CS398 Applied cloud computing final project

May 2, 2018

Group name:	tql
Group members:	Chuchao Luo
	Ruiyang Chen
	Ziang Wan
	Xiaoping Hua
Gitlab link:	https://gitlab.engr.illinois.edu/cs398-sp18-final-project/tql

1 Target

In this project, we are trying to create a sophisticated model that given several characteristics of a property, predicts its value after tax. We also want to exploit the data to find some interesting patterns behind the data.

2 Cloud Computing Framework

We use Google cloud platform to perform cloud computing. Google Dataproc and Google Cloud ML are significant parts in our project.

2.1 Definitions

Google Dataproc A fast and safe spark cluster service offered by Google. The size of cluster will scale up and down according to workload automatically.

Google Cloud ML A online machine learning service offered by Google. It takes in a python package which contains the model and other dependencies, and trains the model with designated data set in various location: local, Google Cloud storage buckets etc. The trained model will be dumped to Google Cloud storage buckets.

2.2 Workflow

The raw data has many NA value and about 60 features. Therefore data cleaning is necessary otherwise the model will be very large and tend to overfit. We used Google Dataproc to clean data.

- Select important features
- Fill reasonable value to some NA value, and drop all other NA values.
- Export new dataset

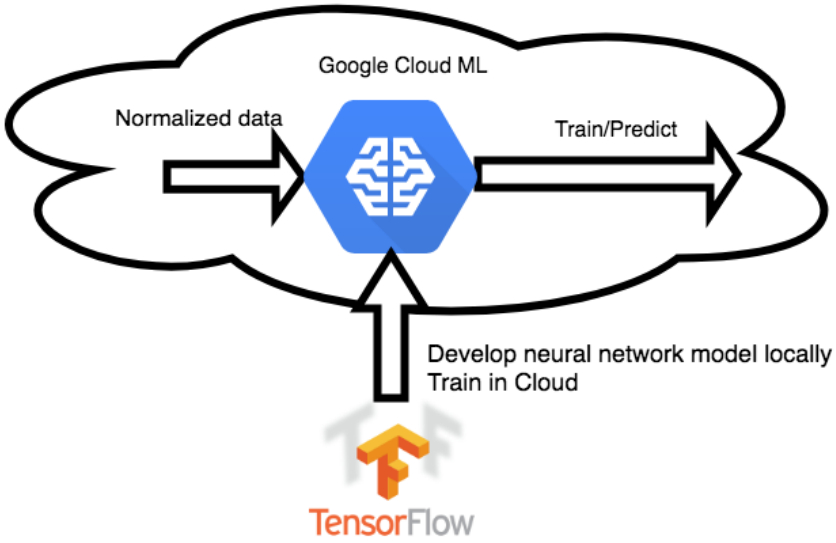
Relative files:

- scripts/preprocessing.py (defines a pyspark job)
- makefile (script to submit jobs to Google Dataproc)

Once we get the cleaned and normalized dataset. We need to come up with a regression neural network model. We implemented the model with Tensorflow locally and submitted it to Google Cloud ML to train and predict.

Relative files:

- trainer/model1.py (defines the neural network model)
- trainer/task.py (defines the job to be run by Google Cloud ML)
- makefile (script to submit python package to cloud ML)
- setup.py (defines dependencies)
- others



3 Data Overview

We got the original data from Kaggle competition: Zillow Prize: Zillows Home Value Prediction (Zestimate). **Note:** The original competition is to minimize the log error of Zestimate and real price. However we want to do our estimate.

3.1 Cleaned dataset

The dataset comes with more than 50 properties, with a lot of empty or problematic values. We extracted 11 of the properties for training:

- air conditioning typeid
- bathroom count
- bedroom count
- building quality typeid
- calculated bathroom number
- calculated finished square feet
- FIPS county code
- zip code
- room count
- year built
- tax amount

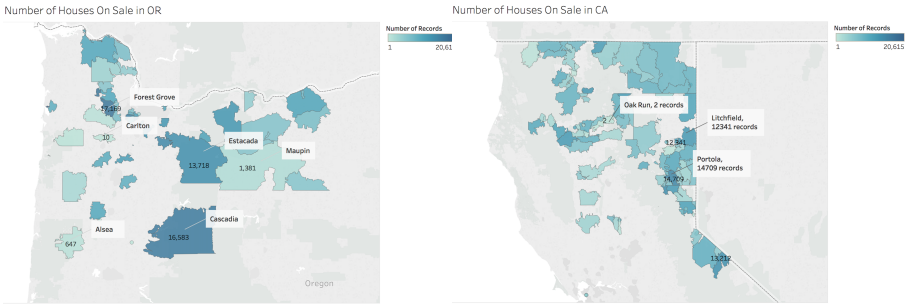
3.2 Visualization and interesting patterns

We selected a few properties from the complete dataset (properties_2017.csv) and did some visualizations on them.

3.2.1 Number of houses in the record of CA and OR

Properties Used:

- Zip code
- Distinct count
- Filter: non-null values



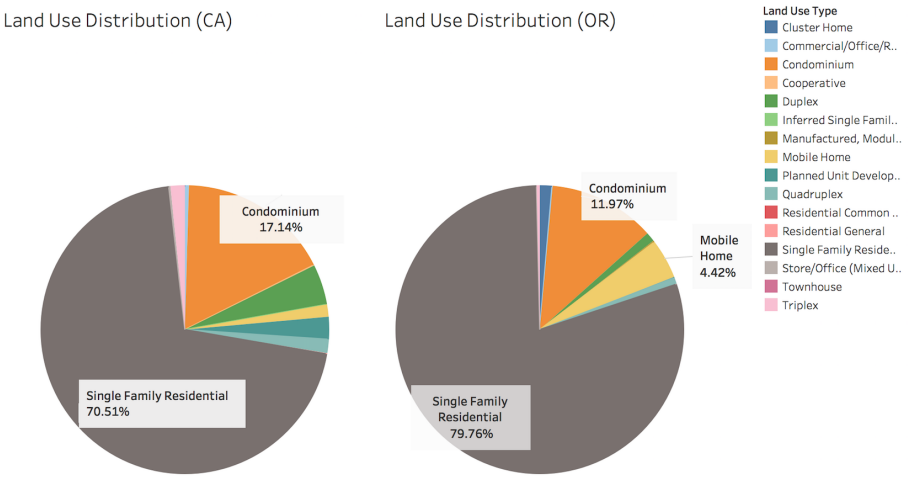
Note: Both graphs are of the same scale.

Comparing the number of houses in OR and CA, we observed that in OR, larger amount of records usually occurs in larger land area. While in CA, larger amount of records usually occurs in smaller land area. Therefore, we assume that the houses are more spread out in OR than in CA.
Based on this fact, we made a guess that there are less condominiums and more single family residential in OR than in CA.

3.2.2 Land Use Distribution in CA and OR

Properties Used:

- Zip code (to distinguish between CA and OR)
- Land use type
- Distinct count
- Filter: non-null values

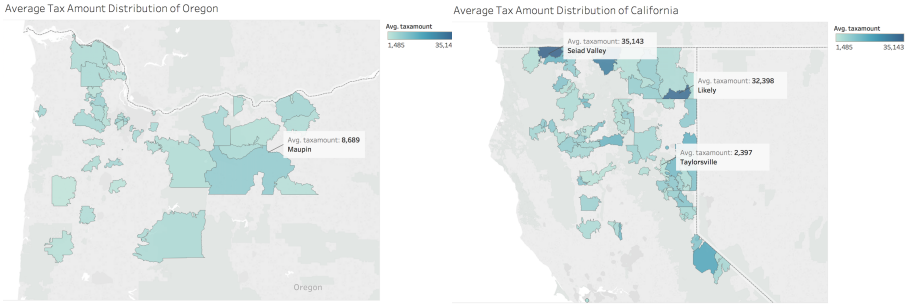


This graph shows that OR has a larger proportion of single family residential and smaller proportion of condominiums among all houses in the record.
This confirms that our guess above ("there are less condominiums and more single family residential in OR than in CA") is true.

3.2.3 Tax Amount in CA and OR

Properties Used:

- Zip code
- Average tax amount collected in each zip code



Note: Both graphs are of the same scale.

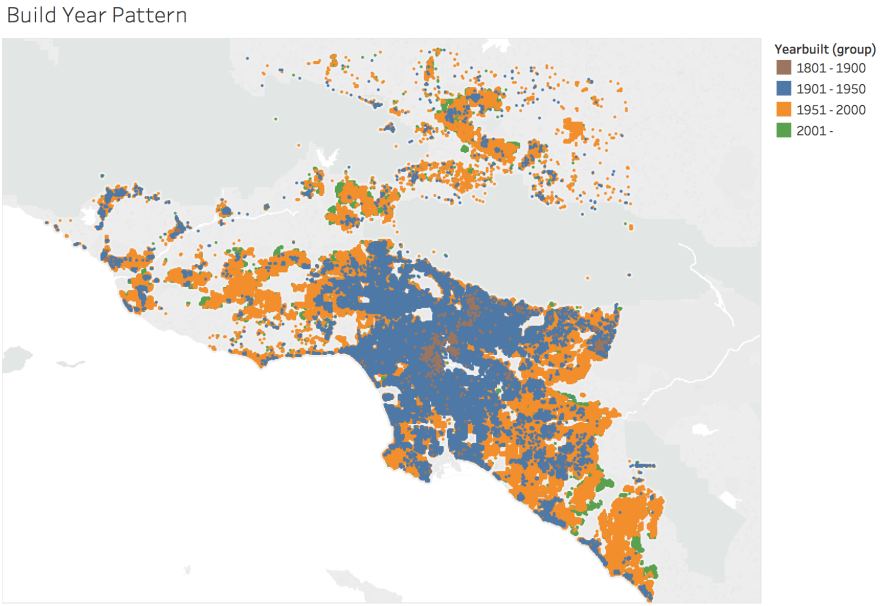
By researching roughly about property tax in both states, we found that the property tax rate of CA is slightly higher than the tax rate in OR. We would like to confirm this by visualizing the overall tax amount of both states.

By mapping average tax amount in each area, and comparing between OR and CA, we observed that overall, CA has a higher tax amount (darker color) than OR.
We also observed an interesting point that the highest tax amount occurs in Seiad Valley, CA, where Seiad Valley is a small town with a population of only 300 people.

3.2.4 House Build Year Pattern in CA

Properties Used:

- Year built
- Latitude and longitude of houses
- Filter: non-null values, CA-only values



We sectioned build years of houses into 4 groups: 1801-1900, 1901-1950, 1951-2000, 2001-now (assume 2017). By plotting houses with their build years onto the map, we observed that in the past 200 years, the house were built in a radial pattern: they gradually spread outward from the center of the land. Also, we found that most of the houses on sale were built before 2000.

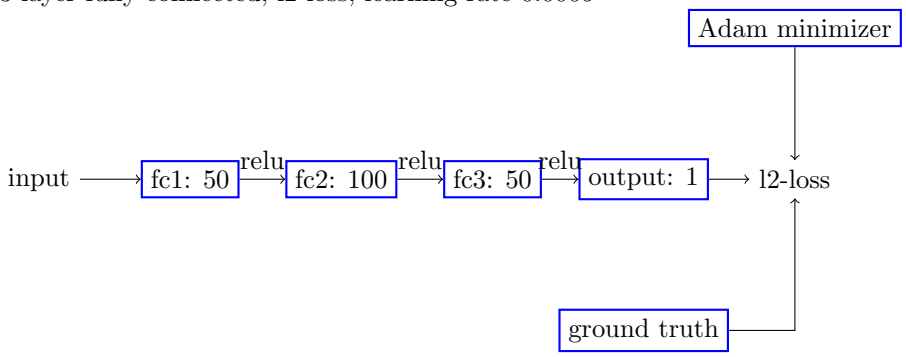
4 Model

We trained 6 different models and got the result. In this section, we will compare the accuracy of these models

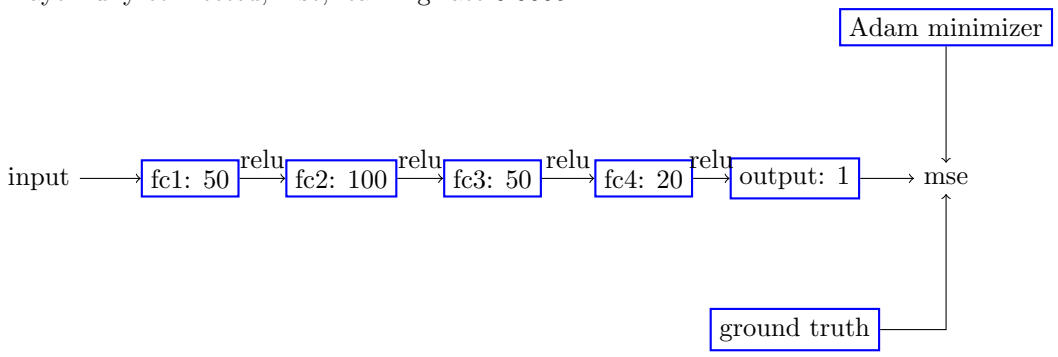
train dataset size	$\approx$ 125MB
test dataset size	$\approx$ 125MB
epoch	10
learning-rate	$0.0001 \sim 0.0005$
activation function	relu

Note: fc=fully connected layer
Note: Our model is insensitive to the size of input data, and can be easily generalized to larger datasets.

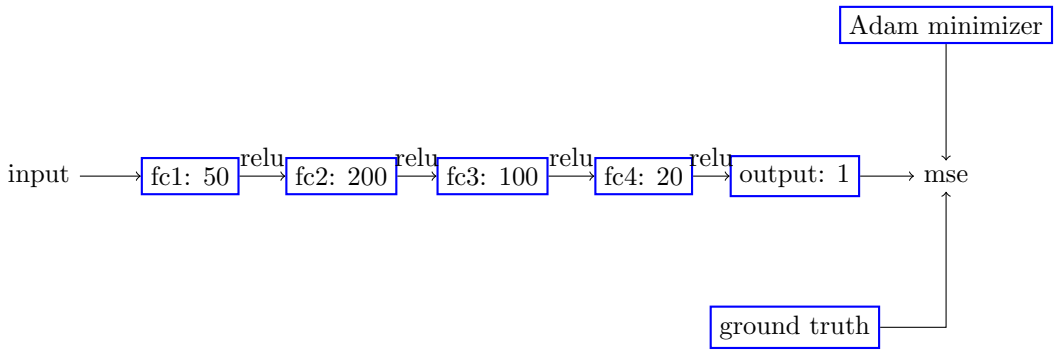
a. 3-layer fully connected, l2-loss, learning-rate 0.0005



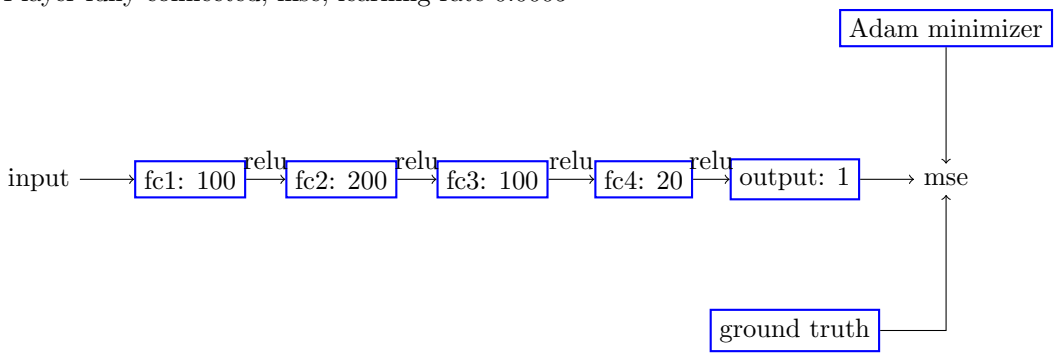
b. 4-layer fully connected, mse, learning-rate 0.0005



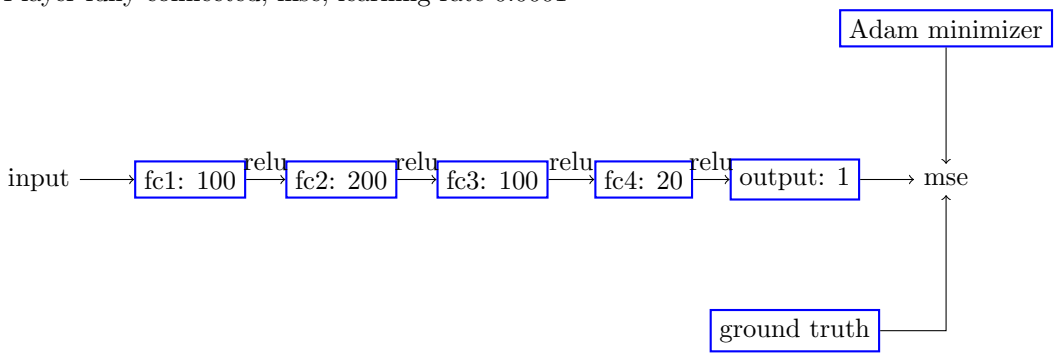
c. 4-layer fully connected, mse, learning-rate 0.0005



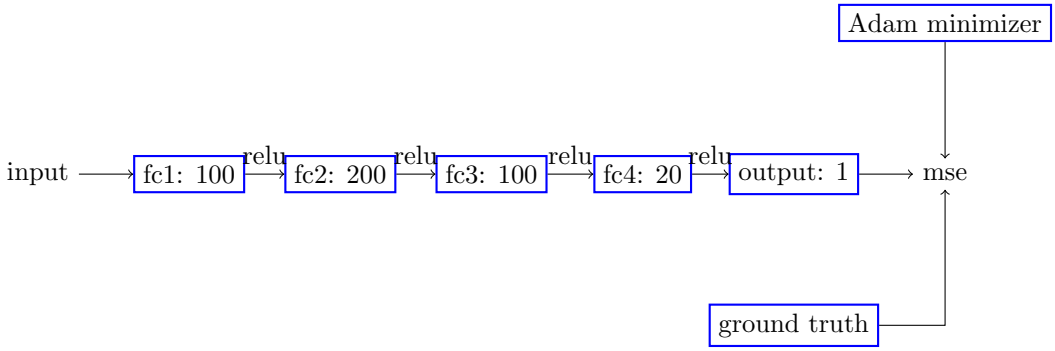
d. 4-layer fully connected, mse, learning-rate 0.0005



e. 4-layer fully connected, mse, learning-rate 0.0001



f. 4-layer fully connected, mse, learning-rate 0.0001, add *lotsizesquarefeet* feature

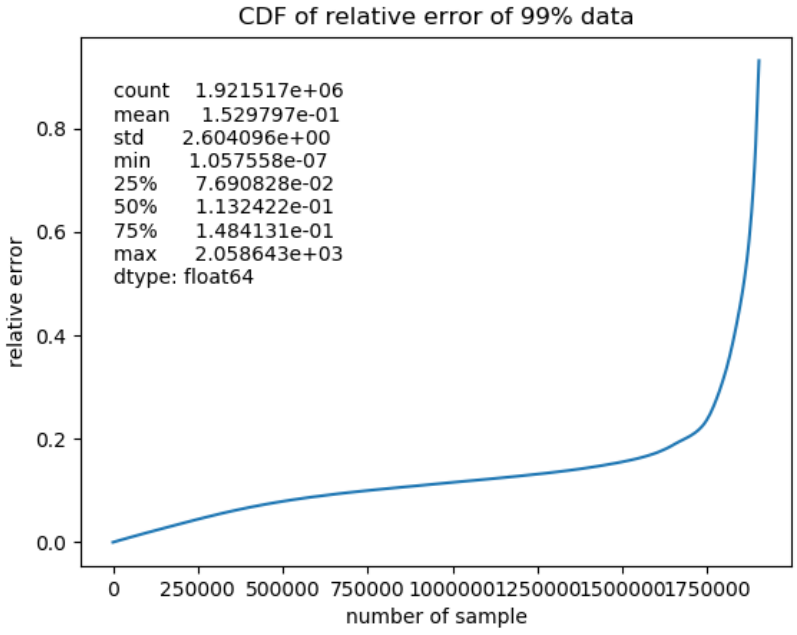


5 Result and Analyze

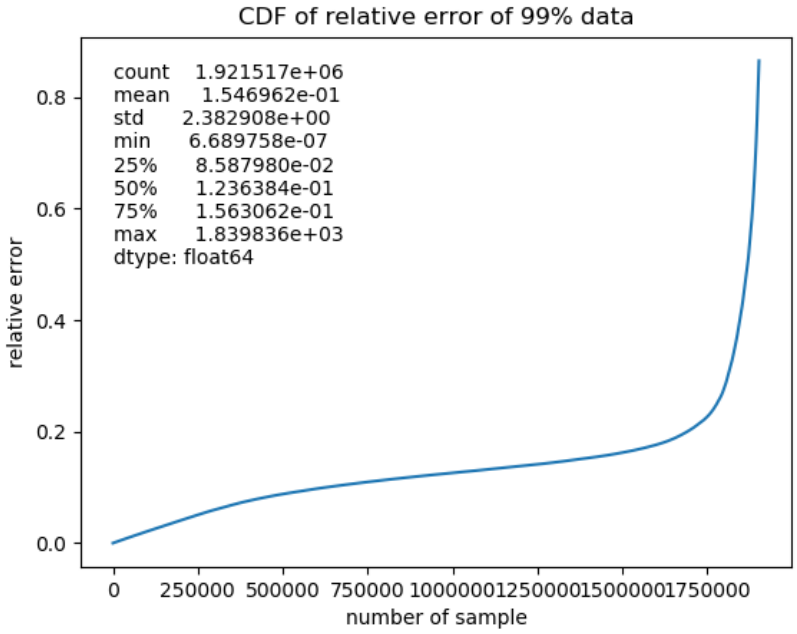
5.1 Relative error

$$\text{relative error} = \frac{|\text{predict} - \text{ground truth}|}{\text{ground truth}}$$

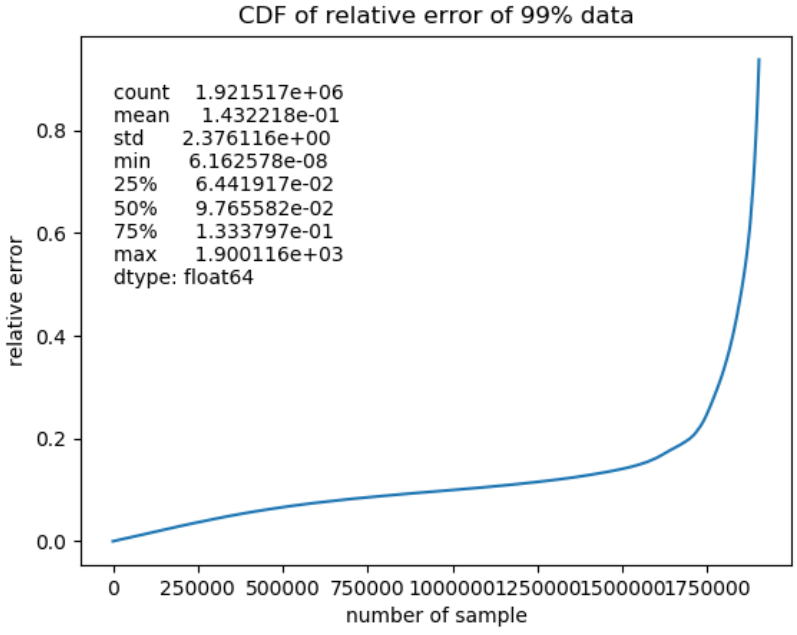
Overall, the accuracy is decent, 75% of predictions are ubder 15% relative error. However about $1\% \sim 2\%$ of the predictions have very high error.



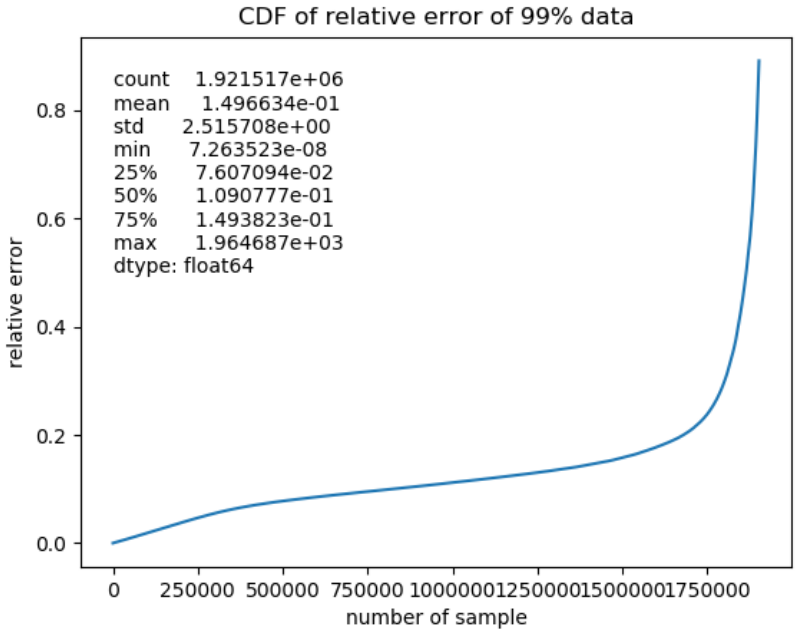
a.


CDF of relative error of 99% data
count 1.921517e+06
mean 1.546962e-01
std 2.382908e+00
min 6.689758e-07
25% 8.587980e-02
50% 1.236384e-01
75% 1.563062e-01
max 1.839836e+03
dtype: float64
relative error
0.0
0.2
0.4
0.6
0.8
0
250000
500000
750000
1000000
1250000
1500000
1750000
number of sample

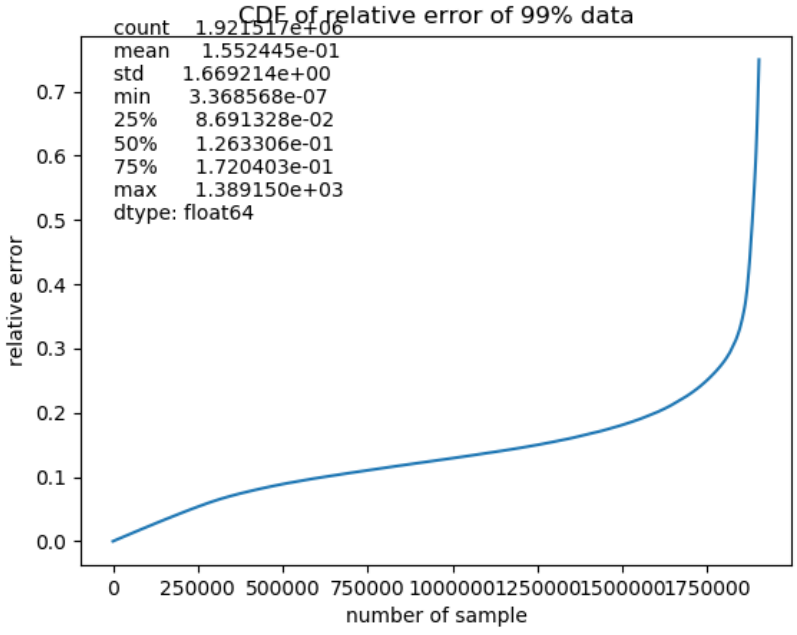
b.


CDF of relative error of 99% data
count 1.921517e+06
mean 1.432218e-01
std 2.376116e+00
min 6.162578e-08
25% 6.441917e-02
50% 9.765582e-02
75% 1.333797e-01
max 1.900116e+03
dtype: float64
relative error
0.0
0.2
0.4
0.6
0.8
0
250000
500000
750000
1000000
1250000
1500000
1750000
number of sample

c.

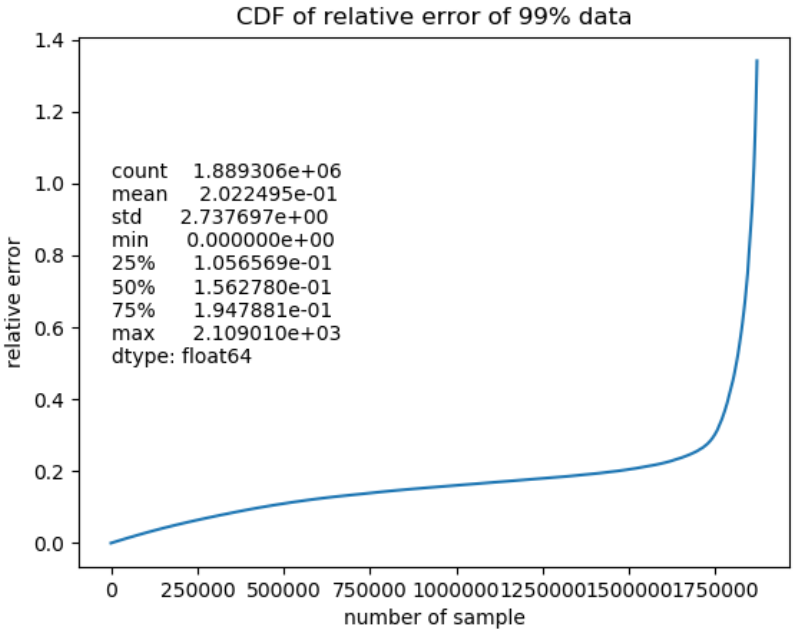

CDF of relative error of 99% data
count 1.921517e+06
mean 1.496634e-01
std 2.515708e+00
min 7.263523e-08
25% 7.607094e-02
50% 1.090777e-01
75% 1.493823e-01
max 1.964687e+03
dtype: float64
relative error
0.0
0.2
0.4
0.6
0.8
0
250000
500000
750000
1000000
1250000
1500000
1750000
number of sample

d.

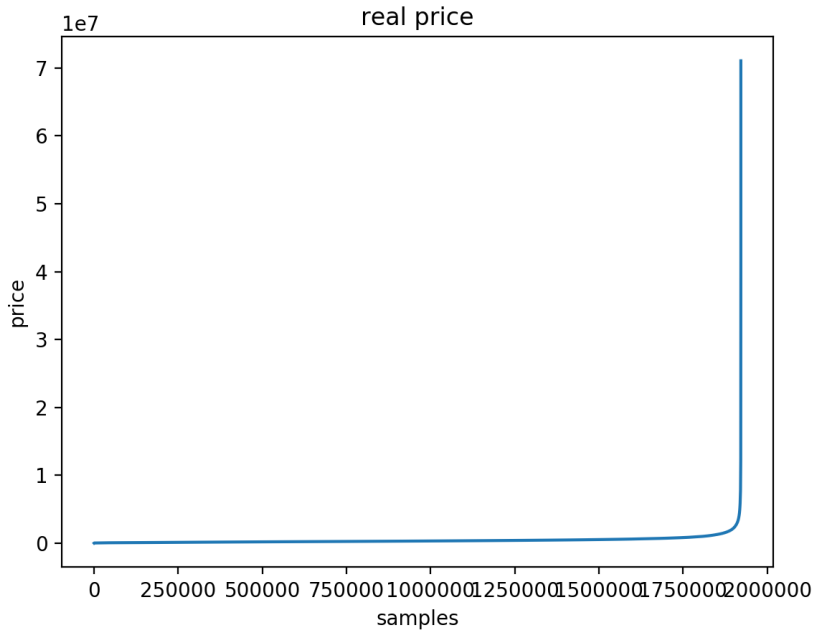

CDF of relative error of 99% data
count 1.921517e+06
mean 1.552445e-01
std 1.669214e+00
min 3.368568e-07
25% 8.691328e-02
50% 1.263306e-01
75% 1.720403e-01
max 1.389150e+03
dtype: float64
relative error
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
250000
500000
750000
1000000
1250000
1500000
1750000
number of sample

e.

f.



After inspecting the original data, there exists some significant abnormal price. Some prices are pretty small. For example, there are houses that only sold for 32 dollars and houses with more than 200 bedrooms but sold for around 100k dollars. For those extremely low price, the relative error is generally quite huge. Our guess is that if we remove those outlier values, the overall accuracy can be improved and our neural network can predict better.



Relevant files:

- scripts/analyzes_results.py

6 Performance

We achieved around 6-10x speed up by using the Google ML engine to train our neural network. Because the neural network has more than 20 layers, it will take around 60 minutes to train on a local CPU (3 miniutes for each epoch), around 6 minutes to train on Cloud. When we trained locally, we had to gather our RAM cards on to one machine to get 64GB RAM. If we don't do so, the memory will overflow. However, on GCP ML, we do not need to worry about memory.

7 The Hardness We Faced

- It is hard to decide which features to use, what we have is our own experience
- Initially we try to run a linear regression between attributes and house prices on AWS cluster. However, the model does not work and we get NaN as our results. The code is in linear_regression_attempt/prog.py

- The neural network is hard to train, because it has much more parameters compared to linear regression and other models.
- We decide to use GCP ML Engine to train our model because we are using tensorflow, and it turns out that the GCP API is really hard to use.