Lesson 1: Machine Learning:

https`://classroom.udacity.com/courses/ud120/lessons/2410328539/concepts/24185385370923

Aanbevolen statistics classes:
https://eu.udacity.com/course/intro-to-descriptive-statistics--ud827

https://eu.udacity.com/course/intro-to-inferential-statistics--ud201

Source staat nu (geconverteerd naar Python 3) op HAN GitHub:
https://github.com/HANICA/MachineLearningP3

Lesson 2: Supervised learning (Markov models)

DEF **Supervised learning** learning waarbij je hebt al een idee hebt over wat het goede antwoord zou moeten zijn.

DEF **Decision surface**: scheiding tussen factoren uit je beslissingsmodel.

1.17 Sk learn naive bayes with python.

http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

Ik had sklearn nog niet: pip3 install sklearn
Numpy had ik al, anders pip3 install numpy

fit(X, Y)
X -> features
Y -> labels

19

In ClassifyNB.py:

```
def classify(features_train, labels_train):

    from sklearn.naive_bayes import GaussianNB
    ### import the sklearn module for GaussianNB
    ### create classifier
    ### fit the classifier on the training features and labels
    ### return the fit classifier

    ### your code goes here!

    clf = GaussianNB()
```

```
    clf.fit(features_train, labels_train)
    return clf
```

DEF **Accuracy**: number of points classified correctly / all points in test set

Accuracy in sklearn:
score methode

20: Calculating NB Accuracy

```
def NBAccuracy(features_train, labels_train, features_test, labels_test):
    """ compute the accuracy of your Naive Bayes classifier """
    ### import the sklearn module for GaussianNB
    from sklearn.naive_bayes import GaussianNB

    ### create classifier

    clf = GaussianNB()

    ### fit the classifier on the training features and labels
    clf.fit(features_train, labels_train)

    ### use the trained classifier to predict labels for the test features
    pred = clf.predict(features_test)


    ### calculate and return the accuracy on the test data
    ### this is slightly different than the example,
    ### where we just print the accuracy
    ### you might need to import an sklearn module
    accuracy = clf.score(features_test, labels_test)

    return accuracy
```

alternatief:

```
from sklearn import accuracy_score
print(accuracy_score(pred, labels_test))
```

21: training en testing op meerdere datasets, doe je dat niet krijg je wellicht last van *overfitting*

DEF **Overfitting**: thinking that you know better what's going on than you actually know.

Tip: gebruik 90% van je data om je model te trainen en dan 10% van je data om te testen.

Probabilistic Inference: Bayes Rule

41: Ik kreeg wat issues met project (python2 en python3 die met elkaar in de clinch liggen), ik heb daarom https://docs.python.org/2/library/2to3.html gebruikt om startup.py te converteren. Helaas is ook de dataset niet correct… (enron_mail_20150507)

Fix: url = "http://zoo.cs.yale.edu/classes/cs458/lectures/sklearn/ud/ud120-projects-master/enron_mail_20150507.tgz"

B.t.w. Cloudpickle.py had ook een issue, imp is deprecated, library imp vervangen door importlib lijkt te werken

Ik converteer alle code naar Python 3 en zet hier bijgewerkte versie neer: https://github.com/HANICA/MachineLearningP3

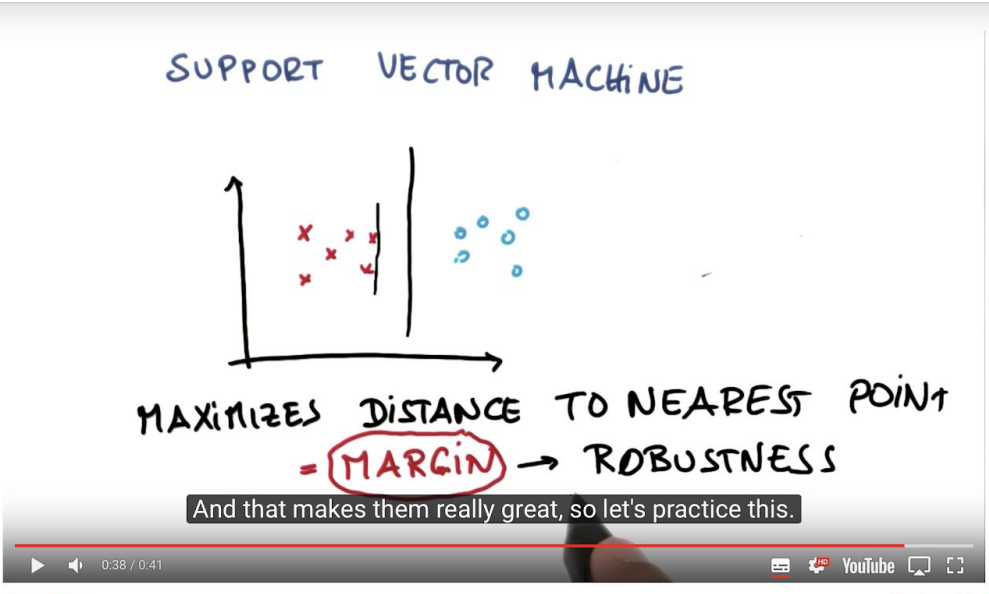
Lesson 3: Supervised learning (Support Vector Machine)

Margin: afstand tussen scheidingslijn en dichtstbijzijnde punt(en) van de twee klassen (van punten)

SVM maximizes distance to nearest point, also called margin.

Terms: margin, robustness, noise

Als marge verder van punten af ligt zal noise minder snel de label van een punt naar ‘verkeerde kant’ laten flippen (meer robuust).



Inside of SVM is to maximize robustness of your result.

SVM
Prio 1: correcte klassificatie
Prio 2: maximaliseren van marge

Individual outliers will be tolerated even if they are then incorrectly classified.
SVM is somewhat robust to outliers.

12: http://scikit-learn.org/stable/modules/svm.html

```
import sys
from class_vis import prettyPicture
from prep_terrain_data import makeTerrainData

import matplotlib.pyplot as plt
import copy
import numpy as np
import pylab as pl


features_train, labels_train, features_test, labels_test = makeTerrainData()


########################## SVM #################################
### we handle the import statement and SVC creation for you here
from sklearn.svm import SVC
clf = SVC(kernel="linear")


#### now your job is to fit the classifier
#### using the training features/labels, and to
#### make a set of predictions on the test data
clf.fit(features_train, labels_train)


#### store your predictions in a list named pred
pred = clf.predict(features_test)

from sklearn.metrics import accuracy_score
acc = accuracy_score(pred, labels_test)


def submitAccuracy():
    return acc
```

19 Term *Kernel trick*: distuingishing between datasets using non-lineair division lines (translating x,y to multi dimensional space and back) -> non lineair separation

Applying kernel tricks:
http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

SVM C param: controls tradeoff between *smooth decision boundary* and *classifying training points correctly*
Grotere C -> meer correcte punten, complexer model (minder *smooth*)

γ (gamma) high values of gamma result in more wiggly (less smooth) decision boundaries and low values of gamma result a more smooth (less wiggly) decision boundaries

24. Overfitting: you don't want very wiggly decision boundaries as they will correctly represent / classify your training data but will be possibly bad in classifying new examples. Using the C and γ (gamma) and kernal parameters we can control the amount of overfitting: tuning is wel belangrijk!

SVM: langzaam op grote datasets, ook minder robuust dan Naive Bayes bij veel ruis (noise) tegen overfitting.

> C: higher accuracy
clf = svm.SVC(C=10000.0, kernel='rbf', gamma='auto')

no. of Chris training emails: 7936
no. of Sara training emails: 7884
Fit (training)...
training time: 0.118s
Predict...
prediction time: 1.06s
Accuracy...
accuracy: 0.8925

Lesson 4: Supervised learning (Decision Trees)

http://scikit-learn.org/stable/modules/tree.html
http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Quiz 8:

```
#!/usr/bin/python

""" lecture and example code for decision tree unit """

import sys
from class_vis import prettyPicture, output_image
from prep_terrain_data import makeTerrainData

import matplotlib.pyplot as plt
import numpy as np
```

```
import pylab as pl
from classifyDT import classify

features_train, labels_train, features_test, labels_test = makeTerrainData()



### the classify() function in classifyDT is where the magic
### happens--fill in this function in the file 'classifyDT.py'!
clf = classify(features_train, labels_train)

clf.predict(features_test)


#### grader code, do not modify below this line

prettyPicture(clf, features_test, labels_test)
output_image("test.png", "png", open("test.png", "rb").read())

# classifyDT.py

def classify(features_train, labels_train):

    ### your code goes here--should return a trained decision tree classifer

    clf = tree.DecisionTreeClassifier()
    clf.fit(features_train, labels_train)

    return clf
```

4.9 Ik heb dit apart gedaan vanuit Python op basis van het dt_author_id_jk.py bestand.

http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

min_sample_split van Decision Trees geeft aan hoe ver de boom uitgewerkt wordt, te ver uit laten werken (default = 2) resulteert in overfitting. Accuracy gebruiken om te kijken welke setting optimaal zou kunnen zijn.

Entropy: measure of impurity (contamination) in a bunch of examples

Trying to make subsets that are as pure as possible

Formule voor entropy:

$$H = -\sum_i p_i(\log_2 p_i)$$

(n.b. Foutje in materiaal (- ontbreekt)

Alle samples uit zélfde class dan is entropy == 0 (wiskundig minimum)
Alle samples verdeeld over twee gelijke classes dan is entropy == 1.0 (wiskundig maximum)

ffss (2x fast, 2x slow)

p_i is dan totaal gedeeld door aantal van klasse = 0.5 voor zowel f als s.

p_{fast} = 0.5

p_{slow} = 0.5

entropy = 1.0

Commandline: Python interpreter

```
Python 3.6.5 (default, Jun 11 2018, 12:03:31)
[GCC 4.2.1 Compatible Apple LLVM 9.1.0 (clang-902.0.39.2)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from math import log
>>>
>>> -0.5*log(0.5, 2) -0.5*log(0.5, 2)
1.0
>>>
```

Dus is de entropy 1.0 (maximum impurity)

Information gain = entropy (parent) - [weighted average] * entropy (children)

decision tree algorithm : maximize information gain

4.26 >>> -0.667*log(0.667, 2) -0.333*log(0.333, 2)
0.9179621399872385

4.27 1 - ¾ (0.918)

4.30 >>> -1.0*log(1.0, 2) -1.0*log(1.0, 2)
-0.0

4.32 gini is another measure for purity (alternative to entropy)

4.33 bias-variance dilemma

Lesson 5. Alternatieven

k nearest neighbors

random forest

adaboost (sometimes also called boosted decision tree)

Steps to take:

- Research / understand algorithm
- Get it running in sklearn
- Run it on the terrain classification data
- Quantify the accuracy

Predict moet features_test gebruiken (even overal checken, soms heb ik per ongeluk de train data gebruikt geloof ik) eigenlijk checken in source files uitwerkingen 1 .. 4

ToDo: ik heb plotten van data na analyse nog niet goed werkend…

Accuracy voor your_algorithm_decisionTrees.py is 0.908
Accuracy voor your_algorithm_kMeans.py is 0.3 // Not too good, probably needs more tuning :)

Solution: In fact K-Means is an algorithm for *unsupervised learning* see Lesson 9.

Accuracy voor your_algorithm_kNearestNeighbor.py is 0.928
Accuracy voor your_algorithm_Adaboost.py is 0.0 // Hmm, probably needs more tuning, seems that Adaboost works a little different :)
Accuracy voor your_algorithm_RandomForest.py is 0.3 // Not too good, probably needs more tuning :)

Note I did not do any tuning. First step would be to get a 'visual' and then try to fiddle with the parameters.

Lesson 6. Datasets

In general: more data means better fine-tuned algoritm.

enron_mail_20150507.tar.gz staat ook op de GitHub van HANICA

Datatypes zijn vrij simpel voor 'ons' informatici, hooguit "job titles" die door de docenten als categorisch gezien worden (maar in feite text bevatten.

Lesson 7. Regressions (Continuous Supervised Learning)

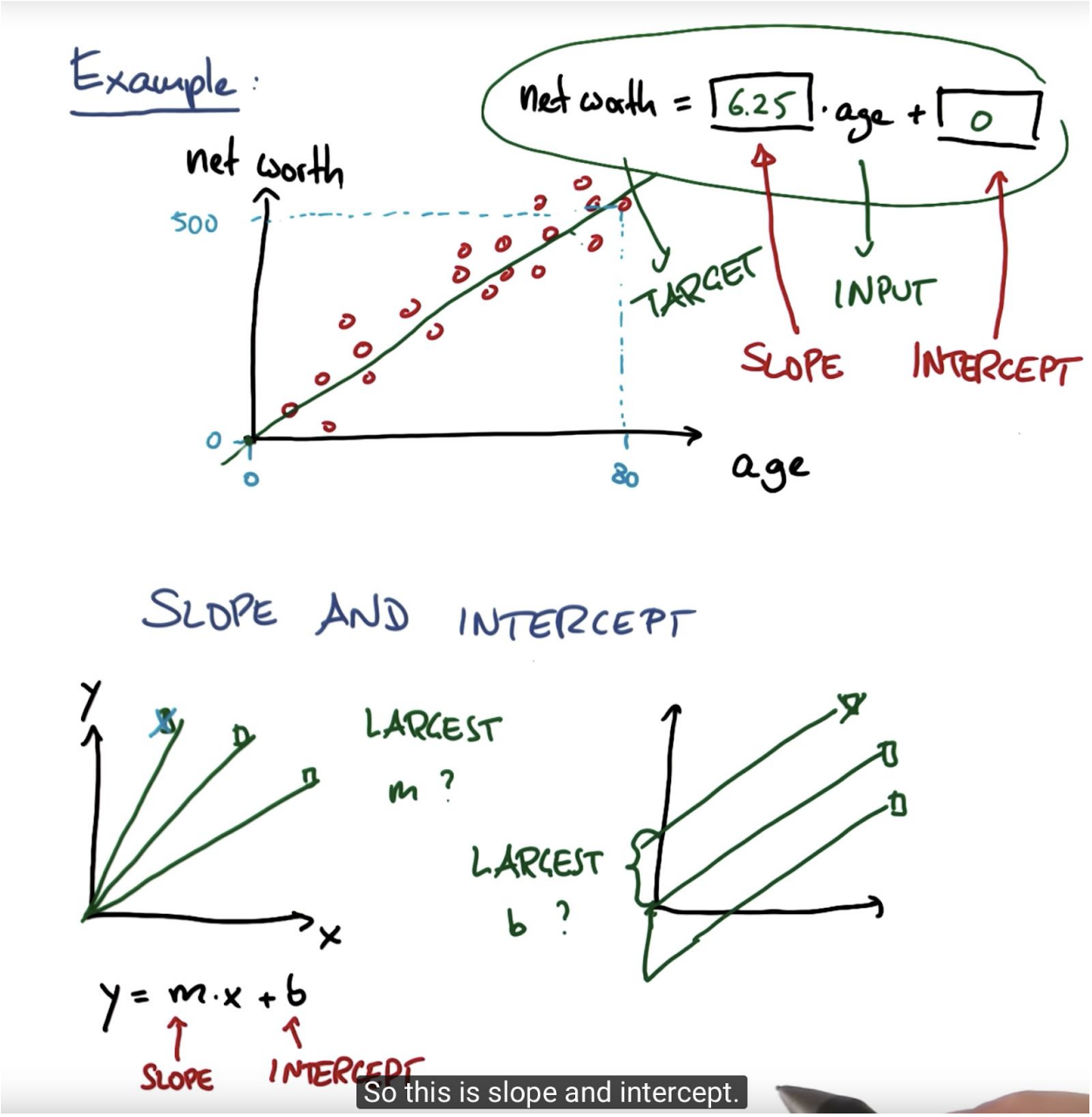
Binary output (e.g. fast and slow): Discrete

Continuous supervised learning does not give discrete (e.g. categorical or binary) but continuous output.

7.8. Continous has some ordering (discrete is more a kind of 'label')

7.11 Structure for result of a supervised learner:

y = factor * x + offset (bijv. net_worth = 6.25 * age + 0 in voorbeeld)



Voor onderste lijn rechts is intercept b negatief.

7.17 Regressie

http://scikit-learn.org/stable/modules/linear_model.html
http://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html

Slope == coefficients

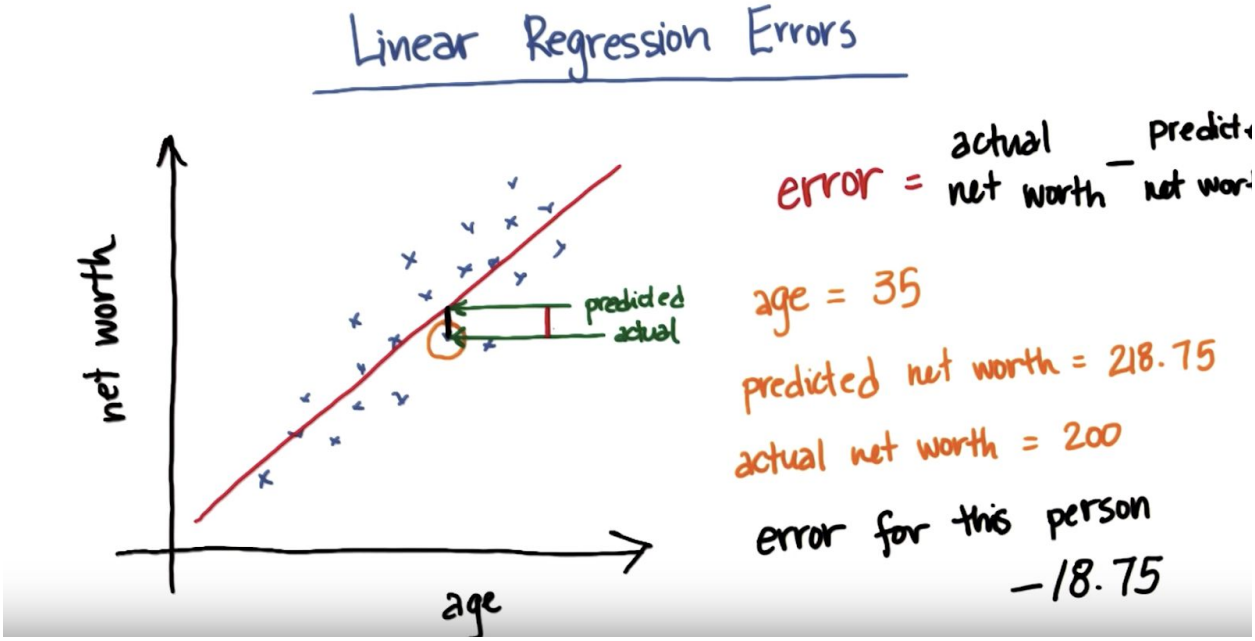
def studentReg(ages_train, net_worths_train):

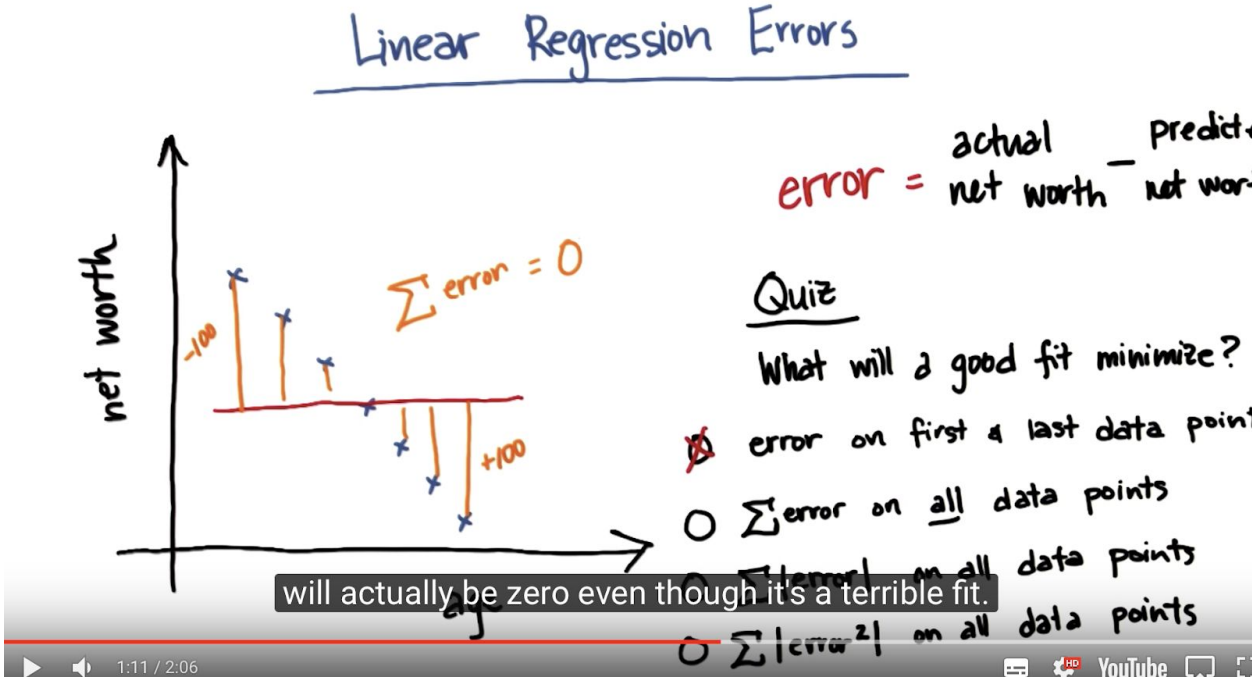
```
### import the sklearn regression module, create, and train your regression
### name your regression reg

### your code goes here!

from sklearn import linear_model
reg = linear_model.LinearRegression()
reg.fit(ages_train, net_worths_train)


return reg
```

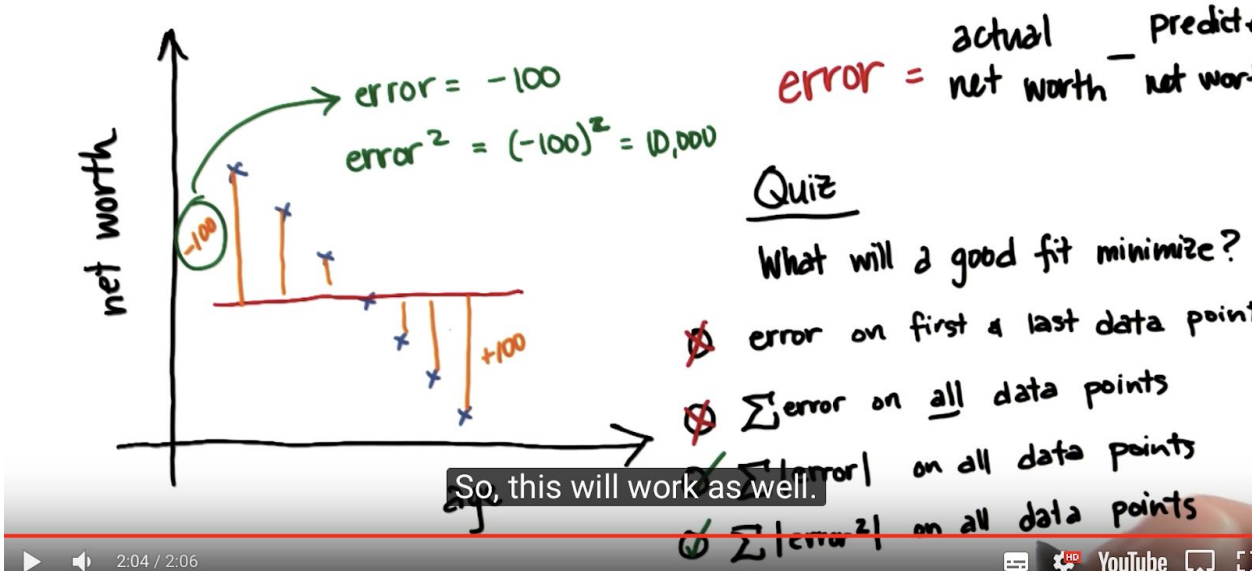
7.20 r-squared (as a performance measure for linear regression), the higher value the better, range 0 .. 1.0

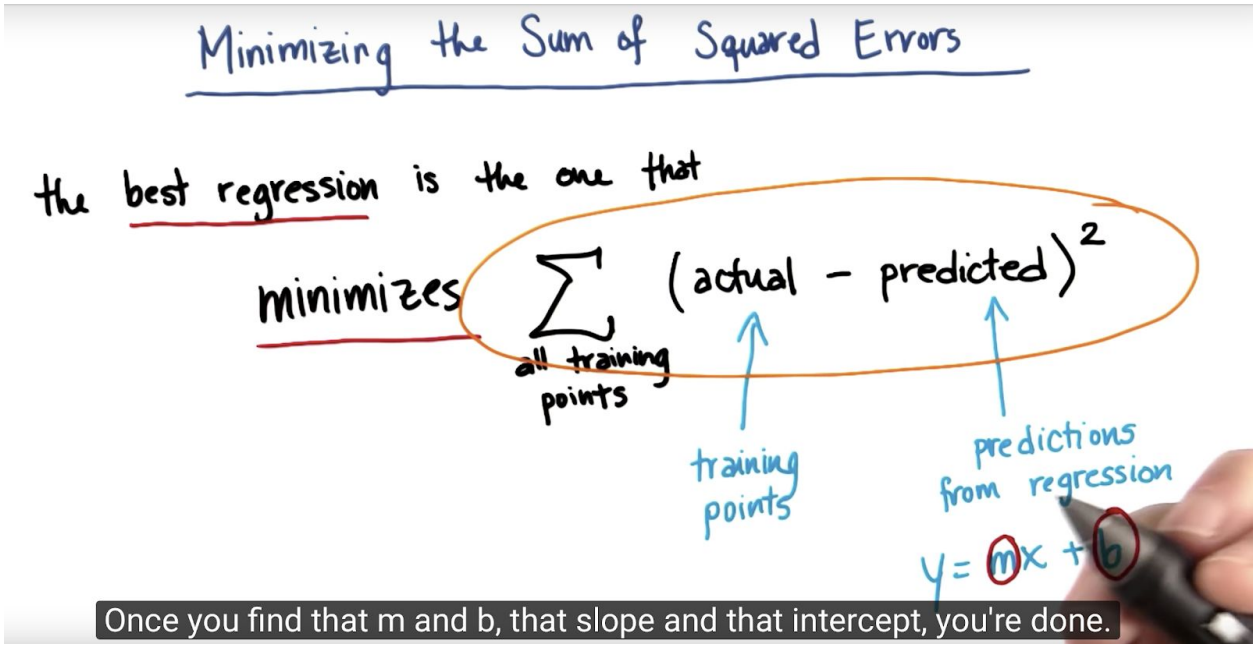
R-squared is used for test dataset but we can compare it with r-squared for the training set

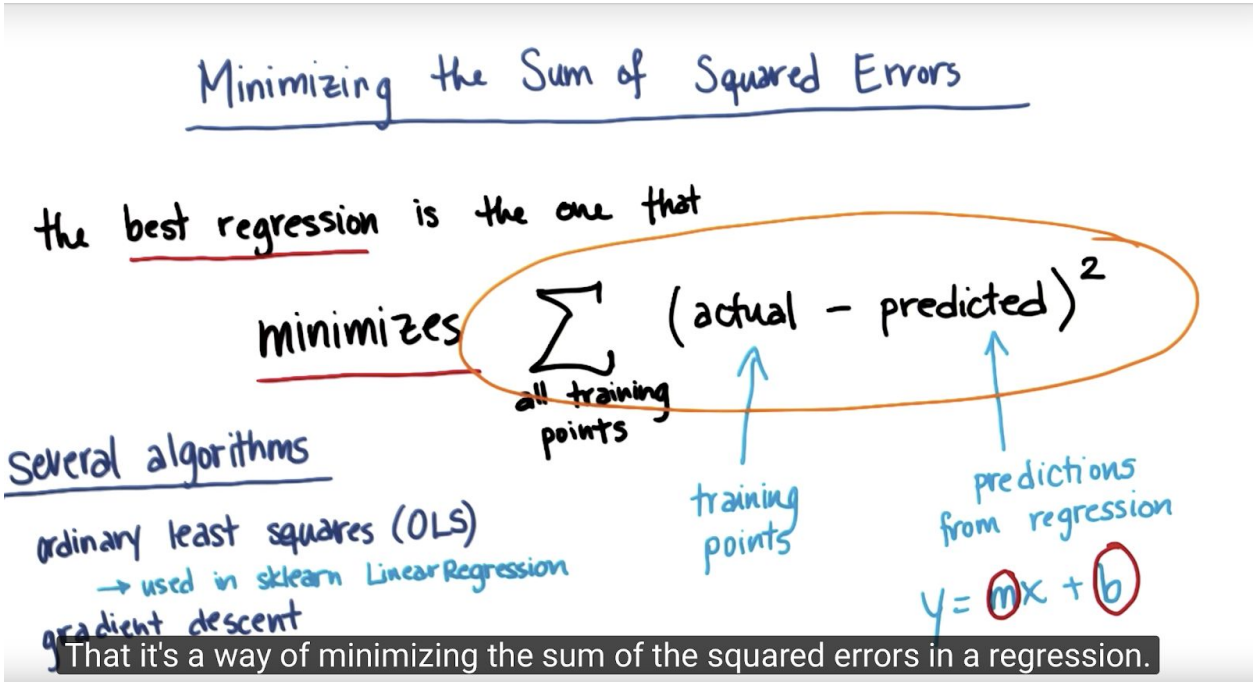
7.21 Error in linear regression: error = actual (e.g. net worth) - predicted (e.g. net worth)

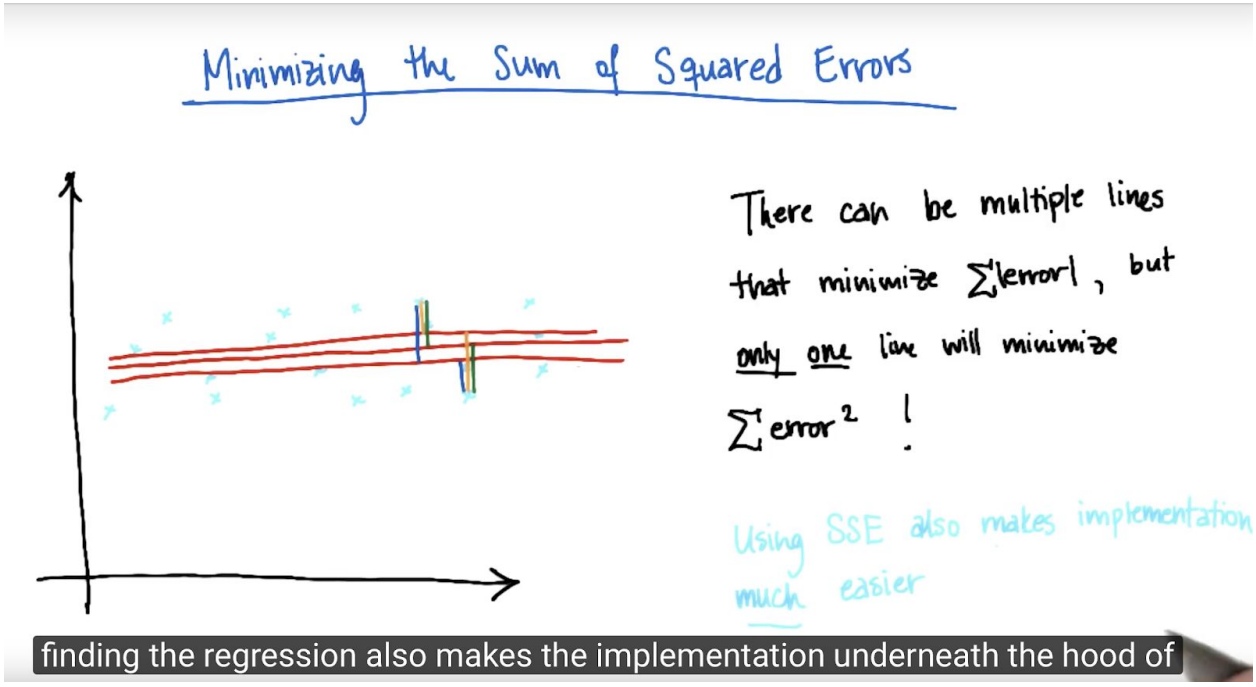



Linear Regression Errors
net worth
age
-100
+100
Σ error = 0
error = actual net worth − predict
Quiz
What will a good fit minimize?
error on first & last data point
Σ error on all data points
Σ |error| on all data points
Σ |error²| on all data points
will actually be zero even though it's a terrible fit.
1:11 / 2:06
YouTube


Linear Regression Errors
net worth
age
-100
+100
error = −100
error² = (−100)² = 10,000
error = actual net worth − predict
Quiz
What will a good fit minimize?
error on first & last data point
Σ error on all data points
Σ |error| on all data points
Σ |error²| on all data points
So, this will work as well.
2:04 / 2:06
YouTube

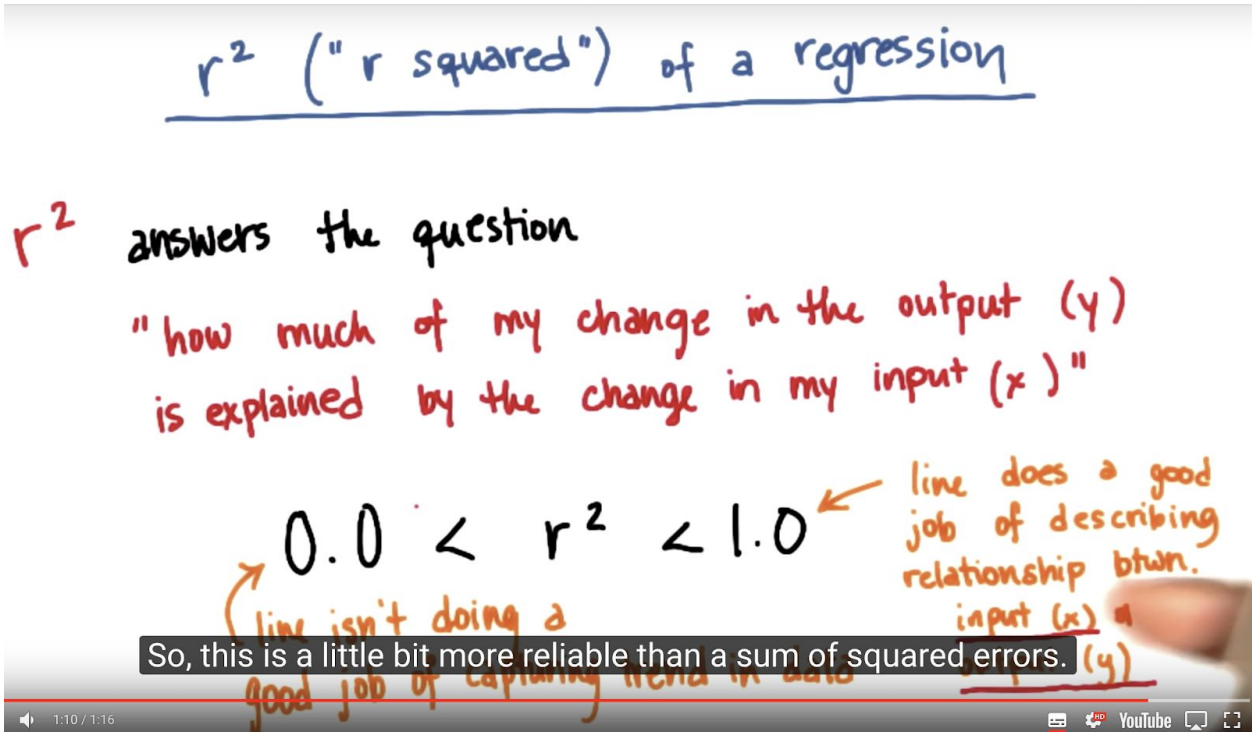

Minimizing the Sum of Squared Errors
the best regression is the one that
minimizes
$\sum_{\text{all training points}} (\text{actual} - \text{predicted})^2$
training points
predictions from regression
$y = mx + b$
Once you find that m and b, that slope and that intercept, you're done.


Minimizing the Sum of Squared Errors
the best regression is the one that
minimizes
$\sum_{\text{all training points}} (\text{actual} - \text{predicted})^2$
training points
predictions from regression
$y = mx + b$
Several algorithms
ordinary least squares (OLS)
→ used in sklearn LinearRegression
gradient descent
That it's a way of minimizing the sum of the squared errors in a regression.

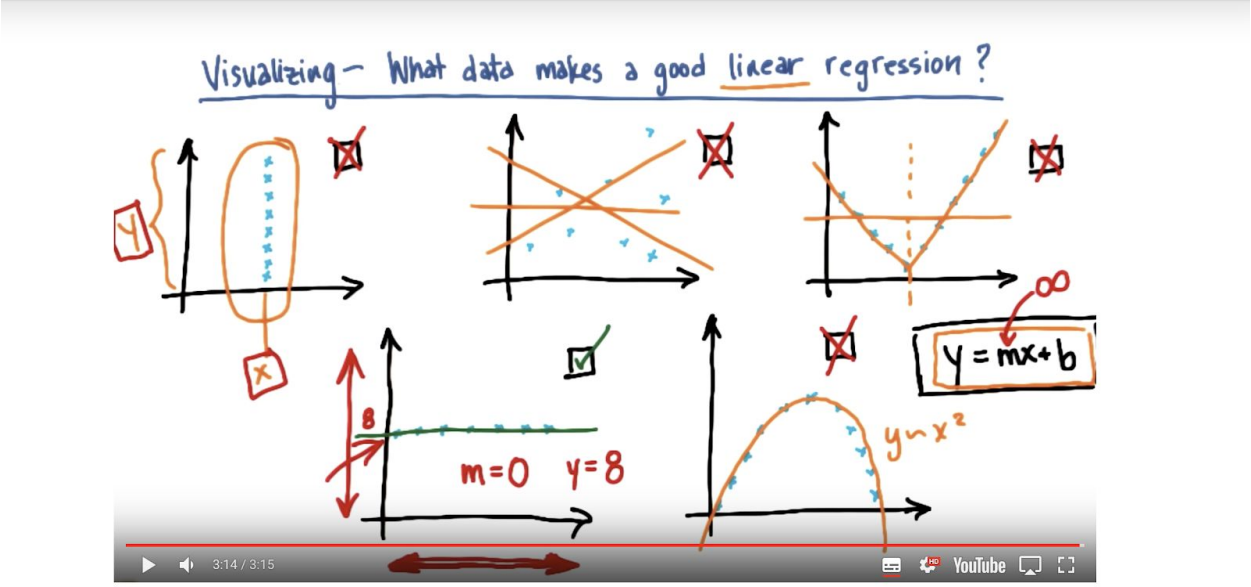


SSE: sum of squared errors.

Bij het hebben van meer data is de SSE ws. hoger daarom zegt het niet altijd wat over de kwaliteit van je regressie: r-squared

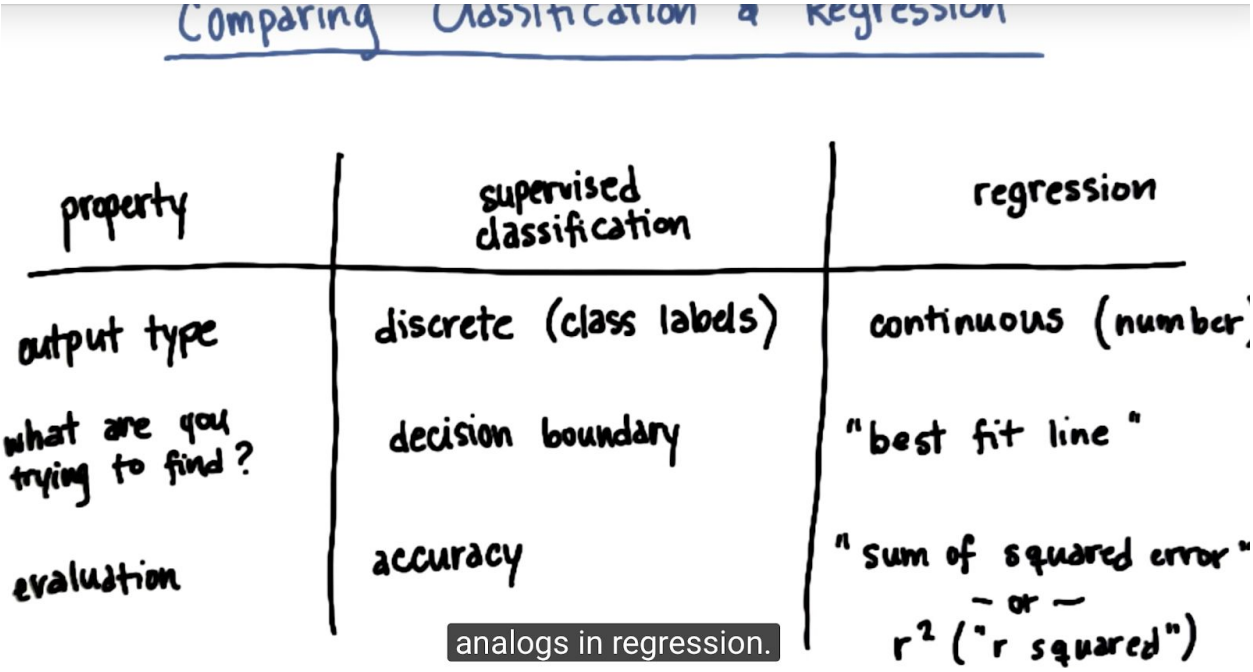


Tip! Eye-balling data using a scatterplot to see if it is a good candidate for regression (in other words y = mx + b)



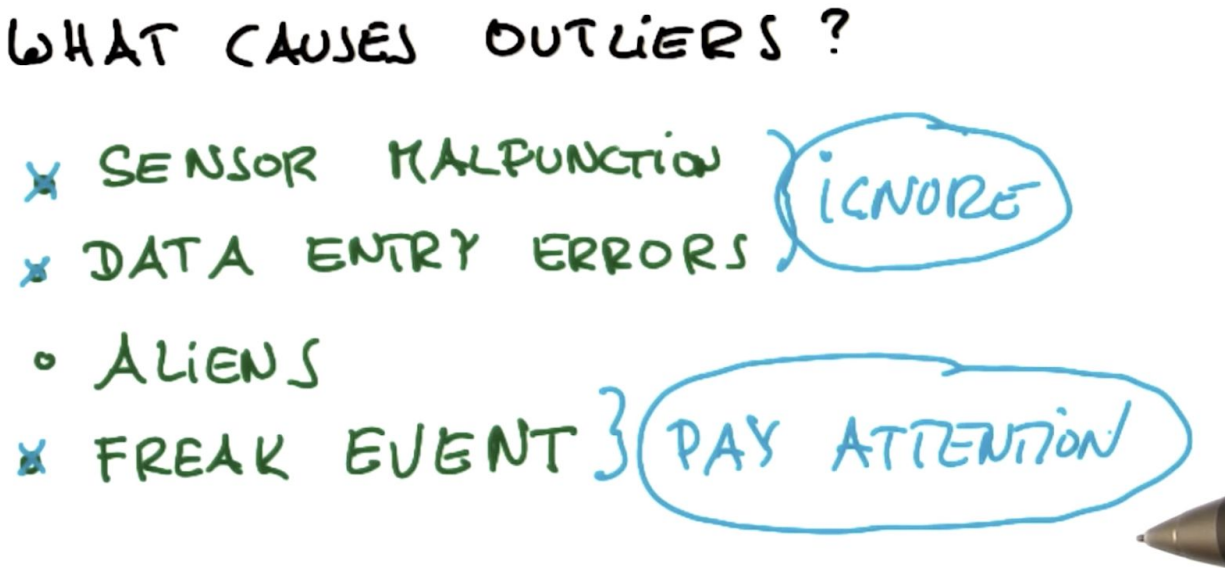
View Intro Back to Quiz

7.35

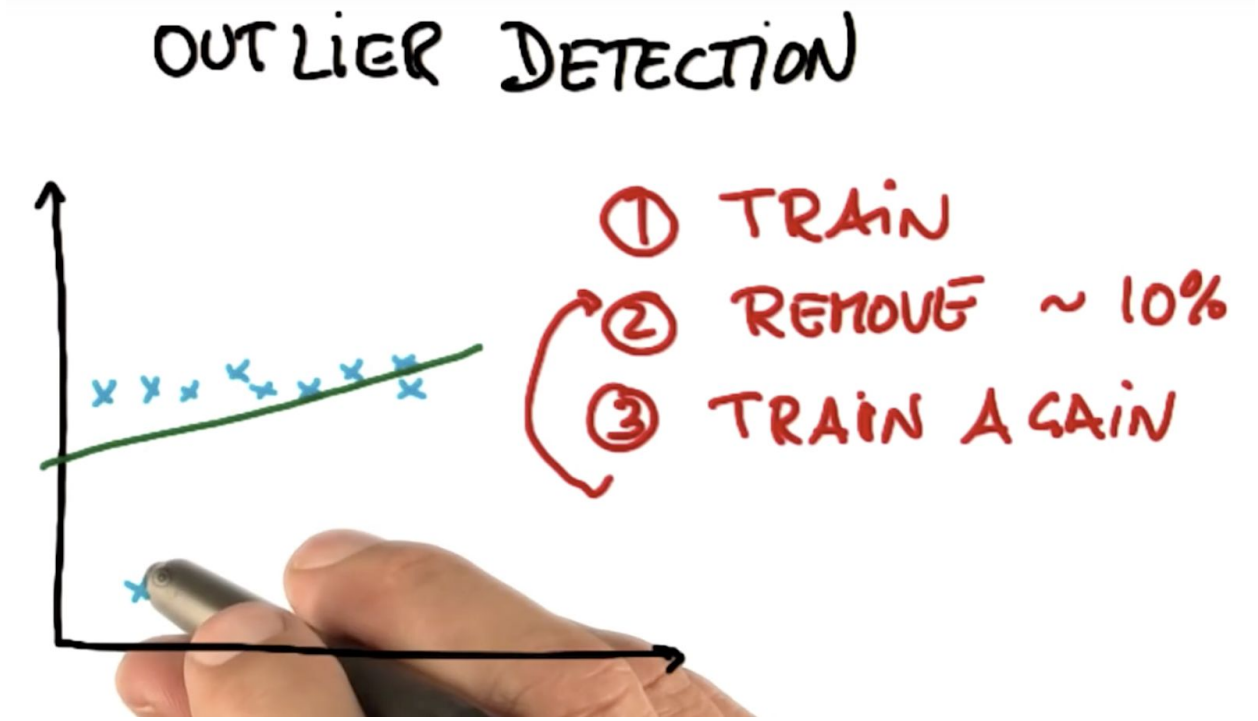


7.36 Multi-variate regression

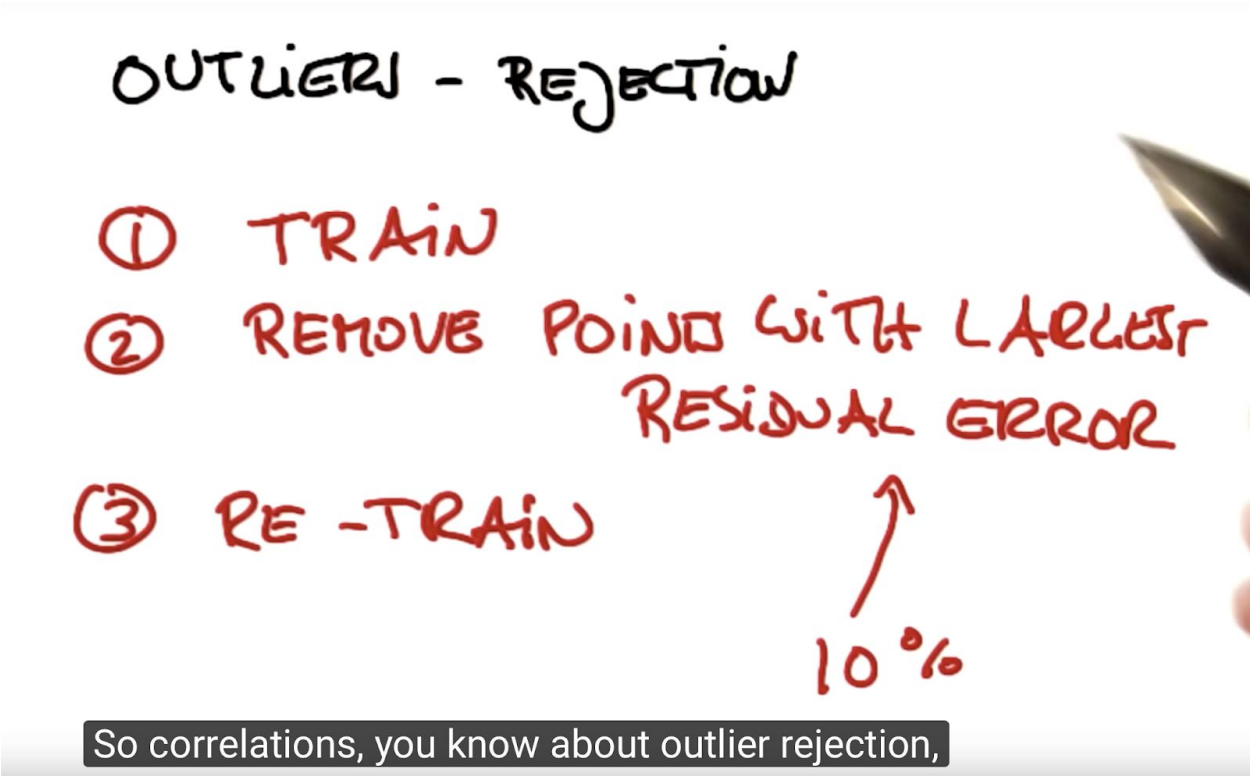
Lesson 8. Outliers



Freak events is bijv. Fraude of zo dus afhankelijk van de situatie wil je outliers negeren of juist oppikken en er wat mee doen!



8. Standaard heuristiek voor outlier detectie.



Die 10% is afhankelijk van casus, dit is de situatie waarbij je dus niet geïnteresseerd bent in outliers, belastingdienst oid is vaak juist geïnteresseerd in outliers en niet in de 'good data points' (=> anomaly detection)

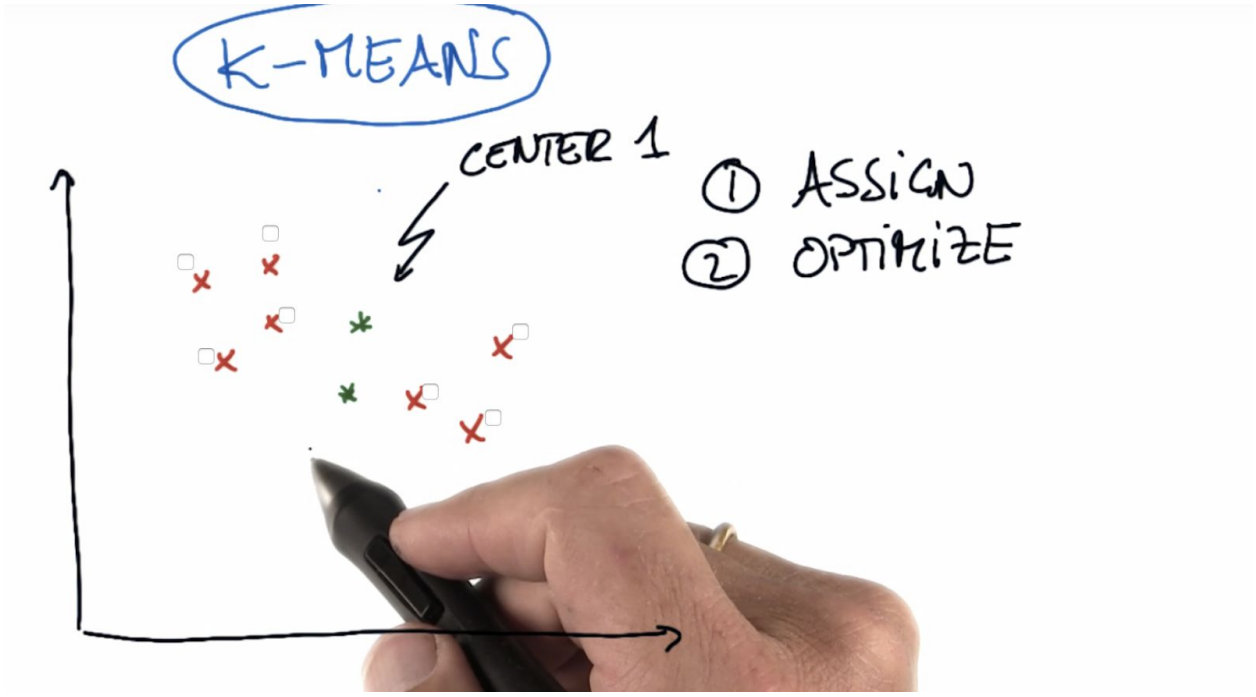
Quiz 13: Score after cleaning.

Coefficient (Slope): [[5.07793064]]
Intercept (Independent factor): [25.21002155]
Mean squared error: 971.79
Variance score: 0.88
Regression score: 0.8782624703664673

After cleaning 10% of outliers:
Coefficient (Slope): [[6.36859481]]
Intercept (Independent factor): [-6.91861069]
Mean squared error: 50483.92
Variance score: -285.42
Regression score: 0.9513734907601892

De slope is wel correct volgens udacity maar de regression score niet helaas, lijkt toch wel (gezien de plot) in de goede richting te zitten.

Lesson 9: Unsupervised Learning



K-Means, assigning clusters (first time at random), then optimizing (and doing this again for the new situation till satisfied (read: till center points do not change location anymore).

https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

Mooi overzicht over clustering algoritmes:
http://scikit-learn.org/stable/modules/clustering.html#k-means

http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

class sklearn.cluster.**KMeans**(*n_clusters=8*, *init='k-means++'*, *n_init=10*, *max_iter=300*, *tol=0.0001*, *precompute_distances='auto'*, *verbose=0*, *random_state=None*, *copy_x=True*, *n_jobs=None*, *algorithm='auto'*)

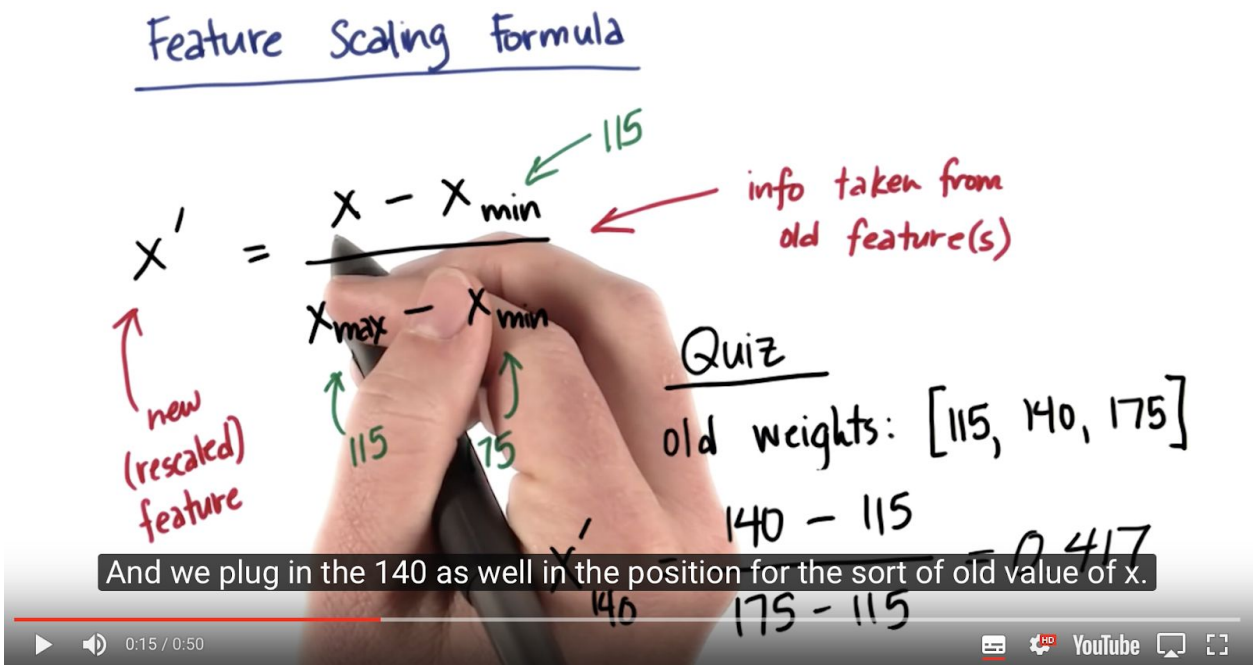
Aantal zaken die je moet definiëren als je begint:

- Aantal clusters (kan een uitdaging zijn) [n_clusters], wel vaak nodig deze aan te passen
- *max_iter* Meestal wel ok, algoritme zal stoppen als er geen betere matches gevonden worden.
- ...

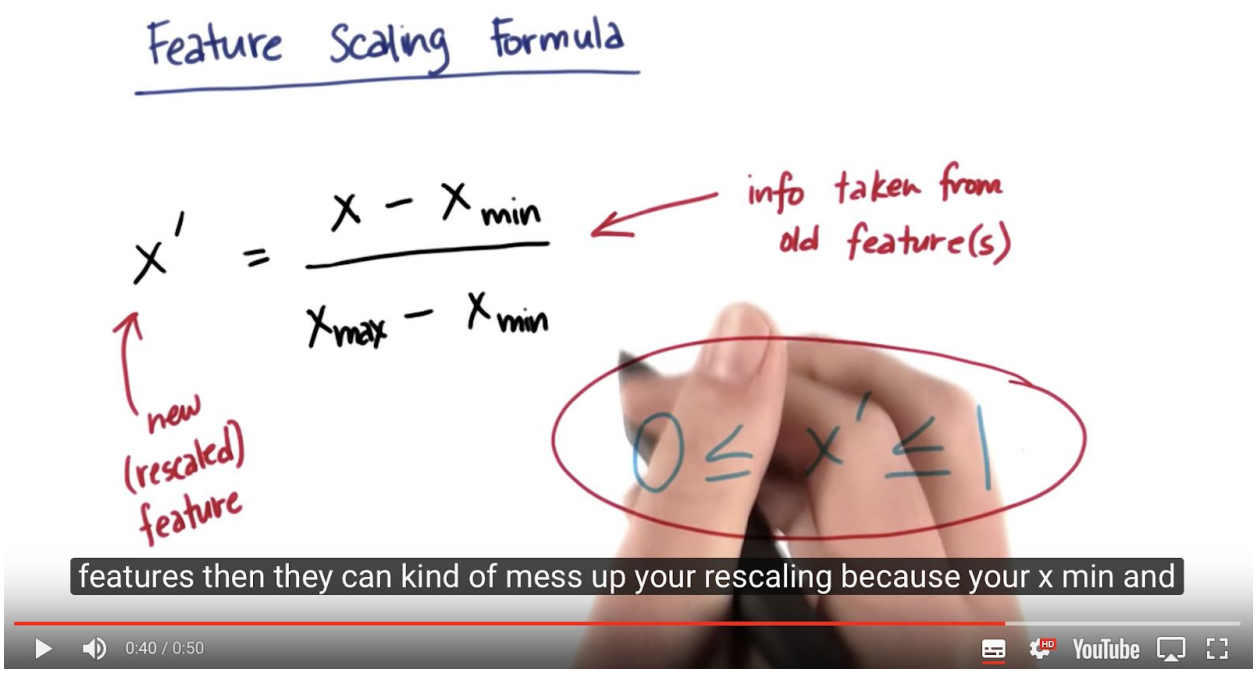
Lesson 10. Feature Scaling

Aantal zaken die je moet definiëren als je begint.

Feature Scaling Formula:



Beware of outliers messing up x(max) en x(min):



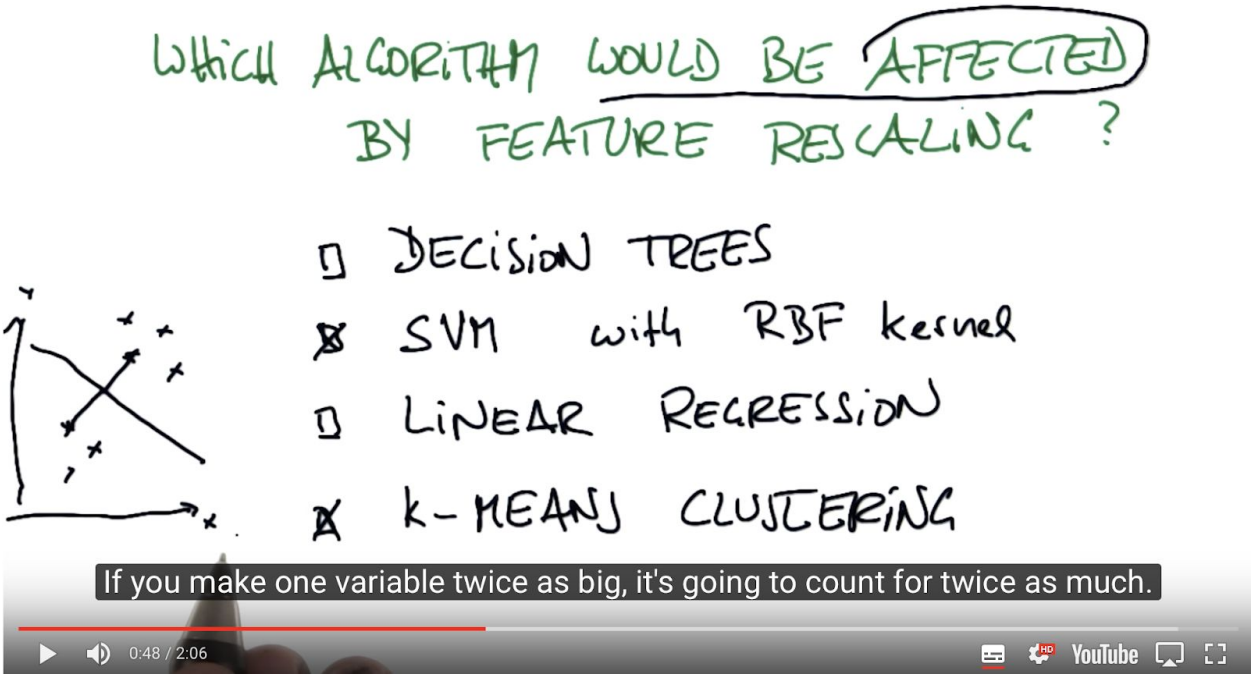
Feature Scaling Formula result is $0 \leq x' \leq 1$

Added feature_scaling folder and also added rescaler_jk.py in ../tools/

http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

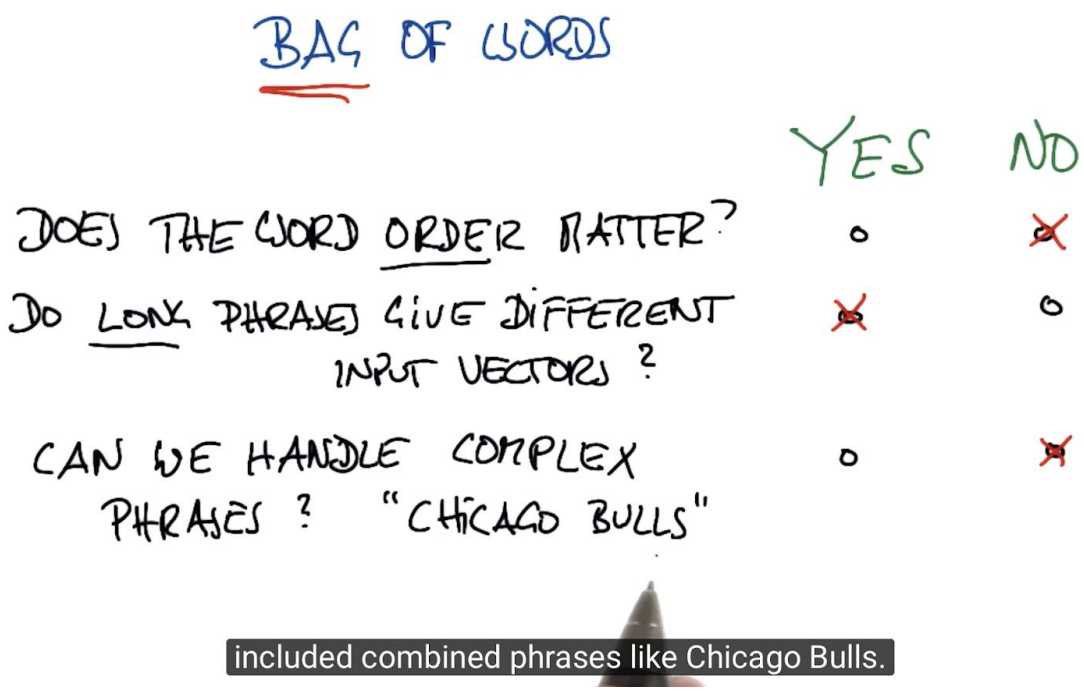
Excercise 10.12:

```
from sklearn.preprocessing import MinMaxScaler
import numpy

weights = numpy.array([[115.], [140.], [175.]])
rescaled_weight = scaler.fit_transform(weights)

>>> rescaled_weight
array([[0.        ],
       [0.41666667],
       [1.        ]])
```



Scaler (10.15): MinMaxScaler

Lesson 11. Text Learning



http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

11.6

```
>>> from sklearn.feature_extraction.text import CountVectorizer
>>> vectorizer = CountVectorizer()
>>> string1 = "hi Katie the self driving car will be late Best Sebastian"
>>> string2 = "hi Sebestian the machine learning class will be great great great Best Katie"
>>> string3 = "hi Katie the machine learning class will be most excellent"
>>> email_list = [string1, string2, string3]
>>> bag_of_words = vectorizer.fit(email_list)
>>> bag_of_words = vectorizer.transform(email_list)
>>> bag_of_words
<3x18 sparse matrix of type '<class 'numpy.int64'>'
        with 32 stored elements in Compressed Sparse Row format>
>>> print(bag_of_words)
  (0, 0)    1
  (0, 1)    1
  (0, 2)    1
  (0, 4)    1
  (0, 7)    1
  (0, 8)    1
  (0, 9)    1
  (0, 13)   1
  (0, 15)   1
  (0, 16)   1
  (0, 17)   1
  (1, 0)    1
  (1, 1)    1
  (1, 3)    1
  (1, 6)    3
```

```
  (1, 7)    1
  (1, 8)    1
  (1, 10)   1
  (1, 11)   1
  (1, 14)   1
  (1, 16)   1
  (1, 17)   1
  (2, 0)    1
  (2, 3)    1
  (2, 5)    1
  (2, 7)    1
  (2, 8)    1
  (2, 10)   1
  (2, 11)   1
  (2, 12)   1
  (2, 16)   1
  (2, 17)   1
>>>
```

Uitleg: (document, word) wordcount

```
>>> print(vectorizer.vocabulary_.get("great"))
6
```

Beetje verwarrende output (komt wel overeen met die van Katie)

Stop word : low information words (lidwoorden, etc.)

11.9 >>> from nltk.corpus import stopwords

```
>>> import nltk
>>> nltk.download('stopwords')
```

Tip van Katie:
nltk.download() en corpora selecteren
nltk.download('all', halt_on_error=False)

len(sw)

11.10 Stemmer brings words down to it's root form

E.g.
Unresponsive
Response
Responsiveness -> respon
Responsivity
Respond

Effect: five dimensions are brought down to one dimension (they ignored semantics though), unresponsive has the opposite semantics of responsive of course!

11.11 Stemmers
from nltk.stem.snowball import SnowballStemmer

stemmer.stem("computerization")

stemmer.stem("unresponsive") -> unrespons (so did not lose the semantics)

11.12 Order

- A. Stemming
- B. Creating bag-of-words

11.13 Weight of terms by frequency

Tf - term frequency (somewhat comparable to bag-of-words)
Idf - inverse document frequency (weight of words from the entire corpus (the documents together as a whole))
Also: rare words get a higher score.

TfIdf: gives a higher score to rare words as they might be most useful in distinguishing different messages

Idf - weighing the words by the inverse frequency of how often they appear in the corpus / messages (more rare: higher score)

11.15 / 11.16 classifying emails

You will be given two text files: one contains the locations of all the emails from Sara, the other has emails from Chris. You will also have access to the *parseOutText()*function, which accepts an opened email as an argument and returns a string containing all the (stemmed) words in the email.

11.17 Dit is volgens mij de correcte naar Python 3 omgebouwde code, antwoord wordt niet geaccepteerd.

```
#!/usr/bin/python

from nltk.stem.snowball import SnowballStemmer
import string

def parseOutText(f):
    """ given an opened email file f, parse out all text below the
        metadata block at the top
        (in Part 2, you will also add stemming capabilities)
        and return a string that contains all the words
        in the email (space-separated)

        example use case:
```

```
        f = open("email_file_name.txt", "r")
        text = parseOutText(f)

        """


    f.seek(0)  ### go back to beginning of file (annoying)
    all_text = f.read()

    ### split off metadata
    content = all_text.split("X-FileName:")
    words = ""
    if len(content) > 1:
        ### remove punctuation
        translator = str.maketrans('', '', string.punctuation)
        text_string = content[1].translate(translator)

        ### project part 2: comment out the line below
        ### words = text_string

        ### split the text string into individual words, stem each word,
        ### and append the stemmed word to words (make sure there's a single
        ### space between each stemmed word)
        from nltk.stem.snowball import SnowballStemmer
        stemmer = SnowballStemmer("english")
        words = text_string.split()
        for word in words:
            print (stemmer.stem(word))
        words = ' '.join(word for word in words)
    return words

def main():
    ff = open("../text_learning/test_email.txt", "r")
    text = parseOutText(ff)
    print (text)

if __name__ == '__main__':
    main()
```

Er is iets aparts aan de hand, output zit er super dicht bij maar ietsjes anders:
“tjonesnsf stephanie and sam need nymex calendars” udacity verwacht “tjonesnsf stephani and sam need nymex calendar”

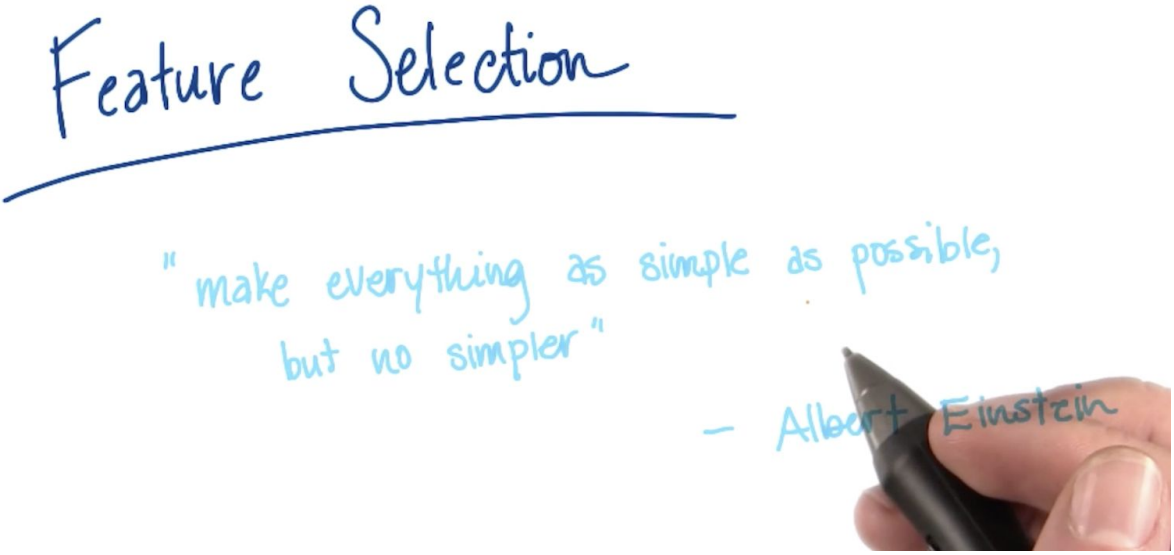
11.21

```
### in Part 4, do TfIdf vectorization here
# see
http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()
output = vectorizer.fit_transform(word_data)
features_in_corpus = vectorizer.get_feature_names()

print("There are " + str(len(features_in_corpus)) + " features:")
print(features_in_corpus)

print("Result for 34597: " + str(features_in_corpus[34597]))
```

Je krijgt daar overigens niet het correcte antwoord mee, maar dat is ws. gevolg van iets afwijkende corpus, zie ook:

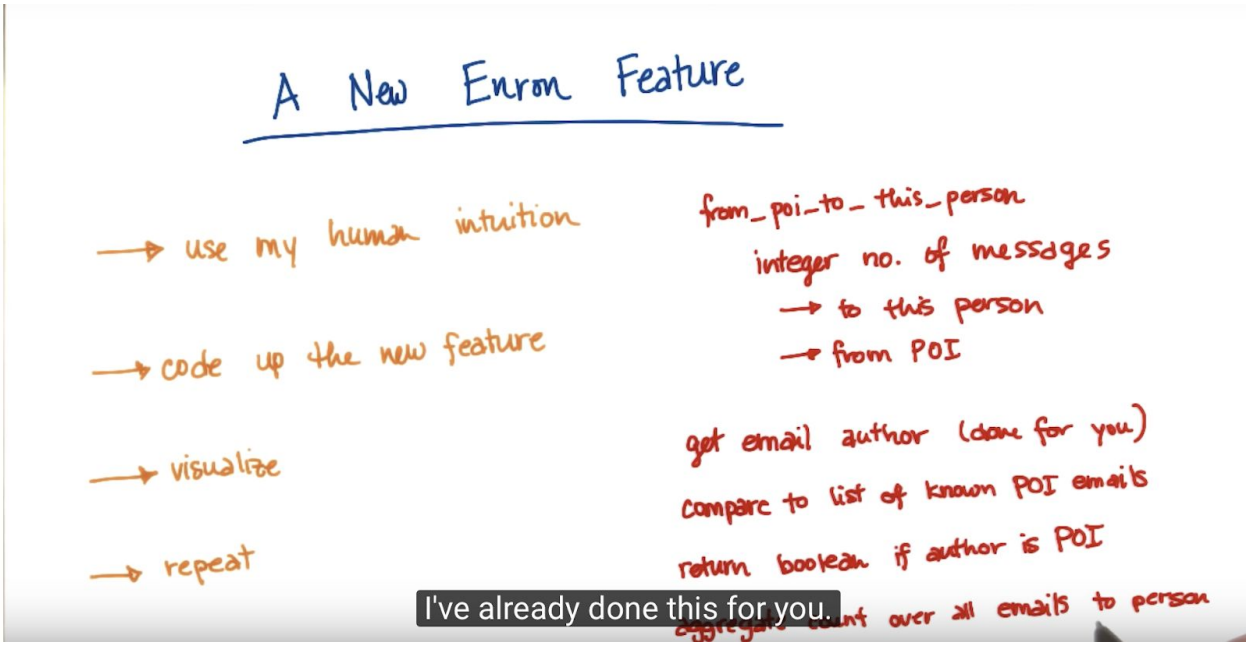
https://discussions.udacity.com/t/broken-quizzes-tfidf-it-accessing-tfidf-features/209861/12

Lesson 12: Feature selection



New feature:

- Stap 1: Intuition (boerenverstand)
- Stap 2: coderen
- Stap 3: visueel inspecteren (geeft feature voldoende onderscheidend vermogen (discriminating power)
- Stap 4: iteratief / opnieuw doorlopen tot je zo optimaal mogelijk resultaat krijgt.



12.3 a new Enron Quiz Feature

Zie voorbeeld code, alleen kan from maar van één afzender zijn dus alleen die checken tegen poi_email_list

```
  if from_emails:
      if from_emails[0] in poi_email_list:
          from_poi = True
```

12.4

```
import pickle
from get_data import getData

def computeFraction( poi_messages, all_messages ):
    """ given a number messages to/from POI (numerator)
        and number of all messages to/from a person (denominator),
        return the fraction of messages to/from that person
        that are from/to a POI
    """


    ### you fill in this code, so that it returns either
    ###     the fraction of all messages to this person that come from POIs
    ###     or
    ###     the fraction of all messages from this person that are sent to POIs
    ### the same code can be used to compute either quantity

    ### beware of "NaN" when there is no known email address (and so
    ### no filled email features), and integer division!
    ### in case of poi_messages or all_messages having "NaN" value, return 0.
    fraction = 0.

    if (poi_messages == 'NaN' or all_messages == 'NaN'):
        return 0
    if (poi_messages > 0):
        fraction = poi_messages / float(all_messages)


    return fraction


data_dict = getData()

submit_dict = {}
for name in data_dict:

    data_point = data_dict[name]

    print
    from_poi_to_this_person = data_point["from_poi_to_this_person"]
    #print(from_poi_to_this_person)
```

```
    to_messages = data_point["to_messages"]
    #print(to_messages)
    fraction_from_poi = computeFraction( from_poi_to_this_person, to_messages )
    print fraction_from_poi
    data_point["fraction_from_poi"] = fraction_from_poi


    from_this_person_to_poi = data_point["from_this_person_to_poi"]
    from_messages = data_point["from_messages"]
    fraction_to_poi = computeFraction( from_this_person_to_poi, from_messages )
    print fraction_to_poi
    submit_dict[name]={"from_poi_to_this_person":fraction_from_poi,
                       "from_this_person_to_poi":fraction_to_poi}
    data_point["fraction_to_poi"] = fraction_to_poi


######################

def submitDict():
    return submit_dict
```

12.5 Feature Bugs

When Katie was working on the Enron POI identifier, she engineered a feature that identified when a given person was on the same email as a POI. So for example, if Ken Lay and Katie Malone are both recipients of the same email message, then Katie Malone should have her "shared receipt" feature incremented. If she shares lots of emails with POIs, maybe she's a POI herself.

Here's the problem: there was a subtle bug, that Ken Lay's "shared receipt" counter would also be incremented when this happens. And of course, then Ken Lay always shares receipt with a POI, because *he is* a POI. So the "shared receipt" feature became extremely powerful in finding POIs, because it effectively was encoding the label for each person as a feature.

We found this first by being suspicious of a classifier that was always returning 100% accuracy. Then we removed features one at a time, and found that this feature

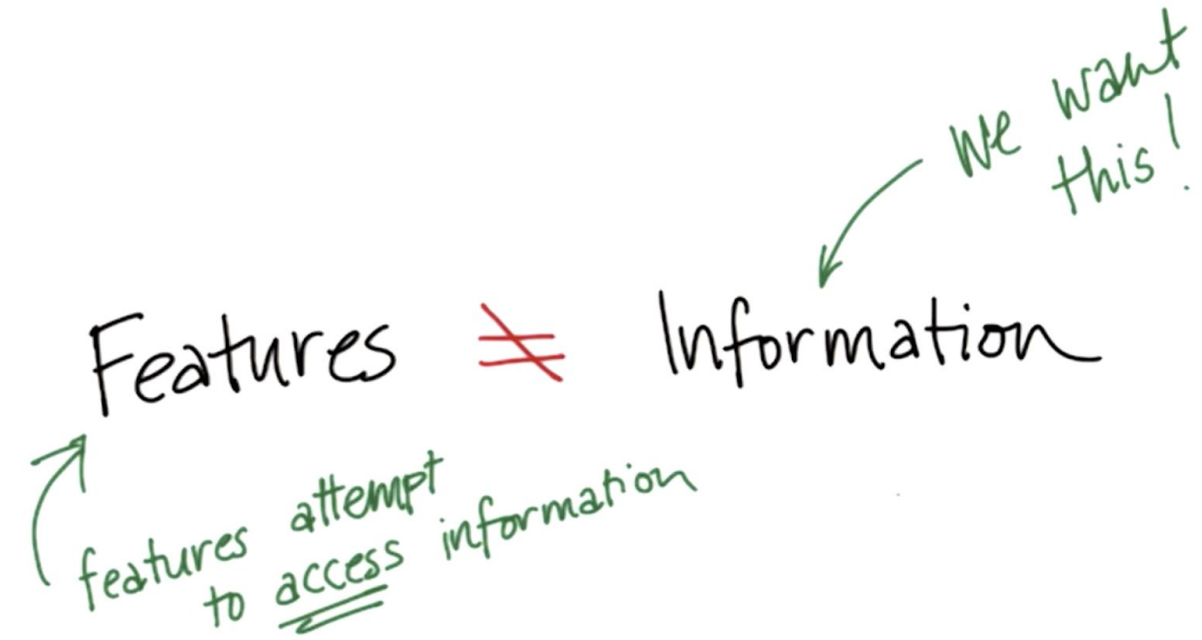
was driving all the performance. Then, digging back through the feature code, we found the bug outlined above. We changed the code so that a person's "shared receipt" feature was only incremented if there was a *different* POI who received the email, reran the code, and tried again. The accuracy dropped to a more reasonable level.

We take a couple of lessons from this:

- Anyone can make mistakes--be skeptical of your results!
- 100% accuracy should generally make you suspicious. Extraordinary claims require extraordinary proof.
- If there's a feature that tracks your labels a little too closely, it's very likely a bug!
- If you're sure it's not a bug, you probably don't need machine learning--you can just use that feature alone to assign labels.

Reasons to ignore certain features:

12.8/9 Information vs features



Tip! Get rid of features that do not give (new) information

There are several go-to methods of **automatically selecting your features in sklearn**. Many of them fall under the umbrella of **univariate feature selection**, which treats each feature independently and asks how much power it gives you in classifying or regressing.

There are two big univariate feature selection tools in sklearn: **SelectPercentile** and **SelectKBest**. The difference is pretty apparent by the names: SelectPercentile selects the X% of features that are most powerful (where X is a parameter) and SelectKBest selects the K features that are most powerful (where K is a parameter).

A clear candidate for feature reduction is text learning, since the data has such high dimension. We actually did feature selection in the Sara/Chris email classification problem during the first few mini-projects; you can see it in the code in tools/email_preprocess.py .

12.10

sklearn.feature_extraction.text.TfidfVectorizer ¶

max_df : float in range [0.0, 1.0] or int, default=1.0

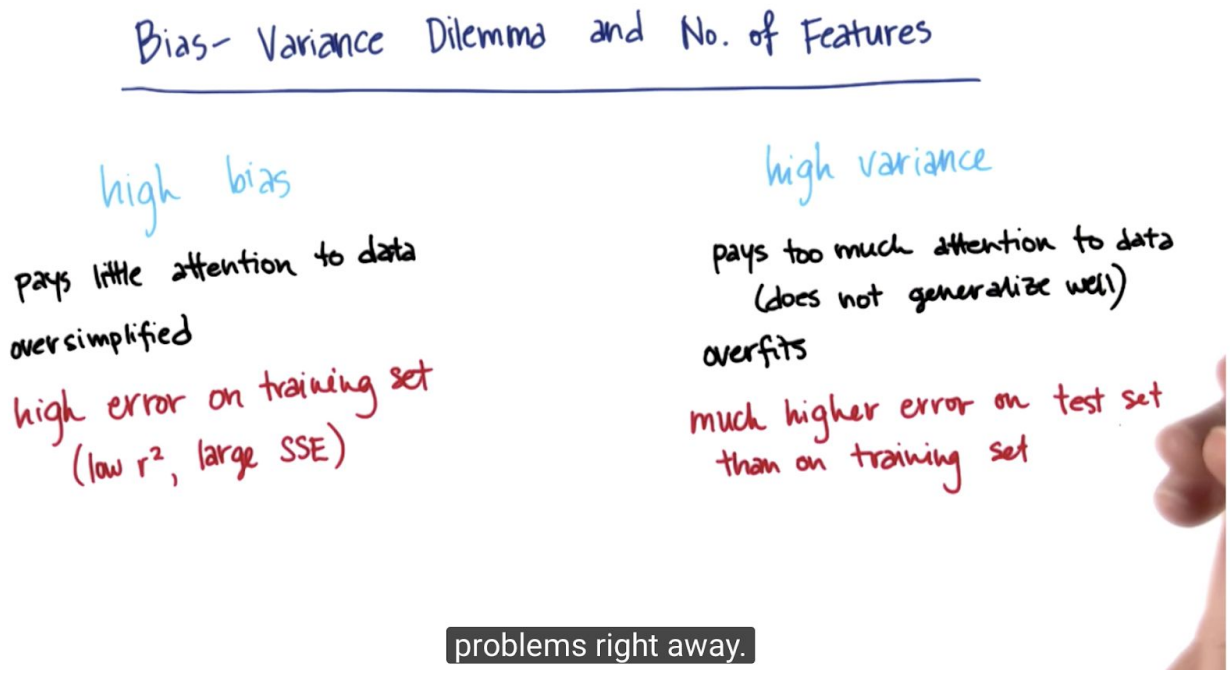
When building the vocabulary ignore terms that have a document frequency strictly higher than the given threshold (corpus-specific stop words). If float, the parameter represents a proportion of documents, integer absolute counts. This parameter is ignored if vocabulary is not None.

min_df : float in range [0.0, 1.0] or int, default=1

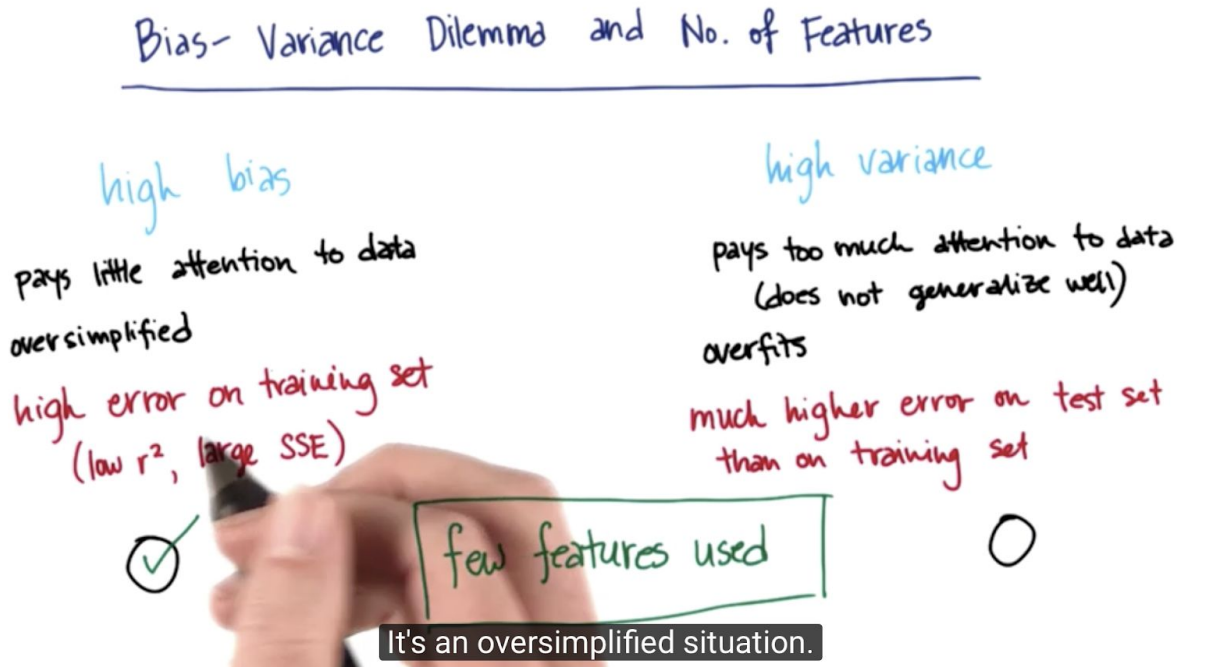
When building the vocabulary ignore terms that have a document frequency strictly lower than the given threshold. This value is also called cut-off in the literature. If float, the parameter represents a proportion of documents, integer absolute counts. This parameter is ignored if vocabulary is not None.

If max_df is e.g. 0.5: words with a document frequency of more than 0.5 will be removed. The answer lies in the "max_df=0.5" parameter.

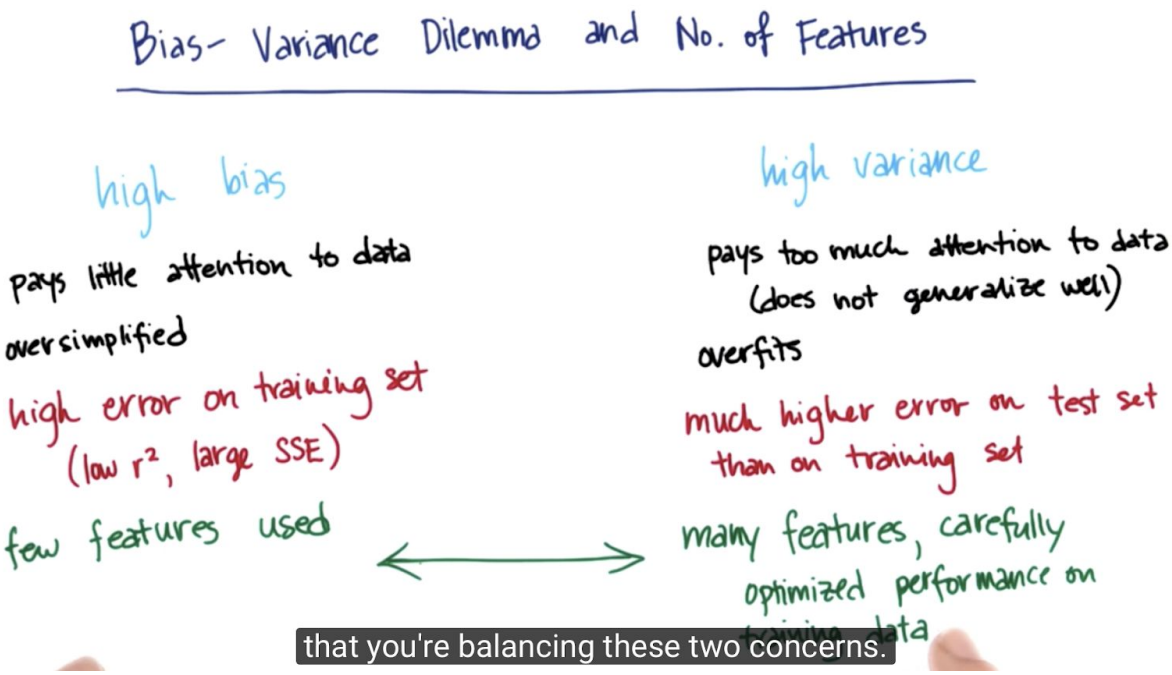
12.11 Bias and Variance Dilemma



12.12 High bias can occur if there are only a few features (small amount of data)



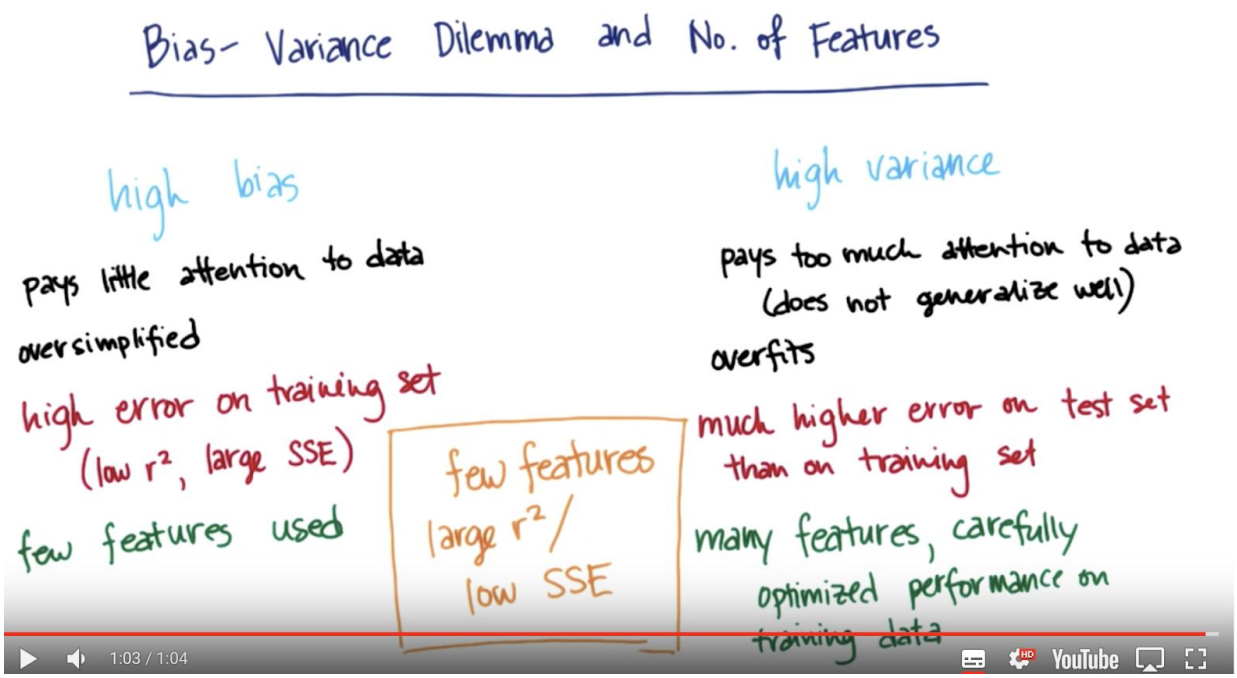
Als je enorm focust op hoge r2 / kleine SSE en linker kant dus optimaliseert loop je meer risico op rechter kant:



M.a.w. Er is dus een ‘trade-off’ tussen bias en variance.

Samengevat:
Je wilt graag:

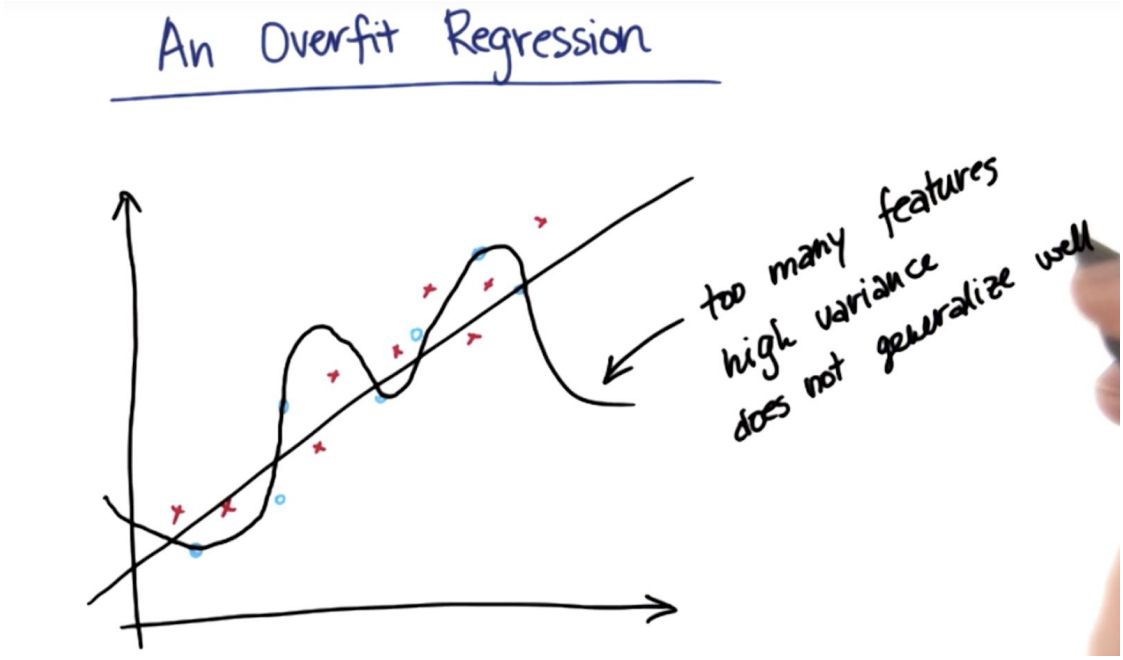
- Weinig features
- Hoge r^2
- Lage sum of squared errors (SSE)

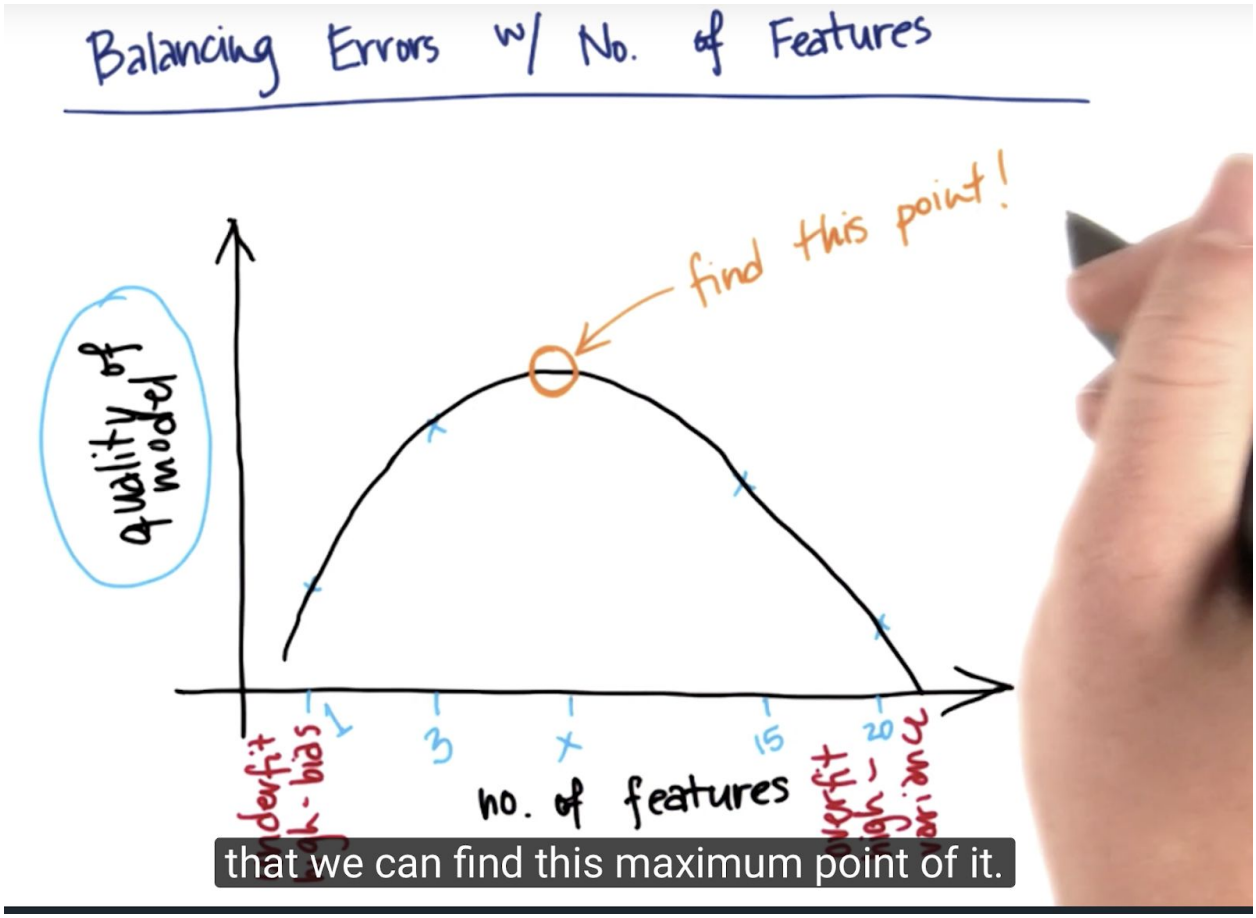


12.13 Overfitting by eye

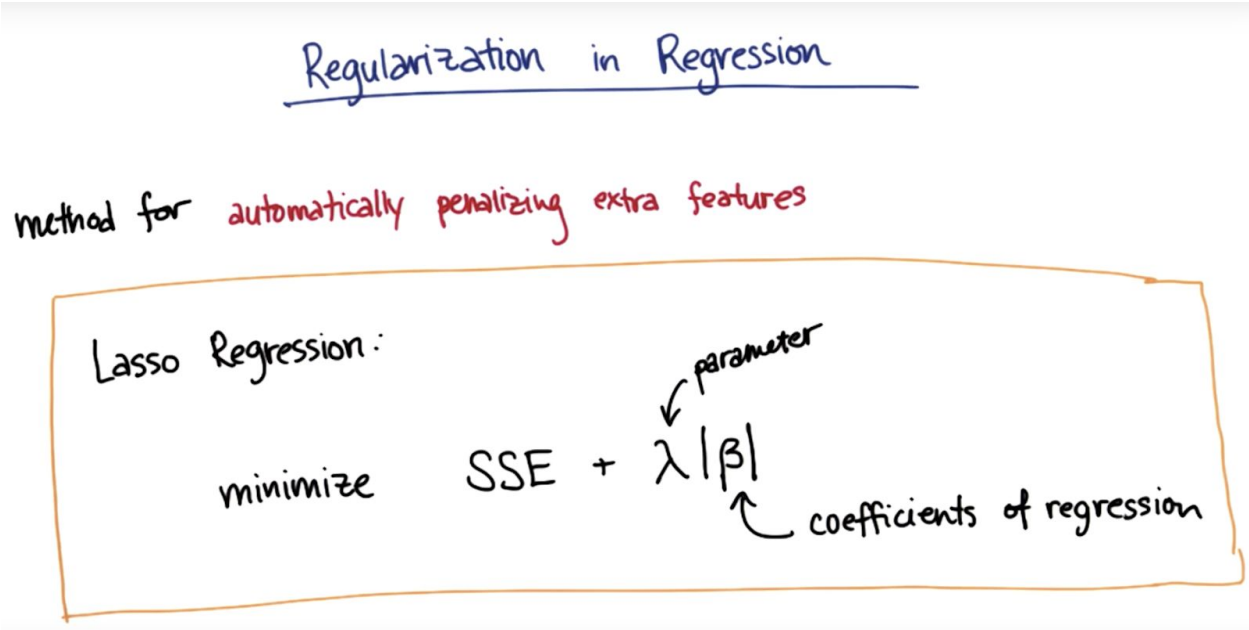
Zo'n plaatje laat zien wat probleem is van overfitting. Als computer bijv. Probeert die SSE enorm naar beneden te brengen zou die 'meanderende' lijn een mooie lage SSE op kunnen leveren, maar extra punten toevoegen. Oplossing is erg 'high-variance' en dus weinig generaliseerbaar terwijl de rechte lijn of evt licht gebogen lijn ook een

acceptabele SSE geeft welke ook de rode punten zou accepteren.

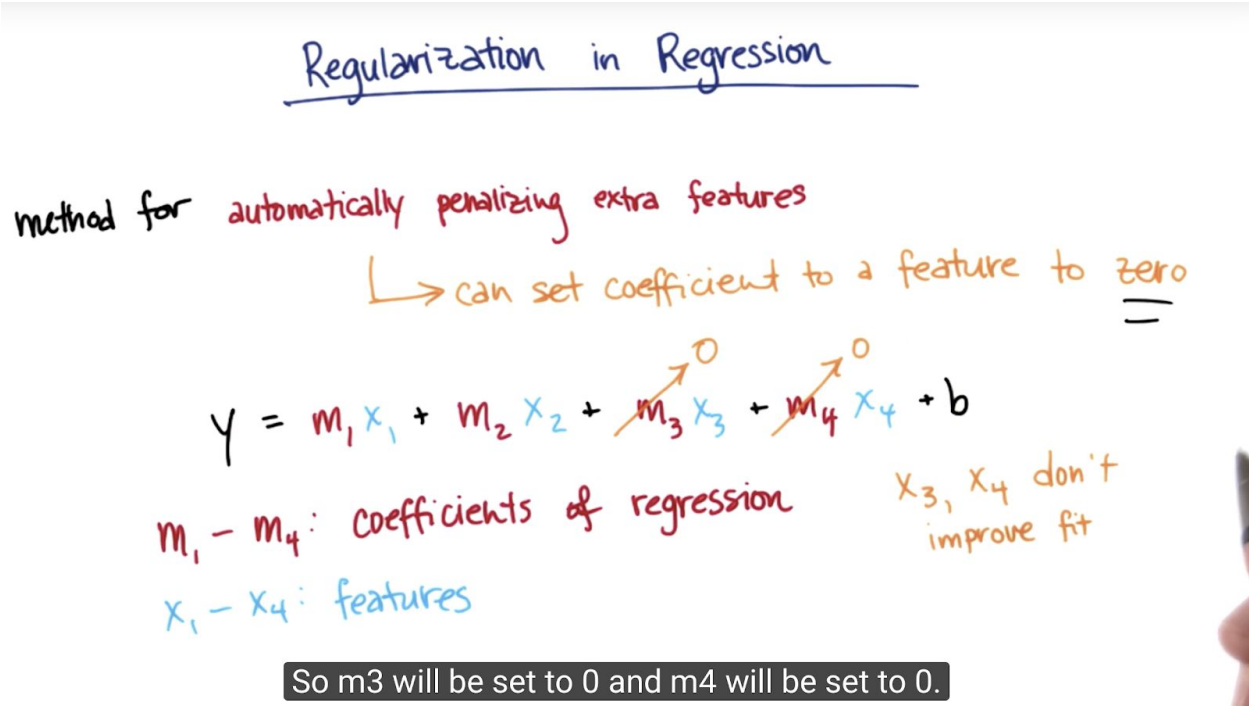




11.15 Regularization: Automating penalization of extra features.



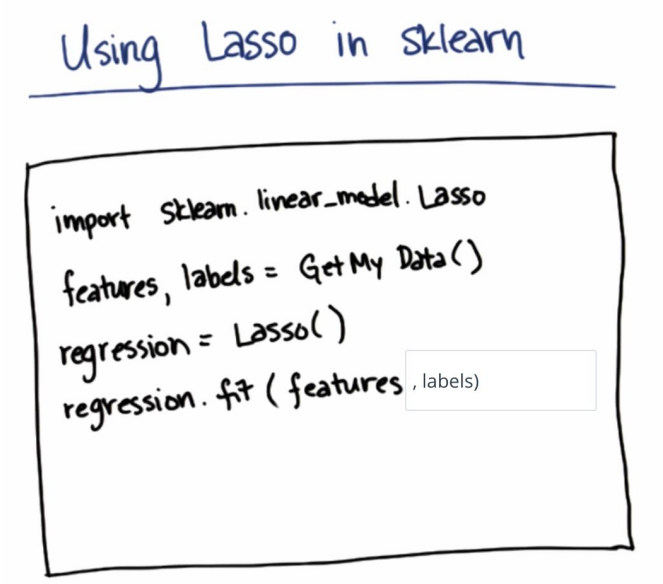
Lasso regression: balancing minimizing SSE + minimizing amount of features



So m3 will be set to 0 and m4 will be set to 0.

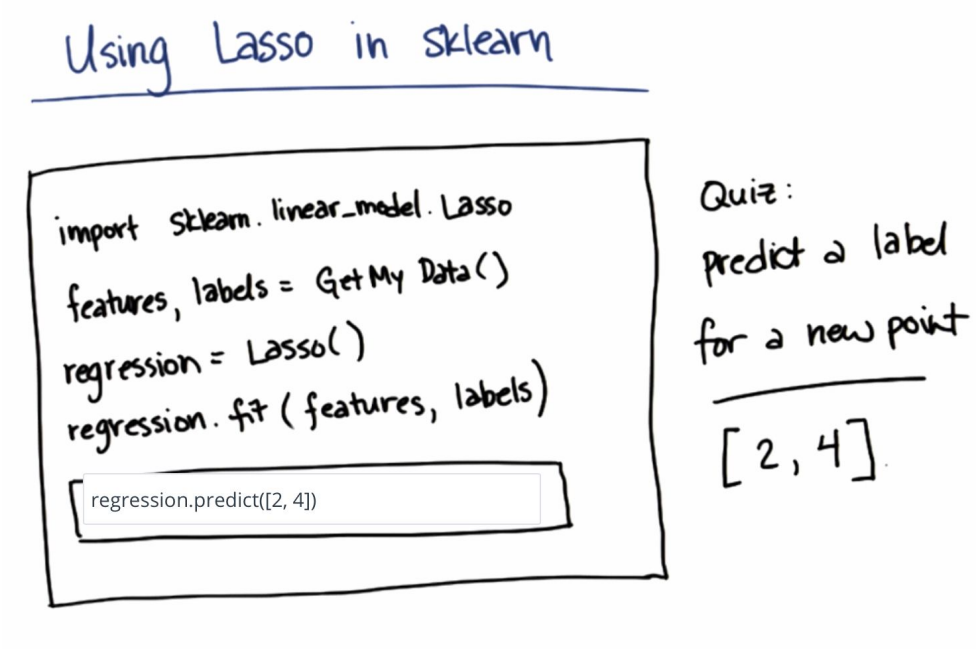
12.17 Lasso Regression

http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html



Regression: supervised algoritme daarom features én labels nodig!

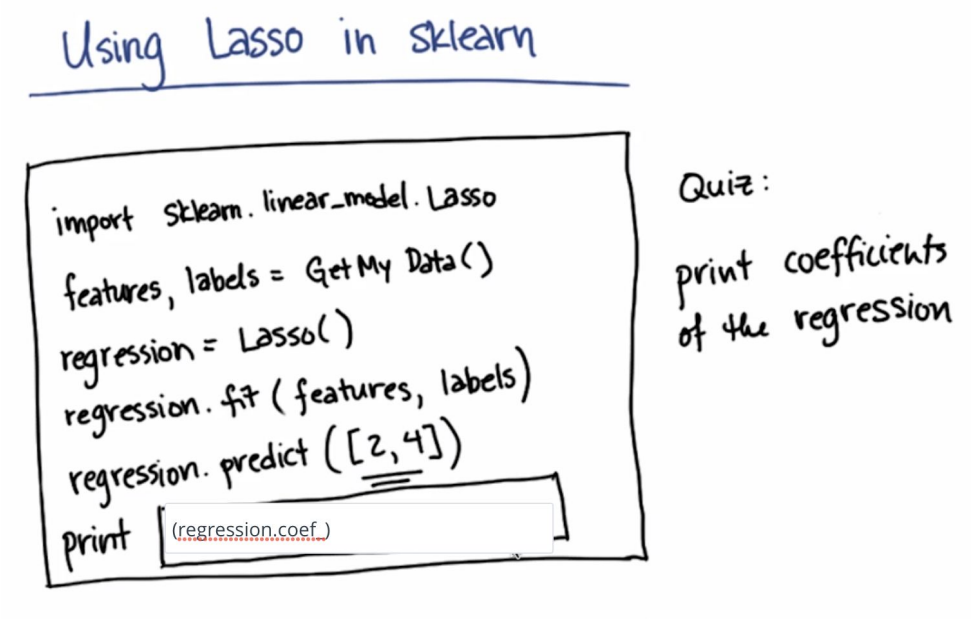
12.18 Lasso Prediction with SKLearn



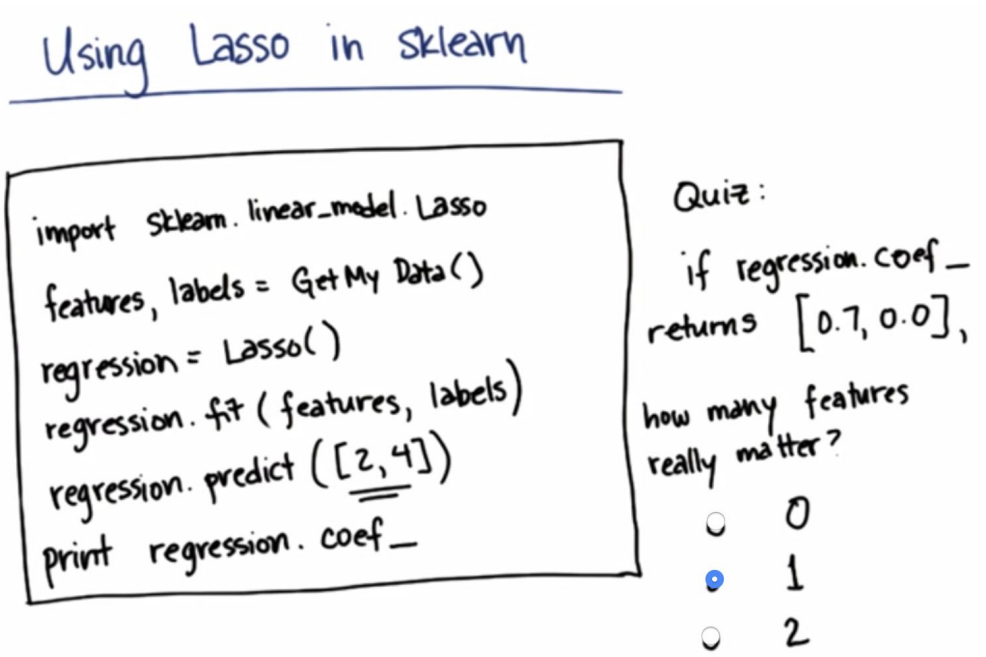
FYI, the documentation requests an array of shape = (n_samples, n_features). It happens in this case that sklearn is lenient and will still work if you give it a one dimensional array of features and you want a single prediction, but the answer regression.predict([[2,4]]) is a little more correct.

12.18 Lasso Prediction with SKLearn

Effect / coefficience (dat lukt ook nog wel zonder opzoeken op sklearn site ;))



12.20 Lasso Prediction with SKLearn



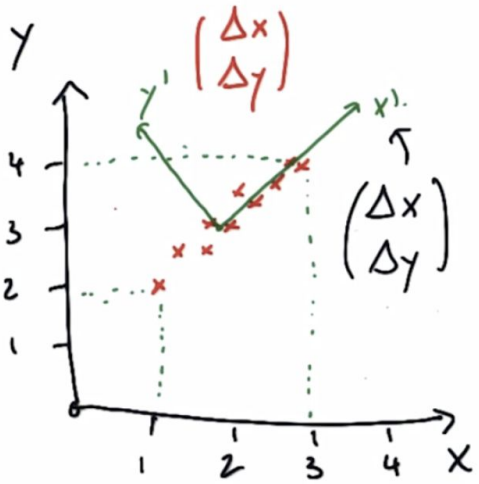
12.21 Overfitting: results of accuracy on TEST is low

Excercise 23. Ik krijg te hoge accuracy, lijkt iets te zijn met vectorize_text waar dit op voortborduurt. (antwoord zou 0.947 moeten zijn, ik krijg 1.0)

Lesson 13: Principal Component Analysis (Unsupervised learning)

a. Determining center of the data
b. Nieuw assenstelsel

PRINCIPAL COMPONENT ANALYSIS - PCA



$x = 2$

$y = 3$

$\Delta x = 1$

$\Delta y = 1$

$\Delta x =$ -1

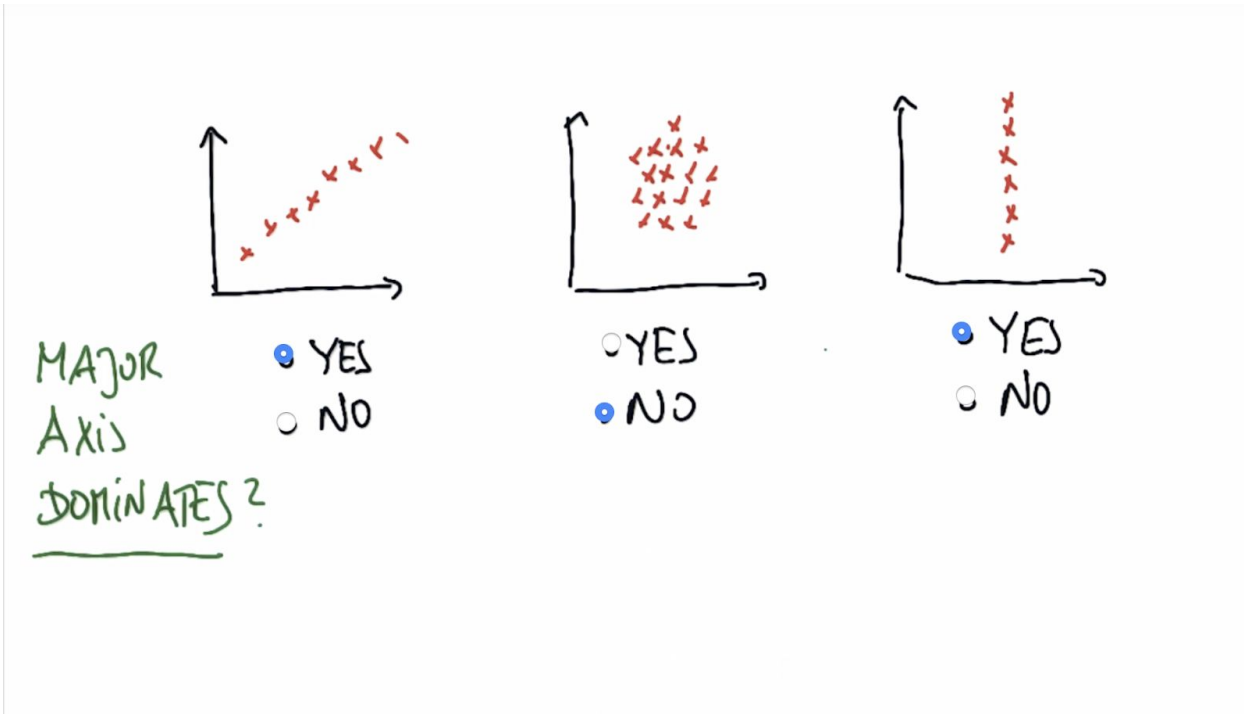
$\Delta y =$ 1

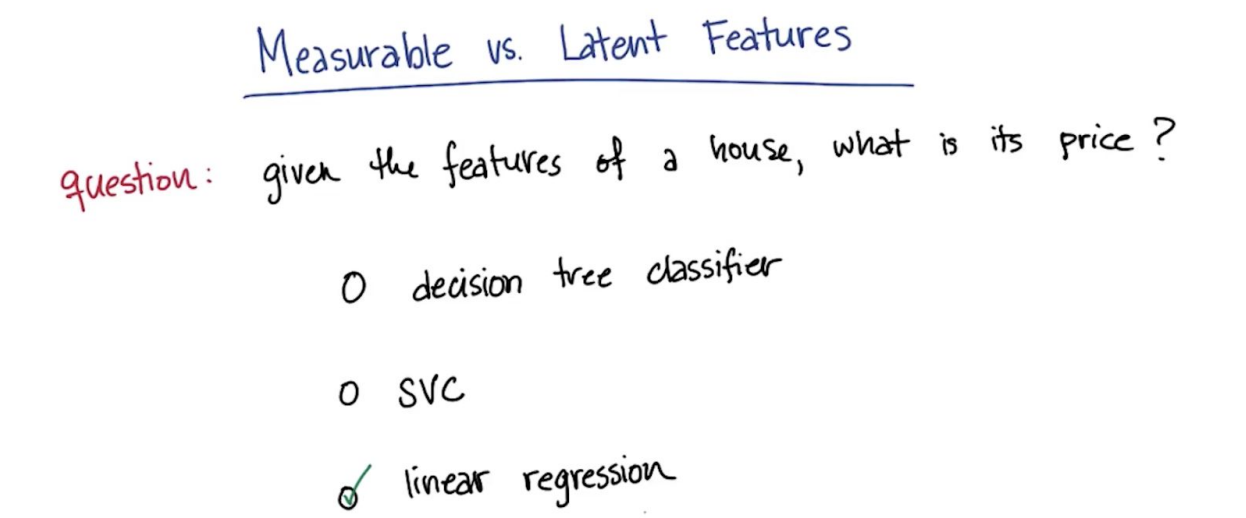
PCA result

- (●) YES
- ( ) NO

- (●) YES
- ( ) NO

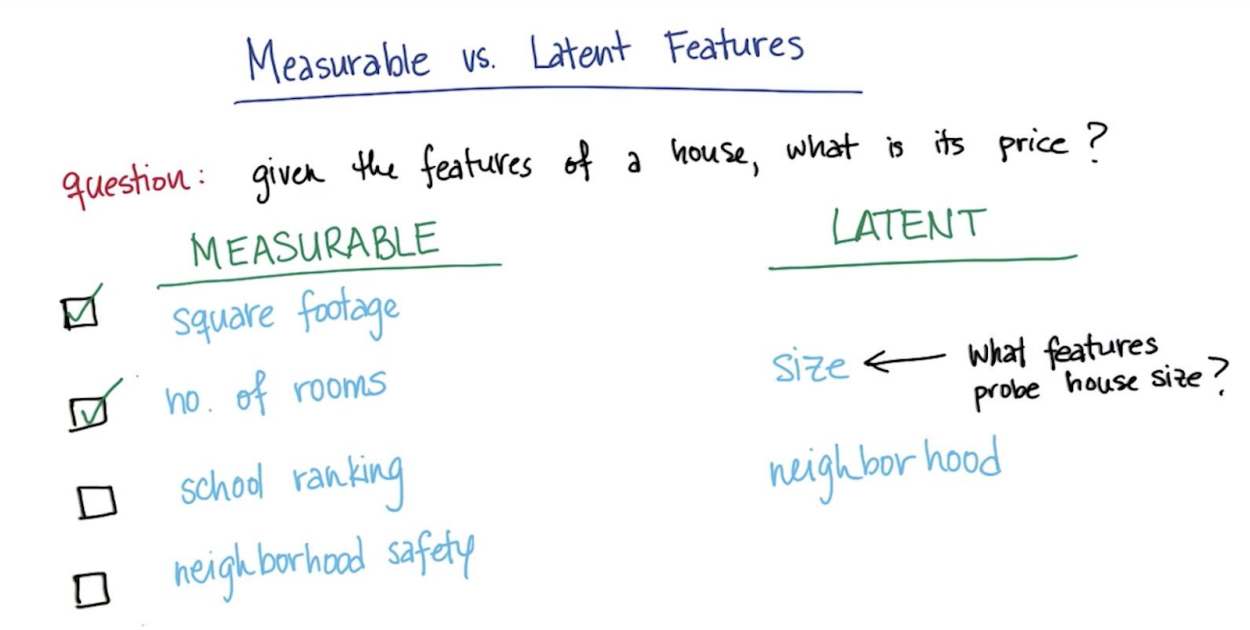
- (●) YES
- ( ) NO



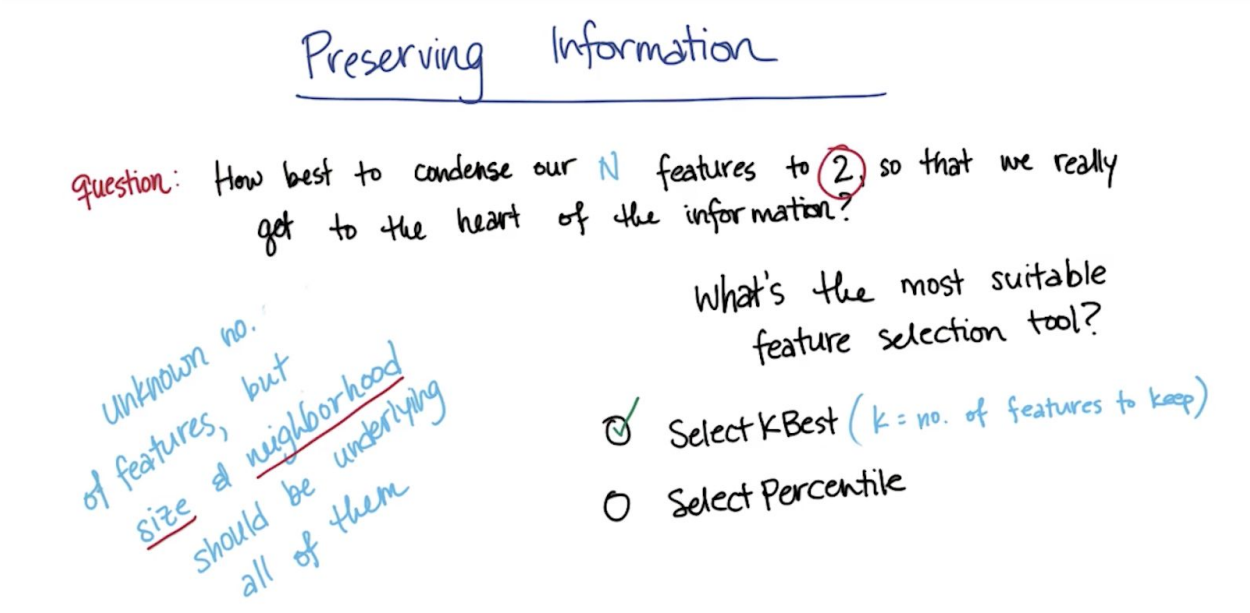


a number as an output, where this number's the price of the house.

14. Huisprijs is continue waarde: Regressie



the number of rooms are both ways of measuring how big the house is.



it will throw away all the features except the two that are the most powerful.

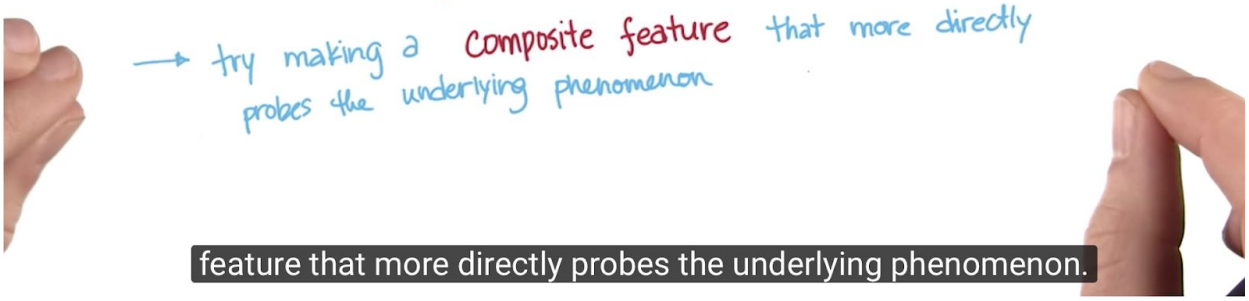
Als je zeker weet hoeveel features je wel of niet wilt kun je ook select percentile gebruiken (zoals in eerdere voorbeeld met 4 features waar je 2 wilt overhouden (dan zet je ‘m op 50% / 0.5)

Preserving Information

Question: How best to condense our 4 features to 2, so that we really get to the heart of the information?

→ many features, but I hypothesize a smaller no. of features actually driving the patterns

→ try making a composite feature that more directly probes the underlying phenomenon



feature that more directly probes the underlying phenomenon.

Preserving Information

Question: How best to condense our 4 features to 2, so that we really get to the heart of the information?

→ many features, but I hypothesize a smaller no. of features actually driving the patterns

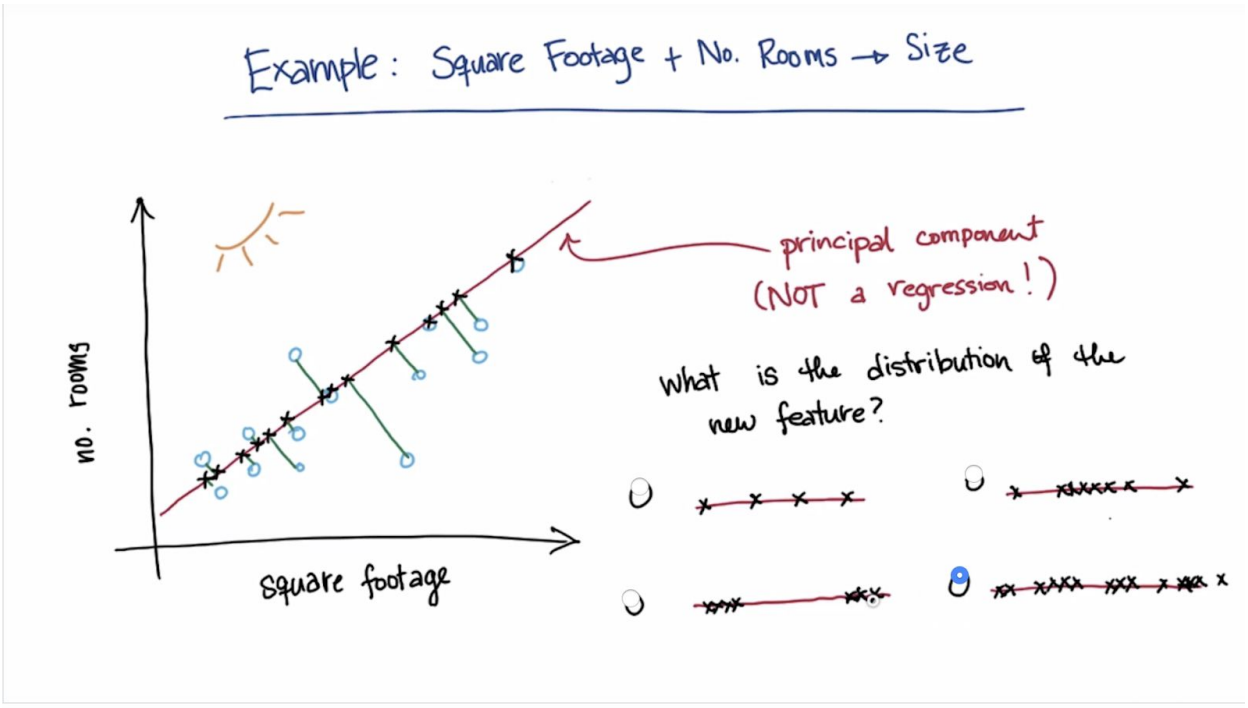
→ try making a composite feature that more directly probes the underlying phenomenon

Principle component

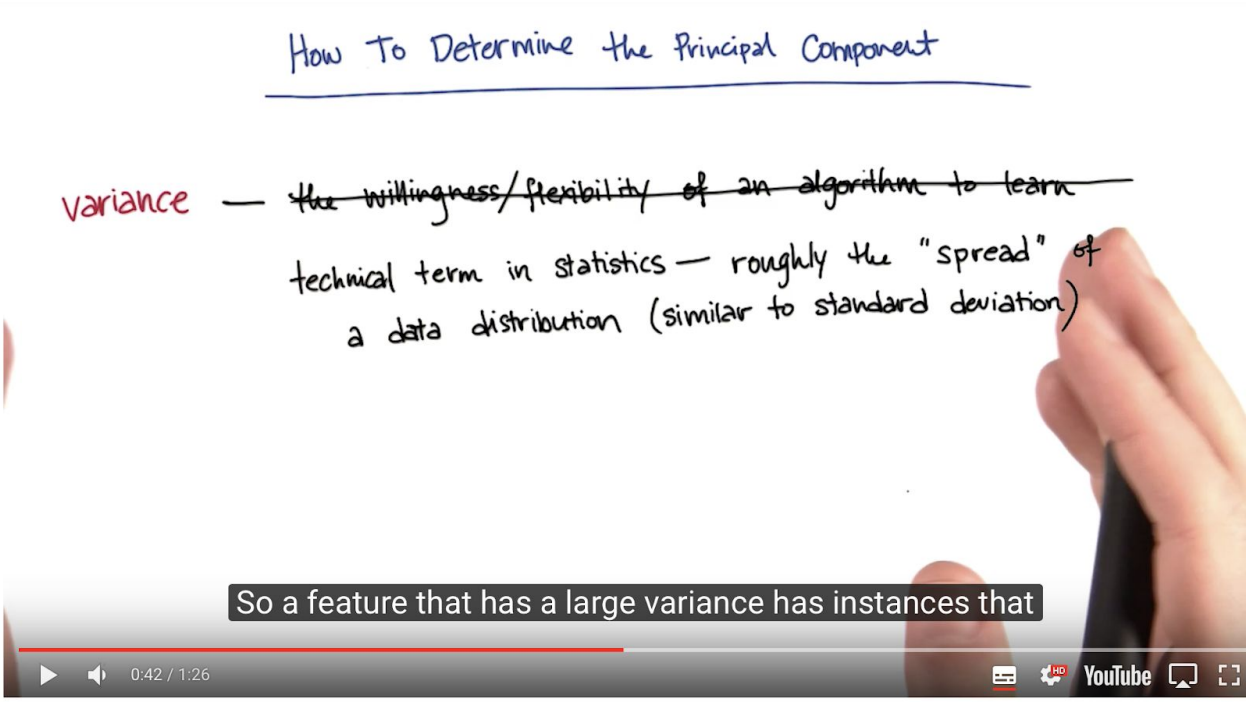
dimensionality reduction,

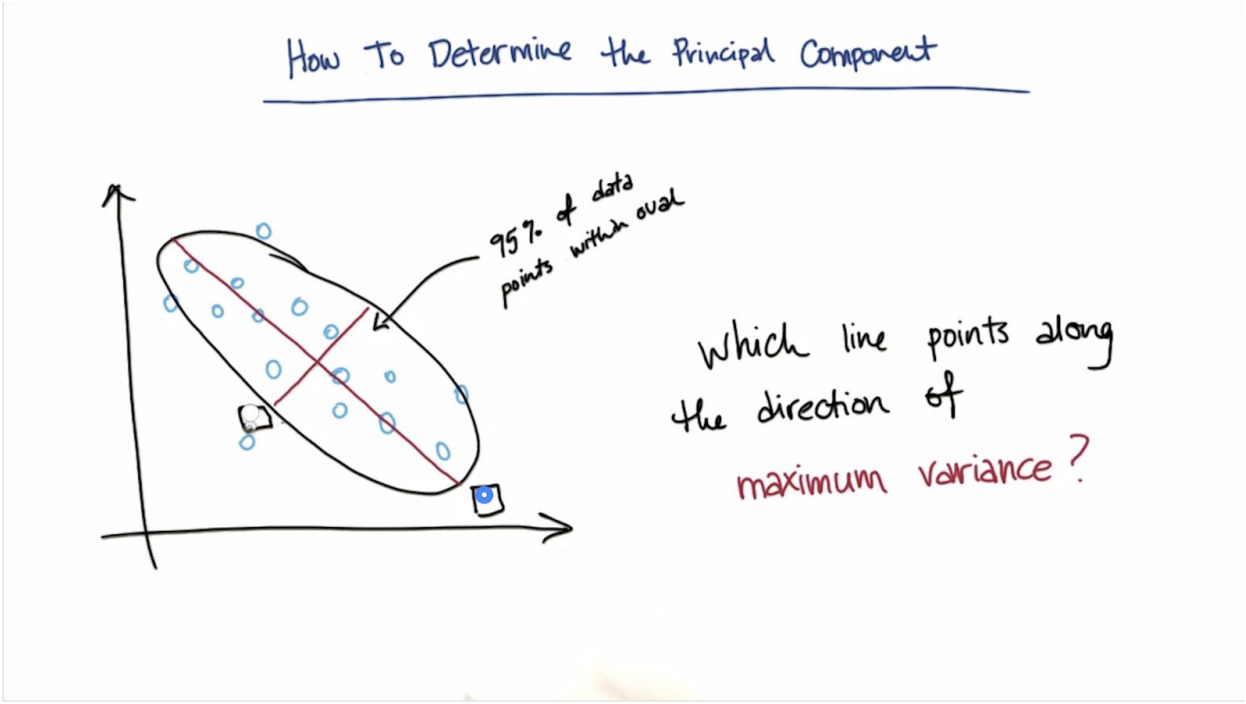
unsupervised learning

that in the latter half of this lesson.

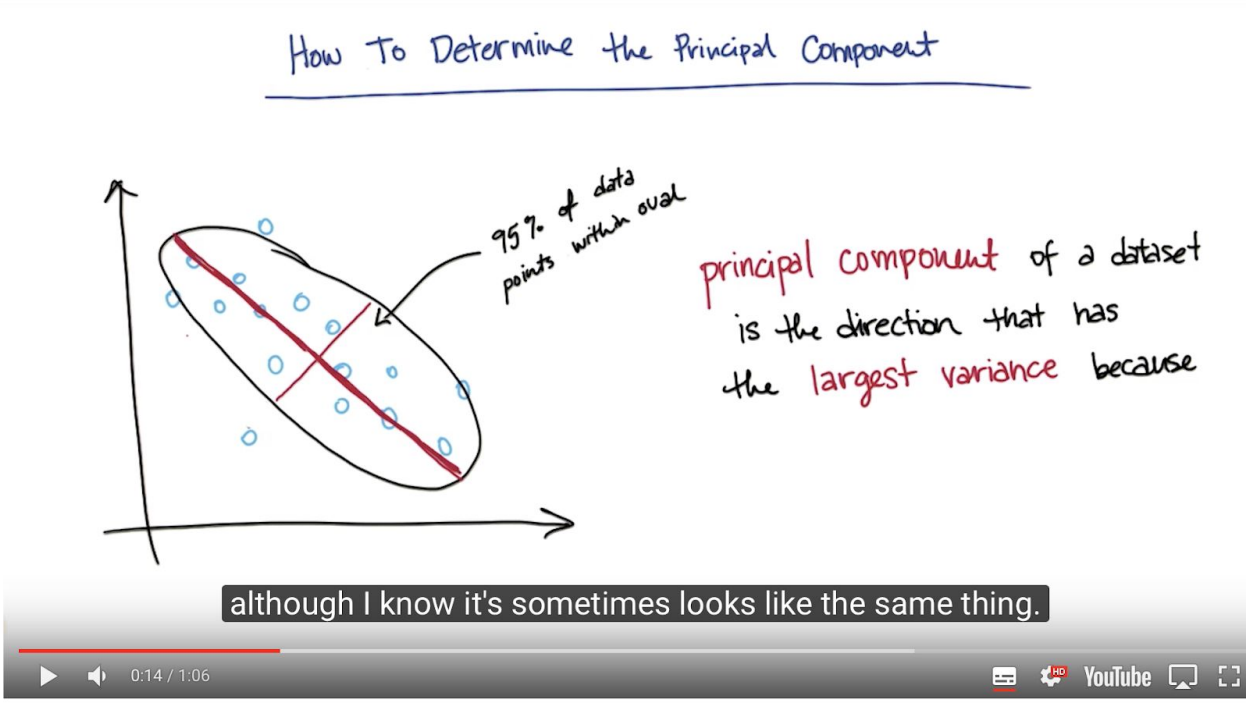

Example: Square Footage + No. Rooms → Size
principal component
(NOT a regression!)
no. rooms
square footage
What is the distribution of the new feature?

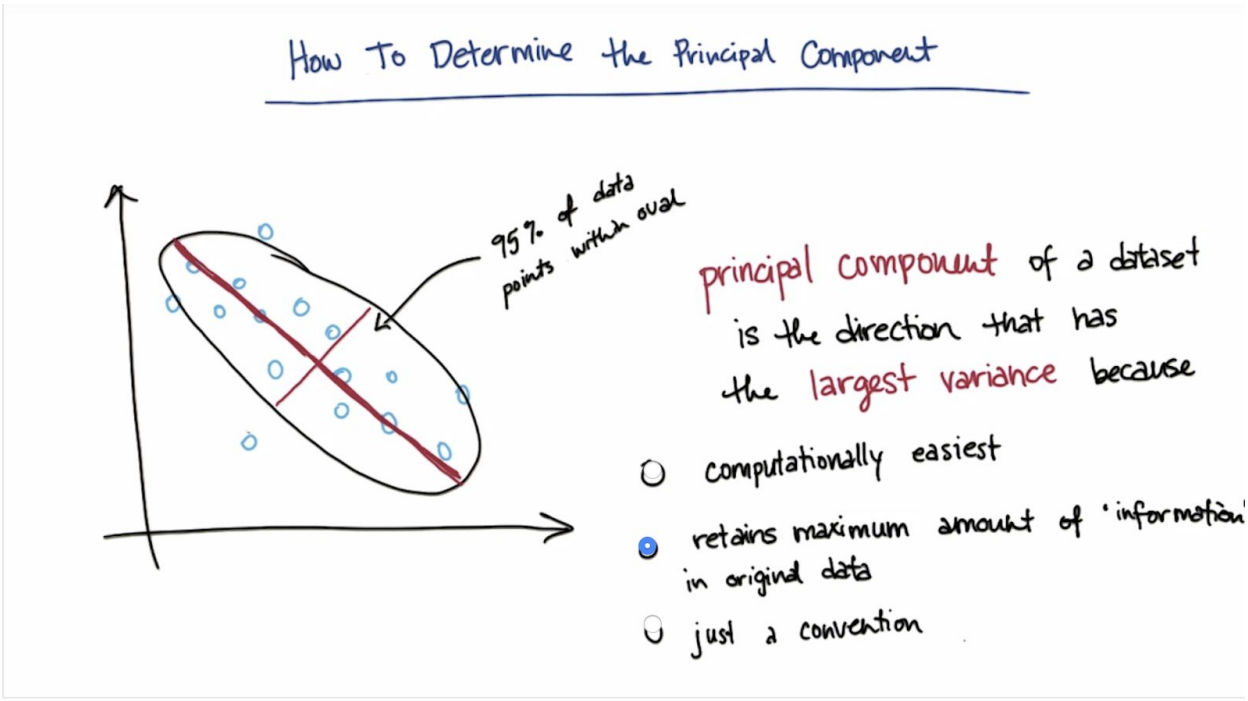
13.18

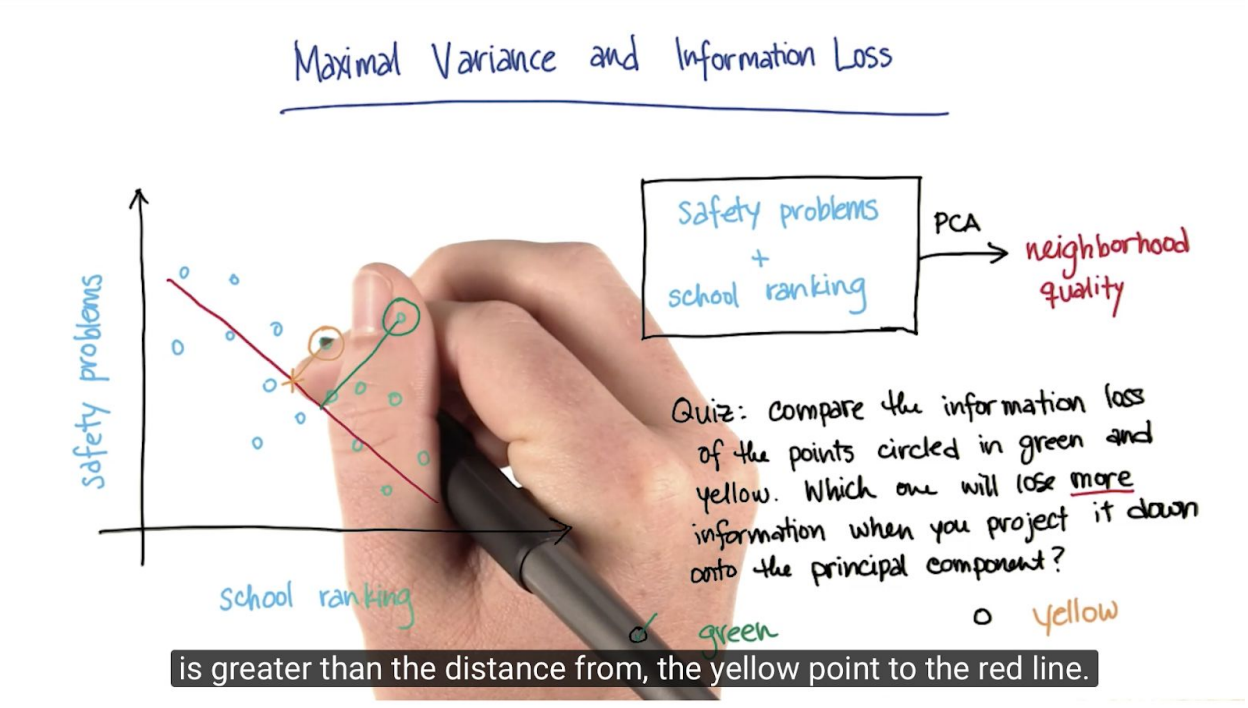

How To Determine the Principal Component
variance — ~~the willingness/flexibility of an algorithm to learn~~
technical term in statistics — roughly the "spread" of a data distribution (similar to standard deviation)
So a feature that has a large variance has instances that
0:42 / 1:26
YouTube


How To Determine the Principal Component
95% of data points within oval
Which line points along the direction of maximum variance?

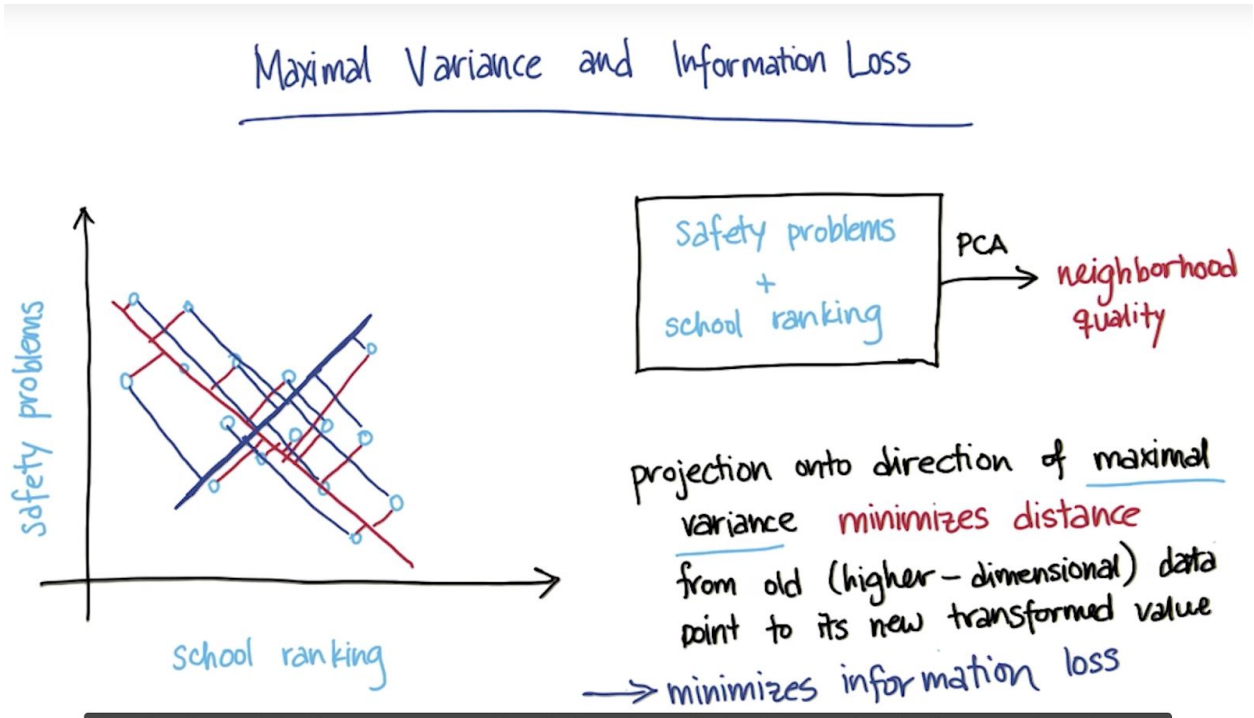
13.19


How To Determine the Principal Component
95% of data points within oval
principal component of a dataset is the direction that has the largest variance because
although I know it's sometimes looks like the same thing.
0:14 / 1:06
YouTube



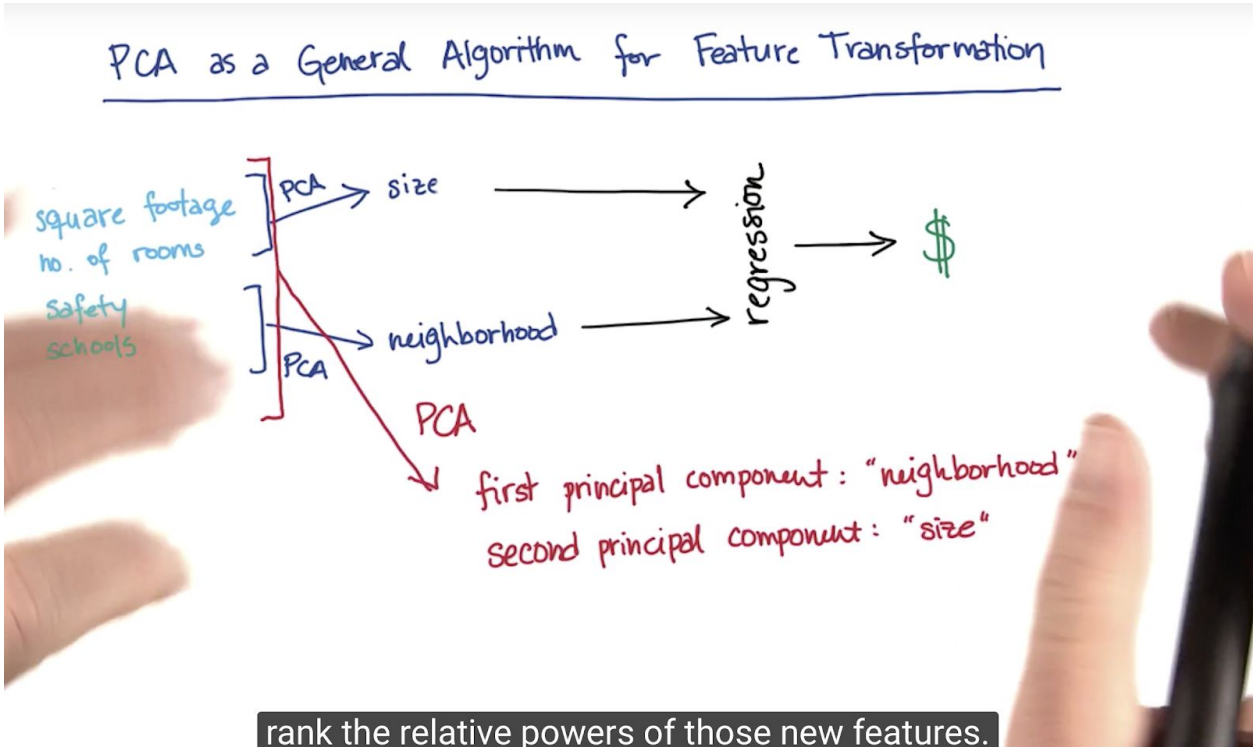


13.21 Minimizing information loss



And what this is necessarily doing is minimizing the information loss.

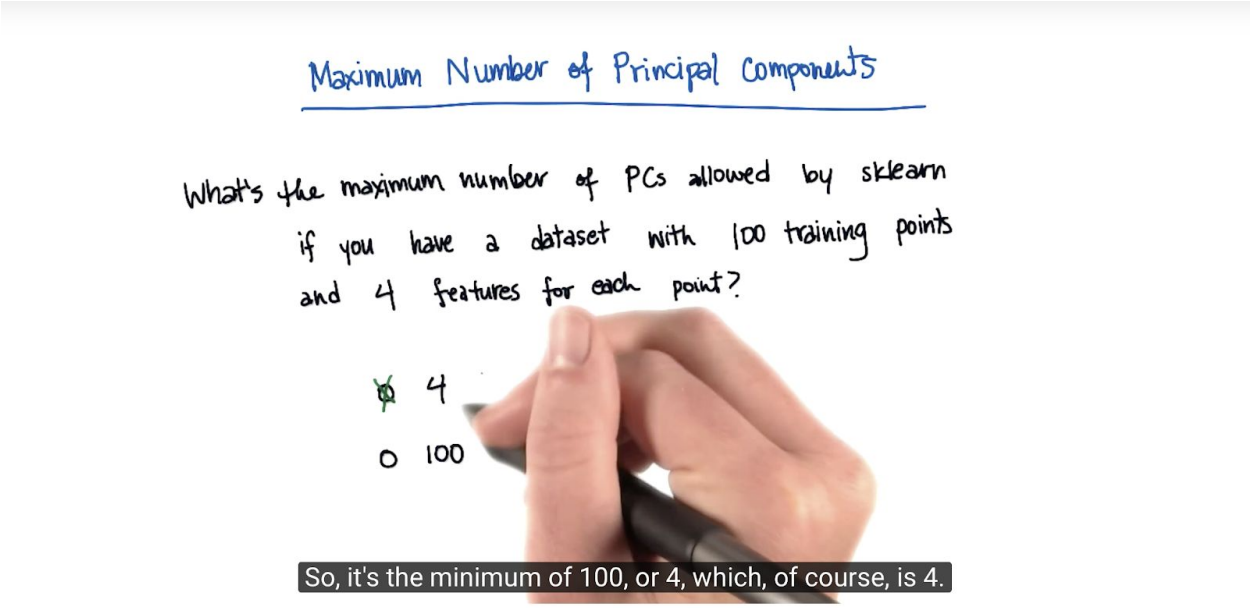
13.23



rank the relative powers of those new features.

13.24 Max number of components in Principal Component Analysis

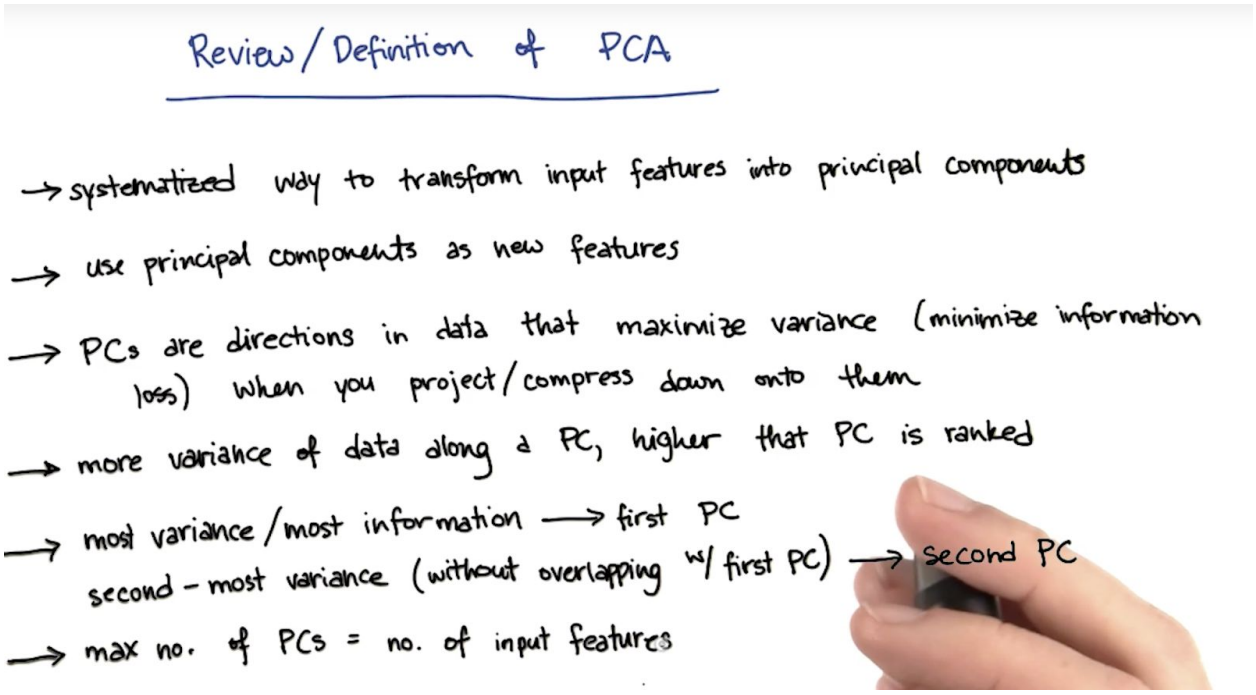
If you look at the sklearn documentation for PCA, you'll see that the max number of PCs is min(n_features, n_data_points). In this case, that means min(4, 100), or 4.



View Intro Back to Quiz

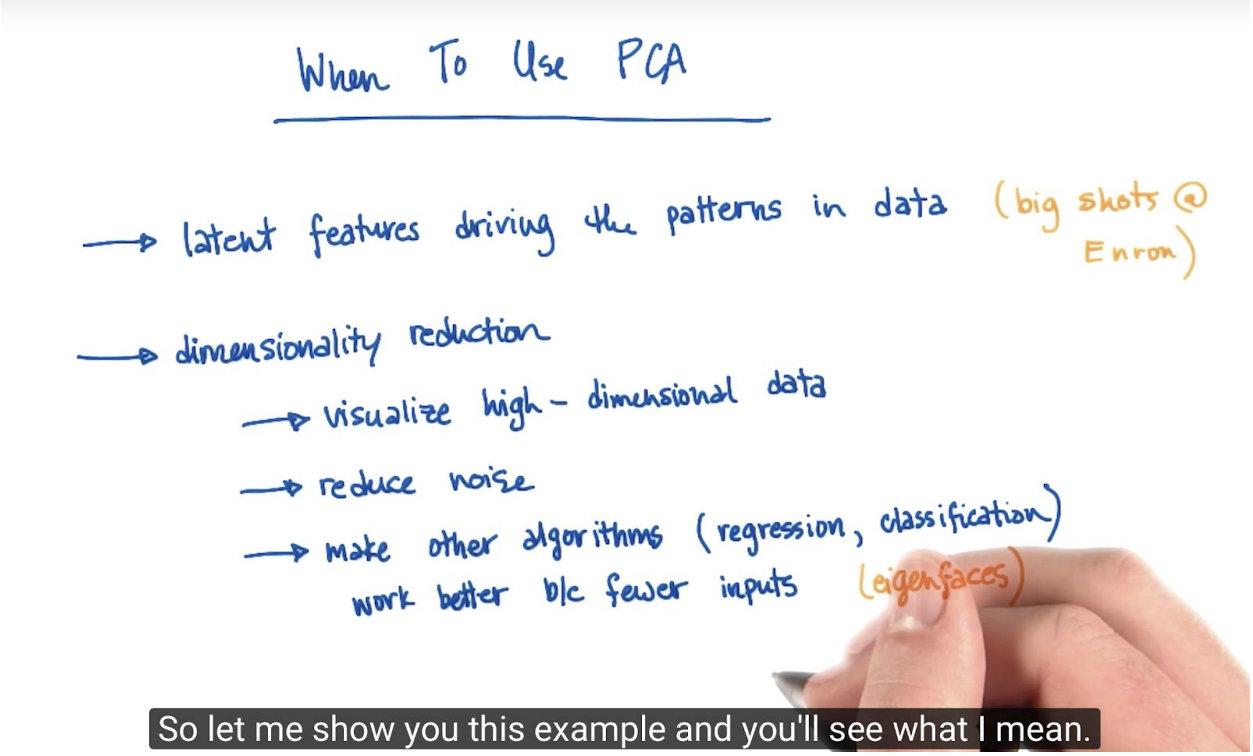
Have questions? Head to the forums for discussion with the Udacity Community.

25. PCA Summary / Definitions

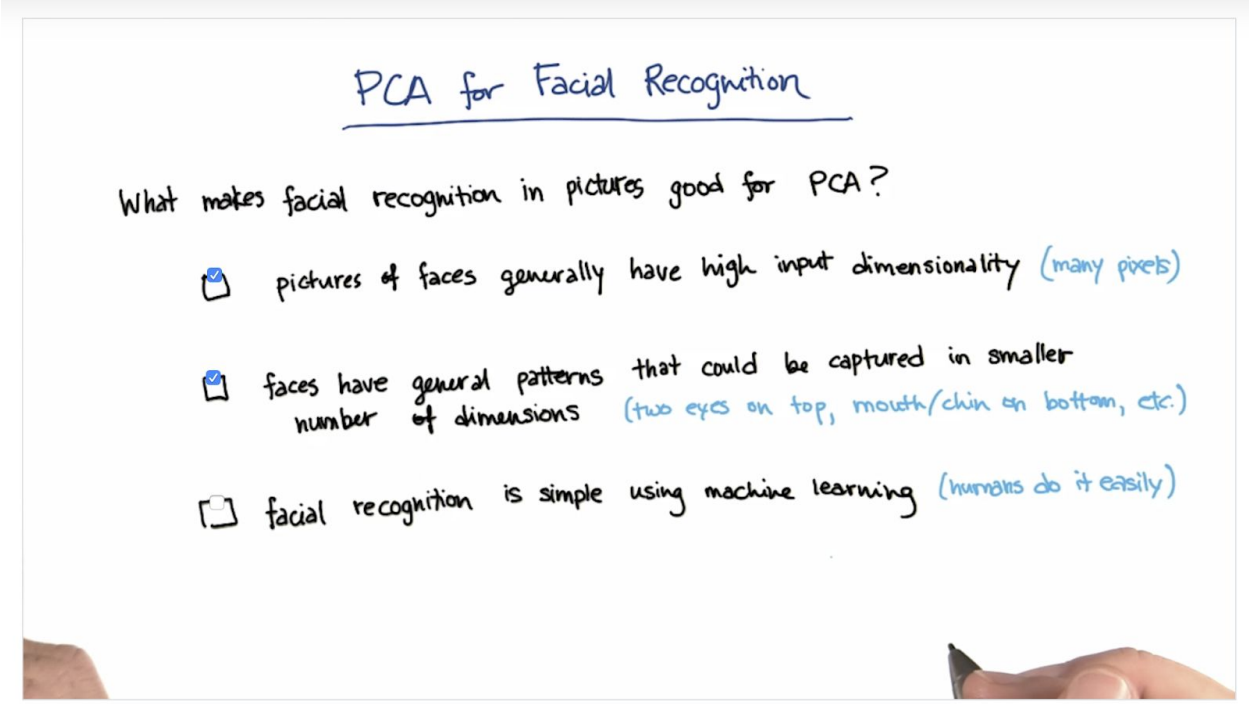


13.28 PCA in sklearn: [28. PCA in sklearn]

13.29 Belangrijk: PCA kun je gebruiken om complexe (multi-dimensionale dataproblemen aan te pakken, als je PCA als voor-analyse doet kun je daarna veel simpelere analyses zoals SVM er op los laten) Dit kan o.a. nuttig zijn voor gezichts- (en Jeroen:) objectherkenning. Ze noemen de term eigenfaces, dat gaat dus over face recognition (eigenwaardes van gezichten), eigenfaces is synoniem voor PCA maar dan in facial recognition setting.



13.30



Leuke demo met veel details over hoe je eea aanpakt:
https://classroom.udacity.com/courses/ud120/lessons/2962298545/concepts/30669485620923

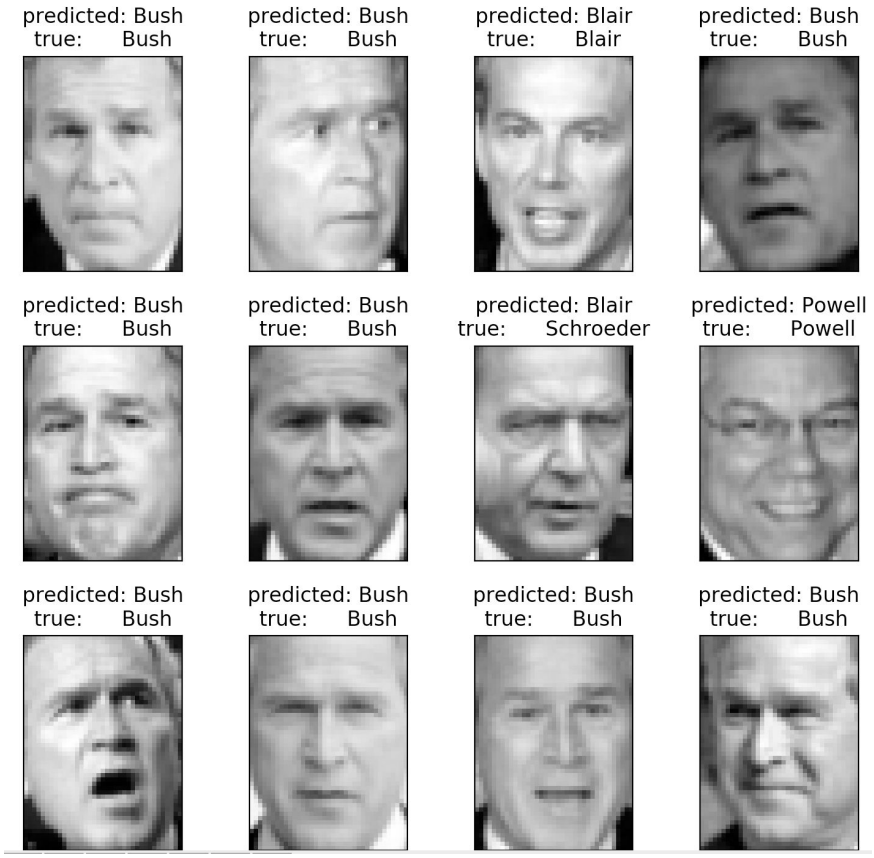
Helaas nog Python 2:
http://scikit-learn.org/0.15/auto_examples/applications/face_recognition.html

Zie ook: wel P3
http://scikit-learn.org/stable/auto_examples/applications/plot_face_recognition.html

Heeft Pillow nodig:
python3 -m pip install Pillow

13.34. Verklaarde variantie:

Explained variance ratio: [0.19346523 0.15116829 …]



Doet ie aardig goed…

13.35
Ideally, we hope that adding more components will give us more signal information to improve the classifier performance.

n_components = 10

F1 scores: precision recall f1-score support

Ariel Sharon	0.10	0.15	**0.12**	13
Colin Powell	0.43	0.53	0.48	60
Donald Rumsfeld	0.26	0.33	0.30	27
George W Bush	0.66	0.58	0.62	146
Gerhard Schroeder	0.17	0.20	0.18	25
Hugo Chavez	0.25	0.13	0.17	15
Tony Blair	0.50	0.39	0.44	36
micro avg	0.46	0.46	0.46	322
macro avg	0.34	0.33	0.33	322
weighted avg	0.49	0.46	0.47	322

n_components = 50

	precision	recall	f1-score	support
Ariel Sharon	0.56	0.69	**0.62**	13
Colin Powell	0.74	0.87	0.80	60
Donald Rumsfeld	0.48	0.52	0.50	27
George W Bush	0.86	0.82	0.84	146
Gerhard Schroeder	0.58	0.56	0.57	25
Hugo Chavez	0.90	0.60	0.72	15
Tony Blair	0.68	0.64	0.66	36
micro avg	0.75	0.75	0.75	322
macro avg	0.69	0.67	0.67	322
weighted avg	0.76	0.75	0.75	322

If you see a higher F1 score, does it mean the classifier is doing better, or worse?

- better
- worse
- need more information

Do you see any evidence of overfitting when using a large number of PCs?

- ○ no, more PCs always gives better performance
- ○ no, more PCs don't change performance
- ○ need more information
- ● yes, performance starts to drop with many PCs

Ik heb 'm nog een keer gedraaid met:

n_components = 250

	precision	recall	f1-score	support
Ariel Sharon	0.64	0.69	0.67	13
Colin Powell	0.75	0.90	0.82	60
Donald Rumsfeld	0.82	0.67	0.73	27
George W Bush	0.91	0.92	0.91	146
Gerhard Schroeder	0.87	0.80	0.83	25
Hugo Chavez	0.80	0.53	0.64	15
Tony Blair	0.82	0.78	0.80	36
micro avg	0.84	0.84	0.84	322
macro avg	0.80	0.76	0.77	322
weighted avg	0.84	0.84	0.84	322

Weinig winst en super traag.

Auto feedback van Sebastian: "Yes, the F1 score starts to drop."

Nog een keer gedraaid met 200

	precision	recall	f1-score	support
Ariel Sharon	0.60	0.69	0.64	13
Colin Powell	0.78	0.88	0.83	60
Donald Rumsfeld	0.73	0.70	0.72	27
George W Bush	0.91	0.88	0.90	146
Gerhard Schroeder	0.81	0.88	0.85	25
Hugo Chavez	0.73	0.53	0.62	15
Tony Blair	0.91	0.86	0.89	36

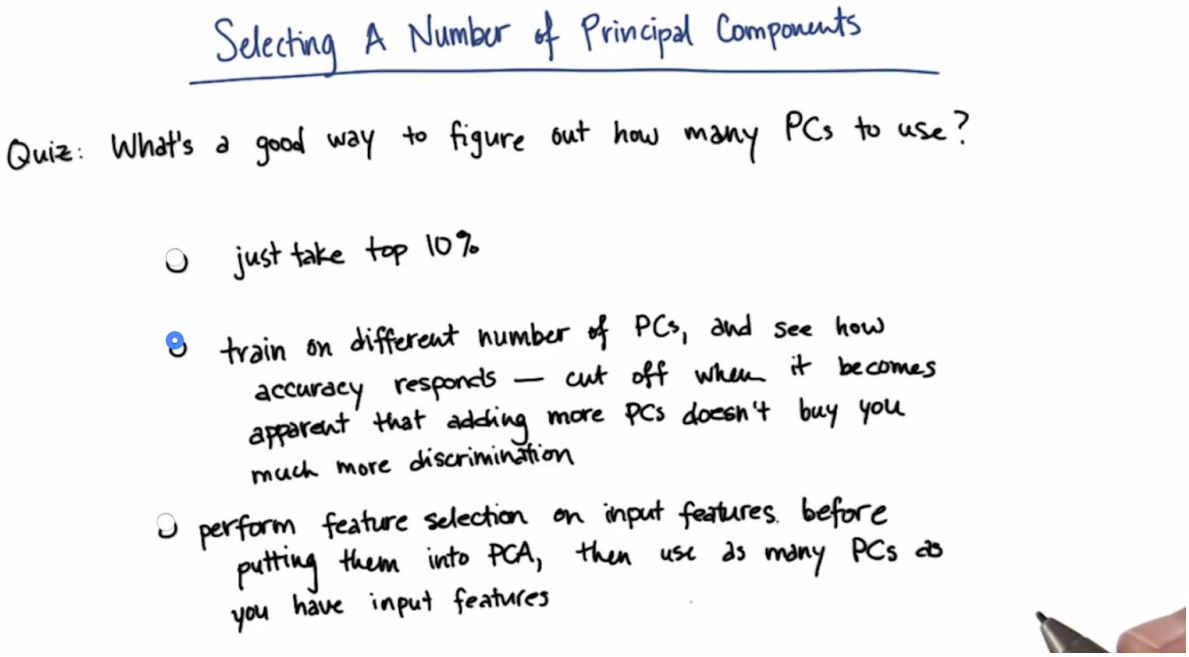
micro avg	0.84	0.84	0.84	322
macro avg	0.78	0.78	0.78	322
weighted avg	0.84	0.84	0.84	322

En om 't af te leren met 500 (kun je mooi ff koffie drinken in tussentijd)

F1 scores:	precision	recall	f1-score	support
Ariel Sharon	0.57	0.92	0.71	13
Colin Powell	0.67	0.88	0.76	60
Donald Rumsfeld	0.63	0.63	0.63	27
George W Bush	0.85	0.77	0.81	146
Gerhard Schroeder	0.62	0.52	0.57	25
Hugo Chavez	0.67	0.53	0.59	15
Tony Blair	0.77	0.64	0.70	36
micro avg	0.74	0.74	0.74	322
macro avg	0.68	0.70	0.68	322
weighted avg	0.75	0.74	0.74	322

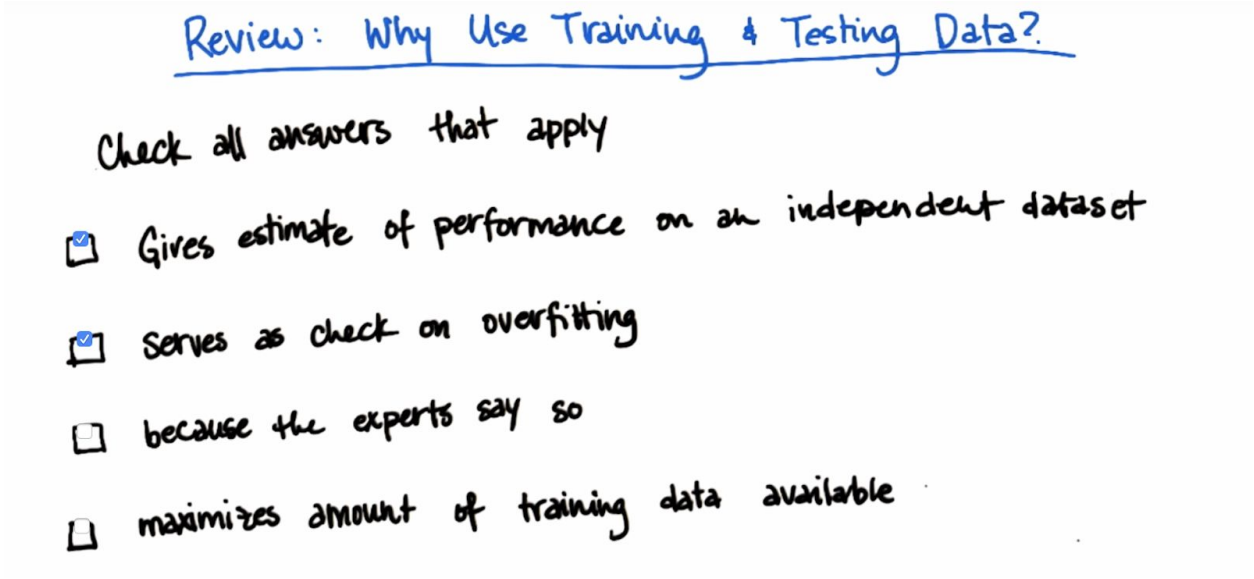
Je ziet nu duidelijk die F1 scores naar beneden gaan (niet voor Sharon, wel voor de anderen overigens).
https://en.wikipedia.org/wiki/F1_score

13.38:



That's right, it's all about trying different things and seeing what works best.

Lesson 14. Cross Validation



Wat ze vertelt is beetje anders in Python 3:

from sklearn.model_selection **import** train_test_split

Dus staat nu allemaal in sklearn.model_selection

14.3

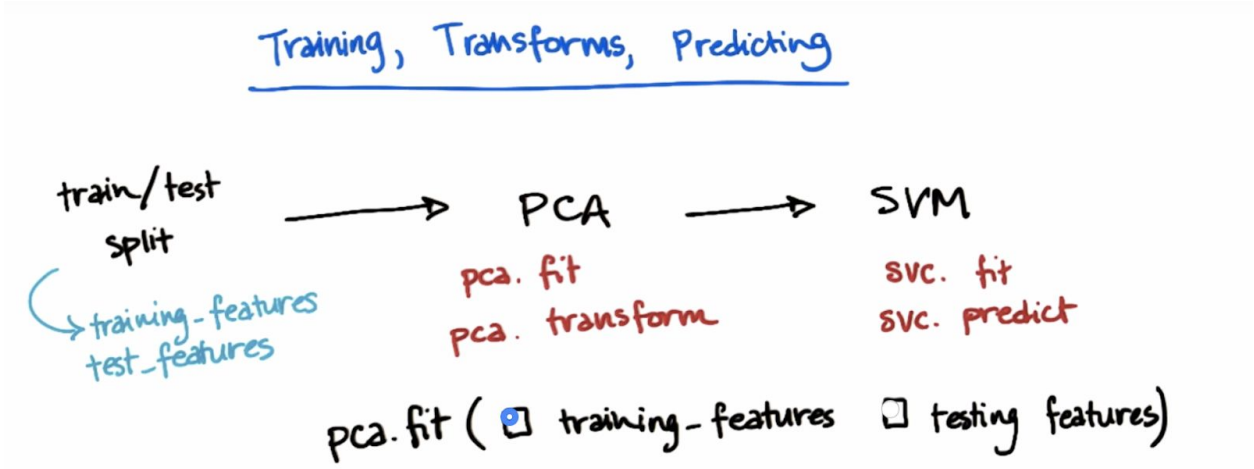
features_train, features_test, labels_train, labels_test = train_test_split(features, labels, test_size=0.25, random_state=42)

In Quiz nog voor Python 2.x:
features_train, features_test, labels_train, labels_test = cross_validation.train_test_split(features, labels, test_size=0.25, random_state=42)

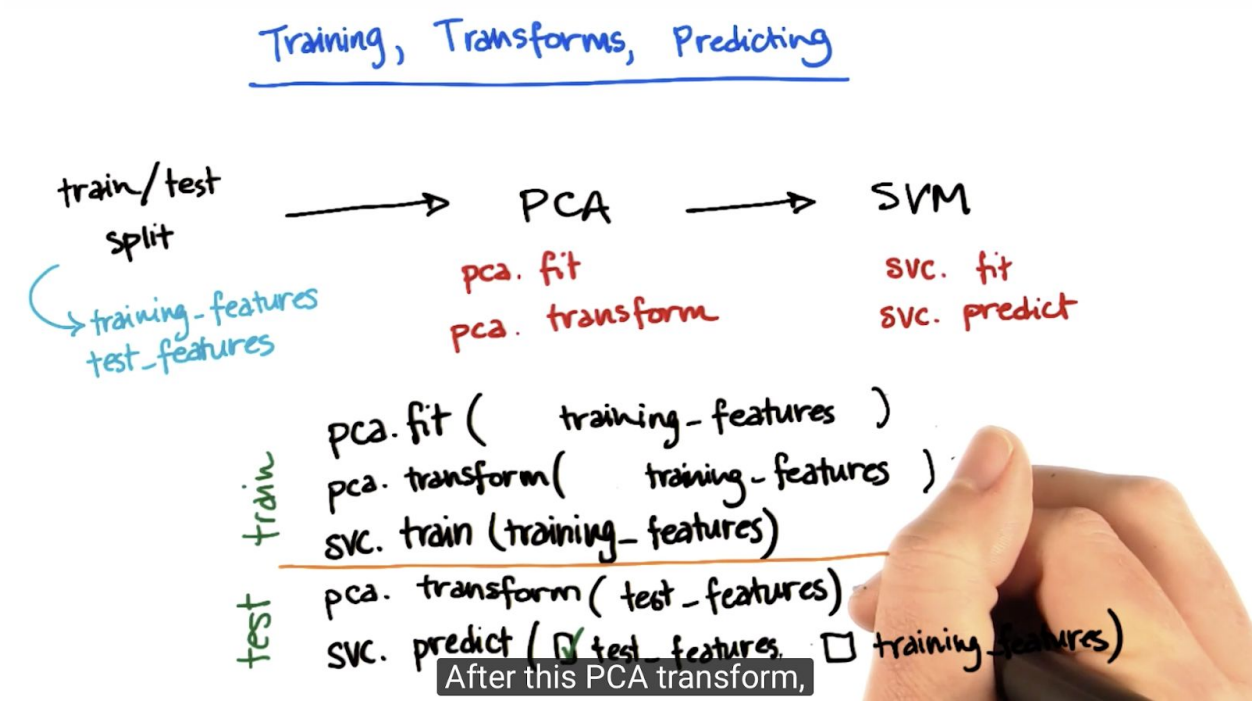
Let op, moet op regel 37, had ik eerst over het hoofd gezien, je hoeft blijkbaar alleen de argumenten van de methode in te vullen…

14.4 Deze is essentieel:

https://classroom.udacity.com/courses/ud120/lessons/2960698751/concepts/29743786260923

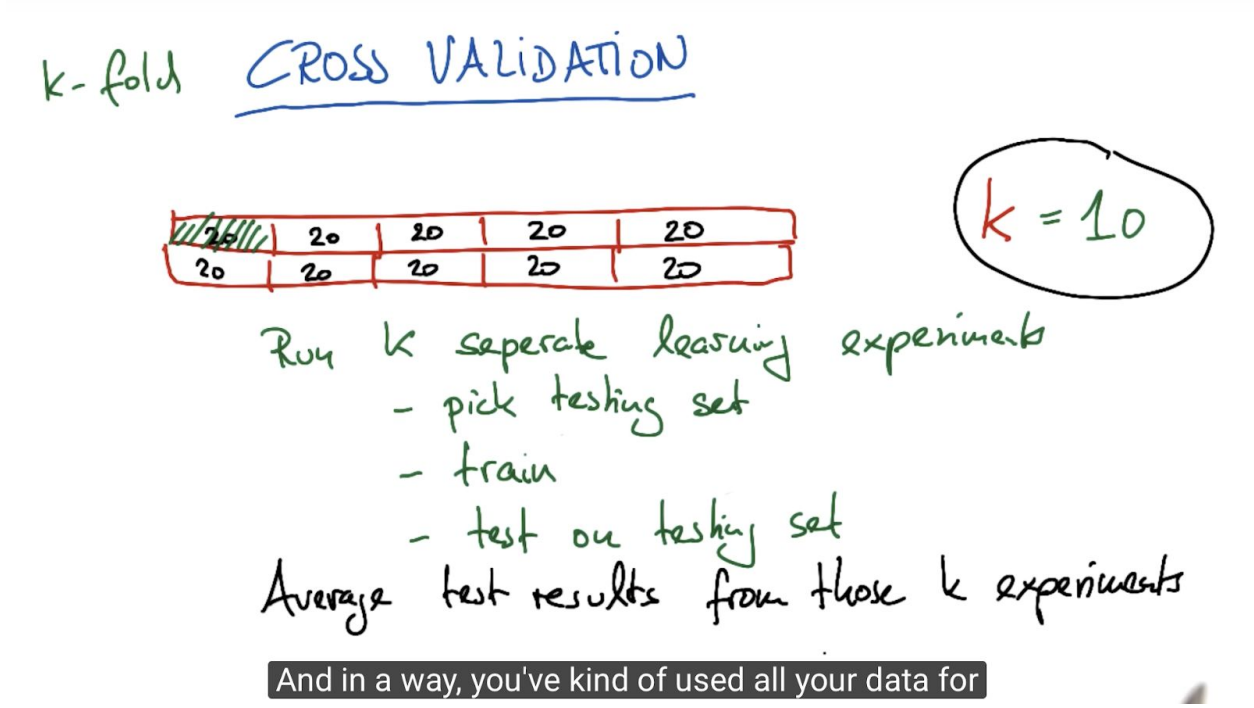


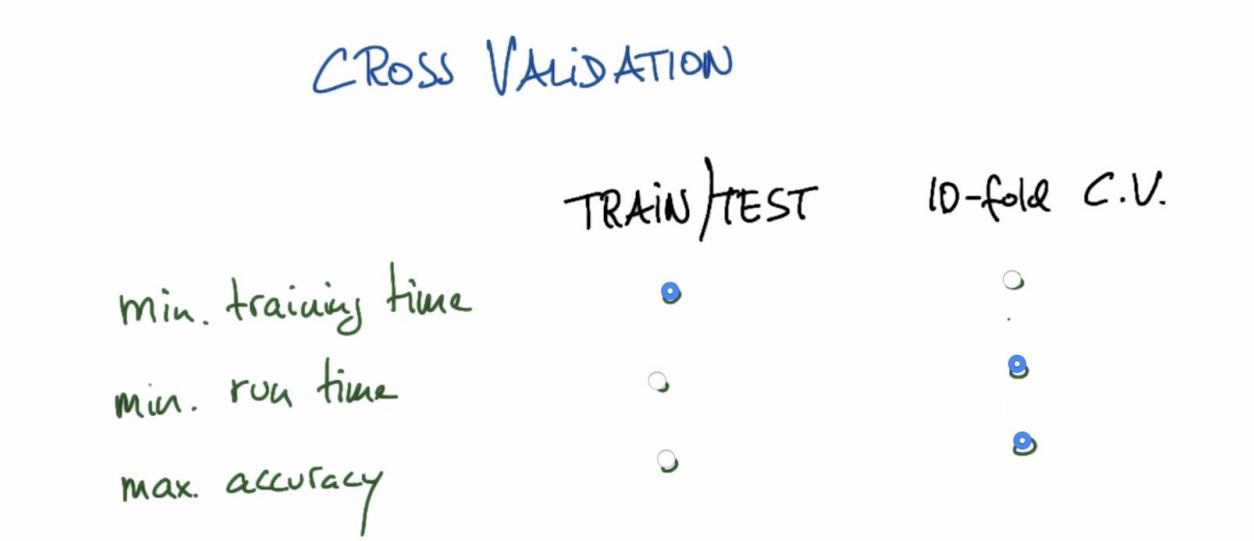
Belangrijke stappen:



Let op! Die fit doe je dus één keer, dan trek je zowel training_features als test_features er uit met pca.transform en voert dat aan je ML functie predict.

14.8 K-Fold Cross Validation





KFold zit nu (uiteraard) in **sklearn.model_selection.KFold**

http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

14.10 Praktische adviezen K-Fold

*If our original data comes in some sort of sorted fashion, then we will want to first **shuffle** the order of the data points before splitting them up into folds, or otherwise randomly assign data points to each fold. If we want to do this using `KFold()`, then we can add the "shuffle = True" parameter when setting up the cross-validation object.*

N.b. train_test_split heeft ook een random_state argument

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
Zie o.a.
https://stackoverflow.com/questions/28064634/random-state-pseudo-random-numberin-scikit-learn

If we have concerns about class imbalance, then we can use the `StratifiedKFold()` class instead. Where `KFold()` assigns points to folds without attention to output class, `StratifiedKFold()` assigns data points to folds so that each fold has approximately the same number of data points of each output class. This is most useful for when we have imbalanced numbers of data points in your outcome classes (e.g. one is rare compared to the others). For this class as well, we can use "shuffle = True" to shuffle the data points' order before splitting into folds.

14.12: GridSearchCV

http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

GridSearchCV is a way of systematically working through multiple combinations of parameter tunes, cross-validating as it goes to determine which tune gives the best performance. The beauty is that it can work through many combinations in only a couple extra lines of code.

Here's an example from the sklearn documentation:

```
parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
svr = svm.SVC()
clf = grid_search.GridSearchCV(svr, parameters)
clf.fit(iris.data, iris.target)
```

Let's break this down line by line.

```
parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
```

A dictionary of the parameters, and the possible values they may take. In this case, they're playing around with the kernel (possible choices are 'linear' and 'rbf'), and C (possible choices are 1 and 10).

Then a 'grid' of all the following combinations of values for (kernel, C) are automatically generated:

('rbf', 1)	('rbf', 10)
('linear', 1)	('linear', 10)

Each is used to train an SVM, and the performance is then assessed using cross-validation.

`svr = svm.SVC()`

This looks kind of like creating a classifier, just like we've been doing since the first lesson. But note that the "clf" isn't made until the next line--this is just saying what kind of algorithm to use. Another way to think about this is that the "classifier" isn't just the algorithm in this case, it's algorithm plus parameter values. Note that there's no monkeying around with the kernel or C; all that is handled in the next line.

`clf = grid_search.GridSearchCV(svr, parameters)`

This is where the first bit of magic happens; the classifier is being created. We pass the algorithm (*svr*) and the dictionary of parameters to try (*parameters*) and it generates a grid of parameter combinations to try.

`clf.fit(iris.data, iris.target)`

And the second bit of magic. The fit function now tries all the parameter combinations, and returns a fitted classifier that's automatically tuned to the optimal parameter combination. You can now access the parameter values via `clf.best_params_`.

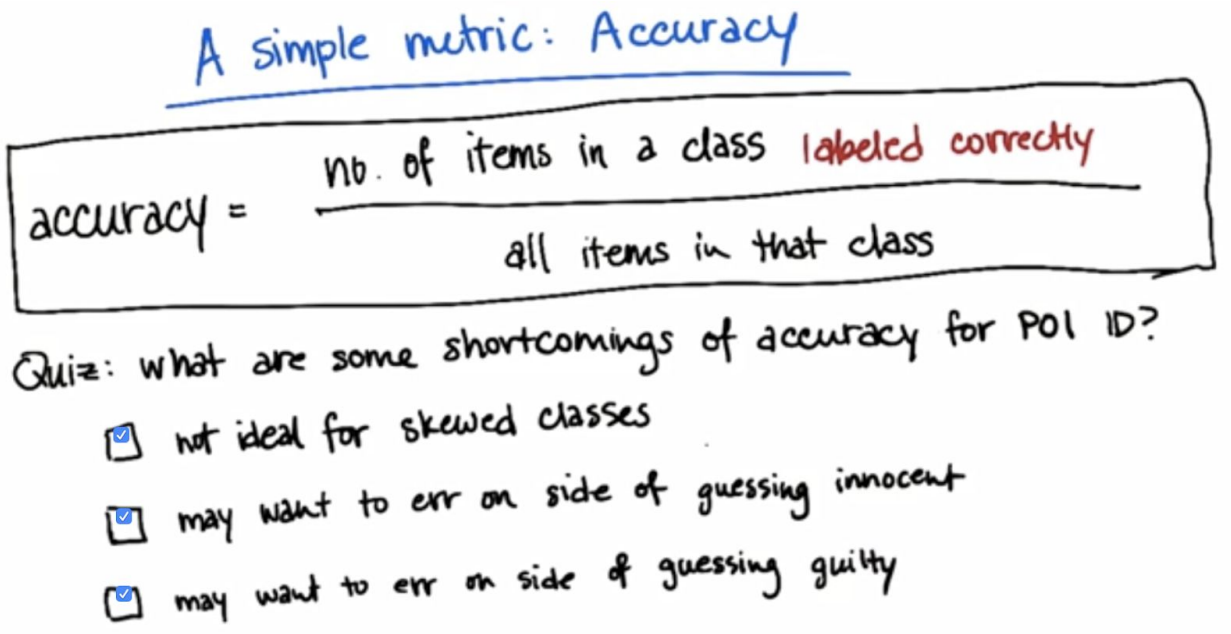
14.13 Eigenface values

http://scikit-learn.org/stable/auto_examples/applications/plot_face_recognition.html

What parameters of the SVM are being auto-tuned with GridSearchCV in the eigenfaces example?

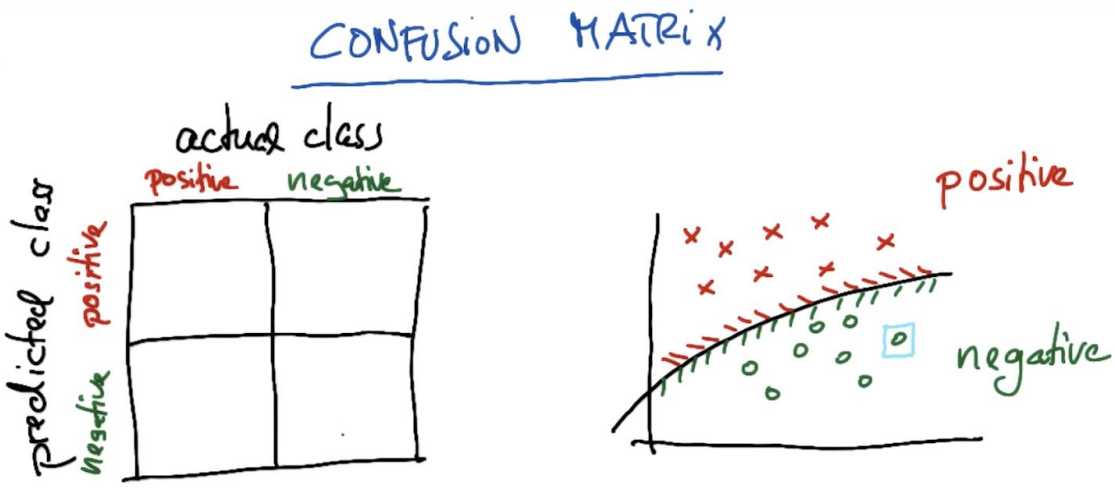
- [x] C
- [x] gamma
- [ ] kernel

Lesson 15 Evaluation Metrics

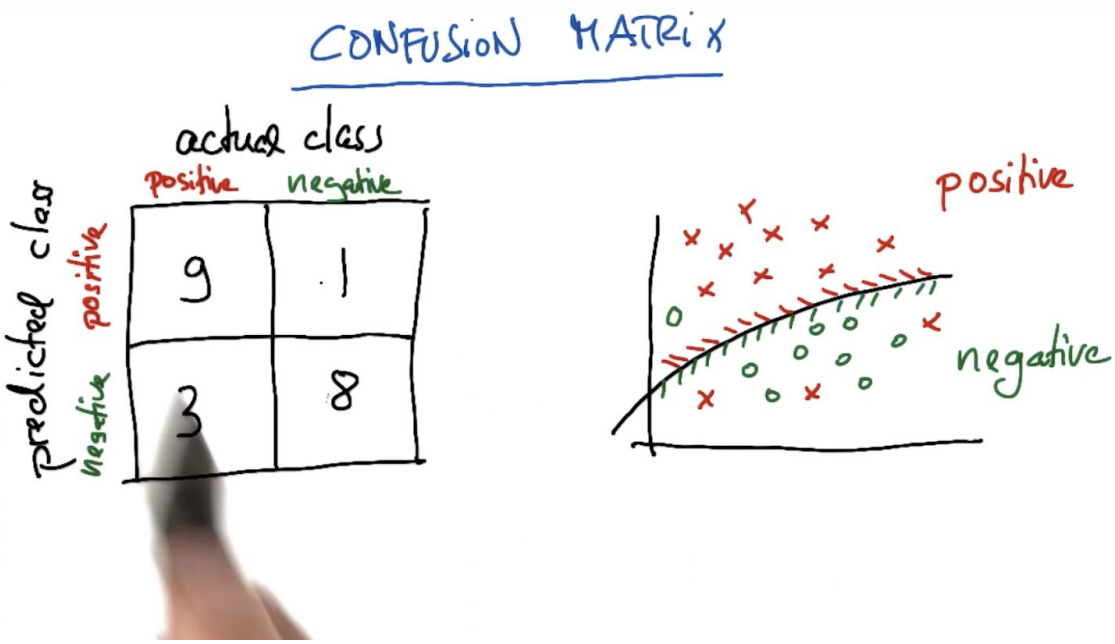


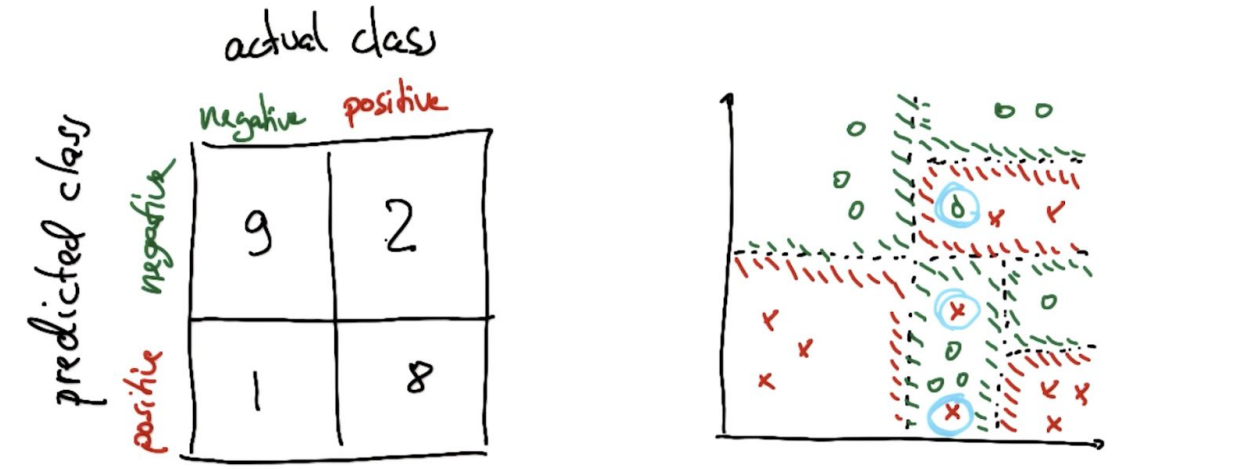
Nuttig: https://classroom.udacity.com/courses/ud120/lessons/2945338635/concepts/30294986070923

15.5. Confusion Matrices

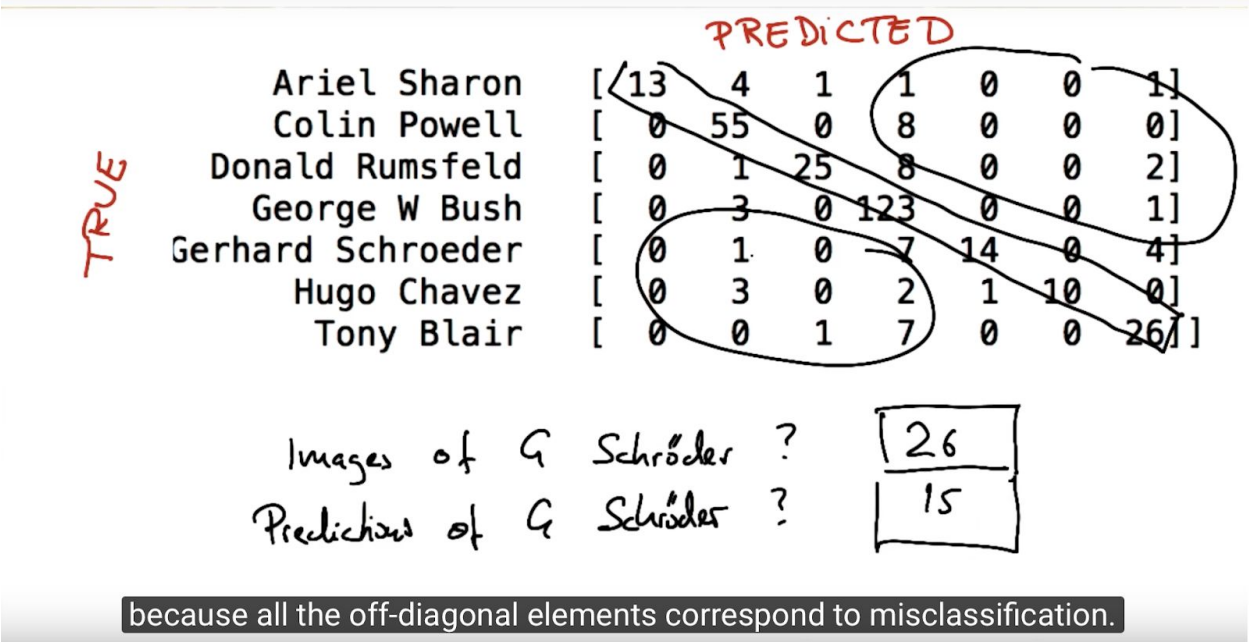


Ingevuld (incl. nieuwe punten):

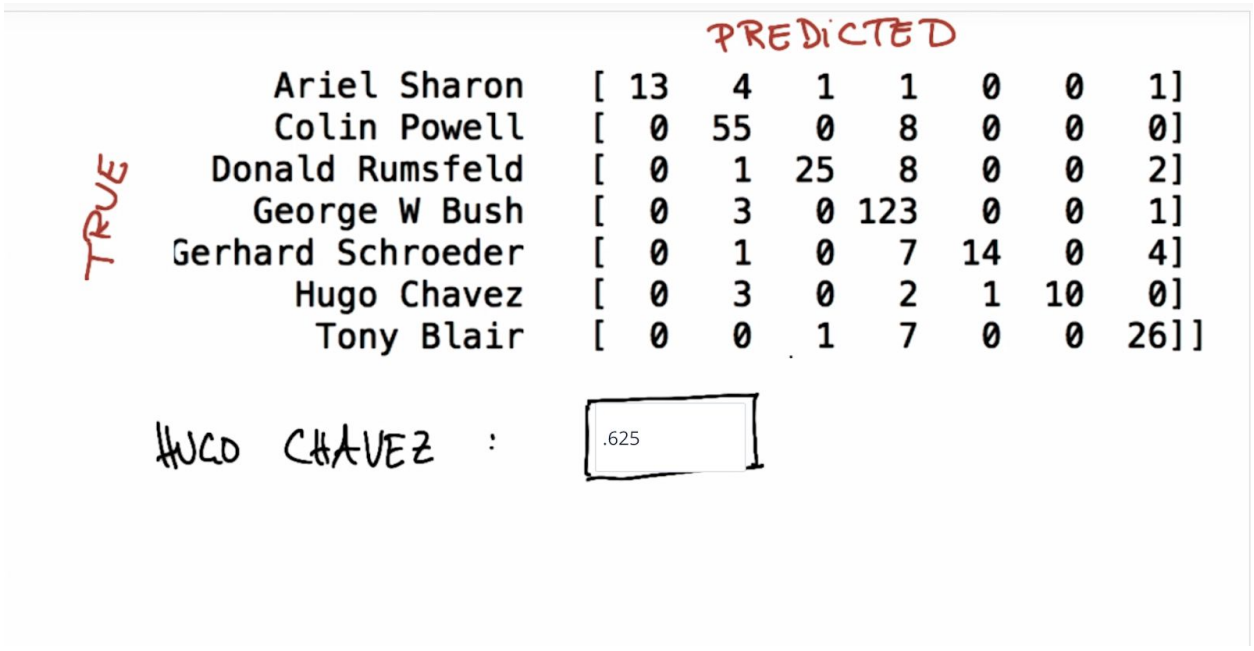




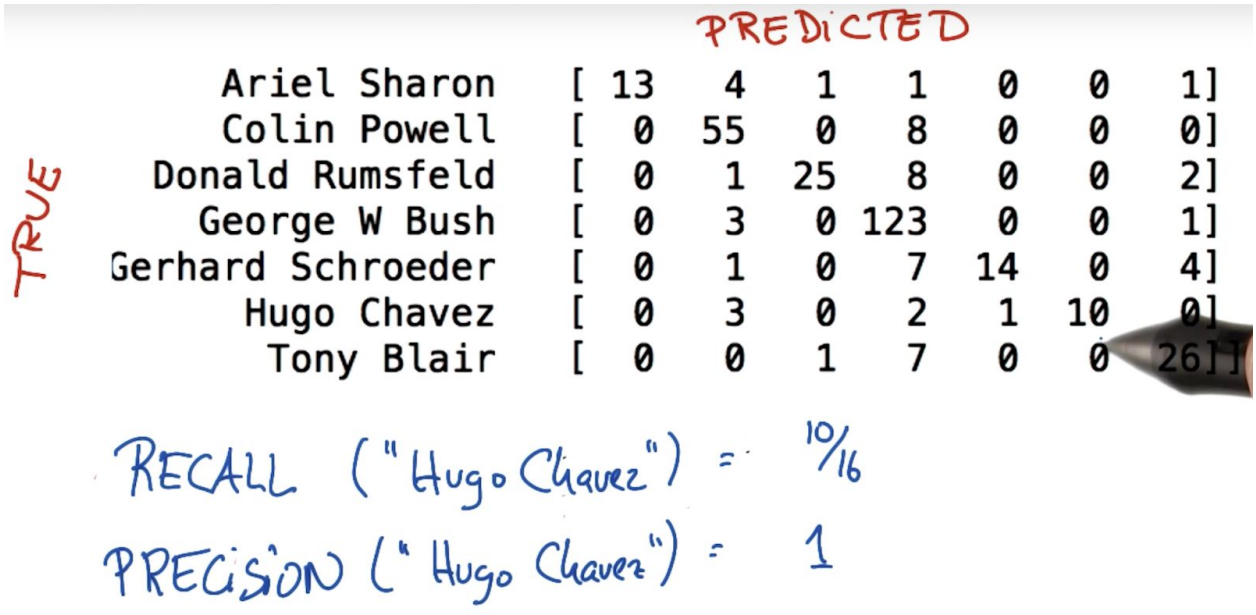
15.13 Als je diagonaal significant hogere waardes heeft dan !diagonaal weet je dat je classificatie algoritme goed presteert:



15.14 Kans dat Hug Chavez correct geclassificieerd wordt: 6x misgekwalificeerd en 10x correct gekwalificeerd 10 / 16 = 0.625



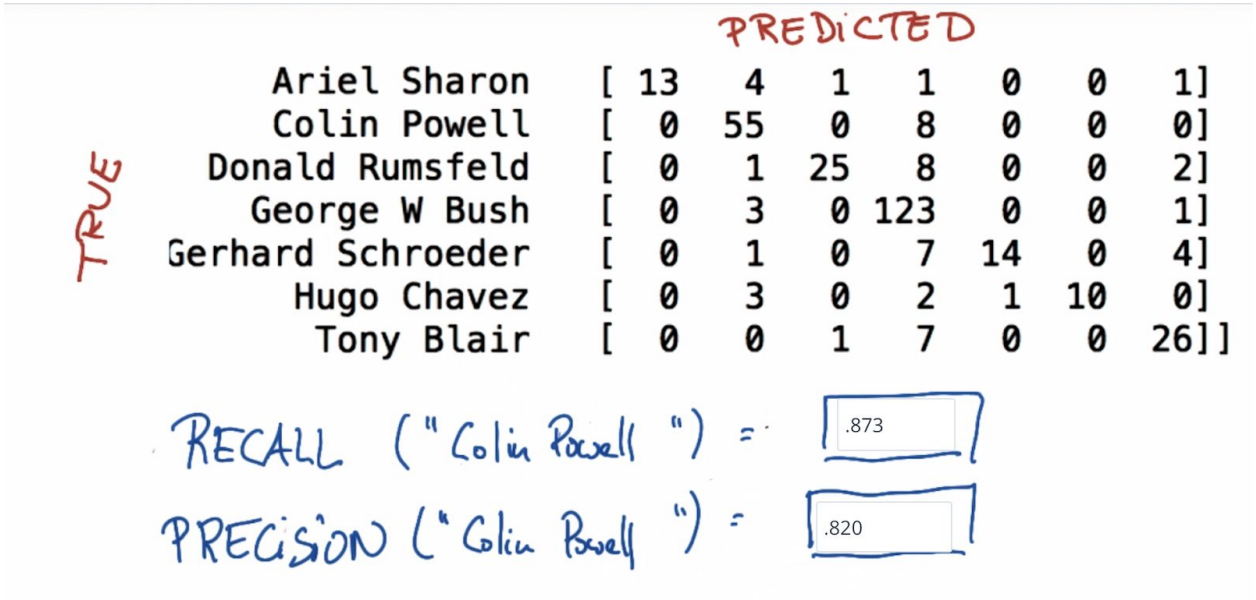
15.15 gegeven predictor die zegt dat het Hugo Chavez is en dat het inderdaad Hugo Chavez is, hoe groot is kans daar op. Als de predictor het zegt is kans 100% (dus P = 1.0) (aangezien de verticale rij (lees: kolom) voor Hugo Chavez laat zien dat rest 0 is). Correct in 10 / 10 van de gevallen.



15.16: Def. **RECALL**: quotiënt van de kans dat ons algoritme Hugo Chavez correct identificeert gegeven dat persoon inderdaad Hugo Chavez is (10 / 16)

Def. **PRECISION**: succesratio waarbij algoritme voorspelt dat het Hugo Chavez is en kans dat het ook daadwerkelijk Hugo Chavez is. (10 / 10)

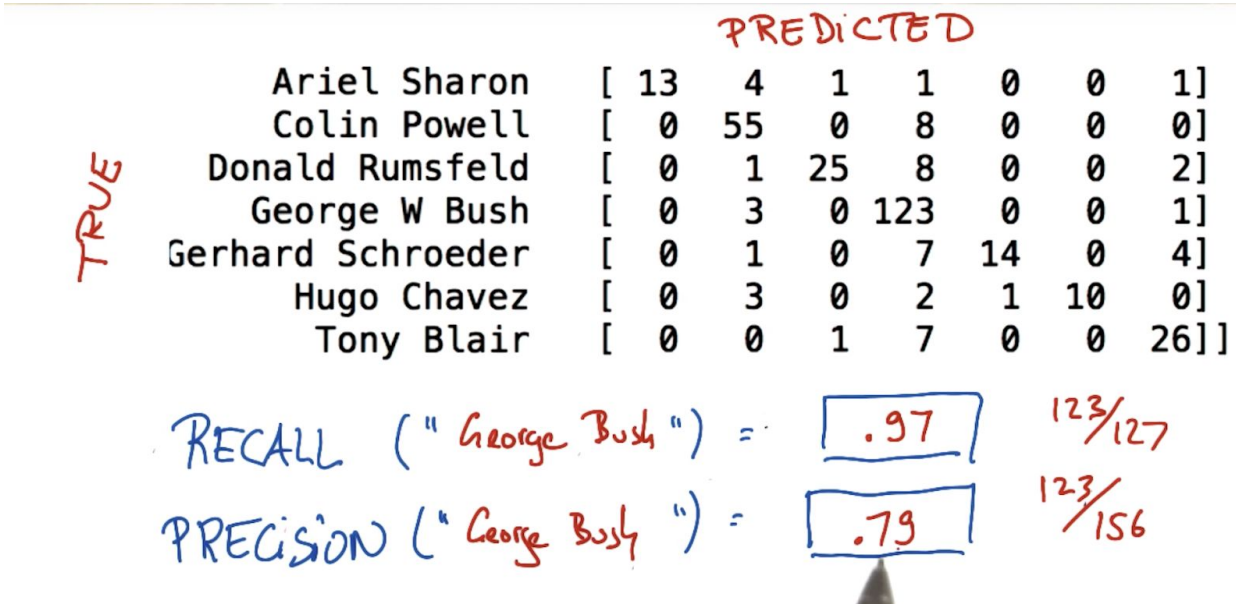
15.17 Colin Powell: recall == 55 / 63 (.873) en precision == 55 / 67 (.820)

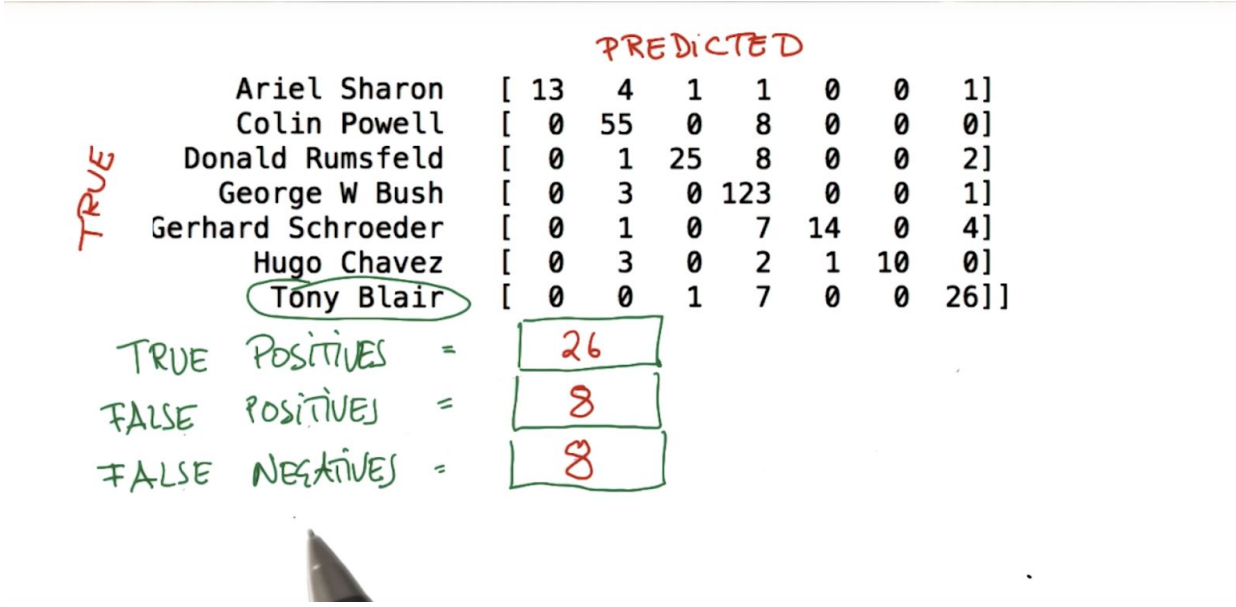


Recall: True Positive / (True Positive + False Negative). Out of all the items that are truly positive, how many were correctly classified as positive. Or simply, how many positive items were 'recalled' from the dataset.

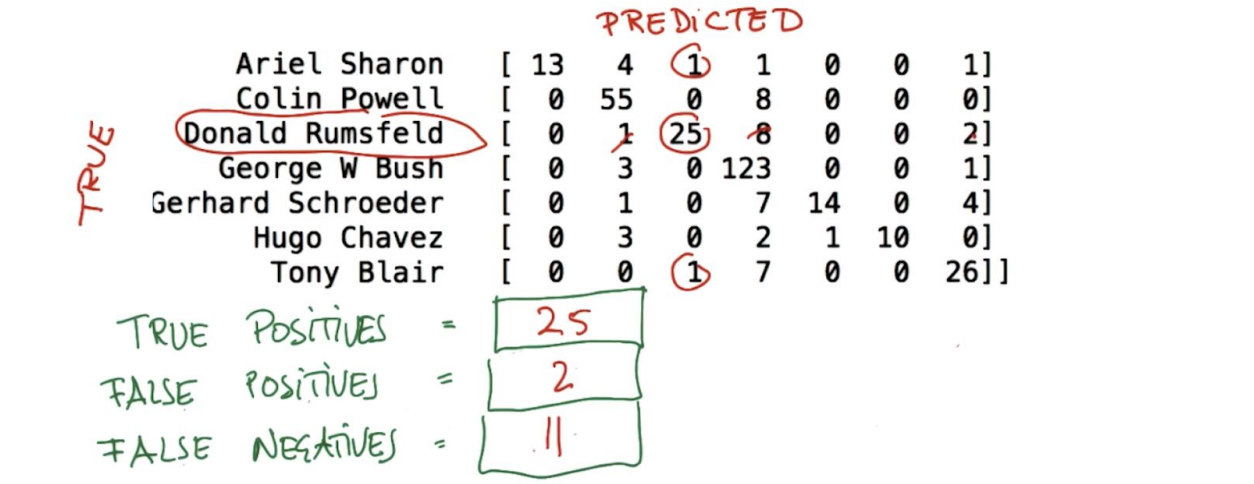
Precision: True Positive / (True Positive + False Positive). Out of all the items labeled as positive, how many truly belong to the positive class.

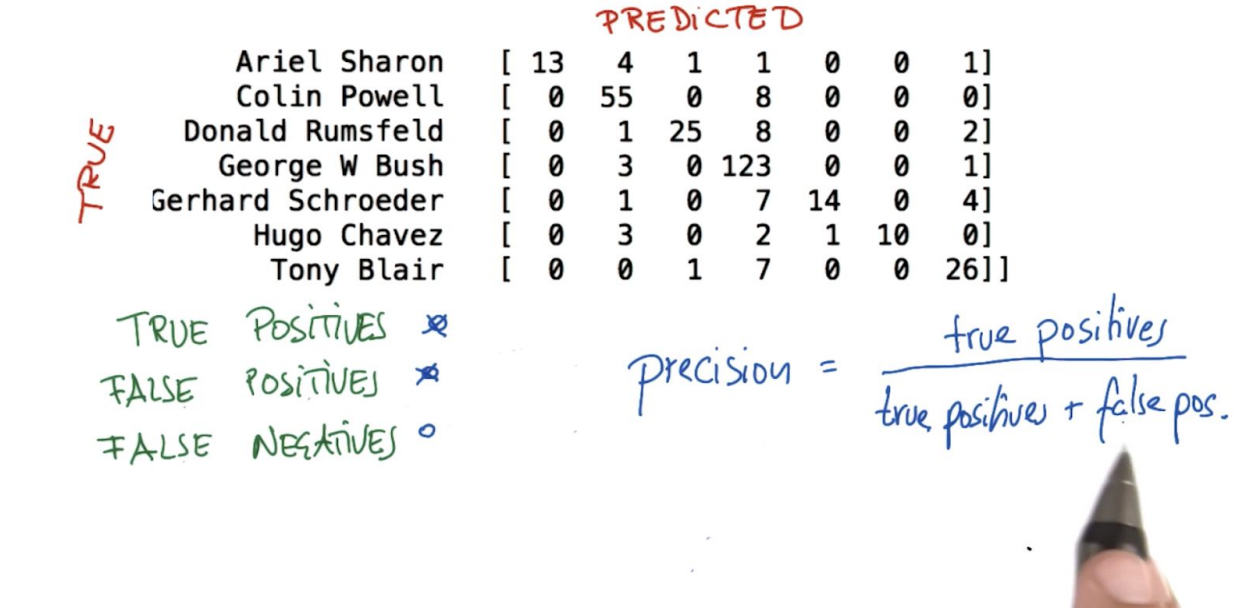
15.18 George W Bush: recall == 123 / 127 (.969) en precision == 123 / 156 (.788)





False positives: predicted incorrectly (iemand anders werd verward met target)
False negatives: recognized incorrectly (target werd niet herkend)





Quiz 15.30

 # 15.30: If your identifier predicted 0. (not POI) for everyone in the test set, what would its accuracy be?
labels_test = []
for i in range(int(sum(pred))):
*   labels_test.append(0.0)*
accuracy = clf.score(features_test, labels_test) # Accuracy is now 0.862

Quiz 15.33 en Quiz 15.34

```
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score

precision = precision_score(labels_test, pred)
recall = recall_score(labels_test, pred)
#15.33
print ("Precision score: " + str(precision))
#15.34
print ("Recall score: " + str(recall))



predictions = [0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1]
true_labels = [0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0]

#15.35
true_negatives = [(x,y) for x, y in zip(predictions, true_labels) if x == y and x == 0.0]
print ("True negatives (Q15.35): "+str(len(true_negatives)))

#15.36
false_positives = [(x,y) for x, y in zip(predictions, true_labels) if x != y and y == 0.0]
print ("False positives (Q15.36): "+str(len(false_positives)))

#15.37
false_negatives = [(x,y) for x, y in zip(predictions, true_labels) if x != y and y == 1.0]
print ("False negatives (Q15.37): "+str(len(false_negatives)))

#15.38
precision_value = len(true_positives) / (len(true_positives) + len(false_positives))
print ("Precision (Q15.38): " + str(precision_value))

#15.39 Recall (True Positive / (True Positive + False Negative))
recall_value = len(true_positives) / (len(true_positives) + len(false_negatives))
print ("Recall (Q15.39): " + str(recall_value))
```

My identifier doesn't have great _____, but it does have good _____. That means that, nearly every time a POI shows up in my test set, I am able to identify him or her. The cost of this is that I sometimes get some false positives, where non-POIs get flagged.

- ○ recall/precision
- ● precision/recall
- ○ F1 score/recall
- ○ precision/F1 score

My identifier has a really great _____. This is the best of both worlds. Both my false positive and false negative rates are _____, which means that I can identify POI's reliably and accurately. If my identifier finds a POI then the person is almost certainly a POI, and if the identifier does not flag someone, then they are almost certainly not a POI.

- ○ F1 score/high
- ○ recall/low
- ● F1 score/low
- ○ precision/low

There's usually a tradeoff between precision and recall--which one do you think is more important in your POI identifier? There's no right or wrong answer, there are good arguments either way, but you should be able to interpret both metrics and articulate which one you find most important and why.

Lession 16: Wrap-up

https://classroom.udacity.com/courses/ud120/lessons/3033138574/concepts/30814187470923

1. Dataset and questioning
2. Feature selection
3. Selecting an algorithm and parameters for it
4. Validation (trustworthiness)