FEATURE ENGINEERING
for Clustering

Alessandra Monaco

06-12-2020

1 Introduction

The task is to cluster California housing dataset. The clustering algorithm chosen is the K-means++.

2 Clustering on raw data

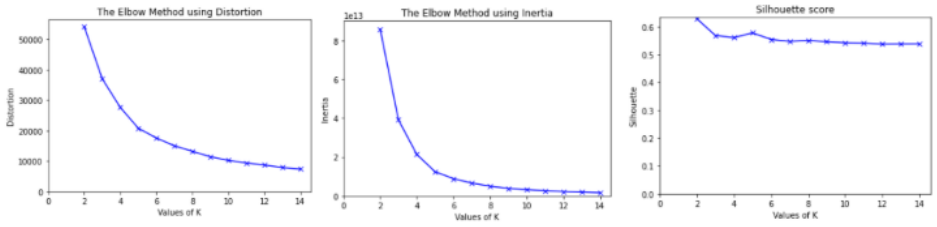
As first attempt, we considered all the features of the dataset, without any change or transformation. We just had to encode `ocean_proximity` with `LabelEncoder`, because we could not run Kmeans++ using categorical data.

```
le = preprocessing.LabelEncoder()
X['ocean_proximity'] = le.fit_transform(X['ocean_proximity'])
```

We also had to drop NaN values to avoid errors:

```
X = X.dropna()
```

We applied Elbow method to find the number of optimal clusters, considering together inertia, distortion and silhouette score, using a specific function that we defined (`elbow(dataframe, max_k)`). The function plots the curves of the different scores of the metrics, from 2 clusters to max_k.

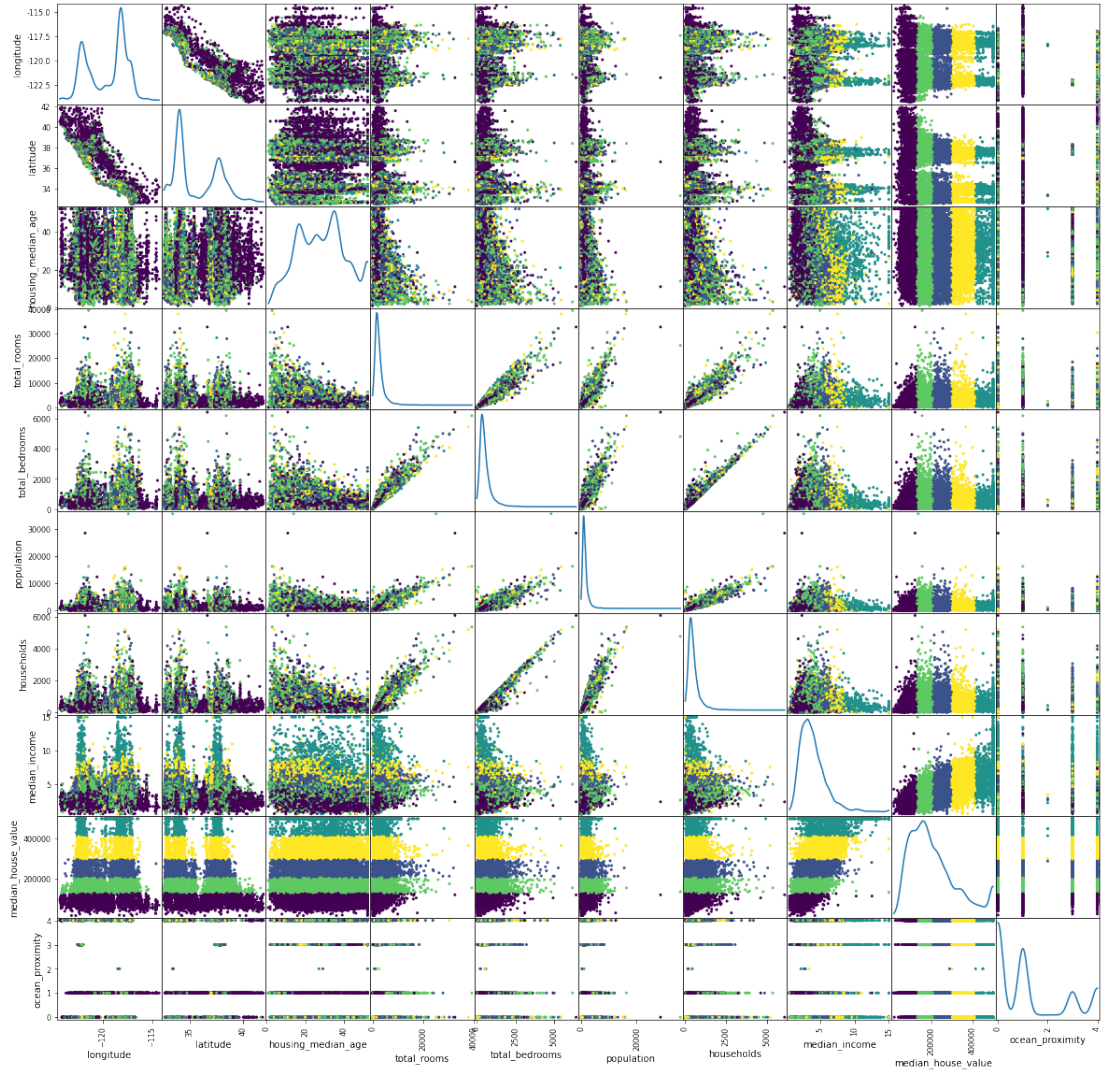


The plots show that the optimal number of clusters is 5. Using 5 clusters and the 10 raw features, we have the following cluster sizes and metrics:

```
|Cluster  0 |:  5660
|Cluster  1 |:  4505
|Cluster  2 |:  1600
|Cluster  3 |:  6362
|Cluster  4 |:  2306

Silhouette score average:  0.5772727824870366
Inertia:  12267350191978.76
Distortion:  20796.243833006538
```

We visualized the clusters plotting the scattermatrix.
It is clearly noticeable that our clustering algorithm is just separating clusters basing on `median_income_value` and `median_income`, but it does not happen because they are the most discriminant and meaningful features, but just because their values are much higher than the other ones, and bias the clustering. When using algorithms like K-means, in fact, it is quite mandatory to **normalize** or **scale** data.



3 Clustering after feature engineering

After understanding the importance of normalization for clustering tasks, we tried to better understand our data doing some **Exploratory Data Analysis**, as part of our Feature Engineering process.

The main steps were:

- printing some aggregating statistics;
- looking for **missing values**: 207 values of `total_bedrooms` were missing. We decided to fill those values taking the median of houses having the same number of `total_rooms`. After that, 15 rows still had a NaN (because there was no house with the same number of rooms), and we decided to just drop them since they were few.
- looking for **correlation** between features, inspecting the scatter matrix. We noticed positive correlations between the pairs `total_rooms and total_bedrooms`, `total_rooms and population`, `total_rooms and households`, `total_bedrooms and households`, `total_bedrooms and population`, `population and households`, `median_house_value and median_income`, and a negative correlation between `latitude and longitude`. Having a lot of correlated features may be overly redundant and you may be using more data than you need to reach the same patterns. Therefore, we decided to reduce the redundancy of the data considering **computations** of correlated features, instead of considering those features individually.
  This is the list of computed features:
  - 'angle' = longitude / latitude
  - 'radius' = sqrt(longitude^2 + latitude^2)
  - 'population_*_income' = population * income
  - 'households_population_ratio' = households / population
  - 'bedrooms/households' = bedrooms / households
  - 'bedrooms_rooms_ratio' = total_bedrooms / total_rooms
  - 'total_other_rooms' = total_rooms - total_bedrooms
  - 'cumulative_wealth' = median_income + median_house_value

Notice that we computed the new features taking into account their correlation, and considering just combinations that seem reasonable for our problem: we did not combine in a random way, because we want to obtain reasonable clusters for the domain that we are considering. The idea of 'reasonable clusters' is really subjective, and it's basically what guided us

in the choice of features: clustering groups items together is based on some similarity related to the features, but there is not a real definition of similarity between houses. We considered relevant the age of the house, the value and the income, the location but also how the house is built, for example relations regarded the number of rooms, and some info about the people living there: this is why we created exactly those features, instead of other ones.

As next step, we changed the encoding of the categorical feature, from LabelEncoding to **OneHotEncoding** using:

```
X_feat = pd.get_dummies(X_feat)
```

There are just 5 categories, so we are not increasing the feature dimensionality too much. (And also, we noticed that OneHot performs really better than LabelEncoding).
At the end we also normalized all the features in [-1,1]:

```
X_feat = X_feat / X_feat.max()
```

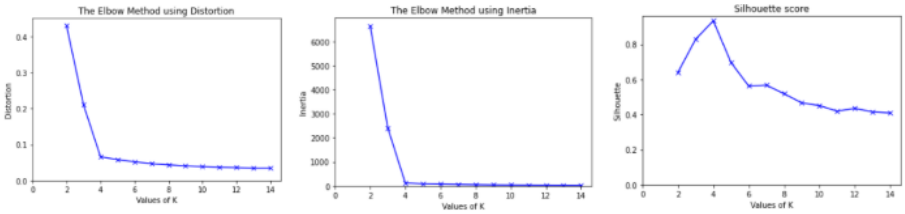
At the end of this process we end up with 14 features (the new ones, 'housing_median_age', and the 5 features obtained with one hot encoding).

Since Kmeans is sensible to outliers (they may have impacts on centroids), we consiered useful to perform **outlier detection and removal**, using the anomaly detection algorithm `IsolationForest`, trying to balance the contamination coefficient to detect the correct amount of outliers (if they are few, it is useless, if they are too many, we may remove too many data). We used a contamination of 0.03, removing 619 outliers.

As last step, we considered quite important to perform **dimensionality reduction** using **PCA**, since the number of features is high and may introduce noise.

```
pca = decomposition.PCA(n_components=3)
pca.fit(X_feat)
X_pca = pca.transform(X_feat)
```

Again, we run the Elbow method, getting a clear improvement:

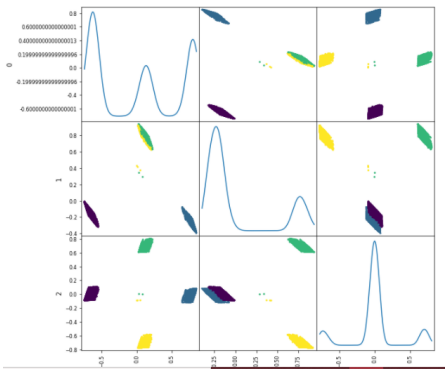


The optimal number of clusters is now 4 (and this is evident especially in the Silhouette curve), and in addition, all the metrics improved a lot:

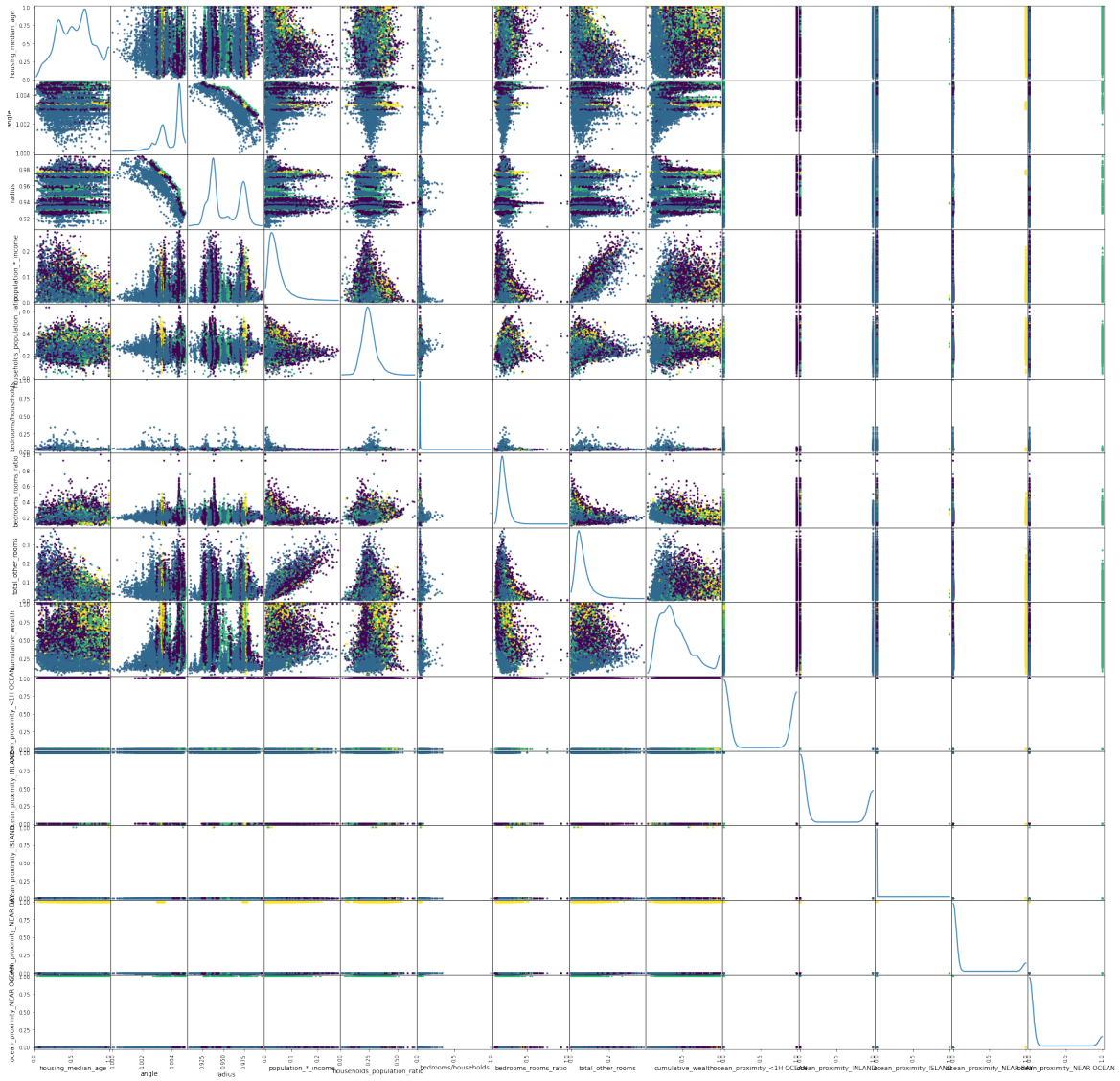
```
|Cluster  0 |:  9056
|Cluster  1 |:  6386
|Cluster  2 |:  2348
|Cluster  3 |:  2216

Silhouette score average:  0.9338677370491744
Inertia:  123.04558667758013
Distortion:  0.06605150705703018
```

The scatter matrix over PCA shows that for every pair of dimensions, the 4 clusters are clearly separated (but there are still few outliers):



Since PCA creates new dimensions using the eigenvectors, we may be interested in comparing the clustering on our features, plotting the scattermatrix (`scattermatrix_feature.png`).



Clusters are more distinct in each dimension, and it's evident that the categorical features are really discriminant and impacting how the clusters are created. It is reasonable, since grouping houses basing on their location it's coherent with our idea of "similar houses", but still we want also the other factors to impact on the clustering.
Therefore, in order to understand if our clusters make sense we performed a sort of **cluster analysis**, basically printing statistics and plotting the distribution of each feature in each cluster.

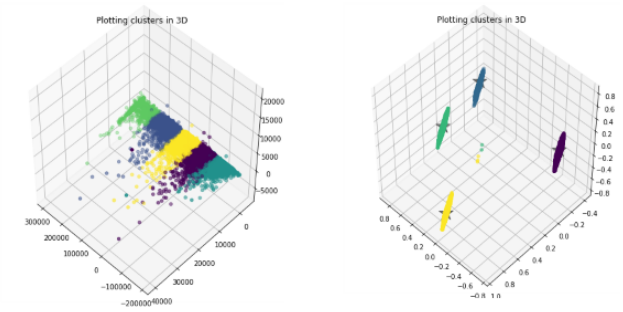
Histograms helped us to understand which features were more DISCRIMINANT (ocean_proximity, cumulative_wealth, housing_median_age, latitude and longitude related features) and less DISCRIMINANT (all room related features). Infact, features regarding rooms and bedrooms have similar statistics in the clusters, they are distributed in a very similar way: the shape of the house does not impact the cluster division, so maybe we should have combined in some other way (future works).

Anyway, our clusters seem to make sense, representing the following groups:

- CLUSTER 1: houses that are not too old neither too recent, in a middle wealth state, mostly located < 1 from the ocean;
- CLUSTER 2: mostly recent houses, more "poor" than the others, mostly inland;
- CLUSTER 3: houses with a quite good wealth state, moslty located on island or near the ocean;
- CLUSTER 4: mostly old houses, wealthy, located on islands or near a bay

4 Comments and conclusions

Our experiment was a step wise improvement: it started with a basic grouping based on house value, and ended with a group division that was based on different factors, not only the house value. The improvement is visible both in all the metrics that we used and visually in the plots. From the 3D plot using PCA dimensions (1st plot is raw data), it is evident that the separation between clusters has increased a lot, together with the **cohesion** between points in a cluster.



Also, the **Silhouette curve** in the first experiment was almost an horizontal line: for every number of clusters the quality would be the same, so it's just the number of cuts that you do on `median_house_value` ranges. For the second experiment the number of clusters impacts the Silhouette score, with a maximum on 4 clusters. Also, **inertia and distortion curves** in the first case are more flatten, requiring a higher number of clusters to sufficiently decrease.

Finally, **running time** needed for K-means++ to converge is slower with raw data, probably due to the higher number of features considered (10 features on raw data vs 3 features after PCA on new features).