- **APPROACH**

Part 1: Iris Clustering

The approach I used for means algorithm works as follows:

Set number of clusters K = 10.

I initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.

```
def initial_centroids(self, X):
    np.random.RandomState(self.random_state)
    random_indx = np.random.permutation(X.shape[0])
    centroids = X[random_indx[:self.n_cluster]]
    return centroids
```

I iterated until there is no change to the centroids. Then I computed the sum of the squared distance between data points and all centroids.

```
def compute_dist(self, X, centroids):
    dist = np.zeros((X.shape[0], self.n_cluster))
    for k in range(self.n_cluster):
        row_norm = norm(X - centroids[k, :], axis=1)
        dist[:, k] = np.square(row_norm)
    return dist

def compute_sse(self, X, labels, centroids):
    dist = np.zeros(X.shape[0])
    for k in range(self.n_cluster):
        dist[labels == k] = norm(X[labels == k] - centroids[k], axis=1)
    return np.sum(np.square(dist))


def compute_centroids(self, X, labels):
    centroids = np.zeros((self.n_cluster, X.shape[1]))
    for k in range(self.n_cluster):
        centroids[k, :] = np.mean(X[labels == k, :], axis=0)
    return centroids
```

After this I assigned each data point to the closest cluster. And computed the centroids for the clusters by taking the average of all data points that belong to each cluster. Max iteration was set as 100.

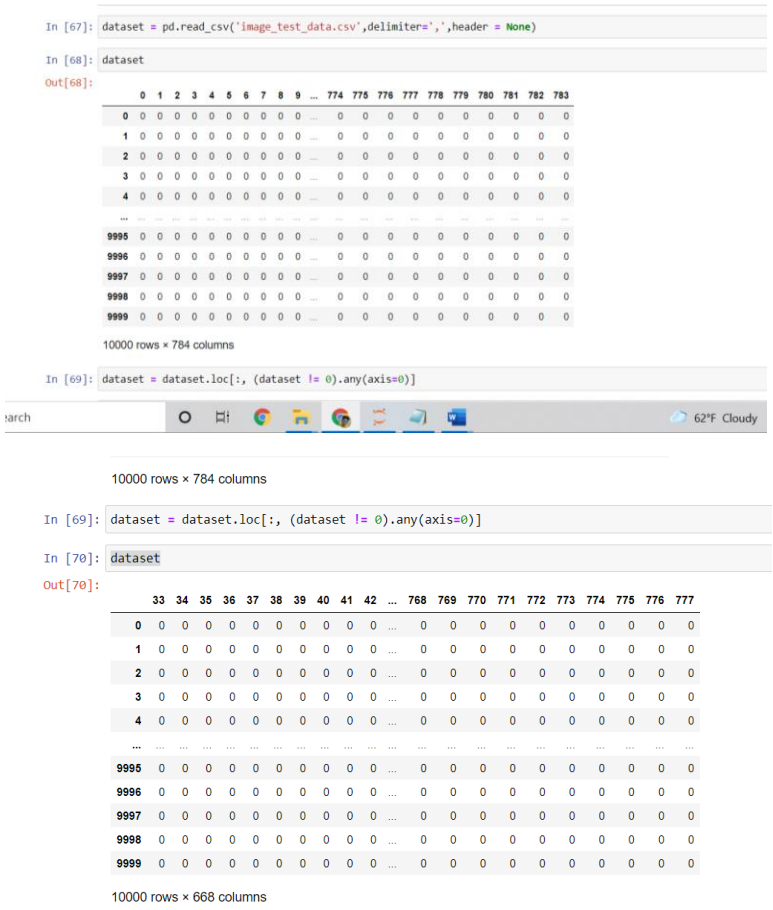
```
def find_nearest_cluster(self, dist):
    return np.argmin(dist, axis=1)


def fit(self, X):
    self.centroids = self.initial_centroids(X)
    for i in range(self.max_iter):
        old_centroids = self.centroids
        dist = self.compute_dist(X, old_centroids)
        self.labels = self.find_nearest_cluster(dist)
        self.centroids = self.compute_centroids(X, self.labels)
        if np.all(old_centroids == self.centroids):
            break
    self.error = self.compute_sse(X, self.labels, self.centroids)

def predict(self, X):
    dist = self.compute_dist(X, old_centroids)
    return self.find_nearest_cluster(dist)
```

Part 2: Image Clustering

For part 2, I used the same approach as first part, but did some pre-processing of data.

For pre-processing in removed the rows and columns which were completely zero. Which helped in increasing accuracy by 5%.

Max iteration was set as 200.



```
In [67]: dataset = pd.read_csv('image_test_data.csv',delimiter=',',header = None)
In [68]: dataset
```

Out[68]:

	0	1	2	3	4	5	6	7	8	9	...	774	775	776	777	778	779	780	781	782	783
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9996	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9997	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9998	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9999	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

10000 rows × 784 columns

```
In [69]: dataset = dataset.loc[:, (dataset != 0).any(axis=0)]
```

10000 rows × 784 columns

```
In [69]: dataset = dataset.loc[:, (dataset != 0).any(axis=0)]
In [70]: dataset
```

Out[70]:

	33	34	35	36	37	38	39	40	41	42	...	768	769	770	771	772	773	774	775	776	777
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9995	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9996	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9997	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9998	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9999	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

10000 rows × 668 columns

Silhouette Score

Silhouette score for 10 clusters was 0.059