Udacity Machine Learning Nanodegree 2020

Capstone Project Report

Neural-Network based Mortgage Prepayment Model

Yuri Turygin

November 2020

1 Problem Statement

In this project we build a neural network-based prepayment model designed to predict the rates at which mortgage borrowers repay their mortgage loans depending on a variety of factors. These repayment rates are called mortgage prepayment speeds. Let us explain how exactly we calculate them.

When a borrower makes her monthly mortgage payment, a part of the payment goes towards paying off interest and another part goes towards paying off mortgage principal. Each month we know exactly how much principal should be left after a borrower makes her payment. That is actually decided at the time of the mortgage origination and this principal repayment schedule is called mortgage amortization schedule. Suppose, the scheduled remaining principal balance after the N^{th} payment on a mortgage loan is C_N, but a borrower has made a payment higher than the necessary by an amount of p_N, then we define a Single Month Mortality (SMM) rate SMM_N to be

$$SMM_N = \frac{p_N}{C_N}.$$

In other words, we measure prepayment speeds as percentage points of the outstanding principal balance. Note that a higher than scheduled mortgage payment does not decrease the scheduled mortgage principal for the next month, because the extra payments are applied towards the tail of a mortgage. For example, if you had 100 mortgage payments left, but one month you made two mortgage payments instead of one, then all is different is that you don't need to make your 100^{th} mortgage payments anymore. Thus, the above definition makes sense irrespective of prepayments observed in the previous months.

Although the model we are going to build is designed to predict SMM, people rarely discuss prepayment speeds in SMM units. It is much more common to discuss mortgage prepayment speeds using an annualized version of SMM called CPR (Constant Prepayment Rate). SMM and CPR are connected via the following simple formula.

$$CPR = 1 - (1 - SMM)^{12}$$

In other words, CPR shows the percentage of the principal loan balance to be gone in one year if prepayment speeds remain the same every month for a year.

Government Sponsored Enterprises (GSEs), such as Fannie Mae, are making a lot of mortgage prepayment data available on a monthly basis, because that data is required to calculate cashflows on many outstanding mortgage bonds, and therefore, is an essential piece of information for mortgage investors. The data consists of monthly snapshots of characteristics and prepayment speeds of all mortgage pools whose credit risk (risk of borrower's default) is guaranteed by a GSE, in this case by Fannie Mae. Besides the monthly pool prepayment speeds, which we discussed above, the data contains many of the average characteristics of borrowers in a pool. Among the few of those are an average of loan sizes on loans in a pool, an average FICO score, average age of loans, what percentage of loans were originated in a given state, who's servicing the loan, and many more. We will use all of that data to build a model which will make prepayment speeds predictions.

More specifically, our dataset consists of all Fannie Mae guaranteed mortgage pools originated in 2010 and later with each pool containing at least 250 loans. That will make the size of the dataset a lot more manageable.

We will calibrate our model to prepayment experience observed in 2010-2016, and we will then test our model on prepayment experience observed over a period from 2017 through Feb 2020.

2 Evaluation Metrics

The problem we are trying to solve here is a regression type problem. Therefore, it is natural to use root mean squared error (RMSE) as a main performance metrics for our model. We could, for example, just use RMSE on the test set to evaluate our models performances and then pick a model with the lowest RMSE to be the winner. We could do that, but it wouldn't be a very meaningful methodology. The problem is in that the dataset of all the mortgage pools is vast and is composed of many subsectors. A model which performs well on average throughout the whole dataset can still be very far off on some important subsectors and that is generally considered unacceptable in the financial industry. We should at least identify the subsectors which we consider important (and different enough) and examine our model performance on each one of them to see if any subsector performance stands out. ***More rigorously, the performance metrics we will use for our model is the maximum of RMSEs over a set of subsectors in our dataset, which we will identify as relevant.***

For the purpose of this project we will use a rather simple set of subsectors. We will simply break down our mortgage pools dataset by pool coupon. Mortgage pool is a bond and as such it has a coupon. The coupon payments come from borrowers' mortgage payments. Thus, a mortgage pool coupon has to be lower than the average mortgage rate of borrowers in a pool. Typically, the difference between average mortgage rate of borrowers and pool coupon is 50-100 bps, i.e. pool coupon is 50-100 bps lower.

Coupons on mortgage pools come in the increments of 0.5%. There are 2.5% mortgage pools, 3% mortgage pools, 3.5% mortgage pools, etc. That is just how that market trades, which is one of the reasons to break up our data like that. Another reason is that different coupons will have a different borrower's profile. Different borrower's will qualify for different mortgage rates, which depend on many factors such is FICO score, the size of the down payment to name a few. Loans with vastly different mortgage rates will end up in mortgage pools of different coupons. Thus, looking at model performance on pools broken down by coupon will give an insight into model's ability to capture the behavior of different types of borrowers. We our project we will use the following set of coupons to measure model performance [2.5%, 3%, 3.5%, 4%, 4.5%, 5%]. Thus, our final metrics will be calculated as follows.

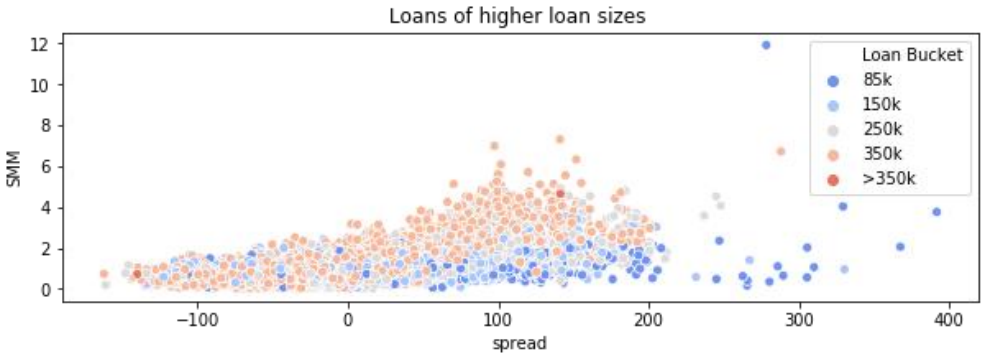
$$RMSE_{model} = \max_{2.5\% \le c \le 5\%} RMSE_{pool\ coupon=c}$$

3 Analysis

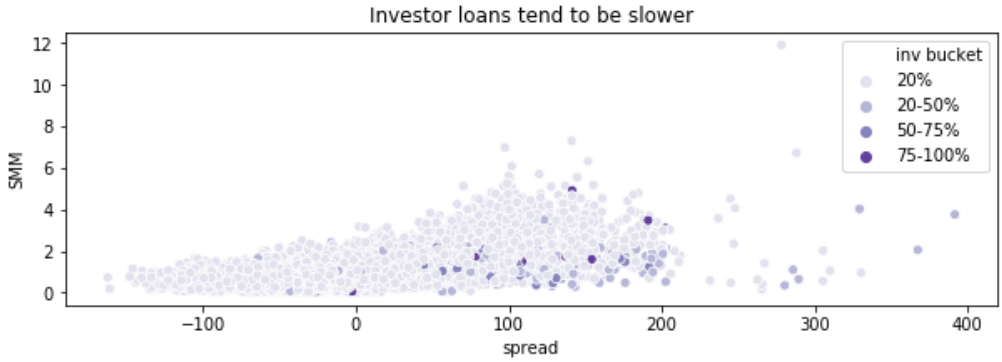
One of the most important factors in mortgage prepayments is refinancing incentive. An incentive to refinance on a loan is defined as (mortgage rate – prevailing mortgage rate available in the market at a time). We measure prepayment incentive in basis points. For example, if borrowers in a pool on average pay a 4% mortgage rate, but the prevailing mortgage rate in the market is 3%, then we say that a pool has 100 bps (=1%) rate incentive. In the data that we use rate incentive is called "spread".

What is "prevailing mortgage rate"? Basically, it a common mortgage rate in the market at any given time. The industry standard is to use Freddie Mac's weekly Primary Mortgage Market Survey (PMMS) rate, which is published on Thursdays at 10am here http://www.freddiemac.com/pmms/. Think of it as an average mortgage rate in the country for that week. We use a 6-weeks average of PMMS rates for 6 weeks prior to an observed prepayment event to calculate the prevailing mortgage rate, and therefore, the rate incentive. The 6 weeks average comes from the fact that it takes on average about 6 weeks to close on a loan.

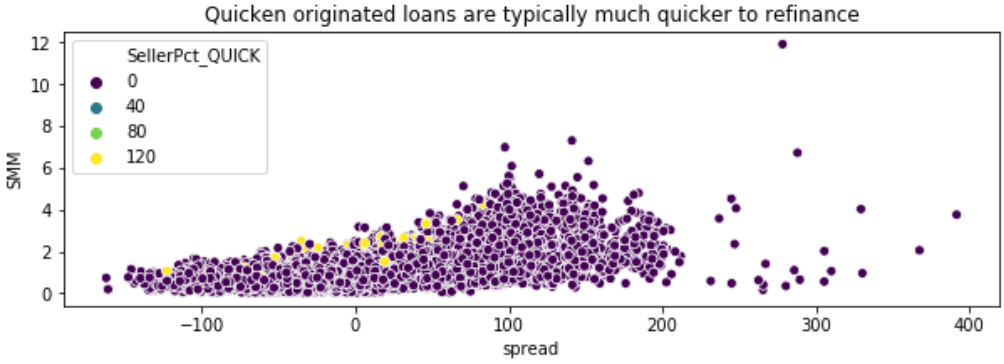
It should be intuitively obvious that the higher the rate incentive the more likely a loan to refinance. This can easily be seen in the data. See below a snapshot of prepayment speeds on 10,000 randomly chosen mortgage pools depending on rate incentive ("spread") and stratified by an average loan sizes of pools. What we see here is twofold – a higher rate incentive corresponds to higher prepayment speeds (SMM), but also that prepayment speeds are faster for loans of higher sizes (at the same level of incentive). It makes sense, because borrowers with bigger mortgages save more dollars by refinancing into a lower rate than borrowers with lower mortgage balances.



Another example for how prepayment speeds depend on mortgage pool characteristics is pools containing investor loans. Investor loans are much more expensive to originate, because GSEs require higher fees to guarantee their credit risk. As a result, investor loans are less likely to refinance than non-investor loans at the same level of rate incentive. This phenomenon can be easily observed on the graph below. We show here how prepayment speeds on pools are different depending on what percentage of loans in a pool are investor loans.



Finally, prepayment speeds on a pool of mortgages can depend on factors other than characteristics of the loans themselves. For example, prepayment speeds can be vastly different depending on who has originated and/or servicing the loans. The prime example of this kind is loans that were originated by Quicken, see the graph below. We show here prepayment speeds on pools with low Quicken concentration of loans vs. pools with higher concentrations of loans originated by Quicken.



Quicken speeds tend to be much faster than similar pools of loans originated by other servicers. As a matter of fact, there are servicers who are “slower” than average and sometimes by a lot. That is why it is important to include servicing composition of a pool as factors to drive the model predictions. Our data includes fields showing what percentage of loans in a pool had been originated by a particular mortgage originator. We have this information for the top 22 industry leading mortgage originators.

It is evident from the discussion above that mortgage prepayments depend on a variety of factors which are many and some of the dependencies are quite complex. It should also be noted that what we are trying to predict here is borrower’s behavior, which can be often be “non-linear” in nature. Therefore, our model has to be very flexible, capable of fitting complex dependencies between many model factors, and therefore, most likely has many parameters.

A typical approach to prepayment modeling is to build a parametric model of the form

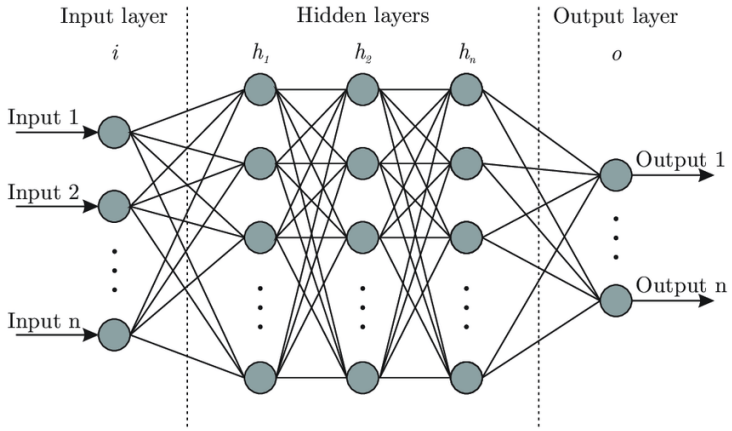
$$\mathrm{SMM(X)} = \mathrm{F_1(X_1)} \times \ldots \times F_{\mathrm{N}}(X_{\mathrm{N}})$$

With each function $F_{\mathrm{i}}(X_{\mathrm{i}})$ modeling that one specific factor X_{i}. The factors are typically fitted sequentially thru least squared errors approximation.

This approach has its advantages and disadvantages. The advantages are that each factor in the model is easily understood and can be tweaked, if necessary. The disadvantages are in that one has to know beforehand what the factors should be an in which order they should be fitted. The later part is usually done through an extensive data exploration phase and for that reason prepayment models sometimes take many months to develop. The other disadvantage is that one has to pick a parametrization function for each factor. For the most part it is fine, and model produces satisfactory results, but then also model "sees" only what you "tell" it to "see".

Those modeling disadvantages is exactly why a neural-network modeling approach seems attractive to us. With this approach we are trying to address the speed of development and model flexibility issues at the expense of model transparency, because neural-networks after all are a kind of a black box.

What is a neural network and why do we think that a neural network approach will work at all here? The general architecture of a neural network is shown on the picture below. It is a mathematical construct consisting of layers and neurons. The circles on the picture below are neurons. The lines connecting the neurons represent rather simple mathematical functions. Neurons are divided into groups – layers of the network. The values of neurons in each layer serve as inputs into these functions which are used to calculate the values of neurons in the next layer. This gets repeated until one runs out of layers. The last layer is called the output layer. The values of neurons in the output layer represent the output of the model. Obviously, the values of neurons in the input layer are the inputs into the model.



Even though the functions connecting neurons in two neighboring layers are rather simple, over the course of several (hidden) layers complexity adds up and the output of a neural network can be a very complex function of inputs into the input layer. That is the trick of neural networks. There is also an efficient and highly parallelizable algorithm called "back propagations", which allows to fit such models to data through minimizing some cost function. In the case of this study, the cost function used is the mean squared error function.

What neural networks do well is they extract new features from the data which helps in making accurate predictions. One can think of neurons in hidden layers as new model features. Going from one hidden layer to the next, the model filters out more and more important and more nuanced features until making the final prediction. That is why very often the number of neurons in each consecutive hidden layer is smaller than in the previous hidden layer. One of the reasons we embarked on this research work was to find a model which would be possible to update quickly as new data comes in. Thus, it is important to pick a model which does not require an extensive feature engineering phase. A feed forward neural network described above is exactly such a model.

4 Benchmark models

We decided to try out two different models as benchmark models for the project – a very simple one and a more nuanced ML model. For the simple model we used a linear regression. Prepayment speeds do not depend on characteristics of borrowers in a pool in a linear way, but it is still interesting to see how well a simple linear model will do predicting residential model prepayments. After all, the output of the model is guaranteed to depend on the model inputs in a way which makes economic sense.

For the mode nuanced model we decided to use is a random forest model. This model builds a series of decision trees randomly picking the features to split a tree at each step, and then averages out the results across all trees to get a random forest model output. These models are interesting to us, because they are "scale invariant", whereas a neural network performs much better (easier to fit) if one scales the model output.

Both benchmark models were fitted on the same train/test datasets as the neural network model.

In the end of this report we will compare the performance metrics of all three model to conclude which one had performed better.

5 Data preprocessing

The data used in fitting of the models in this project is a widely used data provided by Fannie Mae and as such it did not require cleaning and preprocessing.

After having loaded the dataset into the Jupyter notebook, we have examined if data had any missing values and none were missing. We have also examined the ranges of values of each feature and all feature values were in acceptable ranges.

The columns that we thought were appropriate to keep for the model are described in the Jupyter notebook in the section called "Model features selection and defining train/test split".

We had created a "Seasonality" feature which represents the month during which prepayment has occurred. We then turned this categorical "Seasonality" feature into 11 numerical features, and then dropped the "Seasonality" column as redundant.

As a result we were left with dataset appropriate to feed into a mathematical model consisting of 111 columns all of which were numerical.

We then split the data into train and test sets.

For fitting of the benchmark models (linear regression and random forest) we had used the data as is. For fitting of the neural network model we had re-scaled the data to the train set using the MinMaxScaler scaler from sklearn.

6 Implementation

We built a feed-forward neural network consisting of five layers with four hidden layers and one output layer. The network has the following general architecture.

1st hidden layer (512 neurons) , 'relu' activation [0.5 dropout rate used during fitting]

⇓

2nd hidden layer (256 neurons) , 'relu' activation [0.5 dropout rate used during fitting]

⇓

3rd hidden layer (128 neurons) , 'relu' activation [0.5 dropout rate used during fitting]

⇓

4th hidden layer (64 neurons) , 'relu' activation [0.3 dropout rate used during fitting]

⇓

Output layer, 1 neuron, 'relu' activation

In our implementation we have used Keras API into Tensorflow and specifically its Sequential model class. For the fitting of the model we have used the "Adam" optimizer with an 'mse' loss function. To speed up the model calibration process we have used a batch optimization algorithm with a batch size of 1024. The calibration was done over 300 epochs.

All hyper parameters used in the model were chosen through a trial and error process, i.e. through examining the goodness of the model fit on a train set vs. the performance of the model on a test set.

As stated in the 1st section above, for the train set we have used the prepayment experience of 2010 thru 2016, and for the test set we have used the prepayment experience of 2017 thru Feb of 2020.

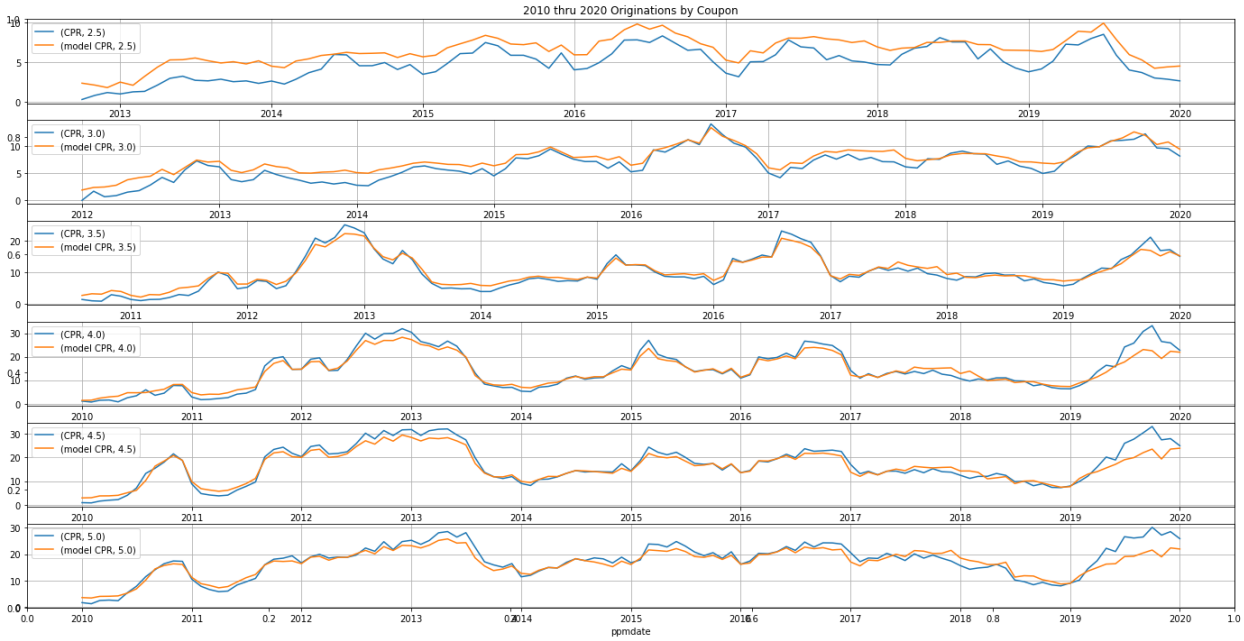
The model was coded up in a Jupyter Notebook and supplied in current repository under a filename of model_fitting.ipynb.

The data used in the model fitting contained 101 features (columns) with only one of them being a categorical variable. That variable is "Seasonality" which is the month of the year during which a prepayment event on a pool has occurred. It turns out that borrowers tend to refinance their mortgages more often in the summer and less often in the winter. Prepayment speeds can differ by 10-15% depending on the time of year all else equal. Hence, "Seasonality" is an important factor in the model. There are 12 months in a year and one way to represent a prepayment month is thru 11 categorical variable, which brought the total number of factors in the model to 111.

7 Results

The neural network model is fitting train set prepayment experience quite nicely and appears to have a good predictive ability into the future as evidenced by the in/out of sample comparisons below. Here we show the model performance on the in sample or test population (2010 thru 2016 prepayment experience) and on the out of sample prepayment experience (2017 – Feb 2020). The pools in the graph below are stratified by the %coupon that is paid to a mortgage investor. They usually appear in the increments of 0.5%. Below we have 2.5%, 3%, 3.5%, 4%, 4.5%, 5% coupon mortgage pools. For a mortgage pool which is paying X% coupon, the average mortgage rate that borrowers are paying is typically 50-100 bps higher. It is natural to stratify our pools by coupon, because that is just how mortgage bond market trades, and also because prepayments speeds on different coupon bonds will be significantly different. Higher coupon bonds will have higher average borrower's mortgage rates and therefore, higher rate incentive. So, prepayment speeds on them will be much higher. Looking at mortgage pools by coupon can give us a valuable insight into model performance and predictive ability.

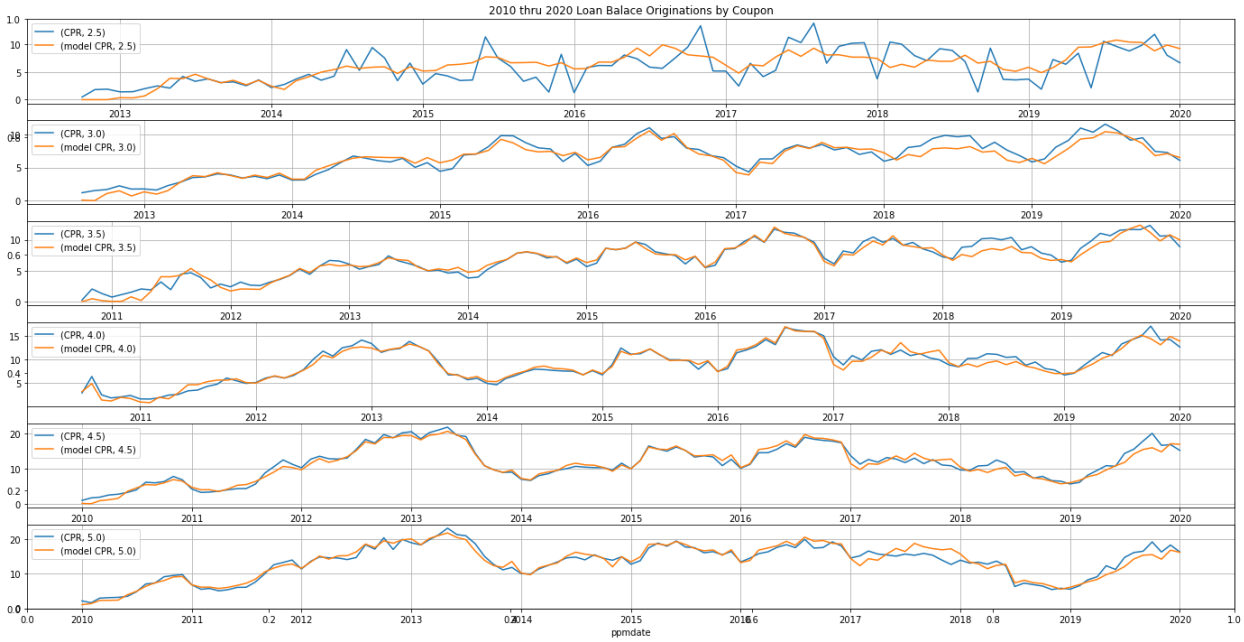
Fig 1.



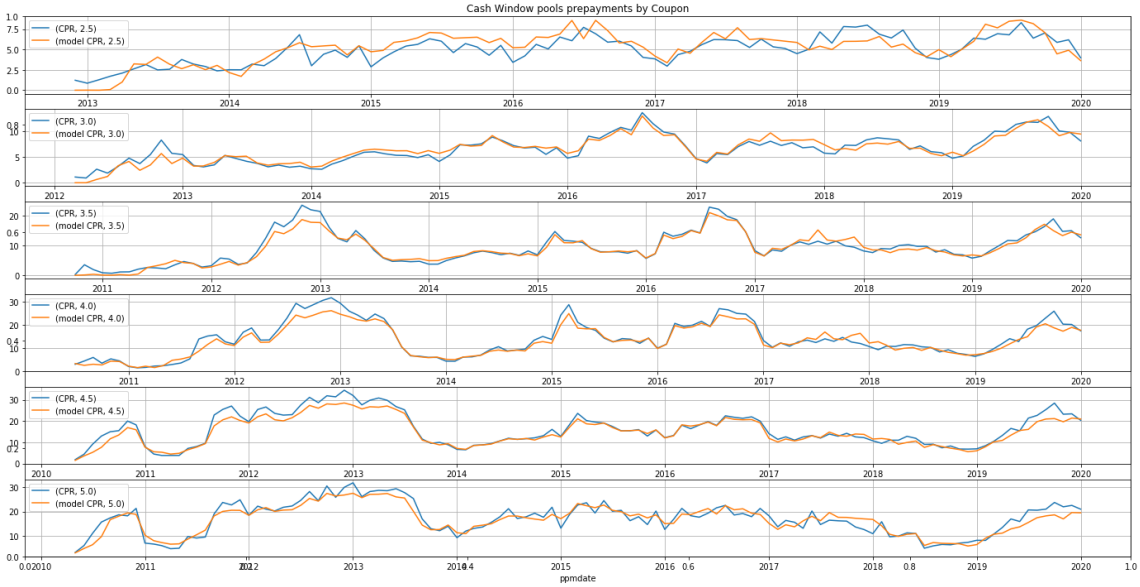
Let us also quantify out performance metrics for the neural network model. The RMSEs by coupon on our test dataset (2017 – Feb 2020 prepayment experience) are we follows, which implies that our model's RMSE = 3.42 CPR.

	NN RMSE (CPR)
Coupon	
2.5	1.66
3.0	1.16
3.5	1.52
4.0	3.42
4.5	3.21
5.0	3.37

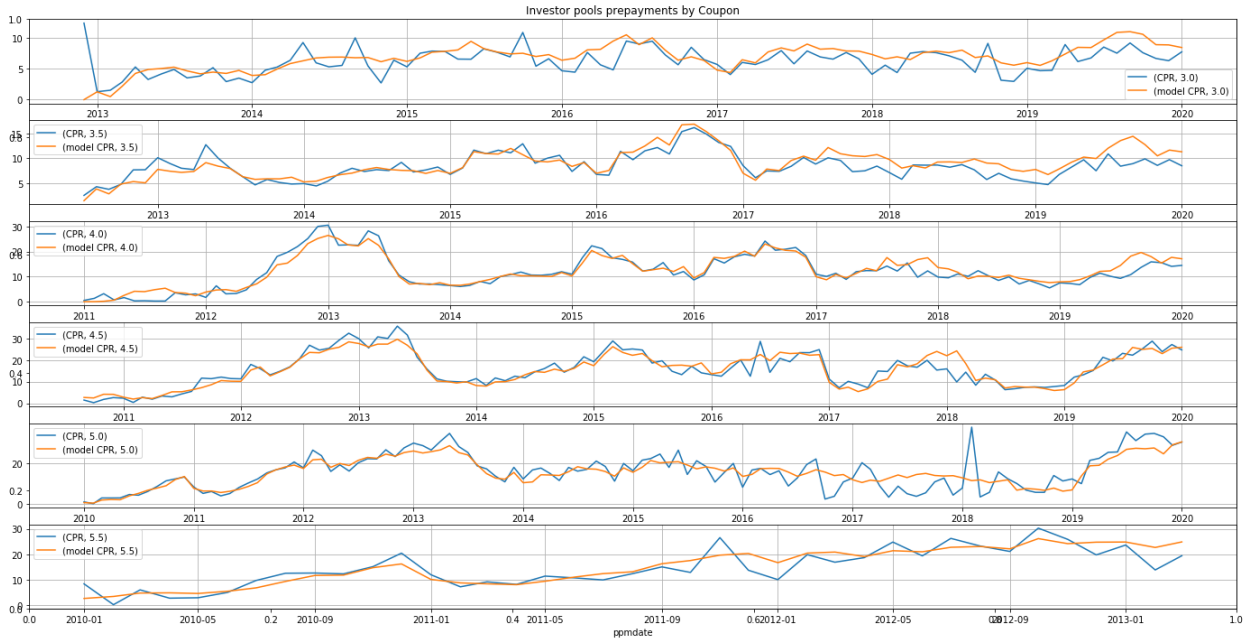
As seen on Fig 1 above, the model fits the historical data quite nicely even during the out of sample period. The biggest problem is in higher coupons through the end of the 2019, when mortgage rates have dropped, and prepayment speeds increased quite sharply. The model is not fitting that period well for 4%, 4.5%, and 5% coupons. However, if we restrict our population to the so-called loan balance pools (pools consisting of loans with less than 200k original loan balance), the model performance improves quite a bit, see the next graph below.



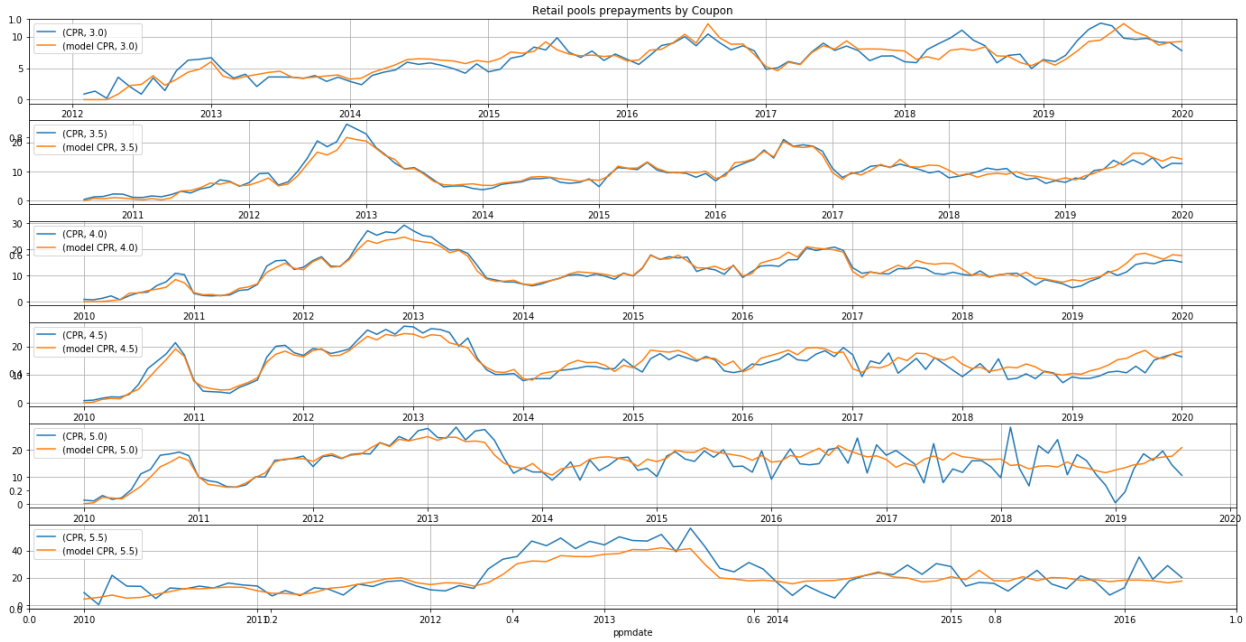
Model performance on the so called “Cash Window” pools is also quite satisfactory, see fig below. Cash Window pools are usually composed of mortgage loans originated by very small originators. They are an important sub category of mortgage pools that requires us to look at separately.



Performance of the model on investor loans is satisfactory as well, see graph below.

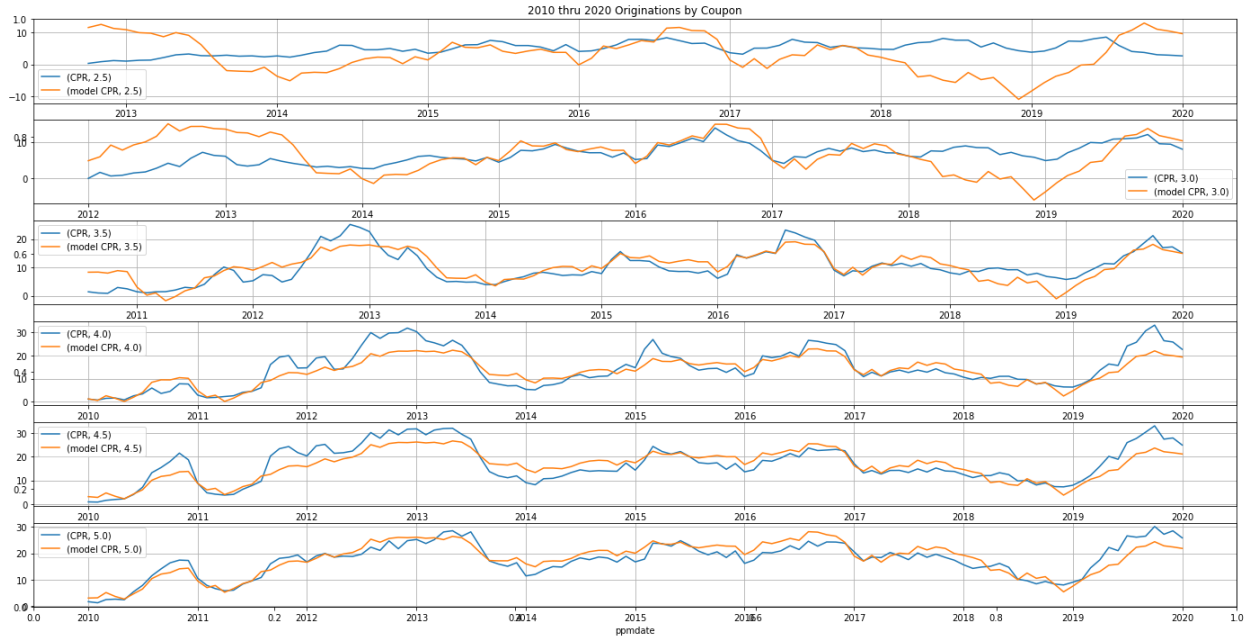


The model performs particularly well on “Retail” pools or pools consisting of loans originated through a retail channel, so not through a Broker or a Correspondent channel. Think of these loans as originated by a bank or a large financial institution. Retail loans usually behave quite tamely when compared to Broker or Correspondent loans. That is because brokers tend to solicitate their clients more aggressively for refinancings as the only way they make money is through origination fees.



8 Comparison to benchmark models

Model performance of the linear model was far worse than that of the neural network model. We can see the problems visually on the charts below.



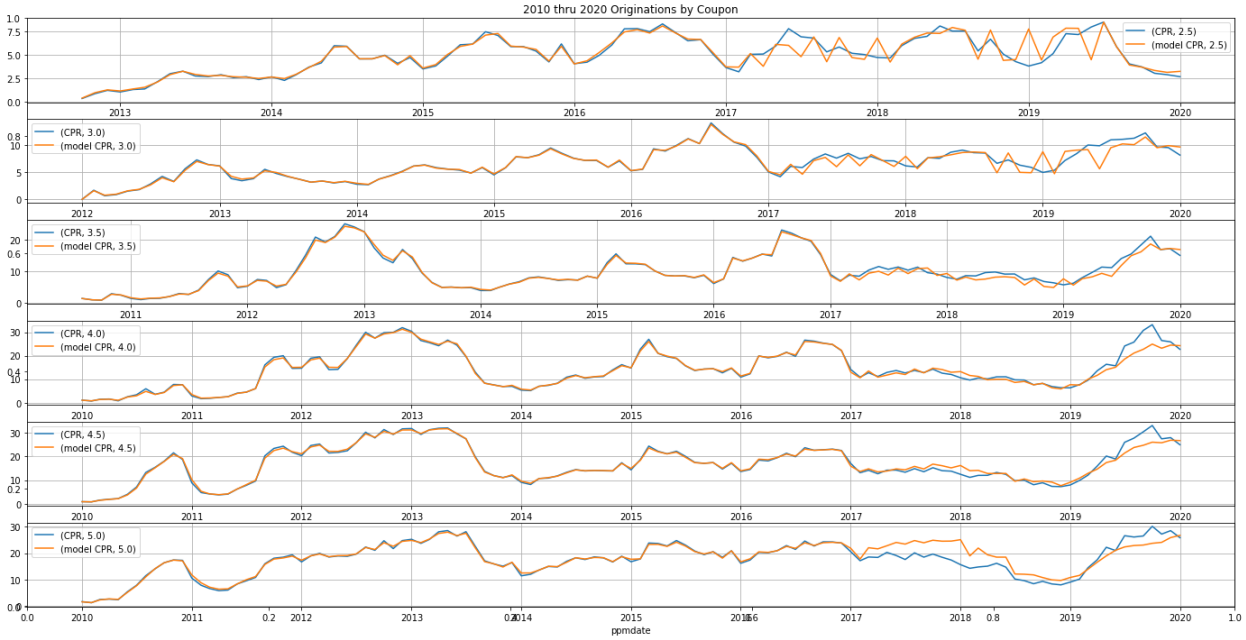
Per our model performance metrics, linear model's performance was far worse at 7.92 CPR

Coupon	LM RMSE (CPR)
2.5	7.92
3.0	5.13
3.5	2.90
4.0	3.94
4.5	3.88
5.0	3.35

A random forest model has performed much better than the linear model, and generally it did on par with the neural-network model. Per our performance metrics, the RMSE model performance was at 4.03 CPR (see table below).

Coupon	RF RMSE (CPR)
2.5	1.19
3.0	1.30
3.5	1.37
4.0	2.52
4.5	2.22
5.0	4.03

However, some of the jumpiness in model output (see below 2.5s & 3s out of sample predictions) is concerning.



In summary, a neural network model seems to be a winner on both our performance metrics and through visually inspecting the model output.

Coupon	LM RMSE (CPR)	RF RMSE (CPR)	NN RMSE (CPR)
2.5	7.92	1.19	1.66
3.0	5.13	1.30	1.16
3.5	2.90	1.37	1.52
4.0	3.94	2.52	3.42
4.5	3.88	2.22	3.21
5.0	3.35	4.03	3.37

9 Conclusions

In this project we have built a neural network-based mortgage prepayment model and have showed that such a model has a satisfactory predictive power and its output depends on model inputs in a way that makes economic sense. The relative speed with which such models can be developed potentially gives a desirable advantage to any mortgage investor who wants to be able to adjust to regime changes in the mortgage market as quickly as possible and to any risk manager who wants to risk manage mortgage assets as accurately as possible. Although, in our study we have chosen to have a 3 year out of sample period, in reality in the industry such models are updated much more frequently. In particular, this modeling approach allows us to update a model on a monthly basis and just in one day. For some types of mortgage trades the short horizon prepayment predictions are the most important, and surely a 1-month old model is capable of making much better predictions than a 3-year old model. So, this approach has a lot of potential and many potential use cases.