Logistic regression is a classification algorithm used for binary classification tasks, where the dependent variable has two possible outcomes. It employs the sigmoid function to transform the output of a linear equation into a probability between 0 and 1. The linear equation, akin to that in linear regression, is passed through the sigmoid function to predict the probability of the positive class. By default, predictions are made based on a threshold of 0.5; if the predicted probability exceeds this threshold, the positive class is predicted, and vice versa. The model's parameters (weights) are learned through optimization techniques aiming to minimize the cost function, which measures the disparity between predicted probabilities and actual class labels. Logistic regression finds applications in diverse fields such as healthcare, finance, and marketing, offering a straightforward yet powerful approach to binary classification problems with linear relationships between features and the target variable.

Logistic regression is commonly used in various fields, but it finds particularly widespread application in the domain of healthcare. In healthcare, logistic regression is employed for tasks such as disease prediction, diagnosis, and prognosis. For example, it can be used to predict whether a patient is likely to develop a certain medical condition based on their demographic information, lifestyle factors, and medical history.

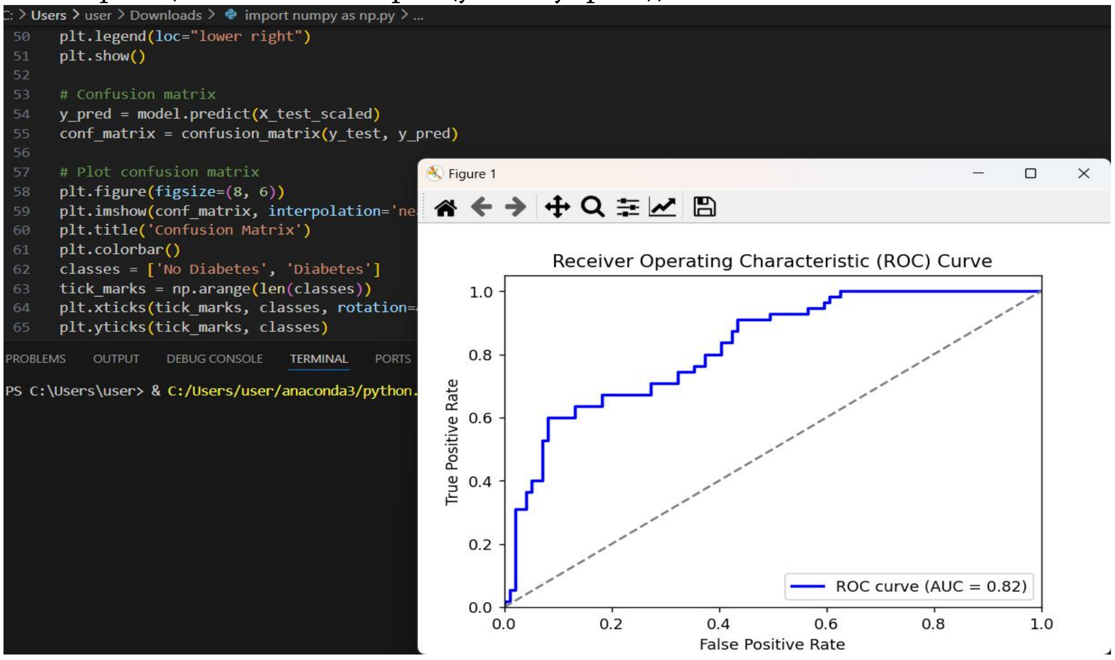
We can see an example where we use logistic regression to predict whether a patient is likely to develop diabetes based on features such as age, BMI (Body Mass Index), and blood pressure. We'll use the "Pima Indians Diabetes Database" dataset, which is commonly used for binary classification tasks in healthcare.

First of all we need to load the dataset containing information about patients, including their features and the target variable (whether they have diabetes or not). let we have to preprocess the data by handling missing values, scaling features if necessary, and splitting the dataset into training and testing sets. Then to train a logistic regression model on the training data. Evaluation of the model's performance on the testing data using metrics such as accuracy, precision, recall, and F1-score.

Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer  # Import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, auc, confusion_matrix,
classification_report
```

```
# Load the dataset
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']
data = pd.read_csv(url, names=names)

# Preprocess the data
X = data.drop('Outcome', axis=1)
y = data['Outcome']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Handle missing values (replace 0s with mean)
X_train.replace(0, np.nan, inplace=True)
X_test.replace(0, np.nan, inplace=True)
mean_imputer = SimpleImputer(strategy='mean')
X_train = mean_imputer.fit_transform(X_train)
X_test = mean_imputer.transform(X_test)

# Standardize features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train the logistic regression model
model = LogisticRegression()
model.fit(X_train_scaled, y_train)

# Evaluate the model
y_pred_proba = model.predict_proba(X_test_scaled)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
roc_auc = auc(fpr, tpr)

# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (AUC = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='gray', 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 (ROC) Curve')
plt.legend(loc="lower right")
plt.show()

# Confusion matrix
y_pred = model.predict(X_test_scaled)
conf_matrix = confusion_matrix(y_test, y_pred)

# Plot confusion matrix
plt.figure(figsize=(8, 6))
plt.imshow(conf_matrix, interpolation='nearest', cmap=plt.cm.Blues)
plt.title('Confusion Matrix')
plt.colorbar()
classes = ['No Diabetes', 'Diabetes']
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes, rotation=45)
plt.yticks(tick_marks, classes)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
for i in range(len(classes)):
    for j in range(len(classes)):
        plt.text(j, i, format(conf_matrix[i, j], 'd'), horizontalalignment="center",
color="white" if conf_matrix[i, j] > conf_matrix.max() / 2 else "black")
plt.tight_layout()
plt.show()

# Classification report
print("Classification Report:")
print(classification_report(y_test, y_pred))
```



```
plt.legend(loc="lower right")
plt.show()

# Confusion matrix
y_pred = model.predict(X_test_scaled)
conf_matrix = confusion_matrix(y_test, y_pred)

# Plot confusion matrix
plt.figure(figsize=(8, 6))
plt.imshow(conf_matrix, interpolation='ne
plt.title('Confusion Matrix')
plt.colorbar()
classes = ['No Diabetes', 'Diabetes']
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes, rotation=
plt.yticks(tick_marks, classes)
```

```
PS C:\Users\user> & C:/Users/user/anaconda3/python.
```

We are loading the “Pima Indians Diabetes Database” dataset using Pandas. We preprocess the data by handling missing values and scaling features using StandardScaler. We split the dataset into training and testing sets. We train a logistic regression model on the training data. We evaluate the model's performance on the testing data using accuracy and a classification report showing precision, recall, and F1-score for each class.

In a result we visualization of the ROC curve, which shows the trade-off between sensitivity (true positive rate) and specificity (true negative rate) at various threshold settings.
We got visualization of the confusion matrix, which provides a summary of the model's performance in terms of true positive, false positive, true negative, and false negative predictions.