Data Science Mid Project Checkpoint

Eric Meissner, Neal Smart, Maneesh Mohanavilasam

November 17, 2015

1 Overview

We are proposing to compete in the Kaggle competition setup by Rossman Stores to predict store sales for the next 6 weeks of business from previous business data.. All the information on the dataset is here. There's information about competitors, daily sales grosses, etc.

1.1 Code Location

Our exploratory code is found on Eric's github.

1.2 Descriptive Statistics and Discussion

```
In [42]: import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt
         %matplotlib inline
         %pwd
```

```
Out[42]: u'/home/eric/dev/projects/data_science/rossman_predictor'
```

Our data came in three major files: train, test, and store. Training has the training data including daily sales and customers over a period of 2 years (1 million data points). , while test data has a period of 6 weeks for particular stores (not all of them, notably). Store has metadata about each store, which presumably affects the sales of a the stores.

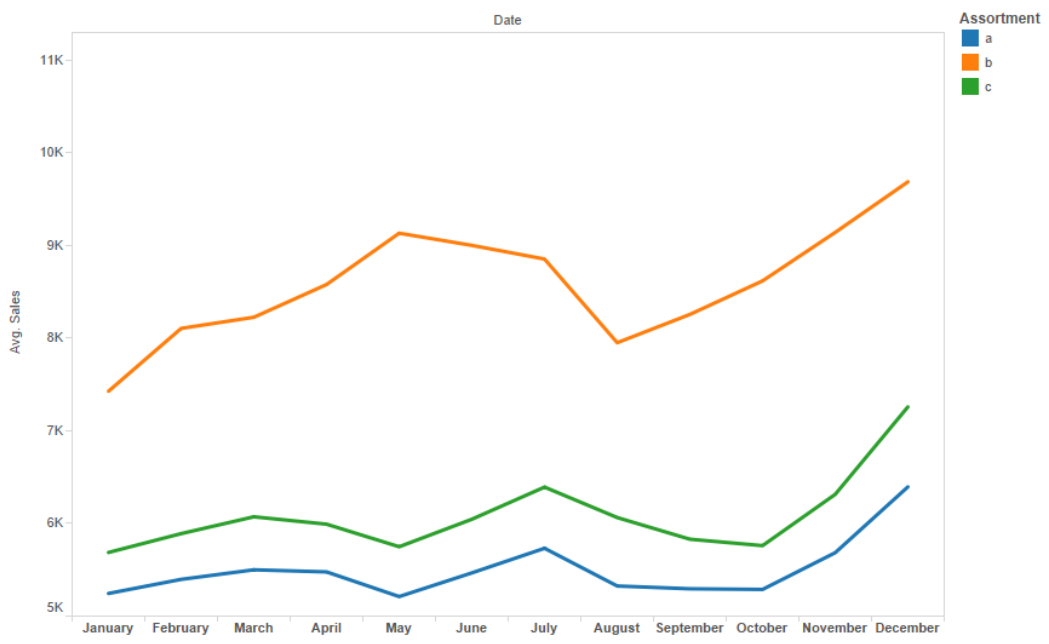
We first merge the store and training data together for ease of analysis, and display general information. We also clean the StateHoliday field for string/int happiness.

```
In [43]: #Partial code taken from (https://www.kaggle.com/mmourafiq/rossmann-store-sales/data-viz/noteb

         train = pd.read_csv('data/train.csv')
         #print(train[:5])
         store = pd.read_csv('data/store.csv')
         #print(store[:5])
         all_data = pd.merge(train, store, on='Store', how='left')
         test = pd.read_csv('data/test.csv')
         all_data['StateHoliday'][all_data['StateHoliday'] == 0 ] = '0'
```

```
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#ind
```

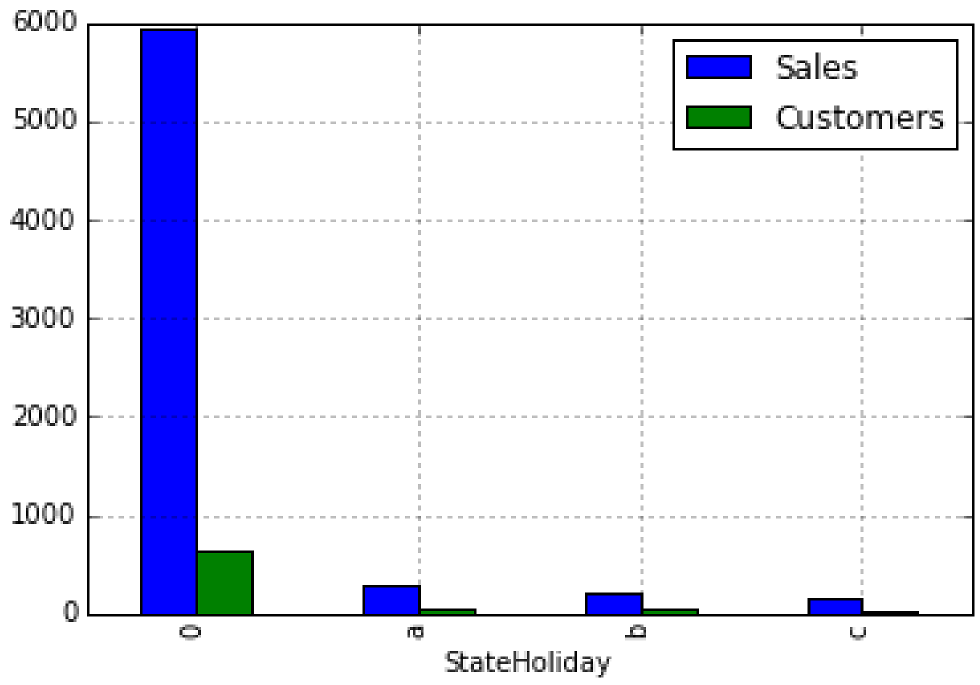
Basic stats of the data. Notice particularly the standard deviation is high for both sales and customers, ipmlyign the data is highly variable and reassuring us that a predictor is necessary.

The assortment of goods that a store has is a major predictor of how well the store will do, as evidenced below. This is notable as the assortments as described on the competition are ordinal variables, but b does significantly better than a and c. A good feature to use to be sure.



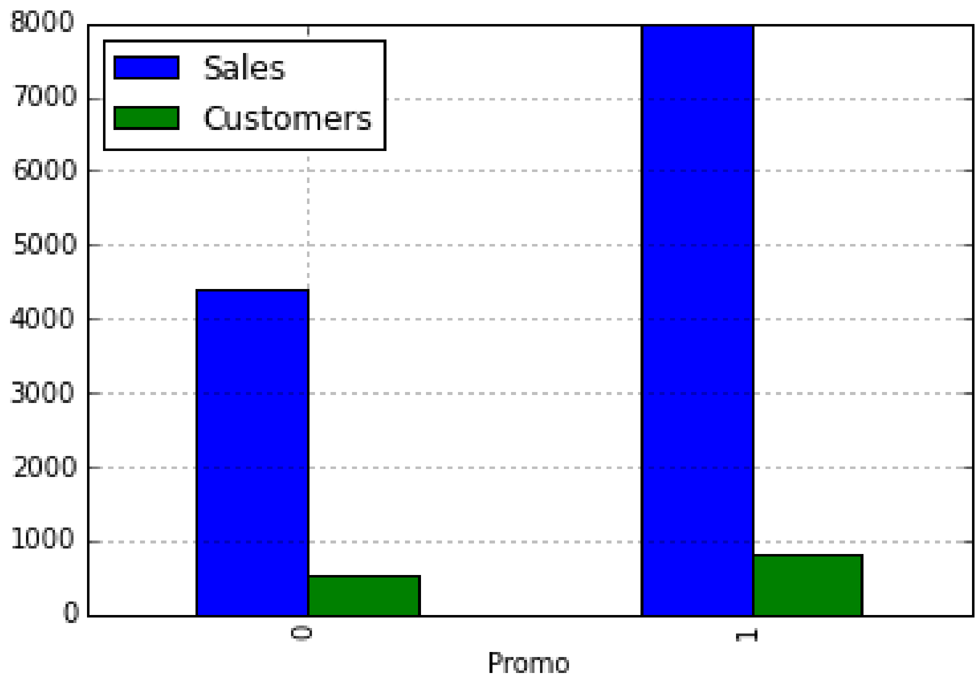
Unsurprisingly, a state holiday strongly affects sales.

```
In [44]: avg_stateholiday = all_data[['Sales', 'Customers', 'StateHoliday']].groupby('StateHoliday').mea
         avg_stateholiday.plot(kind='bar')

Out[44]: <matplotlib.axes._subplots.AxesSubplot at 0x7f950b8f25d0>
```



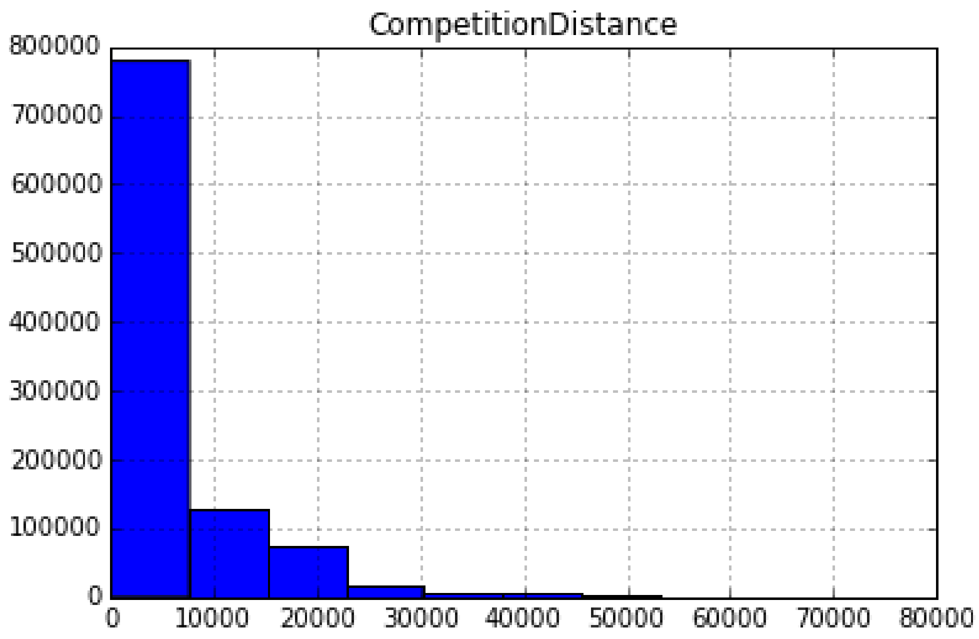
Notably, having a promotion running for a particular day increases sales by quite a large amount, while not increasing customers by nearly the same rate. This implies people are spending more during those days, instead of having simply higher rates of customers.

```
In [45]: avg_promotion = all_data[['Sales', 'Customers', 'Promo']].groupby('Promo').mean()
         avg_promotion.plot(kind='bar')

Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x7f950b992ed0>
```

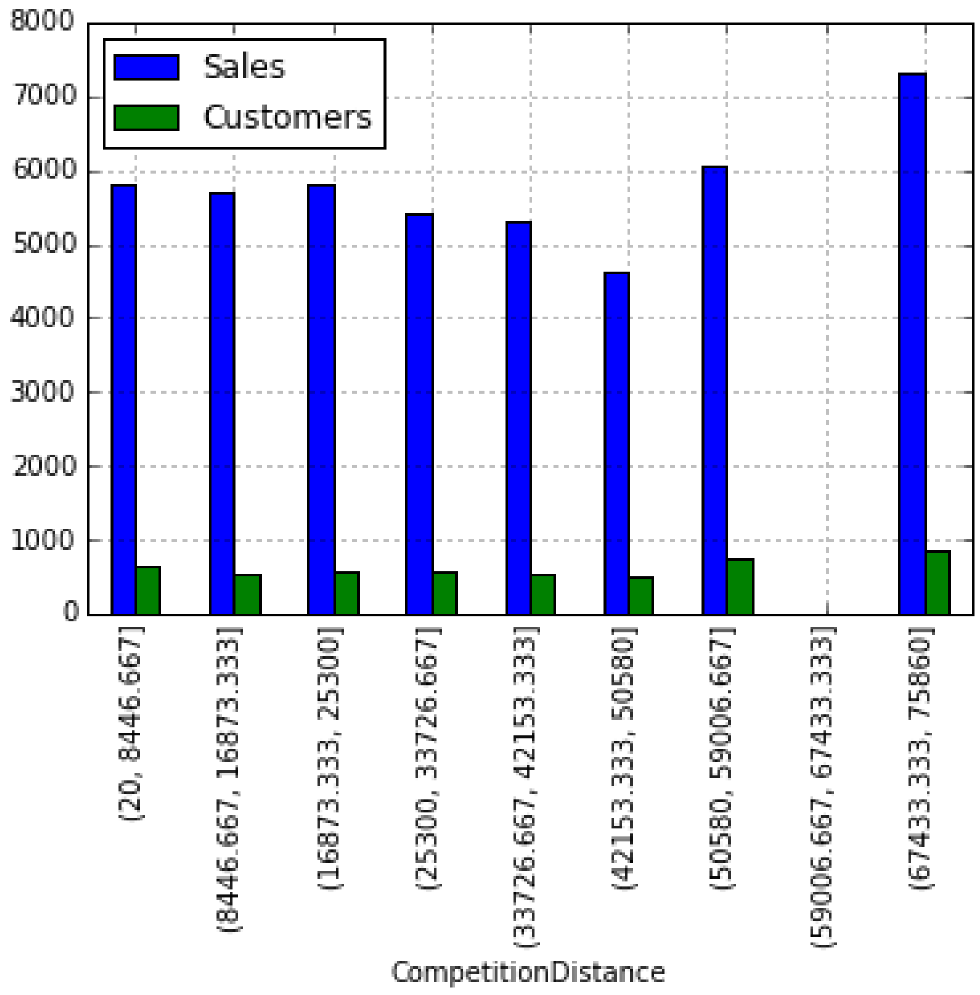


Competition distances are clustered around having close competitors, and in general it doesn't appear to affect sales strongly.

```
In [46]: all_data.hist('CompetitionDistance')

Out[46]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f950b5468d0>]], dtype=object)
```



```
In [47]: # Bin the competition distance with 10 bins
         bins = np.linspace(all_data['CompetitionDistance'].min(), all_data.CompetitionDistance.max(), 

         competition_bins = all_data[['Sales', 'Customers']].groupby(pd.cut(all_data['CompetitionDistanc
         competition_avg = competition_bins.mean()
         competition_avg.plot(kind='bar')

Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x7f950ba39950>
```

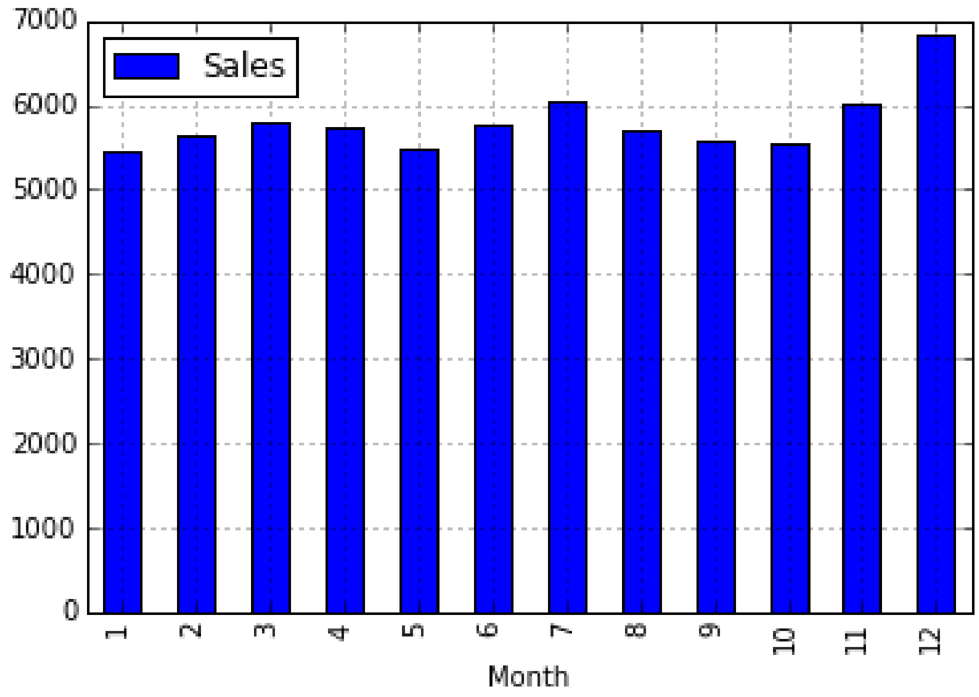


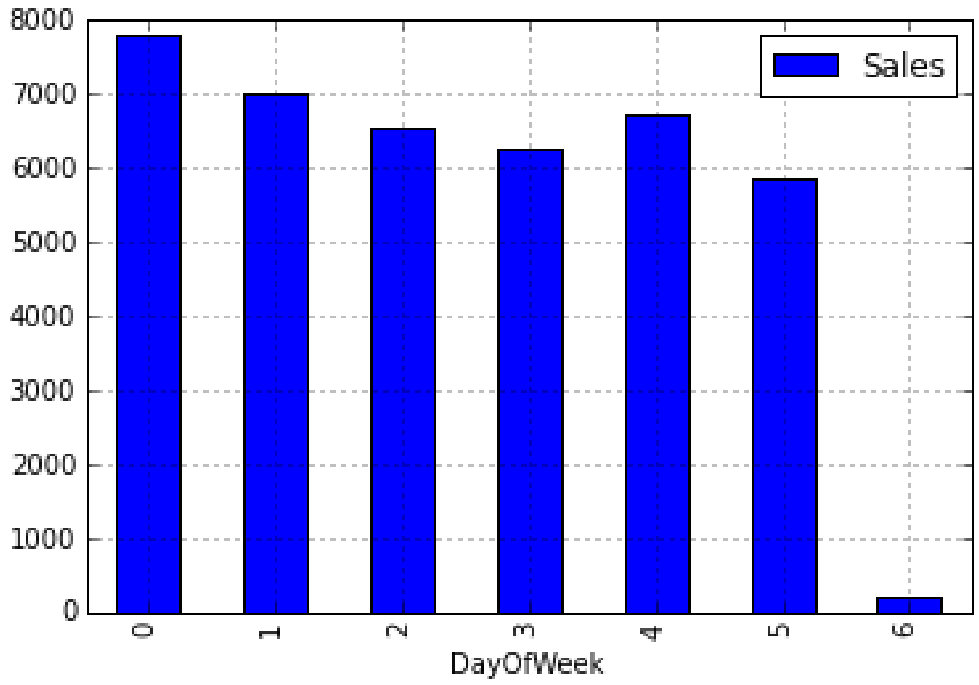
Sundays have almost no sales.

Sales peak in July and December, the peak of summer and Christmas season (The data is from Rossman stores, which are based in Germany so Christmas would be a major holiday.)

```
In [49]: #Done By Neal
         train['Date'] = pd.to_datetime(train['Date'])
         train[:5]['Date'].dt.dayofweek
         train['DayOfWeek'] = train['Date'].dt.dayofweek
         train['Month'] = train['Date'].dt.month
         train['Year'] = train['Date'].dt.year
         avg_month = train[['Sales', 'Month']].groupby('Month').mean()
         avg_month.plot(kind='bar')
         avg_day = train[['Sales', 'DayOfWeek']].groupby('DayOfWeek').mean()
         avg_day.plot(kind='bar')
         #sales by day of week
         sale_dayofweek = pd.pivot_table(train, values='Sales', index=['Year','Store'], columns=['DayOf

         #sales by month
         sale_month = pd.pivot_table(train, values='Sales', index=['Year','Store'], columns=['Month'])
```

```
#sale_dayofweek.plot(kind='box')
#sale_month.plot(kind='box')
```





1.3 Place in Data Science Model

Our model will use a typical data science procedure for batch data analysis including a cleaning phase that feeds into a predictive regression model for sales prediction.

1.4 Next Steps

The next steps mainly include deciding on a particular predictive mode, through both empirical testing and looking at what popular methods for time series regression are used in the field.

Because our goal is to build a sales predictor, visualization of the results isn't a particular goal, though we plan to visualize and discuss major patterns in the data that our analysis finds, such as having a store with a promotion increasing sales by X or discussing the relation of stores to their competitors. We will also include a visualization of our predictor if relevant.

```
In [ ]:
```