**K-means Clustering**

df = pd.read\_csv('./Final\_Refined\_Encoded\_Normalysed.csv')

train, test = train\_test\_split(df, test\_size = 0.3, random\_state = 10)

# X = np.array(train.drop(['Price'], 1).astype(float))

X = np.array(train[['SIM', 'CPU', 'GPU', 'memory\_card', 'weight\_g', 'screen\_to\_body\_ratio', \

'primary\_camera', 'internal\_memory', 'Thickness',\

'display\_size', 'OS', 'radio', 'RAM', 'EDGE'\

]])

y = np.array(train['Price'])

#train.info()

kmeans = KMeans(algorithm='auto', copy\_x=True, init='k-means++', max\_iter=600,

n\_clusters=6, n\_init=10, n\_jobs=1, precompute\_distances='auto',

random\_state=None, tol=0.0001, verbose=0)

scaler = MinMaxScaler()

X\_scaled = scaler.fit\_transform(X)

kmeans.fit(X\_scaled)

correct = 0

for i in range(len(X)):

predict\_me = np.array(X[i].astype(float))

predict\_me = predict\_me.reshape(-1, len(predict\_me))

prediction = kmeans.predict(predict\_me)

if prediction[0] == y[i]:

correct += 1

print("Accuracy Rate:", correct/len(X))

# Reference: <https://www.datacamp.com/community/tutorials/k-means-clustering-python>

**Output:**

➜ python3 k-means\_clustering.py

Accuracy Rate: 0.39406053683609366

➜ python3 k-means\_clustering.py

Accuracy Rate: 0.385208452312964

➜ python3 k-means\_clustering.py

Accuracy Rate: 0.008280982295830954