Contents

1.0 Introduction

Sales forecasting is a crucial part of the financial planning of any retail operation. Forecasting is a self-assessment tool that uses past sales statistics to intelligently predict future performance (Sun, et al., 2008). This is essential for meeting the demands of consumers whilst controlling pricing and optimizing the use of inventory space, as holding excess inventory adds to overhead costs for a business (Guerts & Kelly, 1986).

1.1 Problem Statement

A key challenge of modeling retail data is the need to make decisions based on limited history. Holidays and select major events that come once a year, and so does the chance to see how strategic decisions impacted the bottom line. In addition, the relationship of how markdowns (a reduction in the originally marked retail price of merchandise) affect sales not fully known. The goal is to use historical weekly sales data to develop a model that better understands consumer purchasing patterns so we can predict which departments will be affected by these occasional events and to what extent. By doing so, we hope to provide retailers with the knowledge to more efficiently optimize their shelving and inventory space. We would do so by using machine learning models that estimates future sales when provided with current data about relevant features.

1.2 Hypothesis

To go forward with our analysis, we hypothesize that there are meaningful underlying dependencies between the features provided in the dataset, and that by training a model on these features, we would be able to reuse that model in production (i.e. given the features, the model estimates the result). Also, we assume that these underlying relations can be learned by the model.
We will be using three methods (Linear Regression, Time-Series, Random Forests) and have the following intuitions:

- The LR model might not perform as well as the others since it is very simple.
- The Time-Series model would probably yield better results, since the problem is inherently time-dependent. However ensemble methods (including RFs) are known to be quite powerful.

We reevaluate these statements later on.

2.0 Data

2.1 Data Overview

The dataset (from Kaggle) consists of historical sales data for 45 stores located in different regions of the US between 2010 and 2012. The company runs several promotional markdown events throughout the year. These markdowns precede prominent holidays, the four largest of which are the Super Bowl, Labor Day, Thanksgiving, and Christmas.
The training data consist of 421,570 records:

- Sales_data-set.csv: sales data (2010-02-05 to 2012-11-01) from departments of all stores
- Store_data-set.csv: type and size of the stores
- Features_dataset.csv: additional data related to the stores and regions

Predictors are:

Feature	Description
Store	Store number
Dept	Department number
Date	Starting date of the week (DD/MM/YYYY)
Weekly_Sales	Sales for the given department and store in that week
IsHoliday	Whether the week is a special holiday week
Temperature	Average temperature in the region
Fuel_Price	Cost of fuel in the region
MarkDown 1 to 5	Anonymized data related to promotional markdowns. MarkDown data is only available after Nov 2011, and is not available for all stores at all times. Any missing value is marked NA.
CPI	consumer price index
Unemployment	unemployment rate
Type	type of the store (A, B or C)
Size	size of the store

2.2 Data Pre-processing

It was necessary to combine the 3 datasets into a single dataframe prior to analysis. Below is a snapshot after merging the 3 datasets:

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5
420280	45	98	28/09/2012	508.37	False	18.266667	3.997	4556.61	20.64	1.50	1601.01	3288.25
420281	45	98	05/10/2012	628.10	False	18.272222	3.985	5046.74	0.00	18.82	2253.43	2340.01
420282	45	98	12/10/2012	1061.02	False	12.483333	4.000	1956.28	0.00	7.89	599.32	3990.54
420283	45	98	19/10/2012	760.01	False	13.594444	3.969	2004.02	0.00	3.18	437.73	1537.49
420284	45	98	26/10/2012	1076.80	False	14.916667	3.882	4018.91	58.08	100.00	211.94	858.33

We noticed that 1285 rows (out of 400000) have values less than 0 in weekly sales. One option was to replace these with the mean weekly sales. However, we have discarded these rows completely in our analysis assuming they are data entry errors.

```
main_df = main_df[(main_df['Weekly_Sales']>=0)]
main_df.shape
main_df.head(5)
```

We converted the temperature from Farenheit to Celcius.

```
df_combine['Temperature'] = (df_combine['Temperature']- 32) * 5.0/9.0
df_combine.head(2)
```

Values for all features were converted from ‘N/A’ to 0 using .fillna(0).
We converted the “Type” variable into categorical by :

```
df_combine["Type"] = df_combine["Type"].astype('category')
df_combine["Type"] = df_combine["Type"].cat.codes
df_combine.tail()
```

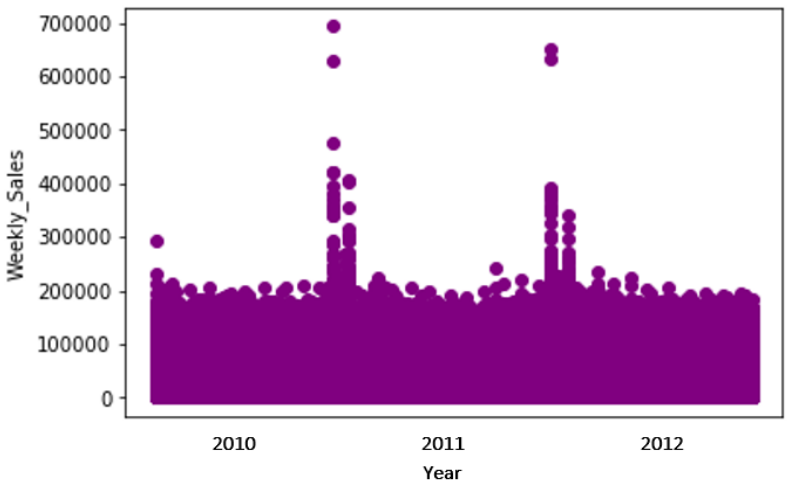
On initial analysis, we found that data is not present for all weeks for all stores. We find this as a major drawback since our analysis depends on the pattern and behavior of the existing data. As there is missing data for many weeks, we may have been biased in our representation.

2.3 Data Exploration

We explored the data to find any correlations or inferences. This helped us in understanding the data before using it for modelling and prediction.

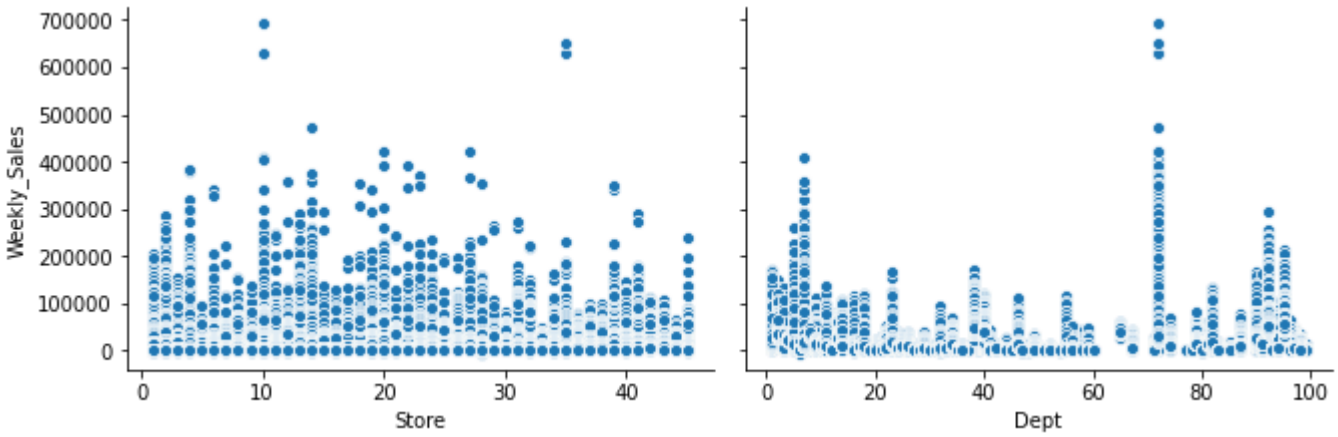
2.3.1 Total Sales vs Time

The figure shows a scatter plot of the total sales vs time. We observe that sales generally follow the same pattern except for 2 spikes in the period, possibly due to seasonal variations or stores enhancing their sales channels.

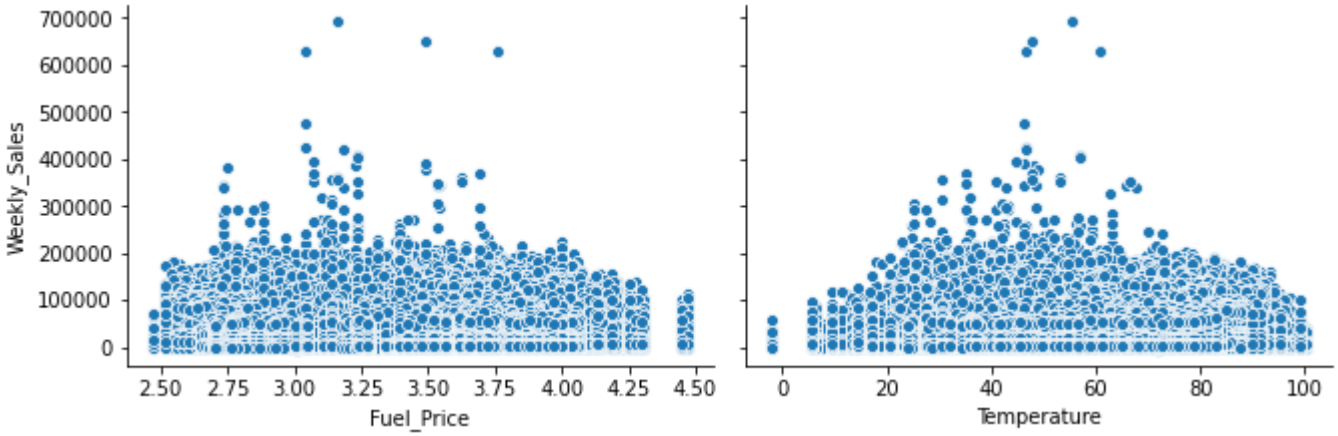


2.3.2 Sales vs Features

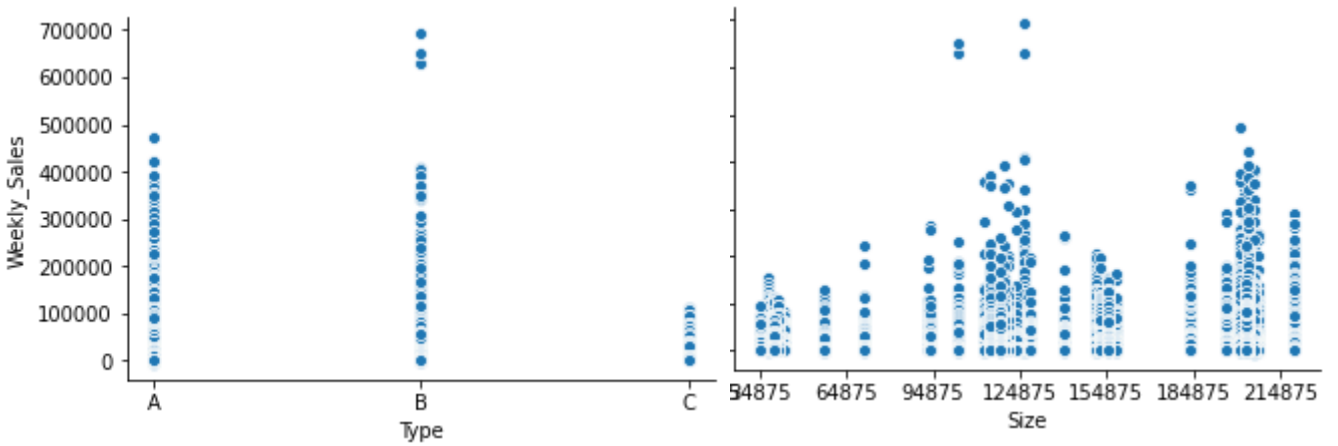
Here, we expand on the previous section and explore the possible relations between Weekly_Sales and other features:



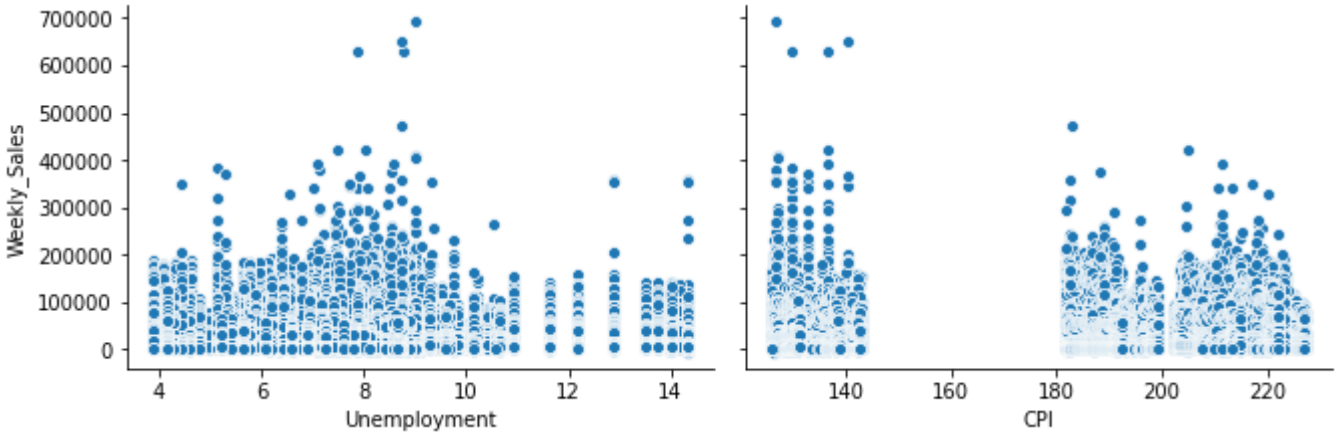
- A particular department makes noticeably higher sales.



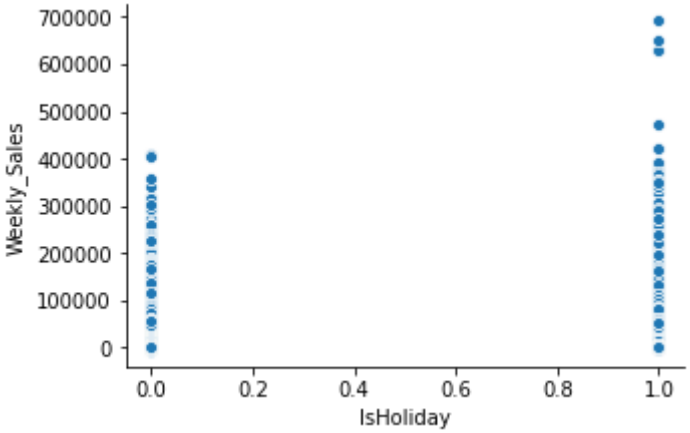
- Observing the small variations in the left figure, Fuel_Price does not seem to be affecting Weekly_Sales. However, a number of high Weekly_Sales are associated with fuel prices ranging from 2.75 - 3.75. This is not enough to make an assumption about the effects of fuel price, as this increase in sales might have been due to other factors at the time.



- There appears to be a positive correlation between weekly sales and size.



- Sales are at their highest when unemployment is between 7 - 9%.
- A general negative correlation is observed in weekly sales and CPI, which would make sense considering the higher the prices are, the less the customers could afford to purchase goods.



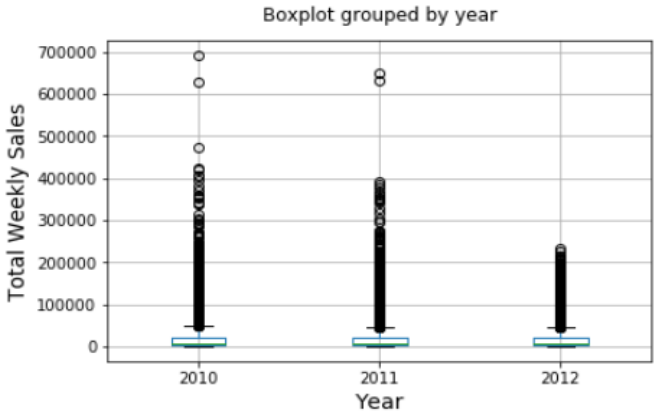
- Sales are generally higher on holidays but not as much as expected (450k compared to 400k). However, there were a few holidays that exceeded the average sales (600k to 700k).

2.3.3 Weekly Sales vs Year

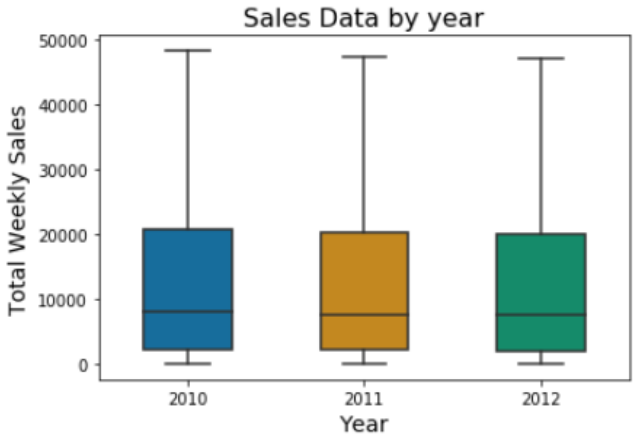
Aggregating the sales of all stores, we find 2012 having the lowest maximum sales and 2011 having the highest overall sale:

Year	Min	Max	Sales Total
2010	0.01	693099.4	2288919000
2011	0.01	649770.2	2448233000
2012	0.01	233140.3	2000156000

Using the boxplot function, we see that each year has multiple outliers. This is illustrated in the figure below:

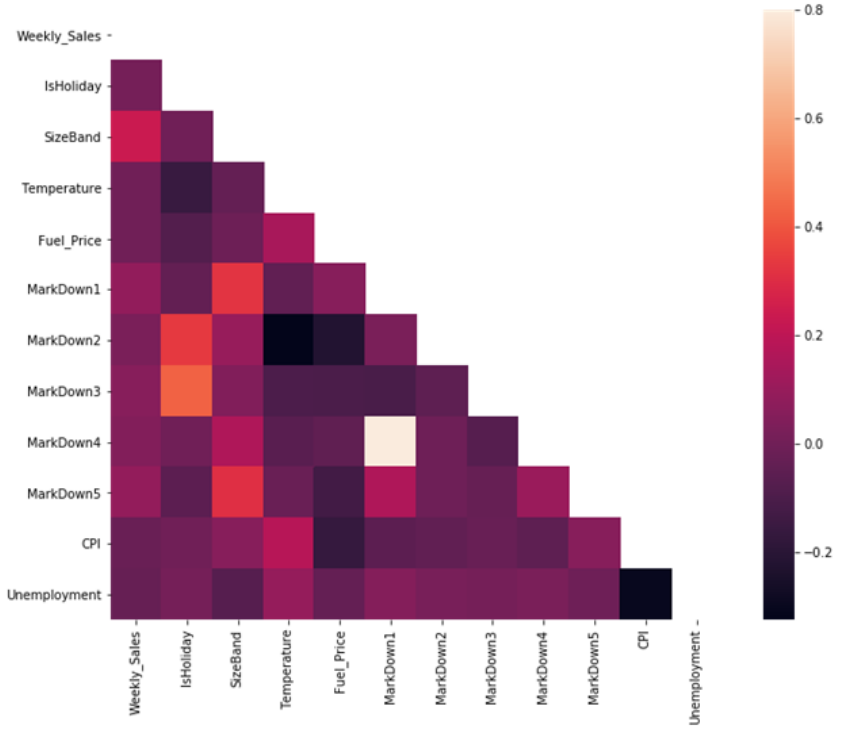


We can attribute this to the fact that the sales increase significantly during certain times and the values go above the average. On removing the outliers (showfliers=False) we see the distribution and data summary.



2.3.4 Correlation Matrix

Then we compute the correlation matrix, ignoring Store and Deptartment:



Surprisingly, most of the exogenous factors, such as CPI, Unemployment and Fuel Price did not appear to have a strong correlation with Weekly Sales.

2.4 Feature Selection

We have adopted different feature selection methods for our models, which are explained in their respective sections:

- Linear Regression
- Time-Series
- Random Forests

3.1 Linear Regression

3.1.1 Description

Linear Regression (LR) is one of the simplest models for machine learning. A multiple LR assumes linear relationships between the features and the target, then fits a line to the data that would result in the lowest residual sum of squared errors (Yan, 2009)

3.1.2 Justification

Our first intuition is that a simple, easily interpretable model like LR is a good starting point for our analysis. Also, this would provide a basis for future model comparisons.

3.1.3 Feature Selection

To implement the LR algorithm more efficiently, some features needed to be transformed. These feature transformations include log-scaling Weekly_Sales and Size (ranging several orders of magnitude) making the values comparable, and encoding time as a new feature (year_week).

```
# time-related features
df.Date = pd.to_datetime(df.Date, format='%d/%m/%Y')
df['week'] = df.Date.dt.week
df['year'] = df.Date.dt.year
df1 = df.copy()
df.drop(['Date'], axis=1, inplace=True)
df['year_week'] =  df['year'].astype(str) + df['week'].astype(str)
df['year_week'] =  df['year_week'].astype(int)
df.loc[df['week']<10, 'year_week'] = df.loc[df['week']<10, 'year'].astype(str) + '0' + df.loc[
    df['week']<10, 'week'].astype(str)
df['year_week'] =  df['year_week'].astype(int)
```

Also, Type is encoded as 0, 1 or 2.

```
df['Type'] = df['Type'].factorize()[0]
```

We use the following features for our LR model: Store, Dept, IsHoliday, Type, Temperature, Fuel_Price, CPI, Unemployment, Size_log, week, year_week.

3.1.4 Model

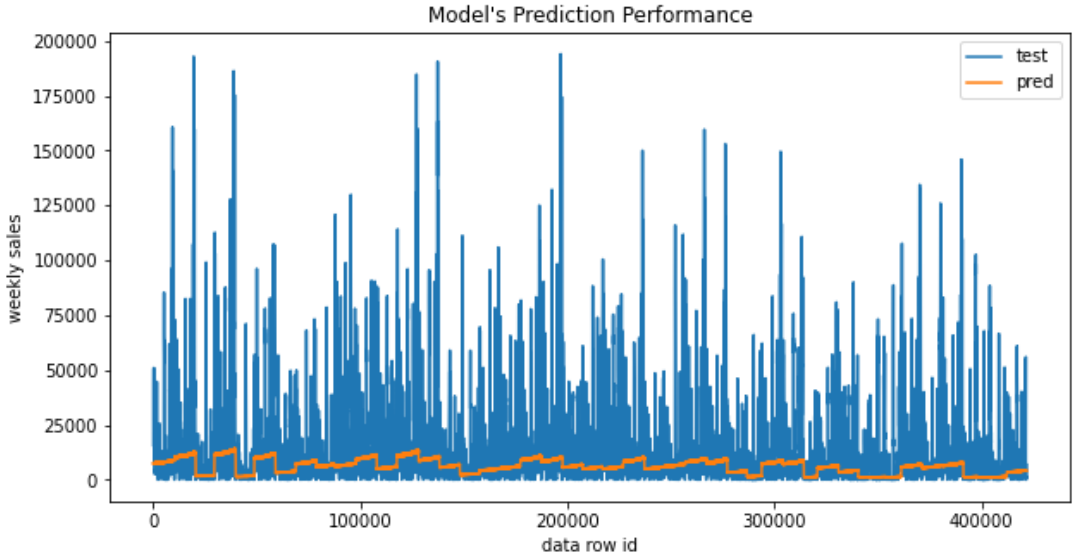
We train a LR model on the 11 features mentioned above. The testing dataset includes all data from the 30th week of 2012 to the 43rd and the training set includes the data prior to that from the 5th week of 2010. By splitting the dataset like so, we simulate what happens during production (making predictions based on available features and previous data), which wouldn't be the case in a random split.

```
# linear regression
X_test, y_test, X_train, y_train = data_split(df, np.linspace(201230, 201243))
model = LinearRegression(copy_X=True, n_jobs=-1)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

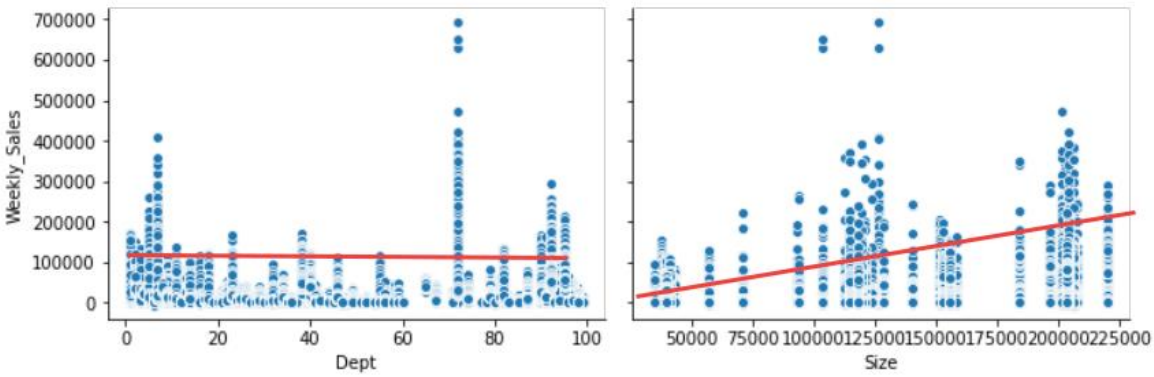
3.3.5 Results

The resulting accuracy is quite low:

```
accuracy (R^2):
 11.562099711020258 %

 coefficients:
  [-1.21495468e-02  1.85250597e-03  5.18903728e-03  5.28490602e-02
 -1.90441458e-03  2.33343359e-02 -7.61869868e-04 -1.65426579e-02
  1.11359026e+00  3.73972419e-03 -3.24703265e-04]
```



This is due to LR's inherent shortcomings. LR performs rather okay while dealing with features such as Size, where a single line could fit the data to some extent. However, when adding features that are categorical in nature, such as Dept (the most important feature- will be explained here), LR can't perform well. The reason is that an increase in a categorical feature's value which is encoded by numerics (Store, Dept) doesn't have any real meaning in the sense of that feature increasing, unlike continuous variables like Size (figure below). Knowing that a number of features are categorical (Dept, Store, Type, IsHoliday, week) and a single line cannot fit them well, we would expect the accuracy of this model to be very low as was speculated here.



Furthermore, backward elimination would actually decrease the accuracy of the LR model since the most important features are mostly categorical. In this case the accuracy would drop from 11.56% to 11.41% by eliminating the bottom 5 features, which yielded the highest accuracy in the RF model.

3.2 Time-Series Analysis

3.2.1 Description

A time-series is a series of data points indexed in time order. Time-series forecasting is the use of a model to predict future values based on previously observed values. Since linear regression provided us with poor results, we decided to use time-series modelling to improve our analysis (Hamilton, 1994).

3.2.2 Justification

Extrapolating beyond the range of data with Linear Regression models is notoriously unreliable as it relies on untestable assumptions about the behaviour of the data beyond their observed support (Cryer, 1986). Given our time-dependent dataset, we decided that a time series analysis is a more reliable forecasting method as it accounts for the fact that data points taken over time may have multiple internal structures such as autocorrelation, trend or seasonal variation (Hamilton, 1994). Further, this method is extensively employed in financial and business forecasts due to the models ability to accurately identify time-based patterns existing in data. Time series is a leading practice used by organizations for decision making and policy planning (Nunnari & Nunnari, 2017).

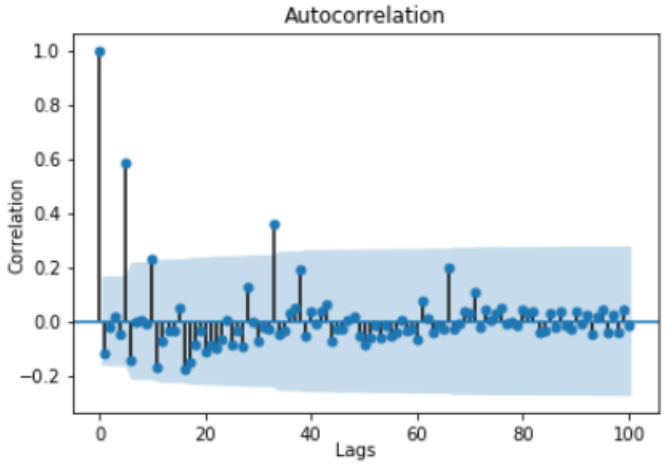
3.2.3 Feature Selection

The feature selection for this part is done here.

3.2.4 Model

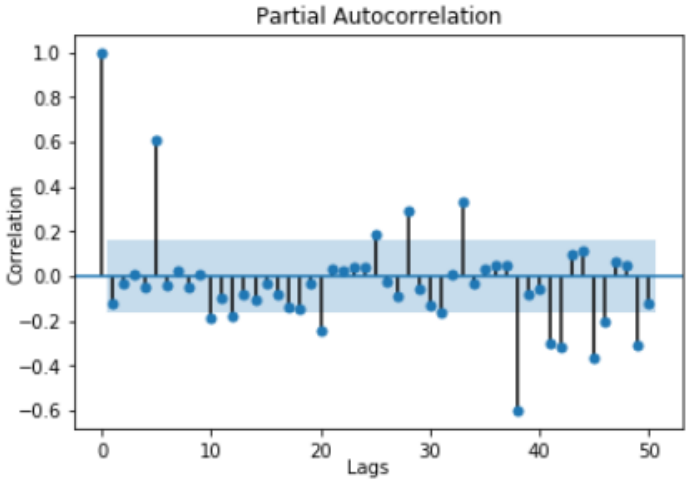
To make use of time-series, we delve with the concepts of Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF). ACF is an (complete) auto-correlation function which gives us values of correlation of any series with its lagged values (Cryer, 1986). We plotted the ACF with 100th lag values, observing that our present value could be directly related to the 45th lag, after which the autocorrelation decays.

```
acf_plot = plot_acf(ts.Weekly_Sales,lags=100)

acf_plot.text(0.5, 0.04, 'Lags', ha='center')
acf_plot.text(0.04, 0.5, 'Correlation', va='center', rotation='vertical')
```



PACF finds correlation of the residuals (after removing the effects explained by earlier lags) with the next lag value. Hence, it is ‘partial’ as we remove already found variations before we find the next correlation (Cryer, 1986). If there is any hidden information in the residual which can be modeled by the next lag, we may observe a strong correlation and we will keep that next lag as a feature.

```
pacf_plot = plot_pacf(ts.Weekly_Sales,lags=50)

pacf_plot.text(0.5, 0.04, 'Lags', ha='center')
pacf_plot.text(0.04, 0.5, 'Correlation', va='center', rotation='vertical')
```



Based on PACF, we should start with an Auto Regressive Model with lags of 1,5,18,33,43,49. The autocorrelation of a time series Y at lag 1 is the coefficient of correlation between Y_t and Y_{t-1}, which is presumably also the correlation between Y_{t-1} and Y_{t-2}. But if Y_t is correlated with Y_{t-1}, and Y_{t-1} is equally correlated with Y_{t-2}, then we should also expect to find correlation between Y_t and Y_{t-2}.Thus, the correlation at lag 1 "propagates" to lag 2 and presumably to higher-order lags. The partial autocorrelation at lag 2 is therefore the difference between the actual correlation at lag 2 and the expected correlation due to the propagation of correlation at lag 1.

A time-series is said to be AutoRegressive (AR) when the present value of the time-series can be obtained using previous values of the same time-series, i.e the present value is the weighted average of its past values (Eshel, 2011).

The AR process of an order p can be written as,

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t$$

Where ϵt is a white noise and $y't\text{-}_1$ and $y't\text{-}_2$ are the lags. Order p is the lag value after which the PACF plot crosses the upper confidence interval for the first time. These p lags will act as our features while forecasting the AR time-series.

3.2.5 Results

We used user-defined functions *fit_ar_model( )* and *predict_ar_model( )* to train our model and get predicted outcomes :-

```
def fit_ar_model(ts, orders):
    X=np.array([ ts.values[(i-orders)].squeeze() if i >= np.max(orders) else np.array(len(orders) * [np.nan]) for i in range(len(ts))])
    mask = ~np.isnan(X[:,:1]).squeeze()
    Y= ts.values
    lin_reg=LinearRegression()
    lin_reg.fit(X[mask],Y[mask])
    print(lin_reg.coef_, lin_reg.intercept_)
    print('Score factor: %.2f' % lin_reg.score(X[mask],Y[mask]))
    return lin_reg.coef_, lin_reg.intercept_
def predict_ar_model(ts, orders, coef, intercept):
    return np.array([np.sum(np.dot(coef, ts.values[(i-orders)].squeeze())) + intercept  if i >= np.max(orders) else np.nan for i in range(len(ts))])
```

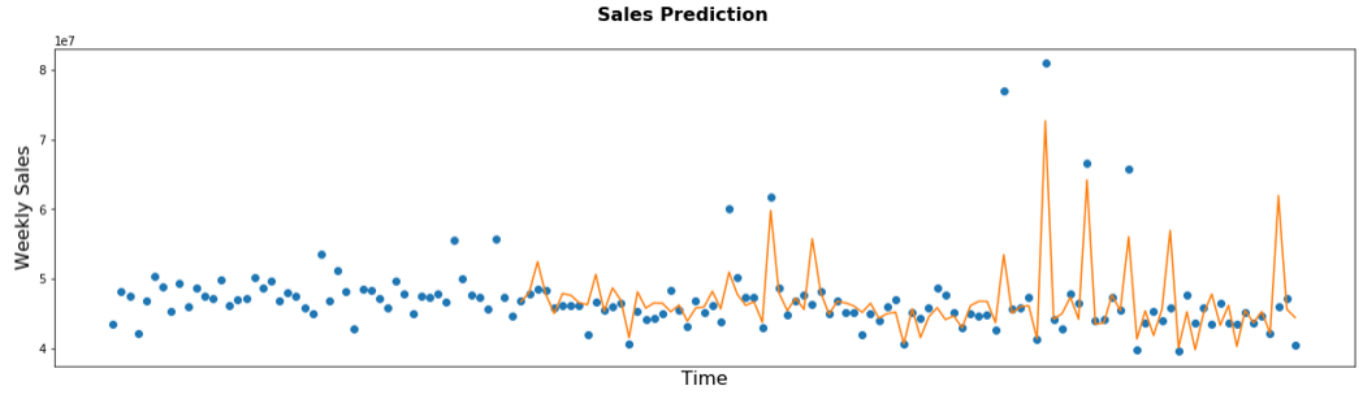
The predictive model has an R^2 score of 0.59. The residual distribution is centered around 0 with an STD of 0.07.

3.2.6 Improving the Accuracy

The accuracy can be improved by training a model on each individual store.The model including the external variables (shown below) increased the accuracy by more than 30% (R^2 score: 0.80 w.r.t 0.60). The standard deviation of the residual improved by about more than 30% (6% w.r.t. 8%).

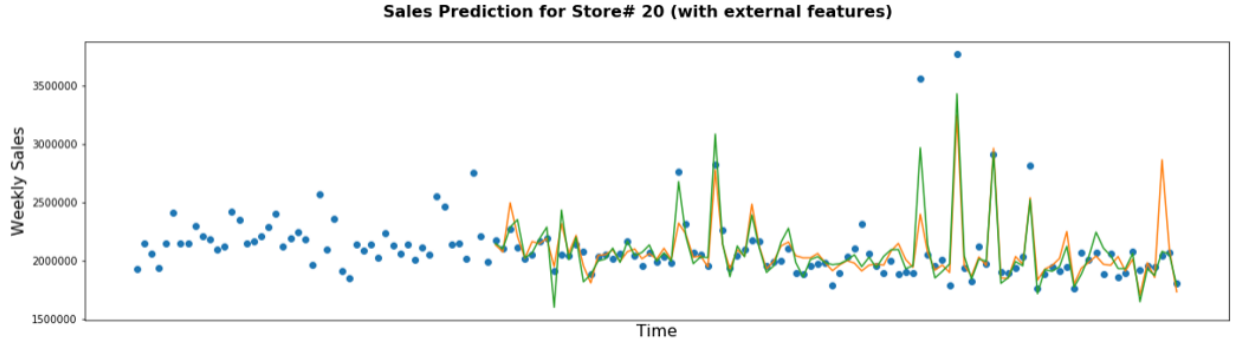
```
from sklearn.linear_model import LinearRegression
orders=np.array([1,5,18,33,43,49])
coef, intercept = fit_ar_model(ts,orders)
pred=pd.DataFrame(index=ts.index, data=predict_ar_model(ts, orders, coef, intercept))
#plt.figure(figsize=(20,5))
fig, ax = plt.subplots(figsize=(20, 5))
ax.plot(ts, 'o')
ax.plot(pred)
ax.tick_params(axis='x', which='both', bottom=False, labelbottom=False)


plt.suptitle('Sales Prediction', fontsize=16, fontweight='bold')
ax.set_xlabel('Time', fontsize=16)
ax.set_ylabel('Weekly Sales', fontsize=16)

plt.show()
```

```
[[-0.09985462  0.53898843 -0.11704762  0.63190804 -0.00981336  0.08508318]] [-1697992.39533637]
Score factor: 0.59
```




```
dfexte=dfext[['Unemployment','Fuel_Price','CPI','Temperature',
              'MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4', 'MarkDown5']]
```

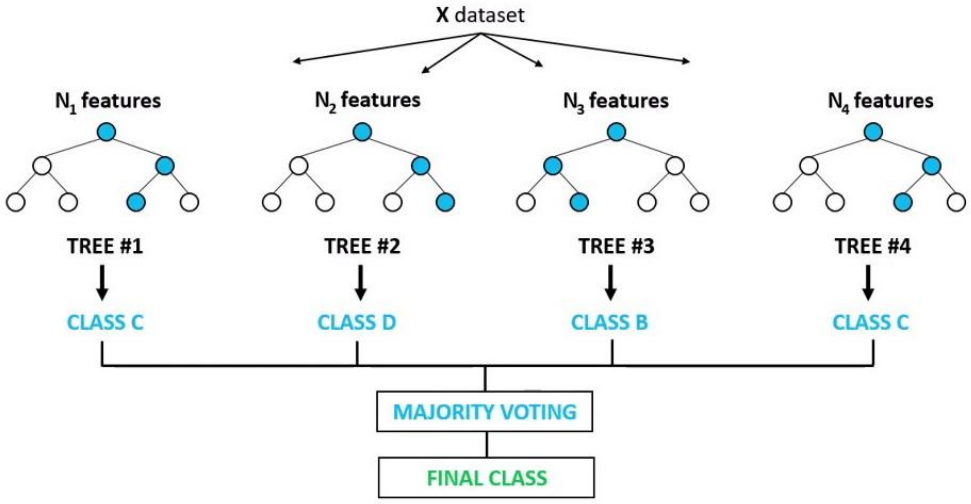
```
[-6.01606377e-02  3.79022610e-01 -1.50898565e-01 -1.06047323e-01
  1.07440838e+00  2.03829130e-02 -1.46134826e-01 -3.22905474e-01
 -2.33368725e-02 -1.67998202e+05 -1.23023930e+05  2.56285040e+03
 -7.53552662e+03  1.30021752e-01 -1.85176725e+01  1.82073988e+00
 -2.72561411e+00 -3.37631960e+00] 1940442.173821163
Score factor: 0.80
```



3.3 Random Forests

3.3.1 Description

We have used “Random Forests” to predict the weekly sales for all stores. Random forests (RFs) are primarily used for classification and also regression (Breiman, 2001).
RFs are ensembles of decision trees (DTs), whose inputs are bootstraps of the training samples. The final RF prediction is the average of all of these DTs’ predictions for a given test sample (bootstrap aggregation). Since each DT has a different bootstrap set, the variance is reduced without affecting the bias. By using this form of aggregation, RFs generally have high accuracy (less overfitting, more robust to noise), but are less interpretable than single DTs (Zhao and Zhang, 2008).



3.3.2 Justification

RFs are extremely flexible and obtain very high accuracies (Pavlov, 2000) while still being prone to overfitting. Furthermore, they have embedded feature selection (Saeys, Abeel and Van de Peer, 2008), hence they can be used for strategizing feature selection. Also they provide good estimates of the test error without repeated model training associated with cross-validation (costly).

3.3.3 Feature Selection

To implement the RF algorithm more efficiently, some features need to be transformed. The feature transformations include log-scaling the weekly sales and size, and encoding the time as a new feature, year_week.

```
# time-related features
df.Date = pd.to_datetime(df.Date, format='%d/%m/%Y')
df['week'] = df.Date.dt.week
df['year'] = df.Date.dt.year
df1 = df.copy()
df.drop(['Date'], axis=1, inplace=True)
df['year_week'] =  df['year'].astype(str) + df['week'].astype(str)
df['year_week'] =  df['year_week'].astype(int)
df.loc[df['week']<10, 'year_week'] = df.loc[df['week']<10, 'year'].astype(str) + '0' + df.loc[
    df['week']<10, 'week'].astype(str)
df['year_week'] =  df['year_week'].astype(int)
```

Also, the type feature is encoded as 0,1 or 2.

```
df['Type'] = df['Type'].factorize()[0]
```

3.3.4 Model

The testing dataset includes all data from the 30th week of 2012 to the 43rd and the training set includes the data prior to that from the 5th week of 2010. First, we train a Random Forest Regressor model to predict the Weekly_Sales_log for all the stores. We use the following eleven features: Store, Dept, IsHoliday, Type, Temperature, Fuel_Price, CPI, Unemployment, Size_log, week, year_week.

```
# Random Forest Regressor model
X_test, y_test, X_train, y_train = data_split(df, np.linspace(201230, 201243))
model = RandomForestRegressor(n_estimators=20, criterion='mse', bootstrap=True, n_jobs=-1,
                              random_state=100, oob_score=True)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Notes:

- Using the “mse” split criterion is more efficient than “mae” as it measures the quality of the split using the mean square error which is equal to variance reduction as feature selection criterion. Additionally, it is less computationally intensive since convergence is yielded sooner.
- We used the R^2 criterion as our performance measure and the out of bag (OOB) score. However, this is not to be confused with the validation score. The OOB score is calculated using only a subset of DTs that don't include the OOB sample in their bootstrap training dataset, but the validation score is calculated by using all DTs.
- No algorithm, including RFs, is safe from overfitting; however, by increasing the number of DTs, we can minimize this risk.

3.3.5 Results

The results for the RF regressor model trained on all eleven features for all stores are:

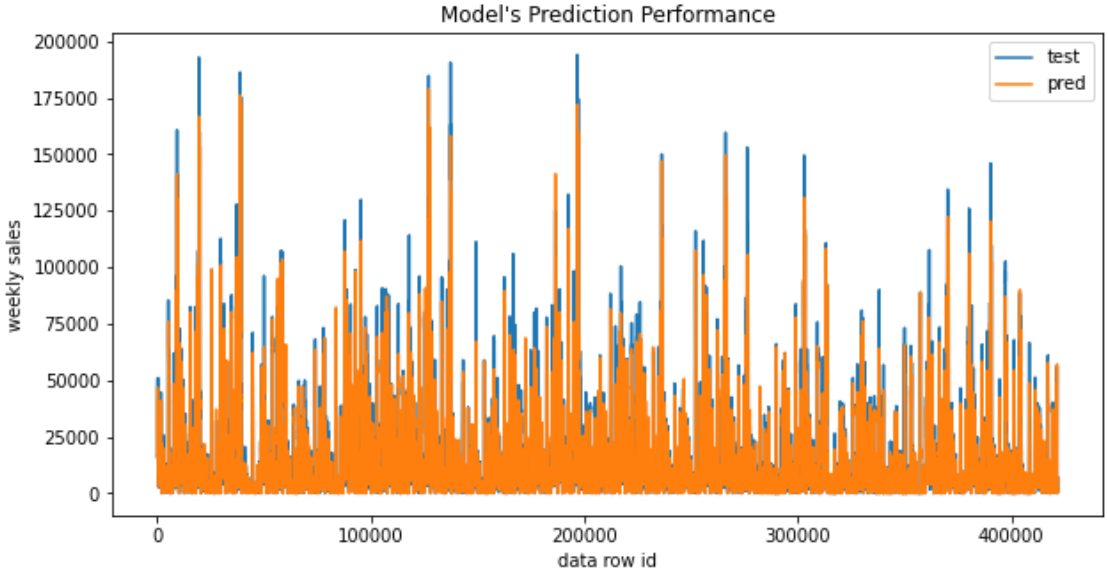
```
accuracy (R^2):
 94.93804973458559 %

 out of bag score:
  0.9722716435091473

 feature importances:
  [5.54546768e-02 7.14675333e-01 5.84760420e-04 1.81861405e-02
 8.68919511e-03 6.71070035e-03 1.64434296e-02 1.01447859e-02
 1.28962630e-01 2.81177653e-02 1.20305824e-02]
```

The accuracy (94.94%) is high, but can be improved via backward feature elimination which is the next topic of discussion. The OOB score (0.97) is quite good as well. These results demonstrate that our model performs well enough so far.
We used the .feature_importances_ method to obtain feature importance scores required for backward feature elimination. The features, in order of prediction power, are: Dept, Size_log, Store, week, Type, CPI, year_week, Unemployment, Temperature, Fuel_Price, IsHoliday.

Notes:

- The most important feature is Dept. This makes sense since departments such as groceries or electronics are more popular than say gardening.
- We had incorporated time as two separate features: year_week (e.g. ‘201205’=’2012’+’05’) and week (ranging from 1 to 52). The results show that the seasonal component captured by week is more informative than year_week. This illustrates the fact that although sales vary with time, there is a certain trend present. This means that there are certain weeks of the year that are associated with higher or lower sales.
- While sales are in general higher during holiday weeks, IsHoliday is the least important feature. This was a surprise as we expected it to be one of the key features. However, the fact that it does not contribute much to prediction accuracy, might be partly because there are far fewer holiday weeks than non-holiday (imbalance in the dataset). Furthermore, the customers’ shopping habits don’t appear to be heavily affected by holidays.

To visualize the performance of the model, we have plotted the Weekly_Sales predictions and true values: As expected based on the performance metric (R^2), the model performs quite well except for weeks with the highest sales, where it predicts lower. This might be because these values are outliers.

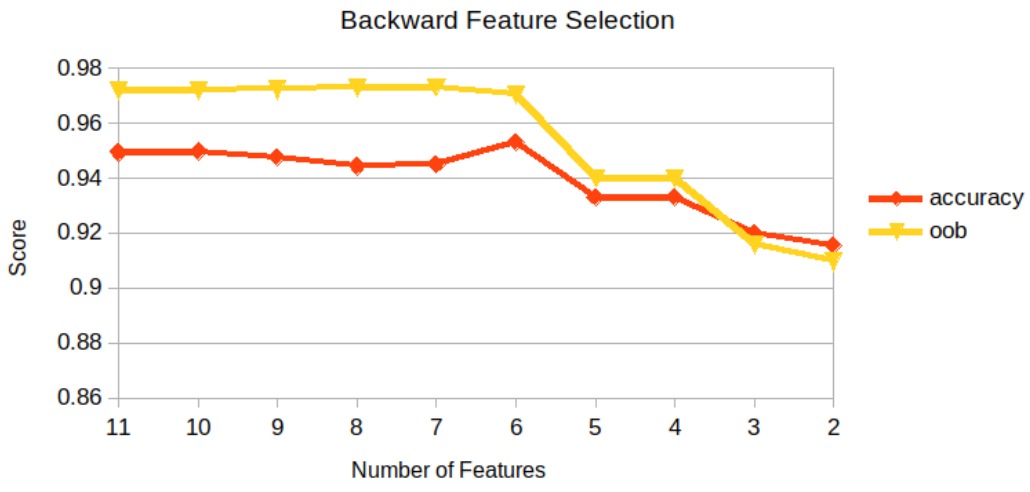


3.3.6 Improving the Accuracy

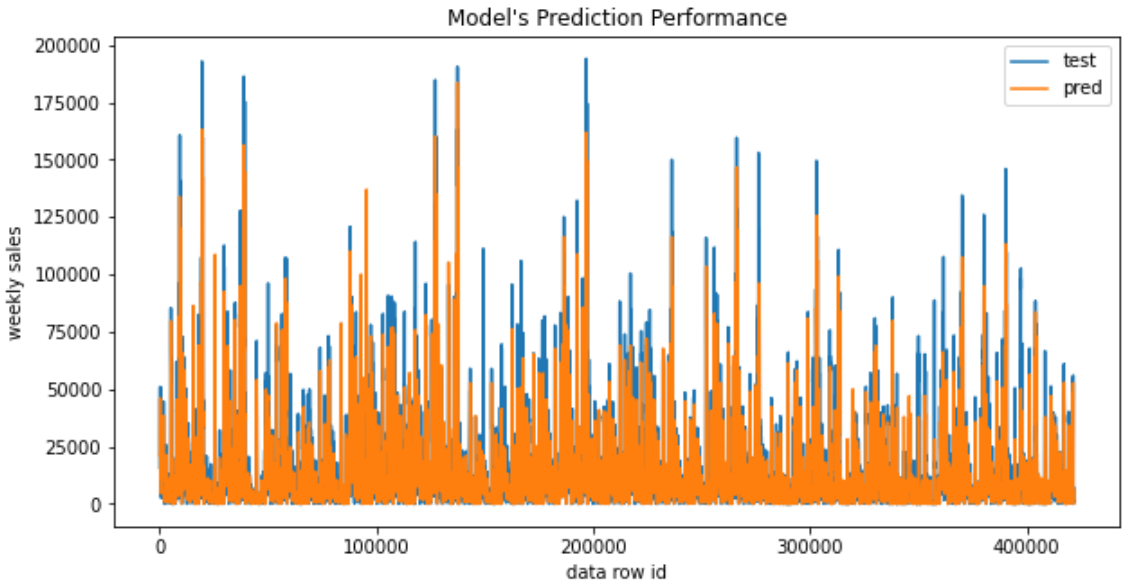
In this section, we demonstrate the effects of feature selection on improving the accuracy. We use the feature importance of the previous section and eliminate features one-by-one using backward feature selection.

n_features	Accuracy% (R^2)	OOB score	features
11	94.93	0.9722	Dept, Size_log, Store, Type, CPI, year_week, Unemployment, Temperature, Fuel_Price, IsHoliday, week
10	94.98	0.9722	Dept, Size_log, Store, Type, CPI, year_week, Unemployment, Temperature, Fuel_Price, week
9	94.77	0.9724	Dept, Size_log, Store, Type, CPI, year_week, Unemployment, Temperature, week
8	94.43	0.9730	Dept, Size_log, Store, Type, CPI, year_week, Unemployment, week
7	94.51	0.9730	Dept, Size_log, Store, week, Type, CPI, year_week
6	95.32	0.9707	Dept, Size_log, Store, week, Type, CPI
5	93.30	0.9402	Dept, Size_log, Store, week, Type
4	93.31	0.9402	Dept, Size_log, Store, week
3	92.01	0.9159	Dept, Size_log, Store
2	91.53	0.9097	Dept, Size_log

The optimum selection of features is the first 6 features, yielding an accuracy of 95.32%.



We also demonstrate that the 2 most important features (Dept and Size_log) contribute to 91.53% of prediction accuracy:



4.0 Results

Model	Accuracy (R^2)	Notes
Linear Regression	11.56%	+ Simple and interpretable + Trained fast - Inadequate for categorical features - Very low prediction accuracy - Extrapolating beyond range of data unreliable
Time-series	59%	+ Consistent with the time-dependent nature of data + Interpretable + Good for extrapolating data - Low prediction accuracy - Not consistent with time-dependent features (Dept, Size, Type)
Random Forests	97.07%	+ High prediction accuracy + Embedded feature selection + Consistent with both categorical and continuous features - Less interpretable - Longer training time

5.0 Conclusion & Discussion

In this project, we first explored the data, gaining some insight into possible connections between the features and weekly sales. We then analyzed these features from a statistical point of view to test the hypotheses we made about these connections. Later we used 3 different machine learning methods to predict the weekly sales, arriving at the conclusions below:

- There is a strong positive correlation between holiday weekends and markdowns, which was expected as Markdowns are strategically planned before holidays.
- There is also a strong positive correlation between weekly sales and the size of the store, which is consistent with our assumptions.
- To our surprise, Temperature had little correlation with weekly sales. This may be explained by cultural factors that we did not anticipate due to the data being of US origin. Unlike in the UK, American customers predominantly drive to retail stores, so extreme cold/hot weather has little impact on sales.
- RFs yield the highest accuracy.
- Time series analysis is more effective if conducted on individual stores rather than an aggregation.

Going forwards, we believe that our RF model could be an invaluable asset to store managers who wish to optimise their inventory space short term. However, it must be noted that regardless of how accurate our model is in training it should be used with caution to predict sales long term. For instance, using RFs, if we have data from 6 months prior, we can only predict 6 months into the future. Further, any model is inherently unable to forecast major economic 'Black Swan' events that can completely change the dynamic of the retail industry. For instance, the 2008 financial crisis or the recent Covid-19 outbreak caused significant drops on sales volume which our model could not have possibly predicted. To combat this, our model may need to incorporate an element of probabilistic risk analysis to account for these random events.

6.0 Bayesian Network Structure Learning Analysis

feature	n_categories	categories	Notes
Dept	3	• 1 • 2 • 3	1. To reduce the dimensionality, we opted to use data only for 3 stores and 3 departments. 2. The *Store* feature is the id of the store and has been removed. 3. The *Dept* feature refers to the type of departments. 4. We incorporated the concept of time by introducing a new feature: *Q* (yearly quarters). 5. There are no entries with CPI in the range 145 to 180, so we created two categories accordingly. 6. We used our understanding of feature correlations to better define the Bayesian structure.
IsHoliday	2	• 0 • 1	
Type	3	• A • B • C	
Temperature	7	• freezing (-20,0] • cold (0,10] • cool (10,15] • normal (15,20] • warm (20,25] • hot (25,30] • very hot (30,40]	
Fuel_Price	3	• low (2.4,2.8] • medium (2.8,4] • high (4,5]	
CPI	2	• low (125,145] • high (180,230]	
Unemployment	3	• low (3.5,6] • average (6,8.5] • high (8.5,15]	
Weekly_Sales_log	5	• very low (-0.015,2.5] • low (2.5,5.25] • medium (5.25,8] • high (8,10.75] • very high (10.75,13.5]	
Size_log	5	• very small (10.45,10.85] • small (10.85,11.25] • average (11.25,11.65] • large (11.65,12.05] • very large (12.05,12.45]	
Q (quarter)	4	• 1 • 2 • 3 • 4	

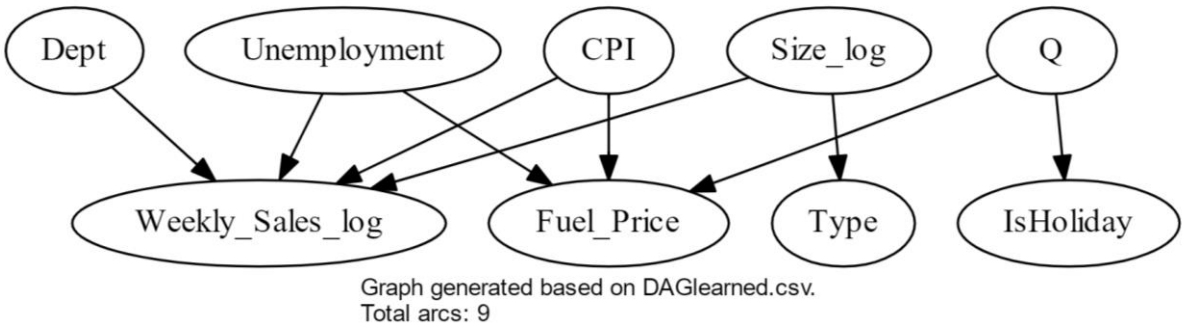
6.1 Dataset

Below is a snapshot of the dataset after categorization. (Code here)

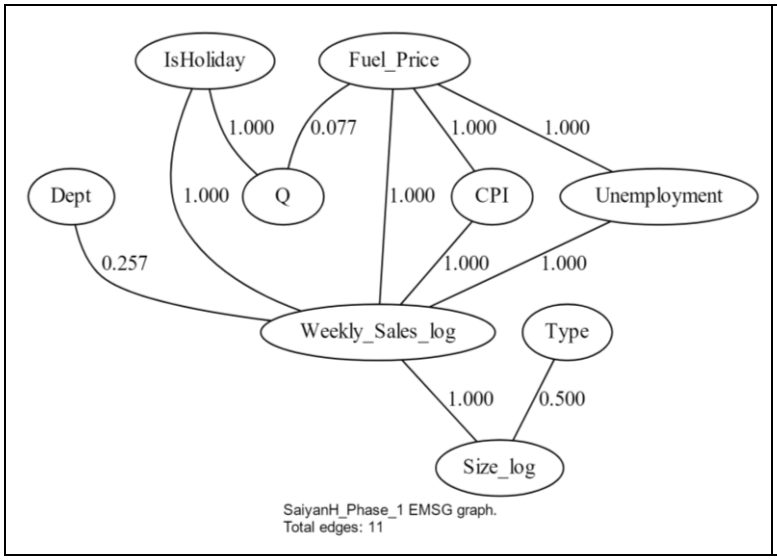
	Dept	IsHoliday	Type	Temperature	Fuel_Price	CPI	Unemployment	Weekly_Sales_log	Size_log	Q
0	1	0	0	cold	low	high	average	high	large	1
1	1	1	0	cold	low	high	average	high	large	1
2	1	0	0	cold	low	high	average	high	large	1
3	1	0	0	cold	low	high	average	high	large	1
4	1	0	0	cold	low	high	average	high	large	1
...	...	...	...	...	...	...	...	...	...	...
20906	3	0	1	hot	medium	high	average	high	small	3
20907	3	0	1	warm	medium	high	average	high	small	3
20908	3	0	1	warm	medium	high	average	high	small	3
20909	3	0	1	warm	medium	high	average	high	small	3
20910	3	0	1	warm	medium	high	average	high	small	4

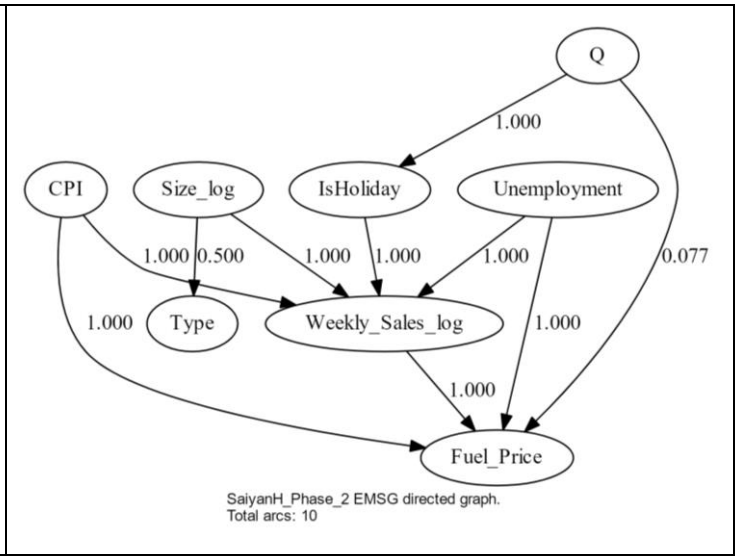
1287 rows × 10 columns

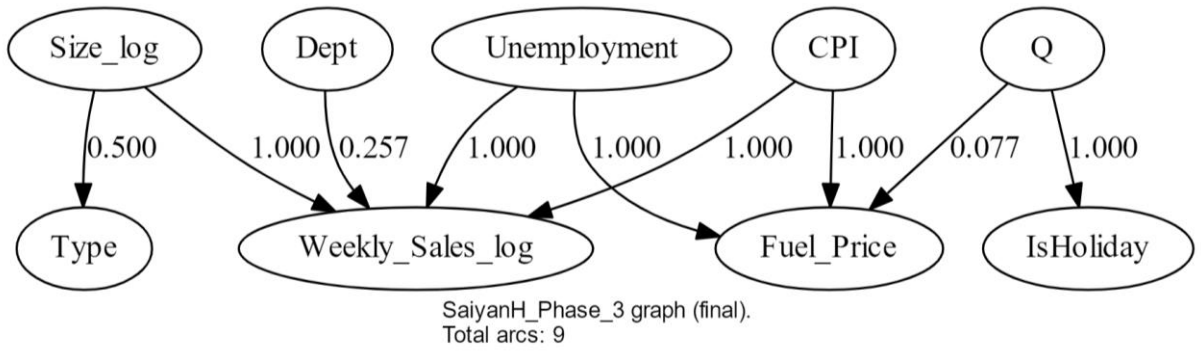
6.2 Questions



To produce the knowledge-based graph we first had to determine what variables were appropriate to include into our DAGtrue.csv file. It was decided that variables which could not be discretized into 7 or fever categories were removed to reduce computational complexity of the model. Secondly, to establish directional arcs between nodes (constraintsDirected.csv) we used the collective knowledge of the group members. When there were disagreements, we deferred to the members with the most domain knowledge. For example, when applying causal constraints amongst the macro-economic factors, like Unemployment, CPI & Fuel_Price.







Phase-1 visualizes an undirected graph (ESMG) that is entirely based on MeanMax Marginal Discrepancy (MMD) associational scores of the established connections (DAGtrue.csv). The MMD represents the discrepancy in marginal probabilities between prior and posterior distributions (range[0,1]), where higher scores indicate stronger dependencies.

The Phase-2 graph visualizes the EMSG after edges have been oriented and insignificant connections have been pruned. To produce this graph, SaiyanH uses the MMD scores to perform constraint-based learning through conditional independence tests across all pairs of nodes conditional on the remaining nodes and classify each triple into either conditional dependence, independence or insignificance.

The Phase-3 graph is the result of score-based learning that modifies the graph from Phase-2 towards the path that maximizes the BIC score. SaiyanH uses a hill-climb method that explores neighbouring graphs in which an edge is reversed, removed, or added, and moves in the direction that increases the BIC score. Once the BIC score cannot be increased attempts are made to escape possible local maximum by performing TABU search. These steps are repeated until the TABU search cannot discover a graph with a higher BIC score. The product is a graph that enables full propagation of evidence by design.

There are more scores in conditionalInsignificance.csv than marginalDep.csv because it is the result of conditional independence tests on sets of triples (in Phase 2), rather than on pairs of nodes (in Phase 1), so there are a greater combinations of possible variables. Further, a majority of variables are classified as conditionally insignificant, due to the strict classification rules of dependence/independence. There are more conditionally dependent variables than conditionally independent variables due to our methodology of selecting variables for our knowledge-based graph, as we only chose variables that we suspected to have causal connections.

CSV File	**Score**
marginalDep.csv	36
conditionalDep.csv	4
conditionalIndep.csv	1
conditionalInsignificance.csv	93

```
Phase 3 completed.
Arcs randomised during phase 2 constraint-based learning: 0
Structure learning elapsed time: 1 seconds total (Phase 1 = 0 secs, Phase 2 = 0 secs).
_____________________________ Evaluation _____________________________
Nodes: 8
Sample size: 1287
TrueDAG arcs: 12
TrueDAG independencies: 24
LearnedDAG arcs: 9
LearnedDAG independencies: 27
_____________________ Confusion matrix stats _____________________
Arcs discovered (TP): 5.0
Partial arcs discovered (TP*0.5): 3.0
False dependencies discovered (FP): 1.0
Independencies discovered (TN): 23.0
Dependencies not discovered (FN): 5.5. [NOTE: # of edges missed is 4.0]
_____________ Stats from metrics and scoring functions _____________
Precision score: 0.722
Recall score: 0.542
F1 score: 0.619
SHD score: 6.500
DDM score: 0.000
BSF score: 0.500
# of independent graphical fragments: 1
_____________________ Inference-based evaluation _____________________
BIC/MDL score -7697.494
# of free parameters 28
BUILD SUCCESSFUL (total time: 9 seconds)
```

The accuracy scores (F1 & BSF) are in agreement with our initial expectations as we did not include many exogenous variables, and only selected variables that we knew had strong causal connections. Additionally, we were confident that we would achieve a low SHD score due to the fact that we only used 8 nodes, which is comparatively low compared to the models in Figure 2.

Metric	**Score**	**Relation to Fig 2 (1k samples)**
F1	0.619	On par with Fig 2
SHD	6.500	Significantly lower than Fig 2
BSF	0.500	On par with Fig 2

We have 1 second structural learning elapsed time which is consistent with the results shown in table 2. This is because the runtime of the model with a similar sample size, number of nodes and number of parameters is also 1 second (3rd row).

Nodes	True edges	Max in-degree	# free param	Sample size	Runtime (sec)
37	46	4	509	0.1k	1
8	8	2	18	0.1k	1
8	8	2	18	1k	1
8	8	2	18	10k	1
8	8	2	18	100k	1

This was expected as we intentionally designed our model to be as simple as possible to reduce dimensionality and computational complexity - we had no variable with more than 7 categories.

BIC is a criterion for model selection among a finite set of models. Our BIC/MDL score at step 4 is higher than at step 3. Step 3 represents the model as we inputted before any constraint-based learning has taken place. Step 4 represents the model after Hill Climbing has completed and edges have been orientated to maximise the BIC score. Therefore, this was in agreement with our expectations as it was clear that our knowledge-based graph was simply our intuition and may not accurately represent the reality of the dataset, hence the BIC score would inevitably increase after learning.

	BIC Score
Step 3	-7924
Step 4	-7697

Free parameters are variables which cannot be predicted precisely or constrained by the model and must be estimated. The number of free parameters is lower at step 4 than at step 3. This was expected as the model in step 3 is unpruned and has not been optimised w.r.t BIC like the model in step 4, so requires less free parameters to express the information in the dataset.

	# of free parameters
Step 3	63
Step 4	28

7.0 References

Breiman, L. (2001) ‘Random forests’, Machine learning, (45), pp. 5–32.

Brockwell, P. J. and Davis, R. A. (2013) Introduction to Time Series and Forecasting. Springer Science & Business Media.

Cryer, J., 1986. Time series analysis. 2nd ed. s.l.:Springer.

D.Geurts, M. & Kelly, J., 1986. Forecasting retail sales using alternative models. International Journal of Forecasting, 2(3), pp. Pages 261-272.

Eshel, G. (2011) ‘Autocorrelation’, Spatiotemporal Data Analysis. doi: 10.23943/princeton/9780691128917.003.0008.

Hamilton, J. D., 1994. TIME SERIES ANALYSIS. Journal of Econometric Theory, 11(3), pp. 625-630.

Lin, S.-P. (1986) Improved Procedures for Estimating a Correlation Matrix.

Nunnari, G. and Nunnari, V. (2017) ‘Forecasting Monthly Sales Retail Time Series: A Case Study’, 2017 IEEE 19th Conference on Business Informatics (CBI). doi: 10.1109/cbi.2017.57.

Pavlov, Y. L. (2000) ‘Random Forests’. doi: 10.1515/9783110941975.

Saeys, Y., Abeel, T. and Van de Peer, Y. (2008) ‘Robust Feature Selection Using Ensemble Feature Selection Techniques’, Machine Learning and Knowledge Discovery in Databases, pp. 313–325. doi: 10.1007/978-3-540-87481-2_21.

Sun, Z.-L., Choi, T.-M., Au, K.-F. & Yu, Y., 2008. Sales forecasting using extreme learning machine with applications in fashion retailing. Decision Support Systems, 46(1), pp. 411-419.

Yan, X. (2009) Linear Regression Analysis: Theory and Computing. World Scientific.

Zhao, Y. and Zhang, Y. (2008) ‘Comparison of decision tree methods for finding active objects’, Advances in Space Research, pp. 1955–1959. doi: 10.1016/j.asr.2007.07.020.

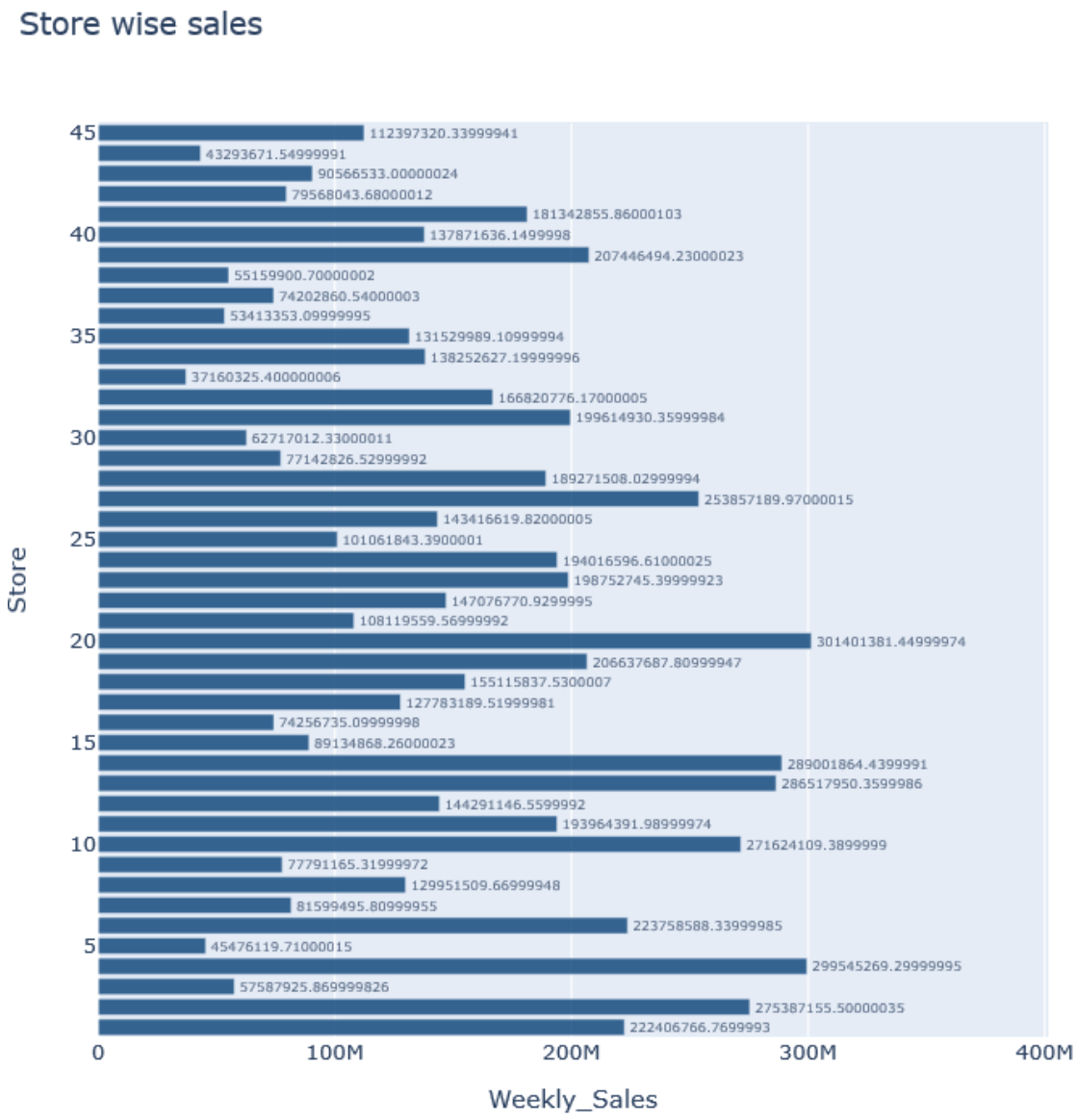
8.0 Appendices

8.1 Code

All of our codes are available at https://github.com/Dorsa-Arezooji/Retail-Data-Analytics.

8.1.1 Data Exploration

```
# Store wise sales
fig = px.bar(Store_Grouped.sort_values('Weekly_Sales', ascending=False).sort_values('Weekly_Sales', ascending=True),
             x="Weekly_Sales", y="Store", title='Store wise sales', text='Weekly_Sales', orientation='h',
             width=700, height=700, range_x = [0, max(Store_Grouped['Weekly_Sales'])+100000000])
fig.update_traces(marker_color='#084177', opacity=0.8, textposition='outside')
fig.show()
```



```
sales_feature = sales_feature.drop(['IsHoliday_y'], axis=1)
sales_feature = sales_feature.rename(columns = {'IsHoliday_x':'IsHoliday'})
sales_feature['IsHoliday'] = sales_feature['IsHoliday'].astype(int)
```

```
# Compute the correlation matrix
corr = sales_feature.drop(['Store', 'Dept'], axis=1).corr()
# Generate a mask for the upper triangle
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
```

```
#correlation matrix
f, ax = pl.subplots(figsize=(12, 9))
sns.heatmap(sales_feature.drop(['Store', 'Dept'], axis=1).corr(), mask=mask, vmax=.8, square=True);
```

```
# visualization-weekly sales vs features
sns.pairplot(df ,height=3, aspect=1.5, y_vars=['Weekly_Sales'],
             x_vars=['Store', 'Dept'])
sns.pairplot(df ,height=3, aspect=1.5, y_vars=['Weekly_Sales'],
             x_vars=['Fuel_Price', 'Temperature'])
sns.pairplot(df ,height=3, aspect=1.5, y_vars=['Weekly_Sales'],
             x_vars=['Type', 'Size'])
sns.pairplot(df ,height=3, aspect=1.5, y_vars=['Weekly_Sales'],
             x_vars=['Unemployment', 'CPI'])
sns.pairplot(df ,height=3, aspect=1.5, y_vars=['Weekly_Sales'],
             x_vars=['IsHoliday'])
```

8.1.2 Methods

8.1.2.1 LR

```
# imports
import numpy as np
import pandas as pd
import seaborn as sns
from datetime import datetime
from sklearn.linear_model import LinearRegression
from sklearn import metrics
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings
warnings.filterwarnings('ignore')
```

```
# size
df['Size_log'] = np.log1p(df.Size)
df.drop(['Size'], axis=1, inplace=True)

# time-related features
df.Date = pd.to_datetime(df.Date, format='%d/%m/%Y')
df['week'] = df.Date.dt.week
df['year'] = df.Date.dt.year
df1 = df.copy()
df.drop(['Date'], axis=1, inplace=True)
df['year_week'] =  df['year'].astype(str) + df['week'].astype(str)
df['year_week'] =  df['year_week'].astype(int)
df.loc[df['week']<10, 'year_week'] = df.loc[df['week']<10, 'year'].astype(str) + '0' + df.loc[
    df['week']<10, 'week'].astype(str)
df['year_week'] =  df['year_week'].astype(int)
```

```
# functions
def data_split(d, t):
    test = d[d.year_week.isin(t)]
    train = d[~d.year_week.isin(t)]
    test.drop(['Weekly_Sales'], axis=1, inplace=True)
    train.drop(['Weekly_Sales'], axis=1, inplace=True)
    X_test = test.drop('Weekly_Sales_log', axis=1)
    y_test = test['Weekly_Sales_log']
    X_train = train.drop('Weekly_Sales_log', axis=1)
    y_train = train['Weekly_Sales_log']
    return X_test, y_test, X_train, y_train

def vis(y_test, y_pred, X_test):
    y = pd.DataFrame(X_test[:], columns=['test', 'pred'])
    y['test'] = np.expm1(y_test)
    y['pred'] = np.expm1(y_pred)
    ax = y.plot(figsize=(10,5),title="Model's Prediction Performance")
    ax.set_xlabel('data row id')
    ax.set_ylabel('weekly sales')
    ax

def accuracy(X_test,y_test, y_pred):
    print('accuracy (R^2):\n', model.score(X_test, y_test)*100, '%')
```

```
# linear regression
X_test, y_test, X_train, y_train = data_split(df, np.linspace(201230, 201243))
model = LinearRegression(copy_X=True, n_jobs=-1)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# performance
accuracy(X_test, y_test, y_pred)
vis(y_test, y_pred, X_test)
print('\n coefficients:\n ', model.coef_)
```

8.1.2.2 Time-series

```
# Total Sales vs Time
#summarize each feature variable
#summarize the target variable
#look for correlation between each feature and the target
#look for correlation between features
# x and y axis values are extracted from the grouped DataFrame
x = main_df.Date
y = main_df.Weekly_Sales

fig = plt.figure()
ax = fig.add_subplot(111)
fig.subplots_adjust(top=0.85)

ax.set_xlabel('Year')
ax.set_ylabel('Weekly_Sales')
plt.tick_params(axis='x', which='both', bottom=False, labelbottom=False)

ax.plot_date(x, y, xdate=True, ydate=False, color='purple')
plt.show()
```

8.1.2.3 RF

```
# imports
import numpy as np
import pandas as pd
import seaborn as sns
from datetime import datetime
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings
warnings.filterwarnings('ignore')
```

```
# size
df['Size_log'] = np.log1p(df.Size)
df.drop(['Size'], axis=1, inplace=True)

# time-related features
df.Date = pd.to_datetime(df.Date, format='%d/%m/%Y')
df['week'] = df.Date.dt.week
df['year'] = df.Date.dt.year
df1 = df.copy()
df.drop(['Date'], axis=1, inplace=True)
df['year_week'] =  df['year'].astype(str) + df['week'].astype(str)
df['year_week'] =  df['year_week'].astype(int)
df.loc[df['week']<10, 'year_week'] = df.loc[df['week']<10, 'year'].astype(str) + '0' + df.loc[
    df['week']<10, 'week'].astype(str)
df['year_week'] =  df['year_week'].astype(int)
```

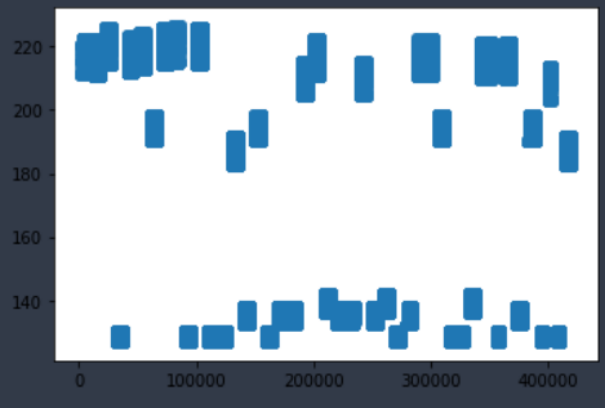
```
# functions
def data_split(d, t):
    test = d[d.year_week.isin(t)]
    train = d[~d.year_week.isin(t)]
    test.drop(['Weekly_Sales'], axis=1, inplace=True)
    train.drop(['Weekly_Sales'], axis=1, inplace=True)
    X_test = test.drop('Weekly_Sales_log', axis=1)
    y_test = test['Weekly_Sales_log']
    X_train = train.drop('Weekly_Sales_log', axis=1)
    y_train = train['Weekly_Sales_log']
    return X_test, y_test, X_train, y_train

def vis(y_test, y_pred, X_test):
    y = pd.DataFrame(X_test[:], columns=['test', 'pred'])
    y['test'] = np.expm1(y_test)
    y['pred'] = np.expm1(y_pred)
    ax = y.plot(figsize=(10,5),title="Model's Prediction Performance")
    ax.set_xlabel('data row id')
    ax.set_ylabel('weekly sales')
    ax

def accuracy(X_test,y_test, y_pred):
    print('accuracy (R^2):\n', model.score(X_test, y_test)*100, '%')
```

```
# Random Forest Regressor model
X_test, y_test, X_train, y_train = data_split(df, np.linspace(201230, 201243))
model = RandomForestRegressor(n_estimators=20, criterion='mse', bootstrap=True, n_jobs=-1,
                              random_state=100, oob_score=True)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

accuracy(X_test, y_test, y_pred)
print('\n out of bag score:\n ', model.oob_score_)
print('\n feature importances:\n ', model.feature_importances_)
vis(y_test, y_pred, X_test)
```

8.1.3 Categorization

```
# categorizing CPI
plt.plot(df['CPI'], '.')
print('number of CPIs in [145,180]: ', len(df[(df['CPI']> 145) & (df['CPI']<180)]))
```

```
number of CPIs in [145,180]:  0
```




```
# categorizing the column values
df1 = df.loc[df['Store'].isin([1,2,3])]
df1 = df1.loc[df['Dept'].isin([1,2,3])]
df1.dropna(inplace=True)
df1['Weekly_Sales_log'] = pd.cut(df1['Weekly_Sales_log'], [-0.015, 2.5, 5.25, 8, 10.75, 13.5],
                                 labels=['very low', 'low', 'medium', 'high', 'very high'])
df1['Size_log'] = pd.cut(df1['Size_log'], [10.45, 11.20, 11.70, 12.45],
                         labels=['small', 'average', 'large'])
df1['Unemployment'] = pd.cut(df1['Unemployment'], [3.5, 6, 8.5, 15], labels=
                             ['low', 'average', 'high'])
df1['CPI'] = pd.cut(df1['CPI'], [125, 145, 180, 230], labels=['low', '', 'high'])
df1['Temperature'] = pd.cut(df1['Temperature'], [-20, 0, 10, 15, 20, 25, 30, 40], labels=
                            ['freezing', 'cold', 'cool', 'normal', 'warm', 'hot', 'very hot'])
df1['Fuel_Price'] = pd.cut(df1['Fuel_Price'], [2.4, 2.8, 4, 5], labels=['low', 'medium', 'high'])
df1.dropna(inplace=True)
```

```
# Q (quarters)
df1['Q'] = 4
df1.loc[df1['year_week']<=201242, 'Q'] = 3
df1.loc[df1['year_week']<=201228, 'Q'] = 2
df1.loc[df1['year_week']<=201214, 'Q'] = 1
df1.loc[df1['year_week']<=201200, 'Q'] = 4
df1.loc[df1['year_week']<=201142, 'Q'] = 3
df1.loc[df1['year_week']<=201128, 'Q'] = 2
df1.loc[df1['year_week']<=201114, 'Q'] = 1
df1.loc[df1['year_week']<=201000, 'Q'] = 4
df1.loc[df1['year_week']<=201042, 'Q'] = 3
df1.loc[df1['year_week']<=201028, 'Q'] = 2
df1.loc[df1['year_week']<=201014, 'Q'] = 1

df1.drop('week', axis=1, inplace=True)
df1.drop('year_week', axis=1, inplace=True)
df1.drop('Store', axis=1, inplace=True)
df1.drop('Weekly_Sales', axis=1, inplace=True)
df1.to_csv('d.csv', index=False)
df1.to_csv('dataset_categorized.csv', index=False)
```

8.2 Data

- Features_dataset.csv

Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	IsHoliday
1	05/02/2010	42.31	2.572	NA	NA	NA	NA	NA	211.0963582	8.106	FALSE
1	12/02/2010	38.51	2.548	NA	NA	NA	NA	NA	211.2421698	8.106	TRUE
1	19/02/2010	39.93	2.514	NA	NA	NA	NA	NA	211.2891429	8.106	FALSE
1	26/02/2010	46.63	2.561	NA	NA	NA	NA	NA	211.3196429	8.106	FALSE
1	05/03/2010	46.5	2.625	NA	NA	NA	NA	NA	211.3501429	8.106	FALSE
1	12/03/2010	57.79	2.667	NA	NA	NA	NA	NA	211.3806429	8.106	FALSE
1	19/03/2010	54.58	2.72	NA	NA	NA	NA	NA	211.215635	8.106	FALSE
1	26/03/2010	51.45	2.732	NA	NA	NA	NA	NA	211.0180424	8.106	FALSE
1	02/04/2010	62.27	2.719	NA	NA	NA	NA	NA	210.8204499	7.808	FALSE
1	09/04/2010	65.86	2.77	NA	NA	NA	NA	NA	210.6228574	7.808	FALSE
1	16/04/2010	66.32	2.808	NA	NA	NA	NA	NA	210.4887	7.808	FALSE
1	23/04/2010	64.84	2.795	NA	NA	NA	NA	NA	210.4391228	7.808	FALSE
1	30/04/2010	67.41	2.78	NA	NA	NA	NA	NA	210.3895456	7.808	FALSE
1	07/05/2010	72.55	2.835	NA	NA	NA	NA	NA	210.3399684	7.808	FALSE

- Sales_data-set.csv

Store	Dept	Date	Weekly_Sales	IsHoliday
1	1	05/02/2010	24924.5	0
1	1	12/02/2010	46039.49	1
1	1	19/02/2010	41595.55	0
1	1	26/02/2010	19403.54	0
1	1	05/03/2010	21827.9	0
1	1	12/03/2010	21043.39	0
1	1	19/03/2010	22136.64	0
1	1	26/03/2010	26229.21	0
1	1	02/04/2010	57258.43	0
1	1	09/04/2010	42960.91	0
1	1	16/04/2010	17596.96	0
1	1	23/04/2010	16145.35	0
1	1	30/04/2010	16555.11	0
1	1	07/05/2010	17413.94	0

- Store_data-set.csv

Store	Type	Size
1	A	151315
2	A	202307
3	B	37392
4	A	205863
5	B	34875
6	A	202505
7	B	70713
8	A	155078
9	B	125833
10	B	126512
11	A	207499
12	B	112238
13	A	219622
14	A	200898