FLIP ROBO

RATING PREDICTION

Submitted by:

P. MANIVANNAN

INTRODUCTION

Problem Statement:

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.

- Conceptual Background of the Domain Problem

As the client adding this new Rating Feature. We need NLP to process the existing reviews to predict their Ratings.

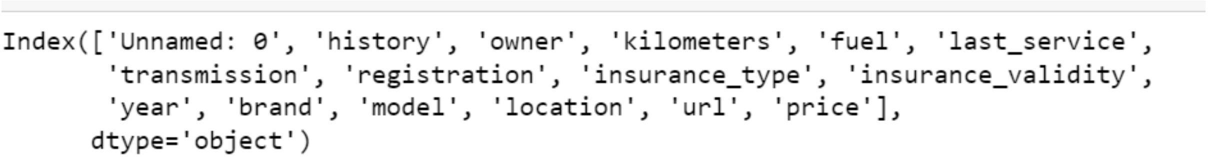
- Review of Literature

The Research is done on online User reviews for various products and how they can be used to predict the ratings for Reviews. First Brief Exploratory Data Analysis is done like Handling Null values and data pre-processing is done using Regex. Then Data is then Fed to Machine Learning Algorithms. after Evaluating and Testing Different Algorithms Best Algorithm is Selected. Based on Evaluation Metrics such as Accuracy score, Cross Validation Score, Confusion Matrix etc. then Hyper Parameter tuning is done on best model using Grid Search CV.

Analytical Problem Framing

- Data Sources and their formats

The data contains 5610 rows and 16 columns. The columns are



```
Index(['Unnamed: 0', 'history', 'owner', 'kilometers', 'fuel', 'last_service',
       'transmission', 'registration', 'insurance_type', 'insurance_validity',
       'year', 'brand', 'model', 'location', 'url', 'price'],
      dtype='object')
```

The Various Product reviews data is collected from Amazon website using web scrapping. Data is collected from different for Different Products and Combined.

The data is in Csv format.

- Data Pre-processing Done
  1) The data has null values which is removed using dropna method of pandas.
  2) Since machine learning algoriths are case sensitive we are converging all to lower cases
  3) Removing Emails and replacing them with word "emailaddress".
  4) Removing Website address and replacing them with word "webaddress".
  5) Removing Phone Numbers and replacing them with word "phonenumber".
  6) Removing Currency Symbols like Dollars and Pounds with word "dollars".
  7) Removing Numbers with word "numbr".
  8) Removing Punctuations.
  9) Removing Whitespaces and removing stop words

- State the set of assumptions (if any) related to the problem under consideration
  1) For Handling the Null values . Null values are dropped considering they will not affect the prediction.
  2) Pre-processing is done using Regex

- Hardware and Software Requirements and Tools Used

The Hardware requirements are 16 Gb RAM,500 GB SSD, at least 6 Cores processor

The software requirements are Windows, Anaconda Framework and libraries used are Pandas, NumPy, Sklearn, SciPy, Matplotlib, Seaborn

We use Pandas for reading the Dataset, Creating Dataframe, much more to handle data in dataset

Certain functions of NumPy are used like log , cbrt ,sqrt skewness transformation . absolute (abs) function used along with Zscore.

SKlearn Library is used for Machine Learning Algorithms like SVC, Random Forest, Linear Regression etc and Metrics for analysing them.

Scipy is scientific python library used for Zscore method etc.

Matplotlib and Seaborn are used for visualizations plotting graphs charts, etc

Finally joblib module is used for saving our model for future use.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  Since the Target Variable is Continuous in nature I should Use Regression Approach to Solve this Problem and build a Regression Model

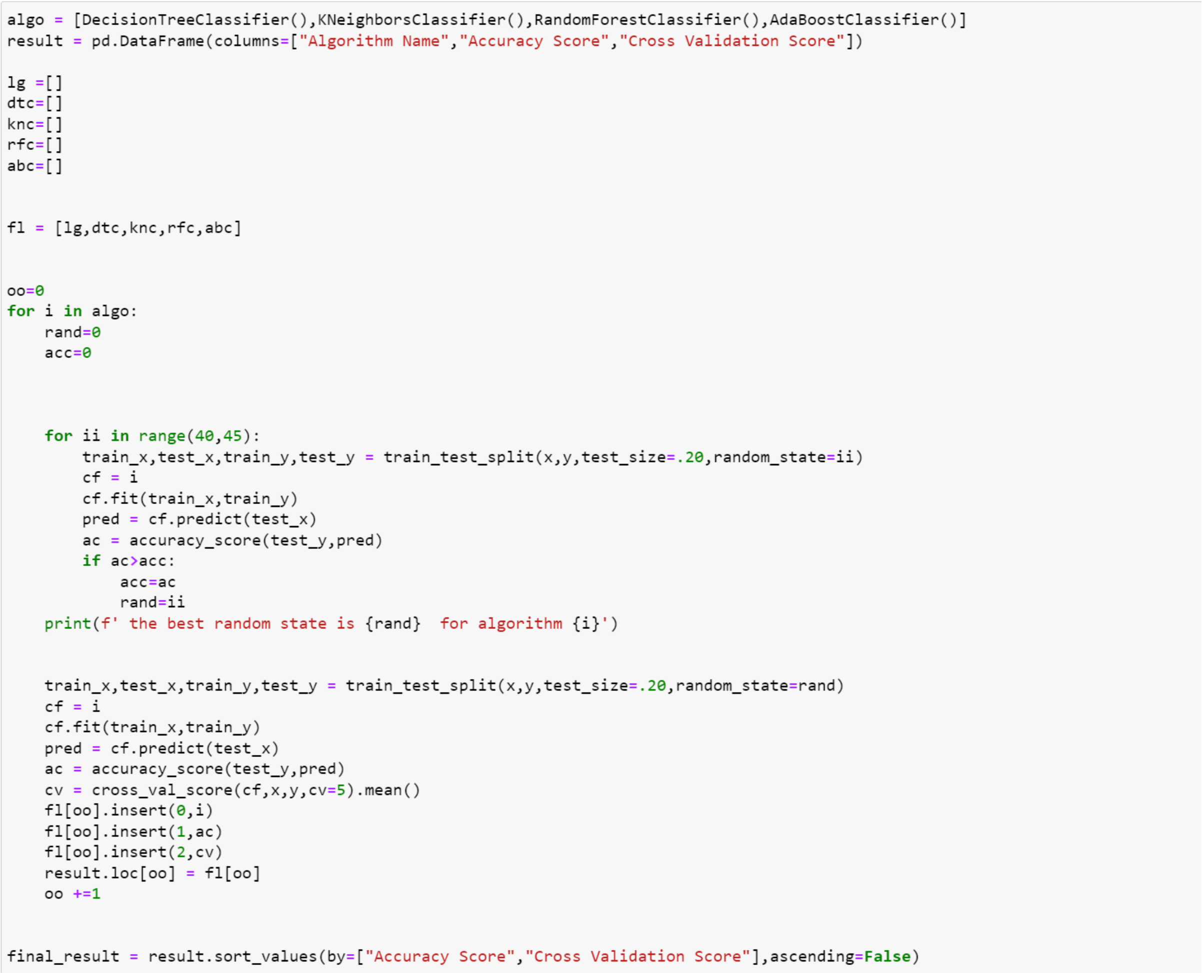
- Testing of Identified Approaches (Algorithms)
  1) Decision Tree Classifier
  2) K Neighbors Classifier
  3) Random Forest Classifier
  4) Ada Boost Classifier
  5) Multinomial NB

- Run and Evaluate selected models

  I have custom written a Code Block for Running all Algorithms and Storing them in DataFrame and Ranking them based on their Accuracy Score. The best part is the code finds the best random state for each algorithm and prints them

Code:

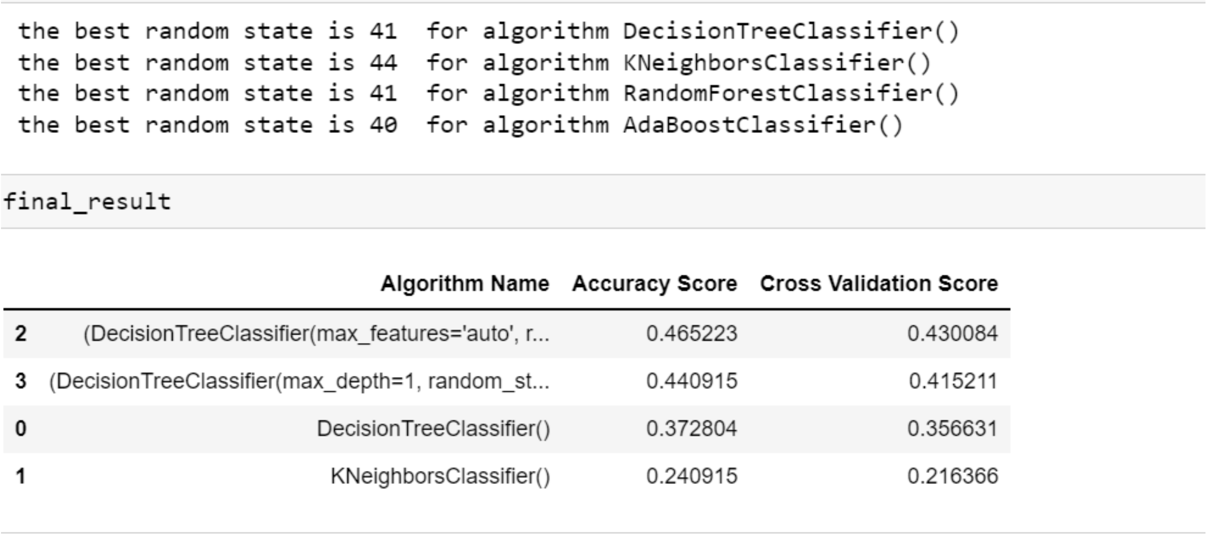
This code block is written by me which ranks the best alogrithm



```
algo = [DecisionTreeClassifier(),KNeighborsClassifier(),RandomForestClassifier(),AdaBoostClassifier()]
result = pd.DataFrame(columns=["Algorithm Name","Accuracy Score","Cross Validation Score"])

lg =[]
dtc=[]
knc=[]
rfc=[]
abc=[]


fl = [lg,dtc,knc,rfc,abc]


oo=0
for i in algo:
    rand=0
    acc=0



    for ii in range(40,45):
        train_x,test_x,train_y,test_y = train_test_split(x,y,test_size=.20,random_state=ii)
        cf = i
        cf.fit(train_x,train_y)
        pred = cf.predict(test_x)
        ac = accuracy_score(test_y,pred)
        if ac>acc:
            acc=ac
            rand=ii
    print(f' the best random state is {rand}  for algorithm {i}')


    train_x,test_x,train_y,test_y = train_test_split(x,y,test_size=.20,random_state=rand)
    cf = i
    cf.fit(train_x,train_y)
    pred = cf.predict(test_x)
    ac = accuracy_score(test_y,pred)
    cv = cross_val_score(cf,x,y,cv=5).mean()
    fl[oo].insert(0,i)
    fl[oo].insert(1,ac)
    fl[oo].insert(2,cv)
    result.loc[oo] = fl[oo]
    oo +=1


final_result = result.sort_values(by=["Accuracy Score","Cross Validation Score"],ascending=False)
```

If you see the code, I have created a list of algorithms which I need to run. And here is the output:

Output1:

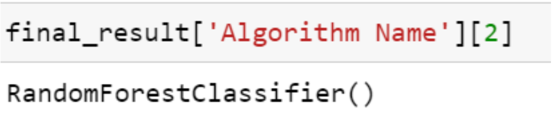


```
the best random state is 41  for algorithm DecisionTreeClassifier()
the best random state is 44  for algorithm KNeighborsClassifier()
the best random state is 41  for algorithm RandomForestClassifier()
the best random state is 40  for algorithm AdaBoostClassifier()
```

```
final_result
```

	Algorithm Name	Accuracy Score	Cross Validation Score
2	(DecisionTreeClassifier(max_features='auto', r...	0.465223	0.430084
3	(DecisionTreeClassifier(max_depth=1, random_st...	0.440915	0.415211
0	DecisionTreeClassifier()	0.372804	0.356631
1	KNeighborsClassifier()	0.240915	0.216366

Output2:



```
final_result['Algorithm Name'][2]
```

```
RandomForestClassifier()
```

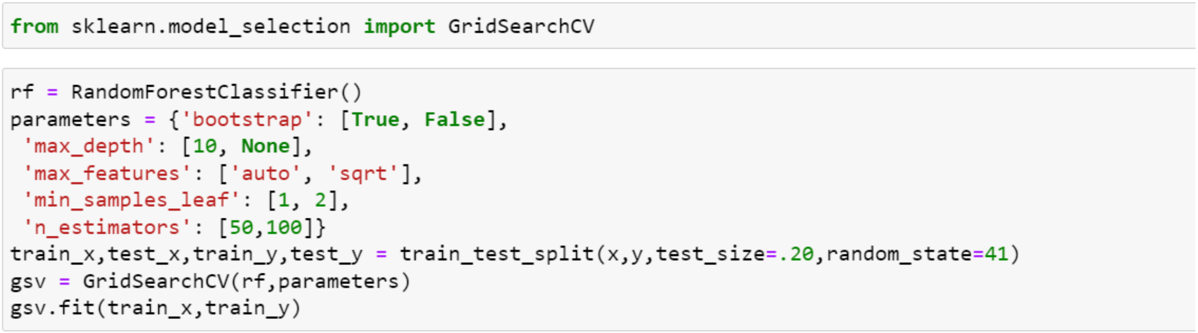
Here you can see the Model Name, Accuracy Score and Cross Validation Score.

Based on this best model is selected and further proceeded for hyperparameter tuning. Here Ridge Regression performed best compared to others.

Here Random Forest Classifier performed best

Hyperparameter Tuning

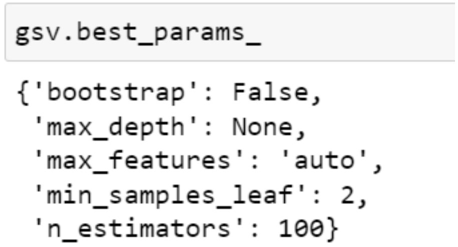
Code:



```
from sklearn.model_selection import GridSearchCV
```

```
rf = RandomForestClassifier()
parameters = {'bootstrap': [True, False],
 'max_depth': [10, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2],
 'n_estimators': [50,100]}
train_x,test_x,train_y,test_y = train_test_split(x,y,test_size=.20,random_state=41)
gsv = GridSearchCV(rf,parameters)
gsv.fit(train_x,train_y)
```

Output:

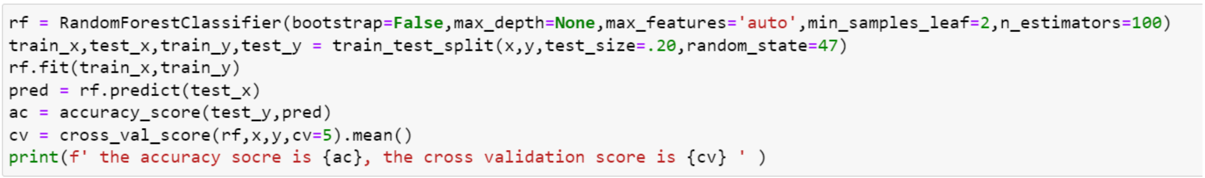


```
gsv.best_params_
```

```
{'bootstrap': False,
 'max_depth': None,
 'max_features': 'auto',
 'min_samples_leaf': 2,
 'n_estimators': 100}
```

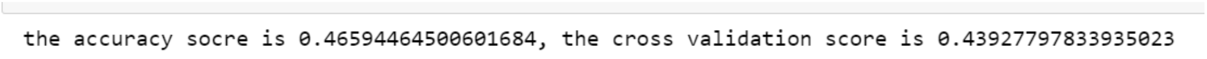
The Best Parameters are listed now we need to run the model with the best parameters

Code:



```
rf = RandomForestClassifier(bootstrap=False,max_depth=None,max_features='auto',min_samples_leaf=2,n_estimators=100)
train_x,test_x,train_y,test_y = train_test_split(x,y,test_size=.20,random_state=47)
rf.fit(train_x,train_y)
pred = rf.predict(test_x)
ac = accuracy_score(test_y,pred)
cv = cross_val_score(rf,x,y,cv=5).mean()
print(f' the accuracy socre is {ac}, the cross validation score is {cv} ' )
```

Output:



```
 the accuracy socre is 0.46594464500601684, the cross validation score is 0.43927797833935023
```

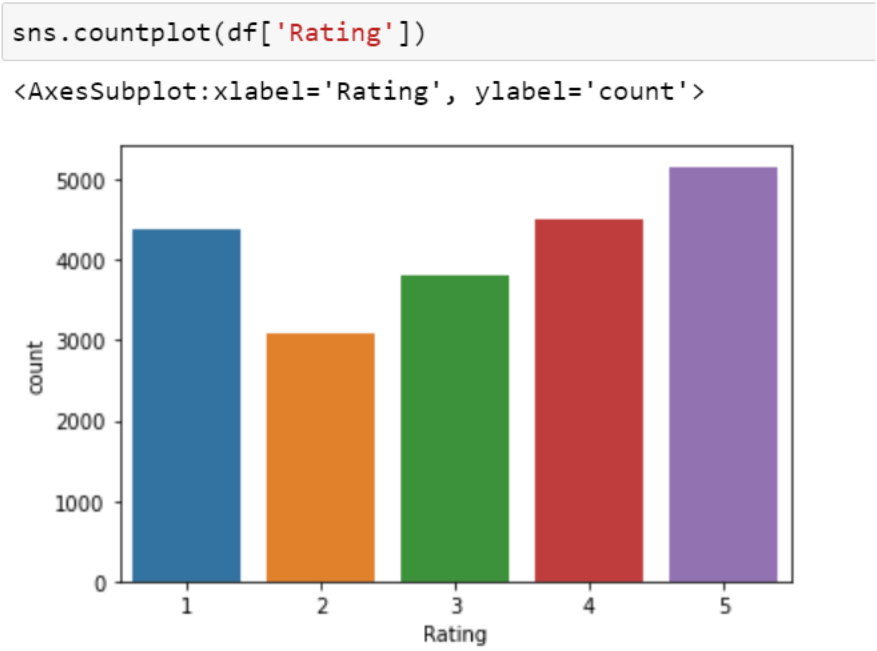
The Best Accuracy Score for Random Forest Classifier After Hyper parameter tuning for Training is 46% and Cross Validation Score is 44%

- ## Key Metrics for success in solving problem under consideration

Since it’s a Classification problem The key metrics used are Accuracy Score, Cross Validation score, confusion matrix, Classification Report

Visualizations

For visualizations the Matplotlib & Seaborn Library are used.

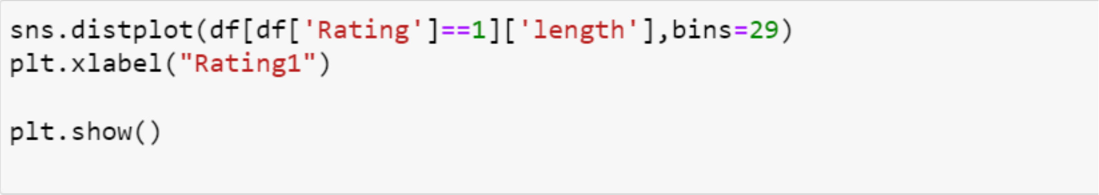


```
sns.countplot(df['Rating'])
```

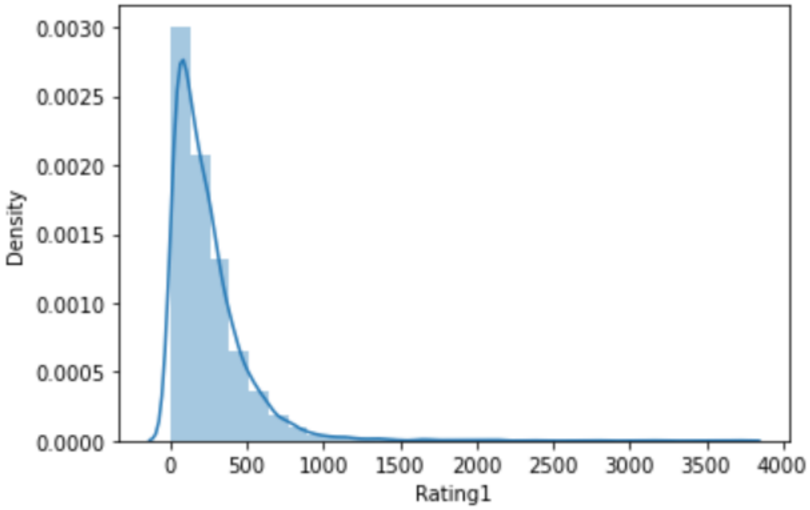
```
<AxesSubplot:xlabel='Rating', ylabel='count'>
```

Observation:

Ratings are almost balanced

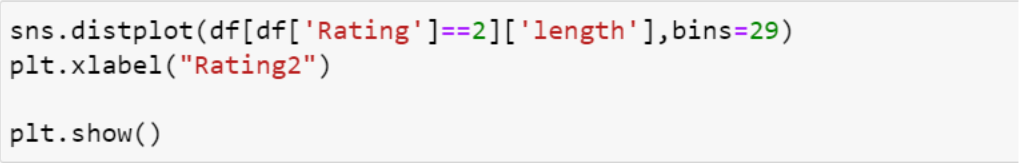


```
sns.distplot(df[df['Rating']==1]['length'],bins=29)
plt.xlabel("Rating1")

plt.show()
```

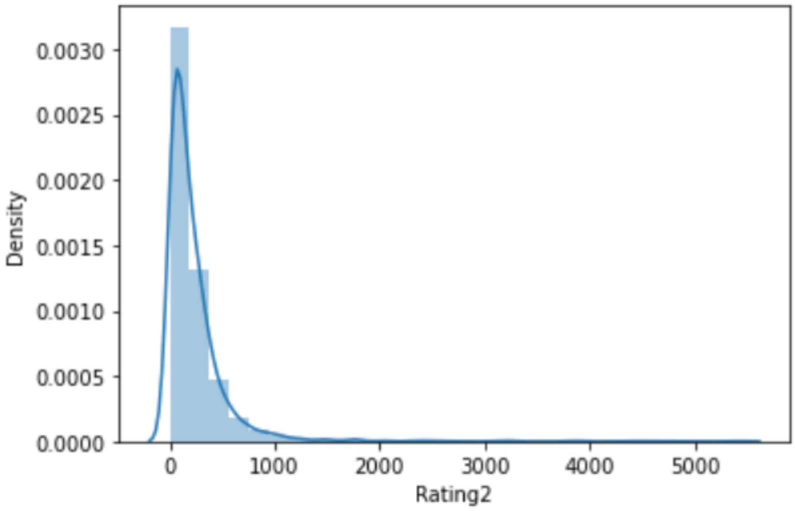


Observation:

Length of Rating 1 before Data cleaning

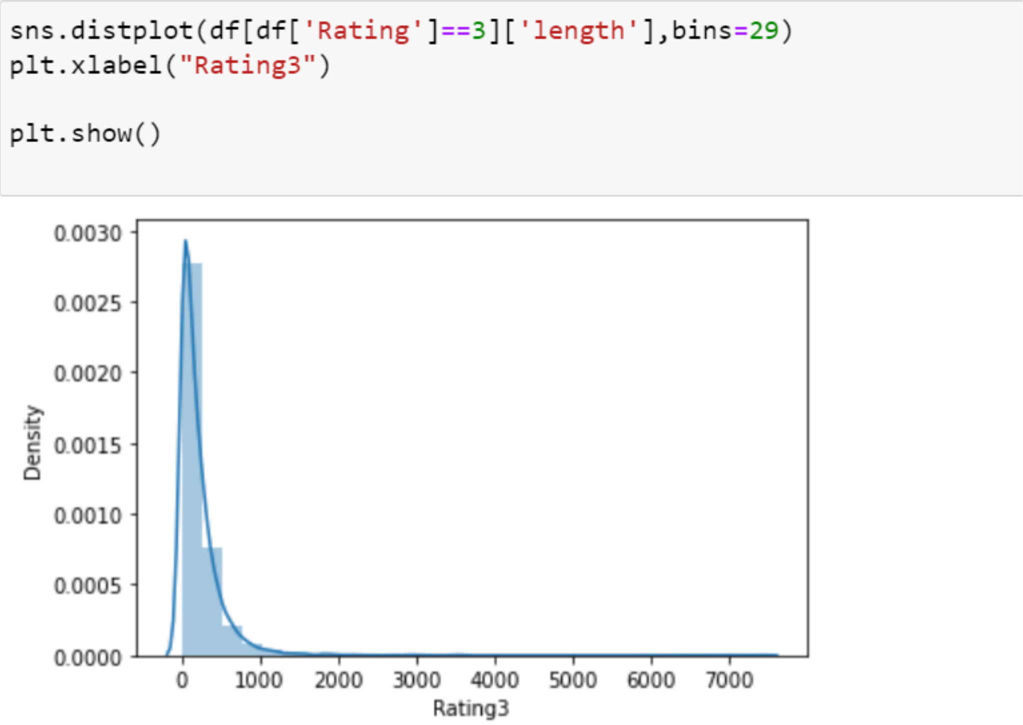


```
sns.distplot(df[df['Rating']==2]['length'],bins=29)
plt.xlabel("Rating2")

plt.show()
```



Observation:

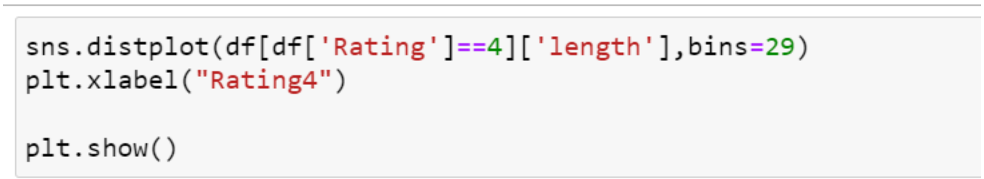
Length of Rating 2 before Data cleaning



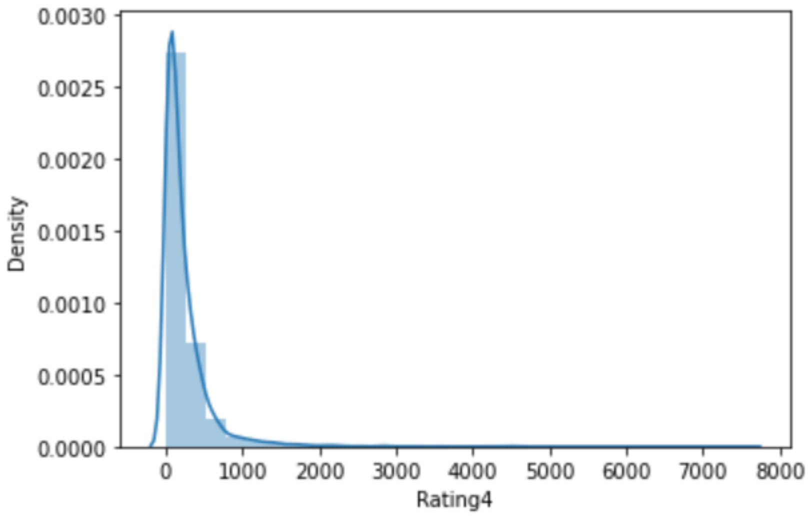
```
sns.distplot(df[df['Rating']==3]['length'],bins=29)
plt.xlabel("Rating3")

plt.show()
```

Observation:

Length of Rating 3 before Data Cleaning

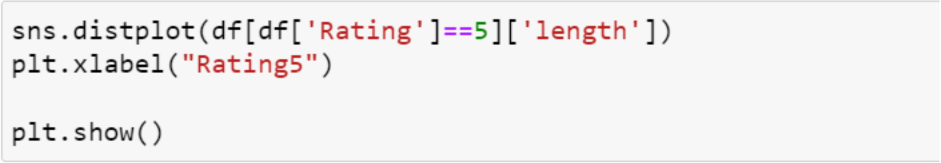


```
sns.distplot(df[df['Rating']==4]['length'],bins=29)
plt.xlabel("Rating4")

plt.show()
```

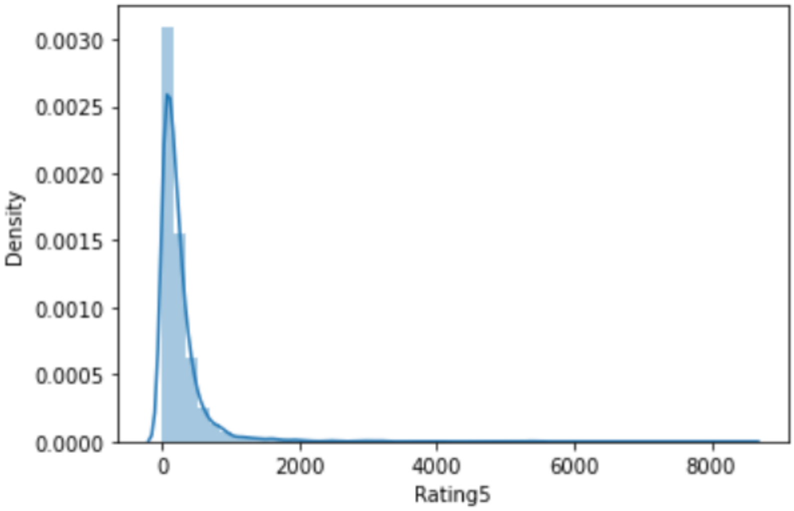


Observation:

Length of Rating 4 Before Data Cleaning

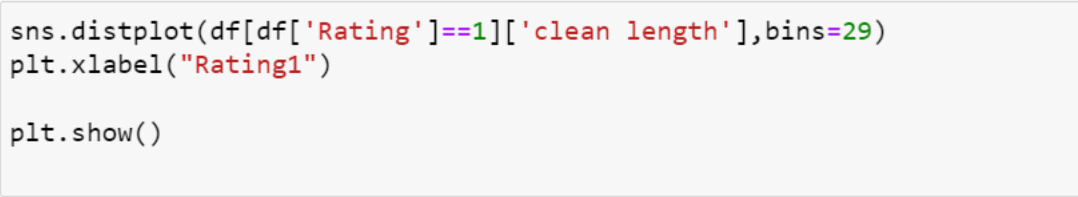


```
sns.distplot(df[df['Rating']==5]['length'])
plt.xlabel("Rating5")

plt.show()
```

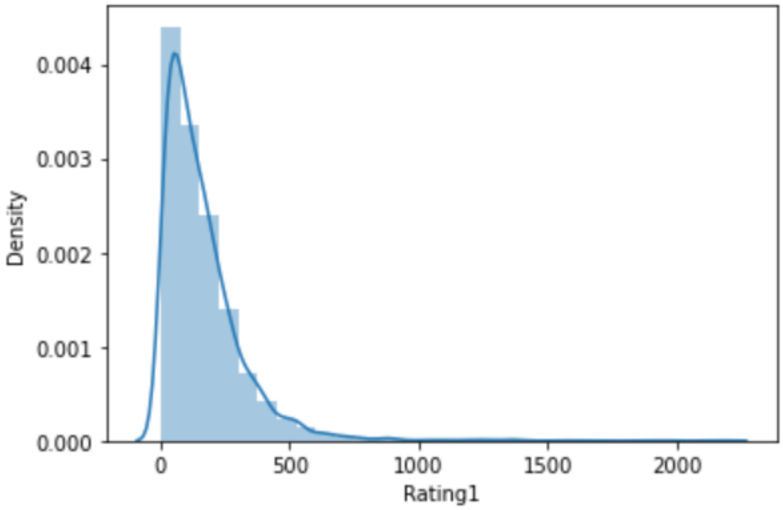


Observation:

Length of Rating 5 Before Data Cleaning

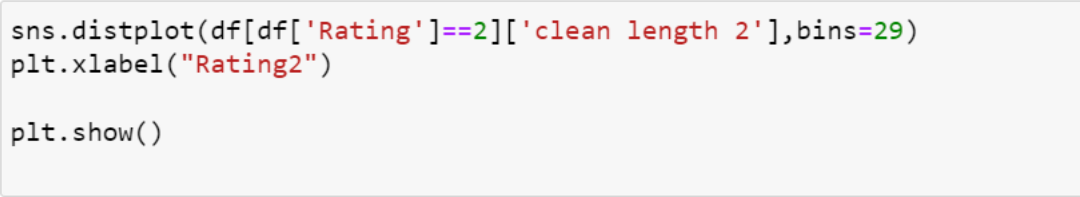


```
sns.distplot(df[df['Rating']==1]['clean length'],bins=29)
plt.xlabel("Rating1")

plt.show()
```

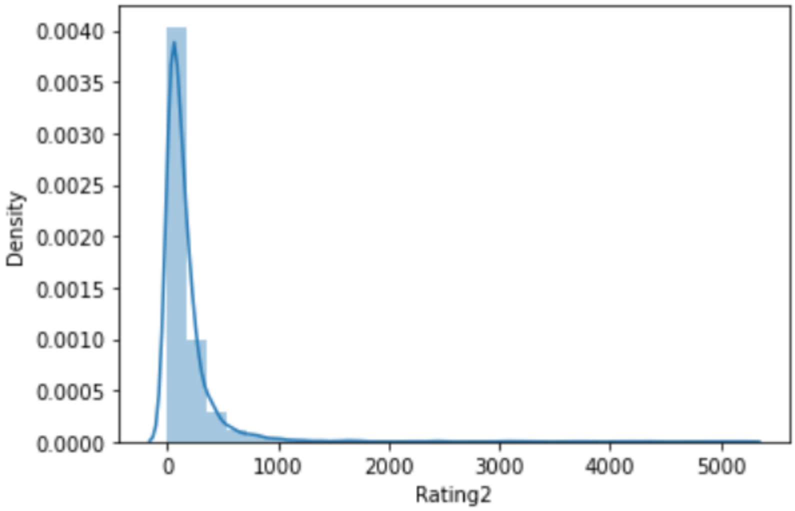


Observation:

Length of Rating 1 After Data Cleaning

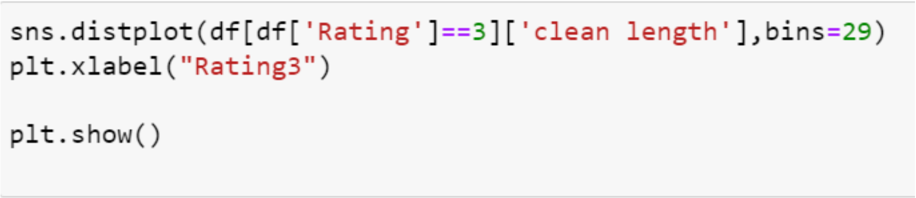


```
sns.distplot(df[df['Rating']==2]['clean length 2'],bins=29)
plt.xlabel("Rating2")

plt.show()
```

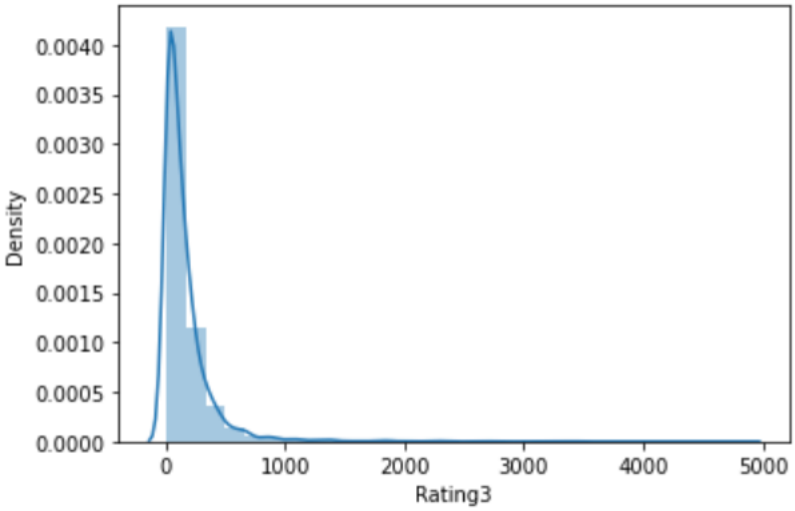


Observation:

Length of Rating 2 After Data Cleaning

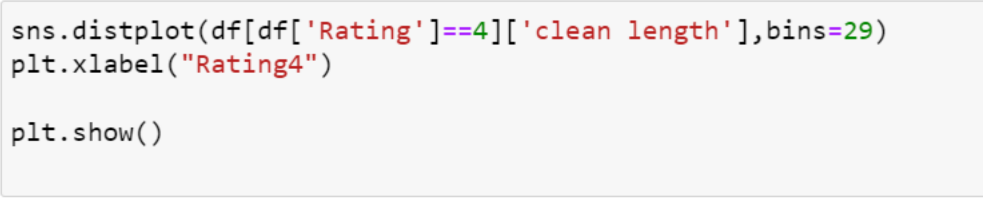


```
sns.distplot(df[df['Rating']==3]['clean length'],bins=29)
plt.xlabel("Rating3")

plt.show()
```

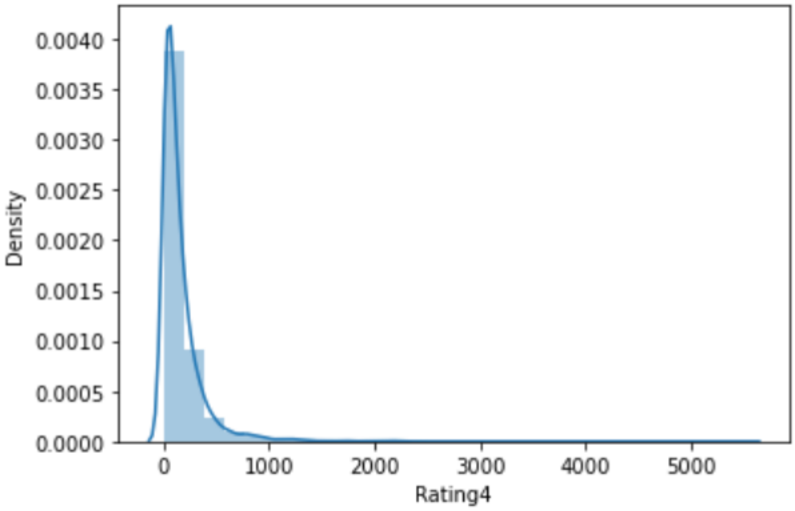


Observation:

Length of Rating 3 after Data Cleaning

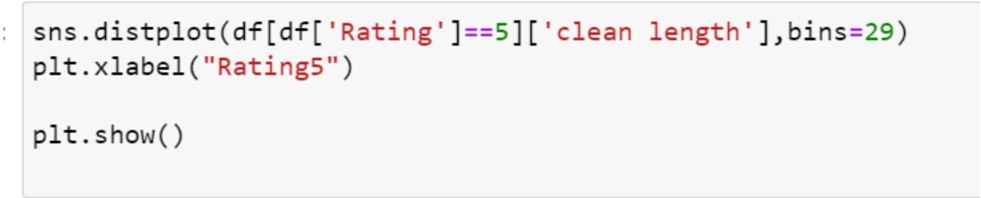


```
sns.distplot(df[df['Rating']==4]['clean length'],bins=29)
plt.xlabel("Rating4")

plt.show()
```

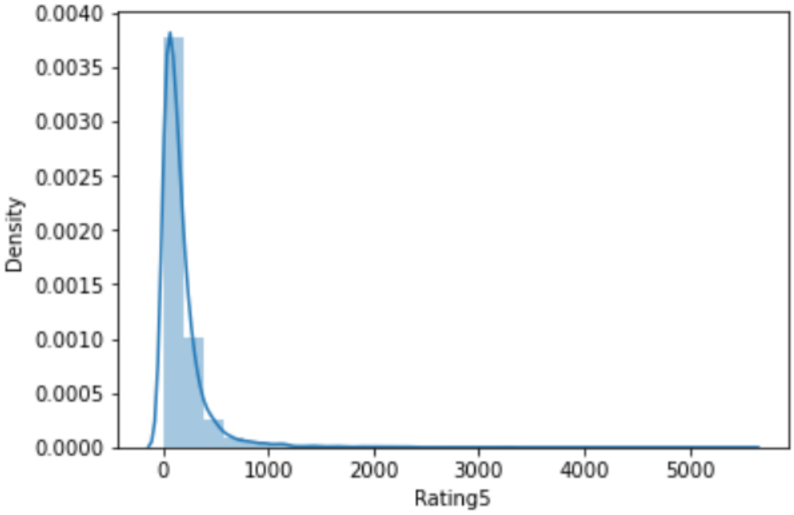


Observation:

Length of Rating 4 after Data Cleaning

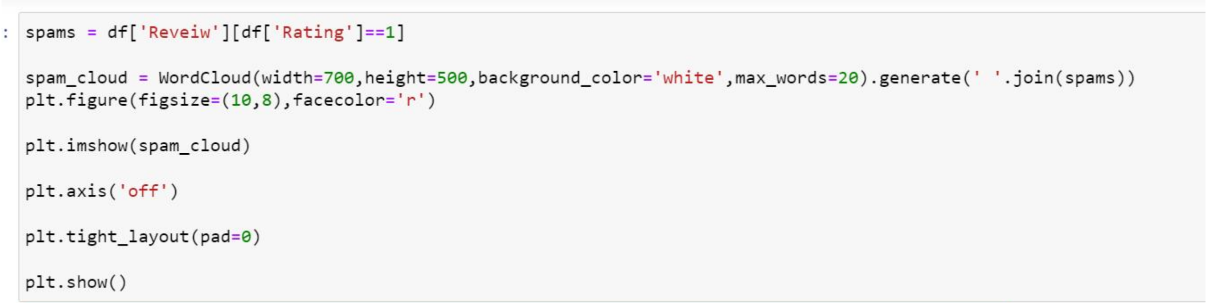


```
sns.distplot(df[df['Rating']==5]['clean length'],bins=29)
plt.xlabel("Rating5")

plt.show()
```

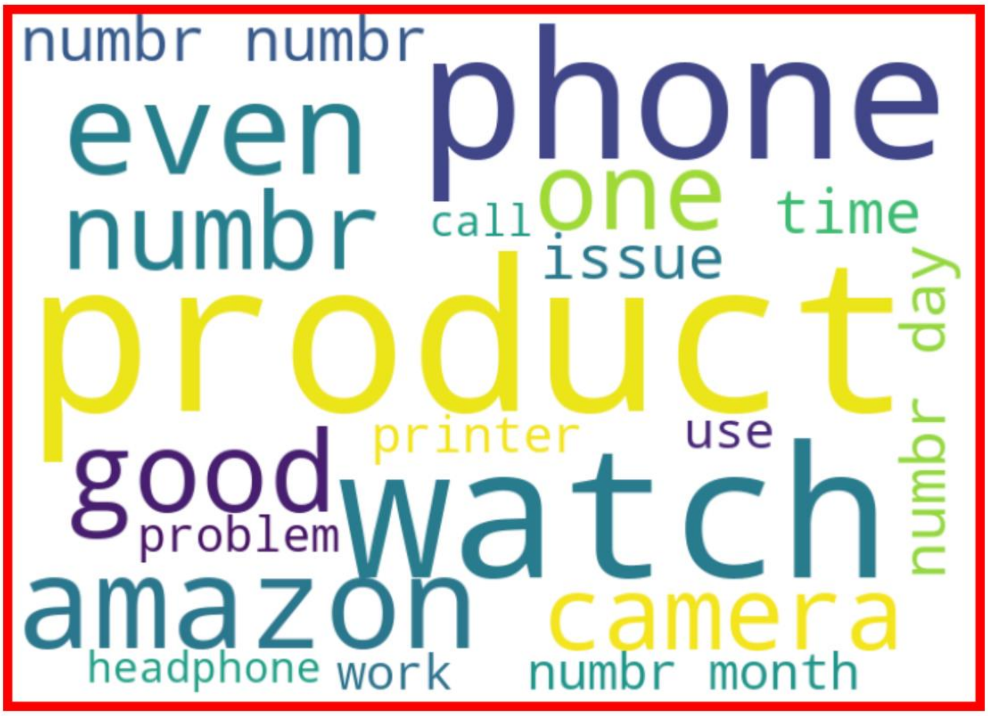


Observation:

Length of Rating 5 After Data Cleaning

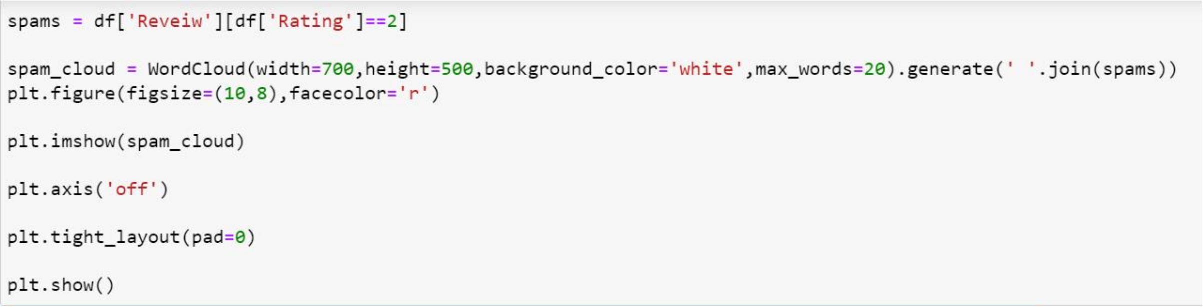


```
spams = df['Reveiw'][df['Rating']==1]

spam_cloud = WordCloud(width=700,height=500,background_color='white',max_words=20).generate(' '.join(spams))
plt.figure(figsize=(10,8),facecolor='r')

plt.imshow(spam_cloud)

plt.axis('off')

plt.tight_layout(pad=0)

plt.show()
```

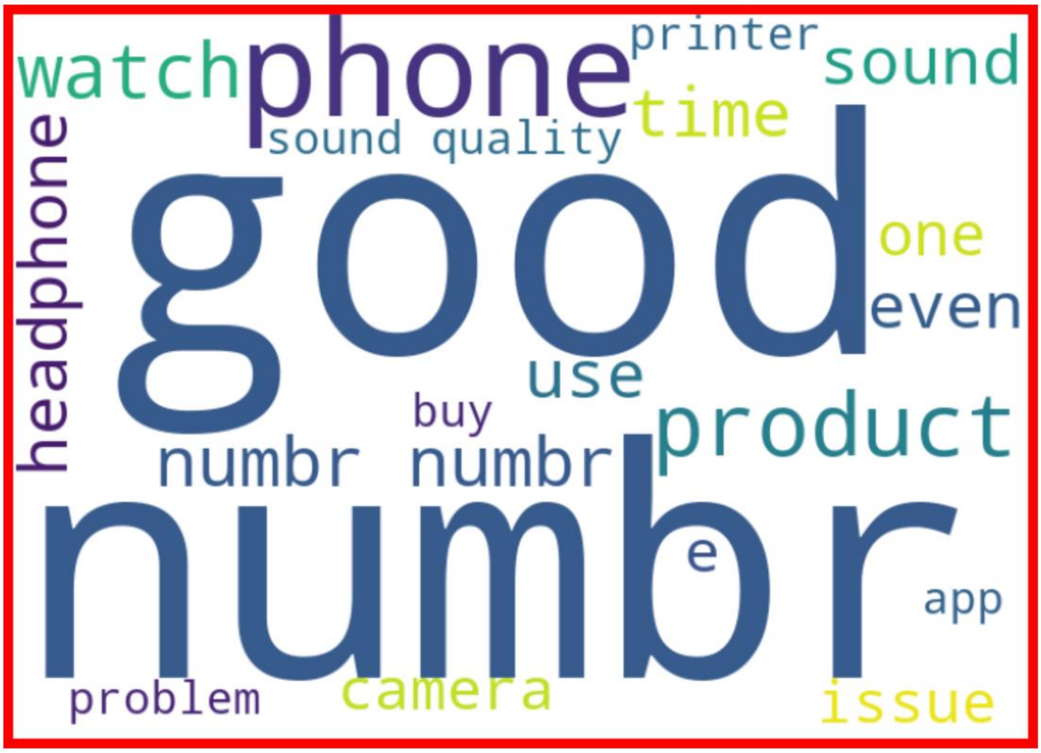


Observation:

Rating 1 Word Cloud Plot. We can see words like issue, problem are present.

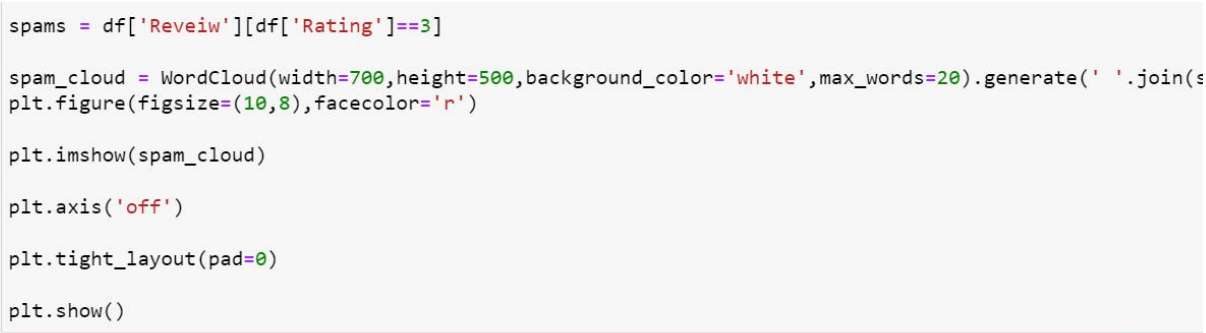


```
spams = df['Reveiw'][df['Rating']==2]

spam_cloud = WordCloud(width=700,height=500,background_color='white',max_words=20).generate(' '.join(spams))
plt.figure(figsize=(10,8),facecolor='r')

plt.imshow(spam_cloud)

plt.axis('off')

plt.tight_layout(pad=0)

plt.show()
```

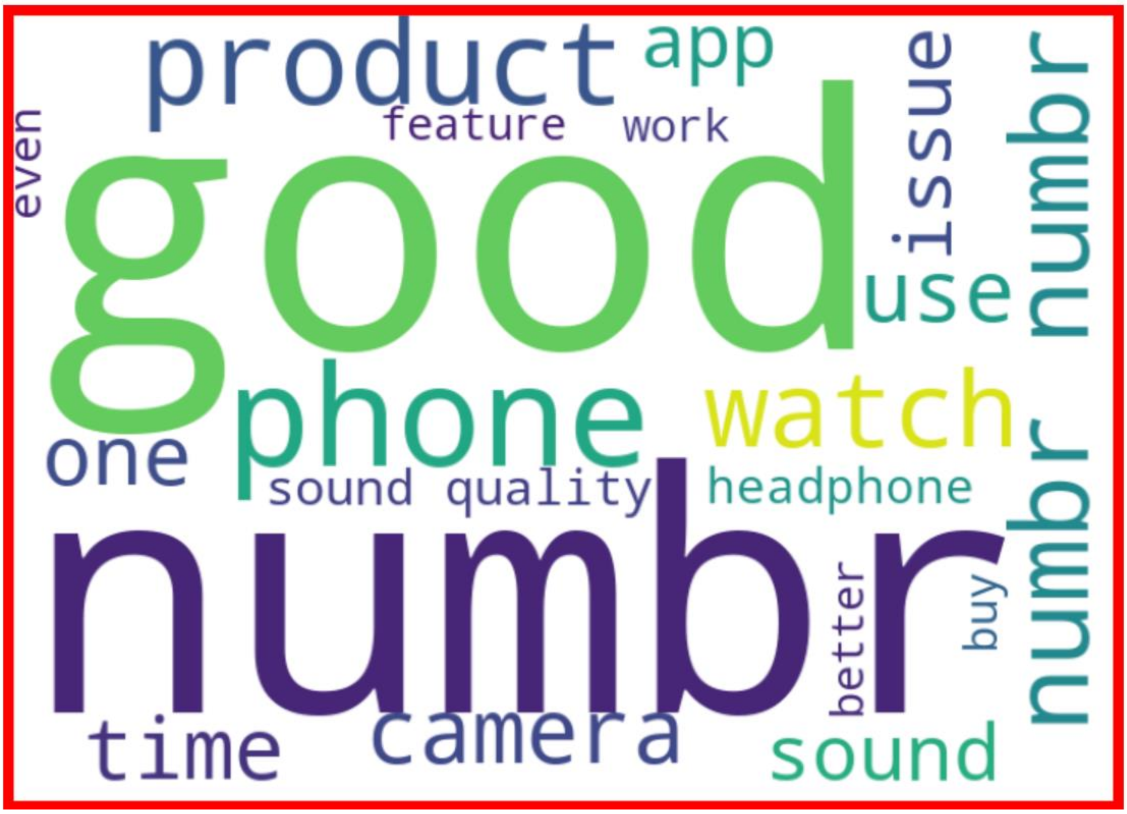


Observation:

Rating 2 Word Cloud Plot. We can see words like issue, problem are present

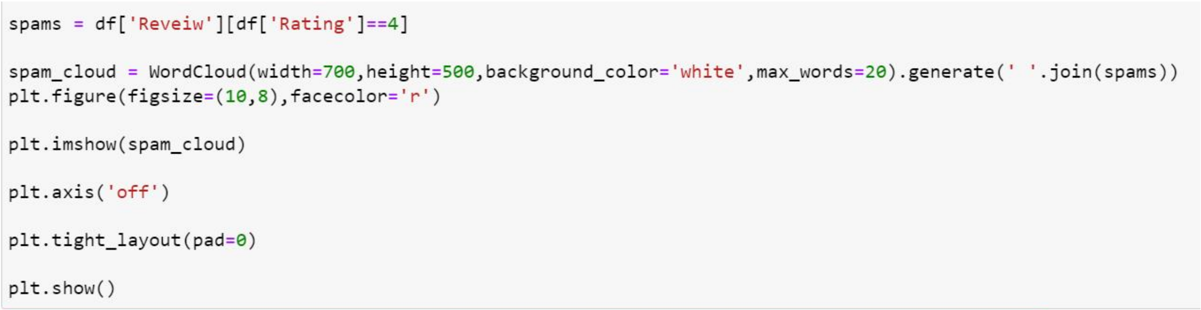


```
spams = df['Reveiw'][df['Rating']==3]

spam_cloud = WordCloud(width=700,height=500,background_color='white',max_words=20).generate(' '.join(s
plt.figure(figsize=(10,8),facecolor='r')

plt.imshow(spam_cloud)

plt.axis('off')

plt.tight_layout(pad=0)

plt.show()
```

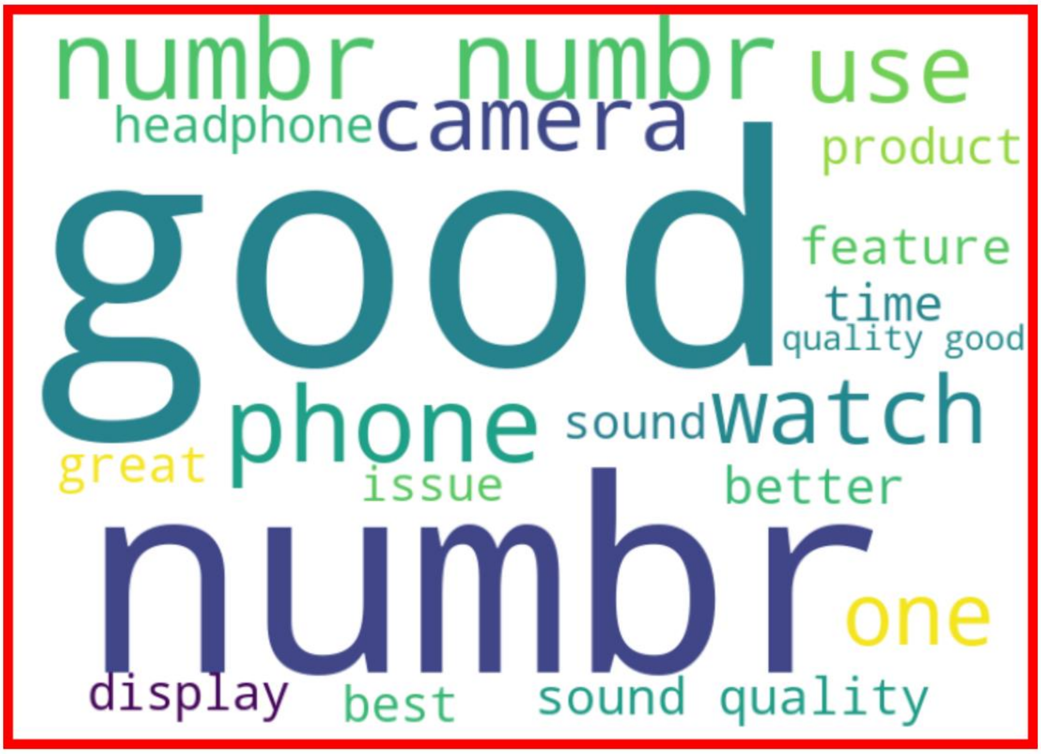


Observation:

Rating 3 Word Cloud Plot. We can see words like better, Even are present

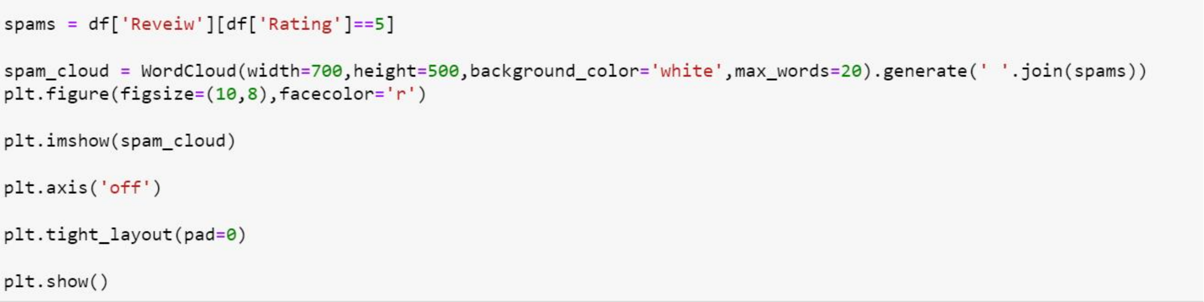


```
spams = df['Reveiw'][df['Rating']==4]

spam_cloud = WordCloud(width=700,height=500,background_color='white',max_words=20).generate(' '.join(spams))
plt.figure(figsize=(10,8),facecolor='r')

plt.imshow(spam_cloud)

plt.axis('off')

plt.tight_layout(pad=0)

plt.show()
```

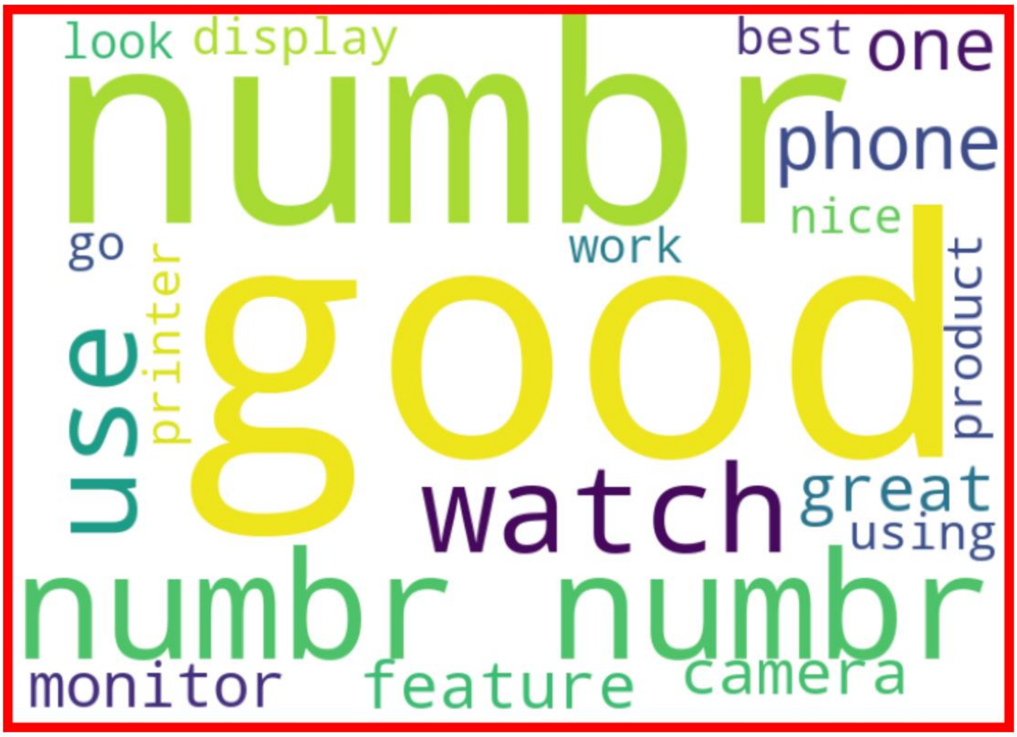


Observation:

Rating 4 Word Cloud Plot. We can see words like best, great, quality good are present



```
spams = df['Reveiw'][df['Rating']==5]

spam_cloud = WordCloud(width=700,height=500,background_color='white',max_words=20).generate(' '.join(spams))
plt.figure(figsize=(10,8),facecolor='r')

plt.imshow(spam_cloud)

plt.axis('off')

plt.tight_layout(pad=0)

plt.show()
```



Observation:

Rating 5 Word Cloud Plot. We can see words like best one, Great, Go is present

- Interpretation of the Results

  Based on the Word Cloud plot, we can see some words influence the Rating Prediction. Words like issue, problem is in lower ratings. And the words like Best one, great, quality good are in higher ratings. So, these words clearly influence the Rating prediction in NLP.

CONCLUSION

- Key Findings and Conclusions of the Study

  We can clearly see that Certain words affect the rating prediction of the NLP Program.

- Learning Outcomes of the Study in respect of Data Science

  We can learn many useful insights from this study. the words people most often use to criticize as well as praise the products. So based on these words NLP process the Rating Prediction.

- Limitations of this work and Scope for Future Work

  The Limitations are Data collection are less. in future massive dataset is needed to get more accurate model