Nhom09.part1

July 11, 2021

BÀI TẬP LỚN MÔN HỌC MACHINE LEARNING

K3- KHOA HỌC DỮ LIỆU

Nhóm 9:

Nguyễn Văn Thế

Lê Trung Hiếu

Nguyễn Thị Kim Duyên

1 Introduction

Bộ dữ liệu Stroke Prediction:

1. Thu thập dữ liệu:

- Theo Tổ chức Y tế Thế giới (WHO) đột quỵ là nguyên nhân gây tử vong thứ 2 trên toàn cầu, chiếm khoảng 11% tổng số ca tử vong. Bộ dữ liệu này được sử dụng để dự đoán liệu một bệnh nhân có khả năng bị đột quỵ hay không dựa trên các thông số đầu vào như giới tính, tuổi tác, các bệnh khác nhau và tình trạng hút thuốc. Mỗi hàng trong bộ dữ liệu cung cấp thông tin liên quan về bệnh nhân.
- Bộ dữ liệu được thu thập từ trang web: https://www.kaggle.com/fedesoriano/stroke-prediction-dataset
- Bộ dữ liệu gồm 5110 quan sát với 12 thuộc tính

2. Các trường dữ liệu:

- id:
  - Là số nhận dạng (ID) của bệnh nhân
  - Là biến định lượng liên tục kiểu số
- gender:
  - Giới tính bệnh nhân
  - Là biến định tính rời rạc kiểu String
  - Nhận 1 trong các giá trị: "Male", "Female" or "Other"
- age:

  - Tuổi của bệnh nhân
  - Là biến định lượng liên tục kiểu số

- hypertension:
  - Chứng cao huyết áp của bệnh nhân
  - Là biến định tính rời rạc kiểu số
  - Nhận 1 trong 2 giá trị: 0 nếu bệnh nhân không bị cao huyết áp và 1 nếu bệnh nhân mắc cao huyến áp

- heart_disease:
  - Tình trạng đau tim của bệnh nhân
  - Là biến định tính rời rạc kiểu số
  - Nhận 1 trong 2 giá trị: 0 nếu bệnh nhân không bị đau tim, và 1 nếu bệnh nhân bị đau tim

- ever_married:
  - Tình trạng hôn nhân của bệnh nhân
  - Là biến định tính rời rạc kiểu String
  - Nhận 1 trong các giá trị: "No" nếu chưa kết hôn hoặc "Yes" nếu đã kết hôn

- work_type:
  - Loại nghề nghiệp của bệnh nhân
  - Là biến định tính rời rạc kiểu String
  - Nhận 1 trong các giá trị: "children", "Govt_jov", "Never_worked", "Private" hoặc "Self-employed"

- Residence_type:
  - Nơi cư trú của bệnh nhân
  - Là biến định tính rời rạc kiểu String
  - Nhận 1 trong các giá trị: "Rural" hoặc "Urban"

- avg_glucose_level:
  - Lượng đường trung bình trong máu của bệnh nhân
  - Là biến định lượng liên tục kiểu số

- bmi:
  - Chỉ số đo lường cơ thể BMI của bệnh nhân
  - Là biến định lượng liên tục kiểu số

- smoking_status:
  - Tình trạng hút thuốc của bệnh nhân
  - Là biến định tính rời rạc kiểu String
  - Nhận 1 trong các giá trị: "formerly smoked", "never smoked", "smokes" or "Unknown"

- stroke:
  - Tình trạng đột quỵ của bệnh nhân
  - Là biến định tính rời rạc kiểu số

– Nhận 1 trong 2 giá trị: 0 nếu bệnh nhân không bị đột quỵ, và 1 nếu bệnh nhân bị đột quỵ

3. Mục đích phân tích dữ liệu:

Dự đoán đột quỵ ở con người Biến đích (mục tiêu phân tích, dự đoán) là biến "stroke" * Tình trạng đột quỵ của bệnh nhân * Là biến định tính rời rạc kiểu số * Nhận 1 trong 2 giá trị: 0 nếu bệnh nhân không bị đột quỵ, và 1 nếu bệnh nhân bị đột quỵ

1.1 Libraries and Ultilities

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

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

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import itertools
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.ticker as mtick
from matplotlib.colors import ListedColormap, LinearSegmentedColormap

%matplotlib inline
import matplotlib

#Common model helpers
from sklearn.preprocessing import (StandardScaler,
                                   LabelEncoder,
                                   OneHotEncoder)
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.tree import DecisionTreeRegressor
```

1.2 Data preprocessing

1.2.1 Loading data

[ ]:
```
df = pd.read_csv('/content/drive/MyDrive/ML&DM/healthcare-dataset-stroke-data.
 ↪csv')
```

[ ]:
```
df.head(10).T
```

[ ]:
```
                                0           1  ...       8        9
id                           9046       51676  ...   27419    60491
gender                       Male      Female  ...  Female   Female
age                            67          61  ...      59       78
```

```
hypertension                      0              0  ...        0        0
heart_disease                     1              0  ...        0        0
ever_married                    Yes            Yes  ...      Yes      Yes
work_type                   Private  Self-employed  ...  Private  Private
Residence_type                Urban          Rural  ...    Rural    Urban
avg_glucose_level            228.69         202.21  ...    76.15    58.57
bmi                            36.6            NaN  ...      NaN     24.2
smoking_status      formerly smoked   never smoked  ...  Unknown  Unknown
stroke                            1              1  ...        1        1

[12 rows x 10 columns]
```

[ ]:
```
# drop id column because it's unnecessary
df.drop('id',axis=1,inplace=True)
```

Attribute Information 1. **id**: unique identifier 2. **gender**: "Male", "Female" or "Other" 3. **age**: age of the patient 4. **hypertension**: 0 if the patient doesn't have hypertension, 1 if the patient has hypertension 5. **heart_disease**: 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease 6. **ever_married**: "No" or "Yes" 7. **work_type**: "children", "Govt_jov", "Never_worked", "Private" or "Self-employed" 8. **Residence_type**: "Rural" or "Urban" 9. **avg_glucose_level**: average glucose level in blood 10. **bmi**: body mass index 11. **smoking_status**: "formerly smoked", "never smoked", "smokes" or "Unknown" *12. **stroke**: 1 if the patient had a stroke or 0 if not* Note: "Unknown" in smoking_status means that the information is unavailable for this patient

[ ]:
```
df.shape
```

[ ]:
```
(5110, 11)
```

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   gender             5110 non-null   object
 1   age                5110 non-null   float64
 2   hypertension       5110 non-null   int64
 3   heart_disease      5110 non-null   int64
 4   ever_married       5110 non-null   object
 5   work_type          5110 non-null   object
 6   Residence_type     5110 non-null   object
 7   avg_glucose_level  5110 non-null   float64
 8   bmi                4909 non-null   float64
 9   smoking_status     5110 non-null   object
 10  stroke             5110 non-null   int64
dtypes: float64(3), int64(3), object(5)
```

```
memory usage: 439.3+ KB
```

[ ]:
```
df.describe()
```

[ ]:
```
               age  hypertension  …          bmi       stroke
count  5110.000000   5110.000000  …  4909.000000  5110.000000
mean     43.226614      0.097456  …    28.893237     0.048728
std      22.612647      0.296607  …     7.854067     0.215320
min       0.080000      0.000000  …    10.300000     0.000000
25%      25.000000      0.000000  …    23.500000     0.000000
50%      45.000000      0.000000  …    28.100000     0.000000
75%      61.000000      0.000000  …    33.100000     0.000000
max      82.000000      1.000000  …    97.600000     1.000000

[8 rows x 6 columns]
```

[ ]:
```
df.isnull().sum()
```

[ ]:
```
gender                 0
age                    0
hypertension           0
heart_disease          0
ever_married           0
work_type              0
Residence_type         0
avg_glucose_level      0
bmi                  201
smoking_status         0
stroke                 0
dtype: int64
```

[ ]:
```
# handling missing values
df['bmi'] = df['bmi'].fillna(df['bmi'].median())
df.isnull().sum()
```

[ ]:
```
gender               0
age                  0
hypertension         0
heart_disease        0
ever_married         0
work_type            0
Residence_type       0
avg_glucose_level    0
bmi                  0
smoking_status       0
stroke               0
dtype: int64
```

1.2.2 Statistics of Categorical and Numerical Data

```
[ ]: # stats of numerical data
     round (df.describe(exclude = 'object'), 2)
```

```
[ ]:           age  hypertension  …      bmi   stroke
     count  5110.00        5110.0  …  5110.00  5110.00
     mean     43.23           0.1  …    28.86     0.05
     std      22.61           0.3  …     7.70     0.22
     min       0.08           0.0  …    10.30     0.00
     25%      25.00           0.0  …    23.80     0.00
     50%      45.00           0.0  …    28.10     0.00
     75%      61.00           0.0  …    32.80     0.00
     max      82.00           1.0  …    97.60     1.00

     [8 rows x 6 columns]
```

```
[ ]: # stats of categorical data
     round (df.describe(exclude = ['float', 'int64']),2)
```

```
[ ]:        gender ever_married work_type Residence_type smoking_status
     count    5110         5110      5110           5110           5110
     unique      3            2         5              2              4
     top    Female          Yes   Private          Urban   never smoked
     freq     2994         3353      2925           2596           1892
```

1.2.3 Initial Insights About Dataset

1. Data from bmi feature is missing.
2. Both categorical and numerical features are present. >* **Categorical Features**: gender, ever_married, work_type, Residence_type, smoking_status >* **Binary Numerical Features**: hypertension, heart_disease, stroke >* **Continous Numerical Features**: age, avg_glucose_level, bmi

```
[ ]: cont_features= ['age','avg_glucose_level','bmi']
     cat_features=␣
      ↪['gender','hypertension','heart_disease','ever_married','work_type','Residence_type','smoki
     target='stroke'
```

body mass index binning: https://www.medicalnewstoday.com/articles/323446#body-mass-index Age binning: https://kidspicturedictionary.com/english-through-pictures/people-english-through-pictures/age-physical-description/ average glucose binning: https://agamatrix.com/blog/normal-blood-sugar-level-chart/

```
[ ]: ## binning of numerical variables

     df['bmi_cat'] = pd.cut(df['bmi'], bins = [0, 19, 25,30,10000], labels =␣
      ↪['Underweight', 'Ideal', 'Overweight', 'Obesity'])
```

```
df['age_cat'] = pd.cut(df['age'], bins = [0,13,18, 45,60,200], labels =␣
 ↪['Children', 'Teens', 'Adults','Mid Adults','Elderly'])
df['glucose_cat'] = pd.cut(df['avg_glucose_level'], bins = [0,90,160,230,500],␣
 ↪labels = ['Low', 'Normal', 'High', 'Very High'])
```

2 Exploring Data Analysis

2.1 Distribution of Targets

[ ]:
```
# Number of each category of the target label
df['stroke'].value_counts()
```

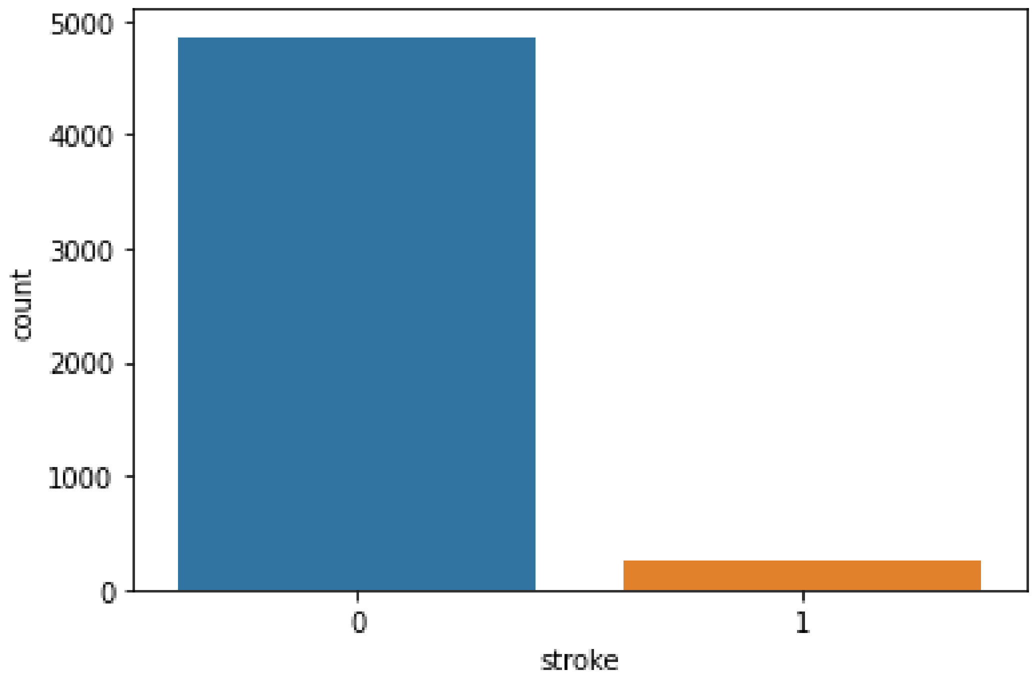
[ ]:
```
0    4861
1     249
Name: stroke, dtype: int64
```

[ ]:
```
# Which category of the target label is how many percentage.
df['stroke'].value_counts()/len(df)*100
```

[ ]:
```
0    95.127202
1     4.872798
Name: stroke, dtype: float64
```

[ ]:
```
sns.countplot(x = "stroke", data = df)
```

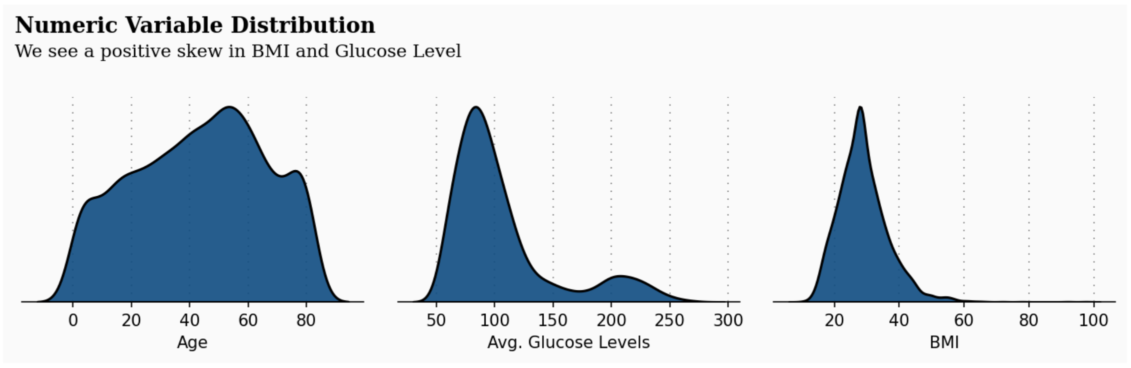
[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8e92a35250>
```



2.1.1 Let's look at the numeric/continuous variable distribtion

[ ]:
```
variables = [variable for variable in df.columns if variable not in ['stroke']]

conts = ['age','avg_glucose_level','bmi']
```

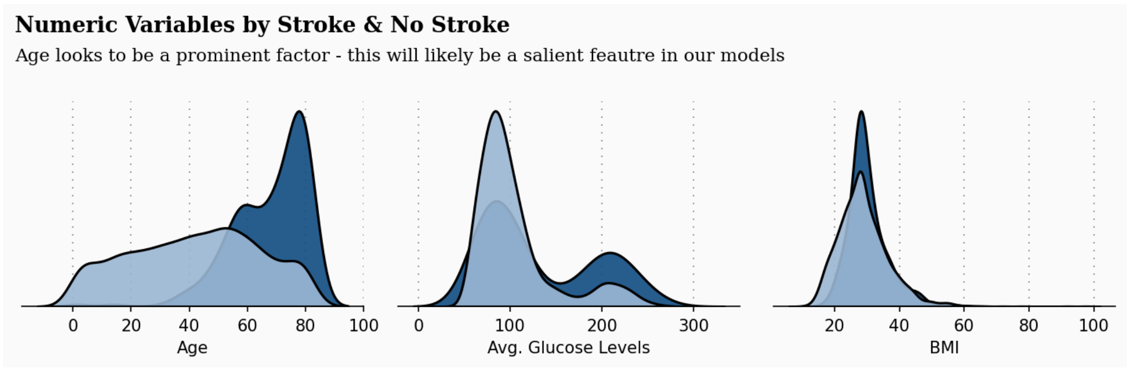
[ ]:
```
fig = plt.figure(figsize=(12, 12), dpi=150, facecolor='#fafafa')
gs = fig.add_gridspec(4, 3)
gs.update(wspace=0.1, hspace=0.4)

background_color = "#fafafa"

plot = 0
for row in range(0, 1):
    for col in range(0, 3):
        locals()["ax"+str(plot)] = fig.add_subplot(gs[row, col])
        locals()["ax"+str(plot)].set_facecolor(background_color)
        locals()["ax"+str(plot)].tick_params(axis='y', left=False)
        locals()["ax"+str(plot)].get_yaxis().set_visible(False)
        for s in ["top","right","left"]:
            locals()["ax"+str(plot)].spines[s].set_visible(False)
        plot += 1

plot = 0
for variable in conts:
        sns.kdeplot(df[variable] ,ax=locals()["ax"+str(plot)], color='#0f4c81',␣
 ↪shade=True, linewidth=1.5, ec='black',alpha=0.9, zorder=3, legend=False)
        locals()["ax"+str(plot)].grid(which='major', axis='x', zorder=0,␣
 ↪color='gray', linestyle=':', dashes=(1,5))
        #locals()["ax"+str(plot)].set_xlabel(variable) removed this for␣
 ↪aesthetics
        plot += 1

ax0.set_xlabel('Age')
ax1.set_xlabel('Avg. Glucose Levels')
ax2.set_xlabel('BMI')


ax0.text(-20, 0.022, 'Numeric Variable Distribution', fontsize=13,␣
 ↪fontweight='bold', fontfamily='serif')
ax0.text(-20, 0.02, 'We see a positive skew in BMI and Glucose Level',␣
 ↪fontsize=11, fontweight='light', fontfamily='serif')

plt.show()
```



Let's see how the distribution of our numeric variables is different for those that have strokes, and those that do not.

```
[ ]: fig = plt.figure(figsize=(12, 12), dpi=150,facecolor=background_color)
gs = fig.add_gridspec(4, 3)
gs.update(wspace=0.1, hspace=0.4)



plot = 0
for row in range(0, 1):
    for col in range(0, 3):
        locals()["ax"+str(plot)] = fig.add_subplot(gs[row, col])
        locals()["ax"+str(plot)].set_facecolor(background_color)
        locals()["ax"+str(plot)].tick_params(axis='y', left=False)
        locals()["ax"+str(plot)].get_yaxis().set_visible(False)
        for s in ["top","right","left"]:
            locals()["ax"+str(plot)].spines[s].set_visible(False)
        plot += 1

plot = 0

s = df[df['stroke'] == 1]
ns = df[df['stroke'] == 0]

for feature in conts:
        sns.kdeplot(s[feature], ax=locals()["ax"+str(plot)], color='#0f4c81',␣
 ↪shade=True, linewidth=1.5, ec='black',alpha=0.9, zorder=3, legend=False)
        sns.kdeplot(ns[feature],ax=locals()["ax"+str(plot)], color='#9bb7d4',␣
 ↪shade=True, linewidth=1.5, ec='black',alpha=0.9, zorder=3, legend=False)
        locals()["ax"+str(plot)].grid(which='major', axis='x', zorder=0,␣
 ↪color='gray', linestyle=':', dashes=(1,5))
        #locals()["ax"+str(plot)].set_xlabel(feature)
        plot += 1
```

```
ax0.set_xlabel('Age')
ax1.set_xlabel('Avg. Glucose Levels')
ax2.set_xlabel('BMI')

ax0.text(-20, 0.056, 'Numeric Variables by Stroke & No Stroke', fontsize=13,␣
↪fontweight='bold', fontfamily='serif')
ax0.text(-20, 0.05, 'Age looks to be a prominent factor - this will likely be a␣
↪salient feautre in our models',
         fontsize=11, fontweight='light', fontfamily='serif')

plt.show()
```



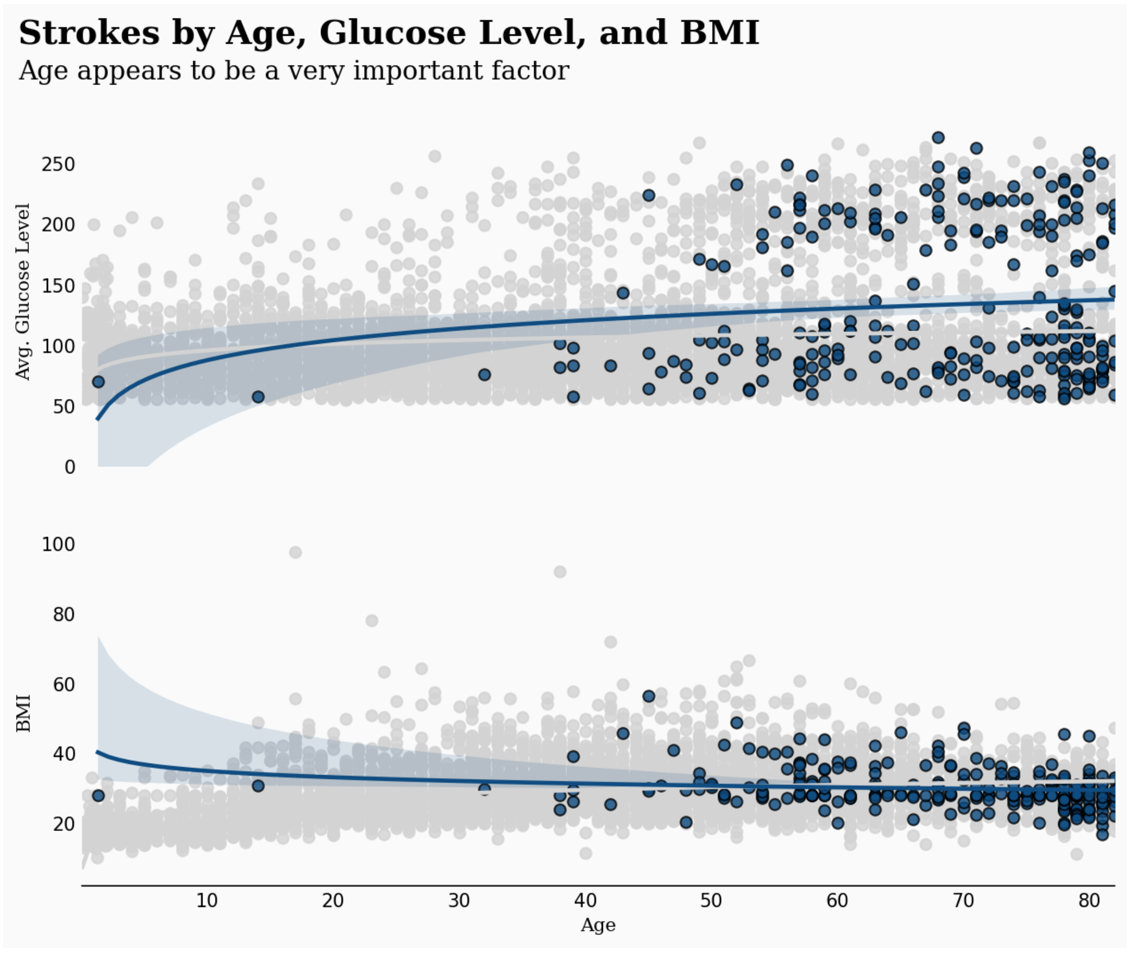
Based on the above plots, it seems clear that Age is a big factor in stroke patients - the older you get the more at risk

[ ]:
```
str_only = df[df['stroke'] == 1]
no_str_only = df[df['stroke'] == 0]
```

[ ]:
```
# Setting up figure and axes

fig = plt.figure(figsize=(10,16),dpi=150,facecolor=background_color)
gs = fig.add_gridspec(4, 2)
gs.update(wspace=0.5, hspace=0.2)
ax0 = fig.add_subplot(gs[0, 0:2])
ax1 = fig.add_subplot(gs[1, 0:2])

ax0.set_facecolor(background_color)
ax1.set_facecolor(background_color)

# glucose

sns.regplot(no_str_only['age'],y=no_str_only['avg_glucose_level'],
            color='lightgray',
            logx=True,
            ax=ax0)

sns.regplot(str_only['age'],y=str_only['avg_glucose_level'],
            color='#0f4c81',
            logx=True,scatter_kws={'edgecolors':['black'],
                                              'linewidth': 1},
            ax=ax0)

ax0.set(ylim=(0, None))
ax0.set_xlabel(" ",fontsize=12,fontfamily='serif')
ax0.set_ylabel("Avg. Glucose Level",fontsize=10,fontfamily='serif')

ax0.tick_params(axis='x', bottom=False)
ax0.get_xaxis().set_visible(False)

for s in ['top','left','bottom']:
    ax0.spines[s].set_visible(False)


# bmi
sns.regplot(no_str_only['age'],y=no_str_only['bmi'],
            color='lightgray',
            logx=True,
            ax=ax1)

sns.regplot(str_only['age'],y=str_only['bmi'],
            color='#0f4c81', scatter_kws={'edgecolors':['black'],
                                              'linewidth': 1},
            logx=True,
            ax=ax1)

ax1.set_xlabel("Age",fontsize=10,fontfamily='serif')
ax1.set_ylabel("BMI",fontsize=10,fontfamily='serif')


for s in ['top','left','right']:
    ax0.spines[s].set_visible(False)
    ax1.spines[s].set_visible(False)


ax0.text(-5,350,'Strokes by Age, Glucose Level, and␣
 ↪BMI',fontsize=18,fontfamily='serif',fontweight='bold')
ax0.text(-5,320,'Age appears to be a very important␣
 ↪factor',fontsize=14,fontfamily='serif')


ax0.tick_params(axis=u'both', which=u'both',length=0)
```

```
ax1.tick_params(axis=u'both', which=u'both',length=0)



plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

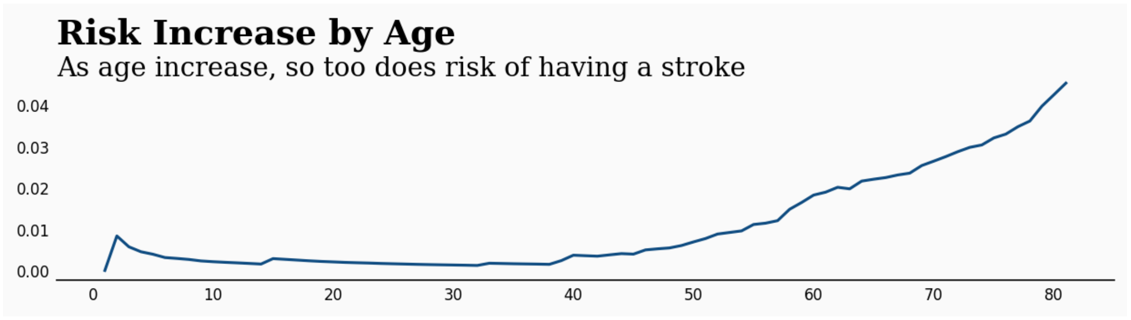


Age is a big factor, and also has slight relationships with BMI & Avg. Glucose levels.

```
[ ]: fig = plt.figure(figsize=(10, 5), dpi=150,facecolor=background_color)
     gs = fig.add_gridspec(2, 1)
     gs.update(wspace=0.11, hspace=0.5)
     ax0 = fig.add_subplot(gs[0, 0])
     ax0.set_facecolor(background_color)


     df['age'] = df['age'].astype(int)

     rate = []
     for i in range(df['age'].min(), df['age'].max()):
         rate.append(df[df['age'] < i]['stroke'].sum() / len(df[df['age'] <␣
      ↪i]['stroke']))

     sns.lineplot(data=rate,color='#0f4c81',ax=ax0)
```

```
for s in ["top","right","left"]:
    ax0.spines[s].set_visible(False)

ax0.tick_params(axis='both', which='major', labelsize=8)
ax0.tick_params(axis=u'both', which=u'both',length=0)

ax0.text(-3,0.055,'Risk Increase by␣
↪Age',fontsize=18,fontfamily='serif',fontweight='bold')
ax0.text(-3,0.047,'As age increase, so too does risk of having a␣
↪stroke',fontsize=14,fontfamily='serif')



plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12: RuntimeWarning:
invalid value encountered in long_scalars
  if sys.path[0] == '':
```



```
[ ]: # Drop single 'Other' gender
     no_str_only = no_str_only[(no_str_only['gender'] != 'Other')]
```

```
[ ]: fig = plt.figure(figsize=(22,15))
     gs = fig.add_gridspec(3, 3)
     gs.update(wspace=0.35, hspace=0.27)
     ax0 = fig.add_subplot(gs[0, 0])
     ax1 = fig.add_subplot(gs[0, 1])
     ax2 = fig.add_subplot(gs[0, 2])
     ax3 = fig.add_subplot(gs[1, 0])
     ax4 = fig.add_subplot(gs[1, 1])
     ax5 = fig.add_subplot(gs[1, 2])
     ax6 = fig.add_subplot(gs[2, 0])
     ax7 = fig.add_subplot(gs[2, 1])
     ax8 = fig.add_subplot(gs[2, 2])

     background_color = "#f6f6f6"
```

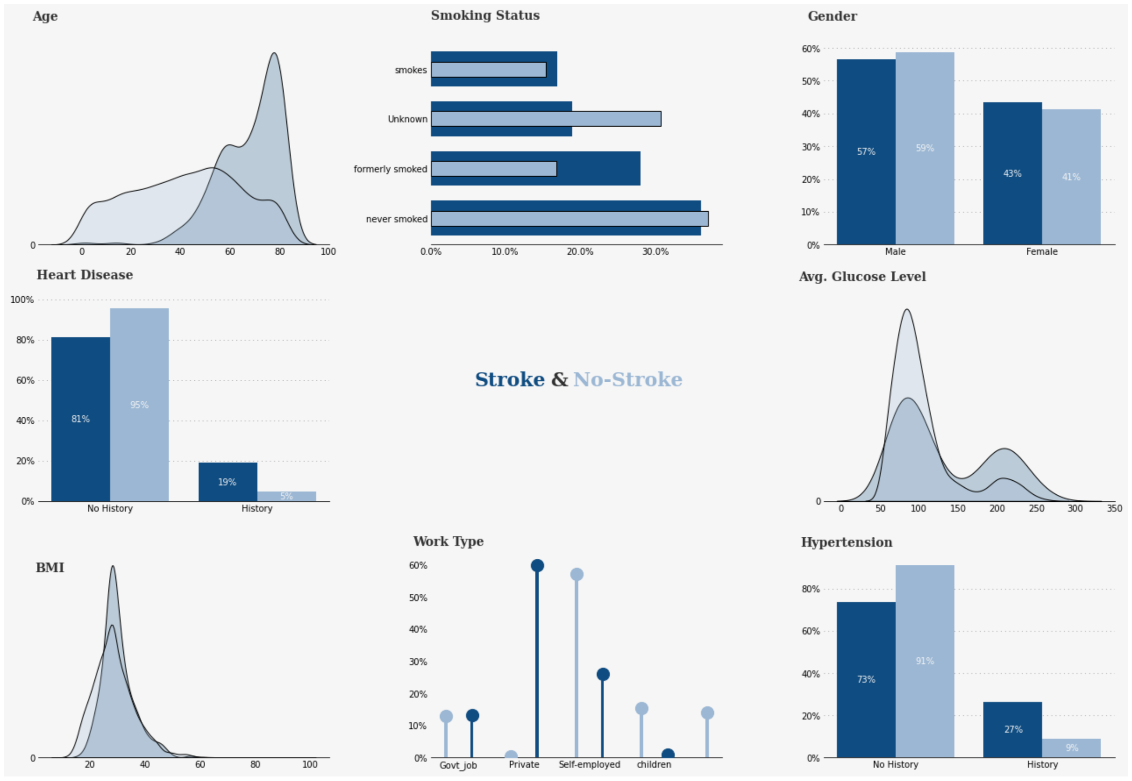
```
fig.patch.set_facecolor(background_color) # figure background color


# Plots

## Age


ax0.grid(color='gray', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
positive = pd.DataFrame(str_only["age"])
negative = pd.DataFrame(no_str_only["age"])
sns.kdeplot(positive["age"], ax=ax0,color="#0f4c81", shade=True,␣
 ↪ec='black',label="positive")
sns.kdeplot(negative["age"], ax=ax0, color="#9bb7d4", shade=True,␣
 ↪ec='black',label="negative")
#ax3.text(0.29, 13, 'Age',
 #         fontsize=14, fontweight='bold', fontfamily='serif', color="#323232")
ax0.yaxis.set_major_locator(mtick.MultipleLocator(2))
ax0.set_ylabel('')
ax0.set_xlabel('')
ax0.text(-20, 0.0465, 'Age', fontsize=14, fontweight='bold',␣
 ↪fontfamily='serif', color="#323232")



# Smoking
positive = pd.DataFrame(str_only["smoking_status"].value_counts())
positive["Percentage"] = positive["smoking_status"].apply(lambda x: x/
 ↪sum(positive["smoking_status"])*100)
negative = pd.DataFrame(no_str_only["smoking_status"].value_counts())
negative["Percentage"] = negative["smoking_status"].apply(lambda x: x/
 ↪sum(negative["smoking_status"])*100)

ax1.text(0, 4, 'Smoking Status', fontsize=14, fontweight='bold',␣
 ↪fontfamily='serif', color="#323232")
ax1.barh(positive.index, positive['Percentage'], color="#0f4c81", zorder=3,␣
 ↪height=0.7)
ax1.barh(negative.index, negative['Percentage'], color="#9bb7d4",␣
 ↪zorder=3,ec='black', height=0.3)
ax1.xaxis.set_major_formatter(mtick.PercentFormatter())
ax1.xaxis.set_major_locator(mtick.MultipleLocator(10))

##
# Ax2 - GENDER
positive = pd.DataFrame(str_only["gender"].value_counts())
```

```
positive["Percentage"] = positive["gender"].apply(lambda x: x/
↪sum(positive["gender"])*100)
negative = pd.DataFrame(no_str_only["gender"].value_counts())
negative["Percentage"] = negative["gender"].apply(lambda x: x/
↪sum(negative["gender"])*100)

x = np.arange(len(positive))
ax2.text(-0.4, 68.5, 'Gender', fontsize=14, fontweight='bold',␣
↪fontfamily='serif', color="#323232")
ax2.grid(color='gray', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
ax2.bar(x, height=positive["Percentage"], zorder=3, color="#0f4c81", width=0.4)
ax2.bar(x+0.4, height=negative["Percentage"], zorder=3, color="#9bb7d4",␣
↪width=0.4)
ax2.set_xticks(x + 0.4 / 2)
ax2.set_xticklabels(['Male','Female'])
ax2.yaxis.set_major_formatter(mtick.PercentFormatter())
ax2.yaxis.set_major_locator(mtick.MultipleLocator(10))
for i,j in zip([0, 1], positive["Percentage"]):
    ax2.annotate(f'{j:0.0f}%',xy=(i, j/2), color='#f6f6f6',␣
↪horizontalalignment='center', verticalalignment='center')
for i,j in zip([0, 1], negative["Percentage"]):
    ax2.annotate(f'{j:0.0f}%',xy=(i+0.4, j/2), color='#f6f6f6',␣
↪horizontalalignment='center', verticalalignment='center')

##
# Ax9 - residence type
positive = pd.DataFrame(str_only["rece"].value_counts())
positive["Percentage"] = positive["gender"].apply(lambda x: x/
↪sum(positive["gender"])*100)
negative = pd.DataFrame(no_str_only["gender"].value_counts())
negative["Percentage"] = negative["gender"].apply(lambda x: x/
↪sum(negative["gender"])*100)

x = np.arange(len(positive))
ax2.text(-0.4, 68.5, 'Gender', fontsize=14, fontweight='bold',␣
↪fontfamily='serif', color="#323232")
ax2.grid(color='gray', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
ax2.bar(x, height=positive["Percentage"], zorder=3, color="#0f4c81", width=0.4)
ax2.bar(x+0.4, height=negative["Percentage"], zorder=3, color="#9bb7d4",␣
↪width=0.4)
ax2.set_xticks(x + 0.4 / 2)
ax2.set_xticklabels(['Male','Female'])
ax2.yaxis.set_major_formatter(mtick.PercentFormatter())
ax2.yaxis.set_major_locator(mtick.MultipleLocator(10))
for i,j in zip([0, 1], positive["Percentage"]):
    ax2.annotate(f'{j:0.0f}%',xy=(i, j/2), color='#f6f6f6',␣
 ↪horizontalalignment='center', verticalalignment='center')
for i,j in zip([0, 1], negative["Percentage"]):
    ax2.annotate(f'{j:0.0f}%',xy=(i+0.4, j/2), color='#f6f6f6',␣
 ↪horizontalalignment='center', verticalalignment='center')


# Heart Dis

positive = pd.DataFrame(str_only["heart_disease"].value_counts())
positive["Percentage"] = positive["heart_disease"].apply(lambda x: x/
 ↪sum(positive["heart_disease"])*100)
negative = pd.DataFrame(no_str_only["heart_disease"].value_counts())
negative["Percentage"] = negative["heart_disease"].apply(lambda x: x/
 ↪sum(negative["heart_disease"])*100)

x = np.arange(len(positive))
ax3.text(-0.3, 110, 'Heart Disease', fontsize=14, fontweight='bold',␣
 ↪fontfamily='serif', color="#323232")
ax3.grid(color='gray', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
ax3.bar(x, height=positive["Percentage"], zorder=3, color="#0f4c81", width=0.4)
ax3.bar(x+0.4, height=negative["Percentage"], zorder=3, color="#9bb7d4",␣
 ↪width=0.4)
ax3.set_xticks(x + 0.4 / 2)
ax3.set_xticklabels(['No History','History'])
ax3.yaxis.set_major_formatter(mtick.PercentFormatter())
ax3.yaxis.set_major_locator(mtick.MultipleLocator(20))
for i,j in zip([0, 1], positive["Percentage"]):
    ax3.annotate(f'{j:0.0f}%',xy=(i, j/2), color='#f6f6f6',␣
 ↪horizontalalignment='center', verticalalignment='center')
for i,j in zip([0, 1], negative["Percentage"]):
    ax3.annotate(f'{j:0.0f}%',xy=(i+0.4, j/2), color='#f6f6f6',␣
 ↪horizontalalignment='center', verticalalignment='center')


## AX4 - TITLE

ax4.spines["bottom"].set_visible(False)
ax4.tick_params(left=False, bottom=False)
ax4.set_xticklabels([])
ax4.set_yticklabels([])
# ax4.text(0.5, 0.6, 'Can we see patterns for\n\n patients in our data?',␣
 ↪horizontalalignment='center', verticalalignment='center',
#          fontsize=22, fontweight='bold', fontfamily='serif', color="#323232")
```

```
ax4.text(0.15,0.57,"Stroke", fontweight="bold", fontfamily='serif',␣
↪fontsize=22, color='#0f4c81')
ax4.text(0.41,0.57,"&", fontweight="bold", fontfamily='serif', fontsize=22,␣
↪color='#323232')
ax4.text(0.49,0.57,"No-Stroke", fontweight="bold", fontfamily='serif',␣
↪fontsize=22, color='#9bb7d4')


# Glucose

ax5.grid(color='gray', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
positive = pd.DataFrame(str_only["avg_glucose_level"])
negative = pd.DataFrame(no_str_only["avg_glucose_level"])
sns.kdeplot(positive["avg_glucose_level"], ax=ax5,color="#0f4c81",ec='black',␣
↪shade=True, label="positive")
sns.kdeplot(negative["avg_glucose_level"], ax=ax5, color="#9bb7d4",␣
↪ec='black',shade=True, label="negative")
ax5.text(-55, 0.01855, 'Avg. Glucose Level',
         fontsize=14, fontweight='bold', fontfamily='serif', color="#323232")
ax5.yaxis.set_major_locator(mtick.MultipleLocator(2))
ax5.set_ylabel('')
ax5.set_xlabel('')




## BMI


ax6.grid(color='gray', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
positive = pd.DataFrame(str_only["bmi"])
negative = pd.DataFrame(no_str_only["bmi"])
sns.kdeplot(positive["bmi"], ax=ax6,color="#0f4c81", ec='black',shade=True,␣
↪label="positive")
sns.kdeplot(negative["bmi"], ax=ax6, color="#9bb7d4",ec='black', shade=True,␣
↪label="negative")
ax6.text(-0.06, 0.09, 'BMI',
         fontsize=14, fontweight='bold', fontfamily='serif', color="#323232")
ax6.yaxis.set_major_locator(mtick.MultipleLocator(2))
ax6.set_ylabel('')
ax6.set_xlabel('')


# Work Type

positive = pd.DataFrame(str_only["work_type"].value_counts())
```

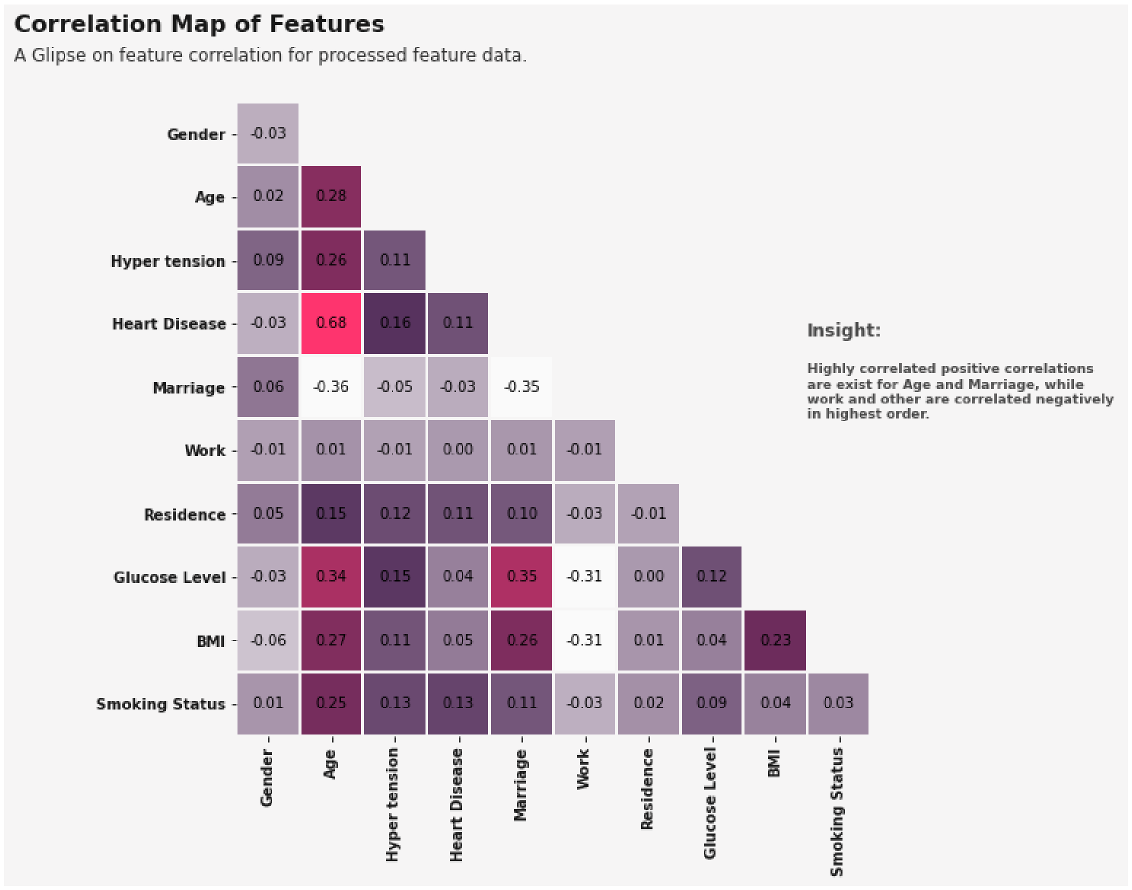
```
positive["Percentage"] = positive["work_type"].apply(lambda x: x/
 ↪sum(positive["work_type"])*100)
positive = positive.sort_index()

negative = pd.DataFrame(no_str_only["work_type"].value_counts())
negative["Percentage"] = negative["work_type"].apply(lambda x: x/
 ↪sum(negative["work_type"])*100)
negative = negative.sort_index()

ax7.bar(negative.index, height=negative["Percentage"], zorder=3,␣
 ↪color="#9bb7d4", width=0.05)
ax7.scatter(negative.index, negative["Percentage"], zorder=3,s=200,␣
 ↪color="#9bb7d4")
ax7.bar(np.arange(len(positive.index))+0.4, height=positive["Percentage"],␣
 ↪zorder=3, color="#0f4c81", width=0.05)
ax7.scatter(np.arange(len(positive.index))+0.4, positive["Percentage"],␣
 ↪zorder=3,s=200, color="#0f4c81")

ax7.yaxis.set_major_formatter(mtick.PercentFormatter())
ax7.yaxis.set_major_locator(mtick.MultipleLocator(10))
ax7.set_xticks(np.arange(len(positive.index))+0.4 / 2)
ax7.set_xticklabels(list(positive.index),rotation=0)
ax7.text(-0.5, 66, 'Work Type', fontsize=14, fontweight='bold',␣
 ↪fontfamily='serif', color="#323232")



# hypertension

positive = pd.DataFrame(str_only["hypertension"].value_counts())
positive["Percentage"] = positive["hypertension"].apply(lambda x: x/
 ↪sum(positive["hypertension"])*100)
negative = pd.DataFrame(no_str_only["hypertension"].value_counts())
negative["Percentage"] = negative["hypertension"].apply(lambda x: x/
 ↪sum(negative["hypertension"])*100)

x = np.arange(len(positive))
ax8.text(-0.45, 100, 'Hypertension', fontsize=14, fontweight='bold',␣
 ↪fontfamily='serif', color="#323232")
ax8.grid(color='gray', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
ax8.bar(x, height=positive["Percentage"], zorder=3, color="#0f4c81", width=0.4)
ax8.bar(x+0.4, height=negative["Percentage"], zorder=3, color="#9bb7d4",␣
 ↪width=0.4)
ax8.set_xticks(x + 0.4 / 2)
ax8.set_xticklabels(['No History','History'])
ax8.yaxis.set_major_formatter(mtick.PercentFormatter())
```

```
ax8.yaxis.set_major_locator(mtick.MultipleLocator(20))
for i,j in zip([0, 1], positive["Percentage"]):
    ax8.annotate(f'{j:0.0f}%',xy=(i, j/2), color='#f6f6f6',␣
 ↪horizontalalignment='center', verticalalignment='center')
for i,j in zip([0, 1], negative["Percentage"]):
    ax8.annotate(f'{j:0.0f}%',xy=(i+0.4, j/2), color='#f6f6f6',␣
 ↪horizontalalignment='center', verticalalignment='center')


# tidy up

for s in ["top","right","left"]:
    for i in range(0,9):
        locals()["ax"+str(i)].spines[s].set_visible(False)

for i in range(0,9):
        locals()["ax"+str(i)].set_facecolor(background_color)
        locals()["ax"+str(i)].tick_params(axis=u'both', which=u'both',length=0)
        locals()["ax"+str(i)].set_facecolor(background_color)


plt.show()
```



```
[ ]: df_copy = df.copy()
     # feature log transformations

     df_copy['age'] = df_copy['age'].apply(lambda x: np.log(x+10)*3)
     df_copy['avg_glucose_level'] = df_copy['avg_glucose_level'].apply(lambda x: np.
      ↪log(x+10)*2)
     df_copy['bmi'] = df_copy['bmi'].apply(lambda x: np.log(x+10)*2)



     # preprocessing - label enconding and numerical value scaling
     ohe = OneHotEncoder()
     ss = StandardScaler()
     le = LabelEncoder()

     ## label encoding of ordinal categorical features
     for col in df_copy.columns:
         df_copy[col] = le.fit_transform(df_copy[col])

     cols = df_copy.columns
     ## normalizing with standard scaler of numerical features
     df_copy[cols] = ss.fit_transform(df_copy[cols])


     # correlation map for all the features
     df_corr = df_copy.corr()
     mask = np.triu(np.ones_like(df_corr, dtype=np.bool))

     fig, ax = plt.subplots(figsize = (8,8))
     fig.patch.set_facecolor('#f6f5f5')
     ax.set_facecolor('#f6f5f5')

     mask = mask[1:, :-1]
     corr = df_corr.iloc[1:,:-1].copy()


     colors = ['#fafafa','#512b58','#fe346e']
     colormap = matplotlib.colors.LinearSegmentedColormap.from_list("", colors)

     # plot heatmap
     sns.heatmap(corr, mask=mask, annot=True, fmt=".2f",cmap = colormap,
                 vmin=-0.15, vmax=0.5, cbar_kws={"shrink": .5, }, ax = ax, cbar =␣
      ↪False,
                 linewidth = 1,linecolor = '#f6f5f5', square = True,annot_kws =␣
      ↪{'size':10, 'color':'black'} )
     # yticks
     ax.tick_params(axis = 'y', rotation=0)
```

```
xticks = ['Gender', 'Age','Hyper tension', 'Heart Disease', 'Marriage', 'Work',␣
 ↪'Residence', 'Glucose Level', 'BMI', 'Smoking Status','Stroke','BMI␣
 ↪Cat','Age Cat']
yticks = ['Gender', 'Age','Hyper tension', 'Heart Disease', 'Marriage', 'Work',␣
 ↪'Residence', 'Glucose Level', 'BMI', 'Smoking Status','Stroke','BMI␣
 ↪Cat','Age Cat']
ax.set_xticklabels(xticks, {'size':10, 'weight':'bold'},rotation = 90, alpha =␣
 ↪0.9)
ax.set_yticklabels(yticks, {'size':10, 'weight':'bold'}, rotation = 0, alpha =␣
 ↪0.9)
ax.text(-3.5,-1.1, 'Correlation Map of Features',{'size': 16, 'weight':'bold'},␣
 ↪alpha = 0.9)
ax.text(-3.5,-0.65, 'A Glipse on feature correlation for processed feature data.
 ↪',{ 'size': 12, 'weight':'normal'}, alpha = 0.8)

ax.text(9,5, 'Highly correlated positive correlations \nare exist for Age and␣
 ↪Marriage, while \nwork and other are correlated negatively \nin highest␣
 ↪order.',{'size': 9, 'weight':'bold'},alpha = 0.7)
ax.text(9,3.7, 'Insight:',{'size': 12, 'weight':'bold'},alpha = 0.7)


fig.show()
```

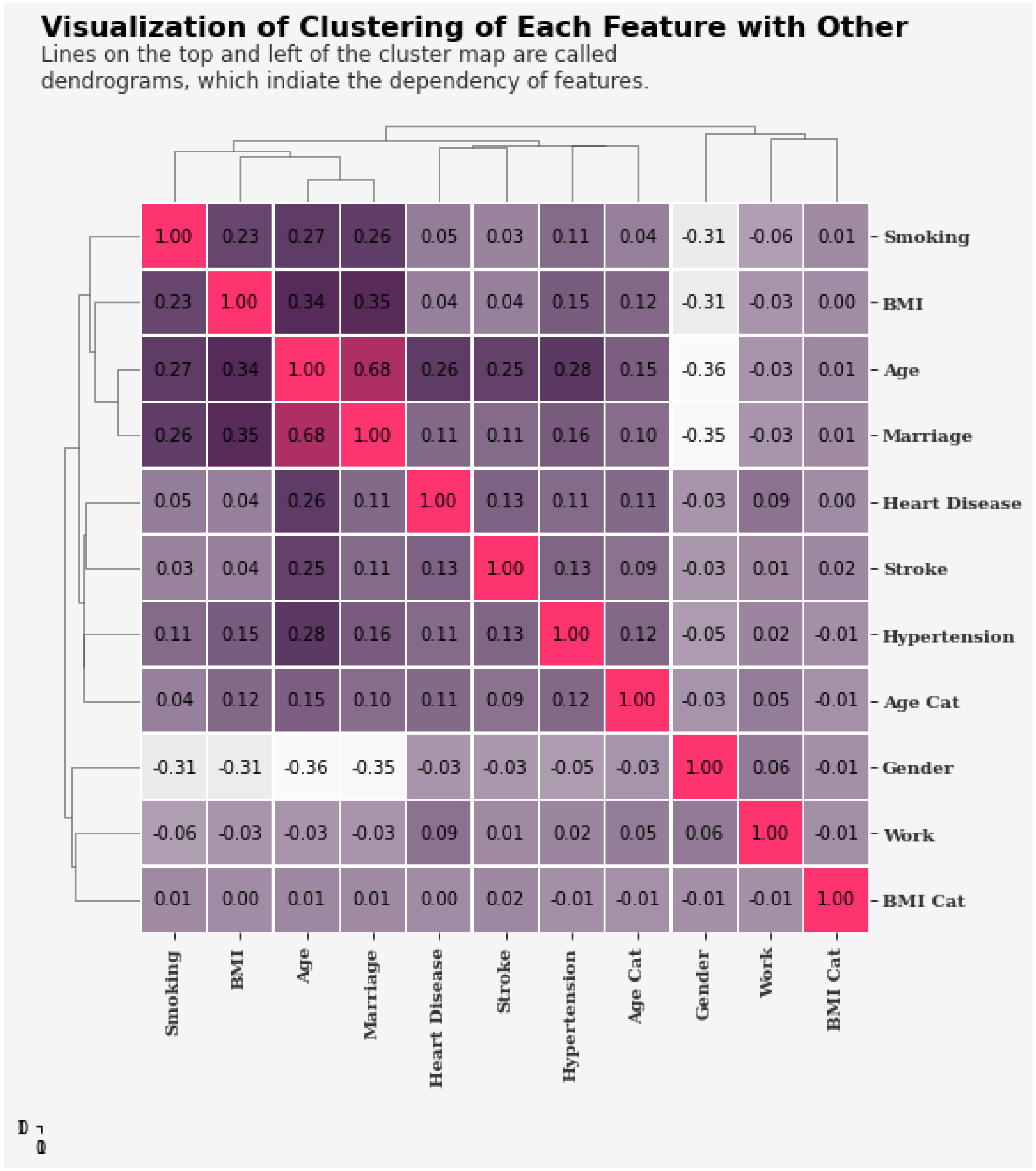


	Gender	Age	Hyper tension	Heart Disease	Marriage	Work	Residence	Glucose Level	BMI	Smoking Status
Gender	-0.03									
Age	0.02	0.28								
Hyper tension	0.09	0.26	0.11							
Heart Disease	-0.03	0.68	0.16	0.11						
Marriage	0.06	-0.36	-0.05	-0.03	-0.35					
Work	-0.01	0.01	-0.01	0.00	0.01	-0.01				
Residence	0.05	0.15	0.12	0.11	0.10	-0.03	-0.01			
Glucose Level	-0.03	0.34	0.15	0.04	0.35	-0.31	0.00	0.12		
BMI	-0.06	0.27	0.11	0.05	0.26	-0.31	0.01	0.04	0.23	
Smoking Status	0.01	0.25	0.13	0.13	0.11	-0.03	0.02	0.09	0.04	0.03

[ ]:
```
labels = ['Smoking', 'BMI','Age', 'Marriage', 'Heart Disease',␣
 ↪'Stroke','Hypertension', 'Age Cat', 'Gender', 'Work', 'BMI Cat',␣
 ↪'Residence','Glucose Level', 'Glucose Cat' ]


g = sns.clustermap(df_corr, annot = True, fmt = '0.2f',
                   cbar= False, cbar_pos=(0,0, 0,0),linewidth = 0.5,
                   cmap = colormap,dendrogram_ratio=0.1,
                   facecolor = '#f6f5f5', figsize = (8,8),square = True,
                   annot_kws = {'size':10, 'color':'black'} )

plt.gcf().set_facecolor('#f6f5f5')
label_args = {'font':18, 'weight':'bold'}
plt.setp(g.ax_heatmap.set_yticklabels(labels), rotation=0, fontsize = 10,␣
 ↪fontfamily = 'Serif', fontweight = 'bold', alpha = 0.8)  # For y axis
plt.setp(g.ax_heatmap.set_xticklabels(labels), rotation=90, fontsize = 10,␣
 ↪fontfamily = 'Serif', fontweight = 'bold', alpha = 0.8) # For x axis
g.fig.text(0,1.065,'Visualization of Clustering of Each Feature with␣
 ↪Other',{'size':16, 'weight':'bold'})
```

```
g.fig.text(0,1.015,'Lines on the top and left of the cluster map are called␣
 ↪\ndendrograms, which indiate the dependency of features.',{'size':12}, alpha␣
 ↪= 0.8)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/matrix.py:1216: UserWarning:
``square=True`` ignored in clustermap
  warnings.warn(msg)
```


Visualization of Clustering of Each Feature with Other
Lines on the top and left of the cluster map are called
dendrograms, which indiate the dependency of features.
Smoking 1.00 0.23 0.27 0.26 0.05 0.03 0.11 0.04 -0.31 -0.06 0.01
BMI 0.23 1.00 0.34 0.35 0.04 0.04 0.15 0.12 -0.31 -0.03 0.00
Age 0.27 0.34 1.00 0.68 0.26 0.25 0.28 0.15 -0.36 -0.03 0.01
Marriage 0.26 0.35 0.68 1.00 0.11 0.11 0.16 0.10 -0.35 -0.03 0.01
Heart Disease 0.05 0.04 0.26 0.11 1.00 0.13 0.11 0.11 -0.03 0.09 0.00
Stroke 0.03 0.04 0.25 0.11 0.13 1.00 0.13 0.09 -0.03 0.01 0.02
Hypertension 0.11 0.15 0.28 0.16 0.11 0.13 1.00 0.12 -0.05 0.02 -0.01
Age Cat 0.04 0.12 0.15 0.10 0.11 0.09 0.12 1.00 -0.03 0.05 -0.01
Gender -0.31 -0.31 -0.36 -0.35 -0.03 -0.03 -0.05 -0.03 1.00 0.06 -0.01
Work -0.06 -0.03 -0.03 -0.03 0.09 0.01 0.02 0.05 0.06 1.00 -0.01
BMI Cat 0.01 0.00 0.01 0.01 0.00 0.02 -0.01 -0.01 -0.01 -0.01 1.00
Smoking
BMI
Age
Marriage
Heart Disease
Stroke
Hypertension
Age Cat
Gender
Work
BMI Cat