Analyzing Flight Review Using NLP

1.1 Introduction

Python has surfaced as a dominant language in Artificial Intelligence and Machine Learning programming because of its simplicity and flexibility, as well as its great support for open source libraries such as keras, pandas, numpy, nltk and TensorFlow. Artificial Intelligence (AI) and Machine Learning (ML) are the new black of the IT industry. While discussions over safety of its development keep escalating, developers expand abilities and capacity of artificial intellect. It became the necessity being widely used for processing and analyzing huge volumes of data. AI helps to handle the work that cannot be done manually anymore because of its significantly increased volumes and intensity. Artificial Neural Networks are implemented to utilize most of the algorithms and functions that are defined in python for creating efficient AI applications.

Neural networks, or neural nets, were inspired by the architecture of neurons in the human brain. A simple "neuron" N accepts input from multiple other neurons, each of which, when activated (or "fired"), cast a weighted "vote" for or against whether neuron N should itself activate. Modern neural nets can learn both continuous functions and, surprisingly, digital logical operations. Neural networks' early successes included predicting the stock market and (in 1995) a mostly self-driving car. In the 2010s, advances in neural networks using deep learning thrust AI into widespread public consciousness and contributed to an enormous upshift in corporate AI spending.

1.2 Objective

i. The main objective of our project is to analyze the reviews given by users and categorize them using AI algorithms into either positive or negative reviews.
ii. We also need to provide a UI wherein we can enter our own review and the AI application will predict its outcome and display the required result as either a positive or a negative review.

1.3 Problem statement

There is an exponential amount of text data pertaining towards user airline reviews and manual analysis of such abundant data is extremely exhausting hence we utilize the use of AI algorithms such as Natural Language Processing(NLP) and Artificial Neural Networks(ANN) to analyze the reviews and give a suitable prediction of whether its a positive or negative review.

2.0 Review of Literature

In the 21st century artificial intelligence (AI) has become an important area of research in virtually all fields: engineering, science, education, medicine, business, accounting, finance, marketing, economics, stock market and law, among others (Halal (2003), Masnikosa (1998), Metaxiotis et al. (2003), Raynor (2000), Stefanuk and Zhozhikashvili (2002) , Tay and Ho (1992) and Wongpinunwatana et al. (2000)).

The solution that we have come up with involves the use of two specific AI algorithms namely NLP and ANN:

i. Natural Language Processing (NLP) is a way of analyzing texts by computerized means. NLP involves gathering of knowledge on how human beings understand and use language. This is done in order to develop appropriate tools and techniques which could make computer systems understand and manipulate natural languages to perform various desired tasks. This paper reviews the literature on NLP. It also covers or gives a hint about the history of NLP. It is based on document analysis. This research paper could be beneficial to those who wish to study and learn about NLP. Review written by: Botswanna International University of Science and Technology.

ii. Artificial Neural Network (ANN) are a powerful paradigm in machine learning inspired by the computation and communication capabilities of the brain. As such they have been the basis for many powerful algorithms with applications in pattern recognition, memory, mapping, etc. Although the final application may differ, the two

components of ANNs remain the same: in analogy with biological systems, they are referred to as neurons and synapses, and correspond to the vertices and the edges of the graph respectively. The classification of ANN is possible in many ways like supervised learning and unsupervised learning. From a structural perspective, ANNs can be divided into two main categories, feed-forward networks, in which the computation is performed in a layer-by-layer fashion from the input to the output of the network; and recurrent networks which have an interconnected network structure including cycles.

There is another technique that is frequently used for the same application known as Sentimental Analysis.

3.0 Data collection

The data set that we have collected has been taken from the web through the following link: https://github.com/quankiquanki/skytrax-reviews-dataset

	airline_name	content	cabin_flown	overall_rating	recommended
0	adria-airways	Outbound flight FRA/PRN A319. 2 hours 10 min f...	Economy	7.0	1
1	adria-airways	Two short hops ZRH-LJU and LJU-VIE. Very fast ...	Business Class	10.0	1
2	adria-airways	Flew Zurich-Ljubljana on JP365 newish CRJ900. ...	Economy	9.0	1
3	adria-airways	Adria serves this 100 min flight from Ljubljan...	Business Class	8.0	1
4	adria-airways	WAW-SKJ Economy. No free snacks or drinks on t...	Economy	4.0	0
5	adria-airways	Sarajevo-Frankfurt via Ljubljana. I loved flyi...	Economy	9.0	1
6	adria-airways	I had flights from Paris to Sarajevo via Ljubl...	Economy	5.0	1
7	adria-airways	LJU to FRA and back both flights were on time....	Economy	9.0	1
8	adria-airways	On my Ljubljana - Munich flight in business cl...	Business Class	8.0	1
9	adria-airways	Flights from LJU to ZRH and back all on time. ...	Economy	10.0	1
10	adria-airways	I was very satisfied with the CRJ 900 on my fl...	Economy	9.0	1
11	adria-airways	I was on JP650 the evening departure to Istanb...	Economy	7.0	1
12	adria-airways	VIE-LJU LJU-MUC Aug 4-15th CRJ200. Both flight...	Economy	8.0	1
13	adria-airways	If I have to fly a regional jet then I prefer ...	Business Class	7.0	1
14	adria-airways	Istanbul-Ljubljana-Munich and return. Pre-take...	Business Class	7.0	1
15	adria-airways	Return flight Paris-Skopje via Ljubljana. All ...	Economy	5.0	1
16	adria-airways	BEG-LJU-BEG with CRJ200 great regional airline...	Economy	9.0	1
17	adria-airways	I fly at least once a month ZRH-LJU-ZRH. My ex...	Economy	5.0	1
18	adria-airways	LTN to LJU outbound with CRJ200 return with CR...	Economy	5.0	1
19	adria-airways	Ljubljana to Brussels with CRJ900. Boarding on...	Economy	8.0	1
20	adria-airways	LJU-FRA and FRA-LJU. First flight on time retu...	Economy	9.0	1
21	adria-airways	LJU-FRA-LJU. Both flights on time. Flight atte...	Economy	5.0	0
22	adria-airways	Smart and efficient service. Small planes but ...	Economy	9.0	1

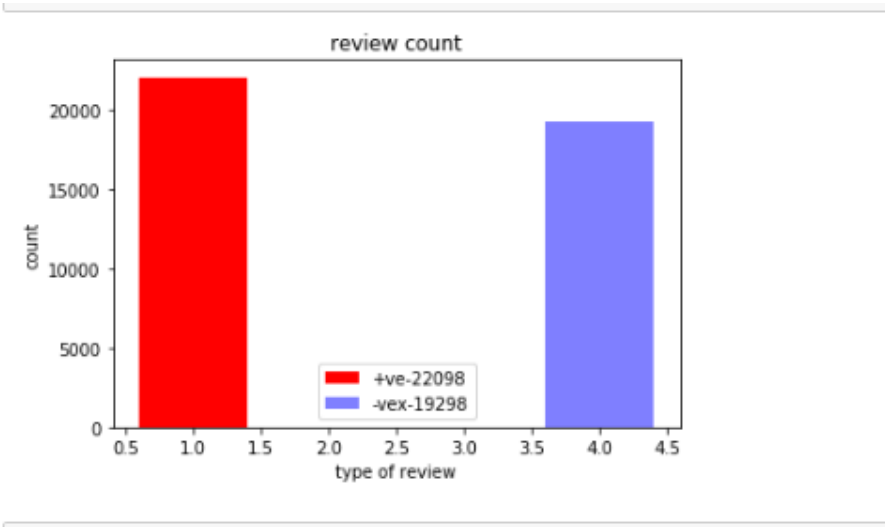
It consists of 34853 rows and 5 columns comprising of a number of airline reviews.

4.0 Methodolgy

4.1 Exploratory Data Analysis

4.1.1 Figures And Tables

The following graph below depicts the number of positive as well as negative reviews that have been used to train the AI application:



4.1.2 Data Modelling

The algorithms that have been used on our design are:

I. Data preprocessing:

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues.

Steps in Data Preprocessing:

Step 1 : Import the libraries
Step 2 : Import the data-set
Step 3 : Check out the missing values
Step 4 : See the Categorical Values
Step 5 : Splitting the data-set into Training and Test Set

In [3]:
```
# Import the Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Why we need splitting ?

Well here it's your algorithm model that is going to learn from your data to make predictions. Generally we split the data-set into 70:30 ratio or 80:20 what does it mean, 70 percent data take in train and 30 percent data take in test. However, this Splitting can be varies according to the data-set shape and size.

Here in our project we have taken the ratio of 95:05,We are training our algorithm with 95% of the data and the rest 5% for testing the algorithm.

II. Using NLP:

Natural Language Processing, or NLP, is the sub-field of AI that is focused on enabling computers to understand and process human languages. The steps involved are:

Step 1: Sentence Segmentation

The first step in the pipeline is to break the text apart into separate sentences. That gives us this:

1. "London is the capital and most populous city of England and the United Kingdom."

2. “Standing on the River Thames in the south east of the island of Great Britain, London has been a major settlement for two millennia.”
3. “It was founded by the Romans, who named it Londinium.”

We can assume that each sentence in English is a separate thought or idea. It will be a lot easier to write a program to understand a single sentence than to understand a whole paragraph.

Coding a Sentence Segmentation model can be as simple as splitting apart sentences whenever you see a punctuation mark. But modern NLP pipelines often use more complex techniques that work even when a document isn’t formatted cleanly.

Step 2: Word Tokenization

Now that we’ve split our document into sentences, we can process them one at a time. Let’s start with the first sentence from our document:

> *“London is the capital and most populous city of England and the United Kingdom.”*

The next step in our pipeline is to break this sentence into separate words or *tokens*. This is called *tokenization*. This is the result:

> *“London”, “is”, “ the”, “capital”, “and”, “most”, “populous”, “city”, “of”, “England”, “and”, “the”, “United”, “Kingdom”, “.”*

Tokenization is easy to do in English. We’ll just split apart words whenever there’s a space between them. And we’ll also treat punctuation marks as separate tokens since punctuation also has meaning.

Step 3: Predicting Parts of Speech for Each Token

Next, we’ll look at each token and try to guess its part of speech — whether it is a noun, a verb, an adjective and so on. Knowing the role of each word in the sentence will help us start to figure out what the sentence is talking about.

We can do this by feeding each word (and some extra words around it for context) into a pre-trained part-of-speech classification model:

The part-of-speech model was originally trained by feeding it millions of English sentences with each word's part of speech already tagged and having it learn to replicate that behavior.

Keep in mind that the model is completely based on statistics — it doesn't actually understand what the words mean in the same way that humans do. It just knows how to guess a part of speech based on similar sentences and words it has seen before.

Step 4: Text Lemmatization

In English (and most languages), words appear in different forms. Look at these two sentences:

I had a **pony**. I had two **ponies**.

Both sentences talk about the noun **pony,** but they are using different inflections. When working with text in a computer, it is helpful to know the base form of each word so that you know that both sentences are talking about the same concept. Otherwise the strings "pony" and "ponies" look like two totally different words to a computer.

In NLP, we call finding this process *lemmatization* — figuring out the most basic form or *lemma* of each word in the sentence.

The same thing applies to verbs. We can also lemmatize verbs by finding their root, unconjugated form. So "**I had two ponies**" becomes "**I [have] two [pony].**"

Lemmatization is typically done by having a look-up table of the lemma forms of words based on their part of speech and possibly having some custom rules to handle words that you've never seen before.

Step 5: Identifying Stop Words

Next, we want to consider the importance of a each word in the sentence. English has a lot of filler words that appear very frequently like "and",

"the", and "a". When doing statistics on text, these words introduce a lot of noise since they appear way more frequently than other words. Some NLP pipelines will flag them as **stop words** —that is, words that you might want to filter out before doing any statistical analysis. Stop words are usually identified by just by checking a hardcoded list of known stop words. But there's no standard list of stop words that is appropriate for all applications. The list of words to ignore can vary depending on your application.

For example if you are building a rock band search engine, you want to make sure you don't ignore the word "The". Because not only does the word "The" appear in a lot of band names, there's a famous 1980's rock band called *The The*!

Step 6.a: Dependency Parsing

The next step is to figure out how all the words in our sentence relate to each other. This is called *dependency parsing*. The goal is to build a tree that assigns a single **parent** word to each word in the sentence. The root of the tree will be the main verb in the sentence. But we can go one step further. In addition to identifying the parent word of each word, we can also predict the type of relationship that exists between those two words.

Just like how we predicted parts of speech earlier using a machine learning model, dependency parsing also works by feeding words into a machine learning model and outputting a result. But parsing word dependencies is particularly complex task and would require an entire article to explain in any detail.

But despite a note from the author in 2015 saying that this approach is now standard, it's actually out of date and not even used by the author anymore. In 2016, Google released a new dependency parser called *Parsey McParseface* which outperformed previous benchmarks using a new deep learning approach which quickly spread throughout the industry. Then a year later, they released an even newer model called *ParseySaurus* which improved things further. In other words, parsing techniques are still an active area of research and constantly changing and improving.

It's also important to remember that many English sentences are ambiguous and just really hard to parse. In those cases, the model will

make a guess based on what parsed version of the sentence seems most likely but it's not perfect and sometimes the model will be embarrassingly wrong. But over time our NLP models will continue to get better at parsing text in a sensible way.

Step 6.b: Finding Noun Phrases

So far, we've treated every word in our sentence as a separate entity. But sometimes it makes more sense to group together the words that represent a single idea or thing. We can use the information from the dependency parse tree to automatically group together words that are all talking about the same thing.

In [30]:
```
import string
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.tokenize import WhitespaceTokenizer
from nltk.stem import WordNetLemmatizer
```

In [31]:
```
def clean_text(text):
    # lower text
    text = text.lower()
    # tokenize text and remove puncutation
    text = [word.strip(string.punctuation) for word in text.split(" ")]
    # remove words that contain numbers
    text = [word for word in text if not any(c.isdigit() for c in word)]
    # remove stop words
    stop = stopwords.words('english')
    text = [x for x in text if x not in stop]
    # remove empty tokens
    text = [t for t in text if len(t) > 0]
    # pos tag text
    pos_tags = pos_tag(text)
    # lemmatize text
    text = [WordNetLemmatizer().lemmatize(t[0], get_wordnet_pos(t[1])) for t in pos_tags]
    # remove words with only one letter
    text = [t for t in text if len(t) > 1]
    # join all
    text = " ".join(text)
    return(text)
```

In [32]:
```
# clean text data
ds["review_clean"] = ds["content"].apply(lambda x: clean_text(x))
```

In [39]:
```
len(ds)*.7
```

Out[39]:
```
24397.1
```

In [51]:
```
train_size = int(len(ds) * .7)
train_content = ds['review_clean'][:train_size]
train_recommended = ds['recommended'][:train_size]

test_content = ds['review_clean'][train_size:]
test_recommended = ds['recommended'][train_size:]
```

In [52]:
```
from keras.preprocessing.text import Tokenizer
```

In [53]:
```
max_words = 1000
tokenize = Tokenizer(num_words=max_words)
tokenize.fit_on_texts(train_content) # only fit on train

x_train = tokenize.texts_to_matrix(train_content)
x_test = tokenize.texts_to_matrix(test_content)
```

In [54]:
```
from keras.utils import to_categorical
```

In [55]:
```
x_train_new=to_categorical(x_train)
```

In [56]:
```
x_test_new=to_categorical(x_test)
```

III. Using Artificial Neural Networks:

Artificial Neural Network (ANN) are a powerful paradigm in machine learning inspired by the computation and communication capabilities of the brain. As such they have been the basis for many powerful algorithms with applications in pattern recognition, memory, mapping, etc. [1]. Although the final application may differ, the two components of ANNs remain the same: in analogy with biological systems, they are referred to as neurons and synapses, and correspond to the vertices and the edges of the graph respectively. The classification of ANN is possible in many ways like supervised learning and unsupervised learning. From a structural perspective, ANNs can be divided into two main categories, feed-forward networks, in which the computation is performed in a layer-by-layer fashion from the input to the output of the network; and recurrent networks which have an interconnected network structure including cycles.

```
In [57]: from keras.models import Sequential
         from keras.layers import Dense

In [58]: model = Sequential()
         model.add(Dense(512, input_shape=(max_words,),activation='relu'))
         model.add(Dense(2,activation='softmax'))

In [70]: model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])

In [72]: train_type=train_type.reshape(-1,1)

In [73]: train_type.shape
Out[73]: (24397, 1)

In [74]: x_train.ndim
Out[74]: 2

In [75]: train_type_new=to_categorical(y=train_type)

In [76]: train_type_new.ndim
Out[76]: 2
```

```
In [95]: history = model.fit(x_train,train_type_new,batch_size=32,epochs=5,verbose=1,validation_split=0.1)

Train on 21957 samples, validate on 2440 samples
Epoch 1/5
21957/21957 [==============================] - 11s 488us/step - loss: 0.0090 - acc: 0.9989 - val_loss: 0.4453 - val_acc: 0.8881
Epoch 2/5
21957/21957 [==============================] - 11s 486us/step - loss: 0.0030 - acc: 0.9997 - val_loss: 0.4920 - val_acc: 0.8914
Epoch 3/5
21957/21957 [==============================] - 10s 478us/step - loss: 0.0017 - acc: 0.9998 - val_loss: 0.5198 - val_acc: 0.8902
ss: 0.0023 - acc - ETA
Epoch 4/5
21957/21957 [==============================] - 11s 491us/step - loss: 9.7777e-04 - acc: 0.9999 - val_loss: 0.5441 - val_acc: 0.
8885
Epoch 5/5
21957/21957 [==============================] - 11s 489us/step - loss: 0.0018 - acc: 0.9997 - val_loss: 0.5639 - val_acc: 0.8910

In [96]: model.save("airlineR.h5")

In [97]: test_type_new=to_categorical(test_type)

In [98]: test_type_new.ndim
Out[98]: 2

In [99]: score = model.evaluate(x_test,test_type_new)
         print('Test accuracy:', score[1])

10456/10456 [==============================] - 1s 75us/step
Test accuracy: 0.8683052792198893
```

test

```
In [136]: temperature they set. Not way to self select.The media content they offer is more of an afterthought. Begs a lot to be desired.A

In [137]: len(new[0])
Out[137]: 1000

In [138]: n=model.predict(new)[0]

In [139]: n=list(n)
```

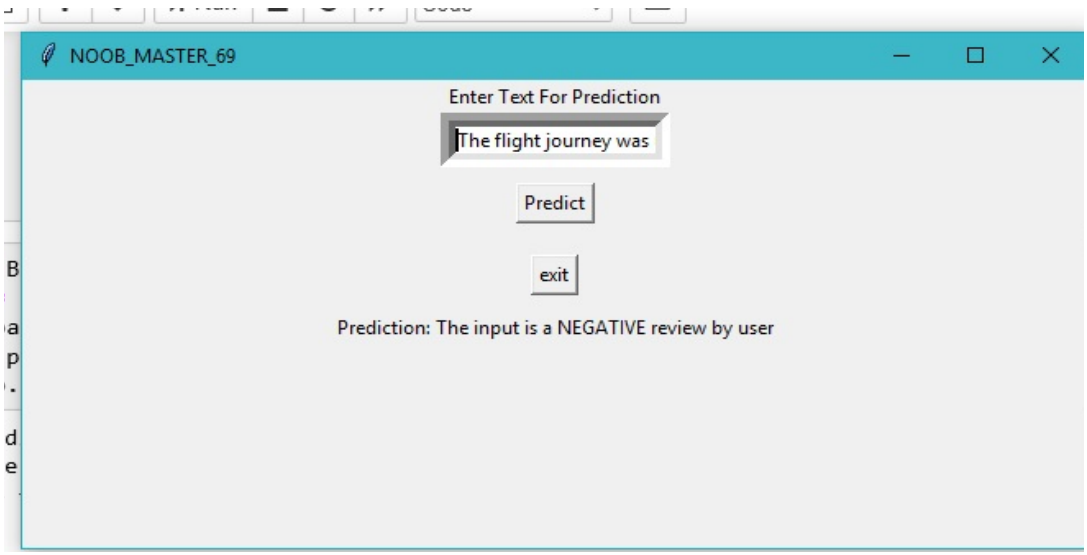
IV. UI implementation using tkinter library:

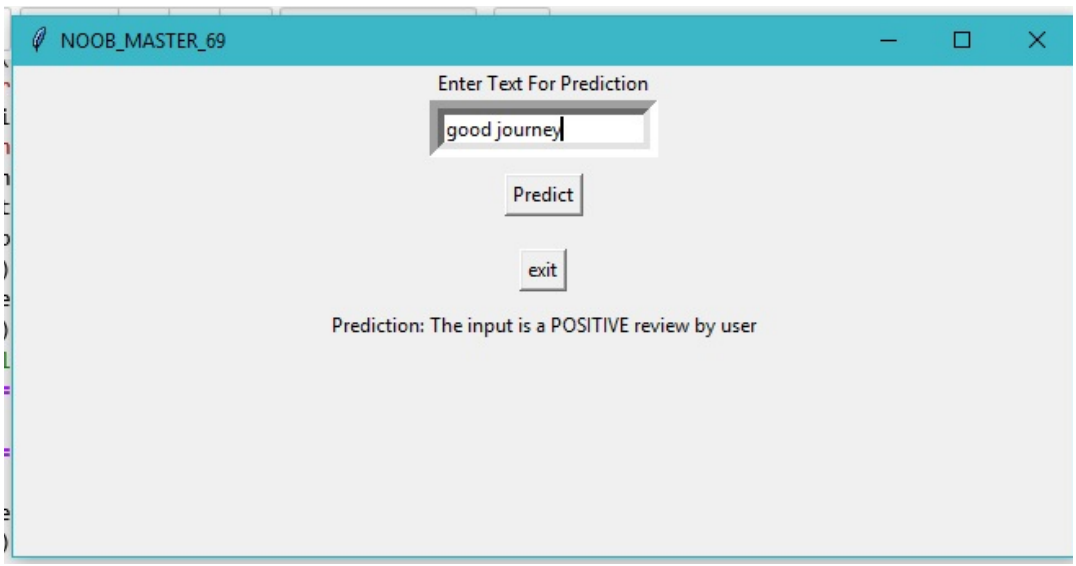
We have designed our User Interface using the library tkinter:

```
In [1]: from tkinter import *
        from keras.models import load_model
        import string

Using TensorFlow backend.

In [2]: import pickle
        #from sklearn.feature_extraction.text import CountVectorizer
        model = load_model('airlineR_new.h5')
        #ith open('CountVectorizer','rb') as file:
         #   cv=pickle.load(file)

WARNING:tensorflow:From C:\Users\DELL\Anaconda3\lib\site-packages\tensorflow\python\framework\op_def_library.py:263: colocate_w
ith (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From C:\Users\DELL\Anaconda3\lib\site-packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensor
flow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.

In [3]: top = Tk()
        top.geometry("550x300+300+150")
        top.resizable(width=True, height=True)
Out[3]: ''
```

In [4]:
```
from nltk.corpus import wordnet

def get_wordnet_pos(pos_tag):
    if pos_tag.startswith('J'):
        return wordnet.ADJ
    elif pos_tag.startswith('V'):
        return wordnet.VERB
    elif pos_tag.startswith('N'):
        return wordnet.NOUN
    elif pos_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN
```

In [5]:
```
import string
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.tokenize import WhitespaceTokenizer
from nltk.stem import WordNetLemmatizer

def clean_text(text):
    # lower text
    text = text.lower()
    # tokenize text and remove puncutation
    text = [word.strip(string.punctuation) for word in text.split(" ")]
    # remove words that contain numbers
    text = [word for word in text if not any(c.isdigit() for c in word)]
    # remove stop words
    stop = stopwords.words('english')
    text = [x for x in text if x not in stop]
    # remove empty tokens
```





6. Findings and Suggestions

The various sources that we have used for the following project are:
i. Skytrax datasets
ii. Several reports on NLP and ANN derived from the web.

One of the key findings we have found was that analysis of flight reviews can be implemented using NLP and ANN as displayed by the model application that has been designed. However, for a more accurate predictive AI a larger dataset should be acquired along with the implementation of Sentemental Analysis.

Conclusion

It is completely possible to use only raw text as input for making predictions. The most important thing is to be able to extract the relevant features from this raw source of data. This kind of data can often come as a good complementary source in data science projects in order to extract more learning features and increase the predictive power of the models.

This report was done by the team “Noob_master_69”