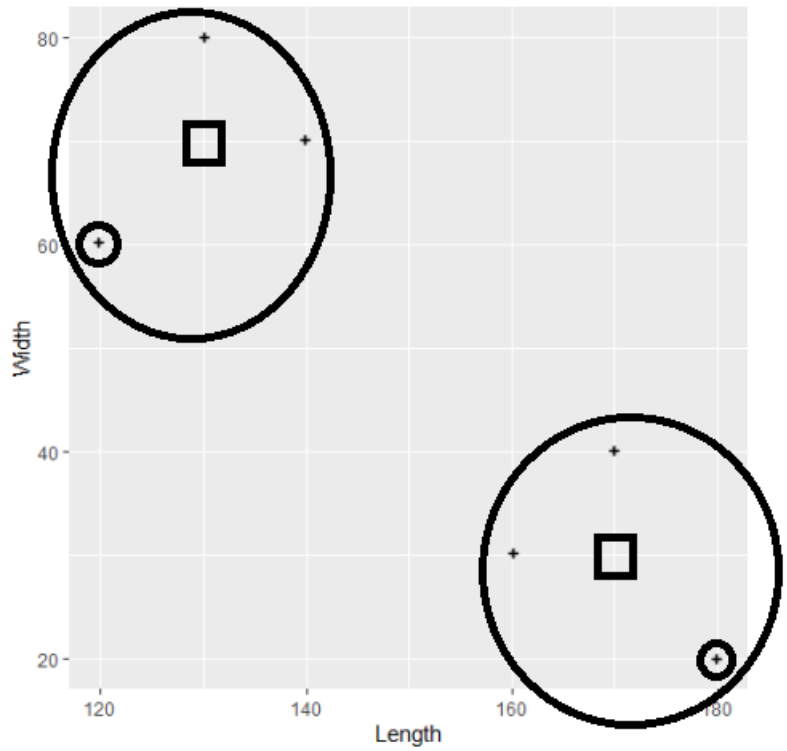
```
# 1. What would the 1-Nearest Neighbor approach predict is the class when the
feature set is {Soymilk, Apple, Eggs} and the following data is used as the
training data (same table as the classwork)? Use the Jaccard similarity to
calculate neighbors. Show your calculations. (Note: do NOT write any code for
this problem. The answers are to be computed by hand.)
'''
Jaccard Similarity=  A B  /  A B

Row A: {Apple, Banana, Soymilk, Yogurt}
Intersection: {Apple, Soymilk} (2 elements)
Union: {Apple, Banana, Soymilk, Yogurt, Eggs} (5 elements)
Jaccard Similarity: 2 / 5 = 0.4

Row B: {Apple, Peanuts, Yogurt}
Intersection: {Apple} (1 element)
Union: {Apple, Peanuts, Yogurt, Soymilk, Eggs} (5 elements)
Jaccard Similarity: 1 / 5 = 0.2

Row C: {Tomatoes, Potatoes, Yogurt}
Intersection: {} (0 elements)
Union: {Tomatoes, Potatoes, Yogurt, Soymilk, Apple, Eggs} (6 elements)
Jaccard Similarity: 0 / 6 = 0

Row D: {Apple, Tomatoes, Potatoes}
Intersection: {Apple} (1 element)
Union: {Apple, Tomatoes, Potatoes, Soymilk, Eggs} (5 elements)
Jaccard Similarity: 1 / 5 = 0.2

Nearest Neighbor = Row A because it has the highest Jaccard Similarity
Class of Row A is Vegetarian

ANSWER: Vegetarian
'''
```

```
'\nJaccard\xa0Similarity=  A B  /  A B \n\nRow A: {Apple, Banana, Soymilk,
Yogurt}\nIntersection: {Apple, Soymilk} (2 elements)\nUnion: {Apple, Banana,
Soymilk, Yogurt, Eggs} (5 elements)\nJaccard Similarity: 2 / 5 = 0.4\n\n\u200bRow
B: {Apple, Peanuts, Yogurt}\nIntersection: {Apple} (1 element)\nUnion: {Apple,
Peanuts, Yogurt, Soymilk, Eggs} (5 elements)\nJaccard Similarity: 1 / 5 =
0.2\n\u200b\nRow C: {Tomatoes, Potatoes, Yogurt}\nIntersection: {} (0
elements)\nUnion: {Tomatoes, Potatoes, Yogurt, Soymilk, Apple, Eggs} (6
elements)\nJaccard Similarity: 0 / 6 = 0\n\nRow D: {Apple, Tomatoes,
Potatoes}\nIntersection: {Apple} (1 element)\nUnion: {Apple, Tomatoes, Potatoes,
Soymilk, Eggs} (5 elements)\nJaccard Similarity: 1 / 5 = 0.2\n\nNearest Neighbor
= Row A because it has the highest Jaccard Similarity\nClass of Row A is
Vegetarian\n\nANSWER: Vegetarian\n'
```

```
# 2. Consider the following dataset. (Note: do NOT write any code for this
problem. The answers are to be computed by hand and marked on the graph. You can
visually guess some of the answers.)
# e. Will the k-means algorithm terminate after this first iteration or will it
continue? Answer in 1-2 sentences.
# ANSWER: The k-means algorithm will continue. Since the cluster centers have
changed from the initial positions, the algorithm will perform additional
iterations to verify if further adjustments to the centers are needed, eventually
converging when the centers no longer change.
# f. If a new point (Length=140, Width=60) is given, to which cluster will it
belong?
'''
To determine the closest cluster for the new point (140, 60):
Distance to New Center 1 (130, 70):
d = sqrt((140-130)^2 + (60-70)^2)   14.14
Distance to New Center 2 (170, 30):
d = sqrt((140-170)^2 + (60-30)^2)   42.43

Since the new point (140, 60) is closer to New Center 1 (130, 70), it would
belong to Cluster 1.
'''
# a-d.
from google.colab import files
from IPython.display import Image
Image(filename="2.png")
```



```
# 3. Consider the file breast-cancer-wisconsin.csv (in the K-means clustering;
Evaluation metrics module on Canvas) which contains "Features computed from a
digitized image of a fine needle aspirate (FNA) of a breast mass." The goal is to
cluster the data based on the features to distinguish Benign and Malignant cases.
# a. Read the data from the file into an object called "mydata". Column 1
("Code") is the anonymized subject code and will not be used here. Columns 2-10
are the 9 features. Column 11 is the diagnosis: [B]enign or [M]alignant.
!pip install pandas
import pandas as pd
mydata = pd.read_csv("breast-cancer-wisconsin.csv")
# i. How many total cases are there in the data?: ___
total_cases = mydata.shape[0]
print(f"Total cases: {total_cases}")
# ii. How many [B]enign cases are there in the data?: ___
benign_cases = mydata[mydata['Class'] == 'B'].shape[0]
print(f"Benign cases: {benign_cases}")
# iii. How many [M]alignant cases are there in the data?: ___
malignant_cases = mydata[mydata['Class'] == 'M'].shape[0]
print(f"Malignant cases: {malignant_cases}")
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages
(2.2.2)
Requirement already satisfied: numpy>=1.22.4 in
/usr/local/lib/python3.10/dist-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas)
(1.16.0)
Total cases: 683
Benign cases: 444
Malignant cases: 239
```

```
# b. Run k-means clustering using all the rows and only the following features:
ClumpThickness, CellSize, and Nuclei. Use nstart=10.
# i. What should be the value of k? k = ___
# ANSWER: In this context, since we are trying to cluster benign and malignant
cases, it makes sense to set   = 2 because we have two classes (Benign and
Malignant). We will proceed with   = 2.
# ii. Give Python code:
from sklearn.cluster import KMeans

features = mydata[['ClumpThickness', 'CellSize', 'Nuclei']]
```

```
kmeans = KMeans(n_clusters=2, n_init=10, random_state=42)
kmeans.fit(features)

print("Cluster Centers:")
print(kmeans.cluster_centers_)
print("\nCluster Labels:")
print(kmeans.labels_)
```

```
Cluster Centers:
[[7.27876106 6.80973451 8.07964602]
 [3.03938731 1.34135667 1.30196937]]

Cluster Labels:
[1 0 1 0 1 0 0 1 1 1 1 1 1 1 0 1 1 1 0 1 0 0 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1
 0 0 0 0 1 0 1 0 1 1 0 0 1 0 0 0 0 0 1 0 0 1 1 0 1 1 0 1 0 0 1 1 0 1 0 0 1
 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 0 1 0 0 1 0 1 0 0
 0 1 1 1 0 1 1 1 1 0 0 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 0 1 1 0 1 0
 0 1 1 0 1 1 0 0 1 1 1 1 0 0 1 1 1 1 1 0 0 0 1 0 1 0 1 1 1 0 0 1 0 0 0 1 0
 0 1 1 1 1 0 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 1 0 0 0 1 1 0 1 1 0 1 0 0 1 0 0
 1 0 0 0 1 0 1 0 0 0 0 1 1 1 1 1 1 0 0 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0
 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 0 0 0 0 1 1 0 1 1 0 0 1 1 0 0 1 0 1 0 0 1 1
 0 1 1 1 1 1 1 0 0 1 1 0 1 0 1 1 0 1 0 0 0 1 1 0 0 1 0 1 1 0 0 1 1 1 0 1 1
 1 1 0 1 1 1 0 1 1 1 0 0 0 0 0 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1
 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 1 0
 1 1 1 0 1 0 1 1 1 1 1 1 0 0 0 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 0 0 1
 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 0 1 1
 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0
 0 0 1 1 0 1 1 1 1 1 1 0 0 1 1 1 0 1 0 1 0 0 0 1 0 1 1 1 1 1 1 1 1 1 0 0 1
 1 0 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 0
 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 0]
```

```
# c. Evaluation: Compare the resulting clusters with the known diagnosis
# i & ii. What is the contingency table of your clustering? (Hint: use the
scikit-learn confusion_matrix() function. You can arbitrarily assign cluster 1/2
to Benign/Malignant)
from sklearn.metrics import confusion_matrix

true_labels = mydata['Class'].apply(lambda x: 0 if x == 'B' else 1)

kmeans = KMeans(n_clusters=2, n_init=10, random_state=42)
kmeans.fit(features)

predicted_labels = kmeans.labels_
```

```
if sum(predicted_labels == true_labels) < sum(predicted_labels != true_labels):
    predicted_labels = 1 - predicted_labels


conf_matrix = confusion_matrix(true_labels, predicted_labels)


print("Contingency Table (Confusion Matrix):")
print(conf_matrix)
```

```
Contingency Table (Confusion Matrix):
[[437   7]
 [ 20 219]]
```

```
# 4. Using the contingency table that you obtained from the previous problem
(3.c), calculate the following metrics (consider Malignant as the Positive
class):
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,
recall_score, f1_score

TN = 437  # True Negatives
FP = 7    # False Positives
FN = 20   # False Negatives
TP = 219  # True Positives

# Calculate metrics for the k-means classifier
accuracy_kmeans = (TP + TN) / (TP + TN + FP + FN)
error_rate_kmeans = 1 - accuracy_kmeans
precision_kmeans = TP / (TP + FP)
recall_kmeans = TP / (TP + FN)
f_score_kmeans = 2 * (precision_kmeans * recall_kmeans) / (precision_kmeans +
recall_kmeans)

print("k-means Classifier Metrics:")
print("Accuracy:", round(accuracy_kmeans, 4))
print("Error Rate:", round(error_rate_kmeans, 4))
print("Precision:", round(precision_kmeans, 4))
print("Recall:", round(recall_kmeans, 4))
print("F-score:", round(f_score_kmeans, 4))

# Calculate metrics for the "silly" classifier (all predictions as Malignant)
# Total counts
total_cases = TN + FP + FN + TP
total_malignant = TP + FN  # All Malignant cases
total_benign = TN + FP     # All Benign cases

# Silly classifier (predicting all as Malignant)
```

```
TP_silly = total_malignant
FP_silly = total_benign
TN_silly = 0
FN_silly = 0

# Silly classifier metrics
accuracy_silly = TP_silly / total_cases
error_rate_silly = 1 - accuracy_silly
precision_silly = TP_silly / (TP_silly + FP_silly)
recall_silly = 1  # FN_silly is 0, so recall is maximized at 1
f_score_silly = 2 * (precision_silly * recall_silly) / (precision_silly +
recall_silly)

print("\nSilly Classifier Metrics:")
print("Accuracy:", round(accuracy_silly, 4))
print("Error Rate:", round(error_rate_silly, 4))
print("Precision:", round(precision_silly, 4))
print("Recall:", round(recall_silly, 4))
print("F-score:", round(f_score_silly, 4))
```

```
k-means Classifier Metrics:
Accuracy: 0.9605
Error Rate: 0.0395
Precision: 0.969
Recall: 0.9163
F-score: 0.9419

Silly Classifier Metrics:
Accuracy: 0.3499
Error Rate: 0.6501
Precision: 0.3499
Recall: 1
F-score: 0.5184
```