sketch-demo

February 8, 2024

[ ]:
```
! pip install sketch
```

[40]:
```
import sketch
import pandas as pd
import plotly.express as px
```

[52]:
```
import seaborn as sns
sns.set_style("darkgrid")
```

[43]:
```
sales_data = pd.read_csv("https://gist.githubusercontent.com/bluecoconut/
 ↪9ce2135aafb5c6ab2dc1d60ac595646e/raw/
 ↪c93c3500a1f7fae469cba716f09358cfddea6343/sales_demo_with_pii_and_all_states.
 ↪csv")
```

[44]:
```
sales_data.columns
```

[44]:
```
Index(['Order ID', 'Product', 'Quantity Ordered', 'Price Each', 'Order Date',
       'Purchase Address', 'Credit Card', 'SSN'],
      dtype='object')
```

[45]:
```
sales_data.head()
```

[45]:
```
   Order ID                   Product  Quantity Ordered  Price Each  \
0    141234                    iPhone               1.0      700.00
1    141235  Lightning Charging Cable               1.0       14.95
2    141236          Wired Headphones               2.0       11.99
3    141237          27in FHD Monitor               1.0      149.99
4    141238          Wired Headphones               1.0       11.99

       Order Date                                   Purchase Address  \
0  01/22/19 21:25  10995 Williams Cliffs, East Michelleborough, A…
1  01/28/19 14:15     1067 Guzman View Suite 342, Tylerton, TX 75901
2  01/17/19 13:33    7616 Lauren Run Apt. 642, South Julia, CO 81368
3  01/05/19 20:33          23081 Kyle Crest, Laurenchester, NY 10177
4  01/25/19 11:59      59764 Spears Mountains, Port Amanda, SC 29826

           Credit Card          SSN
```

```
0  9753-7632-8228-2717  499-70-8008
1  4353-8782-6482-8223  596-54-9892
2  2581-0339-8831-3503  608-47-3943
3  9431-1332-2561-3939  678-46-9684
4  4788-2969-5170-6914  980-25-2977
```

[46]:
```
# Quelles colonnes peuvent contenir des informations PII ?
sales_data.sketch.ask("Quelles colonnes peuvent contenir des informations PII ?
 ↪")
```

```
<IPython.core.display.HTML object>
```

[47]:
```
# Pouvez-vous me donner des noms conviviaux pour chaque colonne ? (sortie sous␣
 ↪forme de liste HTML)
sales_data.sketch.ask("Pouvez-vous me donner des noms conviviaux pour chaque␣
 ↪colonne ? (sortie sous forme de liste HTML)")
```

```
<IPython.core.display.HTML object>
```

[ ]:
```
# Quelles sont les visualisations les plus pertinentes à mettre en évidence ici
```

[48]:
```
sales_data.sketch.ask("Quelles sont les visualisations les plus pertinentes à␣
 ↪mettre en évidence ici ? (sortie en français)")
```

```
<IPython.core.display.HTML object>
```

[49]:
```
sales_data.sketch.howto("Créer un histogramme pour montrer la distribution des␣
 ↪quantités commandées, un graphique en barres pour comparer les différents␣
 ↪produits vendus et un graphique en ligne pour suivre l'évolution des ventes␣
 ↪au fil du temps. Un diagramme en boîte pourrait également être utile pour␣
 ↪visualiser les valeurs aberrantes dans les données de prix (avec la␣
 ↪librairie plotly)")
```

```
<IPython.core.display.HTML object>
```

[50]:
```
# Create histogram of quantity ordered
fig1 = px.histogram(sales_data, x="Quantity Ordered", nbins=10)
fig1.show()
```

[53]:
```
# Create bar chart of products sold
fig2 = px.bar(sales_data, x="Product", y="Quantity Ordered")
fig2.show()
```

```
Output hidden; open in https://colab.research.google.com to view.
```

[54]:
```
# Create line chart of sales over time
fig3 = px.line(sales_data, x="Order Date", y="Price Each")
fig3.show()
```

```
# Create box plot of price data
fig4 = px.box(sales_data, y="Price Each")
fig4.show()
```

```
Output hidden; open in https://colab.research.google.com to view.
```

[55]:
```
sales_data.sketch.howto("Créer des fonctionnalités dérivées de l'adresse")
```

```
<IPython.core.display.HTML object>
```

[56]:
```
# Create a new column for the city
sales_data['City'] = sales_data['Purchase Address'].apply(lambda x: x.
 ↪split(',')[1])

# Create a new column for the state
sales_data['State'] = sales_data['Purchase Address'].apply(lambda x: x.
 ↪split(',')[2].split(' ')[1])

# Create a new column for the zip code
sales_data['Zip Code'] = sales_data['Purchase Address'].apply(lambda x: x.
 ↪split(',')[2].split(' ')[2])

# Create a new column for the street name
sales_data['Street Name'] = sales_data['Purchase Address'].apply(lambda x: x.
 ↪split(',')[0])

# Create a new column for the street number
sales_data['Street Number'] = sales_data['Purchase Address'].apply(lambda x: x.
 ↪split(',')[0].split(' ')[0])

# Create a new column for the street name and number combined
sales_data['Street Address'] = sales_data['Street Number'] + ' ' +␣
 ↪sales_data['Street Name']

# Print the first 5 rows of the updated dataframe
sales_data.head()
```

[56]:
```
   Order ID                   Product  Quantity Ordered  Price Each  \
0    141234                    iPhone               1.0      700.00
1    141235  Lightning Charging Cable               1.0       14.95
2    141236          Wired Headphones               2.0       11.99
3    141237          27in FHD Monitor               1.0      149.99
4    141238          Wired Headphones               1.0       11.99

       Order Date                                   Purchase Address  \
0  01/22/19 21:25  10995 Williams Cliffs, East Michelleborough, A…
```

```
1  01/28/19 14:15     1067 Guzman View Suite 342, Tylerton, TX 75901
2  01/17/19 13:33    7616 Lauren Run Apt. 642, South Julia, CO 81368
3  01/05/19 20:33          23081 Kyle Crest, Laurenchester, NY 10177
4  01/25/19 11:59      59764 Spears Mountains, Port Amanda, SC 29826

           Credit Card          SSN                   City State Zip Code  \
0  9753-7632-8228-2717  499-70-8008  East Michelleborough    AZ    86031
1  4353-8782-6482-8223  596-54-9892              Tylerton    TX    75901
2  2581-0339-8831-3503  608-47-3943           South Julia    CO    81368
3  9431-1332-2561-3939  678-46-9684         Laurenchester    NY    10177
4  4788-2969-5170-6914  980-25-2977           Port Amanda    SC    29826

                 Street Name Street Number                   Street Address
0        10995 Williams Cliffs         10995       10995 10995 Williams Cliffs
1  1067 Guzman View Suite 342          1067  1067 1067 Guzman View Suite 342
2    7616 Lauren Run Apt. 642          7616    7616 7616 Lauren Run Apt. 642
3             23081 Kyle Crest         23081           23081 23081 Kyle Crest
4       59764 Spears Mountains         59764     59764 59764 Spears Mountains
```

[57]:

```python
sales_data.sketch.howto("Obtenez les 5 États les plus rentables")
```

```
<IPython.core.display.HTML object>
```

[58]:

```python
# Group the dataframe by state and sum the total sales for each state
state_sales = sales_data.groupby('State')['Price Each'].sum()

# Sort the states in descending order based on total sales
sorted_states = state_sales.sort_values(ascending=False)

# Get the top 5 most profitable states
top_states = sorted_states.head(5)

# Print the results
print("The top 5 most profitable states are:")
for state, sales in top_states.items():
    print(state, "with a total sales of $", round(sales, 2))
```

```
The top 5 most profitable states are:
CA with a total sales of $ 4222370.22
TX with a total sales of $ 2940575.44
FL with a total sales of $ 2166926.57
NY with a total sales of $ 2070624.71
IL with a total sales of $ 1413911.86
```

[59]:

```python
sales_data.sketch.ask("Y a t-il des valeurs manquantes dans ces données ?")
```

```
<IPython.core.display.HTML object>
```

[60]:

```
sales_data.sketch.ask("Quelles sont les données de type entier ?")
```

```
<IPython.core.display.HTML object>
```

[61]:

```
sales_data.sketch.ask("Quelles sont les données de type catégoriel?")
```

```
<IPython.core.display.HTML object>
```

[62]:

```
sales_data.sketch.ask("Quelles modèles algorithmiques peut-on appliquer pour␣
 ↪ces données ? (sortie sous forme de tableaux HTML)")
```

```
<IPython.core.display.HTML object>
```

[63]:

```
sales_data.sketch.ask("Quelle serait le meilleur modèle algorithmique à␣
 ↪appliquer pour ces données ? (sortie sous forme de listes HTML)")
```

```
<IPython.core.display.HTML object>
```

[65]:

```
sales_data.sketch.howto("prédire si une commande sera annulée ou non en␣
 ↪fonction des différentes variables telles que la quantité commandée, le prix␣
 ↪unitaire, la date de commande, etc avec des données d'entrées et de tests")
```

```
<IPython.core.display.HTML object>
```

[67]:

```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Create a new column in the dataframe to indicate if the order was cancelled␣
 ↪or not
sales_data['Cancelled'] = np.where(sales_data['Quantity Ordered'] == 0, 1, 0)

# Convert the order date column to datetime format
sales_data['Order Date'] = pd.to_datetime(sales_data['Order Date'])

# Extract features from the order date column
sales_data['Month'] = sales_data['Order Date'].dt.month
sales_data['Day'] = sales_data['Order Date'].dt.day
sales_data['Hour'] = sales_data['Order Date'].dt.hour

# Create a new dataframe with only the relevant columns for prediction
df = sales_data[['Cancelled', 'Quantity Ordered', 'Price Each', 'Month', 'Day',␣
 ↪'Hour']]

# Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(df.drop('Cancelled',␣
 ↪axis=1), df['Cancelled'], test_size=0.2, random_state=42)
```

[ ]:

```
# Train a logistic regression model on the training data
lr = LogisticRegression()
lr.fit(X_train, y_train)

# Make predictions on the test data
y_pred = lr.predict(X_test)
```

[ ]:

```
# Make predictions on the test data
y_pred = lr.predict(X_test)

# Calculate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```