# BRAIN STROKE DETECTION

Using Support Vector Machines

## 1. Cleaning data

In [244]:

```
import pandas as pd
from google.colab import drive

drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, c
all drive.mount("/content/drive", force_remount=True).
```

In [245]:

```
raw_data=pd.read_csv('/content/drive/MyDrive/datasets/stroke_data.csv')

raw_data.head()
```

Out[245]:

| | id | gender | age | hypertension | heart_disease | ever_married | work_type | Residence_ty |
|---|---|---|---|---|---|---|---|---|
| **0** | 9046 | Male | 67.0 | 0 | 1 | Yes | Private | Urb |
| **1** | 51676 | Female | 61.0 | 0 | 0 | Yes | Self-employed | Ru |
| **2** | 31112 | Male | 80.0 | 0 | 1 | Yes | Private | Ru |
| **3** | 60182 | Female | 49.0 | 0 | 0 | Yes | Private | Urb |
| **4** | 1665 | Female | 79.0 | 1 | 0 | Yes | Self-employed | Ru |

In [246]:

```
raw_data.describe()
```

Out[246]:

| | id | age | hypertension | heart_disease | avg_glucose_level | b |
|---|---|---|---|---|---|---|
| **count** | 5110.000000 | 5110.000000 | 5110.000000 | 5110.000000 | 5110.000000 | 4909.0000 |
| **mean** | 36517.829354 | 43.226614 | 0.097456 | 0.054012 | 106.147677 | 28.8932 |
| **std** | 21161.721625 | 22.612647 | 0.296607 | 0.226063 | 45.283560 | 7.8540 |
| **min** | 67.000000 | 0.080000 | 0.000000 | 0.000000 | 55.120000 | 10.3000 |
| **25%** | 17741.250000 | 25.000000 | 0.000000 | 0.000000 | 77.245000 | 23.5000 |
| **50%** | 36932.000000 | 45.000000 | 0.000000 | 0.000000 | 91.885000 | 28.1000 |
| **75%** | 54682.000000 | 61.000000 | 0.000000 | 0.000000 | 114.090000 | 33.1000 |
| **max** | 72940.000000 | 82.000000 | 1.000000 | 1.000000 | 271.740000 | 97.6000 |

In [247]:

```
raw_data.isnull().sum()
```

Out[247]:

```
id                     0
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
```

In [248]:

```
import numpy as np

from sklearn.impute import SimpleImputer

imputer=SimpleImputer()

raw_data['bmi']=imputer.fit_transform(np.array(raw_data['bmi']).reshape(-1, 1))
```

In [249]:

```
raw_data.isnull().sum()
```

Out[249]:

```
id                   0
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
```

In [250]:

```
clean_data=raw_data.drop('id',axis=1)
```

In [251]:

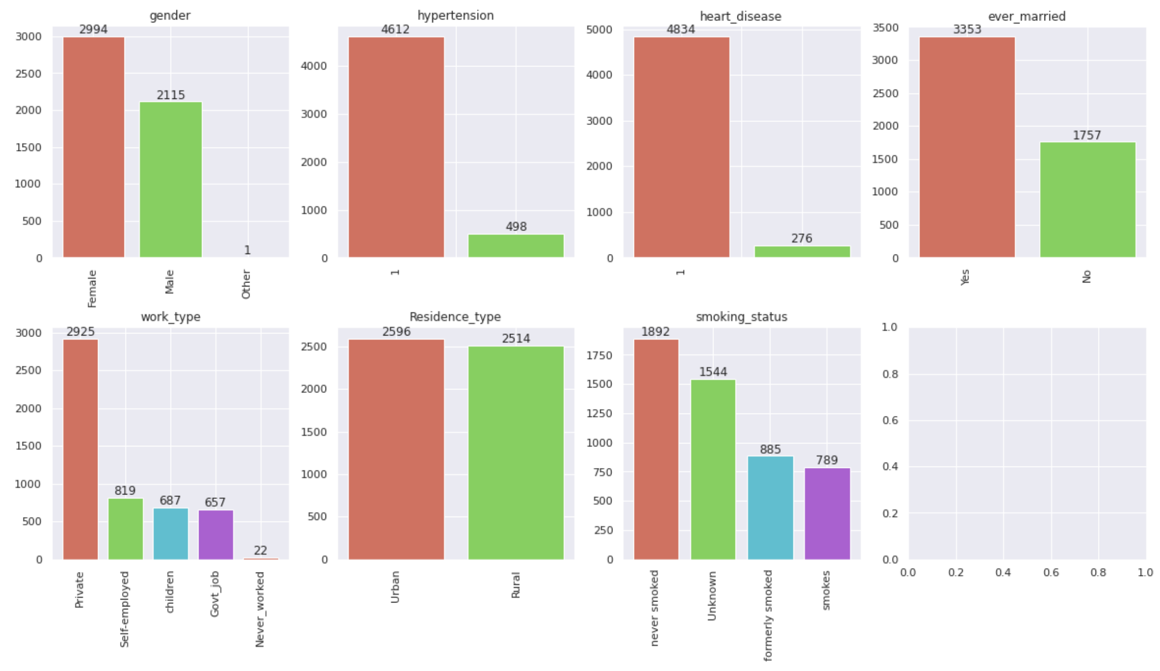
```
clean_data.head()
```

Out[251]:

| | gender | age | hypertension | heart_disease | ever_married | work_type | Residence_type | avg |
|---|---|---|---|---|---|---|---|---|
| **0** | Male | 67.0 | 0 | 1 | Yes | Private | Urban | |
| **1** | Female | 61.0 | 0 | 0 | Yes | Self-employed | Rural | |
| **2** | Male | 80.0 | 0 | 1 | Yes | Private | Rural | |
| **3** | Female | 49.0 | 0 | 0 | Yes | Private | Urban | |
| **4** | Female | 79.0 | 1 | 0 | Yes | Self-employed | Rural | |

In [252]:

```
import matplotlib.pyplot as plt

features=clean_data.columns.tolist()
features.remove('stroke')

con_features=['age','bmi','avg_glucose_level']
cat_features=[feature for feature in features if feature not in con_features]

fig, axes = plt.subplots(2, 4)
plt.subplots_adjust(hspace=0.3)
fig.set_figwidth(20)
fig.set_figheight(10)
feature_index=0
colors=['#CF7261','#87CF61','#61BECF','#A961CF']
for i,ai in enumerate(axes):
    for j,aij in enumerate(axes[i]):
        if(not (i == 1 and j == 3)):
            value_counts=clean_data[cat_features[feature_index]].value_counts()
            value_counts_index=value_counts.index.tolist()
            value_counts_values=value_counts.values.tolist()
            bars=axes[i][j].bar(value_counts_index,value_counts_values,color=colors)
            axes[i][j].set_title(cat_features[feature_index])
            axes[i][j].set_xticklabels(value_counts_index,rotation=90)
            feature_index+=1

            for k,bar in enumerate(bars):
                axes[i][j].text(k,value_counts_values[k], value_counts_values[k],ha='ce
nter',va='bottom')
```



In work type, both children and never worked convey same meaning with regard to employment status. So both categories are merged into one as never worked

In [253]:

```
clean_data['work_type']=clean_data['work_type'].replace('children','Never_worked')
```

## 2. Encoding categorical variables

In [254]:

```
from sklearn.preprocessing import LabelEncoder,OneHotEncoder

label=['ever_married','Residence_type']
one_hot=['gender','work_type','smoking_status']

for feature in cat_features:
    if(feature in label):
        encoder=LabelEncoder()
        clean_data[feature]=encoder.fit_transform(np.array(clean_data[feature]).reshape
(-1,1))
    elif(feature in one_hot):
        dummies=pd.get_dummies(clean_data[feature],prefix=feature)
        clean_data=pd.concat([clean_data,dummies],axis=1)
        clean_data=clean_data.drop([feature],axis=1)

new_features=clean_data.columns.tolist()
new_features.remove('stroke')

cat_features=[feature for feature in new_features if feature not in con_features]

clean_data.head()
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/preprocessing/_label.py:11
5: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

Out[254]:

| | age | hypertension | heart_disease | ever_married | Residence_type | avg_glucose_level | |
|---|---|---|---|---|---|---|---|
| **0** | 67.0 | 0 | 1 | 1 | 1 | 228.69 | 36.6 |
| **1** | 61.0 | 0 | 0 | 1 | 0 | 202.21 | 28.8 |
| **2** | 80.0 | 0 | 1 | 1 | 0 | 105.92 | 32.5 |
| **3** | 49.0 | 0 | 0 | 1 | 1 | 171.23 | 34.4 |
| **4** | 79.0 | 1 | 0 | 1 | 0 | 174.12 | 24.0 |

## 3. Splitting testing data

In [255]:

```
from sklearn.model_selection import train_test_split as tts

x=clean_data.drop('stroke',axis=1)
y=clean_data['stroke']

x_train,x_test,y_train,y_test=tts(x,y,test_size=0.2,random_state=1)

print(x_train.shape,x_test.shape,y_train.shape,y_test.shape)
print(y_train.value_counts())
print(y_test.value_counts())
```

```
(4088, 18) (1022, 18) (4088,) (1022,)
0    3899
1     189
Name: stroke, dtype: int64
0    962
1     60
Name: stroke, dtype: int64
```

## 4. Feature selection

In [256]:

```
features=clean_data.drop('stroke',axis=1)
target=clean_data['stroke']
```

### ANOVA test

In [257]:

```
from sklearn.feature_selection import f_classif

anova_result=pd.DataFrame(columns=['feature','f_score','p_value'])

for feature in con_features:
    result = f_classif(np.array(x_train[feature]).reshape(-1,1),y_train)
    anova_result=anova_result.append({'feature':feature,'f_score':round(result[0][0],5
),'p_value':round(result[1][0],5)},ignore_index=True)

anova_result.sort_values(by=['f_score'],ascending=False,ignore_index=True)
```

Out[257]:

| | feature | f_score | p_value |
|---|---|---|---|
| **0** | age | 240.88241 | 0.0000 |
| **1** | avg_glucose_level | 57.13297 | 0.0000 |
| **2** | bmi | 6.13381 | 0.0133 |

### Chi - Squared test

In [258]:

```
from sklearn.feature_selection import chi2

chi2_result=pd.DataFrame(columns=['feature','chi2_score','p_value'])

for feature in cat_features:
    result = chi2(np.array(x_train[feature]).reshape(-1,1),y_train)
    chi2_result=chi2_result.append({'feature':feature,'chi2_score':round(result[0][0],5
),'p_value':round(result[1][0],5)},ignore_index=True)

chi2_result.sort_values(by=['chi2_score'],ascending=False,ignore_index=True)
```

Out[258]:

| | feature | chi2_score | p_value |
|---|---|---|---|
| **0** | heart_disease | 67.66219 | 0.00000 |
| **1** | hypertension | 57.51814 | 0.00000 |
| **2** | work_type_Never_worked | 24.17382 | 0.00000 |
| **3** | ever_married | 12.10966 | 0.00050 |
| **4** | work_type_Self-employed | 10.57817 | 0.00114 |
| **5** | smoking_status_formerly smoked | 10.41427 | 0.00125 |
| **6** | smoking_status_Unknown | 7.13372 | 0.00756 |
| **7** | Residence_type | 0.68596 | 0.40754 |
| **8** | work_type_Private | 0.54293 | 0.46122 |
| **9** | gender_Male | 0.21359 | 0.64397 |
| **10** | gender_Female | 0.15344 | 0.69527 |
| **11** | smoking_status_never smoked | 0.04851 | 0.82568 |
| **12** | work_type_Govt_job | 0.00033 | 0.98546 |
| **13** | smoking_status_smokes | 0.00004 | 0.99481 |
| **14** | gender_Other | NaN | NaN |

In [259]:

```
x_train=x_train.drop(['work_type_Govt_job','Residence_type','gender_Male','gender_Othe
r','gender_Female','smoking_status_never smoked'],axis=1)
```

## 5. Handling imbalanced data

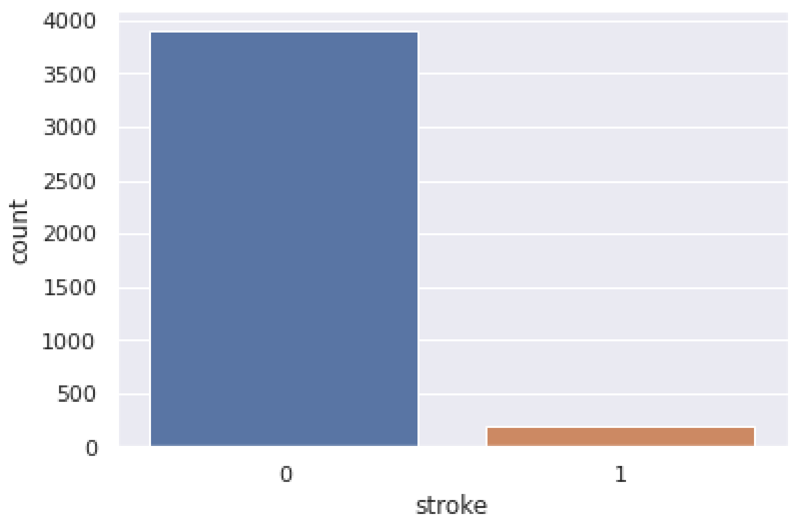
In [260]:

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.set()

print(y_train.value_counts())
sns.countplot(y_train)
```

```
0    3899
1     189
Name: stroke, dtype: int64

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

Out[260]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f63298eed50>
```



In [261]:

```
from imblearn.over_sampling  import RandomOverSampler

sampler = RandomOverSampler(random_state=1)

x_resampled, y_resampled = sampler.fit_resample(x_train, y_train)
```

In [262]:

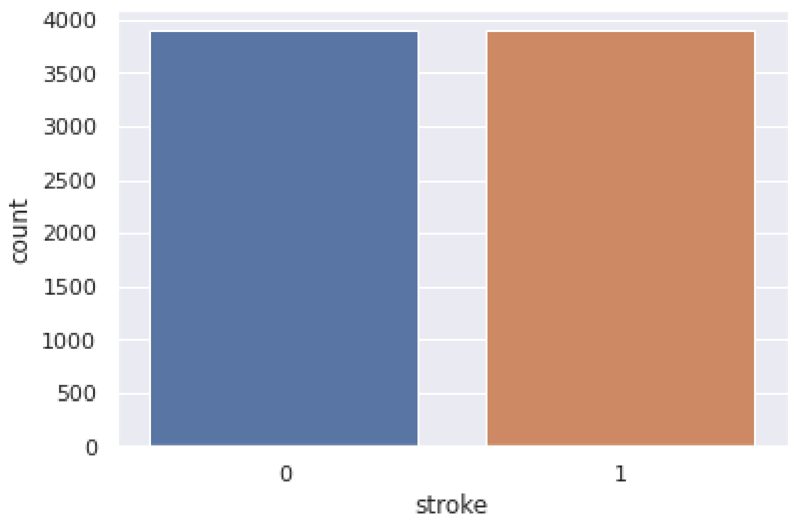
```
print(y_resampled.value_counts())

sns.countplot(y_resampled)
```

```
0    3899
1    3899
Name: stroke, dtype: int64
```

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

Out[262]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6324e28a50>
```



## 6. Modeling

In [270]:

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

model=SVC(C=100,gamma='auto')
model.fit(x_resampled,y_resampled)

y_train_pred=model.predict(x_resampled)

print('Training accuracy {}%'.format(round(accuracy_score(y_resampled,y_train_pred)*100
,2)))
```

```
Training accuracy 99.99%
```

Removing weak features from test set

In [264]:

```
x_test=x_test.drop(['work_type_Govt_job','Residence_type','gender_Male','gender_Other',
'gender_Female','smoking_status_never smoked'],axis=1)
```

Make predictions for test set

In [265]:

```
y_test_pred=model.predict(x_test)

pd.Series(y_test_pred).value_counts()
```

Out[265]:

```
0    992
1     30
dtype: int64
```

In [271]:

```
print('Test accuracy {}%'.format(round(accuracy_score(y_test,y_test_pred)*100,2)))
```

```
Test accuracy 92.37%
```

In [267]:

```
# !jupyter nbconvert --to html "/content/drive/MyDrive/Colab Notebooks/svm.ipynb"
```