DATA SCIENCE COURSE MATERIAL

Sneak a peek at the content of our data science training

MARCH 20, 2018 | THE CRUNCH ACADEMY

WHAT'S INSIDE?

Core modules of the training

1. Introduction to Data Science
2. Python programming basics
3. Python essentials for Data Science
4. Descriptive Analytics and Data Visualization using Python
5. Hypothesis Testing & Predictive Analytics
6. Predictive Analytics using Python and Scikit-Learn 101
7. Predictive Analytics using Python and Scikit-Learn 102

1. INTRODUCTION TO DATA SCIENCE

Learn what the driving forces in the data science ecosystem are

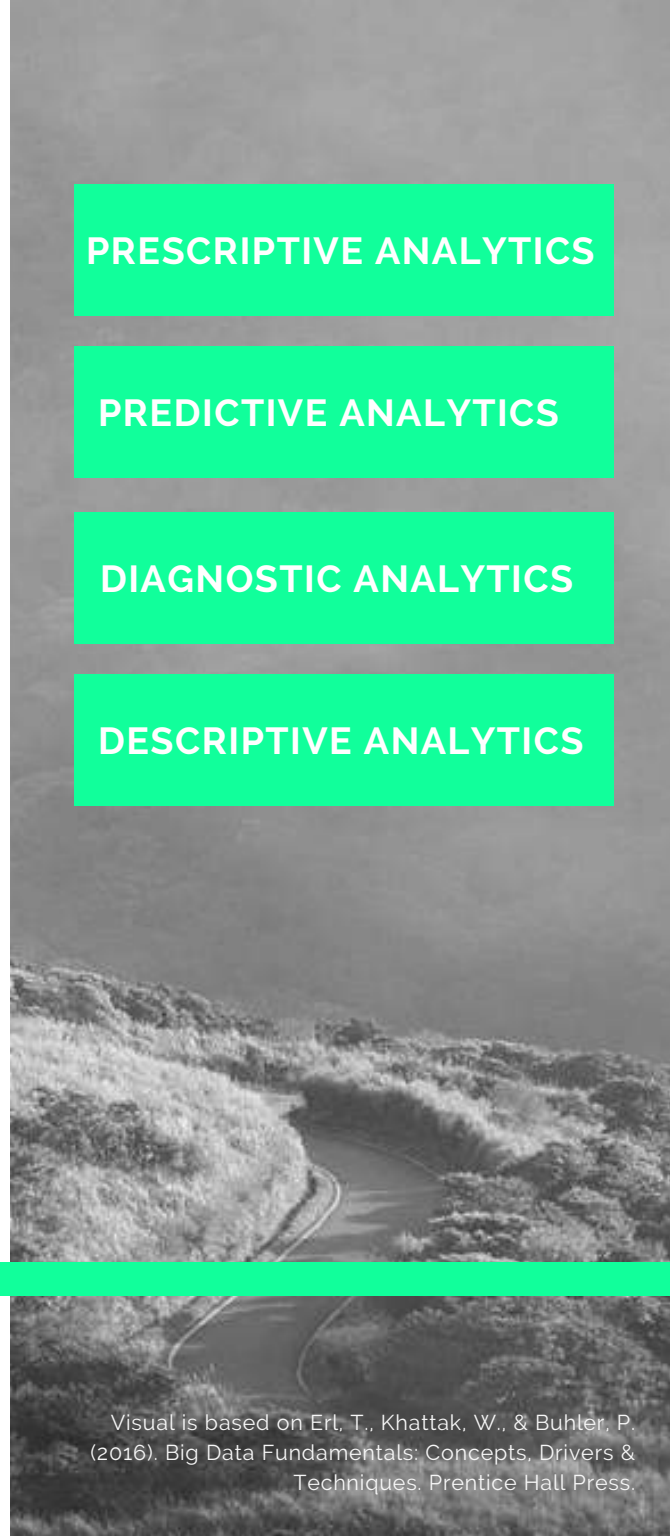
Every self-respecting data scientist has to know the basic principles. Hence, before we jump into the nitty-gritty details we give a solid overview of everything this space has to offer. This is accompanied with an introduction to the basic tools that will be used in the hands-on courses that will follow.

How is analytics evolving in the business perspective?

Making decisions using facts and figures rather than gut feeling has always been one of the focal points of top performing companies. Moving from traditional descriptive and diagnostic analytics to more advanced predictive and prescriptive techniques is a natural evolution.

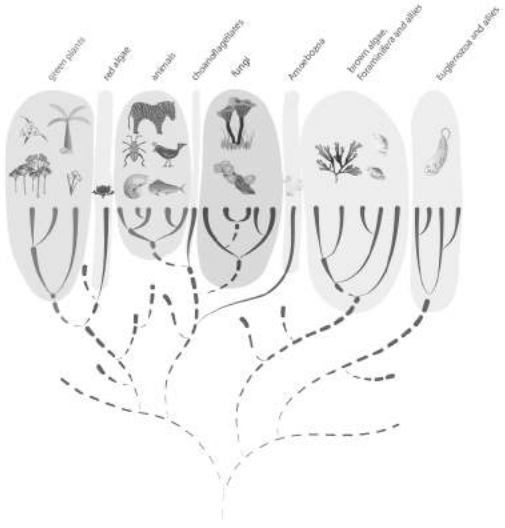
Today's businesses usually have some type of descriptive (what has happened?) and diagnostic (why has this happened?) analytics in place. The move towards a data scientific approach applies in large part moving to techniques that not only tell you about the past, but make predictions about the future.

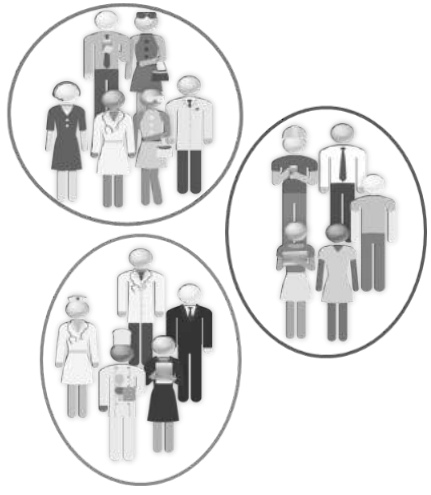
At the extreme end of this spectrum, companies are starting to give decision making power to algorithms (i.e. prescriptive analytics). While still a while off for the majority of companies, experimenting with these techniques is of paramount importance to stay ahead of the curve.



Visual is based on Erl, T., Khattak, W., & Buhler, P. (2016). Big Data Fundamentals: Concepts, Drivers & Techniques. Prentice Hall Press.

What are key Data Science applications and techniques?





CLASSIFICATION

Assigning observations to the correct category is one of the most frequently encountered problems in predictive analytics. Albeit customers with a high churn risk, recognizing hand written digits or detecting riots in images of crowds. This course starts out with logistics regression as the most basic of these techniques, and moves to incrementally complex techniques that allow you to solve highly complex problems.

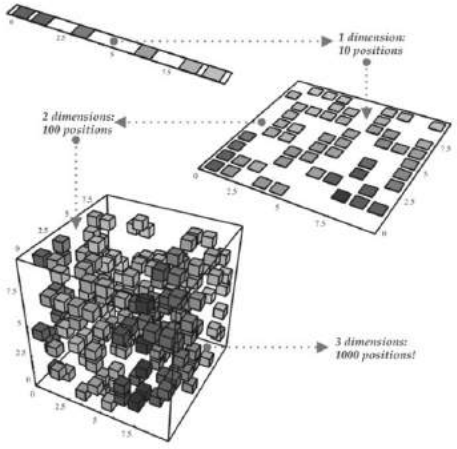
REGRESSION

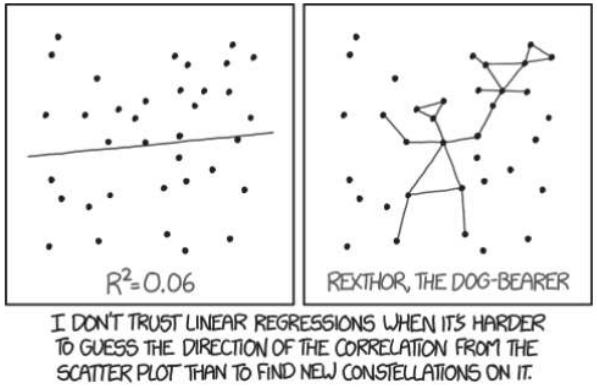
Competing for first place as the most frequently encountered problem is regression: try to predict a value on a continuous spectrum. Problems ranging from predicting the lifetime value of customers to the likelihood that a patient will respond positively to a medial treatment. Again we start with the basics in linear an polynomial regression, moving to support vector machines, tree-based methods and neural networks to tackle these problems.

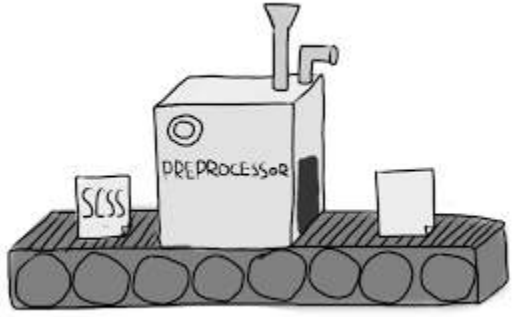
CLUSTERING

Many companies have excellent tools to communicate with segments of customers. However, these tools have no real impact if you don't know what these segments should be. Clustering techniques help you to see structure where there used to be none. Again starting with simple techniques like k-means clustering we build up towards the most recent and advanced techniques.

What are key Data Science applications and techniques?







DIMENSIONALITY REDUCTION

Lots of data is not always a blessing. For example in situations where the number of inputs is so excessive that any model starts to overfit. Fortunately dimensionality reduction techniques can help you to derive what is truly important from your dataset, giving you the exact right amount of variables. During these sessions we will cover both interpretable and non-interpretable techniques for doing this (virtual sensors).

MODEL SELECTION

Testing and validating models is a time consuming task. However, with the right tools and techniques it does not have to be! These sessions will show you how to use cross validations to set your parameters to their best possible values, and even testing multiple models in an automated way. This way you can be certain that the model you select is the best match for the problem you are trying to tackle.

PREPROCESSING

More than 80% of a data scientist's time is spent in the preprocessing phase. During the remaining 20% most data scientists with that they had spent more time on preprocessing. While we cannot solve this issue, we will teach you techniques that shave as much of this time off as possible, leaving you more time to spend on solving problems on a meaningful scale rather than on the level of the data quality.

2. PYTHON PROGRAMMING BASICS

Catch up on your programming skills

Without any doubt Python is the best programming language for data scientists today.

The primary reason for this is that all important packages are available for Python. The secondary reason is that Python makes putting models in production and communication with all kinds of systems easy. Not to mention that it is one of the fastest growing programming languages in general.


Time to get
down and dirty
with the data

DATA SCIENCE CHEAT SHEET

IMPORTING DATA

pd.read_csv(filename) – From a CSV file
pd.read_table(filename) – From a delimited text file (like TSV)
pd.read_excel(filename) – From an Excel file
pd.read_sql(query, connection_object) – Reads from a SQL table/database
pd.read_json(json_string) – Reads from a JSON formatted string, URL or file.
pd.read_html(url) – Parses an html URL, string or file and extracts tables to a list of dataframes
pd.read_clipboard() – Takes the contents of your clipboard and passes it to **read_table()**
pd.DataFrame(dict) – From a dict, keys for columns names, values for data as lists

EXPORTING DATA

df.to_csv(filename) – Writes to a CSV file
df.to_excel(filename) – Writes to an Excel file
df.to_sql(table_name, connection_object) – Writes to a SQL table
df.to_json(filename) – Writes to a file in JSON format
df.to_html(filename) – Saves as an HTML table
df.to_clipboard() – Writes to the clipboard

CREATE TEST OBJECTS

Useful for testing

pd.DataFrame(np.random.rand(20,5)) – 5 columns and 20 rows of random floats
pd.Series(my_list) – Creates a series from an iterable **my_list**
df.index = pd.date_range('1900/1/30', periods=df.shape[0]) – Adds a date index

VIEWING/INSPECTING DATA

df.head(n) – First n rows of the DataFrame
df.tail(n) – Last n rows of the DataFrame
df.shape() – Number of rows and columns
df.info() – Index, Datatype and Memory information
df.describe() – Summary statistics for numerical columns
s.value_counts(dropna=False) – Views unique values and counts
df.apply(pd.Series.value_counts) – Unique values and counts for all columns

SELECTION

df[col] – Returns column with label **col** as Series
df[[col1, col2]] – Returns Columns as a new DataFrame
s.iloc[0] – Selection by position
s.loc[0] – Selection by index
df.iloc[0,:] – First row
df.iloc[0,0] – First element of first column

DATA CLEANING

df.columns = ['a','b','c'] – Renames columns
pd.isnull() – Checks for null Values, Returns Boolean Array
pd.notnull() – Opposite of s.isnull()
df.dropna() – Drops all rows that contain null values
df.dropna(axis=1) – Drops all columns that contain null values
df.dropna(axis=1,thresh=n) – Drops all rows have have less than n non null values
df.fillna(x) – Replaces all null values with x
s.fillna(s.mean()) – Replaces all null values with the mean (mean can be replaced with almost any function from the statistics section)
s.astype(float) – Converts the datatype of the series to float
s.replace(1,'one') – Replaces all values equal to 1 with **'one'**
s.replace([1,3],['one','three']) – Replaces all 1 with **'one'** and 3 with **'three'**
df.rename(columns=lambda x: x + 1) – Mass renaming of columns
df.rename(columns={'old_name': 'new_ name'}) – Selective renaming
df.set_index('column_one') – Changes the index
df.rename(index=lambda x: x + 1) – Mass renaming of index

FILTER, SORT, & GROUPBY

df[df[col] > 0.5] – Rows where the **col** column is greater than 0.5
df[(df[col] > 0.5) & (df[col] < 0.7)] – Rows where 0.7 > **col** > 0.5
df.sort_values(col1) – Sorts values by **col1** in ascending order
df.sort_values(col2,ascending=False) – Sorts values by **col2** in descending order
df.sort_values([col1,col2], ascending=[True,False]) – Sorts values by **col1** in ascending order then **col2** in descending order
df.groupby(col) – Returns a groupby object for values from one column
df.groupby([col1,col2]) – Returns a groupby object values from multiple columns
df.groupby(col1)[col2].mean() – Returns the mean of the values in **col2**, grouped by the values in **col1** (mean can be replaced with almost any function from the statistics section)
df.pivot_table(index=col1,values= [col2,col3],aggfunc=mean) – Creates a pivot table that groups by **col1** and calculates the mean of **col2** and **col3**
df.groupby(col1).agg(np.mean) – Finds the average across all columns for every unique column 1 group
df.apply(np.mean) – Applies a function across each column
df.apply(np.max, axis=1) – Applies a function across each row

JOIN/COMBINE

df1.append(df2) – Adds the rows in **df1** to the end of **df2** (columns should be identical)
pd.concat([df1, df2],axis=1) – Adds the columns in **df1** to the end of **df2** (rows should be identical)
df1.join(df2,on=col1,how='inner') – SQL-style joins the columns in **df1** with the columns on **df2** where the rows for **col** have identical values. how can be one of **'left'**, **'right'**, **'outer'**, **'inner'**

STATISTICS

These can all be applied to a series as well.

df.describe() – Summary statistics for numerical columns
df.mean() – Returns the mean of all columns
df.corr() – Returns the correlation between columns in a DataFrame
df.count() – Returns the number of non-null values in each DataFrame column
df.max() – Returns the highest value in each column
df.min() – Returns the lowest value in each column
df.median() – Returns the median of each column
df.std() – Returns the standard deviation of each column Data Science

INFO
We'll use the following shorthands in this cheat sheet:
df - A pandas DataFrame object
s - A pandas Series object

FIRST THINGS FIRST!
Import these to start:
import pandas as pd
Import numpy as np

EXTRA
We give our Data Science graduates a more extensive cheat sheet! So join our training

4. DESCRIPTIVE ANALYTICS AND DATA VISUALIZATION USING PYTHON

First things first, know what you're working with!

The promise of artificial intelligent algorithms seems to tell people that smart algorithms and data are all you need to get results. The reality is that it still takes a substantial amount of work by expert data scientists to ensure that algorithms can perform.

The goal of this class is to give people more experience in making descriptive analyses using Python. This includes "simple" hypothesis testing, as well as the visualization of these results in an attractive manner. The visualizations for this part are created using Seaborn and the basic Matplotlib functionality of Python.

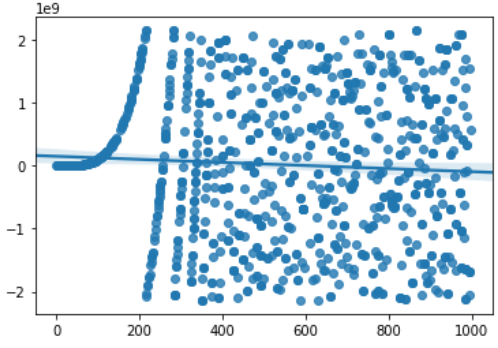
Basic statistics recap: Measure of Concordance

Even simple metrics like correlation can be highly deceiving. Obvious relationships between two variables like this equation: **Y = x^2 + x^3 + x^4** can result in correlation scores that seem to show no relationship whatsoever.

```
# We make a slight change in the relationship...
y = x**2 + x**3 + x**4
c = ss.pearsonr(x, y)[0]
print "Pearson correlation = " + str(c)
```

```
Pearson correlation = -0.0487307029656
```

```
a = sb.regplot(x, y)
```



-0.04

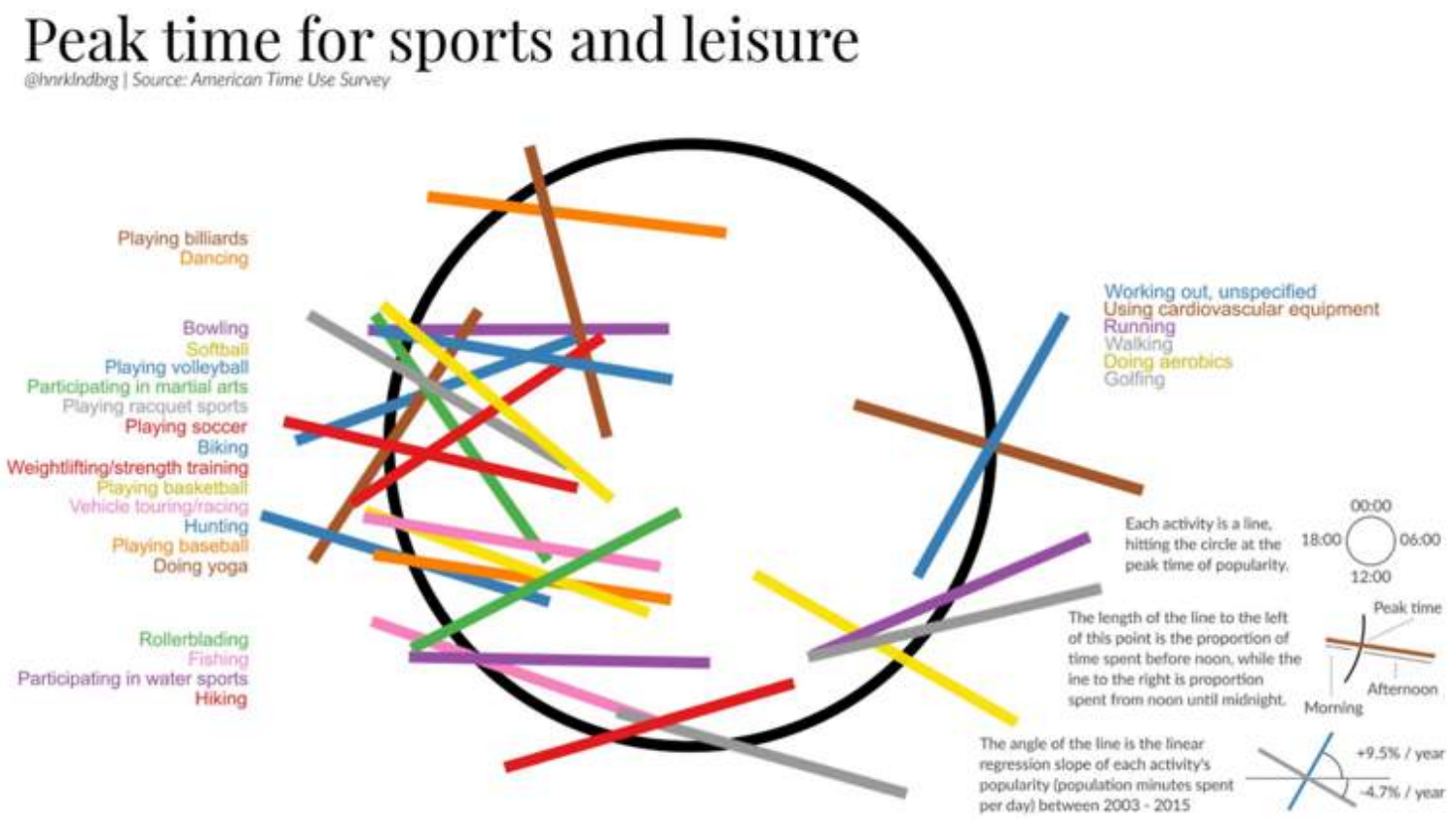
Seemingly there is no relation between these variables...

DON'T RELY SOLELY ON CORRELATION TO UNCOVER ALL RELATIONSHIPS BETWEEN YOUR VARIABLES

Tip & Tricks for Creating Effective Visuals

WHAT NOT TO DO:

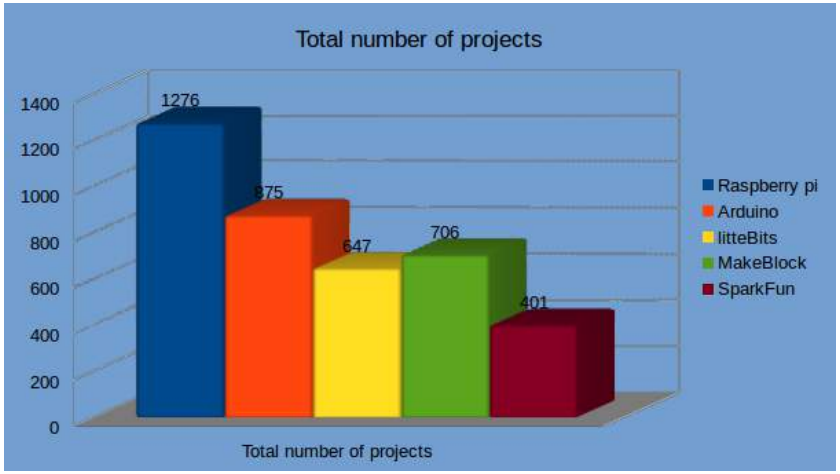
Do not use too many colors and/or categories

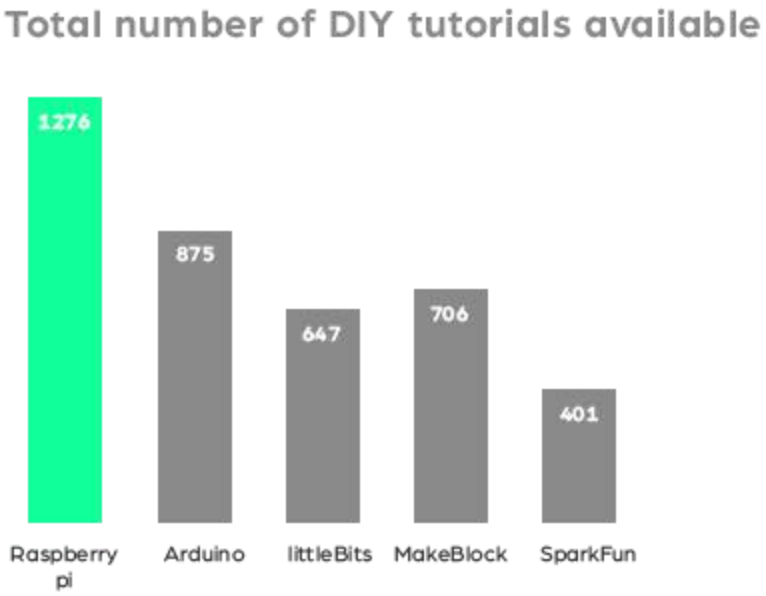


If your visualization needs a manual to understand, it is probably a bad idea...

WHAT TO DO:

Less is More!





The objective of a chart is to make numbers intuitive, make sure that a five year old is able to understand

5. HYPOTHESIS TESTING & PREDICTIVE ANALYTICS

Now it really begins!

This module is the first that moves into the domain of Machine Learning and creating some semblance of predictive models. A key learning in this session is the interrelationship between traditional statistics and machine learning and how these two domains have been moving more towards each other in recent years.

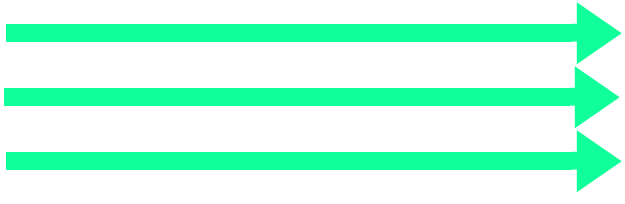
FROM SAMPLE TO POPULATION

SAMPLE

Mean
SD
Correlation
...

'Roman Alphabet'

Estimator?
Bad luck?



Data - Model
Reality - Theory
Observations - Predictions

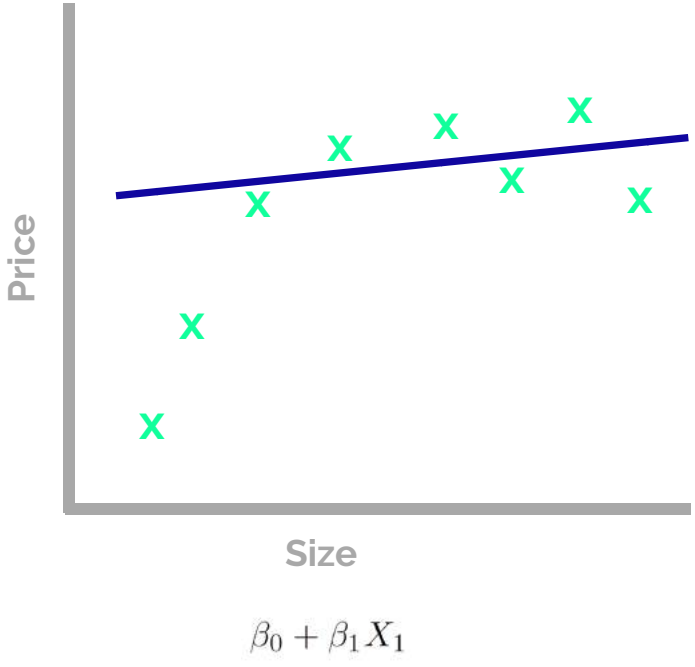
POPULATION

Mean
SD
Correlation
...

'Greek Alphabet'

MACHINE LEARNING PROBLEMS

UNDERFIT



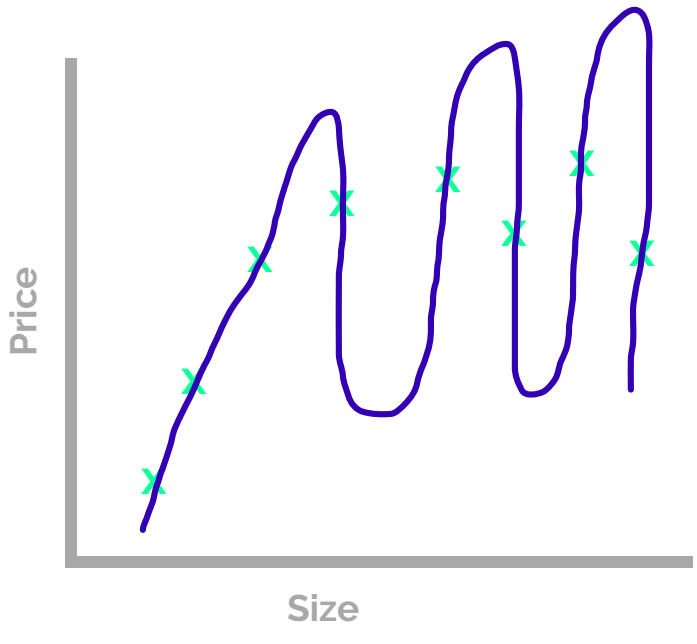
When a model underfits it is unable to capture the full complexity of the situation. This means that a model will perform poorly when you rely on it to make predictions.

The solution in this case is to increase the complexity of the model to hope to capture the full complexity in the situation.

- Predict known data bad (high bias)
- MSE of training set is high
- Bad! -> increase complexity

MACHINE LEARNING PROBLEMS

OVERFIT



$$\beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$$

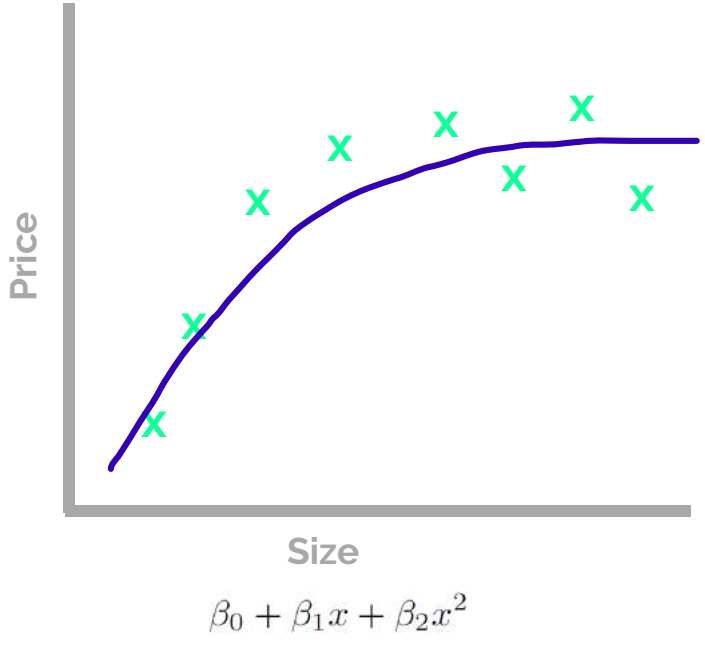
Overfitting typically happens when you use a model that is too complex for the data at hand. Especially in situations where the data is quite complex it can be nontrivial to detect that your model is overfitting. In a sense overfitting is also more dangerous than underfitting since you will expect your model to perform substantially better than it will actually do in practice (whereas in the underfitting case at least you will not have unrealistic expectations when it comes to model performance).

Using a correct test design using cross-validation can help you select a model that is not too complex for the data at hand, and prevent overfitting.

- Predict known data 'too well' (low bias)
- Predict unknown data bad (high variance)

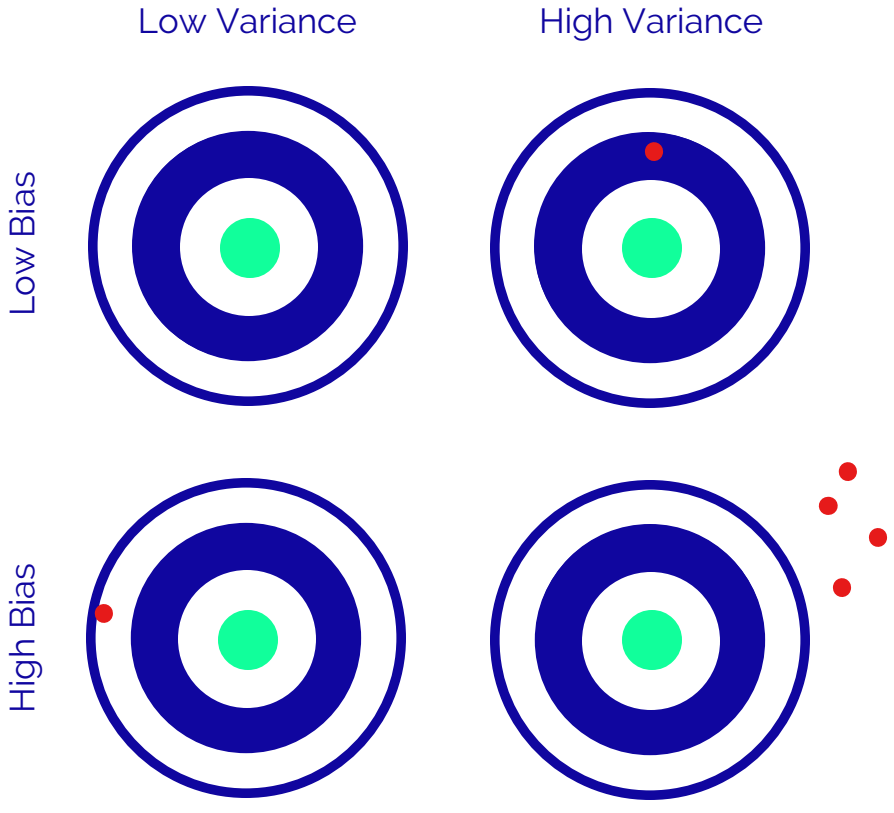
MACHINE LEARNING SOLUTION

BALANCE



A balanced model is what every data scientist should strive to create. This model does not overfit (too much) and captures the actual value from the data.

BIAS - VARIANCE TRADE-OFF



Low bias + low variance: the optimal model that hits close to home every time

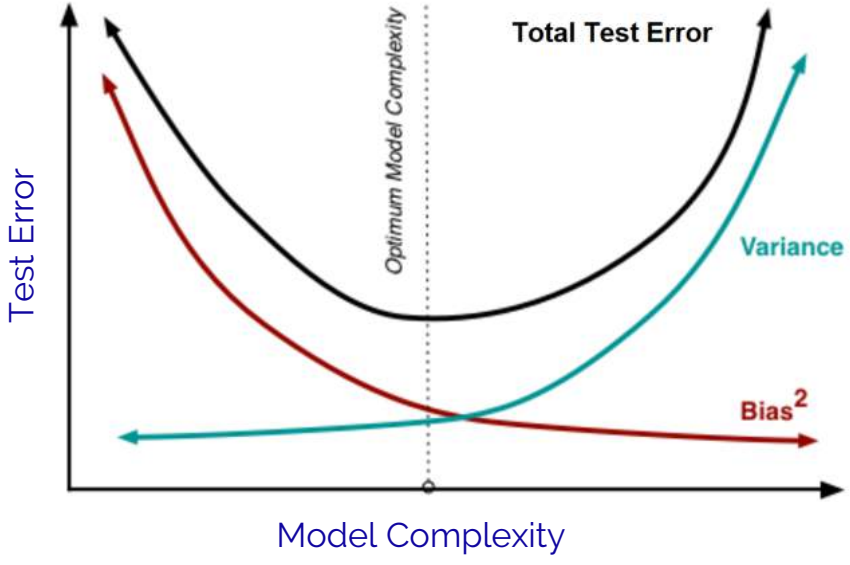
Low bias + high variance: your model is right on average, but can be quite far off target for specific observations.

High bias + low variance: your model is performing consistently, but it is consistently wrong.

High bias + high variance: your model is both off target and contains substantial amounts of noise.

An important note is that high bias is a sign of underfitting, whereas a high variance is a sign of overfitting. As such there is an ideal balance to be found where a model is both as accurate and consistent as possible. Where this balance lies depends on the specifics of the data, but in large part also on the specifics of what you want to do with the model.

BIAS - VARIANCE TRADE-OFF



Working with a training and a test set is one of the simplest ways of preventing overfitting. As the complexity of a model increases the error in both training (= the data that the model is shown) and the test set (= the data that is held back from the model to test its performance) will decrease.

However at some point along this complexity axis it will become apparent that increasing model complexity only improves performance on the training data and the performance on the testing data is starting to deteriorate. At this point in time you are overfitting.

6 & 7. PREDICTIVE ANALYTICS USING PYTHON AND SCIKIT-LEARN 101 & 102

Let's do this!

Are you ready to dive fully into the domain of Data Science? Sklearn contains tools to make your life as a Data Scientist as easy as possible!

FOR DUMMIES: HOW TO BUILD A MODEL IN SKLEARN

STEP BY STEP

1. What do you wish to model?

- (Un)Supervised (see next slide)
- Datatypes?

2. Now look at your data.

3. Determine your modeling technique.

4. If needed, transform variables.

5. Start building your datasets in the following order:

- Train
- Test
- Validate

6. Train your model.

7. Evaluate the model on your training set.

- Check for underfit

8. Test your model

9. Evaluate the model on your test set.

- Check for overfit

10. Determine your final model

11. If possible, validate on validation set

12. Interpret your model!

That's it! Try it out!

SKLEARN MODELS OVERVIEW

SUPERVISED

- Linear regression
- Polynomial regression
- Ridge Regression
- Logistic regression
- Random forest
- Support Vector Machine (see next slide)

UNSUPERVISED

- Clustering
- PCA

What is SVM?

Abbreviation? S(upport) V(ector) M(achine)

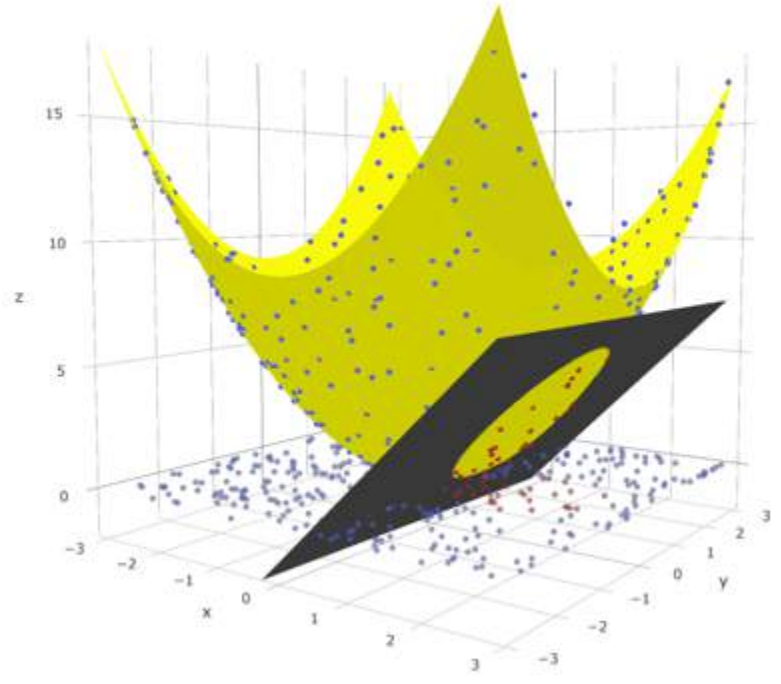
What is it used for? Classifying/clustering algorithm

The options depening on your kernel:

- 'Simple' algorithm (linear)
- **Complex algorithm (non-linear)**

How does it work?

Transform (non-linear) into higher dimensions
Find best hyperplane borders
Transform back



http://efavdb.com/svm-classification/

SKLEARN VALIDATIONS OVERVIEW

METHODS

1. **Simple procedure:** random training and test set
2. **Advanced procedure:** stratified training and test set
3. **Even more advanced procedure:** k fold cross validation

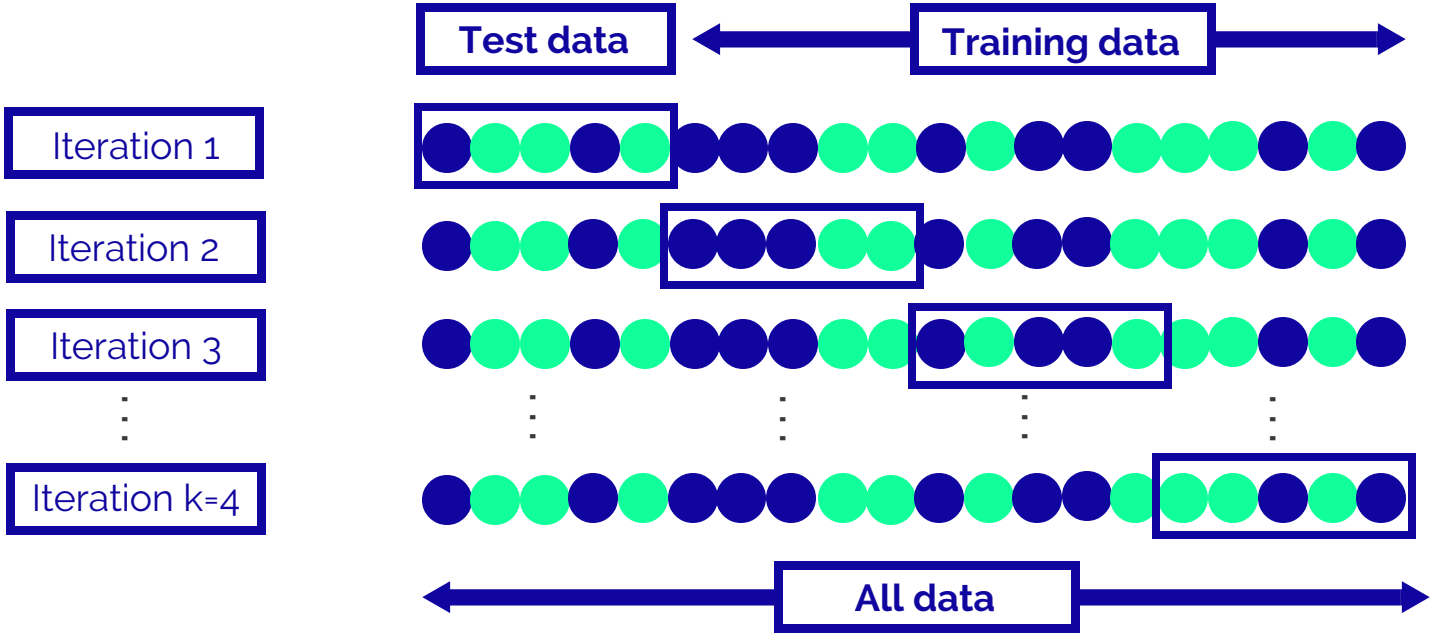
What is K Fold Cross Validation?

What is it?

Cross Validation is a very useful technique for assessing the performance of your models. It helps in knowing how the model would generalize to an independent data set.

When should you use this?

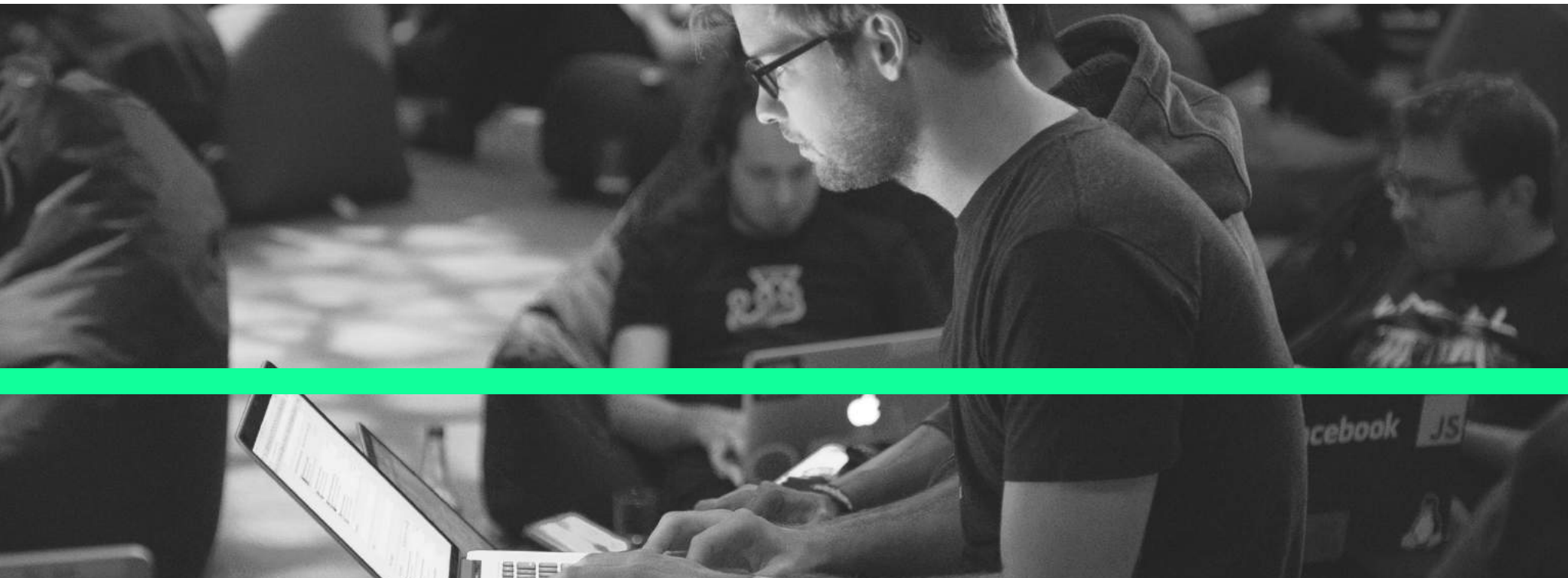
You want to use this technique to estimate how accurate the predictions your model will give in practice.



ADVANCED MODULE HIGHLIGHTS

Feed your curiosity

Get inspired by the more advanced features of Data Science and use this knowledge to keep on challenging yourself to learn more. Your basics are on point, now it's up to you!



Test your Data Science model

Test how good or bad your model will perform in practice by means of our stress test

Two objectives:

Primary objective	The model is able to handle issues automatically and keep functioning
Secondary objective	The model warns you when it cannot handle things on it's own anymore

Optimizing your neural network

Loss/cost function has a slope?

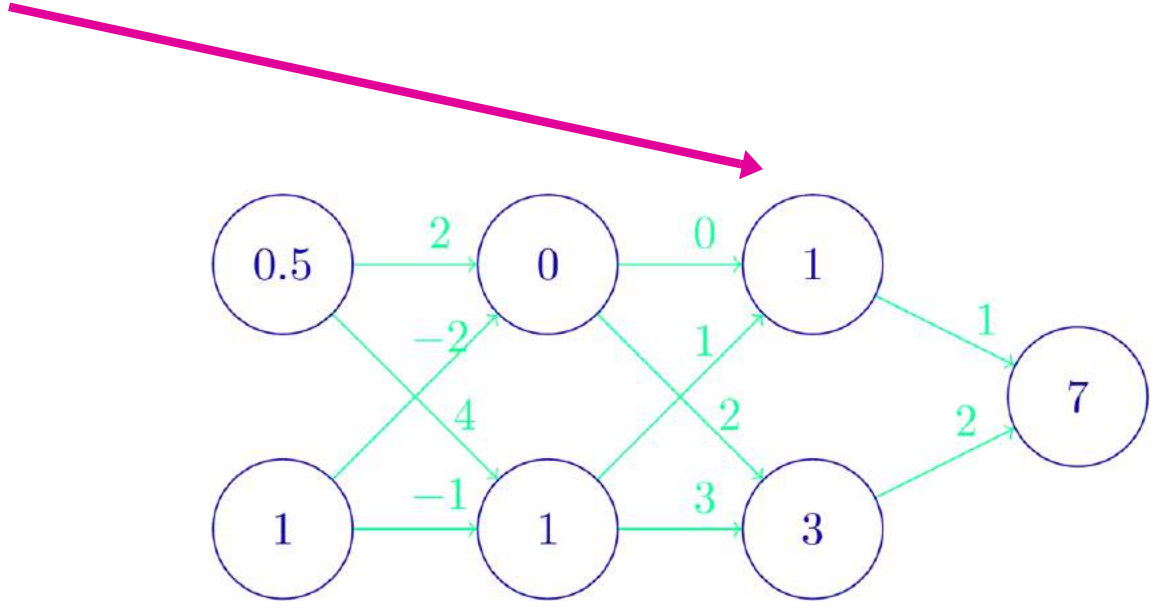
- Gradient (derivation)

Step down the slope

- Change w1
- Change w2
- Change w1?

Only way is up

- **--> Optimized!**



$$\text{MSE} = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}$$

NLP: The Good & The Bad

Very Good

- Spam
- Recognizing parts of sentences

  Verbs, nouns,...
  Locations
  Names
  ...

Moderate

- Translating
- Sentiment analysis
- Extracting "unexpected" components (à la Siri)

Painstakingly Bad

- Interpreting complex speech
- Having a true conversation (Turing test)
- Recognizing plagiarism (when paraphrasing)

CURIOUS ABOUT THE TRAININGS?

Download our training overview **here**

1. The Machine Learning Academy for programmers
2. Data Science Course for Analysts
3. Data Science for Business