# MACHINE LEARNING

Table of Contents

[MACHINE LEARNING 1](#_Toc69309132)

[Machine Learning 2](#_Toc69309133)

[What type of regression model is right for me? 2](#_Toc69309134)

[Logistic regression 2](#_Toc69309135)

[Linear Regression 2](#_Toc69309136)

[Does my Linear Regression model work? 3](#_Toc69309137)

[Preprocessing 3](#_Toc69309138)

[Splitting categorical and numerical data 3](#_Toc69309139)

[Limit the number of correlations 3](#_Toc69309140)

[Pre-processing categorical data 4](#_Toc69309141)

[Pre-processing numerical data 4](#_Toc69309142)

[*Training the Machine* 5](#_Toc69309143)

# Machine Learning

**Essentially Machine learning is putting all factors in one view and finding the patterns through trial and error - > we can’t handle the complexity of this view, but a computer can.**

Machine learning is about predicting and categorising

Its maths but at higher levels of complexity than human brains can handle .

### What type of regression model is right for me?

**Types of Variable**

A independent variable is a known entity (x axis – eg. The size of the bill)

A dependent variable is on the y access. It depends on the independent variable for definition ( eg. Tip size. “For a bill of ??€ we expect a average tip of ??€). Hypothetically the tip size is driven by the bill size

**Which regression model is right for Continuous v. Countable v. Categorical data?**

**Linear regression** is good at predicting continuous data. But linear regression will not handle categorical data. All variables have to be numeric.

**Poisson regression** is well suited to countable (distinct) data. This is how to predict a count. It is not a straight line. It is a curvey line.

**Logistic regression** is best suited to categorical data. It predicts the answer to a yes no question (will person A survive the titanic disaster? Will person A default on their loan?). Puts together a bunch of largely categorical data and looks for patterns.

**Titanic example:**

The titanic challenge is a logistic regression challenge. Will this person survive – yes or no? Factors that predict outcome are ‘class of ticket’ ‘gender’ ‘age’ – basically all the criteria they used to allocate lifeboats

Logistic regression models the probability of an event – the target variable is binary.

Linear is more like a trend analysis – doesn’t work for a binary event.

## Logistic regression

Requires much greater number to inputs (liner could just be two variables). Then its just weighting those yes or nos on the different variables – which ones are the best predictors and how to combine them best in terms of predicting the outcome.

More on this topic?

## Linear Regression

**What is linear regression good for:** this is a good example. Working out an expected standard correlation that would not vary. – so ‘given these factors how likely is this outcome’   
Less suitable for a seasonality changing line. Or an exponentially growing line.

On the y axis you have this thing you are trying to predict

On the x axis you have multiple data points that the machine uses to assess the likelihood of the one outcome.

**EG: Trying to predict customer saving behaviours:**

Y axis = How much is a customer likely to save each month

X axis = what is their job, savings history, age, no of accounts and credit cards etc (ie lots of factors that the human eye could not take in in a glance.

It draws its linear regression line through masses of different data points

The way it does this is by training itself. – finding patterns that spot correlations – again that are beyond the capacity of the human mind.

### Does my Linear Regression model work?

How do we know if its doing a good job?  
- We look at the residuals. – the difference between the line and the data we have. SSE = sum of the squared errors. Goal is to get the line of best fit that minimises the sum of the square of the residuals. (the least squared method)

**Does it make sense?** It doesn’t have to make sense for every single value. It should just work for most points on the line.

We can think it through and for individual variables. But ultimately most models will be best fit line through many variables, in 3 diimensions. Something we cannot handle in our heads.

A scatter plot with a bunch of x axis.

We can make all our modelling better through good preprocessing.

## Preprocessing

We can get our data to come closer together / align better by a variety of different steps. We want our data points to get closer together so that we can more easily spot a pattern. But we don’t want them to get too close otherwise we might find that we have a model that ‘over-fits’ the data. You can visualise this as a line on a graph that goes through every single data point (v. the opposite, where a straight line goes through the data without having muc relevance to any of the data points: under-fitting)

Machine is comparing percentages and amounts and other things so it looks at the position of the data point – not the data. (eg: doesn’t matter if value is 6 or 8 when machine is comparing to a list of percentages. – what matters is that it is somewhere around a percentage point of 7%).

### Splitting categorical and numerical data

It’s important to recognise and handle categorical and numerical data differently. Typically you would split it out categorical and numerical data and clean/process them as separate data frames before re-joining them for analysis. You can split them using a piece of code like this:

num = data.select\_dtypes(include = np.number)

cat = data.select\_dtypes(include = np.object)

### Limit the number of correlations

data sets you input that are too similar – otherwise you are biasing the results towards one very similar set of values. IC 1,2,3 & 4 are too similar – probably should just pick one.

### Pre-processing categorical data

Some ideas:   
- Delete some categories that are not well populated.   
- Group categories so that they contain more similar ‘buckets’ of data. (example in class was donations to a charity by state. Some states had lots of doners, others only a handful. So you can group the ‘long-tail’ and end up with Categories like: ‘California’, ‘Texas’, ‘The NorthWest’, ‘LowDonerStates’. This would create more easily analysable categories and – as long as these categories are business relevant – then it is fine to do it. (in this case, why not. Our model can predict whether it is best to do more campaigning in the big states or try to grow the smaller states…) *note: Sian did this very easily in Tableau. Just highlighting the smaller states on the y axis, left clicking and clicking on the ‘group’ icon. She then got a prompt box to give the grouped categories a new name.*   
- Turn Categories from strings into numbers -> you need to do this with [One Hot Encoding](https://towardsdatascience.com/feature-engineering-for-machine-learning-3a5e293a5114#7c18) before you use a linear regression model (these models only take numeric variables)

### Pre-processing numerical data

Remember: we need to smooth out the data, clip off the edges and align the scales of our different variables to make them easier to compare. – easier to spot correlations/patterns within.

Some ideas:  
- **delete nulls** and empty cells and replace with: the mean, the mode, nothing (NaN). – this is really a data cleaning step, but….

- **remove outliers** We visualise our columns with histograms and boxplots to see the spread of the data and see where it lies across the interquartile ranges. Then we can remove the data points that lie too far outside these.   
A typical approach is to only allow data points that lie within 1.5 times the interquartile range (I think this means 1.5 x the iqr from the mean value – but need to check). Here is the code:

Calculating the interquartile range and setting limits:

iqr = np.percentile(data['monthly\_premium\_auto'],75) - np.percentile(data['monthly\_premium\_auto'],25)\nupper\_limit = np.percentile(data['monthly\_premium\_auto'],75) + 1.5\*iqr\nlower\_limit = np.percentile(data['monthly\_premium\_auto'],25) - 1.5\*iqr

Removing the outliers.

data = data[(data['monthly\_premium\_auto']>lower\_limit) & (data['monthly\_premium\_auto']<upper\_limit

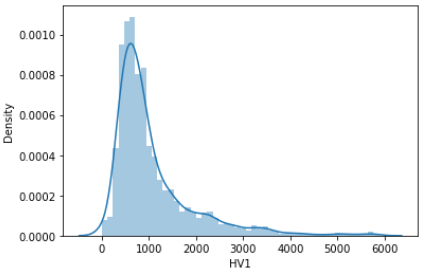
*-* **binning data.** *– the numerical equivalent of grouping (sort of. Actually used for categories as well) – think of the command bins = ?? in a histogram. You are just grouping the data into 5, 10s 20s or etwas…. So that it is easier to read and process.*

**#Numerical Binning ExampleValue Bin**   
0-30 -> Low   
31-70 -> Mid   
71-100 -> High

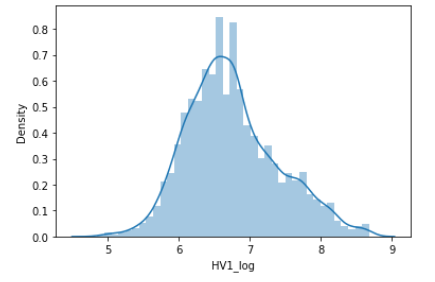
**#Categorical Binning ExampleValue Bin**   
Spain -> Europe   
Italy -> Europe   
Chile -> South America  
Brazil -> South America

\*see main article at the bottom for code

**- normailizing the data** This is an important step in linear regression modelling. We have to get the bell curve on the histograms to look smth like the same. This is usually achieved by some sort of logistical scaling (means squashing the axis scale together along the long-tail so that the it looks more like a normal bell curve and is not so skewed). This:



To this:



This is not as weird as disturbing as it looks. It is just de-weighting the outliers without removing them. We can overlay lots of charts that are so normalized and then spot patterns.

There are lots of normalization options and it is hard to understand them.

This python doc contains the ones that you did in class.

’FINAL.Linear Regression. Pre-Processing. Completed\_ML\_LR\_Students’

Here is a good overview: (scroll to bottom for nice visuals).

<https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html#sphx-glr-auto-examples-preprocessing-plot-all-scaling-py>

## Training the Machine

Training v. Test Data: To teach the machine we take a set of data that has the answers v all the factors. We take 80% of that data with the values in and use it to train the model. Then we take the other 20% to test the data. – this is before we apply it to unknown

## An excellent article on the topic, with code snippets!

[**https://towardsdatascience.com/feature-engineering-for-machine-learning-3a5e293a5114**](https://towardsdatascience.com/feature-engineering-for-machine-learning-3a5e293a5114)

Used this conversion tool

https://word2md.com/