Video Game Sales Prediction

DECEMBER 3

CS - 559
Authored by: Siddhant Barua

Introduction

Objective: Use periodic sales data of video games in distinct regions and other factors, to predict global sales.

Video game sales depend on a multitude of factors and intuitively the most important factor being "is the game good?", the answer to this question is subjective and hence models cannot be built around it. However, we can predict how well a game does based on how it performs in distinct regions.

The performance of video games in terms of their sales vary between distinct regions and this is due to various factors namely – population of the region, whether the game has been released in a region (contingent on government approval), critics score, copyright laws among many others. This leads to the sales data being non standardized, to tackle this we consider the sales only in regions like "North America (N.A)" , "Europe (E.U)", "Japan (Jpn)", and the sales average of that particular game for all other regions are taken and grouped under "Other Region Sales (O.R.S)".

The following factors are taken into consideration while predicting global sales of the video-game : Average number of concurrent users of the game, User rating of the game, Critics Score of that game, Number of critics and finally the Region Sale based on which global sales is to be calculated (Either N.A or E.U or Jpn or O.R.S).

Scope: This project uses a Kaggle dataset which has ([16,719 rows and 16 columns]).
Since we are predicting values i.e. values in target column are continuous in nature, we use regression models for this project.
We then compute the Mean Absolute Error (M.A.E) and hence compare the performance of the algorithms, while also considering the time taken and regression performance.

The approach

Cleaning the Data

We first obtain the dataset , we then analyze the dataset and as depicted in the image below, there are multiple fields in the dataset that are missing values.

2882	Madden N	PSP	2007	Sports	Electronic	0.6	0.04	0	0.07	0.71	75	10	6.8	8	EA Tiburor	E
2883	Blast Corp	N64	1997	Action	Nintendo	0.39	0.09	0.17	0.06	0.71						
2884	LEGO Race	N64	1999	Racing	LEGO Med	0.51	0.18	0	0.01	0.71						
2885	WWF Attit	N64	1999	Fighting	Acclaim Er	0.57	0.13	0	0.01	0.71						
2886	Tactics Og	SNES	1995	Role-Playi	Quest	0	0	0.71	0	0.71						
2887	Madden N	PSP	2008	Sports	Electronic	0.65	0	0	0.06	0.71	68	8	6.5	4	EA Tiburor	E
2888	WarioWar	GC	2003	Puzzle	Nintendo	0.2	0.05	0.44	0.02	0.71						
2889	NBA Live 0	X360	2008	Sports	Electronic	0.5	0.14	0	0.07	0.71	77	32	7.4	14	EA Canada	E
2890	Battle Are	PS	1995	Fighting	Sony Com	0.15	0.1	0.41	0.05	0.71						
2891	Dragon Ba	PS4	2016	Action	Namco Ba	0.24	0.27	0.09	0.1	0.71	73	54	7.8	117	Dimps Cor	T
2892	Dynasty W	PS3	2007	Action	Tecmo Ko	0.18	0.07	0.41	0.04	0.71	59	31	7.3	49	Koei, Ome	T
2893	Jeopardy!	PS	1997	Misc	Hasbro Int	0.39	0.27	0	0.05	0.71						
2894	South Park	PS	1998	Shooter	Acclaim Er	0.39	0.27	0	0.05	0.71						
2895	NCAA Foo	X360	2008	Sports	Electronic	0.65	0	0	0.05	0.71	83	24	7.7	26	EA Tiburor	E
2896	Doom 3: R	XB	2005	Shooter	Activision	0.53	0.15	0	0.03	0.71	77	41	7.9	16	id Softwar	M
2897	SSX	X360	2012	Sports	Electronic	0.38	0.26	0	0.06	0.7	82	65	6.4	186	EA Canada	E

We then undergo the process of cleaning the dataset. As we can see from the image above, there are a lot of missing fields in this dataset. A few key features like Year of Release, Publishers, Critic Score, Critic Count, User Score and User count, that are missing values and they are depicted in the image below.

```
main (5) ×    main (6) ×    main (7) ×
>>> runfile('C:/Users/barua/Desktop/Machine Learning/Final_
Name                  2
Year_of_Release     269
Genre                 2
Publisher            54
Critic_Score       8582
Critic_Count       8582
User_Score         6704
User_Count         9129
Rating             6769
dtype: int64
```

Since there are only two fields in Name and Genre, we manually remove them from the dataset. Year_of_Release has many fields which have vale “N/A”, we first replace them with 0’s and then later replace them with the median value of the Platform they belong to. For instance,

1302	Midway Ar	PS2	2003	Misc	Midway Ga	0.72	0.56	0	0.19	1.46	76	22	tbd		Midway	T
1303	Triple Play	PS	N/A	Sports	N/A	0.81	0.55	0	0.1	1.46						
1304	Dragon Qu	3DS	2013	Role-Playi	Square Eni	0.06	0.09	1.3	0.01	1.46						
1305	Super Mor	GC	2001	Puzzle	Atari	0.95	0.37	0.1	0.04	1.46	87	28	7.9	46	Amusemer	E
1306	SoulCalibu	PS3	2008	Fighting	Ubisoft	0.72	0.4	0.14	0.2	1.46	85	65	7.9	129	Namco	T

The value highlighted with yellow is changed to 0 initially and then later replaced by the median of Year_of_release for all games using the Platform “PS”.

Publisher has multiple fields missing as well but we replace “N/A” values with “unknown” as it is impossible to fill this up with dummy data.

Critic Score and Critic Count, the “N/A” fields are replaced with the median Critic_Scores and Critic_Counts, this is possible because there are not many “N/A” fields in these columns to begin with, hence replacing them with 0’s and then finding the mean is not necessary.

Now the “N/A” values in User_Count is also replaced with the median value of User_Count. But this is not possible for the User_Score, as there are many ‘tbd’ fields in conjunction with “N/A” fields. What we do to clean this data is first, we replace all N/A values with “0” and we replace all “tbd” values with 100 hence not affecting the median too much i.e. ensures that median remains almost the same.

Experiments stated later in the report justify the assertion that accuracy deteriorates if all “N/A” and “tbd” values are replaced with either just 0’s or just 100’s. After this we replace all User_Scores having values 0 and 100 with the median of that column (User_Score) in the dataset.

We for the sake of simplicity only consider top 10 publishers as they own majority of the games in the dataset and then the other publishers are tagged as ‘Other_Dev’.

We then convert all relevant features into integer type for simplicity sake.

We realize that there are some games that release much later than the release date of the platform, and the age of the game is an important factor in depicting global sales, We hence create a new column which we merge to our original Dataset called Age given by the game’s Year_of_release – release date of the console.

For instance,

Name	Platform	Year_of_R	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sal	Global_Sal	Critic_Scoı	Critic_Cou	User_Scor	User_Cour	Developer	Rating
Wii Sports	Wii	2006	Sports	Nintendo	41.36	28.96	3.77	8.45	82.53	76	51	8	322	Nintendo	E
Super Mar	NES	1985	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24						
Mario Kart	Wii	2008	Racing	Nintendo	15.68	12.76	3.79	3.29	35.52	82	73	8.3	709	Nintendo	E
Wii Sports	Wii	2009	Sports	Nintendo	15.61	10.93	3.28	2.95	32.77	80	73	8	192	Nintendo	E
Pokemon I	GB	1996	Role-Playi	Nintendo	11.27	8.89	10.22	1	31.37						
Tetris	GB	1989	Puzzle	Nintendo	23.2	2.26	4.22	0.58	30.26						
New Supei	DS	2006	Platform	Nintendo	11.28	9.14	6.5	2.88	29.8	89	65	8.5	431	Nintendo	E
Wii Play	Wii	2006	Misc	Nintendo	13.96	9.18	2.93	2.84	28.92	58	41	6.6	129	Nintendo	E
New Supei	Wii	2009	Platform	Nintendo	14.44	6.94	4.7	2.24	28.32	87	80	8.4	594	Nintendo	E
Duck Hunt	NES	1984	Shooter	Nintendo	26.93	0.63	0.28	0.47	28.31						
Nintendog	DS	2005	Simulation	Nintendo	9.05	10.95	1.93	2.74	24.67						

The fields highlighted in green, Pokemon released in 1996 and the Gameboy released in 1989. The age of the game is 1996 -1989 Is 7.

This is a more parameter to predict global sales.

Upon doing this there are multiple fields that have value <0

```
                                      Name  ... Age
1340                     Disney's DuckTales  ...  -1
2076                         NFL Fever 2002  ...  -1
12301  ESPN Winter X-Games: Snowboarding 2002  ...  -1
15959     Strongest Tokyo University Shogi DS  ... -19

[4 rows x 22 columns]
```

We need to remove the games that are highly negative as they will be an outlier that will affect prediction accuracy. So the field with Age -19 is removed and the fields with Age -1 are replaced with 0, indicating the game released with the Platform concurrently.

We then use "sklearn.preprocessing.StandardScaler" in order to standardize the necessary fields. Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data (e.g. Gaussian with 0 mean and unit variance).

We then drop all unnecessary Columns and preserve only the following columns, Critic_Score, Critic_Count, User_Score, User_Count, Age, NA_Sales_Log.

The NA_Sales_log gives the best prediction accuracy, however EU_Sales_log, Jpn_Sales_log and Other_sales_log can be used instead of NA_Sales_Log for the sake of experimentation.

We now finally have a clean dataset.

We now randomize the dataset, and then split it into X,y training set and testing set with X having columns Critic_Score_x ,Critic_Count_x ,User_Score_x, User_Count_x, Age_x and EU_Sales_Log while y is the target column ‘Global_Sales’

We now apply our Machine learning Models.

We perform N runs of randomizing dataset and splitting into training and test data, and take the average of MSE (Mean Squared Error) , MAE (Mean Absolute Error) and time taken for N runs of all Machine Learning Models.

We then compare the performance of MSE (Mean Squared Error), MAE (Mean Absolute Error) and time taken.

Experimental Design

Software & libraries used:

- *Software*: python
- *Libraries*: NumPy, pandas, seaborn, matplotlib, time, statistics, sklearn.preprocessing, sklearn.model_selection, sklearn.linear_model, sklearn.metrics, sklearn.ensemble, sklearn.svm and sklearn.metrics

Machine Learning Algorithms:

- Linear Regression
- Random Forest
- Boosting
- SVM (Support Vector Machine) (Uses SVR – Support Vector Regressor)

Dataset used:

- ***https://www.kaggle.com/rush4ratio/video-game-sales-with-ratings***

Experimental setting:

- The project has been tested against n runs i.e. 5,10,20 and the average of MAE, MSE and Time_Taken for all the 4 algorithms are taken and then compared to find out the model that fits best.
- The Training and Testing set are in the order 0.77 and 0.33 respectively.
- For SVR the program has been tested against all kernels i.e. *rbf, linear and polynomial*

Experimental Results

We look at the original dataset

```
Python 3.7.5 (default, Oct 31 2019, 15:18:51) [MSC v.1916 64 bit (AMD64)] on win32
>>> runfile('C:/Users/barua/Desktop/Machine Learning/Final_Project/untitled2/main.py', wdir='C:/Users/barua/Desktop/Machine Learning/Final_Project/untitled2')
                             Name Platform  Year_of_Release         Genre    Publisher  NA_Sales  EU_Sales  JP_Sales  Other_Sales  Global_Sales  Critic_Scor
0                      Wii Sports      Wii           2006.0        Sports     Nintendo     41.36     28.96      3.77         8.45         82.53          76.
1               Super Mario Bros.      NES           1985.0      Platform     Nintendo     29.08      3.58      6.81         0.77         40.24           Na
2                  Mario Kart Wii      Wii           2008.0        Racing     Nintendo     15.68     12.76      3.79         3.29         35.52          82.
3               Wii Sports Resort      Wii           2009.0        Sports     Nintendo     15.61     10.93      3.28         2.95         32.77          80.
4        Pokemon Red/Pokemon Blue       GB           1996.0  Role-Playing     Nintendo     11.27      8.89     10.22         1.00         31.37           Na
...                           ...      ...              ...           ...          ...       ...       ...       ...          ...           ...          ...
16714  Samurai Warriors: Sanada Maru      PS3         2016.0        Action   Tecmo Koei      0.00      0.00      0.01         0.00          0.01           Na
16715             LMA Manager 2007     X360           2006.0        Sports  Codemasters      0.00      0.01      0.00         0.00          0.01           Na
16716      Haitaka no Psychedelica      PSV           2016.0     Adventure Idea Factory      0.00      0.00      0.01         0.00          0.01           Na
16717             Spirits & Spells      GBA           2003.0      Platform      Wanadoo      0.01      0.00      0.00         0.00          0.01           Na
16718          Winning Post 8 2016      PSV           2016.0    Simulation   Tecmo Koei      0.00      0.00      0.01         0.00          0.01           Na

[16719 rows x 16 columns]
```

```
 Critic_Score  Critic_Count  User_Score  User_Count  Developer  Rating
         76.0          51.0           8       322.0   Nintendo       E
          NaN           NaN         NaN         NaN        NaN     NaN
         82.0          73.0         8.3       709.0   Nintendo       E
         80.0          73.0           8       192.0   Nintendo       E
          NaN           NaN         NaN         NaN        NaN     NaN
          ...           ...         ...         ...        ...     ...
          NaN           NaN         NaN         NaN        NaN     NaN
          NaN           NaN         NaN         NaN        NaN     NaN
          NaN           NaN         NaN         NaN        NaN     NaN
          NaN           NaN         NaN         NaN        NaN     NaN
          NaN           NaN         NaN         NaN        NaN     NaN
```

We now look at the Cleaned Dataset

```
       Critic_Score_x  Critic_Count_x  User_Score_x  User_Count_x     Age_x
1            1.226300        3.656261      1.512659      1.618731 -0.640064
2            1.021740        3.656261      1.242551      0.273799 -0.209759
3            0.101221       -0.193050     -0.378097     -0.163239  1.511461
4            0.101221       -0.193050     -0.378097     -0.163239 -1.500674
5            1.942259        3.064059      1.692731      0.895537 -0.640064
                  ...             ...           ...           ...       ...
16710        0.101221       -0.193050     -0.378097     -0.163239  2.802376
16711        0.101221       -0.193050     -0.378097     -0.163239 -1.070369
16712        0.101221       -0.193050     -0.378097     -0.163239  0.650851
16713        0.101221       -0.193050     -0.378097     -0.163239 -0.640064
```

Global_Sales_Log	NA_Sales_Log	EU_Sales_Log	JP_Sales_Log	Other_Sales_Log
3.719409	3.403860	1.521699	2.055405	0.570980
3.597860	2.814210	2.621766	1.566530	1.456287
3.519573	2.810005	2.479056	1.453953	1.373716
3.477232	2.507157	2.291524	2.417698	0.693147
3.442339	3.186353	1.181727	1.652497	0.457425
...	...	...	...	...
0.009950	0.009950	0.000000	0.000000	0.000000
0.009950	0.000000	0.000000	0.009950	0.000000
0.009950	0.009950	0.000000	0.000000	0.000000
0.009950	0.000000	0.000000	0.000000	0.000000

This is then split into training data and test data

Training Data

	Critic_Score_x	Critic_Count_x	User_Score_x	User_Count_x	Age_x	NA_Sales_Log
1189	-3.274015	-1.007327	-0.828277	-0.207463	-0.640064	0.727549
1356	1.021740	2.767958	1.152515	1.714983	1.511461	0.548121
4493	1.124020	0.177076	0.162119	-0.020161	1.941766	0.231112
3194	0.101221	2.545882	0.702335	-0.025364	-0.209759	0.000000
1045	-0.512458	-1.007327	-0.378097	-0.163239	-1.500674	0.737164

```
Process finished with exit code 0
```

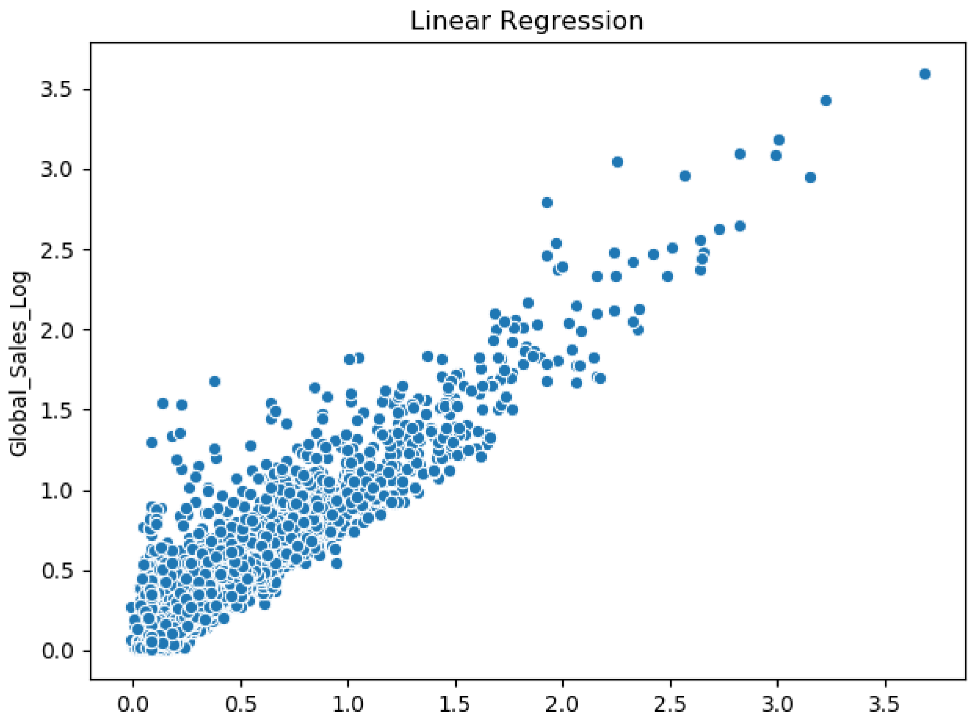
Target Data

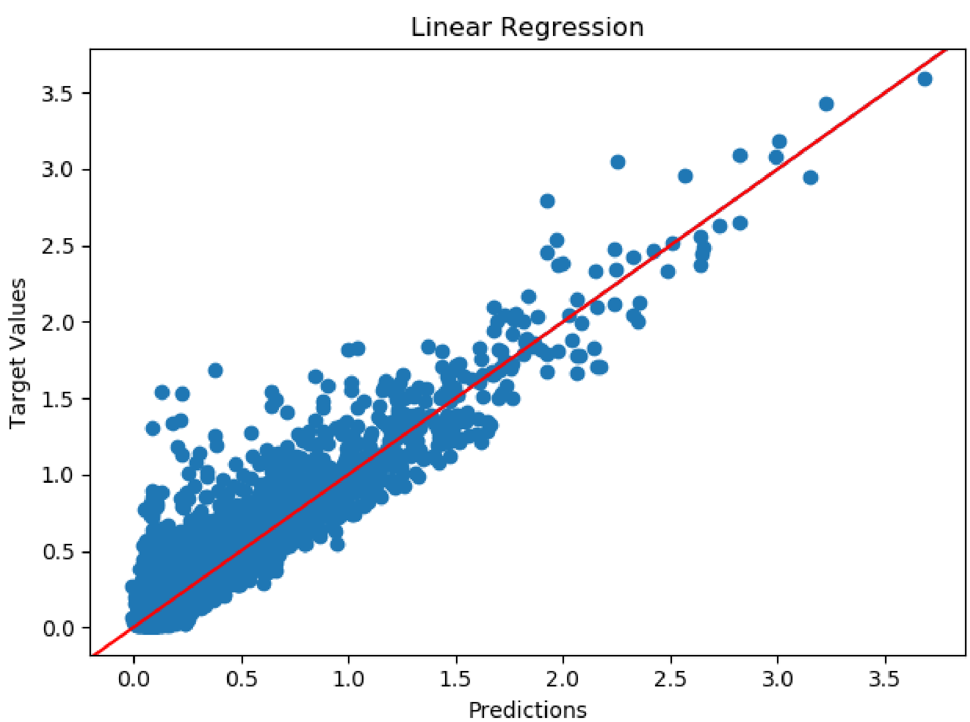
```
2129     0.678034
14718    0.029559
7764     0.173953
12192    0.067659
7224     0.198851
Name: Global_Sales_Log, dtype: float64
```

Now we look at the results we obtain from **Linear Regression** model

```
 ------------------ LINEAR REGRESSION ------------------
Mean Absolute Error (MAE) :0.09531762871628398
Mean Squared Error (MSE) :0.02045355139251478
```

Scatter Plots for **Linear Regression**

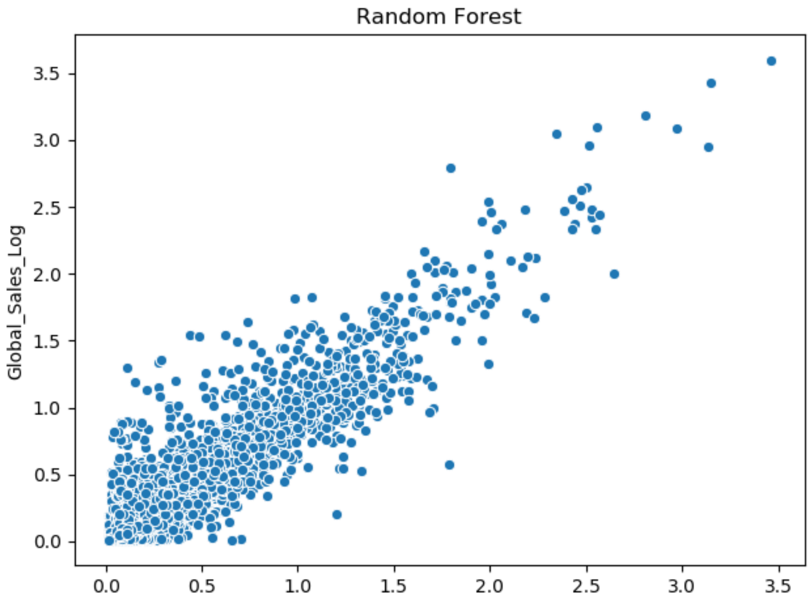


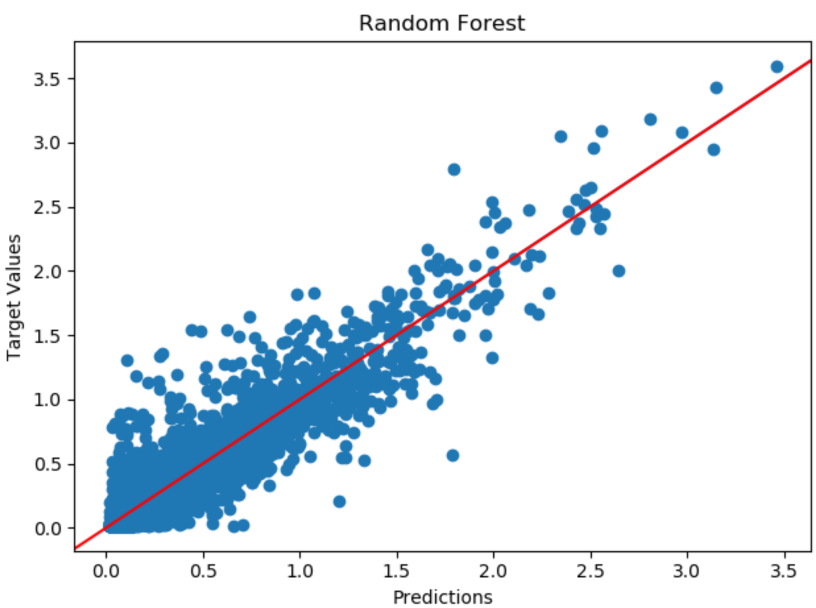


Now we look at the results we obtain from **Random Forest** model

```
----------------- RANDOM FOREST -----------------
Mean Absolute Error (MAE) :0.09612134505747305
Mean Squared Error (MSE) :0.023179380015119468
```

Scatter Plots for **Random Forest**

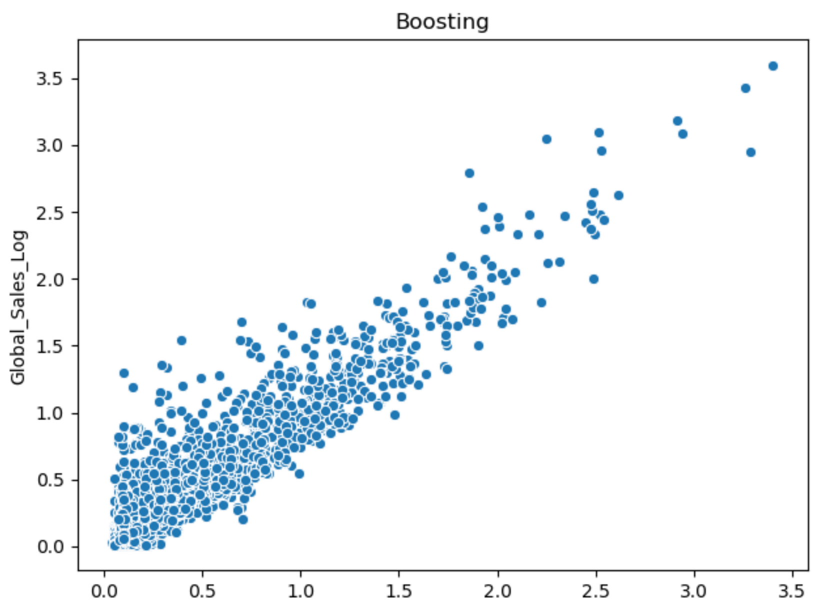


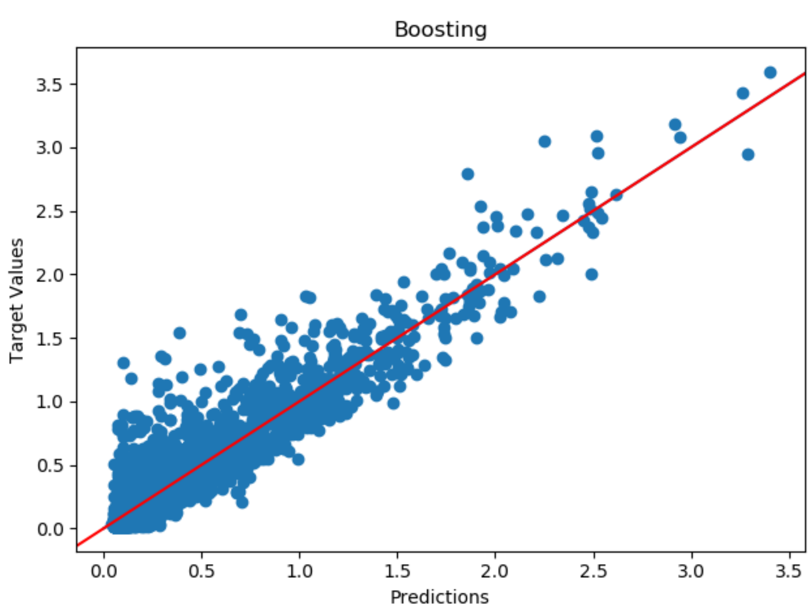


Now we look at the results we obtain from **Gradient Boosting** model

```
 ------------------ BOOSTING ------------------
Mean Absolute Error (MAE) :0.08763795682181524
Mean Squared Error (MSE) :0.0183059006095377
```

Scatter Plots for **Gradient Boosting**

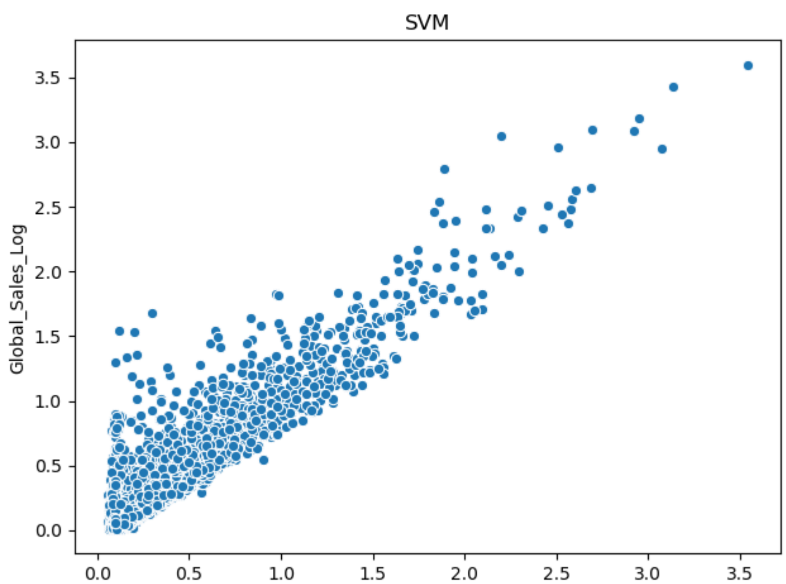


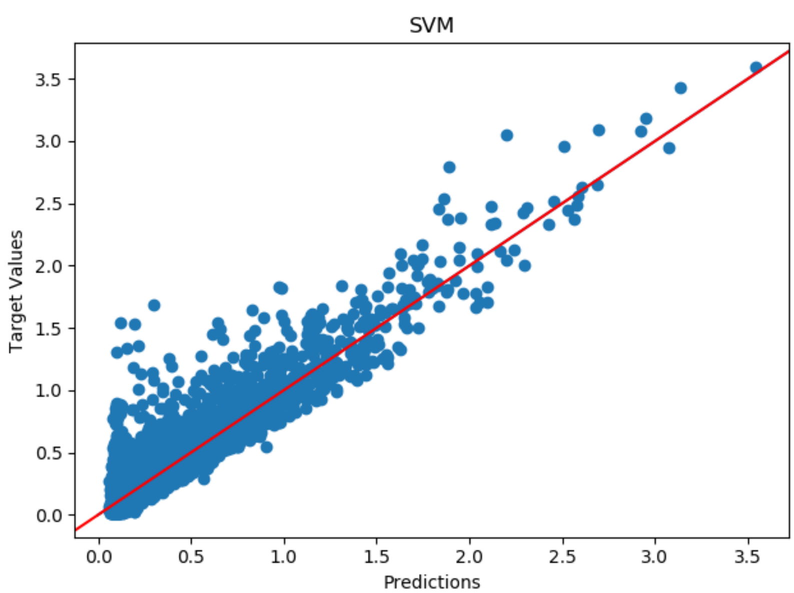


Now we look at the results we obtain from **SVM (kernel = “linear”)** model

```
 ------------------ SVM ------------------
Mean Absolute Error (MAE) :0.09970273933739358
Mean Squared Error (MSE) :0.0211107733175684
```

Scatter Plots for ~~**Gradient Boosting**~~

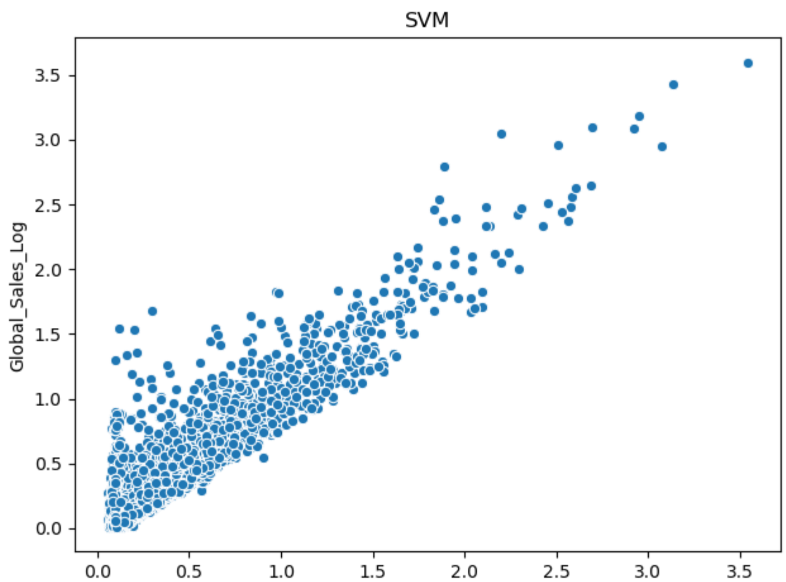


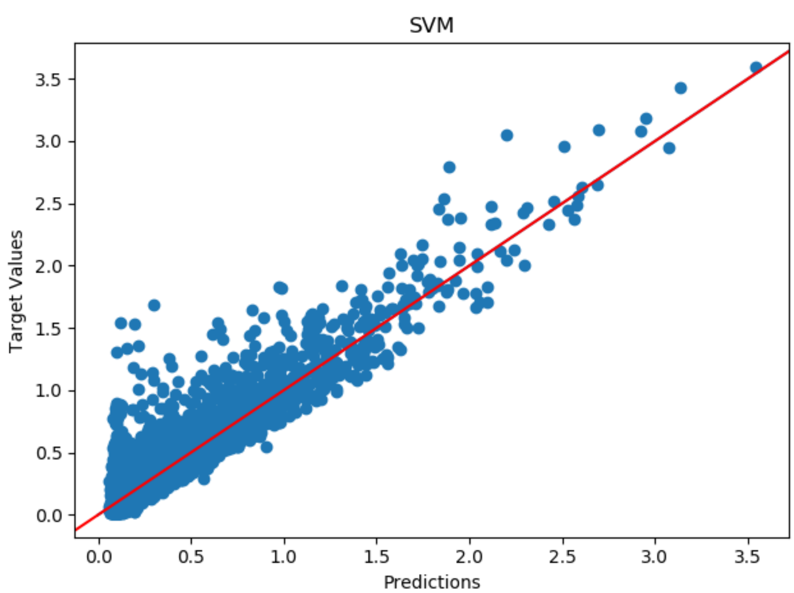


Now we look at the results we obtain from **SVM (kernel = “linear”)** model

```
------------------ SVM ------------------
Mean Absolute Error (MAE) :0.09970273933739358
Mean Squared Error (MSE) :0.0211107733175684
```

Scatter Plots for **SVM**

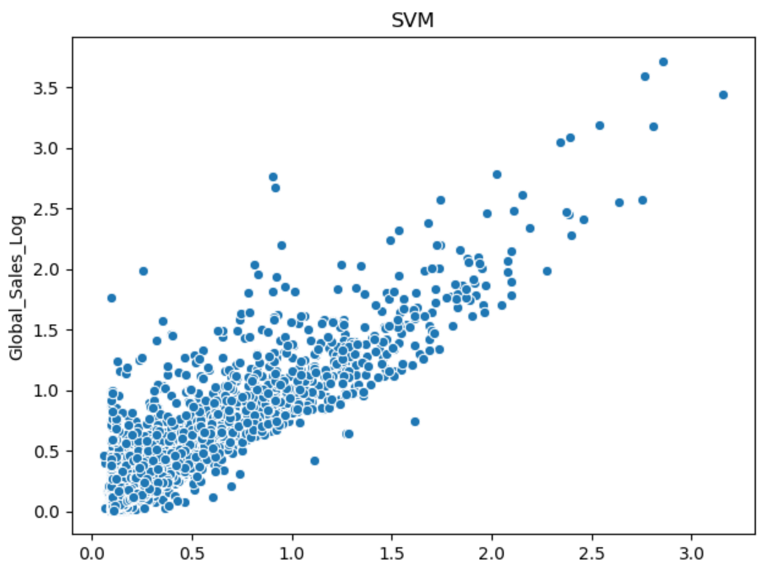


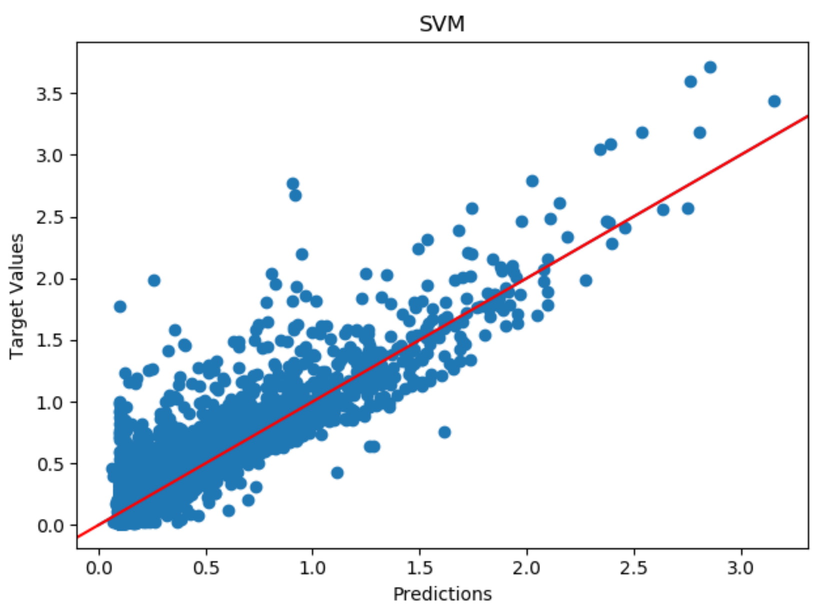


Now we look at the results we obtain from **SVM (kernel = “rbf”)** model

```
 ----------------- SVM -----------------
Mean Absolute Error (MAE) :0.10503621255094148
Mean Squared Error (MSE) :0.024464442888511576
```

Scatter Plots for **SVM**

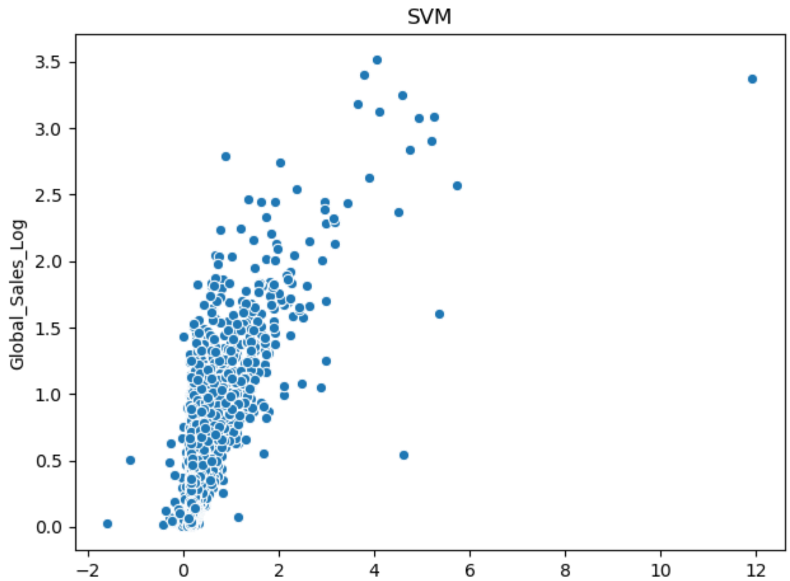


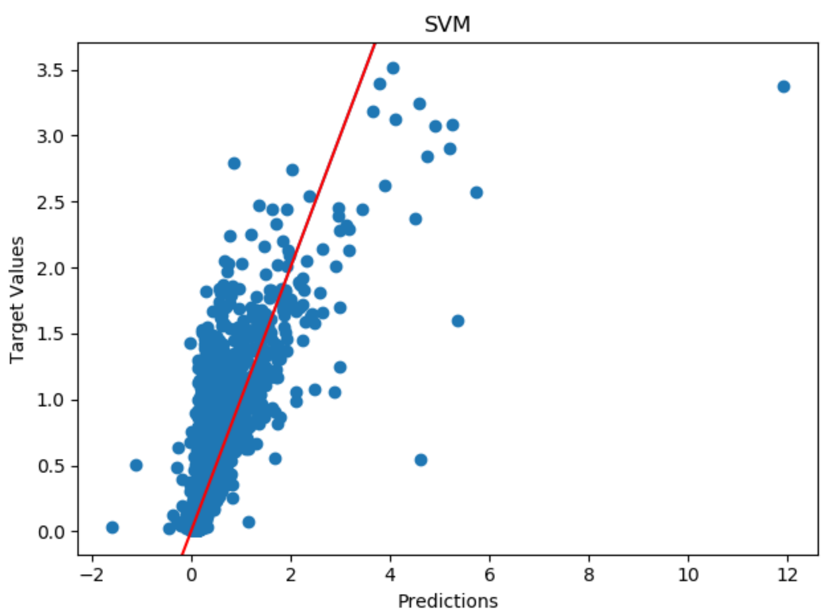


Now we look at the results we obtain from **SVM (kernel = “poly”)** model

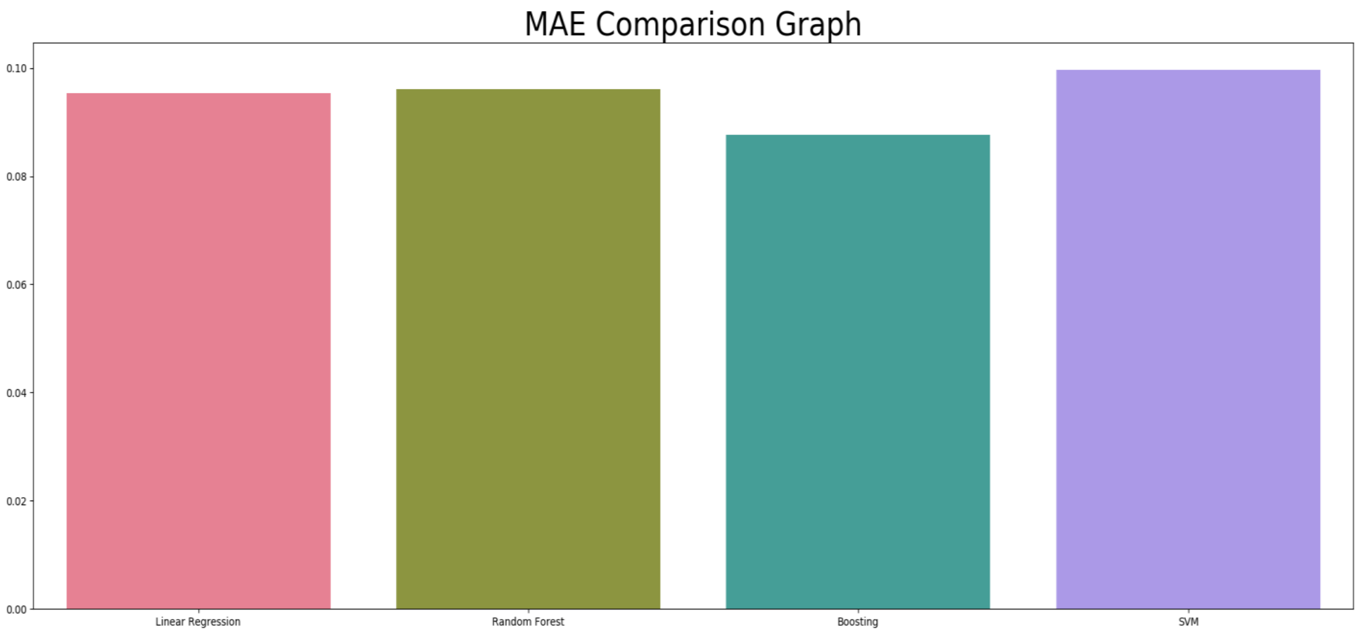
```
 ----------------- SVM -----------------
Mean Absolute Error (MAE) :0.15427244292430375
Mean Squared Error (MSE) :0.08089985259280512
=============================================
```

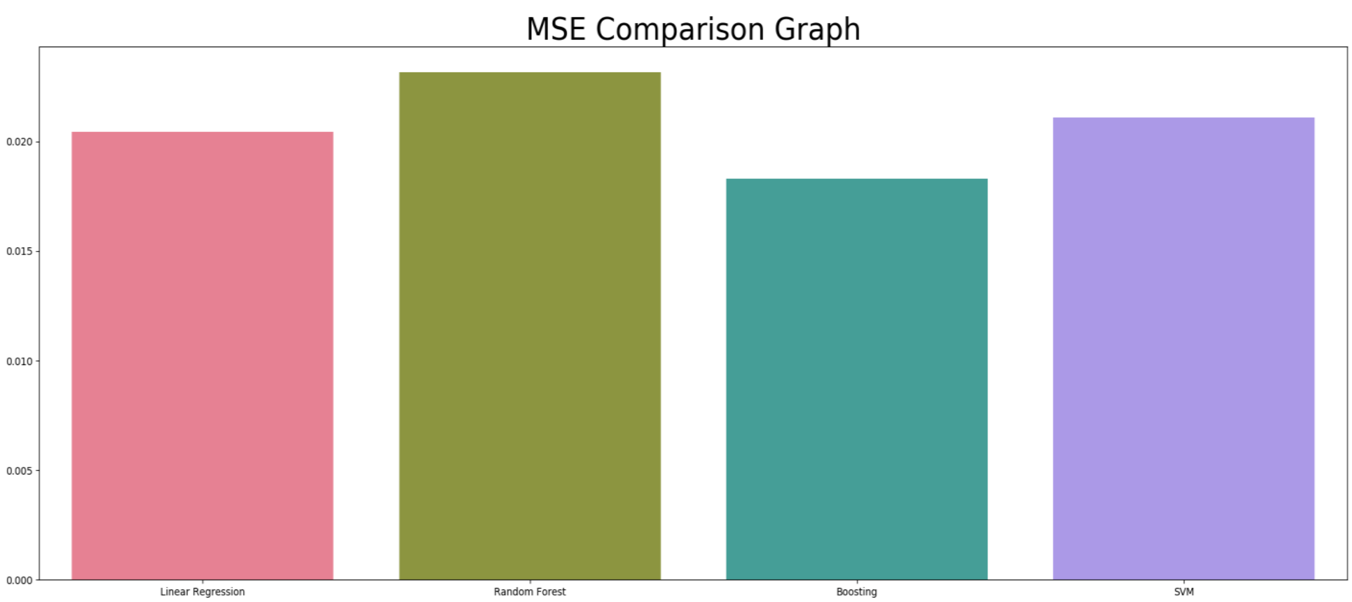
Scatter Plots for **SVM**

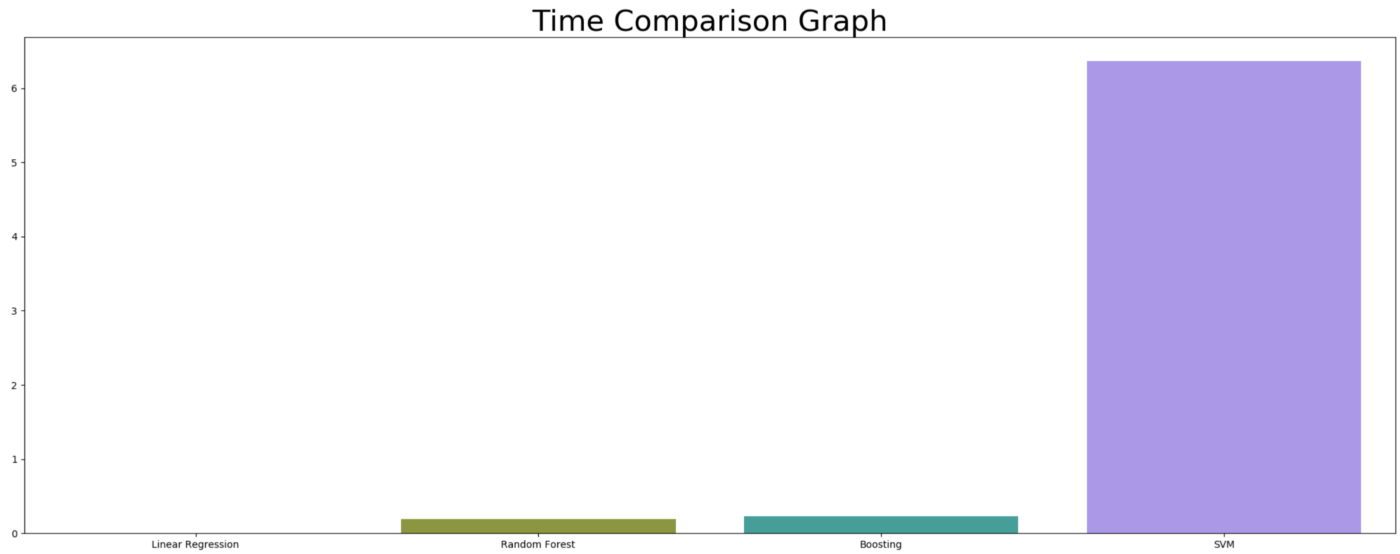




Now we look at comparison metrics i.e. MAE, MSE and time taken for different algorithms







Linear Regression time taken is so less we print it out instead

```
the average time taken for Linear Regression  0.03122849999999744
```

Conclusion

From the experimental results we can conclude the following:

- Gradient Boost has the highest accuracy i.e. lowest Mean Squared Error and Mean Absolute Error
- Gradient Boost has the lowest MSE followed by Linear Regression followed by SVM (SVR) and the highest MSE is reflected by Random Forrest
- Gradient Boost has the lowest MAE as well followed by Linear Regression, followed by Random Forest and the highest MAE is reflected by SVM (SVR)
- In terms of SVM (SVR) the polynomial kernel or poly has the worst performance in terms of MAE and MSE
- In terms of SVM (SVR) the linear kernel has the best performance in terms of MAE and MSE
- In terms of SVM (SVR) the rbf kernel has average performance in terms of MAE and MSE
- In terms of time taken SVM (SVR) performs the worst
- In terms of time taken Linear Regression performs the best

Reference

- https://chartio.com/resources/tutorials/how-to-check-if-any-value-is-nan-in-a-pandas-dataframe/
- https://stackoverflow.com/questions/49538185/what-is-the-purpose-of-numpy-log1p
- https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.median.html
- https://pandas.pydata.org/pandas-docs/stable/
- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
- https://seaborn.pydata.org/
- https://scikit-learn.org/stable/modules/preprocessing.html
- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
- https://scikit-learn.org/stable/modules/classes.html
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
- https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html
- https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html
- Deshmukh, Ratnadeep & Wangikar, Vaishali. (2011). Data Cleaning: Current Approaches and Issues.
- https://medium.com/@williamkoehrsen/machine-learning-with-python-on-the-enron-dataset-8d71015be26d