Stroke

August 25, 2024

1 Stroke Machine Learning Classification Analysis

1.1 Load Libraries

[ ]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

1.2 Load dataset

Data source: kaggle

[ ]:
```
df = pd.read_csv("/content/healthcare-dataset-stroke-data.csv")
df
```

[ ]:

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked
…	…	…	…	…	…	…	…	…	…	…	…
5105	18234	Female	80.0	1	0	Yes	Private	Urban	83.75	NaN	never smoked
5106	44873	Female	81.0	0	0	Yes	Self-employed	Urban	125.20	40.0	never smoked
5107	19723	Female	35.0	0	0	Yes	Self-employed	Rural	82.99	30.6	never smoked
5108	37544	Male	51.0	0	0	Yes					
5109	44679	Female	44.0	0	0	Yes					

```
5108        Private          Rural             166.29  25.6  formerly smoked
5109       Govt_job          Urban              85.28  26.2          Unknown

      stroke
0          1
1          1
2          1
3          1
4          1
…        …
5105       0
5106       0
5107       0
5108       0
5109       0

[5110 rows x 12 columns]
```

[ ]:
```
# number of rows and columns
df.shape
```

[ ]:
```
(5110, 12)
```

[ ]:
```
# Data type and non-null count
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5110 entries, 0 to 5109
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 5110 non-null   int64
 1   gender             5110 non-null   object
 2   age                5110 non-null   float64
 3   hypertension       5110 non-null   int64
 4   heart_disease      5110 non-null   int64
 5   ever_married       5110 non-null   object
 6   work_type          5110 non-null   object
 7   Residence_type     5110 non-null   object
 8   avg_glucose_level  5110 non-null   float64
 9   bmi                4909 non-null   float64
 10  smoking_status     5110 non-null   object
 11  stroke             5110 non-null   int64
dtypes: float64(3), int64(4), object(5)
memory usage: 479.2+ KB
```

1.3 Data Cleaning

[ ]:
```
# drop id
df = df.drop('id', axis=1)
```

[ ]:
```
# How many missing values does bmi column have?
df['bmi'].isnull().sum()
```

[ ]:
```
201
```

[ ]:
```
# replace missing value with mean of bmi column
df['bmi'] = df['bmi'].fillna(df['bmi'].mean())
```

[ ]:
```
# Check number of missing values after replacement
df['bmi'].isnull().sum()
```

[ ]:
```
0
```

[ ]:
```
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                5110 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.head()
```

[ ]:

	gender	age	hypertension	heart_disease	ever_married	work_type
0	Male	67.0	0	1	Yes	Private
1	Female	61.0	0	0	Yes	Self-employed
2	Male	80.0	0	1	Yes	Private
3	Female	49.0	0	0	Yes	Private
4	Female	79.0	1	0	Yes	Self-employed

	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	Urban	228.69	36.600000	formerly smoked	1
1	Rural	202.21	28.893237	never smoked	1
2	Rural	105.92	32.500000	never smoked	1
3	Urban	171.23	34.400000	smokes	1
4	Rural	174.12	24.000000	never smoked	1

[ ]:
```
# descriptive statistics
df.describe()
```

[ ]:

	age	hypertension	heart_disease	avg_glucose_level	\
count	5110.000000	5110.000000	5110.000000	5110.000000	
mean	43.226614	0.097456	0.054012	106.147677	
std	22.612647	0.296607	0.226063	45.283560	
min	0.080000	0.000000	0.000000	55.120000	
25%	25.000000	0.000000	0.000000	77.245000	
50%	45.000000	0.000000	0.000000	91.885000	
75%	61.000000	0.000000	0.000000	114.090000	
max	82.000000	1.000000	1.000000	271.740000	

	bmi	stroke
count	5110.000000	5110.000000
mean	28.893237	0.048728
std	7.698018	0.215320
min	10.300000	0.000000
25%	23.800000	0.000000
50%	28.400000	0.000000
75%	32.800000	0.000000
max	97.600000	1.000000

[ ]:
```
# count of age < 18
len(df[df['age'] < 15])
```

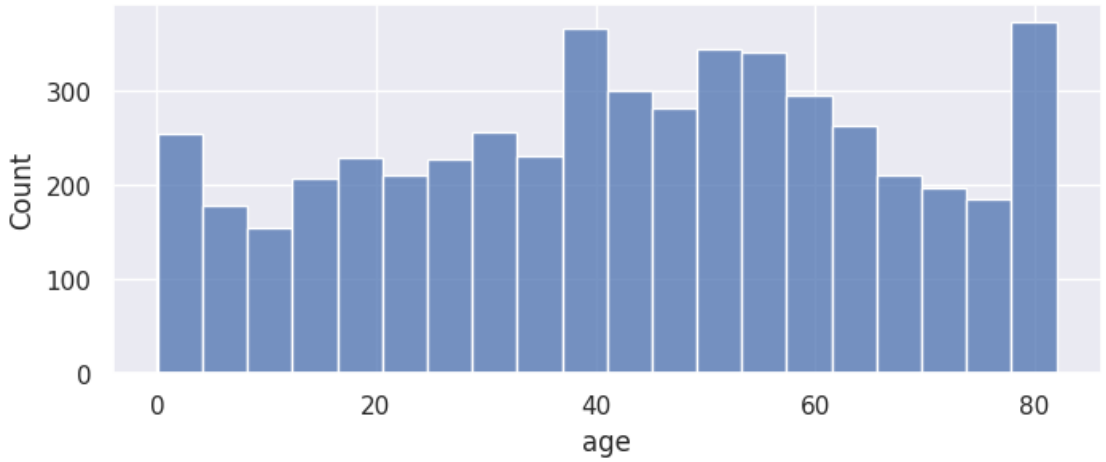
[ ]: 699

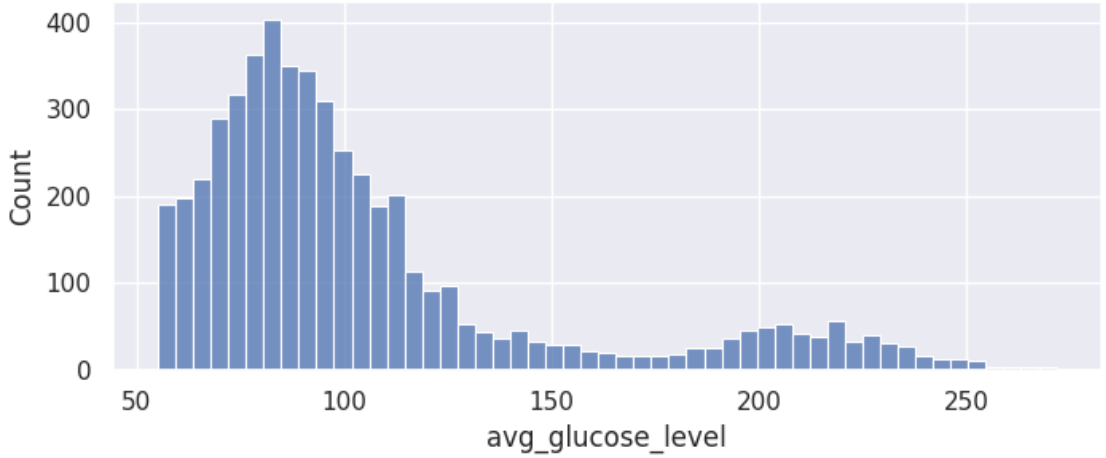
1.4 Exploratory Data Analysis

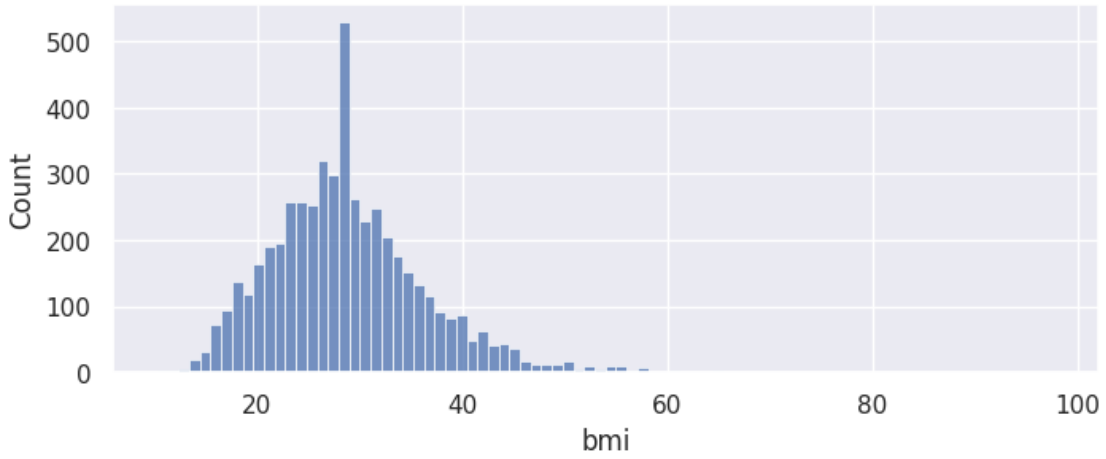
1.4.1 Histogram

[ ]:
```
columns = ['age', 'avg_glucose_level', 'bmi']
```

[ ]:
```
sns.set(rc={"figure.figsize":(8, 3)})

for i in columns:
  sns.histplot(data = df, x = i)
  plt.show()
```

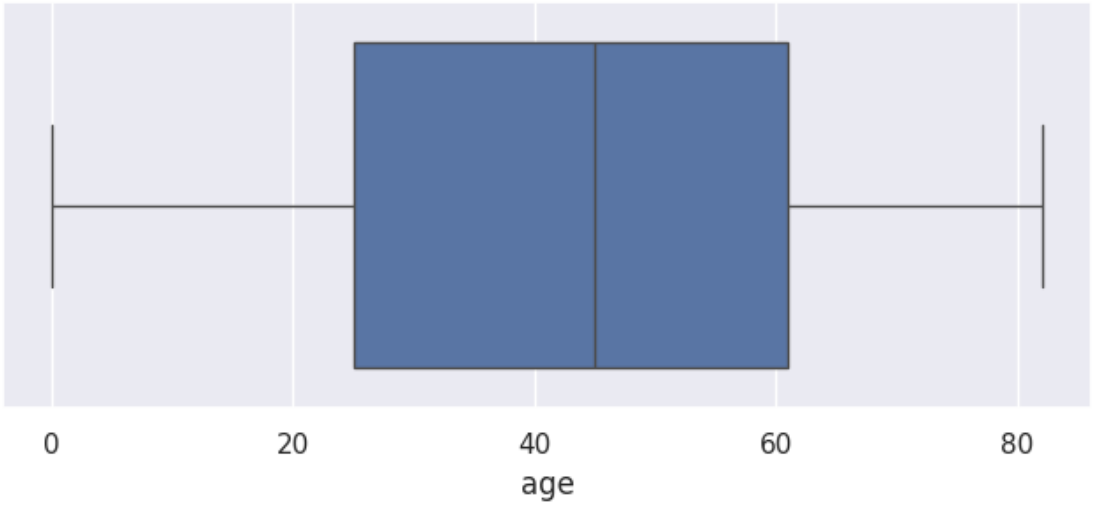

Count
300
200
100
0
0
20
40
60
80
age

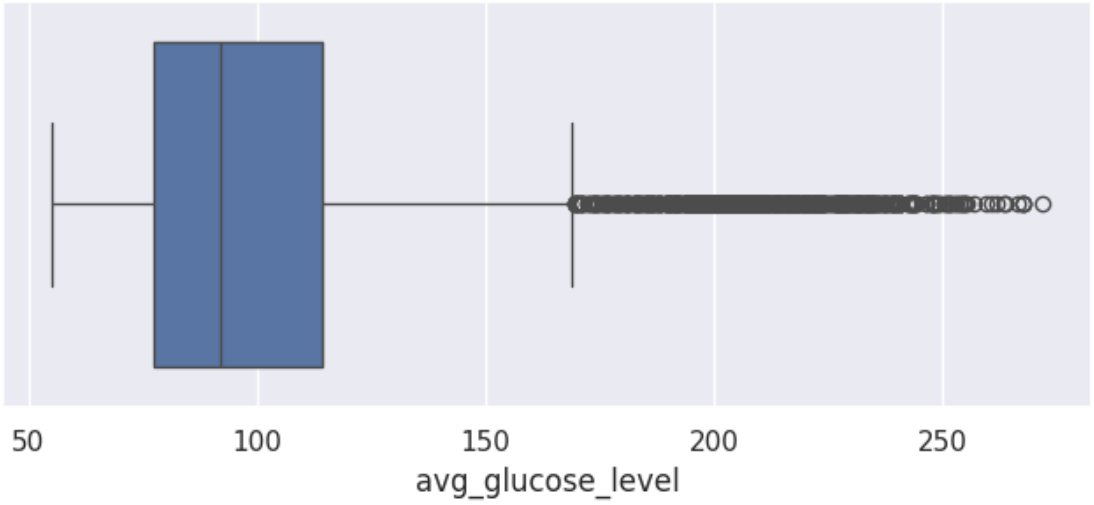

Count
400
300
200
100
0
50
100
150
200
250
avg_glucose_level

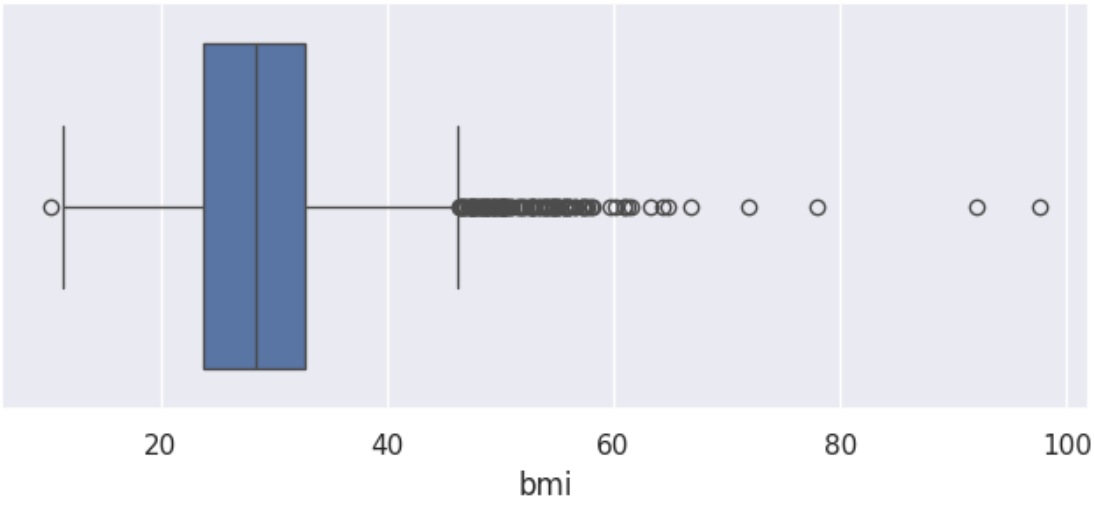

Count
500
400
300
200
100
0
20
40
60
80
100
bmi

1.4.2 Box Plots

```
[ ]: sns.set(rc={"figure.figsize":(8, 3)})

for i in columns:
  sns.boxplot(x = df[i])
  plt.show()
```

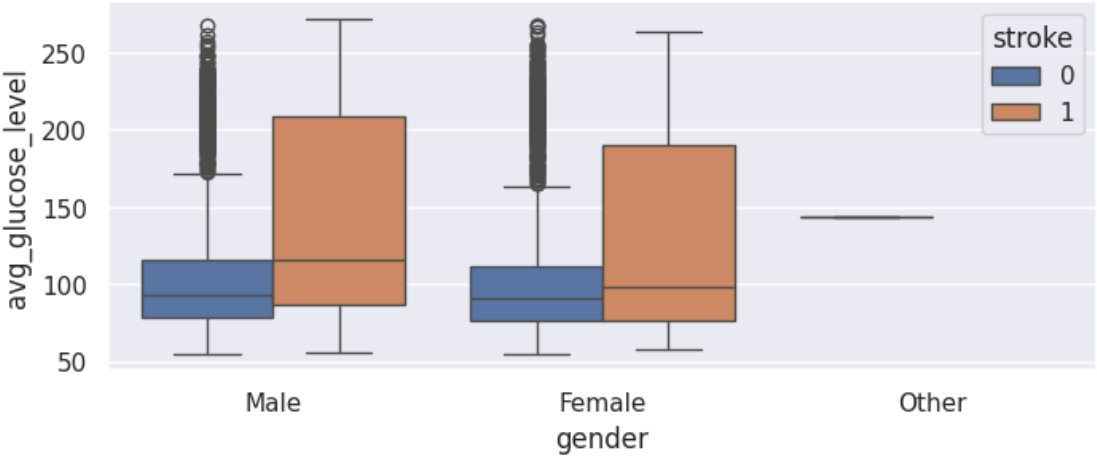






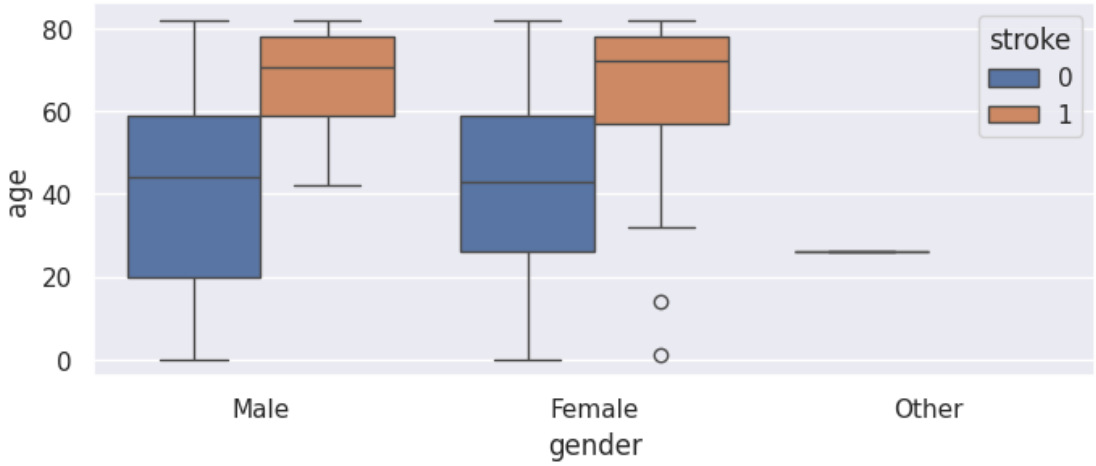
[ ]:
```
sns.boxplot(x = df['gender'], y = df['avg_glucose_level'], hue = df['stroke']  )
```

[ ]: <Axes: xlabel='gender', ylabel='avg_glucose_level'>



[ ]:
```
sns.boxplot(x = df['gender'], y = df['age'], hue = df['stroke']  )
```

[ ]: <Axes: xlabel='gender', ylabel='age'>



1.4.3 Counts

[ ]:
```
len(df[df['bmi'] > 46])
```

[ ]: 129

[ ]:
```
len(df[df['avg_glucose_level'] > 170])
```

[ ]: 622

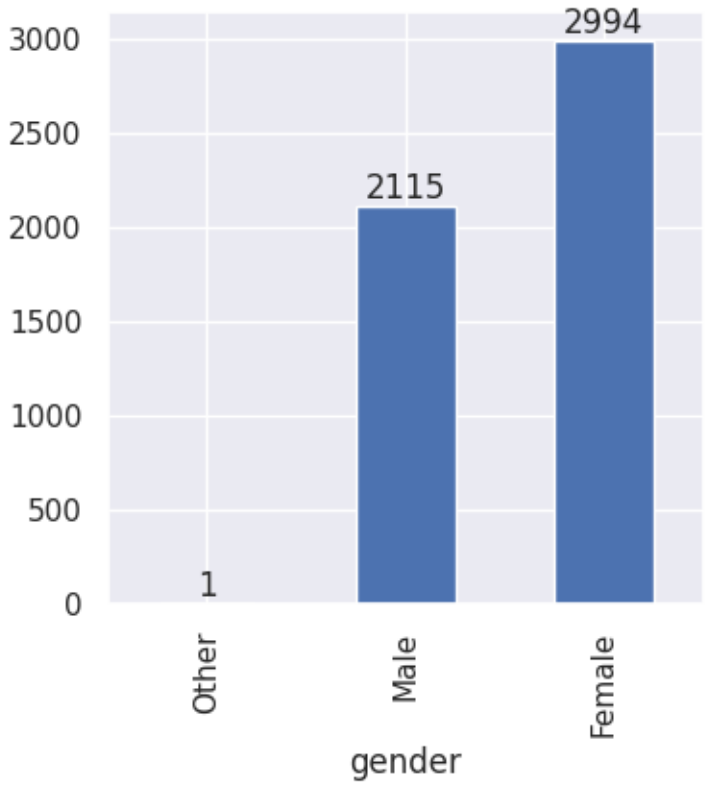
[ ]:
```
df[df['bmi'] > 46]
```

[ ]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
21	Female	52.0	1	0	Yes	Self-employed	Urban	233.29	48.9	never smoked	1
66	Female	70.0	0	0	Yes	Private	Urban	221.58	47.5	never smoked	1
113	Female	45.0	0	0	Yes	Private	Rural	224.10	56.6	never smoked	1
254	Female	47.0	0	0	Yes	Private	Urban	210.95	50.1	Unknown	0
258	Female	74.0	1	0	Yes	Self-employed	Urban	221.29	46.5	never smoked	1
…	…	…	…	…	…	…	…	…	…	…	…
4906	Female	53.0	0	0	Yes	Private	Urban	216.22	54.6	never smoked	0
4952	Male	51.0	1	0	Yes	Self-employed	Rural	214.09	47.3	never smoked	0
5009	Female	50.0	0	0	Yes	Self-employed	Urban	228.69	47.5	never smoked	0
5057	Female	49.0	0	0	Yes	Govt_job	Urban	171.23	47.5	formerly smoked	0
5103	Female	18.0	0	0	No	Private	Rural	85.28	47.5	Unknown	0

```
258          Urban              205.84  54.6     never smoked       0
…            …                    …     …              …       …
4906         Urban               70.51  54.1     never smoked       0
4952         Rural              211.83  56.6     never smoked       0
5009         Rural              126.85  49.5  formerly smoked       0
5057         Urban               69.92  47.6     never smoked       0
5103         Urban               82.85  46.9          Unknown       0

[129 rows x 11 columns]
```

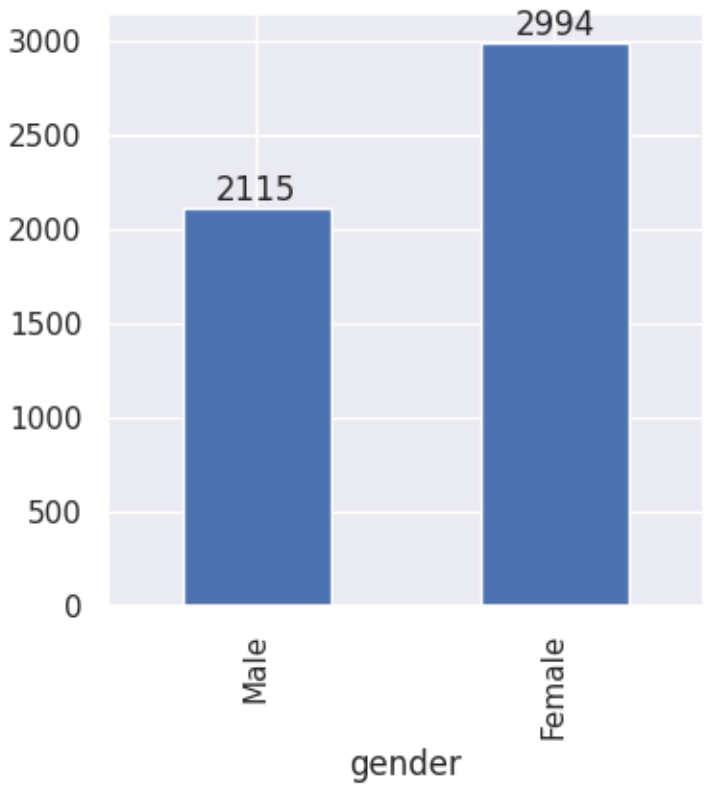
[ ]:
```
ax,fig = plt.subplots(figsize=(4,4))
ax=df.gender.value_counts().sort_values(ascending=True).plot(kind="bar")
ax.bar_label(ax.containers[0])
plt.show()
```



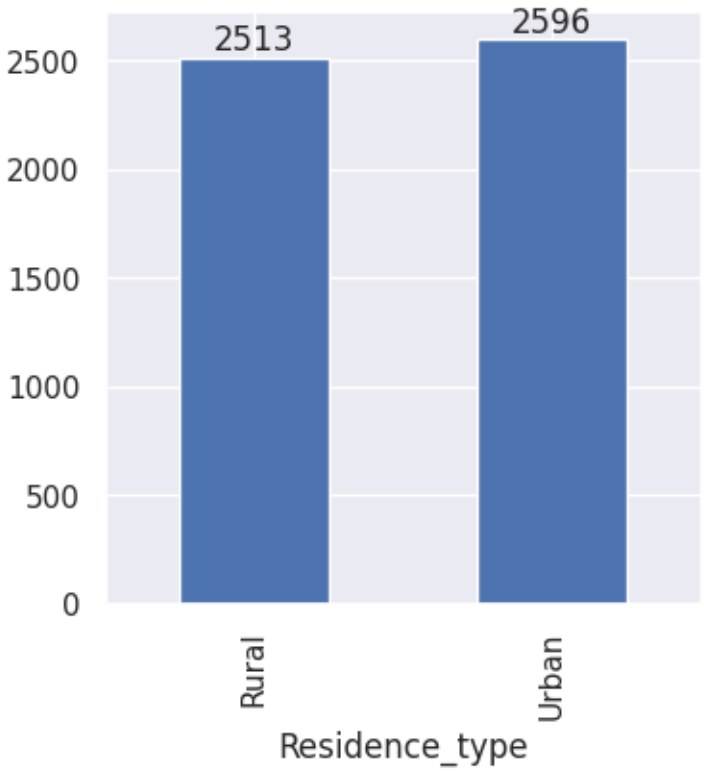
[ ]:
```
#drop row that contains specific 'Other' in gender
df = df[df.gender != 'Other'].copy()
```

[ ]:
```
ax,fig = plt.subplots(figsize=(4,4))
ax=df.gender.value_counts().sort_values(ascending=True).plot(kind="bar")
ax.bar_label(ax.containers[0])
```

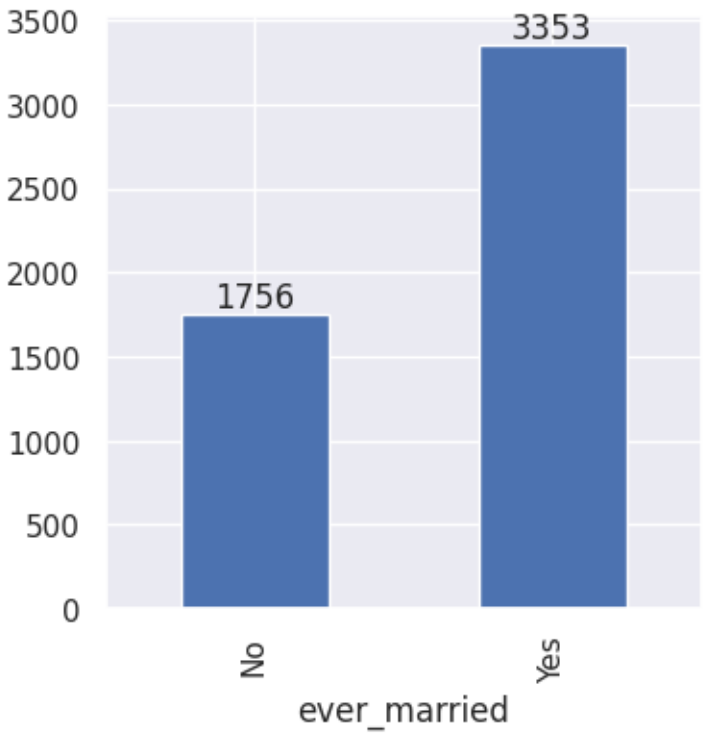
```
plt.show()
```



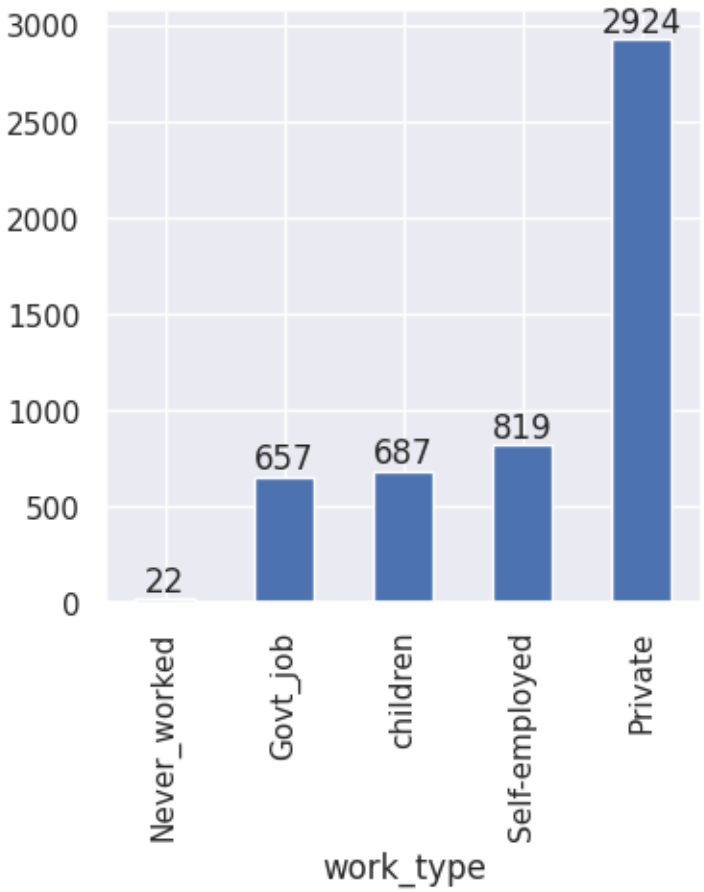
[ ]:
```
ax,fig = plt.subplots(figsize=(4,4))
ax=df.Residence_type.value_counts().sort_values(ascending=True).plot(kind="bar")
ax.bar_label(ax.containers[0])
plt.show()
```



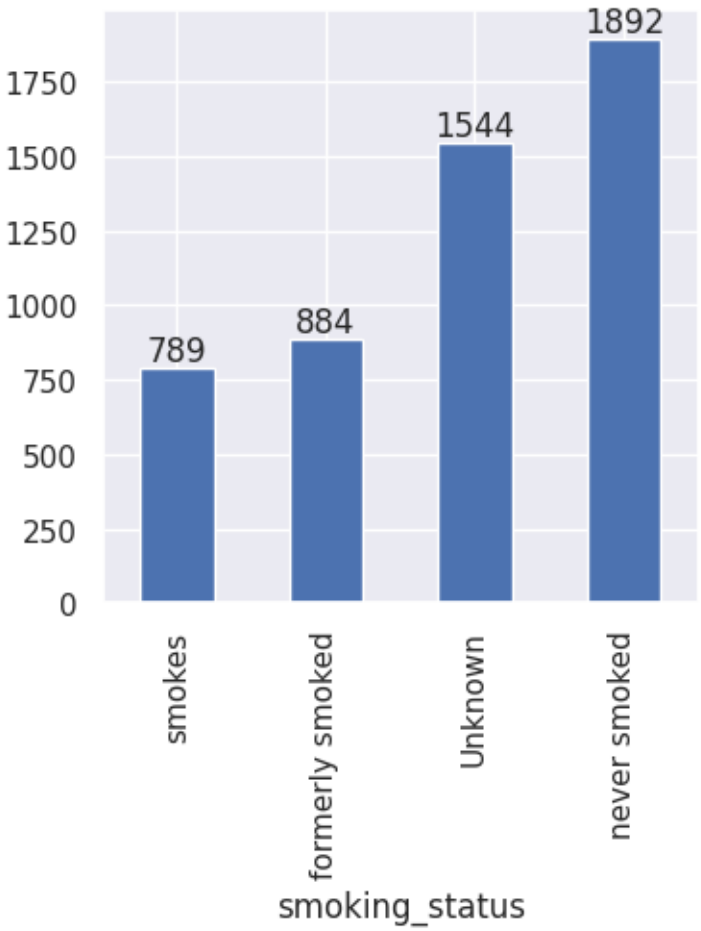
[ ]:
```
ax,fig = plt.subplots(figsize=(4,4))
ax=df.ever_married.value_counts().sort_values(ascending=True).plot(kind="bar")
ax.bar_label(ax.containers[0])
plt.show()
```



[ ]:
```
ax,fig = plt.subplots(figsize=(4,4))
ax=df.work_type.value_counts().sort_values(ascending=True).plot(kind="bar")
ax.bar_label(ax.containers[0])
plt.show()
```



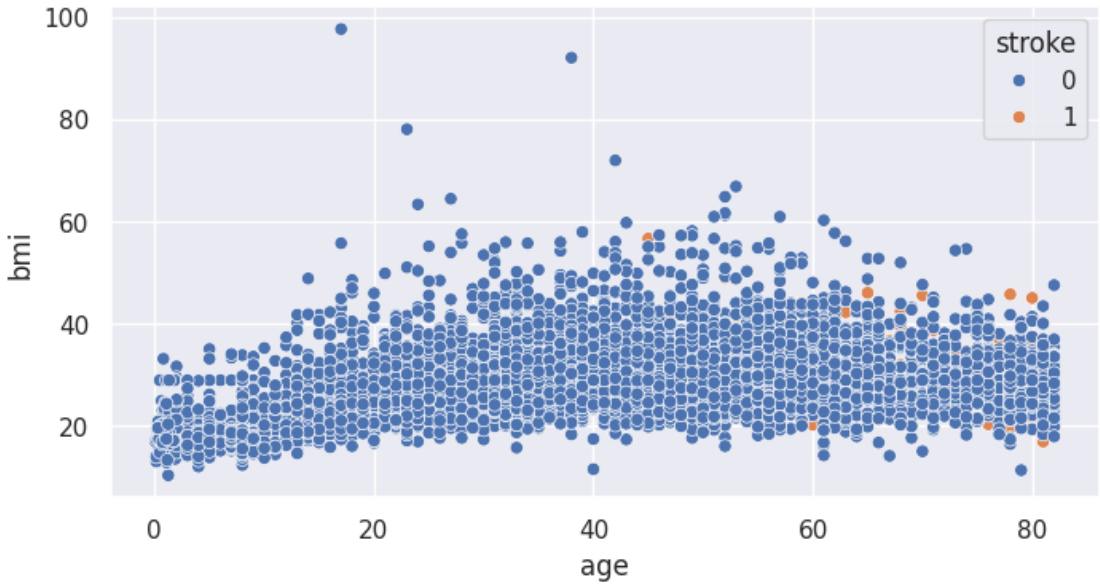
[ ]:
```
ax,fig = plt.subplots(figsize=(4,4))
ax=df.smoking_status.value_counts().sort_values(ascending=True).plot(kind="bar")
ax.bar_label(ax.containers[0])
plt.show()
```



1.4.4 Scatter Plots

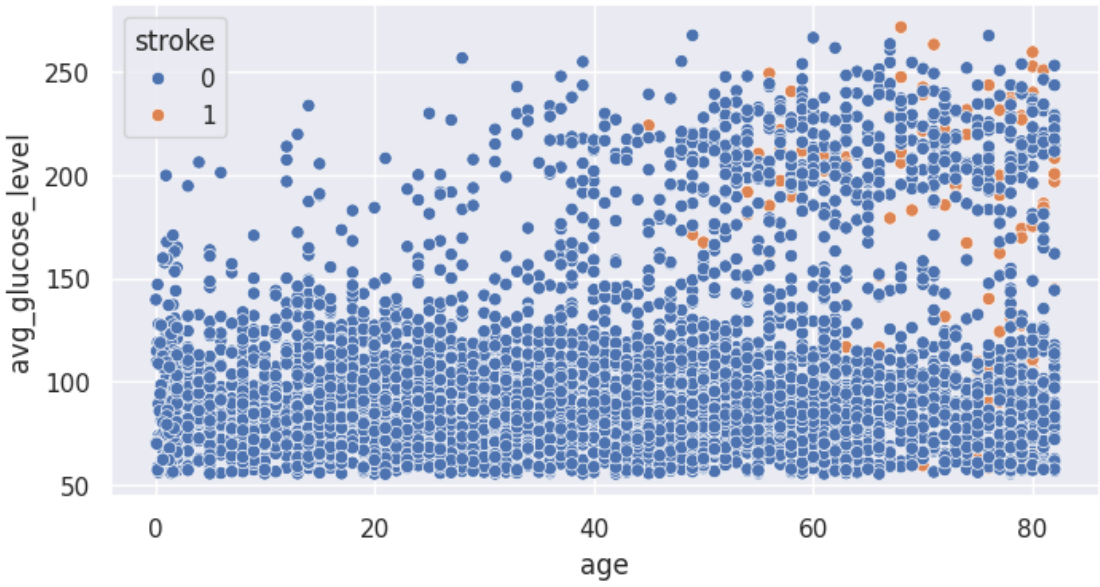
[ ]:
```
sns.set(rc={"figure.figsize":(8, 4)})
sns.scatterplot(data=df, x='age', y='bmi', hue='stroke')
```

[ ]:
```
<Axes: xlabel='age', ylabel='bmi'>
```



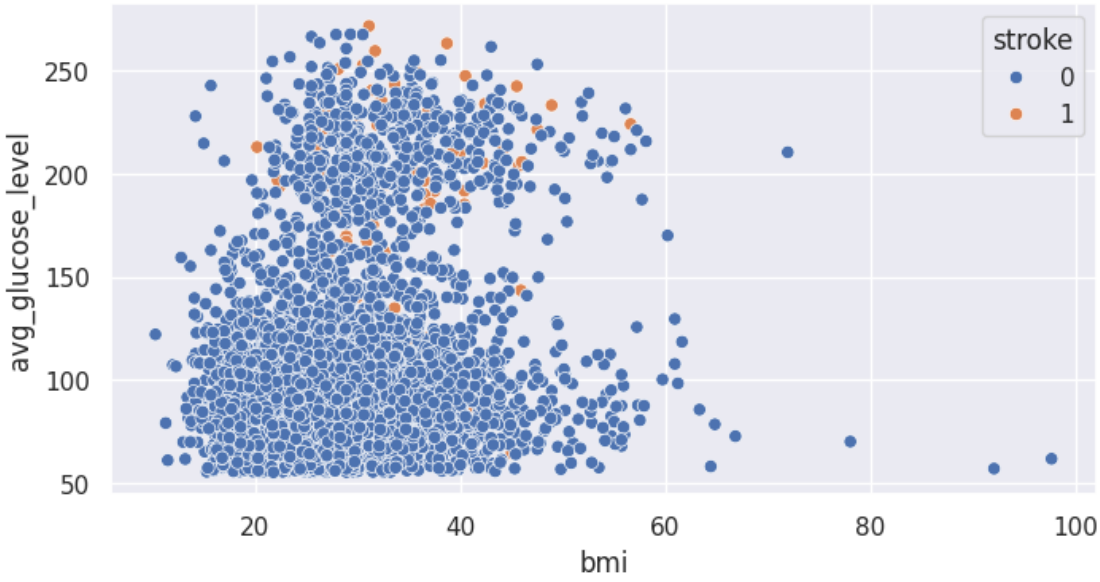
[ ]:
```
sns.scatterplot(data=df, x='age', y='avg_glucose_level', hue='stroke')
```

[ ]:
```
<Axes: xlabel='age', ylabel='avg_glucose_level'>
```



[ ]:
```
sns.scatterplot(data=df, x='bmi', y='avg_glucose_level', hue='stroke')
```

[ ]:
```
<Axes: xlabel='bmi', ylabel='avg_glucose_level'>
```



1.5 Encoding

1.5.1 Columns with two categorical values. Replace column values with 0 and 1.

[ ]:
```
def dummyreplacewith1n0(df,col, value1, value2):
  df[col].replace({value1: 1, value2: 0},inplace=True)
```

[ ]:
```
dummyreplacewith1n0(df,'gender', 'Female', 'Male')
dummyreplacewith1n0(df,'ever_married', 'Yes', 'No')
dummyreplacewith1n0(df,'Residence_type', 'Urban', 'Rural')
```

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

[ ]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	0	67.0	0	1	1	Private	1	228.69	36.600000	formerly smoked	1
1	1	61.0	0	0	1	Self-employed	0	202.21	28.893237	never smoked	1
2	0	80.0	0	1	1	Private	0	105.92	32.500000	never smoked	1
3	1	49.0	0	0	1	Private	1	171.23	34.400000	smokes	1
4	1	79.0	1	0	1	Self-employed	0	174.12	24.000000	never smoked	1

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

```
<class 'pandas.core.frame.DataFrame'>
Index: 5109 entries, 0 to 5109
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   gender             5109 non-null   int64
 1   age                5109 non-null   float64
 2   hypertension       5109 non-null   int64
 3   heart_disease      5109 non-null   int64
 4   ever_married       5109 non-null   int64
 5   work_type          5109 non-null   object
 6   Residence_type     5109 non-null   int64
 7   avg_glucose_level  5109 non-null   float64
 8   bmi                5109 non-null   float64
 9   smoking_status     5109 non-null   object
 10  stroke             5109 non-null   int64
dtypes: float64(3), int64(6), object(2)
memory usage: 479.0+ KB
```

1.5.2 Onehotencode with pd.get_dummies categorical columns with more than two values

[ ]:
```
df=pd.get_dummies(df,columns=['work_type', 'smoking_status'], prefix=['work',␣
↪'smoke'], dtype=int, drop_first=True)
df
```

[ ]:

	gender	age	hypertension	heart_disease	ever_married	Residence_type	\
0	0	67.0	0	1	1	1	
1	1	61.0	0	0	1	0	
2	0	80.0	0	1	1	0	
3	1	49.0	0	0	1	1	
4	1	79.0	1	0	1	0	
…	…	…	…	…	…	…	…
5105	1	80.0	1	0	1	1	
5106	1	81.0	0	0	1	1	
5107	1	35.0	0	0	1	0	
5108	0	51.0	0	0	1	0	
5109	1	44.0	0	0	1	1	

	avg_glucose_level	bmi	stroke	work_Never_worked	work_Private	\
0	228.69	36.600000	1	0	1	
1	202.21	28.893237	1	0	0	
2	105.92	32.500000	1	0	1	
3	171.23	34.400000	1	0	1	
4	174.12	24.000000	1	0	0	

```
…                 …        …     …                 …            …
5105              83.75  28.893237       0                 0            1
5106             125.20  40.000000       0                 0            0
5107              82.99  30.600000       0                 0            0
5108             166.29  25.600000       0                 0            1
5109              85.28  26.200000       0                 0            0

      work_Self-employed  work_children  smoke_formerly smoked  \
0                      0              0                      1
1                      1              0                      0
2                      0              0                      0
3                      0              0                      0
4                      1              0                      0
…                    …              …                    …
5105                   0              0                      0
5106                   1              0                      0
5107                   1              0                      0
5108                   0              0                      1
5109                   0              0                      0

      smoke_never smoked  smoke_smokes
0                      0             0
1                      1             0
2                      1             0
3                      0             1
4                      1             0
…                    …           …
5105                   1             0
5106                   1             0
5107                   1             0
5108                   0             0
5109                   0             0

[5109 rows x 16 columns]
```

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

```
<class 'pandas.core.frame.DataFrame'>
Index: 5109 entries, 0 to 5109
Data columns (total 16 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   gender                5109 non-null   int64
 1   age                   5109 non-null   float64
 2   hypertension          5109 non-null   int64
 3   heart_disease         5109 non-null   int64
 4   ever_married          5109 non-null   int64
```

```
 5   Residence_type          5109 non-null   int64
 6   avg_glucose_level       5109 non-null   float64
 7   bmi                     5109 non-null   float64
 8   stroke                  5109 non-null   int64
 9   work_Never_worked       5109 non-null   int64
 10  work_Private            5109 non-null   int64
 11  work_Self-employed      5109 non-null   int64
 12  work_children           5109 non-null   int64
 13  smoke_formerly smoked   5109 non-null   int64
 14  smoke_never smoked      5109 non-null   int64
 15  smoke_smokes            5109 non-null   int64
dtypes: float64(3), int64(13)
memory usage: 678.5 KB
```

[ ]:
```
# Move Target column (stroke) to the end of the dataframe
df = df.reindex(columns = [col for col in df.columns if col != 'stroke'] +␣
 ↪['stroke'])
df
```

[ ]:

	gender	age	hypertension	heart_disease	ever_married	Residence_type	\
0	0	67.0	0	1	1	1	
1	1	61.0	0	0	1	0	
2	0	80.0	0	1	1	0	
3	1	49.0	0	0	1	1	
4	1	79.0	1	0	1	0	
…	…	…	…	…	…	…	
5105	1	80.0	1	0	1	1	
5106	1	81.0	0	0	1	1	
5107	1	35.0	0	0	1	0	
5108	0	51.0	0	0	1	0	
5109	1	44.0	0	0	1	1	

	avg_glucose_level	bmi	work_Never_worked	work_Private	\
0	228.69	36.600000	0	1	
1	202.21	28.893237	0	0	
2	105.92	32.500000	0	1	
3	171.23	34.400000	0	1	
4	174.12	24.000000	0	0	
…	…	…	…	…	
5105	83.75	28.893237	0	1	
5106	125.20	40.000000	0	0	
5107	82.99	30.600000	0	0	
5108	166.29	25.600000	0	1	
5109	85.28	26.200000	0	0	

	work_Self-employed	work_children	smoke_formerly smoked	\
0	0	0	1	

```
1                       1              0                       0
2                       0              0                       0
3                       0              0                       0
4                       1              0                       0
…                     …              …                       …
5105                    0              0                       0
5106                    1              0                       0
5107                    1              0                       0
5108                    0              0                       1
5109                    0              0                       0

      smoke_never smoked  smoke_smokes  stroke
0                       0             0       1
1                       1             0       1
2                       1             0       1
3                       0             1       1
4                       1             0       1
…                     …             …      …
5105                    1             0       0
5106                    1             0       0
5107                    1             0       0
5108                    0             0       0
5109                    0             0       0

[5109 rows x 16 columns]
```

[ ]:
```
df[df['stroke']==1].sort_values(by='age')
```

[ ]:
```
     gender    age  hypertension  heart_disease  ever_married  Residence_type  \
162       1   1.32             0              0             0               1
245       1  14.00             0              0             0               0
182       1  32.00             0              0             1               0
118       1  38.00             0              0             0               1
133       1  38.00             0              0             1               0
..      …    …            …              …             …               …
42        0  82.00             0              1             1               1
56        1  82.00             0              0             1               0
188       0  82.00             0              0             1               0
23        0  82.00             0              1             1               0
35        1  82.00             1              1             0               0

     avg_glucose_level        bmi  work_Never_worked  work_Private  \
162              70.37  28.893237                  0             0
245              57.93  30.900000                  0             0
182              76.13  29.900000                  0             1
118              82.28  24.000000                  0             0
133             101.45  28.893237                  0             1
```

```
..                  …        …                  …            …
42              144.90  26.400000                  0            1
56               59.32  33.200000                  0            1
188              86.62  29.500000                  0            1
23              208.30  32.500000                  0            1
35               84.03  26.500000                  0            1

     work_Self-employed  work_children  smoke_formerly smoked  \
162                   0              1                      0
245                   0              1                      0
182                   0              0                      0
118                   1              0                      1
133                   0              0                      1
..                  …            …                      …
42                    0              0                      0
56                    0              0                      0
188                   0              0                      1
23                    0              0                      0
35                    0              0                      1

     smoke_never smoked  smoke_smokes  stroke
162                   0             0       1
245                   0             0       1
182                   0             1       1
118                   0             0       1
133                   0             0       1
..                  …           …     …
42                    0             1       1
56                    1             0       1
188                   0             0       1
23                    0             0       1
35                    0             0       1

[249 rows x 16 columns]
```

1.6 Split into Features and Target

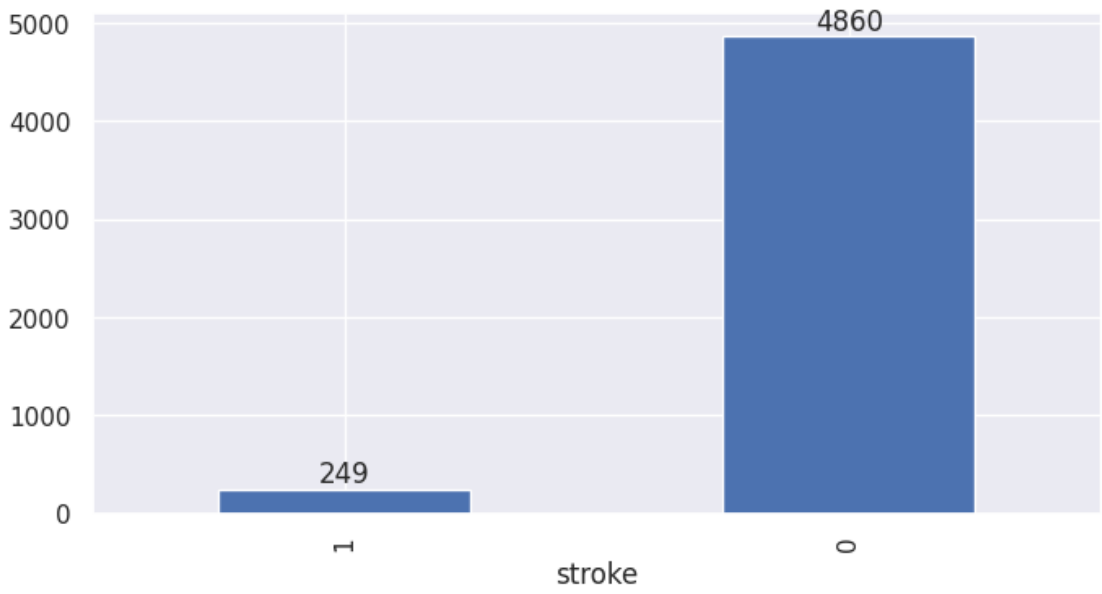
[ ]:
```
X = df.drop('stroke',axis=1)
y = df['stroke']
```

[ ]:
```
print(X.shape, y.shape)
```

```
(5109, 15) (5109,)
```

1.7 Check Target for Imbalance

```
[ ]: ax=df.stroke.value_counts().sort_values(ascending=True).plot(kind="bar")
     ax.bar_label(ax.containers[0])
     plt.show()
```




```
[ ]: from imblearn.over_sampling import RandomOverSampler
     ros = RandomOverSampler()
     X_res,y_res = ros.fit_resample(X,y)

     #Before and after oversampling counts
     from collections import Counter
     print('Original dataset shape {}'. format(Counter(y)))
     print('Resampled dataset shape {}'. format(Counter(y_res)))
```

```
Original dataset shape Counter({0: 4860, 1: 249})
Resampled dataset shape Counter({1: 4860, 0: 4860})
```

Used RandomOverSampler to rebalance the class distribution for the target “stroke”.

1.8 Split into Train and Test

```
[ ]: # split into train and test datasets
     from sklearn.model_selection import train_test_split

     X_train, X_test, y_train, y_test = train_test_split(X_res,y_res,test_size = 0.
      ↪33)
```

```
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(6512, 15) (3208, 15) (6512,) (3208,)
```

1.9 Scale Data

[ ]:
```
# Scaling the data
#from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import RobustScaler
#scaler = StandardScaler()
scaler = RobustScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

1.10 Best Parameters

[ ]:
```
#from sklearn.model_selection import GridSearchCV

# Define the parameter grid
#param_grid = {'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.001]}

# Create a grid search object
#grid = GridSearchCV(SVC(), param_grid, refit=True, verbose=2)

# Fit the grid search object to the training data
#grid.fit(X_train_scaled, y_train)

# Get the best parameters
#best_params = grid.best_params_

#print(best_params)
#{'C': 100, 'gamma': 1}
```

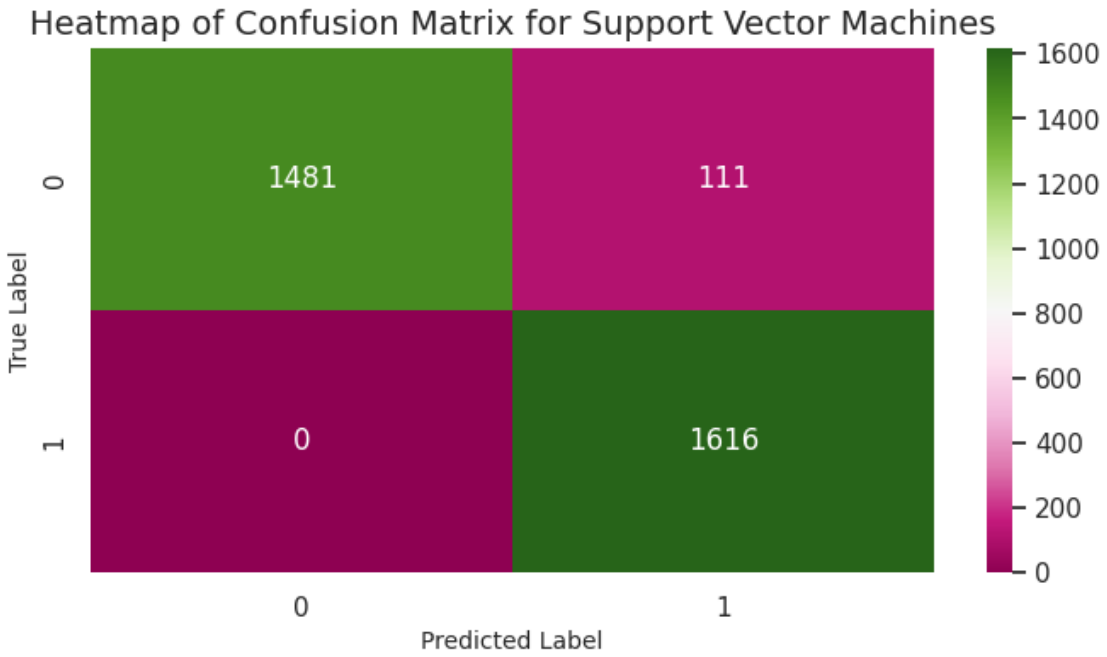
1.11 Model

[ ]:
```
from sklearn.svm import SVC

#define model
model_svm = SVC(kernel = 'rbf', C=100, gamma = 1, random_state=42)

#fit
model_svm.fit(X_train_scaled, y_train)

#predict
svm_pred = model_svm.predict(X_test_scaled)
```

1.12 Metrics

[ ]:
```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

print("Classification Report for Support Vector Machines")
print(classification_report(y_test,svm_pred))

#classes = ['No Stroke', 'Stroke']

#sns.heatmap(confusion_matrix(y_test,svm_pred), annot=True,␣
 ↪fmt="d",cmap="PiYG",xticklabels=classes, yticklabels=classes)

sns.heatmap(confusion_matrix(y_test,svm_pred), annot=True, fmt="d",cmap="PiYG")

plt.title('Heatmap of Confusion Matrix for Support Vector Machines', fontsize␣
 ↪=14) # title with fontsize 20

plt.xlabel('Predicted Label', fontsize = 10) # x-axis label with fontsize 15
plt.ylabel('True Label', fontsize = 10) # y-axis label with fontsize 15
plt.show()
```

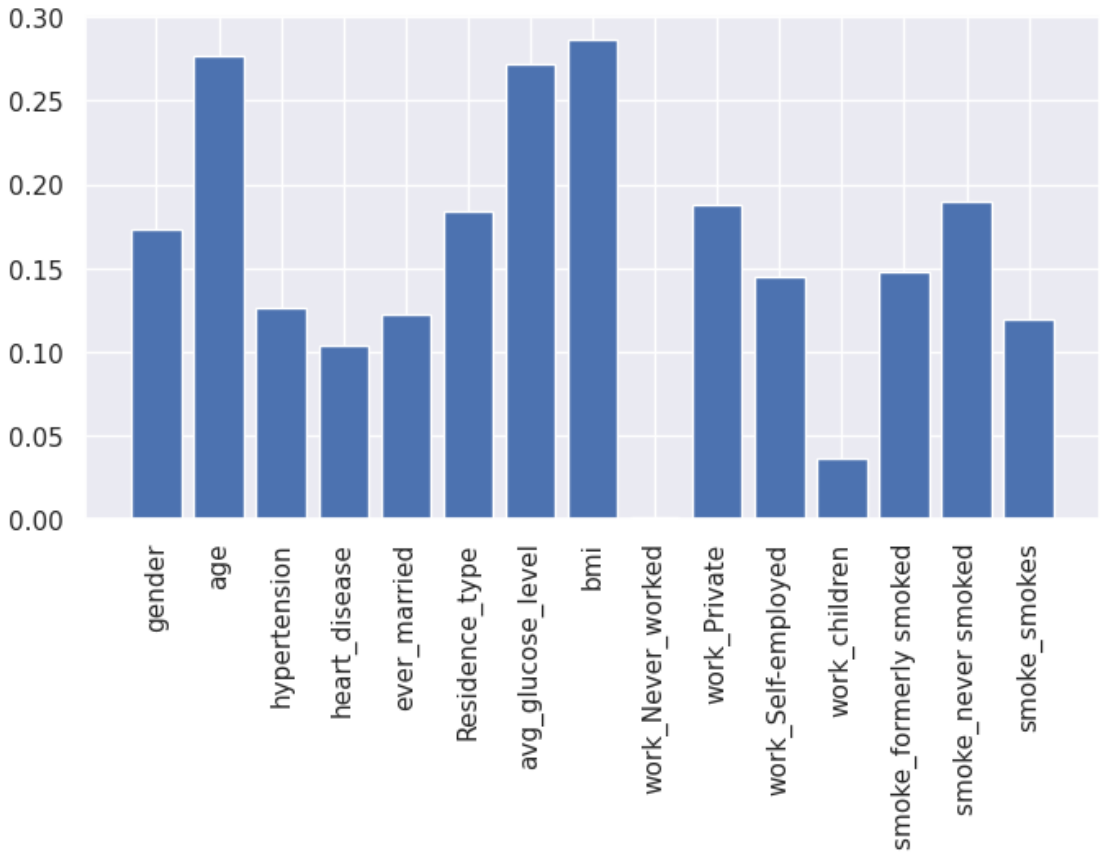
```
Classification Report for Support Vector Machines
              precision    recall  f1-score   support

           0       1.00      0.93      0.96      1592
           1       0.94      1.00      0.97      1616

    accuracy                           0.97      3208
   macro avg       0.97      0.97      0.97      3208
weighted avg       0.97      0.97      0.97      3208
```



1.13 Feature Importance

[ ]:
```
from sklearn.inspection import permutation_importance
 feature_importances = permutation_importance(
 model_svm, X_test_scaled, y_test, n_repeats=10, random_state=42 )
```

[ ]:
```
import matplotlib.pyplot as plt
 features = X_train.columns
 importances = feature_importances.importances_mean
 plt.bar(features, importances)
 plt.xticks(rotation=90)
 plt.show()
```



1.14 Findings

- Age range for stroke patients is 40 - 80
- Average glucose level for stroke patients is above 100 with more activity above 150
- BMI for stroke patients is 20 - 60
- Best parameters for SVM model: {'C': 100, 'gamma': 1}
- The SVM model accuracy is 97%.
- Based on model results the most important features are age, bmi, and average glucose. The secondary features are gender, residence (urban or rural), work is private, and never smoked.

[ ]:

1.15 Limitations

There is a great imbalance between the number of stroke and non-stroke target values. The data contained 249 observations for stroke and 4860 for non-stroke. Imbalance over sampling was used to achieve equal number of stroke and non-stroke target values.

[ ]: