

**M.S.C.I.T.**

**GOVIND SAINI (07)**

**VALIA C.I. COLLEGE OF COMMERCE & VALIA LC COLLEGE OF ARTS  
CES ROAD D.N NAGAR**

(Affiliated to University Of Mumbai)

**Mumbai-Maharashtra-400053**

**DEPARTMENT OF INFORMATION TECHNOLOGY**



**CERTIFICATE**

This is to certify that the Journal entitled **Deep Learning** is bonafied work of  
**GOVIND SAINI** bearing Roll No: **07** submitted in partial fulfillment of the  
requirements for the award of degree of **BACHELOR OF SCIENCE** in  
**INFORMATION TECHNOLOGY** from University of Mumbai.

**Date:**

**Internal Guide**

# INDEX

Sr. No.	Practical	Pg. No.
1	<p><b>Write a Program to demonstrate following operations.</b></p> <ul style="list-style-type: none"> <li>i) Create Vector, Matrix and Tensor</li> <li>ii) Multiplication of two : Vector, Matrix and Tensor</li> <li>iii) Addition of two : Vector, Matrix and Tensor</li> <li>iv) Multiply Matrix with Vector</li> <li>v) Matrix Dot product and Matrix Inverse</li> </ul>	1
2	<b>Performing matrix multiplication and finding Eigen vectors and Eigen values using TensorFlow</b>	5
3	<b>Solving XOR problem using deep feed forward network.</b>	8
4	<b>Implementing deep neural network for performing binary classification task.</b>	12
5	<p>A. Using deep feed forward network with two hidden layers for performing multiclass classification and predicting the class.</p> <p>B. Using a deep feed forward network with two hidden layers for performing classification and predicting the probability of class.</p> <p>C. Using a deep feed forward network with two hidden layers for performing linear regression and predicting values.</p>	16
6	<p>A. Evaluating feed forward deep network for regression using K-Fold cross validation.</p> <p>B. Evaluating feed forward deep network for multiclass Classification using K-Fold cross-validation.</p>	29

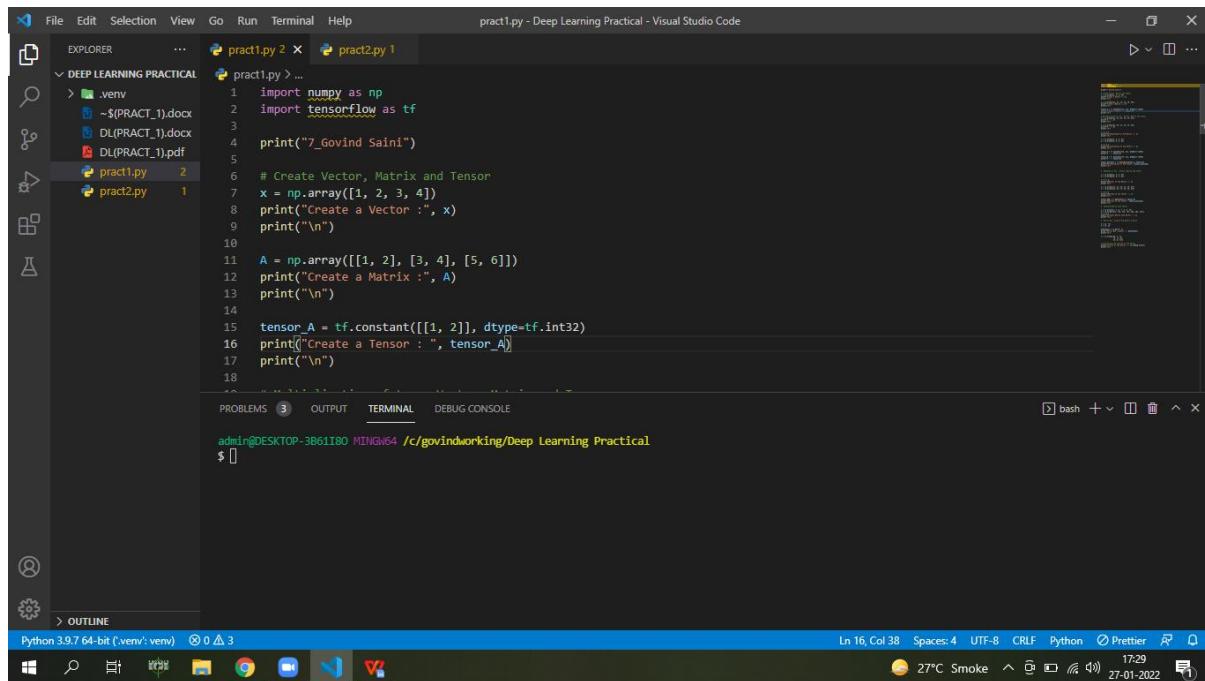
<b>7</b>	<b>Implementation of convolutional neural network to predict numbers from number images</b>	<b>37</b>
<b>8</b>	<b>Demonstrate recurrent neural network that learns to perform sequence analysis for stock price.</b>	<b>41</b>
<b>9</b>	<b>Performing encoding and decoding of images using deep Autoencoder.</b>	<b>48</b>
<b>10</b>	<b>Denoising of images using Autoencoder.</b>	<b>54</b>

# PRACTICAL NO : 1

**Aim : Write a Program to demonstrate following operations**

- 1) Create Vector, Matrix and Tensor
- 2) Multiplication of two : Vector, Matrix and Tensor
- 3) Addition of two : Vector, Matrix and Tensor
- 4) Multiply Matrix with Vector
- 5) Matrix Dot product and Matrix Inverse

**Code :**



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files in the "DEEP LEARNING PRACTICAL" folder: pract1.py (selected), pract2.py, .venv, and some documentation files.
- Code Editor:** Displays the content of pract1.py:

```
1 import numpy as np
2 import tensorflow as tf
3
4 print("Govind Saini")
5
6 # Create Vector, Matrix and Tensor
7 x = np.array([1, 2, 3, 4])
8 print("Create a Vector :", x)
9 print("\n")
10
11 A = np.array([[1, 2], [3, 4], [5, 6]])
12 print("Create a Matrix :", A)
13 print("\n")
14
15 tensor_A = tf.constant([[1, 2]], dtype=tf.int32)
16 print("Create a Tensor : ", tensor_A)
17 print("\n")
```
- Terminal:** Shows the command run in the terminal: `admin@DESKTOP-BB61I80 MINGW64 /c/govindworking/Deep Learning Practical`.
- Status Bar:** Shows Python 3.9.7 64-bit ('venv' venv) and other system information like date and time.

```

19 # Multiplication of two : Vector, Matrix and Tensor
20 A = np.array([[1, 2], [3, 4], [5, 6]])
21 print("A = ", A)
22 print("\n")
23
24 B = np.array([[2, 5], [7, 4], [4, 3]])
25 print("B = ", B)
26 print("\n")
27
28 C = A * B
29 print("Multiplication of two Matrix :", C)
30 print("\n")
31
32 x = np.array([1, 2, 3, 4])
33 y = np.array([5, 6, 7, 8])
34
35 z = x * y
36 print("Multiplication of two Vector :", z)
37 print("\n")
38
39 tensor_A = tf.constant([[4, 2]], dtype=tf.int32)
40 print("A : ", tensor_A)
41
42 tensor_B = tf.constant([[7, 4]], dtype=tf.int32)
43 print("B : ", tensor_B)
44
45 tensor_multiply = tf.multiply(tensor_A, tensor_B)
46 print("Multiplication of two Tensor", tensor_multiply)
47 print("\n")

```

```

50 # Addition of two : Vector, Matrix and Tensor
51
52 x = np.array([1, 2, 3, 4])
53 y = np.array([5, 6, 7, 8])
54
55 z = x + y
56 print("Addition of two Matrix :", z)
57 print("\n")
58
59 A = np.array([[1, 2], [3, 4], [5, 6]])
60 B = np.array([[2, 5], [7, 4], [4, 3]])
61
62 C = A * B
63 print("Addition of two vector :", C)
64 print("\n")
65
66 tensor_add = tf.add(tensor_A, tensor_B)
67 print("Addition of two Tensor", tensor_add)
68 print("\n")
69

```

The screenshot shows a Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** pract1.py - Deep Learning Practical - Visual Studio Code
- Left Sidebar:** Explorer, DEEP LEARNING PRACTICAL, .venv, pract1.py.
- Code Editor:** The file pract1.py contains Python code for matrix operations using NumPy. The code includes multiplying a matrix by a vector, calculating the dot product of two vectors, and calculating the inverse of a matrix.
- Bottom Navigation:** PROBLEMS, OUTPUT, TERMINAL (selected), DEBUG CONSOLE.
- Terminal:** admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/Deep Learning Practical
- Status Bar:** Ln 74, Col 1, Spaces: 4, UFT-8, CRLF, Python, Prettier.
- Bottom Icons:** Windows Start, Search, Task View, Chrome, File Explorer, File Manager, Taskbar icons, and a clock showing 10:21, 21-01-2022.

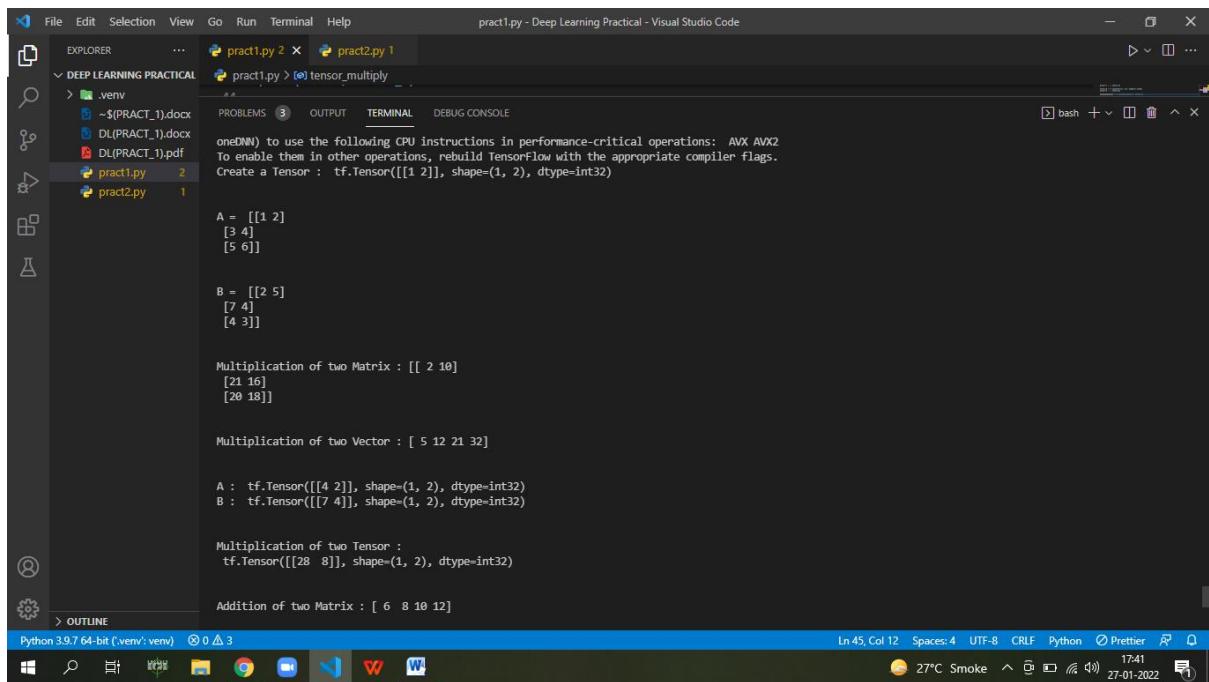
## **Output :**

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows files in the ".venv" directory, including "pract1.py" (2 changes), "pract2.py" (1 change), ".env", and several "DL(PRACT\_1).\* files.
- Code Editor:** Displays a Python script for matrix operations. The code includes matrix multiplication with a vector, matrix dot product, and matrix inverse.
- Terminal:** Shows the output of running the script:

```
admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/Deep Learning Practical
$ py pract1.py
7_Govind Saini
Create a Vector : [1 2 3 4]

Create a Matrix : [[1 2]
 [3 4]
 [5 6]]
```
- Status Bar:** Shows Python 3.9.7 64-bit (.venv/.venv) and system status (27°C, Smoke, battery level, signal strength).
- Bottom Bar:** Includes icons for file, search, terminal, and help.



pract1.py - Deep Learning Practical - Visual Studio Code

```

File Edit Selection View Go Run Terminal Help
pract1.py 2 x pract2.py 1
pract1.py > tensor_multiply
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Create a Tensor : tf.Tensor([1 2], shape=(1, 2), dtype=int32)

A = [[1 2]
[3 4]
[5 6]]

B = [[2 5]
[7 4]
[4 3]]

Multiplication of two Matrix : [[ 2 10]
[21 16]
[20 18]]

Multiplication of two Vector : [ 5 12 21 32]

A : tf.Tensor([[1 2]], shape=(1, 2), dtype=int32)
B : tf.Tensor([[7 4]], shape=(1, 2), dtype=int32)

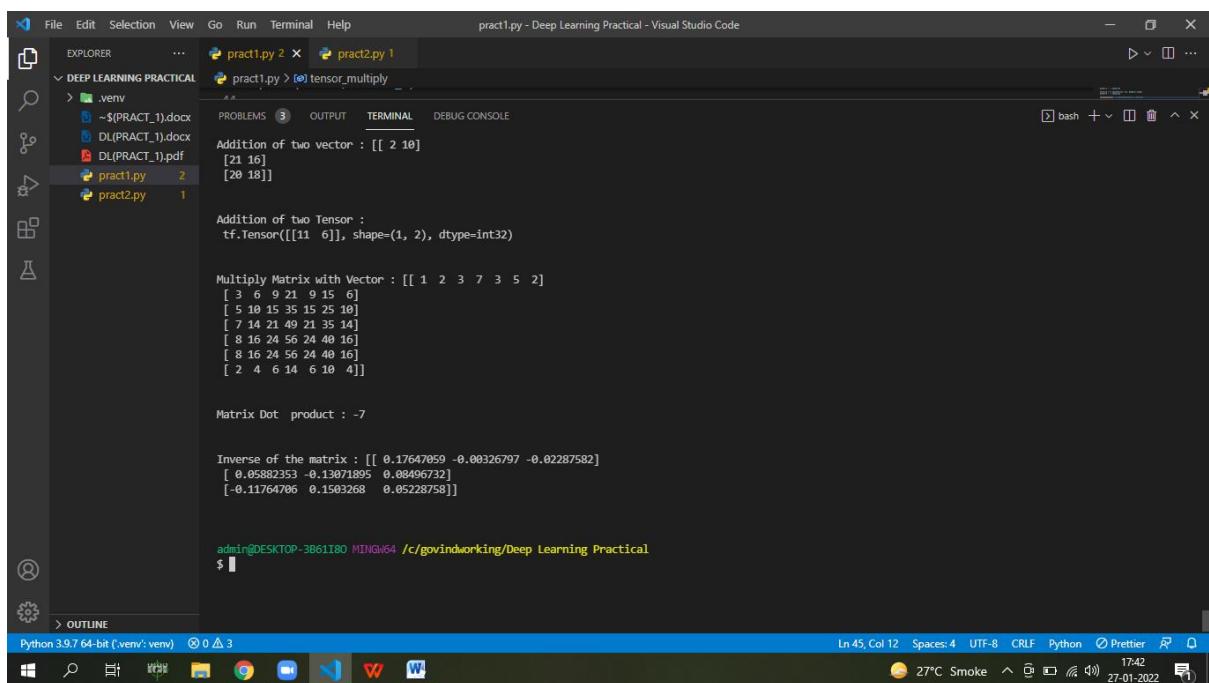
Multiplication of two Tensor :
tf.Tensor([[28 8]], shape=(1, 2), dtype=int32)

Addition of two Matrix : [ 6 8 10 12]

```

Python 3.9.7 64-bit ('venv': venv) 0 △ 3

Ln 45, Col 12 Spaces: 4 UTF-8 CRLF Python Prettier 17:41 27°C Smoke 27-01-2022



pract1.py - Deep Learning Practical - Visual Studio Code

```

File Edit Selection View Go Run Terminal Help
pract1.py 2 x pract2.py 1
pract1.py > tensor_multiply
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Addition of two vector : [[ 2 10]
[21 16]
[20 18]]

Addition of two Tensor :
tf.Tensor([[11 6]], shape=(1, 2), dtype=int32)

Multiply Matrix with Vector : [[ 1 2 3 7 3 5 2]
[ 3 6 9 21 9 15 6]
[ 5 10 15 35 15 25 10]
[ 7 14 21 49 21 35 14]
[ 8 16 24 56 24 40 16]
[ 8 16 24 56 24 40 16]
[ 2 4 6 14 6 10 4]]

Matrix Dot product : -7

Inverse of the matrix : [[ 0.17647059 -0.00326797 -0.02287582]
[ 0.05882353 -0.13071895 0.08496732]
[-0.11764706 0.1503268 0.05228758]]

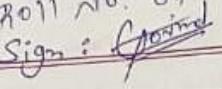
```

admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/Deep Learning Practical
\$

Python 3.9.7 64-bit ('venv': venv) 0 △ 3

Ln 45, Col 12 Spaces: 4 UTF-8 CRLF Python Prettier 17:42 27°C Smoke 27-01-2022

## Practical No : 2

Name: Govind Saini  
Roll No: 07  
Sign: 

Date: 27.01.2022  
Page.....

### Practical - 2

Q)

Aim:-  
Write a program to perform matrix multiplication and finding eigenvectors and eigenvalues using TensorFlow.

#### Description:-

Q) what is matrix decomposition?  
 → matrix decomposition is the process of creating multiply matrices whose product is the original matrix.

Q) what are eigenvalues and eigenvectors?  
 → eigenvalues:

An eigenvalue of  $A$  is a scalar  $\lambda$  such that the equation  $A\mathbf{v} = \lambda\mathbf{v}$  has a nontrivial solution  
 → eigenvalues may be equal to zero.

#### Eigenvectors:-

→ An eigenvector of  $A$  is a nonzero vector  $\mathbf{v}$  in  $\mathbb{R}^n$  such that  $A\mathbf{v} = \lambda\mathbf{v}$ , for some scalar  $\lambda$   
 → eigenvectors are by definition nonzero.

Q)

One problem based example given below:-

**Code :**

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows files in the current workspace, including `pract1.py`, `pract2.py`, and several documents like `.venv`, `$(PRACT_1).docx`, etc.
- Code Editor:** The `pract2.py` file is open, displaying code for tensor operations and eigenvalue calculations.
- Terminal:** Shows the command `admin@DESKTOP-3B611B0 MINGW64 /c/govindworking/Deep Learning Practical`.
- Status Bar:** Displays Python 3.9.7 64-bit (`venv`: `venv`), Ln 26, Col 1, Spaces: 4, UTF-8, CRLF, Python, Prettier, and system status (31°C, Smoke, 28-01-2022).

## **Output :**

The screenshot shows a Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help
- Title Bar:** pract2.py - Deep Learning Practical - Visual Studio Code
- Explorer:** DEEP LEARNING PRACTICAL folder containing .venv, ~\${PRACT\_1}.docx, DL(PRACT\_1).docx, DL(PRACT\_1).pdf, pract1.py, and pract2.py.
- Terminal:** The terminal window displays the output of a Python script:

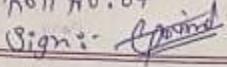
```
Matrix A:
[[3.0263908 5.916276]
 [8.942717 7.5672064]]

Eigen Vectors:
[[ -0.78932774 -0.61397287]
 [ 0.61397287 -0.78932774]]

Eigen Values:
[-3.929627 14.523224]

admin@DESKTOP-BB61I80 MINGW64 /c/govindworking/Deep Learning Practical
$ ]]
```
- Status Bar:** Ln 26, Col 1 | Spaces: 4 | UTF-8 | CRLF | Python | Prettier | 1548 | 31°C Smoke | 28-01-2022 |

## Practical No : 3

Name: Govind Saini  
 Roll No: 07  
 Sign: 

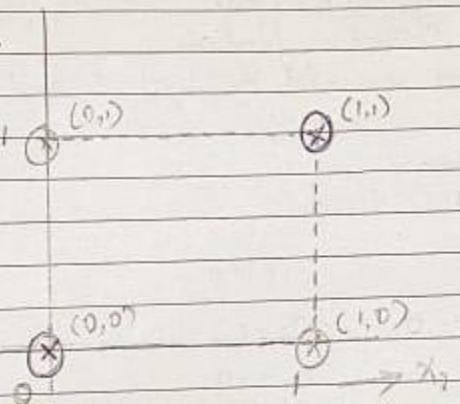
Date: 5.12.2022  
Page.....

Practical - 3

3) Aim:-  
Solving XOR problem using deep forward network.

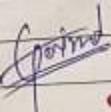
Description:-  
XOR problem:-  
Single layer perceptron, no hidden neurons  
Cannot classify input pattern - not linearly separable  
The Exclusive OR (XOR) problem

$x_1$	$x_2$	$y$
0	0	0
0	1	1
1	0	1
1	1	0

$x_2 \uparrow$   
  
 $\rightarrow x_1$

$\otimes \rightarrow 0$  pattern  
 $\circ \rightarrow 1$  pattern

Our algorithm - regardless of how it works must correctly output the XOR value each of 4 points. will be modelling this as a classification problem, so class 1 would represent an XOR value of 1, while class 0 would represent a value of 0.

  
 ©rehies

Name: Govind Saini  
Roll No: 07  
Sign: Govind

Date..... / ..... / .....  
Page.....

Single layer perceptron:-  
Let's model the problem using a single layer perceptron.

Input Data :-  
The data we'll train our model on is the value we saw for the AND function

Data	Target
[0, 0]	0
[0, 1]	1
[1, 0]	1
[1, 1]	0

The Data :-  
we next create our training data. This data is the same of each kind of logic gate, since they all take in two boolean value or input.

Training Function:-  
Here, we cycle through the data identifyingly keeping track how many consecutive datapoint we correctly classified if we managed to classify everything in one stretch, we terminate our algorithm.

perception class :-  
To bring everything together, we create a simple perception class with the function we just discussed. We have some instance Variable like the training data, target, number of input nodes and learning rate.

Govind

**Code :**

```

File Edit Selection View Go Run Terminal Help • pract3.py - Deep Learning Practical - Visual Studio Code
EXPLORER ... pract3.py 3 pract4.py 7 pt4.py pract5a.py pract5b.py pract5c.py pract2.py p3.py
DEEP LEARNING PRACTICAL > .venv
DL(PRACT_1).docx DL(PRACT_1).pdf DL(PRACT_2).docx DL(PRACT_2).pdf p3.py pract1.py pract2.py pract3.py 3 pract4.py 7 pract5a.py pract5b.py pract5c.py pt4.py sonar.csv
pract3.py > ...
1 from tflearn import DNN
2 from tflearn.layers.core import input_data, fully_connected
3 from tflearn.layers.estimator import regression
4
5 # Training examples
6 X = [[0, 0], [0, 1], [1, 0], [1, 1]]
7 Y = [[0], [1], [1], [0]]
8
9 # input layer of size 2
10 input_layer = input_data(shape=[None, 2])
11
12 # hidden layer of size 2
13 hidden_layer = fully_connected(input_layer, 2, activation='relu')
14
15 # output layer of size 1
16 output_layer = fully_connected(hidden_layer, 1, activation='sigmoid')
17
18 # use Stochastic Gradient Descent and Binary Crossentropy as loss function
19 regression = regression(output_layer, optimizer='Adam',
20                         loss='categorical_crossentropy', learning_rate=5)
21 model = DNN(regression)
22 print("\n")
23 print("7_Govind Saini")
24

```

PROBLEMS 10 OUTPUT TERMINAL DEBUG CONSOLE

admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/Deep Learning Practical

\$ []

Ln 22, Col 12 Spaces: 4 UTF-8 CRLF Python 3.9.7 (.venv:venv) ⚡ Prettier ⌂ 16:59 26°C Smoke ⌂ 05-02-2022 ⌂

```

File Edit Selection View Go Run Terminal Help • pract3.py - Deep Learning Practical - Visual Studio Code
EXPLORER ... pract3.py 3 pract4.py 7 pt4.py pract5a.py pract5b.py pract5c.py pract2.py p3.py
DEEP LEARNING PRACTICAL > .venv
DL(PRACT_1).docx DL(PRACT_1).pdf DL(PRACT_2).docx DL(PRACT_2).pdf p3.py pract1.py pract2.py pract3.py 3 pract4.py 7 pract5a.py pract5b.py pract5c.py pt4.py sonar.csv
pract3.py > ...
12 # hidden layer of size 2
13 hidden_layer = fully_connected(input_layer, 2, activation='relu')
14
15 # output layer of size 1
16 output_layer = fully_connected(hidden_layer, 1, activation='sigmoid')
17
18 # use Stochastic Gradient Descent and Binary Crossentropy as loss function
19 regression = regression(output_layer, optimizer='Adam',
20                         loss='categorical_crossentropy', learning_rate=5)
21 model = DNN(regression)
22 print("\n")
23 print("7_Govind Saini")
24
25 # fit the model
26 model.fit(X, Y, n_epoch = 100, show_metric = True)
27
28 # predict all examples
29 print('Expected: ', [i[0] > 0 for i in Y])
30 print('Predicted: ', [i[0] > 0 for i in model.predict(X)])
31
32 print("hidden layer", model.get_weights(
33     hidden_layer.W), model.get_weights(hidden_layer.b))
34 print("output layer", model.get_weights(
35     output_layer.W), model.get_weights(output_layer.b))
36

```

PROBLEMS 10 OUTPUT TERMINAL DEBUG CONSOLE

admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/Deep Learning Practical

\$ []

Ln 22, Col 12 Spaces: 4 UTF-8 CRLF Python 3.9.7 (.venv:venv) ⚡ Prettier ⌂ 16:59 26°C Smoke ⌂ 05-02-2022 ⌂

## Output :

```

pract3.py > ...
12 # hidden layer of size 2
13 hidden_layer = fully_connected(input_layer, 2, activation='relu')
14

PROBLEMS 10 OUTPUT TERMINAL DEBUG CONSOLE

7_Govind Saini
-----
Run id: 9ZNEYU
Log directory: /tmp/tflearn_logs/
-----
Training samples: 4
Validation samples: 0
--
Training Step: 1 | time: 0.402s
| AdaDelta | epoch: 001 | loss: 0.0000 - binary_acc: 0.0000 -- iter: 4/4
--
Training Step: 2 | time: 0.003s
| AdaDelta | epoch: 002 | loss: 0.0000 - binary_acc: 0.4500 -- iter: 4/4
--
Training Step: 3 | time: 0.002s
| AdaDelta | epoch: 003 | loss: 0.0000 - binary_acc: 0.4909 -- iter: 4/4
--
Training Step: 4 | time: 0.006s
| AdaDelta | epoch: 004 | loss: 0.0000 - binary_acc: 0.4977 -- iter: 4/4
--
Training Step: 5 | time: 0.003s
| AdaDelta | epoch: 005 | loss: 0.0000 - binary_acc: 0.4993 -- iter: 4/4
--
Training Step: 6 | time: 0.004s
| AdaDelta | epoch: 006 | loss: 0.0000 - binary_acc: 0.4998 -- iter: 4/4
--
Training Step: 7 | time: 0.008s
| AdaDelta | epoch: 007 | loss: 0.0000 - binary_acc: 0.4999 -- iter: 4/4
--
Training Step: 8 | time: 0.005s

```

Ln 22, Col 12 Spaces: 4 UTF-8 CRLF Python 3.9.7 ('venv': venv) ⚡ Prettier 17:00 26°C Smoke 05-02-2022

```

pract3.py > ...
12 # hidden layer of size 2
13 hidden_layer = fully_connected(input_layer, 2, activation='relu')
14

PROBLEMS 10 OUTPUT TERMINAL DEBUG CONSOLE

-- Training Step: 94 | time: 0.009s
| AdaDelta | epoch: 094 | loss: 0.0000 - binary_acc: 0.5000 -- iter: 4/4
-- Training Step: 95 | time: 0.004s
| AdaDelta | epoch: 095 | loss: 0.0000 - binary_acc: 0.5000 -- iter: 4/4
-- Training Step: 96 | time: 0.003s
| AdaDelta | epoch: 096 | loss: 0.0000 - binary_acc: 0.5000 -- iter: 4/4
-- Training Step: 97 | time: 0.014s
| AdaDelta | epoch: 097 | loss: 0.0000 - binary_acc: 0.5000 -- iter: 4/4
-- Training Step: 98 | time: 0.003s
| AdaDelta | epoch: 098 | loss: 0.0000 - binary_acc: 0.5000 -- iter: 4/4
-- Training Step: 99 | time: 0.003s
| AdaDelta | epoch: 099 | loss: 0.0000 - binary_acc: 0.5000 -- iter: 4/4
-- Training Step: 100 | time: 0.005s
| AdaDelta | epoch: 100 | loss: 0.0000 - binary_acc: 0.5000 -- iter: 4/4
--

Expected: [False, True, True, False]
Predicted: [True, True, True, True]
hidden layer [[-0.01443598 -0.02471033]
[ 0.00417513  0.03096291]] [0. 0.]
output layer [[ 0.00126254]
[-0.00187762]] [0.]
```

admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/Deep Learning Practical
\$

Ln 22, Col 12 Spaces: 4 UTF-8 CRLF Python 3.9.7 ('venv': venv) ⚡ Prettier 17:00 26°C Smoke 05-02-2022

## Practical No : 4

Name: Govind Saini  
Roll No: 07.  
Sign: Govind

Date: 20.1.2020

Page.....

### Practical - 4

4)

Aim:-

Implement deep neural network for performing binary classification task.

Description:-

Describe the problem statement:-

The given dataset comprises of the file transmitted sonar signal is a frequency-modulate chirp, rising in frequency.

The data set contains signal obtained from a variety of different aspect angles, spanning 90 degrees for the cylinder and 180 degrees for the rock.

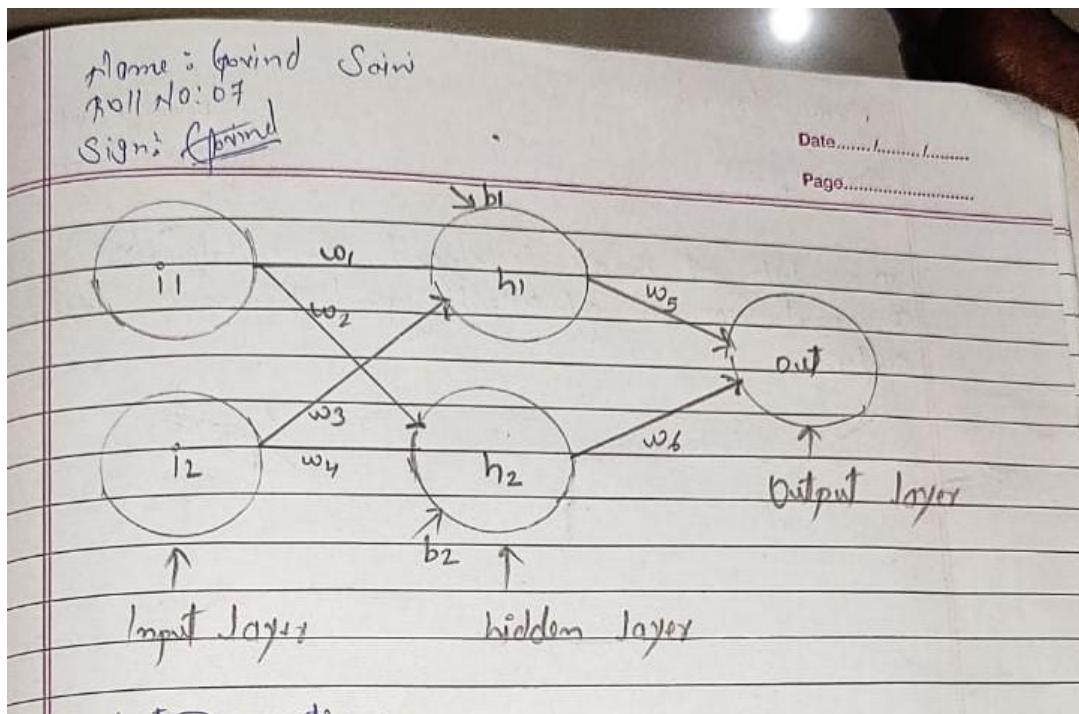
Explain the activation function used in hidden and output layer with equation

Activation function:-

Activation function decides, whether a neuron should be activated or not by calculating weighted sum and further adding bias with it.

The purpose of the activation function is to introduce non-linearity into the output of a neuron.

Variant	Equation
Linear function	i.e. $y = ax$
Sigmoid function	i.e. $A = 1/(1+e^{-x})$
Tanh function	i.e. $\tanh(x) = 2 * \text{sigmoid}(2x) - 1$ <span style="float: right;"><i>(not in notes)</i></span>

Data set Description:-

Title :- Sonar, Mines vs Rock

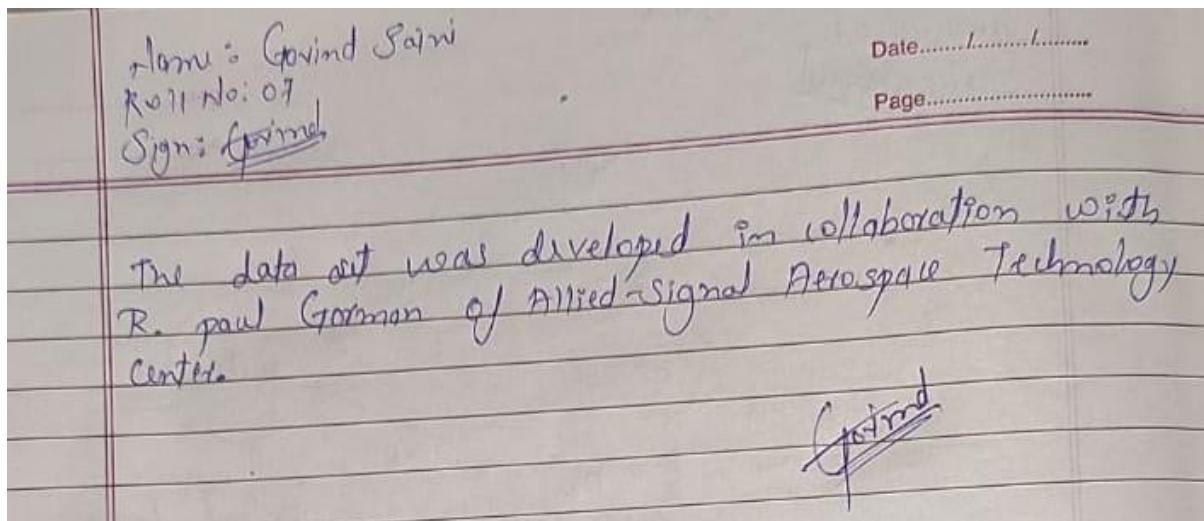
This data set can be used in a number of different ways to test learning speed, quality of ultimate learning ability to generate, or combinations of these factors. It is a binary classification problem that requires a model to differentiate rocks from metal cylinders.

field information

field name	Order	Type (format)
attribute-1	1	number (default)
attribute-2	2	number (default)
:		
class	61	string (default)

Data set source or citation:-

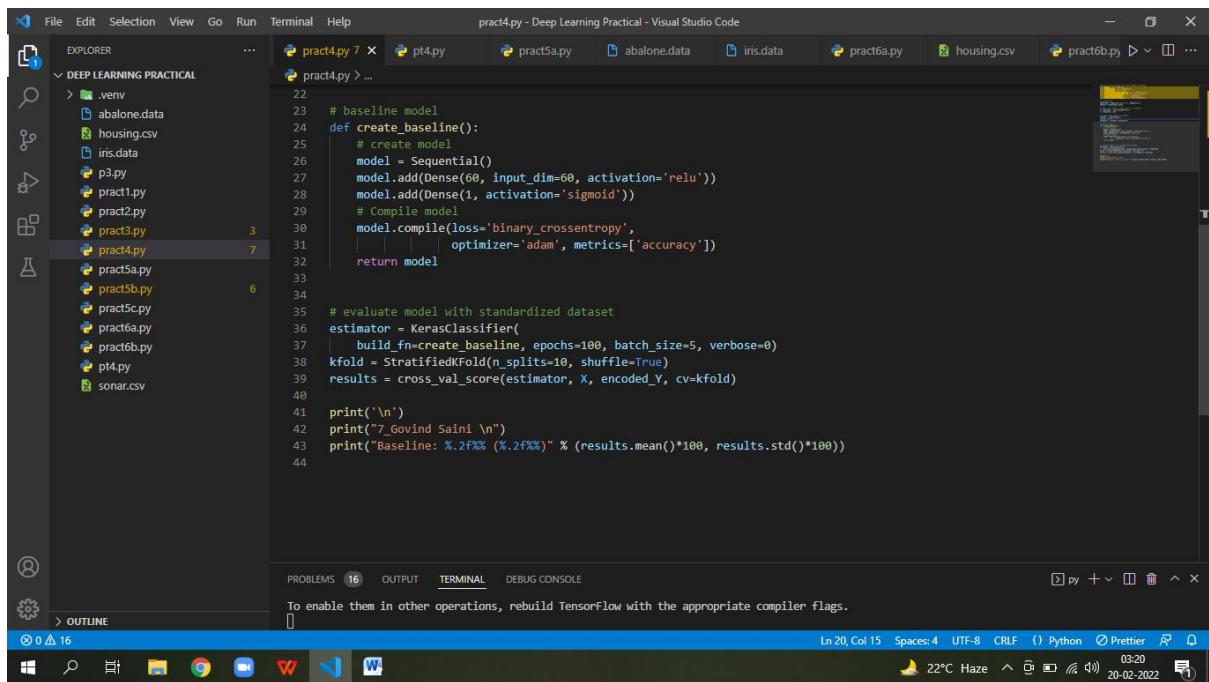
The dataset was contributed to the benchmark collection by Terry Sejnowski, now at the Salk Institute and University of California at San Diego. Govind



## **Code :**

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** On the left, it shows a tree view of files under "DEEP LEARNING PRACTICAL". The file "pract4.py" is currently selected.
- Code Editor:** The main area displays the content of "pract4.py". The code uses pandas to read a CSV dataset, defines a Sequential model with two Dense layers, and performs cross-validation using StratifiedKFold. It also encodes the class values using LabelEncoder.
- Terminal:** At the bottom, the terminal output shows a message about TensorFlow compiler flags.
- Status Bar:** The status bar at the bottom right shows the current environment: Python 3.0, 22°C Haze, and a date of 20-02-2022.

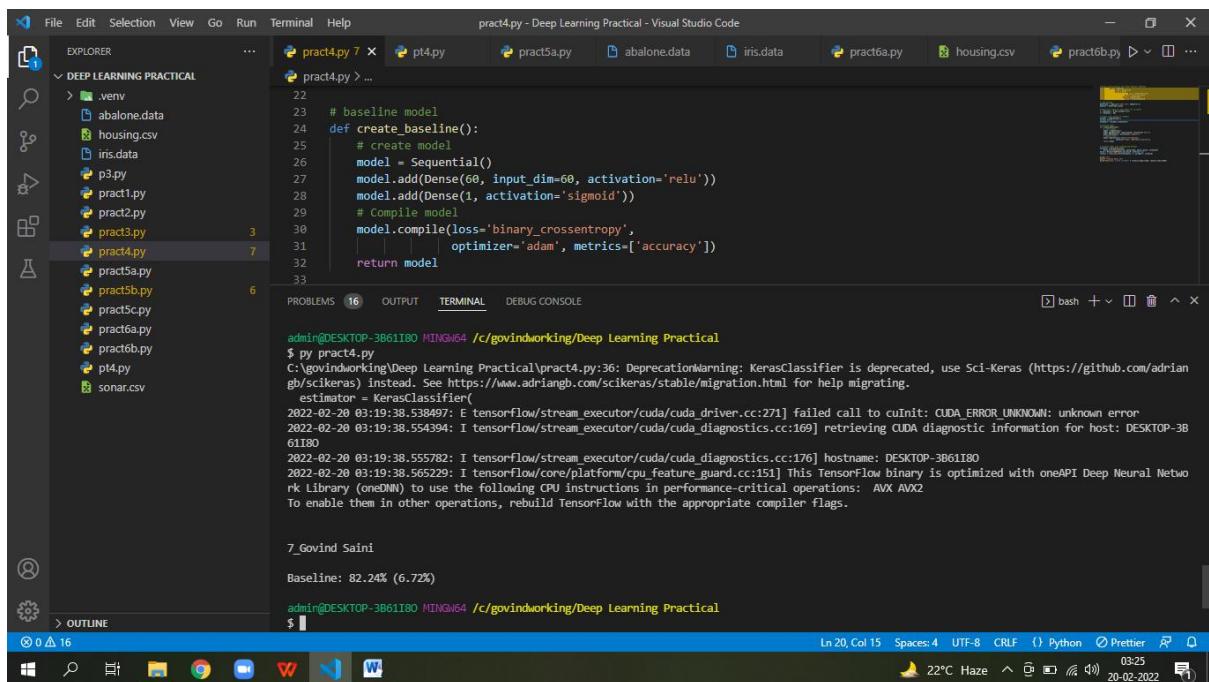


```

22
23     # baseline model
24     def create_baseline():
25         # create model
26         model = Sequential()
27         model.add(Dense(60, input_dim=60, activation='relu'))
28         model.add(Dense(1, activation='sigmoid'))
29         # Compile model
30         model.compile(loss='binary_crossentropy',
31                         optimizer='adam', metrics=['accuracy'])
32
33     return model
34
35 # evaluate model with standardized dataset
36 estimator = KerasClassifier(
37     build_fn=create_baseline, epochs=100, batch_size=5, verbose=0)
38 kFold = StratifiedKFold(n_splits=10, shuffle=True)
39 results = cross_val_score(estimator, X, encoded_Y, cv=kFold)
40
41 print('\n')
42 print("7_Govind Saini \n")
43 print("Baseline: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
44

```

## Output :



```

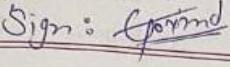
admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/Deep Learning Practical
$ py pract4.py
C:\govindworking\Deep Learning Practical\pract4.py:26: DeprecationWarning: KerasClassifier is deprecated, use Sci-Keras (https://github.com/adriangb/scikeras) instead. See https://www.adriangb.com/scikeras/stable/migration.html for help migrating.
    estimator = KerasClassifier(
2022-02-20 03:19:38.538497: E tensorflow/stream_executor/cuda/cuda_driver.cc:271] failed call to cuInit: CUDA_ERROR_UNKNOWN: unknown error
2022-02-20 03:19:38.554394: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: DESKTOP-3B61I80
2022-02-20 03:19:38.555782: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: DESKTOP-3B61I80
2022-02-20 03:19:38.556229: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

7_Govind Saini
Baseline: 82.24% (6.72%)

admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/Deep Learning Practical
$ 

```

## Practical No : 5

Name: Govind Saini  
 Roll No: 07  
 Sign: 

Date: 29.12.2020  
Page.....

Practical - 5

5a)

Aim:-

Using deep feed forward network with two hidden layers for performing multi-class classification and predicting the class

Description:-

problem Description:-

- this dataset is well studied and is a good problem for practising on neural network because all of the input variable are numeric and have the same scale in centimeters. Each instance
- the properties of an observed flower measurement and the output variable is specific Iris species.

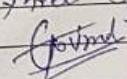
Dataset Descriptions:-

Title: Iris flower dataset

- The Iris flower dataset is a well-studied problem and so such we can expect to achieve a model accuracy in the range of 95% to 99%.
- This provide a good target to aim for when developing our models.

feature:-

- The Iris Dataset contains four features (length and width of sepal and petal) of 150 sample of three species



Name: Govind Saini  
Roll No: 07  
Sign: Govind

Date...../...../.....  
Page.....

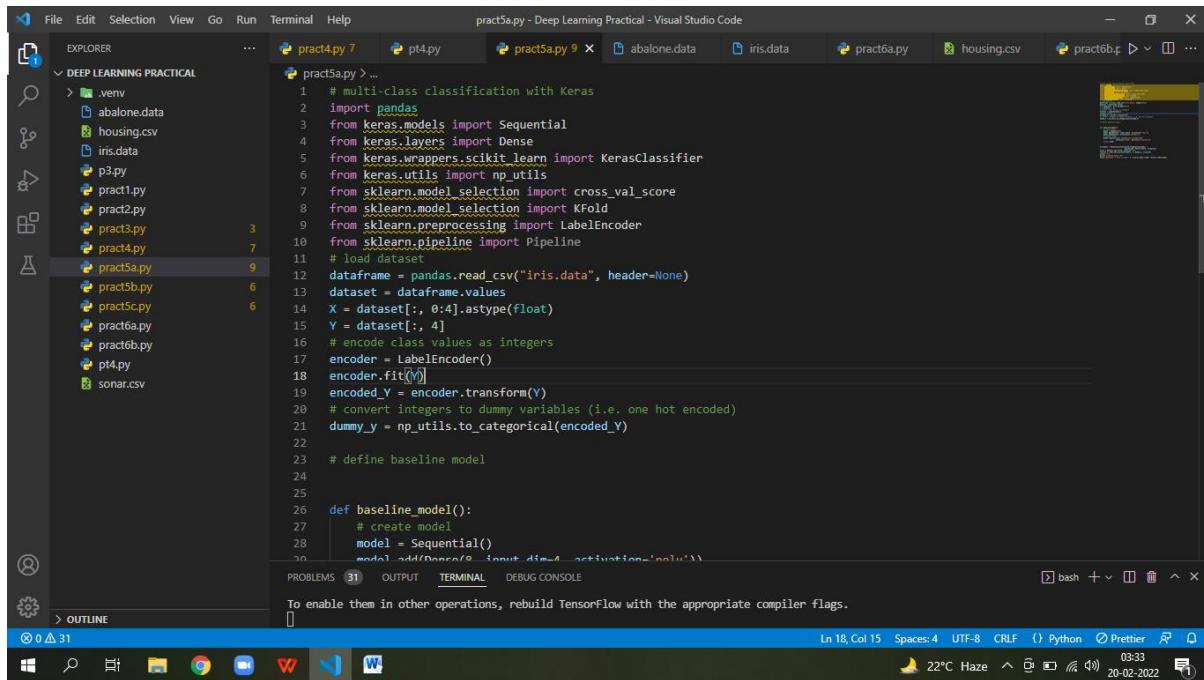
of Iris (Iris setosa, Iris virginica and Iris versicolor).  
These measures were used to create a linear discriminant model to classify the species.

Attributes information:

- i) Sepal length in cm
- ii) Sepal width in cm
- iii) Petal length in cm
- iv) Petal width in cm
- v) Class
  - Iris Setosa
  - Iris Versicolour
  - Iris Virginica

Govind

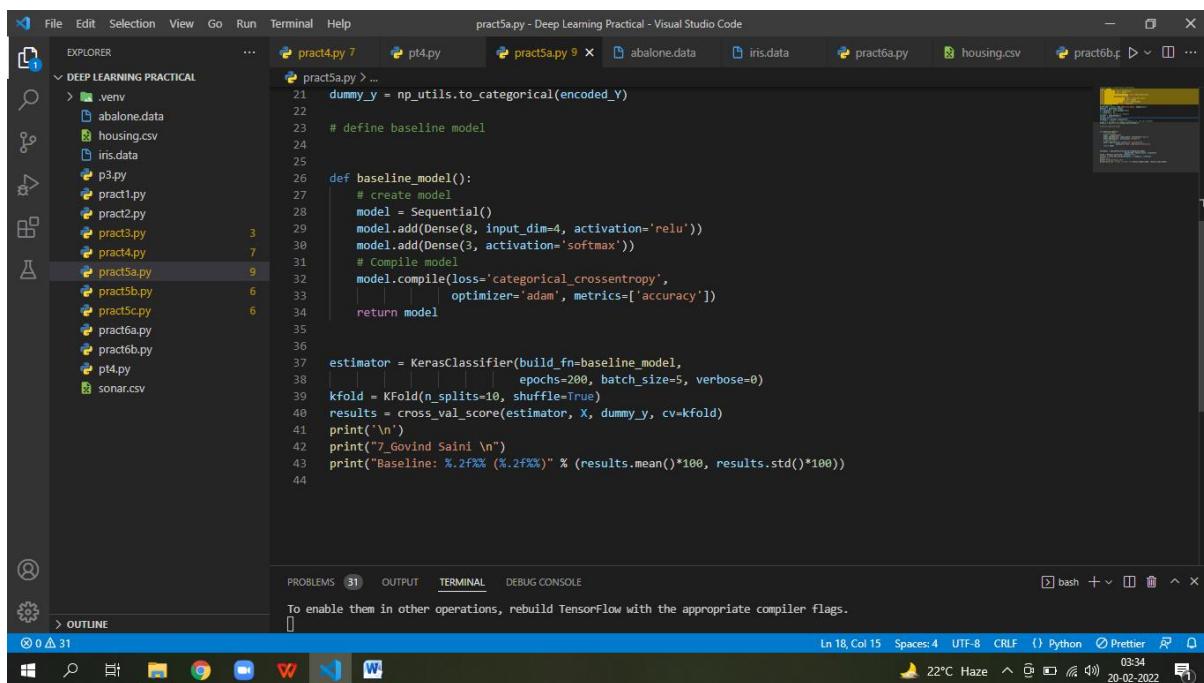
Code :



```

pract5a.py > ...
1 # multi-class classification with Keras
2 import pandas
3 from keras.models import Sequential
4 from keras.layers import Dense
5 from keras.wrappers.scikit_learn import KerasClassifier
6 from keras.utils import np_utils
7 from sklearn.model_selection import cross_val_score
8 from sklearn.model_selection import KFold
9 from sklearn.preprocessing import LabelEncoder
10 from sklearn.pipeline import Pipeline
11 # load dataset
12 dataframe = pandas.read_csv("iris.data", header=None)
13 dataset = dataframe.values
14 X = dataset[:, 0:4].astype(float)
15 Y = dataset[:, 4]
16 # encode class values as integers
17 encoder = LabelEncoder()
18 encoder.fit(Y)
19 encoded_Y = encoder.transform(Y)
20 # convert integers to dummy variables (i.e. one hot encoded)
21 dummy_y = np_utils.to_categorical(encoded_Y)
22
23 # define baseline model
24
25
26 def baseline_model():
27     # create model
28     model = Sequential()
29     model.add(Dense(8, input_dim=4, activation='relu'))
30     model.add(Dense(3, activation='softmax'))
31     # Compile model
32     model.compile(loss='categorical_crossentropy',
33                    optimizer='adam', metrics=['accuracy'])
34     return model
35
36
37 estimator = KerasClassifier(build_fn=baseline_model,
38                             epochs=200, batch_size=5, verbose=0)
39 kfold = KFold(n_splits=10, shuffle=True)
40 results = cross_val_score(estimator, X, dummy_y, cv=kfold)
41 print('\n')
42 print("Govind Saini \n")
43 print("Baseline: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
44

```



```

pract5a.py > ...
21 dummy_y = np_utils.to_categorical(encoded_Y)
22
23 # define baseline model
24
25
26 def baseline_model():
27     # create model
28     model = Sequential()
29     model.add(Dense(8, input_dim=4, activation='relu'))
30     model.add(Dense(3, activation='softmax'))
31     # Compile model
32     model.compile(loss='categorical_crossentropy',
33                    optimizer='adam', metrics=['accuracy'])
33     return model
34
35
36
37 estimator = KerasClassifier(build_fn=baseline_model,
38                             epochs=200, batch_size=5, verbose=0)
39 kfold = KFold(n_splits=10, shuffle=True)
40 results = cross_val_score(estimator, X, dummy_y, cv=kfold)
41 print('\n')
42 print("Govind Saini \n")
43 print("Baseline: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
44

```

## Output :

```
pract5a.py > ...
21 dummy_y = np_utils.to_categorical(encoded_Y)
22
23 # define baseline model
24
25
26 def baseline_model():
27     # Create model
28     model = Sequential()
29     model.add(Dense(8, input_dim=4, activation='relu'))
30     model.add(Dense(3, activation='softmax'))
31
32     # Compile model
33     model.compile(loss='categorical_crossentropy',
34                   optimizer='adam', metrics=['accuracy'])
35
36
3ls.
WARNING:tensorflow:5 out of the last 13 calls to <function Model.make_test_function.<locals>.test_function at 0x00000231488E1700> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

7_Govind Saini
Baseline: 97.33% (4.42%)
admin@DESKTOP-3B611B0 MINGW64 /c/govindworking/Deep Learning Practical
```

Name: Govind Saini  
Roll No: 07  
Sign: *Govind*

Date: 24.12.2020

Page.....

56

### Aim:-

Using a deep feed forward network with two hidden layers for performing classification and predicting the probability of class.

### Description:-

#### Problem Description:-

- When using neural network model such as regular deep feedforward net and convolution nets for classification task over some set of class label, one wonders whether it is possible to interpret the output

$$\text{i.e. } y = [0.02, 0.05, 0.93]$$

As the probability of some input being in a class equal to the respective component value.

#### Algorithms (Flowchart):-

One way to aggregate the result of each individual neural net model is to use a softmax as the ensemble output to give a final probability. In order to automatically determine the optimal weighting of the final software averaging.

Deriving the softmax function for multinomial classification problems starting from simple logistic regression. Using the softmax activation function in the output layer of a deep neural net to represent a categorical distribution over class label, and obtaining the probabilities of each input element belonging to a label.

*Govind*

Orechies

Name: Govind Saini  
 Roll No. of  
 Sign: Govind

Date..... / ..... / ..  
 Page.....

Building a robust ensemble neural net classifier with softmax output aggregation using the Keras functional API

Learning Model 1      Learning Model 2      Learning Model n  
 ...  
 Merge layer  
 Backpropagate  
 softmax aggregate  
 Output

Dataset Description:-

`sklearn.datasets.load_wine`  
 Load and return the wine dataset (classification)  
 Now, new in version 0.18.  
 The wine dataset is a classic and very easy multi-class classification dataset.  
 classes      3  
 samples per class      [59, 71, 48]  
 samples total      178  
 Dimensionality      13  
 features      real, positive

Data:

Dictionary-like object, with the following attributes

Govind

Name : Govind Saini  
 Roll No. 07  
 Sign : Govind

Date ..... / ..... / .....

Page.....

data : {ndarray, dataframe} of shape (178, 13)  
 The data matrix. If as\_frame=True, data will be pandas DataFrame

target : {ndarray, series} of shape(178)  
 The classification target if as\_frame=True, target will be pandas Series

feature names : list  
 The names of the dataset column

target names : list  
 The names of target classes.

frame : Dataframe of shape (178, 14)  
 Only present when as\_frame=True. Dataframe with data and target.

DESC : str  
 The full description of the dataset.

Govind

**Code :**

```

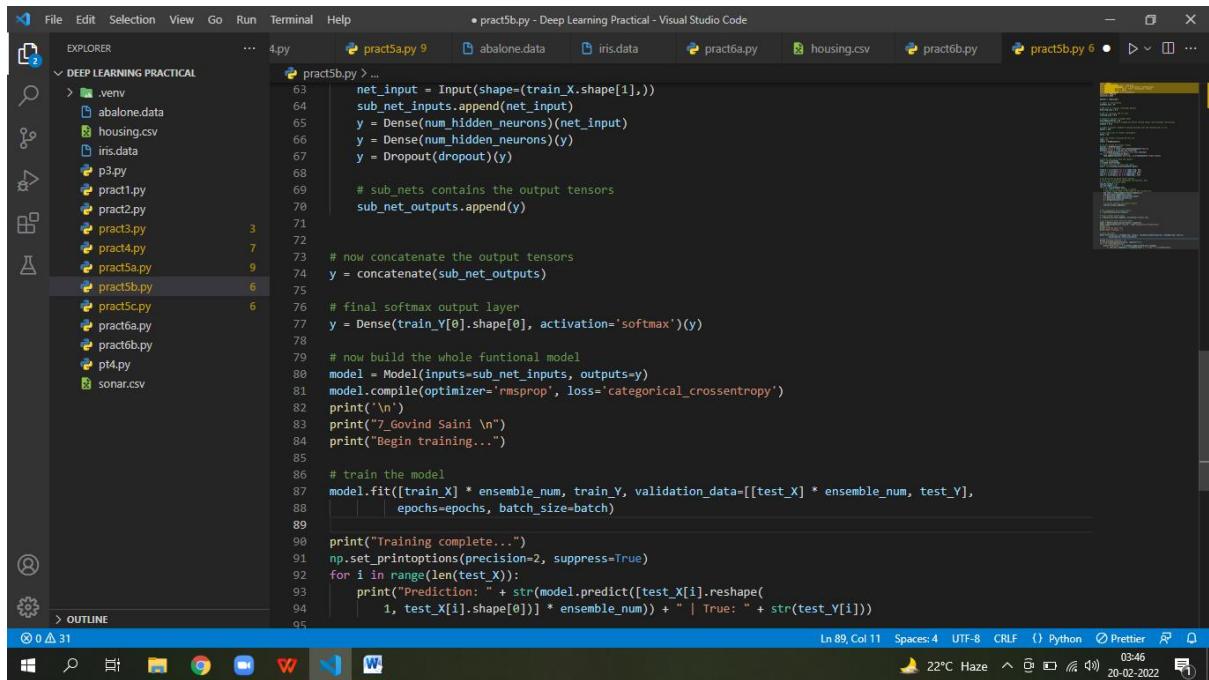
1 import numpy as np
2 from sklearn.datasets import load_wine
3 from sklearn.preprocessing import MinMaxScaler, OneHotEncoder
4 from keras.layers import Dense, Input, concatenate, Dropout
5 from tensorflow.keras import optimizers
6 optimizers.RMSprop
7 optimizers.Adam
8
9
10 dataset = load_wine()
11
12 # number of sub-networks
13 ensemble_num = 10
14
15 # 88% size of original (training) dataset
16 bootstrap_size = 0.8
17
18 # 80% for training, 20% for test
19 training_size = 0.8
20
21 # number of neurons in hidden layer
22 num_hidden_neurons = 10
23 # percentage of weights dropped out before softmax output (this prevents overfitting)
24 dropout = 0.25
25
26 # number of epochs (complete training episodes over the training set) to run
27 epochs = 100
28
29 # mini batch size for better convergence
30 batch = 10
31
32 # get the holdout training and test set
33 temp = []

```

```

31
32 # get the holdout training and test set
33 temp = []
34 scaler = MinMaxScaler()
35
36 # one hot encode the target classes
37 one_hot = OneHotEncoder()
38 dataset['data'] = scaler.fit_transform(dataset['data'])
39 dataset['target'] = one_hot.fit_transform(
40 | np.reshape(dataset['target'], (-1, 1)).toarray())
41 for i in range(len(dataset.data)):
42 | temp.append([dataset['data'][i], np.array(dataset['target'][i])])
43
44 # shuffle the row of data and targets
45 temp = np.array(temp)
46 np.random.shuffle(temp)
47 # holdout training and test stop index
48 stop = int(training_size*len(dataset.data))
49
50 train_X = np.array([x for x in temp[:stop, 0]])
51 train_Y = np.array([x for x in temp[:stop, 1]])
52 test_X = np.array([x for x in temp[stop:, 0]])
53 test_Y = np.array([x for x in temp[stop:, 1]])
54
55 # now build the ensemble neural network
56 # first, let's build the individual sub-networks, each
57 # as a Keras functional model.
58 sub_net_outputs = []
59 sub_net_inputs = []
60 for i in range(ensemble_num):
61 | # two hidden layers to keep it simple
62 | # specify input shape to the shape of the training set
63 net_input = Input(shape=(train_X.shape[1]))

```

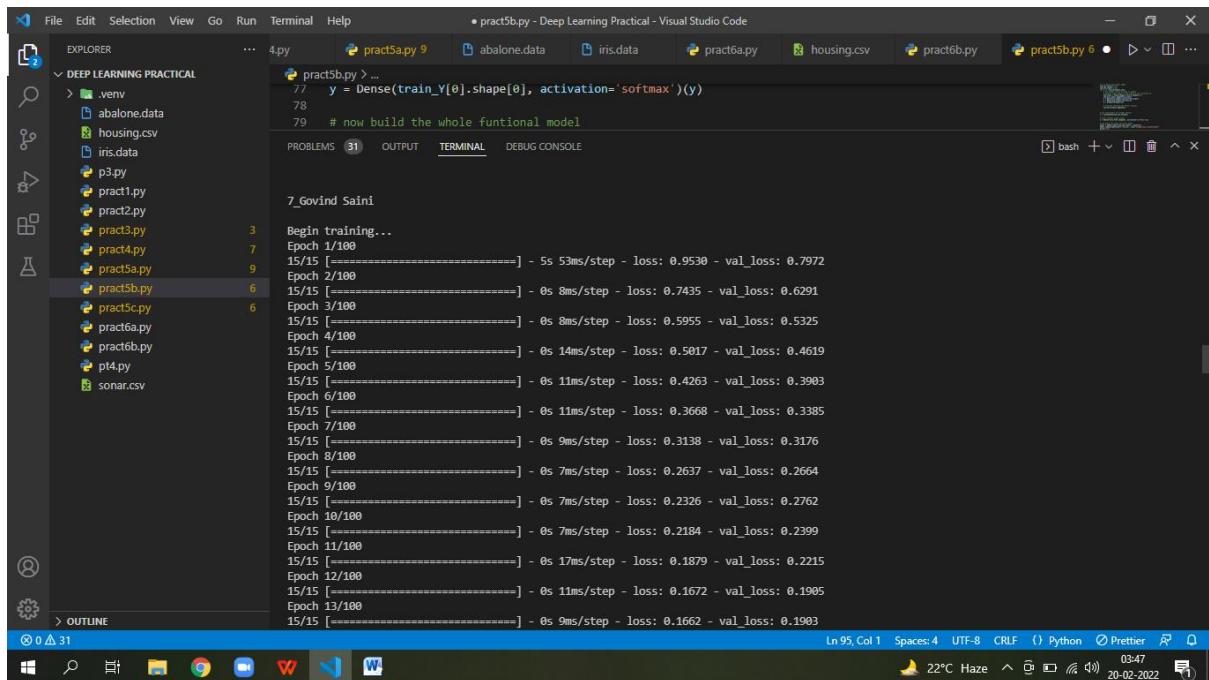


```

File Edit Selection View Go Run Terminal Help
pract5a.py 9 4.py pract5b.py 6 pract6a.py pract6b.py pract6c.py
pract5b.py 6
pract5b.py > ...
63     net_input = Input(shape=(train_X.shape[1],))
64     sub_net_inputs.append(net_input)
65     y = Dense(num_hidden_neurons)(net_input)
66     y = Dense(num_hidden_neurons)(y)
67     y = Dropout(dropout)(y)
68
69     # sub_nets contains the output tensors
70     sub_net_outputs.append(y)
71
72
73     # now concatenate the output tensors
74     y = concatenate(sub_net_outputs)
75
76     # final softmax output layer
77     y = Dense(train_Y[0].shape[0], activation='softmax')(y)
78
79     # now build the whole functional model
80     model = Model(inputs=sub_net_inputs, outputs=y)
81     model.compile(optimizer='rmsprop', loss='categorical_crossentropy')
82     print("\n")
83     print("7_Govind Saini \n")
84     print("Begin training...")
85
86     # train the model
87     model.fit([train_X] * ensemble_num, train_Y, validation_data=[test_X] * ensemble_num, test_Y,
88               epochs=epochs, batch_size=batch)
89
90     print("Training complete...")
91     np.set_printoptions(precision=2, suppress=True)
92     for i in range(len(test_X)):
93         print("Prediction: " + str(model.predict([test_X[i].reshape(
94             1, test_X[i].shape[0]]) * ensemble_num)) + " | True: " + str(test_Y[i]))
95

```

## Output :



```

File Edit Selection View Go Run Terminal Help
pract5a.py 9 4.py pract5b.py 6 pract6a.py pract6b.py pract6c.py
pract5b.py 6
pract5b.py > ...
77     y = Dense(train_Y[0].shape[0], activation='softmax')(y)
78
79     # now build the whole functional model
7_Govind Saini
Begin training...
Epoch 1/100
15/15 [=====] - 5s 53ms/step - loss: 0.9530 - val_loss: 0.7972
Epoch 2/100
15/15 [=====] - 0s 8ms/step - loss: 0.7435 - val_loss: 0.6291
Epoch 3/100
15/15 [=====] - 0s 8ms/step - loss: 0.5955 - val_loss: 0.5325
Epoch 4/100
15/15 [=====] - 0s 14ms/step - loss: 0.5017 - val_loss: 0.4619
Epoch 5/100
15/15 [=====] - 0s 11ms/step - loss: 0.4263 - val_loss: 0.3905
Epoch 6/100
15/15 [=====] - 0s 11ms/step - loss: 0.3668 - val_loss: 0.3385
Epoch 7/100
15/15 [=====] - 0s 9ms/step - loss: 0.3138 - val_loss: 0.3176
Epoch 8/100
15/15 [=====] - 0s 7ms/step - loss: 0.2637 - val_loss: 0.2664
Epoch 9/100
15/15 [=====] - 0s 7ms/step - loss: 0.2326 - val_loss: 0.2762
Epoch 10/100
15/15 [=====] - 0s 7ms/step - loss: 0.2184 - val_loss: 0.2399
Epoch 11/100
15/15 [=====] - 0s 17ms/step - loss: 0.1879 - val_loss: 0.2215
Epoch 12/100
15/15 [=====] - 0s 11ms/step - loss: 0.1672 - val_loss: 0.1905
Epoch 13/100
15/15 [=====] - 0s 9ms/step - loss: 0.1662 - val_loss: 0.1903

```

```

    pract5b.py > ...
    77 y = Dense(train_Y[0].shape[0], activation='softmax')(y)
    78
    79 # now build the whole functional model
    Epoch 98/100
    15/15 [=====] - 0s 7ms/step - loss: 0.0033 - val_loss: 0.0252
    Epoch 99/100
    15/15 [=====] - 0s 7ms/step - loss: 0.0115 - val_loss: 0.0124
    Epoch 100/100
    15/15 [=====] - 0s 9ms/step - loss: 0.0029 - val_loss: 0.0217
    Training complete...
    1/1 [=====] - 0s 369ms/step
    Prediction: [[1. 0. 0.]] | True: [1. 0. 0.]
    1/1 [=====] - 0s 54ms/step
    Prediction: [[0. 0. 1.]] | True: [0. 0. 1.]
    1/1 [=====] - 0s 104ms/step
    Prediction: [[0.03 0.96 0.01]] | True: [0. 1. 0.]
    1/1 [=====] - 0s 41ms/step
    Prediction: [[1. 0. 0.]] | True: [1. 0. 0.]
    1/1 [=====] - 0s 31ms/step
    Prediction: [[0. 1. 0.]] | True: [0. 1. 0.]
    1/1 [=====] - 0s 59ms/step
    Prediction: [[0. 0. 1.]] | True: [0. 0. 1.]
    1/1 [=====] - 0s 57ms/step
    Prediction: [[0.93 0.07 0. ]] | True: [1. 0. 0.]
    1/1 [=====] - 0s 47ms/step
    Prediction: [[1. 0. 0.]] | True: [1. 0. 0.]
    1/1 [=====] - 0s 26ms/step
    Prediction: [[1. 0. 0.]] | True: [1. 0. 0.]
    1/1 [=====] - 0s 75ms/step
    Prediction: [[0. 0. 1.]] | True: [0. 0. 1.]
    1/1 [=====] - 0s 72ms/step
    Prediction: [[0. 0. 1.]] | True: [0. 0. 1.]
    1/1 [=====] - 0s 62ms/step
    Prediction: [[0. 1. 0.]] | True: [0. 1. 0.]

```

```

    pract5b.py > ...
    77 y = Dense(train_Y[0].shape[0], activation='softmax')(y)
    78
    79 # now build the whole functional model
    Prediction: [[0. 1. 0.]] | True: [0. 1. 0.]
    1/1 [=====] - 0s 88ms/step
    Prediction: [[1. 0. 0.]] | True: [1. 0. 0.]
    1/1 [=====] - 0s 48ms/step
    Prediction: [[0.01 0.99 0. ]] | True: [0. 1. 0.]
    1/1 [=====] - 0s 42ms/step
    Prediction: [[0. 1. 0.]] | True: [0. 1. 0.]
    1/1 [=====] - 0s 31ms/step
    Prediction: [[0. 1. 0.]] | True: [0. 1. 0.]
    1/1 [=====] - 0s 47ms/step
    Prediction: [[0. 1. 0.]] | True: [0. 1. 0.]
    1/1 [=====] - 0s 90ms/step
    Prediction: [[0. 1. 0.]] | True: [0. 1. 0.]
    1/1 [=====] - 0s 31ms/step
    Prediction: [[0. 0. 1.]] | True: [0. 0. 1.]
    1/1 [=====] - 0s 42ms/step
    Prediction: [[0. 1. 0.]] | True: [0. 1. 0.]
    1/1 [=====] - 0s 48ms/step
    Prediction: [[0. 0. 1.]] | True: [0. 0. 1.]
    1/1 [=====] - 0s 62ms/step
    Prediction: [[0. 1. 0.]] | True: [0. 1. 0.]
    1/1 [=====] - 0s 47ms/step
    Prediction: [[1. 0. 0.]] | True: [1. 0. 0.]
    1/1 [=====] - 0s 35ms/step
    Prediction: [[0. 0. 57 0.43]] | True: [0. 1. 0.]
    1/1 [=====] - 0s 40ms/step
    Prediction: [[0. 1. 0.]] | True: [0. 1. 0.]

```

admin@DESKTOP-3B611B0 MINGW64 /c/govindworking/Deep Learning Practical

Name:- Govind Saini  
 Roll No: 07  
 Sign: Govind

Date 20.1.2020  
 Page.....

50

Aim:-

Using a deep feed forward network with two hidden layers for performing multilinear regression and predicting values.

Description:-

Problem Description:-

- The dataset describe 13 numerical properties of house in Boston suburbs and is concerned with modeling the price of house in those suburbs in thousand of dollars.
- As a input attribute include things like crime rate, proportion of non-retail business acres, chemical concentrations and more.

Data Description:

Title : Boston house price dataset.

The Boston housing Dataset is a derived from information collected by US Census Service concerning housing in the area of Boston MA. The following describe the dataset columns.

CRIM - per capita crime rate by town

ZN - proportion of residential land zoned for lots over 25,000 sq.ft.

INDUS - proportion of non-retail business acres per town.

Govind

Name:- Gorind Saini  
 Roll No.: 07  
 Sign:- Gorind

Date..... / ..... / .....,  
 Page.....

CHAS - Charles River dummy variable  
 NOX - nitric oxide concentration (parts per 10 million)  
 RM - average number of rooms per dwelling  
 AGE - proportion of owner-occupied units built prior to 1940  
 DIS - weighted distances to five Boston employment centers  
 RAD - index of accessibility to radial highways  
 TAX - full-value property tax per \$10,000.  
 PTRATIO - pupil-teacher ratio by town  
 B -  $100(Bk - 0.63)^2$  where  $Bk$  is the proportion of black by town  
 LSTAT - % lower status of the population  
 MEDV - median value of owner-occupied home in \$1000's

Gorind

Gavind

## Code :

The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- File Explorer (Left):** Shows a folder named "DEEP LEARNING PRACTICAL" containing files like ".venv", "abalone.data", "housing.csv", "iris.data", "p3.py", "pract1.py", "pract2.py", "pract3.py", "pract4.py", "pract5a.py", "pract5b.py", "pract5c.py" (which is the active file), "pract6a.py", "pract6b.py", "pt4.py", and "sonar.csv".
- Code Editor (Center):** Displays the content of "pract5c.py". The code uses pandas to read "housing.csv", splits it into X and Y variables, defines a baseline model (Sequential neural network), and compiles it with mean squared error loss and Adam optimizer.
- Terminal (Bottom):** Shows the command "pract5c - Deep Learning Practical - Visual Studio Code".
- Status Bar (Bottom):** Shows file paths, line numbers (Ln 27, Col 5), tab size (Tab Size: 4), encoding (UTF-8), file type (Python), and a Prettier icon.

The screenshot shows the Visual Studio Code interface. The code editor displays the `t5a.py` file, which contains Python code for a machine learning model. The terminal tab shows the command `python t5a.py` being run, and the status bar indicates the terminal size is 27x5, the file is UTF-8 encoded, and it was last modified at 03:50 on 20-02-2022.

```

File Edit Selection View Go Run Terminal Help
pract5cpy - Deep Learning Practical - Visual Studio Code
EXPLORER ... t5a.py 9 abalone.data iris.data pract6a.py pract6b.py practsb.py 6 pract5cpy 6 ...
DEEP LEARNING PRACTICAL
> .venv
abalone.data housing.csv iris.data
p3.py pract1.py pract2.py pract3.py 3 pract4.py pract5.py 9 pract5b.py 6 pract5cpy 6
pract6a.py pract6b.py p4.py sonar.csv
pract5cpy > ...
14 Y = dataset[:,13]
15
16 # define base model
17 def baseline_model():
18
19     # create model
20     model = Sequential()
21     model.add(Dense(13, input_dim=13, kernel_initializer='normal', activation='relu'))
22     model.add(Dense(1, kernel_initializer='normal'))
23
24     # Compile model
25     model.compile(loss='mean_squared_error', optimizer='adam')
26     return model
27
28 # evaluate model
29 estimator = KerasRegressor(build_fn=baseline_model, epochs=100, batch_size=5, verbose=0)
30 kfold = KFold(n_splits=10)
31 results = cross_val_score(estimator, X, Y, cv=kfold)
32 print("\n")
33 print("7_Govind Saini \n")
34 print("Baseline: %.2f (%.2f) MSE" % (results.mean(), results.std()))

```

PROBLEMS 31 OUTPUT TERMINAL DEBUG CONSOLE

To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

Ln 27, Col 5 Tab Size 4 UTF-8 CRLF Python Prettier 03:50 22°C Haze 20-02-2022

## Output :

The screenshot shows the Visual Studio Code interface with the terminal tab active. The terminal output shows the command `python t5a.py` being run, followed by several warning messages from TensorFlow about deprecated KerasRegressor usage and CUDA errors. The output ends with the message "Baseline: -28.67 (24.37) MSE". The status bar indicates the terminal size is 27x5, the file is UTF-8 encoded, and it was last modified at 03:52 on 20-02-2022.

```

File Edit Selection View Go Run Terminal Help
pract5cpy - Deep Learning Practical - Visual Studio Code
EXPLORER ... t5a.py 9 abalone.data iris.data pract6a.py pract6b.py practsb.py 6 pract5cpy 6 ...
DEEP LEARNING PRACTICAL
> .venv
abalone.data housing.csv iris.data
p3.py pract1.py pract2.py pract3.py 3 pract4.py pract5.py 9 pract5b.py 6 pract5cpy 6
pract6a.py pract6b.py p4.py sonar.csv
pract5cpy > ...
14 Y = dataset[:,13]
15
16 # define base model
17 def baseline_model():
18
19     # create model
20     model = Sequential()
21     model.add(Dense(13, input_dim=13, kernel_initializer='normal', activation='relu'))
22     model.add(Dense(1, kernel_initializer='normal'))
23
24     # Compile model
25     model.compile(loss='mean_squared_error', optimizer='adam')
26     return model
27
28 # evaluate model
29 estimator = KerasRegressor(build_fn=baseline_model, epochs=100, batch_size=5, verbose=0)
30 kfold = KFold(n_splits=10)
31 results = cross_val_score(estimator, X, Y, cv=kfold)
32 print("\n")
33 print("7_Govind Saini \n")
34 print("Baseline: %.2f (%.2f) MSE" % (results.mean(), results.std()))

```

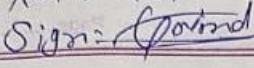
PROBLEMS 31 OUTPUT TERMINAL DEBUG CONSOLE

admin@DESKTOP-3B61180 MINGW64 /c/govindworking/Deep Learning Practical
\$ py pract5cpy
C:\govindworking\Deep Learning Practical\pract5c.py:51: DeprecationWarning: KerasRegressor is deprecated, use Sci-Keras (<https://github.com/adriangb/scikeras>) instead. See <https://www.adriangb.com/scikeras/stable/migration.html> for help migrating.
estimator = KerasRegressor(build\_fn=baseline\_model, epochs=100, batch\_size=5, verbose=0)
2022-02-20 03:49:12.314604: E tensorflow/stream\_executor/cuda/cuda\_driver.cc:271] failed call to cuInit: CUDA\_ERROR\_UNKNOWN: unknown error
2022-02-20 03:49:12.321987: I tensorflow/stream\_executor/cuda/cuda\_diagnostics.cc:169] retrieving CUDA diagnostic information for host: DESKTOP-3B61180
2022-02-20 03:49:12.323772: I tensorflow/stream\_executor/cuda/cuda\_diagnostics.cc:176] hostname: DESKTOP-3B61180
2022-02-20 03:49:12.325986: I tensorflow/core/platform/cpu\_feature\_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.

7\_Govind Saini
Baseline: -28.67 (24.37) MSE

Ln 27, Col 5 Tab Size 4 UTF-8 CRLF Python Prettier 03:52 22°C Haze 20-02-2022

## Practical No : 6

Name: Govind Saini Roll No: 07 Sign: 	Date...../...../..... Page.....
--	------------------------------------

Practical - 6

6a)

Aim:-  
 Evaluating feed forward deep network for regression using k-fold cross validation.

Description:-

problem statement :-  
 Unfortunately, there is no single method that works best for all kinds of problem statement.  
 Often, a custom cross validation technique based on a feature, or combination of features, could be created if that gives the most stable cross validation score while making submission in hackathons.

Algorithm :-

- Pick a number of fold -  $k$  Usually,  $k$  is 5 or 10 but you can choose any number which is less than the dataset length
- split the dataset into  $k$  equal (if possible) parts (they are called folds)
- choose  $k-1$  folds as the training set. The remain fold will be the test set
- Train the model on the training set. On each iteration of cross-validation, you must train a new model independently of the model trained on the iteration.

Govind

Name: Govind Saini  
Roll No: 07  
Sig: Govind

Date...../...../.....  
Page.....

- Validate on the test set.
- Save the result of the validation.

Explain k-fold validation and its merit:

k-fold

Test      Train on ( $k-1$ ) split

- k-fold cross-validation in addition, we outline an overview of the different matrix used to evaluate the chosen classifier.
- In general, the criteria of comparison between algorithm can be done by measuring accuracy, speed, scalability, interpretability and robustness the ability to construct a model efficiently when the classifier is applied to the large set of data.

Dataset description :-

Title :- Ames - Iowa housing.csv

The Ames housing dataset examine features of houses sold in Ames during the 200+10 time frame. The goal is to use the training data to predict

~~Govind~~

Name: Govind Saini  
Roll No: 07  
Sig: Govind

Date...../...../.....  
Page.....

The sale prices of the house in the testing data. The following describe the dataset columns CRIM, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT, MEDV.

~~Govind~~

**Code :**

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** On the left, it lists files and folders. The file `pract6a.py` is currently selected.
- Code Editor:** The main area displays the content of `pract6a.py`. The code uses scikit-learn's Pipeline and KFold to build a regression model. It includes imports for `StandardScaler`, `KerasRegressor`, and `cross_val_score`.
- Terminal:** At the bottom, the terminal window shows the command `admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/Deep Learning Practical` and the prompt `$`.
- Status Bar:** The status bar at the bottom right shows the current file is `pract6a.py`, the line number is 39, the column number is 27, and the encoding is UTF-8.

## **Output :**

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** On the left, it lists files and folders including .env, .venv.rar, abalone.data, ames\_iowa\_housing.csv, ames.csv, car\_evaluation.csv, data.csv, housing.csv, iris.data, p3.py, penguins.csv, pract1.py, pract2.py, pract3.py, pract4.py, pract5a.py (highlighted in yellow), pract5b.py (highlighted in green), pract5c.py (highlighted in blue), pract6a.py (highlighted in orange), pract6b.py (highlighted in red), p4.py, sonar.csv, and training\_labels.csv.
- Terminal:** The main area shows the output of a Python script named `pract6a.py`. The script is using a pipeline of estimators: StandardScaler, RandomForestRegressor, and VotingClassifier. It processes the `abalone.data` file and outputs accuracy metrics for each estimator and the final ensemble model. The output includes:

```
estimators = []
estimators.append('standardization', StandardScaler())
estimators.append('random_forest', RandomForestRegressor(n_estimators=100))
estimators.append('voting_classifier', VotingClassifier(estimators=estimators))

for i in range(1, 10):
    print(f'Epoch {i}/10')
    X_train, X_test, y_train, y_test = train_test_split(
        data.drop(['target'], axis=1),
        data['target'],
        test_size=0.2,
        random_state=i
    )
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = mean_squared_error(y_test, y_pred, squared=False)
    print(f'Accuracy: {accuracy} ({accuracy * 100:.2f} %)
```
- Status Bar:** At the bottom, it shows the current file is `master*`, the commit hash is `0x0 38`, the terminal has 38 tabs open, the current line is Ln 39, Col 27, and the status bar includes icons for search, file, browser, terminal, and system information like temperature (30°C) and battery level (Smoke).

(After changing neuron)

```
19  
20     # create model  
21     model = Sequential()  
22     model.add(Dense(15, input_dim=13,  
23                   kernel_initializer='normal', activation='relu'))  
24     model.add(Dense(20, kernel_initializer='normal', activation='relu'))  
25     model.add(Dense(1, kernel_initializer='normal'))  
26     # Compile model  
27     model.compile(loss='mean_squared_error', optimizer='adam')  
28  
29     return model
```

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows files including `.venv`, `vennvar`, `abalone.data`, `ames_iowa_housing.csv`, `ames.csv`, `car_evaluation.csv`, `data.csv`, `housing.csv`, `iris.data`, `p3.py`, `penguins.csv`, `pract1.py`, `pract2.py`, `pract3.py`, `pract4.py`, `pract5a.py`, `pract5b.py`, `pract5c.py`, `pract6a.py` (the active file), `pract6b.py`, `p4.py`, `sonar.csv`, and `training_labels.csv`.
- Terminal (Top):** Shows the command `pract6a.py 8` running.
- Code Editor (Top):** Displays the content of `pract6a.py` which imports Sequential, Dense, KerasRegressor, cross\_val\_score, and KFold from Keras and Sklearn respectively.
- Output (Bottom):** Shows the execution progress and loss values for each epoch of the neural network training. The output includes:

```
36/36 [=====] - 1s 2ms/step - loss: 21882.5547
Epoch 2/10
36/36 [=====] - 0s 2ms/step - loss: 21849.0645
Epoch 3/10
36/36 [=====] - 0s 2ms/step - loss: 21727.0312
Epoch 4/10
36/36 [=====] - 0s 3ms/step - loss: 21462.0625
Epoch 5/10
36/36 [=====] - 0s 4ms/step - loss: 20784.5332
Epoch 6/10
36/36 [=====] - 0s 3ms/step - loss: 19891.8242
Epoch 7/10
36/36 [=====] - 0s 2ms/step - loss: 18917.7520
Epoch 8/10
36/36 [=====] - 0s 2ms/step - loss: 18159.8301
36/36 [=====] - 0s 2ms/step - loss: 17762.2383
Epoch 9/10
36/36 [=====] - 0s 2ms/step - loss: 17471.3984
4/4 [=====] - 0s 6ms/step - loss: 16654.0977
```
- Bottom Status Bar:** Shows the commit hash `master^`, file count `0`, and other system information.

Name: Govind Saini  
 Roll No: 07  
 Sign: Govind

Date ..... / ..... / .....  
 Page .....

(b) Aim:-

Evaluating feed forward & deep network for multi-class classification using k-fold validation

Description:-

problem Description:-

This is a multi-class classification problem, meaning that there are more than two classes to be predicted. In fact there are three flower species.

This is an important type of problem on which to practice with neural network because the three class value require specialized handling.

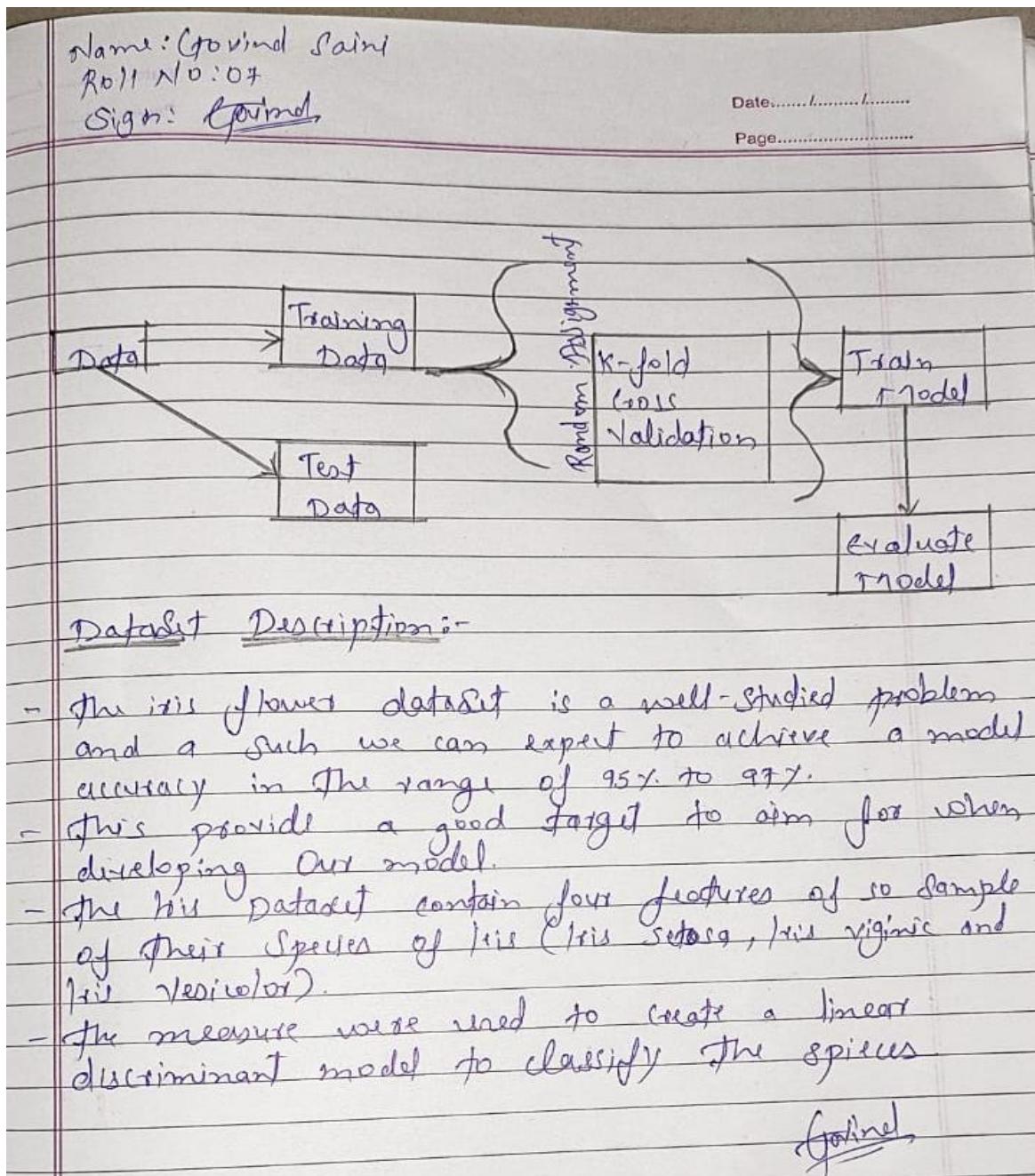
Let us previous we study k-fold validation and this

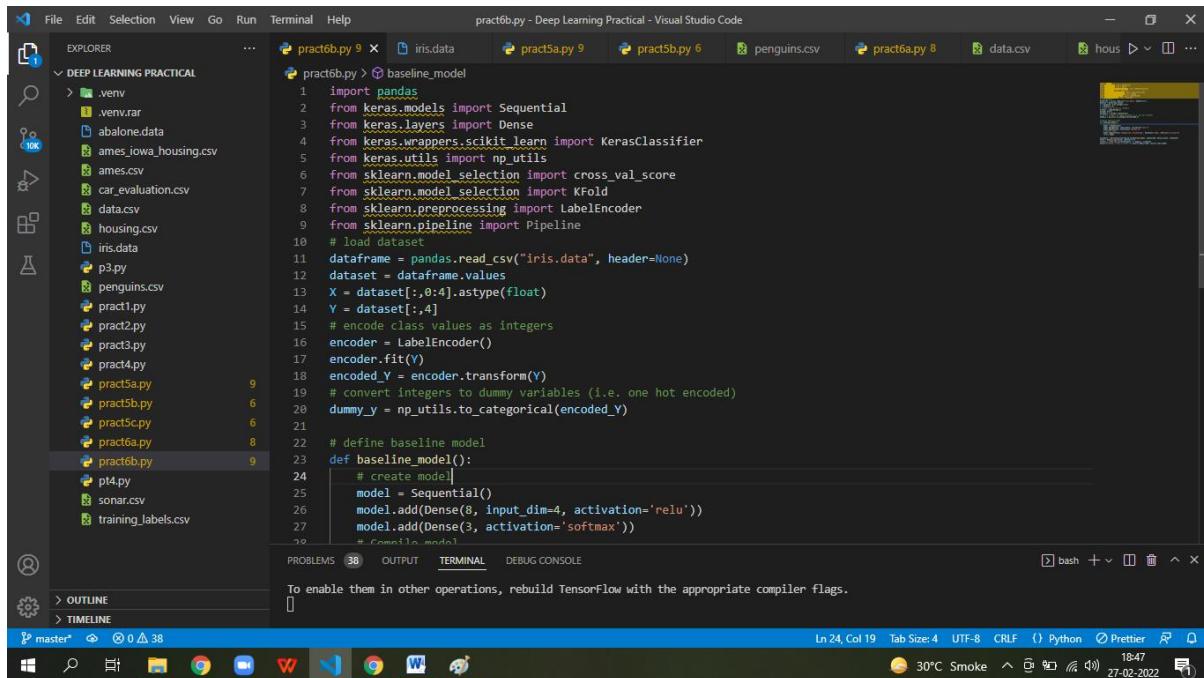
is k-fold Cross Validation:

It is type of predictive analytics and involves the following step

- Original data are split randomly into k subsample (e.g., k=5), which become the training data
- model are estimated using  $k-1$  subsample for each fold with the  $k^{\text{th}}$  subsample serving as the validation sample - process repeat until every subsample has served as the validation data and model result can be averaged across folds.

*Govind*

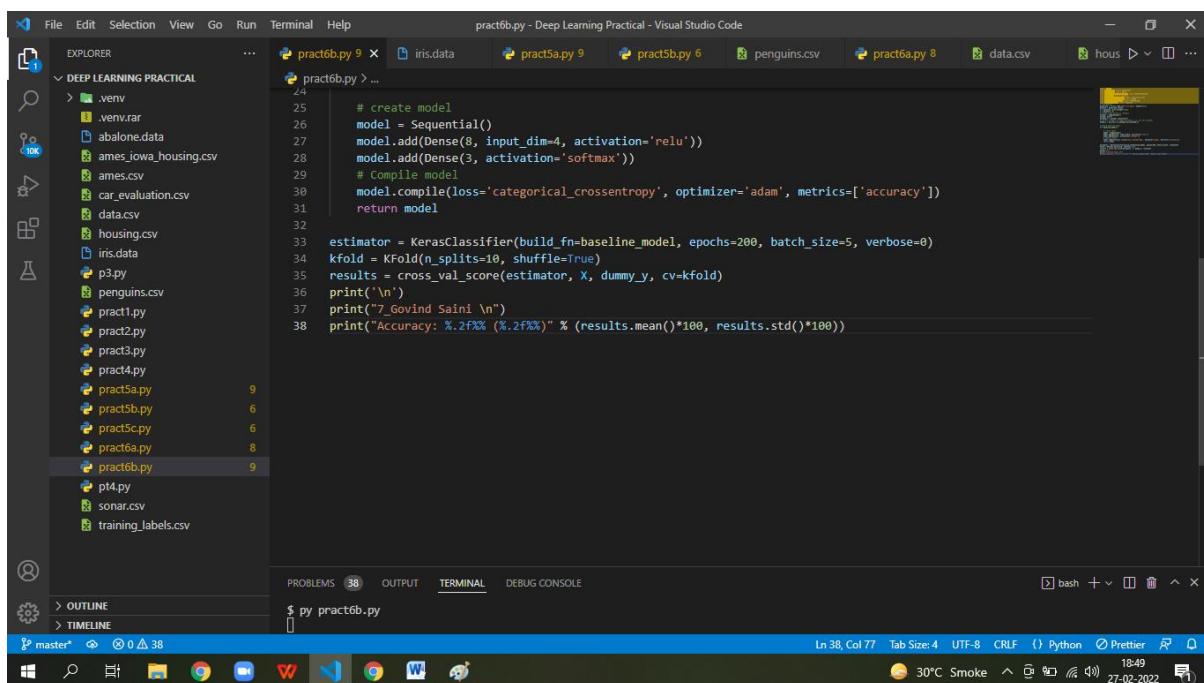
**Code :**



```

1 import pandas
2 from keras.models import Sequential
3 from keras.layers import Dense
4 from keras.wrappers.scikit_learn import KerasClassifier
5 from keras.utils import np_utils
6 from sklearn.model_selection import cross_val_score
7 from sklearn.model_selection import KFold
8 from sklearn.preprocessing import LabelEncoder
9 from sklearn.pipeline import Pipeline
10 # load dataset
11 dataframe = pandas.read_csv("iris.data", header=None)
12 dataset = dataframe.values
13 X = dataset[:,0:4].astype(float)
14 Y = dataset[:,4]
15 # encode class values as integers
16 encoder = LabelEncoder()
17 encoder.fit(Y)
18 encoded_Y = encoder.transform(Y)
19 # convert integers to dummy variables (i.e. one hot encoded)
20 dummy_y = np_utils.to_categorical(encoded_Y)
21
22 # define baseline model
23 def baseline_model():
24     # create model
25     model = Sequential()
26     model.add(Dense(8, input_dim=4, activation='relu'))
27     model.add(Dense(3, activation='softmax'))
28     # Compile model
29     model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
30     return model
31
32
33 estimator = KerasClassifier(build_fn=baseline_model, epochs=200, batch_size=5, verbose=0)
34 kfold = KFold(n_splits=10, shuffle=True)
35 results = cross_val_score(estimator, X, dummy_y, cv=kfold)
36 print('\n')
37 print("Govind Saini \n")
38 print("Accuracy: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))

```

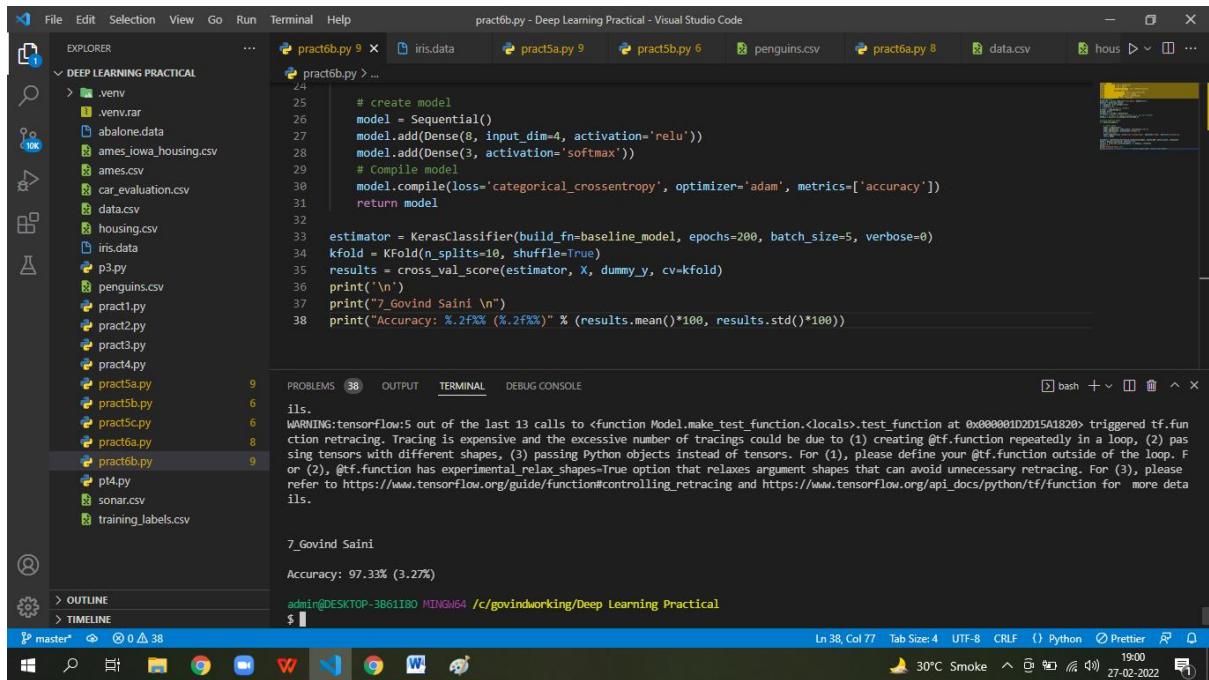


```

$ py pract6b.py

```

## Output:



```

File Edit Selection View Go Run Terminal Help
pract6b.py - Deep Learning Practical - Visual Studio Code

EXPLORER
DEEP LEARNING PRACTICAL
.venv
.venv.rar
abalone.data
ames_iowa_housing.csv
ames.csv
car_evaluation.csv
data.csv
housing.csv
iris.data
p3.py
penguins.csv
pract1.py
pract2.py
pract3.py
pract4.py
pract5.py
pract5b.py
pract5c.py
pract6.py
pract6b.py
pract7.py
p4.py
sonar.csv
training_labels.csv

pract6b.py > ...
24
25     # Create model
26     model = Sequential()
27     model.add(Dense(8, input_dim=4, activation='relu'))
28     model.add(Dense(3, activation='softmax'))
29     # Compile model
30     model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
31     return model
32
33 estimator = KerasClassifier(build_fn=baseline_model, epochs=200, batch_size=5, verbose=0)
34 kfold = KFold(n_splits=10, shuffle=True)
35 results = cross_val_score(estimator, X, dummy_y, cv=kfold)
36 print('\n')
37 print("7_Govind Saini \n")
38 print("Accuracy: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))

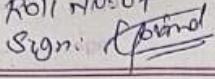
PROBLEMS 38 OUTPUT TERMINAL DEBUG CONSOLE
bash + v ^ x
ils.
WARNING:tensorflow:5 out of the last 13 calls to <function Model.make_test_function.<locals>.test_function at 0x000001D2015A1820> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

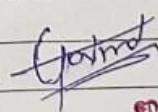
7_Govind Saini
Accuracy: 97.33% (3.27%)

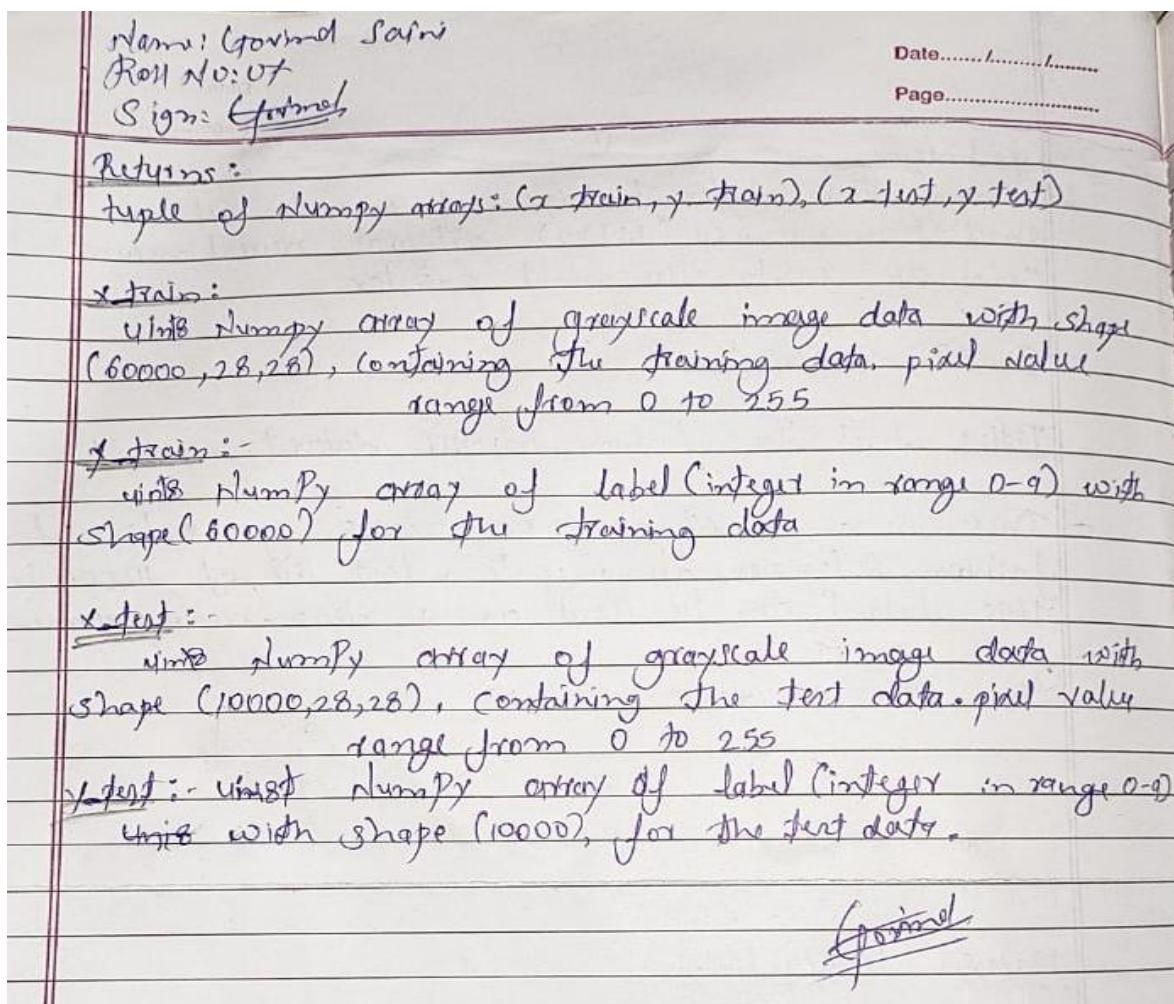
admin@DESKTOP-3B61180 MINGW64 /c/govindworking/Deep Learning Practical
$ |
```

Ln 38, Col 77 Tab Size: 4 UTF-8 CRLF ⚡ Python ⚡ Prettier ⚡ 30°C Smoke ⚡ 19:00 27-02-2022

# Practical No : 7

Name: Govind Saini Roll No.: 07 Sign: 	Date: 9.1.3.2022 Page.....
<u>Practical - 7</u>	
<b>Aim:-</b> Implementation of convolution neural network to predict number from number images.	
<b>Description:-</b> <ul style="list-style-type: none"> <li>- Convolution neural networks (CNN) - the concept behind recent breakthrough and developments in deep learning</li> <li>- CNN have broken the mold and ascended the throne to become the state-of-the-art computer vision technique.</li> <li>- Among the different types of neural network long short term memory (LSTM), artificial neural network (ANN) etc.. CNN are easily most popular.</li> </ul>	
<b>Dataset Description:-</b> <u>Titanic</u> mini mnist dataset	
This is a dataset of 60,000 28x28 grayscale images of the 10 digits, along with a test set of 10,000 images	
Args path	path where to cache the dataset locally (relative to <code>~/.keras/datasets</code> ).

  
**enrichies**



## Code :

```

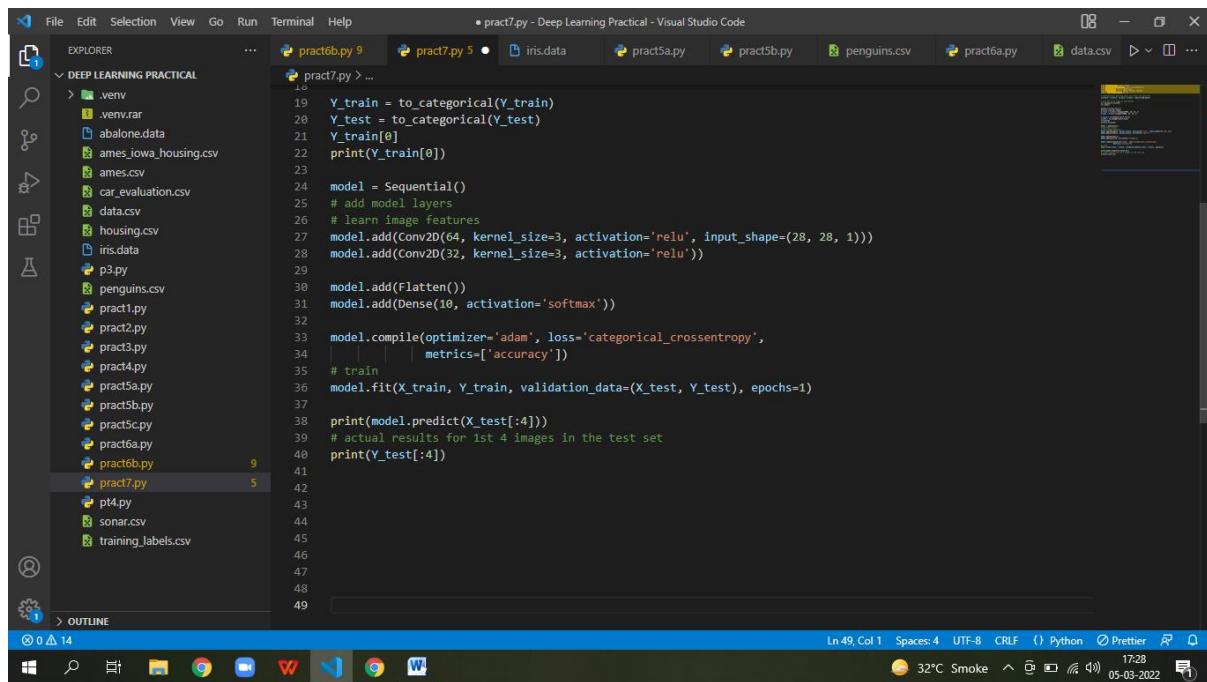
File Edit Selection View Go Run Terminal Help
pract7.py - Deep Learning Practical - Visual Studio Code
EXPLORER
DEEP LEARNING PRACTICAL
> .venv
  .venv.rar
  abalone.data
  ames_iowa_housing.csv
  ames.csv
  car_evaluation.csv
  data.csv
  housing.csv
  iris.data
  p3.py
  penguins.csv
  pract1.py
  pract2.py
  pract3.py
  pract4.py
  pract5a.py
  pract5b.py
  pract5c.py
  pract6a.py
  pract6b.py 9
  pract7.py 5
  iris.data
  pract5a.py
  pract5b.py
  pract5c.py
  pract6a.py
  pract6b.py
  pract7.py
  pt4.py
  sonar.csv
  training_labels.csv
pract7.py > OUTLINE
Ln 1, Col 33 Spaces: 4 UTF-8 CRLF Python Prettier
32° C Smoke 17:27 05-03-2022

```

```

1  from keras.datasets import mnist
2  from tensorflow.keras.utils import to_categorical
3  from keras.models import Sequential
4  from keras.layers import Dense, Conv2D, Flatten
5  import matplotlib.pyplot as plt
6
7  # download mnist data and split into train and test sets
8  (X_train, Y_train), (X_test, Y_test) = mnist.load_data()
9
10 # plot the first image in the dataset
11 plt.imshow(X_train[0])
12 plt.show()
13
14 print("7_Govind Saini")
15 print(X_train[0].shape)
16 X_train = X_train.reshape(60000, 28, 28, 1)
17 X_test = X_test.reshape(10000, 28, 28, 1)
18
19 Y_train = to_categorical(Y_train)
20 Y_test = to_categorical(Y_test)
21 Y_train[0]
22 print(Y_train[0])
23
24 model = Sequential()
25 # add model layers
26 # learn image features
27 model.add(Conv2D(64, kernel_size=3, activation='relu', input_shape=(28, 28, 1)))
28 model.add(Conv2D(32, kernel_size=3, activation='relu'))
29
30 model.add(Flatten())
31 model.add(Dense(10, activation='softmax'))
32
model.compile(optimizer='adam', loss='categorical_crossentropy')

```



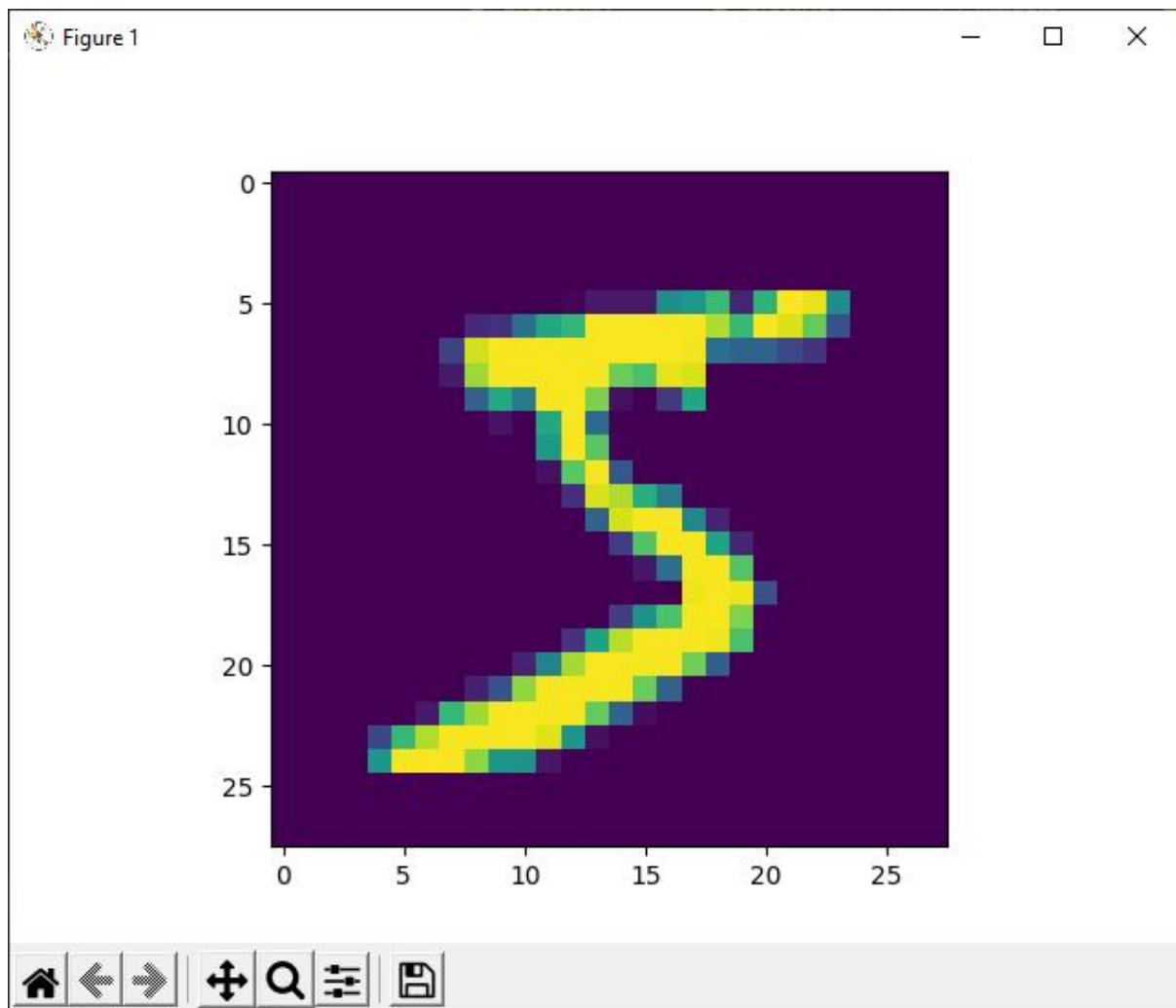
```

File Edit Selection View Go Run Terminal Help
pract6b.py 9 pract7.py 5 iris.data pract5a.py pract5b.py penguins.csv pract6a.py data.csv ...
EXPLORER DEEP LEARNING PRACTICAL
.pract7.py > ...
19     Y_train = to_categorical(Y_train)
20     Y_test = to_categorical(Y_test)
21     Y_train[0]
22     print(Y_train[0])
23
24     model = Sequential()
25     # add model layers
26     # learn image features
27     model.add(Conv2D(64, kernel_size=3, activation='relu', input_shape=(28, 28, 1)))
28     model.add(Conv2D(32, kernel_size=3, activation='relu'))
29
30     model.add(Flatten())
31     model.add(Dense(10, activation='softmax'))
32
33     model.compile(optimizer='adam', loss='categorical_crossentropy',
34                   metrics=['accuracy'])
35
36     # train
37     model.fit(X_train, Y_train, validation_data=(X_test, Y_test), epochs=1)
38
39     print(model.predict(X_test[:4]))
40     # actual results for 1st 4 images in the test set
41     print(Y_test[:4])
42
43
44
45
46
47
48
49

```

Ln 49, Col 1 Spaces: 4 UTF-8 CRLF Python Prettier 32°C Smoke 17:28 05-03-2022

## Output :



The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows a project structure with files like `.venv`, `.venv.rar`, `abalone.data`, `ames_iowa_housing.csv`, `ames.csv`, `car_evaluation.csv`, `data.csv`, `housing.csv`, `iris.data`, `p3.py`, `penguins.csv`, `pract1.py`, `pract2.py`, `pract3.py`, `pract4.py`, `pract5a.py`, `pract5b.py`, `pract5c.py`, `pract6a.py`, `pract6b.py` (highlighted), `pract7.py` (highlighted), `pt4.py`, `sonar.csv`, and `training_labels.csv`.
- Terminal (Top):** The title bar says "pract7.py - Deep Learning Practical - Visual Studio Code".
- Code Editor (Center):** The code for `pract7.py` is displayed, showing imports from `keras` and `keras.layers`, and the creation of a Sequential model with two Conv2D layers.
- Output (Bottom):** The terminal output shows the execution of the script and the resulting output of the printed arrays.
- Status Bar (Bottom):** Shows the current line (Ln 34, Col 36), spaces used (Spaces: 4), and file type (Python). It also displays system information like CPU temperature (32°C), battery level (Smoke), and the date (05-03-2022).

## Practical No : 8

Name: Govind Saini  
Roll No. of  
Signature:

Date: 11.03.2022

Page.....

### Practical - 8

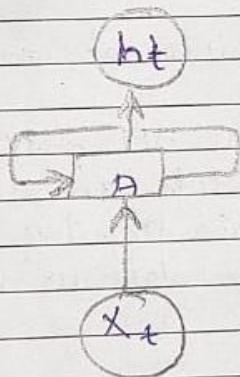
#### 8) Aim:-

Demonstrate recurrent neural network that learns to perform sequence analysis for stock price.

#### Description:-

Recurrent neural networks implement the same concept using machine; they have loop and allows information to persist where feedforward neural network can't.

Let's us a few illustration to demonstrate how a RNN work



when  $x_t$  takes the input  $x_t$ , then  $h_t$  will be the output.

A Rnn is like multiple copies of the same network that passes the message to a successor.

now let's think a little bit about what happens if we unroll previous loop

*Govind*

Dynamilis

Name: Govind Saini  
 Roll No: 07  
 Sign: Govind

Date..... / ..... / .....  
 Page.....

```

graph LR
    A[ ] --> h0((h_0))
    A --> h1((h_1))
    A --> h2((h_2))
    A --> ht((h_t))
    A --> At[ ]
    h0 --> h1
    h1 --> h2
    h2 --> ht
    ht --> At
    At --> xt((x_t))
    xt --> h0
  
```

Here comes the vanishing gradient problem of the RNN, where it can not handle large sequence. Long Short-term memory (LSTM) are designed to handle long-term dependence.

library are Used :-

nstepy :-

It is a library to extract historical and realtime data from nse's website.  
 This library aims to keep the API very simple python is a great tool for data analysis along with the `numpy` stack and the main objective of `nstepy` is to provide analysis ready data-arrays for use with `numpy` stack.

LSTM Recurrent Neural Network :

- Long-short-Term memory RNN belong to the family of deep learning algorithms  
 It is recurrent network because of the feedback connections in its architecture.

Ciprind

Name: Govind Saini  
Room No: 07  
Sign: Govind

Date...../...../.....  
Page.....

Stock prediction :

- In this task, the future stock price of State Bank of India (SBI) are predicted using the LSTM Recurrent Neural Network.
- Our task is to predict stock price for a few days, which is a time series problem.
- LSTM model is very popular in time-series forecasting and this is the reason why this model is chosen in this task.

Dataset Description:

- This dataset contain 1483 observations with 12 attribute. After preprocessing only date and OHLC (Open, High, Low, Close) columns, a total of 5 columns, are taken as these column have main significance in dataset.
- LSTM model is trained on this entire dataset, and for the testing purpose, a new dataset is fetched for the duration of between 01.01.2019 to 18.01.2019.
- Stock price for new duration will predicted by already trained LSTM model, and predicted price will be plotted against the original price to visualize the model accuracy.

Govind

**Code :**

```

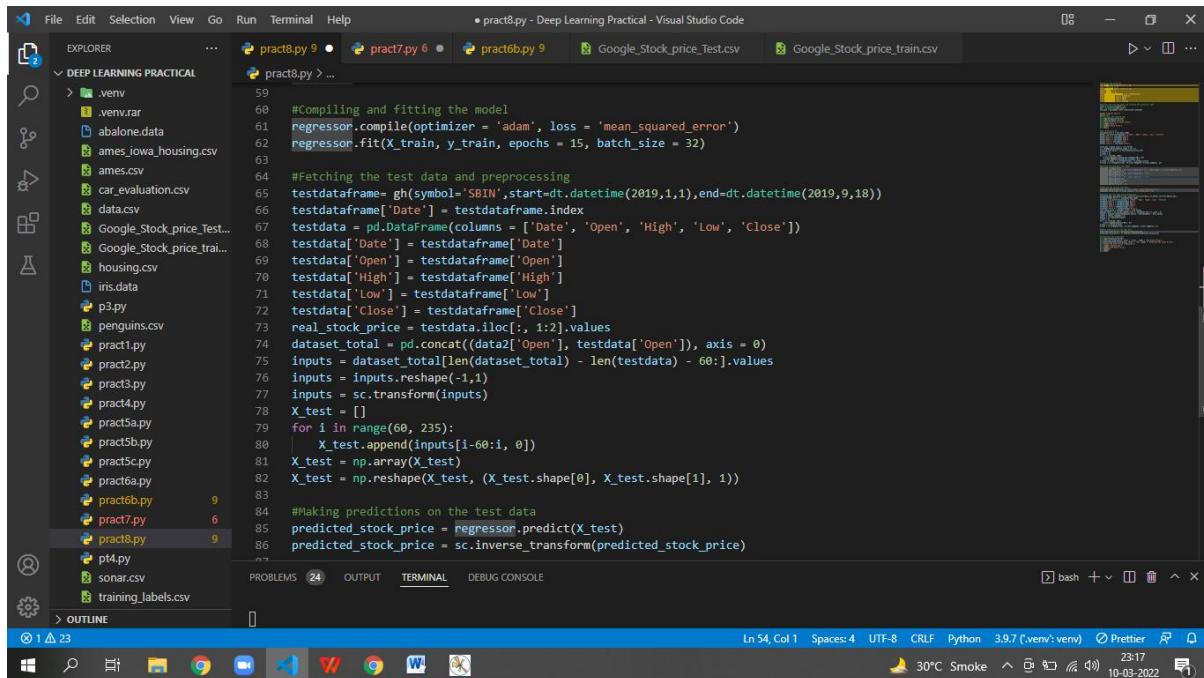
1 #Importing the libraries
2 from nsepy import get_history as gh
3 import datetime as dt
4 from matplotlib import pyplot as plt
5 import numpy as np
6 import pandas as pd
7 from sklearn.preprocessing import MinMaxScaler
8 from keras.models import Sequential
9 from keras.layers import Dense
10 from keras.layers import LSTM
11 from keras.layers import Dropout
12
13 #Setting start and end dates and fetching the historical data
14 start = dt.datetime(2013,1,1)
15 end = dt.datetime(2018,12,31)
16 stk_data = gh(symbol='SBIN',start=start,end=end)
17
18 print("Govind Saini")
19 print("\n")
20 #visualizing the fetched data
21 plt.figure(figsize=(14,14))
22 plt.plot(stk_data['Close'])
23 plt.title('Historical Stock Value')
24 plt.xlabel('Date')
25 plt.ylabel('Stock Price')
26 plt.show()
27
28 #Data Preprocessing
29 #Data Preprocessing
30 #Data Preprocessing
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59

```

```

27
28 #Data Preprocessing
29 stk_data['Date'] = stk_data.index
30 data2 = pd.DataFrame(columns = ['Date', 'Open', 'High', 'Low', 'Close'])
31 data2['Date'] = stk_data['Date']
32 data2['Open'] = stk_data['Open']
33 data2['High'] = stk_data['High']
34 data2['Low'] = stk_data['Low']
35 data2['Close'] = stk_data['Close']
36
37 train_set = data2.iloc[:, 1:2].values
38 sc = MinMaxScaler(feature_range = (0, 1))
39 training_set_scaled = sc.fit_transform(train_set)
40 X_train = []
41 y_train = []
42 for i in range(60, 1482):
43     X_train.append(training_set_scaled[i-60:i, 0])
44     y_train.append(training_set_scaled[i, 0])
45 X_train, y_train = np.array(X_train), np.array(y_train)
46 X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
47
48 #Defining the LSTM Recurrent Model
49 regressor = Sequential()
50 regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.shape[1], 1)))
51 regressor.add(Dropout(0.2))
52 regressor.add(LSTM(units = 50, return_sequences = True))
53 regressor.add(Dropout(0.2))
54 regressor.add(LSTM(units = 50, return_sequences = True))
55 regressor.add(Dropout(0.2))
56 regressor.add(LSTM(units = 50))
57 regressor.add(Dropout(0.2))
58 regressor.add(Dense(units = 1))
59

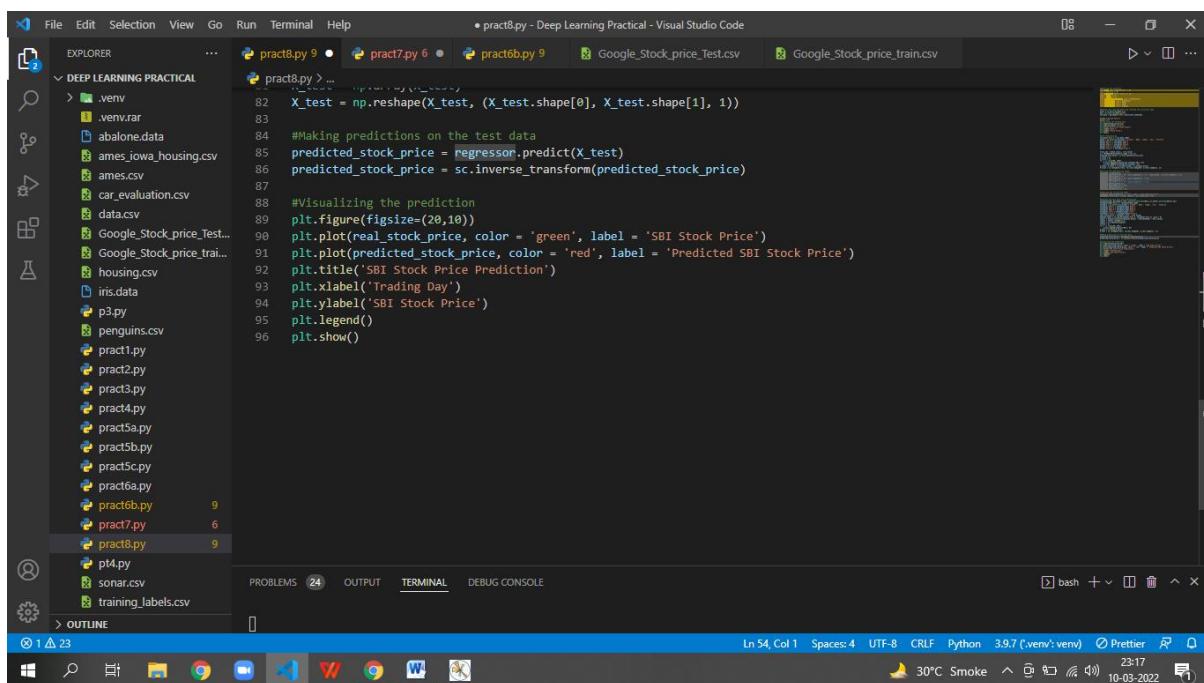
```



```

59 #Compiling and fitting the model
60 regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')
61 regressor.fit(X_train, y_train, epochs = 15, batch_size = 32)
62
63 #Fetching the test data and preprocessing
64 testdataframe= gh(symbol='SBIN',start=dt.datetime(2019,1,1),end=dt.datetime(2019,9,18))
65 testdataframe['Date']= testdataframe.index
66 testdata = pd.DataFrame(columns = ['Date', 'Open', 'High', 'Low', 'Close'])
67 testdata['Date'] = testdataframe['Date']
68 testdata['Open'] = testdataframe['Open']
69 testdata['High'] = testdataframe['High']
70 testdata['Low'] = testdataframe['Low']
71 testdata['Close'] = testdataframe['Close']
72
73 real_stock_price = testdata.iloc[:, 1:2].values
74 dataset_total = pd.concat([data2['Open'], testdata['Open']], axis = 0)
75 inputs = dataset_total[len(dataset_total) - len(testdata) - 60: ].values
76 inputs = inputs.reshape(-1,1)
77 inputs = sc.transform(inputs)
78 X_test = []
79 for i in range(60, 235):
80     X_test.append(inputs[i-60:i, 0])
81 X_test = np.array(X_test)
82 X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
83
84 #Making predictions on the test data
85 predicted_stock_price = regressor.predict(X_test)
86 predicted_stock_price = sc.inverse_transform(predicted_stock_price)

```

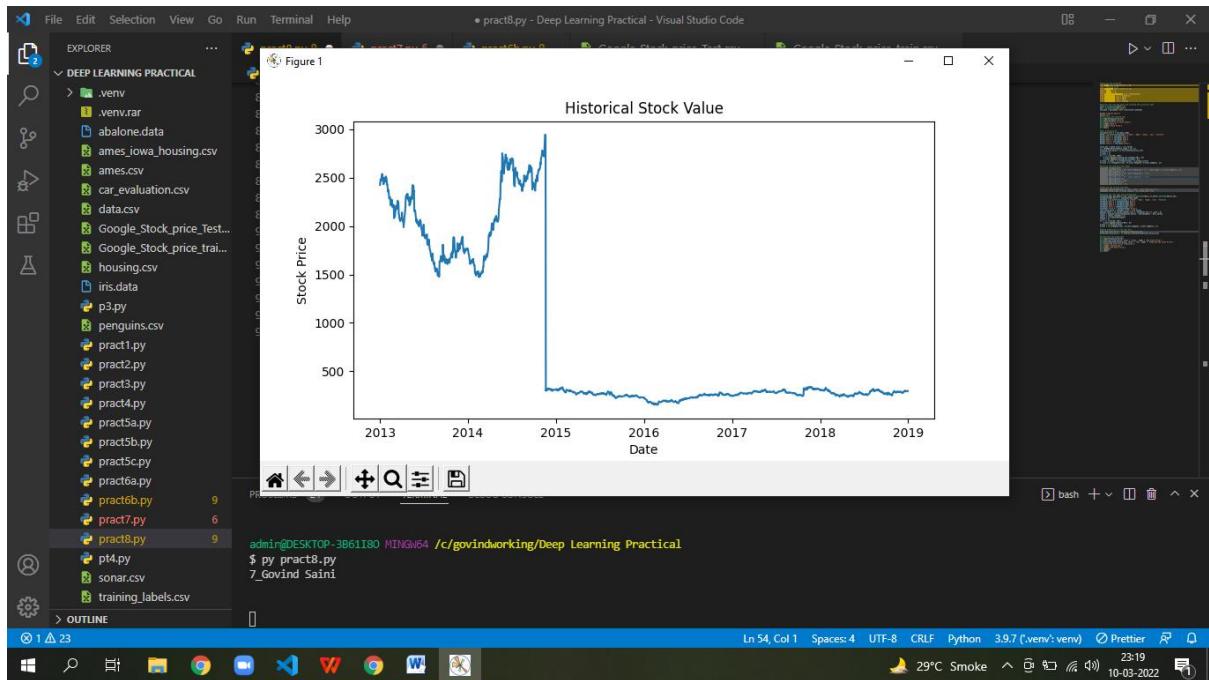


```

87
88 #Visualizing the prediction
89 plt.figure(figsize=(20,10))
90 plt.plot(real_stock_price, color = 'green', label = 'SBI Stock Price')
91 plt.plot(predicted_stock_price, color = 'red', label = 'Predicted SBI Stock Price')
92 plt.title('SBI Stock Price Prediction')
93 plt.xlabel('Trading Day')
94 plt.ylabel('SBI Stock Price')
95 plt.legend()
96 plt.show()

```

## Output :

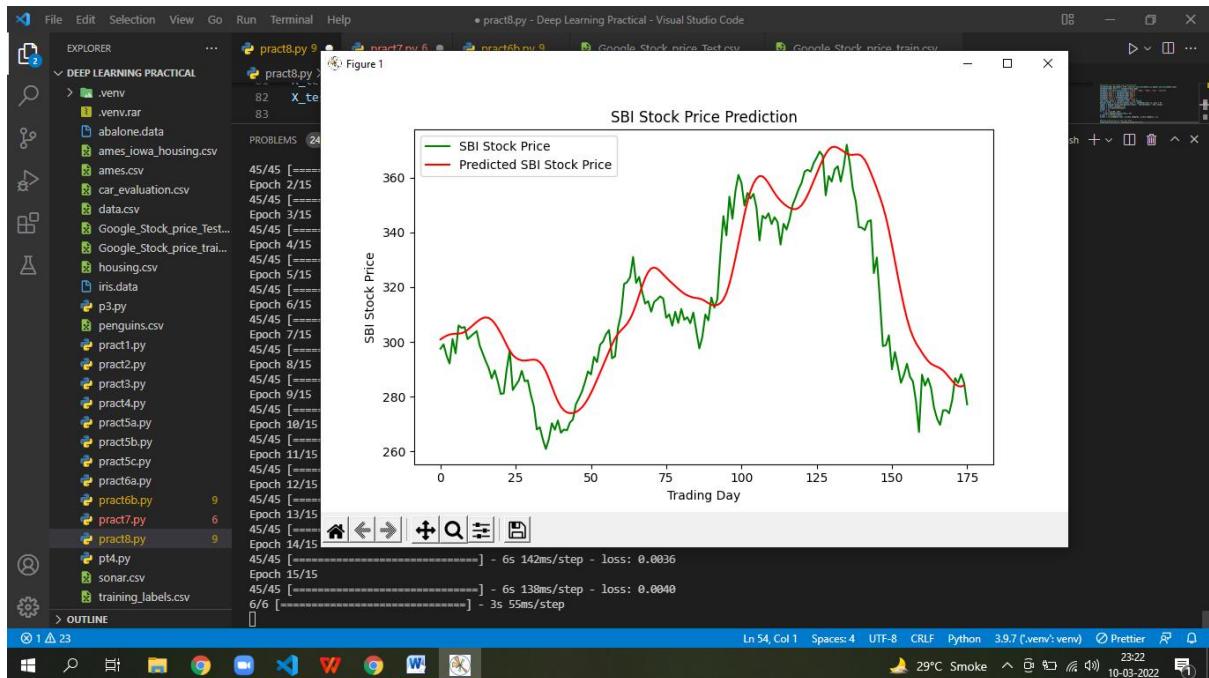


The screenshot shows a Visual Studio Code interface with the 'TERMINAL' tab selected. It displays the output of a Python script named 'pract8.py'. The output shows a series of epochs from 1 to 15, with each epoch taking approximately 1 second and resulting in a loss value between 0.0016 and 0.0048. The status bar at the bottom indicates the file is 3.9.7 ('venv'; venv) and the date is 10-03-2022.

```

pract8.py 9 ● pract7.py 6 ● pract6.py 9 Google_Stock_price_Test.csv Google_Stock_price_train.csv
=====
45/45 [=====] - 19s 138ms/step - loss: 0.0169
Epoch 2/15
45/45 [=====] - 6s 142ms/step - loss: 0.0079
Epoch 3/15
45/45 [=====] - 7s 148ms/step - loss: 0.0075
Epoch 4/15
45/45 [=====] - 6s 138ms/step - loss: 0.0067
Epoch 5/15
45/45 [=====] - 7s 145ms/step - loss: 0.0065
Epoch 6/15
45/45 [=====] - 7s 160ms/step - loss: 0.0060
Epoch 7/15
45/45 [=====] - 6s 143ms/step - loss: 0.0054
Epoch 8/15
45/45 [=====] - 6s 144ms/step - loss: 0.0061
Epoch 9/15
45/45 [=====] - 6s 134ms/step - loss: 0.0049
Epoch 10/15
45/45 [=====] - 6s 132ms/step - loss: 0.0043
Epoch 11/15
45/45 [=====] - 6s 134ms/step - loss: 0.0048
Epoch 12/15
45/45 [=====] - 6s 141ms/step - loss: 0.0052
Epoch 13/15
45/45 [=====] - 6s 142ms/step - loss: 0.0048
Epoch 14/15
45/45 [=====] - 6s 142ms/step - loss: 0.0036
Epoch 15/15
45/45 [=====] - 6s 138ms/step - loss: 0.0040
6/6 [=====] - 3s 59ms/step

```



## Practical No : 9

Name: Govind Saini  
 Roll No: 07  
 Sign:

Date: 19.3.2022  
 Page: \_\_\_\_\_

Practical - 9

Q1) Aim:-

Performing encoding and decoding of images using deep autoencoder.

Description:-

The diagram illustrates a deep autoencoder architecture. It starts with an 'Original Input' image of the digit '2'. This image is processed by an 'Encoder' to produce a 'Compressed representation'. This compressed representation is then passed through a 'Decoder' to reconstruct the original image '2' as the 'Reconstructed Input'.

i) Autoencoding is a data compression algorithm where the compression and decompression function is data-specific.  
 ii) is lossy  
 iii) Learned automatically from example rather than engineered by a human.

Autoencoders are data-specific, which means that they will only be able to compress data similar to what they have been trained on.  
 This is different from say MPEG-2 audio layer III (mp3) compression algorithm, which hold assumption about sound in general, but not about specific types of sound.

#Orchies

Name: Govind Saini  
 Roll No: 07  
 Sign: Govind

Date.....

Page.....

Autoencoders are lossy, which means that the compressed outputs will be degraded compared to the original input (similar to MP3 or JPEG compression). This differs from lesser arithmetic operators.

Autoencoders are learned automatically from data examples, which is useful property. It means that it is easy to train specialized instance of algorithm that will perform well on a specific type of input. It doesn't require any new engineering, just appropriate training data.

### Dataset Description:-

The MNIST dataset contains 70,000 images of handwritten digits (zero to nine) that have been size-normalized and centered in a square grid of pixels.

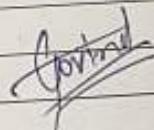
Each image is a  $28 \times 28 \times 1$  array of floating-point numbers representing grayscale intensities ranging from 0 (black) to 1 (white).

The target data consist of one-hot binary vector of size 10, corresponding to the digit classification categories zero through nine. Some example MNIST images are shown below:

Information: \* name : MNIST \* length : 70000

Govind

Name: Govind Saini Roll No: 07 Sign: Govind	Date...../...../..... Page.....
<b>Input Summary:</b> * shape : (28, 28, 1) * range : (0.0, 1.0)	
<b>Target Summary:</b> * shape : (1, 10) * range : (0.0, 1.0)	
<u>Result Descriptions:-</u>	
<p>Here's what we get, the top row is the original digit and the bottom row is the reconstructed digits. we are losing quite a bit of detail with the basic approach.</p> <p>Let's train the model for 10 epochs with the added regularization (the model is less likely to overfit and can be trained longer)</p> <p>The models end with a train loss of 0.11 and test loss of 0.10.</p> <p>The difference between the two is mostly due to the regularization term being added to the loss during training.</p> <p>Here's a visualization of our results.</p>	



**Code :**

```

File Edit Selection View Go Run Terminal Help
pract9.py - Deep Learning Practical - Visual Studio Code
EXPLORER DEEP LEARNING PRACTICAL
.pract9.py
pract10.py
pract9.py > ...
1 import keras
2 from keras import layers
3 from keras.datasets import mnist
4 import numpy as np
5 import matplotlib.pyplot as plt
6 # This is the size of our encoded representations
7 # 32 floats -> compression of factor 24.5, assuming the input is 784 floats
8 encoding_dim = 32
9
10 # This is our input image
11 input_img = keras.Input(shape=(784,))
12 # "encoded" is the encoded representation of the input
13 encoded = layers.Dense(encoding_dim, activation='relu')(input_img)
14 # "decoded" is the lossy reconstruction of the input
15 decoded = layers.Dense(784, activation='sigmoid')(encoded)
16
17 # This model maps an input to its reconstruction
18 autoencoder = keras.Model(input_img, decoded)
19
20 # This model maps an input to its encoded representation
21 encoder = keras.Model(input_img, encoded)
22 # This is our encoded (32-dimensional) input
23 encoded_input = keras.Input(shape=(encoding_dim,))
24 # Retrieve the last layer of the autoencoder model
25 decoder_layer = autoencoder.layers[-1]
26 # Create the decoder model
27 decoder = keras.Model(encoded_input, decoder_layer(encoded_input))
28
29 autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
30
31
32 (x_train, _), (x_test, _) = mnist.load_data()
33 x_train = x_train.astype('float32') / 255.
34 x_test = x_test.astype('float32') / 255.
35 x_train = x_train.reshape((len(x_train), np.prod(x_train.shape[1:])))
36 x_test = x_test.reshape((len(x_test), np.prod(x_test.shape[1:])))
37 print("7_GovindSaini \n")
38 print(x_train.shape)
39 print(x_test.shape)
40
41 autoencoder.fit(x_train, x_train,
42                 epochs=13,
43                 batch_size=256,
44                 shuffle=True, validation_data=(x_test, x_test))
45
46 # Encode and decode some digits. Note that we take them from the *test* set
47 encoded_imgs = encoder.predict(x_test)
48 decoded_imgs = decoder.predict(encoded_imgs)
49
50 # Use Matplotlib (don't ask)
51 n = 10 # How many digits we will display
52 plt.figure(figsize=(20, 4))
53 for i in range(n):
54     # Display original
55     ax = plt.subplot(2, n, i + 1)
56     plt.imshow(x_test[i].reshape(28, 28))
57     plt.gray()
58     ax.get_xaxis().set_visible(False)
59     ax.get_yaxis().set_visible(False)

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Ln 22, Col 45 Spaces: 4 UTF-8 CRLF Python 3.9.7 64-bit ⚡ Prettier 15:11 20-03-2022

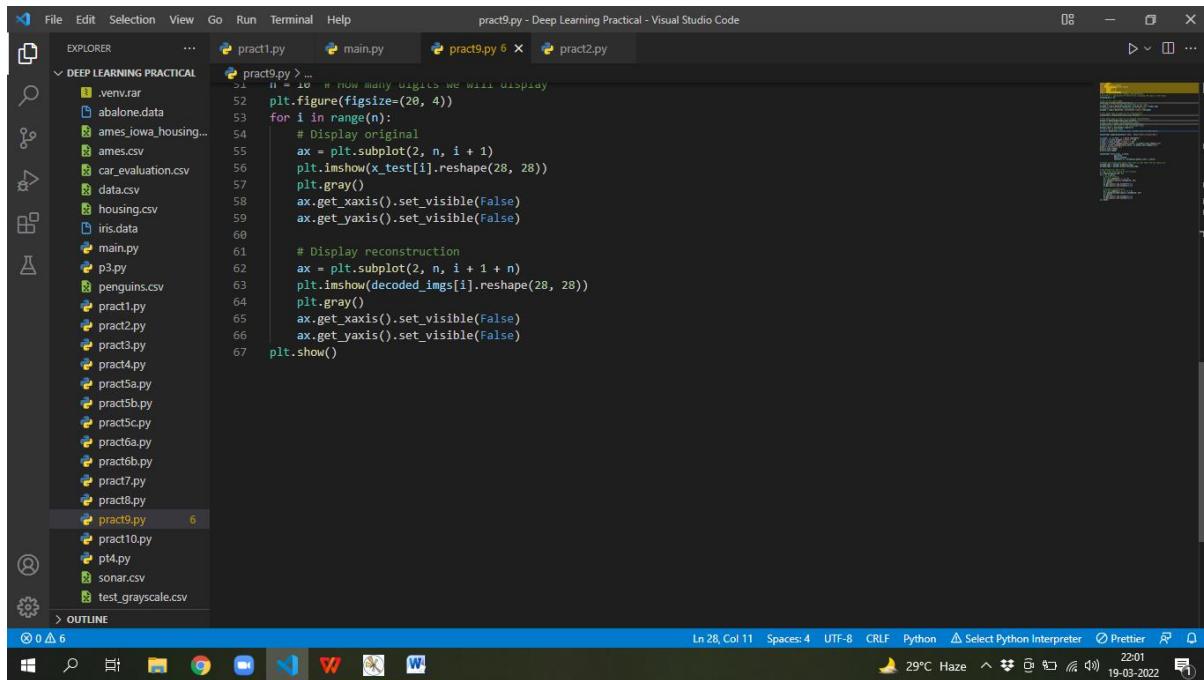
```

File Edit Selection View Go Run Terminal Help
pract9.py - Deep Learning Practical - Visual Studio Code
EXPLORER DEEP LEARNING PRACTICAL
pract1.py
main.py
pract9.py 6
pract2.py
pract9.py > ...
28 decoder = keras.Model(encoded_input, decoder_layer(encoded_input))
29
30 autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
31
32 (x_train, _), (x_test, _) = mnist.load_data()
33 x_train = x_train.astype('float32') / 255.
34 x_test = x_test.astype('float32') / 255.
35 x_train = x_train.reshape((len(x_train), np.prod(x_train.shape[1:])))
36 x_test = x_test.reshape((len(x_test), np.prod(x_test.shape[1:])))
37 print("7_GovindSaini \n")
38 print(x_train.shape)
39 print(x_test.shape)
40
41 autoencoder.fit(x_train, x_train,
42                 epochs=13,
43                 batch_size=256,
44                 shuffle=True, validation_data=(x_test, x_test))
45
46 # Encode and decode some digits. Note that we take them from the *test* set
47 encoded_imgs = encoder.predict(x_test)
48 decoded_imgs = decoder.predict(encoded_imgs)
49
50 # Use Matplotlib (don't ask)
51 n = 10 # How many digits we will display
52 plt.figure(figsize=(20, 4))
53 for i in range(n):
54     # Display original
55     ax = plt.subplot(2, n, i + 1)
56     plt.imshow(x_test[i].reshape(28, 28))
57     plt.gray()
58     ax.get_xaxis().set_visible(False)
59     ax.get_yaxis().set_visible(False)

```

OUTLINE

Ln 28, Col 11 Spaces: 4 UTF-8 CRLF Python ⚡ Select Python Interpreter 22:01 29°C Haze 19-03-2022



```

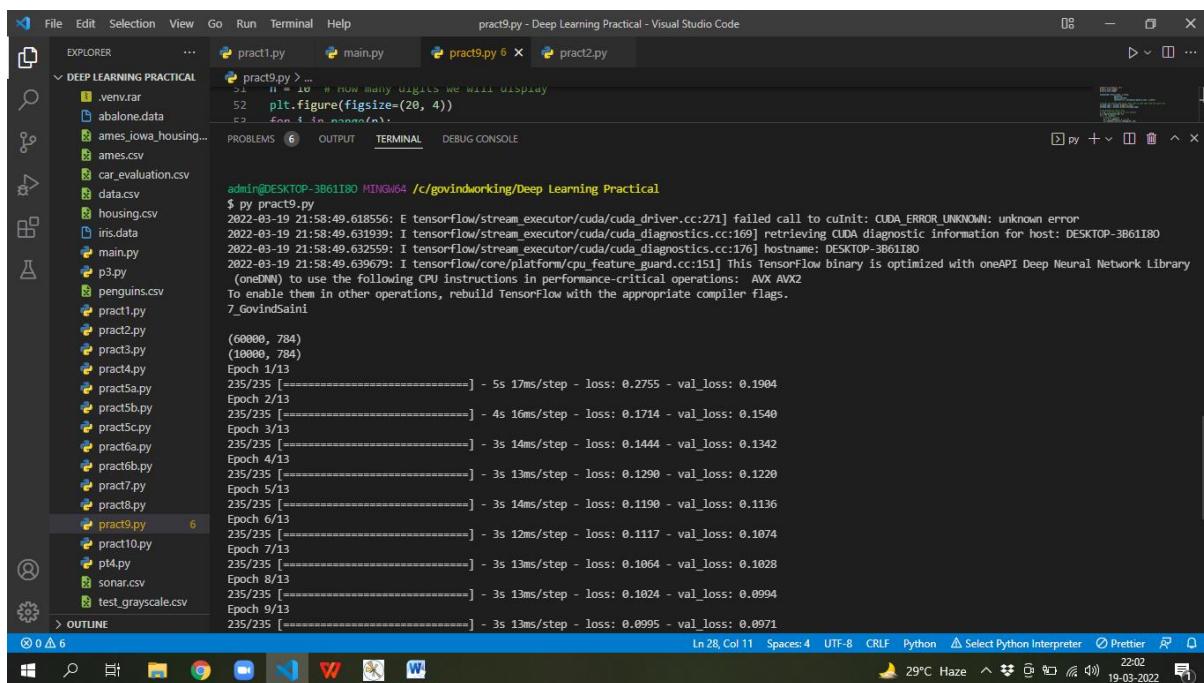
File Edit Selection View Go Run Terminal Help
pract1.py main.py pract9.py 6 pract2.py

EXPLORER DEEP LEARNING PRACTICAL
.venv.rar
abalone.data
ames_iowa_housing...
ames.csv
car_evaluation.csv
data.csv
housing.csv
iris.data
main.py
p3.py
penguins.csv
pract1.py
pract2.py
pract3.py
pract4.py
pract5.py
pract5b.py
pract5c.py
pract6a.py
pract6b.py
pract7.py
pract8.py
pract9.py 6
pract10.py
pt4.py
sonar.csv
test_grayscale.csv

Ln 28, Col 11 Spaces: 4 UTF-8 CRLF Python Select Python Interpreter Prettier
22:01 29°C Haze 19-03-2022

```

## Ouput :



```

File Edit Selection View Go Run Terminal Help
pract1.py main.py pract9.py 6 pract2.py

EXPLORER DEEP LEARNING PRACTICAL
.venv.rar
abalone.data
ames_iowa_housing...
ames.csv
car_evaluation.csv
data.csv
housing.csv
iris.data
main.py
p3.py
penguins.csv
pract1.py
pract2.py
pract3.py
pract4.py
pract5.py
pract5b.py
pract5c.py
pract6a.py
pract6b.py
pract7.py
pract8.py
pract9.py 6
pract10.py
pt4.py
sonar.csv
test_grayscale.csv

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
py + x ^ x ^ x

admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/Deep Learning Practical
$ py pract9.py
2022-03-19 21:58:49.618556: E tensorflow/stream_executor/cuda/cuda_driver.cc:271] Failed call to cuInit: CUDA_ERROR_UNKNOWN: unknown error
2022-03-19 21:58:49.631999: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: DESKTOP-3B61I80
2022-03-19 21:58:49.632559: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:170] hostname: DESKTOP-3B61I80
2022-03-19 21:58:49.639679: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
7_GovindSaini

(60000, 784)
(10000, 784)
Epoch 1/13
235/235 [=====] - 5s 17ms/step - loss: 0.2755 - val_loss: 0.1904
Epoch 2/13
235/235 [=====] - 4s 16ms/step - loss: 0.1714 - val_loss: 0.1540
Epoch 3/13
235/235 [=====] - 3s 14ms/step - loss: 0.1444 - val_loss: 0.1342
Epoch 4/13
235/235 [=====] - 3s 13ms/step - loss: 0.1290 - val_loss: 0.1220
Epoch 5/13
235/235 [=====] - 3s 14ms/step - loss: 0.1190 - val_loss: 0.1136
Epoch 6/13
235/235 [=====] - 3s 12ms/step - loss: 0.1117 - val_loss: 0.1074
Epoch 7/13
235/235 [=====] - 3s 13ms/step - loss: 0.1064 - val_loss: 0.1028
Epoch 8/13
235/235 [=====] - 3s 13ms/step - loss: 0.1024 - val_loss: 0.0994
Epoch 9/13
235/235 [=====] - 3s 13ms/step - loss: 0.0995 - val_loss: 0.0971
Ln 28, Col 11 Spaces: 4 UTF-8 CRLF Python Select Python Interpreter Prettier
22:02 29°C Haze 19-03-2022

```

```

pract9.py > ...
51 n = 10 # HOW MANY DIGITS WE WILL DISPLAY
52 plt.figure(figsize=(20, 4))
53 for i in range(n):
235/235 [=====] - 5s 17ms/step - loss: 0.2755 - val_loss: 0.1904
Epoch 2/13
235/235 [=====] - 4s 16ms/step - loss: 0.1714 - val_loss: 0.1540
Epoch 3/13
235/235 [=====] - 3s 14ms/step - loss: 0.1444 - val_loss: 0.1342
Epoch 4/13
235/235 [=====] - 3s 13ms/step - loss: 0.1290 - val_loss: 0.1220
Epoch 5/13
235/235 [=====] - 3s 14ms/step - loss: 0.1190 - val_loss: 0.1136
Epoch 6/13
235/235 [=====] - 3s 12ms/step - loss: 0.1117 - val_loss: 0.1074
Epoch 7/13
235/235 [=====] - 3s 13ms/step - loss: 0.1064 - val_loss: 0.1028
Epoch 8/13
235/235 [=====] - 3s 13ms/step - loss: 0.1024 - val_loss: 0.0994
Epoch 9/13
235/235 [=====] - 3s 13ms/step - loss: 0.0995 - val_loss: 0.0971
Epoch 10/13
235/235 [=====] - 3s 13ms/step - loss: 0.0975 - val_loss: 0.0956
Epoch 11/13
235/235 [=====] - 3s 13ms/step - loss: 0.0962 - val_loss: 0.0944
Epoch 12/13
235/235 [=====] - 5s 21ms/step - loss: 0.0954 - val_loss: 0.0938
235/235 [=====] - 5s 20ms/step - loss: 0.0948 - val_loss: 0.0933
313/313 [=====] - 1s 2ms/step
313/313 [=====] - 1s 3ms/step

```

# Practical No : 10

Name: Govind Saini  
 Roll No: 07  
 Sign: Govind  
 Date: 19/3/2022  
 Page: \_\_\_\_\_

Practical - 10

Aim:-

Denoising of images using autoencoder

Description:-

	Encoder		Decoder	
Noisy Input	Compressed Representation		Denoised Image	

Denoising auto-encoders are an extension of simple auto-encoders; however it's worth noting that denoising auto-encoders were not originally meant to automatically denoise an image.

Instead, the denoising autoencoder procedure was invented to help:

- i) the hidden layer of the autoencoder learn more robust filters.
- ii) Reduce the risk of overfitting in the autoencoder
- iii) prevent the autoencoder from learning a simple identity function.

Govind

Name: Govind Saini  
Roll No: 07  
Sign: Govind

Date.....  
Page.....

Noise was stochastically added to the input data, and then the autoencoder was trained to return the original, unperturbed signal.

From an image processing standpoint, we can train an autoencoder to perform automatic image pre-processing for us.

A great example would be pre-processing an image to improve the accuracy of an Optical character recognition (OCR) algorithm. If you've ever applied OCR before, you know how just a little bit of the wrong type of noise can dramatically hurt the performance of your OCR method.

Using denoising autoencoders, we can automatically preprocess the image, improve the quality, and therefore increase the accuracy of the downstream OCR algorithm.

Data set Description:-

Title: MNIST

The MNIST dataset contains 70,000 images of handwritten digits (0 to 9) that have been size-normalized and centered in a square grid of pixels. Each image is a  $28 \times 28 \times 1$  array of floating-point numbers representing grayscale intensities ranging from 0 (black) to 1 (white).

Govind

Name: Govind Saini  
 Roll No: 07  
 Sign: Govind

Date: .....  
 Page: .....

The target data consists of one-hot binary vector of size 10, corresponding to the digit classification categories zero through nine. Some sample MNIST images are shown below:

Information: \* name: MNIST \* length: 70000

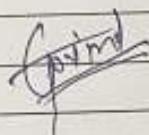
Input Summary: \* shape: (28, 28, 1) \* range: (0.0, 1.0)

Target Summary: \* shape: (10) \* range: (0.0, 1.0)

Result Description:-

Now, let's take a look at the results. Top, the noisy digits fed to the network, and bottom, the digits are reconstructed by the network. On the top we have the original MNIST digit we added noise to while on the bottom we have the output of the denoising autoencoder - we can clearly see that the denoising autoencoder was able to recover the original signal from the image while ignoring the noise.

More advanced denoising autoencoders can be used to automatically pre-process images to facilitate better OCR accuracy.



**Code :**

```
pract10.py > ...
1 import keras
2 from keras.datasets import mnist
3 from keras import layers
4 import numpy as np
5 from keras.callbacks import TensorBoard
6 import matplotlib.pyplot as plt
7
8 print("7_GovindSaini")
9
10 (X_train, _), (X_test, _) = mnist.load_data()
11 X_train = X_train.astype('float32')/255.
12 X_test = X_test.astype('float32')/255.
13 X_train = np.reshape(X_train, (len(X_train), 28, 28, 1))
14 X_test = np.reshape(X_test, (len(X_test), 28, 28, 1))
15 noise_factor = 0.5
16
17 X_train_noisy = X_train+noise_factor * \
18     np.random.normal(loc=0.0, scale=1.0, size=X_train.shape)
19 X_test_noisy = X_test+noise_factor * \
20     np.random.normal(loc=0.0, scale=1.0, size=X_test.shape)
21 X_train_noisy = np.clip(X_train_noisy, 0., 1.)
22 X_test_noisy = np.clip(X_test_noisy, 0., 1.)
23
24 # Use Matplotlib (don't ask) How many digits we will display
25 n = 10

admin@DESKTOP-3B61I80 MINGW64 /c/govindworking/Deep Learning Practical
$
```

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** On the left, it lists files under the "DEEP LEARNING PRACTICAL" folder, including `venv.rar`, `alone.data`, `ames_iowa_housing..`, `ames.csv`, `car_evaluation.csv`, `data.csv`, `housing.csv`, `iris.data`, `main.py`, `p.py`, `p3.py`, `penguins.csv`, `pract1.py`, `pract2.py`, `pract3.py`, `pract4.py`, `pract5a.py`, `pract5b.py`, `pract5c.py`, `pract6a.py`, `pract6b.py`, `pract7.py`, `pract8.py`, `pract9.py`, and `sonar.csv`.
- Code Editor:** The main area displays a Python script named `pract10.py`. The code uses Matplotlib to show reconstructions and performs a series of convolutional and deconvolutional layers on an input image.
- Terminal:** At the bottom, the terminal window shows the command `admin@DESKTOP-3B61I80 MINGW64 /c/govind/working/Deep Learning Practical` and the file `$`.
- Status Bar:** The status bar at the bottom right shows "Ln 26, Col 28", "Spaces: 4", "UTF-8", "CRLF", "Python 3.9.7 64-bit", and "Prettier".

```

47     decoded = layers.Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x)
48
49     # Create the decoder model
50     autoencoder = keras.Model(input_img, decoded)
51     autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
52     autoencoder.fit(X_train_noisy, X_train,
53                      epochs=3,
54                      batch_size=32,
55                      shuffle=True,
56                      validation_data=(X_test_noisy, X_test),
57                      callbacks=[TensorBoard(log_dir='/tmp/tb', histogram_freq=0, write_graph=False)])
58
59     predictions = autoencoder.predict(X_test_noisy)
60
61     m = 10
62     plt.figure(figsize=(20, 2))
63     # Display original
64     for i in range(1, m+1):
65         ax = plt.subplot(1, m, i)
66         plt.imshow(predictions[i].reshape(28, 28))
67         plt.gray()
68         ax.get_xaxis().set_visible(False)
69         ax.get_yaxis().set_visible(False)
70
71     plt.show()

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

admin@DESKTOP-3B61180 MINGW64 /c/govindworking/Deep Learning Practical

Ln 26, Col 28 Spaces: 4 UTF-8 CRLF Python 3.9.7 64-bit Prettier

31°C Haze 15:09 20-03-2022

## Output :

Figure 1

1 import keras  
2 from keras.datasets import mnist  
3 from keras import layers  
4 import numpy as np

admin@DESKTOP-3B61180 MINGW64 /c/govindworking/Deep Learning Practical

\$ py pract10.py  
7\_GovindSaini

Ln 8, Col 21 Spaces: 4 UTF-8 CRLF Python 3.9.7 64-bit Prettier

31°C Haze 15:13 20-03-2022

The screenshot shows a Visual Studio Code interface for a "DEEP LEARNING PRACTICAL" project. The Explorer sidebar lists various files including .venv.rar, abalone.data, ames\_iowa\_housing..., ames.csv, car\_evaluation.csv, data.csv, housing.csv, iris.data, main.py, p.py, p3.py, penguins.csv, pract1.py, pract2.py, pract3.py, pract4.py, pract5a.py, pract5b.py, pract5c.py, pract6a.py, pract6b.py, pract7.py, pract8.py, pract9.py, pract10.py, and sonar.csv. The Terminal tab shows command-line output from a Python session running pract10.py, which includes TensorFlow logs and a neural network visualization. The Output tab displays a grid of handwritten digits from 0 to 9. The status bar at the bottom provides file information (Ln 8, Col 21), encoding (UTF-8), and system details (31°C Haze, 15:22, 20-03-2022).