K-Means with Scikit-Learn and Interpreting Results: Takeaways

Syntax

- Scaling the data:

```
from sklearn.preprocessing import StandardScaler


scaler = StandardScaler()
scaler.fit(df)


df_scaled = scaler.transform(df)
```

- Using the `KMeans` class:

```
from sklearn.cluster import KMeans


model = KMeans(n_clusters=k)
cluster = model.fit_predict(df_scaled)
```

- Other attributes of the `KMeans` object:
    - `model.inertia_` : the inertia resulting from the clusters split
    - `model.cluster_centers_` : the coordinates of the final centroids
    - `model.n_iter_` : the number of iterations needed to converge to the resulting clusters
    - `model.n_features_in_` : the number of features passed to the model
    - `model.feature_namesin_` : the name of the features passed to the model
- Cross-tabulation with Pandas:

```
pd.crosstab(index, columns, values, aggfunc, normalize)
```

Where:

- `index` : the values to be grouped in the rows
- `columns` : the values to be grouped in the columns
- `values` : the values to be aggregated given an aggregation function
- `aggfunc` : the aggregation function
- `normalize` : whether or not and how the values will be normalized

Concepts

- Standardization is performed by calculating the z-score for each observation in a column:

$$z = \frac{(x - \mu)}{\sigma}$$

Where:

- x is the data point.
- μ is the mean.
- σ is the standard deviation

Resources

- KMeans with scikit-learn
- Standardization with scikit-learn
- pd.crosstab