Eugene Kim
Machine Learning Project
September 12, 2021

Definition

Project Overview

Lending and borrowing money have played a crucial role in the economic rise of modern day society. Loans provide time and opportunity for people to start up businesses, finance school tuitions or help pay off large purchases. With an increased demand for loans, it is important for both the investors and borrowers to know whether or not they meet the credit policy of lending companies based on their given information.

In this project, I trained a model to determine if a potential borrower's credit history fulfills LendingClub's credit policy. LendingClub is a popular financial platform that allows people to lend or borrow money. This model is trained on a dataset that contains thirteen different features (twelve are used) with one binary label. In regards to the learning algorithm, the Random Forest Classifier produced the most accurate results.

Problem Statement

The tasks to complete this project involves:

1. Exploratory Data Analysis: Review authenticity of dataset, Examine correlations between features, Check for unusual patterns
2. Data Preprocessing: Check for inconsistent and missing data values, Stratify dataset using relevant feature, Convert all data into numerical values, Apply feature scaling to data
3. Model Training and Testing: Select models to train with dataset, Use cross validation to test the models, Tune hyperparameters to decrease overfitting
4. Evaluation: Choose scoring method and determine model with the highest score, Create visuals of scores, Reanalyze project for improvements

The final product is expected to produce accurate and reliable results based on one's loaning credentials.

Dataset

The dataset used for this project originates from LendingClub between 2007-2010. It is provided as a csv file in https://www.kaggle.com/itssuru/loan-data. It is composed of 9578 instances with no missing values. The dataset consists of thirteen features with one binary classification label. Within the thirteen features, there are twelve numerical and one categorical feature. The original dataset is meant to have 'not.fully.paid' as the label. However, I changed the label to 'credit.policy' to avoid repetitive projects. Thus, 'not.fully.paid' is excluded from this project.

	column	data_type	description
0	credit.policy	int64	if the customer meets the credit underwriting ...
1	purpose	object	the purpose of the loan
2	int.rate	float64	the interest rate of the loan, as a proportion
3	installment	float64	the monthly installments owed by the borrower ...
4	log.annual.inc	float64	The natural log of the self-reported annual in...
5	dti	float64	the debt-to-income ratio of the borrower (amou...
6	fico	int64	The FICO credit score of the borrower
7	days.with.cr.line	float64	the number of days the borrower has had a cred...
8	revol.bal	int64	the borrower's revolving balance (amount unpai...
9	revol.util	float64	the borrower's revolving line utilization rate...
10	inq.last.6mths	int64	the borrower's number of inquiries by creditor...
11	delinq.2yrs	int64	the number of times the borrower had been 30+ ...
12	pub.rec	int64	the borrower's number of derogatory public rec...
13	not.fully.paid	int64	not fully paid

Fig. 1 A data dictionary showing each column with the respective data type and description

Metrics

The label for this dataset is a standard binary classification. When deciding the scoring metric for binary classification, it is important to consider the risks of false positives and false negatives. In

the case of false positives, the investors will be at risk of lending money to borrowers who do not meet the credit policy. In the case of false negatives, the potential borrowers will lose the opportunity of receiving loans. Depending on one's status of an investor or a borrower, false positives and false negatives can cause great problems.

To account for both false positives and negatives, this project will use the F1 scoring metric to determine the model's performance. The F1 Score takes the average scores of precision and recall. Precision divides the true positives with both true and false positives. If the precision is low, the model incorrectly indicates that borrowers meet the credit policy. Recall divides the true positives with both true positives and false negatives. If the recall is low, the model incorrectly indicates that borrowers failed to meet the credit policy. The equations are shown below.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

TP = True positive

TN = True negative

FP = False positive

FN = False negative

Analysis

Data Exploration

Reviewing Authenticity: The dataset originates from LendingClub which is a commonly used financial platform to borrow or invest money. LendingClub was founded in 2006 and claims to have over 3 million members and over 60 million dollars borrowed. Furthermore, this platform has an average of 4.83 star reviews and holds the Bazaarvoice Authentic Reviews Trust Mark. This information is required to authenticate the data from LendingClub.

Inspecting Unusual Patterns: Unusual patterns may include both outliers and strange trends within each feature. Excluding the binary features, three features ('revol.bal', 'inq.last.6mths', 'delinq.2yrs') are strongly skewed to the right indicating a high chance of outliers. There are two

unusual patterns which come from the ‘dti’ and ‘revol.util’ graphs. Both graphs have a high spike at the zero x axis with a sudden dip afterwards. This indicates that it is common for people to have no debt and no spending on their credit cards when receiving loans.

Examining Correlations: The correlation between the label and the thirteen features helps us get a sense of each feature’s significance. The correlation matrix reveals a high FICO score reflects the highest chance of meeting the requirements. Most features have a negative correlation with the label (‘credit.policy’). This also reveals that loaning companies prioritize risk assessment when dealing with potential borrowers. A logistic regression plot is helpful when visualizing correlations with binary attributes.

```
credit.policy        1.000000
fico                 0.348319
days.with.cr.line    0.099026
installment          0.058770
log.annual.inc       0.034906
pub.rec             -0.054243
delinq.2yrs         -0.076318
dti                 -0.090901
revol.util          -0.104095
revol.bal           -0.187518
int.rate            -0.294089
inq.last.6mths      -0.535511
Name: credit.policy, dtype: float64
```

Fig 2 Correlation matrix between features and label

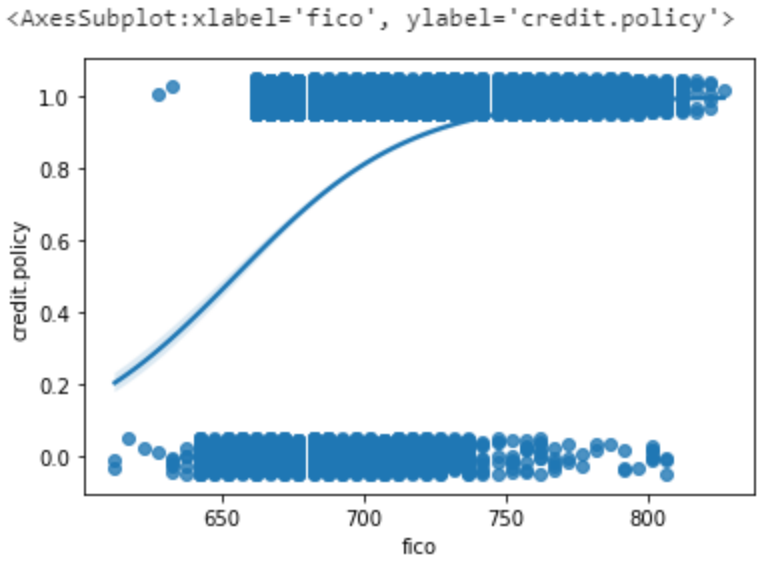

Fig 3 Logistic regression graph comparing ‘fico’ and ‘credit.policy’

Algorithms and Techniques

The necessary model for this project falls into the supervised learning category due to foreknowledge of labeled data. Within this category, a classification algorithm is required to determine whether the output is a 1 or 0. Thus, the project uses two different classifiers, logistic regression and random forest. These two classifiers are the most commonly applied when dealing with binary labels.

Logistic Regression Classifier: The logistic regression classifier finds linear relationships between the attributes and label to form a nonlinear, Sigmoid function. This classifier is a simple and effective method to train the model without having memory or time problems. However, the linearity between the independent and dependent variables prevent logistic regression classifiers from performing well on complex data.

Random Forest Classifier: The random forest classifier consists of multiple decision trees with independent predictions based on randomized observations and features. This generates accurate results without overfitting the data. The downside occurs when the number of decision trees and features within these decision trees are large. The memory and time for this model become expensive.

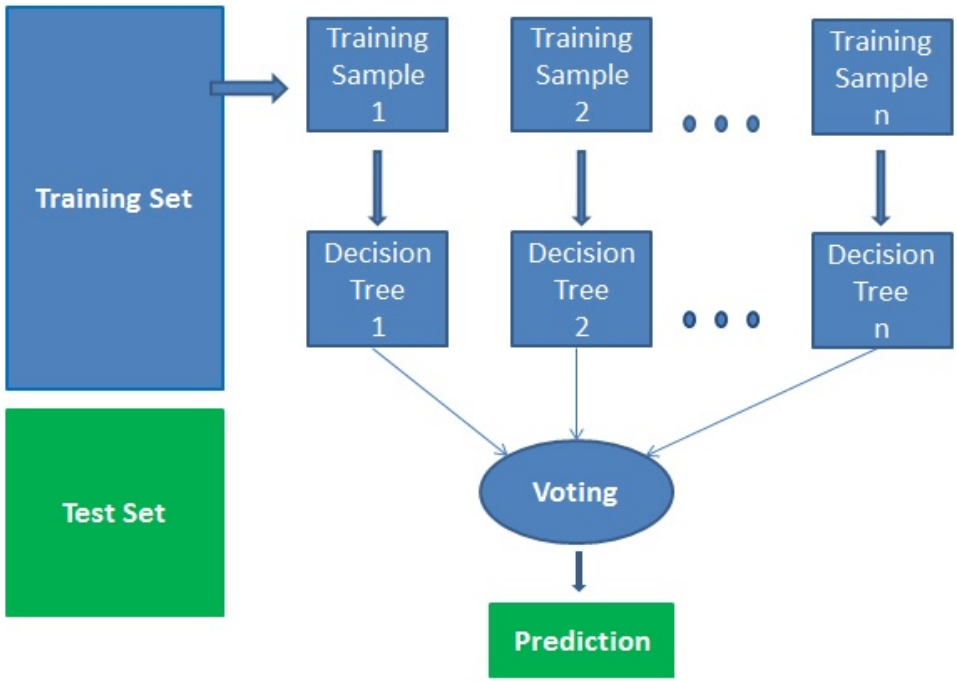

Fig 4 The process of random forest classifier making predictions

The training process requires cross validation to accurately determine the scores of each model to minimize overfitting. This project uses 10 randomized sets from the training data to fulfill cross validation. The official score of the models is the average of all sets combined.

Benchmark

The benchmark for this project is an arbitrary scale from 0 (ineffective) to 100 (very effective). This scale directly correlates a model's F1 score. Although there are various metrics, the F1 score holds the largest weight as it takes into account both precision and recall. Because this project uses a different label from past projects, there are no comparisons to be made for the benchmark.

Ineffective: $0 \leq 49$ (F1 score: $0.00 \leq 0.49$)
Slightly effective: $50 \leq 69$ (F1 score: $0.50 \leq 0.69$)
Effective: $70 \leq 89$ (F1 score: $0.70 \leq 0.89$)
Very Effective: $90 \leq 100$ (F1 score: $0.90 \leq 1.00$)

Methodology

Data Preprocessing

The preprocessing steps to prepare the data are shown below:

1. Stratify the data
2. Split data into training and testing sets
3. Select pre-existing transformers or create custom transformers
4. Create a complete pipeline

Stratify the data: It is important to randomly stratify the data to get equal representation in the population. I chose the 'fico' feature to stratify the data because it has a high positive correlation with the label. The feature is split into five sections depending on the respective FICO score (0-675,676-700,701-725,726-750,751-800).

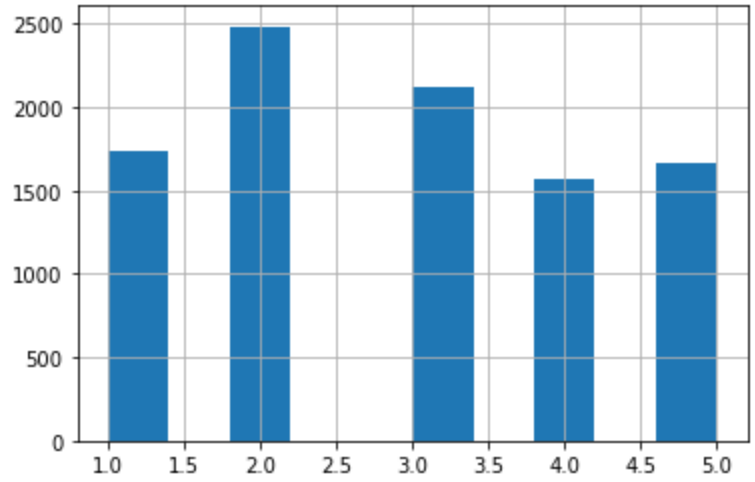

Fig 5 The randomly stratified data measured by count

Split data into training and testing sets: The full dataset is split into a training and testing set with a 8:2 ratio. Within these sets, the features and labels are separated. The total number of instances in the training set comes out to be 7662 and the testing set comes out to be 1916.

Select pre-existing transformers or create custom transformers: There are three transformers used in this project.

1. **SimpleImputer**: An imputation transformer is used to take care of missing values within the dataset. The strategy is set to the median of each feature. This transformer is applied to all the features.
2. **OneHotEncoder**: An encoder for categorical features is also used to transform strings into numerical values. This encoder is only applied to the 'purpose' feature.
3. **Custom Outlier Detector**: This custom transformer detects outliers by evaluating the interquartile range. It automatically changes the value of an outlier into 'n/a'. This transformer is applied to ['int.rate', 'installment', 'log.annual.inc', 'dti', 'fico', 'days.with.cr.line'].

The SimpleImputer works well with the custom outlier detector since all outliers changed into 'n/a' turns into the median.

Create a complete pipeline: Feature scaling is another step required to create a complete pipeline. This project uses standardization to have a common scale with all the features. Once this is done, separate pipelines are created and then combined together to form one complete pipeline.The categorical attributes will initially have a different pipeline than the rest. Not all numerical attributes go through the outlier transformer because removing too much data can decrease the model's performance.

Implementation

Two models, the logistic regression and random forest classifiers, are trained and evaluated according to their performance. The one with the better results is then used on the test set.

These are the steps for the implementation process:

1. **sklearn.fit()**: Train each model using the preprocessed data described above. This learning phase requires fitting both the features and labels.
2. **sklearn.predict()**: Predict values of either the training set or the testing set depending on desired output.
3. **Calculate score**: Calculate the F1 score using cross validation and compare the score to the benchmark scale.
4. **Tune hyperparameters**: Test and choose hyperparameters using grid search.

When predicting the training set, the original labels can be compared with the predicted values to evaluate a model's initial performance. Certain techniques such as hyperparameter tuning can be applied to increase the performance. When predicting the testing set, these techniques should not be done as it risks overfitting the data.

Refinement

Before tuning the hyperparameters, the F1 score of the logistic regression model was approximately 0.941672 while the F1 score of the random forest model was approximately 0.992735. Thus, both models received 'Very Effective' (F1 scores of 0.90-1.00) as their scores.

When tuning the hyperparameters of the random forest classifier, I chose to manipulate 'n_estimators' and 'max_features'. The best parameter for 'n_estimators' from the chosen values of [100, 150, 200] came out to be 150. The best parameter for 'max_features' from the chosen values of ['sqrt', 'log2'] came out to be 'log2'.

The final F1 score from the training set was approximately 0.992737.

Results

Model Evaluation and Validation

The model with the best performance came out to be the random forest classifier (‘n_estimators’: 150, ‘max_features’: ‘log2’).

Average scores of the logistic regression classifier model applied on the training set

- F1 score: 0.9417
- Precision: 0.9185
- Recall: 0.9661
- Accuracy: 0.9037
- Benchmark score: Very Effective

Average scores of the random forest classifier model applied on the training set

- F1 score: 0.9927
- Precision: 0.9889
- Recall: 0.9969
- Accuracy: 0.9877
- Benchmark score: Very Effective

Average score of the random forest classifier model applied on the test set

- F1 score: 0.9907
- Precision: 0.9823
- Recall: 0.9987
- Accuracy: 0.9854
- Benchmark score: Very Effective

In order to lower the chance of overfitting, cross validation is applied to all the scores. K-fold cross validation splits the dataset into ten different subsets and these subsets receive independent scores. The mean of all the scores is the final score used for evaluation. Based on the results above, the training model did not overfit the data.

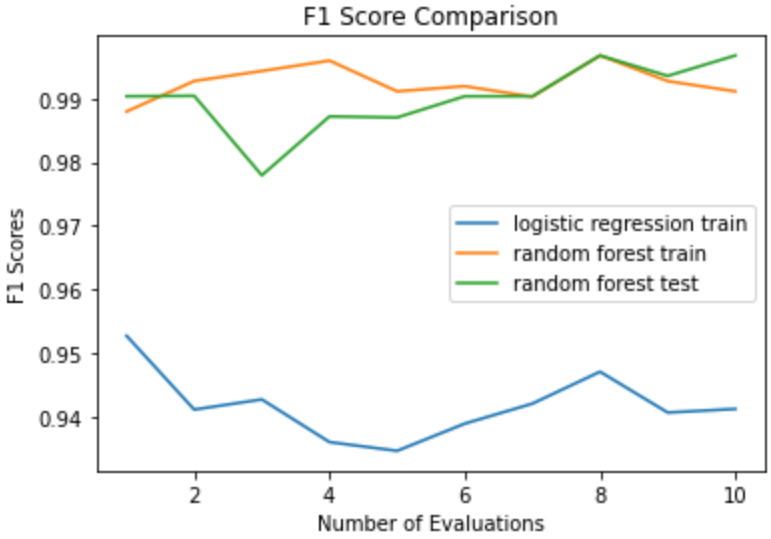

Fig 6 F1 Score comparisons of models using either the training or testing set

Justification

The results of the project indicate that the model is reliable enough to help both investors and borrowers have a better idea of their current credit credentials. If the credit policy is not met, this can raise a red flag to the investors indicating that they should proceed with more caution. For the borrowers, it can help them understand and raise certain scores of their credentials before asking for a loan. This model is beneficial for both sides of the loaning process.

The random forest model can help reduce time and cost for investors as well. When there are thousands of applicants, the model can reduce the large number into a manageable group. Although the model received a high F1 score, there is still a slim chance of wrong predictions. Therefore, when dealing with a small group of potential borrowers, it may be more effective to sort the applicants by hand instead of using the model.

Conclusion

Reflection

The project's summary is listed down below

1. Find reliable dataset with enough instances and features

2. Analyze data by finding patterns and correlations
3. Create a scoring metric for the models
4. Convert dataset using transformers
5. Select and train model with the preprocessed data
6. Refine model by tuning hyperparameters
7. Apply the best model on the test set
8. Analyze results

I found steps three and four to be the most difficult parts of the project. For step three, the problem arose when finding similar projects to this one. There were no similar projects because I changed the label from ‘not.fully.paid’ to ‘credit.policy’. This made it harder to create a proper benchmark for my results. Step four was also difficult due to the time and effort required understanding sklearn’s methodology for transformers and pipelines. All in all, this project was a great learning experience for the start of my data science journey.

Improvement

To achieve the optimal results for this project, the preprocessing data and model refinement needs to be improved. The following are a list of improvements to be made in the future.

- ❖ Test more classifiers on dataset
- ❖ Find dataset with more instances
- ❖ Tune more parameters for each model
- ❖ Create more forms of outlier detectors (z-score, isolation forest, DBSCAN)

The room for improvements is not significant because the current model’s F1 score is already around 0.99. However, the F1 score is still expected to increase when applying the list of improvements above.