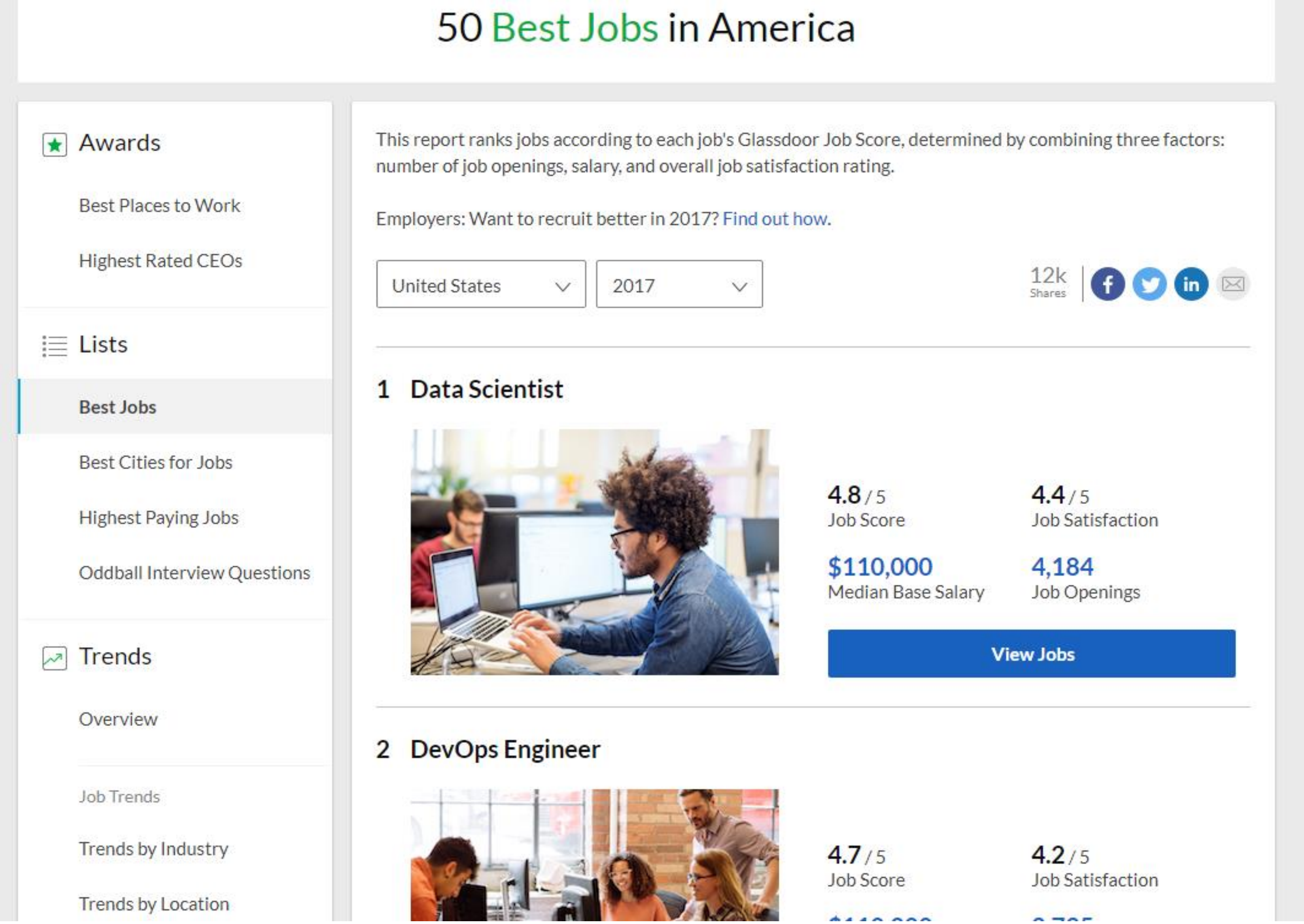
Advanced Practical Course: Data Science

Dr. David Koll

Why Data Science?

“The demand for data scientists is increasing so quickly, that McKinsey predicts that by 2018, there will be a 50 percent gap in the supply of data scientists versus demand.” – The Alexa Blog (http://blog.alexa.com/know-data-science-important/)

Why Data Science?



50 Best Jobs in America

This report ranks jobs according to each job's Glassdoor Job Score, determined by combining three factors: number of job openings, salary, and overall job satisfaction rating.

Employers: Want to recruit better in 2017? Find out how.

United States
2017

1 Data Scientist

4.8 / 5 Job Score

4.4 / 5 Job Satisfaction

$110,000 Median Base Salary

4,184 Job Openings

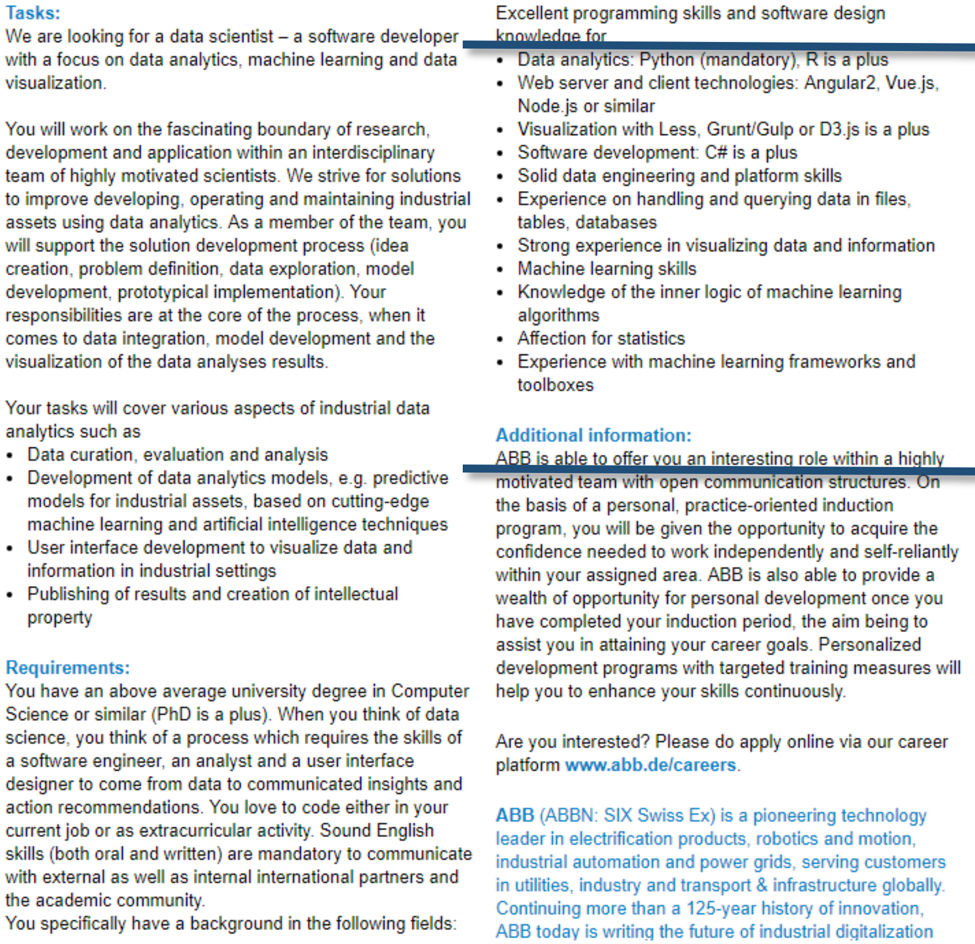
2 DevOps Engineer

4.7 / 5 Job Score

4.2 / 5 Job Satisfaction

– Glassdoor Career Website (http://www.glassdoor.com/)

Why *Practical* Data Science?



Tasks:
We are looking for a data scientist – a software developer with a focus on data analytics, machine learning and data visualization.

You will work on the fascinating boundary of research, development and application within an interdisciplinary team of highly motivated scientists. We strive for solutions to improve developing, operating and maintaining industrial assets using data analytics. As a member of the team, you will support the solution development process (idea creation, problem definition, data exploration, model development, prototypical implementation). Your responsibilities are at the core of the process, when it comes to data integration, model development and the visualization of the data analyses results.

Your tasks will cover various aspects of industrial data analytics such as

- Data curation, evaluation and analysis
- Development of data analytics models, e.g. predictive models for industrial assets, based on cutting-edge machine learning and artificial intelligence techniques
- User interface development to visualize data and information in industrial settings
- Publishing of results and creation of intellectual property

Requirements:
You have an above average university degree in Computer Science or similar (PhD is a plus). When you think of data science, you think of a process which requires the skills of a software engineer, an analyst and a user interface designer to come from data to communicated insights and action recommendations. You love to code either in your current job or as extracurricular activity. Sound English skills (both oral and written) are mandatory to communicate with external as well as internal international partners and the academic community.
You specifically have a background in the following fields: Excellent programming skills and software design knowledge for

- Data analytics: Python (mandatory), R is a plus
- Web server and client technologies: Angular2, Vue.js, Node.js or similar
- Visualization with Less, Grunt/Gulp or D3.js is a plus
- Software development: C# is a plus
- Solid data engineering and platform skills
- Experience on handling and querying data in files, tables, databases
- Strong experience in visualizing data and information
- Machine learning skills
- Knowledge of the inner logic of machine learning algorithms
- Affection for statistics
- Experience with machine learning frameworks and toolboxes

Additional information:
ABB is able to offer you an interesting role within a highly motivated team with open communication structures. On the basis of a personal, practice-oriented induction program, you will be given the opportunity to acquire the confidence needed to work independently and self-reliantly within your assigned area. ABB is also able to provide a wealth of opportunity for personal development once you have completed your induction period, the aim being to assist you in attaining your career goals. Personalized development programs with targeted training measures will help you to enhance your skills continuously.

Are you interested? Please do apply online via our career platform **www.abb.de/careers**.

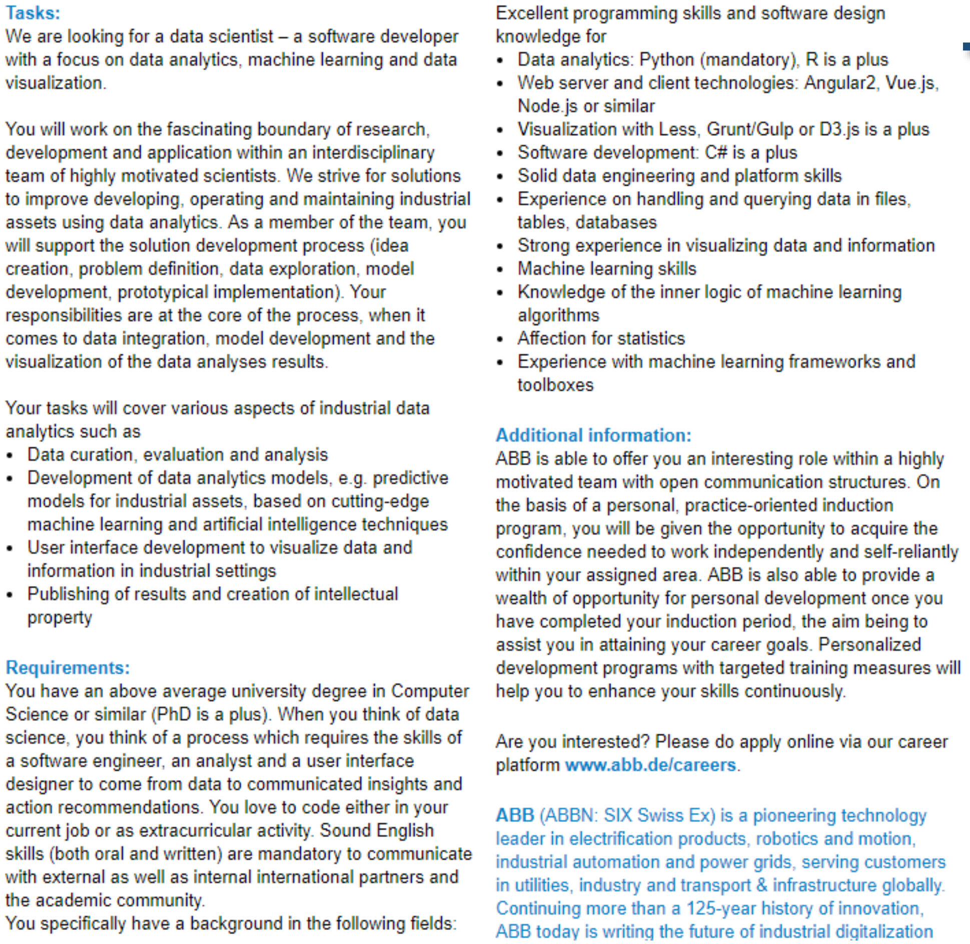
ABB (ABBN: SIX Swiss Ex) is a pioneering technology leader in electrification products, robotics and motion, industrial automation and power grids, serving customers in utilities, industry and transport & infrastructure globally. Continuing more than a 125-year history of innovation, ABB today is writing the future of industrial digitalization

We are looking for a data scientist – a software developer with a focus on data analytics, machine learning and data visualization.

Your tasks will cover various aspects of industrial data analytics such as

- Data Curation, evaluation and analysis
- Development of data analytics models, e.g., predictive models for industrial assets, based on cutting-edge machine learning and artificial intelligence techniques
- User interface development to visualize data and information in industrial settings
- Publishing of results and creation of intellectual property

Why *Practical* Data Science?



Tasks:
We are looking for a data scientist – a software developer with a focus on data analytics, machine learning and data visualization.

You will work on the fascinating boundary of research, development and application within an interdisciplinary team of highly motivated scientists. We strive for solutions to improve developing, operating and maintaining industrial assets using data analytics. As a member of the team, you will support the solution development process (idea creation, problem definition, data exploration, model development, prototypical implementation). Your responsibilities are at the core of the process, when it comes to data integration, model development and the visualization of the data analyses results.

Your tasks will cover various aspects of industrial data analytics such as

- Data curation, evaluation and analysis
- Development of data analytics models, e.g. predictive models for industrial assets, based on cutting-edge machine learning and artificial intelligence techniques
- User interface development to visualize data and information in industrial settings
- Publishing of results and creation of intellectual property

Requirements:
You have an above average university degree in Computer Science or similar (PhD is a plus). When you think of data science, you think of a process which requires the skills of a software engineer, an analyst and a user interface designer to come from data to communicated insights and action recommendations. You love to code either in your current job or as extracurricular activity. Sound English skills (both oral and written) are mandatory to communicate with external as well as internal international partners and the academic community.
You specifically have a background in the following fields: Excellent programming skills and software design knowledge for

- Data analytics: Python (mandatory), R is a plus
- Web server and client technologies: Angular2, Vue.js, Node.js or similar
- Visualization with Less, Grunt/Gulp or D3.js is a plus
- Software development: C# is a plus
- Solid data engineering and platform skills
- Experience on handling and querying data in files, tables, databases
- Strong experience in visualizing data and information
- Machine learning skills
- Knowledge of the inner logic of machine learning algorithms
- Affection for statistics
- Experience with machine learning frameworks and toolboxes

Additional information:
ABB is able to offer you an interesting role within a highly motivated team with open communication structures. On the basis of a personal, practice-oriented induction program, you will be given the opportunity to acquire the confidence needed to work independently and self-reliantly within your assigned area. ABB is also able to provide a wealth of opportunity for personal development once you have completed your induction period, the aim being to assist you in attaining your career goals. Personalized development programs with targeted training measures will help you to enhance your skills continuously.

Are you interested? Please do apply online via our career platform **www.abb.de/careers**.

ABB (ABBN: SIX Swiss Ex) is a pioneering technology leader in electrification products, robotics and motion, industrial automation and power grids, serving customers in utilities, industry and transport & infrastructure globally. Continuing more than a 125-year history of innovation, ABB today is writing the future of industrial digitalization

→

- Data analytics: Python (mandatory), R is a plus
- Web server and client technologies: Angular2, Vue.js, Node.js
- Visualization with Less, Grunt/Gulp or D3.js is a plus
- Software development: C# is a plus
- Solid data engineering and platform skills
- Experience on handling and querying data in files, tables, databases
- Strong experience in visualizing data and information
- Machine learning skills
- Knowledge of the inner logic of machine learning algorithms
- Affection for statistics
- Experience with machine learning frameworks and toolboxes

Course Prerequisites

- Background in Machine Learning is required
  - “Data Science and Big Data Analytics” taught by the SWE group
  - “Machine-Learning” Coursera course by Stanford
  - …

- Background in Programming is required
  - Use R, Python, JAVA, MATLAB, …
  - Anything that offers ML libraries
  - Course materials (e.g., exemplary solutions) will be in Python
  - Recommendation: Python or R

- Make sure to bring enough time!

Course Structure

- **This is a practical course!**
- **Strong focus on practice over theory!**
- **Learn basic theory in prerequisite courses!**

- There will be only four lectures
  - Lecture 1: The Data Science Pipeline – or: How to approach a practical task? **Today!**
  - Lecture 2: The Python Data Science Stack
  - Lecture 3: Advanced Algorithms for Data Science – or: How to compete?
  - Lecture 4: Evaluation and fine tuning of predictive models

Course Structure

- **Strong focus on practice over theory!**

- There will be three practical tasks
  - Task 1: Warming Up
  - Task 2: Exploring and analyzing data
  - Task 3: Competing on Kaggle (classification or regression problem)

- Disclaimer: this semester no core computer networks problem involved
  - If you are interested, send an email for tasks and datasets from previous semesters

Course Structure

- **Please note: course parameters depend on the number of participants.**
  - **Allowed team sizes, presentation lengths and styles, etc., may change throughout the course, depending on how many students approach each task!**

- For each practical task, you will:
  - Work on a real-world dataset
  - Perform exploratory data analysis
  - Build a predictive model
  - Present your findings in class
    - if number of participants allows it
    - Otherwise: submit report document (notebooks or PDFs)

  - Third task:
    - Depending on number of participants either public or face-to-face presentation

  - Summarize your most important findings and methods in a final report

Course HW/SW Requirements

- Ideal: workstation with i5/i7/Ryzen quadcore, 16+ GB RAM, CUDA GPU
- Good: laptop or workstation with dualcore, 8+ GB RAM
- Hardcore mode: single core laptop, <=4GB RAM

- Most important hardware is RAM
- Plus: ~10GB diskspace

- If your hardware only allows hardcore mode, but you would rather not play that way: please send me an email *before starting task 2!*

Course Structure

- For each task, there will be intermediate meetings
  - Opportunity to discuss and share:
    - Problems you have with the data
    - Initial results
    - Ideas for approaching the problem
    - …
- Style of work: preferably in teams of two
  - Working alone possible if number of students allows it
  - Teams of three only when student number is large enough

Course Schedule

2017 Schedule	
19.10.	Lecture 1 (The Data Science Pipeline) Release of first task
26.10.	Lecture 2 (The Python Data Science stack)
02.11.	Task 1: Intermediate meeting
09.11.	No lecture
16.11.	**Task 1: Presentations** Release of second task
23.11.	Lecture 3 (Advanced algorithms for Data Science)
30.11.	Lecture 4 (Evaluation and fine tuning of models)
07.12.	Task 2: Intermediate meeting
14.12.	No lecture
21.12.	**Task 2: Presentations** Release of third task

2018 Schedule	
04.01.	No lecture
11.01.	Task 3: Intermediate meeting I
18.01.	No lecture
25.01.	Task 3: Intermediate meeting II (tentative)
01.02.	No lecture
08.02. **15.02.** **22.02.**	**Task 3: Presentations**
30.03.	**Submission of final report**

Course Grading

- Presentations (or reports): 60%
    - Tasks are contributing to the final grade based on their difficulty
        - Task 1: 10%
        - Task 2: 20%
        - Task 3: 30%

- Intermediate meetings: 20%
    - Bring 2-3 slides with your core findings and/or problems
    - Participate in discussion
    - Note: This category might be dropped if there are too many students

- Final Report: 20%
    - Summarize the key insights and approaches for each task
    - Incorporate feedback of presentation sessions
    - Rough guideline: 8-10 pages double column format
    - Template will be provided

What this course will not cover

- Database foundations (SQL, NoSQL, …) & data engineering techniques (Hadoop, Spark, …)

- In-depth statistics

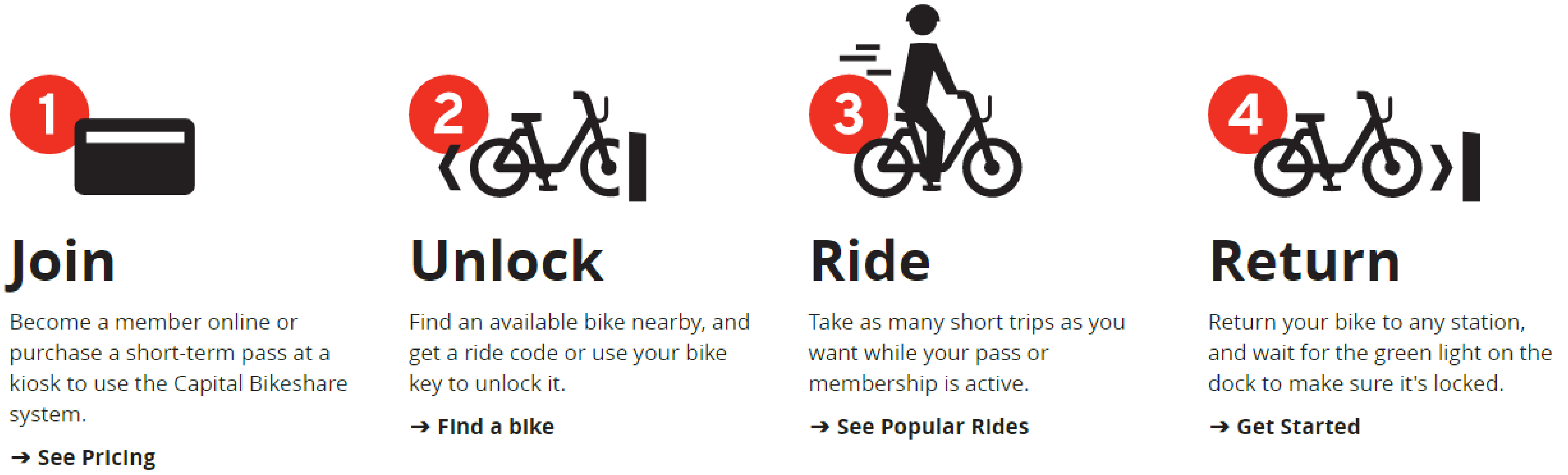
- In-depth algorithmic theory on:
  - Pure regression algorithms
  - Neural networks and deep learning
    - (D/C)NNs are typically used for computer vision problems, which we do not tackle here
    - You are free to use these techniques in the practical part

The Data Science Pipeline

or: How to approach a practical problem

Dr. David Koll

Case study: Bike sharing in Washington D.C.



Join

Become a member online or purchase a short-term pass at a kiosk to use the Capital Bikeshare system.

➔ See Pricing

Unlock

Find an available bike nearby, and get a ride code or use your bike key to unlock it.

➔ Find a bike

Ride

Take as many short trips as you want while your pass or membership is active.

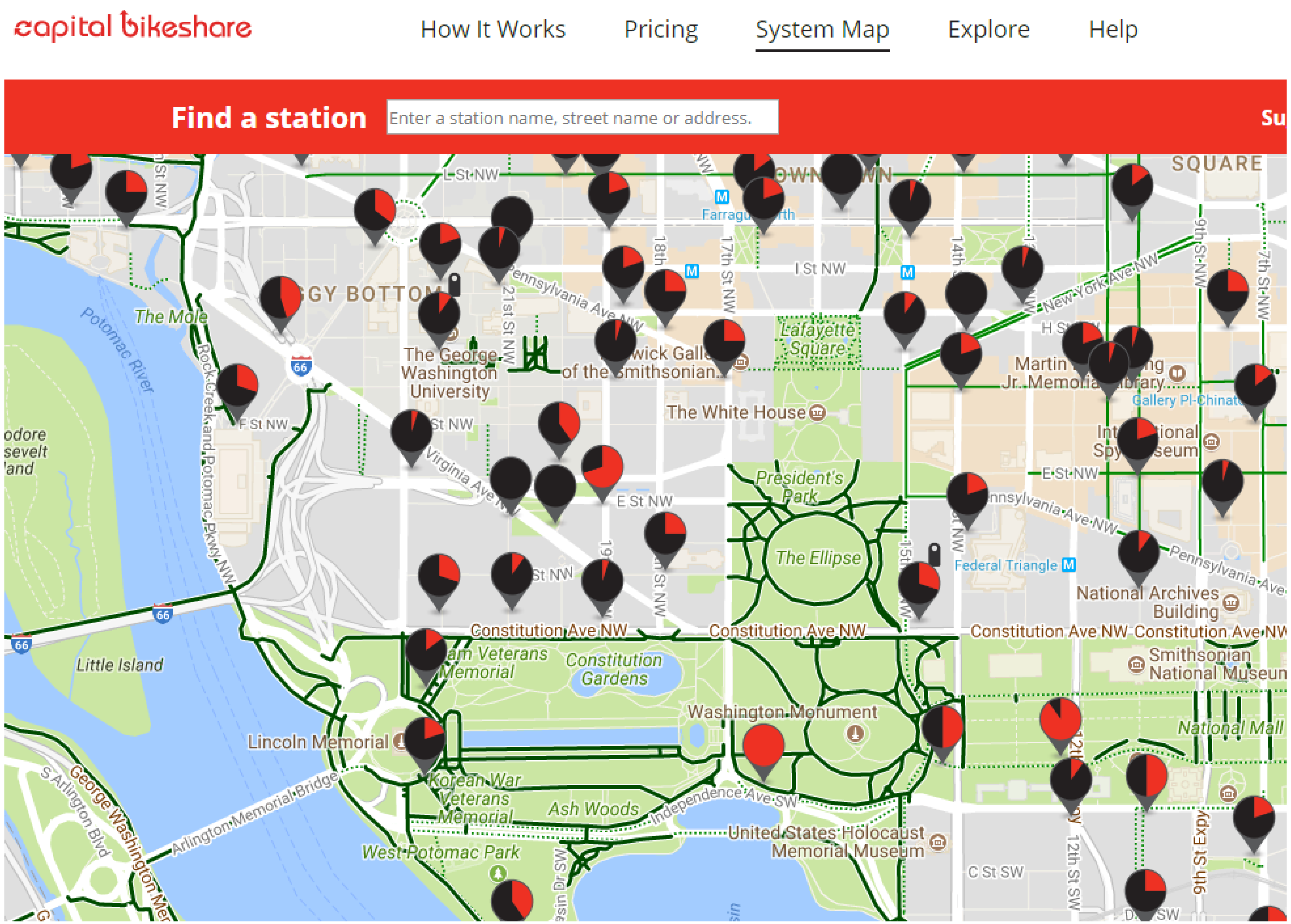
➔ See Popular Rides

Return

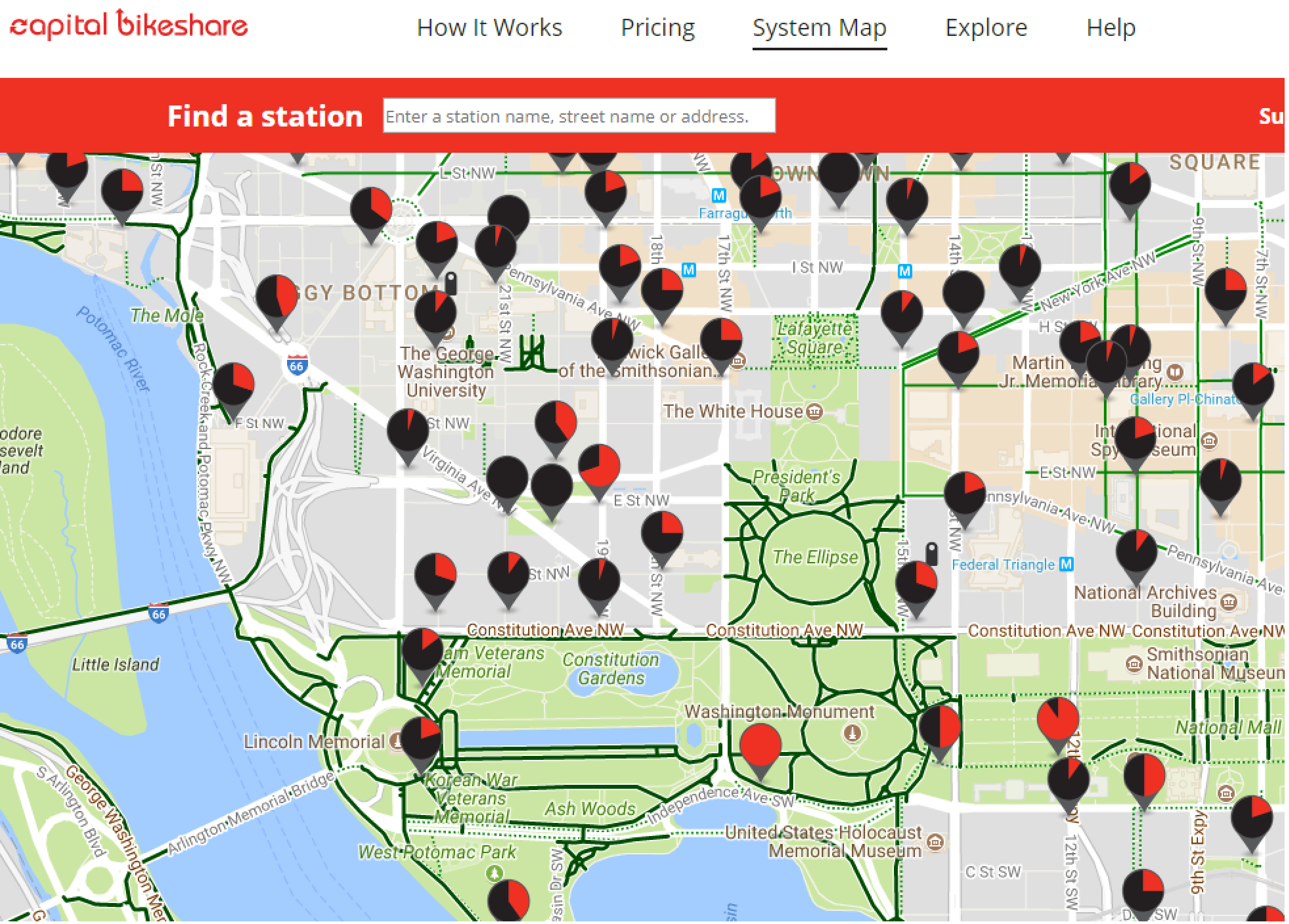
Return your bike to any station, and wait for the green light on the dock to make sure it's locked.

➔ Get Started

Case study: Bike sharing in Washington D.C.



Why is Data Science Important?



System Optimization:

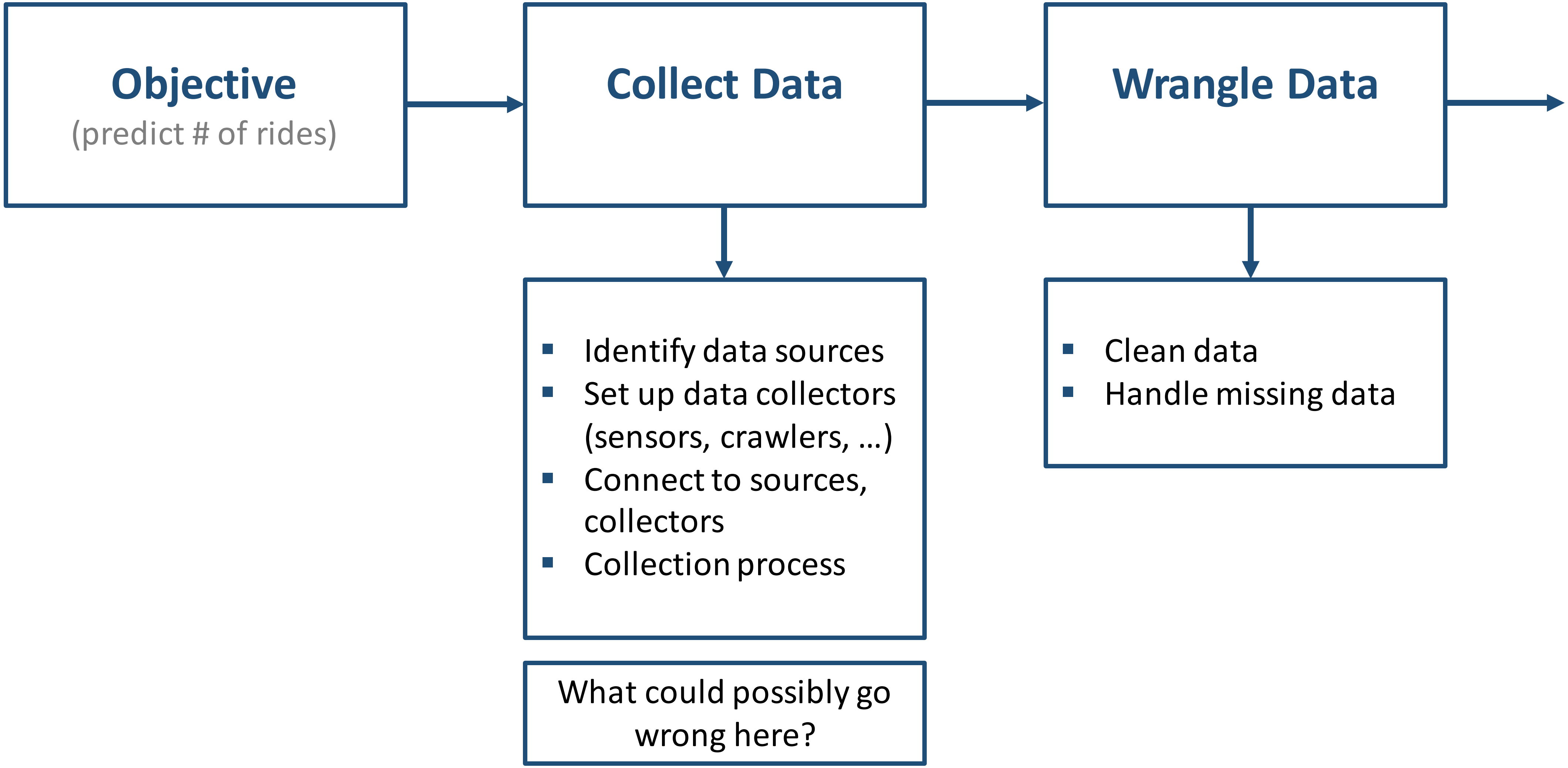
- Replenishment optimization (how many bikes available at what time at which station?)
- Bike tracking for route optimization (suggest optimal/alternative routes/stations to customers)
- Waiting time estimation (in case of empty rack)
- Station placement (based on user movement patterns)
- Individual user recommendations based on user behavior (e.g., advertisements)
- Or simply: amount of customers at a given hour

An exemplary Data Science task

Management: „We need a prediction of usage numbers for our system for every hour of the week“

?

The (Ideal) Data Science Pipeline



Objective

(predict # of rides)

Collect Data

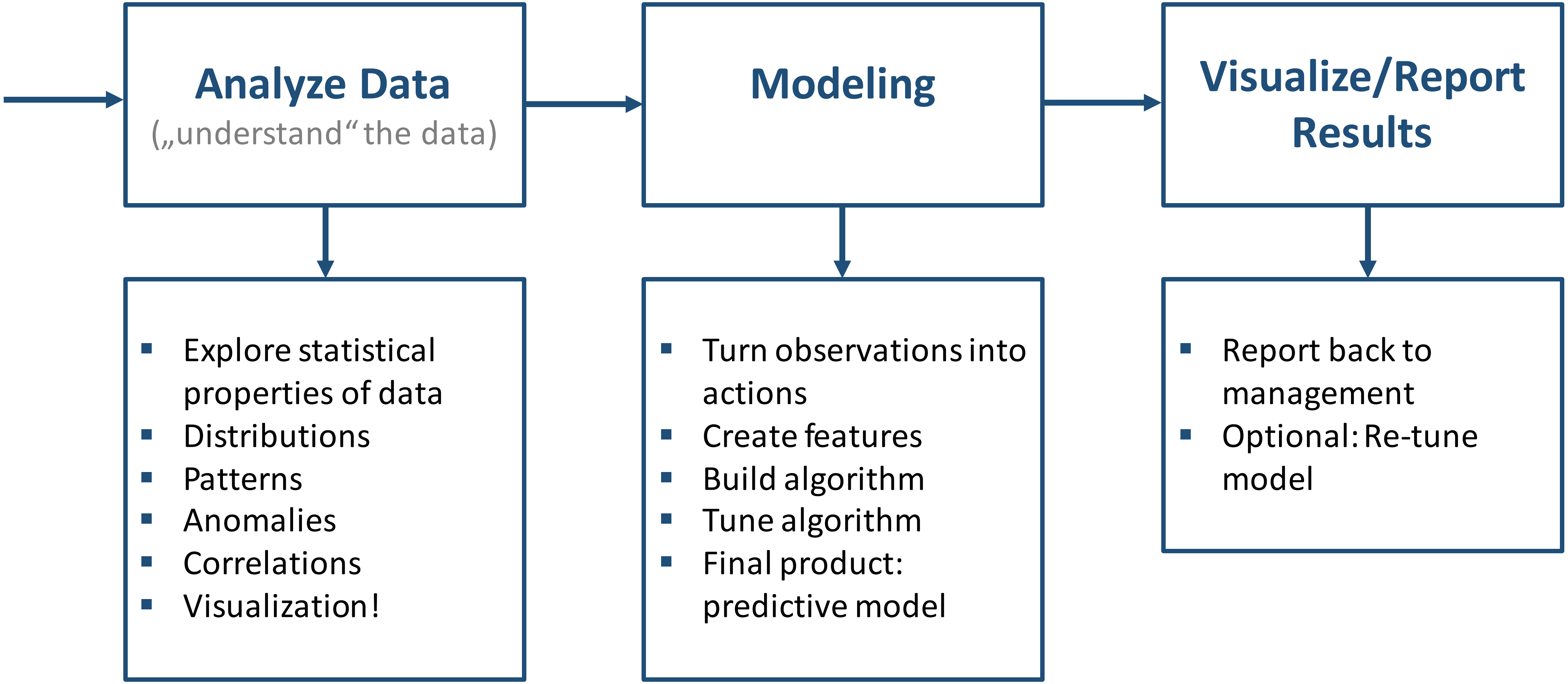
- Identify data sources
- Set up data collectors (sensors, crawlers, …)
- Connect to sources, collectors
- Collection process

What could possibly go wrong here?

Wrangle Data

- Clean data
- Handle missing data

The (Ideal) Data Science Pipeline



Analyze Data

(„understand“ the data)

- Explore statistical properties of data
- Distributions
- Patterns
- Anomalies
- Correlations
- Visualization!

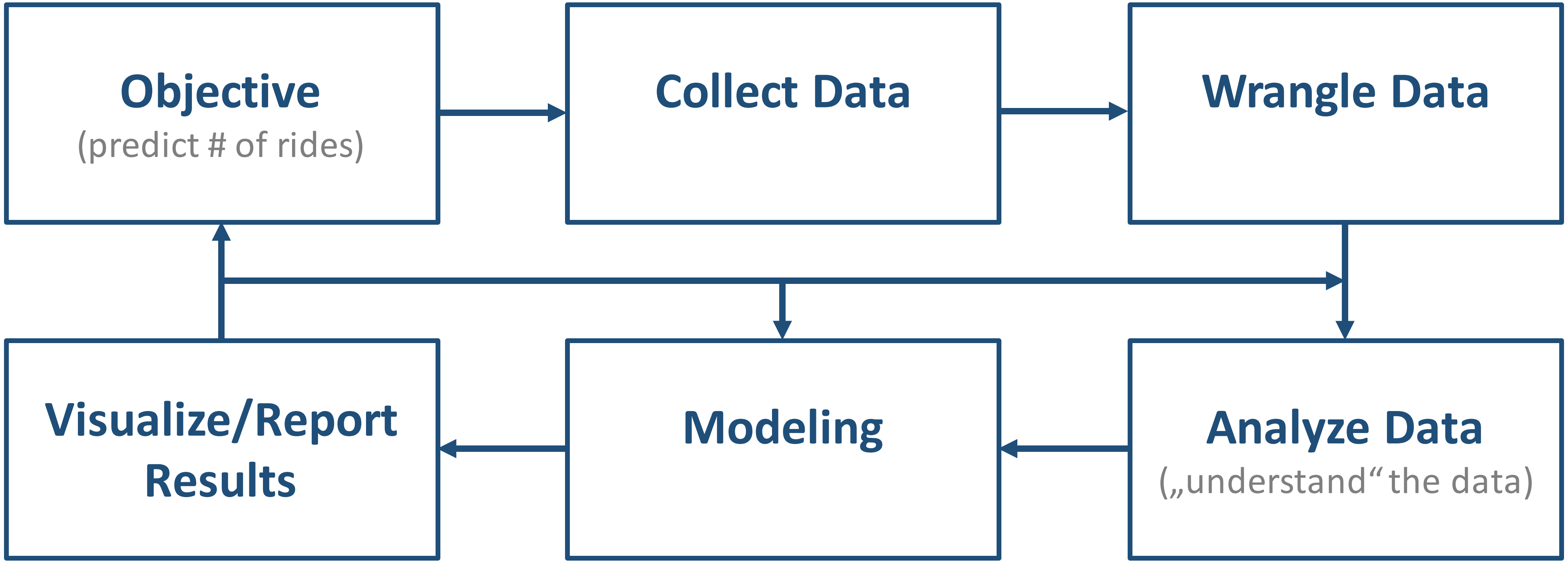
Modeling

- Turn observations into actions
- Create features
- Build algorithm
- Tune algorithm
- Final product: predictive model

Visualize/Report Results

- Report back to management
- Optional: Re-tune model

The (Ideal) Data Science Pipeline



Objective
(predict # of rides)

Collect Data

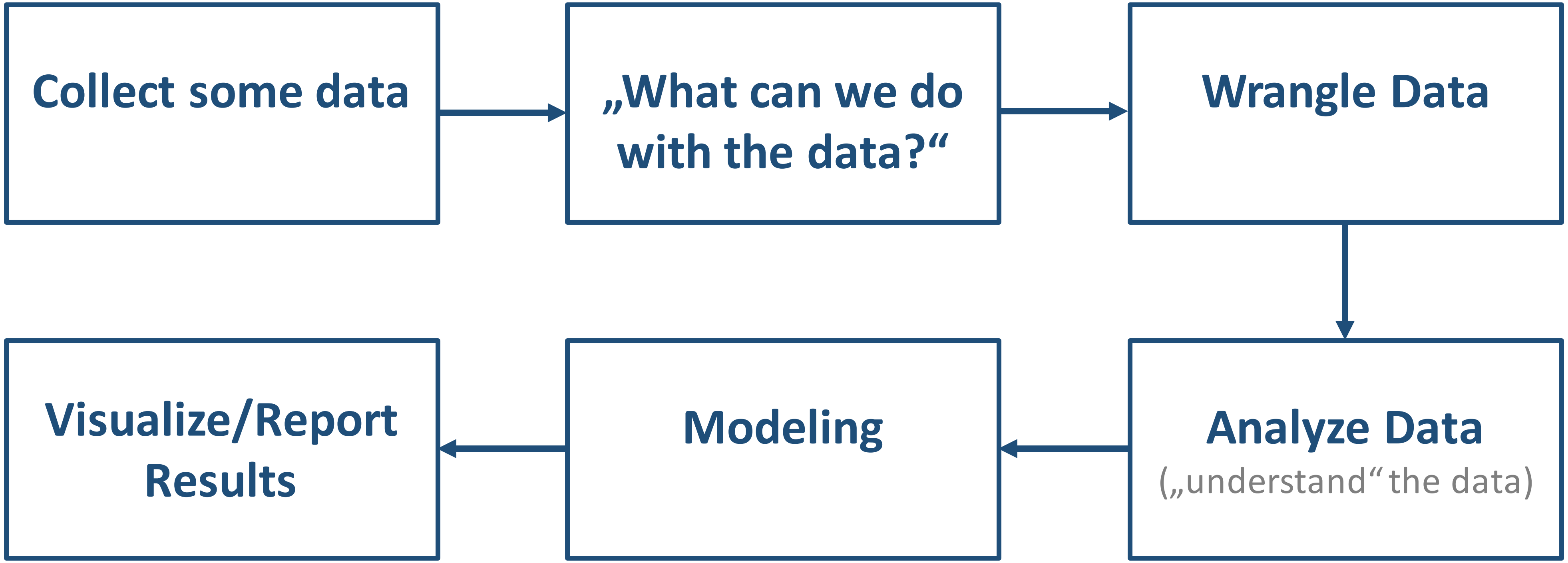
Wrangle Data

Analyze Data
(„understand“ the data)

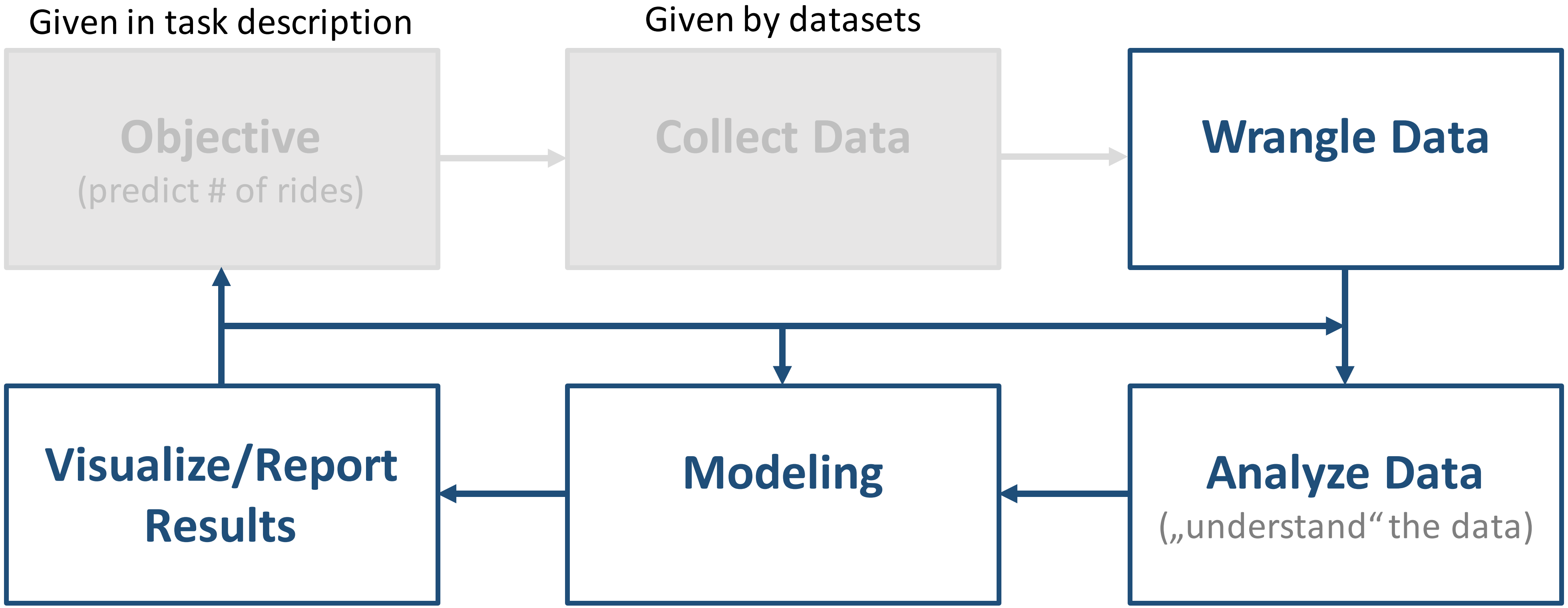
Modeling

Visualize/Report Results

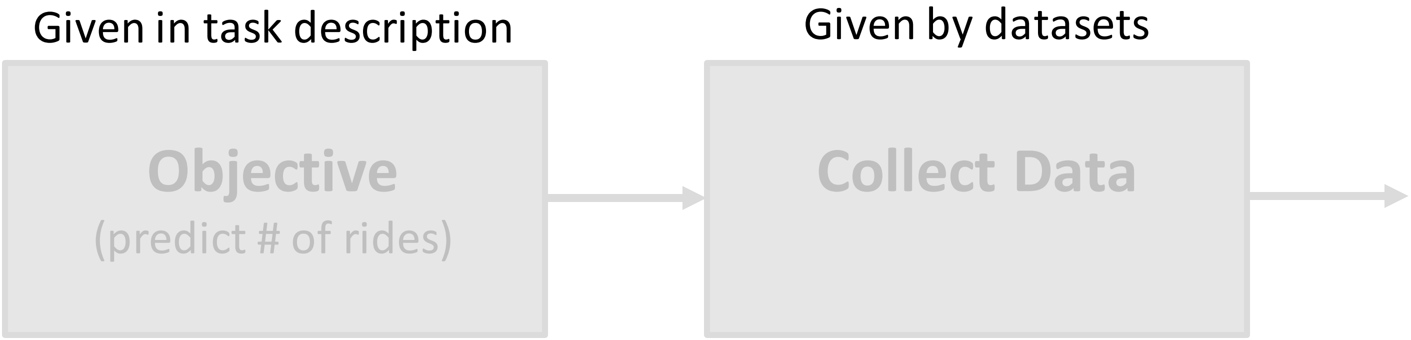
Side Note: A Frequent Real-World Pipeline



In this course…



Case Study: Dataset



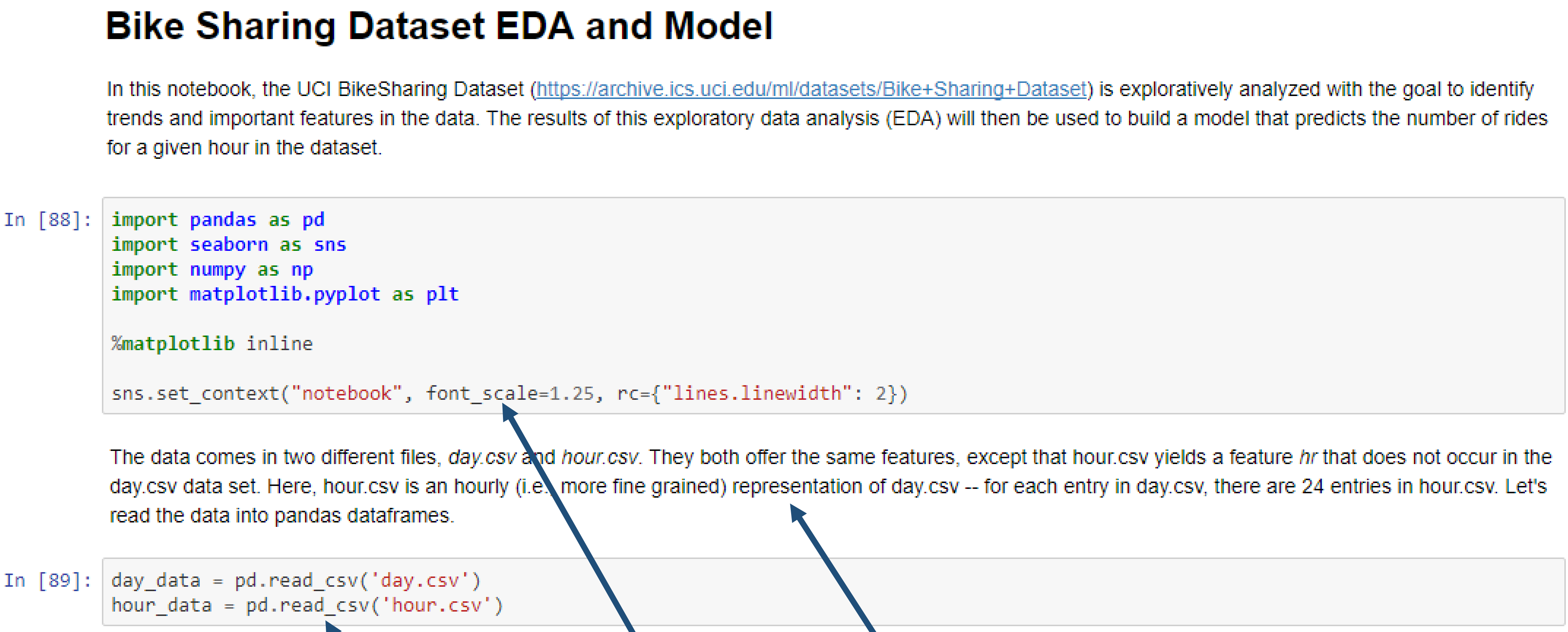
Given in task description

Given by datasets

Data can be found at: https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset

- Split dataset: hourly and daily data of two years (2012/2013) in the bike sharing system operation
- 16 attributes / features
- 17389 instances / rows / feature vectors

Case Study: Use of IPythonNotebook



Bike Sharing Dataset EDA and Model

In this notebook, the UCI BikeSharing Dataset (https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset) is exploratively analyzed with the goal to identify trends and important features in the data. The results of this exploratory data analysis (EDA) will then be used to build a model that predicts the number of rides for a given hour in the dataset.

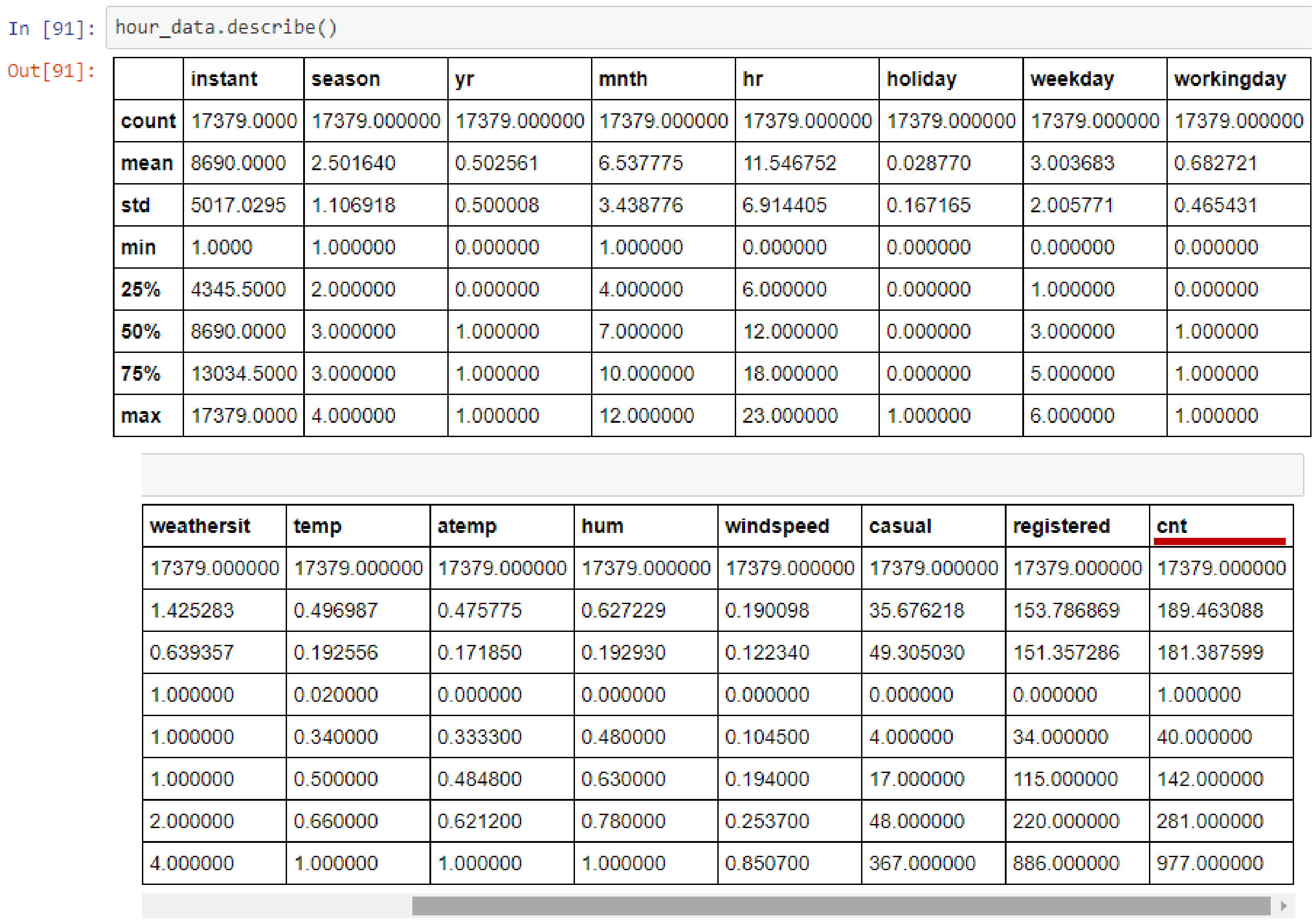
```
In [88]: import pandas as pd
         import seaborn as sns
         import numpy as np
         import matplotlib.pyplot as plt

         %matplotlib inline

         sns.set_context("notebook", font_scale=1.25, rc={"lines.linewidth": 2})
```

The data comes in two different files, *day.csv* and *hour.csv*. They both offer the same features, except that hour.csv yields a feature *hr* that does not occur in the day.csv data set. Here, hour.csv is an hourly (i.e. more fine grained) representation of day.csv -- for each entry in day.csv, there are 24 entries in hour.csv. Let's read the data into pandas dataframes.

```
In [89]: day_data = pd.read_csv('day.csv')
         hour_data = pd.read_csv('hour.csv')
```

- Can combine code, visualization and documentation!

Case Study: Use of IPythonNotebook



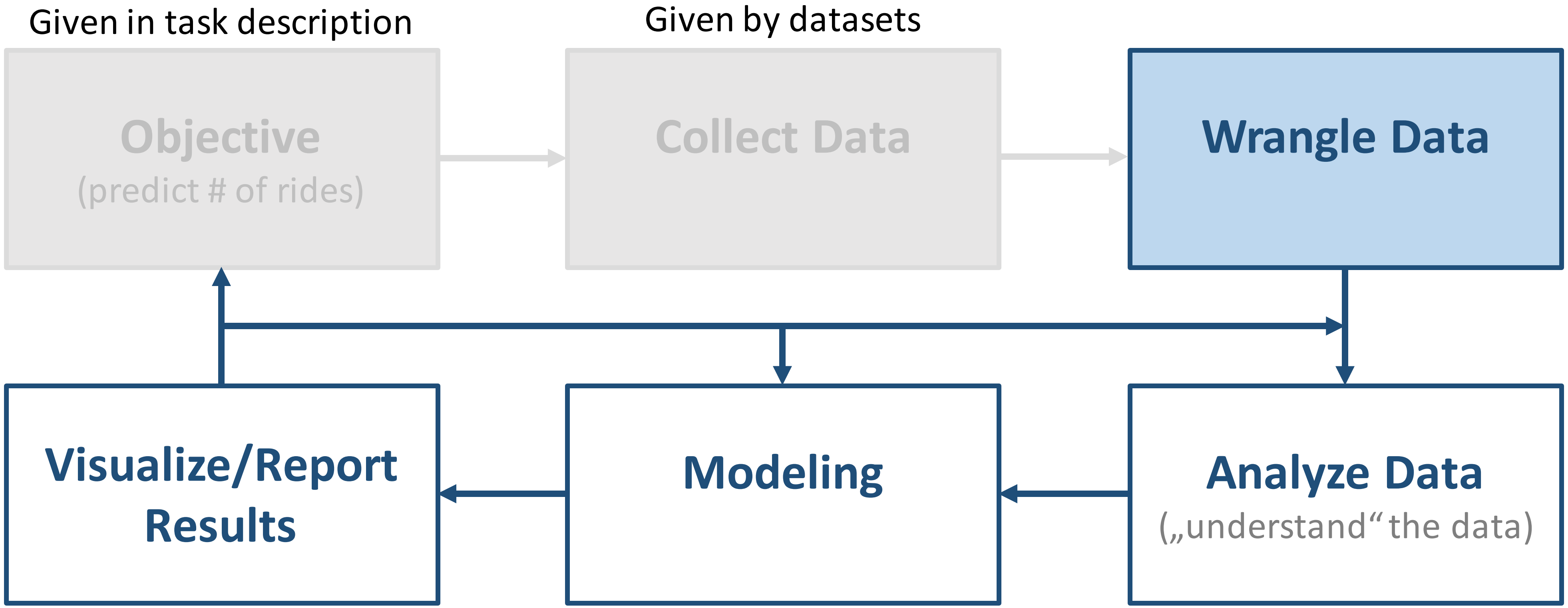
```
In [91]: hour_data.describe()
```

Out[91]:

	instant	season	yr	mnth	hr	holiday	weekday	workingday
count	17379.0000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000
mean	8690.0000	2.501640	0.502561	6.537775	11.546752	0.028770	3.003683	0.682721
std	5017.0295	1.106918	0.500008	3.438776	6.914405	0.167165	2.005771	0.465431
min	1.0000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000
25%	4345.5000	2.000000	0.000000	4.000000	6.000000	0.000000	1.000000	0.000000
50%	8690.0000	3.000000	1.000000	7.000000	12.000000	0.000000	3.000000	1.000000
75%	13034.5000	3.000000	1.000000	10.000000	18.000000	0.000000	5.000000	1.000000
max	17379.0000	4.000000	1.000000	12.000000	23.000000	1.000000	6.000000	1.000000

weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000
1.425283	0.496987	0.475775	0.627229	0.190098	35.676218	153.786869	189.463088
0.639357	0.192556	0.171850	0.192930	0.122340	49.305030	151.357286	181.387599
1.000000	0.020000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000
1.000000	0.340000	0.333300	0.480000	0.104500	4.000000	34.000000	40.000000
1.000000	0.500000	0.484800	0.630000	0.194000	17.000000	115.000000	142.000000
2.000000	0.660000	0.621200	0.780000	0.253700	48.000000	220.000000	281.000000
4.000000	1.000000	1.000000	1.000000	0.850700	367.000000	886.000000	977.000000

In this course…



Wrangling Data

- Cleaning data to make it useable for analysis and model building

- Required, because:
  - Raw data is rarely complete
  - Raw data is rarely consistent
  - Why?

- Different techniques to handle both issues

Missing Data

- E.g., caused by measurement gaps (sensor battery dies)
- Often easy to detect (e.g., look for NaN values)
- A multitude of approaches available

	Original Data			Remove			Impute Column Mean			kNN-Imputation (k=1)		
Item	**A**	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**
1	12	80	1	12	80	1	12	80	1	12	80	1
2	5	NaN	7	-	-	-	5	**286**	7	5	**6**	7
3	4	6	6	4	6	6	4	6	6	4	6	6
4	9	600	15	9	600	15	9	600	15	9	600	15

- Use of strategy depends on nature of data, computation resources, etc.

Inconsistent Data

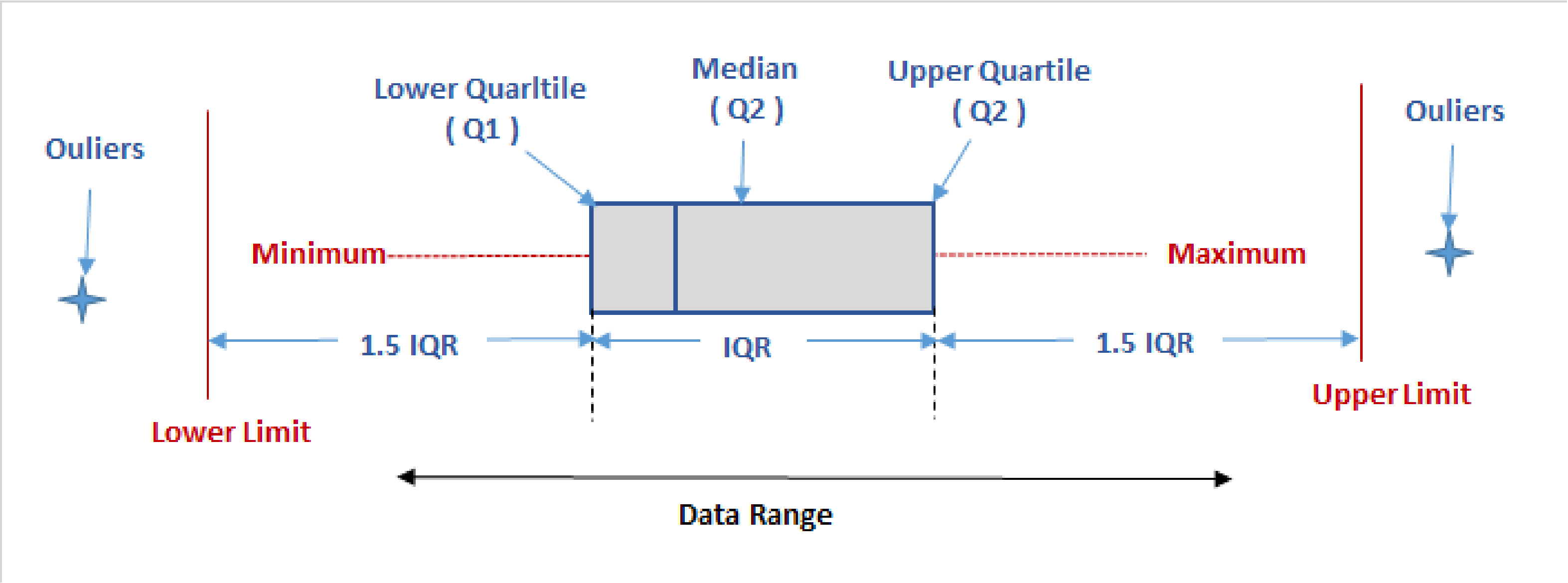
- E.g., caused by measurement errors (sensor blocked, falsely calibrated)
- Harder to detect (need feature context)
- Use imputation to fill in data, pay attention to context!

	Original Data			Remove			Impute Column Mean			kNN-Imputation (k=1)		
Item	**A**	**Temp**	**C**	**A**	**Temp**	**C**	**A**	**Temp**	**C**	**A**	**Temp**	**C**
1	12	80	1	12	80	1	12	80	1	12	80	1
2	5	4	7	5	4	7	5	4	7	5	4	7
3	4	6	6	4	6	6	4	6	6	4	6	6
4	9	**600**	15	-	-	-	9	**30**	15	9	**4**	15

- Use of strategy depends on nature of data, computation resources, etc.

Inconsistent Data – Outlier Handling

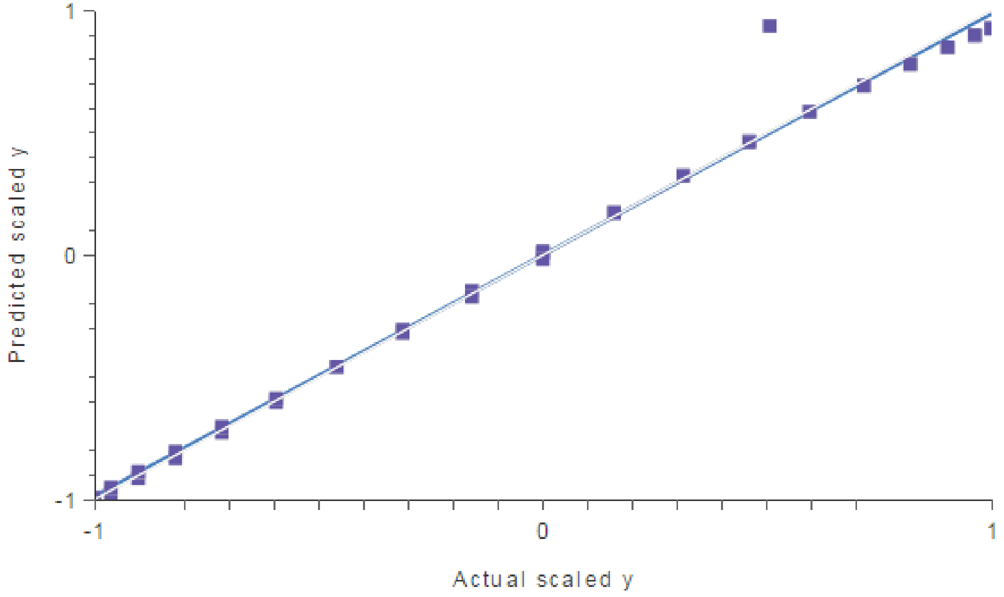
- Outlier Handling
- Detecting outliers?
- Univariate: Use boxplots!



Source: whatissixsigma.net

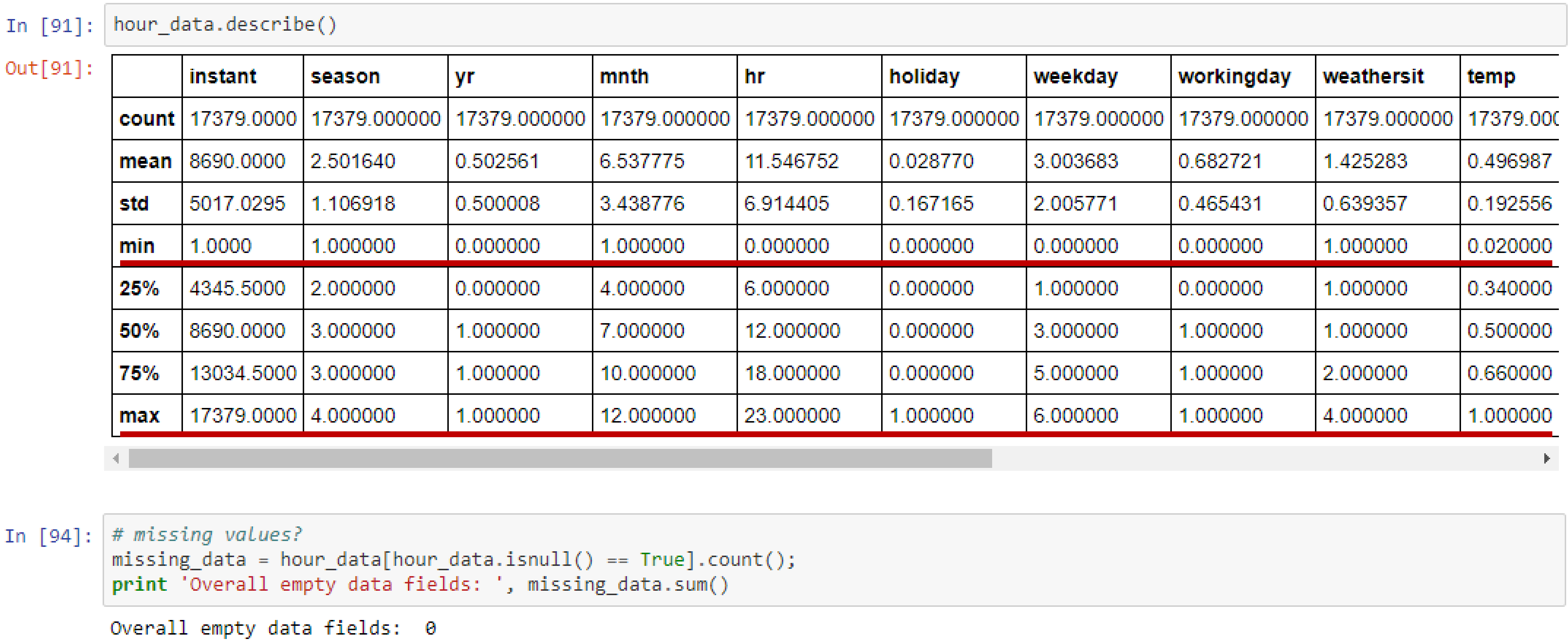
Inconsistent Data – Outlier Handling

- Multivariate: e.g., use linear regression with data point error
  - Note: there are more advanced strategies (e.g., Minkowski error)
    - Feel free to inform yourself on these – strategy selection is important here, too:
    - Outliers may actually be valuable data points!



Source: kdnuggets.com

Case Study: Cleaning Data



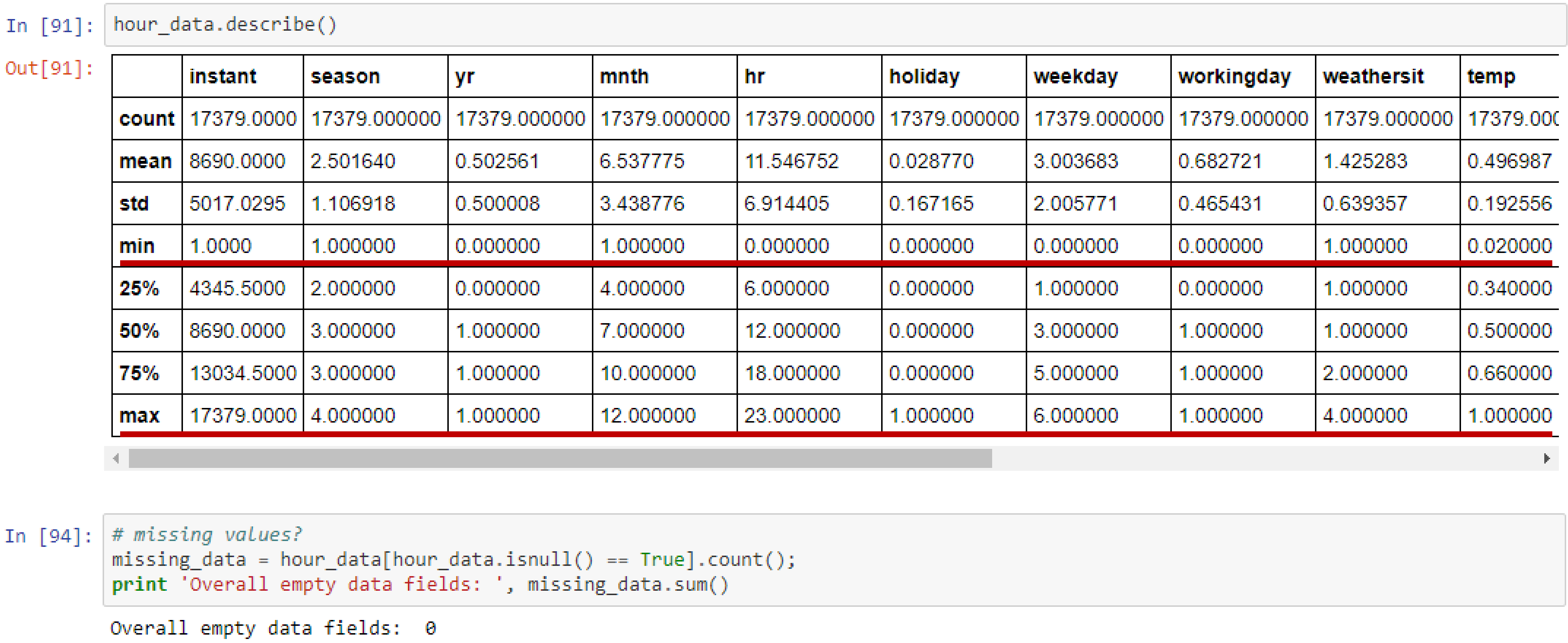
```
In [91]: hour_data.describe()
```

Out[91]:

	instant	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp
count	17379.0000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000
mean	8690.0000	2.501640	0.502561	6.537775	11.546752	0.028770	3.003683	0.682721	1.425283	0.496987
std	5017.0295	1.106918	0.500008	3.438776	6.914405	0.167165	2.005771	0.465431	0.639357	0.192556
min	1.0000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.020000
25%	4345.5000	2.000000	0.000000	4.000000	6.000000	0.000000	1.000000	0.000000	1.000000	0.340000
50%	8690.0000	3.000000	1.000000	7.000000	12.000000	0.000000	3.000000	1.000000	1.000000	0.500000
75%	13034.5000	3.000000	1.000000	10.000000	18.000000	0.000000	5.000000	1.000000	2.000000	0.660000
max	17379.0000	4.000000	1.000000	12.000000	23.000000	1.000000	6.000000	1.000000	4.000000	1.000000

```
In [94]: # missing values?
         missing_data = hour_data[hour_data.isnull() == True].count();
         print 'Overall empty data fields: ', missing_data.sum()

         Overall empty data fields:  0
```

Case Study: Cleaning Data



```
In [91]: hour_data.describe()
```

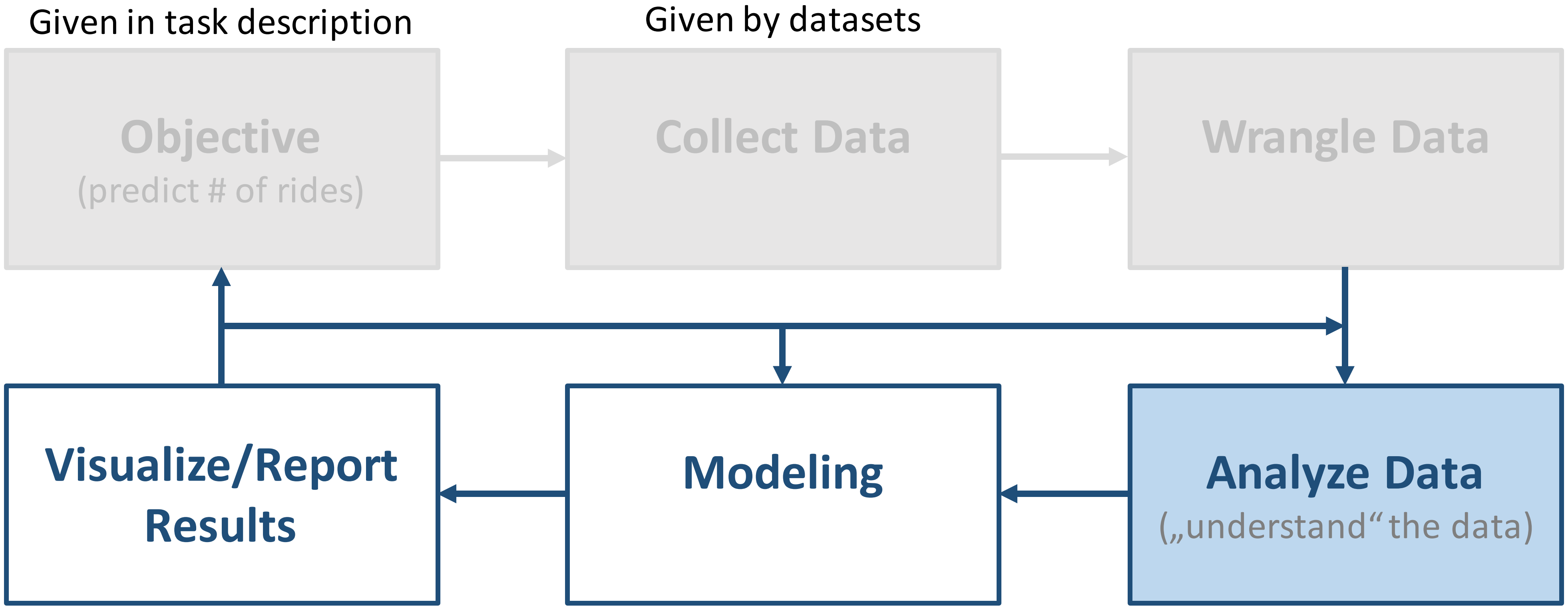
Out[91]:

	instant	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp
count	17379.0000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000
mean	8690.0000	2.501640	0.502561	6.537775	11.546752	0.028770	3.003683	0.682721	1.425283	0.496987
std	5017.0295	1.106918	0.500008	3.438776	6.914405	0.167165	2.005771	0.465431	0.639357	0.192556
min	1.0000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.020000
25%	4345.5000	2.000000	0.000000	4.000000	6.000000	0.000000	1.000000	0.000000	1.000000	0.340000
50%	8690.0000	3.000000	1.000000	7.000000	12.000000	0.000000	3.000000	1.000000	1.000000	0.500000
75%	13034.5000	3.000000	1.000000	10.000000	18.000000	0.000000	5.000000	1.000000	2.000000	0.660000
max	17379.0000	4.000000	1.000000	12.000000	23.000000	1.000000	6.000000	1.000000	4.000000	1.000000

```
In [94]: # missing values?
         missing_data = hour_data[hour_data.isnull() == True].count();
         print 'Overall empty data fields: ', missing_data.sum()

         Overall empty data fields:  0
```

- Nothing (too obvious) to do here ☺

Data Science Pipeline



Given in task description

Given by datasets

Objective

(predict # of rides)

Collect Data

Wrangle Data

Visualize/Report Results

Modeling

Analyze Data

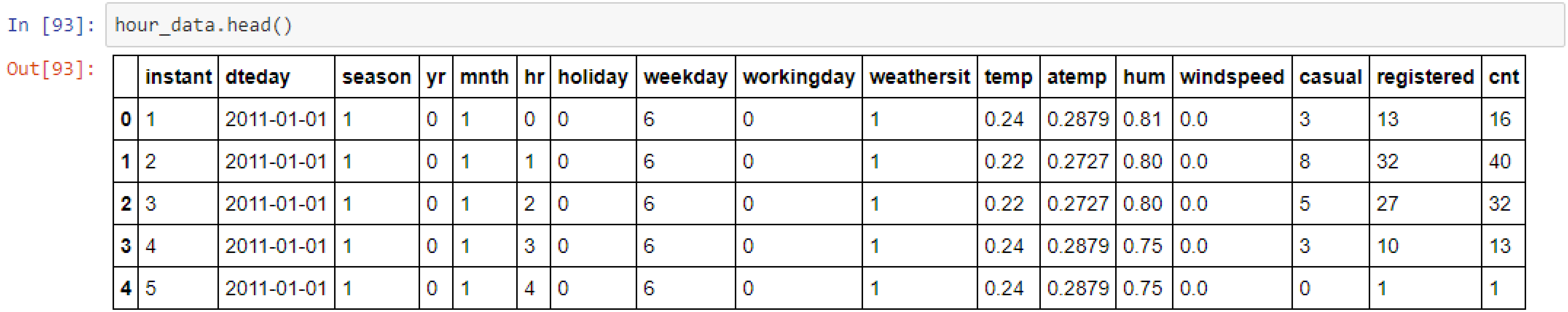
(„understand“ the data)

Analyzing Data: Exploratory Data Analysis (EDA)

- Investigating data with the goal of producing insights
- Gained knowledge should later be transformed into model features

- More colloquially: „trying to understand the data“
  - Understand the features given in the data
  - Find patterns in the data
  - Distribution of features
  - Correlation between features

EDA: A Starting Point

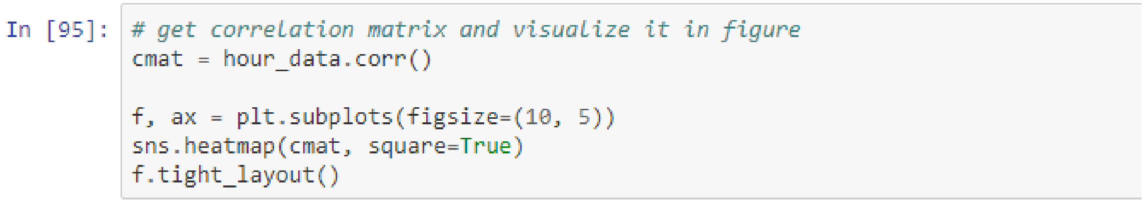


```
In [93]: hour_data.head()
```

Out[93]:

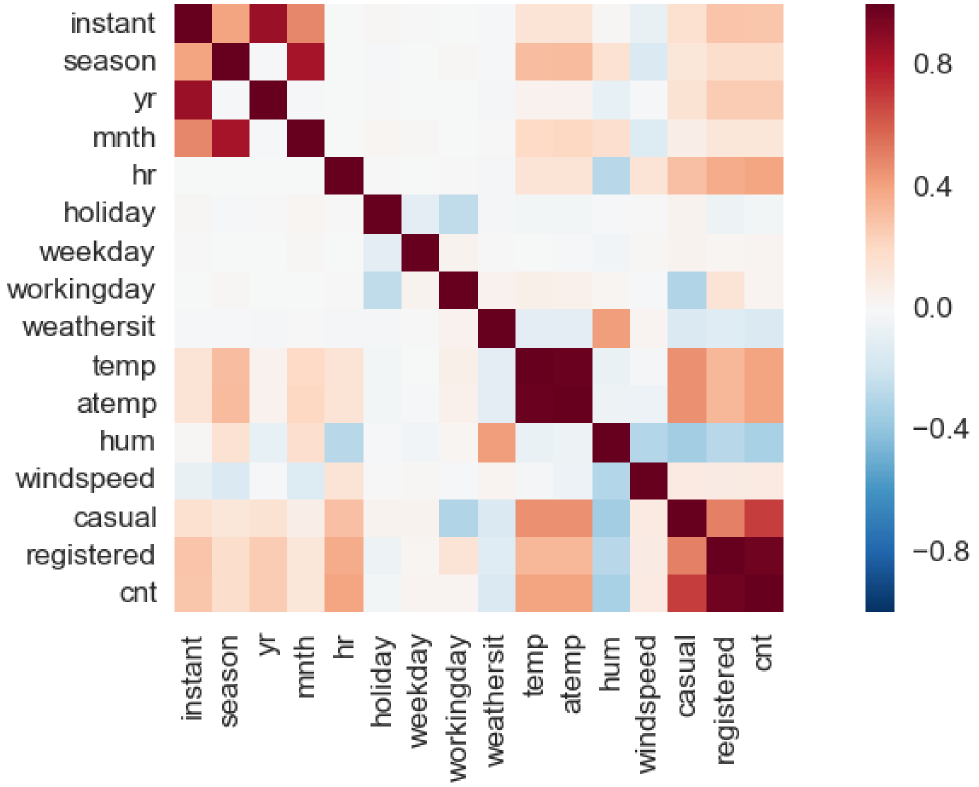
	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	0	6	0	1	0.24	0.2879	0.81	0.0	3	13	16
1	2	2011-01-01	1	0	1	1	0	6	0	1	0.22	0.2727	0.80	0.0	8	32	40
2	3	2011-01-01	1	0	1	2	0	6	0	1	0.22	0.2727	0.80	0.0	5	27	32
3	4	2011-01-01	1	0	1	3	0	6	0	1	0.24	0.2879	0.75	0.0	3	10	13
4	5	2011-01-01	1	0	1	4	0	6	0	1	0.24	0.2879	0.75	0.0	0	1	1

Case Study EDA: Correlations

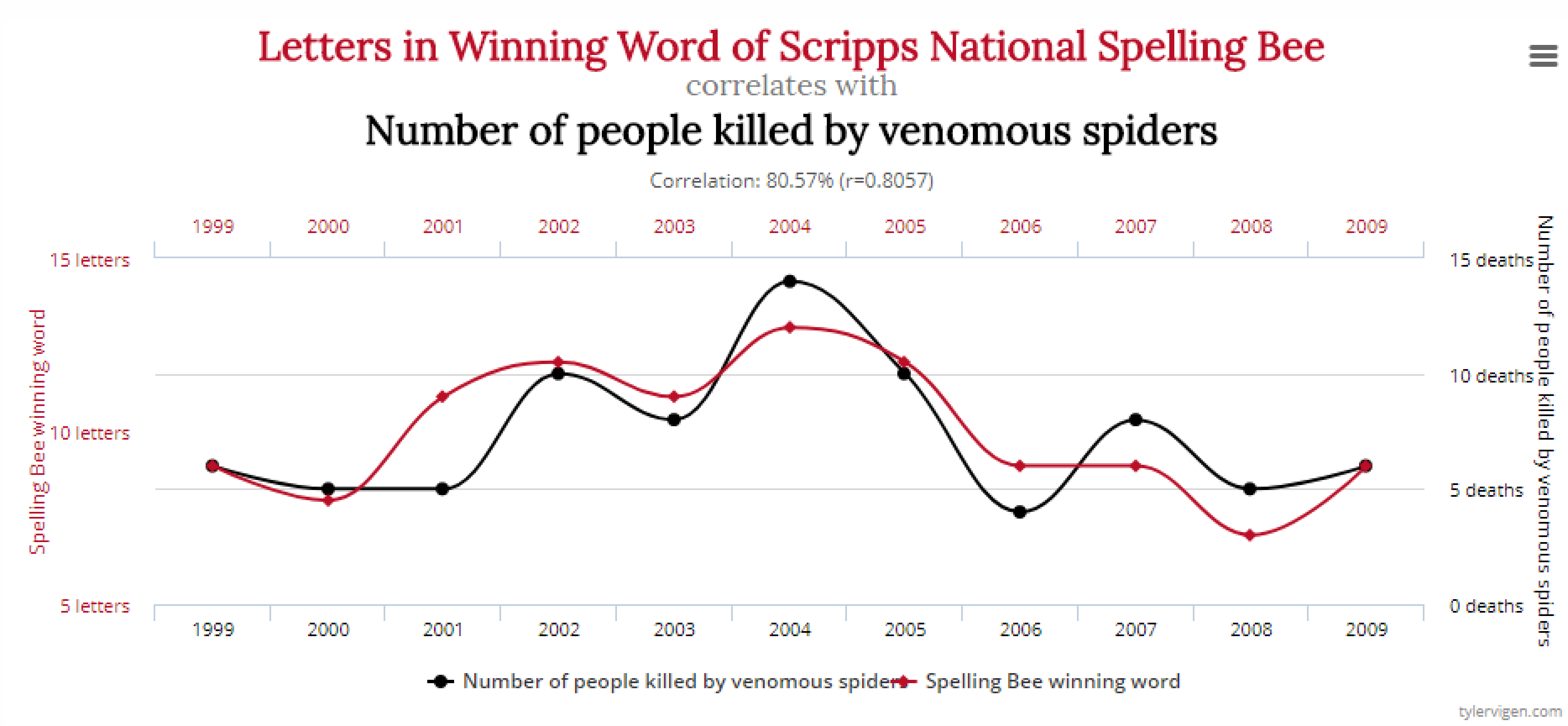


In [95]:

```
# get correlation matrix and visualize it in figure
cmat = hour_data.corr()

f, ax = plt.subplots(figsize=(10, 5))
sns.heatmap(cmat, square=True)
f.tight_layout()
```



Correlation Does Not Imply Causation!



Letters in Winning Word of Scripps National Spelling Bee

correlates with

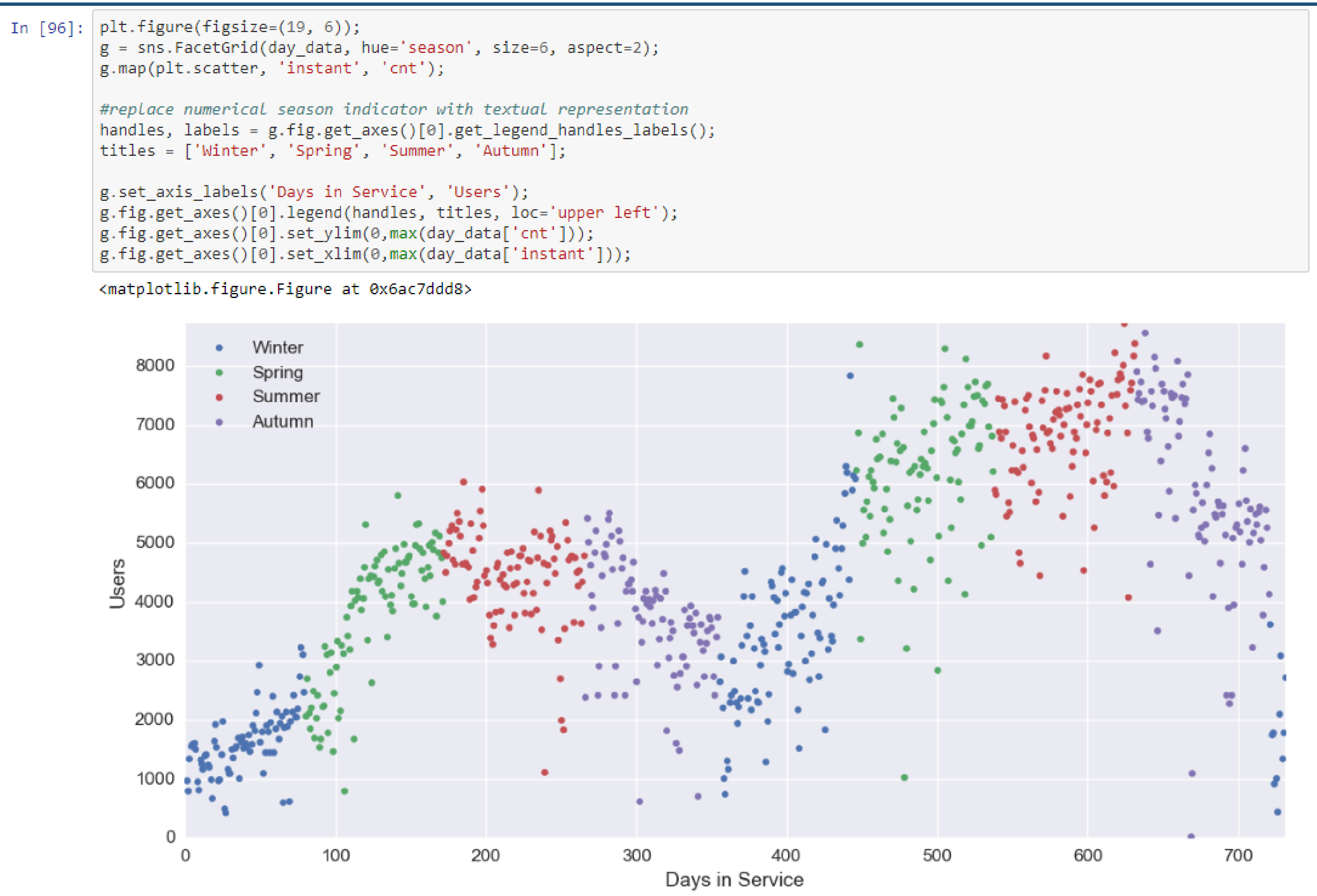
Number of people killed by venomous spiders

Correlation: 80.57% (r=0.8057)

Data sources: National Spelling Bee and Centers for Disease Control & Prevention

http://www.tylervigen.com/spurious-correlations

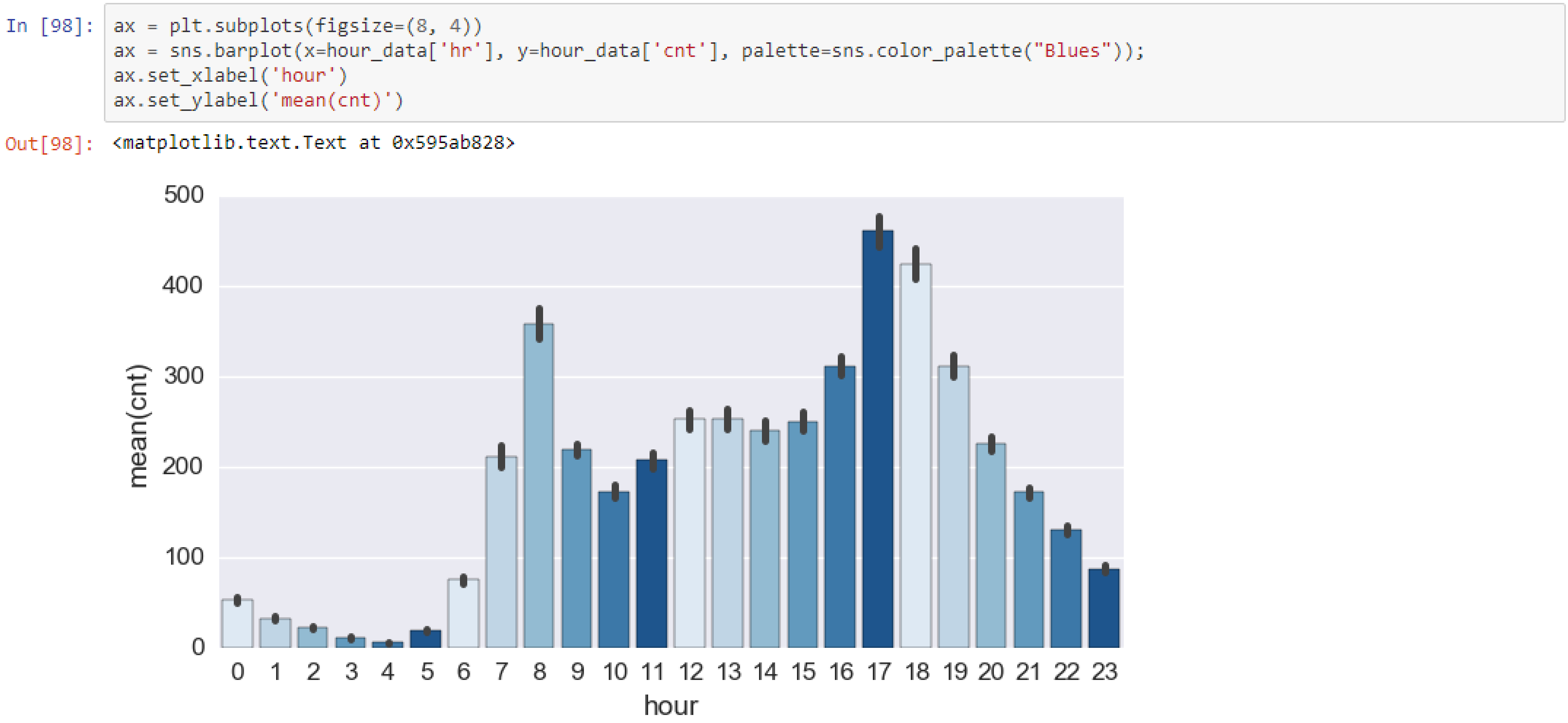
Case Study: Does the season affect bike usage?



```
In [96]: plt.figure(figsize=(19, 6));
         g = sns.FacetGrid(day_data, hue='season', size=6, aspect=2);
         g.map(plt.scatter, 'instant', 'cnt');

         #replace numerical season indicator with textual representation
         handles, labels = g.fig.get_axes()[0].get_legend_handles_labels();
         titles = ['Winter', 'Spring', 'Summer', 'Autumn'];

         g.set_axis_labels('Days in Service', 'Users');
         g.fig.get_axes()[0].legend(handles, titles, loc='upper left');
         g.fig.get_axes()[0].set_ylim(0,max(day_data['cnt']));
         g.fig.get_axes()[0].set_xlim(0,max(day_data['instant']));
```

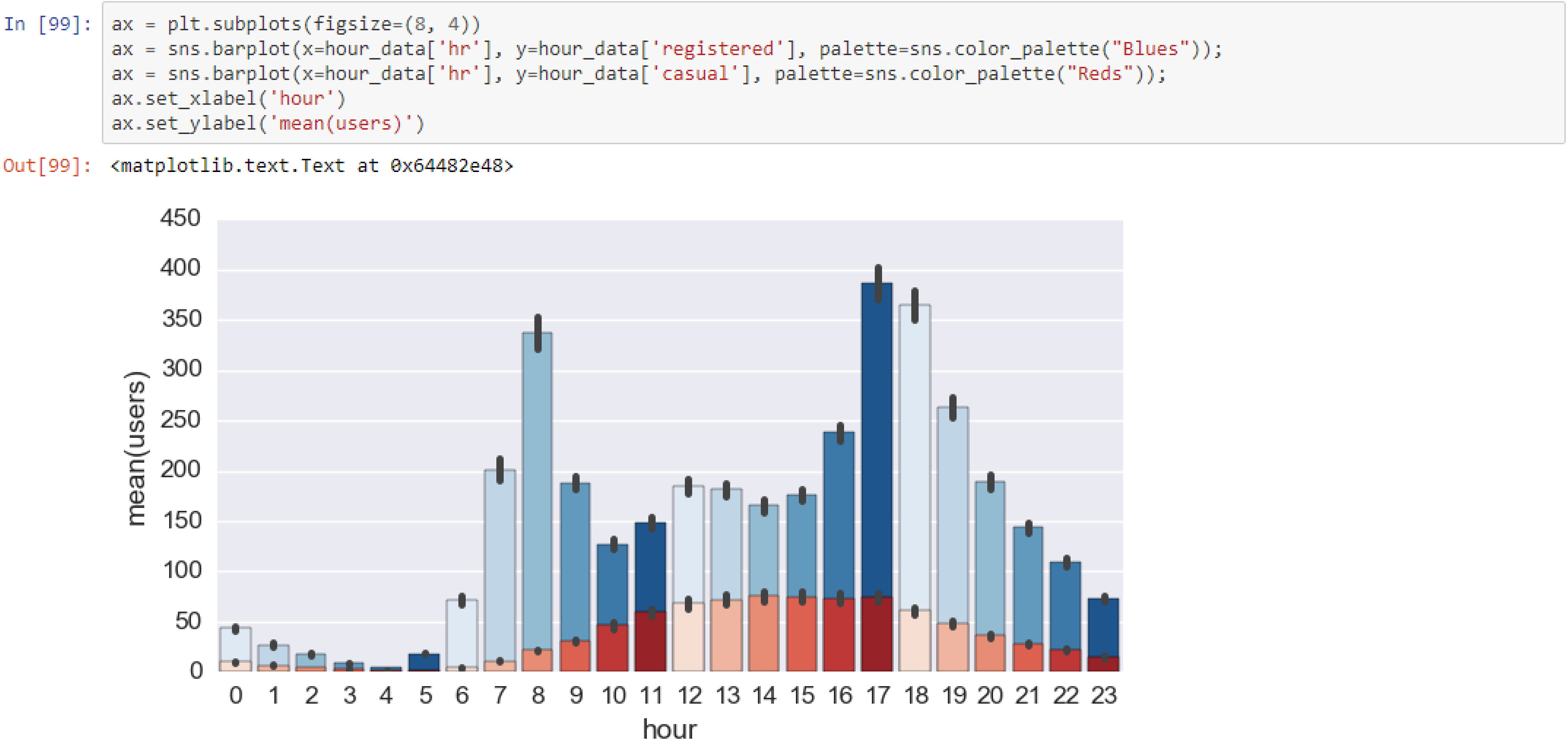
```
<matplotlib.figure.Figure at 0x6ac7ddd8>
```

Case Study: Does the time of day affect bike usage?



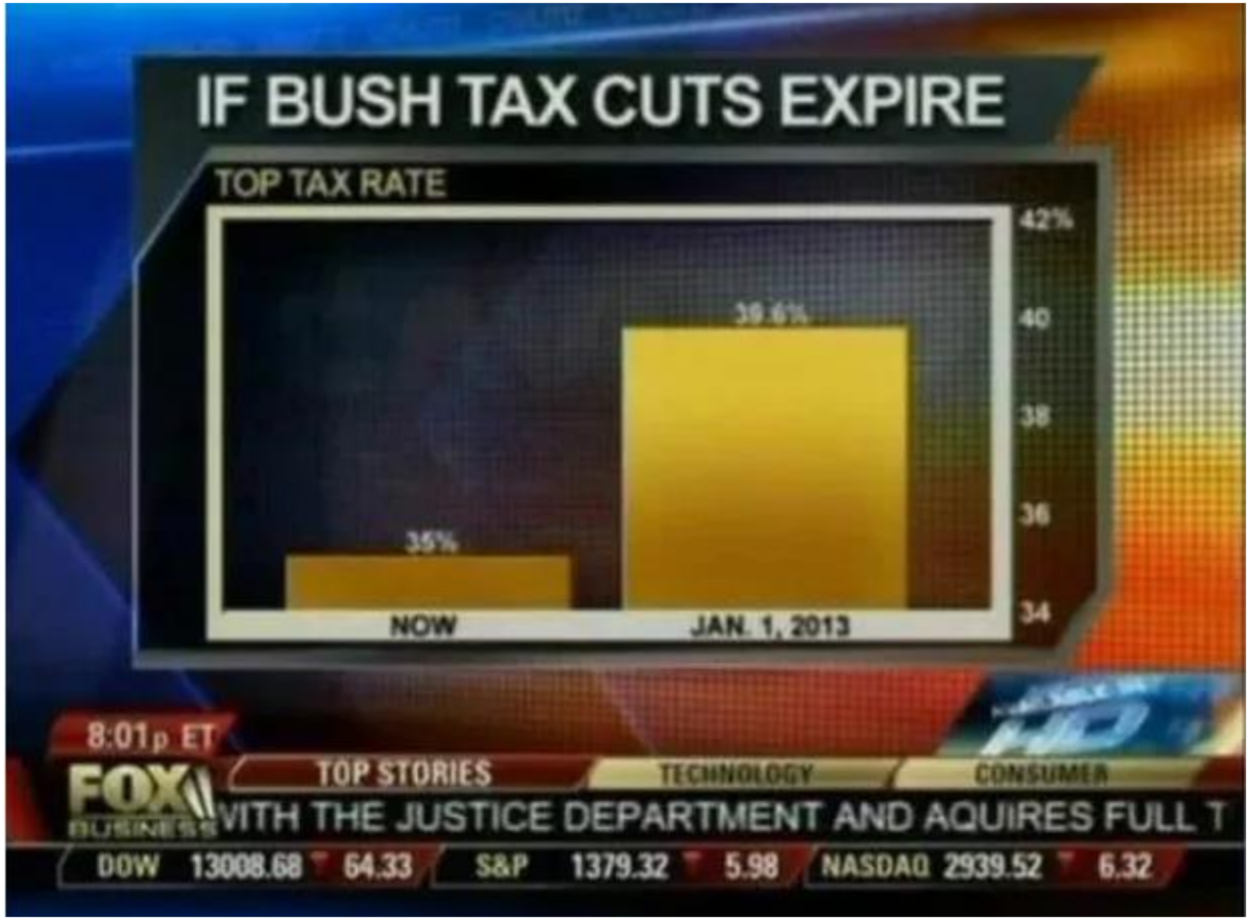
```
In [98]: ax = plt.subplots(figsize=(8, 4))
         ax = sns.barplot(x=hour_data['hr'], y=hour_data['cnt'], palette=sns.color_palette("Blues"));
         ax.set_xlabel('hour')
         ax.set_ylabel('mean(cnt)')

Out[98]: <matplotlib.text.Text at 0x595ab828>
```

Case Study: Difference between user classes?



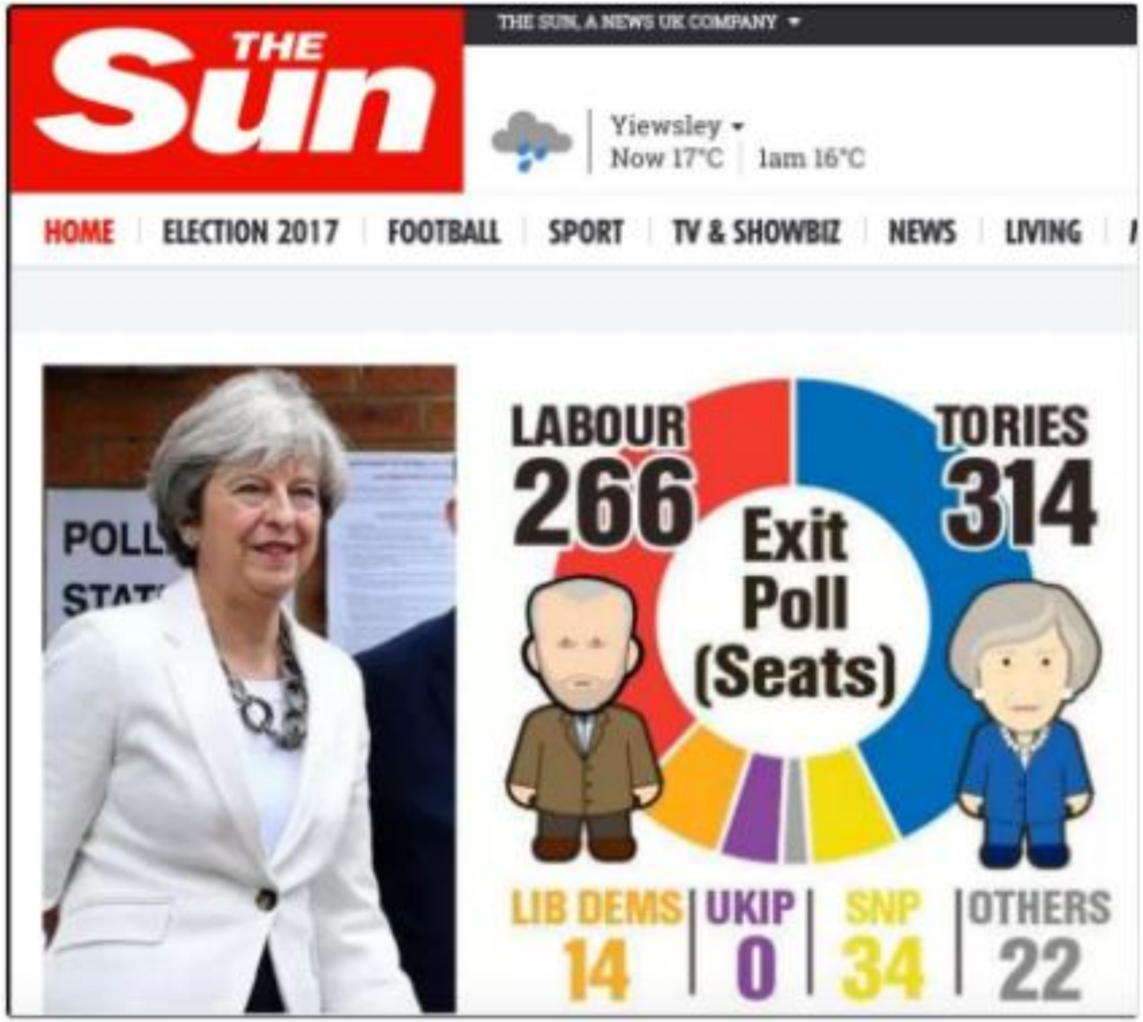
```
In [99]: ax = plt.subplots(figsize=(8, 4))
         ax = sns.barplot(x=hour_data['hr'], y=hour_data['registered'], palette=sns.color_palette("Blues"));
         ax = sns.barplot(x=hour_data['hr'], y=hour_data['casual'], palette=sns.color_palette("Reds"));
         ax.set_xlabel('hour')
         ax.set_ylabel('mean(users)')

Out[99]: <matplotlib.text.Text at 0x64482e48>
```

EDA: Visualizations –Bad Plots

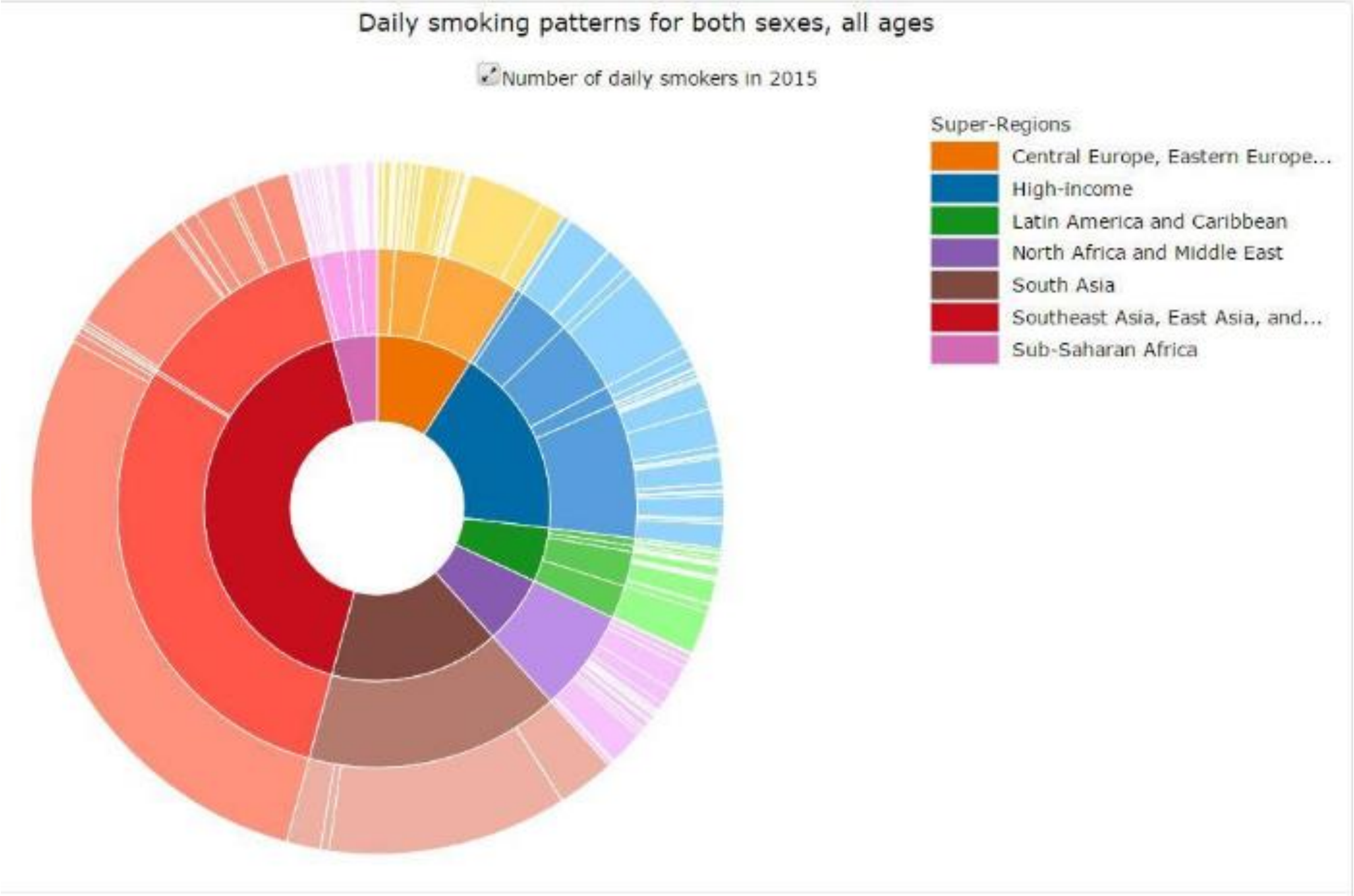


Note: all visualizations taken from http://wtfviz.net

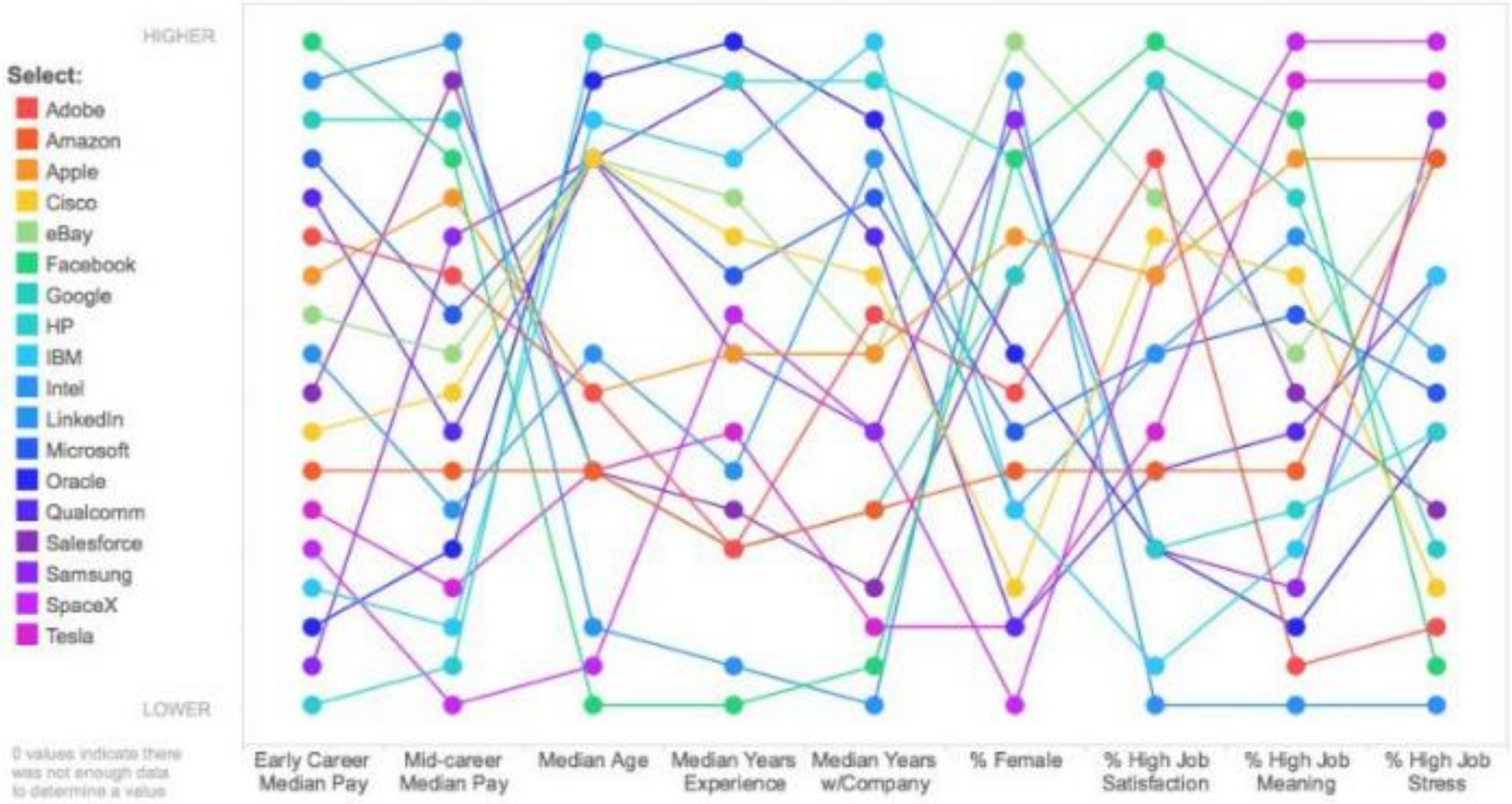
EDA: Visualizations – Bad Plots



EDA: Visualizations – Bad Plots



EDA: Visualizations – Bad Plots

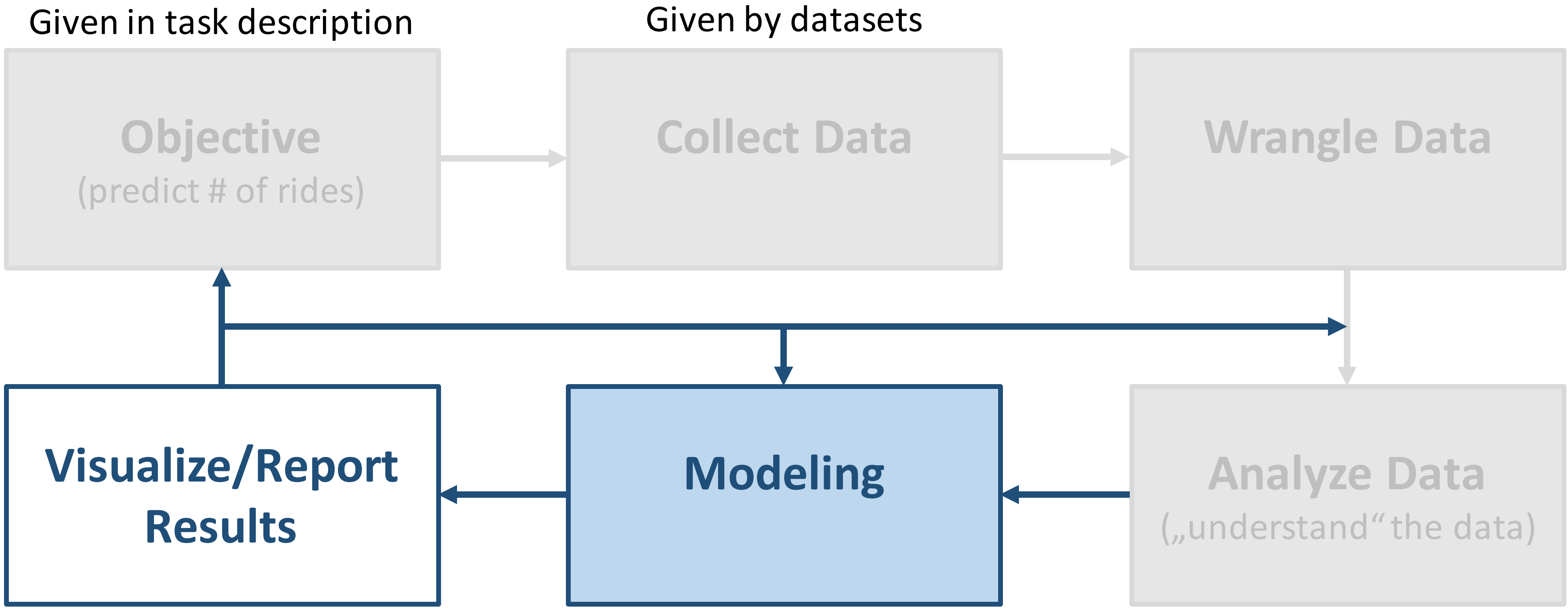


EDA: Plotting Do‘s and Don‘ts

- Do ensure your visualization answers a question
- Do aim for simplicity, consistency
- Do carefully select the right type of plot for your statement
- Do provide consistent and clear labels
- Do make your charts readable

- Don‘t overload graphs
- Don‘t use too many colors
- Don‘t distort data (setting bad axis limits, scales, etc.)
- Don‘t change your style in the course of a project
- Don‘t make readers guess (see clear labels)

Case Study: Next Step


Given in task description

Given by datasets

Objective

(predict # of rides)

Collect Data

Wrangle Data

Visualize/Report Results

Modeling

Analyze Data

(„understand“ the data)

Predictive Modeling

- Major steps: feature engineering and model building

- Feature engineering: turn insights from EDA into features that help with prediction

> Coming up with features is difficult, time-consuming, requires expert knowledge.
> "Applied machine learning" is basically feature engineering.
>
> — Andrew Ng (Co-Founder of Coursera, Chief Scientist at Baidu)

- Critical: domain knowledge

Feature Engineering – Common Practices

- Features are usually engineered as functions of already existing features
- Alternatively: create features from external data sources

- Possible functions: ratios, differences, categorizations, …
- Possible external sources: event databases, weather databases, …

- A creative process!

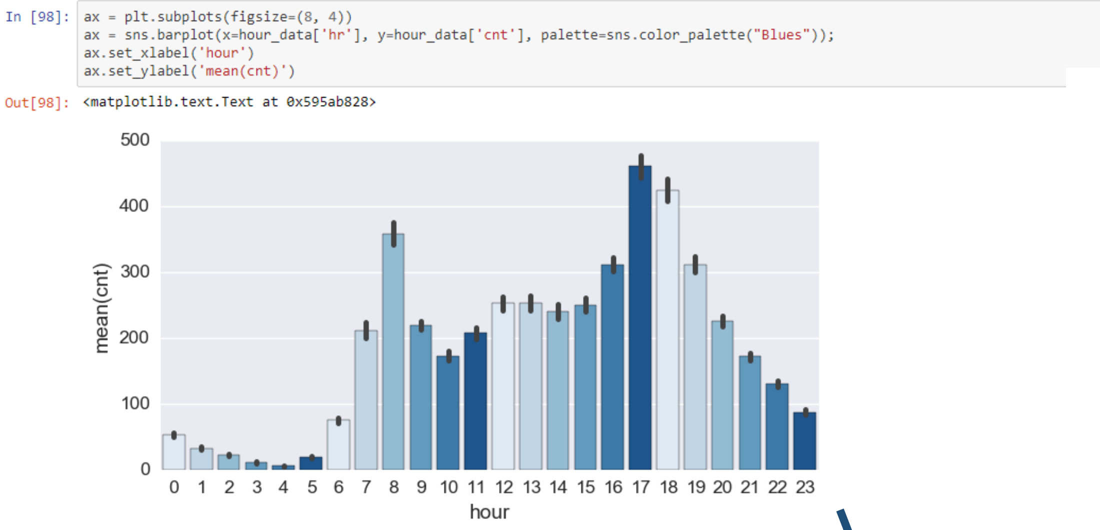
- Sidenote: neural networks can learn features themselves
  - In many cases better than those created by engineering – drawbacks?

Feature Engineering – Examples

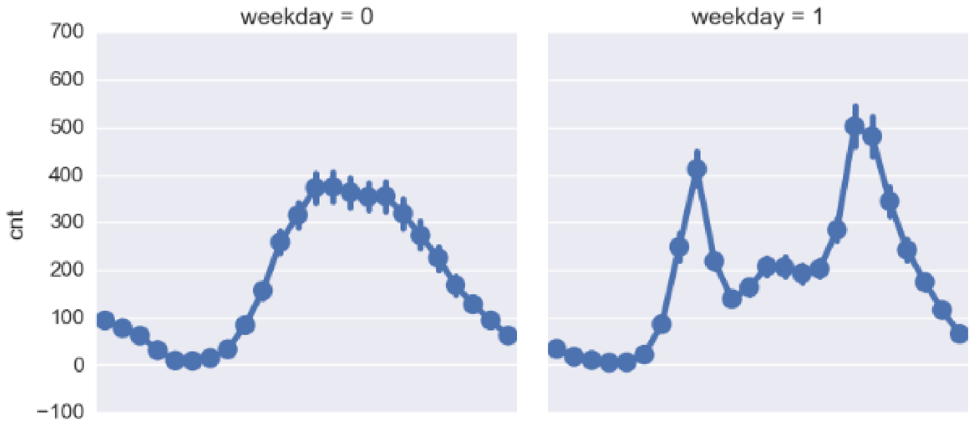
- Consider Twitter:
  - Goal: predict which users are the most influential users in the network
  - Dataset features: number of followers, number of followees
  - Number of followers expressive?
    - Yes, to a certain extent: often following exchanges to boost follower numbers
  - Better feature: **ratio** of followers to followees
    - Intuition behind: really influential users have much more followers than followees

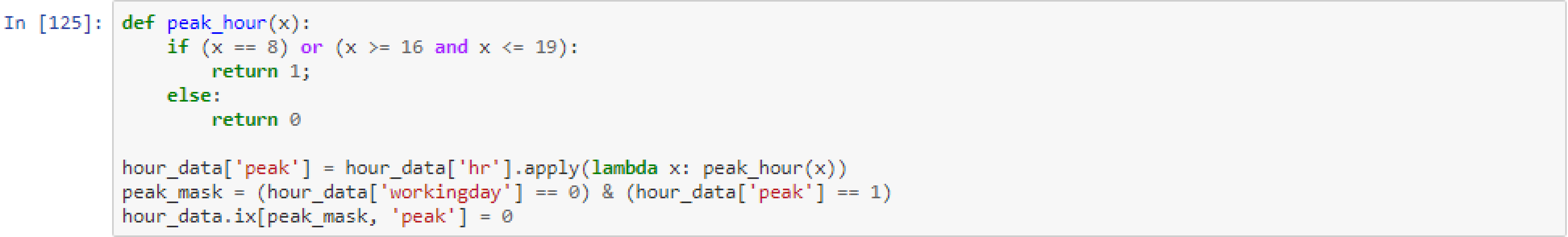
- Consider Intrusion Detection Systems (IDS):
  - Goal: given on packet level data, predict whether packet is malicious
  - Dataset features: packet length, header fields, packet type
  - Good features: **classifications** based on different combinations of features
    - Intuition behind: attacking packets typically follow a certain signature

Case Study: Feature Engineering



```
In [98]:  ax = plt.subplots(figsize=(8, 4))
          ax = sns.barplot(x=hour_data['hr'], y=hour_data['cnt'], palette=sns.color_palette("Blues"));
          ax.set_xlabel('hour')
          ax.set_ylabel('mean(cnt)')

Out[98]:  <matplotlib.text.Text at 0x595ab828>
```





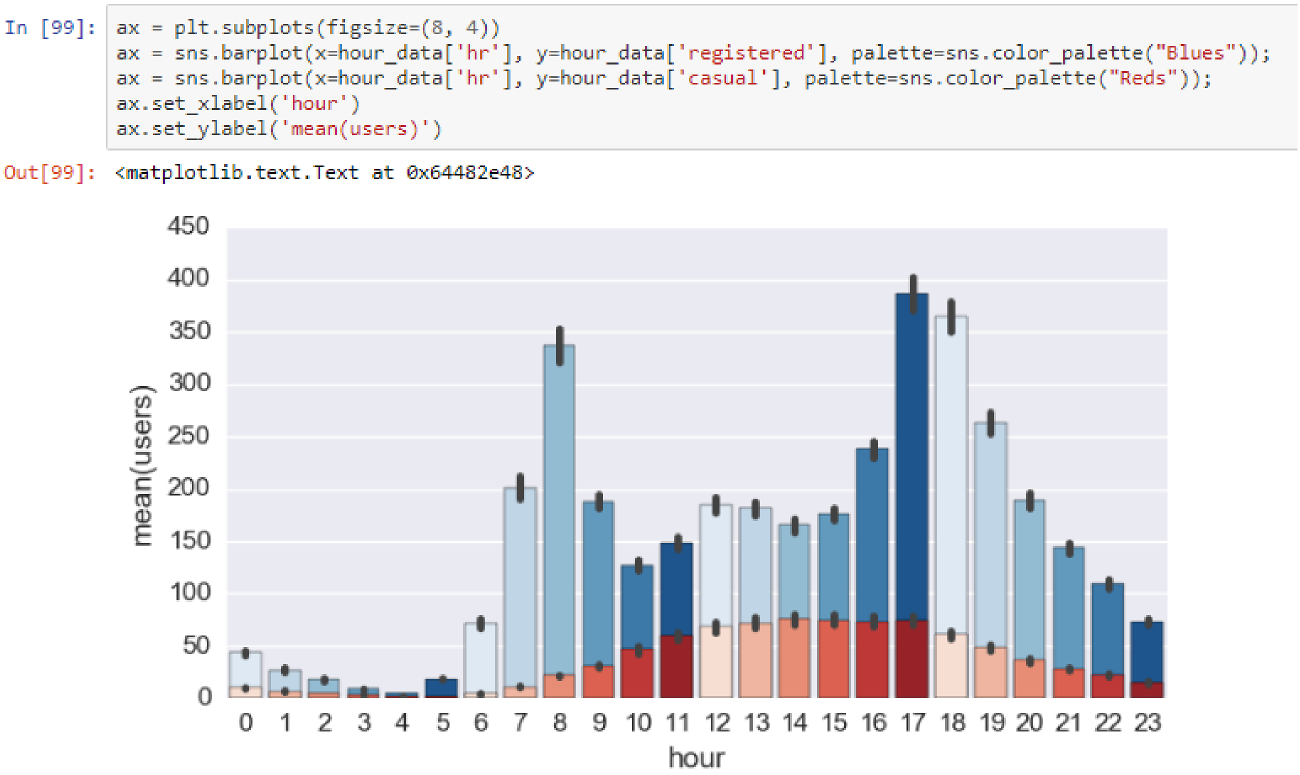
```
In [125]: def peak_hour(x):
              if (x == 8) or (x >= 16 and x <= 19):
                  return 1;
              else:
                  return 0

          hour_data['peak'] = hour_data['hr'].apply(lambda x: peak_hour(x))
          peak_mask = (hour_data['workingday'] == 0) & (hour_data['peak'] == 1)
          hour_data.ix[peak_mask, 'peak'] = 0
```

Building A Model

1. Have a good idea for how to model and evaluate the task.
2. Pick the right algorithm for the task.
3. Split the data into train and test (and possibly validation) data
4. Fit model on train data
5. Test the model on the test data
6. If result is not good enough: return to EDA / feature engineering

Case Study: Building A Model

- Have a good idea to model the task
  - Usually result of EDA
  - In this case: build two models, one for registered, one for casual users



```
In [99]:  ax = plt.subplots(figsize=(8, 4))
          ax = sns.barplot(x=hour_data['hr'], y=hour_data['registered'], palette=sns.color_palette("Blues"));
          ax = sns.barplot(x=hour_data['hr'], y=hour_data['casual'], palette=sns.color_palette("Reds"));
          ax.set_xlabel('hour')
          ax.set_ylabel('mean(users)')

Out[99]:  <matplotlib.text.Text at 0x64482e48>
```

Case Study: Building A Model

- Have a good idea to evaluate the model
  - A lot of metrics available:
    - Mean Absolute Error (MAE)
    - Mean Squared Error (MSE)
    - Area under ROC (AUC)
    - F1-Score
    - … (see later lectures)
  - In this case: our predicted numbers should be as close to reality as possible
    - Need: low average error
    - Decision: use MAE (why: ideas? If not, see later lectures)

Case Study: Building A Model

- Pick the right algorithm for the task.
  - Regression vs classification vs image detection vs…

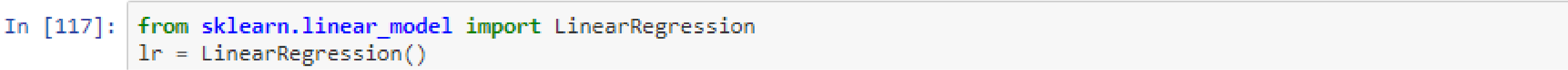
„We need a *prediction of usage numbers* for our system for every hour of the week“

- Clearly a regression problem!
- Would applying a classification model work?

- Possible candidates?
  - Linear regression, Random Forest regression, kNN regression, …

Case Study: Building A Model

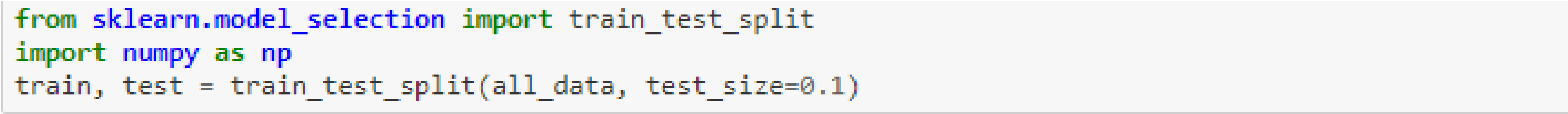
- Pick the right algorithm for the task.



```
In [117]: from sklearn.linear_model import LinearRegression
          lr = LinearRegression()
```

Case Study: Building A Model

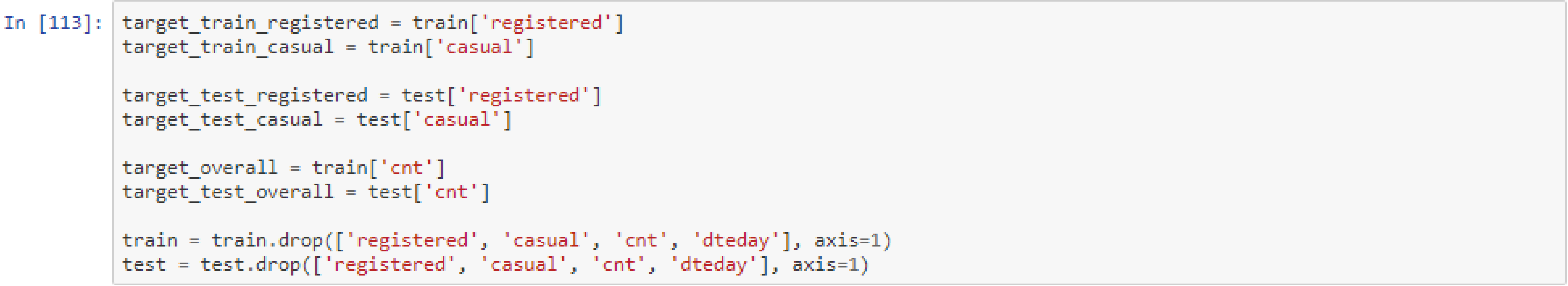
- Split your data into train and test (and possibly validation) data



```
from sklearn.model_selection import train_test_split
import numpy as np
train, test = train_test_split(all_data, test_size=0.1)
```

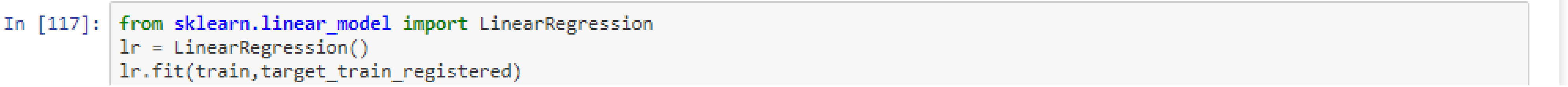
Case Study: Building A Model

- Fit the model on train data
  - First step: determine what needs to be predicted (*target*)



```
In [113]: target_train_registered = train['registered']
          target_train_casual = train['casual']

          target_test_registered = test['registered']
          target_test_casual = test['casual']

          target_overall = train['cnt']
          target_test_overall = test['cnt']

          train = train.drop(['registered', 'casual', 'cnt', 'dteday'], axis=1)
          test = test.drop(['registered', 'casual', 'cnt', 'dteday'], axis=1)
```

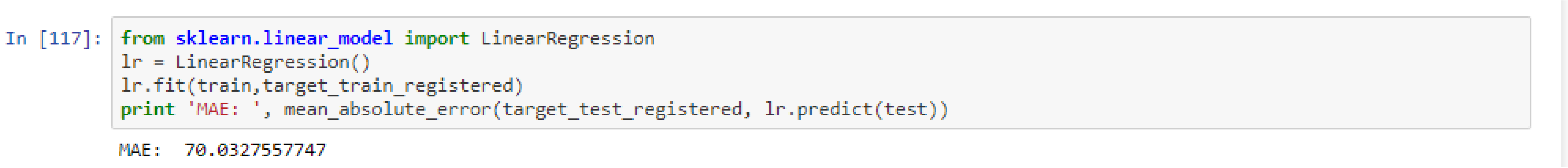
  - Then: fit model so that cost function is minimized wrt target



```
In [117]: from sklearn.linear_model import LinearRegression
          lr = LinearRegression()
          lr.fit(train,target_train_registered)
```

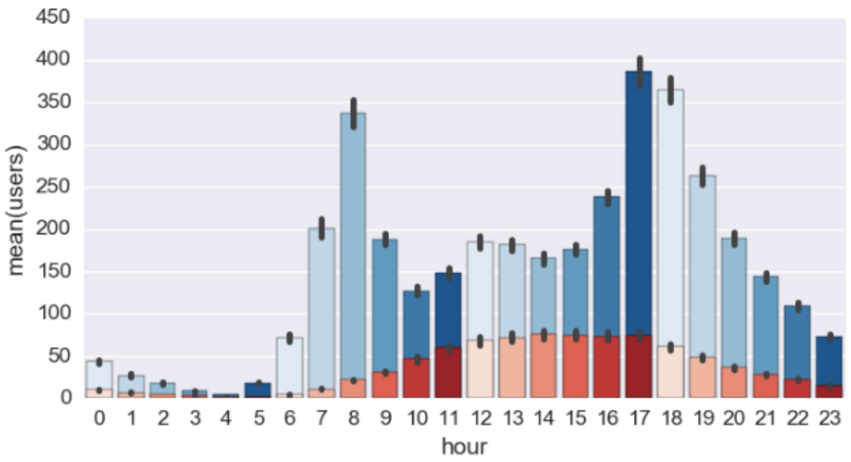
Case Study: Building A Model

- Test the model on test data wrt evaluation metric



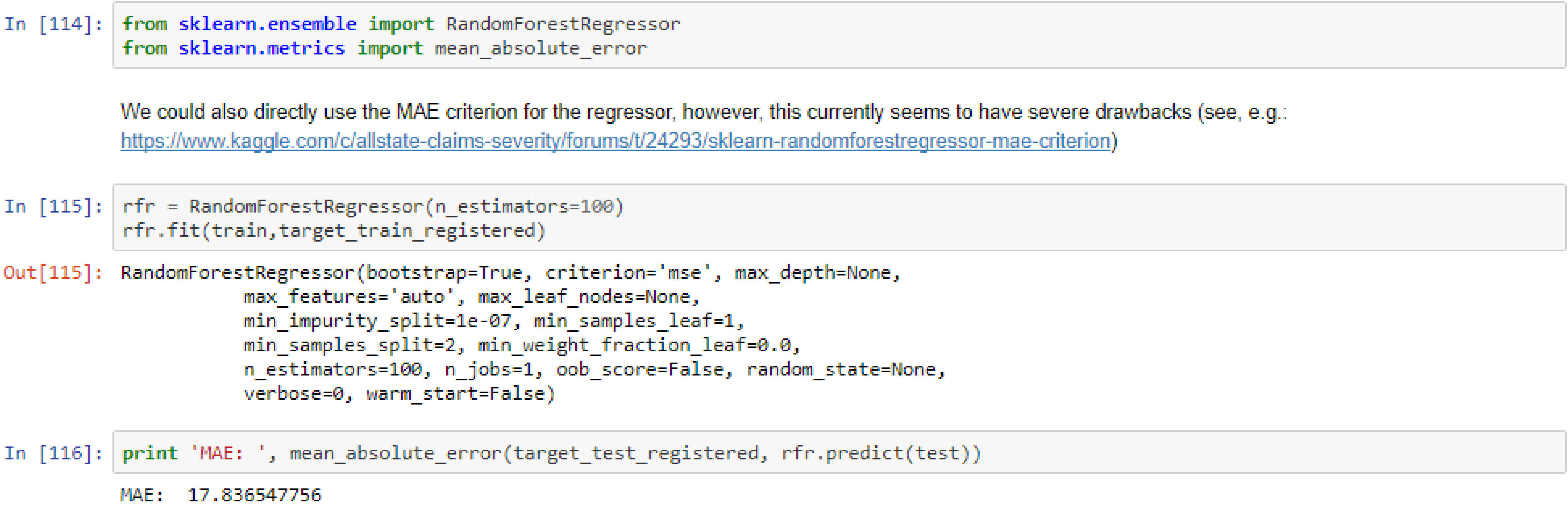
```
In [117]: from sklearn.linear_model import LinearRegression
          lr = LinearRegression()
          lr.fit(train,target_train_registered)
          print 'MAE: ', mean_absolute_error(target_test_registered, lr.predict(test))

          MAE:  70.0327557747
```

- MAE = 70.03 – our hourly prediction is 70 users off by average.
  - Is this good enough?
  - Average user count:
    - ~200 -> more than 30% error
    - Clearly not good enough!



Case Study: Building A Model

- Let‘s try a different algorithm



```
In [114]: from sklearn.ensemble import RandomForestRegressor
          from sklearn.metrics import mean_absolute_error
```

We could also directly use the MAE criterion for the regressor, however, this currently seems to have severe drawbacks (see, e.g.: https://www.kaggle.com/c/allstate-claims-severity/forums/t/24293/sklearn-randomforestregressor-mae-criterion)

```
In [115]: rfr = RandomForestRegressor(n_estimators=100)
          rfr.fit(train,target_train_registered)

Out[115]: RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                     max_features='auto', max_leaf_nodes=None,
                     min_impurity_split=1e-07, min_samples_leaf=1,
                     min_samples_split=2, min_weight_fraction_leaf=0.0,
                     n_estimators=100, n_jobs=1, oob_score=False, random_state=None,
                     verbose=0, warm_start=False)

In [116]: print 'MAE: ', mean_absolute_error(target_test_registered, rfr.predict(test))

          MAE:  17.836547756
```

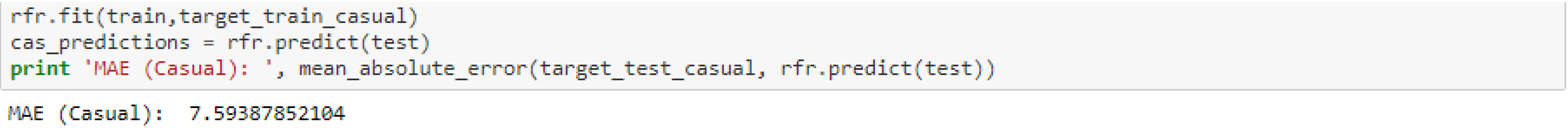
- MAE = 17.84
  - A huge improvement simply by taking a ‚better‘ algorithm.

Case Study: Building A Model

- Test the model on test data wrt evaluation metric
  - Used standard parameters for RandomForest regressor
  - Result can be further improved by tuning parameters and coming up with additional features.
  - Explanation of algorithm & parameter tuning: see later lectures

Case Study: Building A Model

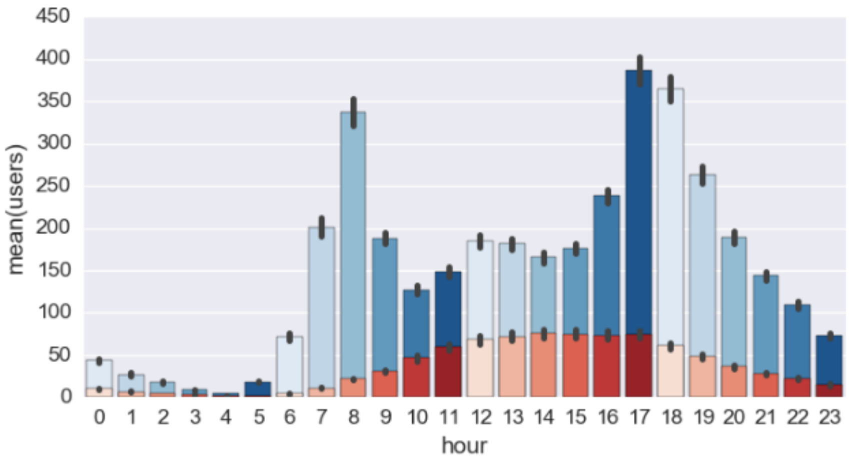
- Repeat for Casual users:



```
rfr.fit(train,target_train_casual)
cas_predictions = rfr.predict(test)
print 'MAE (Casual): ', mean_absolute_error(target_test_casual, rfr.predict(test))
```

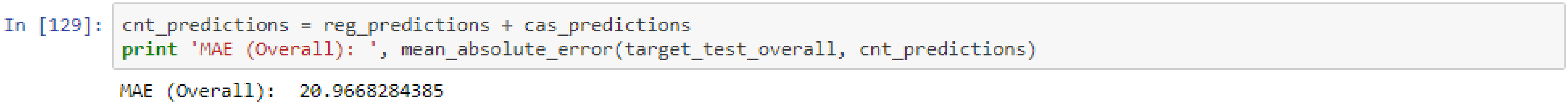
```
MAE (Casual):  7.59387852104
```

- Model is not better:
  - Counts are also lower!



Case Study: Building A Model

- Putting things together:
  - So far, two different counts from two single models.
  - Need to blend these counts into each other (you might even call it *ensembling*)
  - Here: simple sum

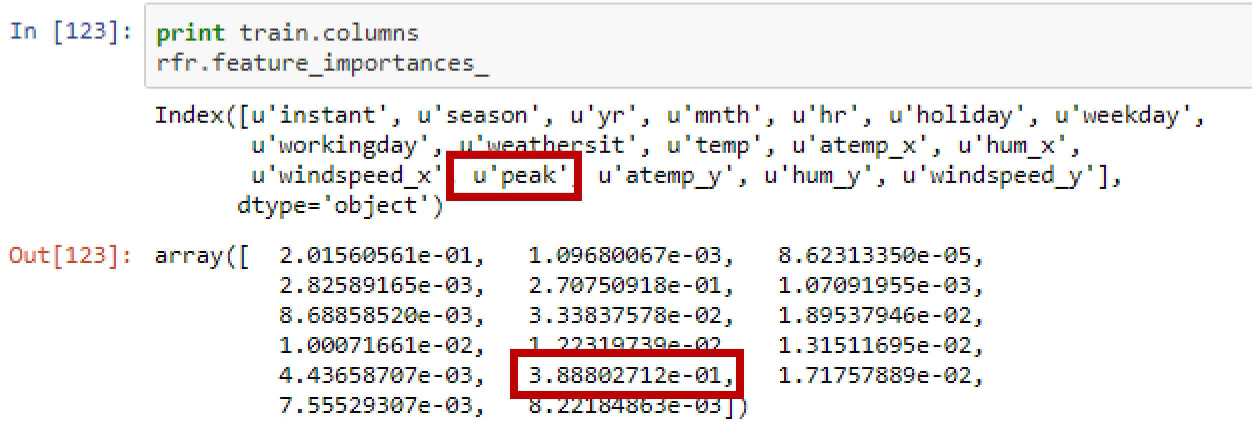


```
In [129]: cnt_predictions = reg_predictions + cas_predictions
          print 'MAE (Overall): ', mean_absolute_error(target_test_overall, cnt_predictions)

          MAE (Overall):  20.9668284385
```

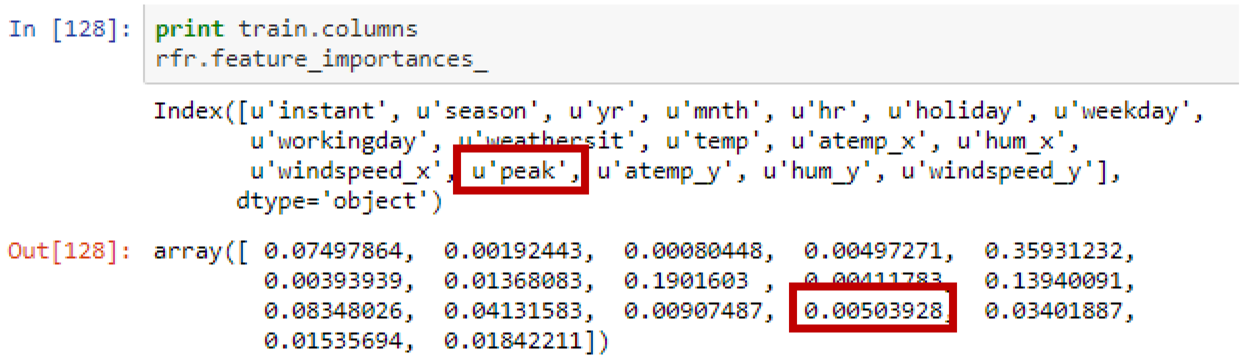
- For every hour, our model can on average predict the accurate ride count with an error of ~21 rides!
  - Good enough?

Case Study: Building A Model

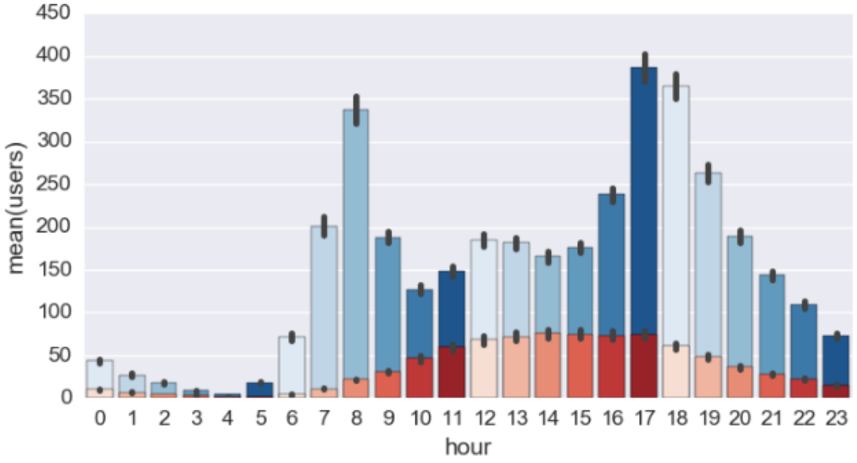
- Evaluating the model (aka: Are our features useful?)



```
In [123]: print train.columns
          rfr.feature_importances_

          Index([u'instant', u'season', u'yr', u'mnth', u'hr', u'holiday', u'weekday',
                 u'workingday', u'weathersit', u'temp', u'atemp_x', u'hum_x',
                 u'windspeed_x', u'peak', u'atemp_y', u'hum_y', u'windspeed_y'],
                dtype='object')

Out[123]: array([  2.01560561e-01,   1.09680067e-03,   8.62313350e-05,
                   2.82589165e-03,   2.70750918e-01,   1.07091955e-03,
                   8.68858520e-03,   3.33837578e-02,   1.89537946e-02,
                   1.00071661e-02,   1.22319739e-02,   1.31511695e-02,
                   4.43658707e-03,   3.88802712e-01,   1.71757889e-02,
                   7.55529307e-03,   8.22184863e-03])
```



```
In [128]: print train.columns
          rfr.feature_importances_

          Index([u'instant', u'season', u'yr', u'mnth', u'hr', u'holiday', u'weekday',
                 u'workingday', u'weathersit', u'temp', u'atemp_x', u'hum_x',
                 u'windspeed_x', u'peak', u'atemp_y', u'hum_y', u'windspeed_y'],
                dtype='object')

Out[128]: array([ 0.07497864,  0.00192443,  0.00080448,  0.00497271,  0.35931232,
                  0.00393939,  0.01368083,  0.1901603 ,  0.00411783,  0.13940091,
                  0.08348026,  0.04131583,  0.00907487,  0.00503928,  0.03401887,
                  0.01535694,  0.01842211])
```



Summary

- In this course, we focus on two key aspects of Data Science:
  - Exploratory Data Analysis (EDA)
  - Predictive Modeling
  - …and how to combine them such that the model benefits from EDA

- We have…
  - …seen a case study of how to explore a simple dataset
  - …transformed results from EDA into features
  - …built a simple, yet expressive model and evaluated an error metric

Summary

- Only used four tools/libraries:
  - *IPython Notebook / Jupyter Notebook*
  - *Pandas* for data manipulation and exploration
  - *Seaborn* for data visualization
  - *Scikit-learn* for predictive modeling and evaluation

- Next week: An overview of the Python Data Science Stack
- Further lectures:
  - Algorithms: How do RandomForests (and others) work?
  - Model evaluation: How to choose metrics, how to prevent overfitting, etc.

Task 1

- Basically: Recap what you have learned today
  - Next week: introduction of Python libraries to do the job
  - Instead of bike sharing: predict wine quality

- We use Kaggle InClass for submissions.
  - You need to register at Kaggle.com (please use university email address)
  - Please set your screenname (not username) to something that identifies you
    - E.g., first name + last name initial, matriculation number, …
    - Or let me know by email which name you are submitting under

- Task description, data, etc at:

https://www.kaggle.com/c/m-inf-1800-ws-17-18-task-1/

Task 1

- Start date: now!
- Deadline: Wed, Nov 15th, 23:59 (4 weeks)
  - Kaggle will not accept further submissions after that deadline, and neither will I
- For this task: no teams
- You will have 5 submissions to the system every day
- You can select 2 submissions for final grading in the end
- Presentations or Submissions (will be decided after 2 weeks): Nov 16th