**DATA ANALYSIS BASIC CHEATSHEET**

**Way to use dataset in google colab from google drive**

from google.colab import drive

drive.mount('/content/drive')

# Example path to your dataset:

path = '/content/drive/MyDrive/dataset.csv'

data = pd.read\_csv(path)

data.head()

**1. Importing Essentials**

data = pd.read\_csv('file.csv') # Load CSV

data.head() # First 5 rows

data.tail() # Last 5 rows

data.shape # (rows, columns)

data.columns # Column names

data.info() # Data types + nulls

data.describe() # Summary stats

data.dtypes # Data types

data.index # Index range

2. Load & Inspect Data

data = pd.read\_csv('file.csv') # Load CSV

data.head() # First 5 rows

data.tail() # Last 5 rows

data.shape # (rows, columns)

data.columns # Column names

data.info() # Data types + nulls

data.describe() # Summary stats

data.dtypes # Data types

data.index # Index range

3. Checking Missing/Duplicate values:

data.isnull().sum() # Count missing values per column

data.notnull().sum() # Count non-null values

data.duplicated().sum() # Count duplicate rows

data.drop\_duplicates(inplace=True) # Remove duplicates

data.fillna(0, inplace=True) # Fill NaN with 0 (example)

4. Data Selection & Filtering

data['column'] # Single column

data[['col1', 'col2']] # Multiple columns

data.iloc[0] # Select by index position

data.loc[0] # Select by index label

data.iloc[:, 0:3] # Select columns by index range

data.loc[data['age'] > 30] # Conditional filtering

data.query('age > 30 and gender == "Male"') # Query filtering

5. Sorting and Indexing

data.sort\_values(by='age', ascending=False)

data.reset\_index(drop=True, inplace=True)

data.set\_index('id', inplace=True)

6. Grouping Aggregration and Counting

data['column'].value\_counts() # Frequency count

data['column'].value\_counts(normalize=True) # Percentage

data.groupby('gender')['age'].mean() # Mean age by gender

data.groupby(['gender', 'city']).size() # Group size

data.groupby('department').agg({'salary': ['mean', 'max', 'min']})

7. Cleaning and Encoding

data.rename(columns={'old': 'new'}, inplace=True)

data['gender'].replace({'M': 'Male', 'F': 'Female'}, inplace=True)

pd.get\_dummies(data, columns=['category'], drop\_first=True) # One-hot encoding

8. Visualization Quickies

data.corr() # Correlation matrix

data.cov() # Covariance matrix

data['col'].mean()

data['col'].median()

data['col'].mode()

9.Correlation & Stats

data.corr() # Correlation matrix

data.cov() # Covariance matrix

data['col'].mean()

data['col'].median()

data['col'].mode()

10. Merging joining and concatenation

pd.concat([df1, df2], axis=0) # Stack rows

pd.concat([df1, df2], axis=1) # Combine columns

pd.merge(df1, df2, on='id', how='inner') # SQL-style merge

11. Sampling and Randomization

data.sample(5) # Random 5 rows

data.sample(frac=0.1) # Random 10% sample

Thanks, both, for meeting. I appreciate we're all working hard on this project but I’ve noticed that sometimes we’re using different terms or working at different paces, and I think that is what’s causing a bit of confusion among us. I think the best way for us would be if you can explain all the technical stuff in simple way as possible and how about we make a timeline and proper plan for the project to complete it on time.

I think if we do that, we can stay productive, avoid miscommunication, and keep things on schedule.

“The new customer portal project is going well, and we’re still aiming to finish by August 10. The team has already completed the main parts of the website, like the login page and dashboard. We’re now working on connecting everything together, but the company helping us with that part said they’ll be about two weeks late. The testing team has started checking things but doesn’t have as many people as planned, so they’ll be updating their staffing plan soon. For training, sessions are set for early August, and the materials are being prepared. Before our next meeting, the vendor will confirm their new delivery date, the testing lead will update their plan, and the training coordinator will share the draft training materials for everyone to review.

Overall, things are progressing well, and with a bit of coordination, I’m confident we’ll stay on track and meet our goals on time.”

# Dataset shape

print("Shape:", data.shape)

# Data types

print("\nData types:\n", data.dtypes)

# Basic summary statistics

print("\nSummary statistics:\n", data.describe())

# Null values

print("\nMissing values:\n", data.isnull().sum())

# Unique values per column

print("\nUnique values per column:\n", data.nunique())

# Duplicates

print("\nDuplicate rows:", data.duplicated().sum())

Data analysis:

Transform to lowercase:

Remove html tags:

Remove urls

Removing non words and non-whitespace charcters

Remove digits

2. Tokenize it

3. Remove stopwords: ask to remove stop words(nltk, spacy and scikit learn can be used to remove stopwords)

4. Stemming

There are various algorithms that can be used for stemming,

· Porter Stemmer algorithm

5. Lemmatization

import nltk  
nltk.download('averaged\_perceptron\_tagger')  
import nltk  
nltk.download('wordnet')

1. NLP (Text) Dataset – Analysis & Preprocessing

📍 A. Data Understanding & Analysis

Step What to Check Why It Matters

Dataset structure Number of samples, labels, text length Helps pick appropriate models

Text length distribution Min/max/mean word counts BERT has token limits

Class distribution Imbalance check Affects model training fairness

Language & encoding Detect UTF-8 issues Avoid tokenization errors

Noise patterns Emojis, URLs, special chars, slang Plan cleaning strategy

Vocabulary richness Unique token count, rare words Guides embedding techniques

🔍 Tools: pandas, matplotlib, seaborn, wordcloud, langdetect

🛠️ B. Preprocessing Techniques

Task Techniques When to Use

Text cleaning Remove HTML tags, URLs, numbers, emojis General preprocessing

Case normalization Lowercasing Most NLP models except GPT-style

Tokenization WordPiece, BPE, SpaCy Mandatory

Stopword removal English stopword lists Traditional ML (not for transformers)

Lemmatization/Stemming NLTK, SpaCy Classic ML pipelines

Rare & frequent word filtering Thresholding Bag-of-words models

Handling contractions e.g., "can't" → "cannot" Better semantic clarity

Spelling correction SymSpell, TextBlob Noisy datasets

Sequence padding/truncation pad\_sequences DL-based NLP

Embedding generation Word2Vec, GloVe, BERT Model-ready format

📦 Tools: nltk, spacy, transformers, gensim, tensorflow.text

📉 2. Time Series Forecasting Dataset – Analysis & Preprocessing

📍 A. Exploratory Checkpoints

Check Purpose

Time format parsing Ensure datetime index

Frequency check Daily, hourly, minutely

Missing timestamp detection For imputation

Trend detection Linear/non-linear growth

Seasonality patterns Weekly, yearly, hourly

Noise/random spikes Consider smoothing

Stationarity test ADF/KPSS test

Outlier detection Z-score/IQR/IsolationForest

📊 Visualizations: Line plots, lag plots, ACF/PACF graphs, seasonal decomposition.

🛠️ B. Preprocessing Techniques

Task Techniques Use Case

Missing data Interpolation, ffill/bfill, Kalman filter Continuous TS

Outliers Winsorization, isolation forest Improve stability

Resampling .resample('D').mean() Uniformity

Differencing First/second order Stationarity for ARIMA

Detrending Regression removal Traditional forecasting

Deseasonalizing STL decomposition Classic TS

Normalization MinMax/StandardScaling Neural TS models

Feature engineering Lag features, rolling mean, EWMA LSTM/ML models

Time features Hour, weekday, month Add season/influence

Train-test split Time-based split (no shuffling) Avoid leakage

📦 Tools: statsmodels, pandas, tsfresh, sklearn, darts, prophet

🖼️ 3. Computer Vision Dataset – Analysis & Preprocessing

📍 A. Dataset Exploration

Check Importance

Image dimensions Uniformity required

Color mode RGB vs grayscale

Class balance Avoid model bias

Image quality Blurry/noisy data detection

Object position/density Helps in augmentation planning

Duplicate/incorrect labels Data cleaning

🔍 Do EDA using: sample grid plots, class-wise image count, pixel intensity distributions.

🛠️ B. Preprocessing Pipeline

Step Techniques Purpose

Standard resizing Resize(224×224) CNN compatibility

Normalization Divide by 255 or Z-score Faster convergence

Data augmentation Flip, crop, rotate, shear, CutMix Prevent overfitting

Handling imbalance Oversampling + augment minority class Balance

Noise reduction Gaussian blur, denoising autoencoders Enhance features

Edge enhancement Sobel, Canny Boost structural learning

Color corrections Histogram equalization Medical/sat imagery

Object detection prep Labeling (YOLO/COCO/PASCAL VOC) Detection tasks

JPG artifacts filtering DCT corrections (advanced) Cleaner features

📦 Tools: opencv, albumentations, torchvision, tensorflow.keras.preprocessing