project

October 30, 2024

[1]:
```python
import kagglehub

# Download latest version
path = kagglehub.dataset_download("fedesoriano/stroke-prediction-dataset")

print("Path to dataset files:", path)
```

```
Downloading from
https://www.kaggle.com/api/v1/datasets/download/fedesoriano/stroke-prediction-
dataset?dataset_version_number=1…

100%|      | 67.4k/67.4k [00:00<00:00, 1.16MB/s]

Extracting files…
Path to dataset files:
C:\Users\Spandana\.cache\kagglehub\datasets\fedesoriano\stroke-prediction-
dataset\versions\1
```

[6]:
```python
import pandas as pd


df = pd.read_csv('healthcare-dataset-stroke-data.csv')
```

[7]:
```python
df.head()
```

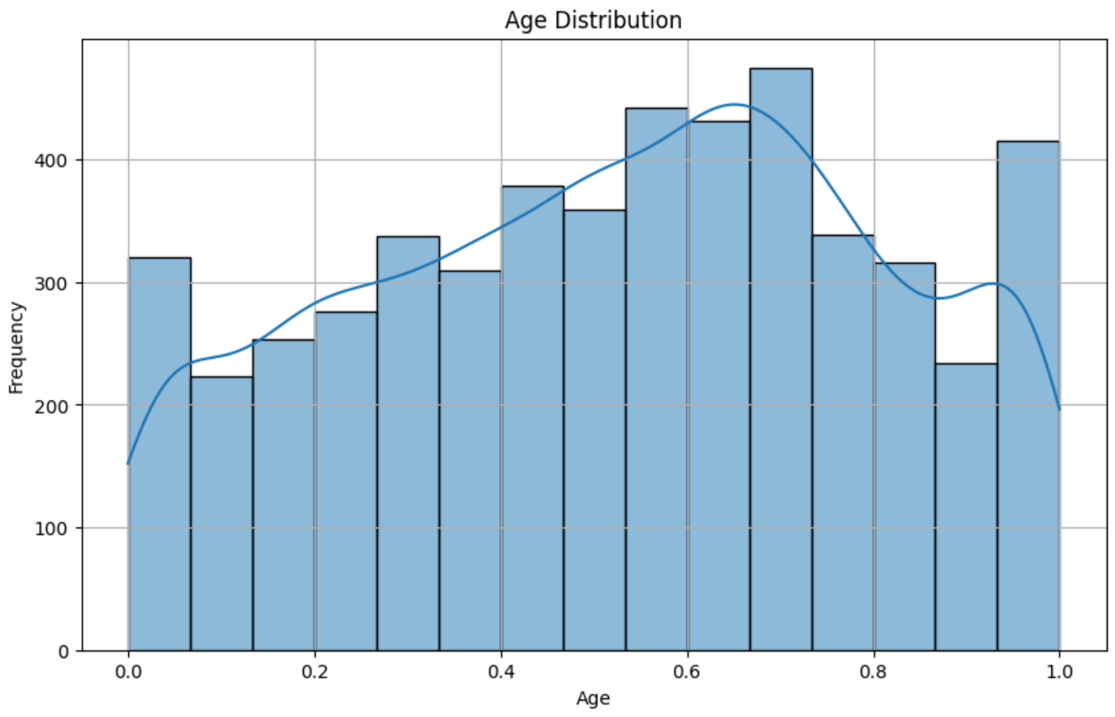
[7]:
```
      id  gender   age  hypertension  heart_disease ever_married  \
0   9046    Male  67.0             0              1          Yes
1  51676  Female  61.0             0              0          Yes
2  31112    Male  80.0             0              1          Yes
3  60182  Female  49.0             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
```

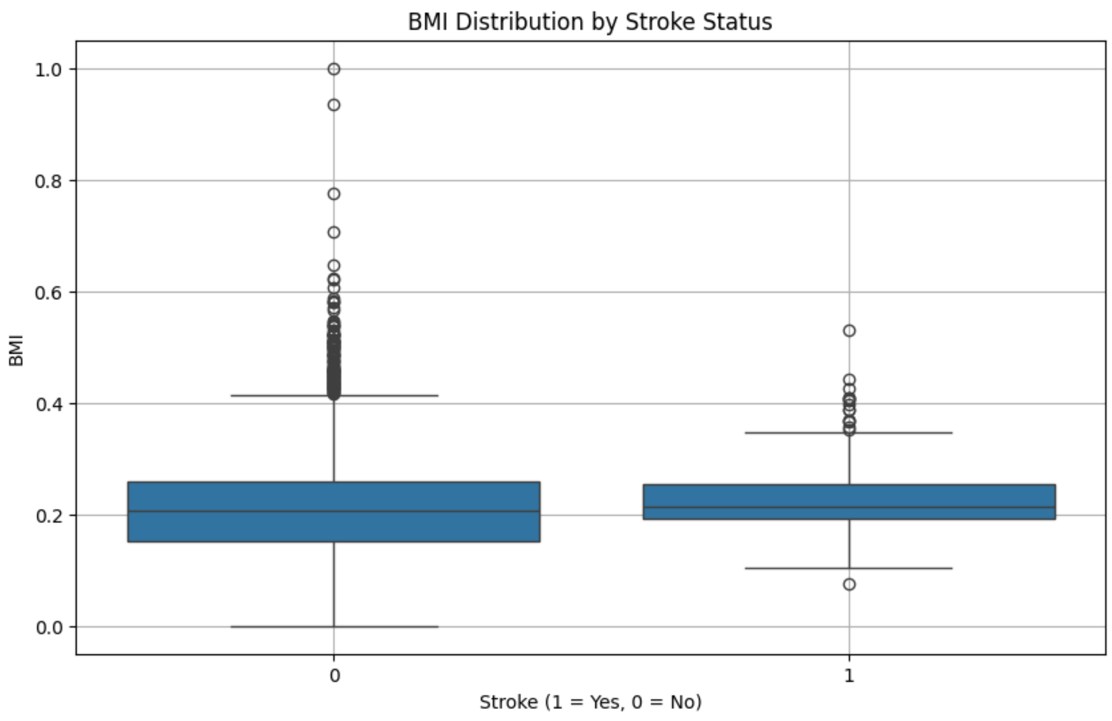
[8]:
```
# 1. Handling missing values
# For simplicity, we will fill missing BMI values with the mean of the column
df['bmi'].fillna(df['bmi'].mean(), inplace=True)

# 2. Converting categorical variables into numerical format
# Using one-hot encoding for categorical variables
df = pd.get_dummies(df, columns=['gender', 'ever_married', 'work_type',␣
 ↪'Residence_type', 'smoking_status'], drop_first=True)

# 3. Feature scaling (optional, depending on your model)
# Normalizing numerical features using Min-Max scaling
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
scaled_features = scaler.fit_transform(df[['age', 'hypertension',␣
 ↪'heart_disease', 'avg_glucose_level', 'bmi']])
scaled_df = pd.DataFrame(scaled_features, columns=['age', 'hypertension',␣
 ↪'heart_disease', 'avg_glucose_level', 'bmi'])

# 4. Concatenate the scaled features back to the DataFrame
df = pd.concat([df.drop(columns=['age', 'hypertension', 'heart_disease',␣
 ↪'avg_glucose_level', 'bmi']), scaled_df], axis=1)
```

```
C:\Users\Spandana\AppData\Local\Temp\ipykernel_2028\3310433539.py:3:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


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

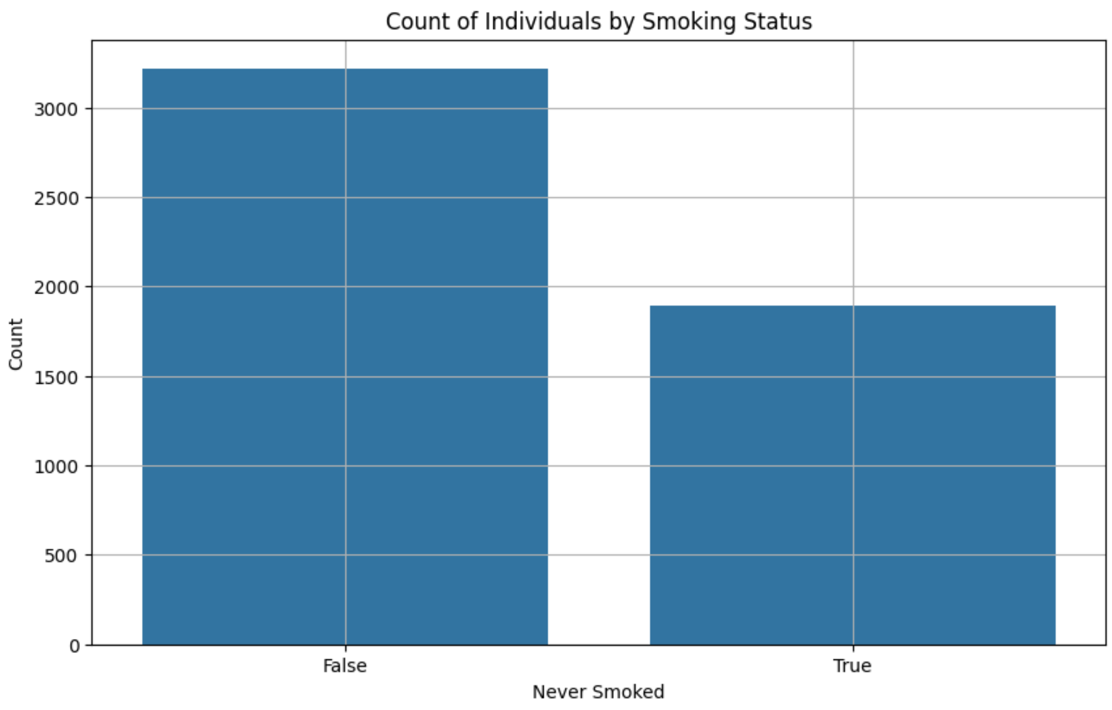
[10]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Histogram for age
plt.figure(figsize=(10, 6))
sns.histplot(df['age'], bins=15, kde=True)
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.grid()
plt.show()
```



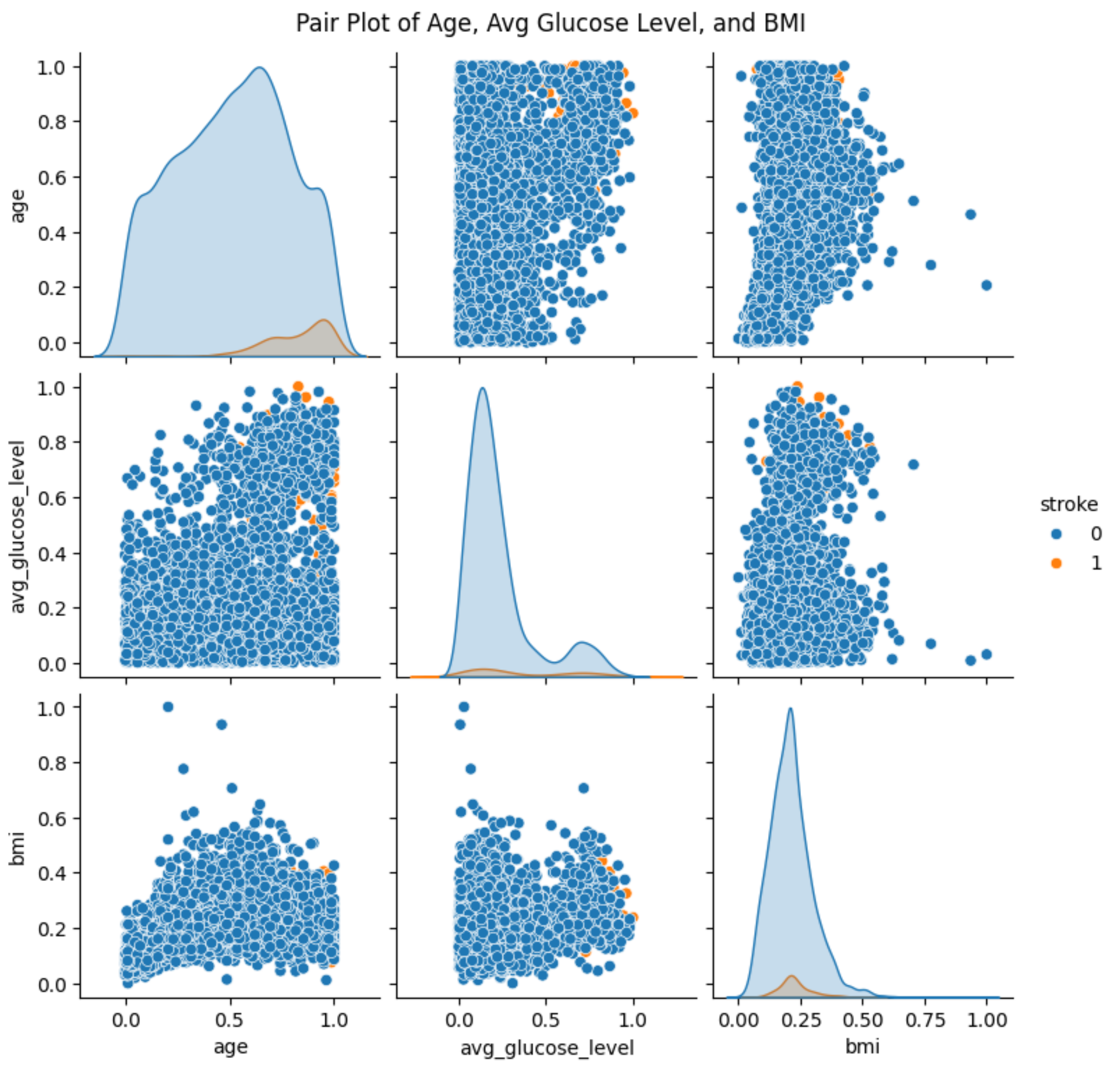
[11]:
```
# Box plot for BMI by stroke status
plt.figure(figsize=(10, 6))
sns.boxplot(x='stroke', y='bmi', data=df)
plt.title('BMI Distribution by Stroke Status')
plt.xlabel('Stroke (1 = Yes, 0 = No)')
plt.ylabel('BMI')
plt.grid()
plt.show()
```



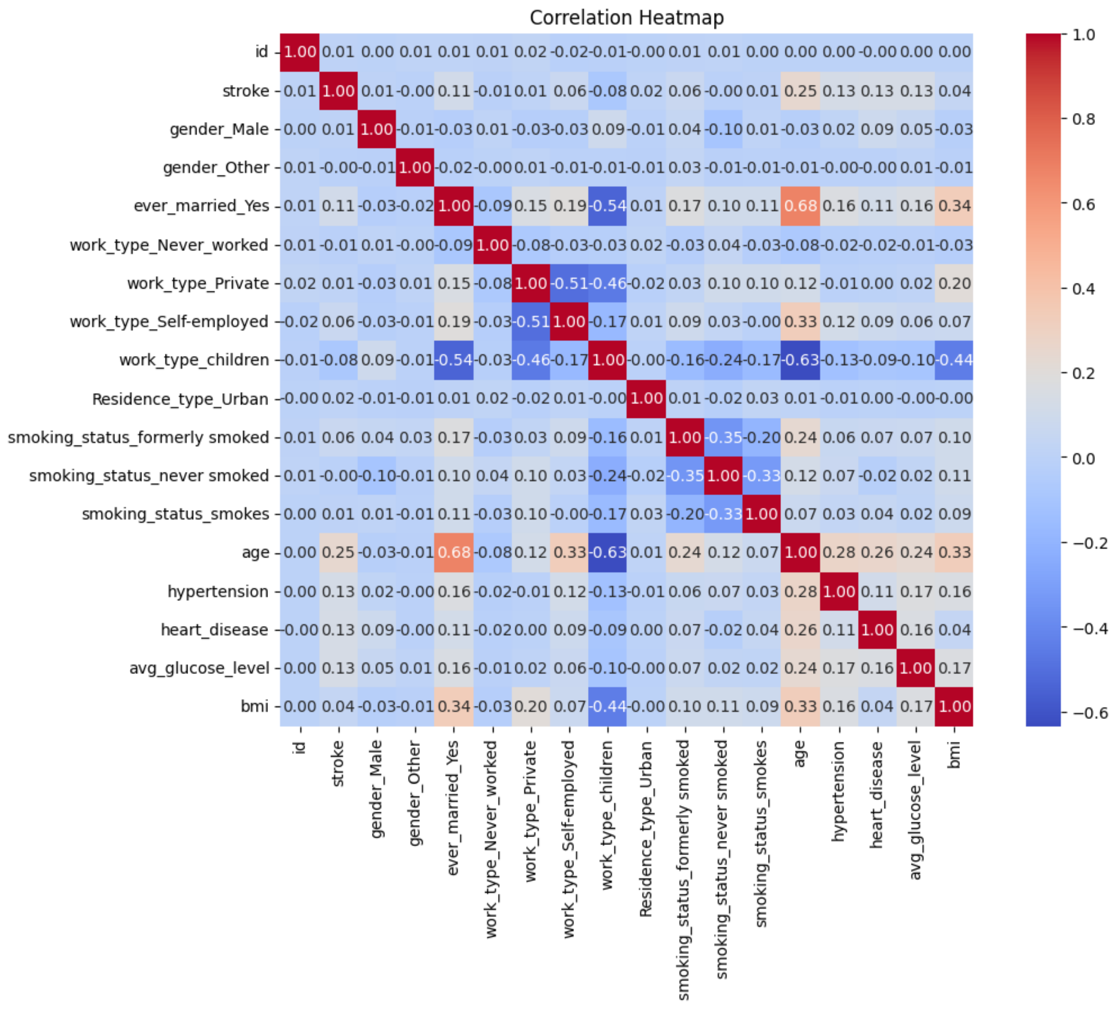
```
[12]: # Count plot for smoking status
plt.figure(figsize=(10, 6))
sns.countplot(x='smoking_status_never smoked', data=df)
plt.title('Count of Individuals by Smoking Status')
plt.xlabel('Never Smoked')
plt.ylabel('Count')
plt.grid()
plt.show()
```



[13]:
```
# Pair plot to visualize relationships
sns.pairplot(df, hue='stroke', vars=['age', 'avg_glucose_level', 'bmi'])
plt.suptitle('Pair Plot of Age, Avg Glucose Level, and BMI', y=1.02)
plt.show()
```



```
[14]: # Heatmap of correlation matrix
      plt.figure(figsize=(12, 8))
      correlation_matrix = df.corr()
      sns.heatmap(correlation_matrix, annot=True, fmt='.2f', cmap='coolwarm',␣
       ↪square=True)
      plt.title('Correlation Heatmap')
      plt.show()
```


Correlation Heatmap
id
stroke
gender_Male
gender_Other
ever_married_Yes
work_type_Never_worked
work_type_Private
work_type_Self-employed
work_type_children
Residence_type_Urban
smoking_status_formerly smoked
smoking_status_never smoked
smoking_status_smokes
age
hypertension
heart_disease
avg_glucose_level
bmi
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

[ ]:

```

```