PRML Data Contest Report

V Sai Krishna and S Sivasubramaniyan - Team White Walkers

August 2020

1 Introduction

The given problem is about predicting whether an employee is going to leave the company or not in the coming few months. The data collected is about the ratings and remarks given by the employees of different companies on a single platform, where the employees can participate according to their choice. Let's see if we can extract useful information for the employers.

2 Data File Descriptions

The following files need to be in the same directory as the script for the script to run

- ratings.csv - ratings given by an employee of a company on the given date
- remarks.csv - remarks entered by the employee about the company on the given date
- remarks_supp_opp.csv- whether an employee was in favor of or opposed the remark of another employee
- train.csv - the training set
- test.csv - the test set

3 Extracting Features

The following 32 relevant features were extracted for each employee over which our model would be trained on:

Features based on employee-specific ratings

- f1 - Count of rating '1' given by the employee
- f2 - Count of rating '2' given by the employee

- f3 - Count of rating '3' given by the employee
- f4 - Count of rating '4' given by the employee
- f5 - Mean rating given by the employee
- f6 - Date weighted rating given by the employee, with the latest rating given the highest weightage
- f7 - Number of ratings given by the employee
- f8 - Latest rating given by the employee

Features based on company-specific ratings

- f11 - Count of rating '1' given by all employees to the company
- f12 - Count of rating '2' given by all employees to the company
- f13 - Count of rating '3' given by all employees to the company
- f14 - Count of rating '4' given by all employees to the company
- f15 - Mean rating given by all employees to the company
- f16 - Date weighted rating given by all employees to the company
- f17 - Number of ratings given by all employees to the company

Features based on employee-specific remarks

- f21 - Average length of remark given by the employee
- f22 - Number of remarks given by the employee
- f23 - Mean of effective support of remarks given by the employee. The effective support of a remark is calculated as the proportion of supporting employees among the total number of employees either supporting or opposing the remark
- f24 - Mean of effective opposition of remarks given by the employee. It is basically 1-f23, might play a role in certain classifiers

Features based on company-specific remarks

- f31 - Average length of remark given by all employees to the company
- f32 - Number of remarks given by all employees to the company
- f33 - Mean of effective support of remarks given by all employees to the company

- f34 - Mean of effective opposition of remarks given by all employees to the company

Features based on employee-specific expression of support or opposition to colleague's remarks

- f41 - Count of 'Support' given by the employee to colleagues
- f42 - Count of 'Oppose' given by the employee to colleagues
- f43 - Proportion of 'Support' given by the employee to colleagues
- f44 - Proportion of 'Oppose' given by the employee to colleagues

Features based on company-specific expression of support or opposition by employees to their colleague's remarks

- f51 - Count of 'Support' given by all employees to their colleagues in the company
- f52 - Count of 'Oppose' given by all employees to their colleagues in the company
- f53 - Proportion of 'Support' given by all employees to their colleagues in the company
- f54 - Proportion of 'Oppose' given by all employees to their colleagues in the company

Date of last rating converted to a single integer as follows

- 366 * (Year - 2014) + 30 * (Month - 1) + (Date)

The above 32 features were extracted from our data and passed on to the model for training.

4 Weighted Accuracy

We define the weighted accuracy function as follows, as necessitated by the problem statement.

```
def lossfunct(output,target):
    ln=(target==0).sum()
    lp=(target==1).sum()
    fp=((output==target) & (target==1)).sum()
    fn=((output==target) & (target==0)).sum()
    loss=(5*fp+fn)/(5*lp+ln)
    return loss
```

5 Training and Model Validation

On training multiple models with the above features, we get the best 5-fold cross-validation weighted accuracy of 85.6 percent using a Random Forest Classifier built with 1000 decision trees, each with a minimum leaf node size of 7 to prevent overfitting to a certain extent. The 5-fold cross-validation code along with the model parameters are as follows:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import KFold
kf = KFold(n_splits=5)
accuracy = 0
for train_index, test_index in kf.split(xtrain):
    X_train, X_test = xtrain[train_index], xtrain[test_index]
    y_train, y_test = ytrain[train_index], ytrain[test_index]
    model = RandomForestClassifier(n_estimators = 1000, class_weight = {0: 1, 1: 5}, min_samples_leaf = 7)
    model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    accuracy += lossfunct(y_pred, y_test)
print(accuracy/5)
```

6 Test Performance

Evaluating our test data through the above-trained model with 32 features, the top 3 performances from our kaggle submissions are reported below:

Model	Public Score	Private Score
min_samples_leaf = 7	89.32	85.35
min_samples_leaf = 8	89.18	85.48
min_samples_leaf = 10	89.04	85.61

Table 1: 32-Feature Random Forest Model Performances on the Testing Set

Adding some 20 more not-so-straightforward features, we were able to achieve slight improvements in accuracy with a best public score of 89.46 and a best private score of 88.06.