Exploratory Data Analysis (EDA) on Titanic Dataset

Overview

Exploratory Data Analysis (EDA) is a critical step in the data analysis process, where the goal is to understand the structure, patterns, and relationships within the data. 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 distribution analysis, handling missing values, visualization of relationships, and statistical summaries.

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**
- **Seaborn**
- **Matplotlib**
- **NumPy**
- **Scipy**

Files Included

1. **EDA Code**: The main script for performing exploratory data analysis on the Titanic dataset.
2. **Visualizations**: Various plots including histograms, boxplots, violin plots, pairplots, jointplots, heatmaps, countplots, barplots, and swarmplots.
3. **Summary Statistics**: Descriptive statistics, skewness, kurtosis, and normality tests.

Code Description

1. Data Loading and Basic Information

```
 import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats

# Load the dataset
file_path = "Titanic-Dataset.csv"  # Update if necessary
df = pd.read_csv(file_path)

# Display basic info
df.info(), df.head()
```

2. Distribution Analysis

```python
numerical_features = ['Age', 'Fare', 'SibSp', 'Parch']

# Histograms & KDE Plots
fig, axes = plt.subplots(2, 2, figsize=(12, 8))
for i, feature in enumerate(numerical_features):
    sns.histplot(df[feature], kde=True, ax=axes[i//2, i%2])
    axes[i//2, i%2].set_title(f'Distribution of {feature}')
plt.tight_layout()
plt.show()

# Boxplots & Violin Plots
fig, axes = plt.subplots(2, 2, figsize=(12, 8))
for i, feature in enumerate(numerical_features):
    sns.boxplot(y=df[feature], ax=axes[i//2, i%2])
    axes[i//2, i%2].set_title(f'Boxplot of {feature}')
plt.tight_layout()
plt.show()

fig, axes = plt.subplots(2, 2, figsize=(12, 8))
for i, feature in enumerate(numerical_features):
    sns.violinplot(y=df[feature], ax=axes[i//2, i%2])
    axes[i//2, i%2].set_title(f'Violin Plot of {feature}')
plt.tight_layout()
plt.show()
```

3. Pairplot and Jointplot

```python
# Pairplot of numerical features
sns.pairplot(df[numerical_features])
plt.show()

# Jointplot for Age vs Fare
sns.jointplot(x=df["Age"], y=df["Fare"], kind="scatter", marginal_kws=dict(bins=25, fill=True))
plt.show()
```

4. Correlation Heatmap

```python
# Heatmap of correlations
plt.figure(figsize=(8, 6))
sns.heatmap(df[numerical_features].corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```

5. Handling Missing Values

```python
# Handling missing values by filling with median for Age and mode for Embarked
df['Age'].fillna(df['Age'].median(), inplace=True)
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)

# Dropping Cabin due to excessive missing values
df.drop(columns=['Cabin'], inplace=True)

# Confirm changes
df.info()
```

6. Categorical Features Analysis

```
categorical_features = ["Pclass", "Sex", "Embarked", "Survived"]

# Countplots
fig, axes = plt.subplots(2, 2, figsize=(12, 8))
for i, feature in enumerate(categorical_features):
    sns.countplot(x=df[feature], ax=axes[i//2, i%2], palette="Set2")
    axes[i//2, i%2].set_title(f'Countplot of {feature}')
plt.tight_layout()
plt.show()

# Barplot of Fare by Pclass
plt.figure(figsize=(8, 5))
sns.barplot(x="Pclass", y="Fare", data=df, estimator=np.median, palette="viridis")
plt.title("Median Fare by Pclass")
plt.show()

# Swarmplot for Age by Survived
plt.figure(figsize=(8, 5))
sns.swarmplot(x="Survived", y="Age", data=df, palette="coolwarm")
plt.title("Age Distribution by Survival")
plt.show()
```

7. Summary Statistics and Normality Tests

```
# Summary statistics
print(df.describe())

# Skewness and Kurtosis
for feature in numerical_features:
    skewness = df[feature].skew()
    kurtosis = df[feature].kurt()
    print(f"{feature}: Skewness = {skewness:.2f}, Kurtosis = {kurtosis:.2f}")

# Normality Tests (Shapiro-Wilk)
for feature in numerical_features:
    stat, p = stats.shapiro(df[feature].dropna())
    print(f"{feature}: Shapiro-Wilk Test p-value = {p:.4f} (Normal if p > 0.05)")
```

Expected Outputs

1. **Visualizations**: Histograms, boxplots, violin plots, pairplots, jointplots, heatmaps, countplots, barplots, and swarmplots.
2. **Summary Statistics**: Descriptive statistics for numerical features.
3. **Skewness and Kurtosis**: Measures of the shape of the distribution.
4. **Normality Tests**: Shapiro-Wilk test results to check for normality.

Use Cases

- **Data Understanding**: Gain insights into the distribution and relationships within the Titanic dataset.
- **Data Cleaning**: Identify and handle missing values 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.
- **Visual Insights**: Visualizations help in identifying patterns and anomalies.
- **Data Quality Improvement**: Helps in identifying and handling missing values 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
- Matplotlib 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.