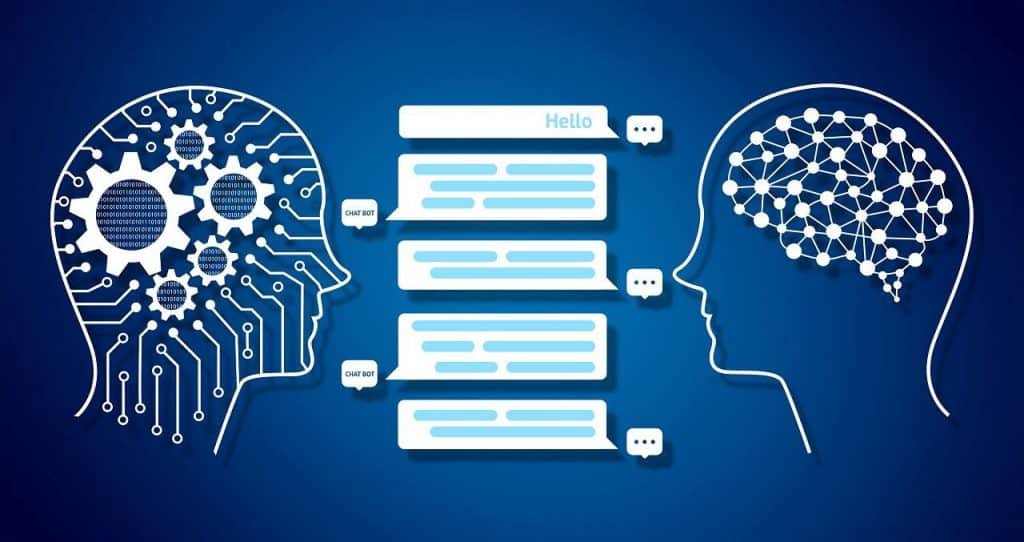
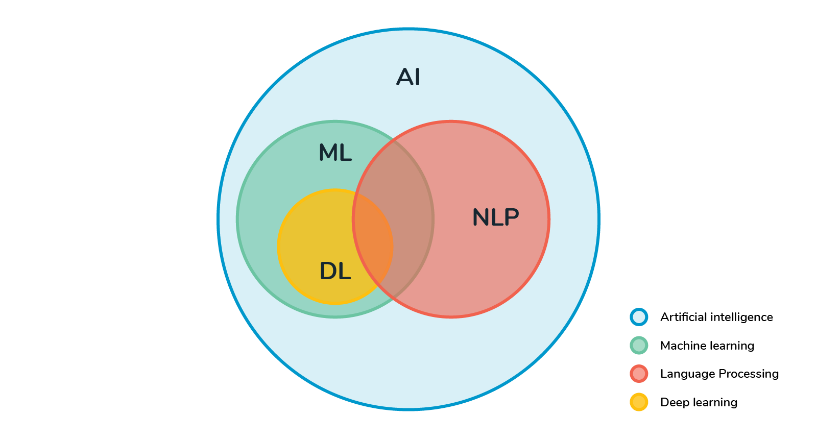
**Hybrid Machine Learning Systems for NLP**

**What is NLP?**

NLP stands for Natural Language Processing and it is a type of Artificial Intelligence that enables machines to automatically manipulate natural language like speech, text, etc. Natural language like speech is one of the most important types of interactions that humans perform. To be able to understand and derive meaning from it automatically is essential when it comes to assimilating huge chunks of data.

NLP is on the rise now, since it is the next step towards understanding human wants and needs. There is also an increase in computational power required to aid this process. In addition, there is a staggering amount of data that is available ready to be structured and analyzed. Many industries like healthcare, finance, entertainment media, use NLP to achieve meaningful analysis.

**Machine Learning in NLP**

First, we must know the differences between NLP, Machine Learning, Artificial Intelligence and Deep Learning to understand how dependent one technique is on the other ones.

Artificial Intelligence is related to making machines perform tasks that require human-like intelligence. Machine Learning is one such application of AI and Deep Learning is a subset of ML which is applied for advanced learning. NLP is also a part of AI, but it can overlap with ML and DL.

NLP uses machine learning algorithms to understand and find the meaning of the text documents that range from small tweets to legal documents. ML and AI aids in improving and automating the text analytics functions of NLP that convert unstructured data into structured information. There are many techniques to make the given unstructured data into useable bits. These techniques help identify sentiment, parts of speech, etc.

Here are some of the popular NLP Machine Learning algorithms:

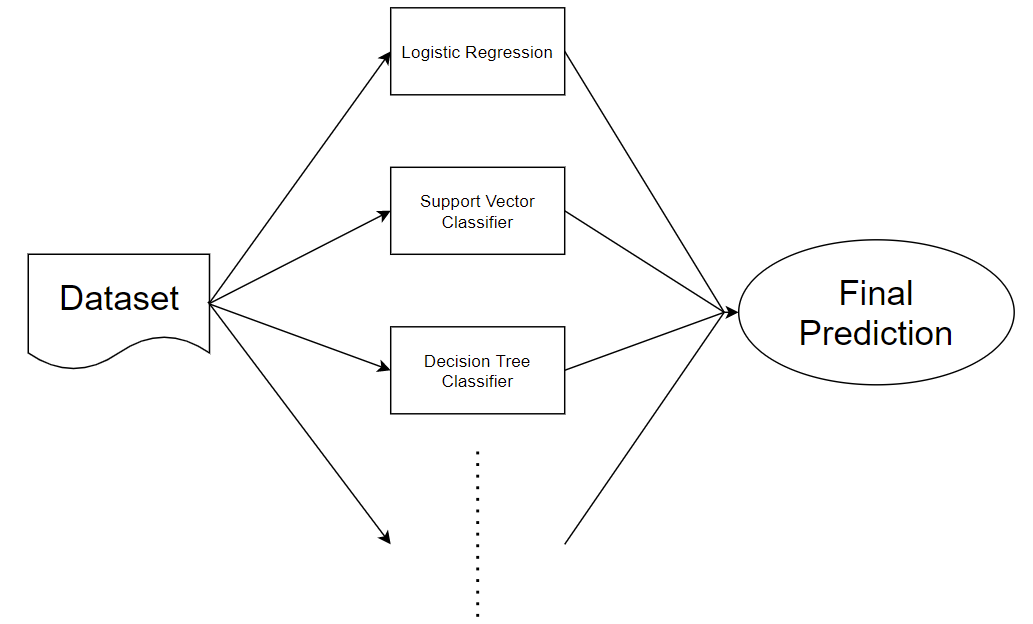
* Random Forest Classification
* Support Vector Machine
* Logistic Regression
* KNN Classification
* Bayesian Networks

**Hybrid Machine Learning Systems:**

Hybrid Machine Learning means exactly that, it is a combination of two or more machine learning algorithms, be it supervised or unsupervised. We might have even used Hybrid Machine Learning systems without our knowledge sometimes. It combines different algorithms or processes from a wide array of selections. This hybridization increases the accuracy of the model since no single model will solve a problem completely.

There are endless ways to combine different models of ML, like classification + classification, classification + clustering, clustering + clustering, and even more.

Using hybrid machine learning systems for NLP is definitely an advantage since it improves the ability of the machine to understand the data better. Different variations of machine learning algorithms can be used in tandem to solve a given problem.



Let us take a sentiment analysis problem and look at how hybrid machine learning systems affect the accuracy of the model.

The dataset contains tweets that talk about the problems of six different major US airlines. Our aim is to train a model to categorize the tweets into positive, negative, and neutral in sentiment. We will use a hybrid machine learning system that uses Logistic Regression and Support Vector Machines Classification to achieve the expected result.

Let us start by importing the necessary libraries

# Importing necessary packages

%matplotlib inline

import numpy as np

import pandas as pd

import re

import nltk

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.feature\_extraction.text import TfidfVectorizer

from nltk.corpus import stopwords

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score

from sklearn.ensemble import VotingClassifier

To load the data, let us download the dataset from Kaggle:

<https://www.kaggle.com/crowdflower/twitter-airline-sentiment>

Upload the downloaded csv file to your Google Drive. Now mount the drive in the colab using the below code snippet.

from google.colab import drive

drive.mount('/content/drive')



Change your directory to the location of the dataset:

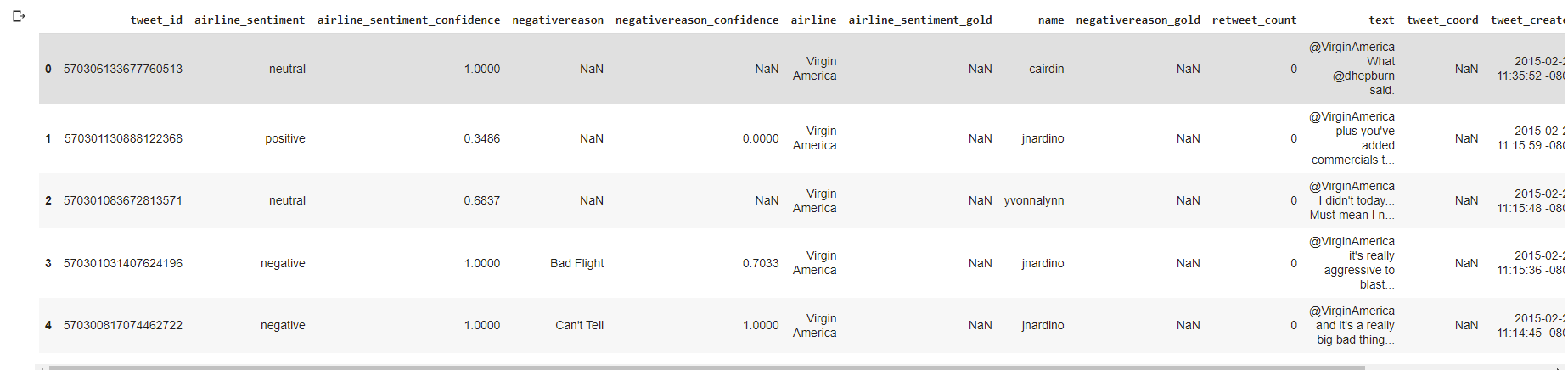
cd /content/drive/MyDrive/tweets



Let us read the comma separated values and clean the column that contains the tweets.

tweets = pd.read\_csv("Tweets.csv", sep = ',')

tweets.head()



Cleaning of data is done by removing all the special characters, single characters, and the stopwords like “during’, ‘out’, ‘very’, ‘having’, etc. These characters carry far less meaning than the other keywords, hence this step is essential. Also, the text is converted to lowercase for uniformity.

# Cleaning the data

text\_set = tweets.iloc[:, 10].values #Selecting the 11th column: text

sentiment\_set = tweets.iloc[:, 1].values #Selecting the 2nd columnn: airline\_sentiment

cleaned\_text\_set = list()

for input\_phrase in range(0, len(text\_set)):

    clean\_text = re.sub(r'\W', ' ', str(text\_set[input\_phrase])) #Removing spl characters and single characters

    clean\_text= re.sub(r'\s+[a-zA-Z]\s+', ' ', clean\_text)

    clean\_text = re.sub(r'\^[a-zA-Z]\s+', ' ', clean\_text)

    clean\_text = clean\_text.lower()

    cleaned\_text\_set.append(clean\_text)

To remove stopwords, we have to download the list of stopwords from nltk library. To extract features from the cleaned version, we will be using TF-IDF features.

import nltk

nltk.download("stopwords")

input\_vector = TfidfVectorizer (max\_features=3000, min\_df=6, max\_df=0.8, stop\_words=stopwords.words('english'))

cleaned\_text\_set = input\_vector.fit\_transform(cleaned\_text\_set).toarray()

We have used the 3000 most frequent words in the data as features. Now let us split the data into training and validation sets. A 80-20 split is optimum.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(cleaned\_text\_set, sentiment\_set, test\_size=0.2, random\_state=42)

Since we are using a hybrid machine learning system with two algorithms, let us test the data with each on the algorithms.

# Logistic Regression

lr = LogisticRegression(random\_state=0, solver='lbfgs', multi\_class='ovr').fit(X\_train, y\_train)

lr\_score = lr.score(X\_train, y\_train)

print("Logistic Regression Accuracy Score: ", lr\_score)

# Support Vector Machine Linear Classification

svc = SVC(kernel='linear')

svc.fit(X\_train, y\_train)

svc\_score = svc.score(X\_train, y\_train)

print("SVM Classification Accuracy Score: ", svc\_score)



It is time to combine the two learning models. We have defined each of the two machine learning models two times that that result in a combination of 4 weak learners. Now, the Max Voting Classifier method. The final class prediction of the ensemble model will be the class which has been predicted mostly by the weak learners.

# create the sub-models

estimators = []

#Defining 2 Logistic Regression Models

model11 = LogisticRegression(random\_state=0, solver='lbfgs', multi\_class='ovr').fit(X\_train, y\_train)

estimators.append(('logistic1', model11))

model12 = LogisticRegression(random\_state=0, solver='lbfgs', multi\_class='ovr').fit(X\_train, y\_train)

estimators.append(('logistic2', model12))

#Defining 2 Support Vector Classifiers

model21 = SVC(kernel = 'linear')

estimators.append(('svm1', model21))

model22 = SVC(kernel = 'linear')

estimators.append(('svm2', model22))

# Defining the ensemble model

ensemble = VotingClassifier(estimators)

ensemble.fit(X\_train, y\_train)

#y\_pred = ensemble.predict(X\_test)

ensemble\_score = ensemble.score(X\_train, y\_train)

print("Ensemble Score: ", ensemble\_score)



Running each of the models on the test data,

final\_pred\_lr = lr.predict(X\_test)

final\_pred\_svc = svc.predict(X\_test)

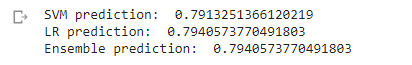
final\_pred\_ensemble = ensemble.predict(X\_test)

# Accuracy score of the final prediction

print("SVM prediction: ", accuracy\_score(y\_test, final\_pred\_svc))

print("LR prediction: ", accuracy\_score(y\_test, final\_pred\_lr))

print("Ensemble prediction: ", accuracy\_score(y\_test, final\_pred\_ensemble))



We can see that the hybrid machine learning model has done better or equal to the other individual learning models. Therefore, a hybrid approach to machine learning for NLP is one of the best ways there is.

**References**

1. <https://www.lexalytics.com/lexablog/machine-learning-natural-language-processing>
2. <https://analyticsindiamag.com/a-hands-on-guide-to-hybrid-ensemble-learning-models-with-python-code/>
3. <https://www.kaggle.com/crowdflower/twitter-airline-sentiment>
4. <https://towardsdatascience.com/a-hybrid-approach-to-natural-language-processing-6435104d7711>
5. <https://jpt.spe.org/hybrid-machine-learning-explained-nontechnical-terms>