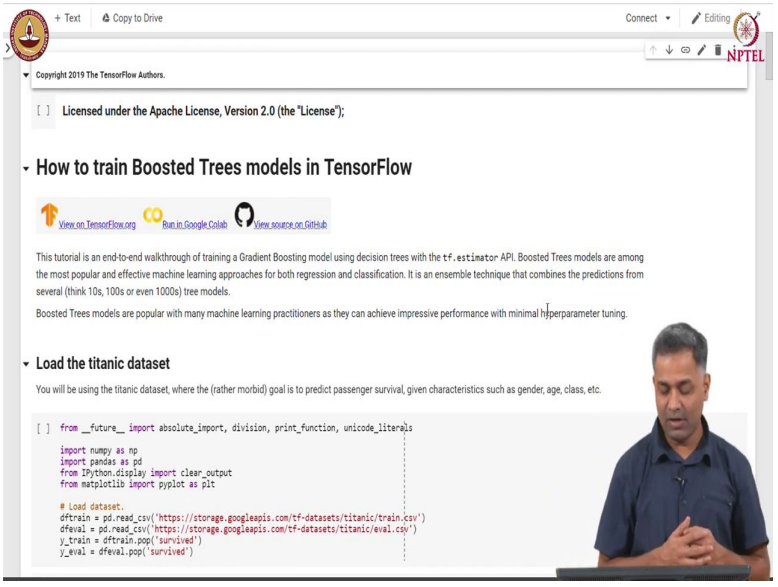
Practical Machine Learning
Dr. Ashish Tendulkar
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 29
Boosted Trees

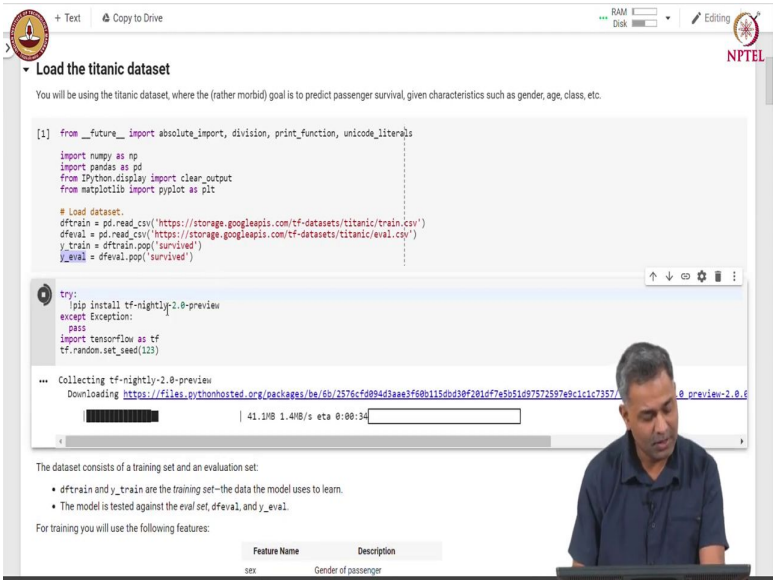
You have already studied how to use tf.estimator API for training linear classifiers. We will use tf.estimator API to train a boosted tree classifiers, which are quite popular in the machine learning community for modeling structured data.

(Refer Slide Time: 00:37)



The boosted tree model is an ensemble technique that combines predictions from tens and hundreds of trees. Another good point about boosted tree models is that they achieve the impressive performance on structured data with minimal hyper parameter tuning. Hyper parameter tuning is one of the big problems with neural networks. And we mostly need to use specialized services for hyper parameter tuning in neural networks because there are just to many hyper parameters.

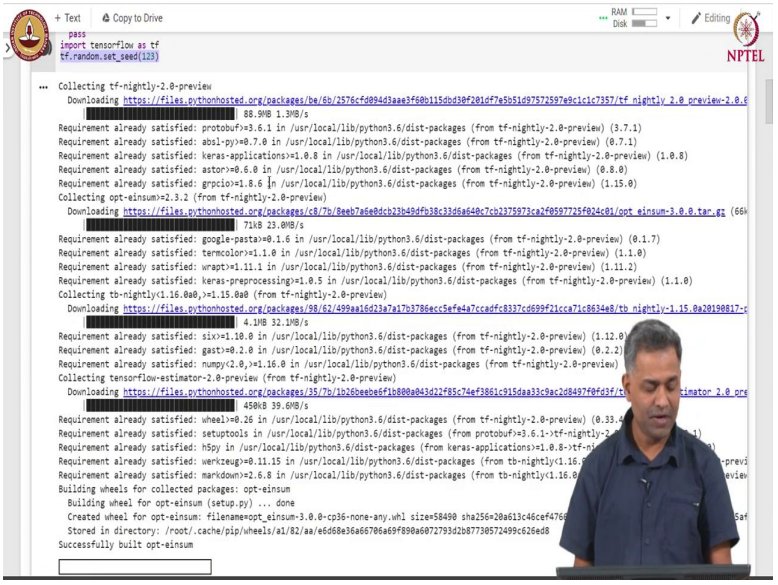
(Refer Slide Time: 01:24)



So, in this exercise we will use titanic dataset where our goal is to predict the probability of passenger surviving. As we have already seen this titanic dataset in the context of logistic regression, we will not spend a lot of time on exploration of titanic dataset.

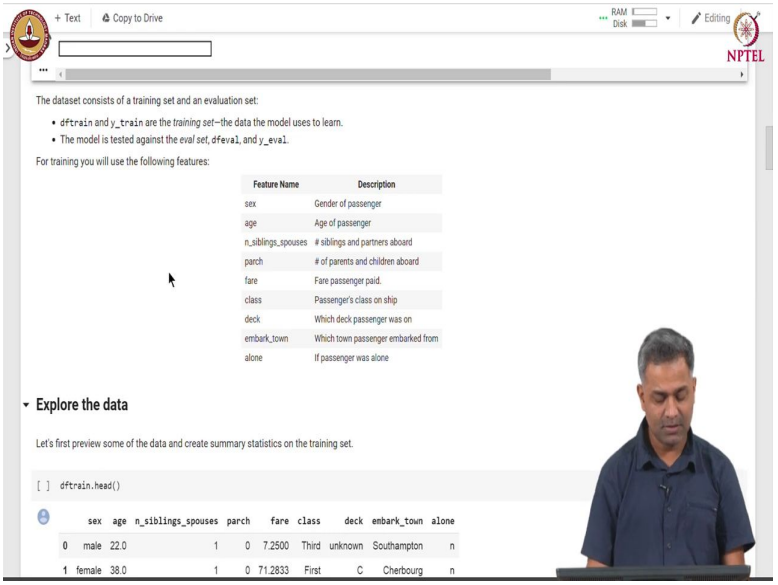
Let us download the titanic dataset. The titanic dataset already has a training and evaluation split. So, we read training and evaluation data in pandas dataframes and pop the label column from the dataframe to create another dataframe for labels. So, we have dftrain and dfeval as data frames containing features. And y_train and y_eval containing labels corresponding to train and the evaluation dataset.

(Refer Slide Time: 02:56)



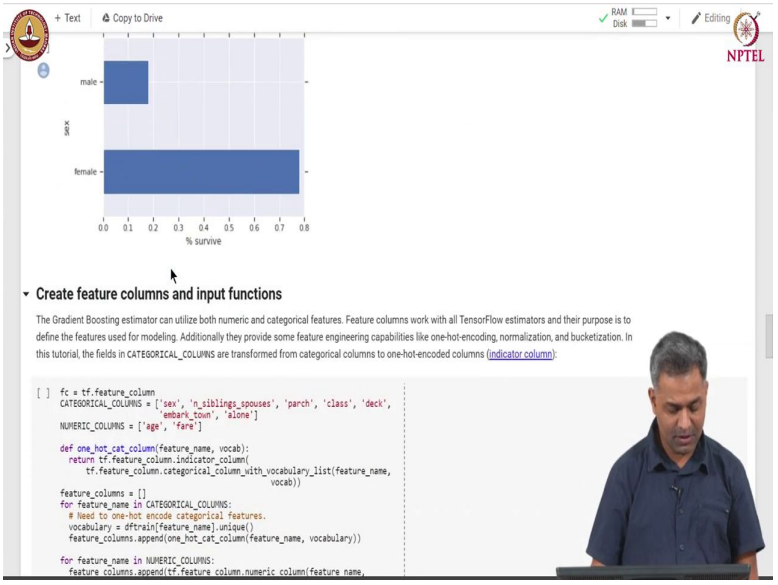
Let us install let us install TensorFlow 2.0 and set a random seed to 123. This helps to make sure that across multiple runs we get the same results.

(Refer Slide Time: 03:13)



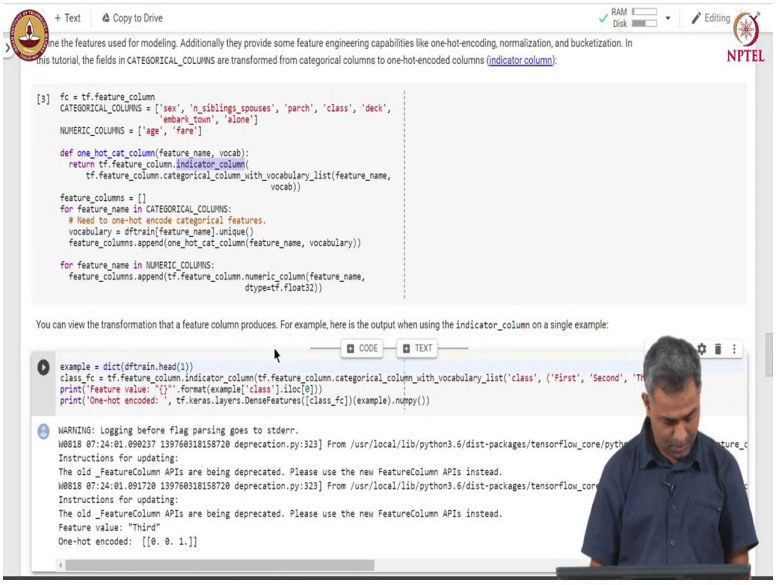
We know that titanic dataset has about nine features that describes each passenger based on their gender, age, their class, their embarkation town and some other features.

(Refer Slide Time: 03:33)



We first convert the features into feature columns. Feature columns work with all TensorFlow estimators and their purpose is to define features for modeling. Gradient boosting estimator can utilize both numeric and categorical features. So, what we will do is, we will convert the categorical features into one hot encoding.

(Refer Slide Time: 04:03)

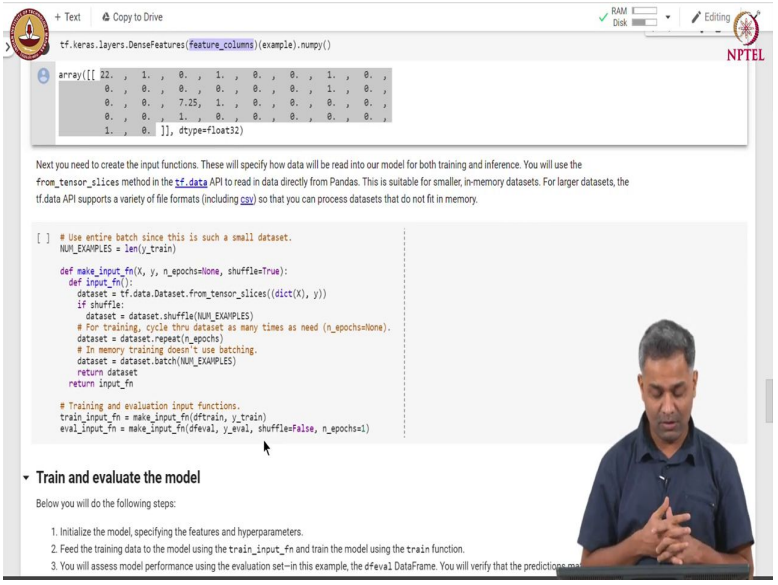


For every categorical column we first obtain the vocabulary and then we pass the vocabulary to categorical column with vocabulary list. And the output of that is fed into the indicator

column, which converts the categorical column or each value in the categorical column into one hot categorical column.

Numerical features are fairly easy to handle. And we simply use numerical columns of feature column to represent each of the numerical features. We combine both categorical features and numerical features into feature_columns.

(Refer Slide Time: 05:19)



Let us look at the dense feature representation of the feature columns. And you can see that all the features are shown over here. Where there are some numerical features and categorical features are converted into one hot encoding as you can see it here. Next we need to create an input function. This will specify how data will be read into our model for both training and inference.

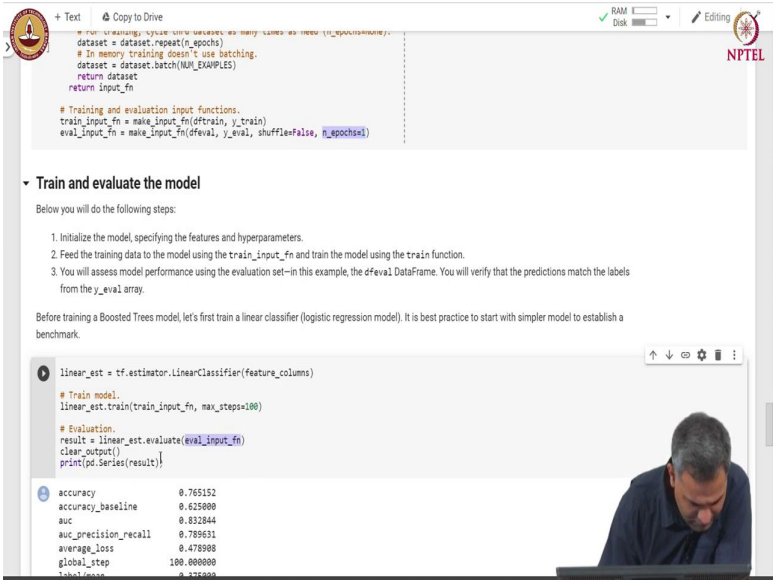
We will use the from_tensor_slices() to create the dataset. Here the dataset is in memory and stored in pandas dataframe. So, after creating the dataset we shuffle the dataset where we set the buffer size to the number of examples. And we cycle through the dataset as many times as we need. Here the n_epoch is set to none.

So, we can cycle through dataset as many times as we need. And we do not really use batching because all the data is in memory. And hence we have dataset.batch operation where

the batch size is the number of examples. So, this input_function returns it to dataset, which will be consumed during training and inference time.

So, we make the input functions for both training and evaluation. At a time of evaluation, we set shuffling to false and number of epochs to one. As the number of epoch is one, this makes sure that we iterate through the evaluation data only once.

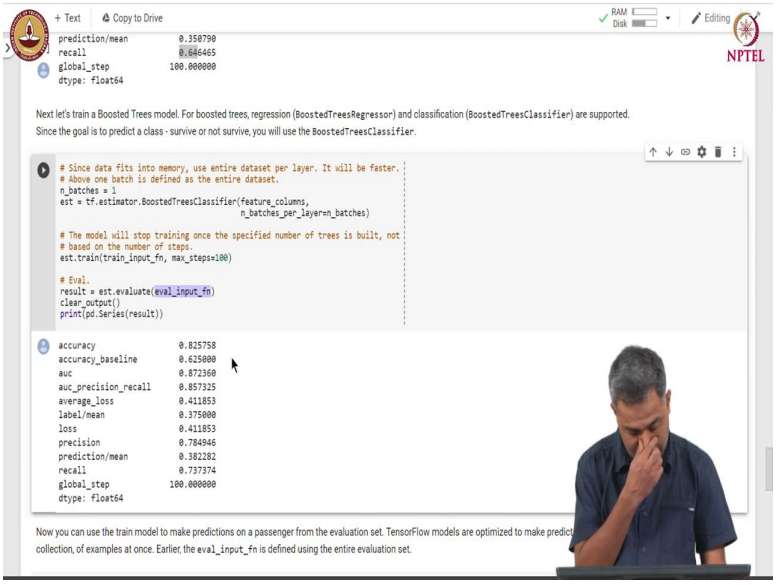
(Refer Slide Time: 07:24)



Before building a boosted tree model let us build a logistics regression classifier to establish a baseline for this problem. So, we define logistic regression classifier with tf.estimator.LinearClassifier. And supply feature_columns as an argument.

We then train the classifier by specifying the input function and we train for maximum_steps. We evaluate the model with the evaluation_input_fn. And finally, we print the result of the evaluation.

(Refer Slide Time: 08:28)

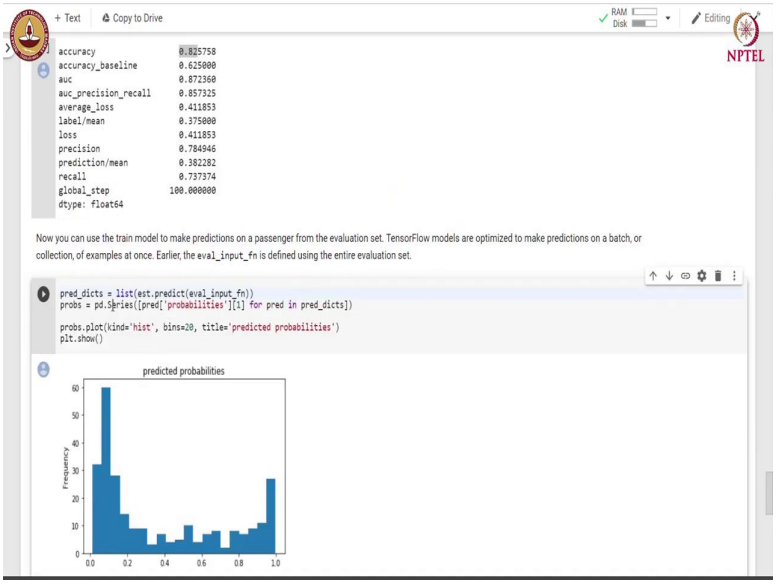


You can see that we got accuracy of 76 percent. The baseline accuracy 62 percent and we got precision of 70 percent and recall of 64 percent. Now, let us train a boosted tree classifier for boosted trees there are TensorFlow supports boosted tree regressor and boosted tree classifier. Here since we are interested in predicting the survival or non survivals.

Here since our object is to predict whether passenger survives or not we are going to use boosted tree classifier. In BoostedTreeClassifier, we specify the feature_columns and the number of batches. We specify the maximum number of steps the model will stop training once the specified number of trees are built. And we evaluate by supplying the evaluation function.

Let us train the boosted tree classifier and check the output. So, after training the boosted tree classifier, we see that it achieves accuracy of 82 percent, which is 6 percentage points higher than the linear classifier. It achieves precision of 78 percent and recall of 73 percent, which is also higher than the linear classifier.

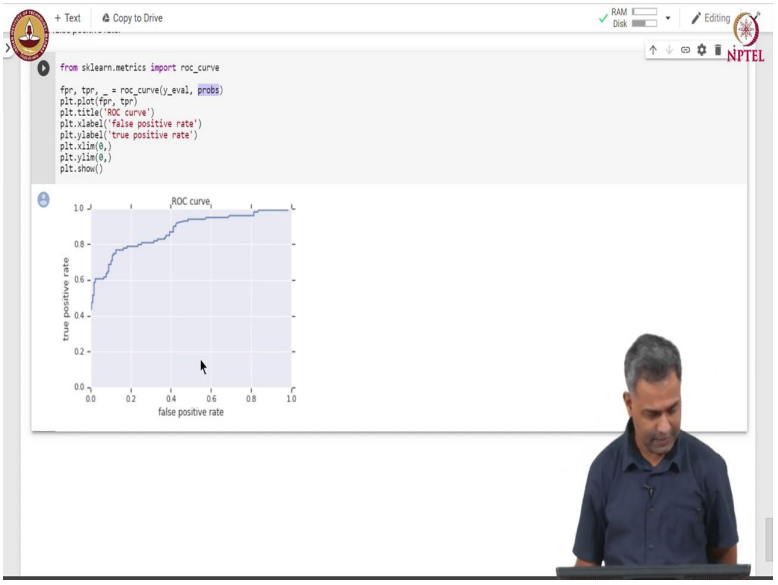
(Refer Slide Time: 10:09)



So, we can see that the model the boosted tree model performs better than the logistic regression model in this particular dataset. So, now, we can use the model to make predictions on a passenger from the evaluation set. TensorFlow models are optimized to make predictions in a batch or collection of examples all at once.

So, we give the eval_input_function and this eval_input_function is defined on the entire evaluation set. We look at the probabilities from prediction and plot it. So, you can see that there are a lot of passengers with probability of surviving as only 0.1. And there are few passengers who have got probability equals to 1.

(Refer Slide Time: 11:32)



Let us plot the ROC curve with a ROC with roc_curve from sklearn.metrics package. For ROC curve, we have to specify the actual labels and the probabilities. And it returns the false positive rate and true positive rate. You plot ROC curve with false positive rate on the x axis and y and true positive rate on the y axis. The ROC curve gives us a better idea about tradeoff between two positive rate and false positive rate.