UTPL

MAESTRIA EN INTELIGENCIA ARTIFICIAL APLICADA

Herramientas para la inteligencia Artificial

Tema: Inspections Restaurants San Francisco CA

Integrantes:

- ✓ Edwin Machado
- ✓ Jose Masache
- ✓ Raul Guevara

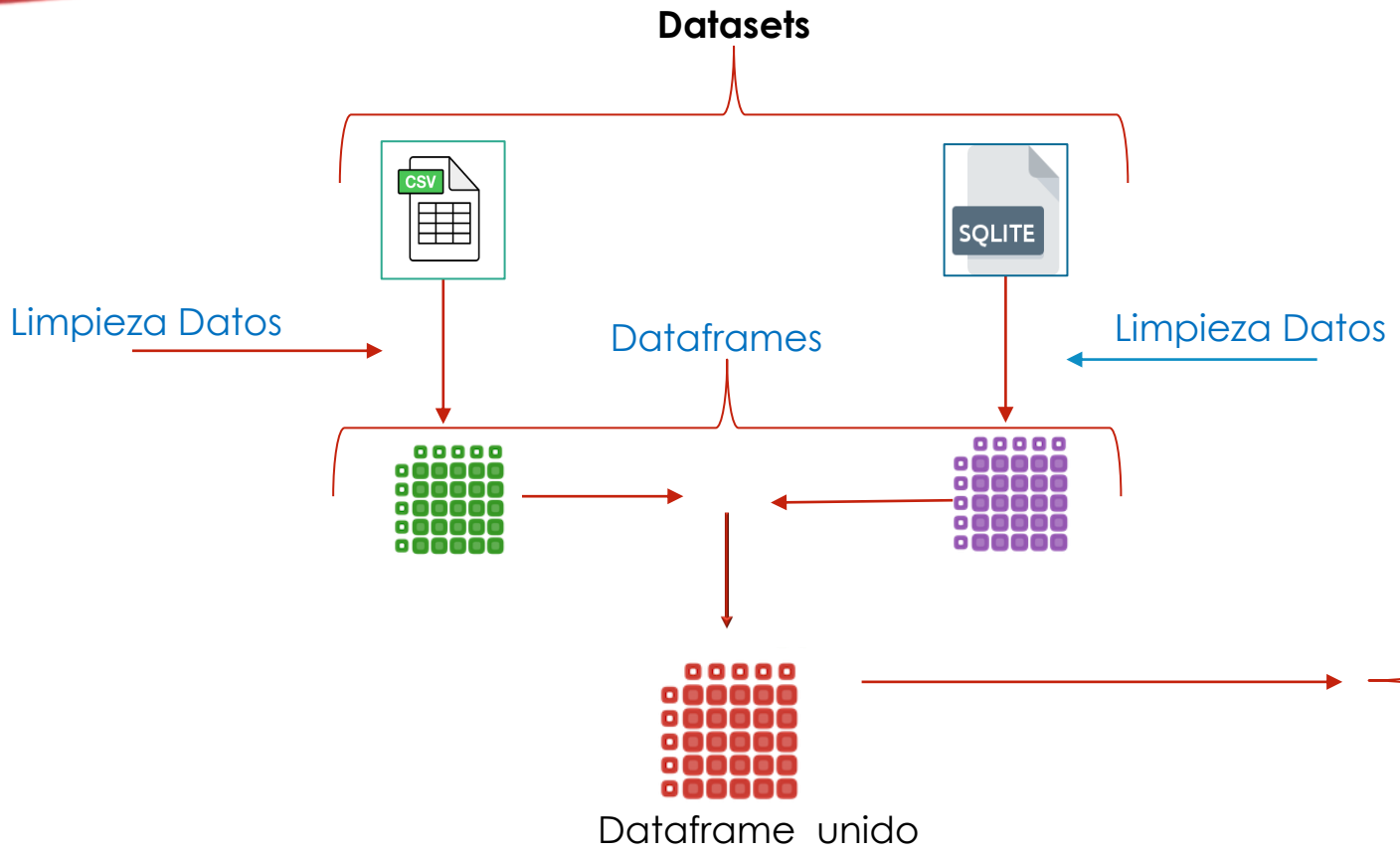
INSPECTIONS RESTAURANTS SAN FRANCISCO CA

Introducción

En el presente trabajo se hace un análisis de datos para inteligencia artificial utilizando librerías de software libre y herramientas colaborativas. El proyecto se fundamenta en el uso de dos conjuntos de datos de diferente tipo: un archivo CSV llamado "Restaurant_Scores_-_LIVES_Standard" y un archivo de tipo SQLite llamado "sfscores.sqlite". Estos archivos corresponden a inspecciones sanitarias en restaurantes de la ciudad de San Francisco, California.

Mediante procesos de limpieza y depuración de los datos, se consolidaron ambos conjuntos en un solo archivo denominado "Restaurant.csv". Este archivo unificado sirvió como base para la creación de diversas visualizaciones que facilitan la interpretación de los resultados, permitiendo obtener información valiosa sobre el estado de los restaurantes y sus prácticas de higiene.

Esquema del proceso



Visualizaciones

matplotlib

bokeh

PyGWalker

INSPECTIONS RESTAURANTS SAN FRANCISCO CA

Marco teórico de las tecnologías/librerías usadas

La plataforma utilizada para el desarrollo del presente trabajo se basó en Jupyter Notebook-entorno local. Jupyter Notebook es un entorno de desarrollo interactivo que permite ejecutar código en vivo a través del navegador web.

Librerías Utilizadas:

- SQLite.-Es una biblioteca de software que proporciona una base de datos relacional ligera y autocontenida.
- Pandas.-Es una librería de Python para la manipulación y análisis de datos.
- Numpy.-Es una librería de Python para computación científica.
- Seaborn.-Es una librería de visualización de datos basada en Matplotlib
- MatPlotlib.-Es una librería de Python para la creación de gráficos estáticos, animados e interactivos.
- Bokeh.-Es una librería de visualización de datos interactiva que permite crear gráficos y dashboards atractivos en el navegador web.
- PyGWalker.-Es una librería de Python diseñada para facilitar la creación de visualizaciones de datos interactivas y exploratorias.

INSPECTIONS RESTAURANTS SAN FRANCISCO CA

Descripción de los datasets usados

El dataset SQLITE(sfscores.sqlite)utilizado en este trabajo proporciona una visión detallada de las inspecciones sanitarias realizadas en los negocios de restaurantes. Cada fila del conjunto de datos representa una inspección específica, identificada por un business_id único que permite rastrear el historial de inspecciones de cada negocio. Además de la identificación del negocio, el dataset incluye información sobre el tipo de violación al reglamento sanitario encontrada durante la inspección, el nivel de riesgo asociado y una descripción detallada.

	business_id	ViolationTypeID	risk_category	description
0	10	103129	Moderate Risk	Insufficient hot water or running water
1	10	103144	Low Risk	Unapproved or unmaintained equipment or utensils
2	10	103119	Moderate Risk	Inadequate and inaccessible handwashing facili...
3	10	103145	Low Risk	Improper storage of equipment utensils or linens
4	10	103154	Low Risk	Unclean or degraded floors walls or ceilings
...	...	...	...	...
40730	89072	103131	Moderate Risk	Moderate risk vermin infestation

INSPECTIONS RESTAURANTS SAN FRANCISCO CA

Descripción de los datasets usados

El conjunto de datos CSV(Restaurant_Scores_-_LIVES_Standard.csv) con una estructura organizada y detallada, este conjunto de datos abarca una amplia gama de información sobre los negocios gastronómicos que operan en la ciudad de San Francisco CA. Cada fila representa un negocio individual, identificado por un número único y acompañado de datos clave como el nombre del establecimiento.

	business_id	business_name	business_address	business_city	business_state	business_postal_code
0	835	Kam Po Kitchen	801 Broadway St	San Francisco	CA	94133
1	905	Working Girls' Cafe'	0259 Kearny St	San Francisco	CA	94108
2	1203	TAWAN'S THAI FOOD	4403 GEARY Blvd	San Francisco	CA	94118

INSPECTIONS RESTAURANTS SAN FRANCISCO CA

Descripción de los pasos realizados en el proyecto

Recopilación de datos: El primer paso es recopilar los datos "Restaurant_Scores_-_LIVES_Standard.csv" y sfscores.sqlite. Ambos archivos se escogieron debido a su relación en un identificador único para los negocios de comida en San Francisco.

Restaurant_Scores_-_LIVES_Standard.csv

sfscores.sqlite

INSPECTIONS RESTAURANTS SAN FRANCISCO CA

Descripción de los pasos realizados en el proyecto

Exploración de datos: Se usa lagoritmos basados en librerías Python para explorar la data tanto del archivo sqlite como el archivo CSV

```
import numpy as np
import pandas as pd
import sqlite3
import matplotlib.pyplot
import seaborn as sns
```

```
connection = sqlite3.connect(database)
# Obtener el nombre de todas las tablas en la base de datos
tables = pd.read_sql("""SELECT * FROM sqlite_master WHERE type = 'table';""", connection)
# Iterar sobre cada tabla y mostrar todas las filas
for table_name in tables['name']:
    # Leer todas las filas de la tabla actual
    table_data = pd.read_sql(f"SELECT * FROM {table_name};", connection)
    print(f"Tabla: {table_name}")
    print(table_data)
    print("\n")
    table_data
```

```
query = "SELECT * FROM inspections "
df = pd.read_sql_query(query, connection)
df
```

INSPECTIONS RESTAURANTS SAN FRANCISCO CA

```
#leyendo archivo csv
df3=pd.read_csv("Restaurant_Scores_-_LIVES_Standard.csv")
#Convert to DateTime
df3["inspection_date"]=pd.to_datetime(df3["inspection_date"])
#Drop unnecessary columns
df3 = df3.drop(["Analysis Neighborhoods","Current Supervisor Districts","Current Police Districts","SF Find Neighborhc
#Limpieza de columnas innecesarias
df3=df3.drop(['business_latitude','business_longitude','business_postal_code','business_location','violation_descripti
#limpieza de datos vacìos
df3.replace(r'^\s*$', np.nan, regex=True, inplace=True)
print("\nDataFrame después de reemplazar valores vacíos por NaN:")
#borrar valores nulos
df3=df3.dropna() #limpiando filas que tengan valores nulos
df3=df3.drop_duplicates() #limpiando filas que tengan valores duplicados
df3
```

INSPECTIONS RESTAURANTS SAN FRANCISCO CA

Limpieza de dataframes: En este proceso se realiza la limpieza de los dataframes generados de cada uno de los datasets.Esta limpieza es de valores nulls, y valores repetidos. Donde al dataframe del dataset sqlite llamaremos df2 y el dataset del archivo csv llamaremos df3. Además de la limpieza se borraron columnas del dataframe df2 y df3 ya tenían valores nulls.

```
#limpieza de datos vacìos
df2.replace(r'^\s*$', np.nan, regex=True, inplace=True)
print("\nDataFrame después de reemplazar valores vacíos por NaN:")
#borrar valores nulos
df2=df2.dropna() #limpiando filas que tengan valores nulos
df2=df2.drop(['date'], axis=1)#borrar columna date
df2=df2.drop_duplicates() #limpiando filas que tengan valores duplicados
df2
```

```
#Limpieza de columnas innecesarias
df3=df3.drop(['business_latitude','business_longitude','business_postal_code','business_location','violation_descripti
#limpieza de datos vacìos
df3.replace(r'^\s*$', np.nan, regex=True, inplace=True)
print("\nDataFrame después de reemplazar valores vacíos por NaN:")
#borrar valores nulos
df3=df3.dropna() #limpiando filas que tengan valores nulos
df3=df3.drop_duplicates() #limpiando filas que tengan valores duplicados
df3
```

```
df3
```

INSPECTIONS RESTAURANTS SAN FRANCISCO CA

Union Dataframes: Nos disponemos a unir los dos dataframes, previo a esto hacemos la conversión a formato string de la columna *"business_id"* con e fin de que la unión con merge se haga efectiva y de acuerdo a esta columna, ya que es común y que funciona como clave primaria. Esta sección del código asegura que la columna *business_id* en ambos DataFrames (df2 y df3) sea del tipo string. Esto es necesario para evitar problemas de compatibilidad durante la unión de los DataFrames.

```
# Convertir la columna 'business_id' a tipo string en ambos DataFrames
df2['business_id'] = df2['business_id'].astype(str)
df3['business_id'] = df3['business_id'].astype(str)`
# Realizar la unión del archivo SQLITE y CSV
dfu = pd.merge(df2, df3, on='business_id', how='right')
dfu
```

INSPECTIONS RESTAURANTS SAN FRANCISCO CA

	business_id	ViolationTypeID	risk_category	description	business_name	business_address	bu
0	835	103139	Low Risk	Improper food storage	Kam Po Kitchen	801 Broadway St	Sa Fr
1	835	103133	Moderate Risk	Foods not protected from contamination	Kam Po Kitchen	801 Broadway St	Sa Fr
2	835	103144	Low Risk	Unapproved or unmaintained equipment or utensils	Kam Po Kitchen	801 Broadway St	Sa Fr

INSPECTIONS RESTAURANTS SAN FRANCISCO CA

Limpieza de dataframe unido: En etapa lo que hacemos es limpiar el dataframe de nulls y duplicados, eso producto de la unión de los dos datasets, donde solamente se anexaron los que tenían el mismo “Business_id”, mientras que los otros quedan fuera, quedando nulls o repetidos.

```
#Limpieza de datos vacios
dfu.replace(r'^\s*$', np.nan, regex=True, inplace=True)
print("\nDataFrame después de reemplazar valores vacíos por NaN:")
#borrar valores nulos
dfu=dfu.dropna() #limpiando filas que tengan valores nulos
dfu=dfu.drop_duplicates('business_id') #limpiando filas que tengan valores duplicados
dfu
```

INSPECTIONS RESTAURANTS SAN FRANCISCO CA

Dataframe final

	business_id	ViolationTypeID	risk_category	description	business_name	business_address	bu
0	835	103139	Low Risk	Improper food storage	Kam Po Kitchen	801 Broadway St	Sa Fra
14	905	103162	Low Risk	Other low risk violation	Working Girls' Cafe'	0259 Kearny St	Sa Fra
18	1203	103154	Low Risk	Unclean or degraded floors walls or ceilings	TAWAN'S THAI FOOD	4403 GEARY Blvd	Sa Fra
22	1345	103132	Moderate Risk	Improper thawing methods	Cordon Bleu	1574 California St	Sa Fra
				Unapproved or			

INSPECTIONS RESTAURANTS SAN FRANCISCO CA

Anexo de columnas: Anexamos columnas en base a la columnas “inspection_date” y “risk_category”

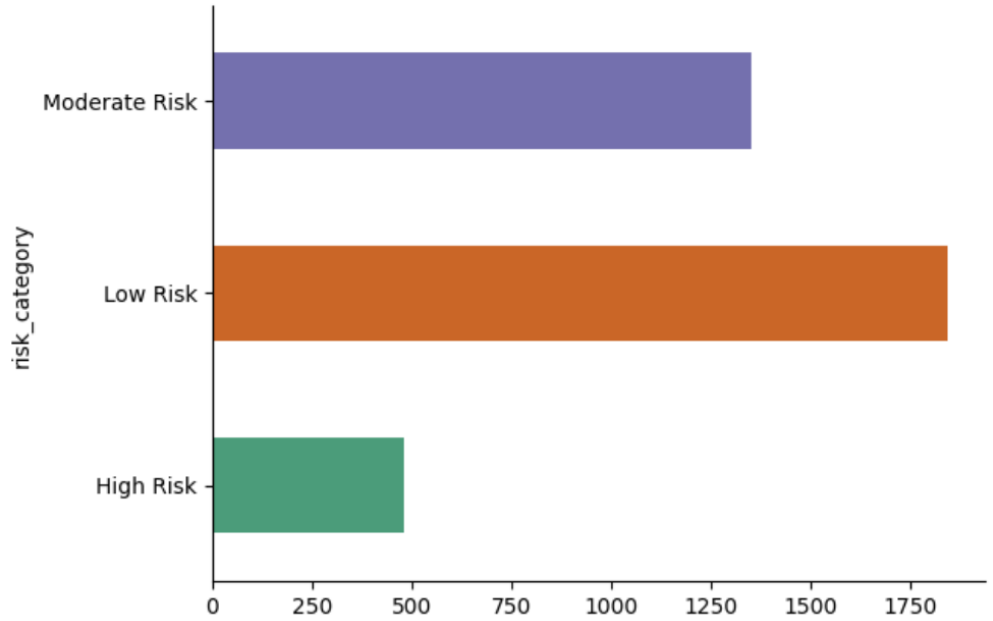
```
#se agrega nuevas columnas en funcion al dataframe
dfu['mes'] = dfu['inspection_date'].dt.strftime('%B')#se muestra el mes de la inspeccion
dfu['día'] = dfu['inspection_date'].dt.strftime('%A')#se muestra el dia de la inspeccion
dfu['año'] = dfu['inspection_date'].dt.strftime('%Y')#se muestra el año de la inspeccion
dfu['Aprobado'] = dfu['risk_category'].apply(lambda x: x == 'Low Risk')#Evalua si aprueba de acuerdo al riesgo
dfu['Observación'] = dfu['risk_category'].apply(lambda x: x == 'Moderate Risk')#Evalua si està en observacion de acuer
dfu['Clausurado'] = dfu['risk_category'].apply(lambda x: x == 'High Risk')#Evalua la clausura de acuerdo al riesgo
dfu
```

	inspection_date	inspection_type	mes	día	año	Aprobado	Observación	Clausurado
	2018-09-17	Routine - Unscheduled	September	Monday	2018	True	False	False
	2019-04-15	Routine - Unscheduled	April	Monday	2019	True	False	False
3	2017-08-03	Routine - Unscheduled	August	Thursday	2017	True	False	False
3	2017-09-28	Routine - Unscheduled	September	Thursday	2017	False	True	False

INSPECTIONS RESTAURANTS SAN FRANCISCO CA

Visualización con Matplotlib - Gráfico de Barras

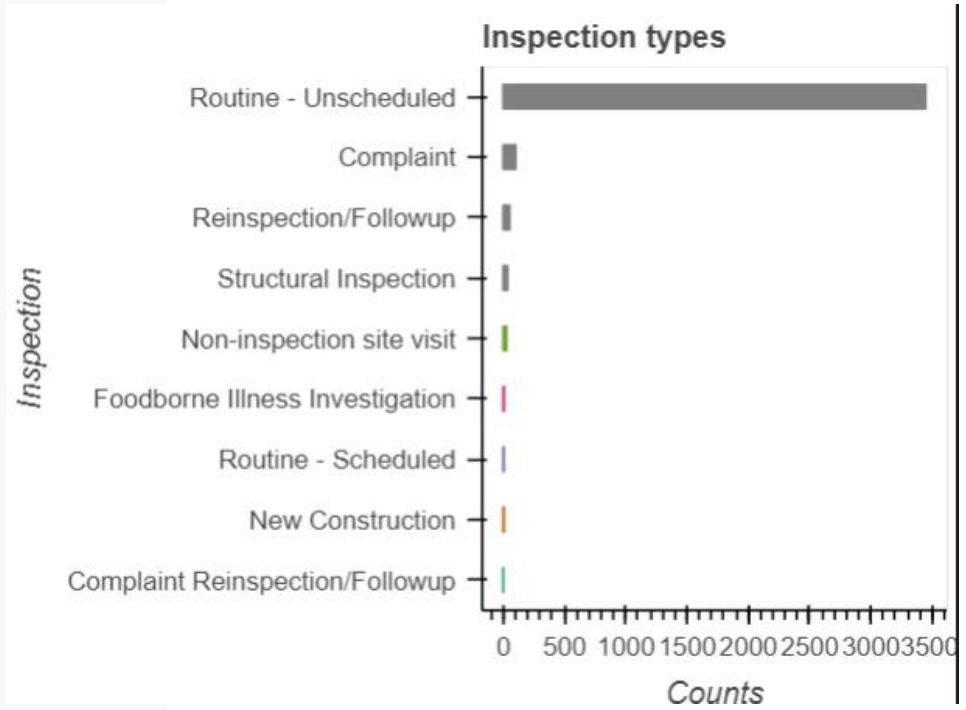
```
# Visualizaciones en Matplotlib
from matplotlib import pyplot as plt
import seaborn as sns
dfu.groupby('risk_category').size().plot(kind='barh', color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```



INSPECTIONS RESTAURANTS SAN FRANCISCO CA

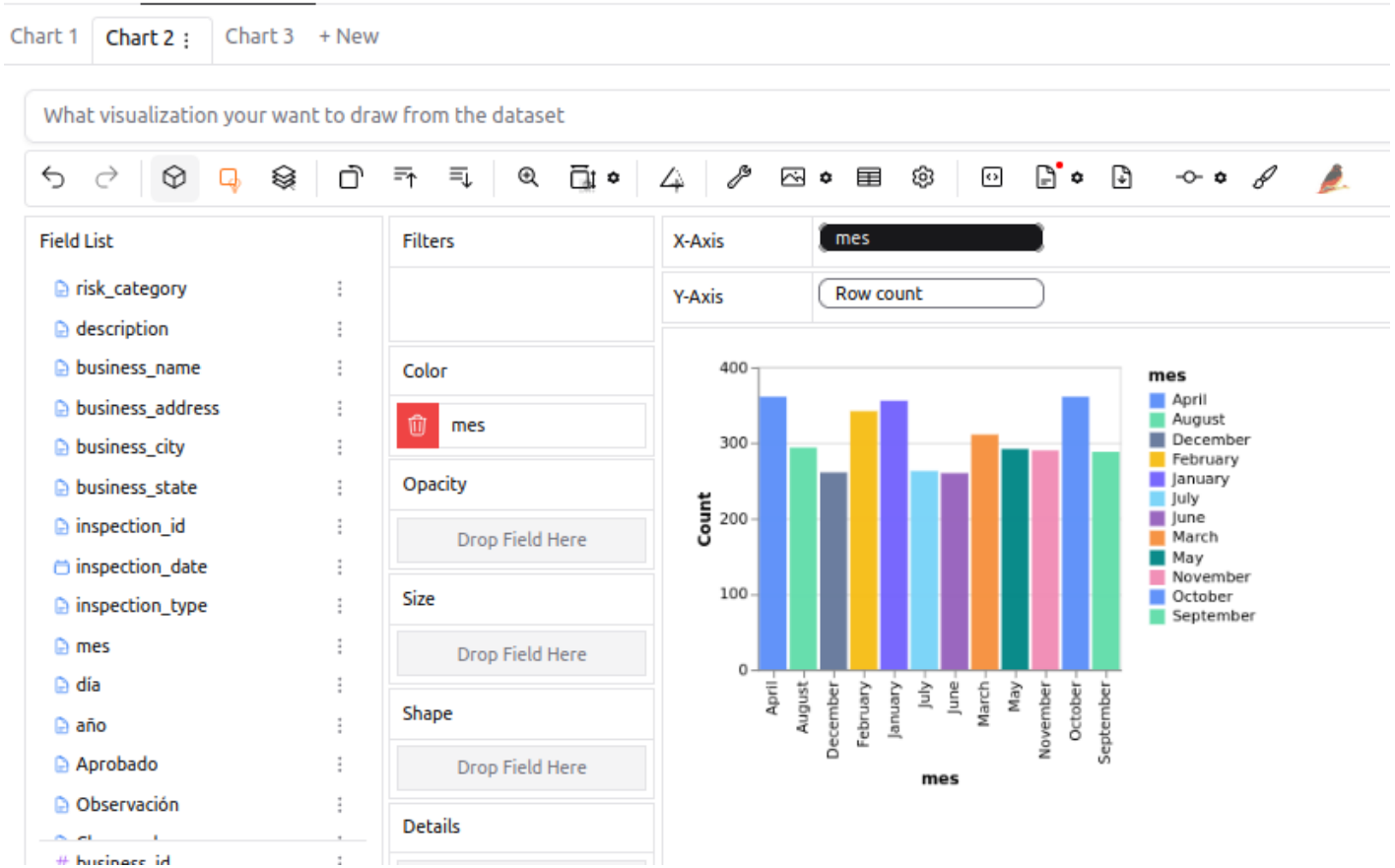
Visualización con Bokeh - Barras Horizontales

```
import pandas as pd
from bokeh.plotting import figure, show
from bokeh.io import output_notebook
from bokeh.transform import factor_cmap
from bokeh.palettes import Dark2_5 as palette
from bokeh.models import ColumnDataSource
# Activar la salida de Bokeh en el notebook
output_notebook()
# dfu = archivo unido
# Agrupar por la columna 'risk_category' y contar
grouped_data = dfu.groupby('inspection_type').size().reset_index(name='counts')
grouped_data = grouped_data.sort_values('counts', ascending=True)
# Convertir los datos a un ColumnDataSource
source = ColumnDataSource(grouped_data)
# Crear la lista de categorías ordenadas
categories = grouped_data['inspection_type'].tolist()
# Crear la visualización de barras horizontales
p1 = figure(y_range=categories, height=300, width=400, title="Inspection types",
           x_axis_label='Counts', y_axis_label='Inspection', toolbar_location=None)
# Agregar las barras al gráfico
p1.hbar(y='inspection_type', right='counts', height=0.4, source=source,
       color=factor_cmap('inspection_type', palette=palette, factors=categories))
# Eliminar las líneas de la parte superior y derecha
p1.outline_line_color = None
p1.ygrid.grid_line_color = None
p1.xgrid.grid_line_color = None
# Mostrar el gráfico
show(p1)
```



INSPECTIONS RESTAURANTS SAN FRANCISCO CA

Visualizaciones con Pygwalker



INSPECTIONS RESTAURANTS SAN FRANCISCO CA

Conclusiones:

- Se ha logrado exitosamente la unión de un archivo SQLite y un archivo CSV utilizando la columna “Business_id" como clave de unión. Esto sugiere que el proceso de combinación de datos fue efectivo y se logró integrar la información de ambos conjuntos de datos de manera coherente. En nuestro caso, la relación del campo "Business_id" nos ayudó a concatenar el campo "Inspection_id" y saber con su código cuáles restaurantes habían sido inspeccionados y los resultados que obtuvieron luego del proceso de inspección.

- Se utilizaron diversas librerías como Pandas, NumPy y Matplotlib para llevar a cabo el proceso de limpieza de datos y la unión de archivos. Esto indica que se aplicó un enfoque técnico y estructurado para manipular los conjuntos de datos, lo que podría haber mejoró la eficiencia y precisión del trabajo. Se menciona que se realizó una limpieza adecuada de los conjuntos de datos, lo que implica que se identificaron y corrigieron posibles errores, inconsistencias o datos faltantes.

- Se añadieron nuevas columnas al conjunto de datos final, incluyendo "día", "mes" y "año", así como evaluaciones de aprobación específicas para los diferentes restaurantes de San Francisco. Esto sugiere que se enriqueció el conjunto de datos con información adicional relevante para el análisis posterior.

- De acuerdo a las gráficas, se puede concluir que el estudio realizado muestra un análisis del riesgo que cada restaurante arroja después de la inspección. Esto revela que la mayoría de restaurantes tienen un bajo riesgo, por lo tanto, aprueban la inspección. Sin embargo, hay algunos en menor cantidad que requieren hacer cambios para mantenerse en operatividad. Mientras que un bajo índice muestra que no pueden seguir operando debido a sus niveles de riesgo de inocuidad. Este análisis proporciona una comprensión clara de la distribución del riesgo entre los restaurantes inspeccionados y sugiere áreas específicas donde se pueden enfocar las medidas correctivas para mejorar la seguridad alimentaria.

INSPECTIONS RESTAURANTS SAN FRANCISCO CA

Bibliografia:

SQLite
Hipp, D. R. (2000). SQLite. Recuperado de https://www.sqlite.org/

Pandas
McKinney, W. (2010). Data Structures for Statistical Computing in Python. En Proceedings of the 9th Python in Science Conference (pp. 51-56). DOI: 10.25080/Majora-92bf1922-00a

NumPy
Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357-362. DOI: 10.1038/s41586-020-2649-2

Seaborn
Waskom, M. L. (2021). seaborn: statistical data visualization. Journal of Open Source Software, 6(60), 3021. DOI: 10.21105/joss.03021

Matplotlib
Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 9(3), 90-95. DOI: 10.1109/MCSE.2007.55

Bokeh
Bokeh Development Team. (2018). Bokeh: Python library for interactive visualization. Recuperado de https://bokeh.org/

PyGWaker (Panel)
HoloViz. (2021). Panel: A high-level app and dashboarding solution for Python. Recuperado de https://panel.holoviz.org/