

Rare event detection

Github repo : <https://github.com/b04901056/Qualcomm-interview>

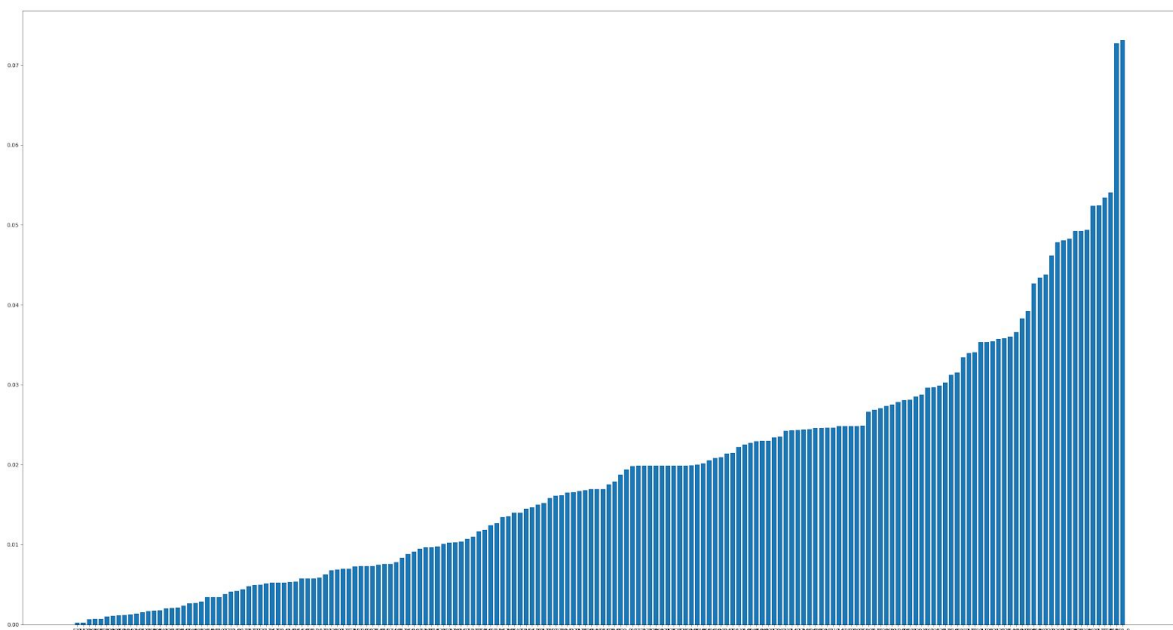
Data Analysis Summary & Cleanup procedure

- (1) Data in attribute #5 in training set and testing set are not consistent
=> Delete (It cannot help forecast label of attribute #4)
- (2) Attribute #6 corresponds to the boolean value of attribute #4
=> Delete (otherwise, the problem will be meaningless, we can forecast the result by simply transform boolean expression to target category)
- (3) Standard deviation of attribute #166,167,168,170,171,172,177 in training set are zero
=> Delete (It cannot help forecast label of attribute #4)
- (4) Some testing samples have missing data or missing target (attribute #4) label
=> For missing data, fill in with the mean of this attribute from the training set;
For missing target, fill in according to the boolean value of attribute #6;
If both attribute #4 and #6 are missing, then give up this sample.

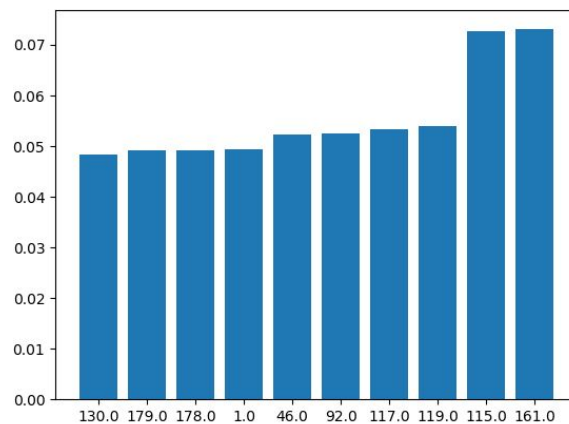
Inputation

Sample #3 in training set have missing attribute #7, I used regression method to predict its value and fill it in.

The two figures below show every variable's correlation coefficient (absolute value) with attribute #4 and 10 features that are most correlated to attribute #4.



```
-- python .\correlation.py .\DataMining_IV_x2\DataSet_IV_Training.csv .\DataMining_IV_x2\DataSet_IV_Test.csv
```



Attribute #117,163 have higher correlation coefficient with attribute #4 compared with other variables.

I extracted 10 features that are most correlated to attribute #4 as the inputs of my Deep neural network (DNN) and attribute #4 as target, and use other samples in training set without missing data as training data to optimize my model for forecasting the value of missing data.

```
Train Epoch: 151 | [2048/2972 (67%)] Total loss: 0.4431
predicted result: [74763.10391301]
```

```
-- python .\inputation_train.py -train .\DataMining_IV_x2\DataSet_IV_Training.csv -u 10 16 32 16 2 1
```

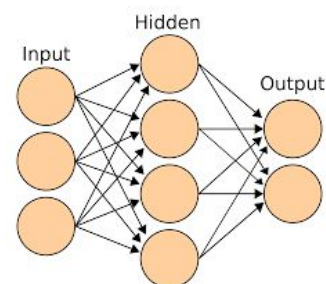
Transformation

- (1) Some attributes are in the form of TRUE/FALSE or category
=> Use one hot encoding to transform these attributes into binary representation
- (2) Normalize each attribute, and record its mean and standard deviation for forecasting
=> $x' = (x - \text{mean}(x)) / \text{std}(x)$

Model proposals & Results

- (1) Deep neural network (DNN)

```
DNN(
  (den): ModuleList(
    (0): Sequential(
      (0): Linear(in_features=170, out_features=32, bias=True)
      (1): ReLU()
      (2): Dropout(p=0.2)
    )
    (1): Sequential(
      (0): Linear(in_features=32, out_features=1, bias=True)
      (1): Dropout(p=0.2)
      (2): Sigmoid()
    )
  )
)
```



(<http://inspirehep.net/record/1507419/plots>)

```
--python .\train.py -train .\DataMining_IV_x2\DataSet_IV_Training.csv -test .\DataMining_IV_x2\DataSet_IV_Test.csv -u 178 32 1
```

Optimizer : Adam (momentum & adaptive learning rate)

Loss function : Binary cross entropy

Activation function : Relu (sparsity and avoiding gradient vanishing)

I used pytorch to implement a DNN with one hidden layer, and introduce dropout layer to avoid overfitting. The sigmoid function can bound the output of the model in $[-1, 1]$ for computation of binary cross entropy loss.

To make my model cost-sensitive, I adjust the weight of BCE loss function. Set higher weight for positive samples, lower one for negative samples.

Result on testing set (confusion matrix):

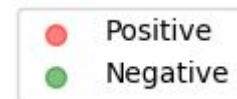
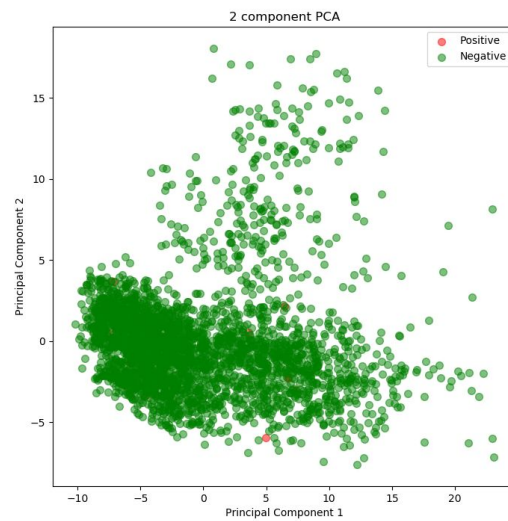
		Truth	
		Positive	Negative
Prediction	Positive	0	35
	Negative	7	1063

My model tends to recognize every sample as negative, because it put too much emphasis on negative samples during training.

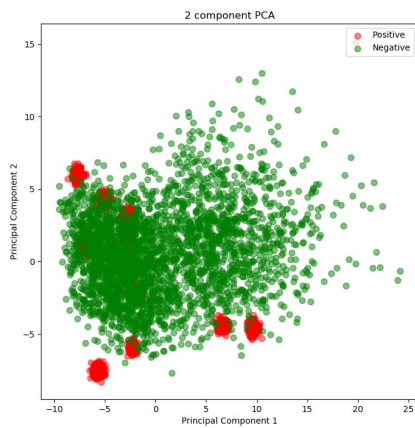
(2) DNN + over/down sampling

Due to imbalanced dataset, DNN model cannot learn how to detect rare event. So I tried oversampling the positive samples by simply copying them and adding small noise (avoid overfitting).

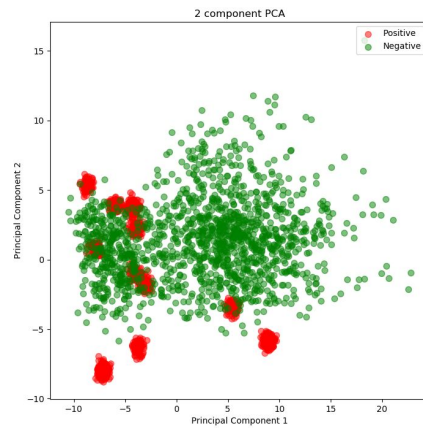
Similarly, I tried downsampling by deleting some negative samples to balance the training set. The figures below demonstrate the effect of over/down sampling (use PCA to map the training data to a two-dimensional plane and plot it with matplotlib).



original training data



oversampling



over & down sampling

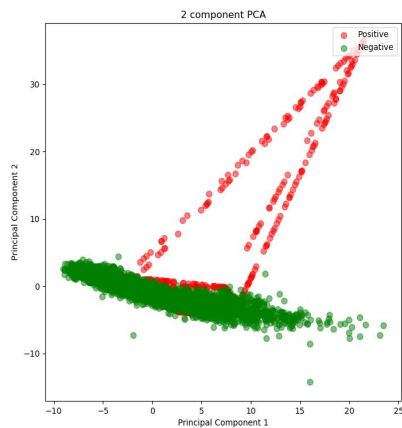
Result on testing set (confusion matrix):

		Truth	
		Positive	Negative
Prediction	Positive	0	94
	Negative	7	1004

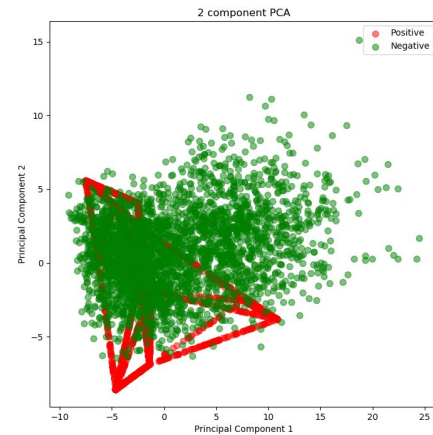
Although my model can detect rare events on validation set, it performs poorly on testing set. I conclude that over/down sampling are not enough to tackle the problem due to overfitting.

(3) DNN + Generated synthetic samples

I found an algorithm called SMOTE on the internet, which introduces randomness in generating the synthetic data instead of copies. I hope that it can help resolve the problem of overfitting.



SMOTE



SMOTE, remove outliers

Result on validation set (confusion matrix):

		Truth	
		Positive	Negative
Prediction	Positive	3	106
	Negative	0	189

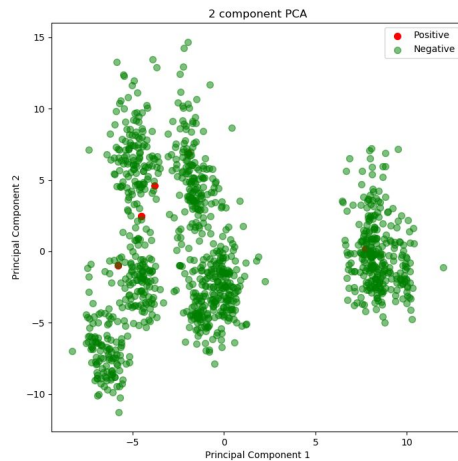
Result on testing set (confusion matrix):

		Truth	
		Positive	Negative
Prediction	Positive	0	10
	Negative	7	1088

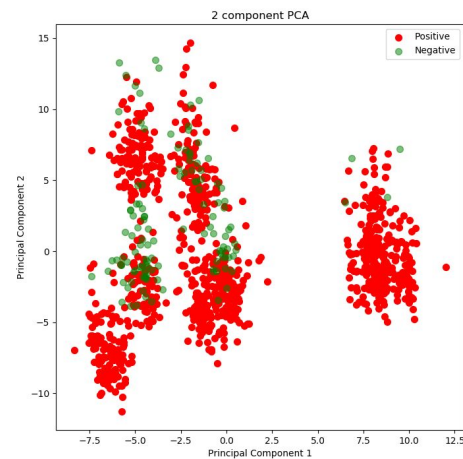
My model performs little bit better on validation set than testing set, so I guess that training set and testing set have different data distribution. I tried adding Dropout layers to prevent overfitting but didn't see much improvement.

SMOTE algorithm still cannot effectively resolve this problem, so I started to consider some classification approaches other than neural-based methods.

(4) Support vector machine (SVM) + Generated synthetic samples



Ground truth



Classification using SVM

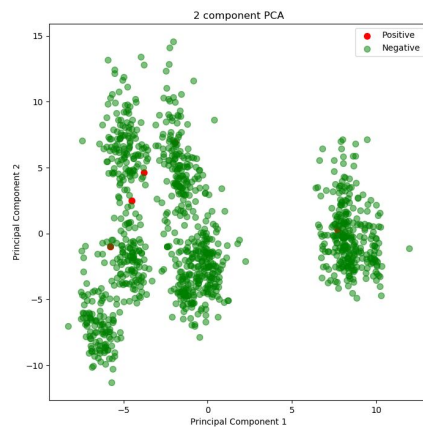
SVM combines hinge loss function and kernel trick, implicitly project data points into high-dimensional feature space to do non-linear binary classification. I use training set to optimize the SVM in sklearn package, and use it to forecast the result (with visualization).

Result on testing set (confusion matrix):

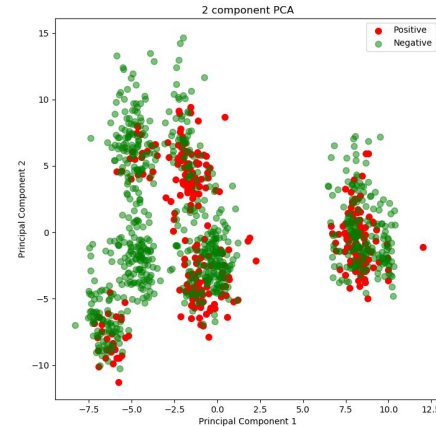
		Truth	
		Positive	Negative
Prediction	Positive	3	120
	Negative	4	978

-- python .\svm.py .\DataMining_IV_x2\DataSet_IV_Training.csv .\DataMining_IV_x2\DataSet_IV_Test.csv

(5) Logistic regression + Generated synthetic samples



Ground truth



Classification using Logistic regression

Logistic regression uses cross entropy between two Bernoulli distribution as loss function and use gradient descent for model optimization. I use training data to optimize the logistic regression model in sklearn package, and use it to forecast the result (with visualization).

Result on testing set (confusion matrix):

		Truth	
		Positive	Negative
Prediction	Positive	1	338
	Negative	6	760

```
-- python .\logistic_regression.py .\DataMining_IV_x2\DataSet_IV_Training.csv .\DataMining_IV_x2\DataSet_IV_Test.csv
```

Conclusion

Neural-based method can learn to detect rare event, but performs poorly on testing set due to overfitting, neither does non neural-based approaches. Although I tried over/down sampling and SMOTE algorithm to balance the dataset, the result is not significant. Therefore, I guess that more balanced dataset and customized methods are necessary to tackle this kind of rare event detection.