Exploratory Data Analysis (EDA) on Titanic Dataset

Overview

Exploratory Data Analysis (EDA) is a critical step in understanding the structure, patterns, and relationships within a dataset. This EDA focuses on the Titanic dataset, which contains information about passengers aboard the Titanic, including their survival status, age, fare, class, and more. The analysis includes handling missing values, detecting duplicates, summarizing statistics, encoding categorical variables, detecting outliers, and creating new features.

Why Perform EDA?

- **Understand Data**: EDA helps in understanding the distribution, trends, and relationships within the data.
- **Identify Issues**: It helps in identifying missing values, outliers, and potential data quality issues.
- **Guide Feature Engineering**: Insights from EDA can guide the creation of new features or the transformation of existing ones.
- **Prepare for Modeling**: EDA ensures that the data is clean and ready for machine learning models.

Prerequisites

- **Python 3.x**
- **Pandas**
- **NumPy**
- **Seaborn**
- **Scipy**

Files Included

1. **EDA Code**: The main script for performing exploratory data analysis on the Titanic dataset.

2. **Summary Statistics**: Descriptive statistics for numerical and categorical features.
3. **Visualizations**: Various plots and visualizations to understand the data distribution and relationships.
4. **Feature Engineering**: Creation of new features based on existing ones.

Code Description

1. Importing Necessary Libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
from scipy.stats import zscore
```

2. Loading the Titanic Dataset

```
# Using seaborn's built-in Titanic dataset
df = sns.load_dataset('titanic')
```

3. Basic Dataset Information

```
# Display first 5 rows
print("First 5 Rows:\n", df.head())

# Display last 5 rows
print("Last 5 Rows:\n", df.tail())

# Dataset shape
print("Dataset Shape:", df.shape)

# Column names and data types
print("Column Information:\n")
df.info()
```

4. Handling Missing Values

```
# Checking for missing values
print("\nMissing Values Per Column:\n", df.isnull().sum())

# Filling missing values for numerical columns with median
df['age'].fillna(df['age'].median(), inplace=True)

# Filling missing values for categorical columns with mode
df['embark_town'].fillna(df['embark_town'].mode()[0], inplace=True)

# Dropping columns with too many missing values
df.drop(columns=['deck'], inplace=True)
```

5. Handling Duplicates

```
# Checking for duplicate rows
print("\nTotal Duplicate Rows:", df.duplicated().sum())

# Removing duplicates
df = df.drop_duplicates()
```

6. Summary Statistics

```
# Displaying summary statistics for numerical columns
print("\nSummary Statistics (Numerical):\n", df.describe())

# Displaying summary statistics for categorical columns
print("\nSummary Statistics (Categorical):\n", df.describe(include=['O'])

# Checking unique values in categorical features
categorical_features = ['sex', 'embark_town', 'class', 'who', 'alive']
for col in categorical_features:
    print(f"\nUnique Values in {col}:\n", df[col].value_counts())

# Checking range and percentiles for numerical features
numerical_features = ['age', 'fare', 'sibsp', 'parch']
for col in numerical_features:
    print(f"\n{col} Percentiles:\n", df[col].quantile([0.01, 0.25, 0.5, (
```

7. One-Hot Encoding

```
import pandas as pd
import numpy as np

# Load dataset (assuming df is already loaded)
df_encoded = df.copy()  # Creating a copy to avoid modifying the original

# Identify categorical columns
categorical_columns = df_encoded.select_dtypes(include=['object', 'categ

# Convert categorical columns to string before encoding
df_encoded[categorical_columns] = df_encoded[categorical_columns].astype(

# Apply One-Hot Encoding
df_encoded = pd.get_dummies(df_encoded, columns=categorical_columns, drop

# Convert boolean values (if any) to integers
df_encoded = df_encoded.astype(float)

# Display the correlation matrix
print("\nCorrelation Matrix:\n", df_encoded.corr())

# Finding highly correlated features (absolute correlation > 0.5)
correlation_matrix = df_encoded.corr().abs()
high_corr = correlation_matrix[correlation_matrix > 0.5]
print("\nHighly Correlated Features (Correlation > 0.5):\n", high_corr)
```

8. Outlier Detection

```
# Using IQR (Interquartile Range) method
for col in ['fare', 'age']:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    outliers_iqr = df[(df[col] < (Q1 - 1.5 * IQR)) | (df[col] > (Q3 + 1.5
    print(f"\nOutliers Detected in {col} Using IQR:\n", outliers_iqr[[col

# Using Z-score method
df['fare_zscore'] = zscore(df['fare'])
outliers_z = df[df['fare_zscore'].abs() > 3]
print("\nOutliers Detected in Fare Using Z-score:\n", outliers_z[['fare',
```

```
# Dropping the temporary Z-score column
df.drop(columns=['fare_zscore'], inplace=True)
```

9. Skewness and Kurtosis

```
# Checking skewness & kurtosis for numerical features
for col in numerical_features:
    print(f"\nSkewness of {col}: {df[col].skew()}")
    print(f"Kurtosis of {col}: {df[col].kurt()}")
```

10. Feature Engineering

```
# Creating a new feature: Family Size
df['family_size'] = df['sibsp'] + df['parch'] + 1

# Creating a new binary feature: Is Alone?
df['is_alone'] = (df['family_size'] == 1).astype(int)

# Checking new feature distributions
print("\nNew Feature Summary:\n", df[['family_size', 'is_alone']].describ
```

11. Conclusion

```
print("\nFinal Dataset Shape After EDA:", df.shape)
```

Expected Outputs

1. **Summary Statistics**: Descriptive statistics for numerical and categorical features.
2. **Correlation Matrix**: A matrix showing the correlation between features.
3. **Outlier Detection**: Identification of outliers using IQR and Z-score methods.
4. **New Features**: Creation of new features such as `family_size` and `is_alone`.

Use Cases

- **Data Understanding**: Gain insights into the distribution and relationships within the Titanic dataset.
- **Data Cleaning**: Identify and handle missing values, duplicates, and outliers.
- **Feature Engineering**: Use insights from EDA to create or transform features for machine learning models.
- **Model Preparation**: Ensure the data is clean and ready for modeling.

Advantages

- **Comprehensive Analysis**: Provides a thorough understanding of the dataset.
- **Data Quality Improvement**: Helps in identifying and handling missing values, duplicates, and outliers.
- **Guides Modeling**: Insights from EDA can guide the selection of features and models.

Future Enhancements

- **Advanced Visualizations**: Incorporate more advanced visualizations like 3D plots or interactive plots.
- **Automated EDA**: Use automated EDA tools like Pandas Profiling or Sweetviz for faster analysis.
- **Feature Engineering**: Use insights from EDA to create new features or transform existing ones.
- **Modeling**: Use the cleaned and analyzed data to build predictive models.

References

- Pandas Documentation
- Seaborn Documentation
- Scipy Documentation
- Titanic Dataset on Kaggle

This README provides a comprehensive overview of the EDA performed on the Titanic dataset, including the purpose, implementation, and potential applications. It also includes instructions for running the code and interpreting the results.