Predictive Modeling Report

CIS432-80
Team 31

Member:
Yuying He
Yingluo You
Tianyao Liu
Sichun Li
Congrong Shen

1. Introduction

This report presents the analysis process of using machine learning algorithms and models to analyze the Home Equity Line of Credit (HELOC) application made by real homeowners. The aim of this analysis is to predict the likelihood that applicants will repay their HELOC account within 2 years. This credit scoring method could be used by financial institutions when deciding whether or not to approve a loan. In this report, we did data cleaning, train-test split, and built six predictive models using the train set. Next, we evaluated the model performance based on different performance metrics. Lastly, an interface was built to visualize and simplify the prediction model for users.

2. Data Cleaning

We did data cleaning to correct and remove corrupt records from the data set. Three parts are included in this stage.

- Deal with missing values and meaningless values

Firstly, we deleted meaningless value. After referring to the data dictionary, we found the following meaningless values: ‘1’ in column ‘MaxDelqEver’ representing ‘No such value’ and all data point with the special value ‘9’ representing ‘No Bureau Record or No Investigation’. We replaced those value with ‘nan’ and dropped rows containing those values using dropna function. Secondly, we replaced special value ‘-7’ and ‘-8’ with median value of the corresponding feature using Imputer. The reason why we choose median is that after describing the dataset, we found that the distribution of record is not normal. Inclinations exists and so do outliers. In this situation, median is a better measurement than mean.

- Dealing with categorical variables

For the output variable ‘RiskPerformance’, we transformed it to a binary variable using LabelEncoder function, with ‘0’ representing ‘bad’ and ‘1’ representing ‘good’.

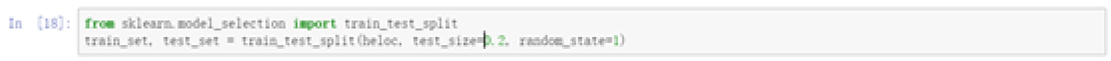
Regarding the input variable, ‘MaxDelq2PublicRecLast12M’ and ‘MaxDelqEver’ are categorical variables. For the ‘MaxDelq2PublicRecLast12M’ variable, both ‘5’ and ‘6’ means ‘unknown delinquency’, while ‘8’ and ‘9’ means ‘the other’. Thus, we marked the former as ‘5’ and the latter as ‘8’. Next, we used get_dummies to transform it to multiple columns after replace each value with a unique character. Lastly, they were concatenated with the rest of the columns. In this way, we could deal them as categories instead of numeric values.

- Feature Scaling

Because the measurement unit of each feature of the system may be different, the indicator needs to be standardized in order to be evaluated in the model and make the process more efficient. Here, we used standard scaler, transforming numeric features with the range (-1, 1) in models except LDA.

3. Module Construction

After data cleaning, we could construct and train models. Firstly, we split data into train and test set randomly by code *train_test_split* and set *test_size* = 0.2.



```
In [18]: from sklearn.model_selection import train_test_split
         train_set, test_set = train_test_split(heloc, test_size=0.2, random_state=1)
```

3.1 Model Used

- Logistic Regression

Logistic regression is a statistical model that in its basic form uses a linear combination of independent variable(predictors) to model a binary dependent variable, although many more complex extensions exist. In this case, the output is the risk performance of each credit card applicant, which is a binary variable. Thus, logistic regression is helpful to describe data and to explain the relationship between risk performance and input features.

- Decision Tree

A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences. It is one way to display an algorithm that only contains conditional control statements. In this case, we aim at splitting the input space to regions and assign a value (which class of Risk Performance, 0 or 1) to each region. So, decision tree is suitable to conduct this prediction task.

- Random Forest

Random forest is a way of averaging multiple deep decision trees. At each split, it randomly selects mm features (out of pp) with the goal of reducing the correlation between classifiers and getting overall better performance using simple trees. In this case, we use random forest as a complement to decision tree.

- Gradient Boosting

Boosting is an algorism in machine learning that is used to reduce bias and variance in supervised learning. It is one of the most powerful techniques for building predictive models.

Being developed from AdaBoost, gradient boosting involves loss function, weak learner, and additive model. It is used in classification and regression problems, producing a prediction model, typically decision trees (CART). In this case we aim at classifying the risk to two categories. Building models based on gradient boosting is suitable in the classification problem.

- Linear Discriminant Analysis

Linear discriminant analysis (LDA) is a generalization of Fisher's linear discriminant, a method used in statistics, pattern recognition, and machine learning to find a linear combination of features that characterizes or separates two or more classes of objects or events. This method is suitable to be used in this classification problem.

- K-Nearest Neighbors

K-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. It classifies new data point according to the label of the k-nearest data points. So, it is suitable to be used in this classification problem.

3.2 Model Construction

After preparing data, we set parameters for each used model. Then, we used GridSearchCV to construct each model with specific parameters and conducted cross validation by setting cv=5 to avoid problems including overfitting and selection bias.

```
model = GridSearchCV(pipeline, hyperparameters[name], cv = 5, n_jobs = -1)
```

Here, GridSearhCV also enabled us to try more values for Hyper-parameter. This will be explained in detail in the next part.

4. Adjust Hyper-parameters

In logistic regression, linspace means return evenly spaced numbers over a specified interval, which starts from 1e-3, end as 1e3, and range in 10.

```
l2_hyperparams = {
    'logisticregression__C': np.linspace(1e-3, 1e3, 10)
}
```

In random forest model, we restricted n_estimator, which is the number of trees in the random forest, that is the number of weak classifiers. We also set max feature as "auto", "sqrt","0.33", which is sqrt of max features, and the num of features to split.

```
rf_hyperparams = {
    'randomforestclassifier__n_estimators': [10, 15, 25, 35, 50, 60],
    'randomforestclassifier__max_features': ['auto', 'sqrt', .33],
    'randomforestclassifier__max_depth': [1, 3, 5, 7]
}
```

In GBDT, n_estimators are the maximum number of iterations of the weak learner. In general, n_estimators was too small, and it is easy to underfit. Generally, a moderate value is selected as 100. Learning_rate is the weight reduction coefficient ν of each weak learner ν, also known as step size. We used the learning_rate and the n_estimaters to determine the fitting effect of the algorithm. Besides, when the model has a large number of samples and many features, we limited it as 5.

```
gb_hyperparams = {
    'gradientboostingclassifier__n_estimators': [25, 50, 80, 100, 200],
    'gradientboostingclassifier__learning_rate': [.05, .1, .2,.5,1],
    'gradientboostingclassifier__max_depth': [1,3,5]
}
```

In LDA, we set default value to do prediction.

In KNN, we set hyper-parameter from 1-5, which would decrease approximate error and also avoid overfitting.

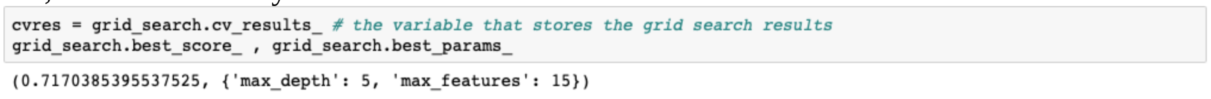
```
#KNN
from sklearn import neighbors
from sklearn.model_selection import GridSearchCV
param_grid = {'n_neighbors':[1,2,3,4,5]}
KNN_clf = neighbors.KNeighborsClassifier()
grid_search = GridSearchCV(KNN_clf, param_grid, cv=5, scoring='accuracy')
grid_search.fit(heloc_features, heloc_labels)

cvres = grid_search.cv_results_

grid_search.best_params_,grid_search.best_score_
```

```
({'n_neighbors': 5}, 0.6765973630831643)
```

We used grid search to find the optimal combination of hyperparameter values with clear results. After completing the grid search on the training data set, we can get the optimal value of the index we specified in the scoring parameter through the best_score_, and the specific parameter information to be tuned can be obtained through the best_params_. A as you can see, the final accuracy will look like below.



```
cvres = grid_search.cv_results_ # the variable that stores the grid search results
grid_search.best_score_ , grid_search.best_params_
```

```
(0.7170385395537525, {'max_depth': 5, 'max_features': 15})
```

5. Evaluate Models

This section will summarize the results of the prediction performance measures for all of the classification algorithms covered in this project. The metrics we used to evaluate our models are accuracy score, recall score, F1 score and precision score.

- Accuracy

Accuracy is the ratio of the number of correct predictions to the total number of input samples. Ideally, the higher the accuracy rate, the better the model.

- Recall

Recall, also refers to the sensitivity, represents the ratio of the number of positive samples correctly identified by the model to the total number of positive samples. In general, the higher Recall is, the more positive samples are predicted correctly by the model, and the better the effect of the model is. Ideally, the higher the recall rate, the better the model.

- F1 score

F1 score, also known as Balanced Score, is defined as the harmonic mean of the accuracy and recall rate. The range of the F1 score is [0,1]. It tells you how precise your classifier is and also how robust it is. Generally, the higher the F1, means the more instances are classified correctly.

- Precision score

Precision refers to the number of true positive results divided by the number of all positive results being predicted by the classifier.

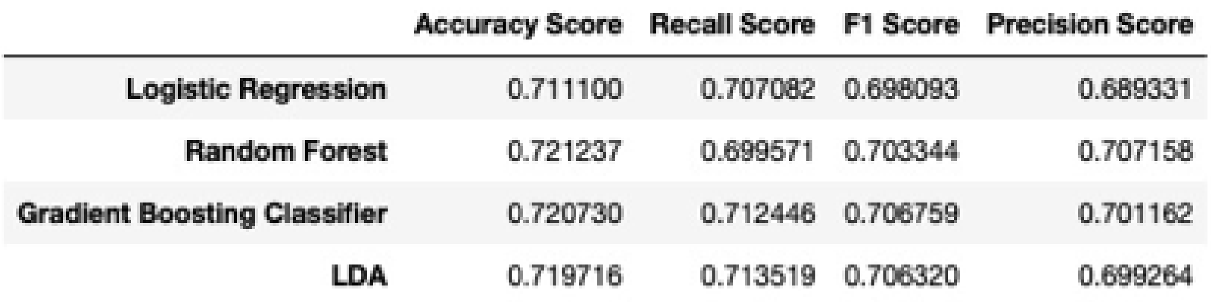
- AUC (Area Under Curve)

AUC is defined as the area under the ROC curve, range is [0.5,1]. The AUC value is used as the evaluation criterion because in many cases the ROC curve does not clearly indicate which classifier is more effective, while as a value, the classifier with a larger AUC is more effective.

Predictive Models Results Summary

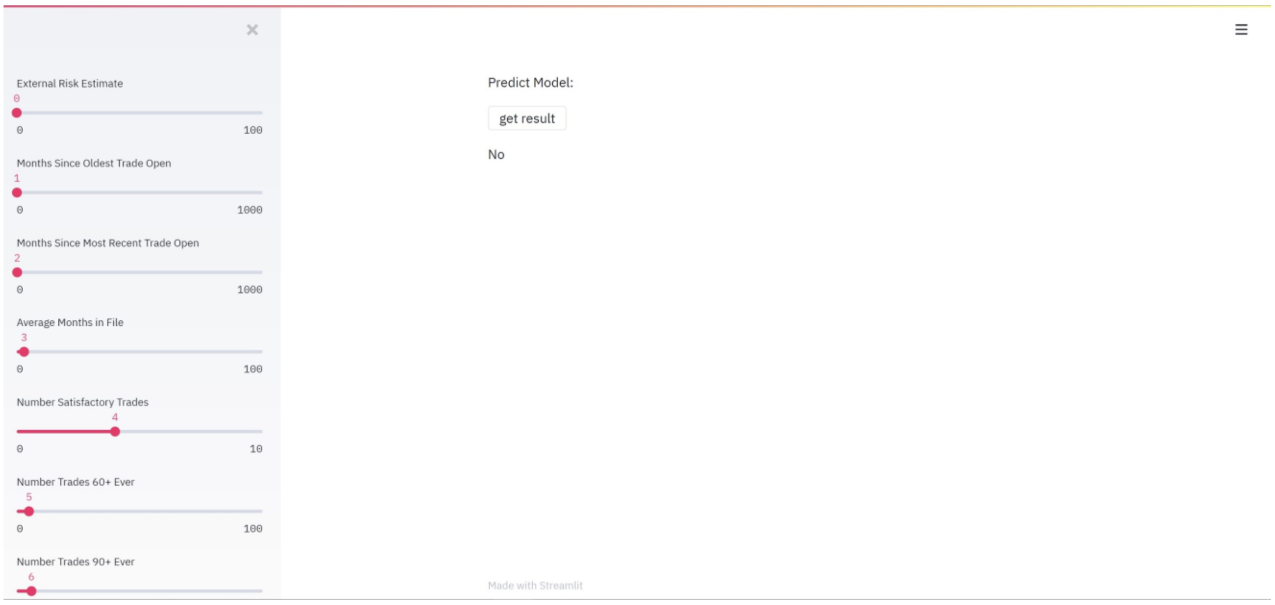
We built up 6 models, they are respectively logistic regression, random forest, gradient boosting classifier, LDA, KNN and decision tree. All the metrics performance of the first four classifiers are shown below. We choose accuracy as our final measurement of model performance.

As we can see, the random forest's accuracy is the highest which is 0.721237. (the accuracy we do individually for KNN is 0.67659736, and decision tree is 0.7170385). Thus, we select random forest is the best model we selected.



	Accuracy Score	Recall Score	F1 Score	Precision Score
Logistic Regression	0.711100	0.707082	0.698093	0.689331
Random Forest	0.721237	0.699571	0.703344	0.707158
Gradient Boosting Classifier	0.720730	0.712446	0.706759	0.701162
LDA	0.719716	0.713519	0.706320	0.699264

6. Interface



External Risk Estimate
0
0 100

Months Since Oldest Trade Open
1
0 1000

Months Since Most Recent Trade Open
2
0 1000

Average Months in File
3
0 100

Number Satisfactory Trades
4
0 10

Number Trades 60+ Ever
5
0 100

Number Trades 90+ Ever
6

Predict Model:

get result

No

Made with Streamlit

We made user interface simple and direct. There is only one page for user interface. On the left side is user input part. We make all input as slide bars because we want to limit users' input as number and within a specific range. By using slide bars, either user or system will not concern about input data type might cause error. On the right side is output window. By clicking get result, user can see based on input, should applicant's profile get approved or not.

7. Conclusion

By analyzing the HELOC data, we constructed prediction models and reached the conclusion that random forest is the best for financial institutions to use, in order to make better decision making. We also thrived to construct a user-friendly interface, enabling user interaction and presenting accurate output.