# Digits Clusterization

**Unsupervised Machine Learning**

Irene Moyano Fernández

18/10/2020

# Framing the problem

Trying to find out the best number of clusters to understand out data, we followed the next steps:

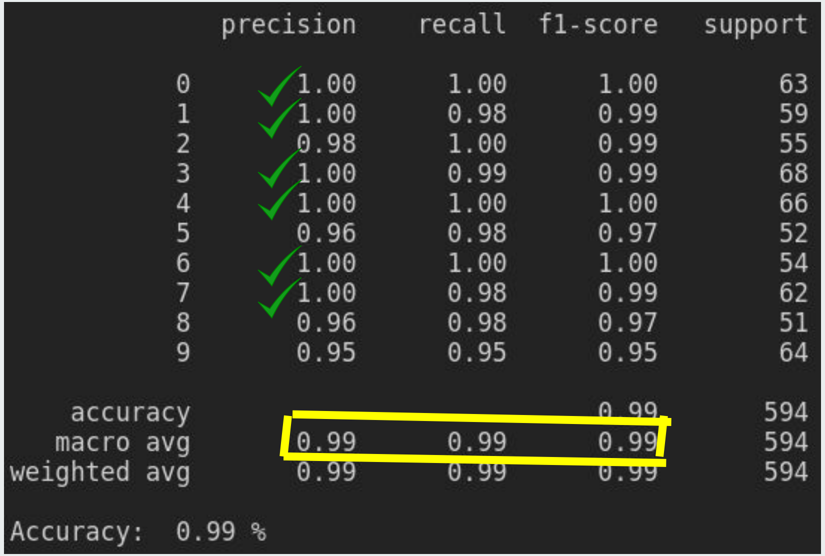
1. Dimensionality Reduction: Finding is the best method for our data (SVM Classifier)
2. Clustering Techniques
   - Kmeans
   - MiniBatchKmeans
   - Birch
   - Agglomerative
3. Performance Conclusions: Metrics
4. Next steps: Tuning

# 1. Dimensionality Reduction

# 1. DIMENSIONALITY REDUCTION: Finding the best method for our data

## PCA

(30 components, 85% variance ratio)

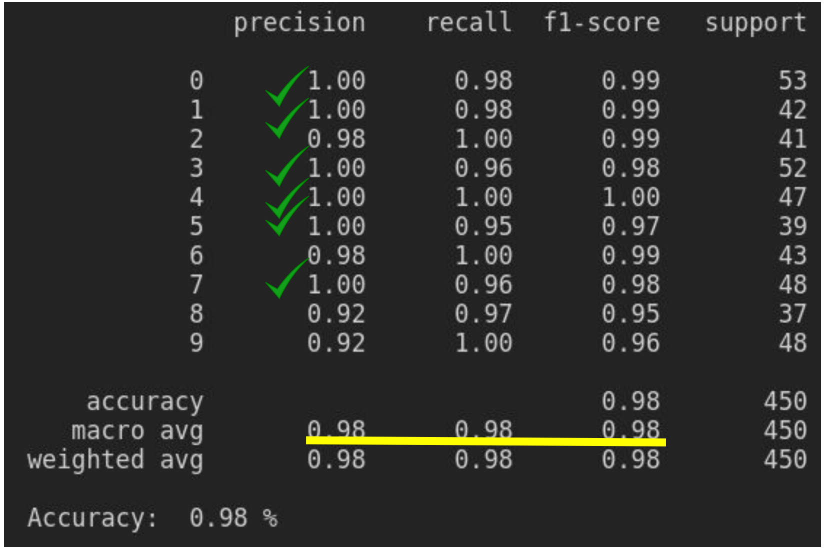


```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        63
           1       1.00      0.98      0.99        59
           2       0.98      1.00      0.99        55
           3       1.00      0.99      0.99        68
           4       1.00      1.00      1.00        66
           5       0.96      0.98      0.97        52
           6       1.00      1.00      1.00        54
           7       1.00      0.98      0.99        62
           8       0.96      0.98      0.97        51
           9       0.95      0.95      0.95        64

    accuracy                           0.99       594
   macro avg       0.99      0.99      0.99       594
weighted avg       0.99      0.99      0.99       594

Accuracy:  0.99 %
```

?

## LDA

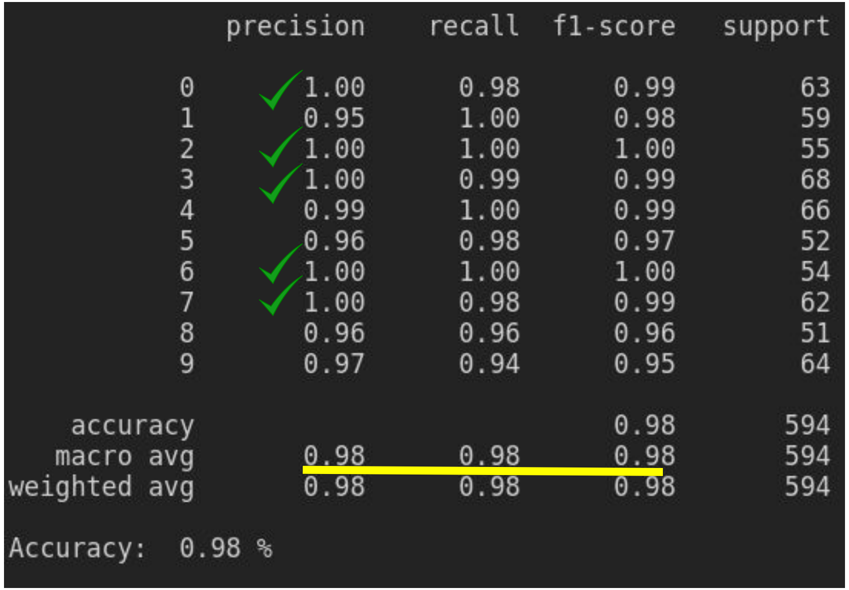
(3 comp, 95% variance ratio)



```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99        53
           1       1.00      0.98      0.99        42
           2       0.98      1.00      0.99        41
           3       1.00      0.96      0.98        52
           4       1.00      1.00      1.00        47
           5       1.00      0.95      0.97        39
           6       0.98      1.00      0.99        43
           7       1.00      0.96      0.98        48
           8       0.92      0.97      0.95        37
           9       0.92      1.00      0.96        48

    accuracy                           0.98       450
   macro avg       0.98      0.98      0.98       450
weighted avg       0.98      0.98      0.98       450

Accuracy:  0.98 %
```

# 1. DIMENSIONALITY REDUCTION: Finding the best method for our data

## SVM on original data



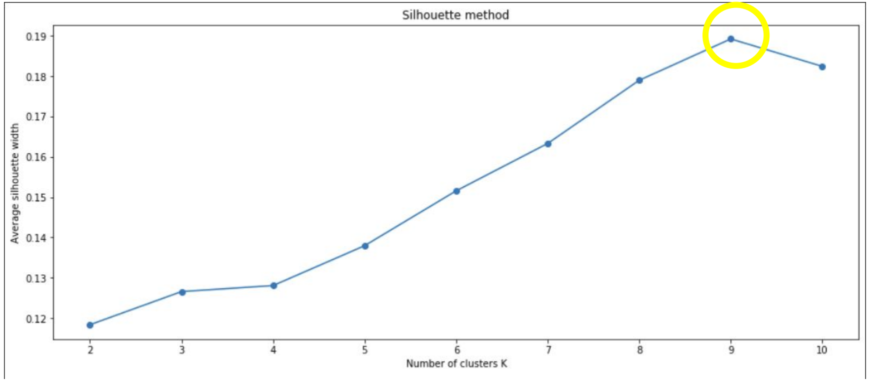
```
              precision    recall  f1-score   support

           0    ✓ 1.00      0.98      0.99        63
           1      0.95      1.00      0.98        59
           2    ✓ 1.00      1.00      1.00        55
           3    ✓ 1.00      0.99      0.99        68
           4      0.99      1.00      0.99        66
           5      0.96      0.98      0.97        52
           6    ✓ 1.00      1.00      1.00        54
           7    ✓ 1.00      0.98      0.99        62
           8      0.96      0.96      0.96        51
           9      0.97      0.94      0.95        64

    accuracy                          0.98       594
   macro avg      0.98      0.98      0.98       594
weighted avg      0.98      0.98      0.98       594

Accuracy:  0.98 %
```

| | ✓ | ✗ (Under 0.95) | Precision | Recall | F1-score |
|---|---|---|---|---|---|
| Original | 5 | 0 | 98% | 98% | 98% |
| PCA | 6 | 0 | 99% | 99% | 99% |
| LDA | 6 | 2 | 98% | 98% | 98% |

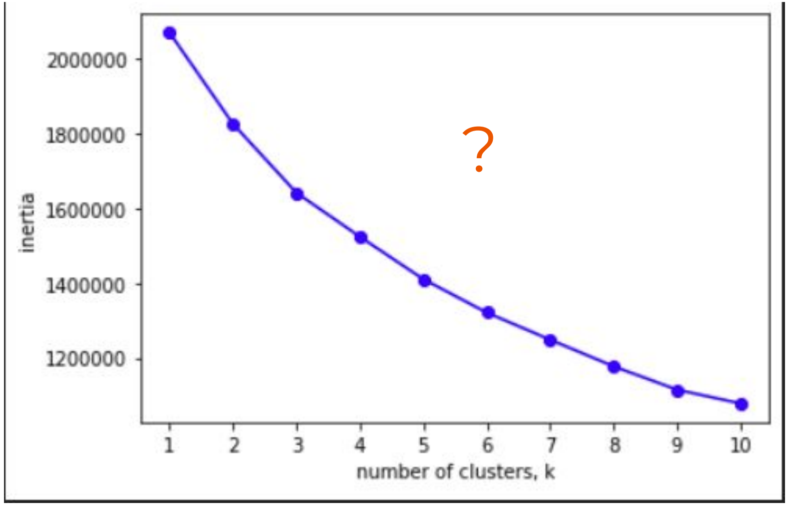
# 2. Clustering Techniques

# 2.0. Checking # clusters

Silhouette Method → 9 clusters



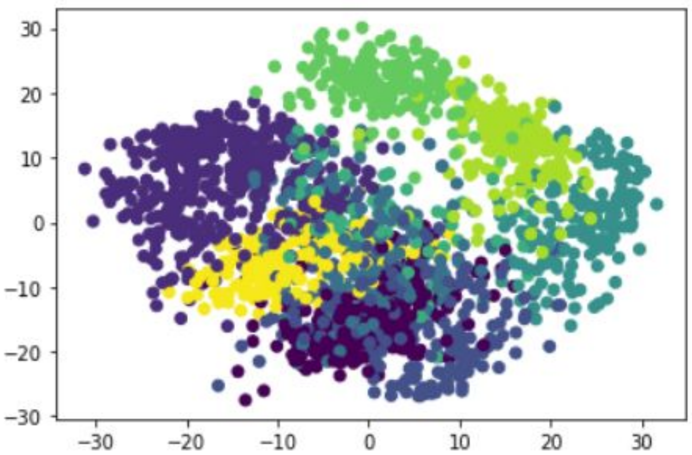
Elbow Method → ?

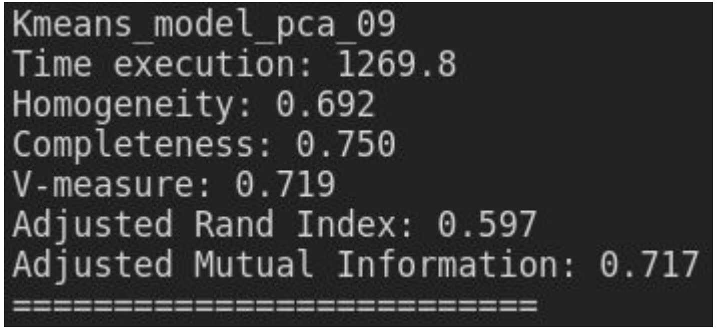


# 2. CLUSTERING TECHNIQUES: Finding the best method for our data

## 2.1. KMeans

- With PCA



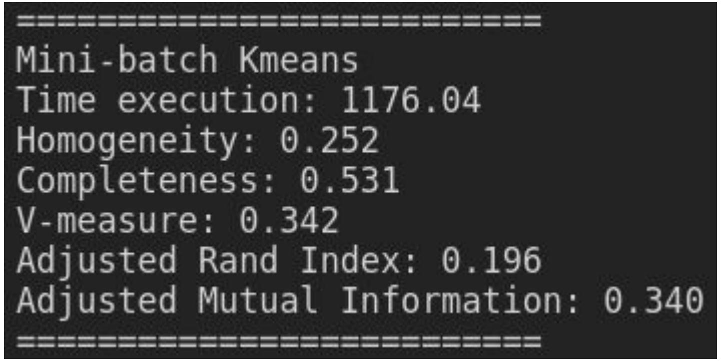


```
Kmeans_model_pca_09
Time execution: 1269.8
Homogeneity: 0.692
Completeness: 0.750
V-measure: 0.719
Adjusted Rand Index: 0.597
Adjusted Mutual Information: 0.717
==========================
```

# 2.2. MiniBatchKMeans (on PCA & 9 clust)

- We expected quite faster results that with KMeans but in this case the difference has just slightly improved:

1269.8 vs 1176.04

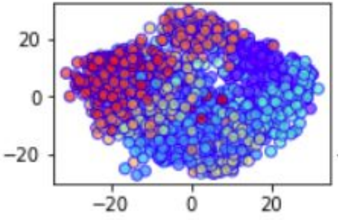


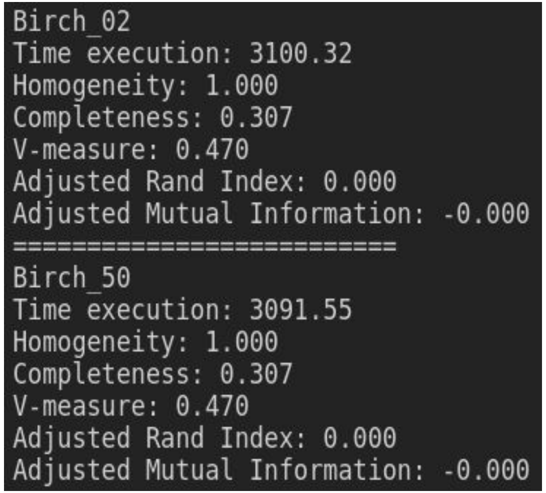
```
==========================
Mini-batch Kmeans
Time execution: 1176.04
Homogeneity: 0.252
Completeness: 0.531
V-measure: 0.342
Adjusted Rand Index: 0.196
Adjusted Mutual Information: 0.340
==========================
```

# 2. CLUSTERING TECHNIQUES: Finding the best method for our data

## 2.3. Birch

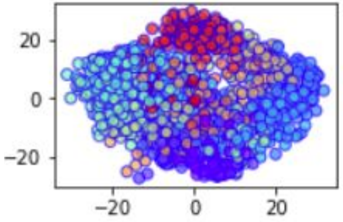
brc_02 = Birch(branching_factor=2, n_clusters=None, threshold=1.5)





```
Birch_02
Time execution: 3100.32
Homogeneity: 1.000
Completeness: 0.307
V-measure: 0.470
Adjusted Rand Index: 0.000
Adjusted Mutual Information: -0.000
==========================
Birch_50
Time execution: 3091.55
Homogeneity: 1.000
Completeness: 0.307
V-measure: 0.470
Adjusted Rand Index: 0.000
Adjusted Mutual Information: -0.000
```

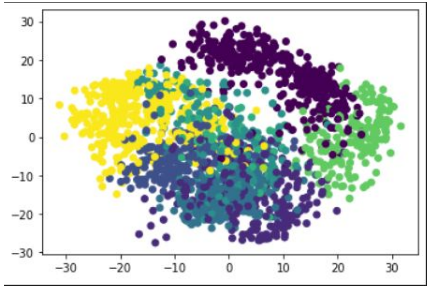
brc_50 = Birch(branching_factor=50, n_clusters=None, threshold=1.5)

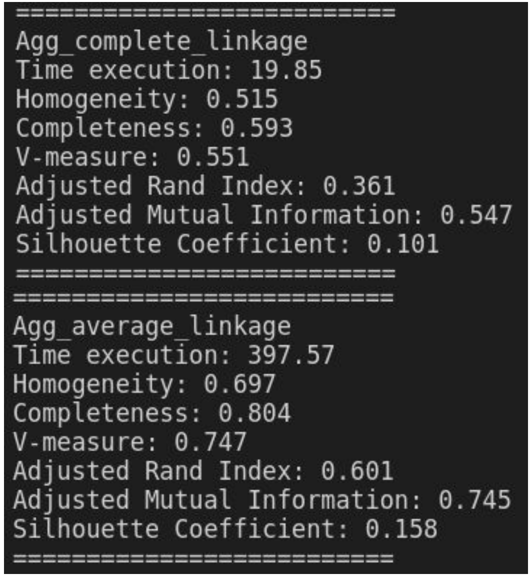


# 2. CLUSTERING TECHNIQUES: Finding the best method for our data

## 2.4. Agglomerative

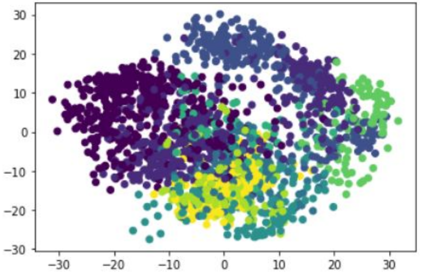
Linkage = **average**





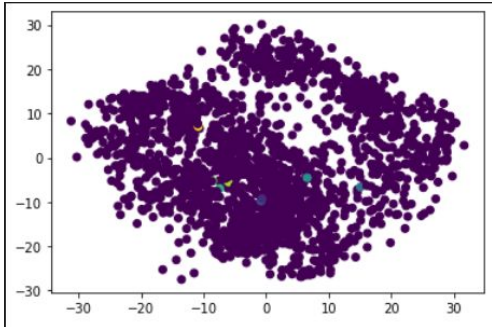
```
============================
Agg_complete_linkage
Time execution: 19.85
Homogeneity: 0.515
Completeness: 0.593
V-measure: 0.551
Adjusted Rand Index: 0.361
Adjusted Mutual Information: 0.547
Silhouette Coefficient: 0.101
============================
============================
Agg_average_linkage
Time execution: 397.57
Homogeneity: 0.697
Completeness: 0.804
V-measure: 0.747
Adjusted Rand Index: 0.601
Adjusted Mutual Information: 0.745
Silhouette Coefficient: 0.158
============================
```

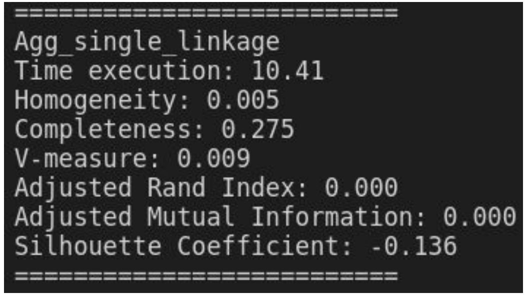
Linkage = **complete**



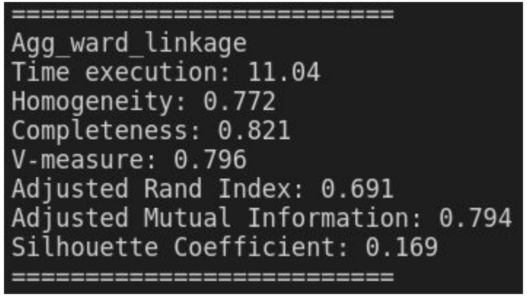
# 2.4. Agglomerative

Linkage = **single**



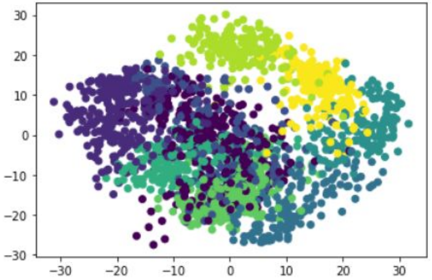


```
============================
Agg_single_linkage
Time execution: 10.41
Homogeneity: 0.005
Completeness: 0.275
V-measure: 0.009
Adjusted Rand Index: 0.000
Adjusted Mutual Information: 0.000
Silhouette Coefficient: -0.136
============================
```



```
============================
Agg_ward_linkage
Time execution: 11.04
Homogeneity: 0.772
Completeness: 0.821
V-measure: 0.796
Adjusted Rand Index: 0.691
Adjusted Mutual Information: 0.794
Silhouette Coefficient: 0.169
============================
```

Linkage = **ward**



# 2. Clustering Conclusions

KMeans, MiniBatchKMeans, Agglomerative, Birch

- **KMeans PCA over KMeans**:

  ↓ Computational cost

- **MiniBatchKmeans over KMeansPCA**:

  ↓ Time
  ↓ Performance

- **Agglomerative**:

  Different linkages perform indistinctly mediocre except for the “single” linkage which is even worse because results in a single cluster.

- **Birch:**

  It has the best performance in Time, Homogeneity, V-measure, and David Bouldin metrics. Completeness is quite good too.

# 3. Performance Conclusions: METRICS

# 3. PERFORMANCE CONCLUSIONS: Metrics

| METRIC | TECHNIQUE | SCORE 1 | TECH 2 | TECH 3 |
|---|---|---|---|---|
| *Time execution:* | OPTICS | 1123.58 | DB-SCAN | Birch_50 |
| *Homogeneity:* | Birch_02, Birch_50 & DBSCAN | 1 | Kmeans | Agg_avg_link |
| *Completeness:* | OPTICS | 1 | Agg_avg_link | Kmeans |
| *V-measure:* | Kmeans | 0.71 | Agg_avg_link | Birch_02 & Birch_50 |
| *Adjusted Rand Index:* | Kmeans | 0.59 | Agg_avg_link | MiniBatch |
| *Adjusted Mutual Information:* | Kmeans | 0.71 | Agg_avg_link | MiniBatch |

# Kmeans

(default n_clusters = 9)

Gràcies!

¡Gracias!

Thank you!