Project Report

On

Predicting Red Hat Business Value

By

Arish Balasubramani (304455611)

Cheryl Maria Jose (305052753)

Janak Soni (305061437)

ABSTRACT

In this project, we are asked to predict the customers who are a real potential to bring Business to Red Hat. The intent of this project is to create a classification algorithm that accurately identifies and predicts which customers have the most potential business value for Red Hat based on their characteristics and behavior.

1. INTRODUCTION

Red Hat, has been gathering a great deal of information over time about the behavior of individuals who interact with them. They're in search of better methods of using this behavioral

data to predict which individuals they should approach, and even when and how to approach them and which individual would provide Business to Red Hat.

We are trying to create a classification algorithm that accurately identifies which customers have the most potential business value for Red Hat based on their characteristics and activities. With an improved prediction model in place, Red Hat will be able to more efficiently prioritize resources to generate more business and better serve their customers

2. DATA SET:

- This competition uses two separate data files that may be joined together to create a single, unified data table: a people file and an activity file (Training and Testing)
- The people file contains all the unique people (and the corresponding characteristics) that have performed activities over time. Each row in the people file represents a unique person. Each person has a unique people_id.
- The activity file contains all the unique activities (and the corresponding activity characteristics) that each person has performed over time. 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.

3. IMPLEMENTATION

In this project, we will be predicting the Business outcome of Red Hat. From the Training and Testing dataset we will take each column to features of activities and behaviors to predict outome.

Following are techniques used in this project:

- Removal of unwanted columns with string data from both Training and Testing dataset.
- Removal of Rows with NaN or Null values from both Training and Testing Dataset.
- Perform OneHot Encoding over both Training and Testing dataset.
- Merging of data frames and avoiding of columns with Null values.
- Using cross validation for data splitting and training the algorithm.
- Perform Dimensionality Reduction using Principle Component Analysis (PCA) technique.
- Train and predict using modified Random Forest Classifier.
- Train and predict using modified AdaBoost Classifier.

Removal of unwanted columns with string data from both Training and Testing dataset:

Using Pandas data frame to read and saving the CSV files of Training and Testing Dataset. Performing preprocessing of Training and Testing Dataset by using ***drop()*** function to drop

columns that don't contribute to characteristic's and behavioral activities, Example: activity_id, date, group_1.

```
#drop feature column that are not required from feature matrix in people table
df = df.drop('group_1', axis=1)
df = df.drop('date', axis=1)
```

Removal of Rows with NaN or Null values from both Training and Testing Dataset:

In the next step of processing the dataset using ***dropna()*** function to check the entire Training and Testing dataset for any NaN or null columns and remove them for better accuracy.

```
# drop colums with NAN for easier machine learning and remove complexity
df_train1 = df_train1.dropna(how='any')
```

Perform OneHot Encoding over both Training and Testing dataset:

The Function ***get_dummies()*** of pandas is a Very important feature extraction and learning method and plays a major role in converting all the different behaviors and characteristic's in every single column of training and testing dataset to individual features, increasing the size of the data frame and total number of feature columns.

```
# perform Onehot encoding on remaining feature colums
cols_to_transform = ['activity_category','char_1','char_2','char_3','char_4','char_5','char_6','char_7','char_8','char_9
for colums in cols_to_transform:
    df_with_dummies = pd.get_dummies(df_train1[colums])
    df_train1 = df_train1.drop(colums, axis=1)
    df_train1 = df_train1.join(df_with_dummies, lsuffix='_left', rsuffix='_right')
```

Merging of data frames and avoiding of columns with Null values:

This function ***merge()*** of pandas is used to merge the activities table training and testing datasets with the behavior table people dataset based on people_id. Here we merge and match only activities of each person to his behaviors and characteristics of the people id table so that no duplication of rows and no null or NaN columns are generated.

```
df_new = df_train1.merge(df, on='people_id',how='left')
df_new.head()
```

Using cross validation for data splitting and training the algorithm:

In this method we are using built in ***train_test_split()*** of the cross_validation library to learn the training dataset by splitting it into training and testing dataset so as to train the algorithm which will help in prediction. We are using "test_size=0.3" means that pick 30% of data samples for testing set, and the rest (70%) for training set.

```
# Randomly splitting the original dataset into training set and testing set
# The function"train_test_split" from "sklearn.cross_validation" library performs random splitting.
# "test_size=0.3" means that pick 30% of data samples for testing set, and the rest (70%) for training set.

from sklearn.cross_validation import train_test_split
#from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=2)
```

```
print x_train.shape
print x_test.shape
print y_train.shape
print y_test.shape
```

```
(110330, 319)
(47285, 319)
(110330L,)
(47285L,)
```

Perform Dimensionality Reduction using Principle Component Analysis (PCA) technique:

From the previous cross validation method we know that we have a large number of feature columns that are available. Which is not good for feature learning and extraction of data, and time complexity and time consumption is large. Hence we are reducing total of 319 feature to 50 features using the PCA method. So, this PCA method helps to reduce the dimension of the matrix such that it ensures that no 2 features provide the same information and all 50 features are the best and are different. We have to fit and transform the training dataset so that we can use the same transformation for testing dataset.

```
# reduce dimensions. Use PCA
pca = PCA(n_components=50)
# train the algorithim
pca.fit(x_train)
# transform the vector for training set
X_transform = pca.fit_transform(x_train)

# put it into a dataframe
X_transform = pd.DataFrame(X_transform)
print X_transform.head()
```

```
# transform the vector for testing set
X_transform_test = pca.fit_transform(x_test)
# put it into a dataframe
X_transform_test = pd.DataFrame(X_transform_test)

X_transform_test.head()
```

Train and predict using modified Random Forest Classifier:

Random forest classifier is an very important and advantageous classifier which helps identify the best tree available for prediction and since the feature size is very big. Random forest suits the best and we are using n_estimators = 19, bootstrap = True, random_state=2 helps in providing better prediction results. After using the training dataset and making algorithm learn from the PCA processed data, now we can do the prediction over the testing dataset and the algorithm proved an accuracy of approx. **85%**.

```
#Random Forest is an classifier using multiple decision trees to arive a best prediction

from sklearn.ensemble import RandomForestClassifier

my_RandomForest = RandomForestClassifier(n_estimators = 19, bootstrap = True,random_state=2)
```

```
#train the RandomForest alg by using the FIT method

my_RandomForest.fit(X_transform,y_train,sample_weight=None)
```

```
#Make prediction over the trained RandomForest alg using the testing sample:

y_RandomForest_predict = my_RandomForest.predict(X_transform_test)

print(y_RandomForest_predict)
print(y_RandomForest_predict.shape)
```

```
[1 1 1 ..., 1 0 0]
(31523L,)
```

Train and predict using modified AdaBoost Classifier:

This is another Famous Classifier which helps us to boost the results of PCA to find better predictions. Using this method helps us to classify and identify which Algorithm works better for this prediction model and best classifier. We achieved an accuracy of 83%.

```
# Adaboost is a classifier which gives weightage to each features and reduces the errors in getting the best prediction

from sklearn.ensemble import AdaBoostClassifier

my_AdaBoost = AdaBoostClassifier(n_estimators=19,random_state=2)
```

```
#train the adaboost alg by using the FIT method

my_AdaBoost.fit(X_transform,y_train,sample_weight=None)
```

```
AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None,
          learning_rate=1.0, n_estimators=19, random_state=2)
```

```
#Make prediction over the trained Adaboost alg using the testing sample:

y_adaboost_predict = my_AdaBoost.predict(X_transform_test)

print(y_adaboost_predict)
print(y_adaboost_predict.shape)
```

```
[1 1 1 ..., 1 0 1]
(47285L,)
```

4. EXPERIMENT:

All the Features matrix and label were made into the table using pandas Data frame. To this data frame we also added all the feature columns of training dataset and people dataset which was available. After careful extraction of features using OneHot encoding technique and PCA Dimensionality reduction to pick the reduced number of important features from 319 to 50 features. The Feature table is randomly split into 70% and 30% for training and testing respectively. Then on training the Random forest using n_estimators = 19, we achieved and accuracy of 85% and on training the AdaBoost using n_estimators = 19, we achieved an accuracy of 83%. We tried change the random split of training and testing dataset to 60% and 40% or any other ratio, along with PCA values to be less or more than 50, and changing the n_estimators to less than 19. We recognized that the accuracy of prediction was falling lesser than 85%

5. CONCLUSION:

Based on the features extraction method that we implemented we were able to achieve an accuracy of more than 85%.

6. REFERENCES:

1. **Kaggle.com**: https://www.kaggle.com/c/predicting-red-hat-business-value
2. **Adaboost**: http://scikit-learn.org/stable/modules/ensemble.html
3. **Random Forest**: http://scikit-learn.org/stable/modules/ensemble.html