



EAST WEST UNIVERSITY

Course Title : Artificial Intelligence
Course Code : CSE365
Section : (01)
Semester : Spring 2022

Project Report

Project Name: Feed Forward Neural Network Implementation using Kaggle Dataset

Submitted To

Jesan Ahammed Ovi
Senior Lecturer
Department of Computer Science and Engineering
East West University

Submitted By

Md. Jannatul Haq	2018-1-60-224
Apu Sardar	2018-1-60-120
Nurunnaher Amy	2017-1-60-147
Afrina Tasbi	2018-1-60-055

Submission Date

19/05/2022

Problem Statement

A stroke is a health condition that causes damage by tearing the blood vessels in the brain. It can also occur when there is a halt in the blood flow and other nutrients to the brain. According to the World Health Organization (WHO), stroke is the leading cause of death and disability globally.

For the implementation of our project, we have taken a dataset from <http://www.kaggle.com>. From this website, we choose “healthcare-dataset-stroke-data”. In this dataset, there is given information from some males and females regarding the issue of stroke. In our dataset, there is some attribute information in the .csv file. With the help of that information, we are going to find the accuracy of this dataset by applying the “Feed Forward Neural Networks” method. Data accuracy depicts what percentage of males and females can undergo a stroke.

This dataset 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. A subset of the original train data is taken using the filtering method for Machine Learning and Data Visualization purposes. Each row in the data provides relative information about a person, for instance; age, gender, smoking status, and accuracy of stroke in addition to other information. Unknown in Smoking status means the information is unavailable. N/A in other input fields implies that it is not applicable.

System Requirements:

- ▶ Processor: Intel(R) Core (TM) i5-7200U CPU @ 2.50GHz 2.71 GHz •
- ▶ RAM: 4.00 GB
- ▶ Operating System: 64-bit operating system, x64-based processor
- ▶ Visual Studio Code
- ▶ Python 3.10.4
- ▶ Jupyter Notebook

System design:

A neural network is a bunch of neurons connected. Neural Networks in a distinct category recognize their unique approach to finding dataset accuracy. However, it is essential to remember that Neural Networks are most frequently employed to solve classification and regression problems using labeled training data. 2 Training Neural Networks involve a complicated process known as backpropagation. We use this backpropagation work in this project by using a python built-in function. We also use loss functions, and logistic functions as activation functions and optimizers and summarize what happens when we “train” a Neural Network. There are some steps of our design are given below:

Step 1: A set of inputs enter the network through the input layer and are multiplied by their weights.

Step 2: Each value is added to receive a summation of the weighted inputs. If the sum value exceeds the specified limit (usually 0), the output usually settles at 1

[Step 3:](#) A single-layer perceptron uses the concepts of machine learning for classification. It is a crucial model of a feedforward neural network.

[Step 4:](#) The outputs of the neural network can then be compared with their predicted values with the help of the delta rule, thereby facilitating the network to optimize its weights through training to obtain output values with better accuracy. This process of training and learning generates a gradient descent.

[Step 5:](#) In multi-layered networks, updating weights are analogous and more specifically defined as backpropagation. Here, each hidden layer is modified to stay in tune with the output value generated by the final layer.

Implementation:

```
import pandas as pd
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

[3] ✓ 0.4s

```
data = pd.read_csv("healthcare-dataset-stroke-data.csv")
data
```

[4] ✓ 0.2s

	id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
0	9046	Male	67.0	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
1	51676	Female	61.0	0	0	Yes	Self-employed	Rural	202.21	NaN	never smoked	1
2	31112	Male	80.0	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
3	60182	Female	49.0	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
4	1665	Female	79.0	1	0	Yes	Self-employed	Rural	174.12	24.0	never smoked	1
...
5105	18234	Female	80.0	1	0	Yes	Private	Urban	83.75	NaN	never smoked	0
5106	44873	Female	81.0	0	0	Yes	Self-employed	Urban	125.20	40.0	never smoked	0
5107	19723	Female	35.0	0	0	Yes	Self-employed	Rural	82.99	30.6	never smoked	0
5108	37544	Male	51.0	0	0	Yes	Private	Rural	166.29	25.6	formerly smoked	0
5109	44679	Female	44.0	0	0	Yes	Govt_job	Urban	85.28	26.2	Unknown	0

5110 rows × 12 columns

[# import pandas as pd](#)

This module in advance python is used to work for the dataset. This module can read any kind of file here to read the .csv file we use.

```
# sklearn.neural_network import MLPClassifier
```

This module is used for data classification and it is scientific learning such as linear regression, Bayesian partial, and so on. Here we use this for neural_network. MLPClassifier is used for backpropagation. Here we just call this object of this class, how many times have we been learning our program.

```
#sklearn.preprocessing import LabelEncoder
```

This header file work for labeling data. Suppose we have a yes or no value then it will label yes to 1 and no to 0.

```
# sklearn.model_selection import train_test_split
```

We call train_test_split in sklearn module. Train and test are used to separate values. We must take some values for the train and some values for the test.

```
#sklearn.metrics import accuracy_score
```

This module is worked to calculate the loss and find the accuracy of the data set.

```
#data_find = LabelEncoder()
```

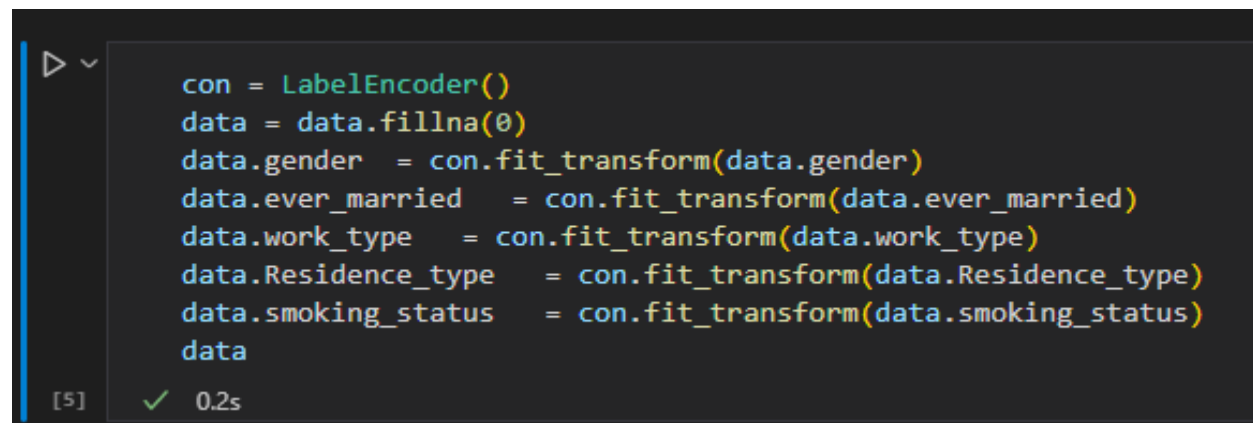
Label the n amount of data from 1 to n in decreasing order.

```
#data = data.fillna(0)
```

If there is a missing or Null value in any column of the .csv file, then it fills that column by inserting 0.

```
# data = pd.read_csv("healthcare-dataset-stroke-data.csv")
```

To read our .csv file we use this module. Here in .csv file Stroke column is considered as y and all other column as conders as x. There are 5110 rows and 12 columns.



```
con = LabelEncoder()
data = data.fillna(0)
data.gender = con.fit_transform(data.gender)
data.ever_married = con.fit_transform(data.ever_married)
data.work_type = con.fit_transform(data.work_type)
data.Residence_type = con.fit_transform(data.Residence_type)
data.smoking_status = con.fit_transform(data.smoking_status)
data
```

[5] ✓ 0.2s

...		id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
	0	9046	1	67.0	0	1	1	2	1	228.69	36.6	1	1
	1	51676	0	61.0	0	0	1	3	0	202.21	0.0	2	1
	2	31112	1	80.0	0	1	1	2	0	105.92	32.5	2	1
	3	60182	0	49.0	0	0	1	2	1	171.23	34.4	3	1
	4	1665	0	79.0	1	0	1	3	0	174.12	24.0	2	1

	5105	18234	0	80.0	1	0	1	2	1	83.75	0.0	2	0
	5106	44873	0	81.0	0	0	1	3	1	125.20	40.0	2	0
	5107	19723	0	35.0	0	0	1	3	0	82.99	30.6	2	0
	5108	37544	1	51.0	0	0	1	2	0	166.29	25.6	1	0
	5109	44679	0	44.0	0	0	1	0	1	85.28	26.2	0	0

5110 rows × 12 columns

Here, in this section, we are labeling our data. After label encoding every attribute, data give a numerical value. It will insert 0 or 1 if there is only a binary variable.

```

feature = data.columns[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]]
feature
[6] ✓ 0.1s

... Index(['id', 'gender', 'age', 'hypertension', 'heart_disease', 'ever_married',
        'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',
        'smoking_status'],
        dtype='object')

▶ ~
train, test = train_test_split(data, test_size = 0.3)
print(len(train))
print(len(test))
[7] ✓ 0.1s

... 3577
    1533

```

Here our total row is 5110 we took 30% (0.3) of data from this for the test. So, reaming 70% of data is for the train. So, then we got 3577 rows and 1533 columns.

```
omega = MLPClassifier(activation = 'logistic', max_iter = 1000, hidden_layer_sizes = (30,30,30, 30, 30), learning_rate_init = 0.01, solve

[8] ✓ 0.1s Python

x_train = train[data.columns[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]]]
y_train = train[data.columns[11]]

x_test = test[data.columns[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]]]
y_test = test[data.columns[11]]

[7] Python
```

Here, the omega is an object, in this, we construct the iteration value 1000 times. The activation function will be a logistic function. There is five hidden layers which layer has 30 neurons, and the learning rate is 0.01. In solver, we use backpropagation work using stochastic gradient descent (SGD).

```
x_train = train[data.columns[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]]]
y_train = train[data.columns[11]]

x_test = test[data.columns[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]]]
y_test = test[data.columns[11]]

[9] ✓ 0.1s
```

Here we are going to define x1, x2, x3..... inputs in x train. We train our data here. This means train row from this column 1-11 we send this for the rain. And the remaining column which is the output column which is the stroke we send this to train. Similarly, for the chosen column and row for the test, we send them to the test.

```
omega = omega.fit(x_train, y_train)

[10] ✓ 0.8s

y_pred = omega.predict(x_test)
y_pred

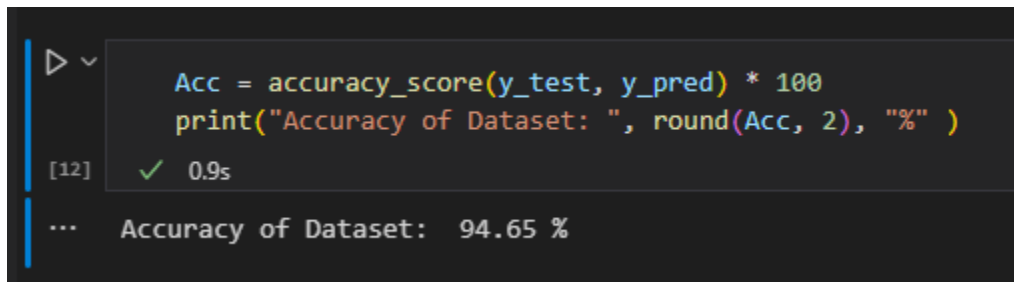
[11] ✓ 0.1s

... array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

Here, omega.fit does the work for both backpropagation and forward propagation. Now from the new defined weight, it will get the equation from this weight. It will put this equation again in

omega. Now if we call x_test the test values and get a new value of y and put it in omega.predict as an array.

Testing Result:

A screenshot of a Jupyter Notebook cell. The code in the cell is:

```
Acc = accuracy_score(y_test, y_pred) * 100  
print("Accuracy of Dataset: ", round(Acc, 2), "%" )
```

 The output of the cell is:

```
[12] ✓ 0.95  
... Accuracy of Dataset: 94.65 %
```

```
Acc = accuracy_score(y_test, y_pred) * 100  
print("Accuracy of Dataset: ", round(Acc, 2), "%" )
```

```
[12] ✓ 0.95  
... Accuracy of Dataset: 94.65 %
```

Future Scope:

There are some limitations in feedforward neural networks. It only works for data. It cannot identify the picture or any 2D array problem. The simplified architecture of Feed Forward Neural Network offers leverage in machine learning. A series of Feedforward networks can run independently with a slight intermediary to ensure moderation. The network requires several neurons to carry out complicated tasks. The handling and processing of non-linear data can be done easily with a neural network that is otherwise complex in perceptron and sigmoid neurons. The excruciating decision boundary problem is alleviated in neural networks. The excruciating decision boundary problem is alleviated in neural networks.

Code:

```
import pandas as pd  
from sklearn.neural_network import MLPClassifier  
from sklearn.preprocessing import LabelEncoder  
from sklearn.model_selection import train_test_split  
from sklearn.metrics import accuracy_score  
  
data = pd.read_csv("healthcare-dataset-stroke-data.csv")  
data  
  
con = LabelEncoder()  
data = data.fillna(0)  
data.gender = con.fit_transform(data.gender)  
data.ever_married = con.fit_transform(data.ever_married)  
data.work_type = con.fit_transform(data.work_type)  
data.Residence_type = con.fit_transform(data.Residence_type)  
data.smoking_status = con.fit_transform(data.smoking_status)  
data
```

```
feature = data.columns[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]]
feature
```

```
train, test = train_test_split(data, test_size = 0.3)
print(len(train))
print(len(test))
```

```
omega = MLPClassifier(activation = 'logistic', max_iter = 1000, hidden_layer_sizes = (30,30,30,
30, 30), learning_rate_init = 0.01, solver='sgd', random_state=None)
```

```
x_train = train[data.columns[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]]]
y_train = train[data.columns[11]]
```

```
x_test = test[data.columns[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]]]
y_test = test[data.columns[11]]
```

```
omega = omega.fit(x_train, y_train)
```

```
y_pred = omega.predict(x_test)
y_pred
```

```
Acc = accuracy_score(y_test, y_pred) * 100
print("Accuracy of Dataset: ", round(Acc, 2), "%" )
```