Actividad

- Regina Rodriguez Sánchez
- Emilio De Quevedo García
- Emilio Rizo De la Mora

Entregar

Archivos PDF de la actividad y la liga de la actividad en su repositorio.

Nota:

Las tareas 1 a la 5 se califican como entregadas o no entregadas.

Al integrarte con su participación en la actividad no se le tomará en cuenta para la calificación.

El límite para entregar las actividades es el viernes antes de las 23:59.

In [ ]:
```
# Si trabajamos en Google Colaboratory corremos las siguientes líneas de código
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive/
```

In [ ]:
```
# Nos cambiamos a la carpeta donde tengamos el repositorio
%cd /drive/MyDrive/Semestre/Arte-analitica
```

```
/content/drive/MyDrive/Semestre/Arte-analitica
```

Insurance dataset

El dataset contiene información demográfica sobre los asegurado en una compañía de seguros:

- age: edad del asegurado principal
- sex: género del asegurado, female o male
- bmi: índice de masa corporal
- children: número de niños que están a cubierta con la póliza.
- smoke: si fuma el beneficiario, yes/no.
- region: donde vive el beneficiario. Solo están en Estados Unidos. Regiones disponibles: northeast, southeast, southwest, northwest
- charges: costo del seguro.

Carga el dataset `data/insurance.csv` y haz un análisis gráfico de las variables.

In [ ]:
```
# Cargar las librerías
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

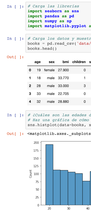
In [ ]:
```
# Carga los datos y muestra los primeros renglones
insurance_df=pd.read_csv('data/insurance.csv')
books.head()
```

Out[ ]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

In [ ]:
```
# ¿Cuáles son las edades de los clientes de la aseguradora?
# Una vez grafica de cómo se distribuye esta variable.
edad=insurance_df['age']
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa2f8399110>
```



In [25]:
```
# En general, quien tiene costos más altos de seguro, ¿los fumadores o los no fumadores?
# En la primedia los fumadores pagan más que los no fumadores

# Haz un gráfico que justifique tu respuesta.

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

sns.barplot(data=books, x = 'smoker', y='charges', estimator=np.mean)

# Eje y título. Colocamos la etiqueta correcta de acuerdo a la orientación.
plt.title('Costos clientes')
plt.xlabel('Fumador')
plt.ylabel('Cargos')
```

Out[25]:
```
Text(0, 0.5, 'Cargos')
```

In [25]:
```
# ¿Cómo varía el cargo por seguro dependiendo del número de hijos?
# Parece ser que la cantidad de hijos no afecta mucho los cargos del seguro, ya que los que tienen 2-3 hijos son los
# que mas pagan
# pero no quiere decir que entre mas hijos mas pagan porque depende de esa cantidad los cargos comienzan a bajar

# El número de hijos es una variable numérica, pero como tiene pocos valores distintos,
# podemos considerarla como una variable categórica.
fig2 = plt.figure(figsize=(7,7))

sns.barplot(data=books, x = 'children', y= 'charges', estimator=np.mean)

# Eje y título. Colocamos la etiqueta correcta de acuerdo a la orientación.
plt.title('Costos clientes hijos')
plt.xlabel('Número de hijos')
plt.ylabel('Cargos')
```

Out[25]:
```
Text(0, 0.5, 'Cargos')
```

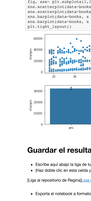
In [26]:
```
# ¿Cuál es la región con más seguros asegurados?
#R= Southeast

# Tamaño de la imagen
fig3 = plt.figure(figsize=(7,7))

# Grafica conteoplot para hacer barras con el número de apariciones de cada género.
sns.countplot(data=books, x = 'region')

# Eje y título. Colocamos la etiqueta correcta de acuerdo a la orientación.
plt.xlabel('Region')
plt.ylabel('Clientes')
```

Out[26]:
```
Text(0, 0.5, 'Clientes')
```



In [27]:
```
# ¿Cómo se relacionan las variables numéricas entre sí?
# Habitualo es un gráfico.
books_corr = books.corr()

# Para graficar el mapa de calor usamos heatmap. No necesitamos especificar x ni y
sns.heatmap(data=books_corr)
```

Out[27]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa2f9922410>
```

In [28]:
```
# ¿Qué relación hay entre el BMI y la edad? ¿Afecta el género en esta relación?
#R= hay poca relación entre esta variables ya que no se nota una línea de tendencia clara y el genero no parece afecta
# tampoco en esta correlación

# Habitualo es un gráfico y verifica si coincide con lo que obtuviste en la pregunta anterior.

fig4 = plt.figure(figsize=(7,7))

# Grafica scatterplot.
sns.scatterplot(data=books, x = 'bmi', y='age', hue='sex')

# Eje y título. Colocamos la etiqueta correcta de acuerdo a la orientación.
plt.title('Relación entre el BMI y la edad y la evaluación por género')
plt.xlabel('bmi')
plt.ylabel('Edad')
plt.grid()
```

In [29]:
```
# ¿Cuál es el cargo promedio por género de un seguro?
fig = plt.figure(figsize=(7,7))

sns.barplot(data=books, x = 'sex', y='charges', estimator=np.mean)

# Eje y título. Colocamos la etiqueta correcta de acuerdo a la orientación.
plt.title('Cargo promedio por genero')
plt.xlabel('Genero')
plt.ylabel('Cargos')
```

Out[29]:
```
Text(0, 0.5, 'Cargos')
```

In [39]:
```
# De una sola imagen, grafica las relaciones que tienen las variables
# age, bmi, smoker y region con charges.
fig, axes= plt.subplots(4,1, figsize=(7,15))
sns.scatterplot(data=books, x = 'age', y= 'charges',ax=axes[0,])
sns.scatterplot(data=books, x = 'bmi', y= 'charges',ax=axes[1,])
sns.scatterplot(data=books, x = 'smoker', y= 'charges',ax=axes[2,])
sns.scatterplot(data=books, x = 'region', y= 'charges',ax=axes[3,])
plt.tight_layout()
```


Guardar el resultado como pdf

- Escribe aquí abajo la liga de tu repositorio.
- (Haz doble clic en esta celda y copia la URL dentro del paréntesis)

Ejpa el repositorio de Regina[Link al repositorio de Regina]

- Exporta el notebook a formato HTML

In [45]:
```
! jupyter nbconvert --to HTML '/content/drive/MyDrive/Semestre/Arte-analitica/5.2) - Actividad - Visualizacion de datos.ipynb'
```

```
[NbConvertApp] WARNING | pattern '/content/drive/MyDrive/Semestre/Arte-analitica/5.2) - Actividad - Visualizacion de datos.ipynb' matched no files
This application is used to convert notebook files (*.ipynb)
        to various other formats.

        WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
=======
The options below are convenience aliases to configurable class-options,
as listed in the "Equivalent to" description-line of the aliases.
To see all configurable class-options for some <cmd>, use:
    <cmd> --help-all

--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
--show-config
    Show the application's configuration (human-readable format)
    Equivalent to: [--Application.show_config=True]
--show-config-json
    Show the application's configuration (json format)
    Equivalent to: [--Application.show_config_json=True]
--generate-config
    generate default config file
    Equivalent to: [--JupyterApp.generate_config=True]
-y
    Answer yes to any questions instead of prompting.
    Equivalent to: [--JupyterApp.answer_yes=True]
--execute
    Execute the notebook prior to export.
    Equivalent to: [--ExecutePreprocessor.enabled=True]
--allow-errors
    Continue notebook execution even if one of the cells throws an error and include the error message in the cell output (the default behaviour is to abort execution). This flag is only relevant if '--execute' was specified, too.
    Equivalent to: [--ExecutePreprocessor.allow_errors=True]
--stdin
    read a single notebook file from stdin. Write the resulting notebook with default basename 'notebook.*'
    Equivalent to: [--NbConvertApp.from_stdin=True]
--stdout
    Write notebook output to stdout instead of files.
    Equivalent to: [--NbConvertApp.writer_class=StdoutWriter]
--inplace
    Run nbconvert in place, overwriting the existing notebook (only
    relevant when converting to notebook format)
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.export_format=notebook --FilesWriter.build_directory=]
--clear-output
    Clear output of current file and save in place,
    overwriting the existing notebook.
    Equivalent to: [--NbConvertApp.use_output_suffix=False --NbConvertApp.export_format=notebook --FilesWriter.build_directory= --ClearOutputPreprocessor.enabled=True]
--no-prompt
    Exclude input and output prompts from converted document.
    Equivalent to: [--TemplateExporter.exclude_input_prompt=True --TemplateExporter.exclude_output_prompt=True]
--no-input
    Exclude input cells and output prompts from converted document.
    This mode is ideal for generating code-free reports.
    Equivalent to: [--TemplateExporter.exclude_output_prompt=True --TemplateExporter.exclude_input=True]
--log-level=<Enum>
    Set the log level by value or name.
    Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR', 'CRITICAL']
    Default: 30
    Equivalent to: [--Application.log_level]
--config=<Unicode>
    Full path of a config file.
    Default: ''
    Equivalent to: [--JupyterApp.config_file]
--to=<Unicode>
    The export format to be used, either one of the built-in formats
    ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'pdf', 'python', 'rst', 'script', 'slides'
    ]
        or a dotted object name that represents the import path for an
        ``Exporter`` class
    Default: ''
    Equivalent to: [--NbConvertApp.export_format]
--template=<Unicode>
    Name of the template file to use
    Default: ''
    Equivalent to: [--TemplateExporter.template_file]
--writer=<DottedObjectName>
    Writer class used to write the
                                        results of the conversion
    Default: 'FilesWriter'
    Equivalent to: [--NbConvertApp.writer_class]
--post=<DottedOrNone>
    PostProcessor class used to write the
                                        results of the conversion
    Default: ''
    Equivalent to: [--NbConvertApp.postprocessor_class]
--output=<Unicode>
    overwrite base name use for output files.
                can only be used when converting one notebook at a time.
    Default: ''
    Equivalent to: [--NbConvertApp.output_base]
--output-dir=<Unicode>
    Directory to write output(s) to. Defaults
                                  to output to the directory of each notebook. To recover
                                  previous default behaviour (outputting to the current
                                  working directory) use . as the flag value.
    Default: ''
    Equivalent to: [--FilesWriter.build_directory]
--reveal-prefix=<Unicode>
    The URL prefix for reveal.js (version 3.x).
        This defaults to the reveal CDN, but can be any url pointing to a copy
        of reveal.js.
        For speaker notes to work, this must be a relative path to a local
        copy of reveal.js: e.g., "reveal.js".
        If a relative path is given, it must be a subdirectory of the
        current directory (from which the server is run).
        See the usage documentation
        (https://jupyter-notebook.readthedocs.io/en/latest/usage.html#forming-jupyter-notebook-urls)
        for more details.
    Default: ''
    Equivalent to: [--SlidesExporter.reveal_url_prefix]
--nbformat=<Enum>
    The nbformat version to write.
    Use this to downgrade notebooks.
    Choices: any of [1, 2, 3, 4]
    Default: 4
    Equivalent to: [--NotebookExporter.nbformat_version]

Examples
--------

    The simplest way to use nbconvert is

            > jupyter nbconvert mynotebook.ipynb

            which will convert mynotebook.ipynb to the default format (probably HTML).

            You can specify the export format with '--to'.
            Options include ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'pdf', 'python', 'rst', '
script', 'slides'].

            > jupyter nbconvert --to latex mynotebook.ipynb

            Both HTML and LaTeX support multiple output templates. LaTeX includes
            'base', 'article' and 'report'.  HTML includes 'basic' and 'full'. You
            can specify the flavor of the format used.

            > jupyter nbconvert --to html --template basic mynotebook.ipynb

            You can also pipe the output to stdout, rather than a file

            > jupyter nbconvert mynotebook.ipynb --stdout

            PDF is generated via latex

            > jupyter nbconvert mynotebook.ipynb --to pdf

            You can get (and serve) a Reveal.js-powered slideshow

            > jupyter nbconvert myslides.ipynb --to slides --post serve

            Multiple notebooks can be given at the command line in a couple of
            different ways:

            > jupyter nbconvert notebook*.ipynb
            > jupyter nbconvert notebook1.ipynb notebook2.ipynb

            or you can specify the notebooks list in a config file, containing:

            c.NbConvertApp.notebooks = ["my_notebook.ipynb"]

            > jupyter nbconvert --config mycfg.py

To see all available configurables, use `--help-all`.
```

- Haz doble clic en el archivo nuevo que se creó dentro de la carpeta `arte-analitica` y en la parte superior derecha dale clic en **Imprimir**
- Imprime el archivo como PDF y súbelo a Canvas.