

PSG COLLEGE OF TECHNOLOGY

**CRISP – DM REPORT**

**19Z610 – MACHINE LEARNING LABORATORY**

**NAME:** R K Ranjith

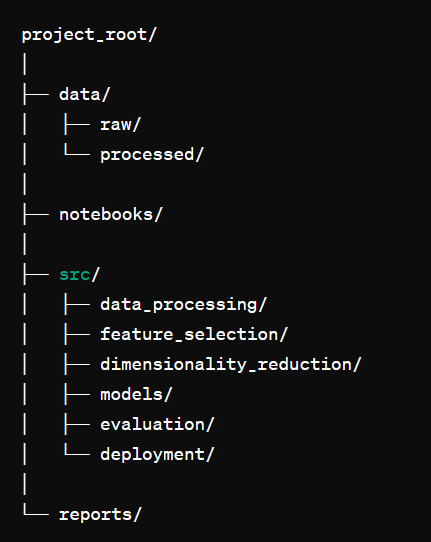
**ROLL NO.:** 21z335

**CLASS:** B.E CSE – G2

# Objective

The purpose of this report is to detail the deployment of a machine learning pipeline designed for regression tasks. This pipeline encompasses several key stages such as data comprehension, data preprocessing, model building, performance evaluation, hyperparameter optimization, unit and integration testing, as well as deployment. Our goal is to leverage a selection of widely used regression algorithms along with suitable methodologies for handling missing data, selecting relevant features, reducing dimensionality, and assessing model performance.

**Directory Structure**



# Design Patterns

To maintain a scalable and maintainable codebase, the pipeline utilizes design patterns such as:

1. Singleton Pattern: For managing configurations and global resources.
2. Factory Pattern: For creating instances of regressors and other objects dynamically.
3. Strategy Pattern: For interchangeable feature selection and dimensionality reduction techniques.

**Pipelines and Abstraction:**

To ensure the modularity and reusability of components, the pipeline is organized using pipelines and abstract methods. Each stage of the pipeline, including data preprocessing, feature selection, modeling, and evaluation, is encapsulated within distinct modules or classes, fostering a structured and maintainable codebase.

**Linters:**

To uphold coding standards and uphold code quality consistently, linting tools are utilized. Tools like Flake8 or Pylint are employed to detect and address issues pertaining to syntax errors, coding style adherence, and potential bugs across the codebase.

**Data Understanding:**

* The initial step in our data exploration process involves mounting Google Drive to access the dataset stored in the cloud, named 'fruit\_data\_with\_colors.txt'. This dataset contains information about a fruit and it’s information. Upon loading the dataset using Pandas, we generate a concise summary showcasing its dimensions and preview a sample of its initial rows to grasp its structure and content. Following this, we present summary statistics that highlight key numerical attributes such as mean, standard deviation, and quartile values. These statistics provide valuable insights into the data's distribution and variability, aiding in our understanding of its characteristics.
* Subsequently, we employ data visualization techniques to delve deeper into the dataset. Specifically, we leverage the Matplotlib and Seaborn libraries to create a histogram focusing

# Code:

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import seaborn as sns

# show up charts when export notebooks

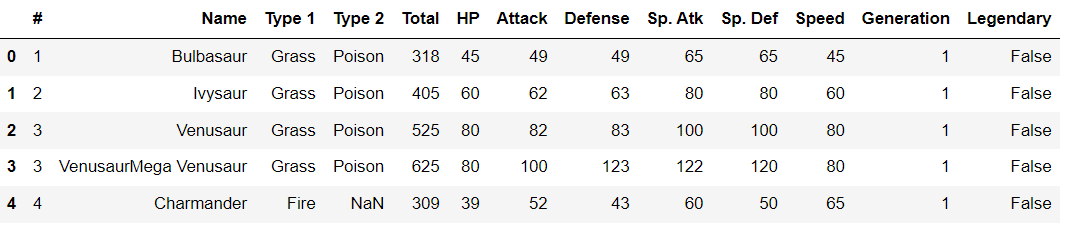
%matplotlib inline

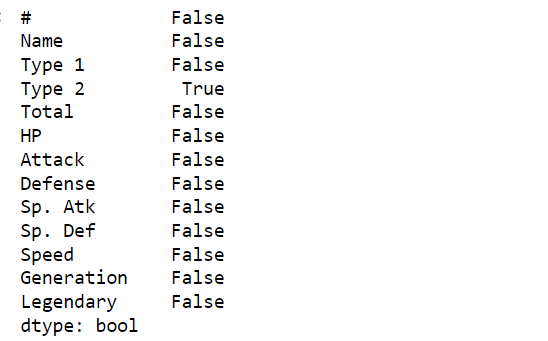
data=pd.read\_csv('Pokemon.csv')

data.head()

data.isna().any()

## **Output:**





**Data Preparation:**

Only a part of the data is selected for clustering and the others are discarded.

**Code:**

types = data['Type 1'].isin(['Grass', 'Fire', 'Water']) # True for pokemon with type1 as 'Grass', 'Fire or 'Water

types

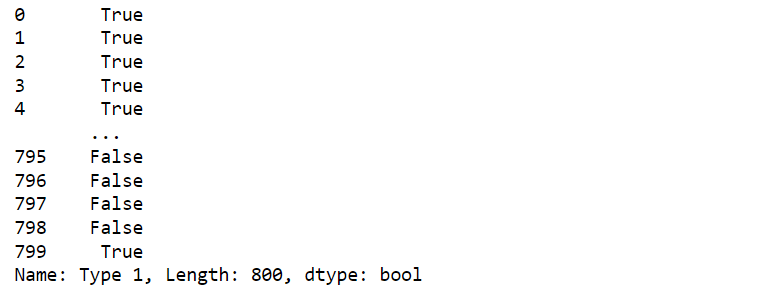
data[types] # Pokemon with type1 as 'Grass', 'Fire or 'Water

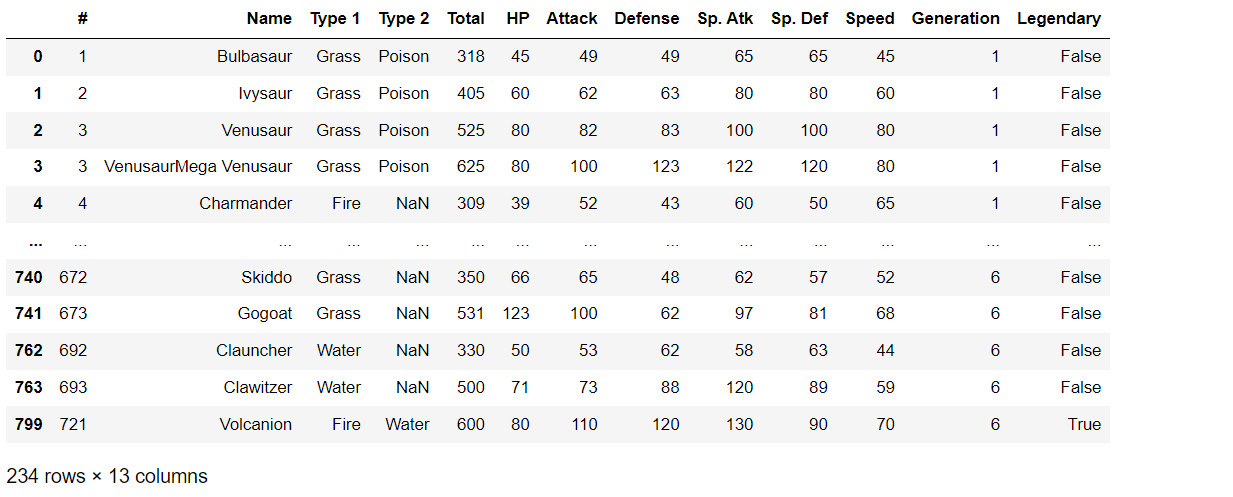
drop\_cols=['Type 1','Type 2','Generation','Legendary','#']

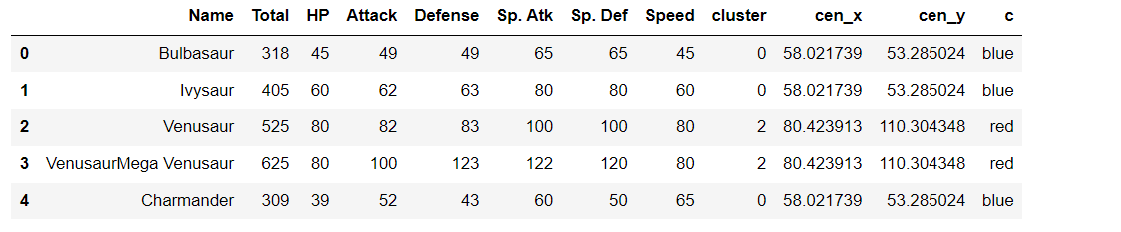
pokemon=data.drop(columns=drop\_cols)

pokemon.head()

# Output:







**Clustering:**

Clustering is a fundamental technique in unsupervised machine learning where data points are grouped together based on their similarities. The primary goal of clustering is to partition a dataset into groups or clusters, such that data points within the same cluster are more similar to each other than to those in other clusters. This allows for identifying patterns, structure, and relationships within the data without prior knowledge of the class labels.

**Types of Clustering Algorithms:**

**Partitioning Methods:** These algorithms partition the data into distinct clusters. Examples include k-means, k-medoids, and Gaussian mixture models.

**Hierarchical Methods**: These algorithms create a hierarchy of clusters, either through agglomerative (bottom-up) or divisive (top-down) approaches.

Density-based Methods: These algorithms find clusters by identifying areas of high density within the data space. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a popular density-based algorithm.

**Model-based Methods:** These algorithms assume that the data is generated by a mixture of underlying probability distributions. Examples include Gaussian mixture models and hierarchical Dirichlet processes.

**Evaluation:** Clustering algorithms need to be evaluated to assess the quality of the clustering results. Common evaluation metrics include silhouette score, Davies-Bouldin index, and the purity of clusters.

**Applications:** Clustering finds applications in various fields such as:

* Customer segmentation in marketing
* Image segmentation in computer vision
* Anomaly detection in cybersecurity
* Document clustering in natural language processing
* Genetics and biological data analysis
* Social network analysis

# Code:

from sklearn.cluster import KMeans

# k means

kmeans = KMeans(n\_clusters=3, random\_state=0)

pokemon['cluster'] = kmeans.fit\_predict(pokemon[['Attack', 'Defense']])

# get centroids

centroids = kmeans.cluster\_centers\_

cen\_x = [i[0] for i in centroids]

cen\_y = [i[1] for i in centroids]

print(cen\_x)

## add to df

pokemon['cen\_x'] = pokemon.cluster.map({0:cen\_x[0], 1:cen\_x[1], 2:cen\_x[2]})

pokemon['cen\_y'] = pokemon.cluster.map({0:cen\_y[0], 1:cen\_y[1], 2:cen\_y[2]})

# define and map colors

colors = ['blue', 'green', 'red']

pokemon['c'] = pokemon.cluster.map({0:colors[0], 1:colors[1], 2:colors[2]})

pokemon.head(20)

plt.figure(figsize=(10,6))

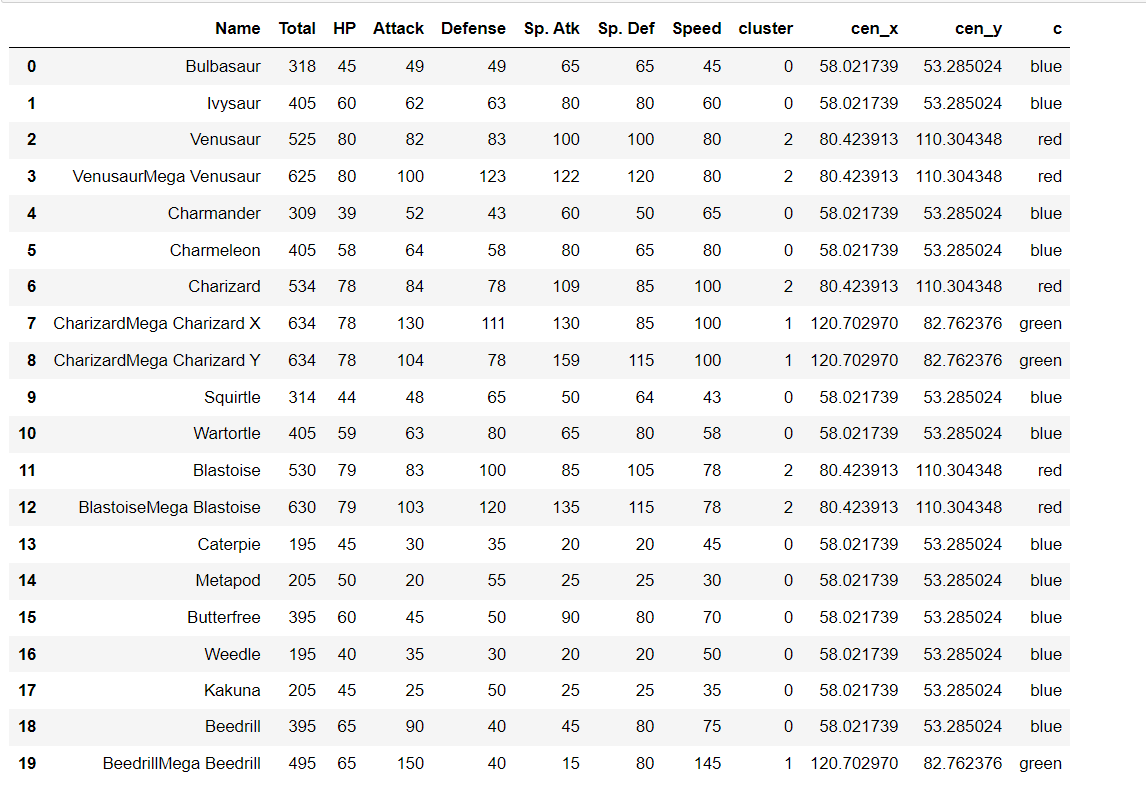
plt.scatter(pokemon.Attack, pokemon.Defense, c=pokemon.c, alpha = 0.6, s=5)

plt.xlabel('Attack')

plt.ylabel('Defense')

plt.show(

**Output:**

****

**Graph Visualization:**

