Project Individual Report

Meijiao Wang

06/30/2020

Introduction:

Based on user's historical behavior data of the Chinese Credit Platform to predict the probability of users overdue in the next 6 months.

For this project, we did data visualization, data cleaning, feature engineering, feature selection, Model Building, and Accuracy those six parts.

During this group work, I did the data visualization before processing. I also found out the model principle which are fit for our data, and choose which models we used in final work.

Detail:

Code:

```
#coding=utf-8

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

#### encoding='gb18030' deal with Chinese ###
train =
pd.read_csv('D:/Download/PPD_Training_Master_GBK_3_1_Training_Set.csv', encoding='gb18030')
test = pd.read_csv('D:/Download/test.csv', encoding='gb18030')
```

```
combine = pd.concat([train,test])



# target 1,0 ratio, we can find from the graph that the ratio is unbalance. we
should deal with it later
combine['target'].value_counts().plot.pie(autopct = '%1.2f%%')
plt.show()


def comulation(col):
    combine.groupby([col,'target'])['target'].count()
    combine[[col,'target']].groupby(['target']).mean().plot.bar()
    title = 'Overdue vs '+ col
    plt.title(title)
    plt.xlabel(['Not overdue', 'Overdue'])
    plt.show()


# 'Education_Info1','Education_Info5' overdue information
edu = ['Education_Info1','Education_Info5']
for i in edu:
    comulation(i)


def count_missing(first_col, last_col):
```

```
    new = combine.loc[:,first_col:last_col]
    cols = list(new.columns)
    count = [combine[col].isnull().sum() for col in cols]
    miss = pd.DataFrame({'cols':cols, 'count':count})
    print(miss)

## all the count of education is 0, shows that the education information is really
## significant for decision making
count_missing('Education_Info1','Education_Info8')


# show the number of missing data
missing=train.isnull().sum().reset_index().rename(columns={0:'missNum'})
# caculate the rate of missing data
missing['missRate']=missing['missNum']/train.shape[0]
# show by sort
miss_analy=missing[missing.missRate>0].sort_values(by='missRate',ascendin
g=False)

fig = plt.figure(figsize=(18, 6))
plt.bar(np.arange(miss_analy.shape[0]), list(miss_analy.missRate.values),
align='center',
```

```
            color=['red', 'green', 'yellow', 'steelblue'])


plt.title('Histogram of missing value of variables')
plt.xlabel('variables names')
plt.ylabel('missing rate')
plt.xticks(np.arange(miss_analy.shape[0]), list(miss_analy['index']))
plt.xticks(rotation=90)


for x, y in enumerate(list(miss_analy.missRate.values)):
    plt.text(x, y + 0.12, '{:.2%}'.format(y), ha='center', rotation=90)
plt.ylim([0, 1.2])


plt.show()
```

Individual Models:

Logistic regression

Logistic regression is one of the generalized linear regression analysis models, and its essence belongs to the classification problem, so it is mainly used when the explanatory variable is a categorical (discrete, such as 0, 1) variable. In the classification problem, logistic regression is better than linear regression, because linear regression will have a negative probability when fitting the explanatory variable to be discrete, which will lead to wrong sample classification. Logistic regression uses a logarithmic function to compress the prediction range between 0 and 1, thereby improving the prediction accuracy.[1]

From the original data, we select a sample dataset of 3000 rows and 50-dimensional features

for model training. The features are processed dynamically according to the cross-validation feedback of the model. Throughout the entire modeling process, the single-factor analysis and the numerical features are discretized for the output of the important feature after the xgboost model is trained. Construct and add combined features. Because linear models lack accurate characterization of non-linear relationships, feature combinations can add non-linear expressions to enhance the expressive power of the model. When the model has too many parameters, it is easy to encounter the problem of overfitting.

It is necessary to control the complexity of the model. The typical approach is to add regular items to the optimization target to prevent overfitting by penalizing too large parameters. L1 regularization tends to make the parameter become 0, and feature selection can be performed for high latitudes. The data has a good effect, so we choose L1 regularity.

KNN:

Compared with other algorithms, the KNN algorithm is a particularly well-implemented and easy-to-understand classification algorithm, which mainly classifies according to the distance between different features. The general classification algorithm must first train a model, and then use the test set to test the model, but the KNN algorithm does not need to train the model, directly using the distance between the sample to be tested and the training sample to achieve classification.

1st. Step: Calculate the distance between test data and each row of training data

2nd. Step: Now, based on the distance value, sort them in ascending order.

3rd. Step: Next, it will choose the top K rows from the sorted array.

4th. Step: Final, it will assign a class to the test point based on the most frequent class of these rows.[2]

DecisionTree-CART

Classification and regression tree is a kind of decision tree algorithm. It is the same as other decision tree algorithms. It is also composed of feature selection, tree generation, and pruning. CART is widely used and is used in the integrated model of trees. This type of decision tree can

solve both classification problems and regression problems.

Therefore, for classification trees and regression trees, CART has two generation methods. For any machine learning model, the training process is similar, the purpose is to make the training set data as good as possible to classify and fit. Whether CART is a classification tree or a regression tree, it is necessary to minimize certain errors in the training process and arrange the training data most appropriately.[3]

SVM

SVM refers to a series of machine learning methods. The basis of such methods is the support vector algorithm. The basic principle of the SVM algorithm is to find a hyperplane that can distinguish between the two types, so that the margin is maximized. [4] All points located on the marginal hyperplanes on both sides are called Support Vector. We use the SVM algorithm to solve the classification problem, which is SVC.

The goal of SVC is to return a "best match" hyperplane to divide or classify your data. After the hyperplane is obtained, the above feature values are provided to the classifier to view the prediction. This algorithm is very suitable for our needs

MLP

MLP can implement nonlinear discriminants, and if used for regression, can approximate the input nonlinear function. In fact, MLP can be used for "universal approximation", that is, it can be proved that any function with continuous input and output can be approximated by MLP. It has been shown that MLP with a hidden layer can learn any nonlinear function of the input.[6]

In the neural network, each node is a perceptron, the basic function of the neurons in the model biological neural network: the electrical signals from the outside are transmitted to the neurons through synapses. When the sum of the signals received by cell exceeds a certain threshold, The cell is activated and sends electrical signals to the next cell through the axon to complete the processing of outside information.

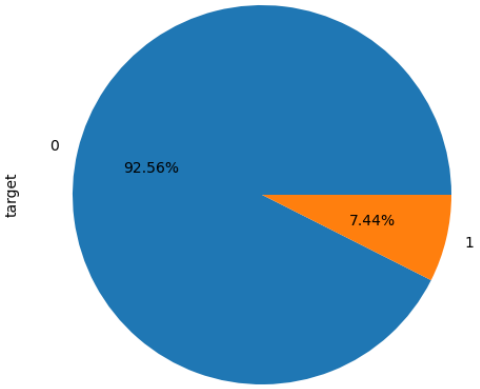
However, the perceptron learning algorithm cannot be directly applied to the parameter learning of the multi-layer perceptron model. The originally proposed learning scheme is: In addition to the last neuron, the weights of all other neurons are fixed in advance. The learning process is only to use the perceptron learning algorithm to learn the weight coefficients of the last neuron.

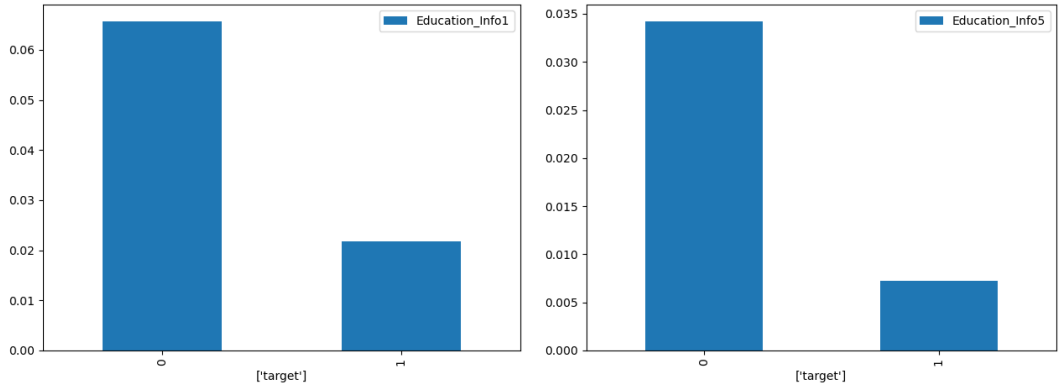
In fact, this is equivalent to transforming the original feature space into a new feature space through the first layer of neurons. Each neuron in the first layer constitutes one dimension of the new space, and then learns with a perceptron in the new feature space. The algorithm constructs a linear classifier.[5]

Obviously, since the weights of the first neuron layer need to be given manually, the performance of the model depends greatly on whether an appropriate first layer neuron model can be designed, and this depends on the problems and data faced. Understand, there is no way to solve the first layer of neuron parameters for any problem. The core idea is the gradient descent method, that is, the degree to which the training samples are misclassified as the target function, and each time an error occurs during training, the weight coefficient is updated in the direction of the negative gradient of the target function relative to the weight coefficient, knowing that the target has not been until the wrong sample.

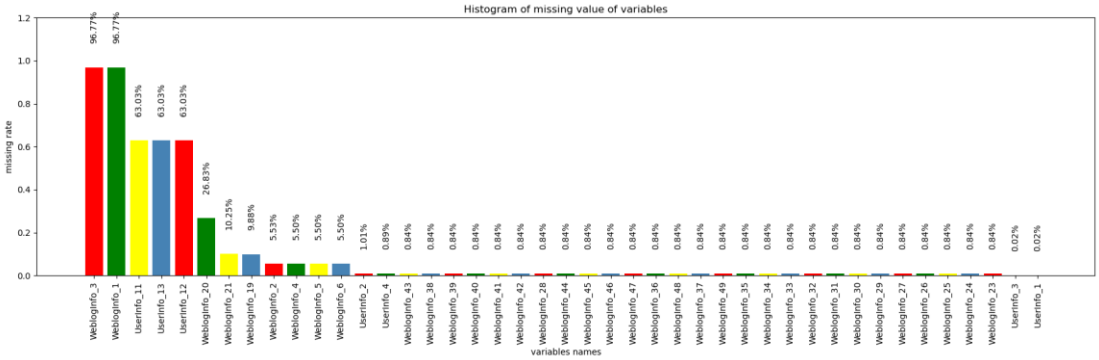
Result:

I found out approximate ratio of training dataset between two results. And we guess ‘target’ may have relationship with ‘Education’. So I make two graph to see the relationship of features-Education_Info1 and Education_Info5 with overdue. But we cannot see the clearly difference.





And then I did the graph of the data missing rate, to see clearly category of missing data.



Summary and Conclusions:

From this project, I learned how to use python to clean a data set, fill in missing values, how to use machine learning for feature engineering, and train models, and make predictions. And have more understanding of how to use the sklearn, pandas, seaborn and other packages in python.

Because we do not understand exactly about this dataset, we do not know some categories' meaning. So there may be errors in the final result. Due to the limited performance of the computer and the proficiency of using Python, it may not be accurate during data training. If I

can have a more comprehensive understanding of the data, it will be more details in processing the data, and the models will also get better results.

There is about 32% code I used from internet.

Reference:

1. [Machine Learning] Logistic regression: https://zhuanlan.zhihu.com/p/74874291 [13 June 2020]
2. KNN Algorithm - Finding Nearest Neighbors: https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_knn_algorithm_finding_nearest_neighbors.htm [n.d]
3. A Beginner's Guide to Classification and Regression Trees: https://www.digitalvidya.com/blog/classification-and-regression-trees/ [23 January 2019]
4. An Introduction to Support Vector Machines (SVM): https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/#:~:text=A%20support%20vector%20machine%20(SVM,on%20a%20text%20classification%20problem. [22 June 2017]
5. Jeshwanth C.(2015) Adaptive Multiple Optimal Learning Factors for Neural Network Training, The University of TEXAS at Arlington
6. Introduction to Artificial Neural Networks(ANN): https://towardsdatascience.com/introduction-to-artificial-neural-networks-ann-1aea15775ef9 [10 October 2019]