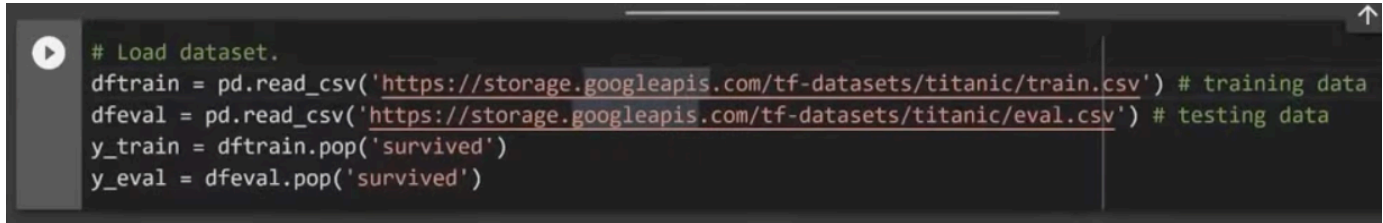


- **the dataset in this example**
 - -> the Titanic dataset
 - -> the probability someone will survive on the Titanic given the dataset
- **process**
 - -> he loads the Titanic dataset



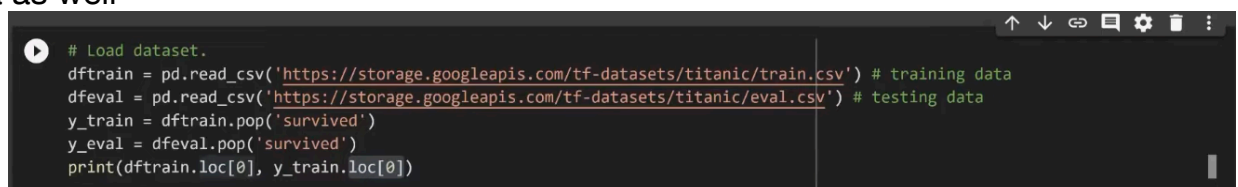
```
# Load dataset.
dftrain = pd.read_csv('https://storage.googleapis.com/tf-datasets/titanic/train.csv') # training data
dfeval = pd.read_csv('https://storage.googleapis.com/tf-datasets/titanic/eval.csv') # testing data
y_train = dftrain.pop('survived')
y_eval = dfeval.pop('survived')
```

- -> csv <- comma separated values
 - this is the format of the dataset
 - -> you can open these files in xls
 - -> there is a boolean value for if certain passengers lived or not
 - -> you can predict if they survived or not
 - -> he's using a linear regression model -> because looking at the variables involved you would think more of one thing would e.g result in a higher chance of surviving
- it's the Titanic dataset
 - -> their age, gender, class, deck, where they are going, if they are alone or if they came with other people
 - -> looking for patterns in the data - you can predict that increasing one variable would increase the others in a linear way
 - -> so he's guessing that the correlation is linear -> and then using a linear regression model
- -> there are two datasets
 - a training dataset
 - -> y_train <- it's taking off a section of the dataset to use for training the model
 - popping it off for this data frame separates the data which we are going to generate (whether the person survived the Titanic or not) from the rest of the data frame



```
# Load dataset.
dftrain = pd.read_csv('https://storage.googleapis.com/tf-datasets/titanic/train.csv') # training data
dfeval = pd.read_csv('https://storage.googleapis.com/tf-datasets/titanic/eval.csv') # testing data
y_train = dftrain.pop('survived')
y_eval = dfeval.pop('survived')
print(y_train)
```

- the indices for data which have been popped off are the same as the indices for data which haven't
- -> y_eval is for testing the model
- -> pd is a pandas data frame object -> storing the data in a spreadsheet form rather than a list / array etc
- -> dftrain.head() <- this shows the head of the data frame
- -> he's used dftrain.loc[0] <- locate row 0 -> and then he's done this on the testing data as well



```
# Load dataset.
dftrain = pd.read_csv('https://storage.googleapis.com/tf-datasets/titanic/train.csv') # training data
dfeval = pd.read_csv('https://storage.googleapis.com/tf-datasets/titanic/eval.csv') # testing data
y_train = dftrain.pop('survived')
y_eval = dfeval.pop('survived')
print(dftrain.loc[0], y_train.loc[0])
```

- -> this is the trained model -> the dataset is loaded into training and testing data sets
 - -> then the data we want to predict is popped off (the column which

- contains it is removed from the dataset)
- -> then he's printing out the values from a machine learning ran on the data using a linear regression approach -> in comparison to the true answers which were originally popped off
 - loc means -> locate row 0 in the dataset
 - -> you could also reference it via the key "age" e.g -> the name of the column (rather than the index of the row).
- a testing dataset
- -> other attributes you can use
 - -> `dftrain.head()` <- prints the head of the data frame
 - -> `dftrain.describe()` <- to print out summary statistics (mean, standard deviation, the number of entries in the dataset)
 - -> the shape of the dataset
 - -> `dftrain.shape` <- this prints out the rows by the columns in the data frame
- -> making graphs on the data
 - -> he's making histograms to see patterns in the data
 - a histogram of the ages of the passengers
 - most of them are in their mid 20's -> this can bias the linear correlation models
 - -> the gender ratio -> many more males
 - -> most people in third class
 - -> women had 4 times the survival rate
- -> .head() by default, shows the first five rows or entries in a data frame