You have proposed a project, collected a data set, wrangled and cleaned up the data, and explored it with descriptive + inferential statistics techniques. Now’s the time to take stock of what you’ve learned. The project milestone is an opportunity for you to practice your data story skills. Your milestone will be reached when you produce an early draft of your final Capstone report. This is a slightly longer (3-5 page) draft that should have the following:

An introduction to the problem: What is the problem? Who is the Client? (Feel free to reuse points 1-2 from your proposal document)

A deeper dive into the data set:

What important fields and information does the data set have?

What are its limitations i.e. what are some questions that you cannot answer with this data set?

What kind of cleaning and wrangling did you need to do?

Are there other datasets you can find, use and combine with, to answer the questions that matter?

Any preliminary exploration you’ve performed and your initial findings. Test the hypotheses one at a time. Often, the data story emerges as a result of a sequence of testing hypothesis e.g. You first tested if X was true, and because it wasn't, you tried Y, which turned out to be true.

Based on these findings, what approach are you going to take? How has your approach changed from what you initially proposed, if applicable?

Add your code and milestone report to the github repository. As before, once your mentor has approved your milestone document, please share the github repository URL on the community and ask the community for feedback.

While we require only one milestone report, we encourage you and your mentor to plan multiple milestones, especially for more complex projects

……………………………………………………………………….

Problem- To predict future store sales of 45 Walmart stores based on limited historical data.

Client- Walmart. Walmart can use sales prediction to make Strategic business decisions to maximize profit earnings at the stores by focusing more on stores that will deliver more revenue to the company vs those that won’t.

Deep dive into the data

We have 3 main datasets in this problem

a. stores.csv-This file contains anonymized information about the 45 stores, indicating the type and size of store.

b. train.csv-This is the historical training data, which covers to 2010-02-05 to 2012-11-01.Store - the store number

Dept - the department number

Date - the week

Weekly\_Sales - sales for the given department in the given store

IsHoliday - whether the week is a special holiday week

c. features.csv-This file contains additional data related to the store, department, and regional activity for the given dates. It contains the following fields:

Store - the store number

Date - the week

Temperature - average temperature in the region

Fuel\_Price - cost of fuel in the region

MarkDown1-5 - anonymized data related to promotional markdowns that Walmart is running. MarkDown data is only available after Nov 2011, and is not available for all stores all the time. Any missing value is marked with an NA.

CPI - the consumer price index

Unemployment - the unemployment rate

IsHoliday - whether the week is a special holiday week

Missing Data

Features is the only dataset with missing data. I replaced NaN with NULL

1. There are some column that have categorical values. I intend to replace the categorical v alues with binary values for easy visualization

features['IsHoliday']=features['IsHoliday'].astype(int)

# replace True and False with 1 and 0. True and False are boolean values

1. Now I intend to merge different dataframe based on common column values

# Merge Stores and Features Dataframe

store\_features=pd.merge(features,stores,on='Store',how='inner').

1. Merge features and train DataFrame

features\_train=pd.merge(features,train,on=['Store','Date','IsHoliday'],how='inner')

1. Now I would like to make new column and categorize the datasets in such a way that it’s easy to group.

I created a new columns based on categorizing   various columns Temperature, FuelPrice, Weekly sales,CPI,Storesize.

|  |  |
| --- | --- |
| Condition | TEMP\_CLASS |
| TEMPERATURE < 32 | ‘Freezing’ |
| TEMPERATURE >= 32 AND TEMPERATURE < 64 | ‘Cold’ |
| TEMPERATURE >= 64 AND TEMPERATURE < 79 | ‘Comfortable’ |
| TEMPERATURE >= 79 AND TEMPERATURE < 95 | ‘Hot’ |
| TEMPERATURE > 95 | ‘Extremely Hot’ |

|  |  |
| --- | --- |
| Condition | FUEL\_CLASS |
| FUEL\_PRICE < 2.75 | ‘Low’ |
| FUEL\_PRICE >= 2.75 AND FUEL\_PRICE < 3.12 | ‘Medium’ |
| FUEL\_PRICE > 3.12 | ‘High’ |

|  |  |
| --- | --- |
| Condition | SALES\_CLASS |
| WEEKLY\_SALES <= 0 | ‘Negative’ |
| WEEKLY\_SALES > 0 AND WEEKLY\_SALES <= 10000 | ‘Low’ |
| WEEKLY\_SALES > 10000 AND WEEKLY\_SALES <= 25000 | ‘Medium’ |
| WEEKLY\_SALES > 25000 AND WEEKLY\_SALES <= 100000 | ‘High’ |
| WEEKLY\_SALES > 100000 | ‘Very High’ |

One of the most important column in the existing data is existing Weekly Sales for the 45 stores.

One the data was all cleaned up and ready, I merged features\_train and store\_features into a single dataframe and called it df.

I created several DV plots to visualize the dependency between the independent and the dependent variable.

I created a function ecdf-

def ecdf(p):

n=len(p)

x=np.sort(p)

y=np.arange(1,n+1)/n

return x,y

x\_ver,y\_ver=ecdf(df['Weekly\_Sales'])—since our target is Weekly\_Sales

Then I plotted the ecdf

plt.plot(x\_ver,y\_ver,marker='.',linestyle='none;)

plt.margins(.02)

After this, I tried to calculate the covariance between the independent variables and the predicted variable

np.cov(df['Temperature\_x'],df['Weekly\_Sales'])

x=np.corrcoef(df['Temperature\_x'],df['Weekly\_Sales']) Pearson correlation

x[0,1]

np.cov(df['Fuel\_Price\_x'],df['Weekly\_Sales'])

x=np.corrcoef(df['Fuel\_Price\_x'],df['Weekly\_Sales'])

x[0,1]

Then I tried Linear regression using least square

Y= B0+B1X

The aim is to calculate B0 and B1 which are the intercept and slope

slope,intercept=np.polyfit(df['Temperature\_x'],df['Weekly\_Sales'],1)

Plot

plt.plot(df['Temperature\_x'],df['Weekly\_Sales'],marker='.',linestyle='none')

plt.margins(0.02)

plt.xlabel('Temperature')

plt.ylabel('Weekly\_Sales')

x=np.array([0,100])

y=slope\*x+intercept

plt.plot(x,y)

Plot2

# Plot the Fuel\_Price\_x versus Weekly\_Sales

plt.plot(df['Size'],df['Weekly\_Sales'], marker='.', linestyle='none')

plt.margins(0.02)

\_ = plt.xlabel('Size')

\_ = plt.ylabel('Weekly\_Sales')

# Perform a linear regression using np.polyfit(): a, b

a, b = np.polyfit(df['Size'],df['Weekly\_Sales'],1)

# Print the results to the screen

print('slope =', a, 'slope')

print('intercept =', b, 'intercept')

# Make theoretical line to plot

x = np.array([0,1000000])

y = a \* x + b

# Add regression line to your plot

\_ = plt.plot(x, y)

The resulting linear regression line didn’t pass through the center of the graph, so it tells me they are not related

One last thing I did was to create heatmap that showed me dirty way of finding correlation