**Project 4**

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

from sklearn.impute import SimpleImputer

from sklearn.decomposition import PCA

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

**## Question 1: Data import exploration and cleaning**

df = pd.read\_excel(r'C:\Users\shaun.rolph\Desktop\Assignmentfour\Food\_Environment\_Atlas.xlsx', sheet\_name='Sheet1')

print(df)

df.head()

df.describe()

df.Economic\_Type\_String.value\_counts()

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text

Description automatically generated

df.shape[0]



**## Cleaning nulls**

columns\_to\_drop = ['Type\_Farming', 'Type\_Mining', 'Type\_Manufacturing','Type\_Government', 'Type\_Recreation', 'Type\_Nonspecialized']

df.drop(columns= columns\_to\_drop, inplace=True)

nulls\_per\_Row= df.isnull().sum(axis=1)

print(nulls\_per\_Row.sort\_values(ascending=False))

df\_clean = df.dropna(axis=1, thresh=(len(df)\*.1), inplace=False)

**## Separate df for just target variable**

df\_target=pd. DataFrame()

df\_target["Economic\_Type\_String"]= df["Economic\_Type\_String"]

print(df\_target)

Graphical user interface, text

Description automatically generated

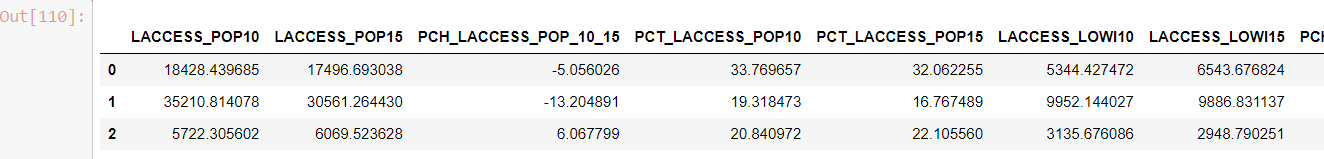
**## Question 2: Prepping and imputing for PCA**

df\_clean = df\_clean.\_get\_numeric\_data()

columns\_to\_drop = ['FIPS']

df\_clean.drop(columns= columns\_to\_drop, inplace=True)

df\_clean.head()



**## Question 2: Imputing**

from sklearn.impute import KNNImputer

**# Create a KNN imputer object**

knn\_imputer = KNNImputer(n\_neighbors=2)

**# KNN imputation**

knn\_imputed = knn\_imputer.fit\_transform(df\_clean)

**# Convert back to DataFrame**

knn\_imputed\_df = pd.DataFrame(knn\_imputed, columns=df\_clean.columns)

print("KNN imputation:")

print(knn\_imputed\_df)

Text

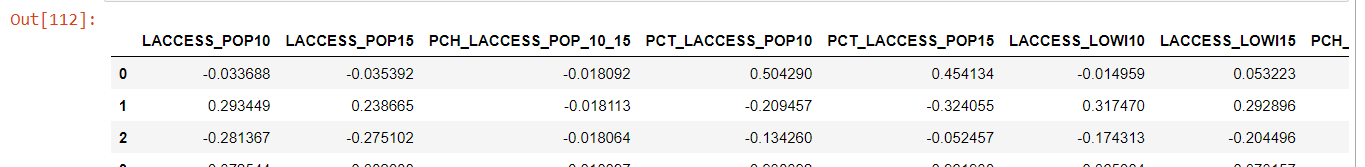
Description automatically generated with low confidence

**##standardize data**

scaler = StandardScaler()

df\_SD = pd.DataFrame(scaler.fit\_transform(knn\_imputed\_df),columns = knn\_imputed\_df.columns)

df\_SD.head()



**##Question 3: PCA**

pca = PCA(n\_components=15)

pca\_arr = pca.fit\_transform(df\_SD)

**## Transform the data to principal components**

X\_pca = pca.transform(df\_SD)

**# Create a DataFrame with PC1 and PC2**

df\_pca = pd.DataFrame(data=X\_pca[:, :2], columns=['PC1', 'PC2'])

**# Sample 100-500 points**

sample\_size = np.random.randint(100, 501)

df\_sampled = df\_pca.sample(sample\_size, random\_state=42)

**## Plot the scatterplot with colored points by county economic type**

plt.figure(figsize=(10, 6))

sns.scatterplot(x='PC1', y='PC2', data=df\_sampled, hue=df.loc[df\_sampled.index, 'Economic\_Type\_String'], palette='Set1')

plt.title('PCA: PC1 vs PC2 (Sampled)')

plt.xlabel('Principal Component 1 (PC1)')

plt.ylabel('Principal Component 2 (PC2)')

plt.legend(title='Economic Type', loc='upper right')

plt.show()

**##Quesiton 4: Scree plot for variance explained**

plt.figure(figsize=(10, 6))

plt.plot(range(1, pca.n\_components\_ + 1), pca.explained\_variance\_ratio\_, marker='o', linestyle='-')

plt.title('Scree Plot')

plt.xlabel('Number of Principal Components')

plt.ylabel('Variance Explained')

plt.xticks(range(1, pca.n\_components\_ + 1))

plt.grid(True)

plt.show()

Chart, scatter chart

Description automatically generated

**## Question 5: KMeans**

from sklearn.cluster import KMeans

**## Run k-means clustering with k=6**

kmeans = KMeans(n\_clusters=6, random\_state=42)

clusters = kmeans.fit\_predict(X\_pca)

**## Add the cluster labels to the original DataFrame**

df['Cluster'] = clusters

**## Create a DataFrame to store the cluster counts for each economic type**

cluster\_counts = df.groupby(['Economic\_Type\_String', 'Cluster']).size().unstack(fill\_value=0)

**## Quesiton 6: Stacked Bar Chart**

**# Normalize cluster counts to create a stacked bar chart**

cluster\_counts\_norm = cluster\_counts.div(cluster\_counts.sum(axis=1), axis=0)

## Plot a stacked bar chart

plt.figure(figsize=(12, 12))

cluster\_counts\_norm.plot(kind='bar', stacked=True, colormap='Set1')

plt.title('Stacked Bar Chart of Economic Types by Clusters')

plt.xlabel('True Economic Type')

plt.ylabel('Proportion of Counties')

plt.legend(title='Cluster', loc='center left', bbox\_to\_anchor=(1, 0.5))

plt.xticks(rotation=45)

plt.show()

**## Quesiton 7: Model Accuracy**

**## Calculate observed accuracy for each economic type**

accuracy = {}

for economic\_type, counts in cluster\_counts.iterrows():

observed\_accuracy = counts.max() / counts.sum()

accuracy[economic\_type] = observed\_accuracy

print("Observed Accuracy in Predicting Each County's Economic Type:")

for economic\_type, acc in accuracy.items():

print(f"{economic\_type}: {acc:.2%}")

Chart, bar chart

Description automatically generated

**## Quesiton 8 k-NN model**

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

## Split data into features (X) and target labels (y)

X = df\_pca

y = df['Economic\_Type\_String']

**## Train a k-NN model**

knn = KNeighborsClassifier(n\_neighbors=6) # Considering 6 economic types

knn.fit(X, y)

**## Predict economic types for each county**

y\_pred = knn.predict(X)

**## Add predicted labels to the DataFrame**

df['Predicted\_Economic\_Type'] = y\_pred

**## Create a DataFrame to store the predicted cluster counts for each economic type**

predicted\_counts = df.groupby(['Economic\_Type\_String', 'Predicted\_Economic\_Type']).size().unstack(fill\_value=0)

**## Normalize predicted cluster counts to create a stacked bar chart**

predicted\_counts\_norm = predicted\_counts.div(predicted\_counts.sum(axis=1), axis=0)

**## Plot a stacked bar chart for predicted economic types**

plt.figure(figsize=(12, 8))

predicted\_counts\_norm.plot(kind='bar', stacked=True, colormap='Set1')

plt.title('Stacked Bar Chart of Predicted Economic Types by Clusters (k-NN)')

plt.xlabel('Economic Type')

plt.ylabel('Proportion of Counties')

plt.legend(title='Predicted Cluster', loc='center left', bbox\_to\_anchor=(1, 0.5))

plt.xticks(rotation=45)

plt.show()

**## Calculate accuracy of k-NN for each economic type**

accuracy\_by\_type = {}

for economic\_type in df['Economic\_Type\_String'].unique():

mask = df['Economic\_Type\_String'] == economic\_type

accuracy\_by\_type[economic\_type] = accuracy\_score(df.loc[mask, 'Economic\_Type\_String'], df.loc[mask, 'Predicted\_Economic\_Type'])

**## Print accuracy for each economic type**

print("Accuracy of k-NN for Each Economic Type:")

for economic\_type, acc in accuracy\_by\_type.items():

print(f"{economic\_type}: {acc:.2%}")

Chart, bar chart

Description automatically generated

Drawing Conclusions:

Write a qualitative assessment of your results:

• Did the k-means method create clusters that seemed to correspond to the true economic types?

Cluster may not align perfectly with the true economic types as it is an unsupervised method and it also dependent on the data chosen and the number of clusters.

• Did the k-NN method perform well with the PCs as the predictors?

It did not perform well, this would be an issue with the data cleaning.

• What method seemed to work better? Can you speculate as to why?

k-NN did as it is a supervised model that learns directly from the labeled data.

• If you re-run a model, do you get similar results each time? Why or why not?

The model result will change if the data splitting is ran and trained again or any other cleaning methods are used in the beginning of the problem set.

• Having gone through this exercise, what would you do differently next time to improve your analysis?

I would focus on cleaning the data more effectively, implement cross validation techniques, and explore additional data to include within the model.