Competitive data analysis. Review lecture

Alexander Guschin, Skolkovo, Spring 2021

1. 🐤 Introduction to competitions: what, how and why
2. 💻 From simple solutions to complex ones
3. 👩🏼‍💻 Organizing your work in teams and solo

Introduction to competitions: what, how and why

Example: is this word a surname?

X (object)	y (target)
human	0
sandwich	0
ivanov	1
student	?
petrov	?

InClass Prediction Competition

DMIA sport intro: Surnames classification

Определите слова, являющиеся фамилиями

115 teams · 3 years ago

Overview Data Notebooks Discussion Leaderboard Rules

Join Competition

Public Leaderboard Private Leaderboard

The private leaderboard is calculated with approximately 75% of the test data.

This competition has completed. This leaderboard reflects the final standings.

Refresh

#	△pub	Team Name	Notebook	Team Members	Score	Entries	Last
1	—	**Кирилл Тушин**			0.98728	25	3y
2	—	**Ольга Филиппова**			0.97510	9	3y
3	—	**Соломенник Михаил**			0.97416	29	3y

Visual Doom Competition @ CIG 2018

Competitions may be divided (1)

- By submit type: csv file or docker image
- By “seriousness”: Kaggle (points, medals, prizes) or Kaggle Inclass
- By data collection: tables, texts, images, etc

Competitions may be divided (2)

- By given amount of time: competition (~week or more) or hackathon (usually less than a week)
- By assesment: clear metrics or jury’s assessment
- By level of certainty: open or closed leaderboard

Competitions may be divided (3)

- By metrics' "stability": from random to LocalCV=PublicLB=PrivateLB

Submits

Given: x_train, y_train, (x_test)
Find: y_test

- We submit csv file: x_test can be analyzed
- We submit docker-image: x_test cannot be analyzed

The diversity of data is enormous:

- tables (tsv, csv)
- texts (txt, json)
- images (jpg, png)
- audiofiles (wav, mp3)
- graphs
- medical data
- time series
- bytecodes
- ... etc

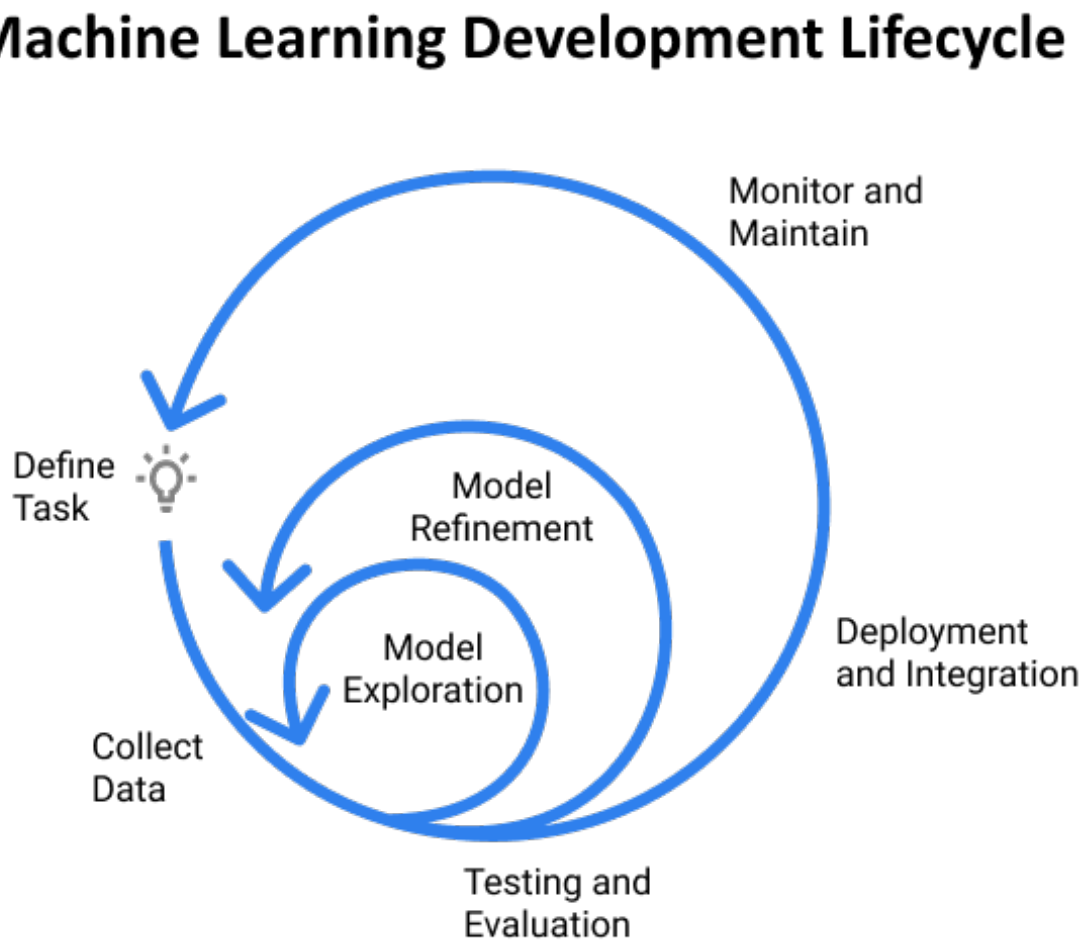
Leaderboard

LB score = **f(y_true, y_pred)**

Example: is this word a surname?

X (object)	**y (target)**	**part**
human	0	train
sandwich	0	train
ivanov	1	train
student	?	public test
petrov	?	private test

Difference between industrial and competitive tasks



https://www.jeremyjordan.me/ml-projects-guide>

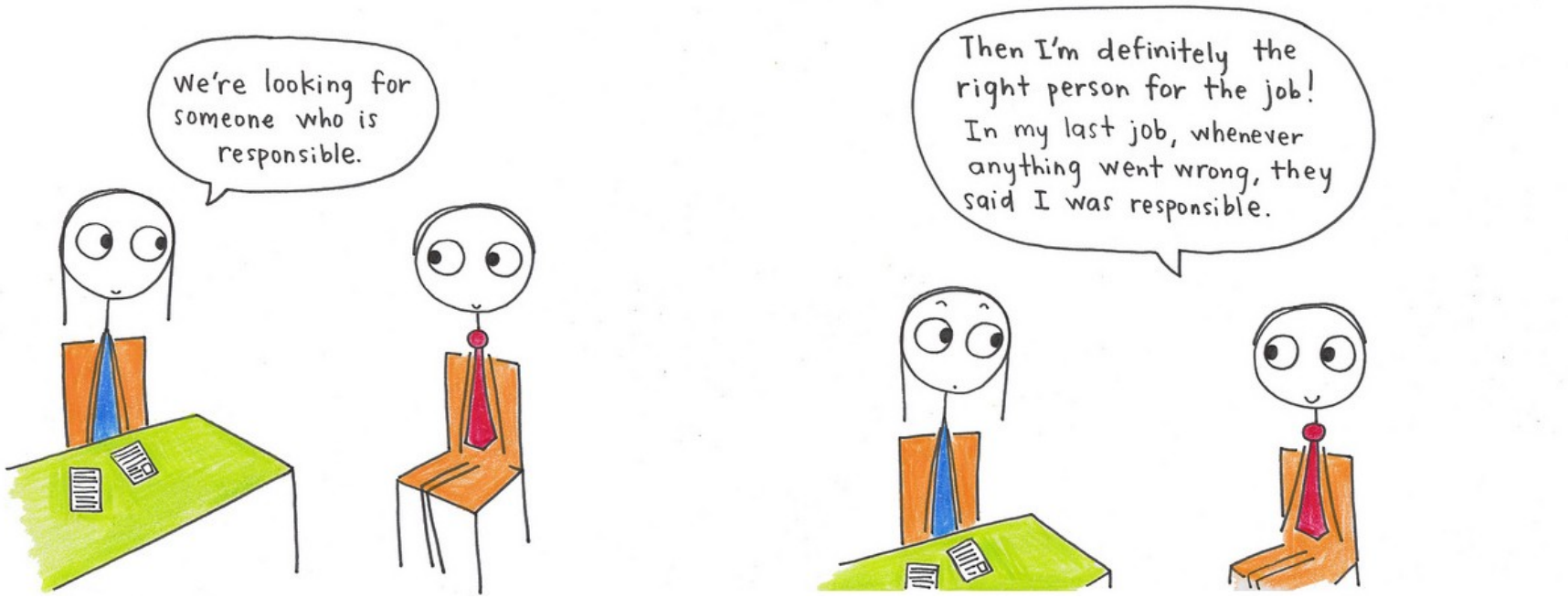
Difference between industrial and competitive tasks

Aspect	**Real life**	**Competition**
Formalization	Y	N
Metric selection	Y	N
Production	Y	N
Computational complexity	Y	N/Y
Data collection	Y	N/Y
Model complexity	Y	N/Y
Metric value	Y	Y

Metric value is the only thing to think about!

Why do organizers need competitions?

- Solving business problems
- PR
- Hiring



Why do participants need competitions?

- Education
- Networking
- Career
- Money (rare occasion!)

Why does society need competitions?

- Solving problem
- Objective comparison between different ML algorithms realizations
- Data
- Ecosystem
- Raising interest in DS

🥇 Prizes

- Platforms
  - Money
  - Swag
  - Medals and points
- Other places
  - Meet-up speeches
  - Blog stories

Typical competition's rules (1)

- 🐎 Deadlines
  - You cannot submit your solution after given deadline (obviously)
  - There is a deadline for the first submit
  - There is a deadline for team assembly
- 💬 Share advice/code/data
  - You can do it in teams
  - You can do it publicly on the competition page
  - You cannot do it privately

Typical competition's rules (2)

- 🕵️ Data
  - External data may be banned
  - ...or partially allowed, but you will need to share this data via competition forum in advance
- 💁‍♀️ Teams
  - You cannot remove team member from your team

Various platforms organize various competitions

- kaggle.com
- drivendata.org
- boosters.pro
- aicrowd.com
- etc

There are agreggated lists of current competitions, e.g.
https://ods.ai/competitions

How to solve competitions

From simple to complex

1. Examine the task and metric
2. Submit constant baseline
3. Write your framework
4. Generate features
5. Tune models
6. Unite solutions

These are the main topics of `How to Win a DS Competition` course (available on Coursera)

Examine the task and metric

1. 👓 Read everything carefully
2. 🔦 Examine data
3. 🔍 Look for unusual patterns and leaks

It is convenient to write all ideas into a list ✏️

Exploratory data analysis

1. Understand your task better
2. Find patterns
3. Find outliers
4. Find leaks and bugs
5. Create new hypotheses and features
6. Understand which models are preferable

Study the subject area - it is interesting and useful in solving other problems.

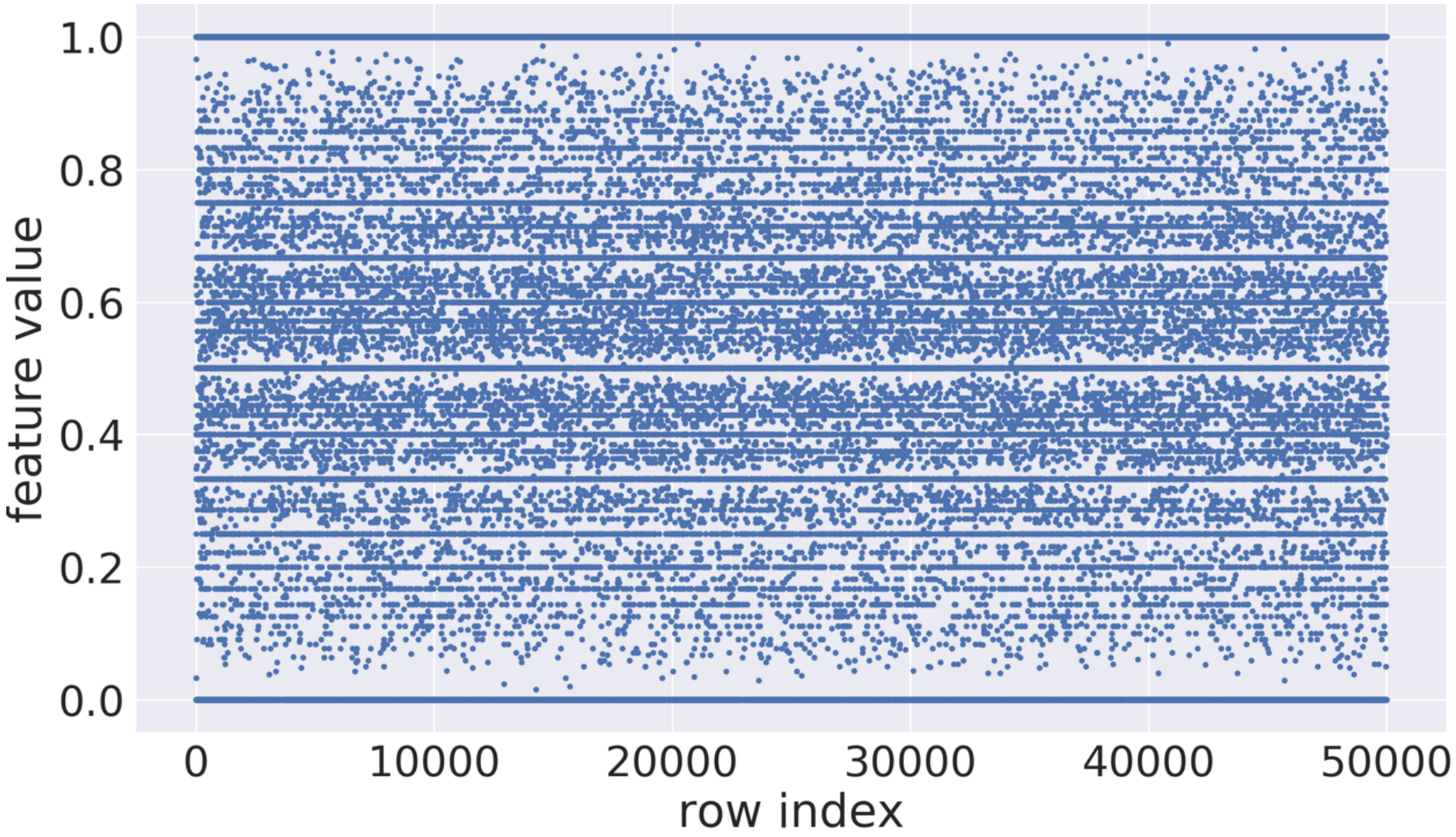
Several ways to analyze given data

1. Features
    1. Distinct features
    2. Group of features
2. Objects
    1. Distinct objects
    2. Group of objects
3. Files
    1. Content
    2. Metadata

Keep an eye on the target

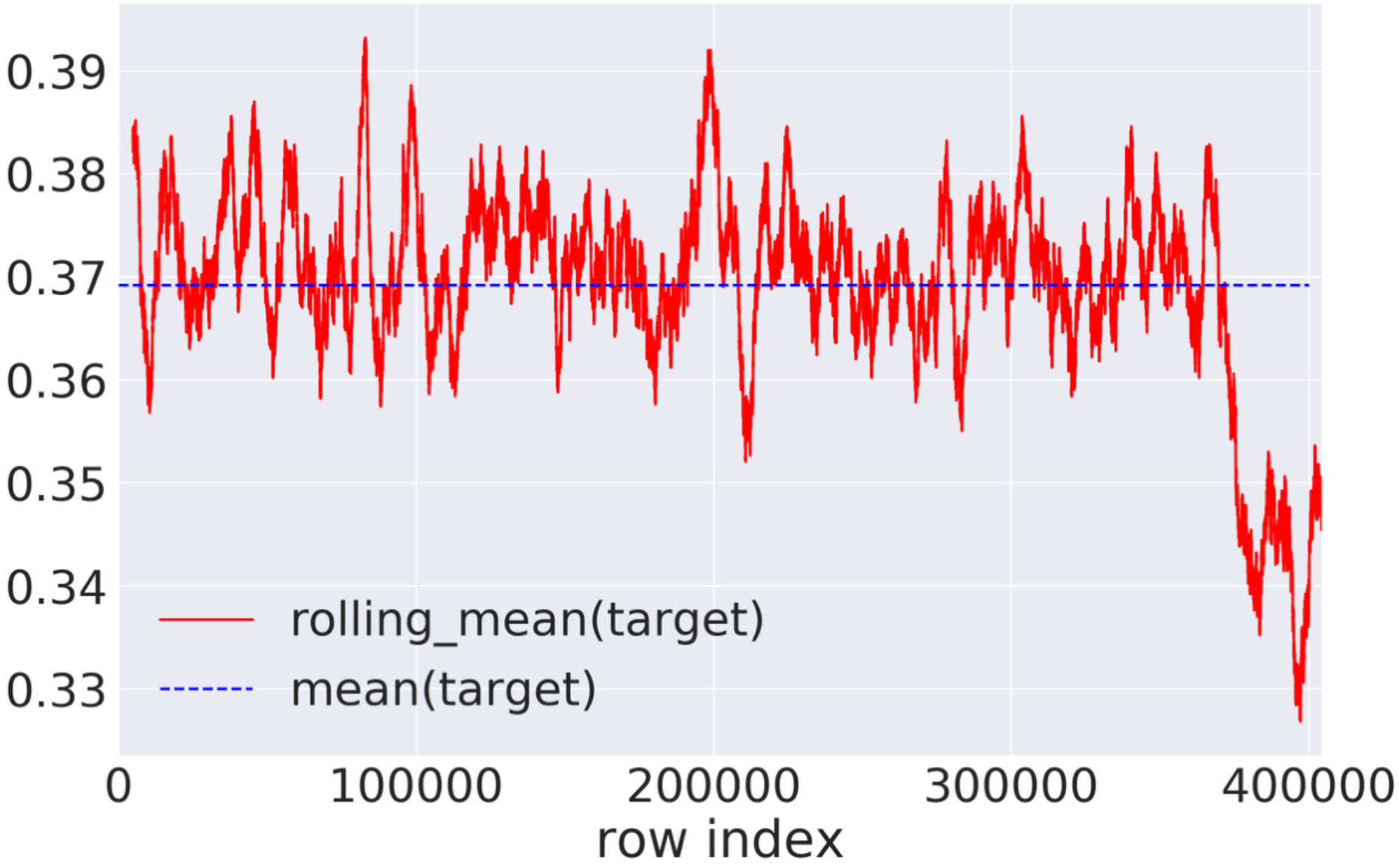
Analyze one feature.
Why?

- reverse anonymization, feature scaling
- add more external data
- check features' value to find potential problems


1.0
0.8
0.6
0.4
0.2
0.0
feature value
0
10000
20000
30000
40000
50000
row index

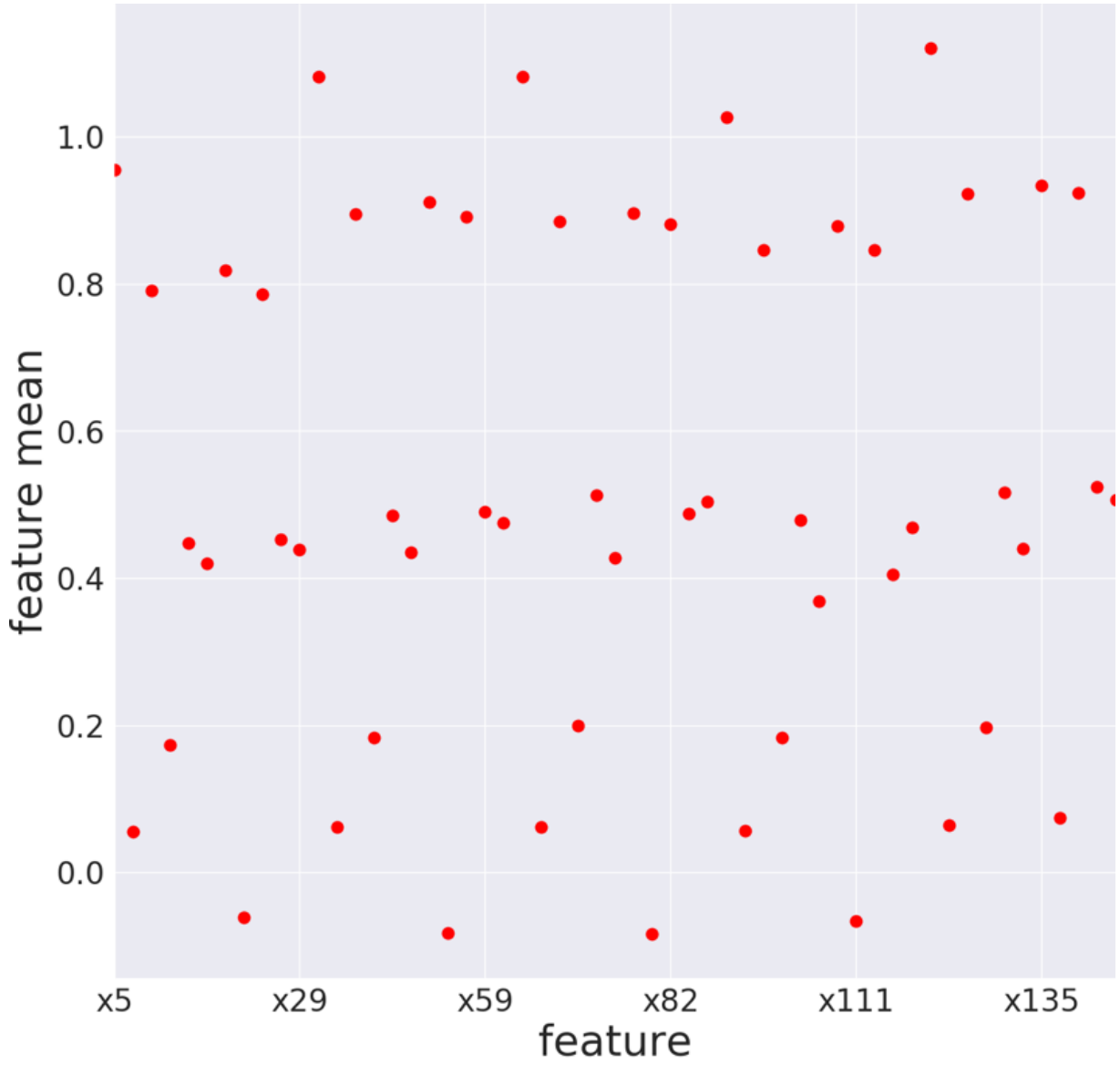
Check features' value to find potential problems:

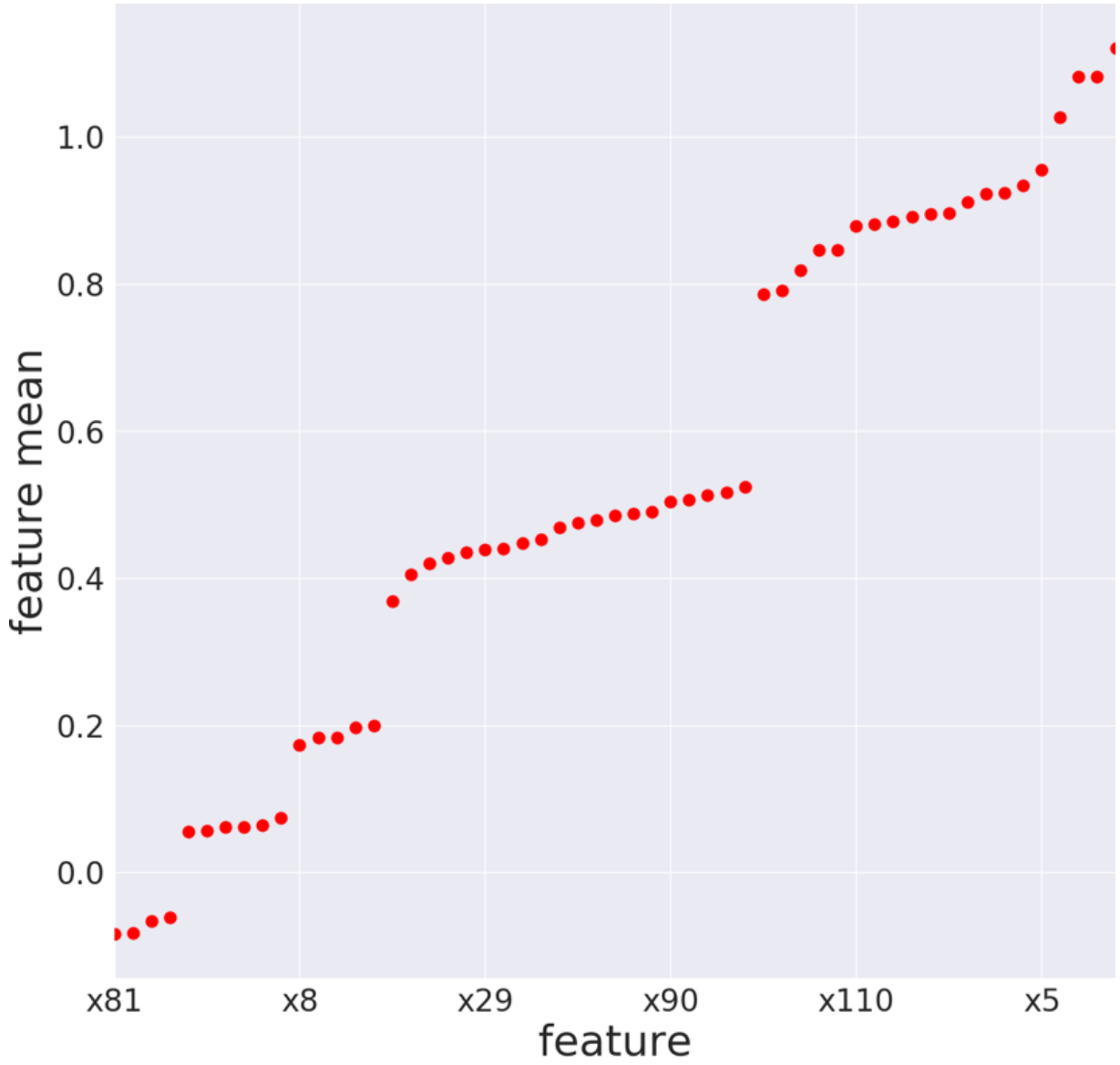
- Unrealistic features' values
- Examine if the feature changes scale / distribution over time


0.39
0.38
0.37
0.36
0.35
0.34
0.33
rolling_mean(target)
mean(target)
0
100000
200000
300000
400000
row index

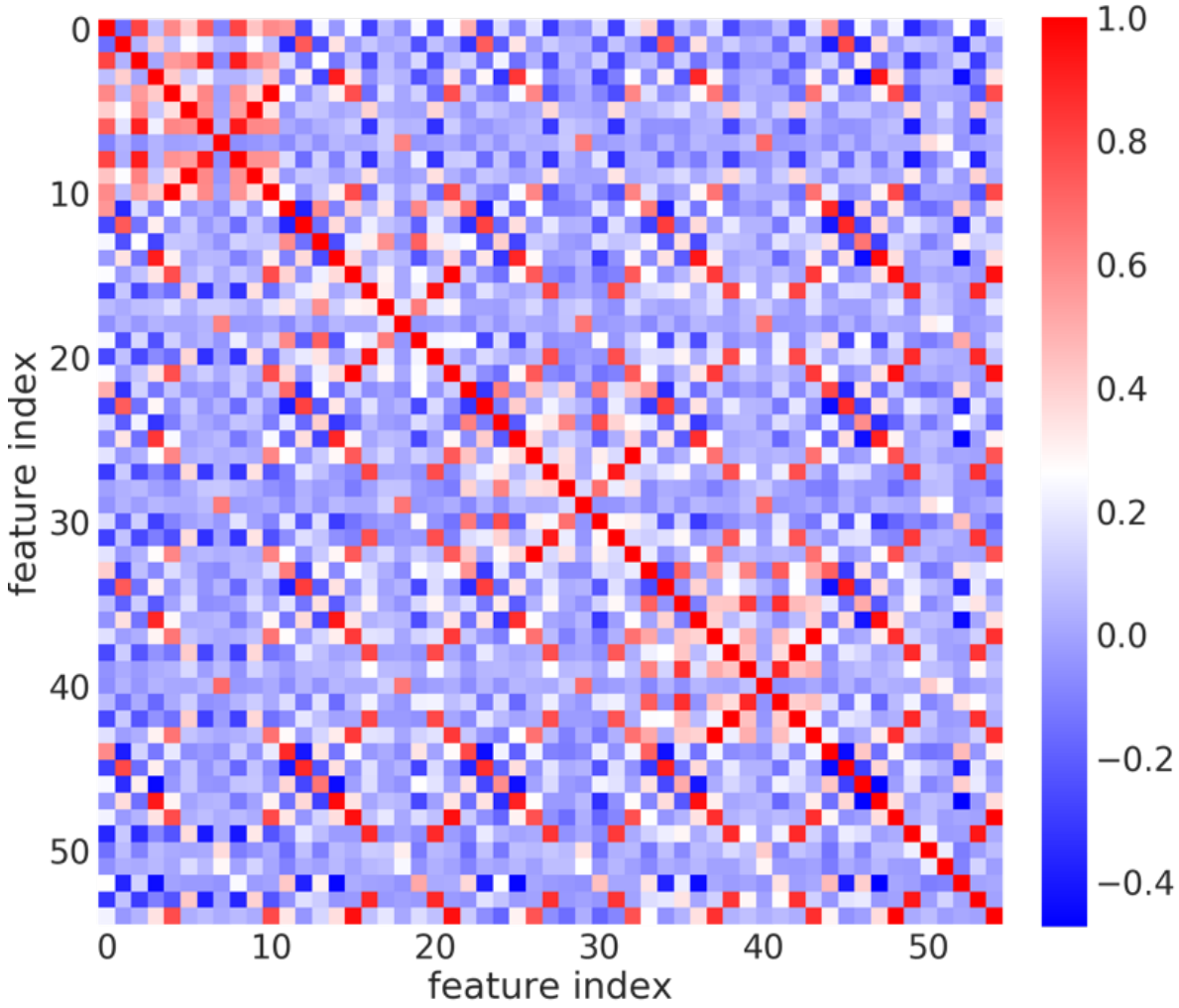
Analyze numerous features
Why?

- Group features
- Analyze possible interactions


feature mean
1.0
0.8
0.6
0.4
0.2
0.0
x5
x29
x59
x82
x111
x135
feature


feature mean
1.0
0.8
0.6
0.4
0.2
0.0
x81
x8
x29
x90
x110
x5
feature

Correlation matrix for features



Analyzing the metric / loss function

- 🍰 Popular metrics - you should know
  1. Optimal constants for metrics
  2. What algorithms optimize this metric?
  3. Features of metrics
- 🌵 Strange metrics - you should try
  1. Answer the questions above theoretically and experimentally
  2. Reduce to or approximate with a popular metric

Classification

- Accuracy
- Logloss
- ROC AUC
- F-score

How to find optimal constant?

Regression

- MSE / RMSE
- MAE
- MAPE

What algorithms optimize these loss functions?
How to make the algorithm to optimize MAPE?

Strange metrics, examples:

- Pearson correlation coefficient
- MAE for integer y_true
- Cohen’s kappa
- ...

Constant baseline

Where to find a starting score for comparison

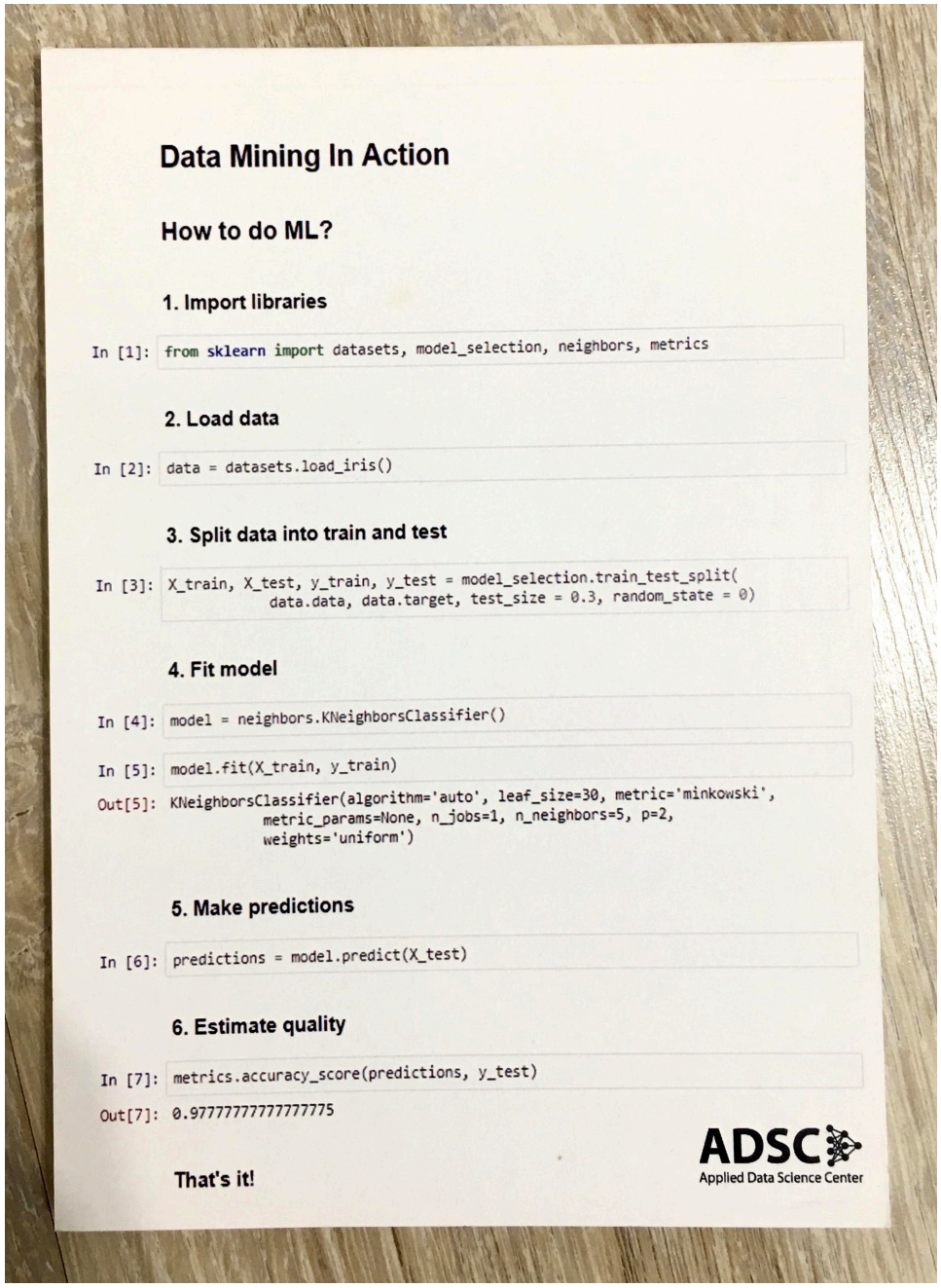
1. Leaderboard
2. Train your model and submit
3. Calculate constant and submit

Why?

1. To evaluate improvement
2. To check Train / Public for offset
3. To estimate how many people from LB is better than a constant
4. There are competitions at which constant gets to the top ...

Developing a framework

- Framework - a notebook / script that allows you to make a submission
- It can and should be improved during the entire competition.
- Its purpose is to facilitate and speed up this process, and preferably - to make it reproducible.



Data Mining In Action

How to do ML?

1. Import libraries

```
In [1]: from sklearn import datasets, model_selection, neighbors, metrics
```

2. Load data

```
In [2]: data = datasets.load_iris()
```

3. Split data into train and test

```
In [3]: X_train, X_test, y_train, y_test = model_selection.train_test_split(
                    data.data, data.target, test_size = 0.3, random_state = 0)
```

4. Fit model

```
In [4]: model = neighbors.KNeighborsClassifier()
```

```
In [5]: model.fit(X_train, y_train)
```

```
Out[5]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=1, n_neighbors=5, p=2,
                    weights='uniform')
```

5. Make predictions

```
In [6]: predictions = model.predict(X_test)
```

6. Estimate quality

```
In [7]: metrics.accuracy_score(predictions, y_test)
```

```
Out[7]: 0.9777777777777775
```

That's it!

ADSC
Applied Data Science Center

Typical scheme

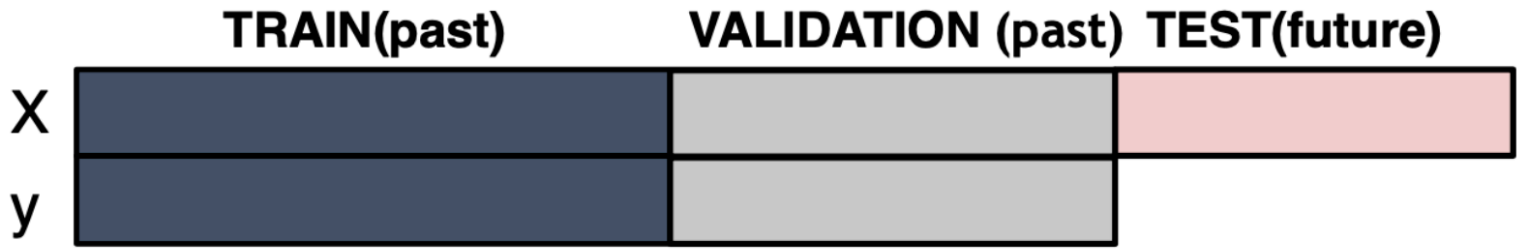
№	Step	💾, because you want...
1	Data loading and preprocessing	to not to wait
2	Splitting into train / test	to reproduce
3	Feature generation	to not to wait
4	Model training	to mix predictions
5	Mixing models	to submit

Splitting into training / validation

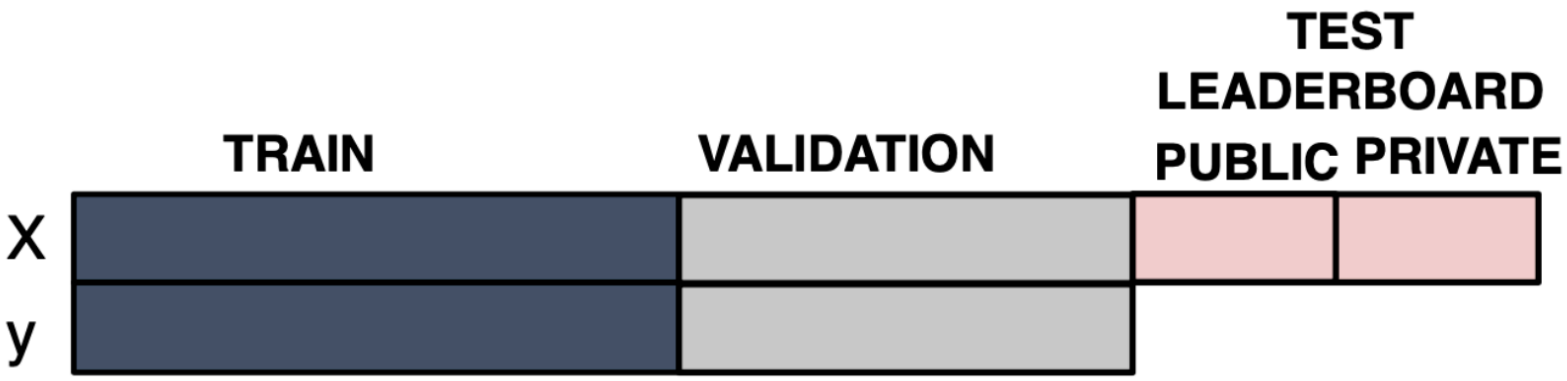
1. Holdout / KFold - in terms of computation speed, stability of metrics
2. Reproduction of the train / test split

What's the difference from DS at work?

In real life



In competitions



Frequent splits

- Random - i.i.d
- By time - there is a time dependence
- By id - there are several samples for one id
- Combinations, e.g., by time and id

Examples of splitting by id

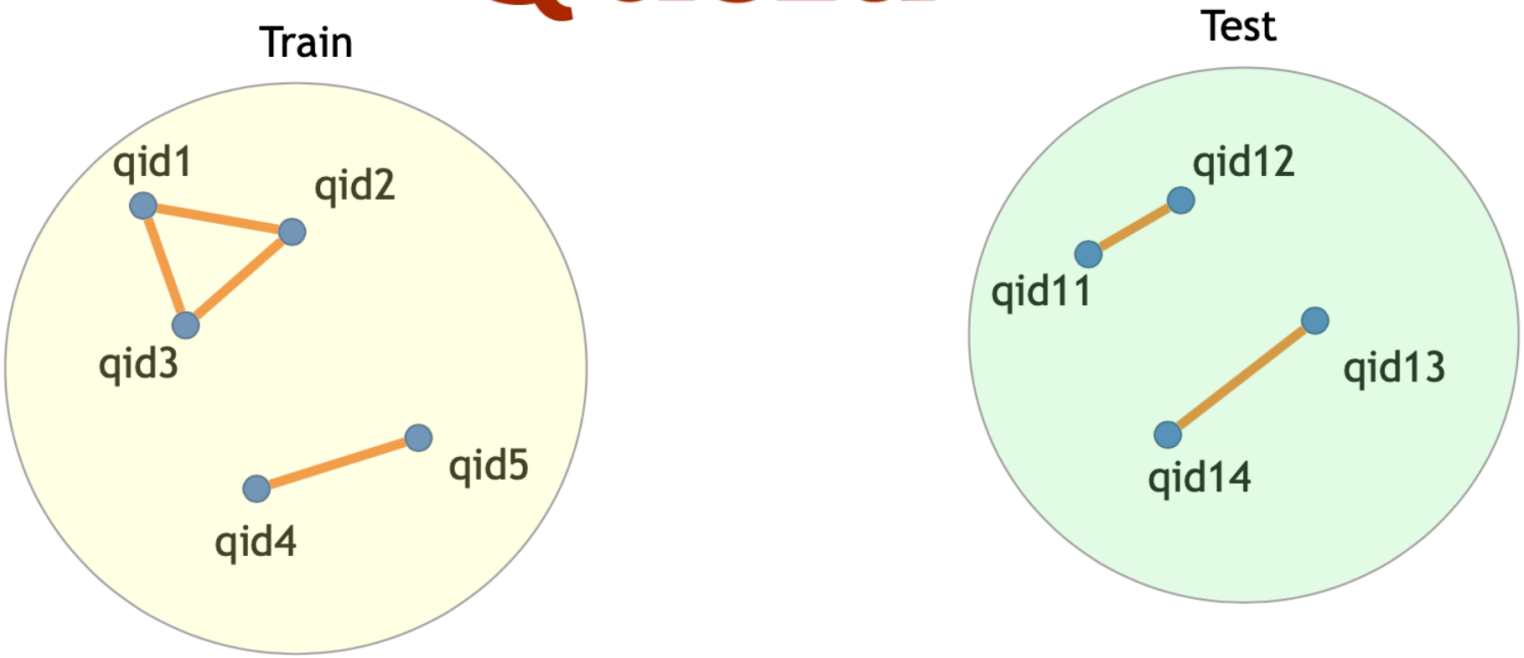
- Client id
- City id
- Advertising company id
- Car travel id

An example of complex validation in Quora Question Pairs

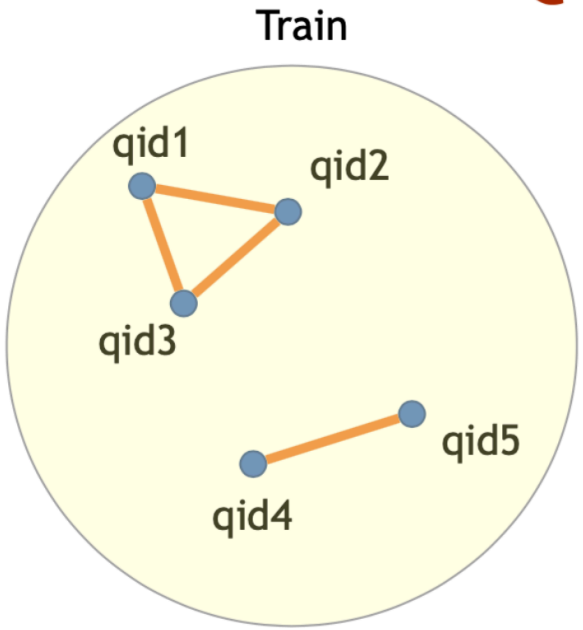
Quora

	id	qid1	qid2	question1	question2	is_duplicate
0	0	1	2	What is the step by step guide to invest in share market in india?	What is the step by step guide to invest in share market?	0
1	1	3	4	What is the story of Kohinoor (Koh-i-Noor) Diamond?	What would happen if the Indian government stole the Kohinoor (Koh-i-Noor) diamond back?	0
2	2	5	6	How can I increase the speed of my internet connection while using a VPN?	How can Internet speed be increased by hacking through DNS?	0
3	3	7	8	Why am I mentally very lonely? How can I solve it?	Find the remainder when [math]23^{24}[/math] is divided by 24,23?	0
4	4	9	10	Which one dissolve in water quikly sugar, salt, methane and carbon di oxide?	Which fish would survive in salt water?	0
5	5	11	12	Astrology: I am a Capricorn Sun Cap moon and cap rising...what does that say about me?	I'm a triple Capricorn (Sun, Moon and ascendant in Capricorn) What does this say about me?	1

Quora



Quora



Разделение 1
Fold1:
qid1 - qid2
qid1 - qid3
...

Fold2:
qid2 - qid3
qid4 - qid5
...

Разделение 2
Fold1:
qid1 - qid2
qid1 - qid3
qid2 - qid3
...

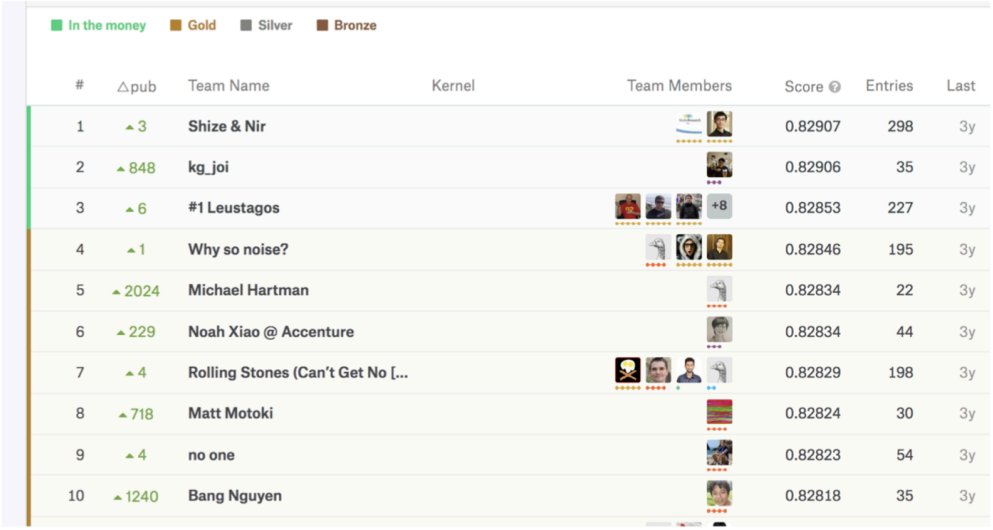
Fold2:
qid4 - qid5
...

Что лучше?

Reasons for the discrepancy between validation and leaderboard

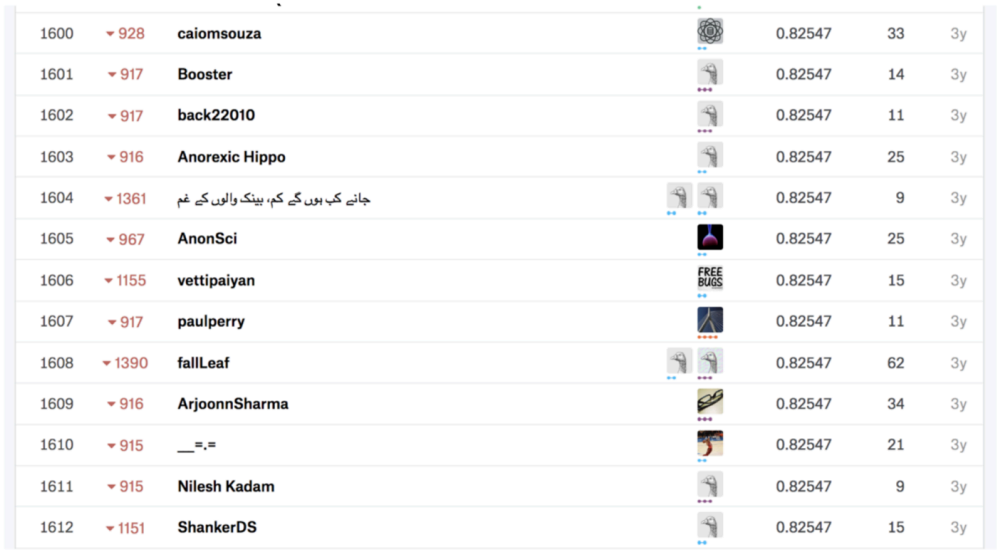
- Splitting a train into training / validation differs from splitting into a train / test
- Data distributions in validation and test are different
- Lack of data in (Public) LB

Leaderboard shuffle



In the money Gold Silver Bronze

#	△pub	Team Name	Kernel	Team Members	Score	Entries	Last
1	▲3	Shize & Nir			0.82907	298	3y
2	▲848	kg_joi			0.82906	35	3y
3	▲6	#1 Leustagos		+8	0.82853	227	3y
4	▲1	Why so noise?			0.82846	195	3y
5	▲2024	Michael Hartman			0.82834	22	3y
6	▲229	Noah Xiao @ Accenture			0.82834	44	3y
7	▲4	Rolling Stones (Can't Get No [...			0.82829	198	3y
8	▲718	Matt Motoki			0.82824	30	3y
9	▲4	no one			0.82823	54	3y
10	▲1240	Bang Nguyen			0.82818	35	3y



1600	▼928	caiomsouza			0.82547	33	3y
1601	▼917	Booster			0.82547	14	3y
1602	▼917	back22010			0.82547	11	3y
1603	▼916	Anorexic Hippo			0.82547	25	3y
1604	▼1361	جانے کب ہوں گے کم، بینک والوں کے غم			0.82547	9	3y
1605	▼967	AnonSci			0.82547	25	3y
1606	▼1155	vettipaiyan			0.82547	15	3y
1607	▼917	paulperry			0.82547	11	3y
1608	▼1390	fallLeaf			0.82547	62	3y
1609	▼916	ArjoonnSharma			0.82547	34	3y
1610	▼915	__=.=			0.82547	21	3y
1611	▼915	Nilesh Kadam			0.82547	9	3y
1612	▼1151	ShankerDS			0.82547	15	3y

Leaderboard shuffle

- Data in Public or Private is not sufficient
- Data in Public and Private is different
- The forecast is not too different from the random
- The metric is not resilient to outliers

Feature generation

Целевая переменная

Числовая переменная

Переменные-счетчики

Категориальные переменные

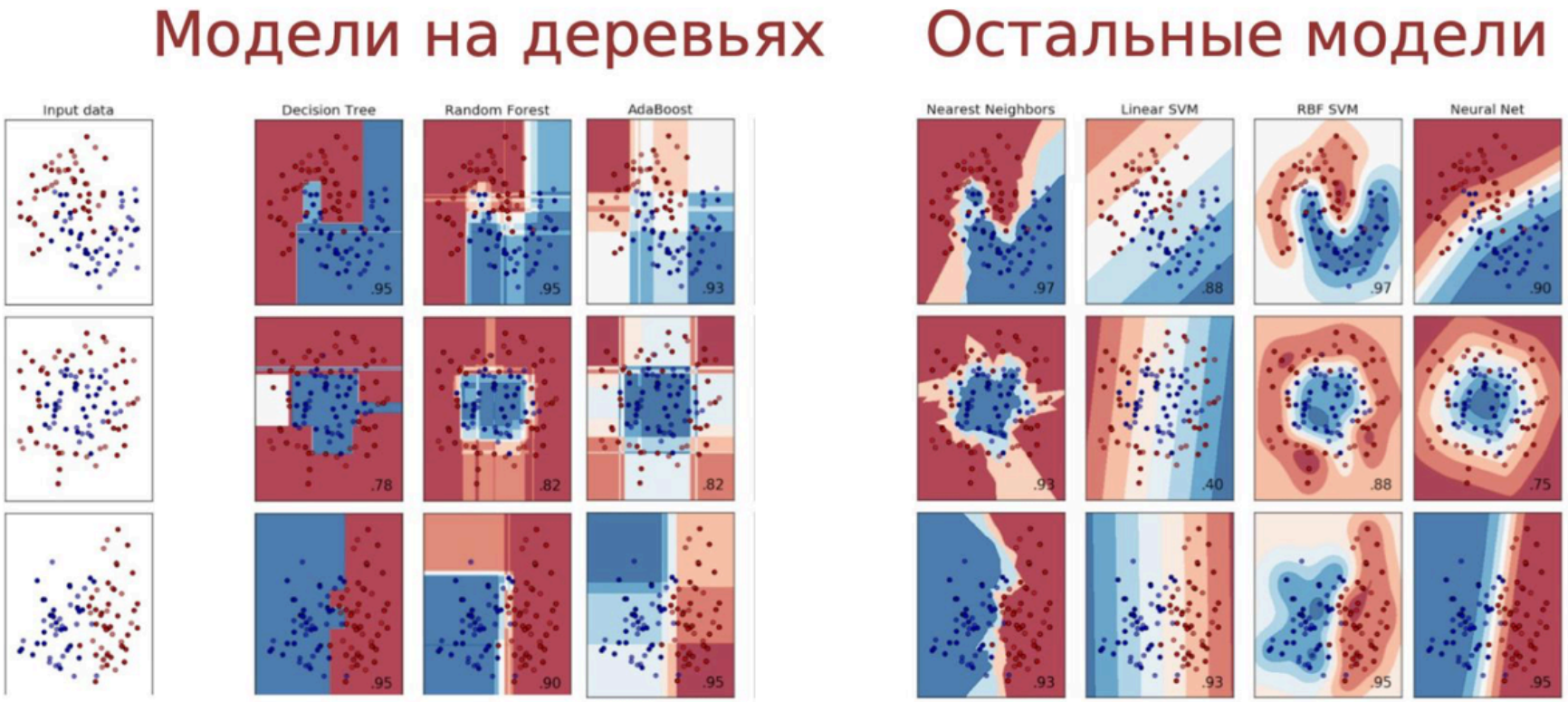
id

Текстовая переменная

	PassengerId	Survived	Pclass	Name
0	1	0	3	Braund, Mr. Owen Harris
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...
2	3	1	3	Heikkinen, Miss. Laina
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)
4	5	0	3	Allen, Mr. William Henry
5	6	0	3	Moran, Mr. James
6	7	0	1	McCarthy, Mr. Timothy J
7	8	0	3	Palsson, Master. Gosta Leonard

	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	male	22.000000	1	0	A/5 21171	7.2500	NaN	S
1	female	38.000000	1	0	PC 17599	71.2833	C85	C
2	female	26.000000	0	0	STON/O2. 3101282	7.9250	NaN	S
3	female	35.000000	1	0	113803	53.1000	C123	S
4	male	35.000000	0	0	373450	8.0500	NaN	S
5	male	29.699118	0	0	330877	8.4583	NaN	Q
6	male	54.000000	0	0	17463	51.8625	E46	S
7	male	2.000000	3	1	349909	21.0750	NaN	S

Dependence on the utilized models



Numerical features: scaling

- MinMax, RobustScaler, StandardScaler...
- Rank:
  rank([-100, 0, 1e5]) == [0,1,2]
  rank([1000,1,10]) = [2,0,1]
- Logarithm:
  np.log(1 + x)
- Raising to a power<1:
  np.sqrt(x + 2/3)

Numerical features: outliers
In features 😺 and target 🥶

- Winsorizing (when outlier has really large value)
- Feature generation (when outliers are related to some events that may be important)
- True values recovery (when it is possible)
- Removing objects
- Rank
  rank([-100, 0, 1e5]) == [0, 1, 2]
  rank([1000, 1, 10]) = [2, 0, 1]

Categorical and ordinal features

- 💻 Typical processing methods
  - Leave as is
  - LabelEncoding
  - One-hot-encoding
  - Frequency coding
  - Numeric feature encoding
  - Target variable encoding
- 🔎 What to pay attention to
  - New categories
  - Number of categories in a feature
  - Small categories

Temporal features

- Periodicity
  Day number, month, season, year, second, minute, hour
- How much time has passed / left
  - From one moment (for all data)
  - The moment depends on the object selection (how many days are left until the weekend)
- The difference between points in time
- Usage for generating features with a sliding window

Spatial structure

- Neighboring (“interesting”) objects
- Aggregates and statistics
- Additional data

Selection and tuning of models

Four-step instruction

1. Understand how algorithms work
2. Choose the most suitable algorithms
3. Select the most important hyperparameters
4. Tuning - manually or automatically

Linear models

😺	🙈
Easy to interpret	Requires data preparation
Simple to implement	Not the best result
“What feature should be changed to make this?”	Post-processing (negative sales)
Works fast	

KNN

😺	🙈
Ensemble model	Sklearn implementation is slow
Coordinates, dimensions, distances	Requires data preparation
Embedding-based training	Not the best result
	Long and expensive for production

Gradient boosting

😺	🙈
Usually the best for both competition and production	Easy to overfit, mess with leaks and incorrect validation
Relatively fast speed	Can’t use linear dependencies of features

Neural networks

😺	🙈
Indispensable for some tasks	Can be unstable on small datasets
Many interesting uses	The winner is the one with the most GPU
Lots of available architectures and saved weights	Hard to tune and to choose architecture
Sometimes can be beautifully interpreted	Often “black box”

Coding by average

- Problem: Categorical features with a large number of unique values are difficult for ML algorithms
- Solution: Let's apply some transformation of the feature that will make the new values sorted by
  - 👍 another feature (for example, when predicting the cost of square meter, replace the feature "city" with the attribute "average salary")
  - ☢️ target (coding by average)

Apartment ID	Cost of square meter	City	Average salary	Average cost of square meter
1	200000	Moscow	80000	180000
2	80000	Volgograd	30000	75000
3	110000	Kazan	40000	100000
4	100000	Ufa	35000	90000
5	60000	Bereznyaki	25000	75000
6	80000	Novosibirsk	30000	60000

How to encode

- Regression
  - Average or median of target
- Classification
  - Target frequency
  - #P - #N
  - LN(#P - #N)

The main problem is overfitting

- Calculating the values of a new feature using K-Fold inside a train
- Smoothing - closer to mean `y_train.mean()` for rare categories
- Expanding mean - we use only objects from `1` to `i-1` to calculate for object `i`

Ensemble

Three people vote independently, the probability of a correct answer is 0.7

Result	Formula	Probability
Three are right	0.7 * 0.7 * 0.7	0.3429
Two are right	0.7 * 0.7 * 0.3 * 3	0.4409
One is right	0.7 * 0.3 * 0.3 * 3	0.189
No one is right	0.3 * 0.3 * 0.3	0.027

Probability to answer correctly - 0.7838

Ensemble

- 🥖 Ingredients
  - Data as diverse as possible
  - Models as diverse as possible
- 🌮 Recipes
  - Averaging
  - Linear combinations
  - Mixture of experts
  - Boosting
  - Stacking

Sources of diversity in data

- 🍀 Preprocessing
  - Data decomposition by characteristics and objects
  - Feature transformation: `log, sqrt, TFiDF, PCA, NMF, tSNE, encoder-decoder, w2v, fasttext, BERT, mean encoding, OHE`
  - Combinations of groups and transformations
- 📎 Sampling
  - By objects (bootstrap, undersampling, oversampling)
  - By features (RSM)

Sources of diversity in data

- 😺 😺 😻 Algorithms of the same family with different parameters
  - Tree depth
  - Number of neighbors
  - Architecture of a neural network
  - Regularization
- 😺 🐴 🐵 Different families of algorithms
  - Linear models
  - Naive Bayes
  - Factorizing machines
  - Trees
  - Unsupervised learning
  - Neural networks (different topology)

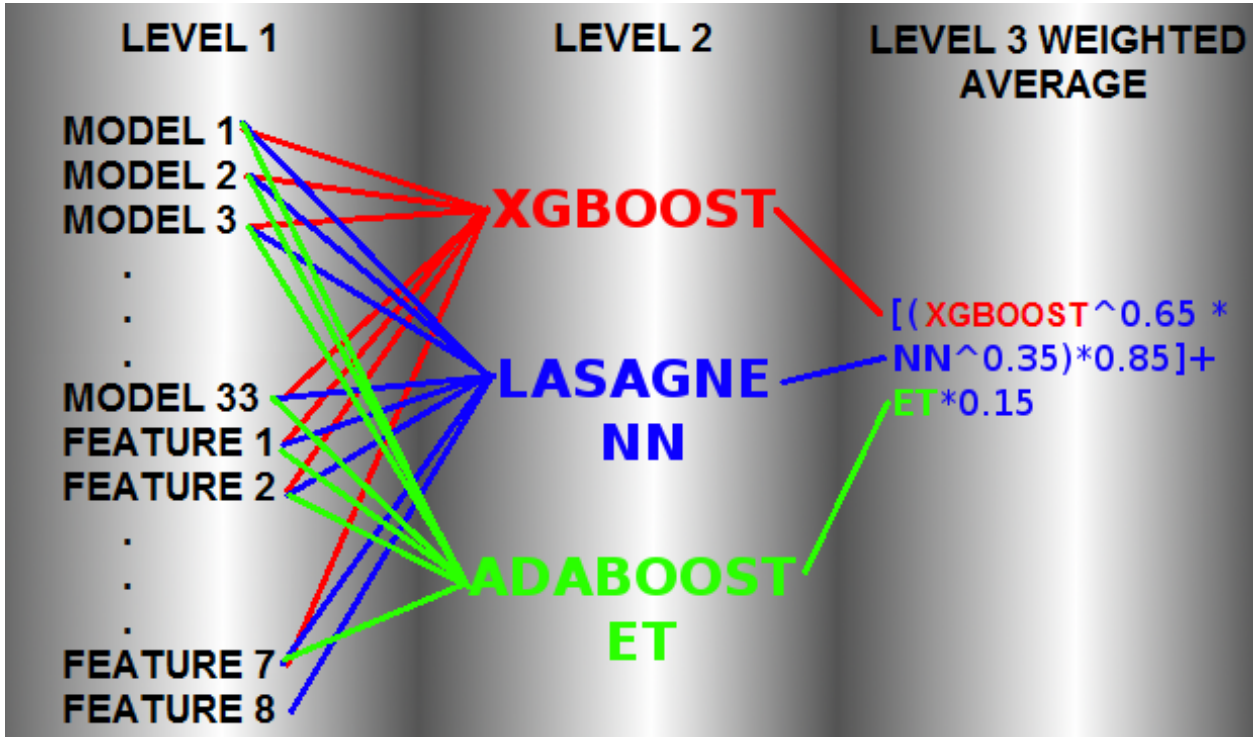
Bootstrap, bagging and voting: examples

- Standard method
  - DecisionTree + sampling + averaging ~= RandomForest
- kaggle-avito-prohibited-content
  - kNN + linear models = solution from top-5
- kaggle-malware
  - xgboost + oversample(a=7) + averaging
- kaggle-tube-pricing
  - NN + dropout\dropconnect + averaging
- any competition
  - xgboost + sampling(subsample, colsample_bytree) + averaging

Bootstrap, bagging and voting

- 😺 Advantages
  - Easy to implement
  - Efficient
  - Reduces spread
  - Prevents overfitting
  - Mix coefficients (weight) can be selected according to the leaderboard
- 🙈 Disadvantages
  - Pretty simple mapping class

Stacking


LEVEL 1
LEVEL 2
LEVEL 3 WEIGHTED AVERAGE
MODEL 1
MODEL 2
MODEL 3
.
.
.
MODEL 33
FEATURE 1
FEATURE 2
.
.
.
FEATURE 7
FEATURE 8
XGBOOST
LASAGNE NN
ADABOOST ET
[(XGBOOST^0.65 *
NN^0.35)*0.85]+
ET*0.15

Stacking

- 🤔 Idea: let ML algorithms create an ensemble
- 😺 Advantages
  - Nonlinear mappings (RF / GBM / NN)
  - One of the most effective ways
- 🙈 Disadvantages
  - Most time consuming
  - Easy to overfit

Stacking

- 🗑️ You cannot “afford” simple mistakes
  - training and validation should not overlap at every moment
- 🗻 If there is a lot of data, there would be no problems
  - there is not a lot of data, so we use the technique of a similar cross-validation
- ⌛ Honest hold-out is required to test ideas
  - at least simple 40 / 30 / 30 partionining

How to solve a competition - a plan (again)

1. Examine Task - EDA
2. Analyze a metric - what to do with popular and strange metrics
3. Make a constant baseline
4. Write a framework
5. Break into train / test
6. Generate features
7. Select and tune models
8. Encode with average
9. Ensemble

Organization of work - in teams and solo

What do you need to participate

1. Time!
2. Coding skill (desire to learn)
3. Hardware (can be rented)

Time

- You need to select your goal in advance
  - Finish in top-5
  - Finish in top-10%
  - Learn from others
  - Try new tasks, approaches, libraries
- How long does it take for DS to solve a problem in the industry or to conduct a research for an article?

Coding skill or desire to learn

- Bash for some libraries and working with a server
- Python, R, Matlab, other languages
- For Python: ipython, jupyter notebook, numpy, pandas, matplotlib, scikit-learn, lightgbm, многие другие

Hardware

- 💻
  - Convenient to work with a laptop with 16+ гб RAM, core i5+, SSD
  - You can rent a server at aws, google cloud, microsoft azure, selectel, etc
- 🚂
  - More datasets => more CPU, RAM and / or SSD required
  - More compute intensive (features and models) => CPU
  - Competitions with pictures / texts => GPU and SSD required

Empirical Tips

1. Check as early as possible that your validation is correct
2. Write out your ideas
3. Read forums
4. Study other participants' code

Solutions' Reproducibility

- Most important
    1. random_state
    2. git
    3. submission files
- Advices
    - Convenient to store the code of all submissions (git)
    - Convenient to store files with submissions (filesystem, git lfs, cloud storage ...)

Operation modes

Make two modes for your notebook / script: validation and test, for example

```
X = pd.read_csv('data/train.csv')
y = pd.read_csv('data/y_train.csv')

MODE = 'validation'
if MODE == 'validation':
    x_train, x_test, y_train, y_test = train_test_split(X, y, ...)
elif MODE == 'test':
    x_train = X
    y_train = y
    x_test = pd.read_csv('data/test.csv')
```

Random_state

1. `train_test_split(random_state=0)`
2. `RandomForestClassifier(random_state=0)`
3. `np.random.seed(0); np.random.choice(...)`

Submissions

1. The organizers may ask you to reproduce the final submission
2. You may need to reproduce some of your submissions
3. You may want to mix up the submissions you made

Regular commits - in the end of your notebook...

```
COMPETITION = 'kaggle.com/...'
```

```
FILE = 'submissions/_LGBM_NN.csv'
MESSAGE = 'LGBM-super-stacked + NN averaged 10 fold'
submit.to_csv(FILE)
```

```
%%bash
git pull
git add StackingLGBM.ipynb
git commit -m '{MESSAGE}'
git push
kaggle competitions submit \
    -c {COMPETITION} -f {FILE} -m {MESSAGE}
```

https://www.kaggle.com/docs/api

Working in team

- 😺 Advantages
  - The fastest way to learn new things is to do it from others or with others
  - More people => more chances to win
- 🙈 Disadvantages
  - Team members may stop doing something
  - Other conflicts may arise
- ℹ️ You cannot remove members from the team on kaggle

What is needed for successful teamwork

🤔

1. Plan your work
2. Share ideas
3. Share code (git)
4. Share submissions (git / cloud storage) and mix them

Usual teamwork

1. A general validation sample is formed
2. Everyone prepares his or her solutions: they process data in their own way, generate features and train models
3. Members' solutions should be regularly mixed

Planning your work and sharing ideas

1. Planning in advance is right
2. You need to sync regularly
3. Calls - for sync, chat - for discussions
4. You need to be ready to combine your solutions
5. Someone has to (be responsible for the team’s work)

Sharing code

1. Create a private repository and give access to team members
2. Commit your code there regularly

Combining solutions

1. Allocate the validation set first! (and check it several times!)
2. Write a script / notebook that will mix predictions
3. Do this regularly to understand your progress

Validation Sample Check

Day 1:

```
A: made validation, train_test_split(test_size=0.2, random_state=1)
B: perfect, go
```

Day 31:

```
A: Check again, is your test_ids [5821, 2440, 2143, ...]?
B: [1734, 2314, 4325, ...]
A: :'(
```

Common reasons:

1. Different versions of python and libraries
2. Different order of lines in the split dataframe

Mixing solutions

🤘

- Write a script / notebook that can select a way to mix your models from a validation sample.
- To avoid mixing with leaderboard, you need to store predictions (sometimes probabilities!) for validation (1) and test (2)
- At the end of the notebook is the code that makes the commit and submission.

Mixing solutions

- 🐵 Steps for complication
  1. Averaging
  2. Averaging with weights (== blending with linear model)
  3. Stacking
- 🕐 Mix regularly
  1. To understand if it works
  2. To spend time more effectively

Competitions are not only about ML. They are more about

1. Making your code work
2. Parsing data in strange formats
3. Working with a dataset that does not fit into memory
4. Finding a bug in the data that the organizers missed
5. Using algorithms you don’t understand
6. Speeding up your calculations
7. Looking through a large amount of data

But real life is also about this!