hmda-map.com: Project Report

Course:	INFO254 — Data Mining and Analytics, Spring 2018
Date:	May 8th 2018
Team:	Avi Dixit, Ashish Sur, Michael Fröhlich, Maximilian Wühr, Rohan Kapuria, Rohan Kar
Links:	https://www.hmda-map.com/#/ https://github.com/FroeMic/DMA18-FinalProject-Server https://github.com/FroeMic/DMA18-FinalProject-SPA
Description:	US home loan data is made public under the Home Mortgage Disclosure Act. However, for citizens without experience in analyzing, aggregating and manipulating large data sets the published information stays inaccessible. The purpose of this project is to change this by providing insight into the housing loan market in the state of California by doing two things; (1) visualize the average approved loan amount per county and (2) visualize personalized predicted loan amounts per county.

Table of Contents

1. Introduction

Each year thousands of banks and other financial institutions report data about mortgages to the public as part of the Home Mortgage Disclosure Act (HMDA)[1]. This data is important because:

- It helps show whether lenders are serving the housing needs of their communities;
- It gives public officials insight that helps them make informed decisions about policies;
- It sheds light on lending patterns that could be discriminatory

The Consumer Financial Protection Bureau (CFPB)[2] provides every year of the Loan Application Register (LAR) data since 2007. That adds up to approximately 15 to 20 million records every year.

Even though the data is accessible to the public through the CFPB's API, the information it holds is not. For citizens without experience in analyzing, aggregating and manipulating large data sets the published information stays inaccessible. The purpose of this project is to change this by providing insight into the housing loan market in the state of California by doing two things in particular: First, a visualization in form of a heat map down to the detail level of counties should provide an easy to understand comparison of the average approved loan amounts. Second, a machine learning model should provide citizens interested in applying for a loan with a prediction of what loan they can expected per county. Exemplary visualizations can be found in Appendix *B. Map Visualization*.

Superimposing the average loan-sizes in form of a heatmap over California enables citizens to reason about housing costs in the respective county. Assuming that housing loans cover 85% to 95% of the costs of buying a housing, it additionally allows to reasons about the total costs of buying a house in a county.

Additionally, citizens would benefit from providing personalized predictions per county, as the could make a more informed decision on where to look for housing. As an example, given that a young professional working in San Francisco wants to purchase a house, they may consider looking for a suitable property in the neighboring counties. By providing a prediction personalized to their income and demographics, they could then start their search in a county where their expected loan allows them to buy a suitable property.
Furthermore, by comparing the predicted loan amounts across different counties, one can identify those regions where the lenders (banks) expect the properties to develop above the market average, by their willingness to provide higher loans.

[1] https://www.ffiec.gov/HMDA/

[2] https://www.consumerfinance.gov/

2. Dataset

The data for this project origins from two sources: First, the home loan register data reported under the HMDA and second, the shapefiles[3] for the counties of the US states used for visualization on the map. In the following a detailed description of the former one is provided.

The CFPB provides a public API[4] that allows access to all recorded loan applications register data since 2007. For data published by a public agency, the platform[5] driving the API is (surprisingly) easy to access and interact with. The CFPB provides additional resources to improve the accessibility of the data, i.e. videos explaining the LAR dataset on a high level[6] and detailed documentation[7].

For the scope of this project we only consider the 2016 LAR dataset for the state of California. This dataset amounts to a 1.56 gigabyte sized *.csv* file with a total of 2,235,971 instances which each has 47 features. A full description of all features can be found in Appendix *A. Field Reference*.

With regards to developing the prediction model, our analysis showed that the following features contributed considerably to the outcome of the predicted loan amount.

Field name	Description	Data type
`applicant_income_000s`	The gross annual income that the lender relied on when evaluating the creditworthiness of the applicant, rounded to the nearest thousand.	integer
`applicant_race_1`	A code representing the first listed race for the primary applicant. The applicant can list up to five races.	integer
`applicant_sex`	A code representing the sex of the primary applicant.	integer
`co_applicant_race_1`	A code representing the first listed race for the co-applicant. The co-applicant can list up to five races.	integer

[3] https://www.census.gov/geo/maps-data/data/cbf/cbf_counties.html

[4] https://api.consumerfinance.gov/data/hmda

[5] http://cfpb.github.io/qu/

[6] https://youtu.be/wR9Tsdqgmuk

[7] http://cfpb.github.io/api/hmda/basics.html

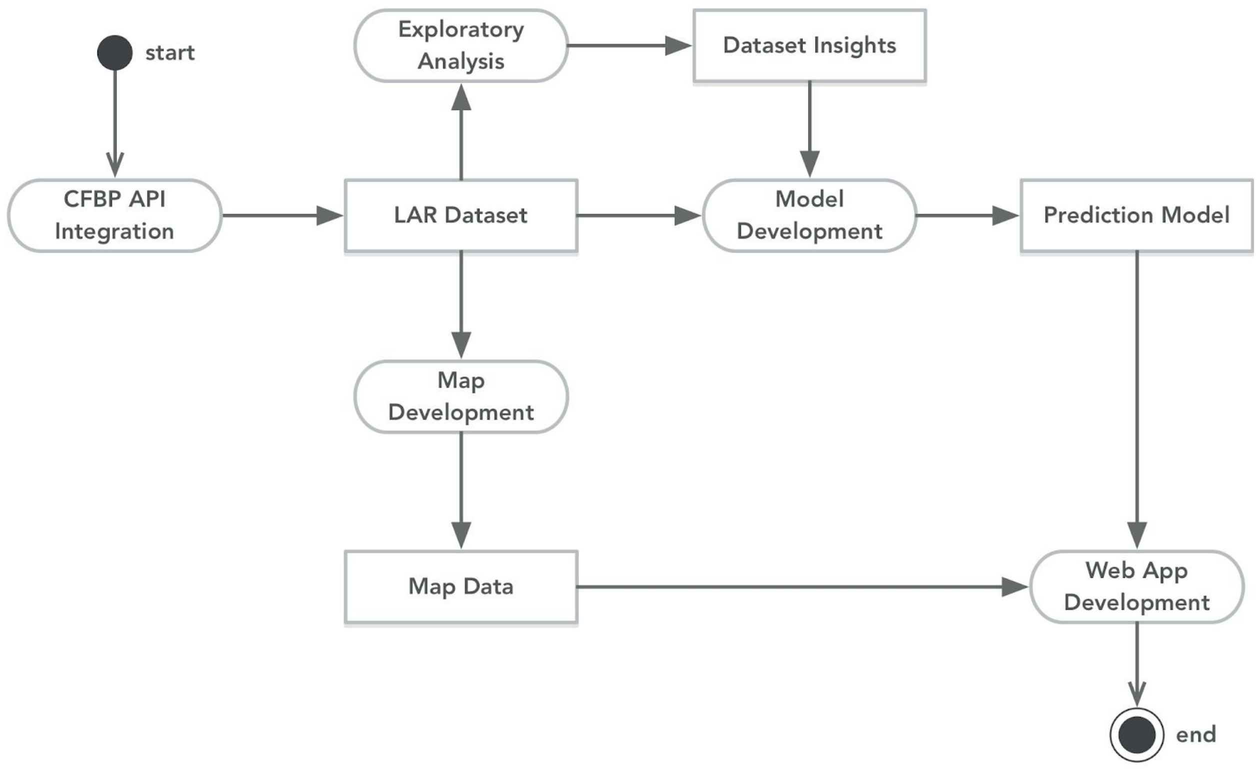
co_applicant_sex	A code representing the sex of the co-applicant.	integer
hud_median_family_income	The median family income in dollars for the MSA/MD in which the tract is located.	integer
loan_purpose	A code representing the purpose of the loan (home purchase, refinance, or home improvement).	integer
number_of_owner_occupied_units	The number of dwellings in the tract that are lived in by the owner.	integer
preapproval	A code representing the pre-approval status of the application.	integer
property_type	A code representing the type of the property.	integer

The target for the prediction is the `loan_amount_000s` field, representing the amount of the loan applied for, in thousands of dollars as an integer.

Furthermore, for mapping single instances to geospatial locations, the `state_code`, `county_code` and the `census_tract_number` fields are particularly interesting. Using them allows to correlate each loan application to a polygon within the shapefiles describing the respective area on the map. Additionally, by grouping the successful loan applications by these fields we calculate the average size of a loan per county.

3. Introduction to Approaches

Our development approach included five major activities which were distributed among all team members. The following picture illustrates the development activities, their outcomes and dependencies.



3.1 CFPB API Integration

Initially, we needed to obtain the raw data from the CFPB's API. The primary outcome of this activity was to have the data for the state California for 2016. As a secondary goal, we wanted to structure code in the ipython notebook in such a way that would allow to download data for other US states in the future easily.

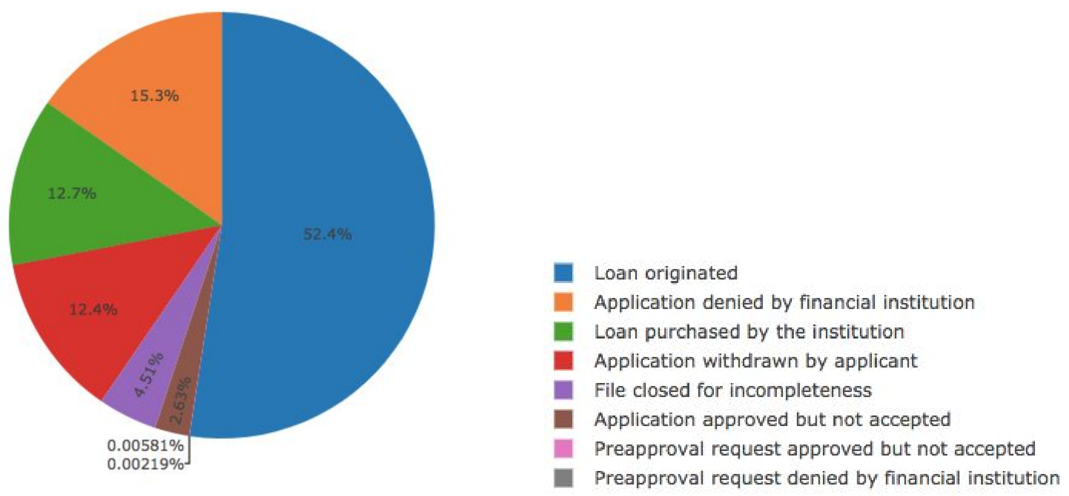
Desired Outcome:	LAR Dataset for California for 2016 in *.csv* format
Contributors:	Maximilian Wühr, Michael Fröhlich

3.2 Exploratory / Statistical Analysis

Once we had the dataset, we needed to get a better understanding of its contents. While all team members were responsible to explore the dataset with regards to their assigned tasks, Rohan Kapuria led the presentation of this effort.

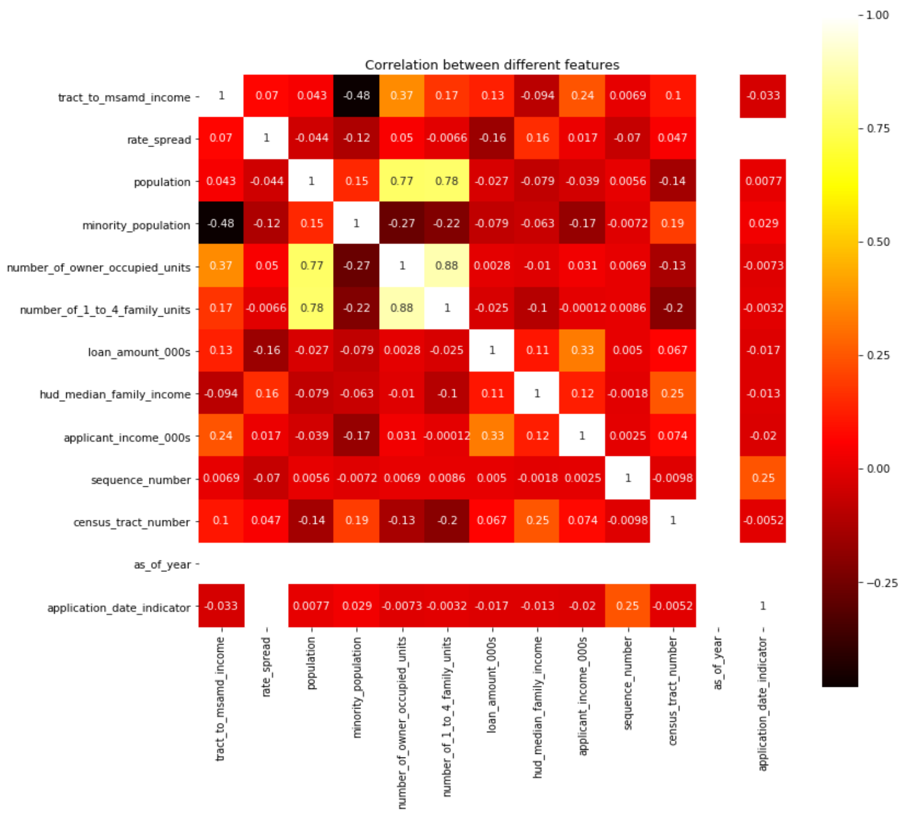
Desired Outcome: Exploratory analysis in form of iPython notebook
Contributors: Rohan Kapuria

Action taken on loan



- A majority of the loans were approved (52.4%) followed by loan denials at 15.3%
- We used only the “loan approved” and “application approved but not accepted” to calculate the base average of each county in CA.

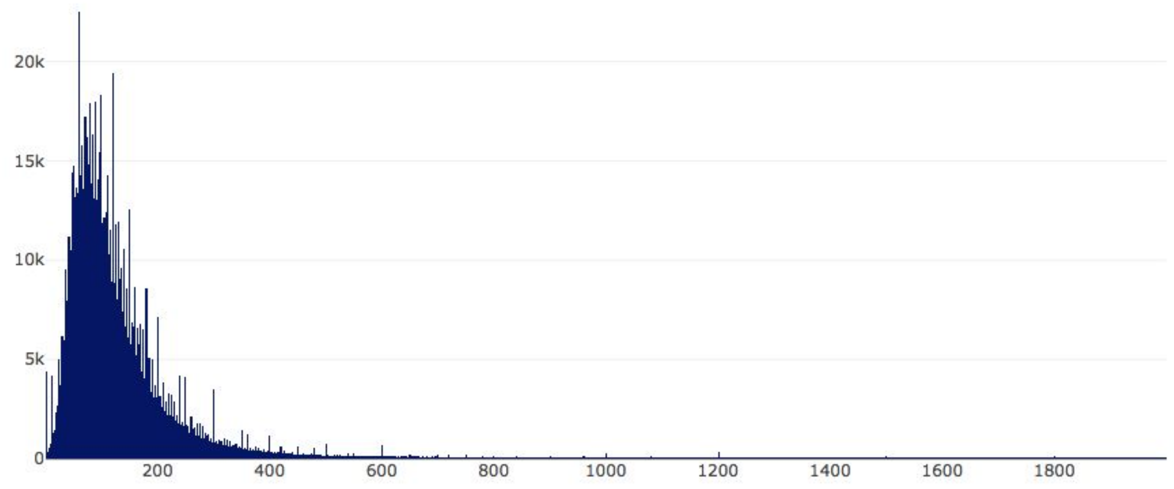
Correlation matrix



- Correlation b/w applicant income and loan amount is 0.33. This seems to be natural. A person with high income would want to buy a bigger more expensive home.
- Owner occupied units vs tract_to_msamd_income = 0.37. Means that there are more “owner occupied units” in an area where the tract avg income is higher than the avg. income of the metropolitan statistical area i.e. so called “richer areas”. This could be used to identify preferred areas by potential home buyers.
- Number of owner occupied units in an area is highly correlated with number of 1-4 family units in that area. This could mean that when the owner lives in a home they generally live in houses (not apartment buildings) that can house a small family (4 people) = 0.7.
- Minority population vs tract_to_msamd_income is strongly negatively correlated = -0.48 i.e. places which have a high minority population have lower income levels compared to the average for the metropolitan statistical area.

Note: 1- to 4-Family Units includes Condominiums, townhouses, single-family homes, mobile homes, boats, RVs, and vans. Apartment buildings are not included.

Frequency distribution of loan amounts



- As expected, loan amount (in 1000s of dollars) is heavily right skewed. We noted this and used log transformations in the models to compensate for this skewness.

3.3 Feature Engineering , Pre-processing and Model Development

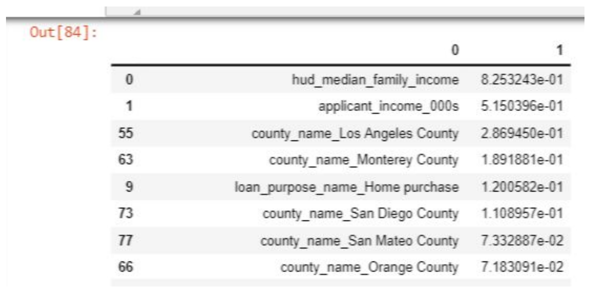
Since we were trying to predict the closest loan amount a particular applicant might receive, we considered this from two perspectives, from a classification point of view, where we divided each loan amount into buckets of say 10k$ and performed a softmax classification to find which bucket a particular loan belonged to, and from a regression perspective where we essentially tried to predict the exact loan amount feasible for that applicant.

After a few discussions, we realized that it made more sense to convert this to a regression problem because the final solution would be a lot more interpretable. We would essentially be able to provide the average error across our dataset and examine each prediction individually to see close/far it was from the actual value. The classification problem on the other hand depended a lot on variable parameters such as how large our buckets were, etc. and it appeared that we needed to define our problem to a much larger extent even before we actually started working on building classifiers.

To start things, we needed to make a decision on what fields we need to keep in our models and what fields were unneeded. We essentially ended up with two strategies for deciding which variables to keep, 1) We envisioned that a prospective user might use this to fill a loan application, and any field that would not have been able to be filled in by the user was dropped, such as fields indicating who actually purchased the loan, 2) Fields that had a lot of null values such as gender and ethnicity of the 5th co-applicant for the loan. Additionally, since we were trying to predict the actual loan amount that was offered to the applicants, we decided to keep only applications that had the loan approved for them.

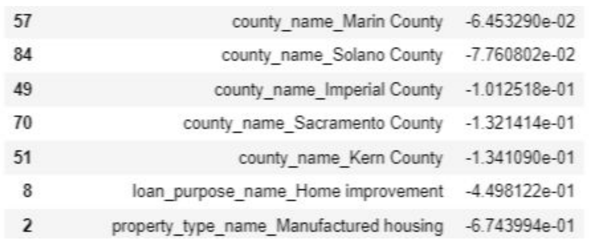
To start with a simple baseline, we used a linear regression models and were able to achieve an accuracy of approx 2100 dollars. We next decided to evaluate the feature importance for this model and as was expected, personal income was the most important feature for this model.

This is a sample of the coefficients with largest absolute values from the model.



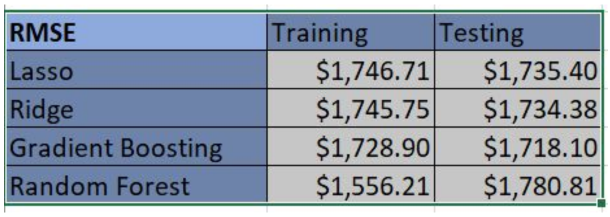
Out[84]:

	0	1
0	hud_median_family_income	8.253243e-01
1	applicant_income_000s	5.150396e-01
55	county_name_Los Angeles County	2.869450e-01
63	county_name_Monterey County	1.891881e-01
9	loan_purpose_name_Home purchase	1.200582e-01
73	county_name_San Diego County	1.108957e-01
77	county_name_San Mateo County	7.332887e-02
66	county_name_Orange County	7.183091e-02

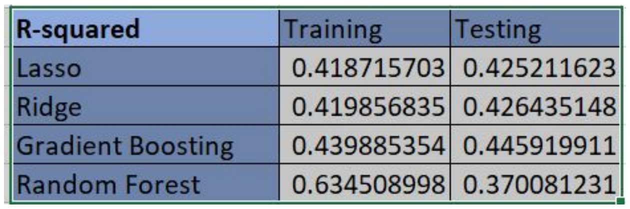


57	county_name_Marin County	-6.453290e-02
84	county_name_Solano County	-7.760802e-02
49	county_name_Imperial County	-1.012518e-01
70	county_name_Sacramento County	-1.321414e-01
51	county_name_Kern County	-1.341090e-01
8	loan_purpose_name_Home improvement	-4.498122e-01
2	property_type_name_Manufactured housing	-6.743994e-01

We next decided to perform grid search using cross validation on all major regression models that were available to us to enable us evaluate whether there were specific strengths/weaknesses in the models, and found that random forest performed the worst, followed by L1 and L2 regressions and the best model was a gradient boosted regressor.



RMSE	Training	Testing
Lasso	$1,746.71	$1,735.40
Ridge	$1,745.75	$1,734.38
Gradient Boosting	$1,728.90	$1,718.10
Random Forest	$1,556.21	$1,780.81



R-squared	Training	Testing
Lasso	0.418715703	0.425211623
Ridge	0.419856835	0.426435148
Gradient Boosting	0.439885354	0.445919911
Random Forest	0.634508998	0.370081231

For all of the above models, we had encoded all the available counties as one-hot vectors and were using those run the inputs into a single model. We had discussed the possibility whether each available county might require models that were tuned to their specific distributions and consequently configured and ran different regressors for all the counties and noted the results.

As seen in the table, practically speaking, the RMSE for the lasso, ridge and GBM were similar, with only a few dollars difference. This difference would not matter for a loan of magnitude of 100,000. As the accuracies were similar, the Ridge regression was chosen to be the final model as it was the an interpretable model and has a lower RMSE than the Lasso and similar R-square values. The model also made sense in a practical world with a RMSE of $1750 and the most heavily weighted features being the family income, followed by personal income, type of loan (refinance loans have larger predictions than home improvements),etc.

Desired Outcome: Prediction Model
Contributors: Avi Dixit, Ashish Sur, Rohan Kar

3.4 Map Development

In parallel to the development of the prediction model we started developing the map visualization. This task involved the aggregation (grouping, averaging) and subsequent serialization of the LAR data into a hierarchical JSON file. We then enriched the JSON with the necessary location data to visualize it on the map and finally stored it in a SQLite database for fast dynamic access.

Desired Outcome: Map Visualization
Contributors: Michael Fröhlich, Rohan Kapuria

3.5 Web Application Development

In the final step we brought all the pieces together by developing a web app. The application consists of two parts: (1) A web application server built with Flask[8] providing map data and loan predictions over an API in JSON format and (2) a Single Page Application (SPA) built with VueJS[9] that visualizes the provided data on a map.

Desired Outcome: Interactive Web Application
Contributors: Avi Dixit, Michael Fröhlich, Maximilian Wühr

[8] http://flask.pocoo.org/

[9] https://vuejs.org/

4. Individual Work

The following section describes the contribution of the individual team members.

4.1 Avi Dixit

I was part of 3 tasks during the running of the project. All of these were centered around working with the data to such as selecting features, transforming data into machine readable forms, deciding the direction of which models to use, what architectures to use, and how to improve our models and decision making based on the metrics. On the Map development part of the project, I was chiefly involved with integrating our final model into the app architecture to generate the predicted values based on the user input.

We had to make certain decisions on how much data we needed to maintain for our models and what format that needed to be in. I took the lead in this task and was chiefly involved with deciding the structure of data that needs to be provided by user as an input from the map screen, and structuring the data to be easily consumable by Machine Learning models. I also worked to generate several versions of the data we had so that we could test different ways of modelling the problem, such as creating different datasets per county, working with normalized/unnormalized data, working with classification buckets, etc.

The next part of my contribution was to drive and plan out the general models we'll use and iterating on them to reach the final version of our loan predictor. Specifically, we worked to create robust models for our data by running grid search using cross validation to arrive at the best hyperparameters for our models. I also led the discussions in creating a Stacked Regression model and to try and create models per county instead of a monolithic model that was able to handle every data input.

The last part of my contribution was to write code that would integrate our final model with the front end map application. I used python features such as Pickle to save the best model to a file. The modular code structure allows us to easily exchange the model everytime we have a more accurate one.

4.2 Rohan Kar

Secondary Goals: Data Exploration, Feature Engineering and baseline model evaluation.

1. Imported pre-processed data [Avi] and segregated features by categorical and numerical: Since most variables were categorical we decided to one-hot encode them
2.
    a. Checked the skewness and kurtosis of the response variable first and chose to normalize through log transform: Observed that Loan Amount in dollars was highly right skewed. Skewness was 118.90 before log transform and -1.53 after log transform.
    b. Checked the skewness of the features and created a list of highly skewed numerical features
    c. Log transformed skewed numerical data as well as the response variable
3. Plotted heatmap of Correlation between features
4. Worked with [Ashish] to shortlist regression models including: Lasso regression, Ridge regression, Random forest Regression, Gradient Boosting Regression.
5. Noted baseline model accuracies and observed overfitting of data for Random Forest Regression [table below] : All the shortlisted models had average deviation from observed values (RMSE) of around 0.5. RF regression seems to have overfit to the training data.
6. Identified feature importances from Random Forest Regression: Applicant family income, Median Family income and population accounted for most of the feature importances.
7. Shared the IPython notebook as a template to perform the same tasks for the final county level data.

Application concepts and techniques learned in class:

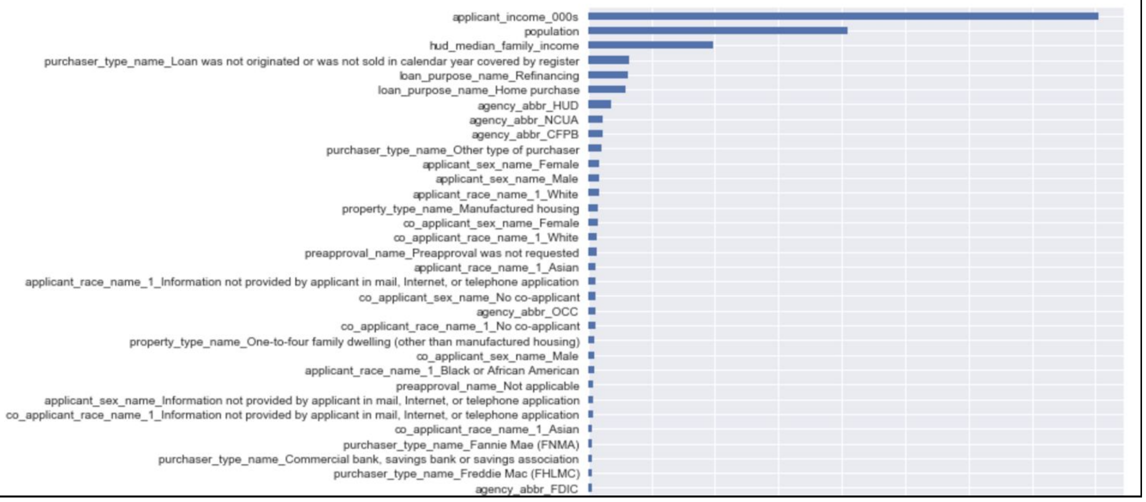
- Lab 1 and 2 helped with data preprocessing and feature engineering
- Plotting methods in Lab 2

Table: Baseline Model accuracies

Model	RMSE (log-dollars)	R Squared
Linear Regression	0.649	0.285
Ridge Regression	0.649	0.285
Random Forest Regression	0.248	0.894
Gradient Boosting Regression	0.550	0.486

Feature Importances (RF Regression)

```
In [31]: model.feature_importances_
         feature_importances = pd.Series(model.feature_importances_,index = X_train.columns)
         feature_importances.sort_values(inplace=True)
         feature_importances.plot(kind='barh', figsize=(8,12))
         plt.show()
```



Skewness and Kurtosis in Response Variable:

```
In [46]: # Check Skewness in the response variable before any normalization
         print("Skewness: %f" % Y_train.skew())
         print("Kurtosis: %f" % Y_train.kurt())

         # Check Skewness in the response variable afternormalization
         print("Skewness: %f" % log_y.skew())
         print("Kurtosis: %f" % log_y.kurt())

         Skewness: 118.905659
         Kurtosis: 21470.176781
         Skewness: -1.530837
         Kurtosis: 6.836100
```

4.3 Ashish Sur

I was incharge of building the models for the project. These are my contributions:

Modelling Approach

My approach towards building the models was two pronged. We needed to get to a good accuracy on the loan prediction and keep the model as interpretable as possible. Achieving a good accuracy is intuitive from a practical standpoint - error must be within a margin practical enough for real world loan prediction. This is very much needed to make our application usable'. Secondly, I needed to keep the model interpretability in mind as the banking sector is a highly regulated industry where understanding the information used by an analytical model is not just a preference, it is often a matter of legal compliance. The costlier the impact of an error, a breach in privacy, or possibility for discrimination, the greater the demand for an interpretable (and accurate) solution.

Model Development

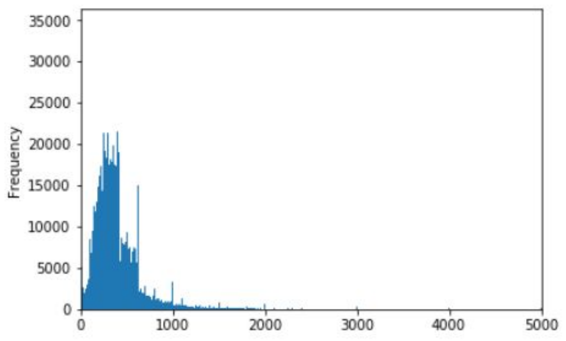
Keeping this in mind, I started using the most 'preferred' models: ordinary linear regressions, lasso and ridge linear regressions. Then i progressed on to trying and testing less interpretive models such as random forests and gradient boosted regressors. Initial trials were run on a sampled subset of the entire dataset just to check how long the model took for training. This was needed to understand how many hours and machines might be needed for this piece of work.

Iterative Feature Engineering

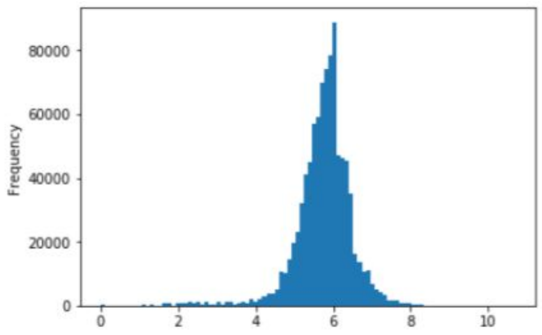
I had to go through three sets of feature iterations before settling on the final features and model.

First iteration, the cleaned dataset i received from Avi was used as is to train the models. The RMSE was of the order of 300,000 dollars for all the models trained. For an average loan amount of 500,000, this RMSE was huge.

Second iteration: To analyse the reason behind this, I checked summary statistics of the loan amounts. Some of the loan amounts were very large (100 million dollars or so). Finding this data plausible and not spurious (luxury villas in Malibu do cost much more than this), i created a histogram and saw that the loans given out were highly right skewed. The median value was about 500,000 dollars. To center this data, I tested predictions on log-dollars. Log-dollars is the value obtained on log-transformed loan values. Using this method, the regression target was log-dollars and we were taking anti-log of the predicted values to arrive at the final loan value. Using this method, the test and training RMSE reduced considerably to approx 2100$ in actual-dollar. This served as a baseline set of models.



Histogram of loan amount (x $1000)



Histogram of loan amount (in log - dollars)

Third iteration: To improve the RMSE, I had an intuition that the first thing a bank's loan officer asks before giving a loan is what is the customer's income and residual income as these are right skewed and the most important features as seen from the models we had trained, i suggested creating features such as log-of-applicant income and log-of-family income. Applicant income and family incomes were again right skewed hence, the above suggestion. RMSE improved by approx 200 dollars with the logged- features.

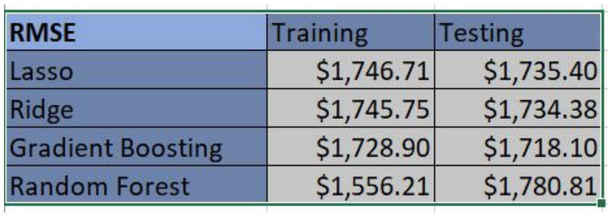
Model Selection

Five-fold cross validation was done on all of the training iterations.

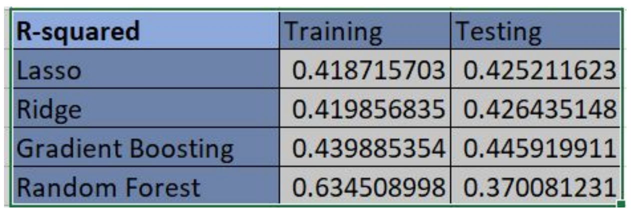
Linear regression with L1 and L2 regularization performed quite similarly. The explanation for this can be given when the coefficients are analysed for these two models - except for a handful of features, all other have very small coefficient values and the coefficients zeroed by L1 regularization were too small to make a difference in results for L2.

Random forest model was a higher variance model. It learned the training data the best out of all the models as seen in R-squared values and RMSE for training. It performed poorly on the test data set. This seems to a case of overfitting, so a gradient boosted regressor was trained with the idea that we might get a higher R-square value and lower RMSE than Linear Regression Models.

The Gradient Boosted Regressor (GBR) did not disappoint. It was observed that the GBR performed the best with the least RMSE and higher R-square on the training and testing data. This model was also the most stable (training and testing RMSEs were closest). For the values, please see table below.



RMSE	Training	Testing
Lasso	$1,746.71	$1,735.40
Ridge	$1,745.75	$1,734.38
Gradient Boosting	$1,728.90	$1,718.10
Random Forest	$1,556.21	$1,780.81

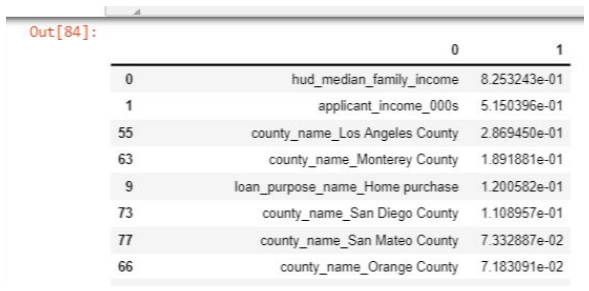


R-squared	Training	Testing
Lasso	0.418715703	0.425211623
Ridge	0.419856835	0.426435148
Gradient Boosting	0.439885354	0.445919911
Random Forest	0.634508998	0.370081231

As seen in the table, practically speaking, the RMSE for the lasso, ridge and GBM were similar, with only a few dollars difference. This difference would not matter for a loan of magnitude of 100,000. As the accuracies were similar, I chose the Ridge regression as it was the an interpretable model and has a lower RMSE than the Lasso and similar R-square values.

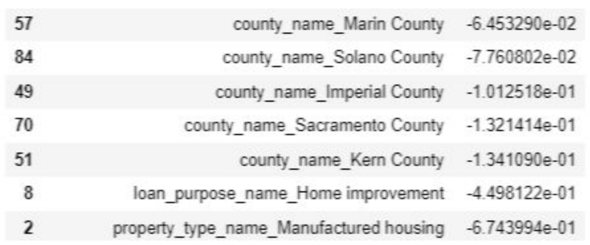
As per the success of the model is considered, for a median loan size of $317,000, the models predicted within an error margin of $1,750 (~0.55%), which is practically a good number. The model coefficients also make sense practically: the most important features are related to personal and family income, type of housing the loan is applied for, purpose of loan (home improvements loans lead to a lower prediction that refinance or purchases.) and some counties are intuitively weighted more heavily (Example: Monterey and Los Angeles have large positive weights, whereas Sacramento county has a large negative weight).

This is a sample of the coefficients with largest absolute values from the model.



Out[84]:

	0	1
0	hud_median_family_income	8.253243e-01
1	applicant_income_000s	5.150396e-01
55	county_name_Los Angeles County	2.869450e-01
63	county_name_Monterey County	1.891881e-01
9	loan_purpose_name_Home purchase	1.200582e-01
73	county_name_San Diego County	1.108957e-01
77	county_name_San Mateo County	7.332887e-02
66	county_name_Orange County	7.183091e-02



57	county_name_Marin County	-6.453290e-02
84	county_name_Solano County	-7.760802e-02
49	county_name_Imperial County	-1.012518e-01
70	county_name_Sacramento County	-1.321414e-01
51	county_name_Kern County	-1.341090e-01
8	loan_purpose_name_Home improvement	-4.498122e-01
2	property_type_name_Manufactured housing	-6.743994e-01

It was also observed that race has very little or low impact on the loan prediction.

Link to Jupyter notebook:
http://nbviewer.jupyter.org/gist/surashish/4c98d4c6396e8a04902bea9d8aefc759

4.4 Maximilian Wühr

CFPB API Integration

I downloaded the LAR dataset for California using their API. The documentation to the API can be found here: https://api.consumerfinance.gov/data/hmda
It was more challenging than I thought, given that we did not have to download data from an API during the course. Since getting the dataset in the first place would have blocked all the others, I asked Michael to help me with it and together we managed to download it.

Source: `download_hdma_data.ipynb`
Concepts: api download, python requests, numpy, pandas, csv

Web Application Development

In the web development task I contributed by creating the SQLite database model, by developing the API endpoint in Flask to retrieve the map data and by adding content to the front end of the web application.

(1) Database model
I developed the database model for the application using SQLAlchemy[10], which is composed of a total of three models:

1. `State`
2. `County`
3. `Census Tract`

The models have a hierarchical relationship to each other: Each `State` has many `Counties`, and each `County` has many `Census Tracts`.

Using a database instead of hardcoded values, allows us to better organize our code base and make changes in the future, as it is quite common that for example census tracts change from one year to another.

Source: https://github.com/FroeMic/DMA18-FinalProject-Server/tree/master/app/models

[10] http://flask-sqlalchemy.pocoo.org/2.3/

(2) API Development
Additionally helped Michael developing the API of the Flask server.

Route	**Methods**	**Description**
`/api/v1/predict`	POST	returns the current version of the api as json
`/api/v1/revision`	GET	returns the revision of the map data in the database as json
`/api/v1/mapdata`	POST	returns the map data requested in the body of the request as json
`/api/v1/predict`	POST	returns the predicted loan sizes for the input configuration requested in the body of the request as json

Source: https://github.com/FroeMic/DMA18-FinalProject-Server/blob/master/app/routes/api.py

4.5 Rohan Kapuria

Exploratory / Statistical Analysis

Summary of Tasks:

- Explored data set to see unique values and counts
- Plotted heatmap of correlation between features
- Plotted various variables using pie charts and bar graphs to see the distribution of the dataset
- See ipython notebook `EDA-California.ipynb` for more details (or download from here: https://drive.google.com/open?id=13nb2i0vUGCsIRG8oQ7hGhw1MHkiAOoOX)

Application concepts and techniques learned in class:

- Concepts learnt in data preprocessing helped in this task
- Tasks performed were similar to those done in Lab 1 of the course
- Plotting methods in Lab 2

Map Development

(1) Aggregating Average Loan Sizes

Summary of Tasks:

- Calculate the average loan amount for each county
- Calculate the average loan amount for each census tract number
- This data was used by Michael to populate the default heat map
- This task involved working with dataframes to group, slice and join datasets
- See ipython notebook `CALIFORNIA_Rohan_Kapuria_Final_project.ipynb` for more details (or download from here:

http://nbviewer.jupyter.org/gist/rohan-kapuria/cf62e5cd804ef76d396d7e03750335a3)

Application concepts and techniques learned in class:

- Concepts learnt in data preprocessing helped in this task
- Tasks performed were similar to those done in Lab 1 of the course

Sample of the final output dataset:
Below are five rows from the output dataframe.

state_ code	state_name	county_ name	Census_ Tract_ number	Avg_loan_ state	avg_loan _county	Avg_loan_ census_num
6	California	Fresno County	53.05	425.33	229.69	486.47
6	California	Placer County	206.01	425.33	357.68	447.89
6	California	Sacramento County	71.05	425.33	276.39	252.21
6	California	Los Angeles County	3020.02	425.33	506.12	497.52
6	California	Los Angeles County	9200.45	425.33	506.12	366.02

(2) Deserializing JSON and Seeding Database

Summary of Tasks:

- I wrote the code that helped seed the database when the application was run the first time.
- Files worked: **app/seed/__init__.py** contained in the seed folder of the following repo https://github.com/FroeMic/DMA18-FinalProject-Server

4.6 Michael Fröhlich

I contributed on three ends to the project. Initially on integrating with the CFPB API and downloading the LAR dataset, after that in our efforts to visualize data on a map and build the project into a full web application. In all steps, I set the secondary goal of writing code that would make it easy to add other states to our application in the future.

CFPB API Integration

At the beginning, I supported Maximilian Wühr in his effort to download data from the CFPB's API. In order to automate this we used python's *request* package. (It was an interesting challenge loading the data over an API, as compared to the lecture, where we always had ready-to-download *.csv* files.)

Source:	`download_hdma_data.ipynb`
Concepts:	preprocessing, pandas, numpy

Map Development

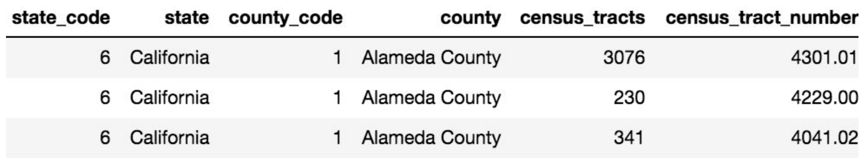
The challenge we faced in the map development stage was largely a data transformation and data fusion challenge. Given two datasets (LAR and US County Shapefiles) we needed to combine their information and transform it into a format that would allows us to display it on a map. We decided to export it as a hierarchical JSON file, as python provides a native API for serializing and deserializing JSON.

(1) Convert Shapefile format into GeoJSON
In order to parse the shapefiles we converted them to GeoJSON format using free online tools such as http://2geojson.com/.

(2) Create Census Lookup Dataframe
To join the map data from the newly created GeoJSON files with the LAR data we needed to match them by key. To avoid loading the entire 1.56GB LAR file each time, we created a lookup data frame containing all keys for counties and census tracts across the United States.

Source:	`export_with_census_tracts.ipynb`
Concepts:	preprocessing, pandas, numpy

Example Output: The first three rows of the California lookup file:

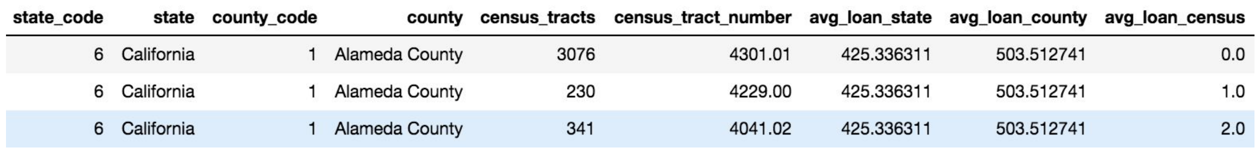


state_code	state	county_code	county	census_tracts	census_tract_number
6	California	1	Alameda County	3076	4301.01
6	California	1	Alameda County	230	4229.00
6	California	1	Alameda County	341	4041.02

(3) Adding Average Loan Size to Lookup Dataframe
With the help of Rohan Kapuria I added the average loan sizes to the lookup file for California. We did so on state, county and census tracts level.

Source: `add_average_data_to_census_lookup.ipynb`
Concepts: preprocessing, dealing with NaN values, pandas, numpy

Example Output: The first three row of the California lookup file with average loan sizes:



state_code	state	county_code	county	census_tracts	census_tract_number	avg_loan_state	avg_loan_county	avg_loan_census
6	California	1	Alameda County	3076	4301.01	425.336311	503.512741	0.0
6	California	1	Alameda County	230	4229.00	425.336311	503.512741	1.0
6	California	1	Alameda County	341	4041.02	425.336311	503.512741	2.0

(4) Prepare Seed JSON
Finally, we joined the geoJSON and the LAR data and serialized into a JSON file.

Source: `prepare_seed_json.ipynb`
Concepts: preprocessing, pandas, numpy, json

(5) Seed Database
After exporting the final seedfile, Rohan Kapuria wrote a script to import it into the DB Schema Maxi created, allowing the Flask server to store and retrieve it in a fast manner.

Note: I wasn't involved in (5). I just wanted to mention this point for completeness

(6) Retrieve Map Data via API
In order to display the generated map data on a map, we integrated Mapbox GLS[11] into the VueJS application and used Axios[12] to dynamically fetch map data from the API. For creating dynamic color ranges depending on the loan sizes displayed we used chroma.js[13].

Web Application Development

To bring all the pieces together and provide a truly useful tool for people without any programming skills, we decided to transform the project into a full web application.

Using Flask and VueJS we completely separated client and server. Flask additionally allowed a relatively easy integration of our prediction model. My tasks involved the majority of the implementation including database modeling, API design, input validation and interface design.

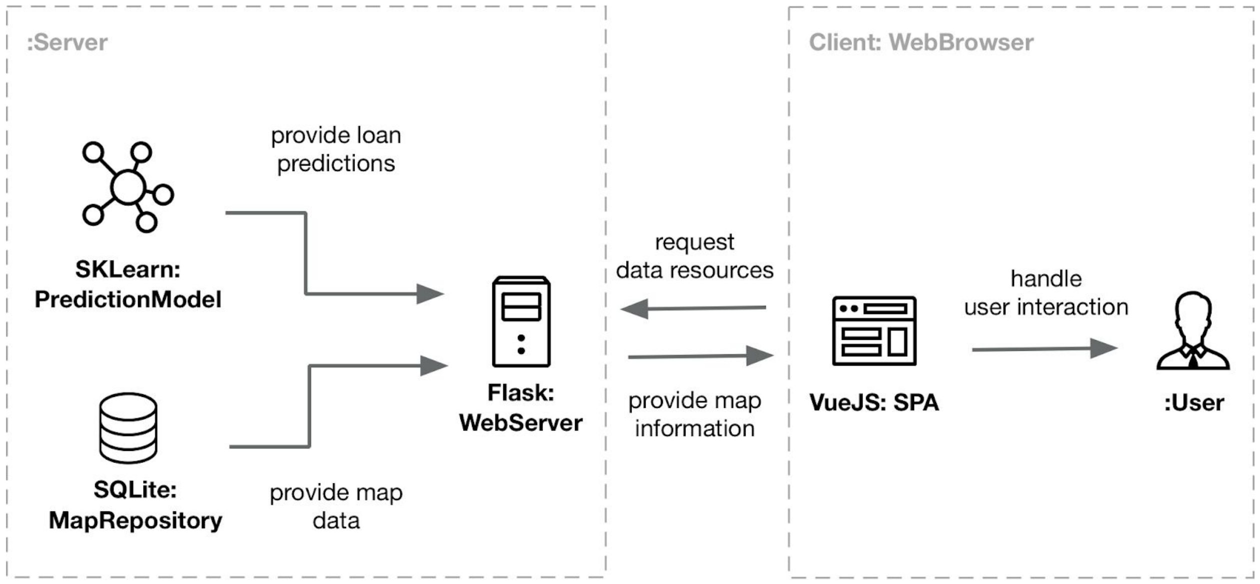
[11] https://www.mapbox.com/mapbox-gl-js/api/

[12] https://github.com/axios/axios

[13] https://gka.github.io/chroma.js/

Rohan Kapuria contributed by seeding the database, Maximilian Wühr during development of the API and Avi Dixit integrated the prediction model into the Flask Server.

The illustration below depicts the most important entities and the architecture of the web application.



The web application server provides an API to the single page application running in the user's browsers. By interacting with the application in the browser, the user requests different resources form the server using the API, i.e. map data, average loan sizes and personalized predictions. The server fetches map data and average loan sizes from a SQLite database. Personalized loan predictions require the additional use of the prediction model.

Source: https://github.com/FroeMic/DMA18-FinalProject-Server
https://github.com/FroeMic/DMA18-FinalProject-SPA

5. Conclusion

The following section discusses the project and the achieved results. It points out real-world implications and lays out future work.

5.1 Discussion

As per the success of the model is considered, for a median loan size of $317,000, the models predicted within an error margin of $1,750 (~0.55%), which is a good practical number and works for ball-park informing a user of the loans he can get in a particular place.

The model coefficients also make sense practically: the most important features are related to personal and family income, type of housing the loan is applied for, purpose of loan (home improvements loans lead to a lower prediction that refinance or purchases).

More expensive counties are intuitively weighted more heavily. Example: Monterey and Los Angeles are large positive weights, whereas Sacramento county has a large negative weights.

Overall, a good, stable and generalizable model for open use on the internet. It is also highly interpretable given regulations that financial models must be such.

5.2 Real-World Implications

We see two specific ways in which this project may affect citizens of California.

(1) Improved accessibility and visibility

The CFPB provides public access to the LAR data. However, for people without programming and data analytic skills this information is still not accessible. Visualizing the most important descriptive information — the average size of approved home loans — per county is a first step towards giving ordinary citizens access to it.

(2) Improved information for home loan borrowers

The second benefit for citizens comes from the prediction of the to be expected loan amounts. Previously, people interested in buying a house could not easily reason about how much of a loan the can expect. Especially, for first-time buyers or people moving to the US this lack of information slowed down plannability and the decision making process all together.

Our solution provides a straightforward interface allowing users to playfully explore how much of a loan the can expect to get.
Young graduates, for example, may use the tool to figure out the salary they need to be approved for the loan in their favorite area. Subsequently, they can use the additional

knowledge to drive salary negotiations or simply plan for the long term.
Similarly, professionals moving to a job in San Francisco can use the visualization on the map to intuitively grasp the housing market around the bay area. With this understanding they can focus their energy on searching for properties in the counties around San Francisco that matches their willingness to pay.

5.3 Future Work

The following subsections describes future steps to improve on the existing solution.

(1) Extend to all US states

The current solution does only provide visualization for the state of California. Extending the existing solution to all US states would be a clear improvement as it would allow citizens from all over the US to visualize the are relevant to them.
Adding new states requires several different steps, including fetching the loan data from the CFPB's API, creating the shapefiles for the map and the training the model. While we designed our code in a way that allows for easy automation of the first two steps, the third still requires human supervision. Therefore, we recommend to add states iteratively, possibly even without the prediction first.

(2) Extend to census tract detail level

Another improvement would be to increase the granularity of the map by going from a per-county-prediction to per-census-tract prediction. Given that the number of census tracts is significantly higher than the number of counties per state, there challenges to be expected: First, the interactive performance and usability of the map will suffer given the current implementation. Second, the model accuracy may diverge between census tracts with an active property market and those where only few training samples exist.

(3) Iteratively improve model accuracy

While we are confident that — given the current RSME of ~ 1750 USD — the developed model provides meaningful predictions, there is still room for improvement. Particularly, training separate models per county instead of using one model for all counties may improve the accuracy. Besides, gradually adding more features to the model may equally increase accuracy.

(4) Retrain models with recent LAR data

It is expected that the CFPB releases the 2017 LAR dataset by September 2018. To keep the visualization meaningful, the map and the prediction model(s) should be updated / retrained with the most recent dataset.

APPENDIX

A. Field Reference

Field name	Description	Data type
`action_taken`	A code representing the action taken on the loan or application, such as whether an application was approved or denied. Loan originated means the application resulted in a mortgage. Loan purchased means that the lender bought the loan on the secondary market.	integer
`action_taken_name`	A description of the action taken.	string
`agency_abbr`	The abbreviated name of the federal agency.	string
`agency_code`	A code representing the federal agency to which the HMDA-reporting institution submits its HMDA data.	integer
`agency_name`	The full name of the federal agency.	string
`applicant_ethnicity`	A code representing the ethnicity of the primary applicant.	integer
`applicant_ethnicity_name`	The ethnicity of the primary applicant.	string
`applicant_income_000s`	The gross annual income that the lender relied on when evaluating the creditworthiness of the applicant, rounded to the nearest thousand.	integer
`applicant_race_1`	A code representing the first listed race for the primary applicant. The applicant can list up to five races.	integer
`applicant_race_2`	A code representing the second listed race for the primary applicant.	integer
`applicant_race_3`	A code representing the third listed race for the primary applicant.	integer
`applicant_race_4`	A code representing the fourth listed race for the primary applicant.	integer
`applicant_race_5`	A code representing the fifth listed race for the primary applicant.	integer

applicant_race_name_1	The first listed race for the primary applicant. The applicant can list up to five races.	string
applicant_race_name_2	The second listed race for the primary applicant.	string
applicant_race_name_3	The third listed race for the primary applicant.	string
applicant_race_name_4	The fourth listed race for the primary applicant.	string
applicant_race_name_5	The fifth listed race for the primary applicant.	string
applicant_sex	A code representing the sex of the primary applicant.	integer
applicant_sex_name	The sex of the primary applicant.	string
application_date_indicator	A code representing the date of the application. "0" means the application was made on or after 1/1/2004; "1" means the application was made before 1/1/2004; "2" means the application date is not available.	integer
as_of_year	The reporting year of the HMDA record.	integer
census_tract_number	The number of the census tract for the property. This code is only unique when combined with the state and county codes.	string
co_applicant_ethnicity	A code representing the ethnicity of the co-applicant.	integer
co_applicant_ethnicity_name	The ethnicity of the co-applicant.	integer
co_applicant_race_1	A code representing the first listed race for the co-applicant. The co-applicant can list up to five races.	integer
co_applicant_race_2	A code representing the second listed race for the co-applicant.	integer
co_applicant_race_3	A code representing the third listed race for the co-applicant.	integer
co_applicant_race_4	A code representing the fourth listed race for the co-applicant.	integer
co_applicant_race_5	A code representing the fifth listed race for the co-applicant.	integer
co_applicant_race_name_1	The first listed race for the co-applicant. The co-applicant can list up to five races.	string

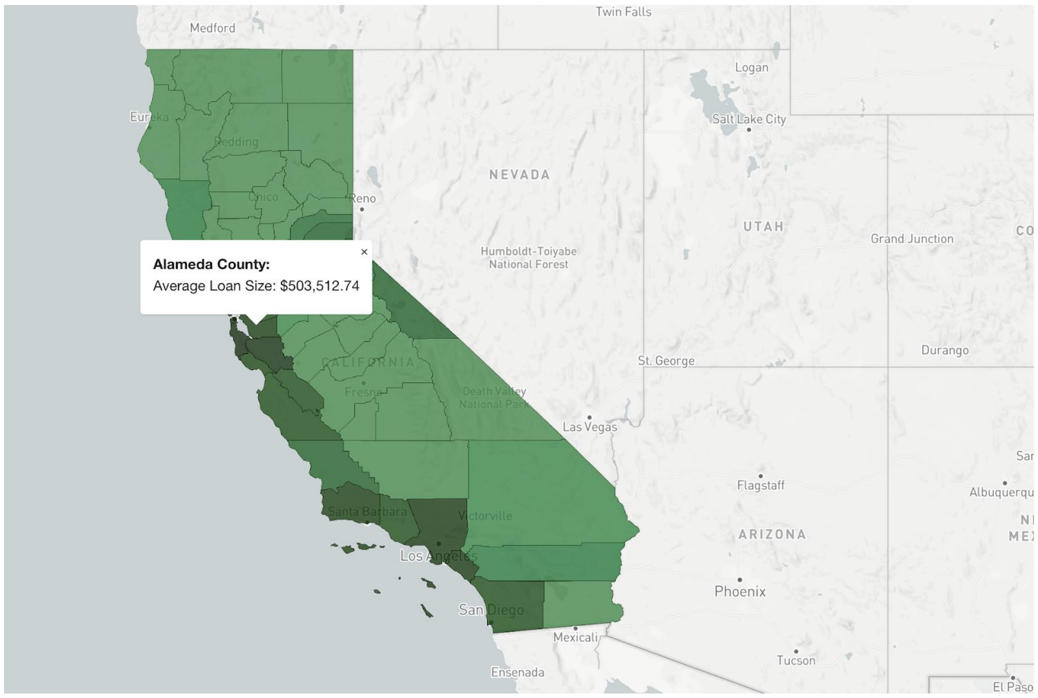
co_applicant_race_name_2	The second listed race for the co-applicant.	string
co_applicant_race_name_3	The third listed race for the co-applicant.	string
co_applicant_race_name_4	The fourth listed race for the co-applicant.	string
co_applicant_race_name_5	The fifth listed race for the co-applicant.	string
co_applicant_sex	A code representing the sex of the co-applicant.	integer
co_applicant_sex_name	The sex of the co-applicant.	string
county_code	A three-digit code representing the county of the property. This code is only unique when combined with the state code.	integer
county_name	The name of the county containing the property. Some counties have the same name, so we recommend combining this with state.	string
denial_reason_1	A code representing the first reason for denial of the application. Lenders may report up to three denial reasons, but such reporting is optional.	integer
denial_reason_2	A code representing the second reason for denial of the application.	integer
denial_reason_3	A code representing the third reason for denial of the application.	integer
denial_reason_name_1	The first reason for denial of the application. Lenders may report up to three denial reasons, but such reporting is optional.	string
denial_reason_name_2	The second reason for denial of the application.	string
denial_reason_name_3	The third reason for denial of the application.	string
edit_status	A code representing the edit failure status of the application.	integer
edit_status_name	The edit failure status of the application.	string
hoepa_status	A code representing whether a loan is subject to the Home Ownership and Equity Protection Act of 1994 (HOEPA).	integer

hoepa_status_name	The HOEPA status of the application.	string
hud_median_family_income	The median family income in dollars for the MSA/MD in which the tract is located.	integer
lien_status	A code representing the lien status. Most mortgages are secured by a lien against the property. In the event of a forced liquidation, first lien holders will generally get paid before subordinate lien holders.	integer
lien_status_name	The lien status.	string
loan_amount_000s	The amount of the loan applied for, in thousands of dollars.	integer
loan_purpose	A code representing the purpose of the loan (home purchase, refinance, or home improvement).	integer
loan_purpose_name	The purpose of the loan.	string
loan_type	A code representing the type of loan applied for. Many loans are insured or guaranteed by government programs offered by Federal Housing Administration (FHA), the Department of Veterans Affairs (VA), or the Department of Agriculture's Rural Housing Service (RHS) or Farm Service Agency (FSA). All other loans are classified as conventional.	integer
loan_type_name	The type of loan applied for.	string
minority_population	The percentage of minority population to total population for the census tract, carried to two decimal places.	number
msamd	A code representing the Metropolitian Statistical Area/Metropolitian Division (MSA/MD) the property is located in. An MSA is a region with relatively high population density at its core (usually a single large city) and close economic ties throughout. Larger MSAs are divided into MDs.	undefined
msamd_name	The Metropolitian Statistical Area/Metropolitian Division (MSA/MD) the property is located in.	string
number_of_1_to_4_family_units	The number of dwellings in the tract that are built to house fewer than 5 families.	integer
number_of_owner_occupied_units	The number of dwellings in the tract that are lived in by the owner.	integer

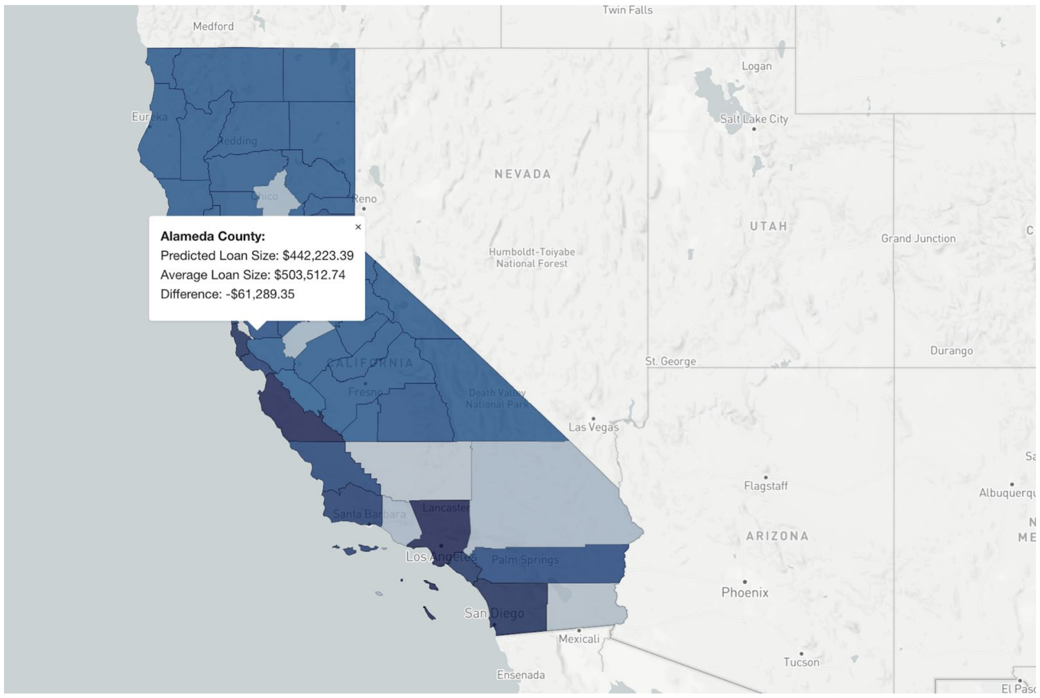
`owner_occupancy`	A code representing the owner-occupancy status of the property. Second homes, vacation homes, and rental properties are classified as "not owner-occupied as a principal dwelling".	integer
`owner_occupancy_name`	The owner-occupancy status of the property.	string
`population`	The total population in the tract.	integer
`preapproval`	A code representing the pre-approval status of the application.	integer
`preapproval_name`	The pre-approval status of the application.	string
`property_type`	A code representing the type of the property.	integer
`property_type_name`	The type of the property.	string
`purchaser_type`	A code representing the type of institution purchasing the loan.	integer
`purchaser_type_name`	The type of institution purchasing the loan.	string
`rate_spread`	The rate spread for the loan, which is the difference between the loan's annual percentage rate (APR) and the average prime offer rate (APOR).	number
`respondent_id`	A code representing the bank or other financial institution that is reporting the loan or application.	integer
`sequence_number`	A one-up number scheme for each respondent to make each loan unique.	string
`state_abbr`	The two-letter abbreviation for the state the property is located in.	undefined
`state_code`	A two-digit code representing the state the property is located in.	integer
`state_name`	The name of the state the property is located in.	string
`tract_to_msamd_income`	The percentage of the median family income for the tract compared to the median family income for the MSA/MD, rounded to two decimal places.	number

B. Map Visualization

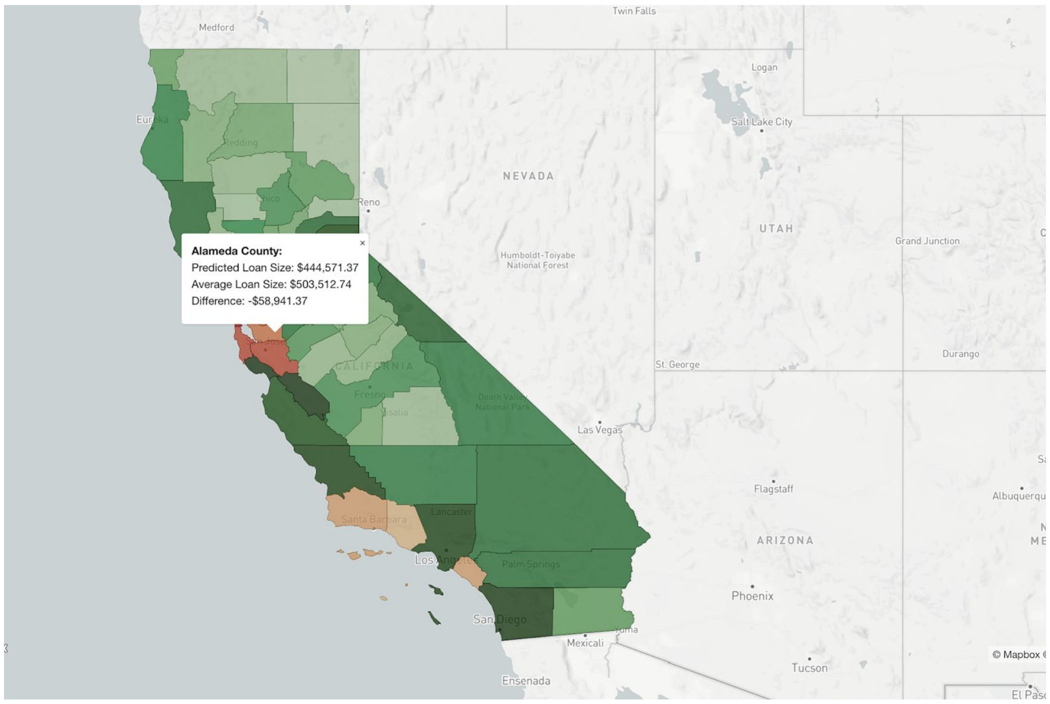
A map showing the average loan amount per county. (darker means higher)



A map showing the predicted loan amount per county. (darker means higher)



A map showing the normalized difference between average and predicted loan amount per county. (red means that the predicted loan amount is lower than the average, green means higher)


Alameda County:
Predicted Loan Size: $444,571.37
Average Loan Size: $503,512.74
Difference: -$58,941.37