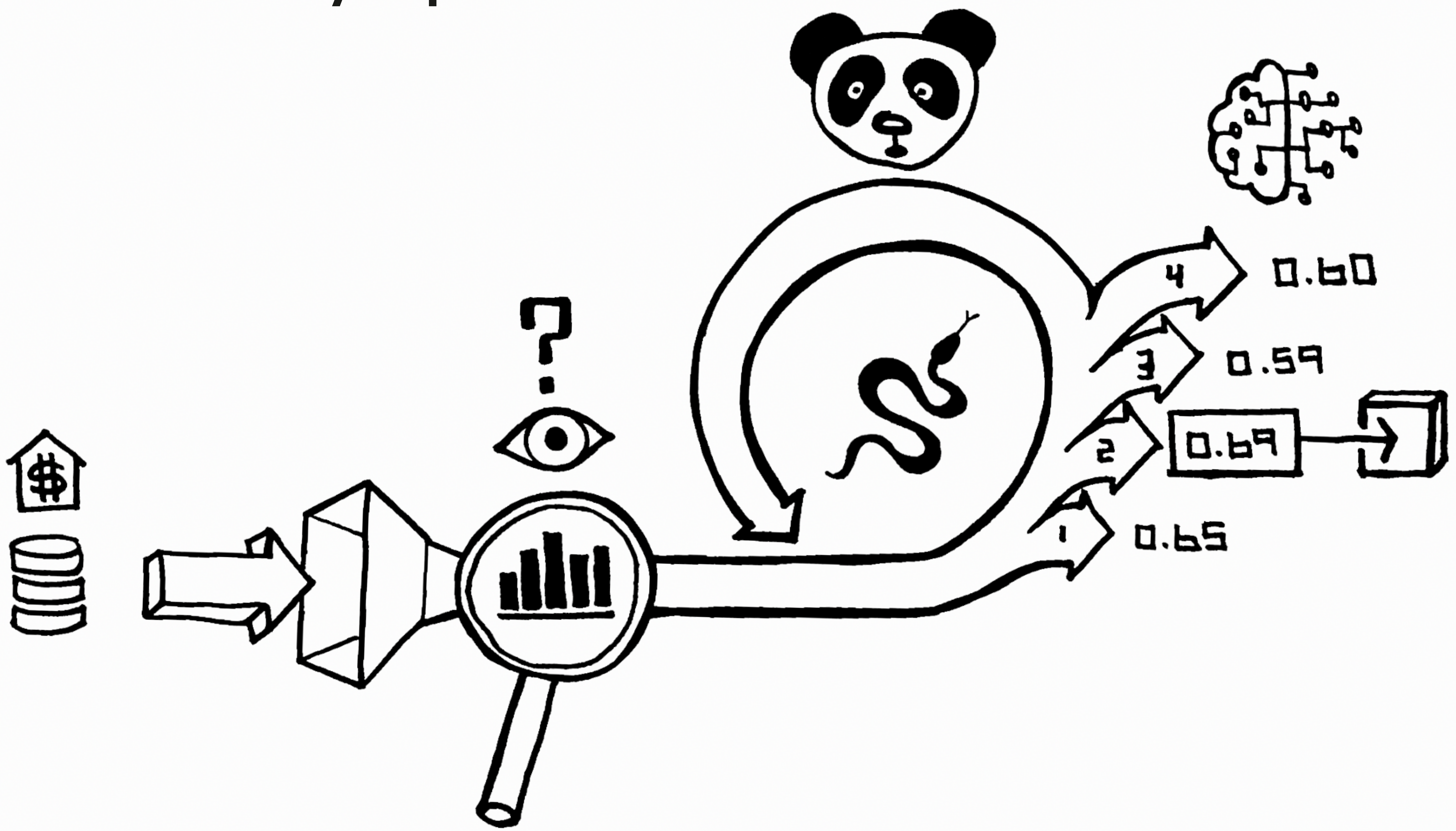
Housing Regret-ssion

How to not properly predict house pricing

BY DEEP-SOUTH

First Steps

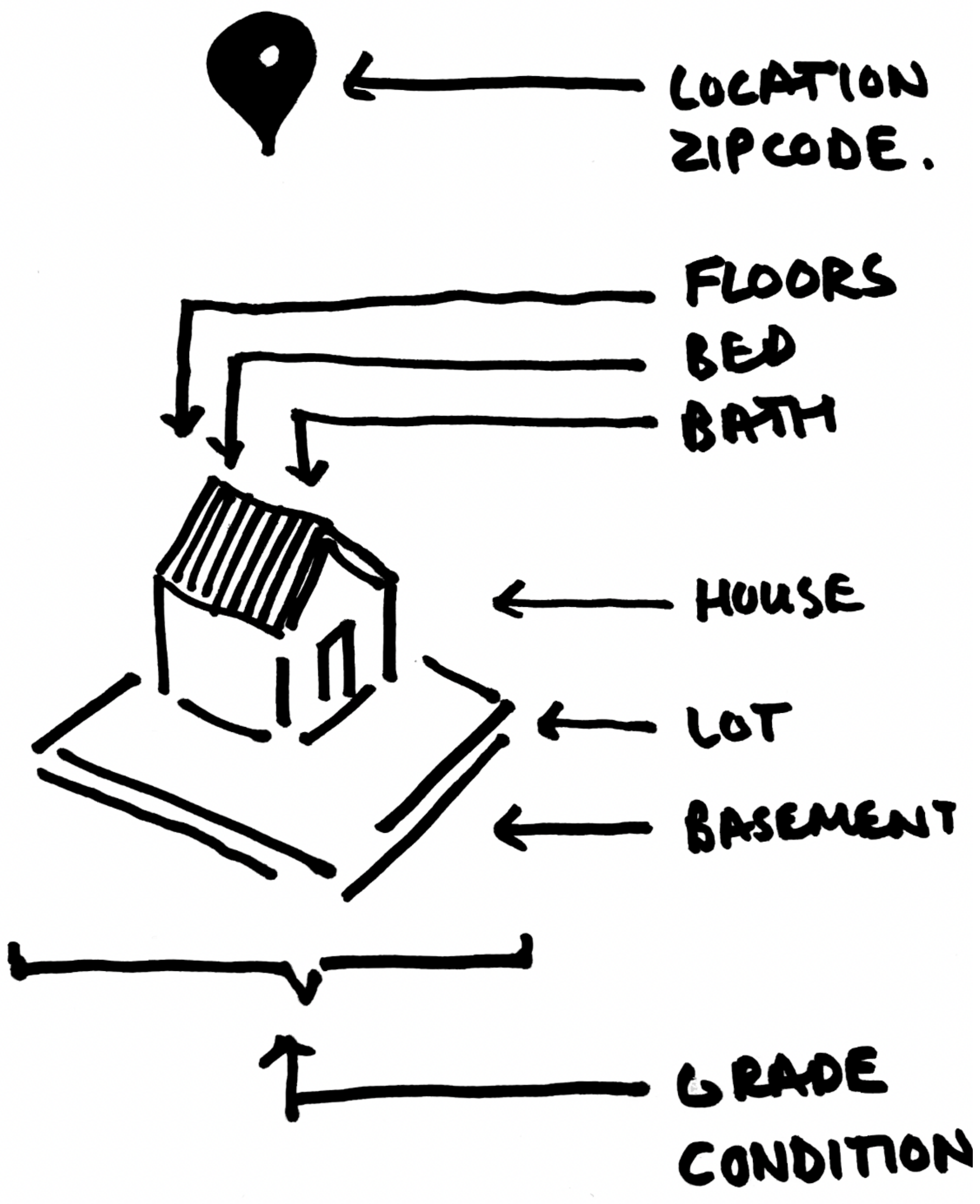
We followed the data analysis process



Data in — Visualize, Clean, prepare — Train model & test data (& repeat) — Results

The Data

Our visual analysis



Original Data

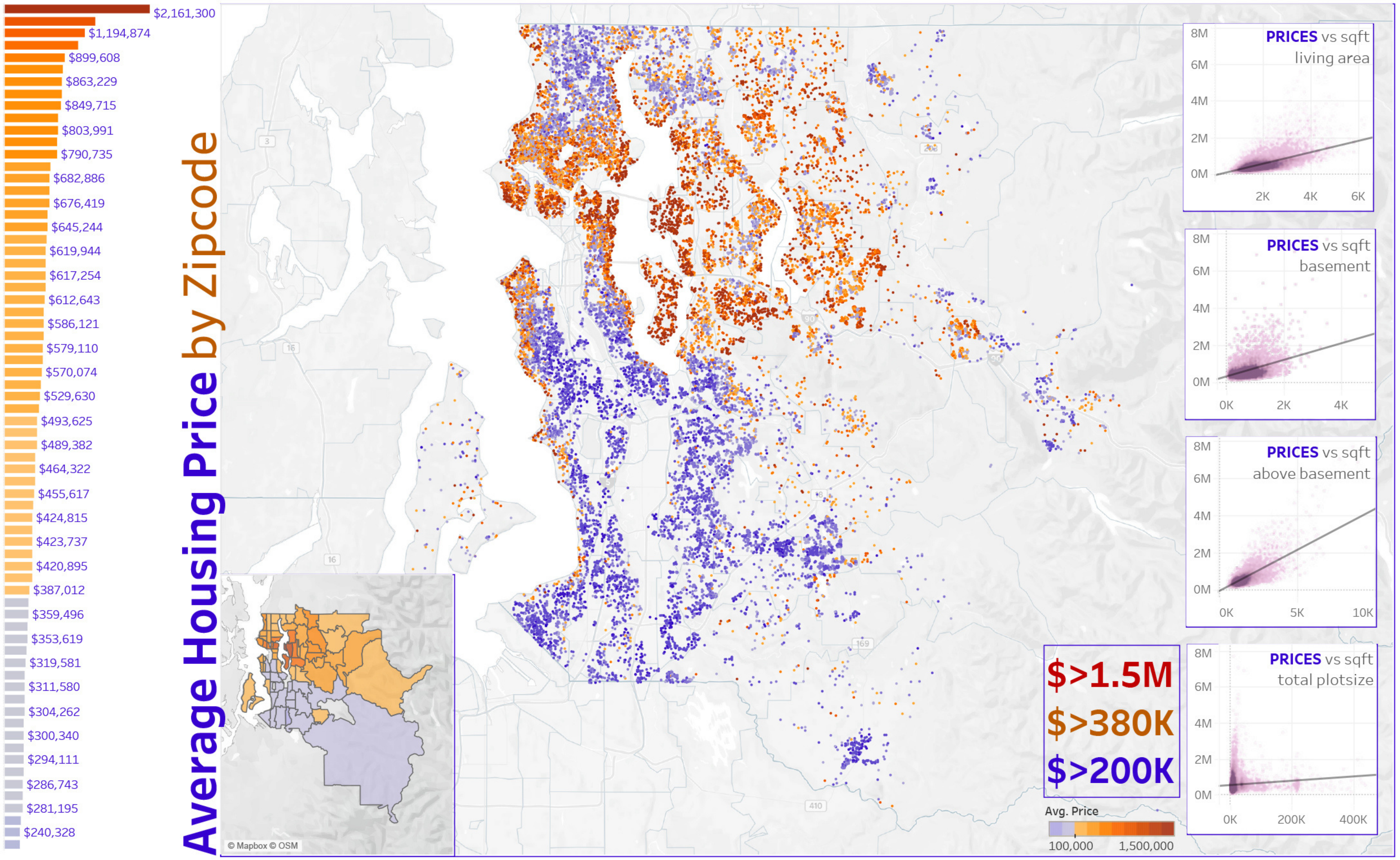
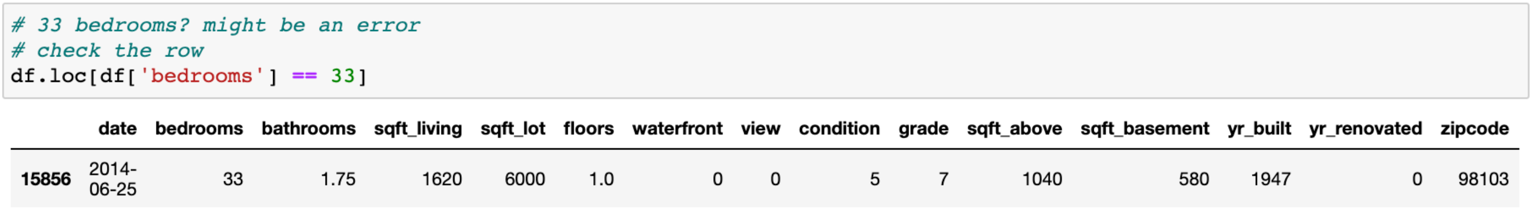


Tableau Analysis

The Process

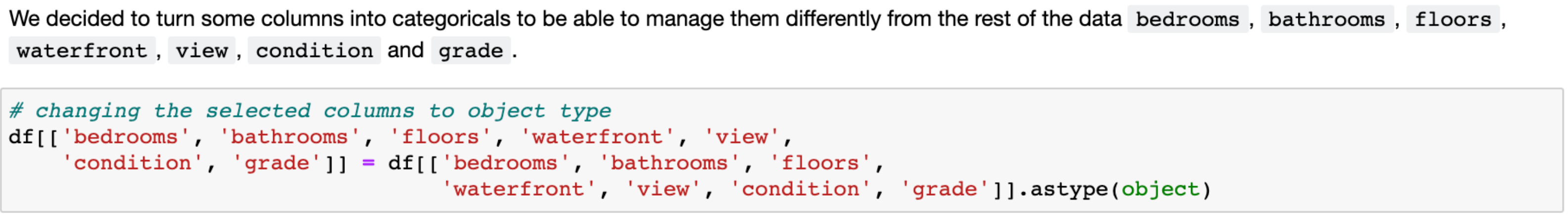
We cleaned the data

- Looking for outliers and 'weird' numbers that could potentially affect our predictions



```
# 33 bedrooms? might be an error
# check the row
df.loc[df['bedrooms'] == 33]
```

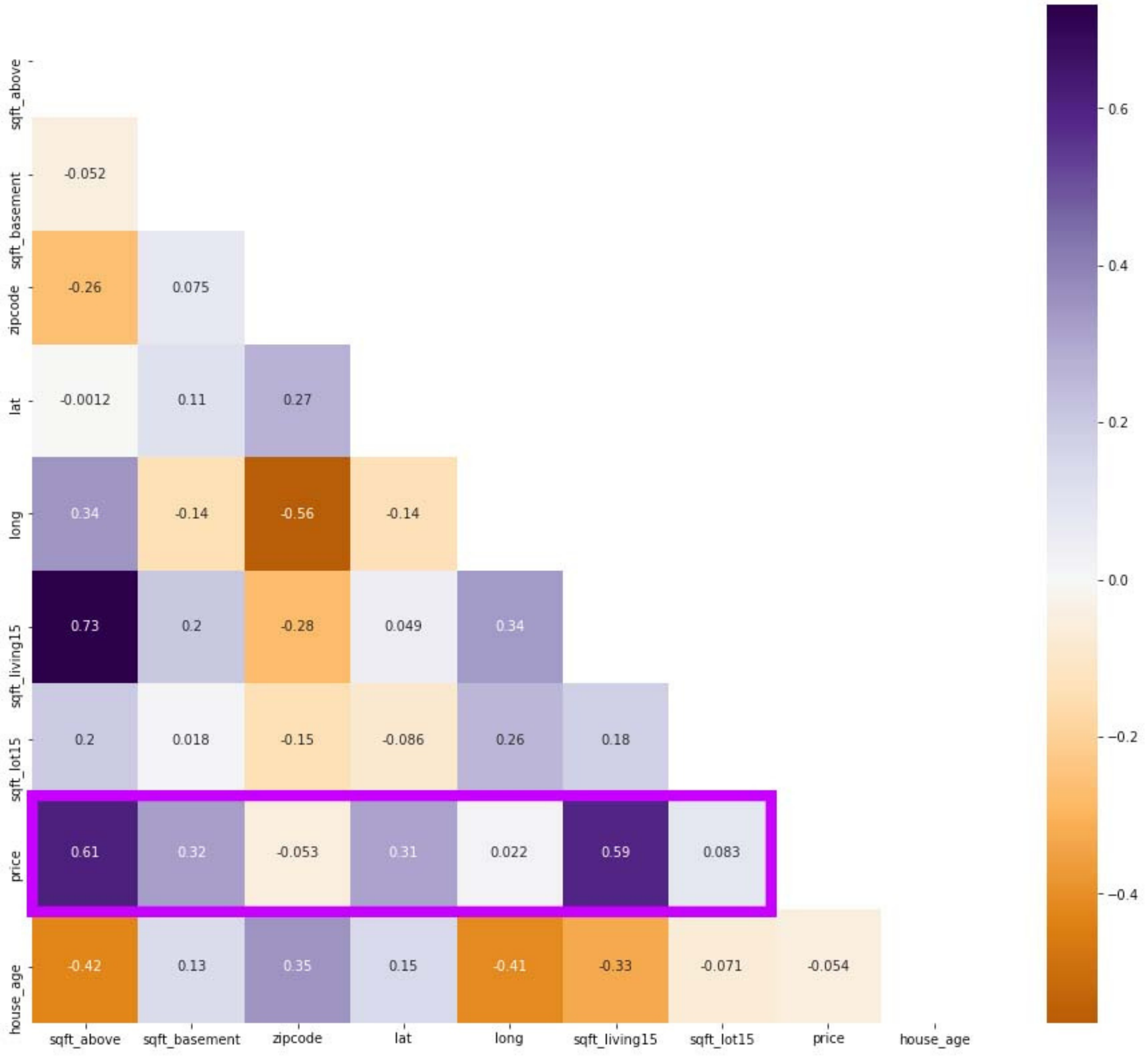
	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode
15856	2014-06-25	33	1.75	1620	6000	1.0	0	0	5	7	1040	580	1947	0	98103



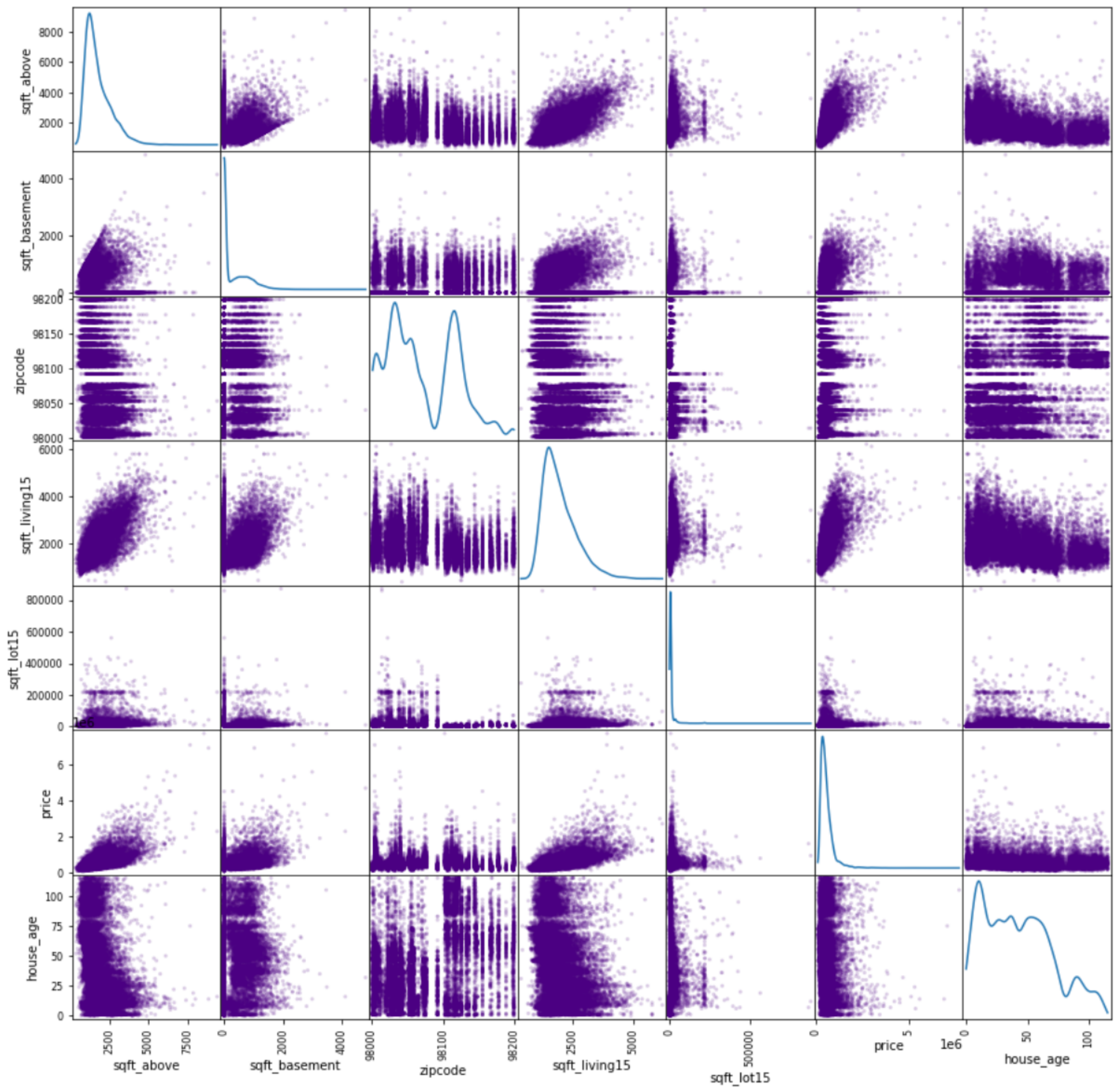
We decided to turn some columns into categoricals to be able to manage them differently from the rest of the data `bedrooms`, `bathrooms`, `floors`, `waterfront`, `view`, `condition` and `grade`.

```
# changing the selected columns to object type
df[['bedrooms', 'bathrooms', 'floors', 'waterfront', 'view',
    'condition', 'grade']] = df[['bedrooms', 'bathrooms', 'floors',
                                 'waterfront', 'view', 'condition', 'grade']].astype(object)
```

- Checking for correlations/data distribution



	sqft_above	sqft_basement	zipcode	lat	long	sqft_living15	sqft_lot15	price	house_age
sqft_above									
sqft_basement	-0.052								
zipcode	-0.26	0.075							
lat	-0.0012	0.11	0.27						
long	0.34	-0.14	-0.56	-0.14					
sqft_living15	0.73	0.2	-0.28	0.049	0.34				
sqft_lot15	0.2	0.018	-0.15	-0.086	0.26	0.18			
price	0.61	0.32	-0.053	0.31	0.022	0.59	0.083		
house_age	-0.42	0.13	0.35	0.15	-0.41	-0.33	-0.071	-0.054	



The Models

We tried different models with different approaches

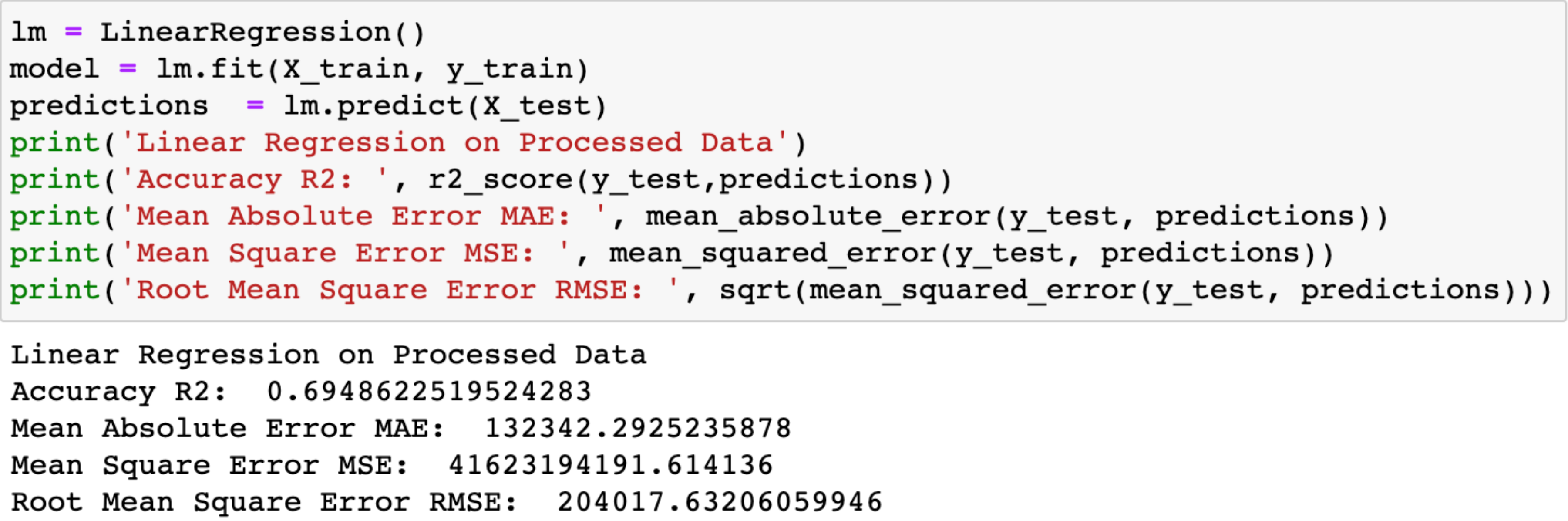
- Linear regression, KNN, Decision Tree & Random Forest (the last 2 just to experiment)
- 4 Iterations: 1.- Raw data
  2.- StandardScaler on numerical columns & GetDummies in the categoricals
  3.- Removing all the outliers
  4.- Using only a few obvious columns instead of all of them (floors, bedrooms, condition, sqft_lot15 & price)

	1st	2nd	3rd	4th
LR	65%	69%	59%	60%
KNN	54%	65%	61%	53%
DT	63%	56%	51%	29%
RF	80%	78%	77%	56%

The Results

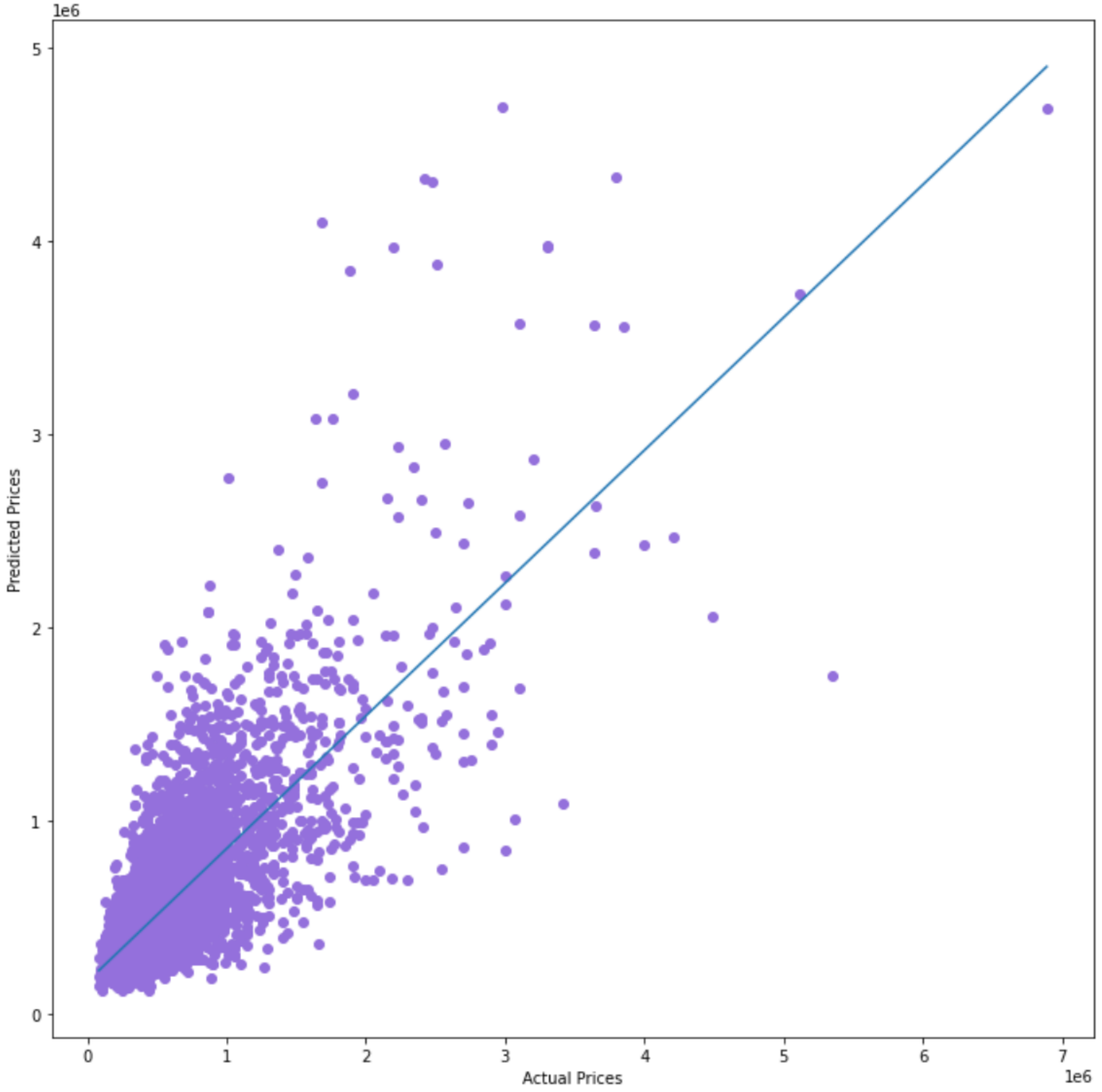
We got a price prediction accuracy of 69% and a mean absolute error of $132,342

- Removing some unnecessary columns (according to our criteria)
- Using Linear Regression
- Normalizing the categorical data with GetDummies
- Scaling the numerical data with StandardScaler



```
lm = LinearRegression()
model = lm.fit(X_train, y_train)
predictions  = lm.predict(X_test)
print('Linear Regression on Processed Data')
print('Accuracy R2: ', r2_score(y_test,predictions))
print('Mean Absolute Error MAE: ', mean_absolute_error(y_test, predictions))
print('Mean Square Error MSE: ', mean_squared_error(y_test, predictions))
print('Root Mean Square Error RMSE: ', sqrt(mean_squared_error(y_test, predictions)))
```

```
Linear Regression on Processed Data
Accuracy R2:  0.6948622519524283
Mean Absolute Error MAE:  132342.2925235878
Mean Square Error MSE:  41623194191.614136
Root Mean Square Error RMSE:  204017.63206059946
```



The End?

We now know we shouldn't take data for granted

- Our next possible steps include reusing columns we drop at the beginning just because we saw that they wouldn't be necessary
- Also learning more about decision trees and random forest to be able to find a better fit for a dataset like this one