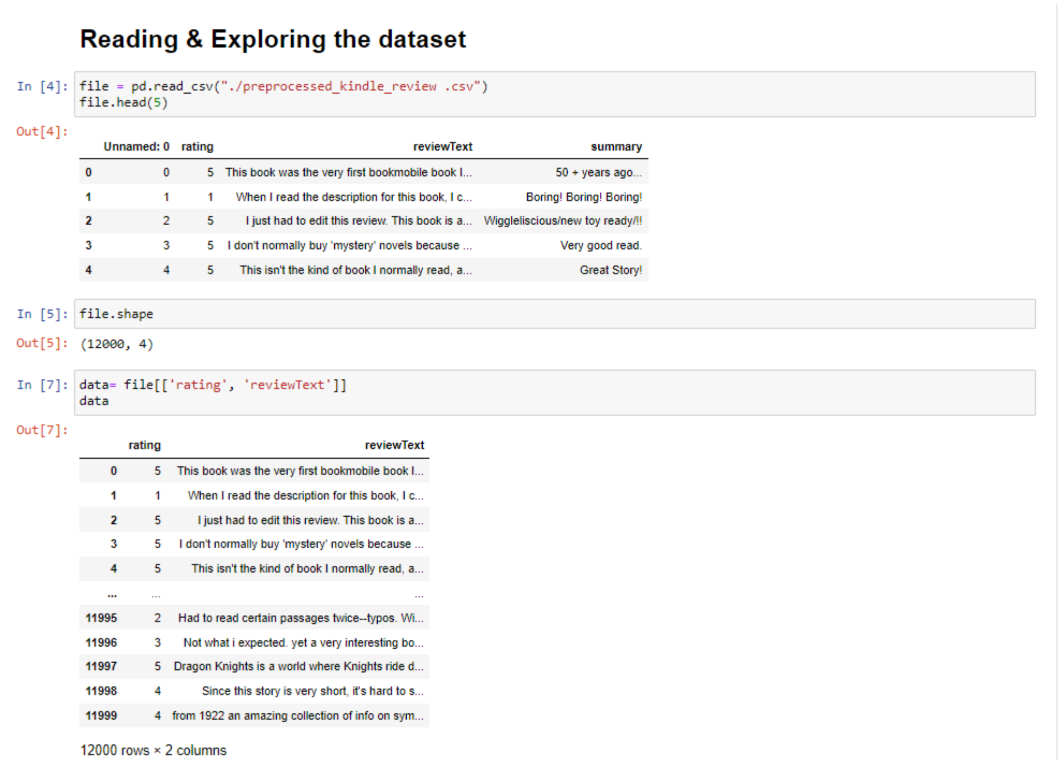
Amazon Rating Reviews for Sentiment Analysis

Problem Statement

The goal is to develop a model to predict user rating to enhance the recommendation of similar items to users. So, we need to classify reviews as positive or negative for Kaggle Data Set about Amazon Products Reviews. Which product categories has lower reviews / maybe inferior products? Which product have higher reviews / maybe superior products?

Data Collections:

The dataset consists of reviews and product information from amazon and users' ratings.



Reading & Exploring the dataset

```
In [4]: file = pd.read_csv("./preprocessed_kindle_review .csv")
        file.head(5)
```

Out[4]:

	Unnamed: 0	rating	reviewText	summary
0	0	5	This book was the very first bookmobile book I...	50 + years ago...
1	1	1	When I read the description for this book, I c...	Boring! Boring! Boring!
2	2	5	I just had to edit this review. This book is a...	Wiggleliscious/new toy ready/!!
3	3	5	I don't normally buy 'mystery' novels because ...	Very good read.
4	4	5	This isn't the kind of book I normally read, a...	Great Story!

```
In [5]: file.shape

Out[5]: (12000, 4)

In [7]: data= file[['rating', 'reviewText']]
        data
```

Out[7]:

	rating	reviewText
0	5	This book was the very first bookmobile book I...
1	1	When I read the description for this book, I c...
2	5	I just had to edit this review. This book is a...
3	5	I don't normally buy 'mystery' novels because ...
4	5	This isn't the kind of book I normally read, a...
...	...	...
11995	2	Had to read certain passages twice--typos. Wi...
11996	3	Not what i expected. yet a very interesting bo...
11997	5	Dragon Knights is a world where Knights ride d...
11998	4	Since this story is very short, it's hard to s...
11999	4	from 1922 an amazing collection of info on sym...

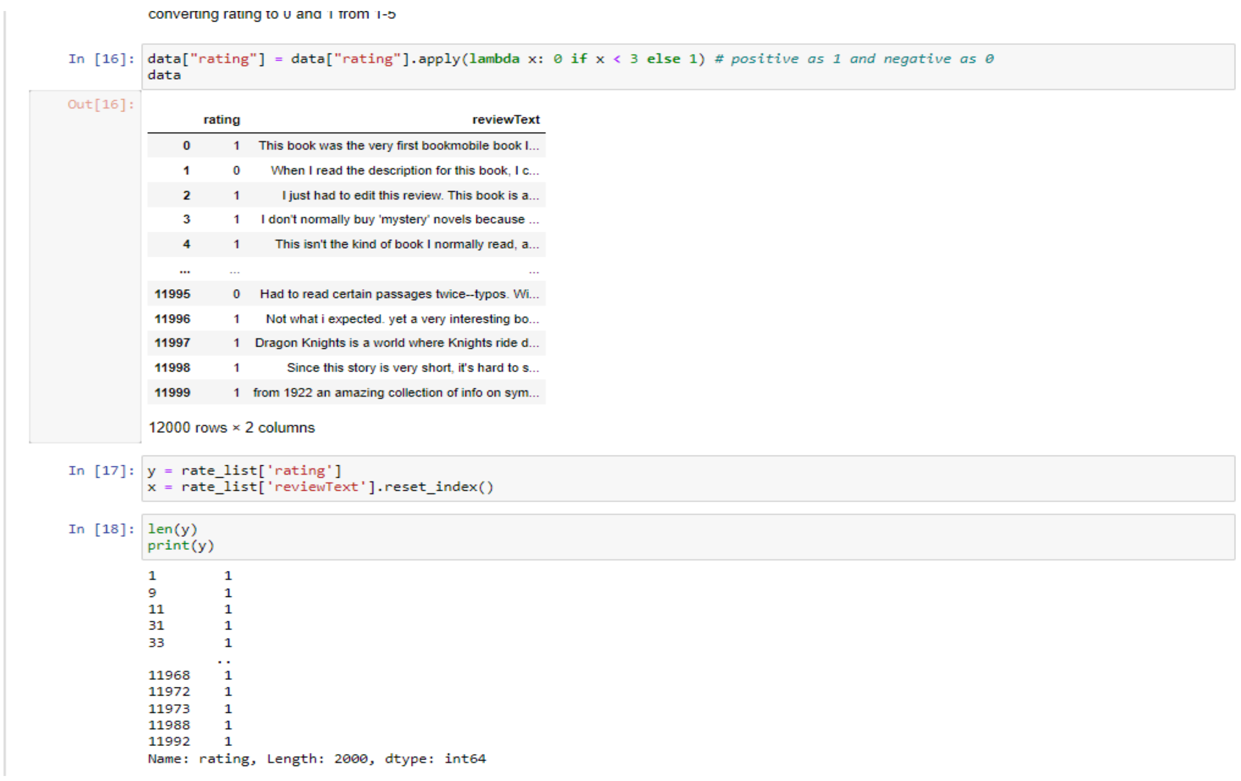
12000 rows × 2 columns

Data Wrangling

we go over some simple techniques to clean and prepare text data for modeling with machine learning.

- Simple text cleaning processes
- Lexicon-based text processing using the NLTK (Natural Language Toolkit) library
- Stop words removal
- Stemming
- Lemmatization

Since the ratings of the reviews were not distributed normally, I decided to decrease rating classes from 5 to 2 by merging Rating 1-2 as ‘0’ and Rating 3-4-5 as ‘1’.



converting rating to 0 and 1 from 1-5

```
In [16]: data["rating"] = data["rating"].apply(lambda x: 0 if x < 3 else 1) # positive as 1 and negative as 0
         data
```

Out[16]:

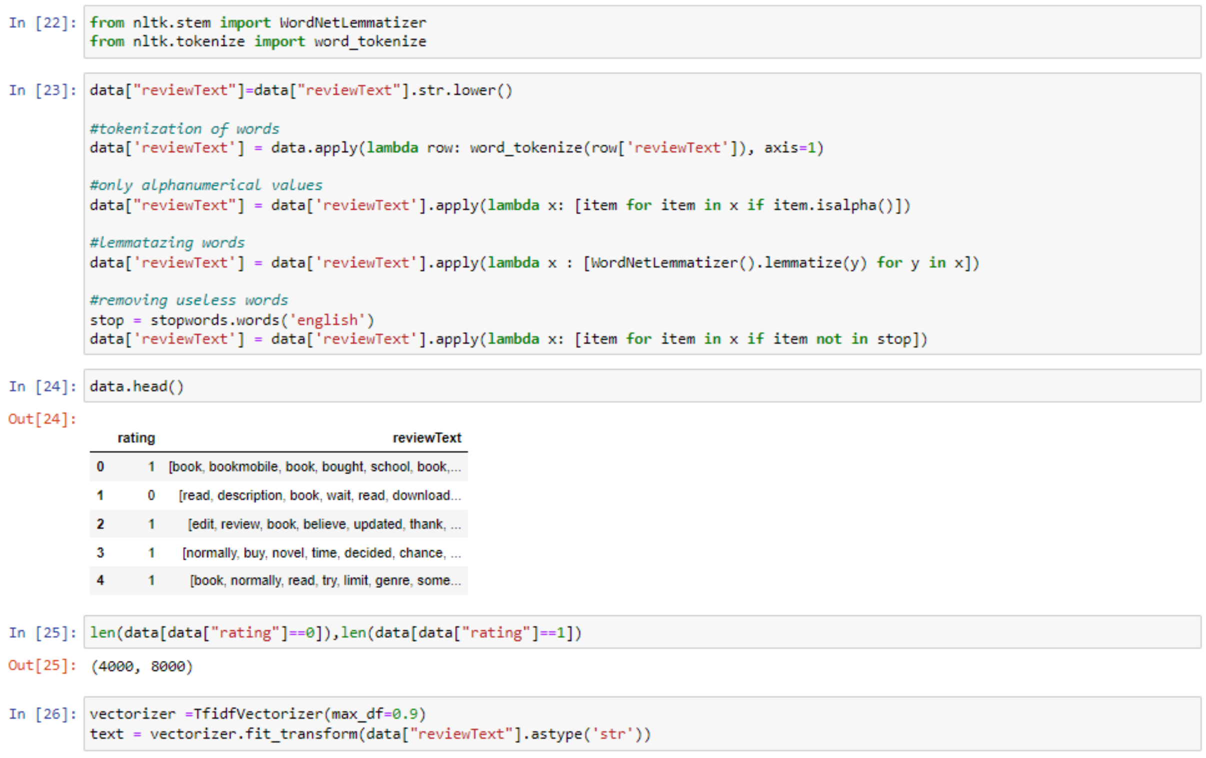
	rating	reviewText
0	1	This book was the very first bookmobile book I...
1	0	When I read the description for this book, I c...
2	1	I just had to edit this review. This book is a...
3	1	I don't normally buy 'mystery' novels because ...
4	1	This isn't the kind of book I normally read, a...
...	...	...
11995	0	Had to read certain passages twice--typos. Wi...
11996	1	Not what i expected. yet a very interesting bo...
11997	1	Dragon Knights is a world where Knights ride d...
11998	1	Since this story is very short, it's hard to s...
11999	1	from 1922 an amazing collection of info on sym...

12000 rows × 2 columns

```
In [17]: y = rate_list['rating']
         x = rate_list['reviewText'].reset_index()

In [18]: len(y)
         print(y)

         1        1
         9        1
         11       1
         31       1
         33       1
                 ..
         11968    1
         11972    1
         11973    1
         11988    1
         11992    1
         Name: rating, Length: 2000, dtype: int64
```



```
In [22]: from nltk.stem import WordNetLemmatizer
         from nltk.tokenize import word_tokenize

In [23]: data["reviewText"]=data["reviewText"].str.lower()

         #tokenization of words
         data['reviewText'] = data.apply(lambda row: word_tokenize(row['reviewText']), axis=1)

         #only alphanumerical values
         data["reviewText"] = data['reviewText'].apply(lambda x: [item for item in x if item.isalpha()])

         #lemmatazing words
         data['reviewText'] = data['reviewText'].apply(lambda x : [WordNetLemmatizer().lemmatize(y) for y in x])

         #removing useless words
         stop = stopwords.words('english')
         data['reviewText'] = data['reviewText'].apply(lambda x: [item for item in x if item not in stop])

In [24]: data.head()
```

Out[24]:

	rating	reviewText
0	1	[book, bookmobile, book, bought, school, book,...
1	0	[read, description, book, wait, read, download...
2	1	[edit, review, book, believe, updated, thank, ...
3	1	[normally, buy, novel, time, decided, chance, ...
4	1	[book, normally, read, try, limit, genre, some...

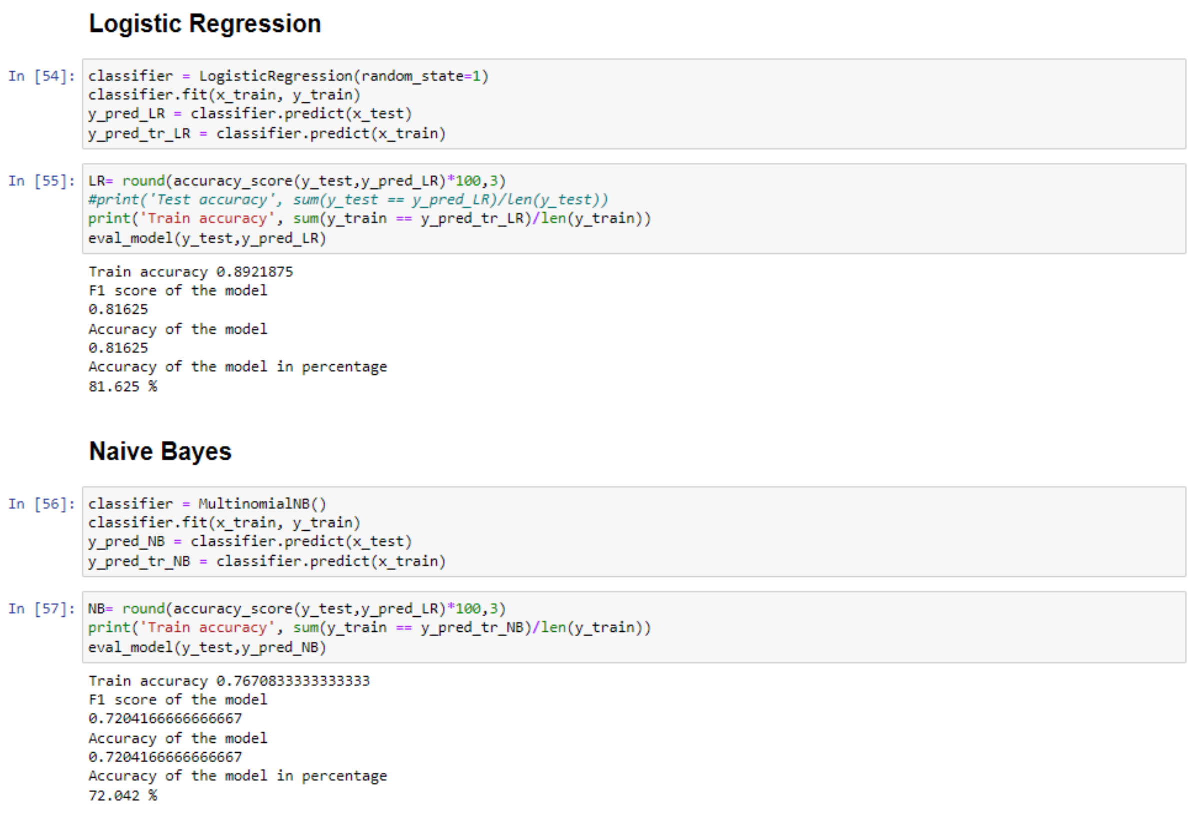
```
In [25]: len(data[data["rating"]==0]),len(data[data["rating"]==1])

Out[25]: (4000, 8000)

In [26]: vectorizer =TfidfVectorizer(max_df=0.9)
         text = vectorizer.fit_transform(data["reviewText"].astype('str'))
```

Data Modelling

This is a supervised binary classification problem. We are trying to predict the sentiment based on the reviews left by customers in Amazon e-commerce online platform. We used Python's Scikit Learn libraries to solve the problem. In this context, we implemented Logistic Regression, Random Forest, Naive Bayes, and XGBOOST algorithms.



Logistic Regression

```
In [54]: classifier = LogisticRegression(random_state=1)
         classifier.fit(x_train, y_train)
         y_pred_LR = classifier.predict(x_test)
         y_pred_tr_LR = classifier.predict(x_train)
```

```
In [55]: LR= round(accuracy_score(y_test,y_pred_LR)*100,3)
         #print('Test accuracy', sum(y_test == y_pred_LR)/len(y_test))
         print('Train accuracy', sum(y_train == y_pred_tr_LR)/len(y_train))
         eval_model(y_test,y_pred_LR)
```

```
Train accuracy 0.8921875
F1 score of the model
0.81625
Accuracy of the model
0.81625
Accuracy of the model in percentage
81.625 %
```

Naive Bayes

```
In [56]: classifier = MultinomialNB()
         classifier.fit(x_train, y_train)
         y_pred_NB = classifier.predict(x_test)
         y_pred_tr_NB = classifier.predict(x_train)
```

```
In [57]: NB= round(accuracy_score(y_test,y_pred_LR)*100,3)
         print('Train accuracy', sum(y_train == y_pred_tr_NB)/len(y_train))
         eval_model(y_test,y_pred_NB)
```

```
Train accuracy 0.7670833333333333
F1 score of the model
0.7204166666666667
Accuracy of the model
0.7204166666666667
Accuracy of the model in percentage
72.042 %
```

Random Forest

```
In [58]: classifier = RandomForestClassifier()
         classifier.fit(x_train, y_train)
         y_pred_RF = classifier.predict(x_test)
         y_pred_tr_RF = classifier.predict(x_train)
```

```
In [59]: RF= round(accuracy_score(y_test,y_pred_RF)*100,3)
         print('Train accuracy', sum(y_train == y_pred_tr_RF)/len(y_train))
         eval_model(y_test,y_pred_RF)
```

```
Train accuracy 1.0
F1 score of the model
0.7879166666666667
Accuracy of the model
0.7879166666666667
Accuracy of the model in percentage
78.792 %
```

XGBoost

```
In [40]: conda install -c conda-forge xgboost
```

...

```
In [60]: from xgboost import XGBClassifier
         classifier = XGBClassifier()
         classifier.fit(x_train, y_train)
         y_pred_XG= classifier.predict(x_test)
         y_pred_tr_XG = classifier.predict(x_train)
```

...

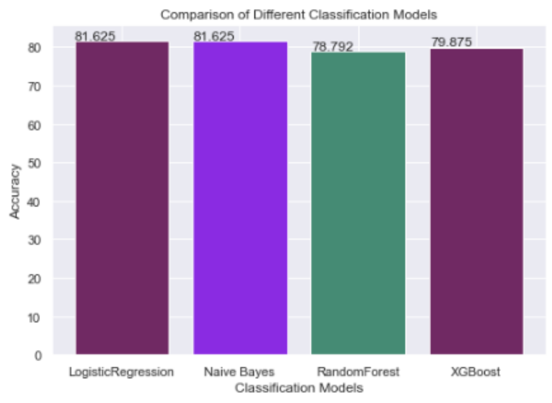
```
In [61]: XGB=round(accuracy_score(y_test,y_pred_XG)*100,3)
         print('Train accuracy', sum(y_train == y_pred_tr_XG)/len(y_train))
         eval_model(y_test,y_pred_XG)
```

```
Train accuracy 0.9210416666666666
F1 score of the model
0.79875
Accuracy of the model
0.79875
Accuracy of the model in percentage
79.875 %
```

Compare the accuracy:

Based on the accuracy percentage, Logistic Regression classifier and Naïve Bayes Classifier give the fastest and most accurate classifiers.

In [65]:
```
sns.set()
fig = plt.figure()
ax = fig.add_axes([0, 1, 1, 1])
Models = ['LogisticRegression','Naive Bayes','RandomForest' , 'XGBoost']
Accuracy=[LR, NB, RF, XGB]
ax.bar(Models, Accuracy, color=['#702963','#8a2be2', '#458B74', '#702963'])
for p in ax.patches:
    ax.annotate(str(p.get_height()), (p.get_x() * 1.005, p.get_height() * 1.005))
plt.title('Comparison of Different Classification Models')
plt.ylabel('Accuracy');
plt.xlabel('Classification Models')
plt.show()
```



What's Next?

1. Implemented deep learning technique with Keras.
2. For feature selection, I can apply threshold for word occurrence with using min_df/max_df, PCA and Singular Value Decomposition.
3. In case the dataset is not balanced, I can apply SMOTE technique.