PROJECT REPORT

An Luo	axl162930
Lizhong Zhang	lxz160730
Jiangyue Xi	jxx160130

AUGUST 7, 2017

CS 6350 BIG DATA MANAGEMENT AND ANALYTICS

The University of Texas at Dallas

Contents

1. Introduction

1.1 Background

During a person's life, a home often costs most. Zillow is an online real estate database company, which provides the information of the homes you are looking for and estimates the home value. Zillow company created Zestimate prediction system which is based on hundreds of millions house datasets, using specific algorithm and combining each home's characteristics and market conditions to make an initial estimation of home's value.

In order to improve the accuracy of the Zestimate system and give people a more trusted way to estimate home price, Zillow holds a "Zillow Prize" competition during 2017. Our project is working on the first round of this competition.

1.2 Problem Description

The goal of our project is to predict the log-error between the estimated result and the actual sale price of the home.

First, we perform feature analysis using statistical methods and select the appropriate features in the dataset. Second, during data preprocessing, we format and clean the new dataset preparing for machine learning. For the training and prediction part, we use random forest regression and gradient-boosted tree regression and then make a comparison between both methods. We will choose a better method and obtain a final model to do predictions.

In this project, we will use two methods to build model and perform prediction. The first one is Dataframe-based model building, the second is RDD-based model building.

2. Related Works

Home value estimation is always a popular problem. So lots of works have been done previously. Hu G, et al. [1] attempt to use home datasets to build multivariate regression models and apply maximum information coefficient statistics based home price and predicted values to evaluate regression models. Nghiep, et al. [2] compared two methods artificial neural networks (ANN) and multiple regression analysis (MRA), according to the prediction performance, ANN does better than MRA when a moderate to large data sample size is used.

In recent years, we found that using Random Forests and Gradient-Boosted Trees method in the training part can get a more satisfactory result especially in Kaggle Competition.

3. Dataset and Language Description

3.1 Dataset Introduction

In our project, we use two datasets to predict the log-error between the estimated result and the actual sale price of the home.

The log-error is defined as:

$$\text{logerror} = \log(\text{Zestimate}) - \log(\text{SalePrice})$$

Dataset 1—properties_2016.csv, including the following:

- All the properties for 2016
- 1 unique identify number and 57 home features
- 2985218 row (properties)

Table 3-1 Features Description in properties_2016.csv

	Old Name	Formatted Name	Data Type	Description
1	parcelid	id_parcel	Int	Unique identifier for parcels
2	airconditioningtypeid	aircon	Str	Type of cooling system
3	architecturalstyletypeid	architectural_style	Str	Architectural style
4	basementsqft	area_basement	Int	Finished living area below/partially below ground
5	bathroomcnt	num_bathroom	Int	Number of bathrooms in home
6	bedroomcnt	num_bedroom	Int	Number of bedrooms in home
7	buildingclasstypeid	framing	Str	The building framing type
8	buildingqualitytypeid	quality	Str	Overall assessment of condition of building
9	calculatedbathnbr	num_bathroom_calc	Int	Number of bathrooms in home
10	decktypeid	deck	Str	Type of deck present on parcel
11	finishedfloor1squarefeet	area_firstfloor_finished	Int	Size of the finished living area on the first floor
12	calculatedfinishedsquarefeet	area_total_calc	Int	Total finished living area of the home
13	finishedsquarefeet12	area_live_finished	Int	Finished living area
14	finishedsquarefeet13	area_liveperi_finished	Int	Perimeter living area
15	finishedsquarefeet15	area_total_finished	Int	Total area
16	finishedsquarefeet50	area_unknown	Int	Size of the finished living area on first floor
17	finishedsquarefeet6	area_base	Int	Base unfinished and finished area
18	fips	no_fip	Str	Federal Information Processing Standard Code
19	fireplacecnt	num_fireplace	Int	Number of fireplaces in a home (if any)
20	fullbathcnt	num_bath	Int	Number of full bathrooms in home
21	garagecarcnt	num_garage	Int	Total number of garages on the lot
22	garagetotalsqft	area_garage	Int	Total number of square feet of all garages
23	hashottuborspa	flag_tub	Str	Does the home have a hot tub or spa
24	heatingorsystemtypeid	heating	Str	Type of home heating system
25	latitude	latitude	Int	Latitude of middle of parcel multiplied by 10E6
26	longitude	longitude	Int	Longitude of middle of parcel multiplied by 10E6
27	lotsizesquarefeet	area_lot	Int	Area of the lot in square feet
28	poolcnt	num_pool	Int	Number of pools on the lot (if any)

29	poolsizesum	area_pool	Int	Total square footage of all pools on property
30	pooltypeid10	s_h	Str	Spa or Hot Tub
31	pooltypeid2	pool_sh	Str	Pool with Spa/Hot Tub
32	pooltypeid7	pool_nosh	Str	Pool without hot tub
33	propertycountylandusecode	zoning_landuse_county	Str	County land use code the county
34	propertylandusetypeid	zoning_landuse	Str	Type of land use the property is zoned for
35	propertyzoningdesc	zoning_property	Str	Description of the allowed land uses
36	rawcensustractandblock	rawcensus	Int	Census tract and block ID combined
37	regionidcity	region_city	Str	City in which the property is located
38	regionidcounty	region_county	Str	County in which the property is located
39	regionidneighborhood	region_neighbor	Str	Neighborhood of property
40	regionidzip	region_zip	Str	Zip code in which the property is located
41	roomcnt	num_room	Int	Total number of rooms
42	storytypeid	story	Int	Type of floors in a multi-story house
43	threequarterbathnbr	num_75_bath	Int	Number of 3/4 bathrooms in house
44	typeconstructiontypeid	material	Str	Type of construction material
45	unitcnt	num_unit	Int	Number of units
46	yardbuildingsqft17	area_patio	Int	Patio in yard
47	yardbuildingsqft26	area_shed	Int	Storage shed/building in yard
48	yearbuilt	build_year	Str	The Year the principal residence was built
49	numberofstories	num_story	Int	Number of stories
50	fireplaceflag	flag_fireplace	Str	Is a fireplace in this home
51	structuretaxvaluedollarcnt	tax_building	Int	The assessed value of the built structure
52	taxvaluedollarcnt	tax_total	Int	The total tax assessed value of the parcel
53	assessmentyear	tax_year	Int	The year of the property tax assessment
54	landtaxvaluedollarcnt	tax_land	Int	The assessed value of the land area
55	taxamount	tax_property	Int	The total property tax assessed for assessment year
56	taxdelinquencyflag	tax_delinquency	Int	Property taxes for this parcel are past due 2015
57	taxdelinquencyyear	tax_delinquency_year	Int	Year for the unpaid property taxes were due
58	censustractandblock	census	Int	Census tract and block ID combined

Dataset 2—train_2016.csv, including the following:

- All the home transactions from 1/1/2016 to 12/31/2016
- 1 unique identify number and 2 home transaction features
- 90276 row (transactions)

Table 3-2 Features Description in train_2016.csv

	Old Name	**Formatted Name**	**Data Type**	**Description**
1	parcelid	id_parcel	Int	Unique identifier for parcels
2	logerror	logerror	Double	$\text{logerror} = \log(\text{Zestimate}) - \log(\text{SalePrice})$
3	transactiondate	transac	String	Transaction Date

3.2 Language and Tool Description

- R language

We use R language for data statistical analysis.
Software version: 3.3.2 (2016-10-31)

- Python

The machine learning part on Spark. We use python for data dimensionality reduction, data transform, training and prediction.

- Databricks

We use databricks to perform data preprocessing, training and prediction.
Databricks runtime version: Spark 2.0 (Auto-updating, Scala 2.10).
Driver Type: Community Optimized 6.0 GB Memory, 0.88 Cores, 1 DBU.

- Library

MLlib, ML, Pipeline

- Evaluation Metrics

MSE and RMSE

4. Data Pre-Processing

4.1 Statistical Analysis

We use R language for the basic data analysis. The distribution of logerror is shown in figure 4-1. This distribution is almost a normal distribution. The Zestimate overestimates the sale price when a logerror is positive and underestimate it with a negative logerror.

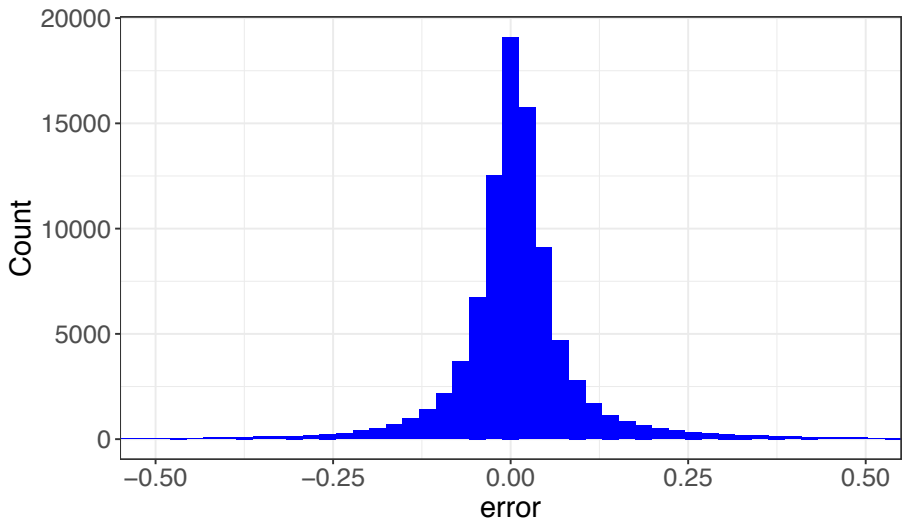

Figure 4-1: Logerror Distribution

From the table 3-1, we know that there are 58 home features in the property raw data. Given that the original feature names are hard to understand, we modified them to some meaningful names (formatted name in table 3-1) in the process. A brief plot regarding the missing percentage with these 58 home features is shown in Figure 4-2. We see that there are many missing values in this dataset.

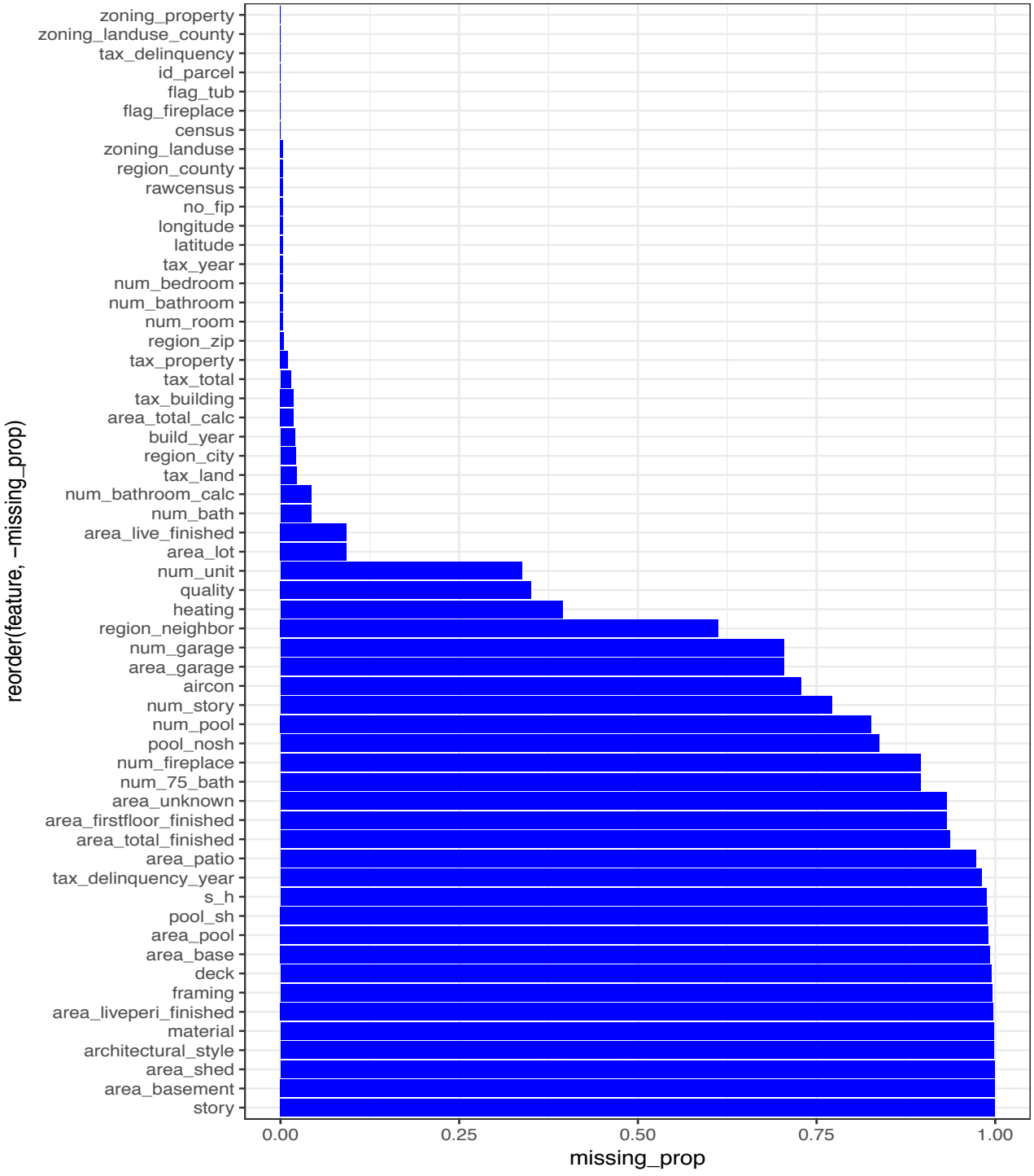

Figure 4-2: Missing Percentage of Each Attributes

4.2 Dimensionality Reduction

From figure 4-2, we can see that the missing percentage in some columns are nearly 0% which are not fit for training and predicting. In order to better prepare data for machine learning, we discard the features with missing percentage greater than 80%. Now we have 37 features in total including the identity number. The new missing percentage for each features is shown in Figure 4-3.

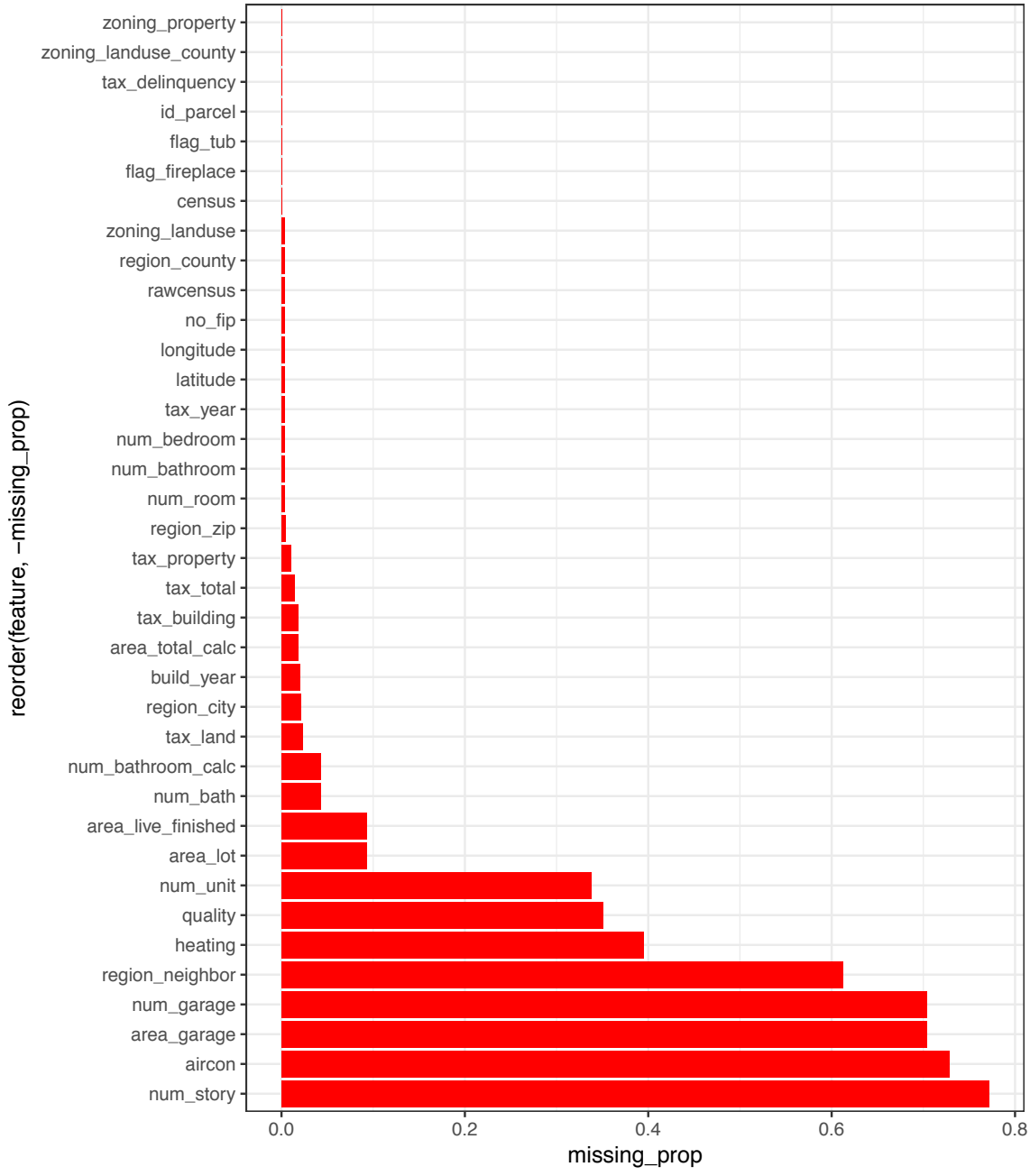

Figure 4-3: Missing Percentage of Each Attributes

4.3 Correlation

To further reduce dimension and to have a first look on the relation between each feature and the logerror, we calculate the correlation of the dataset. Since there are many features, we group them in to several groups based on their common characteristics.

- Number Features

This feature set includes the number of bathrooms, bedrooms, garages, units, etc. The correlation between these features and logerror is shown in Figure 4-4.

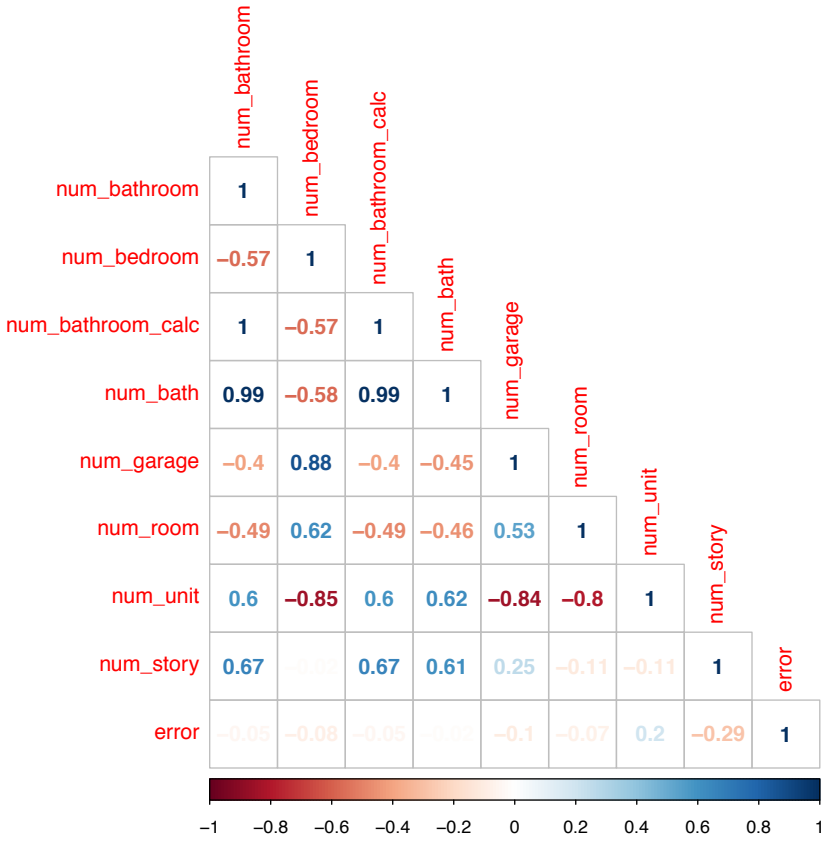

Figure 4-4: Correlation of Number Features

From the plot, we can see that the number features have a very small negative correlation with logerror. And some features are basically have the same correlation, for example num_bathroom, num_bathroom_calc and num_bath. So we treat these three features as a single feature.

- Area Features

This feature contains the number of total area, garage area, etc. The correlation between these features and logerror is shown in Figure 4-5.

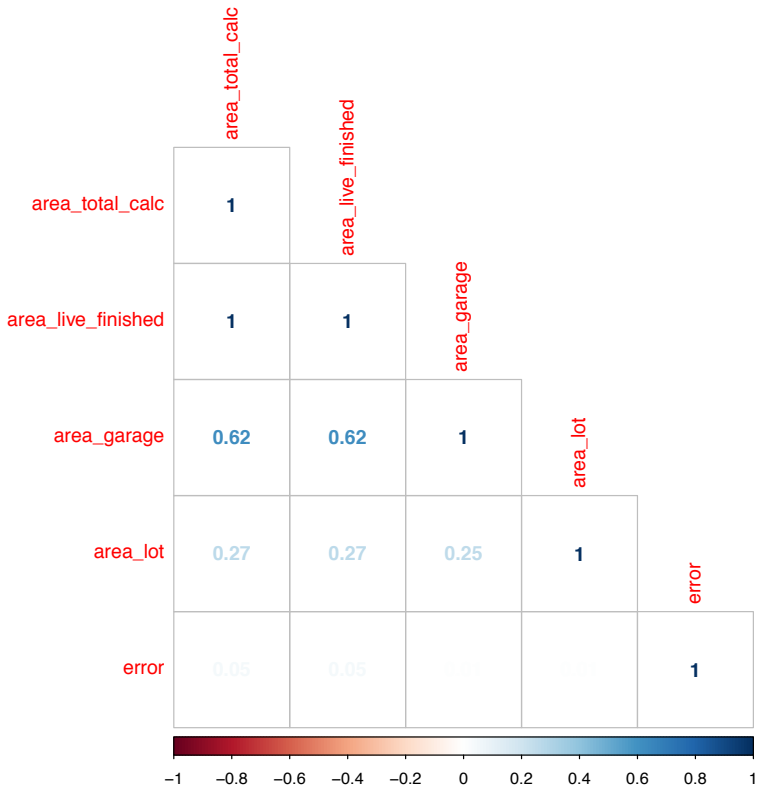

Figure 4-5: Correlation of Area Features

From the plot, we can see that area_total_calc and area_live_finished are also the same feature. And all the features nearly have no correlation with the logerror.

- Tax feature

This feature includes all the tax related features. The correlation between tax features and logerror is shown in Figure 4-6. From the plot, we can see that the correlation between tax and logerror is relatively small and tax_total and tax_land are also the same feature. So we treat these two features as a single feature.

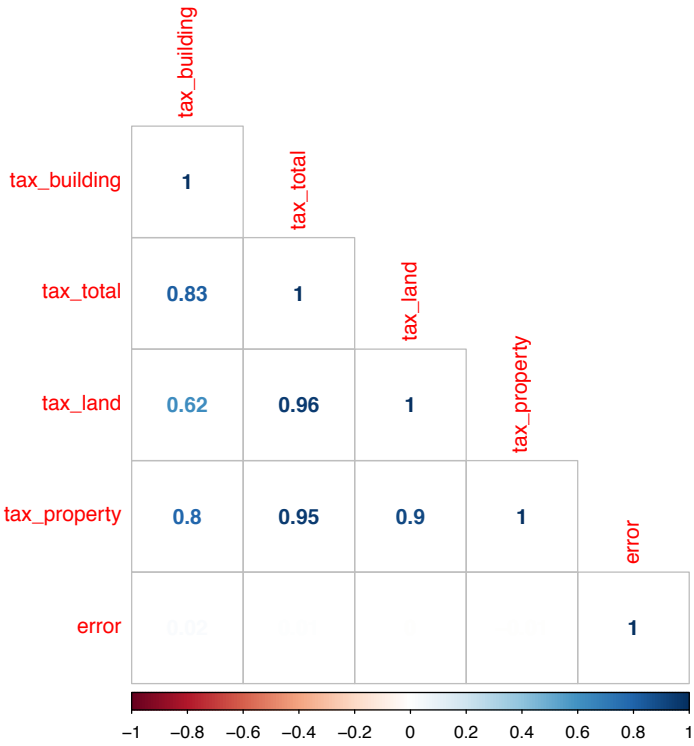

Figure 4-6: Correlation of Tax Features

According the correlation analysis, we remove 4 features again from the properties_2016.csv dataset. But we find several features have a same meaning values, such like census, rawcensus, no_fips, and other some pairs features. Finally, we have 27 features in dataset.

Table 4-1 Features Description in dataset after removing

	Formatted Name	Data Type	Description
1	id_parcel	Int	Unique identifier for parcels
2	aircon	Str	Type of cooling system
3	num_bathroom	Int	Number of bathrooms in home
4	num_bedroom	Int	Number of bedrooms in home
5	quality	Str	Overall assessment of condition of building
6	area_total_calc	Int	Total finished living area of the home
7	no_fip	Str	Federal Information Processing Standard Code
8	num_garage	Int	Total number of garages on the lot
9	area_garage	Int	Total number of square feet of all garages
10	heating	Str	Type of home heating system
11	latitude	Int	Latitude of middle of parcel multiplied by 10E6
12	longitude	Int	Longitude of middle of parcel multiplied by 10E6
13	area_lot	Int	Area of the lot in square feet
14	zoning_landuse_county	Str	County land use code the county

15	zoning_landuse	Str	Type of land use the property is zoned for
16	zoning_property	Str	Description of the allowed land uses
17	region_city	Str	City in which the property is located
18	region_neighbor	Str	Neighborhood of property
19	region_zip	Str	Zip code in which the property is located
20	num_room	Int	Total number of rooms
21	num_unit	Int	Number of units
22	build_year	Str	The Year the principal residence was built
23	num_story	Int	Number of stories
24	tax_building	Int	The assessed value of the built structure
25	tax_year	Int	The year of the property tax assessment
26	tax_land	Int	The assessed value of the land area
27	tax_property	Int	The total property tax assessed for assessment year

In the train_2016.csv, we find there is no need to do any change. So the dataset is as follows.

Table 4-2 Features Description in train_2016.csv

	Old Name	Formatted Name	Data Type	Description
1	parcelid	id_parcel	Int	Unique identifier for parcels
2	logerror	logerror	Double	$\text{logerror} = \log(\text{Zestimate}) - \log(\text{SalePrice})$
3	transactiondate	transac	String	Transaction Date

The input dataset of this project will be two datasets. We will join these two datasets as one after we do data transformation and impute missing values so that the model will be more accurate and obtain more convinced predictions.

4.4 Data Transformation and Missing Value Mechanism

During this project, we find that there are lots of categorical features and missing values. Therefore, we need do data transformation for categorical features and impute the missing values using some method. We have two methods to process the data, and build a model. One is change the data file format into LibSVM and the other is upload the original file into Spark, and read data as dataframe, use PySpark SQL Functions to process the data.

The concise steps:

1) Do a transformation from original data into 'LibSVM' format: We transfer the data format in .csv file into 'LibSVM' format still in .csv file during which we need figure out which features are categorical and transfer them into numeric, in the meanwhile, impute some missing values because data format as 'LibSVM' need to be all numbers.
2) Just upload original data to Spark, and use PySpark functions to read the data, transfer categorical data to numeric data and impute the missing values.

4.4.1 Data Transformation

At this stage, we identify the features with the categorical values and transform them to numeric values. The features we transformed are 'aircon', 'quality', 'heating', 'zoning_landuse_county', 'zoning_landuse', 'zoning_property', 'region_city', 'region_neighbor', 'region_zip', 'build_year', 'no_fip'.

For DataFrame-based model building, we transform two datasets to LibSVM format in local, which can be directly read by regression methods into DataFrame format. And during this step, we transfer the categorical features and impute the missing values.

For RDD-based model building, first we read the original data into dataframe. And then we use SQL functions to transfer the categorical features to numeric number.

4.4.2 Missing Value Mechanism

After removing some features according to their missing percentage, we still have features with missing values in our dataset.

Before transfer categorical features to numeric number, we first fill ‘unknow’ into blanks in categorical columns. Then we transfer them to numeric number together. Finally, we fill the mean of each column into corresponding column.

5. Solutions and Methods

Regression and classification are the most popular methods for training and predicting data. We use random forest regression and gradient-boosted tree regression methods to implement our prediction.

5.1 Theoretical Foundation

5.1.1 Random Forest Regression Introduction

The first regression method we used is random forest regression, which is a popular family of regression and classification methods. In order to minimize result overfitting, random forests combine a number of decisions tress. Decision trees are non-parametric models to perform a sequence of simple tests for each instance, traversing a binary tree data structure until a decision is reached. The model includes an ensemble of decision trees that each tree will outputs a Gaussian

distribution. The model will perform an aggregation over the trees to find a Gaussian distribution closest to the combined distribution for all trees. [3]

Table 5-1 Input of Random Forest Regression [4]

Param Name	Types(s)	Default	Description
labelCol	Double	"label"	Label to predict
featuresCol	Vector	"features"	Feature vector

Table 5-2 Output of Random Forest Regression [4]

Param Name	Types(s)	Default	Description	Notes
predictionCol	Double	"prediction"	Label to predict	
rawPrdictionCol	Vector	"rawPrediction"	Vector of length # classes, with the counts of training instance labels at the tree node which makes the prediction	Classification Only
probabilityCol	Vector	"probability"	Vector of length # classes equal to rawPrediction normalized to a multinomial distribution	Classification Only

The flow chart of Random Forests Regression is as following:

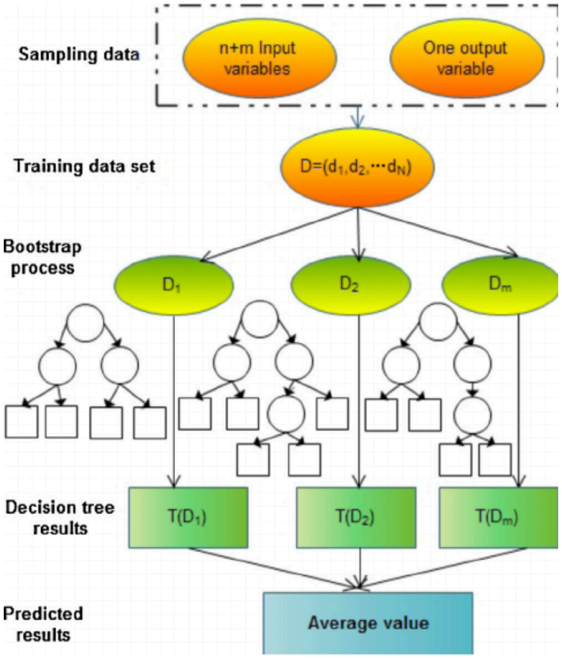

Figure 5-1 Flow Chart of Random Forests

5.1.2 Gradient-Boosted Tree Regression Introduction

The second regression method we used is gradient-boosted tree (GBTs) regression. GBTs are ensembles of decision trees. In order to reduce a loss function, GBTs iteratively train decision trees using boosting. Boosting means that each tree is dependent on prior trees, and learns by fitting the residual of the trees that preceded it. This regression method is a supervised learning

method, and therefore requires a labeled dataset whose column must contain numerical values. [5]

Table 5-3 Input of GBTs [4]

Param Name	Types(s)	Default	Description
labelCol	Double	"label"	Label to predict
featuresCol	Vector	"features"	Feature vector

Table 5-4 Output of GBTs [4]

Param Name	Types(s)	Default	Description	Notes
predictionCol	Double	"prediction"	Predict label	

5.2 Practical Application in Project

5.2.1 Model Selection

First, we join two datasets properties_2016.csv and train_2016.csv to produce a joint dataset which contains 90276 examples and each example has 29 features. We use this joint dataset for model selecting.

- Random forest model selection

1) The model read the dataset into RDD format.
2) Dropping the columns we need to remove according to the analysis in part 4
3) Transforming categorical features into numeric features
4) Imputing missing values with the mean of the values for this feature
5) Creating a labeledPoints and split data into training data and test data with the ratio of 7:3
6) Training and predicting the model with using functions RandomForest.trainRegressor() and predict().
7) Compute test error: compute Mean Squared Error (MSE) on test data

For this model processing, we keep changing two parameters that are numTrees (number of trees) and maxDepth (depth of the tree) to find that the MSE hardly changed when the numTrees equals to 7 and maxDepth equals to 8. See the following table.

Table 5-5 Result of Random Forest Model

Name	Parameters	Value	testMSE
Model 1	numTrees maxDepth	3 4	0.0257550440562
Model 2	numTrees maxDepth	5 6	0.02562358963
Model 3	numTrees maxDepth	7 8	0.02562358963

- Gradient-boosted tree model selection

1) The model read the dataset into RDD format.
2) Dropping the columns we need to remove according to the analysis in part 4
3) Transforming categorical features into numeric features
4) Imputing missing values with the mean of the values for this feature
5) Creating a labeledPoints and split data into training data and test data with the ratio of 7:3
6) Training and predicting the model with using functions GradientBoostedTrees.trainRegressor() and predict().
7) Compute test error: compute Mean Squared Error (MSE) on test data

Table 5-6 Gradient-Boosted Tree Model

Name	Parameters	Value	testMSE
Model 1	numIterations	3	0.029762005897
Model 2	numIterations	5	0. 029754133206
Model 3	numIterations	7	0.0297508622876

Comparing two outputs from random forest regression and gradient-boosted tree regression model from table 5-5 and 5-6, we choose Model 2 of random forest regression for our training and prediction due to its small testMSE. The reason the former regression method have better result is because random forests combine a number of decisions trees to reduce overfitting.

Based on the above result, we have chosen the Random Forest Tree model to do predictions.

5.2.2 RDD Format Dataset Predicting

First we try to use RDD format dataset for training and predicting. During this processing, we use the joint dataset between properties_2016.csv dataset and train_2016.csv for the training part. This joint dataset contains 90276 examples and each example has 29 features.

Then we try to use the whole properties_2016.csv dataset for predicting. The steps are as following:

1) The model read the whole properties_2016.csv dataset and train_2016.csv into RDD format.
2) Dropping the columns we need to remove according to the analysis in part 4 from the properties_2016.csv dataset
3) Transforming categorical features into numeric features in properties_2016.csv dataset
4) Imputing missing values with the mean of the values for this feature in properties_2016.csv dataset
5) Join this clean dataset with train_2016.csv to produce a new dataset for training

6) For this new joint dataset, creating a labeledPoints and perform training
7) Predicting the whole properties_2016.csv dataset with the model using functions predict().
8) Computing test error: compute Mean Squared Error (MSE) on test data

5.2.3 Dataframe Format Dataset Predicting

Then we try to use DataFrame format dataset for training and predicting. Because the databricks cluster crashes due to the large amount of data processing, we split the dataset to several smaller datasets and use one of the datasets to perform training and predicting. Steps are as follows:

On local:
1) Transforming categorical features into numeric features in the split dataset with 200,000 examples from properties_2016.csv
2) Joining dataset between this split dataset and train_2016.csv for the training and predicting. This new joint dataset contains 6000 examples and each example has 29 features.
3) Transforming joined dataset to LibSVM format in local, which can be read by regression methods into DataFrame format.

On databricks:
1) Input of the model is a joint dataset with the LibSVM format we are created in local
2) The model read the dataset into DataFrame format
3) Spliting the dataset into training and testing sets with the ratio of 7:3
4) Training the model with chain indexer and forest in a pipeline using functions Pipeline() and fit()
5) Make prediction with function transform()
6) Computing test error: compute Root Mean Squared Error (RMSE) on test data

6. Results and Analysis

We use R language to do the data statistical analysis and Python in databricks to preprocessing, training, and predicting dataset. After comparison, we choose random forest regression model for machine learning.

For the first method (RDD Format Dataset Predicting), we get a MSE = 0.025299, RMSE = 0.159056, Tree Ensemble Model regressor with 7 trees.

For the second method (Dataframe Format Dataset Predicting), we get a MSE = 0.025862, RMSE = 0.160817, Random Forest Regression Model with 20 trees.

Data cleaning is the hardest part in the project, we need distinguish categorical features, transfer them to numeric features and impute missing values, each of which will produce a bias if we perform different methods. Based on above results, we consider that if we do better in data cleansing stage, and choose more model type to train the data, we will obtain a satisfying output.

7. Conclusion

From the whole process, we know that it is hard and essential for data preprocessing. A good appropriate dataset will improve the performance of the prediction. During the project, we acknowledge that how to select appropriate features for the machine learning, do transformation for categorical features, impute missing values and choose the fittest model to perform predictions.

In the future, we will try different methods to do a data cleansing, and choose other models to build. We think we can improve the accuracy.

8. Contribution of Team Members

Table 8-1: Contribution of Team Members

	Task	Name
1	Material Collection	Ann Luo Lizhong Zhang Jiangyue Xi
2	Dataset Analysis (dimensionality reduction, correlation analysis)	Ann Luo Lizhong Zhang Jiangyue Xi
3	Random Forest Regression Model	Lizhong Zhang Jiangyue Xi
4	Gradient-Boosted Tree Regression Model	Lizhong Zhang Ann Luo
5	Report Editing	Ann Luo Lizhong Zhang Jiangyue Xi

References

[1] W. J. F. W. Hu G., Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing 2012, vol. 443, Springer, Berlin, Heidelberg, 2012.

[2] A. C. Nghiep Nguyen, Predicting Housing Value: A Comparison of Multiple Regression Analysis and Artificial Neural Networks, vol. 22, Journal of Real Estate Research, 2001, pp. 313-336.

[3] Microsoft Azure, "Decision Forest Regression," Microsoft, 2017. [Online]. Available: https://msdn.microsoft.com/en-us/library/azure/dn905862.aspx.

[4] Apache Spark 2.2.0, "Random forest regression," 6 8 2017. [Online]. Available: https://spark.apache.org/docs/latest/ml-classification-regression.html#random-forests.

[5] Microsoft Azure, "Boosted Decision Tree Regression," Microsoft, 2017. [Online]. Available: https://msdn.microsoft.com/en-us/library/azure/dn905801.aspx.