Coursera: How to Win a Data Science Competition: Learn from Top Kagglers

DJ's note

2018

Contents

1 Week 1

1.1 Recap of main ML algorithms

1.1.1 Random Forests

Random forest is a collection of many decision trees. Each decision tree is trained on a subset of variables and observations from your dataset.

Each tree cuts the data based on your predictors into groups that are as similar as possible based on your target variable. As each tree is trained on different variables and data, the trees can vary considerably.

So, when we want to predict our target given our inputs, we ask our decision trees and take the average prediction (or majority vote for classification) given those values.

Are you seeing the forest from the trees? When is a RF a poor choice relative to other algorithms? What does a Random Forest model look like? What's happening inside the black box?

1.1.2 H2O

H20 is the open-source *in-memory* prediction engine for Big Data Science.

`http://manishbarnwal.com/blog/2017/03/28/h2o_with_r/`

`https://github.com/h2oai/h2o-3`

1.2 Overview of methods

Scikit-Learn (or sklearn) library

Overview of k-NN (sklearn's documentation)

Overview of Linear Models (sklearn's documentation)

Overview of Decision Trees (sklearn's documentation)

Overview of algorithms and parameters in H2O documentation

Additional Tools

Vowpal Wabbit repository

XGBoost repository

LightGBM repository

Interactive demo of simple feed-forward Neural Net

Frameworks for Neural Nets: Keras,PyTorch,TensorFlow,MXNet, Lasagne

Example from sklearn with different decision surfaces

Arbitrary order factorization machines

1.3 Software/Hardware requirements

1.4 Feature processing and generation with respect to models

1.4.1 Different features

- Numeric
- Ordinal: ordered categorical feature (ticket classes, driver license, and education
- Text
- ID
- Categorical

1.4.2 Feature processing and generation pipelines depend on a model type

e.g. One Hot encoders is useful, however, Random Forrest does not need this

1.4.3 Numeric feature preprocessing

Outliers

Rank

If no time to handle outliers in linear and NN models

scipy.stats.rankdata

Note: to apply rank to the test data, you need to

- Store the creative mapping from feature values to their rank
- Cacatenate train and test data before applying rank transformation

Numeric feature

1. Numeric feature processing is different for tree and non-tree models
    a. Tree-based models don't depend on scaling
    b. Non-tree based models hugely depend on scaling
2. Most often used preprocessings are
    a. MinMaxScaler [0, 1]
    b. StandardScaler to mean ==0, std==1
    c. Rank sets spaces between sorted values

d np.log(1+x) and np.sqrt(1+x)

3. Feature generations is powered by:

   a Prior knowledge

   b Exploratory data analysis

1.4.4 Categorical and ordinal features

1. **Ordinal**

   a Ordered categorical (Ticket class, education level, and driver's license etc)

   b Ordinal vs Numerical: The differences between numerical classes are the same. It's not the case for Ordinal classes, e.g. Ticket Class

   c Label encoding: maps categories to numbers (For tree-based models)
   Alphabetical (sorted: sklearn.preprocessing.LabelEncoder)
   Order of appearance: pandas.factorize

   d Frequency encoding: maps categories to their frequencies (For tree-based models usually)
   pandas.factorize

2. **Categorical**

   a One-hot encoding (For non-tree based models)
   sklearn.preprocessing.OneHotEncoder
   Allows non-tree based models (KNN, linear models) to take categorical features and improve models

   b

1.4.5 Date and coordinates

1. **Date and Time**

   a Periodicity
   Capture repetitive patterns

   b Time since a particular event
   Row-independent moment (e.g. Date stamp)
   Row-dependent important moment (e.g. Number of days left until next holidays)

c Difference between dates

2. **Coordinates**

   a Known Coordinates

   Rental Prices

   Rotate coordinates may improve models

   b Unkown Coordinates

   Extracting interesting places from train/test data or additional data

   Canters of clusters

   Adding aggregated statistics (e.g. Divide the map by squares, find the most expensive flat and calculate distance of the nearest shops etc)

1.4.6 Missing data

1. Missing values can be hidden (replaced by e.g. NaNs)
2. The choice of method to fill NaN depends on the situation (e.g. check histogram). Usual ways to deal with NaN is to replace them with

   out of range value, e.g. -999

   mean

   median
3. Binary feature "isnull" can be beneficial
4. In general, filling NaNs after feature generation
5. XGBboost can handle NaN

1.4.7 Further reading and resource

1. **Feature preprocessing**

   a Preprocessing in Sklearn `https://scikit-learn.org/stable/modules/preprocessing.html`

   b Andrew NG about gradient descent and feature scaling `https://www.coursera.org/learn/machine-learning/lecture/xx3Da/gradient-descent-in-practice-i-feature-scalin`

   c Feature scaling and the effect of standardisation for machine learning algorithms `http://sebastianraschka.com/Articles/2014_about_feature_scaling.html`

d Andrew Ng's Machine Learning Course in Python (Linear Regression) https://medium.com/@ben_lau93/andrew-ngs-machine-learning-course-in-python-linear-regressi

2. **Feature generation**

a Discover Feature Engineering, How to Engineer Features and How to Get Good at It https://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-fe

b Discussion of feature engineering on Quora https://www.quora.com/What-are-some-best-practi

1.5 Feature extraction from text and images

1.5.1 Text

1. Preprocession: Lowercase, stemming, lemmarization, stopwords

(Stemming and lemmarization: to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form.)

2. Bag of Words:

Huge vectors: guarantees you clear interpretation. Each feature is turned by means of having a huge amount of features. One for each unique word.

Ngrams: can be applied to include words interactions for text

TFiDF: can be use of post-processing

3. Word2Vec:

Small vectors: can be applied to include words interactions for text

Pre-trained models

Resource - Something to vec https://gist.github.com/nzw0301/333afc00bd508501268fa7bf40cafe4e

PyText for faster NLP development https://code.fb.com/ai-research/pytext-open-source-nlp-fram

1.5.2 Images

New images integration is effective.

Fine tuning Augmentation is used to increase the number of images for the purpose of fine tuning.

1.6 Software/Hardware requirements

1.7 Week 1 Questions

1. In general case, should we apply these transformations to all numeric features, when we train a non-tree-based model?

   A Yes, we should apply a chosen transformation to all numeric features

   B No, we should apply it only to few numeric features, leaving other numeric features as they were

2. Can frequency encoding be of help for non-tree based models?

   A Yes, it can

   B No, it can't

2 Week 2

2.1 Exploratory data analysis (EDA) is an art

2.1.1 Get domain knowledge

2.1.2 Check if the data is intuitive

1. **Does the data agree with domain knowledge?**
2. **Understand how the data was generated**
    a Visualisation
    b Generate a new column (e.g. Relative days = Date-min(Date))
    c You could correct the data manually (Age more than 200 years)
    d You could add some rows in the train data to make train set similar to the test set. (e.g. `https://jljdbvgniaoatlyhacdqym.coursera-apps.org/notebooks/readonly/reading_materials/EDA_video2.ipynb`

2.1.3 Exploring anonymized data

1. Data which are sensitive and changed by organisers (on purpose)
2. Data which looks like hash value or with no meaningful column names

2.1.4 Explore individual features

1. Try to decode the column meanings by checking value_counts and histogram distribution
   e.g. values could be shifted by adding a number
2. Guess the types of the columns
    a df.dtypes()
    b df.info()
    c x.value_counts()
    d x.isnull()

2.1.5 Feature Visualisation

1. **Check value distribution**
    a Histogram: plt.hist(x)

i. Can be misleading (Can compare histograms using values or logarithm of the values)

ii. Accumulate values in bins

iii. If you see a high peak in the histogram (e.g. means). It could be the organisers filled missing values using mean values. How can we use this information? We can replace the missing values we found with not numbers, nulls again. For example, xgboost has a special algorithm that can fill missing values on its own and so, maybe xgboost will benefit from explicit missing values.

b Index vs Value plot Use row index as x axis, feature value as y axis plt.plot(x,'.'). This can help understand whether data is shuffled and whether there are a lot of repetitive values.

c Scatter plot: Color code y values, check whether the column is a class feature; plt.scatter(range(len(x)),x, c=y)

d Statistics

2. **Don't make a hypothesis based on a single plot**

2.1.6 Explore feature relations

1. **Find relations between pairs.**

   a Scatter plot plt.scatter(x1,x2). If the a part of the test data is overlapped with training data, this is bad.

   b Corr plot

2. **Find feature groups.**

   a Corr plot + clustering Diagonal (x1+x2=1). For tree based model, we could generate new feature, x1/x2 ratio or x1 and x2 difference pd.scatter_matrix(df) df.corr(), plt.matshow()

   b Plot(Index vs feature statistics) df.mean().sort_values().plot(style='')

2.1.7 Dataset cleaning and other things to check

1. **Dataset cleaning**

   a **Constant features** The train and test data set could be just a fraction of original data (e.g. only the data in 2018 was selected)

If **traintest**.nunique(axis=1) ==1, then the feature could be deleted from the training set

If **train**.nunique(axis=1) ==1, the feature should also be deleted from the training set

b **Duplicated features** one of the duplicated columns with different column names should be removed traintest.T.drop_duplicates()

The feature are identical, but feature level are different

Use **label encoding** on the two features from the **top to the bottom** row (the 1st appeared is 0, 2nd is 2)

i. for f in categorical_features: traintest[f]=traintest[f].factorize()

ii. traintest.T.drop_duplicates()

2. **Other things to check**

   a Duplicated rows in training set could be removed

   b Check if data is shuffled

2.1.8 Further reading and resource: Visualization tools

1. **Seaborn** `https://seaborn.pydata.org/`
2. **Plotly** `https://plot.ly/python/`
3. **Bokeh** `https://github.com/bokeh/bokeh`
4. **ggplot** `http://ggplot.yhathq.com/`
5. **Graph visualization with NetworkX** `https://networkx.github.io/`
6. **Biclustering algorithms for sorting corrplots** `http://scikit-learn.org/stable/auto_examples/bicluster/plot_spectral_biclustering.html`

2.2 Validation

2.2.1 Validation and overfitting

1. **Why do we need validation?**

   a Validation helps us to evaluate the quality of the model

   b Validation helps us to select the model which will perform best on unseen data

2. **What is underfitting?**

a Underfitting refers to not capturing enough patterns in the data

3. **What is overfitting?**

   a Overfitting refers to capturing noise

   b Capturing patterns which do not generalise to test data

   c In competition, overfitting refers to low model's quality on test data, which was unexpected due to validation scores

2.2.2 Validation strategies

Main validation strategies (schemes): holdout, K-Fold, LOO

The main rule you should know — never use data you train on to measure the quality of your model. The trick is to split all your data into training and validation parts.

Below you will find several ways to validate a model.

1. **Holdout scheme:**

   Split train data into two parts: partA and partB. Fit the model on partA, predict for partB. Use predictions for partB for estimating model quality. Find such hyper-parameters, that quality on partB is maximized.

2. **K-Fold scheme:**

   Split train data into K folds. Iterate though each fold: retrain the model on all folds except current fold, predict for the current fold. Use the predictions to calculate quality on each fold. Find such hyper-parameters, that quality on each fold is maximized. You can also estimate mean and variance of the loss. This is very helpful in order to understand significance of improvement.

3. **LOO (Leave-One-Out) scheme:**

   Iterate over samples: retrain the model on all samples except current sample, predict for the current sample. You will need to retrain the model N times (if N is the number of samples in the dataset). In the end you will get LOO predictions for every sample in the trainset and can calculate loss.

Notice, that these are validation schemes are supposed to be used to estimate quality of the model. When you found the right hyper-parameters and want to get test predictions don't forget to retrain your model using all training data.

2.2.3 Common validation problems

1. If we have big dispersion of scores on validation stage, we should do extensive validation Below you will find several ways to validate a model.
    a Average scores from different KFold splits
    b Tune model on one split, evaluate score on the other
2. If submission's score do not match local validation score, we should
    a Check if we have too little data in public LB
    b Check if we overfitted
    c Check if we chose correct splitting strategy
    d Check if train/test have different distibutions
3. Expect LB shuffle because of
    a Randomness
    b Little amount of data
    c Different public/private distributions

2.3 Data Leakage

How can data be leaked? Peculiar examples

2.4 Week 2 Questions

1. Suppose we are given a huge dataset. We did a KFold validation once and noticed that scores on each fold are roughly the same. Which validation type is most practical to use?
    A We can use a simple holdout validation scheme because the data is homogeneous.
    B We should keep on using KFold scheme as the data is homogeneous and KFold is the most computationally efficient scheme.
    C Leave-one-out because the data is not homogeneous.
2. Suppose we are given a medium-sized dataset and we did a KFold validation once. We noticed that scores on each fold differ noticeably. Which validation type is the most practical to use?
    A Holdout

B LOO

C KFold

3. The features we generate depend on the train-test data splitting method. Is this true?

A Tue

B False

4. What of these can indicate an expected leaderboard shuffle in a competition?

A Most of the competitors have very similar scores

B Little amount of training or/and testing data

C Different public/private data or target distributions

3 Week 3

3.1 Metrics optimization

3.1.1 Motivation

3.1.2 Regression metrics

1. **Regression**
    a **MSE: Mean Square Error**
    b **RMSE: Squared Mean Square Error**
    c **RMSE: Squared Mean Square Error**
    d **MSE vs MAE**
        i. Do you have outliers in the data? Use MAE
        ii. Are you sure they are outliers? Use MAE
        iii. Or they are just unexpected values we should still care about? Use MSE
2. **Discussed the metrics, sensitive to relative errors:**
    a **MSPE** Weighted version of MSE; Root MSPE
    b **MAPE** Weighted version of MAE
    c **MSLE** Log scaled version of MSE, Root MSLE

3.2 Advanced features I

1. **Mean encoding**
2. **Regularization**
    a CV loop inside training data
        i. Robust and intuitive
        ii. Usually decent results with 4-5 folds across different datasets
        iii. Perfect feature for LOO scheme
        iv. Target variable leakage is still present even for KFold scheme
    b Smoothing
        i. Alpha controls the amount of regularization
        ii. Only works together with some other regularization method

```
y_tr = df_tr['target'].values #target variable
skf = StratifiedKFold(y_tr,5, shuffle=True,random_state=123)

for tr_ind, val_ind in skf:
    X_tr, X_val = df_tr.iloc[tr_ind], df_tr.iloc[val_ind]
    for col in cols: #iterate though the columns we want to encode
        means = X_val[col].map(X_tr.groupby(col).target.mean())
        X_val[col+'_mean_target'] = means
    train_new.iloc[val_ind] = X_val

prior = df_tr['target'].mean() #global mean
train_new.fillna(prior,inplace=True) #fill NANs with global mean
```

Figure 1: Regiarization.CV loop

c Adding random noise

i. Noise degrades the quality of encoding

ii. How much noise should we add?

iii. Usually used together with LOO

d Sorting and calculating expanding mean

i. Least amount of leakage

ii. No hyper parameters

iii. Irregular encoding quality

iv. Built - in in CatBoost

3. **Extensions and generalizations**

3.3 Metrics optimization

4 Week 4

4.1 Hyperparameter tuning

4.2 Advanced features II

4.2.1 Matrix factorization

Matrix factorization: MF, or matrix decomposition. MF is a way of reducing a matrix into its constituent parts. It can be applied for transforming categorical features into real-valued.

1. **Standard MF**
    a SVD
    b PCA
    X_all = np.concatenate ([X_train, X_test])
    pca.fit (X_all)
    X_train_pca = pca.transform(X_train)
    X_test_pca = pca.transform(X_test)
2. **TruncatedSVD**
    a For sparse matrices
    b `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html`
3. **Non-negative Matrix Factorization (NMF)**
    a Ensures that all latent factors are non-negative
    b Good for counts-like data

4.2.2 Feature Interaction

1. **Frequent operations for feature interaction**
    a Multiplication
    b Sum
    c Diff
    d Division
2. **Extract high-order interactions from decision trees**

a Map each leaf into a binary feature (yes or no)

b sklearn: tree_model.apply()

c xgboost: booster.predict(pred_leaf=True)

4.2.3 t-SNE

t-SNE :Project points from high-dimensional space into small dimensional space, so the distances between points are approximately preserved

1. **Try different perplexities** 5-100

2. **How to use t-SNE effectively?**
   Martin Wattenberg, `https://distill.pub/2016/misread-tsne/`

3. **Tools**

   a tsne 0.1.8 python (fast)

   b t-SNE R (single-cell)

   c sklearn :sklearn.manifold.TSNE

4. **Note**

   a tSNE is a great tool for visualization

   b It can be used as feature as well

   c Be careful with interpretation of results

   d Try different perplexitie

5 Week 5

5.1 Kaggle past solutions

1. http://ndres.me/kaggle-past-solutions
2. http://www.chioka.in/kaggle-competition-solutions
3. https://github.com/ShuaiW/kaggle-classification

5.2 A lot of cheetsheets

https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning