File Edit View Run Kernel Tabs Settings Help

k_means_in_action.ipynb

Markdown Python 3

Students Do: Clustering costumers for e-commerce

Once you have prepared the data, it's time to start looking for patterns that could lead you to define customers clusters. After talking with the CFO of the company about the next quarter goals, you figured out that one way to understand customers, from the available data, is to cluster them according their spending capacity, however you have to find how many groups you can define.

You decide to use your new unsupervised learning skills and put K-Means in action!

[1]:
```
# Initial imports
import pandas as pd
from sklearn.cluster import KMeans
from path import Path
import plotly.express as px
import hvplot.pandas
```

Instructions

Accomplish the following tasks and use K-Means to cluster the customers data.

1. Load the data you already cleaned on a DataFrame and call it `df_shopping`.

[2]:
```
# Loading data
file_path = Path("../Resources/shopping_data_cleaned.csv")
df_shopping = pd.read_csv(file_path)
df_shopping.head(10)
```

[2]:

	Genre	Age	Annual Income	Spending Score (1-100)
0	1	19	15.0	39
1	1	21	15.0	81
2	0	20	16.0	6
3	0	23	16.0	77
4	0	31	17.0	40
5	0	22	17.0	76
6	0	35	18.0	6
7	0	23	18.0	94
8	1	64	19.0	3
9	0	30	19.0	72

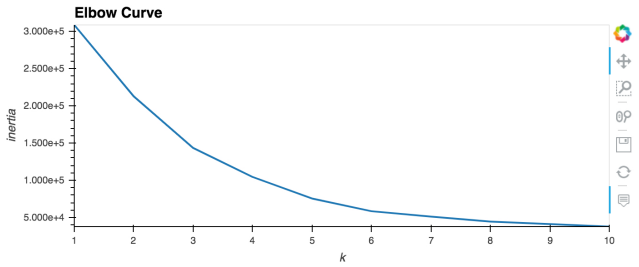
2. Find the best number of clusters using the Elbow Curve.

[3]:
```
inertia = []
k = list(range(1, 11))

# Calculate the inertia for the range ok k values
for i in k:
    km = KMeans(n_clusters=i, random_state=0)
    km.fit(df_shopping)
    inertia.append(km.inertia_)

# Create the Elbow Curve using hvPlot
elbow_data = {"k": k, "inertia": inertia}
df_elbow = pd.DataFrame(elbow_data)
df_elbow.hvplot.line(x="k", y="inertia", xticks=k, title="Elbow Curve")
```

[3]:



3. Create a function called `get_clusters(k, data)` that finds the k clusters using K-Means on `data`. The function should return a DataFrame copy of `Data` that should include a new column containing the clusters found.

[4]:
```
def get_clusters(k, data):
    # Initialize the K-Means model
    model = KMeans(n_clusters=k, random_state=0)

    # Fit the model
    model.fit(data)

    # Predict clusters
    predictions = model.predict(data)

    # Create return DataFrame with predicted clusters
    data["class"] = model.labels_

    return data
```

4. Use the `get_clusters()` function with the two best values for `k` according to your personal opinion; plot the resulting clusters as follows and postulate your conclusions:

   - Create a 2D-Scatter plot using `hvPlot` to analyze the clusters using `x="Annual Income"` and `y="Spending Score (1-100)"`.

   - Create a 3D-Scatter plot using Plotly Express to analyze the clusters using `x="Age"`, `y="Spending Score (1-100)"` and `z="Annual Income"`.

Analyzing Clusters with the First Best Value of k

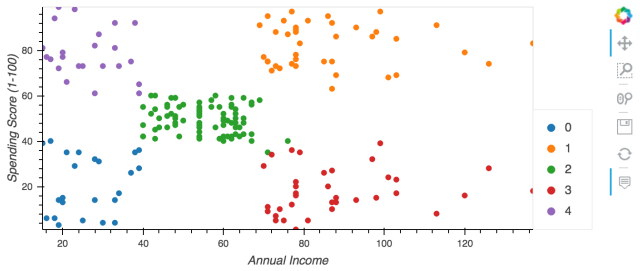
[5]:
```
# Looking for clusters the first best value of k
five_clusters = get_clusters(5, df_shopping)
five_clusters.head()
```

[5]:

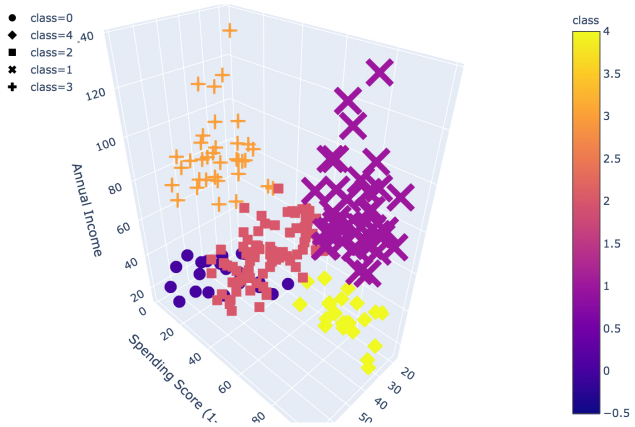
	Genre	Age	Annual Income	Spending Score (1-100)	class
0	1	19	15.0	39	0
1	1	21	15.0	81	4
2	0	20	16.0	6	0
3	0	23	16.0	77	4
4	0	31	17.0	40	0

[6]:
```
# Plotting the 2D-Scatter with x="Annual Income" and y="Spending Score (1-100)"
five_clusters.hvplot.scatter(x="Annual Income", y="Spending Score (1-100)", by="class")
```

[6]:



[7]:
```
# Plotting the 3D-Scatter with x="Annual Income", y="Spending Score (1-100)" and z="Age"
fig = px.scatter_3d(
    five_clusters,
    x="Age",
    y="Spending Score (1-100)",
    z="Annual Income",
    color="class",
    symbol="class",
    width=800,
)
fig.update_layout(legend=dict(x=0, y=1))
fig.show()
```



Analyzing Clusters with the Second Best Value of k

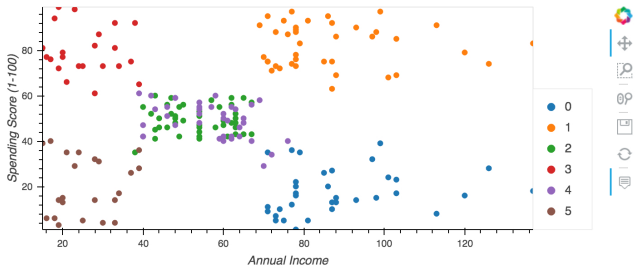
[8]:
```
# Looking for clusters the seconf best value of k
six_clusters = get_clusters(6, df_shopping)
six_clusters.head()
```

[8]:

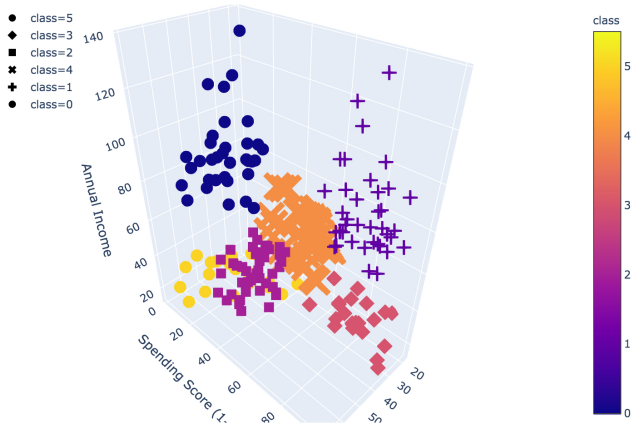
	Genre	Age	Annual Income	Spending Score (1-100)	class
0	1	19	15.0	39	5
1	1	21	15.0	81	3
2	0	20	16.0	6	5
3	0	23	16.0	77	3
4	0	31	17.0	40	5

[9]:
```
# Plotting the 2D-Scatter with x="Annual Income" and y="Spending Score (1-100)"
five_clusters.hvplot.scatter(x="Annual Income", y="Spending Score (1-100)", by="class")
```

[9]:



[10]:
```
# Plotting the 3D-Scatter with x="Annual Income", y="Spending Score (1-100)" and z="Age"
fig = px.scatter_3d(
    five_clusters,
    x="Age",
    y="Spending Score (1-100)",
    z="Annual Income",
    color="class",
    symbol="class",
    width=800,
)
fig.update_layout(legend=dict(x=0, y=1))
fig.show()
```



Sample Conslusions

- The best two values for `k` are `k=5` and `k=6` since on those values of `k` the curve turns showing an elbow.
- After visually analyzing the clusters, the best value for `k` seems to be `6`. Using `k=6`, a more meaningful segmentation of customers can be done as follows:
  - *Cluster 1*: Medium income, low annual spend
  - *Cluster 2*: Low income, low annual spend
  - *Cluster 3*: High income, high annual spend
  - *Cluster 4*: Low income, high annual spend
  - *Cluster 5*: Medium income, low annual spend
  - *Cluster 6*: Very high income, high annual spend
- Having defined these clusters, we can formulate marketing strategies relevant to each cluster aimed to increase revenue.