Machine learning mini project reg no 185

October 20, 2024

[1]:
```
pip install tensorflow
```

```
Requirement already satisfied: tensorflow in c:\users\crazy\anaconda3\lib\site-
packages (2.17.0)
Requirement already satisfied: tensorflow-intel==2.17.0 in
c:\users\crazy\anaconda3\lib\site-packages (from tensorflow) (2.17.0)
Requirement already satisfied: absl-py>=1.0.0 in
c:\users\crazy\anaconda3\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in
c:\users\crazy\anaconda3\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in
c:\users\crazy\anaconda3\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in
c:\users\crazy\anaconda3\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in
c:\users\crazy\anaconda3\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in
c:\users\crazy\anaconda3\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (3.11.0)
Requirement already satisfied: libclang>=13.0.0 in
c:\users\crazy\anaconda3\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.3.1 in
c:\users\crazy\anaconda3\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (0.4.1)
Requirement already satisfied: opt-einsum>=2.3.2 in
c:\users\crazy\anaconda3\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (3.4.0)
Requirement already satisfied: packaging in c:\users\crazy\anaconda3\lib\site-
packages (from tensorflow-intel==2.17.0->tensorflow) (23.2)
Requirement already satisfied:
protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3
in c:\users\crazy\anaconda3\lib\site-packages (from tensorflow-
```

```
intel==2.17.0->tensorflow) (3.20.3)
Requirement already satisfied: requests<3,>=2.21.0 in
c:\users\crazy\anaconda3\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (2.32.2)
Requirement already satisfied: setuptools in c:\users\crazy\anaconda3\lib\site-
packages (from tensorflow-intel==2.17.0->tensorflow) (69.5.1)
Requirement already satisfied: six>=1.12.0 in c:\users\crazy\anaconda3\lib\site-
packages (from tensorflow-intel==2.17.0->tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in
c:\users\crazy\anaconda3\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
c:\users\crazy\anaconda3\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (4.11.0)
Requirement already satisfied: wrapt>=1.11.0 in
c:\users\crazy\anaconda3\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (1.14.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
c:\users\crazy\anaconda3\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (1.67.0)
Requirement already satisfied: tensorboard<2.18,>=2.17 in
c:\users\crazy\anaconda3\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (2.17.1)
Requirement already satisfied: keras>=3.2.0 in
c:\users\crazy\anaconda3\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (3.6.0)
Requirement already satisfied: numpy<2.0.0,>=1.26.0 in
c:\users\crazy\anaconda3\lib\site-packages (from tensorflow-
intel==2.17.0->tensorflow) (1.26.4)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
c:\users\crazy\anaconda3\lib\site-packages (from astunparse>=1.6.0->tensorflow-
intel==2.17.0->tensorflow) (0.43.0)
Requirement already satisfied: rich in c:\users\crazy\anaconda3\lib\site-
packages (from keras>=3.2.0->tensorflow-intel==2.17.0->tensorflow) (13.3.5)
Requirement already satisfied: namex in c:\users\crazy\anaconda3\lib\site-
packages (from keras>=3.2.0->tensorflow-intel==2.17.0->tensorflow) (0.0.8)
Requirement already satisfied: optree in c:\users\crazy\anaconda3\lib\site-
packages (from keras>=3.2.0->tensorflow-intel==2.17.0->tensorflow) (0.13.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
c:\users\crazy\anaconda3\lib\site-packages (from
requests<3,>=2.21.0->tensorflow-intel==2.17.0->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in
c:\users\crazy\anaconda3\lib\site-packages (from
requests<3,>=2.21.0->tensorflow-intel==2.17.0->tensorflow) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
c:\users\crazy\anaconda3\lib\site-packages (from
requests<3,>=2.21.0->tensorflow-intel==2.17.0->tensorflow) (2.2.2)
Requirement already satisfied: certifi>=2017.4.17 in
```

```
c:\users\crazy\anaconda3\lib\site-packages (from
requests<3,>=2.21.0->tensorflow-intel==2.17.0->tensorflow) (2024.7.4)
Requirement already satisfied: markdown>=2.6.8 in
c:\users\crazy\anaconda3\lib\site-packages (from
tensorboard<2.18,>=2.17->tensorflow-intel==2.17.0->tensorflow) (3.4.1)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
c:\users\crazy\anaconda3\lib\site-packages (from
tensorboard<2.18,>=2.17->tensorflow-intel==2.17.0->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
c:\users\crazy\anaconda3\lib\site-packages (from
tensorboard<2.18,>=2.17->tensorflow-intel==2.17.0->tensorflow) (3.0.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in
c:\users\crazy\anaconda3\lib\site-packages (from
werkzeug>=1.0.1->tensorboard<2.18,>=2.17->tensorflow-intel==2.17.0->tensorflow)
(2.1.3)
Requirement already satisfied: markdown-it-py<3.0.0,>=2.2.0 in
c:\users\crazy\anaconda3\lib\site-packages (from rich->keras>=3.2.0->tensorflow-
intel==2.17.0->tensorflow) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
c:\users\crazy\anaconda3\lib\site-packages (from rich->keras>=3.2.0->tensorflow-
intel==2.17.0->tensorflow) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in c:\users\crazy\anaconda3\lib\site-
packages (from markdown-it-py<3.0.0,>=2.2.0->rich->keras>=3.2.0->tensorflow-
intel==2.17.0->tensorflow) (0.1.0)
Note: you may need to restart the kernel to use updated packages.
```

[85]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,␣
↪recall_score, roc_auc_score, roc_curve
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
```

[29]:
```
data = pd.read_csv('C://Users//crazy//Downloads//ml project.csv')

# Display the first few rows of the dataset
print(data.head())
```

```
   Age  Sex  Chest pain type   BP  Cholesterol  FBS over 120  EKG results  \
0   70    1                4  130          322             0            2
1   67    0                3  115          564             0            2
2   57    1                2  124          261             0            0
3   64    1                4  128          263             0            0
```

```
4   74    0                2  120          269             0            2

   Max HR  Exercise angina  ST depression  Slope of ST  \
0     109                0            2.4            2
1     160                0            1.6            2
2     141                0            0.3            1
3     105                1            0.2            2
4     121                1            0.2            1

   Number of vessels fluro  Thallium Heart Disease
0                        3         3      Presence
1                        0         7       Absence
2                        0         7      Presence
3                        1         7       Absence
4                        1         3       Absence
```

[67]:
```
data.describe()
```

[67]:
```
              Age         Sex  Chest pain type          BP  Cholesterol  \
count  270.000000  270.000000       270.000000  270.000000   270.000000
mean    54.433333    0.677778         3.174074  131.344444   249.659259
std      9.109067    0.468195         0.950090   17.861608    51.686237
min     29.000000    0.000000         1.000000   94.000000   126.000000
25%     48.000000    0.000000         3.000000  120.000000   213.000000
50%     55.000000    1.000000         3.000000  130.000000   245.000000
75%     61.000000    1.000000         4.000000  140.000000   280.000000
max     77.000000    1.000000         4.000000  200.000000   564.000000

       FBS over 120  EKG results      Max HR  Exercise angina  ST depression  \
count    270.000000   270.000000  270.000000       270.000000      270.00000
mean       0.148148     1.022222  149.677778         0.329630        1.05000
std        0.355906     0.997891   23.165717         0.470952        1.14521
min        0.000000     0.000000   71.000000         0.000000        0.00000
25%        0.000000     0.000000  133.000000         0.000000        0.00000
50%        0.000000     2.000000  153.500000         0.000000        0.80000
75%        0.000000     2.000000  166.000000         1.000000        1.60000
max        1.000000     2.000000  202.000000         1.000000        6.20000

       Slope of ST  Number of vessels fluro    Thallium
count   270.000000               270.000000  270.000000
mean      1.585185                 0.670370    4.696296
std       0.614390                 0.943896    1.940659
min       1.000000                 0.000000    3.000000
25%       1.000000                 0.000000    3.000000
50%       2.000000                 0.000000    3.000000
75%       2.000000                 1.000000    7.000000
max       3.000000                 3.000000    7.000000
```

[69]:
```
# TO FIND THE NULL VALUES
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 270 entries, 0 to 269
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Age                     270 non-null    int64
 1   Sex                     270 non-null    int64
 2   Chest pain type         270 non-null    int64
 3   BP                      270 non-null    int64
 4   Cholesterol             270 non-null    int64
 5   FBS over 120            270 non-null    int64
 6   EKG results             270 non-null    int64
 7   Max HR                  270 non-null    int64
 8   Exercise angina         270 non-null    int64
 9   ST depression           270 non-null    float64
 10  Slope of ST             270 non-null    int64
 11  Number of vessels fluro 270 non-null    int64
 12  Thallium                270 non-null    int64
 13  Heart Disease           270 non-null    object
dtypes: float64(1), int64(12), object(1)
memory usage: 29.7+ KB
```

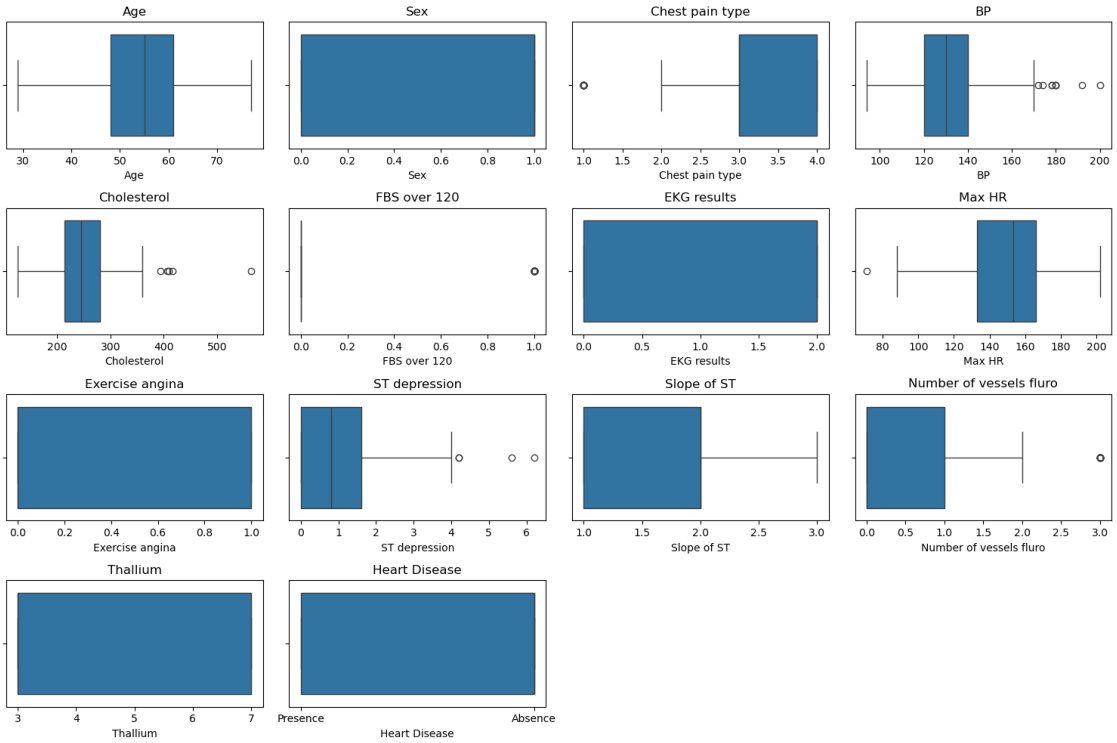
[71]:
```
# TO VERIFY THE NULL VALUES
data.isna().sum()
```

[71]:
```
Age                     0
Sex                     0
Chest pain type         0
BP                      0
Cholesterol             0
FBS over 120            0
EKG results             0
Max HR                  0
Exercise angina         0
ST depression           0
Slope of ST             0
Number of vessels fluro 0
Thallium                0
Heart Disease           0
dtype: int64
```

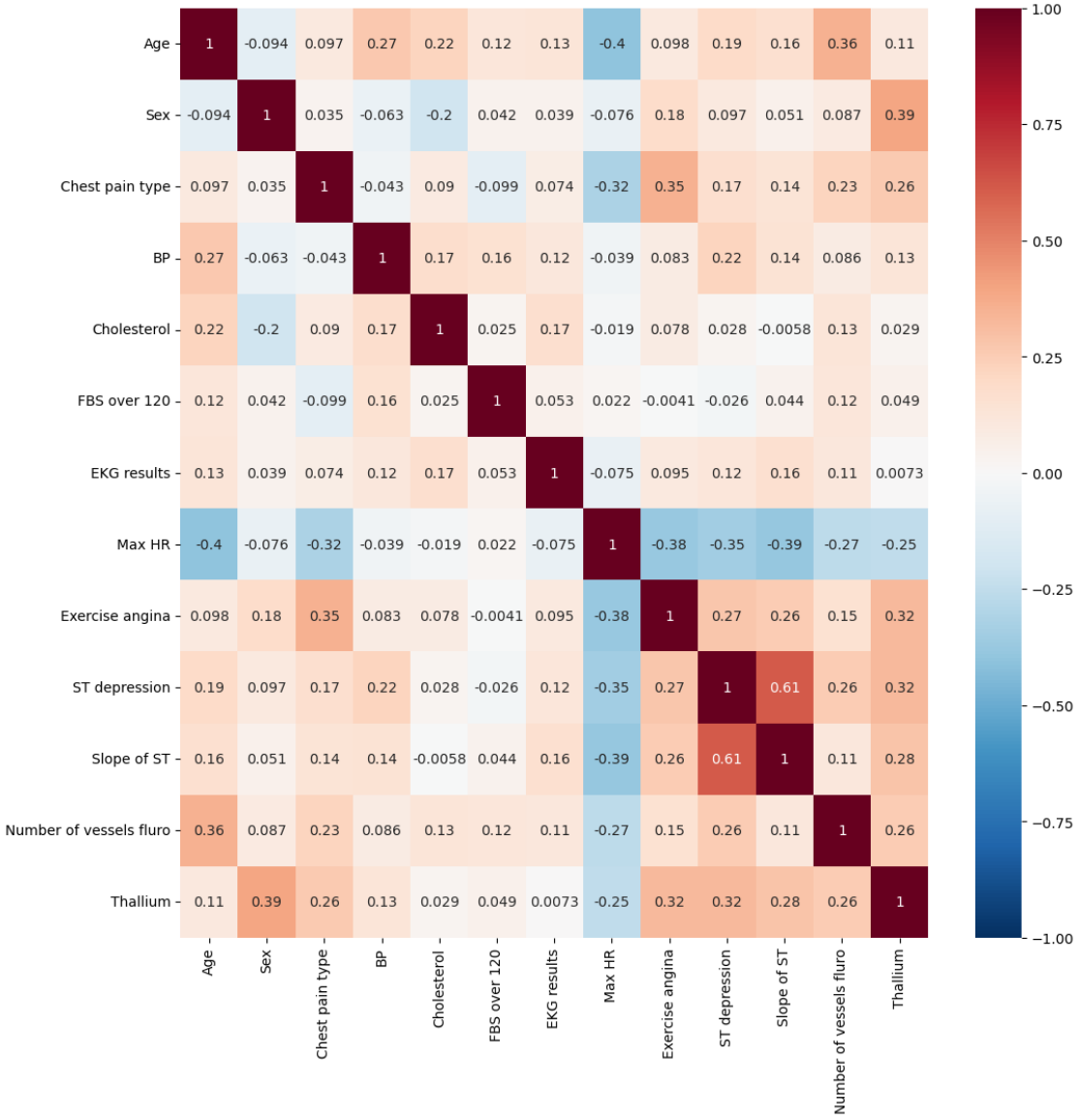
[73]:
```
# TO FIND THE DUPLICATE VALUES
data.duplicated().sum()
```

[73]:
```
0
```

[79]:
```
columns = ['Age', 'Sex', 'Chest pain type', 'BP', 'Cholesterol', 'FBS over 120',
       'EKG results', 'Max HR', 'Exercise angina', 'ST depression',
       'Slope of ST', 'Number of vessels fluro', 'Thallium', 'Heart Disease']
plt.figure(figsize=(15, 10))

# Adjust grid size to 3 rows and 3 columns (7 subplots in total)
for i, col in enumerate(columns):
    plt.subplot(4, 4, i + 1)
    sns.boxplot(x=col, data=data)
    plt.title(col)

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



[81]:
```
# Finding corelation in  the dataset
numeric_data = data.select_dtypes(include=[np.number])

# Create the heatmap
plt.figure(figsize=(12, 12))
sns.heatmap(numeric_data.corr(), vmin=-1.0, center=0, cmap='RdBu_r', annot=True)
plt.show()
```



[31]:
```
print("Column names:", data.columns)

# Strip any leading/trailing spaces from column names
data.columns = data.columns.str.strip()
```

```
Column names: Index(['Age', 'Sex', 'Chest pain type', 'BP', 'Cholesterol', 'FBS
over 120',
       'EKG results', 'Max HR', 'Exercise angina', 'ST depression',
       'Slope of ST', 'Number of vessels fluro', 'Thallium', 'Heart Disease'],
      dtype='object')
```

[35]:
```
label_column = data.columns[-1]  # Assumes the last column is the label

# Data Preprocessing
X = data.drop(label_column, axis=1)
y = data[label_column]

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[37]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Logistic Regression using GridSearchCV to find the best parameters
log_reg = LogisticRegression()
```

[45]:
```
param_grid = {
    'C': [0.1, 1, 10, 100],
    'penalty': ['l1', 'l2'],  # L1 for Lasso, L2 for Ridge
    'solver': ['liblinear']   # Works with l1 and l2 penalties
}

# Grid search for best parameters
grid_search = GridSearchCV(log_reg, param_grid, cv=5, scoring='accuracy',␣
 ↪verbose=1)
grid_search.fit(X_train_scaled, y_train)

# Best Parameters from GridSearch
print("Best Parameters: ", grid_search.best_params_)
best_log_reg = grid_search.best_estimator_
y_pred = best_log_reg.predict(X_test_scaled)
```

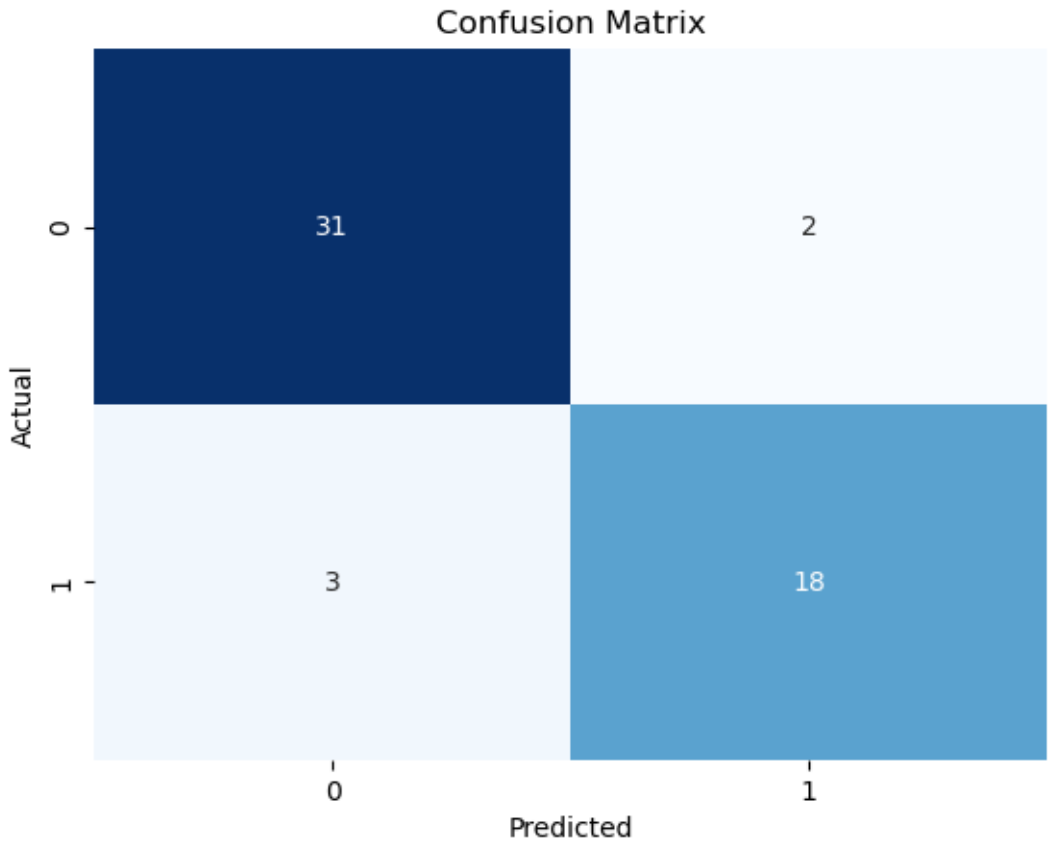
```
Fitting 5 folds for each of 8 candidates, totalling 40 fits
Best Parameters:  {'C': 1, 'penalty': 'l2', 'solver': 'liblinear'}
```

[49]:
```
# Model Evaluation
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, pos_label='Presence')  # Adjust␣
 ↪pos_label
recall = recall_score(y_test, y_pred, pos_label='Presence')  # Adjust pos_label

# Get predicted probabilities for the positive class
y_pred_prob = best_log_reg.predict_proba(X_test_scaled)[:, 1]  # Probability of␣
 ↪the 'Presence' class

# Compute ROC AUC
```

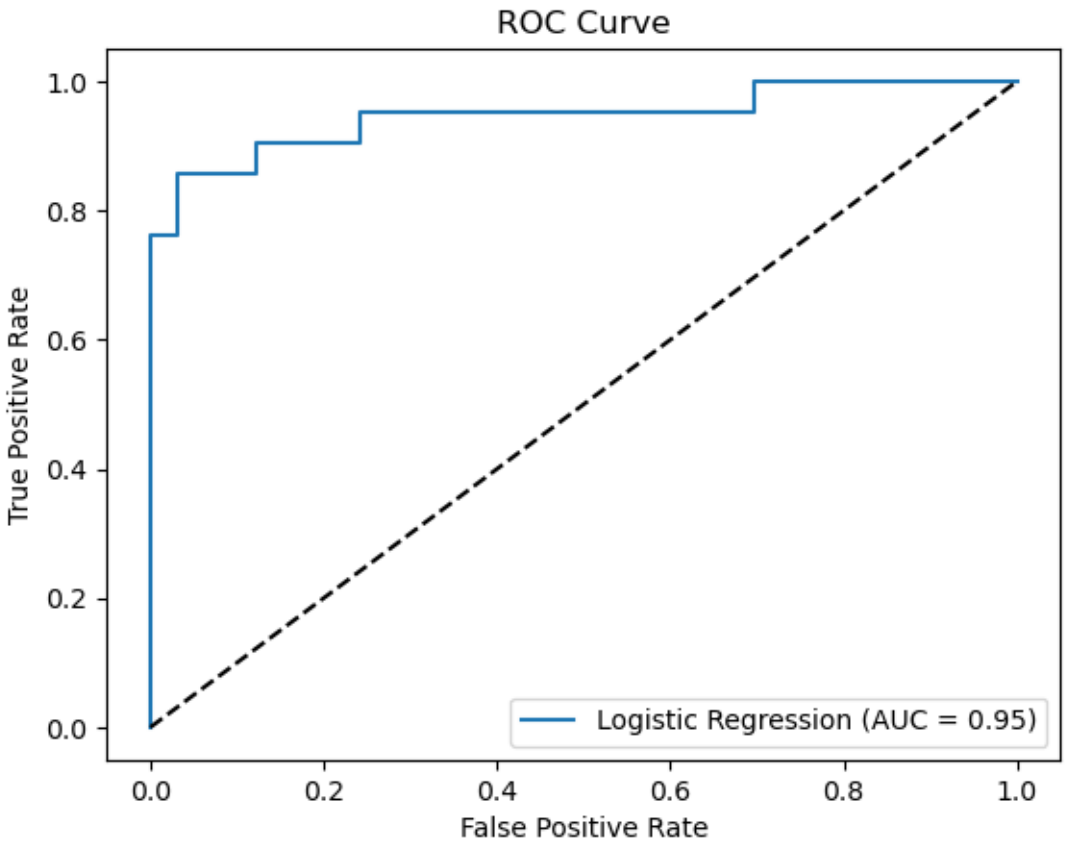
```
roc_auc = roc_auc_score(y_test.map({'Absence': 0, 'Presence': 1}), y_pred_prob)

print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"ROC-AUC: {roc_auc:.2f}")
```

```
Accuracy: 0.91
Precision: 0.90
Recall: 0.86
ROC-AUC: 0.95
```

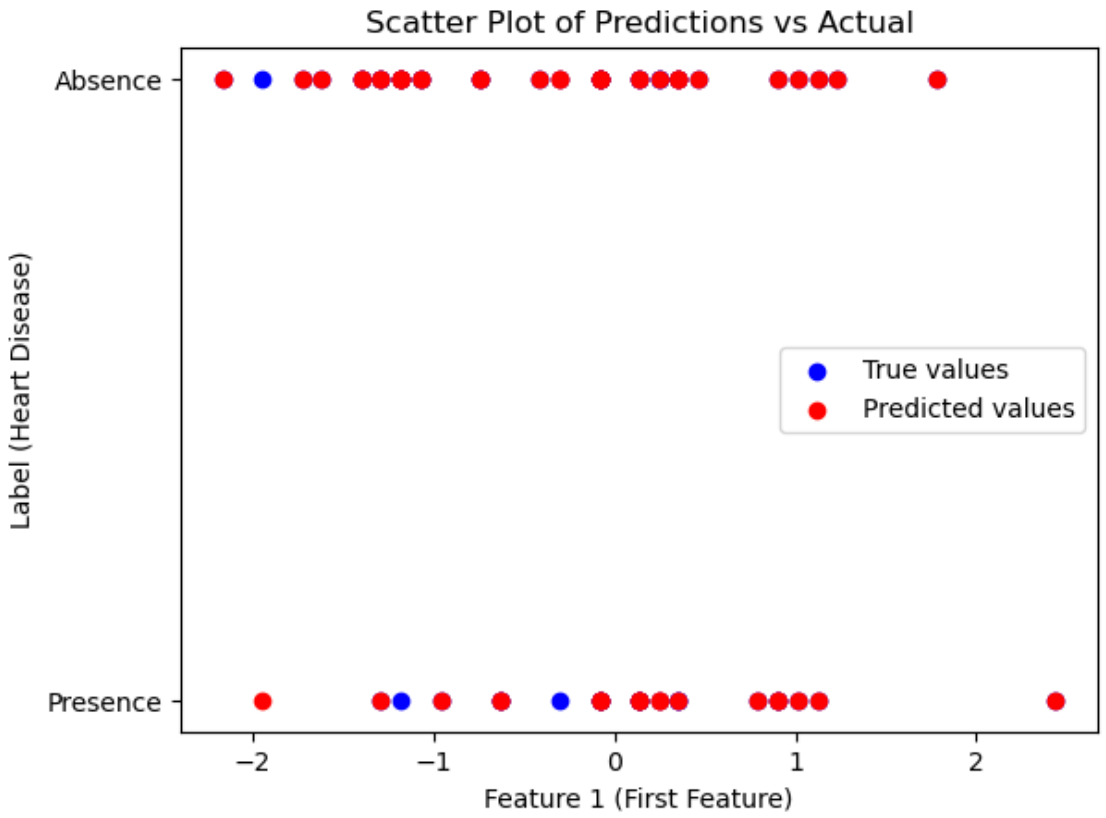
[51]:
```
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



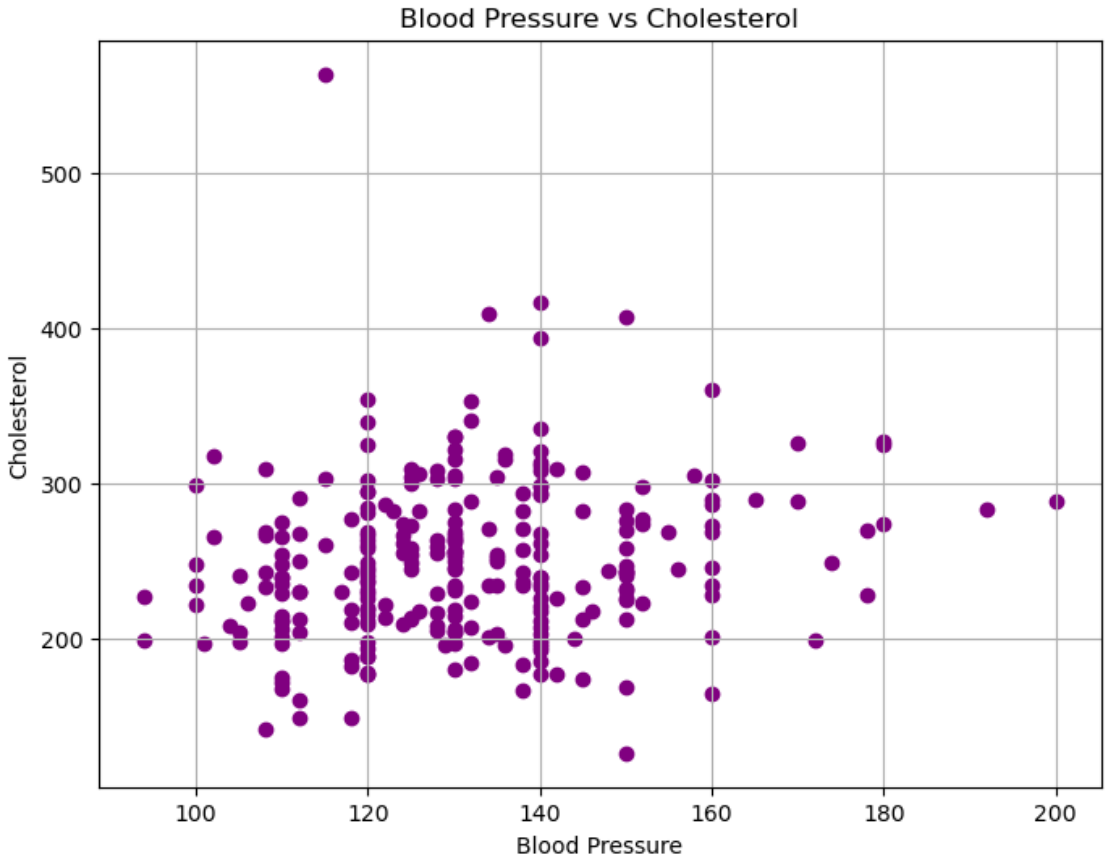
[55]:
```
# Get predicted probabilities for the positive class
y_pred_prob = best_log_reg.predict_proba(X_test_scaled)[:, 1]  # Probability of␣
 ↪the 'Presence' class

# Map labels for ROC curve
y_test_numeric = y_test.map({'Absence': 0, 'Presence': 1})

# Compute ROC AUC
roc_auc = roc_auc_score(y_test_numeric, y_pred_prob)

# ROC Curve
fpr, tpr, thresholds = roc_curve(y_test_numeric, y_pred_prob, pos_label=1)  # 1␣
 ↪for 'Presence'
plt.plot(fpr, tpr, label=f'Logistic Regression (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc="lower right")
plt.show()
```



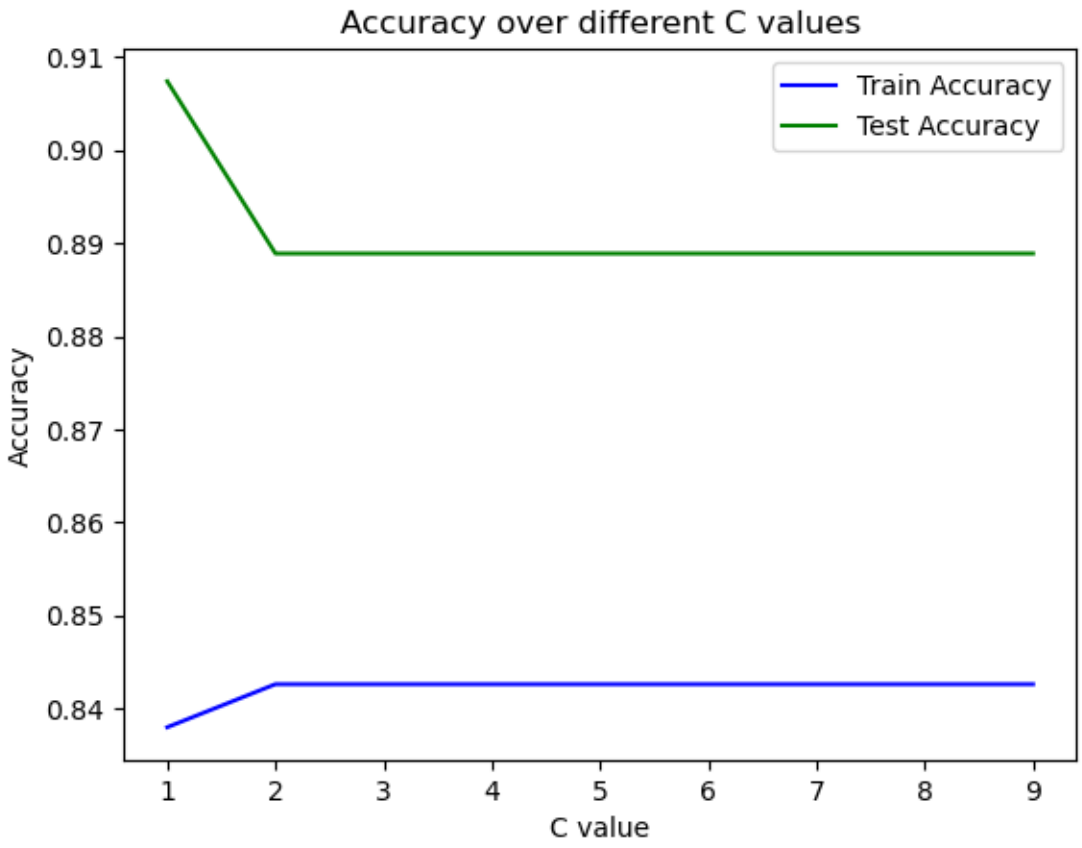
[57]:
```
# Scatter Plot
plt.scatter(X_test_scaled[:, 0], y_test, color="blue", label="True values")
plt.scatter(X_test_scaled[:, 0], y_pred, color="red", label="Predicted values")
plt.title('Scatter Plot of Predictions vs Actual')
plt.xlabel('Feature 1 (First Feature)')
plt.ylabel('Label (Heart Disease)')
plt.legend()
plt.show()
```



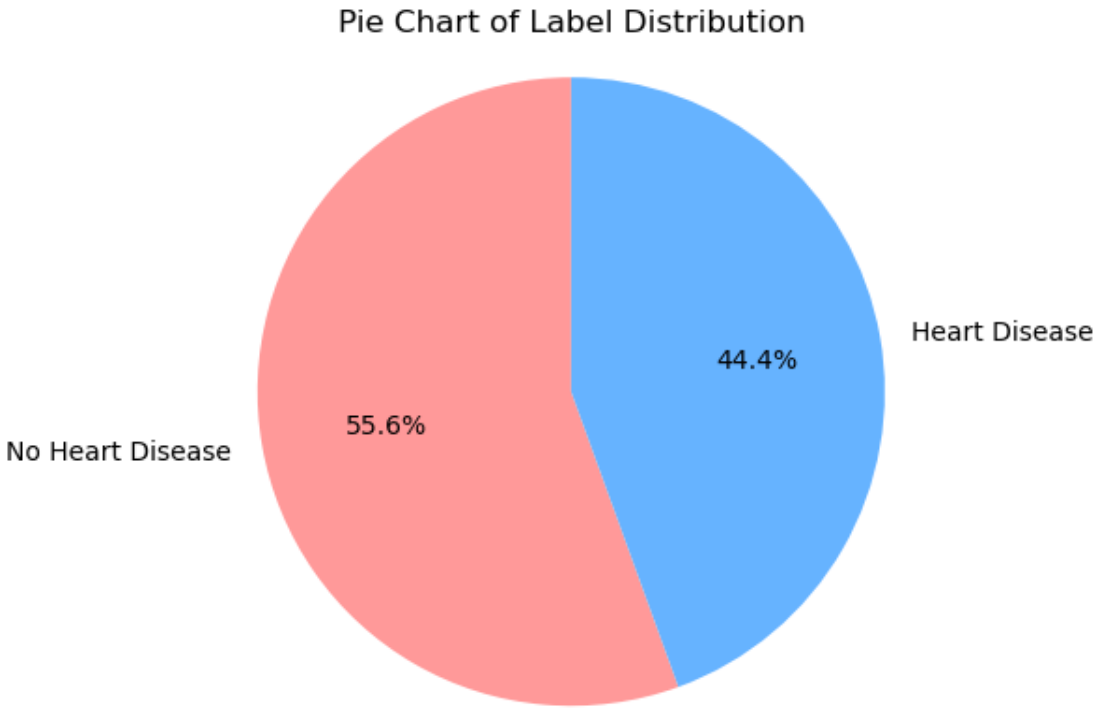
[77]:
```
plt.figure(figsize=(8,6))
plt.scatter(data['BP'], data['Cholesterol'], color='purple')
plt.title('Blood Pressure vs Cholesterol')
plt.xlabel('Blood Pressure')
plt.ylabel('Cholesterol')
plt.grid(True)
plt.show()
```



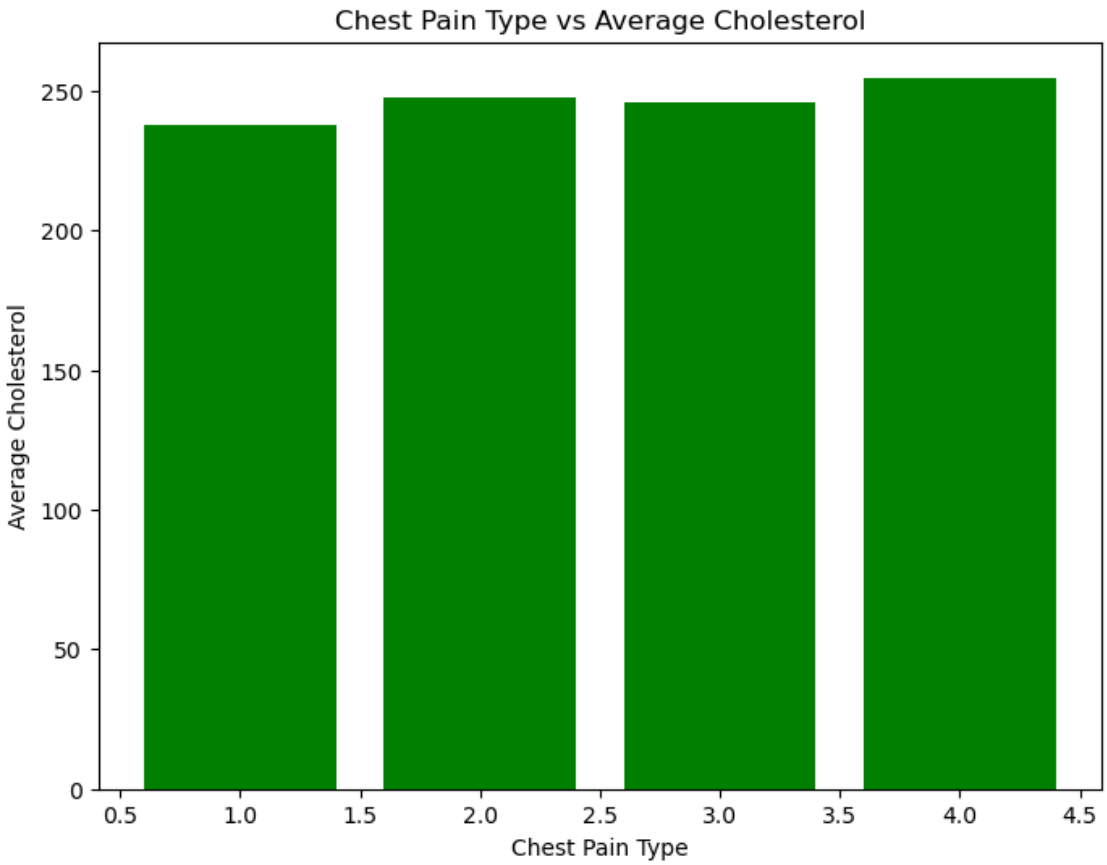
[59]:
```
# Line Plot for accuracy over training
train_acc = []
test_acc = []
for i in range(1, 10):
    model = LogisticRegression(C=i, solver='liblinear').fit(X_train_scaled,␣
 ↪y_train)
    train_acc.append(model.score(X_train_scaled, y_train))
    test_acc.append(model.score(X_test_scaled, y_test))

plt.plot(range(1, 10), train_acc, label='Train Accuracy', color='blue')
plt.plot(range(1, 10), test_acc, label='Test Accuracy', color='green')
plt.title('Accuracy over different C values')
plt.xlabel('C value')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



[65]:
```
# Pie Chart for label distribution
labels = ['No Heart Disease', 'Heart Disease']
sizes = data[label_column].value_counts()  # Use the identified label column

plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90,␣
 ↪colors=['#ff9999','#66b3ff'])
plt.axis('equal')
plt.title('Pie Chart of Label Distribution')
plt.show()
```



[75]:
```
chest_pain_groups = data.groupby('Chest pain type')['Cholesterol'].mean()
plt.figure(figsize=(8,6))
plt.bar(chest_pain_groups.index, chest_pain_groups.values, color='g')
plt.title('Chest Pain Type vs Average Cholesterol')
plt.xlabel('Chest Pain Type')
plt.ylabel('Average Cholesterol')
plt.show()
```


Chest Pain Type vs Average Cholesterol
Average Cholesterol
0
50
100
150
200
250
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
Chest Pain Type

[ ]: