Walmart Analytics: Report

GitHub:
ParthSharma-2 /Walmart-Sales-Analytics

Contents

1.0 Data Engineering **2**
1.1 Data Overview 2
1.2 Data Exploration 3
2.0 Linear Regression **5**
2.1 Description 5
2.2 Justification 5
2.3 Feature Selection 5
2.4 Model 5
2.5 Results 6
3.0 Random Forests **7**
3.1 Description 7
3.2 Justification 7
3.3 Feature Selection 7
3.4 Model 8
3.5 Results 8
3.6 Improving the Accuracy 10
4.0 Comparison of Models **12**
5.0 Bayesian Network Structure Learning **13**
6.0 References **16**

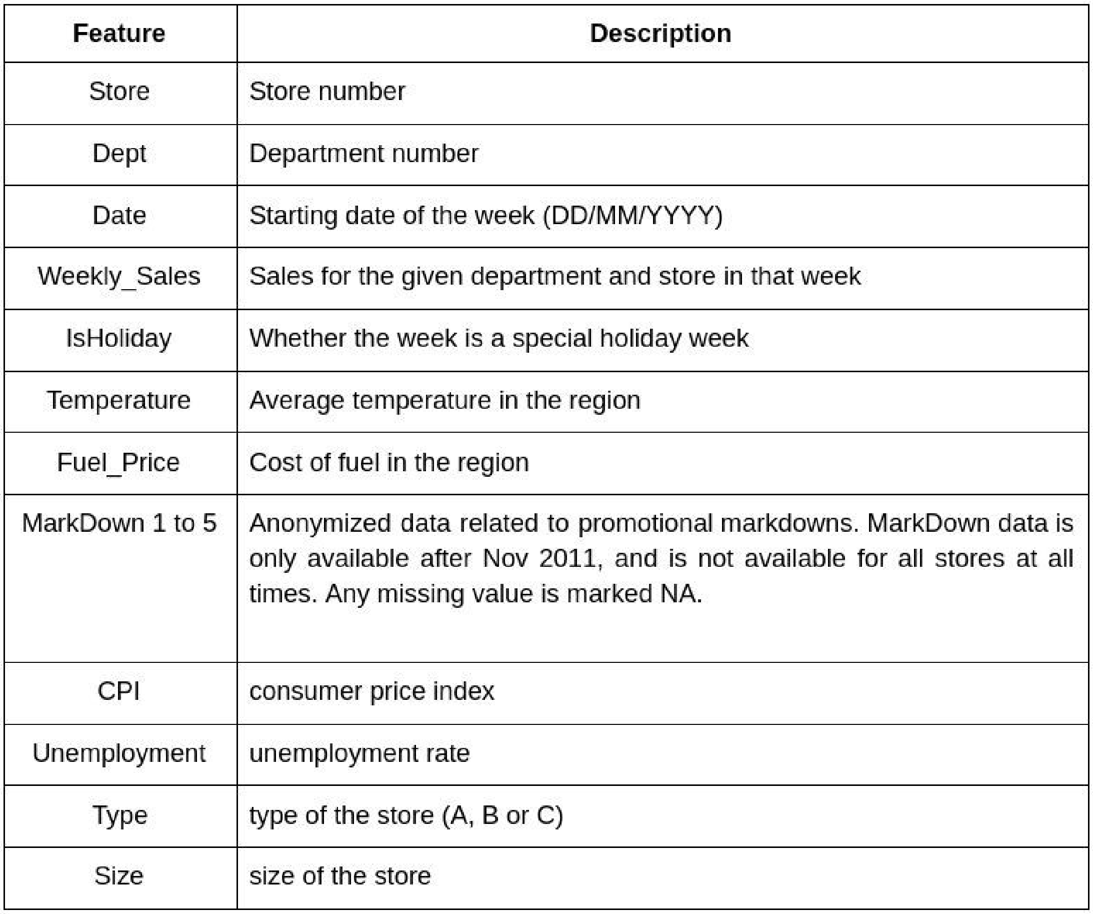
1.0 Data Engineering

1.1 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 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

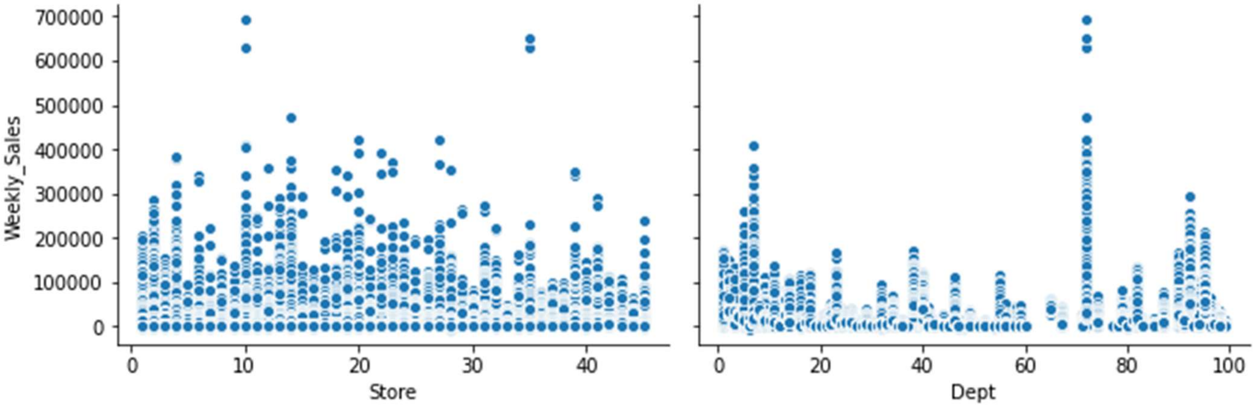
Predictors are:



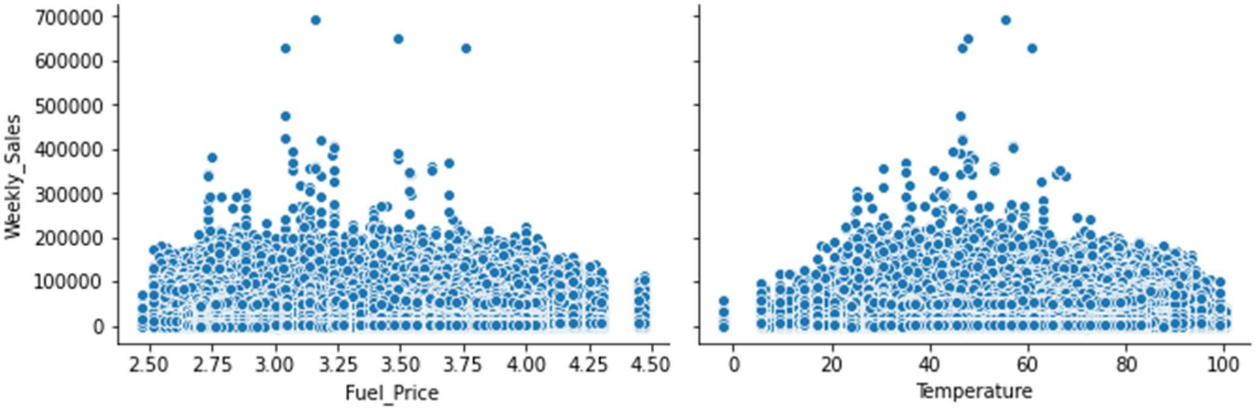
Feature	Description
Store	Store number
Dept	Department number
Date	Starting date of the week (DD/MM/YYYY)
Weekly_Sales	Sales for the given department and store in that week
IsHoliday	Whether the week is a special holiday week
Temperature	Average temperature in the region
Fuel_Price	Cost of fuel in the 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

1.2 Data Exploration

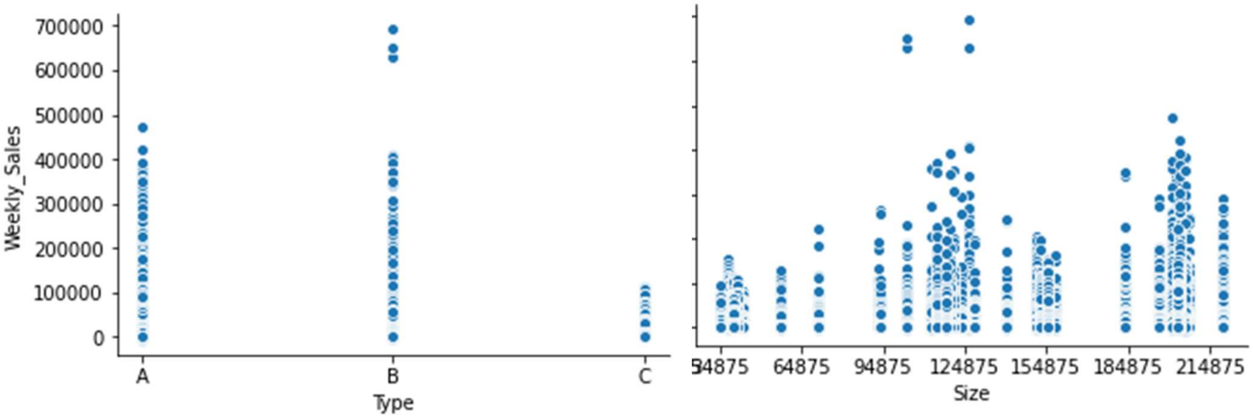
Exploring the dataset for possible relations between *Weekly_Sales* and other features:



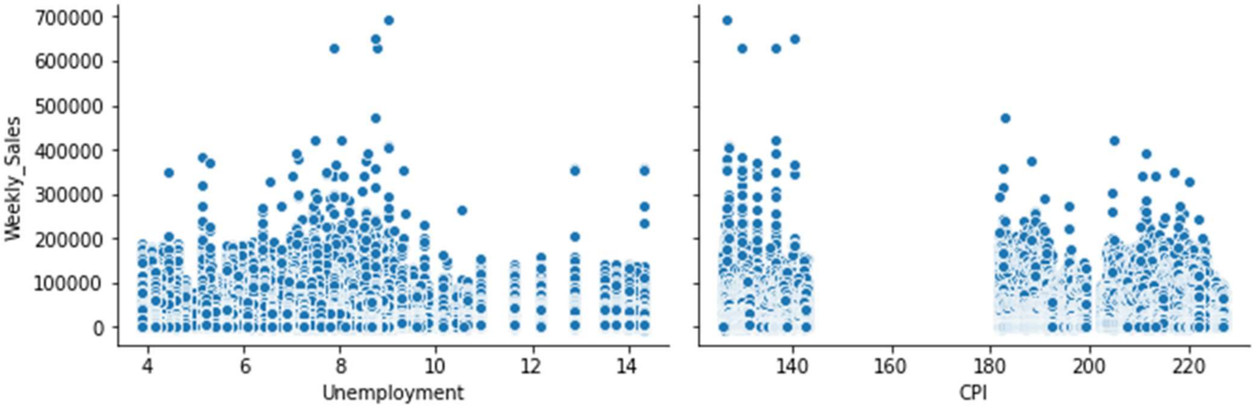
- A particular department makes noticeably higher sales.



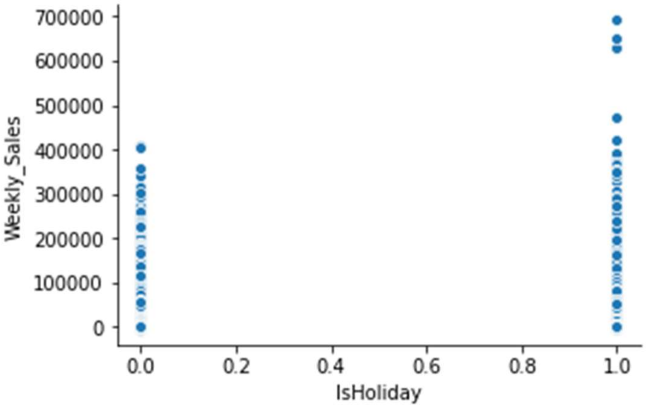
- Observing the small variations in the left figure, *Fuel_Price* does not seem to be affecting *Weekly_Sales*. However, a number of high *Weekly_Sales* are associated with fuel prices ranging from 2.75 - 3.75. This is not enough to make an assumption about the effects of fuel price, as this increase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

2.0 Linear Regression

2.1 Description

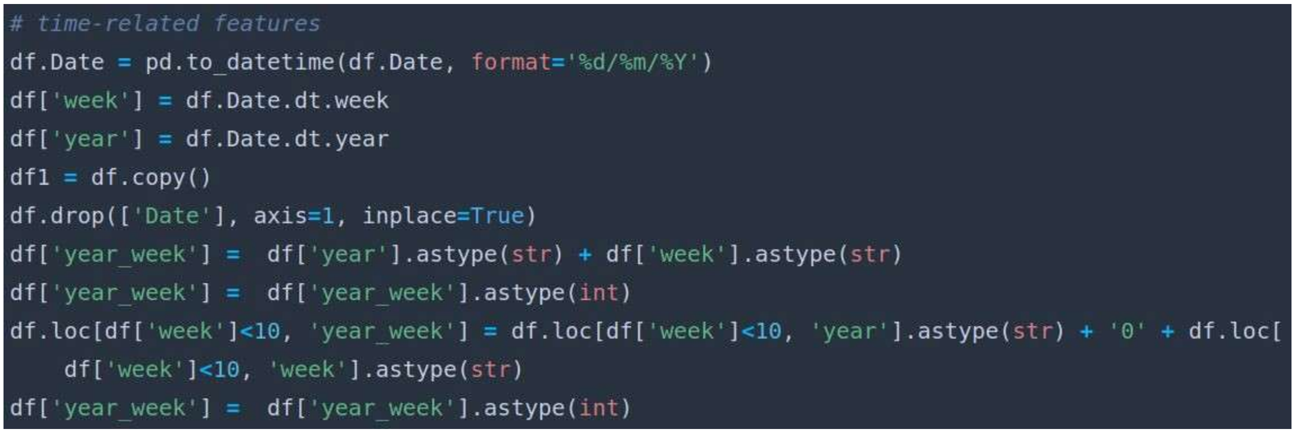
Linear Regression (LR) is one of the simplest models for machine learning. A multiple LR assumes linear relationships between the features and the target, then fits a line to the data that would result in the lowest residual sum of squared errors (Yan, 2009).

2.2 Justification

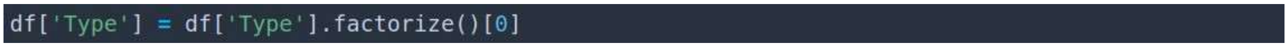
A first intuition is that a simple, easily interpretable model like LR is a good starting point for analysis. Also, this would provide a basis for future model comparisons.

2.3 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)
```

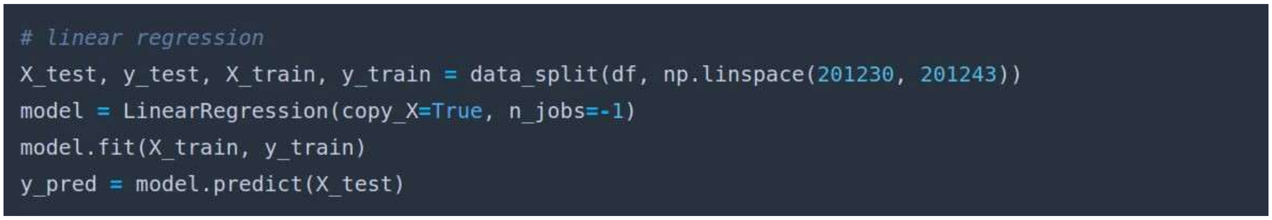


```
df['Type'] = df['Type'].factorize()[0]
```

The LR model is trained on these 11 features: *Store, Dept, IsHoliday, Type, Temperature, Fuel_Price, CPI, Unemployment, Size_log, week, year_week*.

2.4 Model

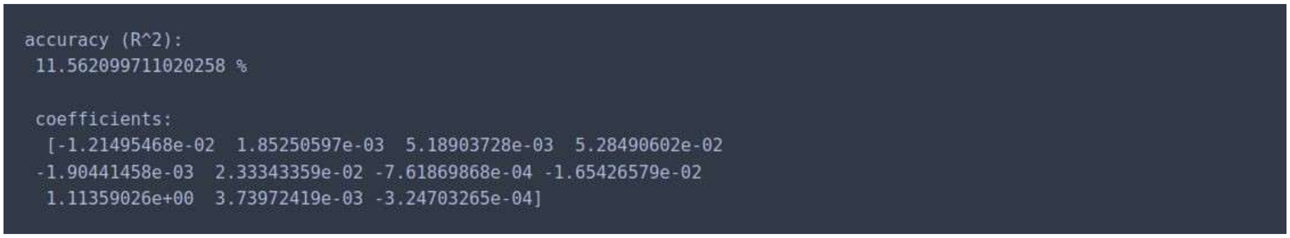
The testing dataset includes all data from the 30th week of 2012 to the 43rd and the training set includes the data prior to that from the 5th week of 2010. By splitting the dataset like so, what happens during production is simulated (making predictions based on available features and previous data), 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)
```

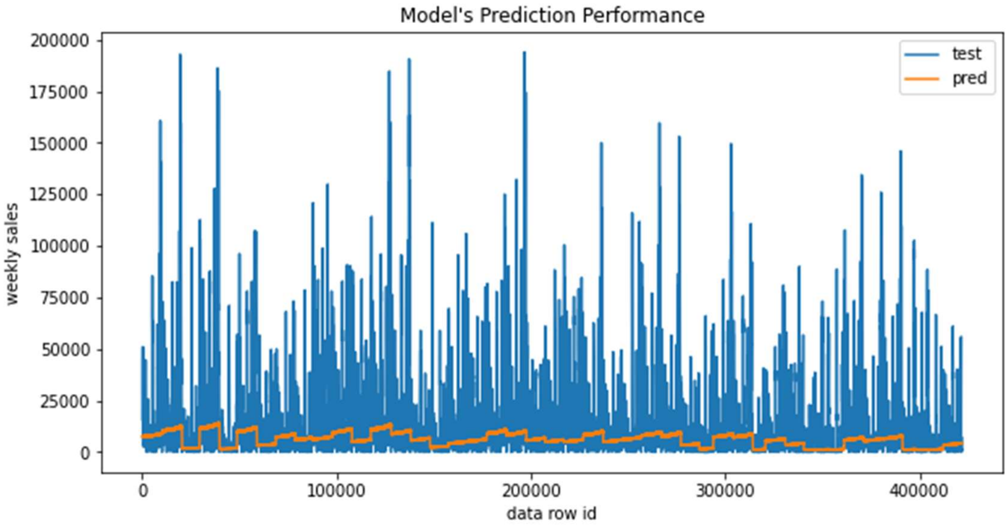
2.5 Results

The resulting accuracy is quite low:



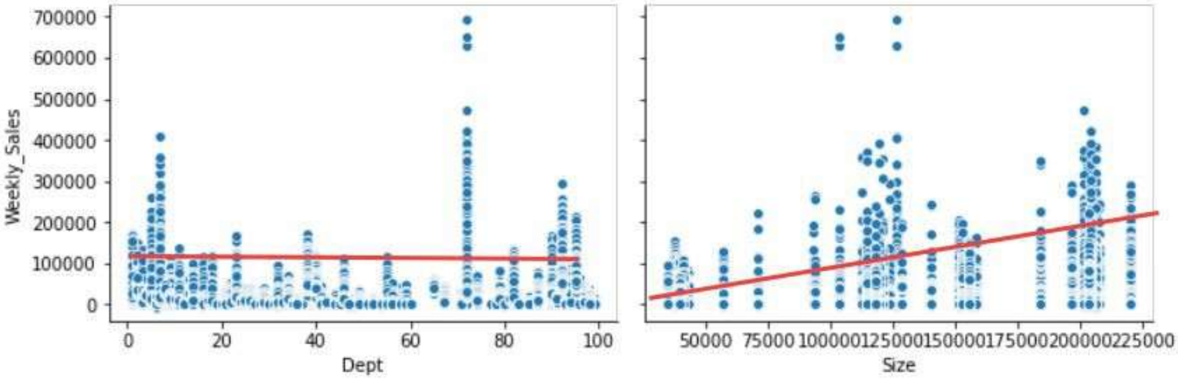
```
accuracy (R^2):
 11.562099711020258 %

 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 can not fit them well, we would expect the accuracy of this model to be very low as was speculated.

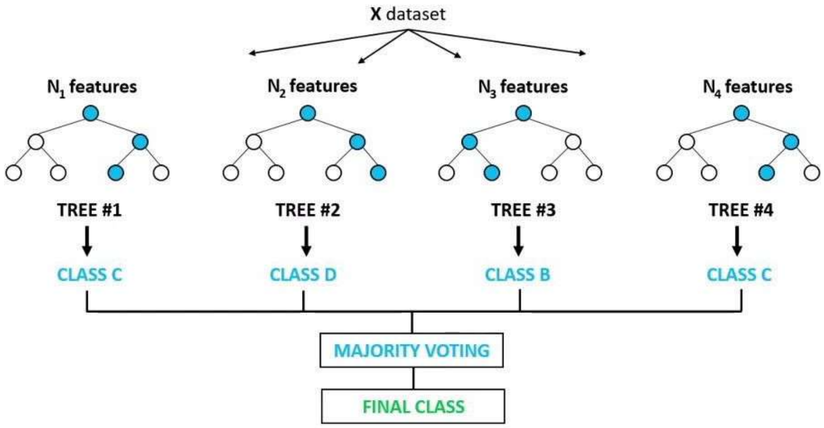
Furthermore, backward elimination would actually decrease the accuracy of the LR model since the most important features are mostly categorical. In this case the accuracy would drop from 11.56% to 11.41% by eliminating the bottom 5 features, which yielded the highest accuracy in the RF model.



3.0 Random Forests

3.1 Description

Random forests (RFs) are primarily used for classification, however they can also be used for 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).

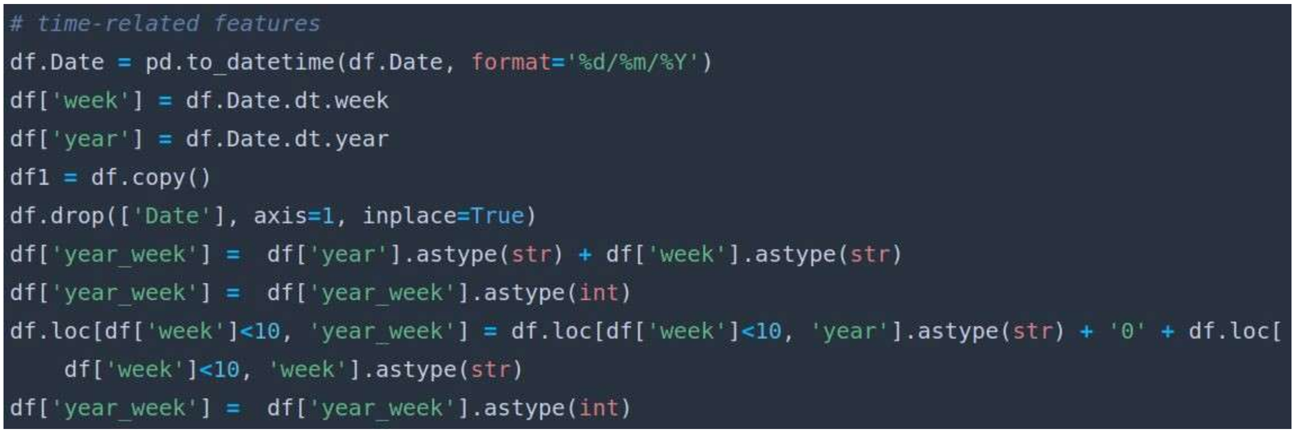


3.2 Justification

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 which is costly.

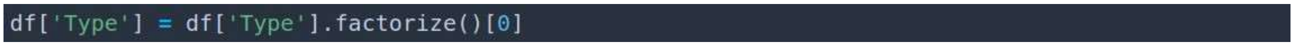
3.3 Feature Selection

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)
```

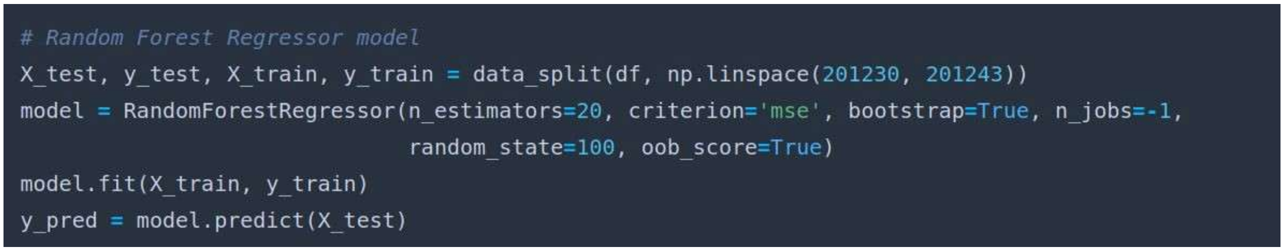
Also, the type feature is encoded as 0,1 or 2.



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

3.4 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, a Random Forest Regressor model is trained to predict the Weekly_Sales_log for all the stores. The following eleven features are used as predictors: *Store, Dept, IsHoliday, Type, Temperature, Fuel_Price, CPI, Unemployment, Size_log, week, year_week.*



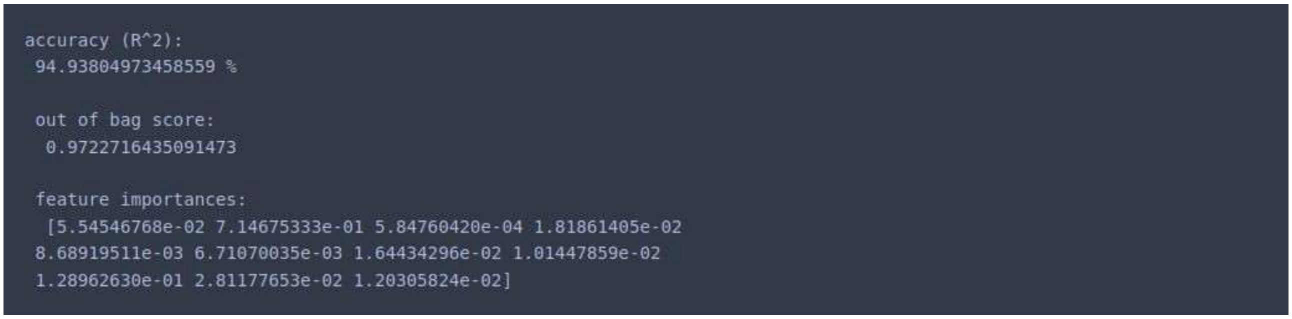
```
# Random Forest Regressor model
X_test, y_test, X_train, y_train = data_split(df, np.linspace(201230, 201243))
model = RandomForestRegressor(n_estimators=20, criterion='mse', bootstrap=True, n_jobs=-1,
                              random_state=100, oob_score=True)
model.fit(X_train, y_train)
y_pred = 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 (lower CPU load).
- The R^2 criterion is taken as the performance measure along with 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, this risk can be minimized.

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]
```

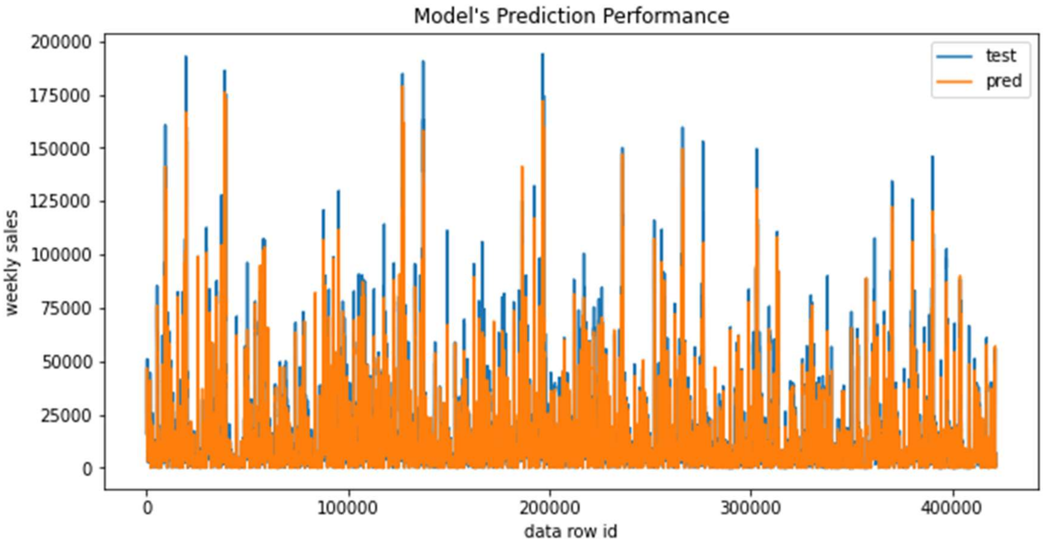
The accuracy (94.94%) is high, but can be improved via backward feature elimination which is the next topic of discussion. The OOB score (0.97) is quite good as well. These results demonstrate that the RF model performs well enough so far.

Next, the *.feature_importances_* method is used 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.
- Time has been incorporated into the model 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, a certain trend is 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 it was expected to be one of the key features. However, the fact that it does not contribute much to prediction accuracy, might be partly because of the fact that 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, the *Weekly_Sales* predictions and true values are plotted:



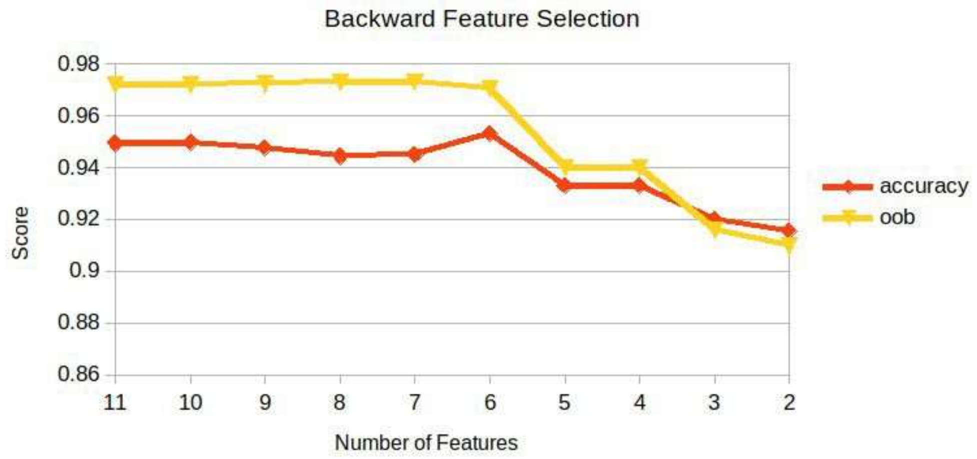
As expected based on the performance metric (R^2), the model performs quite well except for weeks with the highest sales, where it predicts lower sales than actual. This might be because these values are outliers.

3.6 Improving the Accuracy

In this section, the effects of feature selection on improving the accuracy are demonstrated. The feature importances of the previous section are used to eliminate features one-by-one using backward feature selection.

A summary of the backward elimination process is depicted in the table below:

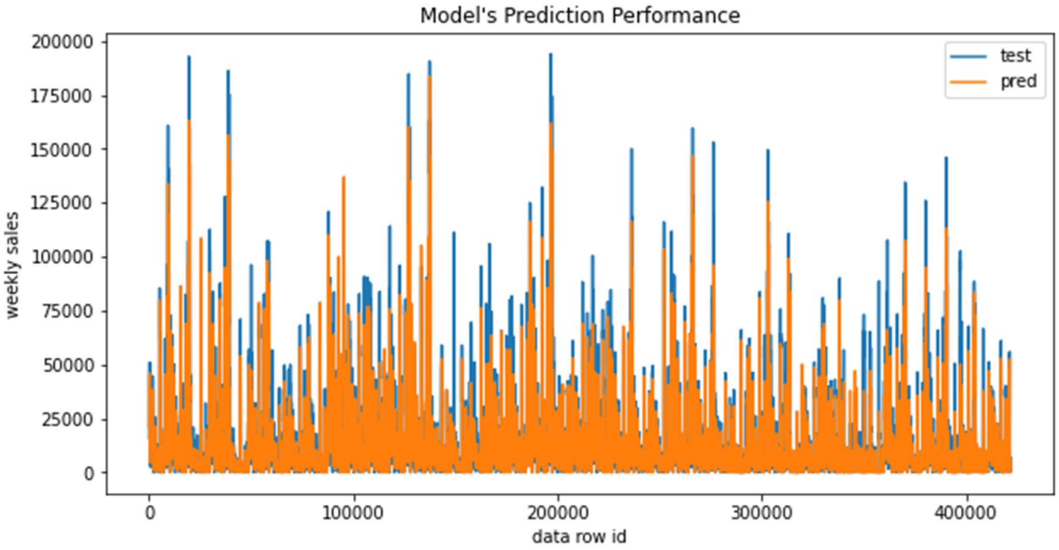
n_features	Accuracy% (R^2)	OOB score	features
11	94.93	0.9722	Dept, Size_log, Store, Type, CPI, year_week, Unemployment, Temperature, Fuel_Price, IsHoliday, week
10	94.98	0.9722	Dept, Size_log, Store, Type, CPI, year_week, Unemployment, Temperature, Fuel_Price, week
9	94.77	0.9724	Dept, Size_log, Store, 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



Notes:

- The optimum selection of features is the first 6 features, yielding an accuracy of 95.32%.
- The 2 most important features (*Dept* and *Size_log*) contribute to 91.53% of prediction accuracy.

The figure below illustrates the performance of the RF model using only the two most important features:

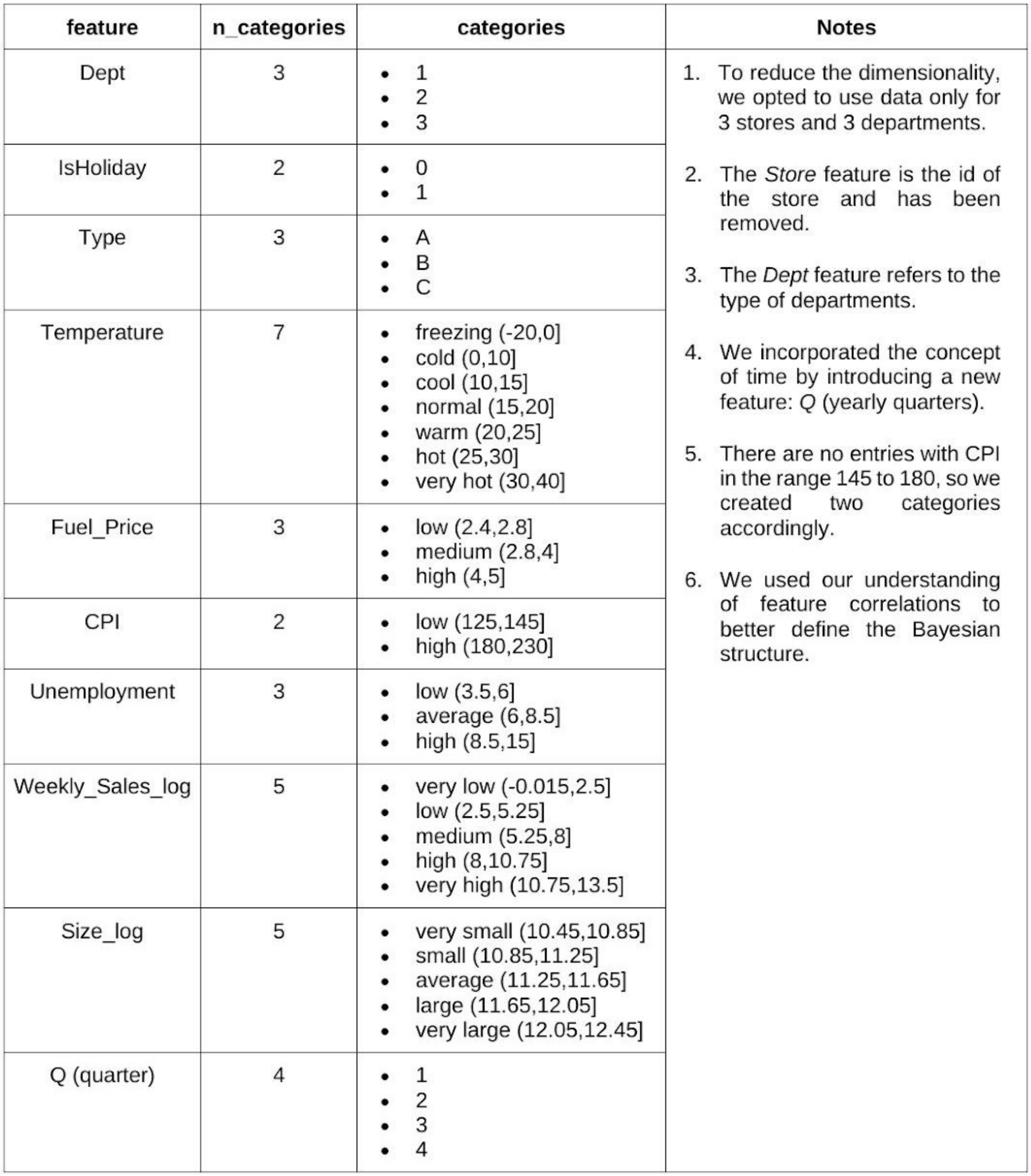


4.0 Comparison of Models

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
Random Forests	97.07%	+ High prediction accuracy + Embedded feature selection + Consistent with both categorical and continuous features - Less interpretable - Longer training time

5.0 Bayesian Network Structure Learning

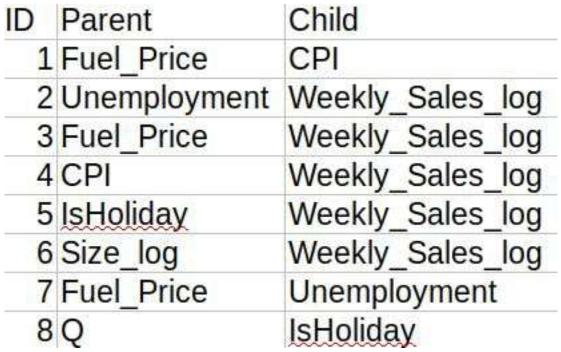
In this section, Bayesys is used for learning the Bayesian structure of the dataset. First, the dataset needs to be categorized in order to reduce the dimensional complexity. Below are the categories along with the resulting categorized dataset:



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	

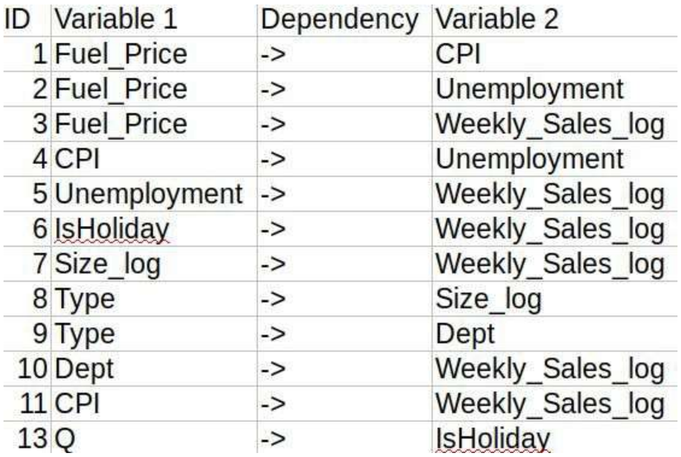
The categorized dataset and the following csv files are the inputs to the algorithm:

- constraintsDirected.csv:



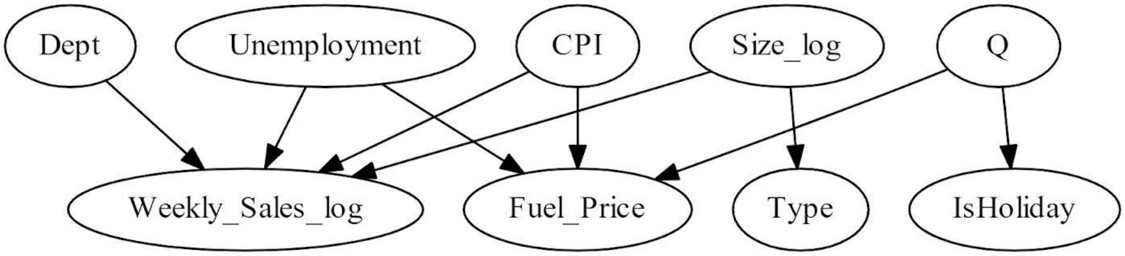
ID	Parent	Child
1	Fuel_Price	CPI
2	Unemployment	Weekly_Sales_log
3	Fuel_Price	Weekly_Sales_log
4	CPI	Weekly_Sales_log
5	IsHoliday	Weekly_Sales_log
6	Size_log	Weekly_Sales_log
7	Fuel_Price	Unemployment
8	Q	IsHoliday

- DAGtrue.csv



ID	Variable 1	Dependency	Variable 2
1	Fuel_Price	->	CPI
2	Fuel_Price	->	Unemployment
3	Fuel_Price	->	Weekly_Sales_log
4	CPI	->	Unemployment
5	Unemployment	->	Weekly_Sales_log
6	IsHoliday	->	Weekly_Sales_log
7	Size_log	->	Weekly_Sales_log
8	Type	->	Size_log
9	Type	->	Dept
10	Dept	->	Weekly_Sales_log
11	CPI	->	Weekly_Sales_log
13	Q	->	IsHoliday

The resulting directed graph:



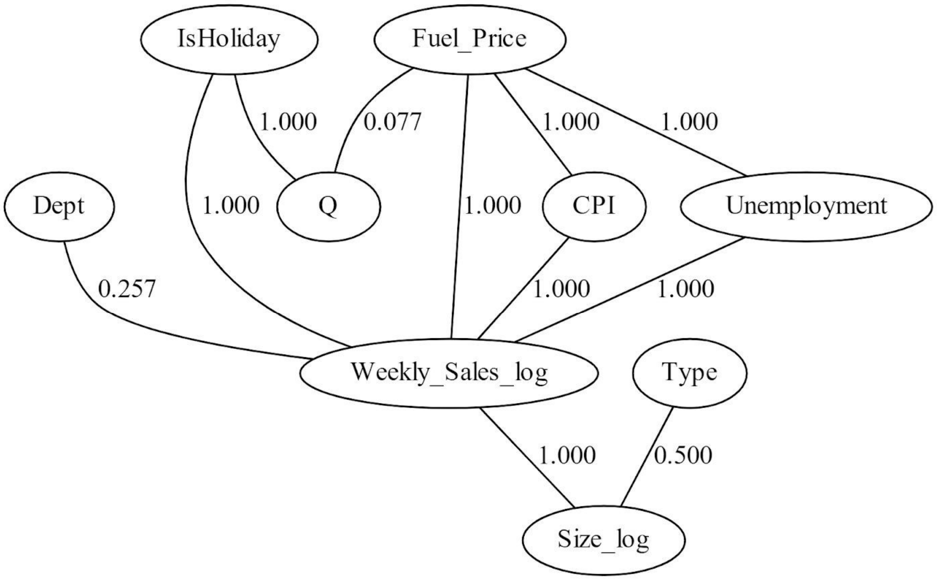
Graph generated based on DAGlearned.csv.
Total arcs: 9

Notes:

- Based on the final learned graph, there is a causal link between *Q* (quarter) and *IsHoliday* and also *Q* and Fuel_Price, meaning that these two features are time-dependent (are affected by *Q*) as was expected.
- *Size_log* affects the *Weekly_Sales_log* and also determines the *Type* of the store (A, B, or C).

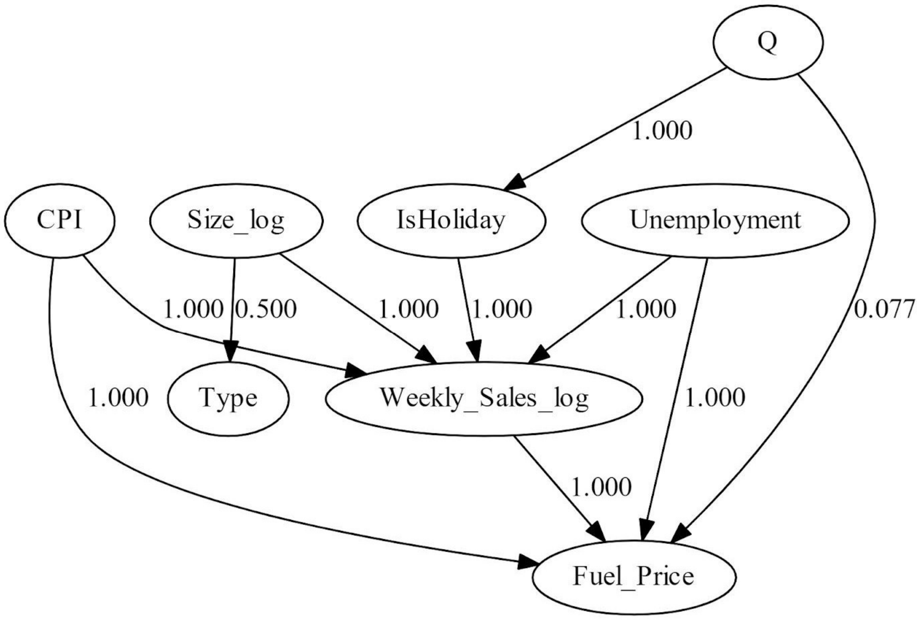
- Some causal links, however, are not quite as easy to understand: *Unemployment* affects *Fuel_Price*. There might be a financial/economical explanation behind this finding.
- The features contributing directly to (affecting) the *Weekly_Sales_log* are: *Dept*, *Unemployment*, *CPI*, and *Size_log*.

- Phase 1 EMSG graph:



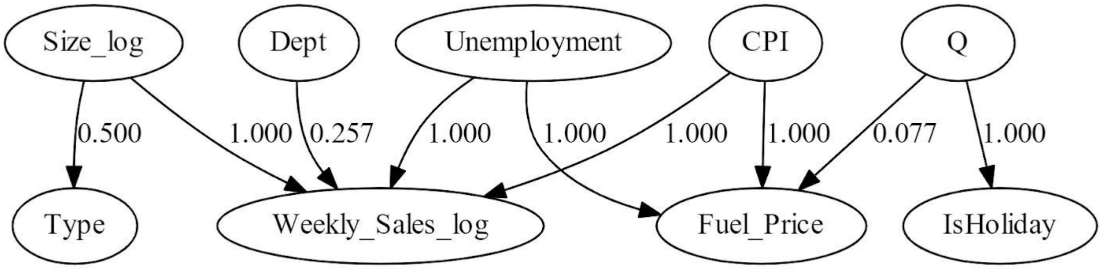
SaiyanH_Phase_1 EMSG graph.
Total edges: 11

- Phase 2 EMSG directed graph:



SaiyanH_Phase_2 EMSG directed graph.
Total arcs: 10

- Phase 3 graph (final):



SaiyanH_Phase_3 graph (final).
Total arcs: 9

6.0 References

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

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.

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.