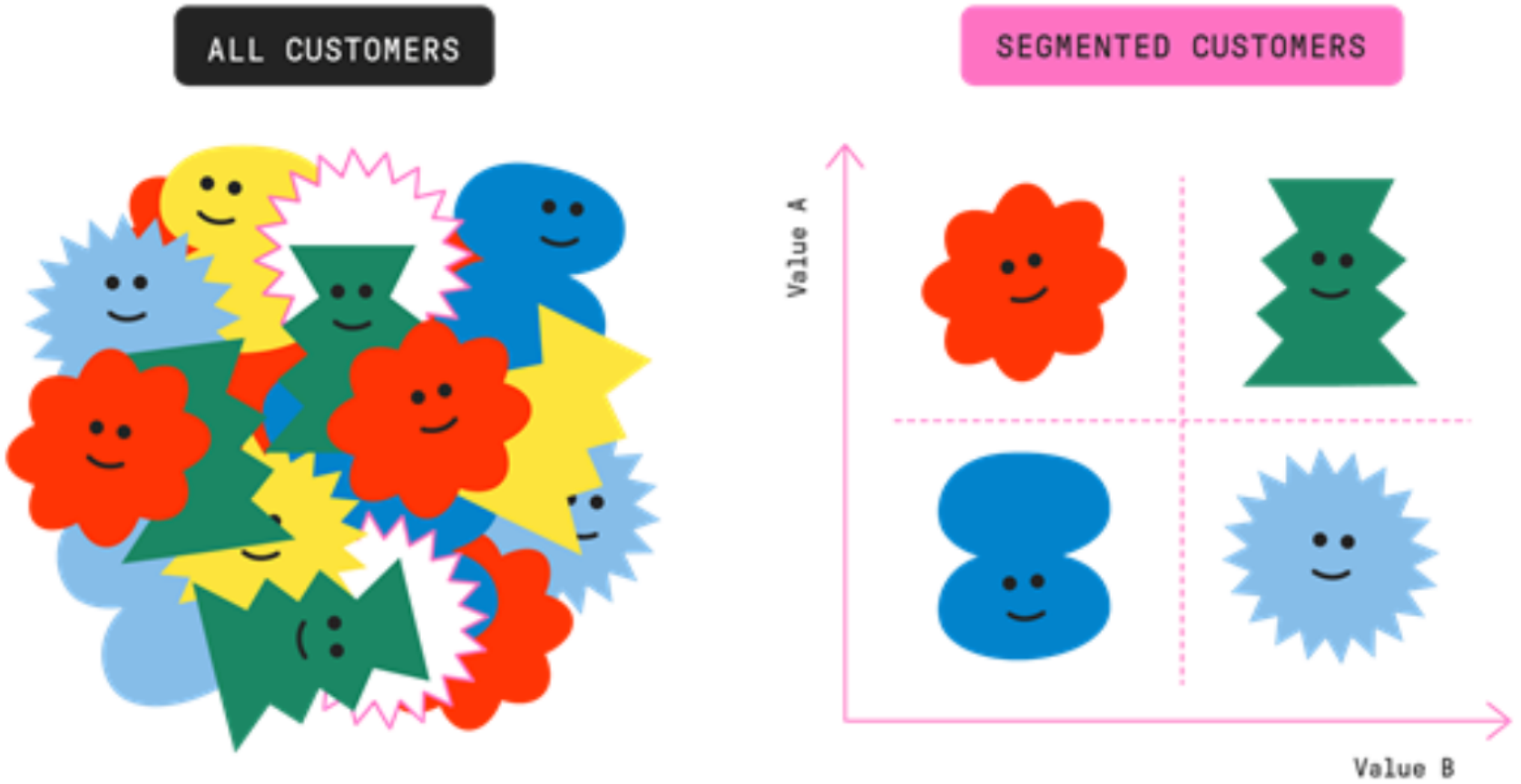
Activity 1 - Data Clustering ¶

Data clustering in big data refers to the process of dividing a large dataset into smaller, more manageable subgroups or clusters based on their similarities, with the goal of identifying patterns, relationships, and insights in the data.

This lab uses the Credit Card dataset https://www.kaggle.com/code/des137/customer-segmentation-credit-cards/data (https://www.kaggle.com/code/des137/customer-segmentation-credit-cards/data)



Load the data using pandas and inspect the head.

In [ ]:
```
import pandas as pd
import numpy as np

#Load the data using pandas read_csv function.
credit_data = pd.read_csv("CC_GENERAL.csv")
credit_data.head()
```

Use the describe function to get a feel for the data, and the categorys

In [ ]:
```
credit_data.describe()
```

Use the info function to get a feel for the different categories and there counts and data type.

In [ ]:
```
credit_data.info()
```

Using the drop function, remove the 'CUST_ID' column as we don't need this piece of information.

In [ ]:
```
credit_data.drop(['CUST_ID'], axis=1, inplace=True)
credit_data.describe()
```

Inspect the original data see if we have an NA or missing values

In [ ]:
```
credit_data.isna().mean() * 100
```

Based on the column(s) you found had missing values, replace the data with an appropriate fill value (Median, Mean etc.)

In [ ]:
```
credit_data['TENURE'].fillna(credit_data['TENURE'].median(), inplace=True)
credit_data.describe()
```

For each data column, plot the Kernel Density Estimate using the seaborn package.

In [ ]:
```
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.figure_factory as ff
import seaborn as sns
```

Inpsect the KDE plots and consider which columns you think are important to the Credit Card dataset.

Consider how the plots are skewed, and the variation across the plots. Because we're going to be using clustering to get a good visualisation we want to include the skewness.

In [ ]:
```
cols = ['BALANCE', 'ONEOFF_PURCHASES', 'INSTALLMENTS_PURCHASES', 'CASH_ADVANCE', 'ONEOFF_PURCHASES_FREQUENCY','PURCHA
```

Plot a a heatmap of the correlations of your columns

In [ ]:
```
plt.figure(figsize=(14,14))
sns.heatmap(credit_data.corr(), annot=True)
plt.show()
```

In [ ]:
```
plt.figure(figsize=(10,6))
sns.pairplot(
    credit_data,
#     x_vars=['PRC_FULL_PAYMENT', 'BALANCE_FREQUENCY', 'CASH_ADVANCE_FREQUENCY','CASH_ADVANCE_TRX'],
#     y_vars=['PRC_FULL_PAYMENT', 'BALANCE_FREQUENCY', 'CASH_ADVANCE_FREQUENCY','CASH_ADVANCE_TRX'],kind="reg", plot_
    x_vars=['CASH_ADVANCE_FREQUENCY',],
    y_vars=['CASH_ADVANCE_TRX'],kind="reg", plot_kws={'line_kws':{'color':'red'}})
plt.show()
```

In [ ]:
```
plt.figure(figsize=(10,6))
sns.pairplot(
    credit_data,
#     x_vars=['PRC_FULL_PAYMENT', 'BALANCE_FREQUENCY', 'CASH_ADVANCE_FREQUENCY','CASH_ADVANCE_TRX'],
#     y_vars=['PRC_FULL_PAYMENT', 'BALANCE_FREQUENCY', 'CASH_ADVANCE_FREQUENCY','CASH_ADVANCE_TRX'],kind="reg", plot_
    x_vars=['CASH_ADVANCE_TRX','PRC_FULL_PAYMENT'],
    y_vars=['CASH_ADVANCE_FREQUENCY','BALANCE_FREQUENCY'],kind="reg", plot_kws={'line_kws':{'color':'red'}})
plt.show()
```

Use the elbow method to find a good KMeans clustering for our dataset.

You'll want to consider clusters in the range of 1 to 10

https://en.wikipedia.org/wiki/Elbow_method_(clustering) (https://en.wikipedia.org/wiki/Elbow_method_(clustering))

Optionally: You could consider if using a PCA before clustering has any effect on the number of clusters.

Display the elbow plot, by plotting the numbers of clusters against the models inertia

In [ ]:
```
from sklearn.cluster import KMeans

kmeans_models = [KMeans(n_clusters=k).fit(credit_data) for k in range (1, 10)]
innertia = [model.inertia_ for model in kmeans_models]

plt.plot(range(1, 10), innertia)
plt.title('Elbow method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```

In [ ]:
```
kmeans = KMeans(n_clusters=2, random_state=23)
kmeans.fit(credit_data)
```

Add the labels to our original dataset with a column called 'cluster_id'

In [ ]:
```
#add the cluster ids to the table
credit_data['cluster_id'] = kmeans.labels_
```

Use a scatter plot to show the ONEOFF_PURCHASES vs PURCHASES overall.

Using the cluster-ids to color the points.

Is there any pattern to how our clustering represents singular purchases vs. many purchases.

In [ ]:
```
plt.figure(figsize=(10,6))
fig=px.scatter(credit_data,x='ONEOFF_PURCHASES', y='PURCHASES', color='cluster_id',
               title='Distribution of clusters based on One off purchases and total purchases', trendline="ols")
fig.show()
```

Use a scaterplot to show the CREDIT_LIMIT vs. Purchases.

Use the cluser-id to color the points.

This will show the distribution of the clusters based on the credit limit and total purchases.

What does our plot tell us?

In [ ]:
```
plt.figure(figsize=(10,6))
fig=px.scatter(credit_data,x='CREDIT_LIMIT', y='PURCHASES', color='cluster_id',
               title='Distribution of clusters based on Credit limit and total purchases', trendline="ols")
fig.show()
```