BROCCOLI*

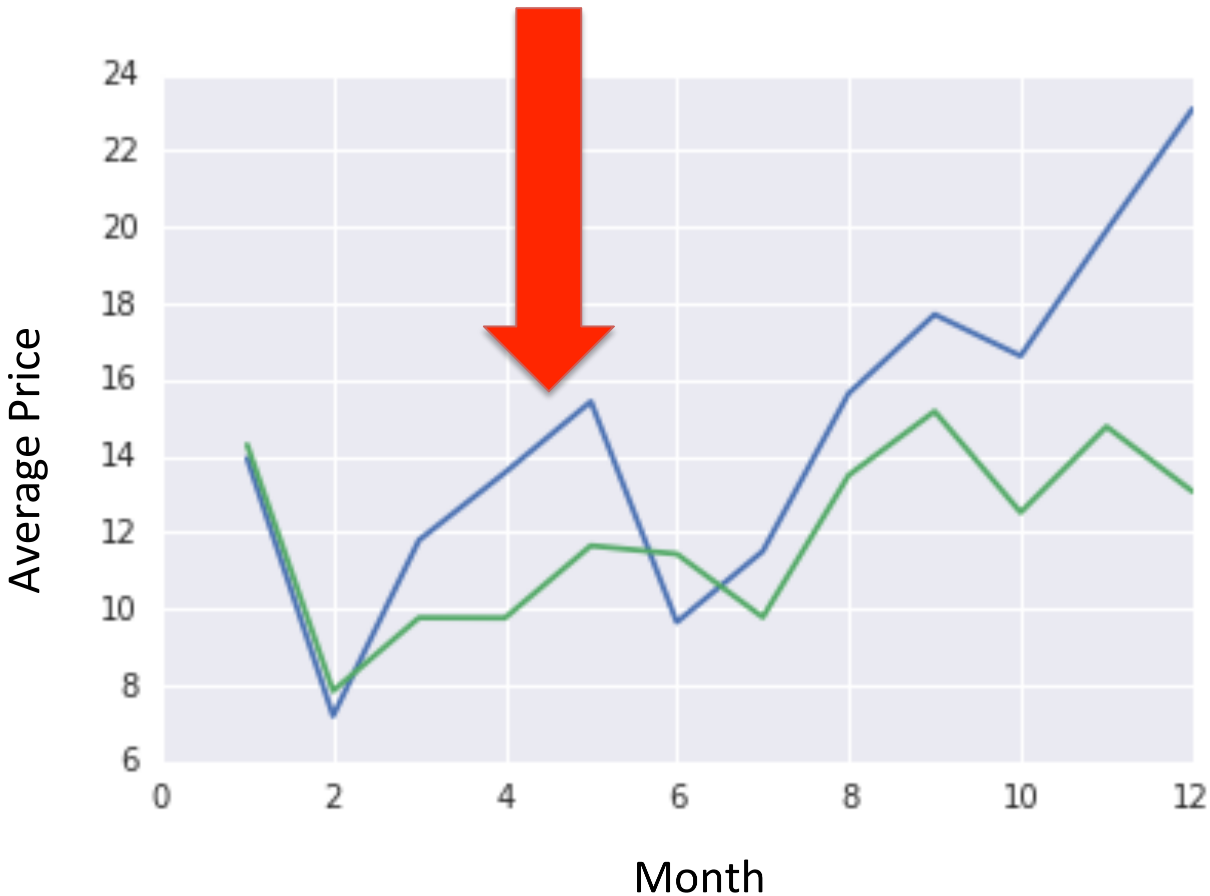
the great California brassica crisis of 2015

**my least favorite vegetable and its not because I'm picky, its science. Tell your mom [more here]()*

what the hell happened in 2015?

average broccoli price by season

higher than average price spike in September



peak demand drove prices up

drought conditions affected crop harvest and planting earlier in the season

affected all brassicas (cauliflower, broccoli, romanesco)

2015
2011-2016

contract structures for produce

- pay market price
- contract
  - we pay x$ a pound, no matter what, for the agreed upon time frame
  - farmers don't love this for obvious reasons
- contract plus
  - contract plus a percentage if market value goes above an agreed upon amount
    - example: we agree to pay $12 a pound unless the price goes above $15, then we'll add a dollar to our base price for every dollar the market raises (if market is $18, we would then be paying $15)
  - where do you set the trigger?

THE DATA

brought to you by the United States Department of Agriculture

8744 rows

organized by date and sale point 2012-2015 (drought years)

- city (in CA)
- package (case size)
- variety (crown cut or bunch)
- date
- low price
- high price
- mostly low price
- mostly high price
- season (year)
- demand tone
- market tone
- comment

- 32 columns total with a lot of nulls and empty columns
- market and demand tones were entered as strings with typos and no formatting

cleaning the data

- standardize the varieties and turn into integers

```
#convert variety strings to categories for bunch = 0, and crown cut = 1)
b['variety'] = b.variety.map({'BUNCH':0, 'CROWN CUT':1})
```

```
b['variety'].unique()
```

```
array([0, 1])
```

- create an average price column and an average mostly column (just in case)

```
#adding 2 coloumns with the average price for each data point, both 'mostly" and straight price
b['avg_price'] = b[['low_price', 'high_price']].mean(axis=1)
b['avg_mostly_price'] = b[['mostly_low', 'mostly_high']].mean(axis=1)
```

```
#replace all isnulls with overall average
b['avg_price'].mean()
```

```
11.80257815419112
```

```
b['avg_mostly_price'].mean()
```

```
11.493161428135432
```

```
b['avg_price'].fillna(11.80,inplace=True)
```

```
b['avg_mostly_price'].fillna(11.49,inplace=True)
```

- standardized and shorten the column names, drop empty columns, fill nulls with averages, added a month column

cleaning the data

market and demand tone, the messiest bits

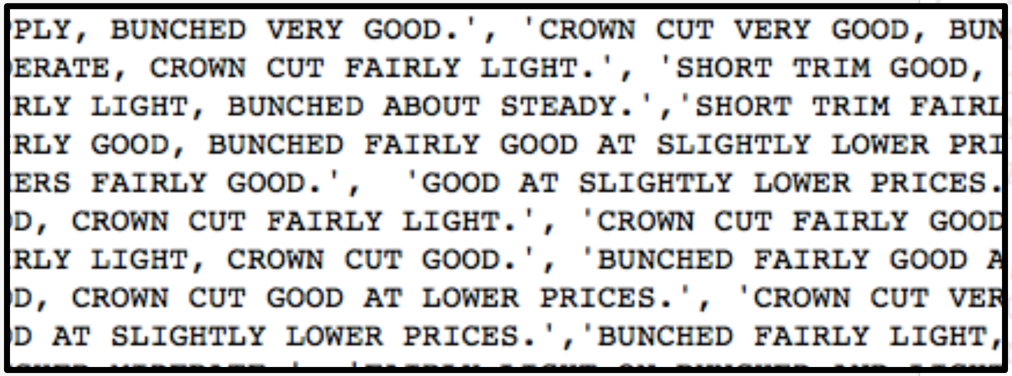
- not standardized, entered by hand (with typos), strings, with an unintuitive hierarchy structure (is fairly light, better than moderate or mostly moderate?

```
'''
breaks demand tone into 3 categories, bunched, crown cut and general based on a 5 pt scale
VERY LIGHT = 0
LIGHT = 1
FAIRLY LIGHT = 2
MODERATE = 3
FAIRLY GOOD = 4
GOOD = 5
VERY GOOD = 6
and then fill any na with the coloumn average
'''
```

```
'\nbreaks demand tone into 3 categories, bunched, crown cut and general based on a 5 pt scale\nVERY LIGHT = 0\nLIGHT = 1\nFAIRLY LIGHT = 2\nMODERATE = 3\nFAIRLY GOOD = 4\nGOOD = 5\nVERY GOOD = 6 \nand then fill any na with the coloumn average\n'
```

```
#breaks "bunched" demand out of demand tone
def f(x):
    if 'BUNCHED VERY LIGHT' in str(x):
        return 0
    elif 'BUNCHED LIGHT' in str(x):
        return 1
    elif 'BUNCHED FAIRLY LIGHT' in str(x):
        return 2
    elif 'BUNCHED MODERATE' in str(x):
        return 3
    elif 'BUNCHED FAIRLY GOOD' in str(x):
        return 4
    elif 'BUNCHED GOOD' in str(x):
        return 5
    elif 'BUNCHED VERY GOOD' in str(x):
        return 6

b['demand_bunch'] = b['demand_tone'].apply(f)
```




```
b['demand_bunch'].mean()
```

```
3.152444870565676
```

BROCCOLI PRICES PREDICTED

(ish)

models: trees, lines and buckets

- 3 machine learning models
  - decision tree (rmse 4.69, null accuracy 2%, 3 splits)
    - curious what a random forest would look like with more splits and more options to work through the noise
  - linear regression (27% accurate, null accuracy 2%)
    - too difficult to predict the price exactly
    - this was a little heartbreaking
  - logistic regression (60% accurate, null accuracy 52%)
    - split it into 5 price buckets to predict
    - worked much better but the range was still very high

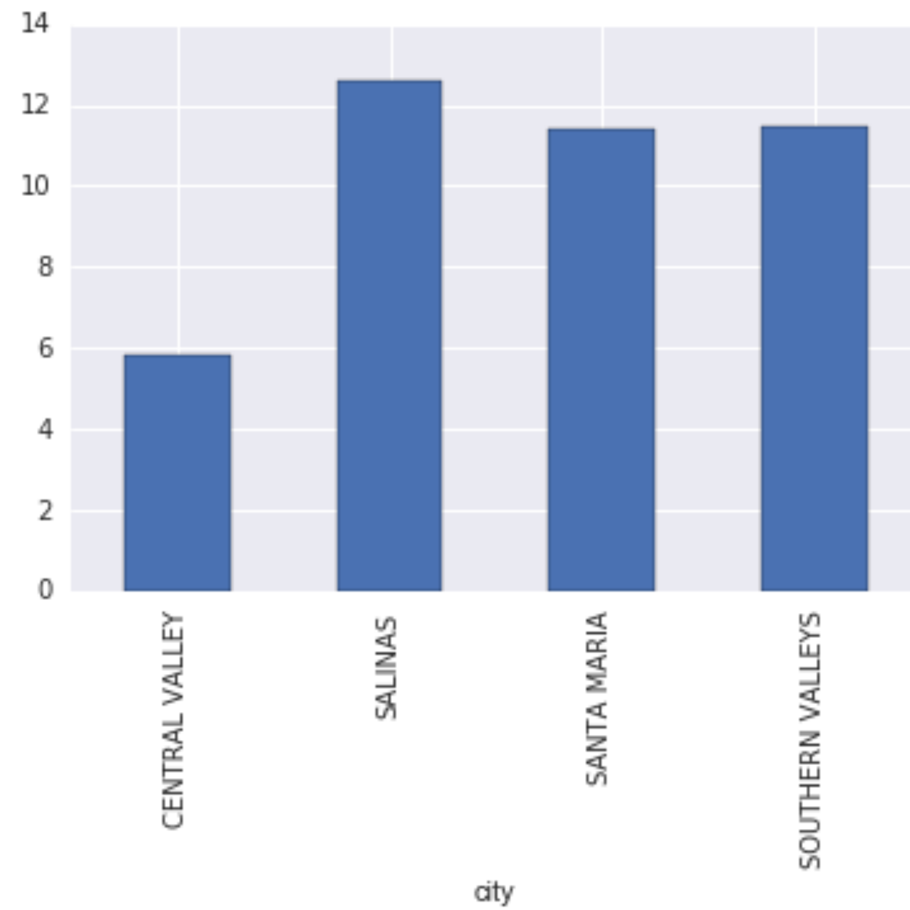
surprises

- correlations were not what I expected
  - high correlation between season and month were not surprising
  - low correlation between market tone and weather
    - possible data integrity issue?

	feature	importance
2	season	0.474276
3	month	0.350854
5	demand_general	0.109849
1	variety	0.065021
0	package	0.000000
4	weather	0.000000
6	market_tone_general	0.000000

surprises

- low linear regression score
- high variation by location



challenges

- data cleaning was the largest challenge
- finding a model that worked well
- choosing the best parameter was more challenging than I thought it would be
- integrity and scope of the initial dataset
  - first time I worked with data too big to look at the whole thing reasonably for patterns before I started, had to rely on the machine learning models


$18.77
projected price of broccoli in September 2016