

Universidad Autónoma de Chiapas, Facultad de Negocios
Campus IV Tapachula Chiapas

Reporte del proyecto Kmeans para WillCityTap_Customer

Presenta:

b180022, Jose Colombio González Pérez

Grado y Grupo:

8vo semestre, 'D'

Materia:

Big-Data

Docente:

Dr. Christian Mauricio Castillo Estrada

Tapachula Chiapas a 23 de mayo de 2024

Introduccion

Este proyecto se enfoca en el análisis y la interpretación de datos para WallCityTap S.A., con el objetivo de respaldar la toma de decisiones estratégicas. Utilizando técnicas de clustering como K-means y herramientas como Shiny, hemos generado visualizaciones interactivas y análisis detallados que proporcionan información valiosa sobre el comportamiento de los consumidores, preferencias de pago y oportunidades de mercado. A través de este informe, se presentan los resultados obtenidos y se responden a los cuestionamientos clave de los directivos de la empresa.

Reporte del proyecto Kmeans para WillCityTap_Customer

En este proyecto estaré trabajando con UbuntuBuggy en su versión LTS 24.0.1, la implementación lo dividiremos en pasos para poder entender mejor y no perdernos:

Paso 1: Creación de la estructura del proyecto

Primero, creamos la estructura de directorios y archivos:

en nuestra terminal haremos uso del comando `mkdir` para la creación del proyecto, haremos lo siguiente:

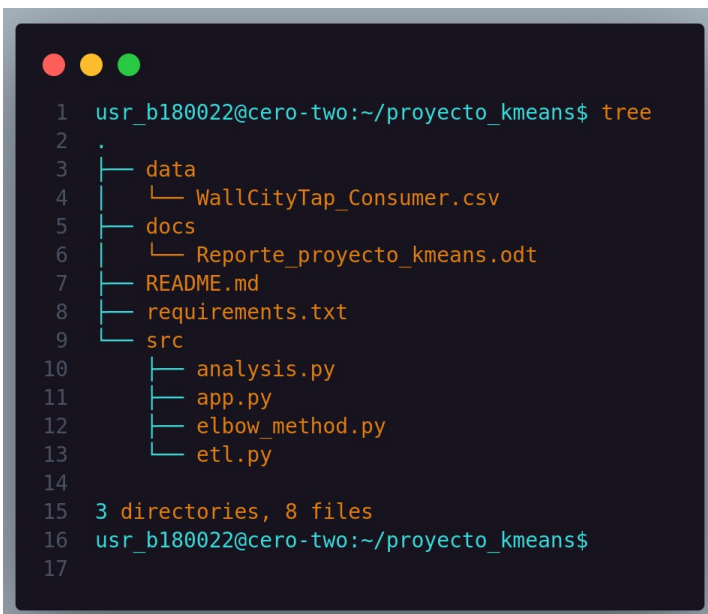
```
mkdir proyecto_kmeans
```

```
cd proyecto_kmeans
```

```
mkdir data src docs
```

```
touch data/WallCityTap_Consumer.csv src/app.py src/analysis.py  
src/etl.py src/elbow_method.py README.md requirements.txt
```

Nos quedaría de la siguiente manera:



```
1  usr_b180022@cero-two:~/proyecto_kmeans$ tree
2  .
3  ├── data
4  │   └── WallCityTap_Consumer.csv
5  ├── docs
6  │   └── Reporte_proyecto_kmeans.odt
7  ├── README.md
8  ├── requirements.txt
9  └── src
10     ├── analysis.py
11     ├── app.py
12     ├── elbow_method.py
13     └── etl.py
14
15  3 directories, 8 files
16  usr_b180022@cero-two:~/proyecto_kmeans$
17
```

Paso 2: Crear el archivo requirements.txt

Añadimos a nuestro proyecto recién creado las bibliotecas necesarias en requirements.txt:

```
pandas  
matplotlib  
seaborn  
scikit-learn  
dash  
shiny
```

Luego instalamos las dependencias con el comando:

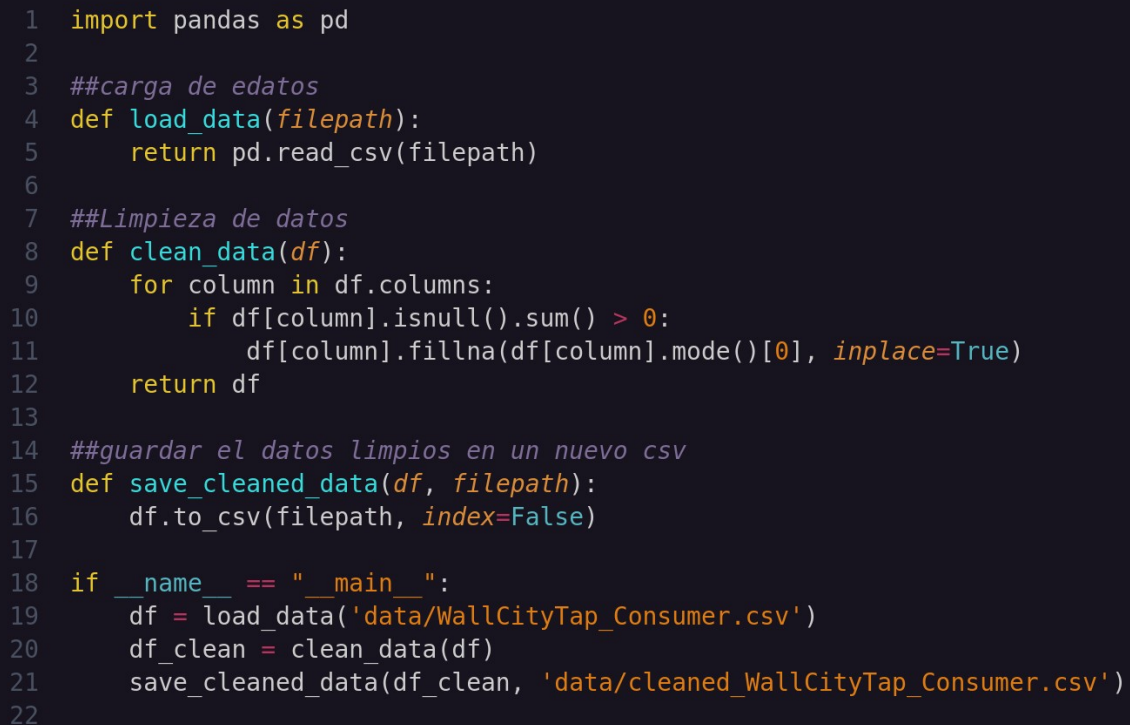
```
bash: pip install -r requirements.txt
```

con esto finalizamos el paso 2.

Paso 3: Escritura del script para la limpieza de datos (etl.py)

En src/etl.py, definimos las funciones para la carga, limpieza y guardado de datos:

En este paso se verifica si hay datos nulos o no validos se eliminan para poder generar su análisis correctamente.



```

1  import pandas as pd
2
3  ##carga de edatos
4  def load_data(filepath):
5      return pd.read_csv(filepath)
6
7  ##Limpieza de datos
8  def clean_data(df):
9      for column in df.columns:
10         if df[column].isnull().sum() > 0:
11             df[column].fillna(df[column].mode()[0], inplace=True)
12     return df
13
14 ##guardar el datos limpios en un nuevo csv
15 def save_cleaned_data(df, filepath):
16     df.to_csv(filepath, index=False)
17
18 if __name__ == "__main__":
19     df = load_data('data/WallCityTap_Consumer.csv')
20     df_clean = clean_data(df)
21     save_cleaned_data(df_clean, 'data/cleaned_WallCityTap_Consumer.csv')
22

```

De acuerdo a proyecto primero vamos a centrarnos en la identificación del valor óptimo de K utilizando el método Elbow Method para posteriormente aplicar el algoritmo K-means para guardar los resultados y los clusters de datos para después proceder a visualizarlos con Shiny, Esto implica realizarlo en 2 pasos, continuación se describe.

1. Identificar el valor óptimo de K utilizando el método del codo(Elbow_Method).
2. Aplicar el algoritmo K-means con el valor K óptimo y guardar los resultados.

Paso 4: Realizar análisis y aplicar el algoritmo K-means (analysis.py)

En este paso, crearemos dos script separados:

1. 'elbow_method.py': Para identificar el numero óptimo de clusters(K).
2. 'analysis.py': Para aplicar K-means utilizando el K óptimo y guardar los resultados.

Script para identificar el valor optimo de K (elbow_method.py)
creamos el nuevo script en src/elbow_method.py, y calculamos y graficamos la WCSS para diferentes valores de K :

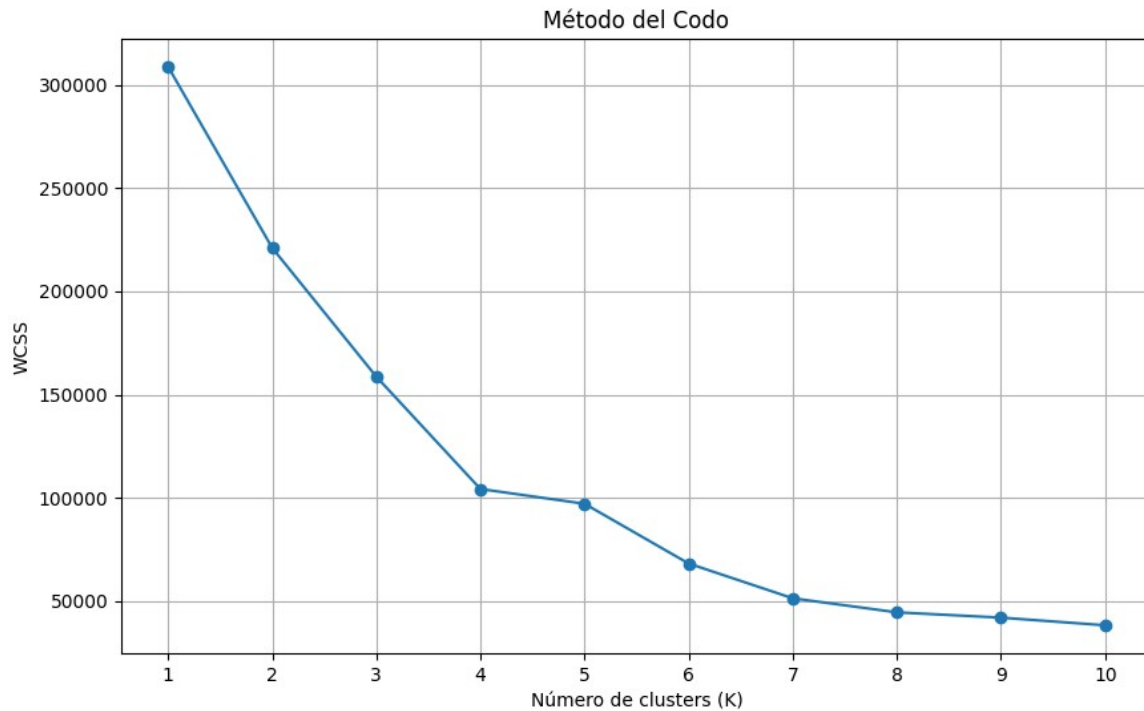
Método del codo (Elbow Method)

El método del codo implica:

1. Calcular WCSS (Within-Cluster Sum of Squares) para diferentes valores de K .
2. Graficar WCSS vs.

Identificar el "codo" en la gráfica, donde la tasa de disminución de WCSS se reduce notablemente. Este punto sugiere un buen valor de K .
Aquí se muestra la grafica del WCSS vs K . El cual nos ayuda a identificar visualmente el "codo".

[Nota]: Observar detalladamente el grafico donde la reducción de WCSS se vuelven menos significativa es el 'codo'.



Vamos a interpretar la grafica de la siguiente manera:

- ➔ En el eje x están los valores de K (numero de clústeres).
- ➔ En el eje y está el WCSS.
- ➔ Ahora se busca el punto donde la pendiente de la curva de vuelve menos pronunciada . En ese punto, significa que es el codo y se representa a el valor óptimo de K. Es decir con respecto al grafico anterior, el "codo" parece estar en es valor de $K = 3$ o 4 , pero puede variar dependiendo de los datos que tengamos.

Ahora les mostrare el script que implementa esto.

Script para aplicar K-means con el K óptimo (analysis.py)

Una vez identificado el valor óptimo de K (en este caso puede ser 3 o 4), utilizamos ese valor en 'src/analysis.py' para aplicar K-means y guardar los resultados en un nuevo csv para poder visualizarlo en el Dashboard usando Shiny.

En este paso el script analysis.py, Se cargó y analizamos los datos limpios, aplicando K-means para generar los clusters y guardar los resultados.

```
1 import pandas as pd
2 from sklearn.cluster import KMeans
3
4 # Cargar datos limpios
5 df = pd.read_csv('data/cleaned_WallCityTap_Consumer.csv')
6
7 # Definir el número óptimo de clusters (K) basado en el método del codo
8 optimal_k = 3 # Supongamos que el método del codo sugiere que 3 es el valor óptimo
9
10 # Aplicar K-means
11 kmeans = KMeans(n_clusters=optimal_k, random_state=42)
12 df['Cluster'] = kmeans.fit_predict(df[['Age', 'Annual_Income', 'Spending Score (1-100)']])
13
14 # Guardar los resultados del clúster
15 df.to_csv('data/WallCityTap_Consumer_clustered.csv', index=False)
16
```

Recordemos que el valor de K se selecciona visualmente a partir de la gráfica del método del codo, identificando el punto donde la disminución de WCSS se estabiliza. Este valor se actualiza en el script analysis.py, y luego se ejecuta para realizar el clustering con K-means para su posterior uso en el Dashboard de Shiny.

Paso 5: Generar Dashboard interactivo con Shiny for python (app.py)

Ahora que hemos identificado el valor óptimo de k y aplicado el algoritmo K-means, podemos proceder a diseñar el Dashboard dinámico utilizando Shiny para python. El cual debe visualizar el conjunto de datos en un componente DataTable, visualizar la aplicación del algoritmo K-means Clustering mediante un componente Plot, con sus respectivos controles de manipulación de entrada de datos.

Para ello es necesario instalar y agregar al archivo 'requirements.txt' una nueva librería llamada 'plotly'. Y también el paquete 'kaleido', que Plotly utiliza para exportar figuras a archivos de imagen para poderlo mostrar en el navegador, .

Usaremos el script 'src/app.py' que nos servirá como nuestro Dashboard interactivo, incluirá un componente DataTable para mostrar los datos y un componente Plot para visualizar los resultados del clustering K-means con un control para manipular el número de clústeres.

```

1 import pandas as pd
2 import plotly.express as px
3 from sklearn.cluster import KMeans
4 from shiny import App, ui, render, reactive
5 import tempfile
6 import os
7
8 # Cargar los datos limpios
9 df = pd.read_csv('data/WallCityTap_Consumer_clustered.csv')
10
11 # Definir la interfaz de usuario
12 app_ui = ui.page_fluid(
13     ui.h2("Dashboard de Clustering K-means"),
14     ui.layout_sidebar(
15         ui.panel_sidebar(
16             ui.input_slider("k_slider", "Número de clusters (K)", 3, 5, 3),
17             ui.input_action_button("run_analysis", "Ejecutar Análisis")
18         ),
19         ui.panel_main(
20             ui.div( ui.h2("Datos generales"),
21                 ui.output_table("data_table"),
22                 ui.output_image("cluster_plot"),
23             ),
24             ui.br(),
25             ui.h2("Graficos relacionados a las preguntas de los directivos"),
26             ui.output_image("age_plot"),
27             ui.output_image("payment_plot"),
28             ui.output_image("potential_customers_plot")
29         )
30     )
31 )
32
33 # Definir la lógica del servidor
34 def server(input, output, session):
35     @reactive.Calc
36     def clustered_data():
37         k = input.k_slider()
38         kmeans = KMeans(n_clusters=k, random_state=42)
39         df['Cluster'] = kmeans.fit_predict(df[['Age', 'Annual_Income', 'Spending Score (1-100)']])
40         return df
41
42     @output
43     @render.table
44     def data_table():
45         return clustered_data()
46
47     @output
48     @render.image
49     async def cluster_plot():
50         clustered_df = clustered_data()
51         fig = px.scatter(
52             clustered_df, x='Annual_Income', y='Spending Score (1-100)', color='Cluster',
53             hover_data=['Age'], title=f"Clustering K-means (K={input.k_slider()})"
54         )
55         fig.update_layout(legend_title="Clusters", legend=dict(orientation="h", yanchor="top", y=-0.1, xanchor="center", x=0.5))
56         with tempfile.NamedTemporaryFile(delete=False, suffix=".png") as tmpfile:
57             fig.write_image(tmpfile.name)
58             tmpfile.seek(0)
59             return {"src": tmpfile.name, "alt": "Cluster Plot - Gráfico de Clústeres"}
60
61     @output
62     @render.image
63     async def age_plot():
64         age_data = df[df['Age'] > 50]
65         fig = px.histogram(age_data, x='Spending Score (1-100)', nbins=20, title="Puntaje de Gasto de Clientes Mayores de 50 Años")
66         fig.update_layout(legend_title="Edad", legend=dict(orientation="h", yanchor="top", y=-0.1, xanchor="center", x=0.5))
67         with tempfile.NamedTemporaryFile(delete=False, suffix=".png") as tmpfile:
68             fig.write_image(tmpfile.name)
69             tmpfile.seek(0)
70             return {"src": tmpfile.name, "alt": "Age Plot - Gráfico de Edad"}
71
72     @output
73     @render.image
74     async def payment_plot():
75         payment_counts = df['Payment Methods'].value_counts().reset_index()
76         payment_counts.columns = ['Medio de Pago', 'Cantidad']
77         fig = px.pie(payment_counts, names='Medio de Pago', values='Cantidad', title="Métodos de Pago Preferidos")
78         with tempfile.NamedTemporaryFile(delete=False, suffix=".png") as tmpfile:
79             fig.write_image(tmpfile.name)
80             tmpfile.seek(0)
81             return {"src": tmpfile.name, "alt": "Payment Methods Plot - Gráfico de Métodos de Pago"}
82
83     @output
84     @render.image
85     async def potential_customers_plot():
86         clustered_df = clustered_data()
87         potential_customers = clustered_df[clustered_df['Cluster'] == clustered_df['Cluster'].mode()[0]]
88         fig = px.scatter(
89             potential_customers, x='Annual_Income', y='Spending Score (1-100)', color='Age',
90             title="Clientes Potenciales para Altas Ventas"
91         )
92         fig.update_layout(legend_title="Edad", legend=dict(orientation="h", yanchor="top", y=-0.1, xanchor="center", x=0.5))
93         with tempfile.NamedTemporaryFile(delete=False, suffix=".png") as tmpfile:
94             fig.write_image(tmpfile.name)
95             tmpfile.seek(0)
96             return {"src": tmpfile.name, "alt": "Potential Customers Plot - Gráfico de Clientes Potenciales"}
97
98 # Crear la aplicación Shiny
99 app = App(app_ui, server)
100
101 if __name__ == "__main__":
102     app.run()
103

```

cuestionamientos de los directivos de WallCityTap S.A.

Análisis: Filtrar los datos para consumidores respecto lo indicado.

1. ¿Es viable financieramente lanzar una promoción del 2x1 + meses a interés para los consumidores con una edad mayor a 50 años?

En la gráfica de puntaje de gasto de clientes mayores de 50 años, observamos que los puntos de mayor concentración se encuentran en los niveles de puntaje de gasto más bajos (hasta el 7 en el eje y). Esto sugiere que los consumidores mayores de 50 años tienden a gastar menos en comparación con otros grupos de edad. Por lo tanto, una promoción como la mencionada podría no ser tan atractiva para este segmento de clientes, ya que podrían tener un menor interés en realizar compras adicionales.

2. ¿Cuál es el medio de pago preferido de nuestros consumidores?

Según el gráfico de métodos de pago preferidos, el 44.5% de los consumidores utiliza efectivo (Cash) como su método de pago preferido, seguido por el 38% que utiliza tarjeta de crédito (Tcredit) y el 17% que utiliza tarjeta de débito (Tdebit). Esto indica que el efectivo sigue siendo el medio de pago más utilizado entre nuestros clientes, seguido de cerca por las tarjetas de crédito.

3. Identificar y describir los consumidores potenciales, para generar mayores ventas y obtener mayores ingresos.

En la gráfica de clientes potenciales para altas ventas, observamos que hay una mayor concentración de clientes en el grupo de edad de 50

a 55 años (color naranja), lo que sugiere que este segmento de clientes puede representar un potencial de ventas considerable. Además, notamos que los grupos de edad más jóvenes (20 años, color azul) y los de edad más avanzada (400 años, color marrón) tienen una menor presencia en términos de puntos en el gráfico, lo que podría indicar áreas donde se podrían enfocar esfuerzos para captar más clientes y aumentar las ventas. Por otro lado, el grupo de edad de 70 años (color amarillo) muestra una presencia más limitada, lo que podría indicar una menor oportunidad de ventas en este segmento específico.

Conclusión

En conclusión, el análisis realizado proporciona información valiosa que respalda la toma de decisiones estratégicas para el desarrollo de nuevas campañas de productos y ofertas dirigidas a los consumidores de WallCityTap S.A. El uso del algoritmo K-means nos permitió identificar patrones de comportamiento en los datos de los consumidores, lo que a su vez nos ayudó a segmentarlos en grupos significativos. Estos grupos nos brindan una comprensión más profunda de las preferencias y características de nuestros clientes, lo que a su vez nos permite diseñar estrategias de marketing más efectivas y personalizadas.

Además, la implementación de un Dashboard dinámico a través de Shiny nos proporciona una plataforma interactiva para visualizar y analizar los datos de manera eficiente. Las diferentes visualizaciones generadas, como el gráfico de dispersión para el análisis de

clusters, el histograma de puntaje de gasto para clientes mayores de 50 años, el gráfico de pastel de métodos de pago preferidos, y el gráfico de dispersión de clientes potenciales, ofrecen una visión completa y detallada de nuestro mercado objetivo y las oportunidades potenciales de crecimiento.

Estructura Final:

```
proyecto_kmeans.  
├─ data  
│   ├── cleaned_WallCityTap_Consumer.csv  
│   ├── WallCityTap_Consumer_clustered.csv  
│   └─ WallCityTap_Consumer.csv  
├─ docs  
│   ├── img_codes  
│   │   ├── code_analysis.png  
│   │   ├── code_CreatedProjectbash.png  
│   │   ├── code_etl.png  
│   │   ├── code_grafico_cueestionamientos.png  
│   │   └─ Figure_1_elbowMethod.png  
│   └─ Reporte_proyecto_kmeans.odt  
├─ hola.bash  
├─ README.md  
├─ requirements.txt  
└─ src  
    ├── analysis.py  
    ├── app.py  
    ├── elbow_method.py  
    ├── etl.py  
    └─ __pycache__  
        └─ app.cpython-310.pyc
```

5 directories, 17 files

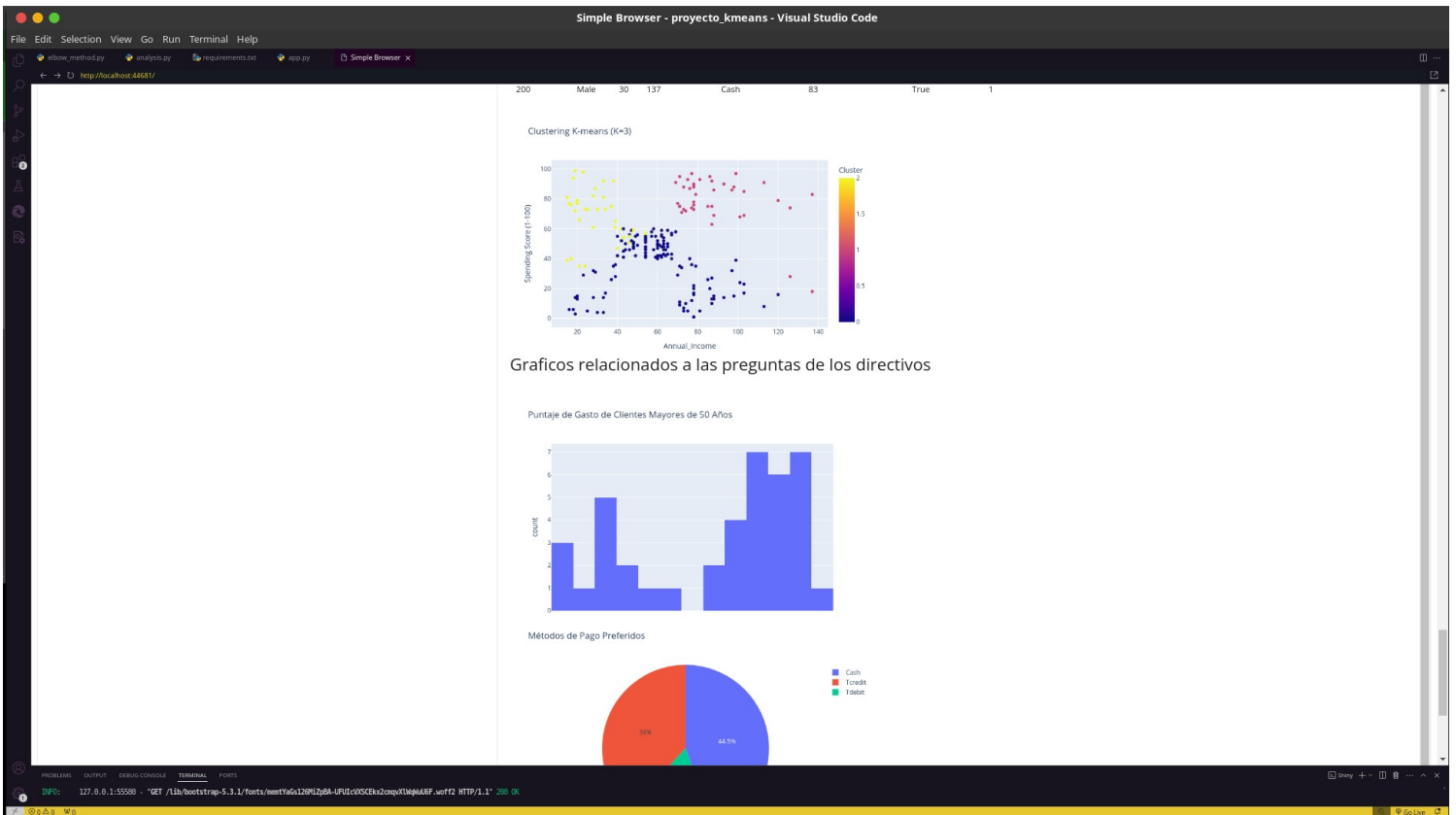
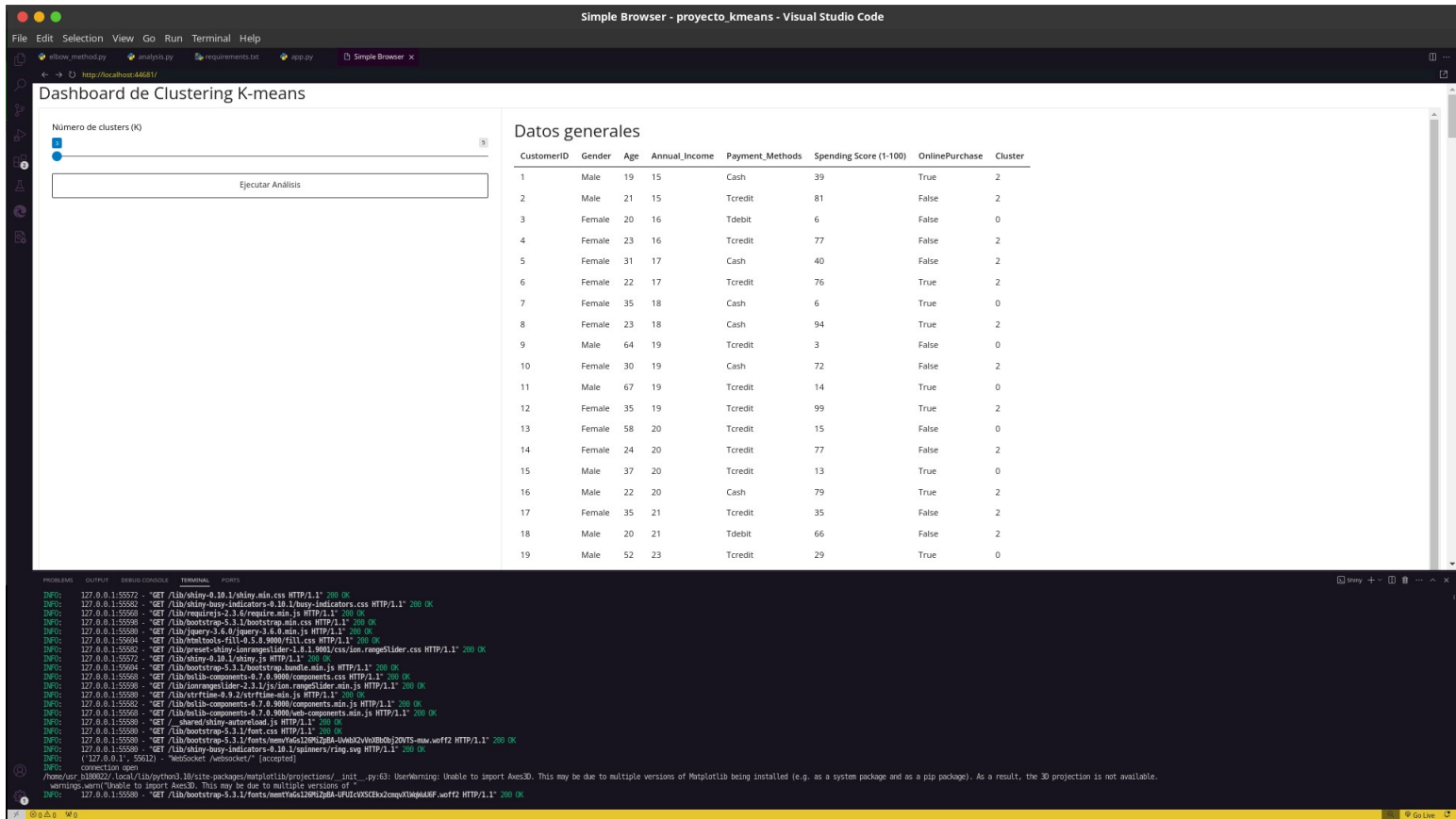
Descargar Archivo o proyecto de mi repositorio de GitHub:

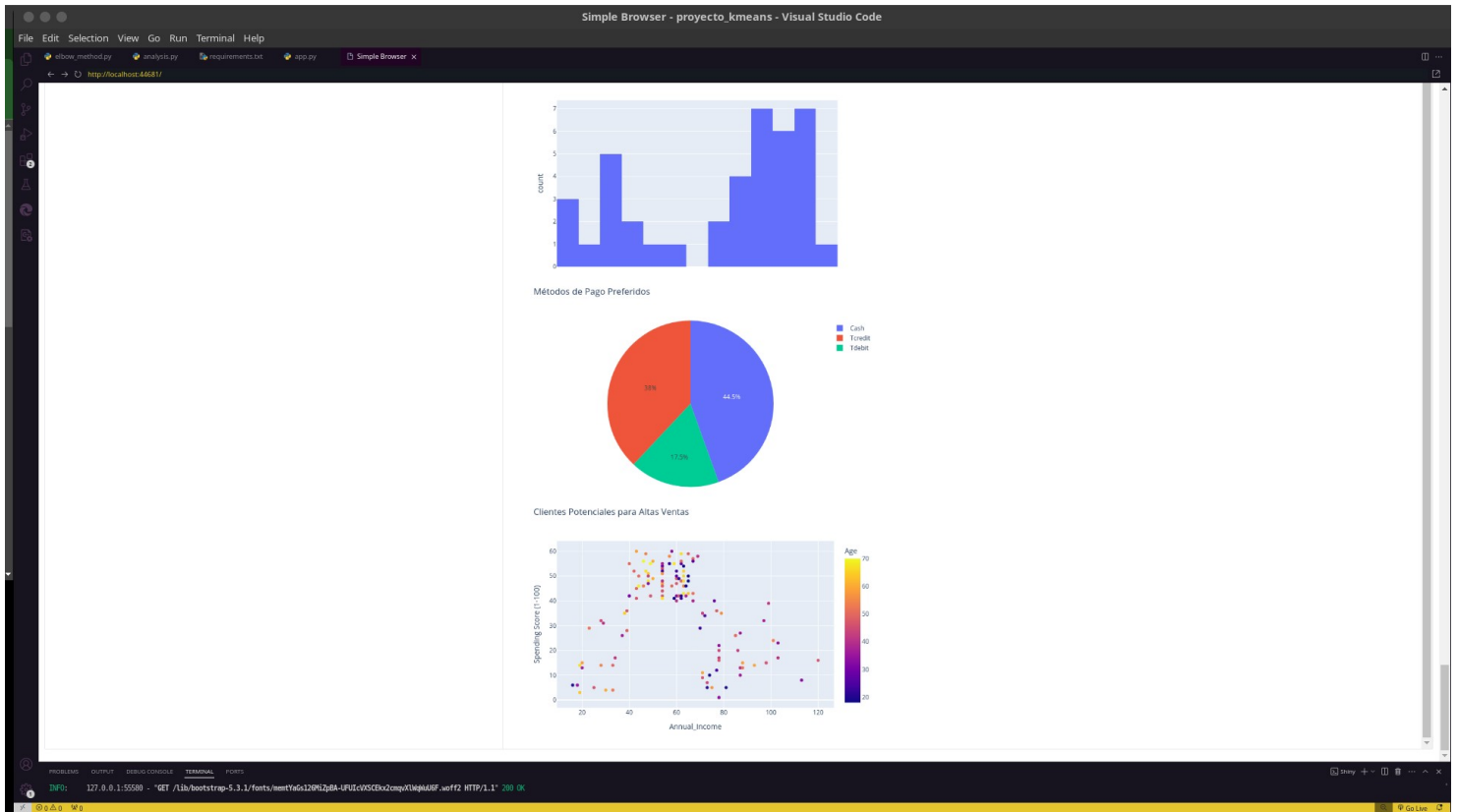
Guardare la practica en una carpeta llamada proyecto_kmeans.

Dependiendo del numero de practicas.

En este caso la guardare en proyecto Final. En el repositorio de github el cual contiene todo el código y su requerimientos:

Anexos





Referencias y tutoriales:

<https://shiny.posit.co/r/gallery/start-simple/kmeans-example/>

<https://github.com/posit-dev/py-shiny-templates/tree/main/dashboard>

<https://towardsdatascience.com/k-means-data-clustering-bce3335d2203>

<https://www.aprendemachinelearning.com/k-means-en-python-paso-a-paso/>

<https://rpubs.com/rdelgado/399475>

<https://rpubs.com/Dariel1102/1046632>

<https://shiny.posit.co/py/docs/ui-html.html>