In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_selection import mutual_info_classif
from sklearn.preprocessing import LabelEncoder
```

In [5]:
```
data=pd.read_csv('brain_stroke/full_data.csv')
data.head()
```

Out[5]:

	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type
0	Male	67.0	0	1	Yes	Private	Urban
1	Male	80.0	0	1	Yes	Private	Rural
2	Female	49.0	0	0	Yes	Private	Urban
3	Female	79.0	1	0	Yes	Self-employed	Rural
4	Male	81.0	0	0	Yes	Private	Urban

In [6]:
```
data.describe()
```

Out[6]:

	age	hypertension	heart_disease	avg_glucose_level	bmi	s
count	4981.000000	4981.000000	4981.000000	4981.000000	4981.000000	4981.0(
mean	43.419859	0.096165	0.055210	105.943562	28.498173	0.04
std	22.662755	0.294848	0.228412	45.075373	6.790464	0.2'
min	0.080000	0.000000	0.000000	55.120000	14.000000	0.0(
25%	25.000000	0.000000	0.000000	77.230000	23.700000	0.0(
50%	45.000000	0.000000	0.000000	91.850000	28.100000	0.0(
75%	61.000000	0.000000	0.000000	113.860000	32.600000	0.0(
max	82.000000	1.000000	1.000000	271.740000	48.900000	1.0(

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4981 entries, 0 to 4980
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   gender             4981 non-null   object 
 1   age                4981 non-null   float64
 2   hypertension       4981 non-null   int64  
 3   heart_disease      4981 non-null   int64  
 4   ever_married       4981 non-null   object 
 5   work_type          4981 non-null   object 
 6   Residence_type     4981 non-null   object 
 7   avg_glucose_level  4981 non-null   float64
 8   bmi                4981 non-null   float64
 9   smoking_status     4981 non-null   object 
 10  stroke             4981 non-null   int64  
dtypes: float64(3), int64(3), object(5)
memory usage: 428.2+ KB
```

In [8]:
```
data.duplicated()
```

Out[8]:
```
0       False
1       False
2       False
3       False
4       False
        ...  
4976    False
4977    False
4978    False
4979    False
4980    False
Length: 4981, dtype: bool
```

In [9]:
```
data.isnull().sum()
```

Out[9]:
```
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
```

In [10]:
```
data.columns
```

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

In [11]:
```
data.shape
```

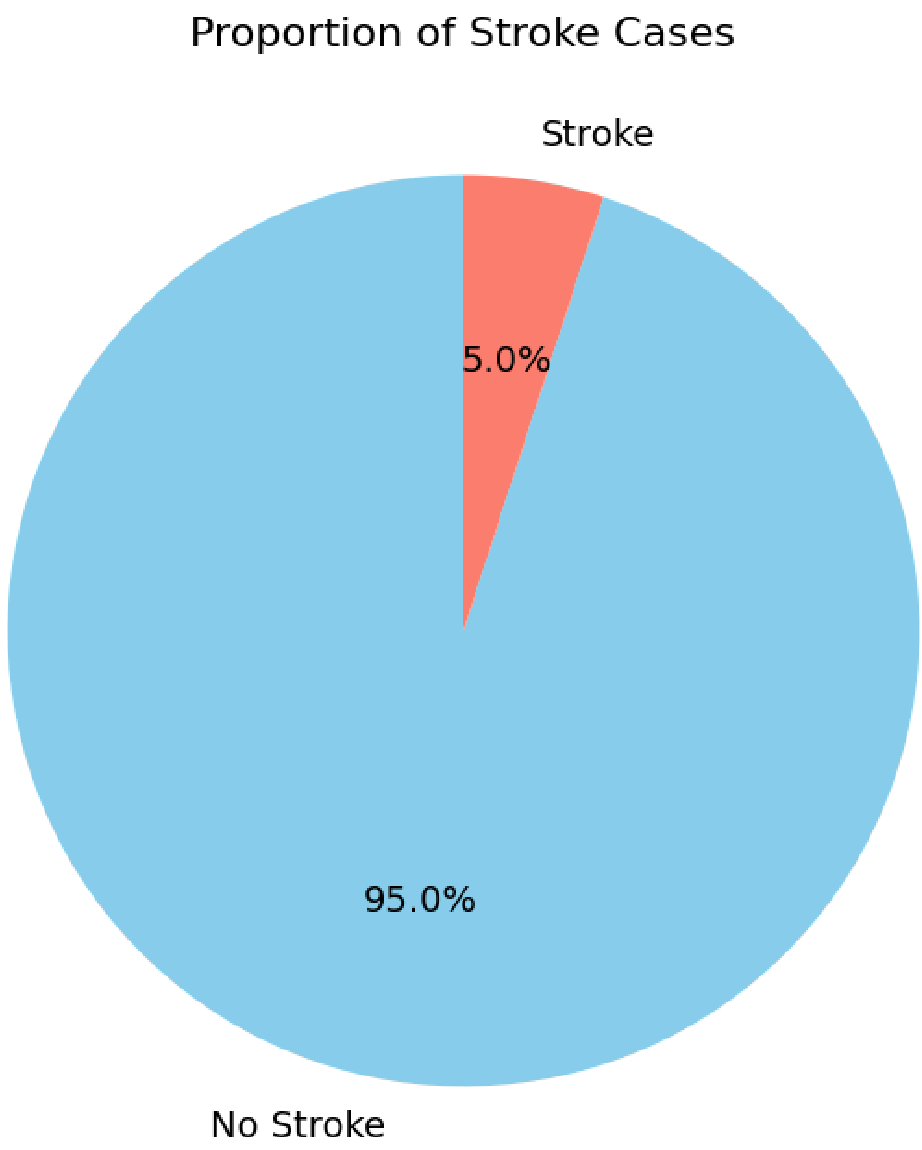
Out[11]:
```
(4981, 11)
```

In [12]:
```
categorical_cols = data.select_dtypes(include=['object']).columns
label_encoders = {}

for col in categorical_cols:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le

X = data.drop("stroke", axis=1)
y = data["stroke"]
mutual_info = mutual_info_classif(X, y, discrete_features=True)

feature_scores = pd.DataFrame({'Feature': X.columns, 'Score': mutual_info})
feature_scores = feature_scores.sort_values(by="Score", ascending=False)

print("Top Features Based on MRMR:")
print(feature_scores)
```

```
Top Features Based on MRMR:
             Feature     Score
7  avg_glucose_level  0.173985
1                age  0.045305
8                bmi  0.035911
4       ever_married  0.006953
5          work_type  0.006769
2       hypertension  0.006322
3      heart_disease  0.005921
9     smoking_status  0.002774
6     Residence_type  0.000136
0             gender  0.000039
```

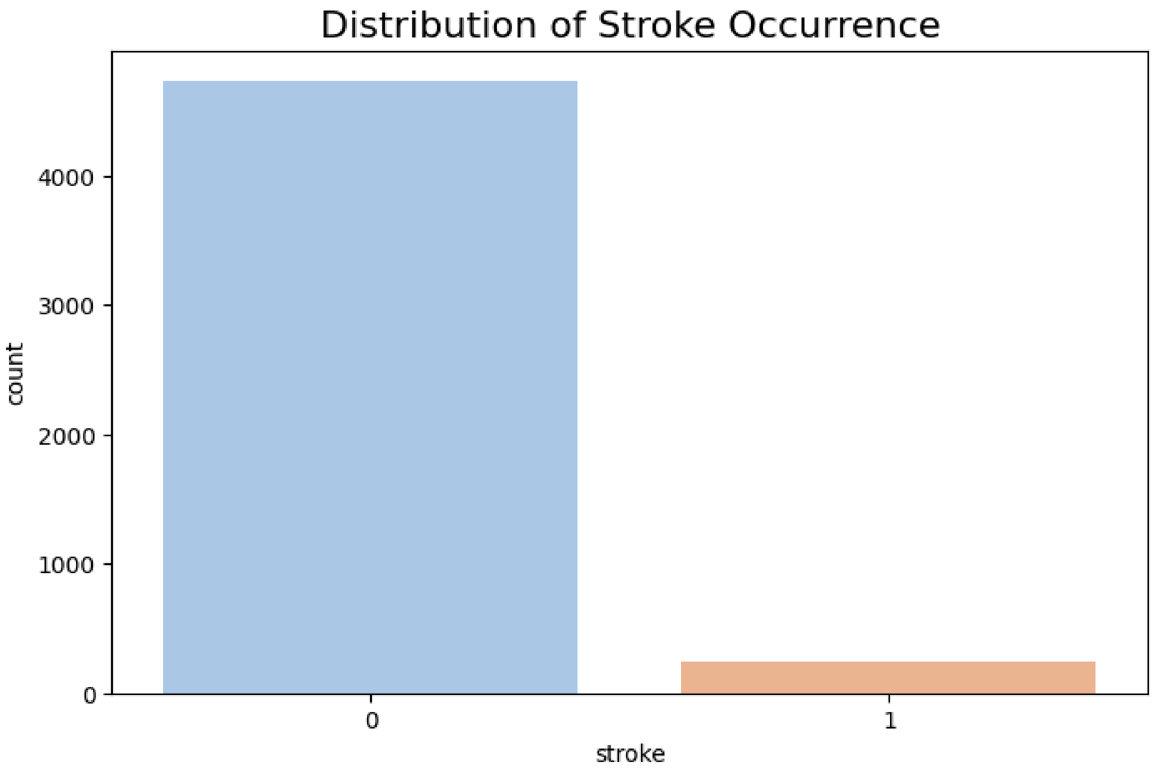
```
c:\Users\Dave Pooja\AppData\Local\Programs\Python\Python312\Lib\site-packages\skl
earn\metrics\cluster\_supervised.py:66: UserWarning: Clustering metrics expects d
iscrete values but received continuous values for label, and binary values for ta
rget
  warnings.warn(msg, UserWarning)
c:\Users\Dave Pooja\AppData\Local\Programs\Python\Python312\Lib\site-packages\skl
earn\metrics\cluster\_supervised.py:66: UserWarning: Clustering metrics expects d
iscrete values but received continuous values for label, and binary values for ta
rget
  warnings.warn(msg, UserWarning)
c:\Users\Dave Pooja\AppData\Local\Programs\Python\Python312\Lib\site-packages\skl
earn\metrics\cluster\_supervised.py:66: UserWarning: Clustering metrics expects d
iscrete values but received continuous values for label, and binary values for ta
rget
  warnings.warn(msg, UserWarning)
```

In [13]:
```
stroke_counts = data['stroke'].value_counts()
labels = ['No Stroke', 'Stroke']

# Plot pie chart
plt.figure(figsize=(8, 8))
plt.pie(stroke_counts, labels=labels, autopct='%1.1f%%', colors=['skyblue', 'sal
plt.title("Proportion of Stroke Cases", fontsize=16)
plt.show()
```



In [14]:
```
plt.figure(figsize=(8, 5))
sns.countplot(data=data, x='stroke', palette='pastel')
plt.title("Distribution of Stroke Occurrence", fontsize=16)
plt.show()
```

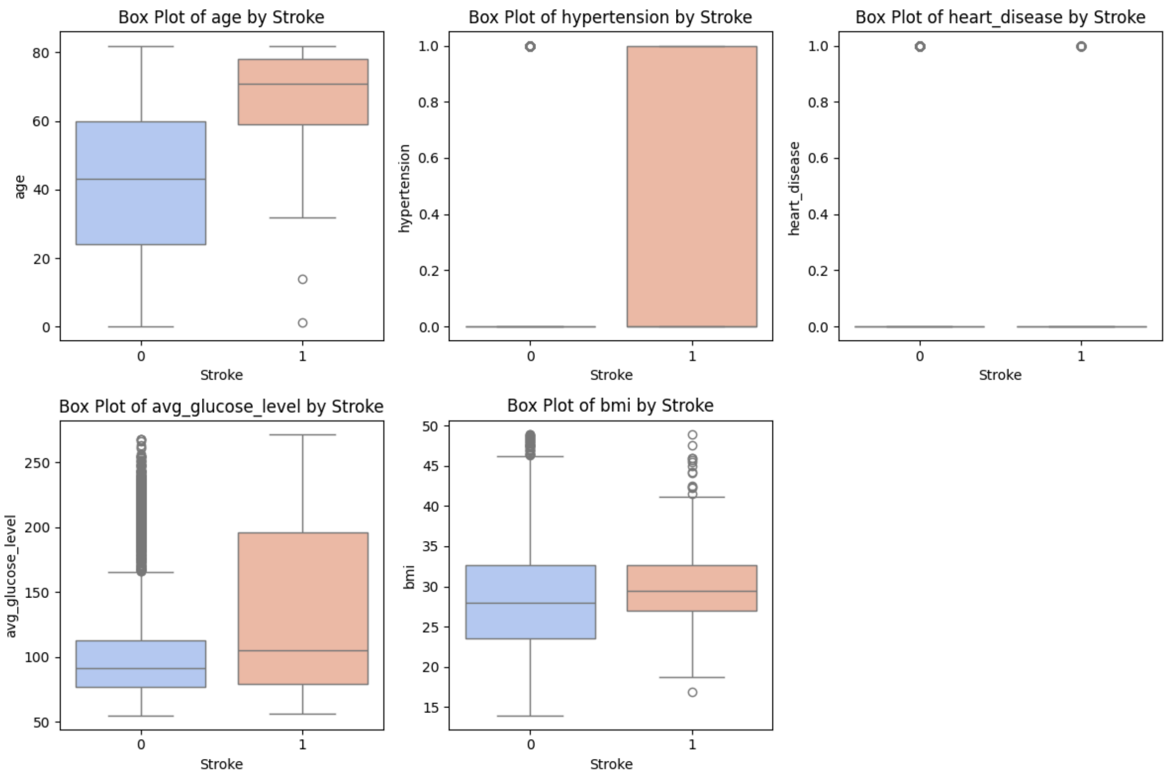
```
C:\conda_temp\ipykernel_45072\574069809.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.countplot(data=data, x='stroke', palette='pastel')
```



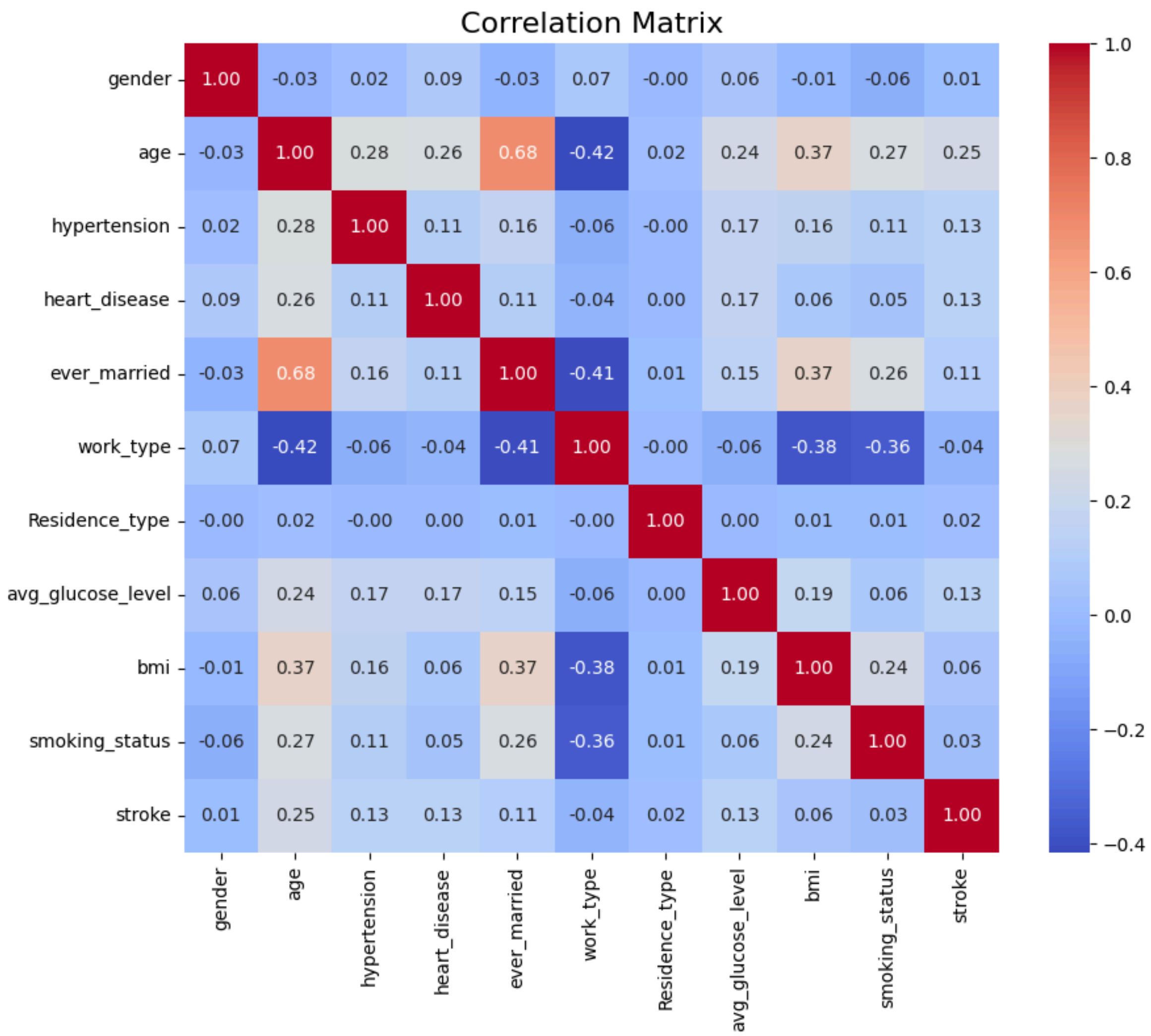
In [17]:

```
features = ["age", "hypertension", "heart_disease", "avg_glucose_level", "bmi"]
plt.figure(figsize=(12, 8))
for i, feature in enumerate(features, 1):
    plt.subplot(2, 3, i)
    sns.boxplot(data=data, x="stroke", y=feature, palette="coolwarm")
    plt.title(f"Box Plot of {feature} by Stroke")
    plt.xlabel("Stroke")
    plt.ylabel(feature)

plt.tight_layout()
plt.show()
```

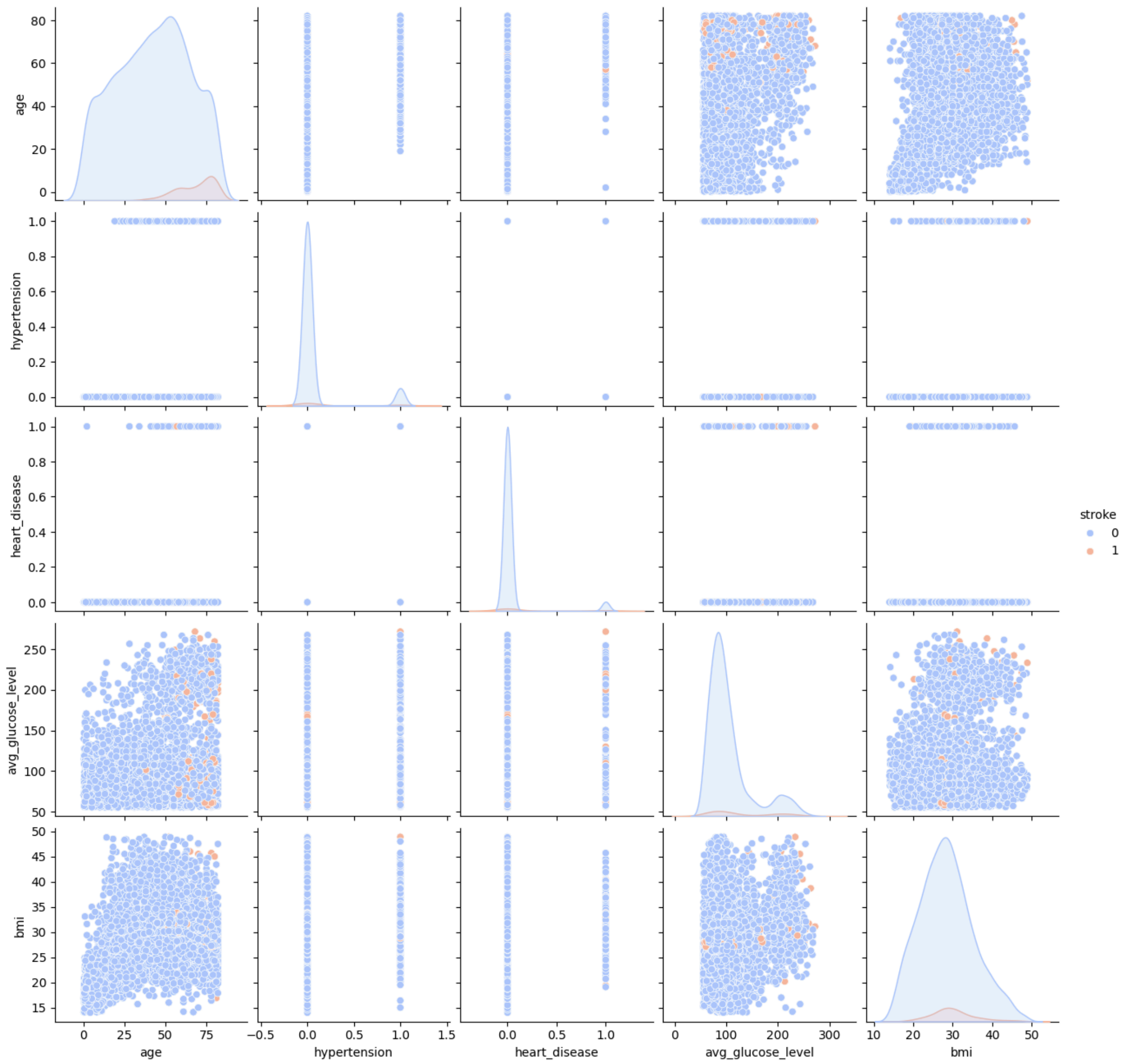
```
C:\conda_temp\ipykernel_45072\3727190078.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.boxplot(data=data, x="stroke", y=feature, palette="coolwarm")
C:\conda_temp\ipykernel_45072\3727190078.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.boxplot(data=data, x="stroke", y=feature, palette="coolwarm")
C:\conda_temp\ipykernel_45072\3727190078.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.boxplot(data=data, x="stroke", y=feature, palette="coolwarm")
C:\conda_temp\ipykernel_45072\3727190078.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.boxplot(data=data, x="stroke", y=feature, palette="coolwarm")
C:\conda_temp\ipykernel_45072\3727190078.py:5: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  sns.boxplot(data=data, x="stroke", y=feature, palette="coolwarm")
```



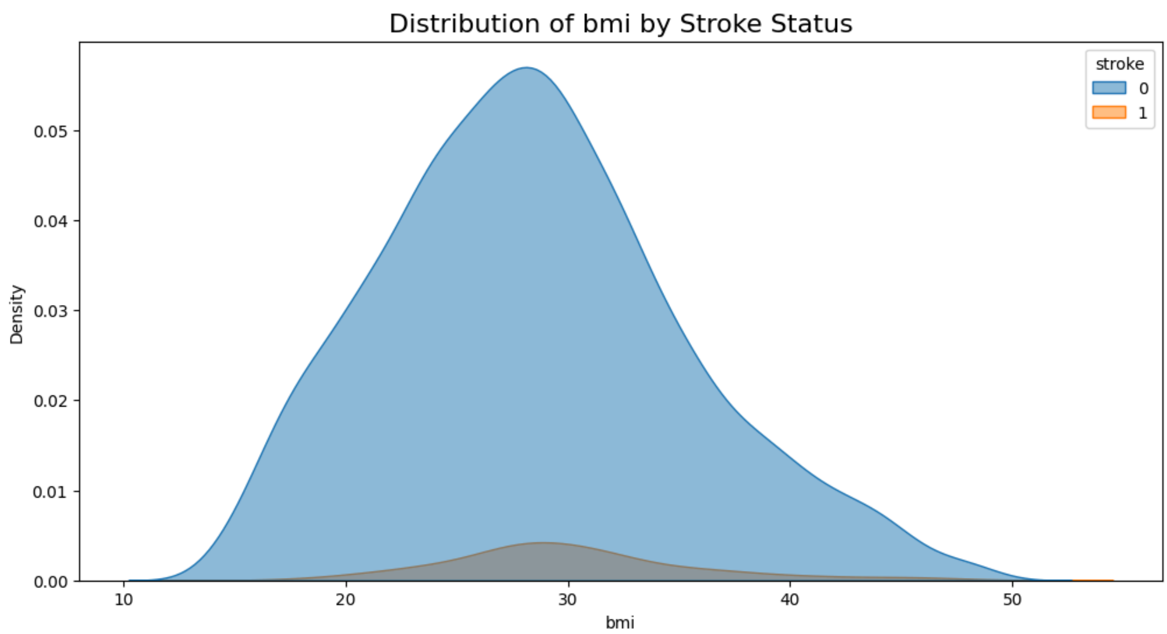
In [15]:
```
plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Matrix", fontsize=16)
plt.show()
```

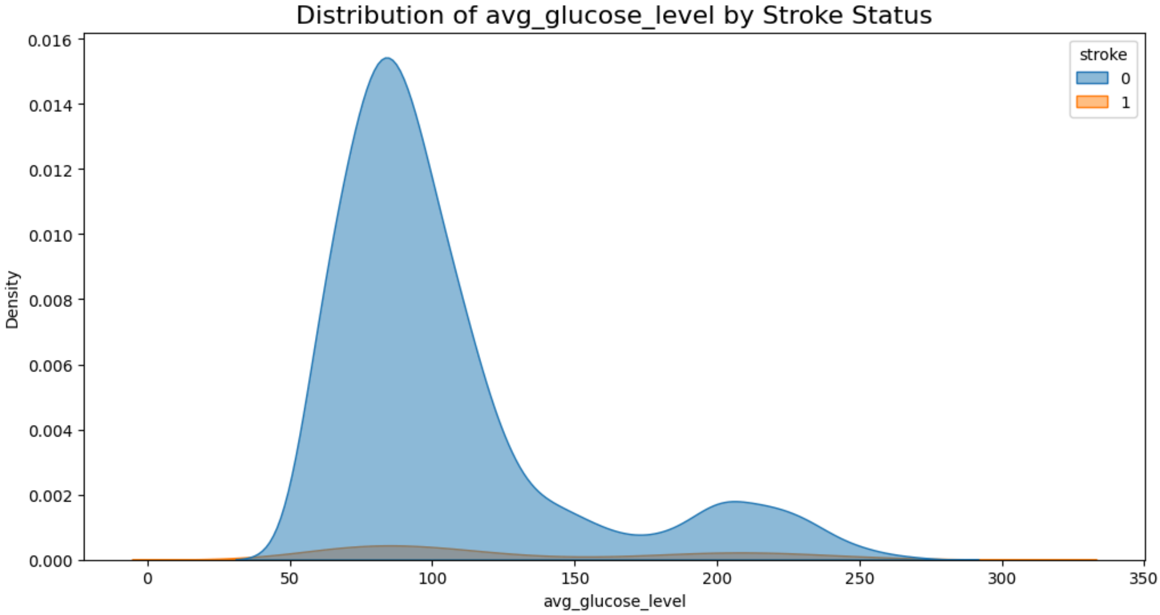


In [16]:
```
selected_features = ["age", "hypertension", "heart_disease", "avg_glucose_level"
sns.pairplot(data[selected_features], hue="stroke", palette="coolwarm", diag_kin
plt.show()
```



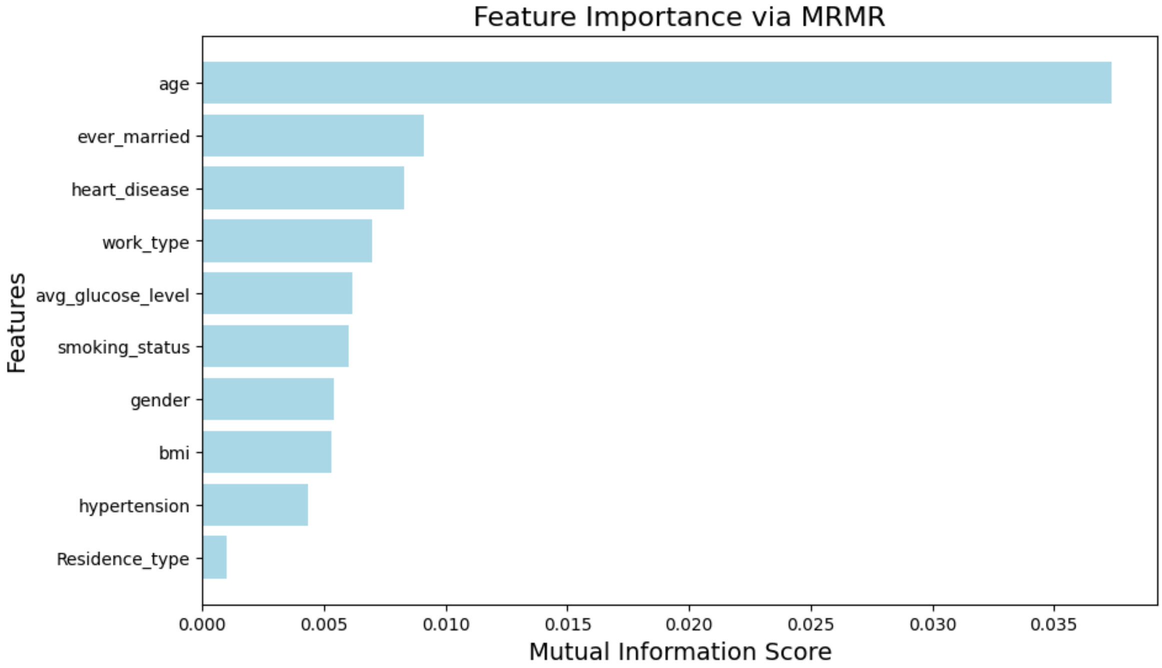
In [22]:
```
for column in ["bmi", "avg_glucose_level"]:
    plt.figure(figsize=(12, 6))
    sns.kdeplot(data=data, x=column, hue="stroke", fill=True, alpha=0.5)
    plt.title(f"Distribution of {column} by Stroke Status", fontsize=16)
    plt.show()
```





In [24]:
```
X = data.drop("stroke", axis=1)
y = data["stroke"]


mutual_info = mutual_info_classif(X, y, discrete_features='auto')


feature_scores = pd.DataFrame({'Feature': X.columns, 'Score': mutual_info}).sort

print("Top Features Based on MRMR (Mutual Information):")
print(feature_scores)


plt.figure(figsize=(10, 6))
plt.barh(feature_scores['Feature'], feature_scores['Score'], color='lightblue')
plt.xlabel("Mutual Information Score", fontsize=14)
plt.ylabel("Features", fontsize=14)
plt.title("Feature Importance via MRMR", fontsize=16)
plt.gca().invert_yaxis()
plt.show()
```

```
Top Features Based on MRMR (Mutual Information):
             Feature     Score
1                age  0.037389
4       ever_married  0.009116
3      heart_disease  0.008281
5          work_type  0.006998
7  avg_glucose_level  0.006142
9     smoking_status  0.006036
0             gender  0.005417
8                bmi  0.005290
2       hypertension  0.004366
6     Residence_type  0.001023
```



Result

Features with High Importance:

Features such as age, hypertension, and avg_glucose_level are likely to rank high, as they are strong predictors of stroke according to medical literature. Categorical features like smoking_status and work_type may show lower importance if their relationship to stroke is weak.

MRMR's Contribution:

By prioritizing maximum relevance (e.g., strong correlation with stroke) and minimum redundancy (avoiding features that overlap in information), MRMR effectively narrows down a set of key predictive features.

Interpretation:

These results can guide feature selection for building more efficient machine learning models, reducing dimensionality and noise, and improving predictive performance.