Housing Market Crash Prediction Using Machine Learning and Historical Data

A Project Report

Presented to

Chris Pollett

Department of Computer Science

San José State University

In Partial Fulfillment

Of the Requirements for the Class

CS 297

By

Parnika De

December, 2019

Abstract

The housing crisis of 2008 was caused by faulty banking policies and using credit derivatives of mortgages for investment purposes. The banking policy that initiated this crisis was subprime mortgages which were given to people with low credit scores. Then the credit derivatives like MBS, CDS and CDOs were built on top of these subprime mortgages, which were the common types of mortgages during that period. Generally, these credit derivatives positively affect the revenue system of a country, but low-quality mortgages resulted in low quality credit derivatives. This project focuses on building various machine learning techniques such as Linear regression, Hidden Markov Model, Long Short-Term Memory using historical data to understand whether the housing market crash in the near future or not.

Index Terms — **Subprime mortgage, credit derivatives, linear regression, hidden markov model, long short-term memory.**

I. Introduction

The 2008 housing crisis devastated the American economy. But before a recession comes there are markers to show trends that all housing recessions follow. The factors that led us to the 2008 recession [2]:

1) Inflated housing prices, that created a housing bubble
2) Relaxed banking policies that led to high borrowing rate
3) Relaxed overall financial regulation i.e., how poorly the regulating bodies worked
4) Policies developed by banks to give more subprime mortgages

The mortgages were made more lucrative when the Federal Reserve Bank reduced the interest rates extremely low for short-term loans, along with easy availability of subprime mortgages. There were more and more people buying houses. As a result, the house prices started going up very quickly as there was a lot of demand for houses and the supply was not that high. Also, people thought that housing market is the pillar for investment as housing market had never crashed before. But everything changed in 2007-2009. During this period everyone tried to sell their house as the interest rates were too high for them. There were also people who couldn't pay their mortgages which led banks to take their houses and sell in the already slow market. The house prices which were the highest a year ago reached the rock-bottom. There were more houses for sale than there were buyers to buy.

In this project I would build machine learning model with historical housing data and other mortgage related data to see whether the trends of 2008 are there or not. These ML models will check how similar the housing market is now to that of the 2008 recession.

II. DELIVERABLE 1: DATA CLEANING AND PREPARATION

The aim of this deliverable is to clean the datasets that are going to used later to design ML models. Data cleansing or data cleaning is the process of identifying and removing (or correcting) inaccurate records from a dataset, table, or database and refers to recognizing unfinished, unreliable, inaccurate or non-relevant parts of the data and then restoring, remodeling, or removing the dirty or crude data. Data cleaning techniques may be performed as batch processing through scripting or interactively with data cleansing tools. After cleaning, a dataset should be uniform with other related datasets in the operation. The discrepancies identified or eliminated may have been basically caused by user entry mistakes, by corruption in storage or transmission, or by various data dictionary descriptions of similar items in various stores. There are various reasons for a dataset to be cleaned:

1. Remove Unwanted observations
2. Fix Structural Errors
3. Filter Unwanted Outliers
4. Handle Missing Data
    a. Missing categorical data
    b. Missing numeric data

The dataset I used consisted housing prices for all of California's counties. There was a need for data cleaning as it had raw data with many missing values. Therefore, the problem we will be dealing with is handling missing numeric data. For missing numeric data, we should flag and fill the values.

1. Flag the observation with an indicator variable of missingness.
2. Then, fill the original missing value with 0 just to meet the technical requirement of no missing values.

By using this technique of flagging and filling, we are essentially allowing the algorithm to estimate the optimal constant for missingness, instead of just filling it in with the mean. To do this I used Pandas a library of Python. In Pandas the missing values are represented as NaN, which translated to “Not a number”. To fill those NaN values the pandas function that is used is pandas.fillna(). Now we can fill these missing values with 0 or we could also choose to fill the missing house prices with price of previous or next value.

```
import pandas as pd

def main():
    # The Housing Data
    df_house = pd.read_excel('House_price.xls', index_col=0).dropna(how='all').fillna(0)
    print(df_house.head())
```

```
                  CA           Alameda  ...  S.F. Bay Area  Inland Empire
Date                                    ...
1990-01-01  194952.0  226148.902299  ...  227365.834348            0.0
1990-02-01  196273.0  219306.000000  ...  234739.457236            0.0
1990-03-01  194856.0  225162.000000  ...  235336.501496            0.0
1990-04-01  196111.0  229333.000000  ...  233178.496107            0.0
1990-05-01  195281.0  232291.000000  ...  235881.361604            0.0
```

Fig 1: Results of cleaning the data set

III. Deliverable 2: Coding Hidden Markov Model

The aim of this deliverable was to understand the how Hidden Markov Model (HMM) works. HMM is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobservable or hidden states. Markov Model is formed by augmenting Markov chains. Let us first give a brief introduction to Markov Chains, a type of a random process.

We begin with a few “states” for the chain, $\{S_1,\ldots,S_n\}$; For instance, if our chain represents average annual weather, to simplify the problem we can have 2 values either Hot or Cold. Suppose the probability of a hot year followed by another hot year is 0.7 and the probability that a cold year is followed by another cold year is 0.6. This matrix is called state transition probabilities,

Also suppose that current research indicates a correlation between the size of tree growth rings and temperature. For simplicity, we only consider three different tree ring sizes, small, medium and large, or S, M and L, respectively. Finally, suppose that based on available evidence, the probabilistic relationship between annual temperature and tree ring sizes. This matrix is also called observation probability matrix.

To start the problem, we would also need to know the initial state distribution matrix which specifies the starting state. Then if were given a set of observations then we will be able to find the hidden states.

A = state transition probabilities

B = observation probability matrix

π = initial state distribution

$O = (O_0, O_1, \ldots, O_{T-1})$ = observation sequence.

To compute the hidden states, we have to solve three questions and these questions are discussed by Prof. Mark Stamp in his HMM paper. The coding for the above problems were done in python. To solve the above problems described in the paper I have first prepared the data. After preparing the data, I have calculated the alpha pass or the forward algorithm by multiplying the A matrix with the probabilities of occurrence. After calculating the alpha-pass, I calculated the beta-pass or the backward algorithm. The backward algorithm is calculated by starting the matrix backwards. The calculation is similar to the alpha pass the only difference is the starting point of the matrix.

After the calculation of these two matrices we would use them to calculate the gamma and di-gamma. Di-gammas are used to find the best fit values for the model. To calculate di gammas we get all the values before gamma from alpha matrix and all the values after gamma from the beta matrix. Then subtract beta matrix from alpha matrix to get the di-gamma. After the di-gammas are found we scale the HMM model and update the original matrix with values that are calculated. Therefore, after each calculation there is an updated A, B and Pi values. After updating we check the probabilities of these values. This is how I have coded the HMM.

IV. DELIVERABLE 3: LEARN AND CODE LONG SHORT-TERM MEMORY

The aim of this deliverable is to learn about long short-term memory and also learn to code it using TensorFlow framework. Long Short-Term Memory is a kind of recurrent neural network. In RNN output from the last step is fed as input in the current step. LSTM was designed by Hochreiter & Schmidhuber. It tackled the problem of long-term dependencies of RNN in which the RNN cannot predict the word stored in the long-term memory but can give more accurate predictions from the recent information. As the gap length increases RNN does not give efficient performance. LSTM can by default retain the information for long period of time. It is used for processing, predicting and classifying on the basis of time series data. The structure of LSTM network typically consists of:

1. Forget Gate
2. Input gate
3. Output gate

To train a network we need to pass the training set information through the above 3 gates. LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn. The forget gate would get the information from the previous layer and then choose to keep the amount of information it wants to keep and the percent of information you want to discard.

The next layer is the input layer which decides what new information we're going to store in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" decides which values we'll update. Next, a tanh layer creates a vector of new candidate values, $\tilde{C}t$, that could be added to the state. In the next step, we'll combine these two to create an update to the state. Finally, we need to decide what we're going to output. This output will be based on our cell state but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state we're going to output. Then, we put the cell state through tanh (to push the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

After learning about LSTM network, the next step was to use the TensorFlow framework to train the LSTM and to do this I have used the Austin weather dataset. The trend of the test data matches quite nicely with the testing data set for the LSTM network. The LSTM coding mainly followed these steps:

Step 1: Define the network. To define the network, we create an instance of the sequential class and add the layers of the network on to this class.

Step 2: We define the data and scale it to fit into a scalar of size(-1, 1)

Step 3: After this we define the window size. This tells how many data points we are going to use to predict the next datapoint in the sequence.

Step 4: After all these we divide the dataset into testing dataset and training dataset. I have used

80% of the data for training purposes and rest 20% for testing purposes.

Step 5: Calculate the LSTM weights through forget gate, input gate and output gate calculations.

Step 6: After defining and setting up training data we go on to calculate the testing data and this is done through epochs. I have used 60 epochs. The number of epochs represents how many times the backward and forward algorithm is being executed for a training and testing data.

Step 7: We plot the actual graph with the testing to show the trend.

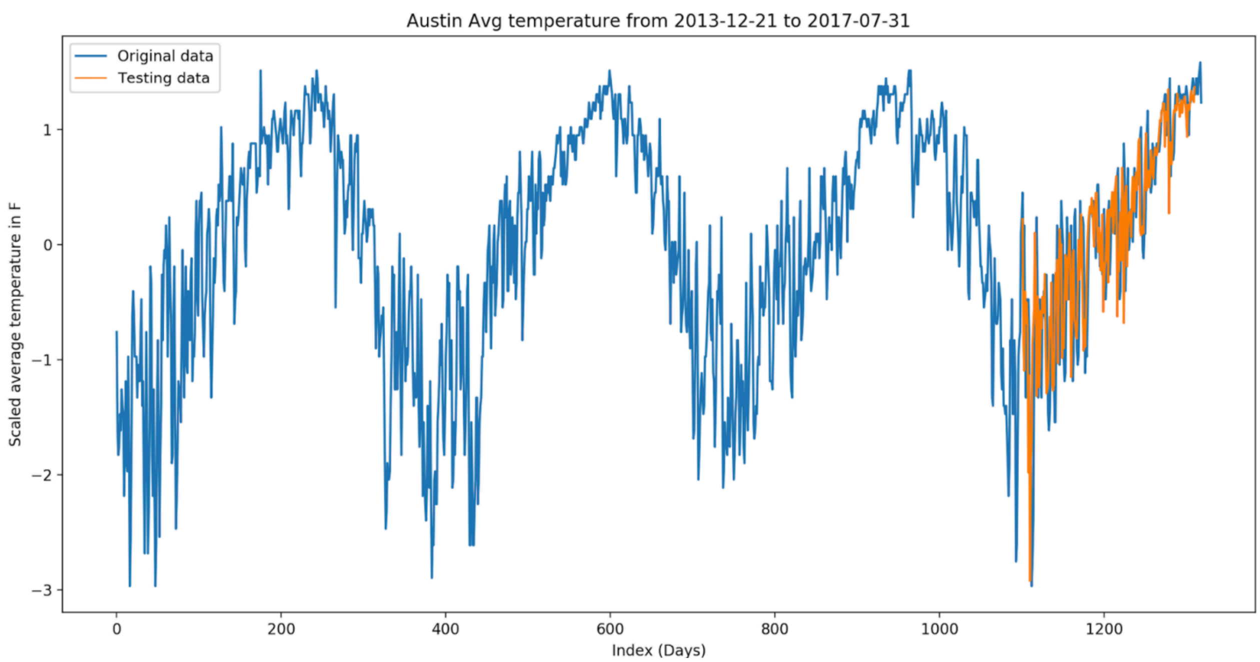

Fig 2: LSTM plot of original data and the testing data

V. Deliverable 4: Apply Linear Regression on Housing Dataset

The aim of this deliverable is to learn to code Linear regression and to find the slope of the house prices for each county and see the trend whether it is increasing or decreasing. A linear regression is a statistical modeling technique which is used to show the linear relationship between a dependent variable and one or more independent variables. To build the linear regression model I have used least squares method. The least squares method is used to find the best fit line. Least squares is a statistical method used to determine the best fit line or the regression line by minimizing the sum of squares created by a mathematical function. The “square” here refers to squaring the distance between a data point and the regression line. The line with the minimum value of the sum of square is the best-fit regression line. After we draw the line we need to minimize the error.

Steps to find the line of best fit for N points:

Step 1: For each (x,y) point calculate x^2 and xy

Step 2: Sum all x, y, x2 and xy, which gives us Σx, Σy, Σx^2 and Σxy (Σ means "sum up")

Step 3: Calculate Slope m:

$m = N\ \Sigma(xy) - \Sigma x\ \Sigma yN\ \Sigma(x^2) - (\Sigma x)^2$ (N is the number of points.)

Step 4: Calculate Intercept b:

$b = \Sigma y - m\ \Sigma xN$

Step 5: Assemble the equation of a line

$y = mx + b$

After finding the best fit line we need to check how good the line is. To find the good ness of the fit R-squared method is used. R-squared does not indicate whether a regression model is adequate. You can have a low R-squared value for a good model, or a high R-squared value for a model that does not fit the data.

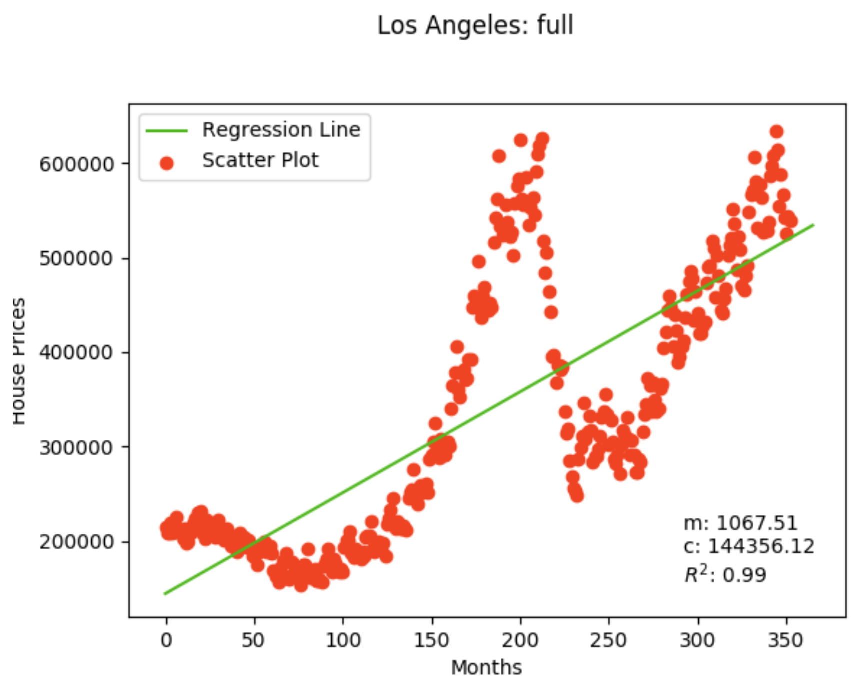

Fig 2: Linear Regression Analysis of median house in LA from Jan 1990 to May 2019

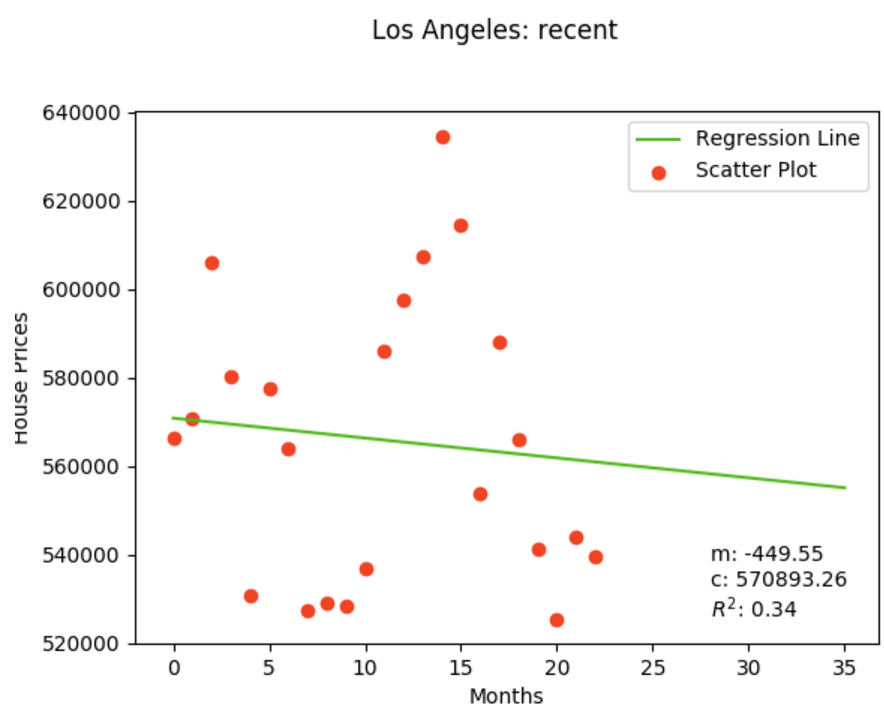

Fig 3: Linear Regression Analysis of median house in LA from June 2016 to May 2019

VI. CONCLUSION

During this semester, as a part of CS297 course, I explored various Machine Learning and statistical techniques. I explored and learned ML techniques like HMM and LSTM and also statistical techniques like Linear Regression. I coded the linear regression and HMM algorithms in python and applied them on datasets to check the working of the algorithms.

To code and build LSTM network I used the TensorFlow framework. To do that I had to learn TensorFlow framework and how it works with python. The ML techniques used as a part of CS 297 can be used in CS 298 project and be extended to be used in housing price datasets. For CS 298 I would explore more data that can be used to check the housing market and use those datasets that are relevant to build all the ML models that has be discussed here.

The model that I would build for CS 298 will show the housing market trends and seeing those trends and comparing that to that of 2008 we will be able to make and a prediction. This all will be done in CS 298 project.

REFERENCES

[1] Y. Demyanyk and I. Hasan, “Financial crises and bank failures: A review of prediction methods”, *Omega*, vol. 38, issue 5, pp.315-324, 2010.

[2] E.J. Schoen, "The 2007–2009 Financial Crisis: An Erosion of Ethics: A Case Study", *J. Bus. Ethics*, vol. 147, pp. 805-830, Dec 2017.

[3] M.R. Hasan and B. Nath, “Stock market forecasting using Hidden Markov Model: A New Approach”, *5th Intl. Conf. on Intel. Sys. Design and Appl., IEEE*, 2006.

[4] F.A. Gers, D. Eck, J. Schmidhuber, "Applying LSTM to time series predictable through Time-Window approaches", *Perspectives in Neural Comput., Springer*, vol. 1, pp. 193-200, 2002.

[5] R.Nyman and P.Ormerod, "Predicting economic recessions using machine learning algorithms", Dec 2016.