Implementation: DBSCAN

Alysia Halverson, Daniel Stein, Han Luong, Mikhaela Anderson

Dataset Introduction

- *Palmer Penguins* (Kmeans Exercise)
  - Characteristics for three species of penguins (source) totaling 344 penguins
  - Importance of domain knowledge

```
penguins.head()
```

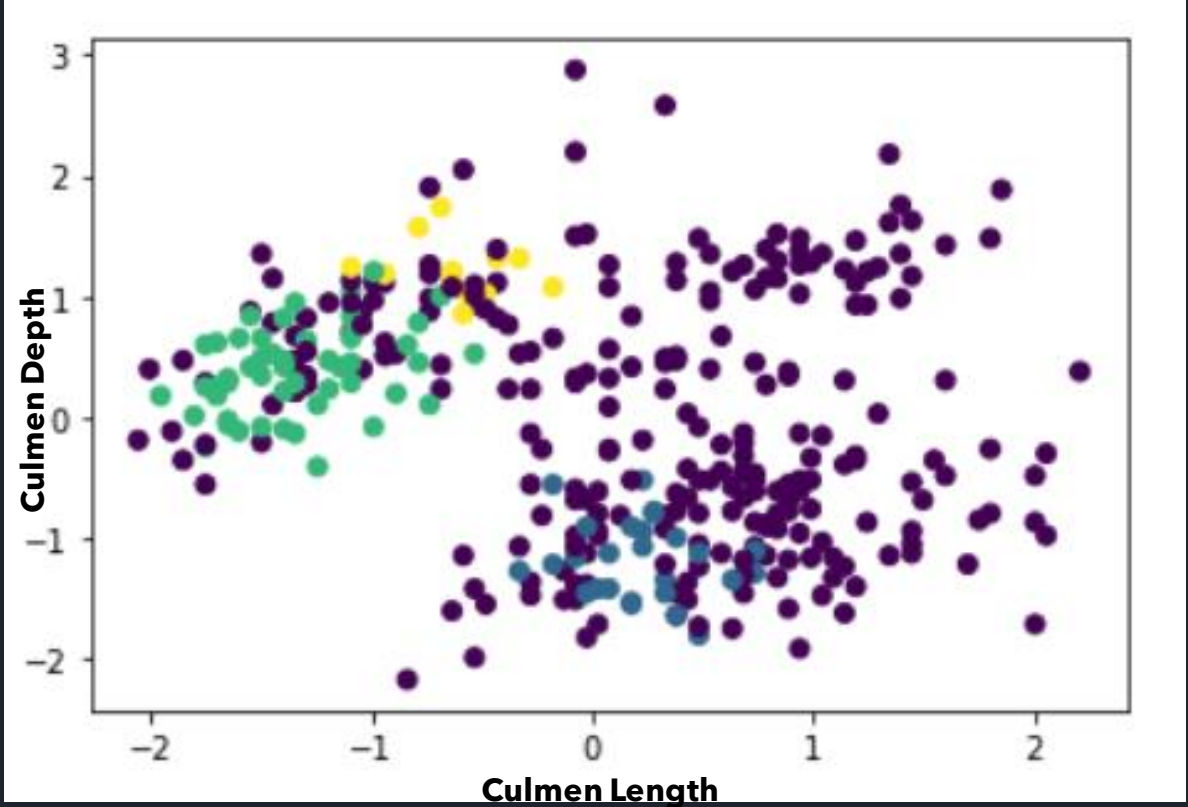
	Region	Stage	Individual ID	Culmen Length (mm)	Culmen Depth (mm)	Flipper Length (mm)	Body Mass (g)	Delta 15 N (o/oo)	Delta 13 C (o/oo)	Comments
0	Anvers	Adult, 1 Egg Stage	N1A1	39.1	18.7	181.0	3750.0	NaN	NaN	Not enough blood for isotopes.
1	Anvers	Adult, 1 Egg Stage	N1A2	39.5	17.4	186.0	3800.0	8.94956	-24.69454	NaN
2	Anvers	Adult, 1 Egg Stage	N2A1	40.3	18.0	195.0	3250.0	8.36821	-25.33302	NaN
3	Anvers	Adult, 1 Egg Stage	N2A2	NaN	NaN	NaN	NaN	NaN	NaN	Adult not sampled.
4	Anvers	Adult, 1 Egg Stage	N3A1	36.7	19.3	193.0	3450.0	8.76651	-25.32426	NaN

```
#renaming the columns so to
penguins.rename(columns={'Indiv

#removing all categorical columns. Th
penguins.drop(['Region','Stage','Indivi

#there are two rows in the dataset that
penguins.dropna(axis=0, thresh=4, inpla

#for the remaining NAs, it seems reaso
impute_mean = SimpleImputer(strategy=
penguins2 = pd.DataFrame(impute_mean

penguins2.columns = penguins.colu
penguins=penguins2
```

Data Cleaning

Drop	Drop categorical columns
Rename	Rename columns
Remove	Remove rows with more than 50% missing values
Replace	Replace remaining NA values with mean values
Standardize	Standardize, scale
Transfer	Transfer column names

DBSCAN Pre-Optimization

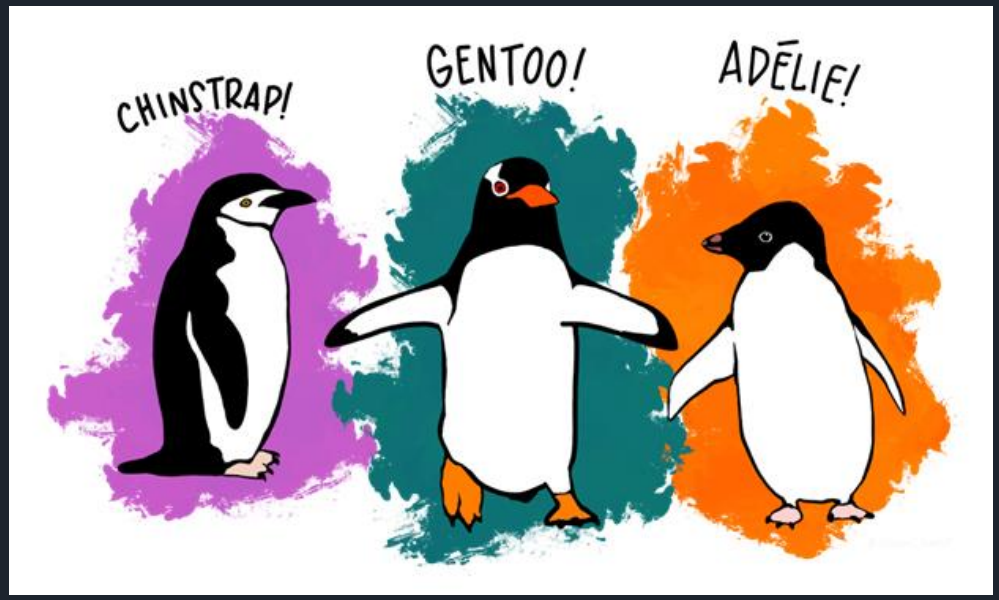



```
db = DBSCAN(eps=.8,min_samples=12).fit(z_penguins)
plt.scatter(z_penguins['CulmenDepth'],z_penguins['CulmenLength'], c=db.labels_)
```

Optimization

Domain Knowledge

Hyperparameter tuning

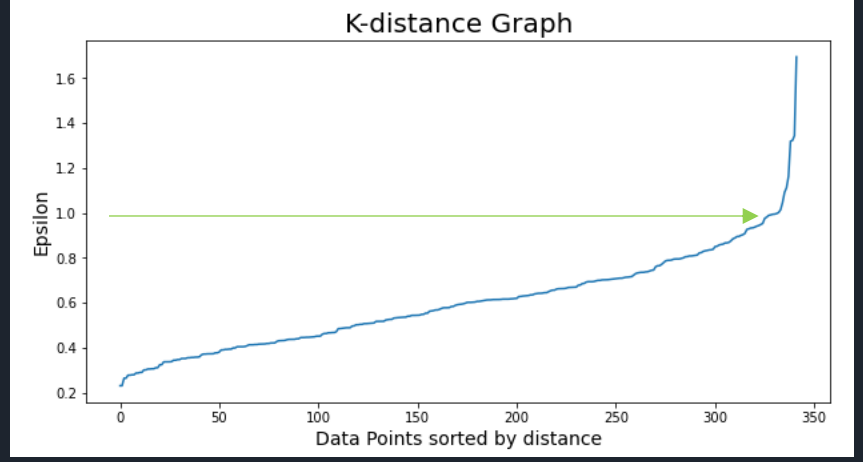


Hyperparameter Tuning

Epsilon

```
neigh = NearestNeighbors(n_neighbors=2)
nbrs = neigh.fit(z_penguins)
distances, indices = nbrs.kneighbors(z_penguins)
```

```
distances = np.sort(distances, axis=0)
distances = distances[:,1]
plt.figure(figsize=(10,5))
plt.plot(distances)
plt.title('K-distance Graph',fontsize=20)
plt.xlabel('Data Points sorted by distance',fontsize=14)
plt.ylabel('Epsilon',fontsize=14)
plt.show()
```

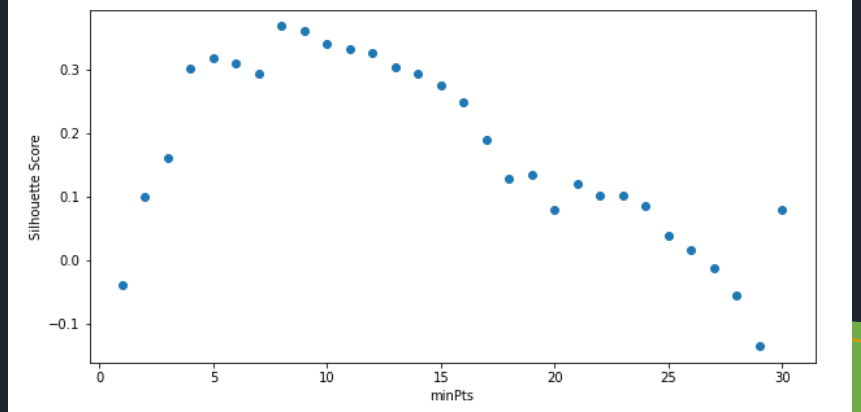


Min_samples

```
silhouettes = []
for mins in range(1,31):
    dbscan = DBSCAN(eps=1,min_samples=mins).fit(z_penguins)
    labels = dbscan.labels_
    silhouettes.append(metrics.silhouette_score(z_penguins, labels))
```

```
mins = range(1,31)
plt.figure(figsize=(10,5))
plt.scatter(x=mins, y=silhouettes)
plt.xlabel('minPts')
plt.ylabel('Silhouette Score')
```

```
Text(0, 0.5, 'Silhouette Score')
```



DBSCAN In Review

- Our most successful DBScan model came about using a K-Distance graph for the Epsilon Hyperparameter, and a combination of silhouette scores and for-loop-outlier-testing for the min_samples value.
- We attempted dimension reducing techniques to find better hyper-parameters but they produced weaker models

Accuracy Scores

K-means (.976)

```
kmeans = KMeans(n_clusters=3, random_state=0)
kclusters = kmeans.fit_predict(X)
print(accuracy_score(kclusters, y))
```

```
0.9757575757575757
```

DBSCAN (.336)

```
dbscan = DBSCAN(eps=1,min_samples=8).fit(X)
df_result = dbscan.labels_
# df_result = pd.DataFrame(dbscan.labels_)
print(accuracy_score(df_result, y))
```

```
0.33636363636363636
```

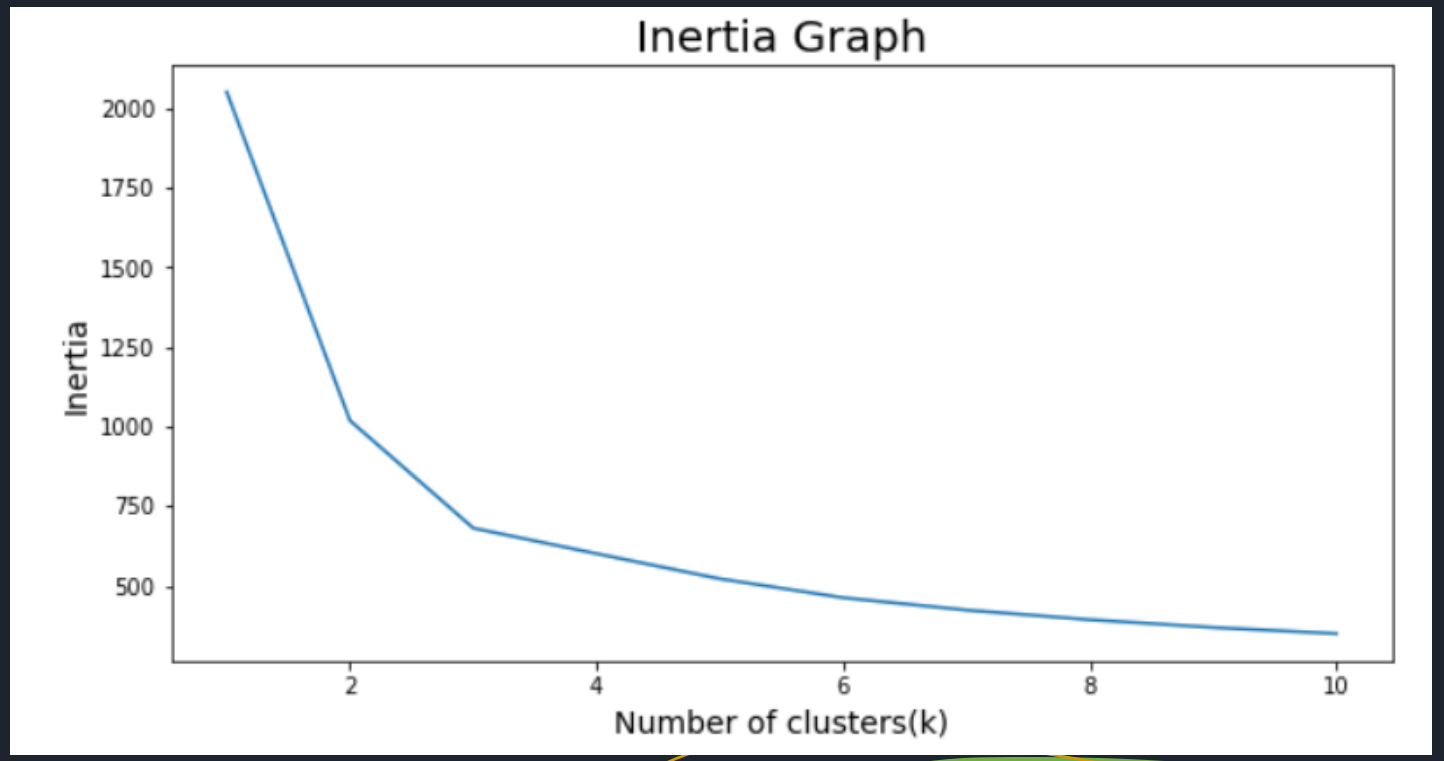
DBSCAN PCA (.327)

```
pca = PCA(n_components=2)
X_reduced = pca.fit_transform(X)
X_reduced = pd.DataFrame(X_reduced)

dbscan=DBSCAN(eps=.37, min_samples=9).fit(X_reduced)
df_result = dbscan.labels_
# df_result = pd.DataFrame(dbscan.labels_)
print(accuracy_score(df_result, y))
```

```
0.32727272727272727
```

K-means Comparison: Clusters Verification

- We used K-means to graph the inertia to verify the ideal number of clusters
- We found that it was able to recognize the separation between the 3 species of penguins as ideal

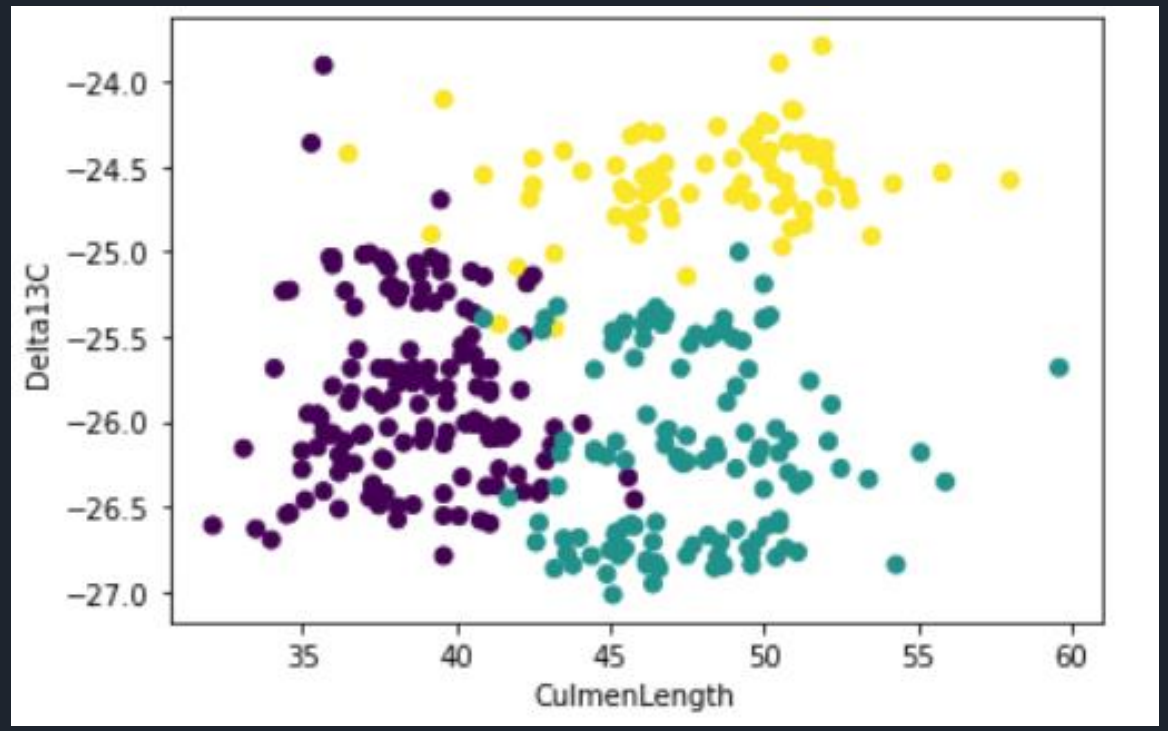


K-means Model

K-means with 3 clusters as indicated on elbow graph

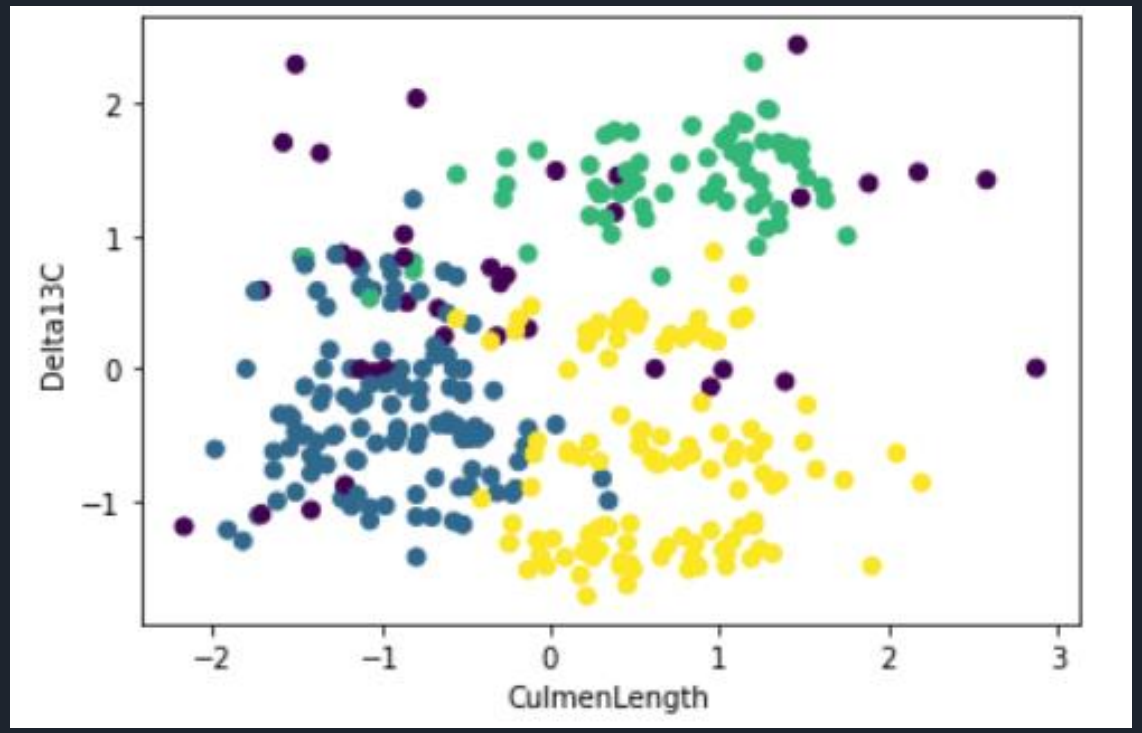
```
kmeans = KMeans(n_clusters=3, max_iter=300, random_state=0 )

y_kmeans = kmeans.fit_predict(z_penguins)

print(pd.DataFrame(y_kmeans).value_counts())

for i in penguins.columns[:4]:
    y="Delta13C"
    penguins.plot(kind="scatter",x=i, y=y,c=y_kmeans);
    plt.scatter(penguins[i], penguins[y], c=y_kmeans);
```

```
0    143
1    123
2     76
dtype: int64
```

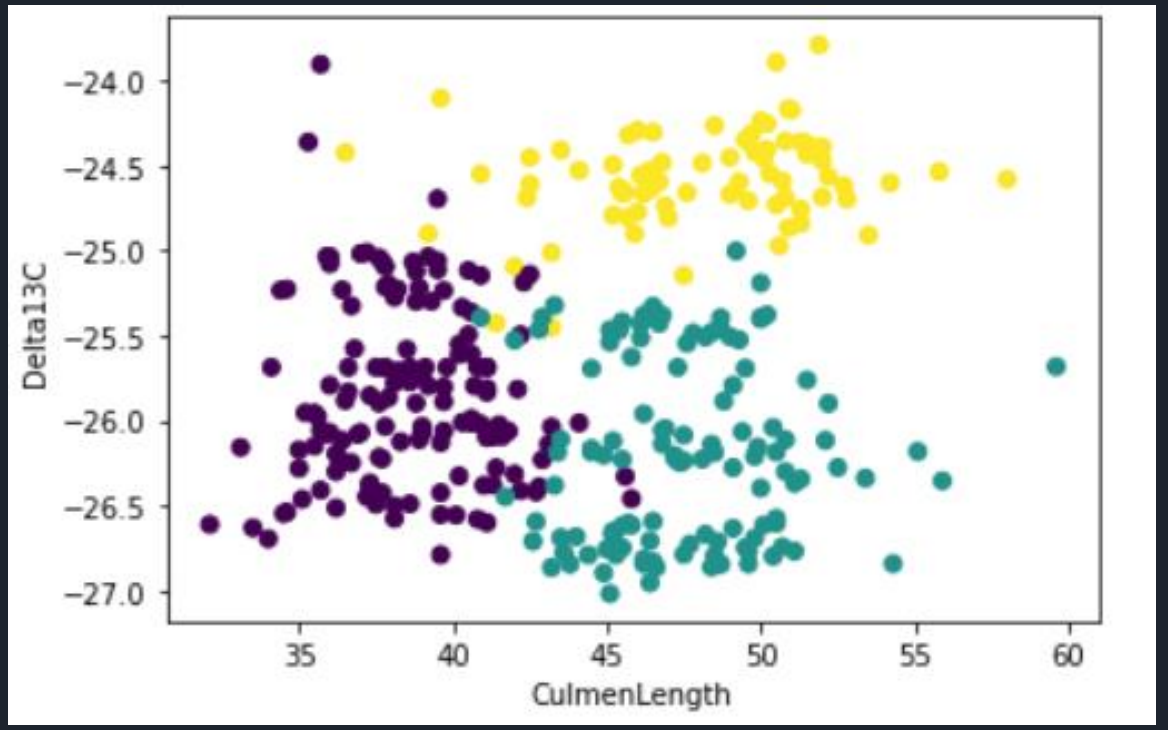


Model Comparison

DBSCAN Model



K-means Model



Final Thoughts

- We agreed that this algorithm is not the best for our dataset. DBSCAN is density based, and our theory is that the penguins' species information is too closely similar for this algorithm to effectively determine the differences. Because K-means is centroid based, it's better at determining data in hypersphere shapes.

Citation

- *Back to Machine Learning Basics – Clustering (2020).* https://rubikscode.net/2020/10/05/back-to-machine-learning-basics-clustering/
- Evaluating Clustering Algorithm – Silhouette Score. https://tushar-joshi-89.medium.com/silhouette-score-a9f7d8d78f29#:~:text=Silhouette%20Score%20is%20a%20metric%20to%20evaluate%20the,of%20how%20well%20our%20clustering%20algorithm%20has%20performed.
- Palmer Penguins - https://www.kaggle.com/datasets/malanep/palmer-penguine/code?resource=download