Exploratory Data Analysis of Red Wine Quality

This report presents an exploratory data analysis (EDA) of the red wine quality dataset. The dataset contains physicochemical and sensory variables related to red wine quality. The analysis includes data cleaning, visualization, and correlation analysis.

1. What is Exploratory Data Analysis (EDA)?

Exploratory Data Analysis (EDA) is an approach to analyzing data sets to summarize their main characteristics, often using visual methods. It helps in understanding the data, identifying patterns, spotting anomalies, and testing hypotheses. EDA is a crucial step in the data analysis process.

2. EDA Step-by-Step Process

1. **Data Collection**: Gather the data from various sources.

2. **Data Cleaning**: Handle missing values, remove duplicates, and correct inconsistencies.

3. **Data Overview**: Understand the structure of the data using summary statistics and data types.

4. **Univariate Analysis**: Analyze individual variables using histograms, box plots, etc.

5. **Bivariate Analysis**: Explore relationships between two variables using scatter plots, correlation matrices, etc.

6. **Multivariate Analysis**: Investigate interactions among multiple variables.

7. **Feature Engineering**: Create new features or modify existing ones to improve model performance.

8. **Data Visualization**: Use visual tools to present findings and insights.

3. Importing Libraries

The following code imports the necessary libraries for data manipulation and visualization.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

4. Loading the Dataset

This code loads the red wine quality dataset into a pandas DataFrame and displays the first few rows.

```
df = pd.read_csv('winequality-red.csv')
df.head()
```

5. Data Overview

The following code provides an overview of the dataset, including data types and summary statistics.

```
df.info()
df.describe()
```

6. Checking for Missing Values and Duplicates

This code checks for missing values and duplicate records in the dataset, and removes duplicates.

```
df.isnull().sum()
df[df.duplicated()]
df.drop_duplicates(inplace=True)
```

7. Correlation Analysis

This code calculates the correlation matrix and visualizes it using a heatmap to identify relationships between variables.

```
correlation_matrix = df.corr()
plt.figure(figsize=(10,6))
sns.heatmap(correlation_matrix, annot=True)
plt.show()
```

8. Data Visualization

This code visualizes the distribution of wine quality scores using a bar plot.

```
df.quality.value_counts().plot(kind='bar')
plt.xlabel('Wine Quality')
plt.ylabel('Count')
plt.show()
```

9. Univariate and Bivariate Analysis

This code performs univariate and bivariate analysis using pair plots and box plots.

```
sns.pairplot(df)
sns.catplot(x='quality', y='alcohol', data=df, kind='box')
```

10. Scatter Plot Analysis

This code creates a scatter plot to visualize the relationship between alcohol content and pH, colored by wine quality.

```
sns.scatterplot(x='alcohol', y='pH', hue='quality', data=df)
plt.show()
```

11. Glossary

1. **Exploratory Data Analysis (EDA)**: A statistical approach to analyze data sets to summarize their main characteristics. EDA uses visual methods to understand data distributions and relationships.

2. **Data Cleaning**: The process of correcting or removing inaccurate records from a dataset. This includes handling missing values, duplicates, and inconsistencies.

3. **Univariate Analysis**: The analysis of a single variable to summarize its characteristics, often using visualizations like histograms and box plots.

4. **Bivariate Analysis**: The analysis of the relationship between two variables, typically visualized using scatter plots or correlation coefficients.

5. **Correlation Matrix**: A table showing correlation coefficients between variables, indicating the strength and direction of their relationships.

6. **Data Visualization**: The graphical representation of information and data, which helps in understanding trends, patterns, and insights from the data.

Conclusion

The exploratory data analysis of the red wine quality dataset reveals insights into the relationships between physicochemical properties and wine quality. Further analysis can be conducted to build predictive models.

EDA and Feature Engineering for Flight Price Prediction

This report covers the exploratory data analysis (EDA) and feature engineering process for predicting flight prices using a dataset obtained from Ease My Trip.

Importing Basic Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

This code imports the necessary libraries for data manipulation, visualization, and analysis.

Loading the Dataset

```
df = pd.read_excel('flight_price.xlsx')
df.head()
```

This code loads the flight price dataset from an Excel file and displays the first few rows.

Data Overview

```
df.info()
```

This code provides a summary of the dataset, including the number of entries and data types.

Descriptive Statistics

```
df.describe()
```

This code generates descriptive statistics for the numerical features in the dataset.

Feature Engineering

```
df['Date'] = df['Date_of_Journey'].str.split('/').str[0]
df['Month'] = df['Date_of_Journey'].str.split('/').str[1]
df['Year'] = df['Date_of_Journey'].str.split('/').str[2]
```

This code extracts the day, month, and year from the 'Date_of_Journey' column.

```
df['Date'] = df['Date'].astype(int)
df['Month'] = df['Month'].astype(int)
df['Year'] = df['Year'].astype(int)
```

This code converts the extracted date components to integer data types.

Dropping Unnecessary Columns

```
df.drop('Date_of_Journey', axis=1, inplace=True)
```

This code drops the 'Date_of_Journey' column as it is no longer needed.

Processing Arrival Time

```
df['Arrival_Time'] = df['Arrival_Time'].apply(lambda x: x.split(' ')[0])
df['Arrival_Hour'] = df['Arrival_Time'].str.split(':').str[0]
df['Arrival_Min'] = df['Arrival_Time'].str.split(':').str[1]
```

This code processes the 'Arrival_Time' column to extract hours and minutes.

Processing Departure Time

```
df['Departure_Hour'] = df['Dep_Time'].str.split(':').str[0]
df['Departure_Min'] = df['Dep_Time'].str.split(':').str[1]
```

This code extracts hours and minutes from the 'Dep_Time' column.

Handling Total Stops

```
df['Total_Stops'].mode()
df['Total_Stops'] = df['Total_Stops'].map({'non-stop': 0, '1 stop': 1, '2 stops': 2,
                                           '3 stops': 3, '4 stops': 4, np.nan: 1})
```

This code maps the 'Total_Stops' categorical values to numerical values for analysis.

One-Hot Encoding Categorical Features

```
from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder()
encoded_features = encoder.fit_transform(df[['Airline', 'Source', 'Destination']])
.toarray()
```

This code applies one-hot encoding to the categorical features 'Airline', 'Source', and 'Destination'.

Saving Encoded Data to Excel

```
an.to_excel('an.xlsx')
```

This code saves the encoded features to an Excel file for further analysis.

Conclusion

This report provided a comprehensive overview of the exploratory data analysis and feature engineering process for flight price prediction. The steps included data loading, processing, and encoding, which are essential for building predictive models.

Glossary

EDA: Exploratory Data Analysis: A technique used to analyze datasets to summarize their main characteristics, often using visual methods.

Feature Engineering: The process of using domain knowledge to extract features from raw data that make machine learning algorithms work.

One-Hot Encoding: A method of converting categorical variables into a form that could be provided to ML algorithms to do a better job in prediction.

DataFrame: A two-dimensional, size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns) in pandas.

Pandas: A fast, powerful, flexible, and easy-to-use open-source data analysis and manipulation tool built on top of the Python programming language.

Google Play Store Apps Analysis Report

Problem Statement

Today, 1.85 million different apps are available for users to download. Android users have even more from which to choose, with 2.56 million available through the Google Play Store. Our objective is to find the most popular category, the app with the largest number of installs, the app with the largest size, etc.

Data Collection

The data consists of 20 columns and 10841 rows.

Steps We Are Going to Follow

- Data Cleaning
- Exploratory Data Analysis
- Feature Engineering

Code Snippet: Importing Libraries and Loading Data

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

warnings.filterwarnings('ignore')

df = pd.read_csv('https://raw.githubusercontent.com/krishnaik06/playstore-Dataset/
main/googleplaystore.csv')
df.head()
```

Data Overview

The dataset has missing values and consists of various features such as App, Category, Rating, Reviews, Size, Installs, Type, Price, Content Rating, Genres, Last Updated, Current Ver, and Android Ver.

Code Snippet: Data Cleaning

```
# Handling Missing Values
df.isnull().sum()
```

```
# Convert Reviews to Integer
df['Reviews'] = df['Reviews'].astype(int)

# Convert Size to Float
df['Size'] = df['Size'].str.replace('M', '000').str.replace('k', '').astype(float)

# Convert Installs and Price
df['Installs'] = df['Installs'].str.replace('+', '').str.replace(',', '').astype(int)
df['Price'] = df['Price'].str.replace('$', '').astype(float)
```

Handling Last Update Feature

```
# Convert Last Updated to datetime
df['Last Updated'] = pd.to_datetime(df['Last Updated'])
df['Day'] = df['Last Updated'].dt.day
df['Month'] = df['Last Updated'].dt.month
df['Year'] = df['Last Updated'].dt.year
```

Exploratory Data Analysis (EDA)

In this section, we will explore the dataset to find insights such as the most popular app category, the app with the largest number of installs, and more.

Code Snippet: EDA

```
# Most Popular Category
df['Category'].value_counts().plot.pie(figsize=(15, 16), autopct='%1.1f%%')

# Top 10 App Categories
category = pd.DataFrame(df['Category'].value_counts())
category.rename(columns={'Category': "Count"}, inplace=True)

plt.figure(figsize=(15, 6))
sns.barplot(x=category.index[:10], y='Count', data=category[:10], palette='hls')
plt.title('Top 10 App Categories')
plt.xticks(rotation=90)
plt.show()
```

Insights

Insights from the EDA show that the Family category has the most number of apps, followed by Games and Tools. The Beauty category has the least number of apps.

Internal Assignments

1. Which Category has the largest number of installations? 2. What are the Top 5 most installed Apps in Each popular Category? 3. How many apps are there on Google Play Store which get 5 ratings?

Code Snippet: Installations Analysis

```
# Which Category has the largest number of installations?
df_cat_installs = df.groupby(['Category'])['Installs']
.sum().sort_values(ascending=False).reset_index()

df_cat_installs.Installs = df_cat_installs.Installs /
1000000000  # converting into billions

plt.figure(figsize=(14, 10))
sns.barplot(x='Installs', y='Category', data=df_cat_installs.head(10))
plt.xlabel('No. of Installations in Billions')
plt.title("Most Popular Categories in Play Store", size=20)
plt.show()
```

Code Snippet: Top 5 Most Installed Apps

```
# Top 5 most installed Apps in Each popular Category
dfa = df.groupby(['Category', 'App'])['Installs'].sum().reset_index()
dfa = dfa.sort_values('Installs', ascending=False)
apps = ['GAME', 'COMMUNICATION', 'PRODUCTIVITY', 'SOCIAL']

plt.figure(figsize=(40, 30))
for i, app in enumerate(apps):
    df2 = dfa[dfa.Category == app]
    df3 = df2.head(5)
    plt.subplot(4, 2, i + 1)
    sns.barplot(data=df3, x='Installs', y='App')
    plt.xlabel('Installation in Millions')
    plt.title(app, size=20)

plt.tight_layout()
plt.subplots_adjust(hspace=.3)
plt.show()
```

Code Snippet: 5 Rated Apps

```
# How many apps are there on Google Play Store which get 5 ratings?
rating = df.groupby(['Category', 'Installs', 'App'])['Rating']
.sum().sort_values(ascending=False).reset_index()
toprating_apps = rating[rating.Rating == 5.0]
print('Number of 5 rated apps:', toprating_apps.shape[0])
toprating_apps.head(1)
```

Key Insights

1. Family category has the most apps (18%) followed by Games (11%).
2. Game category leads in installations with ~35 billion installs.
3. Subway Surfers is the most installed game with 1B+ installs.
4. 271 apps have perfect 5-star ratings.
5. Rating distribution shows most apps have 4-4.5 ratings.
6. 92.6% of apps are free, while 7.4% are paid.

Glossary

App: A software application designed to run on mobile devices.

Category: The classification of apps based on their functionality.

Rating: A score given to an app based on user reviews.

Reviews: The number of user feedback entries for an app.

Size: The storage space required by the app.

Installs: The total number of times the app has been downloaded.

Type: Indicates whether the app is free or paid.

Price: The cost of the app if it is paid.

Content Rating: The age group for which the app is appropriate.

Genres: The specific type of content the app provides.

Last Updated: The date when the app was last modified.

Current Ver: The current version of the app available.

Android Ver: The minimum Android version required to run the app.

Conclusion

In conclusion, the analysis of the Google Play Store dataset provides valuable insights into app categories, ratings, and installations. This information can be useful for developers and marketers in understanding trends and user preferences.