Assignment 1
Python For Data Science

MCQs

1. **What will be the Output after the following statements are executed?**

```
X = 300
Y = 17
X%=Y
print(X)
```

a) 7.6
b) 11
c) 25
d) 300

2. **The function used to perform k-Nearest Neighbour classification is-**
a) sklearn.KNN
b) sklearn.KNearestClassifier
c) sklearn.neighbors.KNeighborsClassifier( )
d) sklearn.neighbors.KNeighborsRegressor()

Descriptive

1. **Describe the working principle of the K-means clustering algorithm. Explain the steps involved in forming clusters and how the centroids are updated during the process?**

=>

Working principle:

K-Means is an unsupervised algorithm that partitions n data points $\{x_i\}_{i=1}^{n}$ in $\mathbb{R}^d$ into K clusters by minimizing the **within-cluster sum of squares (WCSS)**:

$$J = \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - \mu_k\|^2,$$

where C_k is the set of points in cluster k and μ_k is its centroid.

Algorithm steps:

1. **Choose K and initialize centroids $\{\mu_k\}_{k=1}^{K}$** (randomly or with **k-means++** seeding to spread them out).
2. **Assignment step (E-step analogue):** For each point, assign a label

$$c_i = \arg\min_{k \in \{1,\ldots,K\}} \|x_i - \mu_k\|^2$$

(typically Euclidean distance).

3. **Update step (M-step analogue):** For each cluster, recompute the centroid as the mean of its assigned points:

$$\mu_k = \frac{1}{|C_k|} \sum_{x_i \in C_k} x_i.$$

Centroid update intuition:
The mean is the point that minimizes the sum of squared distances to points in a cluster; therefore recomputing μ_k as the mean monotonically decreases (or leaves unchanged) the objective **J**.

Convergence & complexity:
K-Means decreases J every iteration and converges in finitely many steps to a local minimum (not guaranteed global). Time complexity per iteration is *O(nKd)* with *I* iterations, *O(nKId)*.

Practical notes & limitations:

- **Preprocessing:** Standardize features so axes are comparable.
- **Choosing KKK:** Use elbow (SSE vs. KKK), silhouette score, or domain knowledge.
- **Sensitivity:** Results depend on initialization; k-means++ and multiple restarts help.
- **Assumptions/shape:** Works best for roughly spherical, similarly sized clusters; sensitive to outliers and non-convex shapes.
- **Edge cases:** Empty clusters can be re-initialized (e.g., to the farthest point).
- **Distance choice:** Euclidean is standard; other metrics change the objective interpretation.

2. **Consider the following Python code snippet executed in the Spyder IDE:**

```
a=15
b=4
c=a/b
d=a//b
e=a%b
f=a**b
print(c,d,e,f)
```

a) Write the output of the program.

b) Explain why c and d are different even though both perform division.

Answer: a) Output: 3.75 3 3 50625

b) **Explanation:**

- a / b = 15 / 4 = 3.75 -> Division (/) in Python always returns float, even with integers.
- a // b = 15 // 4 = 3 -> Floor Division truncates the decimal part, returning only the integer quotient.
- a % b = 15 % 4 = 3 -> Modulus gives the remainder.
- a ** b = 15 ** 4 = 50625 -> Exponentiation operator.

Thus, / and // behave differently in Python, making explicit control over division possible.