Tiffany Tran, Megan Alicea, Neil Parsa

Professor Cohen

CIS 3920

17 August 2023

**Group Project – Source Code**

**#Cleaning the Data**

# importing libraries

import pandas as pd

# importing the csv file and putting the wanted data into a dataframe

filename = 'data.csv'

cols = ['Route', 'Average Weekday Ridership, 2011', 'Average Weekday Ridership, 2016', 'Ridership Change, 2011-2016', '# of Stops along Route', 'Bus Stops within .1 Miles of a Subway Station', 'Borough']

file = pd.read\_csv(filename, usecols = cols) # only using the selected columns

df = pd.DataFrame(file)

print(df)

# removing the last few rows

df = df.drop(df.tail(8).index)

# renaming the columns of the dataframe

new\_cols = ['Route', 'Avg. 2011 Ridership', 'Avg. 2016 Ridership', 'Ridership Change', '# of Stops', '# of Stops near Subway', 'Borough']

new\_names = {}

count = 0

for name in cols:

new\_names[name] = new\_cols[count]

count += 1

df = df.rename(columns= new\_names)

# creating a series

new\_dict = {}

for name in new\_cols:

new\_dict[name] = []

# putting all the values from the dataframe into th series to alter the data

for i in range(len(df)):

for col in new\_cols:

new\_dict[col].append(df[col][i])

# change the data type for the numerical values

for i in range(len(df)):

route = new\_dict['Route'][i]

if '\*' in new\_dict['Route'][i]:

new\_dict['Route'][i] = route.replace('\*', '')

num = new\_dict['Avg. 2011 Ridership'][i]

new\_dict['Avg. 2011 Ridership'][i] = int(num.replace(',', ''))

num = new\_dict['Avg. 2016 Ridership'][i]

new\_dict['Avg. 2016 Ridership'][i] = int(num.replace(',', ''))

1# putting the dictionary into a dataframe

for key in new\_dict:

df[key] = new\_dict[key]

print(df)

df.to\_csv('new\_data.csv', index = False)

**#Linear Regression**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import accuracy\_score, confusion\_matrix

filename = 'new\_data.csv'

file = pd.read\_csv(filename)

df = pd.DataFrame(file)

print(df)

x = np.array(df['# of Stops near Subway']).reshape(-1,1)

y = np.array(df['Avg. 2011 Ridership'])

plt.scatter(x, y)

plt.xlabel('# of stops')

plt.ylabel('Avg.Ridership')

plt.show()

model = LinearRegression().fit(x,y)

r\_sq = model.score(x, y)

print(f"coefficient of determination: {r\_sq}")

print(f"intercept: {model.intercept\_}")

print(f"slope: {model.coef\_}")

y\_pred = model.predict(x)

print(f"predicted response:\n{y\_pred}")

plt.scatter(x, y\_pred)

plt.xlabel('# of stops near subway')

plt.ylabel('Avg.Ridership')

plt.scatter(x, df['Avg. 2016 Ridership'])

plt.show()

borough = df[['Avg. 2011 Ridership', 'Avg. 2016 Ridership', 'Borough']].groupby(['Borough']).mean()

print(borough)

borough.plot(kind='bar', stacked=False, figsize=(10, 6))

**#Logarithmic Regression**

##Import Libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.neighbors import KNeighborsClassifier

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, confusion\_matrix,classification\_report

import warnings

warnings.filterwarnings('ignore')

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from scipy.stats import mode

##Read csv from path

dataset = pd.read\_csv('new\_data.csv')

##Check csv original

dataset.head()

## FORMATTING COLUMNS

##drop route coulmn

dataset = dataset.drop('Route', axis=1)

##drop Avg. 2011 Ridership coulmn

dataset = dataset.drop('Avg. 2011 Ridership', axis=1)

##drop Ridership Change % coulmn

dataset = dataset.drop('Ridership Change', axis=1)

##drop # of Stops coulmn

dataset = dataset.drop('# of Stops', axis=1)

##drop Borough coulmn

dataset = dataset.drop('Borough', axis=1)

dataset.head()

##Visualize data

sns.displot(dataset, x = 'Avg. 2016 Ridership', hue = '# of Stops near Subway')

##Split Data

Y = dataset.drop('# of Stops near Subway', axis=1)

X = dataset['# of Stops near Subway']

ln\_Y = np.log(Y)

print(X)

print(Y)

### Use the relation ln(Y) = ln(A) - BX to fit X to ln(Y)

exp\_reg = LinearRegression()

exp\_reg.fit(X.values.reshape(-1,1), ln\_Y)

exp\_reg\_weighted = LinearRegression()

exp\_reg\_weighted.fit(X.values.reshape(-1,1), ln\_Y, sample\_weight=np.array(1/((X - 100).values\*\*2)).reshape(-1))

### Get predicted values of Y

Y\_pred = np.exp(exp\_reg.predict(X.values.reshape(-1,1)))

Y\_pred\_weighted = np.exp(exp\_reg\_weighted.predict(X.values.reshape(-1,1)))

### Plot

plt.scatter(X, Y)

plt.plot(X, Y\_pred, label='Default')

plt.plot(X, Y\_pred\_weighted, label='Weighted')

plt.xlabel('X')

plt.ylabel('Y')

plt.legend()

plt.show()

**#Clustering**

# Importing important libraries/packages

import pandas as pd

import numpy as np

import seaborn as sns

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_score

from sklearn.model\_selection import cross\_val\_score

import matplotlib.pyplot as plt

# Loading data

clustering\_data = pd.read\_csv("/content/new\_data.csv")

clustering\_data

# dropping percent change

clustering\_data = clustering\_data.drop('Ridership Change', axis=1)

# Select relevant features for clustering

selected\_features = ['Avg. 2011 Ridership',

'Avg. 2016 Ridership',

'# of Stops',

'# of Stops near Subway']

# Normalize/standardize the selected features

scaler = StandardScaler()

standardized\_data = scaler.fit\_transform(clustering\_data[selected\_features])

# Using the Elbow method - Trying different values of K

wcss = [] # within cluster sum-of-squares

for k in range(1, 11):

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

kmeans.fit(standardized\_data)

wcss.append(kmeans.inertia\_)

# Plot the Elbow Method graph

plt.plot(range(1, 11), wcss, marker='o')

plt.xlabel('Number of Clusters (K)')

plt.ylabel('WCSS')

plt.title('Elbow Method')

plt.show()

# Using Silhouette Score Method

silhouette\_scores = []

for k in range(2, 11):

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

cluster\_labels = kmeans.fit\_predict(standardized\_data)

silhouette\_scores.append(silhouette\_score(standardized\_data, cluster\_labels))

# Plot the Silhouette Score graph

plt.plot(range(2, 11), silhouette\_scores, marker='o')

plt.xlabel('Number of Clusters (K)')

plt.ylabel('Silhouette Score')

plt.title('Silhouette Score')

plt.show()

# Choose the number of clusters (K)

num\_clusters = 5

# Apply K-means clustering

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

clustering\_data['cluster'] = kmeans.fit\_predict(standardized\_data)

# Visualize the clusters (2D scatter plot)

sns.set(style="whitegrid")

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

sns.scatterplot(x='Avg. 2011 Ridership', y='Avg. 2016 Ridership', hue='cluster', data=clustering\_data)

plt.title('K-means Clustering of Bus Routes')

plt.xlabel('Average Weekday Ridership 2011')

plt.ylabel('Average Weekday Ridership 2016')

plt.show()

# Explore cluster characteristics

cluster\_summary = clustering\_data.groupby('cluster')[selected\_features].mean()

print(cluster\_summary)

# Plot the cluster characteristics

fig, axes = plt.subplots(nrows=len(selected\_features), ncols=1, figsize=(10, 6 \* len(selected\_features)))

for idx, feature in enumerate(selected\_features):

ax = axes[idx]

cluster\_summary[feature].plot(kind='bar', ax=ax, color='skyblue')

ax.set\_title(f'Average {feature} by Cluster')

ax.set\_xlabel('Cluster')

ax.set\_ylabel(feature)

ax.set\_xticklabels(cluster\_summary.index, rotation=0)

ax.grid(axis='y', linestyle='--')

plt.tight\_layout()

plt.show()

# Profile each cluster based on additional features (e.g., Borough)

borough\_distribution = clustering\_data.groupby(['cluster', 'Borough']).size().unstack(fill\_value=0)

print(borough\_distribution)

# Plot the bar chart

borough\_distribution.plot(kind='bar', stacked=False, figsize=(10, 6))

plt.title('Cluster Distribution by Borough')

plt.xlabel('Cluster')

plt.ylabel('Number of Routes')

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

plt.xticks(rotation=0)

plt.show()

# Evaluating silhouette score

# Higher silhouette scores indicate better-defined clusters.

silhouette\_avg = silhouette\_score(standardized\_data, kmeans.labels\_)

print("Silhouette Score:", silhouette\_avg)