full data analysis process

**Step 1: Understanding the Data**

* Load the dataset using pandas:
* import pandas as pd
* df = pd.read\_csv("your\_dataset.csv") # Change file format as needed
* Check the first few rows:
* print(df.head())
* print(df.info()) # Data types, missing values, etc.
* print(df.describe()) # Summary statistics
* Identify categorical vs. numerical columns.

**Step 2: Handling Missing Values**

* **Check missing values**:
* print(df.isnull().sum())
* **Decide how to handle them**:
  + Drop columns with too many missing values:
  + df.drop(columns=["col\_to\_drop"], inplace=True)
  + Fill missing values:
  + df["numeric\_col"].fillna(df["numeric\_col"].mean(), inplace=True) # Mean for numerical
  + df["categorical\_col"].fillna(df["categorical\_col"].mode()[0], inplace=True) # Mode for categorical
  + Forward/backward fill:
  + df.fillna(method='ffill', inplace=True) # Forward fill
  + df.fillna(method='bfill', inplace=True) # Backward fill

**Step 3: Handling Duplicates & Outliers**

* **Remove duplicates**:
* df.drop\_duplicates(inplace=True)
* **Detect outliers using IQR**:
* Q1 = df["numeric\_col"].quantile(0.25)
* Q3 = df["numeric\_col"].quantile(0.75)
* IQR = Q3 - Q1
* df = df[(df["numeric\_col"] >= (Q1 - 1.5 \* IQR)) & (df["numeric\_col"] <= (Q3 + 1.5 \* IQR))]

**Step 4: Feature Engineering & Encoding**

* **Convert categorical to numerical**:
* df = pd.get\_dummies(df, columns=["categorical\_col"], drop\_first=True) # One-hot encoding
* **Standardization & Normalization**:
* from sklearn.preprocessing import StandardScaler, MinMaxScaler
* scaler = StandardScaler()
* df["scaled\_col"] = scaler.fit\_transform(df[["numeric\_col"]])

**Step 5: Exploratory Data Analysis (EDA)**

**1. Univariate Analysis**

* **Histogram for distribution**:
* import matplotlib.pyplot as plt
* import seaborn as sns
* plt.figure(figsize=(8, 5))
* sns.histplot(df["numeric\_col"], bins=30, kde=True)
* plt.show()
* **Boxplot for outliers**:
* sns.boxplot(x=df["numeric\_col"])
* plt.show()

**2. Bivariate & Multivariate Analysis**

* **Correlation heatmap**:
* plt.figure(figsize=(10, 6))
* sns.heatmap(df.corr(), annot=True, cmap="coolwarm", fmt=".2f")
* plt.show()
* **Scatter plot for relationships**:
* sns.scatterplot(x=df["feature1"], y=df["feature2"], hue=df["category"])
* plt.show()
* **Pairplot**:
* sns.pairplot(df)
* plt.show()

**Step 6: Model Preparation (if needed)**

* **Splitting data**:
* from sklearn.model\_selection import train\_test\_split
* X = df.drop("target\_col", axis=1)
* y = df["target\_col"]
* X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)
* **Feature scaling**:
* from sklearn.preprocessing import StandardScaler
* scaler = StandardScaler()
* X\_train = scaler.fit\_transform(X\_train)
* X\_test = scaler.transform(X\_test)

**Step 7: Advanced Analysis (Optional)**

* **PCA for dimensionality reduction**:
* from sklearn.decomposition import PCA
* pca = PCA(n\_components=2)
* X\_pca = pca.fit\_transform(X)
* **Clustering (e.g., K-Means)**:
* from sklearn.cluster import KMeans
* kmeans = KMeans(n\_clusters=3)
* df["cluster"] = kmeans.fit\_predict(X)

**Final Thoughts**

* This is a **general framework**, and the specifics will depend on your dataset.
* Let me know **what step you're at**, and I can guide you deeper into it. 🚀