BASIC **UNDERSTANDING OF DATA**

import pandas as pd

df = pd.read\_csv('train.csv')

**#How big is the data?**

len(df)

df.shape

df.count()

df.max()

df.min()

df.value\_counts()

**#How does the data look like?**

df.sample(50)

df.head()

df.tail()

**#What is the data type of cols?**

df.info()

**#are there any missing values?**

df.isnull()

df.isnull().sum()

**#How does the data look mathematically?**

df.describe()

df.describe(include=object)

data.mean()

data.std()

**#Are there duplicate values?**

df.duplicated().sum()

**#How is the correlation between cols?**

df.corr()

df.corr()['columnname']

df.std()

**OUTLIER DETECTION AND REMOVING**

**1) Z-Score (Apply only for uniform data/guassian data)**

upper\_limit=df['coloumn'].mean()+3\*df['coloumn'].std()

lower\_limit=df['coloumn'].mean()-3\*df['coloumn'].std()

# or

z\_score=df['coloum']-df['column'].mean()/df['column']

or

**Trimming for Z-Score z\_score between -1 and +1 otherwise all are outliers**

new\_df = df[(df['z\_score'] < 3) & (df['z\_score'] > -3)]

**2) IOR (apply non-uniform data)**

percentile25 = df['coloumn'].quantile(0.25)

percentile75 = df['coloumn'].quantile(0.75)

iqr = percentile75 - percentile25

lower\_limit = percentile25 - 1.5 \* iqr

upper\_limit = percentile75 + 1.5 \* iqr

**3) Percentile Method**

lower\_limit = df['coloumn'].quantile(0.01)

upper\_limit = df['coloumn'].quantile(0.99)

**#Clapping / Winsorization (clap or compreeses to range limit i.e on boundry)**

df['coloumn'] = np.where( df['coloumn']>upper\_limit, upper\_limit,

np.where( df['coloumn']<lower\_limit, lower\_limit,

df['coloumn']))

**#Trimming (remving rows having outliers from data)**

new\_df = df[(df['coloumn'] < upperlimit) & (df['coloumn'] > lower\_limit)]

new\_df

**EXPLORATORY DATA ANYLYSIS ( Using Seaborn)**

**import seaborn as sns**

**sns.set\_theme(style="darkgrid")**

**#Relational Plot**

sns.relplot(data=tips, x="total\_bill", y="tip")

sns.relplot(data=tips, x="total\_bill", y="tip", hue="smoker")

sns.relplot(data=dowjones, x="Date", y="Price", kind="line")

**#Displot**

sns.displot(penguins, x="flipper\_length\_mm", hue="species")

sns.displot(penguins, x="flipper\_length\_mm", kde=False)

sns.displot(penguins, x="flipper\_length\_mm", hist=False)

sns.displot(penguins, x="flipper\_length\_mm", kde=False, hist=False, rug=True )

**#Jointplot**

sns.jointplot(x="total\_bill", y="tip", data=df, kind="reg");

sns.jointplot(x="total\_bill", y="tip", data=df, kind="hex");

sns.jointplot(x="total\_bill", y="tip", data=df, kind="kde");

**#Pairplot**

sns.pairplot(df)

sns.pairplot(penguins,hue='speices')

**#Pairgrid**

g = sns.PairGrid(df)

g.map\_upper(sns.histplot)

g.map\_lower(sns.kdeplot, fill=True)

g.map\_diag(sns.histplot, kde=True)

**#Catogirical Plot**

sns.catplot(data=tips, x="day", y="total\_bill", hue="smoker", kind="swarm")

sns.catplot(data=tips, x="day", y="total\_bill", hue="smoker", kind="box")

sns.catplot(data=tips, x="day", y="total\_bill", hue="smoker", kind="boxen")

sns.catplot(data=tips, x="day", y="total\_bill", hue="smoker", kind="violin")

sns.catplot(data=tips, x="day", y="total\_bill", hue="smoker", kind="bar")

sns.catplot(data=tips, x="day", y="total\_bill", hue="smoker", kind="count")

sns.catplot(data=tips, x="day", y="total\_bill", hue="smoker", kind="point")

**#Linear\_Regression\_Fit**

sns.regplot(x="total\_bill", y="tip", data=tips);

sns.lmplot(x="total\_bill", y="tip", data=tips)

**#Heatmap**

sns.heatmap(df, annot=True, fmt="d", linewidths=.5, ax=ax)

#PLOTIFY EXPRESS

**EXPLORATORY DATA ANYLYSIS ( Using Plotify Express)**

**import plotly.express as px**

fig = px.line(df, x="species", y="petal\_width") **#Line Plot**

fig = px.bar(df, x="sepal\_width", y="sepal\_length") **#Bar Plot**

fig = px.histogram(df, x="sepal\_length", y="petal\_width") **#Histogram**

fig = px.scatter(df, x="species", y="petal\_width") **#Scatter plot**

fig = px.pie(df, values="total\_bill", names="day") **#Pie Plot**

fig = px.box(df, x="day", y="total\_bill") **#Box Plot**

fig = px.strip(df, x="total\_bill", y="day") **#Strip plot**

fig = px.violin(df, x="day", y="total\_bil **#Violin Plot**

fig = px.funnel(data, x='number', y='stage') **#Funnel plot**

fig = px.sunburst(df, path=['column'], values='total\_bill') **#Sunbrust**

fig = px.ecdf(df, x="total\_bill") **#ECDF**

fig = px.density\_heatmap(df, x="total\_bill", y="tip") **#Density Heatmap**

fig = px.density\_contour(df, x="total\_bill", y="tip") **#Density Contour**

fig = px.scatter\_ternary(df, a="Joly", b="Coderre",c="Berg") **#Scattter Triangle**

fig = px.area(df, x="year", y="pop", line\_group="country") **#Area Plot**

fig = px.parallel\_categories(df) **#Parallal Cat**

fig = px.scatter\_matrix(df) **#Scatter Matrix**

**3D Plot**

fig = **px.scatter\_3d**(df,x='sepal\_length',y='sepal\_width',z='petal\_width',color='species')

fig **= px.line\_3d**(df, x="gdpPercap", y="pop", z="year")

fig **= px.scatter\_matrix**(df,dimensions=["sepal\_length","sepal\_width"],color='id])

fig = **px.parallel\_coordinates**(df,dimensions=['sepal\_width','sepal\_length'],color='id')

**fig.show()**

**Importing Preprocessing Techniques**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

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

from sklearn.preprocessing import OneHotEncoder

from sklearn.preprocessing import LabelEncoder

from sklearn.preprocessing import OrdinalEncoder

from sklearn.preprocessing import MinMaxScaler **#normalization**

from sklearn.preprocessing import RobustScaler

from sklearn.preprocessing import MaxAbsScaler  **#standardization**

from sklearn.impute import SimpleImputer **#univariate analysis**

from sklearn.impute import MissingIndicator **#univariate analysis**

from sklearn.impute import KNNImputer **#multivariate analysis**

from sklearn.experimental import enable\_iterative\_imputer

from sklearn.impute import IterativeImputer **#multivariate analysis**

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline,make\_pipeline **#(pipe.named\_steps)**

from sklearn.preprocessing import PowerTransformer

from sklearn.preprocessing import FunctionTransformer

from sklearn.preprocessing import KBinsDiscretizer **#(also called as bining**)

from sklearn.model\_selection import GridSearchCV **#auto best** **imputation**

from sklearn.preprocessing import binarize

import datetime

Regression Algorithms

from sklearn.linear\_model import LinearRegression

from sklearn.linear\_model import Ridge

from sklearn.linear\_model import Lasso

from sklearn.linear\_model import ElasticNet

from sklearn.svm import SVR

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

import xgboost as xgb

from xgboost import XGBRegressor

import lightgbm as lgb

from lightgbm import LGBMRegressor

import tensorflow as tf

**Classification Algorithms :**

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

import xgboost as xgb

import lightgbm as lgb

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive\_bayes import GaussianNB

**Clustering Algorithms**

from sklearn.cluster import KMeans

from scipy.cluster.hierarchy import linkage, dendrogram

from sklearn.cluster import DBSCAN

from sklearn.mixture import GaussianMixture

import minisom

**Dimensiontionlity Reduction**

from sklearn.decomposition import PCA

from sklearn.manifold import TSNE

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

**Association Learning**

from mlxtend.frequent\_patterns import apriori

from mlxtend.frequent\_patterns import association\_rules

from mlxtend.frequent\_patterns import eclat

**Reinforcement Learning:**

**Deep Q-Networks (DQN)**

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

**Proximal Policy Optimization (PPO)**

import tensorflow as tf

from stable\_baselines3 import PPO

**Actor-Critic Methods**

import tensorflow as tf

from stable\_baselines3 import A2C

**Anomaly Detection Algorithms:**

from sklearn.ensemble import IsolationForest

from sklearn.svm import OneClassSVM

**Recommender Systems:**

**Collaborative Filtering**

from surprise import Dataset, Reader

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

from surprise import KNNBasic

**Content-Based Filtering**

Import necessary libraries/modules for content-based filtering

**Time Series Forecasting:**

**ARIMA (AutoRegressive Integrated Moving Average)**

from statsmodels.tsa.arima\_model import ARIMA

**LSTM-based Models (using TensorFlow)**

import numpy as np

import pandas as pd

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

**Supervised Learning Algorithms:**

**Regression:**

Linear Regression

Ridge Regression

Lasso Regression

Elastic Net Regression

Support Vector Regression

Decision Tree Regression

Random Forest Regression

Gradient Boosting Regression (e.g., XGBoost, LightGBM)

Neural Network Regression

**Classification:**

Logistic Regression

Decision Tree Classifier

Random Forest Classifier

Gradient Boosting Classifier (e.g., XGBoost, LightGBM)

Support Vector Machine (SVM)

k-Nearest Neighbors (k-NN)

Naive Bayes Classifier

Neural Network Classifier

**Unsupervised Learning Algorithms:**

**Clustering:**

k-Means Clustering

Hierarchical Clustering

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

Gaussian Mixture Model (GMM)

Self-Organizing Maps (SOM)

**Dimensionality Reduction:**

Principal Component Analysis (PCA)

t-Distributed Stochastic Neighbor Embedding (t-SNE)

Linear Discriminant Analysis (LDA)

Autoencoders

**Association Rule Learning:**

Apriori Algorithm

Eclat Algorithm

**Semi-Supervised Learning Strategies :**

Self-Training

Multi-View Learning

Co-Training

**Reinforcement Learning Algorithms:**

Q-Learning

Deep Q-Networks (DQN)

Proximal Policy Optimization (PPO)

Actor-Critic Methods

Monte Carlo Tree Search (MCTS)

**Deep Learning Architectures:**

Convolutional Neural Networks (CNN)

Recurrent Neural Networks (RNN)

Long Short-Term Memory (LSTM)

Gated Recurrent Units (GRU)

Transformer

**Ensemble Methods:**

Bagging (Bootstrap Aggregating)

Boosting (AdaBoost, Gradient Boosting, XGBoost, LightGBM)

Stacking

**Other Algorithms:**

Anomaly Detection Algorithms (Isolation Forest, One-Class SVM)

Recommender Systems (Collaborative Filtering, Content-Based Filtering)

Time Series Forecasting (ARIMA, LSTM-based models)

**Working with Different Files Formats**

import pandas as pd

df = pd.read\_csv('aug\_train.csv')

df = pd.read\_json()

df = pd.read\_html()

df = pd.read\_sql\_table()

Opening files from URL

import requests

from io import StringIO

url = "https://raw.githubusercontent.com/cs109/2014\_data/master/countries.csv"

headers = {"User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:66.0) Gecko/20100101 Firefox/66.0"}

req = requests.get(url, headers=headers)

data = StringIO(req.text)

pd.read\_csv(data)

Working with SQL

1) downoad sql data fromat file

2) go to XAMPP

3) go on localhost/phpmyadmin/

create a new database as name sql file

import and chose the sql file

go

4) install following code in jupyter or colab

!pip install mysql.connector

import mysql.connector

conn = mysql.connector.connect(host='localhost',user='root',password='',database='world')

df = pd.read\_sql\_query("SELECT \* FROM countrylanguage",conn)

df

**Working with Json files**

pd.read\_json('train.json')

#from url

pd.read\_json('https://api.exchangerate-api.com/v4/latest/INR')

**General prework on data**

**all data in row no columns**

pd.read\_csv('movie\_titles\_metadata.tsv',sep='\t',names=['sno','name','release\_year','rating','votes','genres'])

**Data with id (make it as index)**

pd.read\_csv('aug\_train.csv',index\_col='enrollee\_id')

**Using specific columns from data**

pd.read\_csv('aug\_train.csv',usecols=['enrollee\_id','gender','education\_level'])

**Skipping amount of rows from the data**

pd.read\_csv('aug\_train.csv',nrows=100)

pd.read\_csv('aug\_train.csv',skiprows=[0,1,2,3])

**Encoding not in utf fromat (unicode decode error)**

pd.read\_csv('zomato.csv',encoding='latin-1')

**Having problem with csv, encoding,unequal in rows and columns (paser erro)**

pd.read\_csv('zomato.csv',encoding='latin-1')

**Changing value types (flaot to int) (column -target)**

pd.read\_csv('aug\_train.csv',dtype={'target':int})

**Handing the date format (object as date to date fromat)**

pd.read\_csv('IPL Matches 2008-2020.csv',parse\_dates=['date'])

**Handling emty or - values in data to (not NaN)**

pd.read\_csv('aug\_train.csv',na\_values=['Male'])