Date Science Summary

Zhiying Wu

June 12, 2018

Abstract

About kaggle: Competitive data science, ML, very best solutions,best predictive capabilities as few mistakes as possible. competing for the prize, participants push through the limits, come up with novel ideas,unique opportunity to learn.

(Here is an operation,run it on all of the data)

Contents

1 Overview

- Exporatory data analysis
- Basic and advanced feature generation and preprocessing
- Various model validation techniques
- Data leakages
- Competition's metric optimization
- Model ensembling
- Hyperparameter tuning

1.1 Points

1. point1
   - feature processing & extraction
2. point2
   - intuition about the data
   - reliable validation
   - data leaks explored
3. point3
   - analyze various metrics for regression and classification
   - optimize them both while training and afterwards
   - mean-encodings
   - work on encoded features
     - categorical features
     - numeric feature
     - time series
4. point4
   - Advanced features
     - statistics and distance-based features
     - metrics factorizations
     - feature interactions
     - t-SNE
   - Hyperparameter optim
   - Ensembles
     - simple linear ensemble
     - bagging,boosting
     - stacking and stacked net approach
5. point5
   - analyze some of winning solutions in competitions
   - Final project

   Data
   - svv files with several columns
   - a text file
   - an archive with pictures
   - a database dump
   - a disabled code
   - or even all together

   Model
   - produce best possible prediction
   - be reproducible
   - something that transforms data into answers

   Submission,Evaluation
   - Accuracy,Logistic Loss,AUC,RMSE,MAE
   - It is what we will try to optimize

2 Feature preprocessing and generation

Numeric features,categorical and ordinal features, Datatime and coordinates, Handing missing values.
why we should care about different features having different types?

- strong connection between preprocessing at our model–often necessary
- common feature generation methods for each feature type–powerful tech

2.1 Preprocessing

(Tree-based model,Non-tree based models)

Scaling:MinMaxScaler,StandardScaler

- $[(x - x_{min})/(x_{max} - x_{min})]$,To[0,1].sklearn.preprocessing.MinMaxScaler
- $[(x - x_{mean})/x_{std}]$,To mean=0,std=1. sklearn.preprocessing.StandardScaler
- $np.log(1 + x)$ Log transform
- $np.sqrt(x + 2/3)$ Raising to the power less than 1

Outliers

Rank

- rank([-100,0,100])=[0,1,2]
- rank([1000,1,10])=[2,0,1]

2.2 Feature generation

Help us making model training more simple and effective. engineer these features using prior knowledge and logic. dig into the data create and check hypothesis(EDA) and use this derived knowledge and our intuition to derive new features.

- Categorical and ordinal features
  - Label encoding
  - Frequency encoding
  - One-hot encoding
- Datetime and coordinates
  - periodicity
  - Time since
  - Difference between dates
- Missing value
  - -999,1 ...
  - mean,median
  - Reconstract value

2.3 Feature extraction

- Bag of words sklearn.feature_extraction.text.CountVectorizer
  - TFiDF. Tern Frequency,Inverse Document Frequency (sklearn.feature_extraction.text.TfidVectorizer)
  - N_grams (sklearn.feature_extraction.text.CountVectorizer:Ngram_range.analyzer)
  - Stopwords (sklearn.featuree_extraction.text.CountVectorzer:Max-df)
- Embeddings (word2vec)
- CNN

3 EDA(Exploratory data analysis),Validation,Data leakages

3.1 EDA

- Get domain knowledge
- check it the data is intuitive
- understand how the data was generated
- Exploring anonymized data
  - Try to decode the features
  - Guess the feature types
    - $df.dtypes.$
      - most likely to be numeric
      - integer -binary(1/0).Event counters or Event categorical
      - object type can be anything even an irregular numeric feature
      - with missing values filled with some text
    - $df.info()$
    - $X.value\_counts()$
    - $X.isnull()$
- Visualizations (EDA is an art, and visualizations are our art tools)
  - Explore individual features
    - $plt.hist(x)$
    - $plt.plot(x,'.')$
    - $plt.scatter(range(len(x),x,c=y)$(c means *color by label*)
    - $df.describe()$
    - $X.mean, X.var, X.value\_counts, X.isnull()$
  - Explore feature relations
    - $plt.scatter(x1,x2)$
    - $plt.scattermatrix(df)$
    - $df.corr(), plt.matshow()$if the metrics looks like a total mess, we can run some kind of clustering like k-means and reorder the features
    - $df.mean().plot(style:'.')$
    - $df.mean().sort\_values().plot(style:'.')$
    - it's good idea to generate new features based on the groups
- Dateset cleaning
  - duplicated and constant feature
    - $train.nunique(axis=1)==1$
    - $traintest.T.drop\_duplicates()$
      - $for f in categorical\_feats:$
        $traintest[f]:traintest[f].factorize()$
      - $traintest.T.drop\_duplicates$
- other things to check

  (check if dataset is shuffled,if it's not, then there is a high chance to find **data leakage**)
- EDA example
- Reverse Engineering
- *detective story*

3.2 Validation and overfitting

- Holdout,kFold,Loo
- Stratification.
  - $smalldatasets$
  - $unbalanceddatasets$
  - $multiclassification$

  (preserve the same target distribution over different folds,make validation more stable)

- Data Splitting Strategies $\begin{cases} Row\ number \\ id \\ time \end{cases}$

  (logic of feature generation depend on the data splitting strategies)

- Submission stage
  (different score in KFold,others will...)

  1. too little data in public leader_board
  2. train and test data are different
  3. incorrect train/test split

- Expect LB shuffle because of
  - Randomness(unpredictable)
  - Little amount of data
  - Different public(private distribution)(**usually the case which time series prediction**)

3.3 Data Leakages

As we unexpected information in the data that allows us to make unrealistically good predictions

- leaks in time series
  - split should be done on time
  - feature may contain information about future
- unexpected information
  - meta data
  - information in IDs
  - Row order

  (`connected to target variable`)

- Leaderboard probing
  - Type of LB probing
  - categories tightly connected with 'id' are vulnerable to LB probing

4 Metric Optimization,Mean encodings

(Loss vs metric. Review the most important metric. Optimization tech for metric)
Metric:*(it's how the competitors are ranked!)*

- **Regression**
  - MSE,RMSE,Rsquared **(just fit the right model)**
  - MAE(more robust to outliers) **(just fit the right model)**
  - (R)MSPE,MAPE **(can be optimized by either resampling the dataset or setting proper sample weights)**
  - (R)MSLE **(is optimized by MSE in log space.like Train:**$Z_i = \log(y_i + 1)$**,Test:**$\hat{y}_i = e^{\hat{z}_i} - 1$**)**

    $$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, MAE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i), MSPE = \frac{100\%}{N}\sum_{i=1}^{N}(\frac{y_i - \hat{y}_i}{y_i})^2, MAPE = \frac{100\%}{N}\sum_{i=1}^{N}(\frac{y_i - \hat{y}_i}{y_i})$$

    (MSPE is weight version of MSE),(MAPE is weight version of MAE)

    $$RMSLE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\log(y_i + 1) - \log(\hat{y}_i + 1))^2} = \sqrt{MSE(\log(y_i + 1), \log(\hat{y}_i + 1))}$$

(RMSLE is MSE in log space)

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

- **Classification**
  - Accuracy,Logloss,AUC
  - Cohen's(Quadratic weighted kappa)

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

  (fit any metric and tune threshold)

  Logloss=$-\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)$

  (just run the right model**(or calibrate others)**)

  AUC(Area Under Curve)= $1 - \frac{\text{incorrect pairs}}{\text{total pairs}}$

  (just run the right model**(or just optimize Logloss)**)

  Cohen's kappa $= 1 - \frac{error}{baselineerror}$ (dataset:10 cats, 90 dogs)(predicted 20 cats, at random 80dogs) $accuracy = 0.2 \times 0.1 + 0.8 \times 0.9 = 0.74$ error=0.26

  weight kappa $= 1 - \frac{weight\ error = \frac{1}{const}\sum_{i=j} C_{ij}Wij}{weighted\ baseline}$,In many cases, the weight matrics are defined in a very simple way

  (**optimize MSE and find right threshold(simple)**,or (custom smooth loss for GBDT or neural nets(harder)).)

  - confusion matrix C

$$\begin{array}{c} \text{T/P} \\ cat \\ dog \\ tiger \end{array} \begin{array}{c} \begin{array}{ccc} cat & dog & tiger \end{array} \\ \begin{pmatrix} 4 & 2 & 3 \\ 2 & 88 & 4 \\ 4 & 10 & 12 \end{pmatrix} \end{array}$$

  - weight matrix W

$$\begin{array}{c} \text{T/P} \\ cat \\ dog \\ tiger \end{array} \begin{array}{c} \begin{array}{ccc} cat & dog & tiger \end{array} \\ \begin{pmatrix} 0 & 1 & 10 \\ 1 & 0 & 10 \\ 1 & 1 & 0 \end{pmatrix} \end{array}$$

  - linear weights $W_{ij} = |i - j|$)

$$\begin{array}{c} \text{T/P} \\ cat \\ dog \\ tiger \end{array} \begin{array}{c} \begin{array}{ccc} cat & dog & tiger \end{array} \\ \begin{pmatrix} 1 & 2 & 3 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix} \end{array}$$

  - Quadratic weights $W_{ij} = (i - j)^2$

$$\begin{array}{c} \text{T/P} \\ cat \\ dog \\ tiger \end{array} \begin{array}{c} \begin{array}{ccc} cat & dog & tiger \end{array} \\ \begin{pmatrix} 0 & 1 & 4 \\ 1 & 0 & 1 \\ 4 & 1 & 0 \end{pmatrix} \end{array}$$

  The Quadratic weighted kappa has been used in several competitions on kaggle, it's usually explained as inter-rate agreement coefficient. How much the predictions of the model agree with ground truth,raters(like how much agrees with professional doctors)

- Loss and metric
  - target metric is what we want to optimize
  - optimization loss is what model optimizes
- Approaches for target metric optimization
  - Just run the right model **(MSE,Logloss)**
  - preprocess train and optimize another metric **(MSPE,MAPE,RMSLE)**
  - optimize another metric.postprocess predictions **(Accuracy,kappa)**
  - write custom loss function **(any,if you can,for example:Quadratic-weight kappa)**
  - optimize another metric.**(use early stopping,which we always can use)**

- Mean encoding
  - likelihood = $\frac{goods}{goods+bads} = mean(target)$
  - weight of Guidence = $\log \frac{goods}{bads} \times 100$
  - count = goods = sum(target)
  - oiff = goods-bads

5 Hyperparameter tuning

XGBoost $\begin{cases} max\_depth \\ subsample \\ colsample_bytree/bylevel \\ min\_child\_weight(\lambda, \alpha) \\ \eta, num\_round, others \\ seeds \end{cases}$ Light GBW $\begin{cases} max\_depth/num\_leaves \\ bagging\_fraction \text{ (0-1)(kind of regularization)} \\ - \\ min\_data\_in\_leaf(\lambda_{11}, \lambda_{12}) \text{ (0,5,15,300)(kind of regular)} \\ learning\_rate, num\_iterations, others \text{ (0.1,0.01...)} \\ *\_seeds \text{ (if } num\_round \times 2\text{: eta/2)} \end{cases}$

(**max_depth/num_leaves**)
(sometimes 2,sometimes 27,if (depth+,no overfitting): a sign that lots of interactions to extract from data/to stop tuning and generate some features.recommend to start with **7**)
(**min_data_in_leaf**)
(it set 0,the mode will be less constrained. the higher,more regularized(freedom))

- Random Forest $\begin{cases} N\_estimators \text{ (the higher the better)} \\ max\_depth \text{ (start with 7,10,20...)} \\ max\_features \\ min\_samples\_leaf \text{ (same as min_child_weight)} \\ others... \\ criterion \text{ (Gini(better\&more often),Entropy)} \\ n\_jobs \text{ (default 1)} \end{cases}$

- Neural Nets $\begin{cases} \text{number of neurons per layer} \\ \text{number of layers} \\ optimizers \begin{cases} SGD + momentum(great) \\ \text{Adam/Adadelta/Adagrad/... (in practice lead to more overfitting)} \end{cases} \\ batchsize \text{ } (32, 64) \\ \text{learning\_ratae (0.1,0.01...)(follow batch size)} \\ regularization \begin{cases} \text{L2/L1 for weights} \\ \text{Dropout/Dropconnect} \\ \text{Static dropconnect} \end{cases} \end{cases}$

- sklearn
  - SVC/SVR (sklearn wraps libLinear and libSVM)
  - Logistic Regression/Linear Regression+regularizations
  - SGDClassifier/SGDRegressor
- Vowpal Wabbit
  - reads data row-by-row directly from hand dray allowing to learn on a very huge datasets
- Regularization parameters(c,α, λ)
  - (SVM($10^{-6}, 10^{-5}...$) start to work slower as c increases)
- Regularization type
  - L1/L2/L1+L2... try each
  - L1(weight sparsity)(the sparsity pattern can be used for feature selection)

- **average**

(You cannot win a competition by tuning parameters.Appropriate features,hack leaks,and insights will give you much more than carefully tuned model built on default features)

6 Tip and Tricks

- one
  - data loading
  - start with fastest models-lightGBW
  - use early stopping
  - Even medium-size datasets like 50,000 or 100,000 rows, just simple train/test split.
  - switch to tuning the models and sampling and stacking only when satisfied with feature engineering
  - fast,dirty...always better.focus on what's realy important the data.Do EDA,try different features,Google domain-specific knowledge.Code is secondary,keep things simple and reasonable.
  - only a couple of notebooks for model training and notebooks for EDA purposes
- two
  - from simple to complex(start with Random Forest rather than Gradient Boosted Decision Trees)
  - From reading data to writing submission file(initial pipeline)
  - keep research reproducible
    - fix random seed
    - write down exactly how any features were generated
    - use version control systems(VCS,for example:git)
    - (situation:go back to model,edit to the ensemble width)
  - Reuseable(into separate functions or even separate model)
    - (expicially important to use same code for train/text stage)
  - Read papers
    - can get ideas about ML-related things.(for example:how to optimize AUC)
    - way to get familiar with problem domain(especially useful for feature generation)
- pipeline
  - Read forums and examine kernels first
    - there are always discussions happending
    - it happends that a competition starts someone finds a bug in the data
  - Start with EDA and a baseline(various leakages)
    - to make sure the data is loaded correctly
    - to check if validation is stable(if correlates with publicly leader board score)
  - Add feature in bulks
    - create all the feature that can make up
    - evaluate many features at once(not 'add on and evaluate')
  - hyperparameters optimization
    - first find the parameters to overfit train dataset
    - and then try to train model
  - Keep it clean
    - very important to have reproducible results(careful about code and script)
  - One notebook per submission (and use git)
  - Custom library!
    - use a library with frequent operations
    - out-of-fold predictions
    - averaging
    - specify a classifier by it's name
    - ...

7 Ensembling

- Averaging(or blending)
- Weight Averaging
- Conditional Averaging
- bagging
  - changing the seed
  - Rwo(sub)sampling or Boostrapping
  - shuffling
  - column(sub)sampling
  - Model-specific
  - Number of models(or bags)
  - (optionally) parallelism the seem
- Boosting
  - weighted based
  - residual based
- stacking (stacking can actually extract the juice from each prediction)
- StackNet

8 What was left

- Advanced Features 2
- mean encoding
  - statistic and distance based features
  - matrix factorizations
  - feature interactions
  - t-SNE
  - PCA