stroke-prediction

February 3, 2024

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:
```
df = pd.read_csv('healthcare-dataset-stroke-data.csv')
```

[3]:
```
df.head()
```

[3]:
```
      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
```

[5]:
```
df.columns
```

[5]:
```
Index(['id', 'gender', 'age', 'hypertension', 'heart_disease', 'ever_married',
       'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',
       'smoking_status', 'stroke'],
      dtype='object')
```

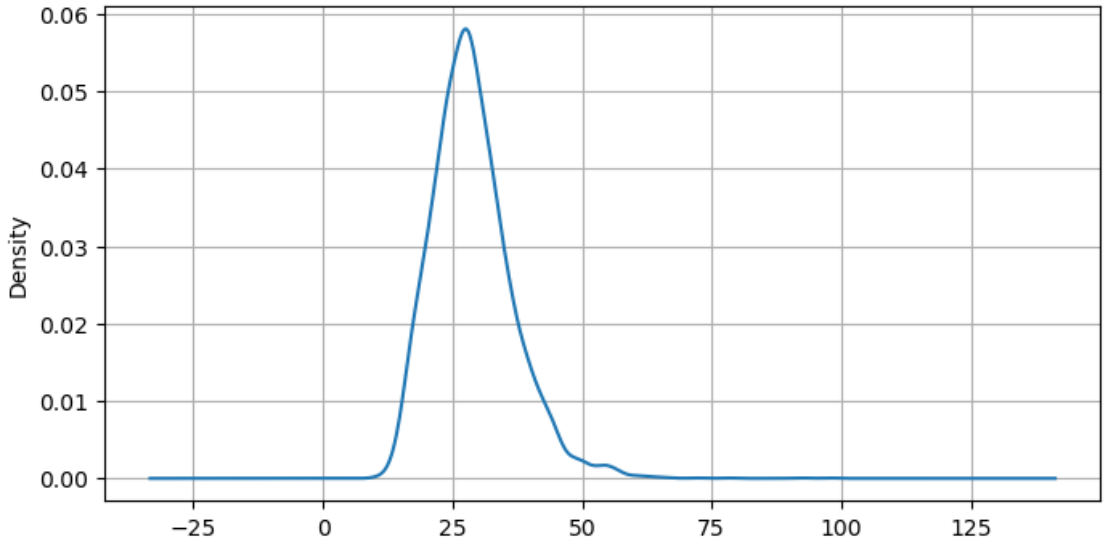
[4]:
```
df.info()
```

```
<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
```

[5]:
```
df.isnull().sum()
```

[5]:
```
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
```

[6]:
```
plt.figure(figsize=(8,4))
df['bmi'].plot(kind='kde')
plt.grid()
```



[7]:
```
df['bmi'].fillna(df['bmi'].mean(), inplace=True)
```

[8]:
```
df.isnull().sum()
```

[8]:
```
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
```

[9]:
```
df.shape
```

[9]:
```
(5110, 12)
```

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

[10]:

	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  \
0        Private          Urban             228.69  36.600000
1  Self-employed          Rural             202.21  28.893237
2        Private          Rural             105.92  32.500000
3        Private          Urban             171.23  34.400000
4  Self-employed          Rural             174.12  24.000000

    smoking_status  stroke
0  formerly smoked       1
1     never smoked       1
2     never smoked       1
3           smokes       1
4     never smoked       1
```

[11]:
```
df.drop('id', axis=1, inplace=True)
df.head()
```

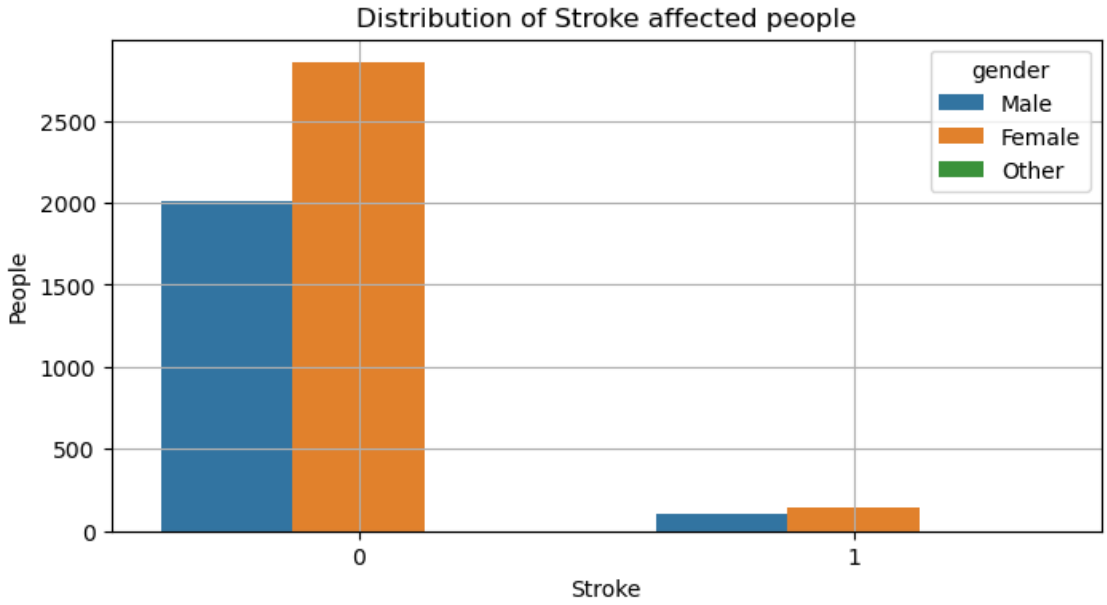
[11]:
```
   gender   age  hypertension  heart_disease ever_married      work_type  \
0    Male  67.0             0              1          Yes        Private
1  Female  61.0             0              0          Yes  Self-employed
2    Male  80.0             0              1          Yes        Private
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
```

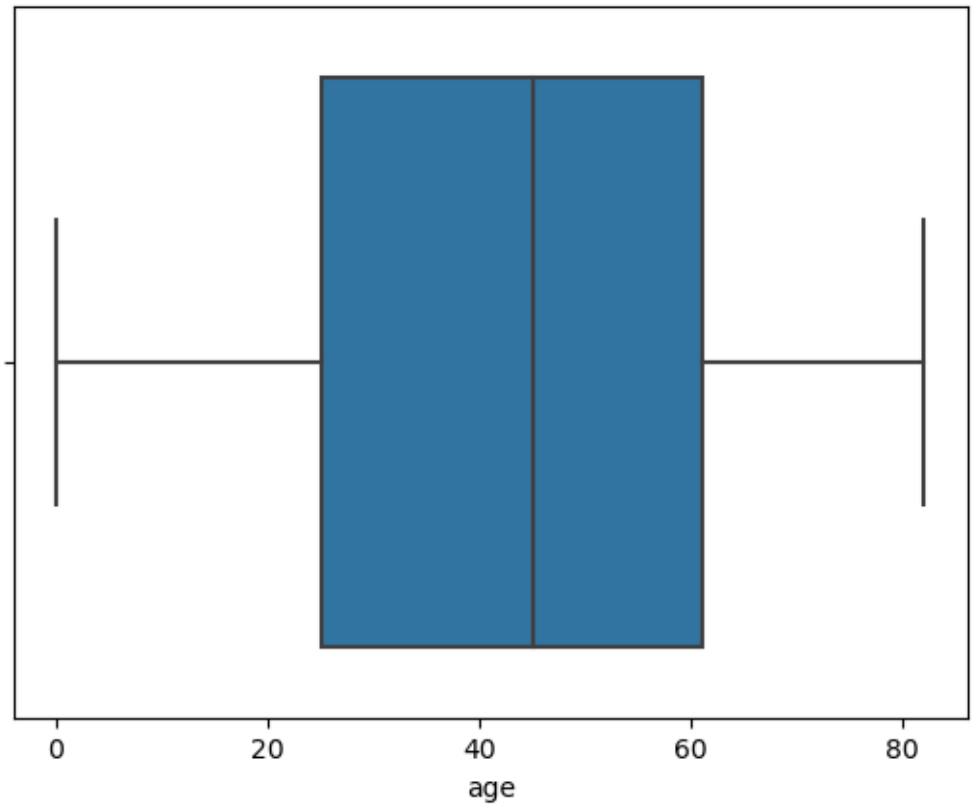
1 EDA

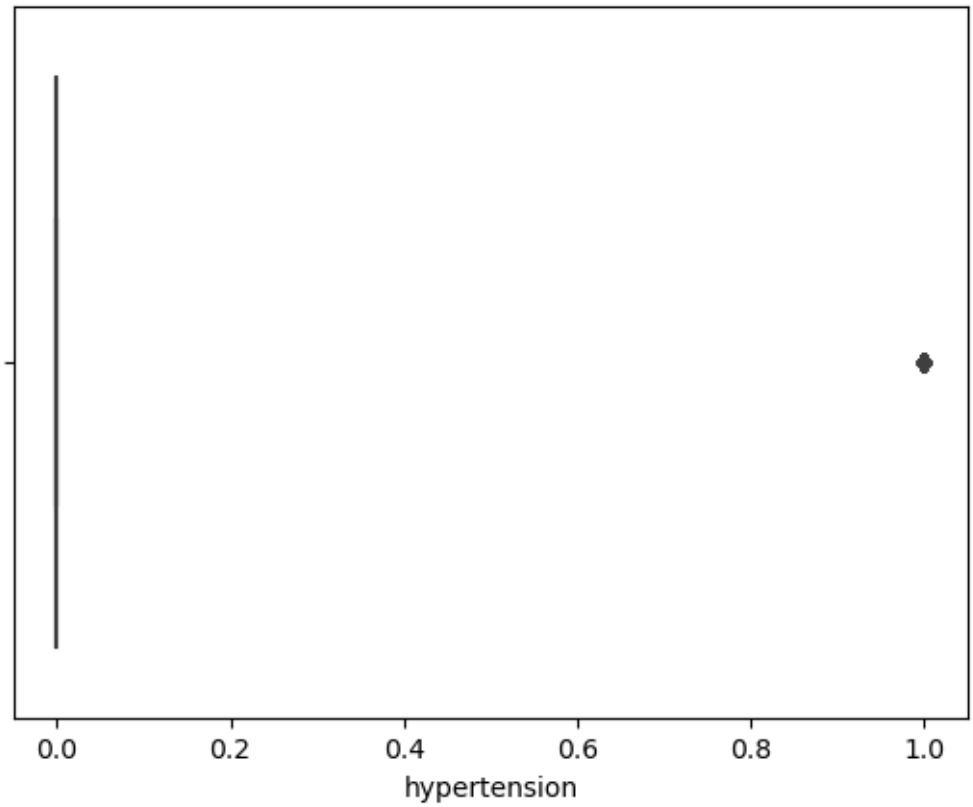
[12]:
```
plt.figure(figsize=(8,4))
sns.countplot(data=df, x='stroke', hue='gender')
plt.xlabel('Stroke')
plt.ylabel('People')
plt.title('Distribution of Stroke affected people')
plt.grid()
plt.show
```

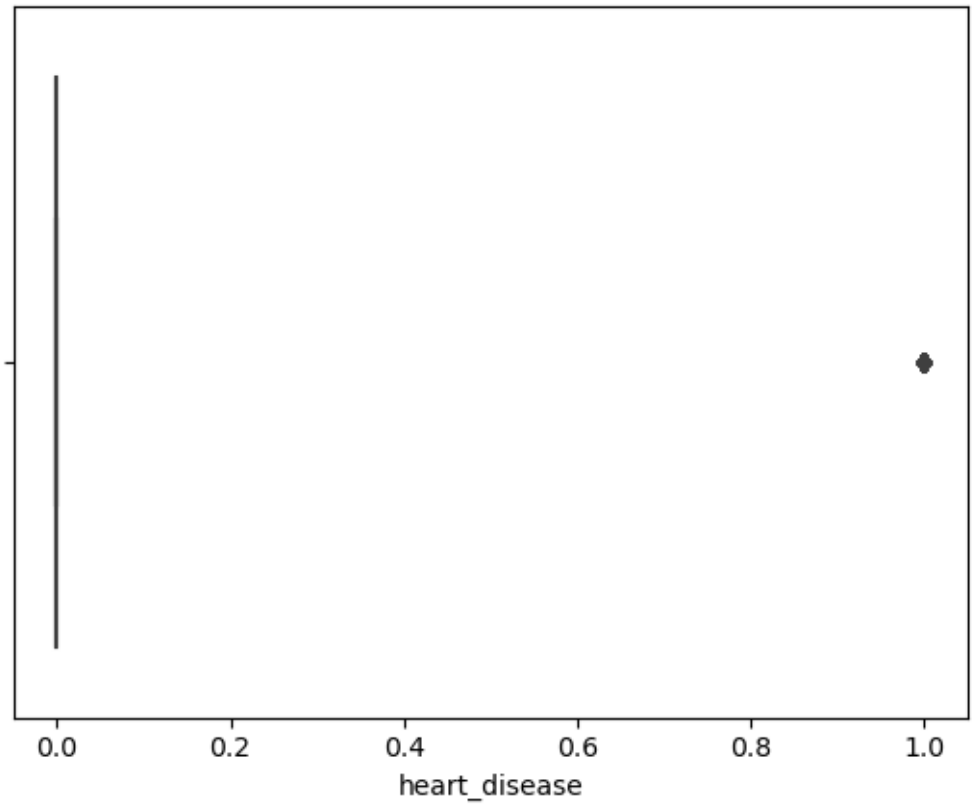
[12]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```

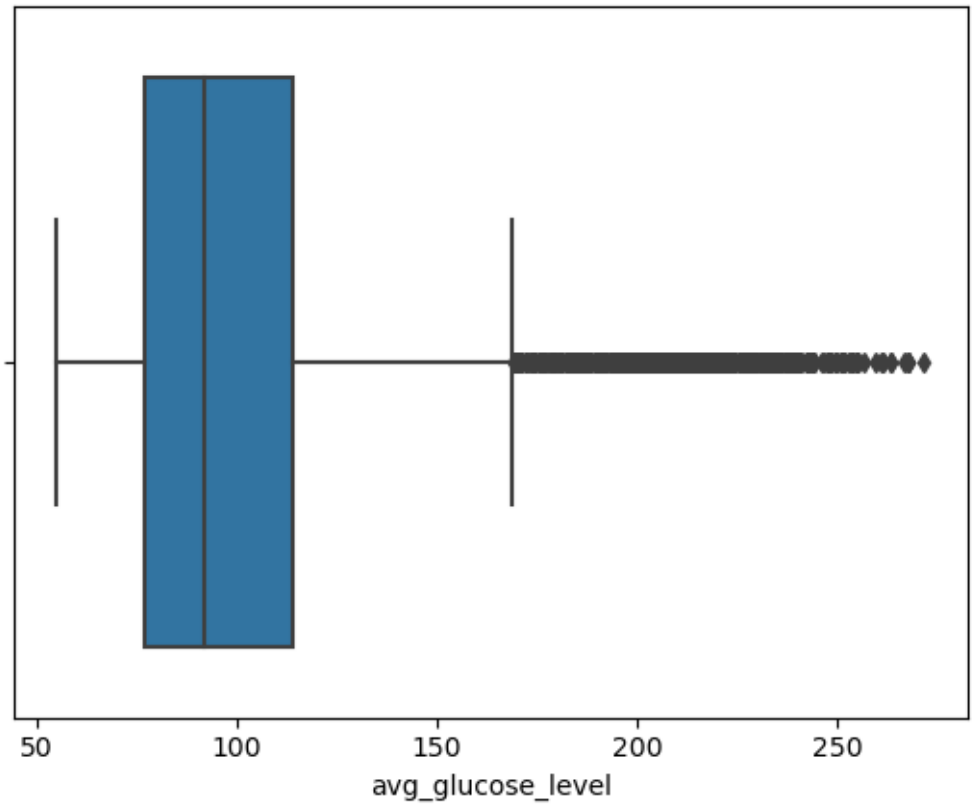


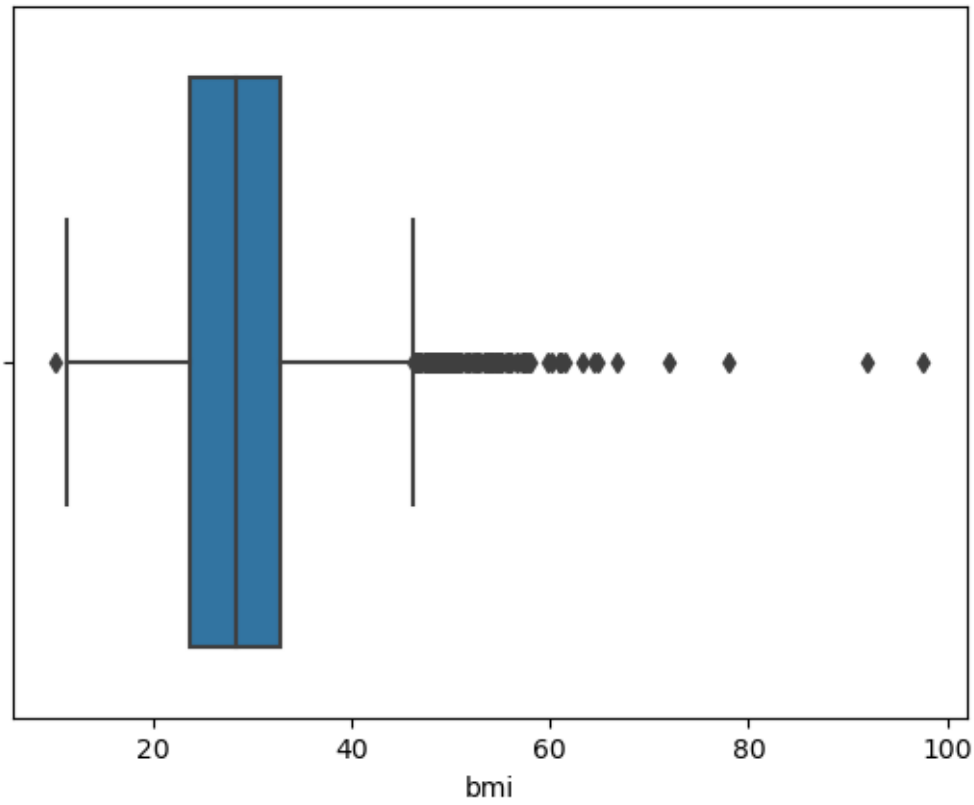
[13]:
```
num = df.select_dtypes(exclude='object')
for I in num.columns:
    sns.boxplot(data=num, x=I)
    plt.show()
```

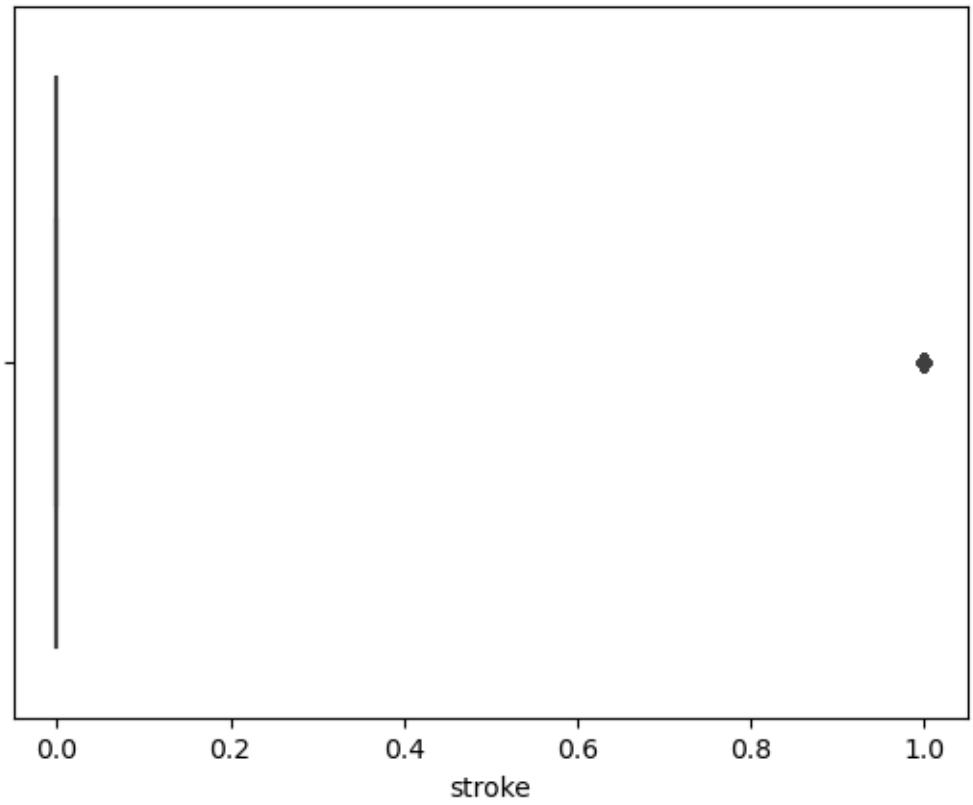

0
20
40
60
80
age


0.0
0.2
0.4
0.6
0.8
1.0
hypertension


0.0
0.2
0.4
0.6
0.8
1.0
heart_disease


50
100
150
200
250
avg_glucose_level


20
40
60
80
100
bmi

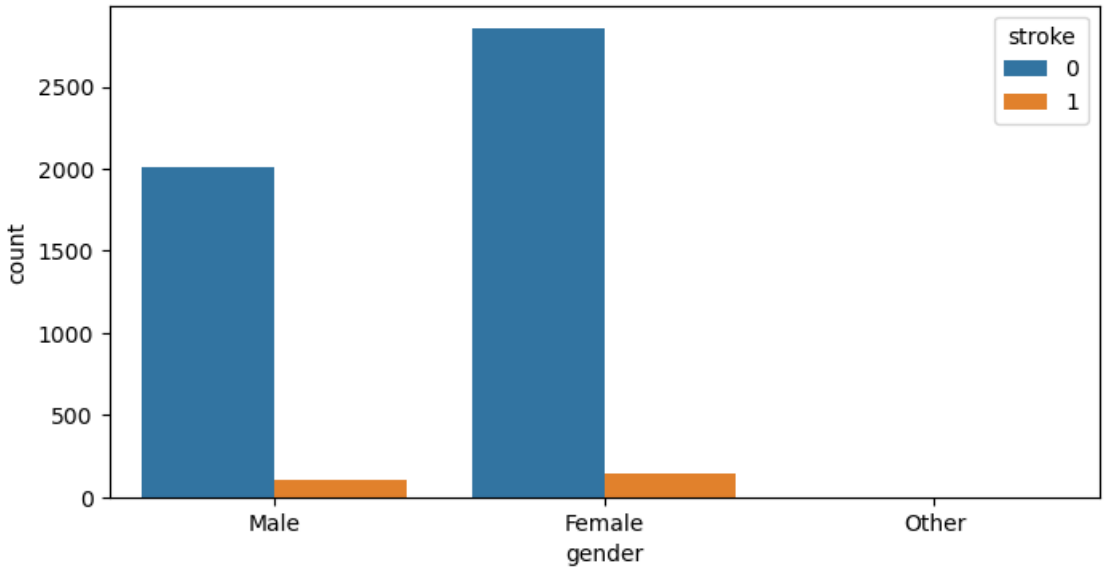


[14]:
```
df['gender'].value_counts()
```

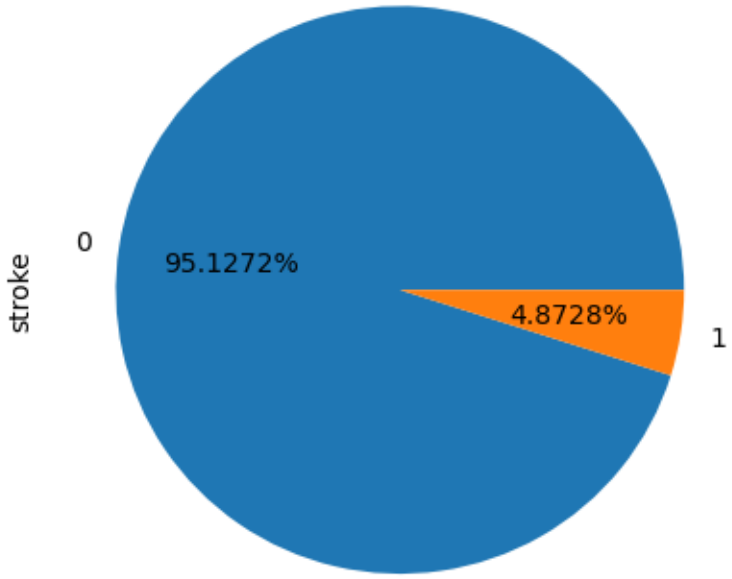
[14]:
```
Female    2994
Male      2115
Other        1
Name: gender, dtype: int64
```

[15]:
```
plt.figure(figsize=(8,4))
sns.countplot(data=df, x='gender', hue='stroke')
```

[15]:
```
<Axes: xlabel='gender', ylabel='count'>
```



[16]:
```
df['stroke'].value_counts().plot(kind='pie', autopct='%.4f%%')
plt.show()
```



[17]:
```
df['age'].value_counts()
```

[17]:
```
78.00    102
57.00     95
52.00     90
54.00     87
51.00     86
         ...
1.40       3
0.48       3
0.16       3
0.40       2
0.08       2
Name: age, Length: 104, dtype: int64
```

[18]:
```
df.groupby('gender').mean()[['age','stroke']]
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_9152\1140679819.py:1: FutureWarning:
The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a
future version, numeric_only will default to False. Either specify numeric_only
or select only columns which should be valid for the function.
  df.groupby('gender').mean()[['age','stroke']]
```

[18]:

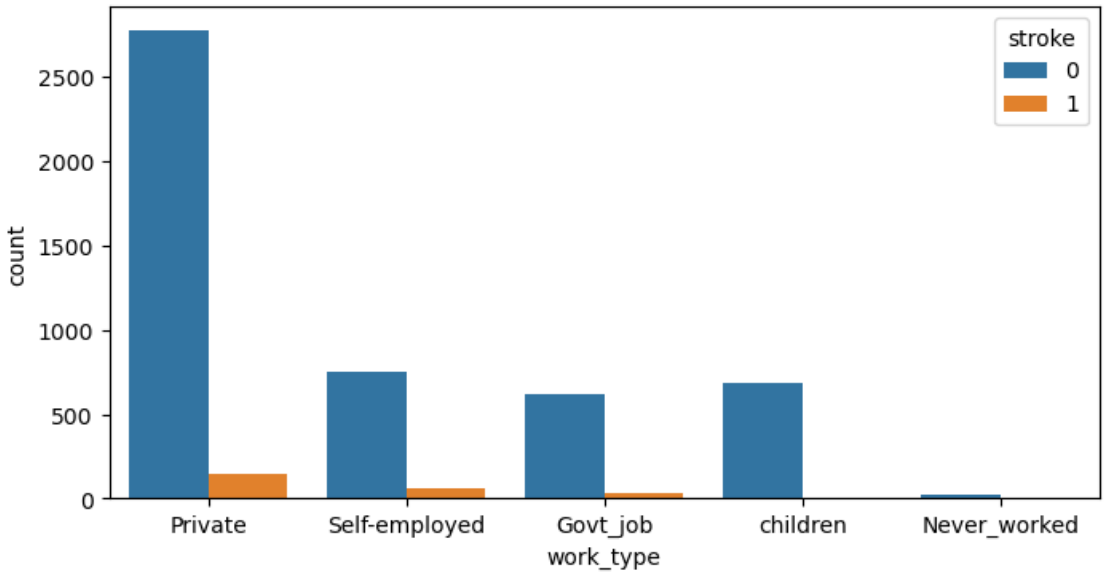
gender	age	stroke
Female	43.757395	0.047094
Male	42.483385	0.051064
Other	26.000000	0.000000

[19]:
```
df['work_type'].value_counts()
```

[19]:
```
Private          2925
Self-employed     819
children          687
Govt_job          657
Never_worked       22
Name: work_type, dtype: int64
```

[20]:
```
plt.figure(figsize=(8,4))
sns.countplot(data=df, x='work_type', hue='stroke')
```

[20]:
```
<Axes: xlabel='work_type', ylabel='count'>
```

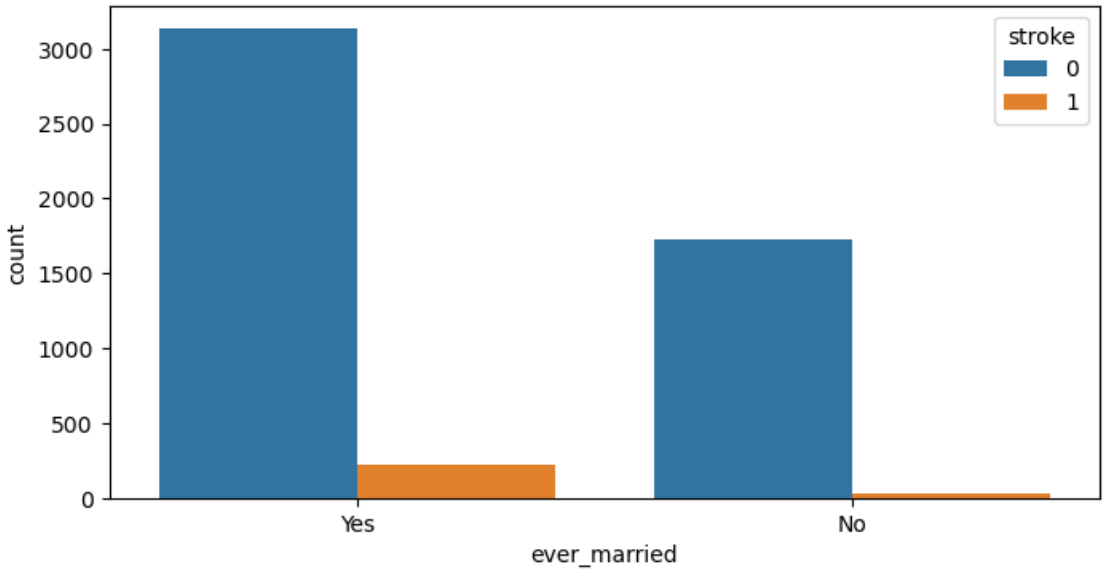


[21]:
```
df['ever_married'].value_counts()
```

[21]:
```
Yes     3353
No      1757
Name: ever_married, dtype: int64
```

[22]:
```
plt.figure(figsize=(8,4))
sns.countplot(data=df, x='ever_married', hue='stroke')
```

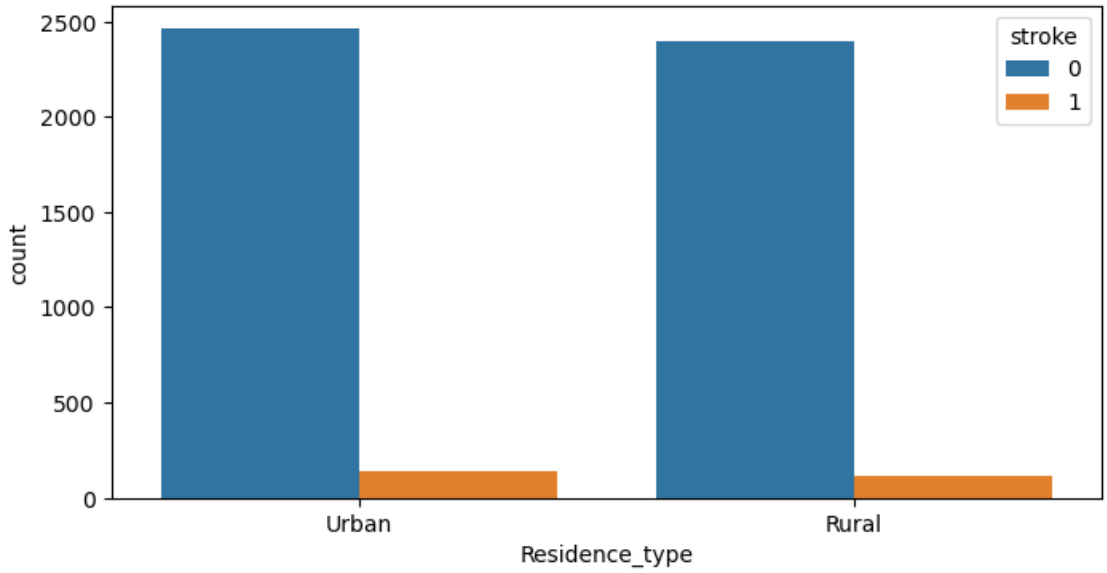
[22]:
```
<Axes: xlabel='ever_married', ylabel='count'>
```



[23]:
```
df['Residence_type'].value_counts()
```

[23]:
```
Urban    2596
Rural    2514
Name: Residence_type, dtype: int64
```

[24]:
```
plt.figure(figsize=(8,4))
sns.countplot(data=df, x='Residence_type', hue='stroke')
plt.show()
```

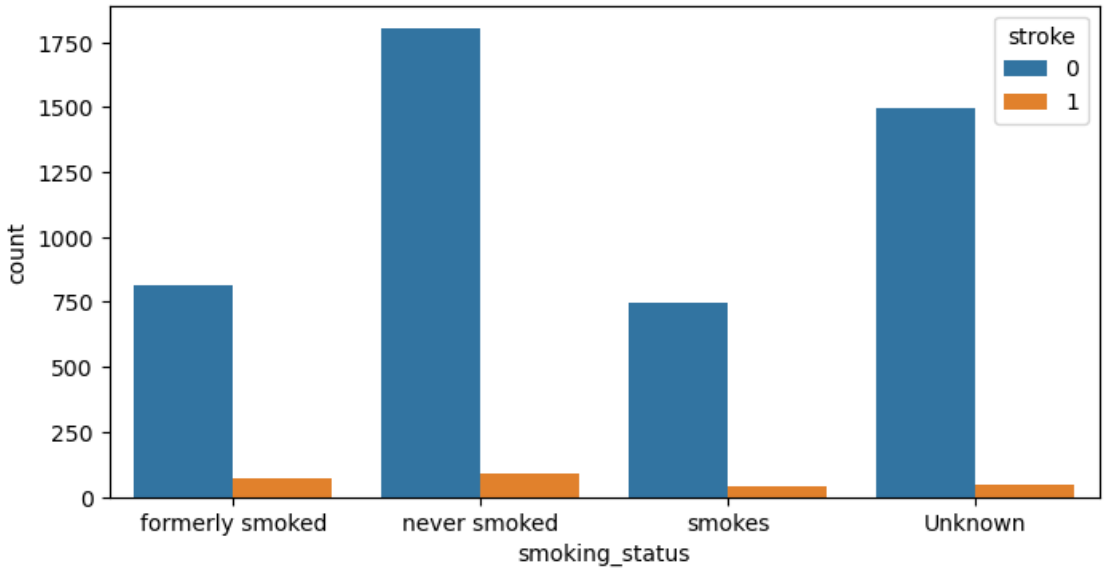


[25]:
```
df['smoking_status'].value_counts()
```

[25]:
```
never smoked       1892
Unknown            1544
formerly smoked     885
smokes              789
Name: smoking_status, dtype: int64
```

[26]:
```
plt.figure(figsize=(8,4))
sns.countplot(data=df, x='smoking_status', hue='stroke')
```

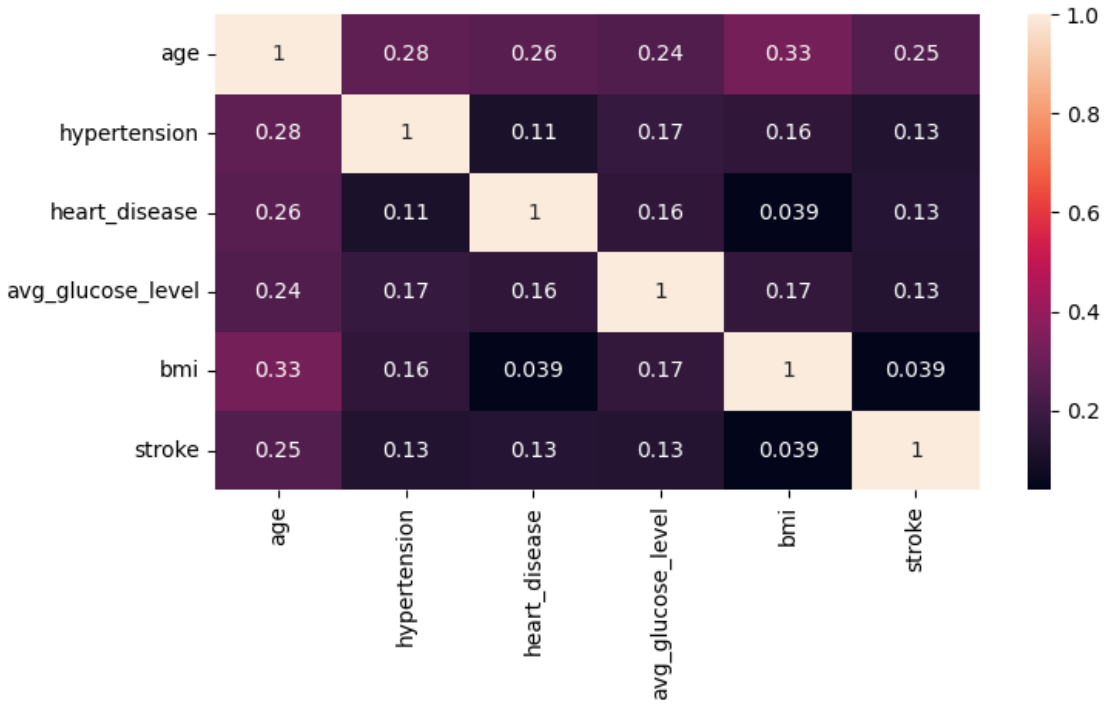
[26]:
```
<Axes: xlabel='smoking_status', ylabel='count'>
```



[27]:
```
plt.figure(figsize=(8,4))
sns.heatmap(df.corr(), annot=True)
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_9152\3820417274.py:2: FutureWarning:
The default value of numeric_only in DataFrame.corr is deprecated. In a future
version, it will default to False. Select only valid columns or specify the
value of numeric_only to silence this warning.
  sns.heatmap(df.corr(), annot=True)
```

[27]: <Axes: >



2 convert categorical data into numerical data

[28]:
```
from sklearn.preprocessing import LabelEncoder
lr=LabelEncoder()
```

[29]:
```
df['gender']=lr.fit_transform(df['gender'])
df['ever_married']=lr.fit_transform(df['ever_married'])
df['work_type']=lr.fit_transform(df['work_type'])
df['Residence_type']=lr.fit_transform(df['Residence_type'])
df['smoking_status']=lr.fit_transform(df['smoking_status'])
```

[30]:
```
df.head()
```

[30]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	1	67.0	0	1	1	2	1	228.69	36.600000	1	1
1	0	61.0	0	0	1	3	0	202.21	28.893237	2	1
2	1	80.0	0	1	1	2	0	105.92	32.500000	2	1
3	0	49.0	0	0	1	2	1	171.23	34.400000	3	1
4	0	79.0	1	0	1	3	0	174.12	24.000000	2	1

```
2               0             105.92  32.500000               2       1
3               1             171.23  34.400000               3       1
4               0             174.12  24.000000               2       1
```

[31]:
```
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   int32
 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   int32
 5   work_type          5110 non-null   int32
 6   Residence_type     5110 non-null   int32
 7   avg_glucose_level  5110 non-null   float64
 8   bmi                5110 non-null   float64
 9   smoking_status     5110 non-null   int32
 10  stroke             5110 non-null   int64
dtypes: float64(3), int32(5), int64(3)
memory usage: 339.5 KB
```

[32]:
```
x = df.drop('stroke', axis=1)
y = df['stroke']
```

[33]:
```
x.head()
```

[33]:
```
   gender   age  hypertension  heart_disease  ever_married  work_type  \
0       1  67.0             0              1             1          2
1       0  61.0             0              0             1          3
2       1  80.0             0              1             1          2
3       0  49.0             0              0             1          2
4       0  79.0             1              0             1          3

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

[34]:
```
y.head()
```

[34]:
```
0    1
1    1
2    1
3    1
4    1
Name: stroke, dtype: int64
```

3 Splitting data into independent and dependent variables

[35]:
```
from sklearn.model_selection import train_test_split
```

[36]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2,␣
 ↪random_state=41)
print('x_train: ', x_train.shape)
print('x_test: ', x_test.shape)
print('y_train: ', y_train.shape)
print('y_test: ', y_test.shape)
```

```
x_train:  (4088, 10)
x_test:  (1022, 10)
y_train:  (4088,)
y_test:  (1022,)
```

4 using for loop

[59]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

[60]:
```
from sklearn.metrics import classification_report, accuracy_score,␣
 ↪confusion_matrix, mean_squared_error
```

[61]:
```
models = {
    'LogisticRegression':LogisticRegression(),
    'KNN':KNeighborsClassifier(),
    'DecisionTree':DecisionTreeClassifier(max_depth=3),
    'RandomForestClassifier':RandomForestClassifier()
}
```

[62]:
```
results = pd.DataFrame(columns=['Model','MSE', 'Accuracy score','conf_matrix'])
```

[63]:
```
for model_name, model in models.items():
    model.fit(x_train,y_train)
    pred = model.predict(x_test)
```

```
    mse = mean_squared_error(y_test, pred)
    accuracy_score1 = accuracy_score(y_test,pred)
    conf_matrix = confusion_matrix(y_test,pred)
    results = results.append({"Model":model_name, 'MSE':mse,'Accuracy score':
 ↪accuracy_score1,
                              'conf_matrix':conf_matrix},ignore_index=True
)
print(results)
```

```
C:\Users\Admin\anaconda3\Lib\site-
packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
C:\Users\Admin\AppData\Local\Temp\ipykernel_9152\1326110178.py:7: FutureWarning:
The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.
  results = results.append({"Model":model_name, 'MSE':mse,'Accuracy
score':accuracy_score1,
C:\Users\Admin\AppData\Local\Temp\ipykernel_9152\1326110178.py:7: FutureWarning:
The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.
  results = results.append({"Model":model_name, 'MSE':mse,'Accuracy
score':accuracy_score1,
C:\Users\Admin\AppData\Local\Temp\ipykernel_9152\1326110178.py:7: FutureWarning:
The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.
  results = results.append({"Model":model_name, 'MSE':mse,'Accuracy
score':accuracy_score1,
```

```
                    Model       MSE  Accuracy score           conf_matrix
0      LogisticRegression  0.057730        0.942270  [[963, 0], [59, 0]]
1                     KNN  0.059687        0.940313  [[960, 3], [58, 1]]
2            DecisionTree  0.059687        0.940313  [[961, 2], [59, 0]]
3  RandomForestClassifier  0.058708        0.941292  [[962, 1], [59, 0]]
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_9152\1326110178.py:7: FutureWarning:
The frame.append method is deprecated and will be removed from pandas in a
future version. Use pandas.concat instead.
  results = results.append({"Model":model_name, 'MSE':mse,'Accuracy
score':accuracy_score1,
```

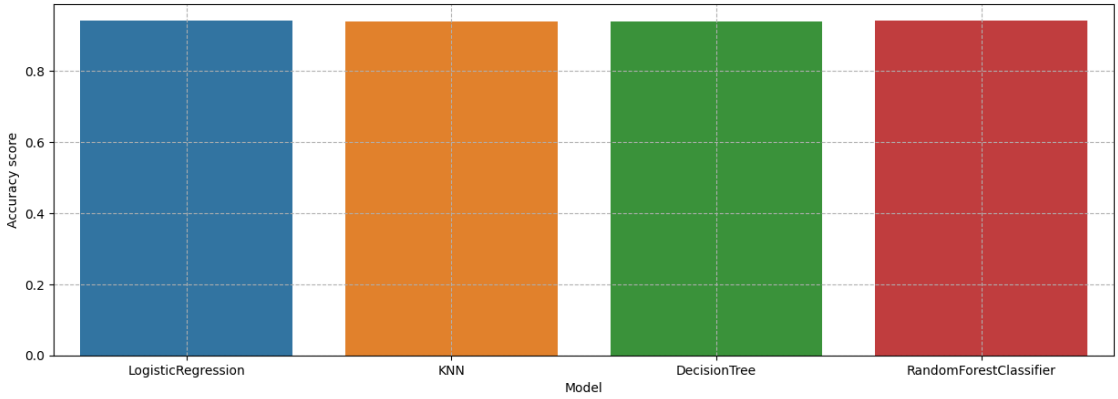
[64]:
```
data = pd.DataFrame(results)
```

[65]:
```
data
```

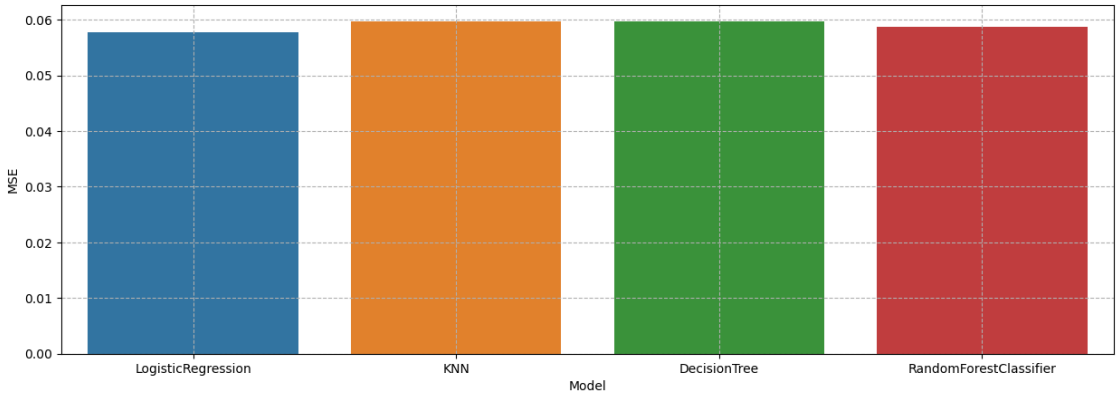
[65]:

	Model	MSE	Accuracy score	conf_matrix
0	LogisticRegression	0.057730	0.942270	[[963, 0], [59, 0]]
1	KNN	0.059687	0.940313	[[960, 3], [58, 1]]
2	DecisionTree	0.059687	0.940313	[[961, 2], [59, 0]]
3	RandomForestClassifier	0.058708	0.941292	[[962, 1], [59, 0]]

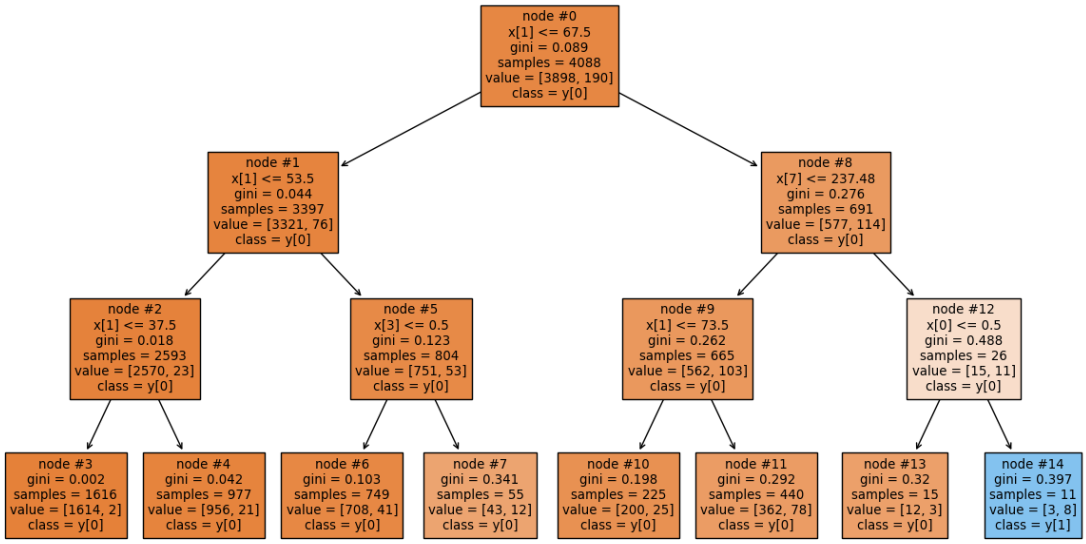
[66]:
```
plt.figure(figsize=(15,5))
sns.barplot(data=data, y='Accuracy score',x='Model')
plt.grid(linestyle='--')
plt.show()
```



[67]:
```
plt.figure(figsize=(15,5))
sns.barplot(data=data, y='MSE', x='Model')
plt.grid(linestyle='--')
plt.show()
```



[70]:
```
from sklearn import tree
plt.figure(figsize=(15,8))
tree.plot_tree(models['DecisionTree'], filled=True, class_names=True,␣
 ↪node_ids=True)
plt.show();
```



[ ]:
```

```