NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz


© 2018 Duke Identity & Diversity Lab.

MSE MachLe
V08: Feature Engineering

Christoph Würsch
Institute for Computational Enineering ICE
Interstaatliche Hochschule für Technik Buchs, FHO

Data, Text, Speech & Sound

Repetition V07: Best Practice (Bias-Variance)

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

1. **Make sure** having a **low bias classifier** before expending the effort to get more data
   a) Take a very flexible / capable / high capacity classifier
      (e.g., SVM with Gaussian kernel; neural network with many hidden units; etc.)
   b) **Increase** the number of **features**
   c) Plot learning curves to survey bias until it becomes low
   d) Measure **generalization performance** using **cross validation**
   e) Use **cross validated grid search** to tune the hyperparameters of the learner
2. **Take** a **low bias algorithm** and **feed** it **tons of data** (ensures low variance) ➔ small test error
3. **Try simpler** algorithms **first** (e.g., naïve Bayes before logistic regression, kNN before SVM); **try different** algorithms
4. **Regularization** combats overfitting by **penalizing** (but still allowing) high **flexibility**
5. **Learning curves** (E_{train} and E_{CV} vs. increasing N) help **diagnosing** problems in terms of **bias** and **variance** ➔ decide what to do next
6. If **more data** is needed: Can be manually **labeled**, artificially **created** (data augmentation) or **bought**
7. Assess **covariate shift** through **2 distinct dev sets**: one resembling training data, one resembling real data

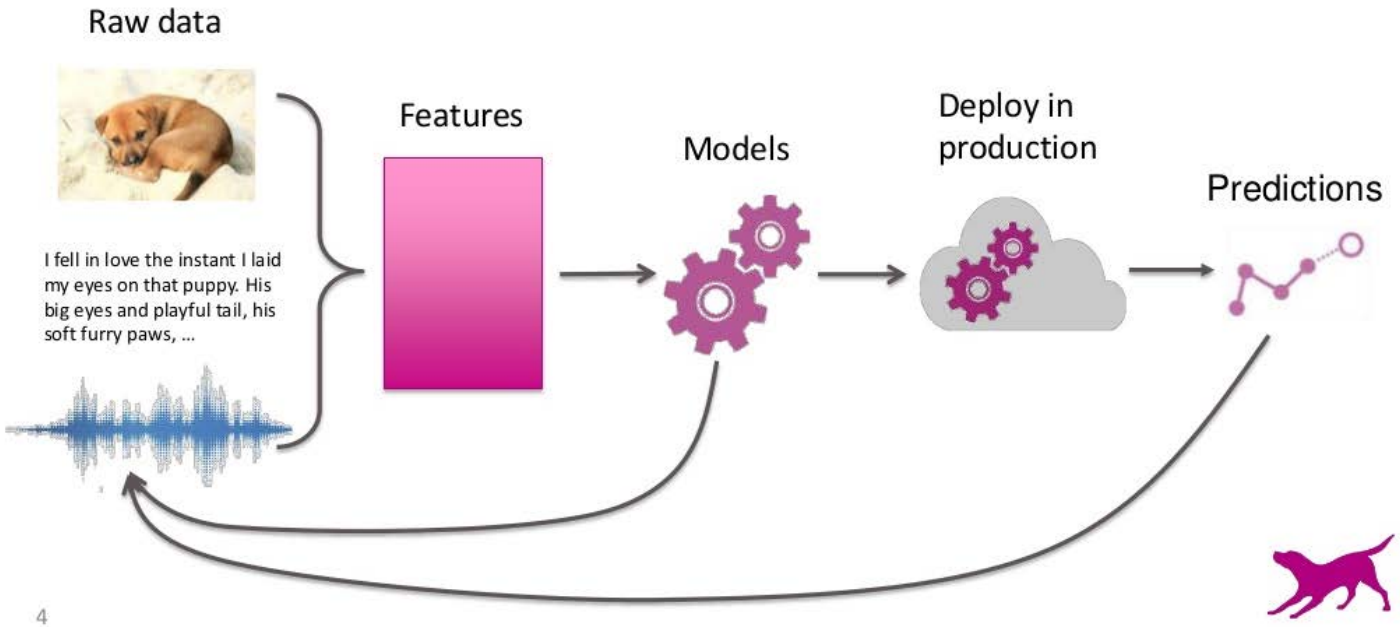
Educational Objectives

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- You use **EDA**, data **preparation** and **cleaning** as necessary steps before starting a ML projekt
- You know how to **generate features** using tansformations (e.g. binning, interaction features)
- You know four approaches for **feature selection** and are able to explain how they work:
  - *Univariate* feature selection (Pearson, F-regression, MIC)
  - *Using linear models and regularization (Lasso)*
  - *Tree-based* feature selection (e.g. using a random forest regressor)
  - *Recursive feature elimination*
- You know how to generate features out of **text data** (stemming, lemmatization, BoW, tf-idf, n-grams, hashing, text2vec)
- You know important features for **audio data: LPC and MFCC**

The Machine Learning Pipeline

- The **machine learning pipeline**



- **Feature** = An individual measurable property of a phenomenon being observed (Christopher Bishop: Pattern Recognition and Machine Learning)

Feature Engineering for Machine Learning, Principles and Techniques for Data Scientists
By Alice Zheng, Amanda Casari, **Publisher:** O'Reilly Media

What is Feature Engineering ?

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- «*Coming up with features is difficult, time-consuming, requires expert knowledge.* ***Applied machine learning is basically feature engineering****.*» (Andrew Ng)
- «*Feature Engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data.*» (Jason Brownlee)
- «***The features you use influence more than everything else the result****. No algorithm alone, to my knowledge, can supplement the information gain given by correct feature engineering.* » (Luca Massaron)

The Data Science Life Cycle

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

Question/Problem Formulation:

- What do we want to know or what problems are we trying to solve?
- What are our hypotheses?
- What are our metrics of success?

Data Acquisition and Cleaning:

- What data do we have and what data do we need?
- How will we collect more data?
- How do we organize the data for analysis?

Exploratory Data Analysis:

- Do we already have relevant data?
- What are the biases, anomalies, or other issues with the data?
- How do we transform the data to enable effective analysis?

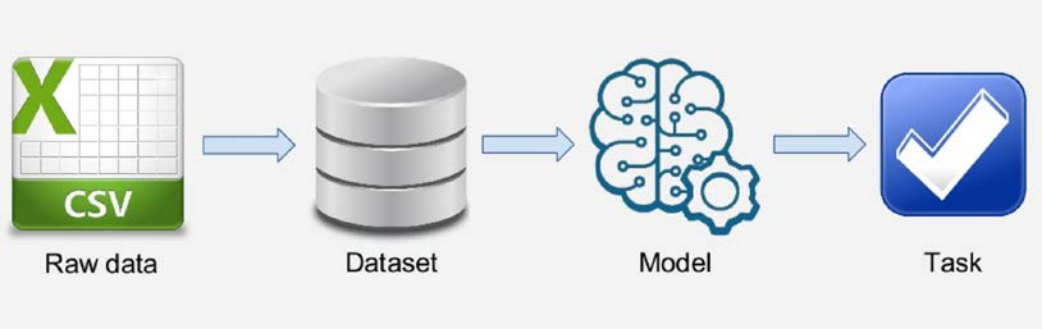
Prediction and Inference:

- What does the data say about the world?
- Does it answer our questions or accurately solve the problem?
- How robust are our conclusions?

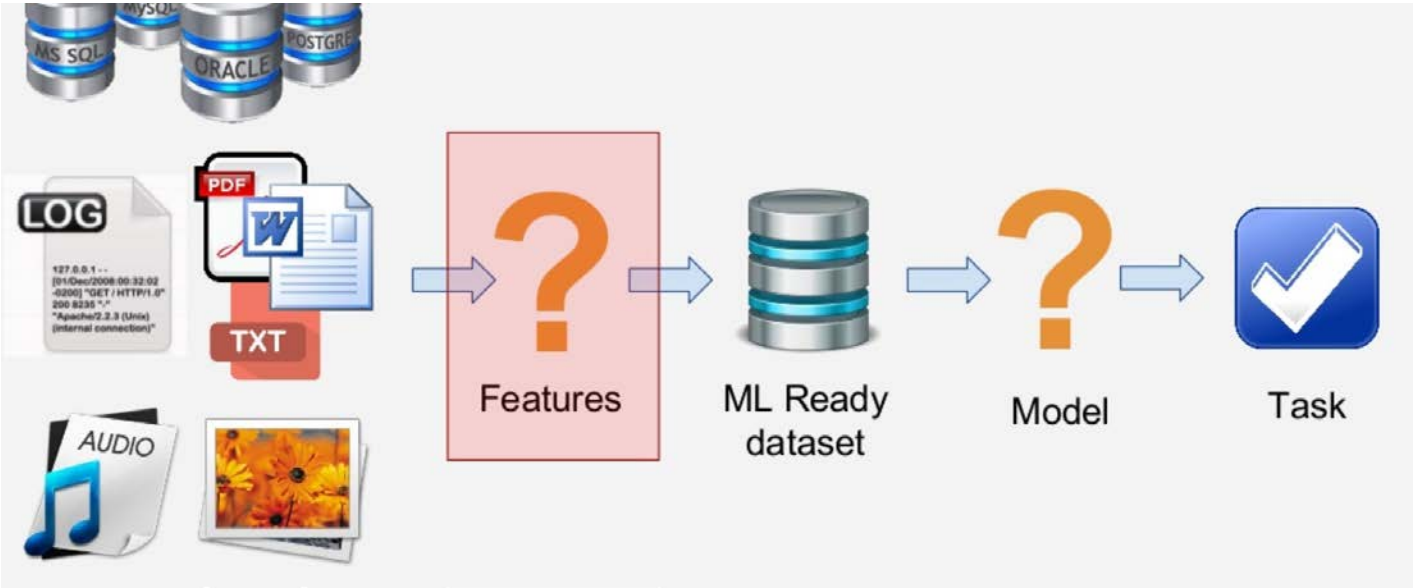
Feature Engineering is hard work

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- The dream:



- Reality:



Feature Engineering - Getting most out of data for predictive models
Gabriel Moreira, Machine Learning Engineer at CI&T

Content

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- **Data Preparation and Exploration (EDA)**
  - Data Cleaning
  - Visualization of data: histogram, scatter-plots, pair-plots, QQ-Plots
- **Feature Generation - Data Transformation**
  - One-Hot-Encoder for categorical variables
  - discretization, binning, quantile binning
  - Scaling (StandardScaler, RobustScaler)
  - Transforming (log, exp, box-cox transform, tf-idf, bag-of-words, cepstrum, power)
- **Feature selection**
  - *Univariate* feature selection (Pearson, F-regression, MIC)
  - *Using linear models and regularization (Lasso)*
  - *Tree-based* feature selection (e.g. using a random forest regressor)
  - *Recursive feature elimination*
- **Outlook: Text, Sound and Images**
  - Text Features, Sound Features, Image Features
- **Automatic Feature generation**
  - After dimensionality reduction (PCA, NMF, k-means)

Exploratory Data Analysis (EDA)

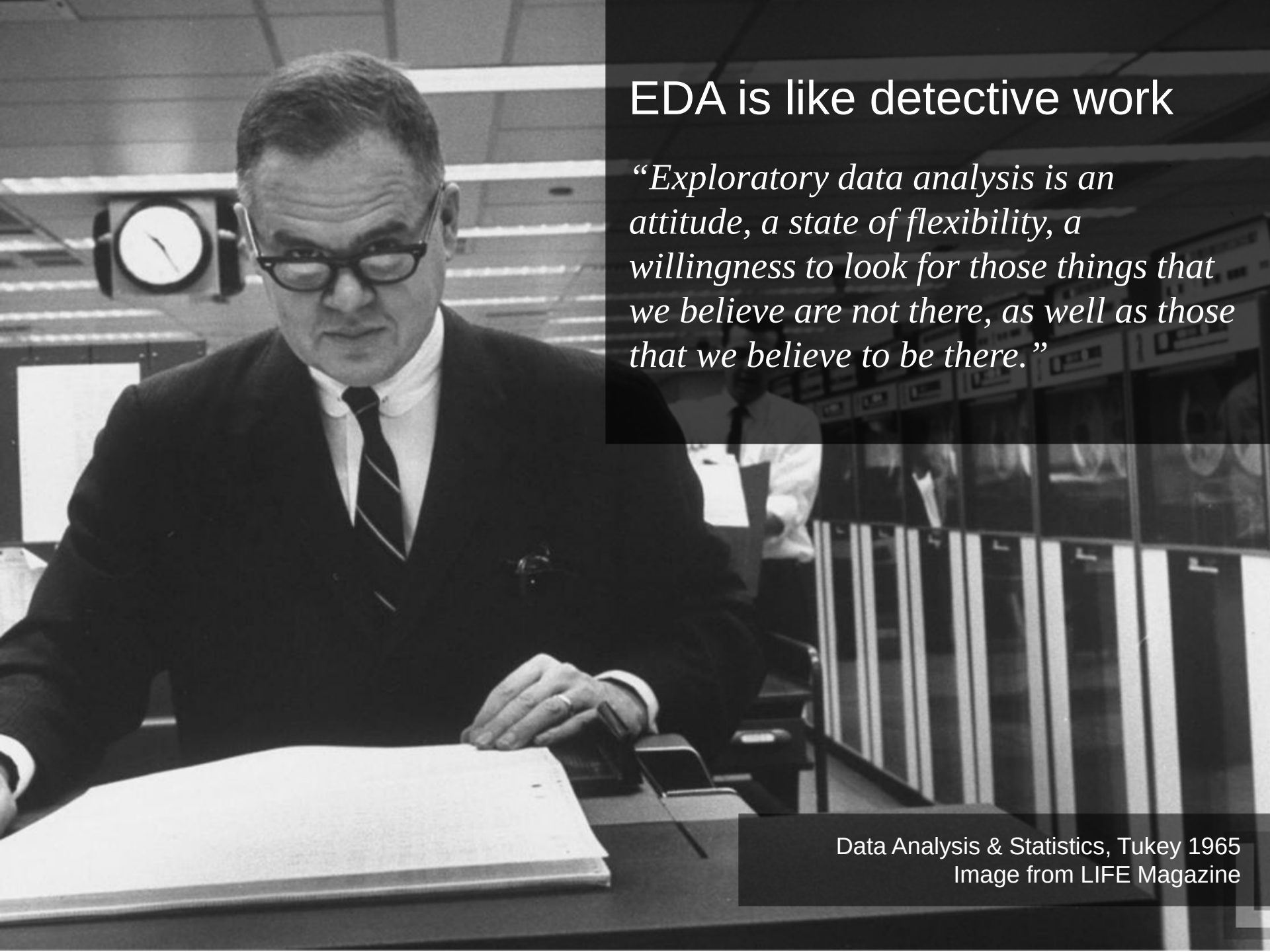
NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

Whether collected by you or obtained from someone else, raw data is seldom ready for immediate analysis. Through ***exploratory data analysis (EDA)*** **we can often:**

- discover important anomalies,
- identify limitations in the collection process,
- and better inform subsequent goal oriented analysis.

Before we start generating features for our models, we have to identify the key properties of data including **structure, granularity, faithfulness, temporality and scope**. We do this by **preparing**, **analyzing** and **visualizing** the data using:

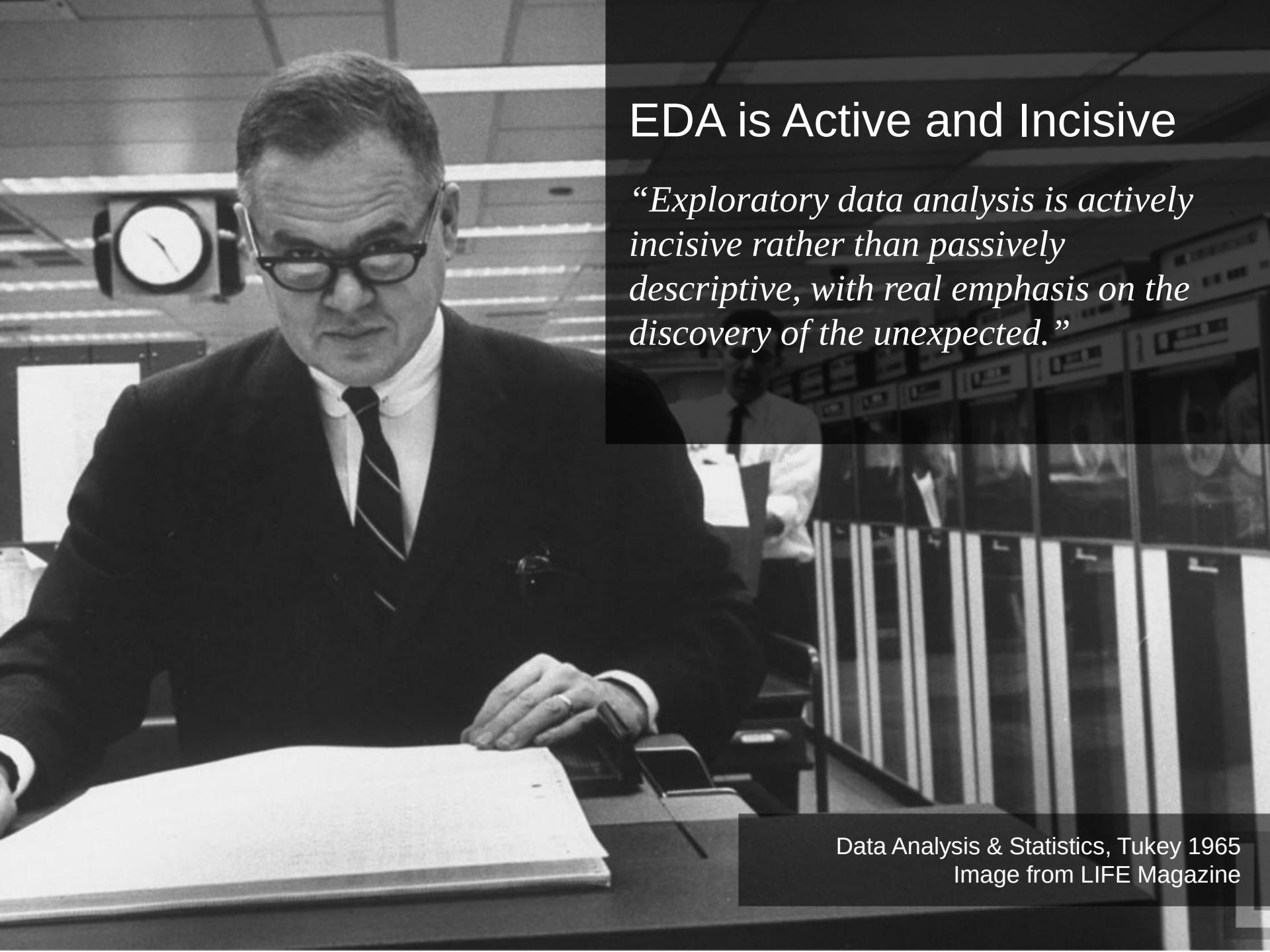
- Histogram, QQ-Plot
- Scatter-Matrix
- Heatmaps



EDA is like detective work

"Exploratory data analysis is an attitude, a state of flexibility, a willingness to look for those things that we believe are not there, as well as those that we believe to be there."

Data Analysis & Statistics, Tukey 1965
Image from LIFE Magazine



EDA is Active and Incisive

“Exploratory data analysis is actively incisive rather than passively descriptive, with real emphasis on the discovery of the unexpected.”

Data Analysis & Statistics, Tukey 1965
Image from LIFE Magazine

GIGO: Garbage In – Garbage Out

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

Heather Hinman (Salford Systems):

> *Do Not Ever ... I mean EVER underestimate the power of good data preparation. THE number one mistake that Modelers make is related to lousy or totally absent data preparation prior to model development. Good data prep includes* ***cleaning****,* ***transforming****, and* ***aggregating*** *model input data as well as the identification and* ***treatment of outliers.***

In other words…

garbage in, garbage out (GIGO)

Feature Preparation | Generation (continous)

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- Data **Cleaning**
  Homogenize missing values and different types of in the same feature, fix input errors, types, etc.
- **Aggregating | Pivoting**
  necessary when the entity to model is an aggregation from the provided data.
- **Imputation** of missing values
  Strategies: mean, median, mode, using a model
- **Binarization**
  transform discrete or continuous numeric features into binary features
- **Binning** (fixed width, adaptive quantile binning)
  Split numerical values into bins and encode with a bin ID
- **Transformation** (eg. Log-Transform, BoxCox)
  Compress the range of large numbers or expand the range of small numbers.
- **Scaling** and **normalization** (min/max, Z-score)
  Scale the numerical variables into a certain range, because models that are smooth functions of input features are sensitive to the scale of the input (e.g. lin. Regression, PCA)
- **Generate Interaction features**

Data Cleansing: unsing pandas (some examples)

https://pandas.pydata.org/pandas-docs/stable/tutorials.html

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

Drop duplicates, drop not availables:

```
df.drop_duplicates()
df.dropna()
```

replace the cells without any data in the column with the value ‘None’

```
df = df.astype(object).where(pd.notnull(df),None)
weekday_counts = df.groupby('weekday').aggregate(sum)
```

replace * in Name with empty string:

```
df[‘Name’]=df[‘Name’].apply(lambda x:str(x).replace(‘*’,’’))
```

Convert dates to **datetime**, keeping only the month

```
df[‘MonthYear’] = pd.to_datetime(df[‘MonthYear’])
df[‘MonthYear’] = df[‘MonthYear’].apply(lambda x: x.date())
```

Pivoting with `pandas.pivot_table`

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

Out[6]:

	regiment	company	TestScore
0	Nighthawks	1st	4
1	Nighthawks	1st	24
2	Nighthawks	2nd	31
3	Nighthawks	2nd	2
4	Dragoons	1st	3
5	Dragoons	1st	4
6	Dragoons	2nd	24
7	Dragoons	2nd	31
8	Scouts	1st	2
9	Scouts	1st	3
10	Scouts	2nd	2
11	Scouts	2nd	3

Create a pivot table of group means, by company and regiment

```
In [7]: pd.pivot_table(df, index=['regiment','company'], aggfunc='mean')
```

Out[7]:

		TestScore
regiment	company	
Dragoons	1st	3.5
	2nd	27.5
Nighthawks	1st	14.0
	2nd	16.5
Scouts	1st	2.5
	2nd	2.5

Imputation: Fill missing values with pandas

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

Create dataframe with missing values

In [6]:
```
raw_data = {'first_name': ['Jason', np.nan, 'Tina', 'Jake', 'Amy'],
        'last_name': ['Miller', np.nan, 'Ali', 'Milner', 'Cooze'],
        'age': [42, np.nan, 36, 24, 73],
        'sex': ['m', np.nan, 'f', 'm', 'f'],
        'preTestScore': [4, np.nan, np.nan, 2, 3],
        'postTestScore': [25, np.nan, np.nan, 62, 70]}
df = pd.DataFrame(raw_data, columns = ['first_name', 'last_name', 'age',
                                       'sex', 'preTestScore', 'postTestScore'])
df
```

Out[6]:

	first_name	last_name	age	sex	preTestScore	postTestScore
0	Jason	Miller	42.0	m	4.0	25.0
1	NaN	NaN	NaN	NaN	NaN	NaN
2	Tina	Ali	36.0	f	NaN	NaN
3	Jake	Milner	24.0	m	2.0	62.0
4	Amy	Cooze	73.0	f	3.0	70.0

Fill in missing in preTestScore with the mean value of preTestScore

inplace=True means that the changes are saved to the df right away

In [7]:
```
df["preTestScore"].fillna(df["preTestScore"].mean(), inplace=True)
df
```

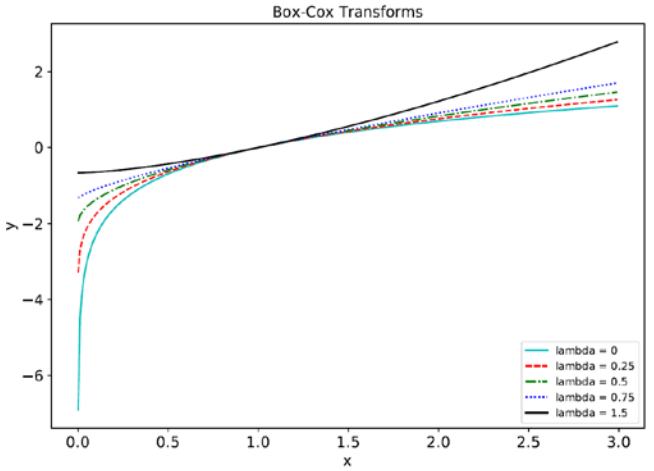
Out[7]:

	first_name	last_name	age	sex	preTestScore	postTestScore
0	Jason	Miller	42.0	m	4.0	25.0
1	NaN	NaN	NaN	NaN	3.0	NaN
2	Tina	Ali	36.0	f	3.0	NaN
3	Jake	Milner	24.0	m	2.0	62.0
4	Amy	Cooze	73.0	f	3.0	70.0

Box-Cox-Transformation

- The **Box-Cox Transformation** can be used to symmetrize positive numerical features (reduce skewness)

$$\tilde{x}_i = \begin{cases} \frac{x_i^\lambda - 1}{\lambda} & \text{if } \lambda \neq 0 \\ \log(x_i) & \text{otherwise} \end{cases}$$



- Generalization: **Yeo-Johnson** transformation (also for neg. data)
- Idea: Choose λ so that the transformed data is as symmetric as possible. Symmetry can be measured with the **skewness** of the data:

$$\gamma_x = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s_x} \right)^3$$

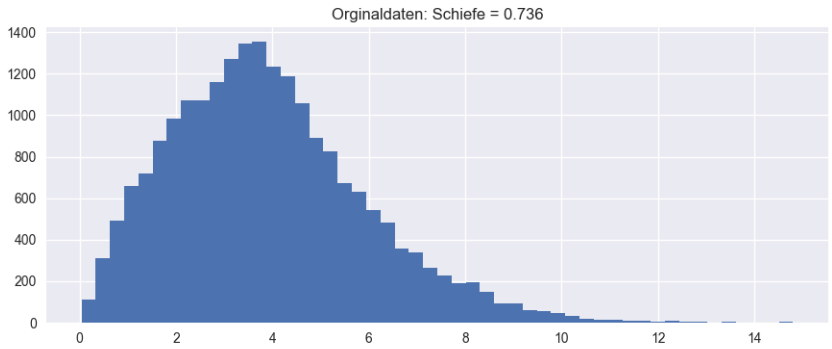
PowerTransformer in sklearn

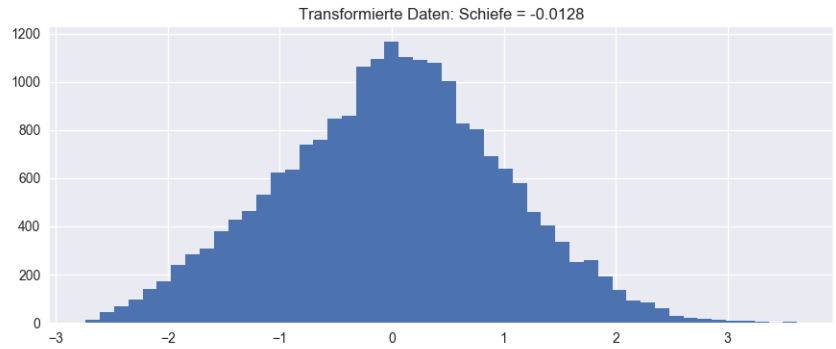
NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- Histogram before and after **PowerTransformer** applied:

```
from sklearn.preprocessing import PowerTransformer

# Select numeric features
weather_num = weather.select_dtypes(include =["float"]) #
PowTrans = PowerTransformer()
# Fit and transform data
weather_num_t = PowTrans.fit_transform(weather_num)
```





Low Carbon London

Feature Preparation | Generation (categorical)

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

One-Hot Encoding (OHE)

```
from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder(handle_unknown='ignore')
X = [['Male', 1], ['Female', 3], ['Female', 2]]
enc.fit(X)
```

Bin counting

```
rnd = np.random.RandomState(42); X = rnd.uniform(-3, 3, size=100)
y = np.sin(X) + rnd.normal(size=len(X)) / 3
X = X.reshape(-1, 1)
enc = KBinsDiscretizer(n_bins=10, encode='onehot')
X_binned = enc.fit_transform(X)
```

Label Encoding

```
le = preprocessing.LabelEncoder()
le.fit(["paris", "paris", "tokyo", "amsterdam"]) >>>list(le.classes_)
['amsterdam', 'paris', 'tokyo']
>>> le.transform(["tokyo", "tokyo", "paris"])
array([2, 2, 1])
```

NLP – Text Analysis: `Vectorization`

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- Text Analysis is a **major application** field for ML algorithms.
- raw data cannot be fed directly to the algorithms as most expect numerical feature vectors with a fixed size rather than the raw text documents with variable length.
  - **tokenizing** strings and giving an integer id for each possible token, for instance by using white-spaces and punctuation as token separators.
  - **counting** the occurrences of tokens in each document.
  - **normalizing** and weighting with diminishing importance tokens that occur in the majority of samples / documents.
- each individual token occurrence frequency (normalized or not) is treated as a feature. The vector of all the token frequencies for a given document is considered a multivariate sample.

`sklearn.feature_extraction.text.CountVectorizer`

Tokenization

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

To analyze a text document, **tokenization** must firstly be performed and groups of word must be obtained.

1. All **common separators, operators, punctuations and non-printable characters are removed**.
2. Then, **stop-words** filtering that aims to filter-out the most frequent words is performed. Samples: “but, perhaps, wonder”—“ama, belki, acaba”.
3. Finally, **stemming and/or lemmatization** is applied to obtain the stem of a word that is morphological root by removing the suffixes that present grammatical or lexical information about the word. In the next slide, we skip the stemming step.

- **NLTK** —Natural Language Toolkit is a suite of open source **Python** modules and data sets supporting research and development in NLP.

```
text = re.sub(r'[\w\-][\w\-\.]+@[\w\-][\w\-\.]+[a-zA-Z]{1,4}', '_EM', text)
text = re.sub(r'\w+:\/\/\S+', r'_U', text)
```

Text Cleaning

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

Raw	Cleaned and split
GALLUP DAILY\nMay 24-26, 2012 \ u2013 Updates daily at 1 p.m. ET; reflects one-day change\ nNo updates Monday, May 28; next update will be Tuesday, May 29.\nObama Approval48%-\nObama Disapproval45%-1\nPRESIDENTIAL ELECTION\nObama47%-\nRomney45%-\ n7-day rolling average\n\n It seems the bump Romney got is over and the president is on his game.	['GALLUP', 'DAILY', 'May', 'u', 'Updates', 'daily', 'pm', 'ET', 'reflects', 'one', 'day', 'change', 'No', 'updates', 'Monday', 'May', 'next', 'update', 'Tuesday', 'May', 'Obama', 'Approval', 'Obama', 'Disapproval', 'PRESIDENTIAL', 'ELECTION', 'Obama', 'Romney', 'day', 'rolling', 'average', 'It', 'seems', 'bump', 'Romney', 'got', 'president', 'game']

- **RE:** the `re` module allows us to use operations over regular expressions, such as substring replacement, e.g. for e-mails and urls.

```
text = re.sub(r'[\w\-][\w\-\.]+@[\w\-][\w\-\.]+[a-zA-Z]{1,4}', '_EM', text)
text = re.sub(r'\w+:\/\/\S+', r'_U', text)
```

Sebastian Raschka, David Julian, John Hearty: Python: Deeper Insights into Machine Learning Leverage benefits of machine learning techniques using Python, Packt, 2016

Stemming and Lemmaztiation: finding word roots

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- **Porter Stemming:** applying simple replacement rules to create word roots by simplifying the range of suffixes, removing suffixes in several passes, with each pass removing a set of suffix types, cleaning up word endings by adding 'e's where needed removing double 'l's.

```
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
stemmer.stem(words)
```

Source Text	Post-lemmatisation
The laughs you two heard were triggered by memories of his own high-flying exits off moving beasts	['The', 'laugh', 'two', 'hear', 'trigger', 'memory', 'high', 'fly', 'exit', 'move', 'beast']

```
from nltk.stem import PorterStemmer, WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
words = lemmatizer.lemmatize(words, pos = 'pos')
```

- Unlike Porter Stemming, **lemmatization** seeks to find actual roots for words. Where a stem does not have to be a real word, a lemma does. Lemmatization also takes on the challenge of reducing synonyms down to their roots (e,g, tome → book).

Word Hashing and the «hash trick»

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- A hash (#) function is a bit of math that maps data to a **fixed size set** of numbers (fixed table).
- We can use a **hash representation** of known words in our **vocabulary**. This avoids the problem of having a very large vocabulary for a large text corpus because we can choose the size of the hash space (size of the vector representation of the document).
- Words are hashed deterministically to the same integer index in the target hash space. A binary score or count can then be used to score the word. **This is called the “hash trick” or “feature hashing“.**
- The challenge is to choose a hash space to accommodate the chosen vocabulary size to minimize the probability of collisions and trade-off sparsity.

Bag-of-Words (BoW) – CountVectorizer

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- The `CountVectorizer` class takes an array of text data, which can be documents or just sentences, and constructs the **bag-of- words** model for us.
- Each index position in the feature vectors shown here corresponds to the integer values that are stored as **dictionary** items in the `CountVectorizer` **vocabulary**.

In [2]:
```
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
count = CountVectorizer()
docs = np.array([
    'The sun is shining',
    'The weather is sweet',
    'The sun is shining and the weather is sweet'])
bag = count.fit_transform(docs)
```

In [3]:
```
print(count.vocabulary_)
```

```
{'the': 5, 'is': 1, 'shining': 2, 'sun': 3, 'and': 0, 'weather': 6, 'sweet': 4}
```

In [4]:
```
print(bag.toarray())
```

```
[[0 1 1 1 0 1 0]
 [0 1 0 0 1 1 1]
 [1 2 1 1 1 2 1]]
```

Note: The `CountVectorizer` class in `scikit-learn` allows us to use different n-gram models via its `ngram_range` parameter. While a 1-gram representation is used by default, we could switch to a 2-gram representation by initializing a new `CountVectorizer` instance with `ngram_range=(2,2)`.

Term frequency inverse document frequency (tf-idf)

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- When we are analyzing text data, we often encounter words that occur across multiple documents from both classes. Those frequently occurring words typically don't contain useful or discriminatory information.
- The **term frequency-inverse document frequency (tf-idf)** can be used to downweigh those frequently occurring words in the feature vectors by scaling by the logarithm of the relative document frequency.

$$\text{tf\_idf}(t,d) = \text{tf}(t,d) \cdot \text{idf}(t,d)$$

$$\text{idf}(t,d) = \log\left[\frac{n_d}{1+\text{df}(d,t)}\right]$$

term frequency-inverse document frequency (tf-idf)

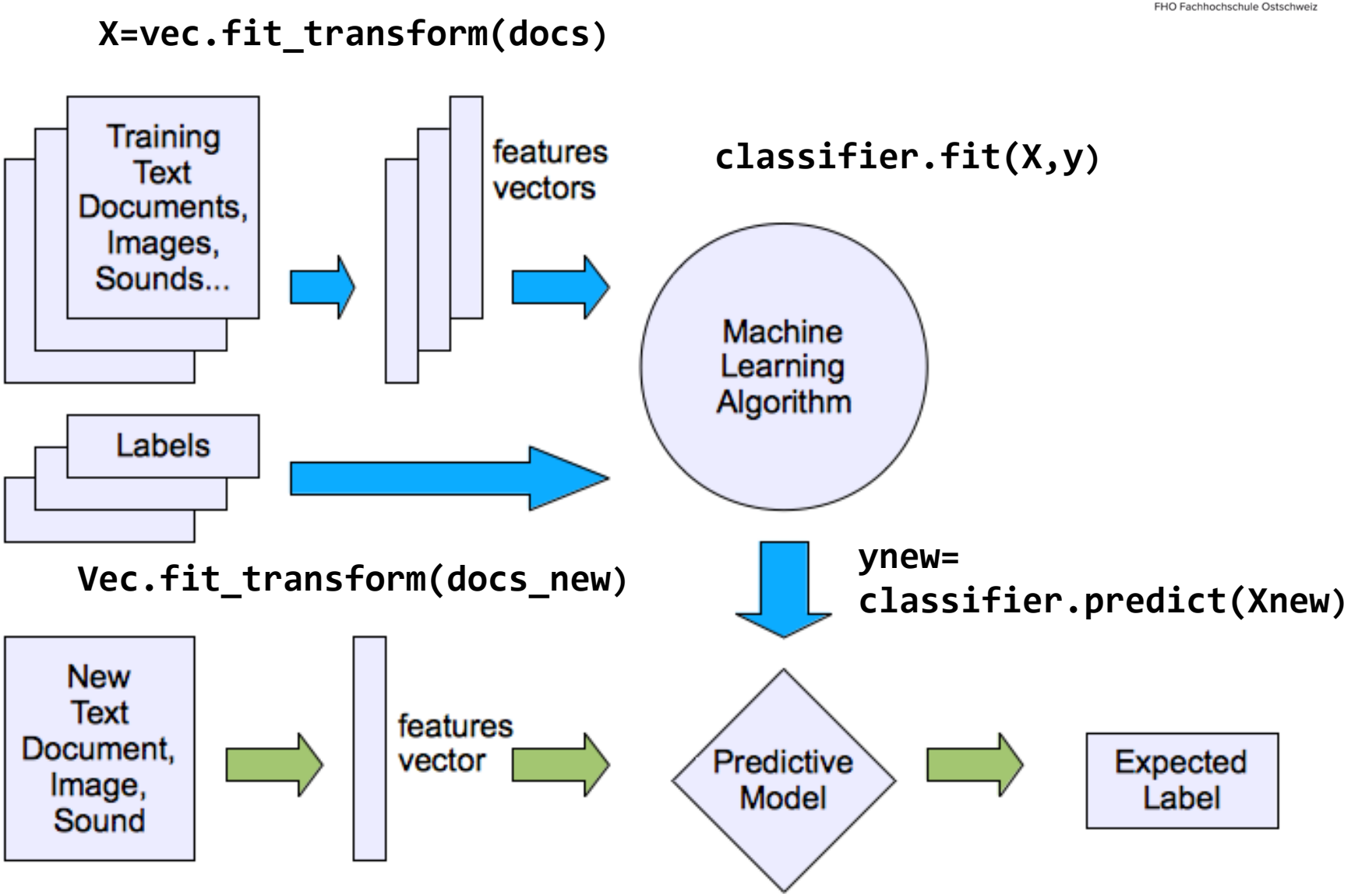
Term frequency: $\text{tf}(t,d)$ # of occurances of term t in all documents d

Document frequency: $\text{df}(t,d)$ # of documents that contain t

n_d # of documents

`Scikit-learn` implements yet another transformer, the `TfidfTransformer`, that takes the raw term frequencies from `CountVectorizer` as input and transforms them into tf-idfs:

Process flow



By Oliver Grisel: https://github.com/ogrisel

Scikit-learn: NLP feature_extraction.

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- Sklearn feature_extraction:

https://scikit-learn.org/stable/modules/feature_extraction.html

`CountVectorizer([…])`	Convert a collection of text documents to a matrix of token counts
`text.HashingVectorizer([…])`	Convert a collection of text documents to a matrix of token occurrences
`text.TfidfTransformer([…])`	Transform a count matrix to a normalized tf or tf-idf representation
`text.TfidfVectorizer([…])`	Convert a collection of raw documents to a matrix of TF-IDF features.

- Natrual Language Toolkit **(NLTK)**

https://www.nltk.org/
http://tinyurl.com/nltk-courses

Feature Selection

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- There are in general two reasons why feature selection is used:
  1. Reducing the number of features, to **reduce overfitting** and **improve the generalization** of models.
  2. To gain a **better understanding** of the features and their relationship to the response variables.
- These two goals are often at odds with each other.
- We will have a look at the following feature selection techniques:
  a) **univariate selection**
  b) **Linear models and regularization**
  c) **Tree based Methods: random forests**
  d) **Recursive feature elimination**

1. Univariate Selection

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- Univariate feature selection works by selecting the best features based on **univariate statistical tests**. Univariate feature selection examines **each feature individually** to determine the strength of the relationship of the feature with the response variable.
- particularly **good for gaining a better understanding** of data but not necessarily for optimizing the feature set for better generalization
- For **regression**: `f_regression`, `mutual_info_regression`
- For **classification**: `chi2`, `f_classif`, `mutual_info_classif`

Note: The methods based on F-test estimate the degree of linear dependency between two random variables. On the other hand, mutual information methods can capture any kind of statistical dependency, but being nonparametric, they require more samples for accurate estimation.

1.1 Pearson Correlation: `[R,p]=pearsonr(X,Y)`

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

$$\rho_{X,y} = \frac{\mathrm{COV}[X,Y]}{\sigma_X \cdot \sigma_Y} = \frac{E[X-\mu_X] \cdot E[Y-\mu_Y]}{\sigma_X \cdot \sigma_Y}$$

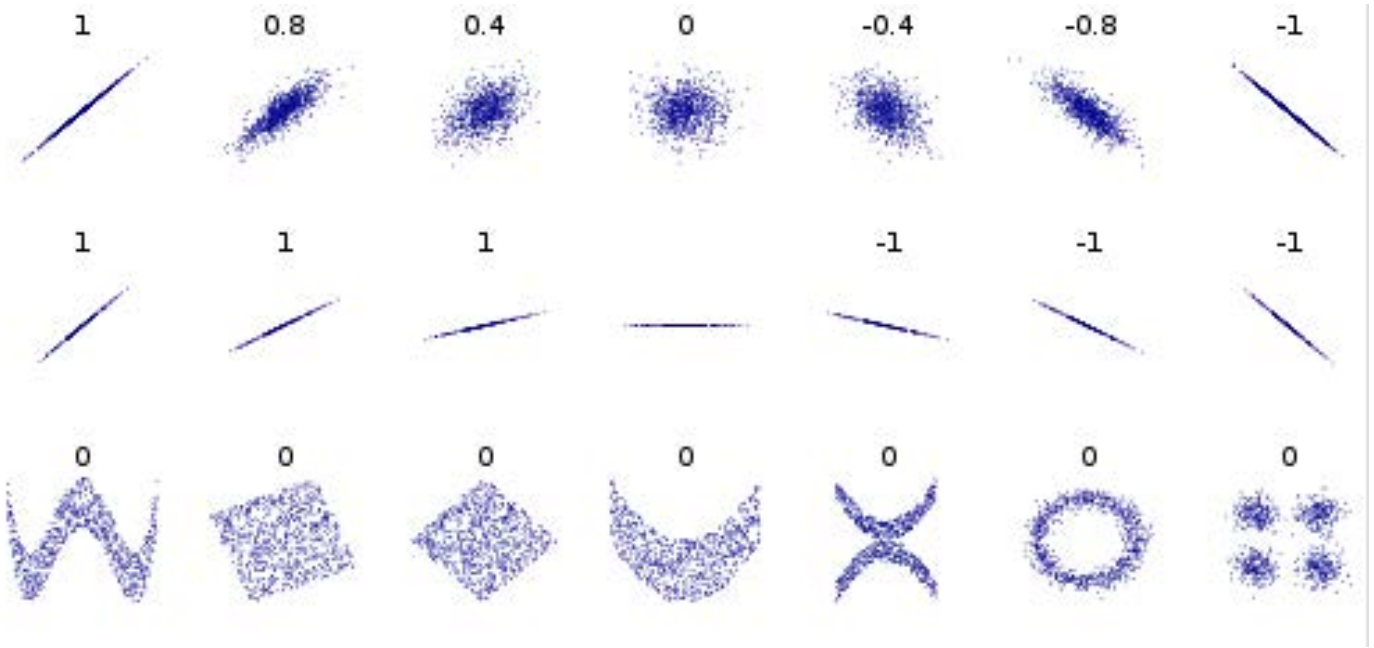
- The **P-value** roughly indicates the probability of an uncorrelated system producing datasets that have a Pearson correlation at least as extreme as the one computed from these datasets.
- drawback of Pearson correlation as a feature ranking mechanism: only sensitive to a ***linear*** relationship

```
1  import numpy as np
2  from scipy.stats import pearsonr
3  np.random.seed(0)
4  size = 300
5  x = np.random.normal(0, 1, size)
6  print "Lower noise", pearsonr(x, x + np.random.normal(0, 1, size))
7  print "Higher noise", pearsonr(x, x + np.random.normal(0, 10, size))
```

```
Lower noise (0.71824836862138386, 7.3240173129992273e-49)
Higher noise (0.057964292079338148, 0.31700993885324746)
```

1.1 Examples

- Pearson Coefficients



- Alternative: F-Regression

1.2 Mutual Information coefficient (MIC)

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- Definition:

$$I(X,Y) = \sum_{\{y \in Y\}} \sum_{\{x \in X\}} p(x,y) \cdot \log\left(\frac{p(x,y)}{p(x) \cdot p(y)}\right)$$

  - can be used for **non-linear dependency**
  - can not be used directly for feature ranking: not a metric and not normalized, so the MI values can be *incomparable* between two datasets.
  - the variables need to be discretized by binning, but the mutual information score can be quite sensitive to bin selection
- Maximal information coefficient is a technique developed to address these shortcomings. It searches for optimal binning and turns mutual information score into a metric that lies in range [0;1].
- In python, MIC is available in the minepy library.

```
x = np.random.uniform(-1, 1, 100000)
print pearsonr(x, x**2)[0]
```

```
-0.00230804707612
```

```
from minepy import MINE
m = MINE()
x = np.random.uniform(-1, 1, 10000)
m.compute_score(x, x**2)
print m.mic()
```

```
1.0
```

1.3 Model Based Feature Ranking

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- one can use an arbitrary machine learning method to build a predictive model for the response variable using each individual feature, and measure the performance of each model.
- e.g. using a **random forest regressor**, for Boston House Dataset
- compare: in case of Pearson coefficient ρ_{XY}, the model is a linear one.

```
from sklearn.cross_validation import cross_val_score, ShuffleSplit
from sklearn.datasets import load_boston
from sklearn.ensemble import RandomForestRegressor

#Load boston housing dataset as an example
boston = load_boston()
X = boston["data"]
Y = boston["target"]
names = boston["feature_names"]

rf = RandomForestRegressor(n_estimators=20, max_depth=4)
scores = []
for i in range(X.shape[1]):
     score = cross_val_score(rf, X[:, i:i+1], Y, scoring="r2",
                             cv=ShuffleSplit(len(X), 3, .3))
     scores.append((round(np.mean(score), 3), names[i]))
print sorted(scores, reverse=True)
```

```
[(0.636, 'LSTAT'), (0.59, 'RM'), (0.472, 'NOX'), (0.369, 'INDUS'), (0.311,
'PTRATIO'), (0.24, 'TAX'), (0.24, 'CRIM'), (0.185, 'RAD'), (0.16, 'ZN'),
(0.087, 'B'), (0.062, 'DIS'), (0.036, 'CHAS'), (0.027, 'AGE')]
```

2. Feature Selection using linear models & regularization

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- Idea: utilize machine learning models for feature ranking: Many machine learning models have some inherent internal ranking of features (e.g. regression models, SVM's, decision trees, random forests)
- This approach can work well even with **simple linear regression models** when the data is **not very noisy** (or there is a lot of data compared to the number of features) and the **features are (relatively) independent**:
- If the features are dependent (multicollinearity), the model becomes unstable, because the decomposition (coefficients) are no longer unique.
- To avoid the **multicollinearity problem**, regularization should be used, either L1 (Lasso) or L2 (Ridge) regularization.

2.1 Lasso feature selection: Boston house prices

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

```
1  from sklearn.linear_model import Lasso
2  from sklearn.preprocessing import StandardScaler
3  from sklearn.datasets import load_boston
4
5  boston = load_boston()
6  scaler = StandardScaler()
7  X = scaler.fit_transform(boston["data"])
8  Y = boston["target"]
9  names = boston["feature_names"]
10
11 lasso = Lasso(alpha=.3)
12 lasso.fit(X, Y)
13
```

```
Lasso model: -3.707 * LSTAT + 2.992 * RM + -1.757 * PTRATIO + -1.081 * DIS +
-0.7 * NOX + 0.631 * B + 0.54 * CHAS + -0.236 * CRIM + 0.081 * ZN + -0.0 *
INDUS + -0.0 * AGE + 0.0 * RAD + -0.0 * TAX
```

- We see that a number of features have coefficient 0. If we increase α further, the solution would be sparser and sparser, i.e. more and more features would have 0 as coefficients.

2.2 Ridge feature selection: Boston house prices

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- Since the coefficients are squared in the penalty expression, it has a different effect from L1-norm, namely **it forces the coefficient values to be spread out more equally**.
- For correlated features, it means that they tend to get similar coefficients.(→ much more stable models)
- While L2 regularization does not perform feature selection the same way as L1 does, it is **more useful for feature "interpretation"**
- Since the relationship between the response variable and features in often non-linear, basis expansion can be used to convert features into a more suitable space, while keeping the simple linear models fully applicable

3. Tree-based Methods for feature selection: (using random forests)

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- **Random forests** are among the most popular machine learning methods thanks to their relatively good accuracy, robustness and ease of use.
- They also provide two straightforward methods for feature selection: **mean decrease impurity** and **mean decrease accuracy**.
- Every node in the decision trees is a condition on a single feature, designed to split the dataset into two so that similar response values end up in the same set.
- The measure based on which the (locally) optimal condition is chosen is called impurity.
- For classification, it is typically either Gini impurity or information gain/entropy (= Kullback Leibler Divergence)
- for regression trees it is variance.

3.1 Information Gain $\mathrm{IG}(T, a)$

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- **Entropy H**: $$H(T) = -\sum_{i=1}^{K} p_i \cdot \log_2 (p_i)$$
- where $p_1, p_2, \ldots, p_K$ are fractions that add up to 1 and represent the percentage of each class $(1 \ldots K)$ present in the child node that results from a split in the tree

- **Information Gain IG**:

$$\mathrm{IG}(T, a) = H(T) - H(T \mid a)$$

$$\mathrm{IG}(T, a) = -\sum_{i=1}^{K} p_i \cdot \log_2 (p_i) - \sum_{a} p(a) \cdot \sum_{i=1}^{K} p(i|a) \cdot \log_2 [p(i|a)]$$

See also: https://en.wikipedia.org/wiki/Decision_tree_learning#Information_gain

Calculating the information gain: Play Golf?

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- **Entropy H:**

Play Golf	
Yes	No
9	5

$$H(\text{Golf}) = H[9,5] = -\left(\frac{9}{14}\right)\log_2\left(\frac{9}{14}\right) - \left(\frac{5}{14}\right)\log_2\left(\frac{5}{14}\right)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

Temperatur	Outlook	Humidit	Windy	Play Golf
hot	overcast	high	false	yes
cool	overcast	normal	true	yes
mild	overcast	high	true	yes
hot	overcast	normal	false	yes
cool	rain	normal	false	yes
mild	rain	normal	false	yes
mild	rain	high	true	no
cool	rain	normal	true	no
mild	rain	high	false	yes
hot	sunny	high	false	no
hot	sunny	high	true	no
mild	sunny	high	false	no
cool	sunny	normal	false	yes
mild	sunny	normal	true	yes

https://www.saedsayad.com/decision_tree.htm

- **Conditional Entropy:**

$$H(\text{Golf} \mid \text{outlook}) = p(\text{sun}) \cdot H[3,2] + p(\text{over}) \cdot H[4,0] + p(\text{rain}) \cdot H[2,3]$$

$$H(\text{Golf} \mid \text{outlook}) = \frac{5}{14} \cdot H[3,2] + \frac{4}{14} \cdot H[4,0] + \frac{5}{14} \cdot H[2,3] \approx 0.693$$

- **Information Gain:**

$$\text{IG}(\text{Golf} \mid \text{outlook}) = H(\text{Golf}) - H(\text{Golf} \mid \text{outlook}) \approx 0.940 - 0.693 = 0.247$$

4. Recursive Feature Elimination (RFE)

- Given an external estimator that assigns weights to features (e.g., the coefficients of a linear model), the goal of recursive feature elimination (RFE) is to *select features by recursively considering smaller and smaller sets of features*.
- First, the estimator is trained on the initial set of features and the importance of each feature is obtained either through a `coef_` attribute or through a `feature_importances_` attribute.
- Then, the least important features are **pruned** from current set of features. That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached.

Example: Random Forest Feature selection

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- Using a random forest regressor:
  ModelBasedFeatureSelection_RandomForest.ipynb

- Using Recursive Elimination of Features (cross validated)
  ModelBasedFeatrueSelection_CV.ipynb

```
sklearn.feature_selection.RFECV
```

Scikit-learn: feature_selection.

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

GenericUnivariateSelect([…])	Univariate feature selector with configurable strategy.
SelectPercentile([…])	Select features according to a percentile of the highest scores.
SelectKBest([score_func, k])	Select features according to the k highest scores.
SelectFpr([score_func, alpha])	Filter: Select the pvalues below alpha based on a FPR test.
SelectFdr([score_func, alpha])	Filter: Select the p-values for an estimated false discovery rate
SelectFromModel(estimator)	Meta-transformer for selecting features based on importance weights.
SelectFwe([score_func, alpha])	Filter: Select the p-values corresponding to Family-wise error rate
RFE(estimator[, …])	Feature ranking with recursive feature elimination.
RFECV(estimator[, step, …])	Feature ranking with recursive feature elimination and cross-validated selection of the best number of features.
VarianceThreshold([threshold])	Feature selector that removes all low-variance features.
chi2(X, y)	Compute chi-squared stats between each non-negative feature and class.
f_classif(X, y)	Compute the ANOVA F-value for the provided sample.
f_regression(X, y[, center])	Univariate linear regression tests.
mutual_info_classif(X, y)	Estimate mutual information for a discrete target variable.
mutual_info_regression(X, y)	Estimate mutual information for a continuous target variable.

5. Automatic Feature Engineering and Selection

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

- Feature engineering steps build on
  - Transformations
  - Aggregations
- **Automated feature engineering** aims to help the data scientist by automatically creating many candidate features out of a dataset from which the best can be selected automatically and used for training.
- **Deep feature** synthesis stacks multiple transformation and aggregation operations (which are called feature primitives in the vocab of featuretools) to create features from data spread across many tables (feature synthesis and subsequent elimination).
- Open Python Library: https://docs.featuretools.com/index.html

Properties of audio signals

Their content and segmentation

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

The sample array $s[n]$ is just 1D

But: Sound still carries **information on** many **different layers** or "dimensions“

Silence ⇔ non-silence

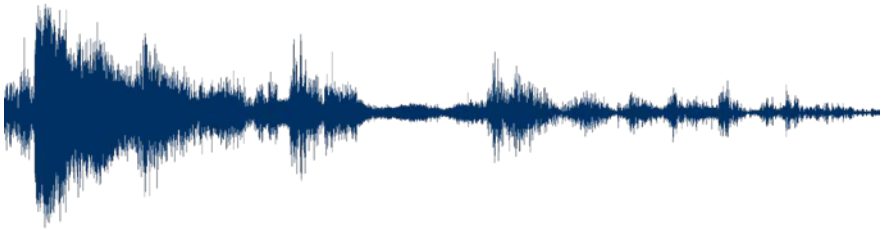
Speech ⇔ music ⇔ noise

Voiced speech ⇔ unvoiced speech

Different musical genres, speakers, dialects, linguistic units, polyphony, emotions, . . .

Definition of audio **segmentation**

- Temporally separate one ore more of the above types from each other into consecutive segments by more or less specialized algorithms



Properties of the speech signal

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

Slowly time-varying

stationary over sufficiently **short period** (5-100ms, phoneme)

Speech range: 100 - 6800Hz (telephone: 300 - 3400Hz)

8kHz sample rate sufficient, 16kHz optimal

Speech frames convey **multiple information**:

Linguistic (phonemes, syllables, words, sentences, phrases, …)

Identity

Gender

Dialect

…

➔ fractal structure

Frame-based processing

From signal to features

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

Feature extraction in general

Reduction in **overall** information

…**while** maintaining or even **emphasizing** the **useful** information

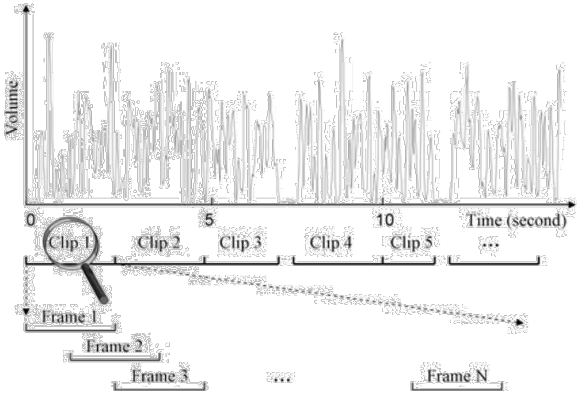
Challenging audio signal properties

Neither stationary (i.e., statistical figures change over time)

➔ **problem** with transformations like Fourier transform **when analyzed in whole**

…nor conveys its meaning in single samples

➔ **problem** when analyzing **per sample**



Source: http://what-when-how.com/video-search-engines/audio-features-audio-processing-video-search-engines/

Solution

Chop into short, usually overlapping chunks called frames

➔ extract features per frame

Prominent parameters: **32ms** frame-**size**, **16ms** frame-**step** (i.e., 50% overlap)

➔ Technically a `double` matrix $f[T][D]$

with $T = 1 + floor\left(\frac{ceil(N - frameSize)}{frameStep}\right)$ the frame count, D the feature dimensionality

Properties of the human auditory system

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

High dynamic range ($120dB$, $q_{dB} = 10 \cdot \log_{10}\left({}^{q}/_{q_{ref}}\right)$ for some quantity q)

Work in the log domain (increase in $3dB$ ➔ loudness doubled)

Performs **short-time spectral analysis** with log-frequency resolution

Similar to wavelet-/Fourier-transform ➔ Mel filter bank

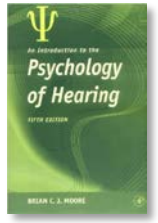
https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html

Masking effects

That's what makes mp3 successful in compressing audio

Channel decomposition via "auditory object recognition"

That's what a machine can not do (except **Melodyne**, and nobody knows why)



…and lots of further interesting material

- But no direct/simple applicability to ASR at the moment

➔ More on the auditory system: Moore, *"An Introduction to the Psychology of Hearing"*, 2004

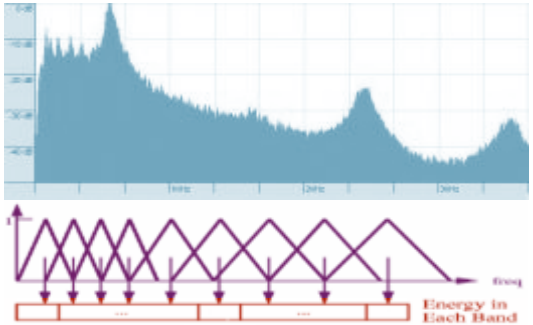
Mel Frequency Cepstral Coefficients (MFCC)

The predominantly used multi-purpose audio feature

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

MFCC extraction process

1. Pre-emphasize: $s[n] = s[n] - \alpha \cdot s[n-1]$
   (**boost high** frequencies to improve SNR; α close to 1, e.g. 0.97)
2. Compute magnitude spectrum: $|FFT(s[n])|$
   (i.e., **time-frequency decomposition** neglecting phase)
3. Accumulate under triangular Mel-scaled filter bank
   (resembles **human ear**)
4. Take DCT of filter bank output, discard all coefficients $> M$
   (i.e., low-pass → **compression**; typically $M \in [8..24]$)



Source: http://developer.nokia.com & http://phys.unsw.edu.au/~jw

Content and meaning of MFCCs

A play with the word “spectrum” and the involved math. operation of convolution

- Low-pass filtered spectrum of a spectrum: “Cepstrum”
- Intuitively: A **compact representation of** a frame’s **smoothed spectral shape**
  ➔ Convey **most** of the **useful information in** a **speech** or **music** signal, but **no** pitch information

Pitch: The perceived tone height (i.e., the tone you would whistle, the melody)

More speech features

Directly from source-filter decomposition

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

Represent source characteristics via **pitch & noise**

1 `double` per frame

Represent filter characteristics with filter coefficients a_k from **LPC analysis**

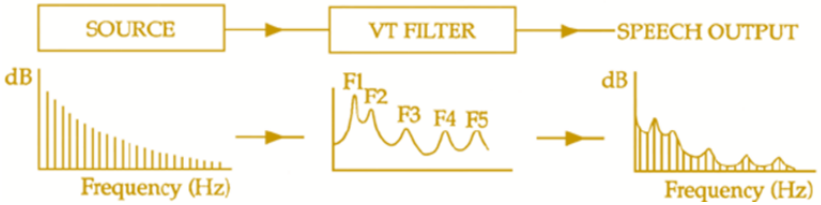
- $8-10$ `double` per frame

$s[n] = \sum_{k=1}^{p} a[k] \cdot s[n-k] + e[n]$ ($e[n]$ being the residual)

Btw.: This is the way it is done in mobile phones

LPC coefficients are also applied as speaker specific features

- Sometimes after further processing
- But **typically**, **MFCCs** are used



Source: Keller, “The Analysis of Voice Quality in Speech Processing”, 2004

GMMs as generative models for voice modeling

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

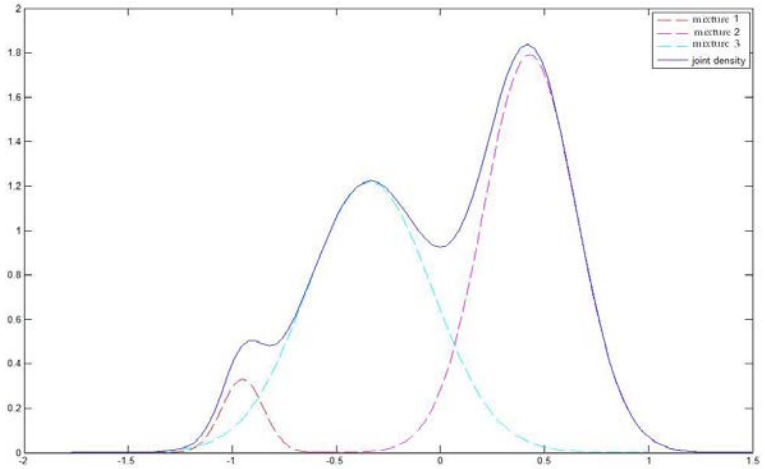
Reference

- Reynolds, Rose, *«Robust Text-Independent Speaker Identification Using Gaussian Mixture Speaker Models»*, 1995

Key ideas

- **Take** the estimated probability density function (**pdf**) $p(x|h)$ of a speaker's D-dim. training vectors x **as** a **model of** his **voice**

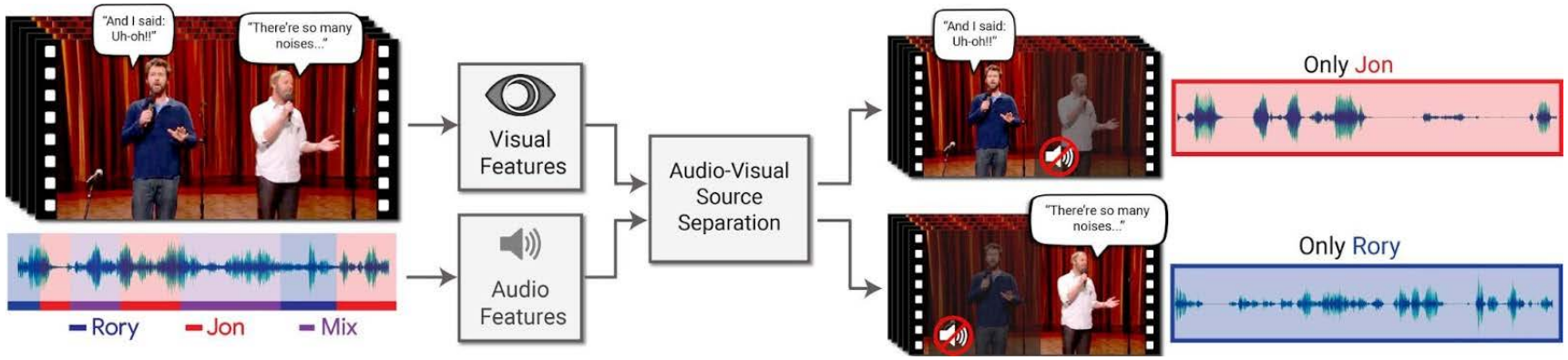
- Model the **pdf as** a **weighted sum of** M D-dimensional **Gaussians** (e.g., $M = 32$, $D = 16$)



GMM with 3 mixtures in 1 dimension. Solid line shows GMM density, dashed lines show constituting Gaussian densities.

Looking to Listen

- «Cocktail Party Problem» soved?
- https://looking-to-listen.github.io/



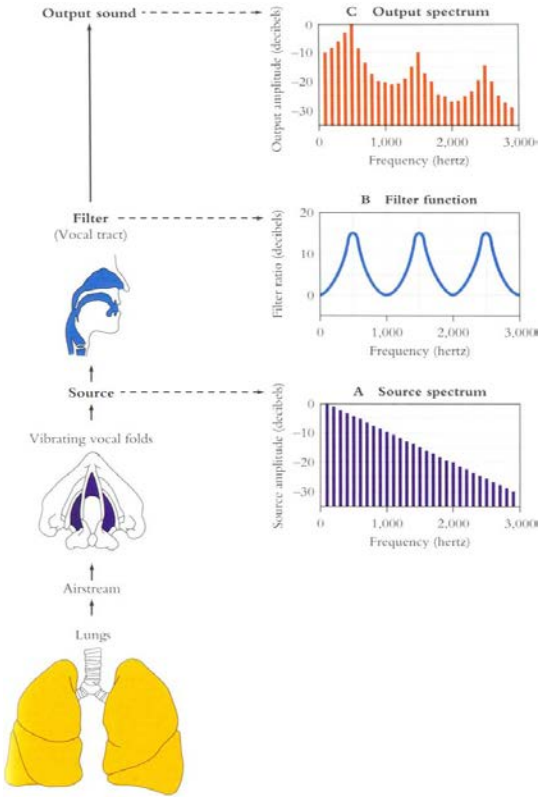
Ariel Ephrat, Inbar Mosseri, Oran Lang, Tali Dekel, Kevin Wilson, Avinatan Hassidim, William T. Freeman and Michael Rubinstein: "Looking to Listen at the Cocktail Party: A Speaker-Independent Audio-Visual Model for Speech Separation", arXiv preprint https://arxiv.org/pdf/1804.03619.pdf

Summary

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

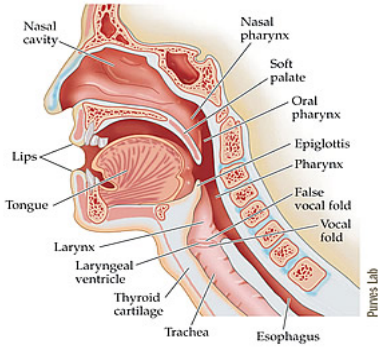
- use **EDA**, data **preparation** and **cleaning** before starting a ML projekt
- **generate features using transformations** (e.g. scaling, encoding, binning, interaction features, …)
- Apply **feature selection:**
  - *Univariate* feature selection (Pearson, F-regression, MIC)
  - *Using linear models and regularization (Lasso)*
  - *Tree-based* feature selection (e.g. using a random forest regressor)
  - *Recursive feature elimination*
- **Text data:** generate features using stemming, lemmatization, BoW, tf-idf, n-grams, hashing, text2vec
- **Audio data: LPC** (linear predicitve coding coefficients) and **MFCC** (Mel Frequency Cepstral Coefficients)

The source filter model of speech production



Source

- Air flows from the lungs through the vocal chords
- Produces noise-like (unvoiced) or
- …periodic (overtone-rich, voiced) excitation signal



The vocal tract; source: DUKE Magazine, Vol. 94, No. 3, 05/06 2008

Filter

- Vocal tract shapes the emitted spectrum

Important physiological parameters

- Size of the glottis determines fundamental frequency (F0) range
- Shape of the vocal tract and nasal cavity determines formant frequencies (F1-5), thus "sound"

Source-filter interaction; source: http://www.spectrum.uni-bielefeld.de/~thies/HTHS_WiSe2005-06/session_05.html

Audio signal features

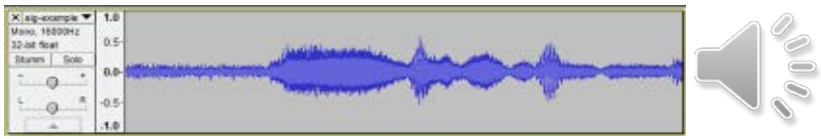
NTB Interstaatliche Hochschule für Technik Buchs

Feature Name	Description
Zero Crossing Rate	The rate of sign-changes of the signal during the duration of a particular frame.
Energy	The sum of squares of the signal values, normalized by the respective frame length.
Entropy of Energy	The entropy of sub-frames' normalized energies. It can be interpreted as a measure of abrupt changes.
Spectral Centroid	The center of gravity of the spectrum.
Spectral Spread	The second central moment of the spectrum.
Spectral Entropy	Entropy of the normalized spectral energies for a set of sub-frames.
Spectral Flux	The squared difference between the normalized magnitudes of the spectra of the two successive frames.
Spectral Rolloff	The frequency below which 90% of the magnitude distribution of the spectrum is concentrated.
MFCCs	Mel Frequency Cepstral Coefficients form a cepstral representation where the frequency bands are not linear but distributed according to the mel-scale.
Chroma Vector	A 12-element representation of the spectral energy where the bins represent the 12 equal-tempered pitch classes of western-type music (semitone spacing).
Chroma Deviation	The standard deviation of the 12 chroma coefficients.

https://python-speech-features.readthedocs.io/en/latest/#
https://github.com/tyiannak/pyAudioAnalysis
https://github.com/tyiannak/pyAudioAnalysis/wiki/3.-Feature-Extraction

The audio signal

The waveform $s[n]$ (a 1D array of N integer samples)



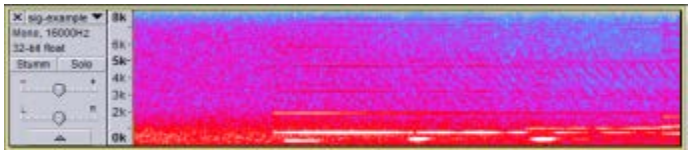
Time domain information (2D: time, amplitude):

Energy (~loudness): $NRG = \frac{1}{N}\sum_n s[n]^2$

Zero crossing rate (~prominent frequency for monophonic signals): $ZCR = \frac{1}{N}\sum_n I(s[n] \cdot s[n-1] < 0)$

Frequency domain information (3D: time, frequency, amplitude):

Time frequency representations via FFT or DWT (phase information typically discarded)



More on signal processing: Smith, *"Digital Signal Processing - A Practical Guide for Engineers and Scientists"*, 2003

Glossary of abbreviations

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

FFT – fast Fourier transform

Standard algorithm to transform a time-domain (time-amplitude) signal into the frequency domain (frequency-amplitude)

DWT – discrete wavelet transform

- Another transformation to the frequency domain, with higher resolution for higher frequencies

DFT – discrete Fourier transform

- The theoretical basis for the FFT algorithm on an array of samples

SNR – signal to noise ratio

- Amplitude of actual signal (what I want to hear) divided by amplitude of any noise (e.g., background music)

DCT – discrete cosine transform

- As DWT, but decomposes the signal solely based on cosine terms (DFT: sine & cosine)

Mel – from the word “melody”

- Unit to measure the pitch of a sound on a scale where an increase in Mel corresponds to the same increase in perceived pitch

SVM – support vector machine

- An often very well-performing supervised machine learning method: give it data (in form of independent feature vectors) of two classes and it learns the discriminative boundary between them

Glossary of abbreviations (contd.)

NTB
Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

LPC – linear predictive coding

- Representing a value of a time series as a linear combination of the last few samples

dB – Dezibel

- Logarithmic unit to express the ratio of two physical quantities, e.g. power or intensity with reference to a "zero" level; a Dezibel is a tenth of a Bel. [after Wikipedia]

mp3 – MPEG-1 or MPEG-2, Audio Layer III

- Lossy audio compression relying heavily on results of psychoacoustics (e.g., masking effects): what can't be heard doesn't need to be coded

ASR – automatic speech recognition

- Joint name for all technologies used to analyze and comprehend human speech with machines

VQ – vector quantization

- A method to represent a set of vectors by a few «representative» vectors (called the «codebook»)

Glossary of abbreviations (contd.)

NTB Interstaatliche Hochschule für Technik Buchs
FHO Fachhochschule Ostschweiz

ATC – audio type classification

BIC – Bayesian information criterion

- Single-value measure to automatically trade-off model complexity and recognition performance

μ – mean vector

- As estimated on a set of vectors

Σ – covariance matrix

- As estimated on a set of vectors; μ and Σ together determine the multivariate Gaussian distribution

δ – delta coefficients vector

- First temporal derivative of some feature, e.g., a MFCC coefficient: $\delta_{d_t} = MFCC_{d_t} - MFCC_{d_{t-1}}$

δδ – delta delta coefficients vector

- Second temporal derivative, i.e. $\delta\delta_{d_t} = \delta_{d_t} - \delta_{d_{t-1}}$ for the d^{th} dimension and t^{th} time step

AANN – Auto-associative neural network (a.k.a. autoencoder)

- Supervised machine learning method that learns to reproduce its input on the output through a sort of bottleneck (e.g., compression) layer