DBSCAN Implementation

By Adam Brewer, Tarick Mehanna, Robert Stewart

Data Processing Steps
(Penguin Dataset)

Steps		Results
1. Find all Null values in the dataset	→	Limited Nulls found, except for 'Comments' column
2. Remove columns with over 50% of Null values	→	Remove 'Comments' column
3. Do general outlier detection	→	No Outliers detected
4. Remove rows with over 50% of Null values	→	Only 2 rows removed
5. Impute with median to columns with Nulls	→	33 values imputed in total
6. Remove categorical columns lacking impactful information for model	→	2 columns removed
7. Standardize all data using sklearn StandardScaler() function	→	mean = 0 and stdev = 1 for each column

Model Creation

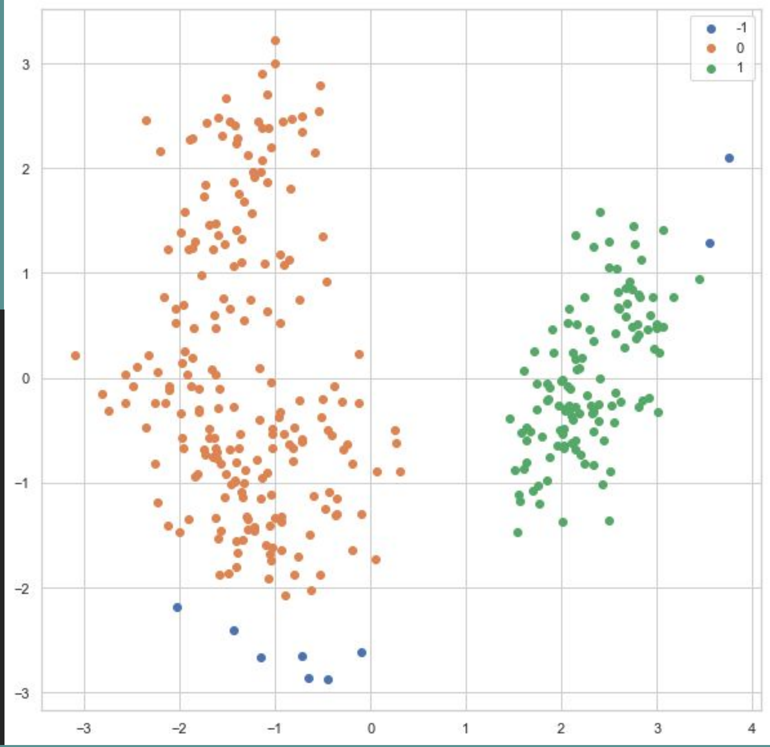
- Created model using **PCA** (principal component analysis) function
  - Converts high dimensional data to low dimensional data by selecting most import features
- Default values for hyperparameters
- Evaluated performance based on **silhouette score**
  - 0.391 - decent value, indicates no overlapping clusters or mislabeled data points

```
def DBScan_Plot_2D(dataframe, epsilon = 0.5, minimum_samples = 5):
    pca = PCA(2)
    df = pca.fit_transform(dataframe)

    db = DBSCAN(eps=epsilon, min_samples=minimum_samples).fit(df)
    label = db.labels_
    u_labels = np.unique(label)
    fig = plt.figure(figsize=(10, 10))
    for i in u_labels:
        plt.scatter(df[label == i,0], df[label == i, 1], label = i)
    plt.legend()
    plt.show()
    return db
```

DBScan

Defaults: epsilon = 0.5, min_samples = 5



Hypertuning - min_samples

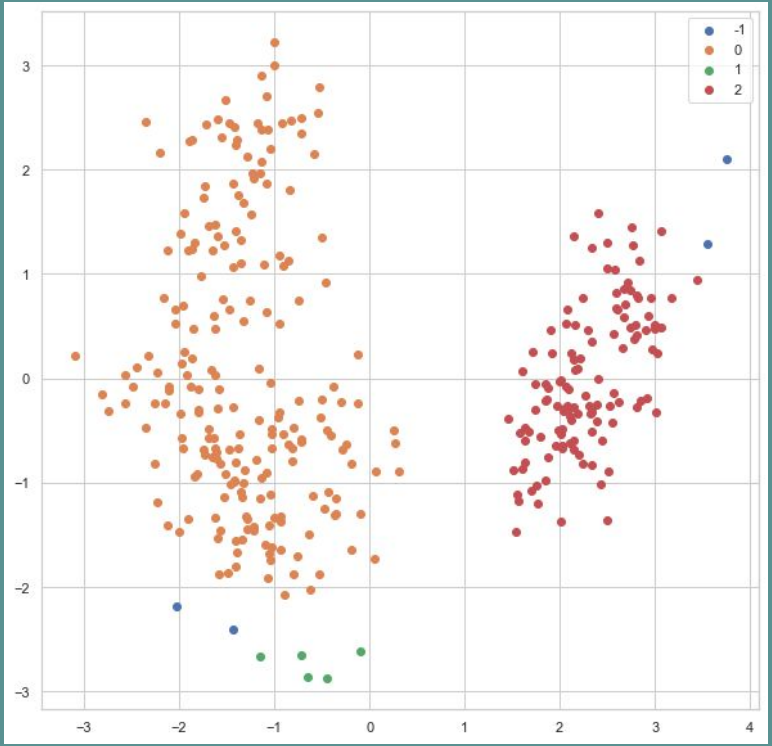
- Parameter should correspond to the size of the data set as well as **‘noisiness’**
- Good rule of thumb: choose a min_samples value that is greater than or equal to the dimensionality of the data set
- Optimal values are typically around **2 times the number of features**

Recall from the previous slide, we use PCA(2), indicating 2 features will be selected

Optimal min_samples for this model appears to be **4**, silhouette score = 0.308

DBScan

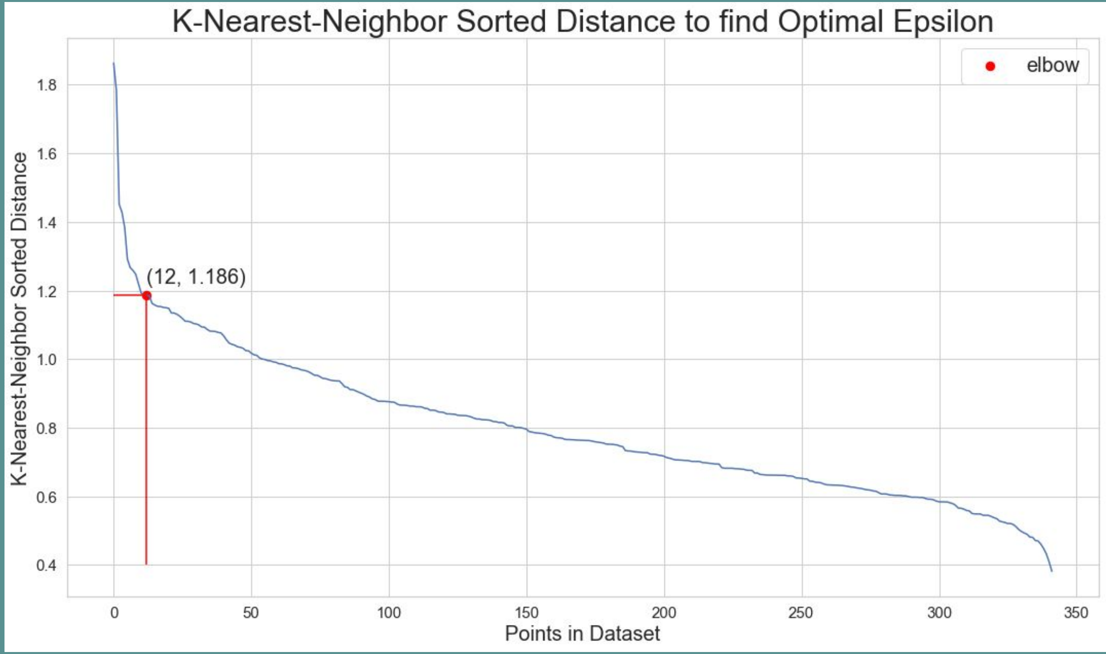
epsilon = 0.5, min_samples = 4



Hypertuning - epsilon

Used K-Nearest Neighbors algorithm to find optimal epsilon distance

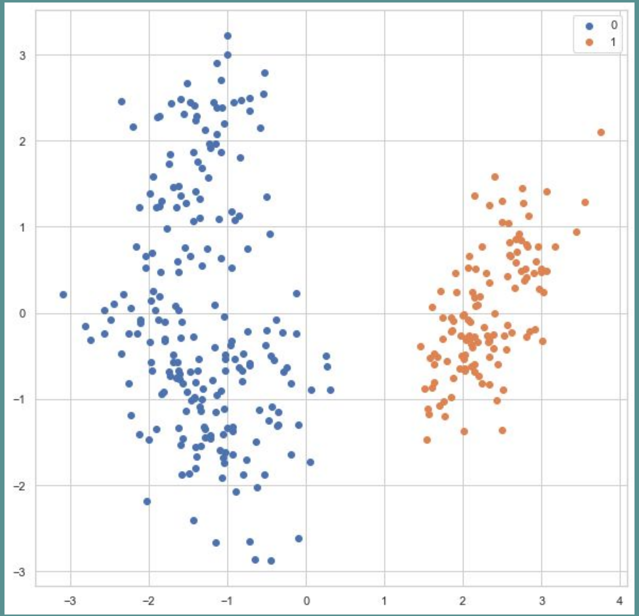
Process: If you have N-dimensional data to begin, then choose n_neighbors in sklearn NearestNeighbors to be equal to 2xN - 1, and find out distances of the K-nearest neighbors for each point in your dataset. Sort these distances out and plot them to find the **elbow** which separates **noisy points** (with high K-nearest neighbor distance) from points (with relatively low K-nearest neighbor distance) **which will most likely fall into a cluster**. The distance at which this elbow occurs is your point of **optimal epsilon,** here seen equal to about **1.186**.



Model Comparisons

DBScan

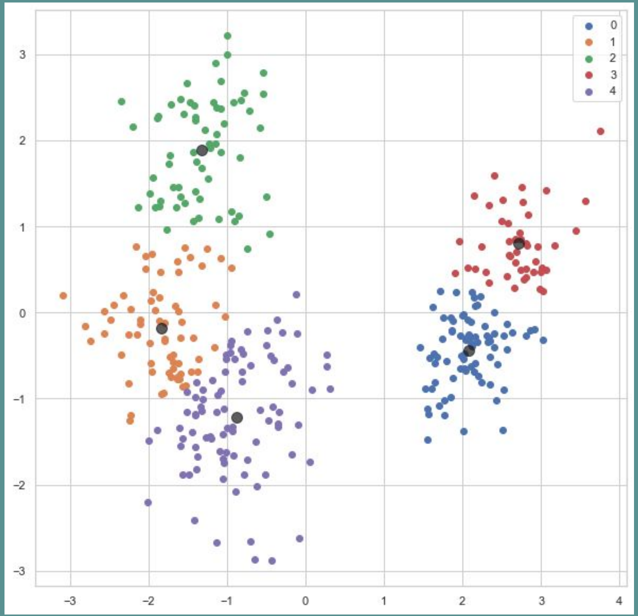
epsilon = 1.186, min_samples = 4



Silhouette Score: 0.462

K-Means

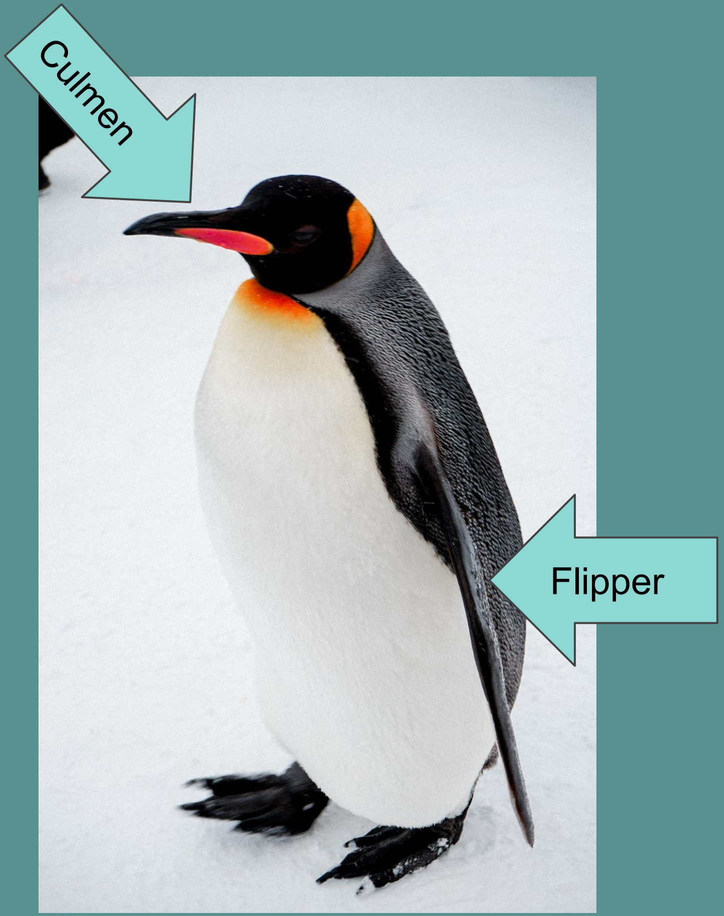
n_clusters = 5



Silhouette Score: 0.453

Conclusion

- Silhouette score, is it useful?
- Data analysis of the clusters
- Improving the model
  - More Data
  - Weighting to certain features
  - Less features



Sources

Special thanks to the Research Team!

Research Product: https://github.com/aslemc/ML-Assessment