Project summary for kmeanseval

Background and motivation

K-means is the most common unsupervised learning algorithm used for clustering today. When clustering data using k-means, one of the most important decisions to make is how many clusters to use ("k"). This is a somewhat subjective decision, representing a tradeoff between how simply the clusters summarize the data (small k) vs. how well the clusters represent the data (large k). Picking a k value is typically done by comparing within-cluster-sum of squared errors ("WSS") and/or silhouette scores ("SS") at different values of k. This process often involves creating a chart to visually inspect how WSS and/or SS change at different values of k. This means that every k-means analysis involves some boilerplate code to loop through the creation of multiple k-means models at different values of k, calculate WSS or SS for them, and plot these results.

Project idea

The goal of this package is to reduce the amount of boilerplate code that goes into selecting a value for k when using k-means. Rather than creating loops to capture WSS and SS and plotting them from scratch, this can all be handled by creating a KMeansEvaluator object which wraps around the scikit-learn k-means interface to fit models and then provides methods to get the desired metrics and plots.

Design considerations

The overarching goal for the design of kmeanseval is to create a simple, lightweight package that can satisfy the majority of use cases for selecting k in a k-means analysis. Most of the design considerations in this project relate to the tradeoff between simplicity and functionality.

- **Algorithm support:** kmeanseval will only support k-means clustering. This does limit the usefulness of the package, however k-means is by far the most popular clustering method. Other clustering methods like GMM, DBSCAN, etc. would add complexity to the code and the interface, requiring support for additional algorithms and evaluation methods, while only serving more niche use cases.
- **Evaluation metric support:** kmeanseval will only support within-cluster-sum of squared errors and silhouette scores as evaluation metrics. Similar to the decision to only support k-means clustering, this simplifies the scope of the project while still providing the needed functionality for the most common use cases.
- **Algorithm source:** kmeanseval wraps around the scikit-learn k-means functions. This will allow any user familiar with using the k-means interface in scikit-learn to easily pick up the interface for kmeanseval.
- **Flexibility considerations:** My goal was to give users the flexibility required for kmeanseval to be useful, and limit flexibility where it isn't needed to simplify the interface and coding requirements. I decided that there were 5 ways users would need to be flexible:
    - Users should be able to create K-means models using any parameters they would be able to use in scikit-learn.

    - Plotting functionality will be kept basic, only allowing for changes in plot size and font size. Trying to anticipate all possible plotting needs is too large a task, and users can get the list of WSS/SS values from the KMeansEvaluator object anyway to make their own custom plots if absolutely necessary.
- **Coding paradigm:** kmeanseval is written in an object-oriented coding style. It works by using a "KMeansEvaluator" class which contains data in the form of a series of sklearn KMeans models. The class has methods that giving the user the WSS/SS for those models or plot the values for WSS/SS.
    - Given the small scope of this project and the fact that code rarely needs to be reused, it would have also made sense to use a procedural coding style and just create individual functions instead of a class for this project (and despite using a class, the way the class works is very "procedural").

Usage

The example workflows below both get a list of WSS for k = 2 through 10 and plot them. When using kmeanseval there are fewer lines of boilerplate code.

Workflow using kmeanseval:

```
Import kmeanseval
import pandas as pd

data = pd.read_csv('data.csv', sep=',')
kme = kmeanseval.KMeansEvaluator(data = data, k_range = range(2, 11))
wss = kme.get_metrics(method = 'wss')
kme.plot_elbow()
```

Workflow using scikit-learn and matplotlib:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

data = pd.read_csv('data.csv', sep=',')
wss = []
K = range(2, 11)
for k in K:
   model = KMeans(n_clusters=k)
   model.fit(data)
   wss.append(model.inertia_)

plt.figure(figsize=(20,10))
plt.plot(K, wss)
plt.xlabel('k')
plt.ylabel('Within-cluster-sum of squares')
plt.title('WSS for K = 2 through 10')
plt.show()
```

Comparison: clusteval

Link to clusteval: https://pypi.org/project/clusteval/

clusteval as described on PyPi: *"clusteval is a Python package for unsupervised cluster evaluation. Three evaluation methods are implemented that can be used to evaluate clusterings; silhouette, dbindex, and derivative. Four clustering methods can be used: agglomerative, kmeans, dbscan and hdbscan."*

Similar to kmeanseval, clusteval is a tool for calculating evaluation metrics in clustering. However, the two packages are functionally very different:

- clusteval supports k-means, agglomerative clustering, dbscan, and hdbscan while kmeanseval only supports k-means
- clusteval doesn't support WSS analysis – instead it supports analysis through silhouette scores, davies-bouldin index, and derivatives