2022 AI534-IA1 competition

Mahdis Safari

I added two features “total_sqft” and “view_waterfront” which you can see their formula in the following code. For total_sqft it’s obvious that total square-feet for a house is the summation of square-feet for basement, interior living space, and the interior housing space that is above ground level. As all these features were left skewed so total_sqft should be also left skewed and that’s why I used cube transformation.

The next feature I added is, I observed that waterfront and view both should have similar effect on the price but waterfront’s effect is much more so I decided to test different values for m in m*waterfront+view and the best value I got was 7.

```
data['total_sqft'] = (data['sqft_living'] + data['sqft_above'] + data['sqft_basement'])**3
data['view_waterfront'] = 7 * data['waterfront'] + data['view']

data = data.drop(["sqft_living"], axis=1)
data = data.drop(["sqft_above"], axis=1)
data = data.drop(["sqft_basement"], axis=1)
```

Then I transformed some other features based on the histogram of the features on kaggle in data section. For those features which histogram was right skewed, I used log transformation and for left skewed I chose cube transformation.

```
#right skewed
data["lat"] = np.log(data["lat"])
data["year"] = np.log(data["year"])
#left skewed
data["bathrooms"] = data["bathrooms"] ** 3
data["view"] = data["view"] ** 3
data["condition"] = data["condition"] ** 3
data["grade"] = data["grade"] ** 3
data["long"] = data["long"] ** 3
```

I tried transformation for almost all of the features regarding how their histogram were, and I kept transformed features that lead to improvement in the MSE.

Then I printed a list of sorted absolute value of weights for features. Then I start dropping features from bottom of that list as long as it it was helpful.

```
for x in sorted(zip(preprocessed_val_data.columns.values, abs(weights)), key=lambda x: x[1],
reverse=True):
    print(x)
```

('grade', 1.3292908107556836)
('total_sqft', 0.8709784815279109)
('lat', 0.8302006314852581)
('age_since_renovated', 0.7946004805530177)
('bathrooms', 0.5151798383411967)
('sqft_living15', 0.43796331487445533)
('yr_renovated', 0.4320177330385808)
('view_waterfront', 0.4123352746384914)
('waterfront', 0.3584387830735865)
('condition', 0.27703608824491877)
('zipcode', 0.2623071416529306)
('year', 0.25284089616658245)
('long', 0.24098695209391932)
('view', 0.21371459878790217)
('floors', 0.18205661723874175)
('yr_built', 0.18175554032423075)
('month', 0.1423781400661072)
('sqft_lot15', 0.10249128429214593)
('sqft_lot', 0.03912694449460258)
('day', 0.0286049418835483)
('bedrooms', 0.0024428187309235485)

So I dropped the following attributes.

```
data = data.drop(["bedrooms"], axis=1)
data = data.drop(["day"], axis=1)
data = data.drop(["sqft_lot"], axis=1)
data = data.drop(["yr_built"], axis=1)
data = data.drop(["sqft_lot15"], axis=1)
data = data.drop(["month"], axis=1)
```

And finally for better performance I decided to shuffle the training dataset before training.

I learned that shuffled dataset helps in decreasing the MSE. Moreover, in feature engineering, we can find a feature as a function of some other features that have bigger weights. For finding these features we should try constructing new features using the knowledge of the problem's context and features.
For features with skewed histogram, we can use proper transformation and then we observe remarkable difference in the weights. Finally, removing features with very small weights can improve our model.