**Lab on k-means Clustering**

In this lab, we’ll work k-means clustering. First, we’ll load the Data from the Rates.csv dataset. Then the data will be explored. In this phase, the data will also be plotted to see any natural clusters in the data. In the Data Transform phase, some of the features (e.g. categorical features) will be converted to integers for the k-means clustering in Phase 4.

The main purpose of this lab is to work on different functions in the k-means.py discussed in Phase 4.

1. **Load the Data**
2. Import the pandas library as "pd".

import pandas as pd

import numpy as np

1. Read the Rates.csv file into a data frame called *policies*.

policies = pd.read\_csv("Rates.csv")

1. **Explore the Data**
2. Inspect the *policies* data frame using the head function.
3. Import pyplot from matplotlib as "plt".

import matplotlib.pyplot as plt

1. Create a scatterplot matrix of the data set.

pd.plotting.scatter\_matrix(

frame = policies,

alpha = 1,

s = 100,

diagonal = 'none');

1. **Question**: Do you see any natural clusters in these data? How Many?

### Transform the Data

* 1. Create a data frame of features for clustering (omit the categorical State column).

X = policies.iloc[:, policies.columns != "State"]

* 1. Inspect the features using the head function

X.head()

* 1. Convert the categorical Gender variable {Female, Male} to integers {0, 1} as

X.Gender = X.Gender.apply(lambda x: 0 if x == "Female" else 1)

1. Inspect the integer Gender series

X.Gender.head()

1. Set the random number seed

np.random.seed(42)

### 4. Cluster with k-Means

Import the k-means class from sklearn, and implement the following functions in kmeans.py:

from sklearn.cluster import KMeans

1. distance\_euclidean(p1, p2,…) # iter= 100
2. distance\_manhattan(p1, p2,…) # iter= 100

Try different values of *k*, and try both Euclidean and Manhattan distances.

1. initialization\_forgy(data, k)
2. iteration\_one(data, means, distance)
3. hasconverged(old\_means, new\_means, epsilon) # Use epsilon=10−2
4. iteration\_many(data, means, distance, numiter, epsilon)
5. performance\_SSE(data, means, distance)

## Forgy Cluster Initialization:

## The method initializes the k-means algorithm. The Forgy method randomly chooses *k* points from the data set and uses these as the initial means. This ensures that the initial clusters are uniformly spread out across the data.

Sum Squared Error (SSE):

k-means clustering tries to locally minimize the Sum Squared Error, where the error associated with each data point is taken as its Euclidean distance (or Manhattan distances) from the cluster center.

1. Create a palette with k colors for each of the k clusters.
2. Map the colors to each of the clusters.
3. Create a scatterplot matrix colored by cluster.
4. Plot a scatterplot of BMI (x-axis) vs. Age (y-axis) colored by the clusters.