Implementation of Deep Learning in UiPath

**Overview:**

[RPA](https://www.sap.com/cmp/dg/crm-ya18-bem-micrrpa/index.html?url_id=text-global-digitalist) is an innovative technology, automating business processes using software robots that perform tedious and repetitive tasks – error susceptibility equals zero.

In this project implementation we are making robots more intelligent by moving one step further i.e. integrating the bots with Deep Learning Algorithms. Connecting Deep Learning with RPA is meaningful whenever business automation is pursued in an integrated and strategic manner. In order to automate business processes efficiently, future-oriented, and strategically, combining these two technologies is essential.

By infusing intelligence into RPA, hence combining machine learning capabilities with process automation, we can design an advanced form of RPA – a bot that is able to analyze, comprehend, and draw conclusions from both structured and unstructured data. This powerful symbiosis is consequently able not just to process, but effectively use data. This newly created **intelligent RPA** analyzes data before acting on it, continuously learns from data, becomes more intelligent over time, and makes smart decisions based on previous learning.

Therefore, automating processes with the help of RPA and ML especially makes sense whenever a huge amount of data needs to be processed, analyzed, compared, and structured. While ML covers the task of thinking and learning, RPA executes. ML functionalities that come to play in connection with RPA are technologies such as image and speech recognition or document information extraction, for example.

**Technology**: UiPath, C#.NET

**Library**: Keras.NET, Numpy.NET

**Project Implementation:**

1. **General Overview:**

In this project development, I have integrated a simple Deep Learning Neural Network in Robotic Process Automation where the robot helps in training the model, evaluate the model and predict the desired output. During the development of the model, I have built custom activities for UiPath software, which predicts the regression value from input data.

1. **How to achieve this?**

To achieve this, I have built different reusable activity are as follow:

* Import or Prepare the Dataset
* Build the Neural Network by adding dense layers of Neurons.
* Perform Batch Normalization on the Input batches of data.
* Introduce Dropouts in the network to overcome Overfitting.
* Compile the Model by assigning the loss function, optimizers.
* Train the Neural Network Model
* Evaluate the model
* Predict or Infer the model

**Step 1: (Prepare the Dataset)**

In this step, we will read the train input data and test input data, and then split that data into X\_train, Y\_train and X\_test and Y\_test. The input data file should be in (.csv) and values must be in integer format. X\_train and Y\_train will be used for training purpose and X\_test and Y\_test will be used for evaluation purpose.

This Prepared Dataset activity will read

Input Parameter:

* Training Data(File path to Training Data)
* Test Data (File path to Training Data)
* Delimiter
* Input Column(Input Features)
* Output Column(Predicting Feature)

Output Parameter:

* X\_Train
* Y\_Train
* X\_Test
* Y\_Test

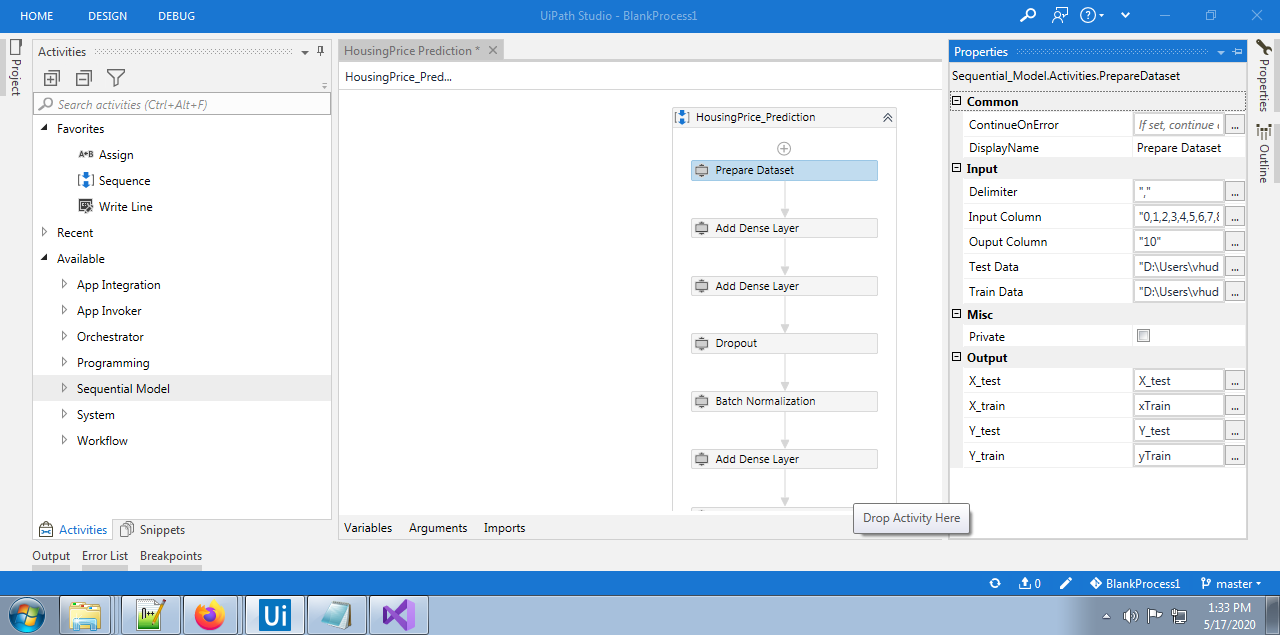


Fig- Preparing Dataset Activity

**Step 2: (Build the Neural Network)**

To create a model we will use Add dense layer activity and we will add multiple dense layers to the Neural Network with different activation functions, filter size and input shape.

Input Parameter:

* Activation Function (e.g.- “relu”)
* Filter Size (e.g.- 32 or 64)
* Input Shape(e.g.- 6,10,20)

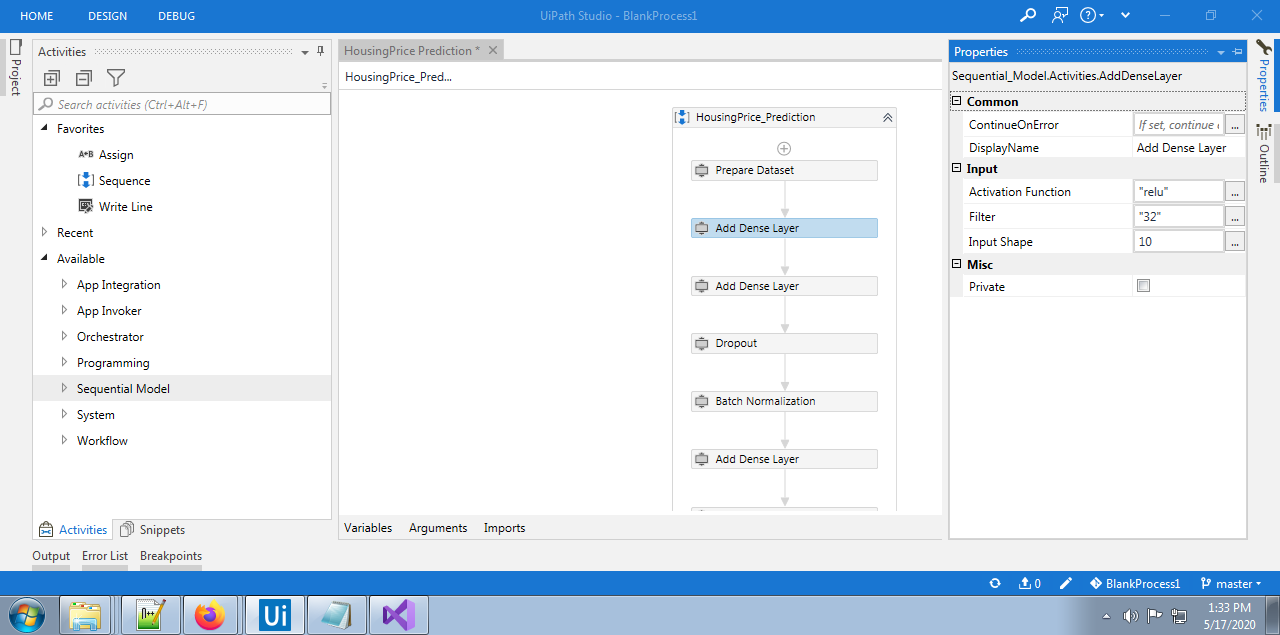


Fig- Add Dense Layer

**Step 3: (Batch Normalization**)

**Batch normalization** (also known as **batch norm**) is a technique for improving

the speed, performance, and stability of [artificial neural networks](https://en.wikipedia.org/wiki/Artificial_neural_network). The Batch

Normalization activity will helps to normalize the batch size of the input data.

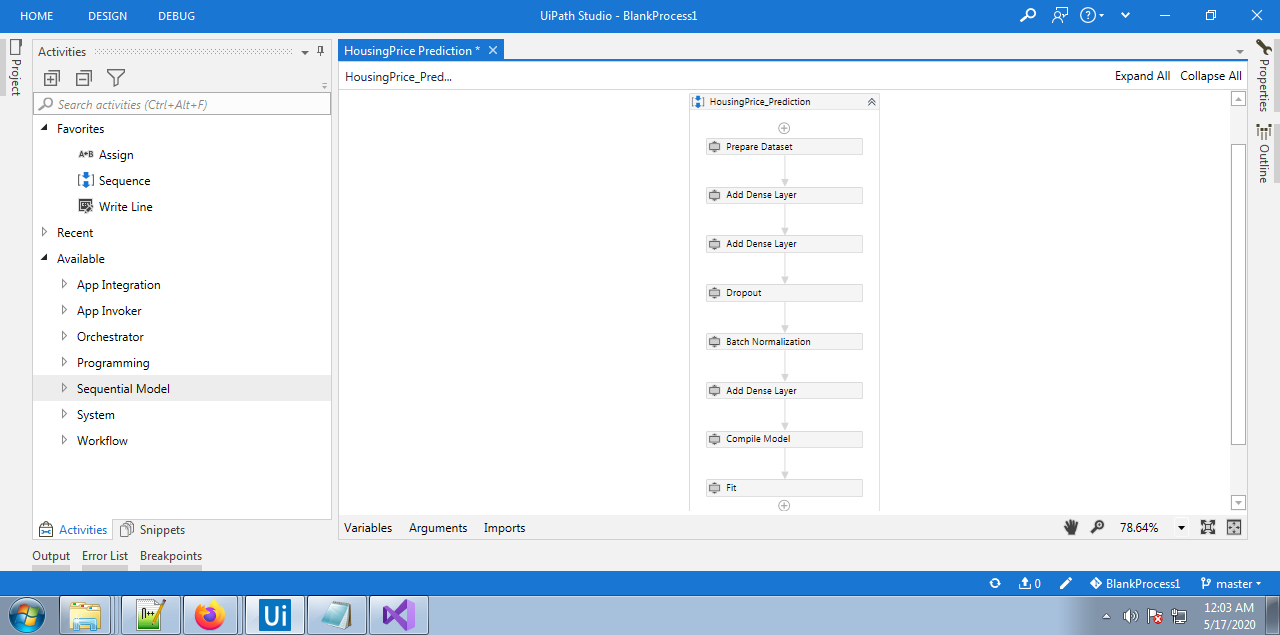


Fig. Batch Normalization

**Step 4: (Adding Dropout Layer**)

Dropout is a regularization method that approximates training a large number of neural networks with different architectures in parallel. It is technique used to improve over-fit on neural networks, dropping out units (both hidden and visible) in a neural network.

Input Parameter:

* Rate (e.g.- 0.2, 0.25 or 0.3)

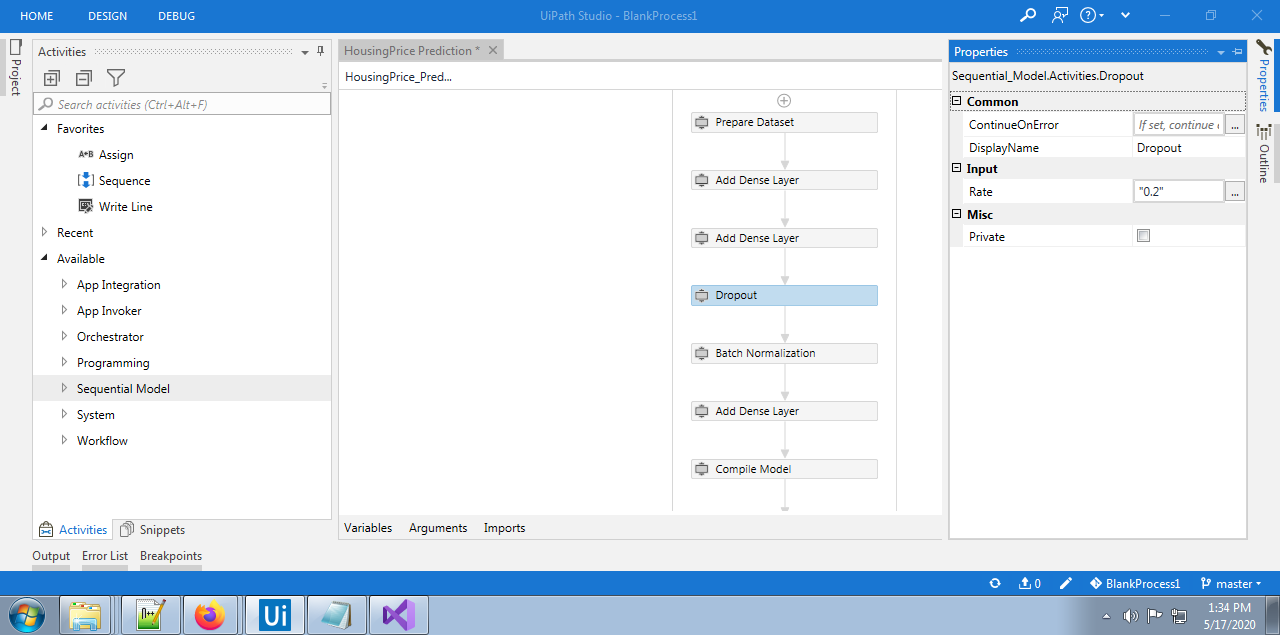


Fig. Dropout

**Step 5: (Compile the Model**)

This activity will help us to compile different layers of the Neural Network and show the summary of the model. The compile method defines the loss function, the optimizer and the evaluation metrics.

Input Parameter:

* Learning Rate
* Optimizer
* Loss Function
* Evaluation Metrics(e.g.- accuracy, RMSE)
* Model Path(Path where we want to save the model )

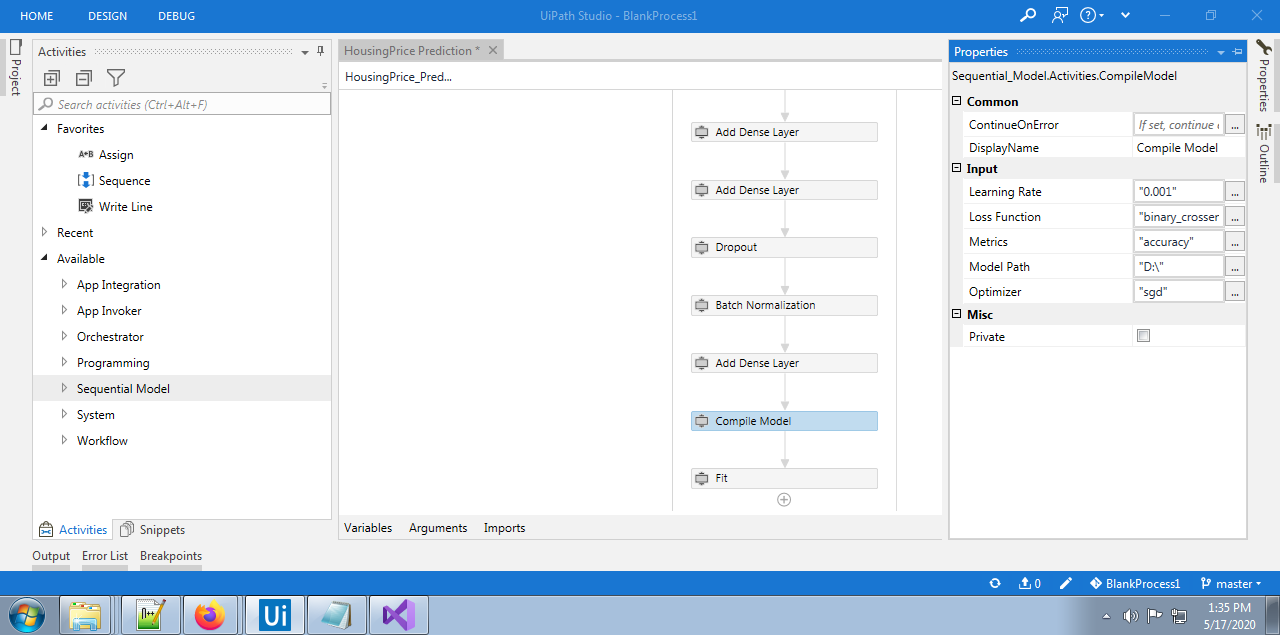


Fig. Compile Model

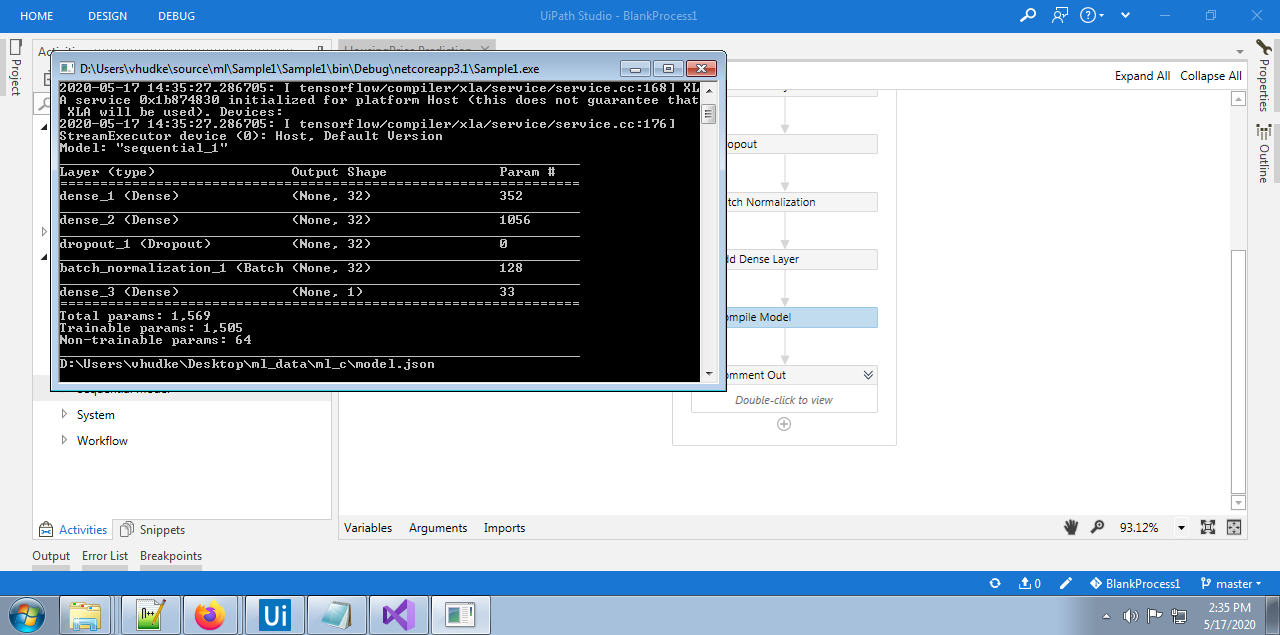


Fig. Compile Model (during execution)

**Step 6: (Fit the Model**)

This activity actually will takes the input data and train that data based on different Epochs, Batch size and compile model.

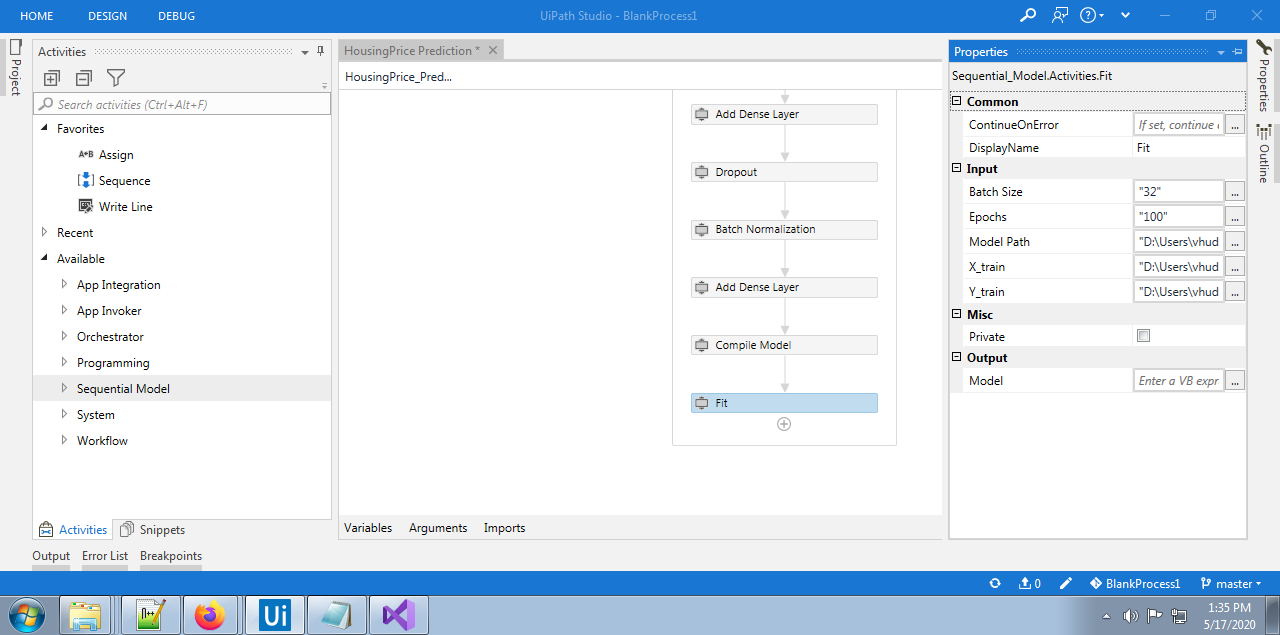


Fig. Fit (during Initialization)

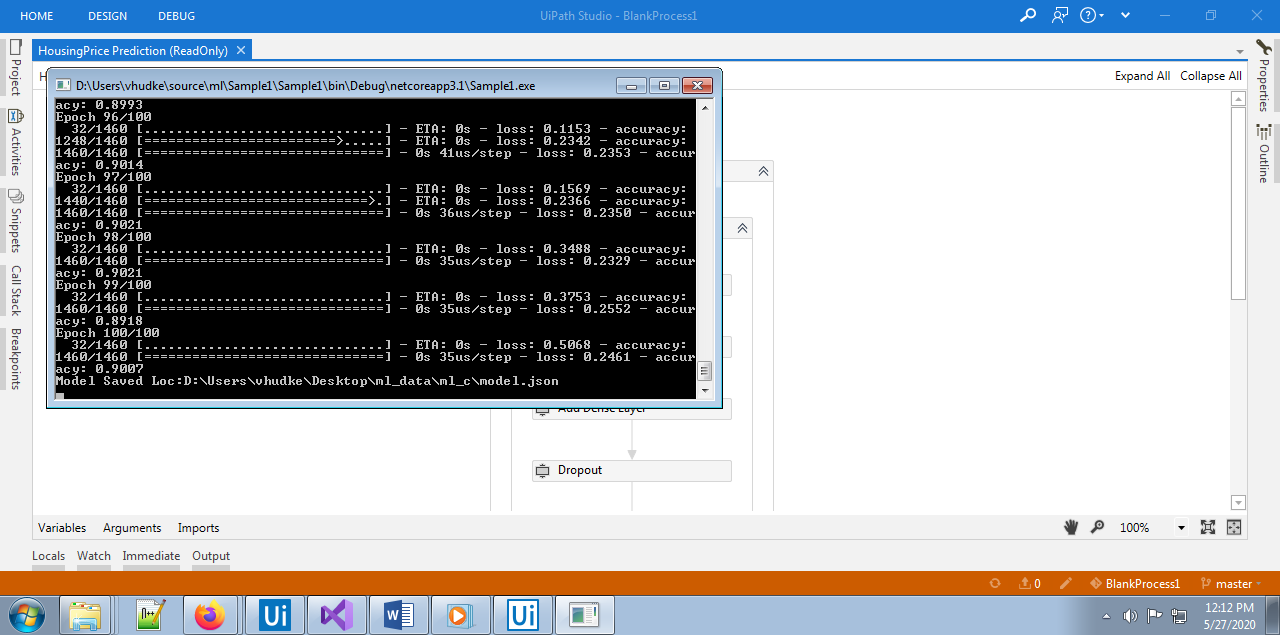


Fig. Fit (during execution)

(Note: We are working on ‘reduce the losses and ‘increase the accuracy’)

**Step 7: (Evaluate the Model**):

This activity actually will takes the input data and evaluate that data based on trained model.

