

Mercari Price Suggestion

Team : Kaggle Masters

Vivek Singh Thakur
MT2020074
IIIT Bangalore
Viveksingh.Thakur@iiitb.org

Boddapati Bala Priyanka
MT2020099
IIIT Bangalore
balapriyanka.boddapati@iiitb.org

Bhashkar Yadav
MT2020136
IIIT Bangalore
Bhashkar.Yadav@iiitb.org

Abstract—Predicting the price of a product is quite difficult since a small difference in products can cause a higher difference in their prices. Since we are trying to predict the price of a product which is a real value this can be solved using regression. We have extracted information from text column item description which has helped in predicting the prices. This is a detailed report on our work on mercari price suggestion.

Index Terms—Word Cloud, Vectorization , TFIDF Vectorizer, Linear Regression, Ridge Regression, Lasso Regression, LGBM

I. PROBLEM STATEMENT

Mercari is an online e-commerce website where users can sell or buy anything. It wants to give price suggestions to the sellers but it is a challenging task as the sellers are allowed to put just about anything on mercari website. Because the price of a product depends on its brand, condition, use of that product and various other things. It is common to get different prices for the same products with different brands and condition of the product and description of the product. So in the Mercari Challenge, we need to build a model that will automatically suggest an appropriate Price for the product selected by the user from the list of products offered by the seller based on the features given in the dataset. Pricing helps in identifying an intermediate between supply and demand.

II. LIBRARIES USED

- **NumPy** It is a python library that consists of multi-dimensional array objects and a collection of functions that helps us in working with those arrays. Numpy helps in performing mathematical functions on arrays. NumPy arrays are much faster than python lists because they are stored at contiguous locations.
- **Pandas** It helps in data analysis, data manipulation, data cleaning and data wrangling with the help of some data structures and operations. Pandas provides a DataFrame object with integrated indexing which is used for the data manipulation such as merging, reshaping.
- **Matplotlib** It is a plotting library for creating interactive visualizations. It is used for creating animated and interactive visualizations in Python.
- **Seaborn** It is also used for data visualizations that uses matplotlib underneath. It provides a high-level interface for drawing attractive statistical graphs.

III. DATASET DESCRIPTION

Dataset was taken from “kaggle.com”

Link of Dataset: <https://www.kaggle.com/c/mercari/data>

Brief of the dataset:

There are 8 features in the dataset out of which price is the Target variable.

- **Train-id**: id of the product (numerical)
- **Name**: the title of the listing. (textual)
- **item condition id**: the condition of the items provided by the seller (numerical)
- **category name**: category of the listing (categorical)
- **brand name**: brand of the product (categorical)
- **price**: the price that the item was sold for (target)
- **shipping**: 1 if shipping fee is paid by the seller and 0 if the shipping fee is paid by the buyer (binary)
- **item description**: the complete description of the item. (textual)

IV. EDA AND PREPROCESSING FOR NUMERICAL DATA

Before doing the preprocessing and feature engineering it is important to analyze the data and identify the distribution of data. It gives valuable insights into the patterns hidden in the data, helps in detecting outliers in the data and finding relations among variables. Statistical tools play an important role in proper visualization of the data. Proper EDA gives interesting features of the data which helps in our data preprocessing, feature engineering and model selection criterion as well.

In Data Preprocessing, the data gets converted in such a way that our model can easily understand and interpret it. It transforms the raw data into meaningful format by cleaning and organizing the raw data that we have. It involves handling missing, standardization, encoding the categorical data. Models that are trained on meaningful data perform better when compared to the ones that are trained on raw data.

There are different techniques for handling missing data. Deleting the rows with missing data. Deleting the columns having large amount of missing data. Imputing the missing values with mean, median or mode. We can predict the missing values using a machine learning algorithm. This method results in better accuracy, unless the missing value has high variance. Standardization is a scaling technique where the values are spread around the mean with a standard deviation 0. In Standardization we are transforming the values in such a way that

the mean is 0 and standard deviation is 1. Standardization is helpful when the data follows Gaussian Distribution.

A. Handling Missing Data

Missing data can be due to reasons like errors in the data entry, information is not available and faulty instruments that are used for recording the data. We have observed that there aren't any missing values in the numerical columns namely item-condition-id, shipping and price. We have removed the products with price zero since there cannot be a product with price zero, these are outliers.

B. Univariate Analysis

It is a simplest form of statistical analysis that involves a single variable. Main goal of this analysis is to describe, summarize and find the patterns in the data.

1) *item-condition-id*: item-condition-id with id 1 is the most frequent one in the products with more than 60 lakh products which contributes to 43 percentage of the total products. While id 5 is the least frequent one with only 0.2 percentage of the total products.

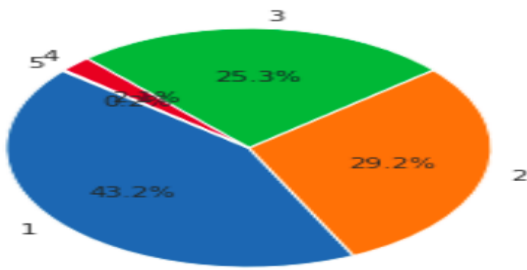


Fig. 1. Item Condition Id

2) *price*: It is seen that our target variable is heavily right skewed. In skewed data tail part can behave as an outlier and these outliers can adversely affect our model performance for the regression models.

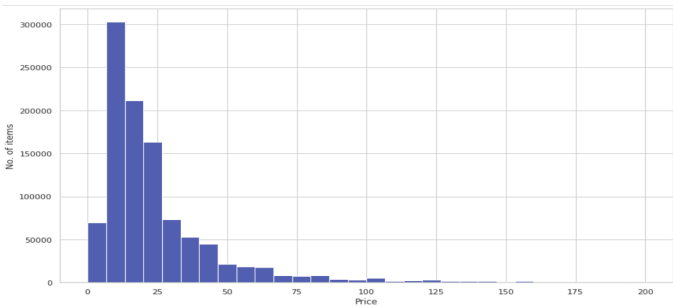


Fig. 2. Price Distribution

So, there is need for removing the skew that is converting the skewed data into normal distribution. A log transformation helps in transforming the right skewed distribution to a normal distribution.

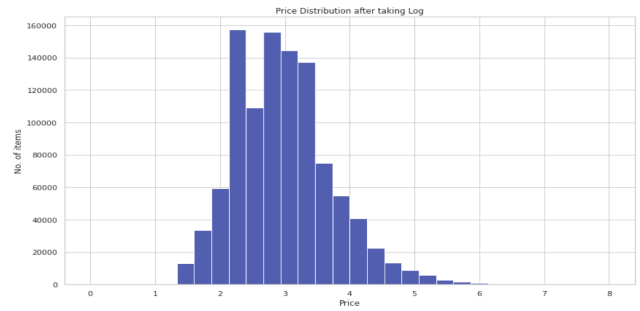


Fig. 3. Price Distribution after removing skew

3) *shipping*: There are a greater number of products where shipping fee is paid by the buyer when compared to the ones where shipping fee is paid by the seller. Out of total number of products, for nearly 55 lakh products shipping fee is paid by the buyer.

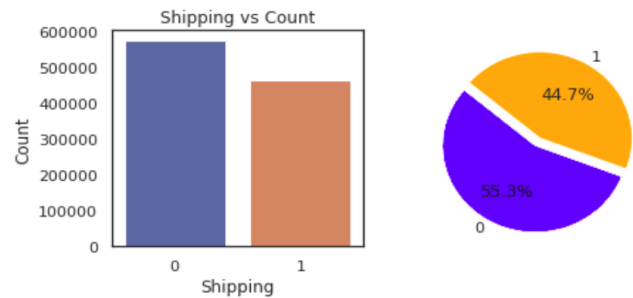


Fig. 4. Shipping

C. Bivariate Analysis

Bivariate analysis is a statistical analysis that helps in finding the relationship between 2 variables. It also helps in finding the strength of association between the 2 variables.

1) *itemconditionid vs price*: Here we used box plots to identify the relation between the itemconditionid and the price. Box Plot: It helps us in describing the numerical with the quartiles. The lines that are extending out of the box are known as whiskers. Box plot gives minimum, maximum, Q1 (First Quartile), Q2 (Median), Q3 (Third Quartile) and IQR = Q3 - Q1. The median of the products having itemconditionid 5 is higher to products with other itemconditionids.

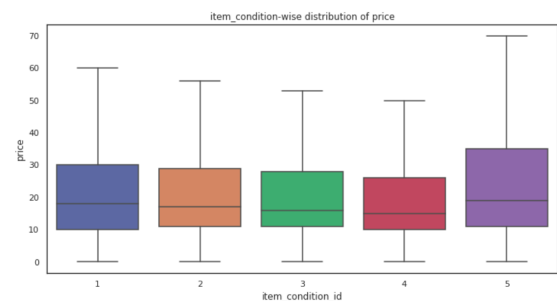



Fig. 5. Item Condition Id vs Price



A violin plot comparing the distribution of logPrice for two shipping statuses: 0 (blue) and 1 (orange). The y-axis is labeled 'logPrice' and ranges from 0 to 8. The x-axis is labeled 'shipping' with categories 0 and 1. The blue violin for shipping=0 is wider at the bottom (lower logPrice) and has a median around 3. The orange violin for shipping=1 is wider at the top (higher logPrice) and has a median around 3.5. Both distributions are roughly bell-shaped but skewed.

Fig. 6. Shipping vs Price

V. EDA AND PREPROCESSING FOR TEXTUAL DATA

A. Handling Missing Data

We have observed that the column category contains 3 subcategories in each row. So it has been divided into 3 columns namely subcategory1, subcategory2 and subcategory3. Missing Values in these columns are replaced with an empty space. Missing values in column brandname is replaced with 'Not Known' and missing values in the column item description is replaced with 'No Description Given' using the fillna method.

B. Univariate Analysis and Bivariate Analysis

After plotting bar graphs for the brand name column we have found out the following things. Nike, Pink, Victoria Secret are the top three brands. There are 11 unique categories in subcategory1, 114 unique categories in subcategory2 and 851 unique categories in subcategory3.

1) *Brand vs Price*: We have grouped all the products which don't have a brand name into Unknown Brand and plotted a graph to see how the price varies between the branded and unbranded products. It is seen that the branded products have highest peakedness when compared to unbranded products.

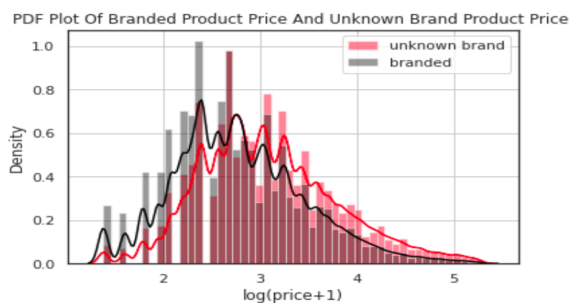


Fig. 7. Brand vs Price

VI. FEATURE ENGINEERING

we remove contraction from words, that is we are replacing the shorthands with their full forms because words "don't" and "do not" have the same meaning but they are treated differently by machine which in turn increases the number of features. Words like "can't" become can not, "don't become "do not".

A. Removing Stopwords

Stopwords are the most common words in the language which donot add much meaning to the sentence.Words like is, am, are some of the stopwords. Since we want to perform sentimental analysis on the text column item description we removed the stopwords as they do not provide any useful information needed for sentimental analysis. We added a new feature named count of stopwords to our data.This is done using nltk(Natural Language Tool Kit)

B. Word Cloud

We have created a word cloud for the item description to identify the words which occurred more frequently. Word Cloud is the graphical representation of words where size of the word indicates it's frequency. Here in the item description brand, new, free, shipping are the most frequent words. Victoria Secret is the most prominent brand as visible in the word cloud. Sellers use the words like new, free to increase the sales of their products.



Fig. 8. Word Cloud

C. New Features

As a part of Feature Engineering we have decided to add few new features like count of each word, isbranded, sentimental score analysis that would help in predicting the price of a product.

1) *isbranded*: We have observed that many products do not have a brand name associated with it and the price of the products varies with brand name. The price of branded products is higher when compared to the ones with unknown

brand. So we added a new feature `isbranded` which would help in predicting the price. Its value is 0 when the product is not branded and 1 when it is branded.

D. Sentiment Score Analysis

Price of a product also depends on the kind of opinion expressed in the item description column. Products with positive opinion can be priced higher when compared to the products with neutral and negative opinion. So we performed a sentiment analysis on the item description column. We have removed the unnecessary multiple spaces, numbers, punctuation, stopwords from the item description and converted upper case letters to lower case before applying the sentiment analysis. We have done sentiment analysis using `nlTK` in python. It returns 4 values namely positive, negative, neutral and compound for each row in the item description column. These positive, negative, neutral and compound scores indicates the proportion of text in the sentence that falls into these categories. If our sentence is rated as 55 percentage positive, 20 percentage negative, 25 percentage neutral then all these should sum up to 100 percentage.

Sentimental Analysis is used for identifying the kind of sentiment in the text that is positive, negative or neutral. It is used to extract opinions from the text. Sentiment Analysis divides the text into sentences and then into phrases and then to token of words. Identifies sentiment corresponding to each phrase and assigns the score on the basis of sentiment in between -1 to +1.

- positive : Any word having a score above 0.05 is considered as positive.
- negative : Any word having a score below -0.05 is considered negative.
- neutral : Any word having a score between -0.05 and 0.05 inclusively is considered neutral.
- compound : It calculates the sum of lexicon ratings which have been normalized between -1 and +1. -1 indicates extreme negative and +1 indicates extreme positive.

```
def sentiment_analysis(data):
    """this function performs sentiment score analysis of each datapoint"""
    sentiment_score = SentimentIntensityAnalyzer()
    sentiment = []
    for sentence in tqdm(data):
        sentiment.append(sentiment_score.polarity_scores(sentence))
    return sentiment
```

Fig. 9. Sentiment Analysis

E. Splitting the data

We need to split the data into train and cross validation data before applying machine learning model on the data. We are using train-test split method of `sklearn` for splitting the data. We train the model on train data and use the cross validation data for prediction that is we are checking the metric RMSLE. A small part of training is taken for cross validation based on its size.

VII. FEATURE EXTRACTION

Our data consists of categorical, textual and numerical columns. But our machine learning model can only understand numerical data. So before giving this data to a model we need to convert it into numerical format which are known as vectors. That is in order to be able to use textual data we have to convert the textual features to some integer form so that our Machine Learning Algorithms may use it for prediction as Machine Learning Algorithms works only on numerical data, therefore, in order to do that we have applied the following methods on textual data in order to extract numerical data out of textual data. That is done using the vectorization. There are various techniques that can be used for Vectorization like `word2vec`, Bag-of-Words, TF-IDF. We have used Bag-of-words for vectorization of categorical data and TF-IDF for vectorization of textual data.

A. Vectorization

We have used Bag-of-Words (BoW) for the vectorization of columns namely `subcategory1`, `subcategory2`, `subcategory3`, `brand name` and `name`. Bag-of-Words helps in identifying the occurrence (frequency) of words within a sentence or a document. It maintains vocabulary of known words and their corresponding count. It ignores the order of words in the sentence. Hence Bag-of-Words can be viewed as an unordered collection of words in a bag. Bag-of-Words is used to extract features from the text. It converts variable length texts into a fixed length vectors where length of the vector is equal to the size of the vocabulary. We have implemented Bag-of-Words using `CountVectorizer()` function of `sk-learn` library. `CountVectorizer()` function converts the text into tokens, constructs a vocabulary of known words and also encodes new documents using the constructed vocabulary. It converts the upper case letters into lower case letters and ignores the punctuation by default.

B. TF-IDF Vectorizer

Here, TFIDF stands for Term Frequency – Inverse Document Frequency.

TF - Term Frequency : It indicates how frequent a word appears in a document. It gives a scoring for frequency of the word in the current document. That is count of word in that document divided by total number of words in the document.

IDF - Inverse Document Frequency : It gives a score of how rare the word is across the documents. That is logarithm of total number of documents divided by number of documents containing that word.

It gives frequency scores for the words which helps in identifying interesting words. In this method what we do is we divide the frequency of each word by the total number of words in the particular document and then we calculate the inverse term frequency for each word by taking log of the number of documents by the number of documents containing that word. In the end, we multiply the term frequency and inverse document frequency and calculate this for each word as shown below. `TfidfVectorizer` of `sk-learn` library that converts

the text into tokens, build the vocabulary and also finds the inverse document frequencies. We have performed TF-IDF on the item description column since each row of that column contains various words we thought instead of simple vectorization we should apply TF-IDF. Finally we have done a sentiment analysis of each row in the item description column which is a sentence using the TF-IDF values computed above to give a particular integer score based on how much positive or negative that sentence is in this sentiment analysis phase.

$$tf_{t,d} = \frac{n_{t,d}}{\text{Number of terms in the document}}$$

$$idf_t = \log \frac{\text{number of documents}}{\text{number of documents with term 't'}}$$

Fig. 10.

TF-IDF score=TF*IDF

C. Feature Scaling

We have added 6 new numerical columns namely count of stopwords, count of words, positive, negative, neutral and compound which are derived from the sentimental analysis. Since these numerical can contain values in different ranges we need to scale them before we feed them to the model to get better performance. Feature Scaling means converting the values into same range so that one feature does not dominate other features. There are various methods for feature scaling like Standardization and Normalization. We have used Standardization for scaling the features. Standardization involves rescaling the distribution such that the values have a mean 0 and standard deviation 1. We have used StandardScaler() of sk-learn library for doing the standardization.

D. Encoding shipping,itemconditionid,isbranded

We have encoded the 3 columns namely shipping, itemconditionid, isbranded using OneHotEncoding. OneHotEncoding works well for columns that does not have large number of different values. It works by creating a dummy binary variable for each distinct value(category) in the column that is encoded. OneHotEncoding is done easily by using getdummies() method of pandas.

E. Combining all the features in matrix

After converting all the features into numerical format we merged all those features in a matrix and converted them into a sparse row format using hstack of scikit-learn. hstack takes a sequence of arrays and merges them in horizontal way. This will be given to our machine learning models.

VIII. MODELS IMPLEMENTED

A. Linear Regression

Linear Regression is a model in supervised machine learning which is used to predict a continuous output variable based on values of one or more input variables. Here Regression simply means to predict values of continuous variable and Linear simply means that our output variable is linearly related with the input variables via a equation where degree of each of the input variables is one. For eg :- let we have two input variables x_1 , x_2 , a bias b and want to predict a continuous variable x_3 then linear regression assumes x_3 can be predicted using the following form of equation: $w_1x_1 + w_2x_2 + b = x_3$. Here w_1 , w_2 are the weights of input variables x_1 , x_2 which are some constants and the bias b is added to make our model more generalized and not bias to any particular input variable.

$$\text{Cost Function} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Here, y_i is the original value of the column to be predicted in the i th row

\hat{y}_i is the predicted value by the algorithm in the i th row

n is the number of data rows in the training data

Fig. 11. Linear Regression

The whole idea of linear regression model revolves around minimizing the above cost function's value. Lower the cost function value, better the linear regression model. In our project this was the first algorithm we used and the input variables were the various columns of our training data set and the output continuous variable to be predicted was the Price column. This model gave us a accuracy of 0.48556 on kaggle which was better than the sample csv submission.

B. Lasso Regression

LASSO here is an acronym for Least Absolute Shrinkage and Selection Operator. It is a regularization technique which avoids overfitting. Regularization reduces the variance of a model without considerable increase in the bias of the model. Lasso Regression uses the concept of shrinkage. Shrinkage indicates the values are shrunk towards mean. Lasso Regression model overcomes the problem of overfitting of data which was with simple Linear Regression. It also helps in feature selection. It performs L1 Regularization in which complex overfitting models are penalized and hence overfitting problem is solved to a greater extent. Here Regularization first sorts the models from least overfit to more overfit and the least overfit model is picked up as the best choice. The major difference here from Linear Regression is the cost function used which is of the form

$$\text{Cost Function} = \sum_{i=1}^n \left[y_i - \sum_{j=0}^p \omega_j \times x_{ij} \right]^2 + \lambda \sum_{j=0}^p |w_j|$$

Here, y_i is the original value of the column to be predicted in i th row

ω_j is the weight of the j th column

x_{ij} is the value of the j th column in the i th row of the training data

Here λ is the tuning parameter and controls the strength of the L1 Regularization Penalty and its various effects are as follows:

- When $\lambda = 0$ then it becomes same as Linear Regression and no parameters are eliminated.
- When λ increases the bias increases.
- When λ variance increases.

When we used this model in our Mercari project it gave a little better accuracy than linear regression which is 0.6304.

Fig. 12. Lasso Regression

When we used this model in our Mercari project it gave a little better accuracy than linear regression which is 0.47986.

C. Ridge Regression

Ridge here refers to the characteristics of the function we are trying to optimize which is the cost function given below:

$$\text{Cost Function} = \sum_{i=1}^n \left(y_i - \sum_{j=0}^p w_j \times x_{ij} \right)^2 + \lambda \sum_{j=0}^p (w_j)^2$$

Here, y_i is the original value of the column to be predicted in i th row

w_j is the weight of the j th column

x_{ij} is the value of the j th column in the i th row of the training data

Here λ is a tuning parameter which controls the strength of L2 Regularization penalty and its various effects are as follows:

- When $\lambda = 0$ then like Lasso, Ridge Regression also becomes same as Linear Regression and no parameters are eliminated.
- When $\lambda = \infty$ then all the weights of input features become 0.

Hence ideal λ is between 0 and ∞

Fig. 13. Ridge Regression

This model uses L2 regularization which makes it better than lasso which uses L1 regularization as in L1 regularization some coefficients may be eliminated but in L2 none are eliminated, also L2 regularization unlike L1 regularization will not result in sparse models. We applied this model to our Mercari Project thinking that Lasso Regression must be eliminating some of the important features due to which we are not able to improve much and it gave a good accuracy of 0.47536.

D. Light GBM

Light GBM is a fast, distributed gradient boosting framework based on decision tree algorithm, used for ranking, classification and many other machine learning tasks. This is a very new model developed by Microsoft Corporation and is now being widely used because of its accuracy, efficiency, and stability. A major advantage of Light GBM over decision trees is that Light GBM grows trees vertically or leaf-wise while classical decision tree algorithms grows tree horizontally or level-wise.

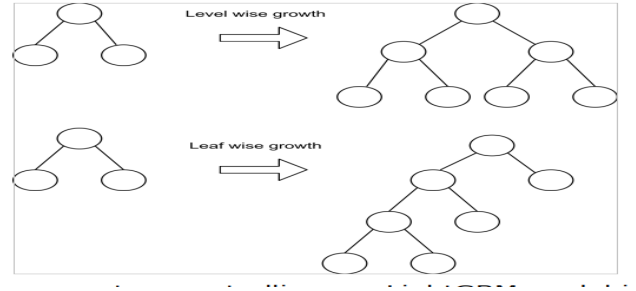


Fig. 14.

The various parameters controlling our LightGBM model in the Mercari Project are given below

```
params= {
    'learning_rate':[0.1], //boosting learning rate
    'max_depth':[15], // maximum depth of the tree
    'n_estimators':[200], // no. Of boosted trees to fit
    'num_leaves':[75], // no. Of leaves in the tree
    'boosting_type':['gbdt'], // traditional Gradient Boosting Decision Tree
}
```

Fig. 15. LightGBM

Using this model with the parameters given below gave us the best accuracy we achieved in this Mercari Project Competition which was about 0.45519.

IX. EVALUATION METRICS

Evaluation Metric used in Mercari project is RMSLE(Root Mean Squared Logarithmic Error.

RMSLE is calculated as follows

$$\text{RMSLE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (\log(p_i + 1) - \log(a_i + 1))^2}$$

Where:

n is the total number of observations in the (public/private) data set,

p_i is your prediction of target, and

a_i is the actual target for i .

$\log(x)$ is the natural logarithm of x ($\log_e(x)$).

Fig. 16. RMSLE

X. TRAINING THE MODEL

After Feature engineering and Feature extraction, next step is to train the data which consists of new features that are added during the feature engineering. We tried different models based on regression to reduce the RMSLE. Following are the different models on which we have trained our data and corresponding RMSLE.

- Linear Regression
- Ridge Regression
- Lasso Regression
- LightGBM(LGBM)

S.No.	Algorithm	RMSLE score
1.	Linear Regression	0.48556
2.	Lasso Regression	0.47986
3.	Ridge Regression	0.47536
4.	LightGBM	0.45519

XI. CONCLUSION

With a RMSLE of around 0.45 it is hard to predict the exact price of a product. LightGBM was comparatively better compared to all other algorithms that we have used.

ACKNOWLEDGMENT

We would like to thank Professor G. Srinivas Raghavan for his detailed lectures and for giving us the opportunity to work on the project and our Machine Learning teaching assistant Amitesh Anand for helping us out whenever we are stuck in numerous occasions and other Teaching Assistants for helping us out in the course.

Leader-board was a great motivation to work on the project and it forced us to read up various articles and blogs which gave us ideas and zeal for the project.

REFERENCES

- [1] Saptashwa Bhattacharya. Ridge and Lasso Regression.[Online]. Available: <https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b>
- [2] Pushkar Mandot. What is LightGBM, How to implement it and how to fine tune parameters?[Online]. Available: <https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc>
- [3] LightGBM Official documentation.[Online]. Available:<https://lightgbm.readthedocs.io/en/latest/>
- [4] Seaborn Official Documentation.[Online]. Available:<https://seaborn.pydata.org/>
- [5] Matplotlib Official Documentation. [Online]. Available:<https://matplotlib.org/>
- [6] Mukesh Chaudhary. TF-IDF Vectorizer.[Online]. Available:<https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a>
- [7] Numpy Official Documentation.[Online]. Available:<https://numpy.org/doc/>
- [8] Pandas Official Documentation.[Online]. Available:<https://pandas.pydata.org/>