|  |  |  |
| --- | --- | --- |
| NOMBRE DE LA ASIGNATURA  Análisis de datos con Python (Nivel 2- Deep learning con Python)  PROFESOR (A)  Pedro Rotta | FECHA  06/02/2022 | TRABAJO N°2  GRUPO 11 |

Alumnos:

Mondragón Payac Billy

Panta Panta Jimy Aldair

Alberca Elera Gustavo Alonso

Del Carmen Rangel Renzo Daniel

Amaya Vega Manuel

|  |  |  |
| --- | --- | --- |
|  | marca-udep.png | **2022** |

1. **Fundamentación**

El presente trabajo busca poner en práctica los conocimientos aprendidos en el curso: Análisis de datos con Python (Nivel 2- Deep learning con Python), en esta oportunidad aplicaremos el uso útil dentro de una base de datos con finalidad médica que almacena datos de afiliados al seguro de salud de 4 de las regiones de los Estados Unidos: northeast, southwest, southeast y northwest . Con esto planteamos un proyecto útil basados en los requerimientos del trabajo encargado en el curso.

1. **Abstract**

El presente trabajo está fundamentado en los requerimientos exigidos aplicados a una situación real como los es el campo de los seguros de vida, específicamente, la predicción del costo de una prima de seguro basados en datos específicos del solicitante. La presente utiliza una base de datos que contiene información de afiliados al seguro de salud estadounidense que almacena información de diversos asegurados cómo: Edad, Género, Índice de masa corporal, Número de hijos, Indicador de si es fumador, Región en la que vive y Prima de seguro. El ejecutable lleva por nombre: “ProyectFinal\_IntermPython2022”, y sobre el cual se utilizarán: Seaborn, pandas,matplotlib y plotly para realizar análisis estadístico de la data y la realización de gráficas de estas. En cuanto al entrenamiento de los modelos realizaremos 4 modelos de Machine Learning como: KNN Neighbors, Algoritmo KNN Regression, Algoritmo de Regresión Logística y Algoritmo de Árbol de decisión. Cada uno de los apartados tienen definidos el valor de predicción final que va teniendo el modelo usado por el usuario.

El funcionamiento del programa se irá presentando conforme se vaya desarrollando la explicación del mismo, por tanto, los resultados obtenidos serán visibles en cada parte del programa y no como un ítem independiente, esto con el fin de reducir el tamaño de la lectura del documento y que este resulte intuitivo para el lector.

1. **Objetivos**
   1. **Objetivos especifico**

Poder predecir los costos que tiene la prima de seguro basado en las características del asegurado.

* 1. **Segundo objetivo**

Analizar los datos mediante gráficas intuitivas que permitan la lectura intuitiva de los datos ingresados según el cliente a analizar

* 1. **Tercer objetivo**

Contar con una herramienta de predicción que permita mejorar en el tiempo y según la data ingresada a su interior, perfeccionando así los modelos de predicción utilizados en su interior

* 1. **Cuarto objetivo**

Con la data obtenida poder brindar un servicio acorde a cada segmento de público que busque una prima de seguro basado en su estilo y características de vida

1. **Metodología**

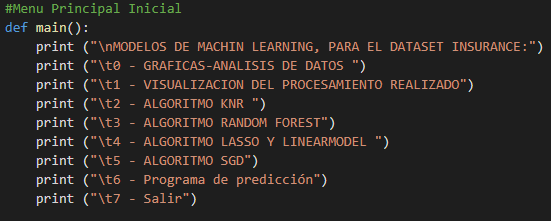
Se han utilizado las diferentes librerías vistas en clase como: *numpy, pandas, myplotlib*, *getpass, myplotlib.pylab*, *plotly.express, matplotlib.pyplot, tensorflow, seaborn, plotly.graph\_objects, get\_dummies* y la gran parte de la familia *sklearn*para la realización de este proyecto, las cuales se verán en mayor profundidad en el desarrollo del programa.

También, nos hemos servido de la base de datos llamada: “insurance.csv” obtenida de la plataforma *kaggle* mediante el enlace: ‘<https://www.kaggle.com/mirichoi0218/insurance>’

Además, se ha realizado la programación del programa en la plataforma *colab.research* de Google utilizando el lenguaje de programación de Python.

1. **Funcionamiento**

El usuario puede ingresar en la plataforma servido desde un menú que le permitirá orientarse dentro del programa, que es el siguiente:



1.- Gráficas – Análisis de datos

2.-Visualización del procesamiento realizado

3.- Algoritmo KNR

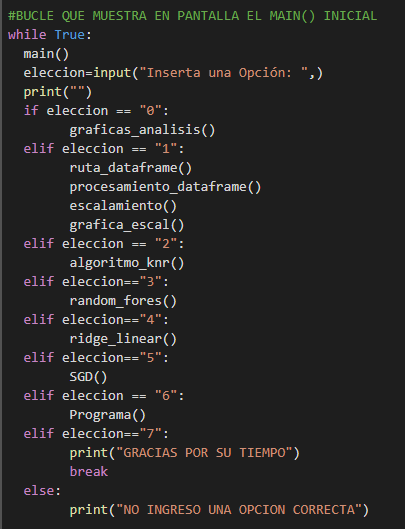
4.- Algoritmo Random Forest

5.-Algoritmo Lasso y Linearmodel

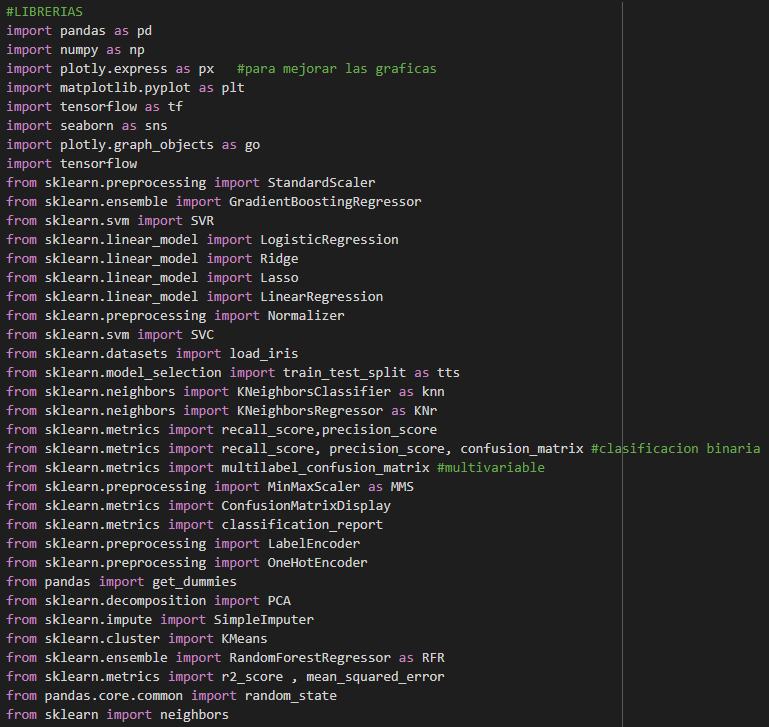
6.- Algoritmo SGD

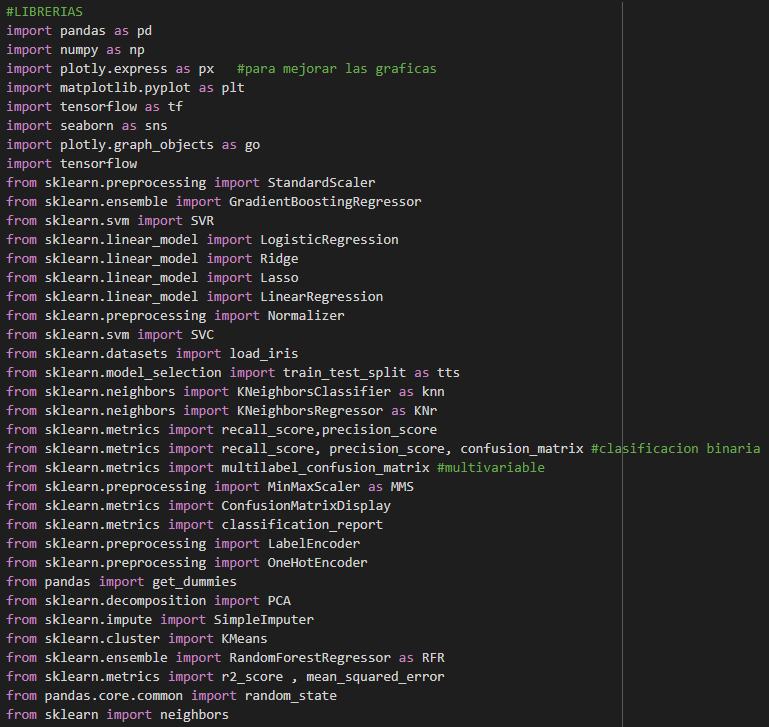
7.- Programa de Predicción

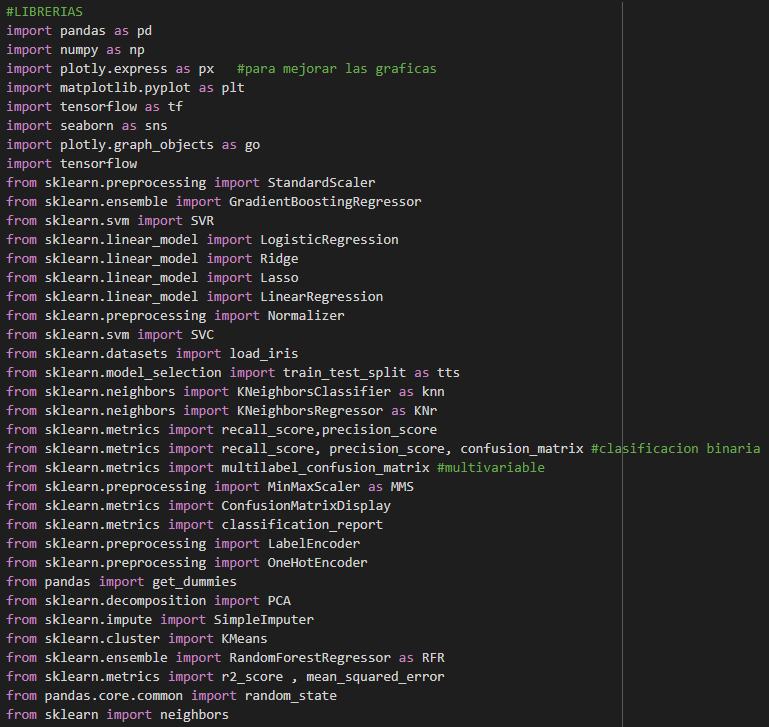
8.- Salir



Como muestra y guía inicial mostramos las librerías en su totalidad que han sido utilizadas para el correcto funcionamiento del programa realizado:

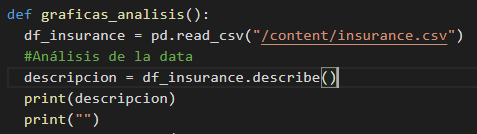






* 1. **Gráficas – Análisis de datos**

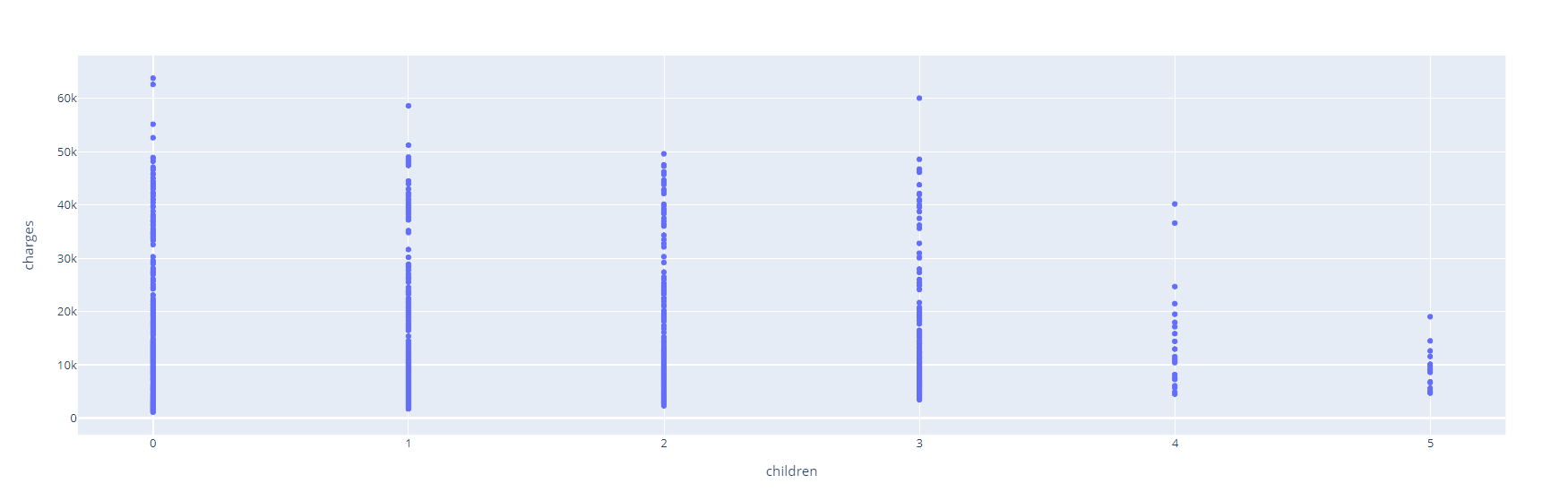
Como inicio enlazamos el dataframe a utiizar, en este caso es “*insurance.csv*” y se realiza el análisis general de la data



Seguido procedemos a realizar un análisis de las gráficas obtenidas directamente desde el dataframe



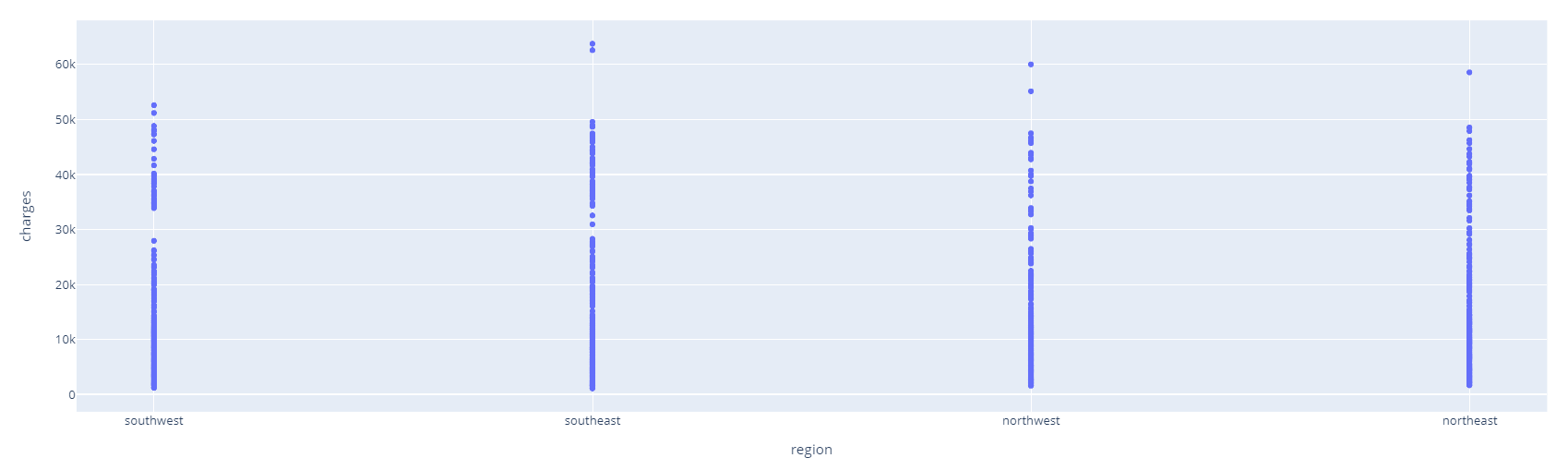
Como resultado, a precia claramente que las personas con mayor cantidad de hijos parecen tener un menor gasto en seguro de salid, sin embargo, esto no es definitivo, puesto que ambos valores, “*children”* y *“charges”* no están relacionados directamente producto del factor socioeconómico. Aún así, la afirmación inicial cobra sentido según los parámetros puestos desde un inicio.



Generamos otra imagen que permite observar la comparativa entre la región y los cargos de seguros, En este apartado podemos afirmar que a los clientes no les afecta los cargos del seguro independientemente de la región en donde se encuentren.



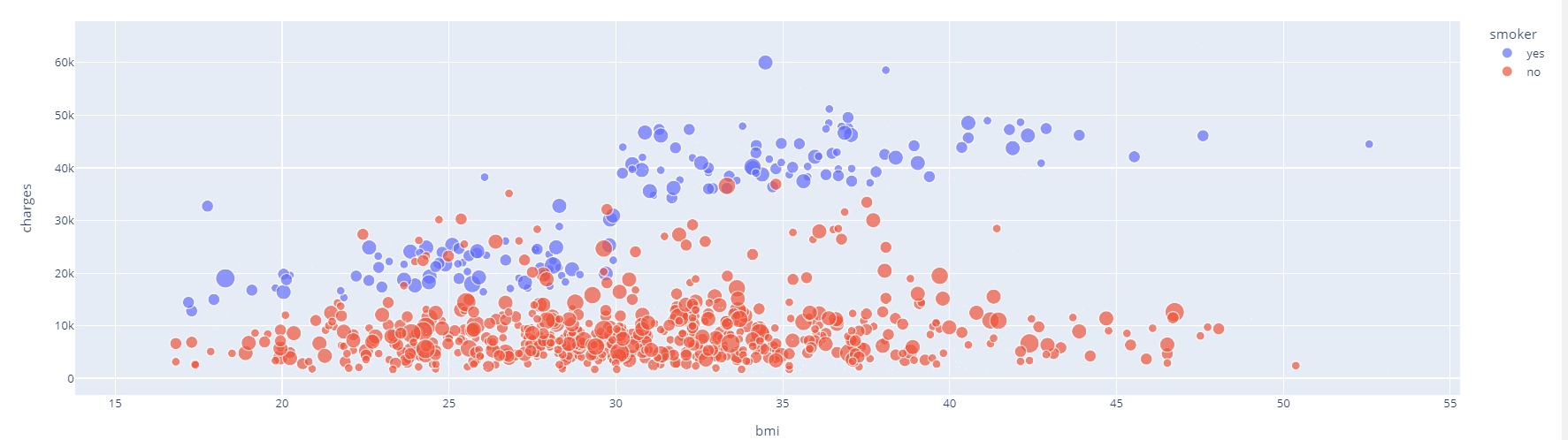
El resultado de fig2 es el siguiente:



Además, mostramos una imagen que nos permite realizar una comparativa a mayor profundidad tomando en cuenta factores como la edad, cargos de seguros, si la persona es fumadora y la cantidad de hijos que tiene.



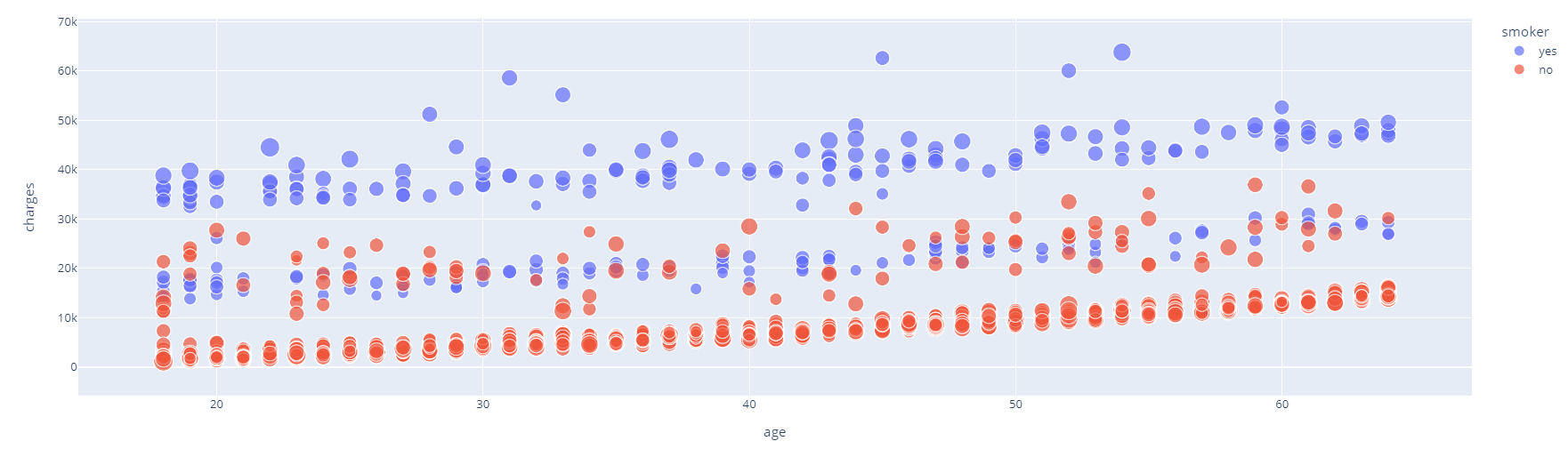
Con esto obtenemos concluimos que ser fumadores es un indicador decisivo en el costo del seguro, las personas con un índice de masa corporal menor a 30 tienen un gasto menos a los 30 mil dólares, incluso si son fumadores; sin embrago, las personas con un índice de masa corporal mayor a 25 y con la característica de ser fumadores tienden a tener un gasto muchísimo más alto que las no fumadoras o con un índice de masa corporal menor a 30.



Finalmente, mostramos una última imagen que toma en cuenta factores como la edad, cargos de seguros, si la persona es fumadora y el índice de masa corporal que tienen.

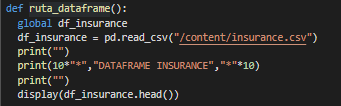


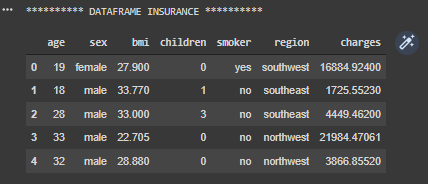
Con esto obtenemos concluimos que ser fumadores es un indicador decisivo en el costo del seguro, las personas con un índice de masa corporal menor a 30 tienen un gasto menos a los 30 mil dólares, incluso si son fumadores; sin embrago, las personas con un índice de masa corporal mayor a 25 y con la característica de ser fumadores tienden a tener un gasto muchísimo más alto que las no fumadoras o con un índice de masa corporal menor a 30.



* 1. **Visualización del procesamiento realizado** 
     1. **Ruta data frame**

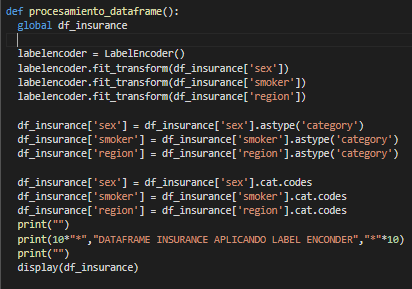
Esta opción lleva a utilizar una variable global, como lo es el de la lectura del CSV utilizado y a la muestra de los primeros datos del data frame



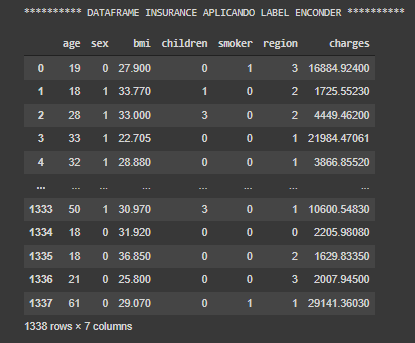


* + 1. **Procesamiento data frame**

En este apartado sabemos la existencia de variables tipo *String*, para evitarlo, los eliminamos utilizando el método *labelencoder* para todo el dataset.

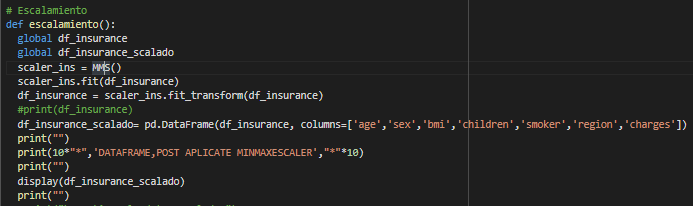


El resultado obtenido es el siguiente:

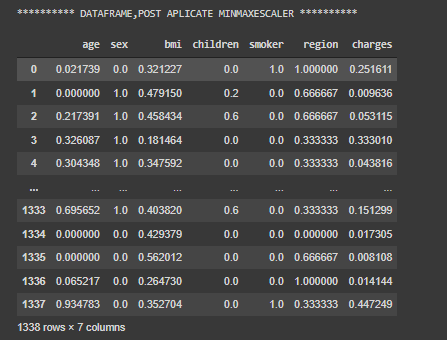


* + 1. **Escalamiento**

En este apartado utilizamos los métodos *scaler* y *fit* para escalar la data analizada y ajustar la misma a fin que el usuario pueda visualizar el procesamiento de la misma en cada una de sus fases.

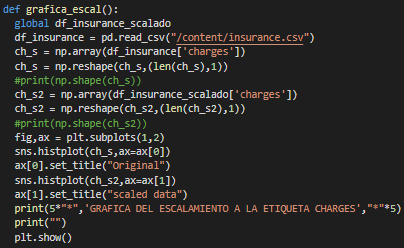


El resultado obtenido es el siguiente:

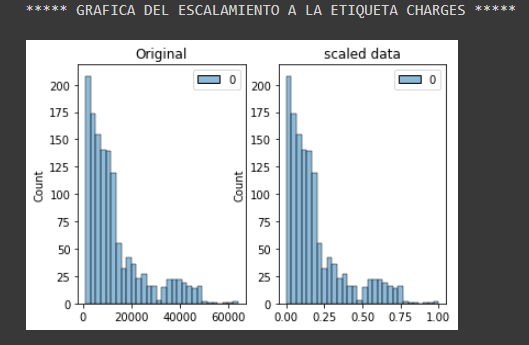


* + 1. **Grafica**

Una ves escalada la data procedemos nuevamente a llamar a la variable global *df\_insurance\_scalado* para poder mostrar método *histplot* la gráfica de la data tratada. En este caso gracias a *numpy* realizamos un *reshape* a la data y *utilizamos pandas* para llamar al dataframe. Aplicamos *array* a la variable global *df\_insurance\_scalado* solo al apartado *charges* con el fin de utilizar como variable de referencia los cargos por seguro de vida. Se plotean tanto la data original como la data tratada para visualizar los cambios y finalmente se le agregan las etiquetas respectivas con el fin de guiar al usuario dentro la lectura gráfica.

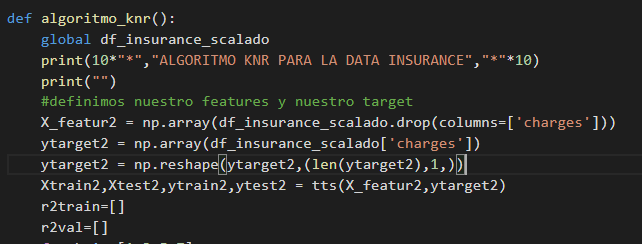


El resultado es el siguiente:

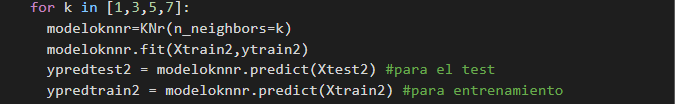


* 1. **Algoritmo KNR**

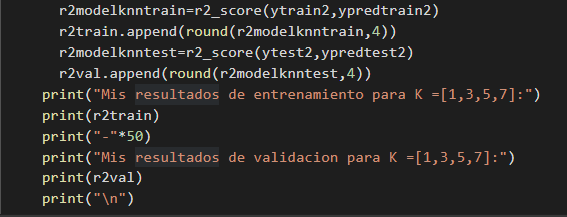
Iniciamos definiendo los *features* y *target* de nuestra data, en un inicio llamamos a la data escalada anteriormente y la definimos como una variable global: *df\_insurance\_scalado*. Dentro de la definición de nuestro *features* utilizamos el método *numpy* para dropear los datos correspondientes a los cargos por prima de seguro: *charges*, como variable *X\_featur2* y como target solo la columna *charges* sin escalar como variable *ytarget2*. Como esta última necesita un *reshape*, se le realiza con el método del mismo nombre para poder iniciar correctamente con el entrenamiento. Finalmente separamos la data ya homogeneizada.



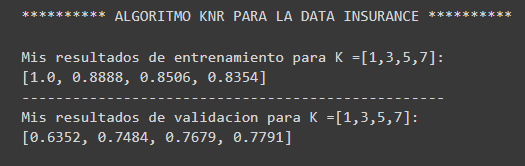
Como paso siguiente utilizamos el hiperparámetro KNr(n\_neighbors = k ) a utilizar dentro del modelo *KRN* y estos son: 1,3,5 y 7; definimos nuestras variables locales para poder realizar el test: *ypredtes2* y *ypredtrain2* para realizar el entrenamiento.



Empezamos el entrenamiento dentro de una variable local: *r2modelknntrain* que llama mediante el método *core* a las variables *ytrain2* y *ypredtrain2,* seguido del redondeo de los datos a fin de mostrar el resultado más limpio posible

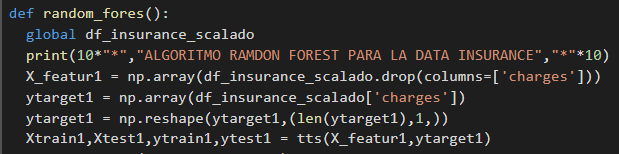


El resultado es el siguiente:



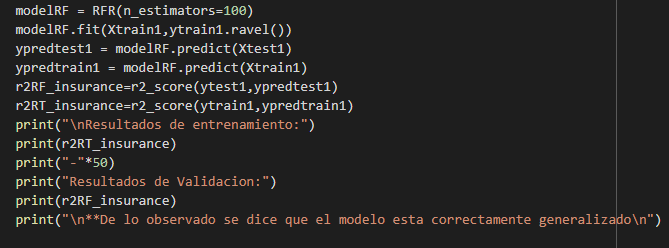
* 1. **Algoritmo Random Forest**

En este apartado volvemos a llamar a nuestra variable global *df\_insurance\_scalado* para trabajar con su data. Iniciamos definiendo los *features* y el *target*, en esta ocasión las variables a utilizar son *X\_featur1* para el *feature* y *ytarget1* para *el target.* Tal como en el modelo anterior realizamos el proceso de *reshape* para la data utilizando el método *reshape* a la variable local *ytarget1* y procedemos a prepararnos para el entrenamiento de la data, no sin antes separar la data ya homogeneizada.

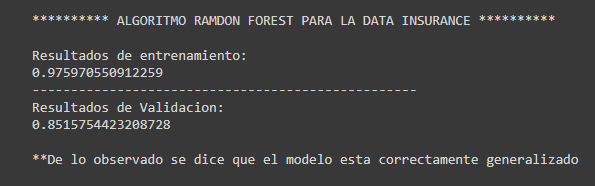


Como paso siguiente generamos el modelo Random Forest para el *feature* y *target* anteriormente utilizado. En esta ocasión creamos una variable local llamada: *modelRF* definidos con un estimador de 100, para luego ajustar la data (mediante el método *fit*) a las variables de entrenamiento mostradas en la imagen anterior (*Xtrain, Xtest, ytrain1 y ytest1*).

Creamos las variables locales *yprestest1 y ypredtrain1* para aplicarles el método de predicción usando la variable local *modelRF* y, finalmente, mostramos la información del modelo utilizando una nueva variable local llamada *r2RF\_insurance.* Esta última mostrará los resultados el entrenamiento de la data y la validación que tendrá basados en el modelo.

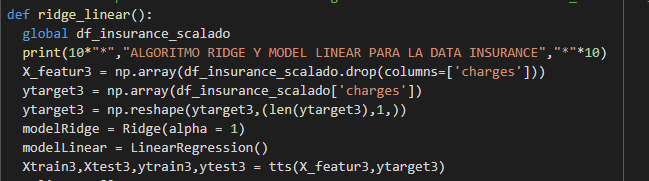


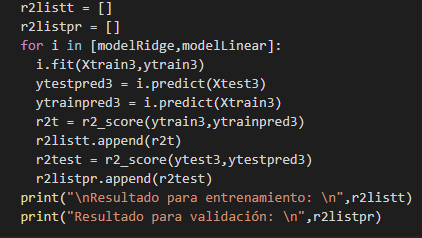
El resultado es el siguiente. Como se observa, el modelo está correctamente generalizado.



* 1. **Algoritmo Lasso y Linearmodel**

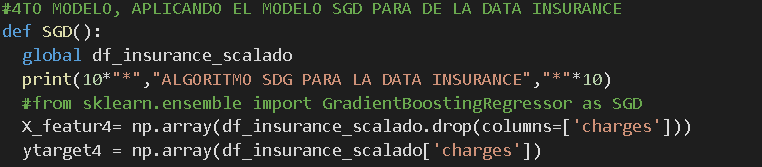
Para la realización de este modelo definimos *ridge\_lineal* llamando a nuestra variable global *df\_insurance\_scalado*. Creamos una variable local *X\_featur3* para el *feature* correspondiente y *ytarget3* para el target a utilizar en este método. Nuevamente, como se mencionó, se realiza el proceso de *reshape* para poder homogeneizar la data a utilizar, convirtiendo a nuestro vector a (n,1). Creamos, en este caso 2 nuevas variables para el modelo a utilizar, la cuales se definen como: *modelRidge* (llamando al método *Ridge)* y *modelLinear* (llamando al modelo *LinearRegression)*





* 1. **Algoritmo SGD**

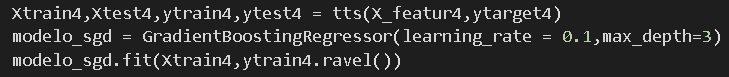
Utilizando nuevamente la variable global *df\_insurance\_*scalado trabajamos con la data bajo los requerimientos del modelo SGD. Iniciamos definiendo nuestros *features* y *target* de nuestra data ya procesada y creamos las variables locales *x\_feature4* y *ytarget4*



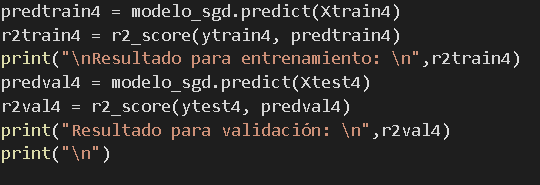
Realizamos la verificación homogeneidad de la data a utilizar:



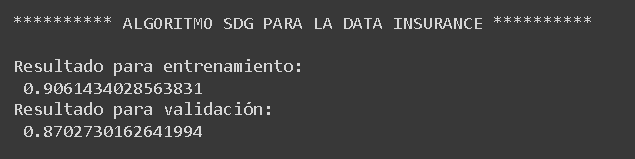
Dividimos la data y entrenamos al modelo utilizando las variables locales *Xtrain4* y *ytrain4*



Predecimos los respectivos resultados para nuestros *Xtrain4* que corresponde a entrenamiento y validación. Finalmente, llamamos al método *r2\_score* para las variables locales *ytrain* y *predtrain4* para que, dentro de otra variable local llamada: *predval4,* permita llamar al modelo SGD mediante el método *modelo\_sgd.predict* utilizando la variable *Xtest4.*

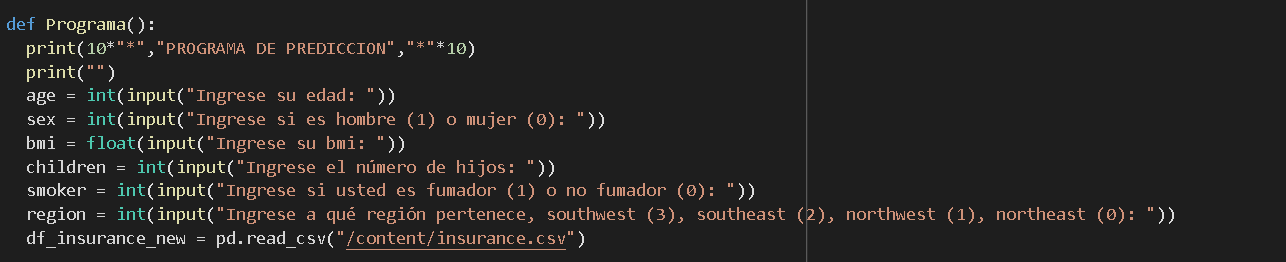


Se puede comprobar que se nuestro modelo tiene una predicción correcta y fiable, por tanto, afirmamos que la data está generalizada.

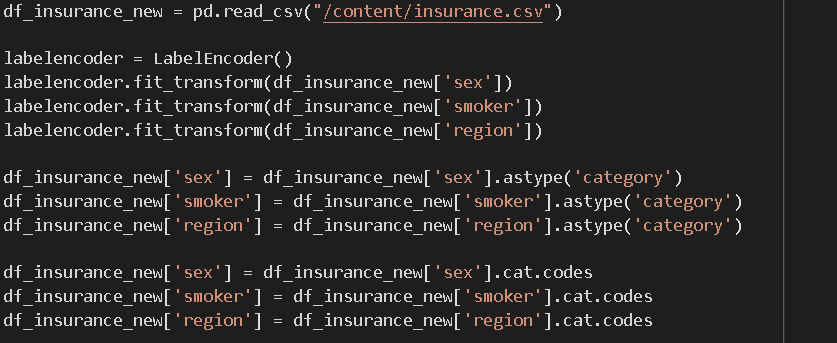


* 1. **Programa de predicción**

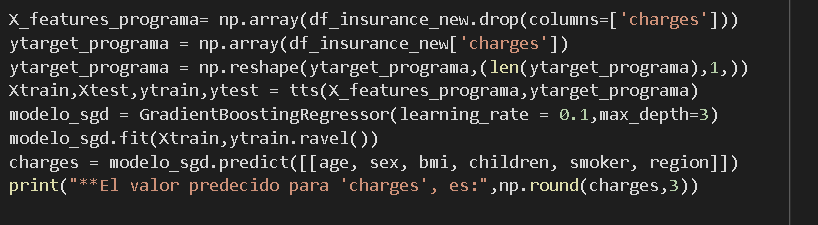
Realizamos la llamada de los *features* mediante variables locales *age, sex, bmi* (índice de masa corporal)*, children (*para conocer cuantos hijos tiene cada persona), *smoker (*para conocer si el sujeto es fumador o no) y *región.*  Todos estos datos deberán ser ingresados por el usuario. Solo la variable *bmi* es de tipo flotante en este caso.



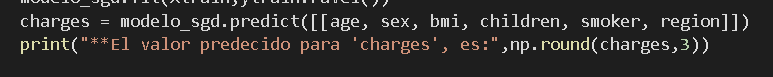
Definimos la data , para este caso usaremos el *dataframe* inicial: *insurance.csv* y para las variables tipo *string* le aplicaremos el método *labelencoder*. Finalmente, reemplazamos por los nuevos datos en nuestra data inicial utilizando *“df\_insurance\_new”.*



Dividimos nuestros *features* y *target* de la data y procedemos a entrenar el modelo con el que obtuvimos mejores resultados, el modelo SDG. Como el usuario puede modificar la data y reutilizar todos y cada uno de los modelos utilizados conforme se utilice mayor cantidad de datos, el modelo mejorará en su predicción. Creamos las variables locales *x\_features\_programa y ytarget\_programa* para empezar la homogeneización de la data, para luego dividirla y empezar su ajuste mediante el método *fit*

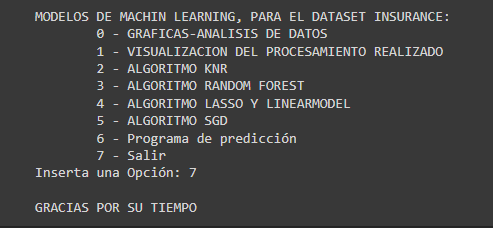


Finalmente, precedimos el valor de la etiqueta *charges* para conocer cuánto de prima de seguro puede alcanzar una persona con las características antes mencionadas.



* 1. **Salir**

La última opción le permite al usuario salir del programa de forma segura y sencilla de la siguiente manera:



1. **Resultados del uso de Python**

Habiendo mostrado la funcionalidad del programa al mismo tiempo que los resultados obtenidos, podemos afirmar que el uso de Python y sus correspondientes librerías nos ayuda a mejorar la comprensión de los procesos matemáticos y estadísticos dentro del análisis de datos. La herramienta, en esta ocasión, ha probado ser de mucha utilidad debido al uso de una interfaz simple y didáctica que permite agilizar la obtención de datos; sin dejar de lado el uso de modelos de machine learning que se han utilizado para cumplir el objetivo principal: Poder predecir los costos que tiene la prima de seguro basado en las características del asegurado.

1. **Conclusiones**

Se concluye que el machine learning resulta muy útil debido a que es capaz de analizar gran cantidad de variables de una forma rápida. Además, el uso de modelos de machine learning permite aprender cada uno de estos, para que de esa manera cada uno realice predicciones cada vez más efectivas, dándonos mayor seguridad al momento de tomar una decisión y saber el correcto plan de acción a usar. Para nuestro caso, analizar el costo que tendría un seguro de vida, dependemos de factores como la edad, genero, peso, cantidad de hijos, ciertas adicciones que afectan a la salud. En términos de utilidad, esta herramienta nos ha dado diversos resultados, los cuales dependerán de cada sujeto analizado el obtener un costo de prima de seguro aproximado, que les permita analizar su situación y tomar las decisiones adecuadas. Utilizando el proyecto antes realizado, afirmamos que el machine learning, nos es de mucha utilidad debido a que hace uso de base de datos, los cuales crecen de forma muy exponencial, el uso del machine learning nos permite sacarle provecho de la mejor manera a las redes de comunicación con el fin poder brindar una predicción certera, confiable y verás. Las empresas pueden beneficiarse mucho, ya que pueden hacer un análisis del por qué se van sus usuarios, por ejemplo, y de esta manera tomar la decisión más adecuada para que sus clientes no se retiren.

1. **Fuentes bibliográficas**

Rotta, P. (2002, 27 febrero). PythonIntermedio2022/clase4 at main · pedrorotta/PythonIntermedio2022. GitHub. Recuperado 5 de febrero de 2022, de https://github.com/pedrorotta/PythonIntermedio2022/tree/main/clase4

Rotta, P. (2022a, enero 24). PythonIntermedio2022/clase1 at main · pedrorotta/PythonIntermedio2022. GitHub. Recuperado 5 de febrero de 2022, de https://github.com/pedrorotta/PythonIntermedio2022/tree/main/clase1

Rotta, P. (2022b, enero 24). PythonIntermedio2022/Lec2\_Plotly.ipynb at main · pedrorotta/PythonIntermedio2022. GitHub. Recuperado 5 de febrero de 2022, de https://github.com/pedrorotta/PythonIntermedio2022/blob/main/Lec2\_Plotly.ipynb

Rotta, P. (2022c, enero 25). PythonIntermedio2022/clase2 at main · pedrorotta/PythonIntermedio2022. GitHub. Recuperado 5 de febrero de 2022, de https://github.com/pedrorotta/PythonIntermedio2022/tree/main/clase2

Rotta, P. (2022d, enero 25). PythonIntermedio2022/Lectura3\_KNN.ipynb at main · pedrorotta/PythonIntermedio2022. GitHub. Recuperado 5 de febrero de 2022, de https://github.com/pedrorotta/PythonIntermedio2022/blob/main/Lectura3\_KNN.ipynb

Rotta, P. (2022e, enero 26). PythonIntermedio2022/clase 3 at main · pedrorotta/PythonIntermedio2022. GitHub. Recuperado 5 de febrero de 2022, de https://github.com/pedrorotta/PythonIntermedio2022/tree/main/clase%203

Rotta, P. (2022f, enero 27). PythonIntermedio2022/Lec4\_supervisados(II).ipynb at main · pedrorotta/PythonIntermedio2022. GitHub. Recuperado 5 de febrero de 2022, de https://github.com/pedrorotta/PythonIntermedio2022/blob/main/Lec4\_supervisados(II).ipynb

Rotta, P. (2022g, enero 27). PythonIntermedio2022/Lec4\_supervisados.ipynb at main · pedrorotta/PythonIntermedio2022. GitHub. Recuperado 5 de febrero de 2022, de https://github.com/pedrorotta/PythonIntermedio2022/blob/main/Lec4\_supervisados.ipynb

Rotta, P. (2022h, enero 27). PythonIntermedio2022/Lec5\_Supervisados2.ipynb at main · pedrorotta/PythonIntermedio2022. GitHub. Recuperado 5 de febrero de 2022, de https://github.com/pedrorotta/PythonIntermedio2022/blob/main/Lec5\_Supervisados2.ipynb

Rotta, P. (2022i, enero 31). PythonIntermedio2022/clase6 at main · pedrorotta/PythonIntermedio2022. GitHub. Recuperado 5 de febrero de 2022, de https://github.com/pedrorotta/PythonIntermedio2022/tree/main/clase6

Rotta, P. (2022j, enero 31). PythonIntermedio2022/Lec6\_Preprocessing.ipynb at main · pedrorotta/PythonIntermedio2022. GitHub. Recuperado 5 de febrero de 2022, de https://github.com/pedrorotta/PythonIntermedio2022/blob/main/Lec6\_Preprocessing.ipynb

Rotta, P. (2022k, febrero 1). PythonIntermedio2022/clase7 at main · pedrorotta/PythonIntermedio2022. GitHub. Recuperado 5 de febrero de 2022, de https://github.com/pedrorotta/PythonIntermedio2022/tree/main/clase7

Rotta, P. (2022l, febrero 1). PythonIntermedio2022/Lec7\_Nosupervisados.ipynb at main · pedrorotta/PythonIntermedio2022. GitHub. Recuperado 5 de febrero de 2022, de https://github.com/pedrorotta/PythonIntermedio2022/blob/main/Lec7\_Nosupervisados.ipynb

Rotta, P. (2022m, febrero 3). PythonIntermedio2022/clase9 at main · pedrorotta/PythonIntermedio2022. GitHub. Recuperado 5 de febrero de 2022, de https://github.com/pedrorotta/PythonIntermedio2022/tree/main/clase9

Rotta, P. (2022n, febrero 4). PythonIntermedio2022/Lec9\_Tensorflow.ipynb at main · pedrorotta/PythonIntermedio2022. GitHub. Recuperado 5 de febrero de 2022, de https://github.com/pedrorotta/PythonIntermedio2022/blob/main/Lec9\_Tensorflow.ipynb

Rotta, P. (2022o, febrero 6). PythonIntermedio2022/clase10 at main · pedrorotta/PythonIntermedio2022. GitHub. Recuperado 6 de febrero de 2022, de https://github.com/pedrorotta/PythonIntermedio2022/tree/main/clase10

Rotta, P. (2022p, febrero 28). PythonIntermedio2022/clase5 at main · pedrorotta/PythonIntermedio2022. GitHub. Recuperado 5 de febrero de 2022, de https://github.com/pedrorotta/PythonIntermedio2022/tree/main/clase5