ENGIE Data Science Challenge

« Automating IT Operations Using Machine Learning »

Sept – Nov 2016

Ajinkya CHANDRAYAN
Data Scientist - Département Solutions Globales
INEO DIGITAL

Content

Preprocessing

- Label Encoding

It is used to transform non-numerical labels to numerical labels.

Language	Python
Library	sklearn.preprocessing; LabelEncoder
Variables	ini, codeClosing,libjob,consumer,ArDailyStatConsumer, VALDailyStatConsumer, VARDailyStatConsumer
Sample script	fullData['ini'] = number.fit_transform(fullData['ini'].astype('str'))

- Treatment of missing values in variables

The data description states that the Resource Plan before January 2015 was not available, which means that the 'slot' field will have no data for the jobs before January 2015. This missing data has been treated by filling in the **mean** values.

- Replacing NaN with 0

Feature Creation / Processing (1/2)

- Create «year», «month», «day», «week»,«day of the week»,«hour»,«minute» from the date variables.

Language	**Python**
Library	**Pandas**
Variables	**datdeb, dateCalcul, datdealversion**
Sample script	**dt = pd.to_datetime(fullData.datdeb).dt fullData["datdeb_Year"] = dt.year**

- Create «milliseconds» from date variables.

Language	**Python**
Library	**Pandas, Numpy**
Variables	**dateCalcul, tradeDate, datcrever, Datmodver**
Sample script	**df1=pd.to_datetime(fullData['dateCalcul']) fullData["dateCalculMS"]= df1.astype(np.int64) // 10**9**

Feature Creation / Processing (2/2)

- Creating «dealtype_count» and «fin_count»

The deals dataset contains the number of deals traded each day by «dealtype». The data description highlights that the number of deals traded each day can be helpful for the model precision.

Language	**Python**
Library	**Pandas, Numpy**
Variables	**dealtype, count**
Sample script	**deals = deals.groupby(['tradeDate','dealtype']).mean().squeeze().unstack().add_suffix('_count') df1 = deals.replace(np.nan,0, regex=True) deals['fin_count'] = deals.apply(lambda row: row['CHC_count'] + row['CPT_count']+ row['CSH_count'] + row['CSHCO_count'] + row['EXFLEX_count']+ row['EXOSCP_count'] + row['EXSCP_count'] + row['FUTCO_count'] + row['GEFWD_count'] + row['GEFWI_count'] + row['GEOPT_count'] + row['GESWA_count'] + row['GETRA_count'] + row['OCH_count'] + row['OPA_count'] + row['OPC_count'] + row['OPTMO_count'] + row['PECUR_count'] + row['STOK_count'] + row['SWA_count'] + row['SWF_count'] + row['SWT_count'] + row['TER_count'] + row['TSC_count'], axis=1)**

Models (1/2)

- This challenge was an opportunity to experiment on some new libraries and approaches.
- In the initial phase I experimented on the H2O library which gave very good results.
- The H2O library is rich and can allow training the models within the available resources.
- **H2O Library**

Language	**R Programming**
Library	**h2o**
Approach	**Ensemble learning**
Algorithms	**GLM, Random Forest, GBM**
Highest Score	**13.6542046128**

Models (2/2)

- In the final stages of the challenge, I moved on to the xgboost library, which is quite popular amongst the kagglers and a personal favourite that almost, always ensures good result.
- I experimented xgboost library with the bag of models approach which after some rounds of parameters tuning gave the highest score.
- **Library XGBOOST**

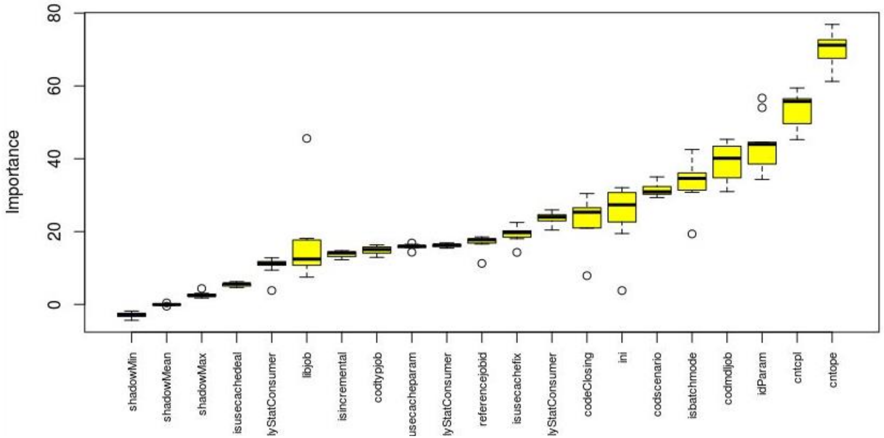
Language	**R Programming**
Library	**Xgboost**
Approach	**Bag of models**
Algorithms	**Gradient tree boosting**
Highest Score	**12.3187479036**

Features Importance

- For feature importance, I experimented on the Boruta algorithm during this challenge.
- Boruta is a feature selection algorithm. Precisely, it works as a wrapper algorithm around Random Forest.

Language	**R Programming**
Library	**Boruta**

	meanImp	medianImp	minImp	maxImp	normHits	decision
codtypjob	14.934670	15.078228	12.911324	16.337305	1.0	Confirmed
ini	25.138846	27.357467	3.808829	32.057937	1.0	Confirmed
cntope	70.250920	71.197926	61.231584	76.894304	1.0	Confirmed
cntcpl	53.932561	55.743292	45.257485	59.435008	1.0	Confirmed
codmdljob	39.074359	40.142600	30.969667	45.325027	1.0	Confirmed
libjob	16.087546	12.438285	7.493510	45.588435	1.0	Confirmed
isincremental	13.883126	14.118249	12.265184	14.762873	1.0	Confirmed
referencejobid	16.995208	17.736344	11.268129	18.536686	1.0	Confirmed
isbatchmode	33.812611	34.597223	19.365210	42.557718	1.0	Confirmed
idParam	43.750994	43.968532	34.310573	56.690496	1.0	Confirmed
codeClosing	23.521509	25.338465	7.900929	30.448458	1.0	Confirmed
codscenario	31.388452	30.911212	29.334952	35.012042	1.0	Confirmed
isusecachedeal	5.465505	5.663818	4.642970	6.251165	1.0	Confirmed
isusecacheparam	15.899516	16.019584	14.354455	16.883566	1.0	Confirmed
isusecachefix	19.310960	19.732971	14.296433	22.516132	1.0	Confirmed
ArDailyStatConsumer	23.617142	23.969306	20.431533	25.953979	1.0	Confirmed
VALDailyStatConsumer	10.531338	11.296081	3.814953	12.788028	0.9	Confirmed
VARDailyStatConsumer	16.233508	16.315605	15.440343	16.870240	1.0	Confirmed



Final Blend

- The final model was trained by tuning the hyper parameters after performing different experiments and submissions.

Language	R Programming
Library	Xgboost
Approach	Bag of models
Algorithms	Gradient tree boosting
Total number of models	50
Final Calculation	Mean of all models
Parameters	shrinkage(eta), rounds, depth, gamma, min.child, colsample.bytree, subsample
Features	all

Tools and Frameworks

Language	Python, R Programming, SQL
Tools	Jupyter Notebook, R Studio, DB Browser for SQLite
Librairies (Python)	Pandas, Numpy, ScikitLearn, csv
Librairies (R Programming)	dplyr,data.table,lubridate,ggplot2,sqldf,xgboost,h2o,boruta

Source Code

- All code : Python + R Programming + SQL

All_Code.txt

- Final datasets (train, test) used for model training & submission :

https://github.com/ajinkyachandrayan/Data-Science-Challenge-1/blob/master/train_GEM_1_x.zip

https://github.com/ajinkyachandrayan/Data-Science-Challenge-1/blob/master/test_GEM_1_x.zip