Here's an example of how you can apply some EDA (Exploratory Data Analysis) processing on a dataset using Python:

# Import necessary libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# Load the dataset

df = pd.read\_csv('your\_data.csv')

# View the first few rows of the dataset

print(df.head())

# Check for missing values

print(df.isnull().sum())

# Get summary statistics of the dataset

print(df.describe())

# Plot histograms for each column

df.hist(figsize=(10, 8), bins=50)

# Plot pairplots for each column

sns.pairplot(df)

# Plot correlation matrix

corr\_matrix = df.corr()

plt.figure(figsize=(10, 8))

sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm')

# Identify outliers using boxplots

df.plot(kind='box', subplots=True, layout=(3, 3), figsize=(10, 8))

# Identify correlations between columns

corr\_matrix = df.corr()

print(corr\_matrix)

# Visualize the distribution of each column

for col in df.columns

plt.figure(figsize=(8, 6))

sns.distplot(df[col])

plt.title(f'Distribution of {col}')

plt.show()

This code applies the following EDA techniques:

1. Viewing the dataset: It loads the dataset and views the first few rows using df.head().

2. Checking for missing values: It checks for missing values using df.isnull().sum().

3. Getting summary statistics: It gets summary statistics of the dataset using df.describe().

4. Plotting histograms: It plots histograms for each column using df.hist().

5. Plotting pairplots: It plots pairplots for each column using sns.pairplot().

6. Plotting correlation matrix: It plots the correlation matrix using sns.heatmap().

7. Identifying outliers: It identifies outliers using boxplots using df.plot().

8. Identifying correlations: It identifies correlations between columns using df.corr().

9. Visualizing distributions: It visualizes the distribution of each column using sns.distplot().

Here are some examples of how you can find more relevant data or graph set values in Python:

Finding Relevant Data

1. Correlation Analysis

You can use the corr() function from Pandas to calculate the correlation between different columns in your dataset.

import pandas as pd

df = pd.read\_csv('your\_data.csv')

correlation = df.corr()

print(correlation)

2. Feature Importance

You can use the feature\_importances\_ attribute from Scikit-learn's RandomForestClassifier or RandomForestRegressor to find the most important features in your dataset.

from sklearn.ensemble import RandomForestClassifier

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

# Load the iris dataset

iris = load\_iris()

X = iris.data

y = iris.target

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train a random forest classifier

rfc = RandomForestClassifier(n\_estimators=100, random\_state=42)

rfc.fit(X\_train, y\_train)

print(rfc.feature\_importances\_)

3. Mutual Information

You can use the mutual\_info\_classif() or mutual\_info\_regression() functions from Scikit-learn to calculate the mutual information between different columns in your dataset.

from sklearn.feature\_selection import mutual\_info\_classif

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

iris = load\_iris()

X = iris.data

y = iris.target

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Calculate the mutual information

mi = mutual\_info\_classif(X\_train, y\_train)

# Print the mutual information

print(mi)

Graph Set Values

1. Box Plots

You can use the boxplot() function from Matplotlib to create box plots for different columns in your dataset.

import matplotlib.pyplot as plt

import pandas as pd

# Load your dataset

df = pd.read\_csv('your\_data.csv')

plt.boxplot(df['your\_column'])

plt.title('Box Plot of Your Column')

plt.show()

2. Histograms

You can use the hist() function from Matplotlib to create histograms for different columns in your dataset.

import matplotlib.pyplot as plt

import pandas as pd

df = pd.read\_csv('your\_data.csv')

plt.hist(df['your\_column'], bins=50)

plt.title('Histogram of Your Column')

plt.show()

3. Scatter Plots

You can use the scatter() function from Matplotlib to create scatter plots for different columns in your dataset.

import matplotlib.pyplot as plt

import pandas as pd

df = pd.read\_csv('your\_data.csv')

# Create a scatter plot

plt.scatter(df['your\_column1'], df['your\_column2'])

plt.title('Scatter Plot of Your Columns')

plt.show()

**MAE (Mean Absolute Error**)

import numpy as np

from sklearn.metrics import mean\_absolute\_error

# Predicted values

y\_pred = np.array([1.2, 2.3, 3.1, 4.2, 5.3])

# Actual values

y\_true = np.array([1.0, 2.0, 3.0, 4.0, 5.0])

# Calculate MAE

mae = mean\_absolute\_error(y\_true, y\_pred)

print("MAE:", mae)

**R^2 (Coefficient of Determination)**

import numpy as np

from sklearn.metrics import r2\_score

# Predicted values

y\_pred = np.array([1.2, 2.3, 3.1, 4.2, 5.3])

# Actual values

y\_true = np.array([1.0, 2.0, 3.0, 4.0, 5.0])

**Finding Keys**

Finding Keys in a Dictionary

You can use the keys() method to find all the keys in a dictionary.

data = {'name': 'John', 'age': 30, 'city': 'New York'}

keys = data.keys()

print(keys) # Output: dict\_keys(['name', 'age', 'city'])

**Finding Keys in a JSON Object**

You can use the json.loads() function to parse a JSON string into a Python dictionary, and then use the keys() method to find all the keys.

import json

json\_data = '{"name": "John", "age": 30, "city": "New York"}'

data = json.loads(json\_data)

keys = data.keys()

print(keys) # Output: dict\_keys(['name', 'age', 'city'])

**Clearing Data**

You can use the clear() method to clear all the key-value pairs from a dictionary.

data = {'name': 'John', 'age': 30, 'city': 'New York'}

data.clear()

print(data) # Output: {}

**Clearing a List**

You can use the clear() method or assign an empty list to the variable to clear all the elements from a list.

data = [1, 2, 3, 4, 5]

data.clear()

print(data) # Output: []

data = [1, 2, 3, 4, 5]

data = []

print(data) # Output: []