**EXPERIMENT - 4**

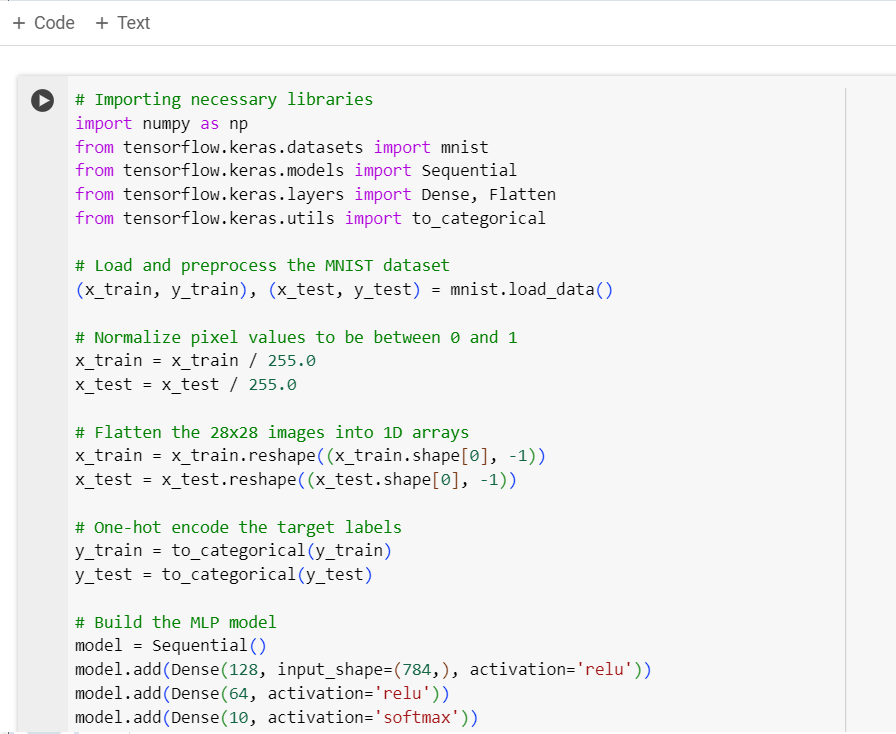
**AIM :** To implement artificial neural networks on a dataset and build a categorical model

**THEORY:**

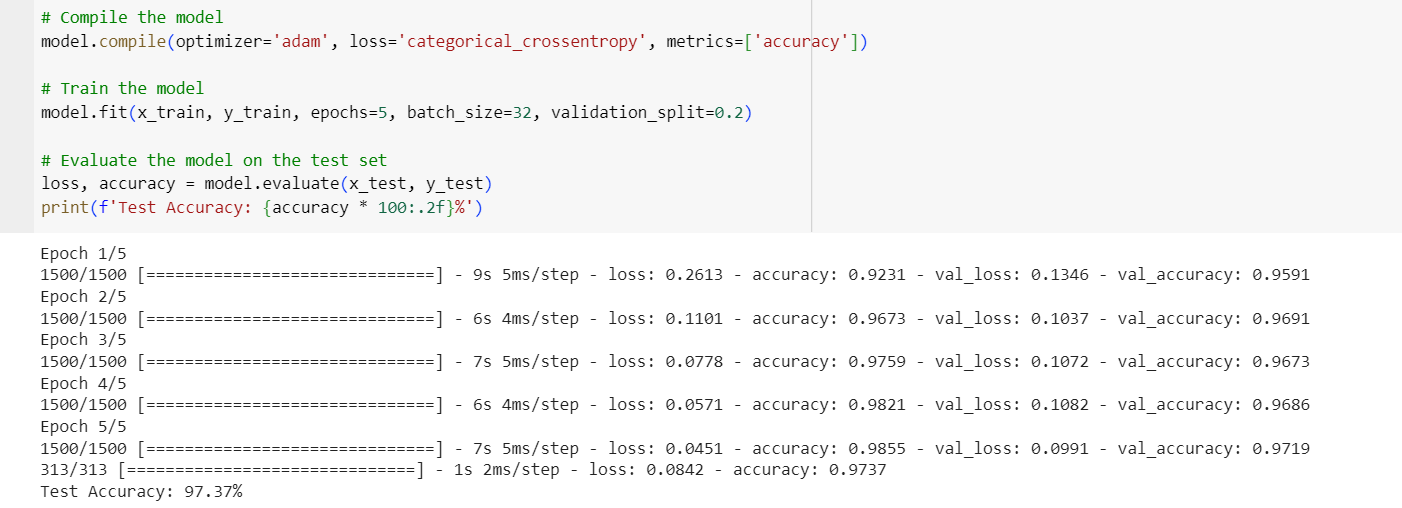
The MNIST dataset is a classic benchmark dataset in machine learning, consisting of 28x28 pixel grayscale images of handwritten digits (0-9). One common approach to categorize the MNIST dataset is by using a neural network model.

**PROGRAM :**

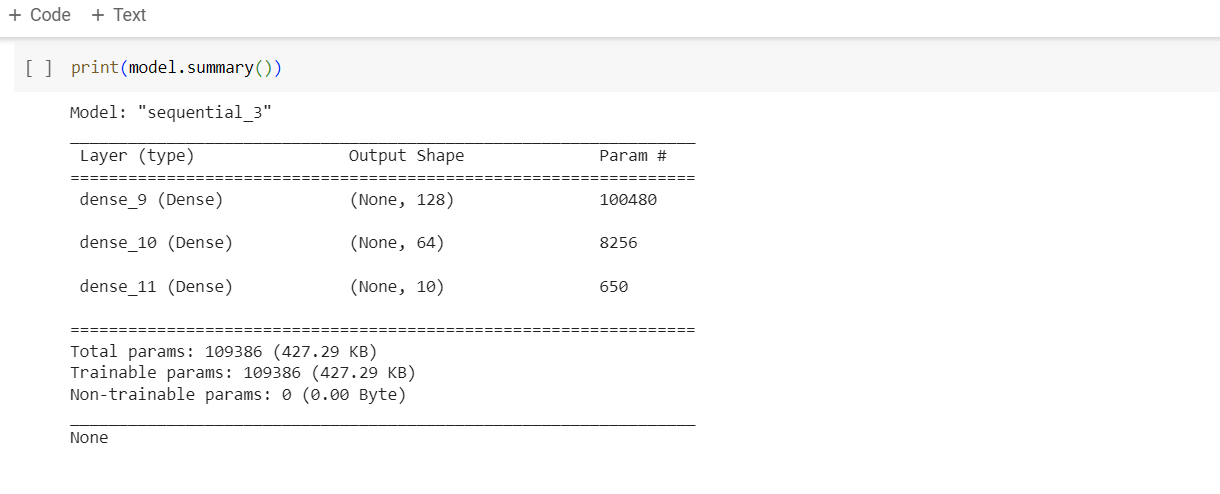
* Loading the data, encoding it and building the neural network.



* Compiling the model with the optimizer Adam, training it and evaluating it.



* A description of the neural network model.



**Result:**

We have successfully developed a neural network model that correctly identifies handwritten numbers 0-9 based on the MNIST dataset with an accuracy of **97.37 %**

We can fine-tune the model's hyperparameters, such as learning rate, batch size,epochs, and network architecture, to optimize performance further.

**EXPERIMENT - 5**

**AIM :** To build and implement a CNN model

**THEORY:**

The CIFAR-10 dataset is a widely used benchmark dataset in computer vision. It consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class.

CNNs have revolutionized the field of computer vision by enabling efficient and accurate image classification tasks like CIFAR-10. Their ability to automatically learn and extract features from raw pixel data makes them indispensable tools for a wide range of applications, from object recognition to medical imaging and beyond.

**PROGRAM :**

* Importing the necessary libraries, downloading, and pre-processing the data for the model

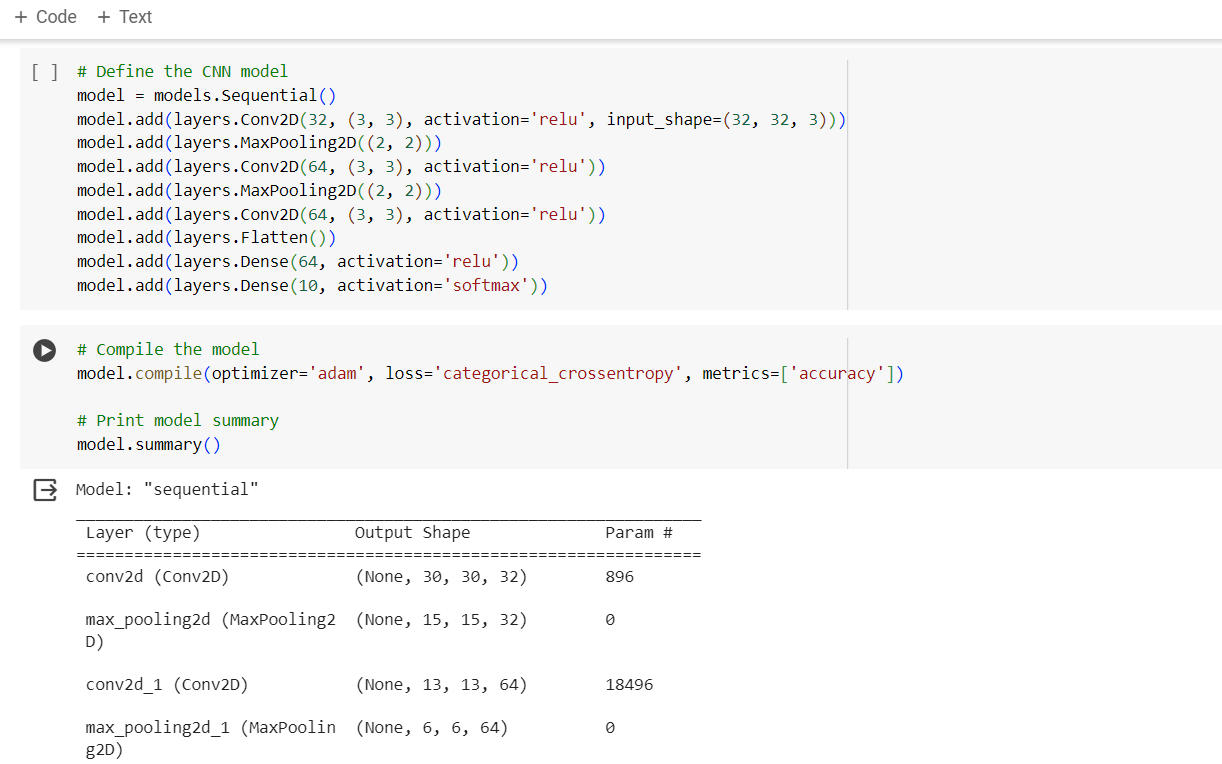


* Displaying some sample images from the CIFAR-10 dataset

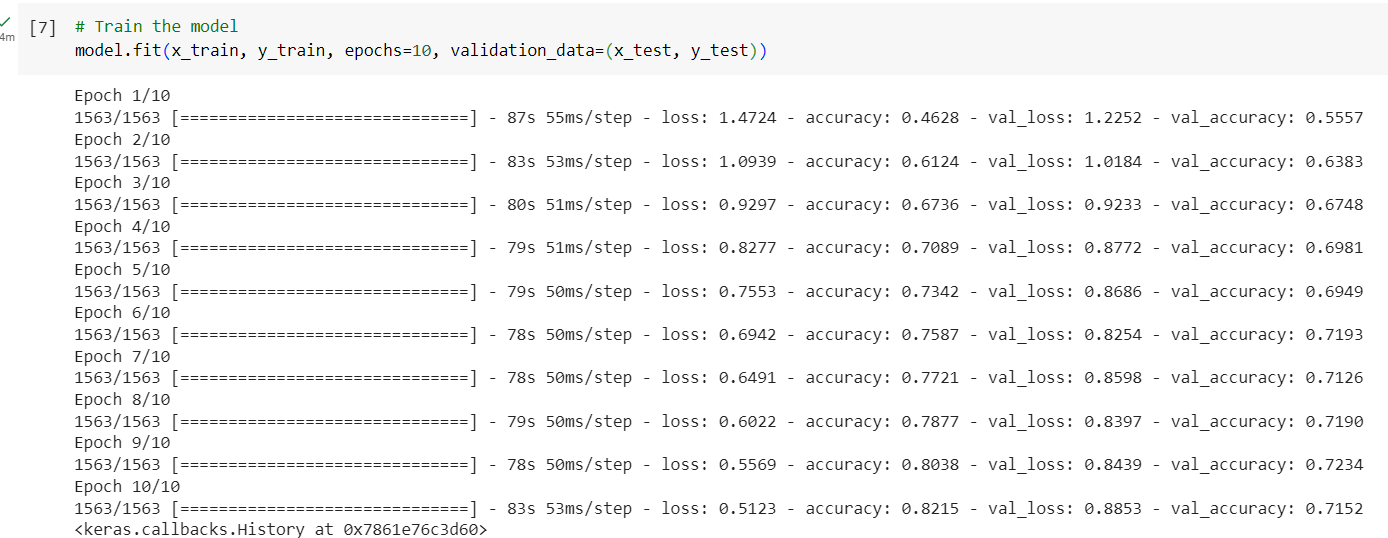


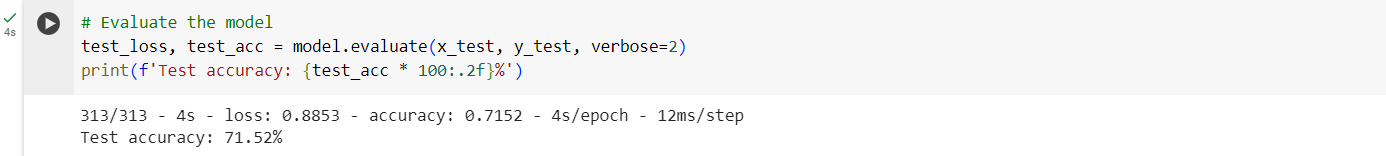


* Building the CNN model and compiling it.



* Training the categorical model



**Result :** We have successfully built a CNN model able to categorize all 10 different image classes of the CIFAR-10 dataset with an accuracy of 71.52% 

**EXPERIMENT - 1**

**AIM :** Exploring Kaggle Website and Downloading Dataset to work upon it.

**THEORY:**

: Kaggle is a popular online platform that provides a collaborative environment for data science and machine learning enthusiasts. Competitions: Kaggle is well-known for hosting data science competitions.

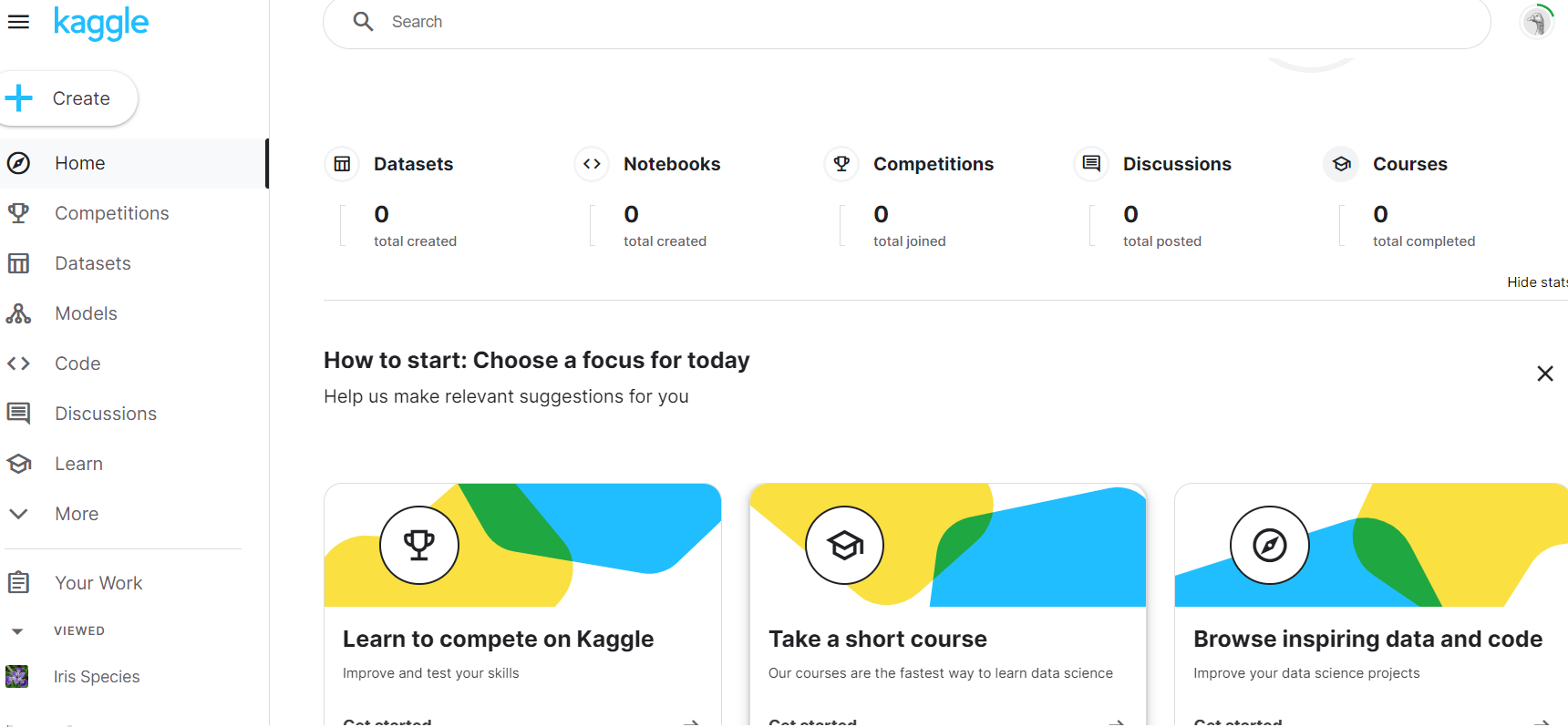
Companies and organizations pose real-world problems, and participants compete to provide the best solutions using their data science and machine learning skills.

Competitions cover a wide range of topics, such as image classification, natural language processing, predictive modeling, and more.

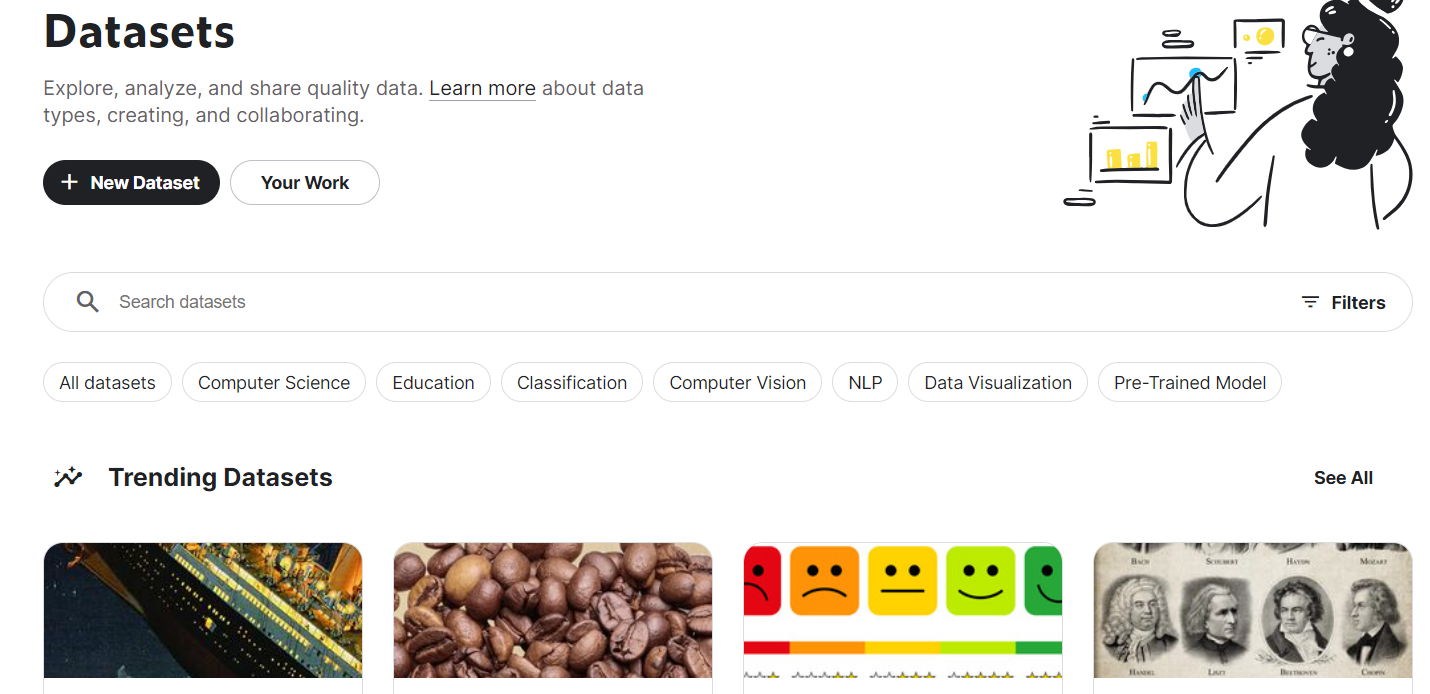
Datasets: Kaggle offers a vast repository of datasets that users can explore and use for their projects or analyses. These datasets cover diverse domains, and many are contributed by the Kaggle community.

Kernels: Kaggle Kernels provide a cloud-based environment for running code, creating and sharing notebooks, and collaborating with others. Users can write code in various languages, including Python and R

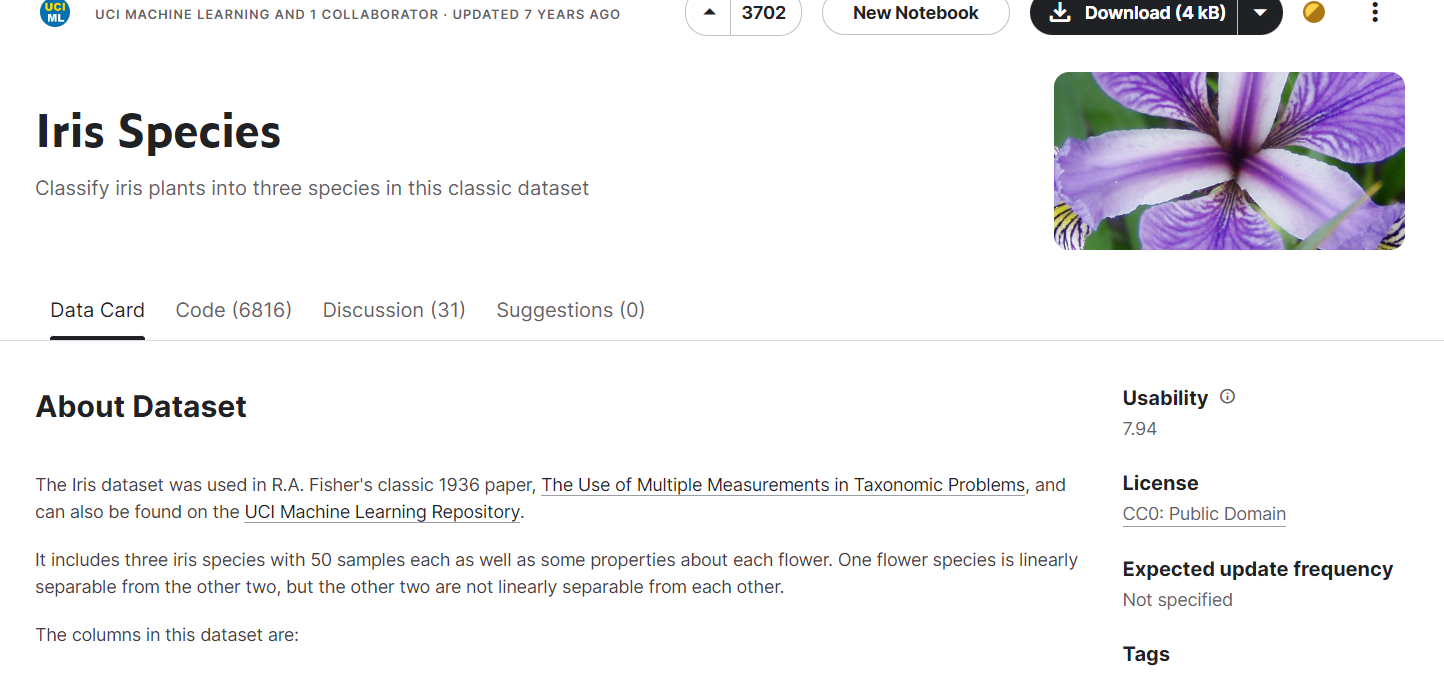
* Kaggle Homepage with all the models, datasets , and events



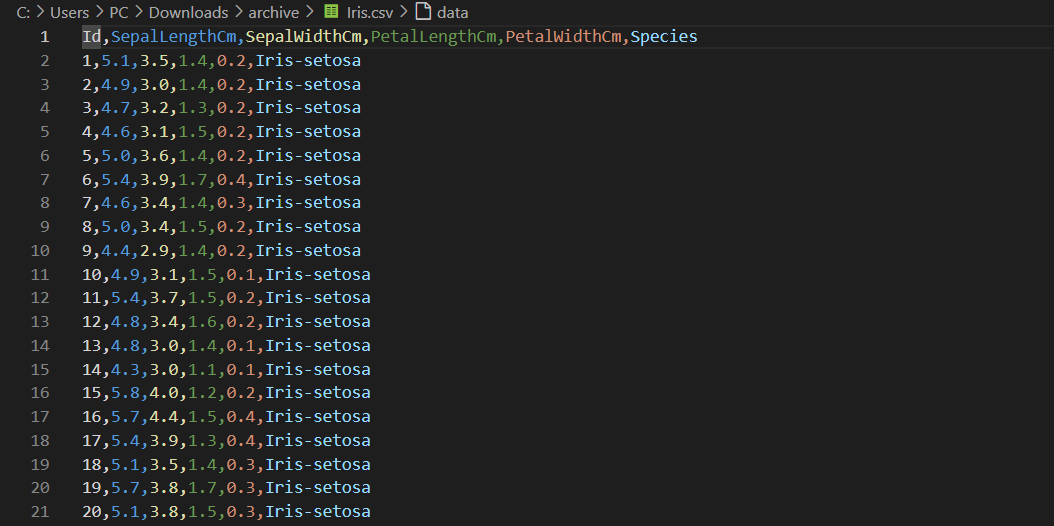
* Dataset page to download different datasets for AI-ML



* Iris dataset available to download as a csv. file



* Downloaded iris dataset



**Result :** We have successfully explored Kaggle and downloaded the iris dataset to work upon it

**EXPERIMENT - 2**

**AIM:** Write a python program on taking a iris data set from Kaggle and Perform these on that data set 1. missing values 2. outliers 3. reputation

**THEORY:**

Import Libraries:

Pandas is imported to work with data frames.

numpy is imported for numerical computations.

seaborn and matplotlib.pyplot are imported for data visualization.

Load the Iris Dataset: The code assumes that you have a CSV file named 'iris.csv'

containing the Iris dataset in the same directory as the script. It loads the dataset into a pandas

DataFrame named iris\_data.

Missing Values: The code checks for missing values in the dataset using the isnull() method

followed by sum() to count the missing values for each column. The result is stored in the

missing\_values variable and then printed.

Outliers: The code visualizes the presence of outliers in the dataset using boxplots. It creates

a figure, plots boxplots for each column in the dataset using sns.boxplot(), and displays the

plot using plt.show().

Reputation: The code assesses the reputation of the dataset by visualizing the distributions

of features. It creates a pairplot using sns.pairplot(), which plots pairwise relationships in the

dataset. The diagonal plots are kernel density estimations (KDEs) which represent the

distribution of each feature. The title is set using plt.suptitle() and the plot is displayed using

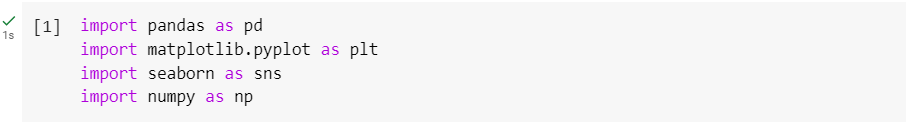
plt.show().

This code provides an overview of the data quality (missing values, outliers) and the

distribution of features in the Iris dataset, helping to understand the dataset's characteristics.

**PROGRAM:**

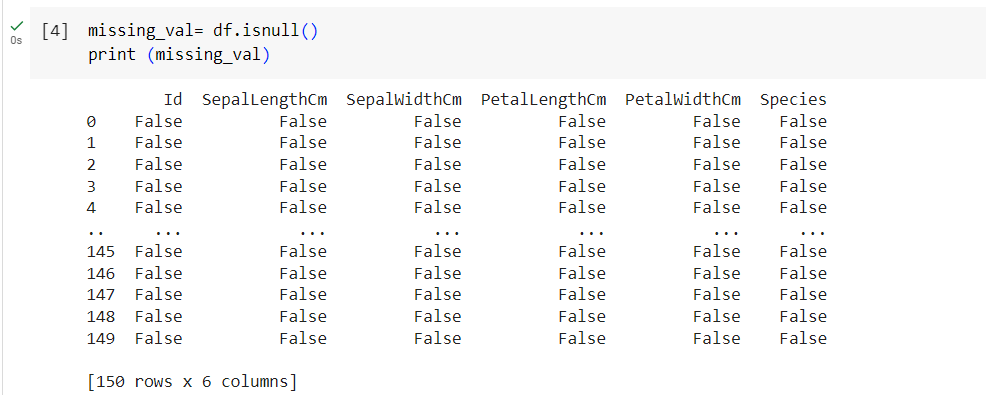
* Importing the necessary libraries, downloading, and pre-processing the data for the model



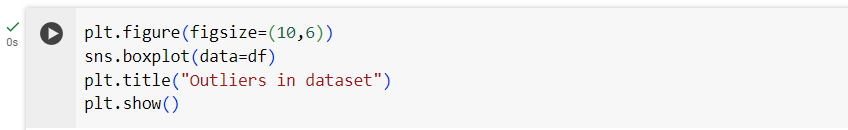
* Reading and checking the data

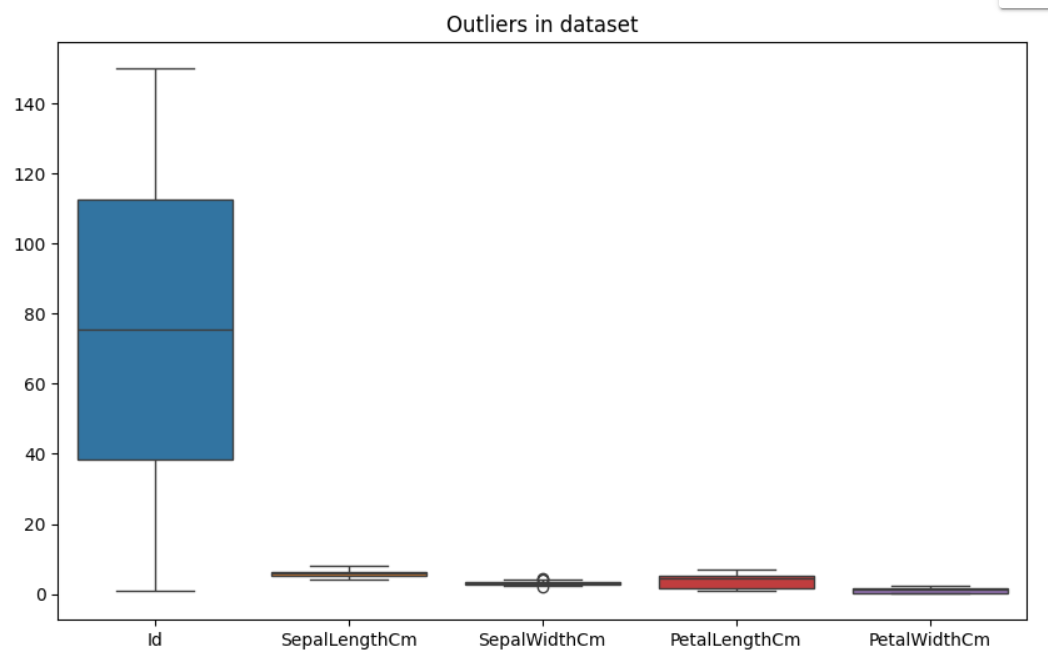


* Visualizing missing values using isnull() function.

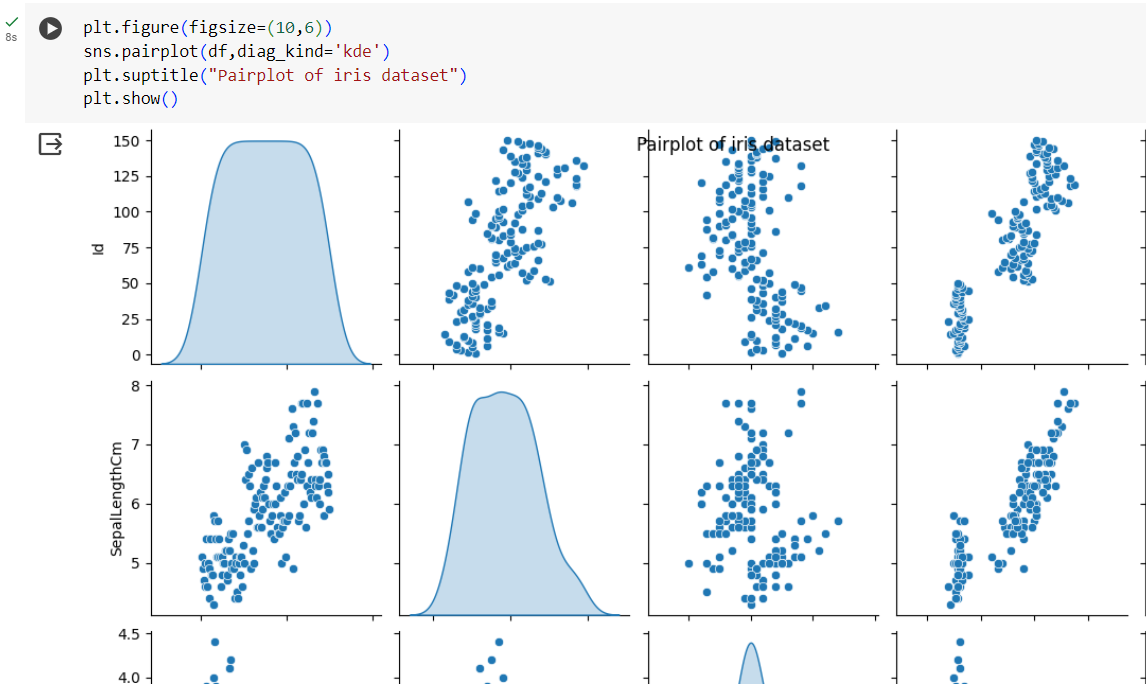


* Visualizing outliers using sns and pyplot libraries. Using boxplot in this case





* Using sns. Pairplot() to show data reputation and relationships



**Result:** We have successfully performed 1. missing values 2. outliers 3. reputation on the dataset

**EXPERIMENT - 3**

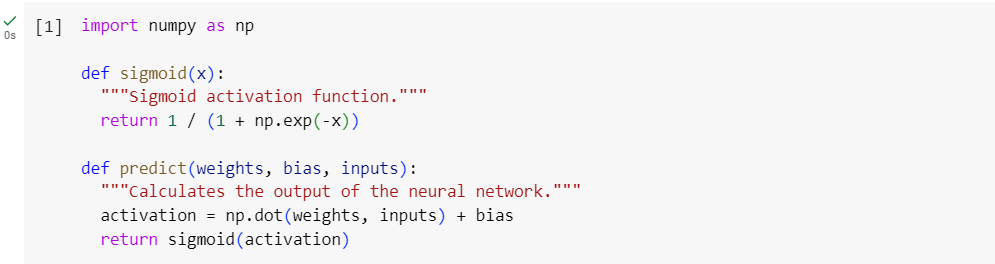
## **AIM:** Write a python program to simulate AND and OR gates using neural networks

**THEORY:**

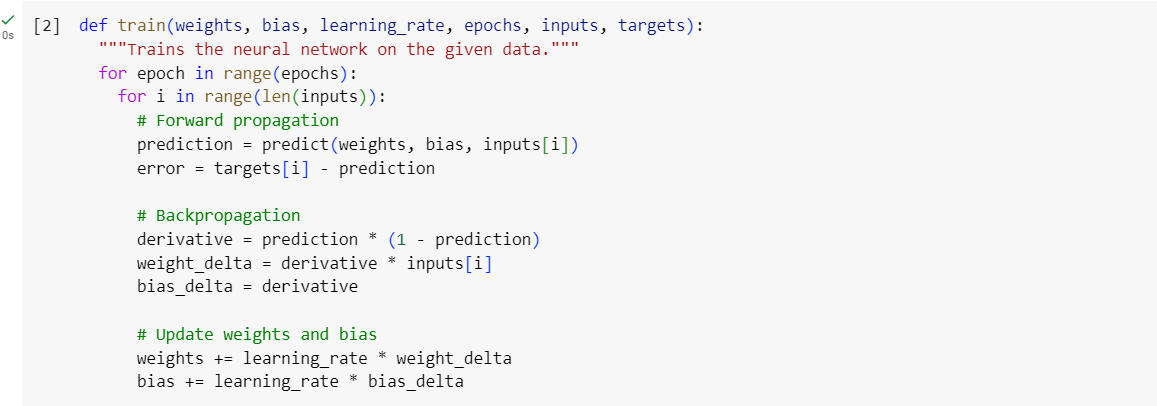
The program simulates AND and OR gates using a simplified machine learning approach inspired by neural networks. It defines a single neuron network with sigmoid activation and trains it on data representing the truth tables of these gates. By adjusting weights and biases through backpropagation, the network learns to mimic the logical behavior of AND and OR operations. While not a full-fledged neural network tackling complex problems, this program demonstrates how machine learning principles can be applied to simulate basic logic gates.

**PROGRAM:**

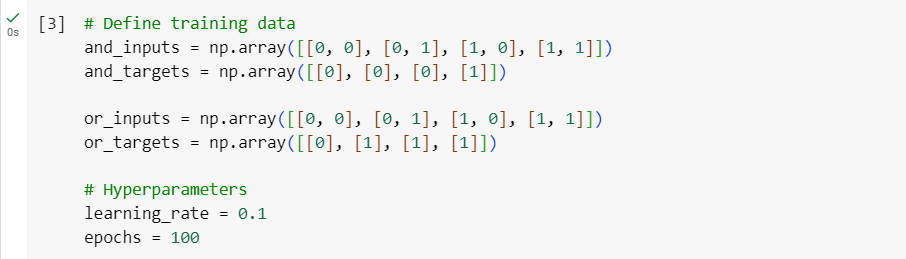
* Importing the necessary libraries, define all the necessary functions of sigmoid , predict etc



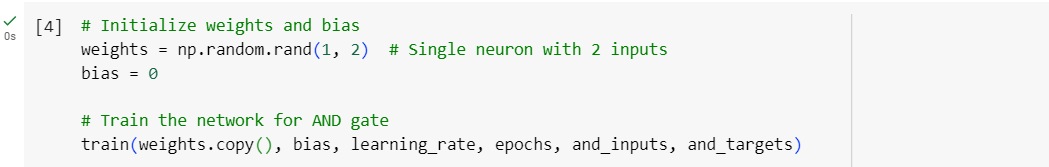
* Define the training function

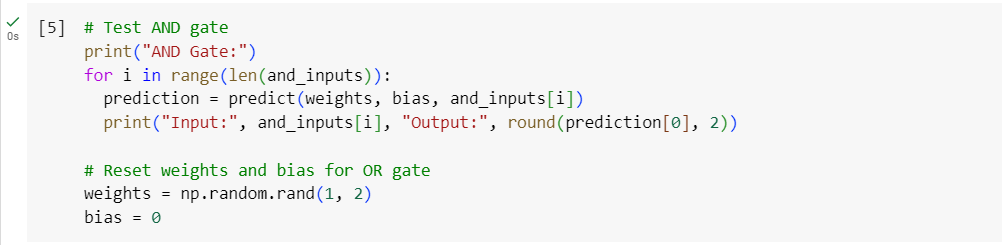


* Define training data and the hyperparameters



* Training the and testing the AND model

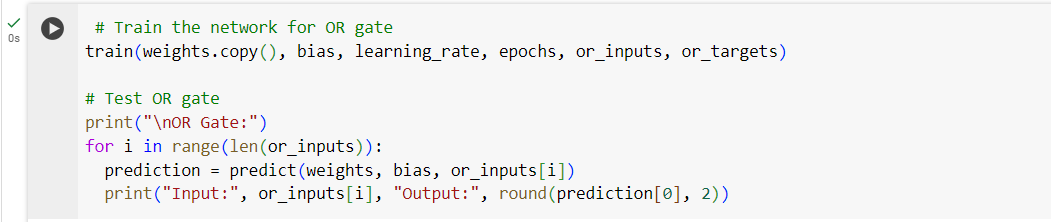




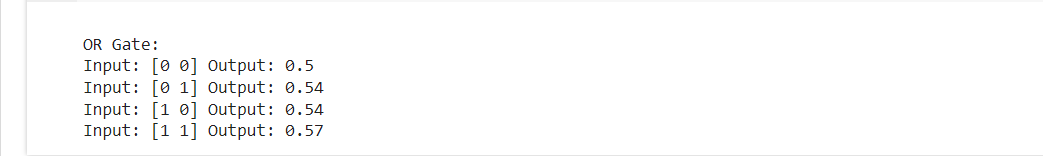
* Output



* Training and testing the OR gate Model



* Output



**Result:** We have successfully implemented the AND and OR gate models using neural networks.

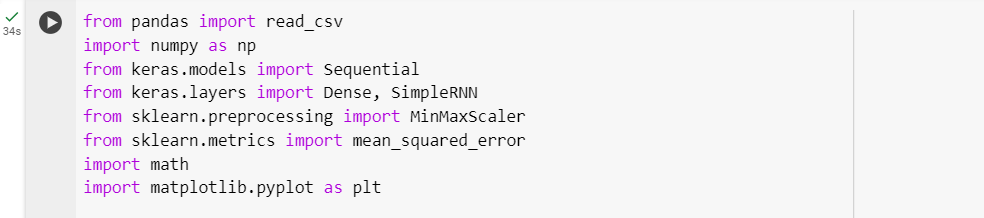
**EXPERIMENT - 6**

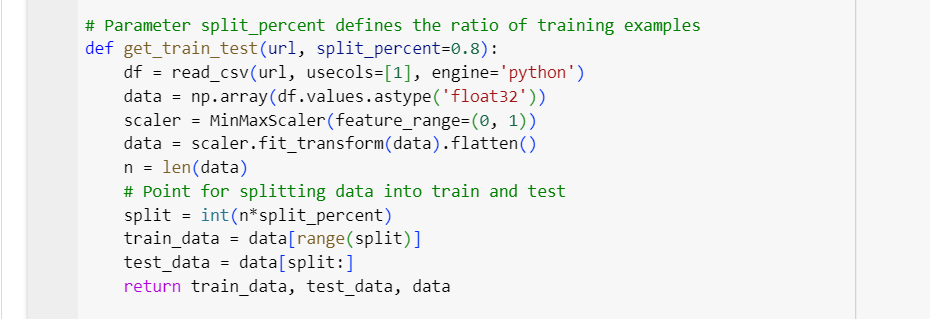
## **AIM:** Implement RNN networks

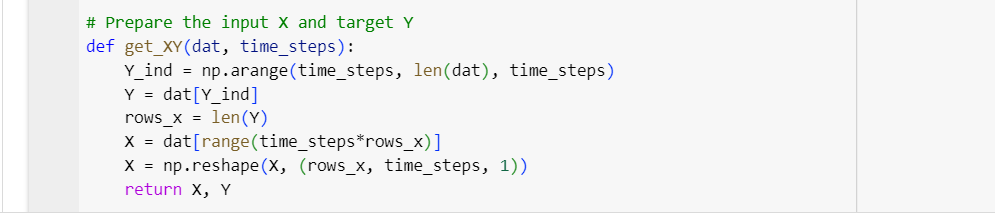
**THEORY:**

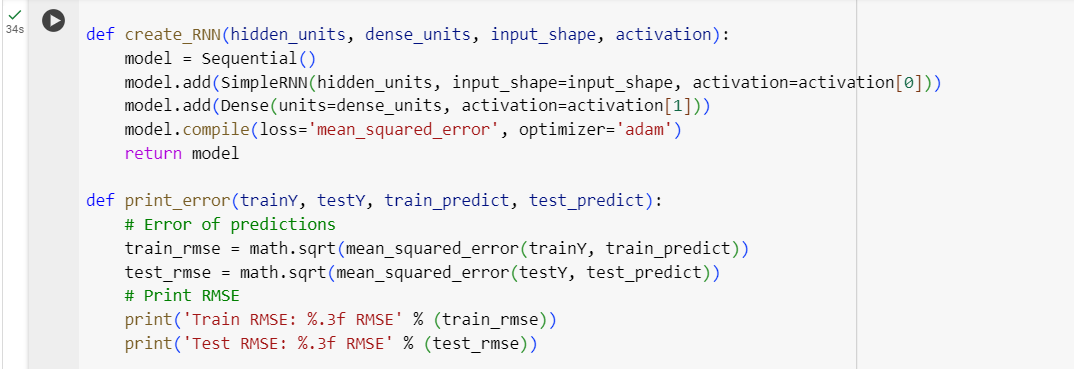
Recurrent Neural Network(RNN) is a type of Neural Network where the output from the previous step is fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other. Still, in cases when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is its Hidden state, which remembers some information about a sequence. The state is also referred to as Memory State since it remembers the previous input to the network. It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks.

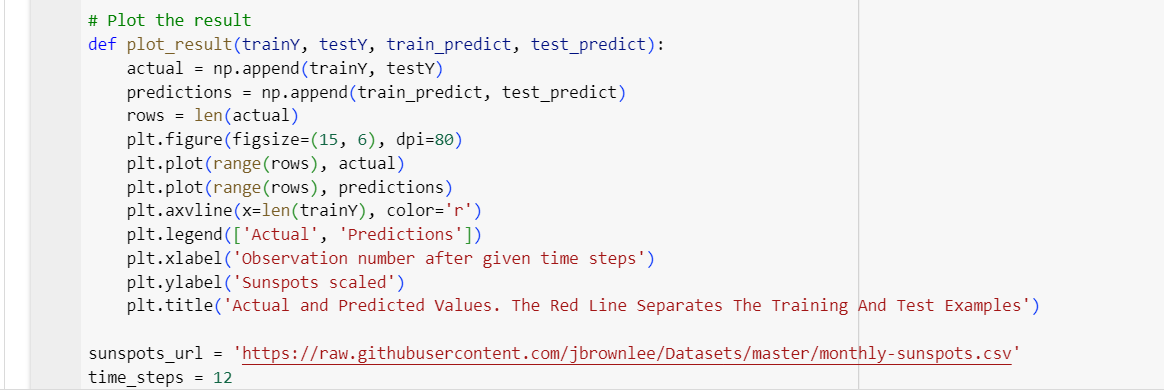
**PROGRAM:**

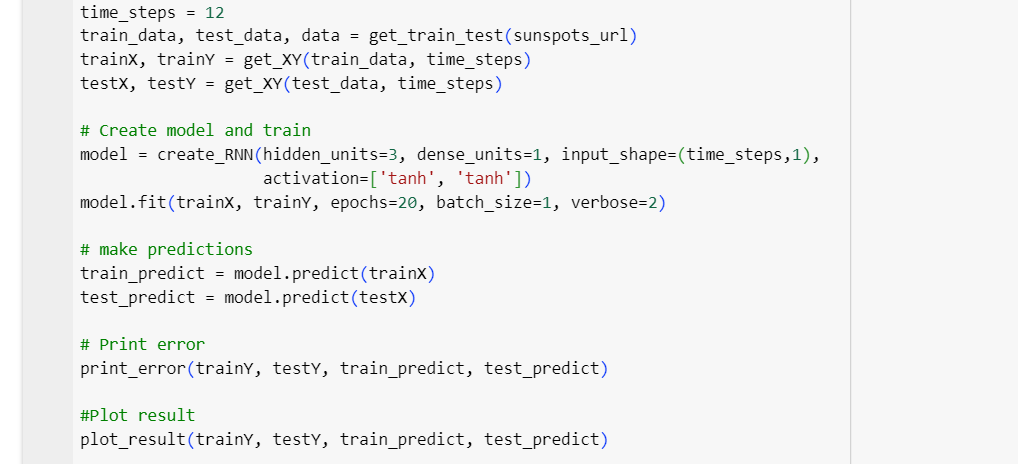




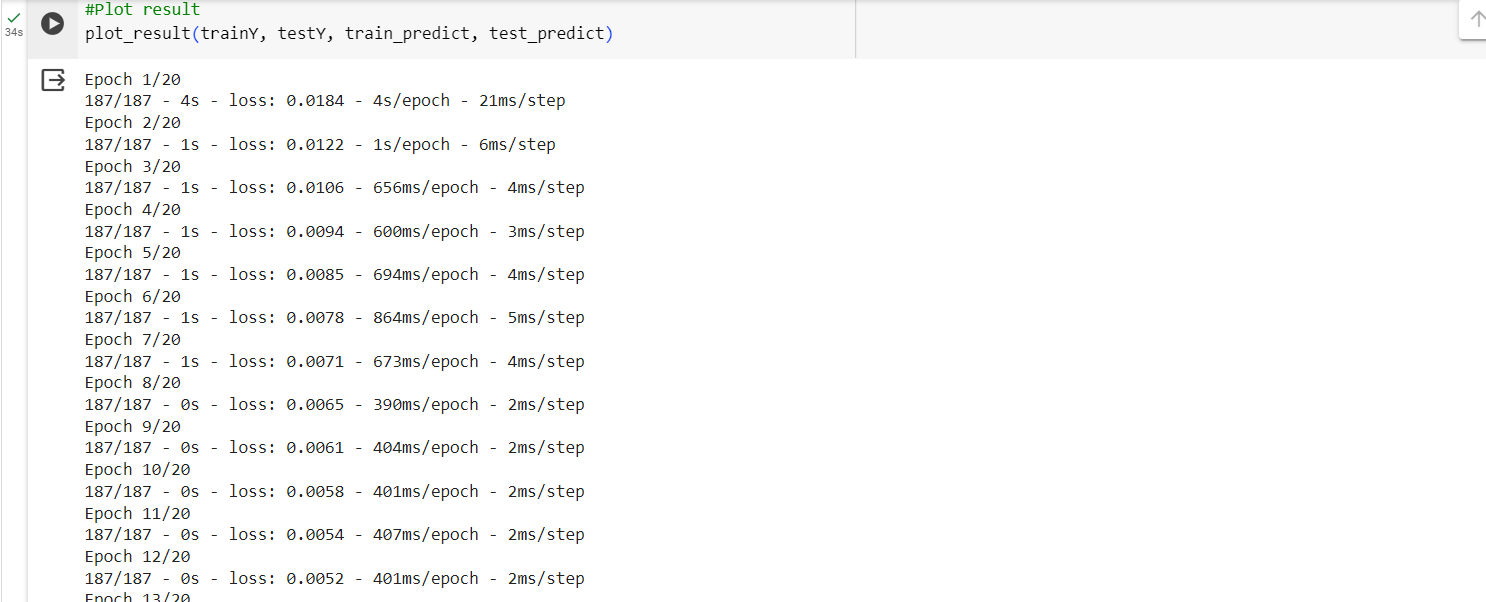


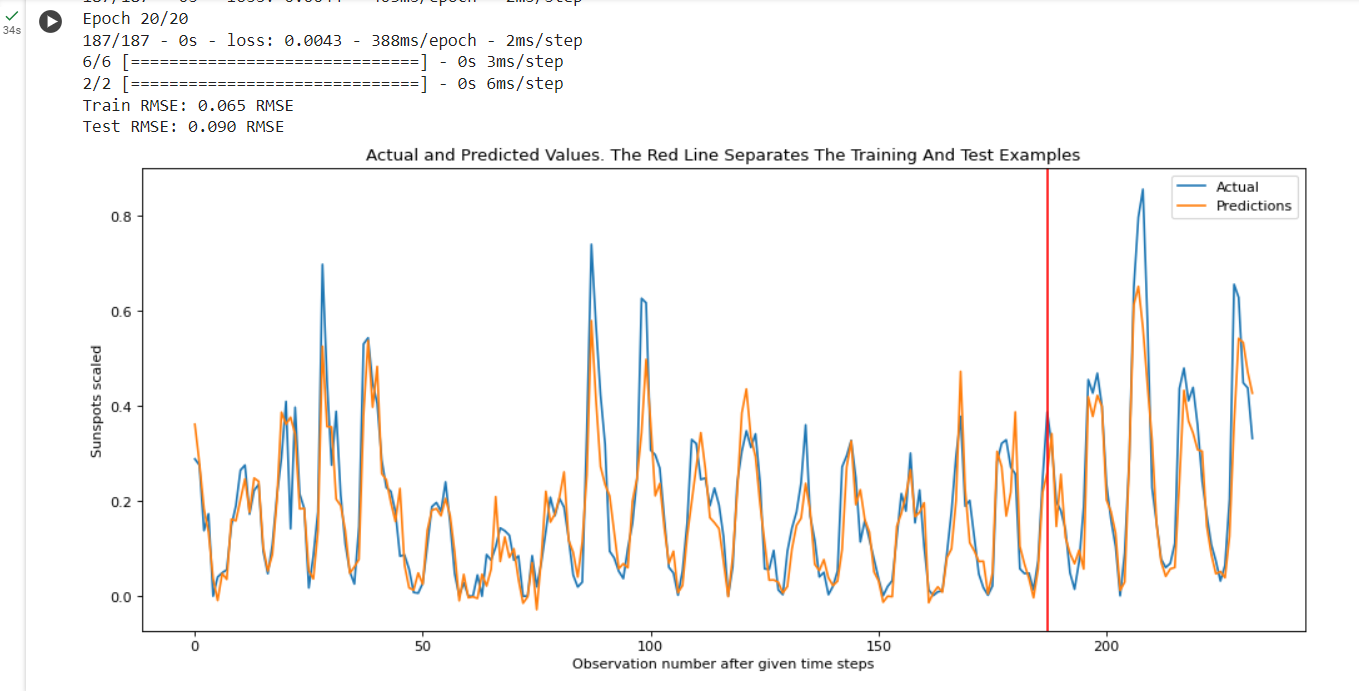






**Result**





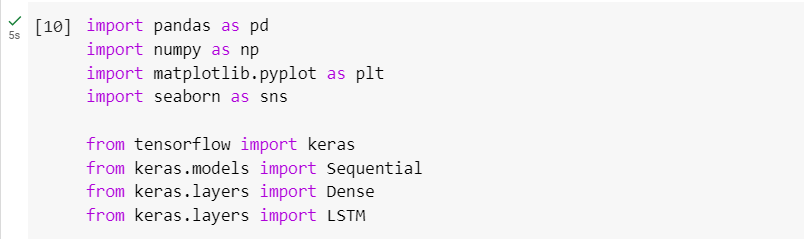
**EXPERIMENT - 7**

## **AIM:** Implement LSTM networks

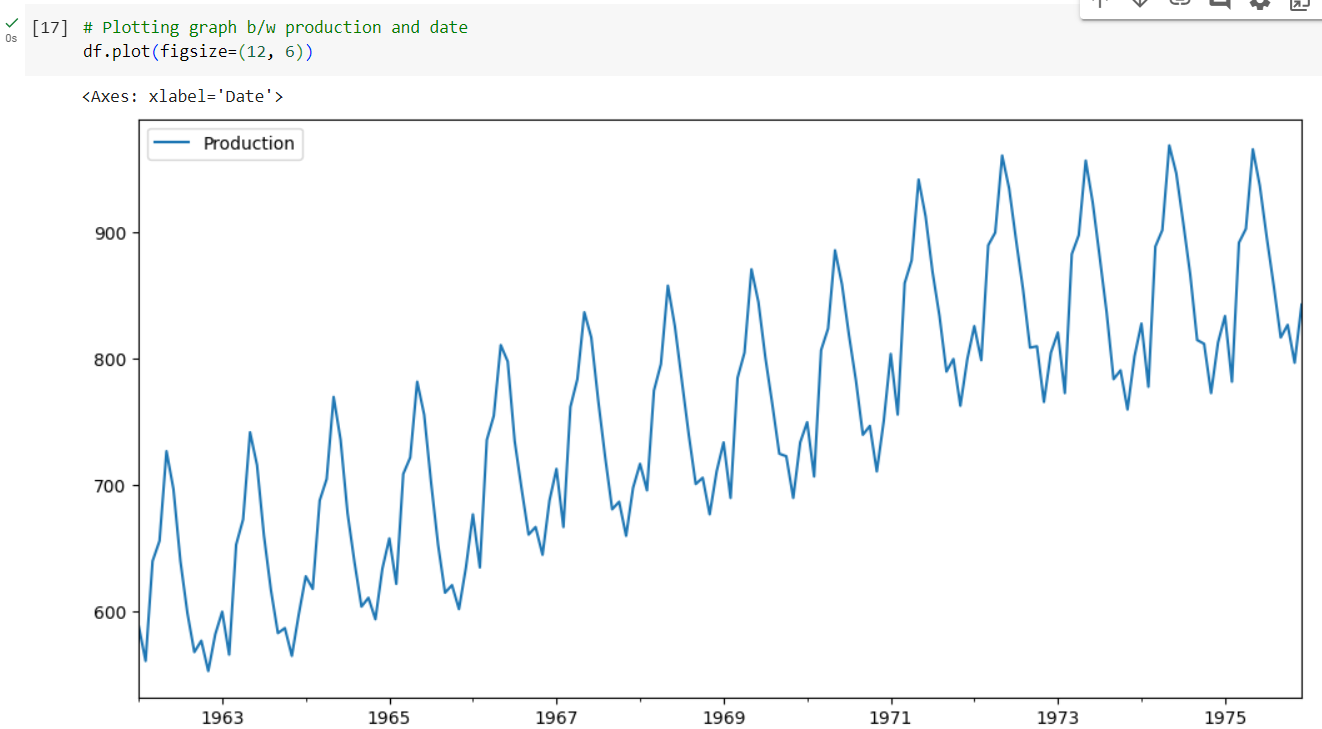
**THEORY:**

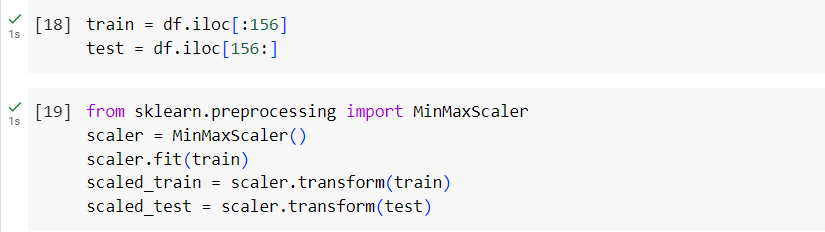
Theory: Long Short-Term Memory is an improved version of recurrent neural network designed by Hochreiter & Schmidhuber. LSTM is well-suited for sequence prediction tasks and excels in capturing long-term dependencies. Its applications extend to tasks involving time series and sequences. LSTM’s strength lies in its ability to grasp the order dependence crucial for solving intricate problems, such as machine translation and speech recognition. A traditional RNN has a single hidden state that is passed through time, which can make it difficult for the network to learn long-term dependencies. LSTMs address this problem by introducing a memory cell, which is a container that can hold information for an extended period. LSTM networks can learn long-term dependencies in sequential data, which makes them well-suited for tasks like language translation, speech recognition, and time series forecasting. LSTMs can also be used in combination with other neural network architectures, such as Convolutional Neural Networks (CNNs) for image and video analysis. The memory cell is controlled by three gates: the input gate, the forget gate, and the output gate. These gates decide what information to add to, remove from, and output from the memory cell. The input gate controls what information is added to the memory cell. The forget gate controls what information is removed from the memory cell. And the output gate controls what information is output from the memory cell. This allows LSTM networks to selectively retain or discard information as it flows through the network, which allows them to learn long-term dependencies.

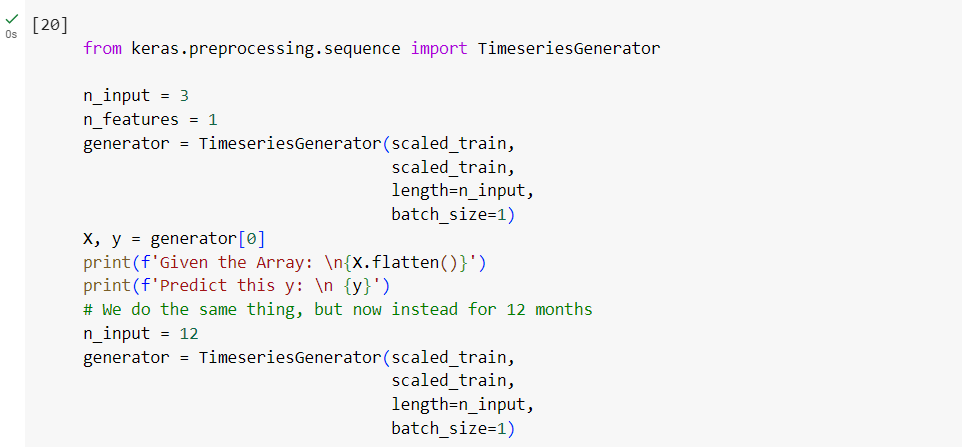
**PROGRAM:**

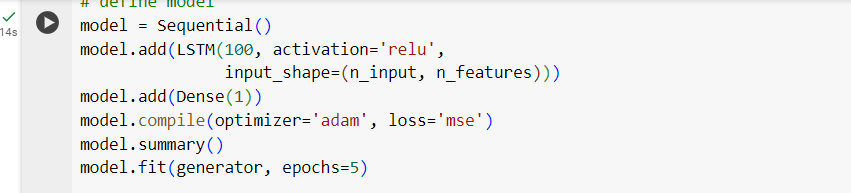




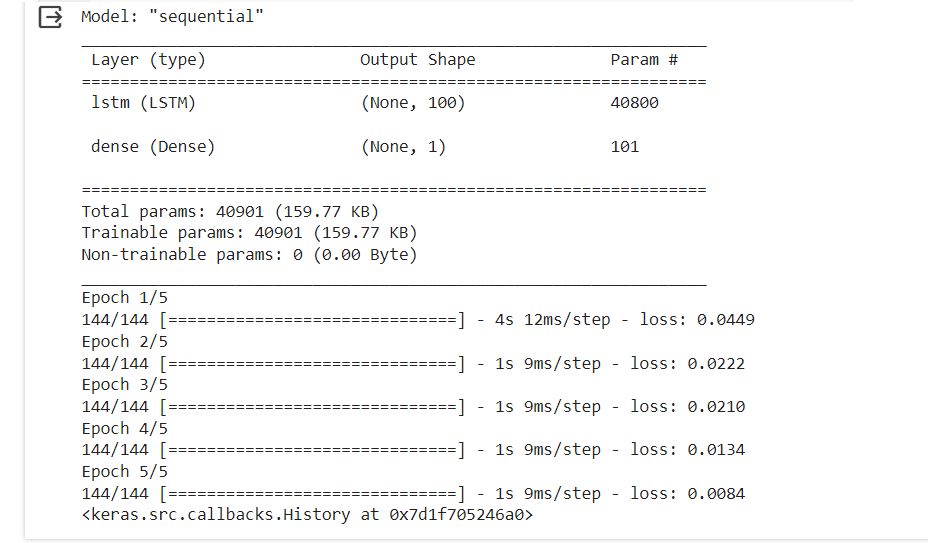








**Result**



**EXPERIMENT - 8**

**AIM:** Perform the sentiment analysis based on data (user choice data) end to end workflow (loading preprocessing modelling compiling).

**THEORY:**

Sentiment analysis is the process of automatically understanding the emotional tone of a piece of text. Machine learning (ML) and deep learning (DL) are powerful tools for this task.

**Deep learning models**, particularly Recurrent Neural Networks (RNNs) like LSTMs, can capture complex relationships between words and account for context better. This leads to more nuanced sentiment analysis, even for informal language.

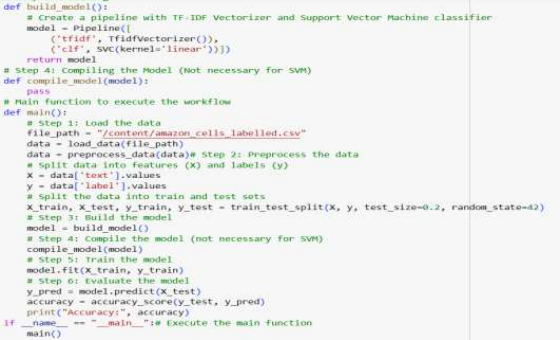
Here's a quick comparison:

* **ML:** Simpler, interpretable models, potentially lower accuracy.
* **DL:** More complex, often black-box models, potentially higher accuracy.

**Deep learning is becoming increasingly popular for sentiment analysis** due to its ability to handle complex language and achieve state-of-the-art results. However, ML approaches can still be valuable for their simplicity and interpretability.

**PROGRAM:**





**Result:**

