Data Mining & Pattern Recognition

Versuch 1 und 2:
Allgemeine Data Mining Prozessschritte

Datum:	
Namen:	
Unterschrift Betreuer:	

Prof. Dr. Johannes Maucher
Medieninformatik Bachelor (MIB)
WS 13/14

Inhaltsverzeichnis

1 Einführung

1.1 Lernziele

In diesem Versuch sollen Kenntnisse in folgenden Themen vermittelt werden:

- **Zugriff auf Daten** auf lokalem Speicher und aus dem Web.
- **Statistische Datenauswertung**
- **Data Mining Prozesskette:** Vorverarbeitung, Merkmalsauswahl, Modellbildung, Visualisierung, Interpretation.
- **scikit-learn:** Ist eine in Python implementierte Open-Source Machine Learning Bibliothek, in der sämtliche Verfahren der Data Mining Prozesskette bereits implementiert sind.
- **Maschinelles Lernen:** Überwachtes und Unüberwachtes Lernen im Prinzip
- **Google Maps** zur Darstellung geographisch strukturierter Daten.

1.2 Protokoll

Pro Gruppe ist ein Protokoll (als .pdf) spätestens vor Beginn des nächsten Versuchs abzugeben. Bitte nummerieren Sie ihre Antworten entsprechend der Nummerierung in der Versuchsanleitung. Die Durchführung und Dokumentation eigener Experimente, die über die Aufgabenstellungen in diesem Dokument hinausgehen, wird ausdrücklich unterstützt und in der Bewertung entsprechend berücksichtigt. Für die in dieser Anleitung beschriebenen Aufgaben sind 2 Termine vorgesehen. Das gesamte Protokoll ist also vor der Durchführung des dritten Versuchs abzugeben.

1.3 Theorie zur Vorbereitung

1.3.1 Data Mining Prozesskette

Unter Data Mining versteht man die Analyse großer Datenmengen mit dem Ziel verborgene Muster, Strukturen oder Modelle zu erkennen. Für diese Analyse werden systematisch Methoden der Künstlichen Intelligenz (genauer: des maschinellen Lernens) angewandt.

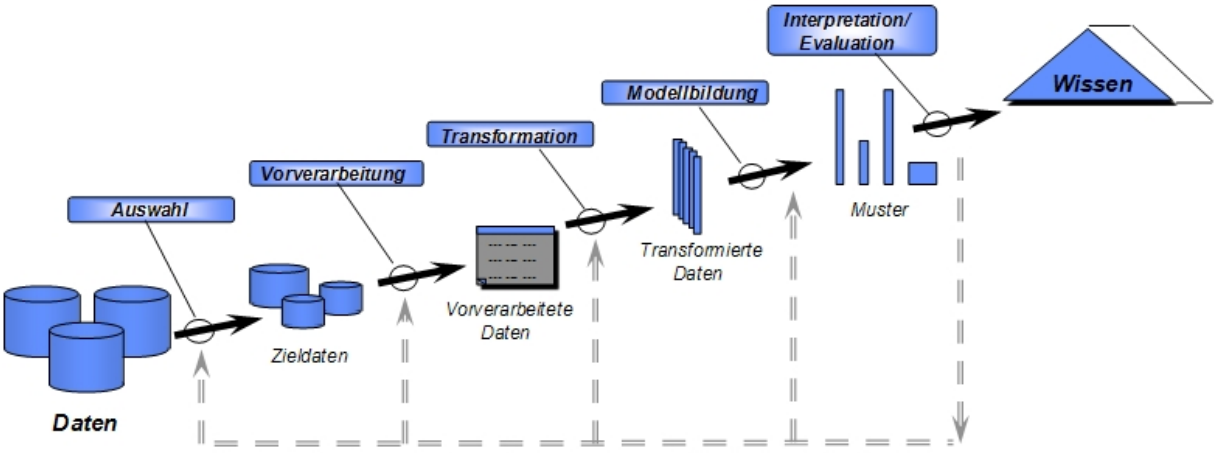


Abbildung 1: Stufen des Data Mining Prozess

Ausgehend von einer möglichst konkreten Aufgabendefinition läßt sich der Data Mining Prozeß zur Lösung dieser Aufgabe allgemein in folgende Prozessschritte unterteilen.

Datenauswahl: Aus dem vorhandenen Datenbestand müssen die Daten ausgewählt werden, welche für die aufgabenspezifische Analyse als sinnvoll und relevant eingeschätzt werden.

Vorverarbeitung: In diesem Prozessschritt kommen Verfahren zur Erkennung, Aufbereitung oder Eliminierung von fehlenden oder falschen Daten zum Einsatz.

Transformation: Ziel der Transformation ist es durch Reduktion oder Projektion die Anzahl der Attribute, mit welchen die Datensätze beschrieben werden, zu reduzieren. Abhängig von der jeweiligen Aufgabe und der im folgenden Prozessschritt verwendeten Lernmethode werden Atrribute die keinen oder nur einen geringen Einfluss auf das Modell haben bestimmt und eliminiert. Sinn dieser Aufgabe ist nicht nur eine Reduktion der Komplexität der nachfolgenden Verarbeitungsschritte sondern auch eine bessere Mustererkennung bzw. Modellbestimmung. Neben der Attributreduktion werden zu diesem Prozessschritt auch vorverarbeitende Methoden wie die Normalisierung der Attribute gezählt.

Modellbildung: Hier werden die Daten mit Hilfe einer Methode des überwachten oder unüberwachten Lernens analysiert. Mit dem gelernten Modell können neue Daten klassifiziert, Vorhersagen getroffen oder auffällige Muster erkannt werden.[1]

Evaluation: Die Daten welche im vorigen Schritt für die Modellbildung zum Einsatz kommen werden **Trainingsdaten** genannt. Das gelernte Modell ist demnach auf die Trainingsdaten angepasst was aber noch lange nicht bedeutet, dass es gut generalisiert, d.h. dass es auch für nicht in der Trainingsmenge enthaltene Daten gute Ergebnisse liefert. Deshalb werden bei der Evaluation neue Daten, die sogenannten **Testdaten**, herangezogen und auf diese das gelernte Modell angewendet. Falls die Anwendung des Modells auf die Testdaten unbefriedigend ist, muss dieses entsprechend verbessert werden.

Interpretation: Das gelernte und getestete Modell impliziert neues, aus den Daten gewonnenes Wissen, wenn das Modell korrekt interpretiert wird.

1.3.2 Daten

Unter Daten wird hier eine Menge von Datensätzen D_i verstanden. Jeder Datensatz wird durch Z Eingabeattribute (oder Features) $A_1, A_2 \ldots, A_Z$ beschrieben. Der Wert des $j.ten$ Eingabeattributs von Datensatz D_i wird mit $d_{i,j}$ bezeichnet.

	$\mathbf{A_1}$	$\mathbf{A_2}$	$\cdots$	$\mathbf{A_Z}$
$\mathbf{D_1}$	$d_{1,1}$	$d_{1,2}$	$\cdots$	$d_{1,Z}$
$\mathbf{D_2}$	$d_{2,1}$	$d_{2,2}$	$\cdots$	$d_{2,Z}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$\mathbf{D_N}$	$d_{N,1}$	$d_{N,2}$	$\cdots$	$d_{N,Z}$

(1)

In Abbildung 2 sind exemplarisch Energieverbrauchsdaten einiger Länder dargestellt. Jedes Land wird durch einen Datensatz mit den Attributen

Oil: Ölenergieverbrauch in Mio Tonnen (p.a.)

Gas: Gasenergieverbrauch in Mio Tonnen Ölequivalent (p.a.)

[1] Dieser Schritt wird häufig auch als Data Mining im engeren Sinn bezeichnet

	Oil	Gas	Coal	Nuclear	Hydro	Total2009	CO2Emm	Longitude	Latitude	Country
1	842.9	588.7	498.0	190.2	62.2	2182.0	5941.9	-98.580002	39.830002	US
2	97.0	85.2	26.5	20.3	90.2	319.2	602.7	-98.308968	56.954681	Canada
3	85.6	62.7	6.8	2.2	6.0	163.2	436.8	-101.956253	23.625740	Mexico
4	22.3	38.8	1.1	1.8	9.2	73.3	164.2	-63.584808	-37.090698	Argentina
5	104.3	18.3	11.7	2.9	88.5	225.7	409.4	-54.387829	-14.242920	Brazil
6	15.4	3.0	4.1	0.0	5.6	28.1	70.3	-71.341087	-35.705219	Chile
7	8.8	7.8	3.1	0.0	9.3	29.0	57.9	-72.958527	4.116410	Colombia
8	9.9	0.4	0.0	0.0	2.1	12.4	31.3	-78.106667	-1.781560	Ecuador
9	8.5	3.1	0.5	0.0	4.5	16.6	35.5	-75.002357	-9.181340	Peru
10	27.4	26.8	0.0	0.0	19.5	73.6	147.0	-66.589043	6.472790	Venezuela
11	13.0	8.4	2.3	0.0	8.3	32.0	68.8	13.345770	47.696510	Austria
12	2.8	6.9	0.0	0.0	0.5	10.2	24.8	47.687069	40.151340	Azerbaijan
13	9.3	14.5	0.0	0.0	0.0	23.9	62.9	27.973749	53.711048	Belarus
14	38.5	15.6	4.6	10.7	0.1	69.4	172.8	0.000000	0.000000	Belgium
15	4.4	2.2	6.3	3.5	0.9	17.4	43.7	25.483120	42.731602	Bulgaria
16	9.7	7.4	15.8	6.1	0.7	39.6	109.5	15.474910	49.803879	Czech Republic
17	8.2	4.0	4.0	0.0	0.0	16.1	50.3	9.516950	56.276089	Denmark
18	9.9	3.2	3.7	5.4	2.9	25.0	52.5	26.067390	64.950142	Finland
19	87.5	38.4	10.1	92.9	13.1	241.9	398.7	1.718190	46.710670	France
20	113.9	70.2	71.0	30.5	4.2	289.8	795.6	10.454150	51.164181	Germany
21	20.2	3.0	7.9	0.0	1.6	32.7	100.4	21.845560	39.072449	Greece
22	7.3	9.1	2.5	3.5	0.1	22.4	53.6	19.504959	47.161160	Hungary
23	1.0	0.0	0.1	0.0	2.8	3.9	3.5	-19.021160	64.963943	Iceland
24	8.0	4.3	1.3	0.0	0.2	13.9	40.1	-8.240550	53.419609	Republic of Ireland
25	75.1	64.5	13.4	0.0	10.5	163.4	434.8	12.573470	42.503819	Italy

Abbildung 2: Beispieldaten Energieverbrauch

Coal: Kohleenergieverbrauch in Mio Tonnen Ölequivalent (p.a.)

Nuclear: Nuklearenergieverbrauch in Mio Tonnen Ölequivalent (p.a.)

Hydro: Wasserenergieverbrauch in Mio Tonnen Ölequivalent (p.a.)

Total 2009: Summe Verbrauch aller o.g. Energieformen in Mio Tonnen Ölequivalent (p.a.)

Longitude: Longitude Koordinate des Landes

Latitude: Latitude Koordinate des Landes

Country: Name des Landes

beschrieben.

Ein Modell weist den durch die Z Eingabeattribute beschriebenen Datensätze ein Ausgabeattribut r oder einen Vektor von Ausgabeattributen $\mathbf{r}$ zu. Die Daten sind dann allgemein von der Form

	$\mathbf{A_1}$	$\mathbf{A_2}$	$\cdots$	$\mathbf{A_Z}$	$\mathbf{r}$
$\mathbf{D_1}$	$d_{1,1}$	$d_{1,2}$	$\cdots$	$d_{1,Z}$	r_1
$\mathbf{D_2}$	$d_{2,1}$	$d_{2,2}$	$\cdots$	$d_{2,Z}$	r_2
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	
$\mathbf{D_N}$	$d_{N,1}$	$d_{N,2}$	$\cdots$	$d_{N,Z}$	r_N

(2)

In den in Abbildung 2 dargestellten Daten ist das Ausgabeattribut die CO_2-Emmission in Mio Tonnen pro Jahr. Im Versuch wird der Wert dieses Ausgabeattributs aus den Werten der o.g. Eingabeattribute geschätzt.

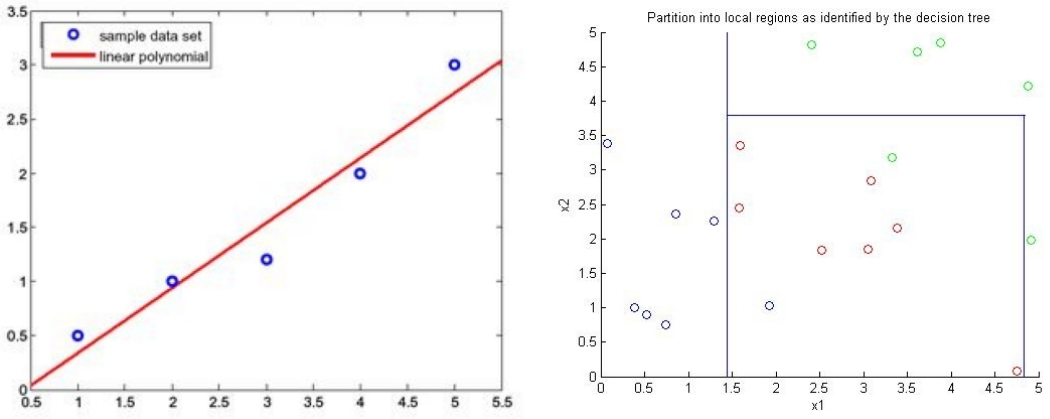


Abbildung 3: Beispiele für Regression (links) und Klassifikation (rechts)

1.3.3 Überwachtes Lernen für Klassifizierung und Regression

Im Prozessschritt *Modellbildung* wird eine möglichst allgemeingültige Zuordnung der Eingabeattribute zu den Ausgabeattributen gelernt. Das Ausgabeattribut kann ein diskreter Wert, welcher die Zugehörikeit des Datensatzes zu einer bestimmten Klasse beschreibt sein. In diesem Fall spricht man von **Klassifizierung**. Das Ausgabeattribut kann aber auch ein numerischer Funktionswert sein. In diesem Fall ist das gelernte Modell eine Funktion. Das entsprechende Verfahren wird **Regression** genannt.

Sowohl für die Klassifikation als auch für die Regression kommen in der Regel Verfahren des überwachten Lernens zum Einsatz. Im Fall des überwachten Lernens müssen Trainingsdaten[2] der in Tabelle 2 gegebenen Form vorliegen, also Datensätze, denen bereits eine *Sollausgabe* **r** zugeordnet ist. Aus diesen *Eingabe/Sollausgabe*-Paaren ermittelt der Lernalgorithmus ein Modell, welches eine allgemeingültige Zuordnung von Eingabe zu Ausgabe beschreibt. Mit dem Modell können dann für alle Daten der Form 1 die zugehörigen Ausgaben **r** berechnet werden.

In Abbildung 3 ist links ein Beispiel für eine Regression dargestellt. Die Trainingsdaten sind hier die blauen Punkte. Jeder Punkt wird durch einen Wert auf der x-Achse (Eingabeattribut A_1) und einen zugehörigen y-Wert (Ausgabeattribut r) beschrieben. Aus der Menge der 5 Trainingsdatensätze wird, z.B. mit einem Single-Layer Perzeptron, eine allgemeine Funktion gelernt, welche in diesem Fall die rot eingetragene Gerade ist.

Auf der rechten Seite in Abbildung 3 ist ein Beispiel für eine Klassifikation dargestellt. Die Trainingsdaten sind hier wieder die Punkte im zweidimensionalen Raum. Jeder Punkt wird durch zwei Eingabeattribute A_1 und A_2 beschrieben. Die Trainingspunkte enthalten zudem die Sollausgabe r, die jetzt die Klassenzugehörigkeit angibt. In der Abbidung ist die Soll-Klassenzugehörigkeit durch die Punktfarbe markiert. Gelernt wird eine allgemeine Regel, welche Datensätzen mit beliebigen Eingabeattributen einer Klasse zuordnet. Diese gelernte allgemeine Regel ist im Bild durch die Begrenzungslinien dargestellt.

1.3.4 Unüberwachtes Lernen für Clustering

Beim unüberwachten Lernen enthalten die verfügbaren Trainingsdaten keine Sollausgabe, sondern nur die Eingabeattributswerte. Die Trainingsdaten sind also von der in Tabelle 1 dargestellten Form. Mit

[2] Trainingsdaten werden häufig auch Beispiele genannt

dem unüberwachten Lernverfahren wird ein Clustering gelernt. Das gelernte Clustering partitioniert die Trainingsdaten derart in Untergruppen, dass Datensätze innerhalb einer Untergruppe untereinander sehr ähnlich sind, Datensätze unterschiedlicher Gruppen jedoch nur ein geringes Maß an Ähnlichkeit aufweisen. Während bei der oben beschriebenen Aufgabe der Klassifizierung mit überwachtem Lernen die Klassenzugehörigkeit der Trainingsdaten schon gegeben ist, muss diese im Fall des unüberwachten Lernens erst gelernt werden. Zum Einsatz kommen derartige Verfahren u.a. für die Kategorisierung von Kunden aber auch in der Bildsegmentierung oder in der nichtlinearen Vektorquantisierung.

1.3.5 Externe Referenzen zur Vorbereitung

Grundlagen in

- Python: `http://www.hdm-stuttgart.de/~maucher/Python/html/index.html`
- Pandas: `http://pandas.pydata.org/pandas-docs/stable/10min.html`.
- Numpy: `http://www.scipy.org/Tentative_NumPy_Tutorial`.
- Scipy:
  - `http://docs.scipy.org/doc/scipy/reference/spatial.distance.html`
  - `http://docs.scipy.org/doc/scipy/reference/cluster.html`.
- Matplotlib: `http://matplotlib.org/users/pyplot_tutorial.html`
- Scikit Learn:
  - `http://scikit-learn.org/stable/index.html`
  - `http://www.hdm-stuttgart.de/~maucher/Python/SklearnIntro/html/index.html`

1.4 Vor dem Versuch zu klärende Fragen

Eine Untermenge der im Folgenden aufgeführten Fragen wird zu Beginn des Versuchs im Rahmen eines Gruppenkolloqs abgefragt. Auf jede Frage sollte von mindestens einem Gruppenmitglied eine Antwort geliefert werden und jedes Gruppenmitglied muss mindestens eine der gestellten Fragen beantworten können.

Aus der in Kapitel 1.3 beschriebenen Theorie

1. Erklären Sie den Sinn der *Transformation* innerhalb des Data Mining Prozesskette.
2. Worin besteht der Unterschied zwischen überwachtem und unüberwachtem Lernen?
3. Beschreiben Sie die Unterschiede zwischen Klassifikation, Regression und Clustering. Nennen Sie für diese 3 verschiedenen Verfahren je ein Anwendungsbeispiel.

Wenn man Elementen aus einer bestehenden Liste eine neue Liste erstellen möchte, Zweck geschaffen wurden die sogenannten List Comprehensions. Die folgende List Comprehension erzeugt aus einer Liste mit ganzen Zahlen eine neue Liste, die die Quadrate dieser Zahlen enthält

```
>>> lst = [1,2,3,4,5,6,7,8,9]
>>> [x**2 for x in lst]
[1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Python Allgemein:

1. Was ist eine Python List-Comprehension?
2. Wie importiert man Daten aus einem Textfile?
3. Wie speichert man Daten aus Python in ein Textfile?
4. Wie hängt man an eine Python-Liste die Elemente einer zweiten Liste an?

Numpy:

1. Nennen Sie zwei verschiedene Möglichkeiten ein Numpy-Array zu erzeugen.
2. Wie legt man ein $(3, 4)$-Array mit ausschließlich 0-Einträgen an?
3. Wie ruft man die Anzahl der Dimensionen, die Anzahl der Elemente pro Dimension und den Datentyp der Arrayelemente ab?
4. Wie wandelt man ein $(3, 4)$-Array in ein $(2, 6)$-Array um?
5. Wie transponiert man ein zweidimensionales Array?
6. Wie multipliziert man zwei Arrays elementweise?
7. Wie führt man eine Matrixmultiplikation zweier zweidimensionaler Arrays A und B aus? Welche Bedingungen müssen A und B erfüllen, damit überhaupt eine Matrixmultiplikation durchgeführt werden kann?
8. Wie greift man auf das Element $(2, 3)$ in einem $(4, 4)$-Array A zu? Wie greift man auf die erste Spalte, wie auf die erste Zeile dieses Arrays zu?
9. Wie berechnet man die Quadratwurzel aller Elemente eines Arrays?
10. Wie legt man eine flache Kopie, wie eine tiefe Kopie eines Arrays an?

Pandas:

1. Wie wird ein Numpy-Array in einen Pandas-Dataframe geschrieben? Wie legt man dabei die Spaltenbezeichnungen und einen Index an?
2. Wie kann auf einzelne Spalten, wie auf einzelne Zeilen eines Pandas Dataframes zugegriffen werden?
3. Wie können Pandas Dataframes sortiert werden?
4. Wie kann zu einem bestehenden Dataframe eine neue Spalte hinzugefügt werden?
5. Wie werden Daten aus einem .csv File in einen Pandas Dataframe geschrieben?
6. Wie wird ein Pandas Dataframe in einem .csv File abgelegt?

Matplotlib:

1. Wie erzeugt man mit Matplotlib einen Plot, wie er in Abbildung 4 dargestellt ist?
2. Wie kann man mehrere Graphen in einen Plot eintragen?
3. Wie erzeugt man mit Matplotlib ein Bild, das 12 Subplots in 3 Zeilen und 4 Spalten geordnet, enthält.
4. Wie erzeugt man mit Matplotlib ein Histogramm?
5. Wie erzeugt man mit Matplotlib einen Boxplot?

Scipy:

- Geben Sie kurz die Schritte an, die für die Durchführung eines hierarchischen Clustering mit Scipy notwendig sind.

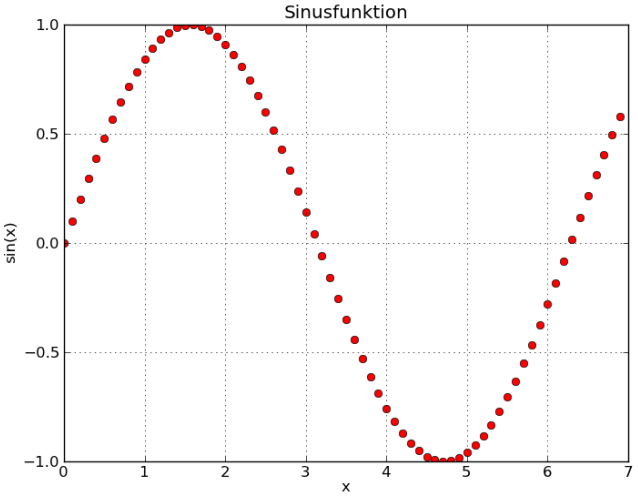


Abbildung 4: Sinusfunktion mit Matplotlib

Scikit Learn:

- Sklearn stellt u.a. eine umfassende Bibliothek von Klassen für das überwachte Lernen bereit. Mit welchem Methodenaufruf werden diese Kassen trainiert? Mit welchem Methodenaufruf können die trainierten Modelle auf neue Eingabedaten angewandt werden um den entsprechenden Ausgabewert zu berechnen?
- Mit welchem Leistungsmaß kann die Qualität eines Regressionsmodells bewertet werden? Wie wird dieses Leistungsmaß mit Sklearn berechnet?
- Mit welchem Leistungsmaß kann die Qualität eines Klassifikationsmodells bewertet werden? Wie wird dieses Leistungsmaß mit Sklearn berechnet?
- Was versteht man unter *x-facher Kreuzvalidierung* und wie wird diese mit Sklearn durchgeführt?

2 Durchführung Teil 1: Energieverbrauch und CO2-Emmission

2.1 Datenverwaltung und Statistik

2.1.1 Daten

Die in diesem Versuch zu analysierenden Daten entstammen dem *BP Statistical Review of World Energy 2010*[3]. Das entsprechende Excel-File steht auf dem Skripteserver zum Download bereit. Ebenfalls zum Download bereit steht die Datei `EnergyMix.txt`. In dieser Datei befinden sich die Daten des o.g. Excel Files, die für den Versuch relevant sind.

2.1.2 Einlesen der Daten, Hinzufügen der GPS Koordinaten, Abspeichern in neuer Datei

Schreiben Sie ein Python Programm `appendGeoCordinates.py` mit folgender Funktion

1. Einlesen der Daten aus `EnergyMix.csv` mit der entsprechenden *Pandas* Methode in ein *Pandas Dataframe* Objekt.
2. Grafische Ausgabe des Energieverbrauchs der Länder (ein Graph pro Energieform) mit den entsprechenden Plot-Methoden von *Pandas* und *Matplotlib*.
3. Berechnung der Geokoordinaten aller Länder mit Hilfe der hier beschriebenen Funktion und der Google Maps API.
4. Erweiterung des *Pandas Dataframe* mit den Spalten *Lat* und *Long*, die die zuvor berechneten Geokoordinaten enthalten.
5. Abspeichern des so erweiterten Dataframe in einer Datei `EnergyMixGeo.csv` mit Hilfe der entsprechenden *Pandas* Methode.

Fragen:

1. Ausgehend von der implementierten Visualisierung des Energieverbrauchs der Länder: Nennen Sie die 3 Ihrer Meinung nach interessantesten Beobachtungen.

Abzugeben sind:

1. Das Programm `appendGeoCordinates.py`
2. Screenshot, der die graphische Darstellung des Energieverbrauchs zeigt
3. Antwort auf die oben gestellte Frage.

2.1.3 Statistik der Daten

Schreiben Sie ein Python Programm `energyStatistics.py` mit folgender Funktionalität:

1. Einlesen der Daten aus `EnergyMix.csv`
2. Mit der entsprechenden *Pandas*-Methode soll eine Zusammenfassung der Statistik aller im File enthaltenen Energieresourcen berechnet und angezeigt werden.
3. Für jedes der Merkmale *Öl, Gas, Kohle, Nuklear* und *Hydro* soll die Statistik in Form eines Boxplot angezeigt werden (siehe `http://matplotlib.sourceforge.net/api/pyplot_api.html#matplotlib.pyplot.boxplot`). Das Merkmal *Total 2009* kann in dieser und allen weiteren Teilaufgaben ignoriert werden.

[3] `http://www.bp.com/sectionbodycopy.do?categoryId=7500&contentId=7068481`

Fragen:

1. Erklären Sie sämtliche Elemente eines Boxplot (allgemein).
2. Diskutieren Sie die im Boxplot angezeigte Statistik der Energieverbrauchdaten.

Abzugeben sind:

1. Das Programm `energyStatistics.py`
2. Bilder der 5 Boxplots.
3. Antworten auf die oben gestellten Fragen.

2.2 Anwendung von Verfahren des unüberwachten Lernens auf Energieverbrauchsdaten

In den folgenden Aufgaben werden zunächst nur die Merkmale *Ländername, Öl, Gas, Kohle, Nuklear* und *Hydro* benötigt.

2.2.1 Hierarchisches Clustering

Schreiben Sie ein Programm `energyClustering.py` mit folgender Funktionalität:

1. Einlesen der Daten aus `EnergyMixGeo.csv`
2. Standardisieren der Daten mit der scikit-learn Methode *scale()*[4]. Dabei soll keine Verschiebung um den Mittelwert vorgenommen werden. Für die weiteren Funktionen sind diese **normierten Daten zu verwenden**.
3. Hierarchisches Clustering der Länder anhand der Merkmale Öl, Gas, Kohle, Nuklear und Hydro:
   - Berechnen Sie zunächst mit der Funktion `pdist` aus dem Paket `scipy.spatial.distance`[5] die Distanzmatrix. Die Distanzmatrix enthält die Distanzen (gemeint sind die Distanzen zwischen den 5-dimensionalen Energieverbrauchs-Vektoren der Länder) aller möglichen Länderpaare. Stellen Sie als Metrik die `correlation` ein.
   - Berechnen Sie mit der Funktion `linkage` aus dem Paket `scipy.cluster.hierarchy`[6] das hierarchische Clustering der Länder nach ihrem relativen Energieverbrauch.
   - Zeigen Sie das hierarchische Clustering mit der Funktion `dendrogram` aus dem Paket `scipy.cluster.hierarchy` an und speichern Sie dieses Bild unter dem Namen `dendrogram.png`.
   - Benutzen Sie die Funktion `fcluster` aus dem Paket `scipy.cluster.hierarchy` um die Länder entsprechend dem zuvor berechneten Clustering in 4 verschiedene Cluster zu unterteilen. Die Funktion weist den Instanzen die entsprechenden Clusterlabels zu.
4. Für jeden der 4 Cluster ist jeweils ein Plot anzulegen. Jeder clusterspezifische Plot soll für jedes zum Cluster gehörige Land einen Graphen über den Verbrauch hinsichtlich der 5 unterschiedlichen Energiequellen enthalten (siehe Abbildung 6). Speichern Sie den Multiplot unter dem Namen `individualClusters.png` ab.
5. Schreiben Sie die Clusterindizees der Datensätze in eine neue Spalte des *Pandas Dataframe* und speichern Sie das so erweiterte Dataframe in der Datei `EnergyMixGeo.csv` ab.

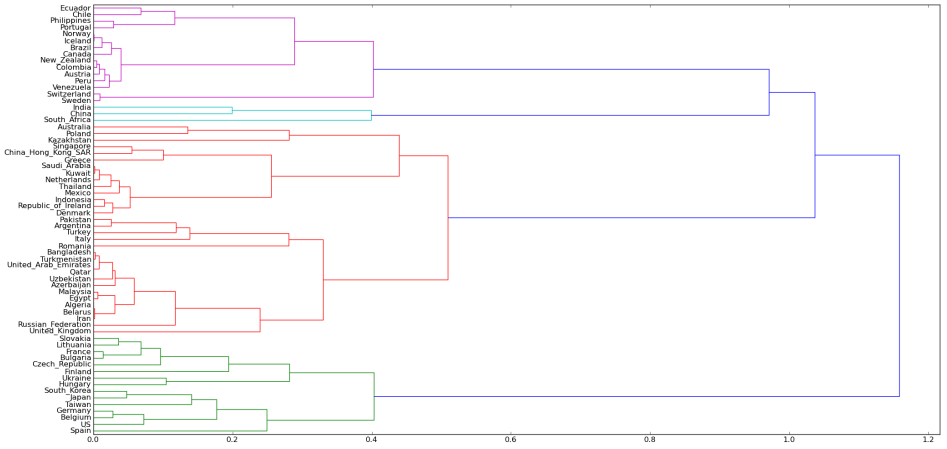


Abbildung 5: Dendrogram Darstellung des hierarchischen Clustering

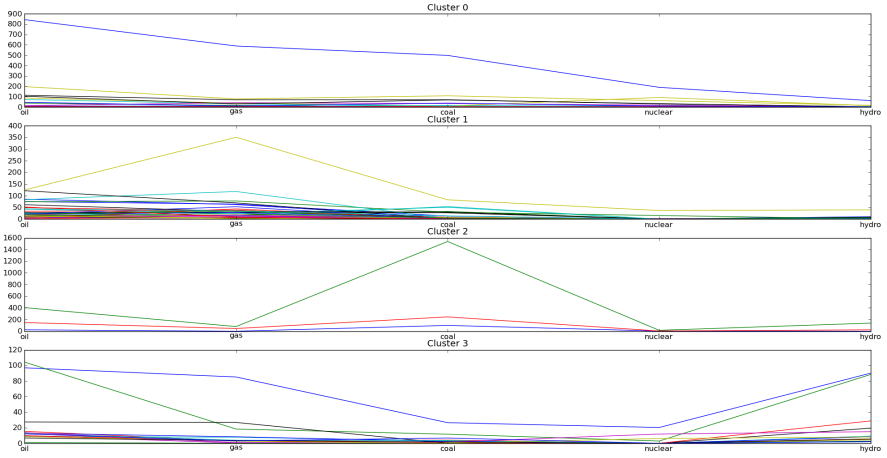


Abbildung 6: Dendrogram Darstellung des hierarchischen Clustering

Fragen:

1. Was wird beim *Standardisieren* gemacht? Welcher Effekt könnte ohne Standardisieren beim Clustering eintreten (insbesondere wenn die euklidische Metrik verwendet wird)?
2. Erklären Sie die beim hierarchischen Clustering einstellbaren Parameter *linkage-method* und *metric*. Welche Metrik ist Ihrer Meinung nach für diese Anwendung geeignet? Warum?
3. Welches Land ist bezüglich des Verbrauchs der hier betrachteten Energiequellen Deutschland am ähnlichsten, wenn für die linkage-method *average* und die Metrik *correlation* konfiguriert wird?
4. Charakterisieren Sie die 4 Cluster. Was ist typisch für die jeweiligen Cluster?

Abzugeben sind:

1. Das Programm `energyClustering.py`
2. Die Bilder `dendrogram.png` und `individualClusters.png`
3. Antworten auf die oben gestellten Fragen.

2.2.2 Dimensionalitätsreduktion

Die Ähnlichkeit zwischen Dateninstanzen läßt sich im Fall von nur 2 oder 3 Merkmalen grafisch im 2- bzw. 3-dimensionalen Raum darstellen. Jede Instanz entspricht in diesem Fall einem Punkt. Unter Verwendung der euklidischen Metrik sind sich dann nahe beieinanderliegende Instanzen ähnlicher als weit voneinander entfernte. Im Fall von hoch-dimensionalen Daten läßt sich die Ähnlichkeit nicht mehr direkt visualisieren. Allerdings gibt es Methoden wie die Multidimensionale Skalierung oder die Isomap, welche eine Transformation in einen niedrig-dimensionalen Raum (meist 2 Dimensionen) vornehmen, wobei die Distanzen zwischen den Instanzen weitestgehend erhalten bleiben. In diesem Abschnitt soll die Ähnlichkeit zwischen den Ländern in der 2-dimensionalen Ebene visualisiert werden nachdem die 5-dimensionalen Daten mit einer Isomap in den 2-dimensionalen Raum transformiert wurden.

Schreiben Sie ein Programm `energyReduceDim.py` mit folgender Funktionalität:

1. Einlesen der Daten aus `EnergyMixGeo.csv` mit Hilfe der entsprechenden *Pandas* Methode.
2. Transformation der 5-dimensionalen Daten (die Verbrauchsdaten hinsichtlich der 5 verschiedenen Resourcen) in den 2-dimensionalen Raum mit der scikit-learn Methode *Isomap()*[7].
3. Darstellung der transformierten Daten in einem 2-dimensionalen Matplotlib-Plot. Jeder Punkt soll mit dem entsprechenden Ländernamen angezeigt werden.

Fragen:

1. Welches Land ist nach dieser Darstellung Deutschland am ähnlichsten?
2. Warum entspricht die hier dargestellte Ähnlichkeit nicht der im oben erzeugten Dendrogram?

Abzugeben sind:

1. Das Programm `energyReduceDim.py`
2. Die grafische Darstellung des Isomap-Resultats im 2-dimensionalen Raum
3. Antworten auf die oben gestellten Fragen.

[4] `http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.scale.html#sklearn.preprocessing.scale`
[5] `http://docs.scipy.org/doc/scipy/reference/spatial.distance.html`
[6] `http://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html`
[7] `http://scikit-learn.org/stable/modules/generated/sklearn.manifold.Isomap.html`

2.3 Überwachtes Lernen: Schätzung der CO2-Emmission

Im Folgenden soll eine Regression der CO2 Emmission durchgeführt werden. D.h. Aus den in 2.2 genannten Eingangsmerkmalen *Öl, Gas, Kohle, Nuklear* und *Hydro* soll durch Anwendung eines überwachten Lernverfahrens ein Modell erlernt werden, welches die Vorhersage der CO2-Emmission aus den Eingangsmerkmalen erlaubt. Vor der eigentlichen Regression soll im ersten Unterkapitel eine Merkmalsauswahl (*engl.: Feature Selection*) implementiert werden.

2.3.1 Feature Selection

Eine gute Merkmalsvorauswahl ist ein entscheidender Faktor in der Data Mining Prozeßkette. Durch das Herausfiltern unrelevanter Merkmale wird zum einen die Komplexität der nachfolgenden Prozeßschritte reduziert. Ausserdem wird eine bessere, insbesondere robustere Modellbildung ermöglicht. Die Anzahl der notwendigen Trainingsdaten steigt stark mit der Anzahl der Eingabemerkmale. D.h. insbesondere dann wenn relativ wenig Trainingsdaten vorliegen sollte die Anzahl der Eingabemerkmale auf das Notwendigste reduziert werden.

Schreiben Sie ein Programm `energyFeatureSelection.py` mit folgender Funktionalität:

1. Einlesen der Daten aus `EnergyMixGeo.csv` mit Hilfe der entsprechenden *Pandas* Methode. Als Eingangsmerkmale sind weiterhin Öl, Gas, Kohle, Nuklear und Hydro zu verwenden. Als Ausgangsmerkmal (Target) dient jetzt die CO2-Emmission.
2. Feature Selection mit der scikit-learn Methode `SelectKBest()`[8] unter Anwendung der scoringfunktion `chi2`.
3. Anzeige des Scores der einzelnen Features. Die Scores sind über das Attribut *scores* des trainierten Objekts der Klasse `SelectKBest()` abrufbar.

Frage:

1. Welche 3 Merkmale haben den stärksten Einfluß auf das Ausgabemerkmal CO2-Emmission? Wie groß sind die vom Programm ausgegebenen Scores?

Abzugeben sind:

1. Das Programm `energyFeatureSelection.py`
2. Antworten auf die oben gestellte Frage.

2.3.2 Regression mit Epsilon-SVR

Für diese Teilaufgabe sind die Eingabemerkmale Öl, Gas, Kohle, Nuclear und Hydro zu verwenden. Das Ausgabemerkmal ist die CO2-Emmission.
Schreiben Sie ein Programm `energyPrediction.py` mit folgender Funktionalität:

1. Einlesen der Daten aus `EnergyMixGeo.csv` mit Hilfe der entsprechenden *Pandas* Methode.
2. Anlegen einer Epsilon-SVR[9] mit linearem Kernel.
3. 10-fache Kreuzvalidierung[10] unter Anwendung des Mean Square Error als score-function.

[8] `http://scikit-learn.org/stable/modules/feature_selection.html#feature-selection`
[9] `http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html#sklearn.svm.SVR`
[10] `http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.KFold.html#sklearn.cross_validation.KFold`

4. Ausgabe des Scores in allen Iterationen der Kreuzvalidierung
5. Nach der 10-fachen Kreuzvalidierung soll die SVR mit allen vorhandenen Datensätzen trainiert werden.
6. Die Ausgabe (Prediction) des trainierten Modells für die Trainingsdaten ist zu berechnen.
7. Die mittlere absolute Abweichung zwischen Soll- und Ist-Ausgabe ist zu berechnen?
8. Soll- und Ist-Ausgabe sollen in einem gemeinsamen Matplotlib-Diagramm visualisiert werden.

Fragen:

1. Optimieren Sie die SVR-Parameter C und Epsilon so dass der Score in der Kreuzvalidierung minimal wird. Welche Werte für C und Epsilon liefern das beste Ergebnis?
2. Für das SVR-Objekt können die Koeffizienten der linearen Abbildung, welche durch die trainierte SVR realisiert wird, ausgegeben werden: `meineSVR.coef_`. Notieren Sie diese Koeffizienten für die beste SVR.
3. Welchen Aufschluss geben diese Koeffizienten über den Einfluss der einzelnen Eingangsmerkmale auf das Ausgangsmerkmal?
4. Wie groß ist die mittlere absolute Differenz zwischen Soll- und Ist-Ausgabe für die beste SVR? Diskutieren Sie dieses Ergebnis.

Abzugeben sind:

1. Das Programm `energyPrediction.py`
2. Antworten auf die oben gestellten Fragen.

2.4 Visualisierung des Clusterings in Google Maps

In Abschnitt 2.2.1 wurden die gefundenen Clusterindizees im File `EnergyMixGeo.csv` abgespeichert. In diesem File befinden sich auch die in der ersten Teilaufgabe dieses Versuchs berechneten Geokoordinaten der Länder. Jetzt, sollen die geclusterten Länder wie in Abbildung 2.4 dargestellt, in Google Maps angezeigt werden, d.h. an die Geokordinaten aller Länder wird ein Icon platziert, wobei für Länder unterschiedlicher Cluster auch unterschiedliche Icons verwendet werden sollen.

Schreiben sie ein Python-Programm `clusters2GoogleMaps.py`, das die Daten aus `EnergyMixGeo.csv` einliest und die Länder mit den clusterspezifischen Icons in Google Maps anzeigt. Beim Klick auf ein Icon, sollen die Attributwerte des entsprechenden Landes angezeigt werden. Verwenden sie hierfür die in `pymaps.py` (kann vom Skripteserver heruntergeladen werden) bereitgestellten Funktionen. Wie diese Funktionen verwendet werden geht aus dem in der main-Methode des Programms `pymaps.py` implementierten Beispiel hervor.

Abzugeben sind:

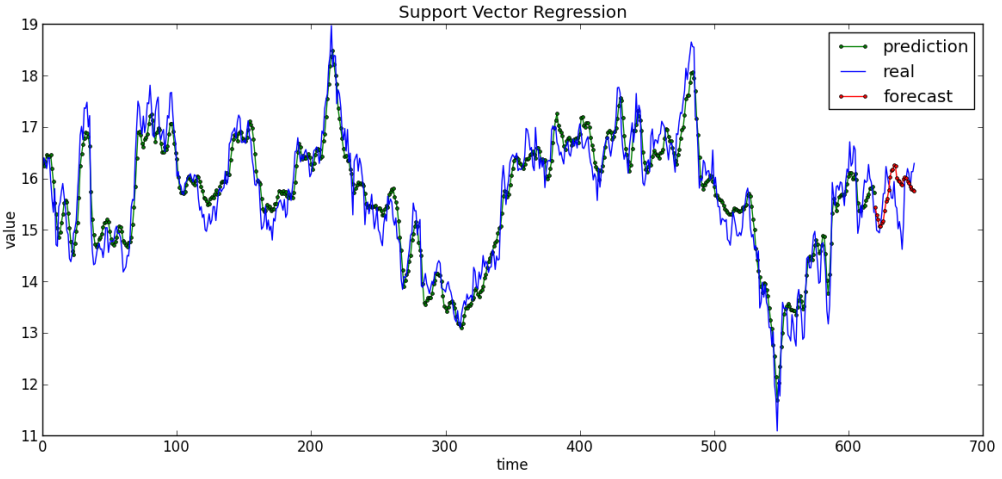
1. Das Programm `clusters2GoogleMaps.py`
2. Das von Ihnen erstellte Html-File mit den geclusterten Ländern in Googlemaps.


Map
Satellite
Hybrid
Greenland
Iceland
Finland
Sweden
Norway
Canada
Russia
Kazakhstan
Mongolia
China
South Korea
United States
North Atlantic Ocean
North Pacific Ocean
Turkey
France
Italy
México
South_Africa: Oil=24.3 Gas: 0.0 Coal: 99.4 Nuclear: 2.7 Hydro: 0.2 Total 2009: 126.8
Perú
Bolivia
Chile
Argentina
DR Congo
Angola
Namibia
Botswana
South Africa
Madagascar
Indian Ocean
South Atlantic Ocean
Indonesia
Papua New Guinea
Australia
South Pacific Ocean
POWERED BY Google
Map data ©2012 MapLink, Tele Atlas - Terms of Use

3 Durchführung Teil 2: Vorhersage und Clustering auf Finanzdaten

3.1 Zeitreihenschätzung: Vorhersage des Aktienkurses

Bei der Zeitreihenschätzung wird von zeitlich geordneten und korrelierten Datensätzen ausgegangen. Z.B. könnten als Trainingsdaten die Kurswerte einer Aktie aus den vergangenen X Tagen herangezogen werden. Ziel ist es dann aus diesen Trainingsdaten den Verlauf der Funktion, d.h. hier des Aktienkurses, in der nächsten Zukunft zu schätzen. Im unten dargestellten Plot ist z.B. der tatsächliche Kursverlauf einer Aktie über 650 Tage durch die blaue Linie eingetragen. Aus diesem Verlauf der vergangenen 600 Tage wird ein Modell berechnet, das den Verlauf des Aktienkurses für die nächsten etwa 50 Tage in die Zukunft vorhersagt. Diese Schätzung ist durch rote Marker dargestellt.



Für die Modellbildung im Fall einer Zeitreihe, kann im Prinzip jedes gewöhnliche Regressionsverfahren (wie z.B. die SVR) benutzt werden. Die Datensätze sind dabei zeitlich geordnet, d.h. der erste Datensatz gehört zur Zeit T, der zweite zur Zeit $T+1$, ... der i.te zur Zeit $T+i-1$. Der i.te Datensatz hat als

- **Ausgangsattribut:** Den Funktionswert an der Stelle $T+i$.
- **Eingabeattribute:** Die w vorhergehenden Funktionswerte, also die Funktionswerte zu den Zeiten $T+i-w$, $T+i-w+1$, ..., $T+i-1$.

D.h. die Anzahl der Eingabeattribute entspricht der Anzahl der vergangenen Werte, von denen man annimmt, dass Sie Einfluss auf den aktuellen Wert haben. Der unten dargestellte Tabellenausschnitt zeigt ein Beispiel für derart zeitlich geordnete Daten. Die letzte Spalte enthält jeweils den zur aktuellen Zeit gehörenden Funktionswert. Die vier anderen Spalten enthalten jeweils die 4 vorhergehenden Funktionswerte als Eingabeattribute.

T-4	T-3	T-2	T-1	T class
31.61	32.09	28.31	27.51	27.46
32.09	28.31	27.51	27.46	27.88
28.31	27.51	27.46	27.88	28.02
27.51	27.46	27.88	28.02	28.06
31.41	31.61	32.09	28.31	27.51
27.46	27.88	28.02	28.06	28.49

3.1.1 Datenbeschaffung

Laden Sie das Fragment `b101_stockMarketFragment.py` vom Skripteserver und führen Sie es aus. Das Programm holt sich den Kursverlauf aller definierten Aktien zwischen April 2009 und März 2012. In dem Fragment werden die Daten in dictionaries abgelegt. Speichern Sie dann eine Kopie des Fragments unter dem Namen `b101_stockMarketFile.py`. Erweitern Sie die Kopie wie folgt:

1. Im bereitgestellten Fragment werden für jeden Tag und für jedes Unternehmen 3 verschiedene Werte aufgezeichnet:
   - Der Kurs beim Öffnen der Börse (*openVals*)
   - Der Kurs beim Börsenschluss (*closeVals*)
   - Der effektive Kurs beim Börsenschluss (*prices*). Im effektiven Kurs sind die Daten in dem Sinne angepasst, dass Sprünge im Kursverlauf, die durch Splittings oder Dividenden verursacht werden, bereits ausgeglichen sind.
2. Schreiben Sie die Daten des Dictionaries, das die effektiven Kurse enthält in einen *Pandas Dataframe*. Jede Spalte des Dataframes gehört dabei zu einem Unternehmen und enthält die Aktienkurse über den gesamten Beobachtungszeitraum. Der Index des Dataframes ist die Datumsliste. D.h. die i.te Zeile des Dataframes enthält die Aktienkurse am i.ten Datum.
3. Schreiben Sie den angelegten Dataframe in ein File `effectiveRates.csv`.

Abzugeben sind:

1. Das Programm `b101_stockMarketFile.py`

3.1.2 Kursvorhersage mit SVR

In dieser Teilaufgabe soll auf Basis der im File `effectiveRates.csv` abgelegten Daten eine Vorhersage des Aktienkurses durchgeführt werden. Schreiben Sie ein Python-Programm `b102_stockMarketPrediction.py` mit folgender Funktionalität:

1. Einlesen der Datei `effectiveRates.csv` in einen Pandas Dataframe.
2. Graphische Anzeige von 5 Aktienkurse Ihrer Wahl mit der Plot-Funktion für Pandas-Dataframes.
3. Festlegen des Unternehmens, für das der Aktienkurs vorhergesagt wird. Aus Gründen der Vergleichbarkeit, sollte hier *Yahoo!* ausgewählt werden.

4. Für das ausgewählte Unternehmen sollen die seriellen Daten in eine zyklische Form, wie sie im oben abgebildeten Tabellenausschnitt dargestellt ist, transformiert werden. Dabei soll der Time-Delay Parameter, also die Anzahl der zu berücksichtigenden vorhergehenden Kurswerte, als Parameter frei einstellbar sein. Als Grundeinstellung wird ein Time-Delay von 24 empfohlen. Dieser soll jedoch später optimiert werden.

5. Festlegen des Trainingszeitraum und des Vorhersagezeitraum. Empfohlen wird für das Training die ersten 650 Instanzen der zyklisch repräsentierten Daten zu verwenden. Als Vorhersagezeitraum kann ein Wert von 30 Tagen voreingestellt werden. **Wichtig: Lösen Sie zunächst die erste der unten aufgeführten Fragen. Diskutieren Sie Ihre Lösung vor der Implementierung mit dem Betreuer.**

6. Anlegen einer SVR mit RBF-Kernel (empfohlene Parameter: $C = 500, epsilon = 0.6$), Training der SVR und Berechnung der Vorhersagen.

7. Durch Vergleich der Vorhersagewerte und der Sollwerte soll der MAE (Mean Absolute Error) berechnet werden.

8. Darstellung des tatsächlichen Kursverlaufs über den gesamten Zeitbereich (Trainings- und Vorhersageperiode) und der vorhergesagten Werte in einem Matplotlib-Plot (wie in der obigen Abbildung).

Fragen:

1. Überlegen Sie sich genau, wie die Datenvektoren des Vorhersagezeitraums (also die Vektoren, die der Methode *prediction()* des trainierten SVR-Modells übergeben werden), aufgebaut werden müssen.

2. Für welche Werte von

   - Time Delay
   - SVR-Parameter C
   - SVR-Parameter *epsilon*

   erreichen Sie die beste Vorhersage? Wie groß ist in diesem Fall der MAE? Erzeugen Sie für diese optimierten Werte den Plot des tatsächlichen und des geschätzten Kursverlaufs und speichern Sie diese Grafik unter dem Namen `stockpredict.png`.

Abzugeben sind:

1. Das Programm `b102_stockMarketPrediction.py`
2. Das erzeugte Grafik `stockpredict.png`
3. Die Antworten auf die o.g. Fragen.

3.2 Clustering der Aktienkursverläufe

In diesem Teilversuch soll eine Menge von 61 Unternehmen anhand Ihrer Aktienkursverläufe geclustert werden. Dabei soll untersucht werden, ob Unternehmen, die aufgrund ähnlicher Kursverläufe in ein gemeinsames Cluster fallen, irgendwie semantisch korrelieren (zusammengehören).

Betrachtet werden sollen genau die Unternehmen, für die in der ersten Teilaufgabe die Kursdaten im File `effectiveRates.csv` angelegt wurden. Allerdings eignen sich die in `effectiveRates.csv` abgelegten Daten für die aktuelle Clusteraufgabe nicht, weil jetzt sowohl der Close-Kurs als auch der Openkurs in

bereinigter Form vorliegen muss. Bereinigt meint dabei, dass Sprünge im Kursverlauf, die durch Splittings oder Dividenden verursacht werden, bereits ausgeglichen sind. Derart bereinigte Daten werden schon im Matplotlib-Beispielpaket *finance* bereitgestellt. Benutzen Sie das auf dem Skripteserer bereitgestellte Codefragment `b103_stockMarketClusteringFragment.py` um diese bereinigten Daten zu laden. Speichern Sie dann eine Kopie des Fragments unter dem Namen `b103_stockMarketClustering.py`. Erweitern Sie die Kopie wie folgt:

1. Berechnung der Differenz zwischen Open und Close für jedes Unternehmen. Es wird angenommen, dass diese Differenz ein gutes Maß für die Bestimmung der Ähnlichkeit zwischen Kursverläufen darstellt.

2. Berechnung einer Ähnlichkeitsmatrix zwischen allen Unternehmen. Der Eintrag in Zeile i, Spalte j dieser Ähnlichkeitsmatrix gibt den Ähnlichkeitswert zwischen dem i.ten und j.ten Unternehmen an. Der Ähnlichkeitswert wird berechnet indem die Pearson-Korrelation auf die Kursdifferenzvektoren (siehe vorige Aufgabe) der beiden Unternehmen angewandt wird. Die Pearson-Korrelation wird z.B. durch die Numpy-Funktion *corrcoef()* berechnet. Diese Methode kann auf ein Array mit allen Differenzvektoren angewandt werden und liefert dann gleich die gewünschte Ähnlichkeitsmatrix.

3. Die Ähnlickeitsmatrix wird der *fit()* methode eines Objekts der scikit-learn Klasse *AffinityPropagation* übergeben. Damit wird ein Clustering berechnet. Auf die gefundenen Clusterlabels kann über das Objektattribut *labels_* zugegriffen werden.

4. Für jedes der gefundenen Cluster sollen alle zum Cluster gehörenden Kursverläufe in einem Plot pro Cluster graphisch dargestellt werden.

Fragen:

1. Analysieren Sie die Clusterzuweisungen. Gehören die Unternehmen, welche in einem Cluster zusammenfallen, irgendwie zusammen?

Abzugeben sind:

1. Das Programm `b103_stockMarketClustering.py`

2. Die Antwort auf die o.g. Frage.

Literatur

[EA] Ethem Alpaydin; Maschinelles Lernen; Deutsche Übersetzung im Oldenbourg Verlag; München, 2008

[OR] Scikit-Learn; Machine Learning Library in Python; `http://scikit-learn.org/stable//`

[MKI] Johannes Maucher; Vorlesung Einführung in die Künstliche Intelligenz - Maschinelles Lernen Teil 1; `http://www.hdm-stuttgart.de/~maucher/Lecture_KI.html`