Sentiment Analysis

Amazon Product Reviews

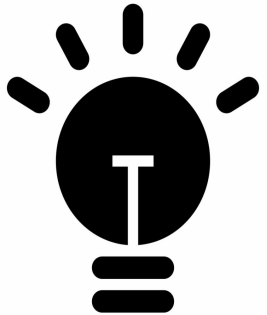
Discussion Points

1. Introduction/Problem statement
2. Schema
3. Exploratory Data Analysis
4. Data Pre-Processing
5. Classification algorithms
6. Results
7. Conclusion

Introduction

Sentiment analysis is the analysis of data to extract subjective information, to better understand the perspective and feelings of the customers on a product or a business using **Natural language processing,** which helps the businesses understand about their reputation in the market.

Shopping online has become a normal way people transact these days, a business without an online presence does not seem to sustain and consequently understanding the opinions of the customers becomes important.

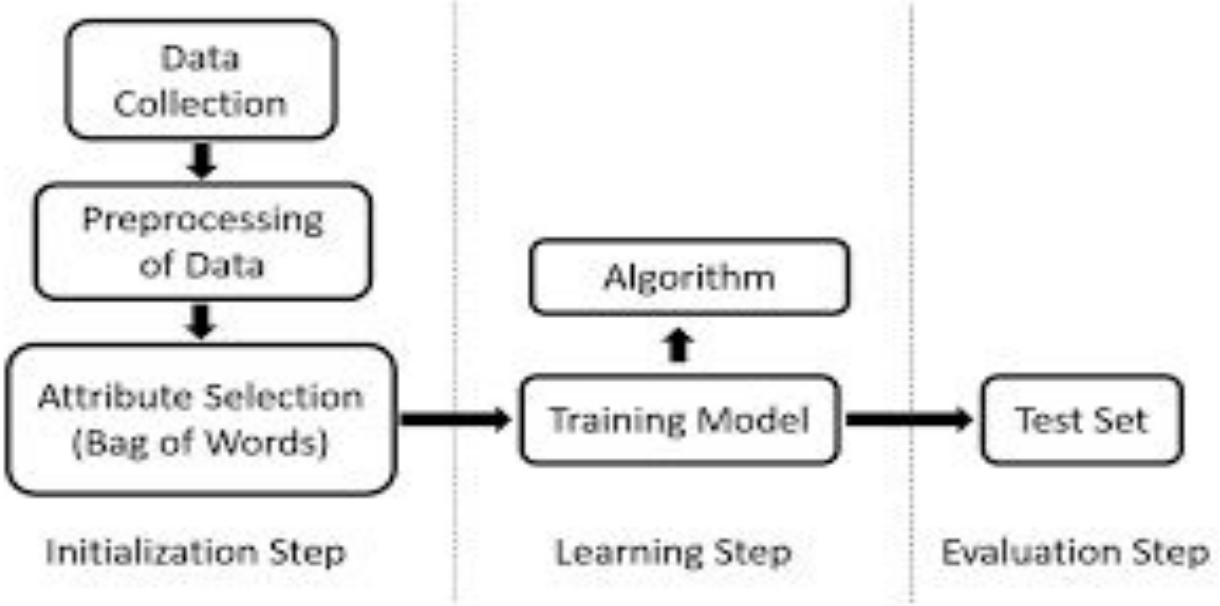


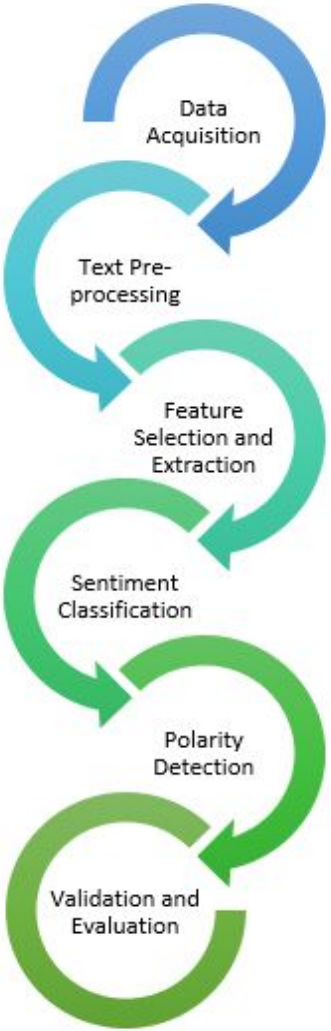
And when it comes to online business **Amazon** is the beast and is ruling the industry.

Problem Statement

The aim of our project is to investigate the efficacy of different Supervised Machine Learning algorithms on performing Sentiment Analysis i.e., tagging the randomly selected Amazon Product Reviews as Positive and Negative.

Schema


Data Collection
Preprocessing of Data
Attribute Selection (Bag of Words)
Algorithm
Training Model
Test Set
Initialization Step
Learning Step
Evaluation Step


Data Acquisition
Text Pre-processing
Feature Selection and Extraction
Sentiment Classification
Polarity Detection
Validation and Evaluation

About the Data

- The dataset '**Consumer reviews on Amazon products**' has been taken from Data.world
- The dataset consists of 28,332 rows and 24 columns.
- The dataset has reviews from different categories like Electronics, Health & Beauty, Toys & Games and Office supplies.
- The columns required for our project are the **review text** and the **review ratings**

Dataset

In [42]:
```
import pandas as pd
data = pd.read_csv('Datafiniti_Amazon_Consumer_Reviews_of_Amazon_Products_May19.csv')
```

In [44]:
```
data = data.rename(columns={"reviews.rating":"Rating", "reviews.text":"Reviews"})
data.head(3)
```

Out[44]:

	id	dateAdded	dateUpdated	name	asins	brand	categories	primaryCategor
0	AVpgNzjwLJeJML43Kpxn	2015-10-30T08:59:32Z	2019-04-25T09:08:16Z	AmazonBasics AAA Performance Alkaline Batterie...	B00QWO9P0O,B00LH3DMUO	Amazonbasics	AA,AAA,Health,Electronics,Health & Household,C...	Health & Bea
1	AVpgNzjwLJeJML43Kpxn	2015-10-30T08:59:32Z	2019-04-25T09:08:16Z	AmazonBasics AAA Performance Alkaline Batterie...	B00QWO9P0O,B00LH3DMUO	Amazonbasics	AA,AAA,Health,Electronics,Health & Household,C...	Health & Bea
2	AVpgNzjwLJeJML43Kpxn	2015-10-30T08:59:32Z	2019-04-25T09:08:16Z	AmazonBasics AAA Performance Alkaline Batterie...	B00QWO9P0O,B00LH3DMUO	Amazonbasics	AA,AAA,Health,Electronics,Health & Household,C...	Health & Bea

3 rows × 24 columns

Cleaned Dataset

```
In [3]: df = data.drop(['id','dateAdded','dateUpdated','name','asins','brand','categories','primaryCategories','imageURLs','keys','revie
          'reviews.doRecommend','reviews.id','reviews.numHelpful','reviews.sourceURLs','reviews.title','reviews.username','source
           'manufacturerNumber','reviews.date','reviews.dateSeen'], 1)
```

```
In [5]: df.isnull().values.any()
```

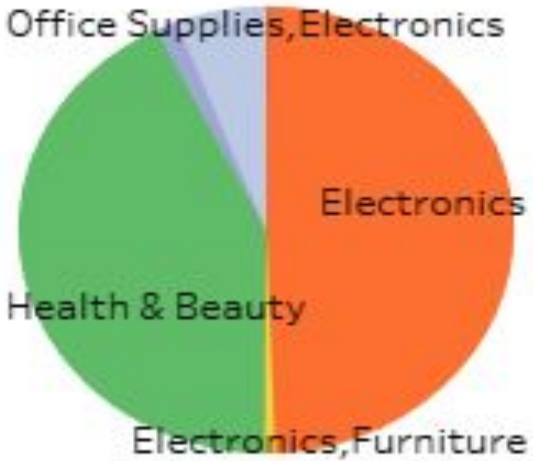
```
Out[5]: False
```

```
In [4]: df.head(5)
```

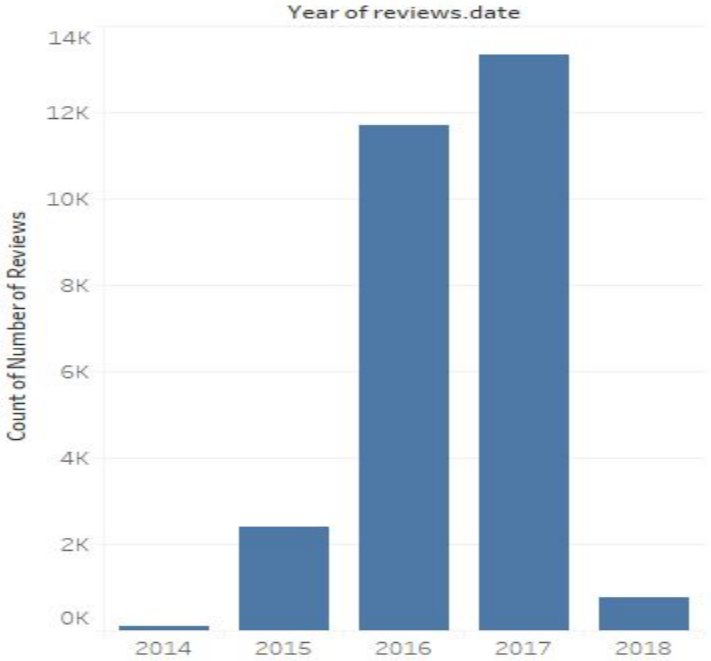
Out[4]:

	Rating	Reviews
0	3	I order 3 of them and one of the item is bad q...
1	4	Bulk is always the less expensive way to go fo...
2	5	Well they are not Duracell but for the price i...
3	5	Seem to work as well as name brand batteries a...
4	5	These batteries are very long lasting the pric...

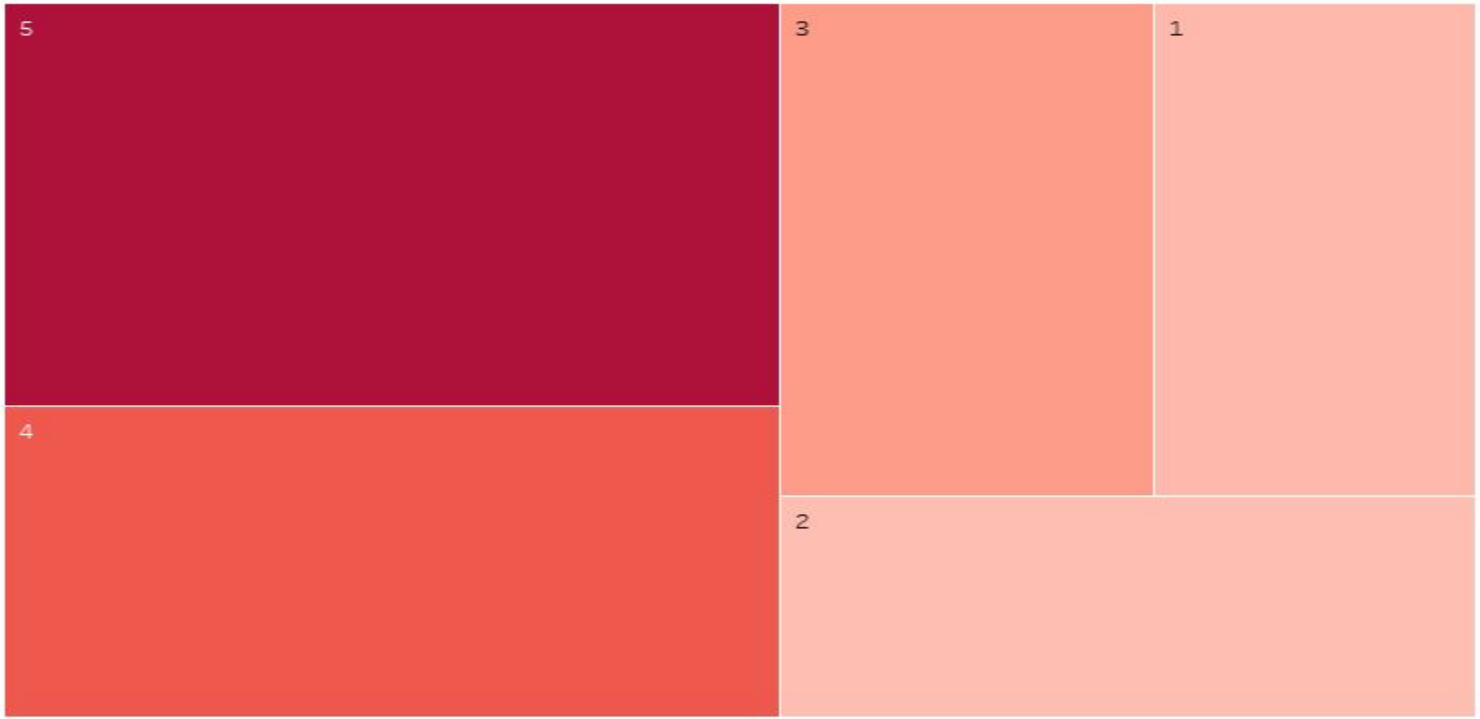
By category



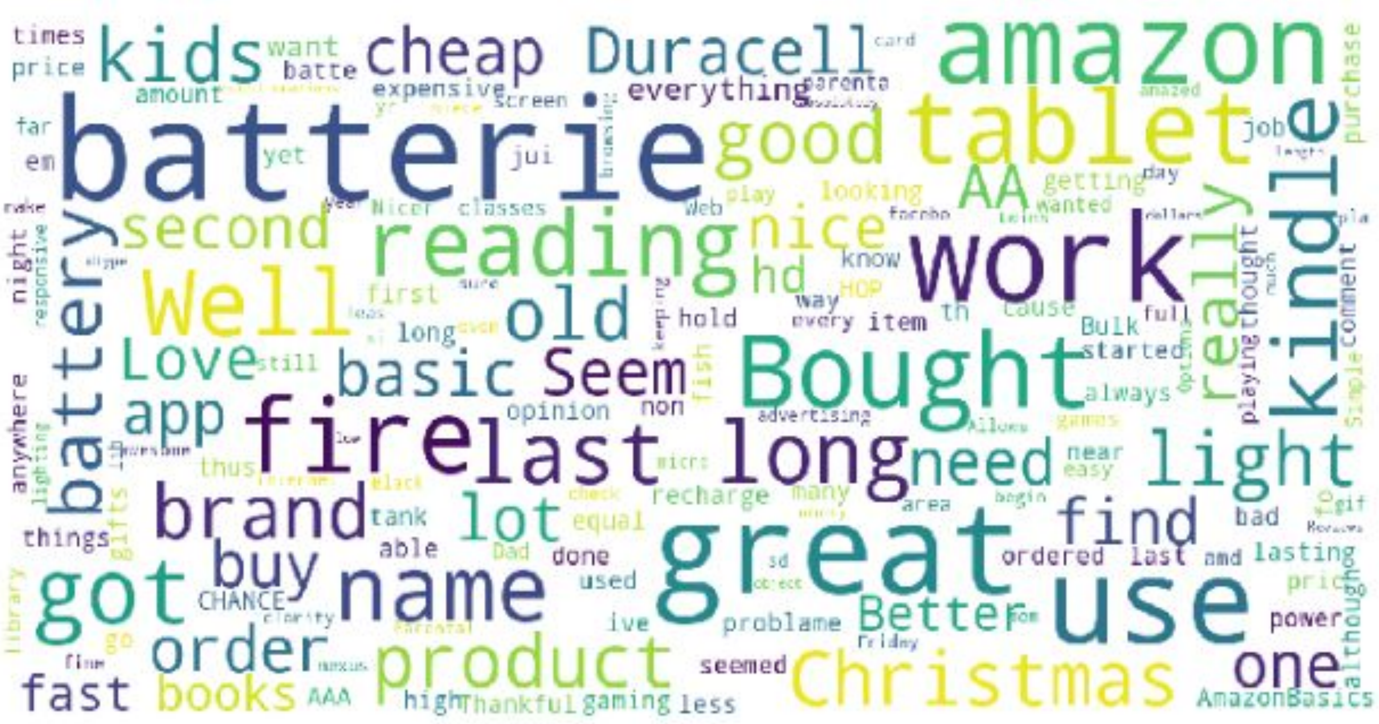
By Year



Data distribution by ratings



Word Cloud



times
price
kids
want
batte
cheap
Duracell
amazon
amount
expensive
screen
everything
far
batterie
good
tablet
job
purchase
em
yet
jui
AA
getting
day
second
reading
nice
work
kindle
really
Nicer
classes
Web
looking
wanted
night
responsive
Well
first
hd
know
old
HOP
way
hold
item
cause
Bulk
full
thought
comment
battery
Love
still
basic
Seem
Bought
started
playing
long
always
app
fire
opinion
non
advertising
games
anywhere
last
long
need
light
near
easy
thus
brand
lot
recharge
many
area
find
bad
things
tank
equal
able
done
ordered
last
amd
lasting
got
buy
name
great
use
used
CHANCE
Better
power
order
ive
problame
Friday
although
product
seemed
Christmas
one
fast
books
AAA
high
Thankful
gaming
less
AmazonBasics

Data Preprocessing

1. Remove **special characters** (‘ ‘, @, _, !, &, ...)
2. Convert all the characters into **lowercase** (A-Z to a-z)
3. **Tokenization**: split all the sentences into words
4. Remove **stopwords** (as, the, in , an , there, it...)
5. **Stemming** : converting a word to its root or base form

   (likes, liked, likely, liking→ ‘Like’)

6. Joining all the words and creating a **corpus**
7. **Feature selection(Bag of words):** selecting the most frequent words

Data Preprocessing

1. Libraries

In [47]:
```
import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

2. The preprocessing

In [40]:
```
ps = PorterStemmer()
corpus = []
for i in range(0, len(df)):
    review = re.sub('[^a-zA-Z]', ' ', df['Reviews'][i])
    review = review.lower()
    review = review.split()

    review = [ps.stem(word) for word in review if not word in stopwords.words('english')]
    review = ' '.join(review)
    corpus.append(review)
```

In [41]:
```
corpus
```

Out[41]:
```
['order one item bad qualiti miss backup spring put pc aluminum make batteri work',
 'bulk alway less expens way go product like',
 'well duracel price happi',
 'seem work well name brand batteri much better price',
 'batteri long last price great',
 'bought lot batteri christma amazonbas cell good notic differ brand name batteri amazon h
hous hand buy',
 'ive problam batteri order past pleas',
 'well look cheap non recharg batteri last quit perfect noth say',
 'hold amount high power juic like energ duracel half price',
 'amazonbas aa aaa batteri done well appear good shelf life buy',
 'find amazon basic batteri equal superior name brand one believ start buy sooner packag
 'first start get amazon basic batteri realli like recent purchas seem last like mayb mix
e test feel brand may last longer howev price hard beat',
```

Feature selection(Bag of words)

```
In [48]: from sklearn.feature_extraction.text import CountVectorizer
         cv = CountVectorizer(max_features=6000)
         X = cv.fit_transform(corpus).toarray()

In [49]: X

Out[49]: 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]], dtype=int64)

In [50]: X.shape

Out[50]: (28332, 6000)
```

Tagging the Reviews

```
In [53]: df['pos_neg'] = [1 if x > 3 else 0 for x in df['Rating']]

In [54]: y = df['pos_neg']

In [55]: y.head(5)

Out[55]: 0    0
         1    1
         2    1
         3    1
         4    1
         Name: pos_neg, dtype: int64
```

Positive

Negative

Splitting the data into Training and Test sets

```
In [56]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)

In [57]: X_train.shape

Out[57]: (22665, 6000)

In [58]: y_train.shape

Out[58]: (22665,)

In [59]: X_test.shape

Out[59]: (5667, 6000)

In [60]: y_test.shape

Out[60]: (5667,)
```

Classification algorithms

1. Naive Bayes Classifier

```
In [12]:
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB().fit(X_train, y_train)

y_pred=model.predict(X_test)
```

```
In [13]:
y_pred
```

```
Out[13]: array([1, 1, 0, ..., 1, 1, 1], dtype=int64)
```

In [15]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[15]:
```
0.9156520204693842
```

In [16]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
```

Out[16]:
```
array([[ 284,  283],
       [ 195, 4905]], dtype=int64)
```

In [17]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.59      0.50      0.54       567
           1       0.95      0.96      0.95      5100

    accuracy                           0.92      5667
   macro avg       0.77      0.73      0.75      5667
weighted avg       0.91      0.92      0.91      5667
```

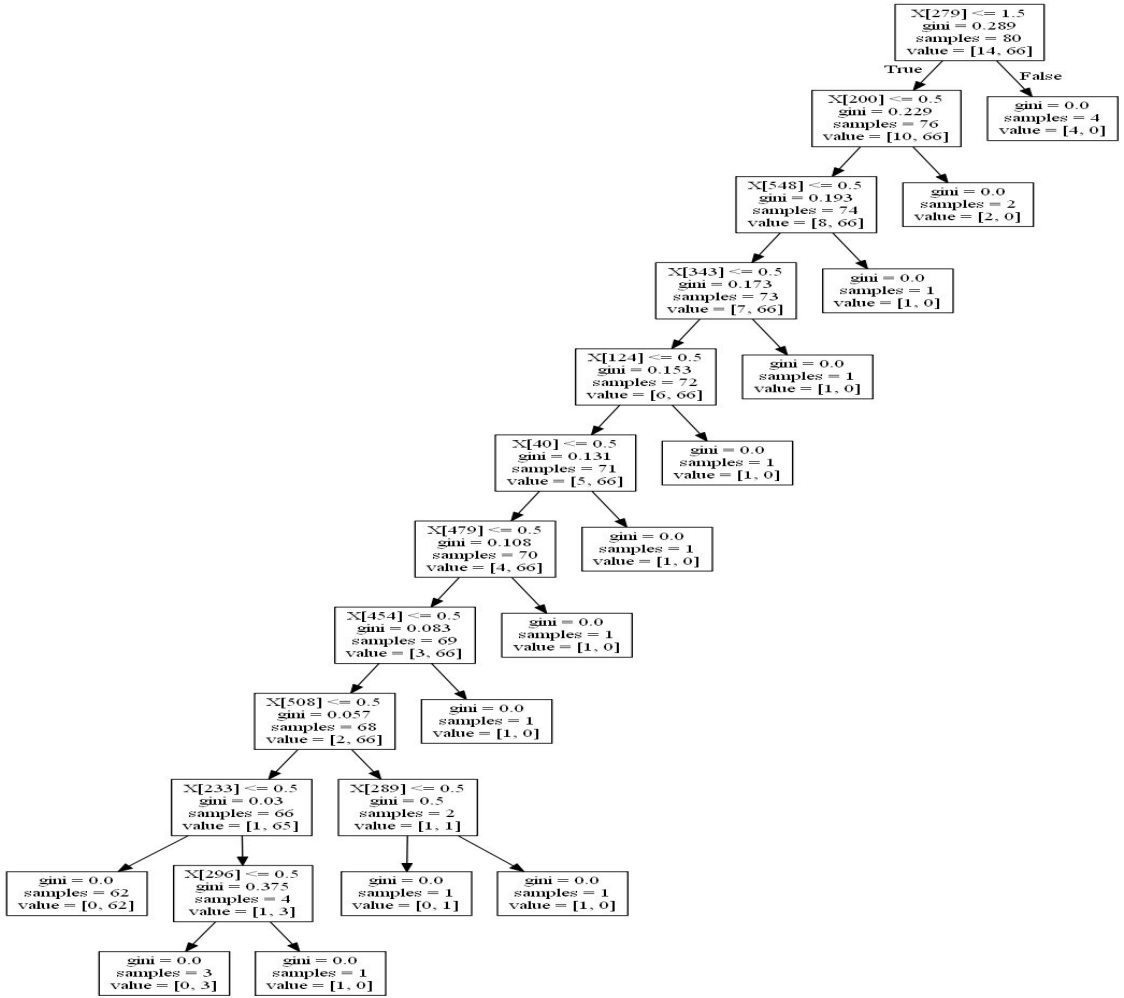
2. Decision Tree Classifier

In [18]:
```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
clf = clf.fit(X_train,y_train)

y_pred2 = clf.predict(X_test)
```

In [19]:
```
y_pred2.shape
```

Out[19]: (5667,)


X[279] <= 1.5
gini = 0.289
samples = 80
value = [14, 66]
True
False
X[200] <= 0.5
gini = 0.229
samples = 76
value = [10, 66]
gini = 0.0
samples = 4
value = [4, 0]
X[548] <= 0.5
gini = 0.193
samples = 74
value = [8, 66]
gini = 0.0
samples = 2
value = [2, 0]
X[343] <= 0.5
gini = 0.173
samples = 73
value = [7, 66]
gini = 0.0
samples = 1
value = [1, 0]
X[124] <= 0.5
gini = 0.153
samples = 72
value = [6, 66]
gini = 0.0
samples = 1
value = [1, 0]
X[40] <= 0.5
gini = 0.131
samples = 71
value = [5, 66]
gini = 0.0
samples = 1
value = [1, 0]
X[479] <= 0.5
gini = 0.108
samples = 70
value = [4, 66]
gini = 0.0
samples = 1
value = [1, 0]
X[454] <= 0.5
gini = 0.083
samples = 69
value = [3, 66]
gini = 0.0
samples = 1
value = [1, 0]
X[508] <= 0.5
gini = 0.057
samples = 68
value = [2, 66]
gini = 0.0
samples = 1
value = [1, 0]
X[233] <= 0.5
gini = 0.03
samples = 66
value = [1, 65]
X[289] <= 0.5
gini = 0.5
samples = 2
value = [1, 1]
gini = 0.0
samples = 62
value = [0, 62]
X[296] <= 0.5
gini = 0.375
samples = 4
value = [1, 3]
gini = 0.0
samples = 1
value = [0, 1]
gini = 0.0
samples = 1
value = [1, 0]
gini = 0.0
samples = 3
value = [0, 3]
gini = 0.0
samples = 1
value = [1, 0]

```
In [20]: accuracy_score(y_test,y_pred2)
```

```
Out[20]: 0.9354155637903653
```

```
In [21]: confusion_matrix(y_test,y_pred2)
```

```
Out[21]: array([[ 397,  170],
                [ 196, 4904]], dtype=int64)
```

```
In [22]: print(classification_report(y_test,y_pred2))
```

```
              precision    recall  f1-score   support

           0       0.67      0.70      0.68       567
           1       0.97      0.96      0.96      5100

    accuracy                           0.94      5667
   macro avg       0.82      0.83      0.82      5667
weighted avg       0.94      0.94      0.94      5667
```

3. Support Vector Machines (SVM)

```
In [17]: from sklearn.svm import SVC

In [18]: svc = SVC(kernel = 'linear')
         svc.fit(X_train,y_train)

Out[18]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
           decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
           kernel='linear', max_iter=-1, probability=False, random_state=None,
           shrinking=True, tol=0.001, verbose=False)

In [19]: y_pred3 = svc.predict(X_test)
```

In [22]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred3)
```

Out[22]:
```
0.9426504323275101
```

In [24]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred3)
```

Out[24]:
```
array([[ 325,  242],
       [  83, 5017]], dtype=int64)
```

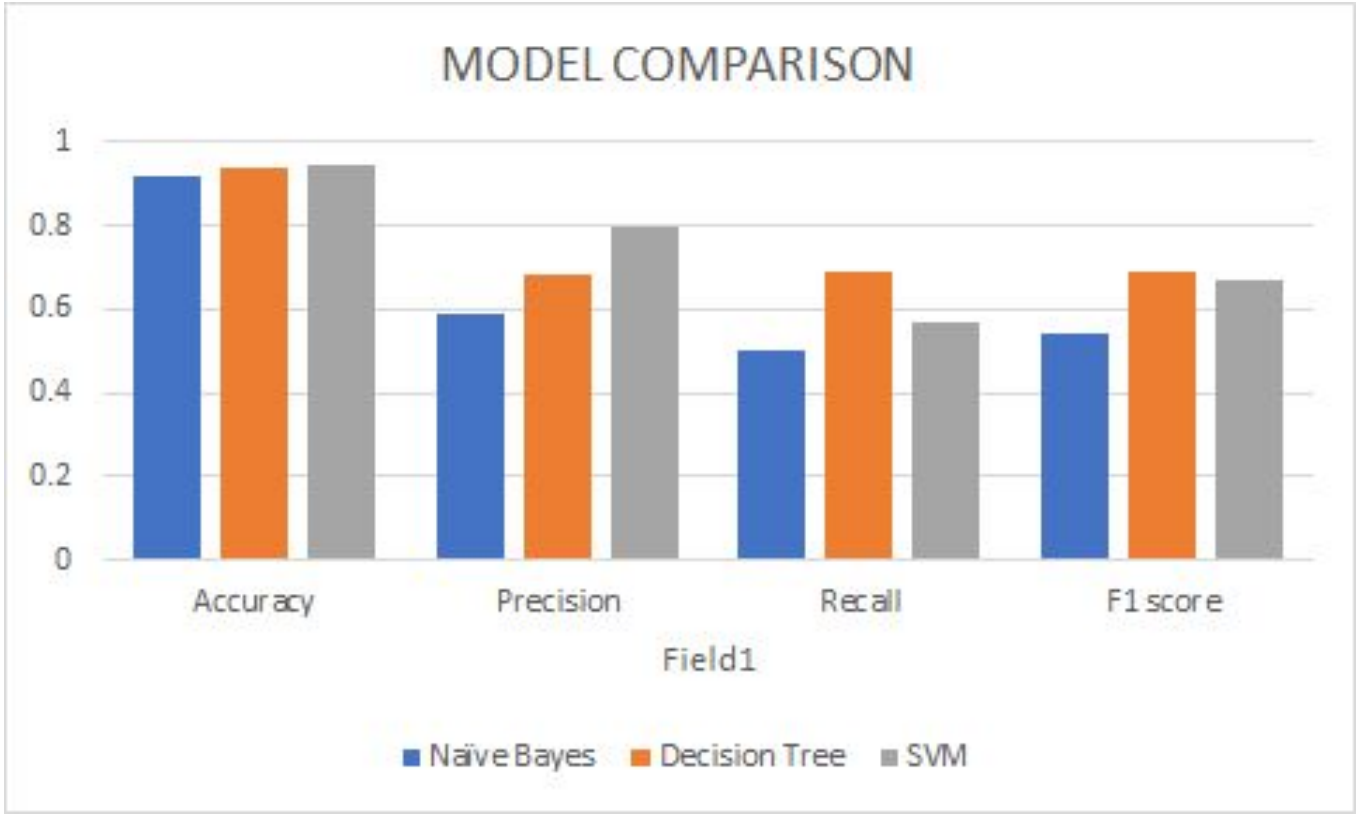
In [25]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred3))
```

```
              precision    recall  f1-score   support

           0       0.80      0.57      0.67       567
           1       0.95      0.98      0.97      5100

   micro avg       0.94      0.94      0.94      5667
   macro avg       0.88      0.78      0.82      5667
weighted avg       0.94      0.94      0.94      5667
```

Results


MODEL COMPARISON
1
0.8
0.6
0.4
0.2
0
Accuracy
Precision
Recall
F1 score
Field1
Naïve Bayes
Decision Tree
SVM

Conclusion

- SVM model does the best with an accuracy of 94.26%
- SVM has a better precision value compared to the other models
- Decision Tree classifier has a better recall value
- Overall, SVM is the better model

Recommendations

1. To increase the accuracy we can use more advanced machine learning algorithms like LSTM Neural Networks.
2. We can try a different preprocessing technique like lemmatization, where the words retain their actual form.
3. We can try a different Feature selection method like the TF-IDF model(term frequency- inverse document frequency)

Any Questions?