# SUMMER PROJECT REPORT

**SACHIN SINGH (18135082)**

**B.Tech in Mechanical Engineering**

**Under Supervision Of:**

**<u>Dr. Saurabh Pratap</u>**

**Department of Mechanical Engineering**

**IIT BHU**

# Data analysis of amazon review dataset

## Description

Every day we come across various products in our lives, on different things across various online platforms, we swipe across hundreds of product choices for similar product under one category. People usually check the reviews of a particular product to be certain of what they could expect from that product.
While this process of going through ‘reviews’ can be difficult, some disadvantageous reviews, also known as spams, makes it even harder for them to decide whether the product is useful or not.

Also product reviews prediction for coming years on the basis of previous year data is very essential for the future production of product. Using the previous data and Using machine learning model most particularly regression model we can predict spam and non spam comments i.e. whether this product’s reviews are useful or not for the given product and determine the overall value of the product in near future.

# Machine Learning

## Abstract :

In this Machine learning Project , we are provided with the real world dataset. This dataset from Amazon contains approx 3 million reviews. Now we were asked to experiment with this data and explore how various machine learning algorithms can be used to find the pattern in data. We were then expected to find which feature of data is important through **Data visualization** and were expected to submit a report about the data set and the various algorithms used and the **accuracy** our model achieved.

## Introduction :

### Machine Learning :

Machine Learning is the science of programming computers so they can learn from data. Machine Learning can be thought of as the study of a list of sub problems, decision making, clustering, classification, forecasting, deep-learning, inductive logic programming, support vector machines, reinforcement learning, similarity and metric learning, genetic algorithms, sparse dictionary learning. Supervised learning, or classification is the machine learning task of inferring a function from a labeled data.

There are so many different types of Machine Learning systems that it is useful to classify them in broad categories based on:

• Whether or not they are trained with human supervision (supervised, unsupervised, semisupervised , and Reinforcement Learning).

• Whether or not they can learn incrementally on the fly ( online versus batch learning).

• Whether they work by simply comparing new data points to known data points, or instead detect patterns in the training data and build a predictive model, much like scientists do (instance-based versus model-based learning).

## 3. Supervised Learning

In supervised learning, the training data you feed to the algorithm includes the desired solutions, called Output or labels.
Various important supervised learning algorithm are k-Nearest Neighbors,Linear Regression, Support Vector Machines ( SVMs) , Decision Trees and Random Forests , Logistic Regression.

## Problems and issues in Supervised Learning:

- Curse of Dimensionality:

In our dataset , input space has a large number of dimensions, and the problem only depends on small subspace of the input space with small dimensions. Input size if huge like it contains approx 3 million rows, So We are using k-fold operation to overcome this.

- Overfitting

If our model performs well on the Training data but it doesn't generalize well on real data or test then this problem is known as overfitting . Complex models like high – degree polynomial can strongly overfits the training data .

- Underfitting

Its the opposite of overfitting, it generally occurs when your model is too simple to learn the underlying structure or pattern of the data. To fix this we can use a more powerful model with more parameters or by feeding better features to the learning algorithm ( **Feature Engnieering )** or by reducing the regularization hyper-parameter .

- Poor - Quality Data

If the Training data is full of errors , outliers and noise then it will make it harder for the system to detect the patterns and best features so our model performance going to decrease . So it is worth to spend some time in cleaning up your training data. And we are going to spend most of the time in doing that .

## Libraries and Module used :

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
```

### Pandas ( pd )

Pandas is an open_source Python Library providing high performance data manipulation and analysis tool using its powerful data structure. It contains time series functionality , fast an efficient DataFrame object with default and customised indexing , tools for loading data from other formats like our data is present in **Json** format which we need to change into simple pandas data frame. Group by data for aggregation and transformations is an excellent feature in pandas.

### Scikit – Learn ( Sklearn )

Scikit- learn or sklearn in short is the most useful and robust library for machine learning in Python . It provides a selection of efficient tools for machine learning and statiscal modelling including classification , regression , clustering and dimensionality reduction . This library is built mainly upon **Numpy , Scipy** and **Matplotlib**.

### Matplotlib

We are using matplotlib package for data visulation . Matplotlib is a comprehensive library for creating static, animated and interative visulations in Python .
As data visualization is the most important part of business activities. There is a huge data in all over the world . All these data's insights help us to interpret which is the best thing to do next related to data. Like nowadays , **Corona vaccine** second dose time is choosen with the help of visualizion of the data.

### Natural Language Processing ( NLP )

NLP is the ability of a computer program to understand human language as it is spoken and written referred to as natural language . It is a component of Artificial intelligence .NLP enables computers to understand natural language . It takes real world input , process it, and make sense of it in a way a computer can understand. At some point in processing , the input is converted to code that the computer can understand .

## Jupyter Notebook

The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

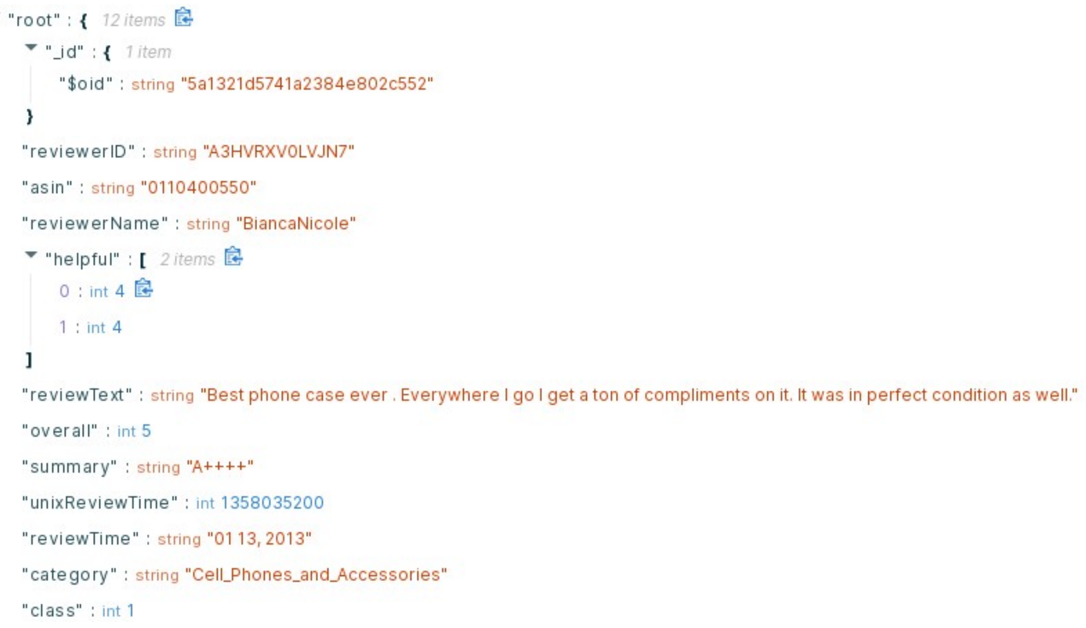
# IMPLEMENTATION

**# Importing the data**

We import the dataset using json module of Python than we convert this data into pandas data frame as pandas dataframe are fast as it takes less time in various operation over columns and tables.

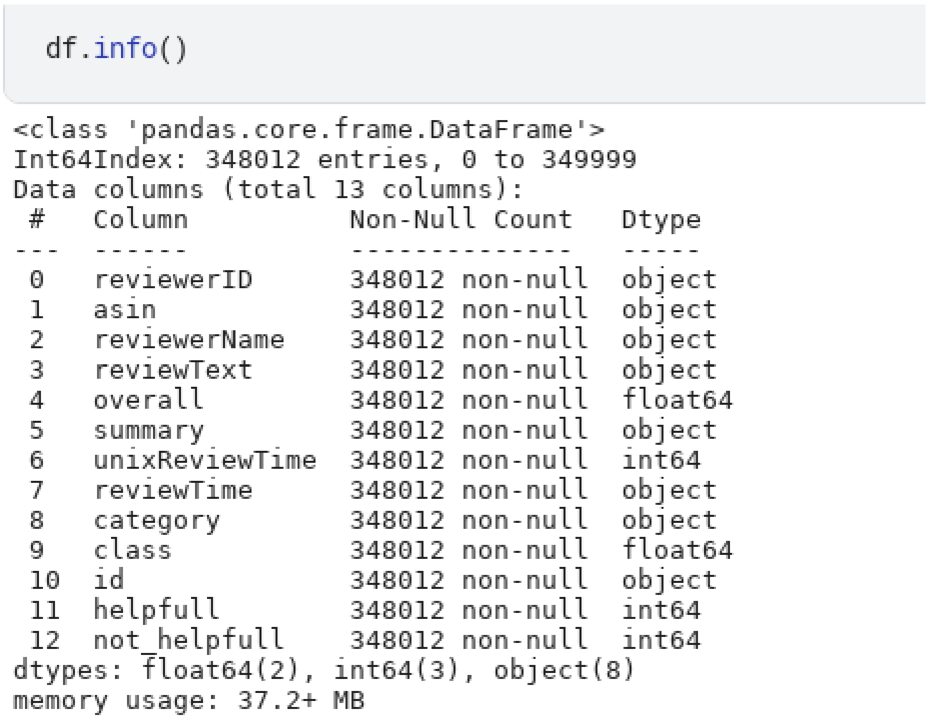
```
import json
from pandas.io.json import json_normalize
with open('Cell_Phones_and_Accessories/Cell_Phones_and_Accessories.json') as json_file:
    data = json_file.readlines()
    data = list(map(json.loads, data))
```

# Data Preview



```
"root" : { 12 items
  "_id" : { 1 item
      "$oid" : string "5a1321d5741a2384e802c552"
  }
  "reviewerID" : string "A3HVRXV0LVJN7"
  "asin" : string "0110400550"
  "reviewerName" : string "BiancaNicole"
  "helpful" : [ 2 items
      0 : int 4
      1 : int 4
  ]
  "reviewText" : string "Best phone case ever . Everywhere I go I get a ton of compliments on it. It was in perfect condition as well."
  "overall" : int 5
  "summary" : string "A++++"
  "unixReviewTime" : int 1358035200
  "reviewTime" : string "01 13, 2013"
  "category" : string "Cell_Phones_and_Accessories"
  "class" : int 1
```

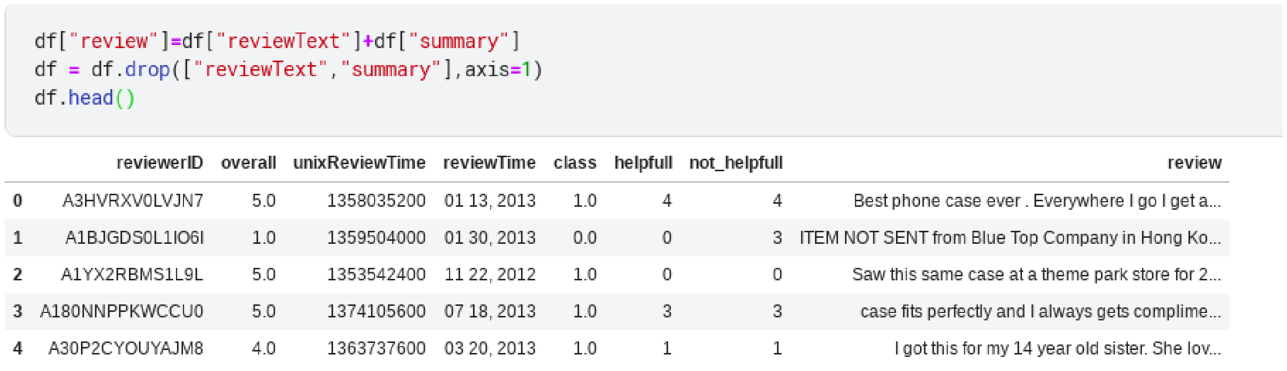
**Dataset Info**



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 348012 entries, 0 to 349999
Data columns (total 13 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   reviewerID      348012 non-null  object
 1   asin            348012 non-null  object
 2   reviewerName    348012 non-null  object
 3   reviewText      348012 non-null  object
 4   overall         348012 non-null  float64
 5   summary         348012 non-null  object
 6   unixReviewTime  348012 non-null  int64
 7   reviewTime      348012 non-null  object
 8   category        348012 non-null  object
 9   class           348012 non-null  float64
 10  id              348012 non-null  object
 11  helpfull        348012 non-null  int64
 12  not_helpfull    348012 non-null  int64
dtypes: float64(2), int64(3), object(8)
memory usage: 37.2+ MB
```

# After removing reviewer name ,category ,id  ,asin  and merging review & summary
  Now  Concatenating Review Text and Summary Column into a single column review

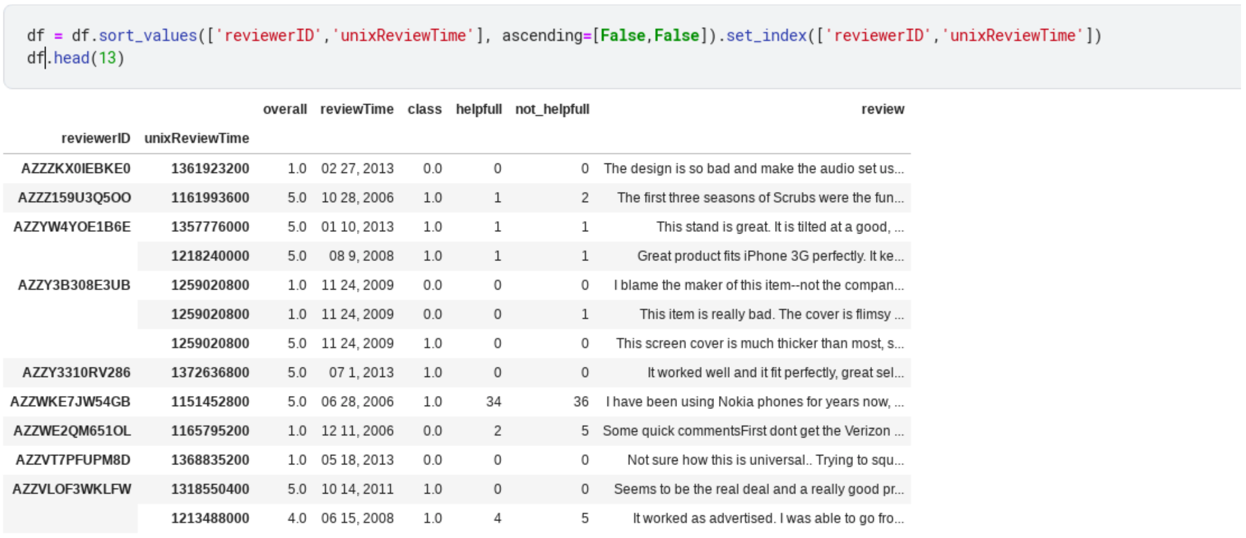


```
df["review"]=df["reviewText"]+df["summary"]
df = df.drop(["reviewText","summary"],axis=1)
df.head()
```

| | reviewerID | overall | unixReviewTime | reviewTime | class | helpfull | not_helpfull | review |
|---|---|---|---|---|---|---|---|---|
| **0** | A3HVRXV0LVJN7 | 5.0 | 1358035200 | 01 13, 2013 | 1.0 | 4 | 4 | Best phone case ever . Everywhere I go I get a... |
| **1** | A1BJGDS0L1IO6I | 1.0 | 1359504000 | 01 30, 2013 | 0.0 | 0 | 3 | ITEM NOT SENT from Blue Top Company in Hong Ko... |
| **2** | A1YX2RBMS1L9L | 5.0 | 1353542400 | 11 22, 2012 | 1.0 | 0 | 0 | Saw this same case at a theme park store for 2... |
| **3** | A180NNPPKWCCU0 | 5.0 | 1374105600 | 07 18, 2013 | 1.0 | 3 | 3 | case fits perfectly and I always gets complime... |
| **4** | A30P2CYOUYAJM8 | 4.0 | 1363737600 | 03 20, 2013 | 1.0 | 1 | 1 | I got this for my 14 year old sister. She lov... |

## # Sorting

Now Sort the reviewer id column by time as one reviewer may right more than review. Now are Data look like this

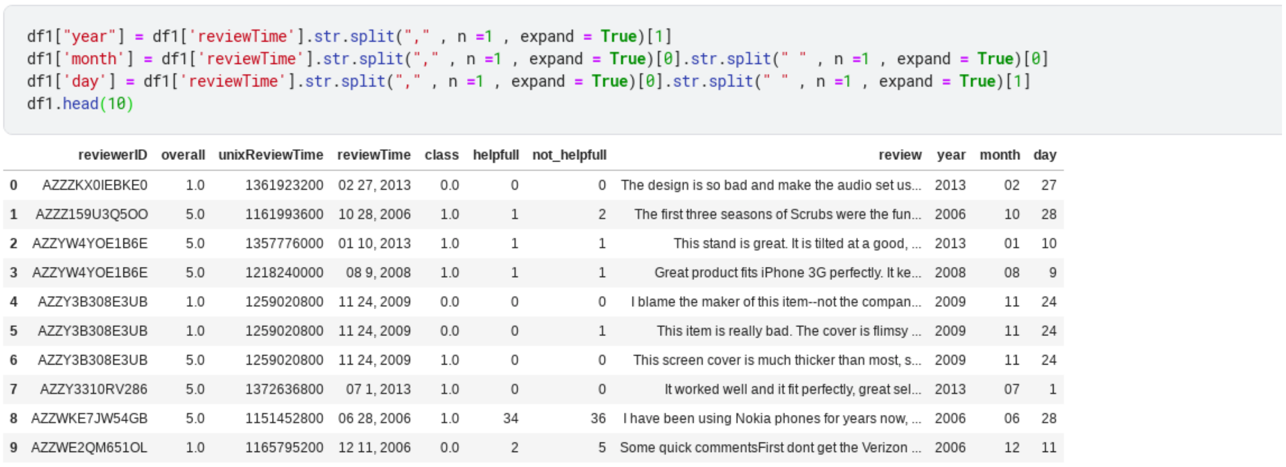


```
df = df.sort_values(['reviewerID','unixReviewTime'], ascending=[False,False]).set_index(['reviewerID','unixReviewTime'])
df.head(13)
```

| | | overall | reviewTime | class | helpfull | not_helpfull | review |
|---|---|---|---|---|---|---|---|
| reviewerID | unixReviewTime | | | | | | |
| AZZZKX0IEBKE0 | 1361923200 | 1.0 | 02 27, 2013 | 0.0 | 0 | 0 | The design is so bad and make the audio set us... |
| AZZZ159U3Q5OO | 1161993600 | 5.0 | 10 28, 2006 | 1.0 | 1 | 2 | The first three seasons of Scrubs were the fun... |
| AZZYW4YOE1B6E | 1357776000 | 5.0 | 01 10, 2013 | 1.0 | 1 | 1 | This stand is great. It is tilted at a good, ... |
| | 1218240000 | 5.0 | 08 9, 2008 | 1.0 | 1 | 1 | Great product fits iPhone 3G perfectly. It ke... |
| AZZY3B308E3UB | 1259020800 | 1.0 | 11 24, 2009 | 0.0 | 0 | 0 | I blame the maker of this item--not the compan... |
| | 1259020800 | 1.0 | 11 24, 2009 | 0.0 | 0 | 1 | This item is really bad. The cover is flimsy ... |
| | 1259020800 | 5.0 | 11 24, 2009 | 1.0 | 0 | 0 | This screen cover is much thicker than most, s... |
| AZZY3310RV286 | 1372636800 | 5.0 | 07 1, 2013 | 1.0 | 0 | 0 | It worked well and it fit perfectly, great sel... |
| AZZWKE7JW54GB | 1151452800 | 5.0 | 06 28, 2006 | 1.0 | 34 | 36 | I have been using Nokia phones for years now, ... |
| AZZWE2QM651OL | 1165795200 | 1.0 | 12 11, 2006 | 0.0 | 2 | 5 | Some quick commentsFirst dont get the Verizon ... |
| AZZVT7PFUPM8D | 1368835200 | 1.0 | 05 18, 2013 | 0.0 | 0 | 0 | Not sure how this is universal.. Trying to squ... |
| AZZVLOF3WKLFW | 1318550400 | 5.0 | 10 14, 2011 | 1.0 | 0 | 0 | Seems to be the real deal and a really good pr... |
| | 1213488000 | 4.0 | 06 15, 2008 | 1.0 | 4 | 5 | It worked as advertised. I was able to go fro... |

## # Changing time format

We separated the review time to day, month and year column.

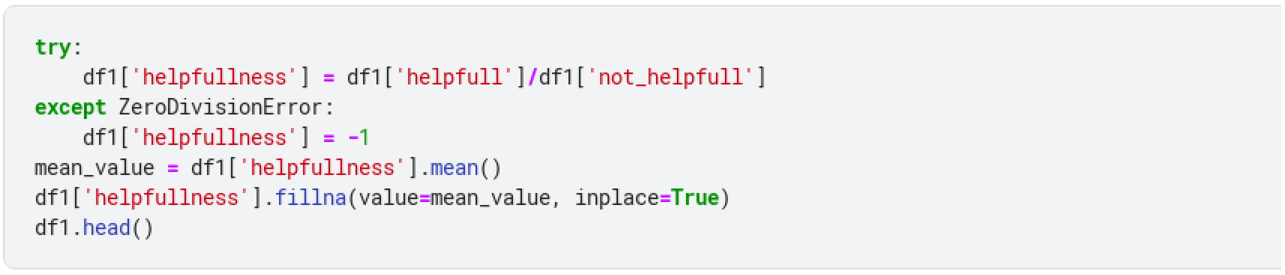


```
df1["year"] = df1['reviewTime'].str.split("," , n =1 , expand = True)[1]
df1['month'] = df1['reviewTime'].str.split("," , n =1 , expand = True)[0].str.split(" " , n =1 , expand = True)[0]
df1['day'] = df1['reviewTime'].str.split("," , n =1 , expand = True)[0].str.split(" " , n =1 , expand = True)[1]
df1.head(10)
```

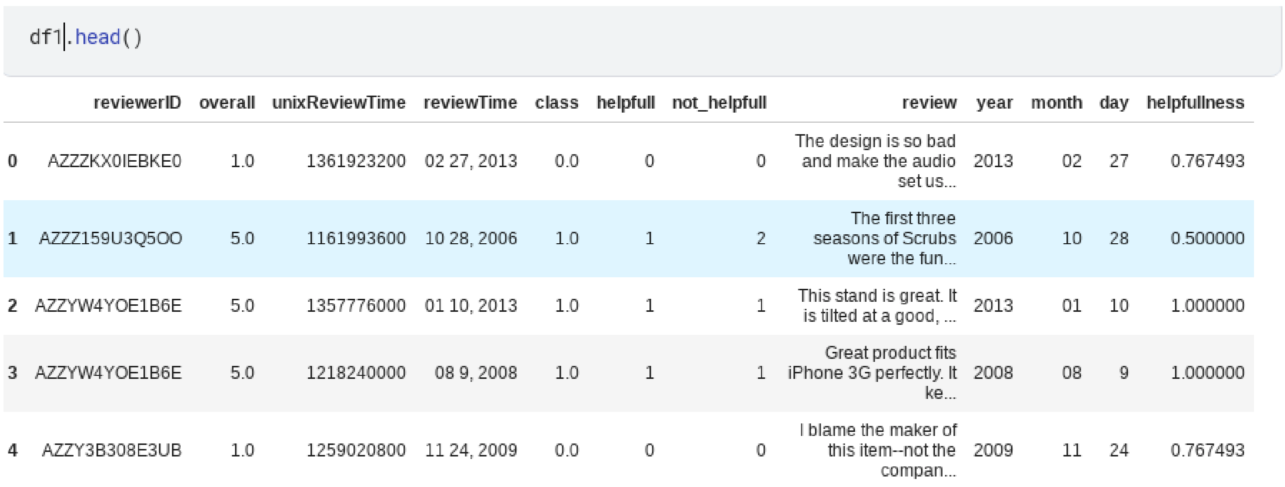
| | reviewerID | overall | unixReviewTime | reviewTime | class | helpfull | not_helpfull | review | year | month | day |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | AZZZKX0IEBKE0 | 1.0 | 1361923200 | 02 27, 2013 | 0.0 | 0 | 0 | The design is so bad and make the audio set us... | 2013 | 02 | 27 |
| 1 | AZZZ159U3Q5OO | 5.0 | 1161993600 | 10 28, 2006 | 1.0 | 1 | 2 | The first three seasons of Scrubs were the fun... | 2006 | 10 | 28 |
| 2 | AZZYW4YOE1B6E | 5.0 | 1357776000 | 01 10, 2013 | 1.0 | 1 | 1 | This stand is great. It is tilted at a good, ... | 2013 | 01 | 10 |
| 3 | AZZYW4YOE1B6E | 5.0 | 1218240000 | 08 9, 2008 | 1.0 | 1 | 1 | Great product fits iPhone 3G perfectly. It ke... | 2008 | 08 | 9 |
| 4 | AZZY3B308E3UB | 1.0 | 1259020800 | 11 24, 2009 | 0.0 | 0 | 0 | I blame the maker of this item--not the compan... | 2009 | 11 | 24 |
| 5 | AZZY3B308E3UB | 1.0 | 1259020800 | 11 24, 2009 | 0.0 | 0 | 1 | This item is really bad. The cover is flimsy ... | 2009 | 11 | 24 |
| 6 | AZZY3B308E3UB | 5.0 | 1259020800 | 11 24, 2009 | 1.0 | 0 | 0 | This screen cover is much thicker than most, s... | 2009 | 11 | 24 |
| 7 | AZZY3310RV286 | 5.0 | 1372636800 | 07 1, 2013 | 1.0 | 0 | 0 | It worked well and it fit perfectly, great sel... | 2013 | 07 | 1 |
| 8 | AZZWKE7JW54GB | 5.0 | 1151452800 | 06 28, 2006 | 1.0 | 34 | 36 | I have been using Nokia phones for years now, ... | 2006 | 06 | 28 |
| 9 | AZZWE2QM651OL | 1.0 | 1165795200 | 12 11, 2006 | 0.0 | 2 | 5 | Some quick commentsFirst dont get the Verizon ... | 2006 | 12 | 11 |

# **Helpfullness from helpful column**

In the main dataframe we are given the helpful and not helpful column which means that a out of b people find that helpful. But there is a problem as many not helpful column contains nan values so we need to care about the zero Division Error and then fill all the nan values in helpfullness column with the mean of the column . And after that we can remove helpful and not helpful column .



```
try:
    df1['helpfullness'] = df1['helpfull']/df1['not_helpfull']
except ZeroDivisionError:
    df1['helpfullness'] = -1
mean_value = df1['helpfullness'].mean()
df1['helpfullness'].fillna(value=mean_value, inplace=True)
df1.head()
```

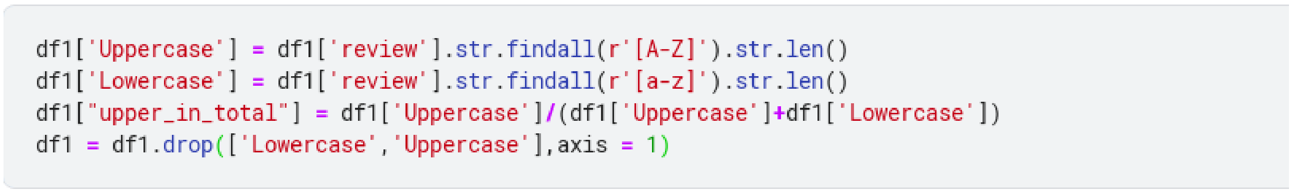


```
df1.head()
```

| | reviewerID | overall | unixReviewTime | reviewTime | class | helpfull | not_helpfull | review | year | month | day | helpfullness |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | AZZZKX0IEBKE0 | 1.0 | 1361923200 | 02 27, 2013 | 0.0 | 0 | 0 | The design is so bad and make the audio set us... | 2013 | 02 | 27 | 0.767493 |
| **1** | AZZZ159U3Q5OO | 5.0 | 1161993600 | 10 28, 2006 | 1.0 | 1 | 2 | The first three seasons of Scrubs were the fun... | 2006 | 10 | 28 | 0.500000 |
| **2** | AZZYW4YOE1B6E | 5.0 | 1357776000 | 01 10, 2013 | 1.0 | 1 | 1 | This stand is great. It is tilted at a good, ... | 2013 | 01 | 10 | 1.000000 |
| **3** | AZZYW4YOE1B6E | 5.0 | 1218240000 | 08 9, 2008 | 1.0 | 1 | 1 | Great product fits iPhone 3G perfectly. It ke... | 2008 | 08 | 9 | 1.000000 |
| **4** | AZZY3B308E3UB | 1.0 | 1259020800 | 11 24, 2009 | 0.0 | 0 | 0 | I blame the maker of this item--not the compan... | 2009 | 11 | 24 | 0.767493 |

# Count Capital letters

We found out that class column has positive correlation with capital letters. Sentences with more ratios of capital letters may grab the attention of readers and it is also unusual to have many capitals in a sentence. Hence we are counting the ratio of total capital letters to all letters.



```
df1['Uppercase'] = df1['review'].str.findall(r'[A-Z]').str.len()
df1['Lowercase'] = df1['review'].str.findall(r'[a-z]').str.len()
df1["upper_in_total"] = df1['Uppercase']/(df1['Uppercase']+df1['Lowercase'])
df1 = df1.drop(['Lowercase','Uppercase'],axis = 1)
```

## # Cleaning the text and Removing the stop words

Stop words are commonly used words so we can ignore them in our text without losing the data meaning

```
stop_words= ['yourselves', 'between', 'whom', 'itself', 'is', "she's", 'up', 'herself', 'here', 'your', 'each'
 'we', 'he', 'my', "you've", 'having', 'in', 'both', 'for', 'themselves', 'are', 'them', 'other',
 'and', 'an', 'during', 'their', 'can', 'yourself', 'she', 'until', 'so', 'these', 'ours', 'above',
 'what', 'while', 'have', 're', 'more', 'only', "needn't", 'when', 'just', 'that', 'were', "don't",
 'very', 'should', 'any', 'y', 'isn', 'who', 'a', 'they', 'to', 'too', "should've", 'has', 'before',
 'into', 'yours', "it's", 'do', 'against', 'on', 'now', 'her', 've', 'd', 'by', 'am', 'from',
 'about', 'further', "that'll", "you'd", 'you', 'as', 'how', 'been', 'the', 'or', 'doing', 'such',
 'his', 'himself', 'ourselves', 'was', 'through', 'out', 'below', 'own', 'myself', 'theirs',
 'me', 'why', 'once', 'him', 'than', 'be', 'most', "you'll", 'same', 'some', 'with', 'few', 'it',
 'at', 'after', 'its', 'which', 'there','our', 'this', 'hers', 'being', 'did', 'of', 'had', 'under',
 'over','again', 'where', 'those', 'then', "you're", 'i', 'because', 'does', 'all']
```

And after this our data will look like this

| | reviewerID | overall | unixReviewTime | class | review | year | month | day | helpfullness | upper_in_tot |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | AZZZKX0IEBKE0 | 1.0 | 1361923200 | 0.0 | design bad make audio set useless cave man not... | 2013 | 02 | 27 | 0.767493 | 0.020979 |
| **1** | AZZZ159U3Q5OO | 5.0 | 1161993600 | 1.0 | first three seasons scrubs funniest ones one f... | 2006 | 10 | 28 | 0.500000 | 0.025959 |
| **2** | AZZYW4YOE1B6E | 5.0 | 1357776000 | 1.0 | stand great tilted good but fixed angle perfec... | 2013 | 01 | 10 | 1.000000 | 0.039801 |
| **3** | AZZYW4YOE1B6E | 5.0 | 1218240000 | 1.0 | great product fits iphone perfectly keeps secu... | 2008 | 08 | 9 | 1.000000 | 0.070175 |
| **4** | AZZY3B308E3UB | 1.0 | 1259020800 | 0.0 | blame maker itemnot company sold itthe item ar... | 2009 | 11 | 24 | 0.767493 | 0.024055 |

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

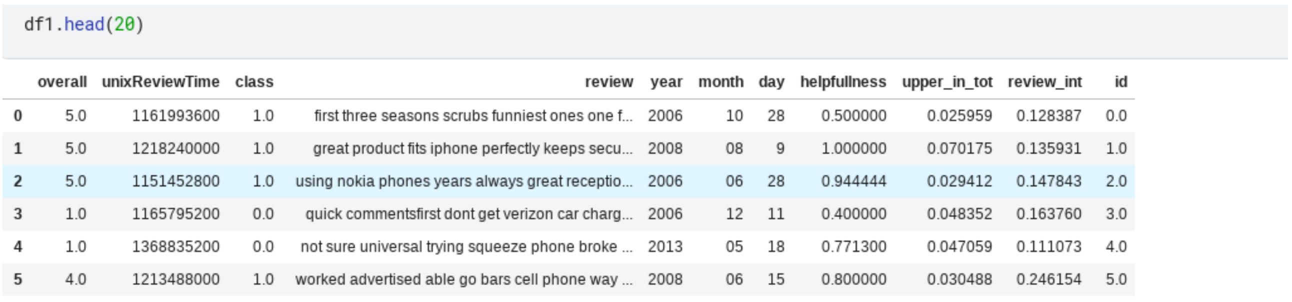
With the help of this technique we can quanitify a word in a document , so we can compute weight to each quantified word and this shows the importance of that word in the document . In our data we are splitting our text data as bi-gram (two words combination) and considered their combine weight .

```
from sklearn.feature_extraction.text import TfidfVectorizer
tf_idf_vectorizer = TfidfVectorizer(max_features=5000,ngram_range=(2,2))
text = tf_idf_vectorizer.fit_transform(df['review'])
tf_idf_vectorizer.vocabulary_
```

```
{'best phone': 545,
 'phone case': 2983,
 'compliments on': 909,
 'on it': 2747,
 'it it': 2006,
 'it was': 2084,
 'was in': 4579,
 'in perfect': 1765,
 'perfect condition': 2954,
 'as well': 399,
 'and it': 215,
 'it been': 1951,
 'two months': 4397,
 'do not': 1021,
 'not use': 2629,
 'use this': 4452,
```

After this we are adding some more columns like **review_int** which is basically the differnce of positive and negative bigrams in the reviewText of respective rows. Here by positive and negative bigram  we mean that which are inclined more toward positive and negative class.

Then we are making new column id in place of reviewerID as reviewerID is an object column and our machine learning model may feel difficulty in processing that.

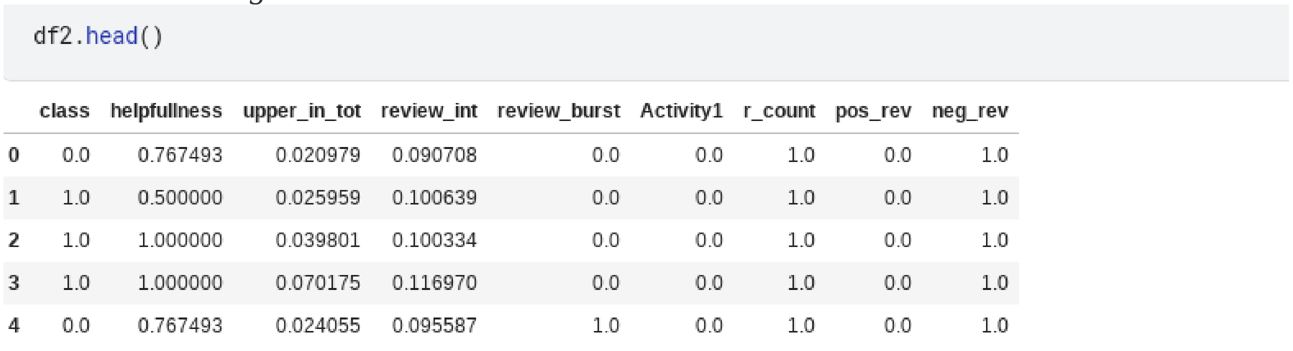


```
df1.head(20)
```

| | overall | unixReviewTime | class | review | year | month | day | helpfullness | upper_in_tot | review_int | id |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 5.0 | 1161993600 | 1.0 | first three seasons scrubs funniest ones one f... | 2006 | 10 | 28 | 0.500000 | 0.025959 | 0.128387 | 0.0 |
| 1 | 5.0 | 1218240000 | 1.0 | great product fits iphone perfectly keeps secu... | 2008 | 08 | 9 | 1.000000 | 0.070175 | 0.135931 | 1.0 |
| 2 | 5.0 | 1151452800 | 1.0 | using nokia phones years always great receptio... | 2006 | 06 | 28 | 0.944444 | 0.029412 | 0.147843 | 2.0 |
| 3 | 1.0 | 1165795200 | 0.0 | quick commentsfirst dont get verizon car charg... | 2006 | 12 | 11 | 0.400000 | 0.048352 | 0.163760 | 3.0 |
| 4 | 1.0 | 1368835200 | 0.0 | not sure universal trying squeeze phone broke ... | 2013 | 05 | 18 | 0.771300 | 0.047059 | 0.111073 | 4.0 |
| 5 | 4.0 | 1213488000 | 1.0 | worked advertised able go bars cell phone way ... | 2008 | 06 | 15 | 0.800000 | 0.030488 | 0.246154 | 5.0 |

Then we are adding one more column of which calculates the difference between the timestamps of the first and last reviews of an author to find out the number of active days an author had been on the system and we also adding the number positive and negative rating by an author and now we don't need the date and Time  column so we are removing that.

After all cleaning our data will look like this.



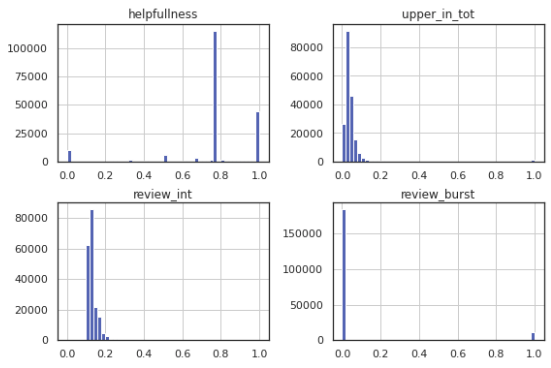
```
df2.head()
```

| | class | helpfullness | upper_in_tot | review_int | review_burst | Activity1 | r_count | pos_rev | neg_rev |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.0 | 0.767493 | 0.020979 | 0.090708 | 0.0 | 0.0 | 1.0 | 0.0 | 1.0 |
| 1 | 1.0 | 0.500000 | 0.025959 | 0.100639 | 0.0 | 0.0 | 1.0 | 0.0 | 1.0 |
| 2 | 1.0 | 1.000000 | 0.039801 | 0.100334 | 0.0 | 0.0 | 1.0 | 0.0 | 1.0 |
| 3 | 1.0 | 1.000000 | 0.070175 | 0.116970 | 0.0 | 0.0 | 1.0 | 0.0 | 1.0 |
| 4 | 0.0 | 0.767493 | 0.024055 | 0.095587 | 1.0 | 0.0 | 1.0 | 0.0 | 1.0 |

## # **Data Visualization**

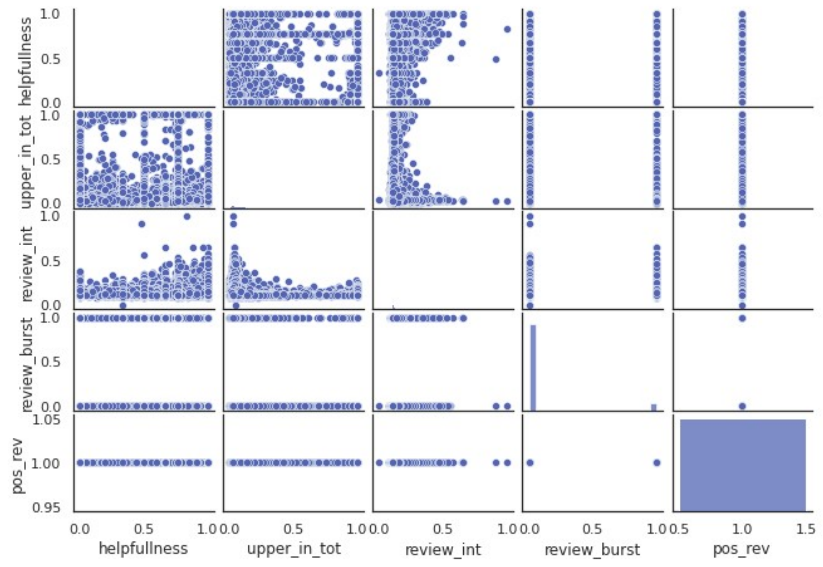
This histogram shows the number  of instances(on the vertical axis)  that have a given value range ( on the horizontal axis ) Frm

```
X.hist(bins = 50 , figsize = (9,6))
plt.show()
```



From figure we can conclude that the most of  the values in upper_in_total , review_int and review_burstiness are inclined toward 0. where as in figure of helpfullness most of the values are inclined toward 0.7 (because of the mean of the values).

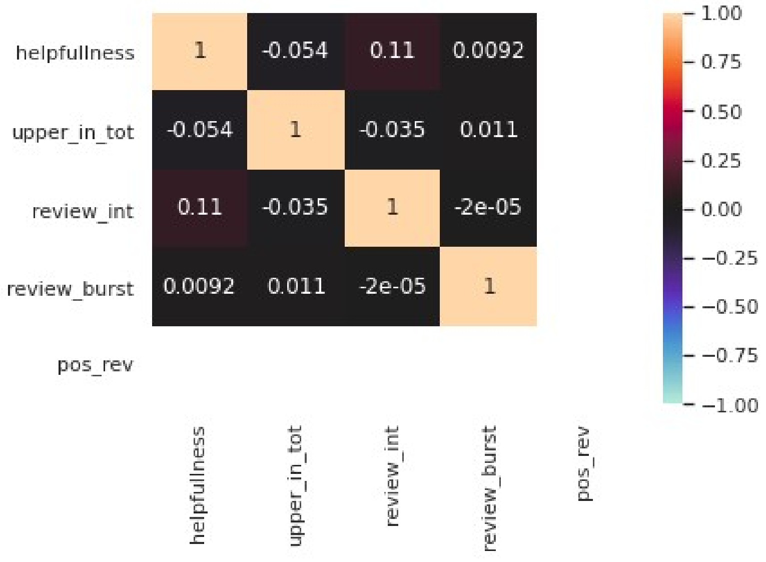
# Plotting every numerical values against every other numerical attributes .



# HeatMap of Correlation Matrix

```python
sns.heatmap(corr, annot = True, vmin=-1, vmax=1, center= 0)
```

```
<AxesSubplot:>
```



|  | helpfullness | upper_in_tot | review_int | review_burst | pos_rev |
|---|---|---|---|---|---|
| helpfullness | 1 | -0.054 | 0.11 | 0.0092 |  |
| upper_in_tot | -0.054 | 1 | -0.035 | 0.011 |  |
| review_int | 0.11 | -0.035 | 1 | -2e-05 |  |
| review_burst | 0.0092 | 0.011 | -2e-05 | 1 |  |
| pos_rev |  |  |  |  |  |

```
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
    thresh = cm.max() / 2.
    for i in range (cm.shape[0]):
       for j in range (cm.shape[1]):
            plt.text(j, i, cm[i, j],
                horizontalalignment="center",
                color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

In [89]:

```
from sklearn.linear_model import LogisticRegression
model_2= LogisticRegression(random_state=42)
model_2.fit(train_X,train_y)
pred_y = model_2.predict(test_X)
error = mean_squared_error(pred_y,test_y)
rmse = np.sqrt(error)
print(rmse)
```

```
0.0
```

In [90]:

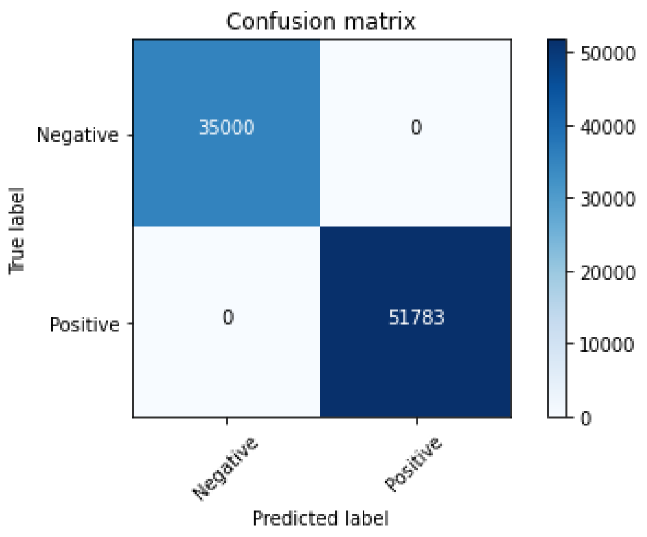
```
print(len(pred_y))
```

```
86783
```

In [91]:

```
from sklearn import metrics
cm = metrics.confusion_matrix(test_y, pred_y)
plot_confusion_matrix(cm, classes=['Negative','Positive'])
```

```
Confusion matrix, without normalization
```



In [92]:

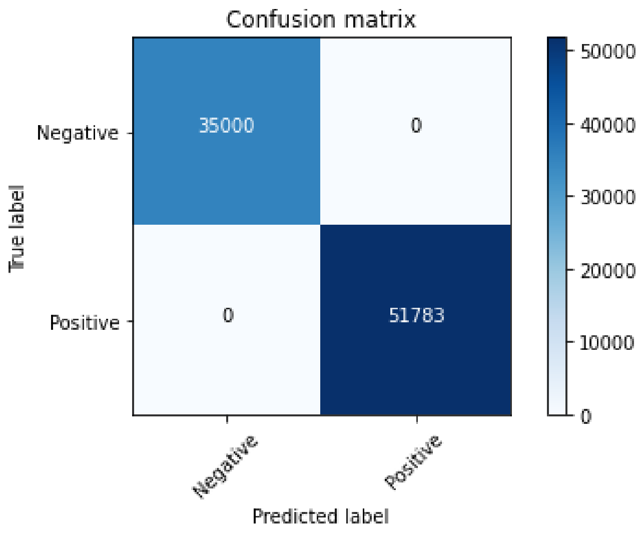
```
from sklearn.naive_bayes import GaussianNB
model_3 = GaussianNB()
model_3.fit(train_X,train_y)
pred_y = model_3.predict(test_X)
error = mean_squared_error(pred_y,test_y)
rmse = np.sqrt(error)
print(rmse)
```

```
0.0
```

In [93]:

```
from sklearn import metrics
cm = metrics.confusion_matrix(test_y, pred_y)
plot_confusion_matrix(cm, classes=['Negative','Positive'])
```

```
Confusion matrix, without normalization
```



In [94]:

```
from sklearn.linear_model import SGDClassifier
model_4 = SGDClassifier(loss="hinge", penalty="l2", max_iter=5)
model_4.fit(train_X,train_y)
pred_y = model_4.predict(test_X)
error = mean_squared_error(pred_y,test_y)
rmse = np.sqrt(error)
print(rmse)
```

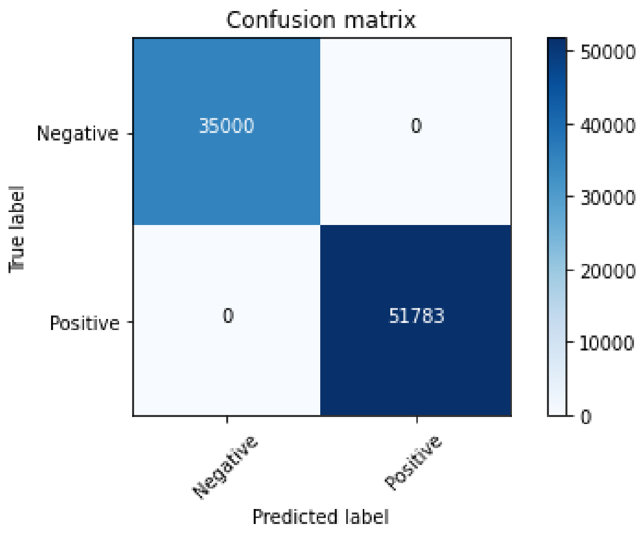
```
0.0
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/linear_model/_stochastic_gradient.py:573:
ConvergenceWarning: Maximum number of iteration reached before convergence. Consider incr
easing max_iter to improve the fit.
  ConvergenceWarning)
```

In [95]:

```
from sklearn import metrics
cm = metrics.confusion_matrix(test_y, pred_y)
plot_confusion_matrix(cm, classes=['Negative','Positive'])
```

```
Confusion matrix, without normalization
```



In [96]:

```
from sklearn.tree import DecisionTreeClassifier
```

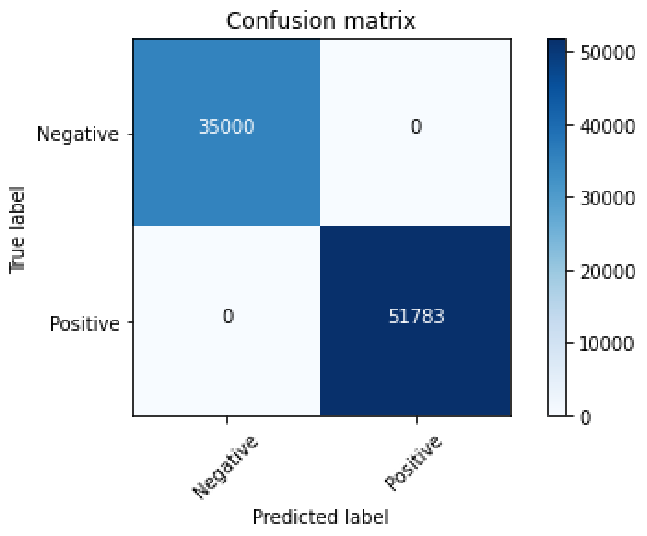
```
model_5 = DecisionTreeClassifier()
model_5.fit(train_X,train_y)
pred_y = model_5.predict(test_X)
error = mean_squared_error(pred_y,test_y)
rmse = np.sqrt(error)
print(rmse)
```

```
0.0
```

In [97]:

```
from sklearn import metrics
cm = metrics.confusion_matrix(test_y, pred_y)
plot_confusion_matrix(cm, classes=['Negative','Positive'])
```

```
Confusion matrix, without normalization
```



In [98]:

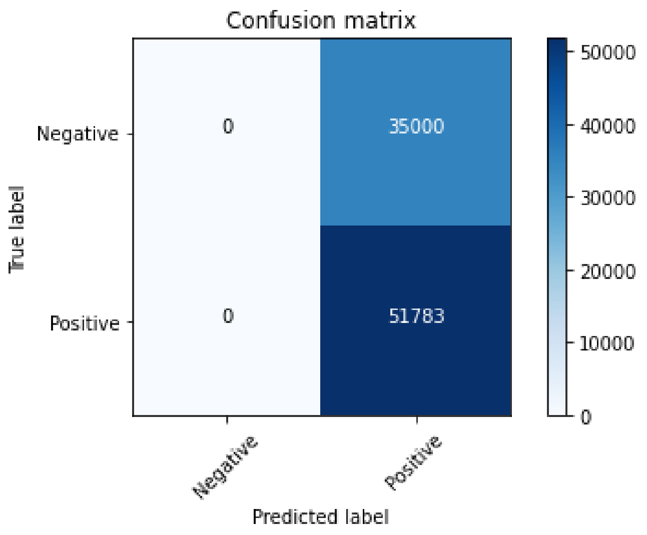
```
from sklearn.naive_bayes import BernoulliNB
model_6 = BernoulliNB()
model_6.fit(train_X,train_y)
pred_y = model_6.predict(test_X)
error = mean_squared_error(pred_y,test_y)
rmse = np.sqrt(error)
print(rmse)
```

```
0.6350628273784095
```

In [99]:

```
from sklearn import metrics
cm = metrics.confusion_matrix(test_y, pred_y)
plot_confusion_matrix(cm, classes=['Negative','Positive'])
```

```
Confusion matrix, without normalization
```



In [100]:

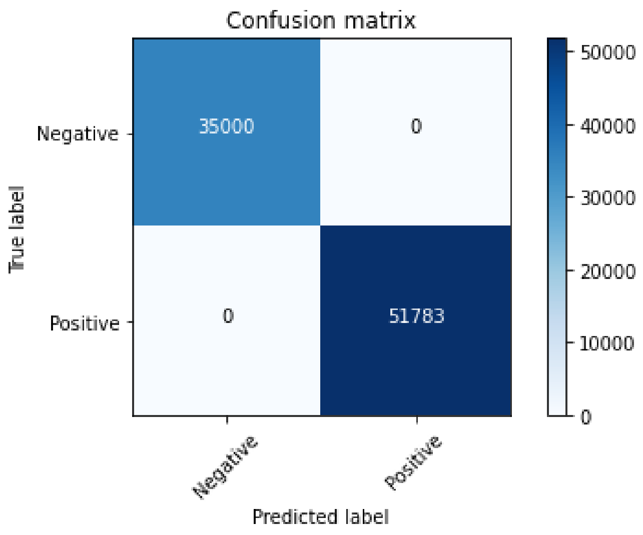
```
from sklearn.neighbors import KNeighborsClassifier
model_7= KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2 )
model_7.fit(X, y)
pred_y = model_7.predict(test_X)
error = mean_squared_error(pred_y,test_y)
rmse = np.sqrt(error)
print(rmse)
```

```
0.0
```

In [101]:

```
from sklearn import metrics
cm = metrics.confusion_matrix(test_y, pred_y)
plot_confusion_matrix(cm, classes=['Negative','Positive'])
```

```
Confusion matrix, without normalization
```



In [102]:

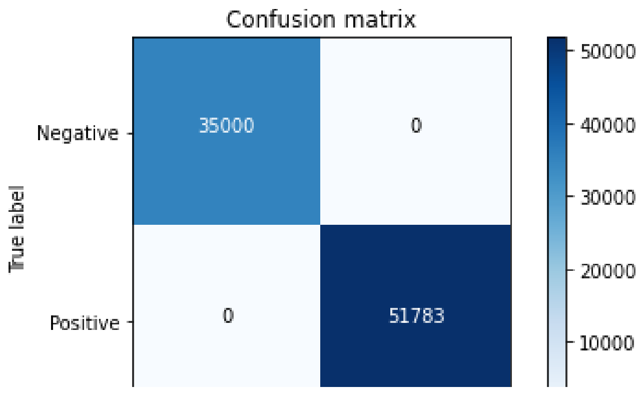
```
from sklearn.neighbors import KNeighborsClassifier
model_7= KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2 )
model_7.fit(train_X, train_y)
pred_y = model_7.predict(test_X)
error = mean_squared_error(pred_y,test_y)
rmse = np.sqrt(error)
print(rmse)
```

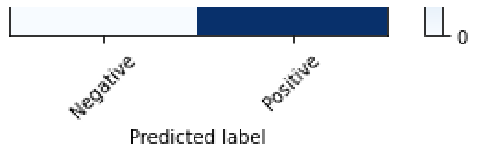
```
0.0
```

In [103]:

```
from sklearn import metrics
cm = metrics.confusion_matrix(test_y, pred_y)
plot_confusion_matrix(cm, classes=['Negative','Positive'])
```

```
Confusion matrix, without normalization
```





In [105]:

```
from sklearn.model_selection import cross_val_score
```

In [106]:

```
model_2= LogisticRegression(random_state=42)
scores = cross_val_score(model_2, X, y, cv=5, scoring='f1_macro')
print(scores)
```

```
[1. 1. 1. 1. 1.]
```

## Conclusion -