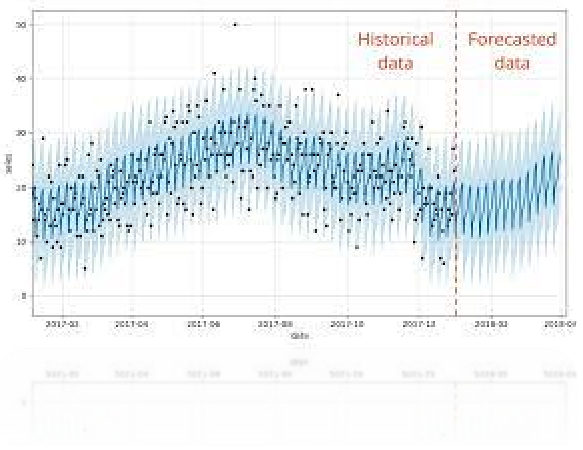


PHASE 2

ABSTRACT

**Phase 2: EDA and Feature Extraction**

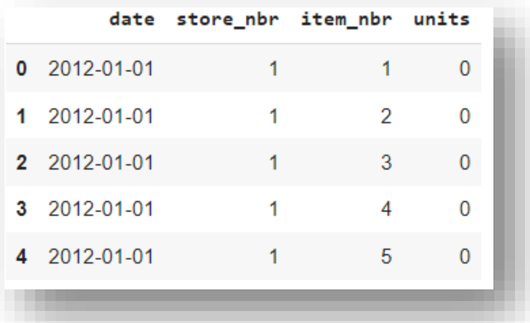
Soumen Chatterjee

Phase 2 – 25-Dec-2021

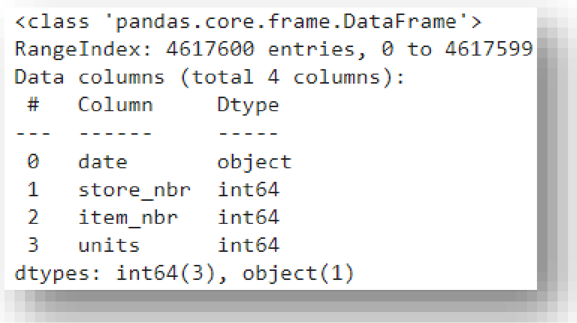
## Data-set level and output-variable analysis:

We will deal with 3 set of Data.

- "train.csv":
  This holds for a specific date at a specific store for item, how much unit(s) is/are sold
  Below tables shows the basic structure of the tabular data-



| | date | store_nbr | item_nbr | units |
|---|---|---|---|---|
| **0** | 2012-01-01 | 1 | 1 | 0 |
| **1** | 2012-01-01 | 1 | 2 | 0 |
| **2** | 2012-01-01 | 1 | 3 | 0 |
| **3** | 2012-01-01 | 1 | 4 | 0 |
| **4** | 2012-01-01 | 1 | 5 | 0 |



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4617600 entries, 0 to 4617599
Data columns (total 4 columns):
 #   Column     Dtype
---  ------     -----
 0   date       object
 1   store_nbr  int64
 2   item_nbr   int64
 3   units      int64
dtypes: int64(3), object(1)
```

While reviewing this data set we found that –

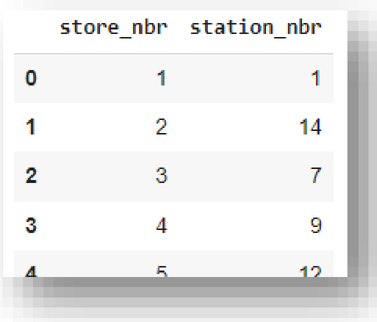
1. There are many occasions where we have 0 units sold. These rows with 0 unit sale won't help us. So we have removed them and kept the required rows only

2. Apart from Units, we have 3 other columns namely date, store_nbr, item_nbr. These variables are fixed in nature. That is, these are constants which cannot be changed as 3 of them are identifiers.

3. So in this data set we focused on the feature "units". Univariate analysis is done at this variable. We can refer the notebook "Phase2_EDA_train.ipynb" for this.

4. We found outliers in this data set, those are treated through IQR

- "key.csv":
  This file is the link between train.csv & weather.csv. This file holds the information like at what weather station which store is located. This file will help us merging 2 other files.

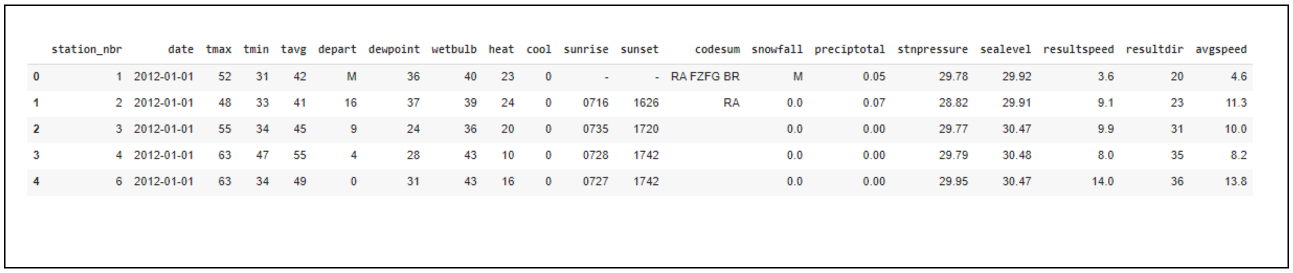


| | store_nbr | station_nbr |
|---|---|---|
| **0** | 1 | 1 |
| **1** | 2 | 14 |
| **2** | 3 | 7 |
| **3** | 4 | 9 |
| **4** | 5 | 12 |

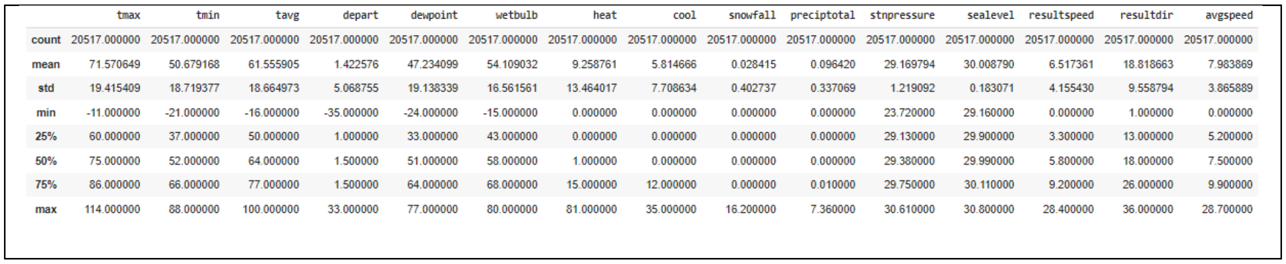
- “weather.csv”:

  This holds the information like on a specific date at a specific weather station what were the weather phenomenon. These details are captured under 20 columns, and we have data 20517 entries.

  Below tables shows the basic structure of the tabular data-



| | station_nbr | date | tmax | tmin | tavg | depart | dewpoint | wetbulb | heat | cool | sunrise | sunset | codesum | snowfall | preciptotal | stnpressure | sealevel | resultspeed | resultdir | avgspeed |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2012-01-01 | 52 | 31 | 42 | M | 36 | 40 | 23 | 0 | - | - | RA FZFG BR | M | 0.05 | 29.78 | 29.92 | 3.6 | 20 | 4.6 |
| 1 | 2 | 2012-01-01 | 48 | 33 | 41 | 16 | 37 | 39 | 24 | 0 | 0716 | 1626 | RA | 0.0 | 0.07 | 28.82 | 29.91 | 9.1 | 23 | 11.3 |
| 2 | 3 | 2012-01-01 | 55 | 34 | 45 | 9 | 24 | 36 | 20 | 0 | 0735 | 1720 | | 0.0 | 0.00 | 29.77 | 30.47 | 9.9 | 31 | 10.0 |
| 3 | 4 | 2012-01-01 | 63 | 47 | 55 | 4 | 28 | 43 | 10 | 0 | 0728 | 1742 | | 0.0 | 0.00 | 29.79 | 30.48 | 8.0 | 35 | 8.2 |
| 4 | 6 | 2012-01-01 | 63 | 34 | 49 | 0 | 31 | 43 | 16 | 0 | 0727 | 1742 | | 0.0 | 0.00 | 29.95 | 30.47 | 14.0 | 36 | 13.8 |



| | tmax | tmin | tavg | depart | dewpoint | wetbulb | heat | cool | snowfall | preciptotal | stnpressure | sealevel | resultspeed | resultdir | avgspeed |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 20517.000000 | 20517.000000 | 20517.000000 | 20517.000000 | 20517.000000 | 20517.000000 | 20517.000000 | 20517.000000 | 20517.000000 | 20517.000000 | 20517.000000 | 20517.000000 | 20517.000000 | 20517.000000 | 20517.000000 |
| mean | 71.570649 | 50.679168 | 61.555905 | 1.422576 | 47.234099 | 54.109032 | 9.258761 | 5.814666 | 0.028415 | 0.096420 | 29.169794 | 30.008790 | 6.517361 | 18.818663 | 7.983869 |
| std | 19.415409 | 18.719377 | 18.664973 | 5.068755 | 19.138339 | 16.561561 | 13.464017 | 7.708634 | 0.402737 | 0.337069 | 1.219092 | 0.183071 | 4.155430 | 9.558794 | 3.865889 |
| min | -11.000000 | -21.000000 | -16.000000 | -35.000000 | -24.000000 | -15.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 23.720000 | 29.160000 | 0.000000 | 1.000000 | 0.000000 |
| 25% | 60.000000 | 37.000000 | 50.000000 | 1.000000 | 33.000000 | 43.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 29.130000 | 29.900000 | 3.300000 | 13.000000 | 5.200000 |
| 50% | 75.000000 | 52.000000 | 64.000000 | 1.500000 | 51.000000 | 58.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 29.380000 | 29.990000 | 5.800000 | 18.000000 | 7.500000 |
| 75% | 86.000000 | 66.000000 | 77.000000 | 1.500000 | 64.000000 | 68.000000 | 15.000000 | 12.000000 | 0.000000 | 0.010000 | 29.750000 | 30.110000 | 9.200000 | 26.000000 | 9.900000 |
| max | 114.000000 | 88.000000 | 100.000000 | 33.000000 | 77.000000 | 80.000000 | 81.000000 | 35.000000 | 16.200000 | 7.360000 | 30.610000 | 30.800000 | 28.400000 | 36.000000 | 28.700000 |

Initial findings at this data set are-

1. If we review the data frame we can see that - apart from 'date' & 'codesum' column rest all should be treated as int64, but they are being treated as string

2. It is because the missing values are updated as 'M',"-",”T”, " "

3. We have replaced 'M',"-", " " with NaN, in the data frame. Converted the columns to required data format like - int64 for further proceedings.

4. We have a multi option categorical feature namely “codesum”, that is encoded

5. We had many NaN values that were filled

## Feature Analysis

- Univariate Feature Analysis:

It is the simplest form of analyzing data. “Uni” means “one”, so in other words if dataset has only one variable. It doesn’t deal with causes or relationships (unlike regression) and its major purpose is to describe; it takes data, summarizes that data and finds patterns in the data.

Univariate Descriptive Statistics (**options we have**):

Some ways you can describe patterns found in univariate data include central tendency (mean, mode and median) and dispersion: range, variance, maximum, minimum, quartiles (including the interquartile range), and standard deviation.

- UNIVARIATE SCATTER PLOT: This plots different observations/values of the same variable corresponding to the index/observation number.

- LINE PLOT (with markers): This plot visualizes data by connecting the data points via line segments. It is similar to a scatter plot except that the measurement points are ordered (typically by their x-axis value) and joined with straight line segments.

- Uni-variate summary plots: These plots give a more concise description of the location, dispersion, and distribution of a variable than an enumerative plot. It is not feasible to retrieve every individual data value in a summary plot, but it helps in efficiently representing the whole data from which better conclusions can be made on the entire data set.

  a) HISTOGRAMS: Histograms are similar to bar charts which display the counts or relative frequencies of values falling in different class intervals or ranges. A histogram displays the shape and spread of continuous sample data. It also helps us understand the skewness and kurtosis of the distribution of the data.

  b) DENSITY PLOTS: A density plot is like a smoother version of a histogram. Generally, the kernel density estimate is used in density plots to show the probability density function of the variable. A continuous curve, which is the kernel is drawn to generate a smooth density estimation for the whole data.

  c) BOX PLOTS: A box-plot is a very useful and standardized way of displaying the distribution of data based on a five-number summary (minimum, first quartile, second quartile (median), third quartile, maximum). It helps in understanding these parameters of the distribution of data and is extremely helpful in detecting outliers.

  d) Distplot (): The distplot () function of seaborn library was earlier mentioned under rug plot section. This function combines the matplotlib hist () function with the seaborn kdeplot () and rugplot () functions.

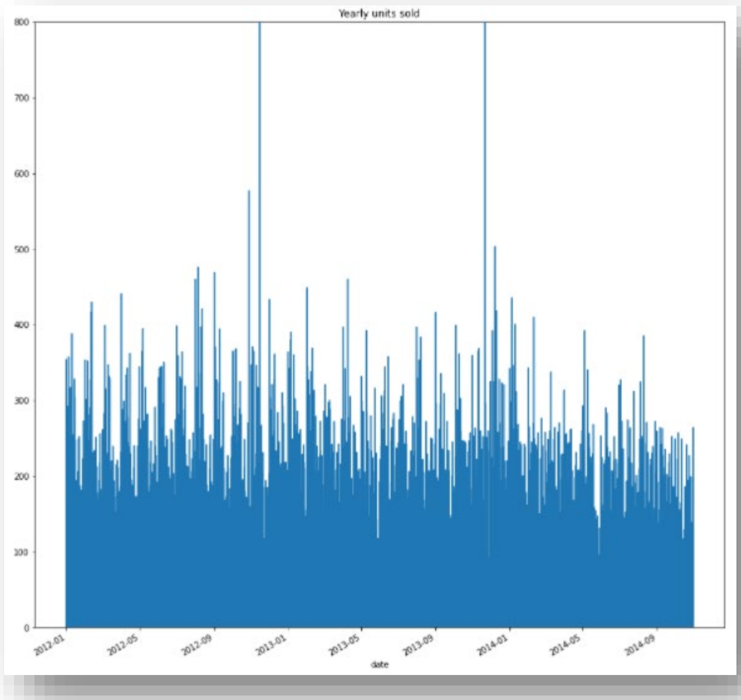
e) VIOLIN PLOTS: The Violin plot is very much similar to a box plot, with the addition of a rotated kernel density plot on each side. It shows the distribution of quantitative data across several levels of one (or more) categorical variables such that those distributions can be compared.

As mentioned earlier, in “train.csv” data set apart from “Units”, we have 3 other columns namely date, store_nbr, item_nbr. These variables are fixed in nature. That is, these are constants which cannot be changed as 3 of them are identifiers.

So we focused on the feature “units”. Univariate analysis is done on this variable. We can refer the notebook “Phase2_EDA_train.ipynb” for this.

We have used below plots here for Univariate analysis –

- ✓ Bar plot:

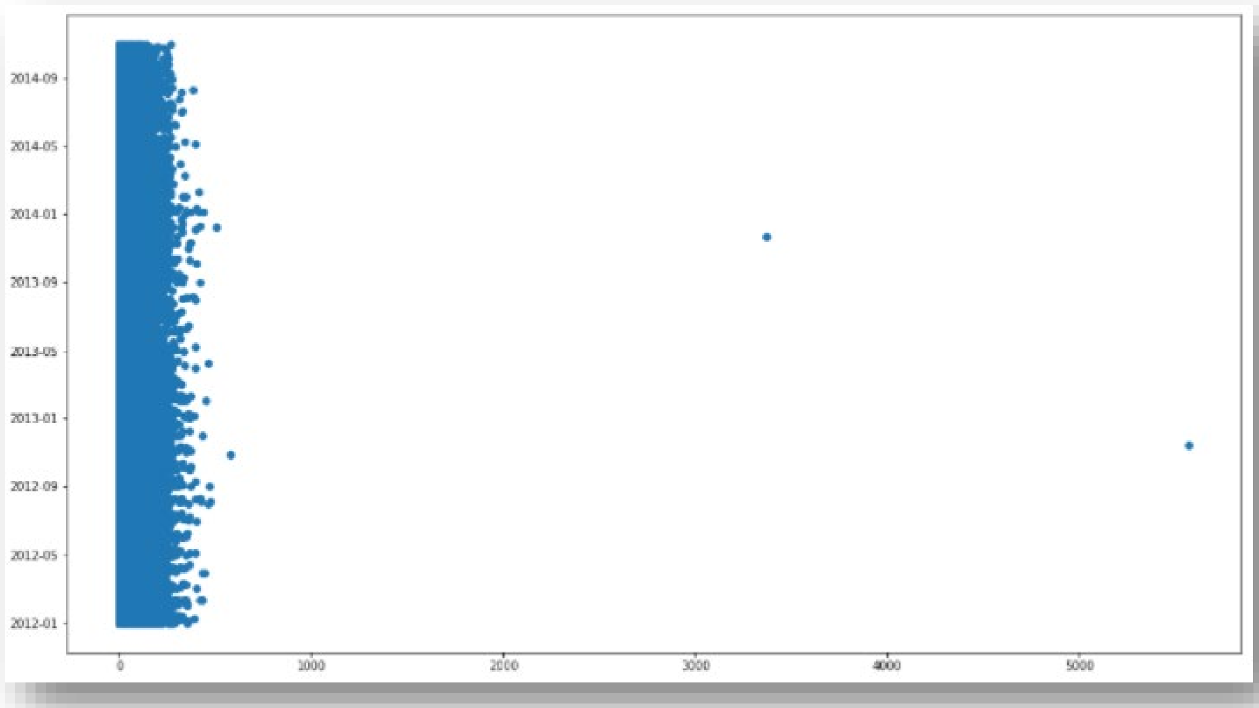


<u>Uses:</u> we have used to highlight separate values, especially the differences between these values. It is extremely useful for comparing values in different categories and can be used to describe the relationship of several variables at once. Using this plot we can clearly see the outliers. Quarter wise units sold, which quarter has pick value. Average unit sold etc.

<u>Advantages</u>: summarize a large dataset in visual form; easily compare two or three data sets; better clarify trends than do tables; estimate key values at a glance.

<u>Disadvantages</u>: require additional written or verbal explanation; can be easily manipulated to give false impressions.

✓ Scatter Plots



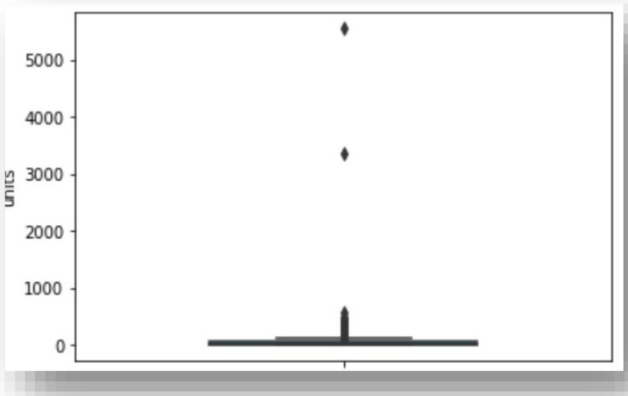
Uses: This plot illustrate paired data, that is, information regarding two related variables, in this case it's date (quarterly) Y axis & Units sold X axis. The resulting pattern (after all the points have been plotted) indicates the strength of the correlation between two variables. It's used here primarily to establish relationships, to see how data is arranged.

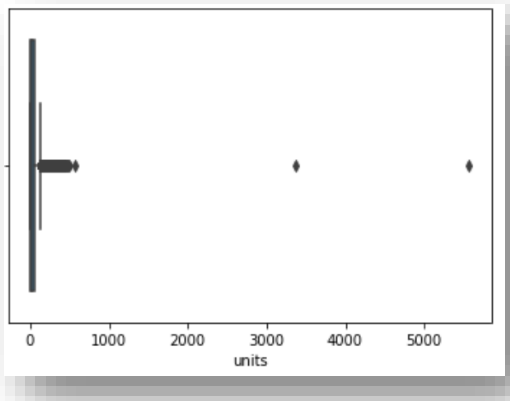
Advantages: Clearly indicates data correlation (illustrates positive, negative, strong, weak relationships); method of illustration non-linear patterns; shows spread of data, outliers; clearly demonstrate atypical relationships; used for data extrapolation and interpolation

Disadvantages: Impossible to label data points, hard to find out exact values; error bars and too many data points can quickly make graph unreadable; cannot show relationship between more than two variables at once

✓ BOX PLOTS:

Uses: It helped to check the distribution of data based on a five-number summary (minimum, first quartile, second quartile (median), third quartile, maximum). It helps in understanding these parameters of the distribution of data and is extremely helpful in detecting outliers.

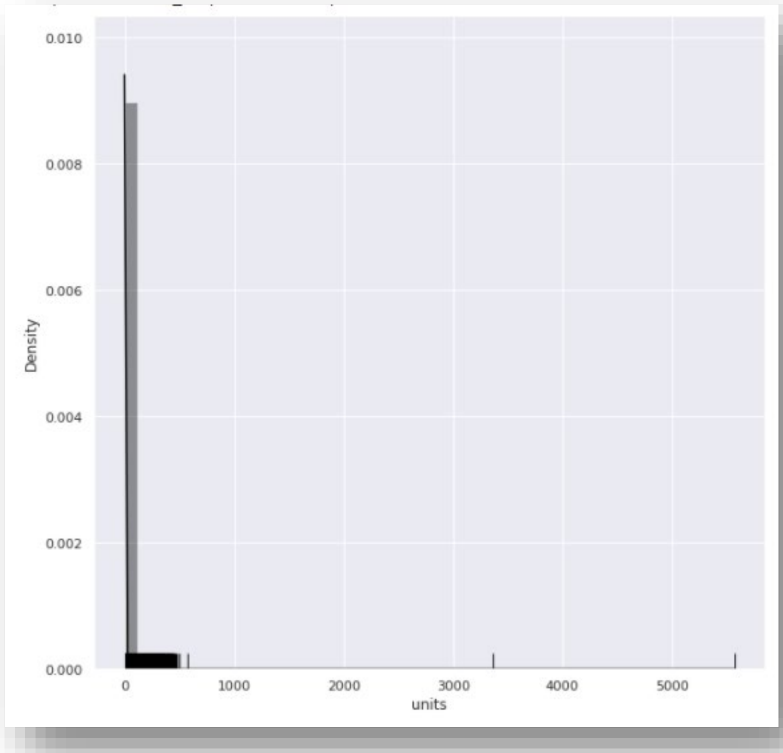




Advantages: It handles large data easily, shows a clear summary, Displays Outliers.

Disadvantages: Exact values and details of the distribution results are not retained, which is an issue with handling such large amounts of data in this graph type.

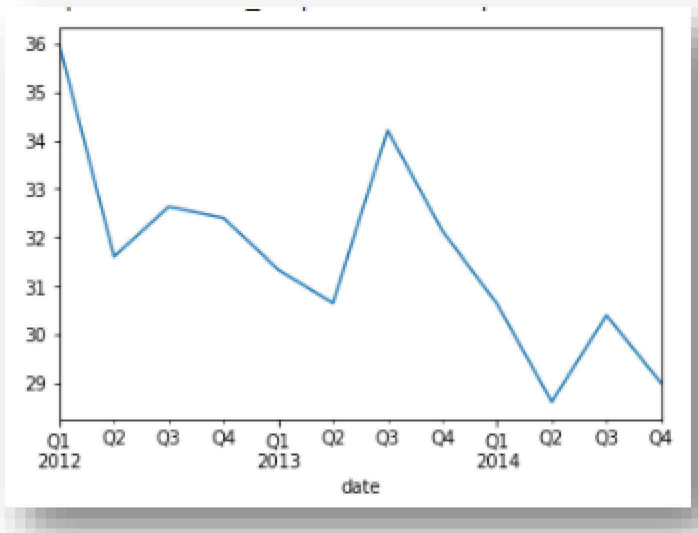
- ✓ Distplot ():



Advantage: An evident advantage is that it can easily display a general distribution of a set of numeric values over a range.

Disadvantage: One major limitation of a distribution plot is that it provides an overview of the values and their distribution. It does not give a detailed account of the ranging and distribution of the values.

- ✓ Line plot:



Uses: It is helpful to see how a value changes over time.

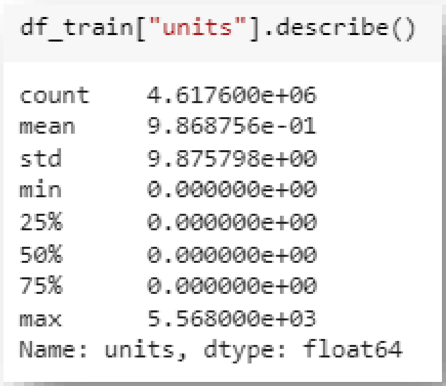
We have used this plot to see the trend of Units sold (irrespective of product, store) against quarterly date. As per our findings, in this case general trend across all the products sold (Units) is downwards. And for each year, Q3 sees more sells compared to other quarters

Advantage: It's better for seeing the rate clearly. Shows trends and relationships between data better than other graphs; compare trends in different groups of a variable; clearly show error values in the data.

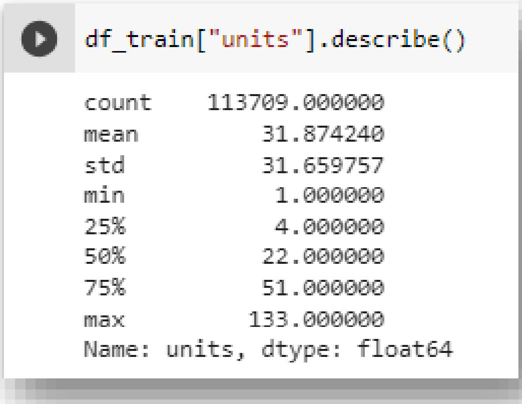
Usually simple to read and understand

Disadvantage: It's harder to compare. For multiple lines on the graph, especially unrelated can be confusing; difficult to make out exact values for data.

- ✓ Descriptive statistics:
  Other than the above descriptive statistics for the units variable was referred, to check the central tendency, variance, min, and max & to gauge outliers



```
df_train["units"].describe()

count    4.617600e+06
mean     9.868756e-01
std      9.875798e+00
min      0.000000e+00
25%      0.000000e+00
50%      0.000000e+00
75%      0.000000e+00
max      5.568000e+03
Name: units, dtype: float64
```

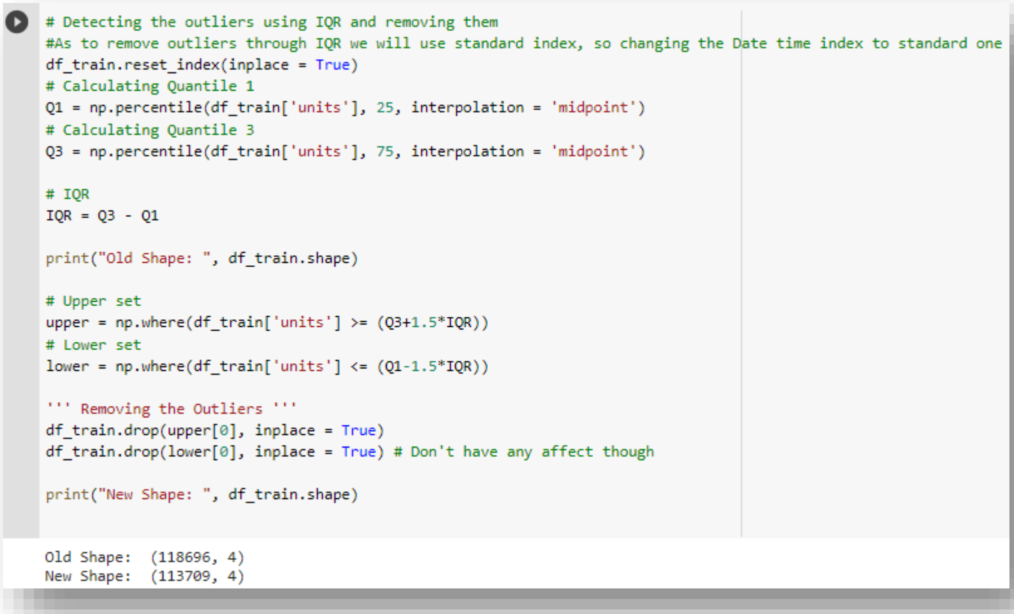


```
df_train["units"].describe()

count    113709.000000
mean         31.874240
std          31.659757
min           1.000000
25%           4.000000
50%          22.000000
75%          51.000000
max         133.000000
Name: units, dtype: float64
```

Before outlier treatment    After outlier treatment

✓ IQR (Interquartile range):
We have used IQR as well, in descriptive statistics, the interquartile range (IQR) is a measure of statistical dispersion, which is the spread of the data. The IQR may also be called the midspread, middle 50%, or H-spread. It is defined as the difference between the 75th and 25th percentiles of the data.
Advantages: It is not affected by extreme values as in the case of range. It is useful in estimating dispersion in grouped data with open ended class.

Disadvantages: IQR as a measure of dispersion is most reliable only with symmetrical data series. Unfortunately, in social sciences most of data distributions are generally asymmetrical in nature. So, its use in social sciences is usually limited to data which are moderately skewed.



```
# Detecting the outliers using IQR and removing them
#As to remove outliers through IQR we will use standard index, so changing the Date time index to standard one
df_train.reset_index(inplace = True)
# Calculating Quantile 1
Q1 = np.percentile(df_train['units'], 25, interpolation = 'midpoint')
# Calculating Quantile 3
Q3 = np.percentile(df_train['units'], 75, interpolation = 'midpoint')

# IQR
IQR = Q3 - Q1

print("Old Shape: ", df_train.shape)

# Upper set
upper = np.where(df_train['units'] >= (Q3+1.5*IQR))
# Lower set
lower = np.where(df_train['units'] <= (Q1-1.5*IQR))

''' Removing the Outliers '''
df_train.drop(upper[0], inplace = True)
df_train.drop(lower[0], inplace = True) # Don't have any affect though

print("New Shape: ", df_train.shape)
```

```
Old Shape:  (118696, 4)
New Shape:  (113709, 4)
```

- Multivariate Feature analysis:

Multivariate means involving multiple dependent variables resulting in one outcome. This explains that the majority of the problems in the real world are Multivariate. For example, in this case we can predict sales based on multiple weather related features like temperature, Snowfall, dew etc.

It is a Statistical procedure for analysis of data involving more than one type of measurement or observation. It may also mean solving problems where more than one dependent variable is analyzed simultaneously with other variables.

Apart from reviewing min, max, std etc. Multivariate feature analysis also includes feature selections.

Different options we have:

- Pairwise plots: Pairwise plots are a great way to look at multi-dimensional data, and at the same time maintain the simplicity of a two-dimensional plot. It allows the analysts to view all combinations of the variables, each in a two-dimensional plot. In this way, they can visualize all the relations and interactions among the variables on one single screen.
  Disadvantage of pairwise plot is – if we have more features then generating output takes time, size of the plot becomes huge, difficult to review

- Correlation Analysis: Often, data sets contain variables that are either related to each other or derived from each other. It is important to understand these relations that exist in the data. In statistical terms, correlation can be defined as the degree to which a pair of variables are linearly related. In some cases, it is easy for the analyst to understand that the variables are related, but in most cases, it isn't. Thus, performing a correlation analysis is very critical while examining any data. Furthermore, feeding data which has variables correlated to one another is not a good statistical practice, since we are providing multiple weightage to the same type of data. To prevent such issues, correlation analysis is a must

- Principal Component Analysis and Factor Analysis: Although machine learning is a game of predicting the result given multiple predictors, there can be times when the number of these predictors is too large. Not only is such a data set difficult to analyze, but the models formed using this are susceptible to overfitting. Therefore, it makes sense to have the number of these variables reduced. Principal component analysis (PCA) and Factor analysis are two of the common techniques used to perform such a dimension reduction.

  PCA reduces the existing number of variables, such that the new set of reduced variables capture most of the total variance present in the existing set of

variables. Many times, we usually have only two or three features in this new set of features. What this allows us is to visualize all of the initial information in 2-D or 3-D plots, and thus aid in exploratory data analysis. Furthermore, these new features are a combination of the initial features which helps us understand the variables which are important.
Therefore, PCA is such a powerful tool for analysts since they now have a much smaller feature set to deal with, and at the same time having preserved most of the information which was initially present. While PCA extracts factors based on the total variance, the Factor Analysis Method extracts factors based on the variance shared by the factors. By providing the factors based on the variance they share, Factor Analysis enables data scientists to examine the underlying trends in the data.

- Other than the above we have Spider Plots, Cluster Analysis, MANOVA (Multivariate Analysis of Variance), Discriminant Analysis, Conjoint Analysis, Multiple Regression Analysis

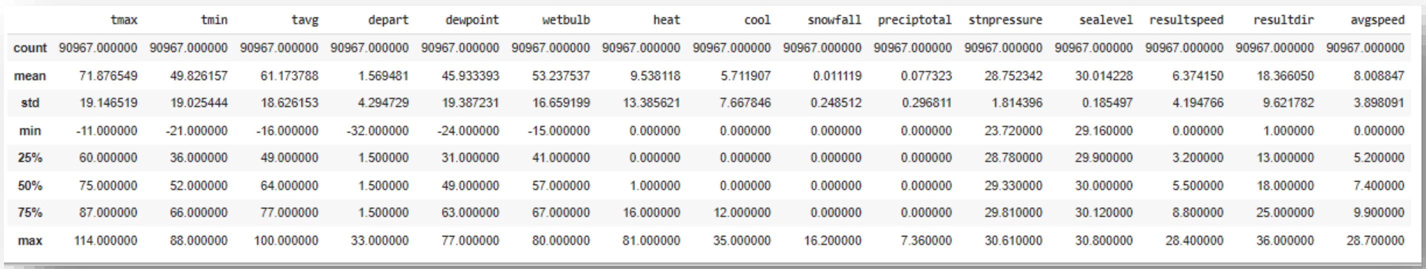
We have done Multivariate Feature analysis/Selection on "Phase2_EDA_Merged.ipynb", which is actually merger of train_cleaned.csv, weather.csv using key.csv file.

After merging we had 113709 entries under 47 columns, 1 dependent target column. So dimensionality here too much, we will need to reduce the same using careful approach.

After train and test spilt we have 90967 train and 22742 test data.

By checking descriptive statistics of the below columns we can find the below:

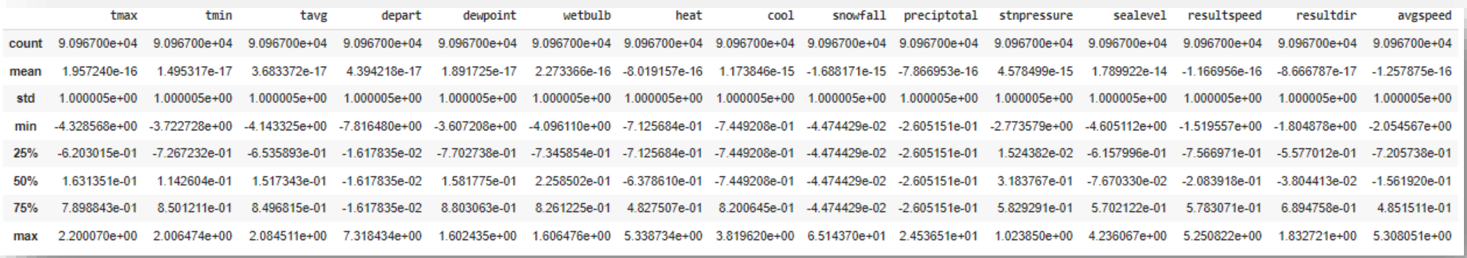
```
['tmax',  'tmin', 'tavg', 'depart', 'dewpoint', 'wetbulb',  'heat', 'cool'
, 'snowfall', 'preciptotal',  'stnpressure',  'sealevel', 'resultspeed',
'resultdir',  'avgspeed']
```



| | tmax | tmin | tavg | depart | dewpoint | wetbulb | heat | cool | snowfall | preciptotal | stnpressure | sealevel | resultspeed | resultdir | avgspeed |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 90967.000000 | 90967.000000 | 90967.000000 | 90967.000000 | 90967.000000 | 90967.000000 | 90967.000000 | 90967.000000 | 90967.000000 | 90967.000000 | 90967.000000 | 90967.000000 | 90967.000000 | 90967.000000 | 90967.000000 |
| mean | 71.876549 | 49.826157 | 61.173788 | 1.569481 | 45.933393 | 53.237537 | 9.538118 | 5.711907 | 0.011119 | 0.077323 | 28.752342 | 30.014228 | 6.374150 | 18.366050 | 8.008847 |
| std | 19.146519 | 19.025444 | 18.626153 | 4.294729 | 19.387231 | 16.659199 | 13.385621 | 7.667846 | 0.248512 | 0.296811 | 1.814396 | 0.185497 | 4.194766 | 9.621782 | 3.898091 |
| min | -11.000000 | -21.000000 | -16.000000 | -32.000000 | -24.000000 | -15.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 23.720000 | 29.160000 | 0.000000 | 1.000000 | 0.000000 |
| 25% | 60.000000 | 36.000000 | 49.000000 | 1.500000 | 31.000000 | 41.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 28.780000 | 29.900000 | 3.200000 | 13.000000 | 5.200000 |
| 50% | 75.000000 | 52.000000 | 64.000000 | 1.500000 | 49.000000 | 57.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 29.330000 | 30.000000 | 5.500000 | 18.000000 | 7.400000 |
| 75% | 87.000000 | 66.000000 | 77.000000 | 1.500000 | 63.000000 | 67.000000 | 16.000000 | 12.000000 | 0.000000 | 0.000000 | 29.810000 | 30.120000 | 8.800000 | 25.000000 | 9.900000 |
| max | 114.000000 | 88.000000 | 100.000000 | 33.000000 | 77.000000 | 80.000000 | 81.000000 | 35.000000 | 16.200000 | 7.360000 | 30.610000 | 30.800000 | 28.400000 | 36.000000 | 28.700000 |

We found that we will need to standardize the data as the data are in different scales.

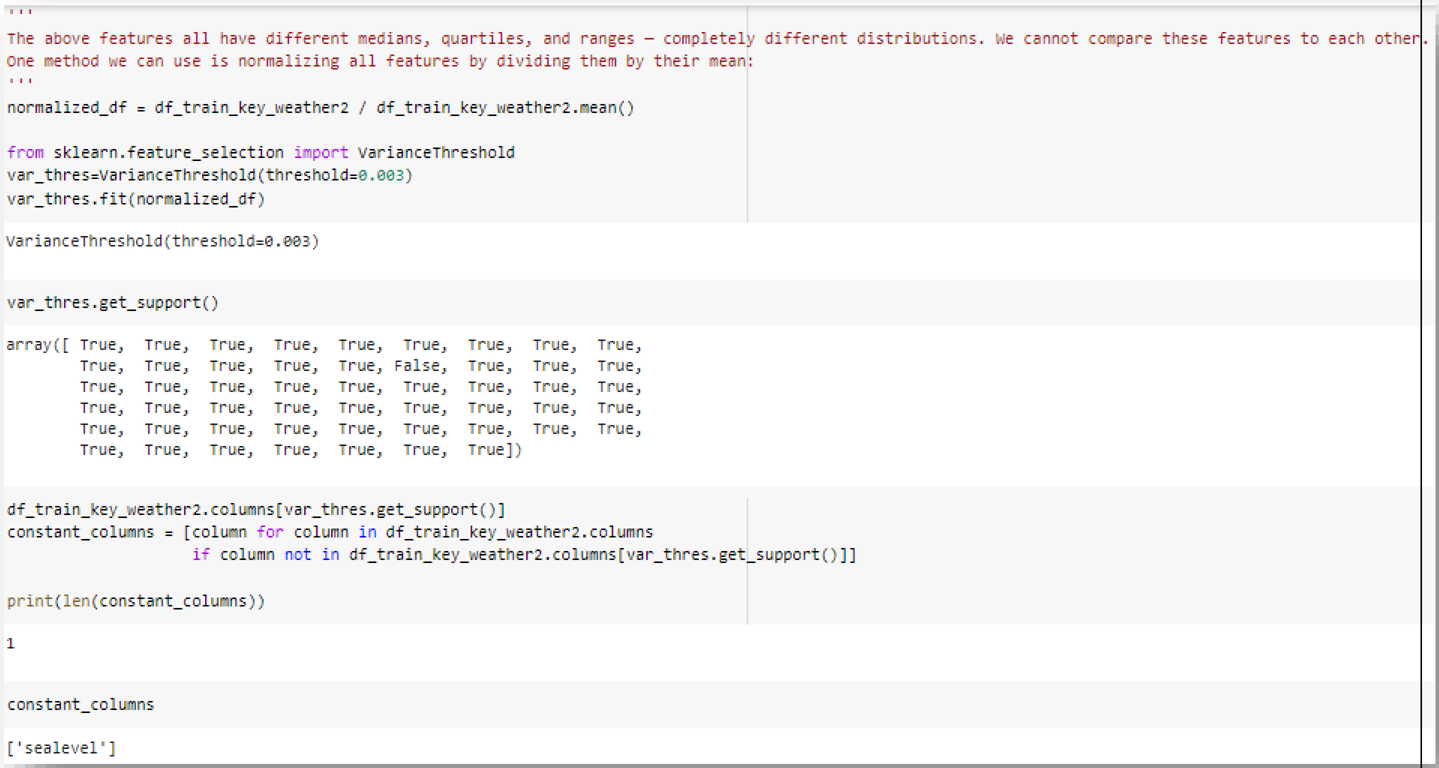
And as we do not have properly distributed data we have not done Normalization.



| | tmax | tmin | tavg | depart | dewpoint | wetbulb | heat | cool | snowfall | preciptotal | stnpressure | sealevel | resultspeed | resultdir | avgspeed |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 9.096700e+04 | 9.096700e+04 | 9.096700e+04 | 9.096700e+04 | 9.096700e+04 | 9.096700e+04 | 9.096700e+04 | 9.096700e+04 | 9.096700e+04 | 9.096700e+04 | 9.096700e+04 | 9.096700e+04 | 9.096700e+04 | 9.096700e+04 | 9.096700e+04 |
| mean | 1.957240e-16 | 1.495317e-17 | 3.683372e-17 | 4.394218e-17 | 1.891725e-17 | 2.273366e-16 | -8.019157e-16 | 1.173846e-15 | -1.688171e-15 | -7.866953e-16 | 4.578499e-15 | 1.789922e-14 | -1.166956e-16 | -8.666787e-17 | -1.257875e-16 |
| std | 1.000005e+00 | 1.000005e+00 | 1.000005e+00 | 1.000005e+00 | 1.000005e+00 | 1.000005e+00 | 1.000005e+00 | 1.000005e+00 | 1.000005e+00 | 1.000005e+00 | 1.000005e+00 | 1.000005e+00 | 1.000005e+00 | 1.000005e+00 | 1.000005e+00 |
| min | -4.328568e+00 | -3.722728e+00 | -4.143325e+00 | -7.816480e+00 | -3.607208e+00 | -4.096110e+00 | -7.125684e-01 | -7.449208e-01 | -4.474429e-02 | -2.605151e-01 | -2.773579e+00 | -4.605112e+00 | -1.519557e+00 | -1.804878e+00 | -2.054567e+00 |
| 25% | -6.203015e-01 | -7.267232e-01 | -6.535893e-01 | -1.617835e-02 | -7.702738e-01 | -7.345854e-01 | -7.125684e-01 | -7.449208e-01 | -4.474429e-02 | -2.605151e-01 | 1.524382e-02 | -6.157996e-01 | -7.566971e-01 | -5.577012e-01 | -7.205738e-01 |
| 50% | 1.631351e-01 | 1.142604e-01 | 1.517343e-01 | -1.617835e-02 | 1.581775e-01 | 2.258502e-01 | -6.378610e-01 | -7.449208e-01 | -4.474429e-02 | -2.605151e-01 | 3.183767e-01 | -7.670330e-02 | -2.083918e-01 | -3.804413e-02 | -1.561920e-01 |
| 75% | 7.898843e-01 | 8.501211e-01 | 8.496815e-01 | -1.617835e-02 | 8.803063e-01 | 8.261225e-01 | 4.827507e-01 | 8.200645e-01 | -4.474429e-02 | -2.605151e-01 | 5.829291e-01 | 5.702122e-01 | 5.783071e-01 | 6.894758e-01 | 4.851511e-01 |
| max | 2.200070e+00 | 2.006474e+00 | 2.084511e+00 | 7.318434e+00 | 1.602435e+00 | 1.606476e+00 | 5.338734e+00 | 3.819620e+00 | 6.514370e+01 | 2.453651e+01 | 1.023850e+00 | 4.236067e+00 | 5.250822e+00 | 1.832721e+00 | 5.308051e+00 |

- ✓ To reduce the dimensionality we have used Variance Threshold Feature selector And Pearson Correlation

- ✓ Variance Threshold Feature selector:
  This removes all low-variance features. This feature selection algorithm looks only at the features (X), not the desired outputs (y), and can thus be used for unsupervised learning. By using this we can remove 10 constant columns, as shown below (Threshold set as 0.001)



```
'''
The above features all have different medians, quartiles, and ranges — completely different distributions. We cannot compare these features to each other.
One method we can use is normalizing all features by dividing them by their mean:
'''
normalized_df = df_train_key_weather2 / df_train_key_weather2.mean()

from sklearn.feature_selection import VarianceThreshold
var_thres=VarianceThreshold(threshold=0.003)
var_thres.fit(normalized_df)
```

```
VarianceThreshold(threshold=0.003)
```

```
var_thres.get_support()
```

```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True, False,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True])
```

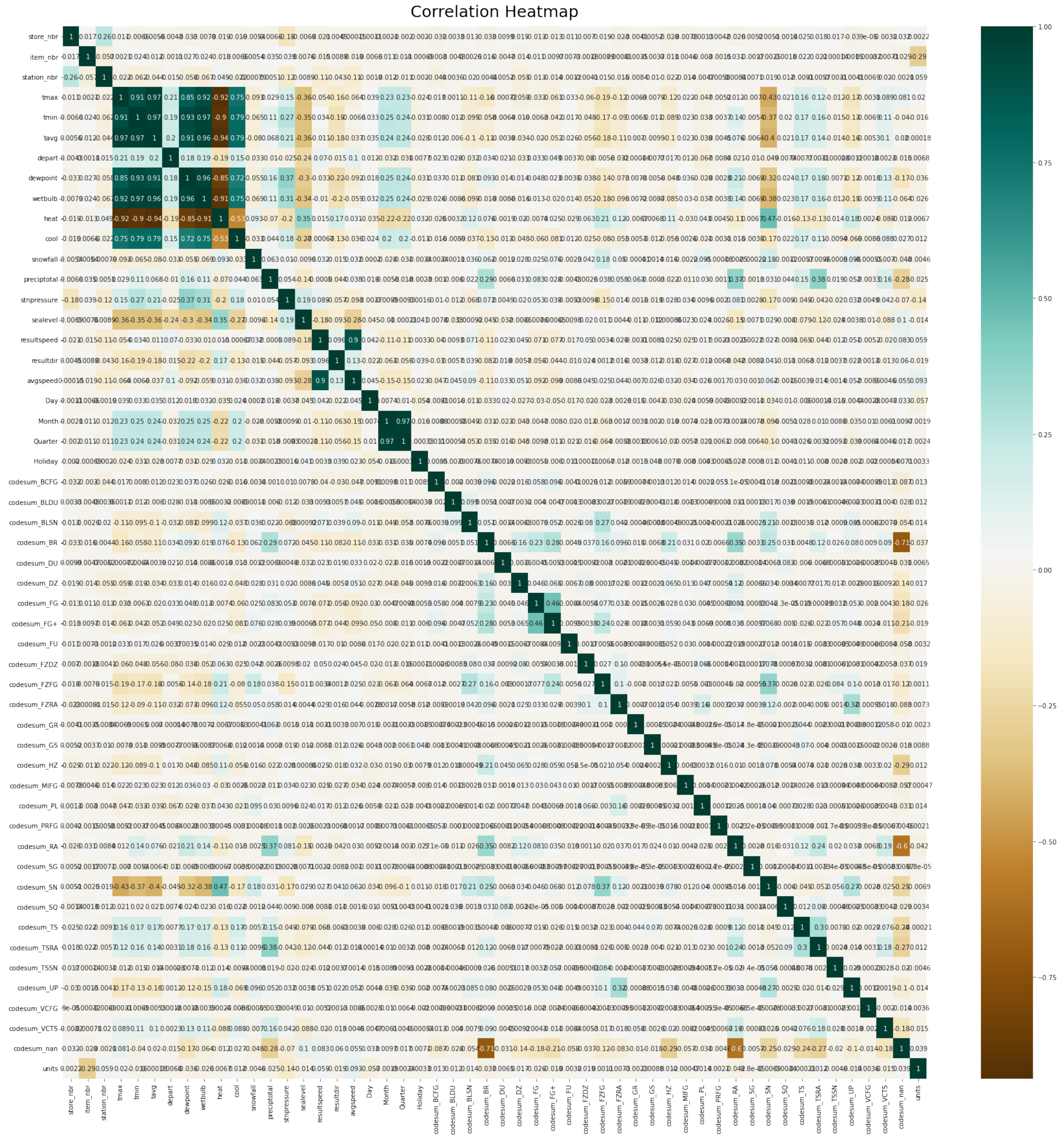
```
df_train_key_weather2.columns[var_thres.get_support()]
constant_columns = [column for column in df_train_key_weather2.columns
                    if column not in df_train_key_weather2.columns[var_thres.get_support()]]

print(len(constant_columns))
```

```
1
```

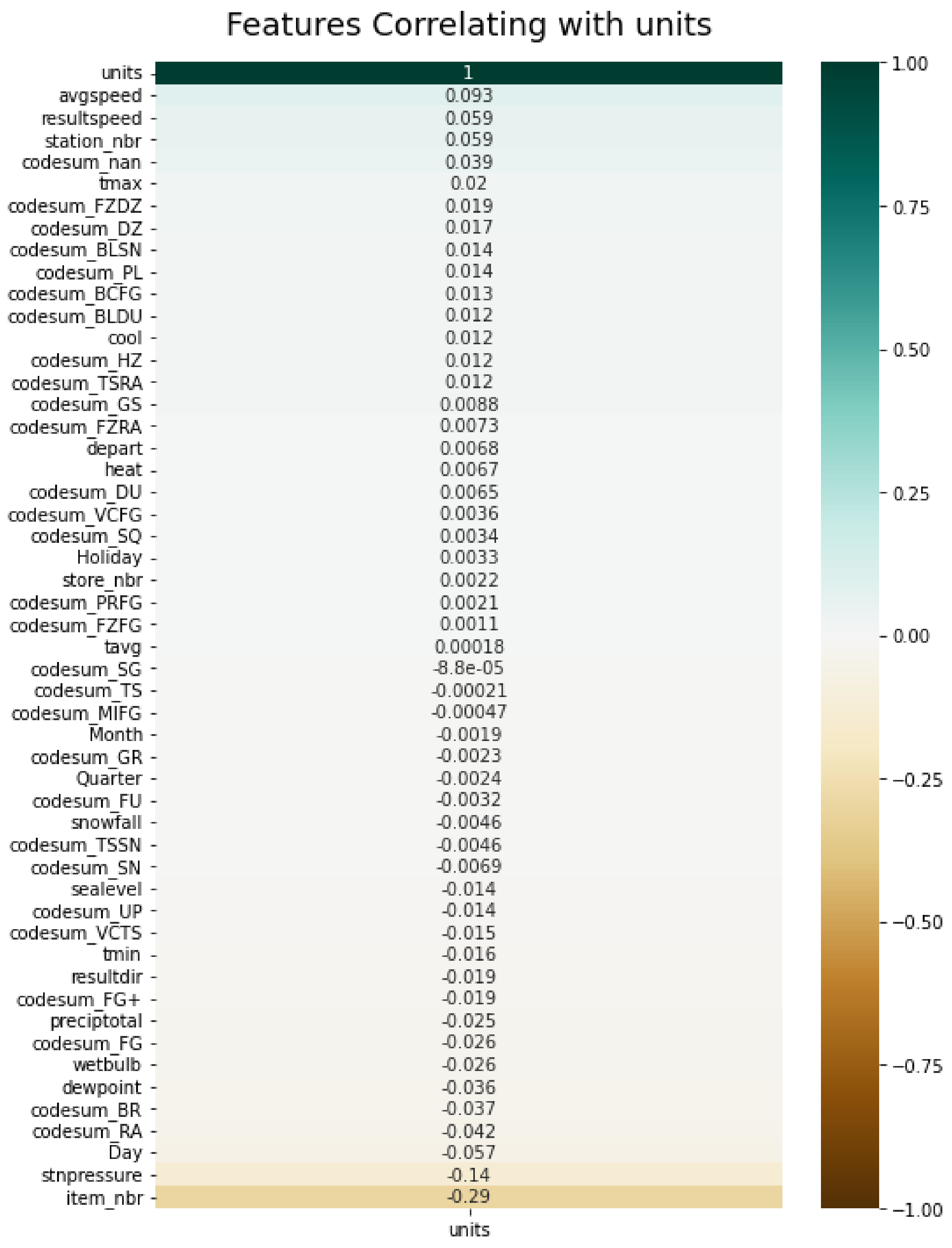
```
constant_columns
```

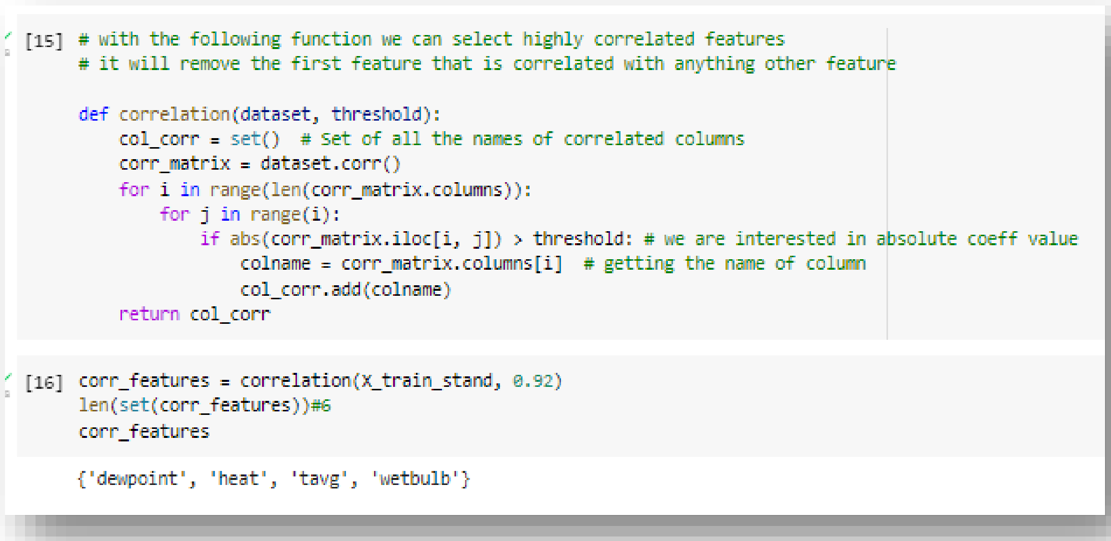
```
['sealevel']
```

✓ Pearson Correlation:
Using this feature we are going to keep only 1 feature out of multiple correlated features, again this will help us reducing the dimensionality.



We have used below function to read from the above and pick one feature among all other corelated features. After removing constants and corelated features we have 33 features with us now.


Features Correlating with units
units 1
avgspeed 0.093
resultspeed 0.059
station_nbr 0.059
codesum_nan 0.039
tmax 0.02
codesum_FZDZ 0.019
codesum_DZ 0.017
codesum_BLSN 0.014
codesum_PL 0.014
codesum_BCFG 0.013
codesum_BLDU 0.012
cool 0.012
codesum_HZ 0.012
codesum_TSRA 0.012
codesum_GS 0.0088
codesum_FZRA 0.0073
depart 0.0068
heat 0.0067
codesum_DU 0.0065
codesum_VCFG 0.0036
codesum_SQ 0.0034
Holiday 0.0033
store_nbr 0.0022
codesum_PRFG 0.0021
codesum_FZFG 0.0011
tavg 0.00018
codesum_SG -8.8e-05
codesum_TS -0.00021
codesum_MIFG -0.00047
Month -0.0019
codesum_GR -0.0023
Quarter -0.0024
codesum_FU -0.0032
snowfall -0.0046
codesum_TSSN -0.0046
codesum_SN -0.0069
sealevel -0.014
codesum_UP -0.014
codesum_VCTS -0.015
tmin -0.016
resultdir -0.019
codesum_FG+ -0.019
preciptotal -0.025
codesum_FG -0.026
wetbulb -0.026
dewpoint -0.036
codesum_BR -0.037
codesum_RA -0.042
Day -0.057
stnpressure -0.14
item_nbr -0.29
units
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00



```
[15] # with the following function we can select highly correlated features
     # it will remove the first feature that is correlated with anything other feature

     def correlation(dataset, threshold):
         col_corr = set()  # Set of all the names of correlated columns
         corr_matrix = dataset.corr()
         for i in range(len(corr_matrix.columns)):
             for j in range(i):
                 if abs(corr_matrix.iloc[i, j]) > threshold: # we are interested in absolute coeff value
                     colname = corr_matrix.columns[i]  # getting the name of column
                     col_corr.add(colname)
         return col_corr
```

```
[16] corr_features = correlation(X_train_stand, 0.92)
     len(set(corr_features))#6
     corr_features

     {'dewpoint', 'heat', 'tavg', 'wetbulb'}
```

## Feature encoding

The performance of a machine learning model not only depends on the model and the hyper parameters but also on how we process and feed different types of variables to the model. Since most machine learning models only accept numerical variables, preprocessing the categorical variables becomes a necessary step. We need to convert these categorical variables to numbers such that the model is able to understand and extract valuable information.

Categorical variables are usually represented as ‘strings’ or ‘categories’ and are finite in number. Further, we can see there are two kinds of categorical data-

Ordinal Data: The categories have an inherent order. In Ordinal data, while encoding, we should retain the information regarding the order in which the category is provided.

Nominal Data: The categories do not have an inherent order. While encoding Nominal data, we have to consider the presence or absence of a feature. In such a case, no notion of order is present.

- Options we have to do feature encoding:

- Label Encoding or Ordinal Encoding:
  We use this categorical data encoding technique when the categorical feature is ordinal. In this case, retaining the order is important. Hence encoding should reflect the sequence.
  In Label encoding, each label is converted into an integer value.

- One Hot Encoding:

We use this categorical data encoding technique when the features are nominal (do not have any order). In one hot encoding, for each level of a categorical feature, we create a new variable. Each category is mapped with a binary variable containing either 0 or 1. Here, 0 represents the absence, and 1 represents the presence of that category.
These newly created binary features are known as Dummy variables. The number of dummy variables depends on the levels present in the categorical variable.

- Dummy Encoding:
  Dummy coding scheme is similar to one-hot encoding. This categorical data encoding method transforms the categorical variable into a set of binary variables (also known as dummy variables). In the case of one-hot encoding, for N categories in a variable, it uses N binary variables. The dummy encoding is a small improvement over one-hot-encoding. Dummy encoding uses N-1 features to represent N labels/categories.

- Other than the above techniques we have - Effect Encoding, Hash Encoder, Binary Encoding, Base N Encoding, Target Encoding

• Here in this project we have a feature namely "codesum", which is multi-option nominal data. It has 29 unique values.
  We have used one hot encoding for this.

  Advantage: One-Hot-Encoding has the advantage that the result is binary rather than ordinal and that everything sits in an orthogonal vector space.

  Disadvantage: The disadvantage is that for high cardinality, the feature space can really blow up quickly and you start fighting with the curse of dimensionality.

  Hence to reduce the dimensionality we have used Variance Threshold Feature selector And Pearson Correlation feature.
  We have not used PCA as we were not sure what number we should keep as parameter (i.e. how many features we should have).

## Advanced Feature encoding mechanisms

There are other advanced feature encoding techniques are also available, based on the requirement below can be used.

- Matrix Factorization

We can do sparse encoding for more-interpretable feature-selecting representations in probabilistic matrix factorization.

Dimensionality reduction methods for count data are critical to a wide range of applications in medical informatics and other fields where model interpretability is paramount. For such data, hierarchical Poisson matrix factorization (HPF) and other sparse probabilistic non-negative matrix factorization (NMF) methods are considered to be interpretable generative models. They consist of sparse transformations for decoding their learned representations into predictions. However, sparsity in representation decoding does not necessarily imply sparsity in the encoding of representations from the original data features. HPF is often incorrectly interpreted in the literature as if it possesses encoder sparsity. The distinction between decoder sparsity and encoder sparsity is subtle but important. Due to the lack of encoder sparsity, HPF does not possess the column-clustering property of classical NMF -- the factor loading matrix does not sufficiently define how each factor is formed from the original features. We address this deficiency by self-consistently enforcing encoder sparsity, using a generalized additive model (GAM), thereby allowing one to relate each representation coordinate to a subset of the original data features. In doing so, the method also gains the ability to perform feature selection.

- Encoding using Deep-Learning:

Many machine learning algorithms and almost all deep learning architectures are incapable of processing plain texts in their raw form. This means that their input to the algorithms must be numerical in order to solve classification or regression problems. Hence, it is necessary to encode these categorical variables into numerical values using encoding techniques. Categorical features are common and often of high cardinality. One-hot encoding in such circumstances leads to very high dimensional vector representations, raising memory and computability concerns for machine learning models. This paper proposes a deep-learned embedding technique for categorical features encoding on categorical datasets. Our technique is a distributed representation for categorical features where each category is mapped to a distinct vector, and the properties of the vector are learned while training a neural network. First, we create a data vocabulary that includes only categorical data, and then we use word tokenization to make each categorical data a single word. After that, feature learning is introduced to map all of the categorical data from the vocabulary to word vectors. Three different datasets provided by the University of California Irvine (UCI) are used for training. The experimental results show that the proposed deep-learned embedding technique for categorical data provides a higher F1 score of 89% than 71% of one-hot encoding, in the case of the long short-term memory (LSTM) model. Moreover, the deep-learned embedding technique uses less memory and generates fewer features than one-hot encoding.

## High-dimensional data visualization:

- t-SNE:
- For high dimensional data visualization we can use t-SNE, which maps the multi-dimensional data to a lower-dimensional space, the input features are not useful and convey no information. Thus, no inferences can be drawn simply from the output of t-SNE alone.
  It is a Dimensionality Reduction technique and not a Clustering technique. But its output can be used as an input feature for other classification or clustering algorithms.

## References

1. https://www.analyticsvidhya.com/blog/2020/07/univariate-analysis-visualization-with-illustrations-in-python/
2. https://www.mygreatlearning.com/blog/introduction-to-multivariate-analysis/
3. https://scikit-learn.org/stable/auto_examples/feature_selection/plot_feature_selection.html
4. https://blog.datadive.net/selecting-good-features-part-i-univariate-selection/
5. https://topepo.github.io/caret/feature-selection-using-univariate-filters.html
6. https://www.statisticshowto.com/univariate/
7. http://www.kmrom.com/Site-En/Articles/ViewArticle.aspx?ArticleID=416
8. https://data-flair.training/blogs/qlik-sense-distribution-plot/
9. https://online.stat.psu.edu/stat555/node/15/
10. https://datatron.com/multivariate-analysis-techniques-for-exploring-data/
11. https://www.analyticsvidhya.com/blog/2020/08/types-of-categorical-data-encoding/
12. https://datascience.stackexchange.com/questions/9443/when-to-use-one-hot-encoding-vs-labelencoder-vs-dictvectorizor
13. https://www.kaggle.com/prashanththangavel/advanced-feature-engineering-feature-encoding
14. https://towardsdatascience.com/the-art-of-effective-visualization-of-multi-dimensional-data-6c7202990c57
15. https://ieeexplore.ieee.org/abstract/document/9512057
16. https://blog.clairvoyantsoft.com/mlmuse-visualisation-of-high-dimensional-data-using-t-sne-ac6264316d7f
17. https://distill.pub/2016/misread-tsne/