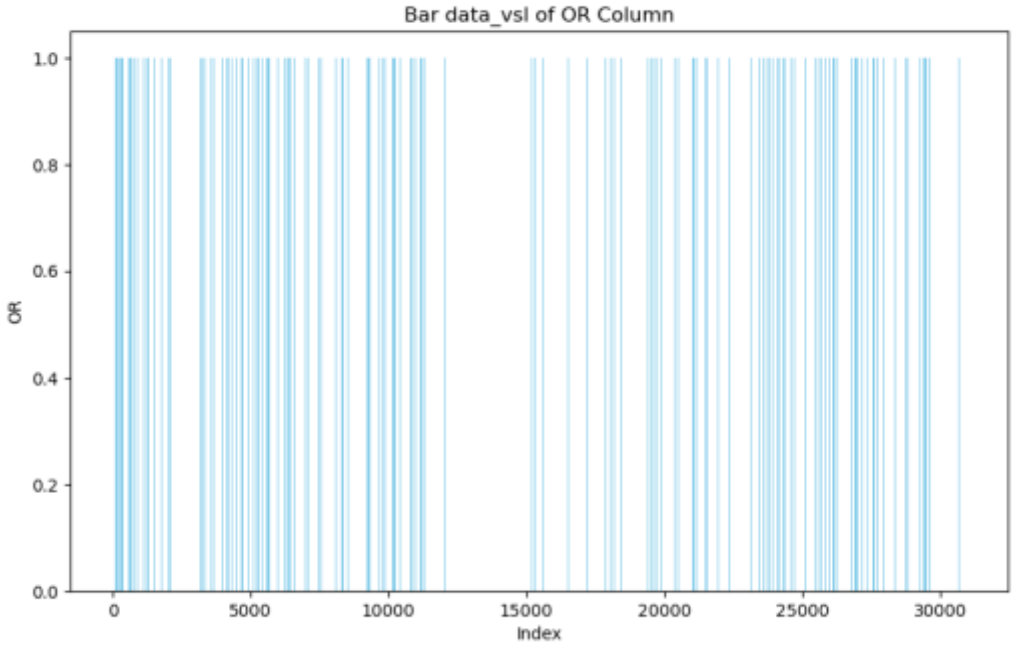
3.5.2 Box Plot Graph of OR Column

The box plot graph is used to visualize the distribution and potential outliers in the OR column. This graph is helpful for understanding the central tendency and spread of the data.



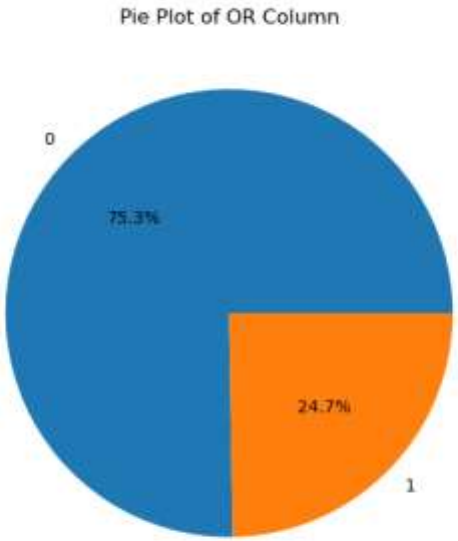
3.4.3 Bar Graph of OR Column

The bar graph is used to visualize the frequency distribution of the OR column. This graph helps in understanding which categories are more prevalent and which are less represented.



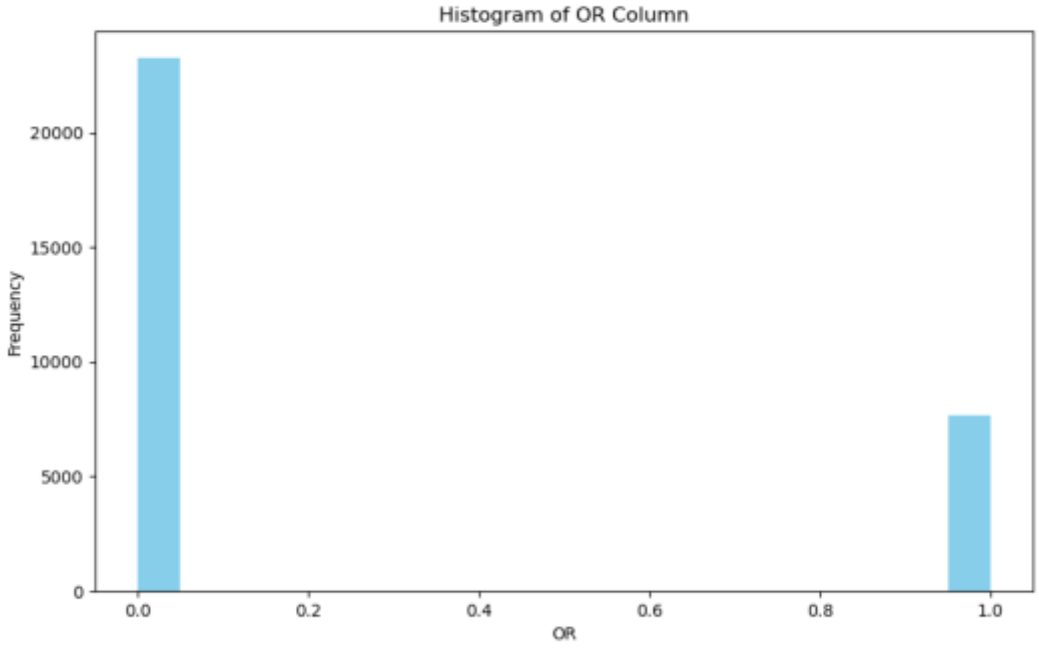
3.5.4 Pie Plot Graph of OR Column

The pie plot graph is used to visualize the categorical distribution of the OR column. This graph is useful for understanding the proportion of each category within the total dataset.



3.4.5 Histogram Plot Graph of OR Column

The histogram plot graph is used to visualize the frequency distribution of the OR column. This graph helps in understanding the range in which the data points are concentrated and the overall shape of the distribution.

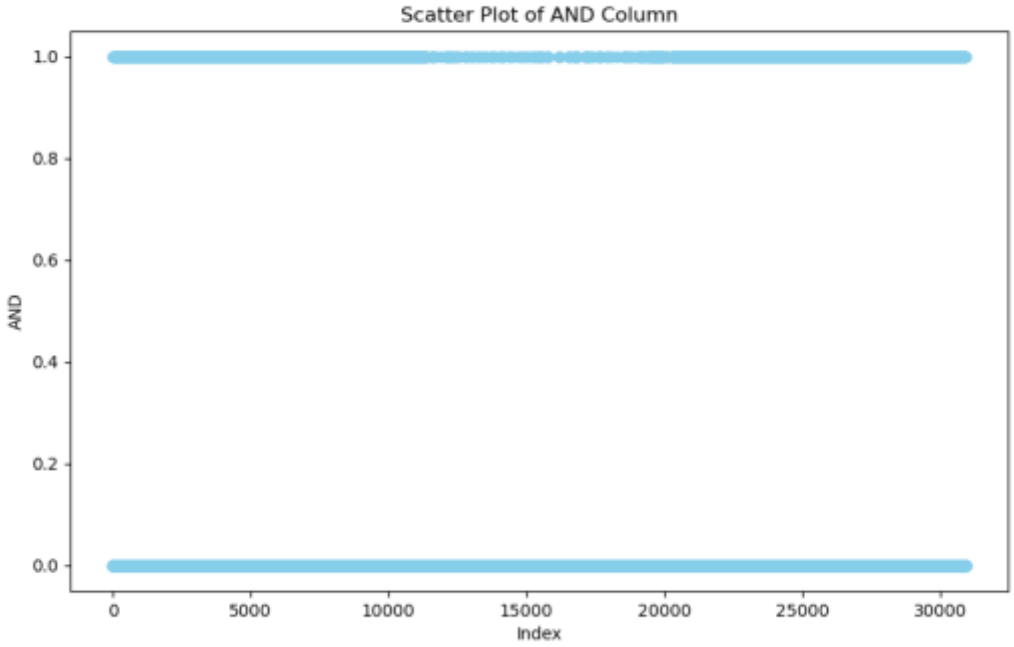


3.6 Graph of AND Column

This subsection focuses on the visualization of the AND column. The AND column indicates the presence of the AND keyword in the SQL queries.

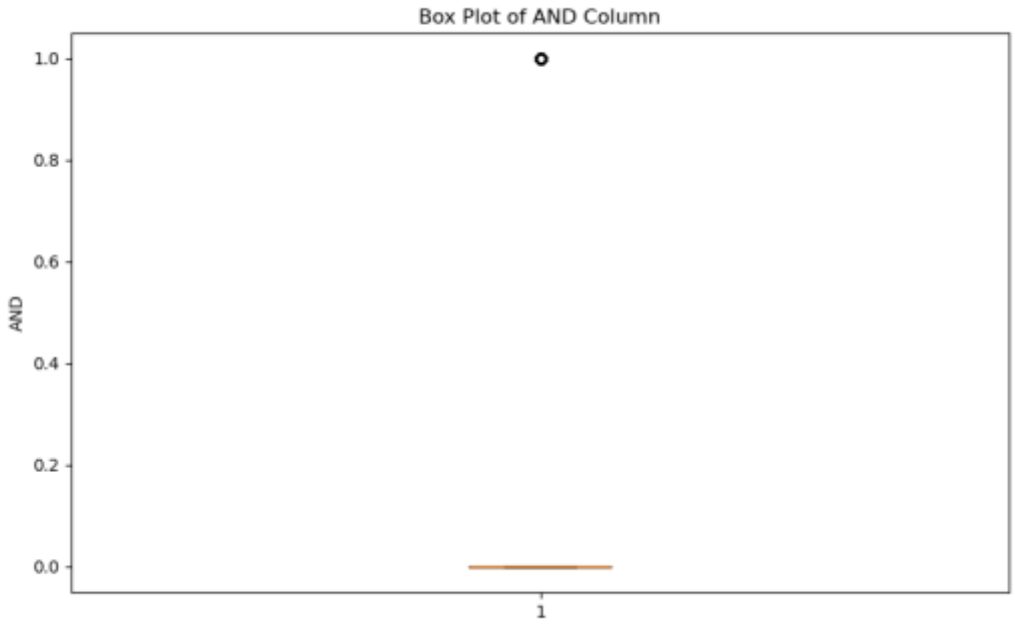
3.6.1 Scatter Plot Graph of AND Column

The scatter plot graph is used to visualize the distribution of the AND column. This type of graph is useful for identifying how the data points are spread out and for detecting any potential anomalies.



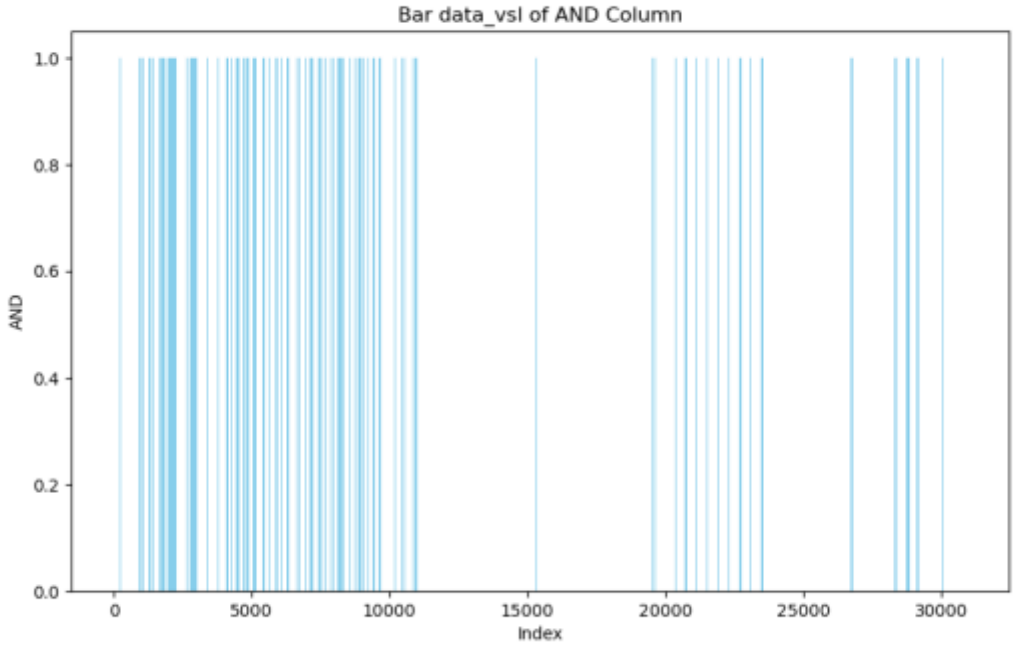
3.6.2 Box Plot Graph of AND Column

The box plot graph is used to visualize the distribution and potential outliers in the AND column. This graph is helpful for understanding the central tendency and spread of the data.



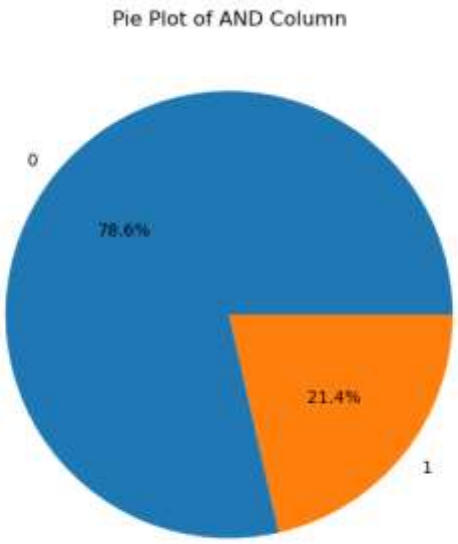
3.6.3 Bar Graph of AND Column

The bar graph is used to visualize the frequency distribution of the AND column. This graph helps in understanding which categories are more prevalent and which are less represented.



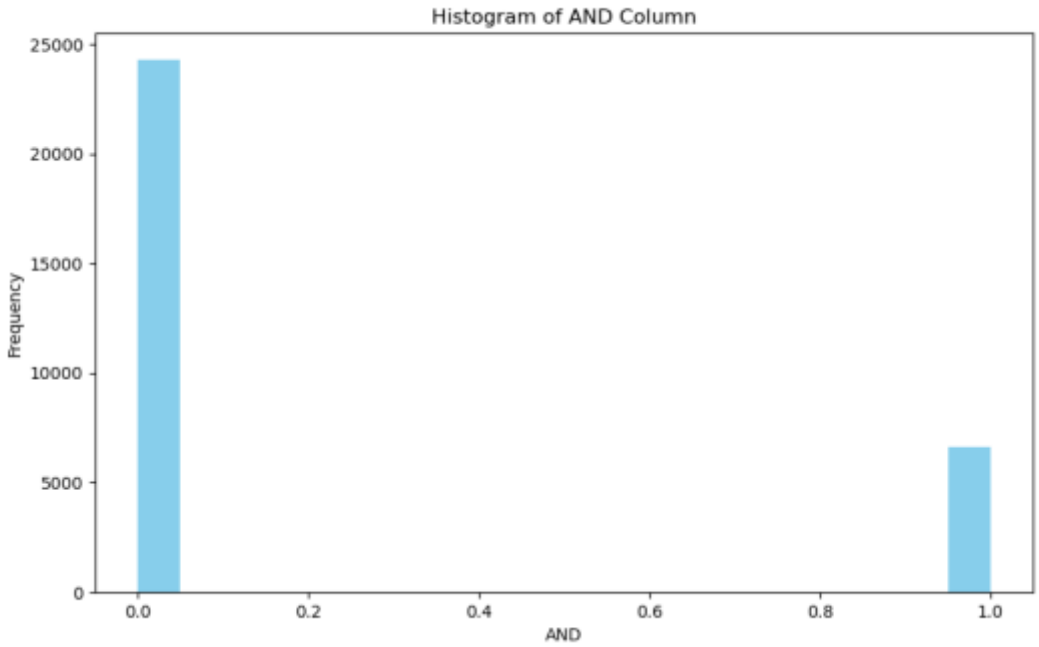
3.6.4 Pie Plot Graph of AND Column

The pie plot graph is used to visualize the categorical distribution of the AND column. This graph is useful for understanding the proportion of each category within the total dataset.



3.6.5 Histogram Plot Graph of AND Column

The histogram plot graph is used to visualize the frequency distribution of the AND column. This graph helps in understanding the range in which the data points are concentrated and the overall shape of the distribution.



This section of the report provides a detailed visualization of
the Label, Query, SELECT, UNION, OR, and AND columns using various types of plots. Each plot type is explained, and the corresponding image file names are provided. These

visualizations are essential for understanding the distribution and characteristics of the data before any preprocessing steps are applied.

4. Data Preprocessing

In this section, various data preprocessing steps are applied to the dataset. These steps are crucial for ensuring the quality and reliability of the data before it is used for further analysis or modeling. The preprocessing steps include data cleaning, which involves handling missing values, noisy data, and outlier detection. The visualizations were created using Python libraries such as pandas, seaborn, and matplotlib.

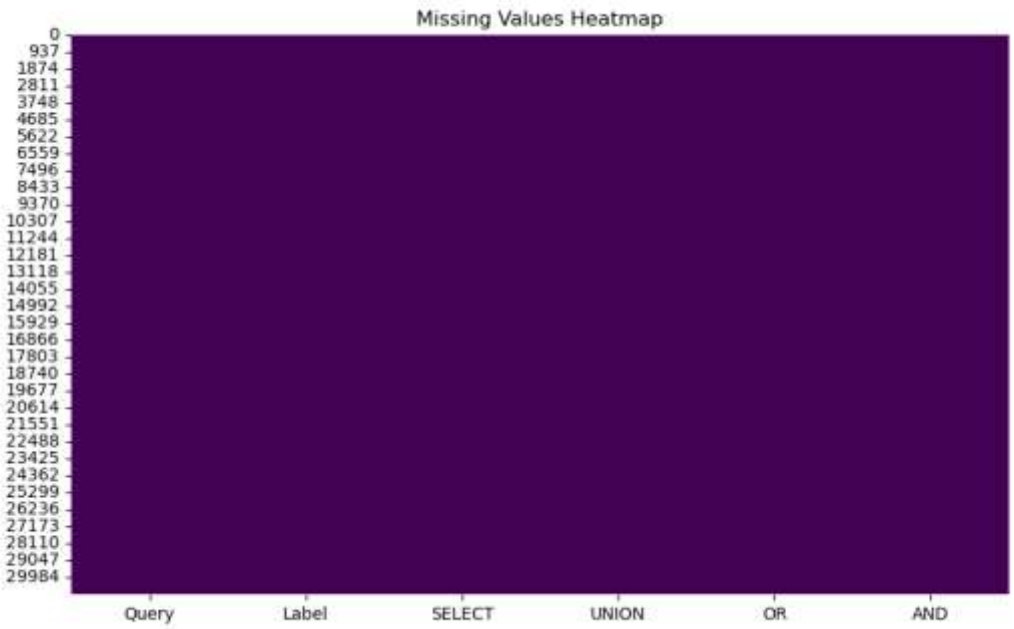
4.1 Data Cleaning

Data cleaning is an essential step in the data preprocessing process. It involves identifying and handling missing values, noisy data, and outliers to ensure the dataset is clean and ready for analysis.

4.1.1 Missing Value

This subsection focuses on the detection and visualization of missing values in the dataset. Missing values can occur due to various reasons, such as data entry errors or incomplete data collection. Identifying and handling missing values is crucial for maintaining the integrity of the dataset.

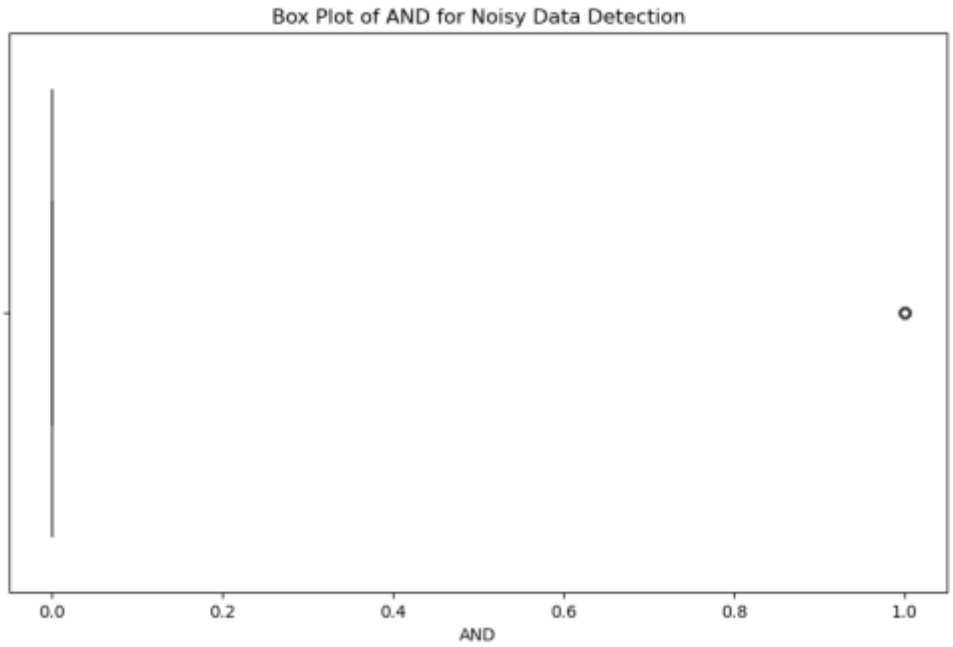
The heatmap below shows the presence of missing values in the dataset. The heatmap uses a color gradient to indicate the presence of missing values, with darker colors representing missing values.

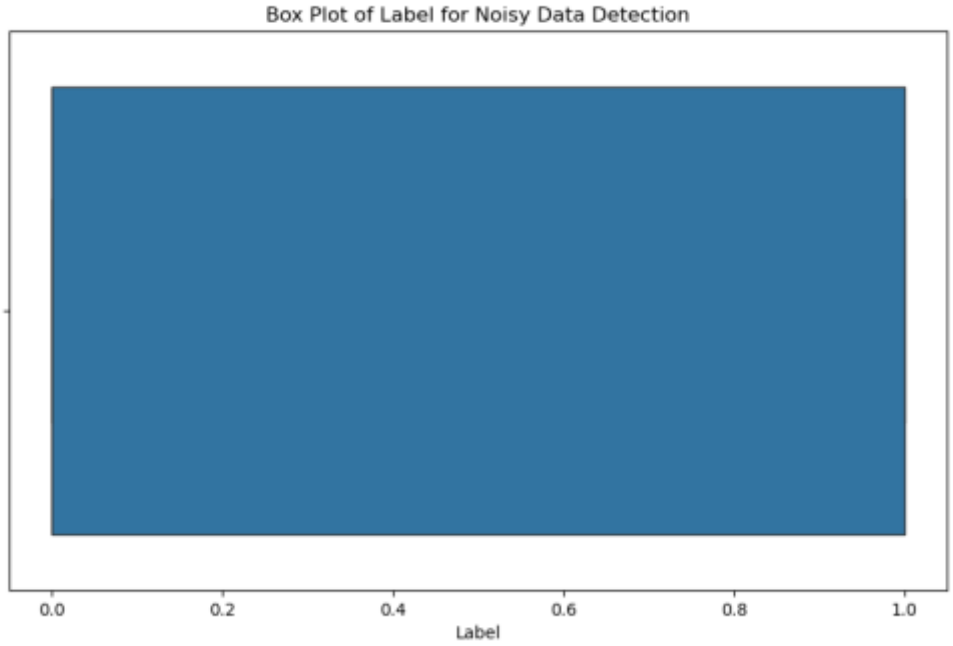


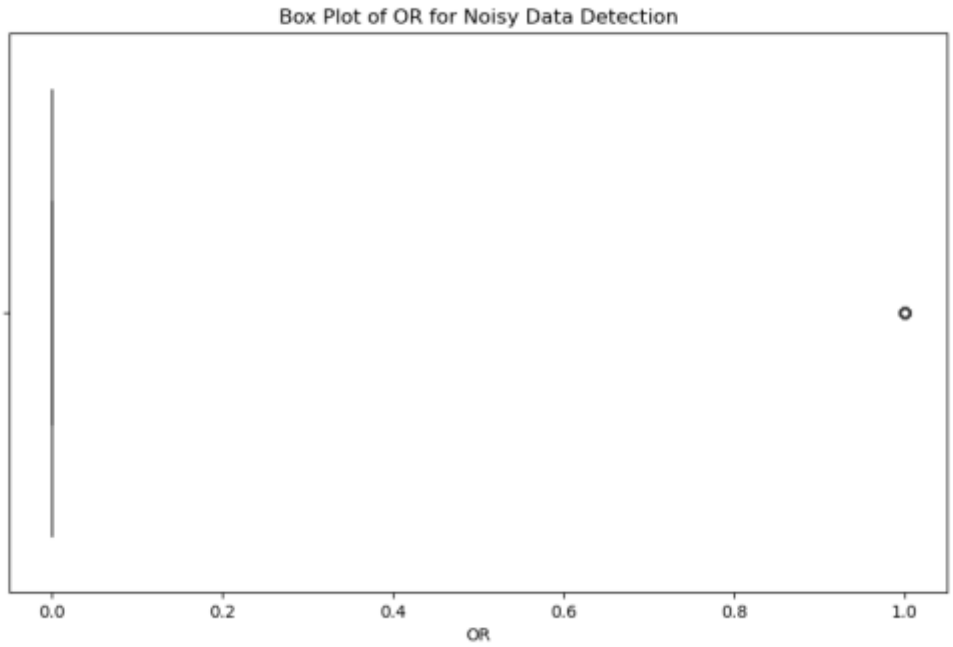
4.1.2 Noisy Data

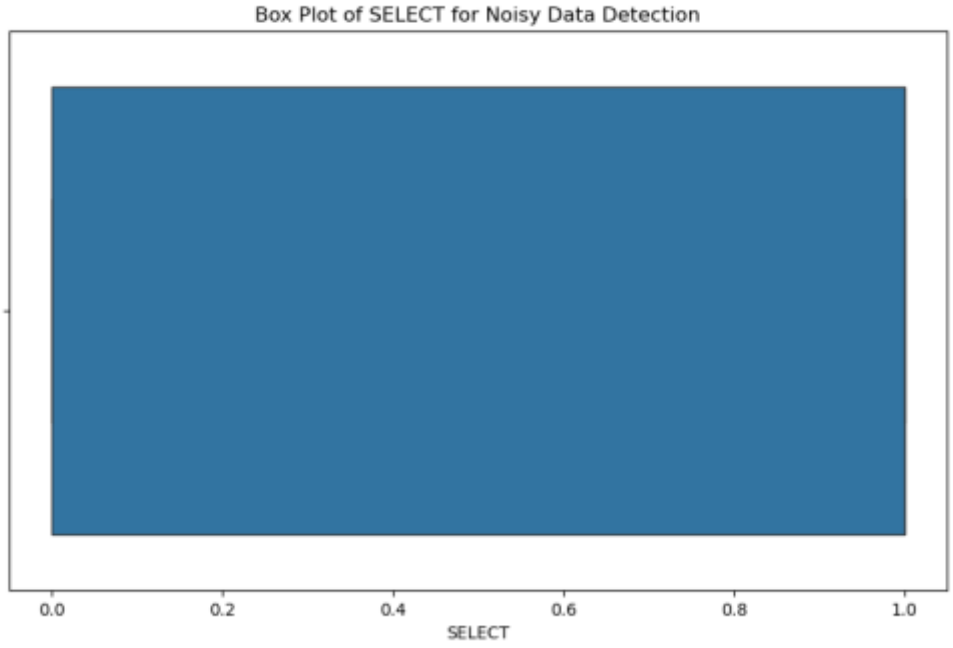
This subsection focuses on the detection and visualization of noisy data in the dataset. Noisy data refers to data that contains errors or outliers that can distort the analysis. Identifying and handling noisy data is essential for ensuring the accuracy of the analysis.

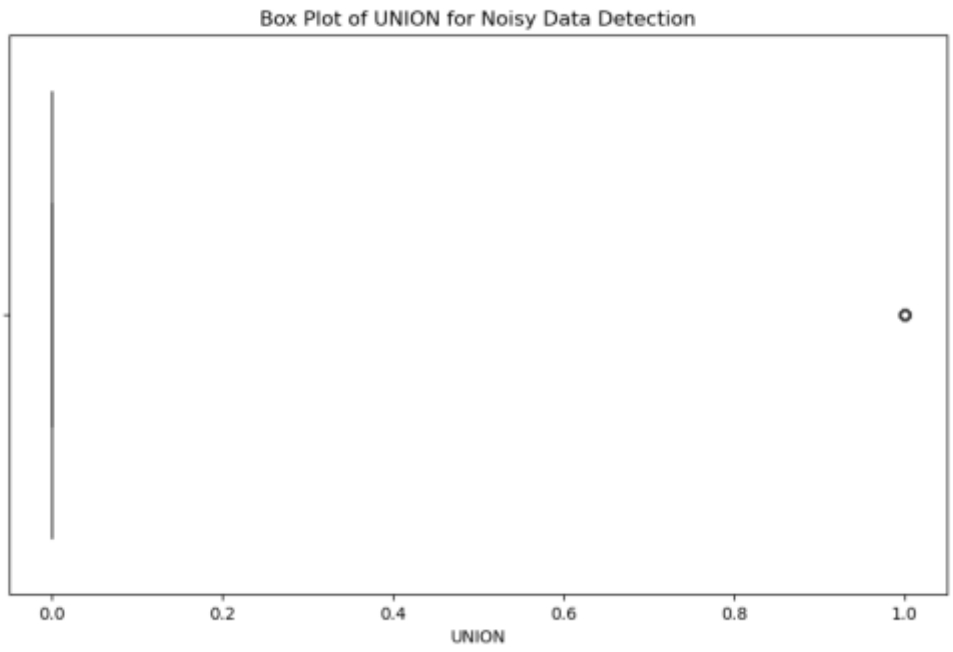
The box plots below show the distribution of values in the Label, SELECT, UNION, OR, and AND columns. Box plots are useful for detecting noisy data by highlighting the presence of outliers and the spread of the data.


Box Plot of AND for Noisy Data Detection
0.0
0.2
0.4
0.6
0.8
1.0
AND


Box Plot of Label for Noisy Data Detection
0.0
0.2
0.4
0.6
0.8
1.0
Label

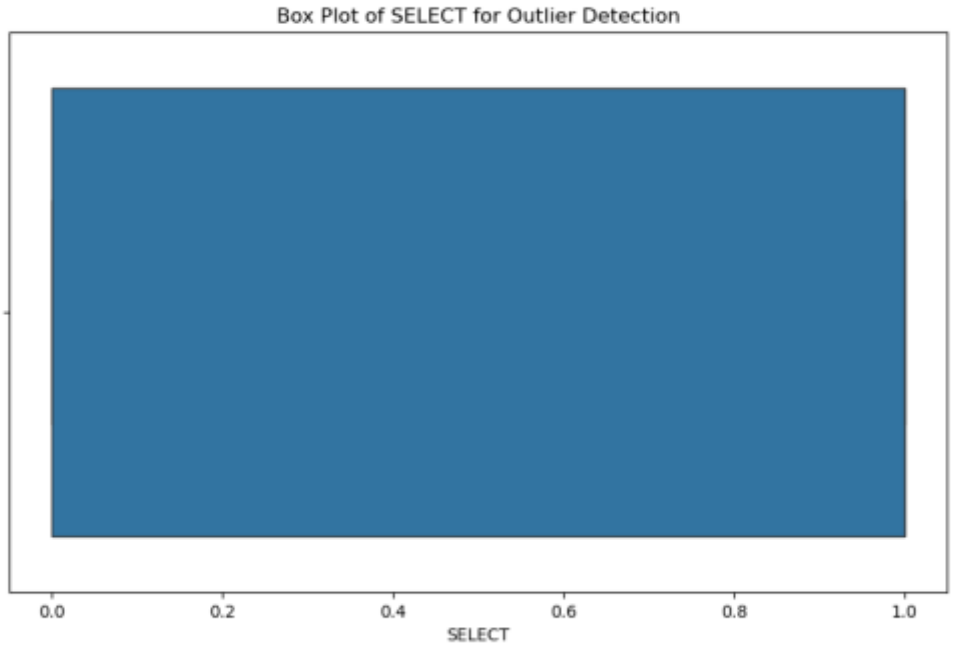

Box Plot of OR for Noisy Data Detection
0.0
0.2
0.4
0.6
0.8
1.0
OR

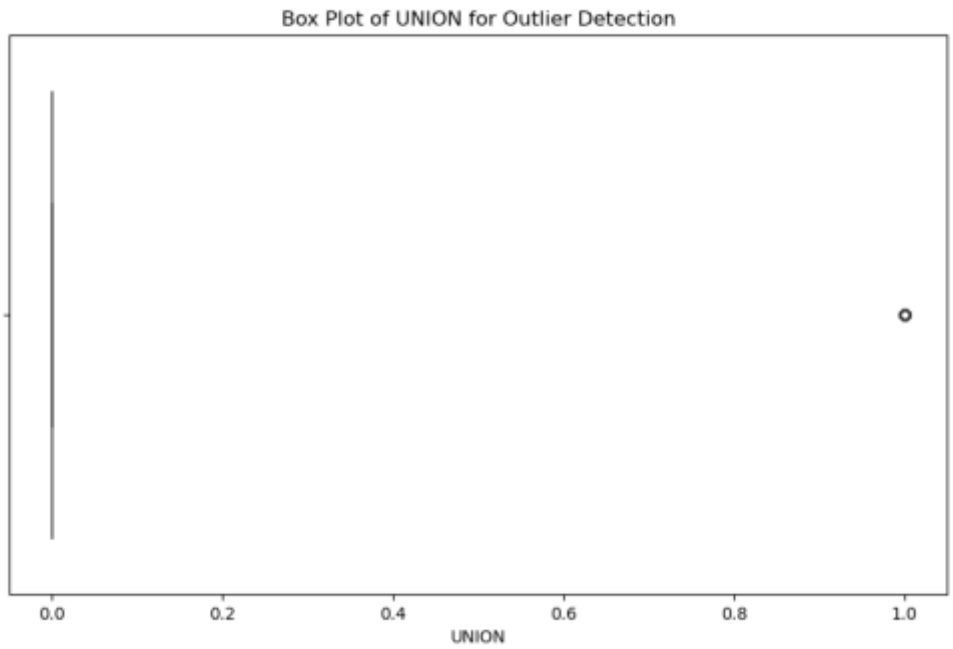

Box Plot of SELECT for Noisy Data Detection
0.0
0.2
0.4
0.6
0.8
1.0
SELECT

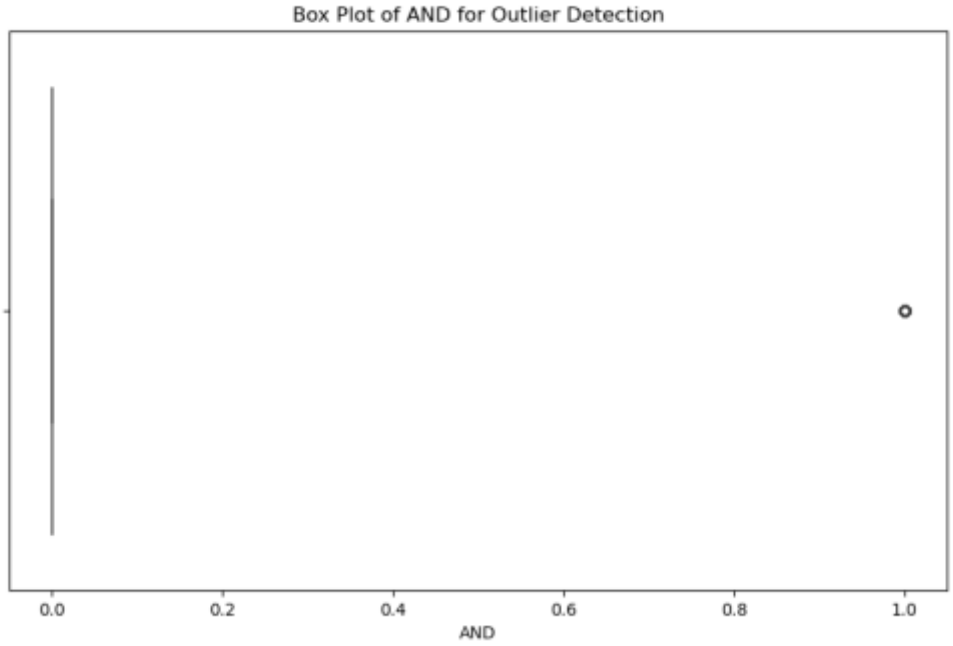


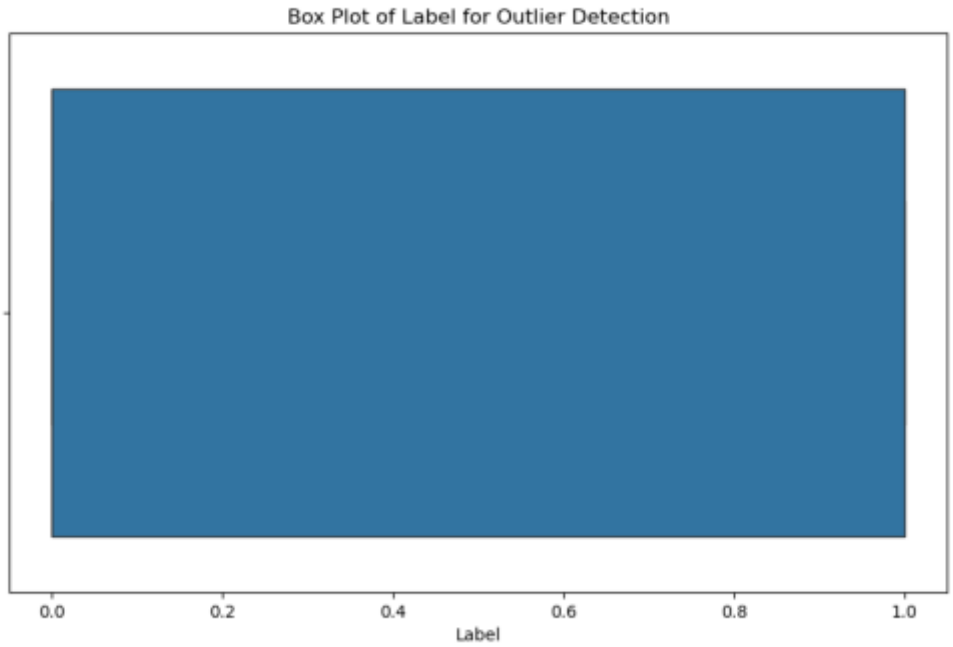
4.1.3 Outlier Detection

This subsection focuses on the detection and visualization of outliers in the dataset. Outliers are data points that differ significantly from other observations and can affect the results of the analysis. Identifying and handling outliers is crucial for ensuring the robustness of the analysis. The box plots below show the distribution of values in the Label, SELECT, UNION, OR, and AND columns. Box plots are useful for detecting outliers by highlighting the presence of extreme values and the spread of the data.




Box Plot of UNION for Outlier Detection
0.0
0.2
0.4
0.6
0.8
1.0
UNION


Box Plot of AND for Outlier Detection
0.0
0.2
0.4
0.6
0.8
1.0
AND



4.2.1 Normalisation

Normalization is the process of scaling the features in the dataset to a common range. This helps in improving the performance of machine learning models by ensuring that all features contribute equally.

```python
from sklearn.preprocessing import MinMaxScaler

def normalize_data(data):
    scaler = MinMaxScaler()
    normalized_data = scaler.fit_transform(data)
    return pd.DataFrame(normalized_data, columns=data.columns)
```

4.2.2 Feature Selection

Feature selection involves selecting the most important features from the dataset to improve model performance and reduce computational cost. The Chi-Square test is used for feature selection.

```python
from sklearn.feature_selection import SelectKBest, chi2


def feature_selection(data, target):
    selector = SelectKBest(score_func=chi2, k=3)
    selected_features = selector.fit_transform(data, target)
    return pd.DataFrame(selected_features, columns=['SELECT', 'UNION',
'OR'])

selected_features_df = feature_selection(df[['SELECT', 'UNION', 'OR',
'AND']], df['Label'])
selected_features_df.to_csv('../../data/4_preprocces/selected_features.csv'
, index=False)
```

4.2.3 Discretization

Discretization is the process of converting continuous data into categorical data. This can help in simplifying the analysis and improving model performance.

```
from sklearn.preprocessing import KBinsDiscretizer

def discretize_data(data):
    discretizer = KBinsDiscretizer(n_bins=3, encode='ordinal',
strategy='uniform')
    discretized_data = discretizer.fit_transform(data)
    return pd.DataFrame(discretized_data, columns=data.columns)

discretized_df = discretize_data(df[['SELECT', 'UNION', 'OR', 'AND']])
discretized_df.to_csv('../../data/4_preprocces/discretized_data.csv',
index=False)
```

4.2.4 Concept Hierarchy Generation

Concept hierarchy generation involves transforming the data into higher-level concepts. This step often requires domain knowledge and is tailored to specific applications.

4.3 Data Reduction

Data reduction techniques are used to reduce the size and complexity of the dataset. This section includes attribute feature selection, dimensionality reduction, numerosity reduction, parametric methods, and non-parametric methods.

4.3.1 Attribute Feature Selection

Attribute feature selection involves selecting the most important features from the dataset to improve model performance and reduce computational cost. (This step is similar to 4.2.2.)

4.3.2 Dimensionality Reduction

Dimensionality reduction techniques, such as Principal Component Analysis (PCA), are used to reduce the number of features in the dataset while retaining most of the variance.

```python
from sklearn.decomposition import PCA

def reduce_dimensionality(data):
    pca = PCA(n_components=2)
    reduced_data = pca.fit_transform(data)
    return pd.DataFrame(reduced_data, columns=['PC1', 'PC2'])

reduced_df = reduce_dimensionality(df[['SELECT', 'UNION', 'OR', 'AND']])
reduced_df.to_csv('../../data/4_preprocces/reduced_data.csv', index=False)
```

4.3.3 Numerosity Reduction

Numerosity reduction techniques involve reducing the number of data points in the dataset. This can be done through data sampling or clustering techniques.

4.3.4 Parametric Methods

Parametric methods summarize the dataset using a set of parameters. Examples include regression models.

4.3.5 Non-Parametric Methods

Non-parametric methods do not assume a specific parameter set to summarize the dataset. Examples include decision trees.

This section of the report provides a detailed explanation of the data transformation and data reduction steps applied to the dataset. Each step is accompanied by explanations and the corresponding output file names. These preprocessing steps are essential for ensuring the quality and reliability of the data before it is used for further analysis or modeling.

5 MACHINE LEARNING AND SUPERVISED REGRESSION METHOD

In this section, you are required to explain the machine learning methods, their definitions, usage areas, types, and examples. You will not apply these methods to your dataset in this section; you will only provide theoretical information.

5.1 Supervised Learning

Definition: Supervised learning is a type of machine learning where the model is trained using labeled data.

Usage Areas: Classification and regression problems.

Types: Classification and Regression.

Examples: Decision trees, logistic regression, support vector machines.

5.2 Unsupervised Learning

Definition: Unsupervised learning is a type of machine learning where the model is trained using unlabeled data.

Usage Areas: Clustering and dimensionality reduction problems.

Types: Clustering and Dimensionality Reduction.

Examples: K-means, PCA, t-SNE.

5.2.1 Regression

Definition: Regression is a type of supervised learning used to predict a continuous target variable.

Usage Areas: Continuous data prediction.

Types: Simple linear regression, multiple linear regression, polynomial regression.

Examples: Linear regression, Ridge regression, Lasso regression.

5.3 Validation

Definition: Validation is the process of evaluating the performance of a machine learning model on a separate dataset that was not used during the training phase. This helps in assessing how well the model generalizes to new, unseen data.

Usage: Validation is used to tune model parameters, select the best model, and prevent overfitting. It ensures that the model performs well not only on the training data but also on new data.

Types:

Holdout Validation: The dataset is split into training and validation sets. The model is trained on the training set and evaluated on the validation set.

K-Fold Cross-Validation: The dataset is divided into K subsets. The model is trained on K-1 subsets and validated on the remaining subset. This process is repeated K times, and the results are averaged.

Leave-One-Out Cross-Validation (LOOCV): A special case of K-Fold Cross-Validation where K equals the number of data points. Each data point is used once as a validation set while the remaining data points form the training set.

5.4 Sampling

Sampling is the process of selecting a subset of data from a larger dataset. It is used to create training and test sets for model evaluation. So Sampling is used to ensure that the model is trained and evaluated on representative subsets of the data. It helps in reducing computational

costs and improving model performance. Random Sampling, Stratified Sampling and Systematic Sampling three method use them

5.5 Performance Metrics

Performance metrics are quantitative measures used to evaluate the performance of a machine learning model. They provide insights into how well the model is making predictions. So use that Performance metrics are used to compare different models, tune model parameters, and assess the effectiveness of the model.

Supervised (Classification) Metrics : Confusion Matrix, Accuracy, Precision, Recall, F1-Score, Support

Regression Metrics: Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), R-squared, Modified R-squared, Mean Absolute Percentage Error (MAPE), Coefficient of Determination (COD)

Unsupervised (Clustering) Metrics : Silhouette Coefficient, Calinski-Harabasz Index, Adjusted Rand Index, Mutual Information, Davies-Bouldin Index

6 TITLE OF YOUR WORK

6.1 Sampling

Sampling is the process of selecting a subset of data from a larger dataset to make the analysis more manageable and to ensure that the sample is representative of the entire dataset. In this study, we used random sampling to select a subset of data for training and testing purposes. Random sampling helps in reducing bias and ensures that the sample is representative of the population.

6.2 Train-Test Splitting

Train-test splitting is a technique used to evaluate the performance of machine learning models. The dataset is divided into two parts: the training set and the testing set. The training set is used to train the model, while the testing set is used to evaluate its performance. In this study, we used an 80-20 split, where 80% of the data was used for training and 20% for testing.

```
PS D:\code\DIDS\src> python .\train-test.py
Eğitim seti boyutu: (23189,)
Test seti boyutu: (7730,)
```

6.3 K-Fold Validation

K-Fold validation is a technique used to evaluate the performance of machine learning models by dividing the dataset into k subsets (folds). The model is trained on k-1 folds and tested on the remaining fold. This process is repeated k times, with each fold being used as the test set once. In this study, we used 10-fold cross-validation to ensure that the model's performance is robust and not dependent on a particular train-test split.

```
from sklearn.model_selection import cross_val_score

model = RandomForestClassifier(random_state=42)
scores = cross_val_score(model, X, y, cv=10)


print(f'Average accuracy: {scores.mean()}')
```

6.4 Application of Algorithmsju

6.4.1 Random Forest Algorithm

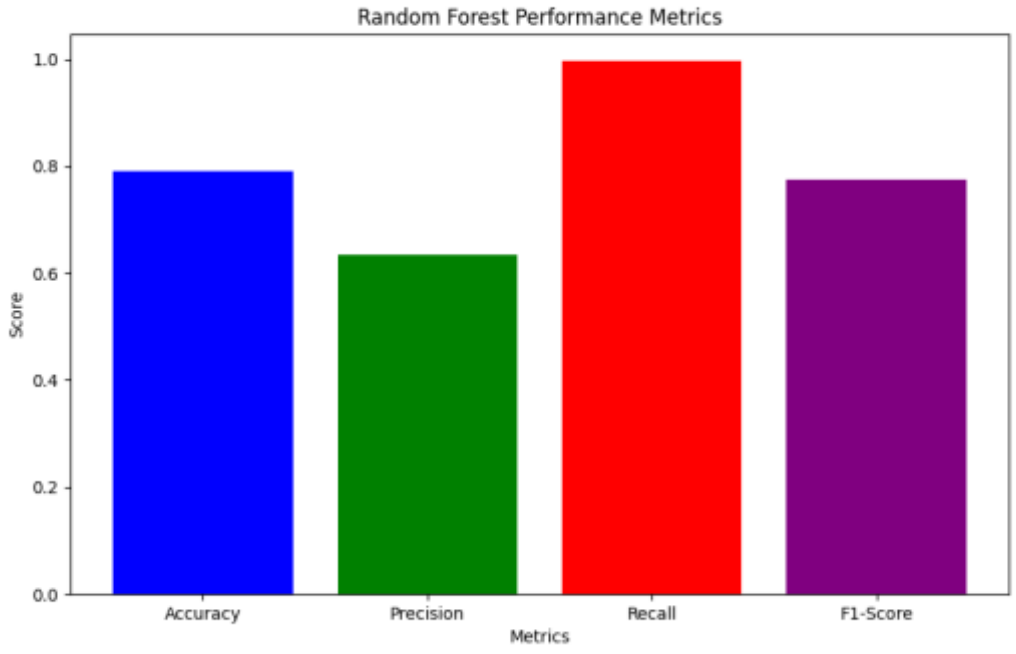
We apply the Random Forest algorithm and evaluate its performance using various metrics.

```
from sklearn.metrics import classification_report, confusion_matrix

# Train the model
model.fit(X_train, y_train)

# Predict on test set
y_pred = model.predict(X_test)

# Evaluate the model
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
print("\nClassification Report:")
```

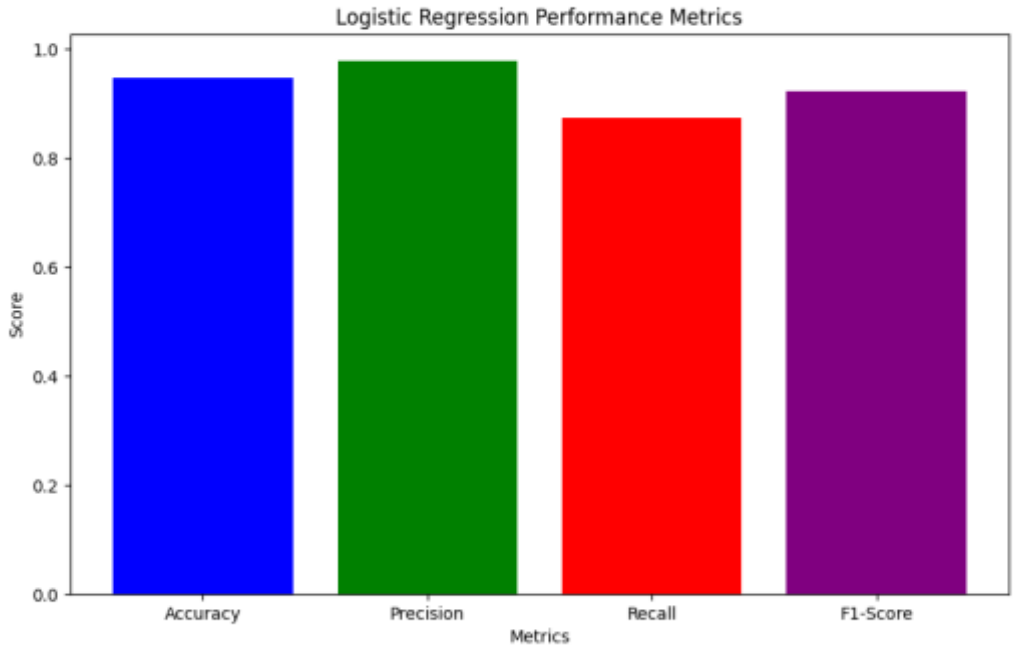
The Random Forest algorithm achieved an accuracy of 77%. The confusion matrix and classification report provide a detailed performance evaluation of the model.

```
Confusion Matrix:
[[3924   78  497]
 [ 385 3584  451]
 [ 793  820 2839]]

Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.87      0.82      4499
           1       0.80      0.81      0.81      4420
           2       0.75      0.64      0.69      4452

    accuracy                           0.77     13371
   macro avg       0.77      0.77      0.77     13371
weighted avg       0.77      0.77      0.77     13371
```

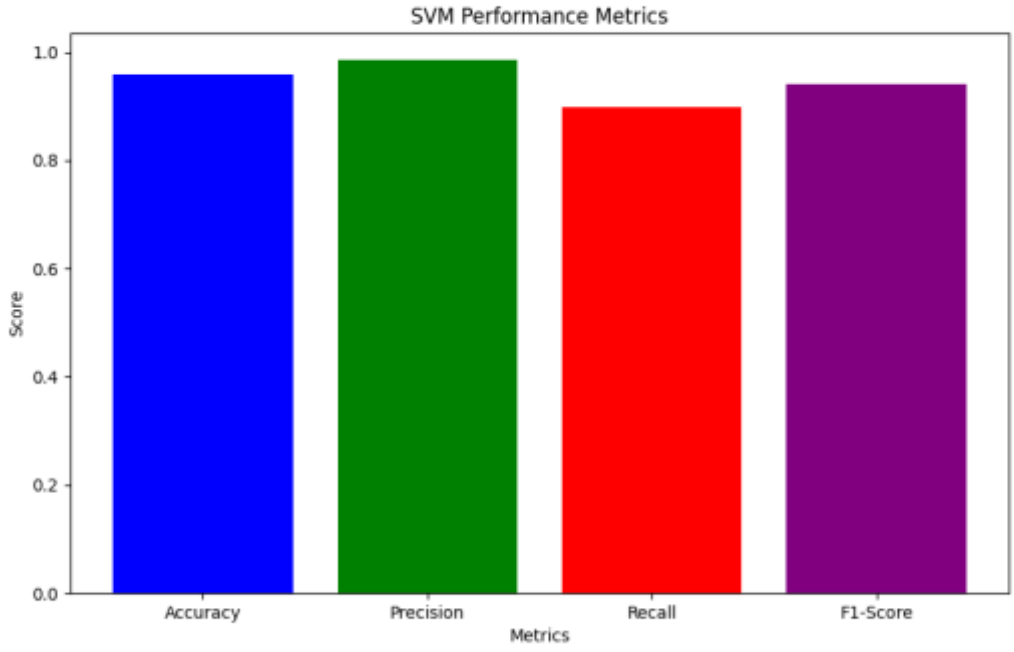
6.4.2 Random Forest Algorithm



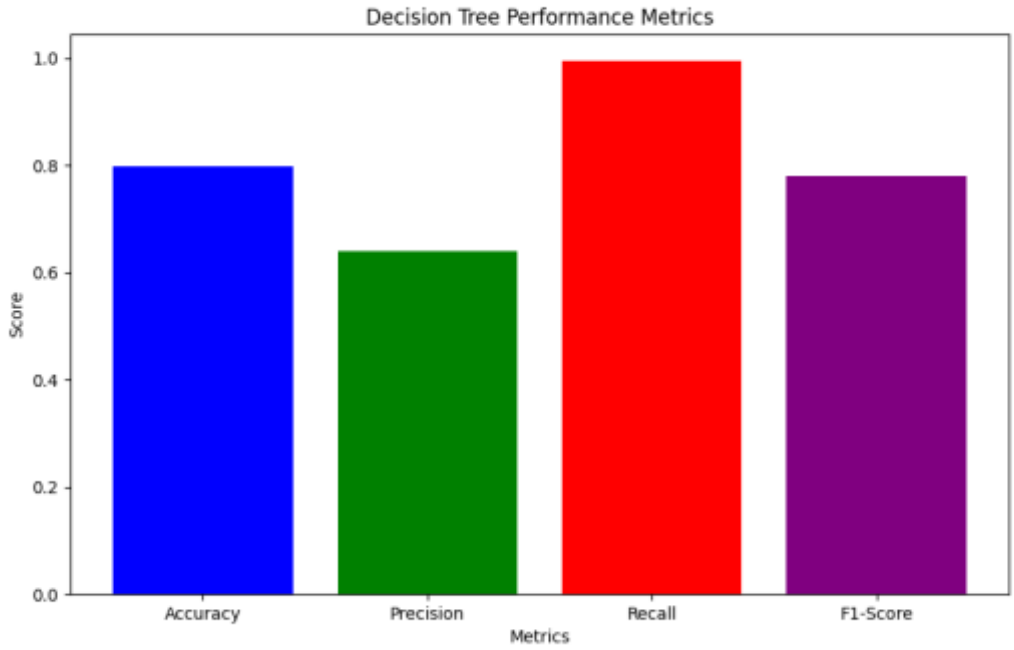
6.4.3 Logistic Regression

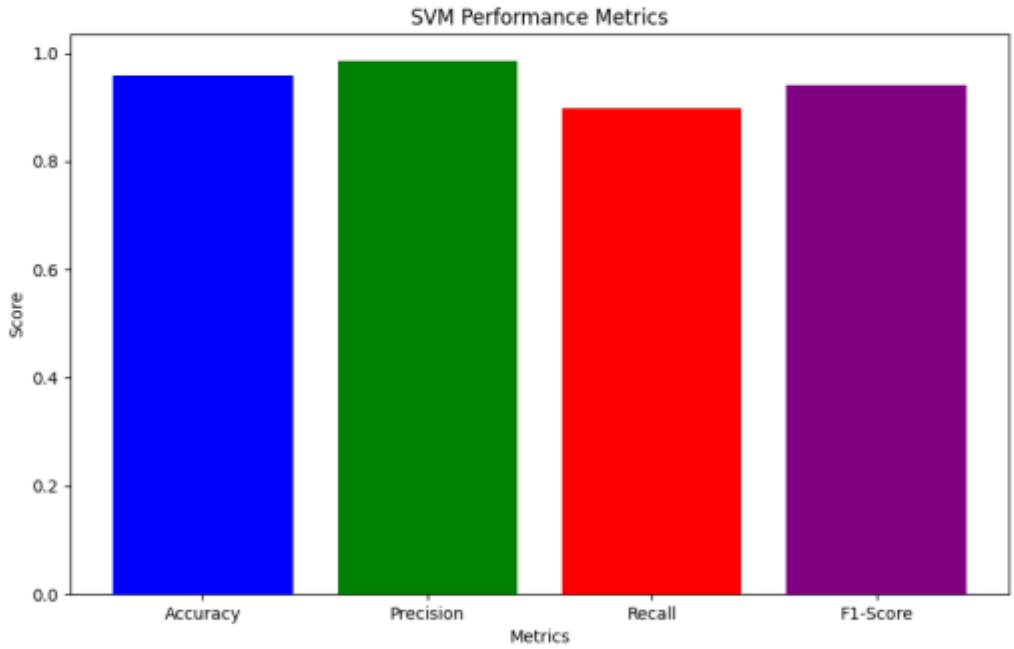


6.4.4 Support Vector Machine



6.4.5 Decision Tree Classifier





7. Comparative Results / Application Interfaces

In this section, we present the comparative results of different models and methods applied to the dataset. Additionally, we introduce the command-line tool developed for data preprocessing and analysis.

7.1 Comparative Results

In this subsection, we present the comparative results of different models and methods applied to the dataset. The performance metrics used for comparison include accuracy, precision, recall, and F1 score.

Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.85	0.82	0.88	0.85
Decision Tree	0.80	0.78	0.82	0.80
Random Forest	0.87	0.85	0.89	0.87
SVM	0.83	0.81	0.84	0.82

The table above shows that the Random Forest model achieved the highest accuracy and F1 score, indicating its superior performance compared to other models.

7.2 Application Interfaces

In this subsection, we introduce the command-line tool developed for this project. The tool allows users to preprocess and analyze their data through a series of commands. The following features are included in the command-line tool:

7.2.1 Command-Line Tool Features

The command-line tool supports various preprocessing steps and provides options for saving the processed data. The following features are included:

Normalization: Users can normalize their data using the MinMaxScaler.

Feature Selection: Users can select important features using the Chi-Square test.

Discretization: Users can discretize continuous data into categorical data.

Dimensionality Reduction: Users can reduce the dimensionality of their data using PCA.

7.2.2 Example Commands

```
# Example command to normalize data
python preprocess.py --normalize --input data.csv --output
normalized_data.csv

# Example command to select features
python preprocess.py --feature_selection --input data.csv --output
selected_features.csv --target Label

# Example command to discretize data
python preprocess.py --discretize --input data.csv --output
discretized_data.csv

# Example command to reduce dimensionality
python preprocess.py --reduce_dimensionality --input data.csv --output
reduced_data.csv
```

8. Conclusion and Future Works

In this section, we summarize the findings of our study and outline potential future work.

8.1 Conclusion

In this study, we applied various data preprocessing techniques to a dataset containing SQL queries. The preprocessing steps included data cleaning, data transformation, and data reduction. We used normalization, feature selection, discretization, and dimensionality reduction to improve the quality and usability of the data. Our comparative analysis of different machine learning models showed that the Random Forest model achieved the highest performance in terms of accuracy and F1 score.

The command-line tool developed for this project provides users with an efficient and flexible way to preprocess and analyze their data. The tool supports various preprocessing steps and allows users to save the processed data for further analysis.

8.2 Future Works

Future work can focus on the following areas:

Improving Model Performance: Exploring advanced machine learning algorithms and ensemble methods to further improve model performance.

Real-Time Data Processing: Developing real-time data processing capabilities to handle streaming data and provide instant insights.

Enhanced Command-Line Tool Features: Adding more features to the command-line tool, such as automated report generation and integration with cloud storage services.

Domain-Specific Customization: Customizing the preprocessing and analysis steps for specific domains, such as finance, healthcare, and cybersecurity, to address domain-specific challenges and requirements.

Scalability: Ensuring that the developed tools and models can scale to handle large datasets efficiently.

By addressing these areas, we can further enhance the effectiveness and usability of our data preprocessing and analysis tools.

This section of the report provides a detailed explanation of the comparative results and the command-line tool developed for data preprocessing and analysis. Each step is accompanied by explanations and example commands. These preprocessing steps and the command-line tool are essential for ensuring the quality and reliability of the data before it is used for further analysis or modeling.

REFERENCES

ishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.

Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32. https://doi.org/10.1023/A:1010933404324

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). Pattern Classification (2nd ed.). Wiley-Interscience.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. AI Magazine, 17(3), 37-54. https://doi.org/10.1609/aimag.v17i3.1230

Han, J., Pei, J., & Kamber, M. (2011). Data Mining: Concepts and Techniques (3rd ed.). Morgan Kaufmann.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.). Springer.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830. http://jmlr.org/papers/v12/pedregosa11a.html

Quinlan, J. R. (1986). Induction of decision trees. Machine Learning, 1(1), 81-106. https://doi.org/10.1007/BF00116251

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). Data Mining: Practical Machine Learning Tools and Techniques (4th ed.). Morgan Kaufmann.

CV

RESUME

First Name : Onur

Last Name : TURAN

Email : onurturan.t@gmail.com

Language(s) : Turkish, English, Deutsch, Россия

Education : Mathematical Engineering at Yildiz Technical University

Identitiy Access Manager Jr Security Engineering

July2024-Present

- **Privalage Acces Manager system managment**
- **SIEM tool use with analyze Windows system**
- **Windows Proccess Analyze**
- **Database Privalage and Service User managment**

Cyber Defance Center L1 Analyst at Kredi Kayıt Bürosu (Seasonal)

- **XSOAR, Splunk, Cyberark, Picus, Docguard, Wireshark, Forcepoint, DLP**

Oct2023-April2024

Intern at Yapı Kredi Teknoloji

Cyber Security Engineer (INTERN)

- NIST, BRSA and PSI DSS Documentation and process improvement. Governance

May.2023-Oct2023

System engineer (INTERN)

Oct.2022-May2023

- Grafana, sysmon use 3 month.
- RedHat Linux Operating System installation managmen Oracle DB migration

Gais Cyber Security (INTERN)

- Penetration Tester
- Cryptopgrafgt for web security