# Projektbeschreibung

* Titel: Predicting Bike Rental Demand
* Kaggle Competition: [Bike Sharing Demand.Forecast use of a city bikeshare system](https://www.kaggle.com/c/bike-sharing-demand)
* 2014 gestartet
* Was es tut: Vorhersage der Anzahl der gemieteten Fahrräder je nach Wetter und Zeit
* Warum nützlich:

"The data generated by these systems makes them attractive for researchers because the duration of travel, departure location, arrival location, and time elapsed is explicitly recorded. Bike sharing systems therefore function as a sensor network, which can be used for studying mobility in a city. In this competition, participants are asked to combine historical usage patterns with weather data in order to forecast bike rental demand in the Capital Bikeshare program in Washington, D.C."

* Datensätze
  + Messungen 2011 - 2012 (2 Jahre)
  + Jeden Tag zählen stündlich die Anzahl der gemieteten Fahrräder zählen und dabei unabhängige Variablen wie Temperatur, Luftfeuchtigkeit, Windgeschwindigkeit, Tag, Jahreszeit usw. berücksichtigen
  + Aufgeteilt in
    - Train
      * Erster bis 19. Tag eines jeden Monats
      * Etwa 1100 Zeilen
    - Test
      * Vom 20. Tag bis Monatsende
      * Etwa 6500 Zeilen
* Bewertung: Root Squared Mean Log Error - Wurzelquadrat des mittleren logarithmischen Fehlers
  + Bewertungsmaßstab zur Messung der Differenz zwischen vorhergesagtem Ergebnis und tatsächlichem Ergebnis
  + Je kleiner, desto besser

<https://medium.com/analytics-vidhya/root-mean-square-log-error-rmse-vs-rmlse-935c6cc1802a>

# Motivation

* Kurs "Machine Learning for Coders"

<https://course18.fast.ai/ml.html>

* Unterrichtet von Jeremy Howard
* Aufgezeichnete Vorlesungen aus dem Master of Data Science Studiengang an der Universität von San Francisco im Jahr 2018
* ML Lesson 1: Introduction to Random Forests

[Intro to Macne Learning: Lesson 1](https://youtu.be/CzdWqFTmn0Y)

Ein Bild, das Tisch enthält.

Automatisch generierte Beschreibung

* Vorhersage von Verkaufspreisen für Planierraupen mit Python Random Forest Regressor

* Gründe für die Wahl der "Bike" Competition
  + Einsteigerfreundliches Projekt ähnlich wie Lektion 1
  + Aufgabe: Vorhersage eines numerischen y-Wertes aus numerischen x-Werten
  + Wurzelquadrat des mittleren logarithmischen Fehlers

# Methode

## Entscheidungsbaum

[Decision Tree Regression](https://www.saedsayad.com/decision_tree_reg.htm#:~:text=Decision%20tree%20builds%20regression%20or,decision%20nodes%20and%20leaf%20nodes)

.

[](https://www.crashkurs-statistik.de/variationskoeffizient/)

[Variationskoeffizient](https://www.crashkurs-statistik.de/variationskoeffizient/)

[Underfitting and Decision Trees](https://medium.com/expedia-group-tech/underfitting-and-decision-trees-72568163d702)

Wie **funktioniert das? - Standard Deviation Reduction**

Berechne die *Standardabweichung* der abhängigen Werte

Für jede unabhängige Variable wird ein Teilsatz aus dem Datensatz erstellt und nach deren Ausprägungen geordnet.

Berechne für jeden Teilsatz die Standardabweichung der abhängigen Werte

Ziehe die Standardabweichung des Teilsatzes von der Standardabweichung aller abhängigen Werte ab. Das ist die *Standard Deviation Reduction*.

Die unabhängige Variable, die dabei die größte Differenz hat, wird die erste Ebene im Baum nach der Wurzel. Nach der teilen wir den gesamten Datensatz und wiederholen das so lange, bis alle Daten verarbeitet sind

Jeder Pfad wird so lange weiter aufgeteilt, bis ein bestimmter Schwellenwert erreicht ist (z.B. *Coefficent of Deviation* < 10%)

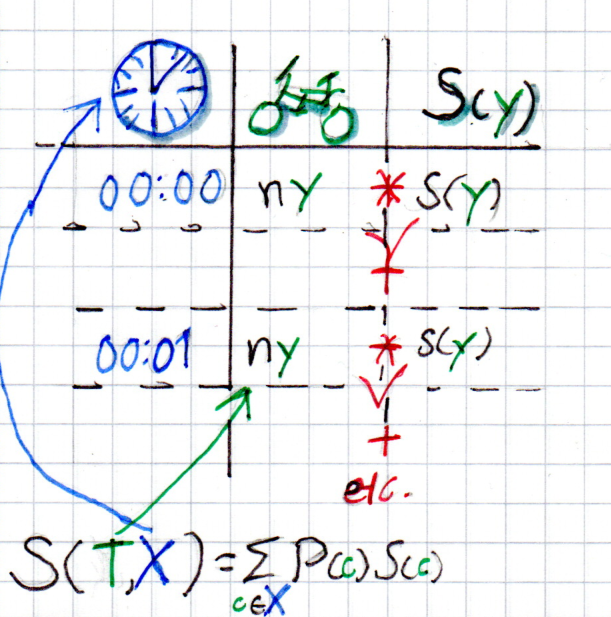
Aus <[*https://trello.com/c/UosNR4jI/95-wie-funktioniert-ein-vorhersagebaum*](https://trello.com/c/UosNR4jI/95-wie-funktioniert-ein-vorhersagebaum)>

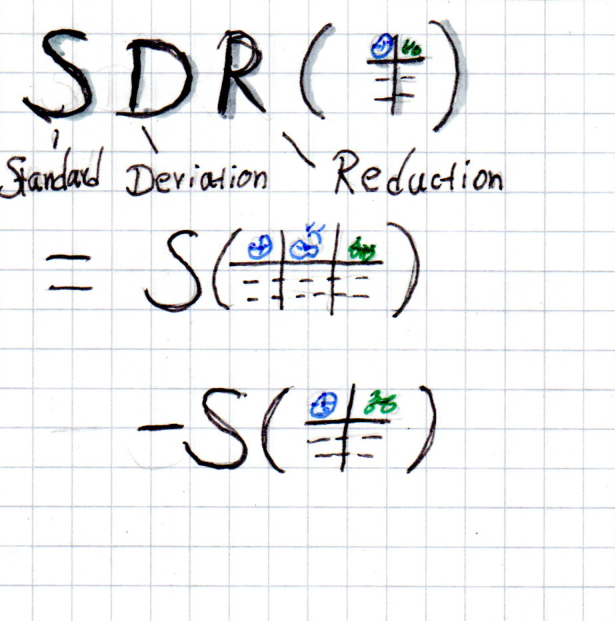
Ein Bild, das Text, Whiteboard enthält.

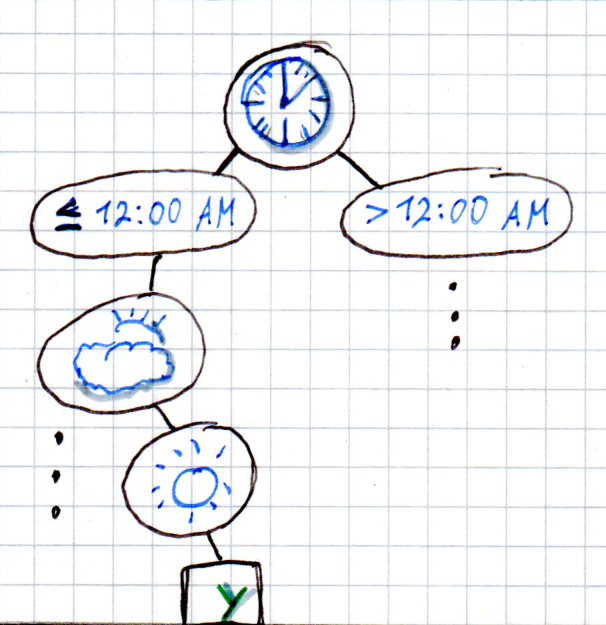
Automatisch generierte Beschreibung

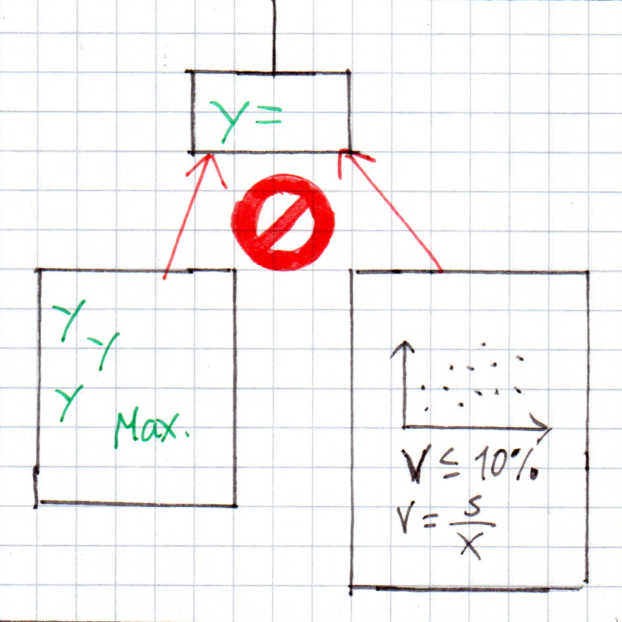
Ein Bild, das Text, Whiteboard enthält.

Automatisch generierte Beschreibung





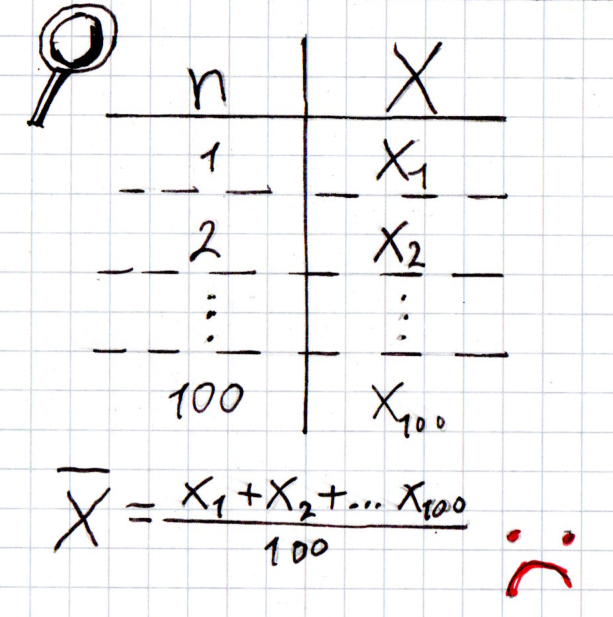


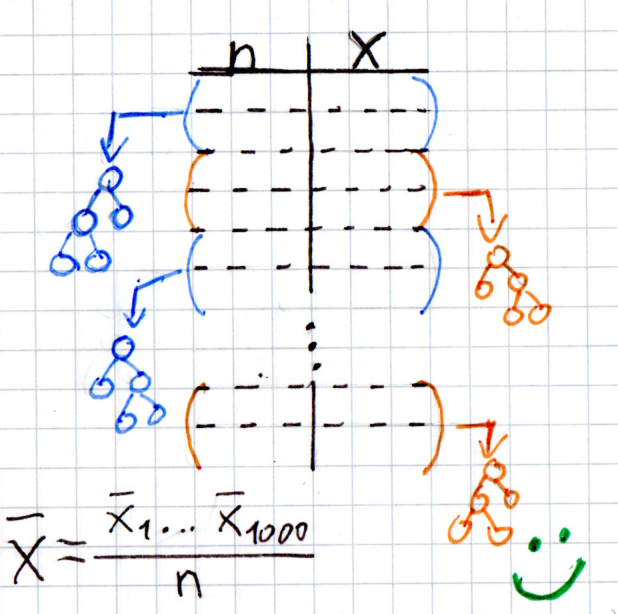


## Bootstrapping

<https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/>

* Methode, um Größen des Datensatzes zu schätzen (z.B. Mittelwert)
* Vorteil gegenüber Mittelwert: Fehlervermeidung
* Datensatz in viele Untersätze teilen (z.B. 1000 Subsets bei n=100)
* Für jedes Subset den Mittelwert berechnen
* Der Mittelwert aller Mittelwerte als Mittelwert des gesamten Datensatzes





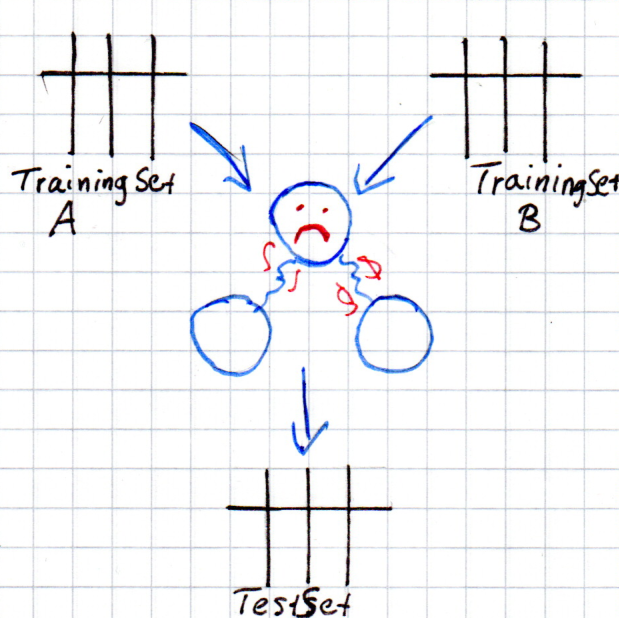
## Bagging

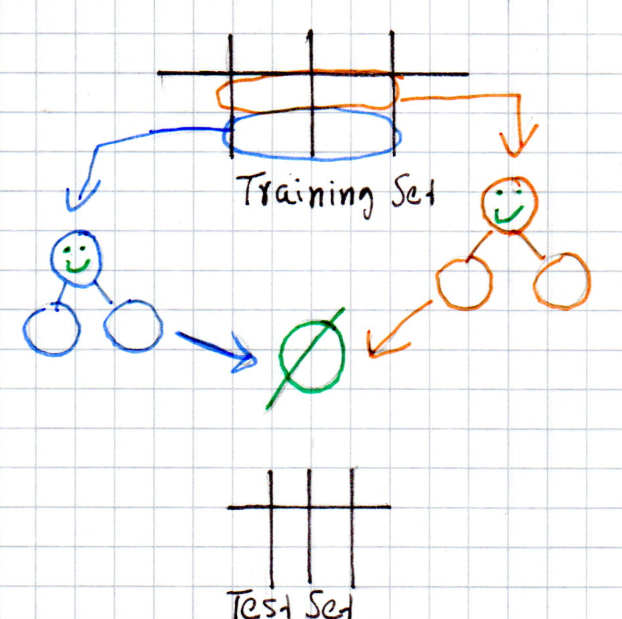
<https://towardsdatascience.com/decision-tree-ensembles-bagging-and-boosting-266a8ba60fd9>

<https://towardsdatascience.com/bagging-decision-trees-clearly-explained-57d4d19ed2d3>

<https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/>

* Problem: Entscheidungsbäume sind ML Algorithmen, die je nach Trainingsdaten stark schwanken
* Lösung: Bagging = Bootstrap auf Algorithmus anwenden
* Datensatz in viele Subsets aufteilen
* Ein Entscheidungsbaum pro Subset
* Wenn man einen neuen Datensatz bekommt, wendet man alle Entscheidungsbäume darauf an und nimmt den Durchschnitt ihrer Ergebnisse





## Random Forest

What is a Random Forest?

<https://www.ibm.com/cloud/learn/random-forest>

Random forest is a commonly-used machine learning algorithm trademarked by Leo Breiman and Adele Cutler, which combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems.

Aus <[*https://www.ibm.com/cloud/learn/random-forest*](https://www.ibm.com/cloud/learn/random-forest)>

Definition: Kombiniert die Ergebnisse mehrerer Entscheidungsbäume

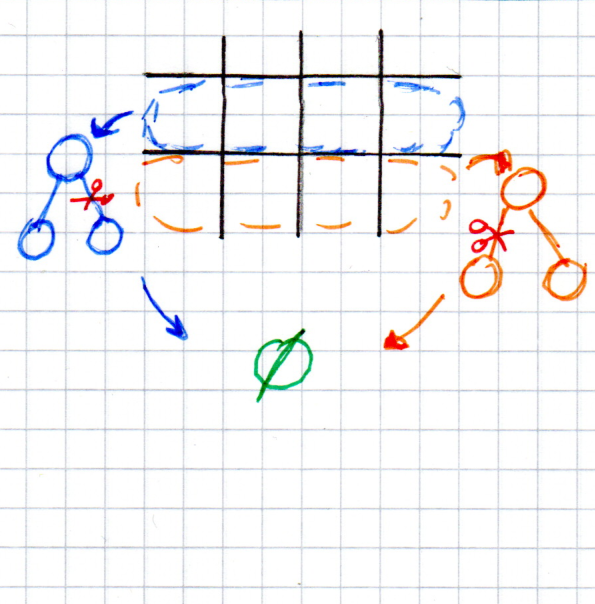
Erweiterte Form des Bagging

Zusätzlich: Feature Randomness

Nicht jeder Baum spielt alle möglichen Merkmale durch  
-> Wenig Korrelation der Bäume untereinander

Aus <[*https://trello.com/c/HnA8TdHJ/97-random-forest*](https://trello.com/c/HnA8TdHJ/97-random-forest)>

Random Forest



# Erster Zyklus

## Schritte

* TrainingSet.py - Datensätze vorbereiten
* Module importieren

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* Dataframe erstellen

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* Logarithmus für jeden Wert der abhängigen Variable "count" berechnen

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* Fastai Funktion "add\_datepart" verwenden



* Dataframe in training set und test set teilen
  + Wurde in der Lektion so gemacht
  + Ich hatte da noch nicht verstanden, dass ich das hier nicht machen muss, weil Testset separat enthalten ist

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* Y Variable aus Testset speichern, dann entfernen

Ein Bild, das Text enthält.

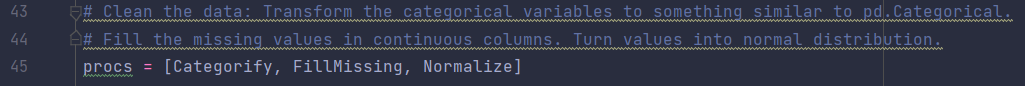
Automatisch generierte Beschreibung

* Die nächsten Schritte führen zu Tabular Pandas. Ich wusste aus dem Video, dass die nächste Aufgabe die Bereinigung der Daten ist und fand die Funktion fill missing als Teil von Tabular Pandas. Also habe ich alle in der Dokumentation aufgeführten Schritte ausgeführt, auch wenn ich sie nicht alle ganz verstanden habe.
* Qualitative und quantitative Variablen definieren

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* Daten bereinigen
  + Categorify
  + Fill missing
  + Normalize



* Splits definieren
* Erstelle TabularPandas Objekt aus Trainingset

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* Datensätze speichern
  + Trainingset nach feather
  + Testset nach csv, weil feather nicht funktioniert hat

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* Train.py - RF über Trainingset laufen lassen
* Module importieren

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* Datensätze importieren

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* sklearn.ensemble.RandomForestRegressor() Objekt erstellen

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* fit method an training set ausühren
* y Variable im Testset vorhersagen

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* Vorhersage auswerten mit Wurzelquadrat des mittleren logarithmischen Fehlers



* Leistung messen
  + Laufzeit
  + RAM



Ein Bild, das Text, Gerät, Anzeige enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

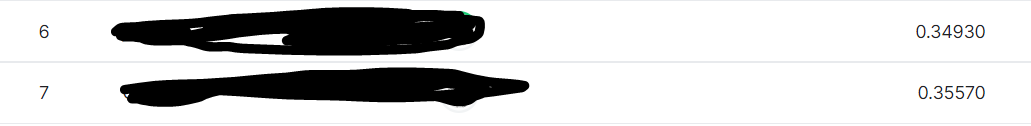
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* Visualisierung

<<Score.png>>

Ranking



# Code Review

## How to review code

## Sources

[Code Review: Checkliste, Tools & Guidelines](https://www.johner-institut.de/blog/fda/code-review/)

[The Ultimate Code Review Checklist](https://www.codegrip.tech/productivity/the-ultimate-code-review-checklist/?utm_source=website&utm_medium=blog&utm_campaign=best-practices-for-code-review-process)

## Criteria for code

**Manageability**: Easy to read

**Architecture**: Uniform design

**Maintainability**: Easy to maintain

**Correctness**: Always produces correct results

**Invalid input/states**: Can the code take in all necessary input?

**Usability**: Easy to use

**Reusability**: No unnecessary elements

## Sam

Leerzeichen zwischen Codeabschnitten!

*Aus <*[*https://github.com/henrike-94/Predicting\_bike\_rental\_demand/issues/27*](https://github.com/henrike-94/Predicting_bike_rental_demand/issues/27)*>*

## Eike

### Lesbarkeit

In Zeile 14 benutzt du einen irrsinnig langen absoluten Pfad, besser wäre, den relativ anzugeben, das wäre auch viel kürzer:

train\_path = "../Datasets/train.csv"

dfFirstCycle = pandas.read\_csv(train\_path, low\_memory=False, parse\_dates=["datetime"])

Am besten bleibst du immer und zwar wirklich konsequent unter 80 Zeichen pro Zeile

Man kann eigentlich 100% der Statements auf mehrere Zeilen strecken, das macht keinen Unterschied in der Laufzeit, macht aber den Code leserlicher

* **Lange Dateipfade vermeiden**
* **Kürzere Zeilen**

Du benutzt wechselnd string-comments, also '''blabla''' und hash-comments, also # blablabla

Das ist etwas irritierend, wenn es nicht passiert um tatsächlich eine Unterscheidung vorzunehmen

Bspw. ist es typisch string-comments für die Dokumentation einer Funktion oder Klasse zu verwenden und #-comments um den Programmfluss zu kommentieren

Dann wechselst du zwischen camelCase und snake\_case für die Variablennamen und in einigen Fällen mischst du sogar

Das ist etwas irritierend

### Einheitlicher Aufbau

Du benutzt wechselnd string-comments, also '''blabla''' und hash-comments, also # blablabla

Das ist etwas irritierend, wenn es nicht passiert um tatsächlich eine Unterscheidung vorzunehmen

Bspw. ist es typisch string-comments für die Dokumentation einer Funktion oder Klasse zu verwenden und #-comments um den Programmfluss zu kommentieren

Dann wechselst du zwischen camelCase und snake\_case für die Variablennamen und in einigen Fällen mischst du sogar

Das ist etwas irritierend

* **Einheitliche Schreibweise für Variablen und Kommentare**

### Leicht zu warten

Das Projektverzeichnis ist etwas unordentlich und könnte aufgeräumt werden

Zb. könnten die pdfs und Tabellen in ein Unterverzeichnis ausgelagert werden

Es gibt außerdem sowohl den Ordner "Datasets", als auch "CsvDatasets", in beiden befinden sich csv-Daten, es ist nicht ersichtlich, wieso welches wo ist

* **Ordnung in Projektordnern halten**

### Nicht redundant

Du hast in der Train.py mehrere imports die nicht bebraucht werden

Das sind

os

matplotlib.pyplot

fastai

make\_regression

mean\_squared\_error

plotly.express

* **Ungenutzte Importe löschen**

# Zweiter Zyklus

## Verbesserungen

* Eine Datei zum Trainieren, eine zum Testen

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* Relative Dateipfade

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* Max. Zeilenlänge 80 Zeichen

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* Einheitliche Schreibung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* Saubere Benennung "train" und "test"
* Kein Splitten

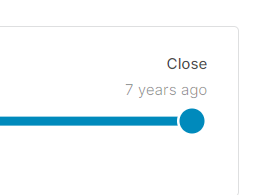
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* Ranking - entfällt

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung



# Gelöste Probleme

## Veralteter Code

<https://github.com/henrike-94/Predicting_bike_rental_demand/issues/18>

<https://ichi.pro/de/maschinelles-lernen-1-lektion-3-275154493557534>

### Proc\_df

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* Funktion im Video, die Daten bereinigt
* Fehlende Werte mit Medianen füllt
* Qualitative in quantitative Werte umwandelt
* Funktion des moduls "fastai.structured"
* Modul wurde nicht gefunden, weder in PyCharm, noch in der PowerShell
* testFilled = fastai.structured.proc\_df(test)

Windows PowerShell  
Copyright (C) Microsoft Corporation. Alle Rechte vorbehalten.

Lernen Sie das neue plattformübergreifende PowerShell kennen – <https://aka.ms/pscore6>

PS C:\Users\henri> python  
Python 3.9.5 (tags/v3.9.5:0a7dcbd, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import fastai  
>>> import fastai.structured  
Traceback (most recent call last):  
 File "<stdin>", line 1, in <module>  
ModuleNotFoundError: No module named 'fastai.structured'  
>>>

Aus <[*https://github.com/henrike-94/Predicting\_bike\_rental\_demand/issues/18*](https://github.com/henrike-94/Predicting_bike_rental_demand/issues/18)>

* In der fastai Dokumentation geguckt, ob die Funktion noch Teil der aktuellen Paketversion ist - keine Ergebnisse
* Nach Ersatzfunktion gesucht
* Gefunden: FillMissing

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* Darüber auf die Klasse TabularPandas gestoßen und den Datensatz als deren Objekt modelliert

<https://docs.fast.ai/tabular.core.html#TabDataLoader>

Integration example

For a more in-depth explanation, see the tabular tutorial

path = untar\_data(URLs.ADULT\_SAMPLE)

df = pd.read\_csv(path/'adult.csv')

df\_main,df\_test = df.iloc[:10000].copy(),df.iloc[10000:].copy()

df\_test.drop('salary', axis=1, inplace=True)

df\_main.head()

cat\_names = ['workclass', 'education', 'marital-status', 'occupation', 'relationship', 'race']

cont\_names = ['age', 'fnlwgt', 'education-num']

procs = [Categorify, FillMissing, Normalize]

splits = RandomSplitter()(range\_of(df\_main))

to = TabularPandas(df\_main, procs, cat\_names, cont\_names, y\_names="salary", splits=splits)

dls = to.dataloaders()

dls.valid.show\_batch()

Gelernt:

Code ist nicht in Stein gemeißelt verfügbar

Dokumentationen checken

### Add\_datepart()

<https://github.com/henrike-94/Predicting_bike_rental_demand/issues/18>

Ein Bild, das Text, Screenshot, Monitor, schwarz enthält.

Automatisch generierte Beschreibung

* Fehlermeldung beim Funktionsaufruf add\_datepart

raise KeyError(f"None of [{key}] are in the [{axis\_name}]")

KeyError: "None of [DatetimeIndex(['2011-01-01 00:00:00', '2011-01-01 01:00:00',\n

Aus <[*https://github.com/henrike-94/Predicting\_bike\_rental\_demand/issues/18*](https://github.com/henrike-94/Predicting_bike_rental_demand/issues/18)>

* Dokumentation checken und Beispiel nachbauen
* Post bei Stack Overflow
* Rückmeldung: add\_datepart wants the name of the column, not its values – Paul H Oct 26 at 15:37

Aus <[*https://github.com/henrike-94/Predicting\_bike\_rental\_demand/issues/18*](https://github.com/henrike-94/Predicting_bike_rental_demand/issues/18)>

* Folgerung: Funktion erkennt "trainingSetFirstCycle.date" nicht als Namen der Spalte
* Varianten ausprobiert, nur "date" hat geklappt

* **Gelernt**:
* Falsche Annahme, das Spalten immer mit "Tabelle.Spalte" bezeichnet werden, weil die Funktion "numpy.log(trainingSetFirstCycle.rent\_count)" das so macht
* Vergegenwärtigt, dass Code ja von Menschen geschrieben wird, deshalb gibt es zwischen Modulen immer Unterschiede in der Benennung

## Value Error

RF1 = sklearn.ensemble.RandomForestRegressor()  
# Run fit method on training set  
RF1.fit(X\_train, Y\_train)  
# Predict y variable in validation set.  
x1=RF1.predict(X\_validate)  
x2=Y\_validate  
score=math.sqrt(sklearn.metrics.mean\_squared\_log\_error(x2, x1))  
print(score)

ValueError: y\_true and y\_pred have different number of output (2!=1)

But they both do have the same number of elements, as they are supposed to:

print(x1)  
[6.43182586 6.43182586 6.43182586 ... 6.43283502 6.43283502 6.43283502]  
print(len(x1))  
5443  
print(len(x2))  
5443

Aus <[*https://github.com/henrike-94/Predicting\_bike\_rental\_demand/issues/23*](https://github.com/henrike-94/Predicting_bike_rental_demand/issues/23)>

* Value Error, weil berechnete und tatsächliche y-Werte unterschiedlichen Output haben
* Definition Output: Was ein Code zurückgibt
* Ursache laut <https://fixexception.com/scikit-learn/y-true-and-y-pred-have-different-number-of-output-param0-param1/>: Verschiedene Number Labels
* Arrays ansehen: y\_true hat solche Zahlenpaare, während y\_pred nur die erwünschten Kommazahlen hat

print(y\_true)

[[5.44300000e+03 4.26267988e+00]

[5.44400000e+03 4.18965474e+00]

[5.44500000e+03 3.36729583e+00]

...

[1.08830000e+04 5.12396398e+00]

[1.08840000e+04 4.85981240e+00]

[1.08850000e+04 4.47733681e+00]]

print(y\_pred)

[6.4309396 6.4309396 6.4309396 ... 6.43084709 6.43084709 6.43084709]

* Wie kommt das?
* "Print (Y\_validate)" offenbart die Lösung. "y\_pred" hatte ich zuerst als "Y\_validate.to\_numpy" definiert. "Y validate" ist ja die Spalte "rent\_count", die aus dem Testset entfernt und separat gespeichert wurde.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

print(Y\_validate)

Unnamed: 0 rent\_count

0 5443 4.262680

1 5444 4.189655

2 5445 3.367296

3 5446 3.663562

4 5447 2.484907

... ...

5438 10881 5.817111

5439 10882 5.484797

5440 10883 5.123964

5441 10884 4.859812

5442 10885 4.477337

[5443 rows x 2 columns]

* "Y\_validate" enthält auch noch die Zeilen-ID-Nummern der ursprünglichen Tabelle "dfFirstCycle"!
* Lösung: "y\_true" neu definieren:

y\_true=Y\_validate.rent\_count.to\_numpy()

print(y\_true)

[4.26267988 4.18965474 3.36729583 ... 5.12396398 4.8598124 4.47733681]

* Gelernt: Schau dir deinen Code an!

## Score

<https://github.com/henrike-94/Predicting_bike_rental_demand/issues/23>

File "<input>", line 78

print("Score RMSLE:", scoreRMSLE)

^

SyntaxError: invalid syntax

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

* Funktioniert aber im interaktiven Modus, kann also nicht an der Syntax der Funktion "print(score)" liegen

# Evaluate prediction using root mean squared log error

scoreRMSLE=math.sqrt(sklearn.metrics.mean\_squared\_log\_error(y\_true, y\_pred))

Backend TkAgg is interactive backend. Turning interactive mode on.

print("Score RMSLE:", scoreRMSLE)

Score RMSLE: 0.35089926356805246

* Skript bis zu verschiedenen Punkten ausgeführt, um zu sehen, ab wann die Fehlermeldung kommt
* Fehlerquelle lokalisiert: Funktion "cpu\_usage = cpu\_load(os.cpu\_count()) \* 100", der die schließende Klammer fehlt. Deshalb hat der Compiler "print(score)" als Teil dieser Funktion interpretiert.
* "cpu\_usage" gelöscht, weil sowieso nicht effektiv

* Gelernt: Schau dir deinen Code an!