Healthcare Stroke Analysis Using Python

Project Description:
This project aims to analyze healthcare data related to strokes using Python, focusing on data preprocessing, visualization, and machine learning model implementation. The goal is to identify key factors associated with stroke occurrences and develop predictive models to aid in early detection and prevention.

Importing libraries

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

Uploading csv file

+ Code + Text

```
stroke_dataset = pd.read_csv("/content/17_Healthcare Stroke Analysis.csv")
```

I - Data Preprocessing

.head()

```
stroke_dataset.head()
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residenc
0	9046	Male	67.0	0	1	Yes	Private	
1	51676	Female	61.0	0	0	Yes	Self-employed	
2	31112	Male	80.0	0	1	Yes	Private	
3	60182	Female	49.0	0	0	Yes	Private	

.tail

```
stroke_dataset.tail()
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Resid
5104	14180	Female	13.0	0	0	No	children	
5106	44873	Female	81.0	0	0	Yes	Self-employed	
5107	19723	Female	35.0	0	0	Yes	Self-employed	
5108	37544	Male	51.0	0	0	Yes	Private	

.shape

```
stroke_dataset.shape
```

```
(5110, 12)
```

.columns

```
stroke_dataset.columns
```

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

.dtypes

```
stroke_dataset.dtypes
```

```
id                   int64
gender              object
age                float64
hypertension         int64
heart_disease        int64
ever_married        object
work_type           object
Residence_type      object
avg_glucose_level  float64
bmi                float64
smoking_status      object
stroke               int64
dtype: object
```

.unique()

```
stroke_dataset["gender"].unique()
```

```
array(['Male', 'Female', 'Other'], dtype=object)
```

```
stroke_dataset["work_type"].unique()
```

```
array(['Private', 'Self-employed', 'Govt_job', 'children', 'Never_worked'],
      dtype=object)
```

```
stroke_dataset["Residence_type"].unique()
```

```
array(['Urban', 'Rural'], dtype=object)
```

```
stroke_dataset["avg_glucose_level"].unique()
```

```
array([228.69, 202.21, 105.92, ...,  82.99, 166.29,  85.28])
```

```
stroke_dataset["smoking_status"].unique()
```

```
array(['formerly smoked', 'never smoked', 'smokes', 'Unknown'],
      dtype=object)
```

.nunique()

```
stroke_dataset.nunique()
```

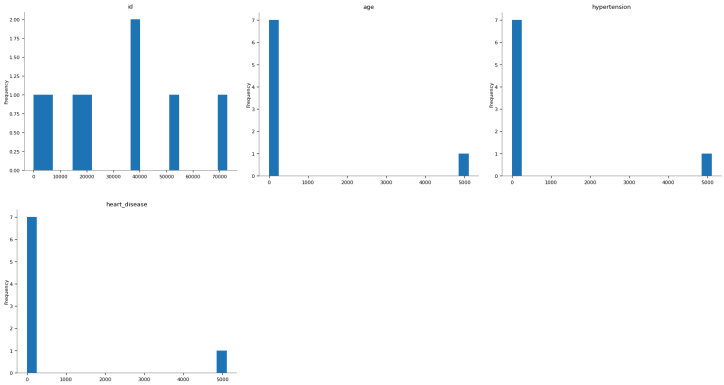
```
id                 5110
gender                3
age                 104
hypertension          2
heart_disease         2
ever_married          2
work_type             5
Residence_type        2
avg_glucose_level  3979
bmi                 418
smoking_status        4
stroke                2
dtype: int64
```

.describe()

```
stroke_dataset.describe()
```

	id	age	hypertension	heart_disease	avg_glucose_level	
count	5110.000000	5110.000000	5110.000000	5110.000000	5110.000000	4909
mean	36517.829354	43.226614	0.097456	0.054012	106.147677	28
std	21161.721625	22.612647	0.296607	0.226063	45.283560	7
min	67.000000	0.080000	0.000000	0.000000	55.120000	10
25%	17741.250000	25.000000	0.000000	0.000000	77.245000	23
50%	36932.000000	45.000000	0.000000	0.000000	91.885000	28
75%	54682.000000	61.000000	0.000000	0.000000	114.090000	33
max	72940.000000	82.000000	1.000000	1.000000	271.740000	97

Distributions



2-d distributions

.value_counts

```
stroke_dataset["gender"].value_counts()
```

```
gender
Female    2994
Male      2115
Other        1
Name: count, dtype: int64
```

```
stroke_dataset["age"].value_counts()
```

```
age
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: count, Length: 104, dtype: int64
```

```
stroke_dataset["heart_disease"].value_counts()
```

```
heart_disease
0    4834
1     276
Name: count, dtype: int64
```

```
stroke_dataset["ever_married"].value_counts()
```

```
ever_married
Yes    3353
No     1757
Name: count, dtype: int64
```

```
stroke_dataset["work_type"].value_counts()
```

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

```
stroke_dataset["Residence_type"].value_counts()
```

```
Residence_type
Urban    2596
Rural    2514
Name: count, dtype: int64
```

```
stroke_dataset["avg_glucose_level"].value_counts()
```

```
avg_glucose_level
93.88     6
91.68     5
91.85     5
83.16     5
73.00     5
         ..
111.93    1
94.40     1
95.57     1
66.29     1
85.28     1
Name: count, Length: 3979, dtype: int64
```

```
stroke_dataset["bmi"].value_counts()
```

```
bmi
28.7    41
28.4    38
26.7    37
27.6    37
26.1    37
        ..
48.7     1
49.2     1
51.0     1
49.4     1
14.9     1
Name: count, Length: 418, dtype: int64
```

```
stroke_dataset["smoking_status"].value_counts()
```

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

```
stroke_dataset["stroke"].value_counts()
```

```
stroke
0    4861
1     249
Name: count, dtype: int64
```

.isnull()

```
stroke_dataset.isnull()
```

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Resi
0	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	
5105	False	False	False	False	False	False	False	
5106	False	False	False	False	False	False	False	
5107	False	False	False	False	False	False	False	
5108	False	False	False	False	False	False	False	
5109	False	False	False	False	False	False	False	

5110 rows × 12 columns

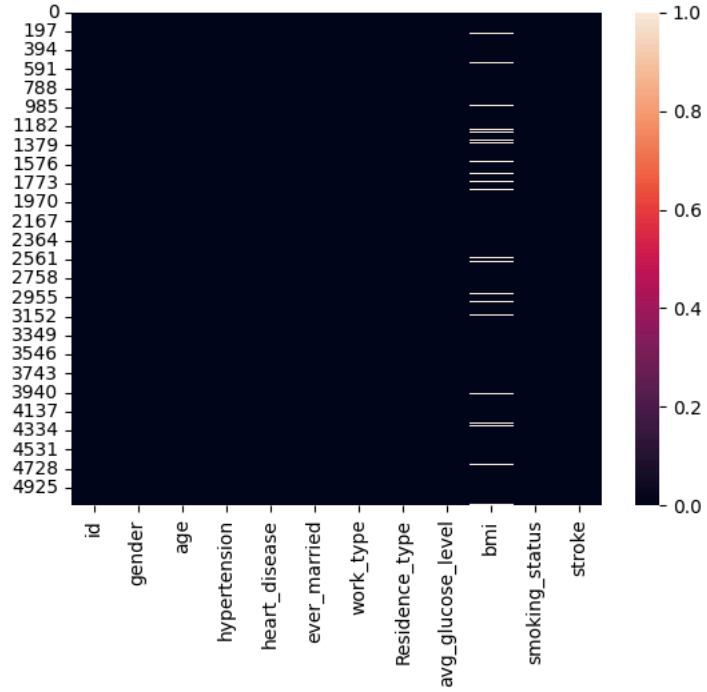
Categorical distributions

```
stroke_dataset.isnull().sum()
```

```
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
```

```
sns.heatmap(stroke_dataset.isnull())
```

```
<Axes: >
```



handling missing values

```
stroke_dataset = stroke_dataset[stroke_dataset['age'] % 1 == 0]
```

```
stroke_dataset.dtypes
```

```
id                   int64
gender              object
age                float64
hypertension         int64
heart_disease        int64
ever_married        object
work_type           object
Residence_type      object
avg_glucose_level  float64
bmi                float64
smoking_status      object
stroke               int64
dtype: object
```

```
# Drop rows with missing values
stroke_dataset.dropna(subset=['bmi'], inplace=True)
```

```
<ipython-input-89-d1976de44458>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  stroke_dataset.dropna(subset=['bmi'], inplace=True)
```

convert bmi to category

```
stroke_dataset['bmi_category'] = pd.cut(stroke_dataset['bmi'],
                                        bins=[0, 18.5, 25, 30, float('inf')],
                                        labels=['Underweight', 'Normal', 'Overweight', 'Obese'],
                                        right=False)
```

```
<ipython-input-90-d69a0d7158b1>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  stroke_dataset['bmi_category'] = pd.cut(stroke_dataset['bmi'],
```

```
print(stroke_dataset[['bmi', 'bmi_category']].head())
```

```
    bmi bmi_category
0  36.6        Obese
2  32.5        Obese
3  34.4        Obese
4  24.0       Normal
5  29.0   Overweight
```

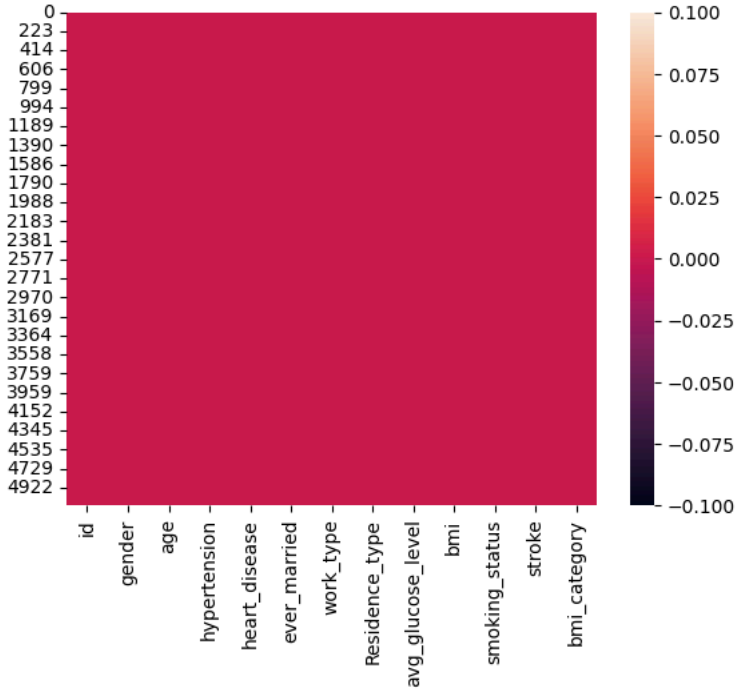
```
bmi_mapping = {
    'Underweight': 0,
    'Normal': 1,
    'Overweight': 2,
    'Obese': 3
}

# Apply the mapping to the bmi_category column
stroke_dataset['bmi_category'] = stroke_dataset['bmi_category'].map(bmi_mapping)

# Drop rows with NaN values in bmi_category (if any category was not mapped)
stroke_dataset = stroke_dataset.dropna(subset=['bmi_category'])

# Convert the bmi_category column to integer type
stroke_dataset['bmi_category'] = stroke_dataset['bmi_category'].astype(int)

# Check the first few rows of the DataFrame to confirm
print(stroke_dataset.head())

stroke_dataset.to_csv('stroke_dataset_with_bmi_category.csv', index=False)
```

```
      id  gender   age  hypertension  heart_disease ever_married  \
0   9046    Male  67.0             0              1          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
5  56669    Male  81.0             0              0          Yes

       work_type Residence_type  avg_glucose_level   bmi   smoking_status  \
0        Private          Urban             228.69  36.6  formerly 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
5        Private          Urban             186.21  29.0  formerly smoked

   stroke  bmi_category
0       1             3
2       1             3
3       1             3
4       1             1
5       1             2
<ipython-input-92-0030ce49cdeb>:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  stroke_dataset['bmi_category'] = stroke_dataset['bmi_category'].map(bmi_mapping)
```

after handling missing values

```
sns.heatmap(stroke_dataset.isnull())
```

```
<Axes: >
```

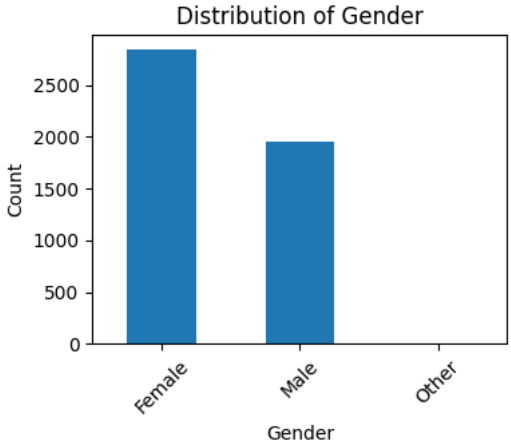


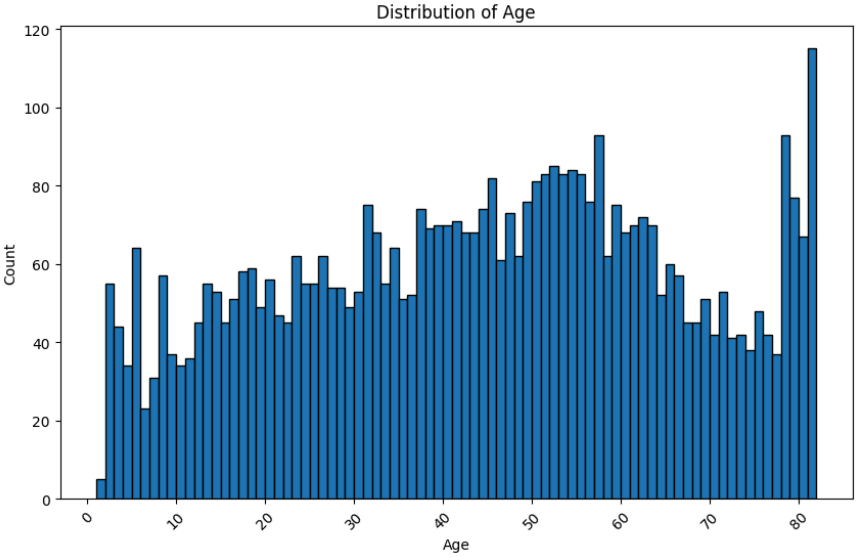
column visualizatoin

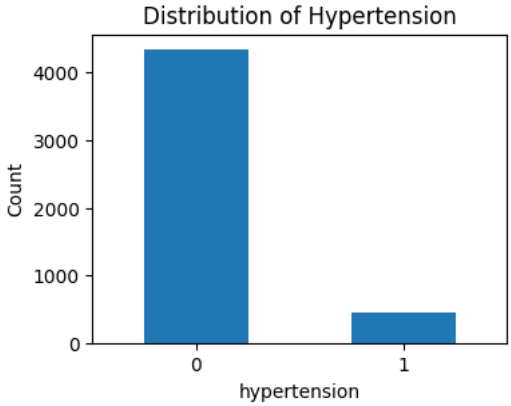
```
# gender
gender_counts = stroke_dataset['gender'].value_counts()
plt.figure(figsize=(4, 3))
gender_counts.plot(kind='bar')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.title('Distribution of Gender')
plt.xticks(rotation=45)
plt.show()

# age
plt.figure(figsize=(10, 6))
plt.hist(stroke_dataset['age'], bins=range(int(stroke_dataset['age'].min()), int(stroke_dataset['age'].max()) + 1, 1), edgecolor='black
plt.xlabel('Age')
plt.ylabel('Count')
plt.title('Distribution of Age')
plt.xticks(rotation=45)
plt.show()

# hypertension
hypertension_counts = stroke_dataset['hypertension'].value_counts()
plt.figure(figsize=(4, 3))
hypertension_counts.plot(kind='bar')
plt.xlabel('hypertension')
plt.ylabel('Count')
plt.title('Distribution of Hypertension')
plt.xticks(rotation=0)
plt.show()

# heart disease
heart_disease_counts = stroke_dataset['heart_disease'].value_counts()
plt.figure(figsize=(4, 3))
heart_disease_counts.plot(kind='bar')
plt.xlabel('heart_disease')
plt.ylabel('Count')
plt.title('Distribution of heart disease')
plt.xticks(rotation=0)
plt.show()

# married
married_counts = stroke_dataset['ever_married'].value_counts()
plt.figure(figsize=(4, 3))
married_counts.plot(kind='bar')
plt.xlabel('ever_married')
plt.ylabel('Count')
plt.title('Distribution of ever married')
plt.xticks(rotation=45)
plt.show()

# work type
work_type_counts = stroke_dataset['work_type'].value_counts()
plt.figure(figsize=(4, 3))
work_type_counts.plot(kind='bar')
plt.xlabel('work_type')
plt.ylabel('Count')
plt.title('Distribution of work type')
plt.xticks(rotation=45)
plt.show()

# Residence type
Residence_type_counts = stroke_dataset['Residence_type'].value_counts()
plt.figure(figsize=(4, 3))
Residence_type_counts.plot(kind='bar')
plt.xlabel('Residence_type')
plt.ylabel('Count')
plt.title('Distribution of Residence type')
plt.xticks(rotation=45)
plt.show()

# avg glucose level
plt.figure(figsize=(10, 6))
plt.hist(stroke_dataset['avg_glucose_level'], bins=20, edgecolor='black')
plt.xlabel('Average Glucose Level')
plt.ylabel('Count')
plt.title('Distribution of Average Glucose Level')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

# bmi category
bmi_category_counts = stroke_dataset['bmi_category'].value_counts()
plt.figure(figsize=(4, 3))
```

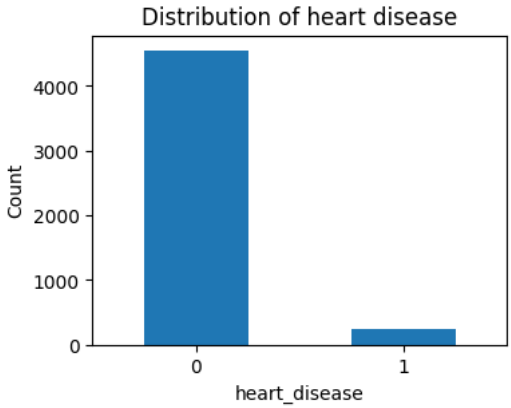
```
bmi_category_counts.plot(kind='bar')
plt.xlabel('bmi_category')
plt.ylabel('Count')
plt.title('Distribution of bmi_category')
plt.xticks(rotation=45)
plt.show()

# smoking status
smoking_status_counts = stroke_dataset['smoking_status'].value_counts()
plt.figure(figsize=(4, 3))
smoking_status_counts.plot(kind='bar')
plt.xlabel('smoking_status')
plt.ylabel('Count')
plt.title('Distribution of avg smoking status')
plt.xticks(rotation=45)
plt.show()

# stroke
stroke_counts = stroke_dataset['stroke'].value_counts()
plt.figure(figsize=(4, 3))
stroke_counts.plot(kind='bar')
plt.xlabel('stroke')
plt.ylabel('Count')
plt.title('Distribution of stroke')
plt.xticks(rotation=0)
plt.show()
```

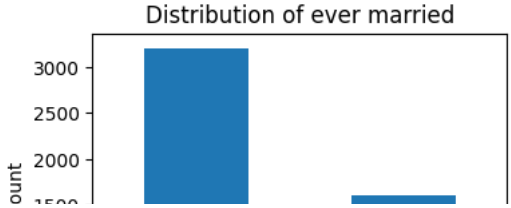
[4]

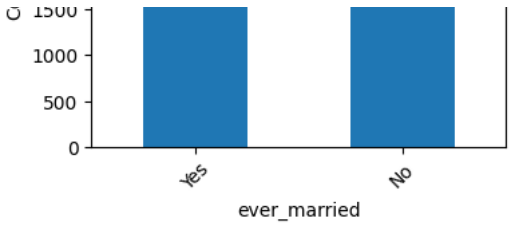

Distribution of Gender
Count
2500
2000
1500
1000
500
0
Female
Male
Other
Gender

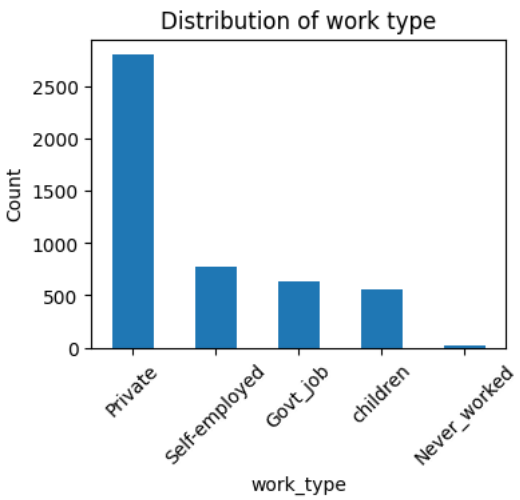

Distribution of Age
Count
120
100
80
60
40
20
0
0
10
20
30
40
50
60
70
80
Age

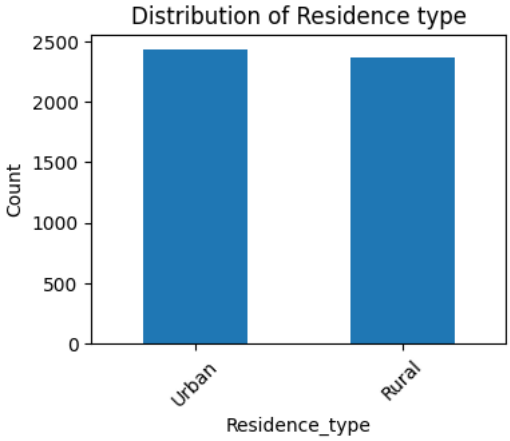

Distribution of Hypertension
Count
4000
3000
2000
1000
0
0
1
hypertension

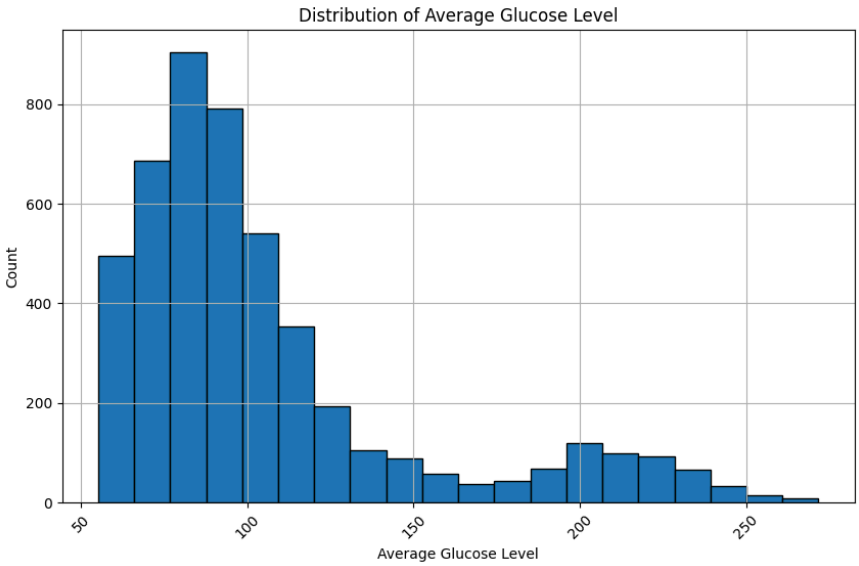

Distribution of heart disease
Count
4000
3000
2000
1000
0
0
1
heart_disease

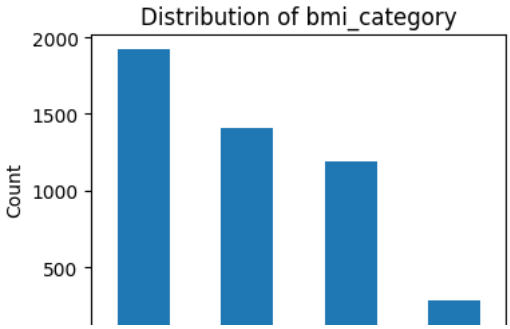

Distribution of ever married
3000
2500
2000

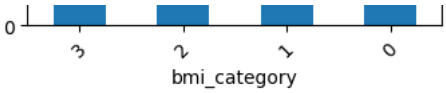

1500
1000
500
0
Yes
No
ever_married

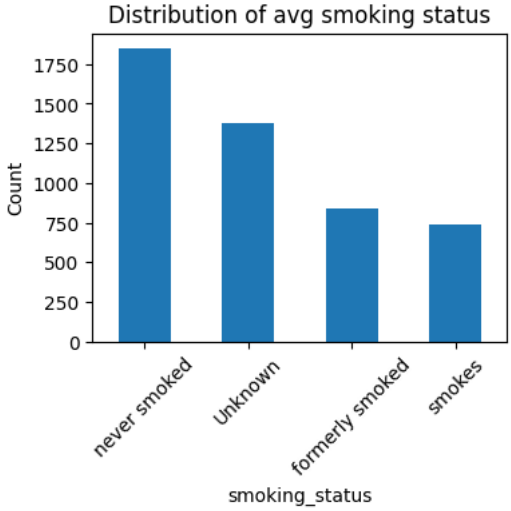

Distribution of work type
Count
2500
2000
1500
1000
500
0
Private
Self-employed
Govt_job
children
Never_worked
work_type

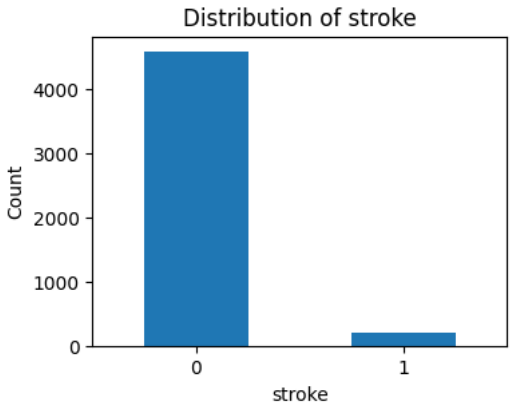

Distribution of Residence type
Count
2500
2000
1500
1000
500
0
Urban
Rural
Residence_type


Distribution of Average Glucose Level
Count
800
600
400
200
0
50
100
150
200
250
Average Glucose Level


Distribution of bmi_category
Count
2000
1500
1000
500

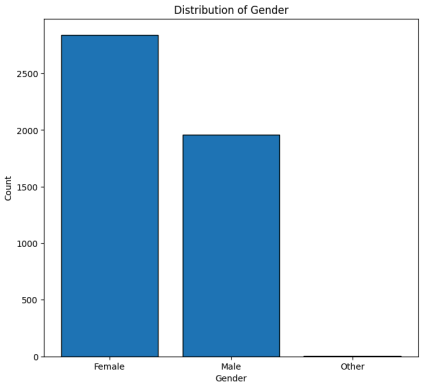


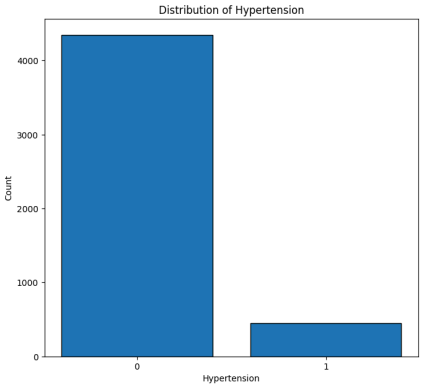


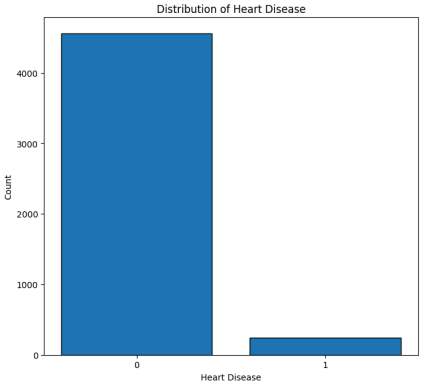


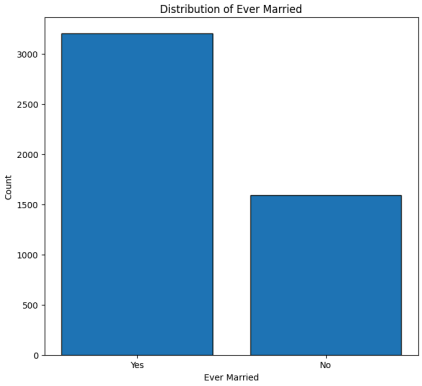
column visualization (by row)

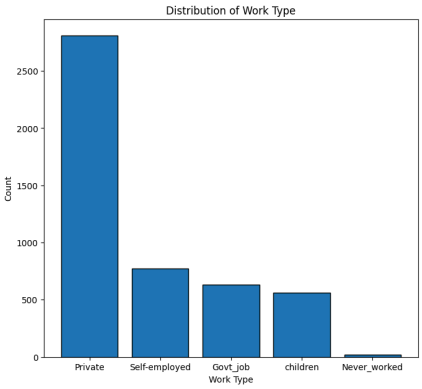
```
def plot_bar(data, title, xlabel, ylabel, ax):
    ax.bar(data.index, data.values, edgecolor='black')
    ax.set_xlabel(xlabel)
    ax.set_ylabel(ylabel)
    ax.set_title(title)

fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(15, 20))
fig.tight_layout(pad=5.0)

# Gender
gender_counts = stroke_dataset['gender'].value_counts()
plot_bar(gender_counts, 'Distribution of Gender', 'Gender', 'Count', axes[0, 0])

# Hypertension
hypertension_counts = stroke_dataset['hypertension'].value_counts()
plot_bar(hypertension_counts, 'Distribution of Hypertension', 'Hypertension', 'Count', axes[0, 1])
axes[0, 1].set_xticks([0, 1])

# Heart Disease
heart_disease_counts = stroke_dataset['heart_disease'].value_counts()
plot_bar(heart_disease_counts, 'Distribution of Heart Disease', 'Heart Disease', 'Count', axes[1, 0])
axes[1, 0].set_xticks([0, 1])

# Ever Married
married_counts = stroke_dataset['ever_married'].value_counts()
plot_bar(married_counts, 'Distribution of Ever Married', 'Ever Married', 'Count', axes[1, 1])

# Work Type
work_type_counts = stroke_dataset['work_type'].value_counts()
plot_bar(work_type_counts, 'Distribution of Work Type', 'Work Type', 'Count', axes[2, 0])

# Residence Type
residence_type_counts = stroke_dataset['Residence_type'].value_counts()
plot_bar(residence_type_counts, 'Distribution of Residence Type', 'Residence Type', 'Count', axes[2, 1])

plt.show()

# Additional plots in a separate figure
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(15, 5))
fig.tight_layout(pad=5.0)

# BMI Category
bmi_category_counts = stroke_dataset['bmi_category'].value_counts()
plot_bar(bmi_category_counts, 'Distribution of BMI Category', 'BMI Category', 'Count', axes[0])

# Smoking Status
smoking_status_counts = stroke_dataset['smoking_status'].value_counts()
plot_bar(smoking_status_counts, 'Distribution of Smoking Status', 'Smoking Status', 'Count', axes[1])

# Stroke
stroke_counts = stroke_dataset['stroke'].value_counts()
plot_bar(stroke_counts, 'Distribution of Stroke', 'Stroke', 'Count', axes[2])
axes[2].set_xticks([0, 1])

plt.show()

# Histograms
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(15, 5))
fig.tight_layout(pad=5.0)

# Age
axes[0].hist(stroke_dataset['age'], bins=range(int(stroke_dataset['age'].min()), int(stroke_dataset['age'].max()) + 1, 1), edgecolor='b
axes[0].set_title('Distribution of Age')
axes[0].set_xlabel('Age')
axes[0].set_ylabel('Count')

# Avg Glucose Level
axes[1].hist(stroke_dataset['avg_glucose_level'], bins=20, edgecolor='black')
axes[1].set_title('Distribution of Average Glucose Level')
axes[1].set_xlabel('Average Glucose Level')
axes[1].set_ylabel('Count')

plt.show()
```

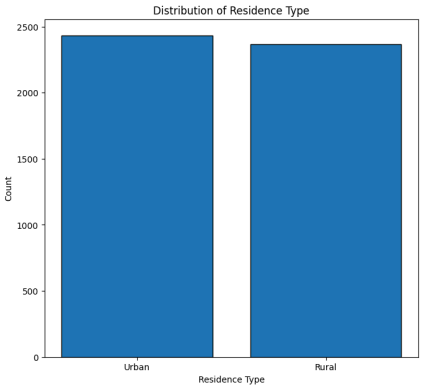

Distribution of Gender
Count
Gender
Female
Male
Other

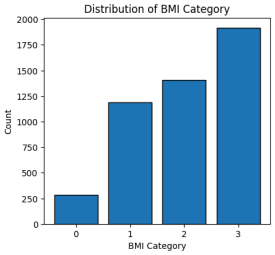

Distribution of Hypertension
Count
Hypertension
0
1

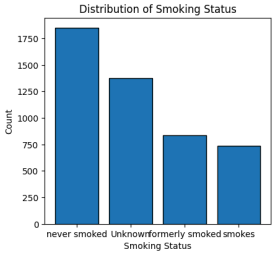

Distribution of Heart Disease
Count
Heart Disease
0
1

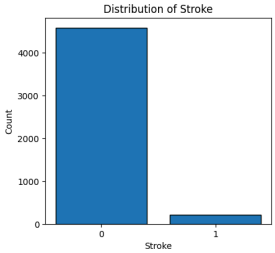

Distribution of Ever Married
Count
Ever Married
Yes
No

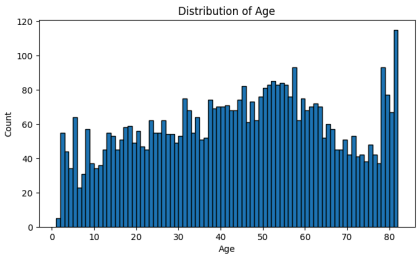

Distribution of Work Type
Count
Work Type
Private
Self-employed
Govt_job
children
Never_worked

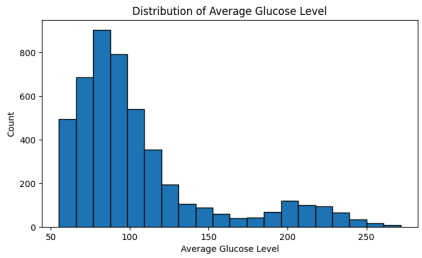

Distribution of Residence Type
Count
Residence Type
Urban
Rural


Distribution of BMI Category
Count
BMI Category


Distribution of Smoking Status
Count
Smoking Status
never smoked
Unknown
formerly smoked
smokes


Distribution of Stroke
Count
Stroke


Distribution of Age
Count
Age


Distribution of Average Glucose Level
Count
Average Glucose Level

II - Multiple Linear Regression

1. Import libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestClassifier
```

2. Loading the dataset

```
stroke_datasetv2 = pd.read_csv("/content/stroke_dataset_with_bmi_category.csv")
```

3. Data preprocessing

```
stroke_datasetv2 = pd.get_dummies(stroke_dataset, columns=['gender', 'ever_married', 'work_type', 'Residence_type', 'smoking_status', ])
```

4. Splitting the data into features and target variable

```
X = stroke_datasetv2.drop(['id', 'stroke'], axis=1)
y = stroke_datasetv2['stroke']
```

5. Train-Test Split

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

6. Model Selection

```
model = LinearRegression()
```

7. Model fitting

```
model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
print(X.dtypes)
```

```
age                                float64
hypertension                         int64
heart_disease                        int64
avg_glucose_level                  float64
bmi                                float64
bmi_category                         int64
gender_Female                         bool
gender_Male                           bool
gender_Other                          bool
ever_married_No                       bool
ever_married_Yes                      bool
work_type_Govt_job                    bool
work_type_Never_worked                bool
work_type_Private                     bool
work_type_Self-employed               bool
work_type_children                    bool
Residence_type_Rural                  bool
Residence_type_Urban                  bool
smoking_status_Unknown                bool
smoking_status_formerly smoked        bool
smoking_status_never smoked           bool
smoking_status_smokes                 bool
dtype: object
```

8. Make predictions

```
y_pred = model.predict(X_test)
```

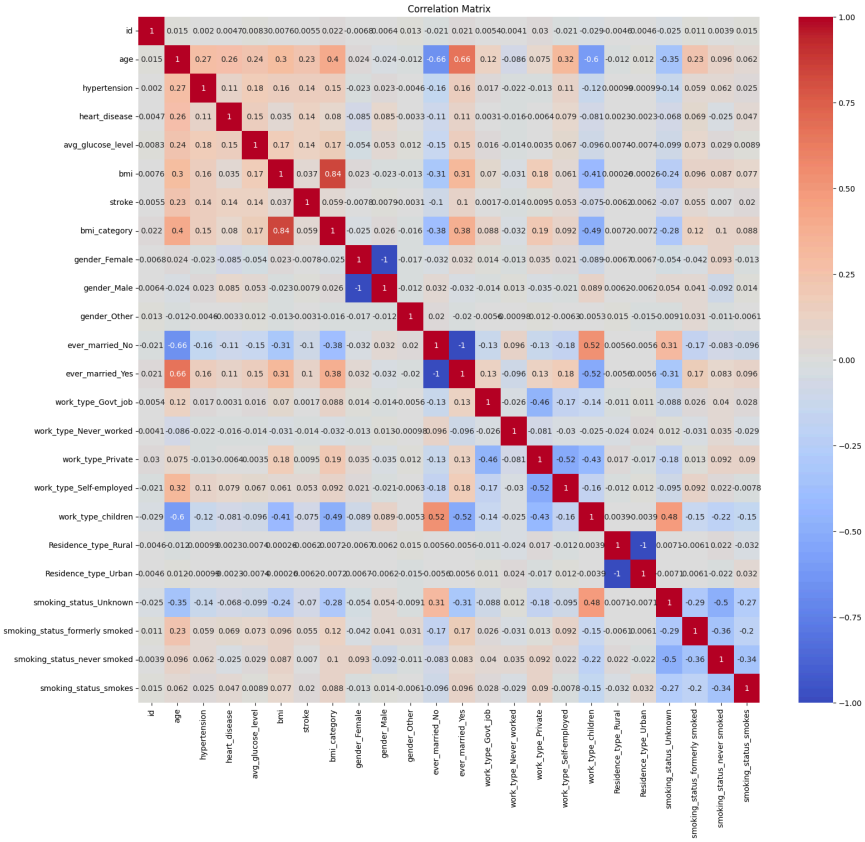
9. Model eveluation

```
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
```

```
Mean Squared Error: 0.04931760145569101
```

10. Visualization

```
# Correlation Heatmap
plt.figure(figsize=(18, 16))
correlation_matrix = stroke_datasetv2.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```




```
#Feature Importance
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)

feature_importances = pd.DataFrame(rf_model.feature_importances_,
                                   index = X_train.columns,
                                   columns=['importance']).sort_values('importance', ascending=False)

plt.figure(figsize=(12, 6))
sns.barplot(x=feature_importances['importance'], y=feature_importances.index)
plt.title('Feature Importance')
plt.show()
```