LendingHome Interview

For Summer 2018 Data Scientist Intern

Zeyu Li
lizeyu_cs@foxmail.com / zyli@cs.ucla.edu
Ph.D. student in Computer Science at UCLA

Motivation

In this section, I will introduce the problem defined for this exercise project.

Fix & Flip loan, in short, is a kind of loan that individual borrow money from lenders to fix the house and resell it quickly to make money. Hence, the amount of transaction would become critical to both lenders and borrowers. Lending agencies can decide how much to lend based on transaction amount. Borrows, or investors, can estimate how much he can earn using the loan. According such consideration, we targeted at predicting the amount of transaction (“transaction_amount” in the given dataset) using other information from the dataset.

Let’s first take a look at the dataset!

- **Dataset**: Fix and Flip loan transaction history of California since 2012.
- **Size:** 158,800
- **Attributes:**
  - Property ID
  - Transaction amount (Numerical feature)
  - Loan amount (Numerical feature)
  - Buyer (String)
  - Lender (Categorical feature)
  - Transaction date (Date)
  - Property type (Categorical feature)
  - Property address (String)
  - Seller (String)
  - Year built (Categorical feature)
  - Square feet (Numerical feature)

Let’s first take a look at the dataset! (Cont’d)

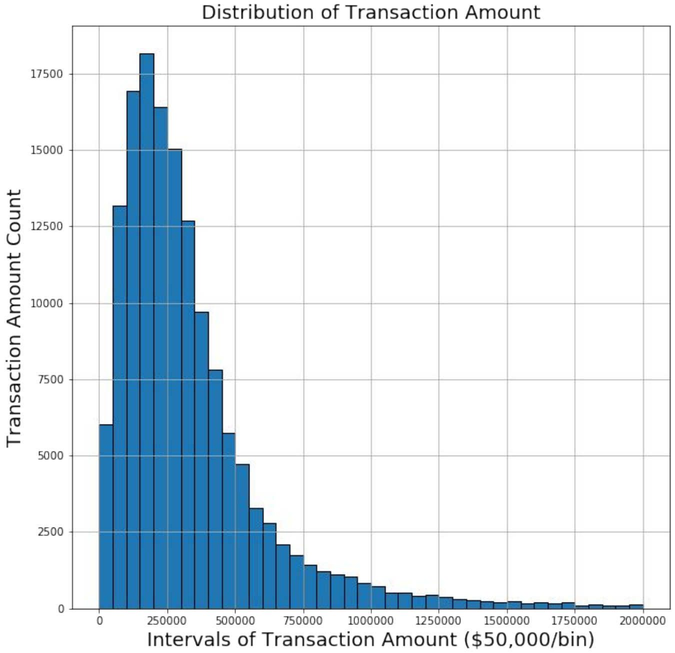
- **Dataset**: Fix and Flip loan transaction history of California since 2012.
- **Size:** 158,800
- **Attributes:**
  - Property ID
  - Transaction amount (Numerical feature) *99.95% are below 1,300,000.*
  - Loan amount (Numerical feature) *49,182 entries are non-zero, ~30.97% of the size of entries.*
  - Buyer (String) *People’s names.*
  - Lender (Categorical feature) *Lending agencies, 10,074 unique agencies, ~69.70% are* “`* Undisclosed`”.
  - Transaction date (Date) *Need further interpretation.*
  - Property type (Categorical feature)
  - Property address (String) *Can extract information from Zipcode.*
  - Seller (String) *People’s names.*
  - Year built (Categorical feature)
  - Square feet (Numerical feature) *6,349 entries have 0 in both Year built and Square feet.*

Data Cleaning before applied to models

Principles for dropping data instances:

1. Remove entries that have a ‘`transaction_amount`’ greater than 1,300,000. Data is getting sparse when ‘transaction_amount’ goes beyond 1,300,000.
2. Remove entries which have 0’s for both ‘`year_built`’ and ‘`sqft`’. These entries are removed since they are missing too much information.

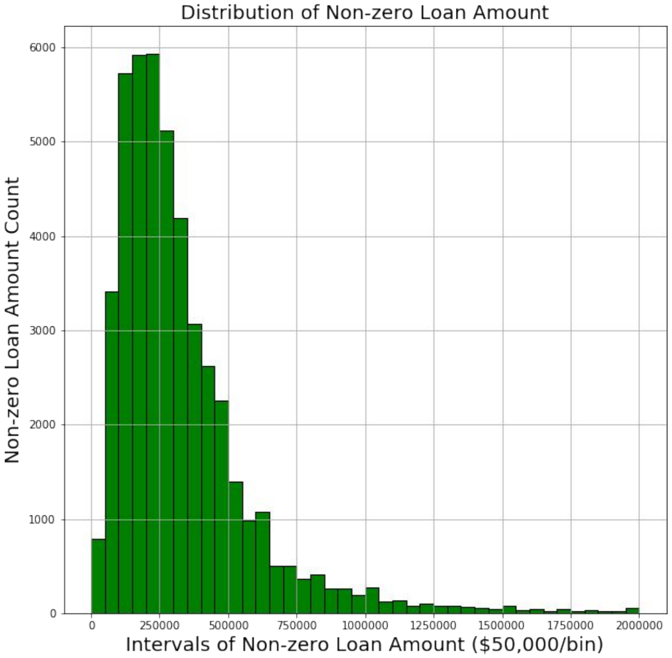
How values are distributed by attributes?



Transaction Amount:

Here’s a histogram of the value distribution of transaction amount attribute. It has a “long tail”. This figure only shows the amount from 0 to 2,000,000, where the “long tail” property has already become obvious. 87.31% of data are concentrated in [0, 500,000].

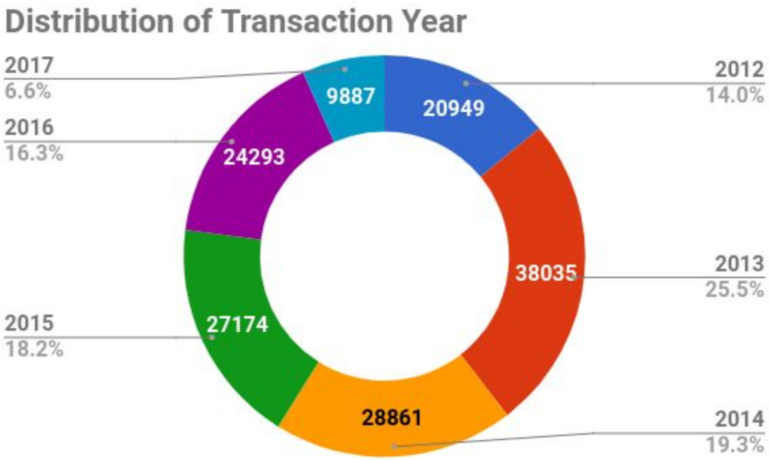
How values are distributed by attributes?



Non-zero Loan Amount:

Around 31% of loan amounts are non-zero. Loan Transaction amount also has a “long tail”. This figure also shows the amount from 0 to 2,000,000, where the “long tail” property has already become obvious. Same as Transaction amount, loan amount also concentrates in [0, 500,000].

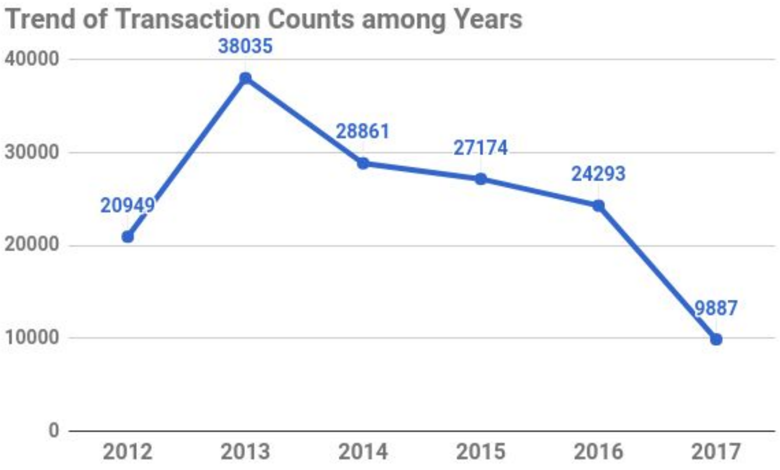
How values are distributed by attributes?



Transaction Year:

This time, we are having a good attribute. That is, the cleaned dataset doesn't have any instances whose "transaction year" is zero. The donut figure shows the proportion of transaction counts among the 6 years (2012 - 2017).

How values are distributed by attributes?

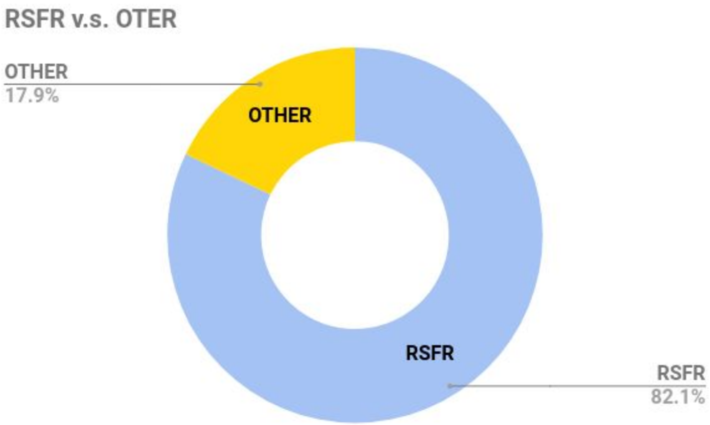


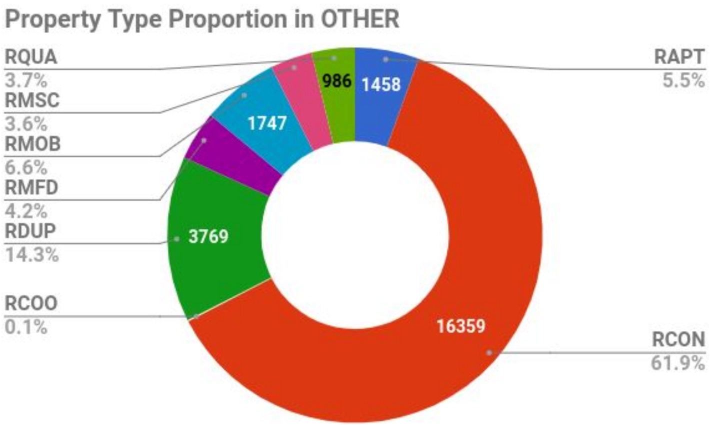
Transaction Year (Cont’d):

Now we explore the from the trend of transaction counts. Fix&Flip loan reached the peak at 2013 and has decreased since then. Note that the data of year 2017 is not complete. However, we can consider it as “complete” because the latest recorded date was “2017-12-07”. It is almost the end of the year.

Studying the reason of the decrease is a nice problem to work on. However, from this dataset, the information of the purchasing power of customers is lacking. We cannot move further with extra datasets.

How values are distributed by attributes?





Property Type

Here we can see that RSFR dominates the property type. It may lead to great bias of the result. However, we will use it since, from our knowledge, we know type is critical to the house price.

What features did we extract?

After all above analysis, here we list the features and target fed to the models.

Numerical Features

- Transaction amount (Target)
- Loan amount
- Age of the real estate when sold
- Square feet
- *Zillow Home Value Index* [1]

Categorical Features

- Property type
- Weekday
- Year sold

What features did we extract? - Age

The age is an important factor to the value of the house. Considering this fact, we utilized “age” as one of the features. Obviously, it is computed as follow:

```
age = year_built - transaction_date.year
```

Also, “transaction_date.year” is also chosen as one of the features. Python package “`datetime`” is used to extract year from “YYYY-mm-dd” format string.

What features did we extract? - Age (Cont'd)

Why "age" + "transaction_date.year" instead of "year_built" + "transaction_date" ? The reason is that in "transaction_date", year should be considered as a categorical feature. It has 2 weak points:

- Losing the information of distance. "transaction_date.year" is also a categorical feature. If both are encoded by encoders such as "one-hot encoder", the distance information, i.e. the age will be missed. You cannot get "age" by subtracting the encoded values.
- Making the feature matrix sparse. Unlike "transaction_date.year", "year_built" has a value range from 1808 to 2017. If encoded by "one-hot", it will increase the feature matrix by 215 columns at most. It can make the feature matrix much much sparser. For sparse matrix, a huge amount of data instances are required. However, our dataset is obviously not a large one.

What features did we extract? - ZHVI

ZHVI is short of Zillow Home Value Index. From the methodology description of the dataset [link], ZHVI is developed from 2005 to approximate the idea home price index by leveraging the valuation Zillow creates on all home, which is also called Zestimate. The dataset provides ZHVI and Peak ZHVI (the maximum ZHVI over months from June, 1996), we can query the ZHVI and Peak ZHVI by zipcode extracted from “property_address”.

The reason we are using ZHVI is that we wish to make use of “property_address”. Originally, it is an string feature, hard to extract information. By integrating Zillow dataset, we could then take advantage of the “property_address” feature. ZHVI is another important factor of transaction amount. *For the rows whose address or zipcode are unavailable (i.e. “NaN”), we used the average ZHVI and PZHVI as their ZHVI and PZHVI.*

What features did we extract? - Weekdays

For each “transaction_date”, we not only extract the year, but also extract the weekdays of each date. We used “`datetime`” package in Python to convert date to one in “Monday”, “Tuesday”, …, “Sunday”. This is another categorical feature.

The motivation of introducing this feature is that weekday may affect the the decision-making of the lenders. Lenders are human. Although the decisions are supposed to be made by a group of people after an entire procedure of evaluation, I still believe that it may be affected by the weekday. When weekends are coming, people are happy and more likely to approve the loan applications.

The last step of feature engineering

- Categorical feature encoding

This article [3] compared 7 major methods for categorical feature encoding, which are Ordinal, One-Hot, Binary, Sum, Polynomial, Backward Difference, and Helmert.

In this problem, we chose Binary Encoder to encode features. For details of this encoder, please refer to this link. There are 2 reasons of choosing binary encoder.

(1) Avoid high sparsity of feature matrix. *N* digit binary encoder is able to encode $2^{N+1}-1$ categories. However, one-hot can only encoder *N* categories.

(2) High accuracy. From [3], we can see that binary encoder has the best performance among 3 tasks.

Settings before training & testing

- **Cross validation:**
    - We used 10-fold cross validation for testing. We choose the shuffled version, `ShuffleSplit` in `Scikit-Learn,` to avoid training data or testing data contain local properties.
- **Models:**
    - We applied the dataset to Linear Regression, Gradient Boosting Regressor, Support Vector Regression, and Neural Network to predict the transaction amount.
    - Python package `Scikit Learn` is used for model implementations.
- **Normalization:**
    - For Neural Network, we normalized each column by dividing each value by the maximum value. Experiments show that normalization greatly enhance the performance.

Models utilized as regressors

- **Linear Regression**
- **Gradient Boosting Regressor**
    - Gradient Boosting Regressor is an ensemble model of weak prediction models, typically decision trees. Gradient boosting can be used in both regression and prediction problems. The result will show that in small datasets, and low dimensional data scenarios, ensembling method has a good performance.
    - Settings:
        - Number of estimator: 100
        - Loss function chosen: Least square

Models utilized as regressors

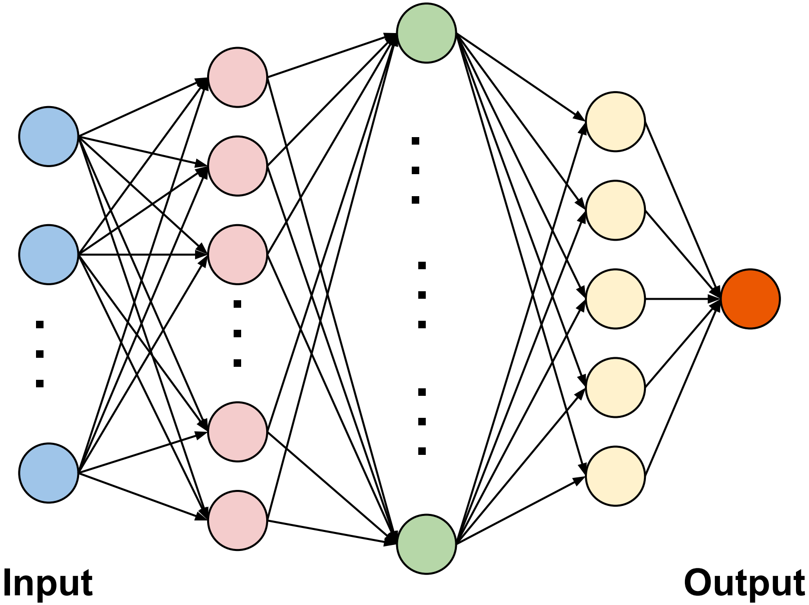
- **Support Vector Regression (SVR):**
    - The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences. Because output is a real number it becomes very difficult to predict the information at hand. Therefore, a margin of tolerance (epsilon) is set in approximation to the SVM. However, the main idea is always the same: to minimize error, individualizing the hyperplane which maximizes the margin, keeping in mind that part of the error is tolerated. [4]

Models utilized as regressors

- **Neural Network**
  - Essentially, it is a Multi-layer Perceptron Regressor. In this model, multiple layers of perception are fully connected with the weighted edges between the layers.
  - Settings (A figure of the architecture is on next page):
    - Layer & Number of perceptrons: 200 - 50 - 5
    - Activation function in perceptrons: `tanh`
    - Optimizer: Stochastic Gradient Descent
    - Maximum iteration : 500
  - Simplified version settings:
    - Layer & Number of perceptrons: 20 - 5
    - Other: same as above.

Models utilized as regressors

- **Neural Network.** Graphical representation



Color	Category	Count	Activation
🟦	Input	13	-
🟥	Hidden Layer 1	200	`tanh`
🟩	Hidden Layer 2	50	`tanh`
🟨	Hidden Layer 3	5	`tanh`
🟧	Output	1	-

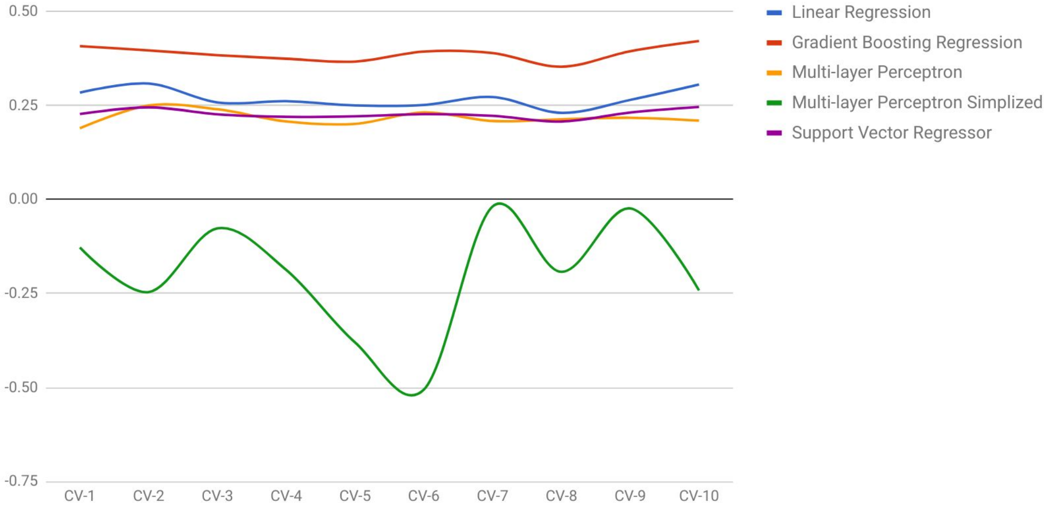
Metrics for evaluation

- R² Score
  - Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of y, disregarding the input features, would get a R² score of 0.0. See this link for more information. wiki

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)}{\sum_{i=0}^{n_{samples}-1} (y_i - \bar{y}_i)} \qquad \bar{y} = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} y_i$$

- Explained Variance Score
  - The explained variance is estimated as follow. The best possible score is 1.0, lower values are worse. $explained\_variance(y, \hat{y}) = 1 - \frac{Var\{y - \hat{y}\}}{Va\{y\}}$

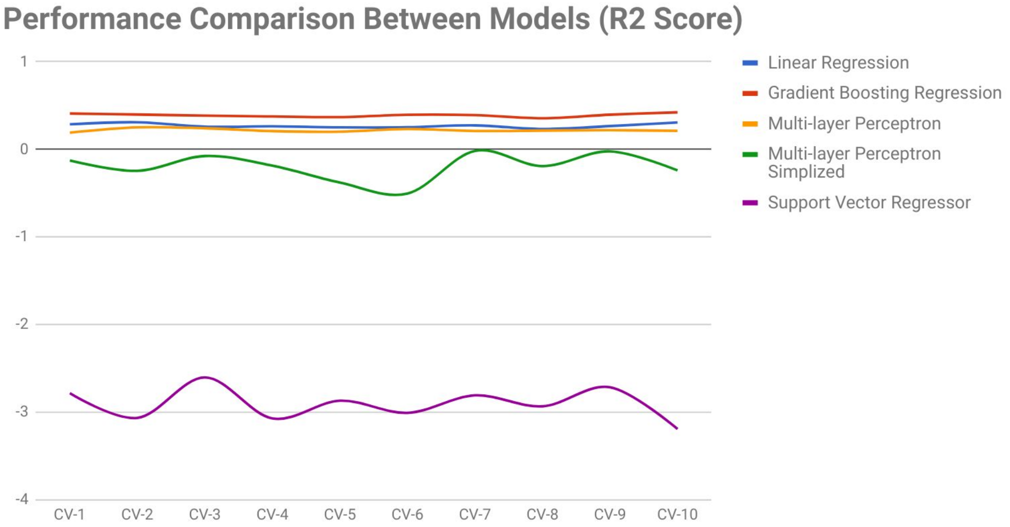
Results - Explained variance score



Performance Comparison Between Models (Explained Variance Score)

This figure shows that gradient boosting regression has the highest explained variance score, representing the best performance. Linear model outperforms the neural network model (multi-layer perceptron) and support vector regression (SVR). One thing to note is that SVR may took much more time than other models since computing the dual form is time-consuming. Simplified neural network model cannot extract enough information and obtain representation ability so the performance is under 0.

Results - R^2 score



Most of the lines are similar to the previous figure. Gradient boosting performs best, then linear regression and multi-layer model, the the last simplified version. However, we can see that support vector regression has a significantly low R^2 score, meaning that SVR may produce a high variance and lead to less reliable future prediction.

Here’s the conclusion!

In this project, we targeted at predicting transaction amount based on the Fix & Flip dataset and Zillow Home Value Index dataset. We tried 4 different regression models, which are linear regression, gradient boosting regression, support vector regression, and neural network regression. Gradient boosting regression, among all models, has the best performance. The R^2 square reaches around 0.4, which represents a good accuracy given such size and quality of the dataset.

In order to better improve the performance, here are my advices:

1. Collect more data, both in number of rows and in number of columns. More rows can train parameters with more instances. More columns can increase the representation ability of the learned model.
2. Enhance the data quality, especially for those critical attributes. For example, we noticed that a great amount entry has 0 in “year_built” or “sqft”. Also, “lender” is a good study problem but over 65% lenders are undisclosed making it infeasible to work with.

References

[1]. Zillow Home Value Index dataset. https://www.zillow.com/research/data/

[2] Python packages. Pandas, Numpy, and Scikit-Learn.

[3] Beyond One-Hot: an exploration of categorical variables.
https://www.kdnuggets.com/2015/12/beyond-one-hot-exploration-categorical-variables.html

[4] Support Vector Machine - Regression. http://www.saedsayad.com/support_vector_machine_reg.htm

Thanks!

Many thanks to LendingHome for reading my CV carefully and providing me this great interview opportunity! Looking forward to working with you in Summer 2018! If you have any questions, or if you believe a video is necessary for this presentation, please feel free to let me know!

Also, many thanks to my cats for tolerating my noisy keyboard hitting!

Zeyu Li
Jan, 3, 2018