Iowa Real Estate — Where We Always “Ames” to Please

Housing predictions by Dodge McIntosh

- Located in the central part of Story County, Iowa, the city of Ames is approximately 30 miles north of Des Moines.
- In 2016, Ames had a population of 66,191.
- *Fun fact* In 2010, Ames was ranked **9th** on CNNMoney's "Best Places to Live" list *(Awesome! The models I built are already accurate enough to land me a job at CNN!)*

Why are we interested in Ames?

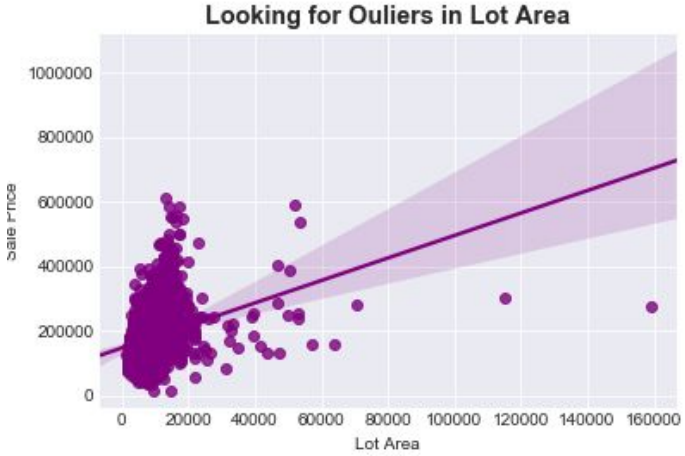
- The Ames Housing Dataset is an exceptionally detailed and robust dataset with over 70 columns of different features relating to over 2000 houses.
- We used that dataset to create two models with the highest possible accuracy. These models predicted the following:
    - The price of a house at sale (**regression**)
    - Whether a house sale was abnormal or not (**classification**)

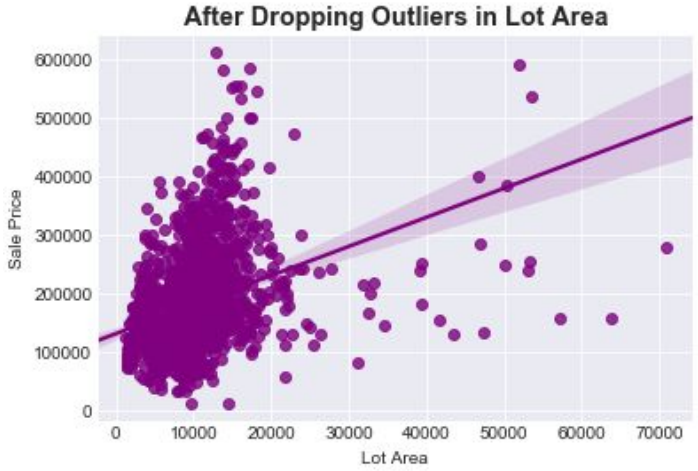
Exploring the Data

- In order to avoid overfitting, I want to keep it as basic and logical as possible.
- This means focusing on:
  - **Land** (lot_area)
  - Building **square footage** total (created feature)
  - **Location** (neighborhoods and zoning)
  - Combined measures of **quality** (created features)
  - And *maybe* a couple other ones up my sleeve.

Picking on Outliers

Outliers can adversely affect your model so I decided to drop them.





I did this with several other features as well. #SorryNotSorryOutliers

Into the Frying NaN

- Any features that we want to feed into our model need to have entirely non-null values.
- There are basically two options for dealing with them:
  a. **Dropping** them
     - Entire **columns** if enough values are missing
     - **Rows** that have that specific feature missing
  b. **Imputing** them
     - Filling them with value at your discretion

Imputer? I Hardly Know Her!

A lot of times it might not even be worth it to impute, but I thought this was a neat way of imputing missing lot_frontage values.

In [51]:
```
# Imputing missing lot_frontage values with the average of one in the same neighborhood.

neigh_frontage_means = train.groupby('neighborhood')['lot_frontage'].mean().to_dict()

train['lot_frontage'] = train.apply(lambda x: neigh_frontage_means[x.neighborhood] if \
                                    np.isnan(x.lot_frontage) else x.lot_frontage, axis=1)
```

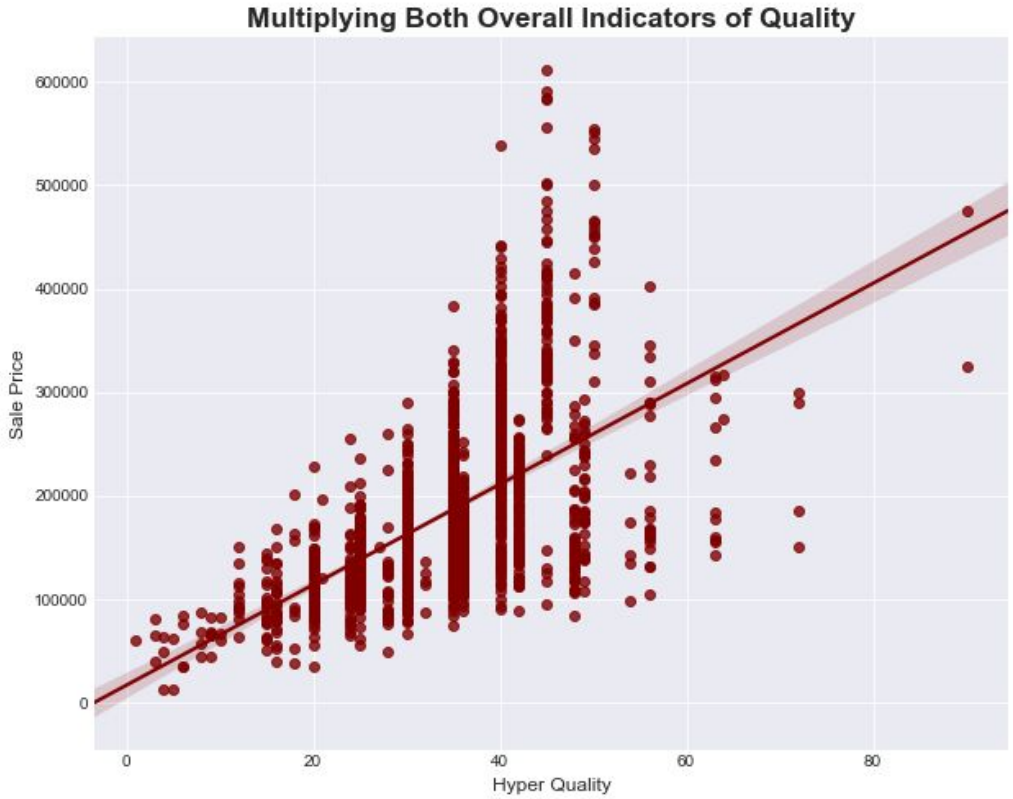
YUGE shoutout to J for helping make this one possible. :raised-hands:

Feature Engineering

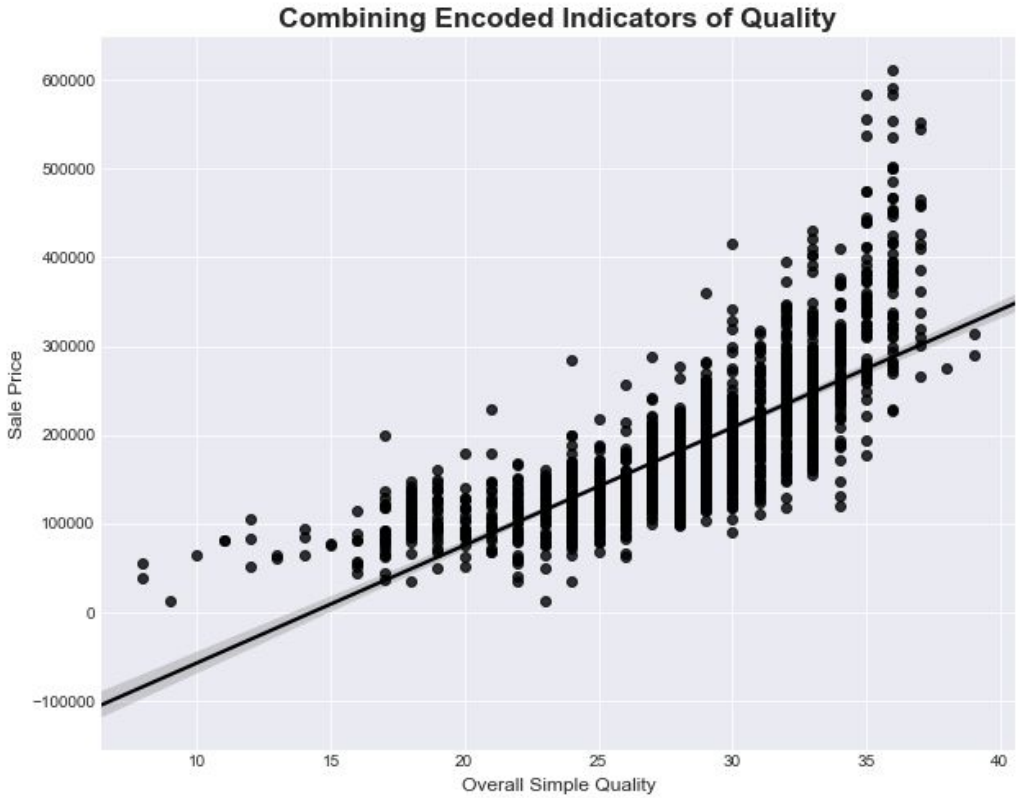
- Some features (like ones that have text values) need to be manipulated before they can be used in our model.
- Hidden relationships and correlations can be revealed by combining others.

---

1. **Overall Simple Quality //** The sum of 10 different quality features that I numerically encoded first.
2. **Hyper Overall Quality //** The product of both “overall” features.
3. **Building Area //** The sum of the above ground liveable area, the total basement sq. ft., and the garage area in sq. ft.
4. **New House //** If the year built was equal to the year sold
5. **Remodeled //** If the year built was not equal to the year remod/add.



Hmm, it does seem kind of correlated, but it gets somewhat weaker as the x increases...



Now there is a definite pattern here but it doesn't look like it's meant for straight linear regression...

There we go! Finally engineered a halfway decent feature that looks to be very strongly correlated with sale price.

Time. to. Model.

Model Details

- My final **features** were the ones I engineered, lot area, lot frontage, dummied neighborhood ones, and dummied zoning ones.
- I ended up using a Lasso model and RandomizedSearch to tune my hyperparameters (gotta give this old guy all the rest it can get!)
    - My **rs.best_score_** from that was **0.855576972493**
    - That was done with an optimal Lasso **alpha** of **0.136**

Generalize It!

- While we want our model to do well in captivity, what we really care about is what happens when we release it into the wild.
- We want it to generalize well — being able to predict accurately on new information instead of being overfit to the training data.

20	▲5	Dodge McIntosh		27742.233...	26	2d

Initially, I was scoring in the 40,000 - 50,000 range, but when the rest of the predictions were scored (70%), I was able to break into the 20,000's.
This tells me that the features I was focusing on and the model I submitted were actually relatively good at predicting the real values.


USTED LEADERSHIP
for a
RONGER AME CA
JEB2016.COM
abcNEWS
.com