PriceMinister Documentation

Release 0.1

cristianpb

Jan 08, 2018

Contents

PriceMinister is one of the leading marketplace of E-commerce in France, and is part of the Rakuten group, leader in Japanese market. The goal of this challenge is to predict if a review of a user on a given product may be useful for others. Results from this challenge will help us to rank and filter the millions of reviews on PriceMinister, in order to increase the quality of user experience. This is a binary classification problem, evaluated trough AUC metric. Data are in French.

CHAPTER 1

Project Organization

```
├── LICENSE
├── Makefile           <- Makefile with commands like `make data` or `make train`
├── README.md          <- The top-level README for developers using this project.
├── data
│   ├── external       <- Data from third party sources.
│   ├── interim        <- Intermediate data that has been transformed.
│   ├── processed      <- The final, canonical data sets for modeling.
│   └── raw            <- The original, immutable data dump.
│
├── docs               <- A default Sphinx project; see sphinx-doc.org for details
│
├── models             <- Trained and serialized models, model predictions, or model␣
↪summaries
│
├── notebooks          <- Jupyter notebooks. Naming convention is a number (for␣
↪ordering),
│                         the creator's initials, and a short `-` delimited␣
↪description, e.g.
│                         `1.0-jqp-initial-data-exploration`.
│
├── references         <- Data dictionaries, manuals, and all other explanatory␣
↪materials.
│
├── reports            <- Generated analysis as HTML, PDF, LaTeX, etc.
│   └── figures        <- Generated graphics and figures to be used in reporting
│
├── requirements.txt   <- The requirements file for reproducing the analysis␣
↪environment, e.g.
│                         generated with `pip freeze > requirements.txt`
│
├── src                <- Source code for use in this project.
│   ├── __init__.py    <- Makes src a Python module
│   │
│   ├── data           <- Scripts to download or generate data
│   │   └── make_dataset.py
```

```
│   │
│   ├── features       <- Scripts to turn raw data into features for modeling
│   │   └── build_features.py
│   │
│   ├── models         <- Scripts to train models and then use trained models to make
│   │   │                 predictions
│   │   ├── predict_model.py
│   │   └── train_model.py
│   │
│   └── visualization  <- Scripts to create exploratory and results oriented␣
↪visualizations
│       └── visualize.py
│
└── tox.ini            <- tox file with settings for running tox; see tox.testrun.org
```

Project based on the Cookiecutter data science project template.

CHAPTER 2

Contents

2.1 Context

Contexte du challenge

PriceMinister permet à ses utilisateurs de partager leur avis sur les produits qu'ils ont achetés. Avec plus d'un million d'avis, ce jeu de données a un énorme potentiel pour améliorer la qualité du site et permettre aux utilisateurs de trouver leur produit qui leur convient le mieux. Chaque utilisateur peut aussi évaluer un avis, en spécifiant si il est utile pour lui ou non. Ce retour est aussi très important pour la qualité des services. Ce jeu de données est original car il contient des avis sur des contenus textuels et non pas sur les produits eux même. Prédire le retour des utilisateurs sur les avis eux même (retour qui est subjectif et personnel), est particulièrement ambitieux, et peut avoir des implications plus général que le e-commerce Ce challenge est issu de la collaboration entre l'equipe Big Data Europe team de Rakuten (Search, Recommendations, Targeting, . . .) (https://global.rakuten.com/corp/careers/bigdata/) et le Rakuten Institute of Technology (Computer Vision, Human Computer Interface, Machine Learning, Deep Learning, . . .) (http://rit.rakuten.co.jp)

Objectifs du challenge

Le but est d'évaluer si un avis utilisateur sur un produit peut être utile pour d'autres utilisateurs. Certains avis sont ainsi partic
très grandes et belles cartes soyeuses, les illustrations sont extrêmement bien réalisées, le manuel d'explications est facile et très lisible, pour la voyance personnelle ou pour tirer les cartes en cérémonial la grandeur des cartes permet d'organiser un vrai tirage professionnel)

et d'autres moins car plus personnels, e.g." JEU TRES BIEN POUR MA FILLE DE 4 ANS.JE RECOMMANDE POUR LES JEUNES ENFANT;TRES SATISFAIT).

Chaque avis est labélisé comme useful (class 1) ou not useful (class 0), en se basant sur leur nombre de retours useful si (nombre de retours positifs / total du nombre de retours) > 0.5 not useful si (nombre de retours positifs / total du nombre de retours) < 0.5

Nous avons enlevé les avis pour lesquels le nombre de retour positifs et le nombre de retours négatifs étaient égaux. Nous utiliserons la mesure AUC (Area Under the Curve) pour l'évaluation, et vous devrez fournir la probabilité d'être utile (class 1) pour chaque avis du jeu de données de test. Le format attendu est un CSV (séparateur ;) avec les champs ID et Probabilité, e.g: ID;Target 39;1 40;1 . . . 156;0

Ne PAS oublier d'ajouter le header (ID;Target) car s'il n'est pas présent, la soumission ne sera pas évaluée.

2.2 Read data

Here are some examples to get you started.

src.data.read_data.**get_stopwords**()
Get french stopwords

src.data.read_data.**read_data**(*test=False*)
Read train and test data

2.3 Features

Here are some examples to get you started.

src.features.build_features.**feature_extraction**(*dataset*, *stopwords*)
Main function to do all feature engineering

src.features.build_features.**get_fasttext**()
Load fasttext french pretrained model

https://fasttext.cc/docs/en/pretrained-vectors.html

src.features.build_features.**get_vec**(*text*, *model*, *stopwords*)
Transform text pandas series in array with the vector representation of the sentence using fasttext model

src.features.build_features.**replace_na**(*dataset*, *labels*)
Fill NaN with 'na'

src.features.build_features.**sent2vec**(*s*, *model*, *stopwords*)
Transform a sentence into a vector using fasttext representation

src.features.build_features.**stack_sparse**(*components*)
Stack sparse vectors horizontally [X_1, X_2, ..]

src.features.build_features.**to_categorical**(*dataset*, *label*)
Transform variable to categorical using one hot encoding

src.features.build_features.**to_sparse_int**(*dataset*, *label*)
Transform to intiger encoding and in sparse from

src.features.build_features.**to_tfidf**(*dataset*, *label*, *stopwords*)
Term frequency–inverse document frequency reflect how important a word is to a document in a collection or corpus

Parameters **ngram_range** – tuple containing the range of ngram sizes to use.

2.4 Models

Here are some examples to get you started.

src.models.train_model.**model_ensembler**(*X_train_tfv*, *X_train_ft*, *y_train*)
Train ensemble model

src.models.train_model.**model_lightgbm**(*X_train*, *y_train*)
Light Gradient Boosting Machine

https://github.com/Microsoft/LightGBM/blob/master/docs/Features.rst

src.models.train_model.**model_ridge**(*X_train*, *y_train*)
Ridge regression Minimizing the residual sum of squares we also have a penalty on the coefficients

http://scikit-learn.org/stable/modules/linear_model.html#ridge-regression

src.models.train_model.**model_xgb**(*X_train*, *y_train*)
Extreme gradient boosting

http://xgboost.readthedocs.io/en/latest/

src.models.train_model.**score_function**(*y_true*, *y_pred*)
Score function auc

src.models.train_model.**split_train**(*X*, *y*, *test_size*, *random_state=7*)
Train and validation split

2.5 Predict functions

src.models.predict_model.**predict_test**(*submission*, *preds1*)
Create submission file from predictions

2.6 Vis

Here are some examples to get you started.

src.visualization.visualize.**plot_roc**(*fpr*, *tpr*)
Plot roc curve

src.visualization.visualize.**plot_scatter**(*values*, *colors*, *ticks*, *ticks_labels*)
Plot 2D representation of sentences

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

s

Index