Flipkart

Sentiment Analysis using

THAO TRAN (★★★★

THU TRAN (★★★

HUITING WU (★★★★

HOME

Flipkart

- Flipkart is primarily known as an online marketplace where customers can browse, select, and purchase a diverse array of products.
- Flipkart Private Limited is an Indian e-commerce company.
- The company initially focused on online book sales before expanding into other product categories such as consumer electronics, fashion, home essentials, groceries, and lifestyle products.
- As of March 2017, Flipkart held a 39.5% market share in the Indian e-commerce industry.

INFO

DATASET

General information:

- Contains 2 variables: Reviews and its corresponding Ratings which given by the users on Flipkart.
- Reviews are strings; Ratings are scales by 1 to 5

Goal:

- To predict whether the review given is positive or negative

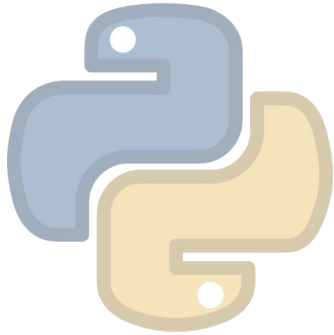
Method:

- Machine Learning (**Decision Tree)**

First 5 lines from data

	review	rating
0	It was nice produt. I like it's design a lot. ...	5
1	awesome sound....very pretty to see this nd th...	5
2	awesome sound quality. pros 7-8 hrs of battery...	4
3	I think it is such a good product not only as ...	5
4	awesome bass sound quality very good bettary l...	5

PYTHON LIBRARY



Import the data:

- ***Pandas*** : For importing the dataset.

Machine Learning Model:

- ***Scikit-learn*** : For importing the model, accuracy module, and TfidfVectorizer. (**Decision Tree Model & Confusion matrix**)
- ***Warning*** : To ignore all the warnings

Data Visualization:

- ***Matplotlib*** : To plot the visualization. **(Word Cloud)**
- ***Seaborn*** : data visualization library based on matplotlib. (**Bar plot**)

Preprocessing Text:

- ***NLTK:*** For text analysis; It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning. Using the stop words and punctuations library and tokenization in this project.
- ***Re:*** let you check if a particular string matches a given regular expression.

OVERVIEW

Data information (checking missing value)

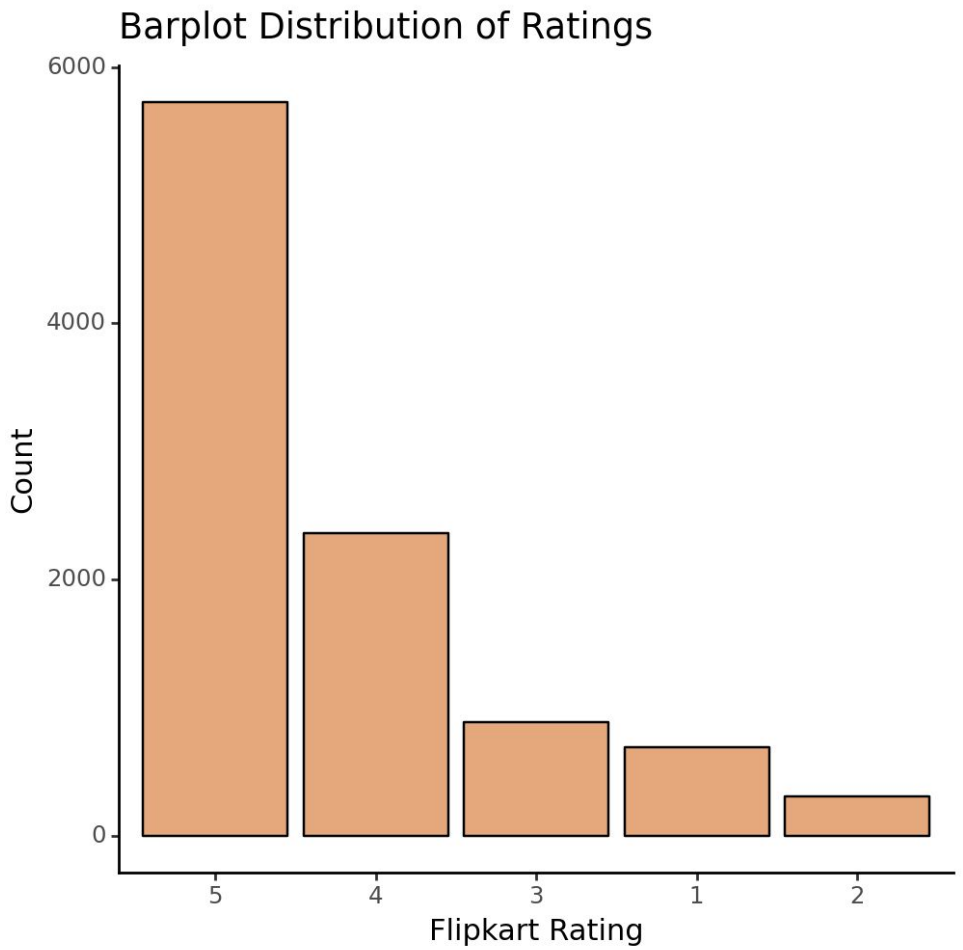
```
RangeIndex: 9976 entries, 0 to 9975
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   review  9976 non-null   object
 1   rating  9976 non-null   int64 
dtypes: int64(1), object(1)
memory usage: 156.0+ KB
None
```

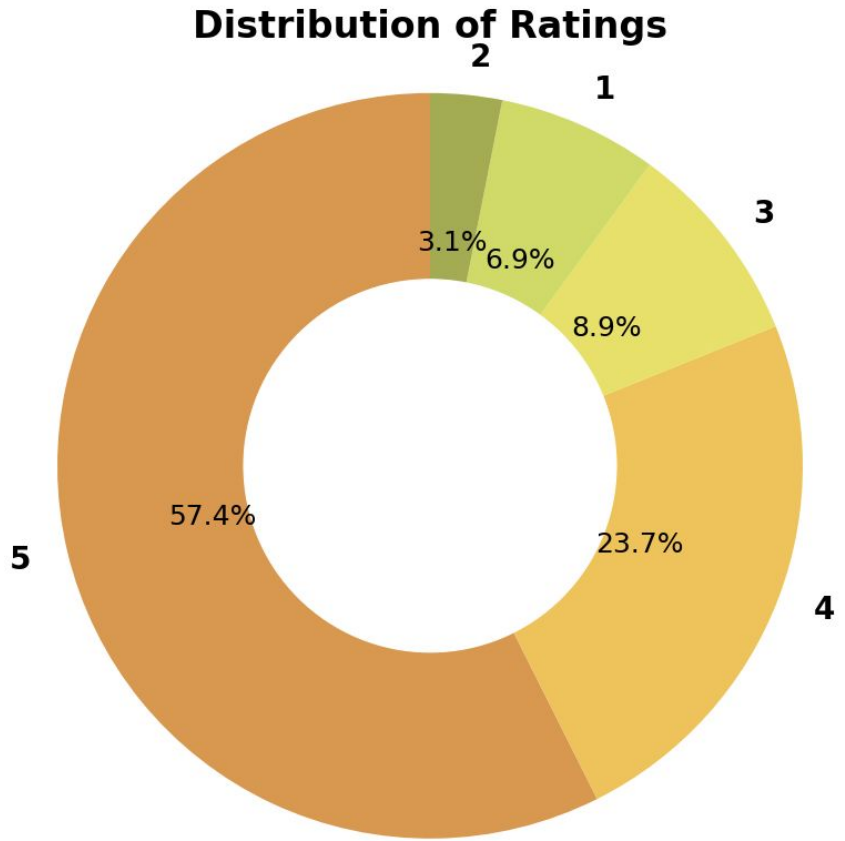
Descriptive statistical measures

Flipkart Statistical Summary

count	mean	std	min	25%	50%	75%	max
9976.0	4.215417000801924	1.167911353164182	1.0	4.0	5.0	5.0	5.0

VISUAL


Barplot Distribution of Ratings
6000
4000
2000
0
Count
5
4
3
1
2
Flipkart Rating


Distribution of Ratings
2
1
3
5
4
3.1%
6.9%
8.9%
57.4%
23.7%

Labeling

```
# rating label(final)
pos_neg = []
# creat a empty list
for i in range(len(data['rating'])):
   # for rows of variable rating
  if data['rating'][i] >= 5:
     # if the rating is greater or equals to 5
    pos_neg.append(1)
  # label the rating as 1
  else:
    pos_neg.append(0)
   # for the rating 4 or less, label it as 0.

data['label'] = pos_neg
# creat a new variable into the data named "label" and with 0 or 1 only
```

Counting the ‘label’

```
data["label"].value_counts()
```

```
1    5726
0    4250
Name: label, dtype: int64
```

Before label

	review	rating
0	It was nice produt. I like it's design a lot. ...	5
1	awesome sound....very pretty to see this nd th...	5
2	awesome sound quality. pros 7-8 hrs of battery...	4
3	I think it is such a good product not only as ...	5
4	awesome bass sound quality very good bettary l...	5

After label

	review	rating	label
0	it was nice produt i like its design a lot its...	5	1
1	awesome soundvery pretty to see this nd the so...	5	1
2	awesome sound quality pros 78 hrs of battery l...	4	0
3	i think it is such a good product not only as ...	5	1
4	awesome bass sound quality very good bettary l...	5	1

Preprocessing text

```
def preprocess_text(text_data):
  preprocessed_text = [] # create a list

  for sentence in tqdm(text_data): # for the sentence in the data
  ### Removing punctuations
    sentence = re.sub(r'[^\w\s]', '', sentence)
  # search the pattern and replace them with a space in the sentence

  ### Converting lowercase and removing stopwords
    preprocessed_text.append(' '.join(token.lower() # converts each word (token) to lowercase
  # to join the individual words of a sentence back together
  # into a single string with spaces in between
                    for token in nltk.word_tokenize(sentence)
          # splits the sentence into individual words;
                    if token.lower() not in stopwords.words('english')))
          # if the words are stopwords in English then split it into words and add into the list.
  return preprocessed_text
```

Before preprocessing

	review	rating	label
0	it was nice produt i like its design a lot its...	5	1
1	awesome soundvery pretty to see this nd the so...	5	1
2	awesome sound quality pros 78 hrs of battery l...	4	0
3	i think it is such a good product not only as ...	5	1
4	awesome bass sound quality very good bettary l...	5	1

After preprocessing

	review	rating	label
0	nice produt like design lot easy carry looked ...	5	1
1	awesome soundvery pretty see nd sound quality ...	5	1
2	awesome sound quality pros 78 hrs battery life...	4	0
3	think good product per quality also design qui...	5	1
4	awesome bass sound quality good bettary long l...	5	1

10 common words with label “1” (Positive review)

```
# 10 COMMON WORDS WITH LABEL "1" (POSSITIVE)
from collections import Counter
# Filter the DataFrame to select rows with Label=1
label_1_df = data[data['label'] == 1]
# Combine the text from these rows into a single string
text_for_label_1 = ' '.join(label_1_df['review'])
# Tokenize the text into words
words = text_for_label_1.split()
# Count the occurrences of each word
word_counts = Counter(words)
# Find the most repeated words
most_common_words = word_counts.most_common(10)
# Display
print("Most common words in 'Text' with Label=1:")
for word, count in most_common_words:
    print(f"{word}: {count}")
```

WORDS	COUNT
good	2169
product	1548
sound	1379
quality	1301
bass	979
nice	935
best	835
awesome	722
productread	688
read	629

Positive words

WORDS	COUNT
good	2169
product	1548
sound	1379
quality	1301
bass	979
nice	935
best	835
awesome	722
productread	688
read	629

Negative words

WORDS	COUNT
good	2080
sound	1295
quality	1188
product	1003
bass	681
nice	470
bluetooth	417
battery	409
use	363
productread	360

NOTE:
Some words has been repeated in both “positive” and “negative” words, such as : “good”, “product”, “quality”.... since we consider 4 stars rating as a negative review. Besides that, people can use “not”, “ no” …in front of positive word to describe their thought.

Words cloud visual (Positive reviews)

GENERAL ANALYSIS

CONVERT TEXT TO NUMBER

- **Computers Prefer Numbers:**
  - Machines understand numbers, not words. Converting text into numbers makes it easier for computers to process information.
- **Mathematics for Learning:**
  - Numeric representations allow machines to use math to recognize patterns efficiently, aiding in the learning process for tasks like prediction and classification.
- **Efficiency and Prediction:**
  - Converting text to numbers speeds up computations, making machine learning models more efficient. This numeric understanding helps machines predict outcomes and make decisions based on data.

TF-IDF SCORE (Term Frequency-Inverse Document Frequency)

- TF-IDF measures word importance. "TF" counts word frequency in a document, and "IDF" considers how unique a word is across all documents.
- A higher TF-IDF score suggests that a term is more important or relevant within a specific document.
- It helps identify terms that are distinctive and carry meaningful information within the context of that document.

TF-IDF SCORE (Term Frequency-Inverse Document Frequency)

```
cv = TfidfVectorizer(max_features=2500)
# transform a collection of text documents into a TF-IDF matrix
# max_features: specifies the maximum number of features (words) to include in the TF-IDF matrix


X = cv.fit_transform(data['review'] ).toarray()
# fits the vectorizer to the 'review' data and transforms it into a TF-IDF matrix
# converts the TF-IDF matrix into a dense NumPy array
```

+ Code + Text

```
X # The feature matrix = TF-IDF matrix
```

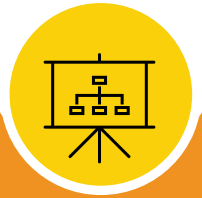
Feature matrix

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```









MACHINE LEARNING

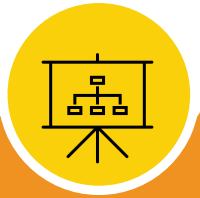
DATA SPLITING

- **Training Set:**
  - Model learns patterns and relationships.
  - Like a practice ground for the model.
- **Test Set:**
  - Evaluates how well the model applies learning.
  - Simulates real-world scenarios with new, unseen data.
- **Why Split?**
  - Ensures model can handle new situations.
  - Prevents overfitting by avoiding memorization of specific data.
- **Evaluation:**
  - Metrics like accuracy assess model performance.
  - Allows adjustments for better predictions.



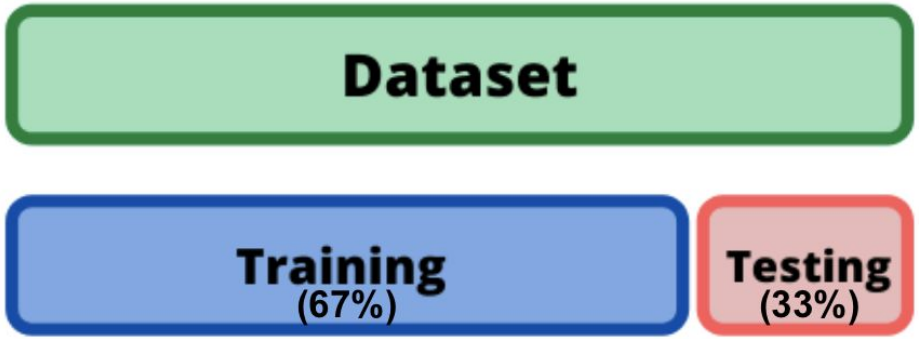






MACHINE LEARNING

DATA SPLITING



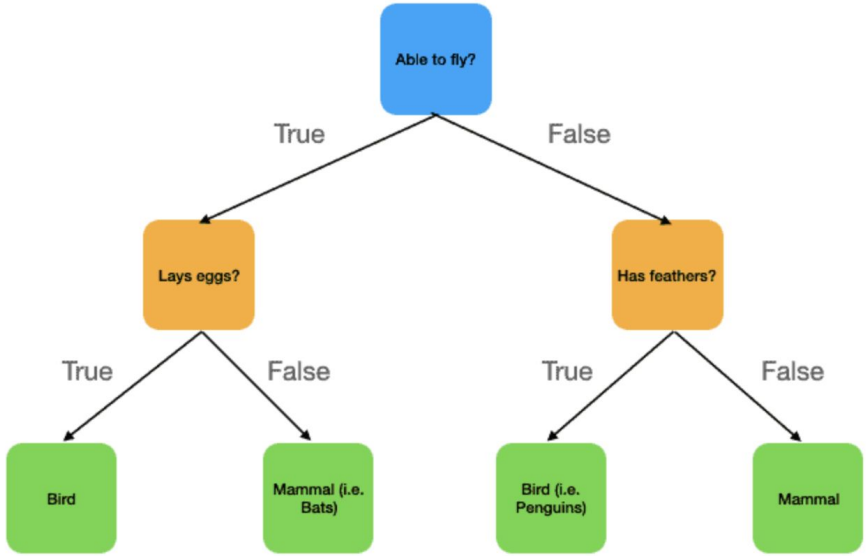

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, data['label'],
                                   test_size=0.33,
                                   stratify=data['label'],
                                   random_state = 42)
```

DECISION TREE

- A powerful machine learning algorithm for classification and regression tasks.
- A family of non-parametric supervised learning models
- Can be applied to both classification and regression problems.

EXAMPLE OF DECISION TREE

- **Root node**: (BLUE)
  - Starting point of the decision tree.
  - Represents the entire dataset.
- **Interior nodes**: (ORANGE)
  - Decision-making point within the tree.
  - Poses a question or condition based on a feature.
- **Leaf nodes**: (GREEN)
  - End point of a decision tree branch.
  - Provides the final prediction or outcome.



DECISION TREE

Fitting data with Decision Tree model

```
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score
model = DecisionTreeClassifier(random_state=0)
model.fit(X_train,y_train)
```

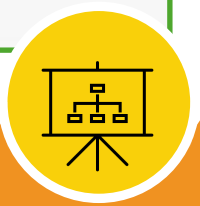
Visual Decision Tree model

```
# Visualization of the Decision Tree
plt.figure(figsize=(12, 8))
plot_tree(model, filled=True, class_names=True,max_depth=2)
plt.show()
```



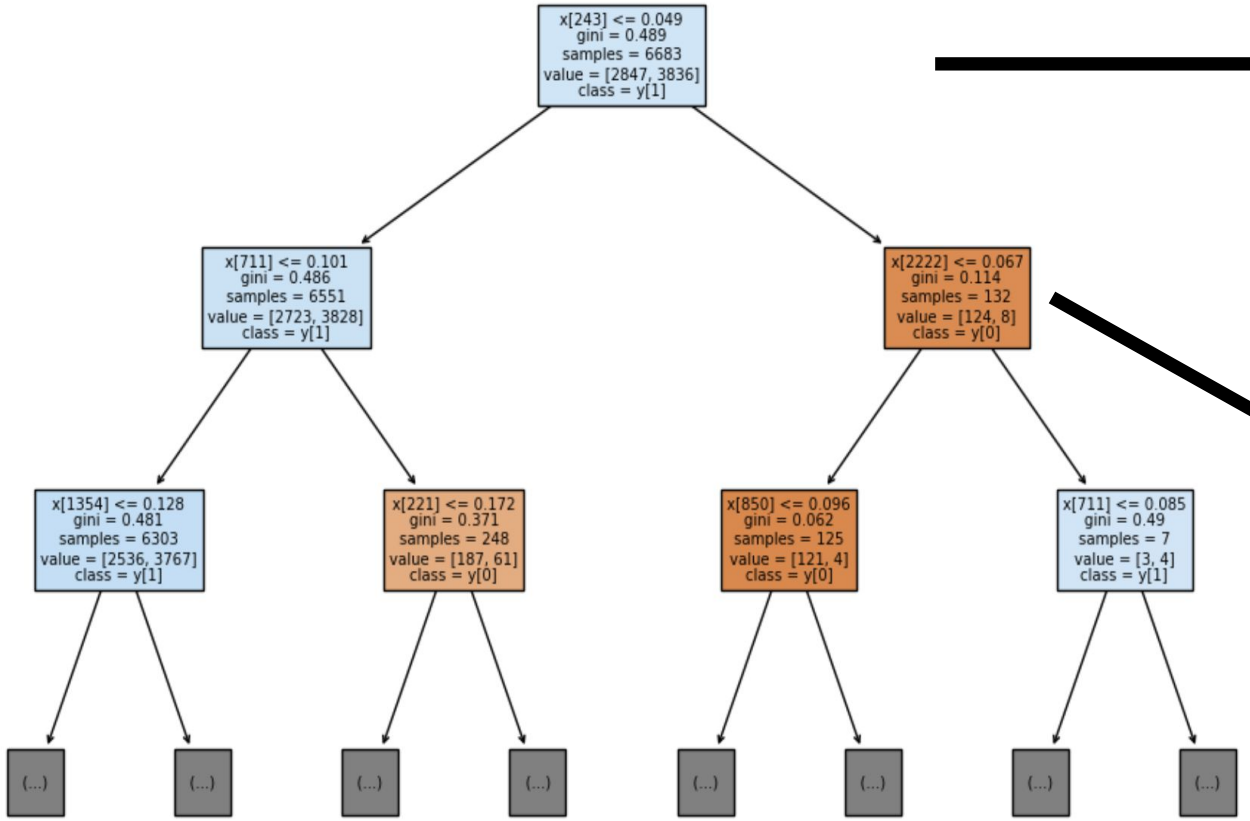


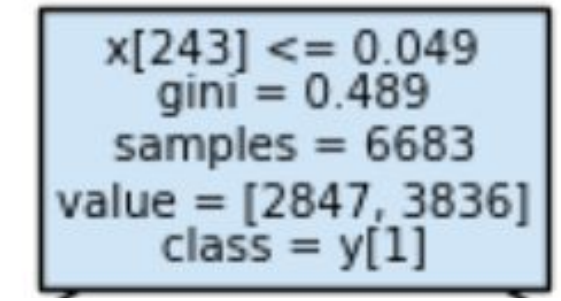


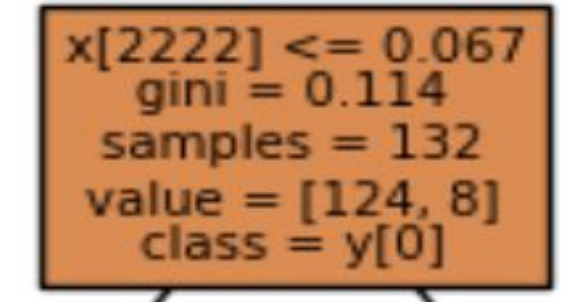


DECISION TREE

Flipkart Decision Tree







DECISION TREE

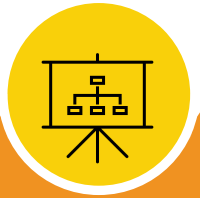
```
from sklearn.metrics import accuracy_score , roc_curve, auc
# Define a function to display accuracy and AUC
def compute_accuracy_AUC(Model, y_test, y_hat, pred_prob):
    accuracy = accuracy_score(y_test, y_hat)
    fpr, tpr, thresholds = roc_curve(y_test, pred_prob[:,1])
    AUC = auc(fpr, tpr)
    print(f'{Model} model has accuracy {accuracy}, AUC {AUC}')
    return accuracy, AUC

#testing the model on the test set
y_pred= model.predict(X_test)
y_prob= model.predict_proba(X_test)
model_accuracy, model_AUC = compute_accuracy_AUC('Decision Tree', y_test, y_pred,  y_prob)
```

```
Decision Tree model has accuracy 0.6771940479805648, AUC 0.6733786632574943
```









MACHINE LEARNING

CONCLUSION

- The sentiment analysis using “decision tree” gives us the accuracy rate of 67.72% and AUC of 67.34%
- It’s not a good rate.
- Some further improvement can be made to improve the accuracy rate :
  - ✔ Drop the duplicate reviews
  - ✔ Relabel the data with 4 and 5 stars as “positive” reviews, and others as negative .
  - ✔ Using different ratio to split the data in training set and test set .
  - ✔ Using other machine learning method: Logistic Regression, Random Forest, SVM, Cross-Validation...