PEMBUATAN MACHINE LEARNING

Adrian Syah Abidin

20/463588/TK/51580

In [56]:
```
# menambahkan library yang dibutuhkan
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
```

In [57]:
```
# mengimport data set
fileLocation = 'D:\KULIAH SEM 3\TUGAS AI\Data_ML.xlsx'
df = pd.read_excel(fileLocation, index_col = 0, header=2)
df.head()
```

Out[57]:

emiten	mcap	tvalue
DSSA	18.32	0.344
BYAN	93.17	9.750
MEGA	58.26	0.495
ITMG	28.05	153.110
MLBI	16.86	0.681

In [58]:
```
# menyederhakan data set
z=df.iloc[0:22,0:4]
z
```

Out[58]:

emiten	mcap	tvalue
DSSA	18.320	0.3440
BYAN	93.170	9.7500
MEGA	58.260	0.4950
ITMG	28.050	153.1100
MLBI	16.860	0.6810
ICBP	102.920	90.7100
UNVR	158.320	136.6000
GGRM	62.770	15.0700
EDGE	10.470	0.1040
GEMS	25.410	0.5250
ARTA	0.725	1.7100
TINS	11.950	190.1300
OMRE	1.600	0.0009
SUPR	16.950	0.6020
PNSE	0.370	0.0420
PUDP	0.154	1.2900
INDF	57.070	60.2900
SCCO	2.140	0.0060
ACES	23.580	67.9500

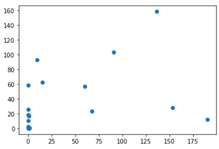
In [59]:
```
z.describe()
```

Out[59]:

	mcap	tvalue
count	19.000000	19.000000
mean	36.267842	38.389995
std	42.878273	61.036524
min	0.154000	0.000900
25%	6.305000	0.419500
50%	18.320000	1.290000
75%	57.665000	64.120000
max	158.320000	190.130000

In [60]:
```
# menampilkan persebaran data
plt.scatter(z['tvalue'],z['mcap'])
```

Out[60]:
```
<matplotlib.collections.PathCollection at 0x8708130>
```



In [68]:
```
# class KMeansClustering() :
#     def __init_(self, X,num_clusters):
#         self.K = num_clusters
#         self,plot_figure = True
#         self,max_iterations = 100
#         self,num_example, self.num_featurs = X.shape


#     def initialize_random_centroids(self, X):
#         centroids = np.zeros((self.K, self.num_features))

#         for k in range(self.K):
#             centroid = X[np.random.choice(range(self.num_example))]
#             centroids[k] = centroid

#         return centroid

#     def create_clusters(self, X, centroids):
#         cluster = [[] for _ in range(self.K)]

#         for point_idx, point in enumerate(X):
#             closest_centroid = np.argmin(np.sqrt(np.sum((point -centroids)**2, axis=1)))
#             clusters[closest_centroid].append(point_idx)

#         return clusters

#     def calculate_new_centroids(self, clusters, X):
#         centroids = np.zeros((self.K, self.num_features))

#         for idx, clusters in enumerate(clusters):
#             new_centroid = np.mean(X[cluster], axis=0)
#             centroids[idx] = new_centroid

#         return centrois


#     def predict_cluster(self, clusters, X):
#         y_pred = np.zeros(self.num_examples)

#         for cluster_idx, cluster in enumerate(clusters):
#             for sample_idx in cluster:
#                 y_pred[sample_idx] = cluster_idx

#         return y_pred

#     def plot_fig(self, X, y):
#         plt.scatter(X[:, 0], X[:, 1],c=y, s=40, cmap=plt.cm.Spectral)
#         plt.show

#     def fit(self, X):
#         centroids = self.initialize_random_centroids(X)

#         for it in range(self.max_iterations):
#             clusters = self.create_clusters(X, centroids)

#             previous_centroids = centroids
#             centroids = self.calculate_new_centroids(clusters, X)

#             diff = centroids - previous_centroids

#             if not diff.any():
#                 print("kriteria terminasi terpenuhi, k-means telah konvergen")
#                 break

#         y_pred = self.predict_cluster(cluster, X)

#         if self.plot_figure:
#             self.plot_fig(X, y_pred)

#         return y_pred


#     num_clusters = 2
#     X, _ = z(n_samples = z['tvalue'], n_features=2, centers=num_clusters)

#     Kmeans = KMeansClustering(X, num_clusters)
#     y_pred = Kmeans.fit(X)
#     #(GAGAL :"))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-68-ed23d44b8492> in <module>
---> 1 class KMeansClustering() :
      2     def __init_(self, X,num_clusters):
      3         self.K = num_clusters
      4         self,plot_figure = True
      5         self,max_iterations = 100

<ipython-input-68-ed23d44b8492> in KMeansClustering()
     72
     73     num_clusters = 2
---> 74     X, _ = z(n_samples = z['tvalue'], n_features=2, centers=num_clusters)
     75
     76     Kmeans = KMeansClustering(X, num_clusters)

TypeError: 'DataFrame' object is not callable
```

In [118…
```
tvalue1 = [0.344,9.75,0.495,153.11,0.681,90.71,136.6,15.07,0.104,0.525,1.71,190.13,0.0009,0.602,0.0420,1.29,60.29,0.006,67.95]
mcap1   = [18.320,93.170,58.260,28.050,16.860,102.920,158.320,62.770,10.470,25.410,0.725,11.950,1.6,16.950,0.370,0.154,57.070,2.140,23.580]
P = tvalue1[0:18] + mcap1[0:18]

# nyerah :)

    # def calculate_data1():
```

In [112…
```
z.values
```

Out[112…
```
array([[1.8320e+01, 3.4400e-01],
       [9.3170e+01, 9.7500e+00],
       [5.8260e+01, 4.9500e-01],
       [2.8050e+01, 1.5311e+02],
       [1.6860e+01, 6.8100e-01],
       [1.0292e+02, 9.0710e+01],
       [1.5832e+02, 1.3660e+02],
       [6.2770e+01, 1.5070e+01],
       [1.0470e+01, 1.0400e-01],
       [2.5410e+01, 5.2500e-01],
       [7.2500e-01, 1.7100e+00],
       [1.1950e+01, 1.9013e+02],
       [1.6000e+00, 9.0000e-04],
       [1.6950e+01, 6.0200e-01],
       [3.7000e-01, 4.2000e-02],
       [1.5400e-01, 1.2900e+00],
       [5.7070e+01, 6.0290e+01],
       [2.1400e+00, 6.0000e-03],
       [2.3580e+01, 6.7950e+01]])
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: