IT353 : Deep Learning

Lab Assignment 2 - Feed Forward Neural Network

Kesanam Ashinee
211AI023

Dataset

The Stroke classification Dataset picked from kaggle is used to predict whether a patient is likely to get a stroke based on the input parameters like gender, age, various diseases, and smoking status. Each row in the data provides relevant information about the patient.

Attributes information:

- id: unique identifier
- gender: "Male", "Female" or "Other"
- age: age of the patient
- hypertension: 0 if the patient doesn't have hypertension, 1 if the patient has hypertension
- heart_disease: 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease
- ever_married: "No" or "Yes"
- work_type: "children", "Govt_jov", "Never_worked", "Private" or "Self-employed"
- Residence_type: "Rural" or "Urban"
- avg_glucose_level: average glucose level in blood
- bmi: body mass index
- smoking_status: "formerly smoked", "never smoked", "smokes" or "Unknown"*
- stroke: 1 if the patient had a stroke or 0 if not

Link: Stroke Prediction Dataset

Google Colab Notebook

All code can be found here - Code Notebook

Tasks

Implement an Feed Forward network with following Architectures:

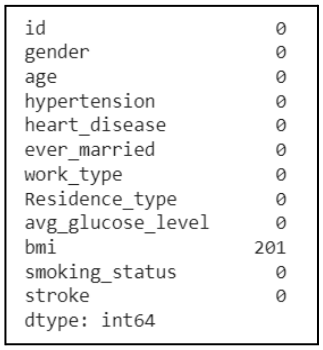
A. A FFN which starts with X units in first layer and narrows down to classification layer with 2 output forms
B. A FFN which starts with X units in first layer and increases the hidden layer size and narrows down to classification layer with 2 output forms

Compute Accuracy, Precision, Recall, F1 Score for Train, Validation, Test set and visualize.

Implementation:

Data Preprocessing:

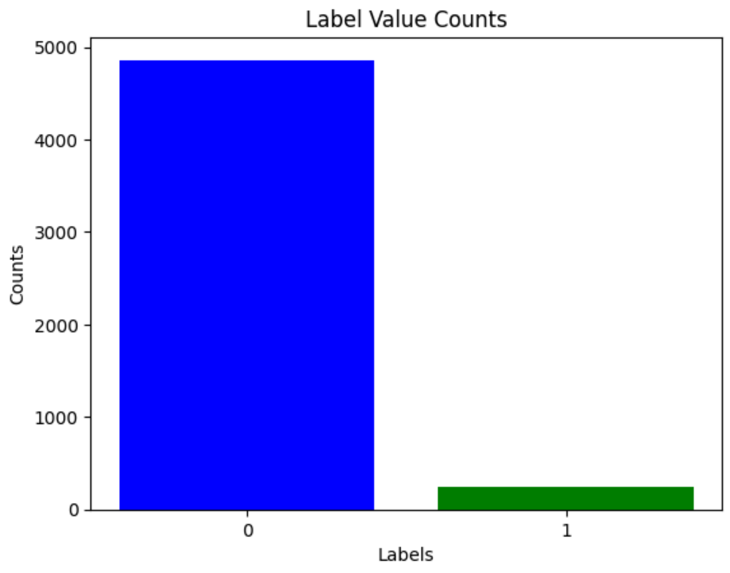
- **Handling Missing Values (Imputation):**
  - The initial step involves checking for missing values in the dataset using the .isna().sum() method. In this case, there were 201 missing values in the 'BMI' column.



```
id                    0
gender                0
age                   0
hypertension          0
heart_disease         0
ever_married          0
work_type             0
Residence_type        0
avg_glucose_level     0
bmi                 201
smoking_status        0
stroke                0
dtype: int64
```

  - To handle missing values, the code utilizes the SimpleImputer class from scikit-learn, which provides a simple strategy for imputing missing values.
  - It separates the dataset into two subsets based on gender ('Male' and 'Female') using boolean indexing.
- **Imputation Process:**
  - The SimpleImputer is instantiated with the strategy set to 'mean', which replaces missing values with the mean of the non-missing values in the column.
  - The imputer is then fitted to the 'BMI' column of the entire dataset using fit().
  - After fitting, it transforms the 'BMI' column separately for the male and female datasets using transform() and assigns the imputed values back to the 'BMI' column for each subset.
- **Concatenating Datasets:**
  - After imputation, it concatenates the male and female datasets back together using pd.concat(), with the ignore_index=True parameter to reset the index of the concatenated DataFrame.
- **Counting Unique Values in the Target Column ('Stroke'):**
  - The code counts the number of unique values in the 'stroke'

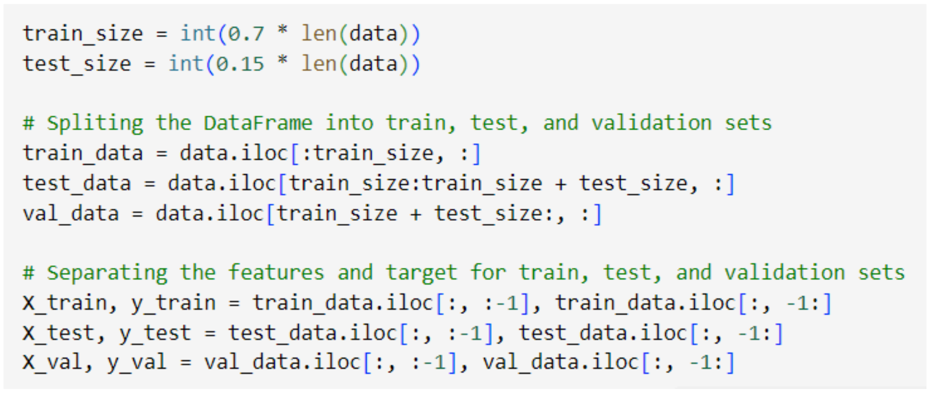
column using np.unique().
    - It then creates a bar chart using Matplotlib to visualize the distribution of the 'stroke' values.



- **Changing Categorical Values to Numerical Values:**
    - This section converts categorical variables into numerical representations, which is often necessary for machine learning algorithms.
    - It utilizes the replace() method to map categorical values to numerical values for each categorical feature:
        - 'gender': 'Male' and 'Female' are replaced with 0 and 1, respectively.
        - 'ever_married': 'Yes' and 'No' are replaced with 0 and 1, respectively.
        - 'work_type': Categorical work types are replaced with numerical codes (e.g., 'Private' is 1, 'Govt_job' is 2, etc.).
        - 'Residence_type': 'Urban' and 'Rural' are replaced with 0 and 1, respectively.
        - 'smoking_status': Categorical smoking statuses are replaced with numerical codes (e.g., 'formerly smoked' is 1, 'never smoked' is 2, etc.).

Train, Validation, Test Sets Splitting:

The dataset is split into train, val and test datasets in the ratio of 70:15:15



```
train_size = int(0.7 * len(data))
test_size = int(0.15 * len(data))

# Spliting the DataFrame into train, test, and validation sets
train_data = data.iloc[:train_size, :]
test_data = data.iloc[train_size:train_size + test_size, :]
val_data = data.iloc[train_size + test_size:, :]

# Separating the features and target for train, test, and validation sets
X_train, y_train = train_data.iloc[:, :-1], train_data.iloc[:, -1:]
X_test, y_test = test_data.iloc[:, :-1], test_data.iloc[:, -1:]
X_val, y_val = val_data.iloc[:, :-1], val_data.iloc[:, -1:]
```

Model Architecture:

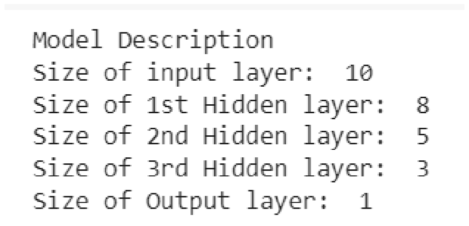
- **Input Layer:**
  - The input layer size is determined by the input_size parameter passed during the initialization of the model.
  - The input layer accepts input data, which is passed to the first hidden layer.
- **Hidden Layers:**
  - The FNN consists of multiple hidden layers, the number and size of which are determined by the hidden_sizes parameter passed during the initialization of the model.
  - Each hidden layer applies an activation function (in this case, the sigmoid function) to the weighted sum of inputs from the previous layer and adds a bias term.
  - The weights and biases for each hidden layer are randomly initialized during the initialization of the model.
  - The activation function used in the hidden layers is the sigmoid function (sigmoid(x)), which squashes the weighted sum of inputs to a value between 0 and 1.
- **Output Layer:**
  - The output layer is the final layer of the neural network, which produces the model's predictions.
  - The size of the output layer is determined by the output_size

    parameter passed during the initialization of the model.
  - The output layer also applies the sigmoid activation function to produce the final output.
- **Initialization of Weights and Biases:**
  - Weights and biases are initialized randomly using a normal distribution (Gaussian distribution) with mean 0 and standard deviation 1.
  - The weights and biases for the hidden layers are stored in lists (self.weights and self.biases).
- **Forward Propagation:**
  - The forward() method computes the output of the neural network given an input X.
  - It iterates through each layer, computing the weighted sum of inputs, applying the sigmoid activation function, and passing the output to the next layer.
  - The output of the last layer is returned as the final output of the network.
- **Backward Propagation:**
  - The backward() method implements the backpropagation algorithm to update the weights and biases of the network based on the error between the predicted output and the true output (y).
  - It computes the error at the output layer and propagates it backward through the network, updating the weights and biases using gradient descent.
- **Training:**
  - The train() method trains the neural network by iterating over the dataset for a specified number of epochs.
  - It computes the output of the network, calculates the loss (mean squared error), and updates the weights and biases using backpropagation.
  - It prints the loss at regular intervals to monitor the training progress.
- **Prediction:**
  - The predict() method is used to make predictions using the trained neural network.
  - It computes the output of the network given an input X, which represents the predicted values for the input data.

A FFN which starts with X units in first layer and narrows down to classification layer with 2 output forms

Model Description:

- The feedforward neural network (FFNN) architecture consists of an input layer with 10 neurons, followed by three hidden layers with 8, 5, and 3 neurons respectively, and an output layer with 1 neuron.

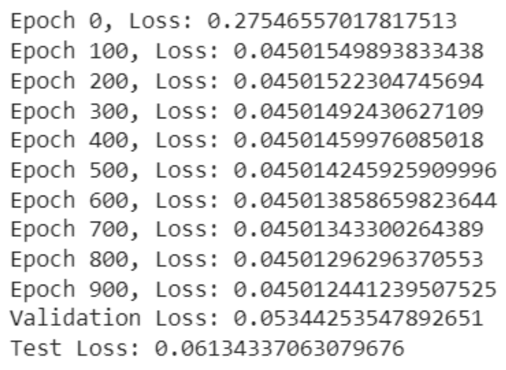


```
Model Description
Size of input layer:  10
Size of 1st Hidden layer:  8
Size of 2nd Hidden layer:  5
Size of 3rd Hidden layer:  3
Size of Output layer:  1
```

- The network is trained using the provided data with 1000 epochs and a learning rate of 0.01.

Training Loss:

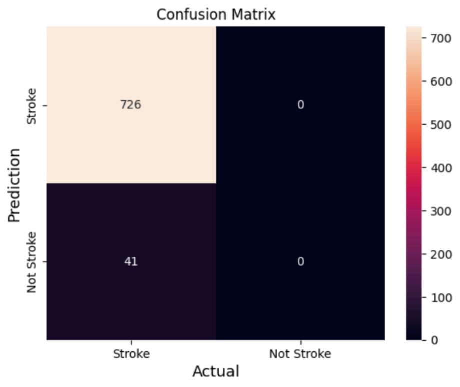
- The training loss decreases steadily over the epochs, indicating that the model is learning and optimizing its parameters effectively.



```
Epoch 0, Loss: 0.27546557017817513
Epoch 100, Loss: 0.04501549893833438
Epoch 200, Loss: 0.04501522304745694
Epoch 300, Loss: 0.04501492430627109
Epoch 400, Loss: 0.04501459976085018
Epoch 500, Loss: 0.045014245925909996
Epoch 600, Loss: 0.045013858659823644
Epoch 700, Loss: 0.04501343300264389
Epoch 800, Loss: 0.04501296296370553
Epoch 900, Loss: 0.045012441239507525
Validation Loss: 0.05344253547892651
Test Loss: 0.06134337063079676
```

Metrics for Validation Set:

- Confusion Matrix Visualization: The confusion matrix for the validation set shows the distribution of actual and predicted labels. However, the heatmap appears to show low values for the "Stroke" class prediction, suggesting potential issues with class imbalance or model performance.



- Classification Report: The precision, recall, and F1-score for each class (0 and 1) are reported, along with support (the number of true instances for each class). The "1" class (stroke) has low precision, recall, and F1-score, indicating poor performance in predicting instances of stroke.

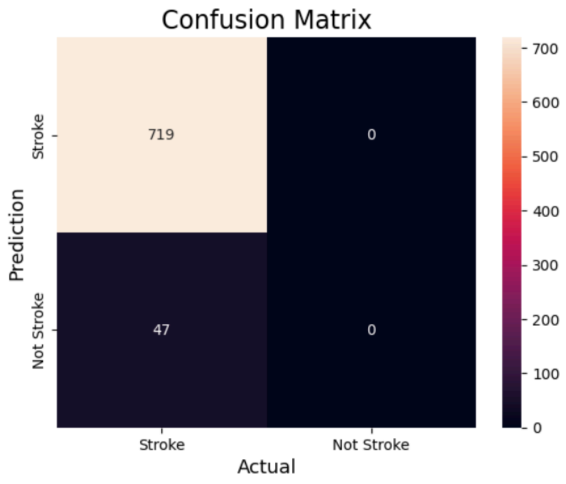
```
Validation Classification Report:
              precision    recall  f1-score   support

           0       0.95      1.00      0.97       726
           1       0.00      0.00      0.00        41

    accuracy                           0.95       767
   macro avg       0.47      0.50      0.49       767
weighted avg       0.90      0.95      0.92       767
```

- Validation Accuracy: The validation accuracy is around 94.65%, indicating the proportion of correctly classified instances in the validation set.

```
Validation Accuracy: 0.9465449804432855
```

Metrics for Test Set:

- Confusion Matrix Visualization: Similar to the validation set, the confusion matrix for the test set visualizes the distribution of actual and predicted labels.



- Classification Report: The classification report provides precision, recall, F1-score, and support for each class (0 and 1). Again, the "1" class (stroke) shows poor performance with low precision, recall, and F1-score.

```
Test Classification Report:
              precision    recall  f1-score   support

           0       0.94      1.00      0.97       719
           1       0.00      0.00      0.00        47

    accuracy                           0.94       766
   macro avg       0.47      0.50      0.48       766
weighted avg       0.88      0.94      0.91       766
```

- Test Accuracy: The test accuracy is approximately 93.86%, indicating the proportion of correctly classified instances in the test set.

```
Test Accuracy: 0.9386422976501305
```

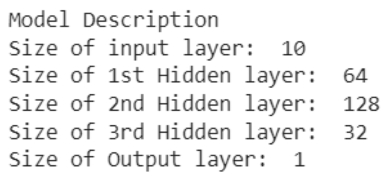
Interpretation:

- While the model achieves high accuracy overall, it struggles particularly with correctly predicting instances of stroke (class "1").
- The precision, recall, and F1-score for the "1" class are all low, suggesting that the model may need further refinement or feature engineering to better capture patterns associated with stroke.
- Consideration should be given to addressing class imbalance if present, as it may skew the model's predictions and evaluation metrics.

A FFN which starts with X units in first layer and increases the hidden layer size and narrows down to classification layer with 2 output forms

Model Architecture:

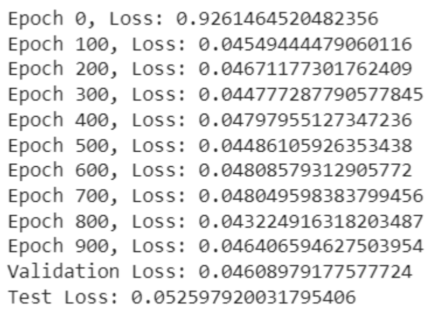
- The feedforward neural network (FFNN) architecture includes an input layer with 10 neurons, followed by three hidden layers with 64, 128, and 32 neurons respectively, and an output layer with 1 neuron.
- The model is designed for binary classification tasks, specifically for predicting stroke occurrences.



```
Model Description
Size of input layer:  10
Size of 1st Hidden layer:  64
Size of 2nd Hidden layer:  128
Size of 3rd Hidden layer:  32
Size of Output layer:  1
```

Training:

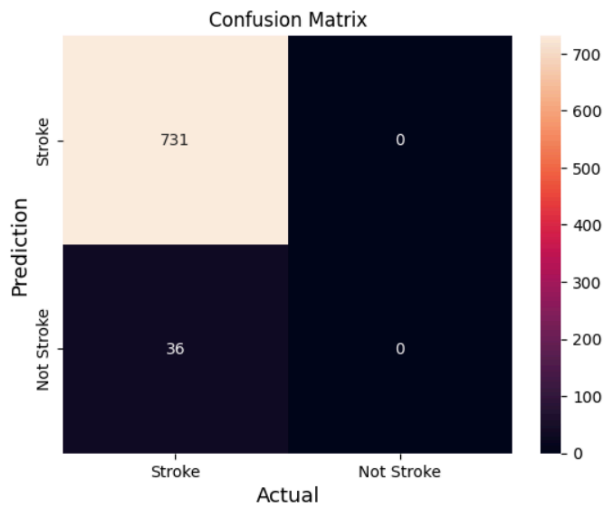
- The model is trained using the provided training data for 1000 epochs with a learning rate of 0.01.
- The training loss starts at a high value and gradually decreases over epochs, indicating the model's learning process.



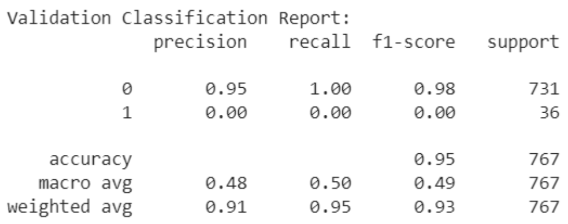
```
Epoch 0, Loss: 0.9261464520482356
Epoch 100, Loss: 0.04549444479060116
Epoch 200, Loss: 0.04671177301762409
Epoch 300, Loss: 0.044777287790577845
Epoch 400, Loss: 0.04797955127347236
Epoch 500, Loss: 0.04486105926353438
Epoch 600, Loss: 0.04808579312905772
Epoch 700, Loss: 0.048049598383799456
Epoch 800, Loss: 0.043224916318203487
Epoch 900, Loss: 0.046406594627503954
Validation Loss: 0.04608979177577724
Test Loss: 0.052597920031795406
```

Validation Set Performance:

- Confusion Matrix Visualization: The confusion matrix for the validation set displays the distribution of actual and predicted labels. However, there seems to be an issue with predicting instances of stroke, as evidenced by the low values in the "1" class.

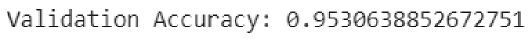


- Classification Report: The classification report provides precision, recall, and F1-score for each class (0 and 1) along with support (number of instances). Unfortunately, the model performs poorly in predicting instances of stroke ("1" class) with low precision, recall, and F1-score.



```
Validation Classification Report:
              precision    recall  f1-score   support

           0       0.95      1.00      0.98       731
           1       0.00      0.00      0.00        36

    accuracy                           0.95       767
   macro avg       0.48      0.50      0.49       767
weighted avg       0.91      0.95      0.93       767
```

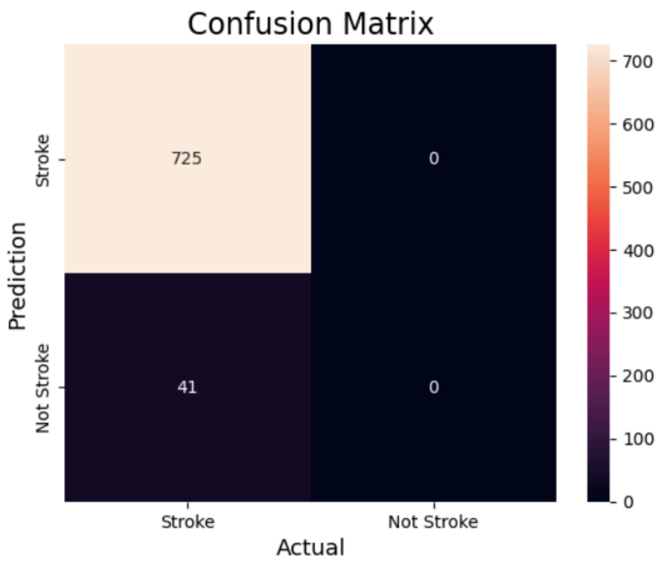
- Validation Accuracy: The validation accuracy stands at approximately 95.31%, indicating the proportion of correctly classified instances in the validation set.



```
Validation Accuracy: 0.9530638852672751
```

Test Set Performance:

- Confusion Matrix Visualization: The confusion matrix for the test set shows a similar distribution of actual and predicted labels as observed in the validation set.



- Classification Report: The classification report provides precision, recall, and F1-score for each class (0 and 1) along with support (number of instances). Again, the model struggles to predict instances of stroke ("1" class) with low precision, recall, and F1-score.

```
Test Classification Report:
              precision    recall  f1-score   support

           0       0.95      1.00      0.97       725
           1       0.00      0.00      0.00        41

    accuracy                           0.95       766
   macro avg       0.47      0.50      0.49       766
weighted avg       0.90      0.95      0.92       766
```

- Test Accuracy: The test accuracy is around 94.65%, indicating the proportion of correctly classified instances in the test set.

```
Test Accuracy: 0.9464751958224543
```

Interpretation:

- Despite achieving high overall accuracy, the model's performance in identifying instances of stroke (class "1") is inadequate.
- The precision, recall, and F1-score for the "1" class are all low, suggesting that the model may require further optimization or feature engineering to capture stroke patterns effectively.