HR Analytics: Using Machine Learning to Predict Employee Turnover

CSC 529—Case 3
Name: Yenong Du
ID: 1291549

I. Introduction

The keys of a successful company are long-term success, a healthy work environment, and high employee retention. But when a company suffers a high rate of employee turnover, that results in huge monetary losses of the company. Some costs are tangible such as training expenses and the time it takes from when an employee starts to when they become a productive member. And some are intangible, such as new product ideas, great project management, or customer relationships. Recognizing and understanding what factors that were associated with attrition will allow companies to increase employee productivity, thus the company could grow constantly. Predicting turnover is at the forefront of needs of Human Resources in the company, which gives managers the opportunity to take corrective steps to build and preserve the successful business.The goal of this study is to understand the key variables that influence employee turnover and to come up with a model to classify an employee's risk of attrition. The implementation of this model will help management to create or improve different retention strategies on targeted employees, furthermore to create better decision-making actions.

II. Data Description

The dataset I used for this study is Datasets from Kaggle.com. There are 14999 observations and 9 variables. The dataset describing the characteristics of the relationship with turnover is numerical and string, which is used to distinguish left from the stayed samples. The features are simulated. The dependent feature is Turnover, which has 2 classes: 1 is left, 0 is stayed. The rest 8 attributes are continuous, ordinal, discrete and categorical attributes

Independent attributes:
Satisfaction: a numeric indictor, the level of satisfaction which is graded by employees
evaluation: a numeric indicator,the last evaluation which is graded by company
projectCount: an integer,the number of projects which is dealed with by employees at the same time
averageMonthlyHours: the average monthly hours the employee work
yearsAtCompany: an integer value, the time the employee spent at the company
workAccident: a boolean value, the number of whether they have had a work accident
promotion: a boolean value, the number of whether they have had a promotion in the last 5 years
department: IT, RandD, accounting, HR, management, marketing, product management, sales, support, technical
salary: 3-level ordinal variable, the salary of employees

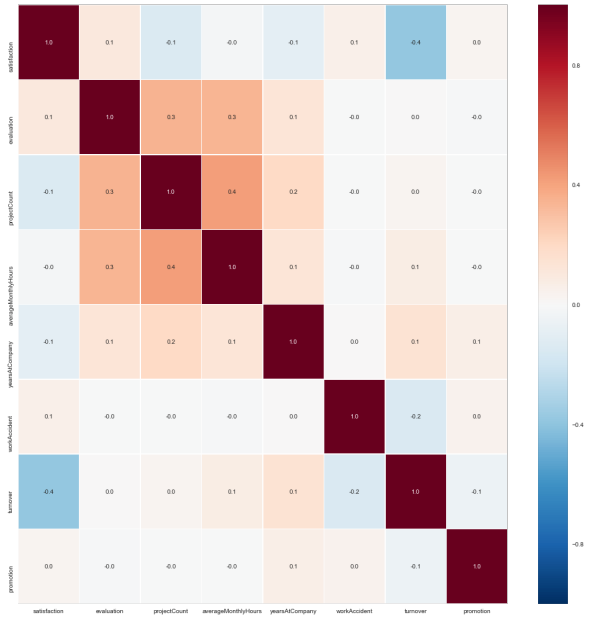
III. Data Cleaning

In order to prepare the data for training in different classifiers, I take a simple look at the dataset. The dataset from Kaggle.com is super clean and contains no missing values. I rename the certain columns for better readability: satisfaction_level—> satisfaction, last_evaluation—> evaluation, number_project —> projectCount, average_montly_hours—>averageMonthlyHours, time_spend_company—>yearsAtCompany, Work_accident—> workAccident, promotion_last_5years—> promotion, sales—> department, left—>turnover. The feature—salary, an ordinal feature, has 3 classes. The feature—department has 10 different kinds of departments. I convert them into numerical values. The features, evaluation, satisfaction are continuous variables, which separately split them into 5 classes, and convert them into numerical values. The feature—averageMonthlyHours is discrete variable, the range from less than 100 hours to 350 hours. I separate it to 5 classes and convert to numerical value in order to easily analyze the data.

IV. Data Analysis

From the heat map(fig. 1), there are moderate positively correlation, around 0.3 and 0.4 between projectCount, evaluation and averageMonthlyHours, and moderate negatively correlation, around - 0.4 between satisfaction and turnover. The employee who spent more hours and did more projects are evaluated highly.Moreover, people who experience lower satisfaction tend to choose leaving the company more.

fig. 1 The heat map



After heat map, I check the relation between turnover and other variables (fig. 2). I find that employees with low salary has the highest turnover attempt. The probability of high salary given turnover is 6.63% and the probability of low salary given turnover is 29.7%, which make me deep think about what the work department is provided for low, medium and high salaries. 50% of the employees in the

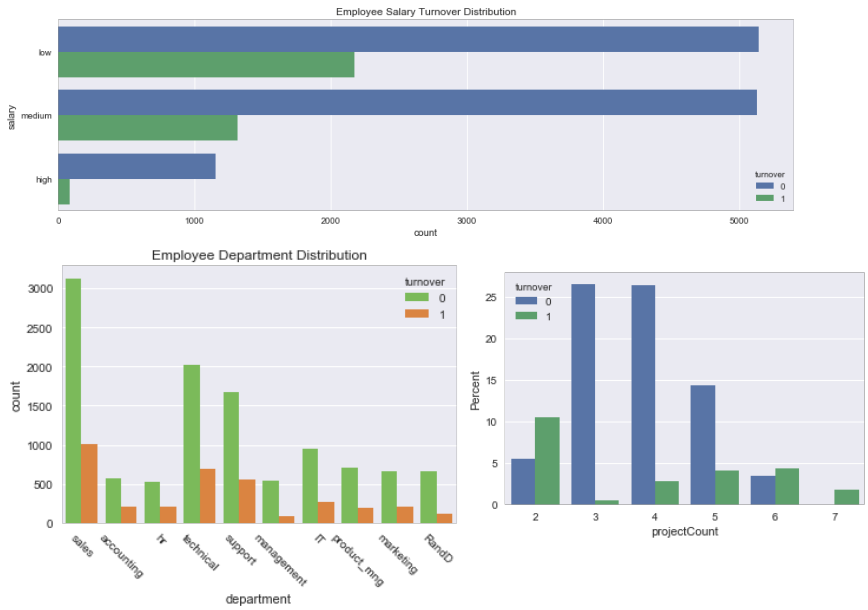


fig.2 the relation between turnover and salary, department, and projectCount

sales department have low salary, which is similar to the trend that the sales department is the top one to have employee turnover. The second and third ones are technical and support department. And the management department had the smallest amount of the turnover.

From the bar graph of projectCount and turnover, the employee with 2, 6 or 7 projects left the company, especially the employees with 7 projects all left the company. Then I find that the employees in the technical department relatively have higher probability having 6 or 7 projects, which explain why the technical department is the second place to have employee turnover.

Based on the heat map, I know that projectCount, evaluation, and averageMonthlyHours have relatively higher positive correlation. It is logical thinking that as the projectCount increased, so did average monthly hours. From fig. 3, employees who didn't have a turnover had a consistent evaluation despite the increase in projects. There is an increase in evaluation for employees who did more projects within the turnover group, however the employees with low performance only have 2 projects, and they tend to leave the company more. On the contrary, the employees with high performance also tend to leave. The sweet spot for employees staying in the company is within 0.6-0.8.

fig.3. box plot

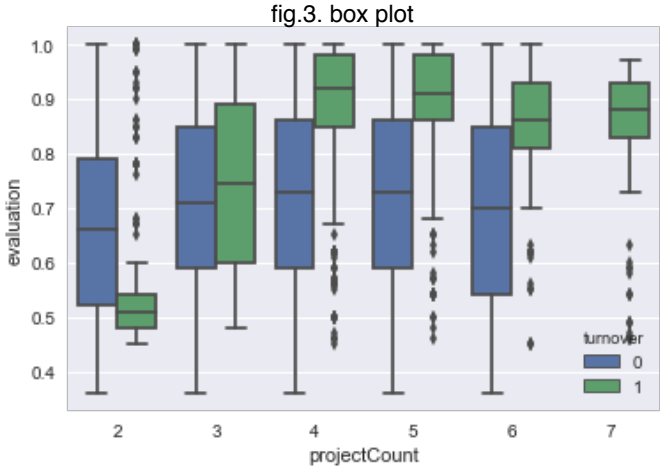


fig. 4

From fig. 4, there are 3 distinct clusters for employees who left the company. Cluster 1 is hardworking but sad employee which is a good indication that employees who left the company were good workers but felt horrible at their job. Cluster 2 is employees who were badly evaluated and felt bad at work. Cluster 3 is the employees who loved their work and were evaluated highly for their performance.

V. Experimental Results

I split dataset into two part: training set and test set. In order to evaluate the models for several time, I try not only feature selection, but also feature engine. Some of the variables have continuous-value with a large range. At the beginning of the study, I normalize them and dummy the ordinal, binary variables. Then I get the accuracy of the NB model around 0.81 with feature selection. Because there's no tuning parameters procedure, I go back to rebuild feature engine. I separate some variables into 5 classes, and don't choose dummy variables.

Naive Bayes

The target variable is binary. And on the data cleaning, I convert all variables into ordinal, nominal and discrete variables, which is good at applying Bayes' theorem. Because not every variable is binary-valued variable. I apply Gaussian Naive Bayes for this study. I perform sklearn.naive_bayes.GaussianNB on the training set, and run 10-fold cross validation to evaluate predictive model. The results are shown in table 1.

table 1	training set	test set	10-fold CV
accuracy	0.793	0.793	0.794(+/- 0.023)

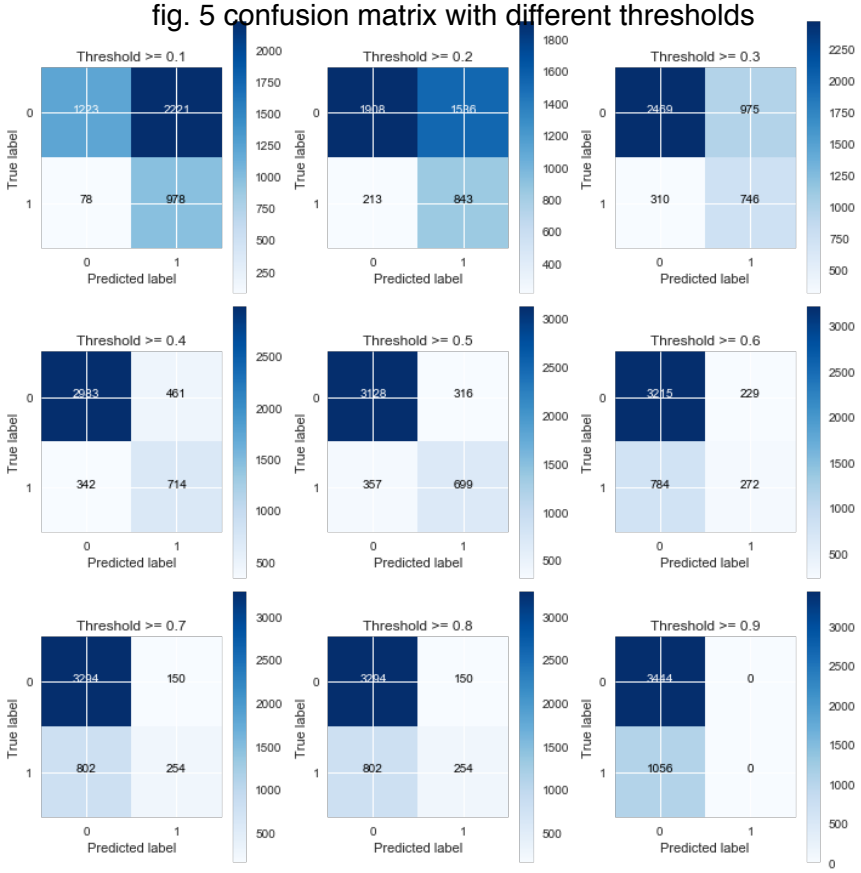
The model of GaussianNB has very few tunable parameters. In order to evaluate the performance of the model, I use feature selection—chi-square statistical test. Based on the result of feature_selection.SelectPercentile, 3 variables —satisfaction, workAccident and yearsAtCompany are chosen as the selected variables. Using selected variables, I perform the GaussianNB again. The results are shown in table 2.

table 2	training set	test set	10-fold CV
accuracy	0.846	0.850	0.846(+/- 0.017)

The results are much better. Because there is no tuning parameters step, the next step is changing classification threshold. The method I used is predict_proba(), that returns the probabilities for each class. I could control precision and recall by changing the threshold to assign a record to class 1. The results are shown in table 3, and fig. 5, 6.

table 3	precision	recall	f1-score	support
0	0.90	0.91	0.90	3444
1	0.69	0.66	0.68	1056
avg/total	0.85	0.85	0.85	4500

fig. 5 confusion matrix with different thresholds



```
Recall metric in the testing dataset:  0.926136363636
Recall metric in the testing dataset:  0.798295454545
Recall metric in the testing dataset:  0.706439393939
Recall metric in the testing dataset:  0.676136363636
Recall metric in the testing dataset:  0.661931818182
Recall metric in the testing dataset:  0.257575757576
Recall metric in the testing dataset:  0.24053030303
Recall metric in the testing dataset:  0.24053030303
Recall metric in the testing dataset:  0.0
```

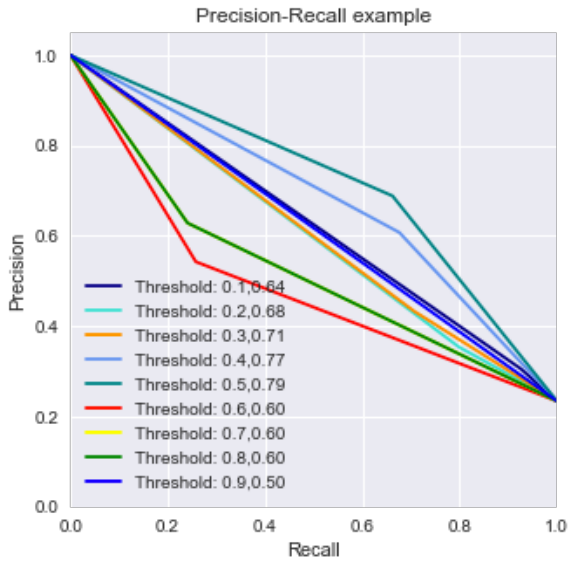


fig. 6 precision_recall_curve

VI. Experimental Analysis

In the exploratory analysis, very few people in sales department have high salaries, and a lot of purple with lower salaries in sales, technical and support departments. People in the management department have higher probability on higher salary. In the total of 14999 employees that compose the dataset, 3599 people have left. ProjectCount, evaluation and averageMonthlyHours are positive correlation. When employees have more projects, they will spend more time on working, and the company will evaluate them with high ability. Therefore, the trends on turnover are all similar. I can make the hypothesis that there is two groups of employee who tend to leave at the both peaks of the bimodal curve. Those who work less because they have gets lower scores. The characteristics of people who choose turnover are working on many projects, low salary, lower or higher evaluation, or working in the company around 5 years. Those characteristics are triggered for some problems: why hardworking employees doesn't like his/her jobs, and why hardworking employees love their jobs, but leave the company, and Why employees who left worked more hours than employees who didn't, even with the same project count? Those questions should be taken into account for further survey. Based on this dataset, I have some guess. Firstly, when someone spend 3-5 years at a company, he/she is start to be treated as a 'professional' in his area which make him/her a job-hopper to pursue higher salary. Secondly, when a person spend 6 or more years at a certain company, he/she must be used to it and would not like to go unless he/she has to.

An advantage of the naive Bayes classifier is that it requires a small amount of training data to estimate the parameters necessary for classification. My dataset has 14999 observations, and the target

variable is binary. The naive Bayes classification is suited for the dataset. The accuracy is one of the basic performance measures for this classification. Before feature selection, the accuracy is found to be 79%. Using satisfaction, workAccident and yearsAtCompany as features in the model, the accuracy is around 85%. Because it is binary classification, the default threshold is 0.5, I choose to adjust this threshold to understand how the choice of threshold will affect my model. This pattern is very clear: the more you lower the required probability to put a certain in the class '1' category, more records will be put in that bucket. This implies an increase in recall, but at the same time, a decrease in precision. Even though recall is the goal metric, keeping the model being accurate as a whole is also the goal. By visually seeing the performance of the model depending on the threshold we choose, we can investigate a sweet spot where recall is high enough while keeping a high precision value. When threshold is 0.5, the area under ROC is the largest one, 0.79.

VII. Conclusion

With all of the above information, the summary below is why the employees probably leave. Employees leave when they are underworked or overworked (less than 150 hr/month, or more than 250 hr/month). Employees with either really high or low evaluations should be taken into consideration for high turnover rate. Employees with low or medium salaries are the bulk of employee turnover. Employees with light or heavy working projects are at risk of leaving the company. Employees with 4 or 5 years experience in the same company have high probability of leaving the company. Employee satisfaction is the highest indicator for employee turnover. Combined with the questions I raise above, I suggest that the dataset should be updated from the features from satisfaction survey. Work Accident and Years at company are the other two biggest factors in determining turnover.

In this employee turnover problem, we would have an estimate of the probability that he/she will leave the company, rather than simply predicting whether he/she will leave. Because it's about people decision, the model is used to arm management and HR with much better relevant information for better decision making. Considering retaining the employees, the company should take into account the strategy that different costs are weighed based on the type of employee being treated.

Naive Bayes classifiers are extremely fast for both training and prediction, and they provide straightforward probabilistic prediction. Although they couldn't perform as well as more complicated model, such as random forest classifier, this classifier is a good choice as an initial baseline classification. Thus, the next step is to explore more sophisticated models with some baseline knowledge from naive Bayes model.

Appendix:

Coding:

```
#load the library
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as matplot
import seaborn as sns
%matplotlib inline
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split,KFold
from sklearn import preprocessing
from sklearn.metrics import roc_auc_score,accuracy_score, classification_report, precision_score, recall_score,
confusion_matrix, precision_recall_curve
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn import metrics,tree
from sklearn.feature_selection import SelectKBest,chi2
from sklearn import feature_selection,cross_validation
```

```
import math
import itertools
from itertools import cycle

#read the dataset
data=pd.read_csv('HR_comma_sep.csv')

#check missing data
data.isnull().any()
#rename the columns
data = data.rename(columns={'satisfaction_level': 'satisfaction',
                    'last_evaluation': 'evaluation',
                    'number_project': 'projectCount',
                    'average_montly_hours': 'averageMonthlyHours',
                    'time_spend_company': 'yearsAtCompany',
                    'Work_accident': 'workAccident',
                    'promotion_last_5years': 'promotion',
                    'sales' : 'department',
                    'left' : 'turnover'
                    })

#data exploratory
#correlation map
f,ax=plt.subplots(figsize=(18,18))
sns.heatmap(data.corr(),annot=True,linewidths=.5,fmt='.1f')
plt.show()
#salary vs turnover
f, ax = plt.subplots(figsize=(15, 4))
sns.countplot(y="salary", hue='turnover', data=data).set_title('Employee Salary Turnover Distribution')
plt.show()
#department vs turnover
color_types = ['#78C850','#F08030','#6890F0','#A8B820','#A8A878','#A040A0','#F8D030',
               '#E0C068','#EE99AC','#C03028','#F85888','#B8A038','#705898','#98D8D8','#7038F8']
# Count Plot (a.k.a. Bar Plot)
sns.countplot(x='department',hue='turnover', data=data, palette=color_types).set_title('Employee Department Distribution')
# Rotate x-labels
plt.xticks(rotation=-45)
plt.show()
#project count vs turnover
ax = sns.barplot(x="projectCount", y="projectCount", hue="turnover", data=data, estimator=lambda x: len(x) / len(data) * 100)
ax.set(ylabel="Percent")
plt.show()
#evaluation vs turnover
fig = plt.figure(figsize=(15,4),)
ax=sns.kdeplot(data.loc[(data['turnover'] == 0),'evaluation'] , color='b',shade=True,label='no turnover')
ax=sns.kdeplot(data.loc[(data['turnover'] == 1),'evaluation'] , color='r',shade=True, label='turnover')
ax.set(xlabel='Employee Evaluation', ylabel='Frequency')
plt.title('Employee Evaluation Distribution - Turnover V.S. No Turnover')
plt.show()
#average monthly hours vs turnover
#KDEPlot: Kernel Density Estimate Plot
fig = plt.figure(figsize=(15,4))
ax=sns.kdeplot(data.loc[(data['turnover'] == 0),'averageMonthlyHours'] , color='b',shade=True, label='no turnover')
ax=sns.kdeplot(data.loc[(data['turnover'] == 1),'averageMonthlyHours'] , color='r',shade=True, label='turnover')
ax.set(xlabel='Employee Average Monthly Hours', ylabel='Frequency')
plt.title('Employee AverageMonthly Hours Distribution - Turnover V.S. No Turnover')
plt.show()
#project count--evaluation vs turnover
sns.boxplot(x="projectCount", y="evaluation", hue="turnover", data=data)
plt.show()sns.lmplot(x='satisfaction', y='evaluation', data=df,
           fit_reg=False, # No regression line
           hue='turnover')
#plot of satisfaction vs evaluation
sns.lmplot(x='satisfaction', y='evaluation', data=data, fit_reg=False,hue='turnover')
```

```
plt.show()
#kmeans cluster
# Graph and create 3 clusters of Employee Turnover
kmeans = KMeans(n_clusters=3,random_state=2)
kmeans.fit(data[data.turnover==1][["satisfaction","evaluation"]])
kmeans_colors = ['green' if c == 0 else 'blue' if c == 2 else 'red' for c in kmeans.labels_]

fig = plt.figure(figsize=(10, 6))
plt.scatter(x="satisfaction",y="evaluation", data=data[data.turnover==1],
        alpha=0.25,color = kmeans_colors)
plt.xlabel("Satisfaction")
plt.ylabel("Evaluation")
plt.scatter(x=kmeans.cluster_centers_[:,0],y=kmeans.cluster_centers_[:,1],color="black",marker="X",s=100)
plt.title("Clusters of Employee Turnover")
plt.show()
#years at company vs turnover
ax = sns.barplot(x="yearsAtCompany", y="yearsAtCompany", hue="turnover", data=data, estimator=lambda x: len(x) / len(data)
* 100)
ax.set(ylabel="Percent")
plt.show()

#feature engine
data["salary"]=data["salary"].map({"high":0,"low":1,"medium":2})
data["department"]=data["department"].map({"IT":0,"RandD":1,"accounting":2,"hr":3,"management":4,"marketing":
5,"product_mng":6,"sales":7,"support":8,"technical":9})
data['evaluation']=pd.cut(data['evaluation'],5,labels=[0,1,2,3,4])
data['satisfaction']=pd.cut(data['satisfaction'],5,labels=[0,1,2,3,4])
data['averageMonthlyHours']=pd.cut(data['averageMonthlyHours'],5,labels=[0,1,2,3,4])

#Model
#define the function
kfold = KFold(n_splits=10)
def classification_model(model,data_train,data_test,target_train,target_test):
    model=model.fit(data_train,target_train)
    predition=model.predict(data_test)
    print("Test Accuracy: ",model.score(data_test,target_test))
    print("Train Accuracy: ",model.score(data_train,target_train))
    scores=cross_val_score(model,data_train,target_train,scoring='accuracy',cv=kfold,n_jobs=4)
    print("Accuracy in 10-fold CV: %0.3f (+/- %0.3f)" % (scores.mean(), scores.std() * 2))
def featurePercentile(model,data_train,target_train):
    percentiles = range(1, 100, 5)
    results = []
    for i in range(1, 100, 5):
        fs = feature_selection.SelectPercentile(feature_selection.chi2, percentile=i)
        diag_train_fs = fs.fit_transform(data_train, target_train)
        scores = cross_validation.cross_val_score(model, diag_train_fs, target_train, cv=10)
        print (i,scores.mean())
        results = np.append(results, scores.mean())

    optimal_percentil = np.where(results == results.max())[0]
    print ("Optimal percentile of features:{0}".format(percentiles[optimal_percentil]), "\n")
    optimal_num_features = int(math.floor(percentiles[optimal_percentil]*len(data_train.columns)/100))
    print ("Optimal number of features:{0}".format(optimal_num_features), "\n")
def plot_confusion_matrix(cm, classes,normalize=False,title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=0)
```

```
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        #print("Normalized confusion matrix")
    else:
        1#print('Confusion matrix, without normalization')

    #print(cm)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
def model_rp_cm(clf,data_train,target_train,data_test,target_test):
    clf=clf.fit(data_train,target_train)
    prediction=clf.predict(data_test)
    print(classification_report(target_test,prediction))
    pred_proba=clf.predict_proba(data_test)

    thresholds = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
    plt.figure(figsize=(10,10))

    j = 1
    for i in thresholds:
        target_test_predictions_high_recall = pred_proba[:,1] > i

        plt.subplot(3,3,j)
        j += 1
        Cm=confusion_matrix(target_test,target_test_predictions_high_recall)
        np.set_printoptions(precision=2)
        print("Recall metric in the testing dataset: ", Cm[1,1]/(Cm[1,0]+Cm[1,1]))
        class_names = [0,1]
        plot_confusion_matrix(Cm, classes=class_names, title='Threshold >= %s'%i)

    colors = cycle(['navy', 'turquoise', 'darkorange', 'cornflowerblue', 'teal', 'red', 'yellow', 'green', 'blue','black'])
    plt.figure(figsize=(5,5))

    j=1
    for i,color in zip(thresholds,colors):
        target_test_predictions_prob = pred_proba[:,1] > i
        fpr,tpr,thresholds=metrics.roc_curve(target_test,target_test_predictions_prob)
        roc_auc=metrics.auc(fpr,tpr)
        precision, recall, thresholds = precision_recall_curve(target_test,target_test_predictions_prob)

        # Plot Precision-Recall curve
        plt.plot(recall, precision, color=color,label='Threshold: %s,%0.2f'%(i,roc_auc))
        plt.xlabel('Recall')
        plt.ylabel('Precision')
        plt.ylim([0.0, 1.05])
        plt.xlim([0.0, 1.0])
        plt.title('Precision-Recall example')
        plt.legend(loc="lower left")


train,test=train_test_split(data_D,test_size=0.3)
target_train=train['turnover']
data_train=train.drop('turnover',axis=1)
target_test=test['turnover']
```

```
data_test=test.drop('turnover',axis=1)
GNB=GaussianNB()
classification_model(GNB,data_train,data_test,target_train,target_test)
#feature selection
featurePercentile(GaussianNB(),data_train,target_train)

select_feature = SelectKBest(chi2, k=3).fit(data_train, target_train)
train_select=select_feature.transform(data_train)
test_select=select_feature.transform(data_test)

a=select_feature.scores_
b=data_train.columns
feature_dict={}
for i in range(len(a)):
    feature_dict[b[i]]=a[i]
for key,value in sorted(feature_dict.items(),key=lambda x:x[1]):
    print ("%s: %s" % (key, feature_dict[key]))
classification_model(GNB,train_select,test_select,target_train,target_test)
# AUC
model_rp_cm(GNB,train_select,target_train,test_select,target_test)
```