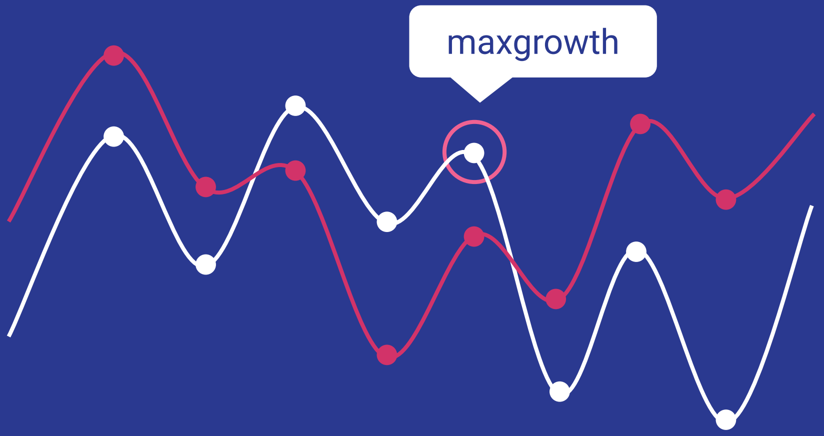
Place and Promotion Analytics

By Prof.Vishal Chugh



What is Place Analytics ?

Place Analytics is the process of analyzing where and how customers prefer to purchase a product or service. It looks at the effectiveness of distribution channels (supermarkets, online platforms, convenience stores, direct sales, etc.), customer satisfaction with these channels, and how location or accessibility influences buying decisions.

Importance of Place Analytics

NPD	Franchising	Warehouses
Identifying place where customers prefer to buy product	Identifying best location for Franchise outlet based on customer segmentation	Identifying best location for warehouses to store goods so that it takes less time to reach customers

Steps in Place Analytics

Survey / Feedback

Perform regular surveys or feedbacks from time to time to understand what customer wants. This will help business to stay ahead in the competition

Clustering

Using Clustering techniques to identify best clusters to sell the goods

PCA

Converting Multiple Dimensions into two dimensions

What is Promotion Analytics ?

Promotion Analytics evaluates the effectiveness of marketing campaigns and communication efforts in driving customer awareness, recall, engagement, and purchase decisions. It measures how well promotions (discounts, coupons, ads, influencer campaigns, etc.) influence consumer behavior.

Importance of Promotion Analytics

NPD

Identifying where the promotions are effective and accordingly make marketing campaigns for that specific segmented customers

Maximum ROMI

Get Maximum ROMI by targeting right customer segments at right place and at right time

Effective Budgets

Using the marketing budget effectively by ensuring promotions are done to right set of segmented customers

Steps in Promotion Analytics

Survey / Feedback

Perform regular surveys or feedbacks from time to time to understand what customer wants. This will help business to stay ahead in the competition

Clustering

Using Clustering techniques to identify best clusters where promotions are performing well

PCA

Converting Multiple Dimensions into two dimensions

What We Will Learn

“**Place Analytics** optimizes where to sell.

Promotion Analytics optimizes whom to communicate & influence purchase.”

Principal Component Analysis

"Principal Component Analysis (PCA) is a dimensionality reduction technique used in data analysis and machine learning.
It transforms a large set of variables into a smaller set of new variables (called principal components) that still capture most of the important information (variance) in the data."

Importance of PCA

1. Finds hidden patterns in high-dimensional data.

2. Removes redundancy (when features are correlated).

3. Creates new “axes” (principal components) that summarize the data.

How PCA works

Step 1: Standardize the data.

Step 2: Identify directions of maximum variance.

Step 3: Rotate the dataset to these directions (principal components).

Step 4: Keep the top components that explain most variance, drop the rest.

Why it is used

1. To simplify datasets without losing much information.

2. To visualize high-dimensional data in 2D or 3D plots.

3. To speed up clustering process

Two Types of Machine Learning Algorithm

Supervised

Supervised Learning is a type of machine learning where the model is trained using labeled data (i.e., input features + known correct answers).

The goal is to learn the relationship between inputs and outputs so the model can predict outcomes for new data.

Unsupervised

Unsupervised Learning is a type of machine learning where the model is trained on unlabeled data (i.e., only inputs, no correct answers).

The goal is to discover hidden patterns, structures, or groupings in the data.

Examples

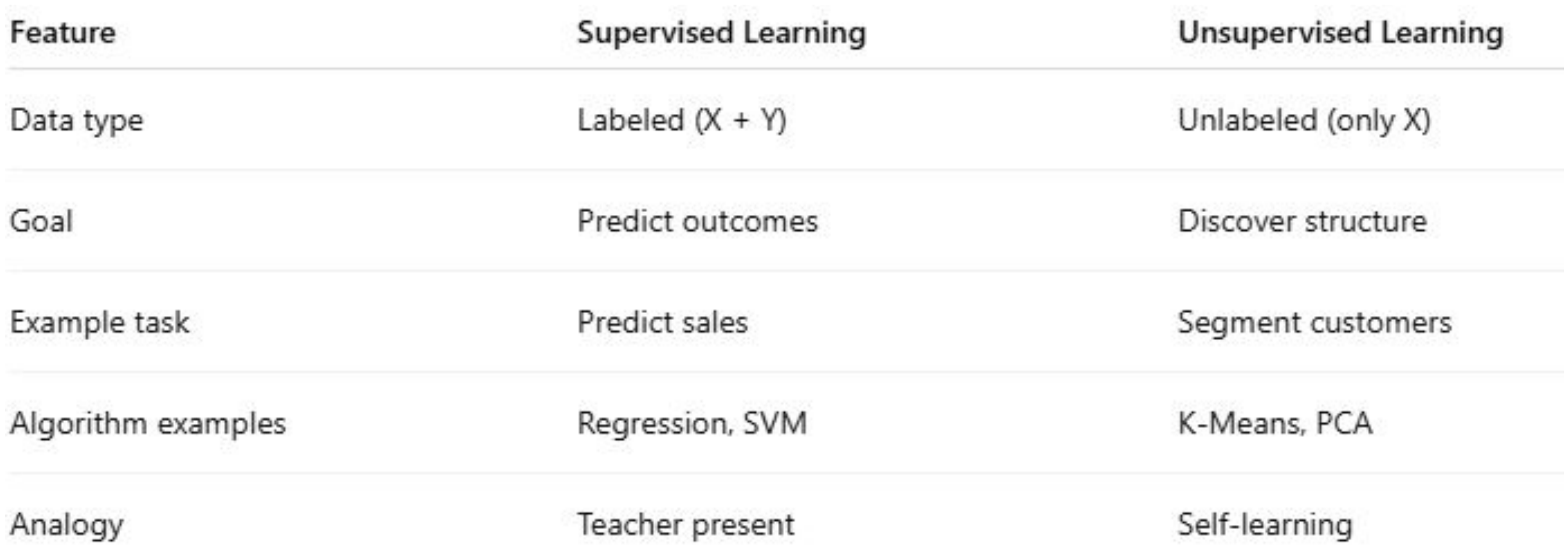
Supervised

1. Predicting sales revenue based on marketing spend (Regression).
2. Classifying emails as Spam or Not Spam.
3. Predicting whether a customer will churn or stay.
4. **Common algorithms:** Linear Regression, Logistic Regression, Decision Trees, Random Forests, Neural Networks.

Unsupervised

1. Customer segmentation (grouping customers by purchase behavior).
2. Market basket analysis (which products are bought together).
3. PCA for dimensionality reduction.
4. **Common algorithms**: K-Means Clustering, Hierarchical Clustering, PCA, Association Rules.

Comparison Table



Feature	Supervised Learning	Unsupervised Learning
Data type	Labeled (X + Y)	Unlabeled (only X)
Goal	Predict outcomes	Discover structure
Example task	Predict sales	Segment customers
Algorithm examples	Regression, SVM	K-Means, PCA
Analogy	Teacher present	Self-learning

What is K Means Clustering ?

K-Means is an unsupervised machine learning algorithm used to group data points into K clusters based on their similarity.

It assigns each data point to the nearest cluster center (called a centroid), and keeps updating centroids until the clusters are stable.

How K Means Clustering Works

1. Choose the cluster range.

2. Assign each data point to the nearest centroid.

3. Recalculate centroids (average of all points in the cluster).

4. Repeat until centroids stop moving.

Why K Means Clustering is Preferred

1. Simplicity & Easy to Understand
2. Fast & Scalable
3. Works Well in Practice
4. Flexibility
5. Integration with Preprocessing
6. Human-Friendly Results

K Means Clustering Usefulness

1. Helps identify natural customer segments (e.g., high spenders, discount lovers, loyal customers).

2. Simplifies large datasets into patterns.

3. Commonly used in marketing for segmentation, targeting, and positioning.

What is Silhouette Score ?

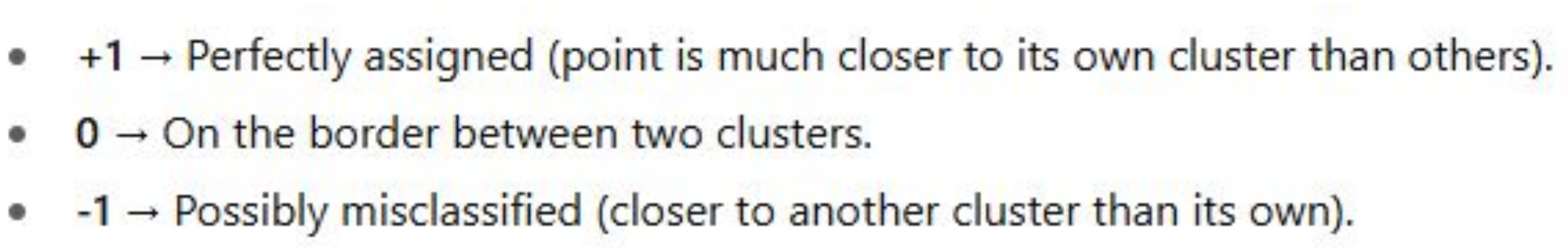
The Silhouette Score is a metric that measures how well data points fit within their assigned cluster compared to other clusters.

Silhouette Score

"It compares the average distance to points in the same cluster (cohesion) vs. the average distance to points in the nearest other cluster (separation).

Score ranges from -1 to +1."

Silhouette Score Interpretation



- **+1** → Perfectly assigned (point is much closer to its own cluster than others).
- **0** → On the border between two clusters.
- **-1** → Possibly misclassified (closer to another cluster than its own).

Why Silhouette Score is Useful

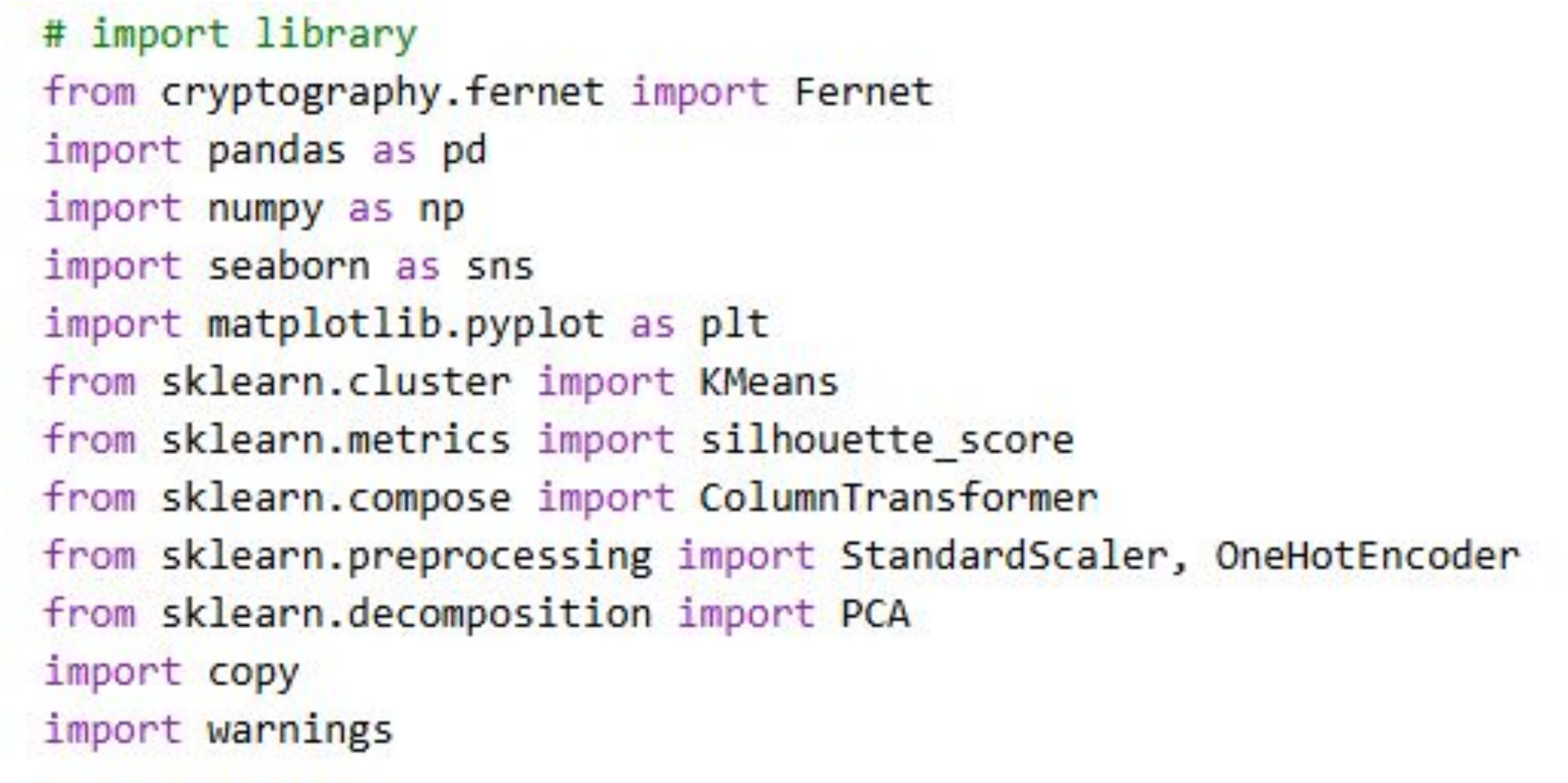
“Helps decide the optimal number of clusters (K).

Higher silhouette score = better-defined clusters.”

Explore the Data in Python- Place Analytics

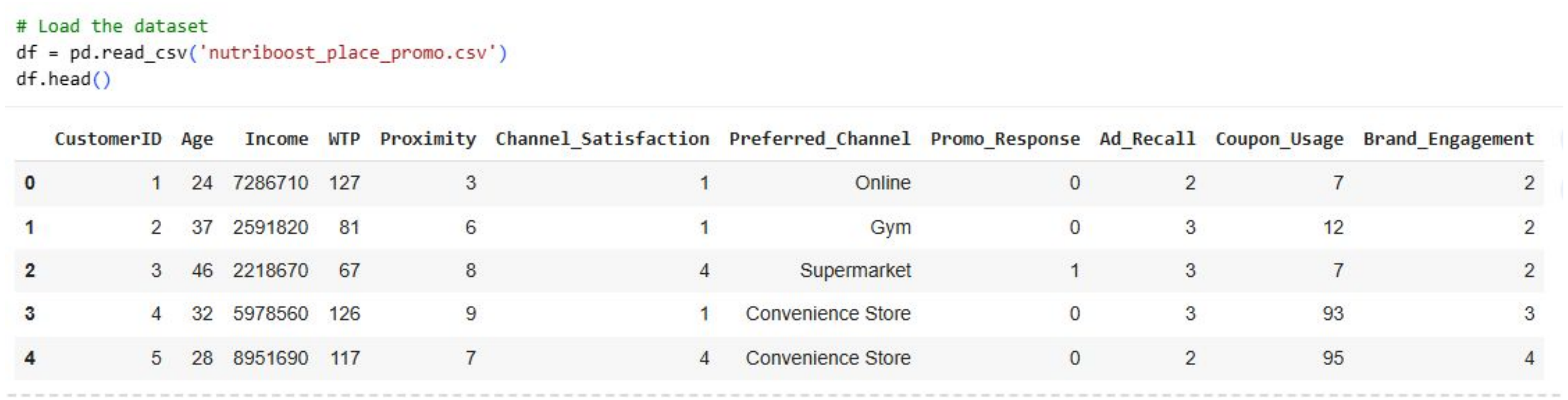
By Prof.Vishal Chugh

Import Libraries



```
# import library
from cryptography.fernet import Fernet
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.decomposition import PCA
import copy
import warnings
```

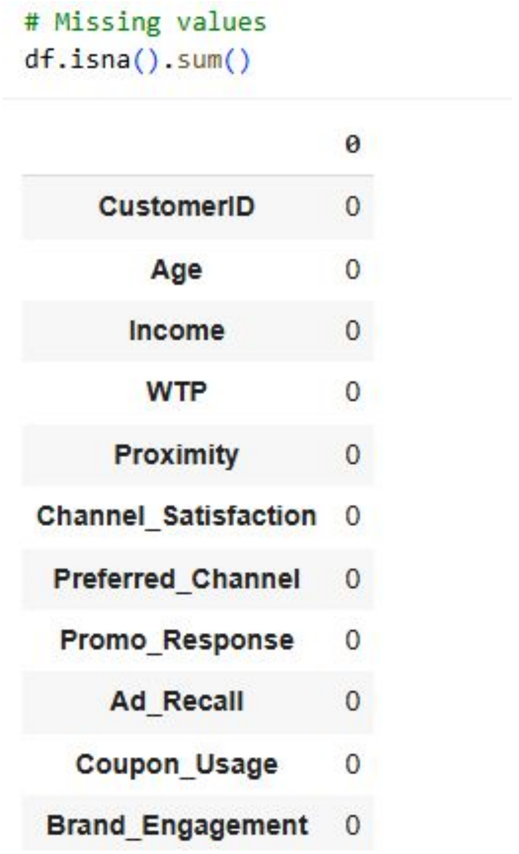
After Decrypting the Data



```python
# Load the dataset
df = pd.read_csv('nutriboost_place_promo.csv')
df.head()
```

	CustomerID	Age	Income	WTP	Proximity	Channel_Satisfaction	Preferred_Channel	Promo_Response	Ad_Recall	Coupon_Usage	Brand_Engagement
0	1	24	7286710	127	3	1	Online	0	2	7	2
1	2	37	2591820	81	6	1	Gym	0	3	12	2
2	3	46	2218670	67	8	4	Supermarket	1	3	7	2
3	4	32	5978560	126	9	1	Convenience Store	0	3	93	3
4	5	28	8951690	117	7	4	Convenience Store	0	2	95	4

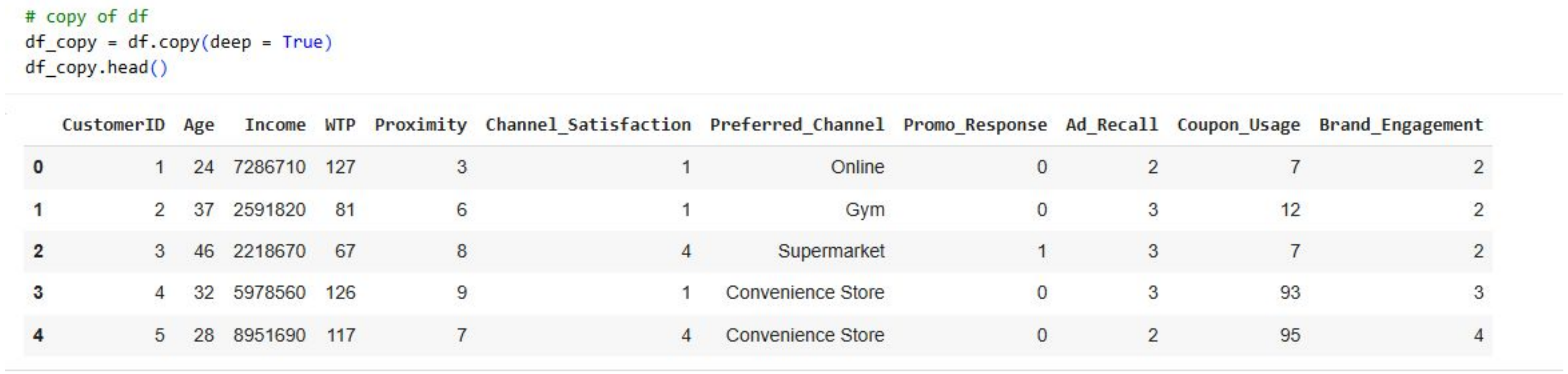
Check for the missing values



```python
# Missing values
df.isna().sum()
```

	0
CustomerID	0
Age	0
Income	0
WTP	0
Proximity	0
Channel_Satisfaction	0
Preferred_Channel	0
Promo_Response	0
Ad_Recall	0
Coupon_Usage	0
Brand_Engagement	0

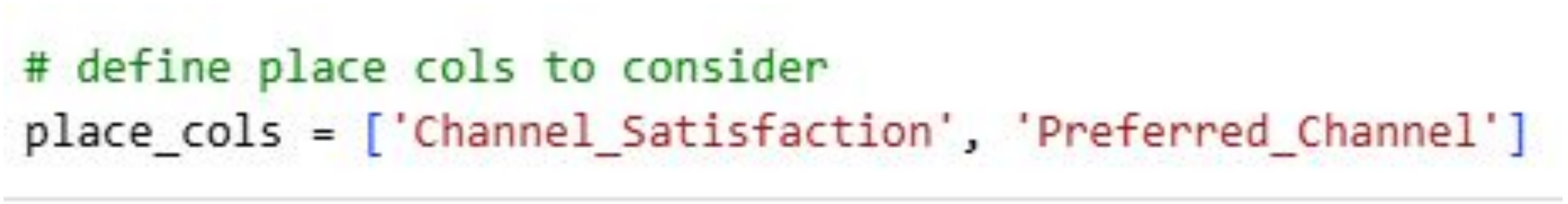
Copy of the Original Data Set



```python
# copy of df
df_copy = df.copy(deep = True)
df_copy.head()
```

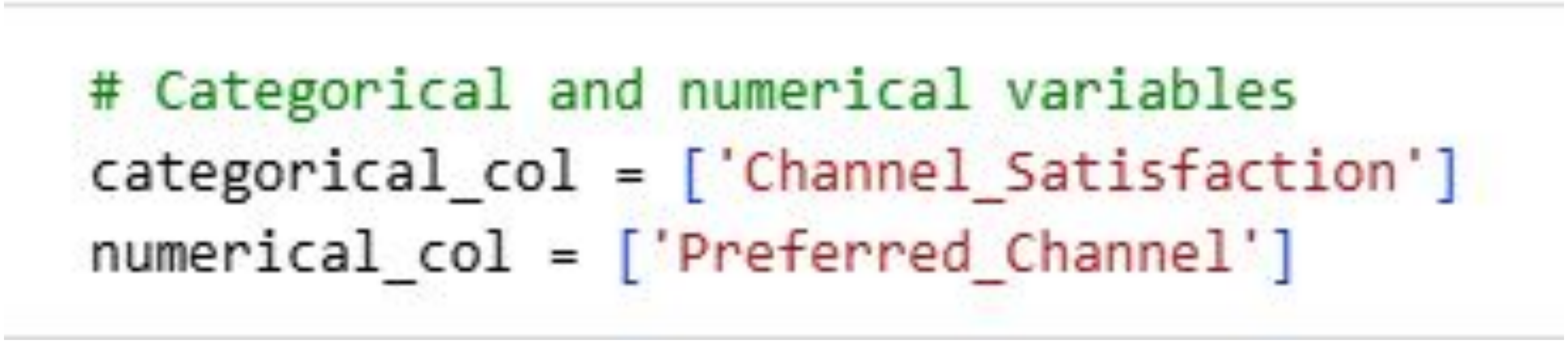
	CustomerID	Age	Income	WTP	Proximity	Channel_Satisfaction	Preferred_Channel	Promo_Response	Ad_Recall	Coupon_Usage	Brand_Engagement
0	1	24	7286710	127	3	1	Online	0	2	7	2
1	2	37	2591820	81	6	1	Gym	0	3	12	2
2	3	46	2218670	67	8	4	Supermarket	1	3	7	2
3	4	32	5978560	126	9	1	Convenience Store	0	3	93	3
4	5	28	8951690	117	7	4	Convenience Store	0	2	95	4

Place Columns To Consider

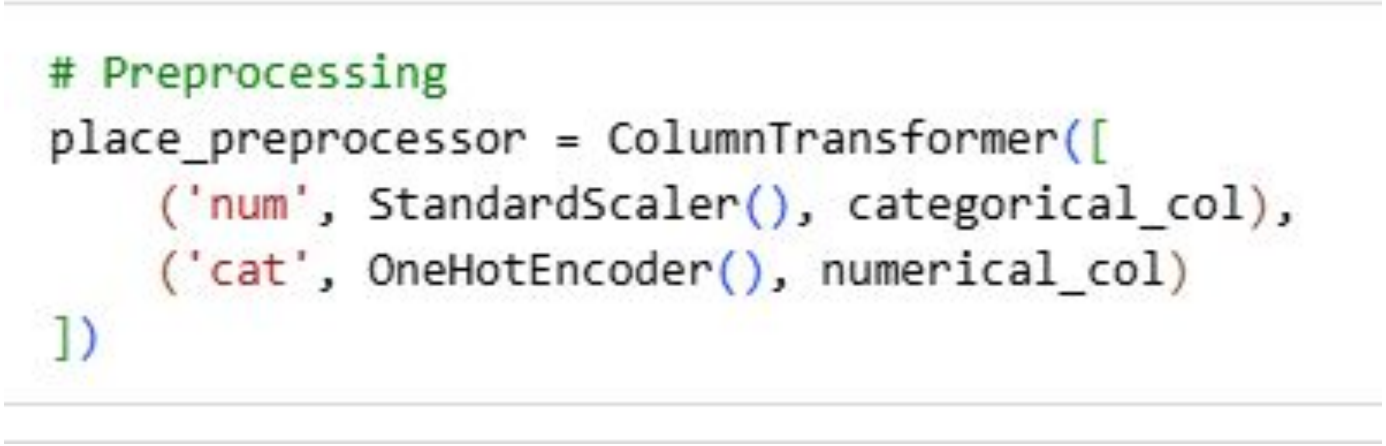


```python
# define place cols to consider
place_cols = ['Channel_Satisfaction', 'Preferred_Channel']
```

Separate Numerical and Categorical Cols

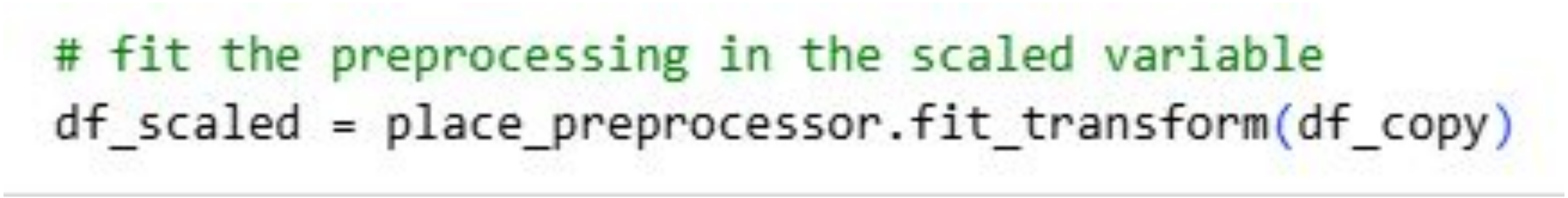

```
# Categorical and numerical variables
categorical_col = ['Channel_Satisfaction']
numerical_col = ['Preferred_Channel']
```

Preprocessing



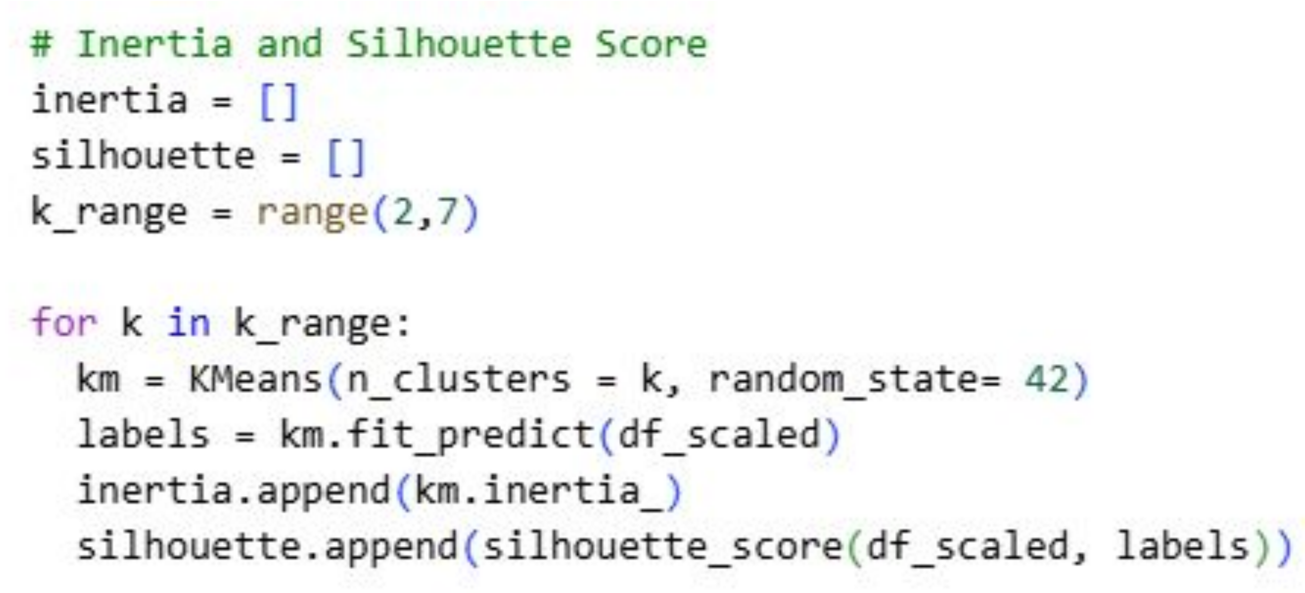
```
# Preprocessing
place_preprocessor = ColumnTransformer([
    ('num', StandardScaler(), categorical_col),
    ('cat', OneHotEncoder(), numerical_col)
])
```

Fit the preprocessed values



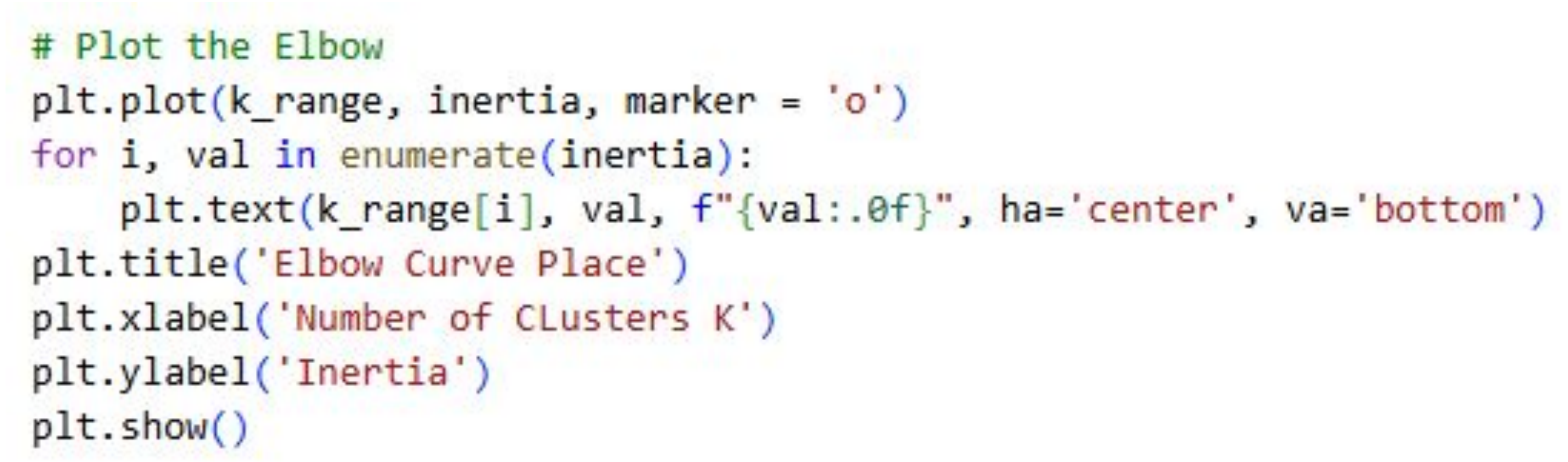
```
# fit the preprocessing in the scaled variable
df_scaled = place_preprocessor.fit_transform(df_copy)
```

Inertia and Silhouette Score



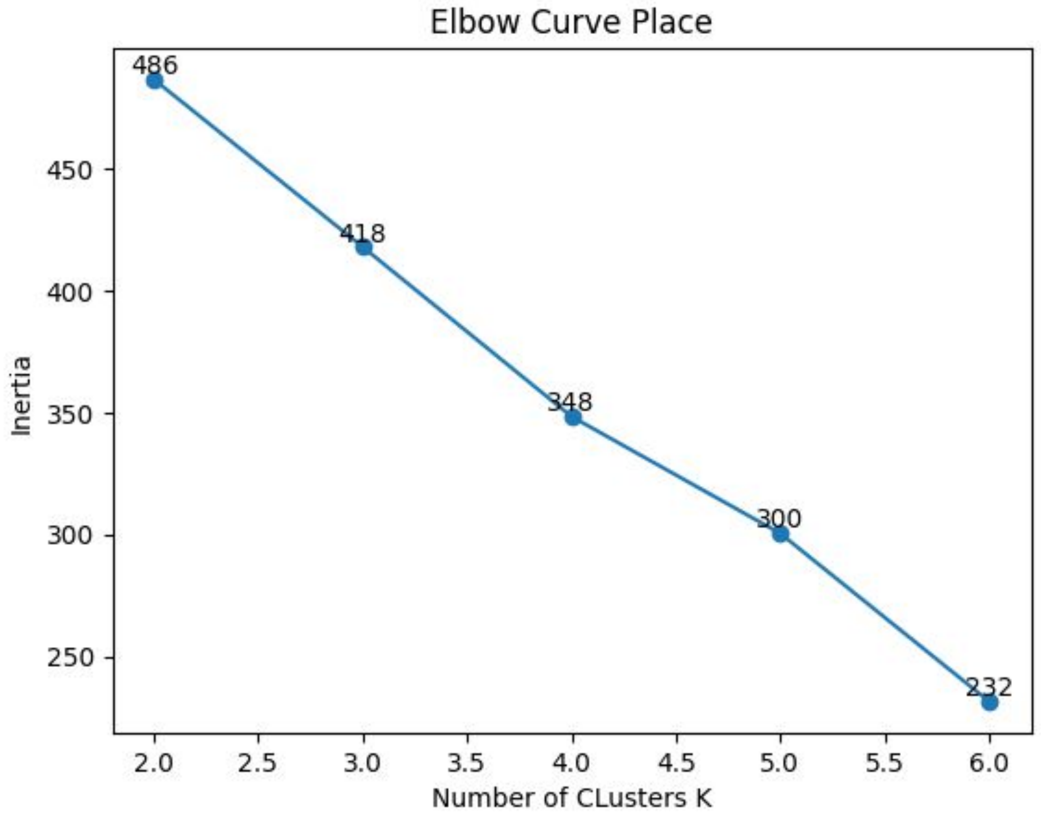
```
# Inertia and Silhouette Score
inertia = []
silhouette = []
k_range = range(2,7)

for k in k_range:
  km = KMeans(n_clusters = k, random_state= 42)
  labels = km.fit_predict(df_scaled)
  inertia.append(km.inertia_)
  silhouette.append(silhouette_score(df_scaled, labels))
```

Plot the Elbow Curve

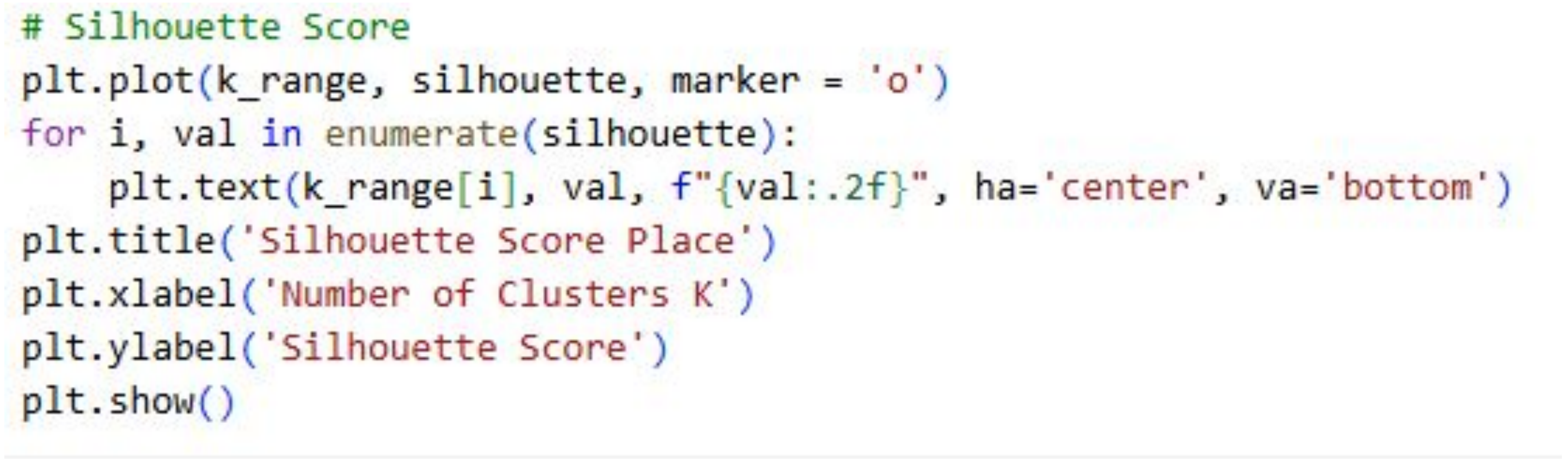


```python
# Plot the Elbow
plt.plot(k_range, inertia, marker = 'o')
for i, val in enumerate(inertia):
    plt.text(k_range[i], val, f"{val:.0f}", ha='center', va='bottom')
plt.title('Elbow Curve Place')
plt.xlabel('Number of CLusters K')
plt.ylabel('Inertia')
plt.show()
```

Elbow Curve

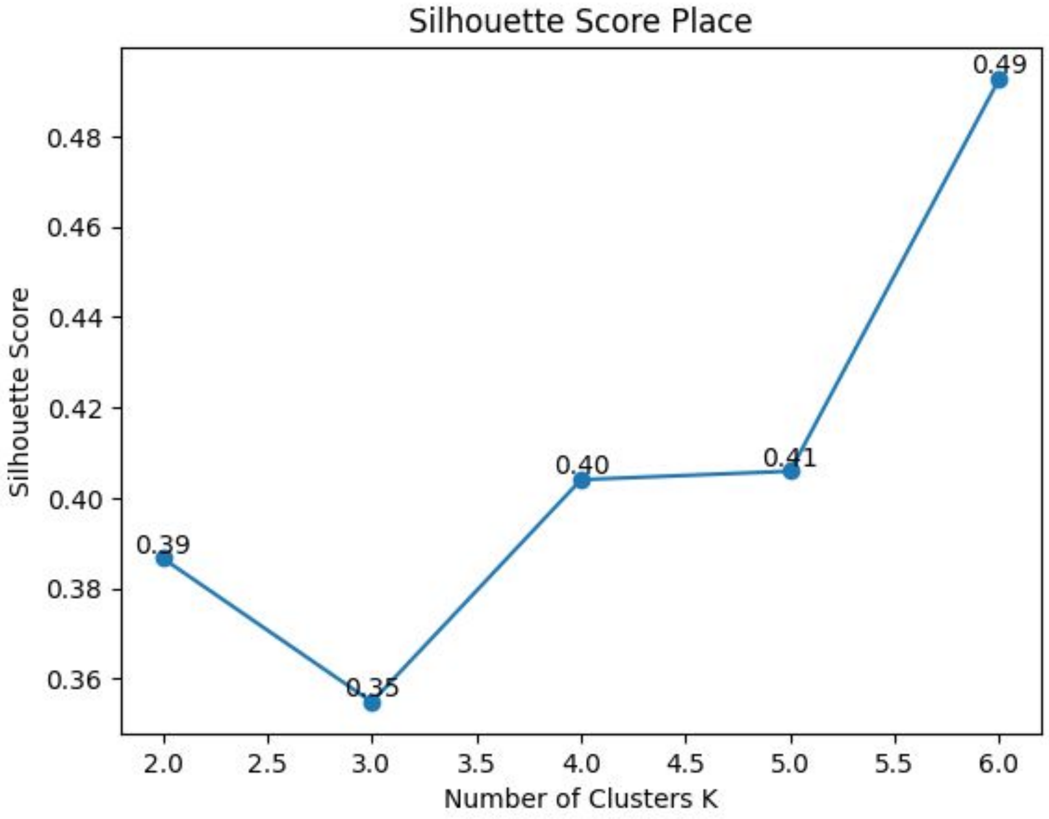

Elbow Curve Place
486
418
348
300
232
450
400
350
300
250
Inertia
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
Number of CLusters K

Silhouette Score

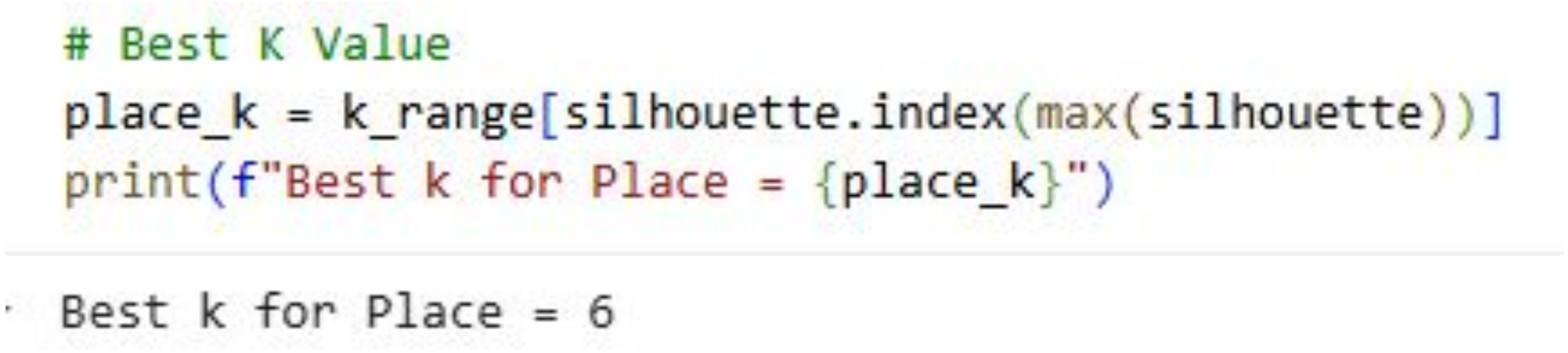


```python
# Silhouette Score
plt.plot(k_range, silhouette, marker = 'o')
for i, val in enumerate(silhouette):
    plt.text(k_range[i], val, f"{val:.2f}", ha='center', va='bottom')
plt.title('Silhouette Score Place')
plt.xlabel('Number of Clusters K')
plt.ylabel('Silhouette Score')
plt.show()
```

Silhouette Score Line Chart

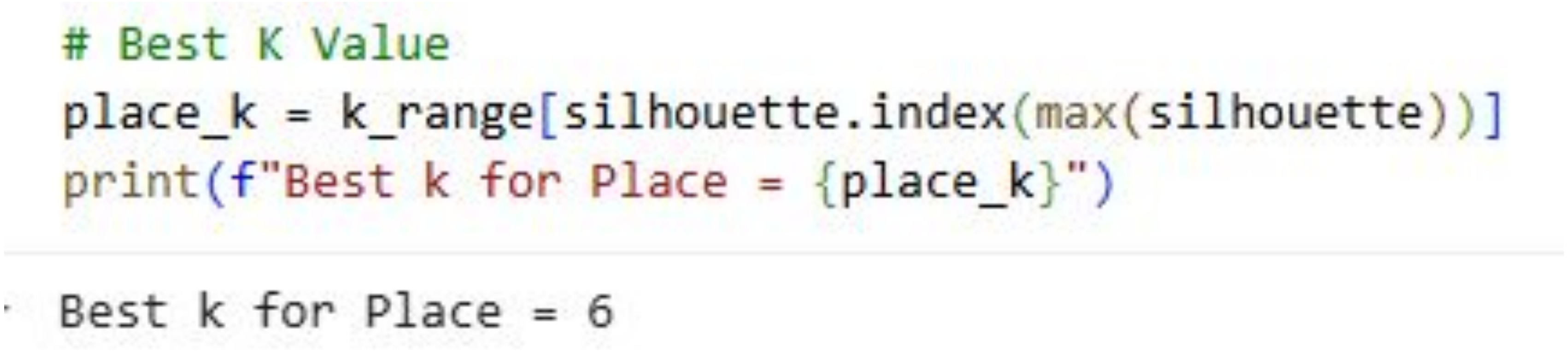


Best K Value


```
# Best K Value
place_k = k_range[silhouette.index(max(silhouette))]
print(f"Best k for Place = {place_k}")
```

```
Best k for Place = 6
```

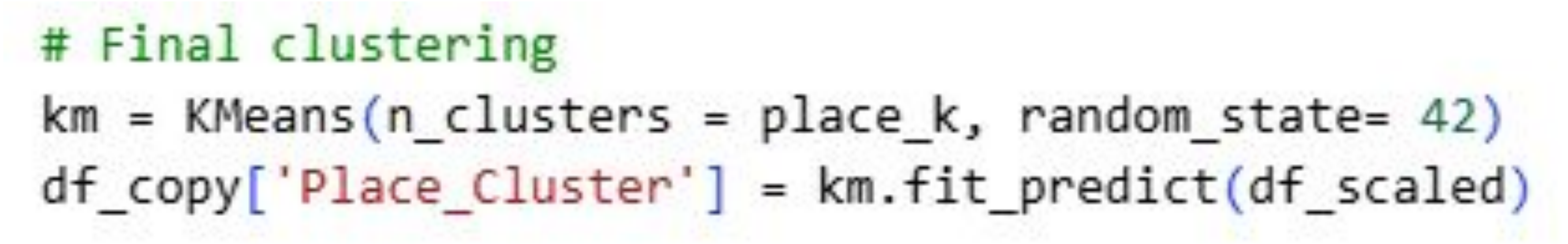
Best K Value



```
# Best K Value
place_k = k_range[silhouette.index(max(silhouette))]
print(f"Best k for Place = {place_k}")
```

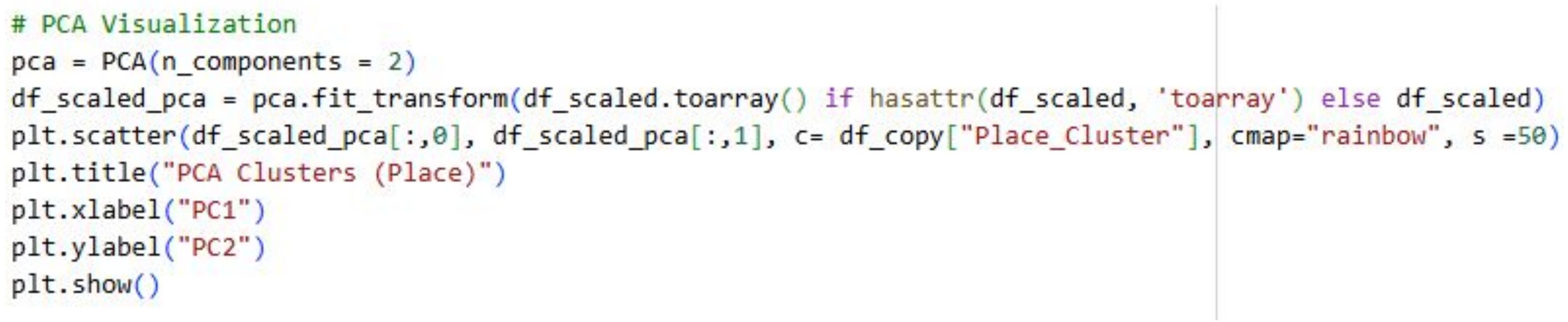
```
Best k for Place = 6
```

Final Clustering



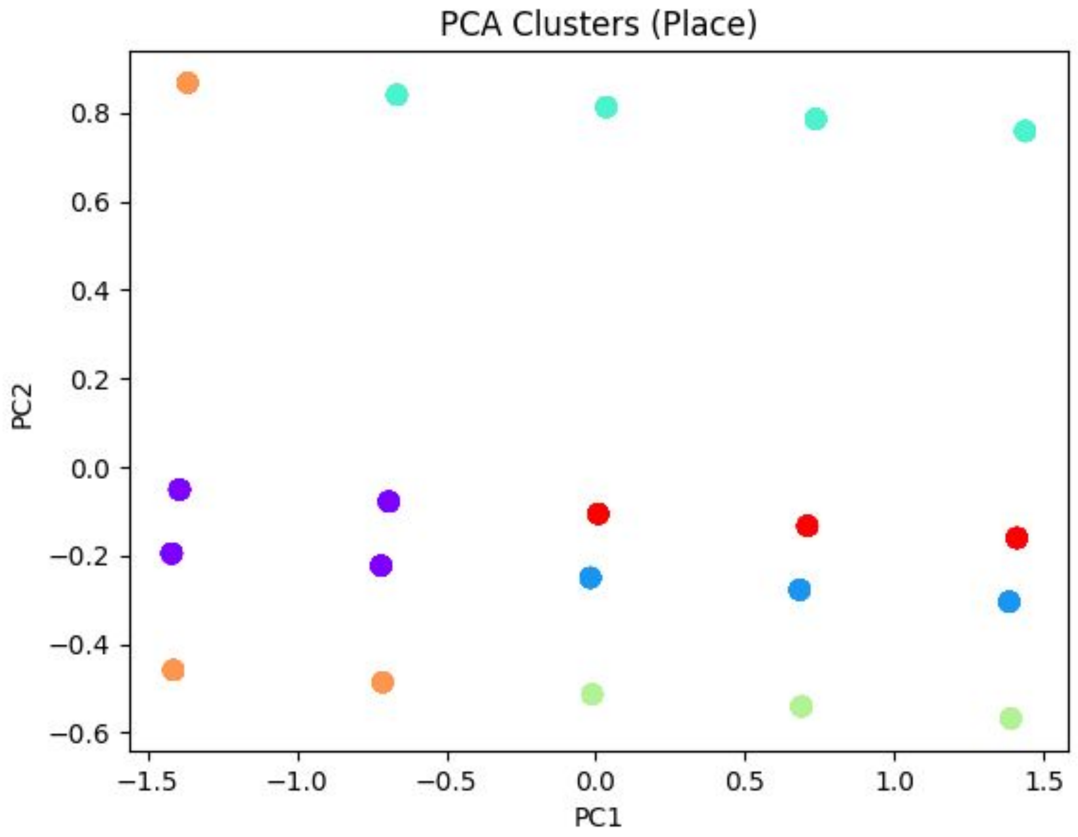
```
# Final clustering
km = KMeans(n_clusters = place_k, random_state= 42)
df_copy['Place_Cluster'] = km.fit_predict(df_scaled)
```

PCA Visualization

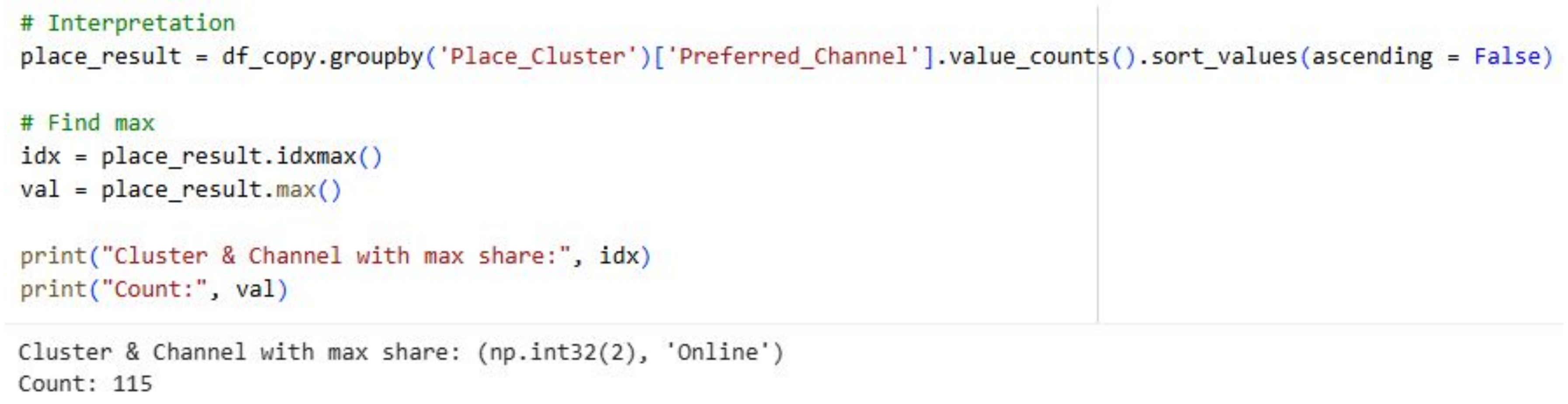


```
# PCA Visualization
pca = PCA(n_components = 2)
df_scaled_pca = pca.fit_transform(df_scaled.toarray() if hasattr(df_scaled, 'toarray') else df_scaled)
plt.scatter(df_scaled_pca[:,0], df_scaled_pca[:,1], c= df_copy["Place_Cluster"], cmap="rainbow", s =50)
plt.title("PCA Clusters (Place)")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.show()
```

PCA Visualization



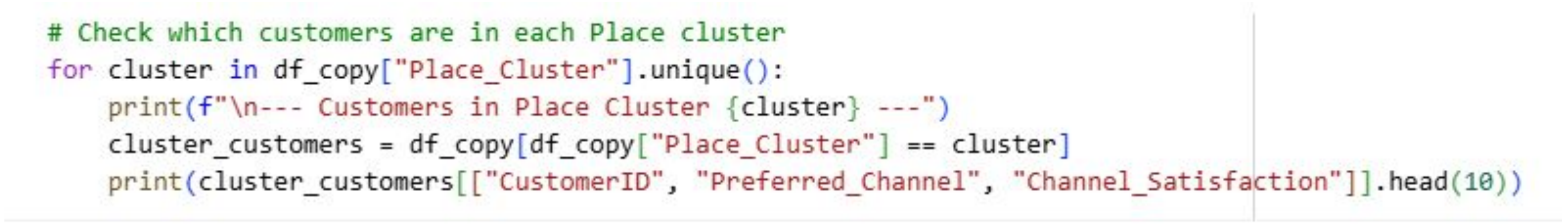
Interpretation



```
# Interpretation
place_result = df_copy.groupby('Place_Cluster')['Preferred_Channel'].value_counts().sort_values(ascending = False)

# Find max
idx = place_result.idxmax()
val = place_result.max()

print("Cluster & Channel with max share:", idx)
print("Count:", val)
```

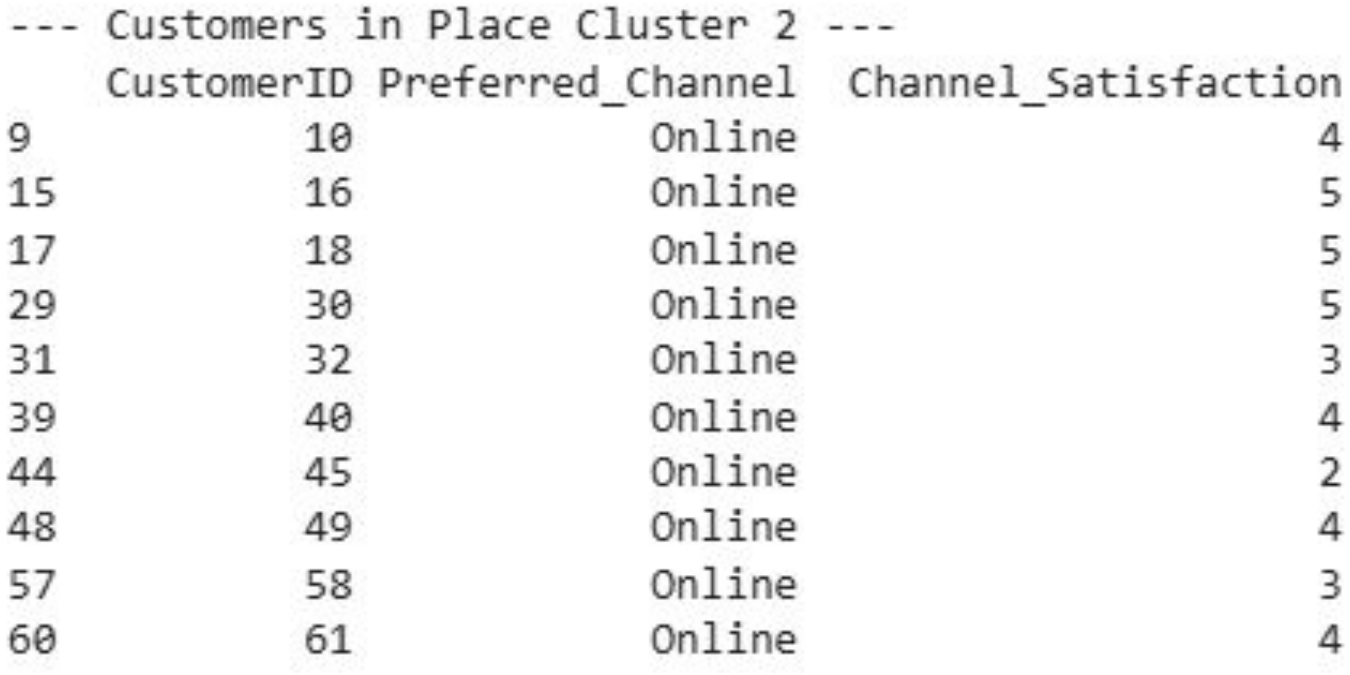
```
Cluster & Channel with max share: (np.int32(2), 'Online')
Count: 115
```

Identify Customers in Clusters



```
# Check which customers are in each Place cluster
for cluster in df_copy["Place_Cluster"].unique():
    print(f"\n--- Customers in Place Cluster {cluster} ---")
    cluster_customers = df_copy[df_copy["Place_Cluster"] == cluster]
    print(cluster_customers[["CustomerID", "Preferred_Channel", "Channel_Satisfaction"]].head(10))
```

You Get Different Customers in Each Clusters

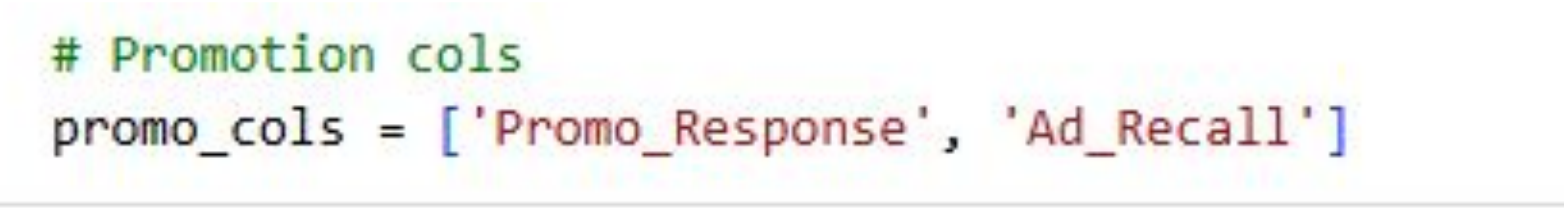


```
--- Customers in Place Cluster 2 ---
    CustomerID Preferred_Channel  Channel_Satisfaction
9           10            Online                     4
15          16            Online                     5
17          18            Online                     5
29          30            Online                     5
31          32            Online                     3
39          40            Online                     4
44          45            Online                     2
48          49            Online                     4
57          58            Online                     3
60          61            Online                     4
```

Explore the Data in Python - Promotion Analytics

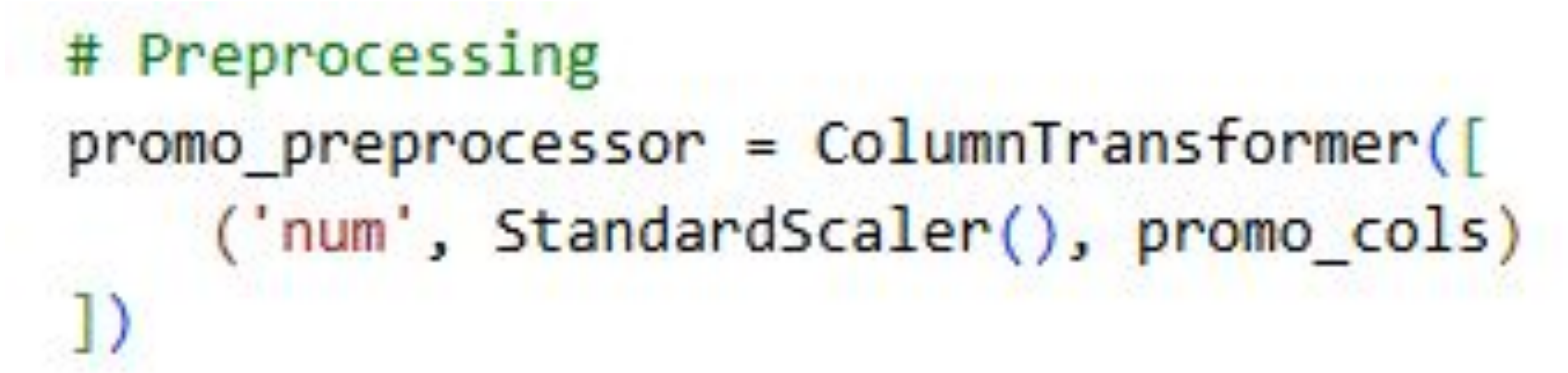
By Prof.Vishal Chugh

Promotion Columns To Consider



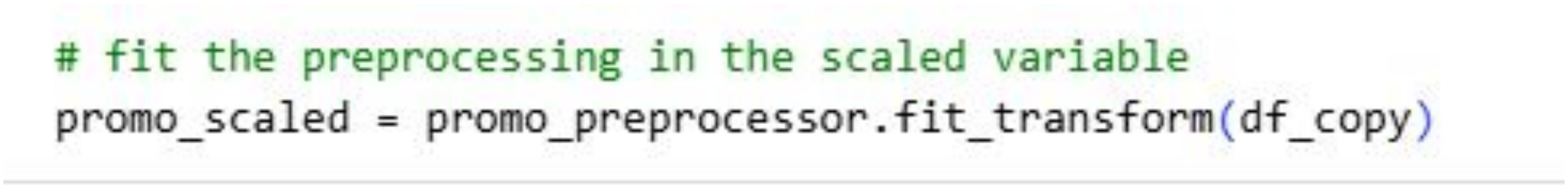
```
# Promotion cols
promo_cols = ['Promo_Response', 'Ad_Recall']
```

Preprocessing



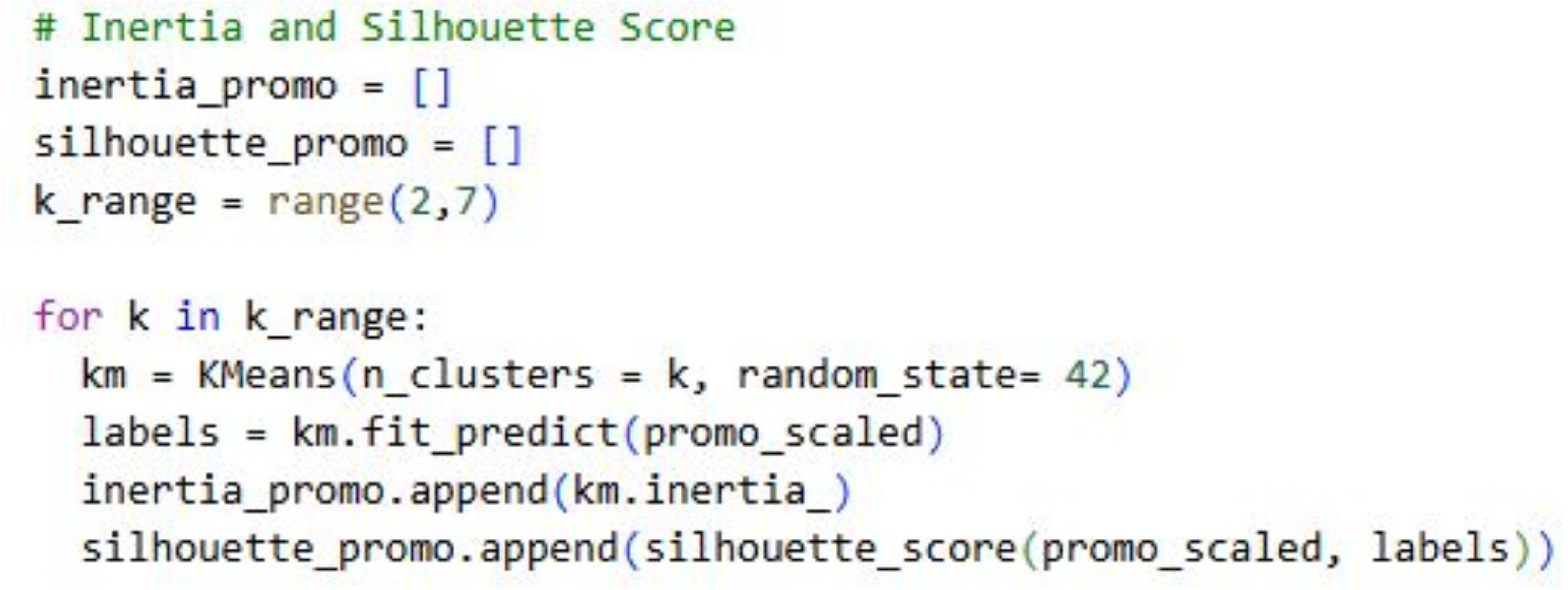
```
# Preprocessing
promo_preprocessor = ColumnTransformer([
    ('num', StandardScaler(), promo_cols)
])
```

Fit the preprocessed values



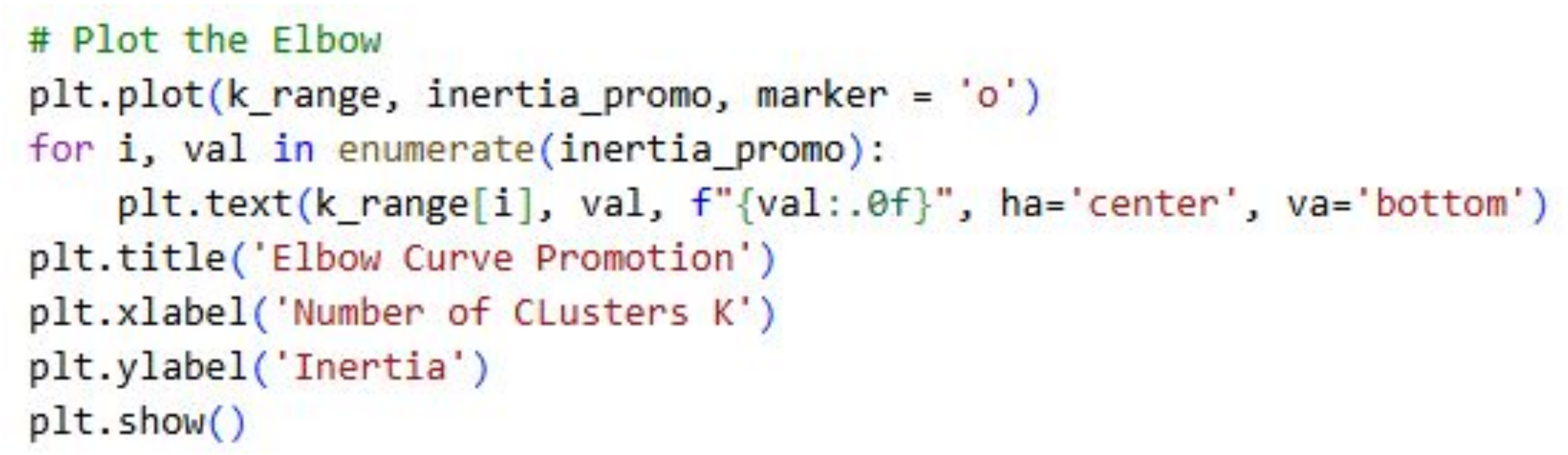
```
# fit the preprocessing in the scaled variable
promo_scaled = promo_preprocessor.fit_transform(df_copy)
```

Inertia and Silhouette Score



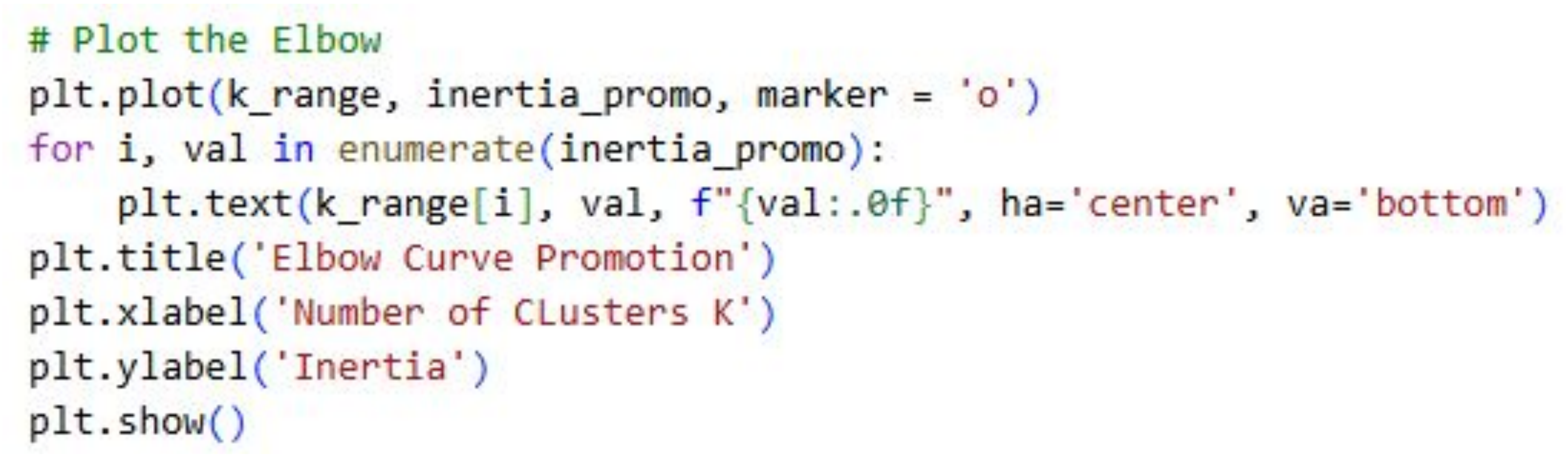
```python
# Inertia and Silhouette Score
inertia_promo = []
silhouette_promo = []
k_range = range(2,7)

for k in k_range:
  km = KMeans(n_clusters = k, random_state= 42)
  labels = km.fit_predict(promo_scaled)
  inertia_promo.append(km.inertia_)
  silhouette_promo.append(silhouette_score(promo_scaled, labels))
```

Plot the Elbow Curve



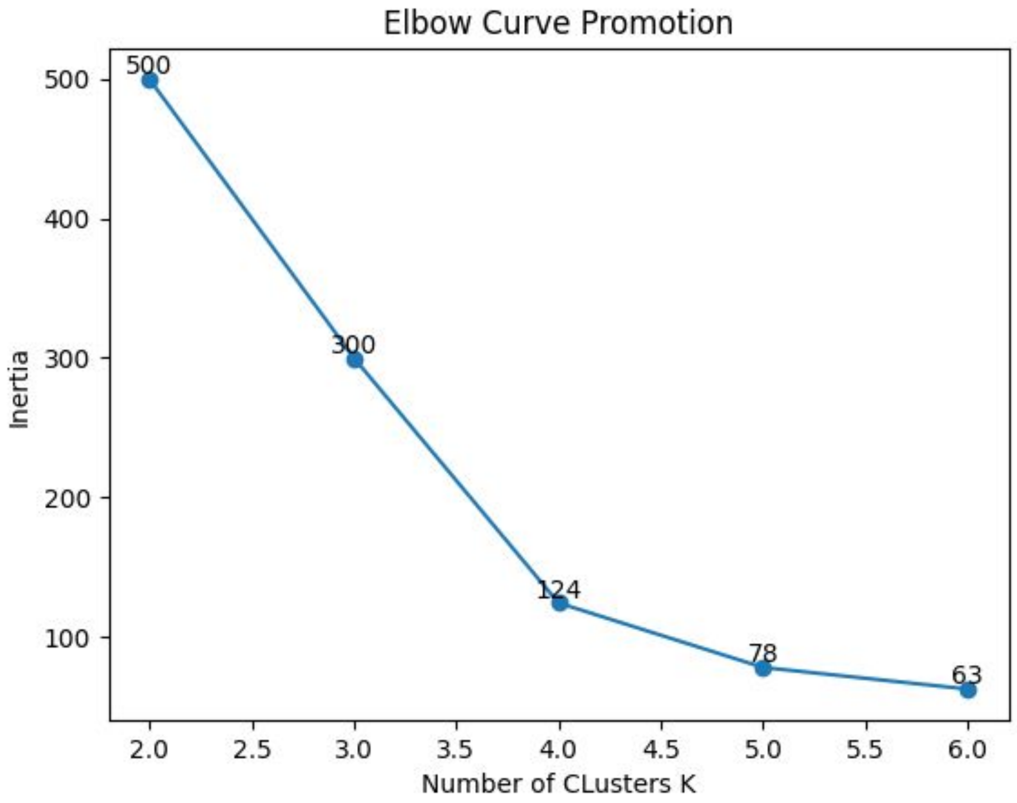
```
# Plot the Elbow
plt.plot(k_range, inertia_promo, marker = 'o')
for i, val in enumerate(inertia_promo):
    plt.text(k_range[i], val, f"{val:.0f}", ha='center', va='bottom')
plt.title('Elbow Curve Promotion')
plt.xlabel('Number of CLusters K')
plt.ylabel('Inertia')
plt.show()
```

Plot the Elbow Curve



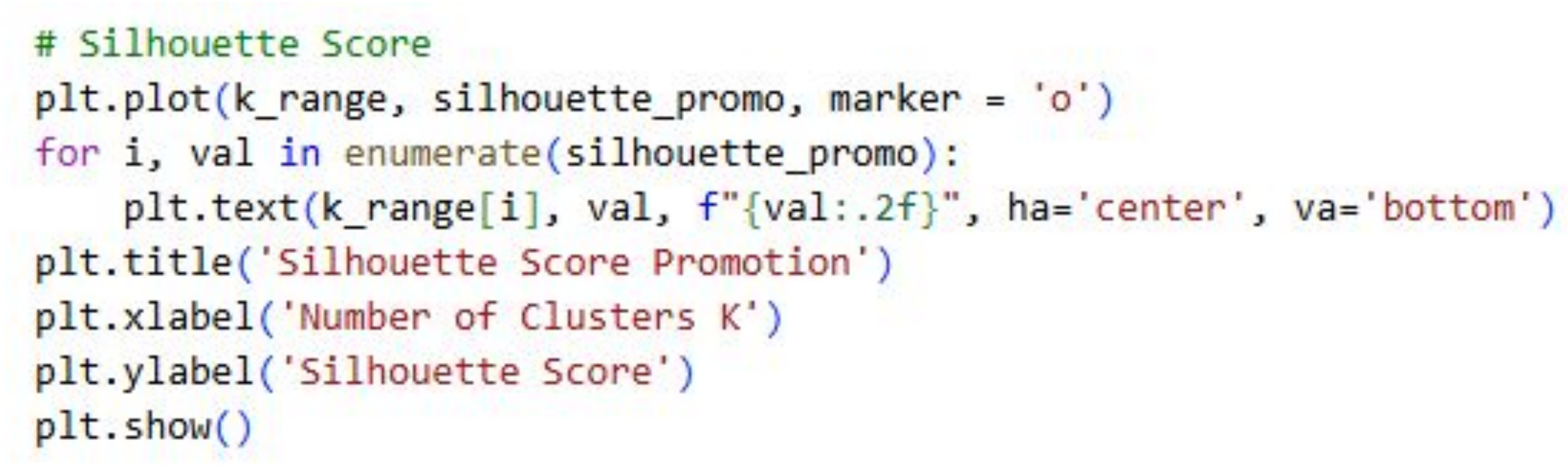
```python
# Plot the Elbow
plt.plot(k_range, inertia_promo, marker = 'o')
for i, val in enumerate(inertia_promo):
    plt.text(k_range[i], val, f"{val:.0f}", ha='center', va='bottom')
plt.title('Elbow Curve Promotion')
plt.xlabel('Number of CLusters K')
plt.ylabel('Inertia')
plt.show()
```

Elbow Curve



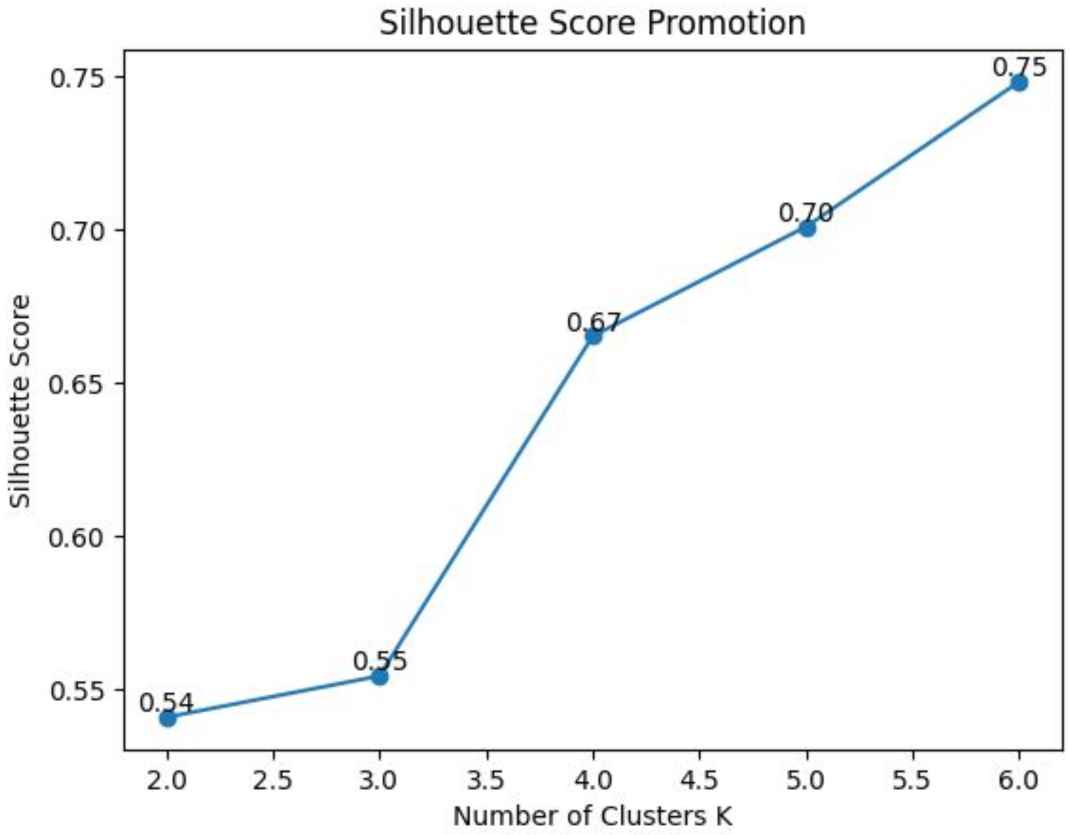
Elbow Curve Promotion
500
300
124
78
63
Inertia
500
400
300
200
100
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
Number of CLusters K

Silhouette Score

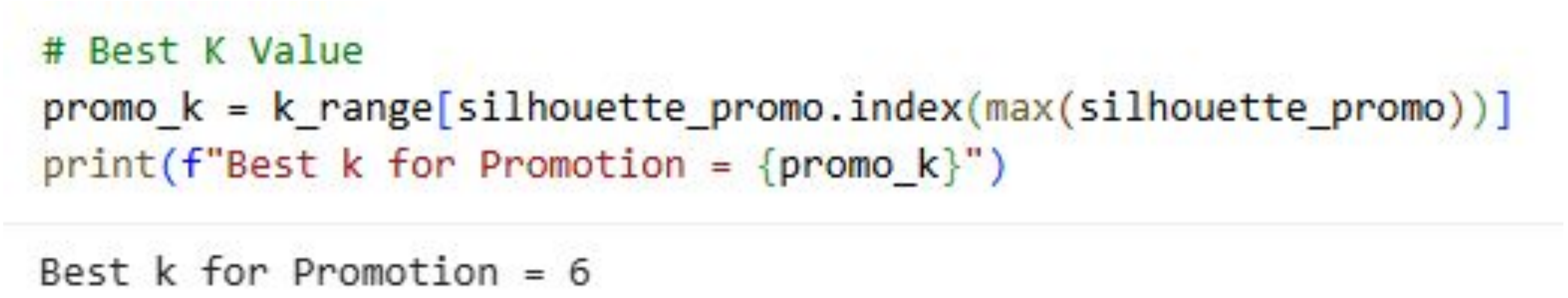


```python
# Silhouette Score
plt.plot(k_range, silhouette_promo, marker = 'o')
for i, val in enumerate(silhouette_promo):
    plt.text(k_range[i], val, f"{val:.2f}", ha='center', va='bottom')
plt.title('Silhouette Score Promotion')
plt.xlabel('Number of Clusters K')
plt.ylabel('Silhouette Score')
plt.show()
```

Silhouette Score Line Chart



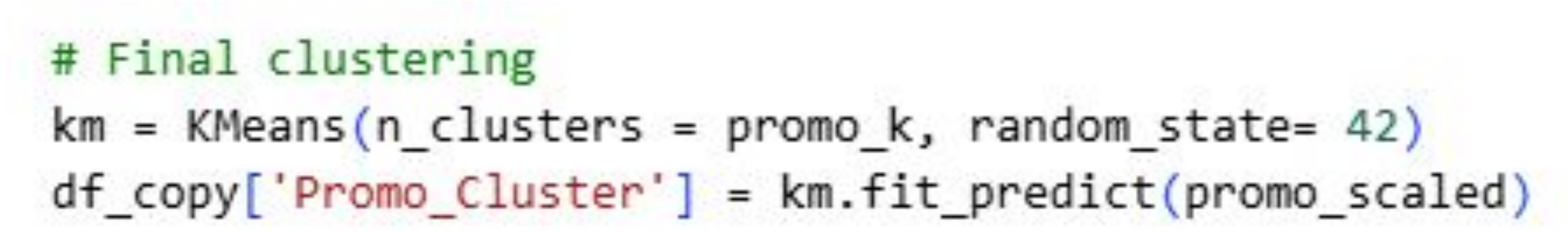
Best K Value



```
# Best K Value
promo_k = k_range[silhouette_promo.index(max(silhouette_promo))]
print(f"Best k for Promotion = {promo_k}")
```

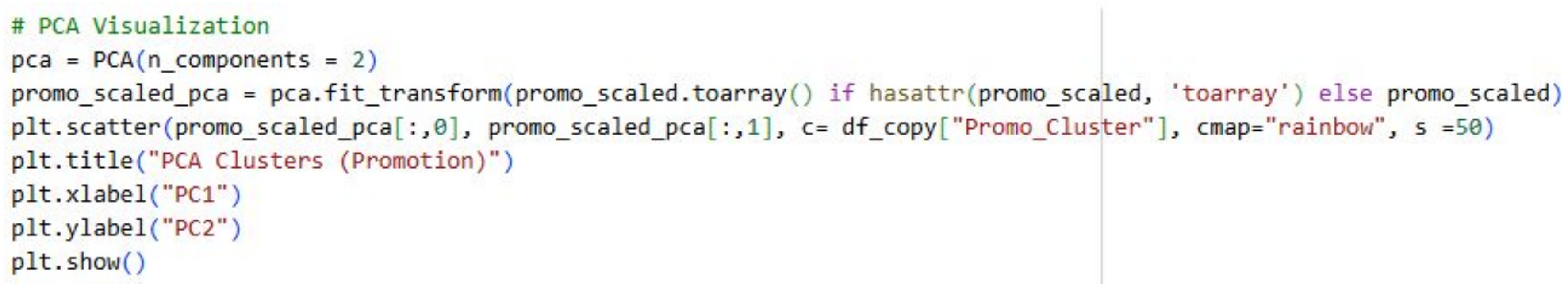
```
Best k for Promotion = 6
```

Final Clustering



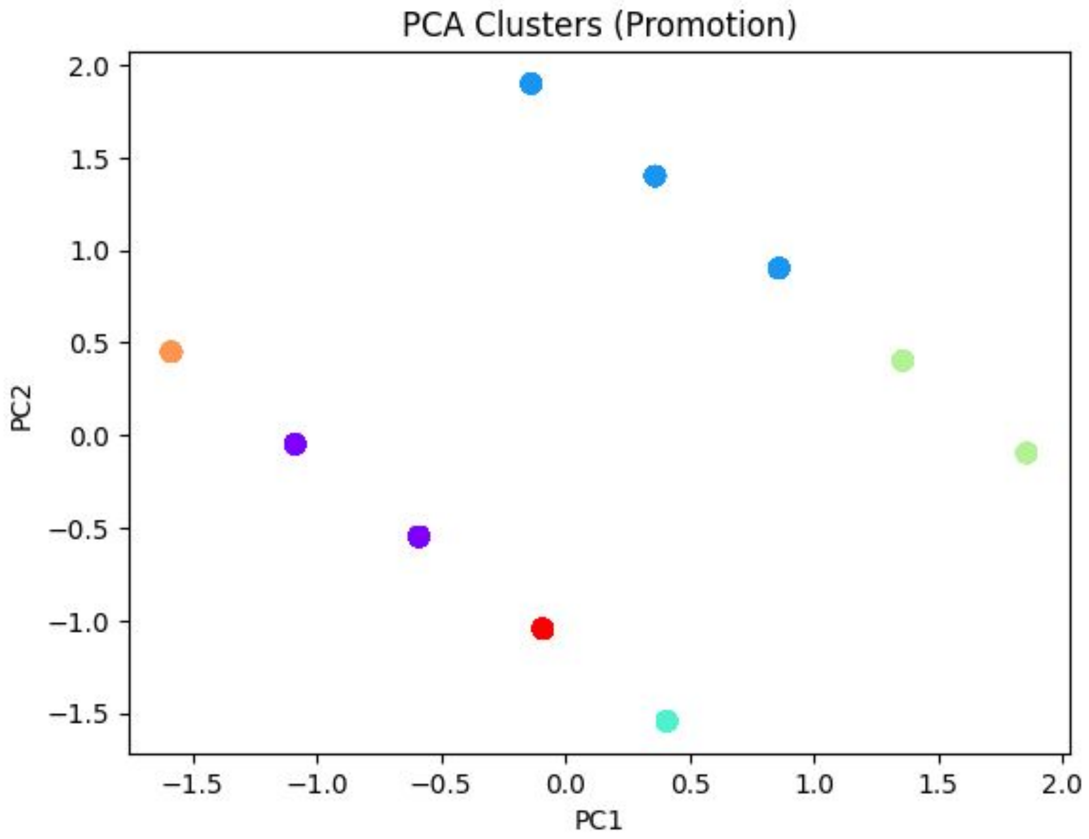
```
# Final clustering
km = KMeans(n_clusters = promo_k, random_state= 42)
df_copy['Promo_Cluster'] = km.fit_predict(promo_scaled)
```

PCA Visualization

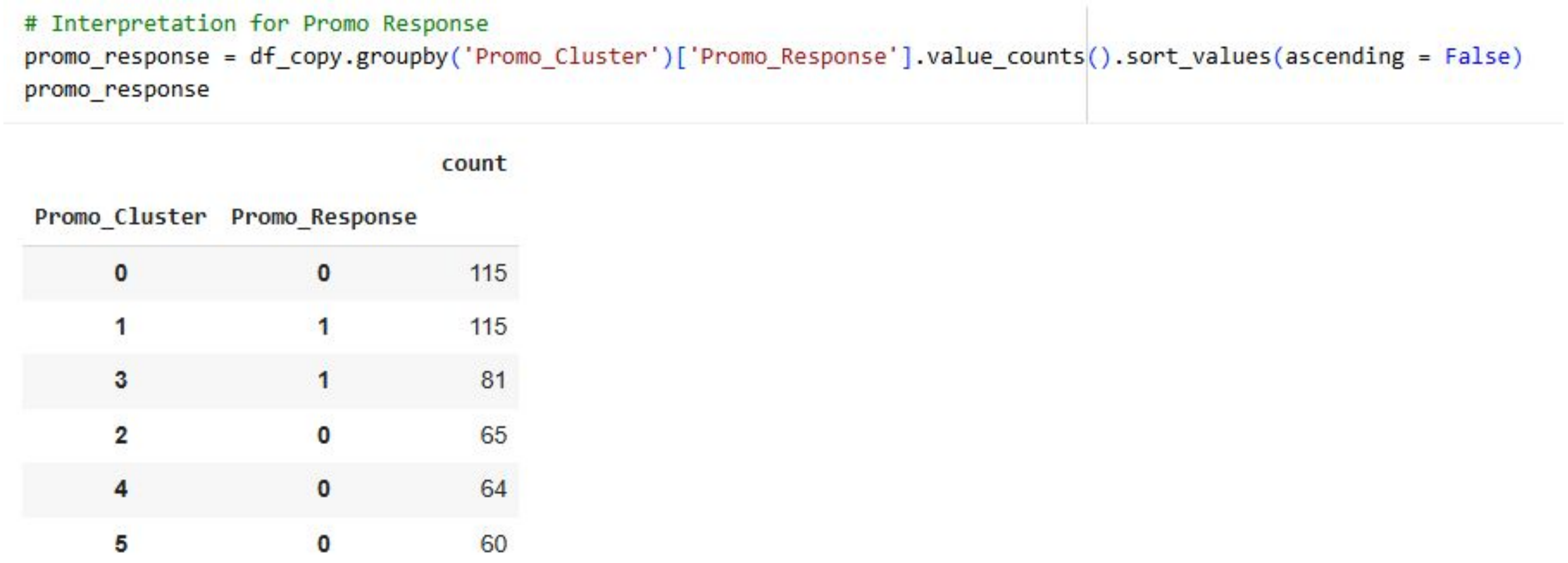


```
# PCA Visualization
pca = PCA(n_components = 2)
promo_scaled_pca = pca.fit_transform(promo_scaled.toarray() if hasattr(promo_scaled, 'toarray') else promo_scaled)
plt.scatter(promo_scaled_pca[:,0], promo_scaled_pca[:,1], c= df_copy["Promo_Cluster"], cmap="rainbow", s =50)
plt.title("PCA Clusters (Promotion)")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.show()
```

PCA Visualization



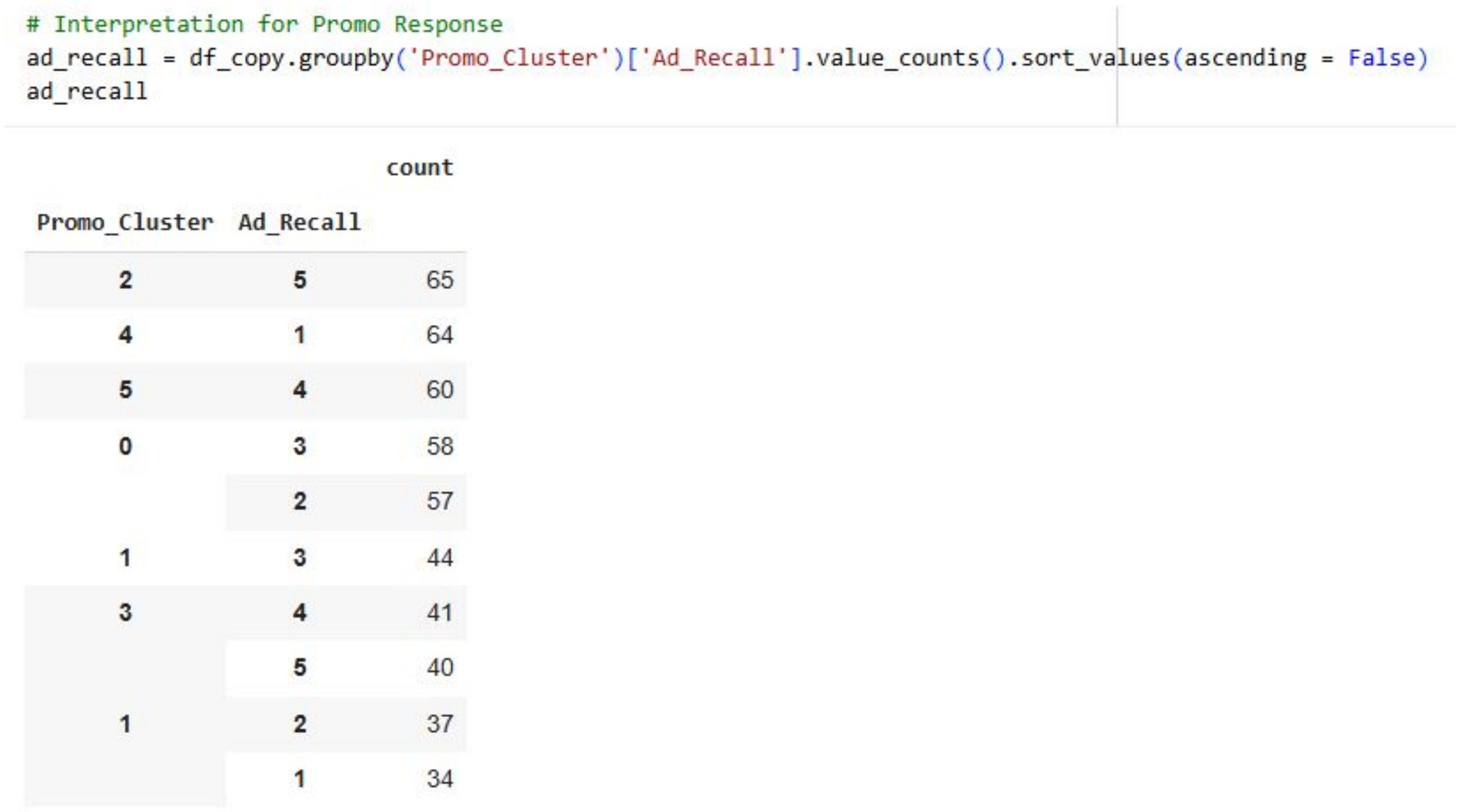
Interpretation for Promo Response



```
# Interpretation for Promo Response
promo_response = df_copy.groupby('Promo_Cluster')['Promo_Response'].value_counts().sort_values(ascending = False)
promo_response
```

Promo_Cluster	Promo_Response	count
0	0	115
1	1	115
3	1	81
2	0	65
4	0	64
5	0	60

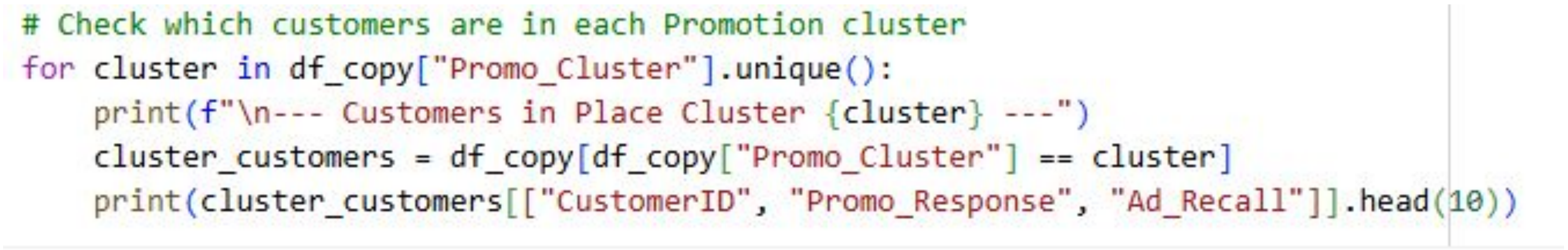
Interpretation for Ad Recall



```python
# Interpretation for Promo Response
ad_recall = df_copy.groupby('Promo_Cluster')['Ad_Recall'].value_counts().sort_values(ascending = False)
ad_recall
```

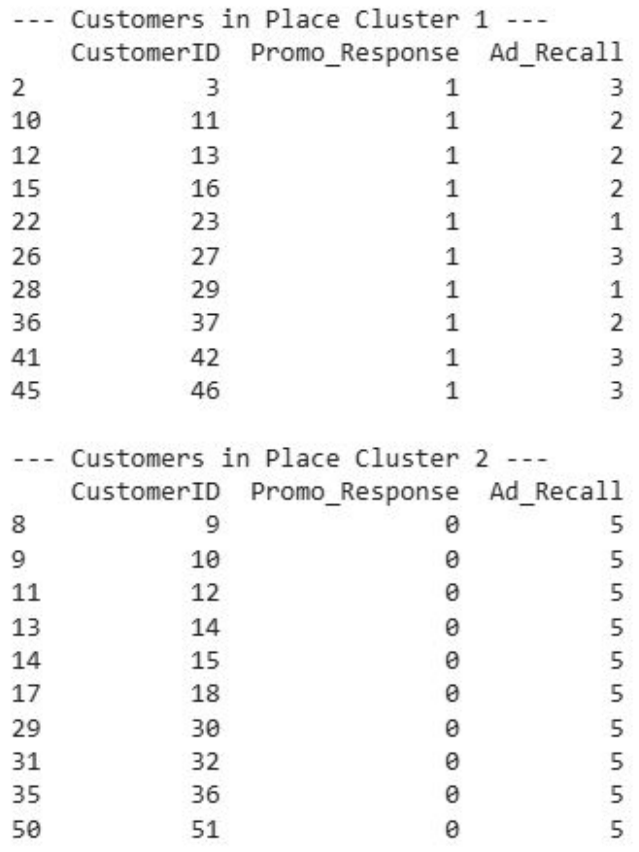
```
                            count
Promo_Cluster  Ad_Recall
      2            5           65
      4            1           64
      5            4           60
      0            3           58
                   2           57
      1            3           44
      3            4           41
                   5           40
      1            2           37
                   1           34
```

Identify Customers in Clusters



```
# Check which customers are in each Promotion cluster
for cluster in df_copy["Promo_Cluster"].unique():
    print(f"\n--- Customers in Place Cluster {cluster} ---")
    cluster_customers = df_copy[df_copy["Promo_Cluster"] == cluster]
    print(cluster_customers[["CustomerID", "Promo_Response", "Ad_Recall"]].head(10))
```

You Get Different Customers in Each Clusters



```
--- Customers in Place Cluster 1 ---
    CustomerID  Promo_Response  Ad_Recall
2            3               1          3
10          11               1          2
12          13               1          2
15          16               1          2
22          23               1          1
26          27               1          3
28          29               1          1
36          37               1          2
41          42               1          3
45          46               1          3

--- Customers in Place Cluster 2 ---
    CustomerID  Promo_Response  Ad_Recall
8            9               0          5
9           10               0          5
11          12               0          5
13          14               0          5
14          15               0          5
17          18               0          5
29          30               0          5
31          32               0          5
35          36               0          5
50          51               0          5
```