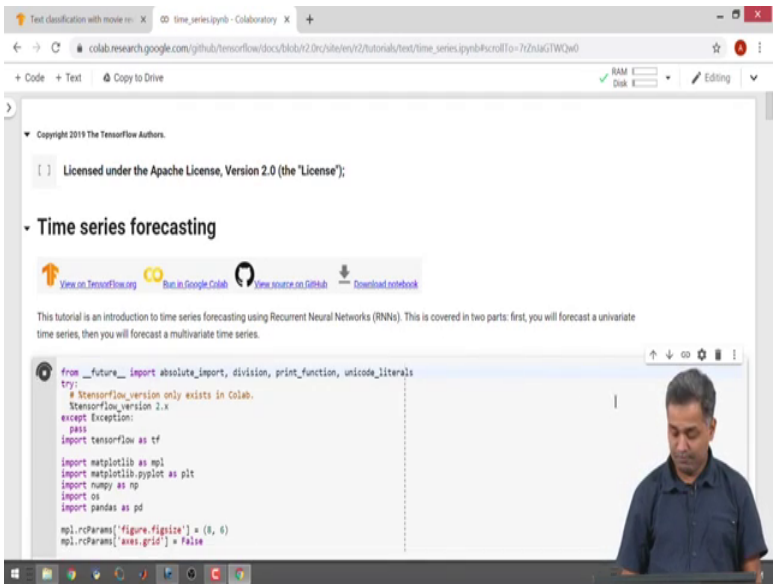
Practical Machine Learning
Dr. Ashish Tendulkar
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

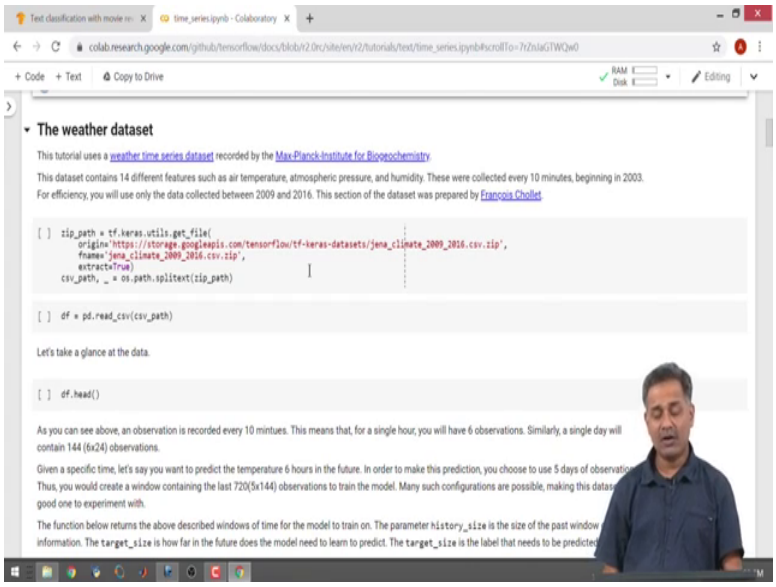
Lecture - 31
Text Generation with RNNs

(Refer Slide Time: 00:22)



Now, We will use RNNs to build models for time series data. We will build two models one for prediction of the univariate time series and second for predictions of multivariate time series. So, let us begin by importing necessary packages.

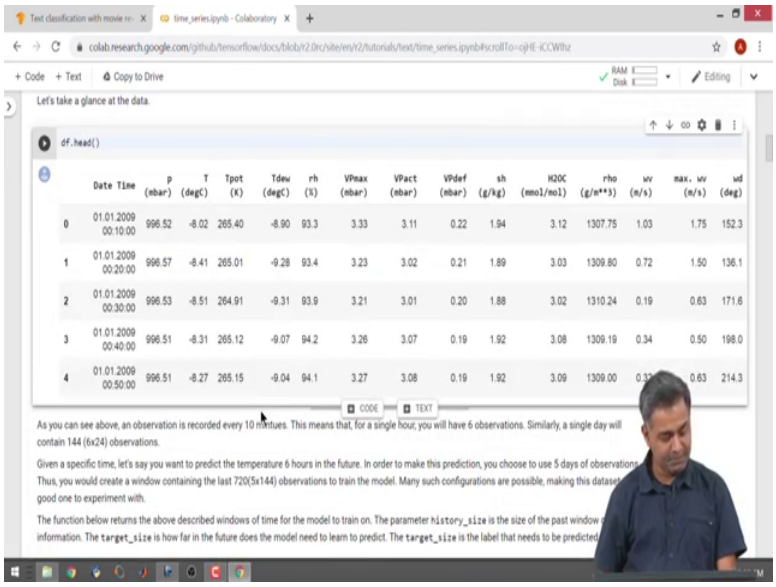
(Refer Slide Time: 00:51)



Here we are going to use a weather time series dataset which is recorded by Max Planck Institute for Biogeochemistry. The dataset contains 14 different features such as air temperature, atmospheric pressure and humidity. These were collected every ten minutes beginning in 2003, for efficiency we will use only the data collected between 2009 and 2016.

This data set is prepared by Francois Chollet, we will download the dataset. Dataset is in CSV file.

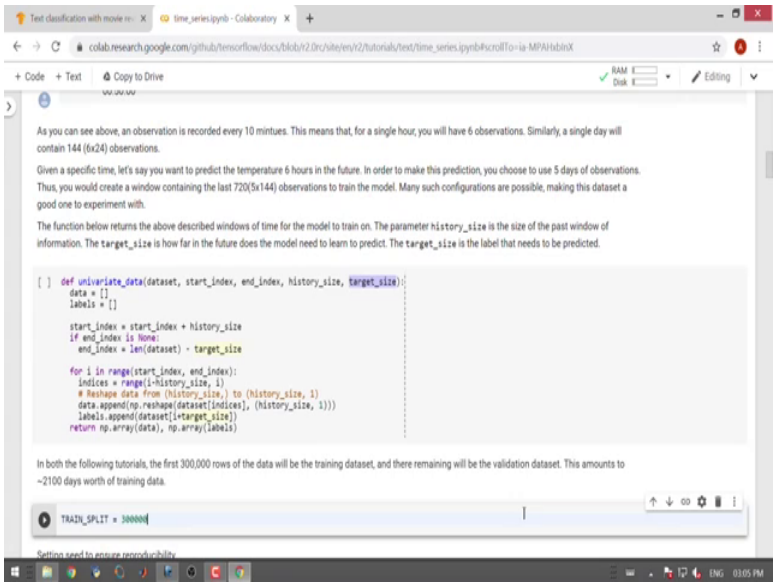
(Refer Slide Time: 01:39)



Let us look at top five entries of the dataset. See you can see that the observations are recorded every 10 minutes for a single hour we will have 6 observations. And in a day we will have 144 observations. Given a specific time, let us say if you want to predict temperature for 6 hours in the future; in order to make this prediction we choose to use 5 days of observation. Thus, you need to create a window containing last 720 observations to train the model.

Many such configurations are possible making this dataset a good dataset for experimentation. So, if we take last 5 days of data; in a day there are 144 observations. So, we take 720 observations in the history and from this 720 observations; we will try to make next 36 observations because we want to predict temperature for 6 hours in the future and in every hour there are 6 observations. So, there are in all 36 observations that we want to predict.

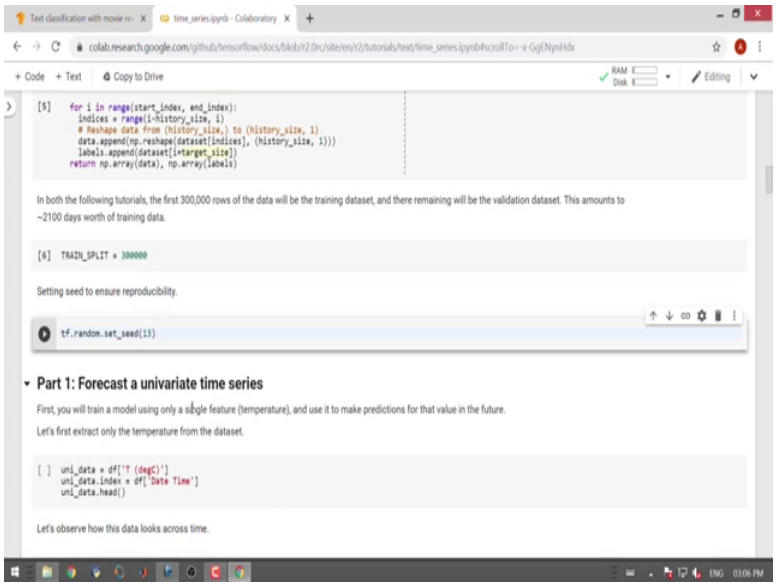
(Refer Slide Time: 03:03)



The univariate_data() function returns the windows of time for which the model is to be trained. It takes the parameter history_size which is the size of the past window of information. Target_size is how far in the future does the model need to learn to predict; the target_size is the label that needs to be predicted.

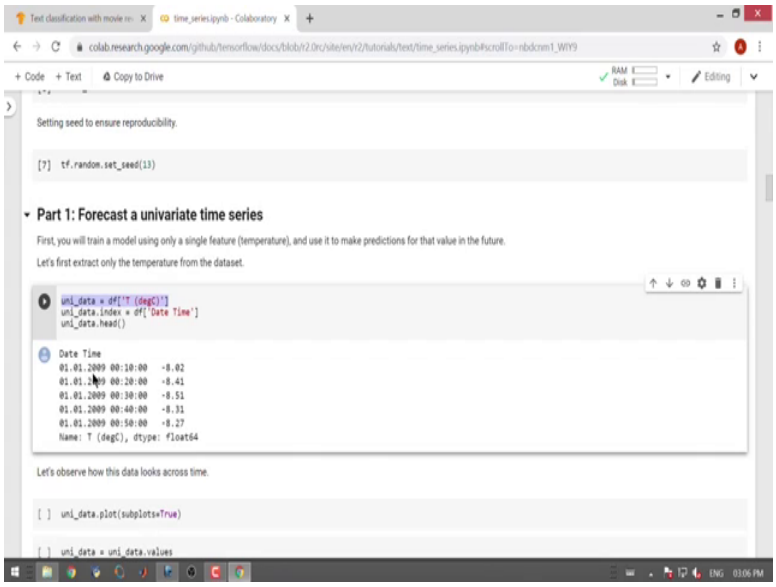
So, we define start and end index of the data and we perform some kind of a selection in the history to get the data and the corresponding labels. For this exercise we will use first 300000 rows of the data for training and remaining ones for validation. So, this amounts to about 2100 days worth of training data.

(Refer Slide Time: 04:21)



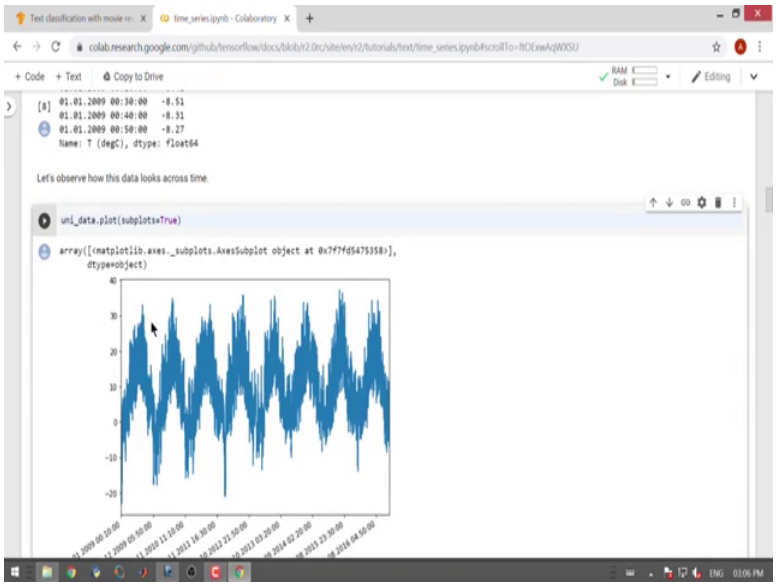
So, let us define the training split. We will set a seed to ensure that the results are reproducible.

(Refer Slide Time: 04:30)

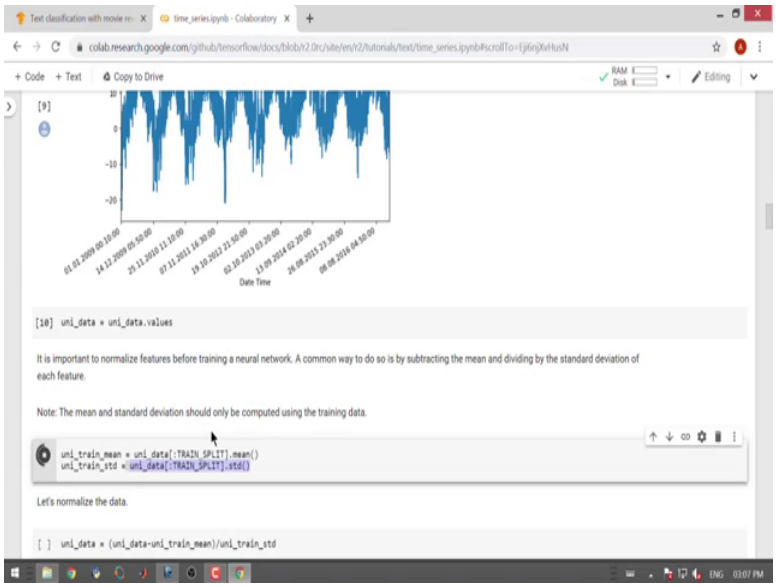


Let us begin with the first part that deals with forecasting a univariate time series. Here we will use a single variable called temperature and we will use it to make predictions for the temperature values in the future. We first extract the temperature information from the dataset.

(Refer Slide Time: 05:09)

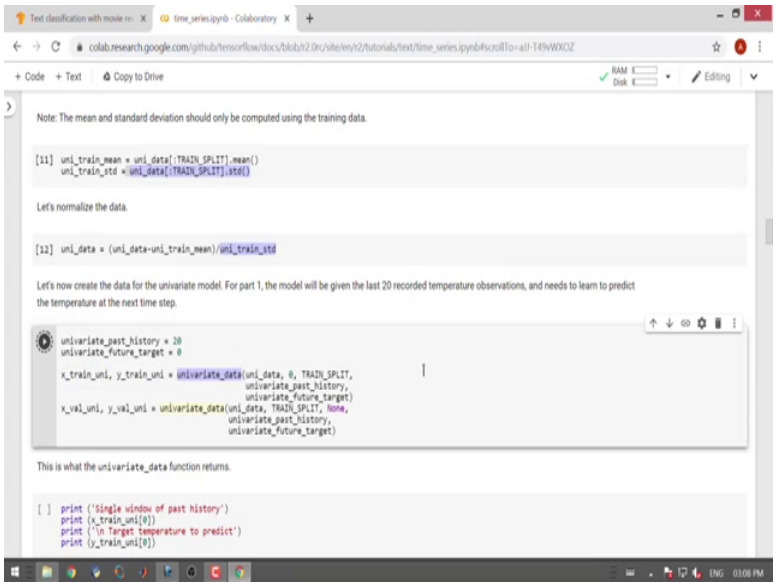


(Refer Slide Time: 05:13)



Let us look at the temperature data. This is temperature data as recorded on different dates. We will normalize the features before training a neural network, we will use a z-score normalization technique for normalization, for that we require to compute mean under standard deviation.

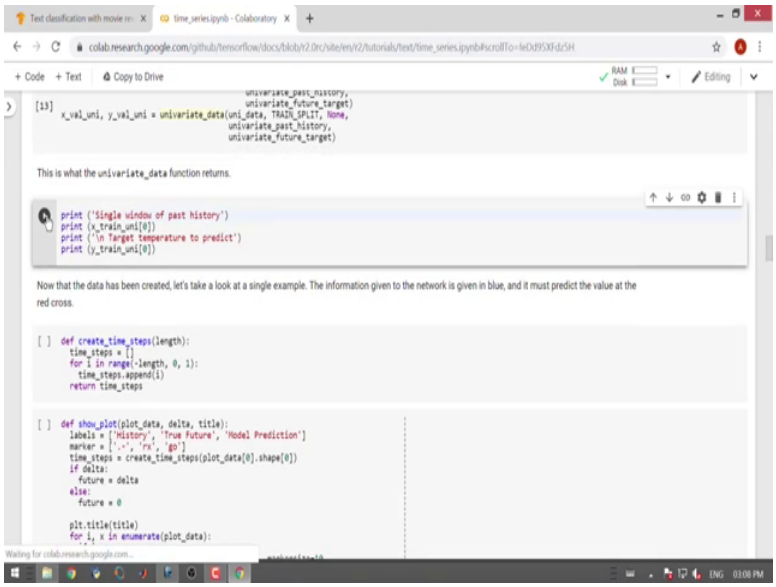
(Refer Slide Time: 05:39)



And here we normalize the data by subtracting mean from every data point and dividing the resultant by the standard deviation. Now that we have normalized the data, we will create the data for univariate model. The model will be given last 20 recorded temperatures and it needs to predict the temperature at the next time. So, we use univariate_data() function that we define for extracting the training and the target data.

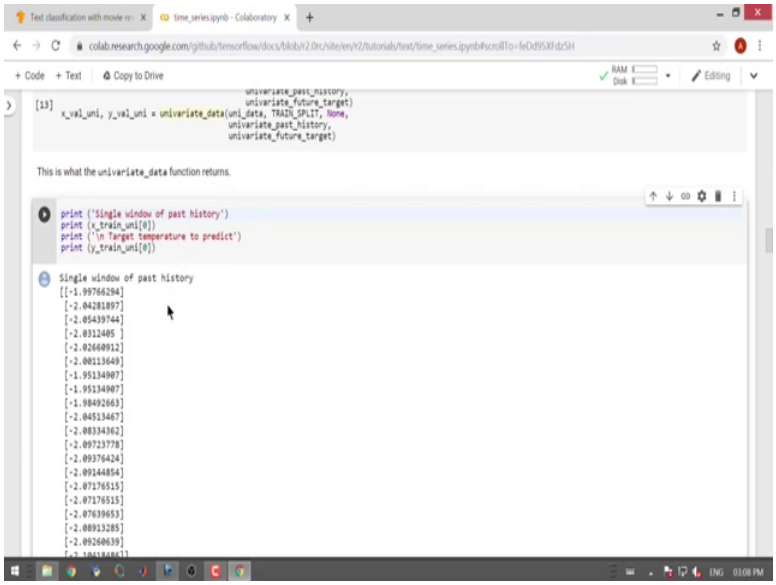
So, we apply the same function to obtain training and validation datasets.

(Refer Slide Time: 06:34)

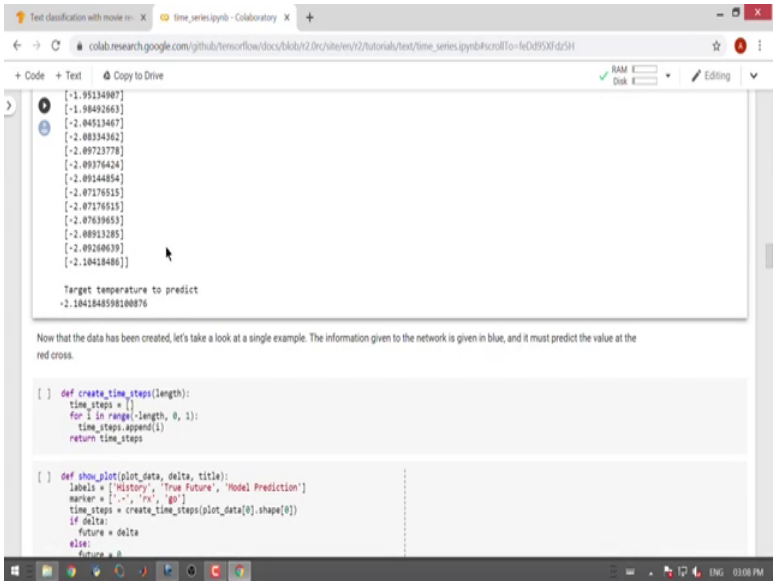


So, let us look at what this particular function returns.

(Refer Slide Time: 06:39)

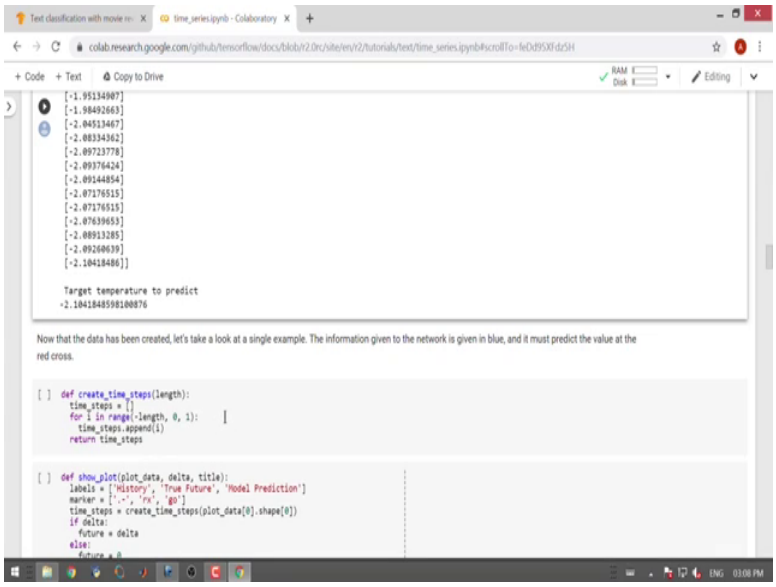


(Refer Slide Time: 06:42)

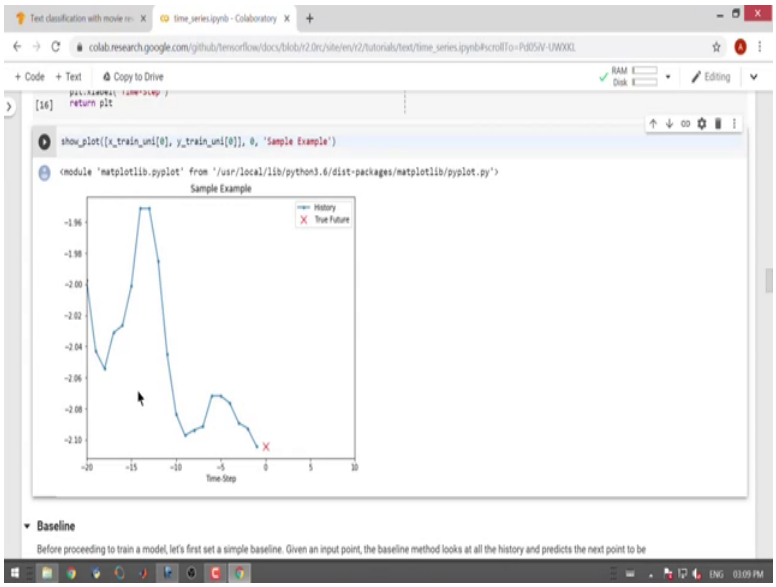


So, this is a single window of past history having 20 values and you also get the target temperature to predict.

(Refer Slide Time: 06:51)

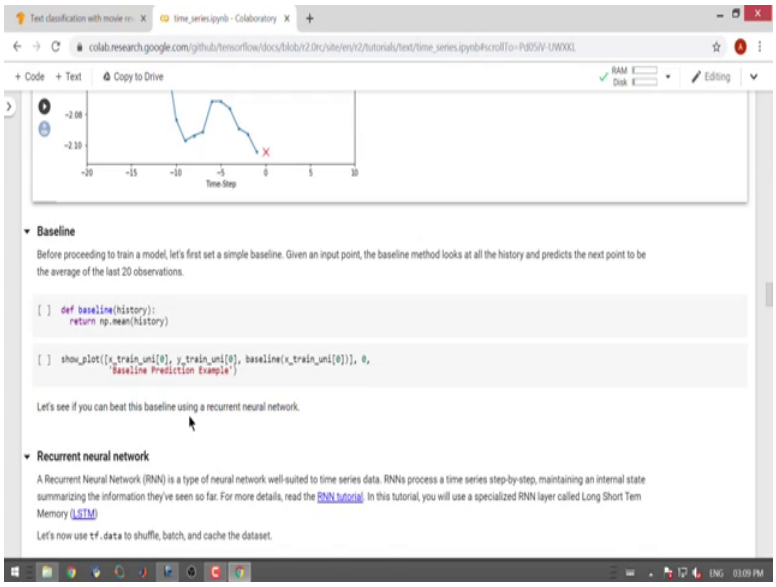


(Refer Slide Time: 07:01)



Now, that the data has been created; let us look at a single example. The information given to the network is in blue; so this is the information that is provided to the network and you want to predict the point marked by the red cross and this particular cross is the actual value.

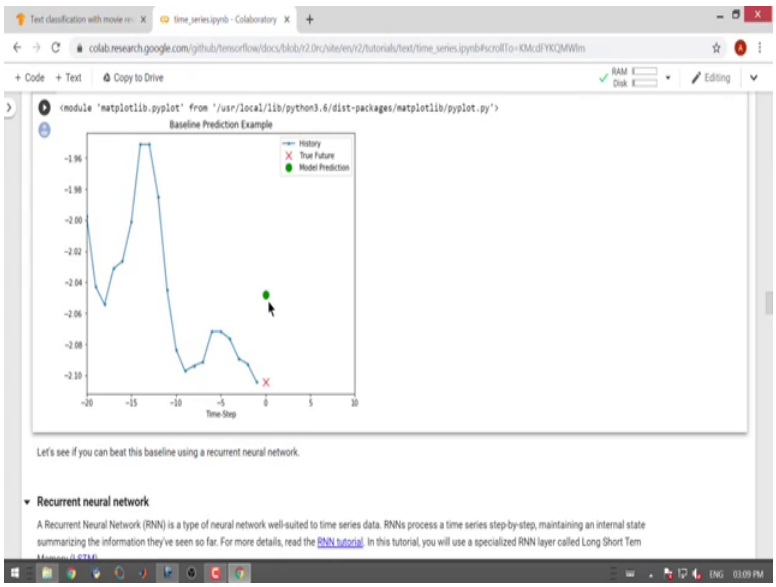
(Refer Slide Time: 07:19)



So, before proceeding to training a model; it is always a good idea to establish some kind of baseline. Here, the simple baseline could be looking at a historical data and predicting next point as the average of last 20 observations.

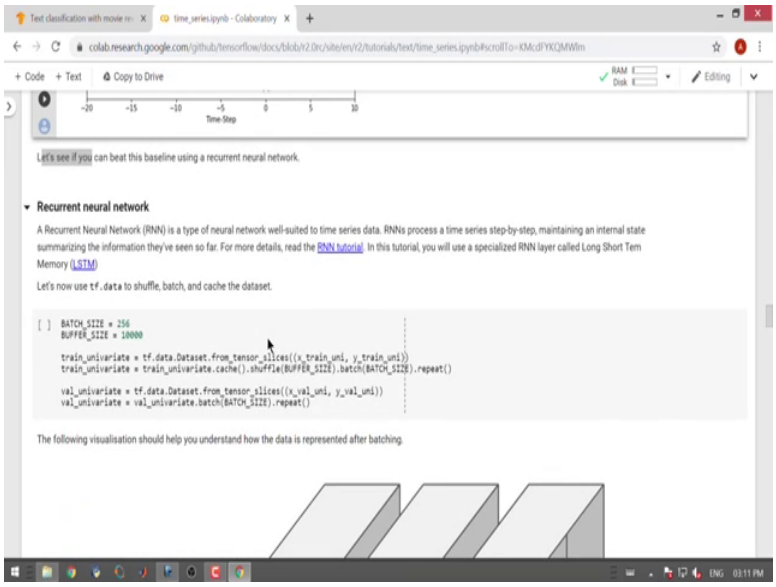
So, let us look at how this baseline performs; we simply np dot mean on the history.

(Refer Slide Time: 07:50)



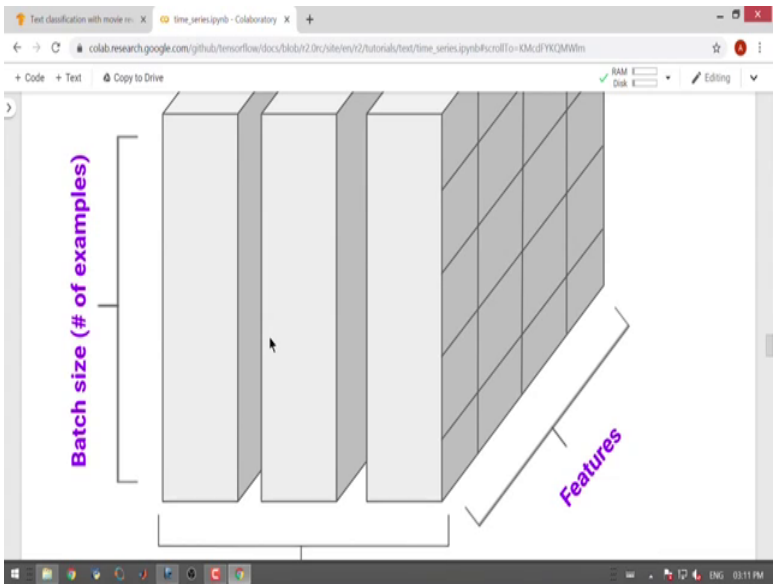
So, you can see that the model prediction on the base model is quite off from the two feature. Now, we will try to see if we can do anything better than the base line.

(Refer Slide Time: 08:09)



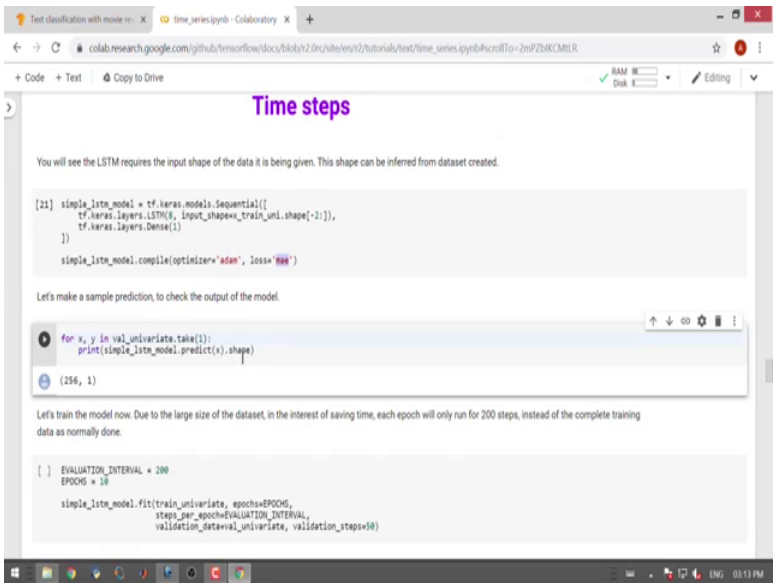
So, here we will train a recurrent neural network model which is suitable for handling sequential data or a time series data which is also an example of a sequence data. RNN process a time series step by step maintaining an internal state summarizing the information they have seen so far. Here we will use a LSTM models for modeling the time series data. Before we begin we prepare dataset objects for training and validation. We will first shuffle the object then batch it and we cache the dataset.

(Refer Slide Time: 09:04)



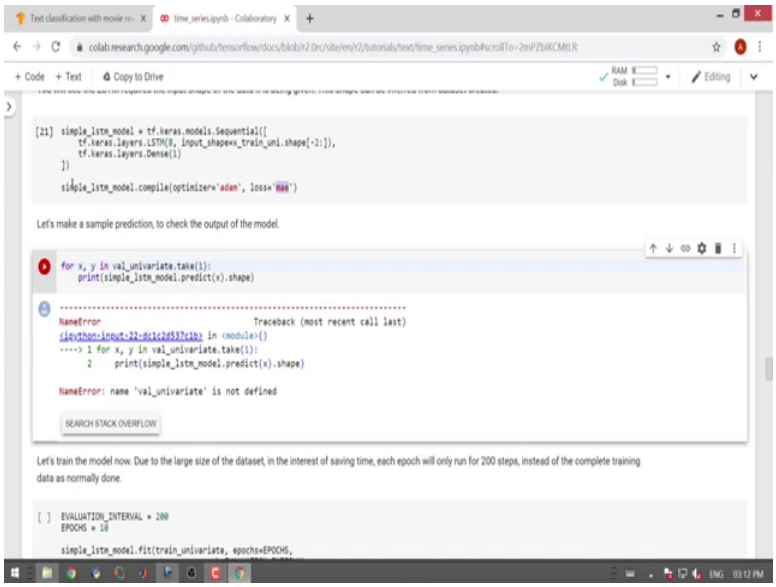
For every time step we have number of features and this is a batch up example and here in the third axis you have features.

(Refer Slide Time: 09:31)



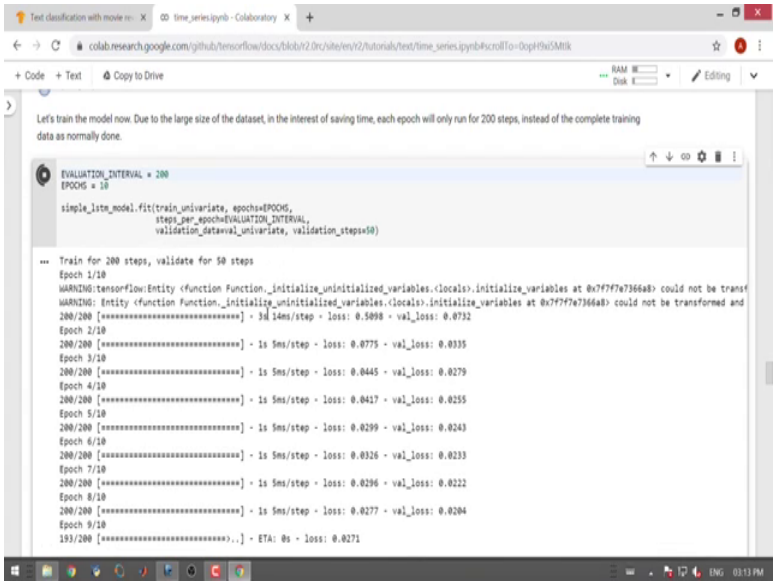
Let us build a simple LSTM model which gives out 8 outputs and output of LSTM is passed through a dense layer with a single unit. Note that we are not using any activation here because we are trying to solve a regression type of problem. We use *Adam* as an optimizer and we are going to optimize the minimum absolute error as a loss.

(Refer Slide Time: 10:19)



Let us make a sample prediction and check the output of the model.

(Refer Slide Time: 10:35)



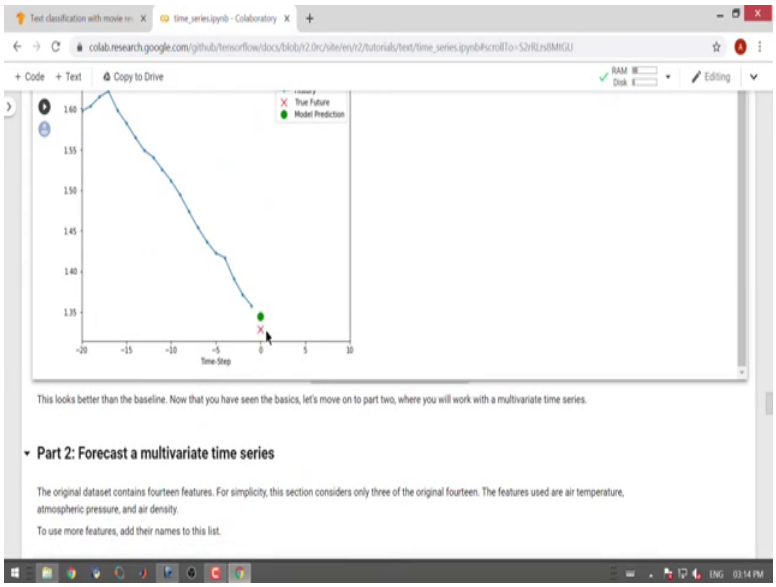
So, let us send the model; now we will run the model only for 200 steps and we are going to have 10 epochs; in each epoch you only train a model for 200 steps.

(Refer Slide Time: 10:44)

Let us predict using simple LSTM model; you can see that in the first case the prediction is very close.

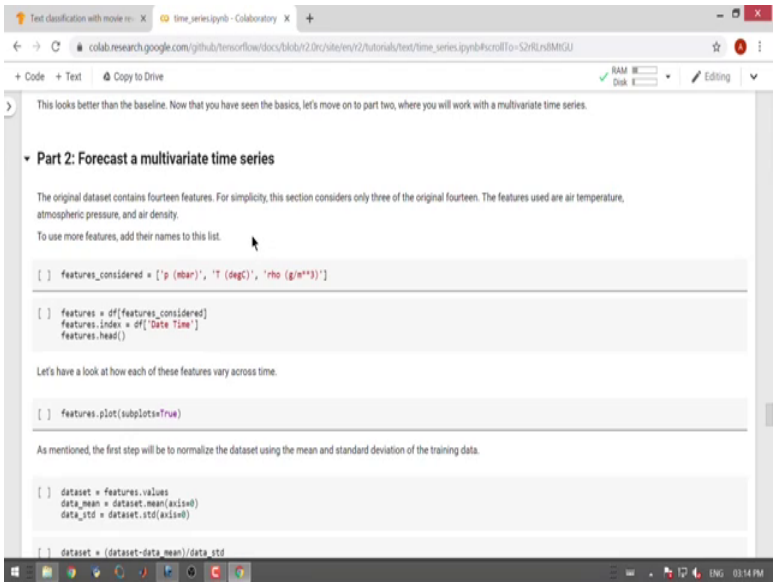
(Refer Slide Time: 10:53)

(Refer Slide Time: 10:55)



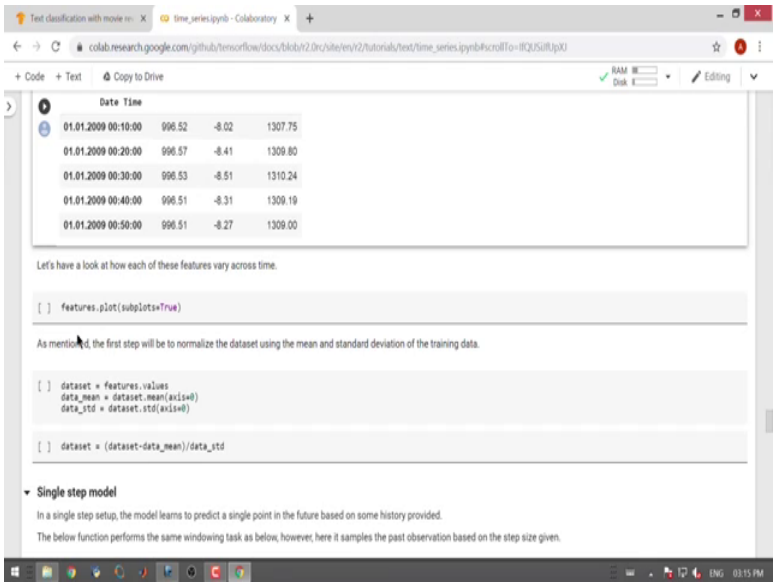
Second case it is rightly off. Third case it is quite close. So this looks better than the baseline. Now, that you see in the basics of how to train a RNN model for time series forecasting, we will move on to the next part where we will do time series forecasting for multivariate time series.

(Refer Slide Time: 11:20)



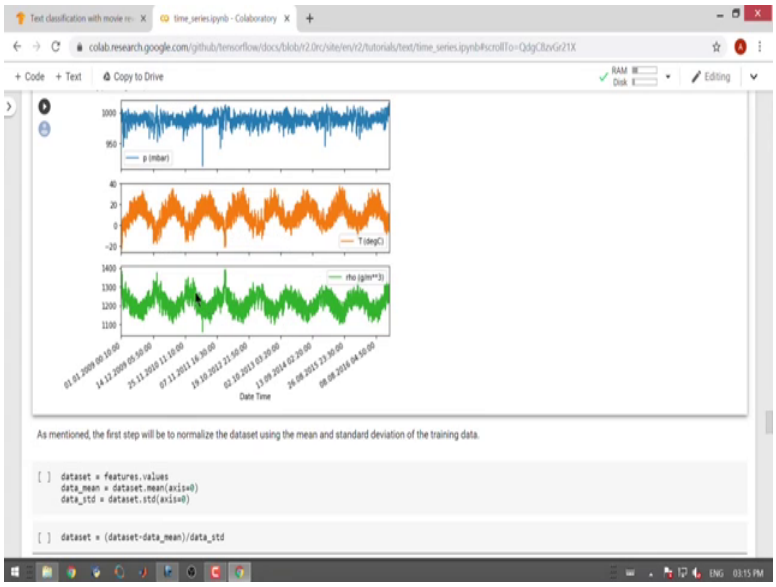
So, for the sake of this exercise; we will just select 3 of the 14 features this features are air temperature, atmospheric pressure and the air density.

(Refer Slide Time: 11:33)



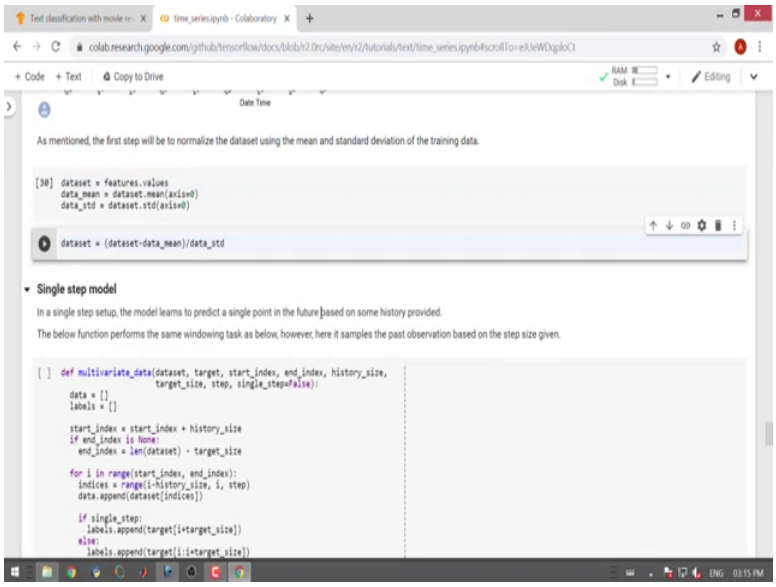
Let us look at how each of these features vary across time.

(Refer Slide Time: 11:39)



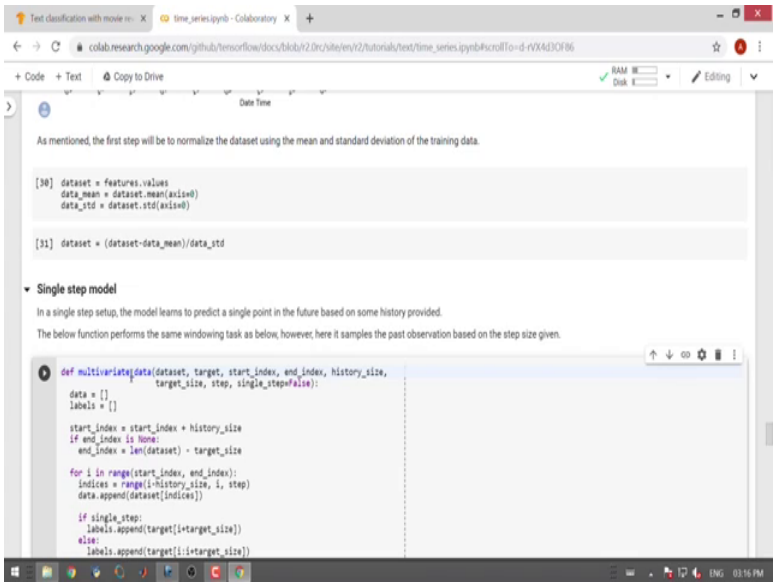
So, you can see that this features are on different scale and they have different variability across time.

(Refer Slide Time: 11:50)



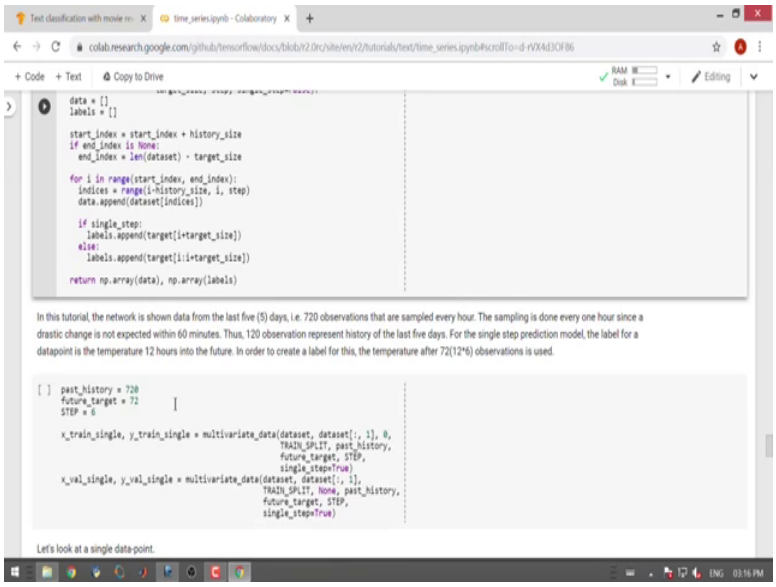
Let us normalize the data using mean and standard deviation. Now, we will use a single step model, this model lost to predict a single point in the feature based on some history provided.

(Refer Slide Time: 12:06)



So, let us define the function for performing the window in task. Here we will sample the past observations based on the step size that is given to us.

(Refer Slide Time: 12:18)

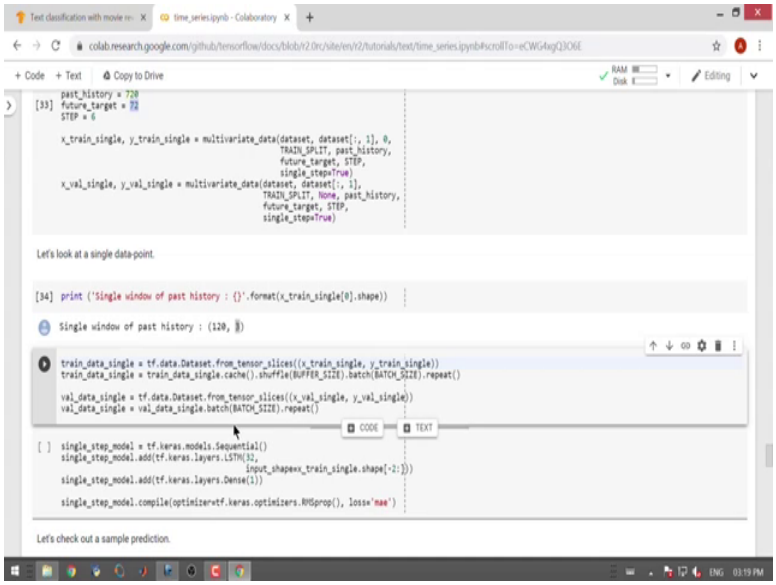


So, in this exercise we will show last 5 days of data to the network; last 5 days of data will constitute 720 observations that are sampled every hour. The sampling is done every 1 hour; since drastic change is not expected within 60 minutes.

Thus, 120 observation represent history of the last 5 days. For the single step prediction model the label for a data point is the temperature 12 hours into the future. In order to create a label for this the temperature after 72 observations is used. Note that we are having 6 observations per hour; so in 12 hours we will have total 72 observations; that is why we use temperature which is after 72 observations in the future.

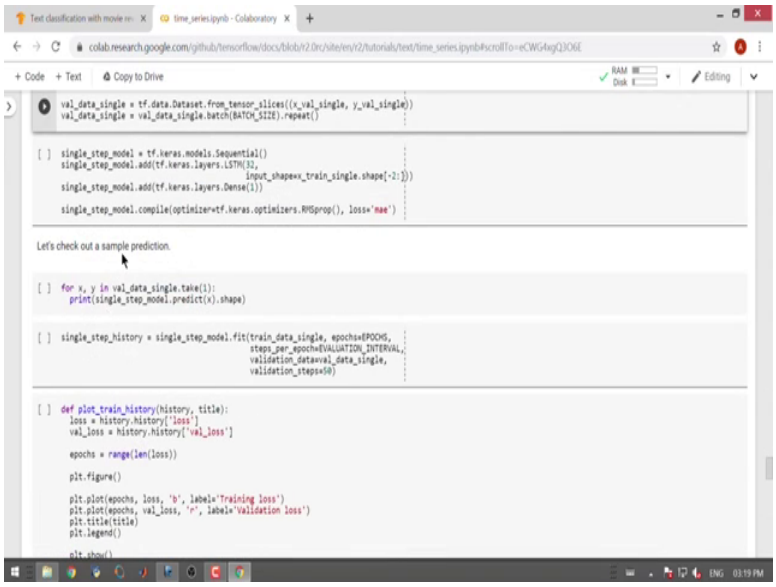
So, we use past history of 720 points and we want to predict the target in the future which is 72 observations away from the current point in the future.

(Refer Slide Time: 13:40)



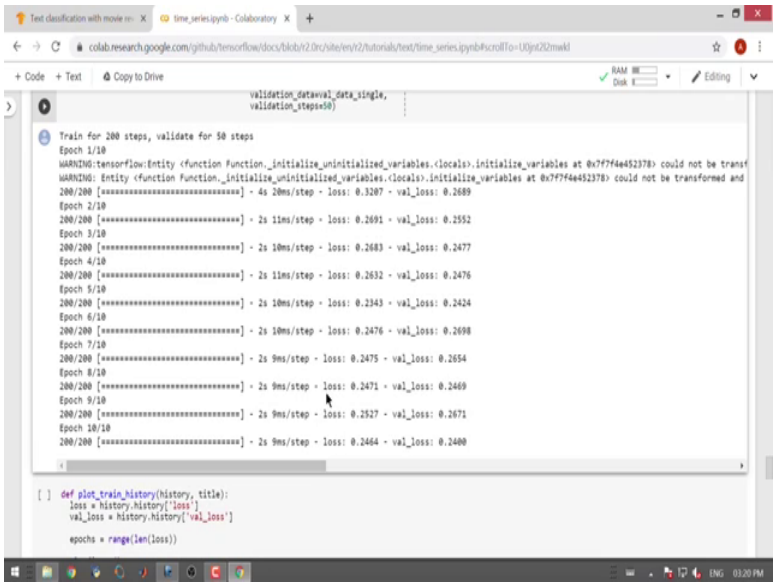
Let us create training and validation data sets; the shape of the train data of a single point. So, we have 120 points and each point has got 3 features. So, we construct data set object with from_tensor_slice and then we shuffle it, batch it and then cache the dataset; in case of validation we only batch the dataset.

(Refer Slide Time: 14:12)



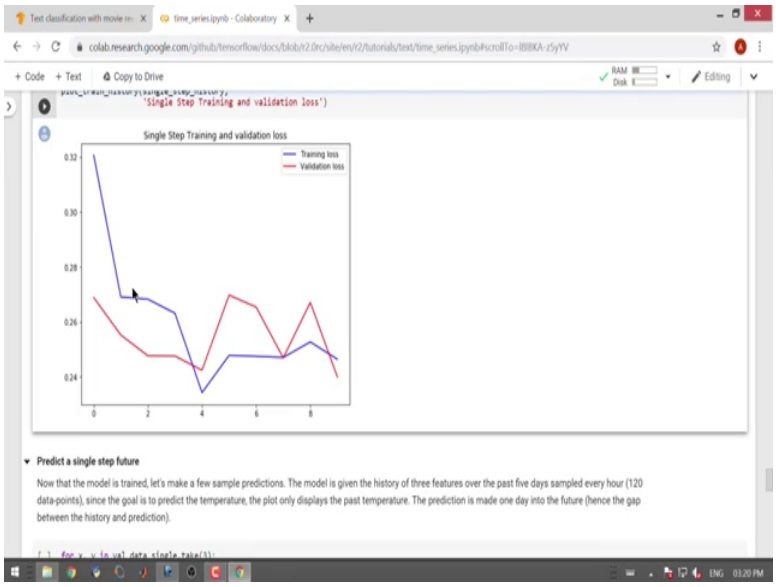
We build the same sequential model with LSTM layer and a denser layer; use RMS prop as an optimizer and mean absolute error as the loss() function.

(Refer Slide Time: 14:44)



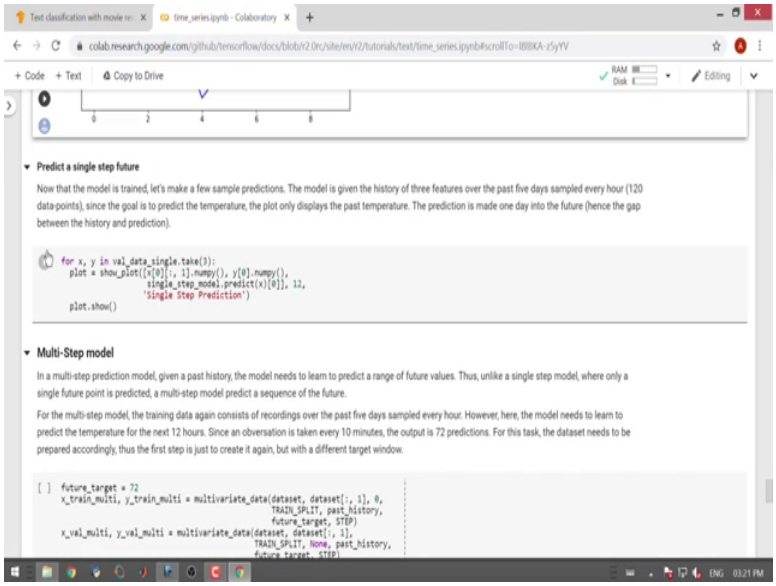
Let us check out the sample predictions; let us train the model.

(Refer Slide Time: 15:18)



Let us look at how training progressed. So, the training loss is coming down; validation loss came down went up it is slightly a zigzag kind of pattern in the validation loss, not so smooth.

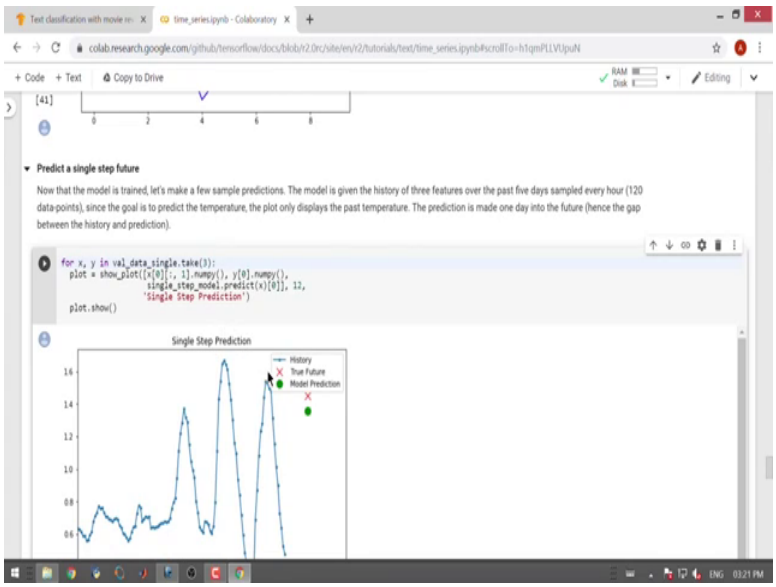
(Refer Slide Time: 15:39)



Now that the model is trained let us make a few sample predictions. The model is given the history of three features for the past 5 days sampled every hour which is 120 data

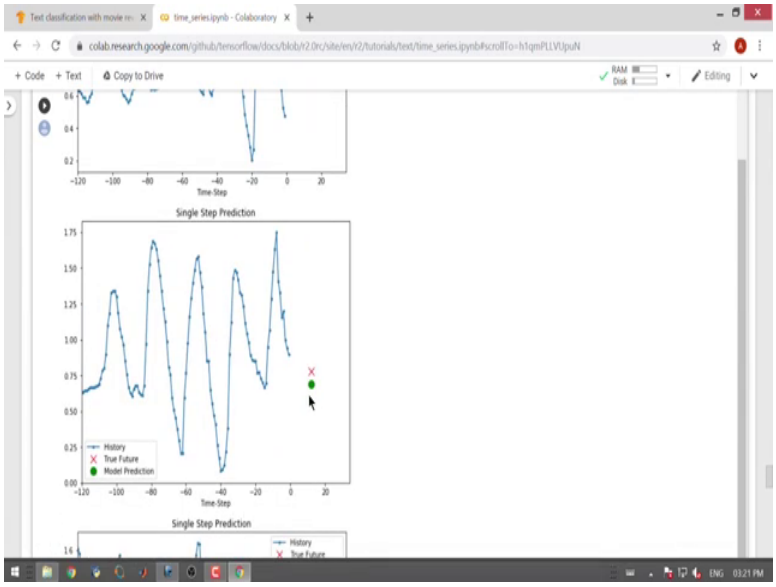
points. Since the goal is to predict the temperature, we only display the past temperature in the plot and the prediction is made one day in the future.

(Refer Slide Time: 16:12)



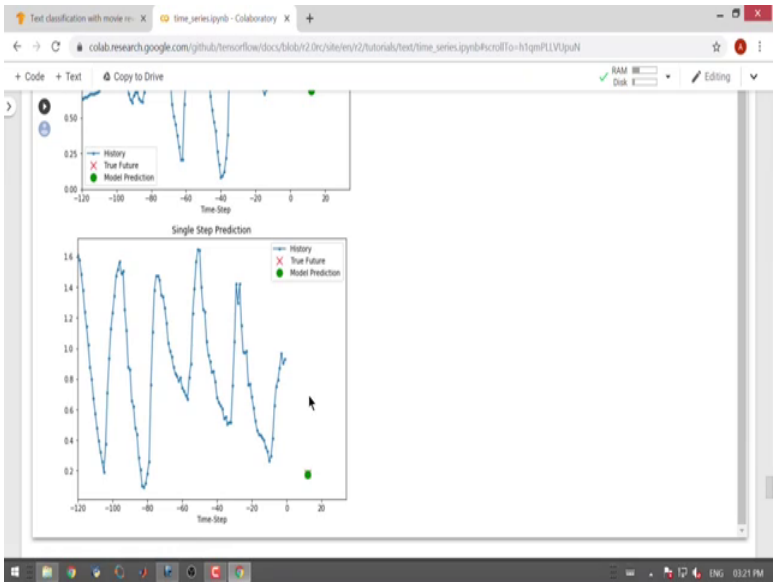
So, you can see for first two examples; for first example they are very close.

(Refer Slide Time: 16:21)



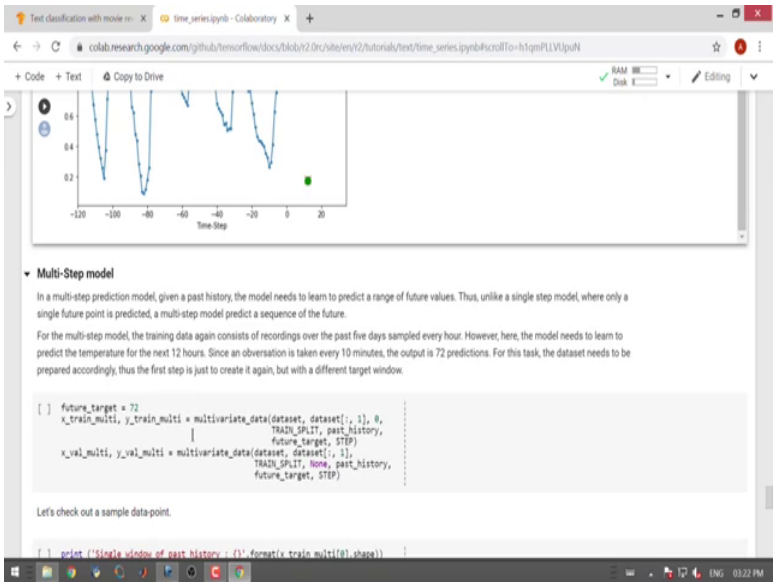
Second example also are very close.

(Refer Slide Time: 16:22)



Third example they are very close. So, you can see that using more features; you are able to predict the temperature quite accurately.

(Refer Slide Time: 16:33)



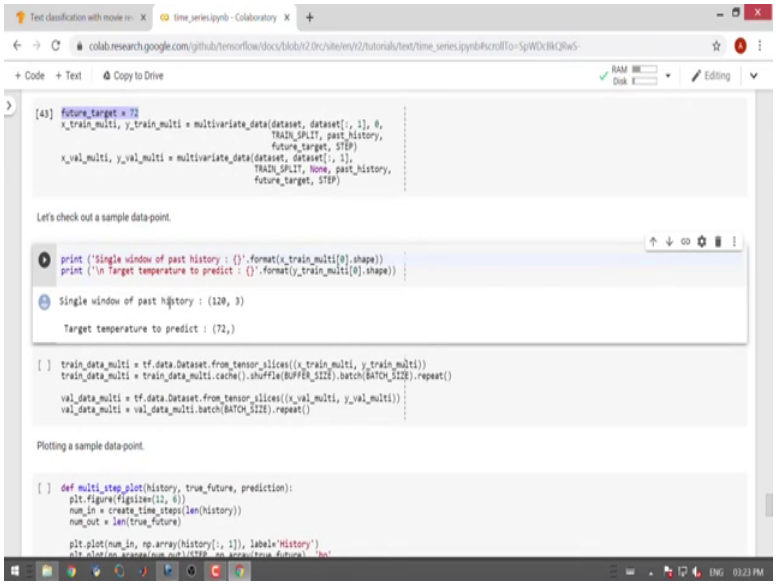
In multi step model, instead of predicting just one value we will try to predict multiple values in the future.

So, you can see that for the multi step model the training data consists of recording over the past five days sampled every hour which will be 120 points. However, the model

needs to learn to predict the temperature for the next 12 hours; since an observation is taken every 10 minutes, will have 72 predictions over next 12 hours.

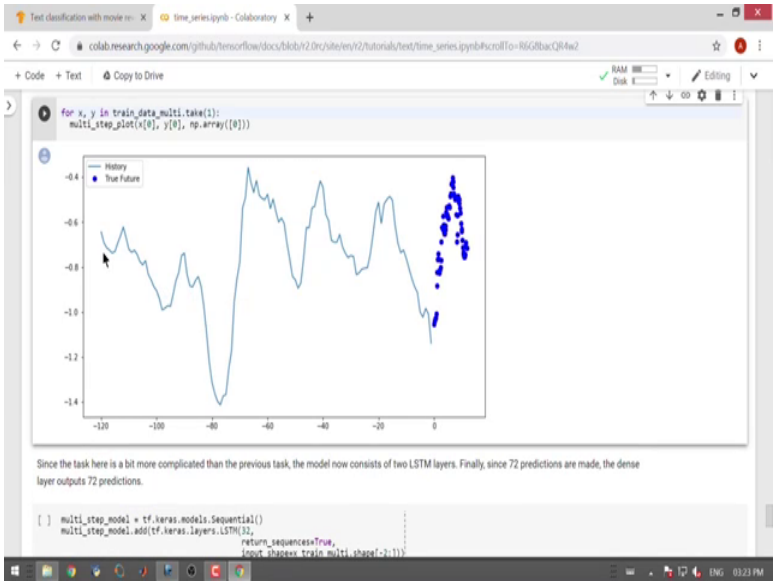
So, our target has now 72 values. So to predict the 72 values from the last 120 values.

(Refer Slide Time: 17:50)



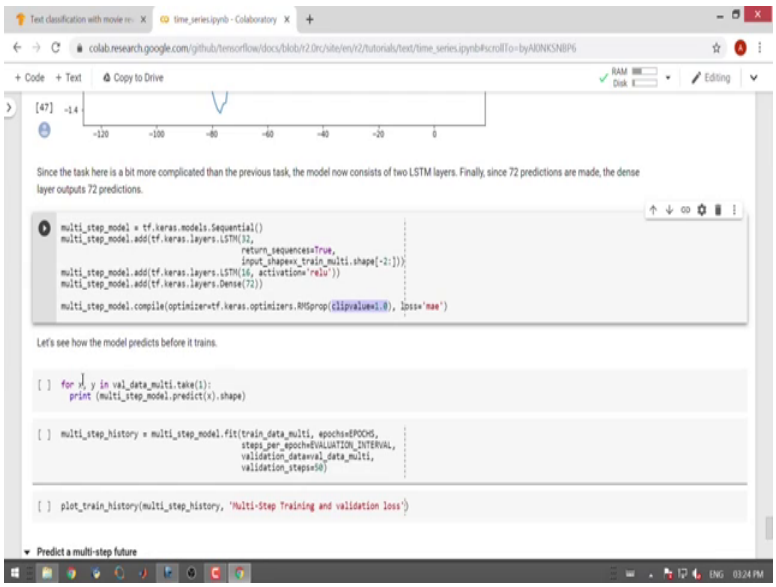
Let us create a data set, you can see that we have 120 points in the training set; each has 3 features and we want predict 72 different values in the future.

(Refer Slide Time: 18:06)



We create the data set. So, the smooth line here is a history and the dots here are kind of predicted values.

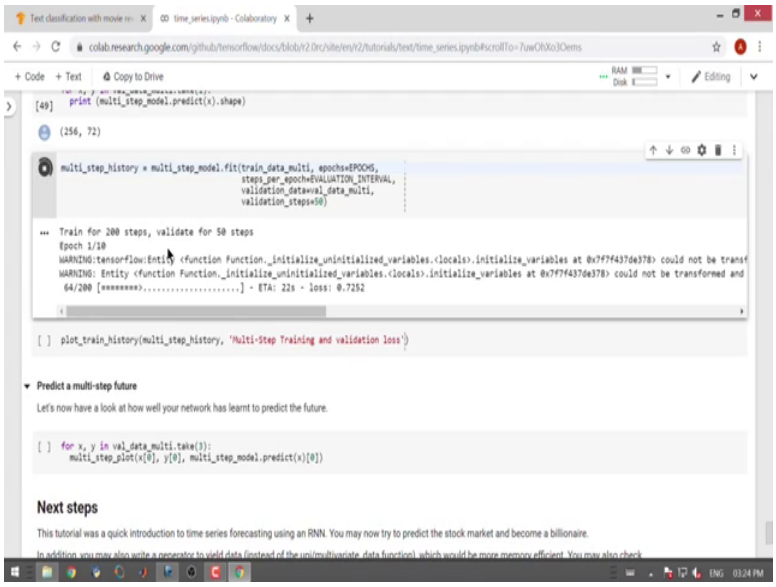
(Refer Slide Time: 18:17)



Since this task is a bit more complicated than the previous task; we will use two LSTM layers. Finally, the 72 predictions are made using a dense layer that outputs 72 predictions.

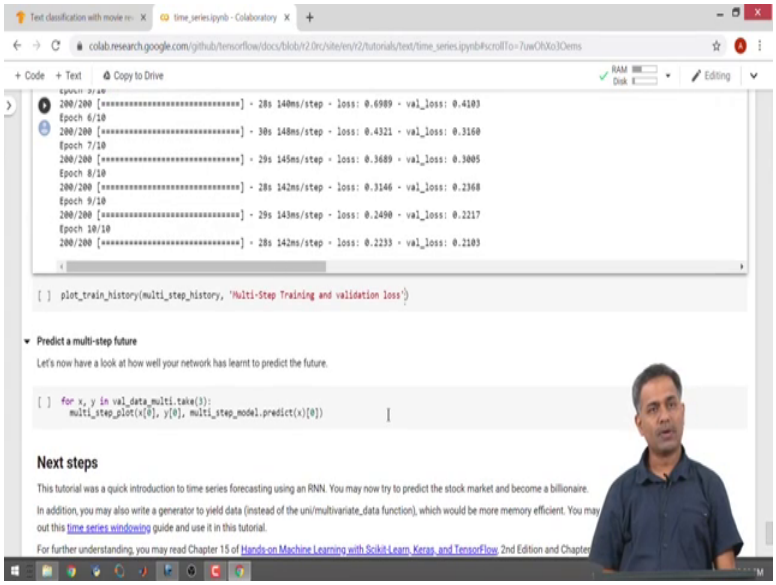
We are using *RMSProp* as an optimizer and we are using *clipvalue* as argument for clipping the gradient. We use mean absolute error as a loss() function here; let us look at how the model predicts before it trains.

(Refer Slide Time: 18:57)



Let us train the model; we will train the model for 200 steps in every epoch.

(Refer Slide Time: 09:10)

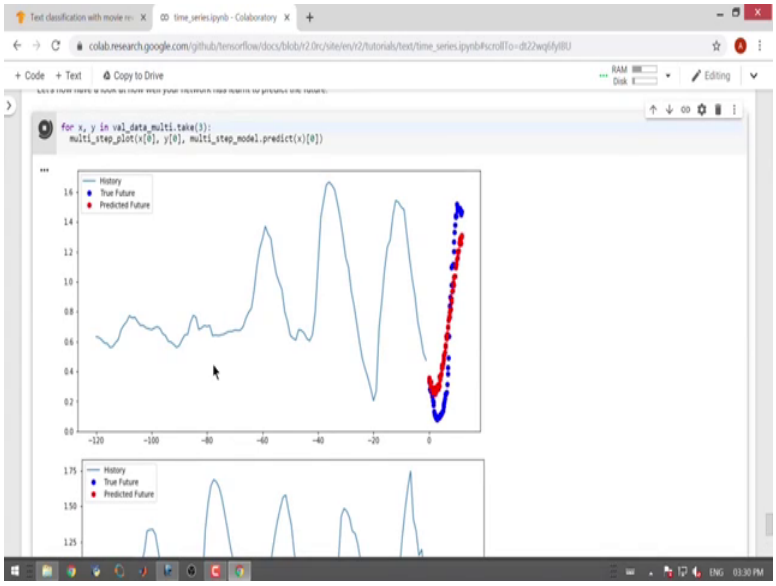


Now, that the model is trained; let us look at how the model training preceded.

(Refer Slide Time: 19:18)

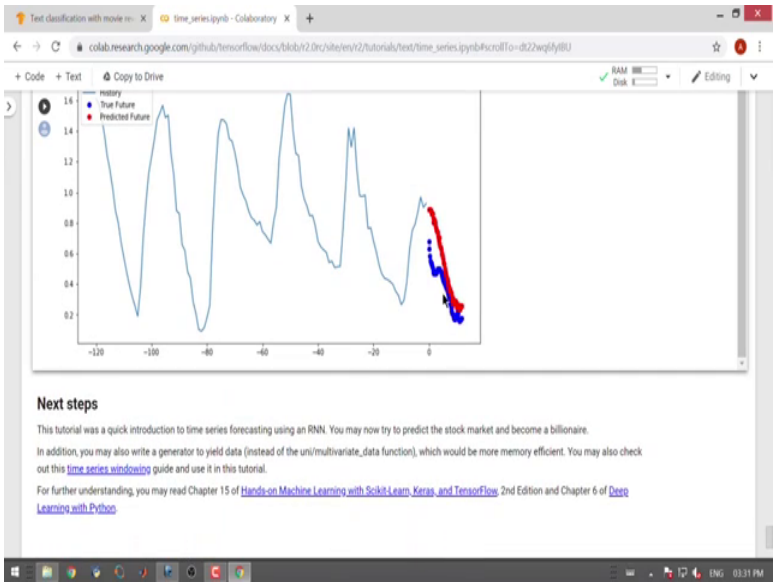
Let us predict how well the module has learn to predict the future.

(Refer Slide Time: 19:26)



So, you can see that the future; the true values in the future are in blue whereas, the model prediction is in red. You can see that in the first case it is able to predict the next values quite better.

(Refer Slide Time: 19:49)



Also in the second case and third case there are some deviations, but model seems to be doing a decent job of predictions. You can see that the model is quite smooth it is not able to get into some of the nitty-gritty details like that there are some abrupt turns; model is not able to learn that perfectly, but overall it is doing a decent job. So, that was it from modeling time series with RNNs.