# Reprice

Mohammad H., Kunal M., and Don F.

# Introduction

Main purpose:

A web-based housing price estimator based on physical and geographical attributes of a property.

Step:

- Retrieve recent three months data(median property values) from Zillow
- Break the process into multiple models each predicting at one level
- Final prediction along with previous predictions mapped to a readable format and presented to the end users.

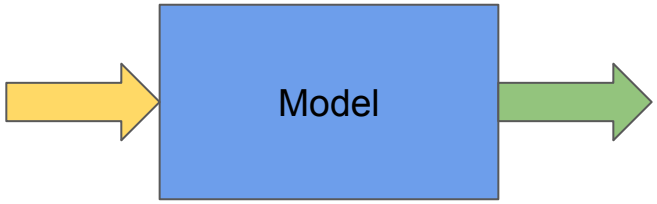
Goal:

Providing a better insight to the end users.

# Modeling

- Goal:

- State
- County
- City
- Neighborhood
- Square Footage

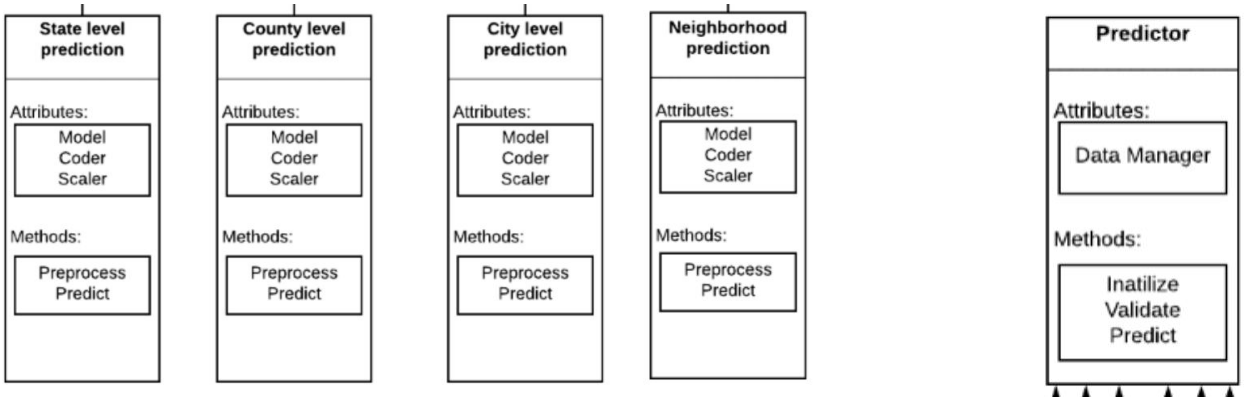



Median Value of Property

# Approach

- A data driven approach
  - Use a dataset of past observations to generate a predictive model
    - Zillow Research
  - Use data mining
  - Supervised Machine Learning
    - Regression

# Class Design



# Data Preprocessing

- Remove Categorical Data
  - Use label encoding

| Label | Mapped Value |
| --- | --- |
| California | 1 |
| Texas | 2 |
| New York | 3 |
| ... | ... |

- A major issue
  - Treating nominal value as ordinal!
  - Use one-hot-encoding
    - High dimensions!

# Data Preprocessing

- Dealing with missing values
  - Simplest way -> just drop those columns (or rows)
    - Reduces the dataset by a lot
  - Fill the missing values by propagating the last valid reading

| | | | | | |
|---|---|---|---|---|---|
| 126 | Nan | Nan | 132 | 141 | 153 |

| | | | | | |
|---|---|---|---|---|---|
| 126 | 132 | 132 | 132 | 141 | 153 |

# Data Preprocessing

- The dataset is median property value through time
  - From 1996-04 to 2019-02
  - This is a time series forecasting problem
    - We should use this!
    - Pass a temporal window of size N on the data

# Data Preprocessing

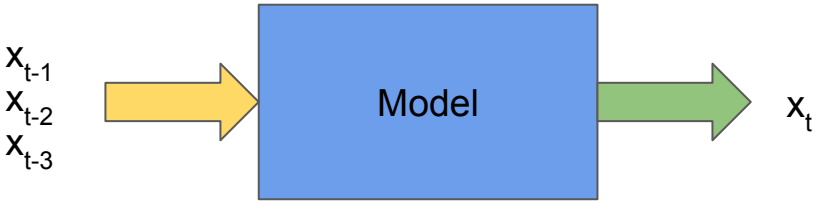
For W = 3 we have

| ... | 2018-07 | 2018-08 | 2018-09 | 2018-10 | 2018-11 | 2018-12 | 2019-01 | 2019-02 |
|---|---|---|---|---|---|---|---|---|
| ... | 1618 | 1616 | 1625 | 1631 | 1624 | 1611 | 1605 | 1595 |

| | | | |
|---|---|---|---|
| 1618 | 1616 | 1625 | 1631 |
| 1616 | 1625 | 1631 | 1624 |
| 1625 | 1631 | 1624 | 1611 |
| 1631 | 1624 | 1611 | 1605 |
| 1624 | 1611 | 1605 | 1595 |

# Model Generation

- Our goal is then to train a model such that




- Many options available
  - Use Random Forest
  - An Ensemble method
  - Bootstrap sample on training set

# Model Generation

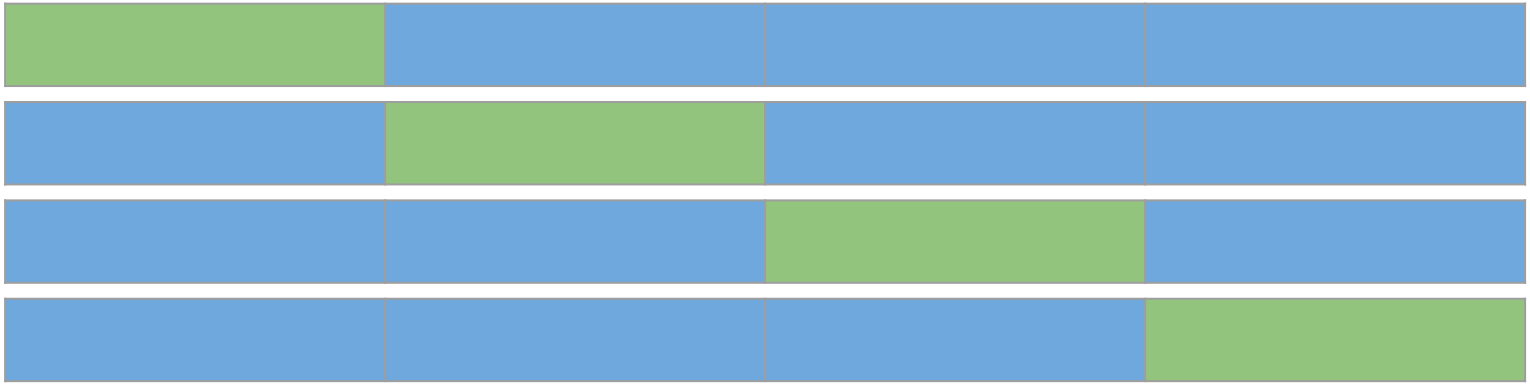
- A model should be evaluated on both
  - How good has it learned the data it has seen
  - How good will it perform on unseen instances

- Split the data into two partitions: train and test
  - Different approaches to partition the data
  - K-Fold cross validation

- Models have different hyperparameters that should be tuned
  - Main hyperparameters of RF:
    - Number of trees
    - Maximum Depth

# Model Generation

- How does K-Fold works?
- Let K = 4

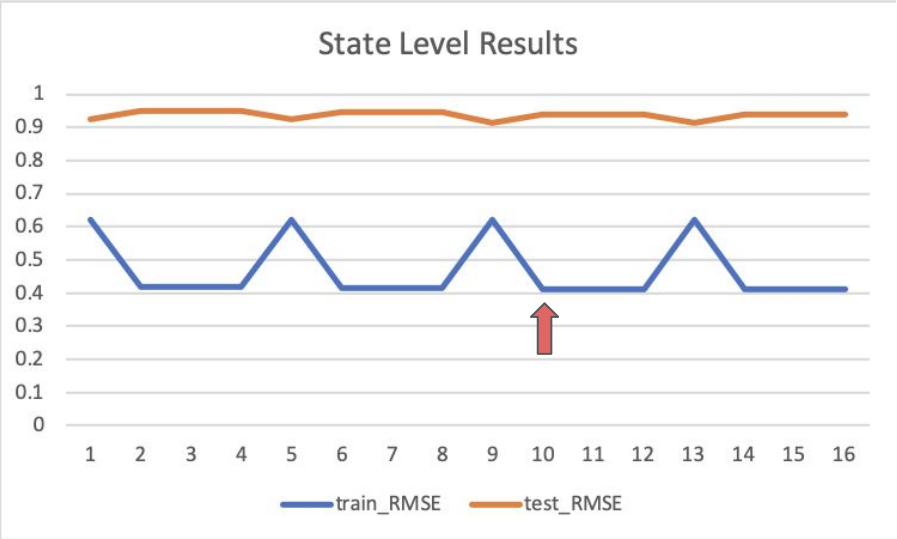


- Find the average performance over train and test folds

# Model Generation

- State Level results

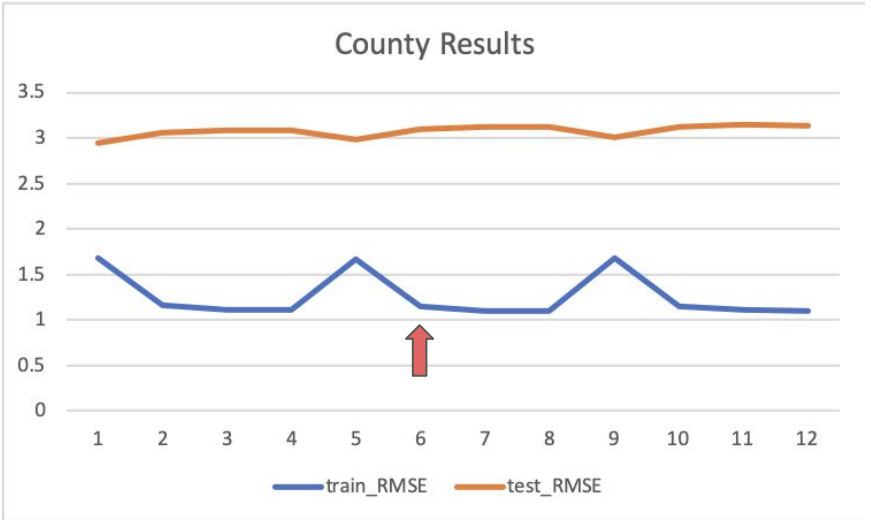
| NumTrees | MaxDepth | train_RMSE | test_RMSE |
|---|---|---|---|
| **50** | 10 | 0.62162027 | 0.92602842 |
| **50** | 20 | 0.41727384 | 0.94974451 |
| **50** | 40 | 0.41702547 | 0.95119794 |
| **50** | None | 0.41702547 | 0.95119794 |
| **100** | 10 | 0.62017223 | 0.92378084 |
| **100** | 20 | 0.41437331 | 0.94572822 |
| **100** | 40 | 0.41364818 | 0.9451869 |
| **100** | None | 0.41364818 | 0.9451869 |
| **200** | 10 | 0.62087221 | 0.91512749 |
| **200** | 20 | 0.41235177 | 0.93856964 |
| **200** | 40 | 0.41169916 | 0.93940947 |
| **200** | None | 0.41169916 | 0.93940947 |
| **300** | 10 | 0.62047211 | 0.91488519 |
| **300** | 20 | 0.41145572 | 0.93898733 |
| **300** | 40 | 0.41077391 | 0.93994872 |
| **300** | None | 0.41077391 | 0.93994872 |



# Model Generation

- County Level results

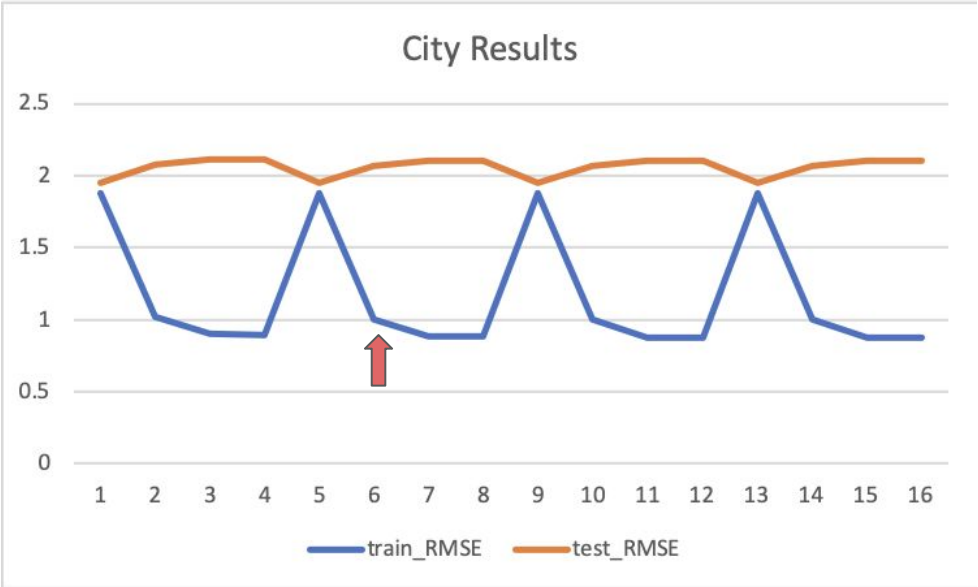
| NumTrees | MaxDepth | train_RMSE | test_RMSE |
|---|---|---|---|
| 50 | 10 | 1.67594346 | 2.94473008 |
| 50 | 20 | 1.15858898 | 3.0522253 |
| 50 | 40 | 1.10936165 | 3.08133405 |
| 50 | None | 1.11024803 | 3.08729137 |
| 100 | 10 | 1.67197548 | 2.98097406 |
| 100 | 20 | 1.14632176 | 3.09217671 |
| 100 | 40 | 1.0975603 | 3.11638795 |
| 100 | None | 1.09788871 | 3.11575141 |
| 200 | 10 | 1.6784228 | 3.01242582 |
| 200 | 20 | 1.15127639 | 3.11876398 |
| 200 | 40 | 1.10550031 | 3.14771783 |
| 200 | None | 1.10410864 | 3.13968955 |



# Model Generation

- City Level results

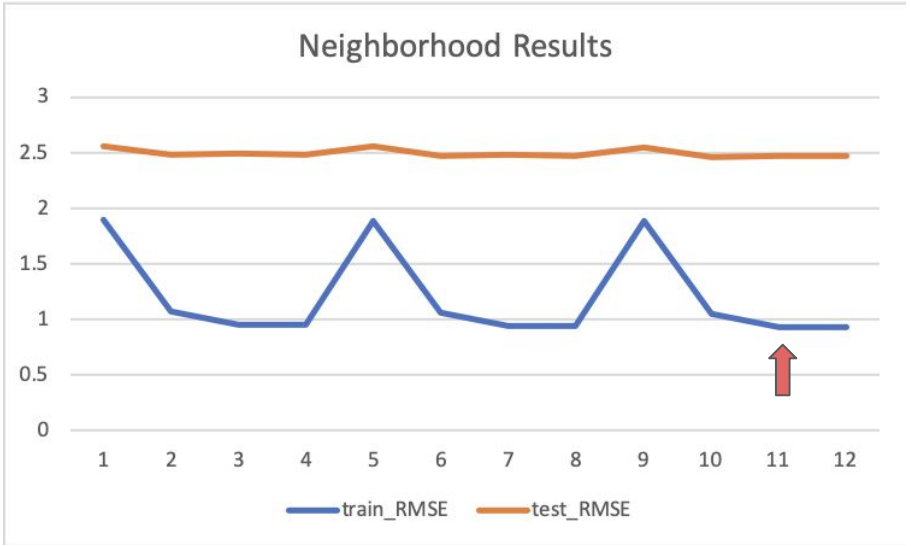
| NumTrees | MaxDepth | train_RMSE | test_RMSE |
|---|---|---|---|
| 50 | 10 | 1.87903369 | 1.94937614 |
| 50 | 20 | 1.01666371 | 2.07748822 |
| 50 | 40 | 0.89891274 | 2.10995676 |
| 50 | None | 0.89535141 | 2.11390464 |
| 100 | 10 | 1.87791521 | 1.94900443 |
| 100 | 20 | 1.00374325 | 2.06958945 |
| 100 | 40 | 0.88330688 | 2.10141736 |
| 100 | None | 0.87916972 | 2.10631649 |
| 200 | 10 | 1.87636907 | 1.94882629 |
| 200 | 20 | 0.99871646 | 2.07112504 |
| 200 | 40 | 0.87702477 | 2.10267584 |
| 200 | None | 0.87305475 | 2.1064558 |
| 300 | 10 | 1.87671412 | 1.94818481 |
| 300 | 20 | 0.99824585 | 2.06968712 |
| 300 | 40 | 0.8765403 | 2.10146241 |
| 300 | None | 0.87240651 | 2.10522948 |



# Model Generation

- Neighborhood Level results
- Inputs the past three observations of the given neighborhood + state,county and city level predictions

| NumTrees | MaxDepth | train_RMSE | test_RMSE |
|---|---|---|---|
| 50 | 10 | 1.89263277 | 2.55196371 |
| 50 | 20 | 1.06672797 | 2.47624789 |
| 50 | 40 | 0.94860363 | 2.4875991 |
| 50 | None | 0.94813228 | 2.48276219 |
| 100 | 10 | 1.88820993 | 2.55295955 |
| 100 | 20 | 1.0615877 | 2.46440168 |
| 100 | 40 | 0.9453708 | 2.4764172 |
| 100 | None | 0.94307551 | 2.47454546 |
| 200 | 10 | 1.88272521 | 2.54445563 |
| 200 | 20 | 1.05356985 | 2.46251338 |
| 200 | 40 | 0.9343783 | 2.46950086 |
| 200 | None | 0.93113509 | 2.46679908 |



# Designed Test Plan

- Draw a random sample of size n = 1000
- Use the model to predict the values
- Find the mean squared error of the predictions

$$\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

* $n$ is the number of data points
* $Y_i$ represents observed values
* $\hat{Y}_i$ represents predicted values

- If RMSE is greater than $3.0, the test is failed
  - Back to the drawing board!
- Else, the model has passed the test

Figure from: https://study.com

# New Predictions for Future Values

- Need to have the most recent values at all the levels for the past 3 months
  - How do we get that?

Import pandas as pd

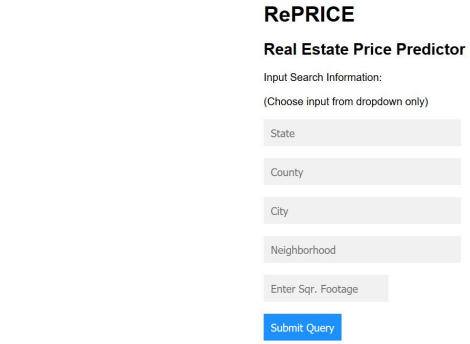
pd.read_csv(usl, header = 0, usecols = cols_month[-3:])

# UI/Server communication

- HTML and CSS
  - Used to make Dynamic web pages
  - Communicates with server by sending form data using the POST method
- Flask
  - Python micro web framework
  - Used to set up routing and serving of HTML and CSS pages

# UI/Server communication

- How it works:
  - User inputs form data on the ‘homepage’
  - Flask server receives data as an Immutable list
  - Data is separated and stored in distinct variables

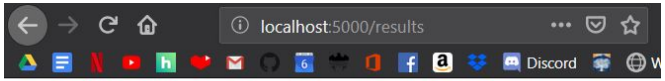



```
ImmutableMultiDict([('myState', ''), ('myCounty', ''), ('myCity', ''),
```

```
state = request.form.getlist('myState')
county = request.form.getlist('myCounty')
city = request.form.getlist('myCity')
neighborhood = request.form.getlist('myNeighborhood')
```

# UI/Server communication

```
return render_template("results.html", result = result)
```




**The final predicted value is "result"**

- How it works cont'd:
  - Variables are used by the predictor model to do calculation
  - Predictor result is stored in result variable and sent back to the front end as a value.
  - The values is displayed by the 'results' page