Red Hat Business Value Prediction

xz1863 Xinya Zhao

qt296 Qinxin Tan

rh2544 Ruojun Hong

Business Understanding

Like most companies, Red Hat is able to gather a great deal of information over time describing behaviors of individuals who interact with them. For example, when an individual visit their website, how many times he/she has clicked through a specific product link or what keywords he/she used for information search. With such batch of behavioral data, Red Hat is in search of better methods to determine which individuals have the potential to become their future customers and can bring business value to the company. In fact, each individual may have various activities interacting with Red Hat, some of which are good indicators of great customer potential while others not. An accurate identification of such valuable activities is critical, for only when these activities are correctly distinguished, can the company take further relative actions to approach the relative subjects, like sending promotion email of relevant products or offering special discounts. This kind of targeted marketing can be cost-saving and enables Red Hat to efficiently prioritize resources to generate more business and to better serve their customers.

From a data mining perspective, this business problem is to correctly classify each individual activity into one of the two classes - with business value or without any business value. An improved classification model will work for this prediction. To be more specific, our target variable is whether an activity of an individual has business value for Red Hat.

Data Understanding

The original data we use contains two separate tables: the people file and the activity file. The people file contains all of the unique people (and the corresponding characteristics) that have performed activities over time. Each row in the people file pertains to characteristics of a unique

person. Each person has a unique people_id. The activity file describes all of the unique activities (and the corresponding activity characteristics) that each person has performed by date. Each row in the activity file represents a unique activity performed by a person on a certain date. Each activity has a unique activity_id. The data instances from these two data sources are shown below.

1	people_id	char_1	group_1	char_2	date	char_3	char_4	char_5	char_6
2	ppl_100025	type 2	group 36096	type 3	8/26/22	type 14	type 6	type 8	type 3
3	ppl_100025	type 2	group 36096	type 3	8/26/22	type 14	type 6	type 8	type 3

The people file instance

people_id	activity_id	date	activity_category	char_1	char_2	char_3	char_4	char_5
ppl_253974	act2_4499446	12/27/22	type 2					
ppl_389401	act2_2296940	6/15/23	type 4					
ppl_24888	act1_232867	3/8/23	type 1	type 2	type 2	type 3	type 3	type 2

The activity file instance

A good news is that labeled data for our defined problem is available. The business value outcome is described by a yes/no field attached to each unique activity in the activity file. The outcome field indicates whether or not each person has completed the outcome within a fixed window of time after each unique activity was performed.

Nearly all features (11/14 in act file, 38/41 in people file, and 49/55 in total) contained in our dataset are categorical, and among them 28 from the people dataset are binary. Figure 1 and 2 in the appendix shows the class number distribution of nearly all categorical features from both two data sets, except the 2 features whose number of class values exceeds 3000. From the charts and statistics of features extracted from the data, we can conclude that the data available to this problem is typical categorical data, most features of which are with a medium number of categories (between 15 to 40). This kind of categorical data are welcomed by many classifiers in

data mining techniques, like decision tree and logistic regression, where the feature vectors can be used to train the models with or without some further processing. Having a clear understanding of the data enabled us to make reasonable conversions in the next steps of data preparation for model training.

Data Preparation

Our data preparation stage mainly deals with 3 problems in the data: excessive data volume, missing value, and improper data type.

The original people data set and activity one include 189,118 and 219,7291 records respectively, which when merged up will be such a large volume that it will be very time-consuming for model training and parameter tuning. Due to time constraints and the scale limit of the sklearn libraries, we first randomly sampled about 40,000 activities from the activity file and pulled out all people records related to those activities. Also, considering the possible negative effect caused by data size reduction on the testing accuracy, we tested our model using holdout evaluation and cross-validation techniques in the further practice.

Based on the data summary of our datasets, we found that there are several columns within people file that contain large amounts of missing values. Since those columns are all categorical variables describing people's characteristics, we substituted these values with a new defined category "type 0", which will then be smoothly processed if further dummies are needed to be created for these relevant fields.

When dealing with the feature "activity date", we considered the time-series factors like seasonality and trend, which may contribute to some special patterns in the data. For example, enterprises may be more likely to update their information system software, which is the main

products of Red Hat, in the beginning of a new year. While tree induction models can work well with date-time variables, other classifiers like logistic regression only allow numeric data type. Thus, we first convert the date to 3 categorical fields: year, month and date. Then with all the categorical features (including binary ones), we unify the data format by using integers to represent different categories accordingly, which make the data acceptable to most data mining classifiers. What's more, considering logistic regression and SVM models should have better performance with sparse data, we dummy-coded every categorical variable for further model optimization. For example, there are 3 unique types of "char_3" feature of every people. Instead of having just 1 column, we split "char_3" to 3 separate columns, char_3_1, char_3_2 and char_3_3, filling each cell with a binary value 0 or 1 representing the specific type.

At last, we merged the two datasets on perople_id. The final data after we processed was $43946\ \textit{rows} \times 404\ \textit{columns}$, in which each row describes an activity and its associated person who performed it.

Modeling and Evaluation

1. Classifiers Choice

Based on our data understanding, our team selected to use a supervised classification model, as labeled data is available. The 3 most popular data mining classification models, Decision Trees, Logistic Regression and Supported Vector Machine were examined and executed in our exercise. There are many alternatives, including k-nearest neighbors, bayesian classifications, neural networks etc. But the team ruled out that k-Nearest Neighbor (k-NN) wouldn't be intuitive to stakeholders, since our objective was not to find records that are close to each other.

2. Evaluation Metrics Choice

Before modeling, it is essential to choose proper evaluation metrics that can reflect the business interest in model performance. The business problem to solve is to distinguish the activities that are most likely to indicate business value in the subjects. Only when these true positive activities are correctly distinguished, further profits could be brought to the company after some target marketing. But if some activities are falsely recognized as positive by the model, then the company will suffer a loss caused by wrong targeting cost. The other two occasions – truly predicted negative activities and falsely predicted negative – will not bring any profits or costs to the company because the relevant population are not targeted. Thus, just like most churn cases, Red Hat concerns true positives and false positives more over true negatives and false negatives. This leads us to absorb ROC-AUC value as our evaluation metric, which does not equally award all the true predictions like accuracy score. What's more, AUC is also favorable for ranking entities according to their likelihood to be positive, which may satisfy stakeholders' interests in the probabilities to be positive. At the same time, some other AUC-relative performance metrics like Lift are also used during the evaluation considering their intuitive business friendly acceptability.

3. Models Overview

	Decision Tree	Logistic Regression	SVM
Pre-processing	None	Dummies Creation, Scalarization	Dummies Creation
Feature Selection	Information Gain, Uni-variable Performance	Regularization, Random Forest Feature Importance, Polynomial Feature	Regularization, Random Forest Feature Importance,
Complexity Control	Mini_samples_split, Mini_leaf_split,	Regularization C	Regularization C Gamma
Baseline AUC	0.875	0.816	0.884
Best AUC	0.937	0.909	0.910
Used Techniques	Cross Validation	Grid Search, Pipeline, Cross Validation	Grid Search, Cross Validation

4. Optimization and Evaluation Framework & Procedure

Decision Tree

Decision tree creates a model that predicts the value of the target variable by learning simple decision rules inferred from the data features. Since tree models are potent to deal with categorical features, we did not create dummies for the model training.

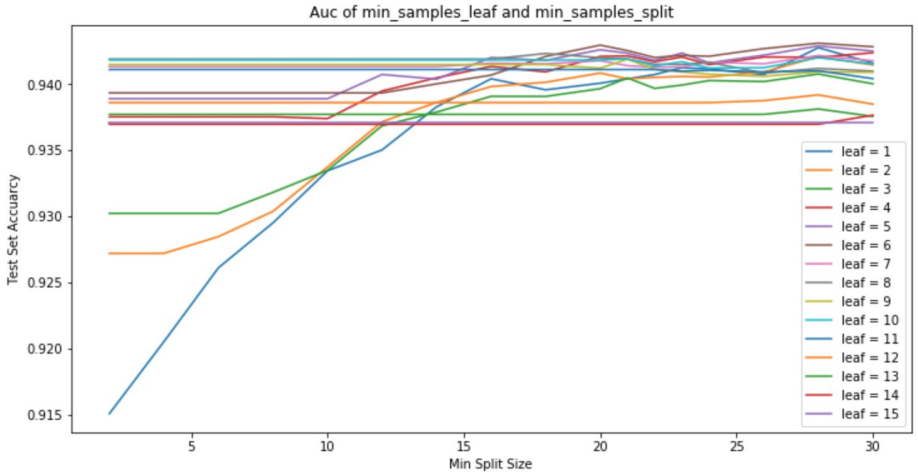
The model was initially trained and tested on the small sampled dataset, where it was trained with 75% of the data and 25% hold out test data. Then using “entropy” criterion to train the baseline model got an AUC of 0.875, which is pretty high. Then for the optimization, there

are two aspects we focused on. One is feature extraction and the other is hyper parameter optimization. The parameter optimization is based on the improved model with selected features.

In the all 58 features, not all of them are valuable. First, we did uni-variable performance evaluation. We calculated AUCs of only use one feature as the input to predict the target variable, and then compared the AUC relative to each feature. The ROC curves are shown in appendix Picture 3.

Then we built models with top N features from above and got the AUC of each model, where N is from 1 to 57. According to the result, we found that the model with the top 27 features, nearly half of the total, had the best AUC, which reached 0.913. Relative ROC curves are shown in appendix Picture 4.

Next, we optimized two hyper parameters of the model: the min_samples_split and the min_samples_leaf. Since those two parameter can control the max depth of the decision tree, there is little need to optimize the max_depth. Based on the selected 27 top feature, we first trained the model with a wide range of min_samples_splits and min_samples_leafs values. Then, with the pre-understanding, we set the best ranges of those two parameters as below.



Auc of different min_samples_leaf and min_samples_split

We got the optimal parameter min_samples_split of 28 and min_samples_leaf of 6.

To avoid overfitting, we used cross validation and larger dataset to evaluate the generalization performance of the tuned model. The results are shown below. (Merge_sample2 is double size of Merge_sample1.)

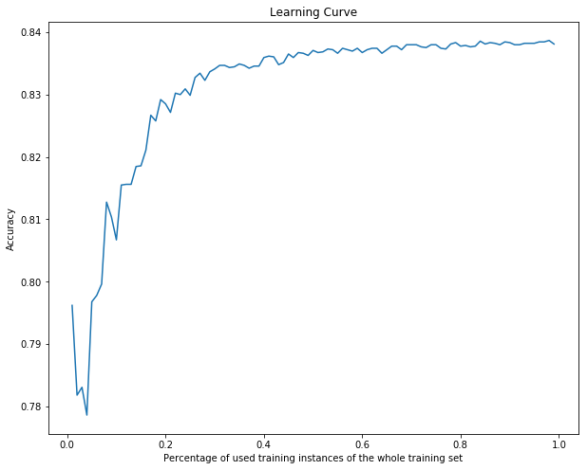
	Merge_sample1	Merge_sample2
AUC using 0.25 holdout data	0.936	0.940
AUC using cross validation	0.908	0.900
Accuracy using cross validation	0.827	0.815

From the table, we can clearly tell that after optimization the AUC has increased, which indicated the effectiveness of our optimization. And the consistent high AUC also proves that our model is not overfitting.

Logistic Regression

For the logistic regression model, we used the grid search techniques with pipeline to do the feature engineering work. The baseline model was trained with 0.75 of the small sampled original dataset and tested using 0.25 holdout data. The initial AUC was 0.816. Then we used grid search together with cross validation to tune the two parameters – penalty and C, which are widely used to control the model complexity. We got an improved AUC of 0.887 and a best hyper-parameter combination of 'l2' and '0.1' for penalty and C. Then, based on this framework, we built up a pipeline to include more engineering procedures. We included standard scaler to preprocess the data and got an slightly improved performance(auc) of 0.893. Then we further added another procedure to the pipeline– to include quadratic features generated by original features. However, due to the large size of data (for both features and records), this tuning is not completed. We went on with the optimization by preprocessing all the categorical features to be substituted by their dummies, and this time, we got an exciting improved AUC update of 0.909.

At last, we had feature selection in the pipeline without any preprocessing of the data to see is feature elimination will improve the model. However, it turned out that the model got an even worse performance comparing with the initial baseline model, and the new AUC is only 0.808. Thus, we did not do further experiment of different combination of this procedure and other practiced ones.



Learning curve

Based on all our practice, we can tell the best model is the one executed on sparse data (data with dummies) with tuned hyper-parameters of 'l2' for penalty and 1.0 for C. These parameters not only work to control the complexity but also limit the number (or weight) of different features. To ensure our high AUC is not the result of overfitting, we went further for the testing of the optimized model by executing on a double-sized dataset and used 0.75 versus 0.25 for training and testing. The result is that the AUC decreased by less than 0.03, which is acceptable considering data variance and randomness.

SVM

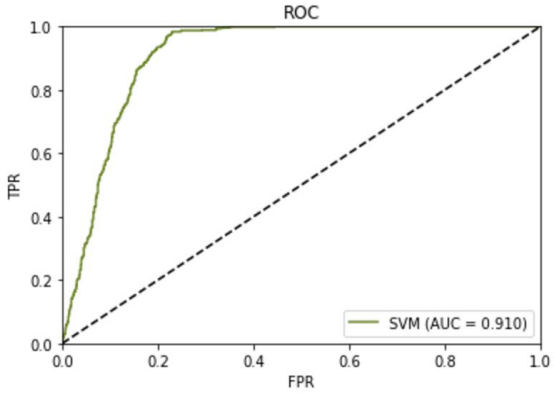
The library we used was *sklearn.svm.SVC* (*Supported Vector Classification*) in Python. As the documentation indicates, the implementation is based on libsvm. The fit time complexity is more than quadratic with the number of samples which makes it hard to scale to dataset with more than a couple of 10000 samples, thus we sampled our dataset again in order to fit the model to data. The dataset after another random sampling includes $8789\ rows \times 404\ columns$.

sklearn.svm.LinearSVC is implemented in terms of liblinear rather than libsvm and therefore has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples. We decided not to use linear kernel because an SVM model with linear kernel is similar to logistic regression that we discussed earlier.

We first quickly fitted the model to the training set using default hyper-parameters: $C = 1.0,\ gamma = auto,\ kernel = 'rbf'$, calculated score and plotted ROC curves on the test set. The result was not bad. The AUC was approximately 0.884 and the accuracy was 0.828, as shown by the screenshot we took (see Figure 5 in the appendix). The AUC is the main metric we decided to use to evaluate our model.

We then processed to optimize the model by feature selection and hyper-parameters tuning. As discussed before, we would not use LinearSVC to explore SVM with a linear kernel. However, we may use LinearSVC to explore linear correlations in order to achieve feature selection. Another library we used was s*klearn.feature_selection.SelectFromModel*. After some experiments of parameters of LinearSVC, when C=0.01 it gives us 70 out of 404 remaining features. Feature selection alone improved both AUC score and accuracy by approximately 2%.

After this, we applied grid search to find the optimal hyper-parameters of the SVM model. The library we used was *sklearn.model_selection.GridSearchCV*, which also came with a cross-validation parameter letting us specify the k fold. We performed grid search on $'gamma' : [1e^{-6}, 1e^{-5}, 1e^{-4}, 1e^{-3}, 1e^{-2}], 'C' : [1, 10, 100, 1000, 10000]$ and found the best parameters $\{c = 1000, gamma = 1e^{-4}\}$. Applying this pair of parameters increased AUC score and accuracy by both 1%. Below is the screenshot of the best ROC curve, with feature selection and hyperparameter tuning.



Best ROC curve

5. Final Optimal Algorithm Choice

Based on all our experiments, we would like to choose the improved decision tree as our optimal algorithm. Although logistic regression also got a comparable performance and could perform well with large amounts of features, it is not that intuitive to business stakeholders as decision tree and the data preprocessing (to generate dummies for nearly all the features) will be time consuming and require considerable resource of computing and storing when the dataset is large. SVM got a good performance too but has a fatal disadvantage that it can only be trained by a limited size of data, which is very likely to contribute to overfitting or bad prediction when

there are lots of missing values existing in the dataset. What's more, since we thought that Red Hat would want to know the probabilities, which would be required to generate intuitive business friendly accepted metrics like Lift, SVM is also not appropriate. Thus, the decision tree model which is with good complexity control and very efficient will be our final choice.

Deployment

To apply our model to this business problem, we need to collect all features from the target activity and its associated customer and dummy code those features as we did at the data preparation stage, making these features consistent. Profitable activities can be clearly defined by activities being predicted to be 1. Of course, the customers performing those activities are profitable customers who have business value, and worth approaching.

There might be new profitable customers and new correlations. As we collect new data, the model should be updated regularly. We could also collect the feedbacks of actual outcome to prove that our model actually works. We had no idea how Redhat collected the data. There might be biased data, so we should be aware of any outliers and closely monitor any abnormal drop of accuracy.

The data is personal private information. The company should protect customers privacy when using those data to avoid the danger of information leek. Red Hat has already encrypted the released dataset for this purpose.

Due to time and scale constraints, we sampled the data randomly in order to fit the model to it. This might affect the accuracy of our prediction. In the future, we should be able to migrate our project to Hadoop ecosystem to better serve our purpose as the size of the dataset grows.

Appendix

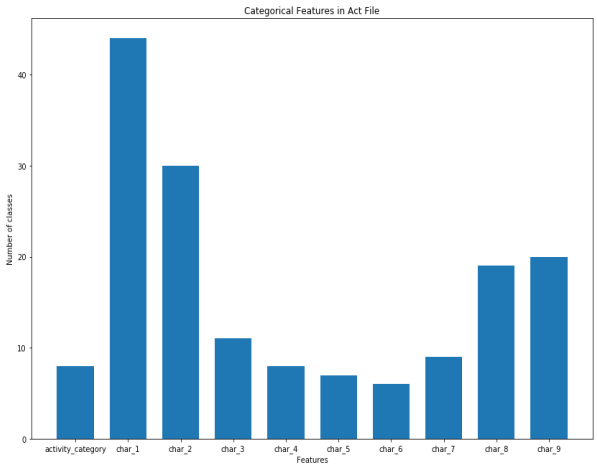


Figure 1: Categorical Features in Act File

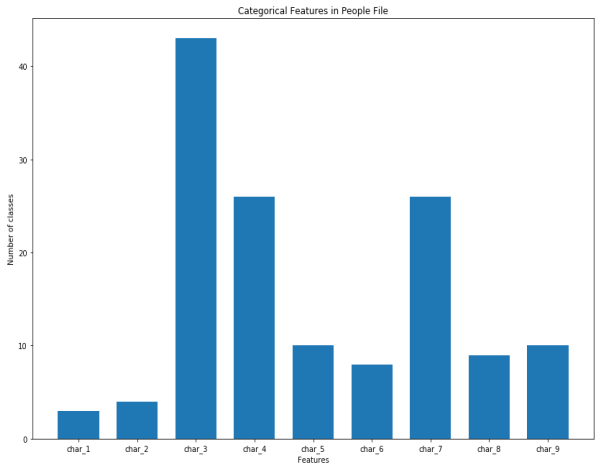


Figure 2: Categorical Features in People File

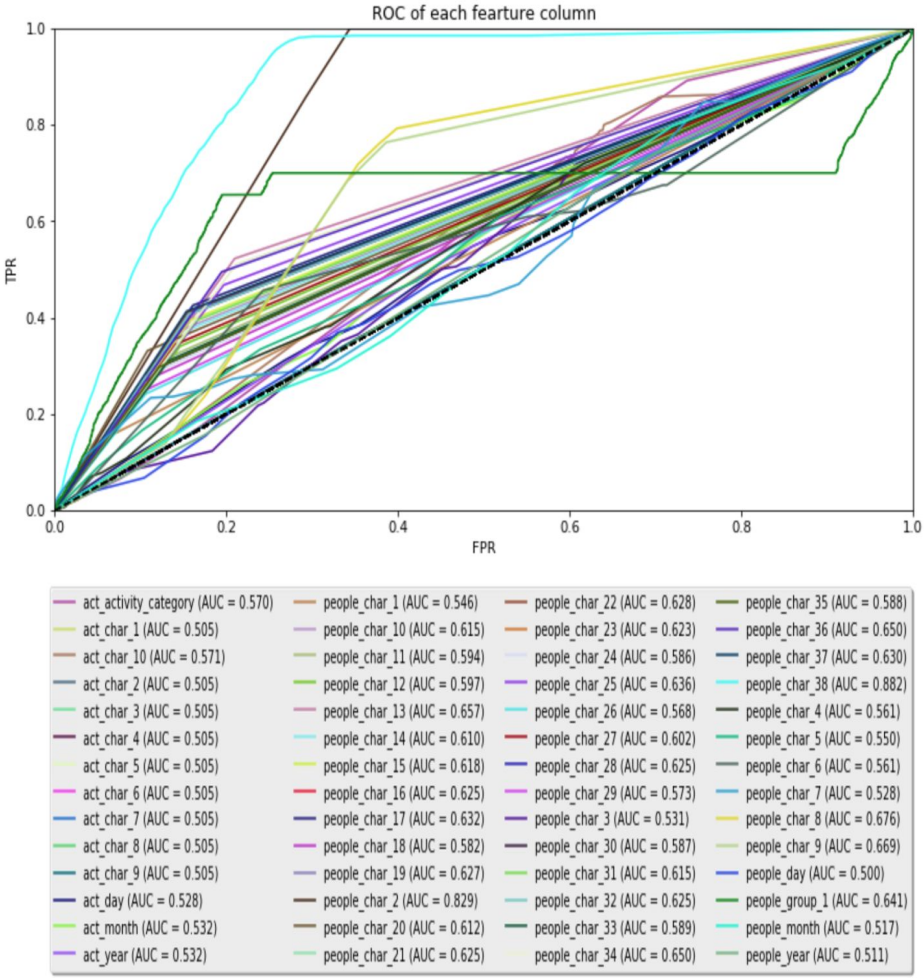


Figure 3: Decision Tree ROC curve of each feature columns

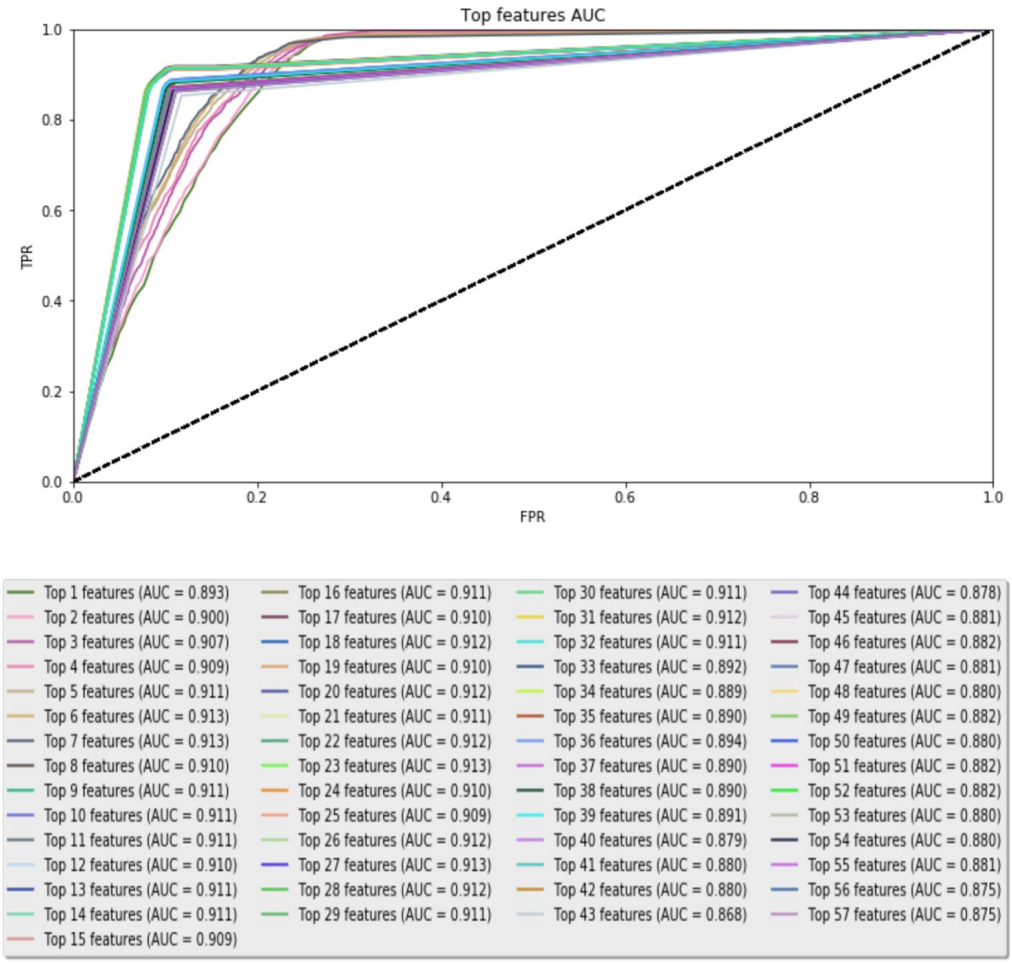


Figure 4: Decision Tree ROC curves of Top N features

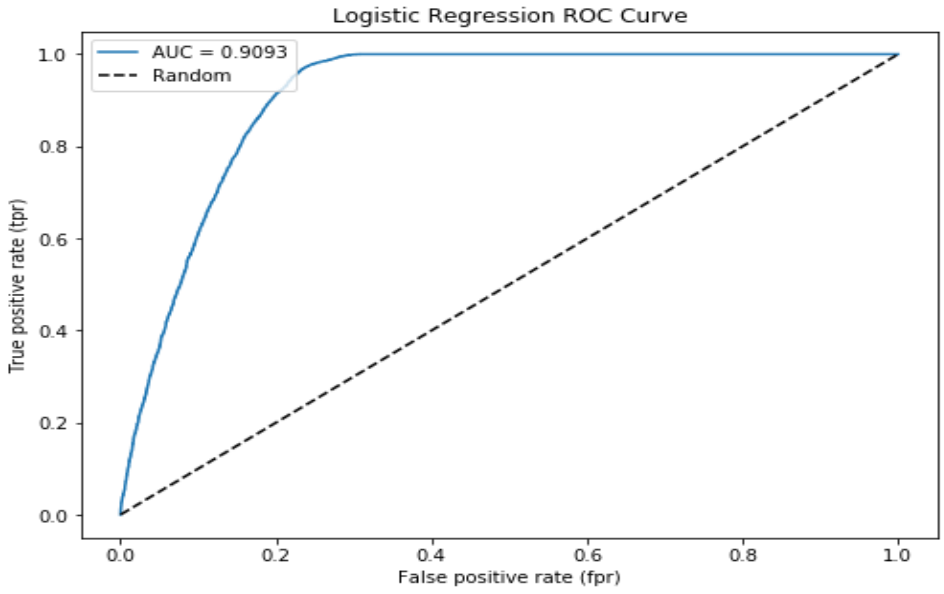


Figure 5: Logistic Regression ROC Curve of the Best Model

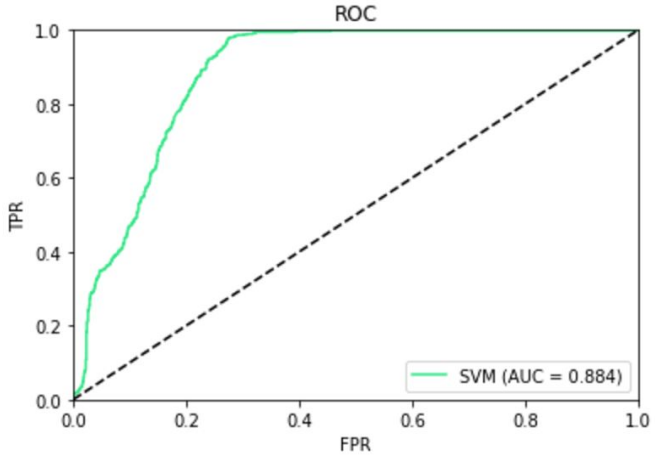


Figure 6: ROC curves of SVM without any performance tuning

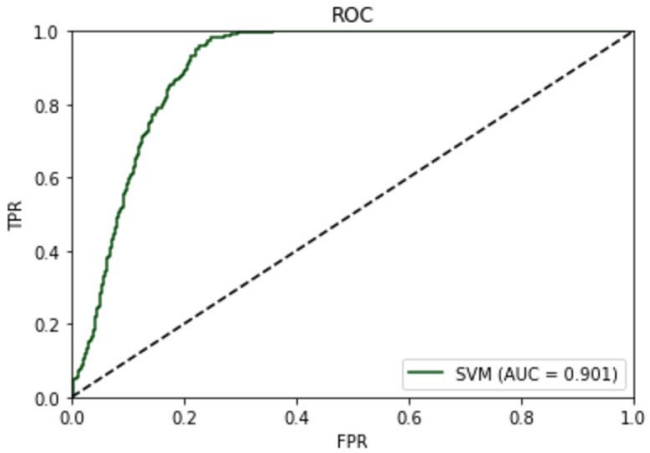


Figure 7: ROC Curve of SVM with feature selection