Analytik in der Datenbank

Modellanwendung näher an den Daten

Balázs Bárány

Berater für Analytik und Data Warehousing

Linuxtage Graz 2014

Inhalt

Analytik bzw. analytics

- Deskriptiv
- Prädiktiv
- Explorativ

Analytik bzw. analytics

- Deskriptiv
- Prädiktiv
- Explorativ

- Statistik, Data Mining, KDD, Machine Learning, Data Science, ...?

Analytik bzw. analytics

- Deskriptiv
- Prädiktiv
- Explorativ

- Statistik, Data Mining, KDD, Machine Learning, Data Science, ...?

- Anwendungsfelder
  - Klassifizierung, Regression, Assoziation, Clustering

Methoden und Werkzeuge

- Verfahren und Algorithmen
  - Informiert
  - Uninformiert

Methoden und Werkzeuge

- Verfahren und Algorithmen
  - Informiert
  - Uninformiert
- Software
  - Open Source: R, RapidMiner, Weka, KNIME, ...
  - Proprietär: SPSS, SAS, ...

Freie Werkzeuge

- RapidMiner
  - „Commercial source model"
  - Community-Version von Sourceforge.net und Github
  - Aktuelle Version 6: interaktives Tutorial, Usability-Verbesserungen

Freie Werkzeuge

- RapidMiner
  - „Commercial source model"
  - Community-Version von Sourceforge.net und Github
  - Aktuelle Version 6: interaktives Tutorial, Usability-Verbesserungen
- R „environment for statistical computing and graphics"
  - Befehlszeile, Skripting
  - GUIs

Data-Mining-Prozess

Cross Industry Standard Process for Data Mining, Kenneth Jensen/Wikimedia Commons

Themen dieses Vortrags

Cross Industry Standard Process for Data Mining, Kenneth Jensen/Wikimedia Commons

Projekt in der Praxis

- Daten liegen in einer relationalen Datenbank
- Analyse, Modellerstellung, Evaluierung in der Analysesoftware

Projekt in der Praxis

- Daten liegen in einer relationalen Datenbank
- Analyse, Modellerstellung, Evaluierung in der Analysesoftware
- Ziel: Anwendung des Modells auf neue Daten

Projekt in der Praxis

- Daten liegen in einer relationalen Datenbank
- Analyse, Modellerstellung, Evaluierung in der Analysesoftware
- Ziel: Anwendung des Modells auf neue Daten
  - Neue Daten lesen, ID-Feld zwischenspeichern
  - Modell anwenden
  - Ergebnis zurückschreiben

Projekt in der Praxis

- Daten liegen in einer relationalen Datenbank
- Analyse, Modellerstellung, Evaluierung in der Analysesoftware
- Ziel: Anwendung des Modells auf neue Daten
  - Neue Daten lesen, ID-Feld zwischenspeichern
  - Modell anwenden
  - Ergebnis zurückschreiben
  - Probleme:
    - Ineffiziente Roundtrips
    - Interaktion zwischen mehreren Systemen
    - Batch-Verarbeitung, Latenz

Integration in die Datenerfassung

- Etwas besser:
  - Integrierte Software

Integration in die Datenerfassung

- Etwas besser:
  - Integrierte Software
    - Ablösung des Legacy-Systems?

Integration in die Datenerfassung

- Etwas besser:
  - Integrierte Software
    - Ablösung des Legacy-Systems?
  - Integration über offene Schnittstellen
    - Webservices, remote procedure calls, ...

Integration in die Datenerfassung

- Etwas besser:
  - Integrierte Software
    - Ablösung des Legacy-Systems?
  - Integration über offene Schnittstellen
    - Webservices, remote procedure calls, ...
    - Predictive Model Markup Language (PMML)

Integration in die Datenerfassung

- Etwas besser:
  - Integrierte Software
    - Ablösung des Legacy-Systems?
  - Integration über offene Schnittstellen
    - Webservices, remote procedure calls, ...
    - Predictive Model Markup Language (PMML)
- Integration in den Datenbankserver

Modellierungsverfahren

- Lineare Regression
  - Gleichung in der Form $y = ax + b$

Modellierungsverfahren

- Lineare Regression
  - Gleichung in der Form $y = ax + b$
  - Erweiterungen: nicht normalverteilte Daten, nichtlinear, multivariat

Modellierungsverfahren

- Lineare Regression
  - Gleichung in der Form $y = ax + b$
  - Erweiterungen: nicht normalverteilte Daten, nichtlinear, multivariat
- Naive-Bayes-Methode

Modellierungsverfahren

- Lineare Regression
  - Gleichung in der Form $y = ax + b$
  - Erweiterungen: nicht normalverteilte Daten, nichtlinear, multivariat
- Naive-Bayes-Methode
- Entscheidungsbäume

Modellierungsverfahren

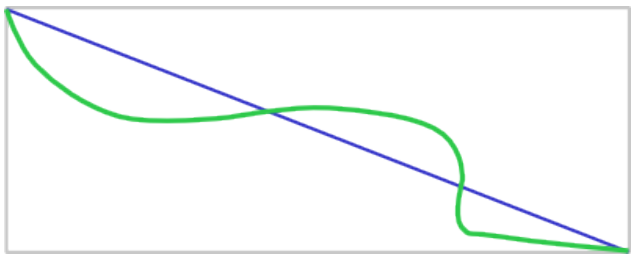
- Lineare Regression
  - Gleichung in der Form $y = ax + b$
  - Erweiterungen: nicht normalverteilte Daten, nichtlinear, multivariat
- Naive-Bayes-Methode
- Entscheidungsbäume
- Support Vector Machines

Modellierungsverfahren

- Lineare Regression
  - Gleichung in der Form $y = ax + b$
  - Erweiterungen: nicht normalverteilte Daten, nichtlinear, multivariat
- Naive-Bayes-Methode
- Entscheidungsbäume
- Support Vector Machines
- ... und viele mehr

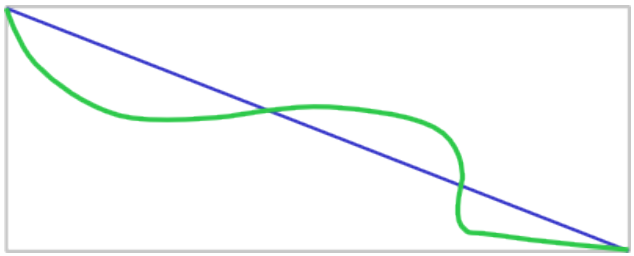
Beispiel: lineare Regression

- Fragestellung: Höchstgeschwindigkeit von „Kurvigkeit" des Straßenabschnitts abhängig?
- Methode: Länge des Straßenabschnitts / Länge der Geraden zwischen Start- und Endpunkt



Beispiel: lineare Regression

- Fragestellung: Höchstgeschwindigkeit von „Kurvigkeit" des Straßenabschnitts abhängig?
- Methode: Länge des Straßenabschnitts / Länge der Geraden zwischen Start- und Endpunkt



Beispiel

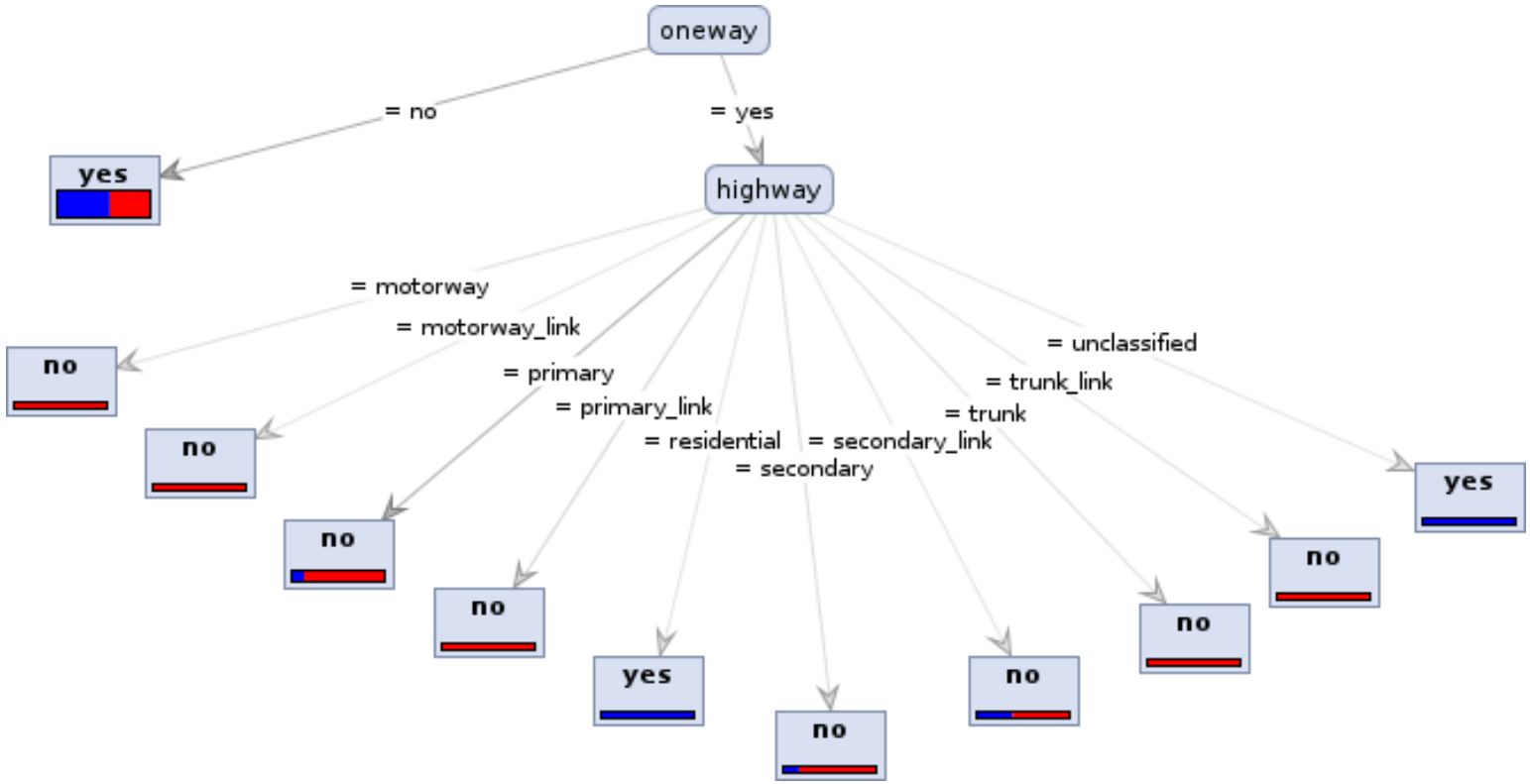
```
> mod <- lm(geschw ~ kurvigkeit, data=strkurv)
> coef(mod)
(Intercept)    kurvigkeit
  86.14846      -63.79144
```

Beispiel: Entscheidungsbaum

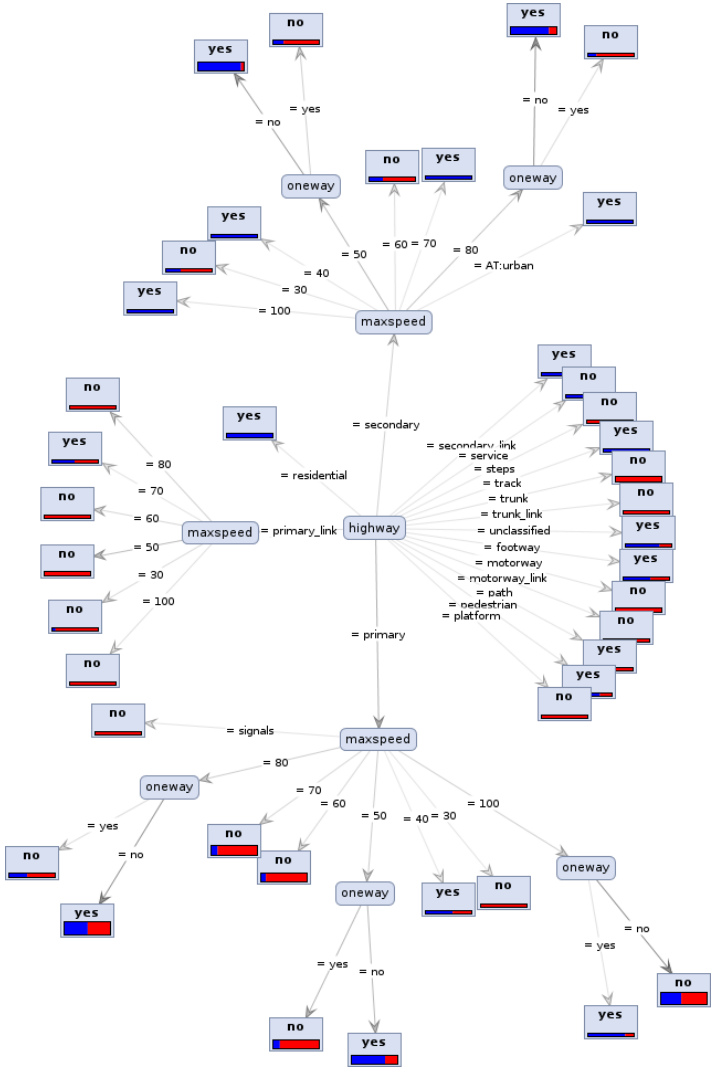
- Fragestellung: Darf die Straße mit Fahrrädern befahren werden?
- Methode: Entscheidungsbaum in RapidMiner

Beispiel: Entscheidungsbaum

- Fragestellung: Darf die Straße mit Fahrrädern befahren werden?
- Methode: Entscheidungsbaum in RapidMiner



Komplexer Entscheidungsbaum



genauer, aber viel komplexer!

Modelle in SQL-Datenbanken

- Lineare Regression und Varianten: trivial
  - Parameter eintragen, View oder Stored Procedure erstellen und aktualisieren

Beispiel

```
CREATE VIEW maxspeed_prediction AS
SELECT osm_id,
  GREATEST(
    86.14846 + -63.79144 * (
      ST_Length(geo) / ST_Distance(ST_StartPoint(geo), ST_EndPoint(geo)) - 1),
     20.0
  ) AS maxspeed_pred
FROM österreich_strasse
WHERE maxspeed is null
   AND ST_Distance(ST_StartPoint(geo), ST_EndPoint(geo)) > 0
```

Variante für einfachere Wartung

- View-Lösung nicht leicht automatisch zu aktualisieren
- Besser: Parameter-Tabelle und Funktion

Beispiel

```
CREATE TABLE maxspeed_pred_param (
  intercept double precision, kurvigkeit double precision
);


CREATE FUNCTION kurvigkeit ... ;


SELECT osm_id, GREATEST(20.0,
    mpp.intercept + mpp.kurvigkeit * kurvigkeit(way)
  ) AS maxspeed_pred
FROM osm_austria_roads
CROSS JOIN maxspeed_pred_param mpp ;
```

Regelbasierte Modelle

- Entscheidungsbäume und Rule-Modelle in SQL
  - Baum: Pfade verfolgen, Entscheidungen in CASE WHEN ... THEN überführen

Regelbasierte Modelle

- Entscheidungsbäume und Rule-Modelle in SQL
  - Baum: Pfade verfolgen, Entscheidungen in CASE WHEN ... THEN überführen
  - „Tree to Rules"-Operator in RapidMiner

Beispiel

1 if oneway = no then yes (608 / 439)
2 if oneway = yes and highway = motorway then no (0 / 32)
3 if oneway = yes and highway = motorway_link then no (0 / 5)
4 if oneway = yes and highway = primary then no (44 / 256)
5 if oneway = yes and highway = primary_link then no (2 / 82)
6 if oneway = yes and highway = residential then yes (5 / 0)
7 if oneway = yes and highway = secondary then no (11 / 47)
8 if oneway = yes and highway = secondary_link then no (2 / 3)
9 if oneway = yes and highway = trunk then no (0 / 28)
10 if oneway = yes and highway = trunk_link then no (0 / 11)
11 if oneway = yes and highway = unclassified then yes (2 / 0)

Regeln in SQL überführt

- Syntaxänderungen für SQL

Beispiel

```
SELECT osm_id,
  CASE WHEN oneway = 'no' THEN true
     WHEN oneway = 'yes' AND highway = 'motorway' THEN false
     WHEN oneway = 'yes' AND highway = 'residential' THEN true
     WHEN ...
     ELSE null
  END as fahrrad_erlaubt
FROM osm_austria_roads;
```

Wartung aufwändig und fehleranfällig!

Komplexe Modelle

- komplexe Datenstrukturen und Berechnungen
  - schwer in SQL abzubilden, Wartungsaufwand bei Änderungen

Komplexe Modelle

- komplexe Datenstrukturen und Berechnungen
  - schwer in SQL abzubilden, Wartungsaufwand bei Änderungen
- Muß es SQL sein?
  - PostgreSQL ist multilingual
  - PL/Perl, PL/Python, PL/JavaScript, PL/Java ...

Komplexe Modelle

- komplexe Datenstrukturen und Berechnungen
  - schwer in SQL abzubilden, Wartungsaufwand bei Änderungen
- Muß es SQL sein?
  - PostgreSQL ist multilingual
  - PL/Perl, PL/Python, PL/JavaScript, PL/Java ...
  - **PL/R**

Überblick: PL/R

- Erstes Release 2003

Überblick: PL/R

- Erstes Release 2003
  - Oracle: 2010 (mind. 23.000 $)

Überblick: PL/R

- Erstes Release 2003
  - Oracle: 2010 (mind. 23.000 $)
- Aktuelle Version 8.3.0.15 für PostgreSQL 9.3

Überblick: PL/R

- Erstes Release 2003
  - Oracle: 2010 (mind. 23.000 $)
- Aktuelle Version 8.3.0.15 für PostgreSQL 9.3
- Lizenz: GPL

Sicherheitsüberlegungen

- Installation als „untrusted language”

Sicherheitsüberlegungen

- Installation als „untrusted language”
- Funktionen erstellen nur als Datenbankadministrator

Sicherheitsüberlegungen

- Installation als „untrusted language”
- Funktionen erstellen nur als Datenbankadministrator
- Freigabe der Ausführung für unprivilegierte Benutzer

Sicherheitsüberlegungen

- Installation als „untrusted language”
- Funktionen erstellen nur als Datenbankadministrator
- Freigabe der Ausführung für unprivilegierte Benutzer

Beispiel

```
# Mit Admin-Rechten
CREATE FUNCTION r_add(zahl1 double precision, zahl2 double
precision)
    RETURNS double precision AS $func$
        return(zahl1 + zahl2)
    $func$
LANGUAGE 'plr';
GRANT EXECUTE ON FUNCTION r_add(double precision, double
precision)
      TO normaluser;
```

Funktionalität

- Aggregations- und Window-Funktionen

Funktionalität

- Aggregations- und Window-Funktionen
- Trigger-Funktionen in R

Funktionalität

- Aggregations- und Window-Funktionen
- Trigger-Funktionen in R
- Datenbankabfragen, PostgreSQL-Umgebung

Funktionalität

- Aggregations- und Window-Funktionen
- Trigger-Funktionen in R
- Datenbankabfragen, PostgreSQL-Umgebung
- Gleichnamige R-Funktion ohne Funktions-Body einbinden

Beispiel

```
# Liefert n normalverteilte Zufallszahlen
CREATE FUNCTION rnorm(n integer, mean double precision, sd
double precision)
RETURNS double precision[]
AS ”
LANGUAGE ’plr’;
```

Implementierung einer Vorhersagefunktion

Vorbereitungen

- Benötigte Funktionen im R-Interpreter anlegen

Implementierung einer Vorhersagefunktion

Vorbereitungen

- Benötigte Funktionen im R-Interpreter anlegen

einmalig:

Beispiel

```
select install_rcmd('
    meinefunktion <-function(x)
      {print(x)}
');
```

Implementierung einer Vorhersagefunktion

Vorbereitungen

- Benötigte Funktionen im R-Interpreter anlegen

einmalig:

Beispiel

```
select install_rcmd('
    meinefunktion <-function(x)
      {print(x)}
');
```

permanent: Tabelle `plr_modules`

Implementierung einer Vorhersagefunktion

Vorbereitungen

- Benötigte Funktionen im R-Interpreter anlegen

einmalig:

Beispiel

```
select install_rcmd('
    meinefunktion <-function(x)
      {print(x)}
');
```

permanent: Tabelle `plr_modules`

- Funktionskontext vs. globaler Kontext

Daten und Modelle in PL/R bringen

- Entwicklung des Modells am Desktop
- Speichern: `save(objekt, file="/pfad/zu/objekt.Rdata")`

Daten und Modelle in PL/R bringen

- Entwicklung des Modells am Desktop
- Speichern: `save(objekt, file="/pfad/zu/objekt.Rdata")`
- (Übertragung auf den Server oder gemeinsames Dateisystem)

Daten und Modelle in PL/R bringen

- Entwicklung des Modells am Desktop
- Speichern: `save(objekt, file="/pfad/zu/objekt.Rdata")`
- (Übertragung auf den Server oder gemeinsames Dateisystem)
- Laden in den Datenbankserver
  - `load("/pfad/zu/objekt.Rdata", .GlobalEnv)`

Daten und Modelle in PL/R bringen

- Entwicklung des Modells am Desktop
- Speichern: `save(objekt, file="/pfad/zu/objekt.Rdata")`
- (Übertragung auf den Server oder gemeinsames Dateisystem)
- Laden in den Datenbankserver
  - `load("/pfad/zu/objekt.Rdata", .GlobalEnv)`
  - Beim Start aus der plr_modules-Tabelle

Beispiel

```
for (datafile in Sys.glob("/pfad/*.Rdata")) {
    load(datafile, .GlobalEnv)
}
```

Anwendung des Modells in der Datenbank

Beispiel

```
CREATE OR REPLACE FUNCTION bicycle_vorhersage(
        highway TEXT, maxspeed TEXT, oneway TEXT
) RETURNS TEXT AS $func$
  daten <- data.frame(bicycle=NA,
              highway=highway, maxspeed=maxspeed, oneway=oneway)
  bicycle <- predict(bicycle_mod_modell, newdata=daten)
  return(as.character(bicycle))
$func$ LANGUAGE 'plr';
```

Anwendung des Modells in der Datenbank

Beispiel

```
CREATE OR REPLACE FUNCTION bicycle_vorhersage(
        highway TEXT, maxspeed TEXT, oneway TEXT
) RETURNS TEXT AS $func$
  daten <- data.frame(bicycle=NA,
              highway=highway, maxspeed=maxspeed, oneway=oneway)
  bicycle <- predict(bicycle_mod_modell, newdata=daten)
  return(as.character(bicycle))
$func$ LANGUAGE 'plr';
```

- Anwendung:

Beispiel

```
select bicycle_vorhersage('residential', '70', 'yes');
no
select bicycle_vorhersage('residential', '30', 'no');
yes
```

Vorhersage in der Tabelle nachtragen

Beispiel

```
UPDATE osm_austria_roads
SET bicycle = bicycle_vorhersage(coalesce(highway, 'NA'),
        coalesce(maxspeed, 'NA'), coalesce(oneway, 'NA'))
WHERE highway = 'residential'
  AND oneway = 'yes'
  AND bicycle IS NULL;
```

Probleme

- character <> factor

Probleme

- character <> factor
- NULL <> NA

Probleme

- character <> factor
- NULL <> NA
- Datenbankzugriff aus PL/R

Probleme

- character <> factor
- NULL <> NA
- Datenbankzugriff aus PL/R
- allgemein: viel Ausprobieren und Übung

Ausblick

- RapidMiner mit PL/Java

Ausblick

- RapidMiner mit PL/Java
- Python-basierte Modelle mit PL/Python

Ausblick

- RapidMiner mit PL/Java
- Python-basierte Modelle mit PL/Python
- Direkte Erweiterungen in der Datenbank
- PMML-Unterstützung

Fragen?

- Balázs Bárány, <balazs@tud.at>