Iowa Liquor Sales

Predicting Sales Prices for the Following Year

1. Introduction

The Dataset

- The state of Iowa gathers and releases yearly data on liquor sales.
- I was interested in how much data the state of Iowa had collected (over 12.5 million rows) of liquor sales across about five years.
- For more about Iowa’s plan concerning collecting and releasing many datasets, see their webpage.
- As of late-2020, the website from which one can download the Iowa Liquor Sales datasets as a CSV files from Kaggle (3.23 GB) and the official government webpage for general information.

Purpose and Hypothesis

- My theoretical clients are major liquor retailers in Iowa (and perhaps similar places) who would benefit from knowing profits for the following year.

2. Data Cleaning

Data Cleaning

- The dataset is deceptively dirty.
  - 1 complete table
  - Many discrepancies in the pandas Series

1. Make all strings lowercase
   a. pandas Series
   b. Column titles
2. Dates
   a. Move ‘date’ column to first place.
   b. Place rows in chronological order.
   c. Insert ‘year’, ‘month’, and ‘day’ columns.

Data Cleaning: Missing Values

county_number and county are lost.

category_name and category: about half of the category_name column can be redeemed.

- I don’t need to redeem it for the ML model.

Fill NaNs with numerics.

	Total	Missing Percent
county_number	79178	0.629
county	79178	0.629
category_name	16086	0.128
category	8020	0.064
zip_code	2420	0.019
address	2376	0.019
store_location	2375	0.019
city	2375	0.019
sale	10	0.000
bottle_cost	10	0.000
state_bottle_retail	10	0.000
vendor_number	3	0.000
vendor_name	1	0.000

Data Cleaning: category_name and category

category_name NaNs:

- 16,086 → 8,020

```
def sample_from_dict(d, sample=4):
    keys = random.sample(list(d), sample)
    values = [d[k] for k in keys]
    return dict(zip(keys, values))
```

```
merged_dict = dict(zip(list_num, list_name))
sample_from_dict(merged_dict)
```

```
{1082000.0: 'SINGLE MALT SCOTCH',
 1062400.0: 'American Flavored Vodka',
 1062050.0: 'Cocktails / RTD',
 1082015.0: 'Holiday VAP'}
```

```
def fillNan(row,axis=1):
    if row.category_name == 'nan':
        return merged_dict.get(row.category)
    else:
        return row.category_name

df.category_name = df.apply(fillNan,axis=1)
```

Data Cleaning: Location Chunk

Location chunk of columns:

- zip_code
- address
- store_location
- city

zip_code has many non-numeric characters like hyphens.

	Total	Missing Percent
county_number	79178	0.629
county	79178	0.629
category_name	16086	0.128
category	8020	0.064
zip_code	2420	0.019
address	2376	0.019
store_location	2375	0.019
city	2375	0.019
sale	10	0.000
bottle_cost	10	0.000
state_bottle_retail	10	0.000
vendor_number	3	0.000
vendor_name	1	0.000

Data Cleaning: zip_code Non-Numerics

```
import re
def remove_non_nums(i_str):
    return re.sub(r'\D', '', str(i_str))
```

```
df.zip_code = df.zip_code.apply(remove_non_nums)
```

The return statement uses the re module to substitute all non-digit characters with an empty string: ‘’.

Using the apply method (opposed to remove_non_nums(df.zip_code)) sped the process up exponentially.

Data Cleaning: Sales-Related Columns

I filled NaN with 0, and still had 10 missing values. Upon further inspection, there were many non-numerics.

	Total	Missing Percent
county_number	79178	0.629
county	79178	0.629
category_name	16086	0.128
category	8020	0.064
zip_code	2420	0.019
address	2376	0.019
store_location	2375	0.019
city	2375	0.019
sale	10	0.000
bottle_cost	10	0.000
state_bottle_retail	10	0.000
vendor_number	3	0.000
vendor_name	1	0.000

Data Cleaning: Sales-Related Columns

I convert all ‘nan’ values to the string ‘0.’

Strip the ‘$’ character (the only non-numeric character)

- ```
  df.sale = [x.strip('$') for x in df.sale]
  ```

Convert to numeric using pd.to_numeric()

Use sklearn’s SimpleImputer to impute the median for the sales-related columns.
  ```

Sales-Related Columns

```
4316 rows of the above contain zero. They were NaN before converted to 0.
 3094 rows contain NaNs for all three columns: state_bottle_retail, bottle_cost, sale.
         - 3094 rows are state_bottle_retail.
         - 3094 rows are bottle_cost.
         - 3094 rows are both state_bottle_retail and bottle_cost.
 4316 rows are sale alone, with 1222 more than the other factors.
```

Why are there 1,222 more missing values in ‘sale’?

- Most transactions missing a sales column only sold 1 bottle, and after them, it is 2 and 3 bottles.
- Managers may benefit from knowing that the fewer bottles purchased per transaction increase the chance of not recording the sales price.

Making pandas Series Values Lowercase

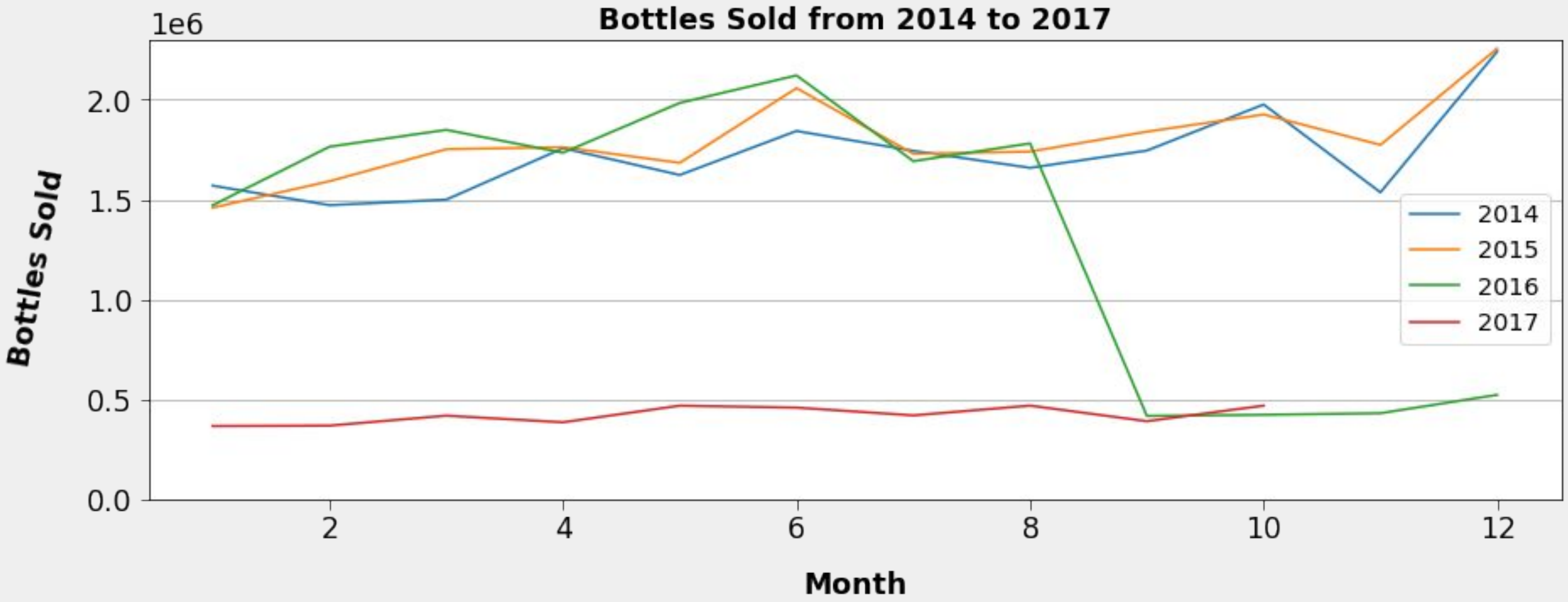
```
for object_column in object_column_list:
    df.loc[:,object_column] = df.loc[:,object_column].str.lower().str.strip().str.split().str.join(' ')
    print(object_column)
    gc.collect()
```

For-loop

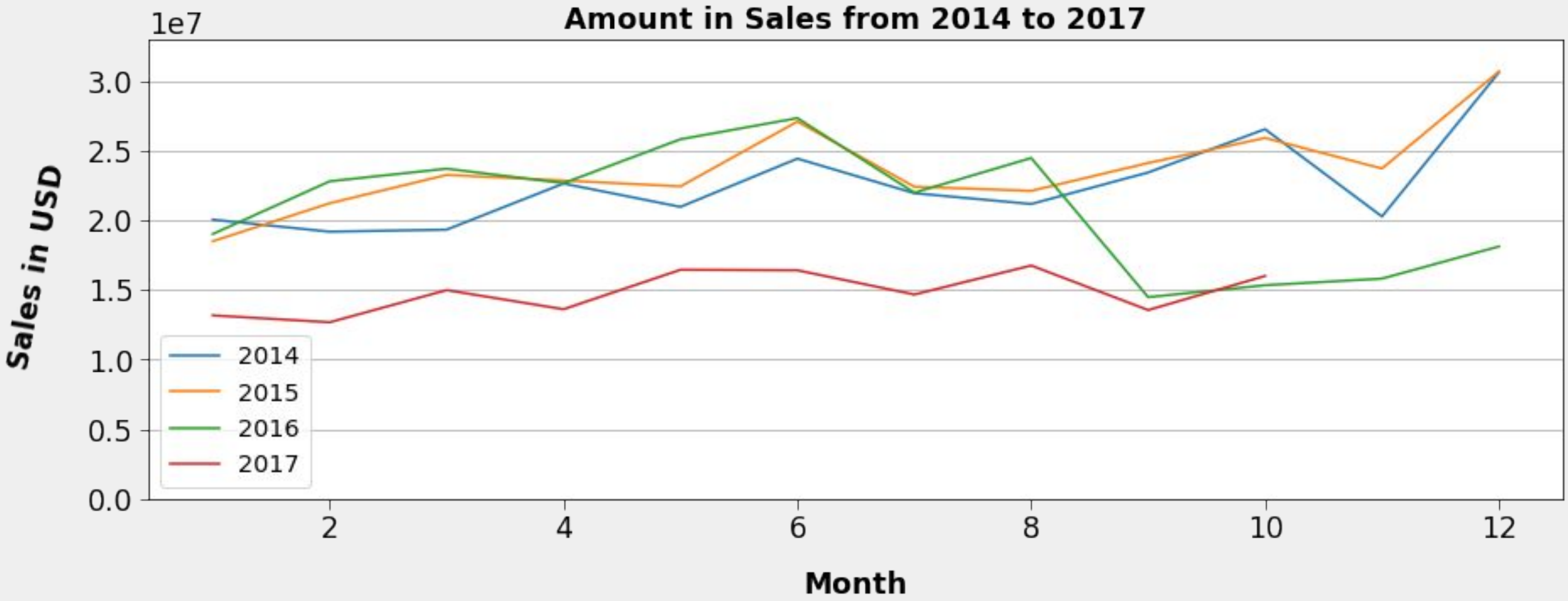
- All columns with the ‘object’ dtype equals such as lower case values and no extra or trailing spaces.
- Print each column after finishing.
- (Garbage Collector is a module that cleans up excess data in order to free up memory.)

3. EDA

EDA: Bottles Sold

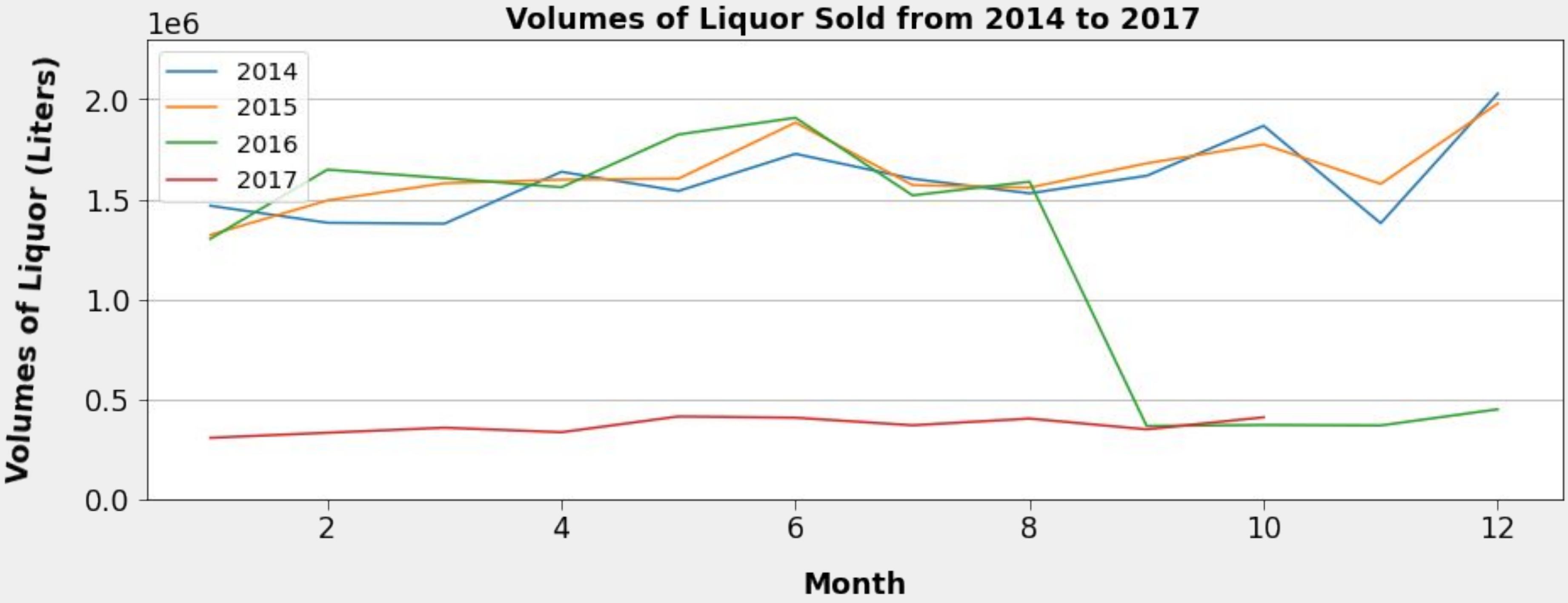


EDA: Sales

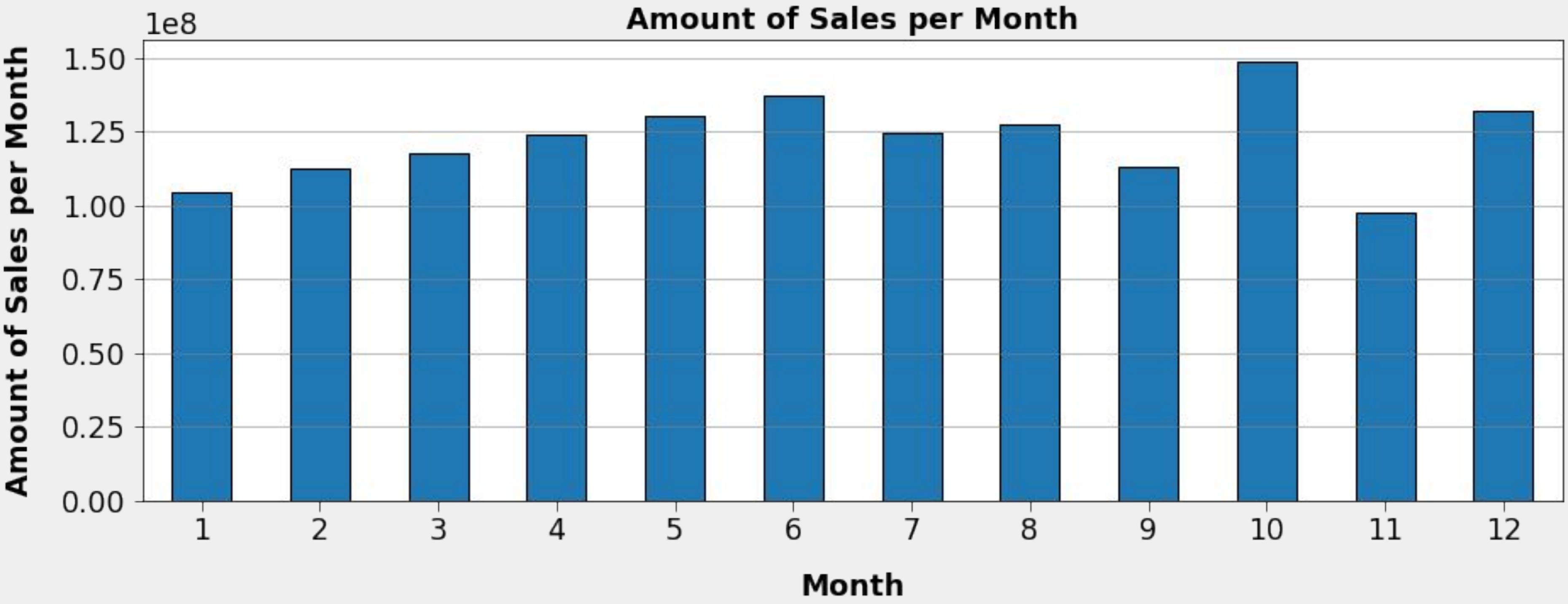


Amount in Sales from 2014 to 2017
1e7
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Sales in USD
2
4
6
8
10
12
Month
2014
2015
2016
2017

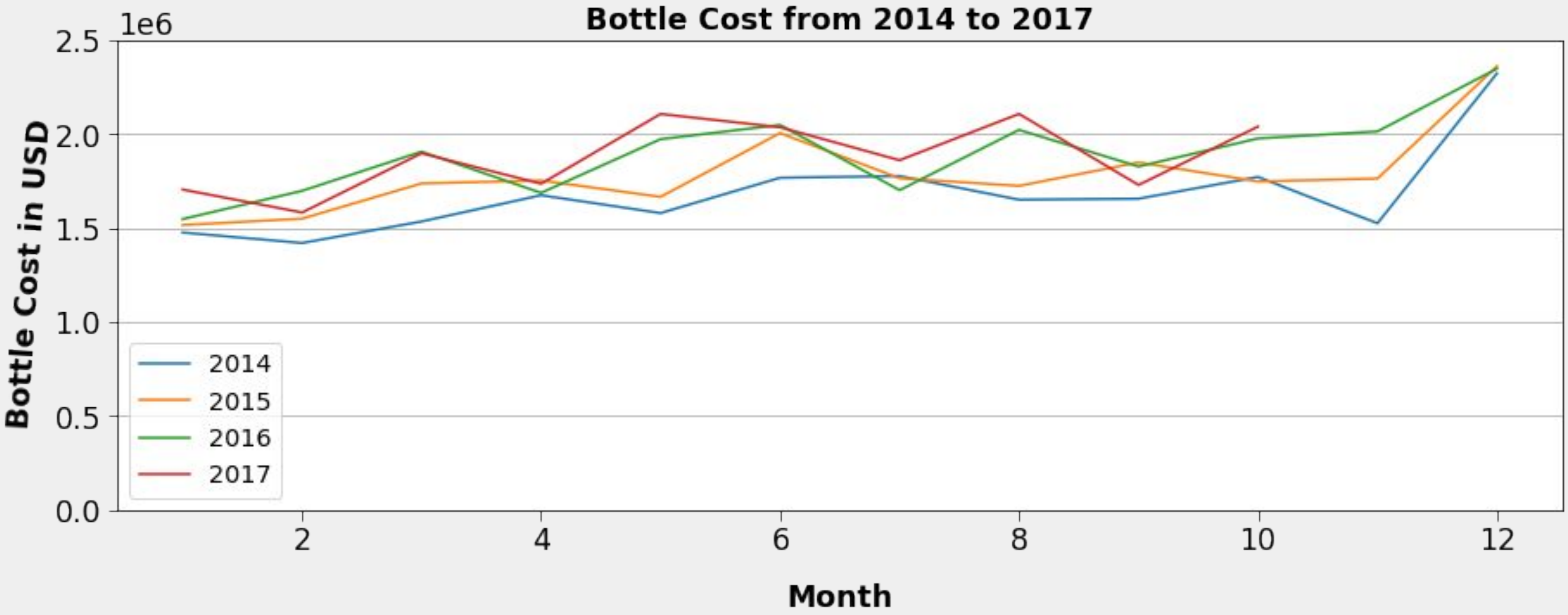
EDA: Volumes Sold



EDA: Aggregate Seasonality



EDA: Bottles Sold



EDA: Drop in Sales, Not in Bottle Cost

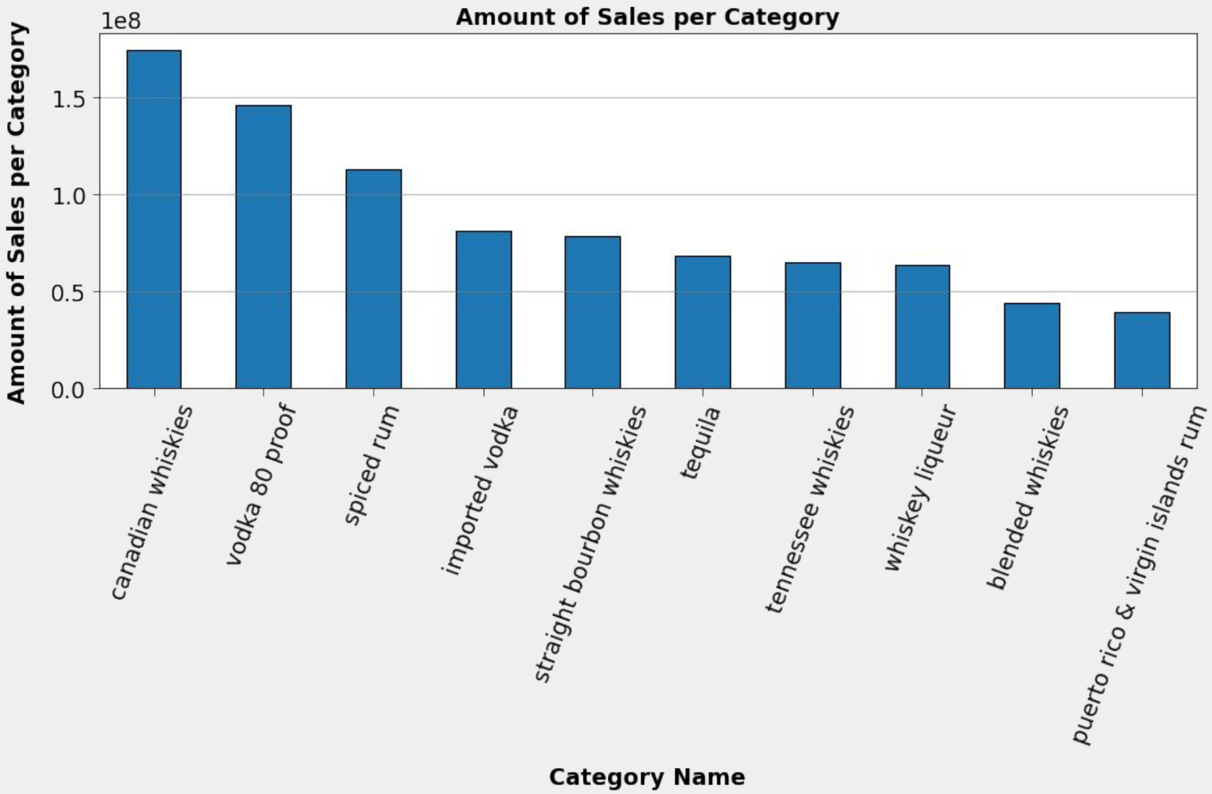
The prices have stayed the same, yet liquor stores seem to be selling less. The first explanation is to suspect something wrong with data acquisition, but the Iowa Liquor dataset website does not indicate any problems or changes in collecting data, which makes sense because they release the data by year. And the change began mid-2016. I'm no economist, but it seems odd that the amount of bottles being sold did not noticeably affect the pricing of bottles.

A second explanation may be found in the minutiae of the no-doubt complex interplay between Iowa liquor laws and the rapid growth of micro-breweries roughly around the time of 2016. There are many articles one this topic, and how laws are more friendly to breweries than liquor stores.

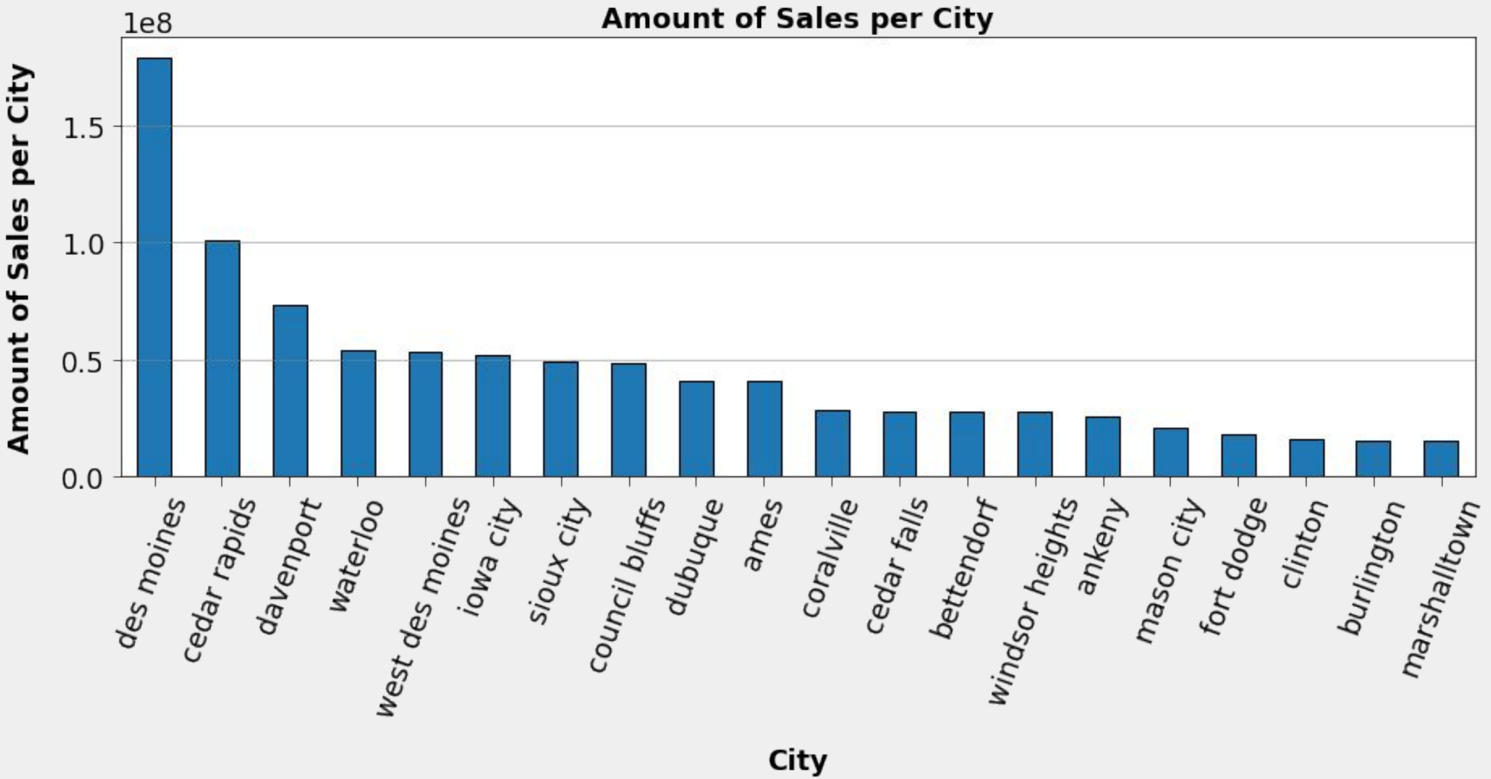
Here are some news articles on the interplay between changing Iowa liquor laws, dispeace about them, and the rise of microbreweries:

- Article 1
- Article 2

EDA

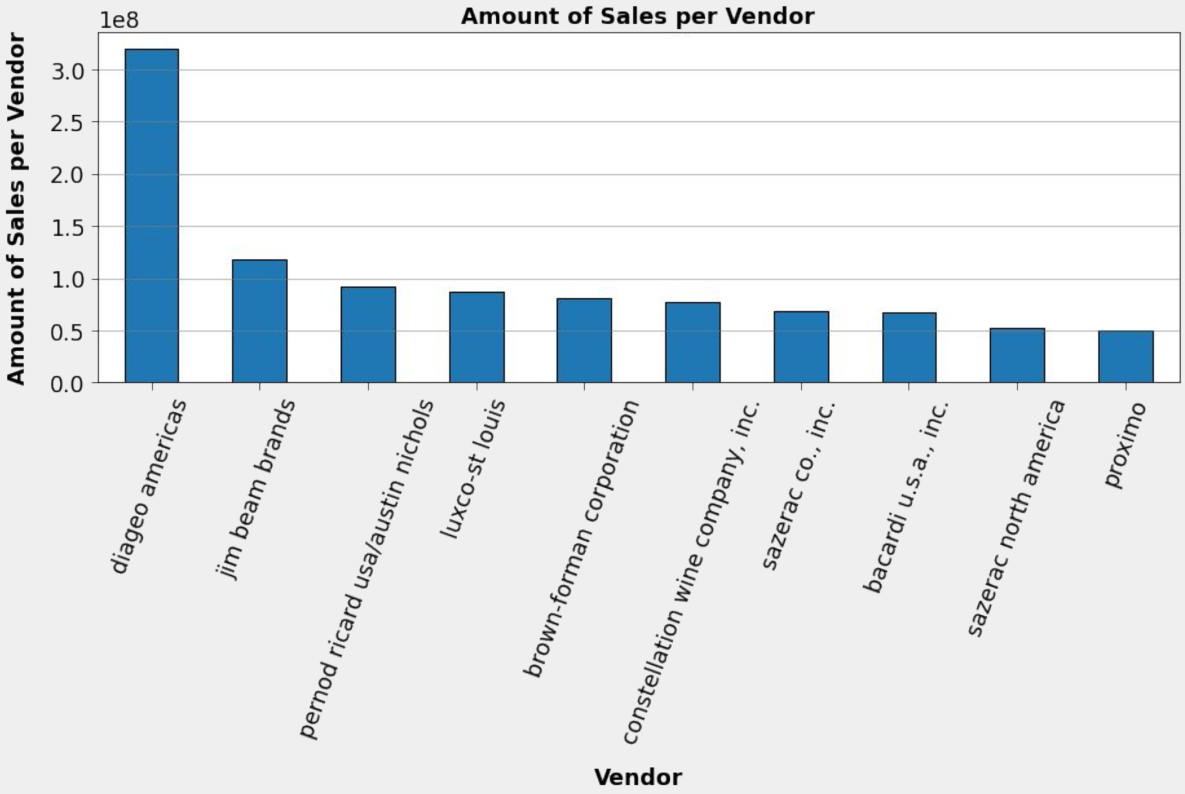

Amount of Sales per Category
1e8
Amount of Sales per Category
0.0
0.5
1.0
1.5
canadian whiskies
vodka 80 proof
spiced rum
imported vodka
straight bourbon whiskies
tequila
tennessee whiskies
whiskey liqueur
blended whiskies
puerto rico & virgin islands rum
Category Name

EDA

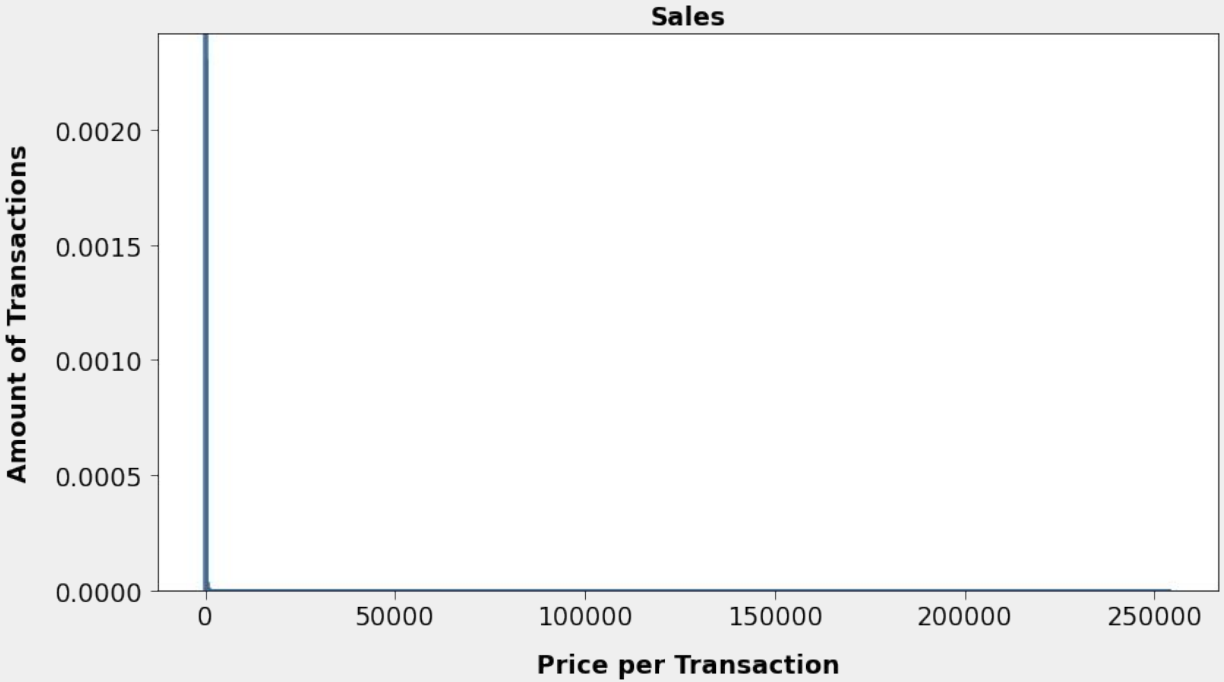


Amount of Sales per City
1e8
Amount of Sales per City
0.0
0.5
1.0
1.5
des moines
cedar rapids
davenport
waterloo
west des moines
iowa city
sioux city
council bluffs
dubuque
ames
coralville
cedar falls
bettendorf
windsor heights
ankeny
mason city
fort dodge
clinton
burlington
marshalltown
City

EDA: Categories

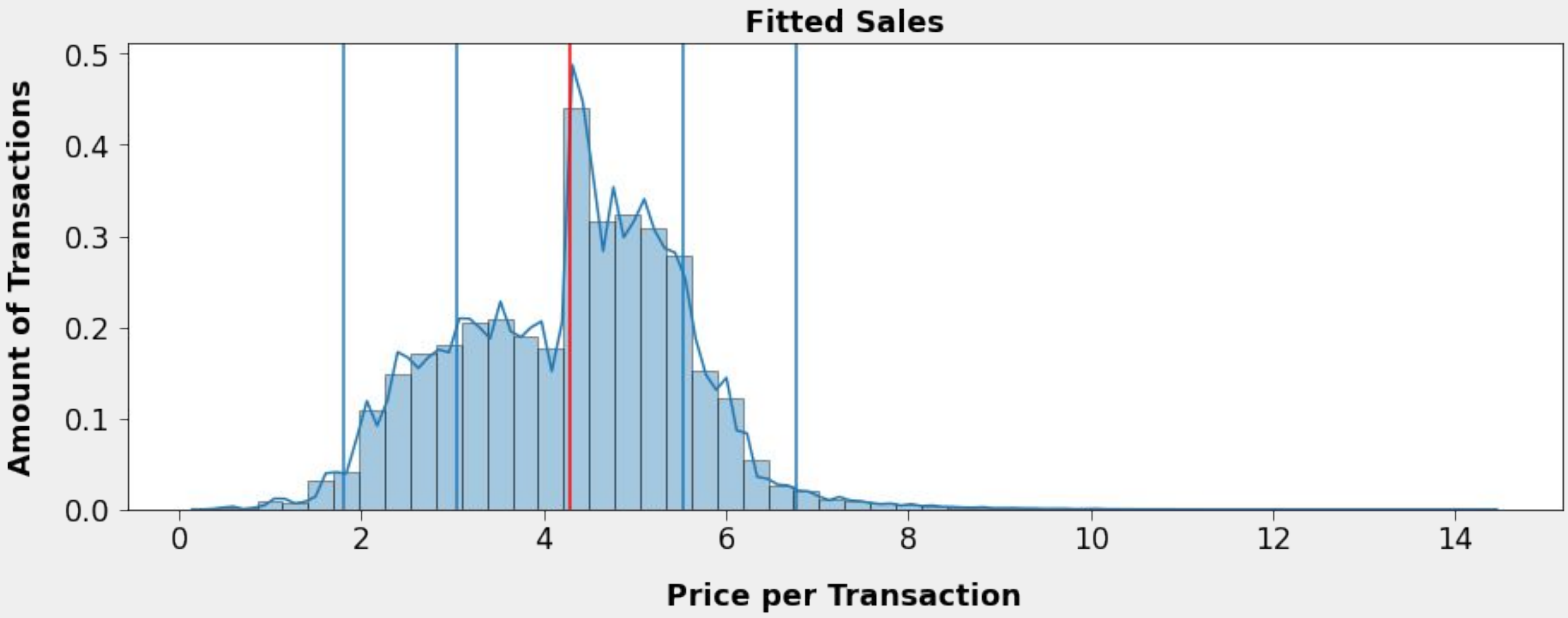


EDA: Distribution and Outliers



EDA: Distribution, Sales Less than $300

EDA: Entire Distribution after Boxcox



The red line is the mean, and the blue lines are each standard deviation.

EDA: Entire Distribution after Boxcox

Normal Data: excess kurtosis of normal distribution (should be 0): 44,774.07
Fitted Data kurtosis: 0.2

Normal Data:

- Mean: 114.48
- STD: 377.75

The STD being larger than the mean shows that the data has immense spread.

Fitted Data

- Mean: 4.28
- STD: 1.24

EDA: Outliers

Normal Data IQR:

- Q1: 25.48, Q3: 132.72, IQR: 107.24

Fitted Data:

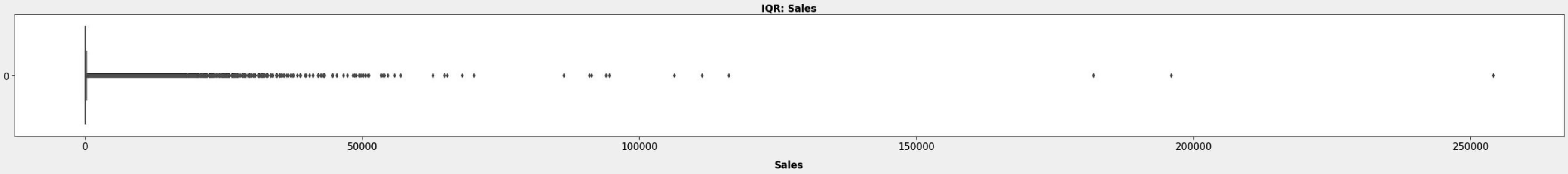
- Q1: 3.36, Q3: 132.72, IQR: 129.36

Instances 2 STDs above the mean

- Dataset: 116,323
- Fitted Data: 221,282

EDA: Outliers

Normal Data



Fitted Data

EDA: Predictive Power

Predictive Power Scores (pps) (between 0 and 1, where 1 is extremely predictive)

- Predictive power scores detect non-linear relationships between data that (unlike correlations) are not necessarily symmetric. For example, the city in which someone lives may be discovered if one knows the zip code, but the zip code will not be found if one only knows the city.

bottles_sold and vol_sold are the best predictors of sales across the four-year timespan.

x	y	ppscore
sale	sale	1.000000
fd	sale	0.968598
bottles_sold	sale	0.304367
vol_sold	sale	0.303635

4. Machine Learning (ML)

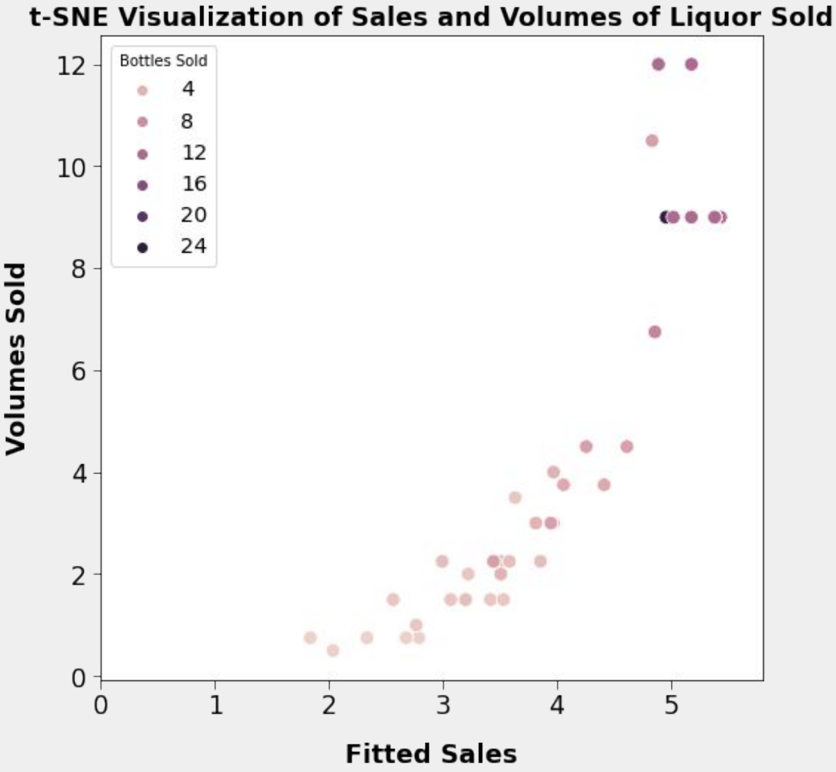
ML

- Purpose: predict the future sales prices.
- Supervised
- Continuous data
- I use regressors.
- The dataset was too large for my laptop, so I reduced the data to 100 rows only for the models.
- Imperfect iteration:
    a. For-loop of non-parameterized models
    b. Feature selection
    c. Parameterized, decision-tree models

ML: t-SNE Visualization

It seems that there is something like a positive, exponential relationship between sales and volumes of liquor sold.

Assigning the hue to the number of bottles sold indicates a positive, somewhat significant relationship between the previous relationship and the amount of bottles sold.



ML: Round 1a. For-Loop

The victors of the for-loop in descending order are:

- GradientBoostingRegressor: 0.9831
- XGBRegressor: 0.9803
- RandomForestRegressor: 0.9711
- LinearRegression: 0.9210
- Ridge: 0.9206

These are good baselines, and the linear models did surprisingly well (better than SVR: -2.2593).

ML: Round 1b. Feature Selection

I engage in feature selection, dropping the columns with no variance: year, month, day.
I notice the perfect correlation in the heatmap below between two variables and drop one (state_bottle_retail).

Unfortunately, the first object disagreed with the last two (despite it keeping five instead of three). I later went with the final two voters, keeping only bottle_cost, bottles_sold, and vol_sold. But for now, I only dropped vendor_number and store_subnumber.

```
(store_number       1
 county_number      0
 category           1
 vendor_number      0
 item_number        1
 pack               0
 bottle_vol         1
 bottle_cost        2
 bottles_sold       2
 vol_sold           2
 store_subnumber    1
 dtype: int64,
                        0      1      2
 store_number        True  False  False
 county_number      False  False  False
 category            True  False  False
 vendor_number      False  False  False
 item_number         True  False  False
 pack               False  False  False
 bottle_vol          True  False  False
 bottle_cost        False   True   True
 bottles_sold       False   True   True
 vol_sold           False   True   True
 store_subnumber     True  False  False)
```

ML: Round 2a. For-Loop

The decision trees are still in the lead, and Gradient Boosting Regressor won of the three.

```
LinearRegression()
    model score: 0.9187
Ridge()
    model score: 0.9189
Lasso()
    model score: 0.7941
ElasticNet()
    model score: 0.8474
LinearSVR()
    model score: -0.3231
RandomForestRegressor()
    model score: 0.9719
GradientBoostingRegressor()
    model score: 0.9886
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.300000012, max_delta_step=0, max_depth=6,
             min_child_weight=1, missing=nan, monotone_constraints='()',
             n_estimators=100, n_jobs=0, num_parallel_tree=1, random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
             tree_method='exact', validate_parameters=1, verbosity=None)
    model score: 0.9883
```

These scores are slightly better after feature selection.

ML: Round 2b. Feature Selection

Let’s take the vote more seriously and drop more rows: 'category', 'item_number', 'pack', 'bottle_cost', 'vol_sold'.

Later, we will use X_reduce and drop county_number, as the two decision-tree RFE models voted unanimously.

```
X.head()
```

	store_number	county_number	bottle_vol	bottles_sold
0	5022	77	750	4
1	2460	100	750	2
2	2590	57	750	5
3	2648	77	750	6
4	4312	78	750	12

ML: Round 2c. Parameterized Decision-Tree Models

Gradient Boosting Regressor performed best among itself, Random Forest, and XGB.

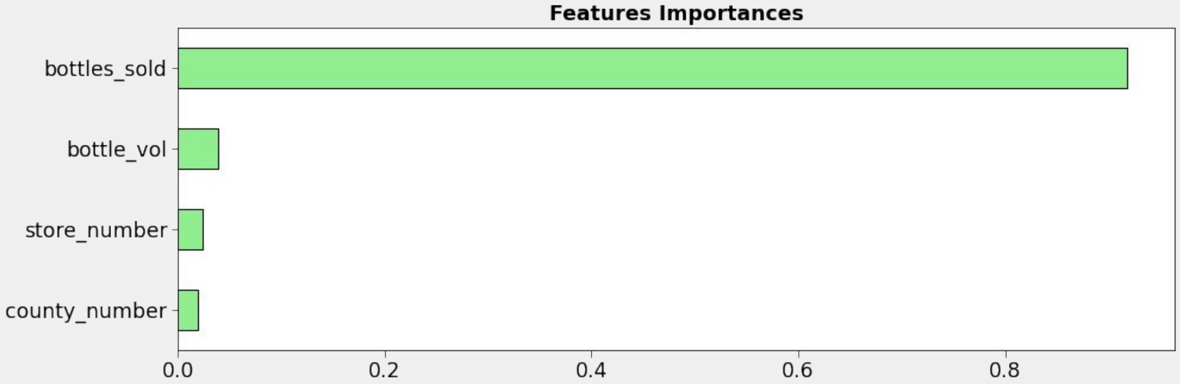
There was agreement on feature importance except for a very small range on the importance of the lesser values (.05 - .12).

Let's drop county_number and see if the models improve.

```
MSE: 0.054%
RMSE: 0.027
Score: 0.951
Explained Variance Score: 0.95
```

Without feature selection, the scores were:

- MSE: .05
- RMSE: .023
- Score: .95759



ML: Round 3a. Feature Selection

Let’s listen to the decision trees’ votes and drop county_number.

	bottle_cost	bottles_sold	vol_sold
0	6.50	4	3.00
1	10.00	2	1.50
2	6.50	5	3.75

ML: Round 3b. For-Loop

Much better scores all around. Notice that Linear SVR is positive.

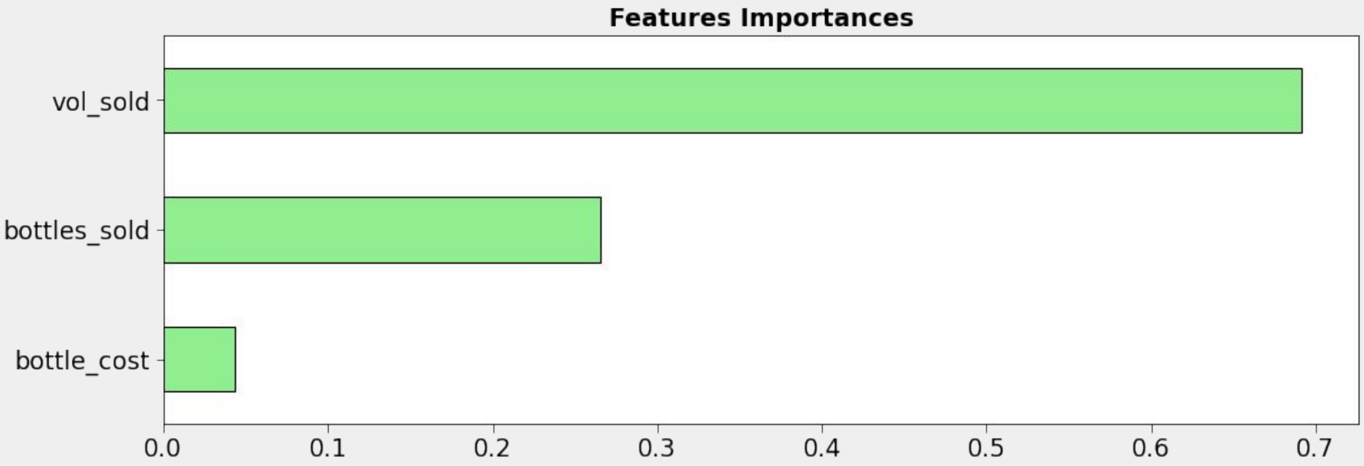
The XGB model performed the best (**.9973**), so let's parameterize that.

```
LinearRegression()
    model score: 0.9311
Ridge()
    model score: 0.9310
Lasso()
    model score: 0.7601
ElasticNet()
    model score: 0.8353
LinearSVR()
    model score: 0.9295
RandomForestRegressor()
    model score: 0.9927
GradientBoostingRegressor()
    model score: 0.9930
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning_rate=0.300000012, max_delta_step=0, max_depth=6,
             min_child_weight=1, missing=nan, monotone_constraints='()',
             n_estimators=100, n_jobs=0, num_parallel_tree=1, random_state=0,
             reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
             tree_method='exact', validate_parameters=1, verbosity=None)
    model score: 0.9973
```

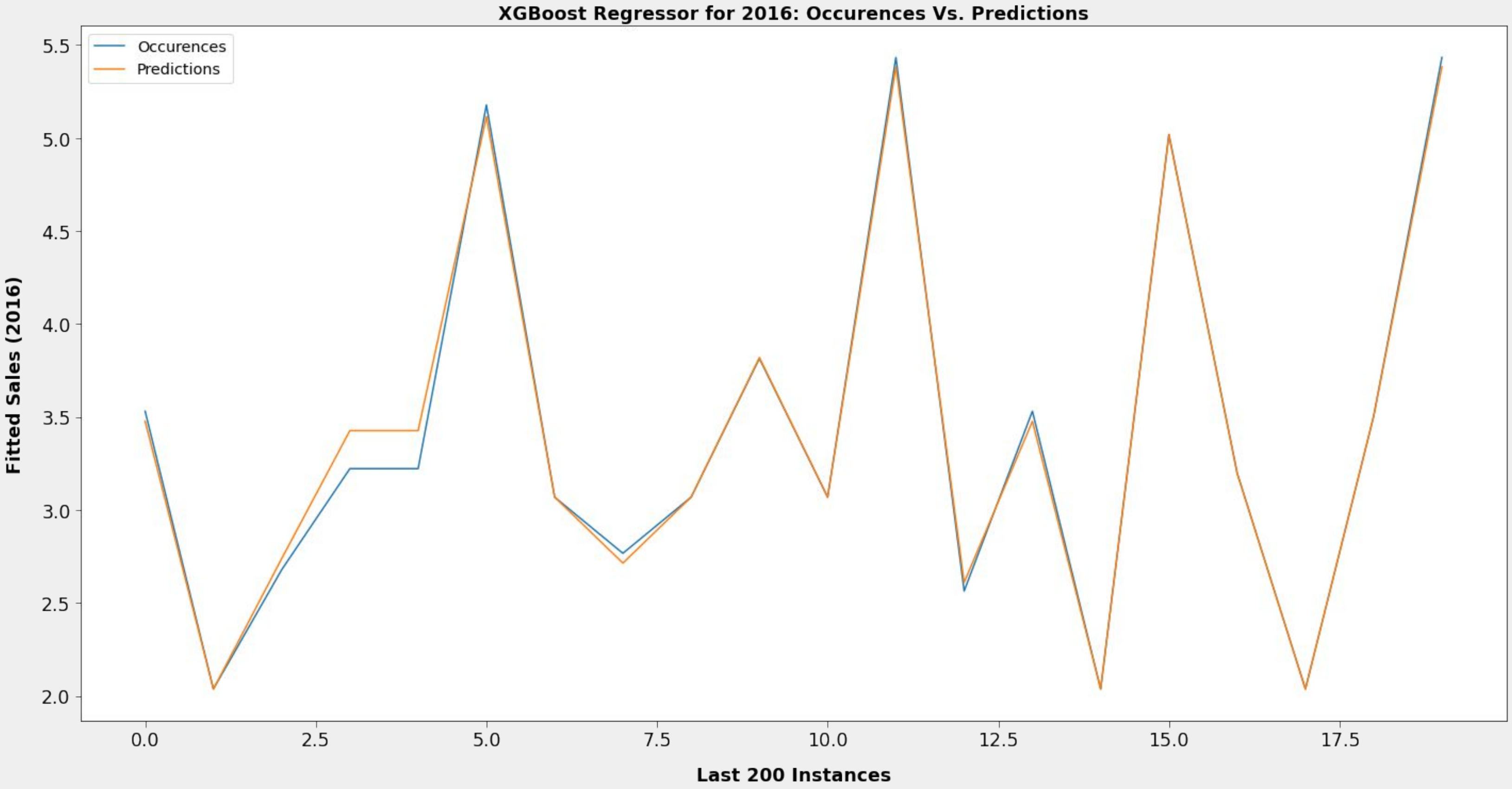
ML: Round 3c. Parameterized XGB Regressor

Maximal accuracy

```
MSE: 0.005%
RMSE: 0.003
Score: 0.995
Explained Variance Score: 1.00
```



ML: Round 3c. Parameterized XGB Regressor



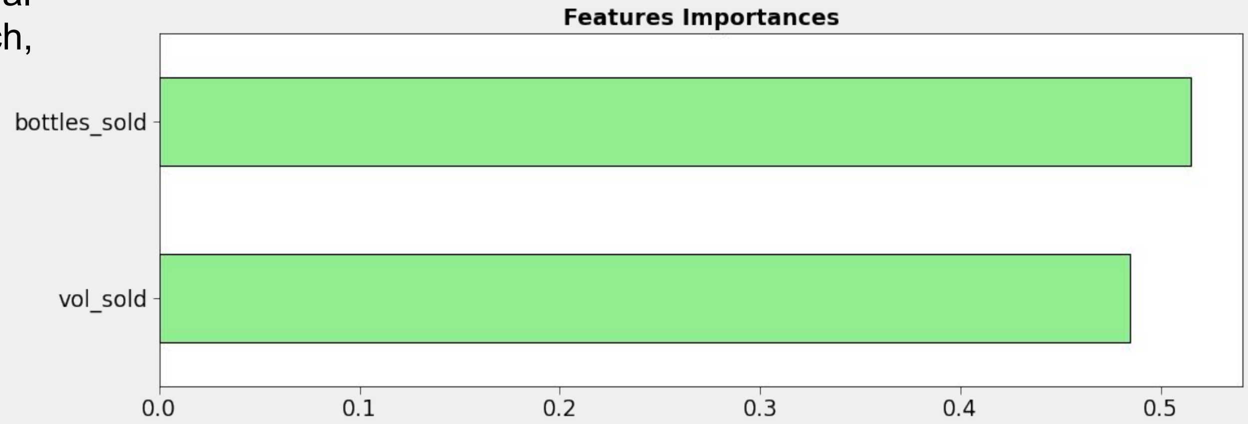
ML: Round 3d. Feature Importance

Let’s perform the same model but drop ‘bottle_cost’, so that our table has only two columns.

The score isn’t terrible, but significantly drops.

It seems to hold each in almost equal esteem, but that doesn’t tell us much, as the model is weak.

```
MSE: 0.046%
RMSE: 0.023
Score: 0.958
Explained Variance Score: 0.96
```



5. Interpretation

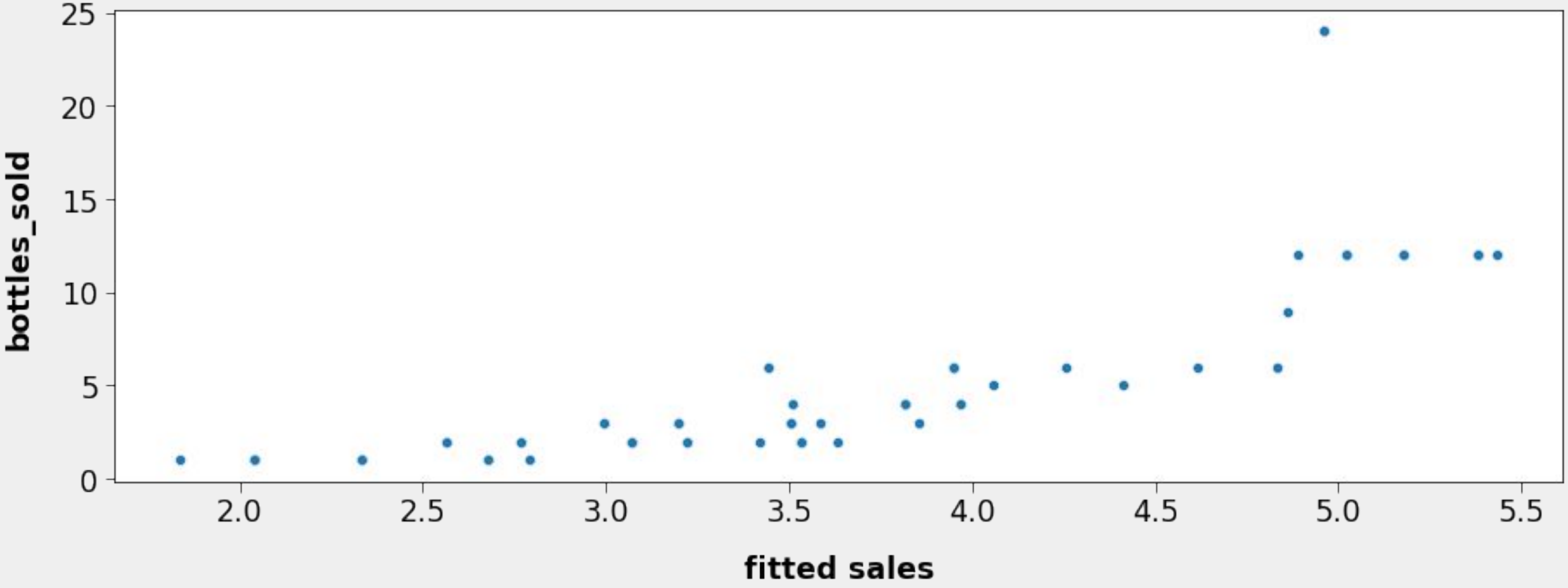
Interpretation

All features aside from the three in the above visualization could not justify their existence in the dataset; they were not worth the cost of another dimension. Dropping the weakest of the three crossed the line, however, and the model accuracy dropped significantly.

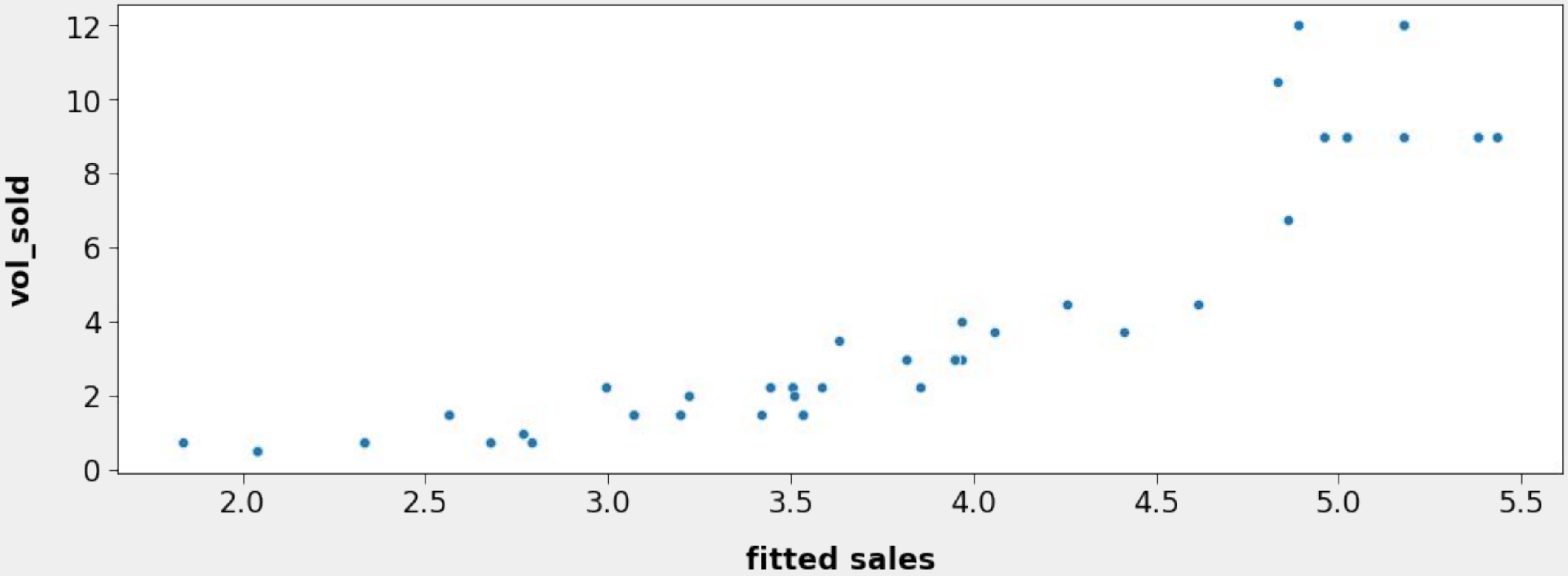
The most important features do not necessarily cause the target variable (sales crammed in boxcox, in this case) to increase or decrease directly. Still, an expert in the business domain should consider the three, and their significant, positive correlation with sales (as shown below).

One starting point to think causally is to suspect that the bottle_cost has a somewhat wide range when it comes to sales, thereby prompting store owners to drop prices more (perhaps in the form of sales).

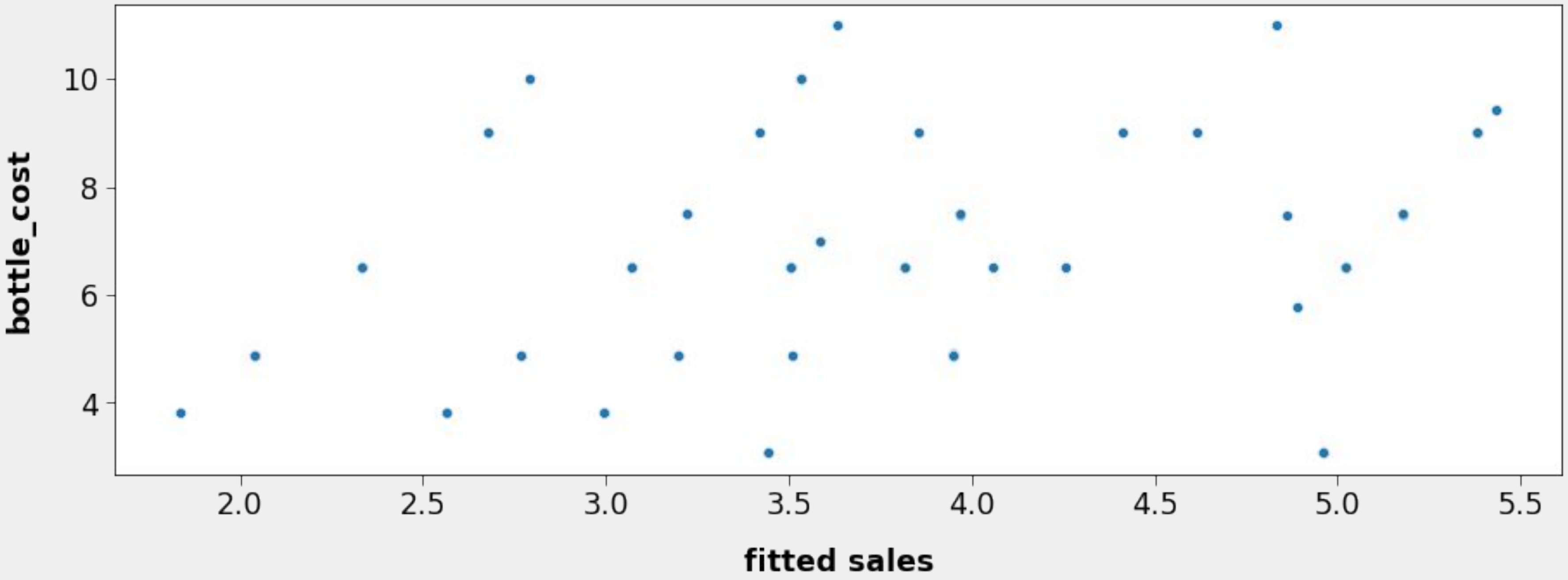
Correlations: R= 0.84



Correlations: R= 0.91



Correlations: R= 0.42



6. What Would I Do Different If _____?

Time, Computational Power

Data Imputation and Outliers

- Data imputation
  - MICE
- Dates
  - FBProphet
- Outliers
  - Drop them
- ML
  - Data Prep: Category module like CatBoost
  - Hyperopt

7. Conclusion

Conclusion

- Dataset
  - Access: government webpage, Kaggle.
- Data Cleaning
  - Recover the recoverable
  - Make values uniform and neat
  - Impute NaN values
- EDA
  - The time dimension
  - Sales in relation to categories of liquor, cities, liquor vendors
  - Distribution and outliers
  - Predictive Power Scores

Conclusion

- ML
  - 3 Rounds
    i. For loop
    ii. Feature selection
    iii. Parameterized model (decision trees)
  - Interpretation
    i. The XGBRegressor scores best and well
    ii. Feature Importance
        - volumes sold, bottles sold, and bottle cost
    iii. Domain experts may experiment with dropping prices even more liberally.