A Few Comments on Competitive Data Science

Dmitry Adamskiy

Outline

EDA

Exploratory Data Analysis

- Normally, the starting point of every project
- Make yourself comfortable with the data
- Informs model selection. . .

Exploratory Data Analysis

- Normally, the starting point of every project
- Make yourself comfortable with the data
- Informs model selection...
- ...data preprocessing ...

Exploratory Data Analysis

- Normally, the starting point of every project
- Make yourself comfortable with the data
- Informs model selection...
- ...data preprocessing ...
- ...validation strategy

Exploratory Data Analysis

- Normally, the starting point of every project
- Make yourself comfortable with the data
- Informs model selection. . .
- . . . data preprocessing . . .
- . . . validation strategy
- So all in all hugely important . . .
- . . . but is somewhat of an art form so we won't talk a lot about it.

Feature types

- Numerical
- Categorical
- Ordinal
- Date/time
- Coordinates
- Text
- ...

Numerical features

Potential pre-processing strategies:

- Scaling
- Rank transformation
- Clipping
- Non-linear monotonic transformation such as $\log$

Exercise: Think which models are sensitive to scaling and which are not.

Categorical features

Potential ways of dealing with them:

- One-hot encoding
- Label encoding
- Frequency encoding
- Target encoding
- Learn Embeddings

Categorical features – one-hot encoding

- One-hot encoding

Categorical features – one-hot encoding

- One-hot encoding
    - Traditional method.

City	City_Moscow	City_London	City_Arzamas-16
Moscow	1	0	0
London	0	1	0
Arzamas-16	0	0	1
London	0	1	0

    - For categories with lots of levels (ids) will results in a large (but sparse) representations – potential problem for sampling features.

Categorical variables – label encoding

City	City_numeric
Moscow	0
London	1
Arzamas-16	2
London	1

Categorical variables – label encoding

City	City_numeric
Moscow	0
London	1
Arzamas-16	2
London	1

- Rather pointless for linear / distance-based predictors or neural nets

Categorical variables – label encoding

City	City_numeric
Moscow	0
London	1
Arzamas-16	2
London	1

- Rather pointless for linear / distance-based predictors or neural nets
- But works fine with tree-based models
- You can encode missing values as -1, this was it is easy to split them off.

Mean encoding

Here is an idea – we encode each level with some statistic of target variable at this level.

City	Mean encoding	Target
Moscow	1	1
London	1	1
Arzamas-16	0	0
Birmingham	0.5	0
London	1	1
Birmingham	0.5	1

Mean encoding

Here is an idea – we encode each level with some statistic of target variable at this level.

City	Mean encoding	Target
Moscow	1	1
London	1	1
Arzamas-16	0	0
Birmingham	0.5	0
London	1	1
Birmingham	0.5	1

There are lots of potential pitfalls here. . .

Competition Metrics

Competition Metrics – why do we care?

- Each competition has it's own evaluation metric. You can check it in the 'evaluation' tab. This is how the models on the leaderboard(s) are scored and ultimately this should be the only thing you care about :)
- So it is very tempting to try to optimize this criterion. However, it is not always possible.
- You should understand how different criteria treat extreme predictions and outliers.
- Some of them are not so easy to understand (like weighted kappa).

Classification: common metrics

- Accuracy
- AUC
- Average log-loss
- (weighted) kappa

Accuracy

- The simplest one possible – a fraction of correctly classified examples:

$$\text{Acc} = \frac{\sum_{i=1}^{N} 1[y_i = \hat{y}_i]}{N}$$

Accuracy

- The simplest one possible – a fraction of correctly classified examples:

$$\text{Acc} = \frac{\sum_{i=1}^{N} 1[y_i = \hat{y}_i]}{N}$$

- Problem: works with hard predictions and does not take into account the confidence.

Accuracy

- The simplest one possible – a fraction of correctly classified examples:

$$\text{Acc} = \frac{\sum_{i=1}^{N} 1[y_i = \hat{y}_i]}{N}$$

- Problem: works with hard predictions and does not take into account the confidence.
- Thus, impossible to optimise directly with gradient-based methods (Exercise: what is the gradient?)

Accuracy

- The simplest one possible – a fraction of correctly classified examples:

$$\text{Acc} = \frac{\sum_{i=1}^{N} 1[y_i = \hat{y}_i]}{N}$$

- Problem: works with hard predictions and does not take into account the confidence.
- Thus, impossible to optimise directly with gradient-based methods (Exercise: what is the gradient?)
- Another problem(?): misleading for imbalanced classes. A classifier could get very high accuracy by always predicting the most frequent class.

AUC

- Assumes that the classifier outputs soft predictions $\hat{z}_i$
- Could be probabilities but not necessarily. You can get hard predictions by choosing a threshold θ and outputting

$$\hat{y}_i = 1[\hat{z}_i > \theta]$$

- (Explanation on the board on how to compute AUC)

AUC

- Assumes that the classifier outputs soft predictions $\hat{z}_i$
- Could be probabilities but not necessarily. You can get hard predictions by choosing a threshold θ and outputting

$$\hat{y}_i = 1[\hat{z}_i > \theta]$$

- (Explanation on the board on how to compute AUC)
- Probabilistic interpretation of AUC: if you pick two examples, the probability that they are misaligned by the classifier is $1 -$ AUC.

Average log-loss

- The classifier outputs probabilities of the labels $p_i = p(y_i = 1)$.
- The loss is

$$-\frac{1}{N}\sum_{i=1}^{N}(y_i \log p_i + (1 - y_i)\log(1 - p_i))$$

- This is directly optimised in many models.
- Exercise: what is the best constant prediction?

Regression metrics

- (R)MSE
- MAE
- R^2
- . . . and many more!

(R)MSE

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

- One of the most common losses for regression.
- Easy to optimise directly.
- $\text{RMSE} = \sqrt{\text{MSE}}$.
- Exercise: What is the best constant prediction?

MAE

- Another common loss.
- Less sensitive to outliers.
- Also could be optimised directly (although not by second order methods).
- Exercise: What is the best constant prediction?

R^2

$$R^2 = 1 - \frac{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

- Could be viewed as a more interpretable variant of MSE.

R^2

$$R^2 = 1 - \frac{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

- Could be viewed as a more interpretable variant of MSE.
- Exercise: What is the range of R^2?

R^2

$$R^2 = 1 - \frac{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

- Could be viewed as a more interpretable variant of MSE.
- Exercise: What is the range of R^2?

Validation

Validation – why do we care?

Simple Question

Why do we need a validation set?

Validation – why do we care?

Simple Question

Why do we need a validation set?

Even Simpler Question

Why do we need a test set?

Validation – why do we care?

Simple Question

Why do we need a validation set?

Even Simpler Question

Why do we need a test set?

- We need to estimate how our model will generalise to unseen data. In real life we expect to use the model in some way after we build it. We need to model the conditions of model deployment in train/test split.

Validation – why do we care?

Simple Question

Why do we need a validation set?

Even Simpler Question

Why do we need a test set?

- We need to estimate how our model will generalise to unseen data. In real life we expect to use the model in some way after we build it. We need to model the conditions of model deployment in train/test split.
- Some say this is the single most important step in the whole process in real-life. If you fail everything else, but you have done this in the right way, you'll know that you failed. Otherwise you might not know it until production.

Validation – why do we care?

Simple Question

Why do we need a validation set?

Even Simpler Question

Why do we need a test set?

- We need to estimate how our model will generalise to unseen data. In real life we expect to use the model in some way after we build it. We need to model the conditions of model deployment in train/test split.
- Some say this is the single most important step in the whole process in real-life. If you fail everything else, but you have done this in the right way, you'll know that you failed. Otherwise you might not know it until production.
- Similarly, we need to mimic the train/test split when we make the train/validation split. In competitions, train/test split is done for us and we need to spend time studying it.

Validation: strategies

- Random split
    - Traditional default option.
    - Assumes data is i.i.d.

Validation: strategies

- Random split
  - Traditional default option.
  - Assumes data is i.i.d.
- Time-based split
  - An obvious choice for time series data.
  - You don't want your model to learnt to predict the past.
- Id-based split
  - If you want to generalise to the unseen ids, you should not put the same id in train and test sets (and thus train/validation)
  - Sometimes in the competitions you need to learn the ids to make the right split (fisheries example).

Data Leaks

Data Leaks

- Exploiting a data leak is something very specific to competitions.

Data Leaks

- Exploiting a data leak is something very specific to competitions.
- But preventing them is something very real-life.

Data Leaks

- Exploiting a data leak is something very specific to competitions.
- But preventing them is something very real-life.
- Roughly speaking, leakage occurs when the information about the test set is somehow included in the training set, either because of the features included or because of the train/test split.

Data Leaks

- Exploiting a data leak is something very specific to competitions.
- But preventing them is something very real-life.
- Roughly speaking, leakage occurs when the information about the test set is somehow included in the training set, either because of the features included or because of the train/test split.
- Crude example:

patient_id	took_drug_x	had_disease
1	1	1
2	0	0
3	0	1
4	1	1

- More subtle examples: have a look on Kaggle, it's fun!

Ensembling

Ensebmling

- Kaggle competitions are rarely won by a single model.
- Rather a number of models are aggregated using various ensembling techniques.
- Common techniques:
  - Bagging (that's how RF work) – more on that later
  - Boosting (that's how GBMs and Adaboost work) – more on that later
  - Stacking