Módulo 2 - Práctica 4

Descubrimientos

Al realizar esta práctica, se nos pidió hacer un modelo de clasificación para determinar si una persona iba a tener un “derrame cerebral” dadas ciertas características de Salud.

Para poder realizar esto, tenemos que ver que nuestros datos estén de buena manera. A lo que nos dimos cuenta que el “bmi” tenía valores nulos, los cales representaban menos del 5% del total, a lo que procedimos a borrarlos.

```
In [8]: #We're dropping these NA values as they are less <5%
        df[df["bmi"].isna()].shape[0]/df.shape[0]*100
```

executed in 2ms, finished 13:50:15 2022-09-24

```
Out[8]: 3.9334637964774952
```

Fuera de eso, nuestro dataset no tenía grandes problemas con sus datos.

Variables Discretas

Para la Exploración y procesamiento de variables discretas, nos dimos cuenta de dos puntos:

- En el género del usuario, tenemos un 0.02% de datos que tienen un género de otros, los eliminamos.
- Tenemos un 0.4% de los datos que en la variable “work_type” los usuarios nunca trabajaron. Eliminamos esas coincidencias

Para el resto de nuestras variables categóricas, procedimos a crear variables dummy, las cuales nos ayudarían de una manera sencilla a hacer nuestro modelo.

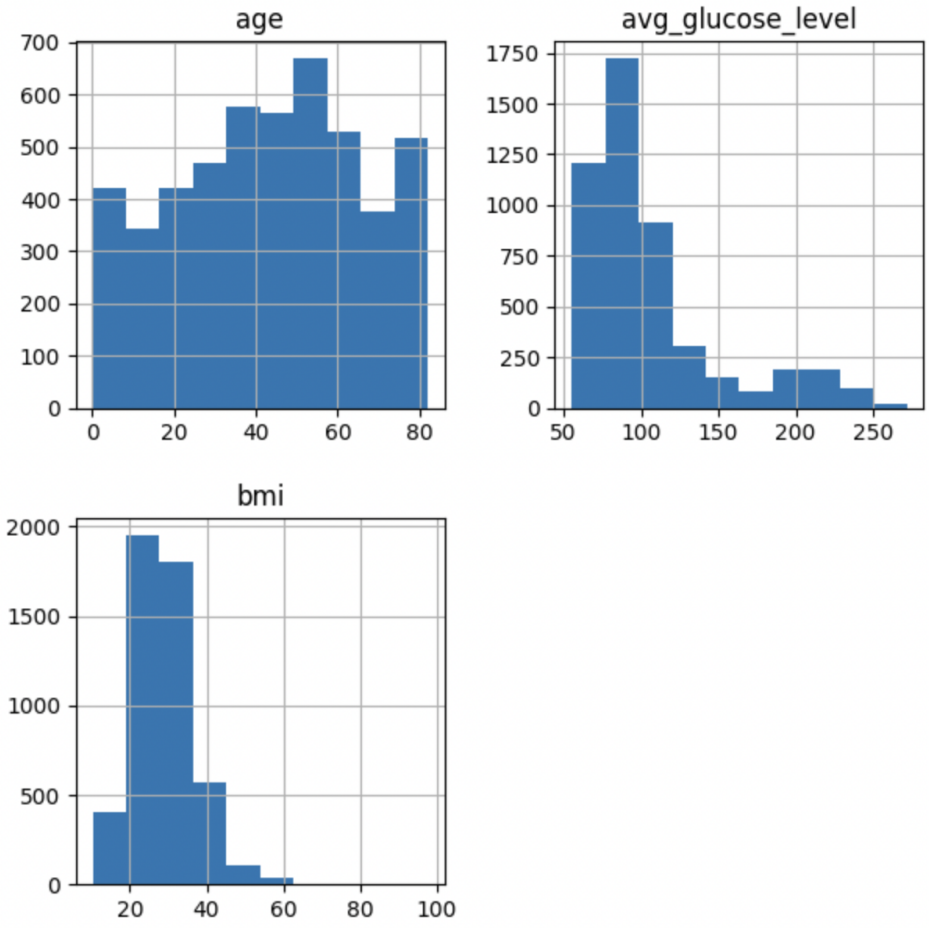
Variables Continuas

Para las variables continuas, hicimos un poco más de tratamiento. Al hacer el análisis de percentiles, nos dimos cuenta que hay algo de diferencia entre el 99% percentil y el valor máximo del BMI.

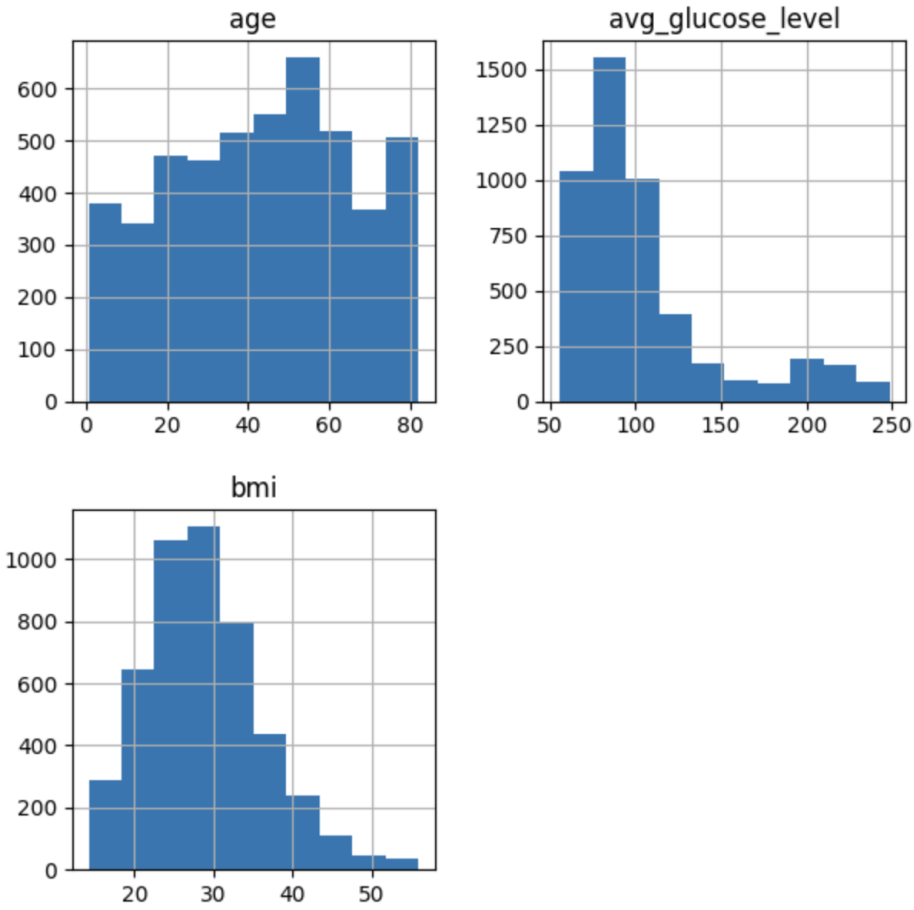
Out[19]:

	count	mean	std	min	1%	10%	50%	90%	99%	max
age	4886.0	42.988973	22.534968	0.08	1.0800	10.000	45.00	74.000	82.000	82.00
avg_glucose_level	4886.0	105.339073	44.481347	55.12	56.3185	65.645	91.68	187.495	240.608	271.74
bmi	4886.0	28.909640	7.853621	10.30	15.1000	19.700	28.10	38.900	53.415	97.60

Al analizar de una forma más gráfica, podemos darnos cuenta que, aunque las distribuciones no se ven mal, las podemos mejorar.



Para poder hacer esto, quitamos los outliers que tengamos dado la cerca percentil. Para este ejercicio, ocupamos 0.005 y el 0.995 percentil. Luego de quitar esos datos, obtenemos las siguientes distribuciones.



Árbol de Decisión

Para poder hacer el árbol de decisión, elegimos usar el *DecisionTreeClassifier* que se encuentra dentro de la librería Scikit-Learn. Luego escalamos nuestras variables continuas.

Procedimos a entrenar y luego ver las métricas de nuestro árbol de decisión, lo cual nos dimos cuenta que no geeralizaba. Procedemos a hacer un randomized Search

```
In [36]: tree_metrics(Tree_Classifier,Xt_t, Yt_t)
```

executed in 6ms, finished 13:50:16 2022-09-24

```
Roc Validate: 1.000
Acc Validate: 1.000
Matrix Conf Validate:
[[3198    0]
[   0  140]]
```

```
In [37]: tree_metrics(Tree_Classifier,Xt_v, Yt_v)
```

executed in 5ms, finished 13:50:16 2022-09-24

```
Roc Validate: 0.542
Acc Validate: 0.920
Matrix Conf Validate:
[[1308   61]
[  54    8]]
```

Hiper Parámetros

Como nos dimos cuenta que nuestro árbol inicial se sobre ajustaba, empezamos a usar una búsqueda con los mejores parámetros por medio de un Randomized Search. Una vez hecho esto, nuestro modelo sí mejoró, pero no parecía ser la mejor opción; ya que, teníamos falsos negativos considerables.

```
In [42]: tree_metrics(best_tree,Xt_t,Yt_t)
```

executed in 6ms, finished 13:50:18 2022-09-24

```
Roc Validate: 0.811
Acc Validate: 0.958
Matrix Conf Validate:
[[3198    0]
[ 140    0]]
```

```
In [43]: tree_metrics(best_tree,Xt_v,Yt_v)
```

executed in 5ms, finished 13:50:18 2022-09-24

```
Roc Validate: 0.804
Acc Validate: 0.957
Matrix Conf Validate:
[[1369    0]
[  62    0]]
```

Red Neuronal

Como nos dimos cuenta que nuestro árbol de decisión parecía ser una no tan buena solución, pasamos a hacer nuestra red neural. En este caso, usamos el Multilayer Perceptron Classifier (*MLPClassifier*) dentro de la paquetería de Scikit-learn.

De igual manera que en nuestro árbol, para las redes neuronales escalamos nuestras variables continuas y procedimos a entrenar.

```
In [51]: nn_metrics(mlp,Xnn_t,Ynn_t)
```

executed in 25ms, finished 13:50:20 2022-09-24

```
Acc Validate: 0.965
Matrix Conf Validate:
[[3198    0]
 [ 116   24]]
```

```
In [52]: nn_metrics(mlp,Xnn_v,Ynn_v)
```

executed in 25ms, finished 13:50:20 2022-09-24

```
Acc Validate: 0.952
Matrix Conf Validate:
[[1360    9]
 [  60    2]]
```

Con este modelo inicial, empezamos a tener mejores resultados que en el árbol, pero tenemos un número considerable de falsos positivos en nuestro set de validación, a lo cual hacemos nuestra búsqueda de hiper parámetros.

Hiper Parámetros

Al hacer la búsqueda con *RandomizedSearch* con 1000 iteraciones, nos dimos cuenta que nuestro modelo no mejoraba considerablemente con respecto al modelo inicial. Pero nos quedamos con esta opción.

```
In [59]: nn_metrics(model_best,Xnn_t,Ynn_t)
```

executed in 39ms, finished 13:50:30 2022-09-24

```
Acc Validate: 0.959
Matrix Conf Validate:
[[3196    2]
[ 135    5]]
```

```
In [60]: nn_metrics(model_best,Xnn_v,Ynn_v)
```

executed in 37ms, finished 13:50:30 2022-09-24

```
Acc Validate: 0.955
Matrix Conf Validate:
[[1366    3]
[  61    1]]
```

Conclusiones

Nuestra mejor opción actual es usar una red neuronal dadas las métricas que se obtuvieron en los diferentes análisis. Para poder ir mejorando el modelo, en un futuro se podría:

- Crear más variables en el dataset.
- No borrar las respuestas que platicamos con anterioridad.