Machine Learning Engineer Nanodegree

Carl Hoenig
October 31st 2018

Capstone Project

Predicting if a loan gets funded using several algorithms

1. Definition

1.1 Project Overview

This project covers two aspects of problem solving using machine learning. What is the question/problem and what tools do we use to answer the question or solve the problem? Let's start with the problem. I want to use an algorithm to predict whether a loan application will be funded, partially funded or not funded at all.
It is important to define the particular problem but, it is equally important to find the right tool to solve the particular problem. Do we use a a simple logistic regression or a complicated and elaborated Deep Neural Network?
We have had the chance to use and learn many algorithms in the Machine Learning Nanodegree. That there are so many algorithms, could very much indicate that there is no "One size fits all" algorithm. So besides answering / solving the problem "Predicting if a loan gets funded", I will compare the performance of several algorithms to see which one performs best for this particular domain and problem.

1.1 Problem Statement and Domain Description

Kiva is a San Francisco based NGO that provides crowd financing for microloans to low income farmers, entrepreneurs and individuals in primarily developing countries. The size of the loans is typically a couple of hundred dollars and they are used to finance a variety of activities such as buying livestock, means of production, goods, education etc.
Kiva loans are, compared to loans in developed countries, very small but the impact that they have on the living conditions of the borrowers can be significant. Predicting if a loan gets funded is therefore critical. If we could identify loans that will not get funded we could give the borrower the chance to revise the loan application and resubmit successfully the application.

The analysis and prediction is based on the Kiva dataset that can be found under the Kaggle Competition "Data Science for Good: Kiva Crowdfunding". The dataset can be downloaded as a csv file under the following link:
(https://www.kaggle.com/kiva/data-science-for-good-kiva-crowdfunding/downloads/kiva_loans.csv)
It consists of 670.000 rows, one for each loan and 20 columns that will give us the features. The main goal of this project is to predict discrete class labels so we are dealing with Classification. Throughout the project I will work with three distinct classes, which are: funded loans,partially funded loans and non funded loans. As we will see later the biggest class is composed of fully funded loans. They make more than 92 % of all loan applications. Our smallest class makes approximately 0.5 % of all loan applications.
The chosen algorithms will have to deal with these imbalances. In other words the algorithms will have to able to find the needle (non funded loans) in the haystack

The problem will be approached from two directions. The first approach will be using standard Sklearn algorithms and the second approach will be using a Keras Neural Network.
As for the class imbalance I will use two techniques to try to improve the results if the chosen algorithms fails. The two techniques will be downsampling the biggest class an the second will consist in augmenting the two smaller classes using the imblearn SMOTE

1.2 Metrics

One thing is sure. We are dealing with classification and the outcome of a loan application gives us the three possible classes that are funded loan, non funded loan and partially funded loan.

Due to the to impact that Kiva loans have on the borrowers life we want to have a high number of correct classifications while at the same time reduce the number of misclassifications. The prediction/outcome combination can result in consequences of different magnitude or gravity:

Prediction	Real Outcome	Consequence
Funded	Non Funded	The borrower will think everything is OK and will not revise his application. The borrowers project will not receive funding and die.
Funded	Partially funded	The borrower will think that his project will be fully funded and will not revise his application. The borrower will receive less funding.
Partially funded	Non funded	The borrower might improve the application. If he doesn't he will not get funding for his project.
Partially funded	Funded	The borrower might improve the application. But even if he doesn't he will receive the full amount.
Non funded	Funded	The borrower will feel that he has to improve his application. But, even if the borrower does nothing he will receive the funds
Non funded	Partially funded	The borrower will feel that he has to improve his application.Even if he doesn't there still will be some funding

I only considered situations where the outcome is different from the prediction. It is imperative to reduce the number of false positives and negatives. In terms of a confusion matrix the perfect model would have all the values in the diagonal. Our metrics will be precision, recall and f beta score.

In order to understand our metrics we will have to look how the predictions match the actual values. We have 4 possible results for the combinations prediction and actual value.

Imagine we have two possible states A and B (B beeing not A ie. the opposite value)

True positives: Our model predicts A and the actual value is A
True negatives: The value we predicts B and the actual value is B

False positives: Our model predicts A but the actual value is B
False Negatives: Our model predicts B but the actual value is A

Our total actual values of A will be composed of True Positives and False Negatives

For our modell I will use the following metrics:

Recall: True positive / (True positives + False Negatives)
Recall gives us the the ratio of the correct predicted members of a class and the total members of a class.
Precision: True positive / (True positives + True Negatives)
Gives us the ratio of the between the correct predicted members of a class and the sum of all correct predictions

F1score: Precision * Recall / Precision+Recall
In order to get the best performing model we want to balance precision and recall. We can combine precision and recall with the harmonic mean from both, called the traditional F-measure or balanced F-score.

1.3 File structure

One of the extra takeaways of this project was the painful epiphany that computing power and problem solving ambition can be two opposing forces. As this project was done on a computer with a rather slow processor I had to divide the problem into three separate Notebooks:

Part 1 Sklearn.ipynb: This notebook includes data description and visualisation and the implementation of several Sklearn algorithms. The notebook includes also a general benchmark

Part 2 Keras.ipynb: The second notebook includes the implementation and fine tuning of a Keras Neural Network

Part 3 Dealing with imbalances and Conclusion.ipynb:
The third part will try to address the problem of class imbalance from two different directions, data augmentation, where I will use SMOTE to increase the two smallest classes with synthetic data points that are similar to their nearest neighbours, and data reduction where I reduce the size of the biggest class in order to give more weight to the two smaller classes.

Note: Due to the advantage that the jupyter notebooks allows code, output and text I have included some more descriptions as text blocks alongside with the code. Where necessary I have added/copied some parts of the findings from this rapport directly into the notebooks.

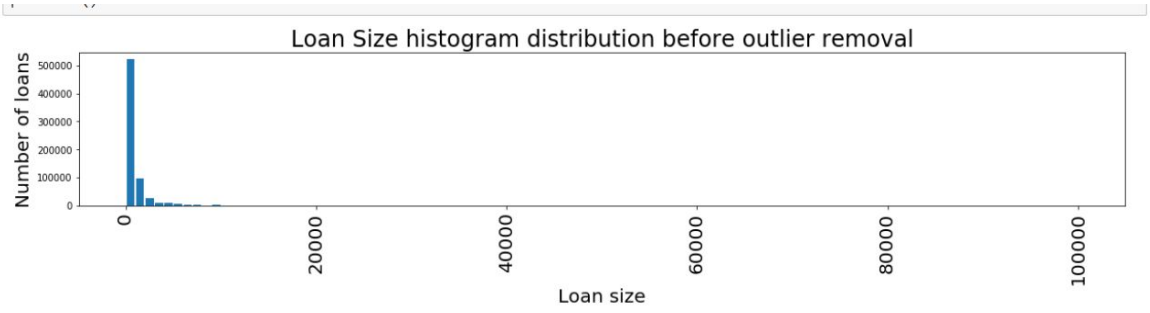
2. Analysis

2.1 Data Exploration

If we start with the premise that lenders evaluate a loan application rationally than we could say that a non funded loans “lacks” something or that it somehow stands out / is different from the rest.
That means that we have to analyze nans and outliers and not treat them as distortions but as factors that could explain the outcome of an application.

Our dataset has an attribute/feature called use. This field gives the borrower the possibility to describe in detail what the loan is going to be used for. When we compare the frequency of nans for this attribute we get that non funded loans make 3.09 % of all loans with nans, but remember that the non funded loans roughly make 0.5 % of the general population. The same happens to the partially funded loans that make up 13.16 % of all loans with nans but only 7.21 % of the general population. If we think about it, that makes sense. Would you lend money to someone who is not telling you what the purpose of the loan is?

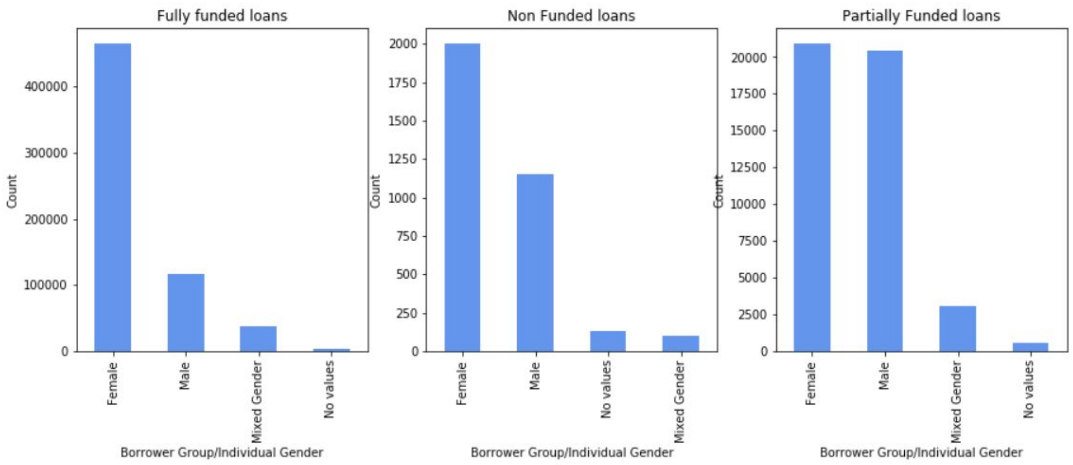
The other aspect that showed some patterns were outliers. The distribution of the loan size is really skewed to the left with some really big outliers.



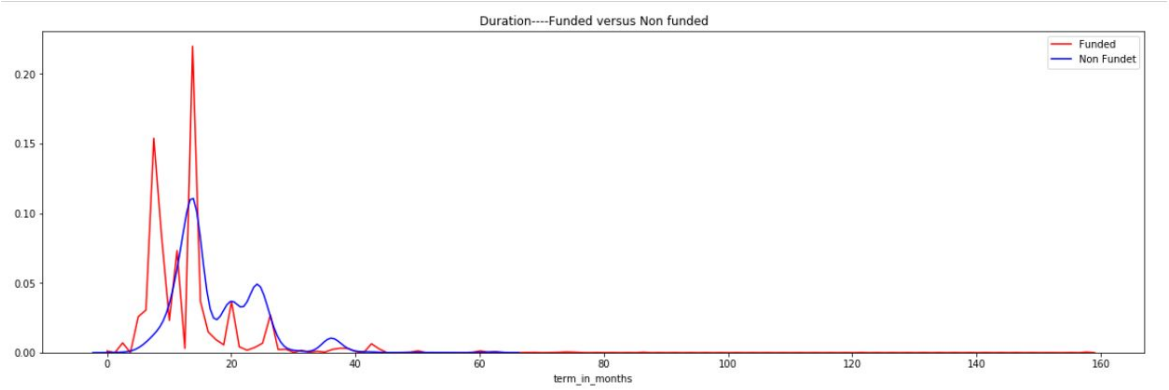
When I tried to remove the outliers based on the interquartile range I noticed that the non funded loans made up 1.1570215519777225 % off all outliers. That is more than double when compared their representation within the general population.

2.2 Exploratory Visualization

I used visualization to see if I could see patterns for the three classes. The result was mixed. Some features showed distinct patterns while others were not as clear.



The graphs above show the gender distribution for the three classes. Note that that men are higher represented for the non funded and partially funded when compared with fully funded loans. We can see clear patterns here. The chances of getting a loan or getting the full amount are lower when you are a man.
Not all features show a clear picture between the outcome and the feature. If we look at the duration of a loan:



We can see that it's more challenging to see a clear pattern. The same goes for partially funded loans versus fully funded.

2,3 Algorithms and Techniques

In order to find the best tool I will compare four sklearn algorithms with a keras neural network, the part twas to design a pipeline with two steps. The first will be rescaling the values of the features in order to give the values attributes the same importance. The mean for the loan

amount is 842 USD but the duration (terms in months) has a mean of 13. If we didn't rescale the loan amount would see more important than the duration.
I will try the following sklearn algorithms: GaussianNB , MLPClassifier,RandomForestClassifier, and AdaBoostClassifier and based on the fbscore for all classes select the most promising.
The four algorithms are described in more detail in the Notebook.

The best algorithm will be fine tuned to improve the performance using RandomSearchCV

In the second part I designed a keras based neural network and fine tuned the model both using GridsearchCV and manual finetung

In the third part of the project I applied two methods to reduce the imbalance of the the two smallest classes.

2.4 Benchmark

To find a benchmark I used the most basic sklearn algorithm, logistic regression and it proved to be quite good in predicting the non funded loans. I got an the following fbscores:

	precision	recall	f1-score	support
full funded	0.95	0.99	0.97	124604
non funded	1.00	0.94	0.97	660
partially funded	0.71	0.25	0.37	8977
avg / total	0.93	0.94	0.93	134241

It showed a surprisingly good result forth biggest class and the most class. It seems that the benchmark performs good in recognizing and separating the two opposite classes but fails in cases that are situated in the middle.

3. Methodology

3.1 Implementation

All three parts of the project follow a three step implementation:

- Preprocessing
- Implementation of the algorithm

Preprocessing:

Remove data that has the answer: The dataset provided includes a feature that is not known at the point in time when the borrower submits the application. The feature is funded amount.
Preprocess Gender: The gender feature contained many unique categories as the loan can be submitted by a single female, male, a group of only males, only females, mixed groups and nans. I reduced the categories to four: female, male, mixed group and nans/No values.
Use: The feature use gives the borrower the possibility to tell more about the intended use of the loan. These feature contained a lot of nans and I want those to be included in the analysis. I therefore created a new feature called description with two values. Description for the cases where the borrower had provided description and

Feature Scaling: There are several features that include numerical values. The range in which these values lie can be quite different so I used MinMaxScaler() to ensure that the values for all features have the same importance.

LabelEncoder: Common for all for the algorithms is the fact that they can't compute strings so the I will have to apply the method Labelencoder to transform the text into numerical values

One hot encoding: Whenever we encode labels they will get different numerical values. The first will become a 1 the second will become a 2 and so on. This has an unintended consequence. The first labels will have small values and the last labels big values. This gives a sort of mathematical order where som former labels will seem more important than others. By converting the numerical values into a series of zeros and one single 1 we remove this artificial importance.

Implementation of the algorithm

Algorithms for Sklearn:

The algorithms used for Sklearn were:

RandomForestClassifier: A random forest is a meta estimator that uses a number of individual decision tree classifiers on various parts(subsamples) of the dataset. The RandomForestClassifier uses averaging to improve the predictive accuracy and keeps at the same time overfitting down. Advantages: Minimizes overfitting, Disadvantages: Computationally expensive.

MLPClassifier Is the implementation of a feedforward neural network. Multilayer perceptrons, are a combination of multiple neurons connected in the form a network that can solve problems when the data is not linearly separable.
Advantages; can learn linear and non-linear models Disadvantages: Complex algorithm, computationally expensive

AdaBoostClassifier which is the short form for Adaptive Boosting is, like the RandomForrestClassifier, a meta-estimator. It combines many weak (simple) learners to create a highly accurate prediction. It uses an iterative process where it changes the sample distribution by modifying the weights attached to each of the instances. The weights of wrongly predicted instances are increased and correctly predicted instances are increased.

Advantages: Like RandomForrest it is less prone to overfitting Disadvantages: AdaBoost can be sensitive to noisy data and outliers.

GaussianNB A classifier that is based on Bayes' Theorem. It assumes the independence among predictors and all of the predictors independently contribute to the probability of a the classification outcome.

Advantages: Fast, very simple and straightforward implementation Disadvantages: The "naive" part. The algorithm makes a very strict assumption that the features are independant.

Architecture for the Neural Network:

The Input Layer: The number of nodes is determined by the number of features

The Output Layer: The number of nodes will be equal to the number of classes, wich in this case it will be three.

The number of hidden layers: The question to ask is whether we need one or more layers. According to The universal approximation theorem single layer neural network can learn anything, when given appropriate parameters. So I will establish only one hidden layer. Source: https://en.wikipedia.org/wiki/Universal_approximation_theorem

The nodes of hidden layer: Deciding the number of nodes for the hidden layer is much more complex, and there is unfortunately no “Golden Rule” as the number will be influenced by the complexity of the problem, the amount of features, the size of the training set.

Activation functions: I will use two different activation functions. The activation function for the output layer will be sigmoid as I am Interested in returning continuous values between 0 and 1 indicating the probability of each of the three outcomes.
The other activation function which will be used for the first and for the hidden layer will be relu as it is more resistant to the vanishing gradient problem.

Dropout In order to reduce overfitting I will include Dropout,

3. 2 Refinement

Algorithm based fine tuning:

I used several techniques to refine the results. I used both algorithm based tuning where I applied GridsearchCV and RamdomsearchCV to find the best performing combination of parameters.

At the same time I used a manual fine tuning for the Neural Network focusing on the learning rate of the optimizer and the number of layers.

According to the Keras documentation (https://keras.io/optimizers/#rmsprop) it is recommended to only modify the learning rate. I applied the changes in two directions i.e. first reducing the learning rate and than increasing the learning rate. The best performance was obtained by increasing the learning rate.

As for the number of nodes in the hidden layer I increased the number until I reached improved results. As increasing the number of nodes has the risk of overfitting I increased the Dropout rate gradually.

Data based approach:
As stated in the first part of this project the classes are very unbalanced. I used two different approaches in order to give the two smallest classes more importance/weight. The two methods

try to solve the problem from two directions. The first one is undersampling meaning that we reduce the biggest class to a level where the smallest classes are similar in size/importance. The second is oversampling where we increase the smallest classes so they can have the same weight as the biggest class.
Undersampling was implemented by randomly removing instances of the class fully funded loans.
Oversampling was done using the SMOTE algorithm. SMOTE stands for Synthetic Minority Over-sampling Technique.
The idea behind the algorithm is to create synthetic data points that are similar to existing data points that share the same characteristics.. By using smote I will increase the size of the two smallest classes so they match the biggest class in size and will therefore remove class imbalance.
Smote is unfortunately not part of Sklearn or Keras and the package has to be installed separately. It can be installed from this site: https://pypi.org/project/imbalanced-learn/
The biggest drawback with smote is computing time as the dataset gets, in this case, multiplied by three.

4. Results

4.1 Model Evaluation and Validation

Algorithm	f1 fully funded	f1 non funded	f1 partially funded
Random forest	0.98	1.00	0.71
Keras	0.98	0.53	0.67
Keras SMOTE	0.96	0.81	0.62
Keras undersampling	0.84	0.71	0.88

The table above shows the results of the approaches I used to solve the original problem that was predicting the outcome of a loan application.
The winning approach was using the Sklearn Randomforest algorithm. It achieved the best results for the class that from the point of vue of the borrower was most critical. The non funded loans were predicted perfectly.
The Keras NN performed well but didn't beat the Randomforest. Resampling the data improved the results for some classes but always at the cost of the other classes.

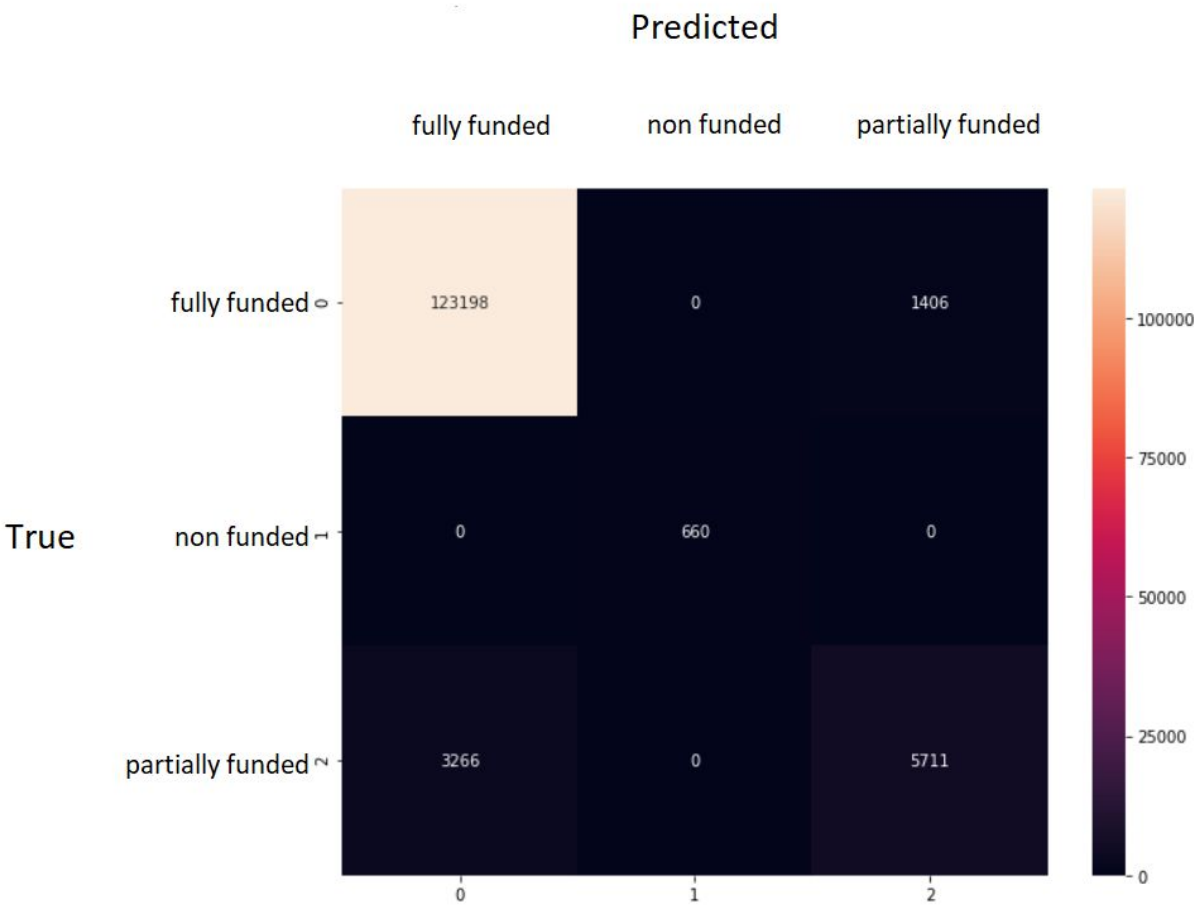
4.2 Justification

First of all, the reason that Keras NN was beaten by the Randomforest is probably due to the fact that neural networks are difficult to design and requiere a great amount of knowledge, experience and craftsmanship. If we use an analogy from the automobile world we could say that a Ferrari is an excellent car but you would not want an inexperienced mechanic to tune the engine.

5. Conclusion

To visualize the point that the random forest algorithm performs very well for the give problem the confusion matrix shows a clear picture.

In the project description I described a series of outcomes that were critical from the borrowers point of vue. The most critical class was composed by the non funded loans. We want not only to predict correctly as many loans as possible but to reduce the misclassifications as much as possible. We don't want to predict a non funded loan as a funded loan as this would lull the borrower into a false sense of security, that everything is OK with the loan. Our algorithm says everything is Ok with the loan application and the borrower would not consider improving the application. In the perfect world all the misclassifications for this type of loan would be 0.



When we look at the confusion matrix and check out the non funded loans we achieved what we wanted Not one of the real/actual non funded loans was predicted as funded or partially funded.

As for the two other classes there are misclassifications. The misclassifications goes both ways between the two classes, partially funded and fully funded. My Random forest algorithm predicted that more than 3000 partially funded loans were fully funded and that 1400 fully funded loans were partially funded. The first prediction/reality outcome is worse as the algorithm is telling the borrower that the loan will be funded when the reality is that he/she will get lower funds.

5.2 Reflection

The needle in the haystack was not the issue: From the human point of vue, predicting the outcome of non funded loans seems a daunting task as they only represent 0.5 % of all loans.

At first glance it seems surprising that even our benchmark was quite good at predicting this particular class. If we think about the two classes funded and non funded as two opposites than it starts to makes sense as instances of this classes have to be further away from each other.

5.3 Improvement

First of of all I would add Principal component analysis to dig deep deeper into the relations between the features an two explain an outcome better.
Given more computing power I would set up a bigger grid for the neural network allowing RandomizedsearchCV not only to find the best parameters but also the best architecture. In other words I would add the number of nodes in the hidden layer as a parameter for the search. This would give a more methodical approach to the question “How many nodes do we need?”