relationalAI

How to approach ML projects

Seifeddine Fezzani

December 11st, 2020

relationalAI

Agenda

01 Introduction

02 EDA and cross-validation strategy

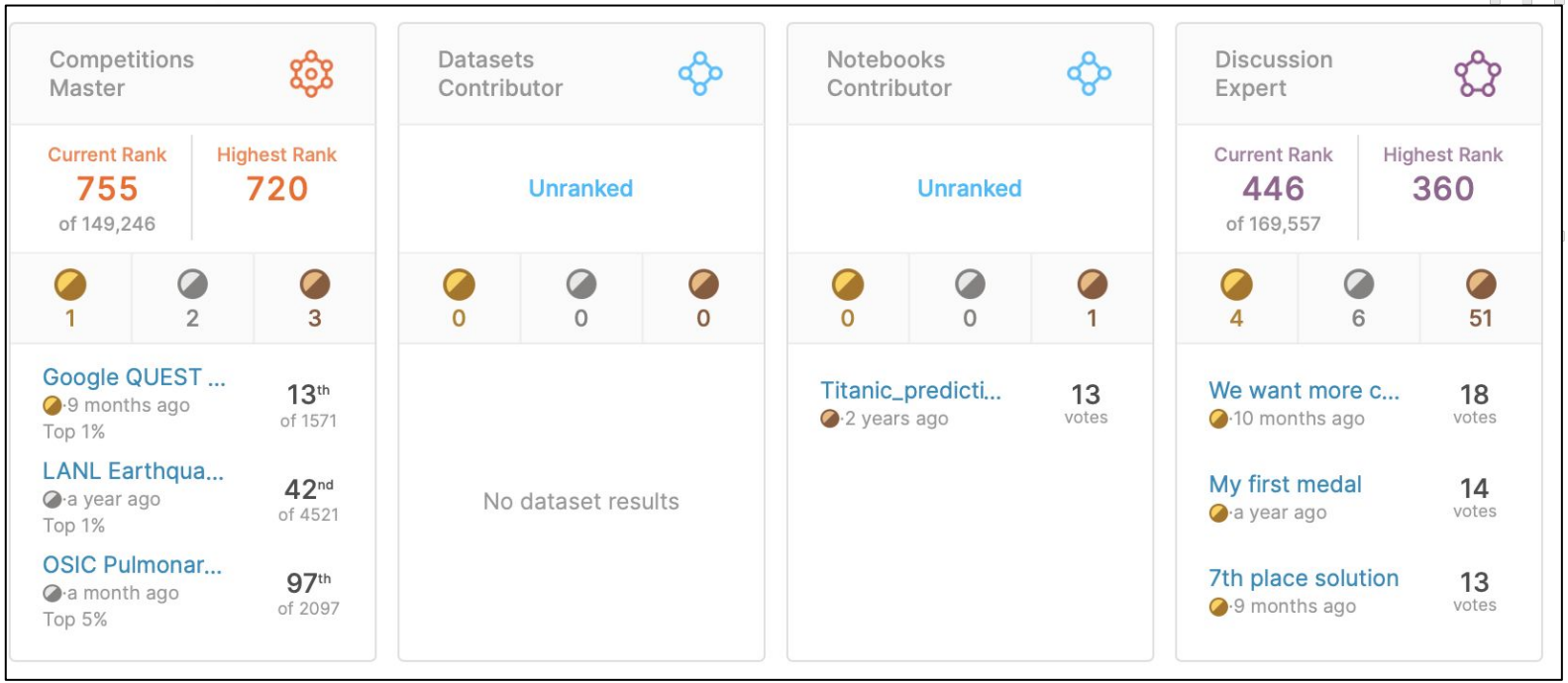
03 Improve results using data

04 Improve results using models

05 Questions & Answers

SUP'COM
Ecole Supérieure des Communications de Tunis

rAI



Kaggle Competitions

Featured Code Competition

OSIC Pulmonary Fibrosis Progression

Predict lung function decline

$55,000
Prize Money

Open Source Imaging Consortium (OSIC) · 2,097 teams · a month ago

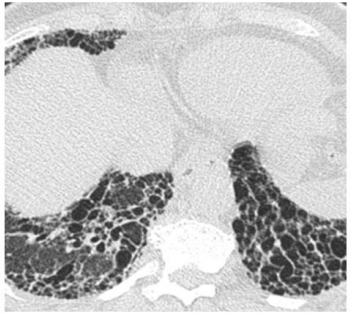
Overview | Data | Notebooks | Discussion | Leaderboard | Rules | Team | My Submissions | Late Submission

Overview

- Description
- Evaluation
- Timeline
- Prizes
- Code Requirements

Imagine one day, your breathing became consistently labored and shallow. Months later you were finally diagnosed with pulmonary fibrosis, a disorder with no known cause and no known cure, created by scarring of the lungs. If that happened to you, you would want to know your prognosis. That's where a troubling disease becomes frightening for the patient: outcomes can range from long-term stability to rapid deterioration, but doctors aren't easily able to tell where an individual may fall on that spectrum. Your help, and data science, may be able to aid in this prediction, which would dramatically help both patients and clinicians.

Current methods make fibrotic lung diseases difficult to treat, even with access to a chest CT scan. In addition, the wide range of varied prognoses create issues organizing clinical trials. Finally, patients suffer extreme anxiety—in addition to fibrosis-related symptoms—from the disease's opaque path of progression.



Open Source Imaging Consortium (OSIC) is a not-for-profit, co-operative effort between academia, industry and philanthropy. The group enables rapid advances in the fight against Idiopathic Pulmonary Fibrosis (IPF), fibrosing interstitial lung diseases (ILDs), and other respiratory diseases, including emphysematous conditions. Its mission is to bring together radiologists, clinicians and computational

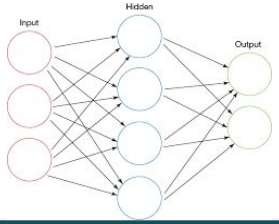
Pipeline (Kaggle)

Step 1

- Business understanding
- Exploratory data analysis
- Extract information
- Create impact
- Set up a reliable validation strategy

Step 2

- Pre-processing
- Features engineering
- Features selection



Step 3

- ML algorithms
- Hyper-parameters tuning
- Ensembling

Step 4

- Submit and see results

relationalAI

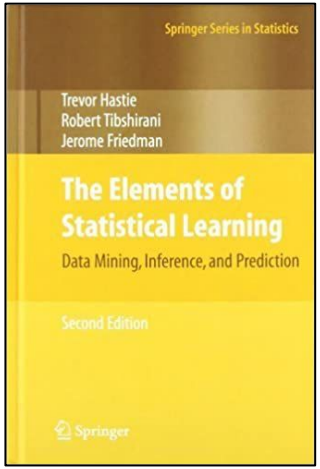
EDA and CV setting

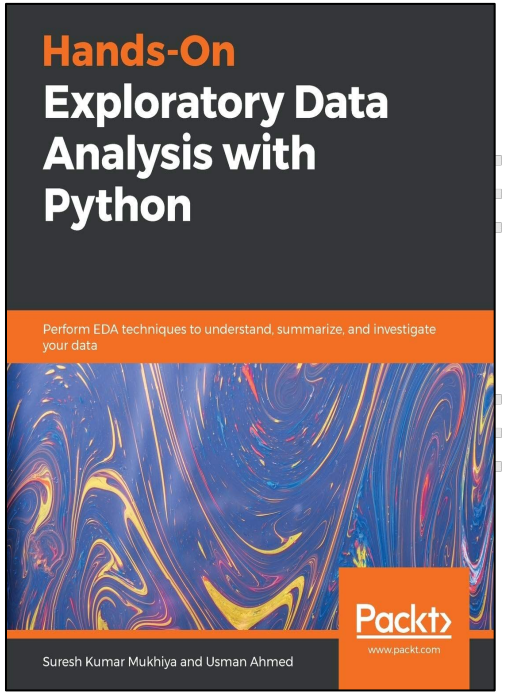
EDA

Exploratory Data Analysis

EDA: Exploratory Data Analysis

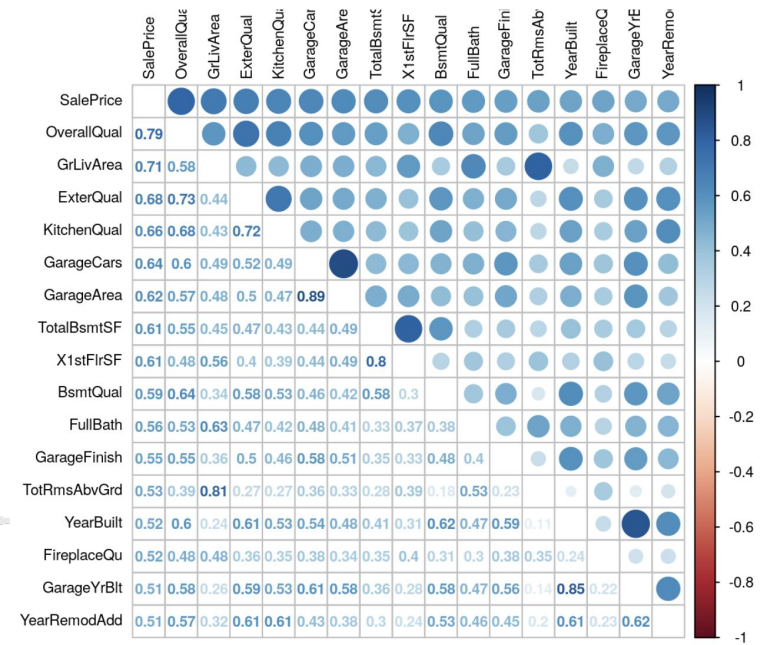
- Ask lots of questions
  - What distributions do you see ?
  - What relationships do you think might benefit the problem ?
- Focus on understanding
  - You are not creating a report
- Your EDA must have **an impact**
- Tell a story ;)

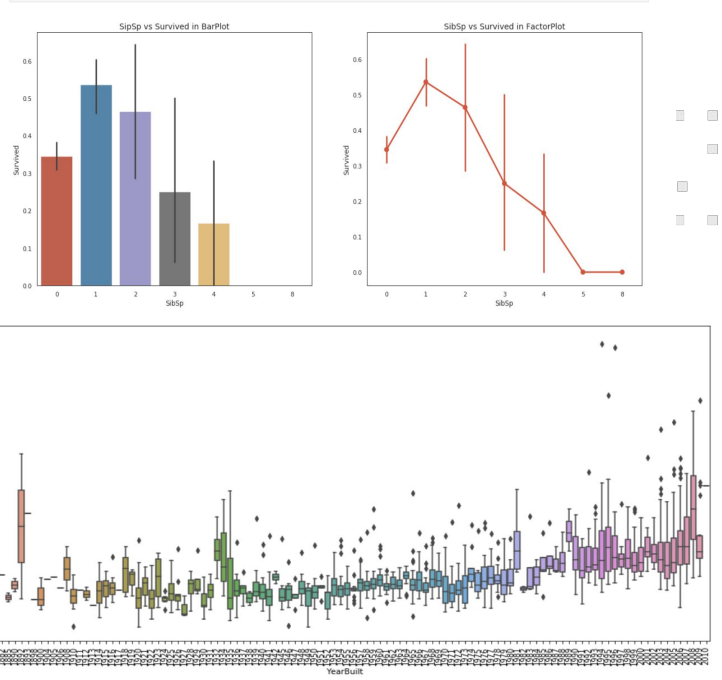




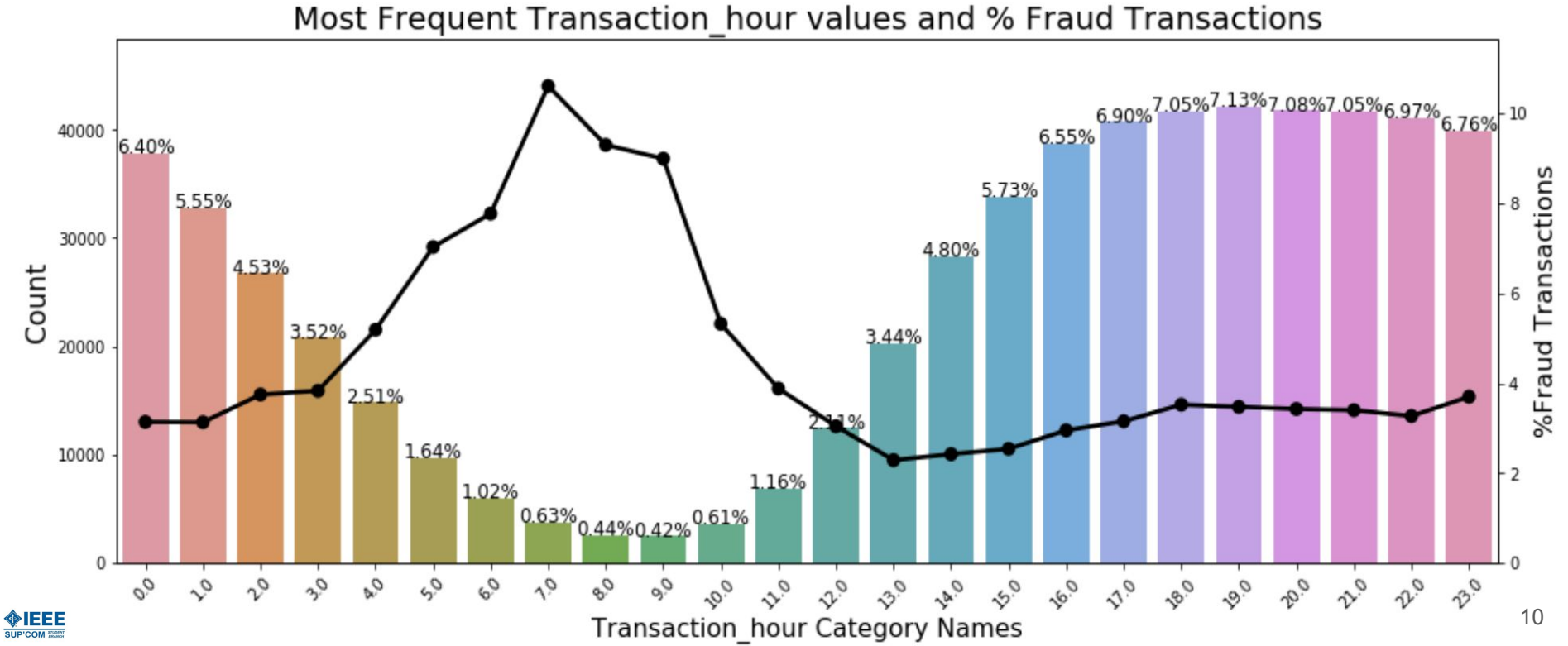
EDA: Don’t do that

- Don’t waste your time making nice graphics with evident analysis
- Focus on discovering new patterns that can lead to hidden information

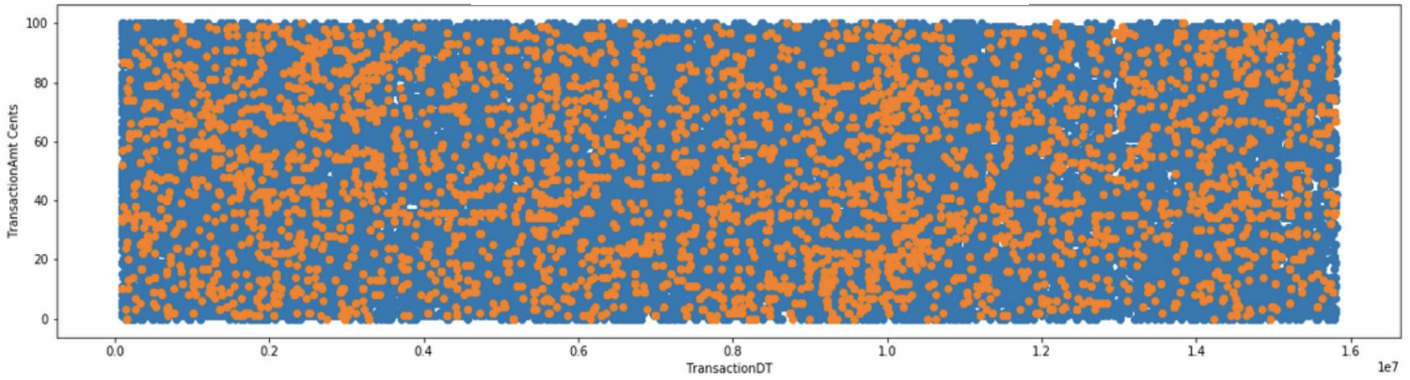


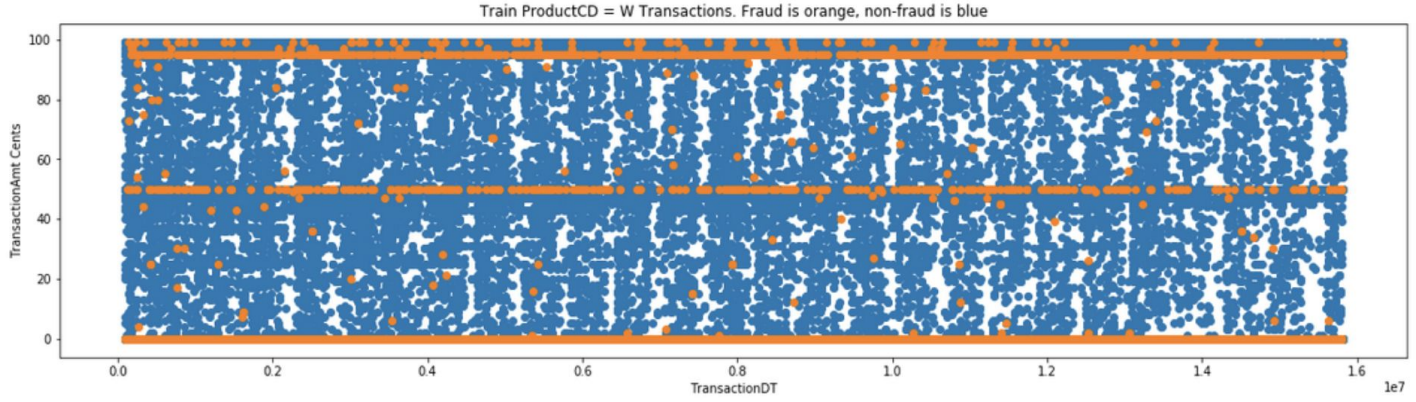


EDA (1) : Fraud detection



EDA (2) : Predictive feature

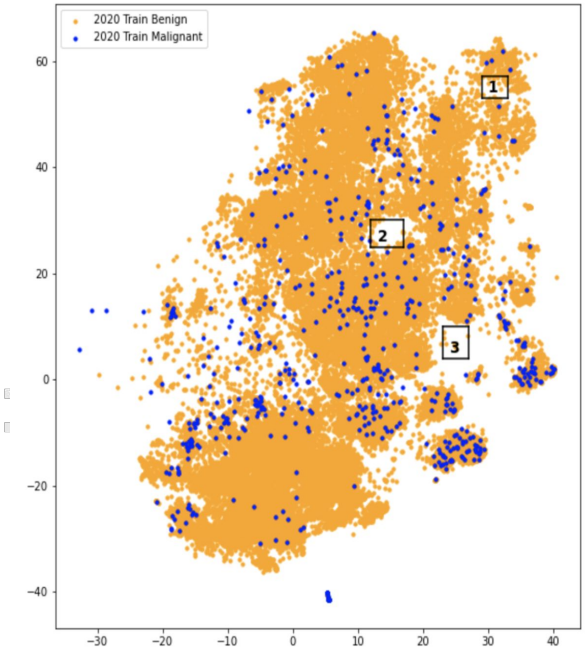


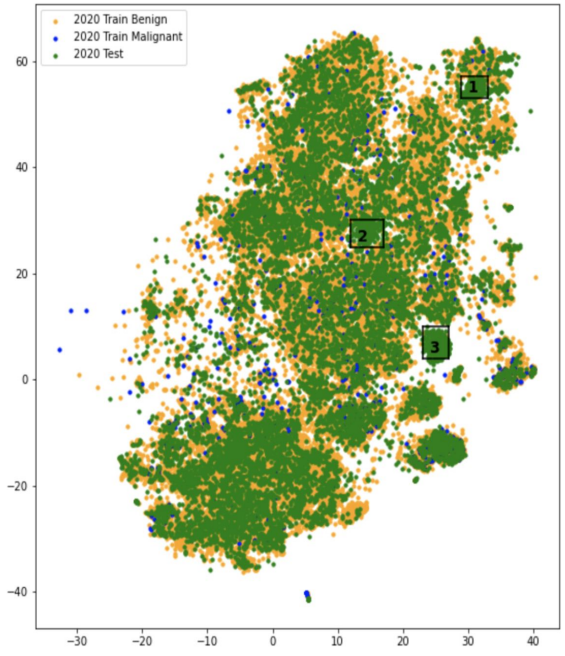


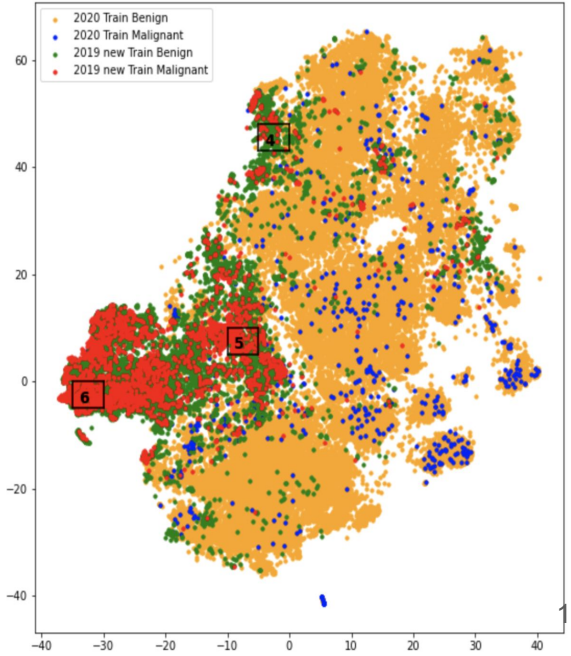
relationalAI

EDA (3) : Embeddings projection

Projected CNN image embeddings into x-y plane using t-SNE (you can use UMAP)





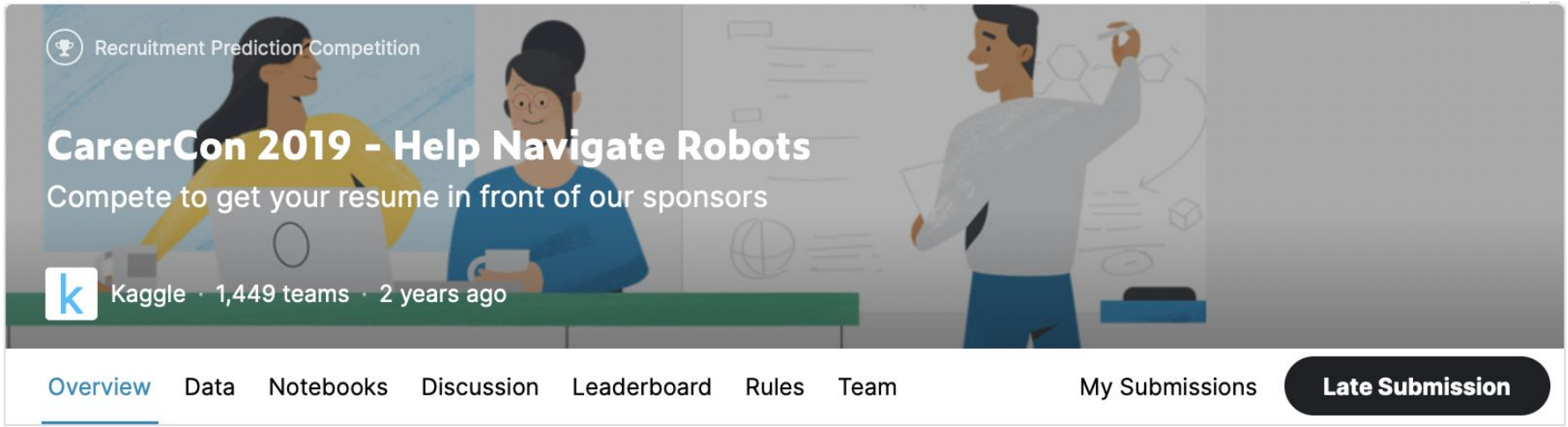


CV

Set up a validation strategy

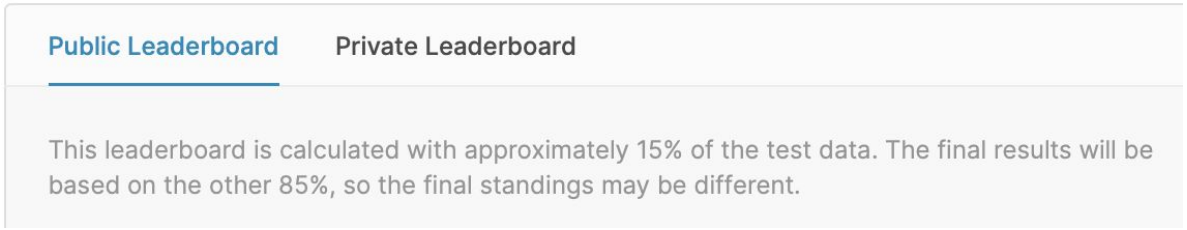
Don’t make the same mistake :’(

789	▼ 686	Seifeddine Fezzani		0.5765	39	2y



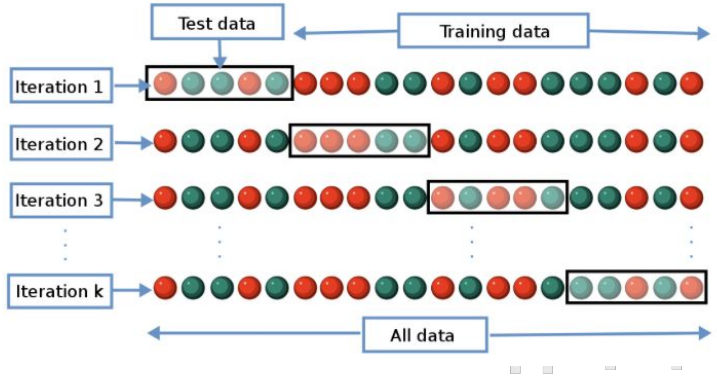
Cross-validation setting

- Be able to know if your model will generalize in the private LB (unseen data)
- Wrong strategy
  - Submit, submit… and see if the public LB score improves

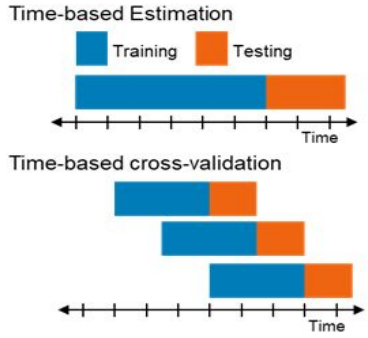


- Right strategy:
  - Understand the problem and set up your validation strategy to have a generalizable solution

Cross-validation strategies



- Random split or k-fold cross-validation
- Stratified k-fold
  - Good when data is unbalanced
- Group k-fold
- Time based cross-validation
  - Train on all train periods except last, and predict last
  - Retrain on all data and submit



- There is an infinity of cross-validation strategies

- When the strategy is fixed, focus on improving your results locally
  - Features engineering, modelling, hyper-parameters tuning : More on this later

Example

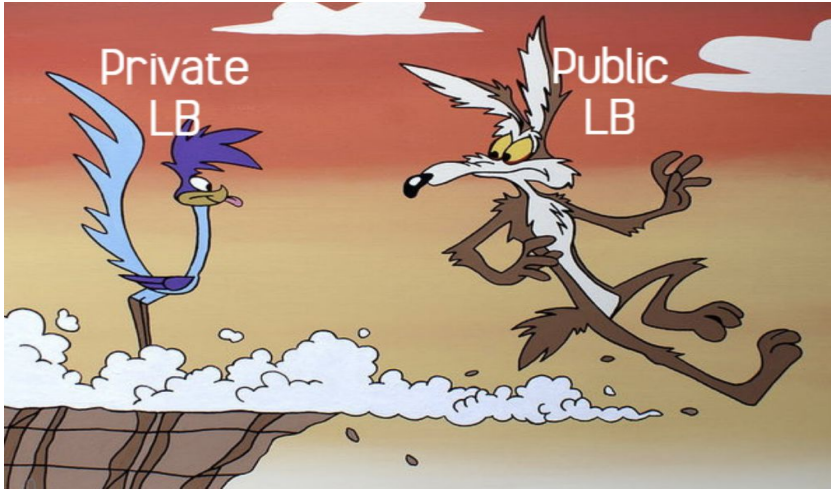
- Which model is the best for you ?

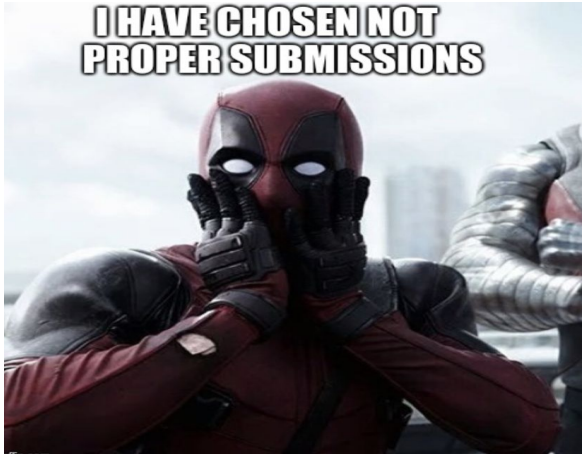
Validation score (accuracy)	Public LB score (accuracy)
0.92	0.91
0.88	0.93

- Trust your CV score **ONLY** when you **trust** your validation strategy

CV memes :p


Public LB: 0.9600
Private LB: 0.9200


Private LB
Public LB


I HAVE CHOSEN NOT PROPER SUBMISSIONS


PUBLIC
PRIVATE


@PadhyRamKrishna
- First time?
NETFLIX


Rank 67 on Kaggle competition public leaderboard
Rank 212 on private leaderboard

relationalAI

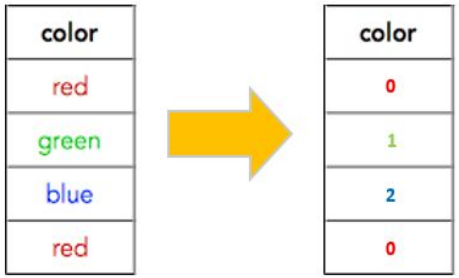
Improve results using data

Preprocessing and Feature Engineering

More data beats clever algorithms, but better data beats more data

-- Peter Norvig --

Deal with categorical features



color
red
green
blue
red

color
0
1
2
0

Label Encoding

Human-Readable

Pet
Cat
Dog
Turtle
Fish
Cat

Machine-Readable

Cat	Dog	Turtle	Fish
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0

One-Hot Encoding

A	0.44 (4 out of 9)
B	0.33 (3 out of 9)
C	0.22 (2 out of 9)

Feature	Encoded Feature
A	0.44
A	0.44
A	0.44
A	0.44
B	0.33
B	0.33
B	0.33
C	0.22
C	0.22

Frequency Encoding

- My secret sauce: Embeddings :D

- Use a neural network to create dense embeddings from categorical variables

role	role 3-D embedding
manager	[0.05, 0.10, 0.96]
engineer	[0.72, 0.66, 0.17]
scientist	[0.75, 0.62, 0.15]
manager	[0.05, 0.10, 0.96]
engineer	[0.72, 0.66, 0.17]
engineer	[0.72, 0.66, 0.17]

FE for tabular datasets

- ***Most creative aspect of data science***
- ***Good EDA => Good features***

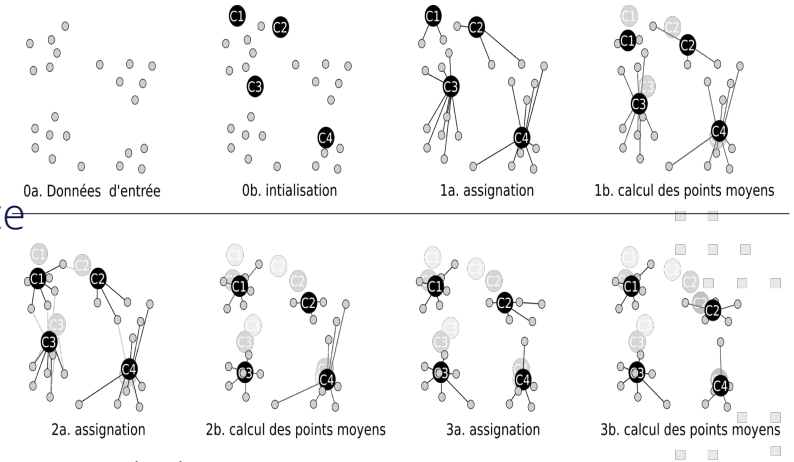
- Count encoding
- Target encoding

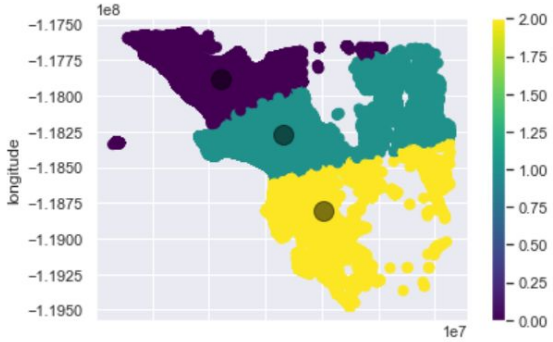
- Time features
- Coordinates features
  - Clustering: K-Means
  - Euclidean distance

- Interactions: Addition, Subtraction, Multiplication, Concatenation
  - Weird interactions can give significant improvement! Be careful of overfitting

Spatial features

- Variables that encode a location in space
  - Cities, countries, addresses…
- Raw longitude latitude
  - K-Means clustering and Elbow
  - Find closeness between a location and a major hub

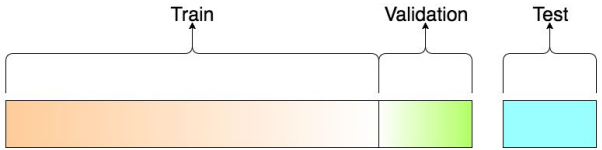




- Group Statistics features [count, nunique, mean, std, max, min, skew]: Target Encoding

Temporal features

- Temporal variables like dates, need good local validation schemes
  - Easy to make mistakes especially if you're using **target encoding**
  - Many people do target encoding before splitting into train and validation




```
for data in [train_df, test_df]:
    data['datetime'] = pd.to_datetime(data['date_time'])
    data['year'] = data['datetime'].dt.year
    data['weekofyear'] = data['datetime'].dt.weekofyear
    data['month'] = data['datetime'].dt.month
    data['dayofweek'] = data['datetime'].dt.dayofweek
    data['hour'] = data['datetime'].dt.hour
```

- Trend features
  - Spend in last week, spend in last month , spend in last year
    - Two consumers with equal spend, can have different behavior
- Closeness to major events

Categorical features and LightGBM / XgBoost / Catboost

- You have the choice of telling these models that these features are categorical or numerical

- Try both ways and see which gives the highest CV

```
for data in [train_df, test_df]:
    for c in categorical_features:
        data[c] = data[c].astype('category')
```

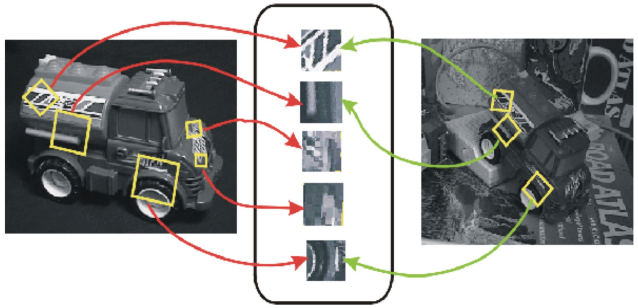
Reduce memory usage

```python
def reduce_mem_usage(df, verbose=True):
    numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']
    start_mem = df.memory_usage().sum() / 1024**2
    for col in df.columns:
        col_type = df[col].dtypes
        if col_type in numerics:
            c_min = df[col].min()
            c_max = df[col].max()
            if str(col_type)[:3] == 'int':
                if c_min > np.iinfo(np.int8).min and c_max < np.iinfo(np.int8).max:
                    df[col] = df[col].astype(np.int8)
                elif c_min > np.iinfo(np.int16).min and c_max < np.iinfo(np.int16).max:
                    df[col] = df[col].astype(np.int16)
                elif c_min > np.iinfo(np.int32).min and c_max < np.iinfo(np.int32).max:
                    df[col] = df[col].astype(np.int32)
                elif c_min > np.iinfo(np.int64).min and c_max < np.iinfo(np.int64).max:
                    df[col] = df[col].astype(np.int64)
            else:
                if c_min > np.finfo(np.float16).min and c_max < np.finfo(np.float16).max:
                    df[col] = df[col].astype(np.float16)
                elif c_min > np.finfo(np.float32).min and c_max < np.finfo(np.float32).max:
                    df[col] = df[col].astype(np.float32)
                else:
                    df[col] = df[col].astype(np.float64)

    end_mem = df.memory_usage().sum() / 1024**2
    print('Memory usage after optimization is: {:.2f} MB'.format(end_mem))
    print('Decreased by {:.1f}%'.format(100 * (start_mem - end_mem) / start_mem))

    return df
```

Feature Engineering (Images)



- Manual features
  - SIFT
    - Detect and describe local features in an image
    - Invariant to image translation, scaling and rotation

=> SIFT and RANSAC are good at **unsupervised image matching**

- Image embeddings
  - Use a pre-trained model on ImageNet
  - Use a self-supervised model like SimCLR
    - Surprisingly good

Data Augmentation (Images)

- Before modelling
  - Imgaug
  - TorchVision
  - Albumentations

```
train_transform = A.Compose(
    [
        A.SmallestMaxSize(max_size=160),
        A.ShiftScaleRotate(shift_limit=0.05, scale_limit=0.05, rotate_limit=15, p=0.5),
        A.RandomCrop(height=128, width=128),
        A.RGBShift(r_shift_limit=15, g_shift_limit=15, b_shift_limit=15, p=0.5),
        A.RandomBrightnessContrast(p=0.5),
        A.Normalize(mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225)),
        ToTensorV2(),
    ]
)
```

- After modelling
  - TTA

Cleaning text

- Lowercasing
  - I work at NASA => i work at nasa
- Dealing with punctuation

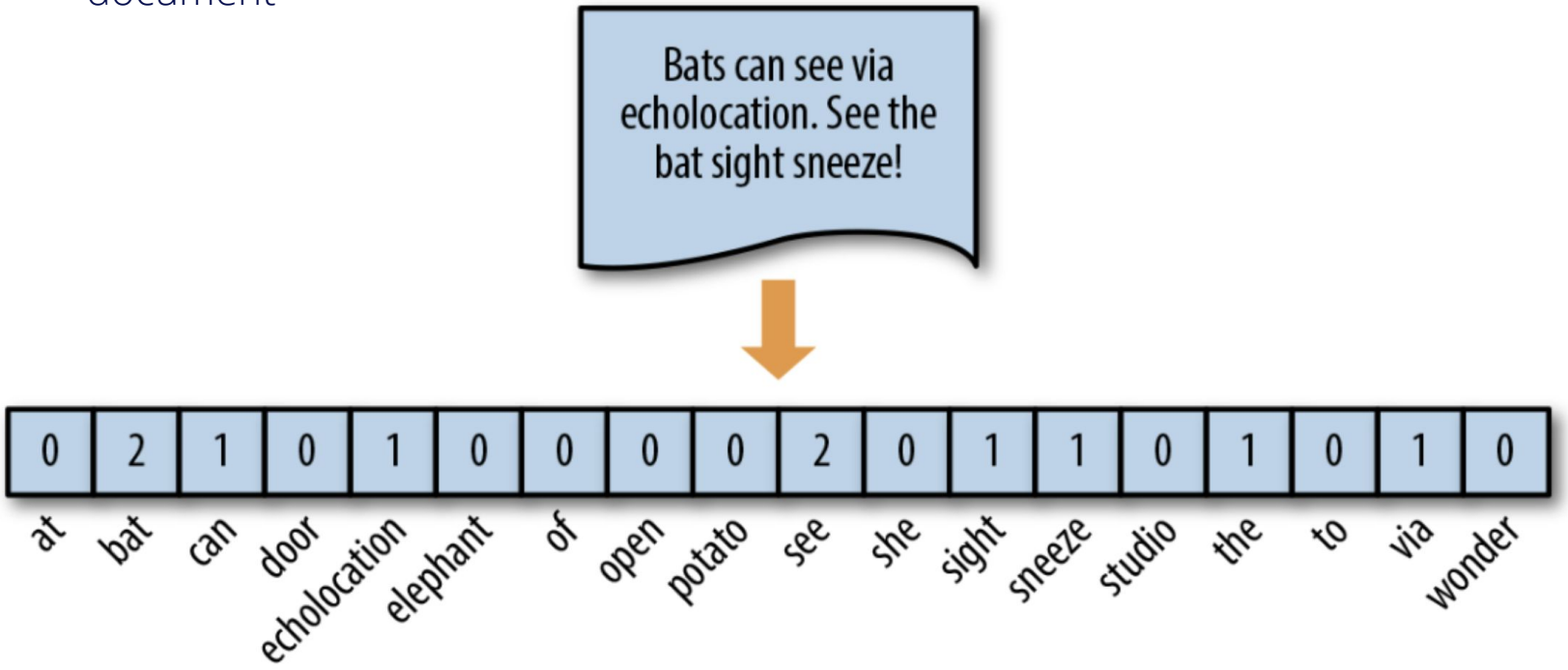
- Stemming
  - Reduce a word to its root : cars => car
- Lemmatization
  - Find semantic root : never be late => never are late

- Removing stop words
  - The, is, I, he, she, it, are…
- Spelling correction
  - I ws here before => I was here before

- Tokenizing

Bag of Words

- BoW creates a set of vectors containing the count of word occurences in the document



TF-IDF

- TF-IDF contains information on the more important words and the less important ones
- Hard to interpret than BoW but usually performs better in machine learning models

- Review 1: This movie is very scary and long
- Review 2: This movie is not scary and is slow
- Review 3: This movie is spooky and good

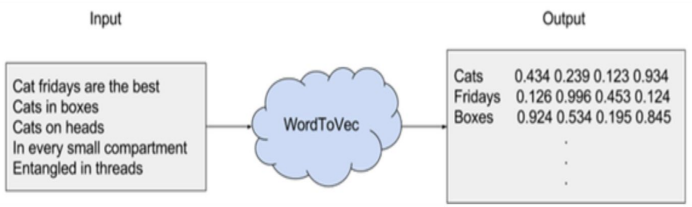
Term	Review 1	Review 2	Review 3	TF (Review 1)	TF (Review 2)	TF (Review 3)	IDF	TF-IDF (Review 1)	TF-IDF (Review 2)	TF-IDF (Review 3)
This	1	1	1	1/7	1/8	1/6	0.00	0.000	0.000	0.000
movie	1	1	1	1/7	1/8	1/6	0.00	0.000	0.000	0.000
is	1	2	1	1/7	1/4	1/6	0.00	0.000	0.000	0.000
very	1	0	0	1/7	0	0	0.48	0.068	0.000	0.000
scary	1	1	0	1/7	1/8	0	0.18	0.025	0.022	0.000
and	1	1	1	1/7	1/8	1/6	0.00	0.000	0.000	0.000
long	1	0	0	1/7	0	0	0.48	0.068	0.000	0.000
not	0	1	0	0	1/8	0	0.48	0.000	0.060	0.000
slow	0	1	0	0	1/8	0	0.48	0.000	0.060	0.000
spooky	0	0	1	0	0	1/6	0.48	0.000	0.000	0.080
good	0	0	1	0	0	1/6	0.48	0.000	0.000	0.080

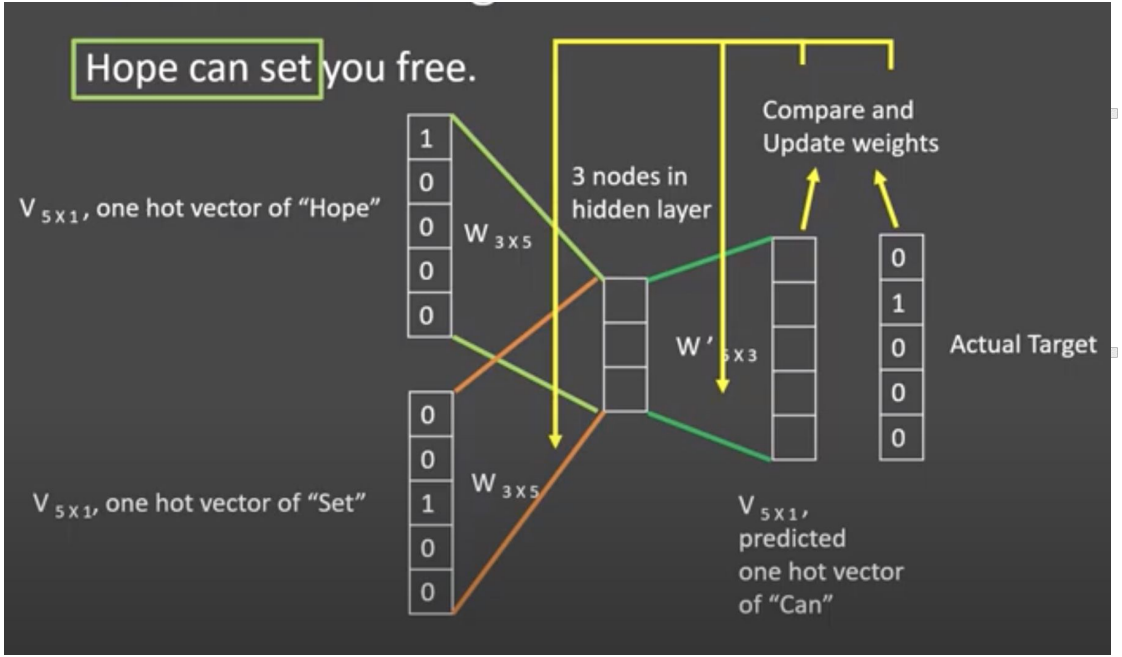
$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

tf_{ij} = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Embeddings with Word2Vec

- Preserves relationship between words
- The word2vec objective function causes the words that occur in similar contests to have similar embeddings





Features Engineering (Text)

- Manual features
  - Number of words/characters etc...
  - Number of toxic words (Classify toxic contents)
  - Number of symbols: (? / & # @ $ £ + _ ^ ¨ ; : = ` * . -) (Spam detection)
  - Jaccard similarity (Text similarity)
  - Edit distance (Text similarity)
  - Aggregations and group statistics [mean, std, max, min]

- BoW
- TF-IDF
- Embeddings
  - Word Mover Distance
  - Features derived from K-Nearest Neighbor and cosine similarity

Feature Selection

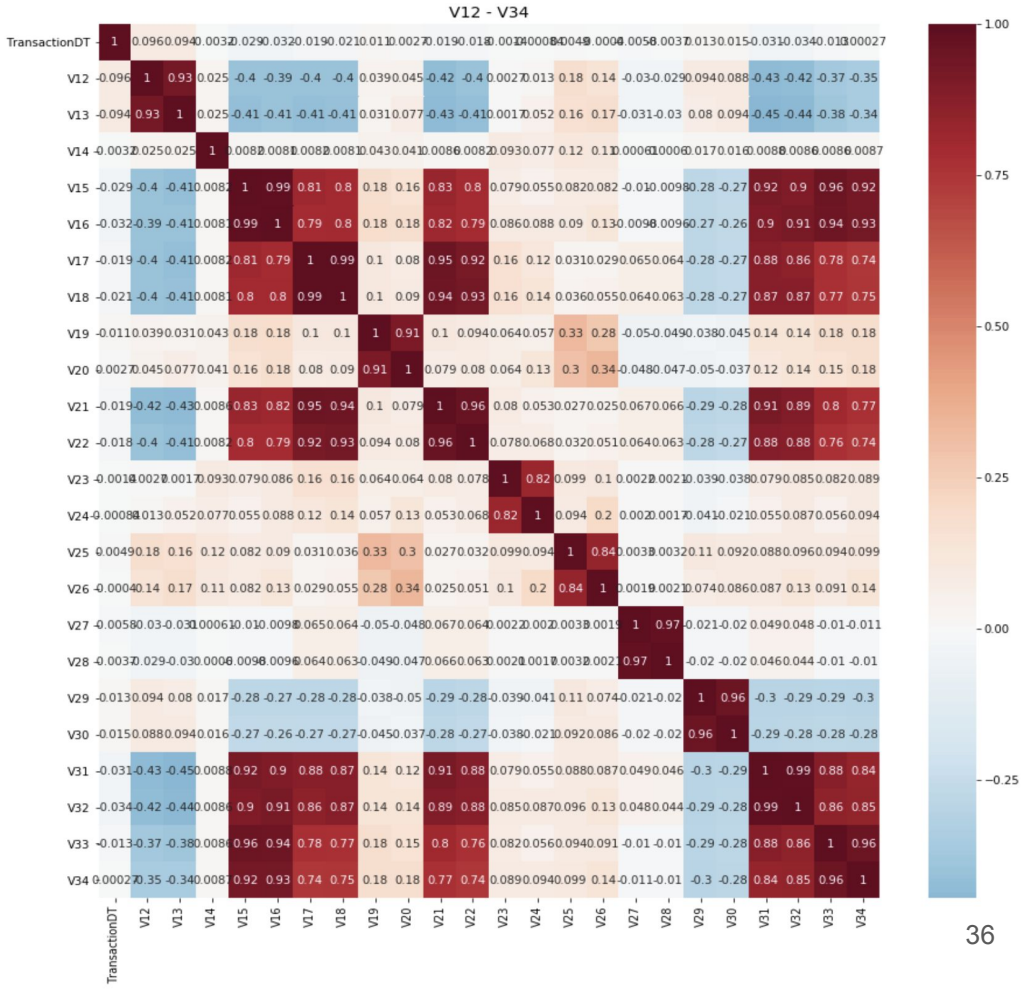
Keep features that improve CV score

Dropping features with a lot of missing values

- Set a threshold : 0.8 - 0.9
- Eliminate features with missing values >= threshold
- See if validation score improves or not

Correlation matrix

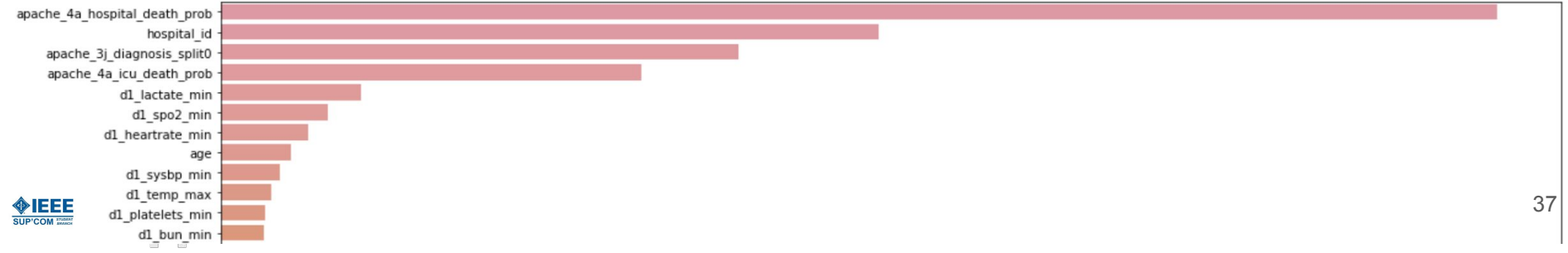
- Eliminate high correlated features
  - Threshold >= 0.8
- See if validation score improves or not
  - Results will depend on the model used



Adversarial Validation

- Goal : Remove features with different distribution in train and test
- Why ? These features will boost the validation score but will perform poorly on test

- Solution:
  - Label train dataset: 0 and test dataset: 1
  - Train a classifier and see if it discriminates between train and test using the features
  - If AUC is high => there are features with different distribution, we find them using features importance
  - Remove top features and retrain the classifier



Technique for binary classification

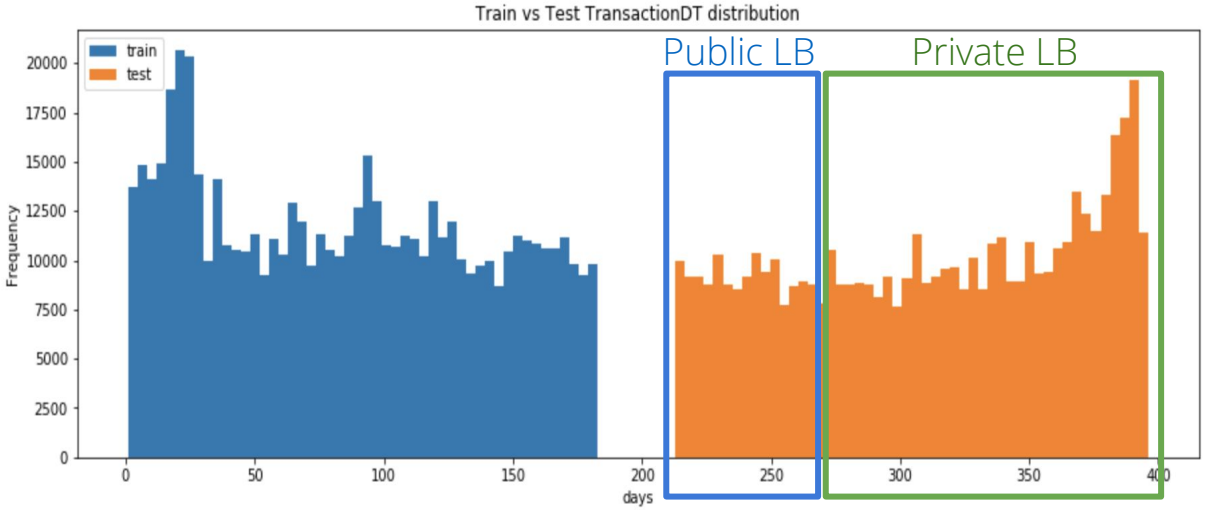
- For each feature in dataframe:
  - Train a classifier (XgBoost for example)
  - If AUC >= 0.5
    - Keep the feature
  - If AUC < 0.5
    - Remove the feature

- See if validation score improves with the new set of features

Feature selection based on time consistency

- Build one model for each feature
- Each model is trained on the first two months of the training data
- Predict on the last two months of the training data

-> Drop features with AUC score < 0.5



relationalAI

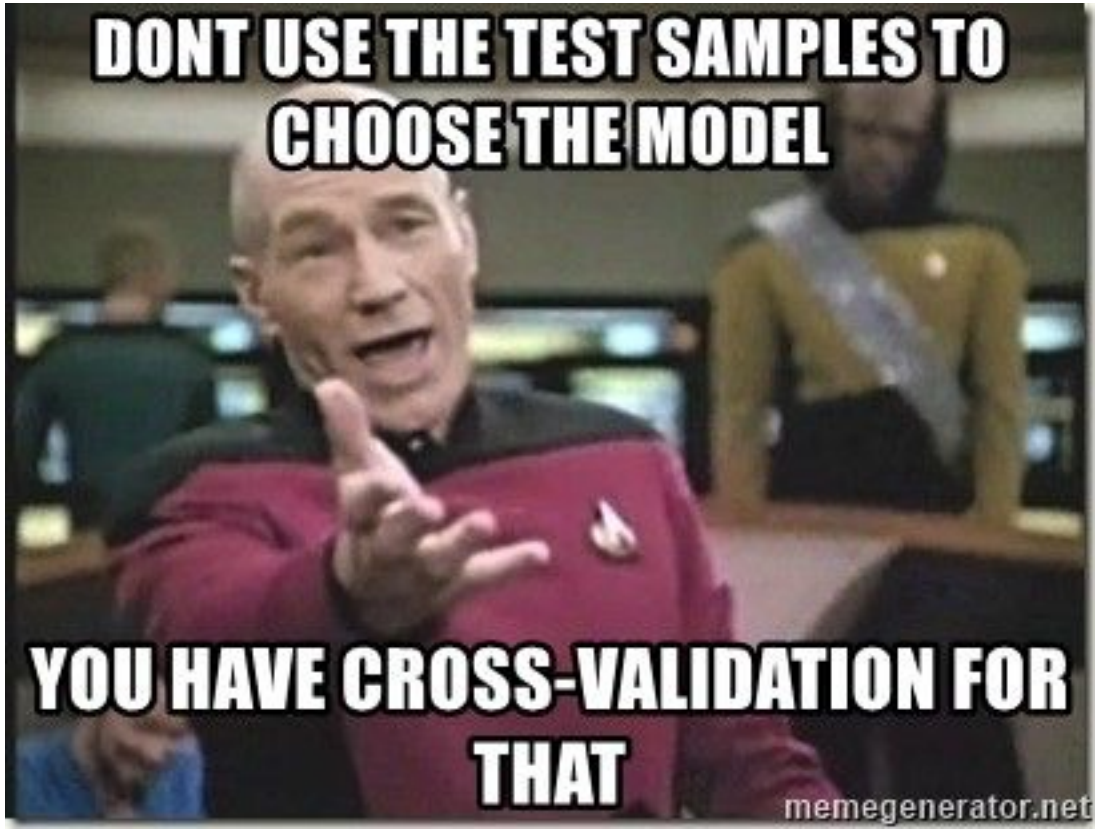
Modelling

Choose the model that gives you the highest CV ;)

Hyperparameters tuning

- Impossible without a reliable a validation strategy
- Don’t overtune your parameters:
  - Focus on tuning only when you have your final features

Common mistake



SVM, logistic regression

SVM, Logistic Regression

- Easy to tune
  - Learning rate
  - Regularization parameter C
    - Increase C if train/validation gap is large
- Few hyper-parameters
  - GridSearch is a good option
- Use sklearn library to train

Gradient boosting decision trees models

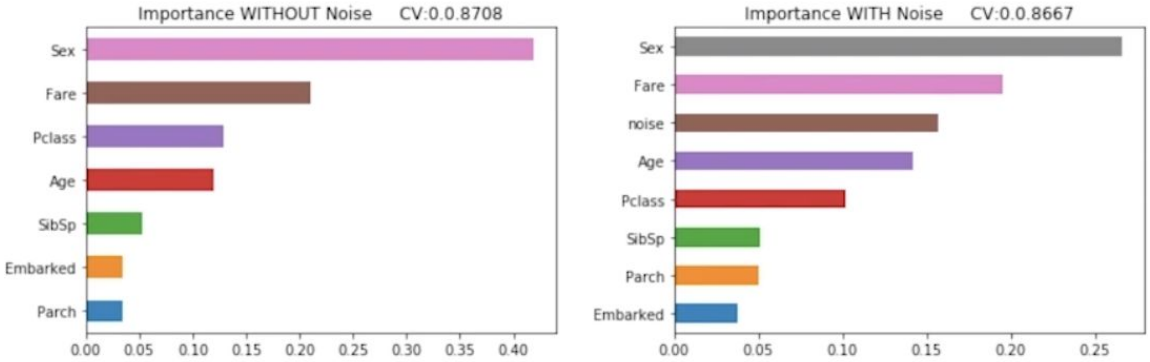
XgBoost / LightGBM / CatBoost

- Base learner = decision trees
  - It uses **entropy** or **gini** as a criterion at each split to find the feature that it should use for splitting

- My tuning steps
  - Start with subsample (colsample_bytree) = 0.7, leave other values to default
    - Decrease subsample value a bit if you have a lot of features and only few of them are dominant in features importance
  - Tune max_depth or number_of_leaves: default number = 7
    - Decrease if train/validation gap is large
  - Tune min_child_weight
    - Increase if train/validation gap is large
  - Play with regularisation parameters

Don’t trust feature importance

- Idea : Introduce a noisy feature and eliminate features with lower importance



- **The noisy feature seems more important than the others : WRONG**

Feature importance

- Low ranked features can be important
  - Decrease **colsample_bytree** value to 0.7-0.8-0.9 instead of 1.0
  - See if it improves the validation score

- Gradient Boosting Decision Trees models pay more attention to features with high entropy
  - Features with uniform distribution give maximum recall

relationalAI

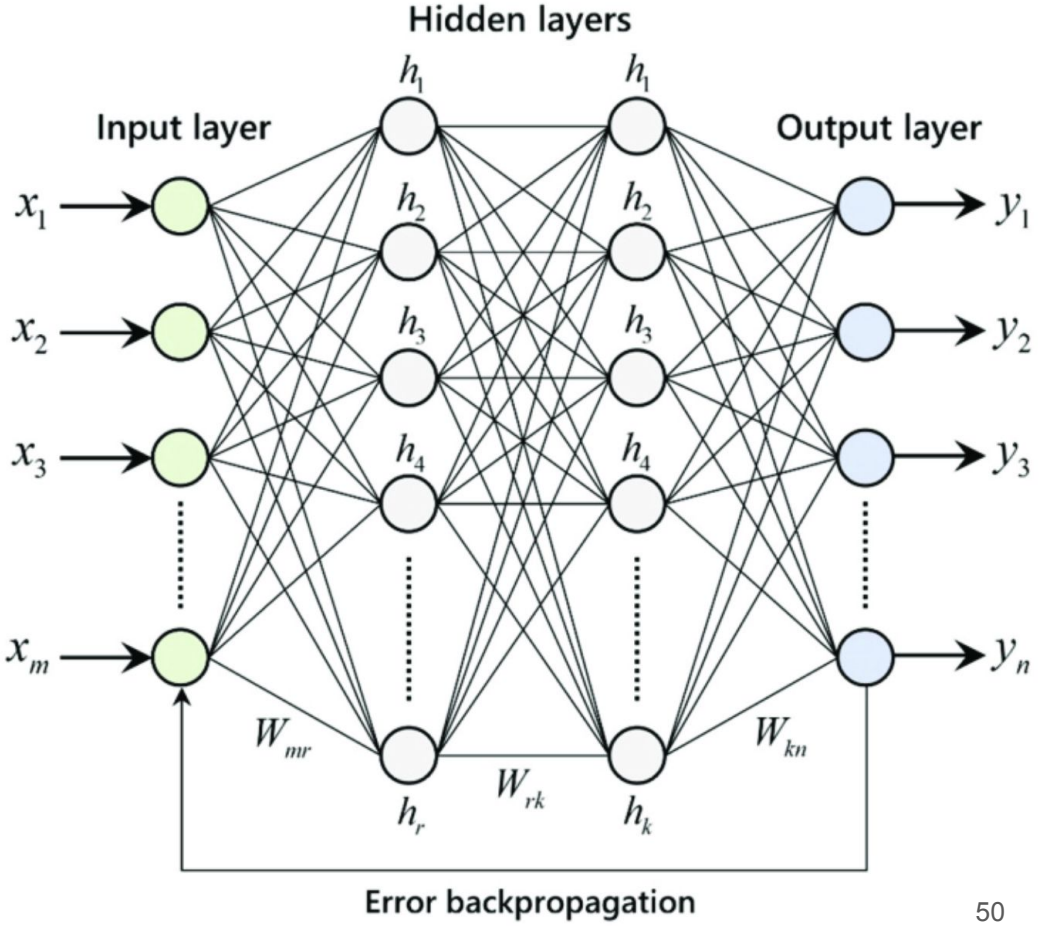
Neural Networks

Neural Networks

- Training process:
  - For x iterations:
    - Forward propagation
    - Compute loss
    - Back propagation
    - Update parameters

Same steps for all NN

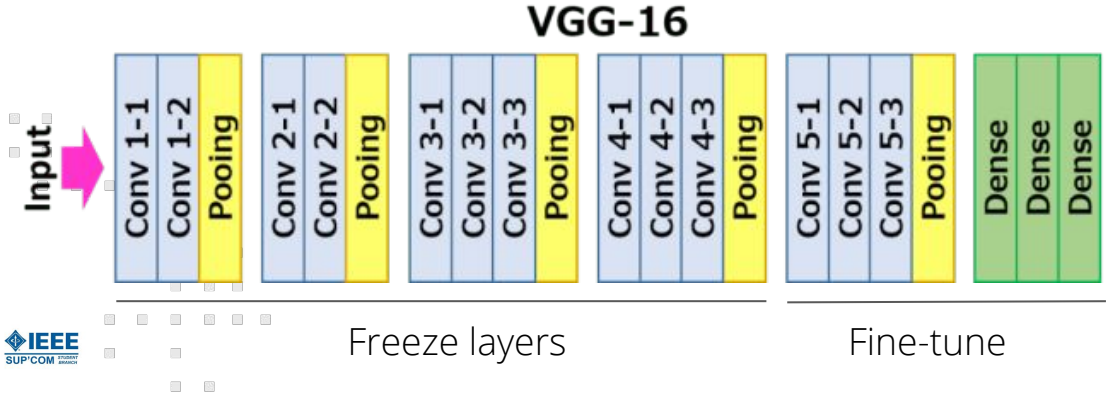
- Hyper-parameters
  - Learning rate
  - Number of layers
  - Number of hidden units
  - L1, L2 regularization

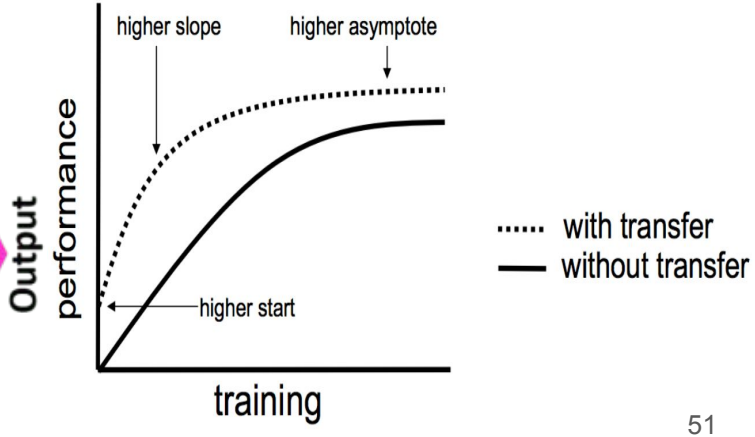


Transfer Learning

- Machine learning technique where a model trained on one task is re-purposed on a second related task
- It's an optimization that allows rapid progress and improved performance in the second task
- Used a lot in computer vision and nlp

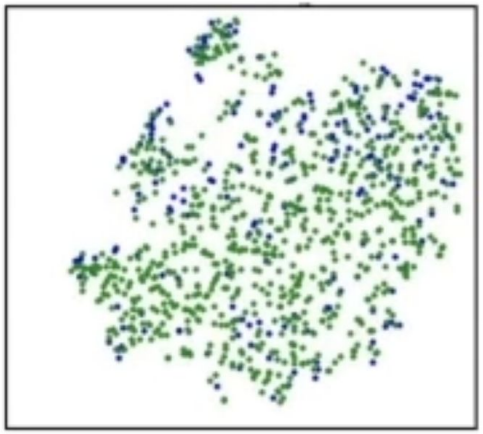
- Transfer Learning => Fine-Tuning



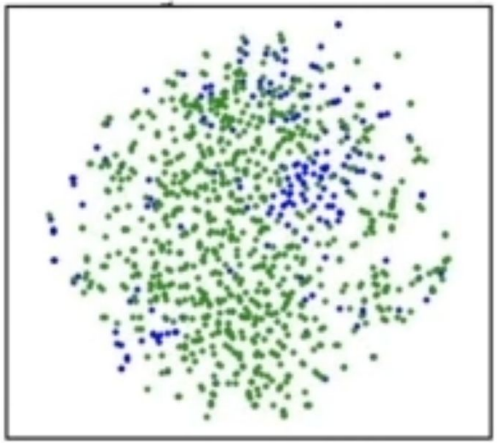


Fine-tuning

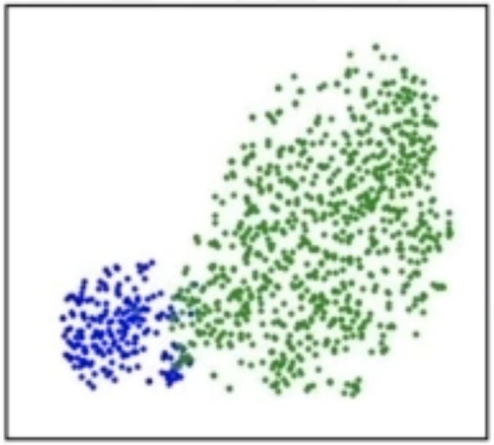
- Fine-tuning always gives better results
- Freeze first weights of NN and train the remaining ones



Random Weights



Pre-Trained



Fine-Tuned

Benchmark Vision: ImageNet and COCO datasets

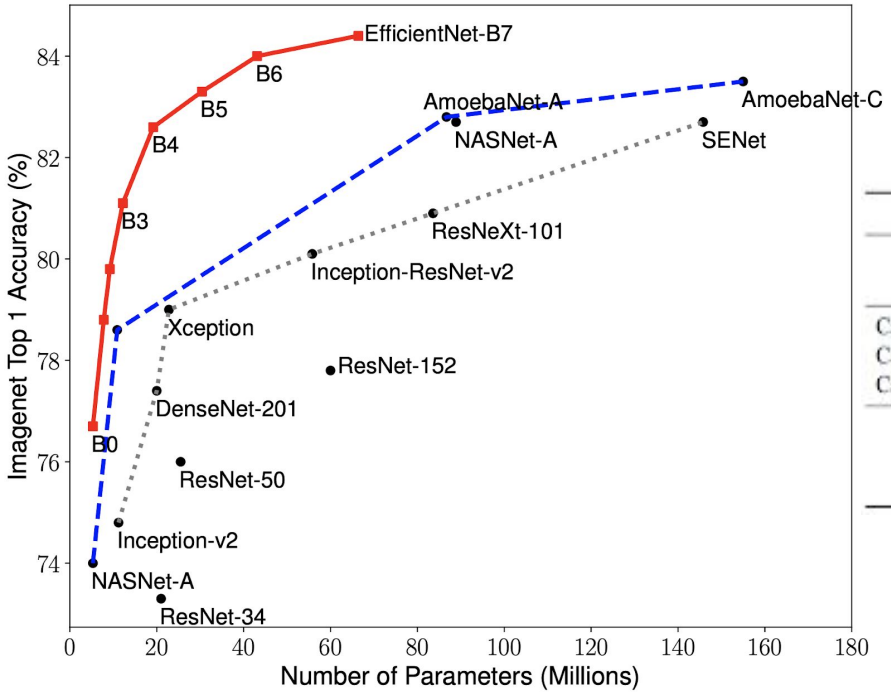
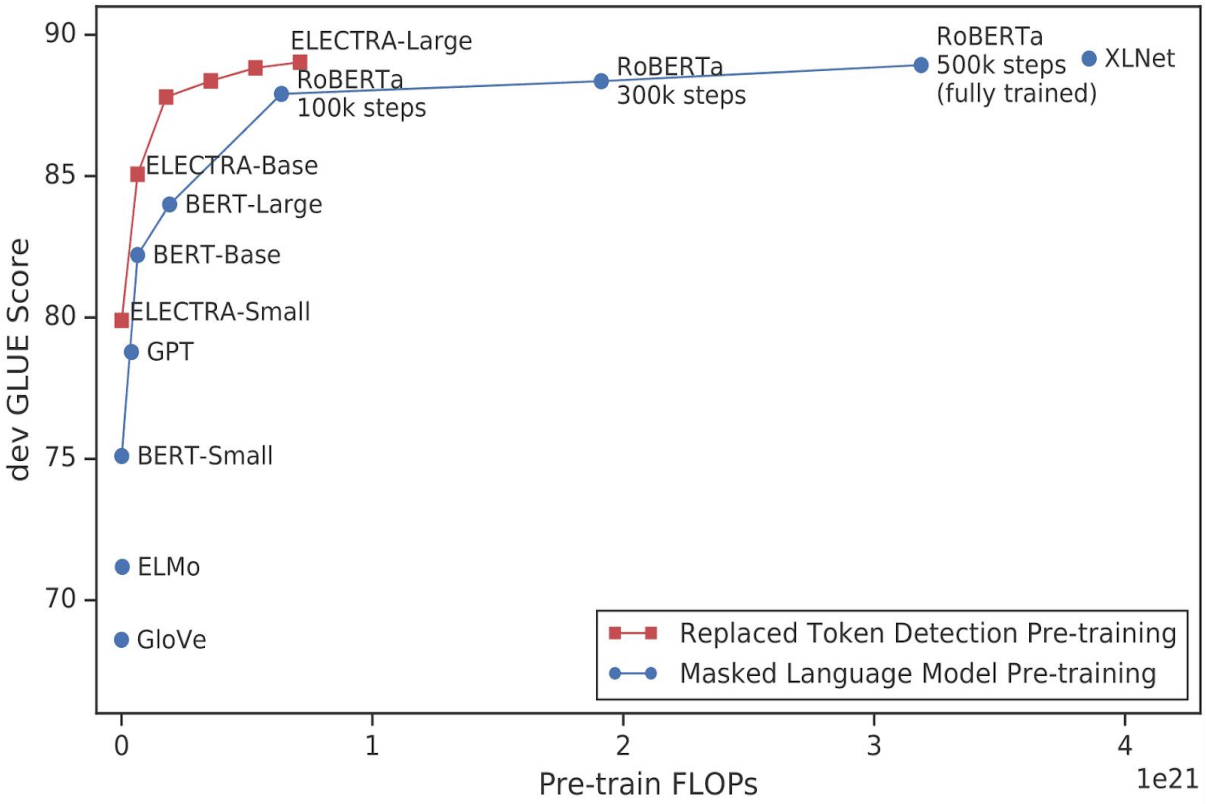


Table 1: Comparison with state-of-the-art methods on COCO test-dev dataset.

Method	Backbone	box AP	mask AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	runtime (fps)
Mask R-CNN [18]	ResNet-50-FPN	39.1	35.6	57.6	38.1	18.7	38.3	46.6	5.3
PANet[28]	ResNet-50-FPN	41.2	36.6	58.0	39.3	16.3	38.1	52.4	-
Cascade Mask R-CNN	ResNet-50-FPN	42.7	36.9	58.6	39.7	19.6	39.3	48.8	3.0
Cascade Mask R-CNN	ResNet-101-FPN	44.4	38.4	60.2	41.4	20.2	41.0	50.6	2.9
Cascade Mask R-CNN	ResNeXt-101-FPN	46.6	40.1	62.7	43.4	22.0	42.8	52.9	2.5
HTC (ours)	ResNet-50-FPN	**43.6**	**38.4**	60.0	41.5	20.4	40.7	51.2	2.5
HTC (ours)	ResNet-101-FPN	**45.3**	**39.7**	61.8	43.1	21.0	42.2	53.5	2.4
HTC (ours)	ResNeXt-101-FPN	**47.1**	**41.2**	63.9	44.7	22.8	43.9	54.6	2.1

Benchmark NLP: SQuAD 2.0 question answering dataset

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar & Jia et al. '18)	86.831	89.452
1 Apr 06, 2020	SA-Net on Albert (ensemble) *QIANXIN*	**90.724**	**93.011**
2 May 05, 2020	SA-Net-V2 (ensemble) *QIANXIN*	90.679	92.948
2 Apr 05, 2020	Retro-Reader (ensemble) *Shanghai Jiao Tong University* http://arxiv.org/abs/2001.09694	90.578	92.978
3 Jul 31, 2020	ATRLP+PV (ensemble) *Hithink RoyalFlush*	90.442	92.877
3 May 04, 2020	ELECTRA+ALBERT+EntitySpanFocus (ensemble) *SRCB_DML*	90.442	92.839
4 Jun 21, 2020	ELECTRA+ALBERT+EntitySpanFocus (ensemble) *SRCB_DML*	90.420	92.799

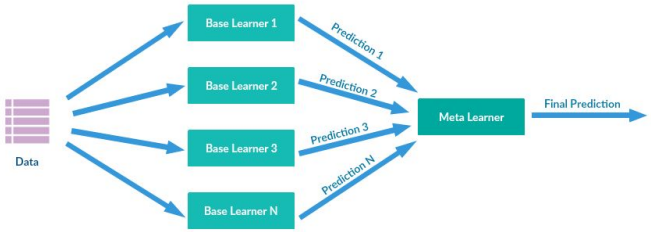


Ensembling

Blending & Stacking

Ensembling

- Train many models and combine them
  - Blending
    - Choose the weights that improves your cv:
      - **y_final = w1*y1 + w2*y2**
  - Stacking
    - Use same folds for all models to avoid data leakage
    - Use out of fold predictions as feature for second level of models
      - **$y\_final = F_w(y1, y2)$**
        - F is the meta learner (ML model)



- Don't start too early in the competition
- Diversity of models is important

Submit :D

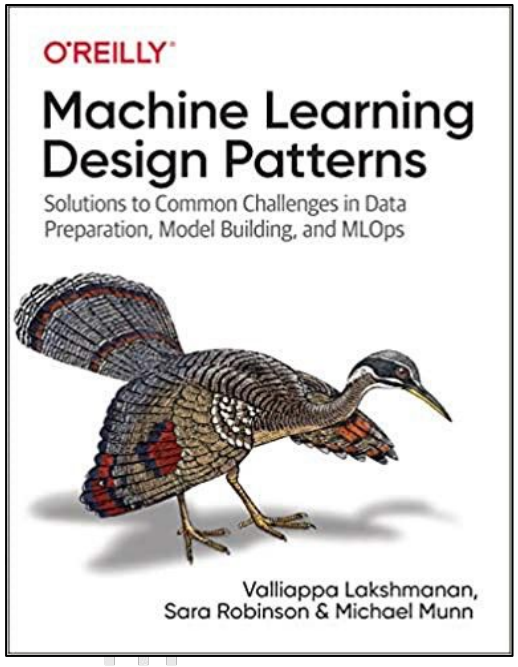
Advices

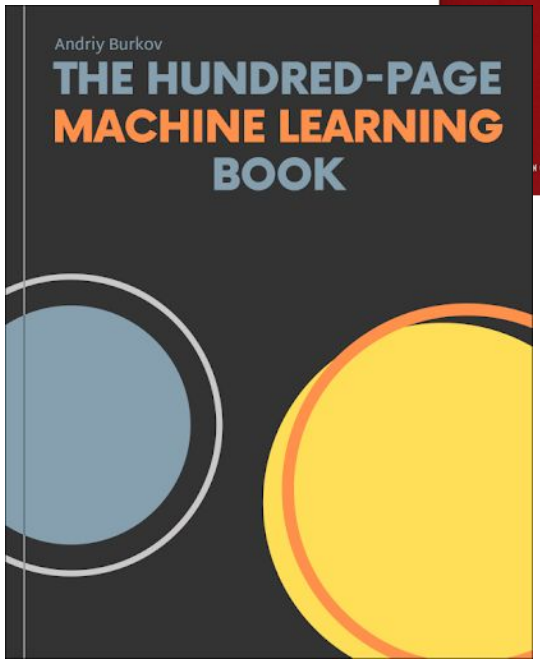
- Always understand the problem
- Don’t use kernels when you start a competition, find new ideas
  - You can surprise yourself

- Work on different types of competitions
- Always start by writing a baseline submission

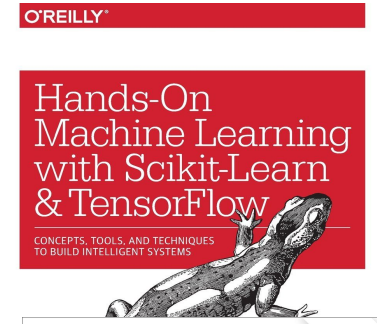
- Be humble
- Be active in discussions and make new connections
- Focus on learning more than ranking
- **Find people that inspire you**

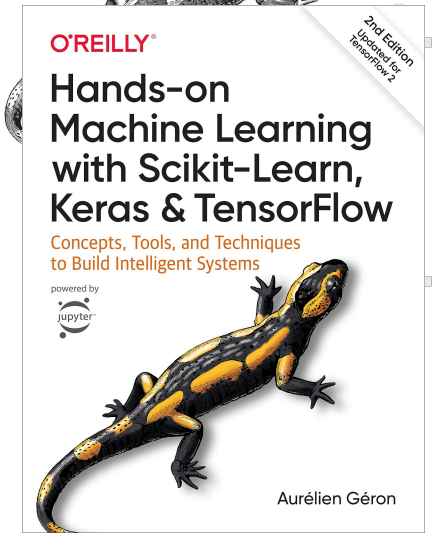
Great books


O'REILLY
Machine Learning Design Patterns
Solutions to Common Challenges in Data Preparation, Model Building, and MLOps
Valliappa Lakshmanan, Sara Robinson & Michael Munn


Andriy Burkov
THE HUNDRED-PAGE MACHINE LEARNING BOOK


DEEP LEARNING with Python
François Chollet


O'REILLY
Hands-On Machine Learning with Scikit-Learn & TensorFlow
CONCEPTS, TOOLS, AND TECHNIQUES TO BUILD INTELLIGENT SYSTEMS


O'REILLY
2nd Edition Updated for TensorFlow 2
Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow
Concepts, Tools, and Techniques to Build Intelligent Systems
powered by jupyter
Aurélien Géron

See you in kaggle ;)

Questions & Answers QA