House Price Prediction Using Machine Learning Methods

Written By:

Farhana Afroze
Email: fafroze@buffalo.edu

Abstract: **House prices are important reflection of economy. In this project we are predicting house prices using various machine learning model. We are using linear regression for continuous output and we used logistic regression and neural network for classification to predict the prices of the house. We first estimate the price of a house using various methods and then compare the models to show how each model doing.**

I. INTRODUCTION

House prices are important for the economy. The development of a housing prices prediction model can assist a house seller or a real estate agent to make better informed decision based on house price evaluation. In this project we will predict house price in King County, Seattle, USA. The goal of this project is to create various machine learning model that will estimate the price of the house given some features and then compare the model and choose the best model which will most accurately predict the house prices in Seattle residential area. We are using liner model and classification model to predict the house prices. The classification model we are using that will classify the high-price and low-price houses for the Seattle residential area. And for the implementation we are using scikit-learn, keras and others python libraries.

II.OVERVIEW OF MACHINE LEARNING METHOD AND DATA PREPROCESS FOR HOUSE PRICE PREDICTION

This section provides data preprocess and use various machine learning algorithm to predict house prices. The dataset we used for the project is from Kaggle site. The link is here: https://www.kaggle.com/harlfoxem/housesalesprediction#kc_house_data.csv

This dataset contains 21 columns and 21613 rows. The feature include on the dataset are: id, date, price, bedrooms, bathrooms, sqft_living, sqft_lot, floors, waterfront, view, condition, grade, sqft_above, sqft_basement, yr_built, yr_renovated, zipcode, lat, long, sqft_living15, sqft_lot15.

A. Linear Regression

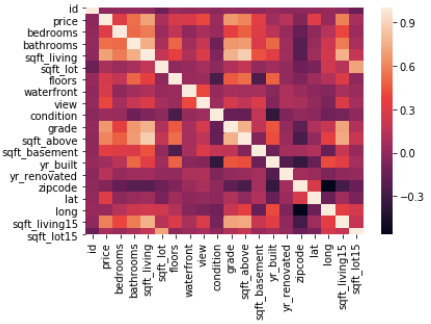
Explore and visualize the data: At first, we would like to explore the data and get statistics for the data. Here is some visualization for the data we got:

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot
count	21613.0	21613.0	21613.0	21613.0	21613.0	21613.0
mean	4580301520.9	540088.1	3.4	2.1	2079.9	15107.0
std	2876565571.3	367127.2	0.9	0.8	918.4	41420.5
min	1000102.0	75000.0	0.0	0.0	290.0	520.0
25%	2123049194.0	321950.0	3.0	1.8	1427.0	5040.0
50%	3904930410.0	450000.0	3.0	2.2	1910.0	7618.0
75%	7308900445.0	645000.0	4.0	2.5	2550.0	10688.0
max	9900000190.0	7700000.0	33.0	8.0	13540.0	1651359.0

	floors	waterfront	view	condition	grade	sqft_above
count	21613.0	21613.0	21613.0	21613.0	21613.0	21613.0
mean	1.5	0.0	0.2	3.4	7.7	1788.4
std	0.5	0.1	0.8	0.7	1.2	828.1
min	1.0	0.0	0.0	1.0	1.0	290.0
25%	1.0	0.0	0.0	3.0	7.0	1190.0
50%	1.5	0.0	0.0	3.0	7.0	1560.0
75%	2.0	0.0	0.0	4.0	8.0	2210.0
max	3.5	1.0	4.0	5.0	13.0	9410.0

	sqft_basement	yr_built	yr_renovated	zipcode	lat	long
count	21613.0	21613.0	21613.0	21613.0	21613.0	21613.0
mean	291.5	1971.0	84.4	98077.9	47.6	-122.2
std	442.6	29.4	401.7	53.5	0.1	0.1
min	0.0	1900.0	0.0	98001.0	47.2	-122.5

Then we can check for any correlations between variables. We will use heatmap to see graphical representation for correlation.



As you can see sqft_living, bathrooms, grade, sqft_above, and sqft_living15 has the highest influences on price.

Data Preprocess: Now we will do some data preprocess for our dataset. At first, we would like to check if there are any null values in the dataset. We don't have any null values in the dataset which is good. Then we dropped two features- id and date as we don't want those as our feature set. Next we split the features as independent and dependent variable. We took price as our dependent variable 'y' which will predict prices based on all other features. After that we did split our dataset into training and testing and we chose .25 for testing and .75 for training. After splitting the dataset into training and testing finally it's time to train our algorithm.

Train Model Using Linear Regression: Next we will train our model using linear regression. We know linear regression model finds the best value for the intercept and slope which results in a line that best fits the data. We found the intercept 5956167.154890 which represents price of as house when features are zero. And the coefficients we found are:

	Coefficient
bedrooms	-41343.4
bathrooms	48154.2
sqft_living	111.3
sqft_lot	0.1
floors	4746.0
waterfront	532525.6
view	53728.8
condition	25947.3
grade	93950.2
sqft_above	71.5
sqft_basement	39.8
yr_built	-2629.2
yr_renovated	12.1
zipcode	-584.1
lat	613046.4
long	-218401.1
sqft_living15	22.0
sqft_lot15	-0.4

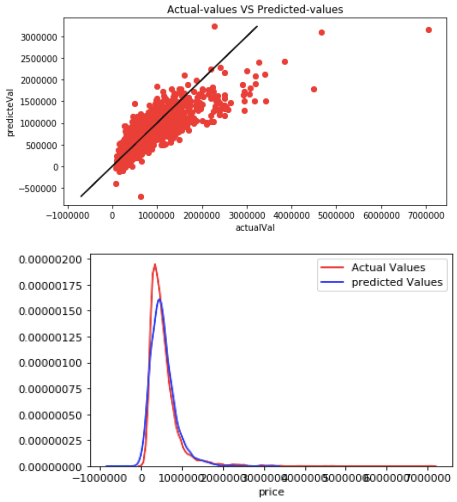
We can see from here sqft_lot and sqft_lot15 has very little effect on price. Next we will predict the price using the model. Next we can check the scores for training and testing model. The scores we got are:
Score for training is: 0.6973094333626098
Score for testing is: 0.7053126793137241. For the testing score we got 0.71 which means that 71% of the variation in house prices data which can be attributed to the independent variables. It tells us how good our model fitted to the data and when it's higher, it's the better. Even though we want higher variability, but this data set is doing pretty good on linear regression model.

Then we predicted on the test data and checked the difference between the predicted and actual value:

	Actual	Predicted
2212	550000.0	507150.2
16859	202000.0	-54053.1
21053	775900.0	670820.2
7757	224950.0	161351.7
12278	312000.0	531015.2
13290	272450.0	191742.7
7992	90000.0	226579.1
8952	522000.0	577203.0
20535	2950000.0	1295209.9
20894	1095000.0	714595.6
14477	153500.0	211447.6
10843	590000.0	806088.8
2178	234000.0	188120.9
5279	368000.0	428550.1
8280	359950.0	477927.3
9853	448175.0	409418.5
4077	875000.0	399629.9
9395	285000.0	262507.3
16760	465000.0	617926.8
19196	1185000.0	1024829.6

As we can see there are some big differences between the actual and predicted values. As we see the model is not doing so good on predictions. Maybe this could be poor features, lack of data, unrelated data or bad assumptions.

Here is some visualization for linear line and we can see how much there is difference between actual and predicted values.



This actually showing us pretty good results. As we can see most of the points are correlated except some are scattered far away from line. We can see some outliers in data plot too.

Model Performance: Next we did evaluate performance using RMSE which full form is Root Mean Squared Error. RMSE for testing set we got 201956.6806622023. Here our RMSE appears large because we have house prices value between 75000 to 7700000. And the mean we found for price was 540088.1

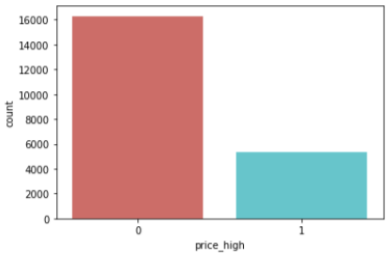
B. Logistic Regression

Now we will use logistic regression to predict our house prices. The data set we are using doesn't have categorical value so we used binarization to get value 0 and 1 on the data set so we can classify the price as

high price and low price. Doing binarization helps to get binary data value if your data set doesn't have binary value for the classification problem you need to tackle with. In King County higher house price, we assumed is over 650,000. And as we don't have categorical value in our dataset which we will need for logistic regression to classify 0 as low price and 1 as high price. So, we made one more column or feature in our dataset called price_high. Now the data set looks like this:

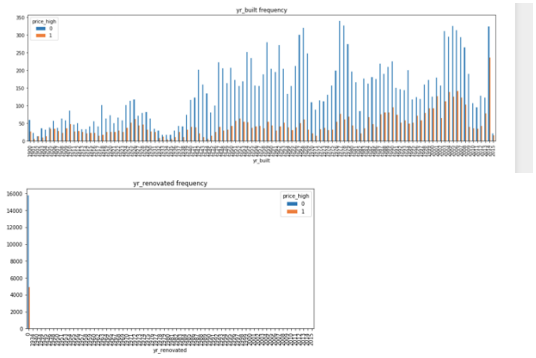
is	sqft_living	sqft_lot	floors	waterfront	view	...	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat	long	sqft_living15	sqft_lot15	price_high
0	1180	5650	1.0	0	0	...	1180	0	1955	0	98178	47.5	-122.3	1340	5650	0
2	2570	7242	2.0	0	0	...	2170	400	1951	1991	98125	47.7	-122.3	1690	7639	0
0	770	10000	1.0	0	0	...	770	0	1933	0	98028	47.7	-122.2	2720	8062	0
0	1960	5000	1.0	0	0	...	1050	910	1965	0	98136	47.5	-122.4	1360	5000	0
0	1680	8080	1.0	0	0	...	1680	0	1987	0	98074	47.6	-122.0	1800	7503	0

We have total 1 is 5324 and total 0 is 16289.



Next we wanted to check if the label price_high has any effect on other features by getting mean values and seeing features frequency so we can get idea how the label has effect on those features.

price_high	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built
0	4602216606.3	392819.8	3.2	1.9	1777.5	13091.3	1.4	0.0	0.1	3.4	7.3	1542.9	234.6	1970.4
1	4513251403.2	990661.9	3.8	2.7	3005.2	21274.0	1.7	0.0	0.6	3.5	8.8	2539.5	465.7	1972.8



As we see here, by seeing these mean values we can see how it's effecting the price_high. You see here waterfront feature doesn't have effect on price_high categorical column. And the frequency dataset showing that 'yr_buit frequency' has good effect on the categorical column but other feature 'yr_renovated frequency' doesn't has effect on price_high categorical feature.

Data Preprocess: As some features doesn't have good effect on those categorical values, we dropped some feature for the independent set. waterfront, condition, lat, long, yr_renovated and date those features that I dropped as I don't find those data as good features. Next we split our dataset as x and y where we chose test size = 0.25 and 0.75 for our training size. And Now it's time to train our algorithm.

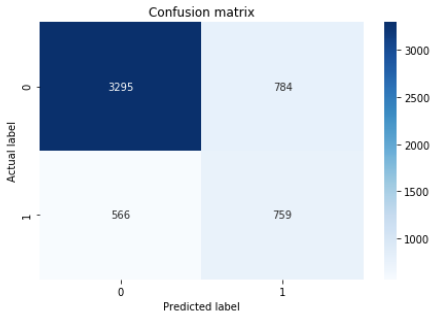
Train Model Using Logistic Regression: After training the model we got the accuracy on test set: 0.7501850481125093 The classification report we got:

	precision	recall	f1-score	support
0	0.85	0.81	0.83	4079
1	0.49	0.57	0.53	1325
accuracy			0.75	5404
macro avg	0.67	0.69	0.68	5404
weighted avg	0.76	0.75	0.76	5404

Here we see we got precision for positive (1) class or high price is 49% and negative (0) or low price is 85%. Precision measures how good our model is when the prediction is positive or negative. We got low for high price or positive class.

We also see we got recall for positive (1) is 57% and negative (0) is 81%. Recall measures how good our model is at correctly predicting positive or negative classes. And we see for positive class or high price it's predicting 57% correctly. And for low price or negative class it's predicting 81% correctly.

The accuracy we got is not bad. Next to evaluate our model for some specific values we used for confusion matrix. We used heatmap to visualize our confusion matrix.



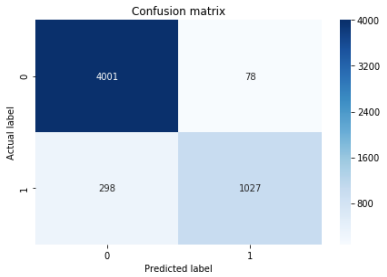
From the confusion matrix we can conclude the following:

True Positive*:* Here we got 759 which we predicted positive result and actual was positive. So that's mean

we have total 759 positive (1) which is high price and we predicted correctly.
***True Negative**:* Here we got 3295 class negative (0) which we predicted correctly as the actual was negative. This belongs to 0 or low price.
***False Negative**:* We predicted negative, but it was actually positive. So, misclassified here. We got total 566 class in negative class (0) but it was supposed to go in positive class (1).
***False Positive**:* We predicted positive, but it was actually negative. So, we again misclassified here. We got total 784 class in positive class (1) but it was supposed to go in negative class (0).

So, we got total 1543 Positive class (1) which is high price. And total 3861 Negative class (0) which is low price. Here we misclassified 1350. We were supposed get 1325 positive class which is high price and 4079 negative class which is low price. So, our model didn't do so good. But we can try to improve our model to eliminate this misclassification.

Improving Model: Now how we can improve the model? As we see in our dataset there are some features values has bigger numbers than other feature values which effects our model score. We tried to use MinMax Scaler to transforms features by scaling each feature to a given scale and we kept the default parameter which is (0,1). So, it's gives features from range between 0 and 1. After doing this we got the new score: 0.930421909696521. So now we got 93% accuracy after doing feature processing which is good. And our confusion matrix now looks like this:



And we can conclude information from our new confusion matrix in the following:
True positive: Here we got total 1027 which is positive class we predicted and actual was positive too. So, it goes to the high price which is 1.
True Negative: Here we got total 4001 which is negative class we predicted and actual was negative too. So, it goes to the low price which is 0.
False Negative: Here we see we got total 298 which belongs to positive class or high price, but we predicted as negative class or low price. So, it goes to low price which is 0.
False Positive: Here we see we got total 78 which belongs to negative class or low price but, we predicted as positive class or high price. And it goes to high price which is 1.

So, we misclassified 376 data. Before improving the model, we misclassified 1350 data. Now it reduces a lot after improving the model. So now we have total positive class with misclassifying is 1105 and total negative class with misclassifying is 4299. If the model would give 100% accuracy, then we would have total positive class is 1325 which belongs to high price and 4079 negative class which belongs to low price. So, we can conclude that our logistic model giving good accuracy after we improved the model. Now we will include classification report picture we got:

	precision	recall	f1-score	support
0	0.93	0.98	0.96	4079
1	0.93	0.78	0.85	1325
accuracy			0.93	5404
macro avg	0.93	0.88	0.90	5404
weighted avg	0.93	0.93	0.93	5404

Here we see we got precision for positive (1) class or high price is 93% and negative (0) or low price is 93%. Precision measures how good our model is when the prediction is positive or negative. We got 93% for both cases for both high price and low price.

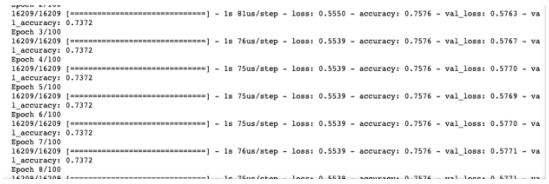
We also see we got recall for positive (1) is 78% and negative (0) is 98%. Recall measures how good our model is at correctly predicting positive or negative classes. And we see for positive class or high price it's predicting 78% correctly. And for low price or negative class it's predicting 98% correctly.

C. Neural Network

We would like to use neural network for our classification model to see how it's doing on our categorical data compared to logistic regression we did. We used keras neural network library for our implementation.

Data Preprocess: At first, we did split train and test data and validation data from test data. We chose 25% for testing data set and 75% for training. For the network architecture we used 2 hidden layers with 35 nodes for each of them. We used the activation as 'relu' function and 'sigmoid' for final output. For the model compiling we used optimizer stochastic gradient descent and for loss we used 'binary_crossentropy' which take 0 and 1 values. We

ran the model 100 times to see how model is performing.

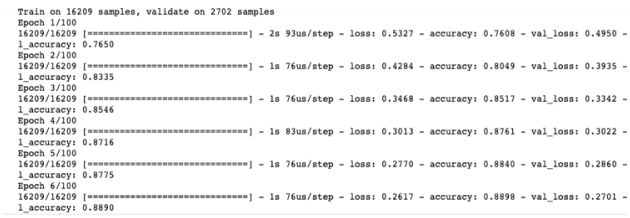


Here we see loss being same and accuracy is not changing either. The accuracy we got for the model is- 0.746484100818634. And the classification report and confusion matrix are given in the following:

```
[[2017    0]
 [ 685    0]]
              precision    recall  f1-score   support

           0       0.75      1.00      0.85      2017
           1       0.00      0.00      0.00       685

    accuracy                           0.75      2702
   macro avg       0.37      0.50      0.43      2702
weighted avg       0.56      0.75      0.64      2702
```

So, here we got True Positive = 0, True Negative = 2017, False Positive = 0 and False Negative = 685. So, for high price we predicted correctly 0 data and for low price we predicted correctly 2017 data. Our precision score giving 0 and recall score giving 0 for positive class or high price which indicates no good. It's indicates not a good model for positive class or high price but for negative class or low price the result is fine. The poor model maybe the cause of poor dataset. So, now we will try to improve our model. We will improve our model by transforming feature using MinMax scaler. This will help to give the data in range in the dataset if we have data in our dataset differs from each other significantly.

Improving Neural network Model: After doing MinMax scaler we want to train our data set again to see what's happening. We again did split our dataset. At first, we did split train and test data and validation data from test data. We chose 25% testing set and 75% for training and for validation data we chose 50% for the test data. Now it's time to run our model to see how it's performing after improving the model. Here is the snapshot about how the model is performing:

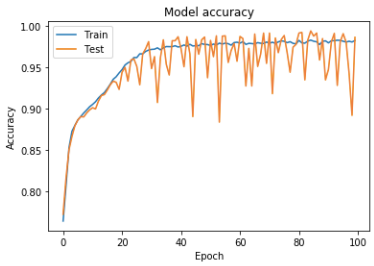


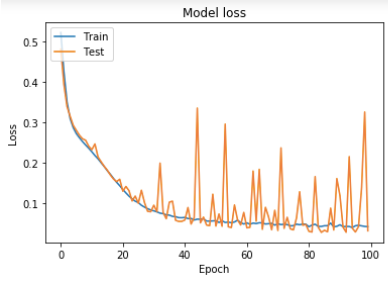
As we see on each iteration loss is decreasing and accuracy is increasing. The accuracy we got here is- 0.988897085 which indicates 99% of the time the model is being able to classify the data correctly. The classification report and confusion matrix ae given in the following:

```
[[1989   27]
 [   3  683]]
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      2016
           1       0.96      1.00      0.98       686

    accuracy                           0.99      2702
   macro avg       0.98      0.99      0.99      2702
weighted avg       0.99      0.99      0.99      2702
```

Here we got True Positive = 683, True Negative = 1989, False Positive = 27 and False Negative = 3. We have total 30 class which we predicted incorrectly. We supposed have total positive class or high price is 686 but we got 683 which is very good. And for the low price or negative class we supposed to have 2016 but we got 1989 which is also very good. The precision we got for the high price 96% and for the recall is 100% whereas precision for the low price we got 100% and recall we got 99% which is really good results. Because we know precision measures how good a model is when prediction is positive or negative and recall measures how good a model is correctly predicting positive or negative classes. So, as we see after improving the model is doing good and classifies data 99% of the time correctly.

Then we wanted to visualize loss and accuracy to see how the improved model did. Here is the snapshot:





As we see the accuracy graph as the iteration going up the model accuracy is increasing but for the validation or test data It's fluctuating but eventually it's increasing. And for the model loss as the time going up loss is getting down and down and for the validation data it's fluctuating but eventually getting less.

III. Comparing the Model

Now it's time to comparing the models we used for our house price prediction.

Models	Accuracy of the models
Linear Regression (linear model)	71% accuracy
Logistic Regression (classification) (after improving the model)	93% accuracy
Neural Network(classification) (after improving the model)	99% accuracy

As we see for classification Neural Network is doing best. We would like to choose Neural network model for our classification problem. And for the linear model the linear regression is not doing bad.

IV. Conclusion

House prices are important for both sellers and buyers. In this project we tried to predict house prices using various machine learning models to see how the features of the houses are effecting to the price which will help real estate to sell the houses and buyers will get idea about the prices of the house by knowing how various feature affecting the price.

Contributors of the project

Name: Ayesha Ismail
Project Part Done: Linear Regression

Name: Farhana Afroze
Project Part Done: Logistic Regression and Neural Network

References

[1].https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8

[2].https://www.datacamp.com/community/tutorials/understanding-logistic-regression-python

[3].https://towardsdatascience.com/machine-learning-fundamentals-via-linear-regression-41a5d11f5220

[4]. https://realpython.com/logistic-regression-python/

[5]. https://keras.io/visualization/

[6]. https://github.com/LeehoLim/housing-price-prediction/blob/master/Housing%20Price.ipynb

[7].https://medium.com/datadriveninvestor/logistic-regression-in-python-423c8d32838b

[8].https://towardsdatascience.com/feature-engineering-for-machine-learning-3a5e293a5114