import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sb

from sklearn.preprocessing import StandardScaler, LabelEncoder

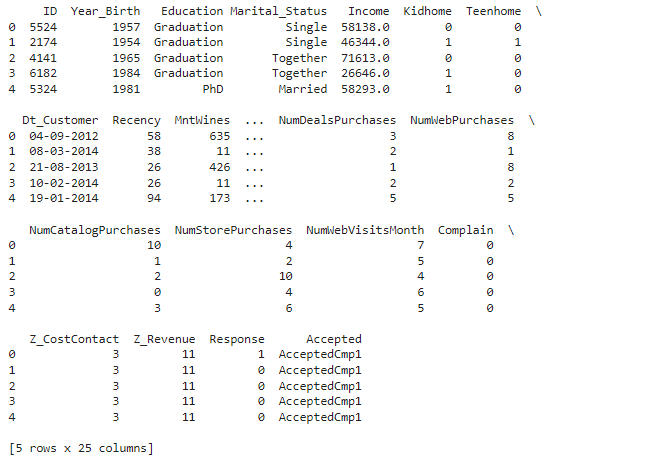
from sklearn.cluster import KMeans

import warnings

warnings.filterwarnings('ignore')

df = pd.read\_csv('new.csv')

df.head()

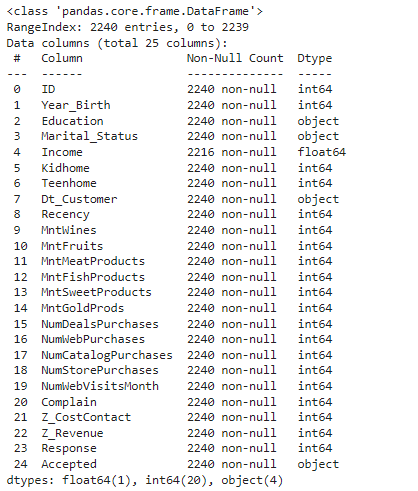


df.shape

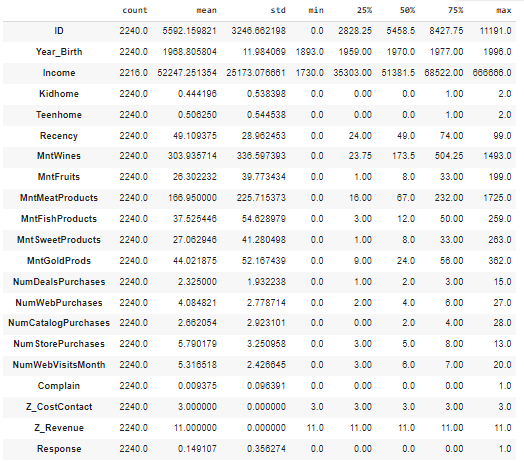
output:

(2240, 25)(2240, 25)

df.info()



df.describe().T



df['Accepted'] = df['Accepted'].str.replace('Accepted', '')

for col in df.columns:

temp = df[col].isnull().sum()

if temp > 0:

print(f'Column {col} contains {temp} null values.')

output:

Column Income contains 24 null values.

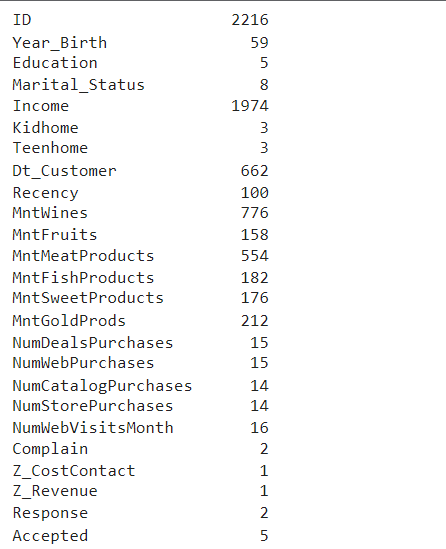
df = df.dropna()

print("Total missing values are:", len(df))

**Output:**

 Total missing values are: 2216

df.nunique()



parts = df["Dt\_Customer"].str.split("-", n=3, expand=True)

df["day"] = parts[0].astype('int')

df["month"] = parts[1].astype('int')

df["year"] = parts[2].astype('int')

df.drop(['Z\_CostContact', 'Z\_Revenue', 'Dt\_Customer'],

axis=1,

inplace=True)

floats, objects = [], []

for col in df.columns:

if df[col].dtype == object:

objects.append(col)

elif df[col].dtype == float:

floats.append(col)

print(objects)

print(floats)

**Output:**

['Education', 'Marital\_Status', 'Accepted']

['Income']

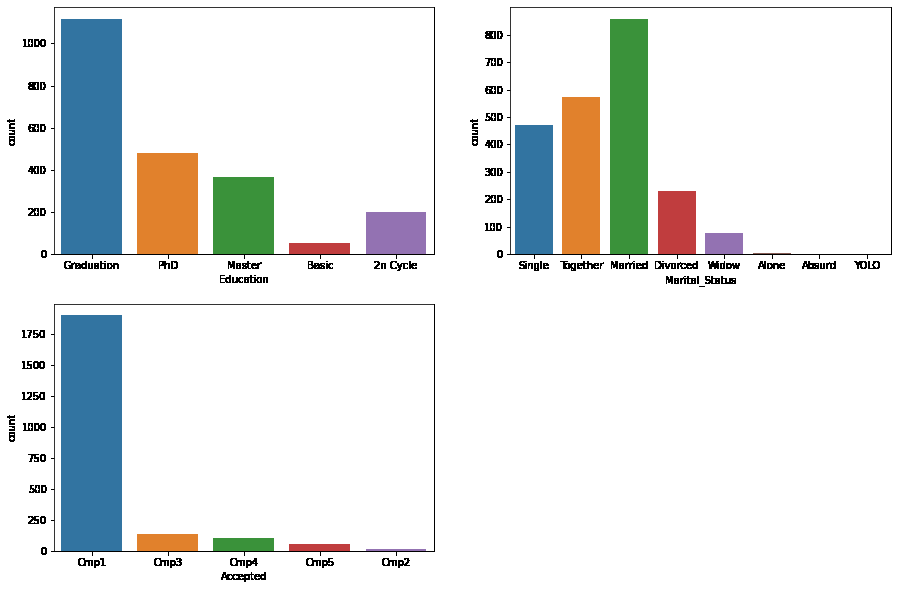
plt.subplots(figsize=(15, 10))

for i, col in enumerate(objects):

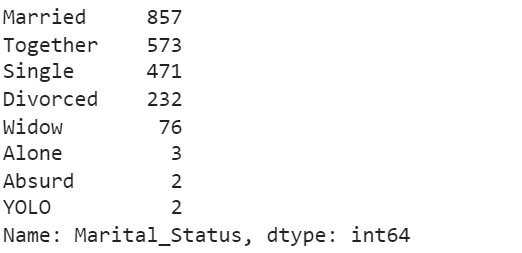
plt.subplot(2, 2, i + 1)

sb.countplot(df[col])

plt.show()



df['Marital\_Status'].value\_counts()



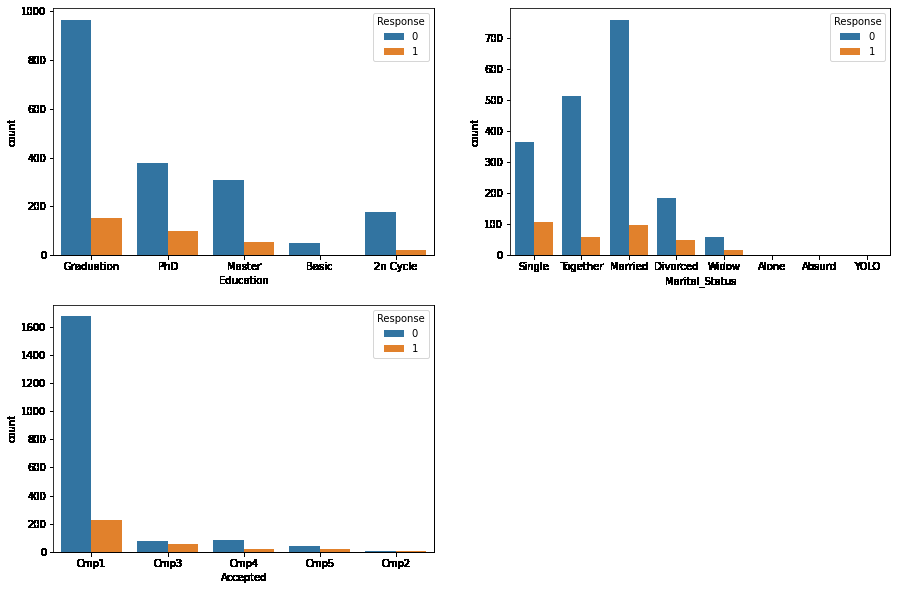
plt.subplots(figsize=(15, 10))

for i, col in enumerate(objects):

plt.subplot(2, 2, i + 1)

sb.countplot(df[col], hue=df['Response'])

plt.show()



for col in df.columns:

if df[col].dtype == object:

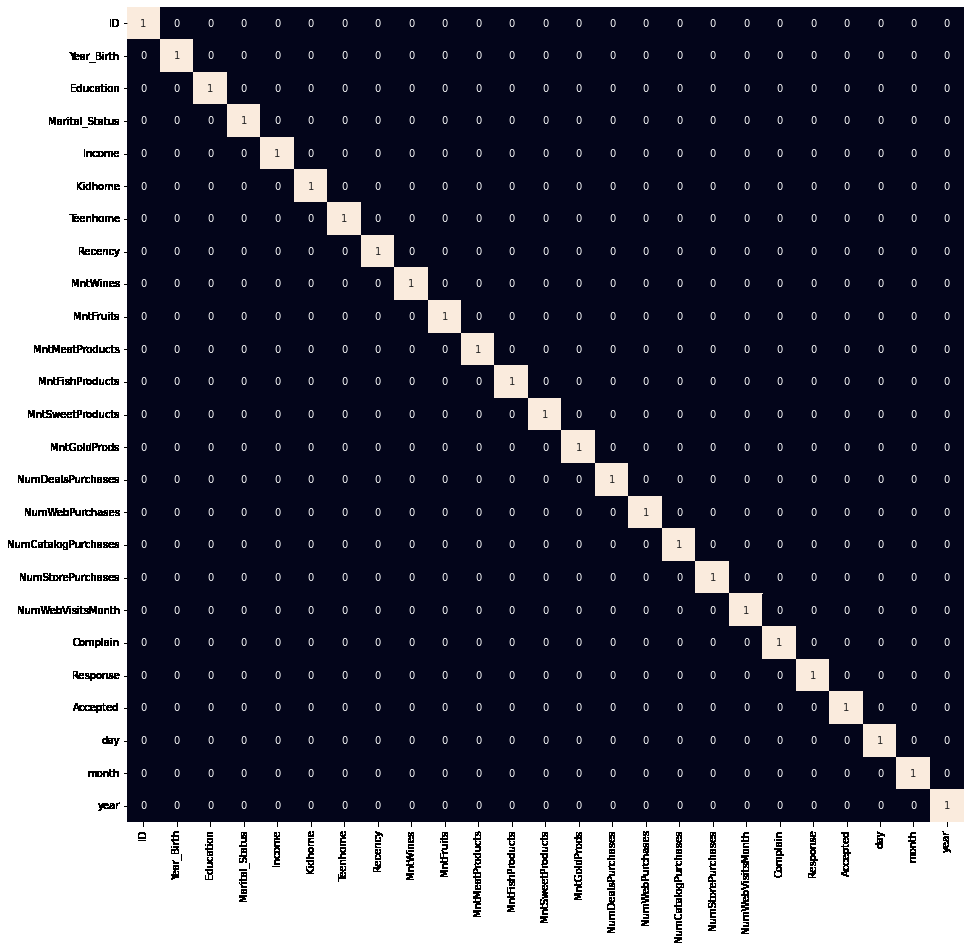
le = LabelEncoder()

df[col] = le.fit\_transform(df[col])

plt.figure(figsize=(15, 15))

sb.heatmap(df.corr() > 0.8, annot=True, cbar=False)

plt.show()



scaler = StandardScaler()

data = scaler.fit\_transform(df)

from sklearn.manifold import TSNE

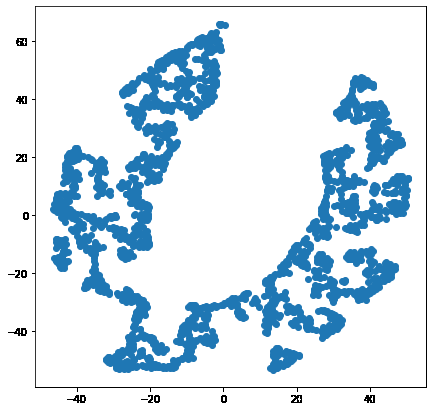
model = TSNE(n\_components=2, random\_state=0)

tsne\_data = model.fit\_transform(df)

plt.figure(figsize=(7, 7))

plt.scatter(tsne\_data[:, 0], tsne\_data[:, 1])

plt.show()



error = []

for n\_clusters in range(1, 21):

model = KMeans(init='k-means++',

n\_clusters=n\_clusters,

max\_iter=500,

random\_state=22)

model.fit(df)

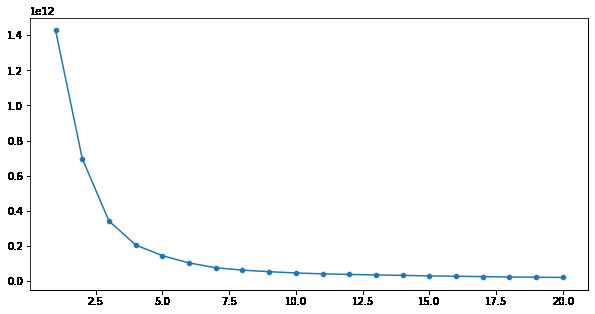
error.append(model.inertia\_)

plt.figure(figsize=(10, 5))

sb.lineplot(x=range(1, 21), y=error)

sb.scatterplot(x=range(1, 21), y=error)

plt.show()



# create clustering model with optimal k=5

model = KMeans(init='k-means++',

n\_clusters=5,

max\_iter=500,

random\_state=22)

segments = model.fit\_predict(df)

plt.figure(figsize=(7, 7))

sb.scatterplot(tsne\_data[:, 0], tsne\_data[:, 1], hue=segments)

plt.show()

