CSE 575: Statistical Machine Learning

Anomaly Detection:
How to find rare/unusual instances/groups

Group 3:

Amrit Bhaskar, Riyank Mukhopadhyay, Shivani Madhunala

Shahana Ragupathy Sumathy, Vagisha Gupta

Introduction

The technique to identify rare items or observations that do not conform to normal or common patterns found in data is referred to as **Anomaly Detection**.

Anomaly detection plays an important role for reliability when dealing with crucial data. Having outliers/anomalies in a dataset can cause significant impact while working with the data.

Anomalies can be detected using various data visualization and exploratory data analysis techniques.

Problem Statement

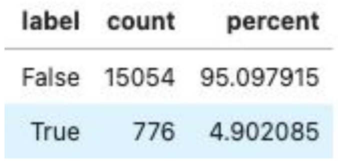
To compare, analyze and contrast between different supervised and unsupervised models for Anomaly detection using labeled and unlabeled data.

Methodology

- Labeled Data - Supervised Models
  - XgBoost
  - SVM
  - Multilayer Perceptron
  - KNN
- Unlabeled Data - Unsupervised Models
  - One class SVM
  - Isolation Forest
  - DBSCAN
  - Prophet
  - GluonTS

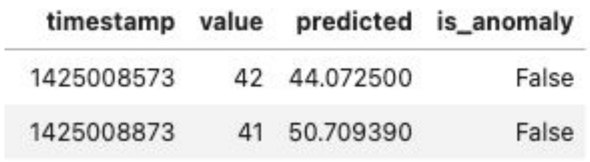
Labeled Data

- Number of rows: 15,830
- Distribution of labels:



label	count	percent
False	15054	95.097915
True	776	4.902085

- Data sample:

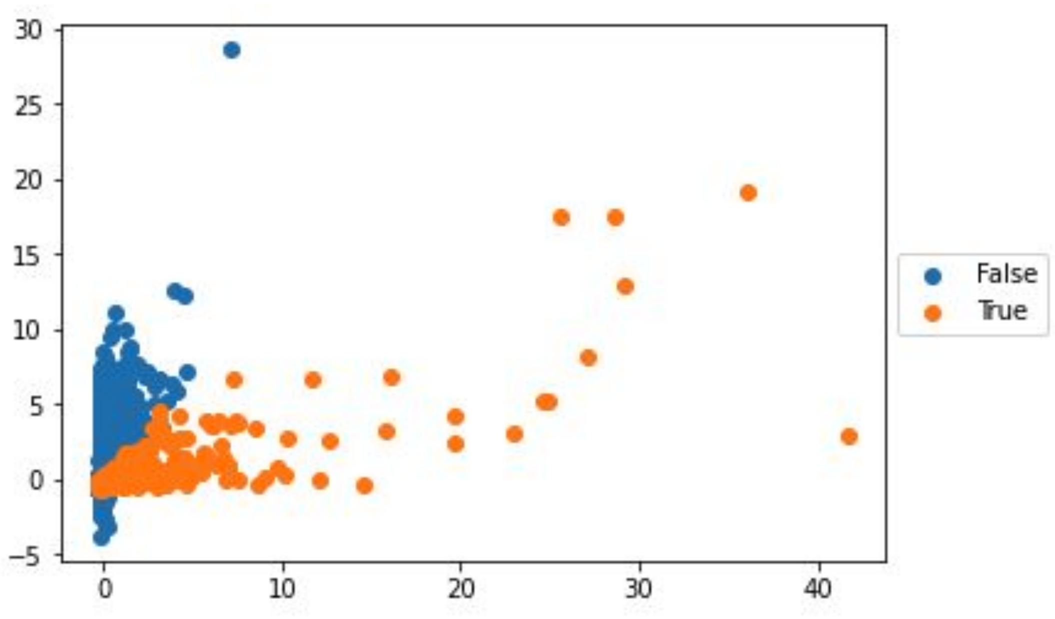


timestamp	value	predicted	is_anomaly
1425008573	42	44.072500	False
1425008873	41	50.709390	False

Data Preprocessing

- StandardScaler
  - Standardize features by removing the mean and scaling to unit variance.
  - The standard score of a sample x is calculated as:
    - z = (x - u) / s , where u is the mean of the training samples, and s is the standard deviation of the training samples.
  - Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data (e.g. Gaussian with 0 mean and unit variance).
- Remove the timestamp column
- Split the dataset into Training and Test Data: 80:20 Ratio.

Training Data

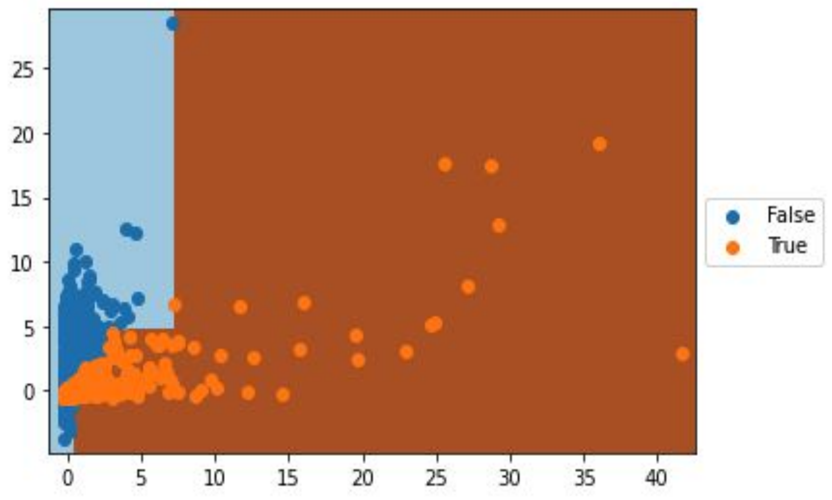


True: Outliers
False: Non-Outliers

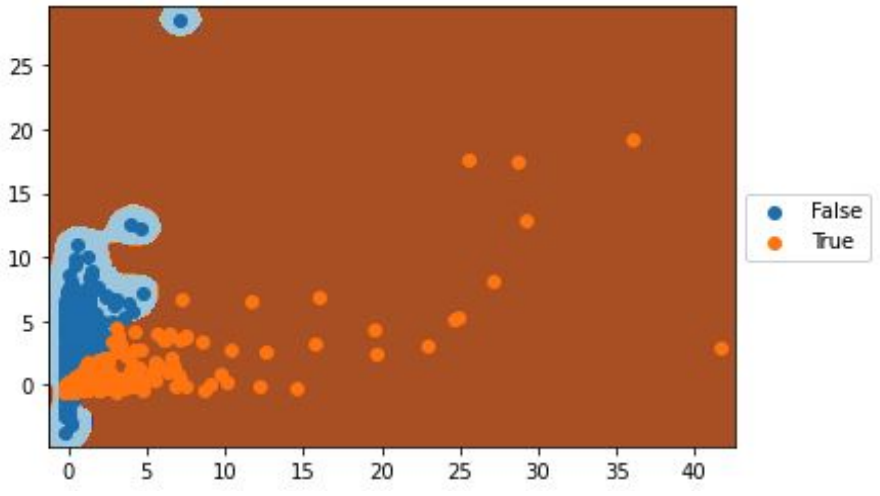
Model Results

Models	Test AUC	Test Accuracy	Test F-1 score	Confusion Matrix
XgBoost	**0.9676**	**97.53**	**0.9736**	**[[3004, 21], [57, 84]]**
SVM	0.9144	97.18	0.9658	[[3024, 1], [88, 53]]
MLP	0.9158	97.47	0.9726	[[3006, 19], [61, 80]]
KNN	0.8695	97.22	0.9716	[[2987, 38], [50, 91]]

Decision Boundaries

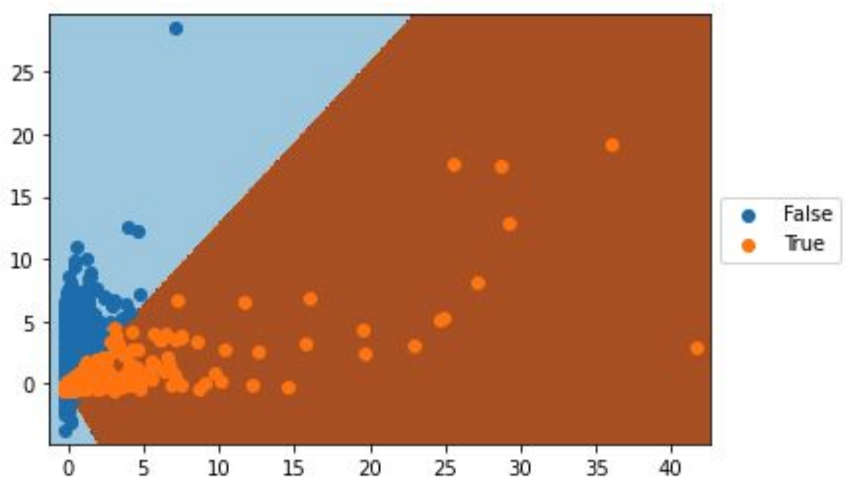


XGBoost

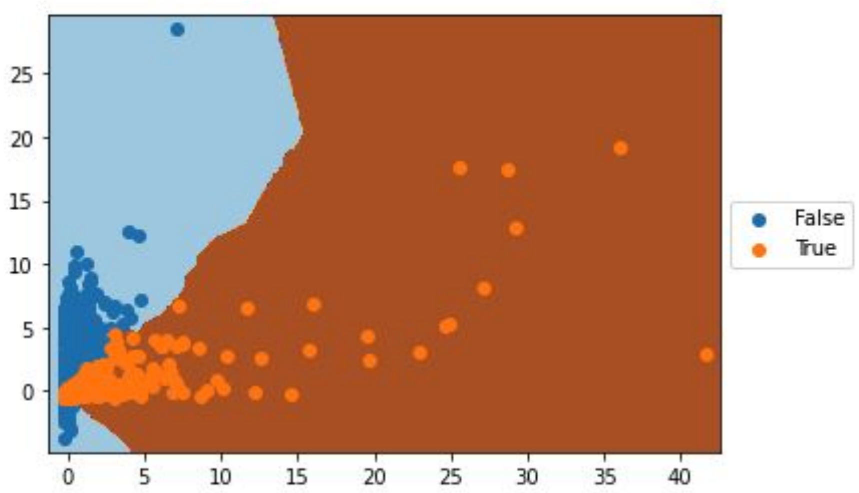


SVM

Decision Boundaries



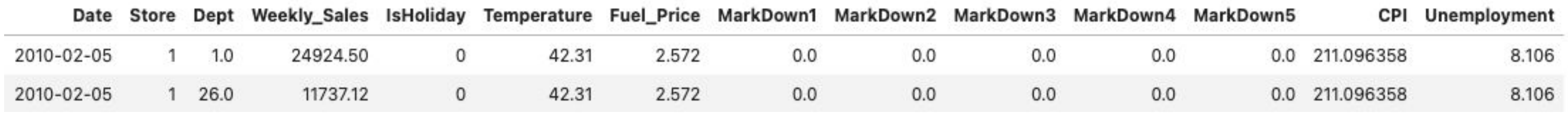
MLP



KNN

Unlabeled Data

- Number of rows: 421,570
- Historical data that contains Walmart Store sales from 2010-02-05 to 2012-11-01 (frequency: weekly)
- Data sample:



Date	Store	Dept	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment
2010-02-05	1	1.0	24924.50	0	42.31	2.572	0.0	0.0	0.0	0.0	0.0	211.096358	8.106
2010-02-05	1	26.0	11737.12	0	42.31	2.572	0.0	0.0	0.0	0.0	0.0	211.096358	8.106

- Columns:
  - **Store** - the store number
  - **Date** - the week of sales
  - **WeeklySales** - sales for the given store
  - **HolidayFlag** - whether the week is a special holiday week 1 – Holiday week 0 – Non-holiday week
  - **Temperature** - Temperature on the day of sale
  - **Fuel_Price** - Cost of fuel in the region
  - **CPI** – Prevailing consumer price index
  - **Unemployment** - Prevailing unemployment rate
  - **MarkDown(1-5)** - The markdowns precede prominent holidays

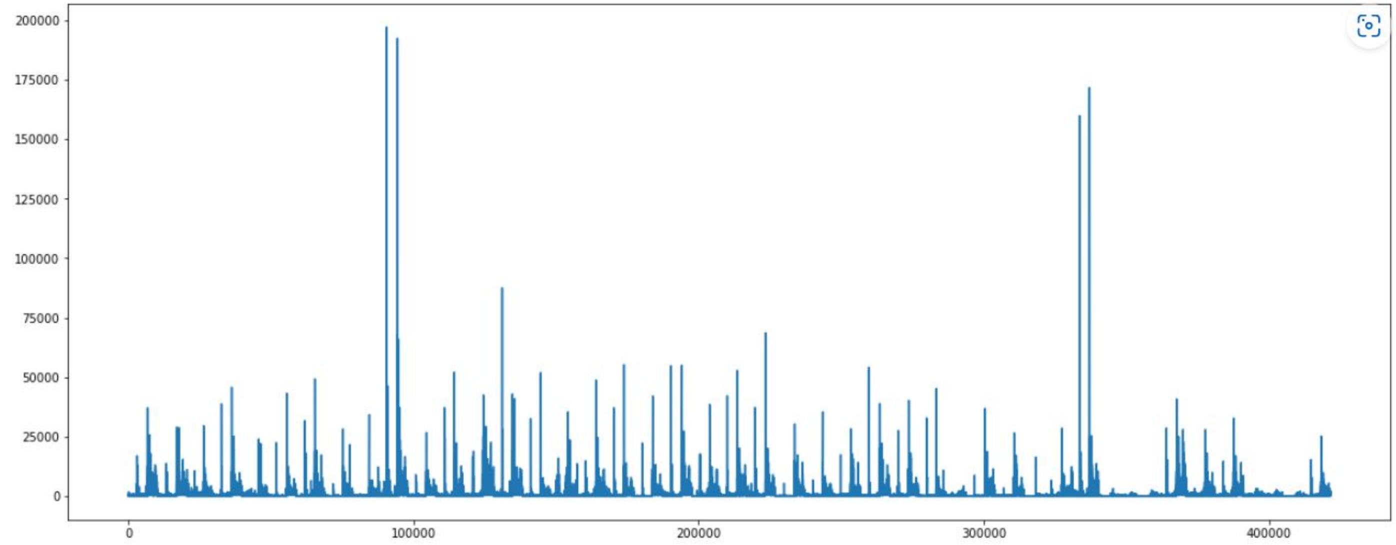
KNN - K-Nearest Neighbors

KNN is usually a supervised ML algorithm used for classification, but we have opted the **unsupervised approach for anomaly detection**.

There is no predetermined labels like ‘outlier’ or ‘not outliers’, it is entirely based upon cut off/threshold values.

KNN Visualization - Distance graph

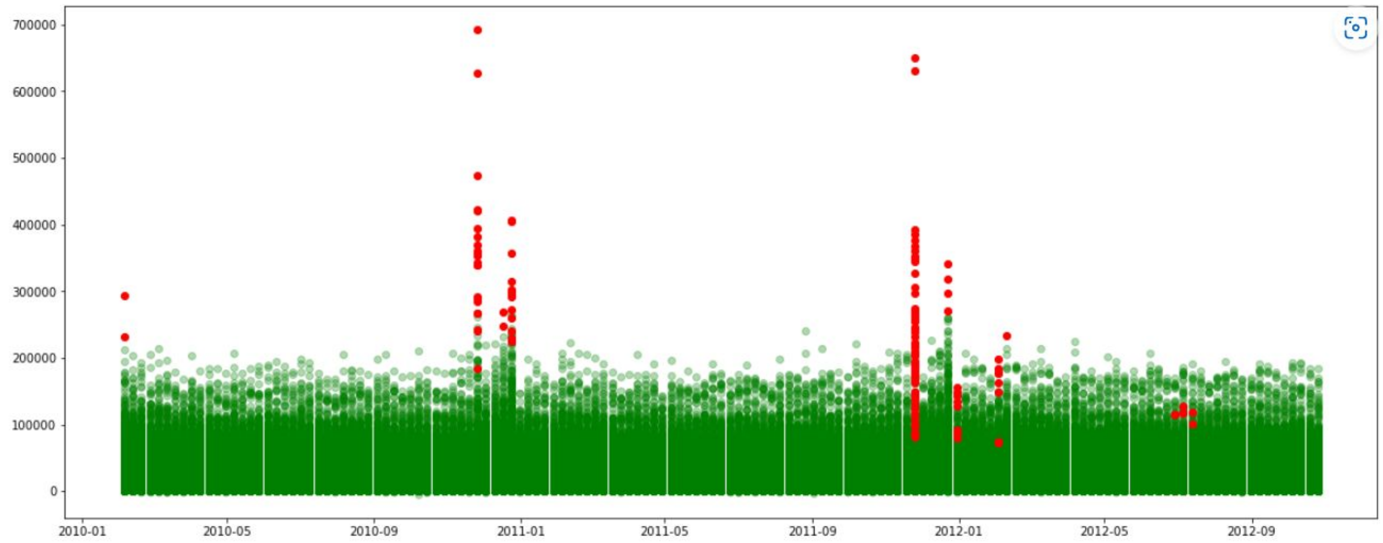
We instantiate the model by setting K = 100 and create the distance matrix.



To visually determine the cutoff values, we take distances more than 50000 and plot the graph, shown on the next slides.

KNN Visualization: Date vs. Weekly sales anomaly plot

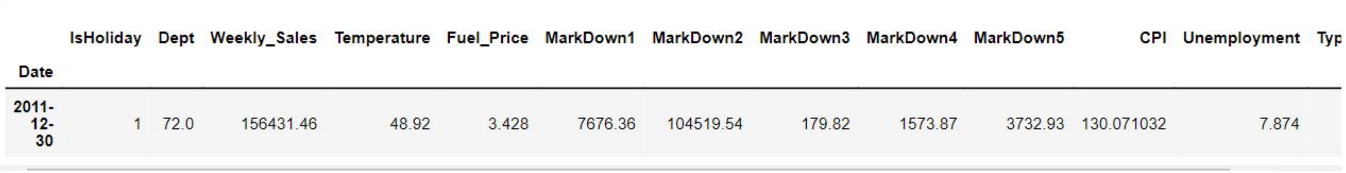
There are a total of 128 outliers in the plot.



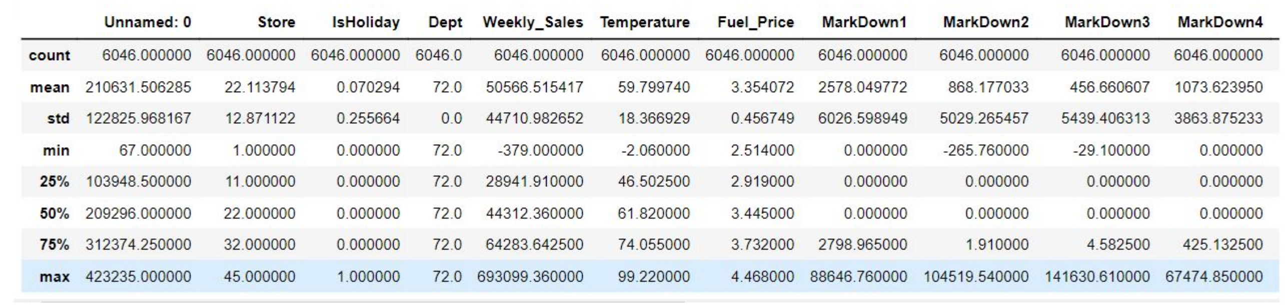
To find the contributing factors for why a data point is an outlier, we try to see the quartiles and how the columns/features affect them.

K-nearest neighbors application: Findings

For example, we come across this row of data:



	IsHoliday	Dept	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	Typ
Date													
2011-12-30	1	72.0	156431.46	48.92	3.428	7676.36	104519.54	179.82	1573.87	3732.93	130.071032	7.874	



	Unnamed: 0	Store	IsHoliday	Dept	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4
count	6046.000000	6046.000000	6046.000000	6046.0	6046.000000	6046.000000	6046.000000	6046.000000	6046.000000	6046.000000	6046.000000
mean	210631.506285	22.113794	0.070294	72.0	50566.515417	59.799740	3.354072	2578.049772	868.177033	456.660607	1073.623950
std	122825.968167	12.871122	0.255664	0.0	44710.982652	18.366929	0.456749	6026.598949	5029.265457	5439.406313	3863.875233
min	67.000000	1.000000	0.000000	72.0	-379.000000	-2.060000	2.514000	0.000000	-265.760000	-29.100000	0.000000
25%	103948.500000	11.000000	0.000000	72.0	28941.910000	46.502500	2.919000	0.000000	0.000000	0.000000	0.000000
50%	209296.000000	22.000000	0.000000	72.0	44312.360000	61.820000	3.445000	0.000000	0.000000	0.000000	0.000000
75%	312374.250000	32.000000	0.000000	72.0	64283.642500	74.055000	3.732000	2798.965000	1.910000	4.582500	425.132500
max	423235.000000	45.000000	1.000000	72.0	693099.360000	99.220000	4.468000	88646.760000	104519.540000	141630.610000	67474.850000

By looking at the quartiles we see that the value of Markdown 2 does not fall between 25% and 75% and is actually the max value.

One Class SVM

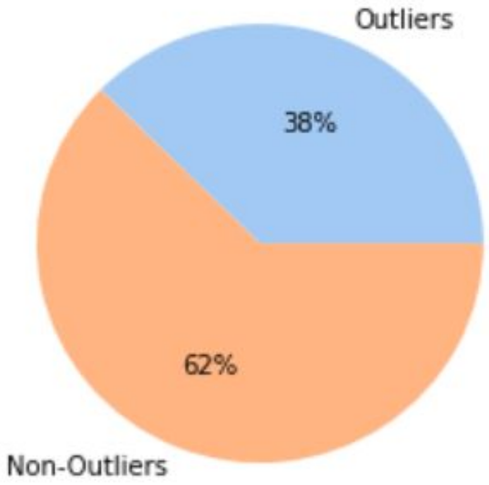
One-Class Support Vector Machine (SVM) is an unsupervised model that learns the boundary for the normal data points and identifies the data outside the border to be anomalies.

Dataset: Walmart sales data(Unlabelled .csv file)

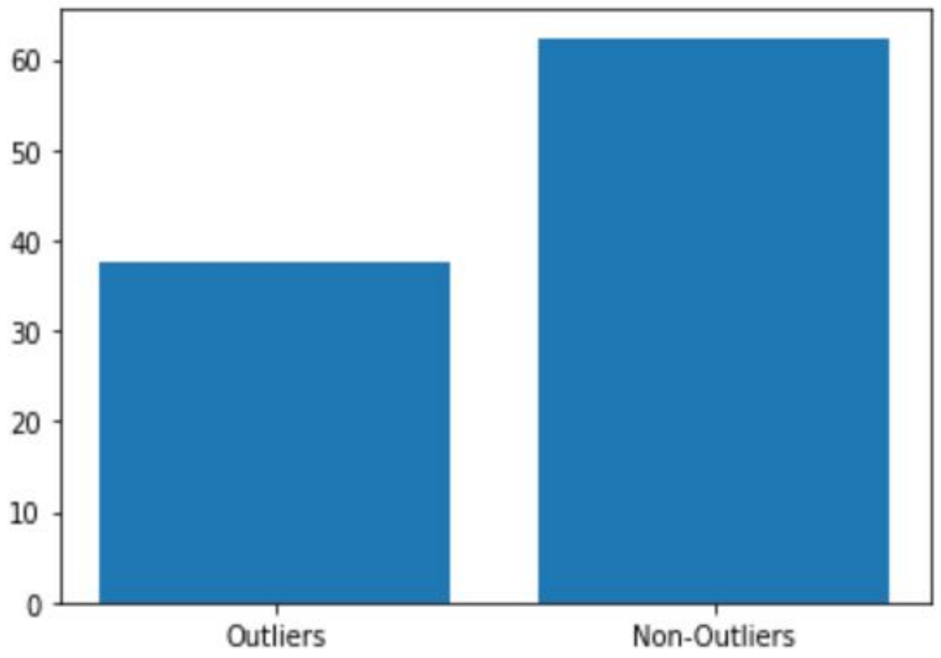
Packages used:
1. Sklearn.svm- for one class svm model
2. Matplotlib.pyplot, seaborn for visualisation

Sample size data: 2000 rows × 16 columns

Steps:
1. Train the model
2. predict anomalies
3. visualize the prediction results



One Class SVM contd.



Since the one-class SVM does not have target labels for the model training process Unlike the regular supervised SVM. We retrieve the outlier percentage as the prediction results for visualisation

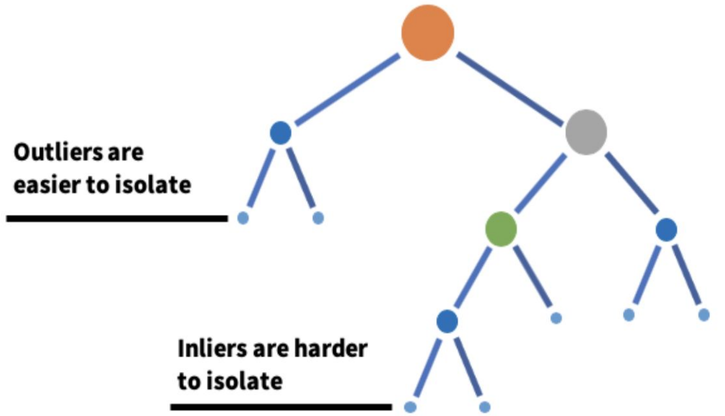
ISOLATION FOREST

Isolation Forests (IF) are an unsupervised model built on decision trees. An Isolation Forest processes randomly subsampled data in a tree structure based on randomly selected features. Because they required more cuts to isolate, samples that travel deeper into the tree are less likely to be anomalies.

Dataset: Walmart sales data(Unlabelled .csv file)
84314 rows × 16 columns

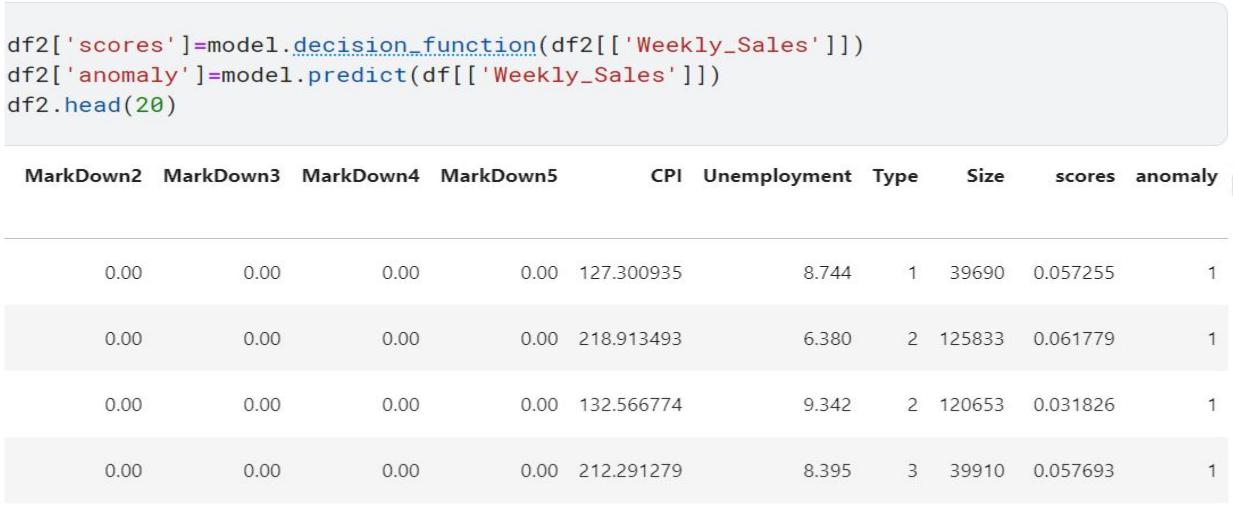
Packages used:
1. Sklearn.ensemble- for isolation forest model
2. Matplotlib.pyplot, seaborn for visualisation

Dataset: Walmart sales data(Unlabelled .csv file)
84314 rows × 16 columns



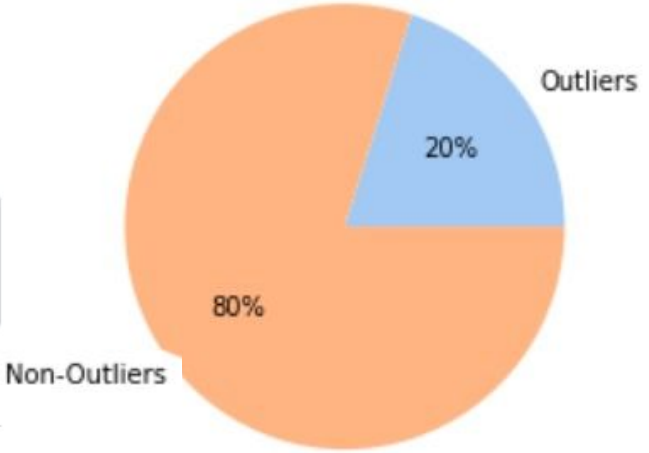
ISOLATION FOREST contd.

(IF assume small percentage of Outliers and hence we
 Visualise the outlier percentage)

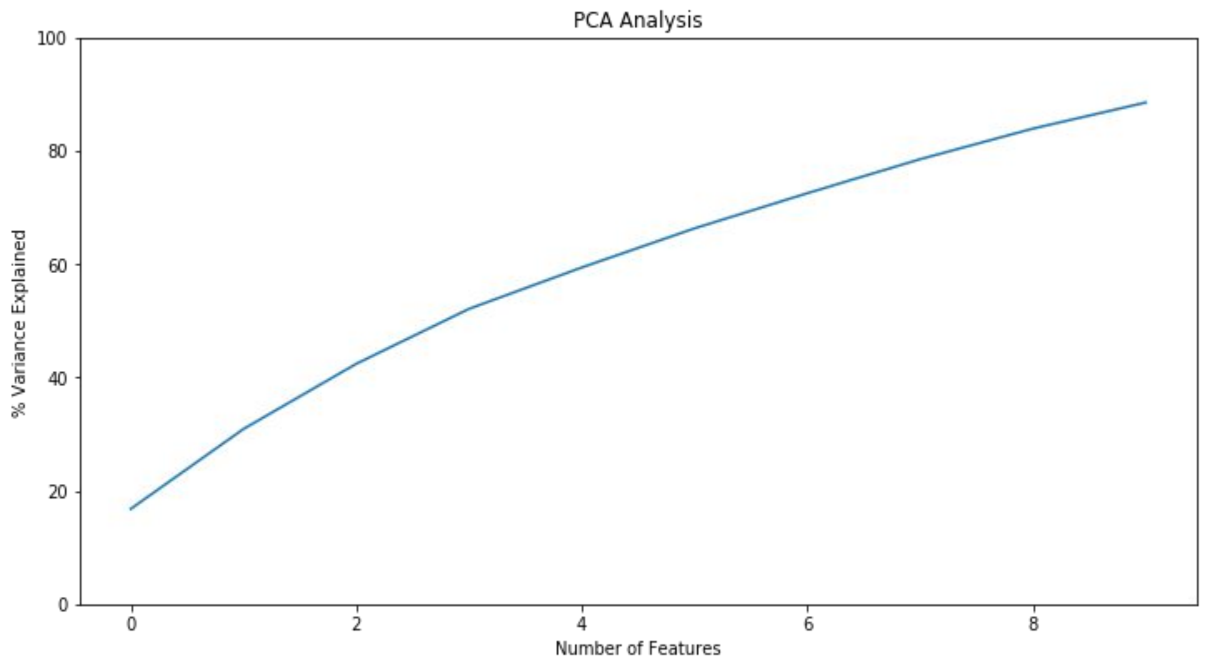


```
df2['scores']=model.decision_function(df2[['Weekly_Sales']])
df2['anomaly']=model.predict(df[['Weekly_Sales']])
df2.head(20)
```

MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	Type	Size	scores	anomaly
0.00	0.00	0.00	0.00	127.300935	8.744	1	39690	0.057255	1
0.00	0.00	0.00	0.00	218.913493	6.380	2	125833	0.061779	1
0.00	0.00	0.00	0.00	132.566774	9.342	2	120653	0.031826	1
0.00	0.00	0.00	0.00	212.291279	8.395	3	39910	0.057693	1



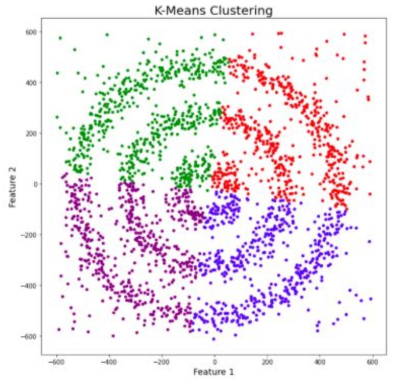
Feature Reduction - Principal Component Analysis(PCA)

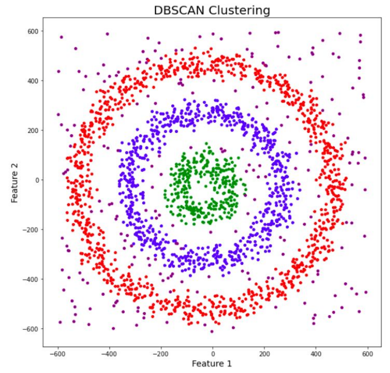


- # of principal component features chosen = **6**
- % Variance accounted for 6 PC Components = **75%**

DBSCAN

- **Density-Based Spatial Clustering of Applications with Noise**
- Unsupervised clustering algorithm that **identifies** distinctive groups/clusters in the data.
- A cluster in data space is a contiguous region of high point density, separated from other such clusters by contiguous regions of low point density.





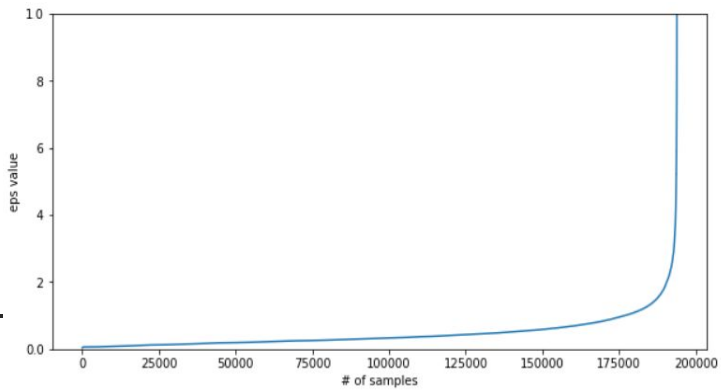
Data Considered

- The total data constitutes of 3331 time series data with a maximum of 143 length.
- We have considered 1500 time series data for the analysis of DBSCAN.
- This was due to the computational complexity of the algorithm.

Model Parameters

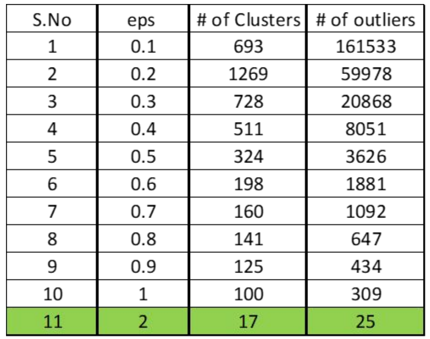
1. Epsilon

- **Radius of a cluster around a point x.**
- Optimal value for ε at the point of maximum curvature i.e. **eps = 2**.

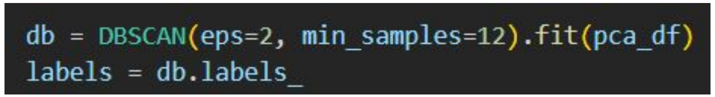


2. Min_points

- **Fewest number of points required to form a cluster.**
- General convention is to keep *minPoints >= Dimensions+1*.
- For Dimensions > 2, minPoints is set to 2 * dimensions (Sander et al., 1998)
- In our case, we have considered **min_points = 2 * 6 = 12.**

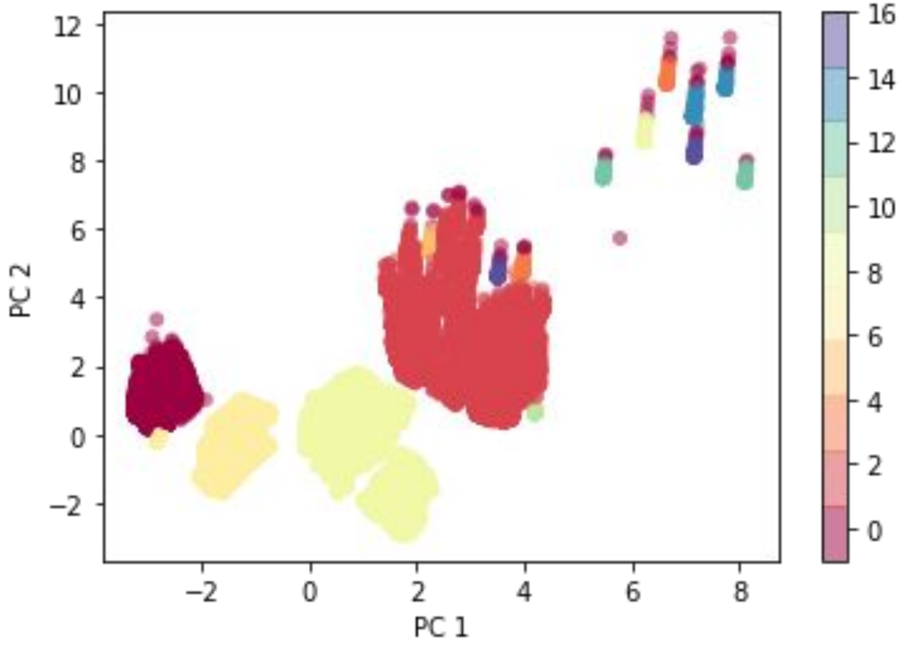


S.No	eps	# of Clusters	# of outliers
1	0.1	693	161533
2	0.2	1269	59978
3	0.3	728	20868
4	0.4	511	8051
5	0.5	324	3626
6	0.6	198	1881
7	0.7	160	1092
8	0.8	141	647
9	0.9	125	434
10	1	100	309
11	2	17	25



```
db = DBSCAN(eps=2, min_samples=12).fit(pca_df)
labels = db.labels_
```

Visualization of DBSCAN clusters and Outliers



Optimal Parameters chosen:

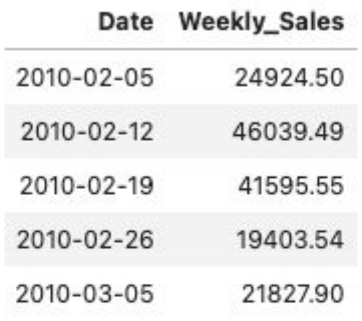
- Eps = 2
- Min_samples = 12

Results:

- # of Clusters = 17
  - Label 0 = Outliers
  - Label 1-16 = Non-outliers
- # of noise points(outliers) = 25

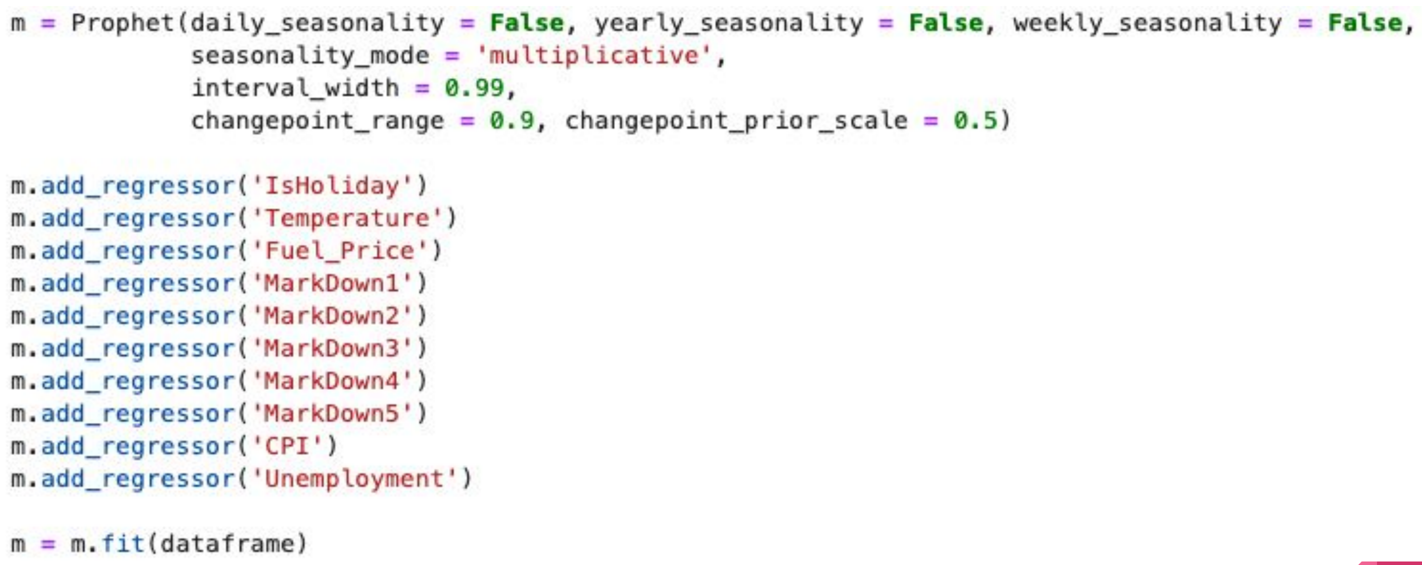
Prophet

- Open-source library for univariate (one variable) time series forecasting developed by Facebook.
- Revisiting Data:
  - Data is stored as Store X Dept and at a week level.
  - So, we can say that 1 Store X Dept constitutes of 1 time series data ranging from 2010-02-05 to 2012-11-01(143 weeks). Therefore, the total data constitutes of 3331 time series data with a maximum of 143 length.
  - One time series data sample:



Date	Weekly_Sales
2010-02-05	24924.50
2010-02-12	46039.49
2010-02-19	41595.55
2010-02-26	19403.54
2010-03-05	21827.90

Prophet - Modeling

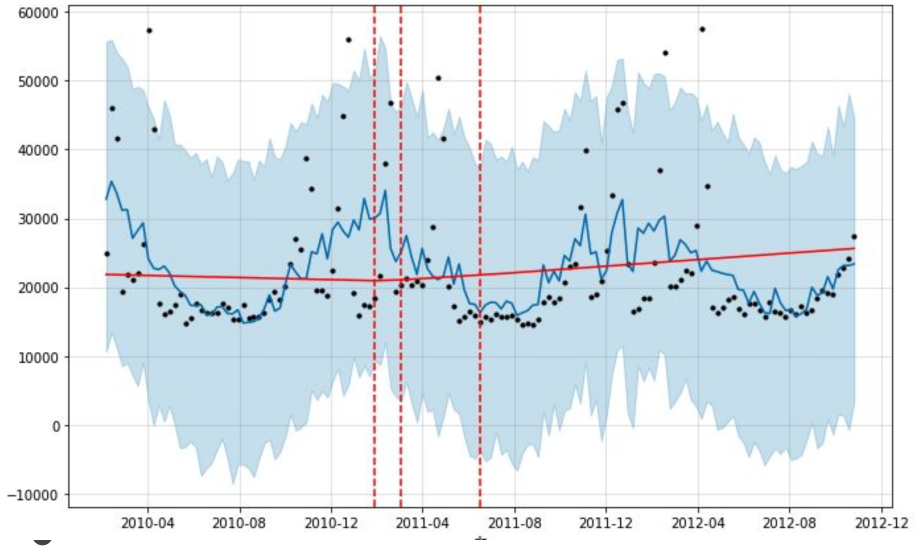


```
m = Prophet(daily_seasonality = False, yearly_seasonality = False, weekly_seasonality = False,
            seasonality_mode = 'multiplicative',
            interval_width = 0.99,
            changepoint_range = 0.9, changepoint_prior_scale = 0.5)

m.add_regressor('IsHoliday')
m.add_regressor('Temperature')
m.add_regressor('Fuel_Price')
m.add_regressor('MarkDown1')
m.add_regressor('MarkDown2')
m.add_regressor('MarkDown3')
m.add_regressor('MarkDown4')
m.add_regressor('MarkDown5')
m.add_regressor('CPI')
m.add_regressor('Unemployment')

m = m.fit(dataframe)
```

Prophet - defining model parameters

- Seeing the difference between the actual values and the forecasted values over time, we used **‘multiplicative’** seasonality.
- **Interval_width** is used to set the uncertainty intervals.
- Changepoints are abrupt changes in time series trajectories. It can be configured by using the **changepoint_range** argument. (0.9 means changepoints will be placed in the first 90% of the time series).
- To avoid underfitting and overfitting, we adjust the argument **changepoint_prior_scale**. Increasing it will make the trend more flexible.
- To capture external factors, model uses **add_regressors**.

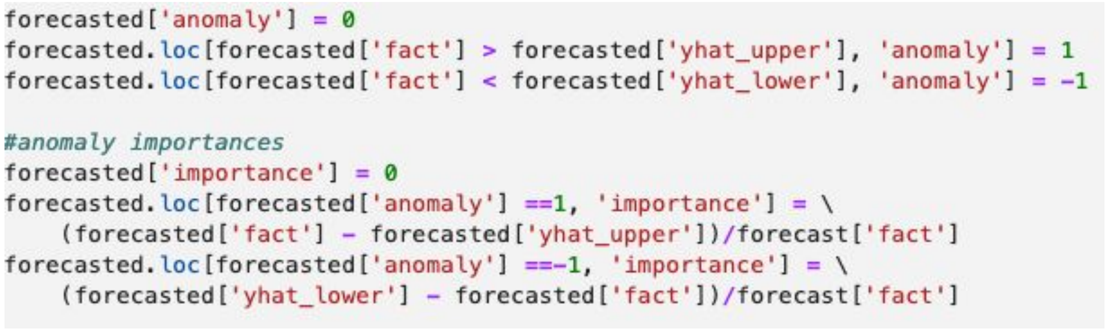
Prophet - Sample Modeling results



- Black dots are the actual values, solid blue curve is the predicted curve.
- Dotted red lines are the changepoints.
- The blue background behind the curve is the uncertainty level. Output of model after training provides the lower and the upper boundaries.
- We may find some black dots outside the interval, those are assumed as outliers here.

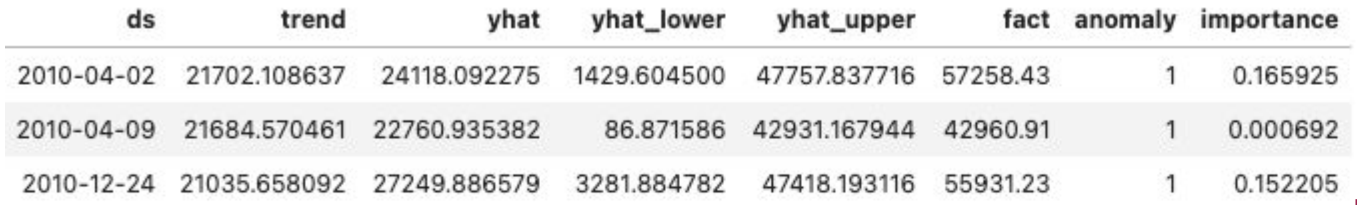
Prophet - Sample Modeling results

- Anomaly importance



```
forecasted['anomaly'] = 0
forecasted.loc[forecasted['fact'] > forecasted['yhat_upper'], 'anomaly'] = 1
forecasted.loc[forecasted['fact'] < forecasted['yhat_lower'], 'anomaly'] = -1

#anomaly importances
forecasted['importance'] = 0
forecasted.loc[forecasted['anomaly'] ==1, 'importance'] = \
    (forecasted['fact'] - forecast['yhat_upper'])/forecast['fact']
forecasted.loc[forecasted['anomaly'] ==-1, 'importance'] = \
    (forecasted['yhat_lower'] - forecasted['fact'])/forecast['fact']
```

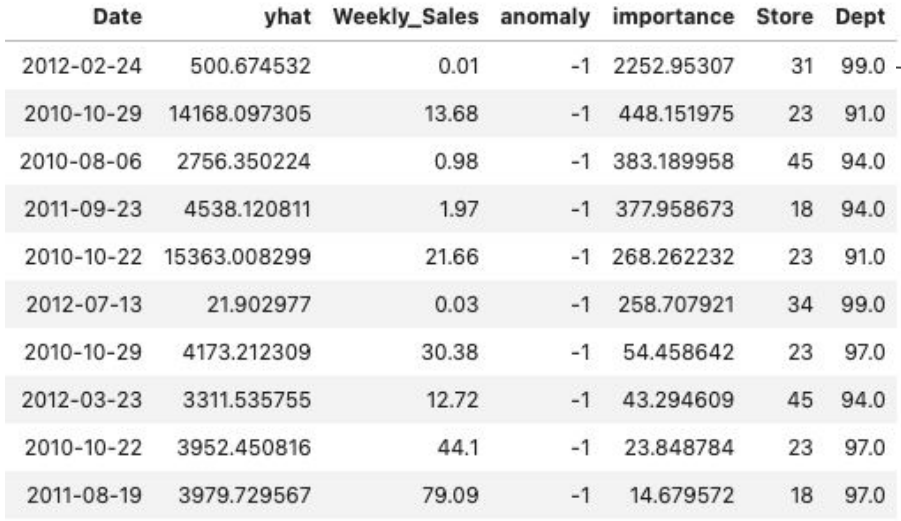
- Sample output



ds	trend	yhat	yhat_lower	yhat_upper	fact	anomaly	importance
2010-04-02	21702.108637	24118.092275	1429.604500	47757.837716	57258.43	1	0.165925
2010-04-09	21684.570461	22760.935382	86.871586	42931.167944	42960.91	1	0.000692
2010-12-24	21035.658092	27249.886579	3281.884782	47418.193116	55931.23	1	0.152205

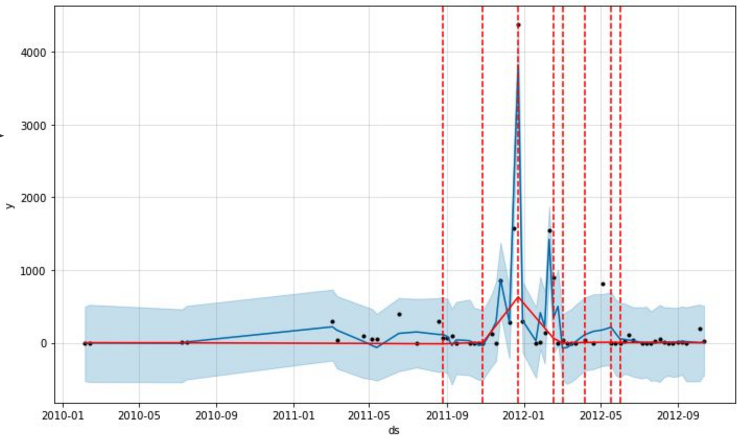
Prophet - Results

- We got 455 outliers in 421570 samples of data with the importance value greater than 0.4.



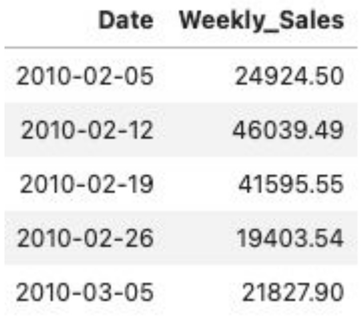
Date	yhat	Weekly_Sales	anomaly	importance	Store	Dept
2012-02-24	500.674532	0.01	-1	2252.95307	31	99.0
2010-10-29	14168.097305	13.68	-1	448.151975	23	91.0
2010-08-06	2756.350224	0.98	-1	383.189958	45	94.0
2011-09-23	4538.120811	1.97	-1	377.958673	18	94.0
2010-10-22	15363.008299	21.66	-1	268.262232	23	91.0
2012-07-13	21.902977	0.03	-1	258.707921	34	99.0
2010-10-29	4173.212309	30.38	-1	54.458642	23	97.0
2012-03-23	3311.535755	12.72	-1	43.294609	45	94.0
2010-10-22	3952.450816	44.1	-1	23.848784	23	97.0
2011-08-19	3979.729567	79.09	-1	14.679572	18	97.0

Time series for store=31 and dept=99



GluonTs

- Gluon Time Series (GluonTS), is a open source library for deep-learning-based time series modeling.
- Revisiting Data:
  - Data is stored as Store X Dept and at a week level.
  - So, we can say that 1 Store X Dept constitutes of 1 time series data ranging from 2010-02-05 to 2012-11-01(143 weeks). Therefore, the total data constitutes of 3331 time series data with a maximum of 143 length.
  - One time series data sample:



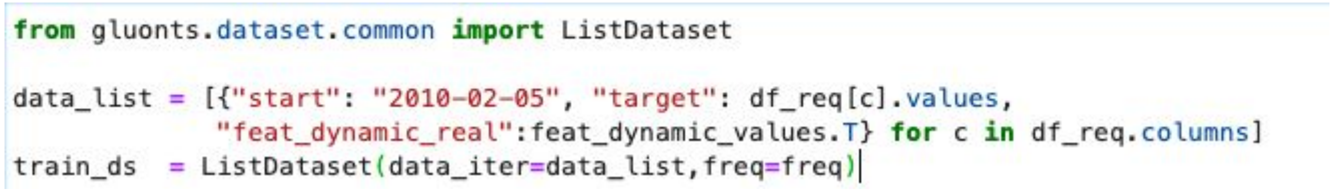
Date	Weekly_Sales
2010-02-05	24924.50
2010-02-12	46039.49
2010-02-19	41595.55
2010-02-26	19403.54
2010-03-05	21827.90

Why GluonTs - DeepAR?

- DeepAR forecasting algorithm is a supervised learning algorithm for forecasting scalar (one-dimensional) time series using recurrent neural networks (RNN).
- Classical forecasting methods, such as autoregressive integrated moving average (ARIMA) or exponential smoothing (ETS), fit a single model to each time series. They then use that model to extrapolate the time series into the future.
- In many applications, however, you have many similar time series across units. For this type of application, we can benefit from training a single model jointly over all of the time series. DeepAR takes this approach. When your dataset contains hundreds of related time series.

DeepAR - Dataset

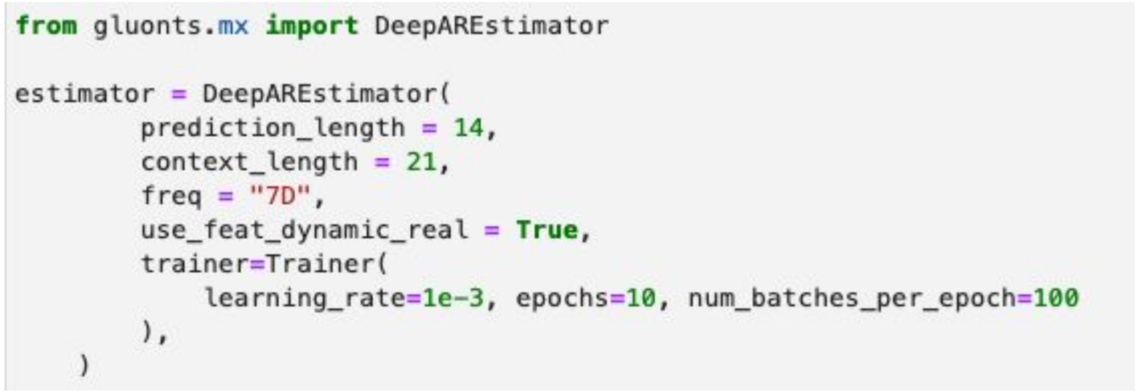
- ListDataset



```
from gluonts.dataset.common import ListDataset

data_list = [{"start": "2010-02-05", "target": df_req[c].values,
             "feat_dynamic_real":feat_dynamic_values.T} for c in df_req.columns]
train_ds  = ListDataset(data_iter=data_list,freq=freq)
```

  - “Start” : Start date of time series.
  - “Target” : Time series target values. Here, it is “weekly sales”.
  - “Feat_dynamic_real”: Regressors values in a numpy array.
  - “Freq”: The frequency of our data, here weekly.

DeepAR - Modeling

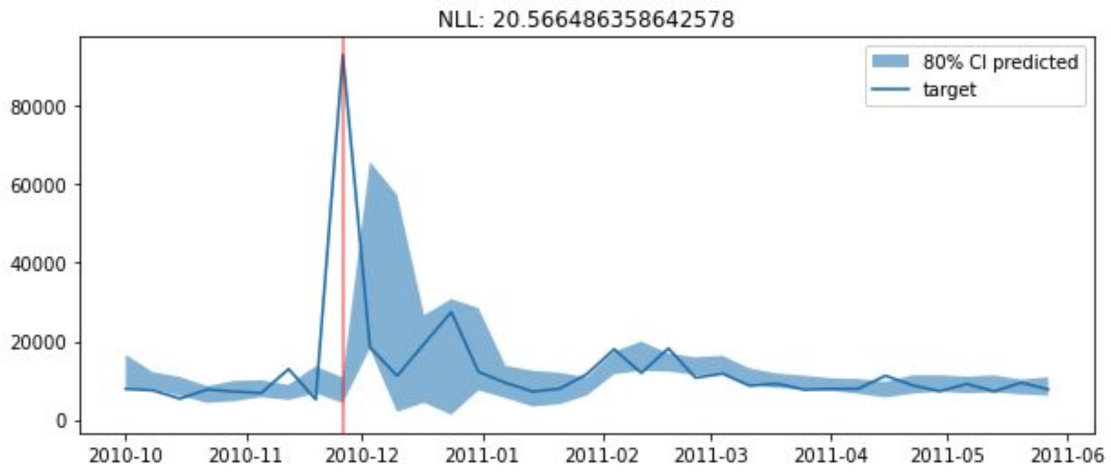


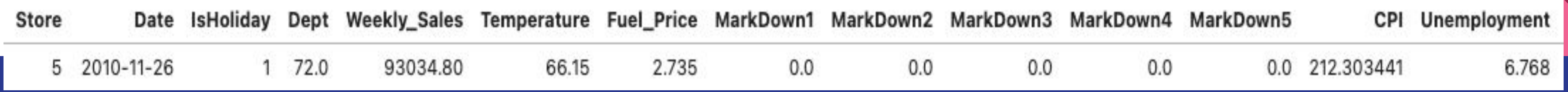
```
from gluonts.mx import DeepAREstimator

estimator = DeepAREstimator(
        prediction_length = 14,
        context_length = 21,
        freq = "7D",
        use_feat_dynamic_real = True,
        trainer=Trainer(
            learning_rate=1e-3, epochs=10, num_batches_per_epoch=100
        ),
    )
```

- Freq: The frequency of our data, here weekly.
- Context_length: Training on this number of days.
- Prediction_length: Predicting these many days.
- use_feat_dynamic_real : Adding regressors.

DeepAR - Sample Modeling Results

- NLL: negative log likelihood loss.
- Solid blue line: original target values.
- Blue background behind the curve is the uncertainty level.
- Red line is the outlier.

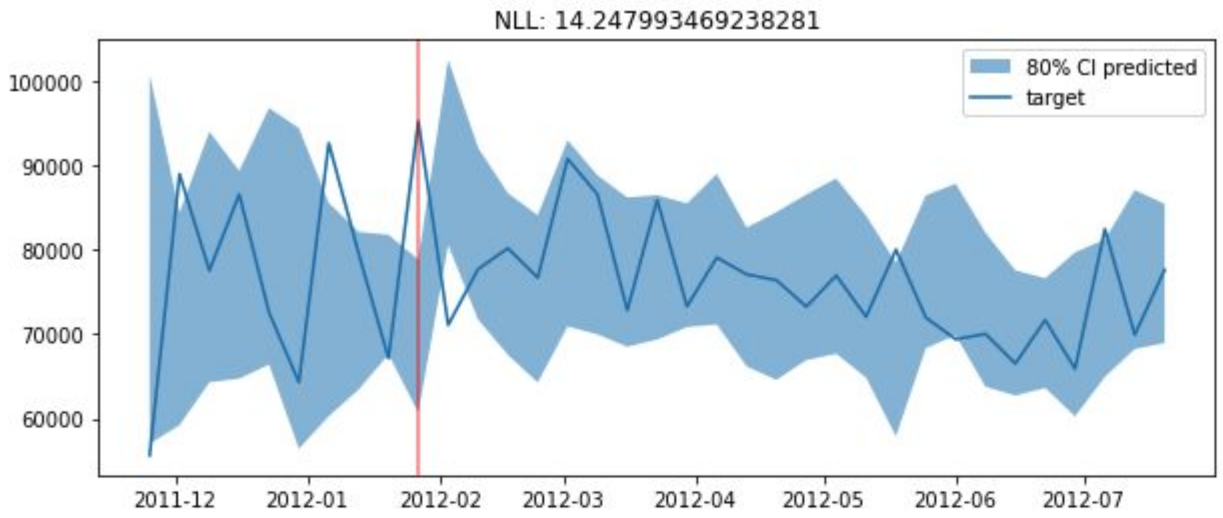




Store	Date	IsHoliday	Dept	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment
5	2010-11-26	1	72.0	93034.80	66.15	2.735	0.0	0.0	0.0	0.0	0.0	212.303441	6.768

DeepAR - Results

- We got 441 outliers in 421570 samples of data with the NLL loss ranging from 14.24 to 20.56.



Store	Date	IsHoliday	Dept	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment
6	2012-01-27	0	38.0	95179.92	53.95	3.290	1510.59	630.92	37.1	17.0	2641.9	221.67692	6.132

DeepAR vs Prophet

- DeepAR - 1 model jointly over all of the time series.(1 model for 3331 time series here).
- Prophet - 1 model for each time series.(3331 model for 3331 time series here).
- DeepAR training time: 38.7 seconds.
- Prophet training time: 76.87 minutes.(≈120 times of DeepAR training).
- Both the models shared 6 common outliers.

Conclusions and future work

- Most of the times, we don’t have the outlier labels. So, Anomaly detection using unsupervised model is more convenient and popular.
- In the context of the unsupervised Walmart data used, It makes more sense to use the time series models as they have some seasonality and trend with respect to the past values.
- To save computation and memory, it is ideal to use the DeepAR. As it takes less time to train and saves disk space.
- It is complex to handle to the DeepAR hyperparameters, as it generalizes across all time series. So, In future, we will look to analyze the hyperparameters complexity.

Thank You!