# Machine Learning Methods for Property Value Forecasting

Daniel Atallah, Nathan Kitts, Xincheng (Steven) Song

## 1. INTRODUCTION

Property is one of the most sought out avenues for investment and is naturally a part of our daily lives. With the demand necessitated by real estate, it is natural that speculations of home value would be made. Everyone has "their own voodoo method of generating pricing estimates" (Napier, 2018, 22:40) which are likely founded in anecdotes and in some cases used to sway a less informed home buyer. Furthermore, home value swings directly impact rental values and renters. This prompted our objective of determining several viable methods for forecasting home values.

Our initial goal was to estimate "fair" market price for individual listings. We believed this could eventually manifest itself as a browser extension that one could use to scan a listing, detect notable features like square footage and number of bedrooms while accounting for neighborhood attributes. We quickly realized that while listing data is technically public, there are restrictions for accessing it. Similar obstacles exist for other granular real estate sources. Unless one is a licensed real estate agent with access to multiple listing service (MLS) databases, the average data scientist would need to build a web scraper which is subject to being blocked by the host website (e.g. Zillow).

In turn, we opted to expand the purview of our analysis – since our hope was to develop multiple proofs of concept for forecasting home values we could instead look at average home values in an overarching community. While cities may have pockets of cheaper housing, even those developments experience the pricing swings of their broader communities. Zillow's readily available and proprietary Zillow Home Value Index (ZHVI) served as a proxy for market price in our analysis.

## 2. DATA

The ZHVI data is available down to the ZIP code grain, however we chose the metropolitan area grain for compatibility with a larger variety of sources and to encapsulate the overarching conditions of a region. This data is smoothed, seasonally adjusted, and reported monthly for most metros since 2000. Per Zillow's research data documentation, ZHVI "reflects the typical value for homes in the 35th to 65th percentile range." ZHVI served as our target variable. In order to maintain a manageable workload, we restricted our data collection by randomly selecting five of the fifteen largest US metros. We selected the Chicago, Dallas, Los Angeles, New York, and Seattle metro areas.

We leveraged data from the Census Bureau for multiple features. Due to inefficiencies in the Census Bureau's API, most data were collected manually or via URL requests. Total population migration was collected at a yearly grain per metro between 2010 and 2022. This data was linearly interpolated to augment it to monthly grain. New home permits were collected as monthly counts by city between 2010 and 2023 – this served as a proxy for metro growth.

Stock market data was captured to explore the relationships between property value and market fluctuations. Stock data was gathered in monthly intervals at the national level from yahoo finance. The stocks of Four Brokerages were explored: Anywhere Real Estate (HOUS), Coldwell Banker Richard Ellis (CBRE), Redfin (RDFN), and RE/MAX (RMAX).

Airborne travel and trade may allude to the connectedness of a city. A monthly count of distinct domestic and international, inbound and outbound, and passenger and cargo airborne routes were collected from the Bureau of Transportation Statistics' website.

Daily crime history, readily available for each city, was manually extracted and aggregated to monthly counts of property and violent crimes. These metrics evaluated safety, a highly sought out variable for home buyers.

40th Percentile monthly rent estimates were collected from the Department of Housing and Urban Development's website to understand competitive rents. These estimates are calculated yearly so we forward-filled them into monthly estimators. Competitive rents correlate with home values.

The Bureau of Labor Statistics calculates yearly median and mean salaries by job family and title. A holistic mean and median were calculated for each year and forward-filled to illustrate standard of living.

Weather and climate were represented using manually collected monthly extracts from the National Oceanic and Atmospheric Administration. These extracts included attributes for humidity, temperature, and precipitation.

Regional politics were illustrated by the number of congress representatives by political affiliation (e.g. Democrat, Republican, Independent, Other). sponsored and cosponsored legislation served as a monthly measure of political activity. This data was collected using Congress' API.

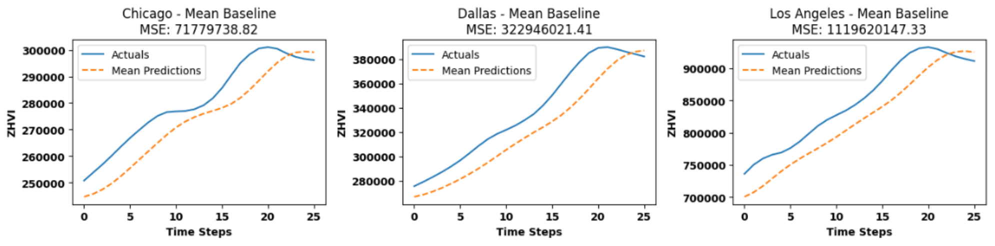
All records in our final merged DataFrame were unique to the city, year, and month. An 80% train-test split was used across models, with the data sorted chronologically.

## 3. METHODS

Multiple methods were leveraged to forecast ZHVI. The goal of this project was to compare the nuances between each method.

### *3.1 Evaluation*

In order to have a concrete basis for comparing models it was critical that we start with a baseline model and a shared evaluation metric. A mean baseline model was created using a 6 month rolling window for each city. Mean squared error (MSE) was used as our evaluation metric due to its interpretability. MSE was calculated for each city's baseline model. Our goal was to minimize MSE for each model.



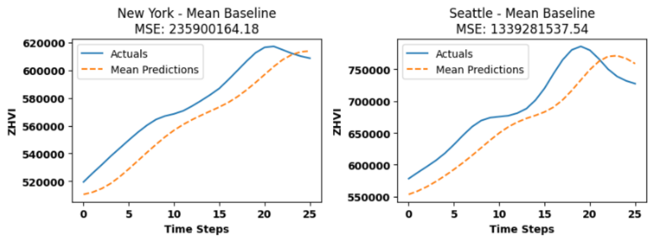


***Figure 3.1.1*** *Mean baseline values compared to actuals for each city.*

## *3.2 LSTM*

The standard approach for making time series forecasts is to parameterize some variation of a Box-Jenkins Model. This approach is quite robust if the underlying assumption that the time series is stationary holds. This assumption is likely violated with property values due to inflation. Neural networks are well-suited for nonstationary financial forecasting since they have an aptitude for mapping nonlinear relationships (Kaastra & Boyd, 1996). The Long Short-Term Memory (LSTM) recurrent neural network (RNN) stood out as a top choice due to its exceptional handling of sequential data and avoidance of the vanishing gradient problem.

In addition to the mean baseline models, baseline LSTM models were fitted using all 85 features. The arbitrary baseline models were double layer LSTMs with an input dimension equal to the number of features, a hidden dimension of 50 nodes, and a fully connected dense layer with an output dimension of one. An Adam optimizer with the default 0.001 learning rate and an MSE loss function were used for all LSTM models. Figure 3.2.1 suggests that the baseline LSTM predictions were far less accurate than the mean baseline predictions portrayed in figure 3.1.1.

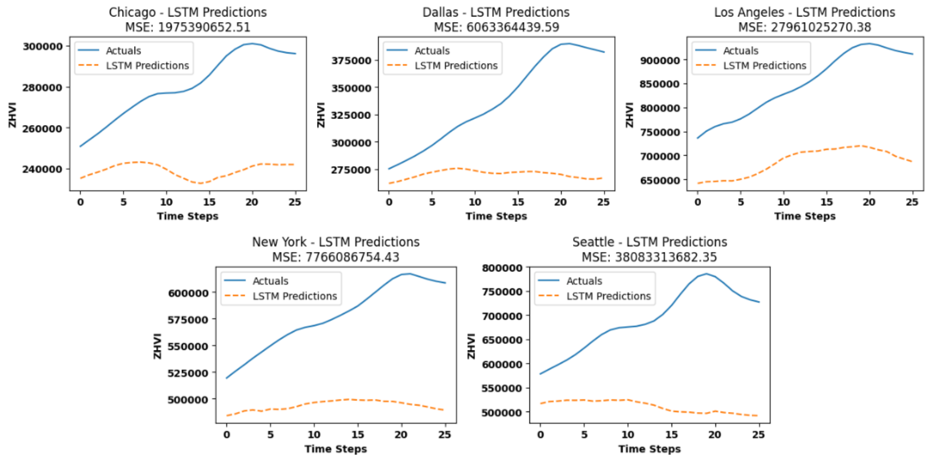


***Figure 3.2.1*** *LSTM predictions compared to actuals for each city.*

### *3.2.1 Window Size*

The baseline LSTM did not utilize a window of data for its predictions. Windows are a notable feature of time series analyses where history informs the future by encapsulating seasonal or contextual information that improves model performance. In some cases, however, too large of a window may cause noise. Six different window sizes were tested. In the case of Chicago, the 3-month rolling window yielded the smallest MSE. A 3-month rolling window was used in all consequent LSTM models.

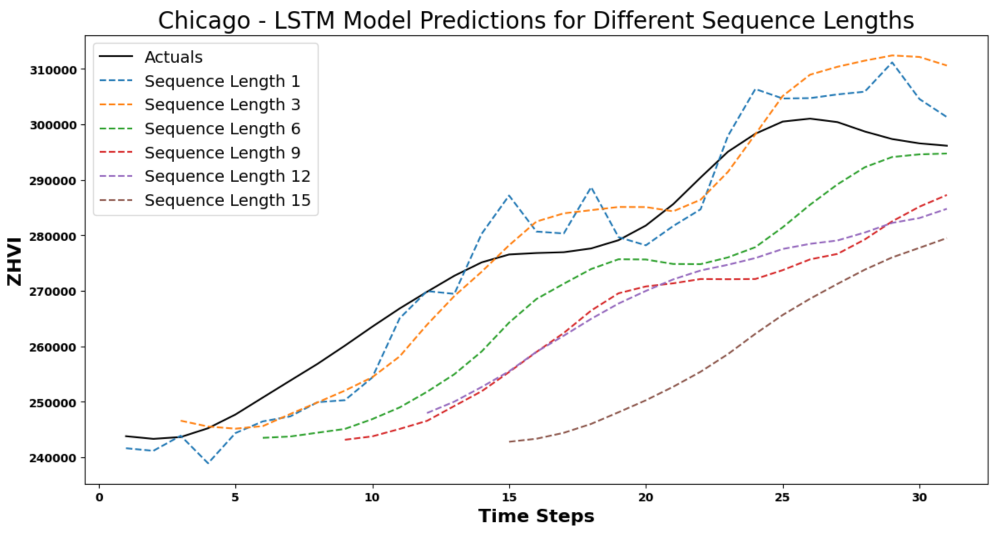


***Figure 3.2.1.1*** *Chicago baseline LSTM performance against actuals using varying temporal windows.*

### *3.2.2 Feature Importance*

All features were scaled using SciKit-Learn's MinMaxScaler. The use of redundant features has the potential to cause overparameterization in a hyperparameterized model. In our case, the breadth of explanatory variables induced noise. A feature importance analysis was conducted to eliminate redundant and noisy features.

Permutation Importance Analysis was used to assess the significance of individual features in our LSTM models by measuring the change in performance when features are randomly shuffled. If a feature is important, disrupting its relationship with the target variable will lead to a significant drop in model performance. Conversely, if the model's performance remains largely unaffected, the feature is likely less important. Our permutation importance analysis yielded the following feature importances for Chicago metro ZHVIs.

```
Top 5 contributing features based on permutation importance:
RDFN_Open: 0.005086022235228682
passenger_domestic_outbound: 0.003394622844354489
CBRE_Close: 0.002675440111349056
ZHVI: 0.0021587451599786893
a_mean: 0.0020992839813892606
```

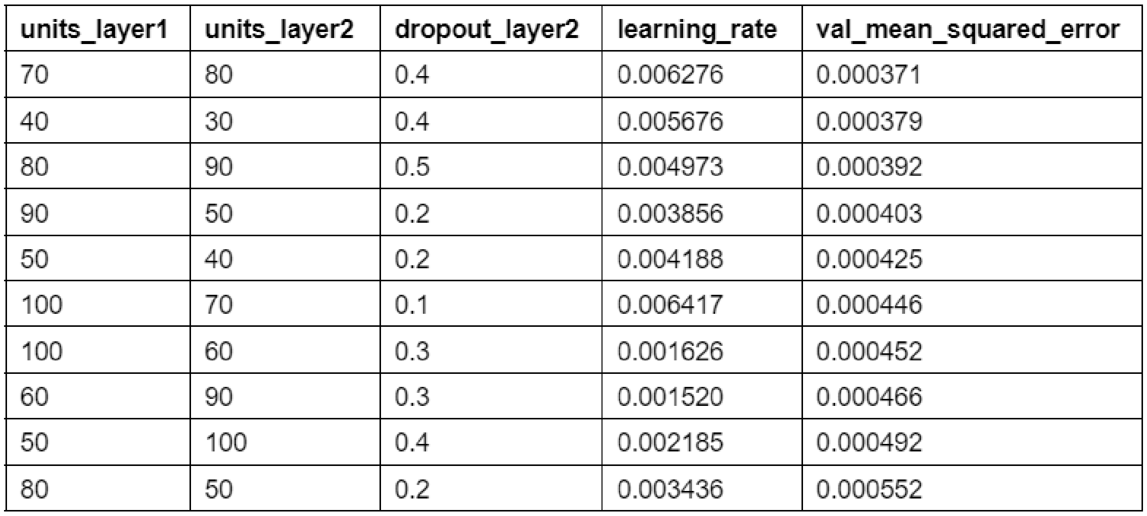
Permutation importance analysis was conducted for each city resulting in a unique set of ideal features for each LSTM.

### *3.2.3 Model Tuning*

We started training with our initial baseline model architecture (described in section 3.2). The first modification was a more informed feature selection – the five most significant features (per the permutation importances) were selected as inputs for each city. The input dimension accommodated the smaller input space. Model performance improved significantly based on feature selection alone.

We observed signs of overfitting during model tuning. Specifically, while training loss decreased, validation loss followed initial decreases before starting to increase. A validation loss increase is indicative of overfitting. We added a dropout layer after the second LSTM layer to mitigate this.

To satisfy our timeline and project scope, we opted to employ random search (over grid search) to isolate the most performant hyperparameters. We searched over the number of units in both LSTM layers, the dropout rate, and the learning rate to maximize MSE. Table 3.2.3.1 illustrates our random search results for Chicago where the most performant model (the one with the lowest MSE) has hidden dimensions of 70 and 80 for each LSTM layer respectively, a dropout rate of 0.4, and a learning rate of 0.006276.



| units_layer1 | units_layer2 | dropout_layer2 | learning_rate | val_mean_squared_error |
|---|---|---|---|---|
| 70 | 80 | 0.4 | 0.006276 | 0.000371 |
| 40 | 30 | 0.4 | 0.005676 | 0.000379 |
| 80 | 90 | 0.5 | 0.004973 | 0.000392 |
| 90 | 50 | 0.2 | 0.003856 | 0.000403 |
| 50 | 40 | 0.2 | 0.004188 | 0.000425 |
| 100 | 70 | 0.1 | 0.006417 | 0.000446 |
| 100 | 60 | 0.3 | 0.001626 | 0.000452 |
| 60 | 90 | 0.3 | 0.001520 | 0.000466 |
| 50 | 100 | 0.4 | 0.002185 | 0.000492 |
| 80 | 50 | 0.2 | 0.003436 | 0.000552 |

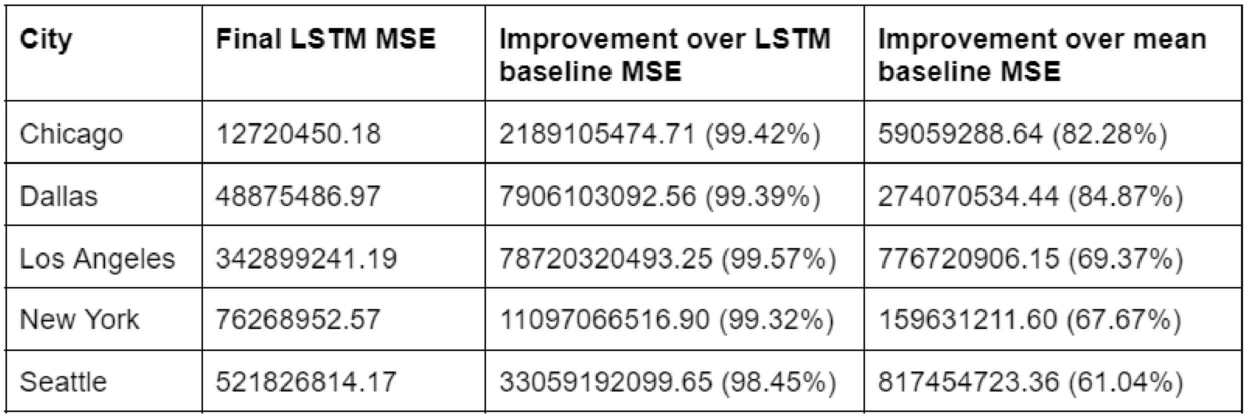
***Table 3.2.3.1*** *Random search parameters and loss amounts for Chicago LSTM.*

### *3.2.4 Results*

The most performant LSTM architecture was selected for each city based on the results of the random search. Despite our effort to prevent overfitting using a dropout layer, overfitting was still realized with each increase in units or epochs. Two additional guardrails for overfitting were implemented:

1. ReduceLROnPlateau - Reduces the learning rate when the validation loss plateaus.
2. EarlyStopping - Stops training when the validation loss either plateaus or starts to increase.

The models were retrained with these additional functions implemented. Table 3.2.4.1 illustrates the MSE improvements of the final LSTMs when compared to the LSTM and mean baseline models for each city.



| City | Final LSTM MSE | Improvement over LSTM baseline MSE | Improvement over mean baseline MSE |
|---|---|---|---|
| Chicago | 12720450.18 | 2189105474.71 (99.42%) | 59059288.64 (82.28%) |
| Dallas | 48875486.97 | 7906103092.56 (99.39%) | 274070534.44 (84.87%) |
| Los Angeles | 342899241.19 | 78720320493.25 (99.57%) | 776720906.15 (69.37%) |
| New York | 76268952.57 | 11097066516.90 (99.32%) | 159631211.60 (67.67%) |
| Seattle | 521826814.17 | 33059192099.65 (98.45%) | 817454723.36 (61.04%) |

***Table 3.2.4.1*** *MSE of final LSTM models for each city compared to baseline results.*

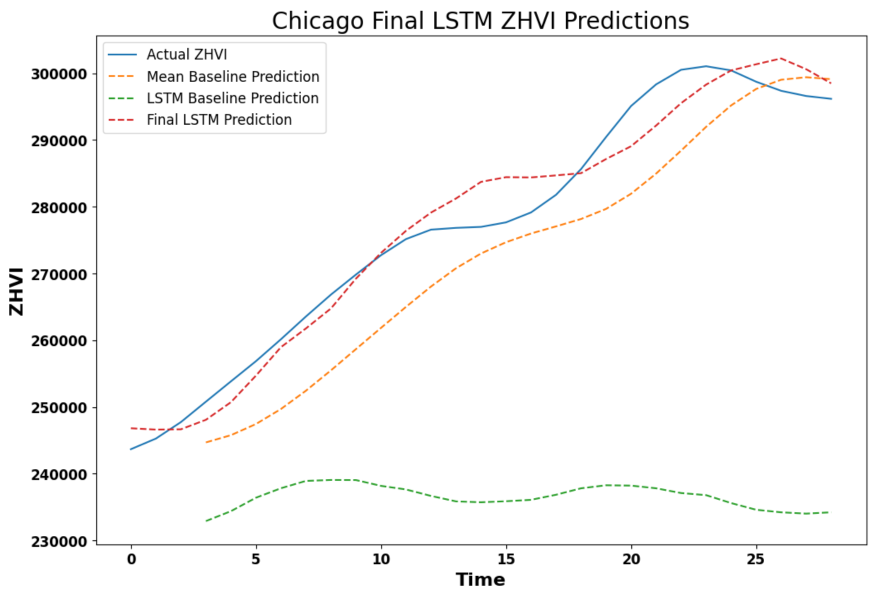


***Figure 3.2.4.1*** *Final LSTM, baseline LSTM, and mean baseline predictions versus actuals.*

## *3.3 Reinforcement Learning*

Reinforcement learning is a type of machine learning that optimizes decision-making. One implementation, Q-learning, takes current state and future action pairs to iteratively learn actions that maximize reward. Deep Q-Learning has the same objective but uses a neural network as its learning agent. Most Q-Learning applications are in robotics, however researchers have recently experimented with using it in financial forecasting (Kaabar, 2024).

We implemented reinforcement learning models using similar LSTM architectures as described in section 3.2 to assess whether defining forecasting policies within a discrete action space was performant.

### *3.3.1 Architecture*

The discrete action space was defined using monthly percent change in ZHVI. Year-over-year changes greater than 7% (0.583% month-to-month) were flagged as significant increases or decreases, changes between 3 - 7% were marked as normal increases or decreases, and anything below 3% percent was considered relatively unchanged – resulting in a 5-action space. All other features were MinMax scaled.

The baseline LSTM Q-Network featured a single layer LSTM, with an input dimension equal to the full feature-set and a hidden dimension of 50 nodes, connected to a linear layer with an output dimension equal to the action space.

The tuned models leveraged similar architectures and parameters derived from random search (shown in table 3.2.4.1) except the output dimension remained equal to the action space and the ReduceLROnPlateau and EarlyStopping functions were removed. These functions were deemed redundant to the epsilon greedy method for balancing exploration and exploitation when selecting actions. Similarly, the same features with significant permutation importance for the LSTM were maintained.

Baseline and tuned models utilized absolute error to calculate reward and Adam optimizers and MSE loss functions to maximize reward. The maximum reward was 0.

### *3.3.2 Results*

Since reward and MSE are not analogous, the total reward achieved using a nearly identical LSTM and the baseline were compared for model performance. As shown in Figure 3.3.2.1, the reinforced LSTM performed approximately 7% worse than the baseline model, pointing to an opportunity for further model training.

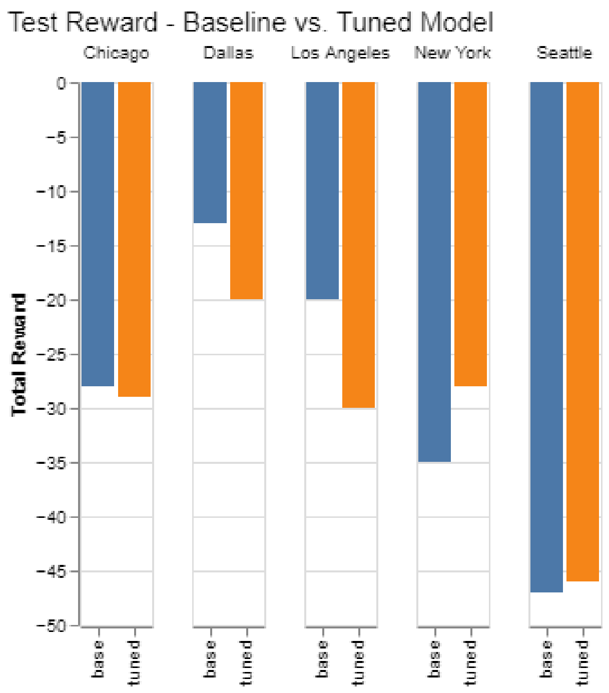


***Figure 3.3.2.1*** *Comparison of total test reward between baseline and tuned Deep Q-Networks.*

The average reward for each test step across baseline models was approximately -0.9 meaning that on average each action taken was about one decision away from the ground truth.

## *3.4 ARIMA*

Despite our sufficiently performant parameterized LSTM models, the team wanted to do its due diligence in comparing new approaches to more traditional ones like ARIMA, SARIMA, and SARIMAX.

### *3.4.1 Preliminary Trending*

AutoARIMA was used to perform gridsearch to guide parameter selection. Chicago metro ZHVI was arbitrarily chosen to build a decomposition plot and conduct an Auto Dickey Fuller (ADF) test for stationarity.

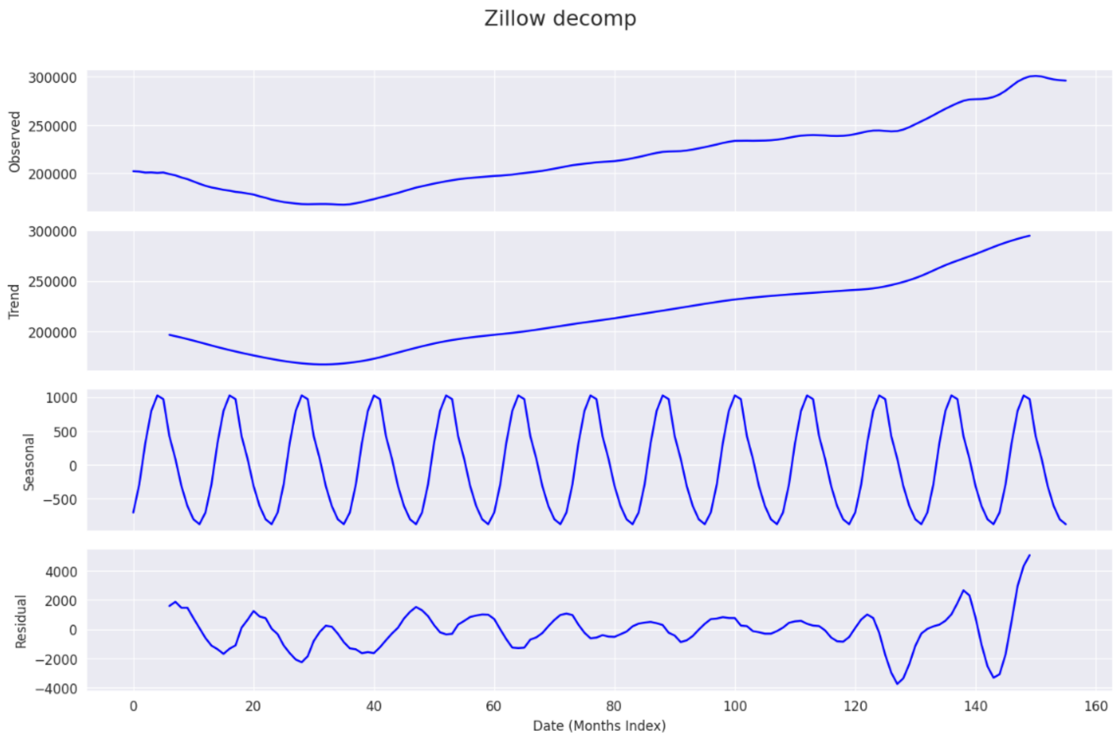


***Figure 3.4.1.1*** *Decomposition of Chicago ZHVI values.*

The ADF test concluded non-stationarity, strong linearity, and insignificance of seasonality. Recursive differencing was applied until stationarity was reached.

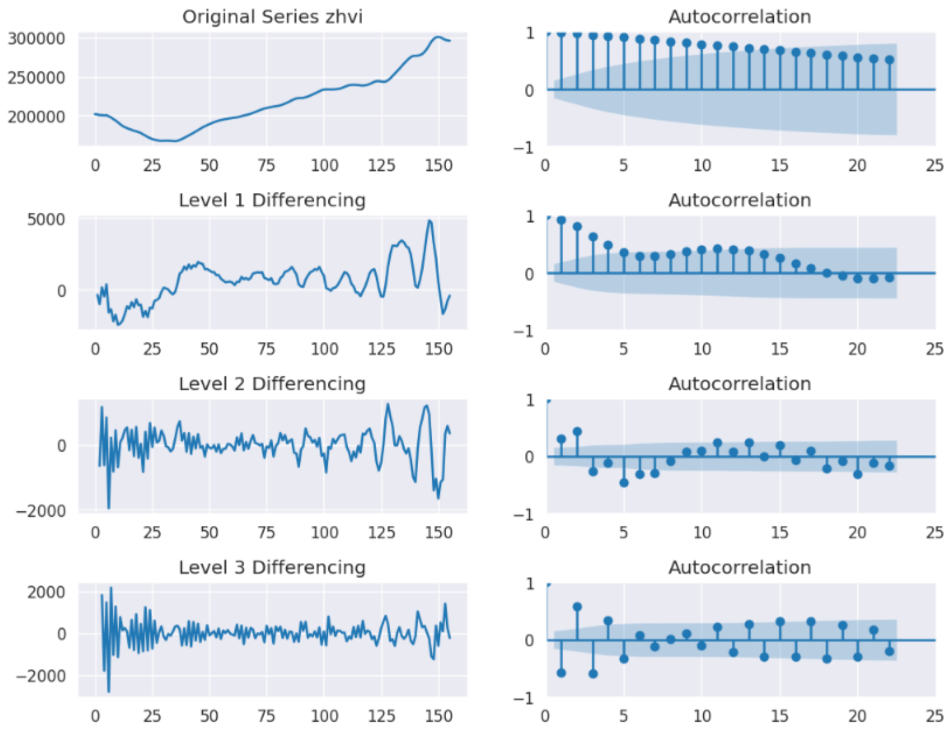


***Figure 3.4.1.2*** *Chicago ZHVI autocorrelation at multiple levels of differencing.*

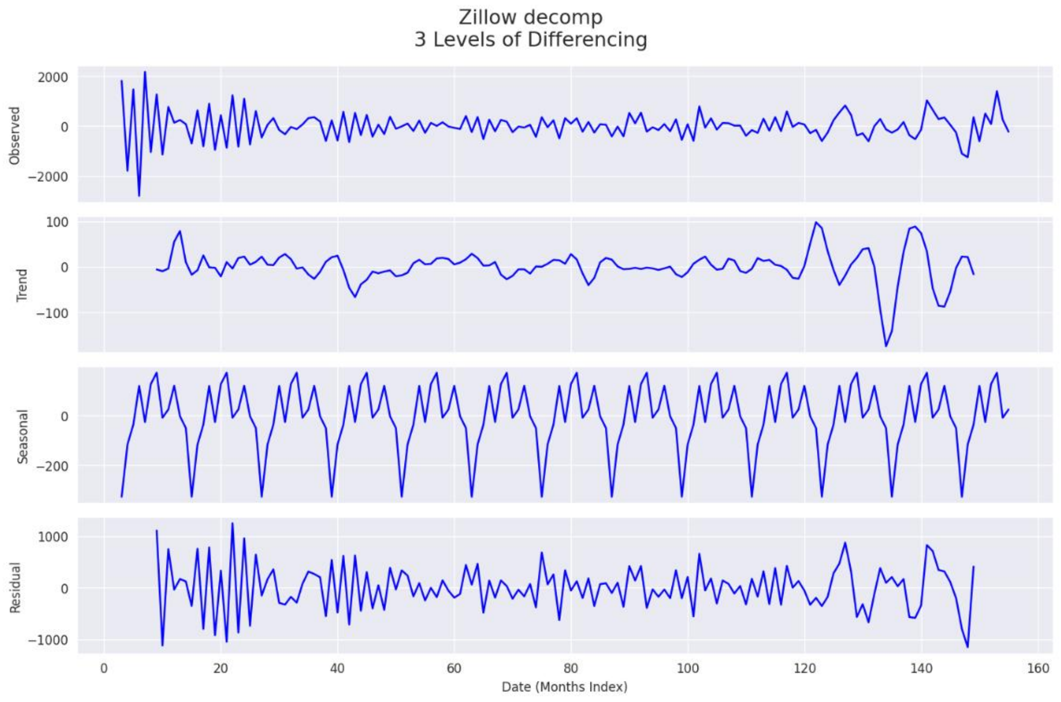


***Figure 3.4.1.3*** *Decomposition of Chicago ZHVI after 3 levels of differencing.*

Figures 3.4.1.2 suggests that stationarity and detrending were achieved at the third level difference while figure 3.4.1.3 portrays seasonality. Based on autocorrelation and visual inspection of plots, the suggested parameters are

- Autoregression: p[1-4] P[1-15],
- Number of differencing: d[2-3] D[2-15],
- Moving average: q[2-5] Q[2-15],
- Seasonality (m): [3,4,12,15].

### *3.4.2 Architecture*

SARIMA and SARIMAX models were implemented for each metro using the statsmodels AutoArima function hosted by the sktime package. For the SARIMAX models, exogenous variables were scaled using sklearn's RobustScaler (chosen to meet the zero mean and unit variance conditions) and dimensionally reduced with PCA where 95% of explained variance was maintained. It is important to note that SARIMAX requires future exogenous value to make its predictions – this is a drawback for real-time forecasting.

Predicted ZHVI values and intervals were inversely transformed back to their original scale for performance evaluation using MSE. Further evaluation was conducted using diagnostic plots and comparisons against the original time series. Figure 3.4.2.1 illustrates the SARIMA and SARIMAX predictions and confidence intervals when compared to actuals. It is evident that SARIMAX achieves smaller confidence intervals however in many cases these intervals and predictions further deviate from the ground truth when compared to SARIMA results. Also, in many cases, SARIMAX parameters defaulted towards zero implying diminished importance of lagged values in favor of exogenous variables.

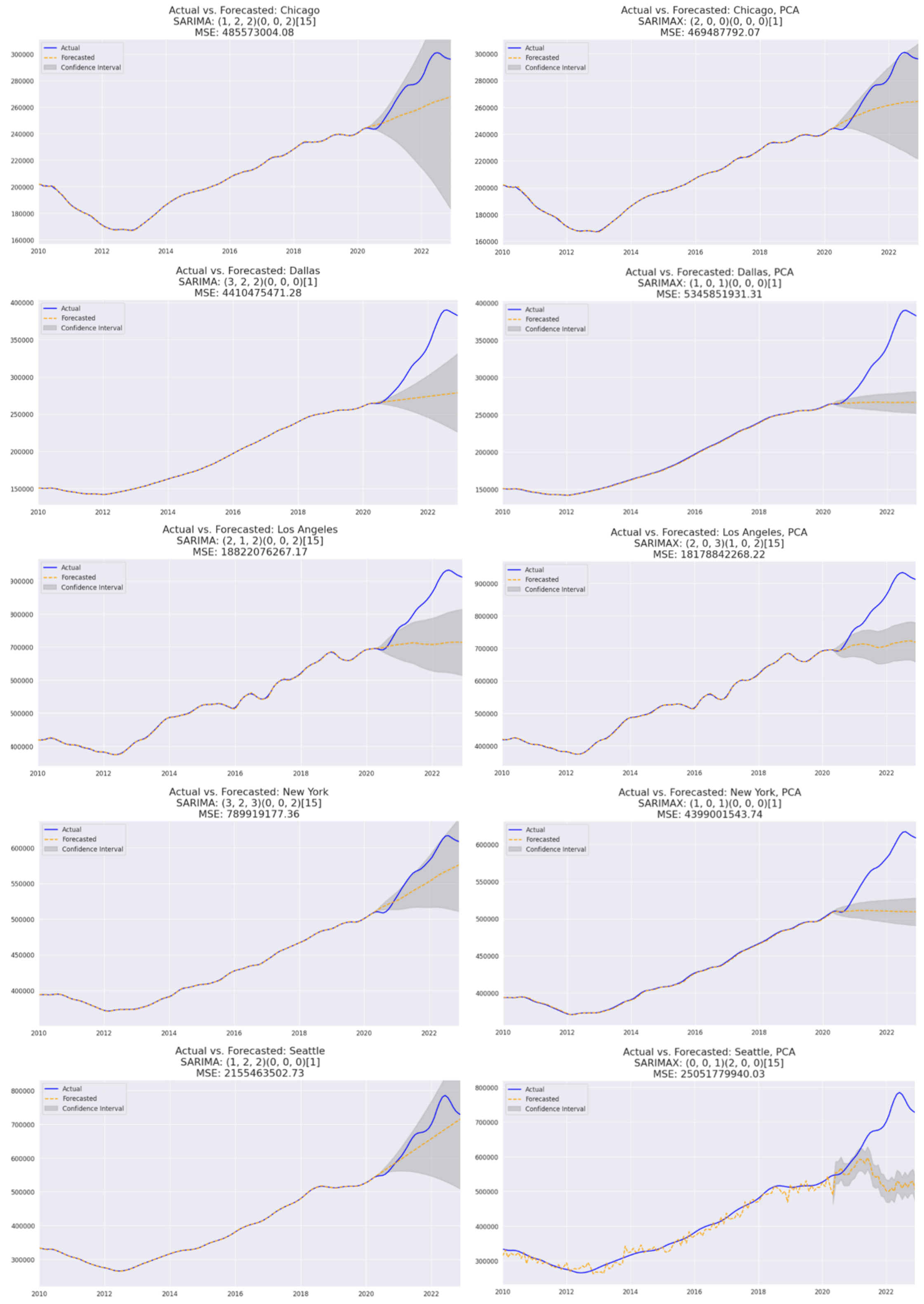
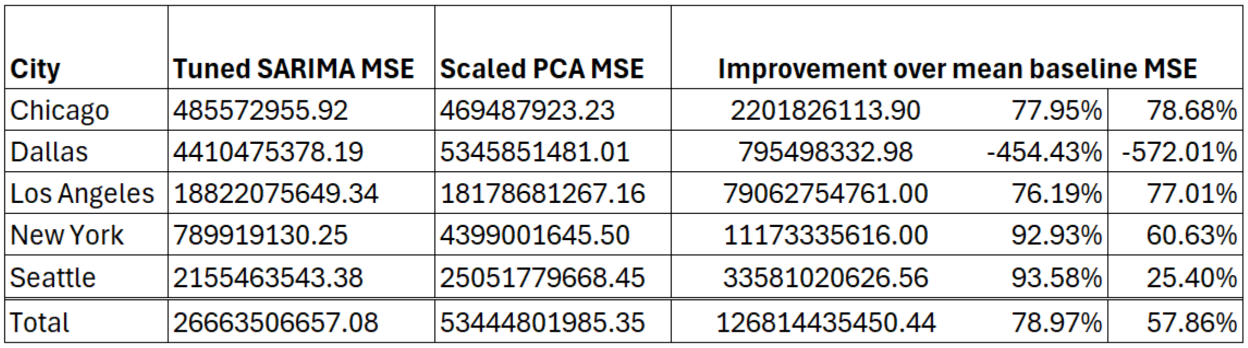


***Figure 3.4.2.1*** *Most performant SARIMA and SARIMAX architectures for each metro.*

## *3.4.3 Results*

Shown in figure 3.4.2.1, SARIMA and SARIMAX maintained reasonable validity for the first 8-10 months of forecasting. These accuracies are more drastically illustrated in table 3.4.3.1 when compared to the mean baseline model's performance.



| City | Tuned SARIMA MSE | Scaled PCA MSE | Improvement over mean baseline MSE | | |
|---|---|---|---|---|---|
| Chicago | 485572955.92 | 469487923.23 | 2201826113.90 | 77.95% | 78.68% |
| Dallas | 4410475378.19 | 5345851481.01 | 795498332.98 | -454.43% | -572.01% |
| Los Angeles | 18822075649.34 | 18178681267.16 | 79062754761.00 | 76.19% | 77.01% |
| New York | 789919130.25 | 4399001645.50 | 11173335616.00 | 92.93% | 60.63% |
| Seattle | 2155463543.38 | 25051779668.45 | 33581020626.56 | 93.58% | 25.40% |
| Total | 26663506657.08 | 53444801985.35 | 126814435450.44 | 78.97% | 57.86% |

***Table 3.4.3.1*** *Tuned SARIMA and SARIMAX MSE compared to Mean Baseline MSE.*

Despite the poor performance of Dallas' model, the overall MSE improvement of SARIMA over mean baseline was 78.97%. While the LSTM models do perform markedly better, these results reveal the sustained efficacy of ARIMA models, even amongst more complex machine learning algorithms.

## 4. ETHICAL CONSIDERATIONS

The driving factor for this project was to inform readers on various forecasting techniques. When informing an audience it is important to assess whether the information is accessible.

Additionally, since we are evaluating multiple models it is critical that we use comparable criteria (e.g. features, window size, train-test split, etc.) to not mislead readers into falsely believing that one model is always superior to another. These models must also be trained on unbiased data; if any of our features disproportionately resembles one facet, our model outcomes may not be generalizable.

Generalizability is especially important when forecasting models have the potential to unjustly influence investment practices, compromising property values for already marginalized groups. Exhibiting full transparency, welcoming constructive feedback, and taking immediate action is the best way to circumvent this.

## 5. DISCUSSION

Our initial goal of making a rental price rating tool was unachievable due to the lack of accessible, granular real estate data. The team notably persevered and defined proxies for many macroeconomic factors and adjusted scope to the available data granularity. While that satisfies the capstone project's requirements, it leaves a lot of area for opportunity to deeply understand property value fluctuations.

While ZHVI closely resembles property values, it more particularly values homes in the 35th to 65th percentile range. This will have negative implications for analyses of communities where property values are heavily skewed. Moreover, this metric, and much of the data we used, is proprietary meaning our team does not have full transparency into how they are calculated nor control over its granularity. This forced us to conform yearly data to monthly using linear interpolation, imputation, and forward filling. In future revisions we would further explore accessible, granular macroeconomic features and expectation maximization (EM) as a more robust interpolation method.

With more time, we would have represented more metros and collected more features. More metros would have enabled clustering, such as K-Means with dynamic time warping, to identify similar cities that contextualize one another. Furthermore, we could develop a unified implementation where all cities are represented in one cohesive model.

Lastly, consistent with our ethical considerations, constant failure analysis and parameter tuning are required to maintain model integrity and constantly refine accuracy.

## 6. STATEMENT OF WORK

Daniel Atallah mined weather, travel, and political data, trained reinforcement learning models, and organized report writing.

Nathan Kitts mined income, rental, and stock data and trained ARIMA models.

Steven Song mined ZHVI, population, new permit, and crime data, trained LSTMs, wrote the ReadMe, and organized the symposium poster.

## 7. REFERENCES

Napier, C. (Host). (2018, March 1). *Interview-Ricardo Barrera, Data Scientist, Trojinn-AI RE Services, Apps-Bellevue* [Radio Broadcast]. In *Block Talk Radio*. https://www.blogtalkradio.com/swas/2018/03/01/interview-ricardo-barrera-data-scientist-trojinn-ai-re-services-apps-bellevue

Kaabar, S. (2024). *Deep Learning for Finance*. O'Reilly Media, Inc.

Kaastra, I., & Boyd, M. (1996). Designing a neural network for forecasting financial and economic time series. Neurocomputing, 10(3), 215–236. https://doi.org/10.1016/0925-2312(95)00039-9

## 8. APPENDIX

*Appendix A: File Share*

| File Name | Description |
|---|---|
| screen_shots/ | Images used in ReadMe instructions |
| temporary_files/ | Storage location ignored by git for any temporary files |
| requirements_windows.txt | Requirements file with windows compatible dependencies |
| requirements_mac.txt | Requirements file with mac compatible dependencies |
| src/Data_Processing/ | Path to all data collection notebooks (run in Colab) |
| src/LSTM_{city_name}.ipynb | Notebook with tuned LSTM for a given city |
| src/LSTM/LSTM_WindowSize_Tuning.ipynb | Notebook for finding optimal LSTM window size |
| src/LSTM/MeanBaseline_LSTMBaseline.ipynb | Notebook for creating baseline models for LSTM |
| src/DQN/dqn.py | Python script containing all DQN classes and functions |
| src/DQN/dqn_{city_name}.ipynb | Notebooks with tuned DQN for a given city |
| src/DQN/dqn_eval.ipynb | Notebook for visualizing DQN performance |
| src/DQN/dqn_base.ipynb | Notebook for creating baseline DQN models |
| src/DQN/dqn_test.ipynb | Sandboxing notebook for DQN |
| src/ARIMA/Arima.ipynb | Notebook for tuning and evaluating all ARIMA models |

Github: https://github.com/StevenSong-sTs/ss24-capstone-team23-datallah-nkitts-steveso

Google Drive: MADS Capstone Team 23

*Appendix B: LSTM*

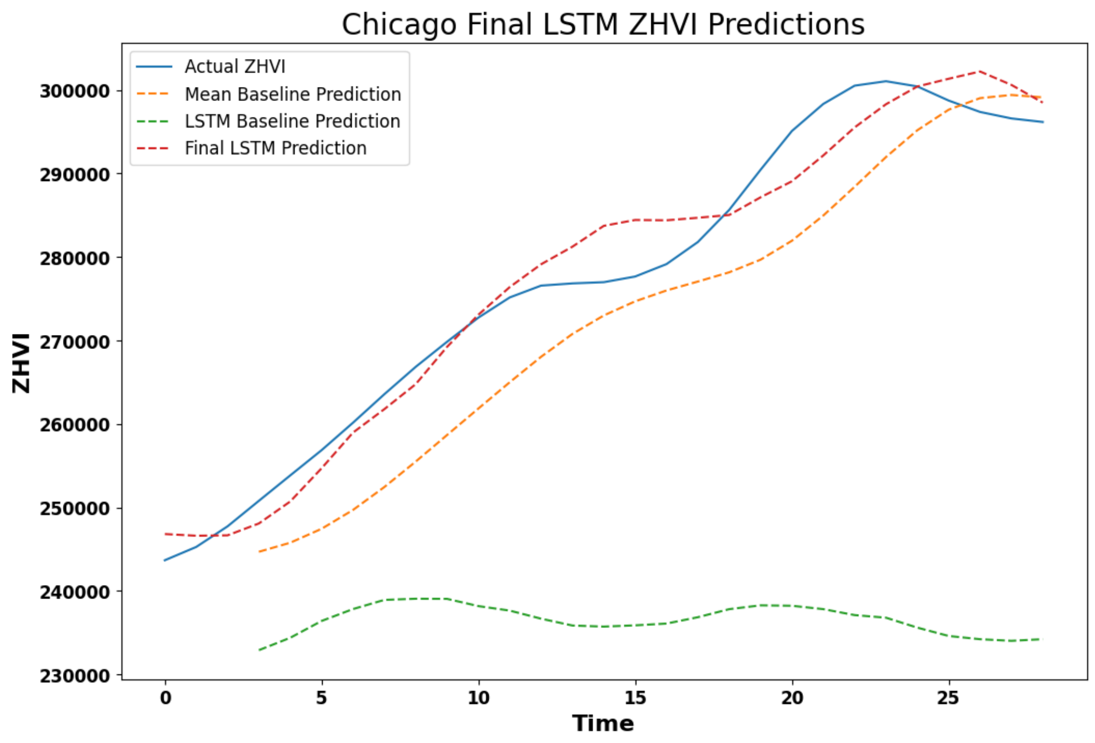


***Figure B.1*** *LSTM final model prediction comparing to baseline models for Chicago*

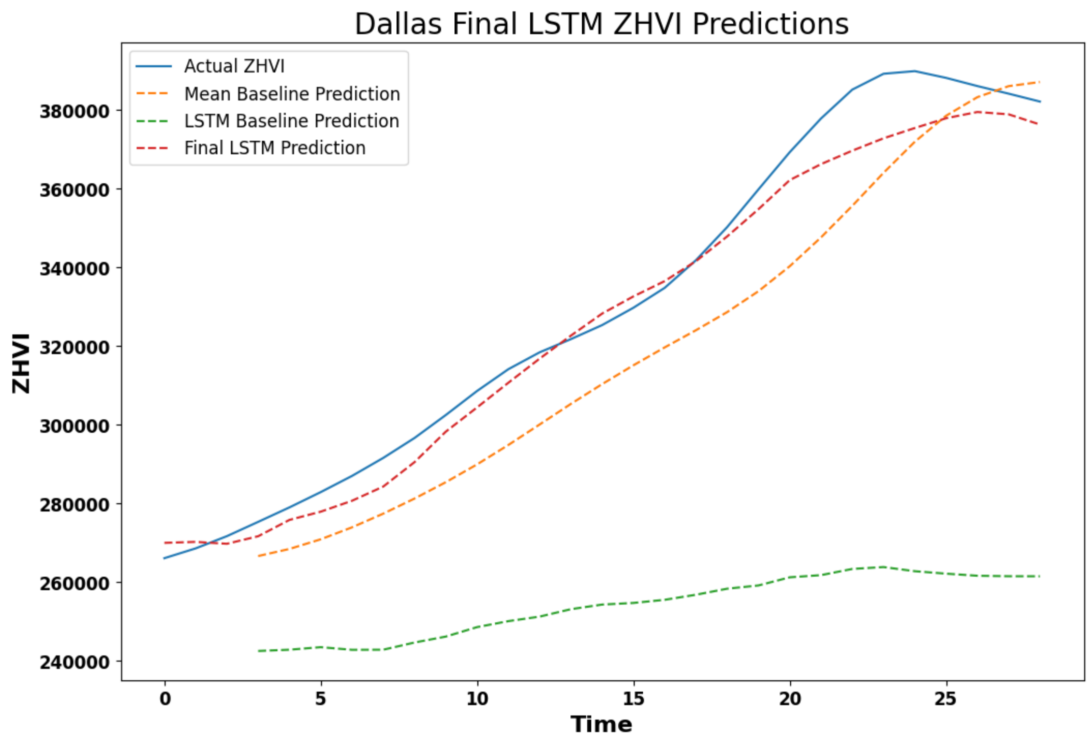


***Figure B.2*** *LSTM final model prediction comparing to baseline models for Dallas*

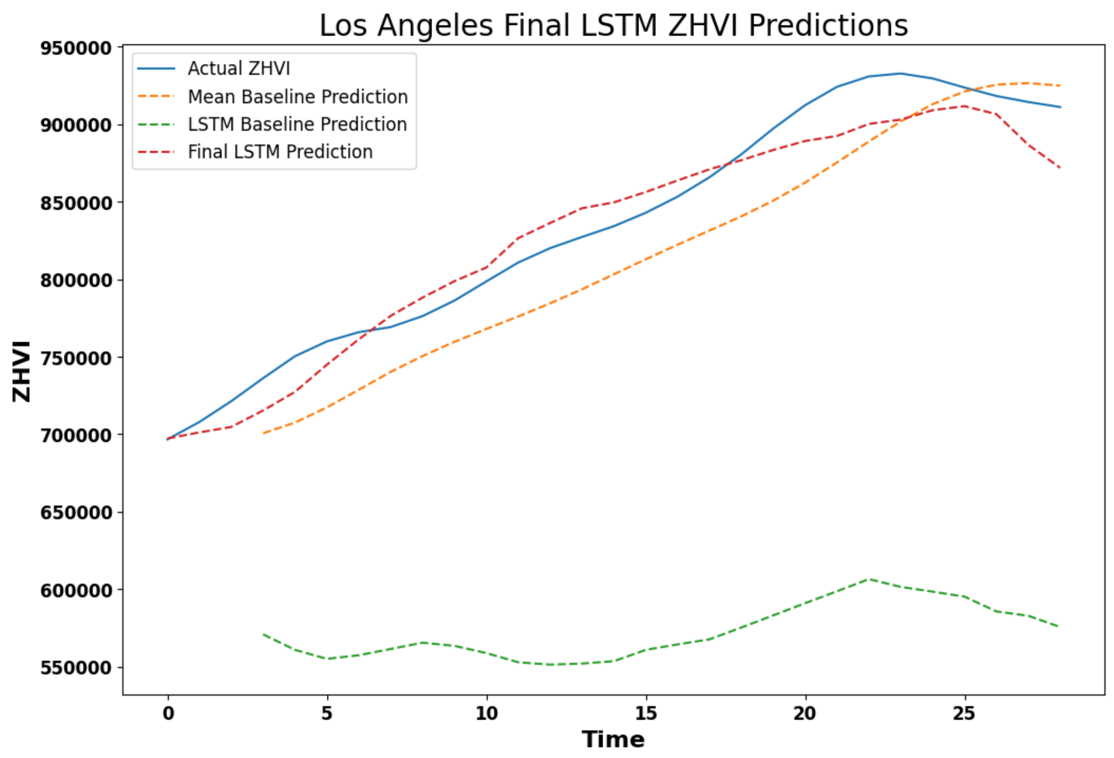


***Figure B.3*** *LSTM final model prediction comparing to baseline models for Los Angeles*

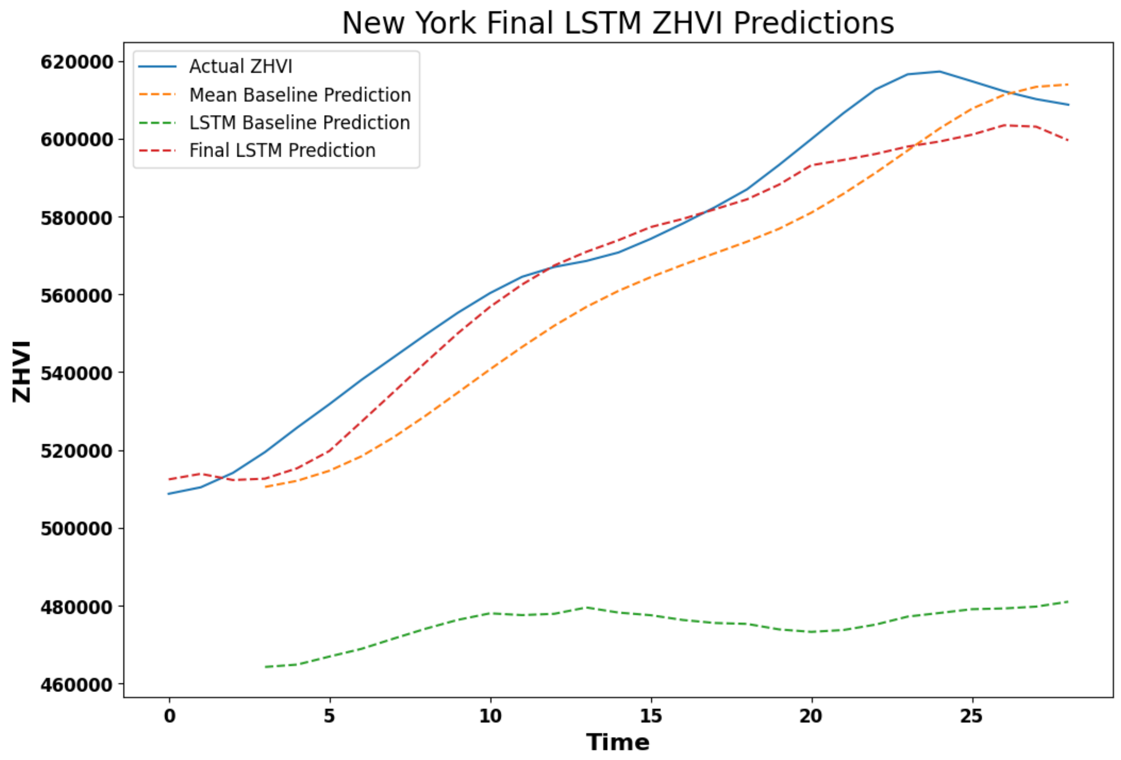


***Figure B.4*** *LSTM final model prediction comparing to baseline models for New York*

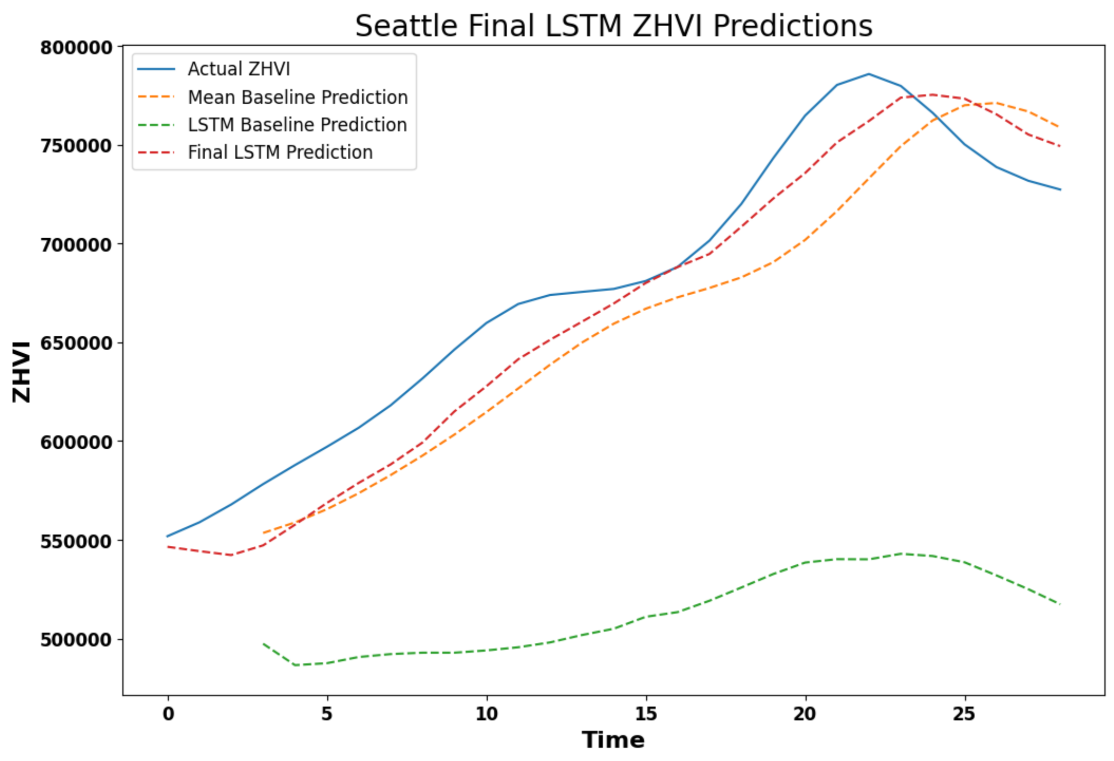
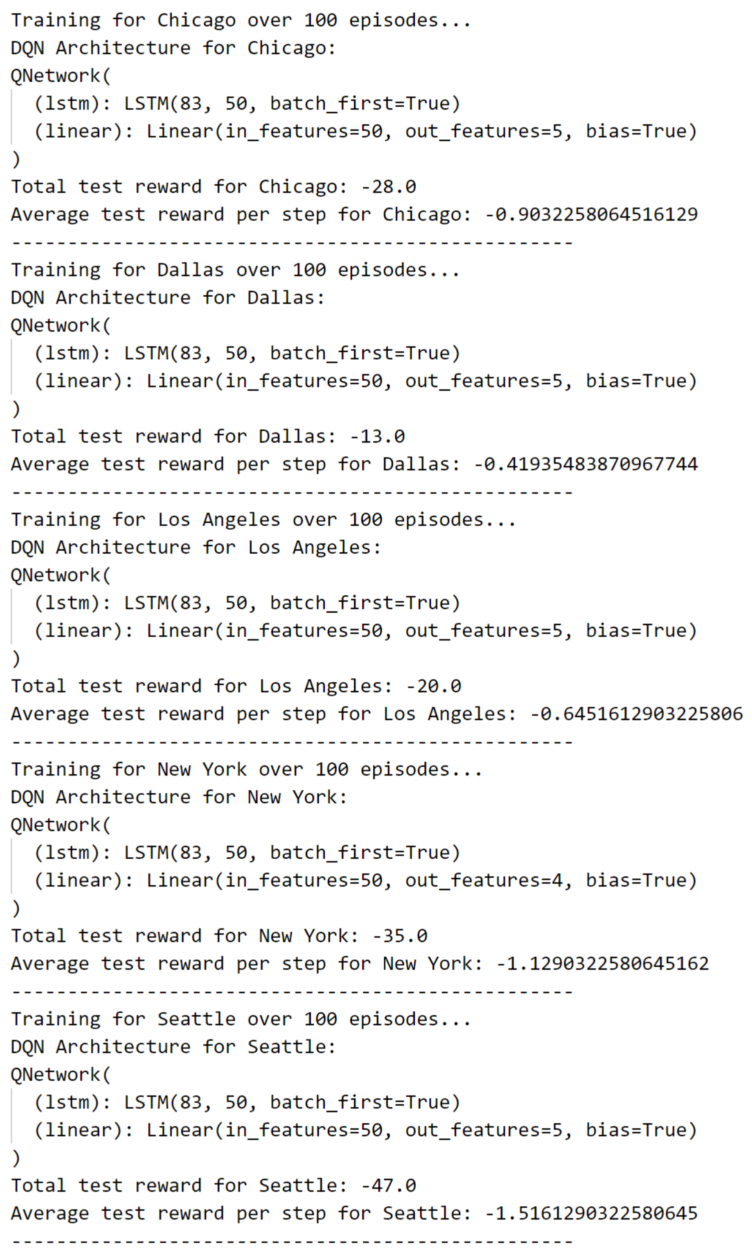


***Figure B.5*** *LSTM final model prediction comparing to baseline models for Seattle*

## Appendix C: DQN



```
Training for Chicago over 100 episodes...
DQN Architecture for Chicago:
QNetwork(
  (lstm): LSTM(83, 50, batch_first=True)
  (linear): Linear(in_features=50, out_features=5, bias=True)
)
Total test reward for Chicago: -28.0
Average test reward per step for Chicago: -0.9032258064516129
--------------------------------------------------
Training for Dallas over 100 episodes...
DQN Architecture for Dallas:
QNetwork(
  (lstm): LSTM(83, 50, batch_first=True)
  (linear): Linear(in_features=50, out_features=5, bias=True)
)
Total test reward for Dallas: -13.0
Average test reward per step for Dallas: -0.41935483870967744
--------------------------------------------------
Training for Los Angeles over 100 episodes...
DQN Architecture for Los Angeles:
QNetwork(
  (lstm): LSTM(83, 50, batch_first=True)
  (linear): Linear(in_features=50, out_features=5, bias=True)
)
Total test reward for Los Angeles: -20.0
Average test reward per step for Los Angeles: -0.6451612903225806
--------------------------------------------------
Training for New York over 100 episodes...
DQN Architecture for New York:
QNetwork(
  (lstm): LSTM(83, 50, batch_first=True)
  (linear): Linear(in_features=50, out_features=4, bias=True)
)
Total test reward for New York: -35.0
Average test reward per step for New York: -1.1290322580645162
--------------------------------------------------
Training for Seattle over 100 episodes...
DQN Architecture for Seattle:
QNetwork(
  (lstm): LSTM(83, 50, batch_first=True)
  (linear): Linear(in_features=50, out_features=5, bias=True)
)
Total test reward for Seattle: -47.0
Average test reward per step for Seattle: -1.5161290322580645
--------------------------------------------------
```

***Figure C.1*** *Baseline DQN architectures with total test reward and average reward per step for each city*

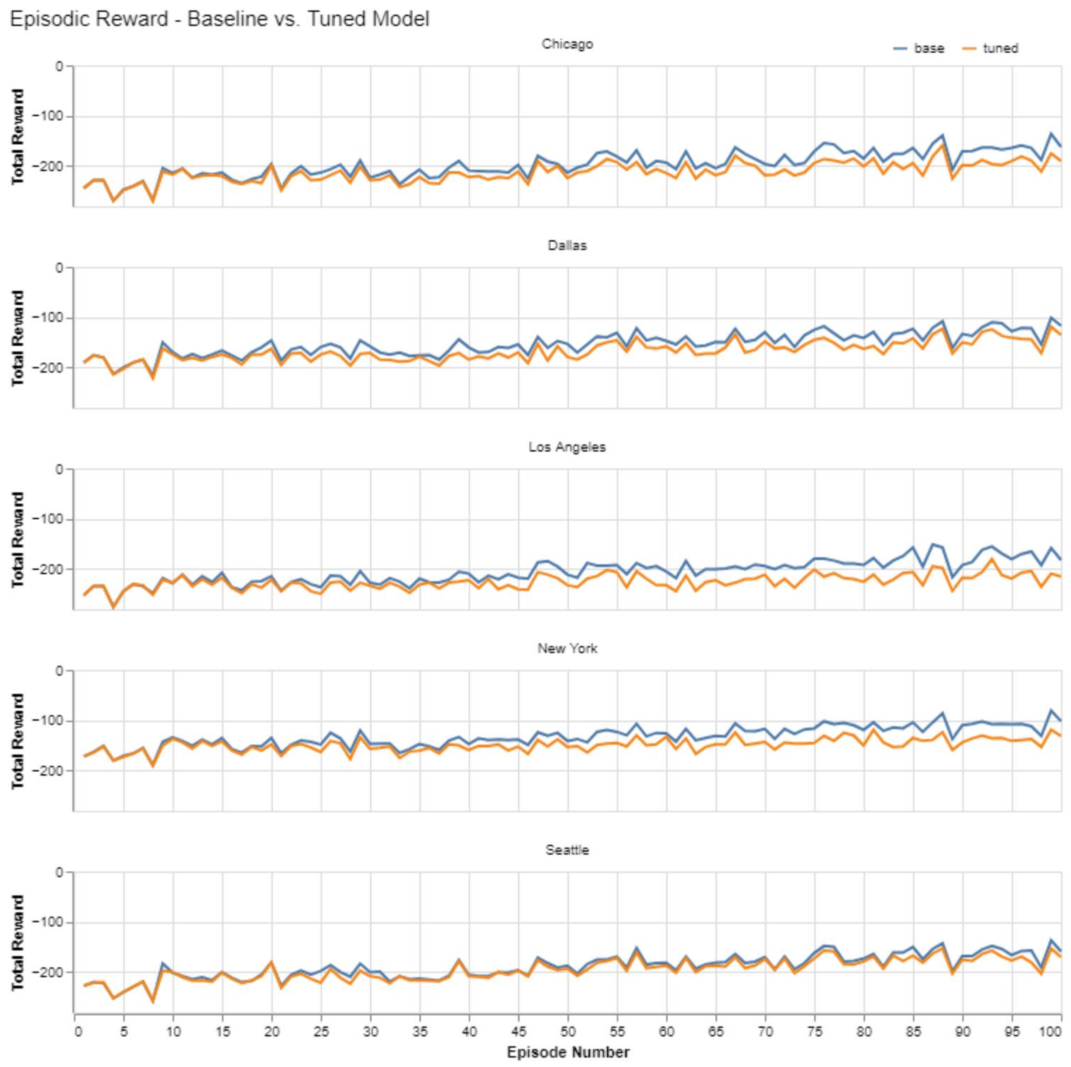


***Figure C.2*** *Total tuned vs. baseline reward per training episode.*

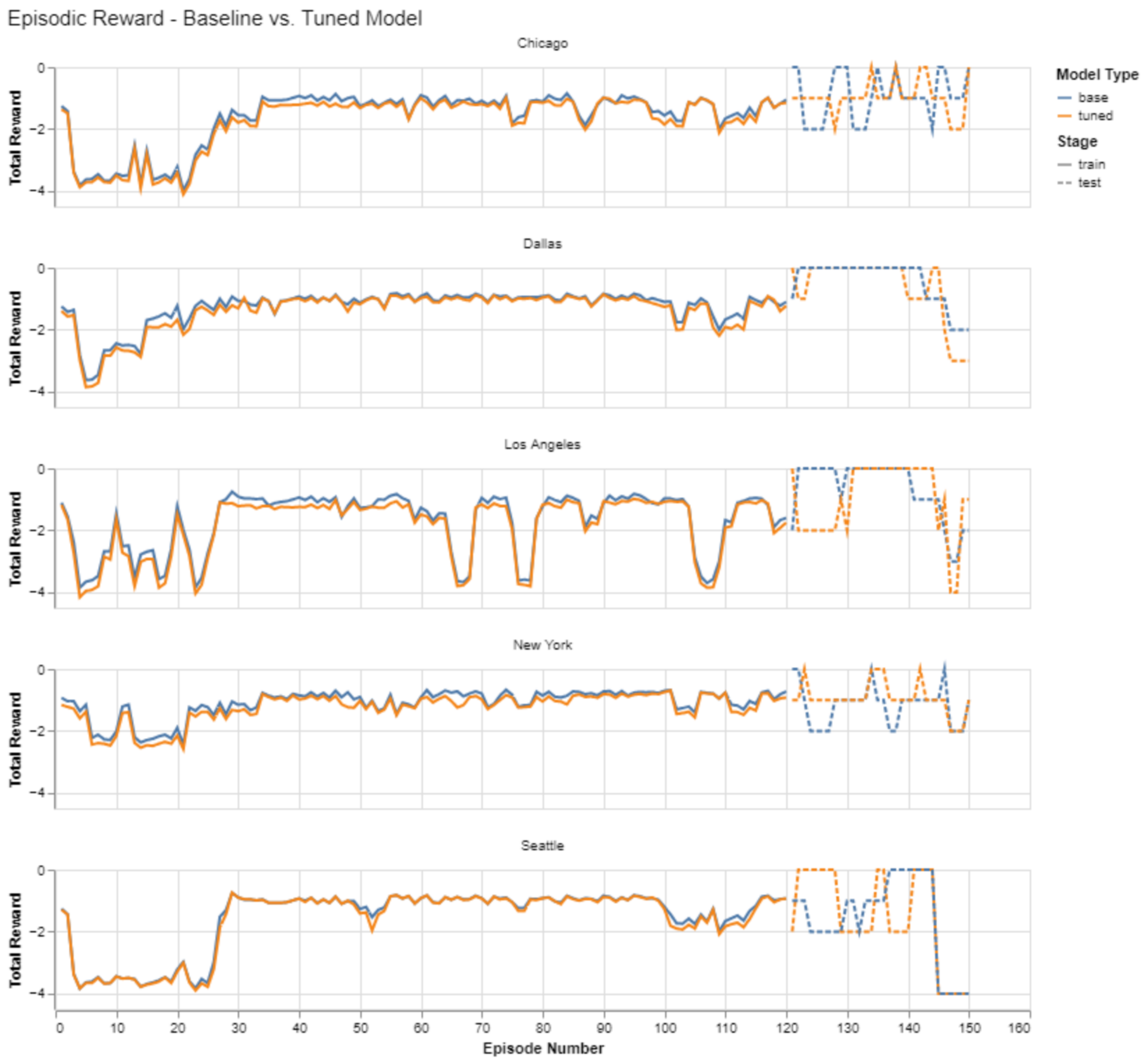


***Figure C.3*** *Average tuned vs. baseline reward per training episode with the test step rewards appended.*

## *Appendix D: Arima*

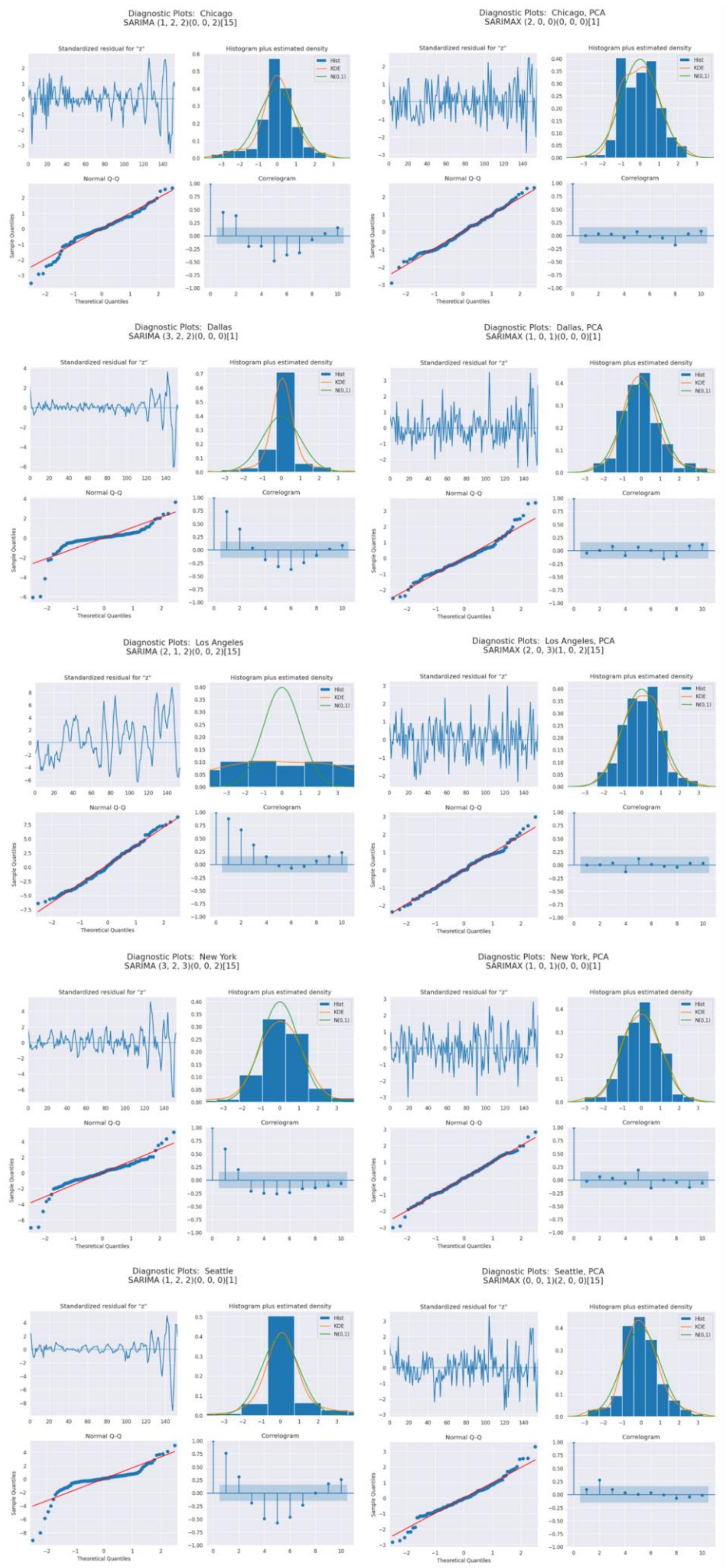


***Figure D.1*** *Diagnostic plots of tuned SARIMA and SARIMAX PCA Models*

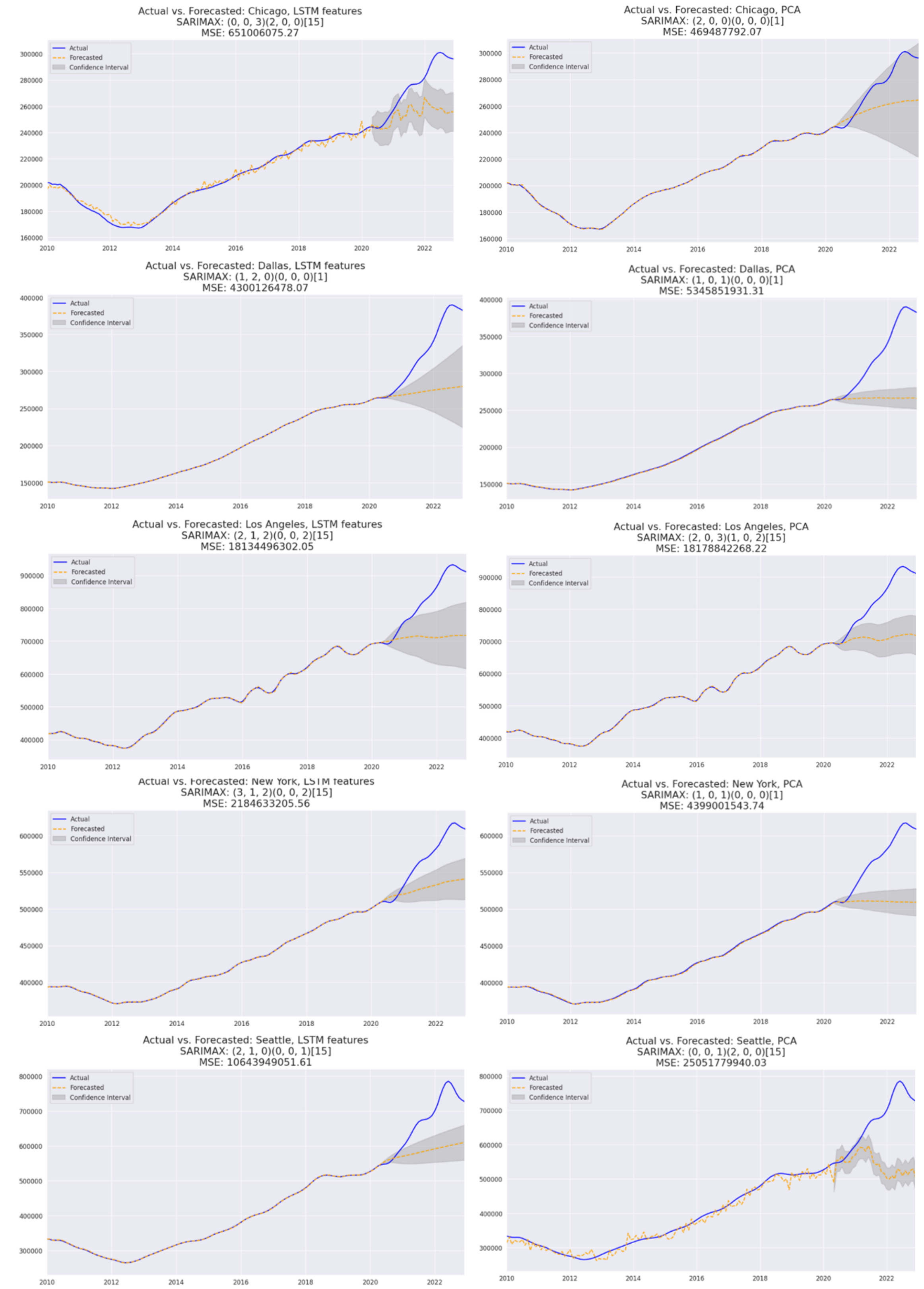


***Figure D.2:*** *SARIMAX models. Left column uses the best features as determined from LSTM analysis. Right column uses PCA of all variables.*