```
Eliud Garza A00827575
```

Eliud Garza A00827575

```
from google.colab import drive
drive.mount("/content/gdrive")

!pwd
```

```
Mounted at /content/gdrive
/content
```

```
%cd "/content/gdrive/MyDrive/7mo Semestre/Modulo 2"

!ls
```

```
/content/gdrive/MyDrive/7mo Semestre/Modulo 2
 brain_stroke.csv     'Momento de Retro: Modulo 2'   PlayDataset.csv
 mc-donalds-menu.csv  'Neural Network.ipynb'         Valhalla23.csv
```

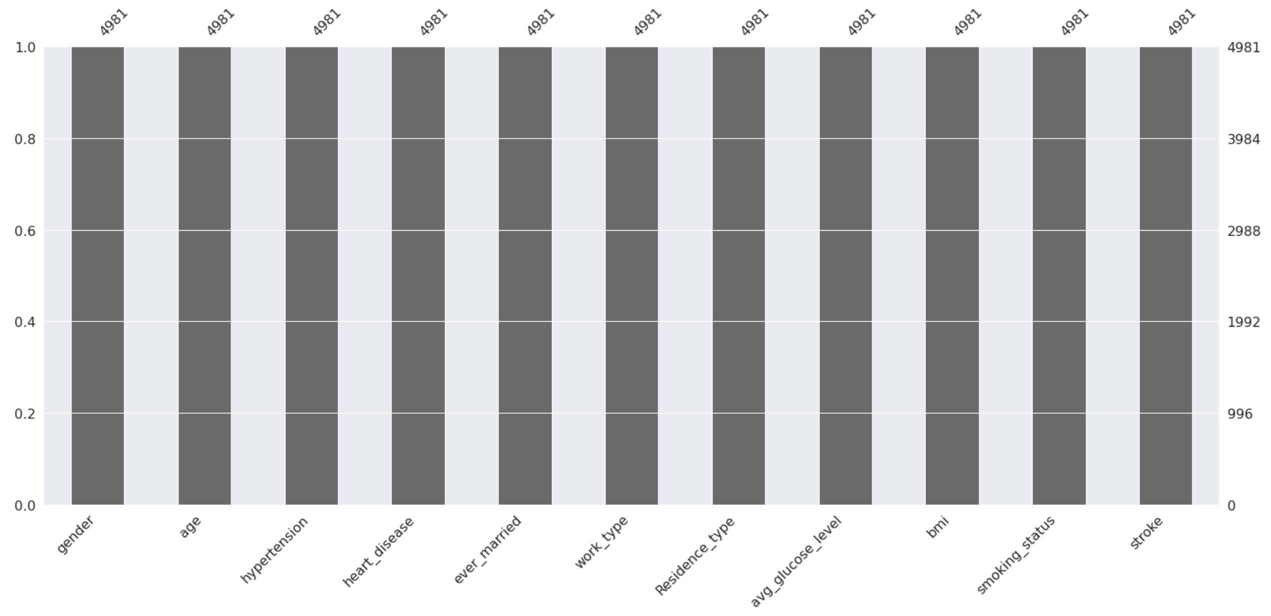
```
import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import missingno as msno
from sklearn.model_selection import train_test_split
```

```
dset = pd.read_csv("brain_stroke.csv")

dset.head(5000)
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residenc
0	Male	67.0	0	1	Yes	Private	
1	Male	80.0	0	1	Yes	Private	
2	Female	49.0	0	0	Yes	Private	
3	Female	79.0	1	0	Yes	Self-employed	
4	Male	81.0	0	0	Yes	Private	

```
msno.bar(dset)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f13d153f2d0>
```



```
dset.info()
```

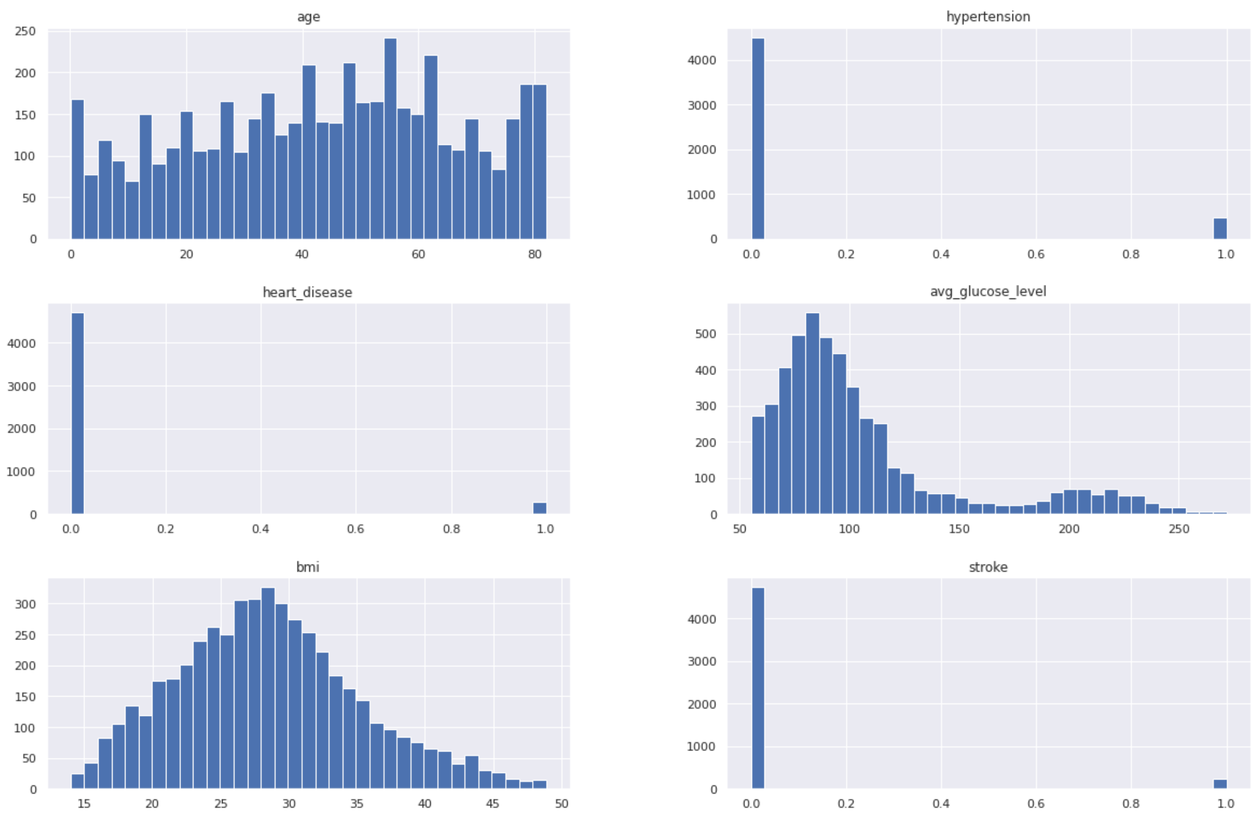
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4981 entries, 0 to 4980
```

```
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   gender             4981 non-null   object 
 1   age                4981 non-null   float64
 2   hypertension       4981 non-null   int64  
 3   heart_disease      4981 non-null   int64  
 4   ever_married       4981 non-null   object 
 5   work_type          4981 non-null   object 
 6   Residence_type     4981 non-null   object 
 7   avg_glucose_level  4981 non-null   float64
 8   bmi                4981 non-null   float64
 9   smoking_status     4981 non-null   object 
 10  stroke             4981 non-null   int64  
dtypes: float64(3), int64(3), object(5)
memory usage: 428.2+ KB
```

```
dset.describe()
```

	age	hypertension	heart_disease	avg_glucose_level	bmi	
count	4981.000000	4981.000000	4981.000000	4981.000000	4981.000000	4
mean	43.419859	0.096165	0.055210	105.943562	28.498173	
std	22.662755	0.294848	0.228412	45.075373	6.790464	
min	0.080000	0.000000	0.000000	55.120000	14.000000	
25%	25.000000	0.000000	0.000000	77.230000	23.700000	
50%	45.000000	0.000000	0.000000	91.850000	28.100000	
75%	61.000000	0.000000	0.000000	113.860000	32.600000	
max	82.000000	1.000000	1.000000	271.740000	48.900000	

```
dset.hist(bins=35, figsize=(20,13))
plt.show()
```



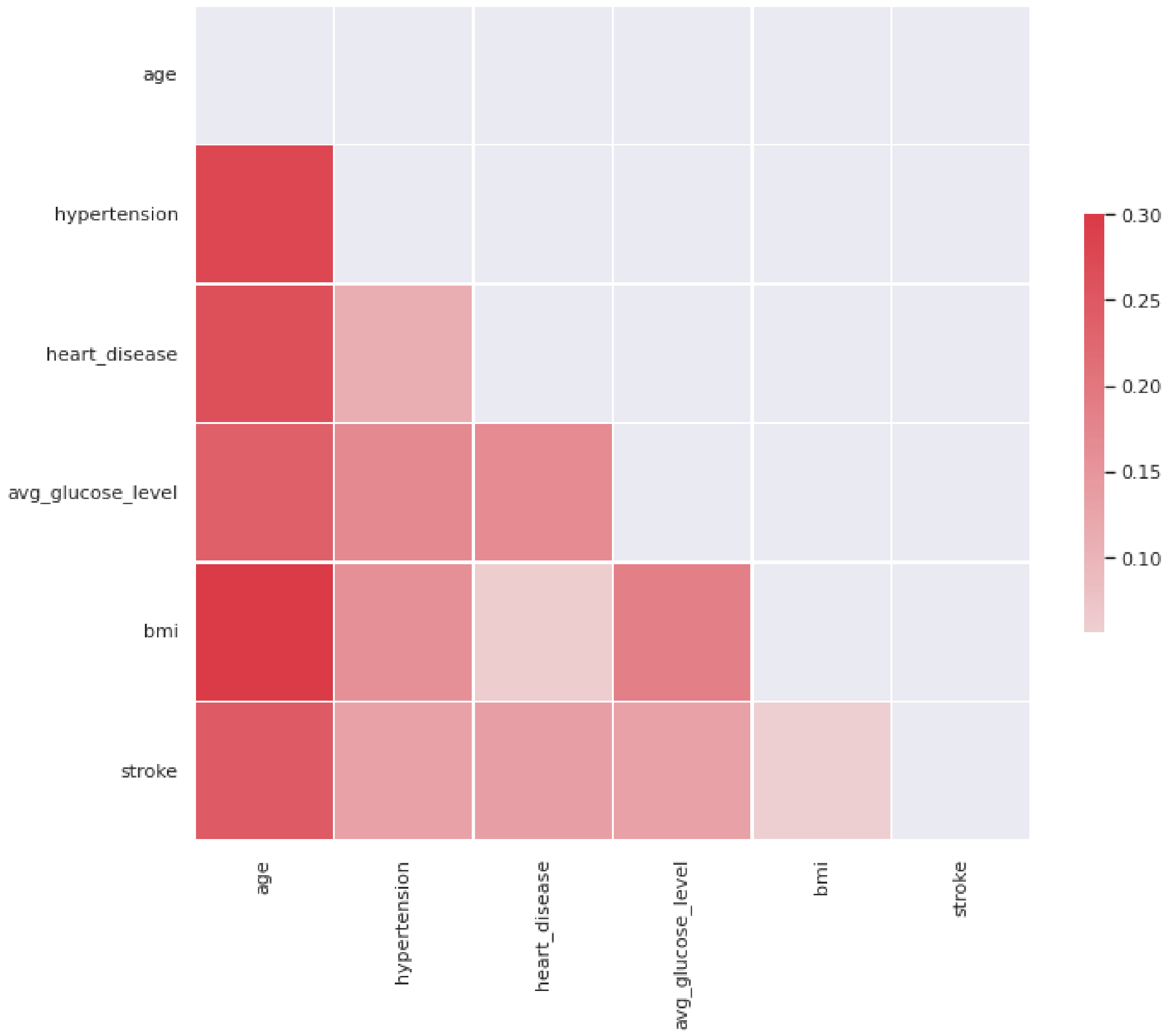
```
cor = dset.corr()
cor
```

	age	hypertension	heart_disease	avg_glucose_level	b
age	1.000000	0.278120	0.264852	0.236763	0.3737
hypertension	0.278120	1.000000	0.111974	0.170028	0.1587
heart_disease	0.264852	0.111974	1.000000	0.166847	0.0609
avg_glucose_level	0.236763	0.170028	0.166847	1.000000	0.1863
bmi	0.373703	0.158762	0.060926	0.186348	1.0000
stroke	0.246478	0.131965	0.134610	0.133227	0.0569

```
# Generate a mask for the upper triangle
mask = np.triu(np.ones_like(cor, dtype=bool))

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
```

```
cmap = sns.diverging_palette(200, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(cor, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

```
     <matplotlib.axes._subplots.AxesSubplot at 0x7f13d102dcd0>
```



```
clean_cat = {"gender": {"Male":0, "Female": 1}, "ever_married": {"No":0,"Yes":1}, "work_type"

dset = dset.replace(clean_cat)

dset.head()
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_t
0	0	67.0	0	1	1	0	
1	0	80.0	0	1	1	0	

```
X = dset.drop(["stroke"], axis = 1)
Y = dset["stroke"]

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.33, random_state=42)
```

```
from sklearn.metrics import SCORERS
from sklearn.tree import DecisionTreeClassifier

decisionTree = DecisionTreeClassifier(max_depth = 4, min_samples_split = 2, min_samples_leaf
decisionTree.fit(X_train, y_train)
Y_pred = decisionTree.predict(X_test)
score = round(decisionTree.score(X_train, y_train) * 100, 2)
print('Decision Tree Classifier predicts with', score, '% of accuracy')
```

```
Decision Tree Classifier predicts with 95.15 % of accuracy
```