Asssignment: Backorders prediction model

Author: Nguyen Minh Chuong

Phone: 0934783385

Date: 31/3/2018

Data analysis:

In this step I will demonstrate some works to explore the raw data:

The used language is Python with its strong support libraries for data science. My IDE(Intergrated Development Environment) is Pycharm which has user friendly interface built from QT. It is also quite easier to use than Python Shell.

I used 3 main libs to understand this data: pandas, numpy and matplotlib:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
data = pd.read_csv('C:\Users\minhchuo\PycharmProjects\datascience\ca.csv')
```

As first, I map data from ca.csv (this is training_set)

This is a classification machine learing model. The reason for that is we have to define test_sample into 2 classes namely: went_to_backorder and not_went_to_backorder. Because we have only 2 classes, we can call this binary classification model.

I took out 5 samples from data and get data information:

```
print data.head(5).transpose()
data.info()
```

There are: 23 features of 1687860 samples.

I excluded *sku* and *went_on_backorder* since they are not the important features.

There are 15 quantative features and 6 Yes/No features

Hencem, I separated them into 2 manin groups:

```
quantfeature = ['national_inv',
                'lead_time',
                'in_transit_qty',
                'forecast_3_month',
                'forecast_6_month',
                'forecast_9_month',
                'sales_1_month',
                'sales_3_month',
                'sales_6_month',
                'sales_9_month',
                'min_bank',
                'pieces_past_due',
                'perf_6_month_avg',
                'perf_12_month_avg',
                'local_bo_qty']
```

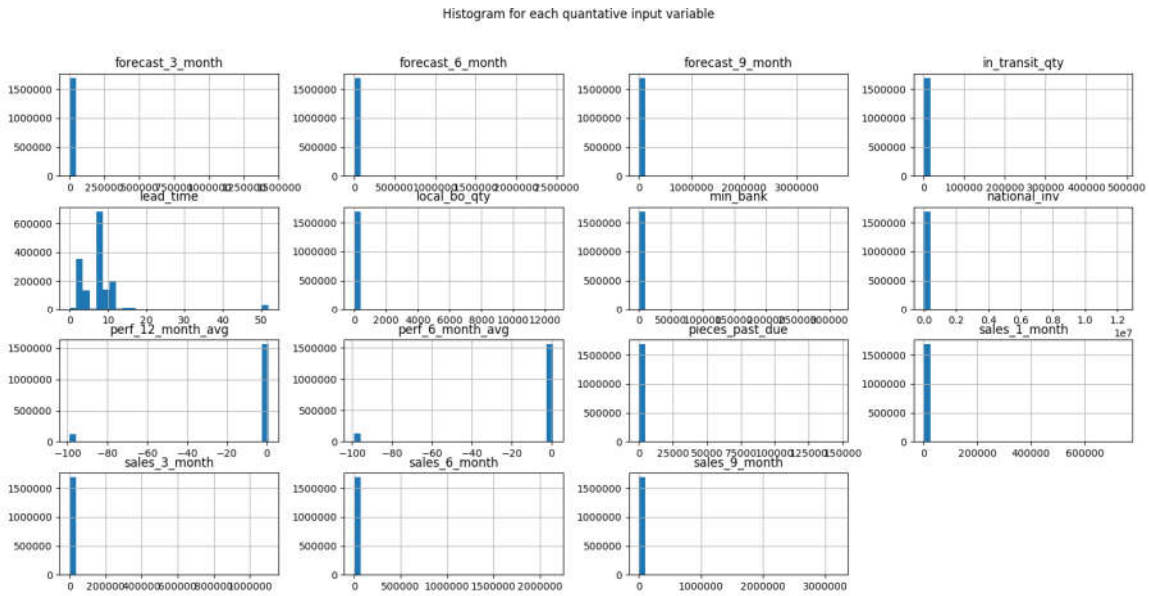
And

```
binfeature = ['potential_issue',
              'deck_risk',
              'oe_constraint',
              'ppap_risk',
              'stop_auto_buy',
              'rev_stop',
              'went_on_backorder']

pred_binfeature = ['potential_issue',
              'deck_risk',
              'oe_constraint',
              'ppap_risk',
              'stop_auto_buy',
              'rev_stop']
```

Let's take a look at quantative values by using histogram of pylab:

```
import pylab as pl
data.drop('sku' ,axis=1).hist(bins=30, figsize=(9,9))
pl.suptitle("Histogram for each quantative input variable")
plt.savefig('histogram')
plt.show()
```



The range of each feature is quite large but values tend to stay near 0 for most of them. We could detect that performance_average for 6 and 12 months are not able to be negative values. I considered them as noises.

Then, I digged into binary features:

```
for col in binfeature:
    data[col] = pd.factorize(data[col])[0]
```

Converted Yes/No to 1/0 values.

Calculated % Yes(1) in total:

```
for col in binfeature:
    print(col, ':', round(data[col].mean()*100, 2),'%')
```

The answer is:

('potential_issue', ':', 0.05, '%')

('deck_risk', ':', 22.96, '%')

('oe_constraint', ':', 0.01, '%')

('ppap_risk', ':', 12.08, '%')

('stop_auto_buy', ':', 3.62, '%')

('rev_stop', ':', 0.04, '%')

('went_on_backorder', ':', 0.67, '%')

As we can see this training set is unbalance. Most of products did not go on backorder.

To conclude, with this type of large, noised, unbalance, missing, many features data set, I choose random forest algorithm to handle the case.

Model building

At first, I treated noise by replacing with a Numpy NaN.

I removed these -99 values

```
data['perf_6_month_avg'] = data['perf_6_month_avg'].replace(-99, np.NaN)
data['perf_12_month_avg'] = data['perf_12_month_avg'].replace(-99, np.NaN)
```

At this stage, I already have 3 columns which have missing values.

I give them the median. Mean is fine either.

```
data = data.fillna(data.median(), inplace=True)
```

I created X_train y_train for random forest model:

```
y_train = data['went_on_backorder']
X_train = data.drop(['sku', 'went_on_backorder'], axis=1)
```

Then, I created a random forest model:

```
rf = RandomForestClassifier(n_estimators=50,max_features=3)
rf.fit(X_train,y_train)
y_test = rf.predict(X_test)
print y_test
```

However, this took a lot of time to run. I decided to rescale the samples to be more balance.

```
big_train = data[y_train == 0]
small_train = data[y_train == 1]
len_small_train = len(small_train)
big_train_lower_sampled = resample(big_train,
```

```
                                        replace=False,
                                        n_samples=len_small_train,
                                        random_state=123)
data_lower_sampled = pd.concat([big_train_lower_sampled, small_train])
y_train_new = data_lower_sampled['went_on_backorder']
X_train_new = data_lower_sampled.drop(['sku', 'went_on_backorder'], axis=1)
```

We all know that the majority of products went to *not backorder* category. I used resample function from sklearn to down scale the 0-result to balance with 1-result in column went_on_backorder

Then the answer is [1,0] as the same with answer from given exercise (1,-1)

```
data_test =
pd.read_csv('C:\Users\minhchuo\PycharmProjects\datascience\data_test_sample.csv')
for col in pred_binfeature:
    data_test[col] = pd.factorize(data_test[col])[0]
X_test = data_test.drop(['sku'], axis=1)

rf = RandomForestClassifier(n_estimators=50,max_features=3)
rf.fit(X_train_new,y_train_new)
y_test = rf.predict(X_test)
print y_test
```

Output []:

```
C:\Users\minhchuo\AppData\Local\Continuum\An
[1 0]

Process finished with exit code 0
```