| **CO4** | **Use the concept of neural networks for learning linear and non-linear activation functions** |
| --- | --- |
| **Task8**: | Create a perceptron with appropriate number of inputs and outputs using Medical data. Train it using fixed increment learning algorithm until no change in weights is required. Output the final weights.  **Platform: Google co-lab, Language: Python** |

**Use Case: Predicting Stroke Risk in Healthcare**

**Background:**

Healthcare providers are increasingly leveraging machine learning to enhance patient care and identify individuals at risk of specific health conditions. In this use case, we focus on predicting the risk of stroke using a dataset that includes various health-related features.

**Objective:**

The primary objective is to develop a predictive model that can assist healthcare professionals in identifying individuals who may be at a higher risk of experiencing a stroke. Early identification of high-risk individuals allows for targeted interventions and preventive measures.

**Data:**

The dataset used in this use case contains information about individuals, including their age, presence of hypertension, heart disease, average glucose level, and body mass index (bmi). The dataset also includes an indicator variable for stroke (1 for individuals who experienced a stroke and 0 for those who did not).

**Algorithm:**

1. Data Collection and Loading:

- Mount Google Drive.

- Read the healthcare dataset from a CSV file stored on Google Drive.

2. Data Preprocessing:

- Fill missing values in the dataset with the mean of their respective columns.

- Select relevant features (age, hypertension, heart disease, avg\_glucose\_level, bmi) and the target variable (stroke).

- Split the dataset into input features (X) and the target variable (y).

- Split the data into training and testing sets.

- Standardize the features using StandardScaler.

3. Model Training:

- Create a Perceptron model with specified hyperparameters (max\_iter, eta0, random\_state).

- Train the Perceptron model on the standardized training data.

4. Model Evaluation:

- Make predictions on the test set using the trained model.

- Calculate and print the accuracy of the model.

- Output the final weights of the perceptron.

5. Visualization:

- Create a confusion matrix and plot it as a heatmap using Seaborn.

- Generate an ROC curve and plot it.

- Create a Precision-Recall curve and plot it.

6. Results and Integration:

- Interpret the results, including accuracy and feature importance.

- Save the generated figures for future reference or reporting.

- Consider integrating the trained model into healthcare workflows for stroke risk assessment.

**Program:**

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import Perceptron

from sklearn.metrics import accuracy\_score, confusion\_matrix, roc\_curve, auc, precision\_recall\_curve

import seaborn as sns

import matplotlib.pyplot as plt

from google.colab import drive

drive.mount('/content/drive')

file\_path = '/content/drive/MyDrive/ML Lab/ExNo08/healthcare-dataset-stroke-data.csv'

data = pd.read\_csv(file\_path)

print(data.head())

data = data.fillna(data.mean())

X = data[['age', 'hypertension', 'heart\_disease', 'avg\_glucose\_level', 'bmi']]

y = data['stroke']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

perceptron = Perceptron(max\_iter=100, eta0=0.1, random\_state=42)

perceptron.fit(X\_train, y\_train)

y\_pred = perceptron.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f'Accuracy: {accuracy}')

final\_weights = perceptron.coef\_

print(f'Final Weights: {final\_weights}')

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(12, 4))

plt.subplot(1, 3, 1)

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['No Stroke', 'Stroke'], yticklabels=['No Stroke', 'Stroke'])

plt.xlabel('Predicted')

plt.ylabel('True')

plt.title('Confusion Matrix')

plt.subplot(1, 3, 2)

fpr, tpr, thresholds = roc\_curve(y\_test, perceptron.decision\_function(X\_test))

roc\_auc = auc(fpr, tpr)

plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC = {roc\_auc:.2f})')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC) Curve')

plt.legend(loc='lower right')

plt.subplot(1, 3, 3)

precision, recall, \_ = precision\_recall\_curve(y\_test, perceptron.decision\_function(X\_test))

plt.plot(recall, precision, color='green', lw=2, label='Precision-Recall curve')

plt.xlabel('Recall')

plt.ylabel('Precision')

plt.title('Precision-Recall Curve')

plt.tight\_layout()

plt.savefig('/content/drive/MyDrive/ML Lab/ExNo10/Figure.png',dpi=500)

plt.show()

id gender age hypertension heart\_disease ever\_married \

0 9046 Male 67.0 0 1 Yes

1 51676 Female 61.0 0 0 Yes

2 31112 Male 80.0 0 1 Yes

3 60182 Female 49.0 0 0 Yes

4 1665 Female 79.0 1 0 Yes

work\_type Residence\_type avg\_glucose\_level bmi smoking\_status \

0 Private Urban 228.69 36.6 formerly smoked

1 Self-employed Rural 202.21 NaN never smoked

2 Private Rural 105.92 32.5 never smoked

3 Private Urban 171.23 34.4 smokes

4 Self-employed Rural 174.12 24.0 never smoked

stroke

0 1

1 1

2 1

3 1

4 1

Accuracy: 0.9158512720156555

Final Weights: [[ 0.18583003 -0.42796629 0.188951 -0.19687049 0.06610429]]

