Actividad

- Regina Rodriguez Sanchez - a01284329
- Emilio De Gyves García - a01351989
- Emilio Rizo De la Mora - a01721612

Entregar

Archivo PDF de la actividad y la liga de la actividad en su repostitorio.

Nota:

Esta actividad se evaluará de acuerdo a la rúbrica en Canvas.

Al integrante que no participe 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.

Importante:

- Colocar nombre en ejes en gráficas
- Nombre en gráficas
- Conclusiones con el nombre de cada alumno
- Contestar cada pregunta

Gastos en seguro

Nuestro objetivo será construir un modelo que nos permita estimar los gastos de seguro dependiendo de edad,sexo, indice de masa corporal, numero de hijos, si se fuma, región.

El dataset consta de las columnas:

- age: edad del beneficiario principal
- sex: female o male
- bmi: indice de masa corporal
- children: numero de niños que estan cubiertos con la poliza.
- smoke: si fuma el beneficiario si/no
- region: región en dónde vive el beneficiario. Estos datos son de Estados Unidos. Regiones disponibles: northeast, southeast, southwest, northwest
- charges: costo del seguro.

Referencia de dataset: https://github.com/stedy/Machine-Learning-with-R-datasets/blob/master/insurance.csv

1.a. Carga los datos del archivo insurance.csv

In [ ]:

```python
from google.colab import drive
drive.mount('/content/drive/')
%cd 'drive/MyDrive/SemanaTec/arte-analitica'
import pandas as pd
import numpy as np
import seaborn as sns


books = pd.read_csv('data/insurance.csv')
```

1.b. Crea unas columnas con las siguientes caracteristicas:

- fuma: 1 si el bvalor de smoke es "yes" y 0 si el valor de smoke es "no"
- region: si es northeast - 0 , southeast- 1, southwest- 2, northwest - 3
- sexo: 0 si es "male" y 1 si es "female"

In [ ]:

```python
books['smoker']  = books['smoker'].map(dict(yes=1, no=0))
books['region'] = books['region'].map(dict(northeast=0, southeast=1, southwest=2, northwest=3))
books['sex'] = books['sex'].map(dict(female=1, male=0))
books.head()
```

Out[ ]:

	age	sex	bmi	children	smoker	region	charges
0	19	1	27.900	0	1	2	16884.92400
1	18	0	33.770	1	0	1	1725.55230
2	28	0	33.000	3	0	1	4449.46200
3	33	0	22.705	0	0	3	21984.47061
4	32	0	28.880	0	0	3	3866.85520

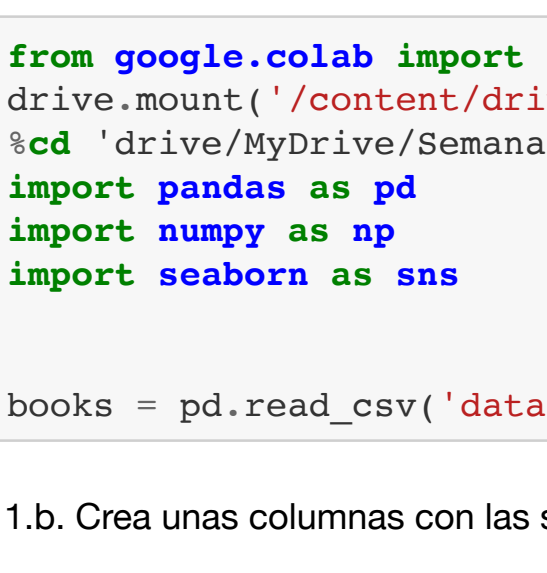
1. Generar una(s) grafica(s) para visuaizar cómo se relaciona cada columna de costo en seguro contra otras variables.

In [ ]:

```python
books_corr = books.corr()
sns.heatmap(data=books_corr)
books.corr()
```

Out[ ]:

	age	sex	bmi	children	smoker	region	charges
age	1.000000	0.020856	0.109272	0.042469	-0.025019	0.002613	0.299008
sex	0.020856	1.000000	-0.046371	-0.017163	-0.076185	0.009346	-0.057292
bmi	0.109272	-0.046371	1.000000	0.012759	0.003750	-0.054428	0.198341
children	0.042469	-0.017163	0.012759	1.000000	0.007673	0.036617	0.067998
smoker	-0.025019	-0.076185	0.003750	0.007673	1.000000	-0.044124	0.787251
region	0.002613	0.009346	-0.054428	0.036617	-0.044124	1.000000	-0.050226
charges	0.299008	-0.057292	0.198341	0.067998	0.787251	-0.050226	1.000000



1. ¿ Qué conclusiones puedes obtener de las graficas anteriores?

In [ ]:

```python
#R= A partir de los resultados de las graficas anteriores la variable con la que mas correlacion
#tiene la variable "charges" es con la de "smoker" con las demas variables tienen muy poca relacion entre si
```

1. **Regresión lineal**. En su forma más simple, consiste en asumir que una variable x y una variable y presentan una relación lineal de la forma:

$$y \approx \beta_0 + \beta_1 \cdot x$$

¿Cuál es el valor de beta_0 , beta_1 y el Score para **Edad vs Costo** , **Indice de masa corporal vs Costo** , **Niños vs Costo** y otros?

In [ ]:

```python
from sklearn.linear_model import LinearRegression
# Edad vs Costo
model1 = LinearRegression()
model1.fit(books[['age']], books['charges'])
print("Age Vs Charges")
print("Beta_0: ",model1.intercept_)
print("Beta_1: ",model1.coef_)
print("Score: ", model1.score(books[['age']],books['charges']), "\n")

# Indice de masa corporal vs Costo
model2 = LinearRegression()
model2.fit(books[['bmi']], books['charges'])
print("BMI Vs Charges")
print("Beta_0: ",model2.intercept_)
print("Beta_1: ",model2.coef_)
print("Score: ", model2.score(books[['bmi']],books['charges']), "\n")

# Niños vs Costo
model3 = LinearRegression()
model3.fit(books[['children']], books['charges'])
print("Children Vs Charges")
print("Beta_0: ",model3.intercept_)
print("Beta_1: ",model3.coef_)
print("Score: ", model3.score(books[['children']],books['charges']), "\n")
```

```
Age Vs Charges
Beta_0:  3165.885006063021
Beta_1:  [257.72261867]
Score:  0.08940589967885804

BMI Vs Charges
Beta_0:  1192.9372089611497
Beta_1:  [393.8730308]
Score:  0.03933913991786264

Children Vs Charges
Beta_0:  12522.495549644098
Beta_1:  [683.08938248]
Score:  0.004623758854459203
```

1. ¿Cuál de los modelos es mejor de acuerdo al score ordenalos del mejor al peor?

1- children vs charges 2- bmi vs charges 3- age vs charges

1. **Regresión lineal múltiple**. Como ahora vamos a incluir más de una variable, el modelo se rescribe a :

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 \cdot x_1 + \hat{\beta}_2 \cdot x_2 + \hat{\beta}_3 \cdot x_3 + \ldots \epsilon$$

Elige las diferentes combinaciones de variables. Ejemplo: Redes y Youtube vs Ventas

¿Cuál es el valor de beta_0, betas_1 , beta_2 , el score y el score ajustado de de 5 combinaciones . Ejemplo: **Edad, Niños, Region vs Costo** ?

In [ ]:

```python
lm1 = LinearRegression()
# La variable X la ponemos con doble corchete!
lm1.fit(books[['age', 'children','region']], books['charges'])
print('Age, Niños, and Region')
print('beta_0: ', lm1.intercept_)
print('betas: ', lm1.coef_)
print('Score: ', lm1.score(books[['age', 'children','region']], books['charges']), '\n')


#Lm2 Niños,Region,BMI and charges

lm2 = LinearRegression()
lm2.fit(books[['bmi', 'children','region']], books['charges'])
print('BMI, Niños, and Region')
print('beta_0: ', lm2.intercept_)
print('betas: ', lm2.coef_)
print('Score: ', lm2.score(books[['bmi', 'children','region']], books['charges']), '\n')

#Lm3 BMI, Smoker, Age and charges
lm3 = LinearRegression()
lm3.fit(books[['bmi', 'smoker','age']], books['charges'])
print('BMI, Fumador, and Age')
print('beta_0: ', lm3.intercept_)
print('betas: ', lm3.coef_)
print('Score: ', lm3.score(books[['bmi', 'smoker','age']], books['charges']), '\n')


#Lm4 AGE,SMOKER, REGION and charges
lm4 = LinearRegression()
lm4.fit(books[['age', 'smoker','region']], books['charges'])
print('EDAD, Fumador, and Region')
print('beta_0: ', lm4.intercept_)
print('betas: ', lm4.coef_)
print('Score: ', lm4.score(books[['age', 'smoker','region']], books['charges']), '\n')

#LM5 REGION,NIÑOS,Sex and charges

lm5 = LinearRegression()
lm5.fit(books[['region', 'children','sex']], books['charges'])
print('REGION, NIÑOS, and Sex')
print('beta_0: ', lm5.intercept_)
print('betas: ', lm5.coef_)
print('Score: ', lm5.score(books[['region', 'children','sex']], books['charges']), '\n')
```

```
Age, Niños, and Region
beta_0:  3477.9425852574423
betas:  [ 255.74320281  576.03582763 -581.99464652]
Score:  0.09528541418900904

BMI, Niños, and Region
beta_0:  1332.1702030704764
betas:  [ 387.62846796  673.54095788 -460.95106427]
Score:  0.04538694017480793

BMI, Fumador, and Age
beta_0:  -11676.830425187763
betas:  [  322.61513282 23823.68449531   259.54749155]
Score:  0.7474771588119513

EDAD, Fumador, and Region
beta_0:  -2127.3923447437264
betas:  [  274.89200764 23834.14229849  -175.3840475 ]
Score:  0.7216563908925873

REGION, NIÑOS, and Sex
beta_0:  14029.253931057001
betas:  [ -572.46196056   692.7124342  -1346.69734976]
Score:  0.01049888229004281
```

1. ¿Cuál modelo es el que nos conviene elegir?

R = el de BMI, Smoker and Age

1. Conclusiones

Emilio Rizo A01721612: Es muy interesante como la libreria puede ser muy util para ecuaciones y funciones matematicas como puede ser la regresión lineal y multiple. Fue muy interesante implementarlo nosotros a el csv file que tuvimos que manipular. Nos puede ser de mucha ayuda en un futuro y realmente puede ahorrarte mucho tiempo. Los dataframes con los que trabajamos fueron muy utiles y me gusto mucho aprender como utilizarlos y modificarlos. Fue un proyecto muy interesante que me ayudo a fortalecer mis herramientas para programar.

Emilio De Gyves A01351989:
La analítica es un área muy útil para estos días, es necesaria para muchos trabajos y avances, en este proyecto pudimos observar una de sus aplicaciones en el mundo laboral, respecto a la actividad, gracias a la grafica pudimos observar que la mayor correlación la tuvieron entre los costos y los fumadores, graficar fue muy útil ya que lo pudimos ver de una manera muy visual, nos costó un poco de trabajo la parte de regresión lineal y la regresión lineal múltiple pero nos sirvió el trabajar en equipo ya que gracias a eso pudimos resolverlo y encontrar el modo.

Regina Rodriguez A01284329: Al terminar esta semana tec aprendi muchas funcionalidades nuevas y hacer un mejor uso de la tecnologia que tenemos a nuestro alcance, como conectar github con google drive y a combinar la estadistica con la programacion. Este proyecto me ayudo s observar todo lo que poemos hacer con las librerias de panda y las librerias de graficos para poder tener mejores predicciones o ver que variables pueden afectar algo como los costos de seguro. Como equipo nos entendimos muy bien y creo que hicimos un buen trabajo en conjunto.

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)

Liga al repositorio de

- Exporta el notebook a formato HTML.

In [5]:

```
! jupyter nbconvert --to HTML '/content/drive/MyDrive/SemanaTec/arte-analitica/6.3 - Actividad Regresion Linea.ipynb'
```

```
[NbConvertApp] WARNING | pattern '/content/drive/MyDrive/SemanaTec/arte-analitica/6.3 - Actividad Regresion Linea.ipy
nb' 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 ou
tput (the default behaviour is to abort conversion). 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: 'html'
    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://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js-html-slideshow)
            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`.
```

In [ ]:

- 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.