Heart Stroke

October 30, 2023

[362]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from numpy import mean
from numpy import std
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import MinMaxScaler
from sklearn.inspection import permutation_importance
from statsmodels.graphics.gofplots import qqplot
from scipy.stats import shapiro
```

[328]:
```
df = pd.read_csv(f'D:\machine learning(sharif)\my project/
 ↪healthcare-dataset-stroke-data.csv')
df.head()
```

[328]:
```
      id  gender   age  hypertension  heart_disease ever_married  \
0   9046    Male  67.0             0              1          Yes
1  51676  Female  61.0             0              0          Yes
2  31112    Male  80.0             0              1          Yes
3  60182  Female  49.0             0              0          Yes
4   1665  Female  79.0             1              0          Yes

       work_type Residence_type  avg_glucose_level   bmi   smoking_status  \
0        Private          Urban             228.69  36.6  formerly smoked
1  Self-employed          Rural             202.21   NaN     never smoked
2        Private          Rural             105.92  32.5     never smoked
3        Private          Urban             171.23  34.4           smokes
4  Self-employed          Rural             174.12  24.0     never smoked
```

```
   stroke
0       1
1       1
2       1
3       1
4       1
```

[329]:
```
df.drop('id', axis=1, inplace=True)
```

[330]:
```
df.head(5)
```

[330]:
```
   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  36.6  formerly smoked       1
1          Rural             202.21   NaN     never smoked       1
2          Rural             105.92  32.5     never smoked       1
3          Urban             171.23  34.4           smokes       1
4          Rural             174.12  24.0     never smoked       1
```

0.1 Data Preprocessing

[331]:
```
df.describe()
```

[331]:
```
               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  4909.000000  5110.000000
mean     28.893237     0.048728
std       7.854067     0.215320
min      10.300000     0.000000
25%      23.500000     0.000000
50%      28.100000     0.000000
75%      33.100000     0.000000
```

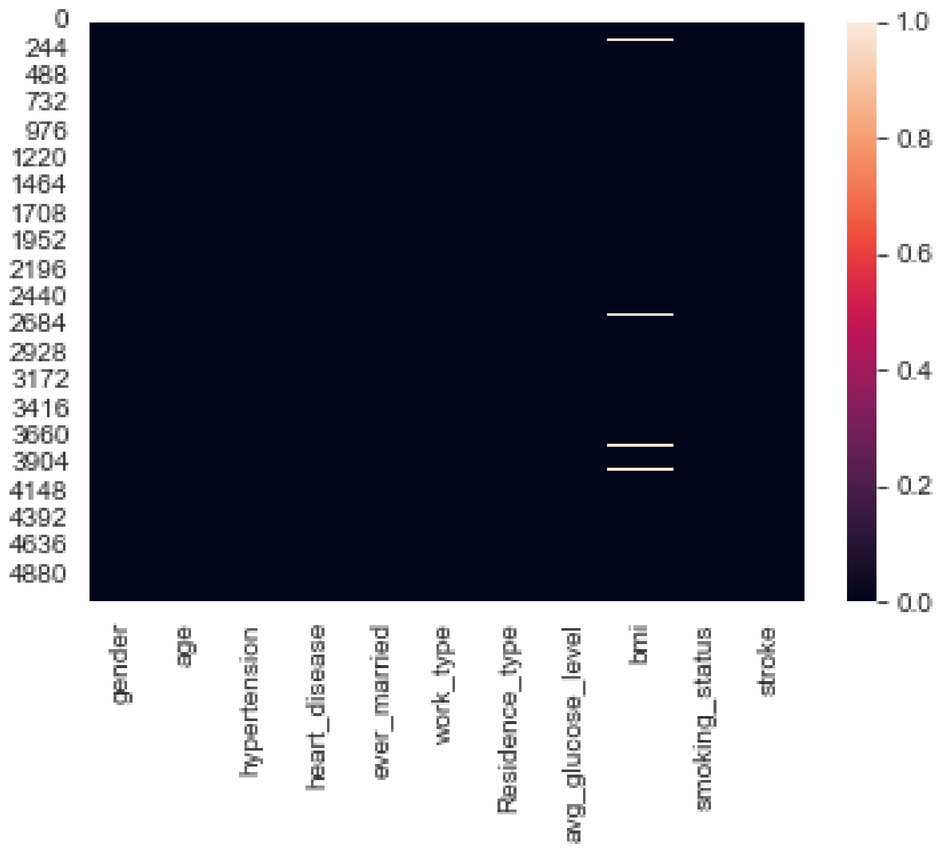
```
max      97.600000     1.000000
```

[332]:
```
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
```

[333]:
```
sns.heatmap(df.isnull())
```

[333]: <AxesSubplot:>



[334]:
```
df.isnull().sum()
```

[334]:
```
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
```

[335]:
```
df['bmi'].fillna(df['bmi'].mean(), inplace=True)
```

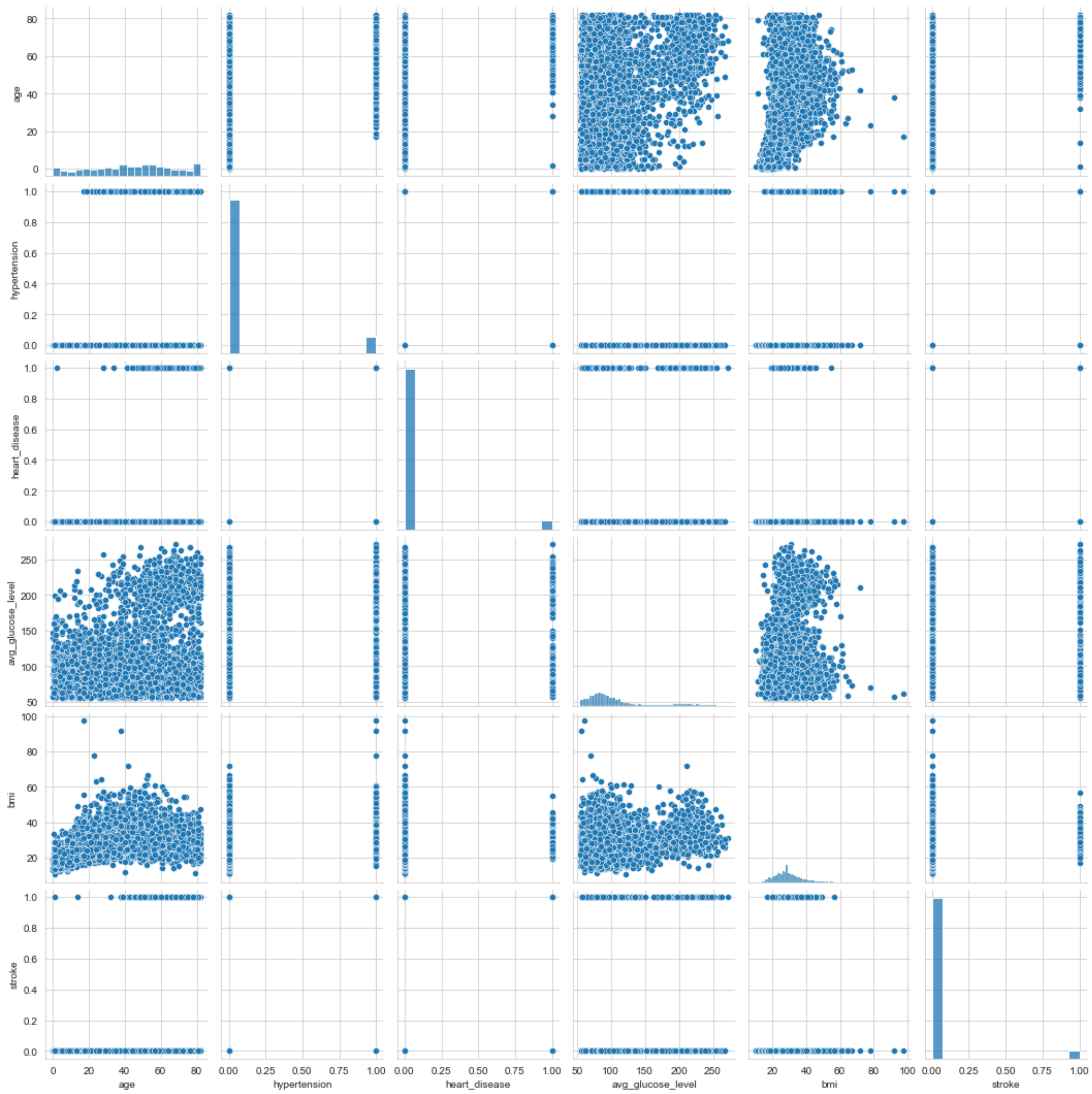
[336]:
```
df['bmi'].isnull().sum()
```

[336]:
```
0
```

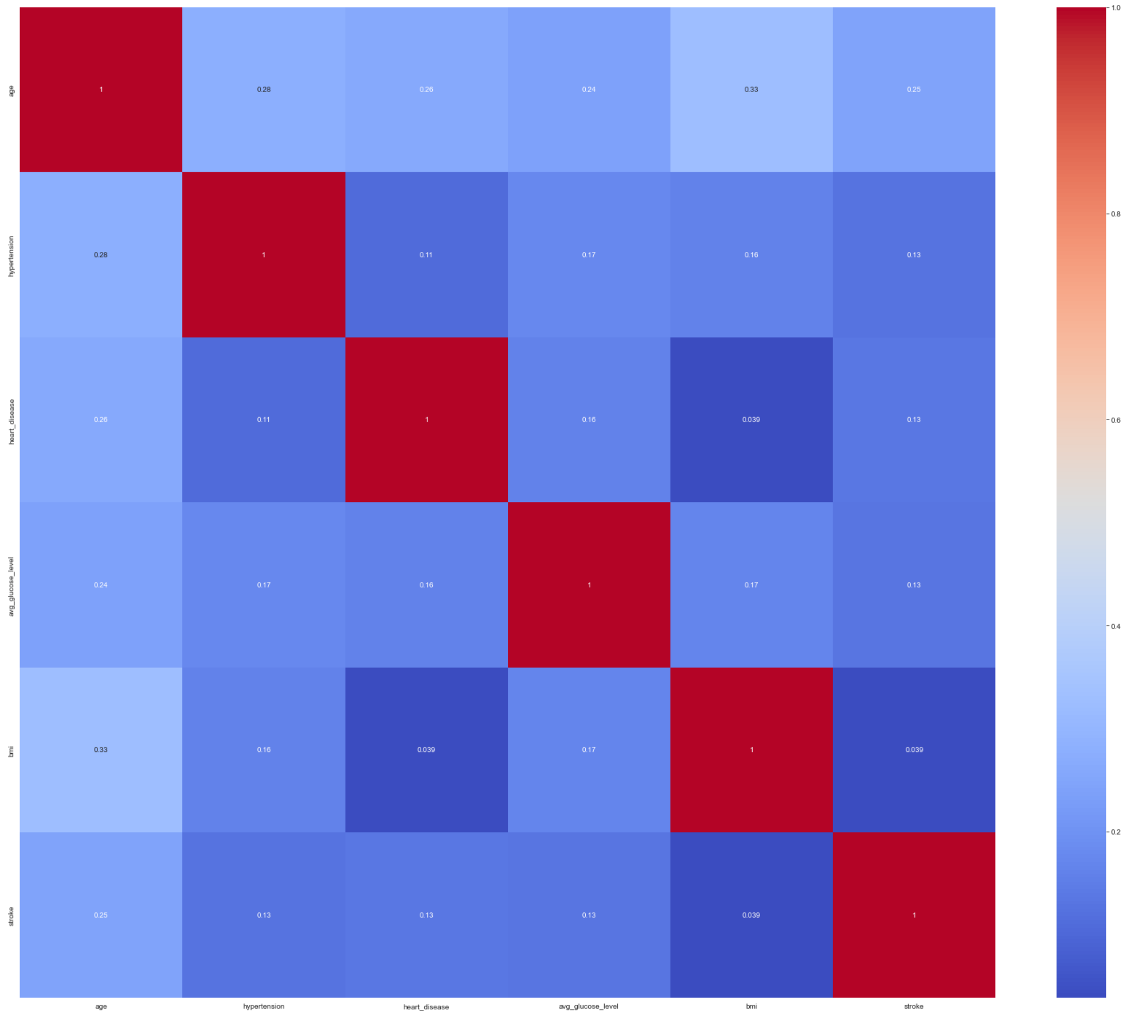
[337]:
```python
df.nunique()
```

[337]:
```
gender                  3
age                   104
hypertension            2
heart_disease           2
ever_married            2
work_type               5
Residence_type          2
avg_glucose_level    3979
bmi                   419
smoking_status          4
stroke                  2
dtype: int64
```

[338]:
```python
sns.set_style('whitegrid')
sns.pairplot(df)
plt.show()
```

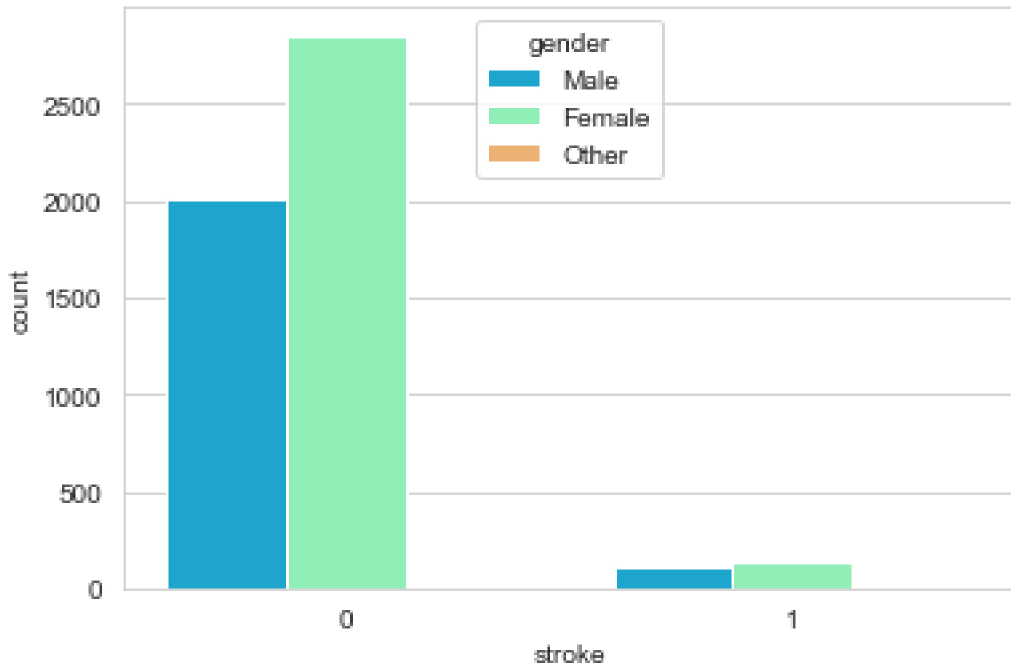


[339]:
```
plt.figure(figsize = (30, 25))
sns.heatmap(df.corr(), annot = True, cmap="coolwarm")
plt.show()
```



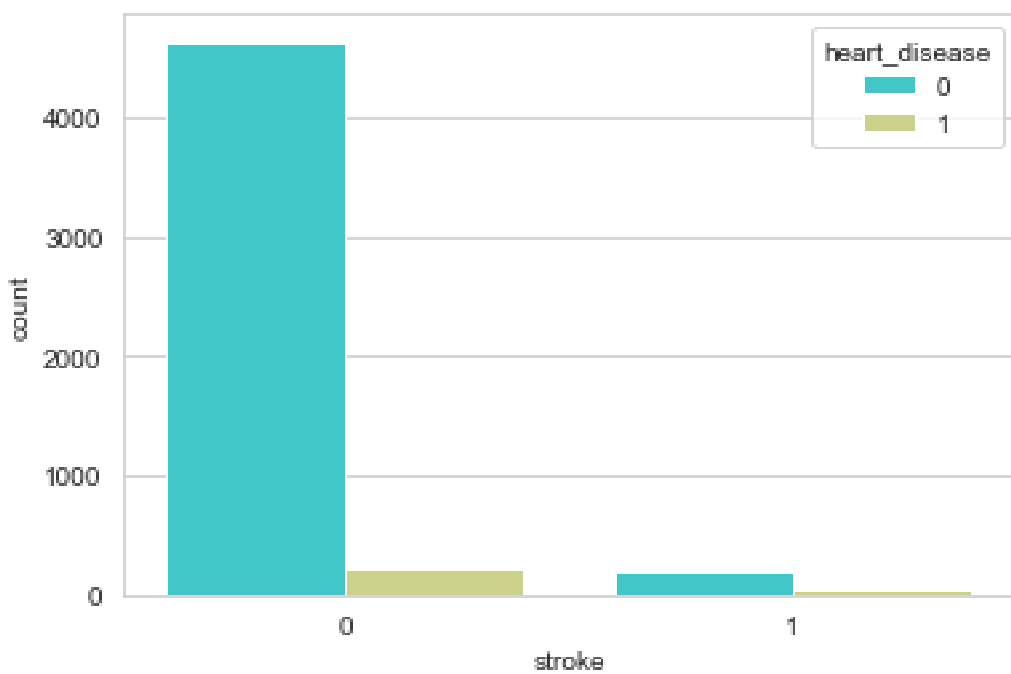
[340]:
```
sns.set_style('whitegrid')
sns.countplot(x='stroke',hue='gender',data=df,palette='rainbow')
```

[340]: <AxesSubplot:xlabel='stroke', ylabel='count'>



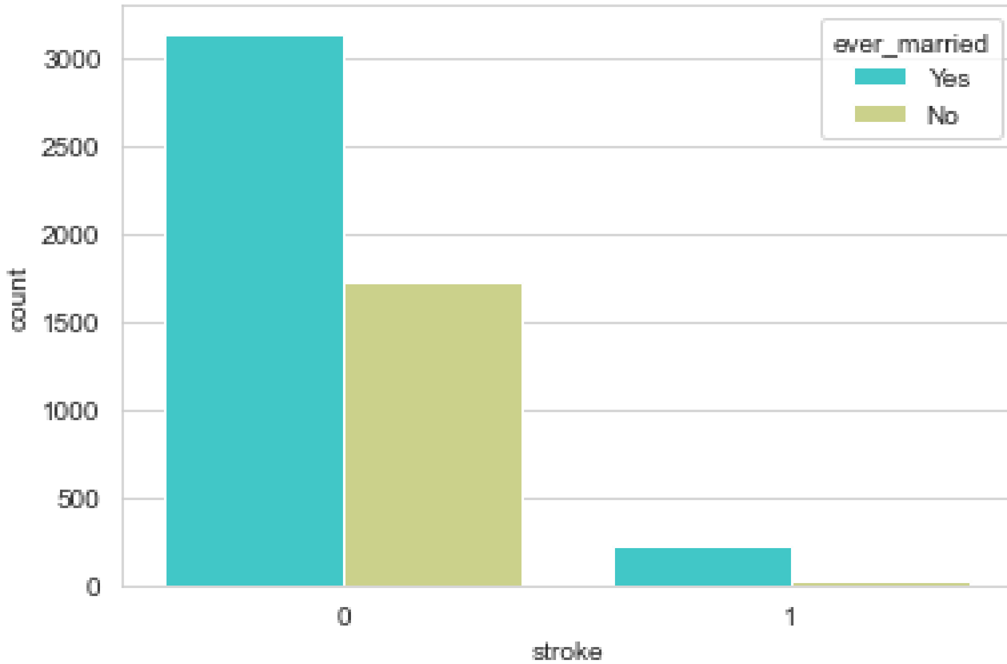
[341]:
```
sns.set_style('whitegrid')
sns.countplot(x='stroke',hue='heart_disease',data=df,palette='rainbow')
```

[341]: <AxesSubplot:xlabel='stroke', ylabel='count'>



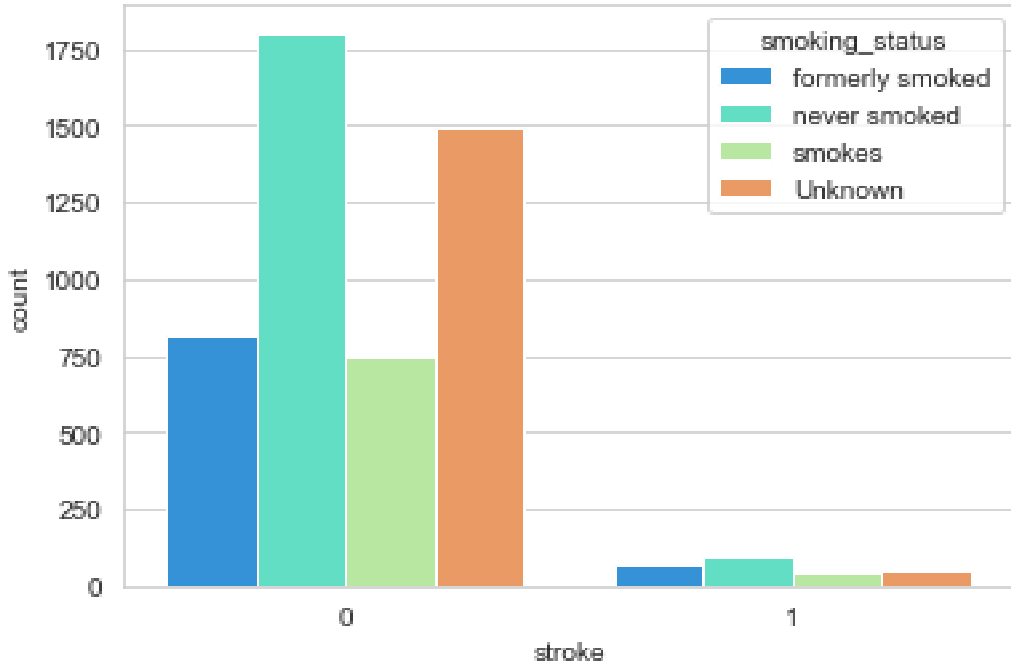
[342]:
```
sns.set_style('whitegrid')
sns.countplot(x='stroke',hue='ever_married',data=df,palette='rainbow')
```

[342]: <AxesSubplot:xlabel='stroke', ylabel='count'>



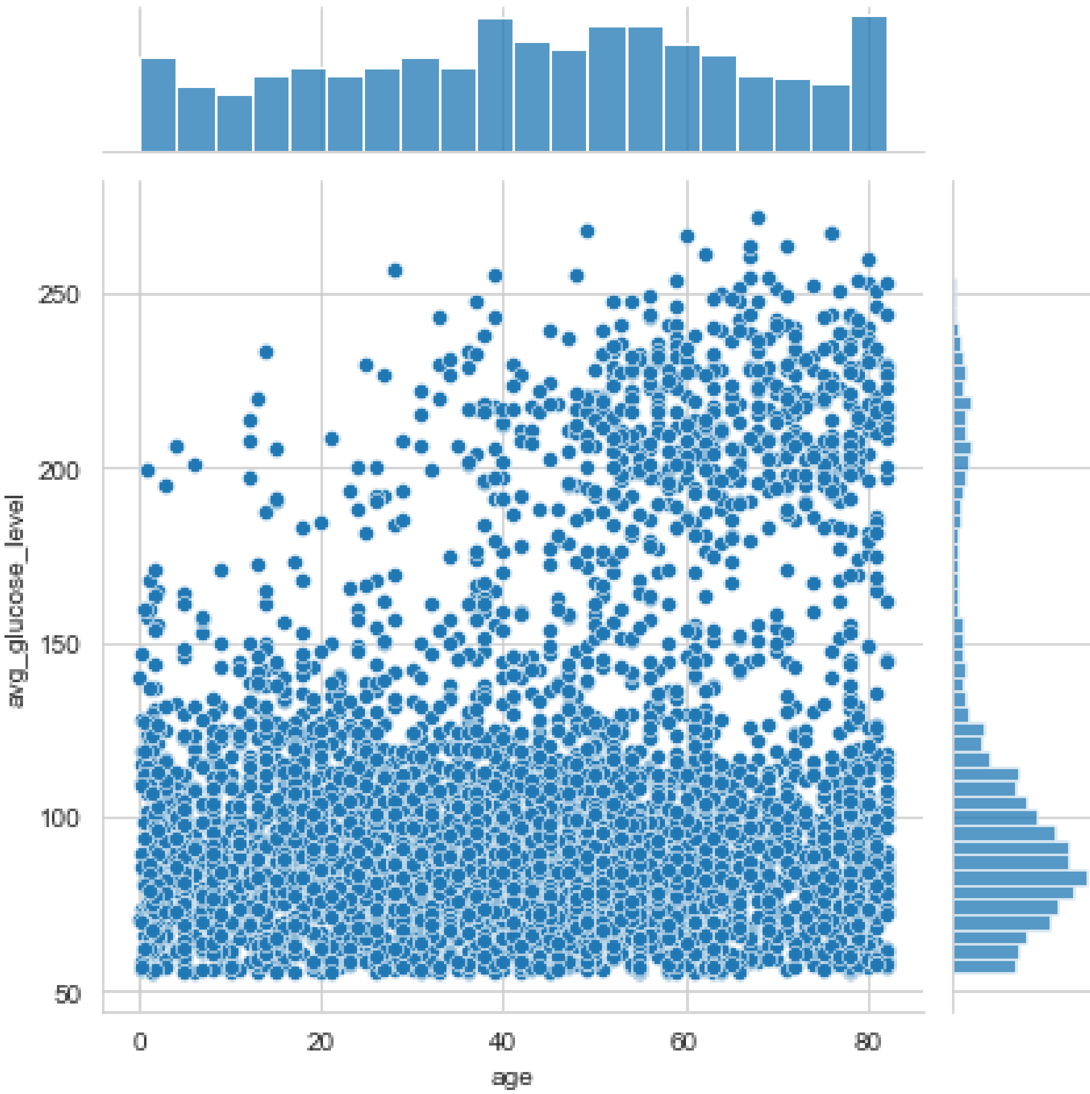
[343]:
```
sns.set_style('whitegrid')
sns.countplot(x='stroke',hue='smoking_status',data=df,palette='rainbow')
```

[343]: <AxesSubplot:xlabel='stroke', ylabel='count'>



[344]:
```
sns.jointplot(y='avg_glucose_level',x='age',data=df,kind='scatter')
```

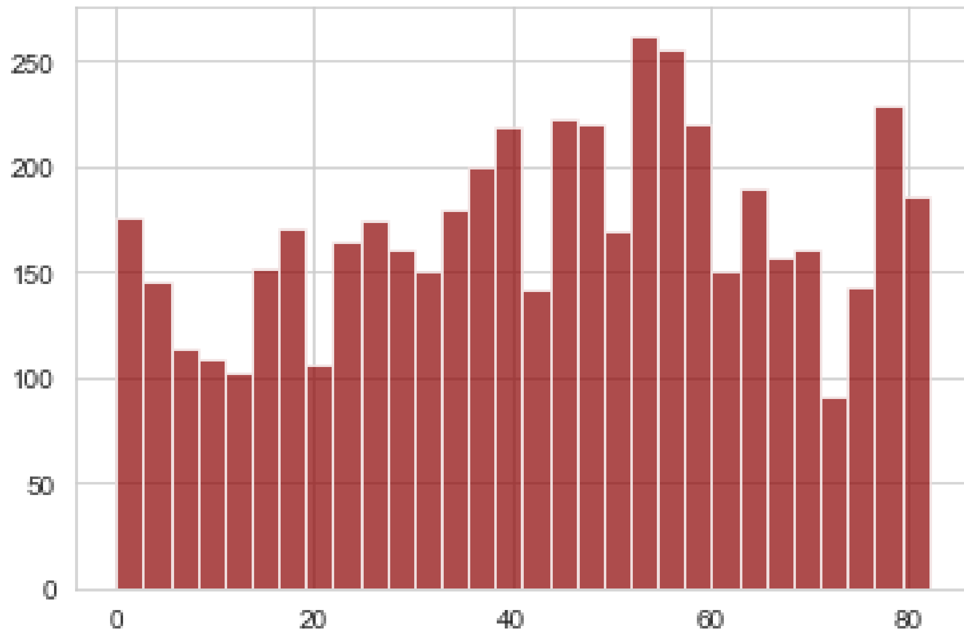
[344]: <seaborn.axisgrid.JointGrid at 0x226b71032e0>



0.2 Normality Test

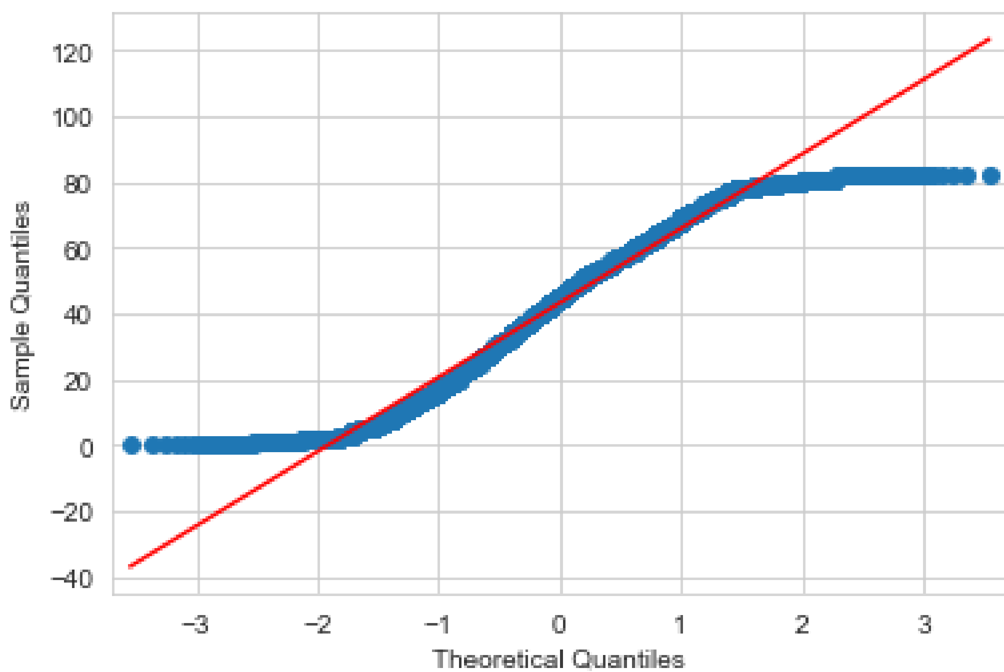
[345]:
```
df['age'].hist(bins=30,color='darkred',alpha=0.7)
```

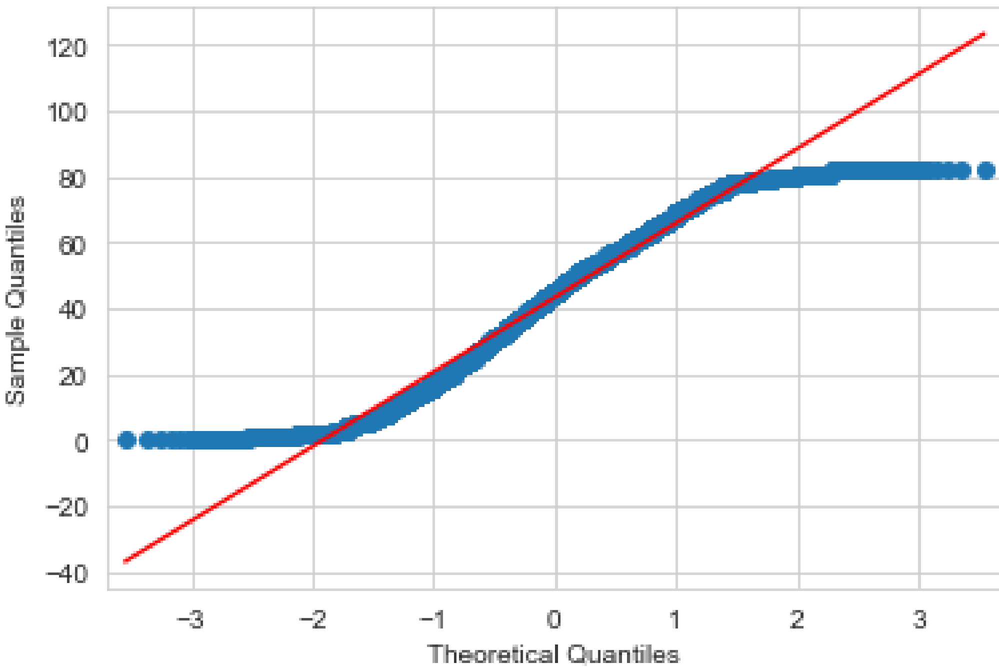
[345]: <AxesSubplot:>



[346]:
```
qqplot(df['age'],line='s')
```

[346]:





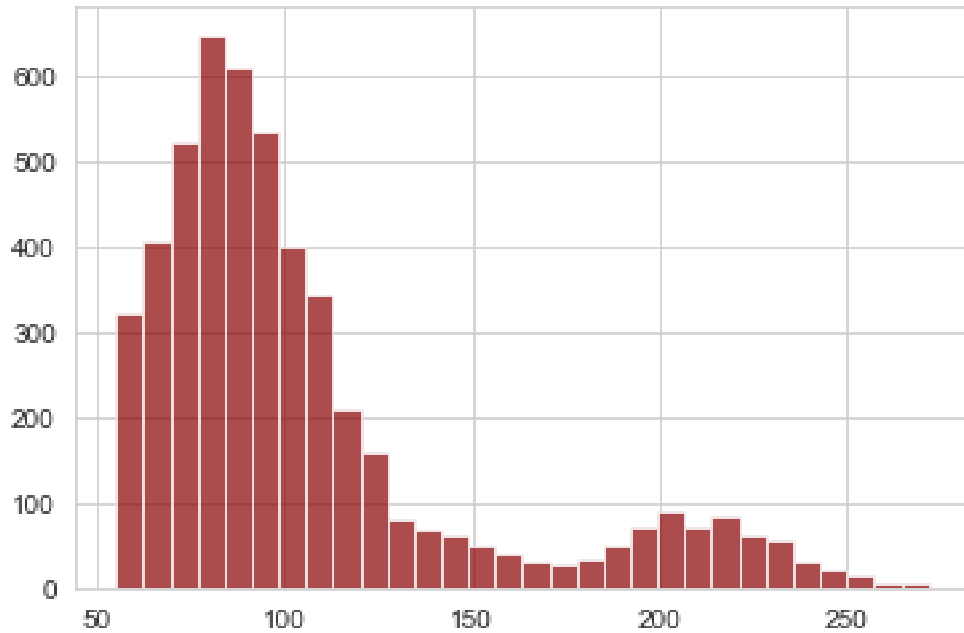
[347]:
```
Statistics, p = shapiro(df['age'])
print(f'Statistics={Statistics:0.3f} p_value={p:0.3f}')

alpha = 0.05

if p >= alpha:
    print('Sample looks Gaussian (fail to reject H0)')
else:
    print('Sample does not look Gaussian (reject H0)')
```

```
Statistics=0.967 p_value=0.000
Sample does not look Gaussian (reject H0)
```

```
C:\Users\USER\anaconda3\lib\site-packages\scipy\stats\morestats.py:1760:
UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
```

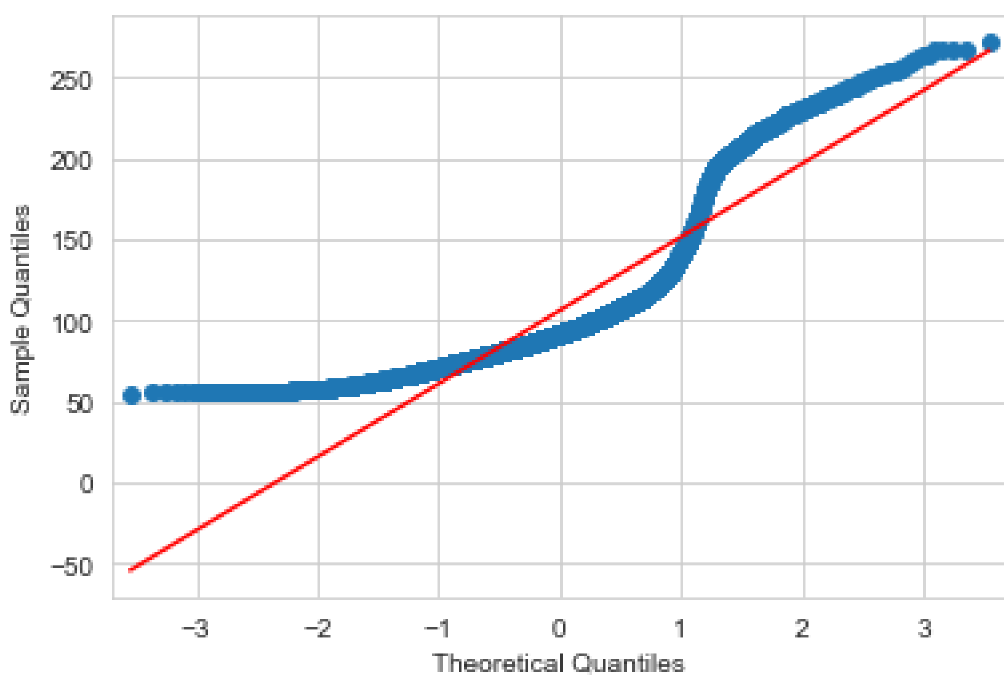
[348]:
```
df['avg_glucose_level'].hist(bins=30,color='darkred',alpha=0.7)
```

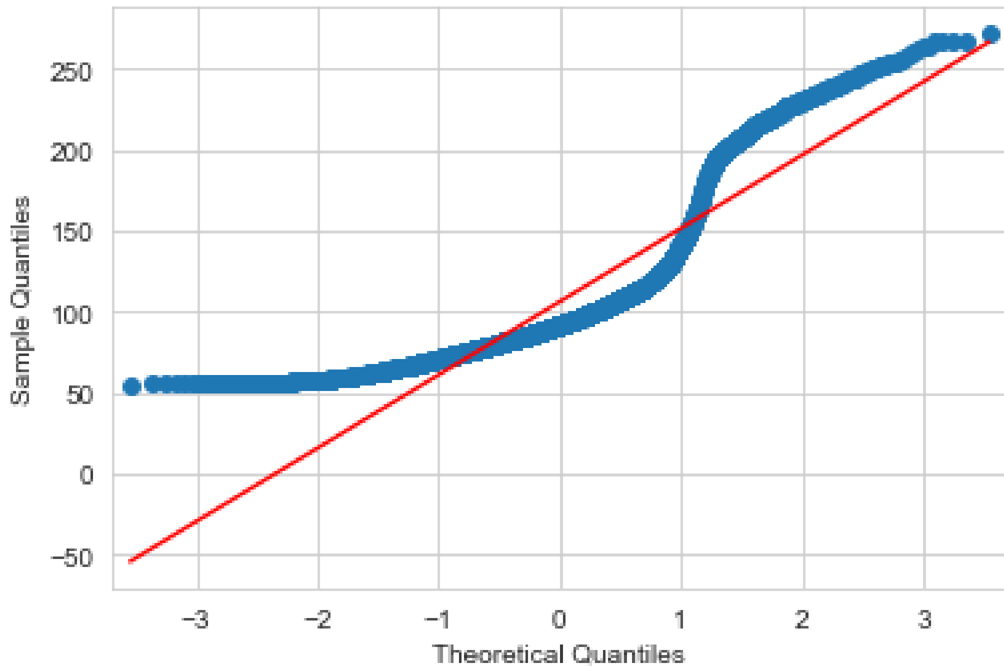
[348]: <AxesSubplot:>



[349]:
```
qqplot(df['avg_glucose_level'],line='s')
```

[349]:



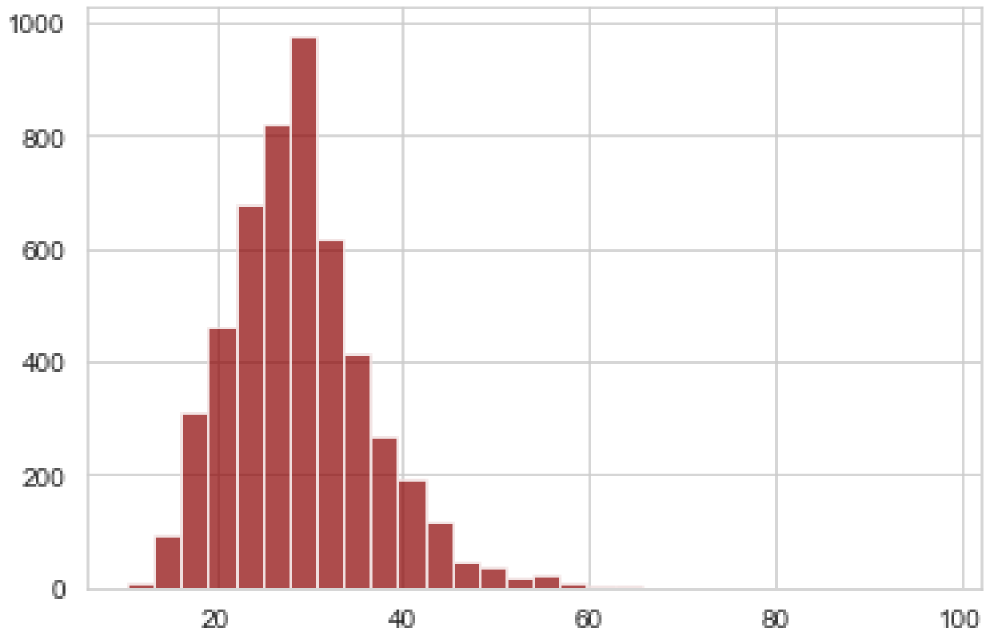


[350]:
```
Statistics, p = shapiro(df['avg_glucose_level'])
print(f'Statistics={Statistics:0.3f} p_value={p:0.3f}')

alpha = 0.05

if p >= alpha:
    print('Sample looks Gaussian (fail to reject H0)')
else:
    print('Sample does not look Gaussian (reject H0)')
```

```
Statistics=0.806 p_value=0.000
Sample does not look Gaussian (reject H0)

C:\Users\USER\anaconda3\lib\site-packages\scipy\stats\morestats.py:1760:
UserWarning: p-value may not be accurate for N > 5000.
  warnings.warn("p-value may not be accurate for N > 5000.")
```

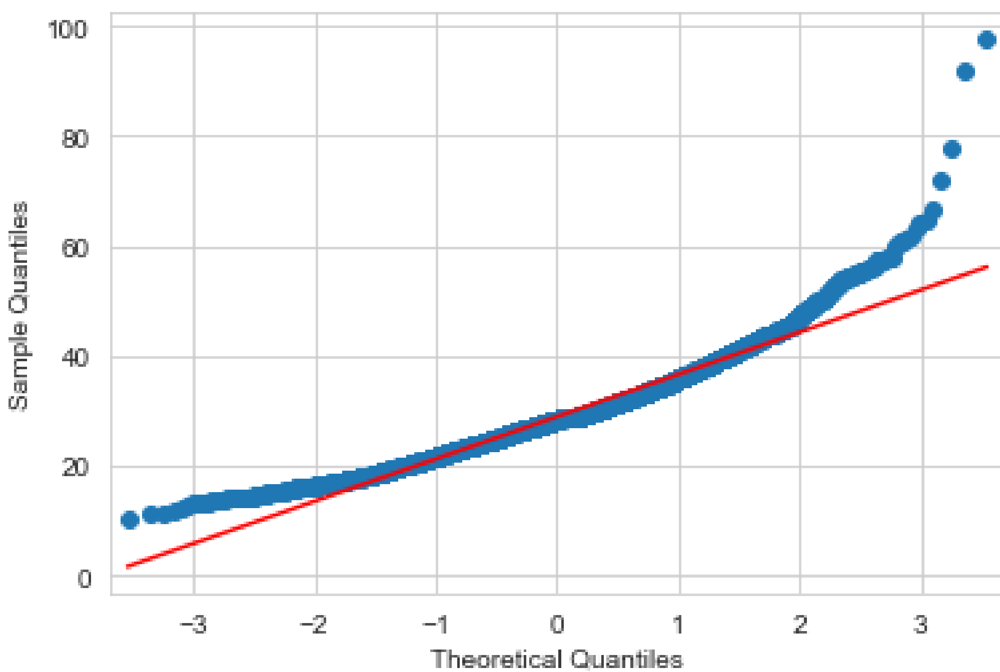
[351]:
```
df['bmi'].hist(bins=30,color='darkred',alpha=0.7)
```

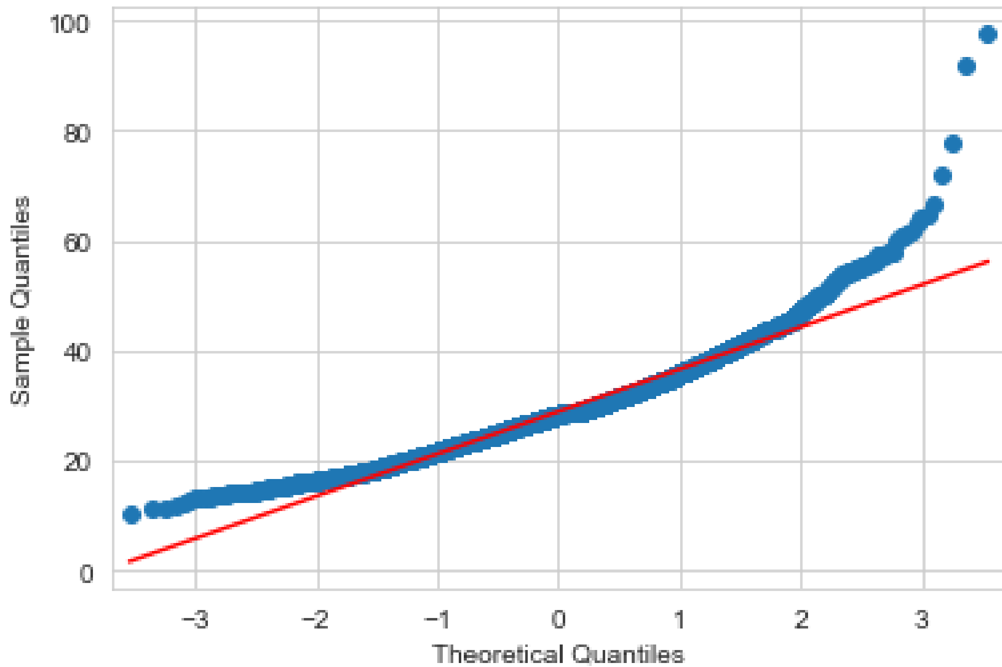
[351]: <AxesSubplot:>



[352]:
```
qqplot(df['bmi'],line='s')
```

[352]:





0.3 Data Transform

[353]:
```
df['ever_married'].replace({'Yes':1, 'No':0}, inplace=True)
df['gender'].replace({'Male':1, 'Female':0,'Other':2}, inplace=True)
df['Residence_type'].replace({'Urban':1, 'Rural':0}, inplace=True)
df['smoking_status'].replace({'formerly smoked':0, 'never smoked':1, 'smokes':
 ↪2, 'Unknown':3}, inplace=True)
df['work_type'].replace({'Private':0, 'Self-employed':1, 'children':2,␣
 ↪'Govt_job':3, 'Never_worked':4}, inplace=True)
```

[354]:
```
df.head()
```

[354]:
```
   gender   age  hypertension  heart_disease  ever_married  work_type  \
0       1  67.0             0              1             1          0
1       0  61.0             0              0             1          1
2       1  80.0             0              1             1          0
3       0  49.0             0              0             1          0
4       0  79.0             1              0             1          1

   Residence_type  avg_glucose_level        bmi  smoking_status  stroke
0               1             228.69  36.600000               0       1
1               0             202.21  28.893237               1       1
2               0             105.92  32.500000               1       1
3               1             171.23  34.400000               2       1
4               0             174.12  24.000000               1       1
```

[355]:
```python
X, y = df.drop('stroke', axis=1), df['stroke']
print(X.shape, y.shape)
numerical_ix = X.select_dtypes(include=['int64', 'float64', "float32"]).columns
print("numerical_ix: ",numerical_ix)
categorical_ix = X.select_dtypes(include=['object','bool']).columns
print("categorical_ix: ",categorical_ix)
t = [('cat', OneHotEncoder(), categorical_ix), ('num', MinMaxScaler(),␣
↪numerical_ix)]
col_transform = ColumnTransformer(transformers=t)
```

```
(5110, 10) (5110,)
numerical_ix:  Index(['gender', 'age', 'hypertension', 'heart_disease',
'ever_married',
       'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',
       'smoking_status'],
      dtype='object')
categorical_ix:  Index([], dtype='object')
```

0.4 Model Training and Testing

0.4.1 Logistic Regression

[356]:
```python
model=LogisticRegression()
cv = RepeatedStratifiedKFold(n_splits=10,n_repeats=5, random_state=1)
scores = cross_val_score(model, X, y, scoring='accuracy', cv=cv,n_jobs=-1)
Accuracy_lg=mean(scores)
print('Accuracy_lg: %.3f (%.3f)' % (mean(scores), std(scores)))
```

```
Accuracy_lg: 0.951 (0.002)
```

0.4.2 SVM

[357]:
```python
model=SVC()
pipeline = Pipeline(steps=[('prep',col_transform), ('m', model)])
cv = RepeatedStratifiedKFold(n_splits=10,n_repeats=5, random_state=1)
scores = cross_val_score(pipeline, X, y, scoring='accuracy', cv=cv,n_jobs=-1)
Accuracy_svm=mean(scores)
print('Accuracy_svm: %.3f (%.3f)' % (mean(scores), std(scores)))
```

```
Accuracy_svm: 0.951 (0.001)
```

0.4.3 K-Nearest Neighbors (KNN)

[358]:
```python
model=KNeighborsClassifier()
pipeline = Pipeline(steps=[('prep',col_transform), ('m', model)])
cv = RepeatedStratifiedKFold(n_splits=10,n_repeats=5, random_state=1)
scores = cross_val_score(pipeline, X, y, scoring='accuracy', cv=cv,n_jobs=-1)
Accuracy_knn=mean(scores)
print('Accuracy_knn: %.3f (%.3f)' % (mean(scores), std(scores)))
```

Accuracy_knn: 0.949 (0.004)

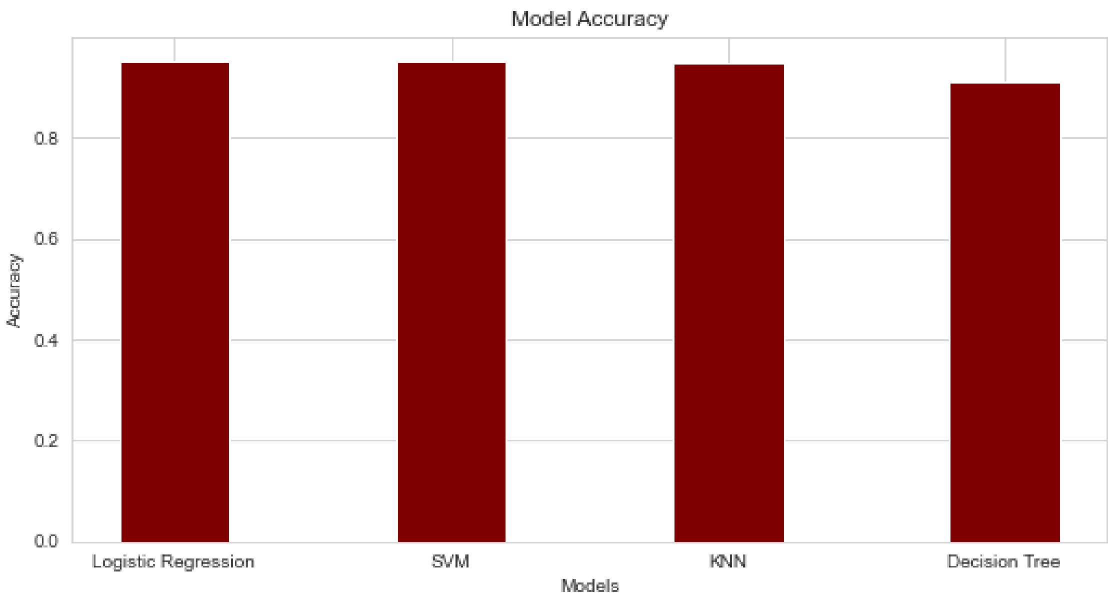
0.4.4 Decision Tree Classifier

[359]:
```
model=DecisionTreeClassifier()
pipeline = Pipeline(steps=[('prep',col_transform), ('m', model)])
cv = RepeatedStratifiedKFold(n_splits=10,n_repeats=5, random_state=1)
scores = cross_val_score(pipeline, X, y, scoring='accuracy', cv=cv,n_jobs=-1)
Accuracy_dt=mean(scores)
print('Accuracy__dt: %.3f (%.3f)' % (mean(scores), std(scores)))
```

Accuracy__dt: 0.911 (0.010)

0.5 Model Comparison

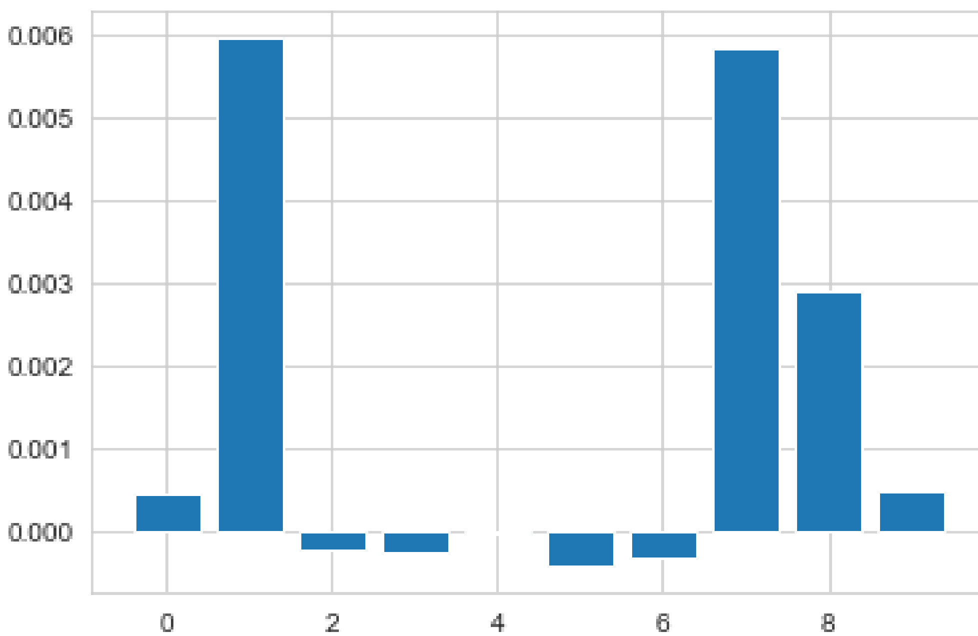
[360]:
```
models = ['Logistic Regression', 'SVM', 'KNN','Decision Tree']
accuracy = [Accuracy_lg, Accuracy_svm, Accuracy_knn,Accuracy_dt]
plt.figure(figsize=(10,5))
plt.bar(models, accuracy, color = 'Maroon', width = 0.4)
plt.xlabel('Models')
plt.ylabel('Accuracy')
plt.title('Model Accuracy')
plt.show()
```



0.5.1 Permutation Feature Importance

[361]:
```
model = KNeighborsClassifier()
model.fit(X, y)
results = permutation_importance(model, X, y, scoring='accuracy')
importance = results.importances_mean
for i,v in enumerate(importance):
    print('Feature: %0d, Score: %.5f' % (i,v))
plt.bar([x for x in range(len(importance))], importance)
plt.show()
```

```
Feature: 0, Score: 0.00043
Feature: 1, Score: 0.00595
Feature: 2, Score: -0.00023
Feature: 3, Score: -0.00027
Feature: 4, Score: -0.00004
Feature: 5, Score: -0.00043
Feature: 6, Score: -0.00035
Feature: 7, Score: 0.00583
Feature: 8, Score: 0.00290
Feature: 9, Score: 0.00047
```



0.6 Conclusion

The model accuracies of Logistic Regression and SVM are quite similar 95.1 %. The accuracy of KNN and Decision Tree Classifier are 94.9 % and 91 % So, we can use any of these models to predict the heart stroke.

The relationship of different features was depicted, but at the end, based on the KNN model, the characteristics that had the greatest impact on the prediction were identified. These features are respectively: 1-age 2-avg_glucose_level 3-bmi

```
[ ]:
```