**A README FILE**

**BY**

**ANDREW IGNATIUS BROWN**

**MASTERSCHOOL**

**ON**

**5TH OCTOBER 2023.**

**TOPIC: CUSTOMER SEGMENTATION OF CUSTOMERS AT TRAVELTIDE (PYTHON CODE).**

**Here we start-up with data. Generating/Calculating Missing Variables. Calculating the perks, metrics, and Haversine Distances as well**

import math

import pandas as pd

import numpy as np

# Read your CSV file

df = pd.read\_csv(r"CsvFromSqlQuery.csv")

# Define the new column names

new\_cols = ['nights', 'hotel\_per\_room\_usd', 'rooms', 'base\_fare\_usd', 'checked\_bags', 'destination\_airport\_lat']

# Generate random integer values for the new column

for col in new\_cols:

df[col] = np.random.randint(0, 100, size=len(df))

#add longitude in negatives

df['destination\_airport\_lon'] = np.random.randint(-100, -1, size=len(df))

##### FIRST CREATE HAVERSINE DISTANCE FUNCTION

#define and produce haversine formular

def haversine\_distance(lat1, lon1, lat2, lon2):

# Radius of the Earth in kilometers

earth\_radius = 6371.0

# Convert latitude and longitude from degrees to radians

lat1 = math.radians(lat1)

lon1 = math.radians(lon1)

lat2 = math.radians(lat2)

lon2 = math.radians(lon2)

# Calculate the differences in latitude and longitude

dlat = lat2 - lat1

dlon = lon2 - lon1

# Calculate the Haversine formula

a = math.sin(dlat/2)\*\*2 + math.cos(lat1) \* math.cos(lat2) \* math.sin(dlon/2)\*\*2

c = 2 \* math.atan2(math.sqrt(a), math.sqrt(1-a))

# Calculate the distance

distance = earth\_radius \* c

return distance

###### HAVERSINE FUNCTION CREATED

# Calculate Haversine distance for each row

df['haversine\_distance'] = df.apply(lambda row: haversine\_distance(row['home\_airport\_lat'], row['home\_airport\_lon'], row['destination\_airport\_lat'], row['destination\_airport\_lon']), axis=1)

#creating the “real\_cost\_of\_flight” column

df['real\_cost\_of\_flight'] = df['base\_fare\_usd'] / df['haversine\_distance']

#calculate the values for the perk

#perk one

df['variable\_one'] = df['hotel\_per\_room\_usd'] + df['page\_click\_sum']

df['perk\_one'] = df['variable\_one'] \* df['nights'] \* df['rooms']

df['perk\_two'] = df['base\_fare\_usd'] + df['page\_click\_sum'] + df['checked\_bags']

df['perk\_four'] = df['real\_cost\_of\_flight'] + df['booked\_hotel'] +df['booked\_flight'] + df['hotel\_discount'] + df['flight\_discount']

#calculating perk 5 (one free night with hotel)

r = df['booked\_hotel'].sum() #sum of hotel and flight booked for each cohort

df['perk\_five'] = df['booked\_hotel'] / r

#sum of hotel and flight booked by cohorts

hotel\_sum = df['booked\_hotel'].sum()

flight\_sum = df['booked\_flight'].sum()

#rate of hotel and flight booked by a cohort

df['booked\_hotel\_ratio'] = df['booked\_hotel'] / hotel\_sum

df['booked\_flight\_ratio'] = df['booked\_flight'] / flight\_sum

# Print the modified DataFrame

print(df)

**CORRELATION MATRIX OF Attributes/FEATURES**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

# Select the columns you want to include in the correlation matrix

# Define the new column names

# Define the new column names

selected\_columns = ['session\_time\_scaled', 'page\_click\_sum', 'discount\_flight\_tf', 'discount\_hotel\_tf', 'nights', 'hotel\_per\_room\_usd', 'rooms', 'real\_cost\_of\_flight', 'checked\_bags', 'hotel\_discount', 'flight\_discount']

dar = df[selected\_columns]

# Calculate the Pearson's correlation matrix

corr\_matrix = dar.corr()

# Create a heatmap using seaborn

plt.figure(figsize=(10, 8)) # Adjust the figure size as needed

sns.set(font\_scale=1.2)

sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm', linewidths=0.5)

plt.title("Pearson's Correlation Heatmap")

plt.show()

**CORRELATION MATRIX FOR NO CANCELLATION FEE**

select\_columns = ['session\_time\_scaled', 'page\_click\_sum', 'hotel\_discount', 'flight\_discount', 'session\_total', 'hotel\_per\_room\_usd', 'booked\_flight', 'real\_cost\_of\_flight', 'booked\_hotel', ]

dar = df[select\_columns]

# Calculate the Pearson's correlation matrix

corr\_matrix = dar.corr()

# Create a heatmap using seaborn

plt.figure(figsize=(10, 8)) # Adjust the figure size as needed

sns.set(font\_scale=1.2)

sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm', linewidths=0.5)

plt.title("Pearson's Correlation Heatmap")

plt.show()

**MACHINE LEARNING FOR NO CANCELLATION PERK WITH THE KNN REGRESSION ALGORITHM.**

import numpy as np

import pandas as pd

from sklearn.preprocessing import StandardScaler

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

from sklearn.neighbors import KNeighborsRegressor

from sklearn.metrics import mean\_squared\_error, r2\_score

import matplotlib.pyplot as plt

# Sample data (replace this with your own dataset)

# Assuming you have a DataFrame 'df' with the necessary columns

# Here, we're selecting specific columns from the DataFrame

X = df[['page\_click\_sum', 'flight\_discount', 'hotel\_discount', 'session\_total', 'real\_cost\_of\_flight', 'hotel\_per\_room\_usd', 'booked\_flight\_ratio']].values

y = df['cancel\_session'].values

# Standardize features (recommended for KNN)

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Create a KNN regressor

k = 7 # Number of neighbors (you can adjust this)

knn\_regressor = KNeighborsRegressor(n\_neighbors=k)

# Fit the model to the entire dataset

knn\_regressor.fit(X\_scaled, y)

# Predict using the model (on the same dataset)

y\_pred = knn\_regressor.predict(X\_scaled)

# Evaluate the model

mse = mean\_squared\_error(y, y\_pred)

r2 = r2\_score(y, y\_pred)

print(f"Mean Squared Error: {mse:.2f}")

print(f"R-squared: {r2:.2f}")

# Plot the results

plt.scatter(y, y\_pred, color='blue')

plt.xlabel('Actual\_Cancellation')

plt.ylabel('Predicted\_Cancellation')

plt.title('KNN Regression')

plt.show()

# Add the predicted values to the DataFrame

df['perk\_three'] = y\_pred

print(df)

**PERFORMING K-MEANS CLUSTERING ON THE DATASET**

# Import necessary libraries

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.datasets import make\_blobs

from sklearn.utils import shuffle

# Generate some example data

X = df['perk\_one'], df['perk\_two'], df['perk\_three'], df['perk\_four'], df['perk\_five']

X = shuffle(X, random\_state=42)

# Create and fit the K-means model

kmeans = KMeans(n\_clusters=n\_clusters)

kmeans.fit(X)

# Get cluster centers and labels

cluster\_centers = kmeans.cluster\_centers\_

#predict cluster

cluster\_labels = kmeans.labels\_

# Plot the data points and cluster centers

plt.scatter(X[:, 0], X[:, 1], c=cluster\_labels, cmap='viridis')

plt.scatter(cluster\_centers[:, 0], cluster\_centers[:, 1], c='red', marker='x', s=200)

plt.title('K-means Clustering')

plt.xlabel('Feature 1')

plt.ylabel('Feature 2')

plt.show()

print(cluster\_labels)

df['K\_means'] = cluster\_labels

df.to\_csv(r"C:\Users\andre\OneDrive\Documents\MasterSchool\_data\_analytics\project\created1\_data.csv", index=False)