Steam Players Country-wise Comparison

This project focuses on taking in the Steam Country Dataset, that include the Countries, its no. of Players, and the Points or XP gained overall by that country till date. Then, the data would be cleaned and three regression models would be applied to predict the number of points if a player plays for 'x' no. of hours. The basic results show the Linear, L1 and L2 regression outputs, and also show the best data frame to use after several changes to the main dataset.

	Rank	Country	Players	Points	Number	Games	Badges	XP	Hours
0	0	Unknown	135643	86532124.75	137664177	31410891	5453964	2252121996	412471186.4
1	1	United States	59698	52132889.98	84410455	20379201	2569218	974895706	236137390.5
2	2	Russian Federation	31058	24753943.31	37279292	7247856	1657378	693424533	104656082.0
3	3	Germany	21750	18214362.15	28423805	6658939	1381807	590848431	86523185.4
4	4	United Kingdom (Great Britain)	16225	15635912.83	25101789	5780703	786975	294574732	68131144.6
5	5	Canada	12483	10999031.70	17436104	4012880	606502	239174494	49354715.3
6	6	China	15605	10847234.66	17138407	6168652	1100485	506417923	54823363.8
7	7	Brazil	15569	10136859.28	16173137	3918546	774781	247132541	54157619.5
8	8	Poland	11358	9085626.49	13802221	3123884	523117	178278957	39163756.6
9	9	Japan	9548	8802348.47	13143190	3328328	698257	265270862	38756414.3

Looking at the original dataset, it was observed that the data is very large, and it may or may not affect the overall model or algorithm.

	Rank	Players	Points	Number	Games	Badges	XP	Hours
count	253.000000	253.000000	2.530000e+02	2.530000e+02	2.530000e+02	2.530000e+02	2.530000e+02	2.530000e+02
mean	126.000000	1836.355731	1.389446e+06	2.187685e+06	5.158577e+05	9.156419e+04	3.623566e+07	6.476821e+06
std	73.179004	9777.432248	6.817831e+06	1.084677e+07	2.517633e+06	4.171119e+05	1.700937e+08	3.185529e+07
min	0.000000	1.000000	5.400000e-01	1.000000e+00	3.000000e+00	2.000000e+00	1.560000e+02	1.390000e+01
25%	63.000000	41.000000	2.191154e+04	3.788700e+04	7.069000e+03	1.427000e+03	4.423540e+05	1.314644e+05
50%	126.000000	102.000000	5.478048e+04	9.217000e+04	2.060400e+04	4.260000e+03	1.762010e+06	3.501921e+05
75%	189.000000	502.000000	2.908862e+05	4.827490e+05	1.040740e+05	2.629900e+04	1.021018e+07	1.667536e+06
max	252.000000	135643.000000	8.653212e+07	1.376642e+08	3.141089e+07	5.453964e+06	2.252122e+09	4.124712e+08

The describe function shows how the large the data is clearly. If we look at the mean of the values in any column, say ‘XP’, it is in the format ‘number’ * e^{07}. This necessarily doesn’t mean that it would ruin the performance of the model, but creating another dataset with cleaner values would show the dominance of the correct dataset, and clear us of the same doubt.

Over the next few steps, the data would be compressed and would be refined to the values at which the Linear model can be implemented.

```
Rank        0
Country     0
Players     0
Points      0
Number      0
Games       0
Badges      0
XP          0
Hours       0
dtype: int64
```

We have no null values in the dataset. But we have a row called as that has the value of ‘Unknown’ in the Country column (the first row of the dataset). It is unnecessary, and removing it would be beneficial. Also, the name United Kingdom (Great Britain) is a big name, that isn't required (row no. 5). Simply converting it to United Kingdom makes the value short and precise. The column – ‘Number’, is also unnecessary and is of no importance. So, that would also be removed.

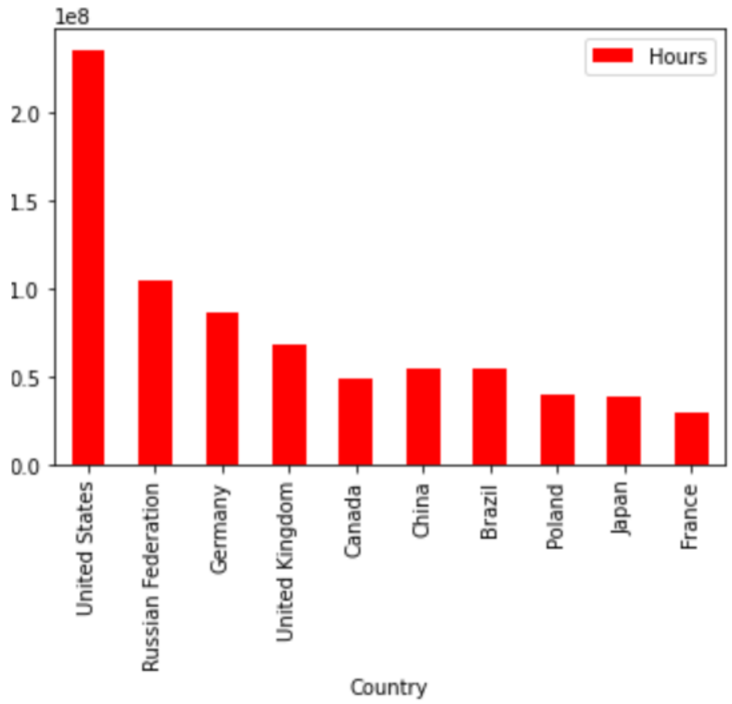
This is how the dataset looks after performing all the above steps.

	Rank	Country	Players	Points	Games	Badges	XP	Hours
1	1	United States	59698	52132889.98	20379201	2569218	974895706	236137390.5
2	2	Russian Federation	31058	24753943.31	7247856	1657378	693424533	104656082.0
3	3	Germany	21750	18214362.15	6658939	1381807	590848431	86523185.4
4	4	United Kingdom	16225	15635912.83	5780703	786975	294574732	68131144.6
5	5	Canada	12483	10999031.70	4012880	606502	239174494	49354715.3

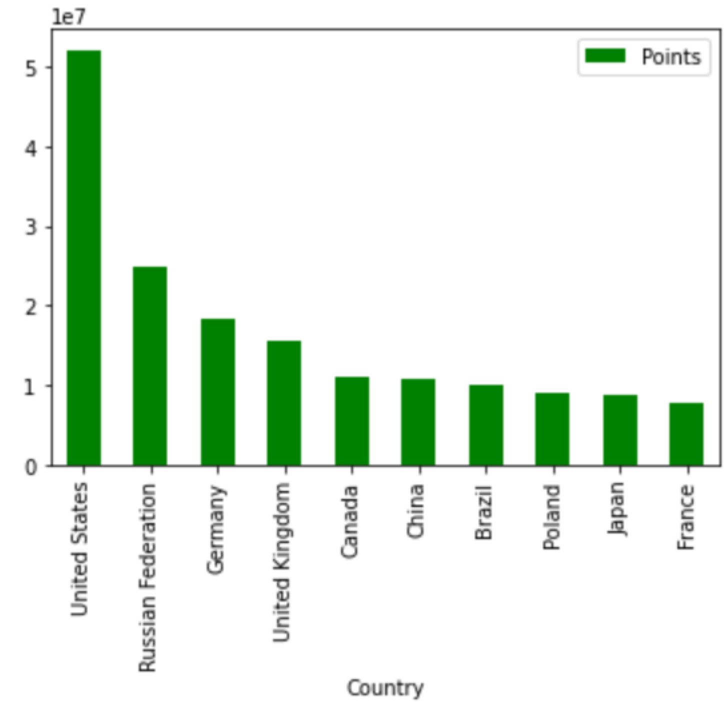
Now, due to a lot of large and different values, it is difficult to interpret how they have ranked the countries and on the basis of what column they say this country is at this position. So, a top 10 dataframe will be created, and it will be used to create different comparative bar graphs. If we find a constant decrease after each country and the value for which we are checking, it would confirm the same as the main factor for ranking the countries.

	Rank	Country	Players	Points	Games	Badges	XP	Hours
1	1	United States	59698	52132889.98	20379201	2569218	974895706	236137390.5
2	2	Russian Federation	31058	24753943.31	7247856	1657378	693424533	104656082.0
3	3	Germany	21750	18214362.15	6658939	1381807	590848431	86523185.4
4	4	United Kingdom	16225	15635912.83	5780703	786975	294574732	68131144.6
5	5	Canada	12483	10999031.70	4012880	606502	239174494	49354715.3
6	6	China	15605	10847234.66	6168652	1100485	506417923	54823363.8
7	7	Brazil	15569	10136859.28	3918546	774781	247132541	54157619.5
8	8	Poland	11358	9085626.49	3123884	523117	178278957	39163756.6
9	9	Japan	9548	8802348.47	3328328	698257	265270862	38756414.3
10	10	France	7675	7817051.60	2362871	384431	130107826	29692105.0

This is the top 10 dataframe. Now, we will create a bar graph comparison between the countries and the other columns.



Here, for example, when we used ‘Hours’ in the comparison, we saw that it decreases constantly till Canada, but increases again after that. So, it is not the main factor of ranking.



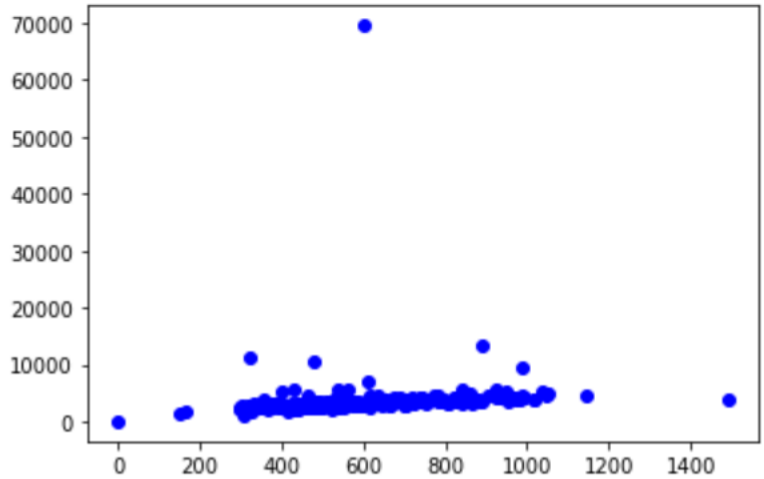
After comparing with all the columns, only the points column shows a constant decrease. So, we can use the 'Points' column in the model as decided.

As discussed previously, we will be creating a dataset with reduced values and better interpretability. To do this, we will create an average dataframe, that takes the values average per player in a country. This can be achieved by dividing each column with the 'Players' column.

	Rank	Country	Players	Points_avg	Games_avg	Badges_avg	XP_avg	Hours_avg
1	1	United States	59698	873.276994	341.371587	43.036919	16330.458407	3955.532690
2	2	Russian Federation	31058	797.023096	233.365188	53.363964	22326.760674	3369.698049
3	3	Germany	21750	837.441938	306.158115	63.531356	27165.445103	3978.077490
4	4	United Kingdom	16225	963.692624	356.283698	48.503852	18155.607519	4199.146046
5	5	Canada	12483	881.120860	321.467596	48.586237	19160.017143	3953.754330
...	...	...	...	...	...	...	...	...
248	248	Mauritania	10	323.150000	78.800000	14.400000	4329.200000	2457.440000
249	249	French Guiana	7	369.667143	157.142857	11.571429	2387.000000	2952.142857
250	250	Anguilla	14	149.268571	44.714286	8.285714	1530.500000	1297.800000
251	251	Moldova, Republic of	2	308.510000	64.500000	58.000000	8743.000000	969.950000
252	252	Yugoslavia	1	0.540000	3.000000	2.000000	156.000000	13.900000

Now, for the final step in this dataframe, we would remove the Rank, Country and Players column as they are of no use after.

	Points_avg	Games_avg	Badges_avg	XP_avg	Hours_avg
1	873.276994	341.371587	43.036919	16330.458407	3955.532690
2	797.023096	233.365188	53.363964	22326.760674	3369.698049
3	837.441938	306.158115	63.531356	27165.445103	3978.077490
4	963.692624	356.283698	48.503852	18155.607519	4199.146046
5	881.120860	321.467596	48.586237	19160.017143	3953.754330
...	...	...	...	...	...
248	323.150000	78.800000	14.400000	4329.200000	2457.440000
249	369.667143	157.142857	11.571429	2387.000000	2952.142857
250	149.268571	44.714286	8.285714	1530.500000	1297.800000
251	308.510000	64.500000	58.000000	8743.000000	969.950000
252	0.540000	3.000000	2.000000	156.000000	13.900000



After looking at the graph, it appears to be very compressed due to some outliers. So, we will remove the outliers to make the dataframe usable and efficient for a model.

```
outliers = []
location = []
for i in range (1, player_avg_dataframe.shape[0]):
    if X[i] > 7000:
        outliers.append(X[i])
        location.append(i)

outliers
location
```

```
[115, 140, 167, 236, 242]
```

We took the values above 7000, and used the append() function to add them to a list named 'outliers', and simultaneously append the locations of the outliers in the dataframe.

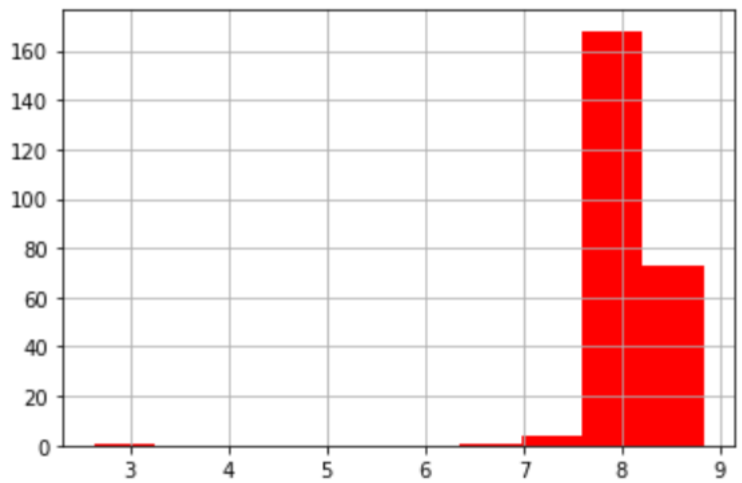
Here, we can conclude that the elements at location 115, 140, 167, 236 and 242 would be considered as outliers. So, we will remove them from the dataframe.

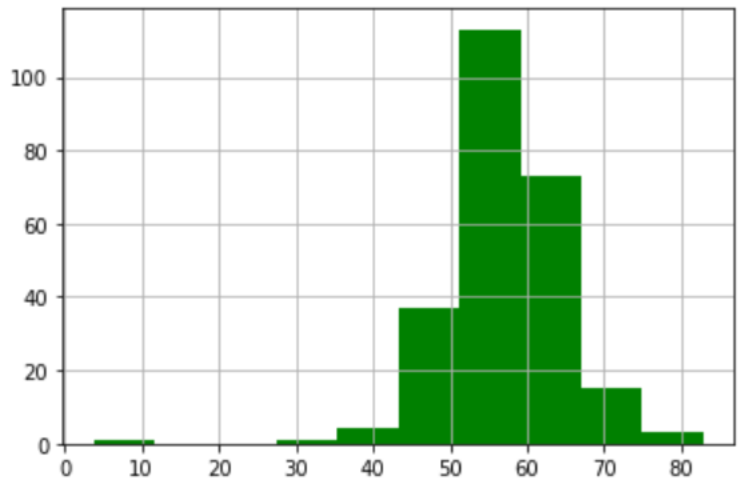
Now, we need to normalize the data with any of the three methods, which are

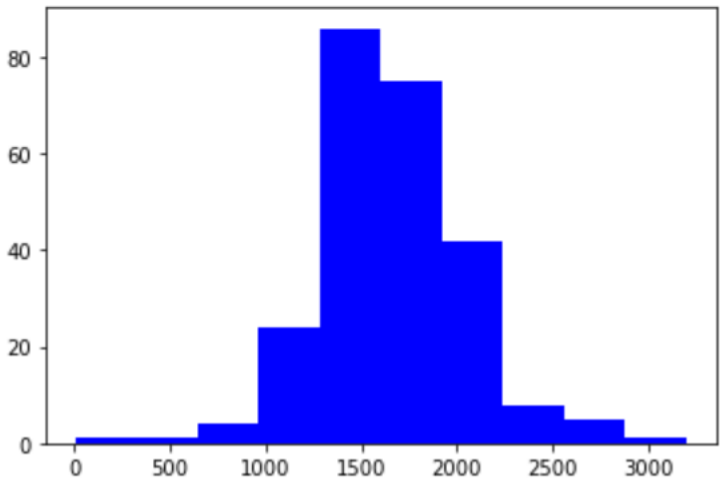
1. Square Root Transformation

2. Log Transformation

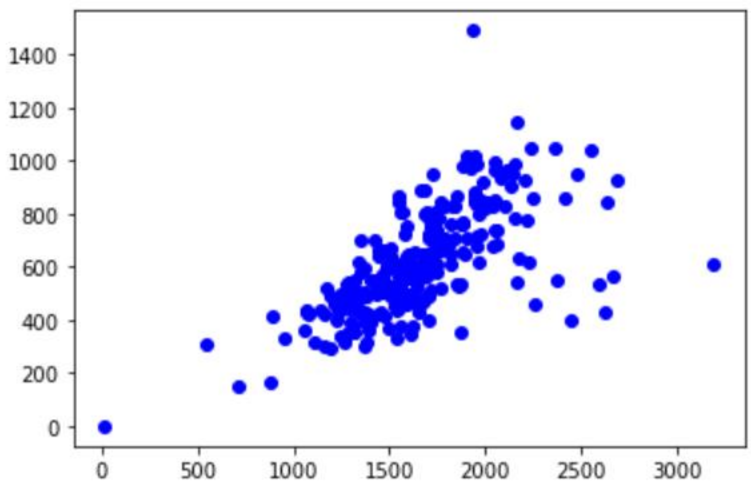
3. Boxcox Method





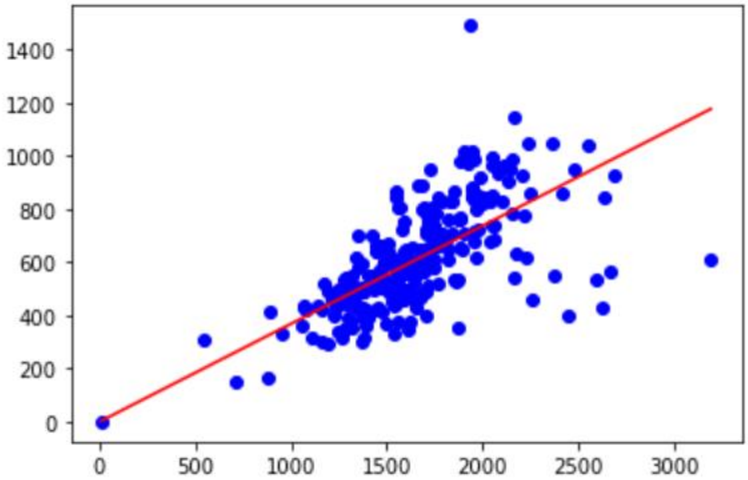


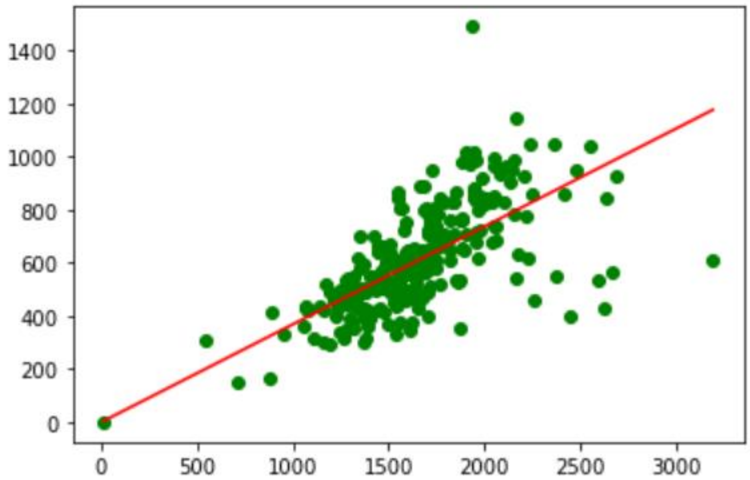
The normaltest for Boxcox method proved out to be having the closest p-value to 0.05. We will be using the boxcox normalized values for the regression models.

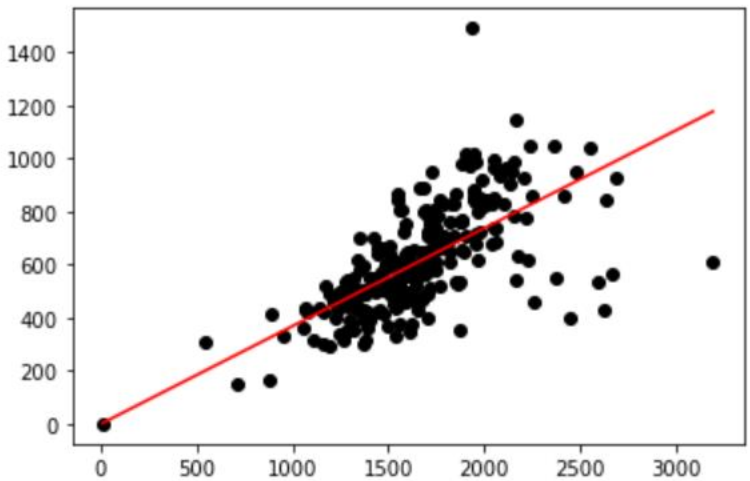


This is the dataset in a scatterplot after doing all the changes.

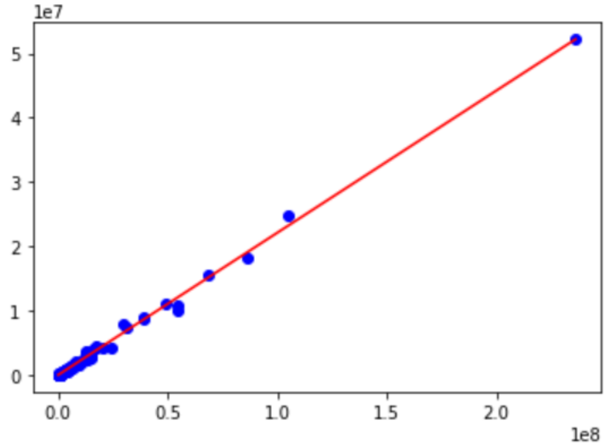
The regression models (Linear regression, L1 and L2) are applied to the dataset.

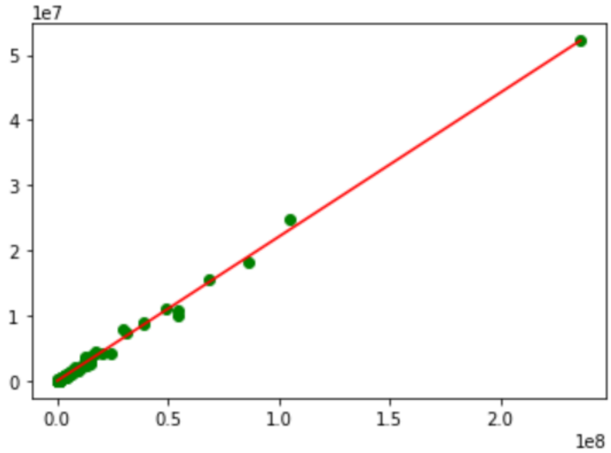


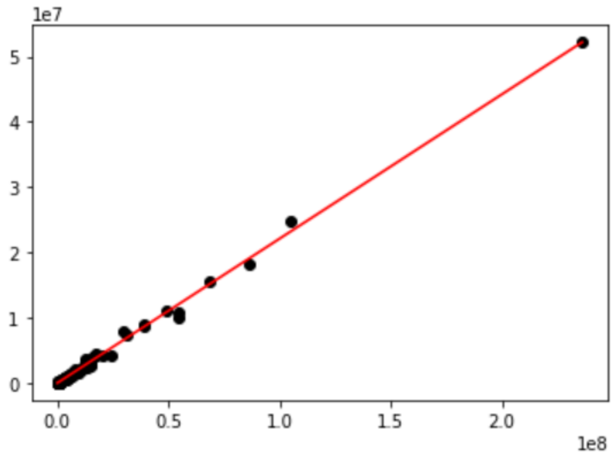




All three Regression techniques give the same values and plots. The r-squared value came out to be around 0.44. Now, we will take the original dataframe and repeat the same steps for regression.







Still the same in all three regressions, but a better visualization of the linear data. This dataset seems to be highly linear in nature, and could be used to predict any value easily. The r2 score of 0.99> shows how less the errors are in the overall dataset, which were not that visible after taking the average of the dataframe values.

Hence, the regression models were successfully applied and visualized.