Walmart

December 11, 2018

1 Walmart Sales Forecasting

Yuha Yi University of Idaho Stats 517 - Statistical Learning and Predictive Modeling

Introduction The dataset contains historical sales data from 45 Walmart stores. Machine learning will be used to predict the sales of each stores using historical markdown data. Additionally, the effects of markdowns on certain holiday weeks that play a large part in weekly sales within the stores will be observed. Some important holidays include the super bowl, Christmas, and Thanksgiving/Black Friday to name a few. The research of this data observes how Walmart properly prepares during critical times of the year to maximize profit while achieving maximum customer satisfaction.

Variable Description The features (8191 x 12) file has some of the variables below Dept – the department number within the store Date – the week IsHoliday – whether the week is a special holiday week Temperature – The average temperature in the region Fuel_Price – Cost of fuel in the region CPI – The consumer price index Unemployment – The unemployment rate MarkDown(1-5) – Anonymized data related to promotional price markdowns from Walmart. Train.csv (422k x 5) contains Historical training data from 5/2/2010 to 26/10/2011, containing columns for Store, Dept, Date, Weekly_Sales and IsHoliday Test.csv (115k x 4) contains test data for 2/11/2012 to 26/7/2013, containing columns for Store, Department, Date and IsHoliday Stores file (45 x 3) has the 45 stores, the store size, and store type where Store type is an arbitrary value for the store type constiting of type A, B, and C

1.0.1 Process

The problems that this project attempts to solve is a real world application that all businesses look for. For regression, we seek the predicted weekly sales. The usage of this information can be used to allocate the right number of workers on busy weeks, know how much purchase to product for the holidays, and estimate overall income for the stores. This let us know which markdown selection is best during which time of year. The first half of the analysis uses regression to answer these questions For unsupervised learning (done in R) clustering is done for the stores (143 weeks x 45 stores) and an arima model is used for forecasting. To do this, an autocorrelation is ran through the store matrix to measure (in distance) the similarity of each store into a time series to view the similarities of the stores. They are then clustered into four individual time series. Association was attempted in this project however it did not end up successfully running, this it is not shown. However, if done properly it has the ability to tell us information such as given a certain department and week, how likely is it to have a markdown sale. This can give valuable information on when to purchase an item for the best price.

```
In [1]: #!pip install pandas_profiling
```

```
In [2]: import numpy as np
        import pandas as pd
        import plotly
        import math
        from numpy import *
        import gc
        import warnings
        import os

        from scipy.misc import imread
        from scipy import sparse
        import scipy.stats as ss
        import matplotlib.pylab as plt
        import seaborn as sns
        import pandas_profiling
        import seaborn as sns
        import plotly.tools as tls
        import plotly.graph_objs as go

        from datetime import datetime

        import statsmodels.api as sm
        from sklearn import metrics
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.cross_validation import train_test_split
        from sklearn.linear_model import LinearRegression
        from sklearn.ensemble import RandomForestRegressor
        from sklearn.tree import DecisionTreeRegressor
        from sklearn.ensemble import AdaBoostRegressor
        from sklearn.ensemble import GradientBoostingRegressor
        from sklearn.svm import SVC, LinearSVC
        from sklearn.naive_bayes import MultinomialNB
        from sklearn.naive_bayes import GaussianNB
        from sklearn.metrics import mean_squared_error as mse
        from sklearn.metrics import mean_absolute_error, mean_squared_error
        from sklearn.neural_network import MLPClassifier
        from sklearn.metrics import roc_curve
        from sklearn.metrics import roc_auc_score
        from sklearn.decomposition import PCA as sklearnPCA
        from sklearn.decomposition import PCA
        from sklearn.cluster import KMeans
        from sklearn.cluster import SpectralClustering
        from sklearn.mixture import GMM
        from sklearn.cluster import AgglomerativeClustering
```

```
C:\Users\yiyuh\Anaconda3\lib\site-packages\pandas_profiling\plot.py:15: UserWarning:


This call to matplotlib.use() has no effect because the backend has already
been chosen; matplotlib.use() must be called *before* pylab, matplotlib.pyplot,
or matplotlib.backends is imported for the first time.

The backend was *originally* set to 'module://ipykernel.pylab.backend_inline' by the following
  File "C:\Users\yiyuh\Anaconda3\lib\runpy.py", line 193, in _run_module_as_main
    "__main__", mod_spec)
  File "C:\Users\yiyuh\Anaconda3\lib\runpy.py", line 85, in _run_code
    exec(code, run_globals)
  File "C:\Users\yiyuh\Anaconda3\lib\site-packages\ipykernel_launcher.py", line 16, in <module>
    app.launch_new_instance()
  File "C:\Users\yiyuh\Anaconda3\lib\site-packages\traitlets\config\application.py", line 658,
    app.start()
  File "C:\Users\yiyuh\Anaconda3\lib\site-packages\ipykernel\kernelapp.py", line 486, in start
    self.io_loop.start()
  File "C:\Users\yiyuh\Anaconda3\lib\site-packages\tornado\platform\asyncio.py", line 127, in s
    self.asyncio_loop.run_forever()
  File "C:\Users\yiyuh\Anaconda3\lib\asyncio\base_events.py", line 422, in run_forever
    self._run_once()
  File "C:\Users\yiyuh\Anaconda3\lib\asyncio\base_events.py", line 1432, in _run_once
    handle._run()
  File "C:\Users\yiyuh\Anaconda3\lib\asyncio\events.py", line 145, in _run
    self._callback(*self._args)
  File "C:\Users\yiyuh\Anaconda3\lib\site-packages\tornado\platform\asyncio.py", line 117, in
    handler_func(fileobj, events)
  File "C:\Users\yiyuh\Anaconda3\lib\site-packages\tornado\stack_context.py", line 276, in null
    return fn(*args, **kwargs)
  File "C:\Users\yiyuh\Anaconda3\lib\site-packages\zmq\eventloop\zmqstream.py", line 450, in _h
    self._handle_recv()
  File "C:\Users\yiyuh\Anaconda3\lib\site-packages\zmq\eventloop\zmqstream.py", line 480, in _h
    self._run_callback(callback, msg)
  File "C:\Users\yiyuh\Anaconda3\lib\site-packages\zmq\eventloop\zmqstream.py", line 432, in _r
    callback(*args, **kwargs)
  File "C:\Users\yiyuh\Anaconda3\lib\site-packages\tornado\stack_context.py", line 276, in null
    return fn(*args, **kwargs)
  File "C:\Users\yiyuh\Anaconda3\lib\site-packages\ipykernel\kernelbase.py", line 283, in dispa
    return self.dispatch_shell(stream, msg)
  File "C:\Users\yiyuh\Anaconda3\lib\site-packages\ipykernel\kernelbase.py", line 233, in dispa
    handler(stream, idents, msg)
  File "C:\Users\yiyuh\Anaconda3\lib\site-packages\ipykernel\kernelbase.py", line 399, in execu
    user_expressions, allow_stdin)
  File "C:\Users\yiyuh\Anaconda3\lib\site-packages\ipykernel\ipkernel.py", line 208, in do_exec
    res = shell.run_cell(code, store_history=store_history, silent=silent)
  File "C:\Users\yiyuh\Anaconda3\lib\site-packages\ipykernel\zmqshell.py", line 537, in run_cel
    return super(ZMQInteractiveShell, self).run_cell(*args, **kwargs)
```

```
  File "C:\Users\yiyuh\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py", line 2662
    raw_cell, store_history, silent, shell_futures)
  File "C:\Users\yiyuh\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py", line 2785
    interactivity=interactivity, compiler=compiler, result=result)
  File "C:\Users\yiyuh\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py", line 2903
    if self.run_code(code, result):
  File "C:\Users\yiyuh\Anaconda3\lib\site-packages\IPython\core\interactiveshell.py", line 2963
    exec(code_obj, self.user_global_ns, self.user_ns)
  File "<ipython-input-2-5243fc62bc76>", line 13, in <module>
    import matplotlib.pylab as plt
  File "C:\Users\yiyuh\Anaconda3\lib\site-packages\matplotlib\pylab.py", line 252, in <module>
    from matplotlib import cbook, mlab, pyplot as plt
  File "C:\Users\yiyuh\Anaconda3\lib\site-packages\matplotlib\pyplot.py", line 71, in <module>
    from matplotlib.backends import pylab_setup
  File "C:\Users\yiyuh\Anaconda3\lib\site-packages\matplotlib\backends\__init__.py", line 16, i
    line for line in traceback.format_stack()



C:\Users\yiyuh\Anaconda3\lib\site-packages\sklearn\cross_validation.py:41: DeprecationWarning:

This module was deprecated in version 0.18 in favor of the model_selection module into which al
```

```
In [3]: pwd

Out[3]: 'C:\\Users\\yiyuh\\Documents\\College\\Fall 2018\\Stat 517 - Machine Learning\\Final Pi

In [4]: train = pd.read_csv("C:\\Users\\yiyuh\\Documents\\College\\Fall 2018\\Stat 517 - Machin
        feature = pd.read_csv("C:\\Users\\yiyuh\\Documents\\College\\Fall 2018\\Stat 517 - Mach
        test= pd.read_csv("C:\\Users\\yiyuh\\Documents\\College\\Fall 2018\\Stat 517 - Machine
        stores = pd.read_csv("C:\\Users\\yiyuh\\Documents\\College\\Fall 2018\\Stat 517 - Machi
        writer=pd.ExcelWriter('Walmart Store Sales Prediction output.xlsx', engine='xlsxwriter

In [5]: #Merge the datasets to include the store numbers into the training and testing
        train_bt = pd.merge(train,stores)
        train = pd.merge(train_bt,feature)
        #
        test_bt = pd.merge(test,stores)
        test= pd.merge(test_bt,feature)

In [6]: train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 421570 entries, 0 to 421569
Data columns (total 16 columns):
Store           421570 non-null int64
Dept            421570 non-null int64
```

```
Date             421570 non-null object
Weekly_Sales     421570 non-null float64
IsHoliday        421570 non-null bool
Type             421570 non-null object
Size             421570 non-null int64
Temperature      421570 non-null float64
Fuel_Price       421570 non-null float64
MarkDown1        150681 non-null float64
MarkDown2        111248 non-null float64
MarkDown3        137091 non-null float64
MarkDown4        134967 non-null float64
MarkDown5        151432 non-null float64
CPI              421570 non-null float64
Unemployment     421570 non-null float64
dtypes: bool(1), float64(10), int64(3), object(2)
memory usage: 51.9+ MB
```

In [7]:
```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 115064 entries, 0 to 115063
Data columns (total 15 columns):
Store            115064 non-null int64
Dept             115064 non-null int64
Date             115064 non-null object
IsHoliday        115064 non-null bool
Type             115064 non-null object
Size             115064 non-null int64
Temperature      115064 non-null float64
Fuel_Price       115064 non-null float64
MarkDown1        114915 non-null float64
MarkDown2        86437 non-null float64
MarkDown3        105235 non-null float64
MarkDown4        102176 non-null float64
MarkDown5        115064 non-null float64
CPI              76902 non-null float64
Unemployment     76902 non-null float64
dtypes: bool(1), float64(9), int64(3), object(2)
memory usage: 13.3+ MB
```

In [8]:
```
numeric_var_train=[key for key in dict(train.dtypes) if dict(train.dtypes)[key] in ['fl
cat_var_train=[key for key in dict(train.dtypes) if dict(train.dtypes)[key] in ['objec
# Train Numerical Data
train_num=train[numeric_var_train]

# Train Categorical Data
```

```python
train_cat=train[cat_var_train]
print (numeric_var_train)
print (cat_var_train)
```

```
['Store', 'Dept', 'Weekly_Sales', 'Size', 'Temperature', 'Fuel_Price', 'MarkDown1', 'MarkDown2
['Date', 'Type']
```

In [9]:
```python
#Creating a total summary of the training dataset all into one function
def var_summary(x):
    return pd.Series([x.count(), x.isnull().sum(), x.sum(), x.mean(), x.median(),  x.s
                     index=['N', 'NMISS', 'SUM', 'MEAN','MEDIAN', 'STD', 'VAR', 'MIN', 'P

num_summary=train_num.apply(lambda x: var_summary(x)).T
num_summary.to_excel(writer,'Numeric_variable Summary',index=True)
num_summary
```

Out[9]:

	N	NMISS	SUM	MEAN	MEDIAN	\
Store	421570.0	0.0	9.359084e+06	22.200546	22.00000	
Dept	421570.0	0.0	1.865882e+07	44.260317	37.00000	
Weekly_Sales	421570.0	0.0	6.737219e+09	15981.258123	7612.03000	
Size	421570.0	0.0	5.764039e+10	136727.915739	140167.00000	
Temperature	421570.0	0.0	2.533217e+07	60.090059	62.09000	
Fuel_Price	421570.0	0.0	1.416908e+06	3.361027	3.45200	
MarkDown1	150681.0	270889.0	1.091898e+09	7246.420196	5347.45000	
MarkDown2	111248.0	310322.0	3.709708e+08	3334.628621	192.00000	
MarkDown3	137091.0	284479.0	1.973317e+08	1439.421384	24.60000	
MarkDown4	134967.0	286603.0	4.566161e+08	3383.168256	1481.31000	
MarkDown5	151432.0	270138.0	7.009750e+08	4628.975079	3359.45000	
CPI	421570.0	0.0	7.217360e+07	171.201947	182.31878	
Unemployment	421570.0	0.0	3.355819e+06	7.960289	7.86600	

	STD	VAR	MIN	P1	\
Store	12.785297	1.634638e+02	1.000	1.000000	
Dept	30.492054	9.297654e+02	1.000	1.000000	
Weekly_Sales	22711.183519	5.157979e+08	-4988.940	5.000000	
Size	60980.583328	3.718632e+09	34875.000	34875.000000	
Temperature	18.447931	3.403262e+02	-2.060	18.300000	
Fuel_Price	0.458515	2.102356e-01	2.472	2.565000	
MarkDown1	8291.221345	6.874435e+07	0.270	17.760000	
MarkDown2	9475.357325	8.978240e+07	-265.760	-0.990000	
MarkDown3	9623.078290	9.260364e+07	-29.100	0.060000	
MarkDown4	6292.384031	3.959410e+07	0.220	3.970000	
MarkDown5	5962.887455	3.555603e+07	135.160	351.970000	
CPI	39.159276	1.533449e+03	126.064	126.111903	
Unemployment	1.863296	3.471872e+00	3.879	4.156000	

	P5	P10	P25	P50	\

Store	3.000000	5.000000	11.000000	22.00000
Dept	4.000000	7.000000	18.000000	37.00000
Weekly_Sales	59.974500	291.097000	2079.650000	7612.03000
Size	39690.000000	39910.000000	93638.000000	140167.00000
Temperature	27.310000	33.980000	46.680000	62.09000
Fuel_Price	2.653000	2.720000	2.933000	3.45200
MarkDown1	149.190000	375.200000	2240.270000	5347.45000
MarkDown2	1.950000	6.980000	41.600000	192.00000
MarkDown3	0.650000	1.650000	5.080000	24.60000
MarkDown4	28.760000	108.710000	504.220000	1481.31000
MarkDown5	715.520000	1070.830000	1878.440000	3359.45000
CPI	126.496258	128.823806	132.022667	182.31878
Unemployment	5.326000	5.965000	6.891000	7.86600

	P75	P90	P95	P99 \
Store	33.000000	40.000000	43.000000	45.000000
Dept	74.000000	92.000000	95.000000	98.000000
Weekly_Sales	20205.852500	42845.673000	61201.951000	106479.586000
Size	202505.000000	204184.000000	206302.000000	219622.000000
Temperature	74.280000	83.580000	87.270000	92.810000
Fuel_Price	3.738000	3.917000	4.029000	4.202000
MarkDown1	9210.900000	15282.470000	21801.350000	41524.030000
MarkDown2	1926.940000	8549.740000	16497.470000	50366.600000
MarkDown3	103.990000	400.090000	1059.900000	63143.290000
MarkDown4	3595.040000	7871.420000	12645.960000	35785.260000
MarkDown5	5563.800000	8337.700000	11269.240000	27754.230000
CPI	212.416993	219.444244	221.941558	225.473509
Unemployment	8.572000	9.816000	12.187000	14.180000

	MAX
Store	45.000000
Dept	99.000000
Weekly_Sales	693099.360000
Size	219622.000000
Temperature	100.140000
Fuel_Price	4.468000
MarkDown1	88646.760000
MarkDown2	104519.540000
MarkDown3	141630.610000
MarkDown4	67474.850000
MarkDown5	108519.280000
CPI	227.232807
Unemployment	14.313000

Data Explanation: Four files are given as the following, features (8191 x 12), stores (45 x 3), test (115k x 4), and train (422k x 5). The stores file contains the store number, size, and type. The store number is a unique identifier to differentiate all the stores, the size variable is measured in square feet, and the type indicates an arbitrary value of the store type consisting of type A, B, or C. The training file contains the store number, department number, the date (which week it is), the weekly sales, and a true/false variable of whether or not the week is considered as a holiday. The test file is identical to the train file without the weekly sales. The weekly sales and individual department sales must be predicted for each week for this file. The features file contains the following variables:

Store - a unique identifier for every Walmart store Dept – the department number within the store Date – the week IsHoliday – whether the week is a special holiday week Temperature – The average temperature in the region Fuel_Price – Cost of fuel in the region CPI – The consumer price index Unemployment – The unemployment rate Weekly_Sales - Sales for the given department within it's respective store MarkDown(1-5) – Anonymized data related to promotional price markdowns from Walmart.

In [10]:
```
def cat_summary(x):
    return pd.Series([x.count(), x.isnull().sum(), x.value_counts()],
                    index=['N', 'NMISS', 'ColumnsNames'])

cat_summary=train_cat.apply(lambda x: cat_summary(x))
cat_summary
```

Out[10]:
```
                                                                    Date  \
N                                                                 421570
NMISS                                                                  0
ColumnsNames  2011-12-23    3027
2011-11-25    3021
2011-12-...

                                                                    Type
N                                                                 421570
NMISS                                                                  0
ColumnsNames  A    215478
B    163495
C     42597
Name: Type...
```

In [11]:
```
#Similarly to above, but for the testing dataset
numeric_var_test=[key for key in dict(test.dtypes) if dict(test.dtypes)[key] in ['floa
cat_var_test=[key for key in dict(test.dtypes) if dict(test.dtypes)[key] in ['object']
# Train Numerical Data
test_num=test[numeric_var_test]

# Train Categorical Data
test_cat=test[cat_var_test]
print (numeric_var_test)
print (cat_var_test)
```

```
['Store', 'Dept', 'Size', 'Temperature', 'Fuel_Price', 'MarkDown1', 'MarkDown2', 'MarkDown3',
['Date', 'Type']
```

In [12]:
```python
num_summary=test_num.apply(lambda x: var_summary(x)).T
#num_summary.to_excel(writer,'Numeric_variable Summary',index=True)
num_summary.head()
```

Out[12]:

	N	NMISS	SUM	MEAN	MEDIAN
Store	115064.0	0.0	2.558817e+06	22.238207	22.000
Dept	115064.0	0.0	5.101883e+06	44.339524	37.000
Size	115064.0	0.0	1.570597e+10	136497.688921	140167.000
Temperature	115064.0	0.0	6.206760e+06	53.941804	54.470
Fuel_Price	115064.0	0.0	4.121070e+05	3.581546	3.606

	STD	VAR	MIN	P1	P5
Store	12.809930	1.640943e+02	1.000	1.000	3.000
Dept	30.656410	9.398155e+02	1.000	1.000	4.000
Size	61106.926438	3.734056e+09	34875.000	34875.000	39690.000
Temperature	18.724153	3.505939e+02	-7.290	11.440	23.980
Fuel_Price	0.239442	5.733244e-02	2.872	2.957	3.161

	P10	P25	P50	P75	P90
Store	5.000	11.000	22.000	33.000	40.000
Dept	7.000	18.000	37.000	74.000	92.000
Size	39910.000	93638.000	140167.000	202505.000	204184.000
Temperature	29.970	39.820	54.470	67.350	79.480
Fuel_Price	3.227	3.431	3.606	3.766	3.866

	P95	P99	MAX
Store	43.000	45.000	45.000
Dept	95.000	98.000	99.000
Size	206302.000	219622.000	219622.000
Temperature	83.820	92.140	101.950
Fuel_Price	3.951	4.079	4.125

In [13]:
```python
pandas_profiling.ProfileReport(train)
```

Out[13]: <pandas_profiling.ProfileReport at 0x2540a54a1d0>

In [14]:
```python
df = pd.concat([train,test],axis=0) # Join train and test
```

```
C:\Users\yiyuh\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: FutureWarning:

Sorting because non-concatenation axis is not aligned. A future version
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=True'.

To retain the current behavior and silence the warning, pass sort=False
```

```
In [15]: df.describe()
```

```
Out[15]:                CPI           Dept     Fuel_Price      MarkDown1  \
         count  498472.000000  536634.000000  536634.000000  265596.000000
         mean      172.090481      44.277301       3.408310    7438.004144
         std        39.542149      30.527358       0.430861    9411.341379
         min       126.064000       1.000000       2.472000   -2781.450000
         25%       132.521867      18.000000       3.041000    2114.640000
         50%       182.442420      37.000000       3.523000    5126.540000
         75%       213.748126      74.000000       3.744000    9303.850000
         max       228.976456      99.000000       4.468000  103184.980000

                     MarkDown2      MarkDown3      MarkDown4      MarkDown5  \
         count  197685.000000  242326.000000  237143.000000  266496.000000
         mean     3509.274827    1857.913525    3371.556866    4324.021158
         std      8992.047197   11616.143274    6872.281734   13549.262124
         min      -265.760000    -179.260000       0.220000    -185.170000
         25%        72.500000       7.220000     336.240000    1570.112500
         50%       385.310000      40.760000    1239.040000    2870.910000
         75%      2392.390000     174.260000    3397.080000    5012.220000
         max    104519.540000  149483.310000   67474.850000  771448.100000

                          Size          Store    Temperature   Unemployment  \
         count  536634.000000  536634.000000  536634.000000  498472.000000
         mean   136678.550960      22.208621      58.771762       7.791888
         std     61007.711799      12.790580      18.678716       1.865076
         min     34875.000000       1.000000      -7.290000       3.684000
         25%     93638.000000      11.000000      45.250000       6.623000
         50%    140167.000000      22.000000      60.060000       7.795000
         75%    202505.000000      33.000000      73.230000       8.549000
         max    219622.000000      45.000000     101.950000      14.313000

                  Weekly_Sales
         count  421570.000000
         mean    15981.258123
         std     22711.183519
         min     -4988.940000
         25%      2079.650000
         50%      7612.030000
         75%     20205.852500
         max    693099.360000
```

```
In [16]: train_corr=pd.DataFrame(train.corr())
         train_corr.to_excel(writer,'Train_Data Corr',index=True)
         train_corr.head()
```

```
Out[16]:                  Store      Dept  Weekly_Sales  IsHoliday      Size  \
         Store         1.000000  0.024004     -0.085195  -0.000548 -0.182881
```

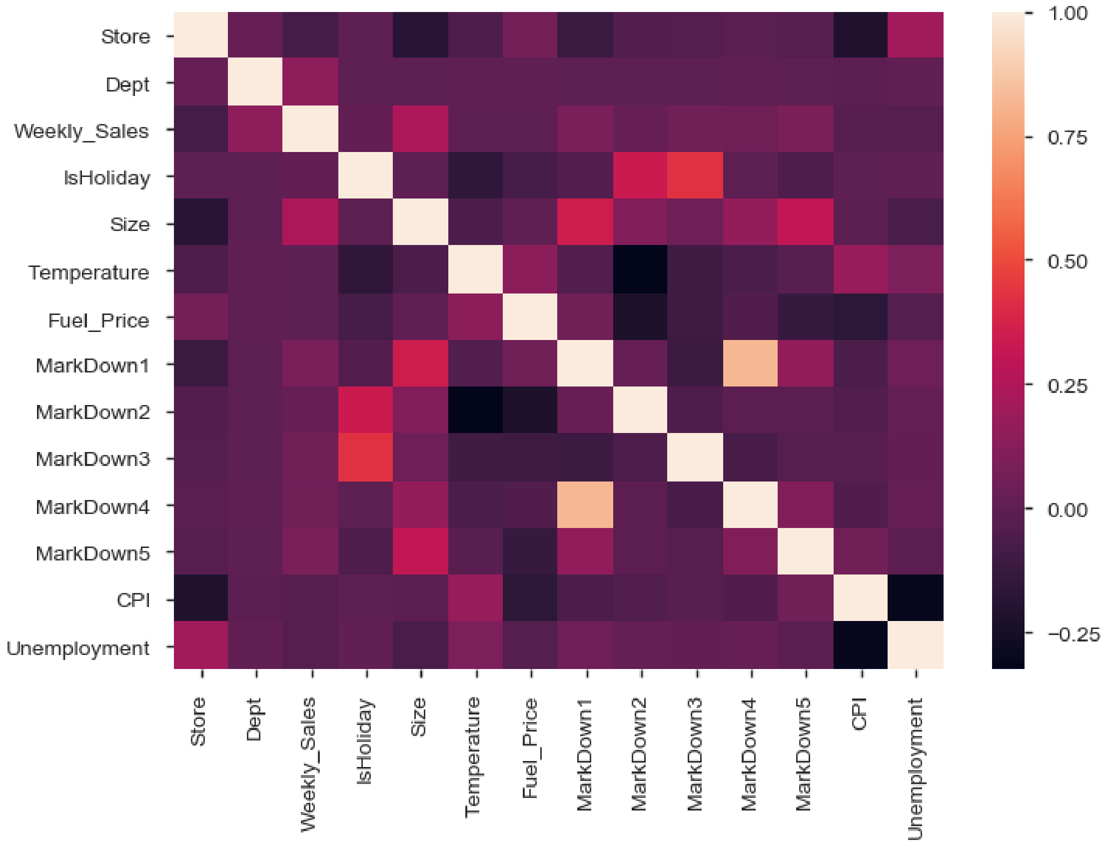
```
Dept          0.024004  1.000000      0.148032   0.000916 -0.002966
Weekly_Sales -0.085195  0.148032      1.000000   0.012774  0.243828
IsHoliday    -0.000548  0.000916      0.012774   1.000000  0.000593
Size         -0.182881 -0.002966      0.243828   0.000593  1.000000

              Temperature  Fuel_Price  MarkDown1  MarkDown2  MarkDown3  \
Store           -0.050097    0.065290  -0.119588  -0.035173  -0.031556
Dept             0.004437    0.003572  -0.002426   0.000290   0.001784
Weekly_Sales    -0.002312   -0.000120   0.085251   0.024130   0.060385
IsHoliday       -0.155949   -0.078281  -0.035586   0.334818   0.427960
Size            -0.058313    0.003361   0.345673   0.108827   0.048913

              MarkDown4  MarkDown5       CPI  Unemployment
Store         -0.009941  -0.026634 -0.211088      0.208552
Dept           0.004257   0.000109 -0.007477      0.007837
Weekly_Sales   0.045414   0.090362 -0.020921     -0.025864
IsHoliday     -0.000562  -0.053719 -0.001944      0.010460
Size           0.168196   0.304575 -0.003314     -0.068238
```

In [17]:
```python
test_corr=pd.DataFrame(test.corr())
#test_corr.to_excel(writer,'Test_Data Corr',index=True)
test_corr.head()
```

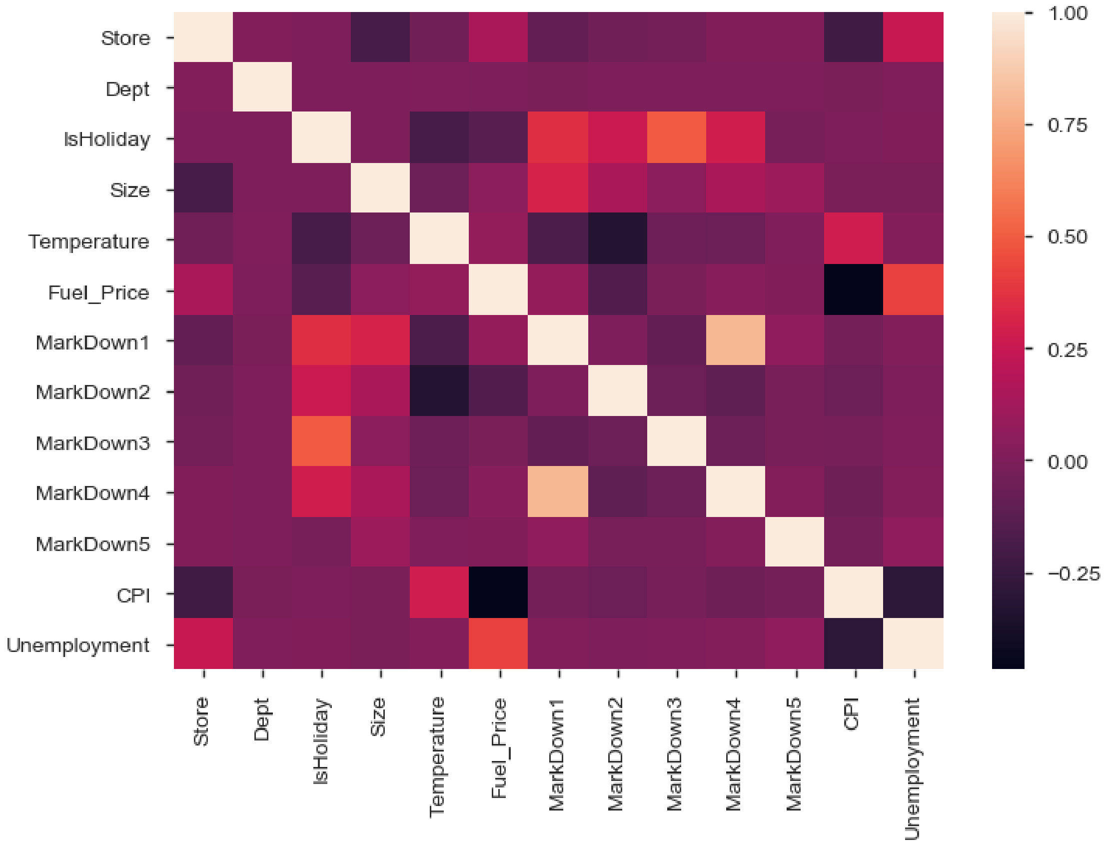
Out[17]:
```
                Store      Dept  IsHoliday      Size  Temperature  Fuel_Price  \
Store        1.000000  0.019627  -0.001166 -0.186845    -0.043495    0.153425
Dept         0.019627  1.000000   0.001249  0.001502     0.003970    0.000554
IsHoliday   -0.001166  0.001249   1.000000 -0.000443    -0.187428   -0.126443
Size        -0.186845  0.001502  -0.000443  1.000000    -0.061256    0.055088
Temperature -0.043495  0.003970  -0.187428 -0.061256     1.000000    0.073938

             MarkDown1  MarkDown2  MarkDown3  MarkDown4  MarkDown5       CPI  \
Store        -0.091707  -0.041370  -0.025177   0.010331   0.010419 -0.214872
Dept         -0.002353   0.001292   0.000247   0.002510   0.000776 -0.006336
IsHoliday     0.355257   0.265402   0.496062   0.289700  -0.019386 -0.001475
Size          0.309614   0.157526   0.050088   0.155448   0.103681 -0.002916
Temperature  -0.168899  -0.324280  -0.049771  -0.059583   0.003937  0.280861

             Unemployment
Store            0.250321
Dept             0.004087
IsHoliday        0.010288
Size            -0.001988
Temperature      0.022136
```

In [18]:
```python
# visualize correlation matrix in Seaborn using a heatmap
sns.heatmap(train.corr())
```

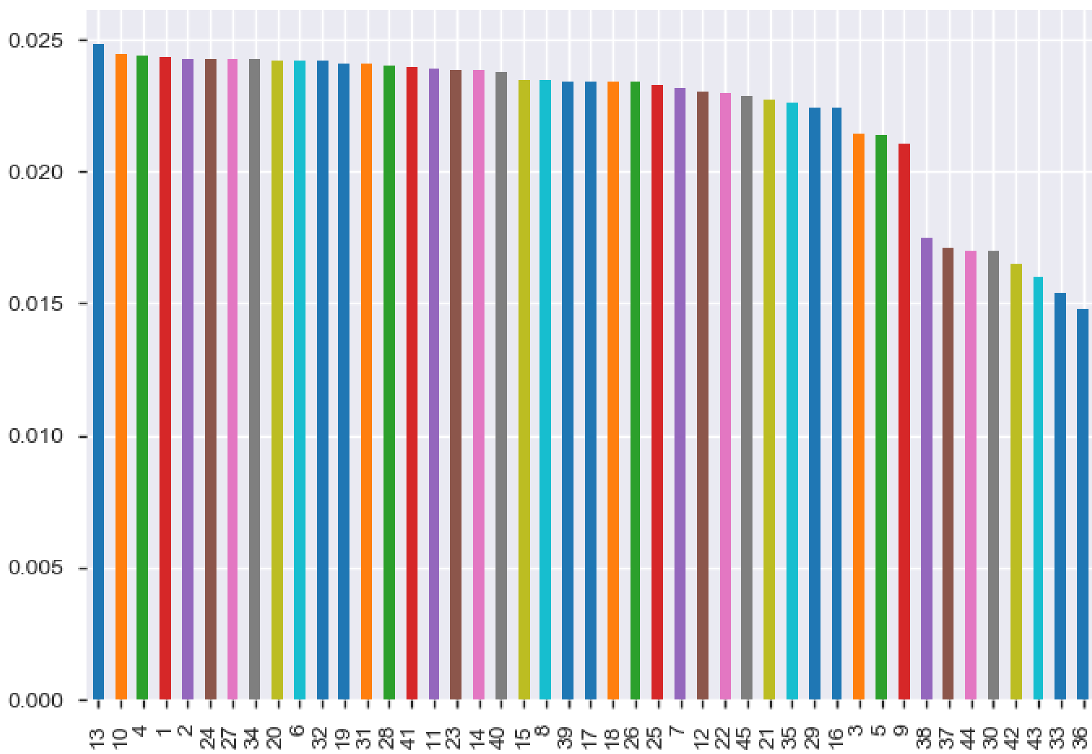
Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x254247be518>
```




```
In [19]: sns.heatmap(test.corr())

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x2541267b2e8>
```



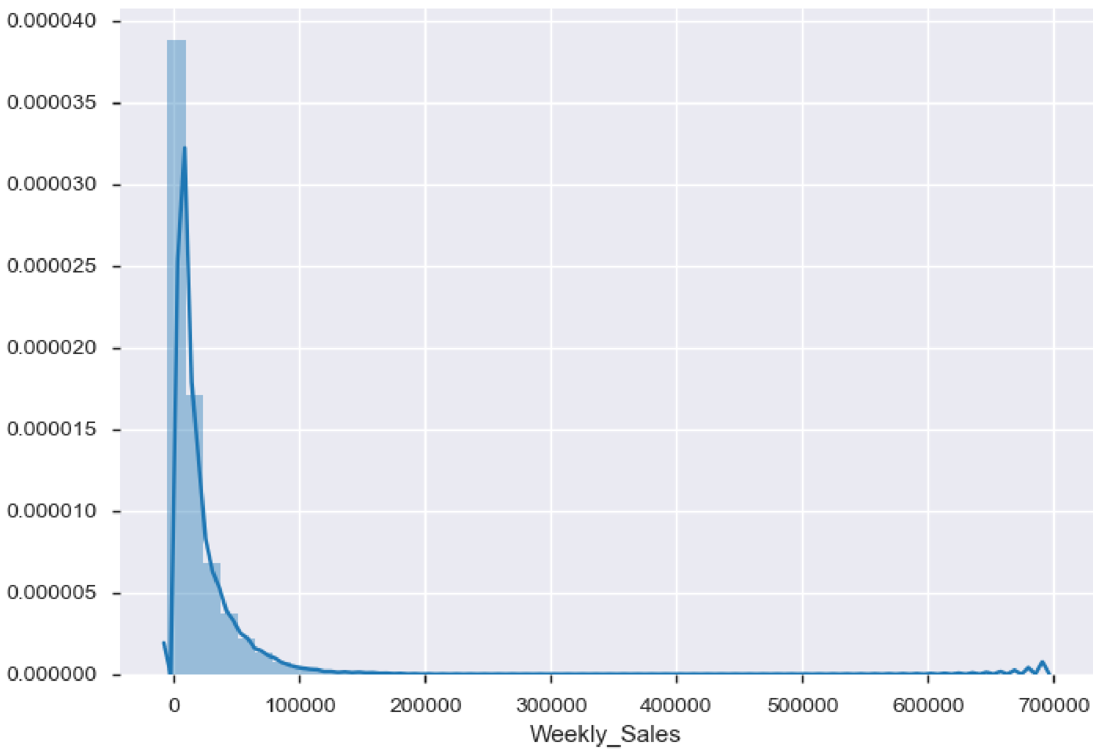

```
In [20]: train['Store'].value_counts(normalize=True).plot(kind = 'bar',fig=(4,5))

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x2542081abe0>
```



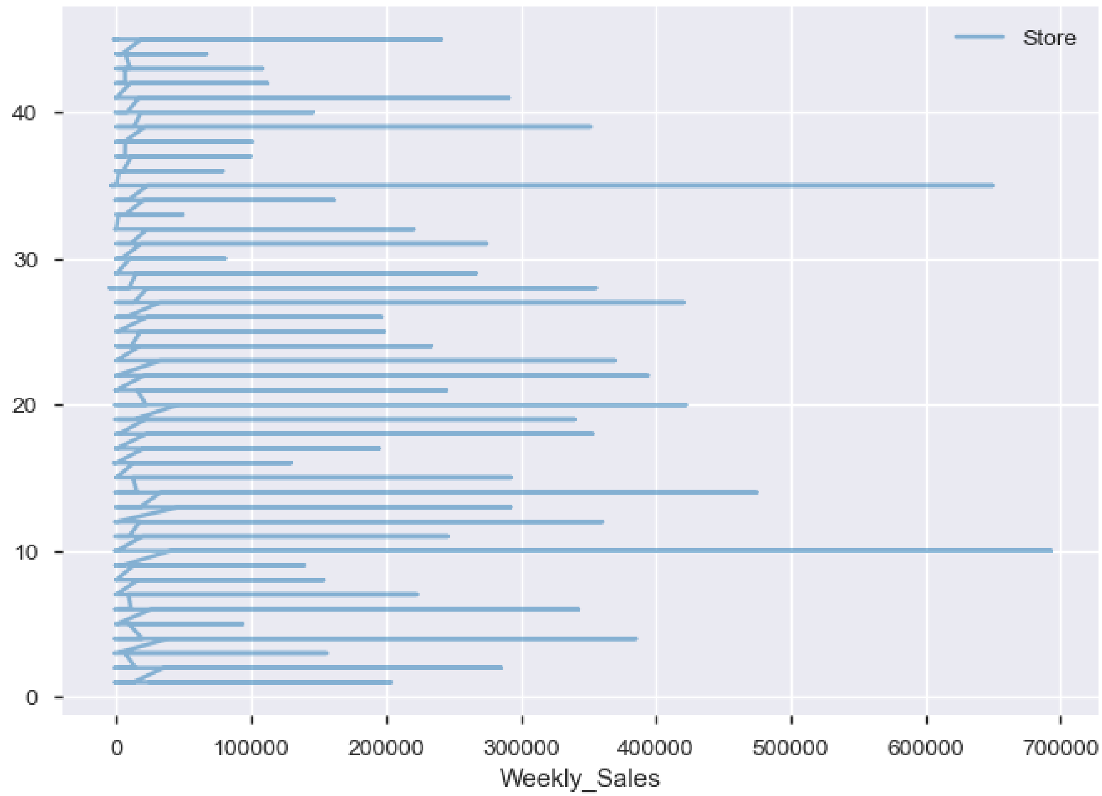

```
In [21]: sns.distplot(train.Weekly_Sales)
```

```
C:\Users\yiyuh\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462: UserWarning:

The 'normed' kwarg is deprecated, and has been replaced by the 'density' kwarg.
```

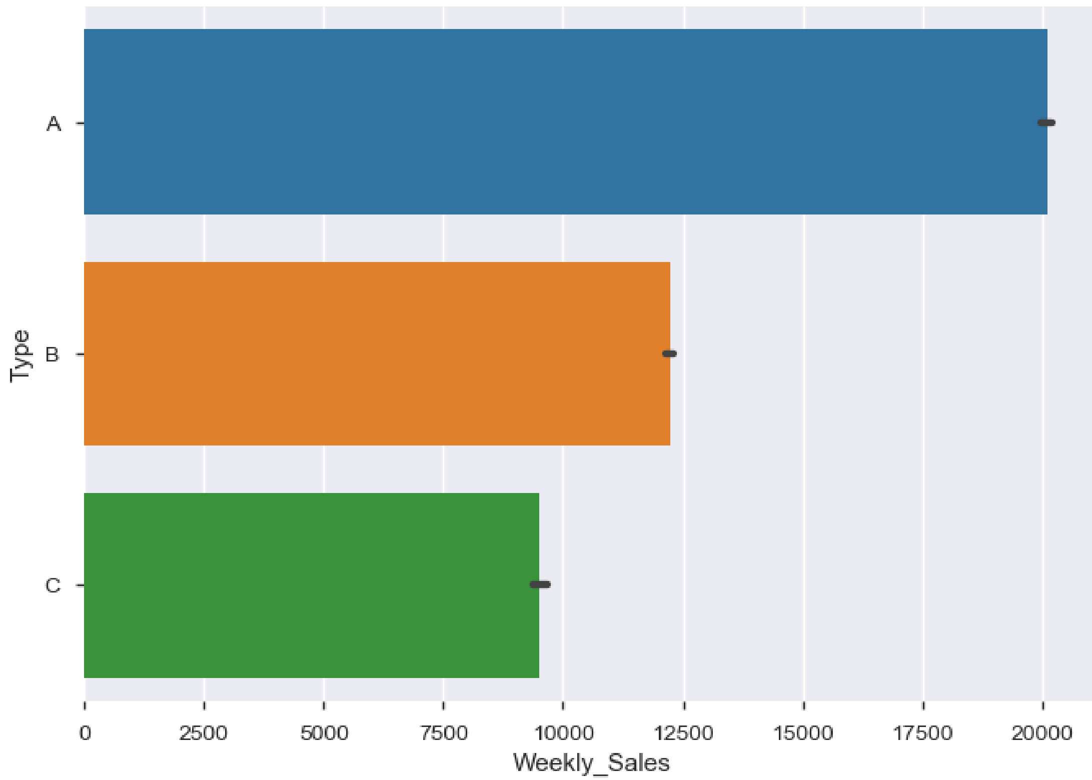
```
Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x254208156a0>
```



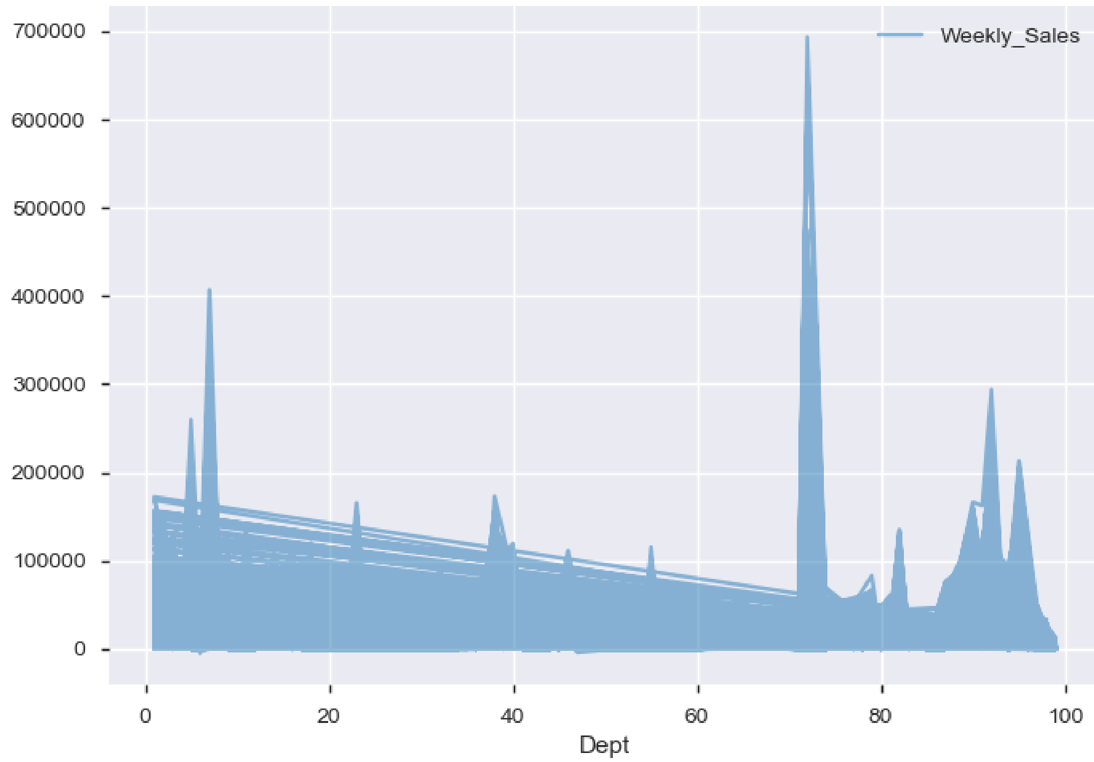

```
In [22]: train.plot(kind='line', x='Weekly_Sales', y='Store', alpha=0.5)

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x2541494add8>
```




```
In [23]: #tips = sns.load_dataset('train')
         sns.barplot(x=train["Weekly_Sales"],y=train["Type"])

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x25415fab240>
```




```
In [23]: train.plot(kind='line', x='Dept', y='Weekly_Sales', alpha=1.5,fig=(4,5))

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x26f09521f98>
```



Data Exploration: The first variable to note is the weekly sales. The average is at 15981 in US dollars. There has been a week where they lost profit with the weekly sale value at -4988.9. Despite most items being at a lower price when there is a holiday within the week, sales are still higher on average when it is a holiday. It seems to be that Wal-Mart is efficient in marking down the right products at the perfect amount while still making profit. That observation seems a whole different project. The maximum sales are from black Friday and Christmas. Store types A and B are similar, however type C has significantly lower amount of weekly sales. The correlation between the variables are tested using the pearson and spearmen methods, visualized with heatmaps below (Figure1, Figure2). The correlation between the size of the store and weekly sales shows to be promising (Figure 3). A general summary of the variables can be seen in the table below. More date variables are adjusted, creating new month variables and adding their holidays within the months. This helps generalize the time frame in which the sales performed in. To help with the projection of the sales for each department in each store, the markdown variables must be observed to understand which department is being marked down during the given holiday.

```
In [25]: print (train.isnull().sum())
         print ("******************************")
         print (test.isnull().sum())
```

```
Store                    0
Dept                     0
Date                     0
```

```
Weekly_Sales         0
IsHoliday            0
Type                 0
Size                 0
Temperature          0
Fuel_Price           0
MarkDown1       270889
MarkDown2       310322
MarkDown3       284479
MarkDown4       286603
MarkDown5       270138
CPI                  0
Unemployment         0
dtype: int64
******************************
Store               0
Dept                0
Date                0
IsHoliday           0
Type                0
Size                0
Temperature         0
Fuel_Price          0
MarkDown1         149
MarkDown2       28627
MarkDown3        9829
MarkDown4       12888
MarkDown5           0
CPI                 0
Unemployment        0
dtype: int64
```

In [26]:
```python
# Some missing data that needs to be taken care of
test['CPI']=test.groupby(['Dept'])['CPI'].transform(lambda x: x.fillna(x.mean()))
test['Unemployment']=test.groupby(['Dept'])['Unemployment'].transform(lambda x: x.fil
```

In [27]:
```python
train=train.fillna(0)
test=test.fillna(0)
```

In [28]:
```python
print (train.isnull().sum())
print ("******************************")
print (test.isnull().sum())
```

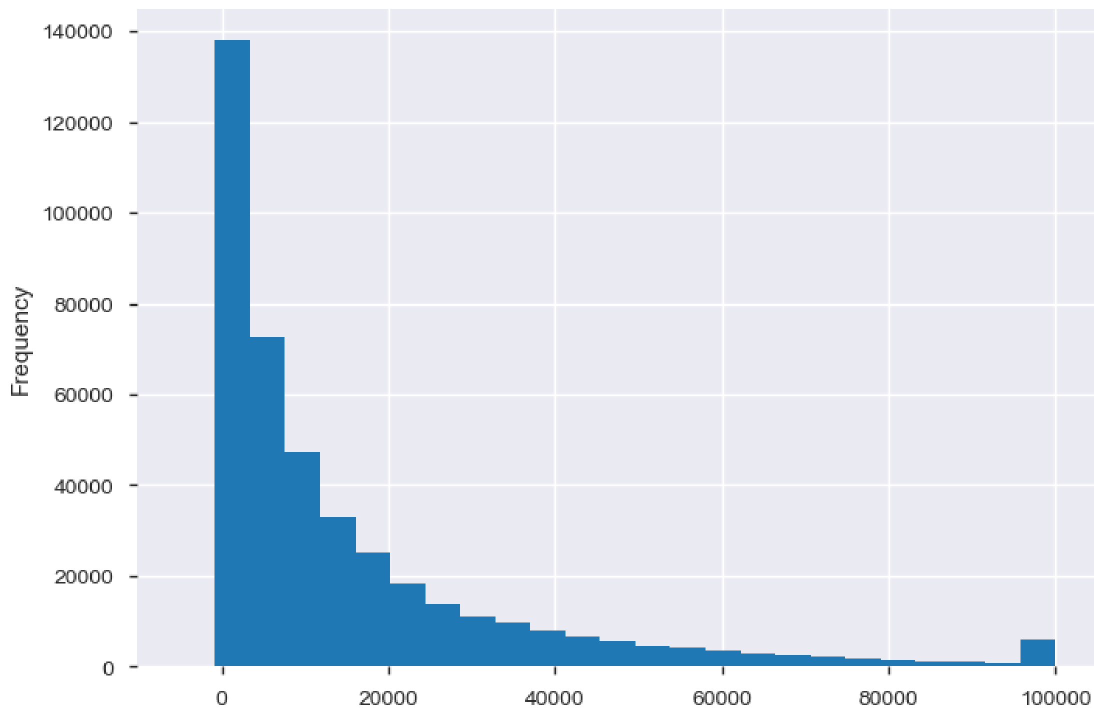
```
Store           0
Dept            0
Date            0
Weekly_Sales    0
IsHoliday       0
```

```
Type           0
Size           0
Temperature    0
Fuel_Price     0
MarkDown1      0
MarkDown2      0
MarkDown3      0
MarkDown4      0
MarkDown5      0
CPI            0
Unemployment   0
dtype: int64
******************************
Store          0
Dept           0
Date           0
IsHoliday      0
Type           0
Size           0
Temperature    0
Fuel_Price     0
MarkDown1      0
MarkDown2      0
MarkDown3      0
MarkDown4      0
MarkDown5      0
CPI            0
Unemployment   0
dtype: int64
```

In [29]:
```
# Taking care of outliers
train.Weekly_Sales=np.where(train.Weekly_Sales>100000, 100000,train.Weekly_Sales)
```

In [30]:
```
train.Weekly_Sales.plot.hist(bins=25)
```

Out[30]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25418c4f978>
```



Feature Extraction

```
In [31]: train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 421570 entries, 0 to 421569
Data columns (total 16 columns):
Store           421570 non-null int64
Dept            421570 non-null int64
Date            421570 non-null object
Weekly_Sales    421570 non-null float64
IsHoliday       421570 non-null bool
Type            421570 non-null object
Size            421570 non-null int64
Temperature     421570 non-null float64
Fuel_Price      421570 non-null float64
MarkDown1       421570 non-null float64
MarkDown2       421570 non-null float64
MarkDown3       421570 non-null float64
MarkDown4       421570 non-null float64
MarkDown5       421570 non-null float64
CPI             421570 non-null float64
Unemployment    421570 non-null float64
dtypes: bool(1), float64(10), int64(3), object(2)
memory usage: 71.9+ MB
```

```
In [32]: train['Date'] = pd.to_datetime(train['Date'])
         test['Date'] = pd.to_datetime(test['Date'])
```

```
In [33]: # Extract date features
         train['Date_dayofweek'] =train['Date'].dt.dayofweek
         train['Date_month'] =train['Date'].dt.month
         train['Date_year'] =train['Date'].dt.year
         train['Date_day'] =train['Date'].dt.day

         test['Date_dayofweek'] =test['Date'].dt.dayofweek
         test['Date_month'] =test['Date'].dt.month
         test['Date_year'] =test['Date'].dt.year
         test['Date_day'] =test['Date'].dt.day
```

```
In [34]: print (train.Type.value_counts())
         print ("*************************")
         print (test.Type.value_counts())
```

```
A    215478
B    163495
C     42597
Name: Type, dtype: int64
*************************
A    58713
B    44500
C    11851
Name: Type, dtype: int64
```

```
In [35]: print (train.IsHoliday.value_counts())
         print ("***************************")
         print (test.IsHoliday.value_counts())
```

```
False    391909
True      29661
Name: IsHoliday, dtype: int64
**************************
False    106136
True       8928
Name: IsHoliday, dtype: int64
```

```
In [36]: train_test_data = [train, test]
```

Converting Categorical Variable 'Type' into Numerical Variable For A=1 , B=2, C=3

```
In [37]: type_mapping = {"A": 1, "B": 2, "C": 3}
         for dataset in train_test_data:
             dataset['Type'] = dataset['Type'].map(type_mapping)
```

Converting Categorical Variable 'IsHoliday' into Numerical Variable

```
In [38]: type_mapping = {False: 0, True: 1}
         for dataset in train_test_data:
             dataset['IsHoliday'] = dataset['IsHoliday'].map(type_mapping)
```

Creating Extra Holiday Variable. If that week comes under extra holiday then 1(=Yes) else 2(=No)

Making New Holiday Variable Based on Given Data

```
In [39]: train['Super_Bowl'] = np.where((train['Date']==datetime(2010, 2, 12)) | (train['Date'
         train['Labour_Day'] = np.where((train['Date']==datetime(2010, 9, 10)) | (train['Date'
         train['Thanksgiving'] = np.where((train['Date']==datetime(2010, 11, 26)) | (train['Da
         train['Christmas'] = np.where((train['Date']==datetime(2010, 12, 31)) | (train['Date'
         #.......................................................................
         test['Super_Bowl'] = np.where((test['Date']==datetime(2010, 2, 12)) | (test['Date']==
         test['Labour_Day'] = np.where((test['Date']==datetime(2010, 9, 10)) | (test['Date']==
         test['Thanksgiving'] = np.where((test['Date']==datetime(2010, 11, 26)) | (test['Date'
         test['Christmas'] = np.where((test['Date']==datetime(2010, 12, 31)) | (test['Date']==
```

Altering the isHoliday value depending on these new holidays

```
In [40]: train['IsHoliday']=train['IsHoliday']|train['Super_Bowl']|train['Labour_Day']|train['
         test['IsHoliday']=test['IsHoliday']|test['Super_Bowl']|test['Labour_Day']|test['Thanks
```

```
In [41]: print (train.Christmas.value_counts())
         print (train.Super_Bowl.value_counts())
         print (train.Thanksgiving.value_counts())
         print (train.Labour_Day.value_counts())
```

```
0    415624
1      5946
Name: Christmas, dtype: int64
0    412675
1      8895
Name: Super_Bowl, dtype: int64
0    415611
1      5959
Name: Thanksgiving, dtype: int64
0    412709
1      8861
Name: Labour_Day, dtype: int64
```

```
In [42]: print (test.Christmas.value_counts())
         print (test.Super_Bowl.value_counts())
         print (test.Thanksgiving.value_counts())
         print (test.Labour_Day.value_counts())
```

```
0    112076
1      2988
Name: Christmas, dtype: int64
0    112100
1      2964
Name: Super_Bowl, dtype: int64
0    112088
1      2976
Name: Thanksgiving, dtype: int64
0    115064
Name: Labour_Day, dtype: int64
```

In [43]:
```
# Since we have Imputed IsHoliday according to Extra holidays. These extra holiday va
# Droping the Extra holiday variables because its redundant.
dp=['Super_Bowl','Labour_Day','Thanksgiving','Christmas']
train.drop(dp,axis=1,inplace=True)
test.drop(dp,axis=1,inplace=True)
```

In [44]:
```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 421570 entries, 0 to 421569
Data columns (total 20 columns):
Store              421570 non-null int64
Dept               421570 non-null int64
Date               421570 non-null datetime64[ns]
Weekly_Sales       421570 non-null float64
IsHoliday          421570 non-null int64
Type               421570 non-null int64
Size               421570 non-null int64
Temperature        421570 non-null float64
Fuel_Price         421570 non-null float64
MarkDown1          421570 non-null float64
MarkDown2          421570 non-null float64
MarkDown3          421570 non-null float64
MarkDown4          421570 non-null float64
MarkDown5          421570 non-null float64
CPI                421570 non-null float64
Unemployment       421570 non-null float64
Date_dayofweek     421570 non-null int64
Date_month         421570 non-null int64
Date_year          421570 non-null int64
Date_day           421570 non-null int64
dtypes: datetime64[ns](1), float64(10), int64(9)
memory usage: 87.5 MB
```

Dropping irrelevant variables: MarkDown5 is highly skewed

```
In [45]: features_drop=['Unemployment','CPI','MarkDown5']
         train=train.drop(features_drop, axis=1)
         test=test.drop(features_drop, axis=1)

In [46]: train.head(2)

Out[46]:    Store  Dept        Date  Weekly_Sales  IsHoliday  Type    Size  Temperature  \
         0      1     1  2010-02-05      24924.50          0     1  151315        42.31
         1      1     2  2010-02-05      50605.27          0     1  151315        42.31

            Fuel_Price  MarkDown1  MarkDown2  MarkDown3  MarkDown4  Date_dayofweek  \
         0       2.572        0.0        0.0        0.0        0.0               4
         1       2.572        0.0        0.0        0.0        0.0               4

            Date_month  Date_year  Date_day
         0           2       2010         5
         1           2       2010         5

In [47]: test.head(2)

Out[47]:    Store  Dept        Date  IsHoliday  Type    Size  Temperature  Fuel_Price  \
         0      1     1  2012-11-02          0     1  151315        55.32       3.386
         1      1     2  2012-11-02          0     1  151315        55.32       3.386

            MarkDown1  MarkDown2  MarkDown3  MarkDown4  Date_dayofweek  Date_month  \
         0    6766.44     5147.7      50.82     3639.9               4          11
         1    6766.44     5147.7      50.82     3639.9               4          11

            Date_year  Date_day
         0       2012         2
         1       2012         2

In [48]: #### train X= Exery thing except Weekly_Sales
         train_X=train.drop(['Weekly_Sales','Date'], axis=1)

         #### train Y= Only Weekly_Sales
         train_y=train['Weekly_Sales']
         test_X=test.drop('Date',axis=1).copy()

         train_X.shape, train_y.shape, test_X.shape

Out[48]: ((421570, 15), (421570,), (115064, 15))
```

After basic data exploration and enough data processing to remove missing values, there is enough knowledge to strategically approach the problem. The important variables to select for a model seem to include the store type, size, the holiday variable, and the dates (whether it be weekly or monthly). The key is relating the months to their respective holiday and gathering the period in when the sales increase for each holiday. During the Christmas season, the sales are increased nearly the entire month of December. However, during the super bowl holiday, the sales see a significant spike only within the two weeks before the super bowl event. A lagged variable for each of the holidays is needed to run an accurate model for this issue. In opposition, black Friday sales are only for a single day.

1.) Linear Regression

```
In [49]: clf = LinearRegression()
         clf.fit(train_X, train_y)
         y_pred_linear=clf.predict(test_X)
         acc_linear=round( clf.score(train_X, train_y) * 100, 2)
         print ('score:'+str(acc_linear) + ' percent')
```

```
score:8.86 percent
```

```
In [51]: #roc_auc_score(train_X, y_pred_linear)
```

2.) Random Forest

This algorithm takes a significant amount of time and may crash processing machine

```
In [52]: #clf = RandomForestRegressor(n_estimators=100)
         #clf.fit(train_X, train_y)
         #y_pred_rf=clf.predict(test_X)
         #acc_rf= round(clf.score(train_X, train_y) * 100, 2)
         #print ("Accuracy: %i %% \n"%acc_rf)
```

The result of Random Forest is at 99.77%

3.) Decision Tree

```
In [53]: clf=DecisionTreeRegressor()
         clf.fit(train_X, train_y)
         y_pred_dt= clf.predict(test_X)
         acc_dt = round( clf.score(train_X, train_y) * 100, 2)
         print (str(acc_dt) + ' percent')
```

```
100.0 percent
```

. . . . **Impressive** . . . . It's interesting to see how much of a difference the classifier makes. The Random forest and the decision tree approach 100 while the linear regression only gets an 8% accuracy. I believe the next step is to use unsupervised machine learning, clustering