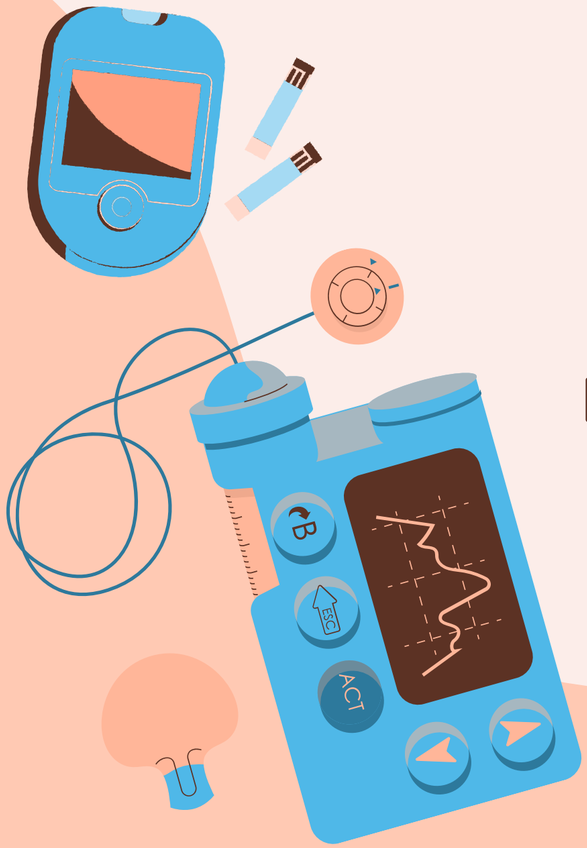
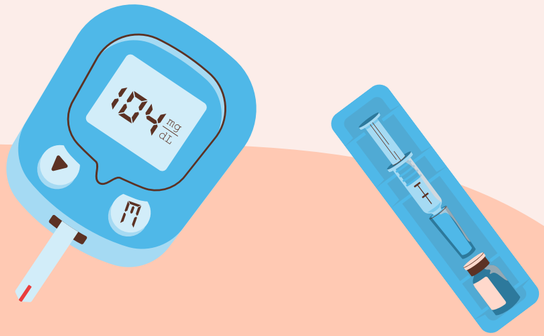
Final project:
Heart Diseases

Presented by: Paola Yunes, Rafael Orihuela,
Emilio Aristegui, Adrian Fraile





Our Problem:

- Healthcare industry: We sought to identify heart disease on time for proper treatment by using Machine Learning techniques.
- Train an algorithm to recognize disease symptoms and predict if a patient is at risk.

What is Heart disease?

- A group of conditions that affect the heart and blood vessels.
- These conditions impact the heart's structure and function, leading to reduced blood flow, heart attacks, and other serious complications.
- Heart disease is a leading cause of death worldwide.

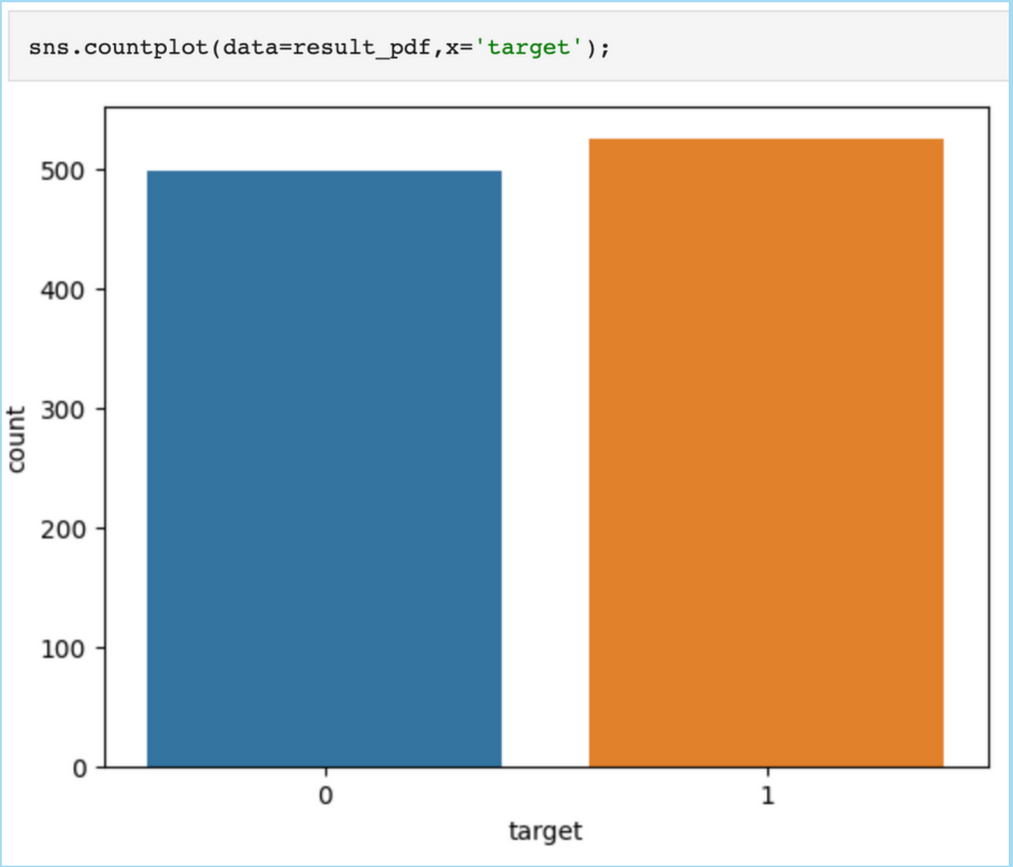
Our Dataset

A public health dataset retrieved from Kaggle that contains 4 databases from: Cleveland, Hungary, Switzerland and Long Beach with 76 attributes, though we will only use 14.

Summary Statistics

Heart Disease Plot

```
sns.countplot(data=result_pdf,x='target');
```

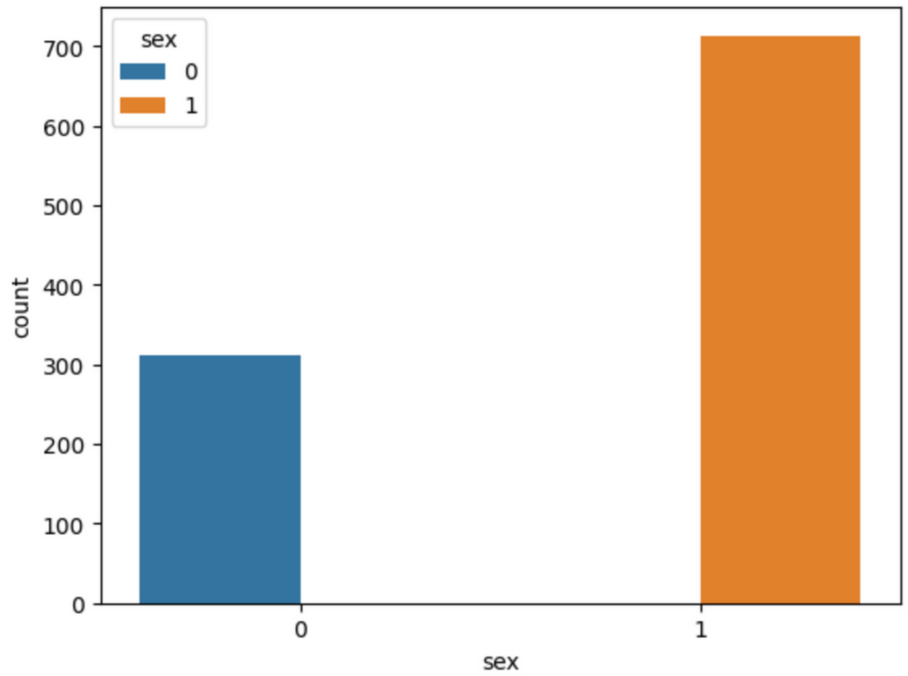


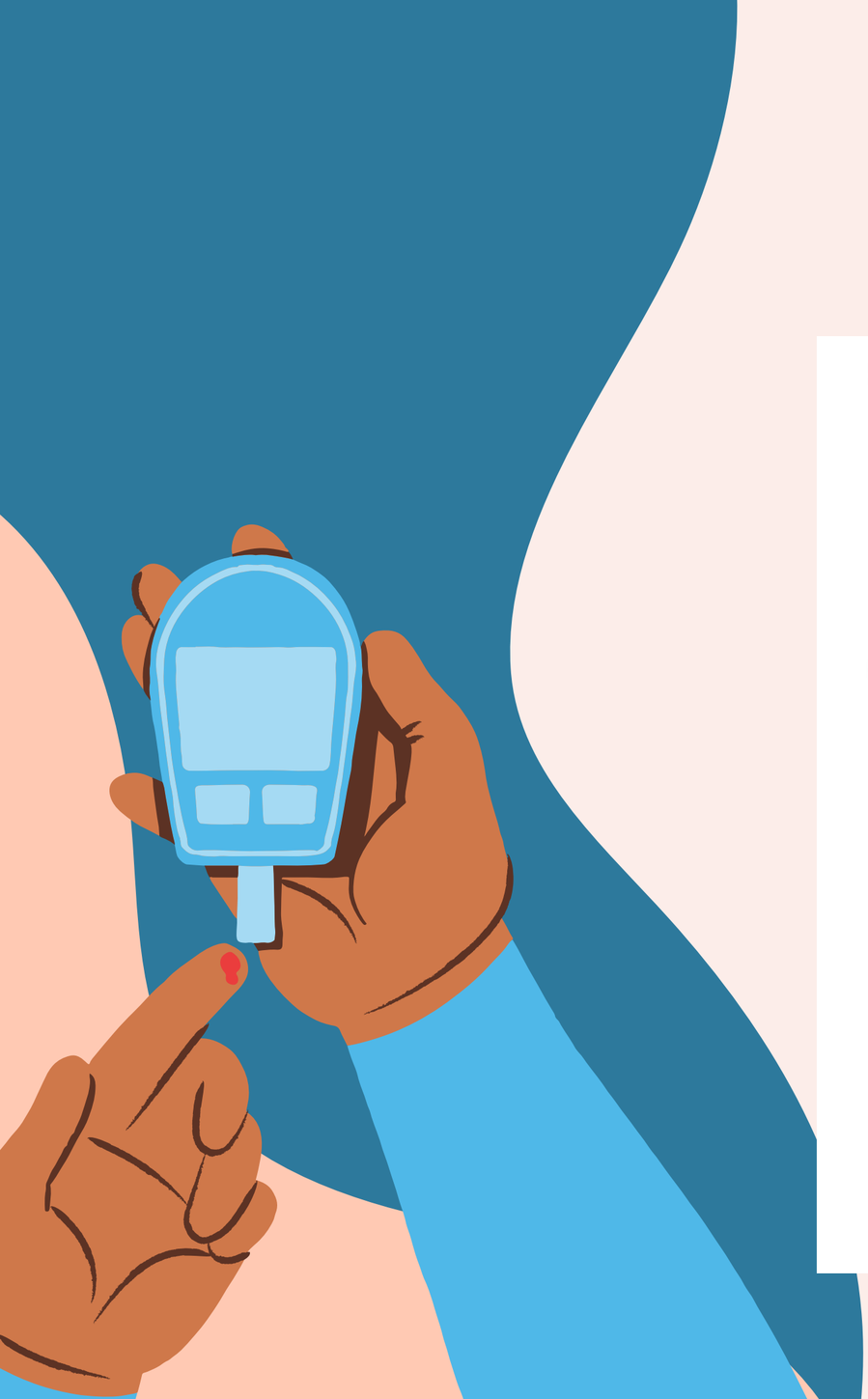
Summary Statistics

Gender Plot

In [12]:

```
sns.countplot(data=result_pdf,x='sex', hue="sex");
#Sex = 0 is woman
#sex = 1 is man
```



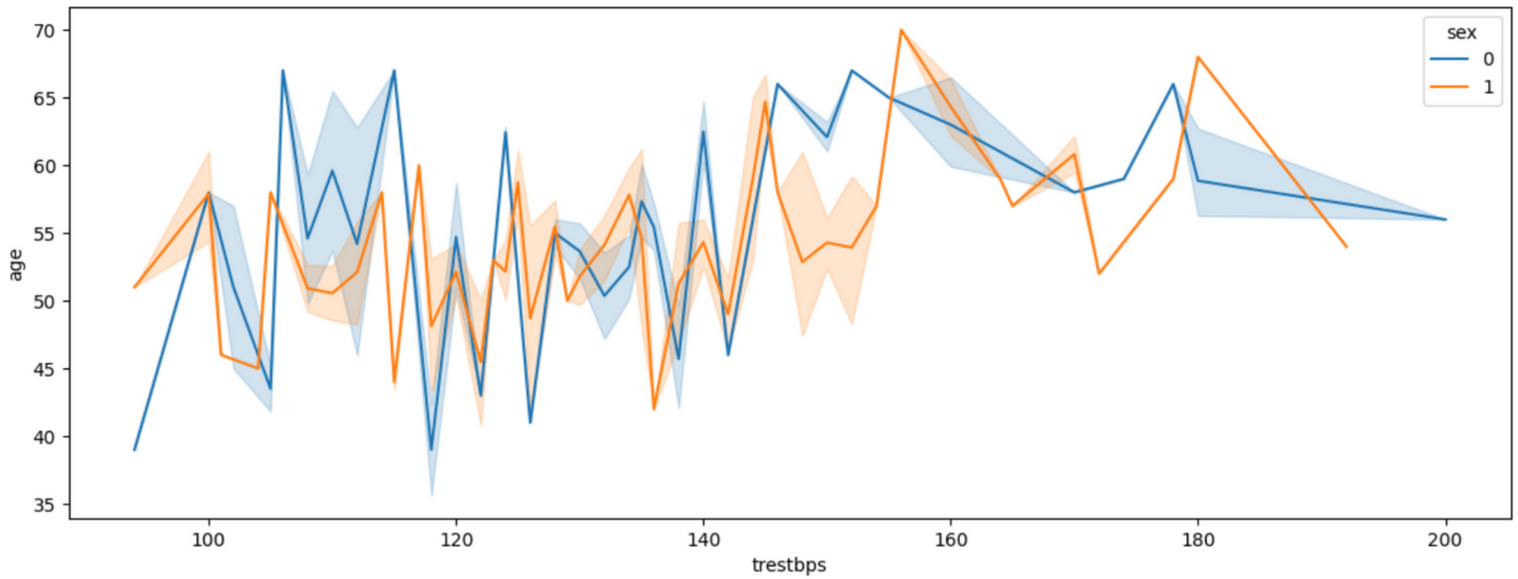


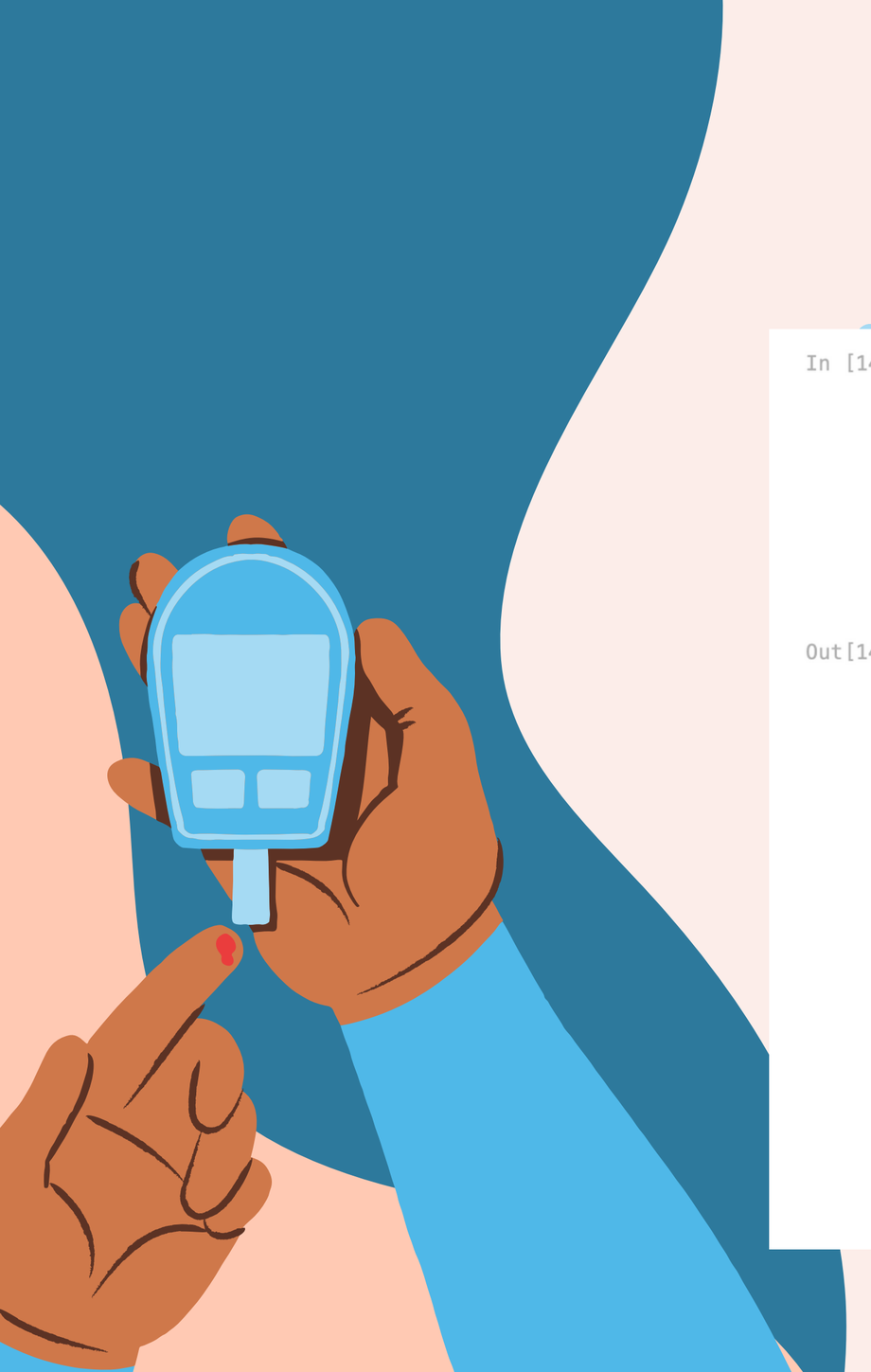
Summary Statistics

Resting Blood Pressure Line Plot

In [13]:
```
# resting blood pressure (in mm Hg on admission to the hospital) vs age

plt.figure(figsize=(14, 5))

sns.lineplot(data = result_pdf,
             x="trestbps",
             y="age",
             hue = "sex",
             )
```

Out[13]: <Axes: xlabel='trestbps', ylabel='age'>



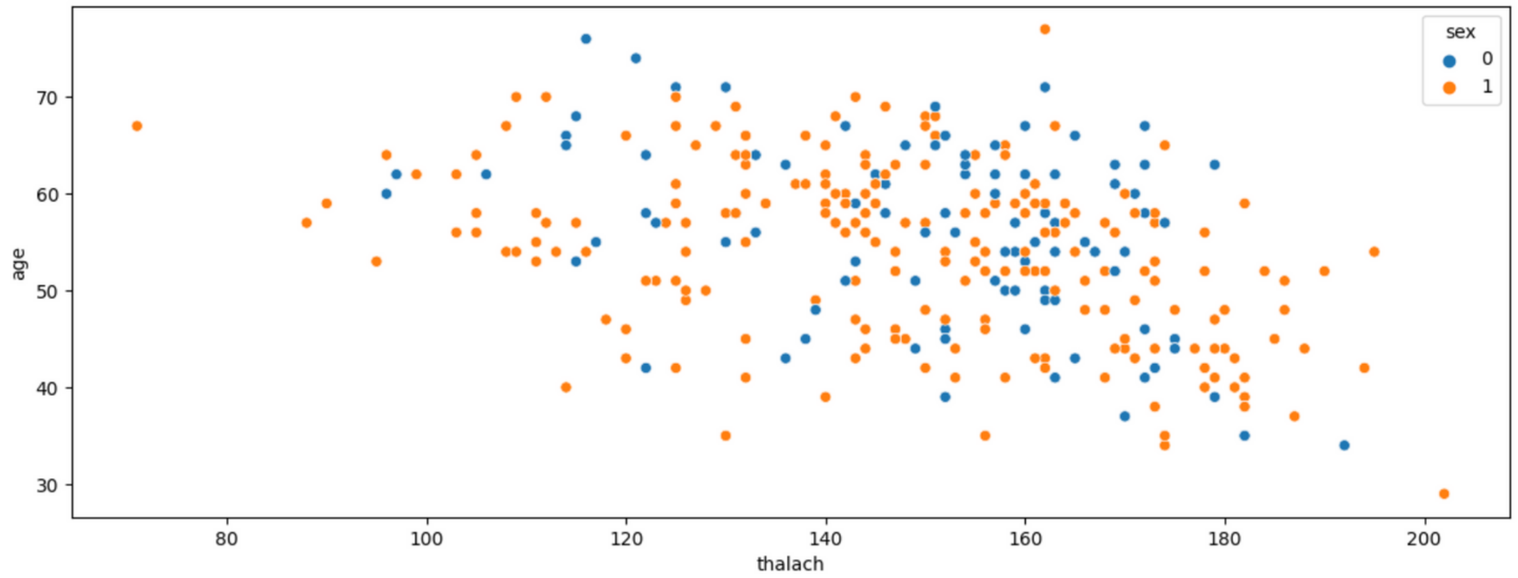


Summary Statistics

Heart Rate Scatterplot

In [14]:
```
# thalach - maximum heart rate achieved

plt.figure(figsize=(14, 5))

sns.scatterplot(data = result_pdf,
                x="thalach",
                y="age",
                hue = "sex",
                )
```

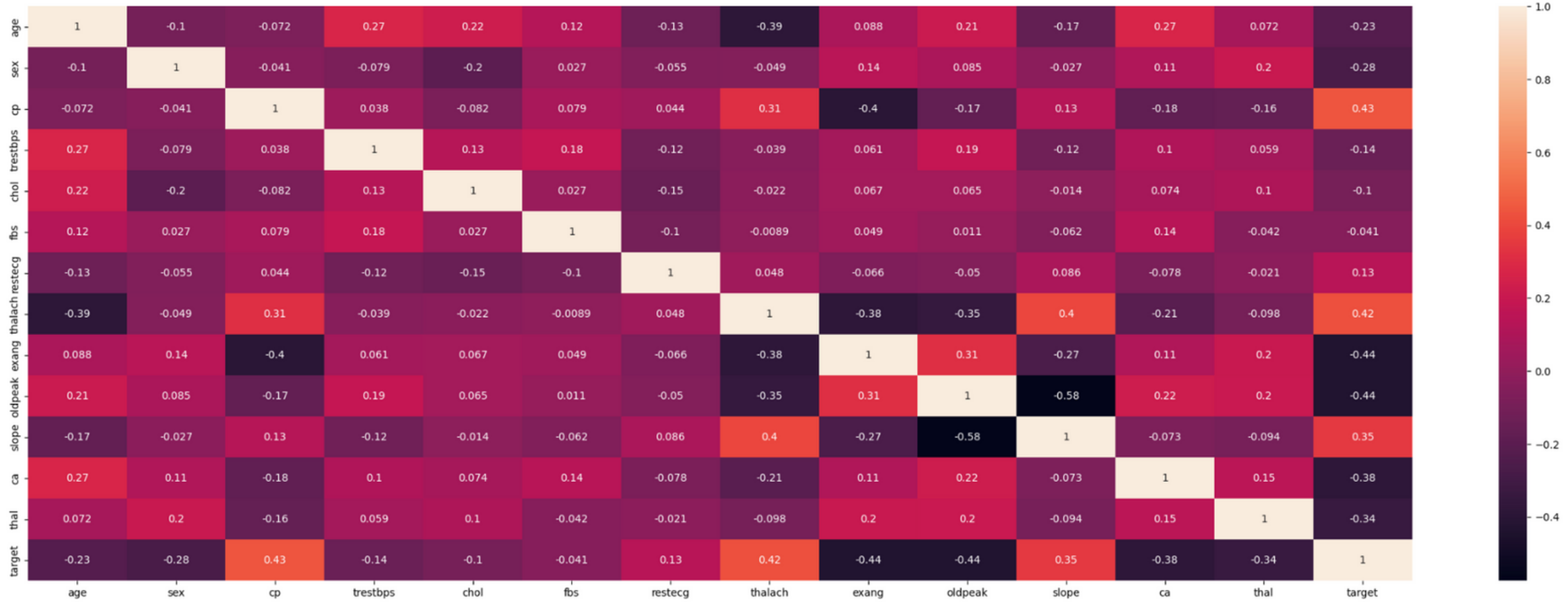
Out[14]: <Axes: xlabel='thalach', ylabel='age'>



Summary Statistics

Features Heatmap

Out[37]: <Axes: >



Data Model Implementation:

A Python script initializes, trains, and evaluates the model

Model: training and evaluating

In [16]:

```
# Split our preprocessed data into our features and target arrays
# Separate features (X) and target (y)
X = result_pdf.drop('target', axis=1)
y = result_pdf['target']

# Split the preprocessed data into a training and testing dataset
X_train, X_test, y_train, y_test = train_test_split(X, y)
#, random_state=50
# Display the shapes of the resulting datasets
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (768, 13)
X_test shape: (257, 13)
y_train shape: (768,)
y_test shape: (257,)
```

In [17]:

```
# Create a StandardScaler instances
scaler = StandardScaler()

# Fit the StandardScaler
X_scaler = scaler.fit(X_train)

# Scale the data
X_train_scaled = X_scaler.transform(X_train)
X_test_scaled = X_scaler.transform(X_test)
```

Data Model Implementation:

The data is cleaned, normalized, and standardized prior to modeling

In [6]:
```
result_pdf = df.select("*").toPandas()
```

In [7]:
```
result_pdf
```

Out[7]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1020	59	1	1	140	221	0	1	164	1	0	2	0	2	1
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	0
1022	47	1	0	110	275	0	0	118	1	1	1	1	2	0
1023	50	0	0	110	254	0	0	159	0	0	2	0	2	1
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3	0

1025 rows × 14 columns

In [8]:
```
result_pdf.dtypes
```

Out[8]:
```
age          object
sex          object
cp           object
trestbps     object
chol         object
fbs          object
```

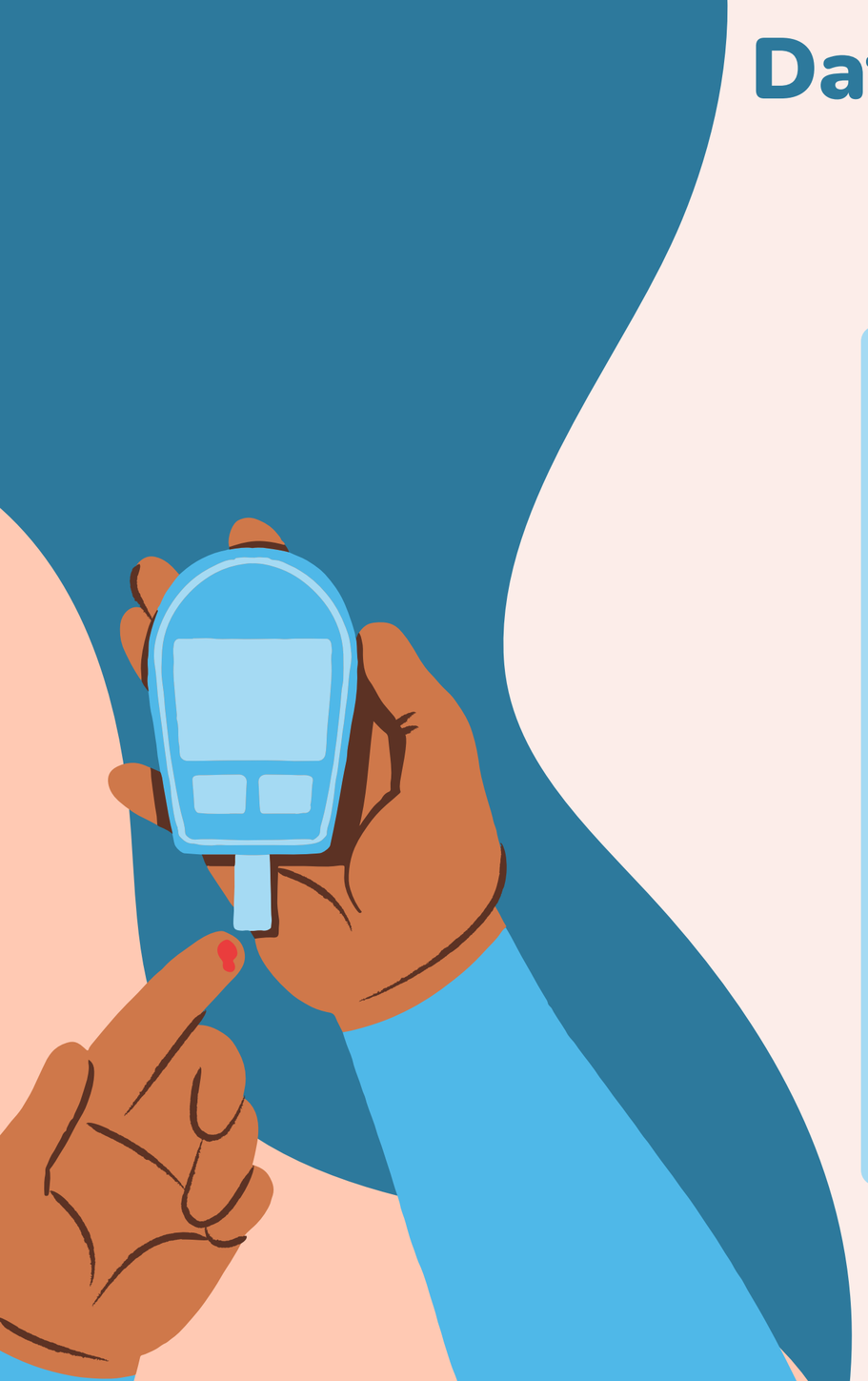
Data Model Implementation:

The model utilizes data retrieved from SQL or Spark

In [3]:
```
# Start Spark session
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("DataFrame Basics").getOrCreate()
```

In [4]:
```
# Read the data from S3 Buckets
from pyspark import SparkFiles
url = "./heart.csv"
spark.sparkContext.addFile(url)
df = spark.read.csv(SparkFiles.get('heart.csv'), sep=',', header = True, ignoreLeadingWhiteSpace=True)

# Show DataFrame
df.show()
```

```
+---+---+---+--------+----+---+-------+-------+-----+-------+-----+---+----+------+
|age|sex| cp|trestbps|chol|fbs|restecg|thalach|exang|oldpeak|slope| ca|thal|target|
+---+---+---+--------+----+---+-------+-------+-----+-------+-----+---+----+------+
| 52|  1|  0|     125| 212|  0|      1|    168|    0|      1|    2|  2|   3|     0|
| 53|  1|  0|     140| 203|  1|      0|    155|    1|    3.1|    0|  0|   3|     0|
| 70|  1|  0|     145| 174|  0|      1|    125|    1|    2.6|    0|  0|   3|     0|
| 61|  1|  0|     148| 203|  0|      1|    161|    0|      0|    2|  1|   3|     0|
| 62|  0|  0|     138| 294|  1|      1|    106|    0|    1.9|    1|  3|   2|     0|
```



Data Model Implementation: Attempt 1

The model demonstrates meaningful predictive power at least 75% classification accuracy or 0.80 R-squared.

In [18]:

```
# ATTEMPT 1
from keras.layers import Activation, Dense
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
nn = tf.keras.models.Sequential()

# Number of input features
input_features = len(X_train_scaled[0])

# First hidden layer
nn.add(Dense(units=80, input_dim=input_features, activation='relu'))

# Second hidden layer
nn.add(Dense(units=30, activation='relu'))

# Output layer
nn.add(Dense(units=1, activation='tanh'))

# Check the structure of the model
nn.summary()

# Compile the model
nn.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 80)                1120

 dense_1 (Dense)             (None, 30)                2430

 dense_2 (Dense)             (None, 1)                 31

=================================================================
Total params: 3,581
Trainable params: 3,581
Non-trainable params: 0
_________________________________________________________________
```

Data Model Implementation: Attempt 2

The model demonstrates meaningful predictive power at least 75% classification accuracy or 0.80 R-squared.

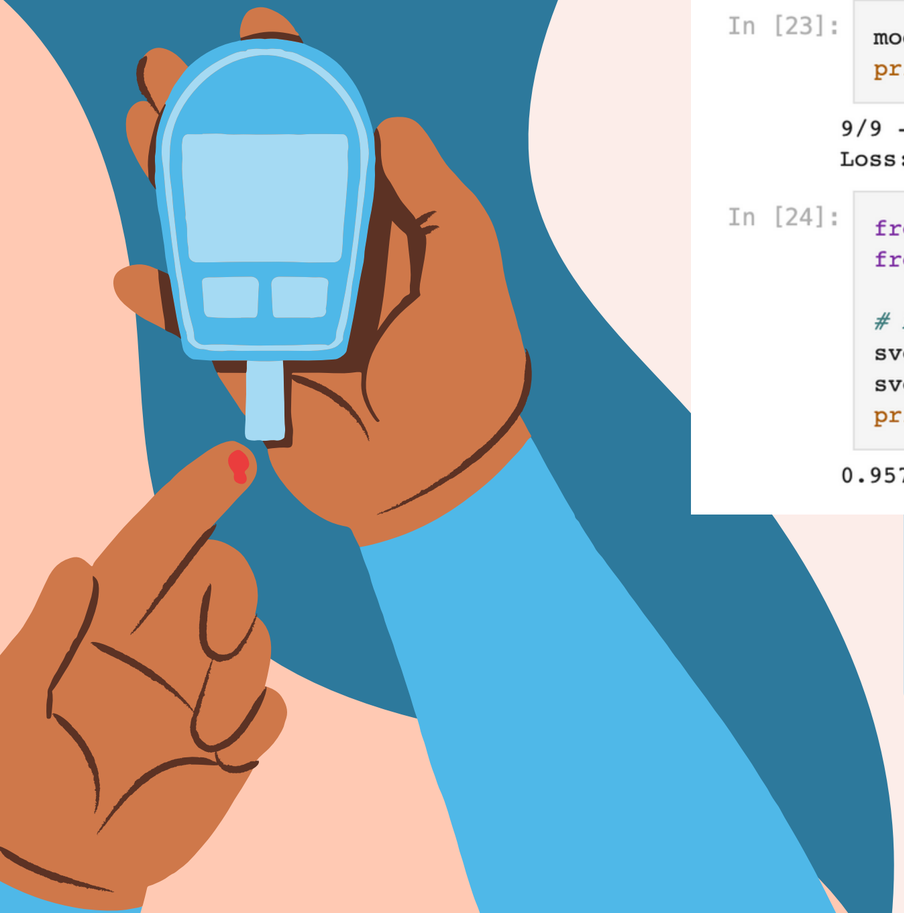
In [20]:

```
# ATTEMPT 2
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
nn = tf.keras.models.Sequential()

# Number of input features
input_features = len(X_train_scaled[0])

# First hidden layer
nn.add(Dense(units=100, input_dim=input_features, activation='relu'))

# Second hidden layer
nn.add(Dense(units=80, activation='relu'))

# Third hidden layer
nn.add(Dense(units=30, activation='relu'))

# Output layer , we need to use softmax because
nn.add(Dense(units=50, activation='softmax'))

# Check the structure of the model
nn.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 100)               1400

 dense_4 (Dense)             (None, 80)                8080

 dense_5 (Dense)             (None, 30)                2430

 dense_6 (Dense)             (None, 50)                1550
```

Data Model Implementation: Attempt 2

The model demonstrates meaningful predictive power at least 75% classification accuracy or 0.80 R-squared.

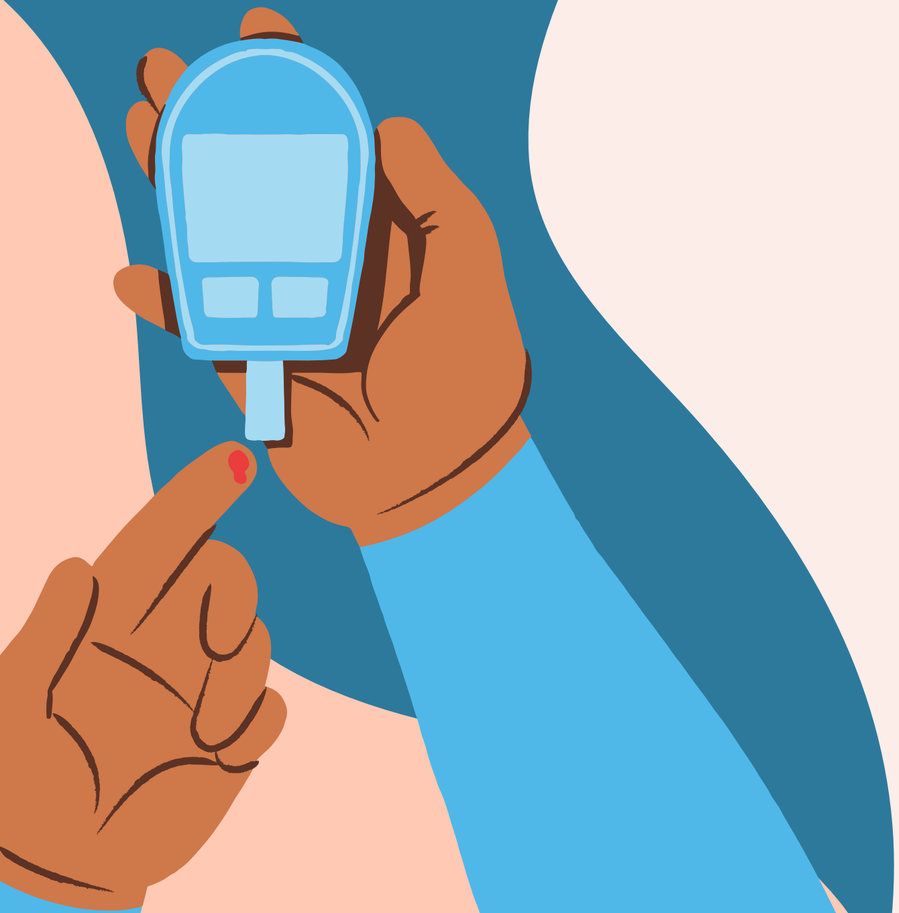


In [23]:
```
model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
9/9 - 0s - loss: 0.2169 - accuracy: 0.9572 - 50ms/epoch - 6ms/step
Loss: 0.21685080230236053, Accuracy: 0.957198441028595
```

In [24]:
```
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.svm import SVC

# Model -  Support Vector Classificator
svc = SVC(C=1.5)
svc.fit(X_train_scaled, y_train)
print(svc.score(X_test_scaled, y_test))
```

```
0.9571984435797666
```



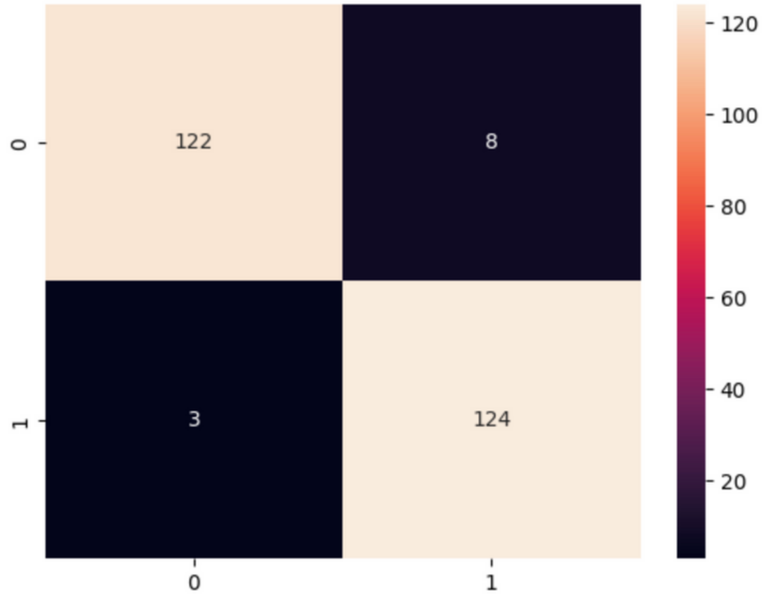
Data Model Implementation: Attempt 2

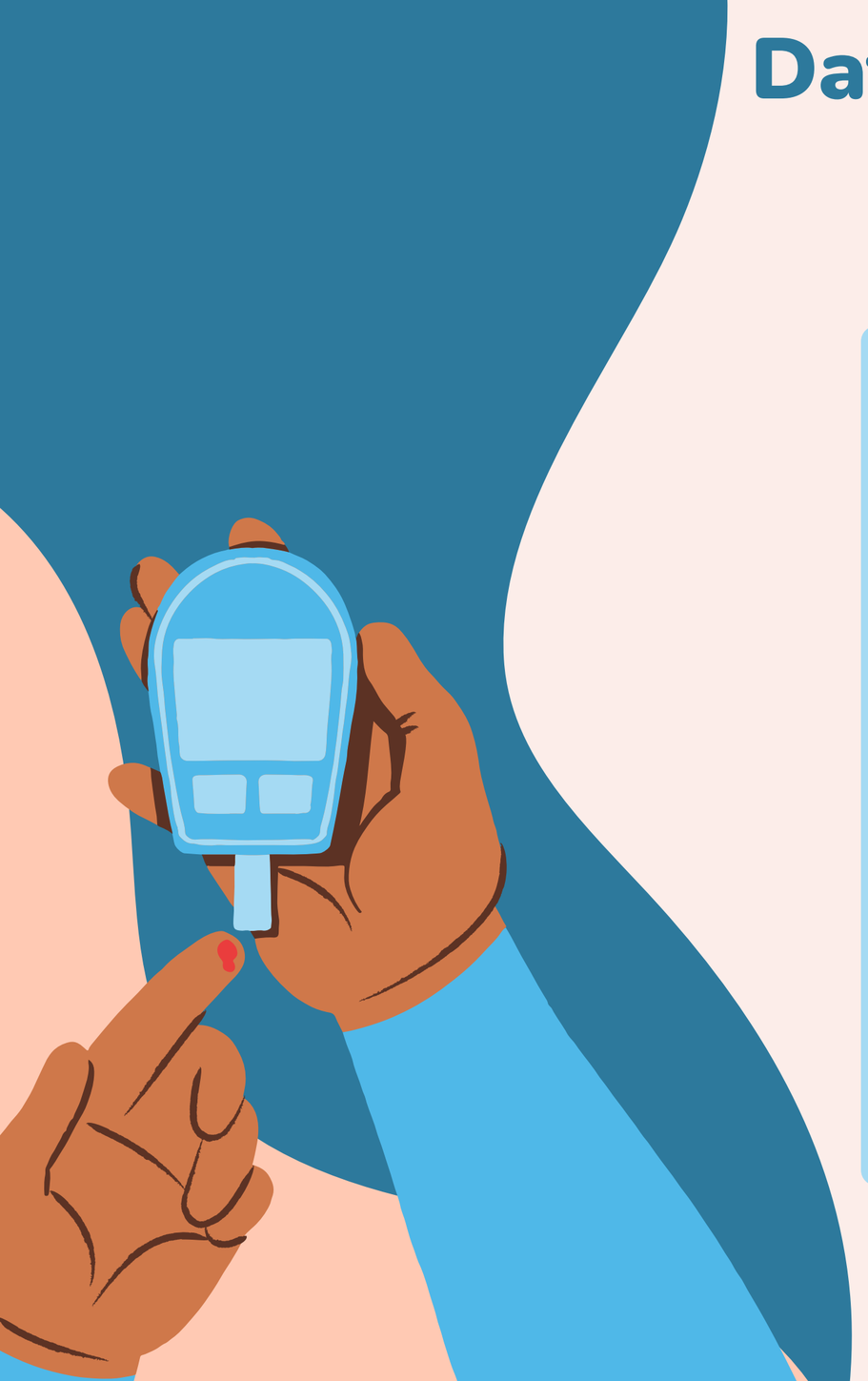
Confusion Matrix

In [25]:

```
# Confusion Matrix
cnf = confusion_matrix(y_test, svc.predict(X_test_scaled))
sns.heatmap(cnf, annot=True, fmt='.0f')

print(classification_report(y_test, svc.predict(X_test_scaled)))
```

```
              precision    recall  f1-score   support

           0       0.98      0.94      0.96       130
           1       0.94      0.98      0.96       127

    accuracy                           0.96       257
   macro avg       0.96      0.96      0.96       257
weighted avg       0.96      0.96      0.96       257
```





Data Model Implementation: Attempt 3

The model demonstrates meaningful predictive power at least 75% classification accuracy or 0.80 R-squared.

In [26]:

```
# Atempt 3
#X = result_pdf.drop('target', axis=1)
X = result_pdf[['cp','thalach','slope']]
y = result_pdf['target']

# Split the preprocessed data into a training and testing dataset
X_train, X_test, y_train, y_test = train_test_split(X, y)
#, random_state=0

# Display the shapes of the resulting datasets
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)
```

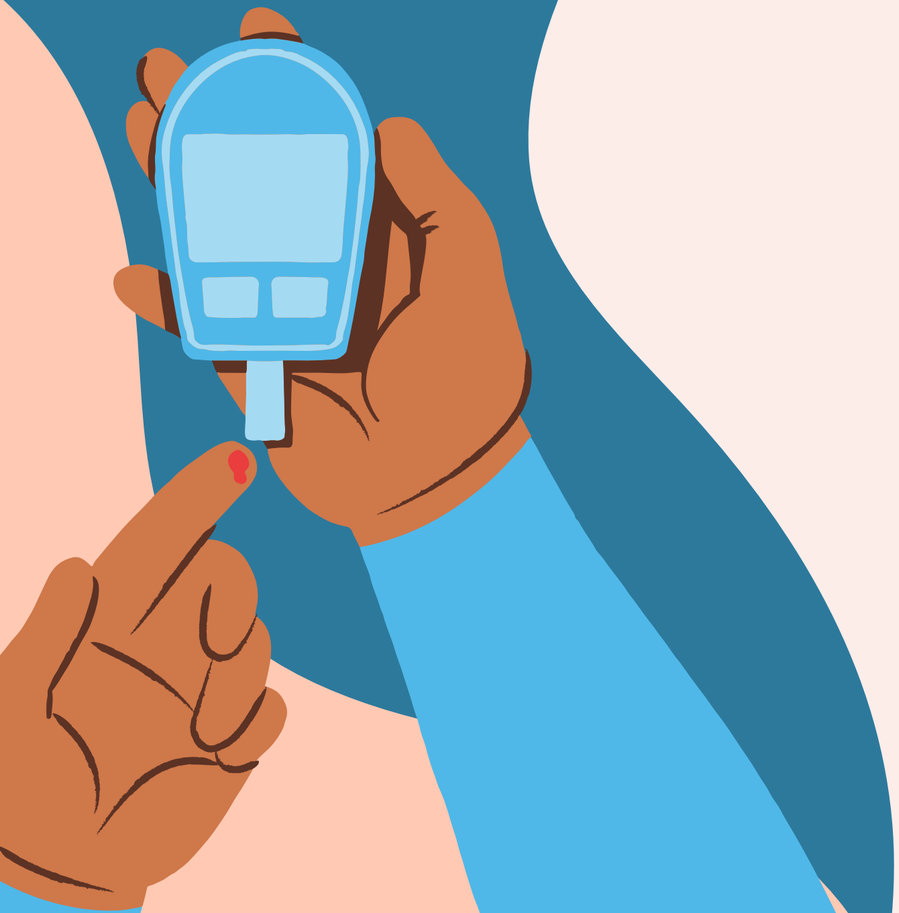
```
X_train shape: (768, 3)
X_test shape: (257, 3)
y_train shape: (768,)
y_test shape: (257,)
```

In [27]:

```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
nn = tf.keras.models.Sequential()

# Number of input features
input_features = len(X_train_scaled[0])

# First hidden layer
nn.add(Dense(units=100, input_dim=input_features, activation='relu'))

# Second hidden layer
nn.add(Dense(units=80, activation='relu'))

# Third hidden layer
nn.add(Dense(units=30, activation='relu'))

# Output layer , we need to use softmax because
nn.add(Dense(units=50, activation='softmax'))

# Check the structure of the model
```

Data Model Implementation: Attempt 3

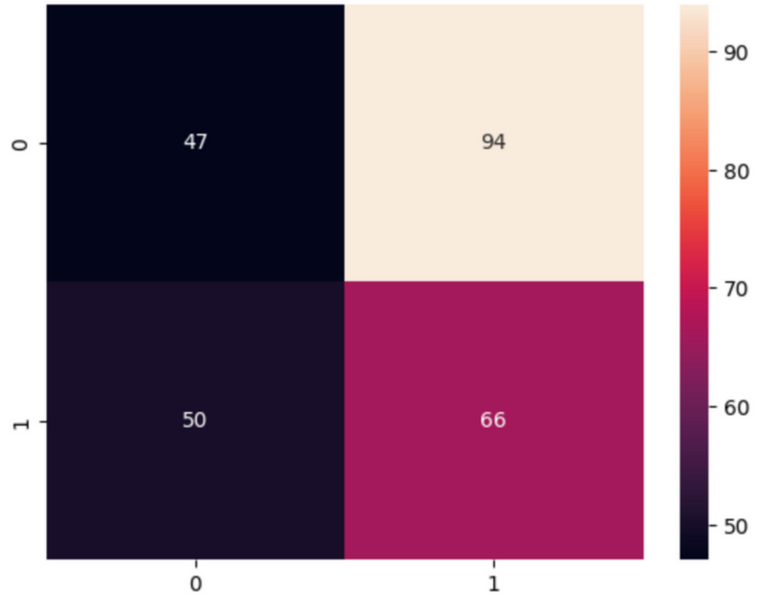
Confusion Matrix



In [33]:

```
# Confusion Matrix
cnf = confusion_matrix(y_test, svc.predict(X_test_scaled))
sns.heatmap(cnf, annot=True, fmt='.0f')

print(classification_report(y_test, svc.predict(X_test_scaled)))
```

```
              precision    recall  f1-score   support

           0       0.48      0.33      0.39       141
           1       0.41      0.57      0.48       116

    accuracy                           0.44       257
   macro avg       0.45      0.45      0.44       257
weighted avg       0.45      0.44      0.43       257
```



Thank you!

WEBSITE:

https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset

GITHUB

https://github.com/emilioaristegui/project-4