ARTIFICIAL INTELLIGENCE

BY: EKTA GUPTA
20100BTCSDSIO7270

INTRODUCTION

- Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision.
- AI requires a foundation of specialized hardware and software for writing and training machine learning algorithms. No one programming language is synonymous with AI, but a few, including Python, R and Java, are popular.
- In general, AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states.

HISTORY AND EVOLUTION OF A.I.

- Here's a brief timeline of the past decades of how AI evolved from its inception:
- Maturation of Artificial Intelligence (1943-1952)
- The birth of Artificial Intelligence (1952-1956)
- The golden years-Early enthusiasm (1956-1974)
- The first AI winter (1974-1980)
- A boom of AI (1980-1987)
- The second AI winter (1987-1993)
- The emergence of intelligent agents (1993-2011)
- Deep learning, big data and artificial general intelligence (2011-present)

HISTORY AND EVOLUTION OF A.I.

- Year 1943: The first work which is now recognized as AI was done by Warren McCulloch and Walter pits in 1943. They proposed a model of artificial neurons.
- Year 1949: Donald Hebb demonstrated an updating rule for modifying the connection strength between neurons. His rule is now called Hebbian learning.
- Year 1950: The Alan Turing who was an English mathematician and pioneered Machine learning in 1950. Alan Turing publishes "Computing Machinery and Intelligence" in which he proposed a test. The test can check the machine's ability to exhibit intelligent behavior equivalent to human intelligence, called a Turing test.
- 1956 - John McCarthy coined the term ‘artificial intelligence’ and had the first AI conference.
- 1969 - Shakey was the first general-purpose mobile robot built. It is now able to do things with a purpose vs. just a list of instructions.

HISTORY AND EVOLUTION OF A.I.

- 2002 - The first commercially successful robotic vacuum cleaner was created.

- 2005 - 2019 - Today, we have speech recognition, robotic process automation (RPA), a dancing robot, smart homes, and other innovations make their debut.

- 2020 - Baidu releases the Linear Fold AI algorithm to medical and scientific and medical teams developing a vaccine during the early stages of the SARS-CoV-2 (COVID-19) pandemic. The algorithm can predict the RNA sequence of the virus in only 27 seconds, which is 120 times faster than other methods.

IMPACT OF A.I. FOR THE BETTER

- Man has long feared the rise of the machine – his own creation becoming smarter and more intelligent than he. But while artificial intelligence and machine learning are rapidly changing the world and powering the Fourth Industrial Revolution, humanity does not need to be afraid.
- Creating New Jobs
- AI “is an opportunity for workers to focus on the parts of their jobs that may also be the most satisfying to them,” says Frantz.
- Bridging Language Divides
- Whether it’s teaching new languages in a personalized way or translating speech and text in real-time, AI-powered language tools from Duolingo to Skype are bridging social and cultural divides in our workplaces, classrooms and everyday lives.

IMPACT OF A.I. FOR THE BETTER

- Transforming Government
- Less paperwork, quicker responses, a more efficient bureaucracy – AI has the power to drastically change public administration, but are governments ready? This tech comes with both risks and opportunities that need to be understood and evaluated
- Delivering Health Care
- AI has the potential to make health care “much more accessible and more affordable,” insists Paul Bates, director of NHS services at Babylon Health. Patients can get an accurate, safe, and convenient answer in seconds – and save health care providers’ money too.
- Creating Art
- Computational creativity is drastically changing the nature of art. Software, more than a tool, is becoming a creative collaborator, merging computer scientist with artist.

A.I. EXPLAINED

- John McCarthy offers the following definition in this 2004 paper " It is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable.“
- Alan Turing’s definition would have fallen under the category of “systems that act like humans.”
- At its simplest form, artificial intelligence is a field, which combines computer science and robust datasets, to enable problem-solving. It also encompasses sub-fields of machine learning and deep learning, which are frequently mentioned in conjunction with artificial intelligence.

A.I. TYPES

- These are the three stages through which AI can evolve, rather than the 3 types of Artificial Intelligence.
- Artificial Narrow Intelligence
- Artificial General Intelligence
- Artificial Super Intelligence

A.I. TYPES

- Artificial Narrow Intelligence (ANI):
- Also known as Weak AI, ANI is the stage of Artificial Intelligence involving machines that can perform only a narrowly defined set of specific tasks. At this stage, the machine does not possess any thinking ability, it just performs a set of pre-defined functions.
- Examples of Weak AI include Siri, Alexa, Self-driving cars, Alpha-Go, Sophia the humanoid and so on. Almost all the AI-based systems built till this date fall under the category of Weak AI.

A.I. TYPES

- Artificial General Intelligence (AGI):
- Also known as Strong AI, AGI is the stage in the evolution of Artificial Intelligence wherein machines will possess the ability to think and make decisions just like us humans.
- There are currently no existing examples of Strong AI, however, it is believed that we will soon be able to create machines that are as smart as humans.
- Artificial Super Intelligence (ASI):
- Artificial Super Intelligence is the stage of Artificial Intelligence when the capability of computers will surpass human beings. ASI is currently a hypothetical situation as depicted in movies and science fiction books, where machines have taken over the world.

A.I. APPLICATION

1. AI Application in E-Commerce

- Personalized Shopping
- AI-Powered Assistants
- Fraud Prevention

2. Applications Of Artificial Intelligence in Education

- Administrative Tasks Automated to Aid Educators
- Creating Smart Content
- Voice Assistants
- Personalized Learning

A.I. APPLICATION

3. Applications of Artificial Intelligence in Lifestyle
   - Autonomous Vehicles
   - Spam Filters
   - Facial Recognition
   - Recommendation System
4. Applications of Artificial Intelligence in Navigation
5. Applications of Artificial Intelligence in Robotics
6. Applications of Artificial Intelligence in Human Resource
7. Applications of Artificial Intelligence in Healthcare
8. Applications of Artificial Intelligence in Agriculture
9. Applications of Artificial Intelligence in Gaming
10. Applications of Artificial Intelligence in Automobiles

UNIT – II

AI INDUSTRY ADOPTION APPROACHES

TOPICS

- AI Industry Impact
- Autonomous Vehicles
- Smart Robotics
- Future Workforce and AI
- Applications of AI
- Focus on AI

AI INDUSTRY IMPACT

- **Voice-Based Search: -**

The development of artificial intelligence (AI) has had a profound impact on nearly every area of human existence. One AI technology that benefits from AI's ability to streamline processes for its users is speech recognition. The latest innovation can transcribe your voicemails into text for you. It can also teach your speech to issue commands with your voice. As a result, Apple, Microsoft, Amazon, Google, Facebook, etc., have given this AI-powered speech recognition technology a lot of attention. Devices and software with built-in voice recognition are already commonplace; examples include Amazon's Echo, Apple's Siri, and Google's Home.

- **AI In Healthcare: -**

The use of AI in healthcare has only recently begun. Machine learning (ML) offers promise in identifying trends within a population, similar how computer vision (CV) can diagnose diseases using X-rays and NLP (natural language processing) can in medication safety. All these advancements for patients will come together once we achieve real information interoperability, which supports the safe interchange of health data.

- **The Impact of AI on Jobs: -**

Artificial intelligence (AI) will have future repercussions for your company. There is the possibility of a significant effect on how your business functions, yet there is no apparent downside. It calls for an optimistic outlook and an eagerness to try something new.

Many sectors of the economy are experiencing profound shifts due to the advent of AI. When applied to commercial processes, AI's ability to recognize patterns and spot anomalies in massive volumes of digital information usher new avenues of opportunity. It can perform a wide variety of everyday activities competently if trained.

Employees are freed from mundane, repetitive work so they may focus on higher-level technical difficulties or enhance the quality of customer care made possible by the advent of AI.

- **Business Intelligence: -**

Data produced by customers, tools, and procedures is overwhelming for businesses. People find that the standard business intelligence tools they've been using still need to cut it. Artificial intelligence-driven solutions will soon replace traditional methods like spreadsheets and dashboards. These instruments can automatically probe the data, learn insights, and make recommendations. These resources will transform how businesses leverage information and make decisions.

- **Education: -**

An area where AI is expected to have a significant impact on education. AI will significantly assist the education sector and system, which needs a considerable overhaul. All that must be done is pinpoint what needs to change and lay out a plan for implementing that change. One factor that could prove to be a game-changer in this regard is the use of AI easy writer in creating a unique, practical, and practically applicable learning path for any subject or topic.

- **Retail: -**

Artificial intelligence will have profound effects on the retail sector. Retailers will invest heavily in artificial intelligence (AI) to provide superior customer service in the following years. Marketers in the retail industry are anticipated to start using AR/VR capabilities. The popularity of interactive, aesthetically pleasing product catalogues, where the final consumer can try out an item before buying it, is predicted to skyrocket.

- Your Gateway to Becoming a Successful AI Expert

AUTONOMOUS VEHICLES

The 5 levels of autonomous vehicles

- Level 0 (No Driving Automation)
- Level 1 (Driver Assistance)
- Level 2 (Partial Driving Automation)
- Level 3 (Conditional Driving Automation)
- Level 4 (High Driving Automation)
- Level 5 (Full Driving Automation)

SMART ROBOTICS

- The latest developments in smart robotics
- Autonomous robots
- Collaborative robots
- Deep learning
- Human-robot interaction
- Cloud robotics
- 5G connectivity

BENEFITS OF SMART ROBOTICS

- Increased efficiency
- Automation
- Cost savings
- Improved safety
- 24/7 operation
- Flexibility
- Scalability

FUTURE WORKFORCE AND AI

- AI is a fast-evolving technology with great potential to **make workers more productive, to make firms more efficient, and to spur innovations in new products and services**. At the same time, AI can also be used to automate existing jobs and exacerbate inequality, and it can lead to discrimination against workers.

- In reality, AI is already at work all around us, impacting everything from our search results, to our online dating prospects, to the way we shop. Data shows that the use of AI in many sectors of business has grown by 270% over the last four years.

- The depictions of AI in the media have run the gamut, and while no one can predict exactly how it will evolve in the future, the current trends and developments paint a much different picture of how AI will become part of our lives.

APPLICATIONS OF AI

- Personalized Shopping
- AI-Powered Assistants
- Fraud Prevention
- Administrative Tasks Automated to Aid Educators
- Creating Smart Content
- Voice Assistants
- Personalized Learning
- Autonomous Vehicles

MAIN FOCUS OF ARTIFICIAL INTELLIGENCE

AI programming focuses on three cognitive skills:

- **Learning processes**: - This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information.

- **Reasoning processes**: - This aspect of AI programming focuses on choosing the right algorithm to reach a desired outcome.

- **Self-correction processes: -** This aspect of AI programming is designed to continually fine-tune algorithms and ensure they provide the most accurate results possible.
- AI-powered machines can store and analyze data to generate insights such as trends, user behaviours, business insights, etc., to predict events and make future projections

PROGRAM CODE:

1. **Import Library**

- import seaborn as sns
- import pandas as pd
- import numpy as np
-

```
In [1]: import seaborn as sns
        import pandas as pd
        import numpy as np
```

2. **Read Dataset**

- ekta = pd.read_csv("C:\\Users\\HP\\Downloads\\iris1.csv")

```
In [2]: ekta=pd.read_csv("C:\\Users\\HP\\Downloads\\Iris1.csv")

In [ ]:
```

df=ekta

df

```
In [3]: df=ekta
        df
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

Artificial Neural Network – CLASSIFICATION
METHOD 1:

3. Analyse dataset

- #Analyzing by describing data

```
In [4]: ##LABEL ENCODER BY REPLACE
        df['Species'].unique()

Out[4]: array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

df['Species']=df['Species'].replace({'Iris-setosa':0, 'Iris-versicolor':1, 'Iris-virginica':2})

df

```
In [5]: df['Species']=df['Species'].replace({'Iris-setosa':0, 'Iris-versicolor':1, 'Iris-virginica':2})
        df

Out[5]:
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	0
1	2	4.9	3.0	1.4	0.2	0
2	3	4.7	3.2	1.3	0.2	0
3	4	4.6	3.1	1.5	0.2	0
4	5	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	2
146	147	6.3	2.5	5.0	1.9	2
147	148	6.5	3.0	5.2	2.0	2
148	149	6.2	3.4	5.4	2.3	2
149	150	5.9	3.0	5.1	1.8	2

150 rows × 6 columns

4. Defining depenedent and independent variable

x=df.iloc[:,1:5]

y=df.iloc[:,5:]

x,y

```
In [6]: x=df.iloc[:,1:5]
        y=df.iloc[:,5:]
        x,y
```

```
Out[6]: (     SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
         0              5.1           3.5            1.4           0.2
         1              4.9           3.0            1.4           0.2
         2              4.7           3.2            1.3           0.2
         3              4.6           3.1            1.5           0.2
         4              5.0           3.6            1.4           0.2
         ..             ...           ...            ...           ...
         145            6.7           3.0            5.2           2.3
         146            6.3           2.5            5.0           1.9
         147            6.5           3.0            5.2           2.0
         148            6.2           3.4            5.4           2.3
         149            5.9           3.0            5.1           1.8

         [150 rows x 4 columns],
              Species
         0          0
         1          0
         2          0
         3          0
         4          0
         ..       ...
         145        2
         146        2
         147        2
         148        2
         149        2

         [150 rows x 1 columns])
```

5. Spliting x and y in to train and test dataset

```
In [7]: from sklearn.model_selection import train_test_split
```

```
In [8]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.25, random_state=15)
```

```
In [9]: y_train.shape
```

```
Out[9]: (112, 1)
```

6. Import and Build Neural Network Function

In [10]:
```
###Import and Build Neural Network Function
import numpy as np
from keras.utils import np_utils
y_train=np_utils.to_categorical(y_train)
###convert in binary
y_test=np_utils.to_categorical(y_test)
y_train[:5],y_test[:5]
```

Out[10]:
```
(array([[1., 0., 0.],
        [0., 1., 0.],
        [0., 0., 1.],
        [1., 0., 0.],
        [1., 0., 0.]], dtype=float32),
 array([[1., 0., 0.],
        [0., 1., 0.],
        [0., 1., 0.],
        [1., 0., 0.],
        [1., 0., 0.]], dtype=float32))
```

In [11]:
```
print(x_train. shape)
print(y_train. shape)
print(x_test.shape)
print(y_test.shape)
```

```
(112, 4)
(112, 3)
(38, 4)
(38, 3)
```

```
#Keras Library used through tensorflow
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentropy

#single layer feedforward neural networks

model=Sequential([
    #input layer
    Dense(units=5,input_shape=(4,),activation='relu'),

    #hidden layer
    #Dense(units=25,activation='sigmoid'),
    #Dense(units=35,activation='sigmoid'),
    #Dense(units=55,activation='sigmoid'),
```

#output layers
Dense(units=3,activation='sigmoid')
])
model.compile(loss='categorical_crossentropy',optimizer='adam',
metrics='accuracy')

In [12]:
```
#Keras Library used through tensorflow
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentropy
```

In [13]:
```
#single layer feedforward neural networks
```

In [14]:
```
model=Sequential([
    #input layer
    Dense(units=5,input_shape=(4,),activation='relu'),

    #hidden layer
    #Dense(units=25,activation='sigmoid'),
    #Dense(units=35,activation='sigmoid'),
    #Dense(units=55,activation='sigmoid'),

    #output layers
    Dense(units=3,activation='sigmoid')
])
model.compile(loss='categorical_crossentropy',optimizer='adam',
             metrics='accuracy')
```

model.summary()

In [15]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 5)                 25

 dense_1 (Dense)             (None, 3)                 18

=================================================================
Total params: 43
Trainable params: 43
Non-trainable params: 0
_________________________________________________________________
```

```
In [16]: model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=100)

Epoch 1/100
4/4 [==============================] - 1s 126ms/step - loss: 1.0779 - accuracy: 0.6786 - val_loss: 1.1236 - val_accuracy: 0.6
316
Epoch 2/100
4/4 [==============================] - 0s 19ms/step - loss: 1.0598 - accuracy: 0.6786 - val_loss: 1.1035 - val_accuracy: 0.63
16
Epoch 3/100
4/4 [==============================] - 0s 20ms/step - loss: 1.0389 - accuracy: 0.6786 - val_loss: 1.0872 - val_accuracy: 0.63
16
Epoch 4/100
4/4 [==============================] - 0s 19ms/step - loss: 1.0261 - accuracy: 0.6786 - val_loss: 1.0715 - val_accuracy: 0.63
16
Epoch 5/100
4/4 [==============================] - 0s 21ms/step - loss: 1.0113 - accuracy: 0.6786 - val_loss: 1.0578 - val_accuracy: 0.63
16
Epoch 6/100
4/4 [==============================] - 0s 21ms/step - loss: 0.9997 - accuracy: 0.6786 - val_loss: 1.0451 - val_accuracy: 0.63
16
Epoch 7/100
```

```
In [17]: model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 5)                 25

 dense_1 (Dense)             (None, 3)                 18

=================================================================
Total params: 43
Trainable params: 43
Non-trainable params: 0
_________________________________________________________________
```

7. Train the MODEL with x_train and y_train:

Now the model prepare
#train the model
d=model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=100)

```
In [19]: #train the model
         d=model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=100)

Epoch 1/100
4/4 [==============================] - 0s 49ms/step - loss: 0.5892 - accuracy: 0.6786 - val_loss: 0.6251 - val_accuracy: 0.65
79
Epoch 2/100
4/4 [==============================] - 0s 18ms/step - loss: 0.5861 - accuracy: 0.6786 - val_loss: 0.6219 - val_accuracy: 0.65
79
Epoch 3/100
4/4 [==============================] - 0s 19ms/step - loss: 0.5828 - accuracy: 0.6786 - val_loss: 0.6187 - val_accuracy: 0.65
79
Epoch 4/100
4/4 [==============================] - 0s 19ms/step - loss: 0.5797 - accuracy: 0.6875 - val_loss: 0.6155 - val_accuracy: 0.65
79
Epoch 5/100
4/4 [==============================] - 0s 17ms/step - loss: 0.5766 - accuracy: 0.6964 - val_loss: 0.6123 - val_accuracy: 0.65
79
Epoch 6/100
4/4 [==============================] - 0s 22ms/step - loss: 0.5735 - accuracy: 0.6964 - val_loss: 0.6093 - val_accuracy: 0.65
79
Epoch 7/100
```

y_pred1=model.predict(x_test)
y_pred1

```
In [20]: y_pred1=model.predict(x_test)
         y_pred1

         2/2 [==============================] - 0s 4ms/step

Out[20]: array([[0.745857  , 0.20119163, 0.00697512],
                [0.11231225, 0.40072647, 0.29852128],
                [0.1007852 , 0.4867104 , 0.47495878],
                [0.69651306, 0.2250626 , 0.01086757],
                [0.702393  , 0.17399114, 0.00579389],
                [0.10089181, 0.46672413, 0.4367631 ],
                [0.02409555, 0.46267518, 0.68820983],
                [0.20294273, 0.36736256, 0.164963  ],
                [0.12383886, 0.43970233, 0.34817368],
                [0.03694218, 0.5348384 , 0.73715234],
                [0.05162916, 0.45852208, 0.5491486 ],
                [0.11032084, 0.4022495 , 0.30408794],
                [0.10647512, 0.38891825, 0.28777432],
                [0.14064859, 0.4271151 , 0.30376276],
                [0.03535389, 0.46205088, 0.62385184],
                [0.66763216, 0.22094017, 0.01141066],
                [0.14822696, 0.41867   , 0.28105068],
                [0.02298459, 0.5378114 , 0.8024925 ],
                [0.72248006, 0.21351749, 0.00875182],
                [0.036745  , 0.49010497, 0.6665392 ],
                [0.11897672, 0.38381225, 0.26184252],
                [0.7359004 , 0.21307406, 0.00830039],
                [0.08333704, 0.39439538, 0.3385552 ],
                [0.18428972, 0.4224983 , 0.25069004],
                [0.77703   , 0.1654017 , 0.00391499],
                [0.7720805 , 0.16437742, 0.00393824],
                [0.02951787, 0.4760314 , 0.67829293],
                [0.04178245, 0.4713938 , 0.61161035],
                [0.01235514, 0.5606576 , 0.88368505],
                [0.0878987 , 0.4350675 , 0.40357518],
                [0.7674335 , 0.17618074, 0.00470406],
                [0.02404032, 0.50267035, 0.75023854],
                [0.04811163, 0.491473  , 0.62312436],
                [0.04092852, 0.5015339 , 0.6682292 ],
                [0.7188535 , 0.18940586, 0.0066679 ],
                [0.764579  , 0.17923878, 0.00495068],
                [0.05970363, 0.46092114, 0.52653307],
                [0.71833026, 0.21590045, 0.00912096]], dtype=float32)
```

8. Predict with x_train and y_train

y_test_class=np.argmax(y_test,axis=1)
y_pred1_class=np.argmax(y_pred1,axis=1)
y_pred1_class,y_test_class

```
In [21]: y_test_class=np.argmax(y_test,axis=1)
         y_pred1_class=np.argmax(y_pred1,axis=1)
         y_pred1_class,y_test_class

Out[21]: (array([0, 1, 1, 0, 0, 1, 2, 1, 1, 2, 2, 1, 1, 1, 2, 0, 1, 2, 0, 2, 1, 0,
                 1, 1, 0, 0, 2, 2, 2, 1, 0, 2, 2, 2, 0, 0, 2, 0], dtype=int64),
          array([0, 1, 1, 0, 0, 1, 2, 1, 1, 2, 2, 1, 1, 1, 2, 0, 1, 2, 0, 2, 1, 0,
                 1, 1, 0, 0, 2, 2, 2, 1, 0, 2, 1, 2, 0, 0, 2, 0], dtype=int64))
```

9. Evaluate using confusion matrix classification report and accuracy score

from sklearn.metrics import classification_report,confusion_matrix,accuracy_score

```
In [22]: from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
```

```
In [23]: print(confusion_matrix(y_test_class,y_pred1_class))
         print(classification_report(y_test_class, y_pred1_class))
         print(accuracy_score (y_test_class, y_pred1_class))
```

```
[[12  0  0]
 [ 0 13  1]
 [ 0  0 12]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       1.00      0.93      0.96        14
           2       0.92      1.00      0.96        12

    accuracy                           0.97        38
   macro avg       0.97      0.98      0.97        38
weighted avg       0.98      0.97      0.97        38

0.9736842105263158
```

METHOD 2:

PROGRAM CODE:

1. **Import Library**

import pydot

import graphviz

from IPython.display import SVG

```
In [25]: import pydot
         import graphviz
         from IPython.display import SVG
```

import keras

from keras.models import Sequential

from keras.layers import Dense

from keras.optimizers import Adam

from keras.utils.vis_utils import model_to_dot

from keras.utils import plot_model

```
In [26]: import keras
         from keras.models import Sequential
         from keras.layers import Dense
         from keras.optimizers import Adam

         from keras.utils.vis_utils import model_to_dot
         from keras.utils import plot_model
```

x=df.iloc[:,0:4].values

y=df.iloc[:,5].values

x,y

```
In [27]: x=df.iloc[:,0:4].values
         y=df.iloc[:,5].values
         x,y
```

```
Out[27]: (array([[ 1. ,  5.1,  3.5,  1.4],
                 [ 2. ,  4.9,  3. ,  1.4],
                 [ 3. ,  4.7,  3.2,  1.3],
                 [ 4. ,  4.6,  3.1,  1.5],
                 [ 5. ,  5. ,  3.6,  1.4],
                 [ 6. ,  5.4,  3.9,  1.7],
                 [ 7. ,  4.6,  3.4,  1.4],
                 [ 8. ,  5. ,  3.4,  1.5],
                 [ 9. ,  4.4,  2.9,  1.4],
                 [ 10. ,  4.9,  3.1,  1.5],
                 [ 11. ,  5.4,  3.7,  1.5],
                 [ 12. ,  4.8,  3.4,  1.6],
                 [ 13. ,  4.8,  3. ,  1.4],
                 [ 14. ,  4.3,  3. ,  1.1],
                 [ 15. ,  5.8,  4. ,  1.2],
                 [ 16. ,  5.7,  4.4,  1.5],
                 [ 17. ,  5.4,  3.9,  1.3],
                 [ 18. ,  5.1,  3.5,  1.4],
                 [ 19. ,  5.7,  3.8,  1.7],
```

y=pd.get_dummies (y).values

y

```
In [28]: y=pd.get_dummies (y).values
         y
```

```
Out[28]: array([[1, 0, 0],
                [1, 0, 0],
                [1, 0, 0],
                [1, 0, 0],
                [1, 0, 0],
                [1, 0, 0],
                [1, 0, 0],
                [1, 0, 0],
                [1, 0, 0],
                [1, 0, 0],
                [1, 0, 0],
                [1, 0, 0],
                [1, 0, 0],
                [1, 0, 0],
                [1, 0, 0],
                [1, 0, 0],
                [1, 0, 0],
                [1, 0, 0],
                [1, 0, 0],
```

Spliting x and y in to train and test dataset

from sklearn.model_selection import train_test_split

x_train, x_test,y_train,y_test=train_test_split(x,y,test_size=.25, random_state=15)

```
In [29]: from sklearn.model_selection import train_test_split
         x_train, x_test,y_train,y_test=train_test_split(x,y,test_size=.25, random_state=15)
```

```
In [30]: print(x_train. shape)
         print(y_train.shape)
         print(x_test.shape)
         print(y_test.shape)

         (112, 4)
         (112, 3)
         (38, 4)
         (38, 3)
```

Import the model/Algorithm

```
In [31]: # building the model
model= Sequential() # here we build sequential model
#input Layer/first layer has 4 neuron or nodes, it could be 100 or 200
model.add(Dense (4, input_shape=(4,), activation='relu'))
'''
#model.add (Dense ((4-nerons/nodes), input_shape=(4,-input feature amt.
alweys use coma represent of input as one dientional array - ),
actiation='relu- relu is rectifire linear unit function-that means if
it get any negative value, it will ake it zero else pass the alue at it is))
'''
#hidden Layer
model.add(Dense (3, activation='softmax')) # if Label/classes more then two
'''
here use three nodes because it has three classes, here using activation
funtion softmax because here label/classes are more then two
'''
Out[31]: '\nhere use three nodes because it has three classes, here using activation \nfuntion softmax because here label/classes are mo
re then two\n'
```

compile the model

model.compile(optimizer='Adam',loss='categorical_crossentropy',

metrics=['accuracy'])

'''we useing here adam optimizer. other stocstice gredient desent

sparse_categorical_crossentropy'''

```
In [32]: # compile the model

model.compile(optimizer='Adam',loss='categorical_crossentropy',
              metrics=['accuracy'])

'''we useing here adam optimizer. other stocstice gredient desent
sparse_categorical_crossentropy'''
Out[32]: 'we useing here adam optimizer. other stocstice gredient desent \nsparse_categorical_crossentropy'
```

model.fit(x_train,y_train, batch_size=100, epochs=1000)

```
In [33]: model.fit(x_train,y_train, batch_size=100, epochs=1000)

Epoch 1/1000
2/2 [==============================] - 1s 18ms/step - loss: 23.0201 - accuracy: 0.2411
Epoch 2/1000
2/2 [==============================] - 0s 7ms/step - loss: 22.7035 - accuracy: 0.3036
Epoch 3/1000
2/2 [==============================] - 0s 6ms/step - loss: 22.4694 - accuracy: 0.3304
Epoch 4/1000
2/2 [==============================] - 0s 11ms/step - loss: 22.2865 - accuracy: 0.3304
Epoch 5/1000
2/2 [==============================] - 0s 7ms/step - loss: 22.1080 - accuracy: 0.3214
Epoch 6/1000
2/2 [==============================] - 0s 17ms/step - loss: 21.9010 - accuracy: 0.3304
Epoch 7/1000
2/2 [==============================] - 0s 17ms/step - loss: 21.6828 - accuracy: 0.3304
Epoch 8/1000
2/2 [==============================] - 0s 16ms/step - loss: 21.4415 - accuracy: 0.3304
Epoch 9/1000
2/2 [==============================] - 0s 14ms/step - loss: 21.1980 - accuracy: 0.3214
Epoch 10/1000
```

model.evaluate(x_test,y_test)

```
In [34]: model.evaluate(x_test,y_test)

2/2 [==============================] - 0s 12ms/step - loss: 0.4919 - accuracy: 0.7895

Out[34]: [0.49188005924224854, 0.7894737124443054]
```

Predict with x_train and y_train

y_pred2=model.predict(x_test)

y_pred2

```
In [35]: y_pred2=model.predict(x_test)
         y_pred2

2/2 [==============================] - 0s 0s/step

Out[35]: array([[9.1064763e-01, 4.6951946e-02, 4.2400502e-02],
               [5.9694212e-02, 5.3801703e-01, 4.0228879e-01],
               [1.9358573e-02, 4.2689103e-01, 5.5375040e-01],
               [8.1531435e-01, 7.3866896e-02, 1.1081878e-01],
               [8.8077152e-01, 4.0391646e-02, 7.8836881e-02],
               [3.2416359e-02, 6.5789223e-01, 3.0969143e-01],
               [3.5596031e-03, 4.4303486e-01, 5.5340552e-01],
               [9.4804160e-02, 1.8779409e-01, 7.1740180e-01],
               [6.2965557e-02, 5.4550928e-01, 3.9152524e-01],
               [3.9606700e-03, 4.8687118e-01, 5.0916821e-01],
               [7.3782774e-03, 2.2816223e-01, 7.6445949e-01],
               [2.9650610e-02, 3.1920001e-01, 6.5114933e-01],
               [4.7471158e-02, 4.6267045e-01, 4.8985836e-01],
               [5.1188480e-02, 5.1086485e-01, 4.3794668e-01],
               [6.2482036e-03, 3.2879314e-01, 6.6495866e-01],
               [8.2897109e-01, 7.4854918e-02, 9.6174024e-02],
               [3.2189570e-02, 3.2319519e-01, 6.4461529e-01],
               [2.8794026e-03, 5.4376090e-01, 4.5335972e-01],
               [8.8304752e-01, 6.2482420e-02, 5.4469988e-02],
               [2.6011942e-03, 3.0438113e-01, 6.9301772e-01],
               [6.5840483e-02, 4.4414219e-01, 4.9001741e-01],
               [8.0545133e-01, 5.0525788e-02, 1.4402285e-01],
               [2.7005251e-02, 4.2657790e-01, 5.4641688e-01],
               [1.1731341e-01, 4.6910688e-01, 4.1357970e-01],
               [8.5443997e-01, 3.2319374e-02, 1.1324063e-01],
               [9.2247564e-01, 4.0206194e-02, 3.7318207e-02],
               [4.9806810e-03, 3.8286701e-01, 6.1215240e-01],
               [2.6669789e-03, 1.6029251e-01, 8.3704048e-01],
               [7.9909666e-04, 4.9465647e-01, 5.0454450e-01],
               [2.8915964e-02, 4.8682001e-01, 4.8426399e-01],
               [8.3768576e-01, 1.0340831e-01, 5.8905832e-02],
               [1.6357146e-03, 2.3374756e-01, 7.6461673e-01],
               [1.4116530e-02, 6.2200850e-01, 3.6387503e-01],
               [4.1775024e-03, 3.6409211e-01, 6.3173044e-01],
               [8.7056792e-01, 5.7897750e-02, 7.1534343e-02],
               [8.8849664e-01, 4.0597942e-02, 7.0905454e-02],
               [6.8253102e-03, 2.2576168e-01, 7.6741296e-01],
               [8.8423294e-01, 5.8642369e-02, 5.7124693e-02]], dtype=float32)
```

y_test_class=np.argmax (y_test, axis=1)

y_pred1_class=np.argmax(y_pred1, axis=1)

y_pred1_class,y_test_class

```
In [36]: y_test_class=np.argmax (y_test, axis=1)
         y_pred1_class=np.argmax(y_pred1, axis=1)
         y_pred1_class,y_test_class

Out[36]: (array([0, 1, 1, 0, 0, 1, 2, 1, 1, 2, 2, 1, 1, 1, 2, 0, 1, 2, 0, 2, 1, 0,
                 1, 1, 0, 0, 2, 2, 2, 1, 0, 2, 2, 2, 0, 0, 2, 0], dtype=int64),
          array([0, 1, 1, 0, 0, 1, 2, 1, 1, 2, 2, 1, 1, 1, 2, 0, 1, 2, 0, 2, 1, 0,
                 1, 1, 0, 0, 2, 2, 2, 1, 0, 2, 1, 2, 0, 0, 2, 0], dtype=int64))
```

Evaluate using confusion matrix classification report and accuracy score

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print(confusion_matrix(y_test_class,y_pred1_class))

print(classification_report(y_test_class,y_pred1_class))

print(accuracy_score(y_test_class,y_pred1_class))

```
In [37]: from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
         print(confusion_matrix(y_test_class,y_pred1_class))
         print(classification_report(y_test_class,y_pred1_class))
         print(accuracy_score(y_test_class,y_pred1_class))

[[12  0  0]
 [ 0 13  1]
 [ 0  0 12]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       1.00      0.93      0.96        14
           2       0.92      1.00      0.96        12

    accuracy                           0.97        38
   macro avg       0.97      0.98      0.97        38
weighted avg       0.98      0.97      0.97        38

0.9736842105263158
```

(12+13+12)/38 #as accuracy

```
In [38]: (12+13+12)/38 #as accuracy

Out[38]: 0.9736842105263158
```

Artificial Neural Network -Regression

In [39]:
```
#Artificial Neural Network - Regression
df
```

Out[39]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	0
1	2	4.9	3.0	1.4	0.2	0
2	3	4.7	3.2	1.3	0.2	0
3	4	4.6	3.1	1.5	0.2	0
4	5	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	2
146	147	6.3	2.5	5.0	1.9	2
147	148	6.5	3.0	5.2	2.0	2
148	149	6.2	3.4	5.4	2.3	2
149	150	5.9	3.0	5.1	1.8	2

150 rows × 6 columns

x=df.iloc[:,1:3]. values

y=df.iloc[:,4].values

x,y

```
In [40]: x=df.iloc[:,1:3]. values
         y=df.iloc[:,4].values
         x,y
```

```
           [5.8, 2.7],
           [6.8, 3.2],
           [6.7, 3.3],
           [6.7, 3. ],
           [6.3, 2.5],
           [6.5, 3. ],
           [6.2, 3.4],
           [5.9, 3. ]]),
 array([0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.2, 0.1,
        0.1, 0.2, 0.4, 0.4, 0.3, 0.3, 0.3, 0.2, 0.4, 0.2, 0.5, 0.2, 0.2,
        0.4, 0.2, 0.2, 0.2, 0.2, 0.4, 0.1, 0.2, 0.1, 0.2, 0.2, 0.1, 0.2,
        0.2, 0.3, 0.3, 0.2, 0.6, 0.4, 0.3, 0.2, 0.2, 0.2, 0.2, 1.4, 1.5,
        1.5, 1.3, 1.5, 1.3, 1.6, 1. , 1.3, 1.4, 1. , 1.5, 1. , 1.4, 1.3,
        1.4, 1.5, 1. , 1.5, 1.1, 1.8, 1.3, 1.5, 1.2, 1.3, 1.4, 1.4, 1.7,
        1.5, 1. , 1.1, 1. , 1.2, 1.6, 1.5, 1.6, 1.5, 1.3, 1.3, 1.3, 1.2,
        1.4, 1.2, 1. , 1.3, 1.2, 1.3, 1.3, 1.1, 1.3, 2.5, 1.9, 2.1, 1.8,
        2.2, 2.1, 1.7, 1.8, 1.8, 2.5, 2. , 1.9, 2.1, 2. , 2.4, 2.3, 1.8,
        2.2, 2.3, 1.5, 2.3, 2. , 2. , 1.8, 2.1, 1.8, 1.8, 1.8, 2.1, 1.6,
        1.9, 2. , 2.2, 1.5, 1.4, 2.3, 2.4, 1.8, 1.8, 2.1, 2.4, 2.3, 1.9,
```

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y, test_size=.25, random_state=15)

```
In [41]: from sklearn.model_selection import train_test_split
         x_train,x_test,y_train,y_test=train_test_split(x,y, test_size=.25, random_state=15)
```

```
In [42]: #mlr

         from sklearn import linear_model
         model=linear_model.LinearRegression()
         model.fit (x_train,y_train)
         print(model.score(x_train,y_train))
         print( '-------------------------------------')
         y_pred=model.predict(x_test)
         print(y_test[:4])
         print(y_pred[:4])
         print('-------------------------------------')
         print(model.coef_[0],model.intercept_)
         print(model.coef_[1],model.intercept_)
         print(model.coef_,model.intercept_)
         print('-------------------------------------')
         print(x_test[:1],x_test[:1,0])
         print('-------------------------------------')
         print('y-b1*x1+b2*x2+c')
         mlr_1=(2.533468405579969*0.08333333+-1.1960867374503343*0.58333333+0.6345516162534967)
         print(mlr_1)
         mlr_=(x_test[:1,0]*model.coef_[0]+x_test[:1,1]*model.coef_[1]+model.intercept_)
         print(mlr_)
```

```
0.754254783437691
------------------------------------
[0.3 1.5 1.2 0.2]
[0.14795672 1.2621681  1.1800194  0.43821581]
-------------------------------------
0.7037412237722136 -1.394796698083412
-0.4983694739418061 -1.394796698083412
[ 0.70374122 -0.49836947] -1.394796698083412
-------------------------------------
[[4.6 3.4]] [4.6]
-------------------------------------
y-b1*x1+b2*x2+c
0.14795671541452682
[0.14795672]
```

In [43]:

```python
from sklearn import metrics

mae=metrics.mean_absolute_error(y_test,y_pred)
mse=metrics.mean_squared_error(y_pred,y_test)
r2sq=metrics.r2_score(y_pred,y_test)
print(mae,mse,r2sq)
```

```
0.3372496113611752 0.17429215121882194 0.4854750463025461
```

In [44]:

```python
x_train[:4,0],x_train[:4],x_test[:4,0],x_test[:4]
```

Out[44]:

```
(array([5.1, 6.6, 7.4, 4.3]),
 array([[5.1, 3.7],
        [6.6, 2.9],
        [7.4, 2.8],
        [4.3, 3. ]]),
 array([4.6, 5.9, 5.5, 4.8]),
 array([[4.6, 3.4],
        [5.9, 3. ],
        [5.5, 2.6],
        [4.8, 3.1]]))
```

x=x_test[:,0]

y=y_test

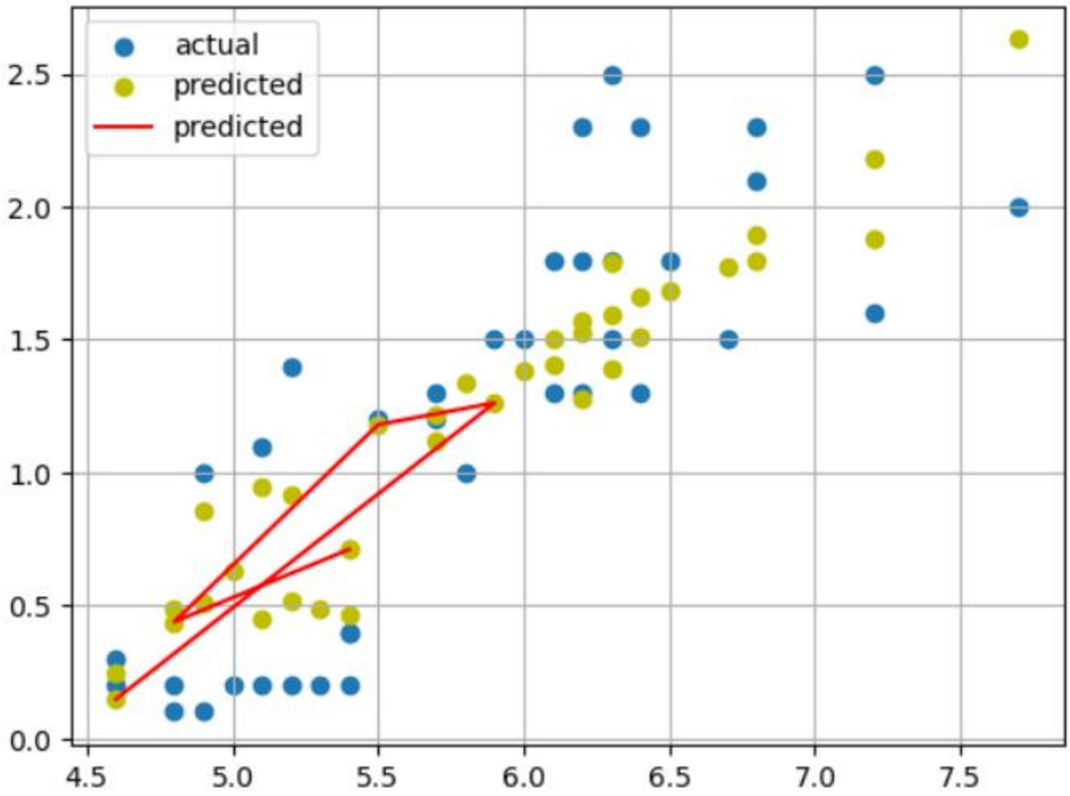
x,y

```
In [45]: x=x_test[:,0]
         y=y_test
         x,y
```

```
Out[45]: (array([4.6, 5.9, 5.5, 4.8, 5.4, 5.7, 7.2, 5.1, 5.2, 6.3, 6.2, 6.2, 6.4,
                 5.8, 6.4, 5. , 5.7, 6.3, 4.9, 7.2, 6.1, 4.6, 6.7, 4.9, 5.3, 5.4,
                 6.8, 6.2, 7.7, 6. , 5.1, 6.8, 6.3, 6.5, 5.4, 5.2, 6.1, 4.8]),
          array([0.3, 1.5, 1.2, 0.2, 0.4, 1.3, 2.5, 1.1, 1.4, 1.8, 1.8, 1.3, 1.3,
                 1. , 2.3, 0.2, 1.2, 2.5, 0.1, 1.6, 1.3, 0.2, 1.5, 1. , 0.2, 0.4,
                 2.1, 2.3, 2. , 1.5, 0.2, 2.3, 1.5, 1.8, 0.2, 0.2, 1.8, 0.1]))
```

```
import matplotlib.pyplot as plt
plt.scatter(x=x_test[:,0],y=y_test,label='actual')
plt.scatter(x_test[:,0],y_pred, c='y',label='predicted')
plt.plot(x_test[:5,0],y_pred[:5], c='r',label='predicted')
plt.legend()
plt.grid()
plt.show()
```

In [46]:
```
import matplotlib.pyplot as plt
plt.scatter(x=x_test[:,0],y=y_test,label='actual')
plt.scatter(x_test[:,0],y_pred, c='y',label='predicted')
plt.plot(x_test[:5,0],y_pred[:5], c='r',label='predicted')
plt.legend()
plt.grid()
plt.show()
```



import matplotlib.pyplot as plt

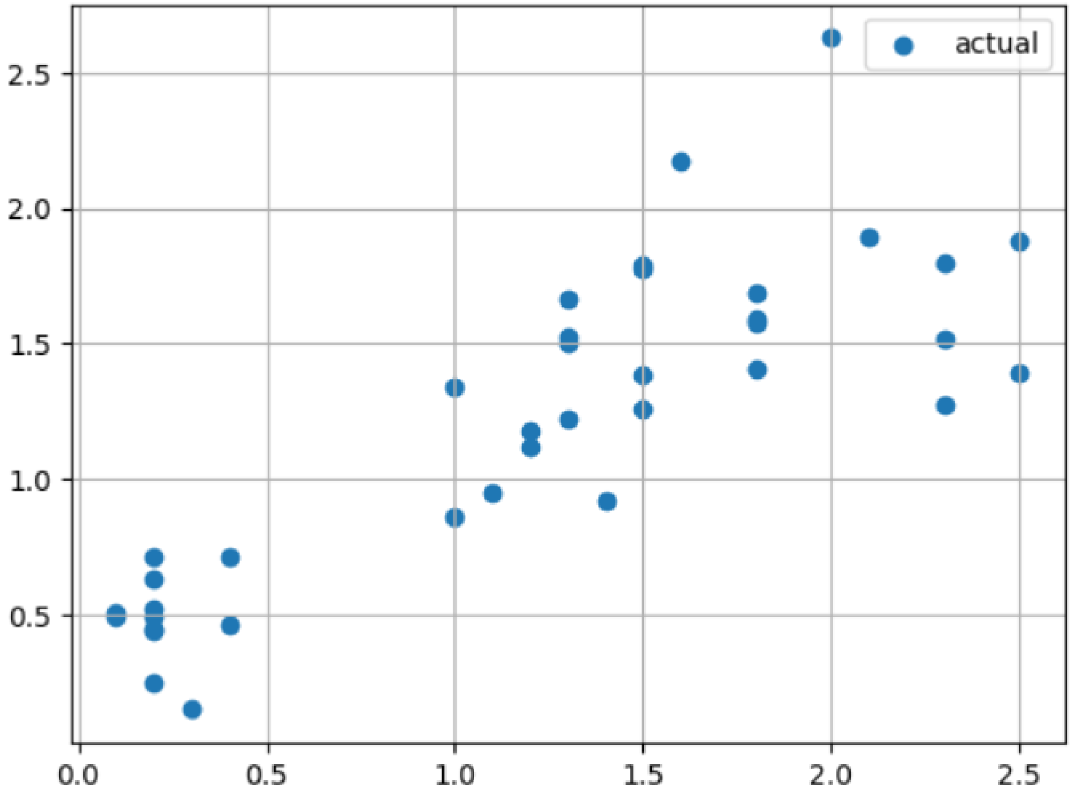
plt.scatter(x=y_test,y=y_pred, label='actual')

plt.legend()

plt.grid()

plt.show()

```
In [47]: import matplotlib.pyplot as plt
         plt.scatter(x=y_test,y=y_pred, label='actual')
         plt.legend()
         plt.grid()
         plt.show()
```



Spliting x and y in to train and test dataset

from sklearn.preprocessing import MinMaxScaler

scaler=MinMaxScaler()

scaler.fit(x_train)

```
In [48]: from sklearn.preprocessing import MinMaxScaler
         scaler=MinMaxScaler()
         scaler.fit(x_train)

Out[48]: MinMaxScaler()
```

x_train=scaler.transform(x_train)

x_test=scaler.transform(x_test)

x_train,x_test

```
In [49]: x_train=scaler.transform(x_train)
         x_test=scaler.transform(x_test)
         x_train,x_test

Out[49]: (array([[0.22222222, 0.70833333],
                 [0.63888889, 0.375     ],
                 [0.86111111, 0.33333333],
                 [0.        , 0.41666667],
                 [0.13888889, 0.58333333],
                 [0.33333333, 0.20833333],
                 [0.44444444, 0.41666667],
                 [0.47222222, 0.41666667],
                 [0.02777778, 0.375     ],
                 [0.30555556, 0.41666667],
                 [0.33333333, 0.625     ],
                 [0.66666667, 0.54166667],
                 [0.38888889, 0.20833333],
                 [0.55555556, 0.29166667],
                 [0.94444444, 0.75     ],
                 [0.5       , 0.41666667],
                 [0.5       , 0.33333333],
                 [0.33333333, 0.16666667],
                 [0.19444444, 0.625     ],
```

pd.DataFrame(x_train).describe()

```
In [50]: pd.DataFrame(x_train).describe()
Out[50]:
```

	0	1
count	112.000000	112.000000
mean	0.429563	0.438616
std	0.235110	0.192528
min	0.000000	0.000000
25%	0.222222	0.333333
50%	0.416667	0.416667
75%	0.590278	0.541667
max	1.000000	1.000000

```
In [51]: pd.DataFrame(x_test).describe()
Out[51]:
```

	0	1
count	38.000000	38.000000
mean	0.426170	0.440789
std	0.217314	0.142252
min	0.083333	0.166667
25%	0.250000	0.343750
50%	0.430556	0.416667
75%	0.555556	0.531250

Import the model/Algorithm

from sklearn.neural_network import MLPRegressor

mlpr=MLPRegressor(hidden_layer_sizes=(1000, 1000,), activation="logistic",

max_iter=10000, solver='lbfgs')

mlpr=MLPRegressor(hidden_layer_sizes=(1000,1000,), activation ='relu',

max_iter=10000, solver='lbfgs')

mlpr=MLPRegressor(hidden_layer_sizes=(1000, 1000,), activation= 'relu',

max_iter=10000, solver= 'lbfgs')

mlpr.fit(x_train, y_train)

```
In [52]: from sklearn.neural_network import MLPRegressor
         mlpr=MLPRegressor(hidden_layer_sizes=(1000, 1000,), activation="logistic",
                         max_iter=10000, solver='lbfgs')
         mlpr=MLPRegressor(hidden_layer_sizes=(1000,1000,), activation ='relu',
                           max_iter=10000, solver='lbfgs')

         mlpr=MLPRegressor(hidden_layer_sizes=(1000, 1000,), activation= 'relu',
                          max_iter=10000, solver= 'lbfgs')

         mlpr.fit(x_train, y_train)

Out[52]: MLPRegressor(hidden_layer_sizes=(1000, 1000), max_iter=10000, solver='lbfgs')
```

Predict with x_train and y_train

y_pred_r=mlpr.predict(x_test)

y_pred_r,y_test

```
In [57]: y_pred_r=mlpr.predict(x_test)
         y_pred_r,y_test

Out[57]: (array([0.20424517, 1.7879318 , 1.25087408, 0.57425398, 0.03932021,
                 1.6290464 , 2.70666423, 2.06644858, 2.49884958, 1.25947858,
                 1.21526937, 1.28974462, 1.21788208, 1.66755274, 1.48954158,
                 0.29842327, 1.43881605, 1.59074591, 0.11554327, 2.13794004,
                 1.17697424, 0.20015279, 1.91251795, 1.35629476, 0.2652608 ,
                 0.41510063, 2.286207  , 2.47268444, 2.00342687, 1.87979468,
                 0.28829388, 2.03991703, 1.89855682, 2.08069283, 0.03932021,
                 0.25715949, 1.42168182, 0.28201587]),
          array([0.3, 1.5, 1.2, 0.2, 0.4, 1.3, 2.5, 1.1, 1.4, 1.8, 1.8, 1.3, 1.3,
                 1. , 2.3, 0.2, 1.2, 2.5, 0.1, 1.6, 1.3, 0.2, 1.5, 1. , 0.2, 0.4,
                 2.1, 2.3, 2. , 1.5, 0.2, 2.3, 1.5, 1.8, 0.2, 0.2, 1.8, 0.1]))
```

jupyter ANN MODEL Last Checkpoint: Last Saturday at 3:11 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
              0.29842327, 1.43881605, 1.59074591, 0.11554327, 2.13794004,
              1.17697424, 0.20015279, 1.91251795, 1.35629476, 0.2652608 ,
              0.41510063, 2.286207  , 2.47268444, 2.00342687, 1.87979468,
              0.28829388, 2.03991703, 1.89855682, 2.08069283, 0.03932021,
              0.25715949, 1.42168182, 0.28201587]),
       array([0.3, 1.5, 1.2, 0.2, 0.4, 1.3, 2.5, 1.1, 1.4, 1.8, 1.8, 1.3, 1.3,
              1. , 2.3, 0.2, 1.2, 2.5, 0.1, 1.6, 1.3, 0.2, 1.5, 1. , 0.2, 0.4,
              2.1, 2.3, 2. , 1.5, 0.2, 2.3, 1.5, 1.8, 0.2, 0.2, 1.8, 0.1]))
```

In [ ]:
```
from sklearn import metrics
```

In [ ]:
```
mae=metrics.mean_absolute_error(y_test,y_pred_r)
mse=metrics.mean_squared_error(y_pred_r,y_test)
r2sq=metrics.r2_score(y_pred_r,y_test)
```

In [ ]:
```
print(mae,mse,r2sq)
```