# herat-precdicted-cnn-1

March 30, 2024

[3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, classification_report
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten, Dense,␣
↪Dropout
```

[4]:
```
from google.colab import files


uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving heart.csv to heart.csv
```

[6]:
```
df = pd.read_csv('heart.csv')

# EDA
# Display first few rows
print("First few rows of the dataset:")
df.head()
```

```
First few rows of the dataset:
```

[6]:

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 52 | 1 | 0 | 125 | 212 | 0 | 1 | 168 | 0 | 1.0 | 2 | | | |
| 1 | 53 | 1 | 0 | 140 | 203 | 1 | 0 | 155 | 1 | 3.1 | 0 | | | |
| 2 | 70 | 1 | 0 | 145 | 174 | 0 | 1 | 125 | 1 | 2.6 | 0 | | | |
| 3 | 61 | 1 | 0 | 148 | 203 | 0 | 1 | 161 | 0 | 0.0 | 2 | | | |
| 4 | 62 | 0 | 0 | 138 | 294 | 1 | 1 | 106 | 0 | 1.9 | 1 | | | |

```
0   2     3       0
1   0     3       0
2   0     3       0
3   1     3       0
4   3     2       0
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x78596a557640>
```

```
from matplotlib import pyplot as plt
_df_0['age'].plot(kind='hist', bins=20, title='age')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_1['sex'].plot(kind='hist', bins=20, title='sex')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_2['trestbps'].plot(kind='hist', bins=20, title='trestbps')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_3['chol'].plot(kind='hist', bins=20, title='chol')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x78596db46bc0>
```

```
from matplotlib import pyplot as plt
_df_4.plot(kind='scatter', x='age', y='sex', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_5.plot(kind='scatter', x='sex', y='trestbps', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_6.plot(kind='scatter', x='trestbps', y='chol', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_7.plot(kind='scatter', x='chol', y='fbs', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x78596a5d8700>
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  palette = list(sns.palettes.mpl_palette('Dark2'))
  xs = series['cp']
  ys = series['age']

  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])
```

```
fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_8.sort_values('cp', ascending=True)
_plot_series(df_sorted, '')
sns.despine(fig=fig, ax=ax)
plt.xlabel('cp')
_ = plt.ylabel('age')
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  palette = list(sns.palettes.mpl_palette('Dark2'))
  xs = series['cp']
  ys = series['sex']

  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_9.sort_values('cp', ascending=True)
_plot_series(df_sorted, '')
sns.despine(fig=fig, ax=ax)
plt.xlabel('cp')
_ = plt.ylabel('sex')
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  palette = list(sns.palettes.mpl_palette('Dark2'))
  xs = series['cp']
  ys = series['trestbps']

  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_10.sort_values('cp', ascending=True)
_plot_series(df_sorted, '')
sns.despine(fig=fig, ax=ax)
plt.xlabel('cp')
_ = plt.ylabel('trestbps')
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  palette = list(sns.palettes.mpl_palette('Dark2'))
  xs = series['cp']
  ys = series['chol']

  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
```

```
df_sorted = _df_11.sort_values('cp', ascending=True)
_plot_series(df_sorted, '')
sns.despine(fig=fig, ax=ax)
plt.xlabel('cp')
_ = plt.ylabel('chol')
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x78596a5d8190>
```

```
from matplotlib import pyplot as plt
_df_12['age'].plot(kind='line', figsize=(8, 4), title='age')
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_13['sex'].plot(kind='line', figsize=(8, 4), title='sex')
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_14['trestbps'].plot(kind='line', figsize=(8, 4), title='trestbps')
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_15['chol'].plot(kind='line', figsize=(8, 4), title='chol')
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x7859f2978f70>
```

```
from matplotlib import pyplot as plt
_df_16['index'].plot(kind='hist', bins=20, title='index')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_17['age'].plot(kind='hist', bins=20, title='age')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_18['sex'].plot(kind='hist', bins=20, title='sex')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_19['trestbps'].plot(kind='hist', bins=20, title='trestbps')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x78596db45480>
```

```
from matplotlib import pyplot as plt
_df_20.plot(kind='scatter', x='index', y='age', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_21.plot(kind='scatter', x='age', y='sex', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_22.plot(kind='scatter', x='sex', y='trestbps', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_23.plot(kind='scatter', x='trestbps', y='chol', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x785967b87e20>
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  palette = list(sns.palettes.mpl_palette('Dark2'))
  xs = series['index']
  ys = series['age']

  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_24.sort_values('index', ascending=True)
_plot_series(df_sorted, '')
sns.despine(fig=fig, ax=ax)
plt.xlabel('index')
_ = plt.ylabel('age')
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  palette = list(sns.palettes.mpl_palette('Dark2'))
  xs = series['index']
  ys = series['sex']

  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_25.sort_values('index', ascending=True)
_plot_series(df_sorted, '')
sns.despine(fig=fig, ax=ax)
plt.xlabel('index')
_ = plt.ylabel('sex')
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  palette = list(sns.palettes.mpl_palette('Dark2'))
  xs = series['index']
  ys = series['trestbps']

  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_26.sort_values('index', ascending=True)
_plot_series(df_sorted, '')
```

```
sns.despine(fig=fig, ax=ax)
plt.xlabel('index')
_ = plt.ylabel('trestbps')
```

```
from matplotlib import pyplot as plt
import seaborn as sns
def _plot_series(series, series_name, series_index=0):
  palette = list(sns.palettes.mpl_palette('Dark2'))
  xs = series['index']
  ys = series['chol']

  plt.plot(xs, ys, label=series_name, color=palette[series_index % len(palette)])

fig, ax = plt.subplots(figsize=(10, 5.2), layout='constrained')
df_sorted = _df_27.sort_values('index', ascending=True)
_plot_series(df_sorted, '')
sns.despine(fig=fig, ax=ax)
plt.xlabel('index')
_ = plt.ylabel('chol')
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x785967b869b0>
```

```
from matplotlib import pyplot as plt
_df_28['index'].plot(kind='line', figsize=(8, 4), title='index')
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_29['age'].plot(kind='line', figsize=(8, 4), title='age')
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_30['sex'].plot(kind='line', figsize=(8, 4), title='sex')
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_31['trestbps'].plot(kind='line', figsize=(8, 4), title='trestbps')
plt.gca().spines[['top', 'right']].set_visible(False)
```

[8]:
```
print("\nSummary statistics of the dataset:")
df.describe()
```

```
Summary statistics of the dataset:
```

[8]:

| | age | sex | cp | trestbps | chol | \ |
|---|---|---|---|---|---|---|
| count | 1025.000000 | 1025.000000 | 1025.000000 | 1025.000000 | 1025.00000 | |
| mean | 54.434146 | 0.695610 | 0.942439 | 131.611707 | 246.00000 | |
| std | 9.072290 | 0.460373 | 1.029641 | 17.516718 | 51.59251 | |
| min | 29.000000 | 0.000000 | 0.000000 | 94.000000 | 126.00000 | |
| 25% | 48.000000 | 0.000000 | 0.000000 | 120.000000 | 211.00000 | |

```
50%      56.000000     1.000000     1.000000   130.000000   240.00000
75%      61.000000     1.000000     2.000000   140.000000   275.00000
max      77.000000     1.000000     3.000000   200.000000   564.00000

              fbs      restecg      thalach        exang      oldpeak  \
count  1025.000000  1025.000000  1025.000000  1025.000000  1025.000000
mean      0.149268     0.529756   149.114146     0.336585     1.071512
std       0.356527     0.527878    23.005724     0.472772     1.175053
min       0.000000     0.000000    71.000000     0.000000     0.000000
25%       0.000000     0.000000   132.000000     0.000000     0.000000
50%       0.000000     1.000000   152.000000     0.000000     0.800000
75%       0.000000     1.000000   166.000000     1.000000     1.800000
max       1.000000     2.000000   202.000000     1.000000     6.200000

             slope           ca         thal       target
count  1025.000000  1025.000000  1025.000000  1025.000000
mean      1.385366     0.754146     2.323902     0.513171
std       0.617755     1.030798     0.620660     0.500070
min       0.000000     0.000000     0.000000     0.000000
25%       1.000000     0.000000     2.000000     0.000000
50%       1.000000     0.000000     2.000000     1.000000
75%       2.000000     1.000000     3.000000     1.000000
max       2.000000     4.000000     3.000000     1.000000
```

[9]:
```
print("\nMissing values in the dataset:")
df.isnull().sum()
```

```
Missing values in the dataset:
```

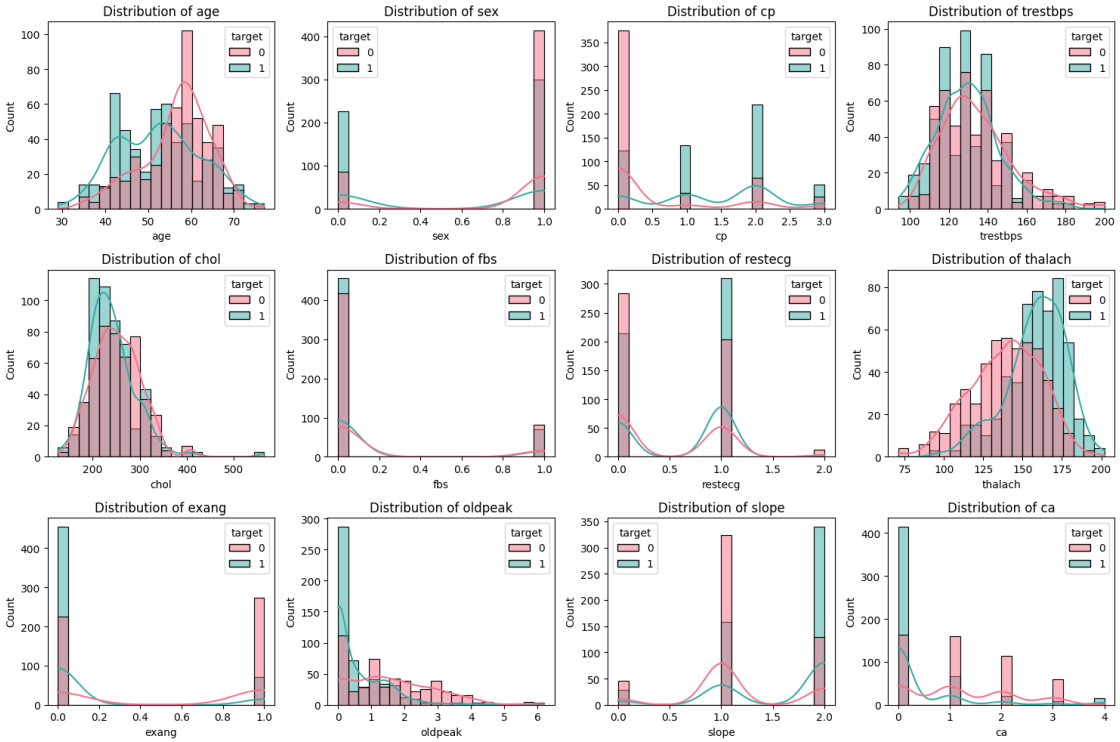
[9]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

[10]:
```
print("\nData types of columns:")
df.dtypes
```

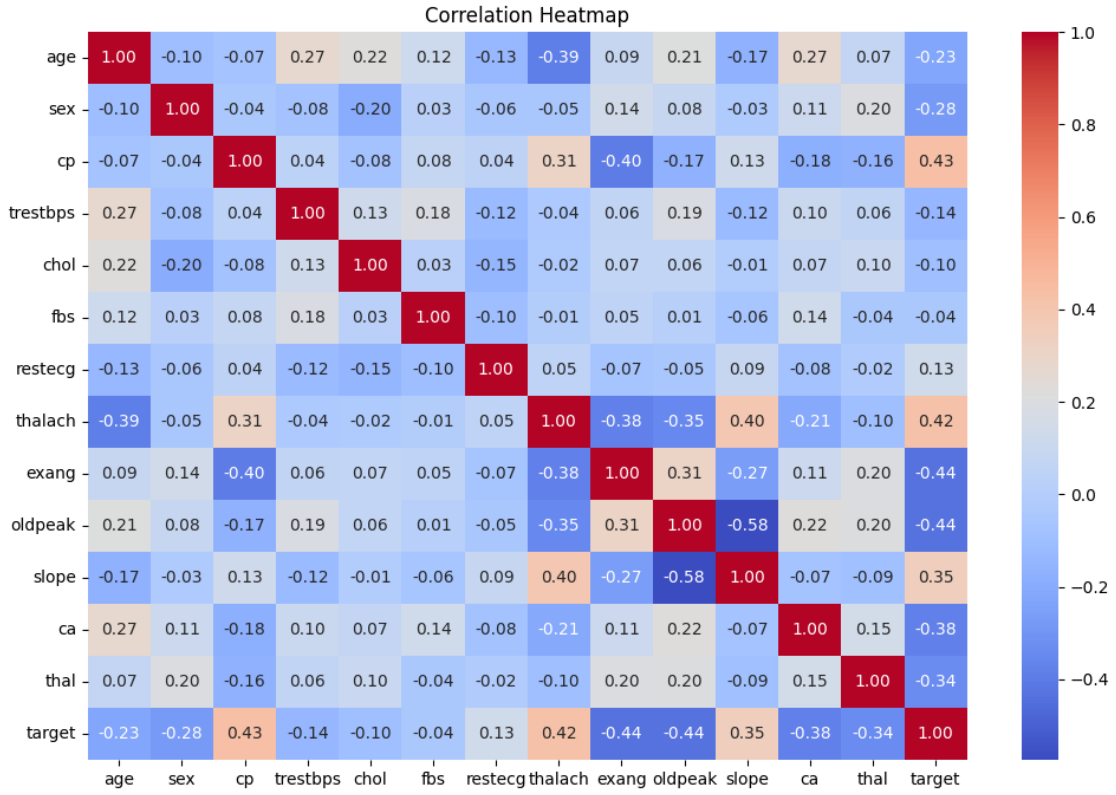
```
Data types of columns:
```

[10]:
```
age           int64
sex           int64
cp            int64
trestbps      int64
chol          int64
fbs           int64
restecg       int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
thal          int64
target        int64
dtype: object
```

[52]:
```
# Visualize the distribution of each feature
plt.figure(figsize=(15, 10))
for i, column in enumerate(df.columns[:-1], 1):
    if i <= 12:  # Check if i exceeds the number of columns in subplot grid
        plt.subplot(3, 4, i)
        sns.histplot(data=df, x=column, hue='target', kde=True, bins=20,␣
 ↪palette='husl')
        plt.title(f'Distribution of {column}')
        plt.xlabel(column)
        plt.ylabel('Count')
plt.tight_layout()
plt.show()
```




[54]:
```
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()
```




[55]:
```
X = df.drop('target', axis=1).values
y = df['target'].values
```

[16]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[17]:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

[18]:
```
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)
```

[19]:
```
model = Sequential([
    Conv1D(filters=32, kernel_size=3, activation='relu', input_shape=X_train[0].
↪shape),
    MaxPooling1D(pool_size=2),
    Dropout(0.2),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
```

```
])
```

[20]:
```
model.compile(optimizer='adam', loss='binary_crossentropy',␣
↪metrics=['accuracy'])
```

[21]:
```
history = model.fit(X_train, y_train, epochs=50, batch_size=32,␣
↪validation_data=(X_test, y_test), verbose=1)
```

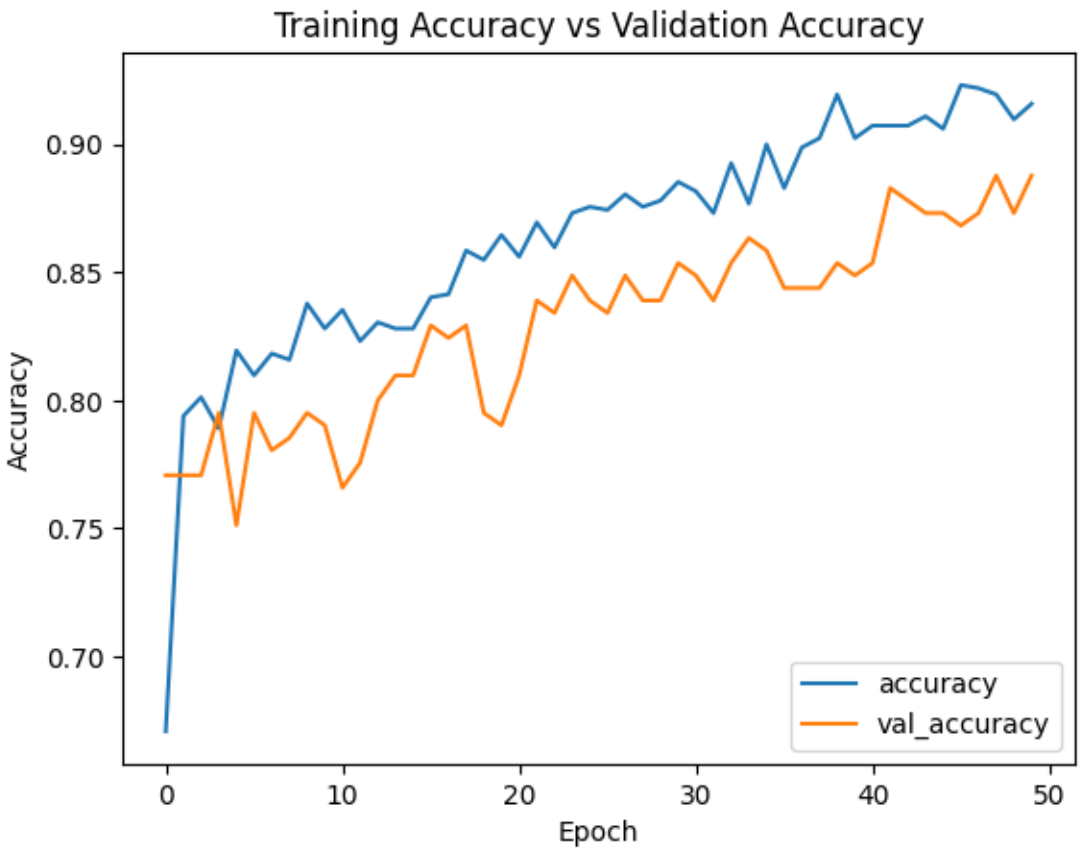
```
Epoch 1/50
26/26 [==============================] - 3s 22ms/step - loss: 0.6252 - accuracy:
0.6707 - val_loss: 0.5300 - val_accuracy: 0.7707
Epoch 2/50
26/26 [==============================] - 0s 7ms/step - loss: 0.4872 - accuracy:
0.7939 - val_loss: 0.4688 - val_accuracy: 0.7707
Epoch 3/50
26/26 [==============================] - 0s 7ms/step - loss: 0.4366 - accuracy:
0.8012 - val_loss: 0.4445 - val_accuracy: 0.7707
Epoch 4/50
26/26 [==============================] - 0s 7ms/step - loss: 0.4247 - accuracy:
0.7890 - val_loss: 0.4434 - val_accuracy: 0.7951
Epoch 5/50
26/26 [==============================] - 0s 7ms/step - loss: 0.4054 - accuracy:
0.8195 - val_loss: 0.4330 - val_accuracy: 0.7512
Epoch 6/50
26/26 [==============================] - 0s 7ms/step - loss: 0.4009 - accuracy:
0.8098 - val_loss: 0.4289 - val_accuracy: 0.7951
Epoch 7/50
26/26 [==============================] - 0s 6ms/step - loss: 0.3985 - accuracy:
0.8183 - val_loss: 0.4146 - val_accuracy: 0.7805
Epoch 8/50
26/26 [==============================] - 0s 8ms/step - loss: 0.3900 - accuracy:
0.8159 - val_loss: 0.4281 - val_accuracy: 0.7854
Epoch 9/50
26/26 [==============================] - 0s 6ms/step - loss: 0.3885 - accuracy:
0.8378 - val_loss: 0.4120 - val_accuracy: 0.7951
Epoch 10/50
26/26 [==============================] - 0s 9ms/step - loss: 0.3908 - accuracy:
0.8280 - val_loss: 0.4060 - val_accuracy: 0.7902
Epoch 11/50
26/26 [==============================] - 0s 9ms/step - loss: 0.3733 - accuracy:
0.8354 - val_loss: 0.4097 - val_accuracy: 0.7659
Epoch 12/50
26/26 [==============================] - 0s 4ms/step - loss: 0.3771 - accuracy:
0.8232 - val_loss: 0.4027 - val_accuracy: 0.7756
Epoch 13/50
26/26 [==============================] - 0s 4ms/step - loss: 0.3712 - accuracy:
0.8305 - val_loss: 0.4036 - val_accuracy: 0.8000
```

```
Epoch 14/50
26/26 [==============================] - 0s 4ms/step - loss: 0.3712 - accuracy:
0.8280 - val_loss: 0.3988 - val_accuracy: 0.8098
Epoch 15/50
26/26 [==============================] - 0s 4ms/step - loss: 0.3567 - accuracy:
0.8280 - val_loss: 0.4001 - val_accuracy: 0.8098
Epoch 16/50
26/26 [==============================] - 0s 4ms/step - loss: 0.3479 - accuracy:
0.8402 - val_loss: 0.3896 - val_accuracy: 0.8293
Epoch 17/50
26/26 [==============================] - 0s 5ms/step - loss: 0.3584 - accuracy:
0.8415 - val_loss: 0.3844 - val_accuracy: 0.8244
Epoch 18/50
26/26 [==============================] - 0s 4ms/step - loss: 0.3415 - accuracy:
0.8585 - val_loss: 0.3772 - val_accuracy: 0.8293
Epoch 19/50
26/26 [==============================] - 0s 4ms/step - loss: 0.3338 - accuracy:
0.8549 - val_loss: 0.3993 - val_accuracy: 0.7951
Epoch 20/50
26/26 [==============================] - 0s 4ms/step - loss: 0.3260 - accuracy:
0.8646 - val_loss: 0.3814 - val_accuracy: 0.7902
Epoch 21/50
26/26 [==============================] - 0s 5ms/step - loss: 0.3301 - accuracy:
0.8561 - val_loss: 0.3786 - val_accuracy: 0.8098
Epoch 22/50
26/26 [==============================] - 0s 5ms/step - loss: 0.3160 - accuracy:
0.8695 - val_loss: 0.3621 - val_accuracy: 0.8390
Epoch 23/50
26/26 [==============================] - 0s 4ms/step - loss: 0.3275 - accuracy:
0.8598 - val_loss: 0.3647 - val_accuracy: 0.8341
Epoch 24/50
26/26 [==============================] - 0s 4ms/step - loss: 0.3154 - accuracy:
0.8732 - val_loss: 0.3597 - val_accuracy: 0.8488
Epoch 25/50
26/26 [==============================] - 0s 4ms/step - loss: 0.3108 - accuracy:
0.8756 - val_loss: 0.3502 - val_accuracy: 0.8390
Epoch 26/50
26/26 [==============================] - 0s 5ms/step - loss: 0.3044 - accuracy:
0.8744 - val_loss: 0.3434 - val_accuracy: 0.8341
Epoch 27/50
26/26 [==============================] - 0s 4ms/step - loss: 0.2937 - accuracy:
0.8805 - val_loss: 0.3495 - val_accuracy: 0.8488
Epoch 28/50
26/26 [==============================] - 0s 4ms/step - loss: 0.2941 - accuracy:
0.8756 - val_loss: 0.3383 - val_accuracy: 0.8390
Epoch 29/50
26/26 [==============================] - 0s 5ms/step - loss: 0.2891 - accuracy:
0.8780 - val_loss: 0.3390 - val_accuracy: 0.8390
```

```
Epoch 30/50
26/26 [==============================] - 0s 5ms/step - loss: 0.2867 - accuracy:
0.8854 - val_loss: 0.3276 - val_accuracy: 0.8537
Epoch 31/50
26/26 [==============================] - 0s 6ms/step - loss: 0.2879 - accuracy:
0.8817 - val_loss: 0.3250 - val_accuracy: 0.8488
Epoch 32/50
26/26 [==============================] - 0s 6ms/step - loss: 0.2966 - accuracy:
0.8732 - val_loss: 0.3215 - val_accuracy: 0.8390
Epoch 33/50
26/26 [==============================] - 0s 7ms/step - loss: 0.2793 - accuracy:
0.8927 - val_loss: 0.3198 - val_accuracy: 0.8537
Epoch 34/50
26/26 [==============================] - 0s 6ms/step - loss: 0.2741 - accuracy:
0.8768 - val_loss: 0.3106 - val_accuracy: 0.8634
Epoch 35/50
26/26 [==============================] - 0s 6ms/step - loss: 0.2704 - accuracy:
0.9000 - val_loss: 0.3120 - val_accuracy: 0.8585
Epoch 36/50
26/26 [==============================] - 0s 7ms/step - loss: 0.2647 - accuracy:
0.8829 - val_loss: 0.3083 - val_accuracy: 0.8439
Epoch 37/50
26/26 [==============================] - 0s 6ms/step - loss: 0.2676 - accuracy:
0.8988 - val_loss: 0.3002 - val_accuracy: 0.8439
Epoch 38/50
26/26 [==============================] - 0s 6ms/step - loss: 0.2446 - accuracy:
0.9024 - val_loss: 0.3046 - val_accuracy: 0.8439
Epoch 39/50
26/26 [==============================] - 0s 6ms/step - loss: 0.2298 - accuracy:
0.9195 - val_loss: 0.2932 - val_accuracy: 0.8537
Epoch 40/50
26/26 [==============================] - 0s 6ms/step - loss: 0.2433 - accuracy:
0.9024 - val_loss: 0.2905 - val_accuracy: 0.8488
Epoch 41/50
26/26 [==============================] - 0s 6ms/step - loss: 0.2381 - accuracy:
0.9073 - val_loss: 0.2792 - val_accuracy: 0.8537
Epoch 42/50
26/26 [==============================] - 0s 6ms/step - loss: 0.2416 - accuracy:
0.9073 - val_loss: 0.2802 - val_accuracy: 0.8829
Epoch 43/50
26/26 [==============================] - 0s 6ms/step - loss: 0.2317 - accuracy:
0.9073 - val_loss: 0.2795 - val_accuracy: 0.8780
Epoch 44/50
26/26 [==============================] - 0s 7ms/step - loss: 0.2244 - accuracy:
0.9110 - val_loss: 0.2661 - val_accuracy: 0.8732
Epoch 45/50
26/26 [==============================] - 0s 8ms/step - loss: 0.2380 - accuracy:
0.9061 - val_loss: 0.2669 - val_accuracy: 0.8732
```

```
Epoch 46/50
26/26 [==============================] - 0s 6ms/step - loss: 0.2153 - accuracy:
0.9232 - val_loss: 0.2578 - val_accuracy: 0.8683
Epoch 47/50
26/26 [==============================] - 0s 5ms/step - loss: 0.2137 - accuracy:
0.9220 - val_loss: 0.2599 - val_accuracy: 0.8732
Epoch 48/50
26/26 [==============================] - 0s 5ms/step - loss: 0.2201 - accuracy:
0.9195 - val_loss: 0.2586 - val_accuracy: 0.8878
Epoch 49/50
26/26 [==============================] - 0s 4ms/step - loss: 0.2108 - accuracy:
0.9098 - val_loss: 0.2586 - val_accuracy: 0.8732
Epoch 50/50
26/26 [==============================] - 0s 4ms/step - loss: 0.2131 - accuracy:
0.9159 - val_loss: 0.2472 - val_accuracy: 0.8878
```

[22]:
```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.title('Training Accuracy vs Validation Accuracy')
plt.show()
```




[23]:
```
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f'Test Loss: {loss:.3f}')
print(f'Test Accuracy: {accuracy:.3f}')
```

```
Test Loss: 0.247
Test Accuracy: 0.888
```

[31]:
```
from tensorflow.keras.models import Sequential
```

[41]:
```
import tensorflow as tf
```

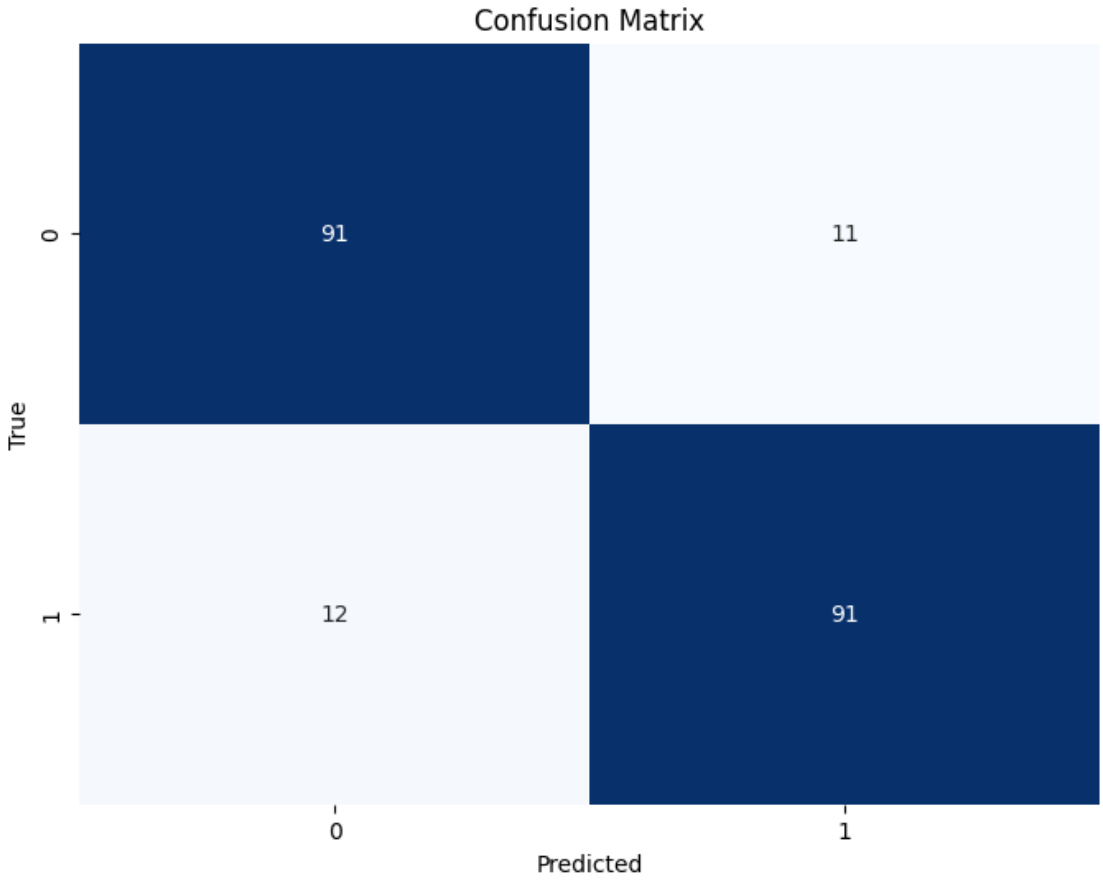
[49]:
```
# Get predicted probabilities
y_pred_prob = model.predict(X_test)

# Convert probabilities to binary predictions
y_pred = (y_pred_prob > 0.5).astype(int)
y_test_binary = y_test.astype(int)

# Confusion matrix and classification report
cm = confusion_matrix(y_test_binary, y_pred)
```

```
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='g', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

```
7/7 [==============================] - 0s 3ms/step
```




[50]:
```
print(classification_report(y_test_binary, y_pred))
```

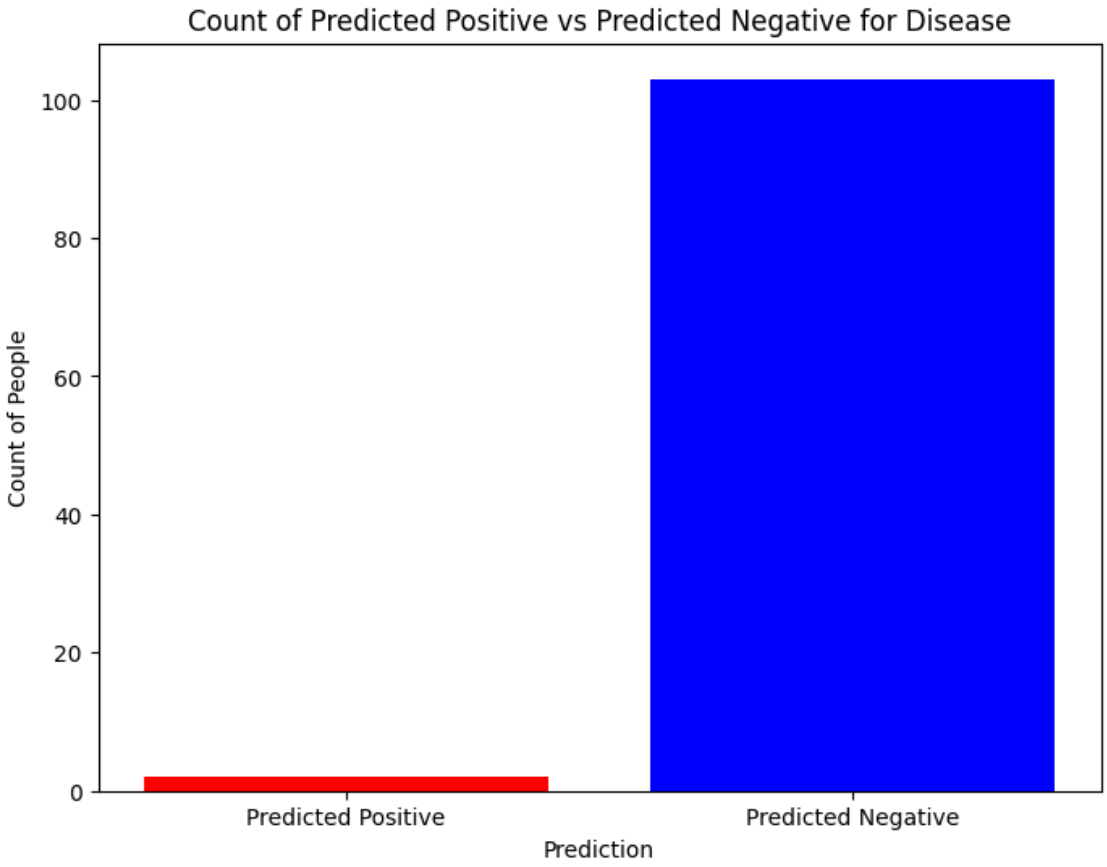
```
              precision    recall  f1-score   support

           0       0.88      0.89      0.89       102
           1       0.89      0.88      0.89       103

    accuracy                           0.89       205
   macro avg       0.89      0.89      0.89       205
weighted avg       0.89      0.89      0.89       205
```

[68]:
```
# Count the number of predicted positive and negative instances
positive_count = predicted_positive.shape[1]
negative_count = predicted_negative.shape[0]

# Create bar plot
plt.figure(figsize=(8, 6))
plt.bar(['Predicted Positive', 'Predicted Negative'], [positive_count,␣
 ↪negative_count], color=['red', 'blue'])
plt.xlabel('Prediction')
plt.ylabel('Count of People')
plt.title('Count of Predicted Positive vs Predicted Negative for Disease')
plt.show()
```

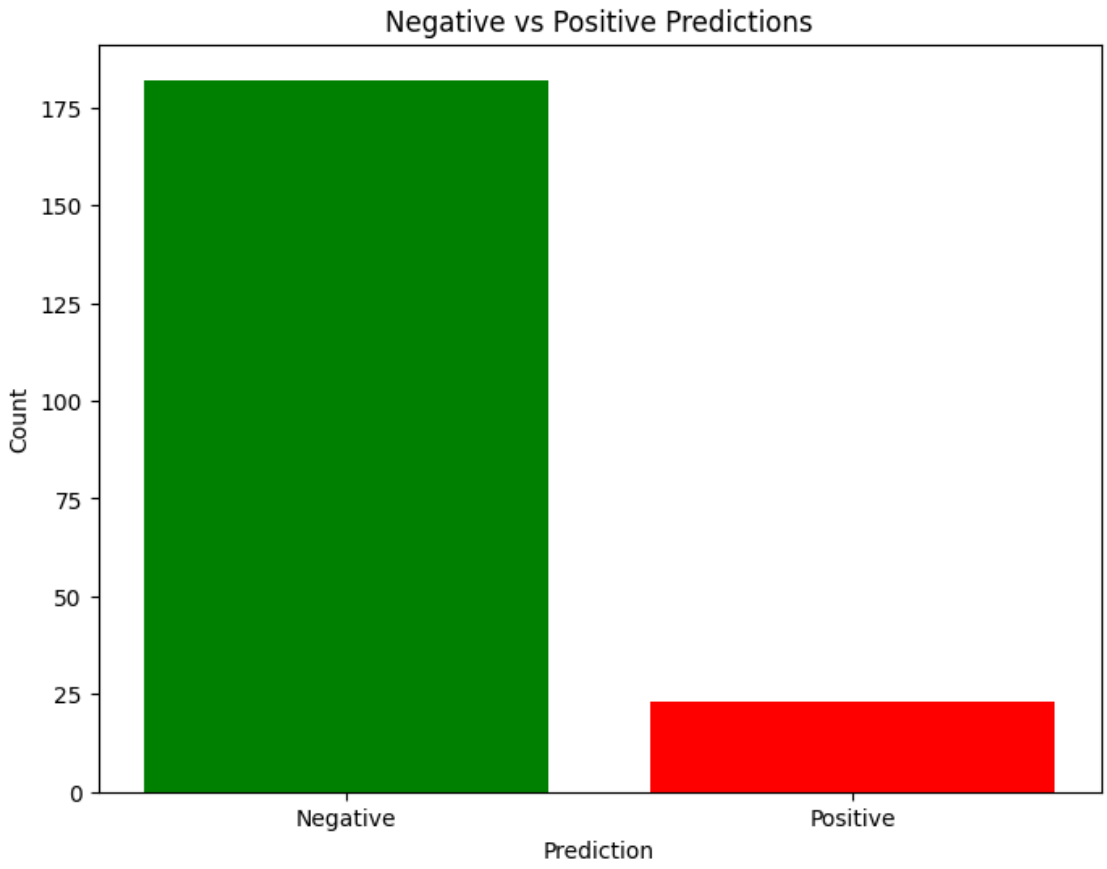



[70]:
```
# Calculate correct and incorrect predictions
correct_predictions = results_df['Actual'] == results_df['Predicted']
incorrect_predictions = results_df['Actual'] != results_df['Predicted']

# Count correct and incorrect predictions
correct_count = correct_predictions.sum()
```

```
incorrect_count = incorrect_predictions.sum()

# Create a bar plot
plt.figure(figsize=(8, 6))
plt.bar(['Negative', 'Positive'], [correct_count, incorrect_count],␣
 ↪color=['green', 'red'])
plt.title('Negative vs Positive Predictions')
plt.xlabel('Prediction')
plt.ylabel('Count')
plt.show()
```




[57]:
```
# Display actual vs predicted values
results_df = pd.DataFrame({'Actual': y_test_binary, 'Predicted': y_pred.
 ↪flatten()})
results_df
```

[57]:

| | Actual | Predicted |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 1 | 1 |

```
2          0           0
3          1           1
4          0           0
..       …           …
200        1           1
201        1           1
202        1           1
203        0           0
204        0           1

[205 rows x 2 columns]
```