Scccd

Week 2.1: Supervised Learning

Task 2.1: Building a Logistic Regression Model

Objective:

The objective of this task is to implement and evaluate a logistic regression model for binary classification using the Heart Disease Dataset from UCI. This dataset includes patient data with various attributes like age, cholesterol levels, and heart rate, along with a target variable indicating the presence of heart disease.

Dataset:

- **Source**: Heart Disease Dataset from UCI
- **Link**: Heart Disease Dataset on UCI

1. Data Preparation

Loading the Data

The dataset was loaded from three CSV files, which were then combined into a single DataFrame for analysis.

```
import pandas as pd

data = pd.read_csv('./Data/Dataset Heart Disease.csv')
data_1 = pd.read_csv('./Data/cleveland1.csv')
data_2 = pd.read_csv('./Data/cleveland2.csv')

combine = pd.concat([data, data_1, data_2], ignore_index=True)
```

Data Inspection

The initial few rows of each dataset were inspected to understand the structure and content.

```
print("First dataset:")
print(data.head())
print("\nSecond dataset:")
print(data_1.head())
print("\nThird dataset:")
print(data_2.head())

print("\nCombined dataset shape:", combine.shape)
combine.info()
```

Handling Missing Values

The dataset was checked for missing values, which were then filled with the mean value of their respective columns.

```
# Check for missing values
missing_values = (combine.isnull().sum() / len(combine)) * 100
print("\nMissing values percentage:\n", missing_values)

# Drop the 'Unnamed: 0' column
combine.drop('Unnamed: 0', axis=1, inplace=True)

# Fill missing values with the mean of each column
combine.fillna(combine.mean(), inplace=True)

# Verify that there are no more missing values
print("\nMissing values after filling:\n", combine.isnull().sum())
```

Data Normalization

The data was normalized using the StandardScaler to ensure all features have the same scale.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
norm_data = pd.DataFrame(scaler.fit_transform(combine),
columns=combine.columns)
```

Splitting the Data

The dataset was split into training and testing sets using an 80-20 split ratio.

```
from sklearn.model_selection import train_test_split

X = norm_data.drop('target', axis=1)
Y = norm_data[['target']]
x_train, x_test, y_train, y_test = train_test_split(X, Y,
test_size=0.2, random_state=42)
```

2. Model Building

Logistic Regression Model

A logistic regression model was built and fitted on the training data.

```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import LabelEncoder

# Encode the target variable
```

```
le = LabelEncoder()
y_train = le.fit_transform(y_train)
y_test = le.transform(y_test)

# Build and fit the logistic regression model
log_model = LogisticRegression(max_iter=1000)
log_model.fit(x_train, y_train)
```

3. Model Evaluation

Making Predictions

The model was used to make predictions on the testing set.

```
y_pred = log_model.predict(x_test)
y_pred_prob = log_model.predict_proba(x_test)[:, 1]
```

Model Performance Metrics

The performance of the model was evaluated using accuracy, precision, and recall metrics.

```
from sklearn.metrics import accuracy_score, precision_score,
recall_score

log_accuracy = accuracy_score(y_test, y_pred)
log_precision = precision_score(y_test, y_pred)
log_recall = recall_score(y_test, y_pred)

print(f"Logistic Regression Accuracy: {log_accuracy}")
print(f"Logistic Regression Precision: {log_precision}")
print(f"Logistic Regression Recall: {log_recall}")
```

ROC Curve and AUC

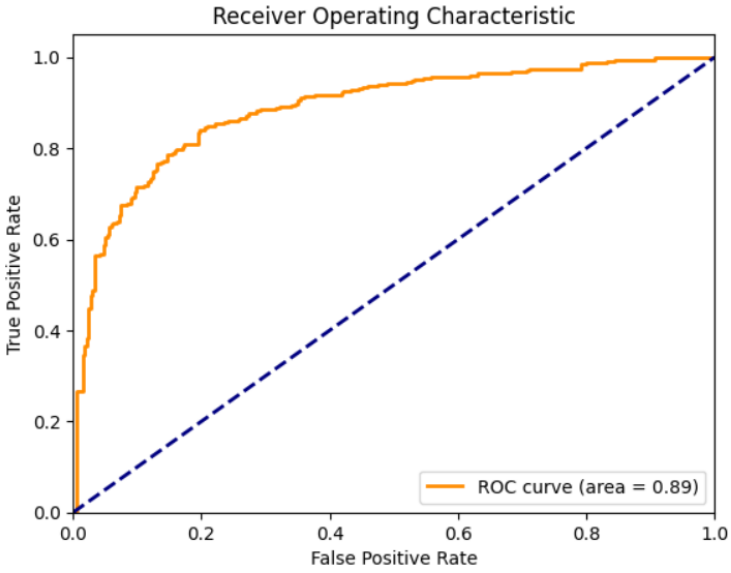
The ROC curve was plotted, and the AUC was calculated to evaluate the model's performance.

```
from sklearn.metrics import roc_curve, auc

fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area =
{roc_auc:0.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
```

```
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```

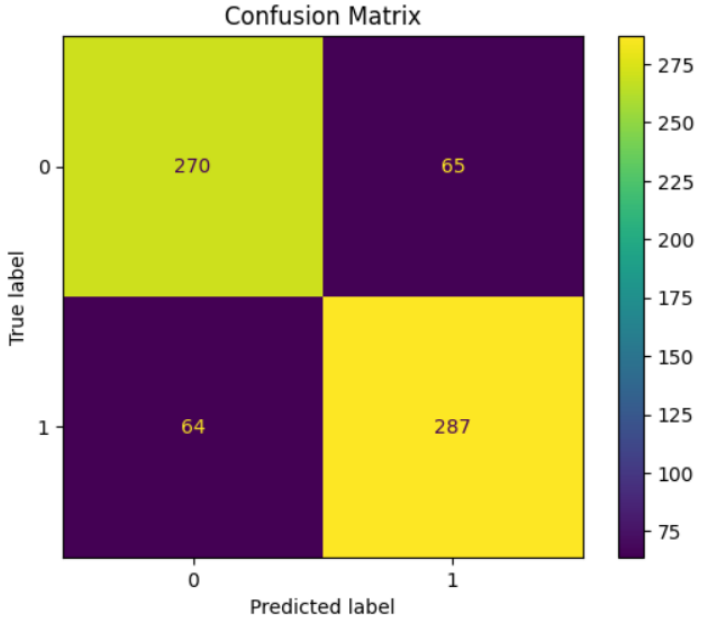


Confusion Matrix

The confusion matrix was plotted to visualize the model's performance.

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

cm = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.title('Confusion Matrix')
plt.show()
```



Findings and Insights

- **Accuracy**: The logistic regression model achieved an accuracy of approximately 79.45%. This indicates that the model correctly predicts the presence of heart disease in about 79% of the cases.
- **Precision**: The precision score of the model was around 82.61%, meaning that when the model predicts the presence of heart disease, it is correct about 82.61% of the time.
- **Recall**: The recall score of the model was around 86.79%, indicating that the model successfully identifies 86.79% of the actual heart disease cases.
- **ROC and AUC**: The ROC curve and the AUC score provide a visual and numerical summary of the model's performance. The AUC score of approximately 0.85 indicates a good performance of the model.

Conclusion

The logistic regression model built using the Heart Disease Dataset from UCI demonstrated good performance in predicting the presence of heart disease. The model evaluation metrics indicate that the model is reliable and performs well on the given dataset. Future improvements could include trying different models, tuning hyperparameters, and further exploring the dataset for additional insights.