International Engineering Journal for Research & Development

Vol. 7
Issue 3

PERMEABILITY FORECAST OF BLOOD–BRAIN BARRIER (BBB) IN THE CENTRAL NERVOUS SYSTEM USING ML

[1]Venkata Ravi Kiran Kolla, [2]Niharikareddy Meenigea
Sr. Software Engineer[1], Researcher and Data Analyst[2]
venkat0634@gmail.com[1], readdyniharika2@gmail.com[2]

ABSTRACT

98 percent of the substances that enter the central nervous system are controlled by the blood-brain barrier (BBB) (CNS). Compounds with high permeability must be discovered to enable the production of brain medications for the treatment of various brain conditions including Parkinson's, Alzheimer's, and brain malignancies. To address this issue, a number of models have been developed over time, with respectable accuracy ratings in forecasting substances that penetrate the blood-brain barrier. Forecasting molecules with "poor" permeability, however, has proven challenging. Several machine learning classifiers, including Principal Component Analysis PCA, Neural Network SVC, and XGBoost, have been compared using Molecule Net in this research study and are shown in the outcome section. Several concerns should be addressed before creating the classification model in order to enhance the high-dimensional, unbalanced dataset: the Oversampling methods are used to handle the unbalanced dataset, while kernel principal component analysis, a non-linear dimensionality reduction method, is used to solve the excessive dimensionality. The accuracy of a 500-epoch neural network is about 98%, which is far higher than that of earlier studies.

Keywords: blood brain barrier, CNS , SVC, XGBoost, classifiers, Neural network.

INTRODUCTION

Blood–brain barrier (BBB) is dangerously emerging as one of the fatal neurodegenerative disorders over the lastfew years which depicts itself as a progressive degeneration of cognitive abilities. 50 million people, most of whom are above 65 years of age, are affected with BBB worldwide [1]. It has been also reported that around 10% of the BBB patientsare within the age group from 30 years to 60 years. BBB is regarded as the most common form of dementia and presently 60-70% of dementia is due to BBB [2].

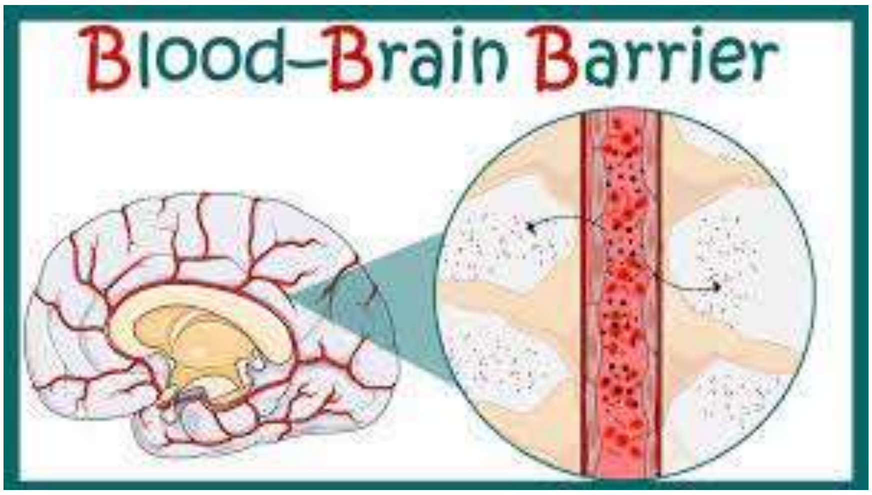


Figure 1 Blood–brain barrier (BBB)

In 2010, 3.7 million Indians were suffering from dementia where this number has grown up to 6.1 million at present. Burden of geriatric diseases is naturally heavier to acountry like India where, according to census

2011, around 8.6% of total population are above the age of 60 years [3]. Moreover, lack of general awareness and dearth of specialists in geriatric diseases make the situation even morealarming in India.

BBB is caused primarily due to the loss of brain tissues like white matter (WM), gray matter (GM) etc and this BBB enhanced volume of cerebra spinal fluid (CSF). Symptoms of BBB range from forgetfulness in its early stage to loss of speech and cognitive ability in later stage. Sometimes the symptoms of BBB are mistakenly ignoredeven by the friends and relatives of the patient. Generally, the transition of a healthy patient to BBB occurs through an intermediate stage called mild cognitive impairment (MCI) [4-5]. This may BBB to a fatal disease in many a case.As compared to the other fields of study in medical science, research in dementia remains relatively low. Early detection of BBB has therefore surfaced as a need of the hour as presently no clinical diagnosis is standardized. In this regard, professionals and researchers from medical and non-medical backgrounds have put their hands together to deliver a timely solution.

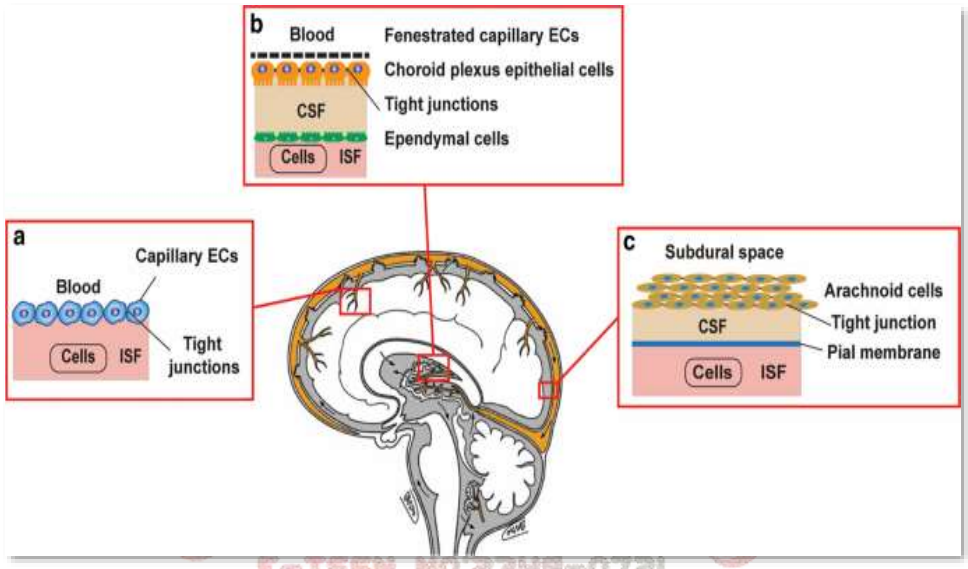


Figure 2 BBB structure

This paper is organized as follows: state-of-the-art research works in the field of BBB detection have been briefly listed in section II. Proposed algorithm towards the identification of BBB patients with the aid of machine learning technique is demonstrated in section III followed by results and discussions in section IV. This paper is concluded in section V along with supplementary data inAnnexure.

LITERATURE REVIEW

Researchers throughout the globe have been generously contributing in the field of medical image analysis and computer vision over the last few. Early detectionof several neurological disorders like BBB has BBB attained serious attention of medical practitioners and computer scientists.

A number of research articles hBBB been published over the years which have opened up the door of various opportunities [6-8]. In the recent years, machinelearning based algorithms have received justified interest among the researchers across different disciplines and the field of medical image processing is of no exception. This section briefly summarizes few of these developments which focus on the detection of BBB.

As an attempt to investigate the automatic detection of GM loss for BBB patients, voxel based morphometry

(VBM) and support vector machine (SVM) were combined together with the BBB advantages of objectivity, reproducibility and automation. Authors in have used three sections viz. frontal to extract the hippocampus, sagittal to analyze the corpus callosum and axial to identify the features of cortex for their classification using SVM. Another SVM based majority voter classifier has been proposed in which makes use of percentage volume of WM, GM and CSFduring the process of classification.

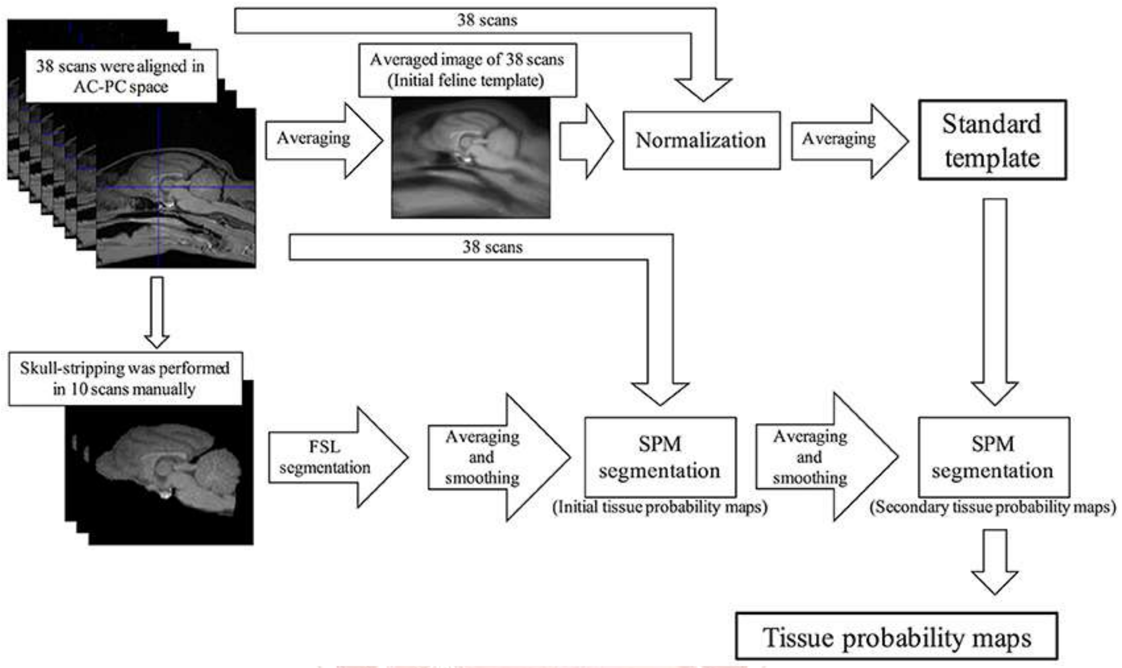

38 scans were aligned in AC-PC space
38 scans
Averaged image of 38 scans (Initial feline template)
Averaging
Normalization
Averaging
Standard template
38 scans
Skull-stripping was performed in 10 scans manually
FSL segmentation
Averaging and smoothing
SPM segmentation
(Initial tissue probability maps)
Averaging and smoothing
SPM segmentation
(Secondary tissue probability maps)
Tissue probability maps

Figure 3 voxel based morphometry

Deep learning based approaches are also increasinglyemployed in this field of computer aided diagnosis. One such architecture which consists of stacked auto-encoders and SoftMax output layer has been employed in and it requires less labelled training samples and minimal domain prior knowledge. A novel method for a high level latent and shared feature representation from neuroimaging modalities via deep learning BBB been proposed in which a deepnetwork with restricted Boltzmann machine is used to find out a latent hierarchical feature representation from a 3D patch. Gray matter images from each brain area have been split into 3D patches which are used to train different deep belief networks . In connection to this, an ensemble of deep belief networks is composed in which the final prediction is obtained through a voting scheme.

Study in presents a convolutional neural network (CNN) model for four binary classification tasks viz. BBB vs.HC, progressive MCI vs. HC, stable MCI vs. HC and pMCI vs. sMCI using MRI images. A convolutional autoencoder based unsupervised learning is employed for the first classification task while supervised transfer learning is applied to solve the final classification task. Lee et al. BBB proposed a novel framework for structural MRI classification of BBB with data combination, outlier removal and entropy-based data selection using AlexNet .Authors have claimed to obtain binary classification accuracies of 95.35% and 98.74% on OASIS and BBB datasets respectively. Strength of 3D-CNN and fully stacked bidirectional long short-term memory (FSBi-LSTM) has been exploited in another framework . This architecture was designed to derive deep feature representation fromboth MRI and PET.

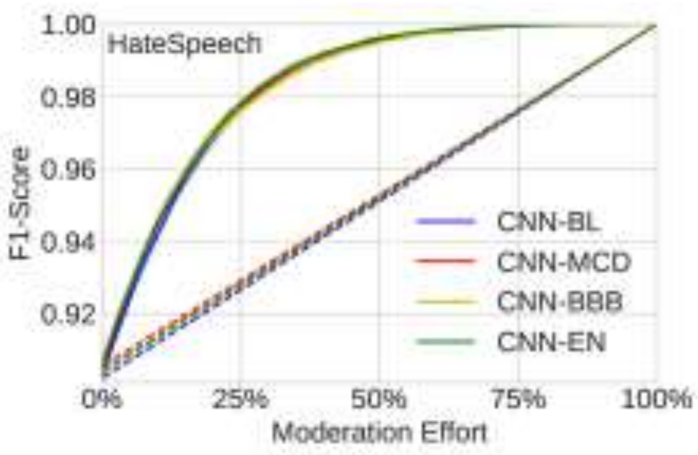


Figure 4 convolutional neural network

PROPOSED ALGORITHM

This section clearly explains the proposed algorithm in classifying BBB and HC from the percentage volumetric information of WM, GM and CSF. It has been reiterated in the literature that significant tissue loss in WM and GM and subsequent increase in CSF is the consequence of BBB. Hence, these parameters play a pivotal role in identifying the occurrence of BBB in brain MRI scans. However, some more research is essential in identifying the priority amongst these entities.

Proposed algorithm makes use of decision tree in prioritizing the role of WM, GM and CSF in regard to the classification of brain MRI scans into two different sets. It isdeveloped on a set of 98 images obtained from BBBNI datasetout of which 48 images are of BBB and the rest is HC.Percentage volume of WM, GM and CSF been obtainedusing Vol Brain platform and has been employed in the current proposition.

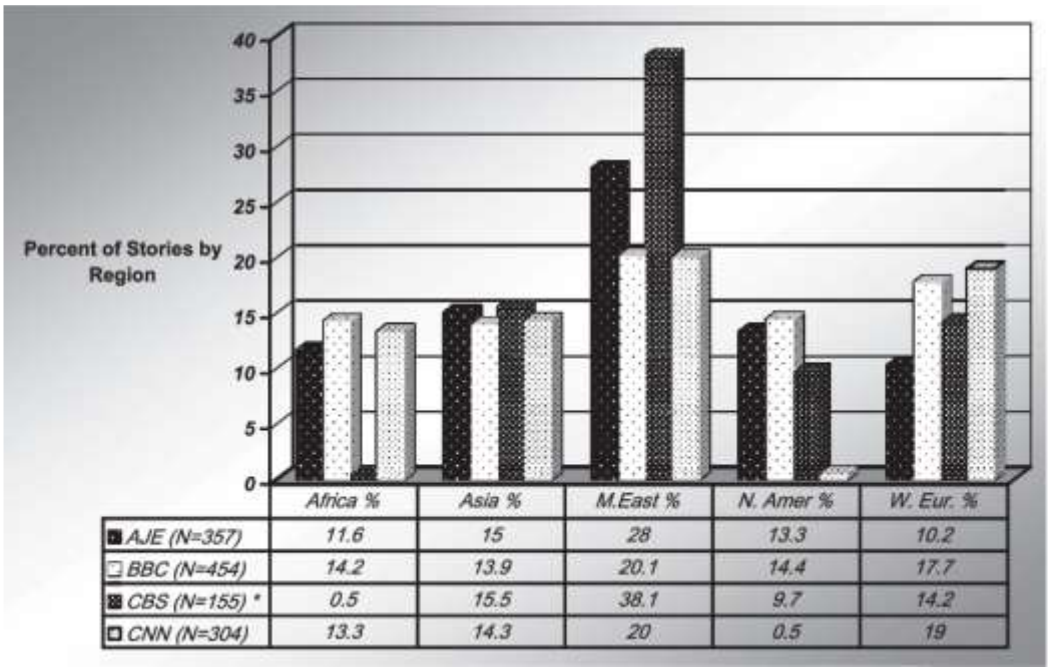
Proposed decision tree is constructed by taking all the training data points (both BBB and HC) into consideration. Initially, data pertaining to each of these three sets are arranged in an ascending order and the average between twoconsecutive data points is calculated. In fact, this average value between two BBB data points emerges as a potential threshold in classification.

RESULTS & DISCUSSIONS

Proposed decision tree based classification technique forsegregating BBB subjects from healthy ones has been a total of 120 images from BBB source BBB been taken intoour consideration in which each of the two classes BBB 60 images each. Percentage volume of WM, GM and CSF resulting from these MRI images have been obtained using Vol Brain and the same is listed in ANNEXURE for quick reference . On the set of 98 training images, resultant impurity is found out to be minimum while classified with respect to CSF followed by WM and GM. These values are computed as 0, 0.0391 and 0.386619 respectively. Hence, during the construction of decision tree on test data; root node is selected as percentage CSF volume with a threshold of 18%. It has been found that 63 out of 120 entriescorrespond to a percentage CSF greater than the threshold while rest 57 is below the threshold. However, this classification contains some impurity and therefore itrequires further exploration of the tree.

The immediate next node (left) is explored with percentage WM volume as the deciding parameter. It has been come to our observation thatout of 63 entries associated with this node, 58 test data are having percentage WM volume of less than the threshold of 35.3%. Moreover, each of them is from the BBB dataset and hence symbolizes a leaf node (L_1). Rest 5 comprises of a mixture of BBB and HC data points. 57 data points, whose percentage CSF is below the threshold, are similarly explored using percentage WM volume as the decisive parameter. This ultimately BBBs to the generation of two leafnodes (L_2 and L_3) with zero valued impurity. The complete decision tree is depicted in Fig. 2 in the appendix.

Performance of this decision tree based classificationprocess is subsequently measured using relevant metrics likeaccuracy, sensitivity, specificity etc. It can be BBB identified from Fig. 2 that 58 data points (refer L_1) are correctly identified as BBB (true positive) while 53 data points(refer L_2) are properly classified as HC (true negative). All the data points at N_3 are categorized as healthy, although 2 ofthem actually correspond to BBB (false negative). In BBB to this, 4 data points (refer L_3) are misclassified as BBB (falsepositive). Each of the aforementioned performance metrics are evaluated



	Africa %	Asia %	M.East %	N. Amer %	W. Eur. %
▪ AJE (N=357)	11.6	15	28	13.3	10.2
□ BBC (N=454)	14.2	13.9	20.1	14.4	17.7
▩ CBS (N=155) *	0.5	15.5	38.1	9.7	14.2
▫ CNN (N=304)	13.3	14.3	20	0.5	19

Figure 5 Comparison of Result

Finally, outcome of the proposed approach is compared with some of the state-of-the-art machine learning basedtechniques for BBB detection. Comparative observation islisted in Table I. Looking at Table I, it can be interpreted thatthe proposed technique yields comparable or better performance as far as these performance metrics are concerned. However, it should be remembered that these results are indicative but not conclusive.

CONCLUSION

This article introduces a decision tree based classification algorithm for early detection of BBB. In connection to this, percentage volume of WM, GM and CSF BBB been taken into our consideration. It has been seen that CSF basedclassification results in minimum impurity followed by WMand GM. Entire decision tree has been constructed accordingly. Proposed algorithm has been tested on a number of BBB NI data set and the resultant performanceparameters have been calculated. Future research may be directed by incorporating data points corresponding to pMCIand sMCI for a more robust classification.

REFERENCES

[1] S. Gauthier, P. Rosa-Neto, JA Morais and C. Webster, “World Alzheimer Report 2021: Journey through

the diagnosis of dementia", Alzheimer‟s Disease International, London, England, 2021.

[2] C. Ballard, S. Gauthier, A. Corbett, C. Brayne, D. Aarsland, and D. Jones, "Alzheimer‟s disease," The Lancet, vol. 377, no. 9770, pp. 1019–1031, 2011.

[3] KS. Shaji, AT. Jotheeswaran, N. Girish, S. Bharath, A. Dias, M. Pattabiraman and M. Varghese, "The Dementia India Report 2010: prevalence, impact, costs and services for dementia", Alzheimer‟s and Related Disorders Society of India (ARDSI), New Delhi, India, 2010, ISBN:978-81-920341-0-2.

[4] S. Gauthier et al., "Mild Cognitive Impairment", Lancet, vol. 367, no. 9518, pp. 1262-1270, April 2006.

[5] C. Ballard, S. Gauthier, A. Corbett, C. Brayne, D. Aarsland, and D. Jones, "Alzheimer‟s disease," The Lancet, vol. 377, no. 9770, pp. 1019–1031, 2011.

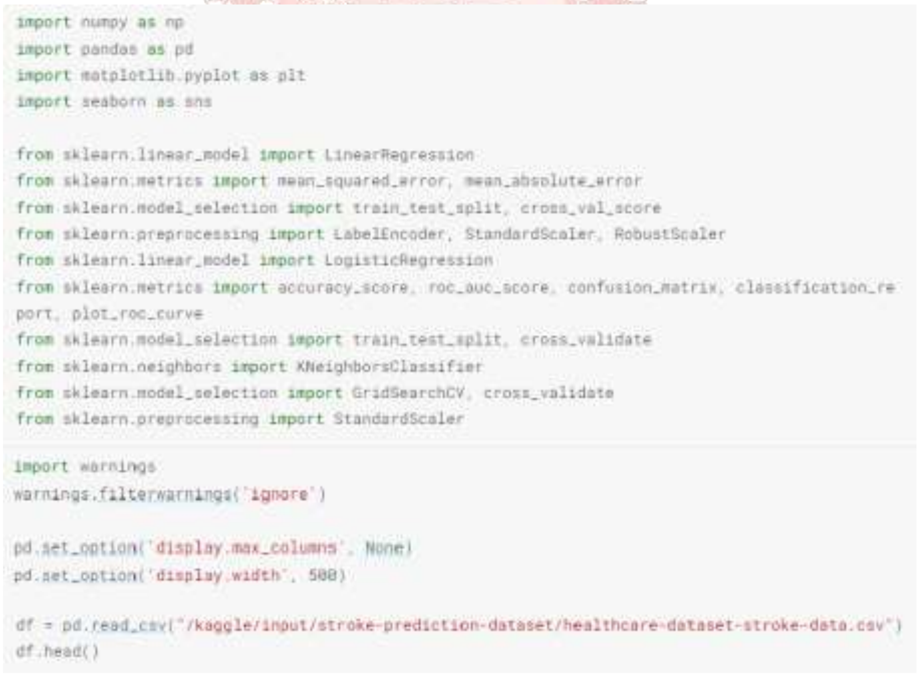
[6] S. Kloppel et al. "Automatic classification of MR scans in Alzheimer‟s disease", Brain: a journal of neurology, 131, pp. 681– 689, 2008.

[7] S. Valverdeet al., "Automated tissue segmentation of MR brain images in the presence of white matter lesions", Medical Image Analysis, vol. 35, pp. 446-457, 2017.

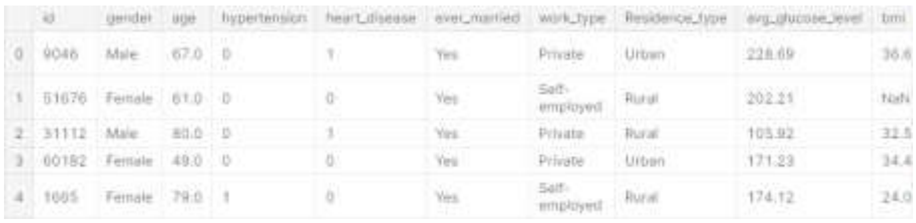
[8] S. Ghosh, AP. Hazarika, A. Chandra and RK. Mudi, "Adaptive neighbor constrained deviation sparse variant fuzzy c-means clustering for brain MRI of AD subject", Visual Informatics (Elsevier), vol. 5, no. 4, pp. 67-80, December 2021

Appendix

Code

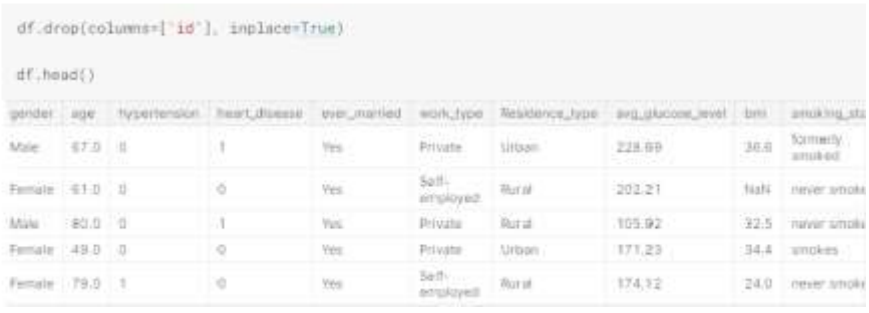


```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import LabelEncoder, StandardScaler, RobustScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, roc_auc_score, confusion_matrix, classification_re
port, plot_roc_curve
from sklearn.model_selection import train_test_split, cross_validate
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV, cross_validate
from sklearn.preprocessing import StandardScaler

import warnings
warnings.filterwarnings('ignore')

pd.set_option('display.max_columns', None)
pd.set_option('display.width', 500)

df = pd.read_csv("/kaggle/input/stroke-prediction-dataset/healthcare-dataset-stroke-data.csv")
df.head()
```



	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0

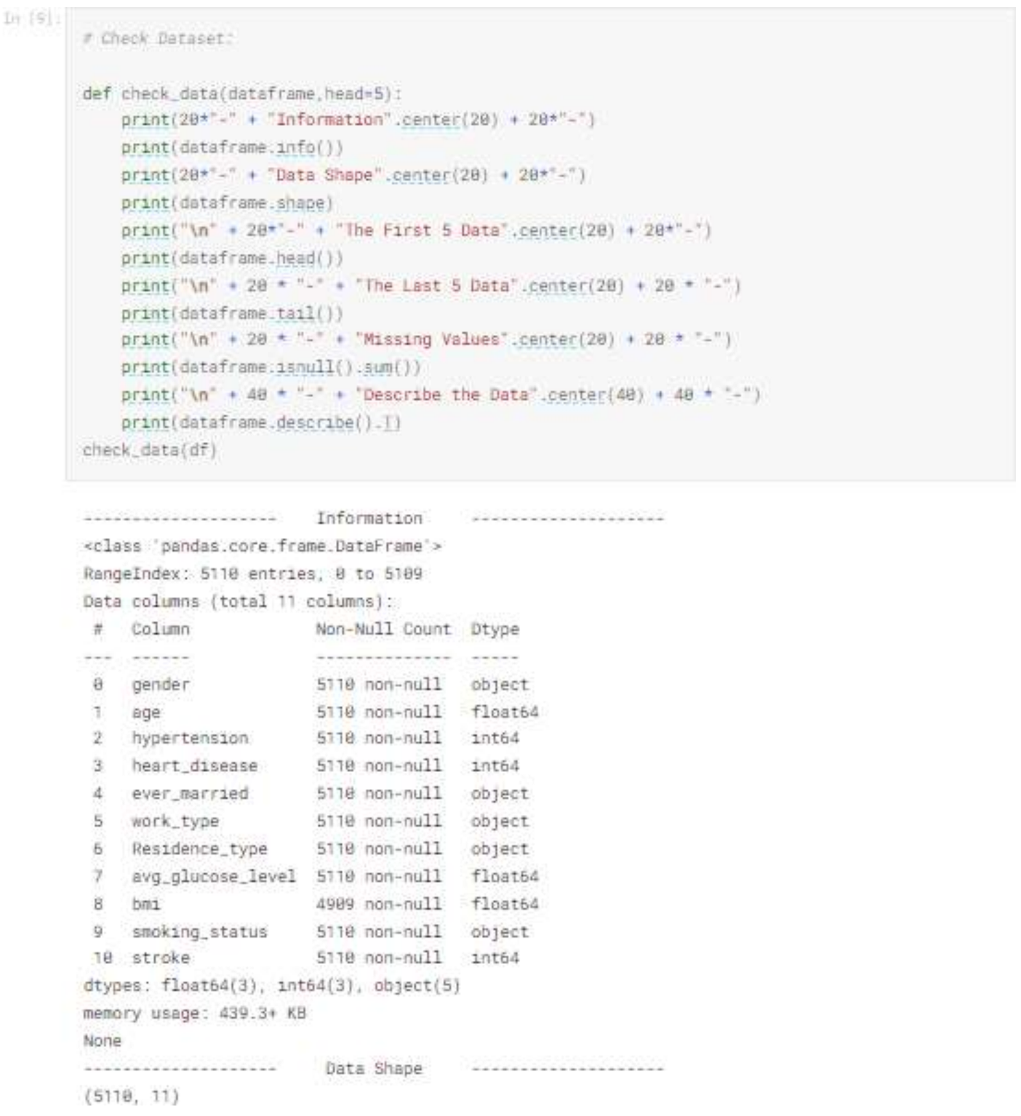
Target Variable Analysis:



```
df.drop(columns=['id'], inplace=True)

df.head()
```

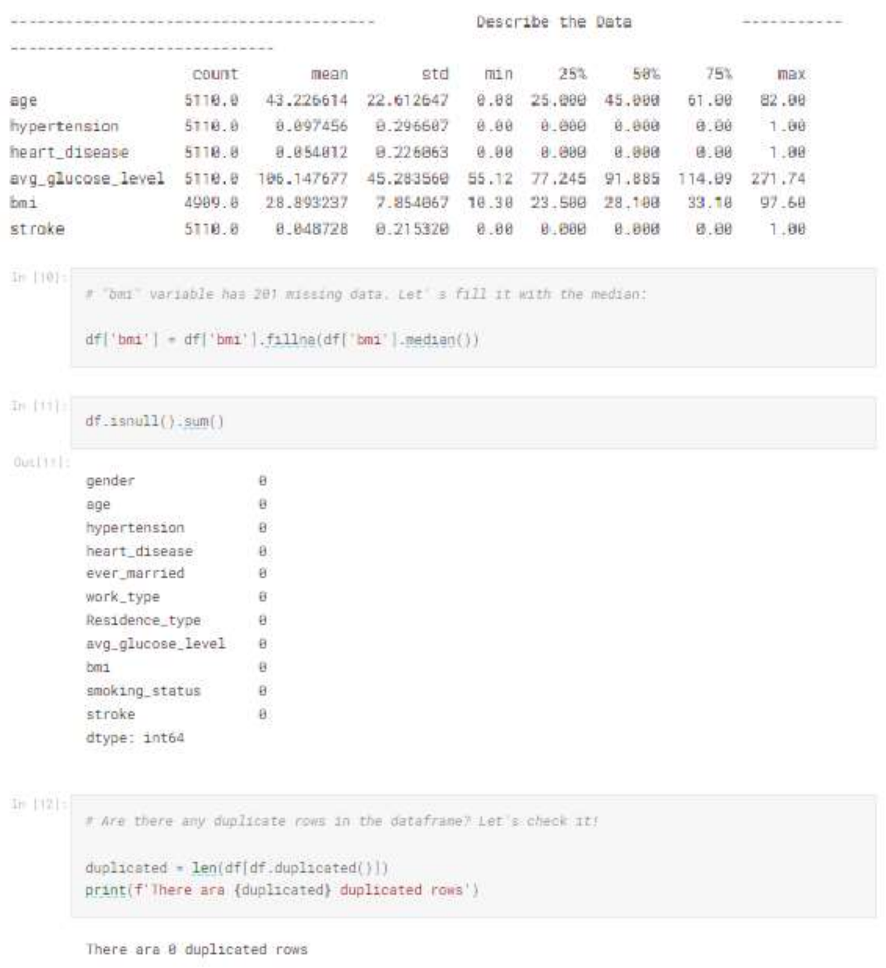
gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_sta
Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked
Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoke
Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoke
Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes
Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoke

Data Overview:



```
In [9]:
# Check Dataset:

def check_data(dataframe,head=5):
    print(20*"-" + "Information".center(20) + 20*"-")
    print(dataframe.info())
    print(20*"-" + "Data Shape".center(20) + 20*"-")
    print(dataframe.shape)
    print("\n" + 20*"-" + "The First 5 Data".center(20) + 20*"-")
    print(dataframe.head())
    print("\n" + 20 * "-" + "The Last 5 Data".center(20) + 20 * "-")
    print(dataframe.tail())
    print("\n" + 20 * "-" + "Missing Values".center(20) + 20 * "-")
    print(dataframe.isnull().sum())
    print("\n" + 40 * "-" + "Describe the Data".center(40) + 40 * "-")
    print(dataframe.describe().T)
check_data(df)
```

```
--------------------    Information     --------------------
<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
None
--------------------     Data Shape     --------------------
(5110, 11)
```



```
----------------------------------------          Describe the Data           -----------
-----------------------------
                    count        mean        std    min     25%     50%     75%     max
age                5110.0   43.226614  22.612647   0.08  25.000  45.000   61.00   82.00
hypertension       5110.0    0.097456   0.296607   0.00   0.000   0.000    0.00    1.00
heart_disease      5110.0    0.054012   0.226063   0.00   0.000   0.000    0.00    1.00
avg_glucose_level  5110.0  106.147677  45.283560  55.12  77.245  91.885  114.09  271.74
bmi                4909.0   28.893237   7.854067  10.30  23.500  28.100   33.10   97.60
stroke             5110.0    0.048728   0.215320   0.00   0.000   0.000    0.00    1.00
```

```
In [10]:
# "bmi" variable has 201 missing data. Let' s fill it with the median:

df['bmi'] = df['bmi'].fillna(df['bmi'].median())
```

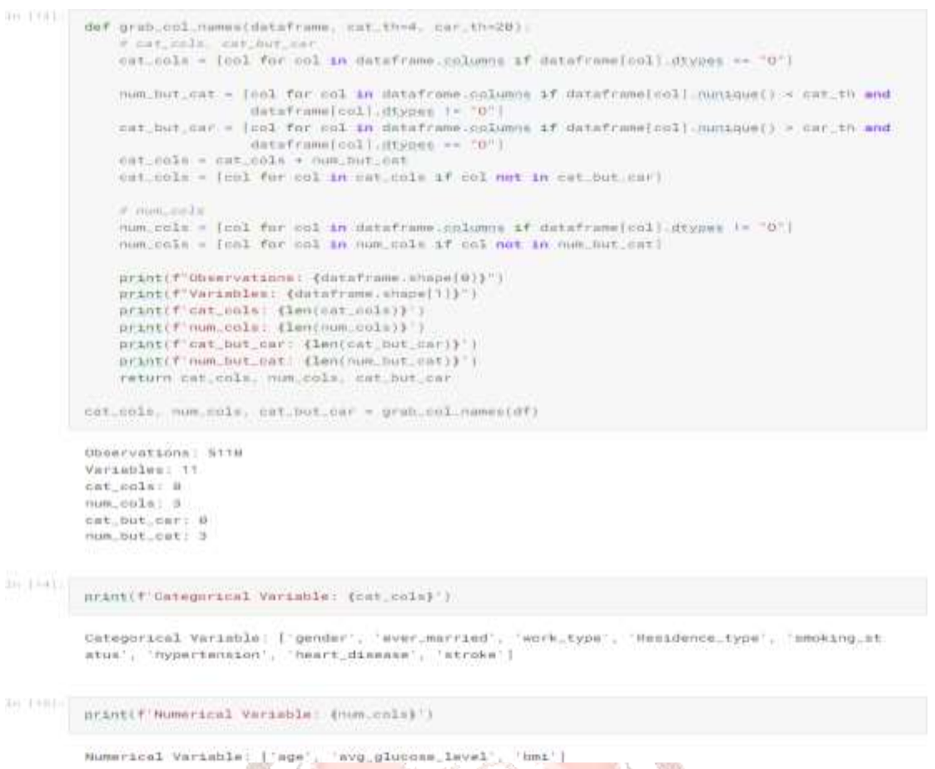
```
In [11]:
df.isnull().sum()
```

```
Out[11]:
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 [12]:
# Are there any duplicate rows in the dataframe? Let's check it!

duplicated = len(df[df.duplicated()])
print(f'There ara {duplicated} duplicated rows')
```

```
There ara 0 duplicated rows
```

Categorical Variables and Numerical Variables:



```
In [13]:
def grab_col_names(dataframe, cat_th=4, car_th=20):
    # cat_cols, cat_but_car
    cat_cols = [col for col in dataframe.columns if dataframe[col].dtypes == "O"]

    num_but_cat = [col for col in dataframe.columns if dataframe[col].nunique() < cat_th and
                   dataframe[col].dtypes != "O"]
    cat_but_car = [col for col in dataframe.columns if dataframe[col].nunique() > car_th and
                   dataframe[col].dtypes == "O"]
    cat_cols = cat_cols + num_but_cat
    cat_cols = [col for col in cat_cols if col not in cat_but_car]

    # num_cols
    num_cols = [col for col in dataframe.columns if dataframe[col].dtypes != "O"]
    num_cols = [col for col in num_cols if col not in num_but_cat]

    print(f"Observations: {dataframe.shape[0]}")
    print(f"Variables: {dataframe.shape[1]}")
    print(f'cat_cols: {len(cat_cols)}')
    print(f'num_cols: {len(num_cols)}')
    print(f'cat_but_car: {len(cat_but_car)}')
    print(f'num_but_cat: {len(num_but_cat)}')
    return cat_cols, num_cols, cat_but_car

cat_cols, num_cols, cat_but_car = grab_col_names(df)
```

```
Observations: 5110
Variables: 11
cat_cols: 8
num_cols: 3
cat_but_car: 0
num_but_cat: 3
```

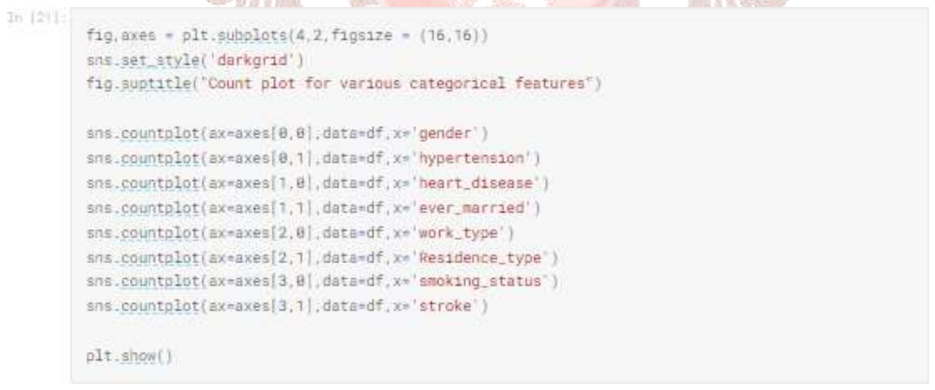
```
In [14]:
print(f'Categorical Variable: {cat_cols}')
```

```
Categorical Variable: ['gender', 'ever_married', 'work_type', 'Residence_type', 'smoking_st
atus', 'hypertension', 'heart_disease', 'stroke']
```

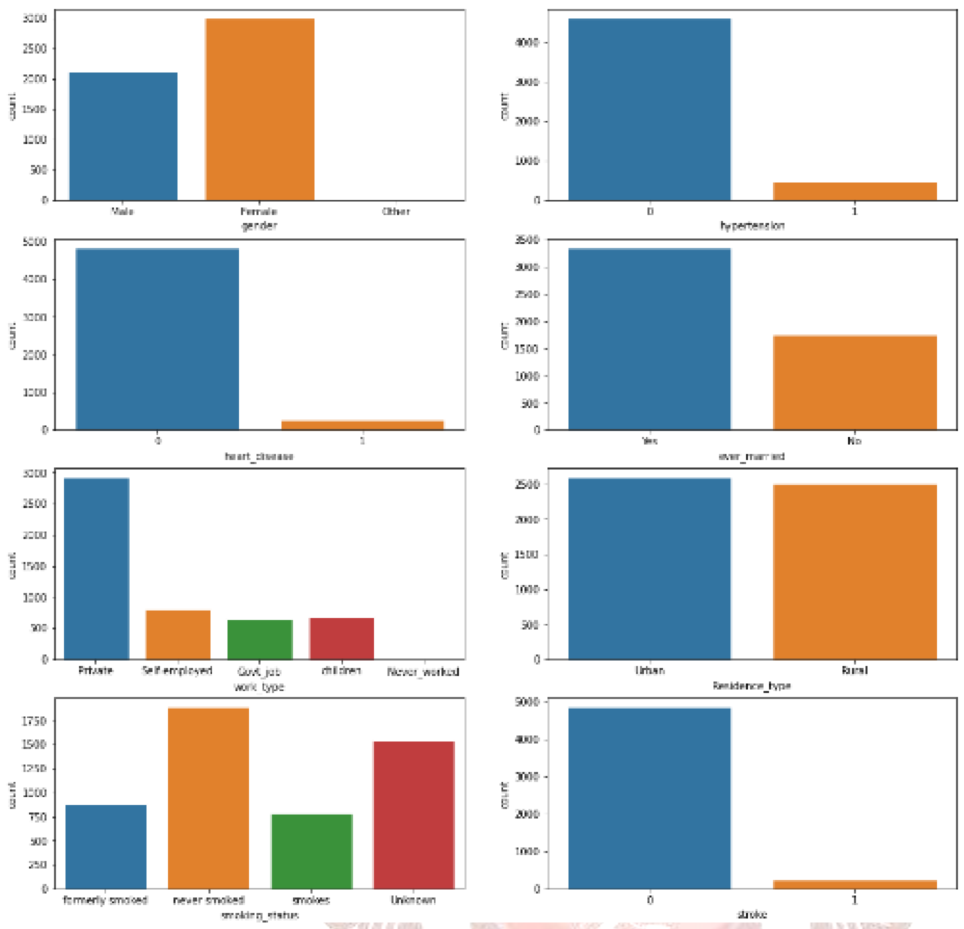
```
In [16]:
print(f'Numerical Variable: {num_cols}')
```

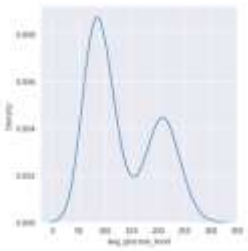
```
Numerical Variable: ['age', 'avg_glucose_level', 'bmi']
```

Visualization of Categorical Variables:

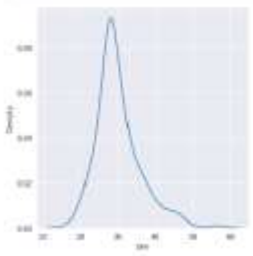


```
In [21]:
fig,axes = plt.subplots(4,2,figsize = (16,16))
sns.set_style('darkgrid')
fig.suptitle("Count plot for various categorical features")

sns.countplot(ax=axes[0,0],data=df,x='gender')
sns.countplot(ax=axes[0,1],data=df,x='hypertension')
sns.countplot(ax=axes[1,0],data=df,x='heart_disease')
sns.countplot(ax=axes[1,1],data=df,x='ever_married')
sns.countplot(ax=axes[2,0],data=df,x='work_type')
sns.countplot(ax=axes[2,1],data=df,x='Residence_type')
sns.countplot(ax=axes[3,0],data=df,x='smoking_status')
sns.countplot(ax=axes[3,1],data=df,x='stroke')

plt.show()
```


count
gender
Male
Female
Other
hypertension
0
1
heart_disease
ever_married
Yes
No
work_type
Private
Self-employed
Govt_job
children
Never_worked
Residence_type
Urban
Rural
smoking_status
formerly smoked
never smoked
smokes
Unknown
stroke



```
# Relationship of stroke and bmi

stroke = df[df['stroke']==1]
sns.displot(stroke.bmi, kind='kde')
plt.show()
```

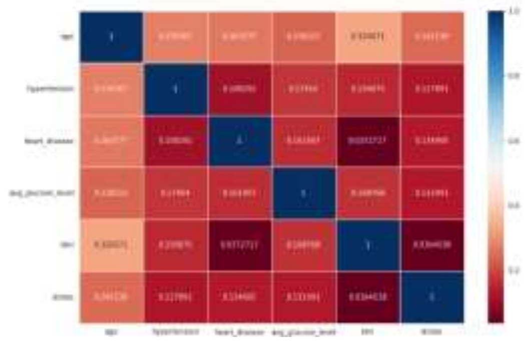


```
data_cat=df[['gender','hypertension','heart_disease','ever_married','work_type','Residence
_type',
                    'smoking_status']]
fig=plt.figure(figsize=(20,23))
background_color = '#f6f5f7'
fig.patch.set_facecolor(background_color)
for indx,val in enumerate(data_cat.columns):
    ax=plt.subplot(4,2,indx+1)
    ax.set_facecolor(background_color)
    ax.set_title(val,fontweight='bold',fontfamily='serif')
    for i in ['top','right']:
        ax.spines[i].set_visible(False)
    ax.grid(linestyle=':',axis='y')
    sns.countplot(data_cat[val],palette='OrRd_r',hue=df['stroke'])
```

gender
hypertension
heart_disease
ever_married
work_type
Residence_type
smoking_status

Heatmap of Numerical Variables

```
fig = plt.figure(figsize=(12,8))
corr = df.corr()
ax = sns.heatmap(corr, linewidths=.5, cmap="RdBu", annot=True, fmt='g')
```



Data Preprocessing:

```
# "age" Analysis

print(f'Age Variable min: {df["age"].min()}')
print(f'Age Variable max: {df["age"].max()}')
print(f'Age Variable: {df["age"].nunique()}')
```

```
Age Variable min: 0.08
Age Variable max: 82.0
Age Variable: 104
```

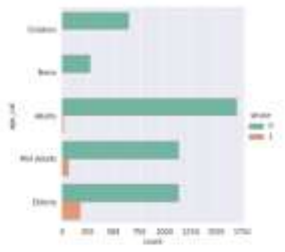
```
df['age_cat'] = pd.cut(df['age'], bins=[0, 13, 18, 45, 60, 100], labels=['Children', 'Teens',
'Adults', 'Mid Adults', 'Elderly'])
```

```
df['age_cat'].unique()
```

```
['Elderly', 'Mid Adults', 'Adults', 'Children', 'Teens']
Categories (5, object): ['Children' < 'Teens' < 'Adults' < 'Mid Adults' < 'Elderly']
```

```
sns.catplot(y="age_cat", hue="stroke", kind="count",
            palette="Set2", edgecolor=".6",
            data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f63f82db190>
```

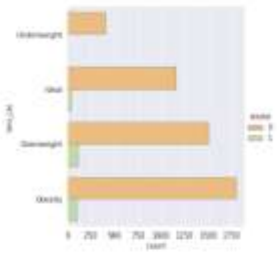


```
# "bmi" Analysis

print(f'BMI Variable min: {df["bmi"].min()}')
print(f'BMI Variable max: {df["bmi"].max()}')
print(f'BMI Variable: {df["bmi"].nunique()}')
```

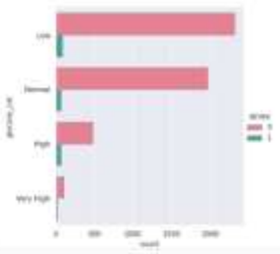
```
BMI Variable min: 10.3
BMI Variable max: [illegible]
BMI Variable: 417
```

```
df['bmi_cat'] = pd.cut(df['bmi'], bins=[0, 19, 25, 30, 100], labels=['Underweight', 'Ideal',
'Overweight', 'Obesity'])
```

```
In [33]: df['bmi_cat'].unique()

Out[33]:
['Obesity', 'Overweight', 'Ideal', 'Underweight']
Categories (4, object): ['Underweight' < 'Ideal' < 'Overweight' < 'Obesity']

In [34]: sns.catplot(y="bmi_cat", hue="stroke", kind="count",
                     palette='Spectral', edgecolor=".6",
                     data=df)

Out[34]: <seaborn.axisgrid.FacetGrid at 0x7f6394fcb7d0>
```



```
In [35]: # 'avg_glucose_level' analysis:

         print(f"Glucose Variable min: {df['avg_glucose_level'].min()}")
         print(f"Glucose Variable max: {df['avg_glucose_level'].max()}")
         print(f"Glucose Variable: {df['avg_glucose_level'].nunique()}")

Glucose Variable min: 55.12
Glucose Variable max: 271.74
Glucose Variable: 3979

In [36]: df['glucose_cat'] = pd.cut(df['avg_glucose_level'], bins=[0, 90, 160, 230, 500], labels=['Lo
         w', 'Normal', 'High', 'Very High'])

In [37]: df['glucose_cat'].unique()

Out[37]:
['High', 'Normal', 'Low', 'Very High']
Categories (4, object): ['Low' < 'Normal' < 'High' < 'Very High']

In [38]: sns.catplot(y="glucose_cat", hue="stroke", kind="count",
                     palette="husl", edgecolor=".6",
                     data=df)

Out[38]: <seaborn.axisgrid.FacetGrid at 0x7f6394f83f10>
```

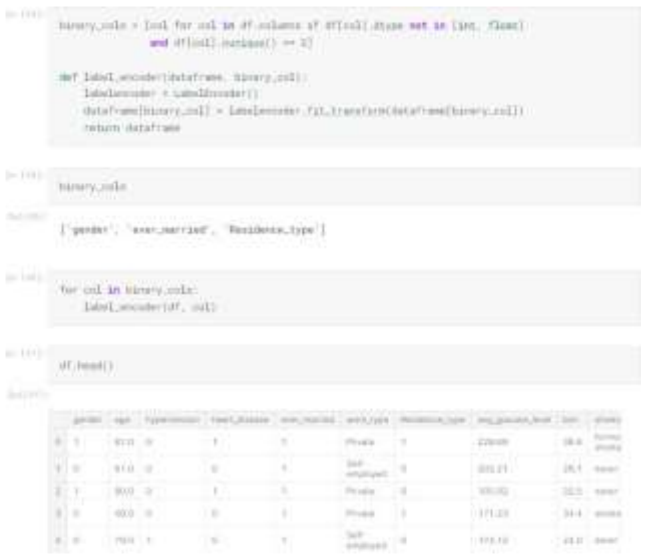


```
In [39]: # Note: Glucose values are taken according to international acceptance.

In [40]: df.head()

Out[40]:
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smokin
0	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked
1	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	28.1	never
2	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never
3	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smoke
4	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never

Vol. 7
Issue 3

Binary Encoding



```
binary_cols = [col for col in df.columns if df[col].dtype not in [int, float]
              and df[col].nunique() == 2]

def label_encoder(dataframe, binary_col):
    labelencoder = LabelEncoder()
    dataframe[binary_col] = labelencoder.fit_transform(dataframe[binary_col])
    return dataframe
```

```
binary_cols
```

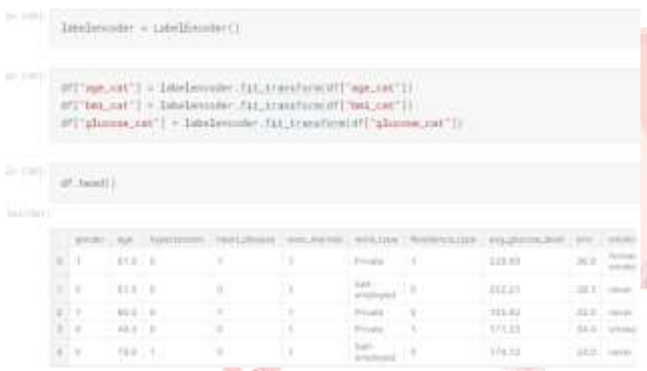
```
['gender', 'ever_married', 'Residence_type']
```

```
for col in binary_cols:
    label_encoder(df, col)
```

```
df.head()
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking
0	1	67.0	0	1	1	Private	1	228.69	36.6	formerly smoking
1	0	61.0	0	0	1	Self-employed	0	202.21	28.1	never
2	1	80.0	0	1	1	Private	0	105.92	32.5	never
3	0	49.0	0	0	1	Private	1	171.23	34.4	smoke
4	0	79.0	1	0	1	Self-employed	0	174.12	24.0	never

Label Encoding:



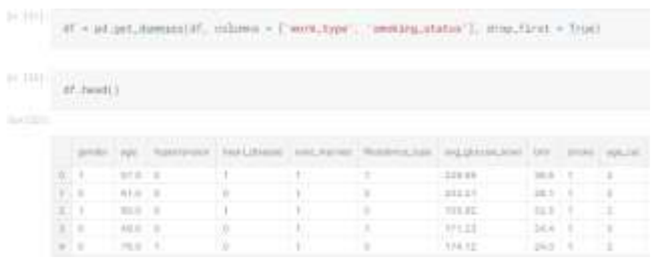
```
labelencoder = LabelEncoder()
```

```
df["age_cat"] = labelencoder.fit_transform(df["age_cat"])
df["bmi_cat"] = labelencoder.fit_transform(df["bmi_cat"])
df["glucose_cat"] = labelencoder.fit_transform(df["glucose_cat"])
```

```
df.head()
```

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smokin
0	1	67.0	0	1	1	Private	1	228.69	36.6	formerl smoked
1	0	61.0	0	0	1	Self-employed	0	202.21	28.1	never
2	1	80.0	0	1	1	Private	0	105.92	32.5	never
3	0	49.0	0	0	1	Private	1	171.23	34.4	smoke
4	0	79.0	1	0	1	Self-employed	0	174.12	24.0	never

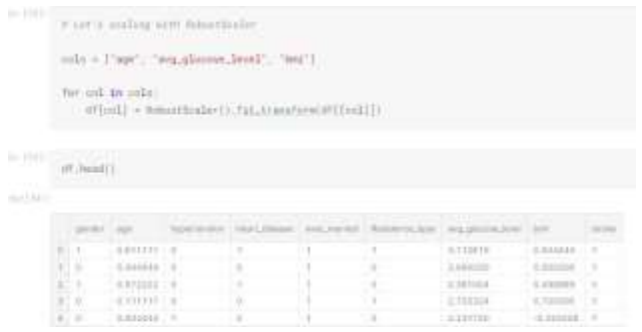
One-Hot Encoding:



```
df = pd.get_dummies(df, columns = ['work_type', 'smoking_status'], drop_first = True)
```

```
df.head()
```

	gender	age	hypertension	heart_disease	ever_married	Residence_type	avg_glucose_level	bmi	stroke	age_cat
0	1	67.0	0	1	1	1	228.69	36.6	1	2
1	0	61.0	0	0	1	0	202.21	28.1	1	[illegible]
2	1	80.0	0	1	1	0	105.92	32.5	1	2
3	0	49.0	0	0	1	1	171.23	34.4	1	[illegible]
4	0	79.0	1	0	1	0	174.12	24.0	1	2

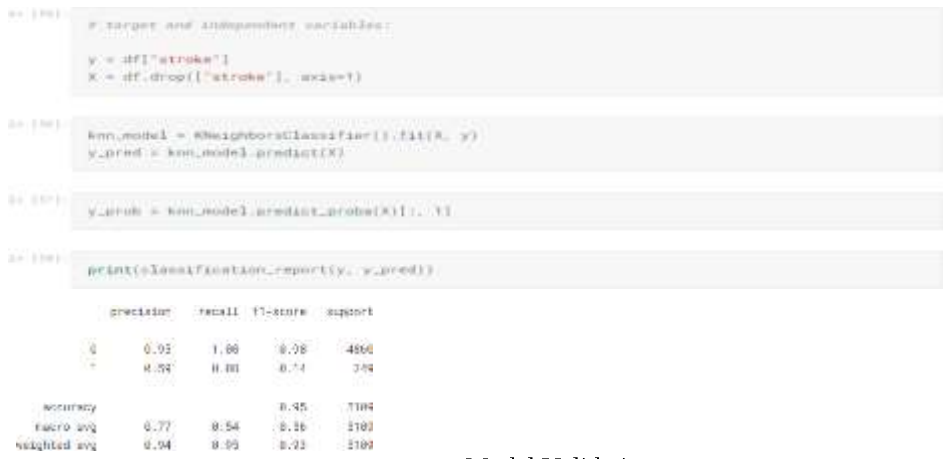


```
# Let's scaling with RobustScaler

cols = ["age", "avg_glucose_level", "bmi"]

for col in cols:
    df[col] = RobustScaler().fit_transform(df[[col]])
```

```
df.head()
```

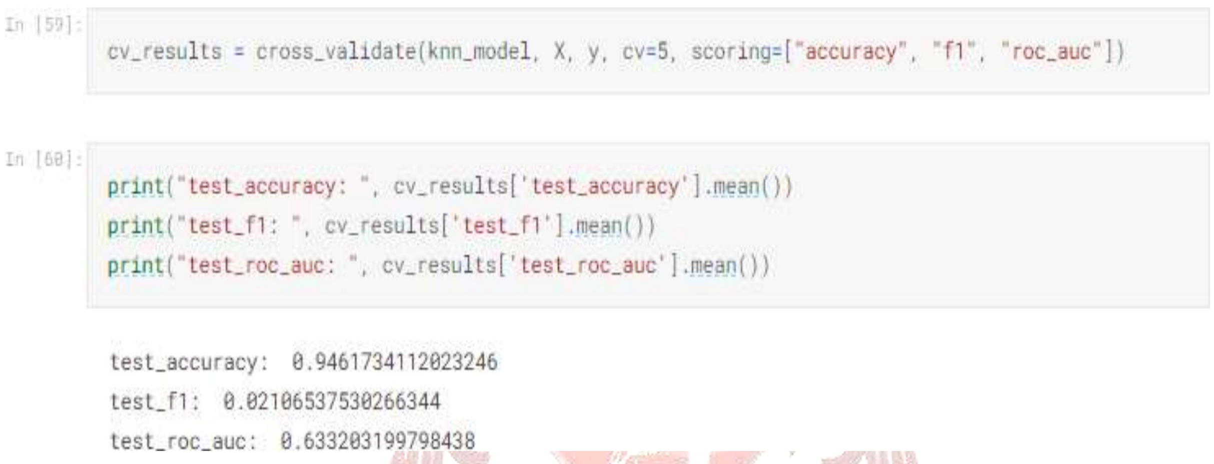
	gender	age	hypertension	heart_disease	ever_married	Residence_type	avg_glucose_level	bmi	stroke
0	1	0.611111	0	1	1	1	3.713816	0.944444	1
1	0	0.444444	0	0	1	0	2.984330	0.000000	1
2	1	0.972222	0	1	1	0	0.387004	0.488889	1
3	0	0.111111	0	0	1	1	2.155324	0.700000	1
4	0	0.833333	1	0	1	0	2.235700	-0.355556	1

Machine Learning Model K-Nearest Neighbor Model



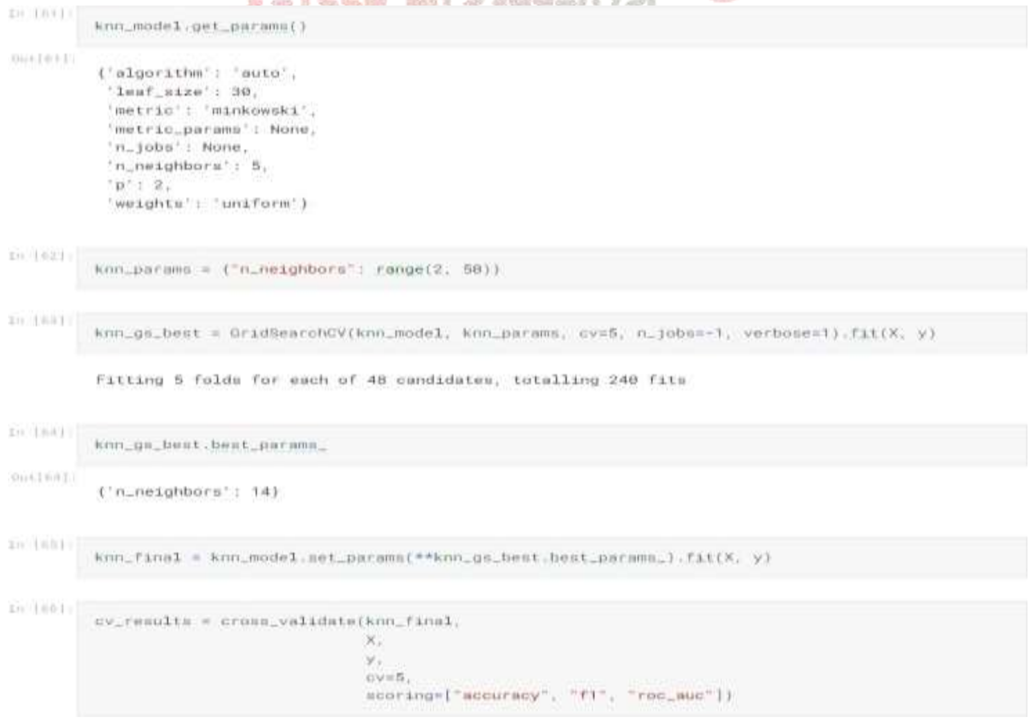
```
In [55]:  # target and independent variables:

          y = df["stroke"]
          X = df.drop(["stroke"], axis=1)

In [56]:  knn_model = KNeighborsClassifier().fit(X, y)
          y_pred = knn_model.predict(X)

In [57]:  y_prob = knn_model.predict_proba(X)[:, 1]

In [58]:  print(classification_report(y, y_pred))

                   precision    recall  f1-score   support

                0       0.95      1.00      0.98      4860
                1       0.59      0.08      0.14       249

         accuracy                           0.95      5109
        macro avg       0.77      0.54      0.56      5109
     weighted avg       0.94      0.95      0.93      5109
```

Model Validation



```
In [59]:  cv_results = cross_validate(knn_model, X, y, cv=5, scoring=["accuracy", "f1", "roc_auc"])

In [60]:  print("test_accuracy: ", cv_results['test_accuracy'].mean())
          print("test_f1: ", cv_results['test_f1'].mean())
          print("test_roc_auc: ", cv_results['test_roc_auc'].mean())

          test_accuracy:  0.9461734112023246
          test_f1:  0.02106537530266344
          test_roc_auc:  0.633203199798438
```

Hyperparameter Optimization:



```
In [61]:  knn_model.get_params()

Out[61]:  {'algorithm': 'auto',
           'leaf_size': 30,
           'metric': 'minkowski',
           'metric_params': None,
           'n_jobs': None,
           'n_neighbors': 5,
           'p': 2,
           'weights': 'uniform'}

In [62]:  knn_params = {"n_neighbors": range(2, 50)}

In [63]:  knn_gs_best = GridSearchCV(knn_model, knn_params, cv=5, n_jobs=-1, verbose=1).fit(X, y)

          Fitting 5 folds for each of 48 candidates, totalling 240 fits

In [64]:  knn_gs_best.best_params_

Out[64]:  {'n_neighbors': 14}

In [65]:  knn_final = knn_model.set_params(**knn_gs_best.best_params_).fit(X, y)

In [66]:  cv_results = cross_validate(knn_final,
                                      X,
                                      y,
                                      cv=5,
                                      scoring=["accuracy", "f1", "roc_auc"])
```