**2.2 Describe the difference in roles assumed by the validation partition and the test partition.**

Validation Partition (sometimes called the “test” partition) is used to assess the performance of each model, so that you can compare models and pick the best one. In some algorithms (e.g. classification and regression trees), the validation partition may be used in automated fashion to tune and improve the model.

Test Partition (sometimes called the “holdout” or “evaluation” partition) is used if we need to assess the performance of the chosen model with new data.

Why have both a validation and a test partition? When we use the validation data to assess multiple models and then pick the model that does best with the validation data, we again encounter another (lesser) facet of the overfitting problem – chance aspects of the validation data that happen to match the chosen model better than other models. Applying the model to the test data, which it has not seen before, will provide an unbiased estimate of how well it will do with new data

**2.7 A dataset has 1000 records and 50 variables. 5% of the values are missing, spread randomly throughout the records and variables. An analyst decides to remove records that have missing values. About how many records would you expect to be removed?**

Probability that a given record would NOT be missing a value is:



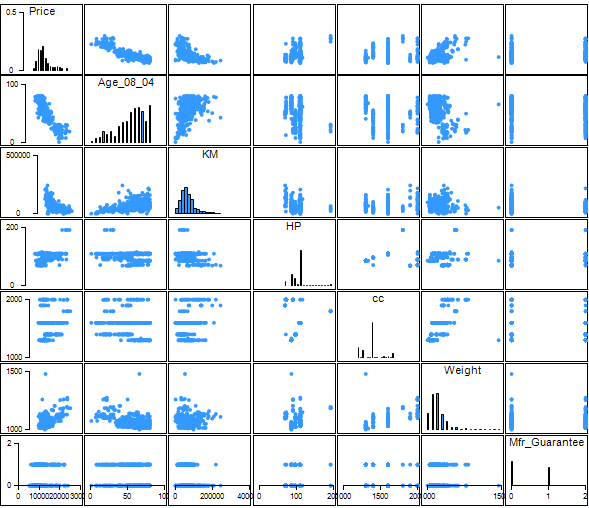
Therefore about 0.923 percent of the records would expect to be removed. (~923 records removed)

**2.11 The dataset ToyotaCorolla.xls contains data on used cars on sale during the late summer of 2004 in the Netherlands. It has 1436 records containing details on 38 attributes including Price, Age, Kilometers, Horsepower, and other specifications.**

**(a) Explore the data using the data visualization (matrix plot) capabilities of the XLMiner. Which of the pairs among the variables seem to be correlated?**

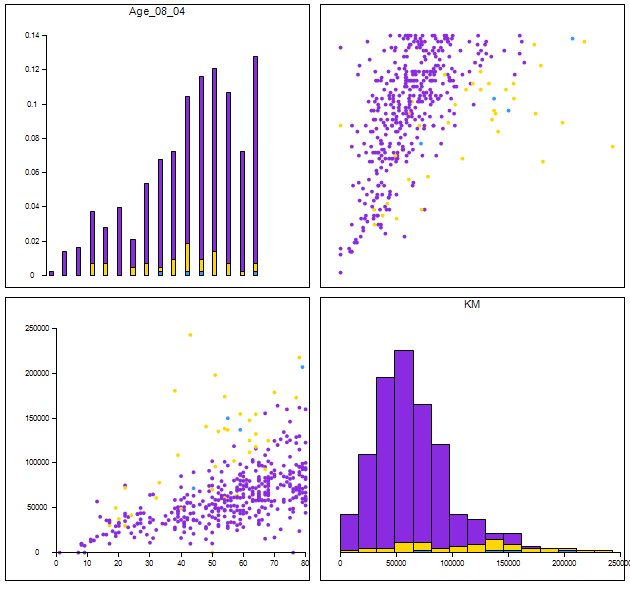
There are many correlations to be observed where the patterns move together either on the X or Y axis. Just to name a few are: Price vs Age / Price vs KM / Price vs HP / Age vs KM / Age vs HP / Age vs cc / Weight vs Price / Weight vs HP.

I have provided zoomed in views on Age vs KM and Price vs Age below.



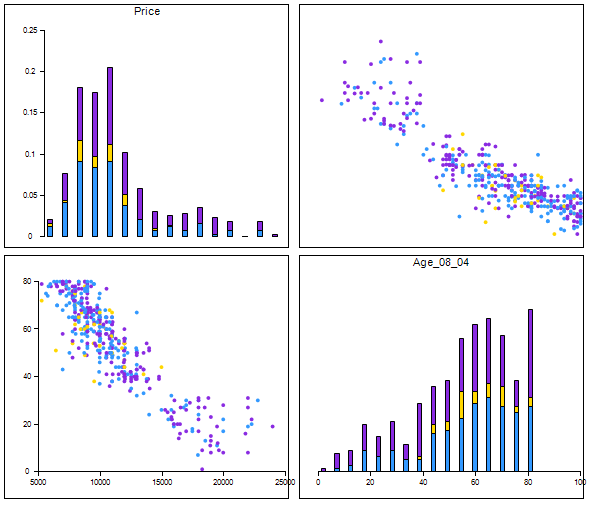
Age vs KM (traveled) separated by Fuel Type in color

Hard Drive:Users:laml:Desktop:Screen Shot 2014-10-29 at 10.42.14 AM.png



Price vs Age separated by # of Doors in color

Hard Drive:Users:laml:Desktop:Screen Shot 2014-10-29 at 10.54.07 AM.png



**2.11 (b) We plan to analyze the data using various data mining techniques to be covered in future chapters. Prepare the data for use as follows:**

**i. The dataset has two categorical attributes, Fuel−Type(3) and Metallic.**

1. **Describe how you would convert these to binary variables.**

The categorical variable Fuel\_Type has 3x values that can be listed as binary or Yes/No values. The categorical variable Met\_Color is already in binary.

|  |  |
| --- | --- |
| **Fuel\_Type** | **Binary** |
| Diesel | Yes/No |
| Petrol | Yes/No |
| CNG | Yes/No |

|  |  |
| --- | --- |
| **Met\_Color** | **Binary** |
| 0 | No |
| 1 | Yes |

1. **Confirm this using XLMiner’s utility to transform categorical data into dummies.**

XLMinor transformed the categorical data for Fuel\_Type into binary dummies by creating the following new categories where Yes = 1 and No = 0. Met\_Color is already in binary form.

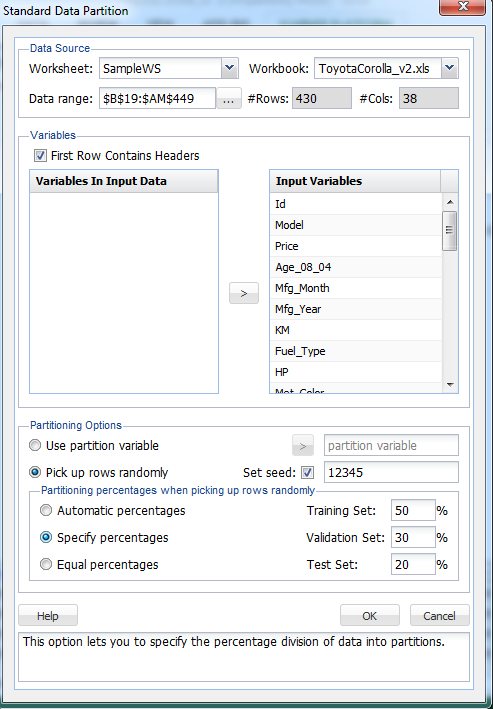
|  |  |  |
| --- | --- | --- |
| **Fuel\_Type\_CNG** | **Fuel\_Type\_Diesel** | **Fuel\_Type\_Petrol** |

1. **How would you work with these new variables to avoid including redundant information in models?**

For Fuel\_Type, note that only two of the variables need to be used - if the values of two are known, the third is also known by reasoning.

For example, given that these three values are the only possible ones, we can know that if a Toyota Corolla is neither CNG nor Diesel, it must be Petrol. In some routines (e.g. regression and logistic regression), you should not use all three variables - the redundant information will cause the algorithm to fail.

1. **Prepare the dataset (as factored into dummies) for data mining techniques of supervised learning by creating partitions using XLMiner’s data partitioning utility. Select all the variables and use default values for the random seed and partitioning percentages for training (50%), validation (30%) and test (20%) sets. Describe the roles that these partitions will play in modeling.**



**Training data** (or training set) refers to that portion of data used to fit a model.

**Validation data** (or validation set) refers to that portion of the data used to assess how well the model fits, to adjust some models, and to select the best model from among those that have been tried.

**Test data** (or test set) refers to that portion of the data used only at the end of the model building and selection process to assess how well the final model might perform on additional data.

**3.3 Laptop Sales at a London Computer Chain: Bar Charts and Boxplots. The file LaptopSalesJanuary2008.xls contains data for all sales of laptops at a computer chain in London in January 2008. This is a subset of the full dataset that includes data for the entire year.**

1. **Create a bar chart, showing the average retail price by store. Which store has the highest average? Which has the lowest?**

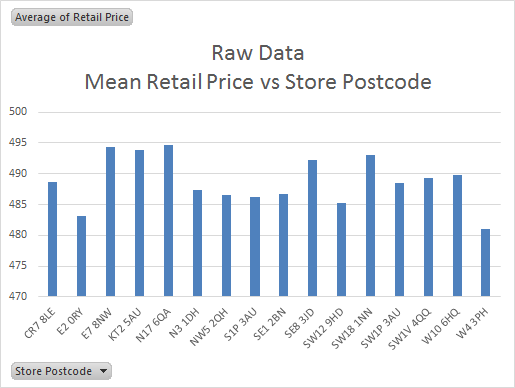
**Raw Data Mean**

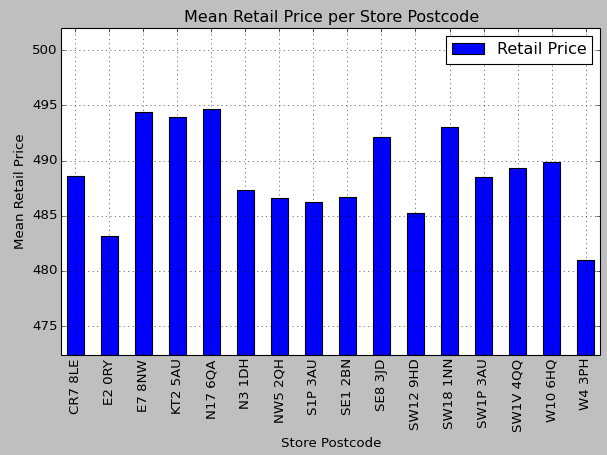
*Highest Retail Price Mean:* 494.6

Store Postcode N17 6QA

*Lowest Retail Price Mean:* 482.2

Store Postcode W4 3PH

****



**Python code:**

import pandas as pd #plot graph

import matplotlib.pyplot as plt #ylabel

#read file into data2 variable

file = '/Users/laml/Documents/Fall 2014/ESD.754 Data Mining/HW1/LaptopSalesJanuary2008.txt'

data2 = pd.read\_csv(file, sep='\t', index\_col='Date', parse\_dates=True)

#create pivot table for Retail Price values categorized by Store Postcode

#plot as bar graph

data2.pivot\_table(values='Retail Price', index=['Store Postcode']).plot(kind='bar', title="Mean Retail Price per Store Postcode", legend=True)

#label y

plt.ylabel("Mean Retail Price")

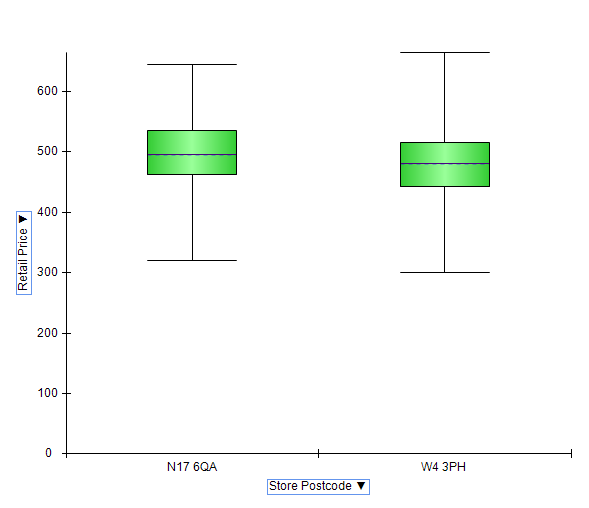
1. **To better compare retail prices across stores, create side-by –side boxplots of retail price by store. Now compare the prices in the two stores above. Do you see a difference between their price distributions? Explain.**

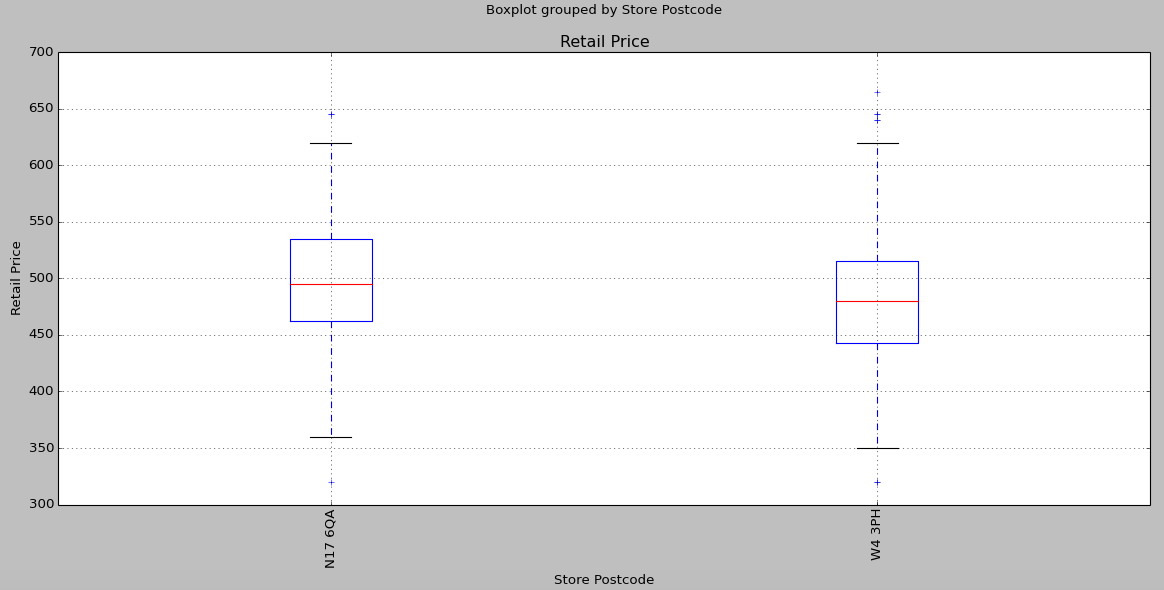
**Raw Data (7956 input data)**

|  |  |
| --- | --- |
| **Count of Retail Price** |  |
| Store Postcode | Total |
| W4 3PH | 159 |
| N17 6QA | 123 |
|  |  |
|  |  |
| **Max of Retail Price** |  |
| Store Postcode | Total |
| W4 3PH | 665 |
| N17 6QA | 645 |
|  |  |
| **Min of Retail Price** |  |
| Store Postcode | Total |
| W4 3PH | 300 |
| N17 6QA | 320 |
|  |  |
|  |  |
| **Avg of Retail Price** |  |
| Store Postcode | Total |
| W4 3PH | 481.0062893 |
| N17 6QA | 494.6341463 |

* Range: W4 3PH (665-300=365) is **greater than** N17 6QA (645-320=325)
* Mean: W4 3PH (481.0) is **less than** N17 6QA (494.6)

**Raw Data Boxplot**

****

****

**Python code:**

import pandas as pd #plot graph

import matplotlib.pyplot as plt #ylabel

#read file into data2 variable

file = '/Users/laml/Documents/Fall 2014/ESD.754 Data Mining/HW1/LaptopSalesJanuary2008.txt'

data2 = pd.read\_csv(file, sep='\t', index\_col='Date', parse\_dates=True)

#create pivot table for Retail Price values categorized by Store Postcode

#plot as boxchart graph with specific values

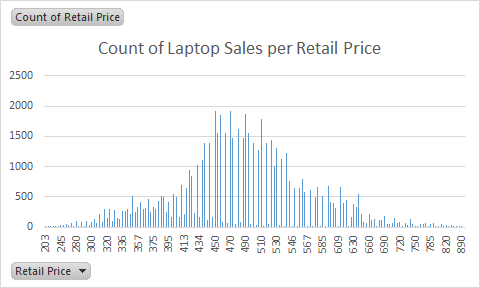
data2[data2['Store Postcode']==('N17 6QA')].append(data2[data2['Store Postcode']==('W4 3PH')]).boxplot(column='Retail Price',by='Store Postcode',rot='90')

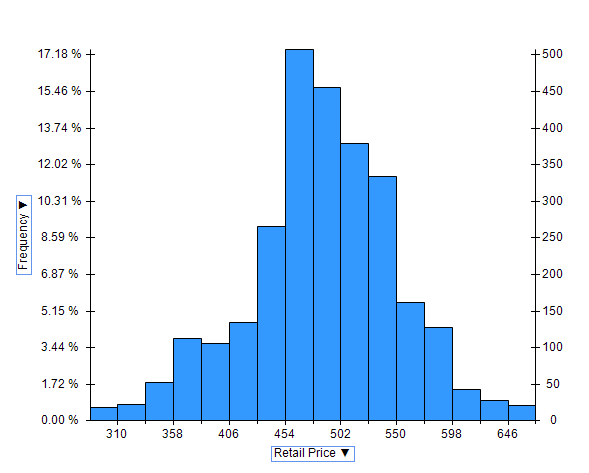
#label y

plt.ylabel("Retail Price")

**3.4 Laptop Sales at a London Computer Chain: Interactive Visualization.**

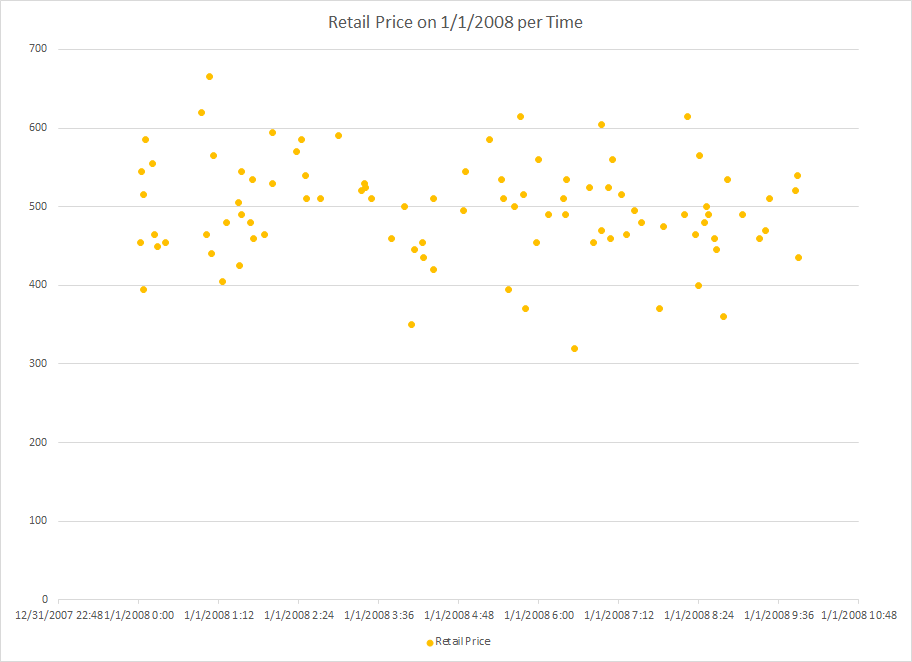
1. **Price Questions**
2. **At what prices are the laptops actually selling?**

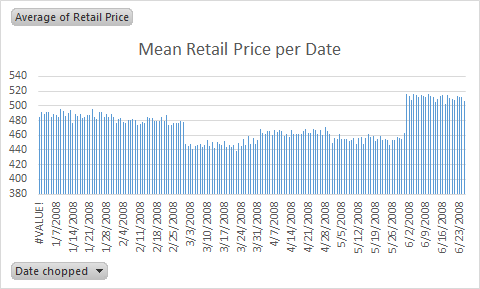
****



1. **Does price change with time?**

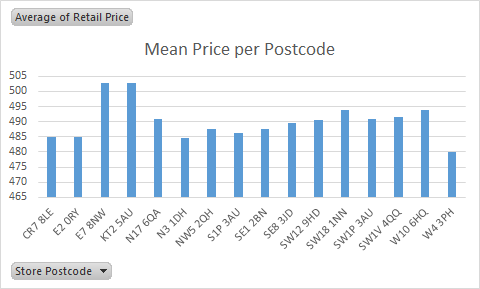
* Price is affected with time as you can see from following charts. Average price per day changes and even price throughout 1 datechanges**.**

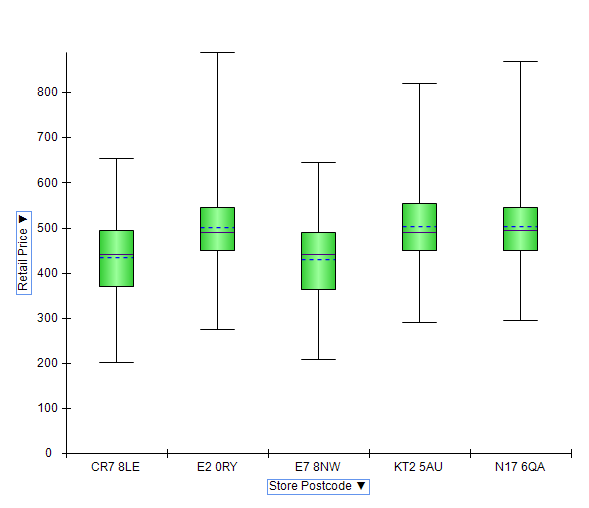
****

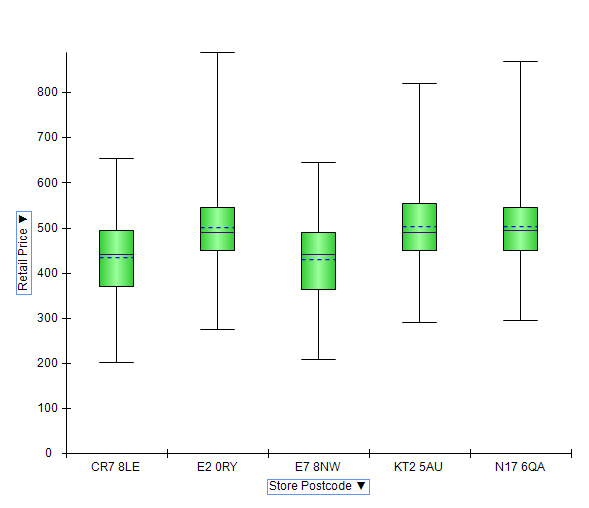
****

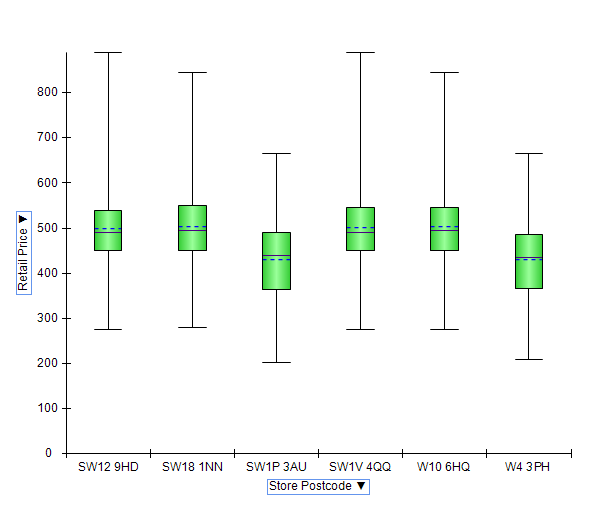
1. **Are prices consistent across retail outlets?**

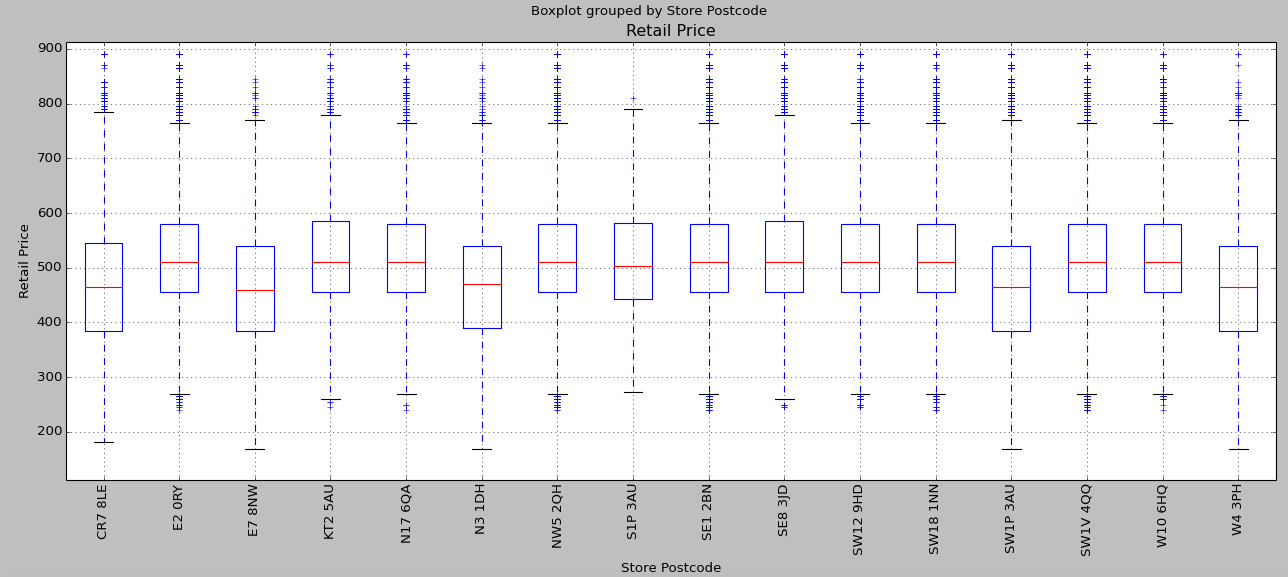
* Based on the graphs for Mean vs Postcode – the lowest Mean value is 479.86 and the maximum Mean value is 502.94 where the difference is only ~22 from each other.
* Based on the boxplots per Store Postcode, the ranges for each lower 25% and upper 75% box are mostly touching each other, which infers that the prices are pretty close to each other.

****

****

****

****

****

**Python code:**

import pandas as pd #plot graph

import matplotlib.pyplot as plt #ylabel

#read file into data2 variable

file = '/Users/laml/Documents/Fall 2014/ESD.754 Data Mining/HW1/LaptopSales.txt'

data2 = pd.read\_csv(file, sep='\t', index\_col='Date', parse\_dates=True)

#create pivot table for Retail Price values categorized by Store Postcode

#plot as boxchart graph

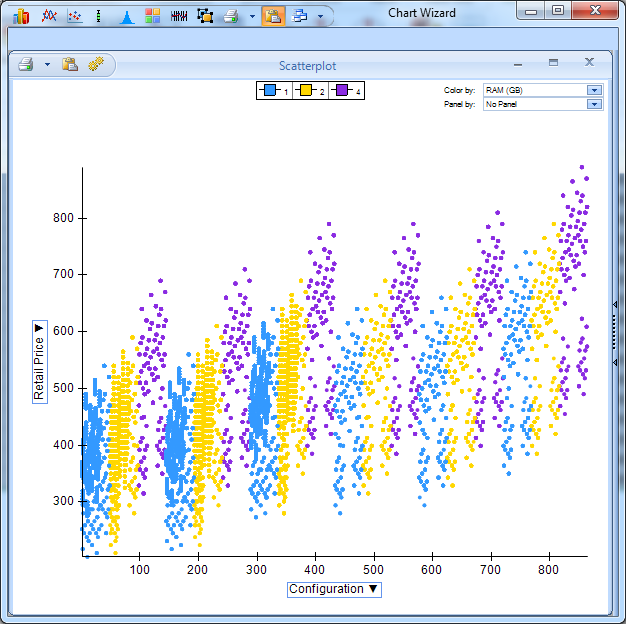
data2.boxplot(column='Retail Price',by='Store Postcode')

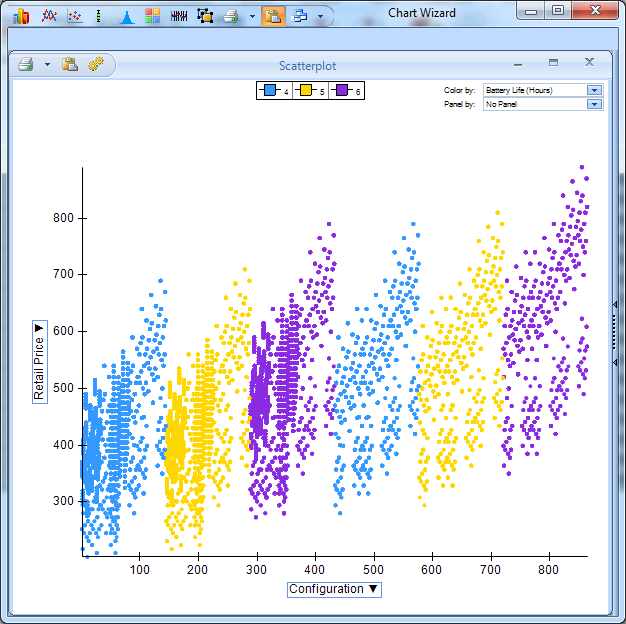
#label y

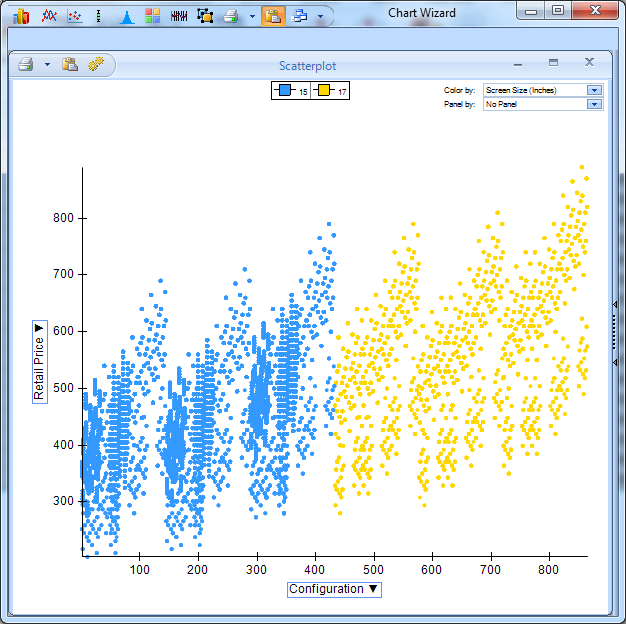
plt.ylabel("Retail Price")

1. **How does price change with configuration?**

* Price increases with increased configuration. I have broken down the graphs by RAM (GB), Battery Life, and Screen Size.

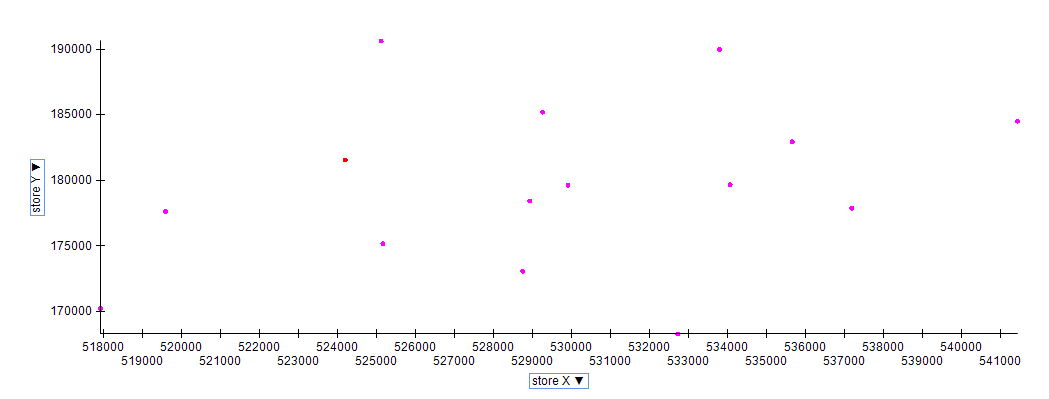
****

****

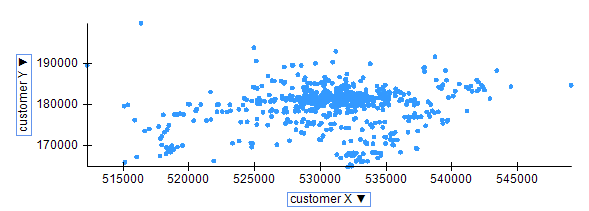
****

1. **Location Questions**
   1. **Where are the stores and customers located?**

**Stores Location X, Y**

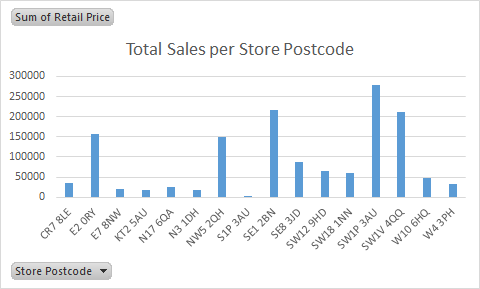
****

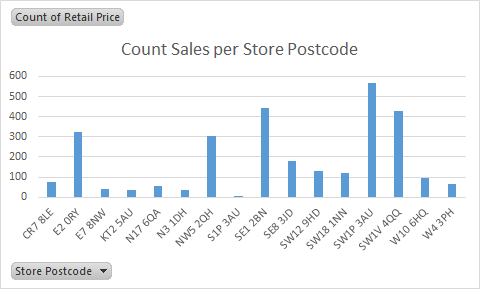
**Customer Location X, Y with Stores in RED**

****

* 1. **Which stores are selling the most?**

|  |  |  |
| --- | --- | --- |
| Store Postcode | Total Revenue | Count of Sales |
| SW1P 3AU | 277825 | 566 |
| SE1 2BN | 216485 | 444 |
| SW1V 4QQ | 210445 | 428 |

****

****

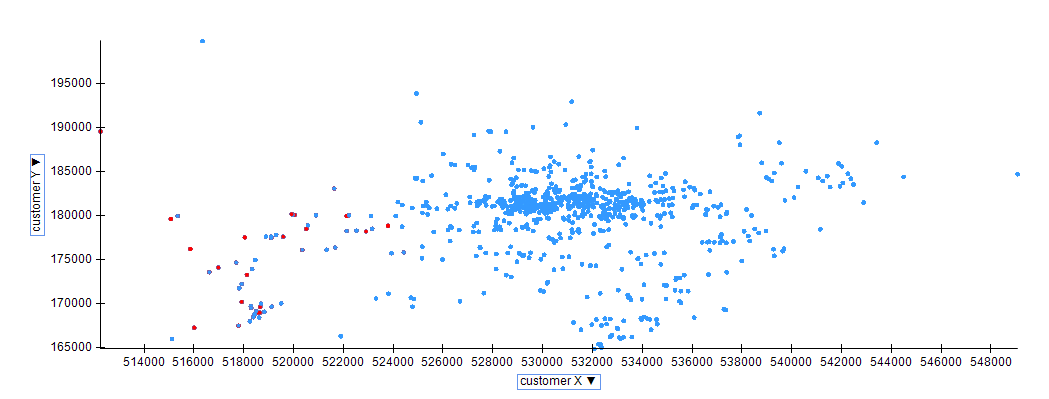
* 1. **How far would customers travel to buy a laptop?**

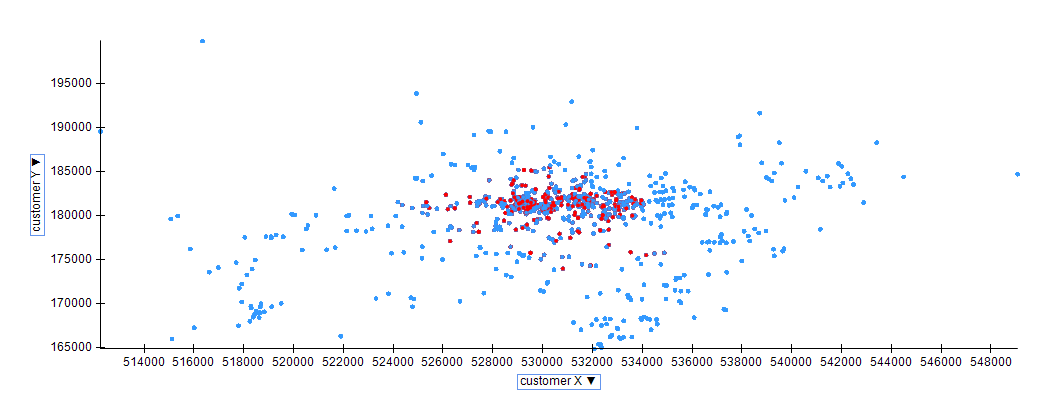
Using the distance formula:

The top 20 distances customers traveled are:

|  |  |  |
| --- | --- | --- |
| No. | Customer Postcode | Distance |
| 1 | WD7 7NP | 19892.14041 |
| 2 | HA5 5PW | 14382.64572 |
| 3 | RM10 8QS | 13722.54718 |
| 4 | KT7 0JP | 13582.22633 |
| 5 | CR0 1NF | 12451.79521 |
| 6 | CR0 4HA | 12400.95404 |
| 7 | CR0 1NA | 12212.56734 |
| 8 | CR0 6PU | 12179.05653 |
| 9 | CR0 1QD | 12155.31892 |
| 10 | CR0 6BR | 12103.88004 |

* 1. **Try an alternative way of looking at how far customers traveled.**
* Small red dots represent customers who traveled to particular store, which is circled.



****

**5.4 A large number of insurance records are to be examined to develop a model for predicting fraudulent claims. Of the claims in the historical database, 1% were judged to be fraudulent. A sample is taken to develop a model, and oversampling is used to provide a balanced sample in light of the very low response rate. When applied to this sample (N = 800), the model ends up correctly classifying 310 frauds, and 270 nonfrauds. It missed 90 frauds, and classified 130 records incorrectly as frauds when they were not.**

1. **Produce the classification matrix for the sample as it stands.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Predicted | |  |
|  |  | Fraud | Nonfraud | 800 |
| Actual | Fraud | 310 | 90 | 400 |
| Nonfraud | 130 | 270 | 400 |
|  | 800 | 440 | 360 |  |

1. **Find the adjusted misclassification rate (adjusting for the oversampling).**

==27.5%

(Total)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Predicted | |  |
|  |  | Fraud | Nonfraud | 40,000 |
| Actual | Fraud | 310 | 90 | 400 |
| Nonfraud | 12,870 | 26,730 | 39,600 |
|  | 40,000 | 13,180 | 26,820 |  |

==32.4% *adjusted for oversampling*

1. **What percentage of new records would you expect to be classified as fraudulent?**

of new records would be expected to be classified as fraudulent correctly.

of new records would be expected to be classified as nonfraudulent correctly.