Building a Predictive model

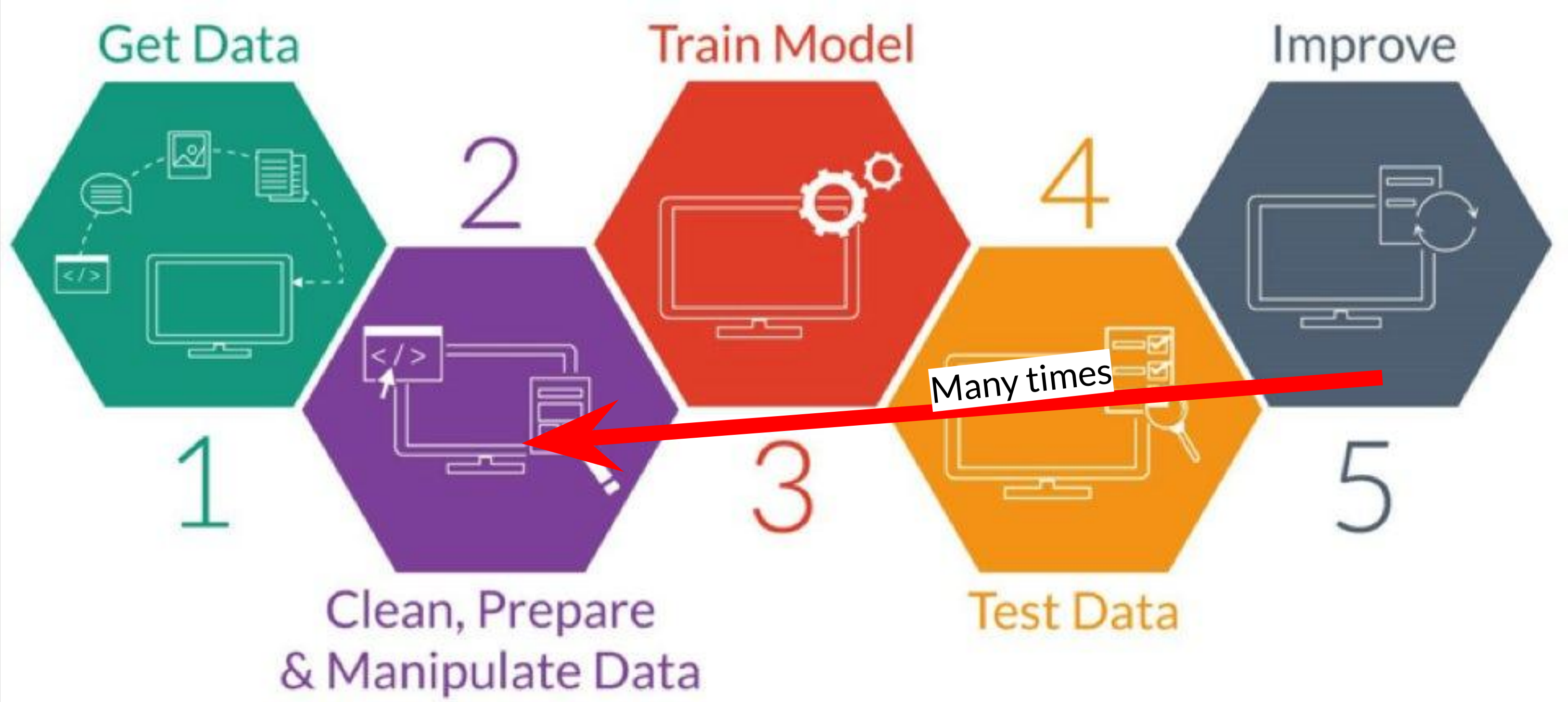
Thomaz Moon

Problem Statement:

A Real estate investment group hired me to teach their analyst (Bob), who is familiar with python, how to make a predictive model.

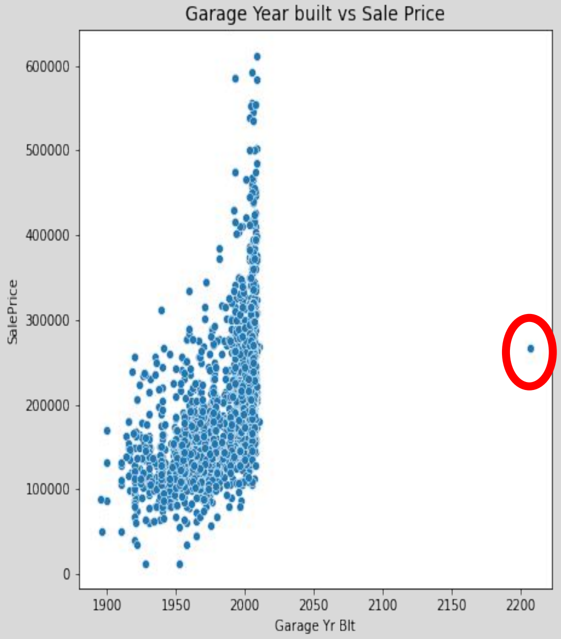
*The more technical things will be in the notebooks, while this presentation is an overview of what we did.

Data Process flow used



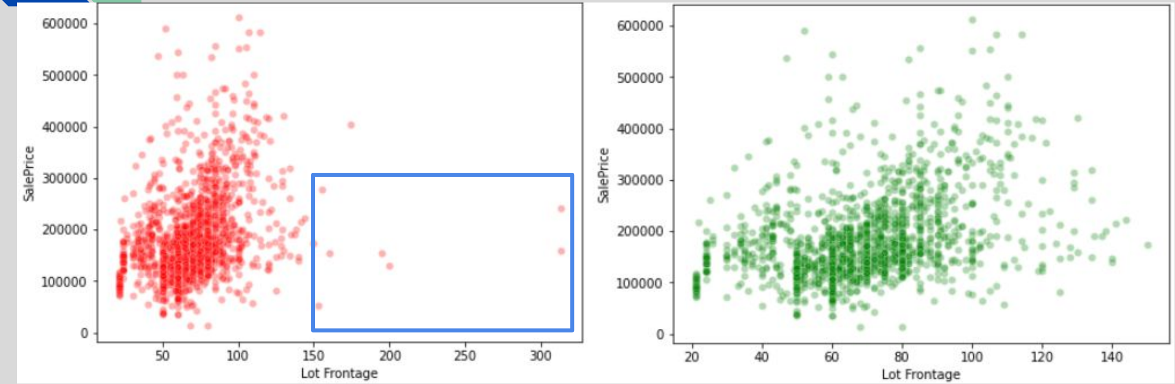
Process of building the models

1. Clean the data
    a. Missing values
    b. Nonsense values
    c. Look for outliers
2. Check for correlations to see where I might want to start
    a. Using .corr()
    b. Heatmaps
3. Make a model and test it
4. Repeat



Cleaned vs uncleaned scatter plots

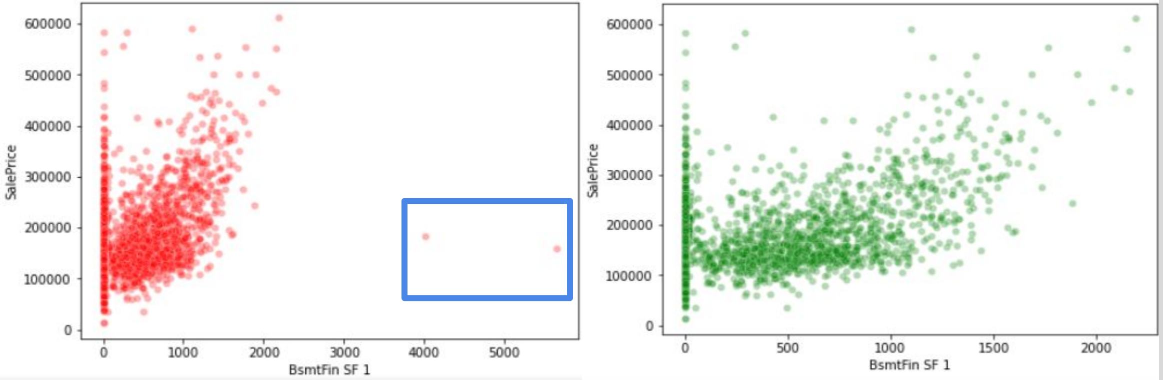
Lot frontage vs Sale Price



Cleaned Data

Basement square foot 1 vs Sale Price

Raw data



Looking at correlations to start making a model

Gets the job done

SalePrice	1.000000
Overall Qual	0.799607
Gr Liv Area	0.731128
Garage Cars	0.679439
Total Bsmt SF	0.672876
Garage Area	0.663448
1st Flr SF	0.659165
Year Built	0.579510
Full Bath	0.560931
TotRms AbvGrd	0.548539
Garage Yr Blt	0.546064
Year Remod/Add	0.545951
Fireplaces	0.446627
BsmtFin SF 1	0.430052
Lot Area	0.367481
Open Porch SF	0.331948
Wood Deck SF	0.307945
Bsmt Full Bath	0.272205
Half Bath	0.267194
2nd Flr SF	0.223481
Bsmt Unf SF	0.181009
Bedroom AbvGr	0.128116
Mo Sold	0.020295
Yr Sold	-0.000367
BsmtFin SF 2	-0.028894
Bsmt Half Bath	-0.066242
Kitchen AbvGr	-0.085796
Enclosed Porch	-0.136347
Overall Cond	-0.171375

Nice DataFrame style

	SalePrice
SalePrice	1.000000
Overall Qual	0.799607
Gr Liv Area	0.731128
Garage Cars	0.679439
Total Bsmt SF	0.672876
Garage Area	0.663448
1st Flr SF	0.659165
Year Built	0.579510
Full Bath	0.560931
TotRms AbvGrd	0.548539
Year Remod/Add	0.545951
Fireplaces	0.446627
BsmtFin SF 1	0.430052

Beautiful Heatmap

Running tests on your model

```python
# This function will just return a Ridge score for X along with the RMSE for easier use
def ridge_it(X):
    X_train, X_test, y_train, y_test = train_test_split(X, y)

    # scale it
    sc = StandardScaler()
    Z_train = sc.fit_transform(X_train)
    Z_test = sc.transform(X_test)

    r = RidgeCV(alphas = np.logspace(0,3, 100), cv = 6, scoring = 'r2')
    r.fit(Z_train, y_train)

    print(f'Train Score: {r.score(Z_train, y_train)}')
    print(f'Test Score: {r.score(Z_test, y_test)}')

    print(f'\n Train RMSE: {metrics.mean_squared_error(y_train, r.predict(Z_train), squared= False)}')
    print(f'Train RMSE: {metrics.mean_squared_error(y_test, r.predict(Z_test), squared=False)}')
```

```python
# Lets just test it first using our X_poly_df
ridge_it(X_poly_df)
```

```
Train Score: 0.9287186789108417
Test Score: 0.9198447095999542

 Train RMSE: 19844.717052914963
Train RMSE: 22613.619236475166
```

- Test your model a few times using different methods.
  - Linear
  - Ridge
  - Lasso

- Checking for bias/variance as well as if you're overfitting your model.
  - Do this while you have the target variable

After the Tests

- Check your parameter's information.
  - Statsmodel
  - Correlation
- Compare training test scores with your real score.
  - Check for bias/variance on the real test
  - Overfit/Underfit

Dep. Variable:	SalePrice	R-squared:	0.947
Model:	OLS	Adj. R-squared:	0.934
Method:	Least Squares	F-statistic:	75.03
Date:	Sun, 26 Sep 2021	Prob (F-statistic):	0.00
Time:	12:29:13	Log-Likelihood:	-20126.
No. Observations:	1799	AIC:	4.094e+04
Df Residuals:	1453	BIC:	4.285e+04
Df Model:	345		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Coef	-1.605e+09	9.99e+08	-1.608	0.108	-3.56e+09	3.53e+08

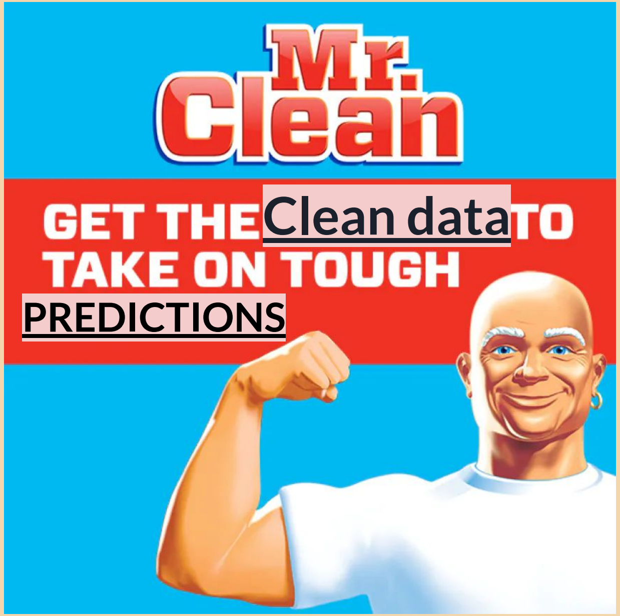
Now you just

Repeat.
Repeat.
Repeat.

Recommendation 1: Clean your data

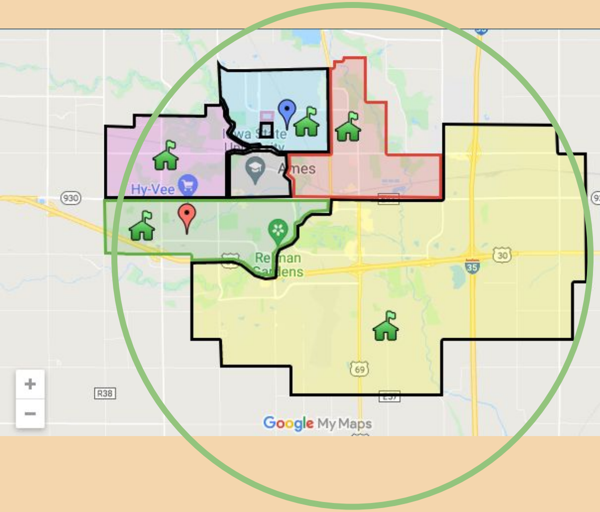
- Clean Clean Clean
- Model
- Clean

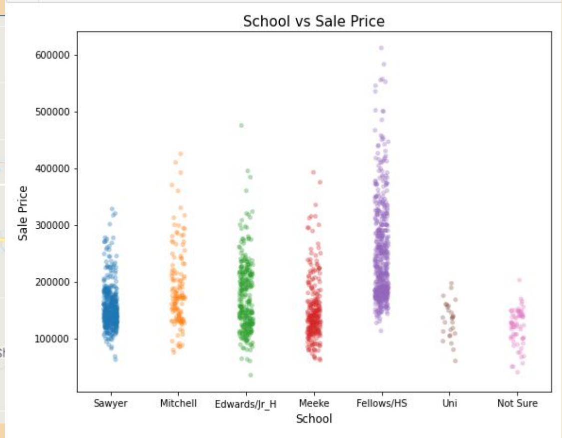
*But make sure what you're cleaning is even going to be used so you don't waste time

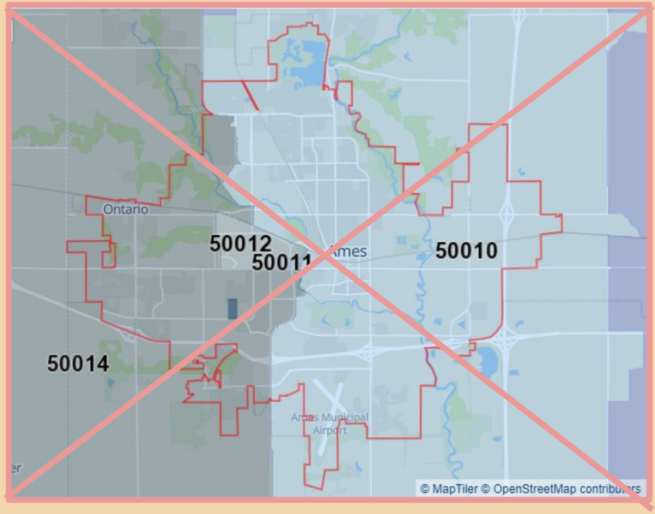


Recommendation 2: Don't limit yourself

- Don't be afraid to look for outside Data if you think it might help your model

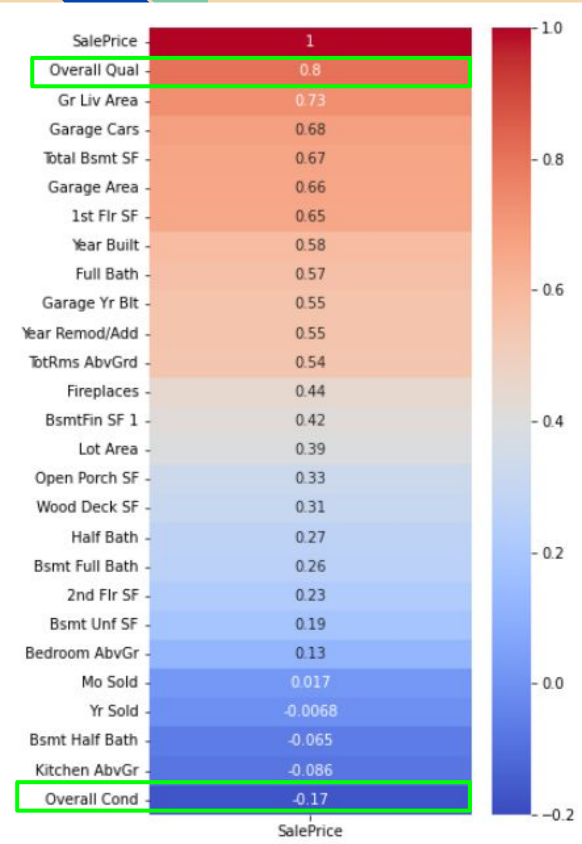




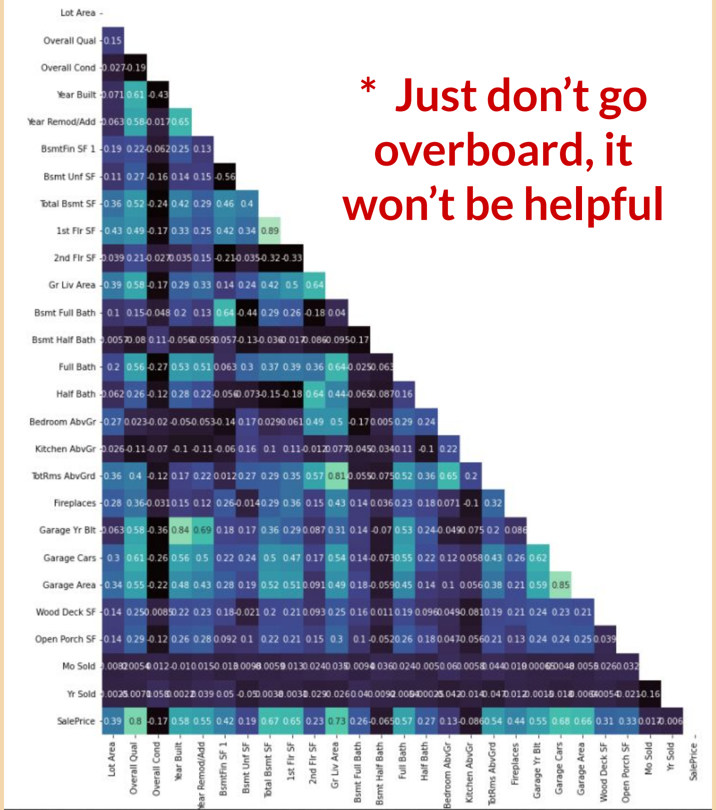


*But make sure it will actually help you

Recommendation 3: Visuals



Use Visuals every now and then. You might see some data you would have otherwise looked over



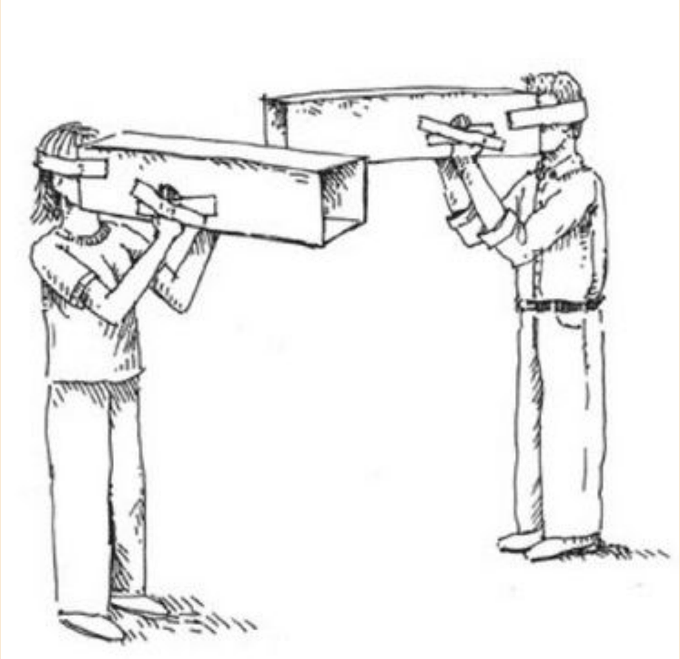
* Just don't go overboard, it won't be helpful

<u>Recommendation 4:</u> Don’t tunnel vision

Don’t focus too much on only one metric. R2 scores for example never go down, even if the features don’t actually help your model.

R-squared:	0.952
Adj. R-squared:	0.940
RMSE Score:	23,375.032

R-squared:	0.947
Adj. R-squared:	0.934
RMSE Score:	22,417.390



Recommendation 5: Prep for missing data

- Try to contact whoever you need to in order to verify or retrieve the data.
- If it's just a few rows, you might want to consider dropping the rows
- Not use the columns that have a lot of missing data.
- Make a model to predict the missing values

```
print(train.shape)
print(test.shape)
```

```
(2051, 81)
```

```
# if we drop Na
train.dropna().shape
```

```
(1508, 60)
```

*Sometimes missing data can be the key to your model if you find it though

Sources

- *Ames Data Set:*
  - *https://www.kaggle.com/c/dsir-830-project-2-regression-challenge/*
- *Cover picture:*
  - https://www.cityofames.org/home/showpublishedimage/6334/635943415687730000
- Data Flow chart:
  - https://econsultancy.imgix.net/content/uploads/2018/10/15142456/upxacademy-flowchart-data-science.png?auto=compress,enhance,format,redeye&crop=faces,entropy,edges&fit=crop&q=60&w=960&h=431
- **Repeat Picture:**
  - **http://blog.vipkid.com/wp-content/uploads/2019/08/Repeat-Blog-Image-2.png**
- **Mr Clean:**
  - **https://contentgrid.thdstatic.com/hdus/en_US/DTCCOMNEW/fetch/FetchRules/Rich_Content/203253133-3700083906-mr-clean-magic-erasers-outdoor-pro-multi-purpos-take-on-tough-messes-7-2020-v1.jpg**
- **Tunnel Vision:**
  - **https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.samatters.com%2Funderstanding-stress-part-5-tunnel-vision%2F&psig=AOvVaw3OUcnbgB23COOiv6s_vwvc&ust=1632793624255000&source=images&cd=vfe&ved=0CAsQjRxqFwoTCNCs37eEnvMCFQAAAAAdAAAAABAD**
- **Missing Data:**
  - **https://www.dataapplab.com/wp-content/uploads/2017/04/missing-data.jpg**