Small Business Owners in the United States

In this project, I am going to focus on business owners in the United States. You'll start by examining some demographic characteristics of the group, such as age, income category, and debt vs home value. Then you'll select high-variance features, and create a clustering model to divide small business owners into subgroups. Finally, you'll create some visualizations to highlight the differences between these subgroups. Good luck!

```
import pandas as pd
import plotly.express as px
import wqet_grader
from dash import Input, Output, dcc, html
from IPython.display import VimeoVideo
from jupyter_dash import JupyterDash
from scipy.stats.mstats import trimmed_var
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
```

Prepare Data

Import

Task 6.5.1: Read the file "data/SCFP2019.csv.gz" into the DataFrame df.

[3]:
```
df = pd.read_csv("data/SCFP2019.csv.gz")
print("df shape:", df.shape)
df.head()
```

```
df shape: (28885, 351)
```

[3]:

	YY1	Y1	WGT	HHSEX	AGE	AGECL	EDUC	EDCL	MARRIED	KIDS	...	NWCAT	INCCAT	ASSETCAT	NINCCAT	NINC2CAT	NWPCTLECAT	INCPCT
0	1	11	6119.779308	2	75	6	12	4	2	0	...	5	3	6	3	2	10	
1	1	12	4712.374912	2	75	6	12	4	2	0	...	5	3	6	3	1	10	
2	1	13	5145.224455	2	75	6	12	4	2	0	...	5	3	6	3	1	10	
3	1	14	5297.663412	2	75	6	12	4	2	0	...	5	2	6	2	1	10	
4	1	15	4761.812371	2	75	6	12	4	2	0	...	5	3	6	3	1	10	

Explore

Task 6.5.2: Calculate the proportion of respondents in df that are business owners, and assign the result to the variable prop_biz_owners. You'll need to review the documentation regarding the "HBUS" column to complete these tasks.

[ ]:
```
is_biz_owner = df["HBUS"] == 1

# Step 2: Calculate the number of business owners
num_biz_owners = is_biz_owner.sum()

# Step 3: Calculate the total number of respondents
total_respondents = len(df)
```

[ ]:
```
prop_biz_owners = num_biz_owners / total_respondents
print("proportion of business owners in df:", prop_biz_owners)
```

```
proportion of business owners in df: 0.2740176562229531
```

Task 6.5.3: Create a DataFrame df_inccat that shows the normalized frequency for income categories for business owners and non-business owners. Your final DataFrame should look something like this:

```
    HBUS    INCCAT  frequency
0      0      0-20   0.210348
1      0    21-39.9   0.198140
...
11     1      0-20   0.041188
```

[6]:
```
inccat_dict = {
    1: "0-20",
    2: "21-39.9",
    3: "40-59.9",
    4: "60-79.9",
    5: "80-89.9",
    6: "90-100",
}
df_inccat = (
    df['INCCAT'].replace(inccat_dict).groupby(df['HBUS']).value_counts(normalize=True).rename("frequency").to_frame().reset_index()
)
df_inccat
```

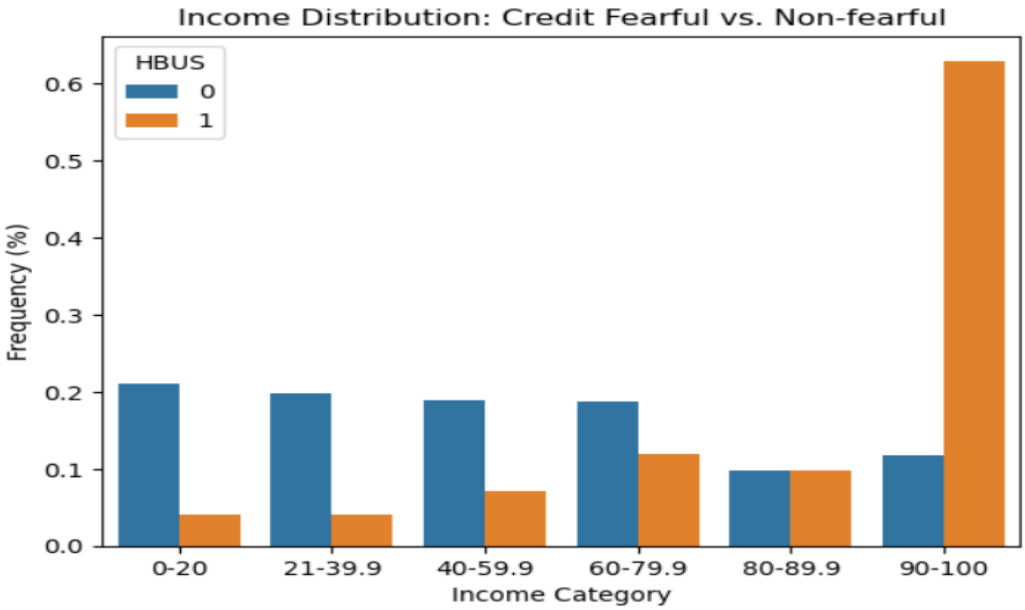
[6]:

	HBUS	INCCAT	frequency
0	0	0-20	0.210348
1	0	21-39.9	0.198140
2	0	40-59.9	0.189080
3	0	60-79.9	0.186600
4	0	90-100	0.117167
5	0	80-89.9	0.098665
6	1	90-100	0.629438
7	1	60-79.9	0.119015
8	1	80-89.9	0.097410
9	1	40-59.9	0.071510
10	1	21-39.9	0.041440
11	1	0-20	0.041188

Task 6.5.4: Using seaborn, create a side-by-side bar chart of df_inccat. Set hue to "HBUS", and make sure that the income categories are in the correct order along the x-axis. Label to the x-axis "Income Category", the y-axis "Frequency (%)", and use the title "Income Distribution: Business Owners vs. Non-Business Owners".

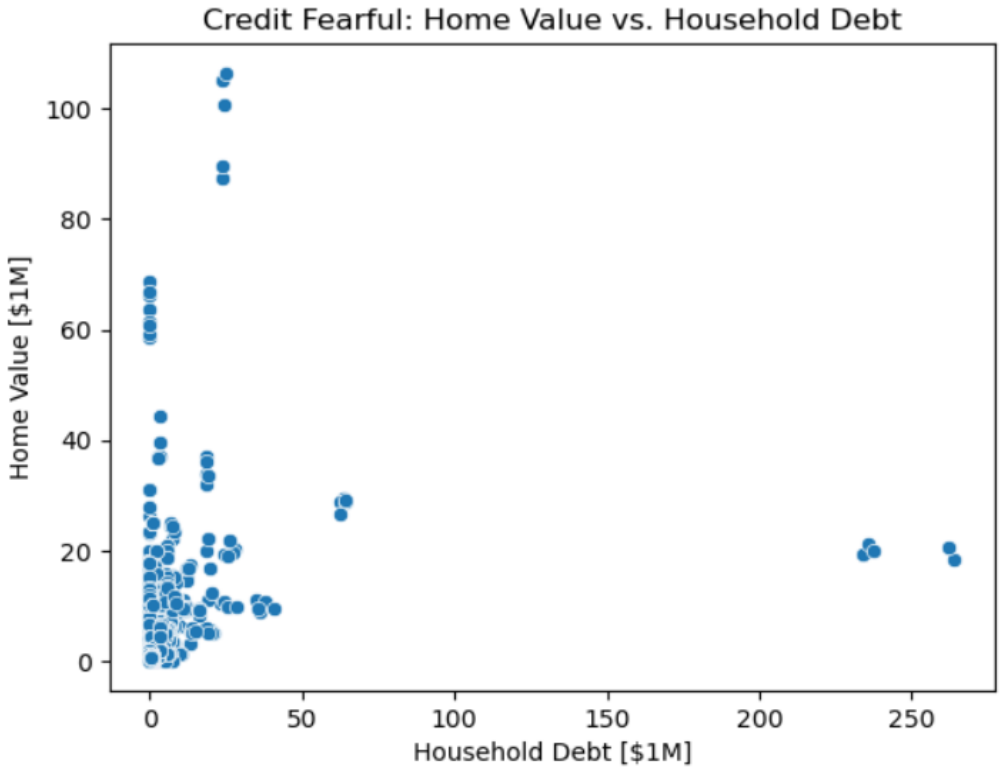
[8]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Create bar chart of `df_inccat`
sns.barplot(
    x="INCCAT",
    y="frequency",
    hue="HBUS",
    data= df_inccat,
    order=inccat_dict.values()
)
plt.xlabel("Income Category")
plt.ylabel("Frequency (%)")
plt.title("Income Distribution: Credit Fearful vs. Non-fearful");
# Don't delete the code below 👇
plt.savefig("images/6-5-4.png", dpi=150)
```



Task 6.5.5: Using seaborn, create a scatter plot that shows "HOUSES" vs. "DEBT". You should color the datapoints according to business ownership. Be sure to label the x-axis "Household Debt", the y-axis "Home Value", and use the title "Home Value vs. Household Debt".

[9]:
```
# Plot "HOUSES" vs "DEBT" with hue as business ownership
sns.scatterplot(x=df["DEBT"]/ 1e6, y =df["HOUSES"]/ 1e6)
plt.xlabel("Household Debt [$1M]")
plt.ylabel("Home Value [$1M]")
plt.title("Credit Fearful: Home Value vs. Household Debt");
# Don't delete the code below 👇
plt.savefig("images/6-5-5.png", dpi=150)
```



.

Task 6.5.6: Create a new DataFrame df_small_biz that contains only business owners whose income is below $500,000.

[10]:
```
df_small_biz = df[(df["HBUS"] == 1) & (df["INCOME"] < 500000)]
print("df_small_biz shape:", df_small_biz.shape)
df_small_biz.head()
```

```
df_small_biz shape: (4364, 351)
```

[10]:

	YY1	Y1	WGT	HHSEX	AGE	AGECL	EDUC	EDCL	MARRIED	KIDS	...	NWCAT	INCCAT	ASSETCAT	NINCCAT	NINC2CAT	NWPCTLECAT	INCP
80	17	171	7802.265717	1	62	4	12	4	1	0	...	3	5	5	5	2	7	
81	17	172	8247.536301	1	62	4	12	4	1	0	...	3	5	5	5	2	7	
82	17	173	8169.562719	1	62	4	12	4	1	0	...	3	5	5	5	2	7	
83	17	174	8087.704517	1	62	4	12	4	1	0	...	3	5	5	5	2	7	
84	17	175	8276.510048	1	62	4	12	4	1	0	...	3	5	5	5	2	7	

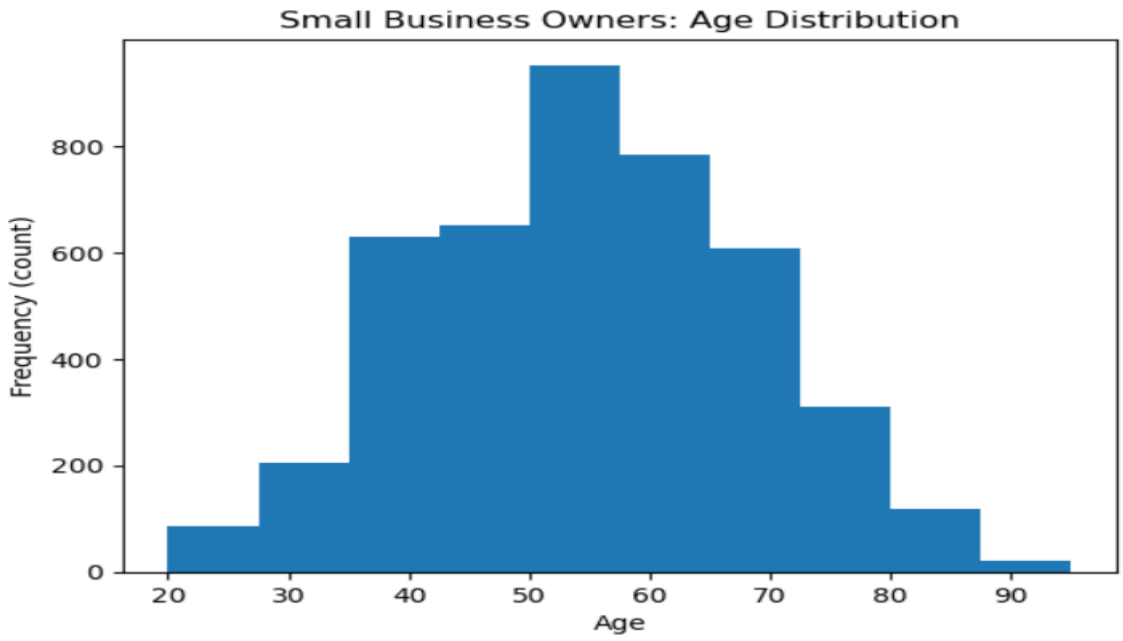
5 rows × 351 columns

Task 6.5.7: Create a histogram from the "AGE" column in df_small_biz with 10 bins. Be sure to label the x-axis "Age", the y-axis "Frequency (count)", and use the title "Small Business Owners: Age Distribution".

[11]:
```
# Plot histogram of "AGE"
plt.hist(df_small_biz["AGE"], bins=10)

# Add labels and title
plt.xlabel("Age")
plt.ylabel("Frequency (count)")
plt.title("Small Business Owners: Age Distribution")

# Show the plot
plt.show()
# Don't delete the code below 👇
plt.savefig("images/6-5-7.png", dpi=150)
```




```
<Figure size 640x480 with 0 Axes>
```

Task 6.5.8: Calculate the variance for all the features in df_small_biz, and create a Series top_ten_var with the 10 features with the largest variance.

[13]:
```
# Calculate variance, get 10 largest features
top_ten_var = df_small_biz.var().sort_values().tail(10)
top_ten_var
```

[13]:
```
EQUITY      1.005088e+13
FIN         2.103228e+13
KGBUS       5.025210e+13
ACTBUS      5.405021e+13
BUS         5.606717e+13
KGTOTAL     6.120760e+13
NHNFIN      7.363197e+13
NFIN        9.244074e+13
NETWORTH    1.424450e+14
ASSET       1.520071e+14
dtype: float64
```

Task 6.5.9: Calculate the trimmed variance for the features in df_small_biz. Your calculations should not include the top and bottom 10% of observations. Then create a Series top_ten_trim_var with the 10 features with the largest variance.

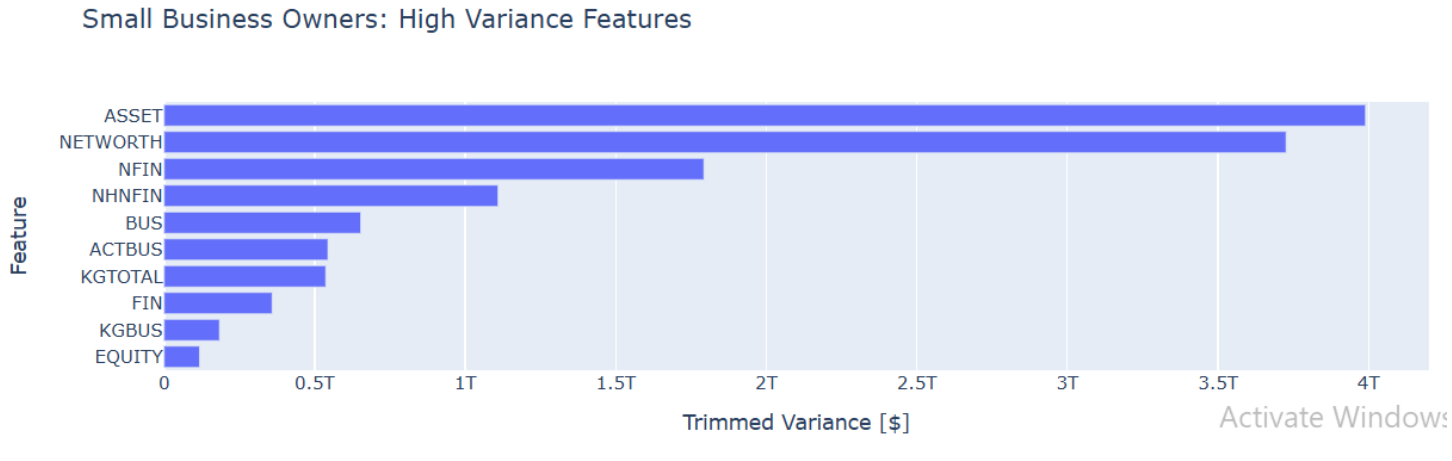
[23]:
```
# Calculate trimmed variance
trimmed_var(df_small_biz['AGE'])
top_ten_trim_var = df_small_biz.apply(trimmed_var, limits=(0.1,0.1)).sort_values().tail(10)
top_ten_trim_var
```

[23]:
```
EQUITY      1.177020e+11
KGBUS       1.838163e+11
FIN         3.588855e+11
KGTOTAL     5.367878e+11
ACTBUS      5.441806e+11
BUS         6.531708e+11
NHNFIN      1.109187e+12
NFIN        1.792707e+12
NETWORTH    3.726356e+12
ASSET       3.990101e+12
dtype: float64
```

Task 6.5.10: Use plotly express to create a horizontal bar chart of top_ten_trim_var. Be sure to label your x-axis "Trimmed Variance [$]", the y-axis "Feature", and use the title "Small Business Owners: High Variance Features".

[25]:
```
# Create horizontal bar chart of `top_ten_trim_var`
fig =px.bar(
    x=top_ten_trim_var,
    y=top_ten_trim_var.index,
    title="Small Business Owners: High Variance Features"
)
fig.update_layout(xaxis_title="Trimmed Variance [$]",yaxis_title="Feature")
# Don't delete the code below 👇
fig.write_image("images/6-5-10.png", scale=1, height=500, width=700)

fig.show()
```



Task 6.5.11: Generate a list high_var_cols with the column names of the five features with the highest trimmed variance.

[26]:
```
high_var_cols = top_ten_trim_var.tail(5).index.to_list()
high_var_cols
```

[26]:
```
['BUS', 'NHNFIN', 'NFIN', 'NETWORTH', 'ASSET']
```

Split

Task 6.5.12: Create the feature matrix X from df_small_biz. It should contain the five columns in high_var_cols.

[28]:
```
X = df_small_biz[high_var_cols]
print("X shape:", X.shape)
X.head()
```

```
X shape: (4364, 5)
```

[28]:

	BUS	NHNFIN	NFIN	NETWORTH	ASSET
80	0.0	224000.0	724000.0	237600.0	810600.0
81	0.0	223000.0	723000.0	236600.0	809600.0
82	0.0	224000.0	724000.0	237600.0	810600.0
83	0.0	222000.0	722000.0	234600.0	808600.0
84	0.0	223000.0	723000.0	237600.0	809600.0

Build Model

Iterate

Task 6.5.13: Use a for loop to build and train a K-Means model where n_clusters ranges from 2 to 12 (inclusive). Your model should include a StandardScaler. Each time a model is trained, calculate the inertia and add it to the list inertia_errors, then calculate the silhouette score and add it to the list silhouette_scores.

Note: For reproducibility, make sure you set the random state for your model to 42.

[32]:
```
n_clusters = range(2,13)
inertia_errors = []
silhouette_scores = []

# Add `for` loop to train model and calculate inertia, silhouette score.
for k in n_clusters:
    model= KMeans(n_clusters=k, random_state=42)
    model.fit(X)
    inertia_errors.append(model.inertia_)
    silhouette_scores.append(silhouette_score(X, model.labels_))

print("Inertia:", inertia_errors[:11])
print()
print("Silhouette Scores:", silhouette_scores[:3])
```

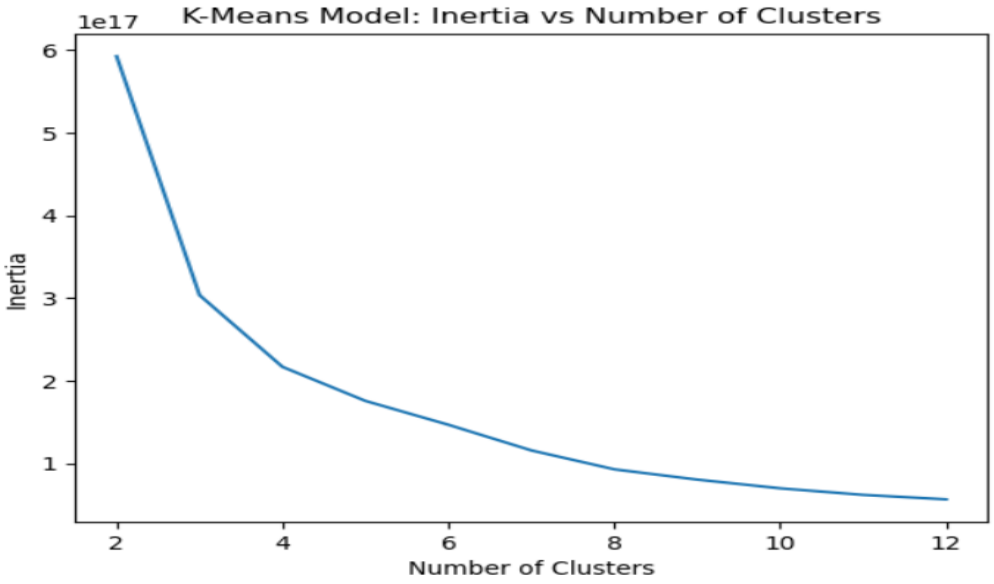
```
Inertia: [5.927671996276195e+17, 3.035245983028536e+17, 2.1665393518732896e+17, 1.7543479282793382e+17, 1.465661085599464e+17, 1.155304566652761
6e+17, 9.261851174290176e+16, 8.017158025799344e+16, 6.961849609125336e+16, 6.170698486096275e+16, 5.6259414464006344e+16]

Silhouette Scores: [0.9577268088436816, 0.8403687512464493, 0.7410560048823569]
```

Task 6.5.14: Use plotly express to create a line plot that shows the values of inertia_errors as a function of n_clusters. Be sure to label your x-axis "Number of Clusters", your y-axis "Inertia", and use the title "K-Means Model: Inertia vs Number of Clusters".

[33]:
```
# Create line plot of `inertia_errors` vs `n_clusters`
plt.plot(n_clusters, inertia_errors)
plt.xlabel("Number of Clusters")
plt.ylabel("Inertia")
plt.title("K-Means Model: Inertia vs Number of Clusters")
# Don't delete the code below 👇
fig.write_image("images/6-5-14.png", scale=1, height=500, width=700)

fig.show()
```

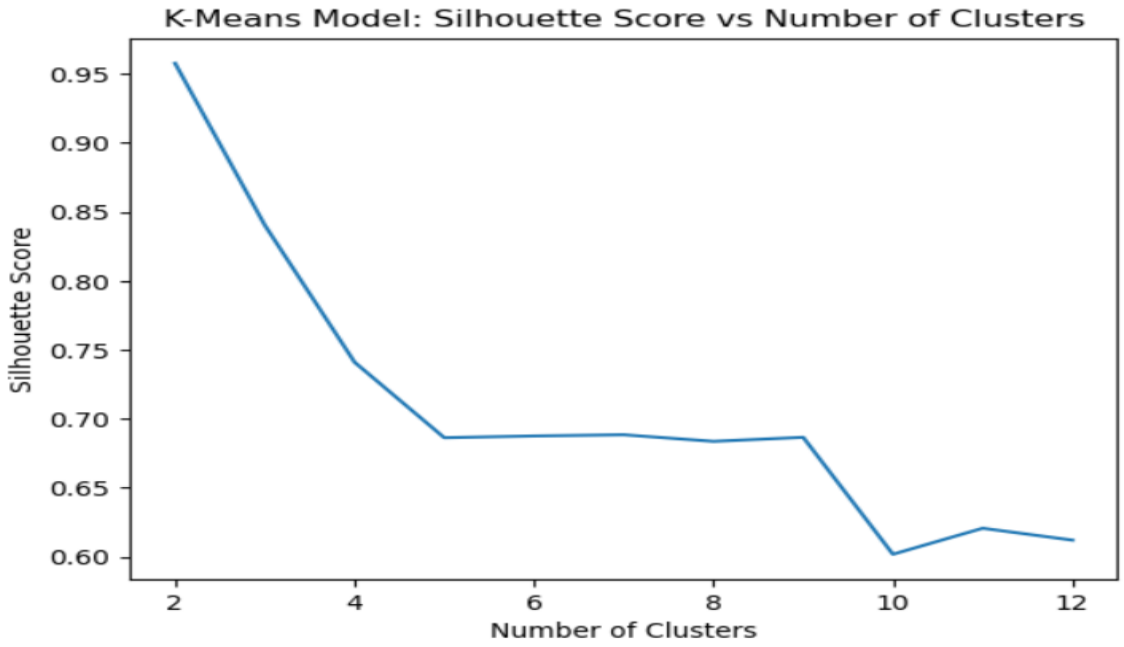


And let's do the same thing with our Silhouette Scores.

Task 6.5.15: Use plotly express to create a line plot that shows the values of silhouette_scores as a function of n_clusters. Be sure to label your x-axis "Number of Clusters", your y-axis "Silhouette Score", and use the title "K-Means Model: Silhouette Score vs Number of Clusters".

[34]:
```
# Create a line plot of `silhouette_scores` vs `n_clusters`
plt.plot(n_clusters, silhouette_scores)
plt.xlabel("Number of Clusters")
plt.ylabel("Silhouette Score")
plt.title("K-Means Model: Silhouette Score vs Number of Clusters")
# Don't delete the code below 👇
fig.write_image("images/6-5-15.png", scale=1, height=500, width=700)

fig.show()
```



Task 6.5.16: Build and train a new k-means model named final_model. The number of clusters should be 3.

Note: For reproducibility, make sure you set the random state for your model to 42.

[36]:
```
final_model = KMeans(n_clusters=3, random_state=42)

final_model.fit(X)
```

```
/opt/conda/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1412: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
```

[36]:
```
KMeans(n_clusters=3, random_state=42)
```

Communicate

Task 6.5.17: Create a DataFrame xgb that contains the mean values of the features in X for the 3 clusters in your final_model.

[37]:
```
labels = final_model.cluster_centers_
xgb = X.groupby(final_model.labels_).mean()
xgb
```

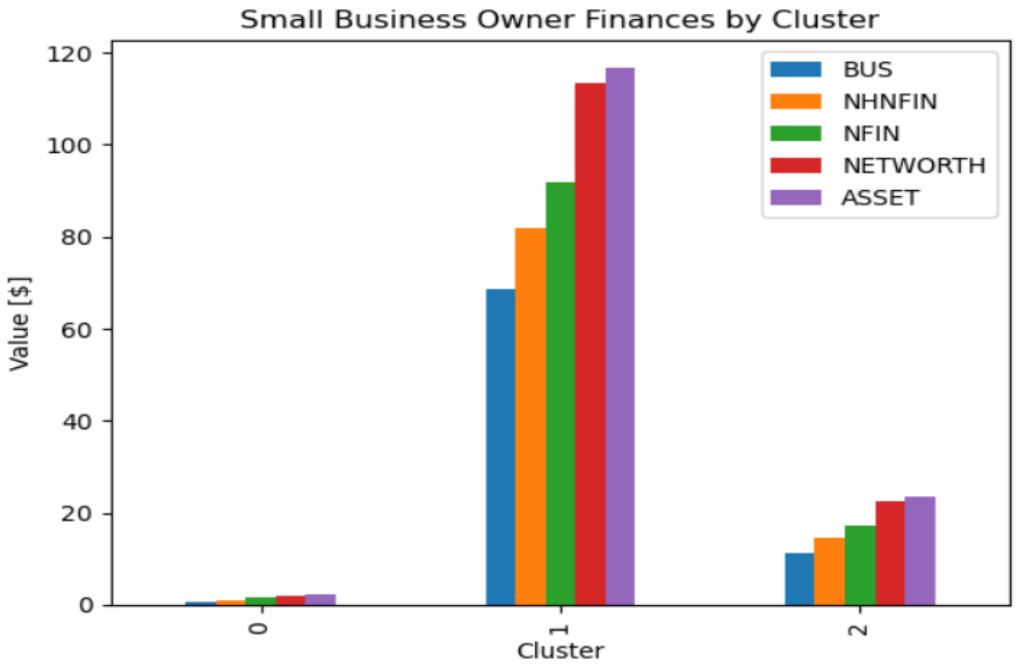
[37]:

	BUS	NHNFIN	NFIN	NETWORTH	ASSET
0	7.247131e+05	9.805064e+05	1.454223e+06	2.001628e+06	2.205959e+06
1	6.874479e+07	8.202115e+07	9.169652e+07	1.134843e+08	1.167529e+08
2	1.122632e+07	1.459438e+07	1.722187e+07	2.237408e+07	2.342352e+07

Task 6.5.18: Use plotly express to create a side-by-side bar chart from xgb that shows the mean of the features in X for each of the clusters in your final_model. Be sure to label the x-axis "Cluster", the y-axis "Value [$]", and use the title "Small Business Owner Finances by Cluster"

[39]:
```
# Create side-by-side bar chart of `xgb`
(xgb/1e6).plot(kind="bar")
plt.xlabel("Cluster")
plt.ylabel("Value [$]")
plt.title("Small Business Owner Finances by Cluster")
# Don't delete the code below 👇
fig.write_image("images/6-5-18.png", scale=1, height=500, width=700)

#fig.show()
```



Task 6.5.19: Create a PCA transformer, use it to reduce the dimensionality of the data in X to 2, and then put the transformed data into a DataFrame named X_pca. The columns of X_pca should be named "PC1" and "PC2".

[40]:
```
# Instantiate transformer
pca = PCA(n_components=2, random_state=42)

# Transform `X`
X_t = pca.fit_transform(X)

# Put `X_t` into DataFrame
X_pca = pd.DataFrame(X_t, columns=['PC1','PC2'])

print("X_pca shape:", X_pca.shape)
X_pca.head()
```

```
X_pca shape: (4364, 2)
```

[40]:

	PC1	PC2
0	-6.220648e+06	-503841.638839
1	-6.222523e+06	-503941.888901
2	-6.220648e+06	-503841.638839
3	-6.224927e+06	-504491.429465
4	-6.221994e+06	-503492.598399

Task 6.5.20: Use plotly express to create a scatter plot of X_pca using seaborn. Be sure to color the data points using the labels generated by your final_model. Label the x-axis "PC1", the y-axis "PC2", and use the title "PCA Representation of Clusters".

[43]:
```
# Create scatter plot of `PC2` vs `PC1`
fig = px.scatter(
    data_frame = X_pca,
    x='PC1',
    y='PC2',
    title="PCA Representation of Clusters"
)
# Don't delete the code below 👇
fig.write_image("images/6-5-20.png", scale=1, height=500, width=700)

fig.show()
```