
UNIVERSITAT DE BARCELONA
BIG SOLUTION SOLUTIONS
Postgraduate of Data Science & Big Data
Estela Martínez
Tatyana Radchenko
Laura Pareja
Rebeca Font
3 July 2018

SUMMARY

0. Aim

1. Data cleaning and preparation

2. Selecting the best classification model

3. Conclusions

AIMS OF THIS PROJECT

Principal:

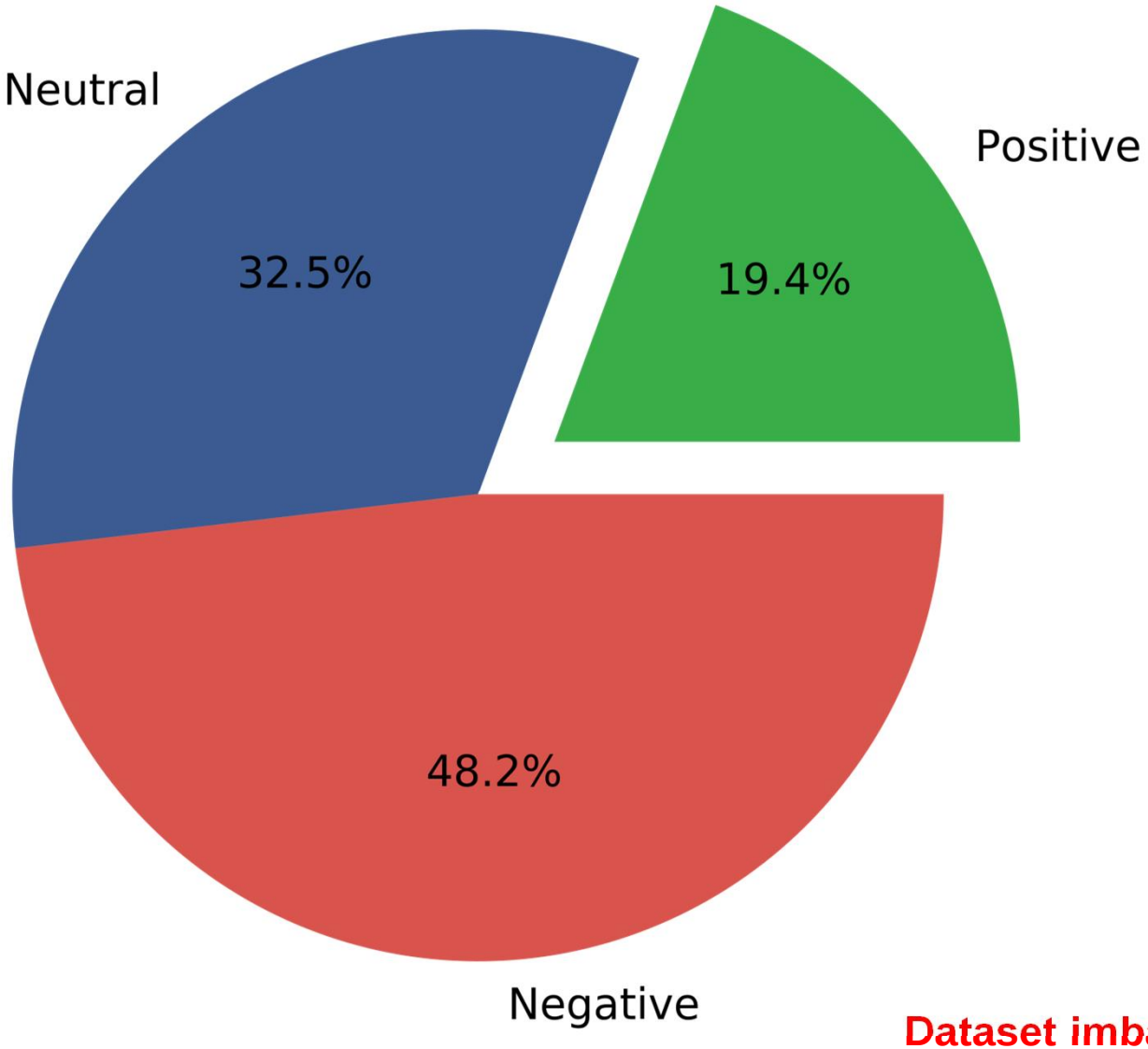
To train a classification model able to classify each twit in three distinct groups: **ópositiveô, ônegativeôor óneutralô**

Secondary:

To make a description of how happy or unhappy the clients were depending on which airline they used.

Set a pattern to properly classify future twits related to Spanish airlines.

DATA DESCRIPTION (I) - Twits by category



Dataset imbalanced!!!

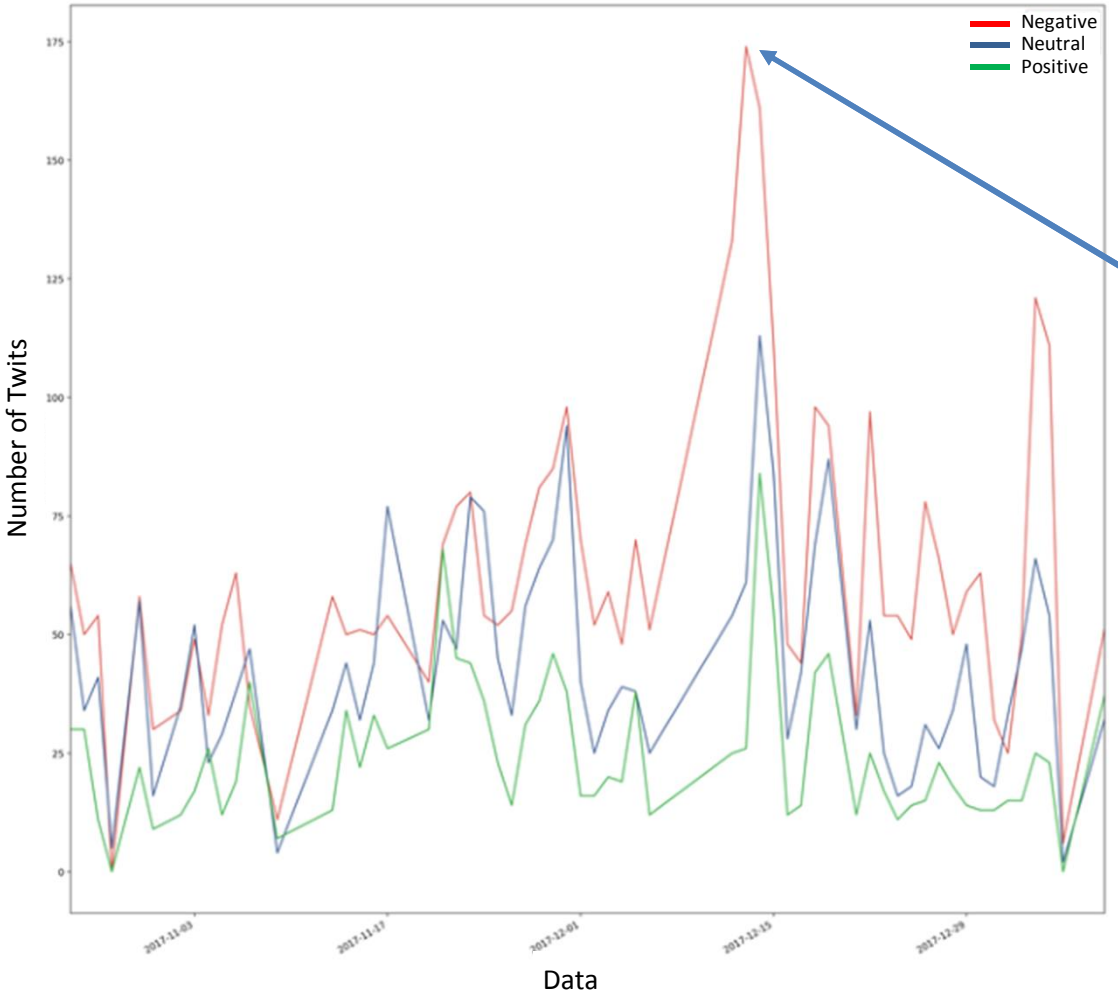
DATA PREPARATION STEP

Initial data set was imbalanced.

Following steps were applied to get balanced data set:

1. Randomly select twits from neutral data set. Amount was equal to amount of twits in positive data set.
2. Randomly select twits from negative data set. Amount was equal to amount of twits in positive data set.
3. Combine positive, new neutral and new negative data sets in one data set for following model training.
4. Train models with initial data set and new data set and compare accuracy score

DATA DESCRIPTION (II) - What happened on 14th of December?



Niki Austrian airline failure strands many passengers

14 December 2017 Share

A Niki jet on the tarmac in Vienna

The Austrian airline Niki has grounded its planes, stranding thousands of passengers, after filing for insolvency protection.

The company has about 20 planes serving resorts in southern Europe and north Africa.

It was founded by Niki Lauda, the Austrian ex-F1 racing champion, but he sold it to Air Berlin in 2011.

Austria's Der Standard daily says about 5,000 Niki passengers are stuck abroad and need to be brought back to Austria.

Most of the stranded passengers are in Majorca. Austrian authorities are now trying to get them transferred to other flights.

Top Stories

Trump narrows down Supreme Court candidates
12 minutes ago

Steel firms Thyssenkrupp and Tata to merge
16 minutes ago

US ambassador quits 'over Trump comments'
5 hours ago

Features

The young Austrian leader sharing power with the far right

Drones, dogs, drilling and desperation

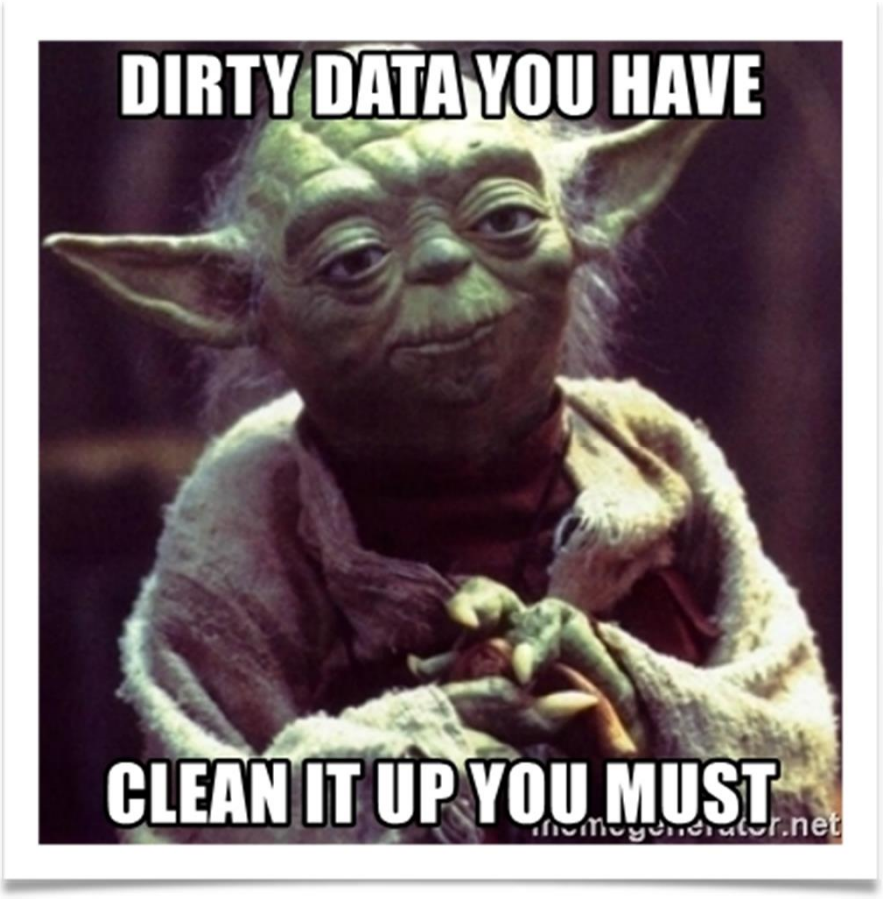
What were the main reasons why people were happy or unhappy with the airline?

DATA DESCRIPTION - THE MOST FREQUENT WORDS

Negative twits

Positive twits

DATA CLEANING



DATA CLEANING

It's a necessary process to ensure data quality.

Essential to **minimize the risk of basing decision-making on information** that is not precise or it's erroneous or incomplete.

Tools used:

1. Regular Expressions
2. Remove punctuations, tags & stopwords
3. Tokenisation
4. Stemming and lemmatizing
5. N-grams

DATA CLEANING - FUNCTION

```
def precleaning(tweet):

    stop_words = set(stopwords.words('spanish'))
    wordnet_lemmatizer = WordNetLemmatizer()
    pat1 = r'@[A-Za-z0-9_]+'
    pat2 = r'https?://[^ ]+'
    combined_pat = r'|'.join((pat1, pat2))
    www_pat = r'www.[^ ]+'

    soup = BeautifulSoup(tweet, 'lxml')
    souped = soup.get_text()
    try:
        bom_removed = souped.decode("utf-8-sig").replace(u"\ufffd", "?")
    except:
        bom_removed = souped

    stripped = re.sub(combined_pat, '', bom_removed)
    stripped = re.sub(www_pat, '', stripped)
    only_letters = re.sub("[^a-zA-Z]", " ",stripped)
    tokens = nltk.word_tokenize(only_letters)[2:]
    lower_case = [l.lower() for l in tokens]

    filtered_result = list(filter(lambda l: l not in stop_words, lower_case))
    lemmas = [wordnet_lemmatizer.lemmatize(t) for t in filtered_result]

    return lemmas

def ngrams(input_list):

    bigrams = [' '.join(t) for t in list(zip(input_list, input_list[1:]))]
    trigrams = [' '.join(t) for t in list(zip(input_list, input_list[1:], input_list[2:]))]

    return bigrams+trigrams

def count_words(input):
    cnt = collections.Counter()
    for row in input:
        for word in row:
            cnt[word] += 1

    return cnt
```

DATA CLEANING - FUNCTION

```
def cleaning(tweet):

    tok = WordPunctTokenizer()
    pat1 = r'@[A-Za-z0-9]+'
    pat2 = r'https?://[A-Za-z0-9./]+'
    pat3=r'(\w+:\/\/\S+)'
    combined_pat = r'|'.join((pat1, pat2,pat3))

    soup = BeautifulSoup(tweet, 'lxml')
    souped = soup.get_text()
    stripped = re.sub(combined_pat, '', souped)
    try:
        clean = stripped.decode("utf-8-sig").replace(u"\ufffd", "?")
    except:
        clean = stripped
    letters_only = re.sub("[^a-zA-Z]", " ", clean)
    lower_case = letters_only.lower()
    # During the letters_only process two lines above, it has created unnecessay white spaces,
    # I will tokenize and join together to remove unneccessary white spaces
    words = tok.tokenize(lower_case)
    return (" ".join(words)).strip()

def tokenize(text):
    tokens = nltk.word_tokenize(text,language='spanish')
    stems = []
    for item in tokens:
        stems.append(nltk.PorterStemmer().stem(item))
    return stems
```

MODEL TESTING

The steps for testing a model are always the same, and the only change is the model to be tested:

1. Prepare the data

   (obtain_data_representation)

2. Create the model

   (eg.) model = BernoulliNB()

3. Train with some data, where `x` are features and `y` is the target category

   model.fit(x, y)

4. Predict new categories for test data

   y_pred = model.predict(test_x)

MODEL TESTING AND SELECTION

Model	Score		
	raw data	imbalanced data	balanced data
Bernouilli Naive Bayes	0.575	0.596	0.811
Multinomial Naive Bayes	0.558	0.580	0.780
Ridge Classifier	0.604	0.598	0.814
Perceptron	0.571	0.535	0.806
Passive Agressive Classifier	0.578	0.547	0.799
Random Forest Classifier	0.567	0.563	0.784
Multinomial Naive Bayes (2)	0.580	0.589	0.796
Perceptron (2)	0.578	0.522	0.791
Ridge Classifier (2)	0.605	0.598	0.814
Passive Agressive Classifier (2)	0.578	0.508	0.789
Random Forest Classifier (2)	0.580	0.573	0.789
Stochastic Gradient Descent Classifier	0.601	0.593	0.815
Support Vector Machines Classifier	0.489	0.322	0.343
K-Neighbours Classifier	0.563	0.305	0.448
Nearest Centroid Classifier	0.564	0.574	0.779
Decision Tree Classifier	0.476	0.472	0.750
Ada Boost Classifier	0.565	0.553	0.763
Extra Trees Classifier	0.585	0.590	0.801
Extra Trees Classifier(2)	0.592	0.586	0.808

These models were selected for following testing and results were submitted to Kaggle system.

Highest score0.63 (Kaggle) was achieved with ETC model.

CONCLUSIONS

1. Cleaning the data is an essential step for ensuring data quality for decision-making, though ‘too much cleaning’ can lead to low variability on data and drop the model performance

2. Balancing the data set can improve the model performance, though it can increase overfitting

3. Quality and quantity of initial data is crucial to get good results!

MORE INFOé

You can find more information on this project in our blog:

https://bigsolutionsolutions.wordpress.com

Or on GitHub:

https://github.com/lparfer/bigSolutionSolutions

THANK YOU!