Forecasting Local Housing Market Trends Using Recurrent Neural Networks

Krishan Bhakta, Srijan Duggal, Chris Fleisher, Pratyusha Karnati, and Anshul Tusnial
Georgia Institute of Technology
{kbhakta3, sduggal8, cfleisher3, gkarnati3, atusnial3}@gatech.edu

Abstract

This paper explores the use of deep learning models to predict average house prices in a county. Understanding the trends of the local housing market can significantly help a home seller or buyer make more informed decisions based on under/overvalued homes. Related works have used traditional ML models to predict home values based on aspects of the home, but these models do not learn the complex relationships between socioeconomic features to predict local market trends. In this study, we created and analyzed a dataset composed of 235 features at the county, state, and federal level for 1584 counties in 44 states, collected from 2010 to 2017. We then used two recurrent neural network architectures, LSTM and vanilla RNN, to predict average local house prices. We found that applying deep learning models to the housing market produces more accurate results than traditional regression models. Our best model has achieved a $\sim 10\%$ *error rate versus a* $\sim 57\%$ *error rate from a baseline linear regression model. Our implementation can be found here: https://github.com/srijanduggal17/HousingPredictions*

1. Introduction

The last global economic crisis caused an increase in interest from academics in the housing market and its relationship with the economy. While much has been said of the national housing market, there is still insufficient research concerning housing at local levels. Understanding these trends at a per-county level is undoubtedly important for buyers and sellers in determining property value and price. However, traditional forms of appraisal have two major drawbacks: not only do such models often fail to account for macroeconomic trends, but they also tend to generalize poorly to other localities. As such, there is a clear need for generalizable models that can predict average house prices at a county level based on both local and macroeconomic trends. The applications are not limited to real estate brokers; such a model would be useful in such domains as urban planning and local public policy.

First attempts to characterize predicting non-linear patterns seen in market or stock prices have included moving averages, autoregressive models, and correlations [4]. However these methods were not robust to the randomness and chaotic data, and artificial intelligence has become a promising tool to use in the research of forecasting time series data [2]. Initial research in this area has focused on single localities and has focused on analyzing the performance of traditional machine learning models. A Kaggle competition concerning house price prediction for homes in the city of Ames, Iowa, for example, led to a paper highlighting the success of regularized regression, support vector machines, and random forest models as an ensemble [1]. Another paper released in 2019 attempts to predict real estate market values in three counties near Los Angeles, California, using decision-tree based models with success [3]. It is noted that neither of these areas of research focus on macro-level trends; rather, they attempt to predict prices on a per-home basis. Of course, there has been much said about housing market forecasting at a national level; Plakandaras *et al*., for example, attempt to forecast the national U.S. house price index using support vector regression [6]. While we obtain some insights from this research, there is nonetheless a dearth of work in forecasting house price indices at a county level. Prior research has also shown that to effectively build accurate prediction models, an appropriate choice of performance metrics must be evaluated especially when performing model selection [7].

For the range of buyers looking at purchasing real estate, this is by no means an easy task. Many factors influence the housing market price. Having an automated housing price predictor may be useful in understanding current trends and potentially have the ability to suggest purchasing strategies based on user preference of important features. Our research, then, fulfils the need for a generalized model that predicts a county-level house price index, taking into account information at federal, state, and local levels. In this paper, we use recurrent neural networks; in particular, models using a vanilla RNN and an LSTM, for forecasting. It is well known that deep learning models are able to learn more complex feature relationships than tradi-

tional machine learning models; given the volatility of the housing market and the time-series nature of the dataset, we feel recurrent neural network models are an important area of research in this topic. Our hypothesis is that a more complex neural network architecture can outperform current standard methods of forecasting market trends.

The remainder of this paper is organized as follows: Section 2 describes dataset compilation, data-preprocessing, and our feature engineering methods. In section 3, we introduce our model architectures. We present our experiments and their results in section 4, while our conclusions and discussions for future work are conveyed in section 5.

2. Dataset and Pre-Processing

Just as there is a lack of previous work in predicting county-level house price indices, there is also a lack of usable pre-compiled datasets for the topic. As such, we have compiled our own dataset using features obtained from the St. Louis Federal Reserve Bank's FRED API, and a by-county target house price index from Zillow's Home Value Index (ZHVI). API calls to FRED enable us to query years of national, state-level, and county-level economic data, which we aggregate as features into our dataset. The majority of this data is originally sourced from the US Census Bureau and the US Bureau of Labor Statistics, and compiled by the St. Louis Federal Reserve. The size of our dataset was limited by the availability of our features on the FRED API across time and geography.

In particular, our dataset includes 235 features for 1584 counties (and county-equivalents, i.e. parishes and boroughs) in 44 states, out of a total 3143 counties and 50 states in the USA. These features are collected at a monthly level for the time period Jan 2010 - Dec 2017 (inclusive). These features include national data like the U.S. unemployment rate, state-level data like state civilian labor force size, and county-level data like county population.

2.1. Data Collection

To aggregate features into our dataset, we first pull a list of counties and the states to which they belong from the FRED API. Next, we figure out which set of features (and time period) are common to a large enough proportion of the counties to give us an adequate amount of data on which we may train our models. For each of these counties, we then pull relevant county-level, state-level, and national feature data from FRED. We then drop any counties with missing data. For some series, the FRED API has a higher-than-monthly frequency of data; we aggregate these series to a monthly frequency. On the other hand, when we encounter series for which FRED data occurs at a lower-than-monthly frequency, we repeat the series value as necessary for each month. For example, under this system, if a series A has annual data with value x for year 2010, we give each month in 2010 the value x. After this step, all of our data appears in a monthly format.

2.2. Dataset Bias

Geographically, our dataset makes for a fairly representative sample of the United States as a whole, although we note that there exists a slight bias towards higher-population counties, since data for lower-population counties is more frequently missing. Unavoidably, our data has a clear bias towards recent data. We attribute this to the fact that U.S. agencies like the Census Bureau and BLS have only very recently begun to record many macroeconomic series at a county level.

2.3. Data Pre-Processing

Once we have compiled the dataset, we pre-process and standardize the features. Many features collected are better represented as a per capita or other percentage measure; we generate those values. For example, the state-level feature "Average Weekly Earnings of Production Employees: Manufacturing" is better represented as a percentage of "Average Weekly Earnings of All Employees". Moreover, we transform the target ZHVI house prices values by taking their natural log. We do so to facilitate model training. Previous research shows that models perform best under loss functions that satisfy two symmetry conditions - symmetry with respect to prediction error rates, and symmetry with respect to the difference between actual and predicted values; the LRMSE (log-root mean square error) metric on the prices satisfies both these conditions [7]. Finally, we use a typical mean-stdev standardization on each of the non-target features x, replacing x by the quantity

$$\frac{x - \hat{x}}{\sigma_x}, \tag{1}$$

where $\hat{x}$ is the mean across all values of x and σ_x is the standard deviation across all values of x.

2.4. Sliding Window Instance Generation

Once we have compiled and pre-processed our data, we further subdivide our dataset into instances using a "sliding window" technique. Let `num_months` denote the number of months of data we wish to train on for each instance; we set this value as `num_months` $\in \{12, 24\}$. That is, each instance in our full dataset represents a snapshot of a county for a given period in time (12 or 24 months). Furthermore, let `months_to_predict` denote the number of months into the future we wish to forecast. For our datasets, we choose `months_to_predict` $\in \{1, 3, 6\}$. Then, for each possible set of consecutive `num_months` + `months_to_predict` months in our data, we create a single instance in our full dataset. Note that we have 96 total months of data for each county in our dataset. Then, for

example, in trying to forecast 1 month into the future using data from the previous 12 months, we would generate a total of $96 - 13 + 1 = 84$ instances (for each of the 1584 counties). An example of the structure of our data can be seen in Figure 1.

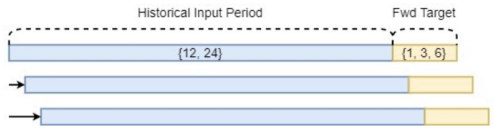

Figure 1. Structure of data fed into our models.

In this manner, we create 6 "full" datasets of these instances, one for each combination of `num_months` and `months_to_predict`. Not only does this process allow us to generate larger sets of data on which we train our models, it also ensures that our models are robust to seasonality and related temporal structures, since the datasets contain instances to forecast each of the 96 months of house prices.

2.5. Feature Selection

Feature engineering is a common tool employed in many machine learning algorithms[2]. Specifically feature selection can be useful in understanding the data or features inputted into a model. This is particularly useful in cases of limited examples to train on. Furthermore, performing feature selection can determine the relative importance of each variable and reduce the chance of overfitting your model with a limited training dataset. Since our goal was to predict a time series target, we attempted to use ensemble methods such as random forest regressor and XGBoost regressor as methods of understanding feature importance for forecasting our target variable[5]. We used a forward feature selection to rank features in terms of how well they contributed to predicting the target. Figure 2 shows the ascending ranked order of the 10 best features.

The top three features found in this initial analysis were "Bachelor's Degree or Higher (5-year estimate)", "Per Capita Personal Income" and "People 25 Years and Over Who Have Completed an Associate's Degree or Higher (5-year estimate)". One limitation of using these ensemble models is that they do not handle time series data well and they have no concept of prior time history. As a result, we did not end up excluding all features except these 10, as the results from a recurrent neural network architecture might be very different. However, this tool can be very useful in understanding the characteristics of a given dataset.

3. Approach

We approached the problem by using the sequential data processing capabilities of recurrent neural networks. We

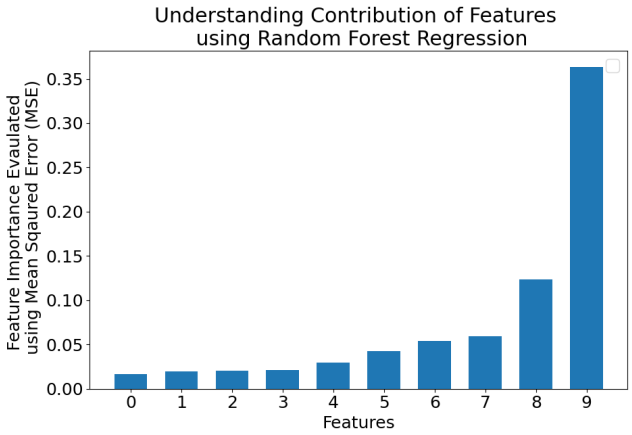

Figure 2. Importance of 10 selected features using a random forest regressor. The most important feature selected was characterized by an educational category which was correlated to the target variable of housing price.

tried two types of recurrent networks: the vanilla RNN and an LSTM. Our RNN architecture consists of a single RNN layer followed by 2 fully-connected layers separated by a ReLU nonlinearity. The fully-connected layer output is directly used to calculate target loss.

It is known that RNNs can have difficulties in learning long-term dependencies due to the vanishing and exploding gradients that occur after propagating gradients down many layers of the network. To address this potential limitation, we use a Long Short-Term Memory (LSTM) model comprised of 2 recurrent layers and a fully-connected layer at the end; each of these layers has an independent set of learned parameters. The output of the model is the output of the fully connected layer. We hoped this model's ability to learn long-term relationships by using memory units would lead to a better prediction since we had eight years of feature data for each instance. Both models were implemented using PyTorch layers.

Finally, to construct a baseline to which we could compare our results, we performed closed-form linear regression on the prices. The inputs to the model were the `num_months` months of prices, and the outputs were the predicted `months_to_predict` prices.

For both of the neural network models, we used a many-to-one, many-to-three, and many-to-six architecture, where we had many months of inputs (12 or 24) and made one, three, or six-months predictions at the end, using a single layer fully connected network. We thought this would be successful because the model architectures encoded the sequential structure of the problem. Additionally, we thought a many-to-few architecture would be useful so that we could predict one, three, or six months into the future.

We anticipated lack of data to be a problem, as well as the

effect of volatility and randomness that is present in many financial and labor markets. The first thing we tried did not work well at all. It was a vanilla RNN with a many-to-one architecture, with a dataset where each instance included features from January to November, and the ground truth for December was predicted. We think this did not work well because there were only about 14000 instances (for training and evaluation), and because information about previous home prices was not being provided. We then decided to create all possible 12 and 24 month sliding windows in our timeframe, rather than only using January to December. This increases our dataset size to around 130000 instances (although each of the 6 datasets differ slightly in size). Additionally, we provided previous months' home prices with the inputs. We anticipated this might lead to overfitting, but we wanted to see the results. The input to each network was a `batch_size` x `num_months` x 235 features tensor. The output of each network was a `batch_size` x `months_to_predict` tensor.

4. Experiments and Results

To test our models' performance, we trained both of them on each of our 6 generated datasets. We carefully considered the evaluation metrics and hyperparameters used in model training.

4.1. Evaluation Metrics

We use log-root mean squared error (LRMSE) as the loss function for our models, and mean absolute percentage error (MAPE) as a standardized evaluation metric to measure model success. As discussed in Section 2.3, we use LRMSE due to the two symmetry conditions it satisfies [7]. Formally, for a dataset of N instances, if p_i are the actual prices and $\hat{p_i}$ are the corresponding predicted prices, then the LRMSE of the prices is given by

$$\text{LRMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\Big[\ln(\hat{p_i}) - \ln(p_i)\Big]^2}, \qquad (2)$$

and the MAPE of the prices is given by

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\frac{|p_i - \hat{p_i}|}{p_i}. \qquad (3)$$

We can see that two instances with the same prediction error ratio $\hat{p_i}/p_i$ will contribute the same amount to the total LRMSE, since $\ln(\hat{p_i}) - \ln(p_i) = \ln(\hat{p_i}/p_i)$; this is the first symmetry condition. Moreover, it is clear that swapping p_i and $\hat{p_i}$ does not change LRMSE; this is the second symmetry condition.

In both our loss function and our evaluation metric, a lower value corresponds to a better-performing model. While LRMSE values are unbounded above (and have a minimum of zero), MAPE values all fall within the range $[0, 100]$. Thus while LRMSE's properties make it well suited as a loss function, we use MAPE as our evalutation metric for its interpretability.

4.2. Hyperparameter Tuning

In both of our models, we consider the following hyperparameters: optimizer, batch size, number of epochs, number of hidden dimensions in the recurrent layers, initial learning rate, exponential learning rate decay. We fix the optimizer as ADAM, the batch size as 512, and the learning rate decay parameter as 0.95. We used ADAM due to its proven success in recurrent neural networks problems. For each dataset, we try all combinations of the following sets: number of hidden dimensions {10, 50, 100}, number of epochs {10, 50, 100}, and initial learning rate {.01, .001}. We chose to only vary these three hyperparameters because we thought they would have the greatest effect on the performance. We ran these experiments on each dataset and model combination. An example of the result of varying the hyperparameters for a given number of epochs is shown in the heatmap in Figure 3.

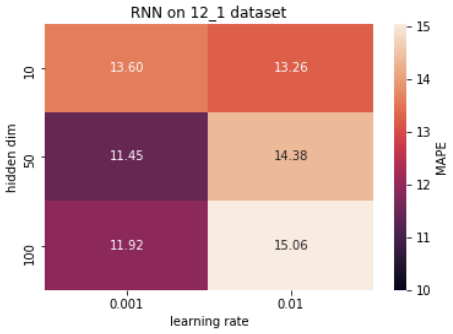

Figure 3. Error for different hyperparameter combinations for an example dataset (using 12 months of data to predict 1 month ahead.

4.3. Results

For each dataset and model combination, we choose the hyperparameter combination with the best MAPE. An example of the learning curves for one dataset and model combination is shown in Figure 4.

As the learning curves show, the validation loss at 10 epochs is pretty high, decreases significantly after 50 epochs, and marginally more after 100 epochs. The curves for all the models have the same general trend in that the validation error tends to converge by the 50 epoch mark. We show the results of the best hyperparameter combinations for each of the 6 datasets in Figure 5 and in Table 1.

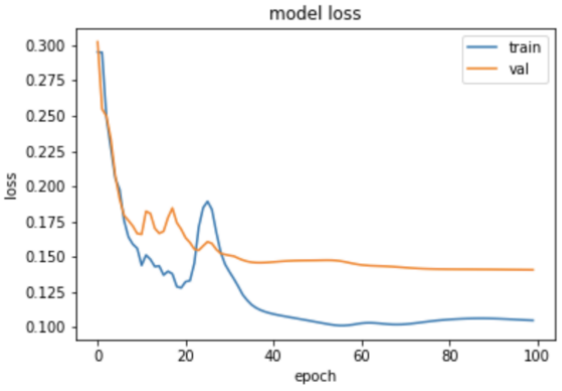

Figure 4. Change in LRMSE by epoch

As a baseline to compare our model to, we use closed-form linear regression on the input features. Overall, we found the initial learning rate of .001 with 50 hidden dimensions for 50 epochs to be good hyperparameter values.

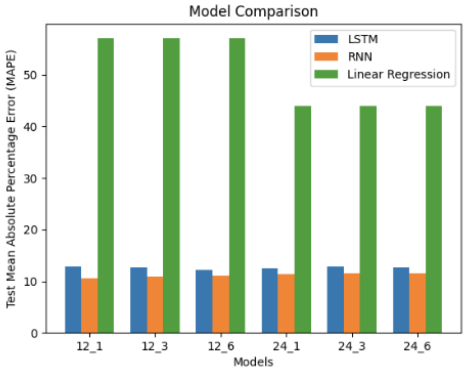

Figure 5. Bar graph comparing the best models for each dataset and the linear regression.

	12_1	12_3	12_6	24_1	24_3	24_6
RNN	10.61	10.97	11.06	11.30	11.59	11.48
LSTM	12.82	12.69	12.13	12.52	12.77	12.71

Table 1. Test MAPE percentage for best hyperparameter combinations for each model (rows) and dataset (columns)

Our optimal model, as shown by Table 1, was the RNN model on the 12_1 dataset with the following hyperparameters:

- number of epochs: 50
- number of hidden dims: 50
- learning rate: 0.001

The loss plot for this optimal model can be seen in Figure 4 and the predicted vs ground truth house prices for a subset of instances can be seen in Figure 6.

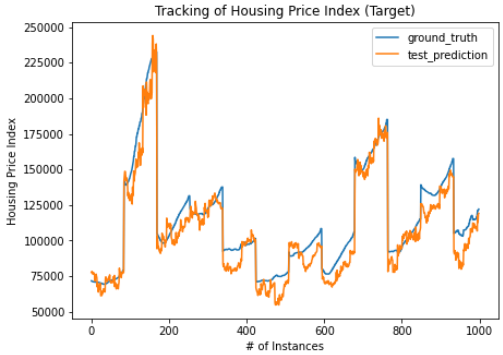

Figure 6. House prices for selected instances in 12_1 dataset (model {RNN, 50,50,.001})

5. Discussion and Future Work

Our project investigated using multiple neural network architectures (i.e. LSTM and RNN) in forecasting housing market trends. We found that our best model (RNN) was able to predict these trends with an $\sim 10\%$ mean absolute percentage error. Furthermore, all of our deep learning models were successful in significantly improving performance from a simple baseline linear regression (57% mean absolute percentage error in predictions). Additionally, all models performed similarly or better on the test dataset than the validation dataset. This shows that this approach generalized well. Our approach showed that finding an optimal model complexity could improve model performance for our given dataset.

Room for improvement in our approach includes better evaluation (we used a single test set rather than a K-fold validation scheme) and feature selection utilizing models that have time history properties. Performing a wider and finer hyperparameter search may yield in a more optimal model selection. More broadly, if successful, these techniques may be able to generalize to forecasting time series data. Future work in this area includes performing feature engineering, model comparisons, and creating an open data source that will allow other researchers to forecast housing market trends with greater accuracy and precision.

6. Work Division

Table 2 on the following page provides information about the delegation of work among team members.

Student Name	Contributed Aspects	Details
Krishan Bhakta	Research, Implementation, and Analysis	Researched prior work done in this field and implemented feature selection scripts. Assisted with training models and analyzed results.
Srijan Duggal	Data Creation, Implementation, Analysis	Researched the FRED API functions and set up how to scrape all the data. Assisted with training models and analyzed the results.
Chris Fleisher	Data Creation and Implementation	Cleaned, combined, and standardized county, state, and country level features into dataset splits by time window. Implemented vanilla RNN and assisted in training models.
Pratyusha Karnati	Data Creation and Implementation	Scraped the datasets for the county, state, and country level data before combination. Implemented, trained, and analyzed the LSTM model.
Anshul Tusnial	Research, Data Creation, Implementation	Researched symmetric loss functions, scraped all data from FRED API, implemented the main model training loop, and did baseline linear regression.

Table 2. Contributions of team members. All members discussed the approaches together, created figures, and wrote the paper

References

[1] Chenchen Fan, Zechen Cui, and Xiaofeng Zhong. House prices prediction with machine learning algorithms, 2018. 1

[2] Bruno Miranda Henriique, Vinicius Amorim Sobreiro, and Herbert Kimura. Literature review: Machine learning techniques applied to financial market prediction. *Elsevier*, 124:226–251, 2019. 1, 3

[3] Yitong Huang. Predicting home value in california, united states via machine learning modeling, 2019. 1

[4] Manish Kumar. Forecasting stock index returns using arima-svm, arima-ann, and arima-random forest hybrid models. *International Journal of Banking, Accounting, and Finance*, 5(3), 2014. 1

[5] Sendhil Mullainathan and Jann Spiess. Machine learning: An applied econometric approach. *Journal of Economic Perspectives*, 31(2), Spring 2017. 3

[6] Vasilios Plakandaras, Rangan Gupta, Periklis Gogas, and Theophilos Papadimitriou. Forecasting the u.s. real house price index. *Economic Modeling*, 45:259–267, February 2015. 1

[7] Miriam Steurer and Robert J. Hill. Metrics for measuring the performance of machine learning prediction models: An application to the housing market, February 2020. 1, 2, 4