

Basic feature engineering

Improving, synthesizing, and injecting new features

Terence Parr

MSDS program

University of San Francisco

Overview

- Huge topic and, after basic cleaning, this is where you'll spend the most time
- Good features are much more important than the model, assuming you pick good one like RF or gradient boosting

“At the end of the day, some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used.” -- Pedro Domingos
From “a few useful things to Know about machine Learning”

Deriving numeric columns

- Goal is to help model: either smaller trees or more accurate or both
- Rent: Longer feature list, description, num photos could be predictive

```
df["num_desc_words"] = df["description"].apply(lambda x: len(x.split()))  
df["num_features"]   = df["features"].apply(lambda x: len(x.split(",")))  
df["num_photos"]     = df["photos"].apply(lambda x: len(x.split(",")))
```

- Create *interaction terms*; ever have to wait for siblings to take a shower? Maybe there is some predictive power in the ratio of bedrooms to bathrooms; maybe beds+baths?

```
df["beds_to_baths"] = df["bedrooms"]/(df["bathrooms"]+1) # avoid div by 0
```

More numeric feature ideas

- Loan or credit card application classifier example; count previous credit card attempts or average previous loan amount (or could derive boolean “has applied previously”)
- For detecting ATM fraud, create column for average previous withdrawal amount; min or max withdrawal could be useful too
- *Rank encoding*: convert raw numbers to their rank. It could be the order is more important than value, which could distract the model. It also squashes outliers

Synthesizing new vars from strings

- Before label encoding, try to derive features from string features
- E.g., Apt data: count words in description or number of features or derive column indicating apt has a doorman, garage, ...

description	features
Top Top West Village location, beautiful Pre-w...	[Laundry In Building, Dishwasher, Hardwood Flo...
Building Amenities - Garage - Garden - fitness...	[Hardwood Floors, No Fee]

Simple string computations

- First normalize

```
df['description'] = df['description'].fillna('')  
df['description'] = df['description'].str.lower()  
df['features'] = df['features'].fillna('')  
df['features'] = df['features'].str.lower()
```

Replace missing with blank

- Then, identify key words or subphrases

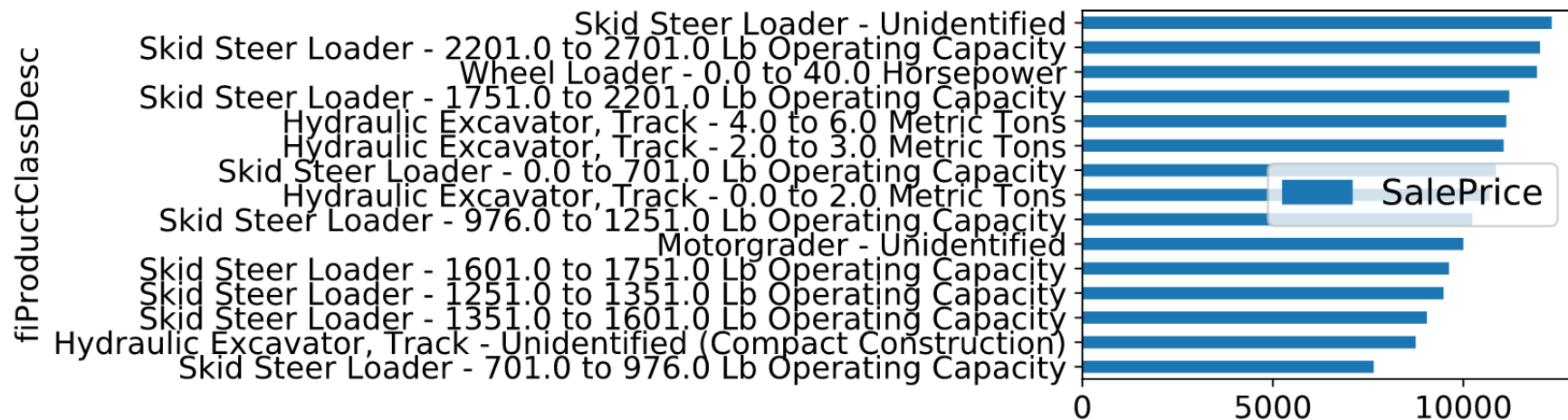
```
df['doorman'] = df['features'].str.contains('doorman')  
...
```

doorman	parking	garage	laundry
False	False	False	False
True	False	False	False
False	False	False	True



Splitting more complicated strings

- Bulldozers with higher operating capacity get higher prices, according to marginal plot



Splitting product class description string

- We can make the information more explicit by splitting the description into four pieces (and put into 4 new columns):

Track Type Tractor, Dozer -	20.0	to	75.0	Horsepower
<i>description</i>	<i>lower</i>		<i>upper</i>	<i>units</i>

- Description is a categorical variable, chosen from a finite set of categories such as “Skip Steer Loader”
- Lower and upper are numerical features
- Units is a category, such as “Horsepower”



Mechanics for splitting strings

- First split into description and spec on '-' char
- Then use regex to extract lower, upper, units

Track Type Tractor, Dozer	-	20.0	to	75.0	Horsepower
<i>description</i>		<i>lower</i>		<i>upper</i>	<i>units</i>

```
df_split = df_raw.fiProductClassDesc.str.split(' - ', expand=True)
df['fiProductClassDesc'] = df_split.values[:,0]
df['fiProductClassSpec'] = df_split[:,1] # temporary column
...
pattern = r'([0-9.\+]*)(?: to ([0-9.\+]*)|\+ )([a-zA-Z ]*)'
df_split = df['fiProductClassSpec'].str.extract(pattern, expand=True)
```

For details see <https://mlbook.explained.ai/bulldozer-feateng.html>

Injecting external data

- Sometimes we can inject data from outside our provided data set to increase model performance
- E.g., if sales for a store is 0, maybe that day was a national holiday or there was a hurricane; was there a pandemic?
- E.g., GPS location is important for rent price, but maybe proximity to cool neighborhood is stronger / more precise?
- E.g., home sales could be affected by many factors external to data set; what is the current consumer confidence index? How many IPOs recently in San Francisco? What is unemployment rate? Emigration rate for area?

Injecting external neighborhood info

- Rent data set has longitude and latitude coordinates, but a more obvious price predictor would be a categorical variable identifying the neighborhood, though, a numeric feature might be more useful
- Use proximity to desirable neighborhoods as a numeric feature (I love living within a few blocks of 50 different restaurants)
- Forbes magazine has an article with neighborhood names; using a mapping website, we can estimate GPS for them
- Compute so-called *Manhattan distance* (also called *L1 distance*) from each apartment to each neighborhood center

Injecting L1 proximity mechanics

- Iterate over neighborhoods and use vector math to compute new column per neighborhood

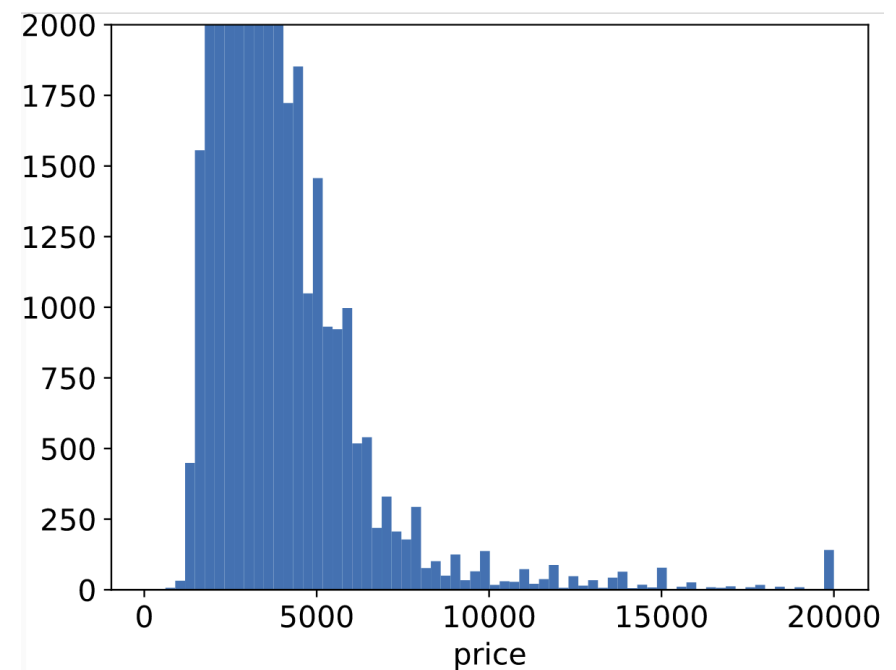
```
hoods = {  
    "hells" : [40.7622, -73.9924],  
    "astoria" : [40.7796684, -73.9215888], ... }  
for hood, loc in hoods.items():  
    # compute L1 manhattan distance  
    df[hood] = np.abs(df.latitude - loc[0]) + \  
                np.abs(df.longitude - loc[1])
```

- BTW, dropping longitude and latitude and retraining a model shows a similar OOB score but shallower trees in my tests

Log in, exp out for regression

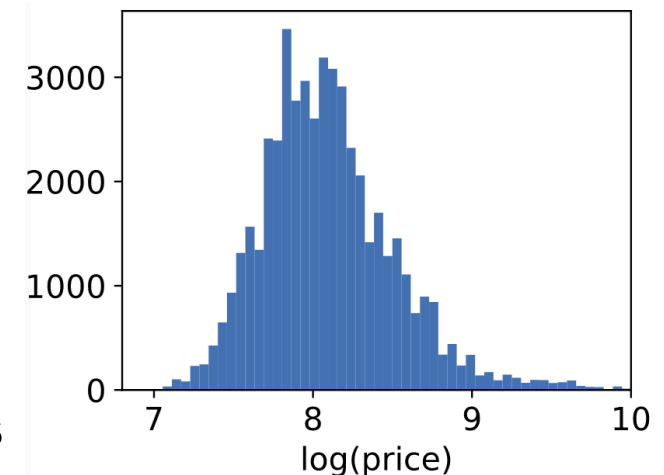
(Could be considered a part of data cleaning)

- Apt rent: consider distribution of prices clipped to less than \$20,000 and zoomed in
- There's a very long right tail, which skews RF predictions based upon mean of leaf y 's and also training based upon MSE
- Many target y , such as prices, are best compared as ratios and long tail makes MAE/MSE subtraction even more wonky



Transforming the target variable

- Goal: a tighter, more uniform target space
- Optimally, the distribution of prices would be a narrow “bell curve” distribution without tail
- Even restricted to \$1k..\$10k it’s still skewed
- BUT, check out what **log** does to distribution of ALL prices, not just < \$20k! (shrinks large values a lot and smaller values a little)
- Max price drops from millions to 10 without having to think or clip prices
- RF on unclipped prices gets $R^2 \sim 0$, but RF trained on $\log(\text{unclipped price})$ gets $R^2 \sim 0.87$



- Recall subtraction in log dollars domain is a ratio in dollar domain
- Training with MSE therefore compares squared ratio of y to \hat{y}

Effect on target space

```
y_pred_log = rf.predict(X_test)
y_pred = np.exp(y_pred_log)
```

- Revisit small region of New York City with outliers
- RF on raw prices predicts \$358,575
- RF on $\log(\text{price})$ predicts 9.92 (in $\log \$$)
- Transform predicted price back to \$ space with $\exp \Rightarrow \$20,395$
- Average in the log price space is less sensitive to outliers

	bedrooms	bathrooms	street_address	price	log(price)
39939	1	1	west 54 st & 8 ave	2300	7.7407
21711	1	1	300 West 55th Street	2400	7.7832
15352	1	1	300 West 55th Street	3350	8.1167
48274	1	1	300 West 55th Street	3400	8.1315
29665	1	1	333 West 57th Street	1070000	13.8832
30689	1	1	333 West 57th Street	1070000	13.8832

Reminder: rectify training and test sets

- Transformations must be applied to features consistently across data subsets (train, validation, test)
- Transformations of validation/test sets can only use data derived from training set
- To follow those rules, we have to remember all transformations done to the training set for later application to the validation and test sets.
- That means tracking the median of all numeric columns, all category-to-code mappings, frequency encodings, and one-hot'd categories
- Special care is required to ensure that one-hot encoded variables use the same name and number of columns in the training and testing sets.

Summary of techniques

- Counting
 - Count number of photos or words in a description
 - for transaction-like data with multiple entries referring to same entity, count number of previous loan applications
 - Frequency encoding an option for nominals (replace **ManagerID** with # apts managed)
- Interaction terms; price per square foot, bedrooms to bathrooms, ...
- Compute min,max,avg of key numerical columns (ATM withdrawal amounts)
- Rank encoding; convert raw numbers to their rank
- Extract important words from strings like "doorman" into Boolean columns
- Extract numeric values from strings, such as "101 Howard"
- Inject external data, such as holiday days or best neighborhoods in SF
- Log transform the target for regressors