Topic 1: Introduction to ML

Three reasons

1) Saves time

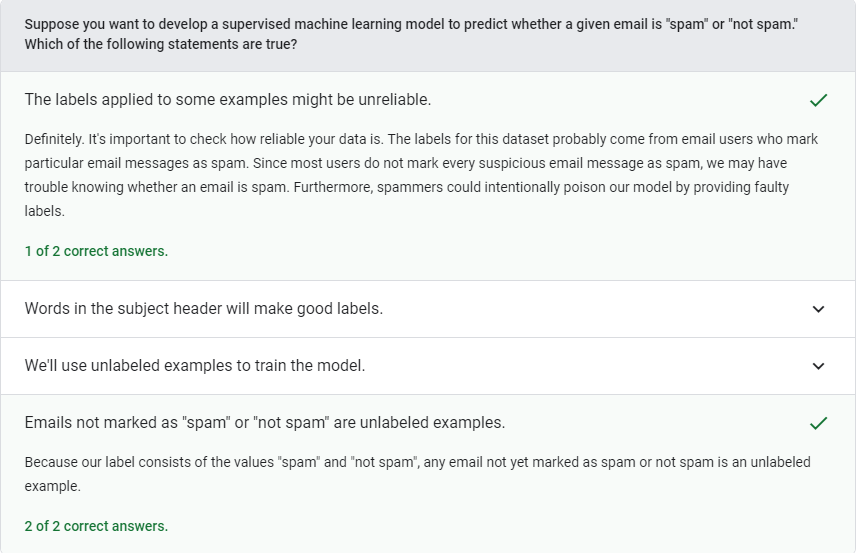
2) Reusability

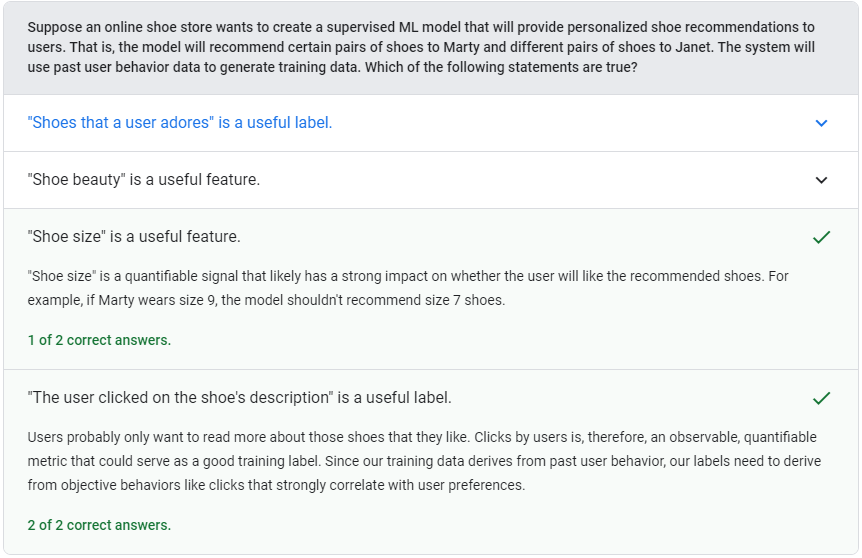
3) Can solve problems we have no idea how to approach by statistics and data

Topic 2: Framing

The video and reading material taught the basic definitions of terms like features (input data) labels (the thing we want to predict), regression and classification

Check your understanding:

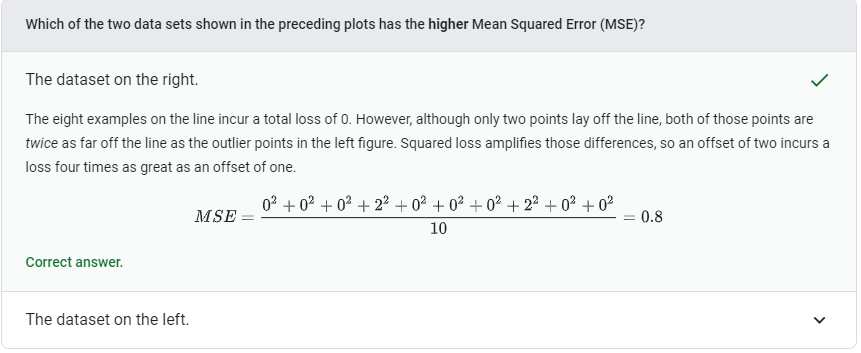




Topic 3: Descending into ML

In the video and the reading material we saw the basic function/model for linear regression, what are weights and biases, we saw mean squared loss function which is commonly used in machine learning but is in no way the best loss function for all circumstances

Check your understanding:



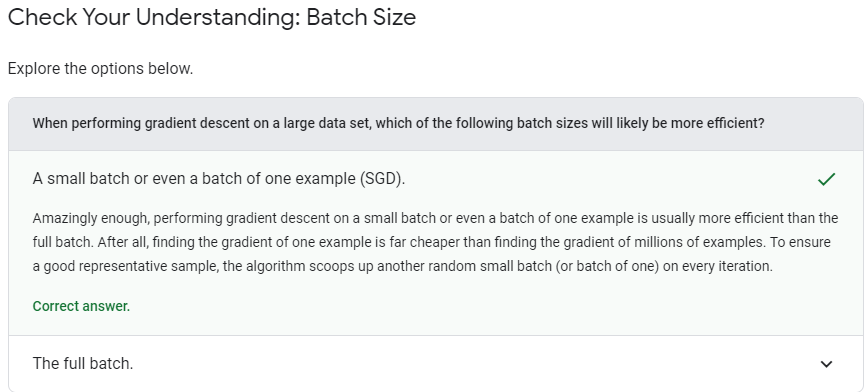
Topic 4: Reducing loss

In the video lecture we see the loss function which is convex usually for our linear regression problems but are known for being notoriously non-convex in convoluted neural networks. We also see in the end that we use scholastic or mini batch gradients to reduce computational power required

In the reading material we see the basic structure of how a machine learning algorithm works, with a model, loss function and a method of updating parameters. Then we see that a nice method of minimising the loss function is the gradient descent algorithm where we move in the direction of negative gradient, with a Goldilocks learning rate so as to achieve fastest reduction in loss without overshooting

We also see inverse of the Hessian matrix is used for getting goldilocks learning rate

Then I Played around with the given dataset



Topic 5: First steps with Tensor Flow

We get introduced to tensor flow API, and see its basic structure

Then we begin to learn tf.keras and see a linear regression model made by using that.

Then we tinker with the hyper parameters and see how they affect the training of the model

We see if the curve doesn’t flatten it means that the number of epochs or the learning rate is less, but if there is a large amount of oscillation it means that the learning rate is too high and the model will not converge properly

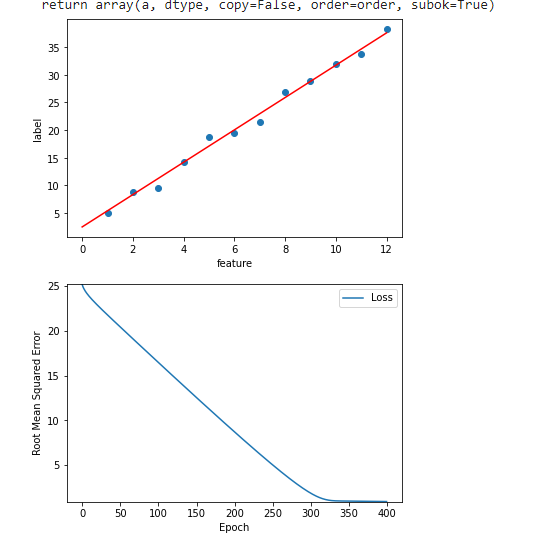
Then we see effect of batch size too and find the minimum batch size for which the model converges in 100 epochs

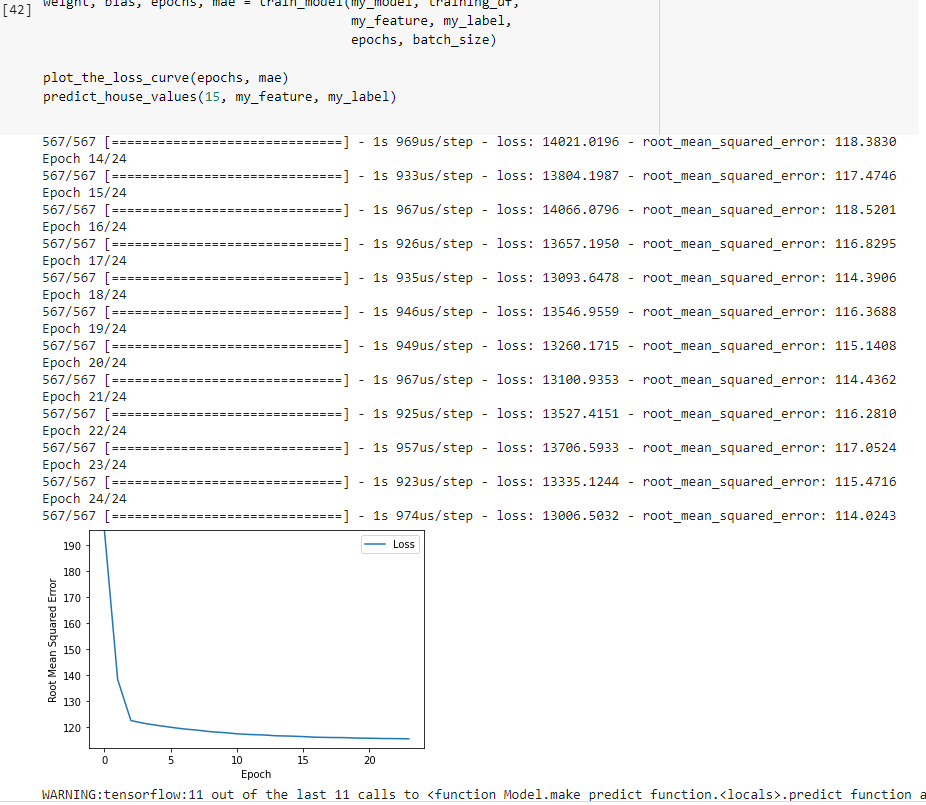
Final conclusion

Lowering the learning rate while increasing the number of epochs or the batch size is often a good combination.

Setting the batch size to a *very* small batch number can also cause instability. First, try large batch size values. Then, decrease the batch size until you see degradation.

Then we also saw correlation matrix and how it tells which features are probably more useful for predictions





Topic 6: Generalization

In the video we see visual examples of how over fitting leads to loss of the capability of the model to correctly judge new previously unseen data values

We also see that to prevent the above mentioned problem we must make the models less complex

Then we see the ML fine print, the basic assumptions that guide generalisation and when are they violated

The following three basic assumptions guide generalization:

We draw examples **independently and identically** (**i.i.d**) at random from the distribution. In other words, examples don't influence each other. (An alternate explanation: i.i.d. is a way of referring to the randomness of variables.)

The distribution is **stationary**; that is the distribution doesn't change within the data set.

We draw examples from partitions from the **same distribution.**

In practice, we sometimes violate these assumptions. For example:

Consider a model that chooses ads to display. The i.i.d. assumption would be violated if the model bases its choice of ads, in part, on what ads the user has previously seen.

Consider a data set that contains retail sales information for a year. User's purchases change seasonally, which would violate stationarity.

Topic 7: Training and Test sets

For our model to be generalized we divide it into two sets the training set and the test set

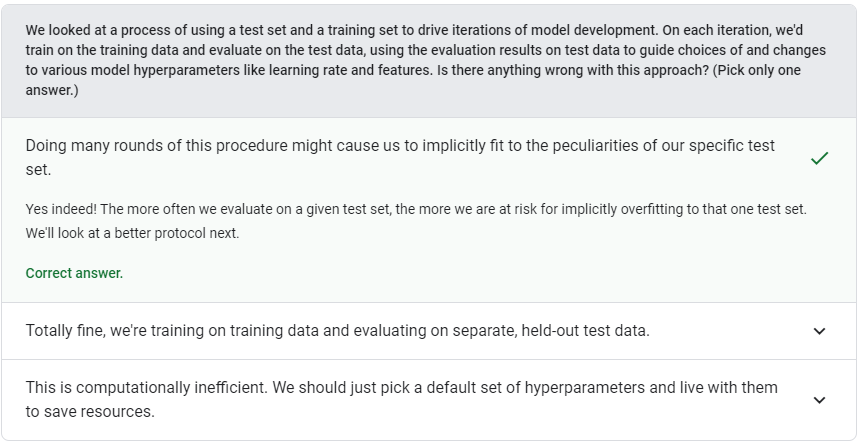
If the data set is large enough we can keep 10-15% for the test set else we have to perform cross validation or such

DO NOT TRAIN ON THE TEST SET (leads to unrealistic accuracy and is very misleading)

Then we play around in the playground exercise

Topic 8: Validation set

Check your intuition:



We see that if we keep testing out our model on the test set again and again and then tweaking it to reduce test loss we are as good as training the model for the peculiarities of the test set too which will lead to loss of generalization

The solution here is to make a third set the validation set, use it for repeatedly tweaking the model and then finally test the final model only once again the actual test data

This will ensure no over fitting

Then we play around with validation split in Validation and Test Sets.ipynb and saw some issues that might occur when we split the data (like if the data was sorted and not randomized etc.)

Topic 9: Representation

WE see feature engineering i.e. transforming the raw data into a feature vector

We then see all numerical variables must be stored as floats and all categorical variables are stored as one hot or multi hot encoding

We also see the sparse representation if there are values that never take a non-zero value

Next we see qualities of good features

* Avoid rarely used discrete feature values

### Prefer clear and obvious meanings

### Don't mix "magic" values with actual data

### Account for upstream instability

Lastly we see methods of cleaning data ,

Scaling to avoid Nan trap and get appropriate weights using Zscore or a linear map

Binning for variables having nonlinear graphs helps a lot. We treat the bins like categorical variables and apply one hot encoding. Making bins according to quartiles removes all problems caused by outliers

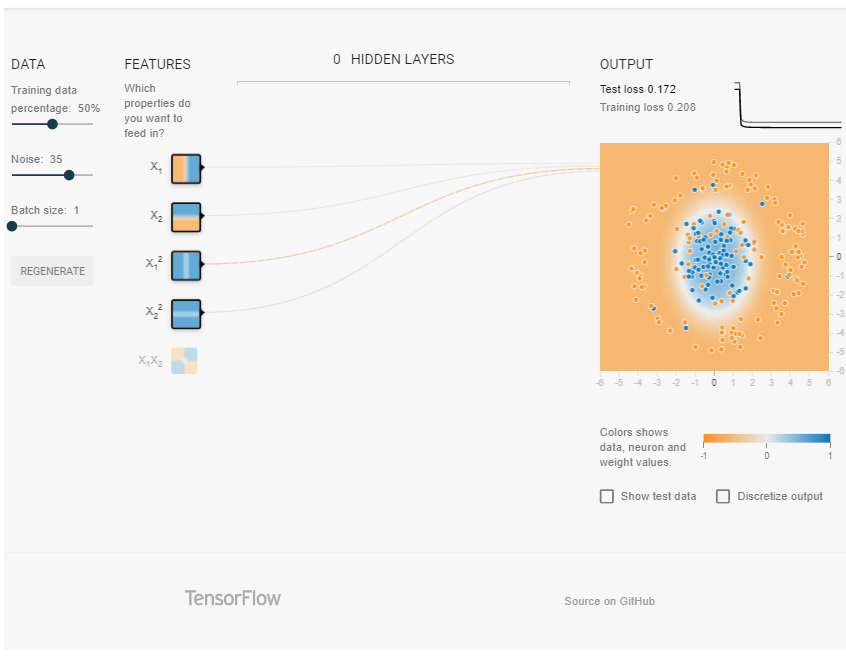
Scrubbing is done by handling duplicate/missing data by a simple program and by handling bad feature or bad labels with the help of histograms, mean, median, max, min, standard deviation etc. It’s a good practice to check if your data makes sense

Topic 10: Feature Crossing

In the video we see that feature crossing is extremely useful as it allows us to incorporate nonlinear relations into a linear model using a synthetic feature created by us. Supplementing scaled linear models with feature crosses has traditionally been an efficient way to train on massive-scale data sets.

Feature crossing is usually used for one hot coding vectors and not on continuously varying floating point numbers. The floating numbers can be binned to get a one hot coding vector and can then be crossed

Then we played with types of crossed features seeing how one could classify data looking like an ellipse



In the Google colab about feature crossing we learn about tf.feature\_column() and similar methods which are useful to transform and represent features

Then we see that crossing binned latitude and binned longitude to form a synthetic feature gives less loss than considering them as separate features

Check your understanding

