RATING PREDICTION

Submitted by:

Roshan Kumar Verma

Acknowledgement

This Project would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.
I would like to thank Flip-Robo Technologies Bangalore for their guidance and Constant supervision as well as for providing necessary information regarding
The project & also for their support in completing the project.
I would like to express my gratitude towards my parents & my SME of Flip-Robo Mohd. Kashif for their kind co-operation and encouragement which help
Me in completion of this project.
My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

Introduction

Business Problem Framing

We have a client who has a website where people write different reviews for technical products. Now they are adding a new feature to their website i.e. The reviewer will have to add stars(rating) as well with the review. The rating is out 5 stars and it only has 5 options available 1 star, 2 stars, 3 stars, 4 stars, 5 stars. Now they want to predict ratings for the reviews which were written in the past and they don't have a rating. So, we have to build an application which can predict the rating by seeing the review. As a Data Scientist, we have to apply our analytical skills to give findings and conclusions in detailed data analysis written in Jupyter notebook.

Conceptual Background of the Domain Problem

Created model can be used to predict ratings of reviews, it might be a good tool for online shopping sites such as Flipkart.com, Amazon.in etc and manufacturer companies who might look into their product ratings and reviews so that they can make their investment according to demand of customers , which might help them to save time and earn more profits.

Review of Literature

Now-a-days people buy more products from online sites, so by using this model, rating prediction and positive or negative review prediction of a product is easy and less time consuming.

Motivation for the Problem Undertaken

Data Science help us to make predictions at areas like health sectors, auto industry, education, media etc. For our project we decided to implement Prediction of Rating of reviews. We have to create a model which will help online sites and manufacturer of a product if they have to modify their product or not according to customer's requirement.

Analytical Problem Framing

Mathematical/ Analytical Modelling of the Problem

We would perform one type of supervised learning algorithms: classification. While it seems more reasonable to perform classification since we have 5 types of Rating i.e., 1, 2, 3, 4, 5. The prediction will base on the dataset which is scrapped from Flipkart and Amazon technical product. Scrapping dataset contains reviews and ratings of different products.

Data Sources and their formats

We scraped more than 20000 rows of data. We can scrape more data as well. More the data better the model. In this section we need to scrape the reviews of different laptops, Phones, Headphones, smart watches, Professional Cameras, Printers, Monitors, Home theatre, Router from different e-commerce websites.

We fetched other data as well. Our columns are –

1. Rating

2. Review

3. Long Review

```
In [4]: 1 df
```

Out[4]:

	Unnamed: 0	Review_title	Review_text	Ratings
0	0	Perfect product!	Amazing Power in this processer..\n\nYou can d...	5
1	1	Highly recommended but wait for upcoming mi la...	I am civil engineer and i installed SketchUp a...	5
2	2	Fabulous!	Super work horse\n\nLatest processor, ticks al...	5
3	3	Very Good	Everything is good except display. The process...	4
4	4	Value-for-money	Good laptop form Redmi... performance wise goo...	4
...	...	...	...	...
27105	27105	Great product	Very Nice Watch at This price..... Display is ...	5
27106	27106	Worth every penny	Really osm 👍👍👍 love it\nGo for it\nDisplay qua...	5
27107	27107	Mind-blowing purchase	Loved the watch, it gives you the premium feel...	5
27108	27108	Awesome	Go for it if you love roud dial watch and your...	5
27109	27109	Does the job	Value for money smartwatch.\n\nPros:-\n1. The ...	3

27110 rows × 4 columns

Data Pre-processing Done

Cleaned the data from junk values. Replace multiple spaces with single space So that it'll easy to classify it.

- The Complete data is divided in the ration of 80:20 for train and test respectively.
- There is no null value present in the data set and almost all the columns type is objective so we don't need to check for outliers.
- Once our data is ready, we'll do further processing.
- I have not dropped any column since the model accuracy is good.

Hardware and Software Requirements and Tools Used

The system requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from client's point of view. System requirements are all of the requirements at the system level that describe the functions which the system as a whole should fulfil to satisfy the stakeholder needs and requirements, and is expressed in an appropriate combination of textual statements, views, and non-functional requirements; the latter expressing the levels of safety, security, reliability, etc., that will be necessary.

Hardware requirements: -
1. Processor — core i5 and above
2. RAM — 8 GB or above
3. SSD — 250GB or above

Software requirements: -
Anaconda

Libraries: -

From sklearn.preprocessing import StandardScaler

As these columns are different in **scale**, they are **standardized** to have common **scale** while building machine learning model. This is useful when you want to compare data that correspond to different units.

from sklearn.preprocessing import Label Encoder

Label Encoder and One Hot Encoder. These two encoders are parts of the SciKit Learn library in Python, and they are used to convert categorical data, or text data, into numbers, which our predictive models can better understand.

From sklearn.model_selection import train_test_split,cross_val_score

Train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. With this function, you don't need to divide the dataset manually. By default, Sklearn train_test_split will make random partitions for the two subsets.

The algorithm is trained and tested K times, each time a new set is used as testing set while remaining sets are used for training. Finally, the result of the K-Fold Cross-Validation is the average of the results obtained on each set.

from sklearn.neighbors import KNeighborsClassifier

K Nearest Neighbor(KNN) is a very simple, easy to understand, versatile and one of the topmost machine learning algorithms. KNN used in the variety of applications such as finance, healthcare, political science, handwriting detection, image recognition and video recognition

from sklearn.linear_model import LogisticRegression

The library sklearn can be used to perform logistic regression in a few lines as shown using the LogisticRegression class. It also supports multiple features. It requires the input values to be in a specific format hence they have been reshaped before training using the fit method.

from sklearn.tree import DecisionTreeClassifier

Decision Tree is a white box type of ML algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as Neural Network. Its training time is faster compared to the neural network algorithm. The time complexity of decision trees is a function of the number of records and number of attributes in the given data. The decision tree is a distribution-free or non-parametric method, which does not depend upon probability distribution assumptions. Decision trees can handle high dimensional data with good accuracy.

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

Just make the comments more appropriate so that we'll get less word to process and get more accuracy. Removed extra spaces, converted email address into email keyword, likely wise phone number etc. Tried to make Comments small and more appropriate as much as it was possible.

Testing of Identified Approaches (Algorithms)

Listing down all the algorithms used for the training and testing.

- KNeighborsClassifier()
- LogisticRegression()
- BernoulliNB()
- DecisionTreeClassifier()
- RandomForestClassifier()

I applied all these algorithms in the dataset.

In [33]:

```
models = [
    LogisticRegression(),
    KNeighborsClassifier(),
    BernoulliNB(),
    DecisionTreeClassifier(),
    RandomForestClassifier(),
    KNeighborsClassifier(n_neighbors=3)
]
```

Run and Evaluate selected models

```
Model for: LogisticRegression()
0.9260420509037256
[[ 393    1    1    3    3]
 [   8   89    0   10    5]
 [   3    1  243   57   39]
 [   1    0    8  935  206]
 [   0    0    0   55 3361]]
              precision    recall  f1-score   support

           1       0.97      0.98      0.98       401
           2       0.98      0.79      0.88       112
           3       0.96      0.71      0.82       343
           4       0.88      0.81      0.85      1150
           5       0.93      0.98      0.96      3416

    accuracy                           0.93      5422
   macro avg       0.94      0.86      0.89      5422
weighted avg       0.93      0.93      0.92      5422
```

```
Model for: KNeighborsClassifier()
0.8731095536702324
[[ 358    6    4   18   15]
 [  13   75    3   12    9]
 [  13    4  246   38   42]
 [   9    2   27  885  227]
 [  14    2   51  179 3170]]
              precision    recall  f1-score   support

           1       0.88      0.89      0.89       401
           2       0.84      0.67      0.75       112
           3       0.74      0.72      0.73       343
           4       0.78      0.77      0.78      1150
           5       0.92      0.93      0.92      3416

    accuracy                           0.87      5422
   macro avg       0.83      0.80      0.81      5422
weighted avg       0.87      0.87      0.87      5422
```

```
Model for: BernoulliNB()
0.8048690520103283
[[ 378    0    1    5   17]
 [  23   31    5   17   36]
 [   5    1  135   41  161]
 [   3    2   11  644  490]
 [  10   17   40  173 3176]]
              precision    recall  f1-score   support

           1       0.90      0.94      0.92       401
           2       0.61      0.28      0.38       112
           3       0.70      0.39      0.50       343
           4       0.73      0.56      0.63      1150
           5       0.82      0.93      0.87      3416

    accuracy                           0.80      5422
   macro avg       0.75      0.62      0.66      5422
weighted avg       0.79      0.80      0.79      5422
```

```
Model for: DecisionTreeClassifier()
0.9575802286978975
[[ 392    4    0    1    4]
 [   2  103    2    0    5]
 [   7    2  294   17   23]
 [   0    0    6 1055   89]
 [   3    1    9   55 3348]]
              precision    recall  f1-score   support

           1       0.97      0.98      0.97       401
           2       0.94      0.92      0.93       112
           3       0.95      0.86      0.90       343
           4       0.94      0.92      0.93      1150
           5       0.97      0.98      0.97      3416

    accuracy                           0.96      5422
   macro avg       0.95      0.93      0.94      5422
weighted avg       0.96      0.96      0.96      5422
```

```
Model for: RandomForestClassifier()
0.9693839911471782
[[ 399    0    0    0    2]
 [   4  101    0    5    2]
 [   6    0  296   26   15]
 [   0    0    4 1064   82]
 [   0    0    1   19 3396]]
              precision    recall  f1-score   support

           1       0.98      1.00      0.99       401
           2       1.00      0.90      0.95       112
           3       0.98      0.86      0.92       343
           4       0.96      0.93      0.94      1150
           5       0.97      0.99      0.98      3416

    accuracy                           0.97      5422
   macro avg       0.98      0.94      0.96      5422
weighted avg       0.97      0.97      0.97      5422
```

```
Model for: KNeighborsClassifier(n_neighbors=3)
0.8922906676503135
[[ 361    6    3   16   15]
 [   6   90    1    9    6]
 [  14    6  256   37   30]
 [  12    6   27  926  179]
 [  12    7   46  146 3205]]
              precision    recall  f1-score   support

           1       0.89      0.90      0.90       401
           2       0.78      0.80      0.79       112
           3       0.77      0.75      0.76       343
           4       0.82      0.81      0.81      1150
           5       0.93      0.94      0.94      3416

    accuracy                           0.89      5422
   macro avg       0.84      0.84      0.84      5422
weighted avg       0.89      0.89      0.89      5422
```

```
: 1 # Best modal is random forest so doing Hyper parameter on that
  2 param_grid = {
  3     'max_depth' : range(10,14),
  4     'max_features' : ['auto', 'sqrt'],
  5     'min_samples_leaf': range(1, 4)
  6 }
  7 gridSearchCV = GridSearchCV(RandomForestClassifier(),param_grid=param_grid,refit=True,verbose=3)
  8 gridSearchCV.fit(X,Y)
```

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
[CV 1/5] END max_depth=10, max_features=auto, min_samples_leaf=1; total time=   2.8s
[CV 2/5] END max_depth=10, max_features=auto, min_samples_leaf=1; total time=   2.4s
[CV 3/5] END max_depth=10, max_features=auto, min_samples_leaf=1; total time=   2.8s
[CV 4/5] END max_depth=10, max_features=auto, min_samples_leaf=1; total time=   2.4s
[CV 5/5] END max_depth=10, max_features=auto, min_samples_leaf=1; total time=   2.7s
[CV 1/5] END max_depth=10, max_features=auto, min_samples_leaf=2; total time=   2.6s
[CV 2/5] END max_depth=10, max_features=auto, min_samples_leaf=2; total time=   2.4s
[CV 3/5] END max_depth=10, max_features=auto, min_samples_leaf=2; total time=   2.6s
[CV 4/5] END max_depth=10, max_features=auto, min_samples_leaf=2; total time=   2.4s
[CV 5/5] END max_depth=10, max_features=auto, min_samples_leaf=2; total time=   2.6s
```

Key Metrics for success in solving problem under consideration

Precision: can be seen as a measure of quality, **higher precision** means that an algorithm returns more relevant results than irrelevant ones.

Recall is used as a measure of quantity and high recall means that an algorithm returns most of the relevant results.

Accuracy score is used when the True Positives and True negatives are more important. **Accuracy** can be used when the class distribution is similar.

```
Out[36]: GridSearchCV(estimator=RandomForestClassifier(),
                      param_grid={'max_depth': range(10, 14),
                                  'max_features': ['auto', 'sqrt'],
                                  'min_samples_leaf': range(1, 4)},
                      verbose=3)
```

```
In [37]: 1 gridSearchCV.best_params_
Out[37]: {'max_depth': 13, 'max_features': 'sqrt', 'min_samples_leaf': 1}
```

```
In [38]: 1 joblib.dump(gridSearchCV.best_estimator_,'randomForestClassifier.obj')
Out[38]: ['randomForestClassifier.obj']
```

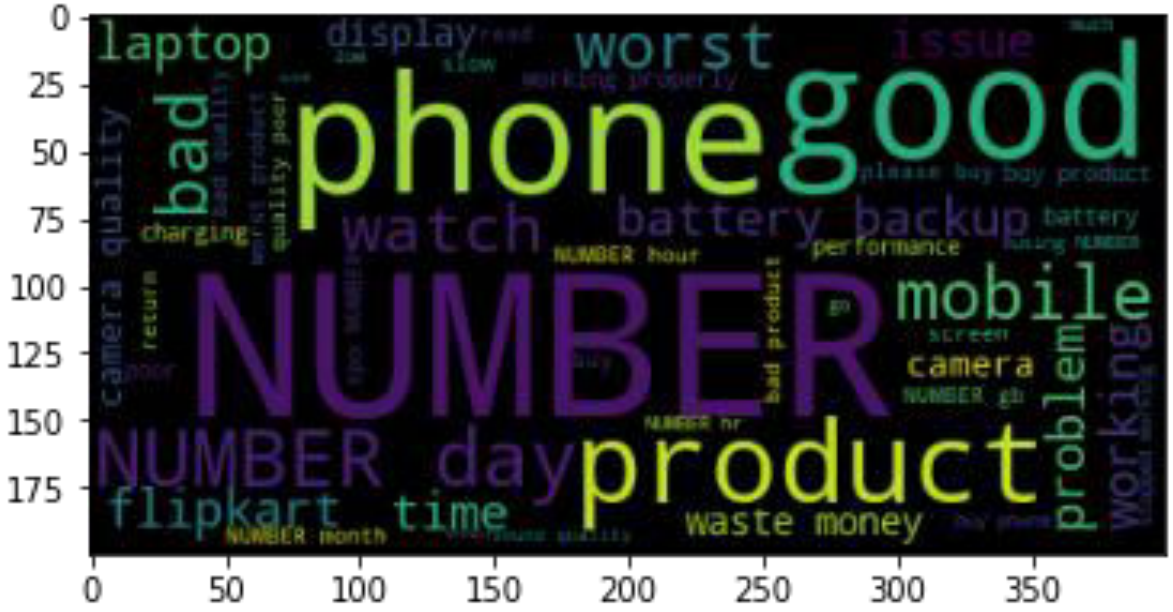
Visualizations

Word Clouds -

1 Rating

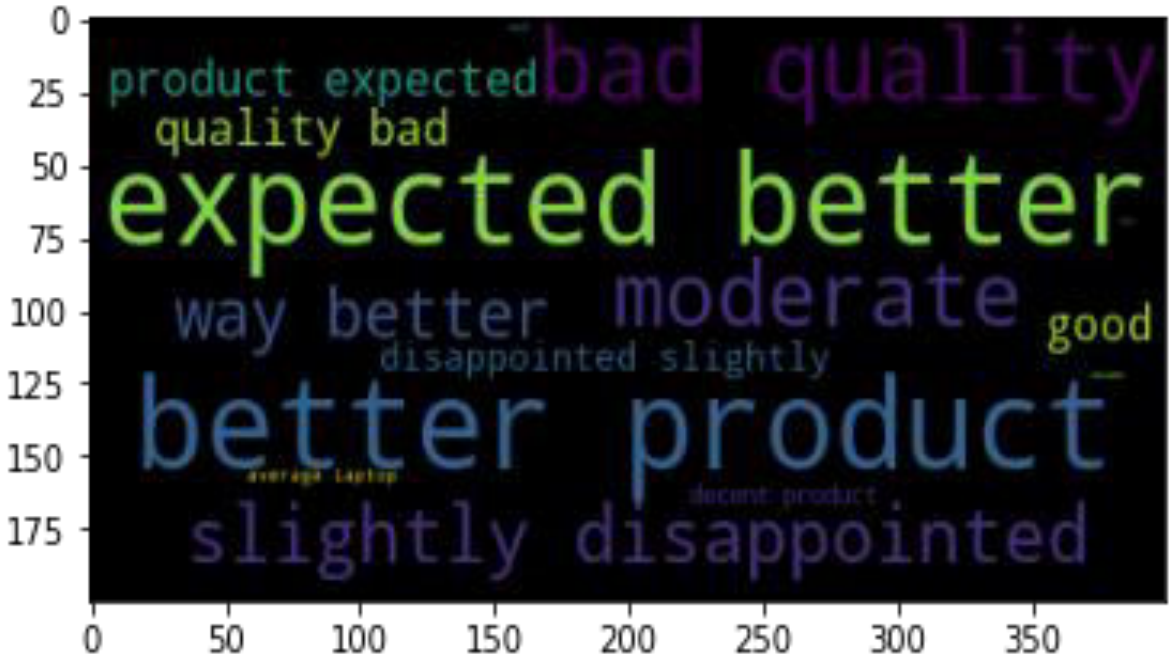
Review Title

Review Description

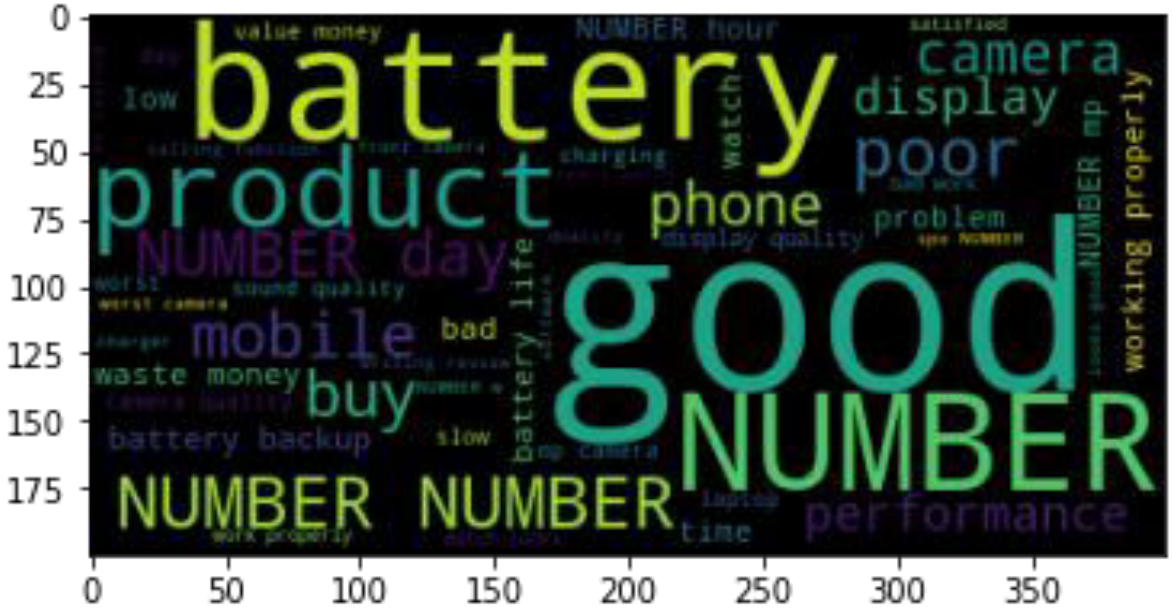


2 ratings

Review Title

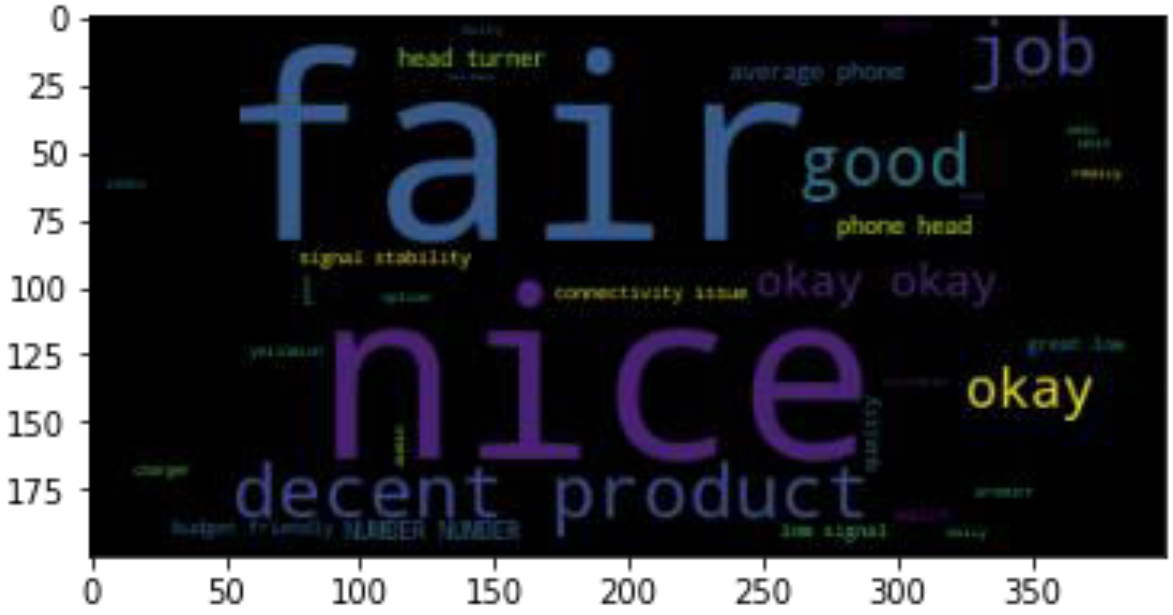


Review Description

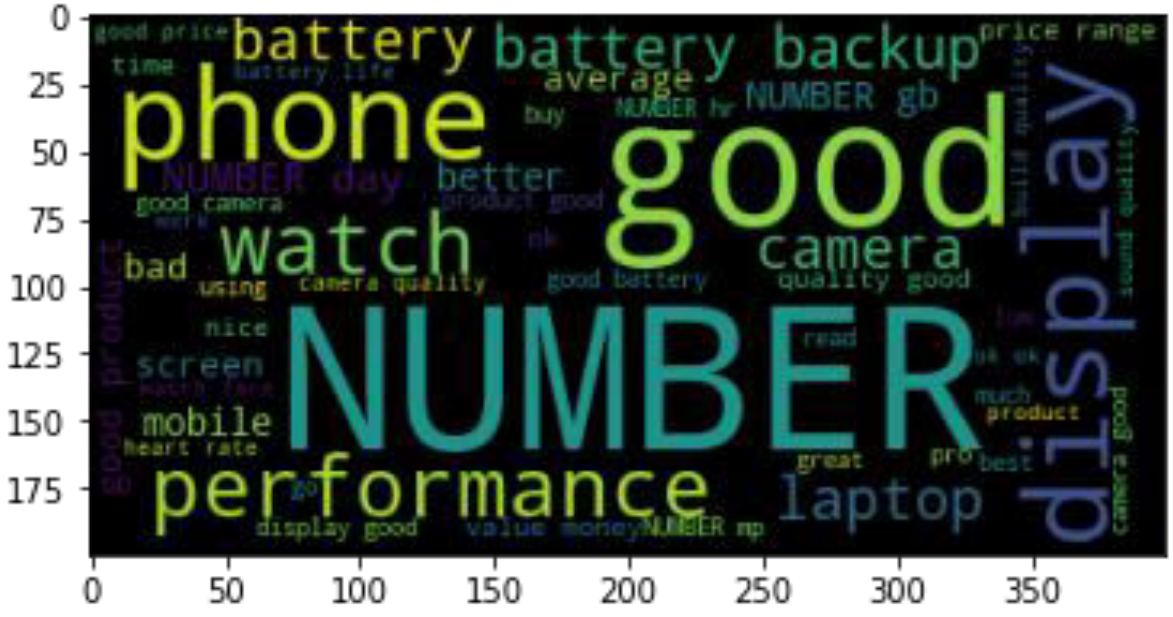


3 ratings

Review Title

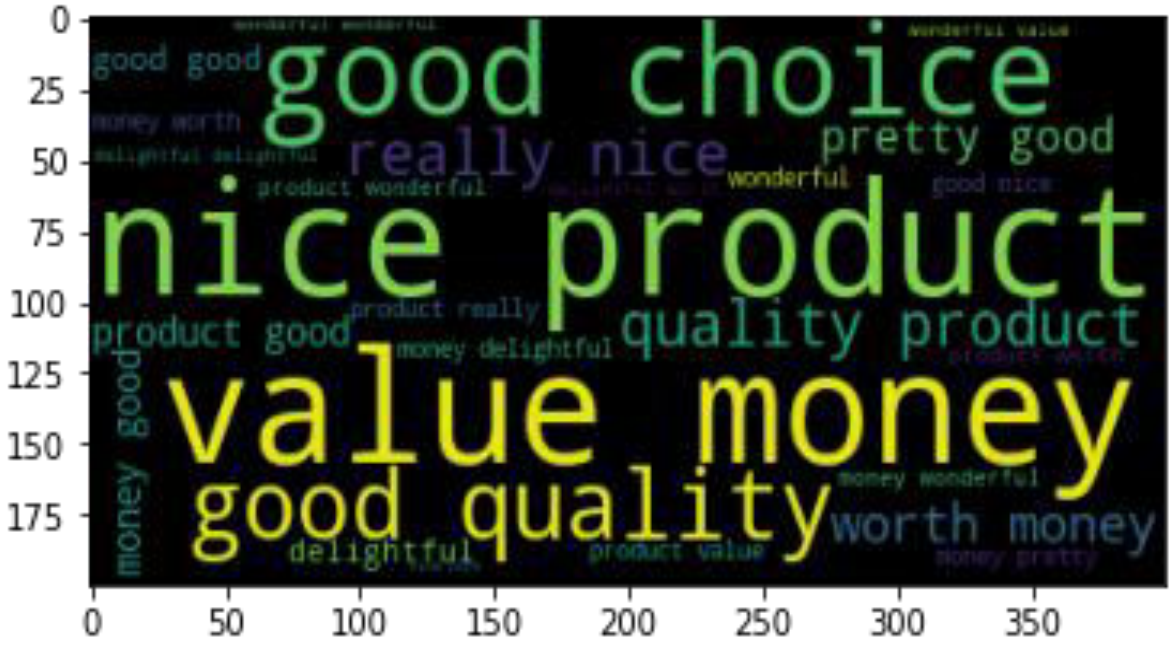


Review Description

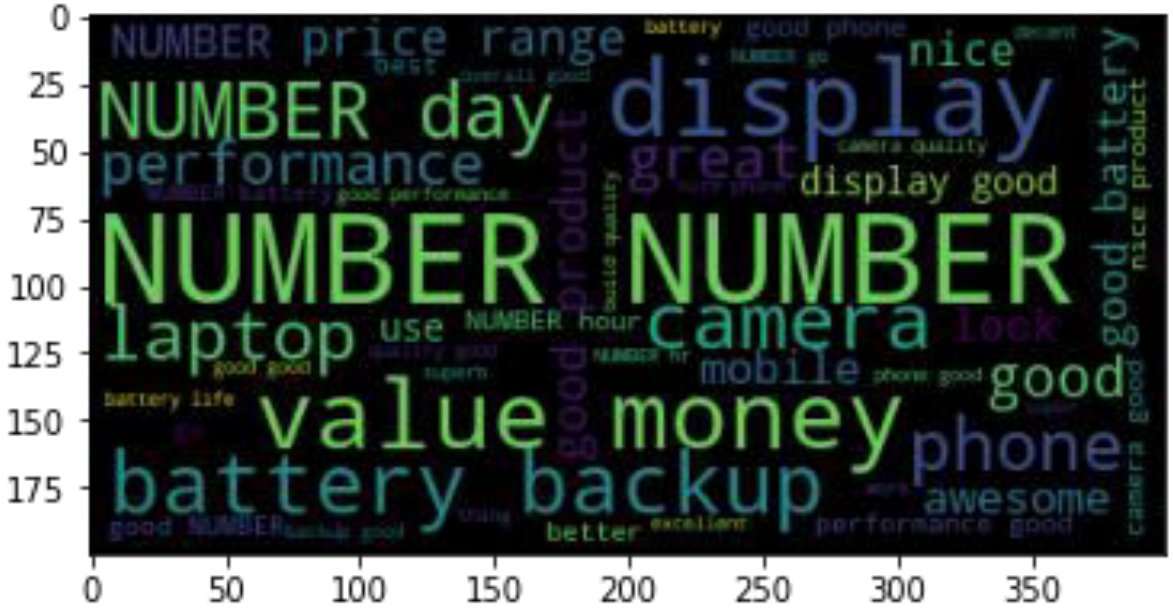


4 ratings

Review Title

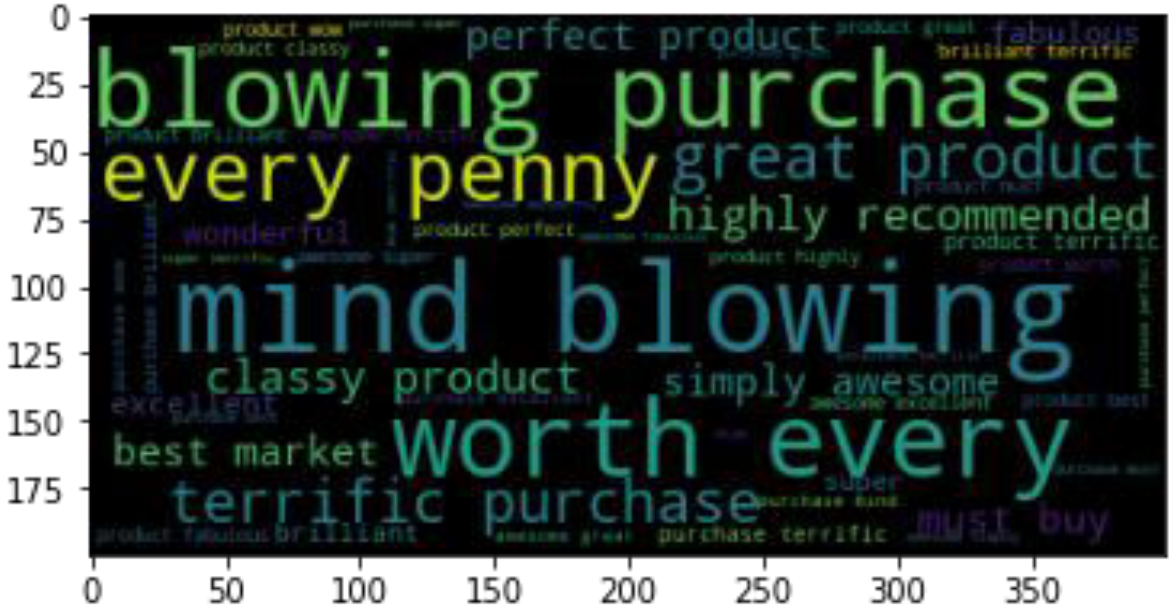


Review Description

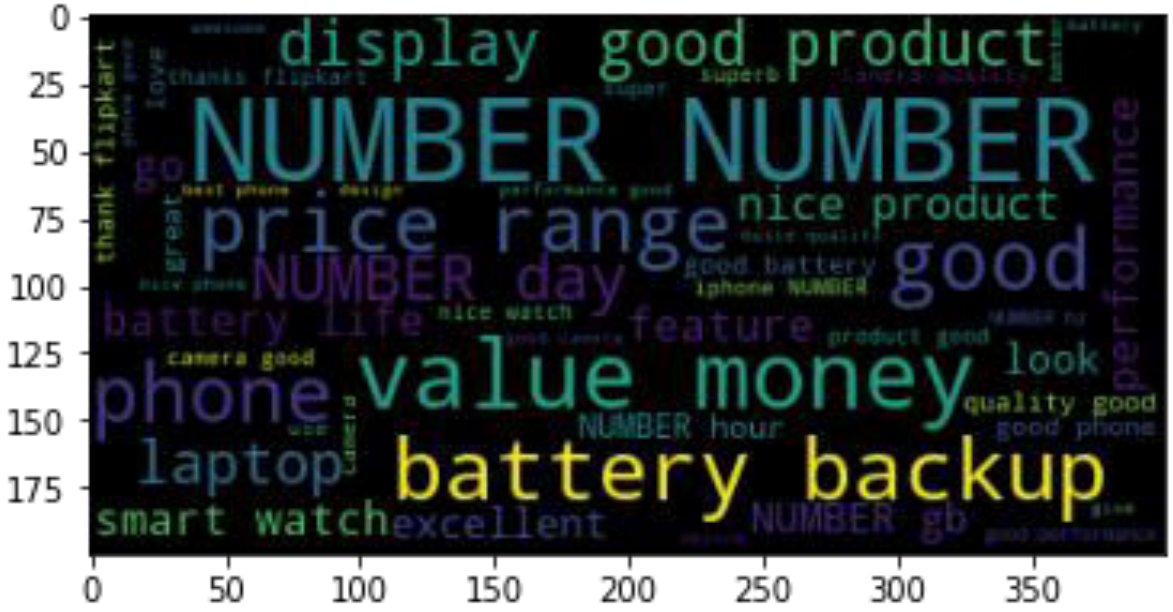


5 ratings

Review title



Review Description



Conclusion

Key Findings and Conclusions of the Study

First, we scrap Reviews and Ratings data of different technical products from flipkart.com using selenium web driver. After scrapping, we save this dataset in a csv file named “flipkartreviews.csv”. Then we pre-process data by cleaning duplicates, noise and unnecessary words which are not helpful for this project. Then we convert text into vectors by using tfidfvectorizer as we know our ml model understands only integers. We choose our best random_state. Then we build our model. Though our dataset is imbalanced, we clearly see that f1 score of SVC is good compared to all other algorithms. So, we check croos_val_score of this model to check if it is overfitting or not. We confirm and conclude that this is the best model. We performed hyperparameter tuning by using GridSearchCV. We save the model by using joblib

Limitations of this work and Scope for Future Work

As we know data is increasing in every second in our day today life. So more the data better the model. If we make this dataset for sentiment analysis, we choose ratings 3 or more as our threshold for being helpful reviews or good or positive reviews and below 3 we choose reviews is not helpful or bad reviews or negative reviews. For example: two people give their reviews on a product as ‘a nice product. But their ratings are ‘4’ and ‘5’ respectively. This is where the model fails to predict whether to choose rating ‘4’ or’5’. Due to increase in data in our daily basics, this model can be used to predict ratings of reviews. It might be a good tool for online shopping sites and manufacturer companies who may predict their customer’s ratings so that they can make their investment according to the demand of customers, which might help them to save time and earn more profits.