Implementing DBSCAN

Gavan VanOver

Philip Waymeyer

Widnie Dorilas

Zheding Zhao

Dataset Introduction

- The dataset we used was the 'Animals' CSV from the K-Means exercise
- 344 rows and 10 columns
- The region and stage columns had the same information

```
df.info()
```

[5] ✓ 0.9s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Region              344 non-null    object 
 1   Stage               344 non-null    object 
 2   Individual ID       344 non-null    object 
 3   Culmen Length (mm)  342 non-null    float64
 4   Culmen Depth (mm)   342 non-null    float64
 5   Flipper Length (mm) 342 non-null    float64
 6   Body Mass (g)       342 non-null    float64
 7   Delta 15 N (o/oo)   330 non-null    float64
 8   Delta 13 C (o/oo)   331 non-null    float64
 9   Comments            26 non-null     object 
dtypes: float64(6), object(4)
memory usage: 27.0+ KB
```

Data Processing

- Drop non-numeric and non-categorical columns
- Where rows have missing values, used Simple Imputer to fill in the missing values using the mean
- Use standard scaler to standardize the data to make sure it's on the same level

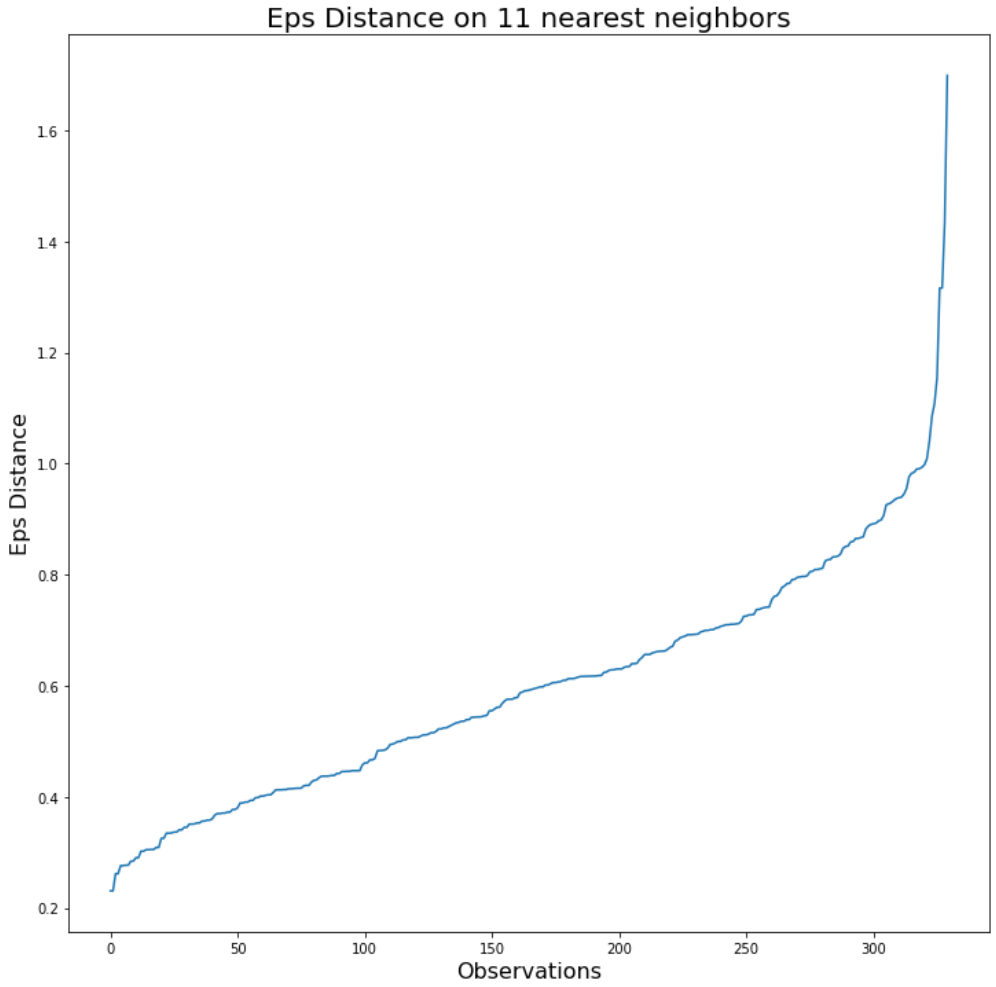
```
# Use SimpleImputer to Fill in missing values by their averages.
impN = SimpleImputer(missing_values=np.nan, strategy='mean')
idf = pd.DataFrame(impN.fit_transform(df))
# Standardize the data to ensure data is on the same scale
ss = StandardScaler()
idf = pd.DataFrame(ss.fit_transform(idf))
idf.columns = df.columns
idf.index = df.index
```

Model Optimization

- Number of Min Samples: 12
- Eps : 1

```
# Optimize epsilon by k nearest neighbors, k is determined by min_sample - 1
k = idf.shape[1] * 2 -1
nbrs = NearestNeighbors(n_neighbors=k).fit(idf)
distances, indices = nbrs.kneighbors(idf)
distances = np.sort(distances, axis=0)
distances = distances[:,1]
plt.figure(figsize=(12,12))
plt.plot(distances)
plt.xlabel('Observations', fontsize=16)
plt.ylabel('Eps Distance',fontsize = 16)
plt.title(f'Eps Distance on {k} nearest neighbors', fontsize=20)
```

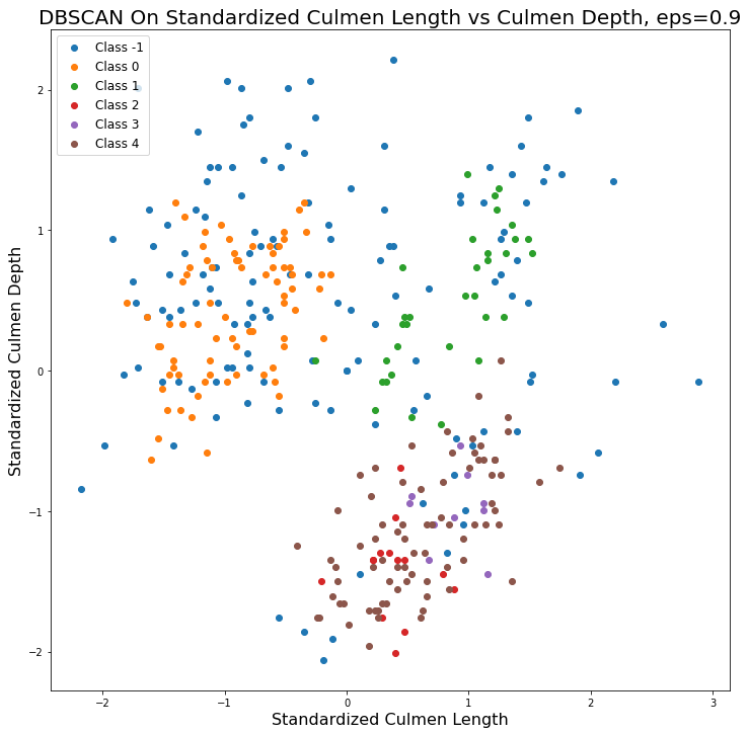
Model Optimization Cont'd

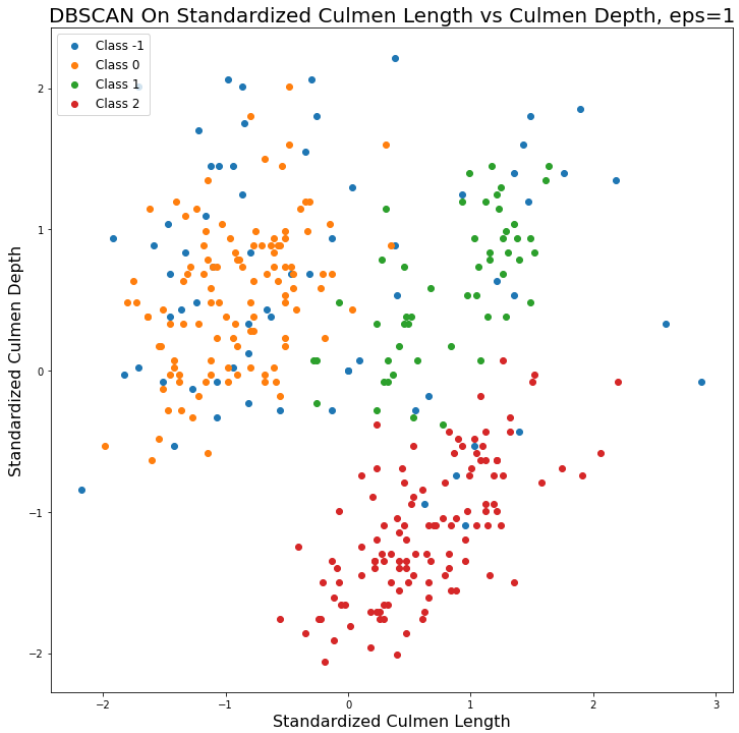


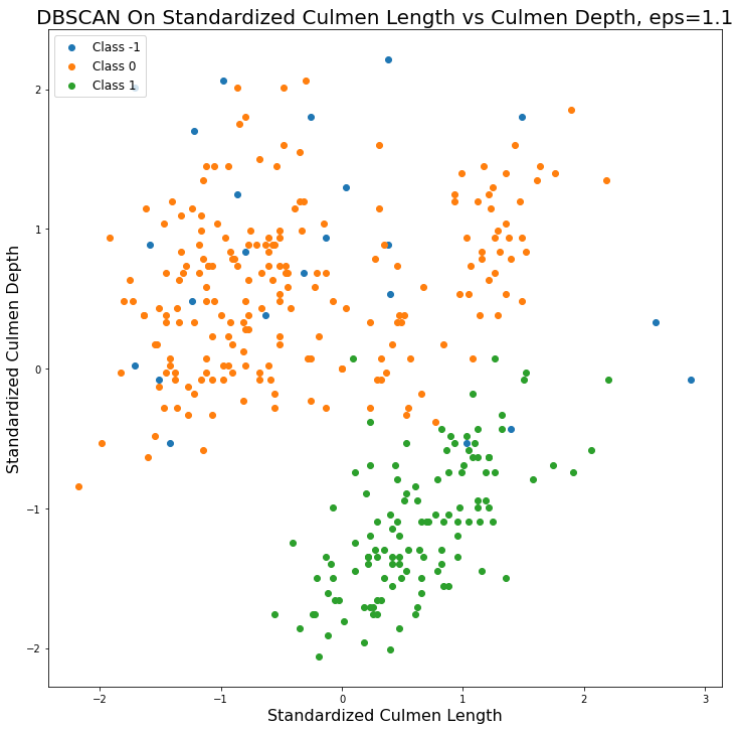
Potential Enhancement

- A better way to impute missing data than the average

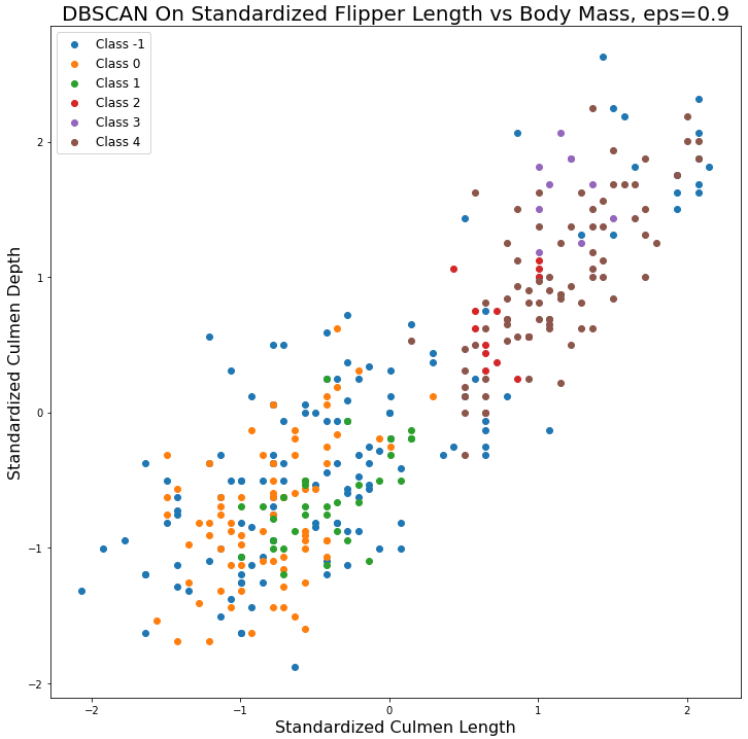
- More careful Analysis on Curvature other than human eyes for optimal eps

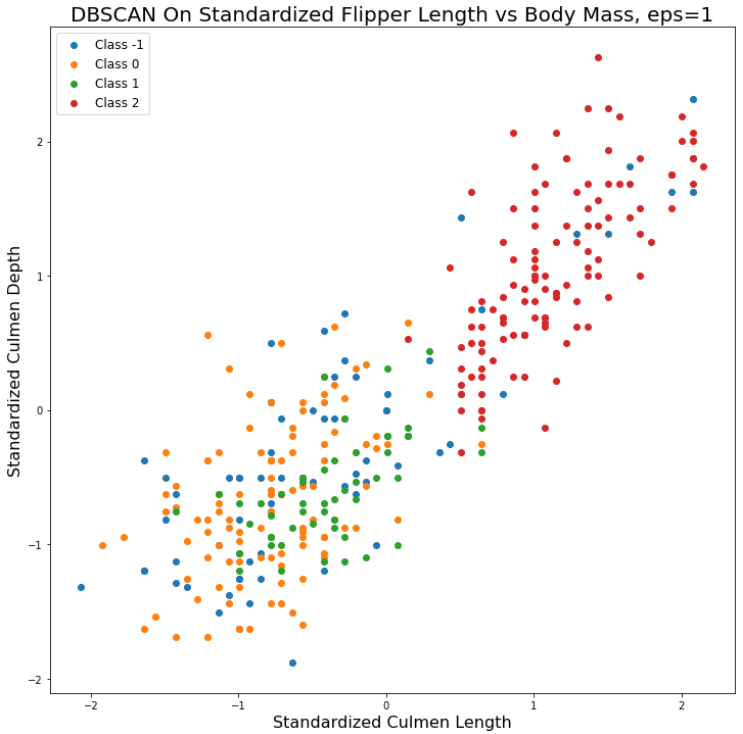


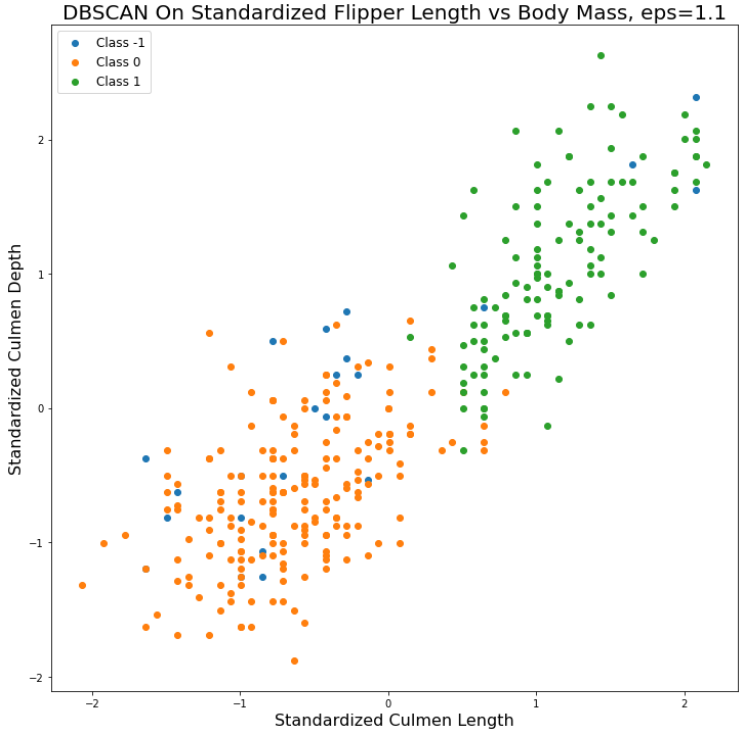




Performance Analysis



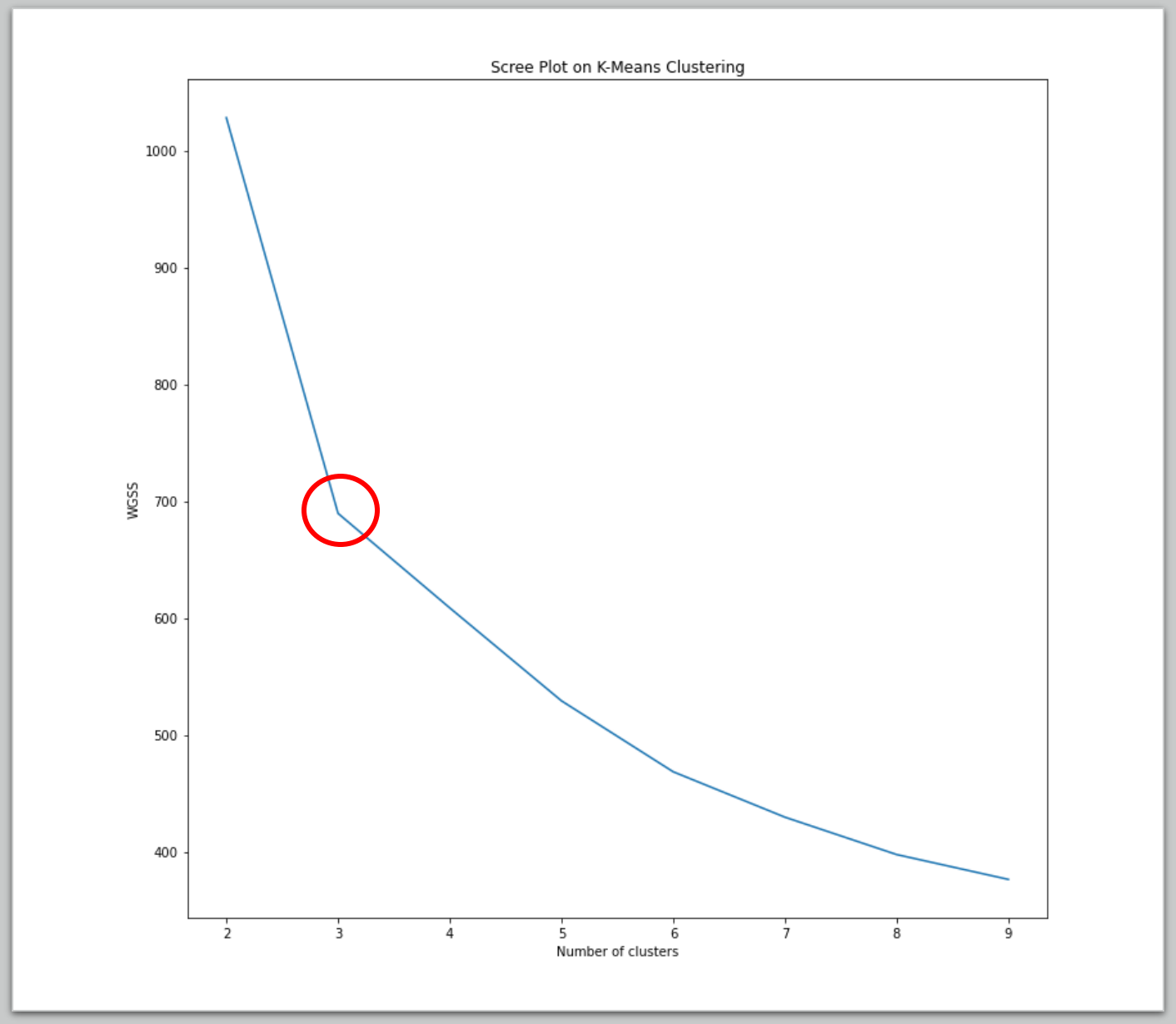




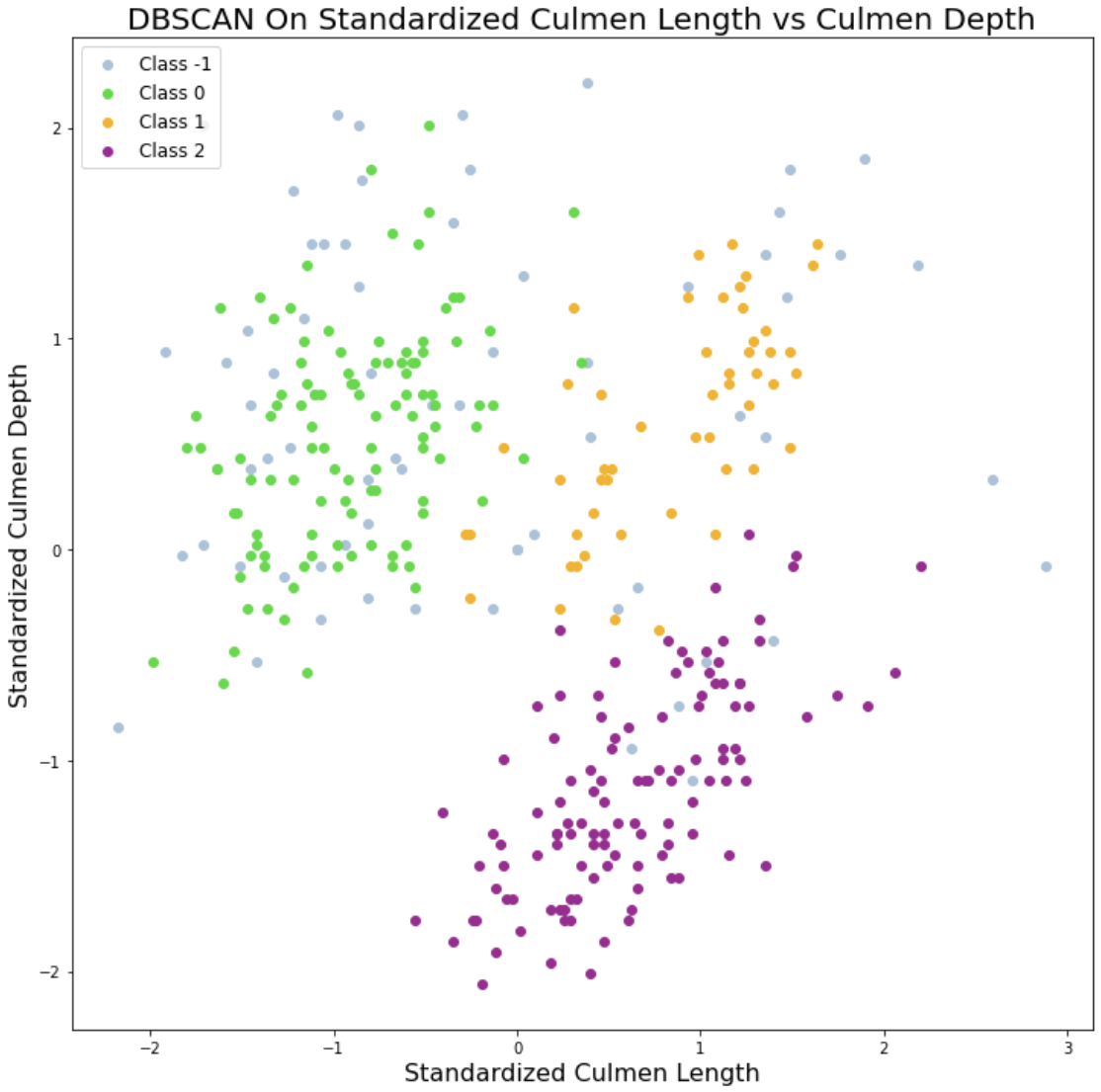
Performance Analysis

Performance Comparison

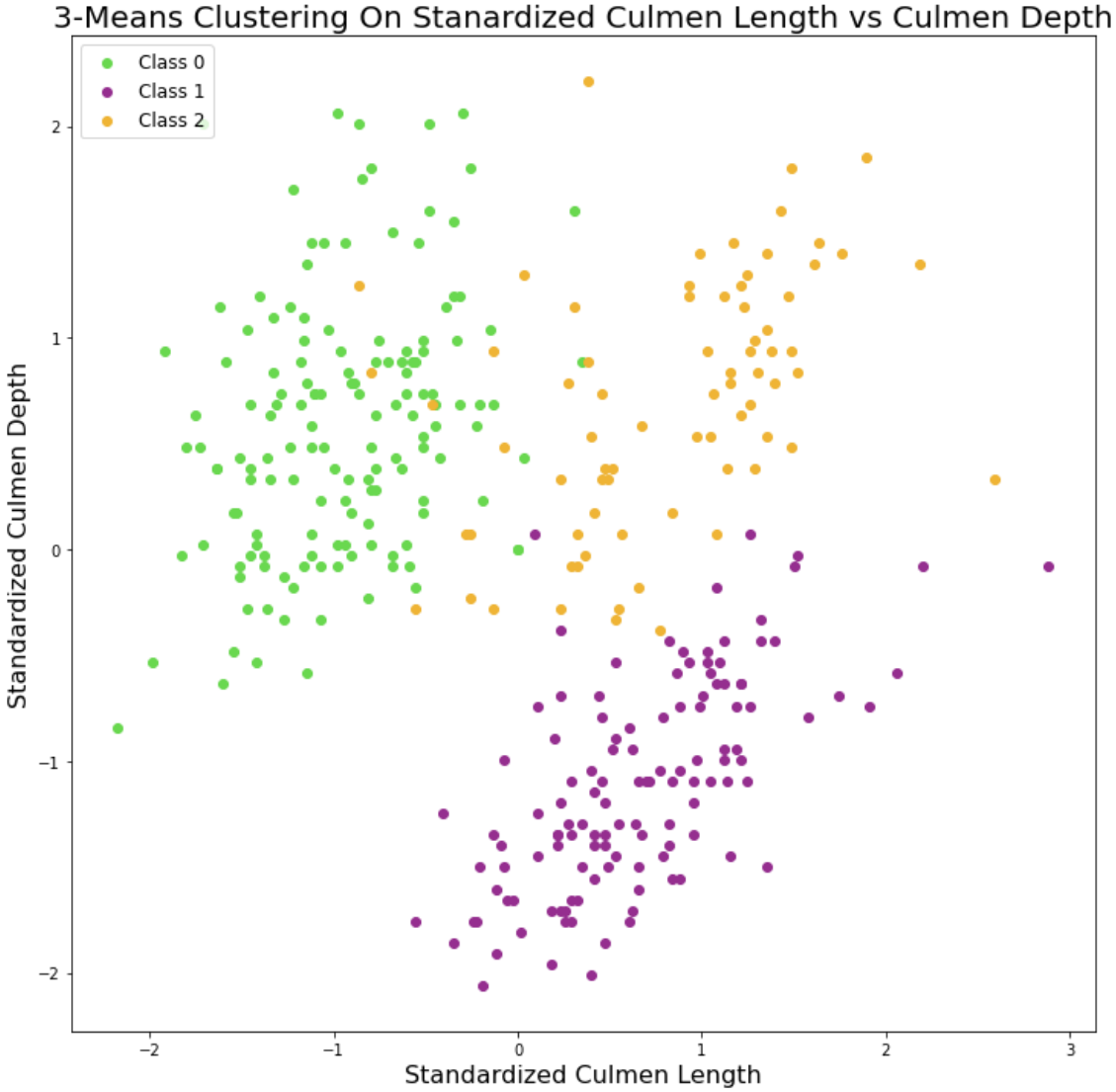
- Let's look at a similar models created using K-means clustering
- The scree plot of the variances for the different clusters, the model with 3 clusters seems to be optimal



DBSCAN



K-Means Clustering



Questions?