DashboardHTLM

May 26, 2025

```
[11]: import requests
      import pandas as pd
      from io import StringIO


      url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
       ↪IBMSkillsNetwork-AI0272EN-SkillsNetwork/labs/dataset/2016.csv"


      response = requests.get(url)
      response.raise_for_status()  # Raise an error for bad responses


      data_bytes = response.content
      data_string = StringIO(data_bytes.decode('utf-8'))


      pandas_data = pd.read_csv(data_string)
```

```
[12]: file_path = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
       ↪IBMSkillsNetwork-AI0272EN-SkillsNetwork/labs/dataset/2016.csv"  # Adjust␣
       ↪accordingly
      df = pd.read_csv(file_path)


      print(df.head())
```

```
       Country          Region  Happiness Rank  Happiness Score  \
0      Denmark  Western Europe               1            7.526
1  Switzerland  Western Europe               2            7.509
2      Iceland  Western Europe               3            7.501
3       Norway  Western Europe               4            7.498
4      Finland  Western Europe               5            7.413

   Lower Confidence Interval Upper Confidence Interval  \
0                      7.460                     7.592
1                      7.428                      7.59
2                      7.333                     7.669
```

```
3                      7.421                      7.575
4                      7.351                      7.475

  Economy (GDP per Capita)   Family Health (Life Expectancy)  Freedom  \
0                  1.44178  1.16374                  0.79504  0.57941
1                  1.52733  1.14524                  0.86303  0.58557
2                  1.42666  1.18326                  0.86733  0.56624
3                  1.57744  1.12690                  0.79579  0.59609
4                  1.40598  1.13464                  0.81091  0.57104

   Trust (Government Corruption)  Generosity  Dystopia Residual
0                        0.44453     0.36171            2.73939
1                        0.41203     0.28083            2.69463
2                        0.14975     0.47678            2.83137
3                        0.35776     0.37895            2.66465
4                        0.41004     0.25492            2.82596
```

[13]:
```python
print("Tipos de datos de las columnas:")
print(df.dtypes)


print("\nInformación del DataFrame:")
print(df.info())


for col in df.select_dtypes(include=['object']).columns:
    try:
        df[col] = pd.to_numeric(df[col])
        print(f"La columna '{col}' se convirtió correctamente a numérica.")
    except ValueError:
        print(f"La columna '{col}' parece contener datos no numéricos.")
```

```
Tipos de datos de las columnas:
Country                           object
Region                            object
Happiness Rank                     int64
Happiness Score                  float64
Lower Confidence Interval        float64
Upper Confidence Interval         object
Economy (GDP per Capita)          object
Family                           float64
Health (Life Expectancy)          object
Freedom                           object
Trust (Government Corruption)    float64
Generosity                       float64
Dystopia Residual                float64
dtype: object
```

```
Información del DataFrame:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 157 entries, 0 to 156
Data columns (total 13 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   Country                        157 non-null    object
 1   Region                         157 non-null    object
 2   Happiness Rank                 157 non-null    int64
 3   Happiness Score                157 non-null    float64
 4   Lower Confidence Interval      153 non-null    float64
 5   Upper Confidence Interval      155 non-null    object
 6   Economy (GDP per Capita)       156 non-null    object
 7   Family                         157 non-null    float64
 8   Health (Life Expectancy)       155 non-null    object
 9   Freedom                        157 non-null    object
 10  Trust (Government Corruption)  157 non-null    float64
 11  Generosity                     157 non-null    float64
 12  Dystopia Residual              157 non-null    float64
dtypes: float64(6), int64(1), object(6)
memory usage: 16.1+ KB
None
La columna 'Country' parece contener datos no numéricos.
La columna 'Region' parece contener datos no numéricos.
La columna 'Upper Confidence Interval' parece contener datos no numéricos.
La columna 'Economy (GDP per Capita)' parece contener datos no numéricos.
La columna 'Health (Life Expectancy)' parece contener datos no numéricos.
La columna 'Freedom' parece contener datos no numéricos.
```

[14]:
```
import numpy as np


df = df.applymap(lambda x: x.strip() if isinstance(x, str) else x)

df.replace('', np.nan, inplace=True)


df = df.convert_dtypes()

print("DataFrame limpio:")
print(df.head())
print("\nTipos de datos después de la conversión:")
print(df.dtypes)
```

```
DataFrame limpio:
       Country          Region  Happiness Rank  Happiness Score  \
0      Denmark  Western Europe               1            7.526
1  Switzerland  Western Europe               2            7.509
```

```
2      Iceland  Western Europe              3           7.501
3       Norway  Western Europe              4           7.498
4      Finland  Western Europe              5           7.413

   Lower Confidence Interval Upper Confidence Interval  \
0                     7.46                     7.592
1                    7.428                      7.59
2                    7.333                     7.669
3                    7.421                     7.575
4                    7.351                     7.475

  Economy (GDP per Capita)   Family Health (Life Expectancy)  Freedom  \
0                1.44178  1.16374                0.79504  0.57941
1                1.52733  1.14524                0.86303  0.58557
2                1.42666  1.18326                0.86733  0.56624
3                1.57744   1.1269                0.79579  0.59609
4                1.40598  1.13464                0.81091  0.57104

   Trust (Government Corruption)  Generosity  Dystopia Residual
0                      0.44453     0.36171            2.73939
1                      0.41203     0.28083            2.69463
2                      0.14975     0.47678            2.83137
3                      0.35776     0.37895            2.66465
4                      0.41004     0.25492            2.82596

Tipos de datos después de la conversión:
Country                          string[python]
Region                           string[python]
Happiness Rank                            Int64
Happiness Score                         Float64
Lower Confidence Interval               Float64
Upper Confidence Interval        string[python]
Economy (GDP per Capita)         string[python]
Family                                  Float64
Health (Life Expectancy)         string[python]
Freedom                          string[python]
Trust (Government Corruption)           Float64
Generosity                              Float64
Dystopia Residual                       Float64
dtype: object

C:\Users\Patricia\AppData\Local\Temp\ipykernel_1664\527787011.py:4:
FutureWarning:

DataFrame.applymap has been deprecated. Use DataFrame.map instead.
```

[15]:
```python
missing_values = df.isnull().sum()
missing_columns = missing_values[missing_values > 0]
print("Columnas con valores faltantes:")
print(missing_columns)


df.fillna(df.mean(numeric_only=True), inplace=True)

print("\nTipos de datos después de la limpieza:")
print(df.dtypes)
```

```
Columnas con valores faltantes:
Lower Confidence Interval    4
Upper Confidence Interval    3
Economy (GDP per Capita)     2
Health (Life Expectancy)     3
Freedom                      1
dtype: int64

Tipos de datos después de la limpieza:
Country                          string[python]
Region                           string[python]
Happiness Rank                            Int64
Happiness Score                         Float64
Lower Confidence Interval               Float64
Upper Confidence Interval        string[python]
Economy (GDP per Capita)         string[python]
Family                                  Float64
Health (Life Expectancy)         string[python]
Freedom                          string[python]
Trust (Government Corruption)           Float64
Generosity                              Float64
Dystopia Residual                       Float64
dtype: object
```

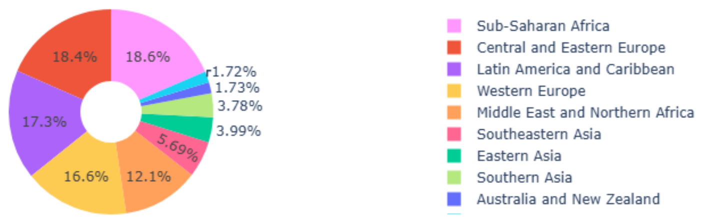
[16]:
```python
import plotly.graph_objects as go


top_10_df = df.nlargest(10, 'Happiness Score')


fig1 = go.Figure()

fig1.add_trace(go.Bar(
    x=top_10_df['Country'],
    y=top_10_df['Economy (GDP per Capita)'],
    name="GDP per Capita",
    marker_color='blue'
))

fig1.add_trace(go.Bar(
    x=top_10_df['Country'],
    y=top_10_df['Health (Life Expectancy)'],
    name="Healthy Life Expectancy",
    marker_color='green'
))


fig1.update_layout(
    title="GDP per Capita & Healthy Life Expectancy of Top 10 Happiest␣
 ↪Countries",
    xaxis_title="Country",
    yaxis_title="Value",
    barmode='group'
)

fig1.show()
```

[17]:
```
import plotly.express as px

selected_columns = ['Economy (GDP per Capita)', 'Family', 'Health (Life␣
 ↪Expectancy)',
                    'Freedom', 'Trust (Government Corruption)', 'Generosity',␣
 ↪'Happiness Score']
sub_df = df[selected_columns]

correlation_matrix = sub_df.corr()

fig2 = px.imshow(correlation_matrix,
                 labels=dict(x="Attributes", y="Attributes",␣
 ↪color="Correlation"),
                 color_continuous_scale='Viridis',
                 width=800,
                 height=600)

fig2.update_layout(
    title="Correlation Heatmap of Selected Attributes",
    xaxis_title="Attributes",
    yaxis_title="Attributes",
    xaxis=dict(side="bottom"),
    yaxis=dict(side="left")
)


fig2.show()
```

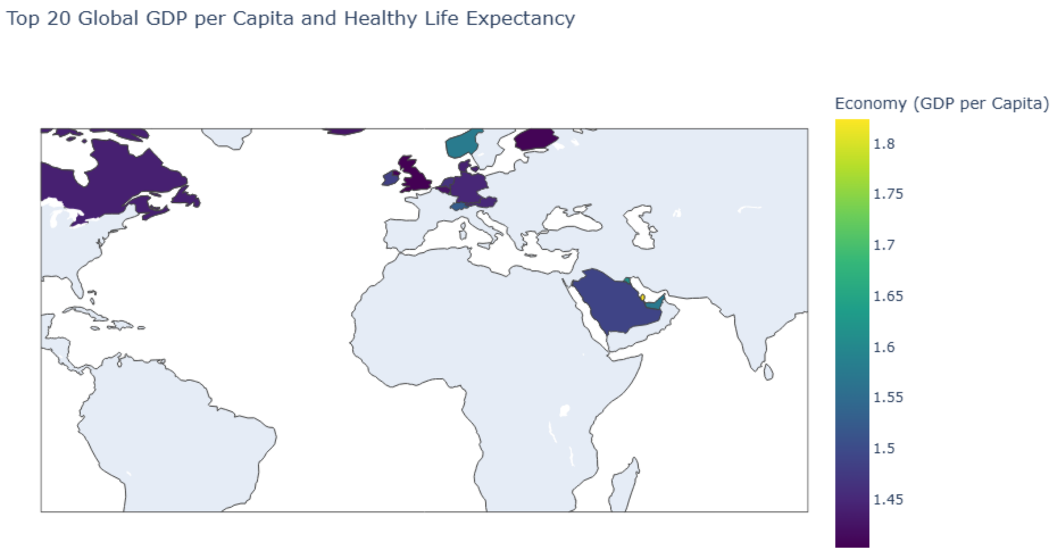
[18]:
```
df_sorted = df.sort_values(by="Economy (GDP per Capita)", ascending=True)


fig3 = px.scatter(
    df,
    x="Economy (GDP per Capita)",
    y="Happiness Score",
    color="Region",  # Color points by Region
    hover_data=["Country"],  # Show country name on hover
    title="Happiness Score vs. GDP per Capita (Colored by Region)"
)


fig3.update_layout(
    xaxis_title="Economy (GDP per Capita)",
    yaxis_title="Happiness Score",
    width=800,
    height=600
)


fig3.show()
```

[28]:
```
region_happiness = df.groupby("Region")["Happiness Score"].sum().reset_index()


fig4 = px.pie(
    region_happiness,
    names="Region",
    values="Happiness Score",
    title="Happiness Score Distribution by Region",
    color="Region",
    hole=0.3
)

fig4.show()
```

Happiness Score Distribution by Region



[30]:
```
df = df.dropna()

df["Economy (GDP per Capita)"] = df["Economy (GDP per Capita)"].astype(float)

top_20_df = df.nlargest(20, "Economy (GDP per Capita)")

fig5 = px.choropleth(
    top_20_df,
    locations="Country",
    locationmode="country names",
    color="Economy (GDP per Capita)",
    hover_name="Country",
    hover_data=["Health (Life Expectancy)"],
    color_continuous_scale="Viridis",
    title="Top 20 Global GDP per Capita and Healthy Life Expectancy" ,
     width=1000,
    height=600
)

fig5.update_geos(
    center={"lat": 20, "lon": 0},
    projection_scale=2,
)

fig5.show()
```



[32]:
```
!pip install dash
```

```
Requirement already satisfied: dash in
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (3.0.4)
Requirement already satisfied: Flask<3.1,>=1.0.4 in
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (from dash) (3.0.3)
Requirement already satisfied: Werkzeug<3.1 in
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (from dash) (3.0.6)
Requirement already satisfied: plotly>=5.0.0 in
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (from dash) (6.0.0)
Requirement already satisfied: importlib-metadata in
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (from dash) (8.7.0)
Requirement already satisfied: typing-extensions>=4.1.1 in
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (from dash) (4.12.2)
Requirement already satisfied: requests in
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (from dash) (2.32.3)
Requirement already satisfied: retrying in
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (from dash) (1.3.4)
Requirement already satisfied: nest-asyncio in
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (from dash) (1.6.0)
Requirement already satisfied: setuptools in
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (from dash) (75.8.0)
Requirement already satisfied: Jinja2>=3.1.2 in
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (from
Flask<3.1,>=1.0.4->dash) (3.1.5)
Requirement already satisfied: itsdangerous>=2.1.2 in
```

```
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (from
Flask<3.1,>=1.0.4->dash) (2.2.0)
Requirement already satisfied: click>=8.1.3 in
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (from
Flask<3.1,>=1.0.4->dash) (8.1.8)
Requirement already satisfied: blinker>=1.6.2 in
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (from
Flask<3.1,>=1.0.4->dash) (1.9.0)
Requirement already satisfied: narwhals>=1.15.1 in
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (from
plotly>=5.0.0->dash) (1.28.0)
Requirement already satisfied: packaging in
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (from
plotly>=5.0.0->dash) (24.2)
Requirement already satisfied: MarkupSafe>=2.1.1 in
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (from
Werkzeug<3.1->dash) (3.0.2)
Requirement already satisfied: zipp>=3.20 in
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (from importlib-
metadata->dash) (3.21.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (from
requests->dash) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (from
requests->dash) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (from
requests->dash) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (from
requests->dash) (2025.4.26)
Requirement already satisfied: six>=1.7.0 in
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (from
retrying->dash) (1.16.0)
Requirement already satisfied: colorama in
c:\users\patricia\anaconda3\envs\proyecto\lib\site-packages (from
click>=8.1.3->Flask<3.1,>=1.0.4->dash) (0.4.6)
```

[34]:
```python
import dash
from dash import html, dcc
```

[36]:
```python
app = dash.Dash()

app.layout = html.Div([
    dcc.Graph(figure=fig1),
    dcc.Graph(figure=fig2),
    dcc.Graph(figure=fig3),
    dcc.Graph(figure=fig4),
    dcc.Graph(figure=fig5),
])

if __name__ == "__main__":
    app.run(debug=False)
```

```
<IPython.lib.display.IFrame at 0x20cccb3bf50>
```

[38]:
```
import webbrowser
webbrowser.open_new("http://127.0.0.1:8050/")
```

[38]:
```
True
```

[ ]:
```
```