Capstone Project 1: Report

Problem:
Planners and Product Managers of Manufacturing companies are faced with the difficulty year in year out to plan and budget for raw materials to be used to build finished goods for the following year. There is a tendency to estimate or plan according to forecasts provided by the sales teams or based on the volume done in the previous year. This leads to situations where demand cannot be met in a timely manner due to understocking or where demand drops off and there is overstock on raw materials for a SKU that ends up not being demanded as forecasted. The latter leads to scrapping of finished goods and hence increased operating costs. The objective here is to analyze solely historical sales data with data science techniques, to predict with a greater amount of precision the number of SKUs that would be required in the following year. This provides the manufacturing teams with the SKU volume information necessary for more accurate raw materials purchases and proper production lines scheduling.

Client:

Manufacturing industries that want to be leaner in their budgeting and planning for production, to minimize losses due to under or over estimating specific products

Data Set:

Four years (2014 - 2018) of global sales data by month, by quarter and by region of all product SKUs sold into the renewable energy industry particularly PV solar.

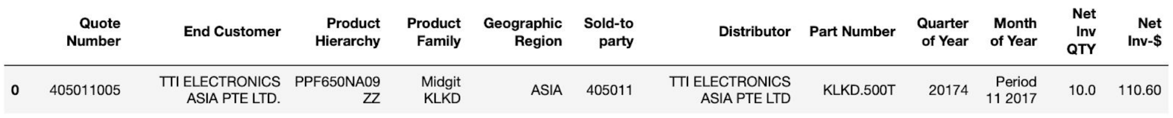
Approach:

Export data from the SAP business operations software in an excel format. Wrangle the data in pandas to eliminate any bad and irrelevant data or outliers. Do an EDA on the data to determine if there are any trends by year, month, by region, and by SKU. Use 4 years of data to predict the monthly SKU requirements for the 5th year.

Data Wrangling

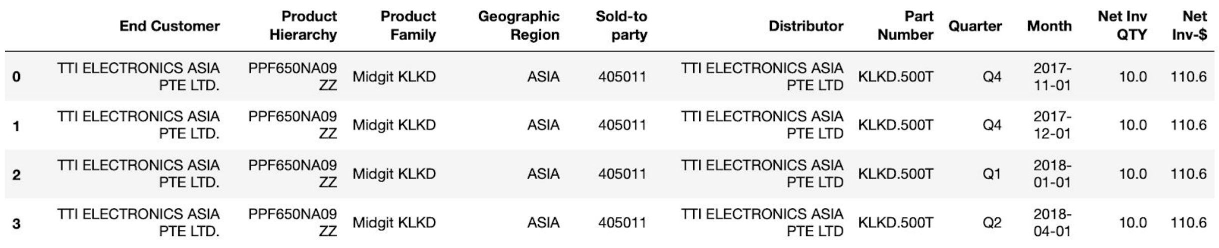
Cleaning Steps Performed:

- Tidy Data Format:
  - Before pulling the data into a .csv file, features of the business operating software we used to arrange the data in a tidy format where columns represent separate variables and each row represents individual observations.



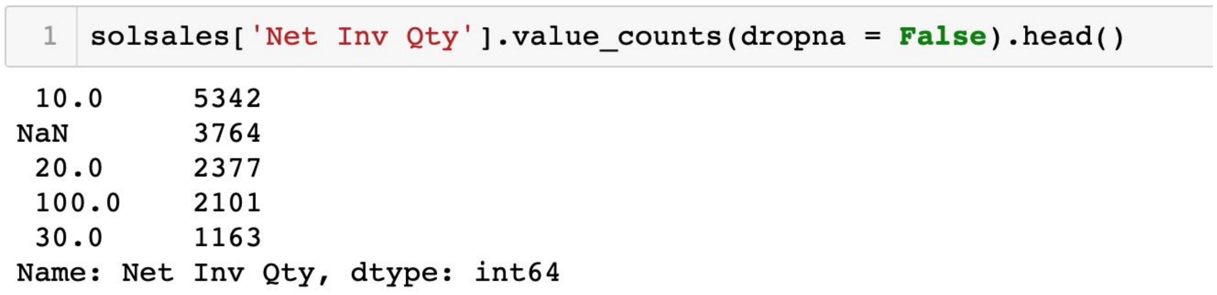
	Quote Number	End Customer	Product Hierarchy	Product Family	Geographic Region	Sold-to party	Distributor	Part Number	Quarter of Year	Month of Year	Net Inv QTY	Net Inv-$
0	405011005	TTI ELECTRONICS ASIA PTE LTD.	PPF650NA09 ZZ	Midgit KLKD	ASIA	405011	TTI ELECTRONICS ASIA PTE LTD	KLKD.500T	20174	Period 11 2017	10.0	110.60

- Inconsistent Column Names:
  - Converted “Month of Year” rows into date time format and renamed it “Month” and “Quarter of Year” into “Quarter” with alphanumeric representations (i.e. Q1, Q2, Q3, Q4).



	End Customer	Product Hierarchy	Product Family	Geographic Region	Sold-to party	Distributor	Part Number	Quarter	Month	Net Inv QTY	Net Inv-$
0	TTI ELECTRONICS ASIA PTE LTD.	PPF650NA09 ZZ	Midgit KLKD	ASIA	405011	TTI ELECTRONICS ASIA PTE LTD	KLKD.500T	Q4	2017-11-01	10.0	110.6
1	TTI ELECTRONICS ASIA PTE LTD.	PPF650NA09 ZZ	Midgit KLKD	ASIA	405011	TTI ELECTRONICS ASIA PTE LTD	KLKD.500T	Q4	2017-12-01	10.0	110.6
2	TTI ELECTRONICS ASIA PTE LTD.	PPF650NA09 ZZ	Midgit KLKD	ASIA	405011	TTI ELECTRONICS ASIA PTE LTD	KLKD.500T	Q1	2018-01-01	10.0	110.6
3	TTI ELECTRONICS ASIA PTE LTD.	PPF650NA09 ZZ	Midgit KLKD	ASIA	405011	TTI ELECTRONICS ASIA PTE LTD	KLKD.500T	Q2	2018-04-01	10.0	110.6

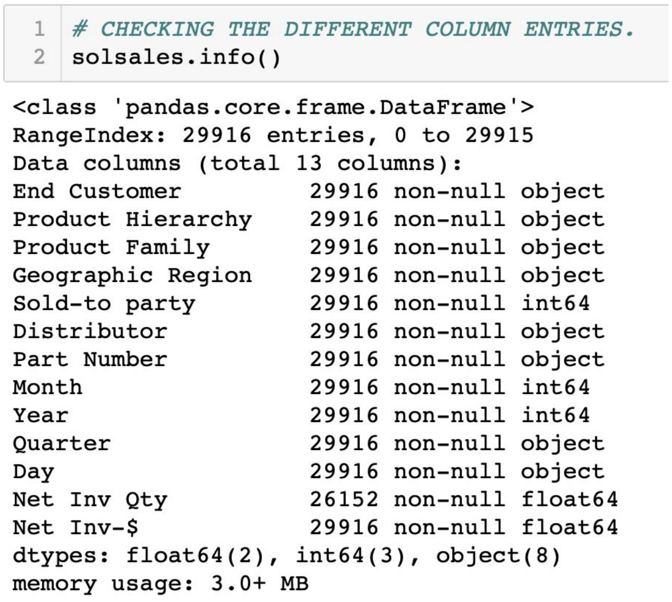
- Missing data:
  - Ran the .info() attribute and realized the “Net Inv Qty column had some missing data.
  - There are 3764 missing values on the Qty column



```
solsales['Net Inv Qty'].value_counts(dropna = False).head()
```

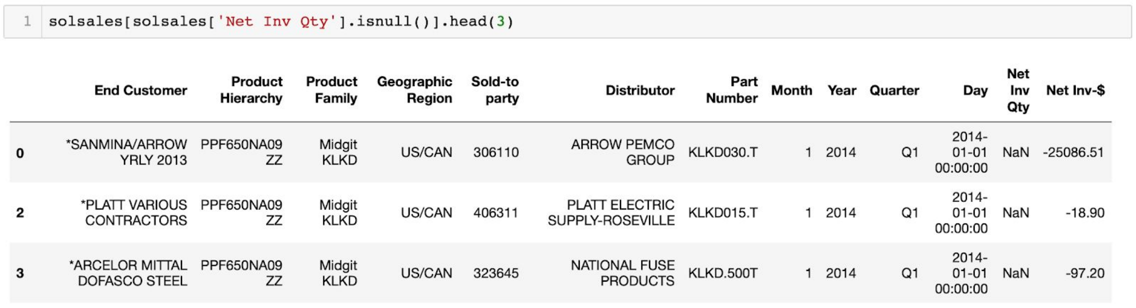
```
 10.0      5342
NaN        3764
 20.0      2377
 100.0     2101
 30.0      1163
Name: Net Inv Qty, dtype: int64
```

  - Looking at all the rows with the NaN values, you quickly realize the associated Net-Inv $ are all negative. This has to do with what is known as “Ship and Debits” from distributors which are not relevant to our analysis



```python
# CHECKING THE DIFFERENT COLUMN ENTRIES.
solsales.info()
```

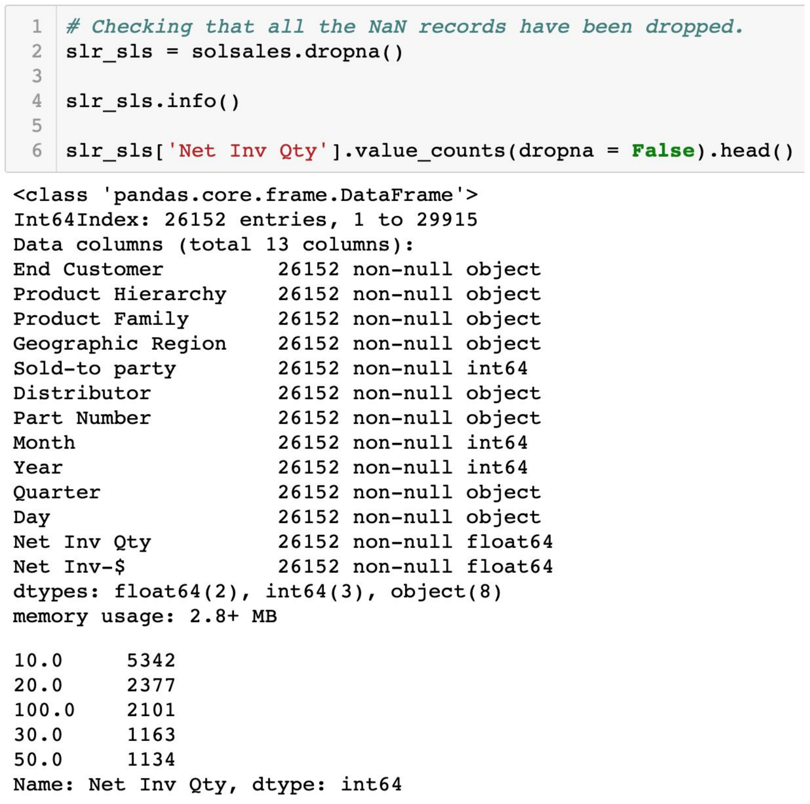
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29916 entries, 0 to 29915
Data columns (total 13 columns):
End Customer         29916 non-null object
Product Hierarchy    29916 non-null object
Product Family       29916 non-null object
Geographic Region    29916 non-null object
Sold-to party        29916 non-null int64
Distributor          29916 non-null object
Part Number          29916 non-null object
Month                29916 non-null int64
Year                 29916 non-null int64
Quarter              29916 non-null object
Day                  29916 non-null object
Net Inv Qty          26152 non-null float64
Net Inv-$            29916 non-null float64
dtypes: float64(2), int64(3), object(8)
memory usage: 3.0+ MB
```



```python
solsales[solsales['Net Inv Qty'].isnull()].head(3)
```

	End Customer	Product Hierarchy	Product Family	Geographic Region	Sold-to party	Distributor	Part Number	Month	Year	Quarter	Day	Net Inv Qty	Net Inv-$
0	*SANMINA/ARROW YRLY 2013	PPF650NA09 ZZ	Midgit KLKD	US/CAN	306110	ARROW PEMCO GROUP	KLKD030.T	1	2014	Q1	2014-01-01 00:00:00	NaN	-25086.51
2	*PLATT VARIOUS CONTRACTORS	PPF650NA09 ZZ	Midgit KLKD	US/CAN	406311	PLATT ELECTRIC SUPPLY-ROSEVILLE	KLKD015.T	1	2014	Q1	2014-01-01 00:00:00	NaN	-18.90
3	*ARCELOR MITTAL DOFASCO STEEL	PPF650NA09 ZZ	Midgit KLKD	US/CAN	323645	NATIONAL FUSE PRODUCTS	KLKD.500T	1	2014	Q1	2014-01-01 00:00:00	NaN	-97.20

-
- We apply the dropna() method to drop all missing value rows.



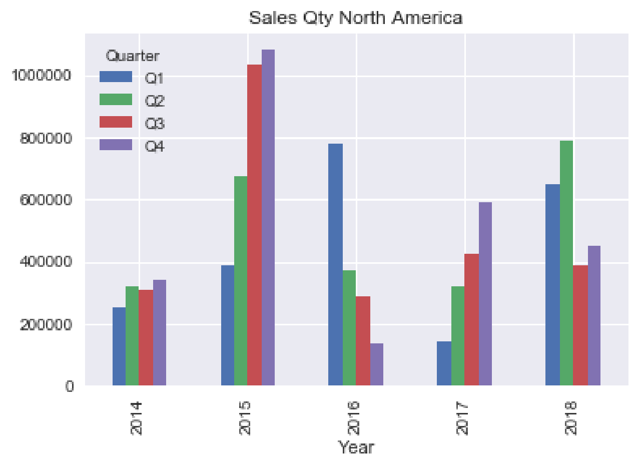
```python
# Checking that all the NaN records have been dropped.
slr_sls = solsales.dropna()

slr_sls.info()

slr_sls['Net Inv Qty'].value_counts(dropna = False).head()
```

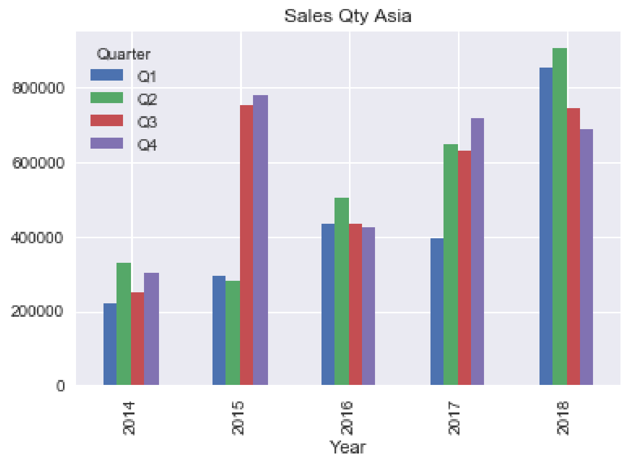
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 26152 entries, 1 to 29915
Data columns (total 13 columns):
End Customer          26152 non-null object
Product Hierarchy     26152 non-null object
Product Family        26152 non-null object
Geographic Region     26152 non-null object
Sold-to party         26152 non-null int64
Distributor           26152 non-null object
Part Number           26152 non-null object
Month                 26152 non-null int64
Year                  26152 non-null int64
Quarter               26152 non-null object
Day                   26152 non-null object
Net Inv Qty           26152 non-null float64
Net Inv-$             26152 non-null float64
dtypes: float64(2), int64(3), object(8)
memory usage: 2.8+ MB

10.0     5342
20.0     2377
100.0    2101
30.0     1163
50.0     1134
Name: Net Inv Qty, dtype: int64
```

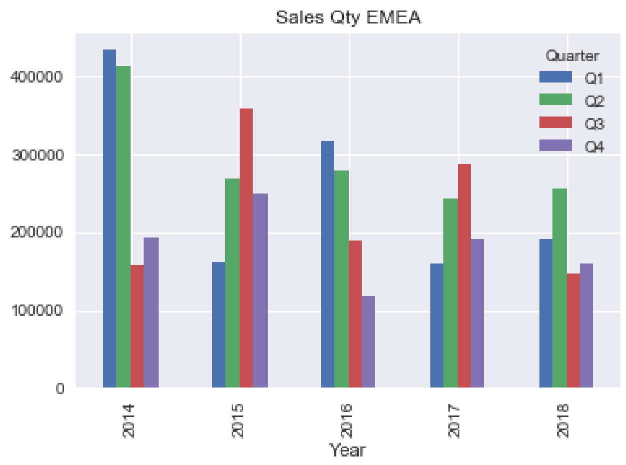
Data Story Telling:

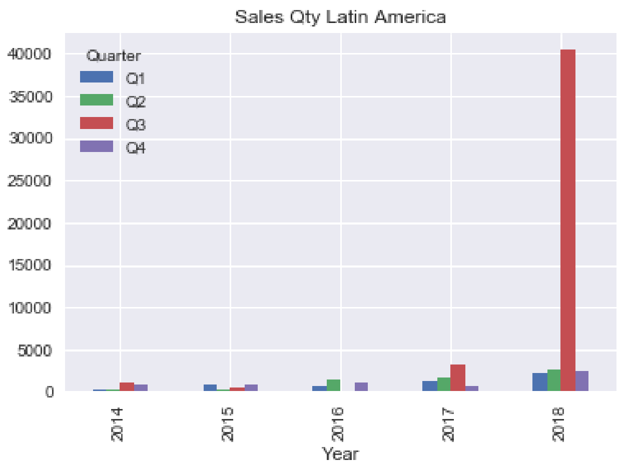
Regional Comparisons by Quarter:

- The strongest markets are the US and Asia followed by Europe and then Latin America
- 2015 was a great year for both US and Asia markets but 2016 saw a steep decline in these markets.
- Since 2014 Europe has been on a consistent decline in units sold while Latin America has been growing slowly having a remarkable quarter in Q3 of 2018

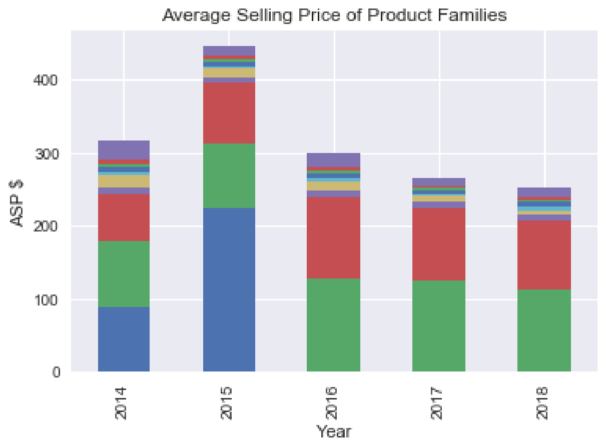




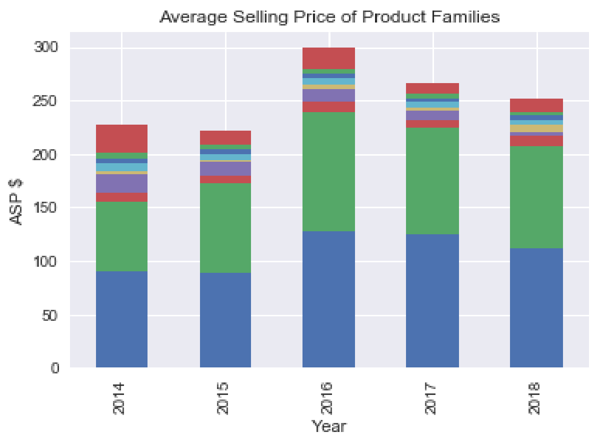




- The average selling prices overall have fluctuated but have seen a drop over the years.
- One of the product families “1000Vdc Solar” did not sell in 2016 - 2018 . It contributed to the high ASP total in 2015.



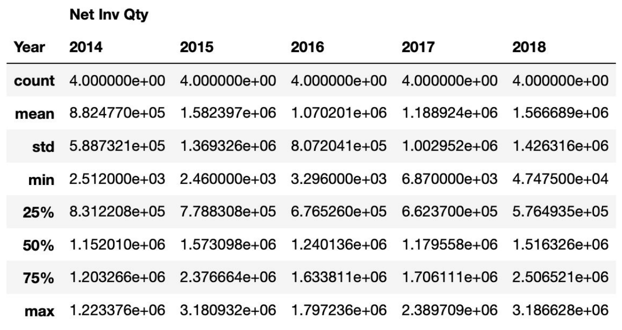
- Taking the this product family off and comparing only product families sold across the 5 years we have that 2016 was the highest year of ASPs but has seen a decline across all product families since then.



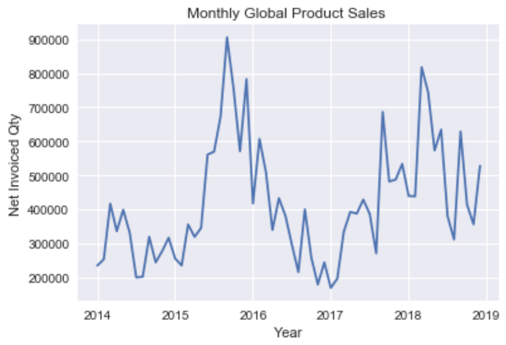
Exploratory Data Analysis

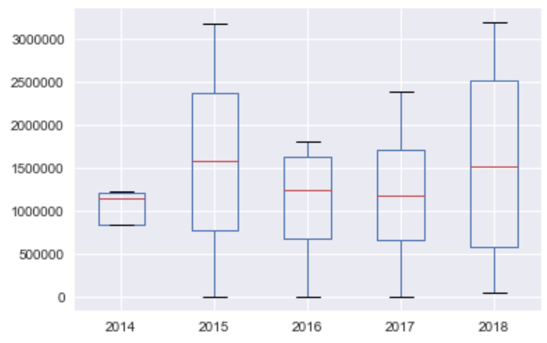
The dataset we set out to analyze is four years (2014 - 2018) of global sales data by month, by quarter and by region of all SKUs sold into the renewable energy industry. The main objective is to analyze the data to understand any trends and patterns and to be able to use the first four years to forecast the fifth year.

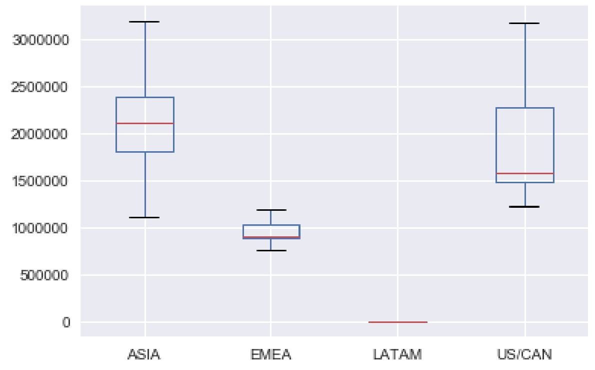
The main variables to consider in the dataset are the quantity of SKUs sold and their year over year variability as shown below. A time series provides of the data provides a visualization of the data:



	Net Inv Qty				
Year	**2014**	**2015**	**2016**	**2017**	**2018**
count	4.000000e+00	4.000000e+00	4.000000e+00	4.000000e+00	4.000000e+00
mean	8.824770e+05	1.582397e+06	1.070201e+06	1.188924e+06	1.566689e+06
std	5.887321e+05	1.369326e+06	8.072041e+05	1.002952e+06	1.426316e+06
min	2.512000e+03	2.460000e+03	3.296000e+03	6.870000e+03	4.747500e+04
25%	8.312208e+05	7.788308e+05	6.765260e+05	6.623700e+05	5.764935e+05
50%	1.152010e+06	1.573098e+06	1.240136e+06	1.179558e+06	1.516326e+06
75%	1.203266e+06	2.376664e+06	1.633811e+06	1.706111e+06	2.506521e+06
max	1.223376e+06	3.180932e+06	1.797236e+06	2.389709e+06	3.186628e+06

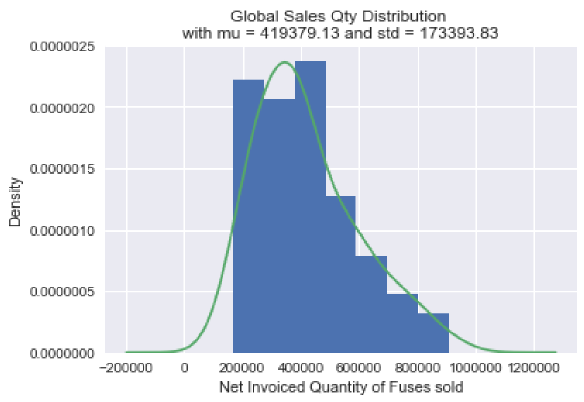


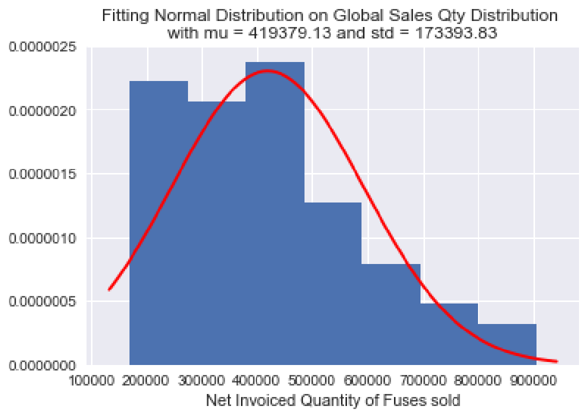


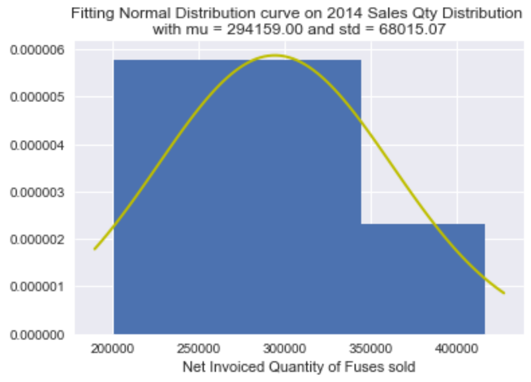


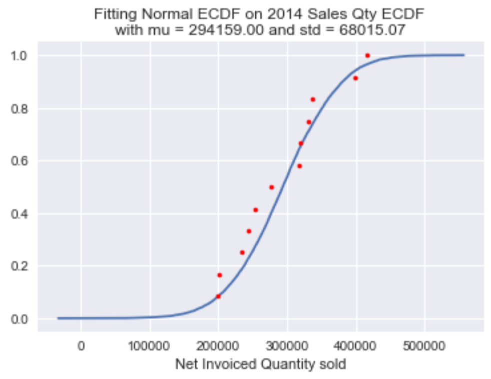
Clearly looking at the summary statistics and plots, it is clear that the best years so far have been 2015 and 2018. The most variability in terms of quantity sold across the 4 regions, is in 2018. The biggest markets have been Asia and the US.

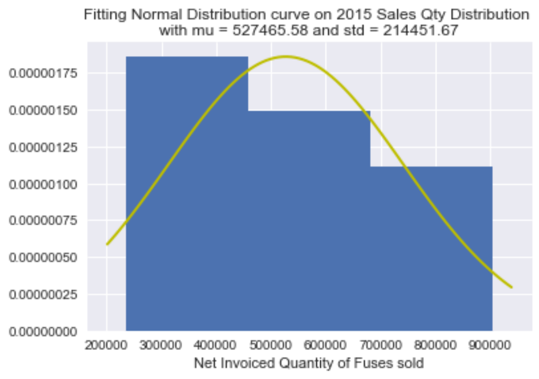
Global and Regional Quantity Sales Distribution:
Doing a normality test and plotting the distribution of the total quantity sold globally for the 5 years of data, it gives a right skewed distribution. Which indicates that there are fewer months in which the very high quantities were bought.

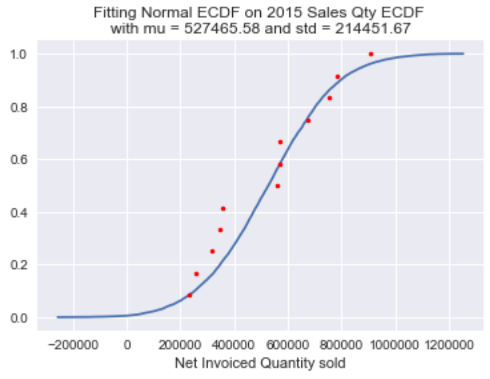


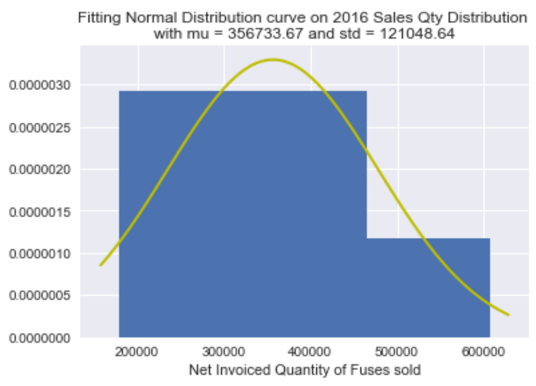


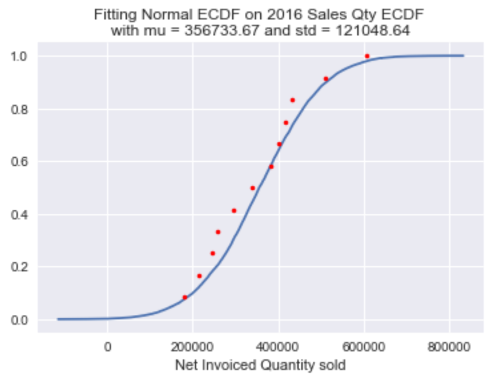


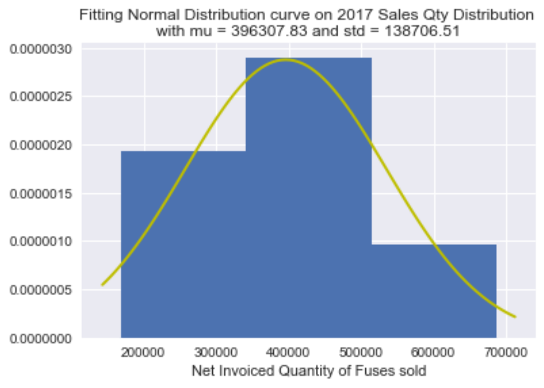


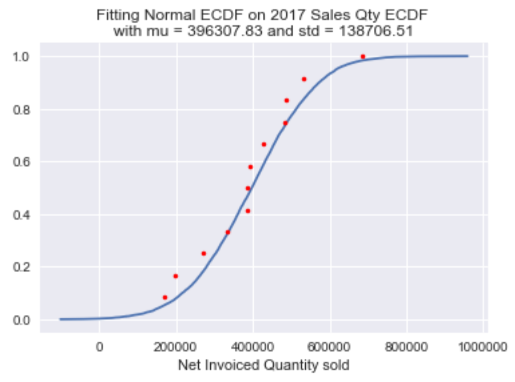


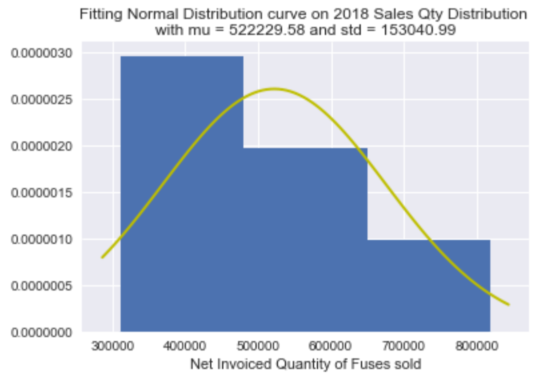


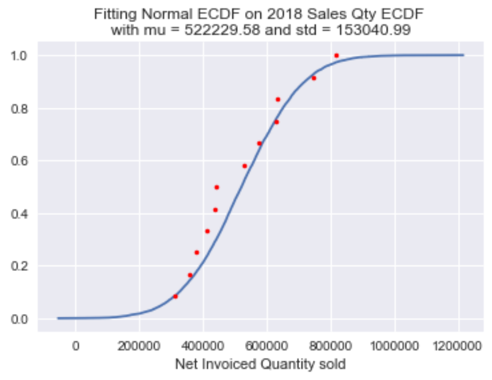












The data varies greatly from year to year and there does not seem to be a specific trend in the sales. The distribution of the sales quantity is not normal. As shown in the plots and ECDFs. Most of the distribution is right skewed which makes sense that few products would be sold at very high quantities. To better forecast the sales quantities, we will need to account for the variability and seasonality of the data. One of the ways to do this is to use an ARIMA model which makes use of Machine Learning processes to predict or forecast sales.

Machine Learning: In-depth Analysis

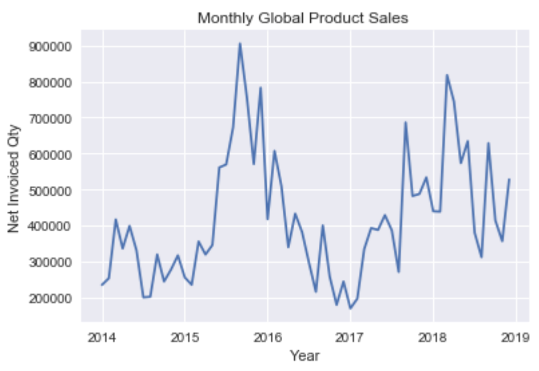
To objective is to forecast the sales quantities per month of products sold into the renewable energy market.

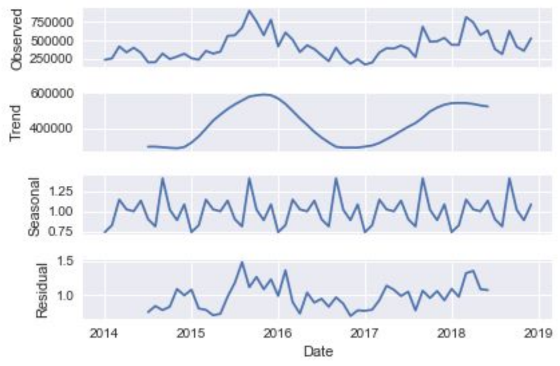
The dataset is composed of 5 years (2014 – 2018) of sales quantities.

Evaluation of Model:
The first 4 years of the data are split into a training and test data. The ARIMA model is used for the prediction of the unit sales

- The performance of the model predictions is evaluated from their RMSE (Root Mean Squared Error) calculated by taking the square root of the output of the mean_squared_error function in the scikit-learn library. This is the criteria used in choosing the optimal hyperparameters of the model.
- To training and test data is split in a ratio of 3:1. 75% of the data is used to train and 25% to test.

Data Analysis:
Analysis of any time series assumes that it is stationary (mean, variance, autocorrelation, etc. are all constant over time). Clearly as earlier shown this is not the case. The means vary from year to year. Decomposing the series using seasonal_decompose, we can see below that there is some trend and some seasonality in the series.



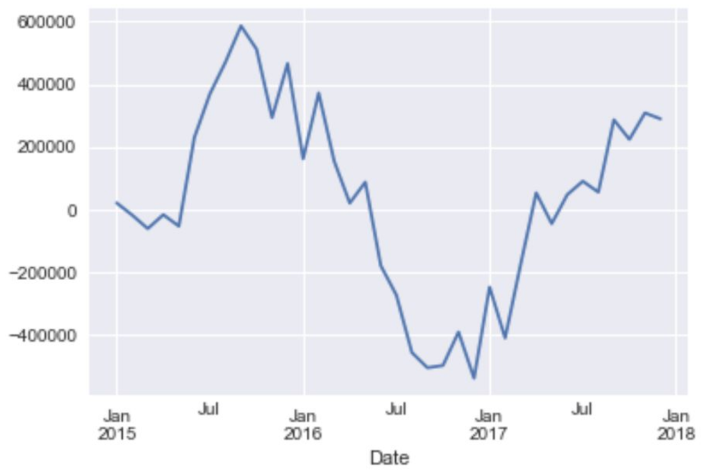


We therefore need to stationarize the series in order to analyze it free of seasonality and trends.

Stationarizing the series:

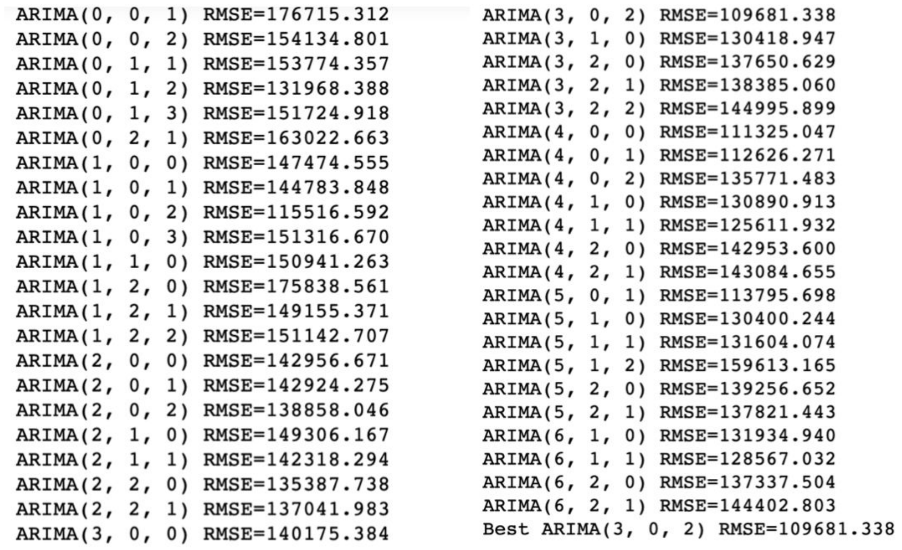
To stationarize the series we use differencing and then use a statistical test (Augmented Dickey-Fuller test) to confirm that the series is stationary. As shown below the test statistic value -3.208282 is smaller than the critical value at 5% of -2.968. This suggests that we can reject the null hypothesis the data has a unit root and is non-stationary with a significance level of less than 5%. A stationary series has no trend, its variations around its mean have a constant amplitude, and it wiggles in a consistent fashion as the plot shows.

```
ADF Statistic: -3.208282
p-value: 0.019510
Critical Values:
        1%: -3.679
        5%: -2.968
        10%: -2.623
```



Grid Search ARIMA Model Parameters (p,d,q):

The number of Autoregressive (AR) terms parameter p (0-6), difference terms parameter d (0-2) and Moving Average (MA) terms parameter q (0-6) have 147 combination possibilities.
It is good to indicate here that we cannot use conventional cross validation approach in time series prediction as we are using historical data to predict future outcomes. We therefore have to split training and test sets using time splitting such that the former is earlier than the latter.
To choose the optimal (p,d,q) hyperparameters, we define a grid search function that iterates over all these combinations and chooses the best based on its RMSE being the smallest as shown below.

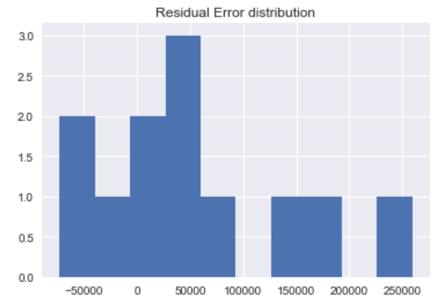


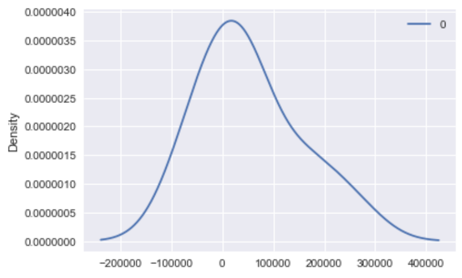
```
ARIMA(0, 0, 1) RMSE=176715.312
ARIMA(0, 0, 2) RMSE=154134.801
ARIMA(0, 1, 1) RMSE=153774.357
ARIMA(0, 1, 2) RMSE=131968.388
ARIMA(0, 1, 3) RMSE=151724.918
ARIMA(0, 2, 1) RMSE=163022.663
ARIMA(1, 0, 0) RMSE=147474.555
ARIMA(1, 0, 1) RMSE=144783.848
ARIMA(1, 0, 2) RMSE=115516.592
ARIMA(1, 0, 3) RMSE=151316.670
ARIMA(1, 1, 0) RMSE=150941.263
ARIMA(1, 2, 0) RMSE=175838.561
ARIMA(1, 2, 1) RMSE=149155.371
ARIMA(1, 2, 2) RMSE=151142.707
ARIMA(2, 0, 0) RMSE=142956.671
ARIMA(2, 0, 1) RMSE=142924.275
ARIMA(2, 0, 2) RMSE=138858.046
ARIMA(2, 1, 0) RMSE=149306.167
ARIMA(2, 1, 1) RMSE=142318.294
ARIMA(2, 2, 0) RMSE=135387.738
ARIMA(2, 2, 1) RMSE=137041.983
ARIMA(3, 0, 0) RMSE=140175.384
ARIMA(3, 0, 2) RMSE=109681.338
ARIMA(3, 1, 0) RMSE=130418.947
ARIMA(3, 2, 0) RMSE=137650.629
ARIMA(3, 2, 1) RMSE=138385.060
ARIMA(3, 2, 2) RMSE=144995.899
ARIMA(4, 0, 0) RMSE=111325.047
ARIMA(4, 0, 1) RMSE=112626.271
ARIMA(4, 0, 2) RMSE=135771.483
ARIMA(4, 1, 0) RMSE=130890.913
ARIMA(4, 1, 1) RMSE=125611.932
ARIMA(4, 2, 0) RMSE=142953.600
ARIMA(4, 2, 1) RMSE=143084.655
ARIMA(5, 0, 1) RMSE=113795.698
ARIMA(5, 1, 0) RMSE=130400.244
ARIMA(5, 1, 1) RMSE=131604.074
ARIMA(5, 1, 2) RMSE=159613.165
ARIMA(5, 2, 0) RMSE=139256.652
ARIMA(5, 2, 1) RMSE=137821.443
ARIMA(6, 1, 0) RMSE=131934.940
ARIMA(6, 1, 1) RMSE=128567.032
ARIMA(6, 2, 0) RMSE=137337.504
ARIMA(6, 2, 1) RMSE=144402.803
Best ARIMA(3, 0, 2) RMSE=109681.338
```

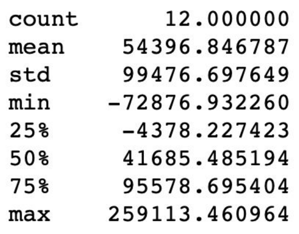
The best (p,d,q) is (3,0,2) chosen by virtue of its RMSE : 109681.3 . This order is used in the ARIMA Models to train the model.

Review of Residual Errors:

Ideally the distribution of residual errors of a model should be normal around a mean of zero to indicate a good fit for the model.
The residual error distribution of the data set is approximately normal but not centered around zero as shown by the plot and summary statistics

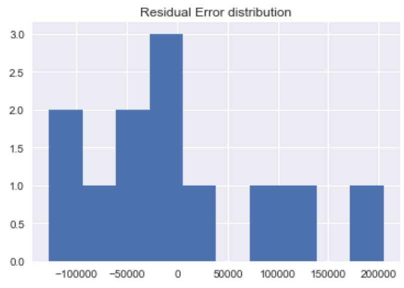


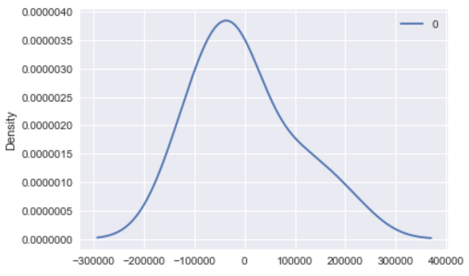


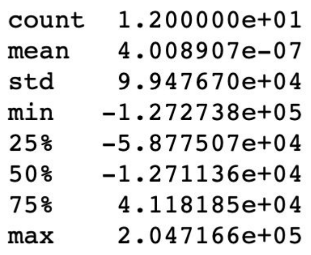


```
count        12.000000
mean      54396.846787
std       99476.697649
min      -72876.932260
25%       -4378.227423
50%       41685.485194
75%       95578.695404
max      259113.460964
```

To bias-correct the prediction, the mean of the residual error 54396.85 is added to each prediction. Making this adjustment gives an improved distribution centered very close to zero with an improved RMSE: 95241.687 from RMSE : 109681.3 as shown below:



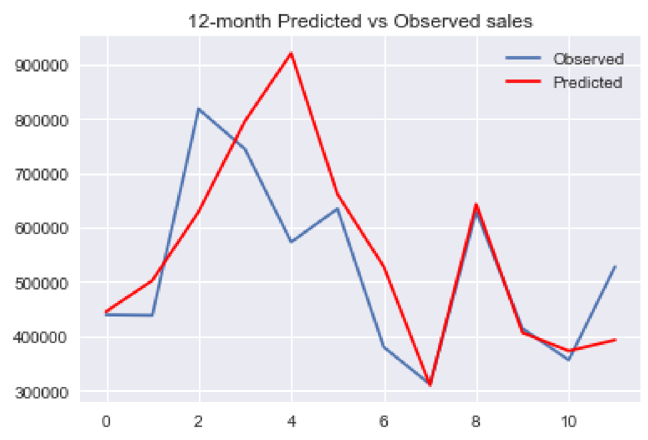




```
count   1.200000e+01
mean    4.008907e-07
std     9.947670e+04
min    -1.272738e+05
25%    -5.877507e+04
50%    -1.271136e+04
75%     4.118185e+04
max     2.047166e+05
```

Model Validation:

The trained model with optimal hyperparameters and a bias-correction is use in a rolling-forecast manner updating the model for each time step. The predictions are then compared to the fifth year of sales which was initially held-out as a validation set. The predictions and forecast errors are as shown below.



	Expected Values	Predicted Values	Forecast Error
2018-01-01	439508.0	445130.90	5622.90
2018-02-01	438470.0	502603.05	64133.05
2018-03-01	818048.0	629047.66	-189000.34
2018-04-01	744324.0	795527.38	51203.38
2018-05-01	573164.0	920468.51	347304.51
2018-06-01	634356.0	661309.71	26953.71
2018-07-01	379933.0	528069.18	148136.18
2018-08-01	311773.0	310283.52	-1489.48
2018-09-01	628897.0	642599.62	13702.62
2018-10-01	414029.0	406589.58	-7439.42
2018-11-01	356277.0	372906.52	16629.52
2018-12-01	527976.0	393080.82	-134895.18

Performance Measures:

The performance measures are as shown below:

	Performance Measures	Value
0	Forecast Bias	28405.118
1	MAE	83875.856
2	MAPE	15.366
3	RMSE	130542.906

The RMSE is 130542.906 which is higher than on the trained model. The forecast bias is positive indicating the model's tendency to over forecast. Some months like August, September and October are almost exact but other months like May, July and March are very off. Overall the model does a pretty good job to forecast the total number of units sold.

Recommendations:

1. The model should be used with the guidance that the predictions would have a tendency to over forecasted.
2. Analysis was done on the total number of units of all product types sold into the renewable industry. Analysis should be done on each product family independently for better outcome as other components potentially add noise to the analysis.

3. Additional features that influence the market like amount of installed capacity and policy change indices could be introduced into the dataset to fine tune the predictions with machine learning techniques like Random Forest and SVM.