NEW PRODUCT PROFITABILITY REPORT

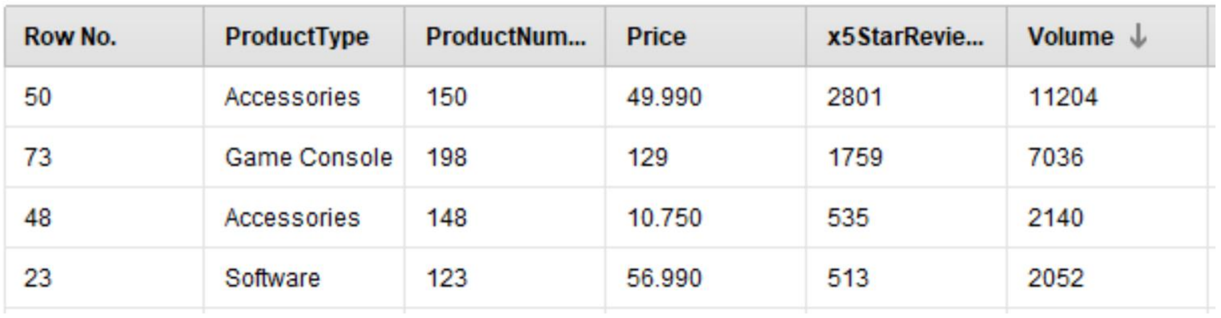
Maja Jurcan

1. Overview of the existing product dataset

Overview of the data in the given dataset shows that there are 18 attributes. The reoccurring data type is integer or real which is needed data type for the regression analysis. There is one polynominal attribute Product Type which will have to be omitted from this analysis.

Our main focus is the sales volume attribute. The range of this attribute is from 0 to 11204 with the average value of 705 and the standard deviation of 1517.

The product with the highest sales volume is the product which has the most 5 star reviews from users. Two out of top four existing product with the highest sales volume are accessories, with the other two being game console and software (Figure 1.1).



Row No.	ProductType	ProductNum...	Price	x5StarRevie...	Volume ↓
50	Accessories	150	49.990	2801	11204
73	Game Console	198	129	1759	7036
48	Accessories	148	10.750	535	2140
23	Software	123	56.990	513	2052

Figure 1.1: Highest sales volume products table

2. Preprocessing the data

The dataset is first cleared of the unnecessary or irregular data.

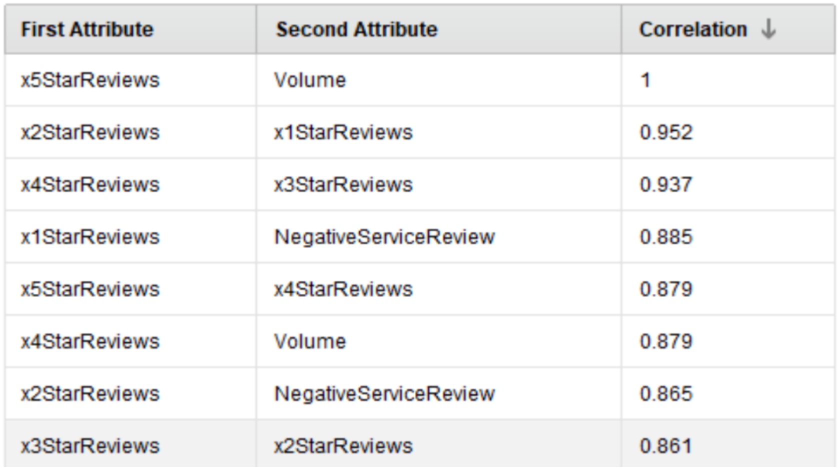
The attribute product type is first to be removed because its polynominal (not a number) and not valid for regression analysis. Removing product number is second in row for removal because it is not relevant.

Best seller rank attribute is also removed because it has 15 missing values out of 80 and it makes problems in linear regression analysis.

At first glance, shipping weight attribute and product depth, width and height attributes don't seem relevant for the sales prediction analysis, but they are left in the dataset unless further analysis shows their irrelevance. Since we are looking for an attribute connection between successful products any of the given data could be useful.

Also, if we want to build a good model and have good prediction of the data, the dataset should be inspected for high rate of correlation, meaning that two independent variables are highly or even directly related to each other. One of those variables needs to be deleted. Choosing which one means that every variable should be inspected, and the one that gets deleted is the one that is less connected to the sales volume attribute.

First, the 5 star review attribute gets removed because of the too high correlation rate with the sales volume attribute. Next, the attributes are removed by the rule explained before and according to the data from the correlation matrix. The correlation matrix shows that the attributes that need to be removed are all the remaining starred reviews except the 4 star review (Figure 2.1).



First Attribute	Second Attribute	Correlation ↓
x5StarReviews	Volume	1
x2StarReviews	x1StarReviews	0.952
x4StarReviews	x3StarReviews	0.937
x1StarReviews	NegativeServiceReview	0.885
x5StarReviews	x4StarReviews	0.879
x4StarReviews	Volume	0.879
x2StarReviews	NegativeServiceReview	0.865
x3StarReviews	x2StarReviews	0.861

Figure 2.1: Correlation table

After cleaning and preprocessing the data we are left with the 11 most relevant attributes.

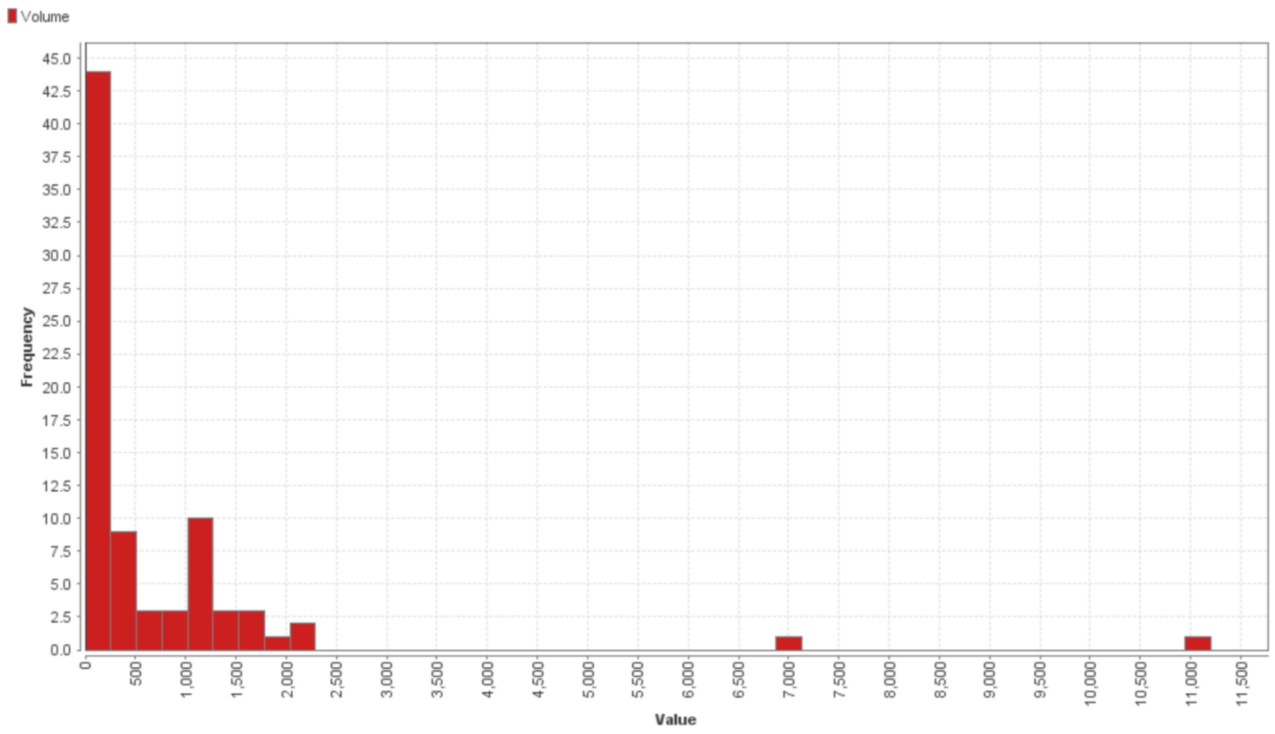
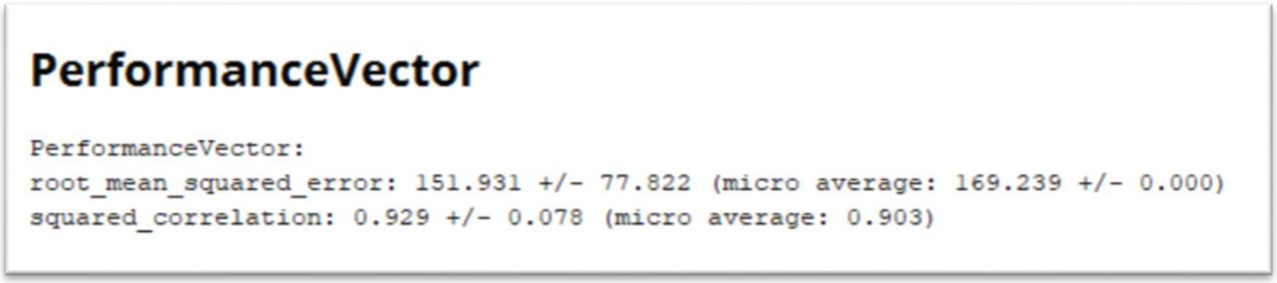


Figure 2.2: Sales volume histogram

Next step is cleaning the data from outliers. As the histogram shows (Figure 2.2) there are some outliers in the dataset and they need to be removed so that the model works as intended. After the inspection of the data and testing the model accuracy upon removing the outliers, the best results were obtained with removal of four observations from the dataset.

3. Model

After testing, the best fitted model for this dataset was the model built with the gradient boosted trees algorithm (Figure 3.1). The settings for the algorithm included the number of the trees of 20 and the learning rate of 0.2. Other parameters were not changed.



PerformanceVector

```
PerformanceVector:
root_mean_squared_error: 151.931 +/- 77.822 (micro average: 169.239 +/- 0.000)
squared_correlation: 0.929 +/- 0.078 (micro average: 0.903)
```

Figure 3.1: Performance metrics of the model (GBT algorithm)

The performance metrics of the model shows that it is highly accurate model and it will be a good predictor for the new attributes dataset.

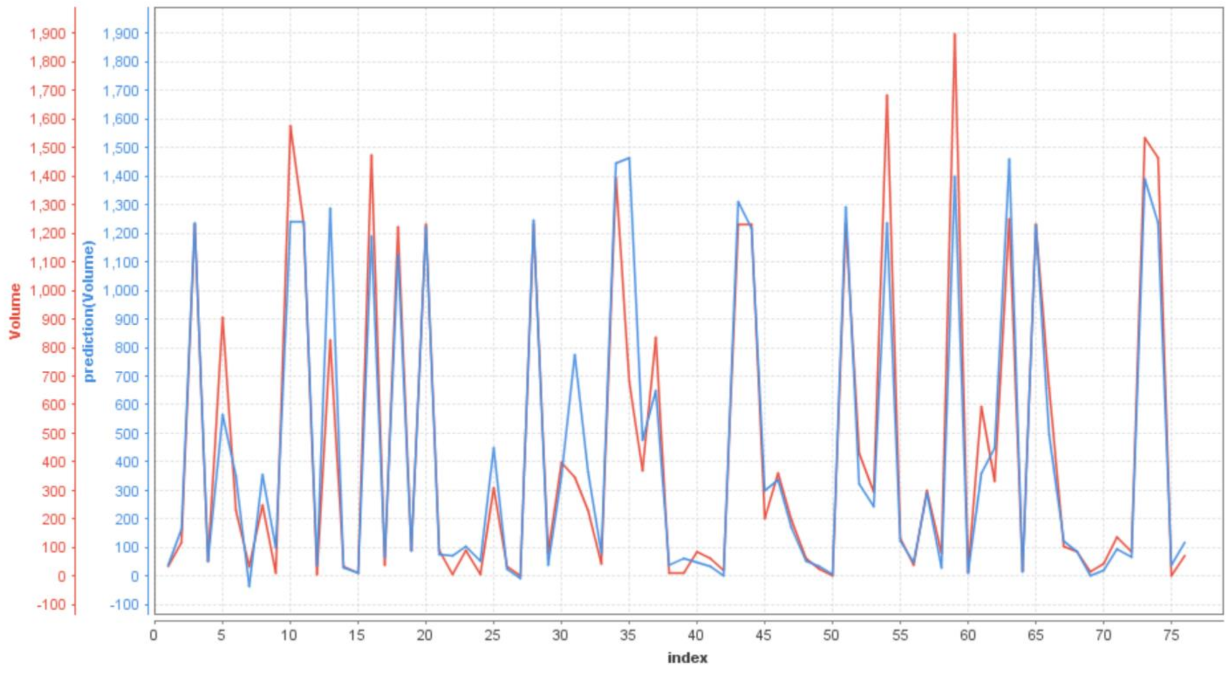
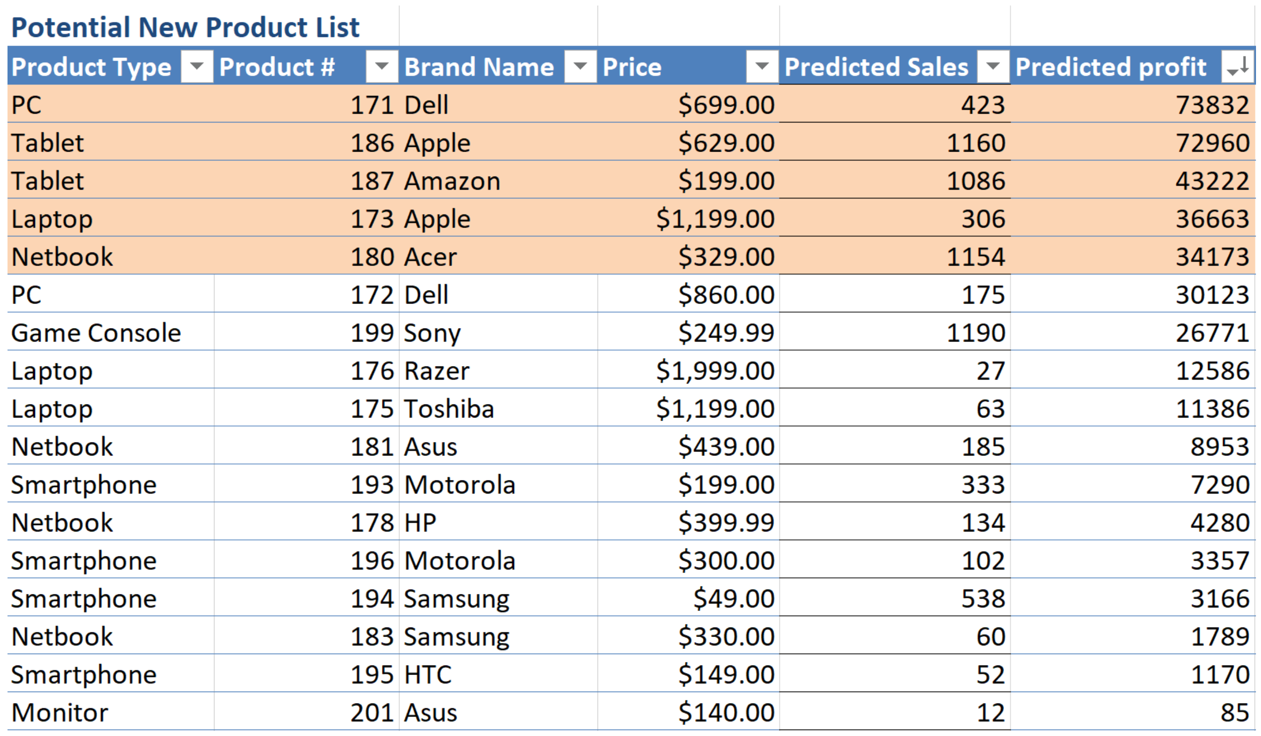


Figure 3.2: Sales volume attribute and predicted sales volume attribute

4. Sales prediction

After applying the model to the new products dataset, the predicted sales volume data is obtained. Then the predicted sales data is used to obtain the predicted profit data using this formula: sales volume x profit margin x price.

First 5 products from the new product dataset are listed below in the table and highlighted (Figure 4.1). Most profitable product is product number 171 (Dell PC) with the listed price of $699 and predicted sales volume of 423 and predicted profit of $73832. Second is the product 186 (Apple tablet) with the listed price of $629 and predicted sales volume of 1160 and predicted profit of $72960. Third product is product number 187 (Amazon tablet) with the listed price of $199 and predicted sales volume of 1086 and predicted profit of $43222. Fourth product is product number 173 (Apple laptop) with listed price of $1,199 and predicted sales volume of 306 and predicted profit of $36663. Last of the top five products is product number 180 (Acer netbook) with the listed price of $329 and predicted sales volume of 1154 and predicted profit of $34173.



Potential New Product List

Product Type	Product #	Brand Name	Price	Predicted Sales	Predicted profit
PC	171	Dell	$699.00	423	73832
Tablet	186	Apple	$629.00	1160	72960
Tablet	187	Amazon	$199.00	1086	43222
Laptop	173	Apple	$1,199.00	306	36663
Netbook	180	Acer	$329.00	1154	34173
PC	172	Dell	$860.00	175	30123
Game Console	199	Sony	$249.99	1190	26771
Laptop	176	Razer	$1,999.00	27	12586
Laptop	175	Toshiba	$1,199.00	63	11386
Netbook	181	Asus	$439.00	185	8953
Smartphone	193	Motorola	$199.00	333	7290
Netbook	178	HP	$399.99	134	4280
Smartphone	196	Motorola	$300.00	102	3357
Smartphone	194	Samsung	$49.00	538	3166
Netbook	183	Samsung	$330.00	60	1789
Smartphone	195	HTC	$149.00	52	1170
Monitor	201	Asus	$140.00	12	85

Figure 4.1: Profitability table of new products

APPENDIX A

1. Linear regression

1.1. Without filtering out the outliers:

```
Performance vector:
root_mean_squared_error: 651.756 +/- 354.664 (micro average: 742.006 +/- 0.000)
squared_correlation: 0.766 +/- 0.253 (micro average: 0.758)
```

1.2. With filtering out the outliers: volume > 3000 (2 outliers):

```
Performance vector:
root_mean_squared_error: 306.034 +/- 166.859 (micro average: 343.795 +/- 0.000)
squared_correlation: 0.802 +/- 0.162 (micro average: 0.719)
```

1.3. With filtering out the outliers: volume > 2060 (3 outliers):

```
Performance vector:
root_mean_squared_error: 284.699 +/- 155.249 (micro average: 326.010 +/- 0.000)
squared_correlation: 0.740 +/- 0.217 (micro average: 0.701)
```

1.4. With filtering out the outliers: volume > 2000 (4 outliers):

```
Performance vector:
root_mean_squared_error: 304.591 +/- 158.658 (micro average: 339.421 +/- 0.000)
squared_correlation: 0.823 +/- 0.153 (micro average: 0.670)
```

1.5. With filtering out the outliers: volume > 1800 (5 outliers):

```
Performance vector:
root_mean_squared_error: 359.343 +/- 325.302 (micro average: 474.672 +/- 0.000)
squared_correlation: 0.687 +/- 0.290 (micro average: 0.372)
```

2. KNN regression

The K-Nearest Neighbor regression model adjusting was done by adjusting the parameter k (number of neighbors when measuring distance) from K=1 to K=6.

2.1. K=1

```
Performance vector:
root_mean_squared_error: 1297.115 +/- 1042.289 (micro average: 1663.993 +/- 0.000)
squared_correlation: 0.000 +/- 0.000 (micro average: 0.000)
```

after filtering the outliers:

```
Performance vector:
root_mean_squared_error: 695.323 +/- 119.417 (micro average: 704.167 +/- 0.000)
squared_correlation: 0.000 +/- 0.000 (micro average: 0.000)
```

2.2. K=2

```
Performance vector:
root_mean_squared_error: 900.612 +/- 978.248 (micro average: 1329.688 +/- 0.000)
squared_correlation: 0.780 +/- 0.166 (micro average: 0.241)
```

after filtering the outliers:

```
Performance vector:
root_mean_squared_error: 269.695 +/- 129.956 (micro average: 302.267 +/- 0.000)
squared_correlation: 0.761 +/- 0.183 (micro average: 0.707)
```

2.3. K=3

```
Performance vector:
root_mean_squared_error: 871.630 +/- 946.794 (micro average: 1286.918 +/- 0.000)
squared_correlation: 0.798 +/- 0.155 (micro average: 0.294)
```

after filtering the outliers:

```
Performance vector:
root_mean_squared_error: 286.835 +/- 120.042 (micro average: 312.134 +/- 0.000)
squared_correlation: 0.750 +/- 0.164 (micro average: 0.697)
```

2.4. K=4

```
Performance vector:
root_mean_squared_error: 857.382 +/- 971.128 (micro average: 1295.451 +/- 0.000)
squared_correlation: 0.732 +/- 0.207 (micro average: 0.297)
```

after filtering the outliers:

```
Performance vector:
root_mean_squared_error: 309.041 +/- 133.691 (micro average: 336.425 +/- 0.000)
squared_correlation: 0.713 +/- 0.186 (micro average: 0.651)
```

2.5. K=5

```
Performance vector:

root_mean_squared_error: 840.738 +/- 991.487 (micro average: 1299.957 +/- 0.000)
squared_correlation: 0.718 +/- 0.213 (micro average: 0.314)
```

after filtering the outliers:

```
Performance vector:
root_mean_squared_error: 334.931 +/- 145.308 (micro average: 363.705 +/- 0.000)
squared_correlation: 0.672 +/- 0.198 (micro average: 0.587)
```

2.6. K=6

```
Performance vector:
root_mean_squared_error: 841.575 +/- 1006.312 (micro average: 1311.836 +/- 0.000)
squared_correlation: 0.687 +/- 0.236 (micro average: 0.309)
```

after filtering the outliers:

```
Performance vector:
root_mean_squared_error: 345.777 +/- 149.550 (micro average: 374.810 +/- 0.000)
squared_correlation: 0.661 +/- 0.195 (micro average: 0.566)
```

3. SVM regression

Support vector machine regression model adjusting was done by adjusting the kernel type parameter and SVM complexity constant C.

3.1. Kernel type:dot; c:0.1

Performance vector:
root_mean_squared_error: 1102.610 +/- 1020.829 (micro average: 1502.611 +/- 0.000)
squared_correlation: 0.618 +/- 0.223 (micro average: 0.011)

after filtering the outliers:
Performance vector:
root_mean_squared_error: 562.644 +/- 101.987 (micro average: 567.988 +/- 0.000)
squared_correlation: 0.667 +/- 0.124 (micro average: 0.001)

Kernel type: dot; c:10

Performance vector:
root_mean_squared_error: 788.940 +/- 849.322 (micro average: 1159.213 +/- 0.000)
squared_correlation: 0.778 +/- 0.258 (micro average: 0.709)

after filtering the outliers:

Performance vector:
root_mean_squared_error: 329.439 +/- 129.560 (micro average: 351.777 +/- 0.000)
squared_correlation: 0.800 +/- 0.184 (micro average: 0.655)

Kernel type: dot; c:100

Performance vector:
root_mean_squared_error: 860.266 +/- 687.615 (micro average: 1101.305 +/- 0.000)
squared_correlation: 0.549 +/- 0.323 (micro average: 0.519)

after filtering the outliers:
Performance vector:
root_mean_squared_error: 578.382 +/- 189.050 (micro average: 607.391 +/- 0.000)
squared_correlation: 0.361 +/- 0.232 (micro average: 0.183)

Kernel type: dot; c:20

```
Performance vector:
root_mean_squared_error: 697.180 +/- 777.467 (micro average: 1044.277 +/- 0.000)
squared_correlation: 0.789 +/- 0.284 (micro average: 0.712)
```

after filtering the outliers:

```
Performance vector:
root_mean_squared_error: 284.906 +/- 151.582 (micro average: 321.601 +/- 0.000)
squared_correlation: 0.784 +/- 0.195 (micro average: 0.671)
```

Kernel type: dot; c:30

```
Performance vector:
root_mean_squared_error: 681.590 +/- 758.843 (micro average: 1020.004 +/- 0.000)
squared_correlation: 0.778 +/- 0.283 (micro average: 0.706)
```

after filtering the outliers:

```
Performance vector:
root_mean_squared_error: 305.916 +/- 140.935 (micro average: 335.981 +/- 0.000)
squared_correlation: 0.753 +/- 0.201 (micro average: 0.636)
```

Kernel type: dot; c:35

```
Performance vector:
root_mean_squared_error: 689.070 +/- 744.476 (micro average: 1014.427 +/- 0.000)
squared_correlation: 0.765 +/- 0.280 (micro average: 0.693)
```

after filtering the outliers:

```
Performance vector:
root_mean_squared_error: 315.278 +/- 141.436 (micro average: 344.715 +/- 0.000)
squared_correlation: 0.738 +/- 0.204 (micro average: 0.616)
```

3.2. kernel type: radial; c=0.1

```
Performance vector:
root_mean_squared_error: 1108.621 +/- 1023.861 (micro average: 1509.083 +/- 0.000)
squared_correlation: 0.189 +/- 0.153 (micro average: 0.005)
```

3.3. kernel type: polynominal; c=0.1

```
Performance vector:
root_mean_squared_error: 1104.280 +/- 1023.714 (micro average: 1505.797 +/- 0.000)
squared_correlation: 0.565 +/- 0.275 (micro average: 0.008)
```

3.4. kernel type: neural; c=0.1

```
Performance vector:
root_mean_squared_error: 1105.160 +/- 1017.538 (micro average: 1502.252 +/- 0.000)
squared_correlation: 0.550 +/- 0.292 (micro average: 0.008)
```

3.5. kernel type: anova; c=0.1

```
Performance vector:
root_mean_squared_error: 1108.034 +/- 1023.674 (micro average: 1508.525 +/- 0.000)
squared_correlation: 0.427 +/- 0.166 (micro average: 0.006)
```

3.6. kernel type: epachnenikov; c=0.1

```
Performance vector:
root_mean_squared_error: 1108.816 +/- 1023.830 (micro average: 1509.205 +/- 0.000)
squared_correlation: 0.143 +/- 0.148 (micro average: 0.005)
```

3.7. kernel type: multiquadric; c=0.1

```
Performance vector:
root_mean_squared_error: 1133.882 +/- 1010.801 (micro average: 1519.015 +/- 0.000)
squared_correlation: 0.000 +/- 0.000 (micro average: 0.000)
```

4. Gradient boosted trees regression

Gradient boosted trees regression model adjusting was done by adjusting the learning rate parameter and number of trees parameter.

4.1. Learning rate:0.1; trees:20

Performance vector:
root_mean_squared_error: 165.056 +/- 72.238 (micro average: 178.689 +/- 0.000)
squared_correlation: 0.927 +/- 0.076 (micro average: 0.902)

learning rate:0.2 trees:20 (Selected parameters for the model)
Performance vector:
root_mean_squared_error: 151.931 +/- 77.822 (micro average: 169.239 +/- 0.000)
squared_correlation: 0.929 +/- 0.078 (micro average: 0.903)

learning rate:0.2; trees:30
Performance vector:
root_mean_squared_error: 157.928 +/- 86.978 (micro average: 178.851 +/- 0.000)
squared_correlation: 0.921 +/- 0.085 (micro average: 0.893)

learning rate:0.2; trees:50
Performance vector:
root_mean_squared_error: 165.442 +/- 96.391 (micro average: 189.514 +/- 0.000)
squared_correlation: 0.916 +/- 0.091 (micro average: 0.881)

learning rate: 0.2; trees:80
Performance vector:
root_mean_squared_error: 175.716 +/- 102.040 (micro average: 200.600 +/- 0.000)
squared_correlation: 0.911 +/- 0.099 (micro average: 0.869)

4.2. learning rate:0.25; trees:20
Performance vector:
root_mean_squared_error: 152.849 +/- 87.397 (micro average: 174.642 +/- 0.000)
squared_correlation: 0.925 +/- 0.083 (micro average: 0.897)

4.3. learning rate:0.3; trees:20
Performance vector:
```
root_mean_squared_error: 162.323 +/- 88.563 (micro average: 183.734 +/- 0.000)
squared_correlation: 0.912 +/- 0.091 (micro average: 0.887)
```

4.4. learning rate:0.5; trees:20

Performance vector:
```
root_mean_squared_error: 179.079 +/- 92.810 (micro average: 199.966 +/- 0.000)
squared_correlation: 0.907 +/- 0.092 (micro average: 0.869)
```

4.5. Learning rate:0.01; trees:20

after filter:
Performance vector:
```
root_mean_squared_error: 453.377 +/- 81.022 (micro average: 459.178 +/- 0.000)
squared_correlation: 0.892 +/- 0.119 (micro average: 0.853)
```