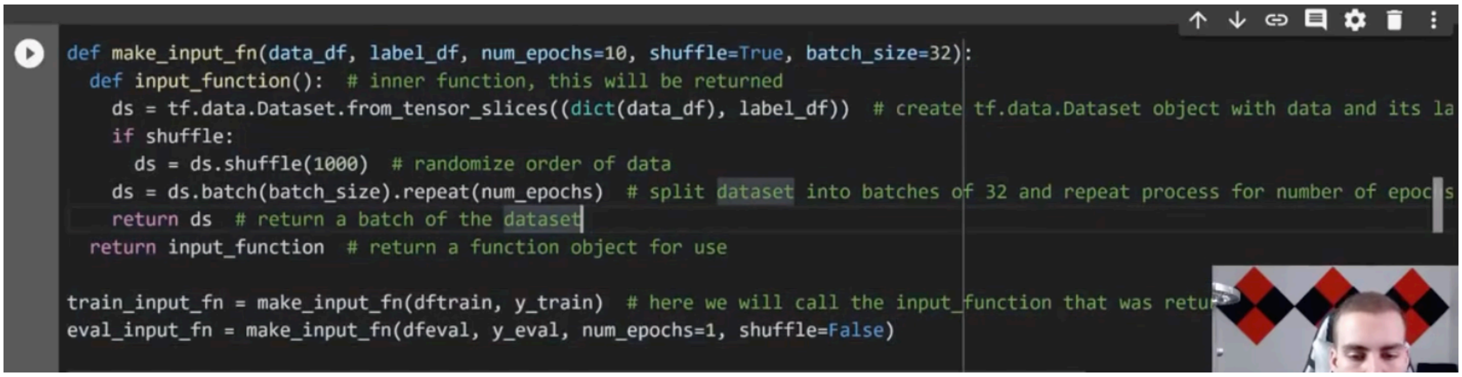
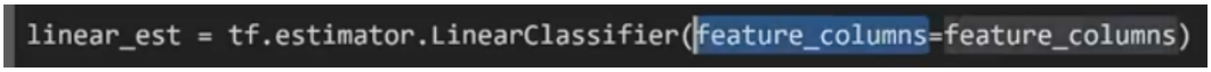
- ***Training a machine learning model***
  - -> for a linear model
  - -> you train the model
  - -> in this case there are 627 rows
  - -> loading the data in in batches to train the model
    - in this case it's batch sizes of 32
    - -> doing this increases the speed of the process
    - -> you're not loading the data all at once
  - -> epochs -> the number of times the model can see the same data
    - feeding the multiple data into the model
    - -> you can give it too much data -> so when you train the model it no longer works
    - -> start with a lower amount of epochs -> an apoch is a batch of data used to train the model - too many and the model might not work
- ***In this example***



```
def make_input_fn(data_df, label_df, num_epochs=10, shuffle=True, batch_size=32):
  def input_function():  # inner function, this will be returned
    ds = tf.data.Dataset.from_tensor_slices((dict(data_df), label_df))  # create tf.data.Dataset object with data and its la
    if shuffle:
      ds = ds.shuffle(1000)  # randomize order of data
    ds = ds.batch(batch_size).repeat(num_epochs)  # split dataset into batches of 32 and repeat process for number of epochs
    return ds  # return a batch of the dataset
  return input_function  # return a function object for use

train_input_fn = make_input_fn(dftrain, y_train)  # here we will call the input_function that was retu
eval_input_fn = make_input_fn(dfeval, y_eval, num_epochs=1, shuffle=False)
```

  - ***Creating / using different batches to train the model with***
    - -> the code is breaking down the data into epochs
    - -> there is an input function
    - -> the inputs
      - the pandas data frame
      - the number of epochs
      - if we are shuffling the training (input) data
      - the number of elements per epoch
    - -> he's passing a dictionary representation of the data frame -> from the label data frame
    - -> the tensors are sliced along different dimensions
    - -> then shuffling the dataset, splitting it into different batches of size 32
    - -> you don't just train the model - you pass different data of specific sizes into the model
    - -> then return the dataset
      - -> it's making an input function and returning the result
  - ***Then training the model***
    - -> then the train input function and the eval function based off of those batches of data
      - and telling it the number of batches to use
    - -> they he's training the model -> calling the input function
  - ***Creating the model***
    - -> passing the data through a linear classifier

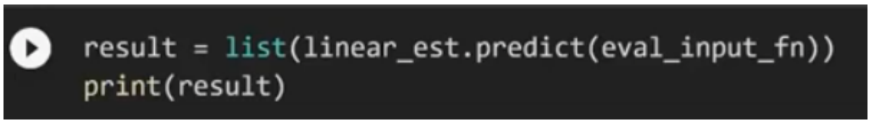


```
linear_est = tf.estimator.LinearClassifier(feature_columns=feature_columns)
```

      - -> this is creating the model for the linear estimator
  - ***Evaluating the model***
    - -> you can evaluate the model (rather than just train it) in this example -> to see how accurate the predicted values are compared to the actual ones
      - -> this prints the accuracy of the model in comparison to the data which showed whether the Titanic victims actually lived or not
    - -> each of the predictions has an accuracy level -> and it averages them for the entire

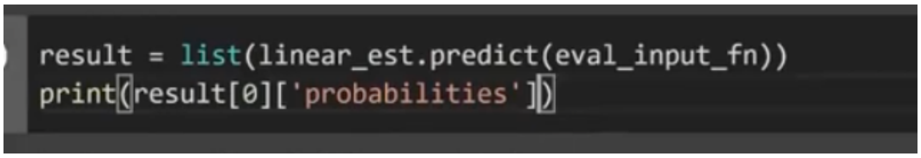
dataset

  - -> you can change the epoch (the batch of data used to train the model) or shuffle the data -> and then its accuracy will change
- ***To use the model***
  - -> tensor flow models make predictions on a lot of pieces of data
  - -> making predictions for every point in the dataset
  - -> results which the predictions gave vs the actual ones
  - -> the .predict() method
    - -> the arguments are the input functions from when the model has been trained
    - -> you need to train the model -> and then there is using it (this is using it)
    - -> training it is another process (picking the right model and batching the data to train it - then testing the accuracy on the test data)



```
result = list(linear_est.predict(eval_input_fn))
print(result)
```

  - -> you need to pass an input function to make a prediction
  - -> he's converted the entire thing into a list to loop through it -> and print the outputs
    - arrays, values, probabilities
    - the dictionary which represents the predictions for each value in the dataset
  - -> then to find the value of those predictions -> he's printing out the dictionary values for one prediction
    - -> each value it predicts has an entire dictionary value associated with it
    - -> to print out the probability the passenger doesn't survive, he's printed targeted the index of one of those elements _



```
result = list(linear_est.predict(eval_input_fn))
print(result[0]['probabilities'])
```

  - -> which he then compares to the actual result
  - -> he's printed out an example for a single passenger
  - -> epochs are the number of times the model will see the same data <- in this example it was when training a linear regression model or the Titanic data set