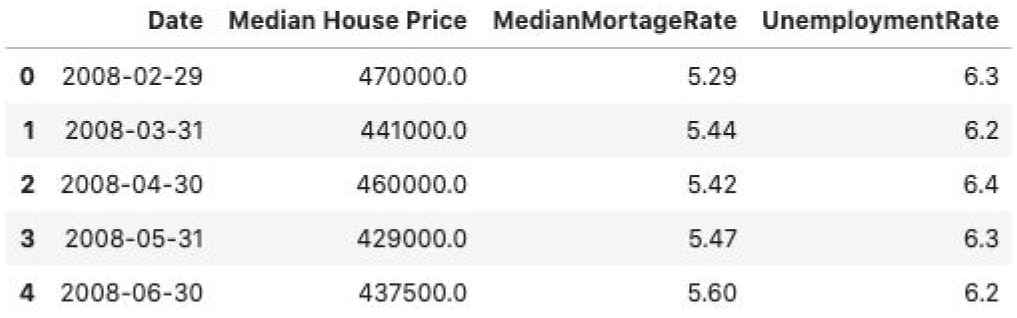
Time Series Final Project

Suren Gunturu
Berkay Canogullari
Remi LeBlanc

Introduction

In this project we are attempting to forecast median house prices for the year 2016. We have data from Zillow that contains monthly data from 2008-2016. In each of these months the data included the median house price, median mortgage rate, and unemployment rate. Below is an example of our data.



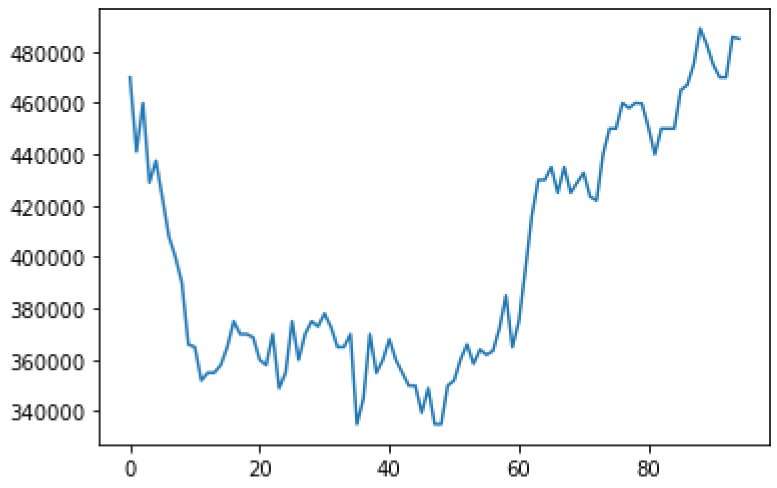
	Date	Median House Price	MedianMortageRate	UnemploymentRate
0	2008-02-29	470000.0	5.29	6.3
1	2008-03-31	441000.0	5.44	6.2
2	2008-04-30	460000.0	5.42	6.4
3	2008-05-31	429000.0	5.47	6.3
4	2008-06-30	437500.0	5.60	6.2

Using the data from 2008-2015 we created many different models with six different methods to predict 2016. The methods we used were: a naive approach, ARIMA/SARMIA, Exponential Smoothing, SARIMAX, VAR, and Prophet. These methods vary from univariate, only using the history of house prices to predict the future, to multivariate, using the exogenous variables of mortgage rate and house prices, as well as historical housing prices to predict the prices in the future.
Each of our models produced different results and based on an error metric we can decide which model best predicts the prices in 2016.

NAIVE APPROACH:

Naive approach is a technique used in time series forecasting where we hand pick our modeling parameters. For this particular implementation of naive approach, we assume that the past data is stationary, doesn't have seasonality, and only depends on past y values and prediction errors. So we will be trying to guess; differencing count, AR order and MA order based on the time series plot, PACF and ACF plots respectively. We try to find a differencing count (d) that makes the data stationary, cut-off value in PACF plot (AR order, p), and cut-off value in ACF plot (MA order, q) just by examining the plots.

Time series plot of the data:



This plot doesn't look stationary (changing mean over time), We need to difference(detrend) it to make it stationary because we can apply our method only with stationary data. That way we're going to determine our differencing order 'd'

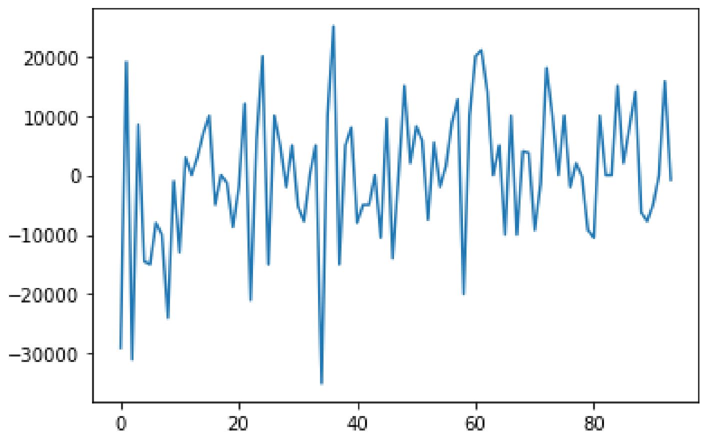
Other than visual inspection, we can apply A.D. Fuller test to check for stationary condition.

```
ADF Results:
Test Statistic   -0.058792
p-value           0.953391
```

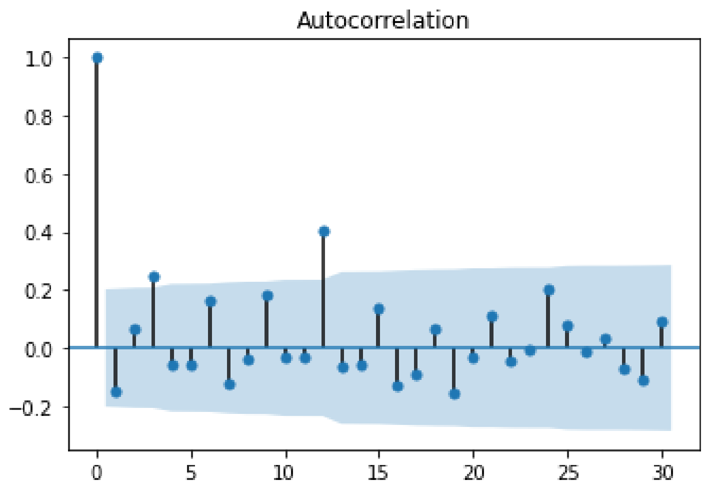
Since we have p-value > 0.05, we confirm that our statement is correct and data isn't stationary.

Difference data once:

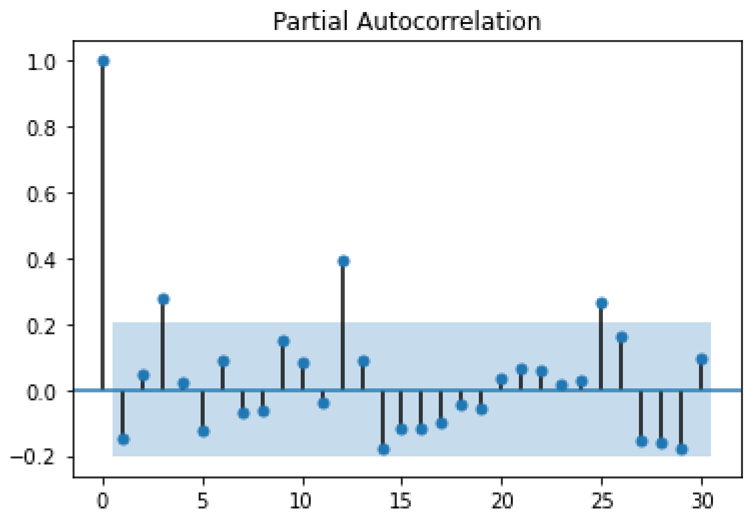
One time differenced time series plot:



One time differenced ACF plot:



One time differenced PACF plot:



Based on time series plot, ACF plot and PACF plot, data looks like it became stationary. We can say that because the time series plot mean is centered around 0 for the whole timeline and neither ACF or PACF plots tail-off. However, we should still apply A. D. Fuller test to make sure we have reached stationary.

One time differenced ADF results:

```
Test Statistic   -3.088139
p-value          0.027443
```

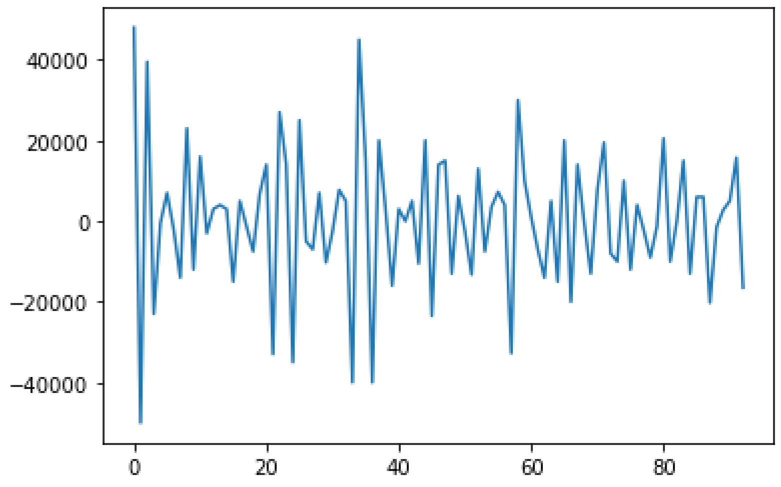
ADF test also suggests that we reached stationary since p-value < 0.05

Based on one time differenced ACF and PACF plots, since there is not cut-off or tail-off, we chose p = q = 0

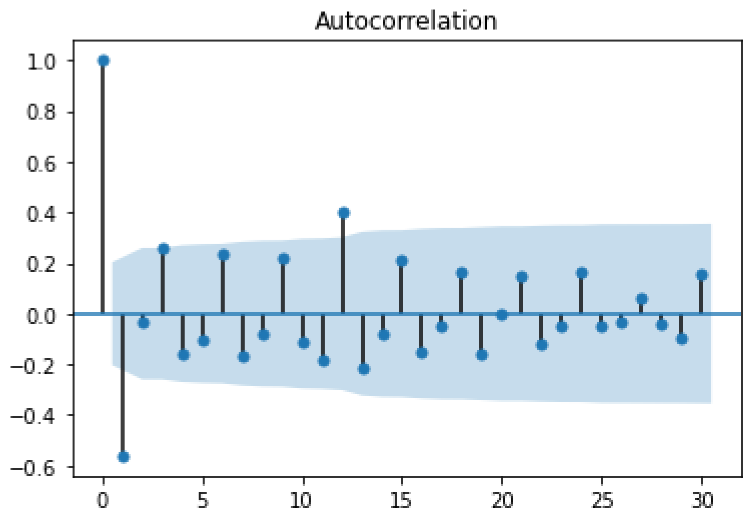
Then our naive approach **candidate model 1 is ARIMA (0,1,0)** since we differentiated data once and reached stationary. By looking at ACF and PACF plots, we determined p = q = 0

We can try to difference the data once more, just to see how we would do, that will be our candidate model 2 for the naive approach

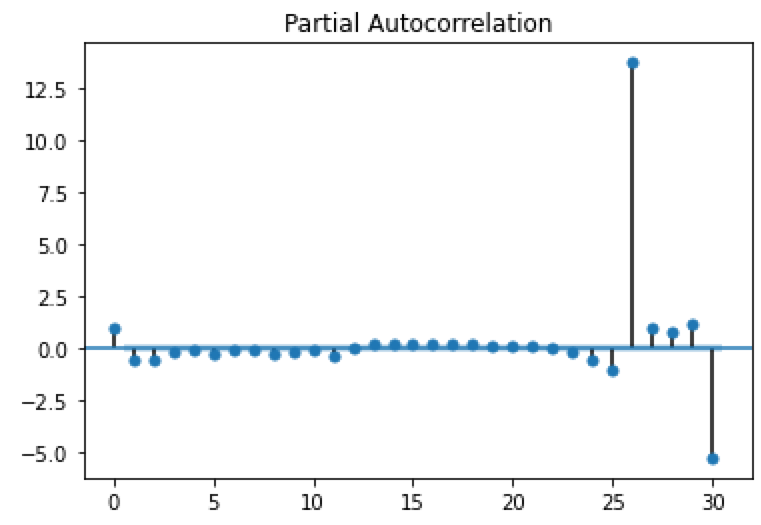
Two time differenced time series plot:



Two time differenced ACF plot:



Two time differenced PACF plot:



Two time differenced ADF results:

```
Test Statistic   -7.390432
p-value          8.027576e-11
```

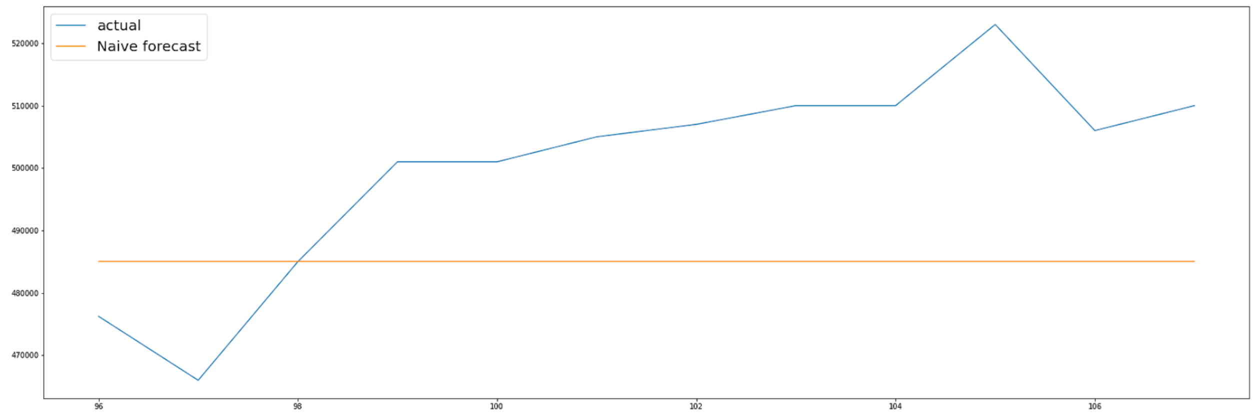
We still have a stationary condition as expected, but this time it looks like we have a cutoff at 1 for ACF plot. This indicates q=1. PACF plot doesn't have cutoff or tail off, this indicates p=0. Our second candidate is ARIMA(0,2,1). We will let CV results decide on which naive model is better

We applied one step 0.8 Train/Test split Cross Validation and used RMSE as a metric to decide on the better model based on CV.

Model 1 CV RMSE = 8161.6
Model 2 CV RMSE = 10714.1

They are really close but we get lower RMSE with candidate 1. Candidate 1 is our finalized model for Naive approach. We want to see how it performs on test set forecast to be able to compare performance with other models based on RMSE.

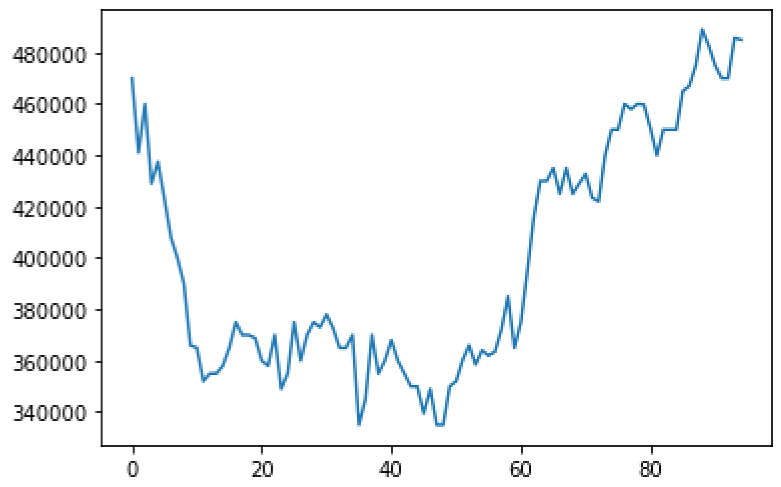
Naive Approach Forecast Results:

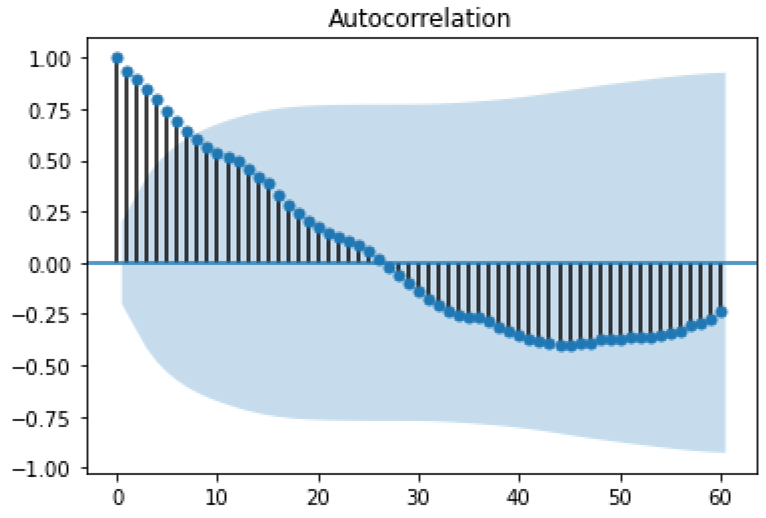


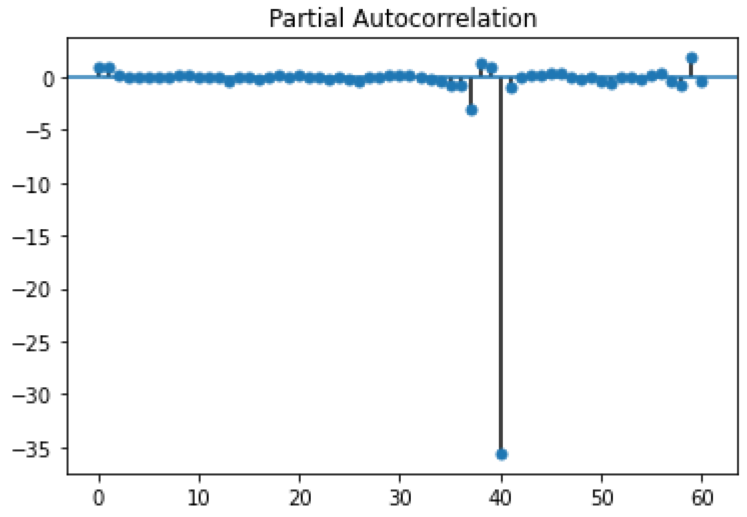
We get RMSE ~ 21590 with the naive approach. But as we can see in the plot, we're predicting the same value over the forecasting period. This doesn't seem pleasing. Let's see if this forecast can be improved by considering the impact of seasonality by choosing candidate models by using SARIMA model.

ARIMA/SARIMA:

ARIMA(Autoregressive Integrated Moving Average) is actually a class of models that explains a given time series based on its own past values, that is, its own lags and the lagged forecast errors, so that equation can be used to forecast future values*. ARIMA models work based on 3 terms; p, d, q. As explained in the Naive Approach, p is the order of the AR term, d is the number of differencing to make data stationary, and q is the order of the MA term. ARIMA doesn't consider seasonality. If we want to consider seasonality, ARIMA model becomes SARIMA (Seasonal ARIMA) with extra seasonal terms; P, D, Q, m.


480000
460000
440000
420000
400000
380000
360000
340000
0
20
40
60
80


Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
10
20
30
40
50
60


Partial Autocorrelation
0
−5
−10
−15
−20
−25
−30
−35
0
10
20
30
40
50
60

Based on the time series and ACF plots we can see that there is a significant trend but it's not certain if there is seasonality. Since we want to consider the effect of trend and also want to see if seasonality has an effect, we're going to use ARIMA and SARIMA models. Since we have monthly data of housing prices, we can consider half a year, and year seasonality. This way, we will have 3 candidates to look at, no seasonality, half year seasonality and year seasonality. We will let the BIC process decide on the parameters p,q,P,Q. Reason we're using BIC but not cross validation is to save time as parameter search using CV takes really long time for SARIMA models. From the naive approach, we know differencing the data once is enough, so we're not going to search for parameter 'd'. We're going to select d=1. Because we wouldn't want to over differentiate and overfit our model

Candidate 1 ARIMA:

Search Range for BIC:

p values = 0, 1, 2, 3, 4, 5
d value = 1 (one time differencing)
q values = 0, 1, 2, 3, 4, 5

Best parameters according to BIC: ARIMA(0,1,4)

Candidate 2 SARIMA with half a year seasonality:

Search Range for BIC:

p values = 0, 1, 2, 3, 4, 5
d value = 1 (one time differencing)
q values = 0, 1, 2, 3, 4, 5
P values = 0,1,2
Q values = 0,1,2
m = 6 (half year seasonality)

Best parameters according to BIC: SARIMA((0, 1, 4), (1, 1, 2, 6))

Candidate 3 SARIMA with yearly seasonality

p values = 0, 1, 2, 3, 4, 5
d value = 1 (one time differencing)
q values = 0, 1, 2, 3, 4, 5

P values = 0,1,2
Q values = 0,1,2
m = 12 (year seasonality)

Best parameters according to BIC: SARIMA((4, 1, 0), (2, 1, 0, 12))

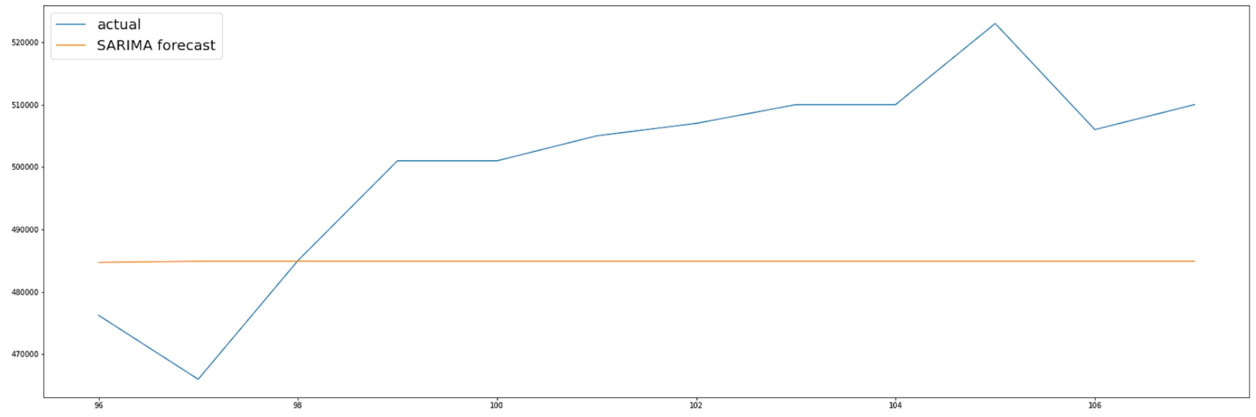
Now that we have our candidate models, we're going to let CV decide on which ARIMA/SARIMA model is best based on the RMSE.

We applied one step 0.8 Train/Test split Cross Validation and used RMSE as a metric to decide on the better model based on CV.

Model 1 CV RMSE = 8339.3
Model 2 CV RMSE = 10906.7
Model 3 CV RMSE = 9433.7

Based on the CV results, ARIMA (0, 1, 4) is the winner(SARIMA with no seasonality)

ARIMA/SARIMA Forecast Results:



We get RMSE ~ 21652 with this approach. Looks like this didn’t perform any better than the naive approach. We should try other alternative algorithms to see if they perform any better.

Exponential Smoothing:

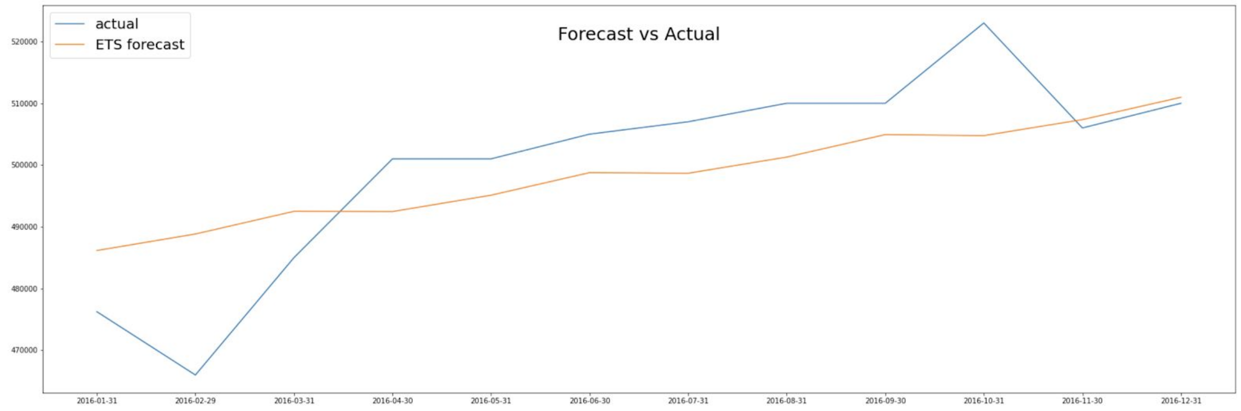
The Holt-Winters Exponential Smoothing is a univariate technique that makes a prediction that is a weighted sum of past observations, with exponentially decaying weights. This technique is a special, less general version of an ARIMA model, but does not require order selection. It is a great “fast and dirty” option to time series forecasting.

In order to pick our model we tried all possible combinations of additive and multiplicative for both trend and seasonality. We chose the model that gave us the lowest mean absolute percent error on our history, training data. Below is each model we tested, with the MAPE score on the right.

```
Trend=additive          Seasonal=additive          0.014495729004909462
Trend=multiplicative    Seasonal=additive          0.01516373012715186
Trend=additive          Seasonal=multiplicative    0.014590703740706803
Trend=multiplicative    Seasonal=multiplicative    0.014381959600608493
Trend=None              Seasonal=additive          0.01457561384117051
Trend=None              Seasonal=multiplicative    0.014382832526105855
```

The model with a multiplicative trend and a multiplicative seasonality had the lowest MAPE of 0.01438. This is the model we used to forecast the 2016 housing prices.

Below is a graph of the actual future data (blue) and our predicted future data (orange).



This model gave us a MAPE of 0.01744, and an **RMSE of 10,522.25**.

SARIMAX:

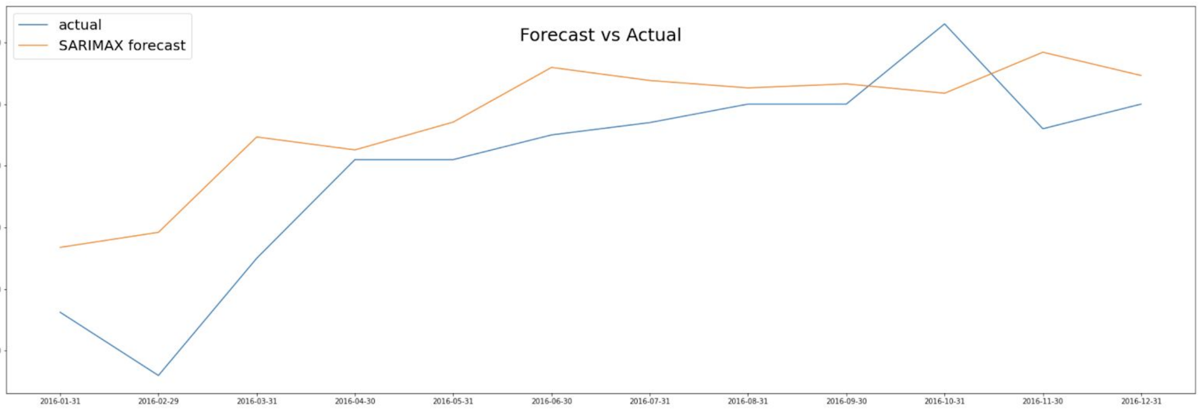
This modeling technique is useful when you have multiple variables that were recorded with the same time steps, that may have relationship/correlation with each other. In our Zillow house data, we have data that would make this technique useful. We have monthly data with features median house price, median mortgage rate, as well as unemployment rate. We used median house price as our endogenous, outcome response variable, and median mortgage rate and unemployment rate as our exogenous variables to help predict the house prices.

In order to get better predictions we normalized our features using MinMaxScalar from scikit learn. We then used the auto_arima function from the pmdarima module to help choose our best model (https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.arima.auto_arima.html). In the auto_arima function we can pass in what values we want to pick from when creating models to compare. Below are the parameters we chose to run this function. The variables p,d,q,P,D,Q,m are the same as described before.

```
start_p=0, start_q=0, max_p=4, max_q=4, start_d=0, max_d=2, m=3,
start_P=0, start_Q=0, max_P=3, max_Q=3, D=1,
```

The best model based on AIC from the auto_arima function was ARIMA(2,0,1)(3,1,0)[3]. From this model we predicted our future data, three times in order to fit our scaler's shape. Once we had our predictions we needed to inverse transform the data back to its original form.

Below is a graph of the actual future data (blue) and our predicted future data (orange).

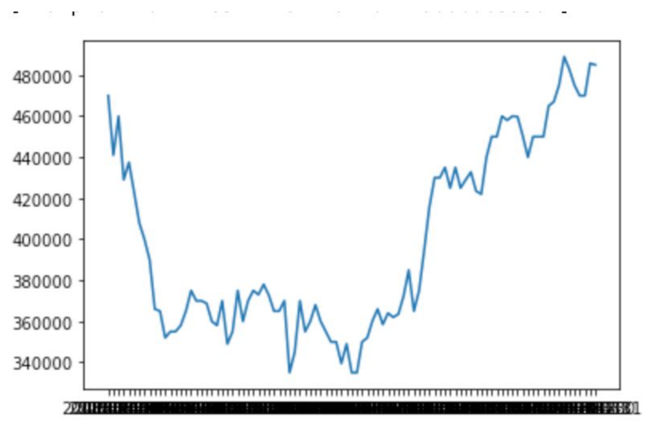


This model gave us a MAPE score of 0.018578, and an **RMSE of 11,287.42**.

VAR:

For both the Var and the Prophet models, we split the training dataset into a train and validation test for testing to see the best differencing numbers for the Var method. The validation data was the last 6 values (the last half year) of the training dataset.

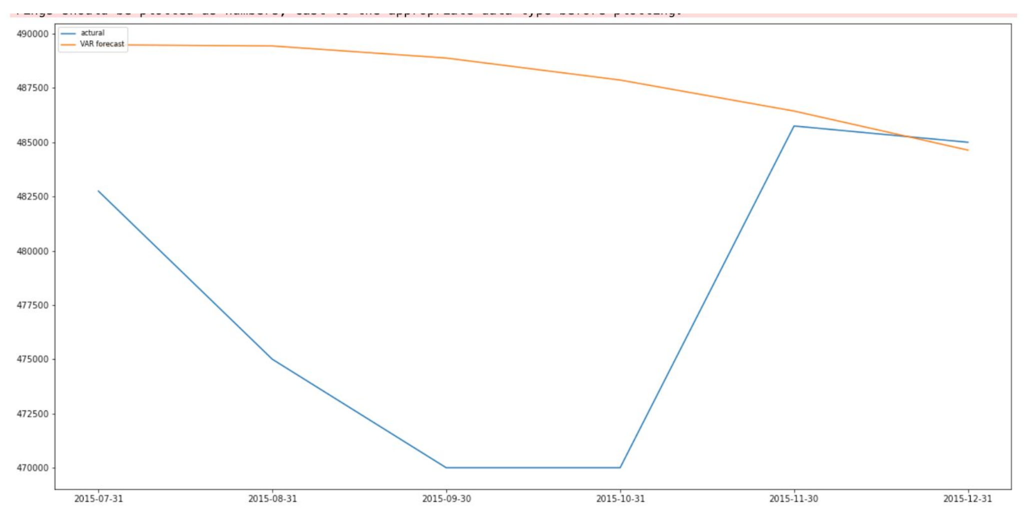
The Var (Vector Autoregressive) method is one that is used for multivariate time series. In other words, we will be modeling the median house price at the current timestep as a linear function of past lags of both itself (the median house price of previous timesteps) as well as previous timesteps of other variables presented in the training data: Median Mortgage Rate and Unemployment Rate. We used this method in order to figure out if there is a relationship between previous time steps of these other variables and the median house price of the current time step. The plot of the time series data is shown as below:



Based on this time series, we can tell that from the beginning to the end of the time range, there isn't much change in median house prices but it does change within the time range as it decreases and then increases back up. Because of this, we had to figure out whether to difference the data or not. In order to do Var, we had to make sure the data for median house prices as well as the other features are stationary and through ADF, we found out that the data is not stationary as shown below:

```
Median House Price
Test Statistic   −0.068773
p−value           0.952461
dtype: float64

Median Mortgage Rate
Test Statistic   −1.728379
p−value           0.416553
dtype: float64

Unemployment Rate
Test Statistic   −1.728379
p−value           0.416553
dtype: float64
```

However, we tried to check using Var on this undifferenced data set to see whether it makes a difference with model results. With this resulting model, we got the following plot comparing forecast with Var of the training data to the actual values in the validation data:



where the x axis is time and the y axis is median house price. This model result was pretty good with the **RMSE at about 12,451**. As shown later, compared to differencing the data, this gave the best result.

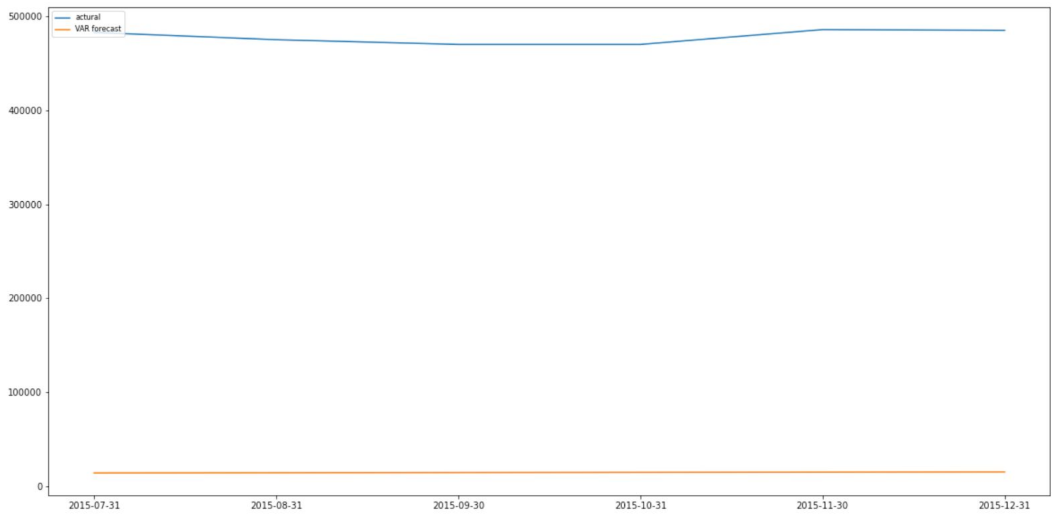
Next, we tried to difference the data once, and once we did this, we got favorable ADF results that show that each variable is stationary as shown below:

```
Median House Price
Test Statistic   -2.872365
p-value           0.048657
dtype: float64

Median Mortgage Rate
Test Statistic   -8.221108e+00
p-value           6.426158e-13
dtype: float64

Unemployment Rate
Test Statistic   -8.221108e+00
p-value           6.426158e-13
dtype: float64
```

So with this undifferenced data, we got a var model with no coefficients, just an intercept term as shown below:

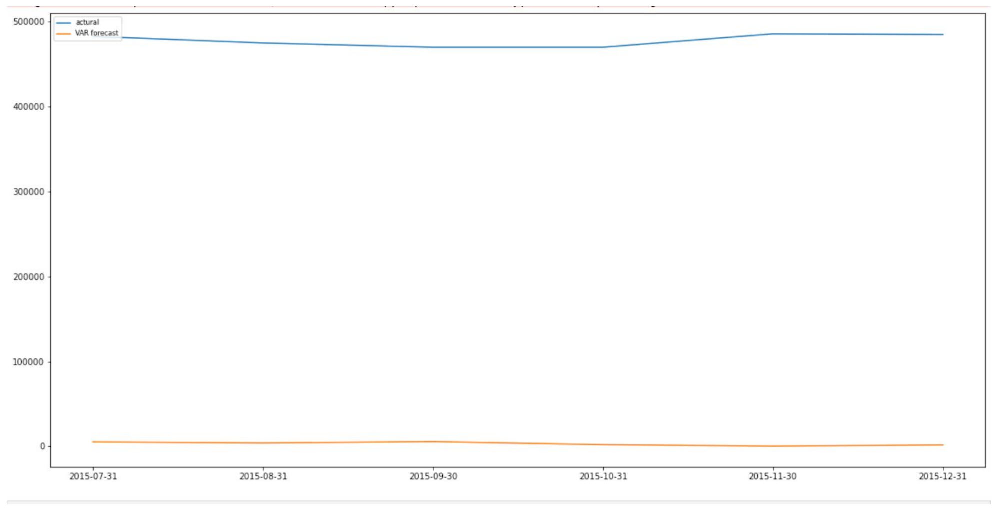
```
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Sun, 07, Mar, 2021
Time:                     18:53:45
--------------------------------------------------------------------
No. of Equations:         3.00000    BIC:                    14.3233
Nobs:                     88.0000    HQIC:                   14.2729
Log likelihood:          -998.109    FPE:                1.52704e+06
AIC:                      14.2388    Det(Omega_mle):     1.47615e+06
--------------------------------------------------------------------
Results for equation Median_House_Price
========================================================================
           coefficient       std. error           t-stat            prob
------------------------------------------------------------------------
const       215.909091      1256.976234            0.172           0.864
========================================================================

Results for equation MedianMortageRate
========================================================================
           coefficient       std. error           t-stat            prob
------------------------------------------------------------------------
const        -0.016364         0.019941           -0.821           0.412
========================================================================

Results for equation UnemploymentRate
========================================================================
           coefficient       std. error           t-stat            prob
------------------------------------------------------------------------
const        -0.007955         0.059632           -0.133           0.894
========================================================================

Correlation matrix of residuals
                      Median_House_Price  MedianMortageRate  UnemploymentRate
Median_House_Price              1.000000          -0.079358          0.153349
MedianMortageRate              -0.079358           1.000000         -0.040362
UnemploymentRate                0.153349          -0.040362          1.000000
```

Because of this, the results were not very good as the final plot we got comparing the forecasted values of the training data and the validation data are as follows:

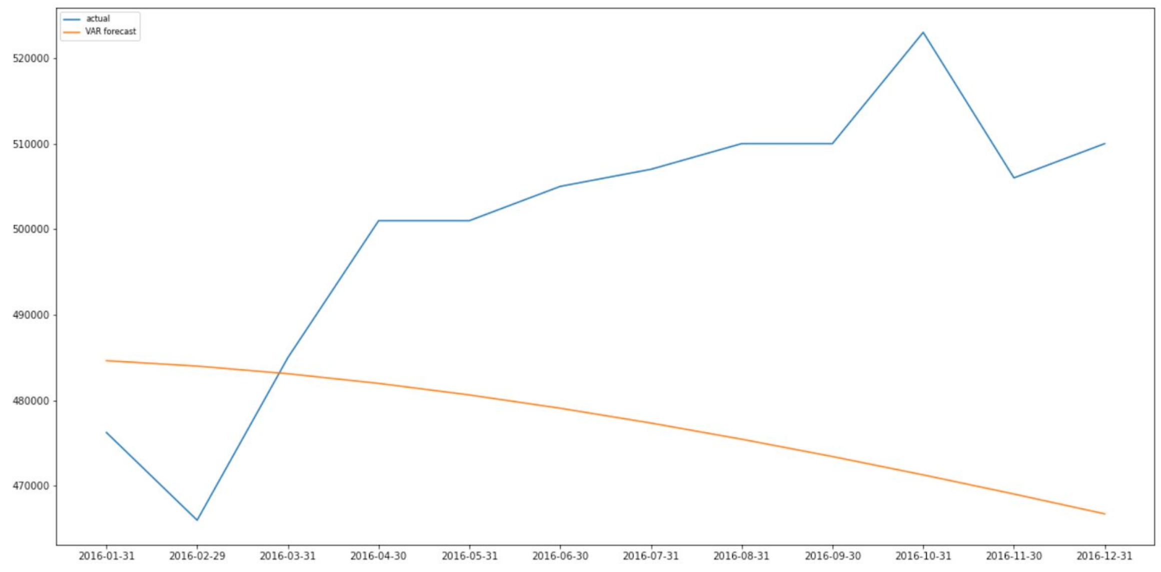


with an **RMSE of more than 463374.4.**

So far, not differencing the data gives the best results. So next, we tried to difference the data one more time (with a difference value of 2). The data differenced twice was even more confident in ADF testing and got coefficients for the Var model. The resulting plot that we got differencing the data twice however was even worse than differencing it once as shown:

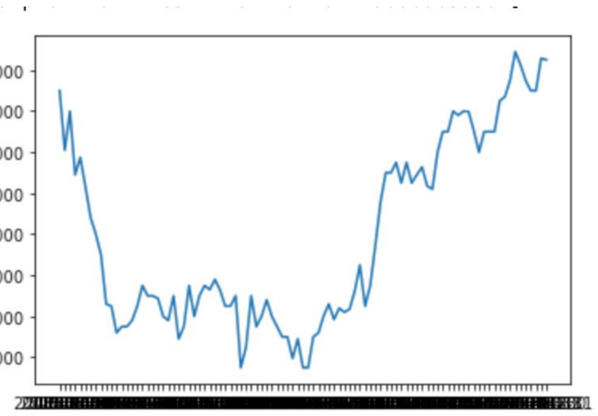


The RMSE we got for this model was **475078.7**. Because of all this, we decided to use the VAR model without differencing at all. Even though VAR requires that the data is stationary, based on the validation forecasting results, it looked like not differencing the data at all gave the best results so we used that model. When we trained that model with the entire training data and tested it on the test results, we got a plot of the test forecasting results as shown below:



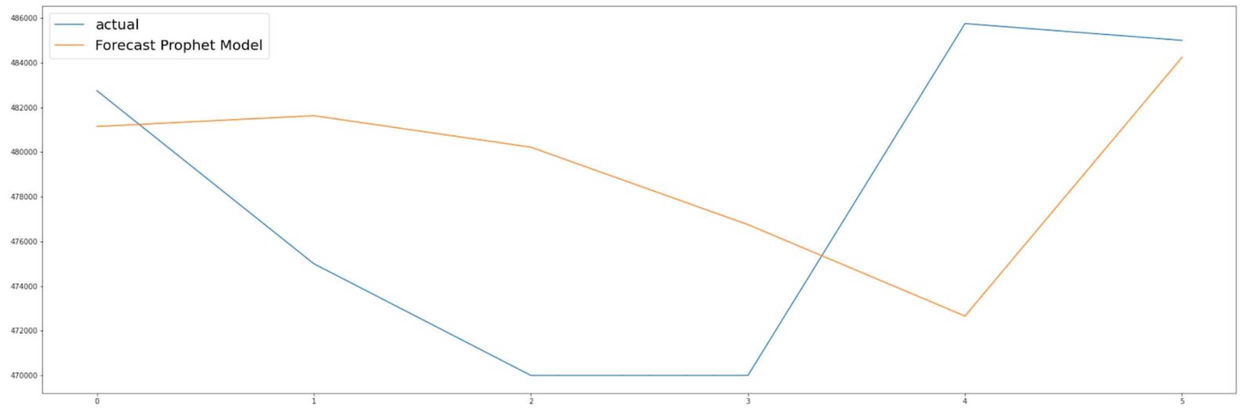
with a test **RMSE of 30502.1.** This result is reasonable, but still not the best in terms of the best model used, so we ended up trying another method: the Prophet model, to see whether it makes better forecasting results.

Prophet:

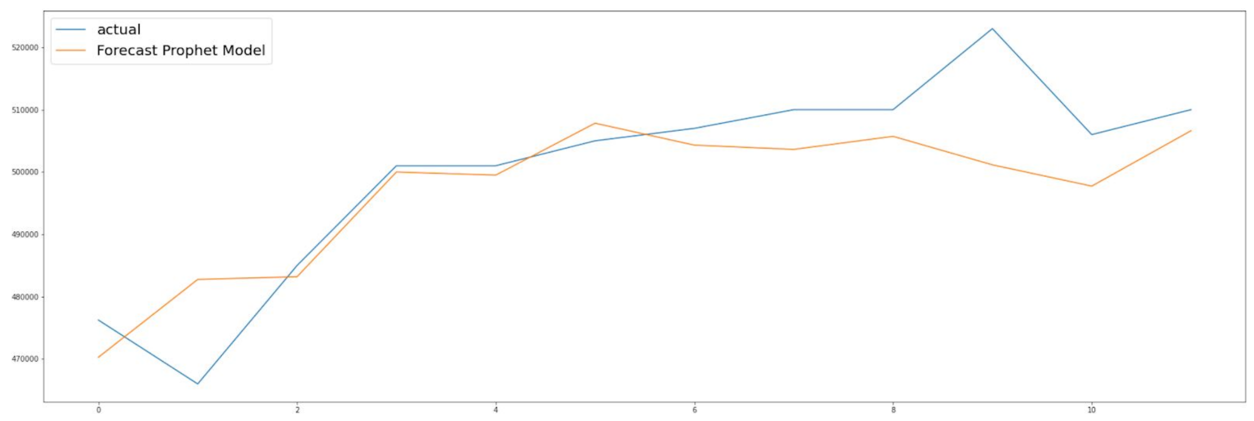


Above as shown again is the time series plot of the median house prices. This plot does have a somewhat repeated pattern that is shown almost like a quadratic graph. Because of this, we decided to look more into Prophet as a viable solution to forecast median house prices. Prophet is also one of the best solutions for long term forecasting, and since the test dataset forecasts data for one year (2016), we thought this could be the best option.

As described in the VAR section, we split the training data into train and validation time series sets where validation was the last half year of 2015. We did this to first test out how well the prophet model behaves. Once we fit the prophet model on the train data, then we modeled the forecast of the next half year with the actual validation data. The plot is shown below:



The validation data is very similar in the plot with the forecasted values from the prophet model. This gave us a validation **RMSE of 7836.33** which is a considerable improvement over other models such as the naive approach and VAR and very similar but better than the results from SARIMAX. Therefore we decided to use this prophet model with the entire training set and test our results with the test set. As shown below is the plot of the forecasted values up to one year after the training set and the actual values from the test set:



The plots for the test set and the forecasted values from prophet look very similar. In fact, the RMSE for this was very good, with the value being **8919.06.** Because of this, the prophet model was the best model we got in terms of long term forecasting of median house prices

Conclusion:

Model	**RMSE on Test Data Set (Jan - Dec 2016)**
Naive Approach	21,590.05
ARIMA/SARIMA	21,652.24
SARIMAX	11,287.42
Exponential Smoothing	10,522.25
Var	30,502.1
Prophet	8,919.06

The main metric that we used to compare models was RMSE so that it would penalize wrong results more based on the distance between the actual and forecasted results. The more the distance from these two values, the higher the rmse value is. From all the models, we saw that Prophet gives the lowest RMSE among all the models as shown above. The final RMSE of the test data set is **8919.06** and so our best model was Prophet.

*https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/