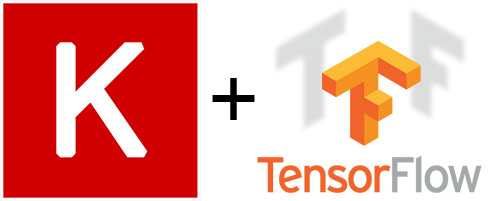
1. What is Tensorflow?

* Free and Open source library specially designed **for machine learning.**
* Designed by google Brain Team.
* Tensorflow is used across a range of tasks but has particular focus on **training and inference of deep neural networks.**
* **Python** is by far the most common language that TensorFlow uses
* TesnsorFlow support running computations on a variety of Types of device**, including CPU, GPU and even mobile operating systems.**

1. Tensorflow API Versions?

* Tensorflow1.x – there were both high-level and low-level APIs that used to be complex and confusing to understand and work with
* TensorFlow2.0 – **Keras(Deep Learning API)** has become an official high-level API. Keras will come along when install TF2.0, hance the integration of Keras with TF does not need any code bride. We can also use Keras code in TensorFlow, Which makes it easy to build something unique.

****

1. **Another ML Library is Py torch –**

* About Py-torch vs vs TensorFlow(Keras)

PyTorch and TensorFlow (with Keras) are two of the most popular deep learning frameworks, and both are widely used for developing and training neural networks. Each has its strengths and weaknesses, and the choice between them often depends on individual preferences, project requirements, and the specific use case. Here are some points to consider when comparing PyTorch and TensorFlow with Keras:

PyTorch and TensorFlow (with Keras) are two of the most popular deep learning frameworks, and both are widely used for developing and training neural networks. Each has its strengths and weaknesses, and the choice between them often depends on individual preferences, project requirements, and the specific use case. Here are some points to consider when comparing PyTorch and TensorFlow with Keras:

1. Ease of Use:

* PyTorch: Known for its dynamic computational graph, PyTorch is considered more "pythonic" and is often praised for its simplicity and ease of use. The dynamic computation graph in PyTorch allows for more flexibility in model building and debugging.
* TensorFlow with Keras: TensorFlow 2.x, with its integration of the high-level Keras API, has become more user-friendly. Keras provides a simple and intuitive interface for building neural networks and is often favored for its ease of use.

1. Computational Graph:

* PyTorch: Utilizes a dynamic computational graph, which is constructed on-the-fly as operations are executed. This makes it easier to work with dynamic structures and is often more intuitive for debugging.
* TensorFlow with Keras: Historically, TensorFlow used a static computational graph. However, with TensorFlow 2.x and eager execution, dynamic graphs are also supported, making it more similar to PyTorch in this regard.

1. Community and Ecosystem:

* PyTorch: Has gained popularity in both research and industry and has a strong presence in the research community. It is known for its active development community and a rich ecosystem of libraries.
* TensorFlow with Keras: TensorFlow has been widely adopted in industry, and it has a large and mature ecosystem. It has strong support for deployment in production environments, and TensorFlow Serving is often used for deploying models.

1. Model Deployment:

* PyTorch: Traditionally, PyTorch has been considered more suitable for research and prototyping, but deployment tools like TorchServe are available for deploying PyTorch models.
* TensorFlow with Keras: TensorFlow has a more mature set of tools for deploying models in production, including TensorFlow Serving, TensorFlow Lite, and TensorFlow.js.

1. Popularity:

* Both PyTorch and TensorFlow are very popular, and the choice between them often comes down to personal preference and the specific requirements of the project.

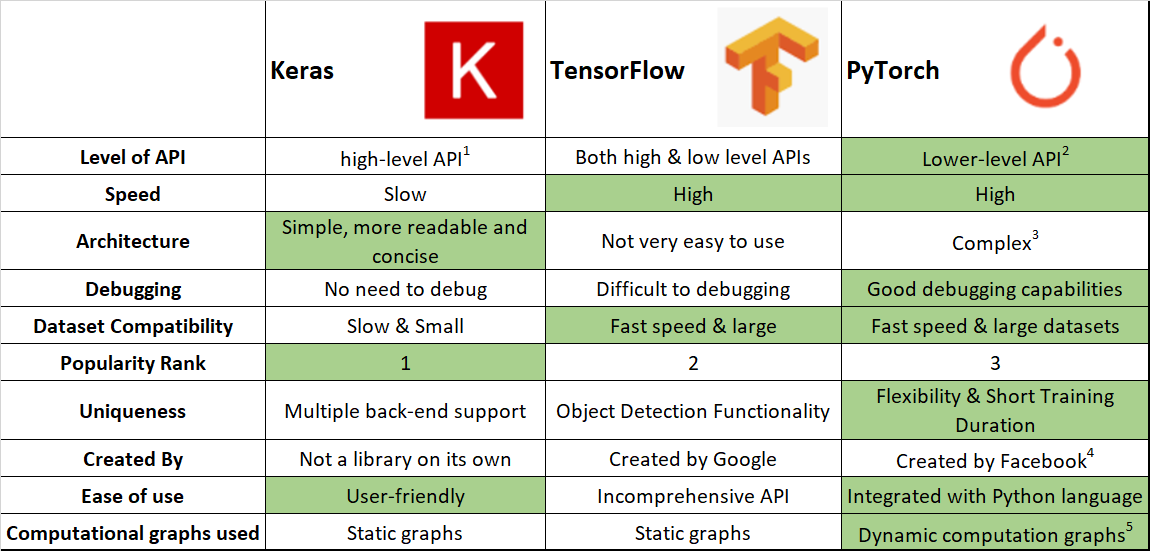
1. Compatibility:

* PyTorch: Known for its "eager execution" mode, which allows operations to be executed immediately as they are called, making it easier to understand and debug code.
* TensorFlow with Keras: TensorFlow 2.x has adopted eager execution by default, providing a more dynamic and intuitive development experience similar to PyTorch.

1. Integration with Other Libraries:

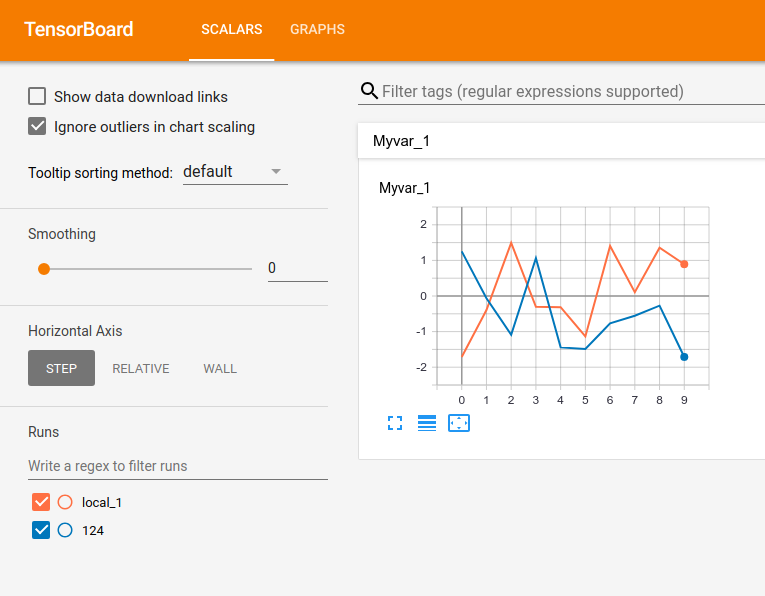
* Both frameworks have good integration with popular machine learning and deep learning libraries.

In summary, both PyTorch and TensorFlow (with Keras) are powerful and capable frameworks, and the choice between them often comes down to personal preference and specific project requirements. Many practitioners use both frameworks depending on the context or the preferences of their team.

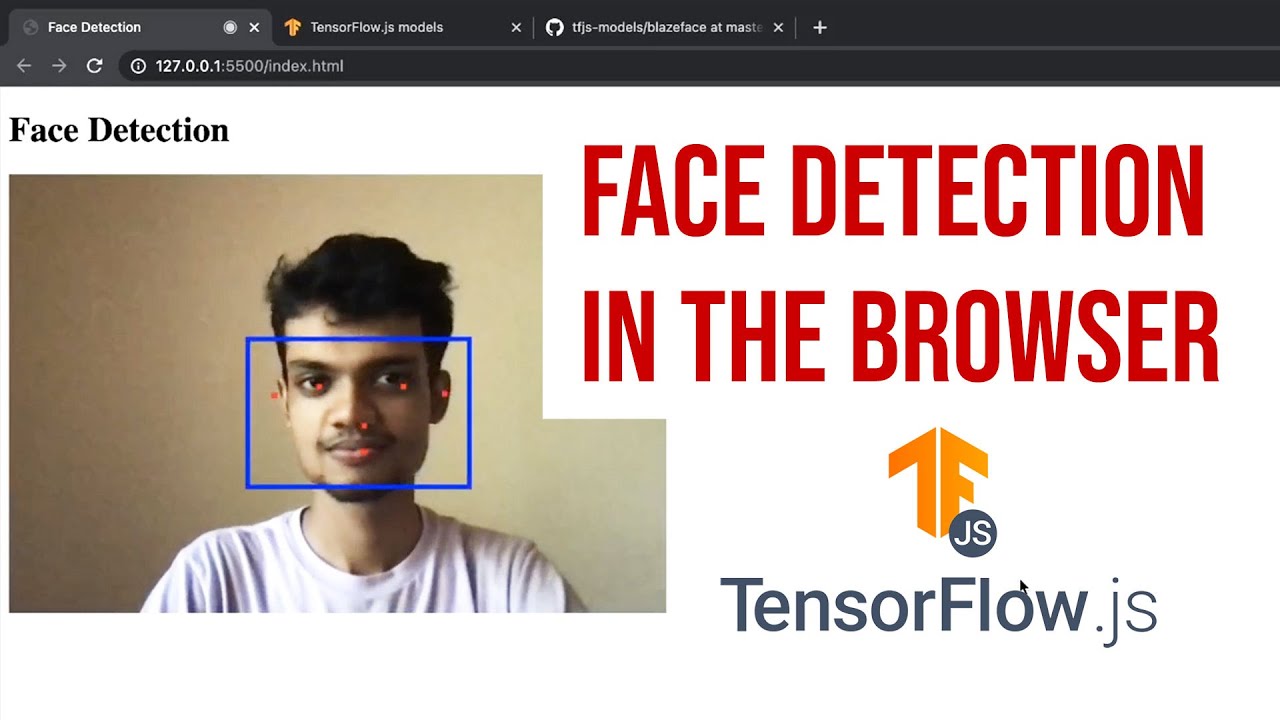


1. Why TensorFlow?

* Easy model building – Build and train ML models easily using intuitive high level APIs like Keras with eager execution, which makes for immediate model iteration and easy debugging.
* Robust ML production anywhere – Easilly tarin and deploy models in the cloud, on-prem, in the browser or on-device on matter what language you use.
* Powerful experimentation fro research – simple and flexible architecture to take new ideas from concept to code, to state of the art models and to publication faster.
* Using tenserboard can visualize the data.

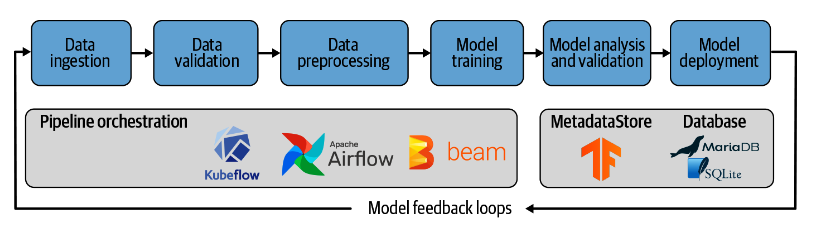


* Can devlop that model – For develop mobile application, web application, using TF lite can build mobile OS systems.
* Learing foundation of TF, go to experts level and you can create your own ML project.
* **Javascript** - Use TF.js to create new ML models and deploy existing model with javascript.



For Mobile and IOT – TF lite on mobile and Embedded devices like Android, IOS, Edge TPU and Raspberry Pi.

For Production – Deploy a production-ready ML pipeline fro training and interface using TF Extended(TFX)

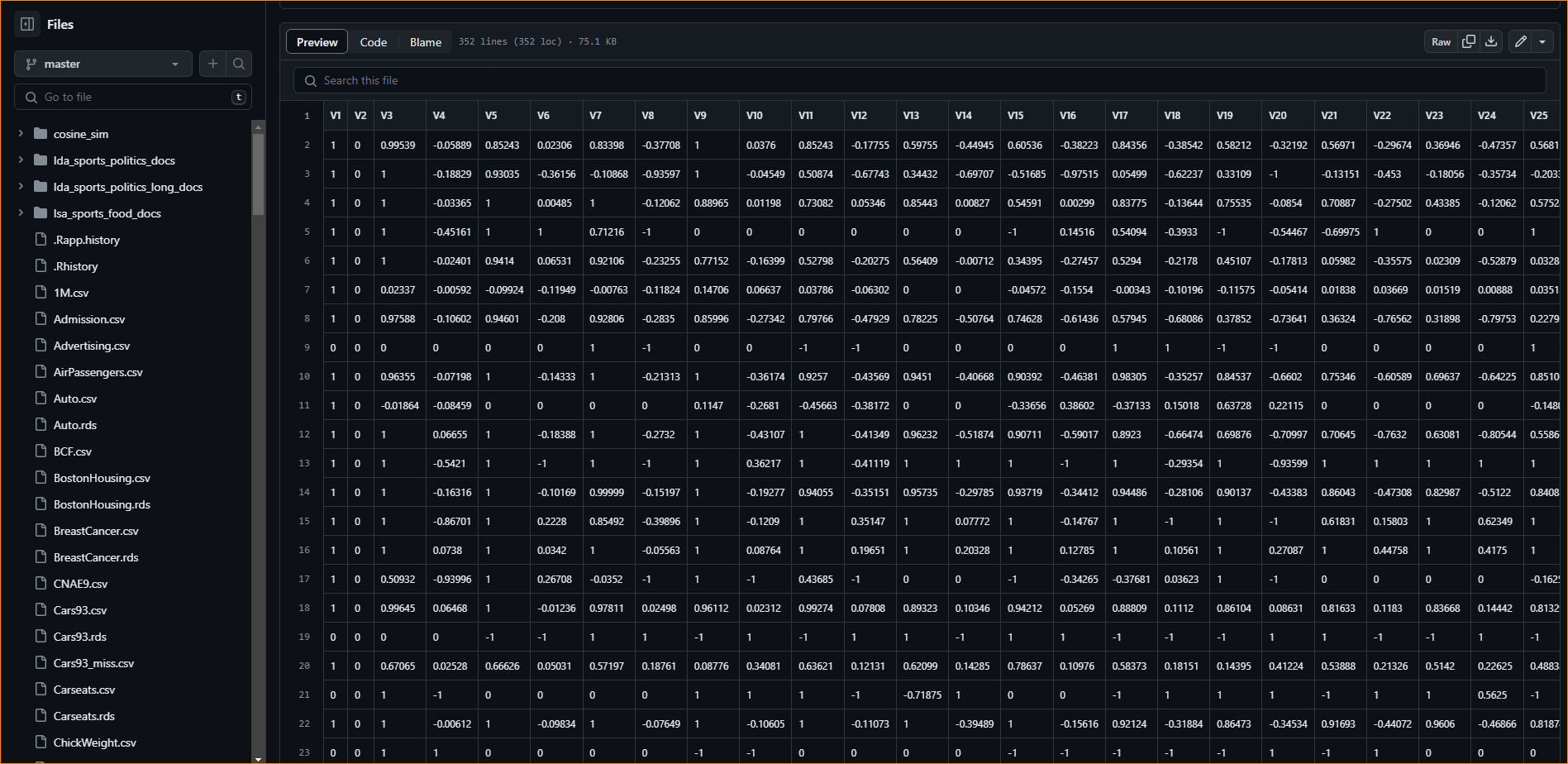


How to build ANN Model TF library -

Exercise - **Tarin Artificial neural Network(ANN) model using tensorFlow**

* We use open source tabular dataset- lonosphere dataset

<https://github.com/selva86/datasets/blob/master/Ionosphere.csv>



This has 34 independent variable

According to the dataset result the encoded Good = 1 and Bad = 0

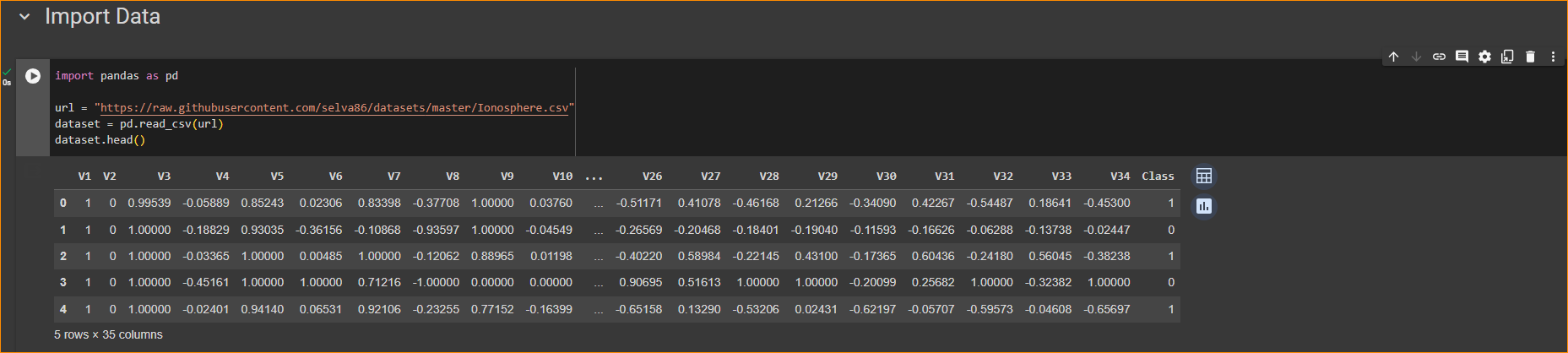


For the implementation Use Google CO-Lab

Steps-

1. Import data to model

Using pandas we arrange data set from github repository



1. Identify the dependent and Independent variable

34 independent variables are there,

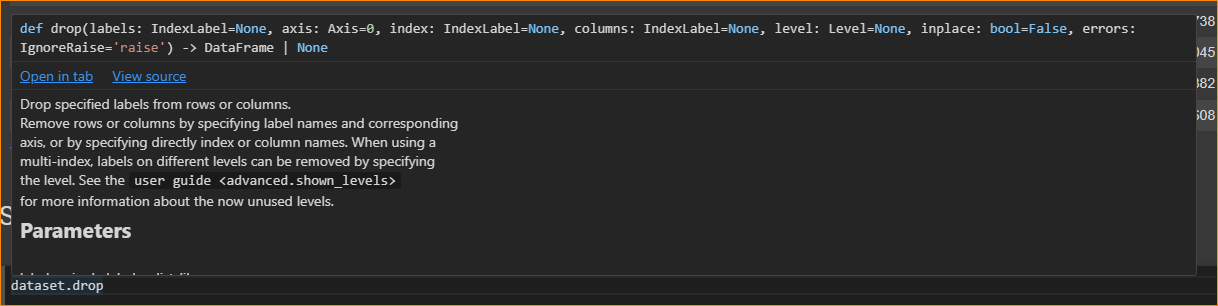
**V1 to V34 – Independent variable**

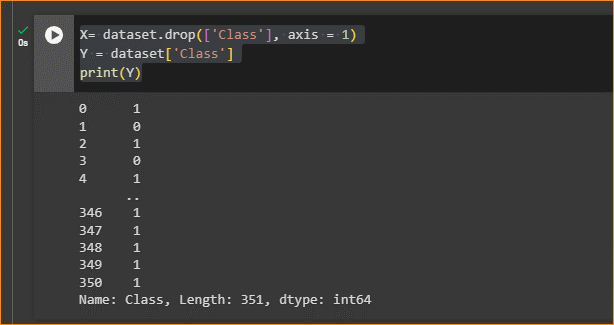
Base on these variable – **Class is the dependent variable**

1. Split dependent and Independent variable

Use drop method to get independent variable-

*Drop-*





Indepentant variable = X

Dependent variable = Y

Explain Code -



This line creates a new Data Frame X by removing the 'Class' column from the original dataset. The axis=1 parameter specifies that you are dropping a column (as opposed to dropping a row, which would be axis=0).



This line creates a new Series Y by selecting only the 'Class' column from the original dataset.

1. When training the model using ML, we want to **split our data into Training and Testing data**

* Training data use to train our model,
* Testing data use to Test the model Whether it works checking.

Use an Independent set of variables as our test and Training data

Y= dependent variables

X= independent variables

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| V1 | V2 | … | … | … | … | V34 | Class |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

Y test

Y train

X test

X train

* Green Box – Training data
* Purple box – Testing data

Why testing data – Check model train well or not, over fitting or not

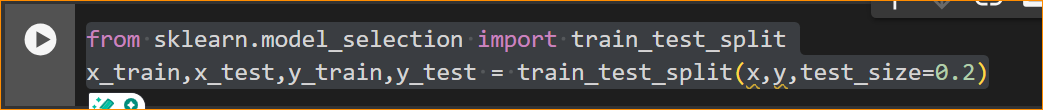
1. How to split- Scikit-Learn

Note-

* Scikit-Learn (or sklearn) is a popular machine learning library for the Python programming language. It provides simple and efficient tools for data analysis and modeling, including various machine learning algorithms for classification, regression, clustering, dimensionality reduction, and more.
* Scikit-Learn is built on NumPy, SciPy, and Matplotlib, and it is designed to work seamlessly with other libraries in the Python ecosystem. It offers a consistent and user-friendly interface for various machine learning tasks, making it a go-to choice for both beginners and experienced practitioners.

train\_test\_split – help to split the dataset into separate

In here How much percentage for Testing data?



Note –

It looks like you're using the train\_test\_split function from the scikit-learn library in Python to split your data into training and testing sets. This is a common step in machine learning workflows to evaluate the performance of a model.

Here's a breakdown of the code:

x and y are your feature matrix (independent variables) and target variable (dependent variable), respectively.

**test\_size=0.2 indicates that 20% of the data will be used for testing, and the remaining 80%** will be used for training.

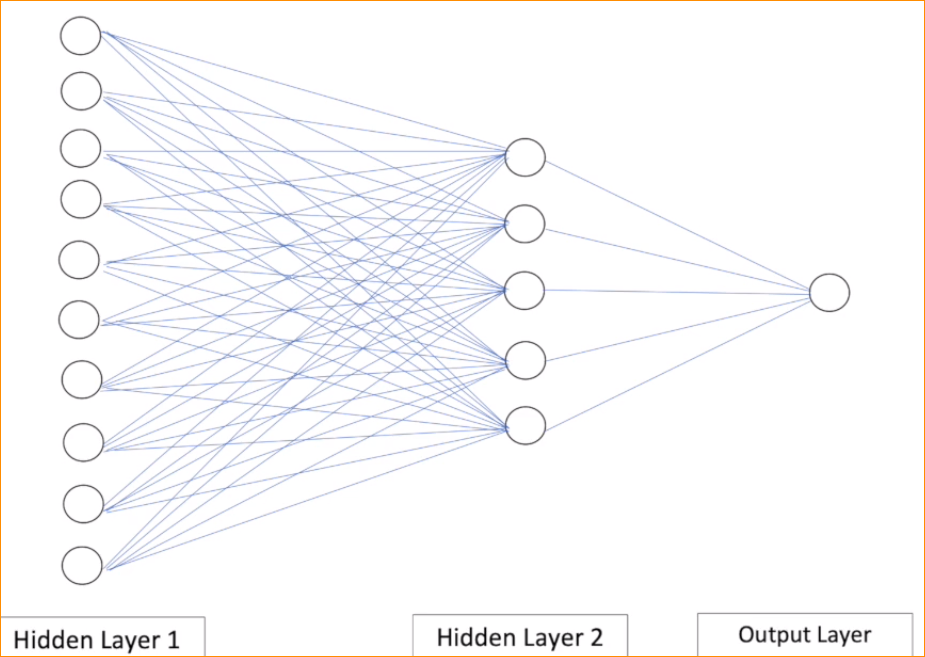
After running this code, you'll have four sets of data:

* x\_train: The features used for training the model.
* x\_test: The features used for testing the model.
* y\_train: The corresponding target values used for training.
* y\_test: The corresponding target values used for testing.

You can then use these sets to train your machine learning model on the training data (x\_train and y\_train) and evaluate its performance on the testing data (x\_test and y\_test). This helps you assess how well your model generalizes to new, unseen data.

1. Implement Model –

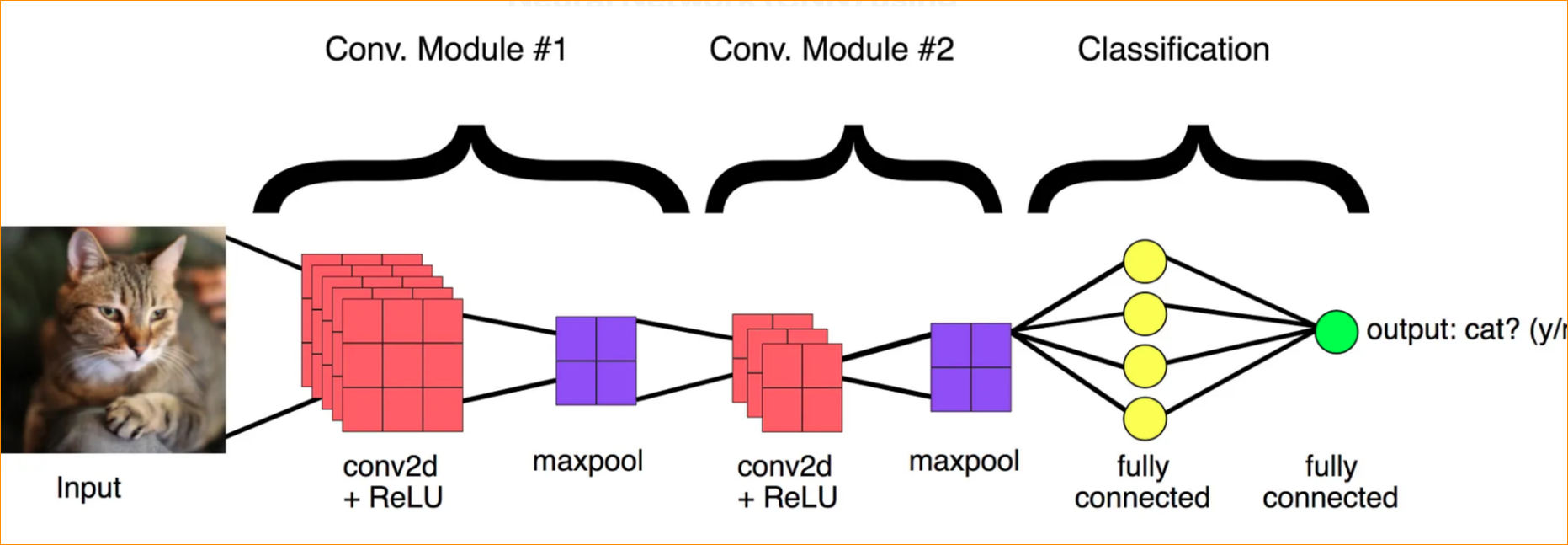
Create a ANN with 2 hidden layers, 1 st layer have 10 neurons and second have 5 neurons.



If we use Google Co-Lab no need TenserFlow. – Already Installed

If you Use conda environment in local machine- Should install TenserFlow

How to Train Model Using layers –



**Note –**

1. **Install Dependences**



1. **Create a Sequential model**



1. **Add a Dense layer with 64 units, input shape of (input\_dim,), and activation function 'relu'**



1. **Add a Dropout layer with a dropout rate of 0.5**



1. **Add another Dense layer with 32 units and activation function 'relu'**



1. **Add another Dropout layer with a dropout rate of 0.5**



1. **Add the output layer with the appropriate number of units and activation function**



1. **Compile the model**
2. model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

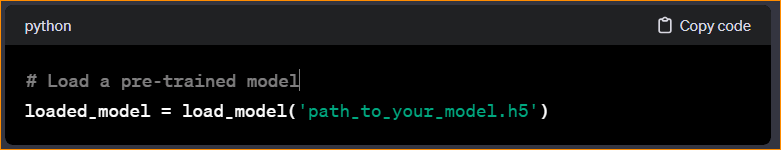


1. **Print a summary of the model architecture**



Make sure to adjust the parameters such as input\_dim and output\_dim according to your specific problem. Also, modify the activation functions, optimizer, and loss function based on the nature of your task (e.g., regression, binary classification, or multi-class classification).

* If you have a pre-trained model saved, you can load it using load\_model:

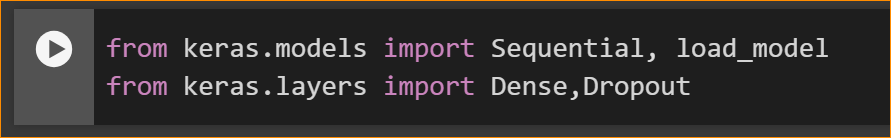


Replace 'path\_to\_your\_model.h5' with the actual path to your saved model file.

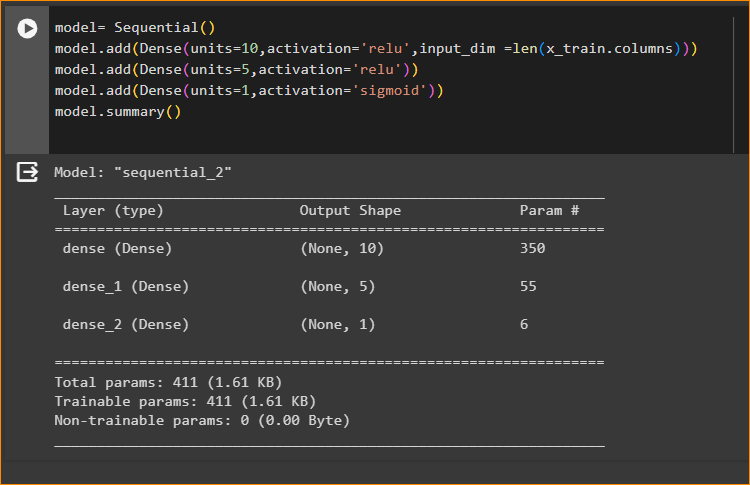
Remember to have your data prepared and preprocess it accordingly before training the model.

Make Our Model-

Import TenserFlow Keras-



Use Sequential method to Stack to the layers.



* Add 1 st layer – Dence layer

In here we are using "Relu" Activation Function - Normally hedding layers we are using Relu Activation Function. Because the values coming from the top layers. (- values that assigning to 0)

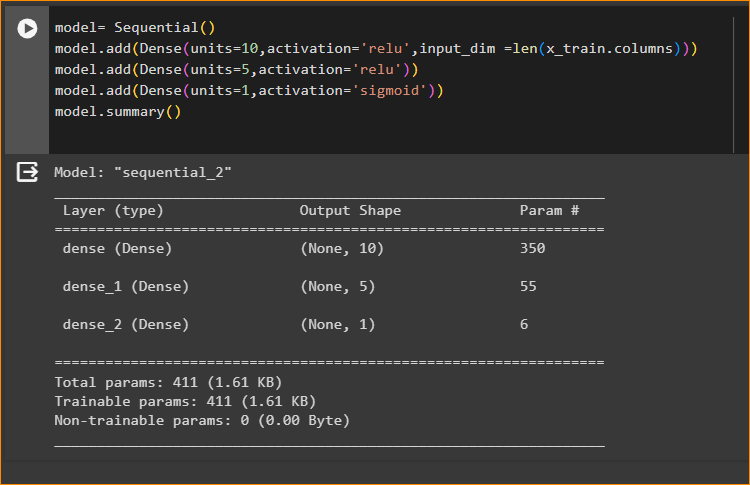
Input dimensions – number of input that we have, in here the length is 34(Independent variable count) – In here X-Train coloums, this is my input diementions. V1-V34

* Add 2nd hidden layer

Have 5 neurons , same activation function

* And output Layer – Ouput 1 or 0, in here Activation function mean ‘sigmoid’
* Get Summary – model.summary()

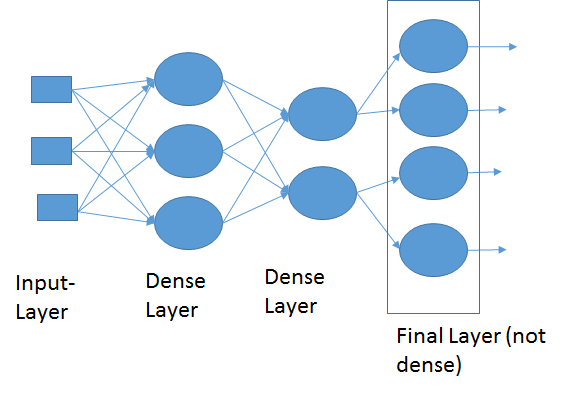
Result -



Note -

What is dance layer?

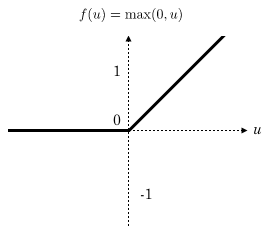
Note



Dense Layer is simple layer of neurons in which each neuron receives input from all the neurons of previous layer, thus called as dense. Dense Layer is used to classify image based on output from convolutional layers. Working of single neuron. A layer contains multiple number of such neurons.

What is “RELU” Activation function?

* ReLU, which stands for Rectified Linear Unit, is an activation function commonly used in artificial neural networks, including deep learning models. The function is defined as
* f(x)=max(0,x), which means it returns zero for all negative input values and for positive values, it returns the input value itself.
* ReLU has become a popular choice in many deep learning architectures due to its simplicity and effectiveness. However, it's worth noting that ReLU is not without its limitations. One drawback is that neurons using ReLU can sometimes become "dead" during training, where they consistently output zero and stop learning. This issue has led to the development of variants such as Leaky ReLU and Parametric ReLU, which aim to address the problem of dead neurons by allowing small, non-zero gradients for negative inputs.



What is sigmoid activation function?

The sigmoid activation function, also known as the logistic sigmoid function, is a type of activation function commonly used in artificial neural networks. It's particularly popular in the output layer of binary classification models, where the goal is to predict probabilities that lie between 0 and 1.



In this equation,

e is the base of the natural logarithm (approximately 2.71828), and

x is the input to the function. The output

σ(x) lies in the range (0, 1), and it can be interpreted as the probability that the given input belongs to the positive class in binary classification problems.

1. Now Compile our model –

This is not a multi class classification, this is binary classification So need to have loss we count, that model properly train or not. Therefore Check ‘binary\_Creossentropy’.

For the optimizer I used ‘sgd’

And the matrixs evaluate the accuracy.



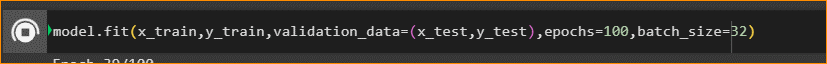
Note -

What is the use of SGD Optimizer?

ML | Stochastic Gradient Descent (SGD) - GeeksforGeeks

Stochastic Gradient Descent (SGD) is a variant of the Gradient Descent algorithm that is used for optimizing machine learning models. It addresses the computational inefficiency of traditional Gradient Descent methods when dealing with large datasets in machine learning projects.

1. Now Training the Model



Give the training data separately – x\_train, y\_train

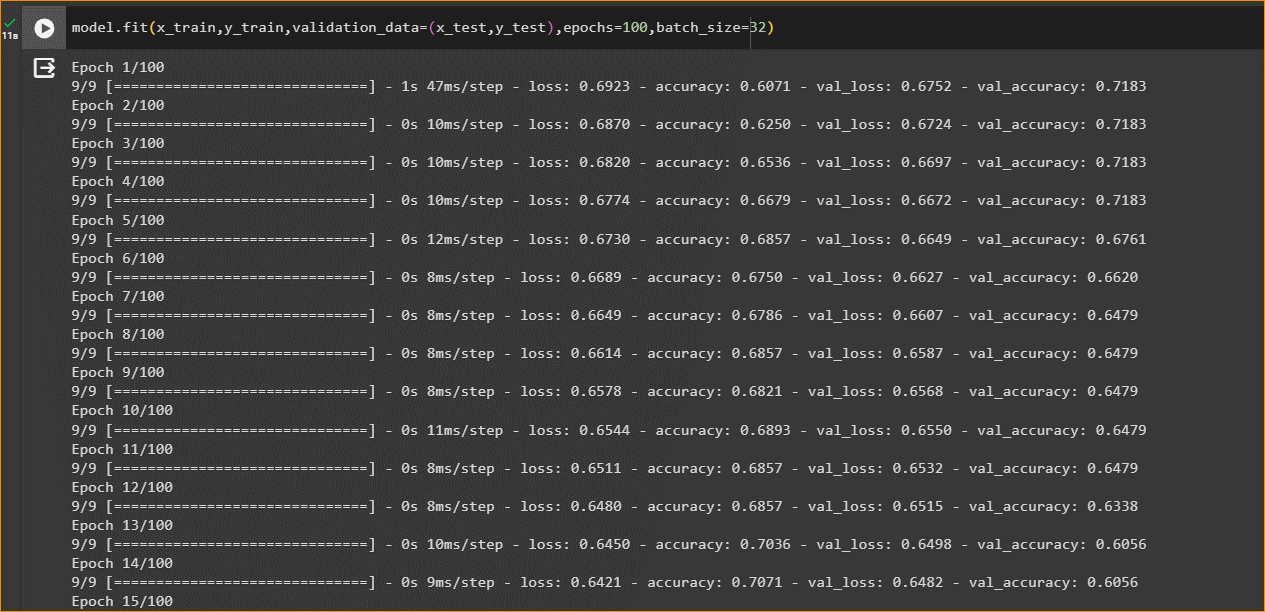
And Add the validation data. Test model - (X-test, Y-test)

Give number of epochs to train the model- number of time data should be train again and again.

Batch size mean total number of training samples present in a single batch- 32

Now model is going to train my data set-

Epochs 100 mean how many time that dataset should be train on this model. (how many cycles)



In here 9/9 mean – number of iteration (number of batches that need to completed.)

Total number of rows that are available in our traning dataset – Trainning data

Tranning data / 32 = 9

End – of model

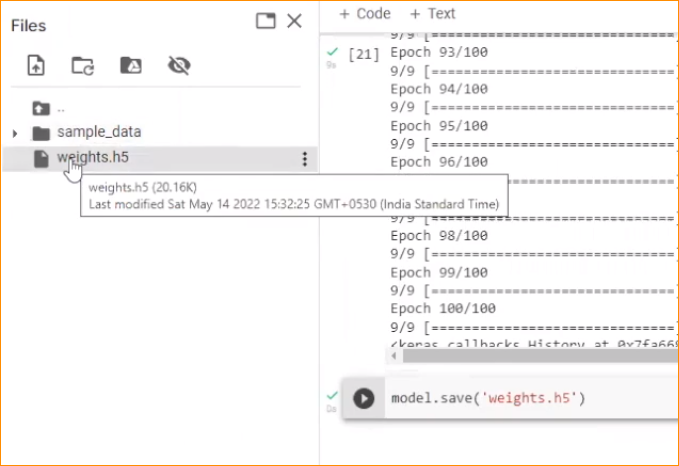


In here finally validation Accuracy is – 0.8310

**Validation Accuracy**: The validation accuracy is a metric that measures the percentage of correctly classified instances in the validation dataset. It is calculated as the number of correctly predicted instances divided by the total number of instances in the validation dataset.

A validation accuracy near 1 (or 100%) typically indicates that the model is performing very well on the validation dataset. In the context of classification tasks, a validation accuracy of 1 means that the model is correctly predicting all instances in the validation dataset. This is a desirable outcome, as it suggests that the model has learned to generalize from the training data to unseen data, making accurate predictions.

1. Save this waights



1. Using this model we can get a prediction- So use that 34 variable in to this model and get prediction