Introduction to Data Science

JP-Morgan workshop

CAMBRIDGE SPARK

@cambridgespark

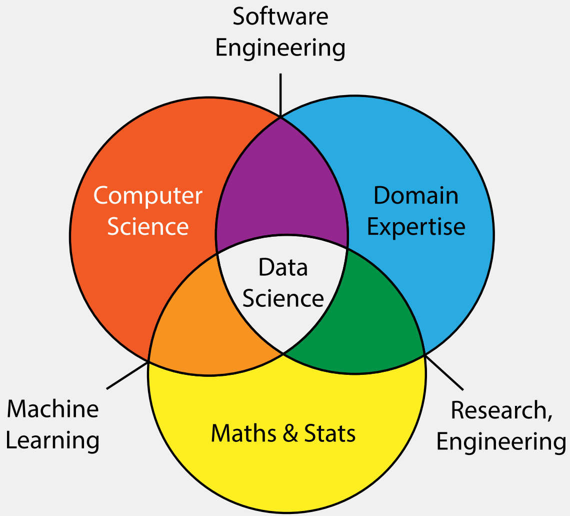
Schedule

- Exploratory Data Analysis (EDA) in Jupyter with Pandas
- Classification methods and the Decision Tree
- Ensemble methods and the Random Forest
- Competition!

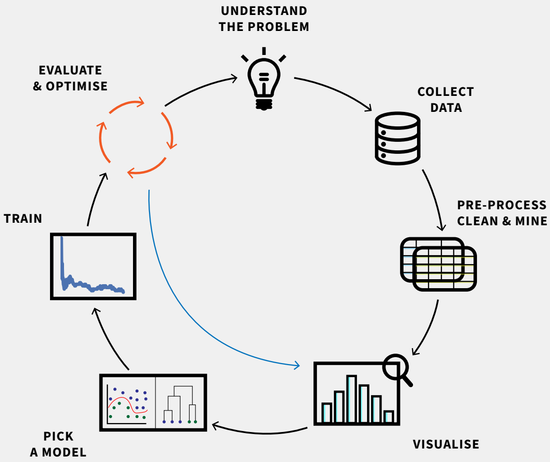
Organisation

- work in pairs on the notebooks
- if you get stuck, ask us and look at the solutions (but try first!)
- if you're ahead, look things up, ask for pointers

Data Science: bridging fields



Data analysis wheel



Jupyter

Starting Jupyter

In the Terminal or Anaconda Prompt:

```
$> jupyter notebook
```

(demo)

⚠ Get the initial path right…

Jupyter is awesome : doc and tab completion

Most important tools:

- Execute a cell: `SHIFT+ENTER`
- Get documentation about `command`: `?command`
- `TAB` completion

(demo)

Other useful commands

Command mode is triggered by pressing `ESC`. Then:

- `A` or `B`, add new cell before or after current cell
- `M` or `Y`, interpret current cell as a markdown cell or as a code cell (default)
- `D` $\times$2, delete current cell

(demo)

Pandas

Pandas , the “excel” of Python

- huge userbase, someone else has asked your question before

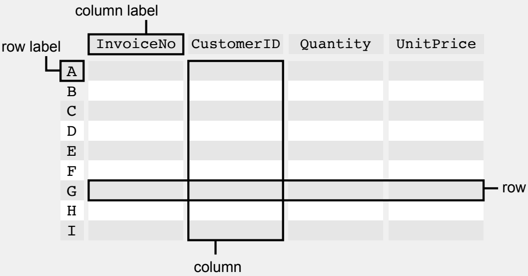
Pandas , the “excel” of Python

- huge userbase, someone else has asked your question before
- today, the basics: loading data, basic manipulations, …

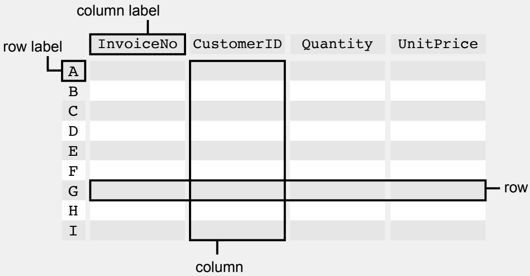
Pandas , the “excel” of Python

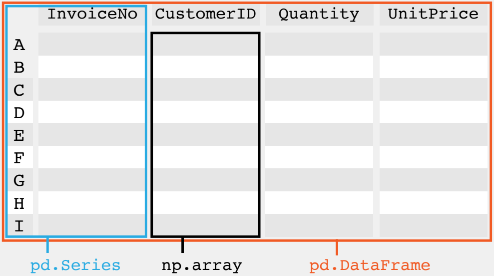
- huge userbase, someone else has asked your question before
- today, the basics: loading data, basic manipulations, …
- central class: `DataFrame`

Anatomy of a pandas dataframe



Anatomy of a pandas dataframe





Loading a data file

```python
import pandas as pd
df = pd.read_csv(fpath, ...)
df.head()
```

Many options:

- `sep`, `header`, `index_col`, ...
- remember to use `?`, Pandas documentation is excellent

Loading a data file

```python
import pandas as pd
df = pd.read_csv(fpath, ...)
df.head()
```

Many options:

- `sep`, `header`, `index_col`, ...
- remember to use `?`, Pandas documentation is excellent

Writing a `DataFrame` is easy too

```python
df.to_csv(fpath)
```

Accessing elements in a `DataFrame`

Accessing one of the column by name:

```
series = df[colname]    # returns a pd.Series object
```

Accessing elements in a `DataFrame`

Accessing one of the column by name:

```
series = df[colname]     # returns a pd.Series object
```

Accessing a specific row of a `pd.Series`, similar to `numpy`:

```
val = series[10]         # returns a value of type dtype
```

Accessing elements in a `DataFrame`

Accessing one of the column by name:

```
series = df[colname]     # returns a pd.Series object
```

Accessing a specific row of a `pd.Series`, similar to `numpy`:

```
val = series[10]         # returns a value of type dtype
```

Accessing elements using row/column names `loc()` :

```
val = df.loc[10, [c for c in df.columns if 'le' in c]]
```

`pandas.Series`: `np.array` with a plus

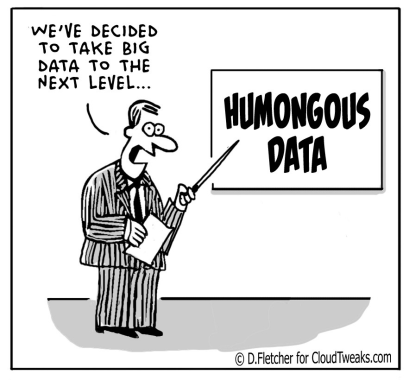
- "Wrapped around" a numpy vector
- Useful methods attached: `unique()`, `max()`, `median()`, ...
- Useful attributes: `hasnans`, `index`, `shape`, ...

Hands-on session

>>> Accessing elements in a `DataFrame`

Feature Engineering

Garbage in, garbage out, . . .



"Humongous data" is no substitute for good pre-processing and feature engineering .

Feature engineering is essential

Coming up with features is difficult, time consuming, requires ***expert knowledge****. "Applied machine learning" is basically feature engineering*

– Andrew Ng

Data comes in many shapes and sizes

Features can be

- continuous
- categorical (may or may not be ordinal)

Data comes in many shapes and sizes

Features can be

- continuous
- categorical (may or may not be ordinal)

Encoding categorical data: one-hot-encoding

	Gender
0	Male
1	Female
2	Not Specified
3	Not Specified
4	Female

	Female	Male	Not Specified
0	0	1	0
1	1	0	0
2	0	0	1
3	0	0	1
4	1	0	0

A few more useful tricks with Pandas

- `pd.describe()` summary statistics
- `drop` to remove row(s) or column(s)
- `groupby`, `apply`, (next slide)

`groupby` followed by `apply`

```
df.groupby('CustomerID')
```

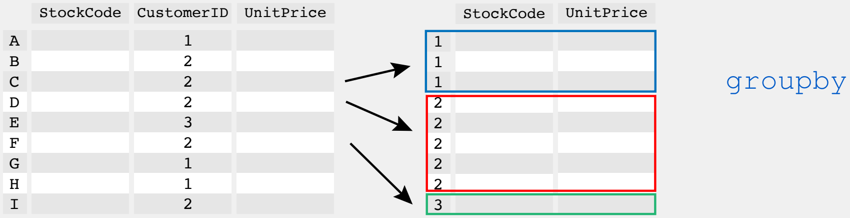
	StockCode	CustomerID	UnitPrice
A		1	
B		2	
C		2	
D		2	
E		3	
F		2	
G		1	
H		1	
I		2	

	StockCode	UnitPrice
1		
1		
1		
2		
2		
2		
2		
2		
3		

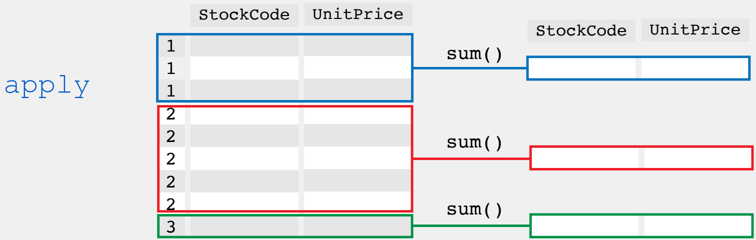
groupby

`groupby` followed by `apply`

`df.groupby('CustomerID')`



`df.groupby('CustomerID').apply(np.sum)`



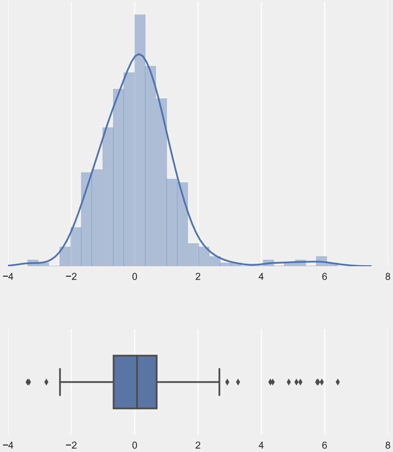
Hands-on session

\>>> Load the retail data set, explore the features.

Preprocessing the data: the important steps

- Selection and Encoding : get rid of nonsensical variables, encode ordinal variables (e.g.: OHE)
- Outliers and Imputation : what to do with rows with extreme values? and with missing values?
- Scaling : weigh features equally (*if your model is sensitive to scaling*)

Dealing with outliers , no silver bullet



- get rid of them?
- focus on them?
- how to decide which are *really* “outliers”?

Dealing with missing values, no silver bullet

What are the pros/cons of the following approaches?

- remove observations (rows) with missing values

Dealing with missing values, no silver bullet

What are the pros/cons of the following approaches?

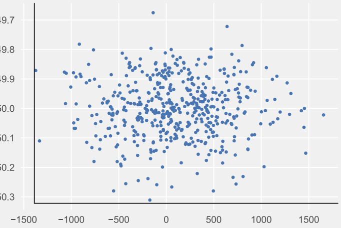
- remove observations (rows) with missing values
- remove features (columns) with missing values

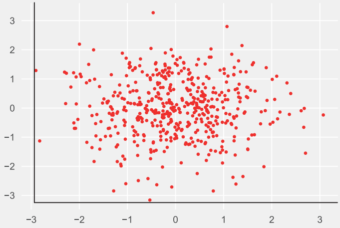
Dealing with missing values, no silver bullet

What are the pros/cons of the following approaches?

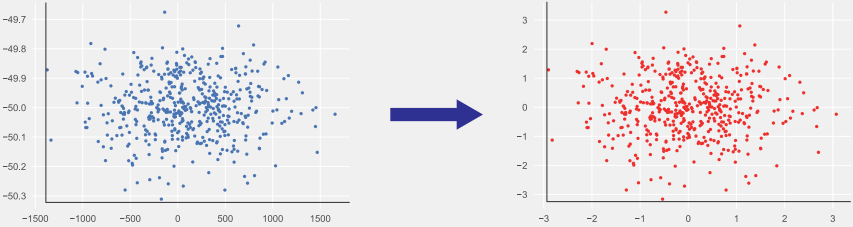
- remove observations (rows) with missing values
- remove features (columns) with missing values
- replace missing values with
  - column mean, median or mode
  - something else

Scaling : putting features on a comparable scale



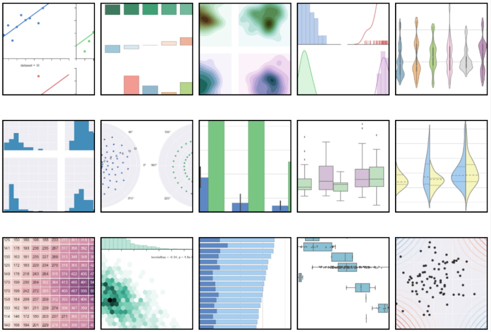


Scaling : putting features on a comparable scale



Typically, we use the standard scaler but you could also use the `MinMax` scaler (all values on [0, 1]).

Visualising data



- What data do I want to plot?
- What type of plot is suitable?
- How to convey a message with a plot?

Hands-on session

>>> Check for missing values, impute if necessary, remove outliers, visualise and scale

Machine Learning

Unsupervised vs. Supervised learning

Unsupervised Learning

- Data points x_i in feature space with p dimensions
- Aim = visualise points or group points based on similarity

Supervised Learning

- Data points x_i **and** response y_i (usually a single value)
- Eg: a transaction and whether it's fraud or not
- Aim = model how $x_i \mapsto y_i$

Building a supervised model

You can represent the data you have as coming from a hidden function f ("nature")

$$\mathrm{x} \quad \rightarrow \quad f(x) = y$$

Building a supervised model

You can represent the data you have as coming from a hidden function f ("nature")

$$\mathrm{x} \quad \rightarrow \quad f(x) = y$$

The aim is to build a model $\widehat{f}$ which approximates f:

$$\mathrm{x} \quad \rightarrow \quad \widehat{f}(x) \approx f(x)$$

Training and testing a model

Workflow:

TRAINING Consider some of the data (= training data):
build a model $\widehat{f}$ which works well on it

TESTING Consider the **rest** of the data (= test data):
check how $\widehat{f}$ is doing on that

Training and testing a model

Workflow:

TRAINING Consider some of the data (= training data):
build a model $\widehat{f}$ which works well on it

TESTING Consider the **rest** of the data (= test data):
check how $\widehat{f}$ is doing on that

⚠ The test data **must be distinct** from the training data in order to assess how the model generalises

Training and testing: the cases

- good on training, poor on testing $\rightarrow$ **overfitting**
- poor on training, poor on testing $\rightarrow$ **underfitting**
- good on training, good on testing $\rightarrow$ good sign

more about this later, now how do we build a model!?

Classification vs Regression

Classification (what you will learn):

- Training data : $(\texttt{point}, \texttt{class})_i$
- **ex**. *Fraud detection*: $(\texttt{email}, \texttt{0/1})_i$

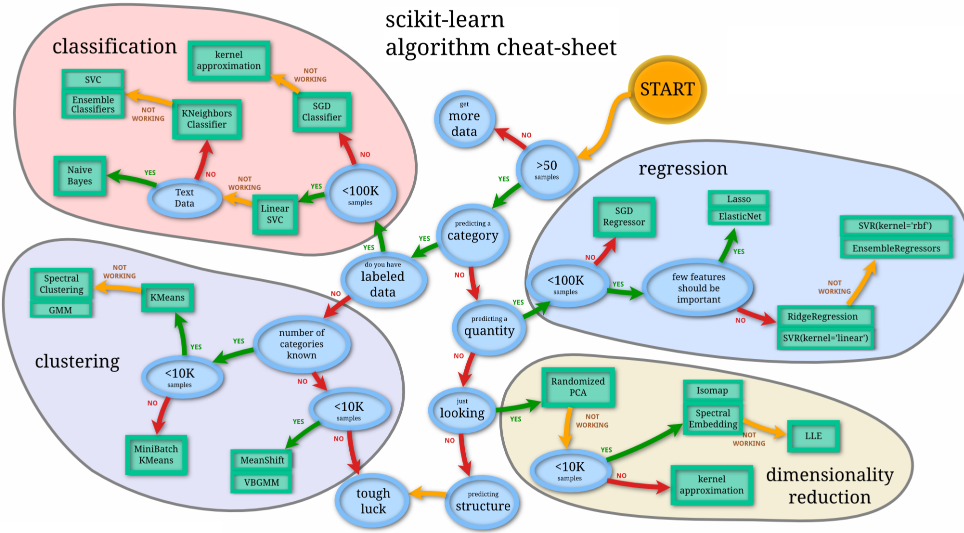
Classification vs Regression

Classification (what you will learn):

- Training data : $(\texttt{point}, \texttt{class})_i$
- **ex**. *Fraud detection*: $(\texttt{email}, \texttt{0/1})_i$

Regression:

- Training data : $(\texttt{point}, \texttt{value})_i$
- **ex**. *Salary prediction*: $(\texttt{characteristics}, \texttt{salary})_i$


scikit-learn
algorithm cheat-sheet
START
classification
kernel approximation
SVC
Ensemble Classifiers
NOT WORKING
KNeighbors Classifier
SGD Classifier
NO
Naive Bayes
YES
Text Data
Linear SVC
<100K samples
get more data
>50 samples
predicting a category
do you have labeled data
regression
SGD Regressor
Lasso
ElasticNet
SVR(kernel='rbf')
EnsembleRegressors
few features should be important
RidgeRegression
SVR(kernel='linear')
predicting a quantity
clustering
Spectral Clustering
GMM
KMeans
number of categories known
<10K samples
MiniBatch KMeans
MeanShift
VBGMM
tough luck
predicting structure
just looking
dimensionality reduction
Randomized PCA
Isomap
Spectral Embedding
LLE
kernel approximation

Testing a binary classifier

On the test data, compare the predicted responses $\hat{y}_i = \widehat{f}(x_i)$ to the actual response y_i:

		PREDICTED	
		Positive	Negative
ACTUAL	Positive	TP	FN
	Negative	FP	TN

- jargon from clinical trials, positive/negative defined by context
- this is the confusion matrix

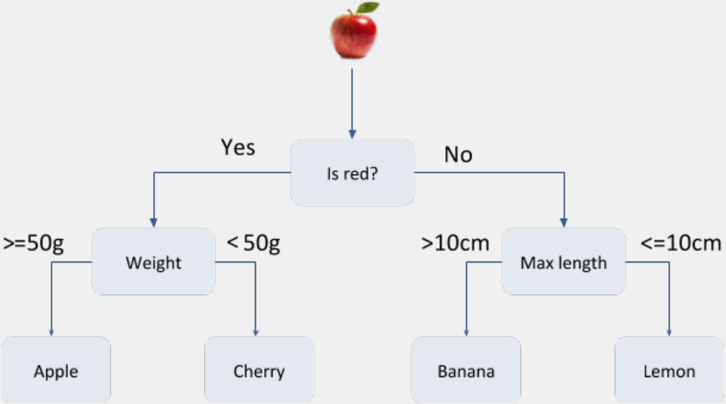
Testing a binary classifier 2

		PREDICTED	
		Positive	Negative
ACTUAL	Positive	TP	FN
	Negative	FP	TN

- Accuracy : $\frac{TP+TN}{N}$, sensitivity/recall : $\frac{TP}{TP+FN}, \ldots$
- metric of importance usually dictated by *context*. In fraud:
  - \+ flagging a clean transaction as fraudulent (ok)
  - \- not flagging a fraudulent transaction (bad)

Decision Trees

Decision trees



Decision Tree Classifier (DTC)

A DTC is a model with a hierarchical structure. Each node corresponds to a feature and a split. To fit a DTC:

Decision Tree Classifier (DTC)

A DTC is a model with a hierarchical structure . Each node corresponds to a feature and a split. To fit a DTC:

- at each level, consider all features and for each, the splitting point that best separates the two classes

Decision Tree Classifier (DTC)

A DTC is a model with a hierarchical structure . Each node corresponds to a feature and a split. To fit a DTC:

- at each level, consider all features and for each, the splitting point that best separates the two classes
- take the feature that offers the best separation and continue

Decision Tree Classifier (DTC)

A DTC is a model with a hierarchical structure . Each node corresponds to a feature and a split. To fit a DTC:

- at each level, consider all features and for each, the splitting point that best separates the two classes
- take the feature that offers the best separation and continue

Different metrics can be used to measure “separation”. Common ones are GINI impurity and information gain (broadly speaking they lead to very similar results).

Decision Tree Classifier (DTC)

A DTC is a model with a hierarchical structure . Each node corresponds to a feature and a split. To fit a DTC:

- at each level, consider all features and for each, the splitting point that best separates the two classes
- take the feature that offers the best separation and continue

Different metrics can be used to measure “separation”. Common ones are GINI impurity and information gain (broadly speaking they lead to very similar results).

Once a DTC is adjusted, classification is trivial: just follow the tree!

When to stop splitting?

- Pick a maximum depth
- Otherwise, get one sample per "leaf node" $\rightarrow$ overfitting (one decision path for every single sample)

Hands-on session

>>> Decision trees

Decision trees, recap

+ Very easy model to interpret
+ No need to normalise data and it handles ordinal/boolean data
- Propensity to overfit

Ensemble models

Getting the intuition

Say you have a set of symptoms and you go see two doctors…

- Doctor A has 60% chance of providing an accurate diagnostic
- Doctor B has 75% chance of providing an accurate diagnostic

Both say you have X. *How confident are you about the diagnostic and why?*

Getting the intuition

Say you have a set of symptoms and you go see two doctors...

- Doctor A has 60% chance of providing an accurate diagnostic
- Doctor B has 75% chance of providing an accurate diagnostic

Both say you have X. *How confident are you about the diagnostic and why?*

How about if you added 5 other doctors to the mix?

Key elements of ensembles

For ensembles to work, the following properties must be met:

- weak learner : any single expert need to be more right than wrong on average ($>$ 50% accurate)

Key elements of ensembles

For ensembles to work, the following properties must be met:

- weak learner : any single expert need to be more right than wrong on average ($>50\%$ accurate)
- diversity : experts need to make different errors

Key elements of ensembles

For ensembles to work, the following properties must be met:

- weak learner : any single expert need to be more right than wrong on average ($>$ 50% accurate)
- diversity : experts need to make different errors

Multiple ways of *aggregating*, simplest: majority voting .

How about doing that with decision trees?

You know that fitting a DTC is easy. How can you come up with a lot of diverse trees?

How about doing that with decision trees?

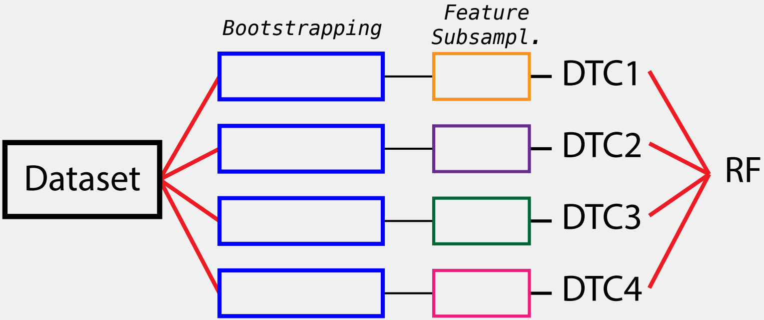
You know that fitting a DTC is easy. How can you come up with a lot of diverse trees?

1. create statistically similar datasets, train a DTC on each

How about doing that with decision trees?

You know that fitting a DTC is easy. How can you come up with a lot of diverse trees?

1. create statistically similar datasets, train a DTC on each
2. only consider a random subset of the features when training


Bootstrapping
Feature
Subsampl.
Dataset
DTC1
DTC2
DTC3
DTC4
RF

Creating statistically similar datasets

- a sample drawn uniformly at random from the dataset will have similar statistical properties (mean, variance, …)

Creating statistically similar datasets

- a sample drawn uniformly at random from the dataset will have similar statistical properties (mean, variance, ...)
- do that multiple times to get bootstrapped datasets

Creating statistically similar datasets

- a sample drawn uniformly at random from the dataset will have similar statistical properties (mean, variance, ...)
- do that multiple times to get bootstrapped datasets
- datasets will contain multiple times some samples and zero times some others $\rightarrow$ diversity

Creating statistically similar datasets

- a sample drawn uniformly at random from the dataset will have similar statistical properties (mean, variance, ...)
- do that multiple times to get bootstrapped datasets
- datasets will contain multiple times some samples and zero times some others $\rightarrow$ diversity
- this can be done in parallel

Feature subsampling

- If there is a dominant feature, it will likely be selected every time as root on most bootstrapped datasets $\rightarrow$ correlation .

Feature subsampling

- If there is a dominant feature, it will likely be selected every time as root on most bootstrapped datasets $\rightarrow$ correlation .
- Randomly selecting a smaller set of features helps avoiding this $\rightarrow$ diversity while reducing correlation.

Ensemble models recap

- aggregate output of multiple weak learners that are diverse will increase the accuracy

Ensemble models recap

- aggregate output of multiple weak learners that are diverse will increase the accuracy
- random forest = many decision trees. Diversity by bootstrapping the data and random subsampling of the features.

The Competition

Tips for the competition

- groups of 2 or 3
- start simple, make it work
- tune it, try more things, repeat
- you can ask us for *technical help* but not for *advice*

Tips for the competition

- groups of 2 or 3
- start simple, make it work
- tune it, try more things, repeat
- you can ask us for *technical help* but not for *advice*

Make sure to **test** your model (`tester.py`) before submission.