# Project 2

Code by Mudit Arora for DAT 301 (Fall 2023)

## Background and Problem Definition

The heart disease dataset has been taken from Kaggle which consists of various factors that causes Heart Disease. I'll be using Logistic Regression to understand which of the factors are more accurate.

Logistic Regression is a statistical method used for binary classification, which predicts the probability of a binary outcome (1/0, True/False, Yes/No) based on one or more independent variables. It is widely used due to its simplicity, interpretability, and efficacy in various applications, especially in fields like medicine, social sciences, and machine learning.

## Data Wrangling, Munging, and Cleaning

### Necessary Libraries and packages

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, roc_auc
```

In [2]:
```
# Importing dataset
df = pd.read_csv("heartdisease.csv")
# First 5 records of the dataset
df.head()
```

Out[2]:

| | male | age | education | currentSmoker | cigsPerDay | BPMeds | prevalentStroke | prevalentHy |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 39 | 4.0 | 0 | 0.0 | 0.0 | 0 | |
| **1** | 0 | 46 | 2.0 | 0 | 0.0 | 0.0 | 0 | |
| **2** | 1 | 48 | 1.0 | 1 | 20.0 | 0.0 | 0 | |
| **3** | 0 | 61 | 3.0 | 1 | 30.0 | 0.0 | 0 | |
| **4** | 0 | 46 | 3.0 | 1 | 23.0 | 0.0 | 0 | |

In [3]:

```
# Last 5 records of the dataset
df.tail(5)
```

Out[3]:

| | male | age | education | currentSmoker | cigsPerDay | BPMeds | prevalentStroke | prevaler |
|---|---|---|---|---|---|---|---|---|
| **4233** | 1 | 50 | 1.0 | 1 | 1.0 | 0.0 | 0 | |
| **4234** | 1 | 51 | 3.0 | 1 | 43.0 | 0.0 | 0 | |
| **4235** | 0 | 48 | 2.0 | 1 | 20.0 | NaN | 0 | |
| **4236** | 0 | 44 | 1.0 | 1 | 15.0 | 0.0 | 0 | |
| **4237** | 0 | 52 | 2.0 | 0 | 0.0 | 0.0 | 0 | |

In [4]:

```
# Getting relevant info of the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4238 entries, 0 to 4237
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   male             4238 non-null   int64  
 1   age              4238 non-null   int64  
 2   education        4133 non-null   float64
 3   currentSmoker    4238 non-null   int64  
 4   cigsPerDay       4209 non-null   float64
 5   BPMeds           4185 non-null   float64
 6   prevalentStroke  4238 non-null   int64  
 7   prevalentHyp     4238 non-null   int64  
 8   diabetes         4238 non-null   int64  
 9   totChol          4188 non-null   float64
 10  sysBP            4238 non-null   float64
 11  diaBP            4238 non-null   float64
 12  BMI              4219 non-null   float64
 13  heartRate        4237 non-null   float64
 14  glucose          3850 non-null   float64
 15  TenYearCHD       4238 non-null   int64  
dtypes: float64(9), int64(7)
memory usage: 529.9 KB
```

In [5]:
```
# Getting description of the dataset
df.describe()
```

Out[5]:

| | male | age | education | currentSmoker | cigsPerDay | BPMeds |
|---|---|---|---|---|---|---|
| **count** | 4238.000000 | 4238.000000 | 4133.000000 | 4238.000000 | 4209.000000 | 4185.000000 |
| **mean** | 0.429212 | 49.584946 | 1.978950 | 0.494101 | 9.003089 | 0.029630 |
| **std** | 0.495022 | 8.572160 | 1.019791 | 0.500024 | 11.920094 | 0.169584 |
| **min** | 0.000000 | 32.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 0.000000 | 42.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 |
| **50%** | 0.000000 | 49.000000 | 2.000000 | 0.000000 | 0.000000 | 0.000000 |
| **75%** | 1.000000 | 56.000000 | 3.000000 | 1.000000 | 20.000000 | 0.000000 |
| **max** | 1.000000 | 70.000000 | 4.000000 | 1.000000 | 70.000000 | 1.000000 |

## Data Cleaning

In [6]:
```
# Checking for null values
df.isnull().sum()
```

Out[6]:
```
male                 0
age                  0
education          105
currentSmoker        0
cigsPerDay          29
BPMeds              53
prevalentStroke      0
prevalentHyp         0
diabetes             0
totChol             50
sysBP                0
diaBP                0
BMI                 19
heartRate            1
glucose            388
TenYearCHD           0
dtype: int64
```

In [7]:
```
# Removing all the null values
df.dropna(axis=0,inplace=True)
```

In [8]:
```
# Minimum value of the dataset
df.min()
```

Out[8]:
```
male                0.00
age                32.00
education           1.00
currentSmoker       0.00
cigsPerDay          0.00
BPMeds              0.00
prevalentStroke     0.00
prevalentHyp        0.00
diabetes            0.00
totChol           113.00
sysBP              83.50
diaBP              48.00
BMI                15.54
heartRate          44.00
glucose            40.00
TenYearCHD          0.00
dtype: float64
```

In [9]:
```
# Maximum value of dataset
df.max()
```

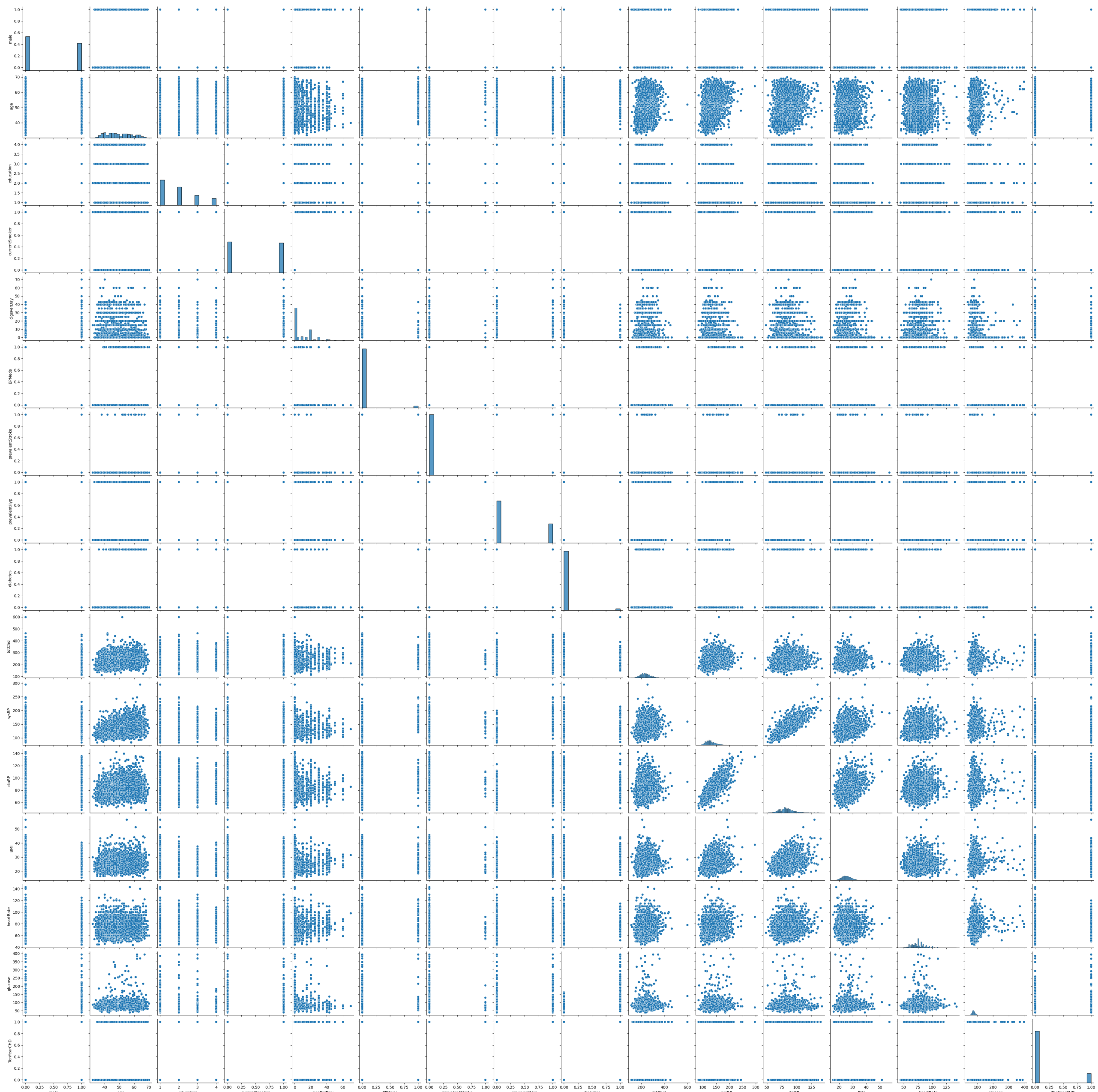
Out[9]:
```
male                1.0
age                70.0
education           4.0
currentSmoker       1.0
cigsPerDay         70.0
BPMeds              1.0
prevalentStroke     1.0
prevalentHyp        1.0
diabetes            1.0
totChol           600.0
sysBP             295.0
diaBP             142.5
BMI                56.8
heartRate         143.0
glucose           394.0
TenYearCHD          1.0
dtype: float64
```

# Exploratory Data Analysis and Data Visualization

In [10]:
```
sns.pairplot(data=df)
```

```
/Users/muditarora/anaconda3/lib/python3.11/site-packages/seaborn/axisgrid.p
y:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

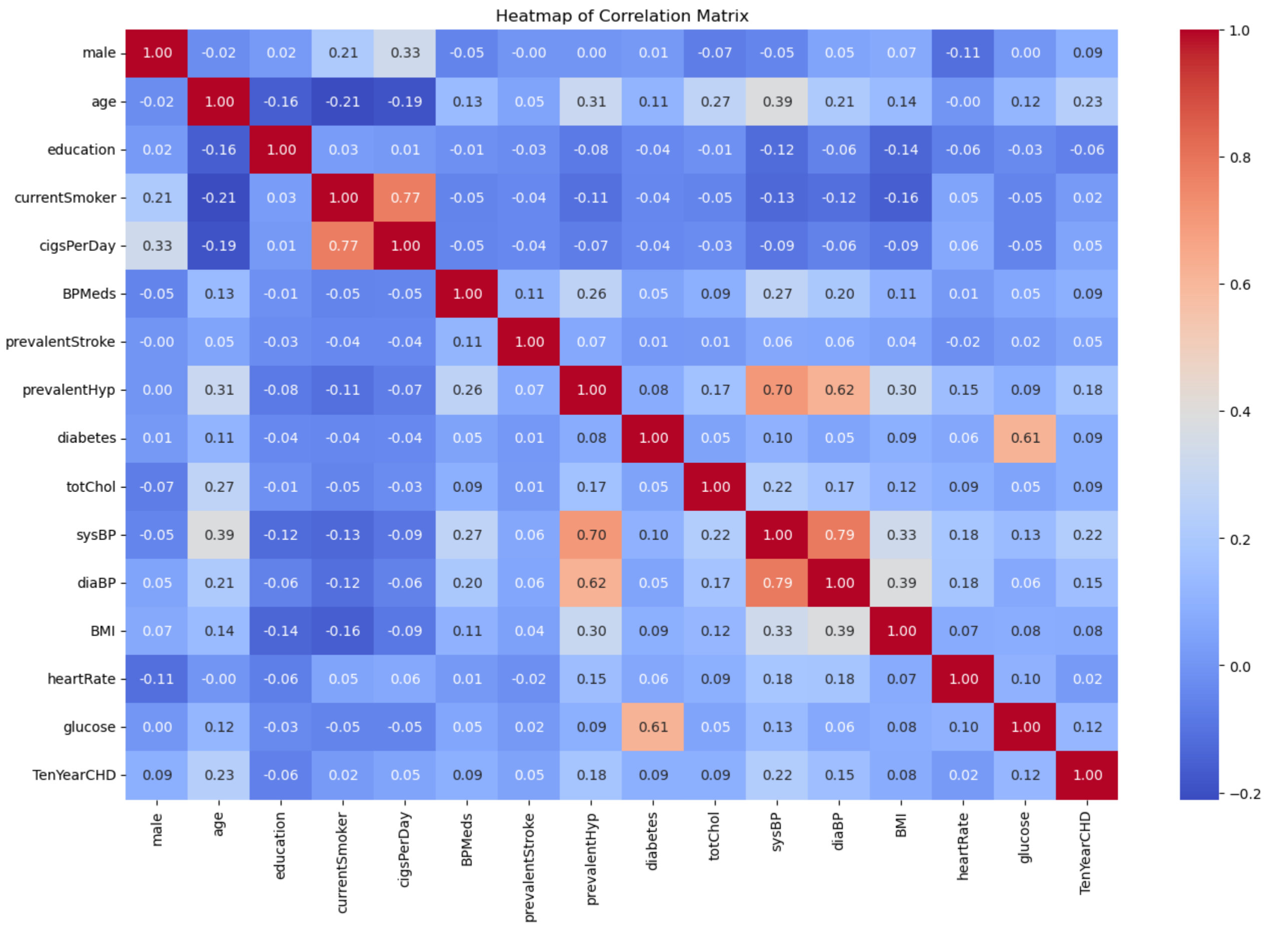
Out[10]:
```
<seaborn.axisgrid.PairGrid at 0x1504f1910>
```



Comparing each column with each other to get a rough idea which are the most relevant factors

In [11]:
```
correlation_matrix = df.corr()

plt.figure(figsize=(16, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Heatmap of Correlation Matrix")
plt.show()
```




Intensity of Color: Represents the strength of the correlation. Warmer colors (red-orange) indicate a positive correlation, whereas cooler colors (blue) signify a negative correlation.

Annotations: Show the actual correlation coefficients, providing precise values for each pair of variables.

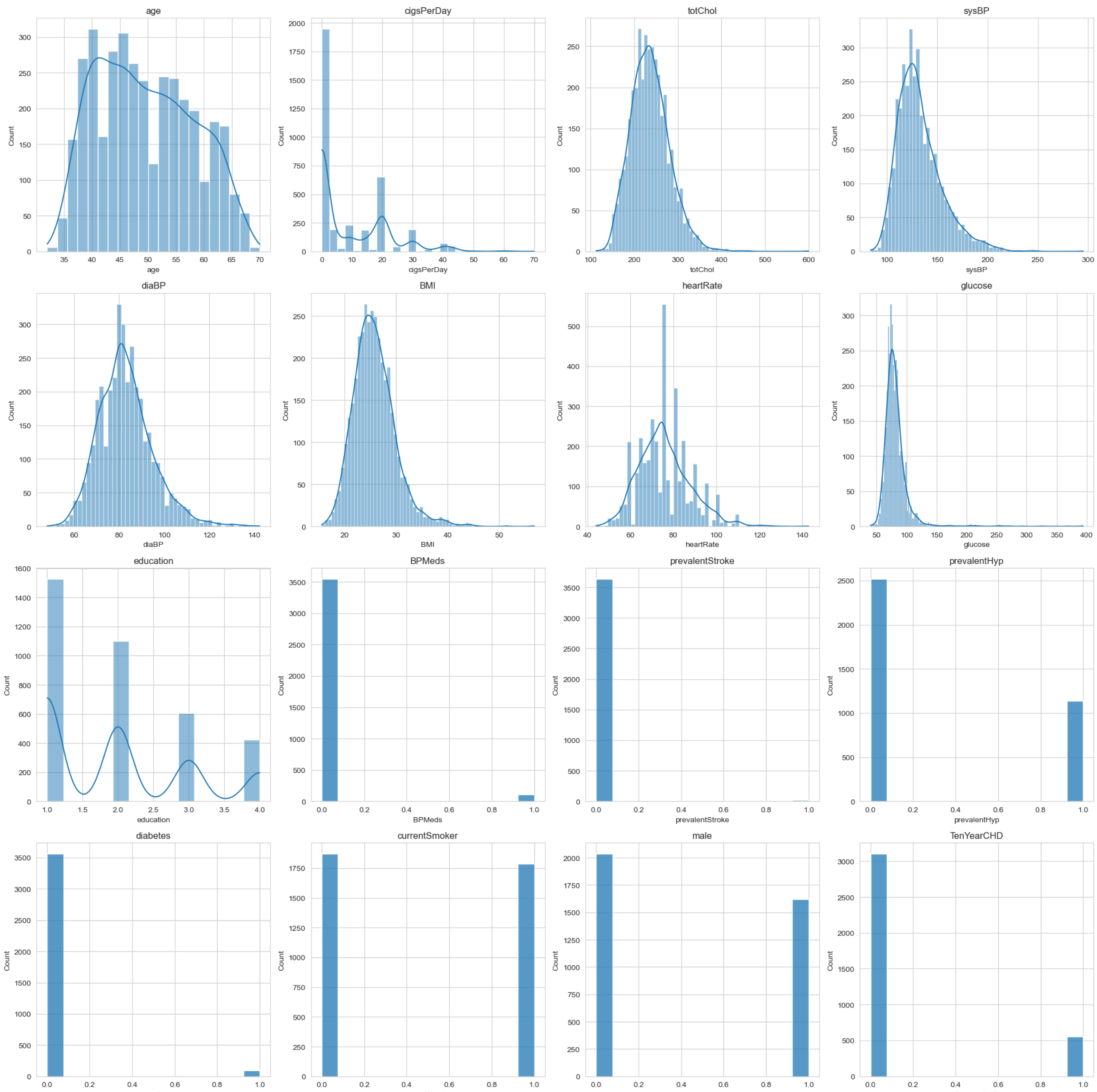
Key Observations:

- High positive correlations are evident between sysBP and diaBP, and between sysBP and prevalentHyp.
- Some variables, like age, have moderate positive correlations with multiple risk factors, such as sysBP, diaBP, totChol, and prevalentHyp.
- The correlation of these variables with TenYearCHD is also notable, though not extremely strong, indicating their relevance in predicting the risk of heart disease.

In [12]:
```
sns.set_style("whitegrid")
fig, axes = plt.subplots(nrows=4, ncols=4, figsize=(20, 20))
cols = ['age', 'cigsPerDay', 'totChol', 'sysBP', 'diaBP', 'BMI', 'heartRate'
        'education', 'BPMeds', 'prevalentStroke', 'prevalentHyp', 'diabetes'

for i, col in enumerate(cols):
    row, col_index = i // 4, i % 4
    sns.histplot(df[col], ax=axes[row, col_index], kde=True if df[col].nunig
    axes[row, col_index].set_title(col)

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



1. Age, CigsPerDay, TotChol, SysBP, DiaBP, BMI, HeartRate, Glucose: These continuous variables show varied distributions.

2. Education, BPMeds, PrevalentStroke, PrevalentHyp, Diabetes, CurrentSmoker, Male, TenYearCHD:

3. Key Observations:

- Risk factors like sysBP, diaBP, and totChol show wide ranges, suggesting variability in these important health indicators.
- Lifestyle factors such as cigsPerDay indicate a significant number of non-smokers or low-frequency smokers.
- The target variable TenYearCHD shows an imbalance with more instances of non-occurrence than occurrence of coronary heart disease in 10 years.

# Logistic Regression

The logistic regression model formula is:

log(p/(1-p)) = B0 + B1X1 + B2X2 +.... BnXn

Where,

- p is the probability of the dependent variable (target) being equal to 1.
- B0, B1, B2,... Bn are the model coefficients.
- X1, X2,.... Xn are the independent variables (features).

In [13]:
```
X = df.drop('TenYearCHD', axis=1)  # Features variable
y = df['TenYearCHD']               # Target variable

# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ran

# Creating and training the logistic regression model
logistic_model = LogisticRegression(max_iter=5000)
logistic_model.fit(X_train, y_train)

y_pred = logistic_model.predict(X_test)

# Model Evaluation
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred, output_dict=True)

# Coefficients
coefficients = np.concatenate([logistic_model.intercept_, logistic_model.coe
coefficients_df = pd.DataFrame(coefficients, index=['Intercept'] + list(X.co

coefficients_df
```

Out[13]:

| | Coefficient |
|---|---|
| **Intercept** | -8.726753 |
| **male** | 0.581196 |
| **age** | 0.066005 |
| **education** | -0.014606 |
| **currentSmoker** | 0.143903 |
| **cigsPerDay** | 0.019316 |
| **BPMeds** | 0.289112 |
| **prevalentStroke** | 1.111227 |
| **prevalentHyp** | 0.188066 |
| **diabetes** | 0.192210 |
| **totChol** | 0.003170 |
| **sysBP** | 0.016850 |
| **diaBP** | -0.001205 |
| **BMI** | 0.003108 |
| **heartRate** | -0.008870 |
| **glucose** | 0.006758 |

The coefficients of the model give insights into the relationship between each feature and the likelihood of having a heart disease in 10 years.

In [14]:
```
probabilities = logistic_model.predict_proba(df.drop('TenYearCHD', axis=1))
p_values = probabilities[:, 1]  # Probabilities of class 1 (TenYearCHD = 1)
p_values = pd.DataFrame(p_values)
p_values
```

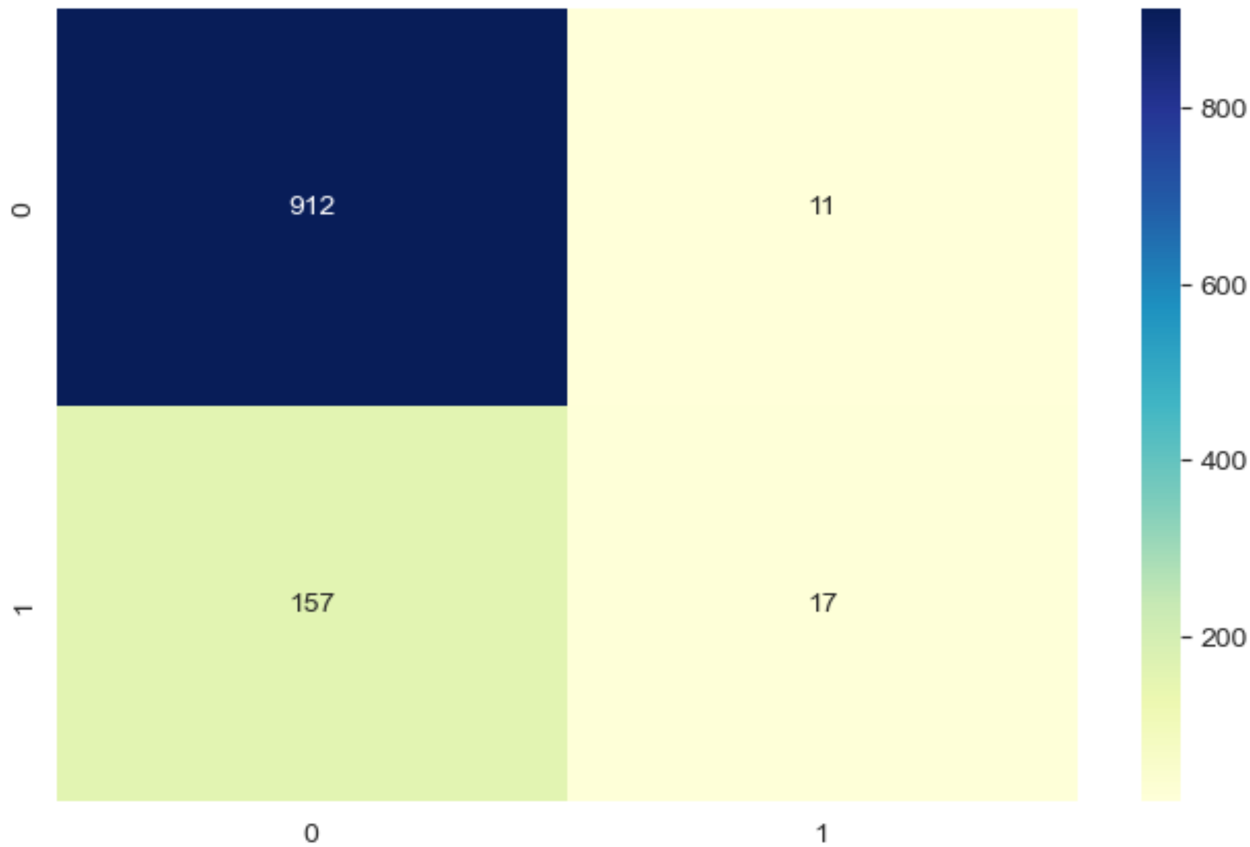
Out[14]:

| | 0 |
|---|---|
| **0** | 0.031819 |
| **1** | 0.038206 |
| **2** | 0.148479 |
| **3** | 0.377950 |
| **4** | 0.094383 |
| **...** | ... |
| **3651** | 0.199943 |
| **3652** | 0.465465 |
| **3653** | 0.369413 |
| **3654** | 0.229149 |
| **3655** | 0.096098 |

3656 rows × 1 columns

In [15]:
```
cm=pd.DataFrame(data=conf_matrix,columns=['Predicted:0','Predicted:1'],index
plt.figure(figsize = (8,5))
sns.heatmap(conf_matrix, annot=True,fmt='d',cmap="YlGnBu")
```

Out[15]:
```
<Axes: >
```




The confusion matrix provides a summary of the prediction results on the test set.

In [16]:
```
class_report_df = pd.DataFrame(class_report).transpose()
class_report_df
```

Out[16]:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| **0** | 0.853134 | 0.988082 | 0.915663 | 923.000000 |
| **1** | 0.607143 | 0.097701 | 0.168317 | 174.000000 |
| **accuracy** | 0.846855 | 0.846855 | 0.846855 | 0.846855 |
| **macro avg** | 0.730138 | 0.542892 | 0.541990 | 1097.000000 |
| **weighted avg** | 0.814116 | 0.846855 | 0.797123 | 1097.000000 |

The classification report includes key metrics like precision, recall, f1-score, and support for both classes (0 and 1). This report helps in understanding the model's performance in terms of correctly predicting each class.

In [17]:
```
cm = confusion_matrix(y_test, y_pred)
TN=cm[0,0]
TP=cm[1,1]
FN=cm[1,0]
FP=cm[0,1]
sensitivity=TP/float(TP+FN)
specificity=TN/float(TN+FP)
print('The acuuracy of the model = TP+TN/(TP+TN+FP+FN) = ',(TP+TN)/float(TP+

'The Missclassification = 1-Accuracy = ',1-((TP+TN)/float(TP+TN+FP+FN)),'\n'

'Sensitivity or True Positive Rate = TP/(TP+FN) = ',TP/float(TP+FN),'\n',

'Specificity or True Negative Rate = TN/(TN+FP) = ',TN/float(TN+FP),'\n',

'Positive Predictive value = TP/(TP+FP) = ',TP/float(TP+FP),'\n',

'Negative predictive Value = TN/(TN+FN) = ',TN/float(TN+FN),'\n')
```

```
The acuuracy of the model = TP+TN/(TP+TN+FP+FN) =  0.8468550592525068
 The Missclassification = 1-Accuracy =  0.15314494074749319
 Sensitivity or True Positive Rate = TP/(TP+FN) =  0.09770114942528736
 Specificity or True Negative Rate = TN/(TN+FP) =  0.9880823401950163
 Positive Predictive value = TP/(TP+FP) =  0.6071428571428571
 Negative predictive Value = TN/(TN+FN) =  0.8531337698783911
```

The negative values are predicted more accurately than the positives.

In [18]:
```
# Calculating the probabilities of the predictions
y_pred_prob=logistic_model.predict_proba(X_test)[:,:]
y_pred_prob_df=pd.DataFrame(data=y_pred_prob, columns=['Prob of no heart dis
y_pred_prob_df.head()
```

Out[18]:

| | Prob of no heart disease (0) | Prob of Heart Disease (1) |
|---|---|---|
| **0** | 0.958792 | 0.041208 |
| **1** | 0.981871 | 0.018129 |
| **2** | 0.560225 | 0.439775 |
| **3** | 0.963730 | 0.036270 |
| **4** | 0.570727 | 0.429273 |

## Plotting the ROC Curve

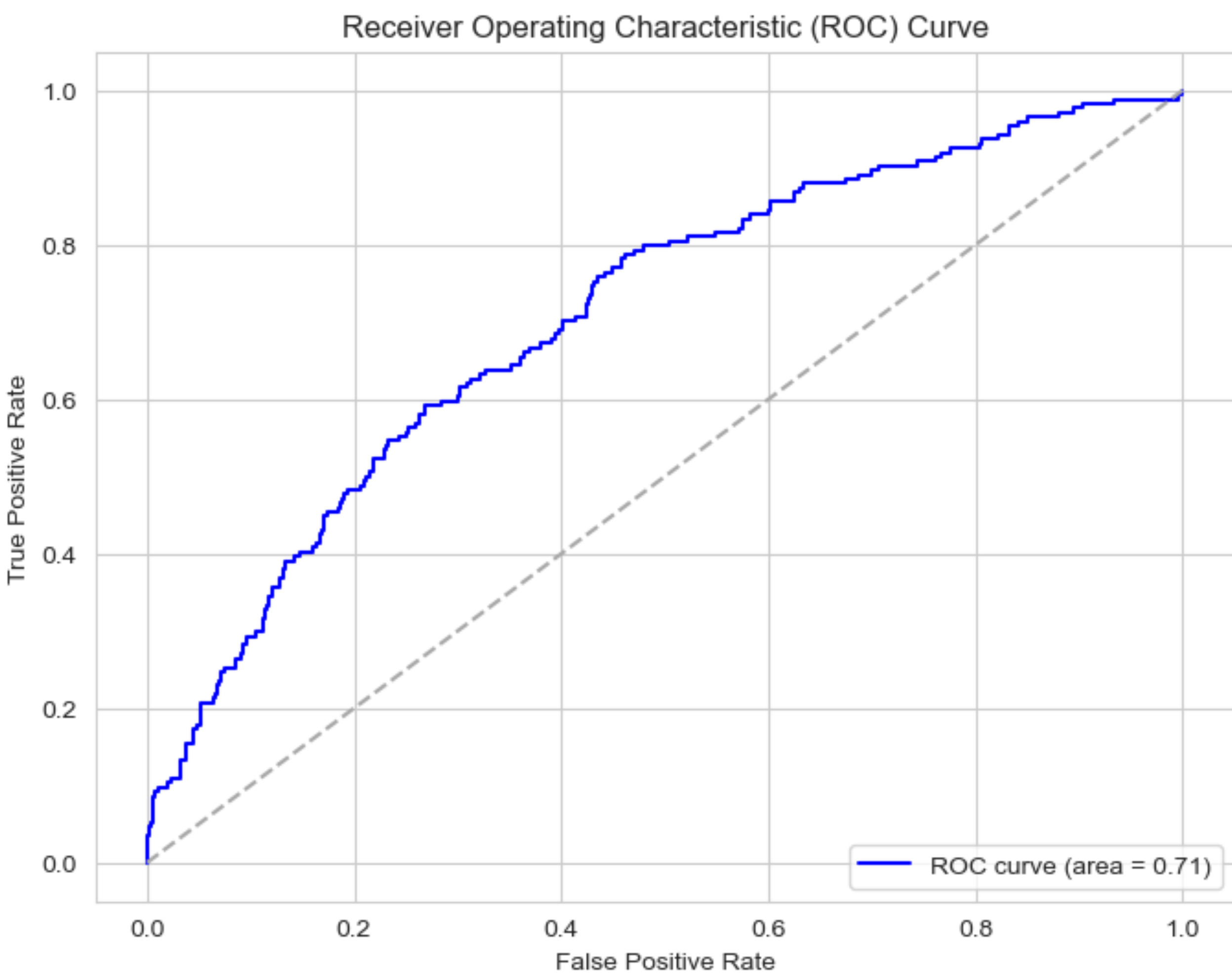
In [19]:
```
# Calculating the probabilities of the predictions
y_pred_prob = logistic_model.predict_proba(X_test)[:, 1]

# Calculating AUC
roc_auc = roc_auc_score(y_test, y_pred_prob)

# Calculating ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)

# Plotting the ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='darkgrey', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()

roc_auc
```




Out[19]: 0.7079550690526892

- ROC Curve: This plot will show the performance of your logistic regression model at all classification thresholds. The curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR). A model that predicts perfectly will have a ROC curve that passes through the top left corner of the plot, indicating a high True Positive Rate and a low False Positive Rate.

- AUC Value: The Area Under the Curve (AUC) provides a single number summarizing the performance of the model across all thresholds.

In our case, AUC value is coming out to be 0.71 which is a good sign that the predicted values are going to be correct.

## Conclusion

- In this project, I conducted a comprehensive analysis of a heart disease dataset obtained from Kaggle to gain insights into the factors associated with heart disease. The primary objective was to use the Logistic Regression to understand the accuracy of predictions and explore the dataset's characteristics.
- The analysis was conducted on a dataset comprising variables like age, gender, cholesterol levels, blood pressure, and other health-related metrics.
- To understand the model's performance more deeply, precision, recall, and F1-score were evaluated too.
- The model's ability to distinguish between positive and negative cases was evaluated using the AUC.

In conclusion, if a new record is added in the dataset, there is a good chance of getting an accurate result