Advanced Machine Learning

Zusammenfassung

Stephan Stofer

12. März 2021

Inhaltsverzeichnis

1 Data Classification **5**
- 1.1 Data Quality Assessment . . . 5
  - 1.1.1 Data Cleaning . . . 5
  - 1.1.2 Approaches to DQA . . . 5
  - 1.1.3 Statistische Kennzahlen . . . 6
- 1.2 Replacement Strategies für NULL Values . . . 7
  - 1.2.1 Feature Engineering . . . 7
  - 1.2.2 Vector Space Model . . . 7
- 1.3 Pandas Profiling . . . 8
- 1.4 Fazit . . . 8

2 History & Development **9**
- 2.1 ML in the Context of AI . . . 9
- 2.2 Development of AI . . . 9
- 2.3 Der Ursprung von AI . . . 9
- 2.4 The Connectionists: ANN's, ML and NN . . . 10
- 2.5 The Human Brain . . . 10
  - 2.5.1 Networks that Learn . . . 10
- 2.6 Deep Learning vs. Machine Learning . . . 10
- 2.7 Three Pillars of Machine Learning . . . 10
- 2.8 Various Options for Hardware Acceleration . . . 10
- 2.9 Performance Measure . . . 10
- 2.10 Remaining Challenges . . . 11

3 Machine Learning Fundamentals **12**
- 3.1 Vector Space Model . . . 12
- 3.2 Data Records . . . 12
- 3.3 Numerical Encoding of Text . . . 13
- 3.4 Distance and Similarity . . . 13
- 3.5 Euclidiean Distance or L^2-Norm . . . 13
- 3.6 Semantik of Similarity . . . 13
- 3.7 Cosine Similarity Intuition . . . 13
- 3.8 Cosine Similarity . . . 14
- 3.9 Euklid vs. Kosinus . . . 14
- 3.10 Manhattan Distance . . . 14
- 3.11 Levenshtein or Edit Distance for Strings . . . 14
- 3.12 Jaccard Similarity for Sets . . . 15
- 3.13 Haversine Distance for GEO Data . . . 15
- 3.14 From Points to Distributions . . . 15
- 3.15 Mahalanobis Distance between Point and Distribution . . . 15
- 3.16 Distance may be sensitive to Scale of Axes . . . 15
- 3.17 Min-Max Normalization . . . 15
- 3.18 Z-Score Normalisierung . . . 16
- 3.19 Normalization Parameters . . . 16

3.20 K-Nearest Neighbors Classification (k-NN) . . . 16
3.21 K-Nearest Neighbors Regression . . . 16
3.22 Hyperparameter . . . 17
3.23 Facts on K-Nearest Neighbors . . . 17

4 Data Preparation for Recommender Systems **18**
4.1 Convert Strings to lower-case format . . . 18
4.2 Tokenizing . . . 18
4.3 Lemmatization . . . 18
4.4 Stemming . . . 19
4.5 Data . . . 19

5 Feature Engineering **20**
5.1 Data . . . 20
5.1.1 Tabular Data . . . 20
5.1.2 Time Series Data . . . 20
5.1.3 Image Data . . . 21
5.1.4 Text Data . . . 21
5.1.5 Feature Engineering vs. Featuree Learning . . . 22
5.2 Feature Engineering for Tabular and Time-Series Data . . . 22
5.2.1 Data Quality Assessment & Data Cleaning . . . 22
5.2.2 Data Imputation . . . 22
5.3 Engineering New Features . . . 22
5.3.1 Grouping . . . 22
5.3.2 Binning . . . 23
5.3.3 De-skewed Data . . . 23
5.3.4 Kernel Trick . . . 23
5.3.5 Expert Knowledge . . . 24
5.3.6 Transform Features . . . 24
5.3.7 Expert Features in Time-Series . . . 24
5.4 Image Data & Computer Vision Applications . . . 24
5.4.1 Edges are an Important Feature . . . 25

Abbildungsverzeichnis

2.1 Entwicklung von Artificial Intelligence 9
2.2 Mehrere Optionen zur Hardwarebeschleunigung 11

3.1 Geometrische Interpretation von Daten 12
3.2 Manhattan Distanz bei Schachbrett-Muster 14
3.3 K-Nearest Neighbors Classification 16
3.4 K-Nearest Neighbors Regression 17

5.1 Feature Engineering 20
5.2 Image Data 21
5.3 Text Feature Extraction 21
5.4 De-skewed Data 23
5.5 Kernel Trick 24
5.6 Spectogram 25

1 Data Classification

Daten werden in zwei Klassen unterteilt. *Numerische* und *Kategorische* Daten. Bei numerische Daten gibt es *stetige* oder *diskrete* Zahlen. Bei Kategorischen sind entweder *ordinal* oder *nominal*. Ordinale haben eine Hierarchie.

1.1 Data Quality Assessment

Daten sind sehr wichtig, der beste ml-Algo nützt nicht, wenn Daten rubbish sind. Mögliche Fehlerquellen:

- Technische Fehler
- Qualität
- schlecht Design
- menschliche Fehler
- Input in Web-Apps (ungeprüfte Eingabefelder)
- Exporte der Daten, falsche Formate - oder Pre-Processing
- Falschangaben durch Benutzer
- Daten haben immer ein Ablaufdaten! (z.B. emailadressen, Adressen)

Ein DQA kommt immer zuerst! Schützt auch die Reputation gegegnüber Kunden.

1.1.1 Data Cleaning

Prozess um Fehler in Daten zu beheben (automatisch)/bereinigen. Duplikate entfernen, null-values entfernen, Datenformate ml-friendly aufbereiten (data wrangling). Die Änderungen müssen dokumentiert und versioniert werden, den data provider darüber informieren und die Ursache für die data quality issues untersuchen.

1.1.2 Approaches to DQA

Dies ist detektiv-Arbeit. Wenn etwas verdächtig erscheint, weitergraben! Die Daten werden überprüft, ob sie vertrauenwürdig sind (plausibilieren).

- Datenquellen und vertrauenwürdigkeit prüfen
- statistische Kennzahlen interpretieren
- daten visualisieren
- Datenranges prüfen (Alter sollte unter 200 sein, Salär > 0, usw.)
- Korrelation zwischen Attributen prüfen (Tachostand und Preis eines Autos)
- Redundanz -> je weniger umso bessere Daten
- Anomalieprüfung in Syntax und Semantik
- NULL Werte und Duplikate erforschen

1.1.3 Statistische Kennzahlen

Geben uns einen Fingerabruck und erste Plausibilisierung der Daten. Die wichtigsten Kennzahlen sind:

- Mittelwert - *mean* - O(n)
- Modus - *mode* die Zahl die am meisten vorkommt
- Median - *median* - O(n * log n), ist aussagekräftiger

1.1.3.1 Schiefheit

Der Mean, Modus und Median geben Auskunft über die Schiefheit der Daten. Wir haben eine negative, Links-Schiefe *skewness* wenn $mean - mode < 0$, wenn positiv, Rechts-Schiefe $mean - mode > 0$

1.1.3.2 Median

Sortiere Datenreihe. Der Median enthält 50% der Daten. Die Quantile entsprechen je 25%. Die Interquartils Differenz (IQR) entspricht Q3 - Q1.

1.1.3.3 Boxplots

Sehr nützlich zur grafischen Darstellung. *Outliers* sind die Werte die grösser sind als $Q3+1.5*IQR$ respektive $Q1-1.5*IQR$. Minimum bzw. Maximum sind die Werte, die gerade noch ind diese Grenze $1.5*IQR$ reinpassen.

Wenn viele Outliers müssen Daten genau angeschaut werden, ob sie trotzdem plausibel sind.

1.1.3.4 Five Number Summary of a Data Distribution

In mit Python kann sehr einfach die $Q1, Q2, Q3$, min und max einer Datenreihe ausgegeben werden:

```
import numpy as np
import pandas as pd

s = pd.Series(np.random.rand(100))
s.describe()
```

Auch Boxplots sind sehr einfach:

```
import matplotlib.pyplot as plt
plt.boxplot(x = [data.Mileage, data.Price], labels=['Mileage', 'Price'])
```

1.1.3.5 Datenverteilung

Die Verteilung wird mit der Varianz betrachtet, wobei diese *sample variance* die Besselkorrektur $(n-1)$ nutzt. Die Standardabweichung entspricht aus der $\sqrt{Var(x)}$

1.1.3.6 Covarianz

Die Covarianz zeigt die Variablität von zwei Datensätzen auf. Ist der Wert positiv, verhalten sich die beiden Daten ähnlich. Ist sie negativ, entsprechend nicht. Ist aber schwierig zu interpretieren, weil sie nicht normiert ist.

1.1.3.7 Covarianzmatrix

Die covarianzmatrix ist sehr wichtig in ML. Sie enthält alle Covarianzen aller Varianzpaare. Die Diagonale kann durch die Varianz von X ersetzt werden.

1.1.3.8 Pearson Korrelation

Covarianz wird durch die Standardabweichung dividiert. Deshalb ergeben sich Werte zwischen 1 (perfekte Korrelation) und -1 (perfect anti-correlation). Damit kann die Datenreihe verglichen werden. Die Korrelationsmatrix kann als Heatmap gut dargestellt werden.

1.2 Replacement Strategies für NULL Values

Kommen immer wieder vor. ML-Algos können selten damit umgehen und müssen bereinigt werden. Je nach Datenumfang sind versch. Verfahren denkbar:

- Zeilen mit NULL Werten löschen
- Fehlende Daten manuell einsetzen
- Globale Konstanten einsetzen (UNKNOWN, $infty$)
- Tendenzen verwenden (Mittelwert für symmetrische Daten, Medien für Schiefedaten)
- Tendenzen auch pro "Klasse" (Eigenschaften) berechnen (z.B Krebskranke und gesunde Patienten)
- Regressionsmodell (sehr aufwändig und ungewohnt in Praxis)

1.2.1 Feature Engineering

Features entsprechen Spalten. Null-Values können also mit ML erzeugen. Information verfügbar für ML-Algo machen.

1.2.2 Vector Space Model

Entspricht einem Datenset welches ausser dem Key nur numerische Werte enthält. Kategorische Daten können sehr einfach in nummersiche Daten transformiert werden. Zum Beispiel werden die Farben alle zu Spalten und entsprechende Zugehörigkeit mit 1 bzw. 0 gekennzeichnet. Diese werden als Dummy-Variable bezeichnet.

```
import pandas as pd
data = pd.read_csv('cars.csv')
data = pd.get_dummies(data)
```

Python code um Daten entsprechend aufzubereiten.

1.2.2.1 Dummy Variable Trap

Mit dem einfügen von Dummy-Variablen muss die *Multikolloniarität* im Auge behalten werden. Wenn n-Dummy Variablen erzeugt werden und $n-1$ Spalten alle 0 sind, wissen wir zu 100, dass die $n$te Spalte 1 sein muss. Dies führt zu unterterminierten Matrizen. Die Matrix kann nicht invertiert werden. Um das zu verhindern, muss eine Spalte gelöscht werden! Es gibt aber Verfahren, die immun dagegen sind (z.B. Entscheidungsbäume).

1.3 Pandas Profiling

Effizient in drei Zeilen Code!

```
import pandas_profiling
data = pd.read_csv('cars.csv‘)
data.profile_report()
```

1.4 Fazit

Bei jedem ML-Projekt ist in Data Quality Assessment Pflicht

2 History & Development

Einführung in die Geschichte und Entwicklung von Machine Learning

2.1 ML in the Context of AI

Machine Learning ist ein Teil von AI, welche normalerweise Menschliche Intelligenz benötigt. Um dies zu erreichen wird ML betrieben. Es lernt ohne explizit progammiert zu werden.

2.2 Development of AI

Mit dieser Disziplin wurde ungefähr 1950 erste Versuche gestartet. Es folgten *Rules Based Systems* und *Logic & Reasoning* zwischen den 1960 bis Anfang 1980.

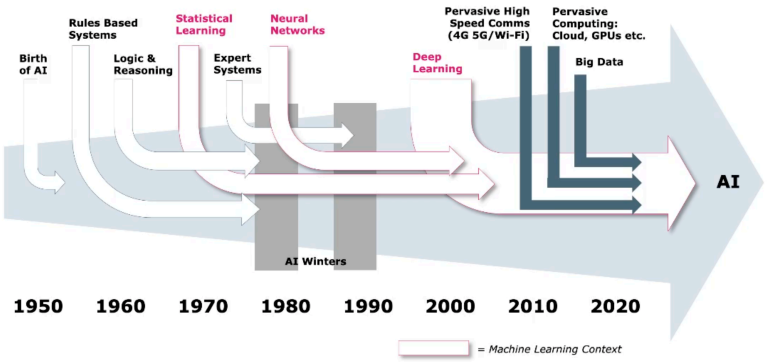

Abbildung 2.1: Entwicklung von Artificial Intelligence

2.3 Der Ursprung von AI

In 1945 erschien erster programmierbarer Computer *ENIAC*. Es entstand eine erste Welle mit viel Forschung und Entwicklung bis Anfang 70er. Die Meinung war, dass Computer alles lösen können. Man merkte aber, dass sie zwar komplexe Aufgaben effizient lösen konnten, jedoch nicht greifen oder etwas erkennen können. Funding wurde immer weniger.

> Es ist einfach einem Computer Intelligenz eines Erwachsenen Menschens beizubringen, jedoch sehr schwierig oder unmöglich ihm die Fähigkeiten eines Einjährigen Kindes, wie Wahrnehmung oder Mobilität, beizubringen.
>
> – Moravec Paradox

Während der zweiten Welle *Expert Systems* wurden vorallem mit bestehenden Wissen gearbeitet und mit den verfügbaren Wissen etwas zu machen. Auch diese Welle wurde durch den zweiten *AI Winter* ausgebremst. > Die Hauptlektion aus 35 Jahre AI-Forschung ist, dass komplexe Probleme sind einfach und die Einfachen sehr schwierig. > > – Steven Pinker

1988 kam die Idee auf von Daten und Erfahrungen zu lernen. Mit dem vermehrte Einsatz von Wahrscheinlichkeiten *Bayesian Networks* kam der Erfolg zurück.

2.4 The Connectionists: ANN's, ML and NN

Artificial Neural Networks (ANN's), Machine Learning und Deep Learning (NN). Wie lernt man überhaupt? Das Gehirn ist sehr flexibel und kann aus verschiedenen Inputs lernen und diese Sinne weiterentwickeln.

2.5 The Human Brain

Das menschliche Gehirn hat verschiedene Areale die für unterschiedliche Funktionen verwantwortlich sind. Es ist massiv vernetzt und verändert sich ständig (durch lernen/Stimulation). Alle Regionen sind aus dem gleichen Neuralen Netz. Jedes Neuron ist mit 1000 bis 10000 anderen verbunden.

2.5.1 Networks that Learn

1887 wurden die Neuronen als Fundamentals Nervensystem entdeckt. 1943 wurde das Neuronenmodel als logische Einheit modelliert. *If* meine Synopsen etwas wahrnehmen (Threshold überschreiten), *then* sende (abfeueren) ich ein Signal durch das Axon, *else* sende ich nichts. Wenn mehrere Neuronen das gleichzeitig feuern, verbinden sie sich - und so entsteht Lernen. 1958 Frank Rosenblatt übertrug die Idee in Computer Vision und erfand das *Perceptron*, ein einzelner Neuralen Netzwork Layer.

2.6 Deep Learning vs. Machine Learning

Machine Learning benötigt menschliches Feature extraction um zu klassifizieren. Deep Learning nutzt grosse neuronale Netzwerke und lernt mit Daten, Algorithmen und Processing Power.

2.7 Three Pillars of Machine Learning

Data, Algorithmen und Porcessing Power. Was natürlich nicht fehlen darf ist Highspeed Kommunikation, welches all die anderen Punkte begünstigt (Concrete).

2.8 Various Options for Hardware Acceleration

2.9 Performance Measure

F1-Score Performance bewertet AI Models auf Basis ihrer Erkennungsrate in Prozent.

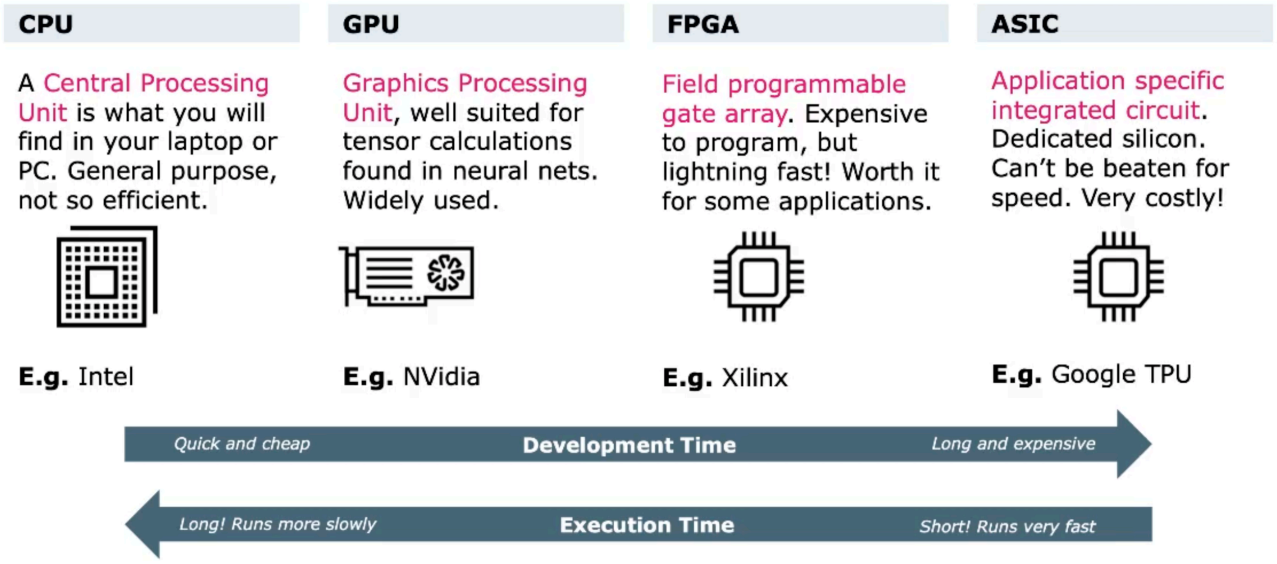

Abbildung 2.2: Mehrere Optionen zur Hardwarebeschleunigung

2.10 Remaining Challenges

- Datenprobleme sind die Grössten Fallstricke in ML-Projekten. 80% des Aufwandes geht auf Datenaufbereitung und lediglich 20% fürs Modelling
- Daten haben Tendenzen, möglichst ausgeglichene Daten verwenden
- ML/DL Training ist sehr Energieintensiv

3 Machine Learning Fundamentals

Grundlagen des ML

3.1 Vector Space Model

Ein Vektor Space Model enthält nur numerische Daten (ausser dem Key). Zum Überführen von kategorischen Daten gibt es mehrere Techniken für die Tranformation. Von nun an gehen wir jeweils davon aus, dass die Daten numerisch vorhanden sind. Der Key wird normalerweise nicht zu den Daten gezählt.

3.2 Data Records

Beinahe alle ML-Verfahren setzen ein *VSM* voraus. Wenn alle Daten als nummersiche Werte voriegen, kann eine jede Zelle als Spalte interpretiert werden.

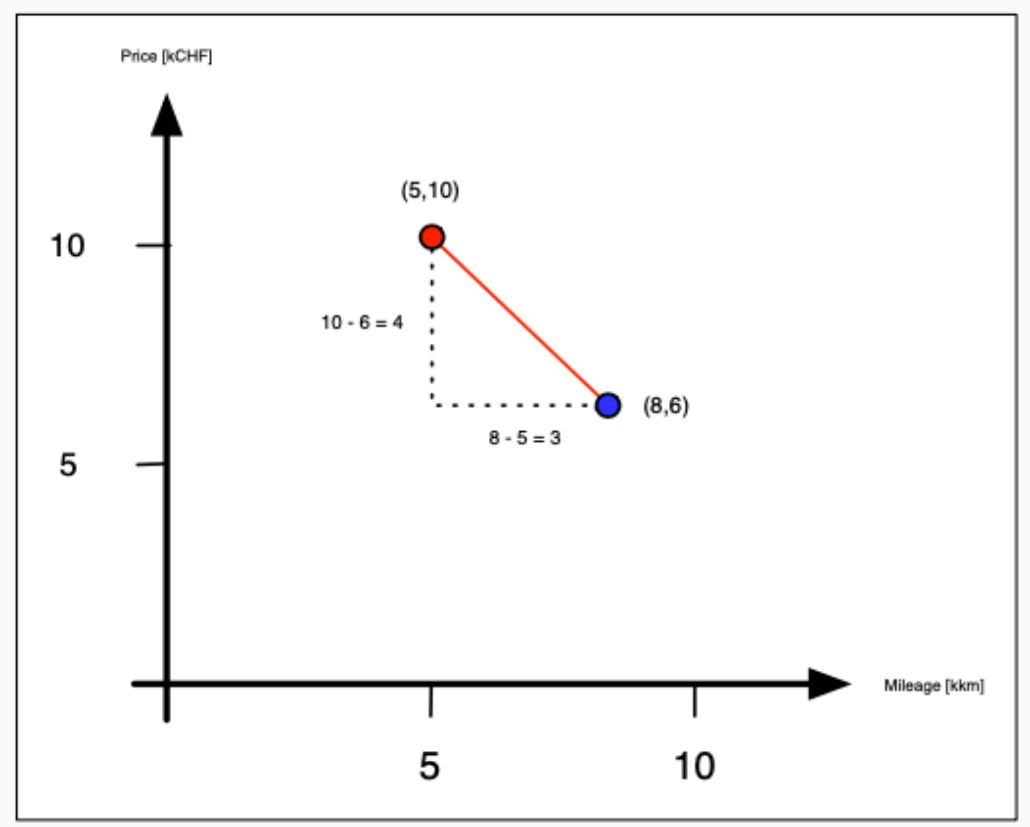

Abbildung 3.1: Geometrische Interpretation von Daten

Die Länge der roten Linie kann mit Hilfe von $\sqrt{a^2 + b^2}$ berechnet werden und als Distanz interpretieret werden.

3.3 Numerical Encoding of Text

Kann man auch ganze Texte oder Bilder in ein Vector Space Model überführen? Bei Bilder können mit Hilfe von Filter transformiert werden. Bei Text funktioniert es mit Hilfe des **TF-IDF Scores** - *term frequency-inverse document frequency*. Diese verbinden Wörter zu numerischen *Vektoren*. Als Basis dient dazu einen Referenz-*Korpus*. Dazu nehmen man alle Wikipediaartikel und stellt diese in Kolonnen dar. Jedes Wort wird nun gemessen anhand der Häufigkeit im ersten Artikel. Danach dividiert durch das Reziproke vom Gesamtwert dieses Wortes über alle Artikel.

3.4 Distance and Similarity

Das Inverse der Distanz ist die Similarität. Je weiter voneinander entfernt, desto unterschiedlicher, respk. je näher umso ähnlicher (Similarität). Auf diesem Konzept basieren alle ML-Algorithmen ausserhalb von Neuronalen Netzen und evtl. Entscheidungsbäume. Beispiele für Distanz und Similarität sind:

- Dating-Site empfiehlt anderes Profil mit den ähnlichsten Vorstellungen
- Auto-Verkaufseite schlägt Preis für Auto anhand der 20 letzten Verkäufe dieses Modell vor
- Webshop empfiehlt Produkte anhand ähnlicher Warenkörbe anderer Benutzer

Wir brauchen also einen Weg die Distanz/Similarität aussagekräftig zu berechnen.

3.5 Euclidiean Distance or L^2-Norm

Ist eine generalisierte Form der Pythagoras Formel welche für eine beliebig grosse Anzahl an Dimensionen verwendet werden kann. Ist eine Zahl $[0, \infty[$ und findet den ähnlichsten Punkt durch die *Minimierung der Distanz* zwischen zwei Punkten. Die Euklidische Distanz wird auch L^2-Norm genannt.

$$euclid(X, Y) = \|X - Y\|_2 = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} = \sqrt{\sum_{i=1}^{2} x_i^2 - 2x_i y_i + y_i^2}$$

3.6 Semantik of Similarity

Je nach Daten ergibt die Distanz nicht exakt wider, wie wir das erwarten würden. Bei unterschiedlichen Anwendungszwecke genügt die Similarität nicht den Ansprüchen. So würde die Distanz bei einem Text der Copy-Pasted wurde weit auseinander liegen (obwohl gleicher Inhalt). Wir müssen also semantisch sinvoll die richtige Methode auswählen. Die Distanzfunktion muss auf die Domäne angepasst werden.

3.7 Cosine Similarity Intuition

Die Kosinus Similarität misst den Winkel θ zwischen zwei Vektoren X und Y. Zwei Punkte auf einer Geraden haben den Winkel 0, aber hätten einen grosse *Euklidische Similarität.*

Die Similarität kann bei vielen Algorithmen per Parameter übergeben werden.

3.8 Cosine Similarity

Durch das Normieren $\frac{X}{\|X\|_2}$ der Vektoren werden die Punkte auf dem Einheitskreis abgebildet. Das Skalarprodukt entspricht der Projektion eines Vektors Y auf den Vektor X. Der Kosinus liegt zwischen $[1, 1]$, für die Distanz verwenden wir aber ein positiven Wert, weshalb wir den Betrag verwenden, wird Kosinus Similarität. 0 bedeutet kleinste Distanz - maximale Similarität und 1 grösste Distanz - Dissimilarität.

$$\text{Kosinusdistanz} = 1 - \text{Kosinus Similarität}$$

3.9 Euklid vs. Kosinus

Die beiden Varianten können ganz unterschiedliche Resultate liefern. Werden die Punkte auf den Einheitskreis normiert, ergeben beide Verfahren dasselbe Resultat.

3.10 Manhattan Distance

Auf einem Schachbrett oder auf der Strasse von Manhattan kann nicht die Euklidische Distanz zur Bestimmung der Entfernung verwendet werden, da man keiner Geraden folgen kann. Die Manhattan Distanz ist definiert durch

$$manhattan(X, Y) = \sum i = 1^n \|x_i - y_i\|$$

Der Wertebereich der Distanz liegt zwischen $[0, \infty[$

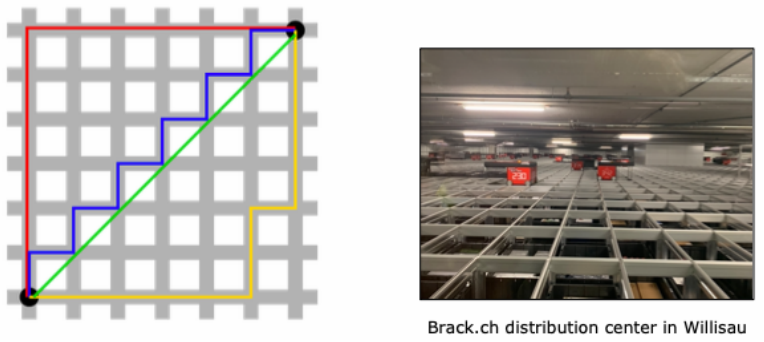

Abbildung 3.2: Manhattan Distanz bei Schachbrett-Muster

3.11 Levenshtein or Edit Distance for Strings

Die Levenshtein Methode zählt die minimale Anzahl an nötigen Operationen an einem Wort um dieses in ein anderes zu überführen. Jede Operation gibt Strafpunkte und werden addiert. Die Summe entspricht der *Levenshtein Distance*. Je grösser umso weniger Similarität zwischen den beiden Worten. Die Operationen werden wie folgt gewertet:

- +1, wenn Buchstabe gelöscht wird
- +1, wenn Buchstabe hinzugefügt wird
- +2, wenn Buchstabe ausgetauscht wird (löschen und hinzufügen) - manchmal wird nur +1 gewertet

Die effiziente Implementierung ist sehr Herausfordernd! Nichts desto trotz, sehr wichtiger und oft verwendeter Algorithmus, zum Beispiel als Wörterbuch. Betrachtet bei Fehler alle Wörter mit kleinster Distanz (Abweichung) und schlägt diese als Korrektur vor.

3.12 Jaccard Similarity for Sets

Betrachtet die Menge einzelnen Wörtere und dividiert sie durch die Gesamte Anzahl an Wörter. Der Wertebereich liegt zwischen $[0, 1]$ und ist definiert durch

$$jaccard(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

Sie kann bei Recommender Systemen angewendet werden. Zum Beispiel wenn zwei Warenkörbe gleiche Produkte enthalten oder wenn zwei Texte gleiches Jargon verwenden.

3.13 Haversine Distance for GEO Data

Flugzeuge reisen nach dieser Distanz, da sie die Krümmung der Erde berücksichtigt. Sie misst in der Atmosphäre die Distanz, ergo ist die Similarität klein, wenn die Distanz gross und vice-versa.

3.14 From Points to Distributions

Manchmal müssen auch Verteilungen beurteilt werden und nicht nur die Punkt zu Punkt Distanz. Es gibt auch die distribution-to-distribution, welche bei der Bildanalyse verwendet wird.

3.15 Mahalanobis Distance between Point and Distribution

Misst die Entfernung des Punkts als wie viele Standardabweichungen der Punkt vom Mittelwert entfernt liegt (Art Euklidische Distanz).

- Erst wird die Verteilung normalisiert indem korrelierende Variablen in unkorrelierende tranformiert werden
- dann wird skaliert, dass die Varianz gleich 1 wird
- zum Schluss wird die euklidsche Distanz zwischen Punkt X und dem Mittelwert berechnet

3.16 Distance may be sensitive to Scale of Axes

Die Einheit von Features können Daten verfälschen (zB. km -> m). Sie dominieren dann das Ergebnis einer anderen Einheit die viel kleiner ausfällt. Deshalb müssen vor dem betreiben von Machine Learning Daten normalisiert werden. Idealerweise haben alle Feature den gleichen Wertebereich.

3.17 Min-Max Normalization

Variante um Daten zu normalisieren und transformiert die Werte ins Intervall $[0, 1]$. Der Grösste Werte in der Spalte erhält den Wert 1, der kleinste der Wert 0. Die Werte dazwischen werden skaliert.

$$x \mapsto \frac{x - \min_X}{\max_X - \min_X}$$

Dies erlaubt die Interpretation in Prozent und man hat keine negativen Werte. Kann aber nicht für *supervised learning* verwendet werden, weil min/max nicht bekannt sind (nach dem Training).

3.18 Z-Score Normalisierung

Die Daten werden so transformiert, dass der Mittelwert 0 ist und die Standardabweichung 1.

$$x \mapsto \frac{x - \mu_X}{\sigma_X}$$

Kann für supervised und unsupervised learning verwendet werden. Der Nachteil ist die fehlende Prozentinterpretation. Kann zu negativen Werten führen (negative Preise oder Anzahl von etwas). Zur Interpretation müsste zurücktransformiert werden.

3.19 Normalization Parameters

Bei Min/Max Normalisation werden (min/max) benötigt, beim Z-Score (mean, std). Wichtig, Parameter aus Trainings-Daten bestimmen (nicht Testdaten), Min/Max nur für supervised learning verwenden (ausser Daten enthalten globale Min/Max). Nomalisierten Parameter speichern um später zu Skalieren.

3.20 K-Nearest Neighbors Classification (k-NN)

Ist wahrscheinlich der einfachste ML-A. Ist ein Verfahren für Regression und Klassifizierung. Zeigt auf, wie wichtig Distanz bzw. Similarität für ML ist.K-Nearest Neighbors muss ein von Hand definierten Parameter k mitgegeben werden. $k = 3$ bedeutet, dass K-Nearest die drei nächsten Punkte suchen und die Klassifikation eines neuen Punkts anhand deren Eigenschaften klassifiziert wird. k-NN mit $k = 1$ wird einfach das Labels des nahesten Punktes übernommen. Bei $k > 1$ wird ein mehrheitsvoting gegenüber den k-nahesten Punkte gemacht.

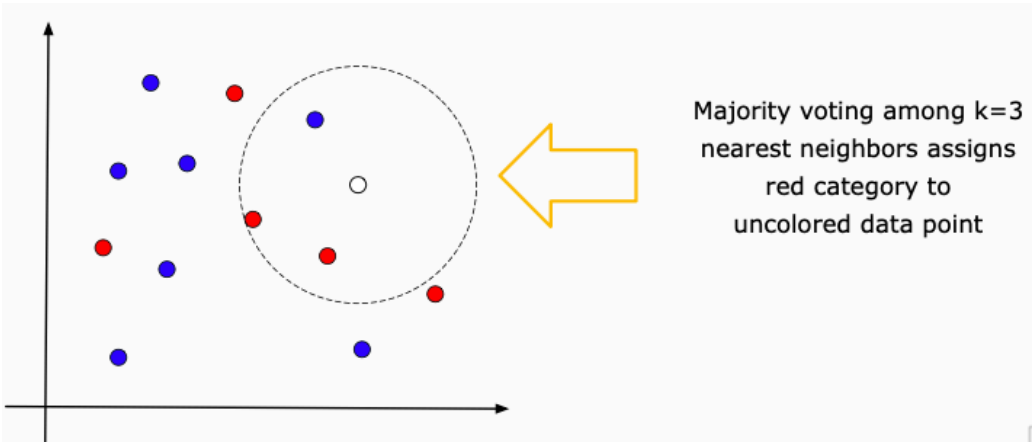

Abbildung 3.3: K-Nearest Neighbors Classification

3.21 K-Nearest Neighbors Regression

Löst Regressionsproblem zum Beispiel Preisvorhersage aus einem Regressionsmodell der k-nahesten Punkte. Funktionsweise mit Parameter k analog k-NN-Klassifizierung.

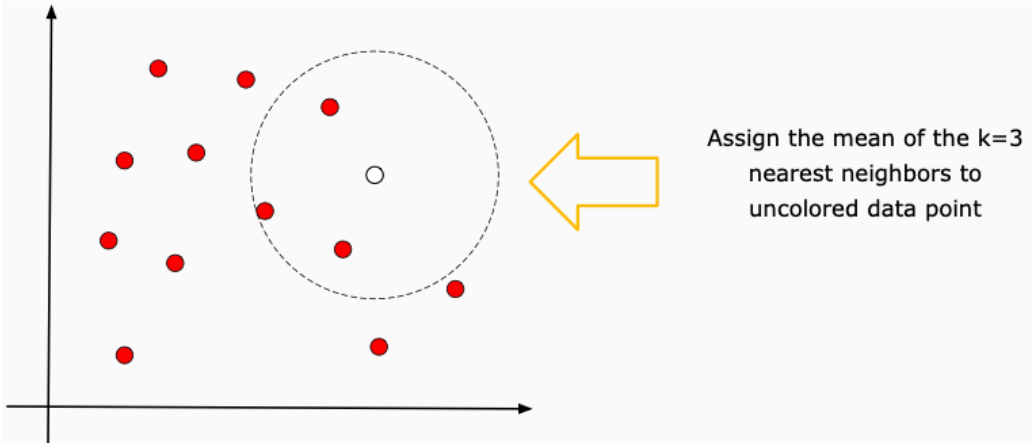

Abbildung 3.4: K-Nearest Neighbors Regression

3.22 Hyperparameter

Der Wert von k wird Hyperparameter genannt und entspricht der Anzahl Nachbarn. Das Resultat kann stark vom gewählten k differieren. Es ist also wichtig dieses möglichst optimal zu wählen (es gibt keine Optimierungsmöglichkeit durch einen Compi). Als weiteren Paramter kann die Distanz/Similaritätswert übergeben werden.

3.23 Facts on K-Nearest Neighbors

- Sehr langsam, weil jedesmal Similarität zu allen Punkten berechnet und dann die Distanz berechnet und nahesten ausgewählt. Dies wird bei jedem Datenpunkt gemacht
- Für kleine Dataset gute Baseline, legt Massstäbe fest für andere Algos
- Mehrheitswahl bedeutet alle Nachbarn sind gleich stimmberechtigt, egal wie weit sie entfernt liegen
- Alternativ kann per Parameter die Distanz berücksichtigt werden $\frac{1}{d}$ mit $d = \text{distance} > 0$
- benötigt am wenigsten Daten, reichen Daten nicht aus, gibt es keine Möglichkeit für ML

4 Data Preparation for Recommender Systems

Um die Daten (strings) verarbeiten zu können müssen diese erst Vorverarbeitet werden

4.1 Convert Strings to lower-case format

Text in lower-case Buchstaben umwandeln damit gleiche Worte immer gleich geschrieben werden.

```
df['description'] = df['description'].str.lower()
```

4.2 Tokenizing

Tokenizing ist ein Verfahren um einen Text zu bereinigen. Das Resultat der Tokenisierung ist eine Liste von Tokens, die als Liste im technischen Sinn, oder als Abfolge von durch Zeilenumbrüche getrennte Tokens. Deren eine Tokenklasse angehängt wird. Während diesem Vorgang werden einige Einzelaufgaben bewältigt. Die Tokens werden auch als *StopWords* bezeichnet.

- Abkürzungen erkennen und isolieren (es gibt auch gleiche Abkürzungen für unterschiedliche Worte)
- Interpunktionen und Sonderzeichen erkennen (Problem diverse Sonderzeichen wie /, @, #, $, usw. gehören oft einem Token an und dürfen nicht isoliert werden)
- kontrahierte Formen expandieren; l'auto ⟶ la auto; gilt das nachher als Artikel oder Pronomen? Führt zu Ambiguität.
- komplexe Tokens erkennen und isolieren; allgemeine Zahlen wie 10 000, Telefonnummer, Datum und Zeit, URLs, Vor- und Nachnamen (was ist mit Titel?)
- ggf. Tokens normalisieren
  - Abkürzungen vereinheitlichen
  - Datums, Zeit- und Massangaben vereinheitlichen
  - Zahlen
- ggf. Tokens klassifizieren (d.h. Tokenklassen bilden); Klassen wie number, date, time, abbr, currency, temp, length usw. bilden
  - Diese Schuhe haben sFr. 147.- gekostet. ⟶ [diese,schuhe,haben,currency(147,sfr,Rp),gekostet,.]
  - Wir treffen uns am 24. April um 15 Uhr. ⟶ [wir,treffen,uns,am,date(24,4,Jr),um,time(15,Min,Sec),.]

4.3 Lemmatization

Die Lemmatisierung ist das Rückführen von Worten in ihre Grundform. So wie sie im Wörterbuch stehen, diese werden als *Lemma* bezeichnet. Das ursprüngliche Wort ist die *Vollform*.

4.4 Stemming

Als Stemming wird die Stammformreduktion oder Normalformenreduktion bezeichnet. Es ist ein Verfahren um verschiedene morphologische Varianten eines Wortes auf ihr gemeinsamen Wortstamm zurückzuführen. Zum Beispiel der **Deklination** von *Wortes* oder *Wörter* zu *Wort* und **Konjugation** von *geseheen* oder *sah* zu *sehen*. Bekanntes Framework ist das `Snowball`von Martin Porter.

4.5 Data

Daraus ergeben sich normalisierte Daten die für das Training und den Test verwendet werden. Das Set ist immer in die zwei Gruppen Test und Training aufzuteilen.

5 Feature Engineering

Ist eine Manipulation an unseren Rohdaten um höhere Performance und bessere Ergebnisse im Kontext von Machine Learnung zu erreichen. Kurz, wir möchten es für die Algorithmen einfacher machen.

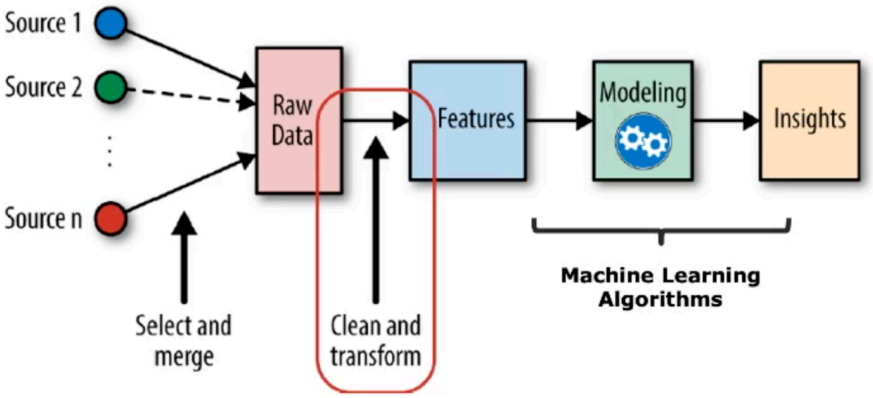

Abbildung 5.1: Feature Engineering

5.1 Data

Als Menschen können wir sehr schnell aus einem Datenset eine Klassifizierung machen, indem wir ihre Eigenschaften miteinander vergleichen. Diese Eigenschaften werden *attributes* oder *feature* genannt. Eigenschaften können sein:

- Farbe
- Form
- usw.

5.1.1 Tabular Data

Meist sind Daten in tabellarisch verfügbar. Jede Zeile entspricht einem Dateneintrag und wird als ein *Datenpunkt (data point)* interpretiert. Die Zeilen enthalten *features* der verschiedenen Datenpunkte. Sofern die Daten nummerisch sind, können wir sie als *multi-dimensionalen Vektoren* in einem *feature space* interpretieren. Die Distanz zwischen zwei Punkten entsprechen einer Similarität (je näher umso ähnlicher).

5.1.2 Time Series Data

Zeitfolgen sind eine Spezialform von tabullarischen Daten welche in einer *chronologischen Sequenz* vorliegen. Eines der Feature ist dabei ein Set aus Zeitstempel. Die Abfolge muss vollständig sein um die Daten korrekt interpretieren zu können. Audiodaten oder Börsenentwicklungen sind gute Beispiele für Zeitfolgen. Diese Sequenzen dürfen nicht gemixt werden, da sie sonst nicht mehr dem Original entsprechen.

5.1.3 Image Data

Auch Bilder sind tabellarische Daten. Ein Schwarweiss-Foto enthält Nummern, welche die *Pixelintensität* repräsentieren im Range von $[0, 255]$ (bei 8bit, 2^8). SW-Bilder haben ein Layer an Pixel, farbige (RGB) haben drei Ebenen. Ein Bild kann auch binär sein, 0 entspricht Schwarz, 1 entspricht weiss.

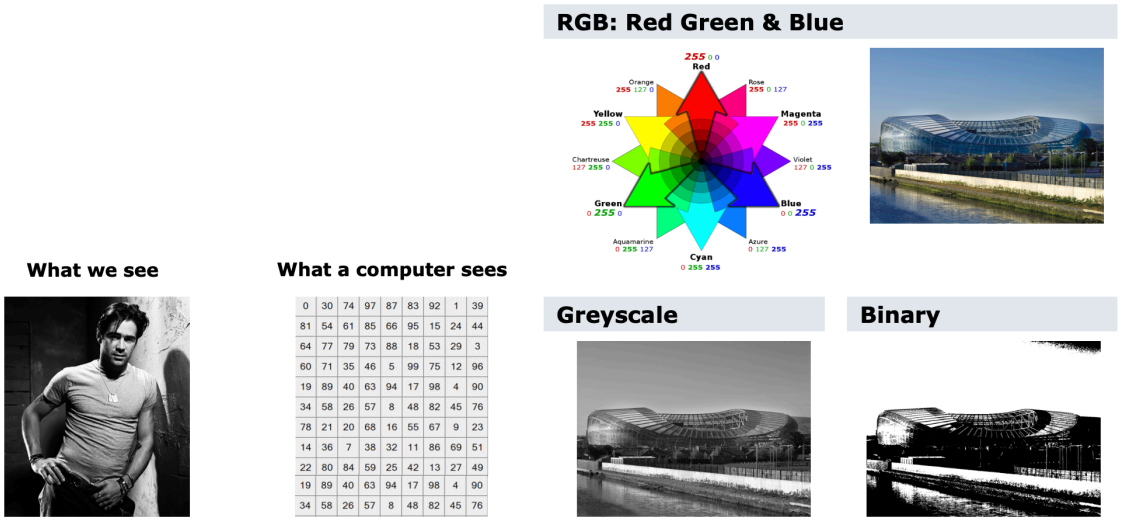

Abbildung 5.2: Image Data

5.1.4 Text Data

Die menschliche Sprache ist eine hohe kognitive Kunst und macht es für Maschinen sehr schwierig von Text zu lernen. Aber Text ist überall und häufig die Kerndaten in eienem ML-Projekt. *Rohe Textdaten* ist oft nicht direkt für ML brauchbar und deshalb idealer Kandidat für *Feature Engineering*.

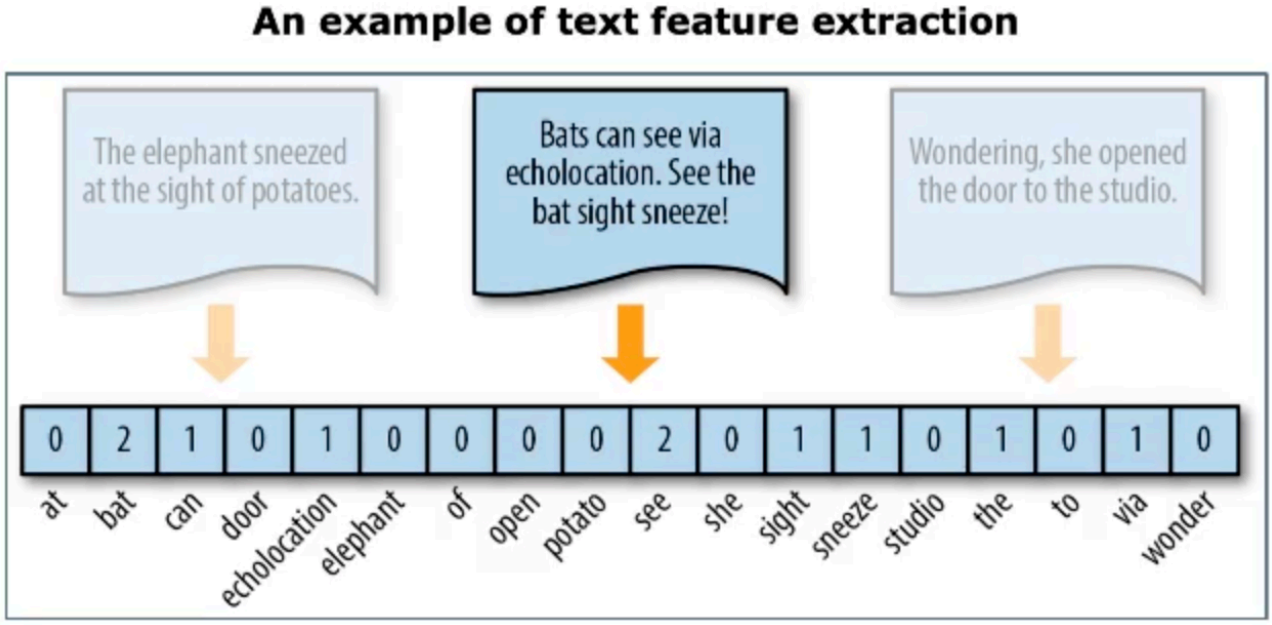

Abbildung 5.3: Text Feature Extraction

5.1.5 Feature Engineering vs. Featuree Learning

Moderne *Deep Learning* Algorithmen können Features direkt aus den Daten lernen. Aber häufig stehen nicht die umfangreichen Daten zur Verfügung die nötig wären um DL zu betreiben, welch auch gelabelt sein müssten. Ausserdem bietet die Genialität und Expertise des Menschen mehr als DL kann.

5.2 Feature Engineering for Tabular and Time-Series Data

Tabellarische Daten werden auch *Panel Data* genannt. Daher hat auch das Python Package *Pandas* seinen Namen (***Pan***el ***Da***ta).

5.2.1 Data Quality Assessment & Data Cleaning

Jedes Projekt startet mit einem DQA und Data Cleaning. *Daten-Normalisierung* ist auch entscheidend um einen dynamischen Wertebereich zwischen den Features zu vereinheitlichen. Die wichtigsten Dinge nochmals erwähnt:

- Originaldaten sicher aufbewahren
- Änderungen an den Daten dokumentieren
- Duplikate entfernen
- Irrelevante Daten entfernen
- Falsche bzw. unterschiedliche Bezeichnungen vereinheitlichen
- Outliers (Ausreisser) auf kausalität prüfen
- Fehlende Daten mit Mittelwert, Median füllen

5.2.2 Data Imputation

Der Prozess um fehlende Daten mit Werten die wir «raten» zu ersetzen wird *Imputation* genannt. Es ist ein Risiko, weil die Schlussfolgerung (Konklusion) verfälscht werden kann. Mit Vorsicht ausführen und nur, falls wenige fehlen. Sonst Feature komplett entfernen.

Nullwerte sind häufig mit *NaN* oder *NA* vermerkt. Mit der Pandas-Methode `df.fillna()` können nulls mit 0, dem *Median*, *Mean*, o.ä. ersetzt werden.

5.3 Engineering New Features

Mit den Techniken von *Grouping* und *Binning*.

5.3.1 Grouping

Mit Datengruppierung können wir *kategorische* Daten mit ähnlichen Typen zusammenfügen.

	Original Feature	New Feature
Bezeichnung	Title of Person	Gender of Person
Values	Mr., Sir, Miss, Lady, Mrs., Fr., Prof., Mx.	Male, Female, Unspecified

5.3.2 Binning

Mit Daten-Binning können *nummerische* Werte in Bereiche gliedern (in einem Bin).

Beispiel von Zeiten am Boston Marathon

	Original Feature	New Feature
Bezeichnung	Finishing Time	Finishing Group
Values	beliebige Zeit	2-3h, 3-4h, >4h

Zweck kann sein, die Leute zu finden, die nächstes Jahr ein anderen Startblock erhalten.

5.3.3 De-skewed Data

Wenn wir Daten binnen, werden einige Bin's mehr Einträge haben als andere. Solche Daten werden schief genannt (skewed). Um das zu beheben können wir auf den Originaldaten den Logarithmus anwenden und die Schiefheit wird oft entfernt. Dies muss aber **vor** dem Binning gemacht werden. Hint: Den Logarithmus kann man nur von positiven Zahlen ziehen. Ergo müssen negative Zahlen erst gemappt werden.

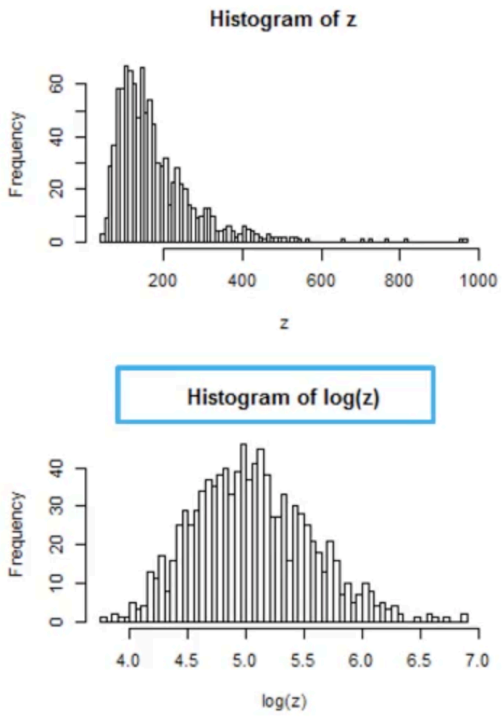

Abbildung 5.4: De-skewed Data

5.3.4 Kernel Trick

Manchmal sind die Daten die ein Feature beschreiben schwierig zu analysieren. Dann kann man ein *Kernel*-Feature hinzufügen. Im Beispiel auf 5.5 fügen wir das Feature $z = x^2 + y^2$ hinzu. Dies wird *Kernel Trick* genannt. Die Kernel-Funktion tranformiert die Daten in einen höher dimensionalen Featureraum. Die Daten können einfacher analysiert werden.

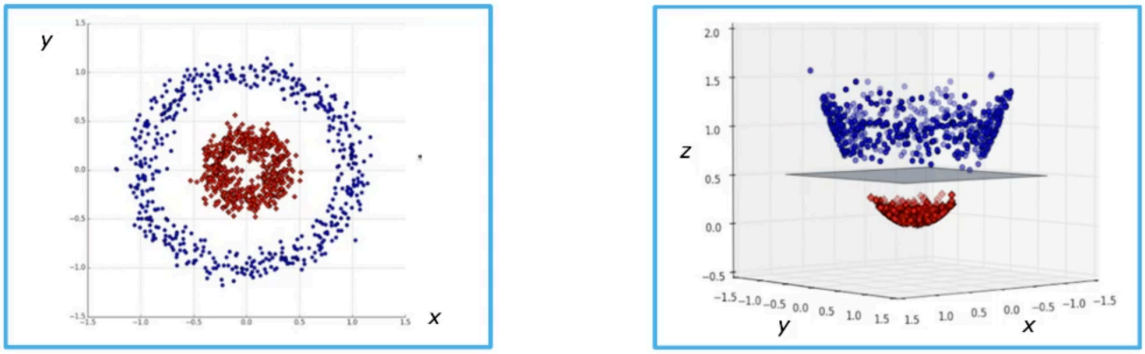

Abbildung 5.5: Kernel Trick

5.3.5 Expert Knowledge

Expertise ist eine der grösste Schlüssel um Daten zu verbessern. Als Beispiel der *Buffett Indicator*, welcher die Entwicklung von Börsenwerte mit einer eigens entwickelten Formel aus Indikatoren voraussagen kann. Dazu brauchte er Expertenwissen um das neue Feature, seinen Indikator, zu generieren. Dies wäre nicht offensichtlich nur aus den Rohdaten.

5.3.6 Transform Features

Daten in Zeitfolgen müssen in ihrer Originalreihenfolge bleiben. *Time-Frequency Transformationen* sind lineare Transformationen, welche die Zeitfolge beibehalten. Solche *Spectogramme* zeigen wie die Frequenz mit der Zeit ändert und können Dingee aufzeigen, die im Rohmaterial verborgen bleiben.

```
import matplotlib.pyplot as plot
from scipy.io import wavfile

signalData = wavfile.read('y.wav')
plot.specgram(signalData)
plot.xlabel('Time')
plot.ylabel('Frequency')
plot.show()
```

5.3.7 Expert Features in Time-Series

Experten sind oft nötig, um richtige Schlüssel zu ziehen, oder auf Features (Formeln oder Betrachtungsweisen) hinzuweisen die nützlich sein können. Damit kann zum Beispiel die *Votalität* abschätzen zu können und somit Risiken besser abzuschätzen.

5.4 Image Data & Computer Vision Applications

Digitale Bilddaten sind numerisch.

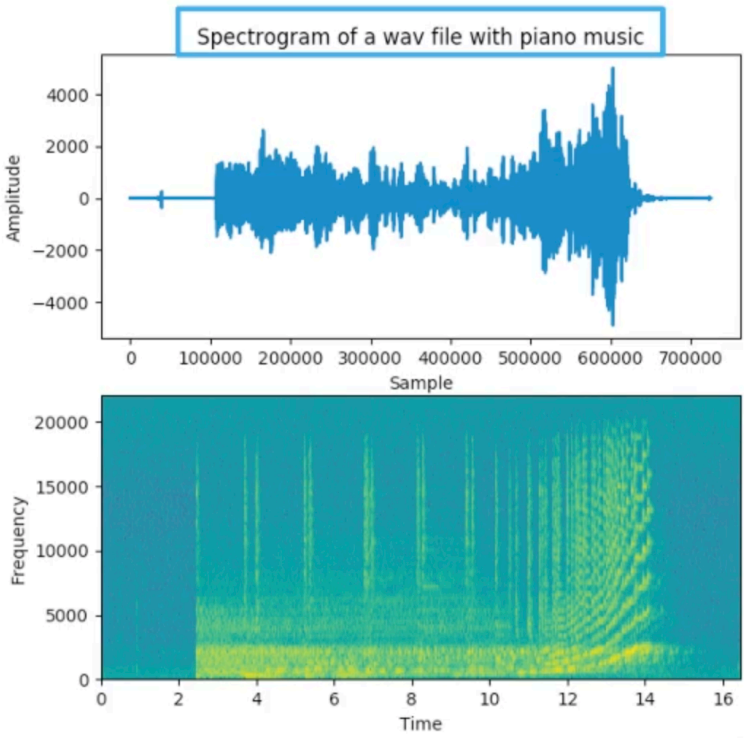

Abbildung 5.6: Spectogram

5.4.1 Edges are an Important Feature

In einem Bild sind *Features* besonders wichtig. Wir können aus einer kleiner Skizze schon viel entnehmen. *Edges* sind eines der wichtigsten Image-Features. Weitere sind: * gerade Linien * Ecken * Objekte * Segmente (gleiches Material, gleiche Farbe) * usw.

Es gibt viele Algorithmen um Feature zu detektieren.

5.4.1.1 Edge Detection

Ein einfache Möglichkeit um Ecken zu finden ist der *Gradient* eines Bilders zu analysieren. Der Gradient $\nabla(I)$ eine Bildes I ist die erste Ableitung in eine Richtung. Eine weitere Möglichkeit ist das übereinanderlegen des Bildes und verschieben um einen Pixel in eine Richtung. Danach Pixelwerte vom Original subtrahieren. Je nach Richtung erhält man die Ableitung nach x bzw. y. Euklidischer Betrag:

$$I_{\text{edges}} = \sqrt{\nabla_x^2(I) + \nabla_y^2(I)}$$