# Capstone Project#1- Data Wrangling- Walmart - Store Sales Forecasting

# [Code at https://github.com/shashanknainwal/SB-Capstone-1/blob/master/data%20cleanup\_Capstone1.ipynb](https://github.com/shashanknainwal/SB-Capstone-1/blob/master/data%20cleanup_Capstone1.ipynb)

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

**Intro to datasets**

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**

1. Features is the only dataset with missing data.

features.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 8190 entries, 0 to 8189

Data columns (total 12 columns):

Store 8190 non-null int64

Date 8190 non-null object

Temperature 8190 non-null float64

Fuel\_Price 8190 non-null float64

MarkDown1 4032 non-null float64

MarkDown2 2921 non-null float64

MarkDown3 3613 non-null float64

MarkDown4 3464 non-null float64

MarkDown5 4050 non-null float64

CPI 7605 non-null float64

Unemployment 7605 non-null float64

IsHoliday 8190 non-null bool

1. I then replaced NaN with NULL

features=features.replace('NaN','NULL')

1. There are some column that have categorical values. I intend to replace the categorical values 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 would like to divide Temperature into 4 major categories

Very Cold: < 10 C/< 50 F

Cold: 10-15 C/50-59 F

Cool: 20-25 C/60-77 F

Warm: 26-31 C/78-88 F

Hot: 36-39 C/89-103+ F

def function(temp):

if temp<50:

return 'very cold'

elif (temp>50) & (temp<59):

return 'cold'

elif (temp>60) & (temp<77):

return 'cool'

elif (temp>78) & (temp<88):

return 'warm'

else:

return 'hot'

store\_features['New\_Temp'] = store\_features['Temperature'].apply(function)

features\_train['New\_Temp'] = features\_train['Temperature'].apply(function)

1. now lets try to classify fuel price as Low, medium and high

def function(fuel):

if fuel<2.75:

return 'low'

elif (fuel>2.75) & (fuel<=3.12):

return 'medium'

else:

return 'high'

store\_features['New\_Fuel\_Price'] = store\_features['Fuel\_Price'].apply(function)

def function(fuel):

if fuel<2.75:

return 'low'

elif (fuel>2.75) & (fuel<=3.12):

return 'medium'

else:

return 'high'

features\_train['New\_Fuel\_Price'] = features\_train['Fuel\_Price'].apply(function)

1. Now lets try to classify Weekly sales price as Negative, low, medium, high and very high

Sales column is only available in one df.

def function(sales):

if sales<0:

return 'negative'

elif (sales>0) & (sales<=10000):

return 'low'

elif (sales>10000) & (sales<=25000):

return 'medium'

elif (sales>25000) & (sales<=100000):

return 'high'

else:

return 'very high'

features\_train['New\_Weekly\_Sales'] = features\_train['Weekly\_Sales'].apply(function)

1. To visualize the data from a better perspective, lets add some aditional categorical attributes , including the HOLIDAY (‘Super Bowl’, ‘Labor Day’, ‘Thanksgiving’, ‘Christmas’). The two weeks before each holiday as (‘Before Super Bowl’, ‘Before Labor Day’, ‘Before Thanksgiving’, ‘Before Christmas’).

def function(date):

if date>= '2010-02-01' and date<='2010-02-07':

return 'SuperBowl Day'

elif date>='2011-01-31' and date<='2011-02-06':

return 'SuperBowl Day'

elif date>='2012-01-30' and date<='2012-02-05':

return 'SuperBowl Day'

elif date>='2013-01-28' and date<='2013-02-03':

return 'SuperBowl Day'

elif date>='2010-08-31' and date<='2010-09-06':

return 'Labour Day'

elif date>='2011-08-30' and date<='2011-09-05':

return 'Labour Day'

elif date>='2012-08-28' and date<='2012-09-03':

return 'Labour Day'

elif date>='2013-08-27' and date<='2013-09-02':

return 'Labour Day'

elif date>='2010-11-18' and date<='2010-11-25':

return 'Thanksgiving'

elif date>='2011-11-17' and date<='2011-11-24':

return 'Thanksgiving'

elif date>='2012-11-15' and date<='2012-11-22':

return 'Thanksgiving'

elif date>='2013-11-21' and date<='2013-11-28':

return 'Thanksgiving'

elif date>='2010-12-18' and date<='2010-12-25':

return 'Christmas'

elif date>='2011-12-18' and date<='2011-12-25':

return 'Christmas'

elif date>='2012-12-18' and date<='2012-12-25':

return 'Christmas'

elif date>='2013-12-18' and date<='2013-12-25':

return 'Christmas'

elif date>= '2010-01-16' and date<'2010-02-01':

return 'Before SuperBowl Day'

elif date>='2011-01-15' and date<'2011-01-31':

return 'Before SuperBowl Day'

elif date>='2012-01-14' and date<'2012-01-30':

return 'Before SuperBowl Day'

elif date>='2013-01-26' and date<'2013-02-03':

return 'Before SuperBowl Day'

elif date>='2010-07-16' and date<'2010-08-31':

return 'Before Labour Day'

elif date>='2011-07-15' and date<'2011-08-30':

return 'Before Labour Day'

elif date>='2012-08-14' and date<'2012-08-28':

return 'Before Labour Day'

elif date>='2013-08-13' and date<'2013-08-27':

return 'Before Labour Day'

elif date>='2010-11-04' and date<'2010-11-18':

return 'Before Thanksgiving'

elif date>='2011-11-03' and date<'2011-11-17':

return 'Before Thanksgiving'

elif date>='2012-11-01' and date<'2012-11-15':

return 'Before Thanksgiving'

elif date>='2013-11-07' and date<'2013-11-21':

return 'Before Thanksgiving'

elif date>='2010-12-04' and date<'2010-12-18':

return 'Before Christmas'

elif date>='2011-12-04' and date<'2011-12-18':

return 'Before Christmas'

elif date>='2012-12-04' and date<'2012-12-18':

return 'Before Christmas'

elif date>='2013-12-04' and date<'2013-12-18':

return 'Before Christmas'

else:

return 'Usual Day'

store\_features['Holiday\_Type'] = store\_features['Date'].apply(function)

features\_train['Holiday\_Type']=features\_train['Date'].apply(function)

1. Last but not least lets unemployment and CPI into ‘Low’, ‘Medium’ and ‘High’. Store size was categorized to ‘Small’, ‘Medium’ and ‘Large’.

def function(CPI):

if CPI=='NULL':

return np.NaN

elif CPI<159:

return 'low'

elif CPI >=159 and CPI <192:

return 'medium'

elif CPI >=192:

return 'high'

store\_features['new\_CPI'] = store\_features['CPI'].apply(function)

features\_train['new\_CPI']=features\_train['CPI'].apply(function)

def function(Unemployment):

if Unemployment=='NULL':

return np.NaN

elif Unemployment<7:

return 'low'

elif Unemployment >=7 and Unemployment <11:

return 'medium'

elif Unemployment >=11:

return 'high'

store\_features['new\_unemployment'] = store\_features['Unemployment'].apply(function)

features\_train['new\_unemployment']=features\_train['Unemployment'].apply(function)

def function(Weekly\_Sales):

if Weekly\_Sales<10000:

return 'small'

elif Weekly\_Sales >=10000 and Weekly\_Sales <20000:

return 'medium'

elif Weekly\_Sales >=20000:

return 'large'

features\_train['new\_weekly\_sales']=features\_train['Weekly\_Sales'].apply(function)