Big Data Project

Medicare Fraud Detection Using Open Source Data

Xinyu (Max) Liu
xinyulrsm@gmail.com

Project link:
http://nbviewer.jupyter.org/github/maxliu/health_care/blob/master/HealthCare_fraud_detection.ipynb

Project overview

- Goal: Build a predictive model for detecting health care fraud
- Data source
  - Part-D and Payment data: **cms.gov**
  - NPI exclusion data: **hhs.gov**
- Technology
  - **Hadoop/Yarn/Spark/Hive**: Part_D and Payment data are stored in a five-node cluster.
  - **Pig**: Pig scripts were written for cleaning data and merging tables.
  - **Python/SciKit-learn**: Implemented a data processing pipeline including scaling and modeling. *t*-test is used to select drugs on training set for feature extraction.
- Algorithm and result
  - **Five different models**: Logistic regression, Gaussian, Random Forest Classifier, Extra Trees Classifier, Gradient Boosting Classifier
  - Average AUC: 0.69 (Random Forest)

Fraudulence in health care industry

Data source: **en.Wikipedia.org/wiki/Medicare_fraud**

Year	People Charged	False Billings (Million)
2010	94	$251
2011	91	$295
2012	2	$1.9
2013	89	$223
2014		
2015	243	$712

Date storage: 5-node Hadoop/yarn/spark cluster

Copy files to Hadoop then move to Hive warehouse

Copy data to hadoop

```
%%bash
$HADOOP_HOME/bin/hdfs dfs -mkdir -p /data/
$HADOOP_HOME/bin/hdfs dfs -put /home/max/data/PARTD_PRESCRIBER_PUF_NPI_DRUG_13.tab    /data
$HADOOP_HOME/bin/hdfs dfs -put /home/max/data/OP_DTL_GNRL_PGYR2013_P01152016.csv      /data
```

Hadoop Overview Datanodes Snapshot Startup Progress Utilities

Browse Directory

/user/hive/warehouse/payment_13 Go!

Permission	Owner	Group	Size	Replication	Block Size	Name
-rwxrwxr-x	hduser	supergroup	1.98 GB	3	128 MB	OP_DTL_GNRL_PGYR2013_P01152016.csv

Permission	Owner	Group	Size	Replication	Block Size	Name
-rwxrwxr-x	hduser	supergroup	2.65 GB	3	128 MB	PARTD_PRESCRIBER_PUF_NPI_DRUG_13.tab

Pig scripts for cleaning data and merging tables

```
%%writefile dataPrep.pig

Register '/usr/local/pig/lib/piggybank.jar';

partd_raw = load '/data/HealthCare/PARTD_PRESCRIBER_PUF_NPI_DRUG_13.tab'
    using org.apache.pig.piggybank.storage.CSVExcelStorage('\t', 'YES_MULTILINE', 'NOCHANGE', 'SKIP_INPUT_HEADER')
    as (

        NPI:    int,
        NPPES_PROVIDER_LAST_ORG_NAME: chararray,
        NPPES_PROVIDER_FIRST_NAME:  chararray,
        NPPES_PROVIDER_CITY:    chararray,
...
        TOTAL_DRUG_COST_GE65:   float
        ) ;


npi_drugs = group partd_raw by NPI;

npi_drugs_table = foreach npi_drugs {
    specilty  = limit partd_raw.SPECIALTY_DESC 1;
    lastname  = limit partd_raw.NPPES_PROVIDER_LAST_ORG_NAME 1;
    firstname = limit partd_raw.NPPES_PROVIDER_FIRST_NAME 1 ;
    city      = limit partd_raw.NPPES_PROVIDER_CITY 1;
    state     = limit partd_raw.NPPES_PROVIDER_STATE 1 ;

    generate
    group as gp,

    COUNT(partd_raw),
...

    SUM(partd_raw.TOTAL_DRUG_COST) ;
};
```

```
df_partD.head()
```

	npi	count	specialty	claim_min	claim_max	claim_sum	supply_min	supply_max	supply_sum	drug_min	drug_max	drug_sum	p
0	1528364486	2	Dentist	14	19	33	176	305	481	97.54	162.96	260.500008	N
1	1437177490	5	Dentist	14	89	188	46	2955	4726	106.12	4346.47	5468.750191	N
2	1154586170	2	Dentist	12	17	29	141	170	311	37.09	119.95	157.039997	N
3	1215042155	9	General Surgery	17	87	522	98	410	2107	84.80	377.74	2120.089966	N
4	1104847011	1	Dentist	11	11	11	78	78	78	59.26	59.26	59.259998	N

Group part_D data by NPI and drug name for future analysis

```
drugs = group partd_raw by (NPI, GENERIC_NAME);

npi_drug_table_2 = foreach drugs  {
        specilty = limit partd_raw.SPECIALTY_DESC 1;
        lastname =     limit partd_raw.NPPES_PROVIDER_LAST_ORG_NAME 1;
        firstname = limit partd_raw.NPPES_PROVIDER_FIRST_NAME 1 ;
        city = limit partd_raw.NPPES_PROVIDER_CITY 1;
        state = limit partd_raw.NPPES_PROVIDER_STATE 1 ;
            generate
            flatten(group),
            COUNT(partd_raw),
            SUM(partd_raw.TOTAL_CLAIM_COUNT),
            SUM(partd_raw.TOTAL_DAY_SUPPLY),
            SUM(partd_raw.TOTAL_DRUG_COST) ;
            };
rmf /data/to_hadoop/partd_13_npi_drug_all.csv
store npi_drug_table_2 into '/data/to_hadoop/partd_13_npi_drug_all.csv' using PigStorage('\t') ;
```

```
npi_drug.head()
```

	npi	drug	count	total_claim_count	total_day_supply	total_drug_cost
0	1003000126	LISINOPRIL	1	1.301030	2.756636	2.005909
1	1003000126	SIMVASTATIN	1	1.255273	2.688420	2.013090
2	1003000126	WARFARIN SODIUM	1	1.079181	2.511883	2.221284
3	1003000142	BACLOFEN	1	1.204120	2.654177	2.148633
4	1003000142	MELOXICAM	1	1.505150	2.924796	2.275749

Group Payment data by NPI for future analysis

```
payment_raw = load '/data/HealthCare/OP_DTL_GNRL_PGYR2013_P01152016.csv'
    using org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'YES_MULTILINE', 'NOCHANGE', 'SKIP_INPUT_HEADER')
    as (
        Covered_Recipient_Type:chararray,
        Teaching_Hospital_ID:chararray,
...
        Name_of_Associated_Covered_Device_or_Medical_Supply4:chararray,
        Name_of_Associated_Covered_Device_or_Medical_Supply5:chararray,
        Program_Year:chararray,
        Payment_Publication_Date:chararray
        ) ;


npi_payment = group payment_raw by (Physician_First_Name,Physician_Last_Name,
                                    Recipient_City, Recipient_State);
npi_payment_table = foreach npi_payment {
    generate
    /*flatten(group),*/
    UPPER(group.Physician_First_Name) as first_name,
    UPPER(group.Physician_Last_Name) as last_name,
    UPPER(group.Recipient_City) as city,
    UPPER(group.Recipient_State) as state
```

```
df.head()
```

ty	claim_min	claim_max	claim_sum	supply_min	supply_max	supply_sum	drug_min	drug_max	drug_sum	payment_count	total_payment
	14	19	33	176	305	481	97.54	162.96	260.500008	0	0
	14	89	188	46	2955	4726	106.12	4346.47	5468.750191	0	0
	12	17	29	141	170	311	37.09	119.95	157.039997	0	0
	17	87	522	98	410	2107	84.80	377.74	2120.089966	0	0
	11	11	11	78	78	78	59.26	59.26	59.259998	0	0

NPI exclusions database used as target

Merged exclusion data to the table as “is_fraud” column

```
df.head()
```

min	claim_max	claim_sum	supply_min	supply_max	supply_sum	drug_min	drug_max	drug_sum	payment_count	total_payment	is_fraud
	19	33	176	305	481	97.54	162.96	260.500008	0	0	0
	89	188	46	2955	4726	106.12	4346.47	5468.750191	0	0	0
	17	29	141	170	311	37.09	119.95	157.039997	0	0	0
	87	522	98	410	2107	84.80	377.74	2120.089966	0	0	0
	11	11	78	78	78	59.26	59.26	59.259998	0	0	0

The data is highly unbalanced.

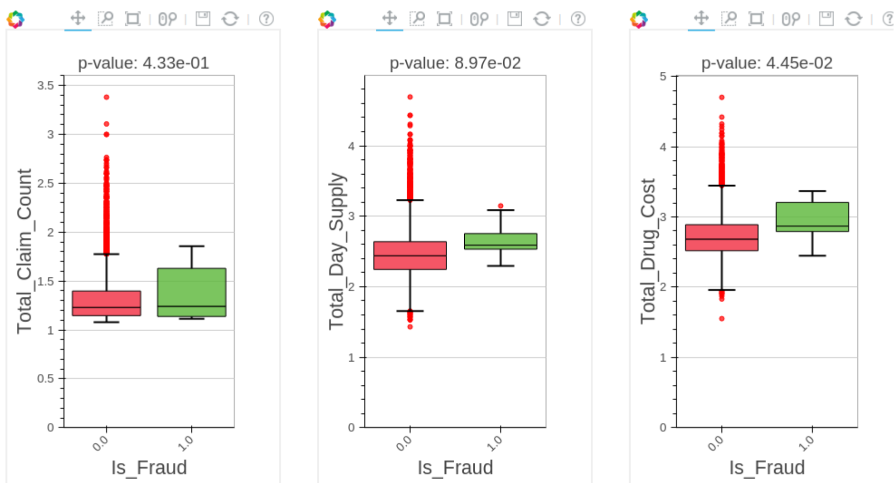
- Only 427 positive samples in total 808076 records (0.05%).
- Possible solutions
  - Down sampling
  - Up sampling
  - Give higher weight topositive samples for model training (used in this project)

t-test: Select the drug feature only when P <= 0.05

An example for "CLOTRIMAZOLE"

No No YES

CLOTRIMAZOLE



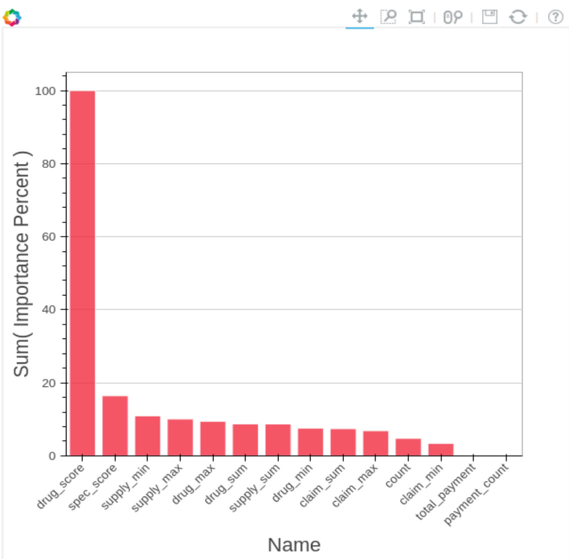
Note that this feature selection was only applied on training set to avoid data leakage.

Five different models used:

Logistic regression, Gaussian, Random Forest Classifier, Extra Trees Classifier, Gradient Boosting Classifier

```
params_0 = {'n_estimators': 100, 'max_depth': 8, 'min_samples_split': 1, 'learning_rate': 0.01}
params_1 = {'n_estimators': 500, 'max_depth': 10, 'min_samples_split': 1, 'class_weight' : {0:1, 1:4000}, 'n_jobs':3}

scaler = StandardScaler()

clfs = [
    LogisticRegression(C=1e5,class_weight={0:1, 1:4000}, n_jobs=3),

    GaussianNB(),

    ensemble.RandomForestClassifier(**params_1),

    ensemble.ExtraTreesClassifier(**params_1),

    ensemble.GradientBoostingClassifier(**params_0)

    ]
```

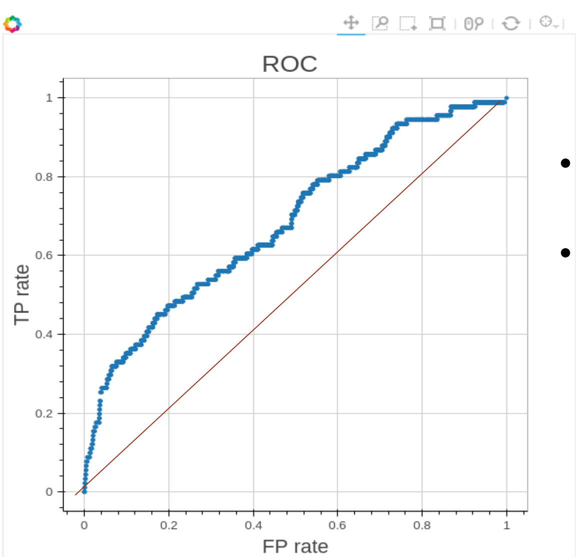
Feature importance analysis (from Random Forest)



Findings

- The “drug_score” feature created become the most important feature
- The payment turned out to be trivial or negligible.

AUC (from Random Forest)



- Average AUC ~ 0.685 (> 0.5)
- This means the predictive model I built can provide useful information on health care fraud detection.

Future work for improvement

- Include more data (e.g., the Part-B dataset)
- Add additional feature (e.g., Page rank)
- Blend multiple model

References

- https://www.versustexas.com/criminal/healthcare-medicare-medicaid-fraud/
  PageRank for Anomaly Detection,
  By Ofer Mendelevitch and Jiwon Seo
- http://www.dataiku.com/blog/2015/08/12/Medicare_Fraud.html
  By Pierre Gutierrez @ Dataiku