**Exploratory Data Analysis (EDA) Notebook Documentation**

### Code

import os  
import pandas as pd  
import numpy as np  
  
# Step 1: Define project root path  
project\_root = os.path.abspath(os.path.join(os.getcwd(), ".."))  
  
# Step 2: Define data paths  
features\_path = os.path.join(project\_root, "data", "processed", "cleaned\_features.csv")  
labels\_path = os.path.join(project\_root, "data", "processed", "encoded\_labels.npy")  
tags\_path = os.path.join(project\_root, "data", "processed", "tag\_classes.txt")  
  
# Step 3: Load data  
X = pd.read\_csv(features\_path)  
Y = np.load(labels\_path)  
  
# Step 4: Load tag classes  
with open(tags\_path, 'r') as f:  
 tag\_classes = f.read().splitlines()  
  
# Step 5: Confirm loading  
print("✅ Data loaded successfully.")  
print(f"📄 Features shape: {X.shape}")  
print(f"🏷️ Labels shape: {Y.shape}")  
print(f"🔖 Number of tag classes: {len(tag\_classes)}")

### Description

This code loads the cleaned feature matrix (X), multi-label encoded targets (Y), and the list of tag classes into memory. The directory paths are dynamically constructed from the project root. After loading, it prints the shapes of the loaded data to confirm success.

### Code

import matplotlib.pyplot as plt  
import seaborn as sns  
  
# Set style  
sns.set(style="whitegrid")  
  
# 1. Feature Sparsity (percentage of zeros)  
feature\_sparsity = 100.0 \* (X == 0).sum().sum() / (X.shape[0] \* X.shape[1])  
print(f"Feature matrix sparsity: {feature\_sparsity:.2f}%")  
  
# 2. Number of tags per sample  
tags\_per\_sample = Y.sum(axis=1)  
print(f"\n Tags per sample (mean ± std): {tags\_per\_sample.mean():.2f} ± {tags\_per\_sample.std():.2f}")  
  
# 3. Plot: Distribution of tags per sample  
plt.figure(figsize=(10, 5))  
sns.histplot(tags\_per\_sample, bins=20, kde=False, color="skyblue")  
plt.title("Distribution of Tags per Sample")  
plt.xlabel("Number of Tags")  
plt.ylabel("Frequency")  
plt.show()  
  
# 4. Frequency of tags across all samples  
tag\_frequencies = Y.sum(axis=0)  
most\_common\_tags = np.argsort(tag\_frequencies)[::-1][:10]  
least\_common\_tags = np.argsort(tag\_frequencies)[:10]  
  
# Display top and bottom tags  
print("\n Most common tags:")  
for i in most\_common\_tags:  
 print(f"{tag\_classes[i]} → {int(tag\_frequencies[i])} occurrences")  
  
print("\n Least common tags:")  
for i in least\_common\_tags:  
 print(f"{tag\_classes[i]} → {int(tag\_frequencies[i])} occurrence(s)")

### Description

This block performs several EDA tasks:

1. Calculates feature sparsity (how many zeros are in the feature matrix).
2. Computes the number of labels assigned to each sample.
3. Plots the distribution of label counts.
4. Displays the most and least frequent mutation tags across the dataset.

### Code

import numpy as np  
import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
# Select a sample of the first 500 columns (as dense)  
X\_dense\_sample = X.iloc[:, :500].values  
  
# Create a DataFrame for easier correlation plotting  
X\_df = pd.DataFrame(X\_dense\_sample)  
  
# Compute correlation matrix  
corr\_matrix = X\_df.corr()  
  
# Plot heatmap (limit max value for visual clarity)  
plt.figure(figsize=(12, 10))  
sns.heatmap(corr\_matrix, cmap='coolwarm', center=0, cbar=True, xticklabels=False, yticklabels=False)  
plt.title("Correlation Heatmap (Sample of 500 Features)")  
plt.show()

### Description

This code visualizes the correlation matrix of the first 500 features in the dataset. This subset is selected to avoid overwhelming the plot due to high dimensionality. It helps to identify multicollinearity or patterns in feature relationships.

### Code

from sklearn.decomposition import TruncatedSVD  
from scipy.sparse import csr\_matrix  
  
# Convert to sparse matrix  
X\_numeric = X.apply(pd.to\_numeric, errors='coerce').fillna(0)  
X\_sparse = csr\_matrix(X\_numeric)  
  
# Reduce dimensions to 150  
svd = TruncatedSVD(n\_components=150, random\_state=42)  
X\_reduced = svd.fit\_transform(X\_sparse)  
  
print(f"Dimensionality reduced. New shape: {X\_reduced.shape}")  
print(f"🔍 Variance explained: {svd.explained\_variance\_ratio\_.sum():.4f}")

### Description

Applies **Truncated SVD** to reduce the high-dimensional, sparse feature matrix to 150 dimensions. This aids in visualization, modeling efficiency, and reducing noise. It also prints the cumulative variance explained by the components.

### Code

import joblib  
import os  
  
# Create a directory to store artifacts if it doesn't exist  
os.makedirs("../artifacts", exist\_ok=True)  
  
# Save reduced features and labels  
joblib.dump(X\_reduced, "../artifacts/X\_reduced.pkl")  
joblib.dump(Y, "../artifacts/Y.pkl")  
  
print("✅ X\_reduced and Y saved to ../artifacts/")

### Description

Saves the processed and dimensionally reduced feature set (X\_reduced) and label matrix (Y) into an artifacts directory using joblib. This ensures the data is readily available for modeling steps without re-running preprocessing.