```
#IMPORT THE LIBRARIES

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import SMOTE
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
```

```
data= pd.read_csv('/content/framingham.csv')
```

```
data.head()
```

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	he
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	
2	1	48	1.0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	
3	0	61	3.0	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	
4	0	46	3.0	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	

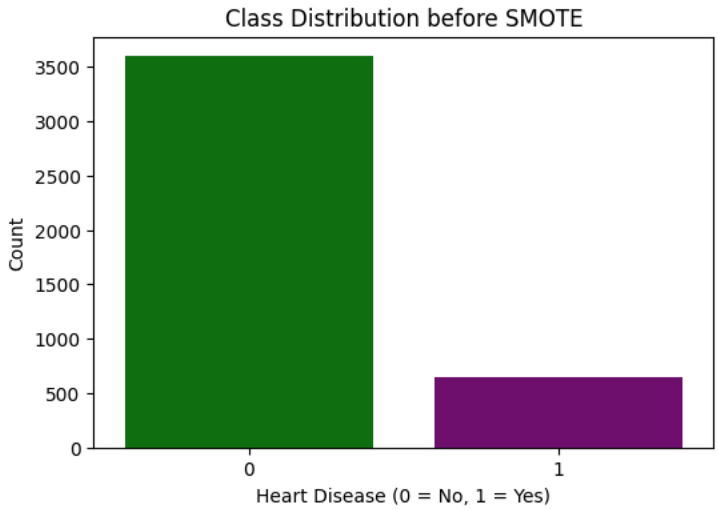
Next steps: Generate code with data | View recommended plots | New interactive sheet

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4240 entries, 0 to 4239
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   male             4240 non-null   int64  
 1   age              4240 non-null   int64  
 2   education        4135 non-null   float64
 3   currentSmoker    4240 non-null   int64  
 4   cigsPerDay       4211 non-null   float64
 5   BPMeds           4187 non-null   float64
 6   prevalentStroke  4240 non-null   int64  
 7   prevalentHyp     4240 non-null   int64  
 8   diabetes         4240 non-null   int64  
 9   totChol          4190 non-null   float64
 10  sysBP            4240 non-null   float64
 11  diaBP            4240 non-null   float64
 12  BMI              4221 non-null   float64
 13  heartRate        4239 non-null   float64
 14  glucose          3852 non-null   float64
 15  TenYearCHD       4240 non-null   int64  
dtypes: float64(9), int64(7)
memory usage: 530.1 KB
```

```
plt.figure(figsize=(6,4))
cols = ["green", "purple"]
sns.countplot(x=data["TenYearCHD"], palette=cols)
plt.title("Class Distribution before SMOTE")
plt.xlabel("Heart Disease (0 = No, 1 = Yes)")
plt.ylabel("Count")
plt.show()
#Data is imbalance
```

```
<ipython-input-6-055a941ef299>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.countplot(x=data["TenYearCHD"], palette=cols)
```




```
data.shape
```

```
(4240, 16)
```

```
data.fillna(data.median(), inplace=True)  # Replaces missing values with column medians
```

```
data.shape
```

```
(4240, 16)
```

```
x = data.drop(columns=["TenYearCHD"])
y = data["TenYearCHD"]
```

```
X.head()
```

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	hea
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	
2	1	48	1.0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	
3	0	61	3.0	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	
4	0	46	3.0	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	

```
y.head()
```

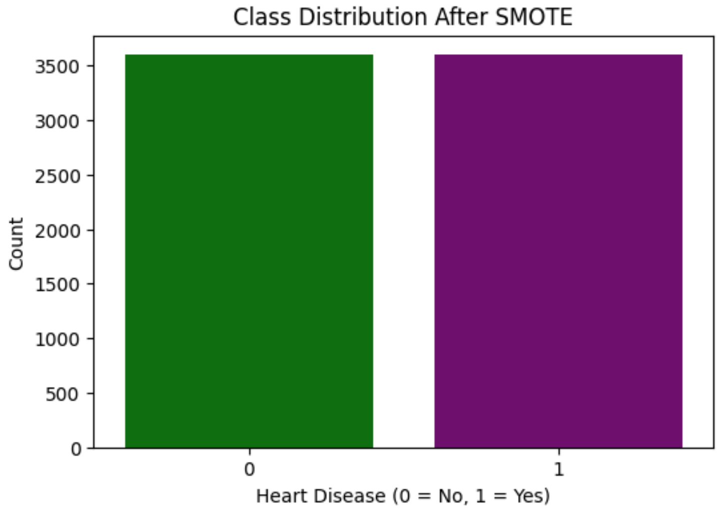
	TenYearCHD
0	0
1	0
2	0
3	1
4	0

```
# Standardize features
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)
```

```
# Apply SMOTE to balance dataset
smote = SMOTE(random_state=42)
```

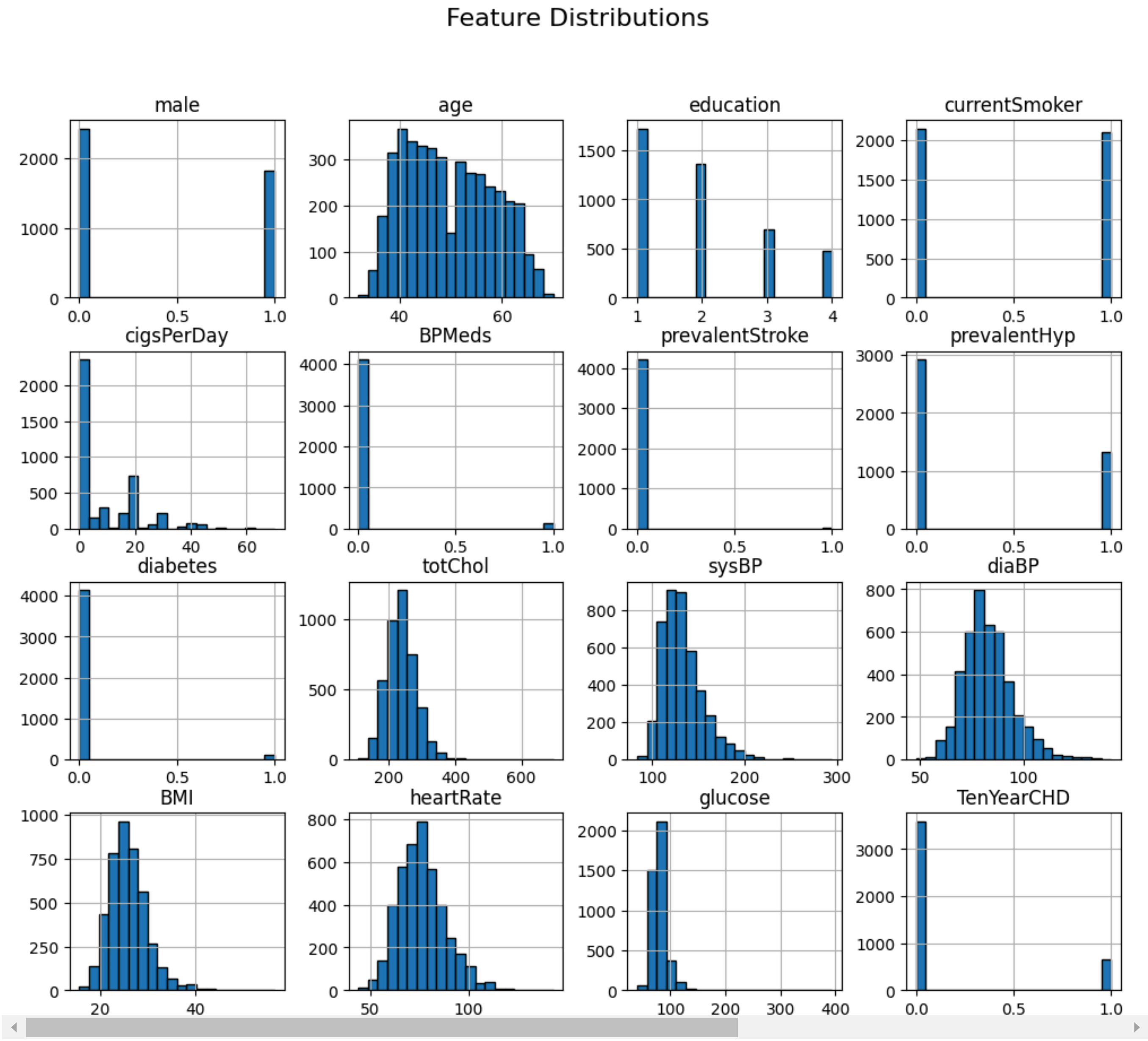
```
x_resampled, y_resampled = smote.fit_resample(x_scaled, y)


# Visualize class distribution after SMOTE
plt.figure(figsize=(6,4))
sns.countplot(x=y_resampled, palette=["green", "purple"])
plt.title("Class Distribution After SMOTE")
plt.xlabel("Heart Disease (0 = No, 1 = Yes)")
plt.ylabel("Count")
plt.show()
```

```
<ipython-input-14-98d785fc8211>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.countplot(x=y_resampled, palette=["green", "purple"])
```

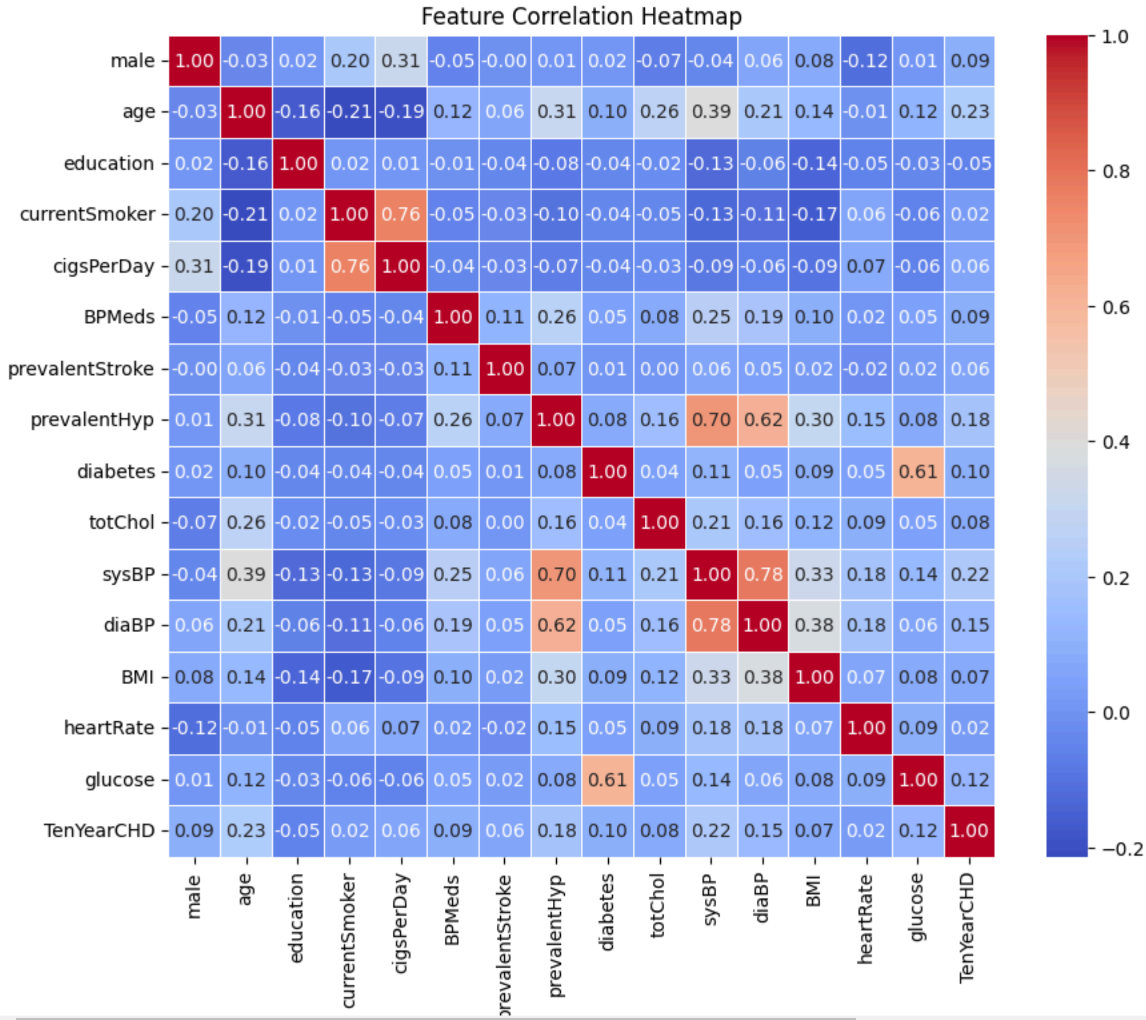



```
plt.figure(figsize=(10, 6))
data.hist(figsize=(12, 10), bins=20, edgecolor='black')
plt.suptitle("Feature Distributions", fontsize=16)
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```




```
# Correlation heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(), annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)
plt.title("Feature Correlation Heatmap")
plt.show()
```




```
# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(x_resampled, y_resampled, test_size=0.2, random_state=42)
```

```
model = Sequential([
    Dense(32, activation="relu", input_shape=(X_train.shape[1],)),
    Dropout(0.2),
    Dense(16, activation="relu"),
    Dropout(0.2),
    Dense(1, activation="sigmoid")  # Output layer for binary classification
])
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` arg
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
# Compile model using ADAM optimizer
model.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy"])
```

```
# Train the model
history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_data=(X_test, y_test), verbose=1)
```

```
Epoch 31/50
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.7121 - loss: 0.5629 - val_accuracy: 0.7116 - val_loss: 0.5580
Epoch 32/50
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.6998 - loss: 0.5691 - val_accuracy: 0.7116 - val_loss: 0.5562
Epoch 33/50
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.7059 - loss: 0.5555 - val_accuracy: 0.7081 - val_loss: 0.5558
Epoch 34/50
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.7253 - loss: 0.5512 - val_accuracy: 0.7192 - val_loss: 0.5548
Epoch 35/50
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.7105 - loss: 0.5611 - val_accuracy: 0.7151 - val_loss: 0.5521
Epoch 36/50
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.7178 - loss: 0.5636 - val_accuracy: 0.7151 - val_loss: 0.5524
Epoch 37/50
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.7258 - loss: 0.5539 - val_accuracy: 0.7151 - val_loss: 0.5534
Epoch 38/50
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.7171 - loss: 0.5498 - val_accuracy: 0.7186 - val_loss: 0.5522
Epoch 39/50
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.7306 - loss: 0.5397 - val_accuracy: 0.7192 - val_loss: 0.5500
Epoch 40/50
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.7174 - loss: 0.5499 - val_accuracy: 0.7269 - val_loss: 0.5496
Epoch 41/50
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.7225 - loss: 0.5415 - val_accuracy: 0.7213 - val_loss: 0.5521
Epoch 42/50
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.7331 - loss: 0.5380 - val_accuracy: 0.7241 - val_loss: 0.5508
Epoch 43/50
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.7309 - loss: 0.5433 - val_accuracy: 0.7206 - val_loss: 0.5511
Epoch 44/50
180/180 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.7407 - loss: 0.5359 - val_accuracy: 0.7206 - val_loss: 0.5479
Epoch 45/50
```