**UCern Question Resolve Status Prediction**

Prepared by Leonardo Ji

Training Data Scientist

Date 12/15/2017

Abstract: Using web scrapping we gathered questions dataset from a UCern group. We can create a model to predict resolved status. The final model has a precision of 70% and recall 98% on predicting “yes” on the resolved status.

**Key Takeaways**

UCern (<https://www.ucern.com>) is a social network site that foster collaboration between development organizations. There are many groups within UCern, for this project we are only looking at one group called “Care Management” that allows engineers to collaborate on technical support issues. Using web scrapping we gathered questions dataset from this group. Typically, an engineer would post a discussion thread question and another support engineer would answer this question. If the question is resolved then the discussion will be marked “Resolved” status. The goal of this project is to find a good Machine Learning model to predict “Resolved” status based on this data. Our final model has a precision of 70% and recall 98% on predicting “yes” on the “Resolved” status.

**Analysis Summary**

We analyze all questions and found a lot of repeated questions. Those repeated questions shared words with each other as high as 55%. The repeated questions are all previously resolved questions. We would like to predict “yes” on the “Resolved” status because if we know a similar question has been resolved before. Then we do not need to spend additional time on this question. This saves engineering resources and time.

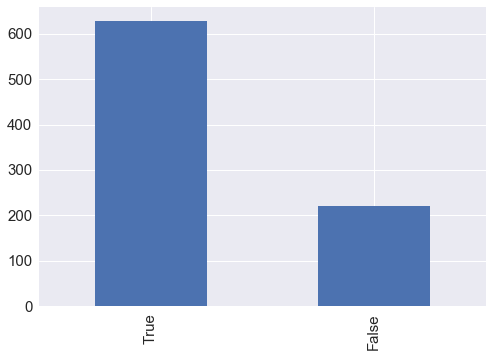
We used the below features from the dataset as input to our models. We think all these features influence the target “Resolved” status.

* type
* subject
* viewCount
* replyCount
* lastActivityDate
* author
* content
* question
* resolved

ViewCount and replyCount are continuous variables. Type and Author are categorical variables. But subject and content columns are text. To use them, we clean the text by removing stopping words, switching to lowercases, and replace common words but in various terms. For example, “post-acute”, “postacute”, “post acute” all means the same thing, we replace them with “postacute”. Then we can use Term-frequency Inverse Document Frequency (Tf-Idf) to generate term document matrix. Tf-Idf term document matrix distinguish important words in one document that is unique vs. all other documents.

The target variable is a binary “Resolved” status. More questions are solved than not resolved. Predicting not resolved status is harder because there are many reasons that question is harder to answer.

Figure 2 Insurance claims (0 is no, 1 is yes)



But in this case, we want to predict “Resolved” status is true, in other words a model with higher Precision and Recall ratios on “Resolved” status is true.

The detail results of all fourteen models listed in Table 1. We tried at least three different combination of the parameters for each one models. We used a method called Grid Search that iteratively running/tuning these models with different parameters. Sometimes we used several rounds of Grid Search to narrow down the parameter further than the first Grid Search. Then Grid Search returns the best parameters for the model. We used 10-fold cross validation to rule out models that were overfitting, overfitting means the model is too specific to the training dataset. For 10-fold cross validation we created ten separate datasets from sampling, then ran training and validation for ten times. Overfitting happens when the model is too specific to the training dataset, and the model does not do well on the new data. Six out of fourteen models were overfitting.

**Table 1. Classification Models Results**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Models | Best Parameters | Accuracy | Precision | Recall | 10-fold  Cross Validation Mean | CV Overfitting  (> 5% from Mean) |
| Logistic Regression | None | 0.582 | 0.7 | 0.7 | 0.546 | Yes |
| KNN | n\_neighbors=25 | 0.698 | 0.70 | 0.99 | 0.696 | Yes |
| Decision Tree | criterion='gini', min\_samples\_leaf = 4, min\_samples\_split = 5 | 0.608 | 0.75 | 0.66 | 0.635 | Yes |
| Random Forest | max\_features = 'auto', min\_samples\_leaf = 4, min\_samples\_split = 4, n\_estimators=1000 | 0.698 | 0.71 | 0.97 | 0.71 | No |
| SVM Linear | kernel='linear', C=0.01, class\_weight='balanced', gamma='auto' | 0.703 | 0.70 | 1.0 | 0.689 | No |
| Bagging | n\_estimators=300, max\_features=40 | 0.693 | 0.70 | 0.97 | 0.728 | No |
| Naive Bayes | None | 0.597 | 0.68 | 0.81 | 0.611 | Yes |
| Adaboost | base\_estimator=MultinomialNB(),algorithm="SAMME",n\_estimators=700,learning\_rate=0.003 | 0.682 | 0.69 | 0.98 | 0.68 | No |
| Extra Trees | criterion = 'gini', max\_depth=50,n\_estimators=100,min\_samples\_split=10,min\_samples\_leaf=5 | 0.693 | 0.70 | 0.98 | 0.698 | No |
| Gradient Boosting | n\_estimators=1000, learning\_rate=0.07, max\_depth = 4 | 0.677 | 0.73 | 0.85 | 0.712 | No |
| XGBoost | colsample\_bytree=1,learning\_rate=0.06,max\_depth = 3,min\_child\_weight=11,n\_estimators=1000,objective="binary:logistic",subsample = 0.6,seed = 1337 | 0.619 | 0.69 | 0.83 | 0.695 | Yes |
| Multi-layer Perceptron | activation='relu', alpha=1e-05, batch\_size='auto', beta\_1=0.9, beta\_2=0.999, early\_stopping=False,epsilon=1e-08, hidden\_layer\_sizes=(10, 1), learning\_rate='constant', learning\_rate\_init=0.001, max\_iter=100, momentum=0.9,nesterovs\_momentum=True, power\_t=0.5, random\_state=1, shuffle=True,solver='lbfgs', tol=0.0001, validation\_fraction=0.1, verbose=False,warm\_start=False | 0.698 | 0.70 | 1.0 | 0.694 | No |
| Stacking | **estimators=[('rf', clf\_rf), ('bag', clf\_bag), ('boost', clf\_boost), ('gb', clf\_gb), ('xt', clf\_xdt), ('nn', clf\_nn)], voting='soft'** | **0.693** | **0.70** | **0.98** | **0.719** | **No** |

**Summary**

* The Best models is those models had higher recall and precision ratios on “Resolved” status of true.
* I used Grid Search to determine the best parameters for each classification models.
* XGBoost is a very popular model used in Kaggle competitions. It is much faster model than Gradient Boosting. But the model was overfitting here. Overfitting was determined by 10-fold cross validation, any one of 10-fold accuracy ratio is greater than 4% from the 10-fold mean accuracy.
* Six out of fourteen models were overfitting.
* The final model we picked is an ensemble stacking model combining best models: Random Forest, Bagging, Adaboost, Extra Trees, Gradient Boosting, Multi-layer Perceptron. This model's accuracy is 69.3%, precision is 70%, recall is 98% on target resolved status is true, and 10-fold cross validations mean accuracy is 71.9%.