Minor Project Report

On

Prediction of Upvotes using Machine Learning

Submitted in partial fulfilment of the requirements for the award of degree of

Masters of computer application (MCA(SE))

Guide:

Prof. Dr. Amit Prakash Singh

Submitted by:

Arun Kumar (01216404518)

Department of Computer Science

University school of information, communication & Technology

Guru Gobind Singh Indraprastha University, New Delhi

(2018-2021)

CERTIFICATE

DECLARATION BY CANDIDATE

This is to certify that the project entitled "***Prediction of upvotes using Machine Learning"*** is a bonafide record of independent project/research work done by me under supervision of ***Prof. Dr. Amit Prakash Singh*** and submitted to Guru Gobind Singh Indraprastha University in partial fulfillment for the award of the Degree of MCA(SE). I Certify the content of the thesis are original.

DATE: 23.04.2020 Arun Kumar (01216404518)

CERTIFICATION BY MENTOR

This is to certify that the thesis entitled "***Prediction of upvotes using Machine Learning*** " is a bonafide record of independent project done by Arun Kumar bearing enrollment number 01216404518 under my supervision. To the best of my knowledge and believe work done by candidate is original and has not been submitted for award of any other degree.

DATE: 23.04.2020 _____

Amit Prakash Singh

(Professor)

University school of information,

Communication and technology

ACKNOWLEDGEMENT

I would like to express my sincere thanks to those who have contributed significantly to this report. It is a pleasure to extend deep gratitude to my guide **Dr. Amit Prakash Singh** for his valuable guidance and support to continuously promote me for the progress of the report. I hereby present my sincere thanks to him for his valuable suggestions towards my report, which helped me in making this report more efficient and user friendly.

I thank each and everyone's efforts who helped me in some or the other way for small and significant things.

Arun Kumar

(01216404518)

LIST OF FIGURES

Table of Contents

ABSTRACT

This project is about how we can predict the upvotes on an upcoming reddit post on the basis of the past interaction of the user on the forums. In order to predict this, I would be using a dataset containing past user engagements on the reddit forum. In this particular dataset the user's past engagement is recorded in terms or reputation, answers, views, user id, username and upvotes. This particular dataset would be used to train our model in order to predict the upvotes. Then our trained model will be used to predict the upvotes on an upcoming reddit post. In order to predict the upvotes, I will be using the regression machine learning algorithms. The machine learning algorithm used would be Stochastic Gradient Descent and Random Forest Regression.

INTRODUCTION

In this era of technological advancement, a majority of people access the Internet almost on a daily basis, be it for social networking, writing reports, researching or simply watching tv shows or movies. One of the commercial aspect of using the Internet is for promotions. Promotion of a new startup idea, promotion of an upcoming movie etc. To promote these the management team puts the content on the social media and checks how many users engage with their content and respond to it positively or negatively. One particular field is reddit, it's a community forum where user's put their queries on the site and other users can either answer it, upvote it, report it etc. The main metric to analyze positive user engagement is through the number of upvotes the post gets.

The project is about how we can analyze, visualize and pre-process a given dataset to predict the number of upvotes on a thread(post) on a public platform like reddit by training a machine learning model. Public online content platforms have a constant challenge to determine the suitable content in time to appropriately promote and therefore improve the overall engagement of the website. This project gives helps us to determine the suitable content with the help of the metric of upvotes.

The sub objectives are as follows:

1. Pre-processing the dataset consisting of tag of the question, number of views received, number of answers, username and reputation of the question author.
2. Analyzing and Visualizing the dataset.
3. Splitting the dataset into training and testing sets.
4. Training the model using the training set.
5. Testing the accuracy of the model using the testing set.
6. Evaluation metric RMSE (root mean squared error) will be used

PROBLEM STATEMENT

Reddit is one of the most used community forum out there. Reddit users are the contributors as well as the consumers of the content on reddit. They key metric for evaluation is user experience, which comes through the way we answer a particular query and gain upvotes. Crowdsourced online content platforms have a constant need to identify the best content in time to appropriately promote and thereby improve the engagement at the website.

This project helps us predict the number of upvotes on a reddit post. By this model we can help commenters get an idea of the projected quality of their comment, which will be done by predicting the number of upvotes on their posts.

The dataset used is as follows:

Variable	Definition
ID	Question ID
Tag	Anonymised tags representing question category
Reputation	Reputation score of question author
Answers	Number of times question has been answered
Username	Anonymised user id of question author
Views	Number of times question has been viewed
Upvotes	(Target) Number of upvotes for the question

Figure 1: Data dictionary

METHODOLOGY ADOPTED

3.1 Architecture of proposed methodology

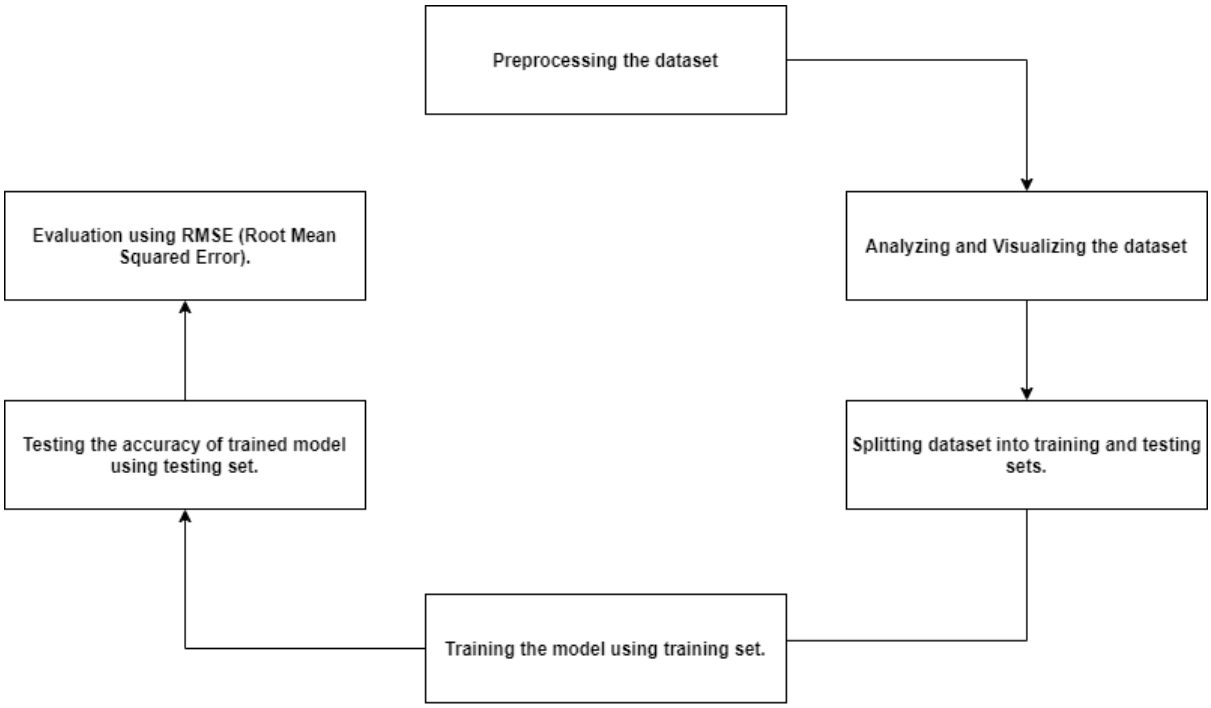

Figure 2: Architecture of proposed methodology

3.1.1 Preprocessing the dataset

Preprocessing the dataset includes various steps which are done to ensure that the data is ready to use in the model. The raw dataset contains various abnormalities like, null values, 0 values, outliers (exceptionally high or exceptionally low values), other irrelevant data etc. In order to use this data to train our model, we have to first preprocess the data. Preprocessing involves certain steps which makes the dataset usable for training our machine learning model.

One Hot encoding of ‘Tag’ attribute

One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction. Basically, a one hot encoding is a representation of categorical variables as binary vectors.

Here, the “Tag” attribute contains the anonymized question categories related to each question. With the help of one hot encoding we will able to use these categories to train our model.

Replacing NULL values in dataset

The raw data in the dataset contains multiple abnormalities, one of which in null values. Null values can’t be handled by our model during training, therefore the null values in the dataset are replaced with the mean of the current attribute. The mean is calculated using the mean() function of pandas library. Replacing the null values with mean of the attribute makes the data usable for training.

Removing outliers

Outliers are the elements in an attribute which have either exceptionally high value or exceptionally low value when compared to other elements of the attribute. These values makes the dataset non uniform, therefore these values must be removed for improved training of our model.

3.1.2 Visualization of dataset

The visualization of dataset is done in order to gain some inference about the data which is presented in the dataset. We can’t derive anything just by looking at the data in the excel sheet or some database. This is when we use the Data Visualization techniques, in which the data is

transformed into pictures, plots and graphs. Using this diagrammatic representation, we can understand the given dataset better. Data Visualization can be done using many ways like bar plots, scatter plots, graphs, sub graphs, pie charts, 3D diagrams, etc.

3.1.3 Splitting dataset into training and testing sets

When the preprocessing and Visualization of the dataset is done then comes the time to split the dataset into training and testing sets. In this step the dataset is divided into two parts namely the training set and the testing set. The training set is used to train our proposed model. With the help of this set or model tries to predict the true function and comes up with an approximation of the true function. This function is then used to predict the values.

After the model is trained, the accuracy of the approximated function is measured using the testing dataset which we produced from the splitting of the given dataset. In this step we predict the already knows target values and compare the produced result of the function with the true result. The lower the difference between the true value and predicted value, the better is our approximated function.

The function used to split the dataset into training and testing datasets is train_test_split() from the sklearn.model_section library.

3.1.4 Training the model using training set

This is one of the most crucial step of the whole machine learning process, in this step our machine learning model is trained with the help of the training data. The model approximates a function which is used to predict the target values. In this step, the values of weight’s and bias’s of the approximate function is calculated using the training set.

3.1.5 Testing the model using testing set

In this step our machine learning model has finally finished training and it is time to cross check the predicted function with the help of the testing dataset obtained from splitting of the given dataset. To achieve this, we predict the values of the testing dataset whose true values we actually know and then compare the predicted values and true values. With this comparison, we calculate the accuracy score of the model. The accuracy score gives us the accuracy of prediction of our trained model. It is defined as the ratio of correct predictions with respect to the total number of input samples.

After reaching the desired accuracy, our model has completed its training phase and is ready to predict the target values i.e., upvotes.

3.1.6 Evaluation using RMSE (Root Mean Squared Error)

The RMSE is an evaluation metric which represents the sample standard deviation of the difference between the predicted values and the true values. It is calculated as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

Figure 3: Evaluation using RMSE

Let a = (predicted value - actual value) ^2

Let b = mean of a = a (for single value)

Then RMSE= square root of b

3.2 DATASETS

3.2.1 Train.xls

Train.xls is basically an Microsoft Excel document (.xls) which contains seven columns or as we call attributes. These attributes are ID, Tag, Reputation, Answers, Username, Views and Upvotes. In this our target value is Upvotes. The ‘ID’ attribute contains basically a unique number attached to the profiles of the users which is used to distinguish every user uniquely. ‘Tag’ attribute contains a number of different characters like ‘a’, ‘g’, ‘z’ etc. which represents unique categories of the context of the reddit post, like sports, health, lifestyle, movie, fashion etc. The ‘Reputation’ attribute contains the reputation of the respected user which is calculated from the past engagements of the user on the reddit forum. The ‘Answer’ attribute contains the number of answers the particular reddit post of the user got. The ‘Username’ attribute again contains a unique username associated with unique user ID. The ‘View’ attribute contains the number of views that the reddit post got. Finally, the ‘Upvotes’ attribute contains the number of upvotes the reddit post got.

3.2.2 Test.xls

Test.xls is basically a Microsoft Excel document (.xls) which contains six columns or as we call attributes. These attributes are ID, Tag, Reputation, Answers, Username and Views. This document doesn’t contain the ‘Upvotes’ attribute because this is the testing file which will actually be used to predict the upvotes. This document is used as an input to the trained machine learning model which predicts the upvotes.

IMPLEMENTATION

4.1 ALGORITHMS IMPLEMENTED

4.1.1 Stochastic Gradient Descent

In Stochastic Gradient Descent the gradient of the loss is calculated each sample at a time and the model is updated along the ways with a decreasing strength schedule aka learning rate. In this, instead of updating the parameters of the approximation function based on the derivative of the dataset at each step, we update it based on the derivative of a randomly chosen sample. As compared to normal gradient descent which iterates through all the input data to compute the gradient.

To use stochastic gradient descent in our code, we first have to compute the derivative of the loss function with respect to a random sample. The math is shown below:

$$L(m, j) = (mA_j - P_j)^2$$

$$\frac{\partial L(m, j)}{\partial m} = 2(mA_j - P_j)(m)$$

Figure 4: SGD formula

The per-sample loss is the squared difference between the predicted and actual values; thus, the derivative is easy to compute using the chain rule.

4.1.2 Random Forest Regression

A Random Forest is a technique that is capable of both the classification and the regression tasks with the help of multiple decision trees and a technique called Bootstrap Aggregation, which is more commonly known as Bagging. Bagging is a technique where each of the tree is trained with different data sample where sampling is done with replacement.

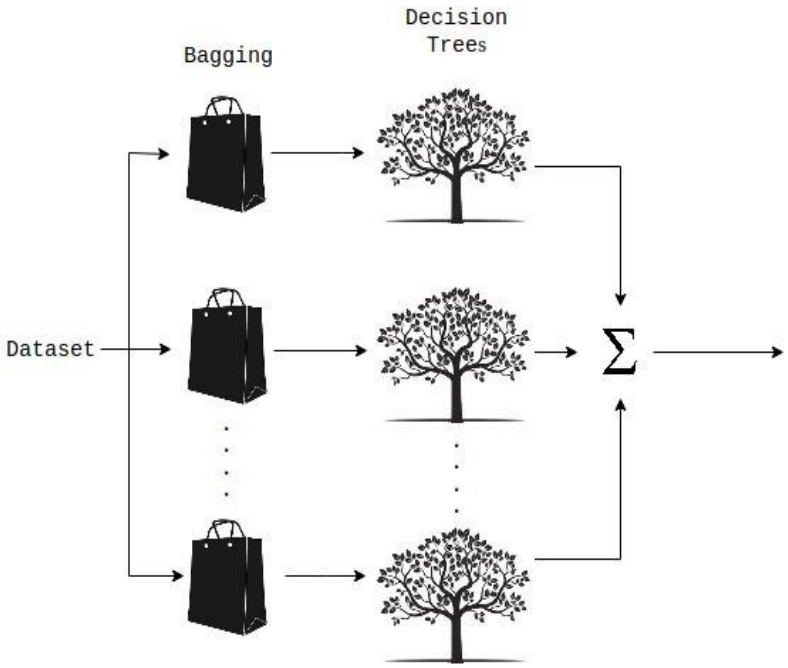

Figure 5: Random Forest Regression Process

The basic idea behind Random Forest Regression is that to combine the output of multiple decision trees to determine the final output rather than depending on individual decision trees.

Experimental Setup [Hardware and Software Requirement]

Since machine learning projects require powerful workstations, as for now I am doing this project on Google Colaboratory a.k.a Google Colab, It allows us to write and execute python code in our browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Hardware configuration of the systems provided by Google Colab:

- Processor – Intel Xeon E2630 v4 – 10 core processor, 2.2 GHz with Turboboost upto 3.1 GHz. 25 MB Cache
- Motherboard – ASRock EPC612D8A
- RAM – 128 GB DDR4 2133 MHz
- 2 TB Hard Disk (7200 RPM) + 512 GB SSD
- GPU – NVidia TitanX Pascal (12 GB VRAM)

Software requirements:

- Operating System: Windows 7 or new versions, Mac, Linux.
- Compatible browser to work on Google Colab.
- Python 3
- TensorFlow
- PyTorch

Simulation and Screenshots

Setting up the environment

At this stage we've just begun to code our project, therefore first of all before actually beginning to import the dataset and start coding, we have to setup the environment by import all the libraries needed for now. Libraries can also be added at later stages as they are needed.

Setting up environment

```
[ ]  import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     import matplotlib.colors
     import pandas as pd
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import accuracy_score, mean_squared_error
     from tqdm import tqdm_notebook

     from sklearn.preprocessing import OneHotEncoder
```

```
[ ]  sns.set()
```

Figure 6: Setting up the environment

Train.xls Dataset

```
[8]  dataframe.tail()
```

	ID	Tag	Reputation	Answers	Username	Views	Upvotes
330040	339800	c	36.0	2.0	84919	1063.0	0.0
330041	253800	c	1649.0	2.0	76730	23319.0	73.0
330042	210756	c	6178.0	2.0	91701	2453.0	15.0
330043	56089	j	89.0	2.0	80245	2107.0	3.0
330044	300553	j	2001.0	4.0	154692	2554.0	37.0

```
[9]  dataframe.shape
```

```
(330045, 7)
```

Figure 7: Train.xls Dataset

Data Visualization

In data visualization we try to depict the relationship in between the dataset visually in order to gain more intel about the dataset. How each attribute affects the target value, the relationship between the target value with respect to each input, etc.

1. **Relationship between Reputation and Upvotes:**

   As we can see, the above figure depicts the relationsip between the Reputation of a User and the Upvotes. With the help of this plot we are able to figure out the dataset contains some outliers. Outliers are the extremely high or extremely low values in an attribute when compared to the other values of the attribute.

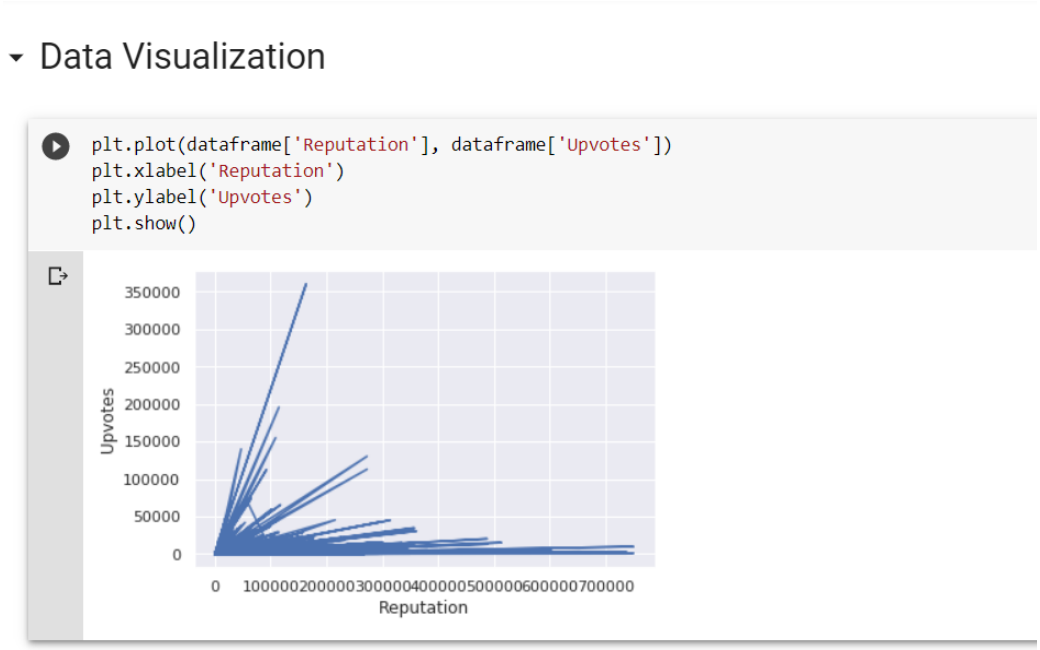

Figure 8: Reputation v/s Upvotes plot

2. **The frequency of different categories in Tag attribute**

   In this plot we are able to observe the varying frequencies of different categories of reddit posts given in our dataset. In this plot the X-axis represents all the differents tags and the Y-axis represents the frequency of those tags. From the

plot it is observed that category c and category j has the highest frequency (almost the same) and category x has the lowest frequency.

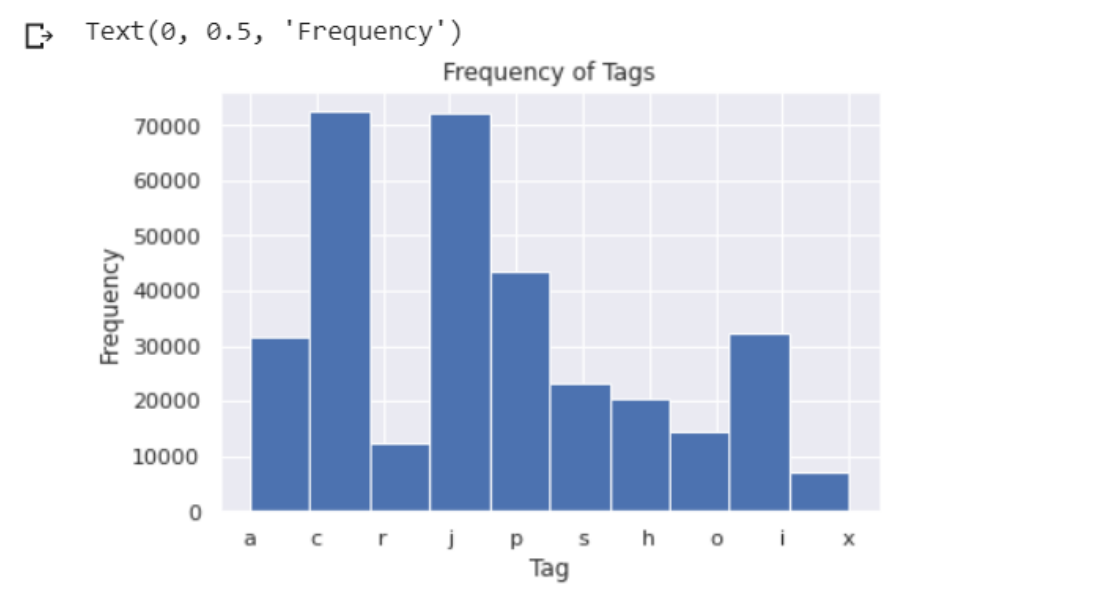

Figure 9: Frequency of Tags

3. Boxplot of Upvotes before applying log

As we can see that the values of upvotes is very sparse in this. It has outliers.

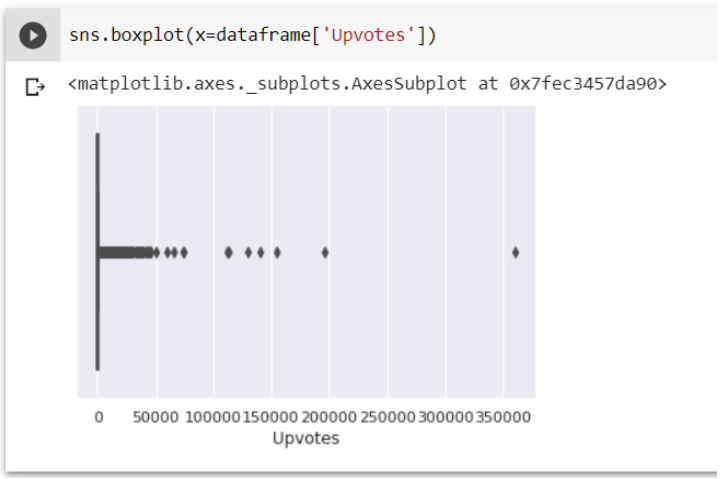

Figure 10: Boxplot of Upvotes before applying log

4. Boxplot of Upvotes after applying log

After applying the log function and re computing the values of the upvotes. Here's the new simplified Upvotes box plot.

```
[ ]  1 sns.boxplot(x=dataframe['Upvotes'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8cbfbc2d30>
```

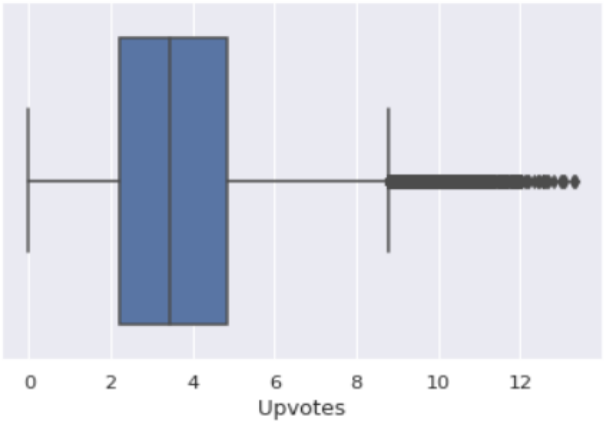

Figure 11: Boxplot of Upvotes after applying log

5. Scatter plot of Reputation v/s Upvotes

```
1 ax = sns.scatterplot(x="Reputation", y="Upvotes", data=dataframe)
```

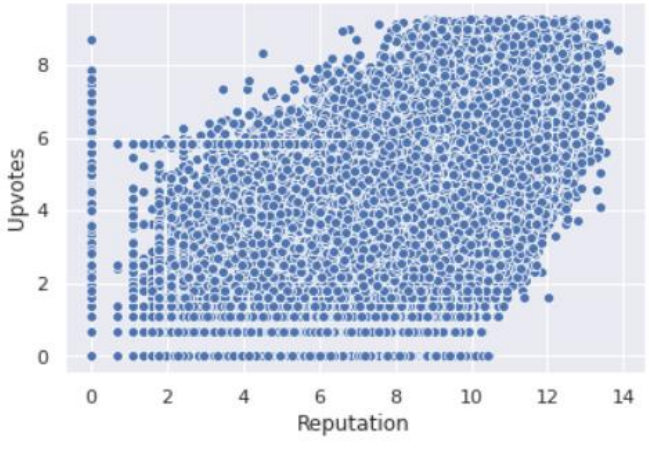

Figure 12: Scatter plot of Reputation v/s Upvotes

6. Scatter plot of Views v/s Upvotes

The given plot depicts the relationship between the Views and the Upvotes of the dataset. We can observe that as the number of views on a post increases the upvotes also tend to increase.

```
1 fig, ax = plt.subplots()
2 ax.scatter(x = train['Views'], y = train['Upvotes'])
3 plt.ylabel('Upvotes', fontsize=13)
4 plt.xlabel('Views', fontsize=13)
5 plt.show()
```

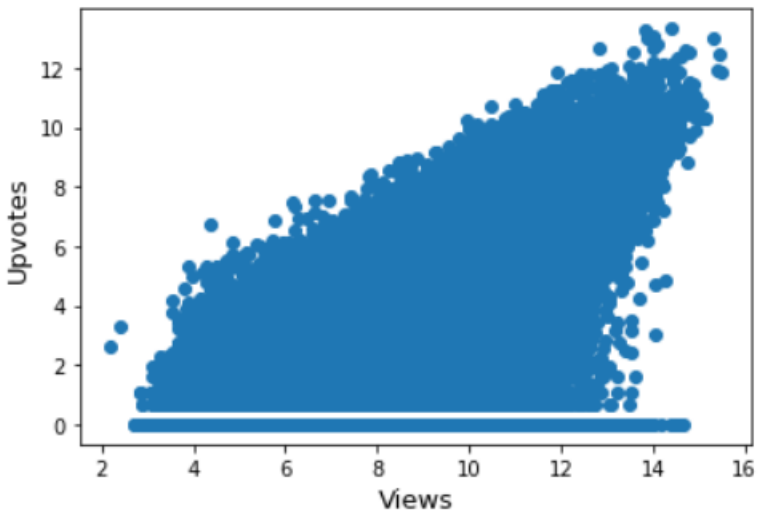

Figure 13: Scatter plot of Views v/s Upvotes

7.Pair plot representing relationship between all the relevant data.

The pair plot function will create a grid of Axes such that each numeric variable in data will by shared in the y-axis across a single row and in the x-axis across a single column. The diagonal Axes are treated differently, drawing a plot to show the univariate distribution of the data for the variable in that column.

In the pair plot below, all the attributes relevant to training our model are plotted using pair plot function. In this I've used the hue as the 'Tag' attribute, which enables us to see clearly the involvement of the tags in all aspects in our data.

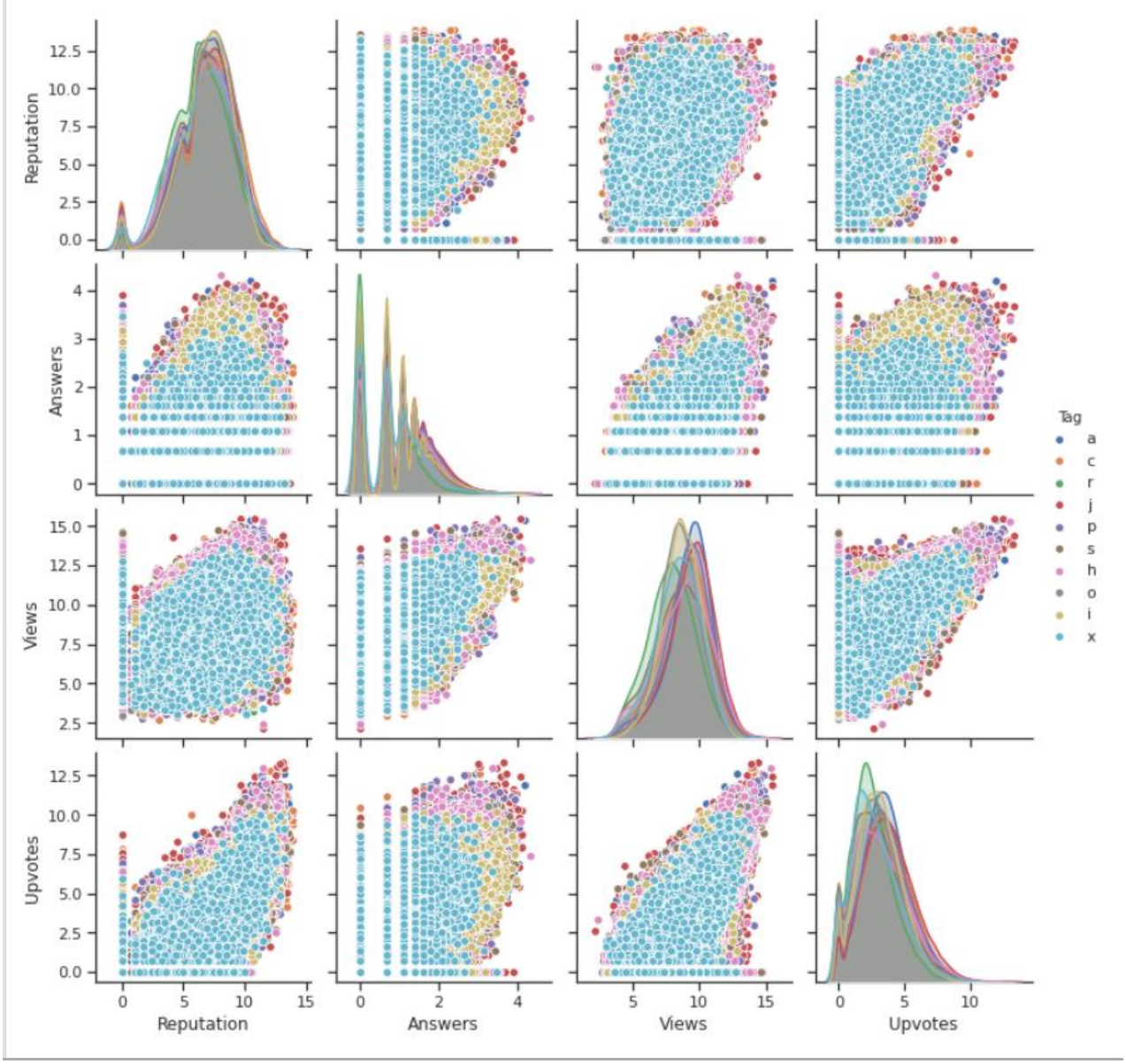

Figure 14: Pair plot of all relevant data

After this step, the pre-processing and visualization of the dataset is essentially done. The coming steps include training of the model with the current optimized dataset for training. After the model is trained and the values of the weights and bias are calculated for the approximation function, it will be time for testing the model with the testing dataset.

Training of machine learning model

This is one of the most crucial step of the whole machine learning process, in this step our machine learning model is trained with the help of the training data. Here, I have used the Stochastic Gradient Descent machine learning algorithm to compute the values of approximation function for prediction of upvotes. The fit function is the main function which takes our training dataset as the input and computes the values of weights and bias by iterating over the data. We can see the model is very efficient as it only took around half a second to iterate over the training data and computed a Bias of -4.203411 and Norm 10.58 with average loss of 0.676671 which is quite good.

```
from sklearn.linear_model import SGDRegressor
reg = SGDRegressor(loss = 'epsilon_insensitive', verbose=1, eta0=0.1)
reg.fit(X_train, Y_train)
```

```
-- Epoch 1
Norm: 10.69, NNZs: 13, Bias: -4.035703, T: 264036, Avg. loss: 0.687685
Total training time: 0.07 seconds.
-- Epoch 2
Norm: 10.70, NNZs: 13, Bias: -4.130944, T: 528072, Avg. loss: 0.677439
Total training time: 0.14 seconds.
-- Epoch 3
Norm: 10.67, NNZs: 13, Bias: -4.211178, T: 792108, Avg. loss: 0.677243
Total training time: 0.22 seconds.
-- Epoch 4
Norm: 10.62, NNZs: 13, Bias: -4.126809, T: 1056144, Avg. loss: 0.677042
Total training time: 0.29 seconds.
-- Epoch 5
Norm: 10.69, NNZs: 13, Bias: -4.182195, T: 1320180, Avg. loss: 0.676911
Total training time: 0.36 seconds.
-- Epoch 6
Norm: 10.61, NNZs: 13, Bias: -4.118334, T: 1584216, Avg. loss: 0.676715
Total training time: 0.43 seconds.
-- Epoch 7
Norm: 10.58, NNZs: 13, Bias: -4.203411, T: 1848252, Avg. loss: 0.676670
Total training time: 0.50 seconds.
Convergence after 7 epochs took 0.50 seconds
SGDRegressor(alpha=0.0001, average=False, early_stopping=False, epsilon=0.1,
             eta0=0.1, fit_intercept=True, l1_ratio=0.15,
             learning_rate='invscaling', loss='epsilon_insensitive',
             max_iter=1000, n_iter_no_change=5, penalty='l2', power_t=0.25,
             random_state=None, shuffle=True, tol=0.001,
             validation_fraction=0.1, verbose=1, warm_start=False)
```

Figure 15: Training of machine learning model

Inferences obtained after training of the model

After the successful training of our machine learning model using the Stochastic Gradient Descent algorithm these inferences are observed. At first we can observe the training data upvote labels which shows some of the true values of upvotes and also the approximated values of upvotes. On observing further we can see that the difference between the true values and approximated values is not significant, which shows the accuracy of our algorithm of selection.

Further, I've presented some data to understand the relationship between the true values and predicted values of Upvotes.

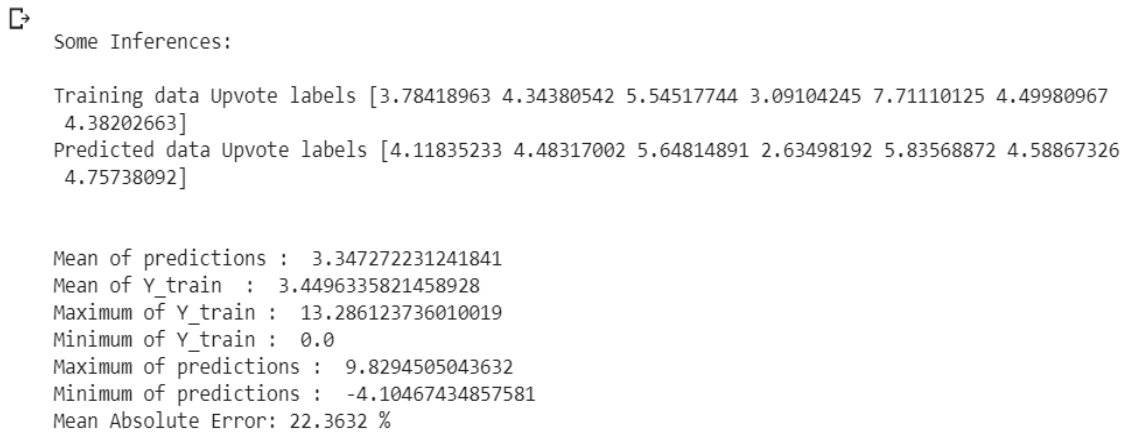

Figure 16:Inferences obtained after training the model

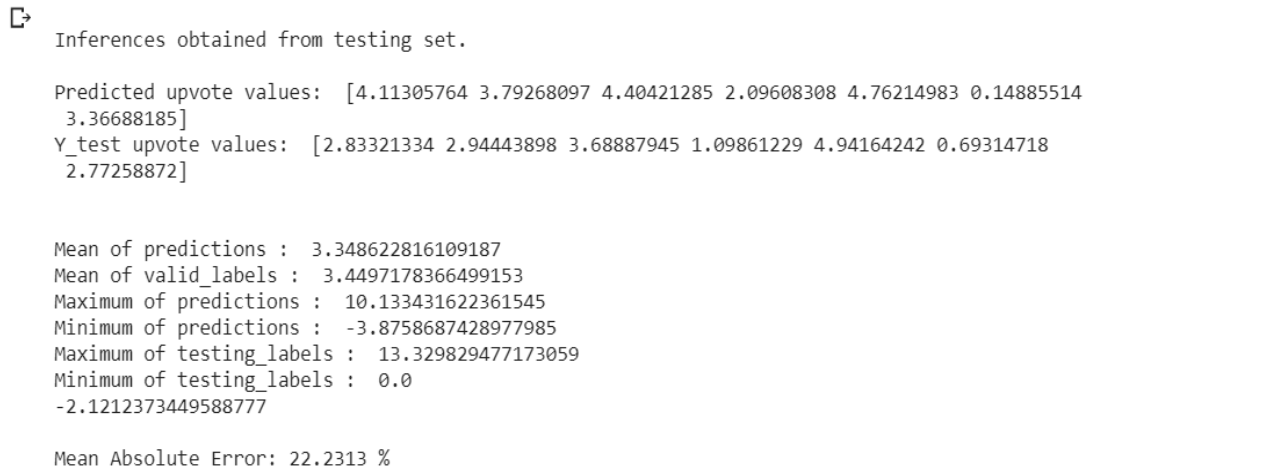

Figure 17: Inferences obtained from testing set.

RESULTS OBTAINED

Successfully predicted upvotes

With the help of SGDRegressor and RandomForestRegressor machine learning algorithms, I successfully approximated the number of upvotes that a reddit post can get. By using Random Forest Regression my model was able to achieve higher accuracy in the upvotes prediction as compared to Stochastic Gradient Descent Regression. Presented below are the screenshots of the results.

1. **From Stochastic Gradient Descent**
    - By using Stochastic gradient descent our model for able to achieve 22.3582% mean absolute error on the training data.

```
Stochastic Gradient Descent
1. Training data


Mean Absolute Error: 22.3582 %


Text(0, 0.5, 'Y_train')
```

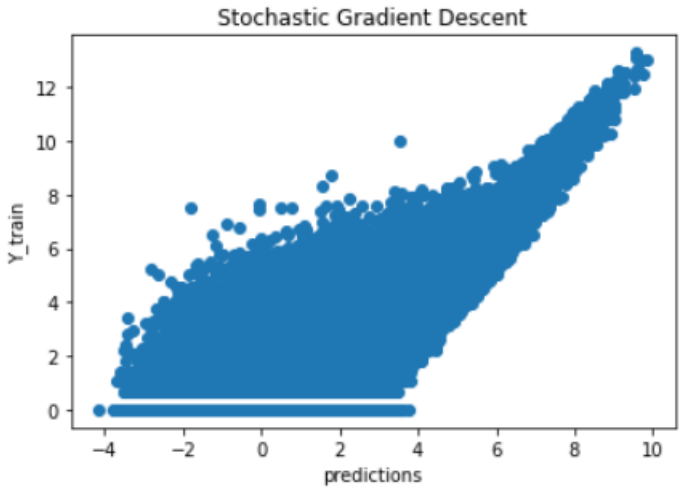

Figure 18: Stochastic Gradient Descent training result

- By using Stochastic Gradient Descent we were able to achieve 22.2192% mean absolute error on the testing data.

```
Stochastic Gradient Descent
2. Testing data


Mean Absolute Error: 22.2192 %


Text(0, 0.5, 'Y_train')
```

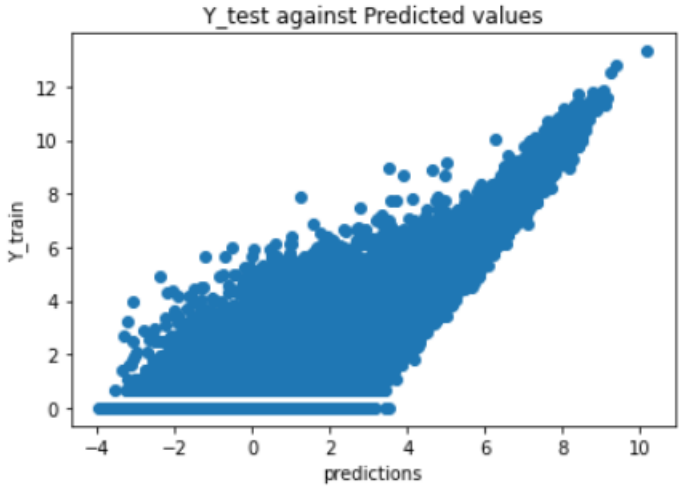

Figure 19: Stochastic Gradient Descent testing result

2. **From Random Forest Regression**
    - By using Random Forest Regression machine learning algorithm, I was able to achieve mean squared error of 0.68591.

```
Random Forest Regression

1. Training data

Mean Squared Error:  0.6859103039332792


[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:    3.9s finished
Text(0, 0.5, 'Y_train')
```

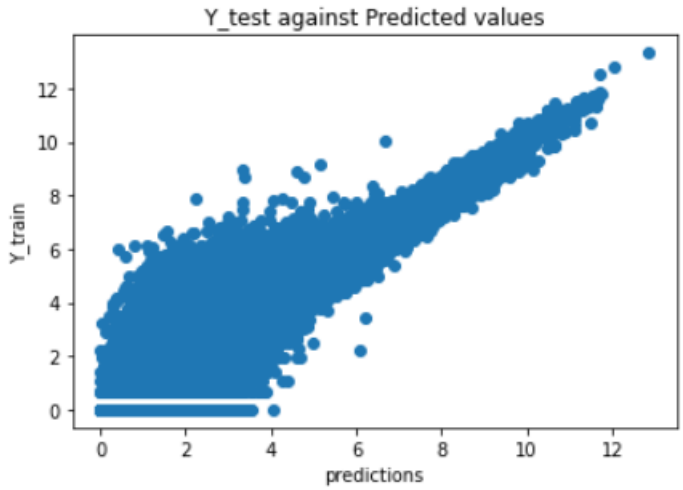

Figure 20: Random Forest Regression training result

Evaluation Metric Result

The evaluation metric adopted here is RMSE (Root Mean Squared Error). RMSE is a frequently used measure of the differences between the true values and the predicted values.

By implementing Random Forest Regression our model was able to achieve RMSE of 0.828197.

```
[34]  1 import math
      2 print('RMSE:',math.sqrt(error))

      RMSE: 0.8281970199978259
```

Figure 21: Resulting RMSE

Some predicted Upvotes

These are some of the predicted Upvotes by our model.

```
[37]  1 Upvotes[0:25]

0     202.777330
1      56.474992
2      28.529894
3       9.956285
4     311.825675
5      11.535757
6      18.576274
7      77.194303
8      48.756023
9       7.185225
10      2.232794
11     43.215943
12      7.378003
13    304.062869
14     27.755441
15    640.403369
16     27.568283
17    132.529619
18    514.054976
19      2.367509
20     17.580955
21    140.766601
22     11.306737
23      4.893780
24     25.723054
Name: Upvotes, dtype: float64
```

Figure 22: Final predicted Upvotes

CONCLUSION AND FUTURE WORK

CONCLUSION

We successfully predicted the number of upvotes that a reddit post can get. The two machine learning algorithms gave comparable accuracies.

- By using Stochastic gradient descent our model for able to achieve 22.3582% mean absolute error on the training data.
- By using Stochastic Gradient Descent we were able to achieve 22.2192% mean absolute error on the testing data.
- By using Random Forest Regression machine learning algorithm, I was able to achieve mean squared error of 0.68591.
- By implementing Random Forest Regression our model was able to achieve RMSE of 0.828197

FUTURE WORK

As for now, we were successful to complete our objective and predict the number of Upvotes on a reddit post. In upcoming future we can also aim to:

- Improve the accuracy of our model.
- Implement other machine learning algorithms and compare the results.
- Predict the number of Answers on a reddit post.
- Predict the number of Views on a reddit post.
- Making attractive GUI (Graphical User Interface) for Windows platform.
- Developing Android application for mobile usage.

REFERENCES

[1] Practice Problem: Predict Number of Upvotes, Analytics Vidhya, Hackathon, 2020. [Dataset]. Available: https://datahack.analyticsvidhya.com/contest/enigma-codefest-machine-learning-1/. [Accessed: Jan 16, 2020].

[2] How to One Hot Encode Sequence Data in Python. Author: Jason Brownlee. [Blog]. Available: https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/ .[Accessed: Feb 24, 2020].

[3] Everything you need to know about hardware requirements for Machine Learning. Author: Himanshu Singh. [Blog]. Available : https://www.einfochips.com/blog/everything-you-need-to-know-about-hardware-requirements-for-machine-learning/ . [Accessed: Feb 24, 2020]

[4] sklearn.linear_model.SGDRegressor [User Guide]. Author: scikit-learn developers (BSD License).Available:https://scikitlearn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html . [Accessed: Mar 27, 2020].

[5] sklearn.ensemble.RandomForestRegressor [User Guide]. Author: scikit-learn developers (BSDLiscense).Available:https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html. [Accessed: Mar 29, 2020].

[6] Understanding Optimization in ML with Gradient Descent & implement SGD Regressor from scratch [Article]. Author: Ruman Khan. Available:https://medium.com/@rumankhan1/understanding-optimization-in-ml-with-gradient-descent-implement-sgd-regressor-from-scratch-4e11dac74c9. [Accessed: Mar 29, 2020].

[7] Linear Regression Using SGD [Article]. Author: Nikhil Cheerla. Available: https://medium.com/deeplearningschool/2-1-linear-regression-f782ada81a53. [Accessed: Apr 5, 2020].

[8] A Beginners Guide to Random Forest Regression [Article]. Author: Krishni. Available: https://medium.com/datadriveninvestor/random-forest-regression-9871bc9a25eb. [Accessed: Apr 8, 2020].

[9] What does RMSE really mean? [Article]. Author: James Moody. Available: https://towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e. [Accessed: Apr 8, 2020].

[10] _stochastic_gradient.py [Github Repository]. Athor: scikit-learn. Available: https://github.com/scikit-learn/scikit-learn/blob/95d4f0841/sklearn/linear_model/_stochastic_gradient.py#L1528 . [Accessed: Apr 10, 2020].

[11] Visualize Machine Learning Data in Python With Pandas [Article]. Author: Jason Brownlee. Available: https://machinelearningmastery.com/visualize-machine-learning-data-python-pandas/. [Accessed Apr 10, 2020].