

Introduction to Machine Learning and Its Basic Application in Python

Pinky Sodhi^a Naman Awasthi^b Vishal Sharma^c

^aAssistant Professor, Computer Applications, Prestige Institute of Management, Gwalior

^bStudents, BCA, Prestige Institute of Management, Gwalior

^cStudents, BCA, Prestige Institute of Management, Gwalior

ARTICLE INFO

Keywords:

Machine Learning,
Python, Scikit-Learn,
AI, ML, Deep
Learning, NumPy,
Matplotlib, Workflow
of machine learning,
NLTK

ABSTRACT

Artificial Intelligence, Machine Learning and Deep Learning are the buzzwords that have been able to grasp the interest of many researchers since various numbers of years. Enabling computers to think, decide and act like humans has been one of the most significant and noteworthy developments in the field of computer science. Various algorithms have been designed over time to make machines impersonate the human brain and many programming languages have been used to implement those algorithms. Python is one such programming language that provides a rich library of modules and packages for use in scientific computing and machine learning. This paper aims at exploring the basic concepts related to machine learning and attempts to implement a few of its applications using python. This paper majorly used Scikit-Learn library of Python for implementing the applications developed for the purpose of research.

Introduction

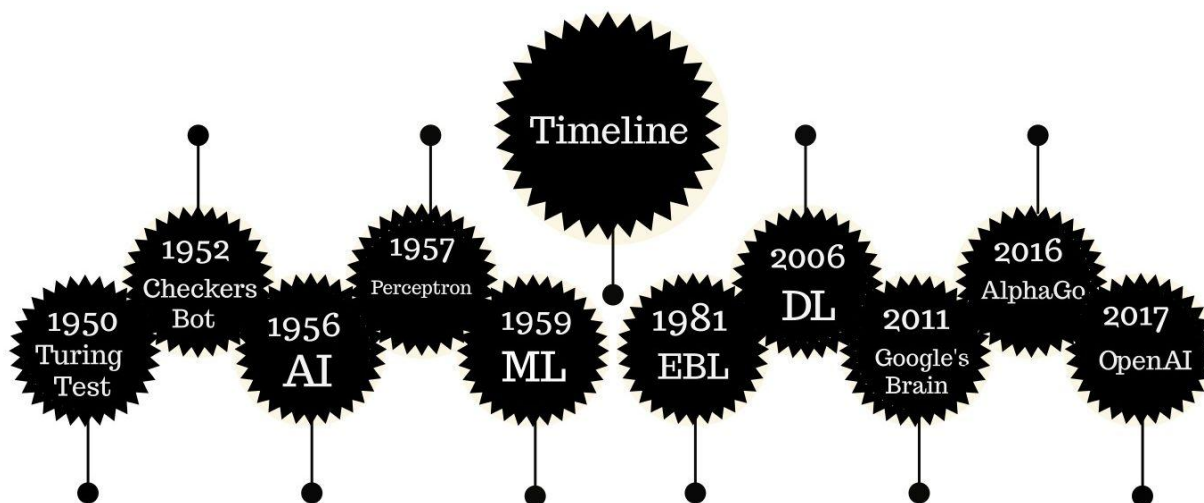
“Computers are able to see, hear and learn. Welcome to the future”- Dave Waters.

Artificial Intelligence and Machine Learning and Deep Learning are the concepts that have been around for quite a few decades now and have been implemented or thought to be implemented many times, to make the machines do possibly everything that the humans can do without being explicitly instructed. Haffner (2016) very appropriately wrote that Machine Learning is composed of algorithms that educate computers to carry out tasks that are performed naturally and effortlessly by humans on a daily basis. For example, reading, deciding and marking an email as spam; or simply looking at the weather and deciding if an umbrella would be required when going out; or merely recognizing the features of a given fruit and identifying whether it is an apple or an orange. Extending this thought, in her article, Priyadharshini (2017) mentioned that

Available on SSRN-Elsevier

machine learning enables computers to find intuitive information by using algorithms that repeatedly learn from data instead of being explicitly programmed about where exactly to look for a piece of information.

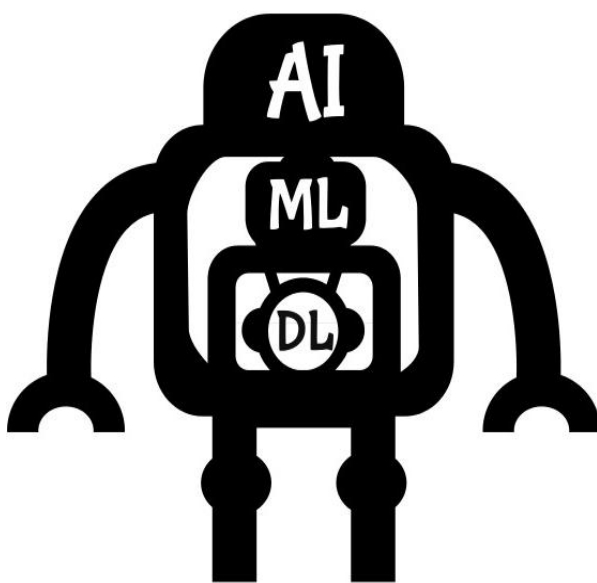
How did it all begin?



Machine Learning is not a very recent concept according to Haffner (2016) and its history dates to over sixty years back. Marr (2016) and Pacheco (2016) also wrote that it all started in 1950 when Alan Turing came up with the idea of “Turing Test” to analyze if computer has true intellect or not. Subsequently in 1952, Arthur Samuel developed the first computer program that could play checkers and was capable of learning new strategies with each new game it played. Frank Rosenblatt, in 1957, designed perceptron (the first computer neural network) to mimic the thinking process of a human brain. Next, in the year 1967, an algorithm named “nearest neighbor” was designed to make the computer capable of identifying very basic patterns like mapping a route in a city. Further in 1979, a mobile robot named “Stanford Cart” that was able to navigate itself through obstacles in a room was developed by the students of Stanford University. As a major step forward, the concept of Explanation Based Learning (EBL) was introduced in 1981 by Gerald Dejong whereby a computer could analyze and discard irrelevant data out of the given training data. Eventually, by 1990s, the approach towards machine learning became more data-driven instead of the traditional knowledge-driven, where programs were developed to analyze and extract conclusions from massive amounts of data. Since then there have been many technological advancements in the field of machine learning that include introduction of deep neural networks (DNN) for image processing, development of “deep face algorithm” by Facebook based on DNN, and many other neural networks. In the last decade, the world of machine learning and artificial intelligence have witnessed tremendous achievements as compared to their earlier years, visible through the victory of IBM’s Watson at Jeopardy and development

of Google's Brain in 2011, launch of machine learning platform by Amazon and creation of distributed machine learning toolkit by Microsoft in 2015 and triumph of Google's AI (AlphaGo) in the world's most complex Chinese board game Go, against Lee Sedol (a professional Go player) winning five times in a row in 2016. According to some proficient Go players, AlphaGo was capable of making millions of ingenious moves than they had ever witnessed before. 2017 also witnessed the introduction to a bot by OpenAI (founded by Elon Musk) that defeated the world's best player in a competitive eSport, Dota 2.

AI, ML and Deep Learning, How Do They Relate?



Web articles by Priyadharshini and Tagliaferri (2017) define Machine learning (ML) as an associate or core sub-area of artificial intelligence (AI) that permits computers to follow a self-learning approach without being programmed explicitly. John McCarthy coined the term “Artificial Intelligence” in 1956 for a new research field in the computer science domain which focused on making machines mimic human cognitive functions. An article by Puget (2016) quoted that the term “Machine Learning” was coined by Arthur Samuel in 1959, who defined it as “a field of study that gives computers the ability to learn without being explicitly programmed”. According

to a research report by Samuel (1959) it is possible to program a digital computer to learn in exactly the same way as humans or animals could learn from real experiences which would eventually reduce the need for detailed programming effort. Progressing further, in 2006, Geoffrey Hinton came up with the phrase “Deep Learning” that, as explained by Brownlee (2016) is a subfield of machine learning that uses neural networks to imitate human like decision making, wherein Neural Networks or Artificial Neural Networks are the algorithms inspired by the function and structure of the human brain.

Why Machine Learning?

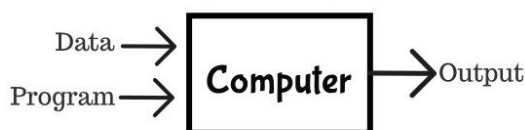
As pointed out by Brownlee (2015), Machine Learning facilitates the computers to program themselves. This very basic idea behind the concept of machine learning describes its importance in today's computing.

As compared to conventional programming that simply represents automation, machine learning is characterized by automating the process of automation. Data and program
Available on SSRN-Elsevier

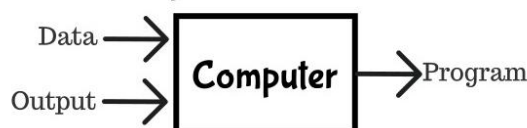
are provided as input to computer during traditional programming to produce a certain output whereas machine learning intakes data and output to create a program itself. As explained by Prof. Mostafa (2012) in one of his lectures on The Learning Problem, conventional programming would only be successful when some mathematical formula could be derived upon for solving a problem but

machine learning does the opposite by breaking this convention and working even if it isn't possible to pin down a mathematical principle, by repeatedly recognizing patterns and developing its own algorithms as and when the data changes. Prof. Mostafa further proved this by citing an example of predicting how a viewer would rate a movie, as there exists a pattern but it is not possible to pin down a mathematical model for the same because the way a person would rate a movie is not related to how they would rate other movies or how other people would rate the same movie.

Conventional Programming:



Machine Learning:



Malhotra (2016), a machine learning researcher at TCS Research stated that data availability and computation power are two of the interesting features that have lead to an extensive use of machine learning. Based on these two key factors, machine learning models are able to dig out patterns from gigantic quantities of data which is not possible for humans because a human brain cannot retain everything for long and also cannot carry out redundant computations continuously for hours and days to arrive at interesting patterns.

Types of Problems Solved Using Machine Learning

Chen (2017), editor of The Eliza Effect mentioned five types of problems that can be solve by applying machine learning:

- **Classification:** used to identify the category to which an object belongs. For example, is it spam? Or is it cancerous?
- **Regression:** used to predict a continuous numeric-valued aspect associated with an object. For example, the probability that a user would click on an ad or stock price prediction.

- **Similarity/ Anomaly:** used to retrieve similar objects or to find anomalies in behavior. For example, searching for similar images or detecting deception in user behavior.
- **Ranking:** used to sort relevant data according to a particular input. For example, Google Page Rank
- **Sequence Prediction:** used to predict the next element in a series of data. For example, predicting the next word in a sentence.
- However, similar objects can also be grouped into sets using Clustering as explained by Gupta, Negi, Vishwakarma, Rawat, & Badhani (2017)

Types of Machine Learning Algorithms

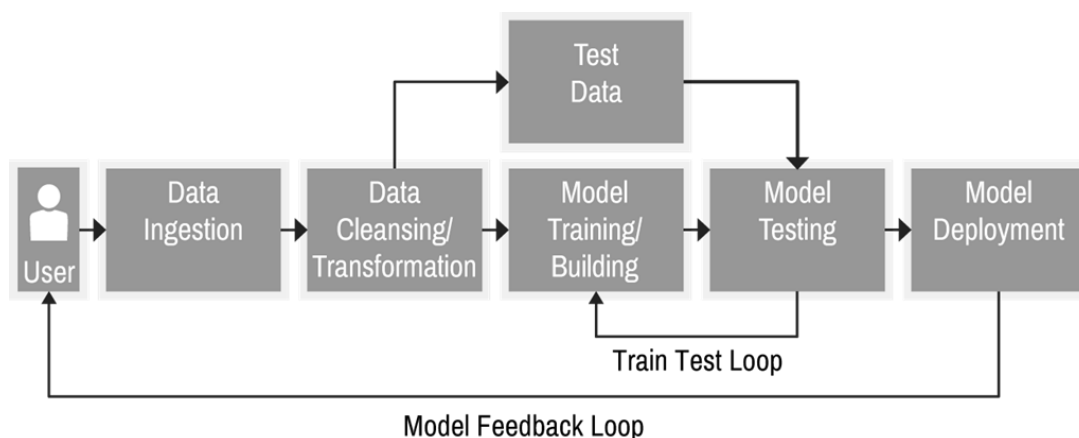
There are three major categories of Machine Learning algorithms:

- **Supervised Learning:** whereby the machine is provided with labeled data for both input as well as expected output during its training and the supervised learning algorithm generates a mapping function that can identify the expected output for a given input. The training process continues till the algorithm reaches the preferred level of accuracy. One of the standardized goals of supervised learning is to make the computer learn a classification system; therefore, it is commonly used to solve classification problems. For example, the machine could be trained to classify a spam e-mail from a valid e-mail, already being used by Google for Gmail spam filtering. Nearest neighbor, Naïve Bayes, Decision Trees, Linear Regression, Support Vector Machines and Neural Networks are a few of the most common algorithms that are included under this category.
- **Unsupervised Learning:** whereby the machine is provided with unlabeled and unclassified input dataset and the unsupervised learning algorithm generates a function to identify hidden structures in the given dataset as per the patterns, similarities and differences that exist among data without any former training. There is no assessment of the level of accuracy of the structure identified by the machine. One of the major focuses of unsupervised learning algorithms could be clustering and association problems. A few of the commonly used unsupervised learning algorithms are k-means algorithm for clustering and Apriori algorithm for association problems.
- **Reinforcement Learning:** the machine is exposed to an environment where it takes decisions on a trial and error basis and learns from its own actions and past experiences. For every correct decision the machine receives a reward feedback from the environment that acts like a reinforcement signal and the information about the rewarded state-action pair is stored. Later on the machine iterates the rewarded behavior whenever faced with a similar situation. Reinforcement learning algorithms have their usage in domains where strategic decision making is the key to success like Self Driving Cars. Few of the most commonly used

reinforcement learning algorithms are Q-Learning and Markov Decision processes.

Machine Learning Workflow

A presentation created by Dimitrov (2017) depicted the Machine Learning workflow in the following diagram:



The sequence of workflow starts with data ingestion that is followed by data preprocessing where the valid and clean data is transformed into a format that is best suited to the requirements of machine learning model used. According to Puget (2016) the training part of the workflow involves the use of a machine learning algorithm that generates a trained model. The trained model is then iteratively tested for accuracy against some test data which ideally must be a different set of data each time the model is tested. After meeting the desired accuracy the model is then deployed so that the application can use the model.

Why Python?

“Python is like a swiss army knife of machine learning” - Thia Kai Xin (Data Scientist, TUM)

Priyadharshini (2017) has cited in her article that although the machine learning models have been known and studied for a long period of time but the ability to apply intricate mathematical computations to big data automatically, iteratively and rapidly has gained impetus over the most recent years.

There are a large number of reasons for python to be considered as a popular language for machine learning. As stated by Harrington (2012) Python is a simple and elegant language with clear syntax having extremely easy text manipulation and processing capabilities and a rich community of developers leading to availability of ample documentation. Python also comes with an interactive shell that allows the programmers to examine and view elements of a program simultaneously while writing the code. Python has been equipped with high-level data types like lists, queues, tuples, dictionaries, etc. that make it easy to implement abstract concepts and do not need to be programmed explicitly by the programmer.

Machine learning could also be easily implemented using other high level languages like MATLAB that provides a number of built-in features for performing matrix mathematics but it is not an open source language. Moving towards languages like C and Java, matrix math libraries are available for them as well but it requires a lot of code to perform simple things in these languages. These limitations have been overcome by python that is an open source, clean, brief and easy to read language with an availability of large number of libraries and packages makes python more popular in various domains including scientific and financial communities. Moreover, python does not require expert programming skills to catch up with its code, even non programmers could easily learn and code in python.

Basic Applications of Machine Learning Using Python

A few basic applications of machine learning have been explored and implemented with respect to this research using python programming language. The names of the applications are as follows:

- Spam Filter
- Sentiment Analysis
- Voice Chat-Bot
- Image Classifier
- Stock Market Prediction

The following specifications have been used to implement and code the above mentioned applications:

Version Used: Python 3.6

Libraries Used: Scikit-learn (majorly used), NumPy, SciPy, NLTK, Pandas, Keras, Matplotlib, PIL, ChatterBot and TextBlob

Available on SSRN-Elsevier

Packages Used: pyttsx,

API Used: speech recognition by Google

According to official python documentation, Scikit-Learn is a simple, efficient and powerful tool that provides various machine learning algorithms for classification, clustering, regression, model selection, etc. It is an open source library easily accessible to anybody and usable in many contexts. In order to work with Scikit-Learn, NumPy must be installed along with SciPy and Matplotlib. NumPy is used to work with arrays; Matplotlib is used to plot graphs, Keras for Preprocessing image, NLTK for word net, Pandas for reading CSV data file, Pyttsx for text to speech, Textblob for sentiment analysis, chatterbot for chatting application and PIL for image manipulation.

▪ **Spam Filter**

- What? A machine learning application built in python to filter spam messages.
- Why? As the name suggests, this application detects and filters unsolicited and unwanted messages received by user.
- Where? This application is currently being used by Google for Gmail spam filtering.
- Implementation using Python? Pandas library was used for reading the CSV data file, then the data was labeled for converting it from word to a vector containing two columns, Status and Message, where the Status was labeled as 1 indicating ham and 0 indicating spam. The labeled data was then given to the model for training. In order to check whether the message input by the user is spam or not, it was first converted from word to vector and then given to the model for prediction.
- Model Used: Multinomial NaiveBayes (NB) from sklearn
- Code Snippet:
 - `tf= TfidfVectorizer ()`
 - `mnb= MultinomialNB ()`
 - `test= input("Enter:")`
 - `test_data= []`
 - `test_data.append(test)`
 - `df= pd.read_csv('SMSSpamCollection', sep= '\t', names= ['Status', 'Message'])`
 - `df.loc[df['Status']== 'ham', "Status"]=1`

- `df.loc[df['Status'] == 'spam', "Status"] = 0`
 - `df_x=df['Message']`
 - `df_y=df['Status']`
 - `x_train=tf.fit_transform(df_x).toarray()`
 - `y_train= df_y.astype('int')`
 - `x_train,x_test,y_train,y_test = train_test_split(x_train,y_train)`
 - `X = tf.transform(test_data).toarray()`
 - `mnf.fit(x_train, y_train)`
 - `pred= mnf.predict(X)`
 - `if pred==1:`
 - `print ("not spam")`
 - `else:`
 - `print("spam")`
 - `pred = mnf.predict(x_test)`
 - `sc = accuracy_score(y_test,pred)`
 - `print ("Accuracy of the Model: { } %"format (sc*100))`
- Results:

```

Python 3.6.5 Shell
File Edit Shell Debug Options Window Help

Enter:Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2
005. Text FA to 87121 to receive entry question(std txt rate)T&C's ap
ply 08452810075over18's
spam
Accuracy of the Model: 94.54414931801867%
>>>
===== RESTART: E:\Research\IC-18\spamfilter.py =====
Enter:I'm gonna be home soon and i don't want to talk about this stuf
f anymore tonight, k? I've cried enough today.
not spam
Accuracy of the Model: 96.05168700646087%
>>>
===== RESTART: E:\Research\IC-18\spamfilter.py =====
Enter:Yes..gauti and sehwaq out of odi series.

not spam
Accuracy of the Model: 95.40559942569993%
>>>
===== RESTART: E:\Research\IC-18\spamfilter.py =====
Enter:Did you hear about the new \Divorce Barbie\"? It comes with all
of Ken's stuff!
not spam
Accuracy of the Model: 96.4824120603015%
>>>
    
```

- Interpretation: The spam filter model was trained on a separate predefined training dataset and also tested on a predefined dataset which was different from training dataset. Both the datasets were labeled to identify spam from not spam. The results show that the model was able to distinguish between

“spam” and “not spam” with an accuracy of about 94% to 96% approximately. However, the accuracy of this model could be further increased by adjusting its parameters.

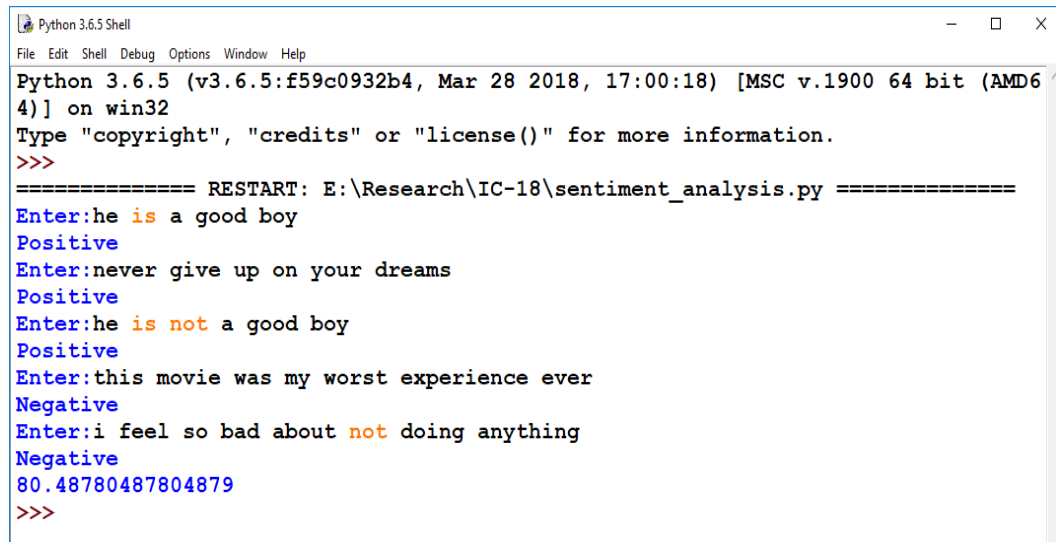
- ***Sentiment Analysis***

- What? A machine learning application built in python to check polarity of given piece of text.
- Why? As the name suggests, this application is used for computationally identifying and categorizing opinions expressed in a piece of text.
- Where? This application is currently being used by Twitter (Twitter Sentiment Analysis) for analyzing the polarity of a particular tweet.
- Implementation using Python? TextBlob library was used for importing the required model; NLTK was used for word-net which provided the initial training data. The new data was labeled and stored in a list by the researchers. This list was then given to the model as new training data. Then the message provided as input by the user was given to the model for predicting whether it was positive or negative.
- Model Used: NaiveBayes Classifier from TextBlob, accuracy_score from sklearn

- Code Snippet:

```
from customdata import data
da=data ()
te = test ()
cl = NaiveBayesClassifier(da)
predicted = []
label = []
inp = input("Enter:")
result = (cl.classify(inp))
if result== "pos":
    print("Positive")
else:
    print("Negative")
for i in te:
    pred = cl.classify(i[o])
    predicted.append(pred)
    label.append(i[1])
sc = accuracy_score(label,predicted)
print(sc*100)
```

- Results:



```
Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 17:00:18) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:\Research\IC-18\sentiment_analysis.py =====
Enter:he is a good boy
Positive
Enter:never give up on your dreams
Positive
Enter:he is not a good boy
Positive
Enter:this movie was my worst experience ever
Negative
Enter:i feel so bad about not doing anything
Negative
80.48780487804879
>>>
```

- **Interpretation:** The model was provided with a previously defined, customized and preprocessed dataset of approximately 3000 positive and negative sentences and then the model was asked to classify between the two of them. The inputs were then arbitrarily given to the model and the outputs were as per expectations around 80% of the times. The accuracy of this model could be improved by increasing the amount of good quality data for its training.

▪ **Voice Chat-Bot**

- **What?** A machine learning application built in python for interacting with an AI.
- **Why?** As the name suggests, this application is used for voice chatting with an AI bot.
- **Where?** A more advanced version of this application is currently being used by Google as Google Assistant, by Microsoft as Cortana, by Apple as Siri and by Amazon as Alexa.
- **Implementation using Python?** Chatterbot library was used for importing the model required, speech recognition API by Google was used for recognizing the user's voice, pyttsx was used to convert text to speech. The model was trained over the data provided by word-net from NLTK. Speech recognizer and microphone was made active by writing a line of code along with setting the output voice property. Then a voice input was provided by the user and the model responded to it accordingly.
- **Model Used:** Chatbot from chatterbot
- **Code Snippet:**

```
import os
```

```

import pyttsx
import speech_recognition as sr
from chatterbot import ChatBot
chatbot = ChatBot('Ron Obvious',
trainer='chatterbot.trainers.ChatterBotCorpusTrainer')
chatbot.train("chatterbot.corpus.english")
r = sr.Recognizer()
m = sr.Microphone()
e= pyttsx.init()
voices =e.getProperty('voices')
e.setProperty ('voice',voices[1])
def my_form ():
    try:
        print ("A moment of silence, please...")
        with m as source: r.adjust_for_ambient_noise(source)
        print ("Set minimum energy threshold to {}".
format(r.energy_threshold))
        r.energy_threshold=400
        for i in range (10):
            print ("Say something!")
            r.dynamic_energy_threshold = False
            with m as source: audio = r.listen(source)
            print ("Got it! Now to recognize it.")
            try:
                #recognize speech using Google Speech Recognition
                value = r.recognize_google(audio)
                x=str(value)
                print ("User: ", x)
                pp=chatbot.get_response(x)
                print ("Bot: ", pp)
                e.say(pp)
                e.runAndWait()
            except:
                print ("sorry")
        except KeyboardInterrupt:
            pass
if __name__=="__main__":
    my_form ()

```

- Results:

```
A moment of silence, please...
Set minimum energy threshold to 407.12393085101485
Say something!
Got it! Now to recognize it..
User:  it's getting late in the evening
Bot:  That's not a very nice thing to say.
Say something!
Got it! Now to recognize it..
sorry
Say something!
Got it! Now to recognize it..
User:  I am very tired
Bot:  Why?
Say something!
Got it! Now to recognize it..
User:  because I had to work for late
Bot:  me working harder is an oxymoron.
Say something!
Got it! Now to recognize it..
sorry
Say something!
Got it! Now to recognize it..
sorry
Say something!
Got it! Now to recognize it..
User:  don't say sorry
Bot:  People worry about things all the time.
Say something!
Got it! Now to recognize it..
```

- Interpretation: The chat bot was trained on a complete corpus of dialog data in English language and then users could converse with it just like any other human being. The bot was able to recognize the words being said to it and respond accordingly as visible in the output screen.

▪ ***Image Classifier***

- What? A machine learning application built in python for image classification.
- Why? As the name suggests, this application is used for categorizing images into groups.
- Where? This application is currently being used in object detection algorithms.
- Implementation using Python? Scikit-learn library was used for importing the required model. The labeled training data was manipulated, resized, reshaped and transposed using PIL and NumPy. This image data was then converted into an array using KERAS library. The model was trained over this data. In order to classify the image given by the user, it was first manipulated, resized, reshaped and transposed and converted to an array of

same size as training data then further the category of given image was predicted by the model.

- Model Used: Multi-Layer Perceptron (MLP Classifier) from scikit-learn
- Code Snippet:

```
dpath= r'E:\Research\IC-18\cat-and-dog\training_set\dogs'
cpath= r'E:\Research\IC-18\cat-and-dog\training_set\cats'
def convert(path):
    label = []
    ad = []
    c=0
    for file in os.listdir(path):
        label.append(file.split('.')[0])
        f=os.path.join(path,file)
        im= Image.open(f)
        im= im.convert(mode='RGB')
        imrs= im.resize((100,100))
        imrs= img_to_array(imrs)/255
        #imrs.transpose((2, 0, 3, 1)).reshape(4,4)
        imrs= imrs.reshape(100*100*3)
        ad.append(imrs)
        c+=1
    print("done")
    return ad,label

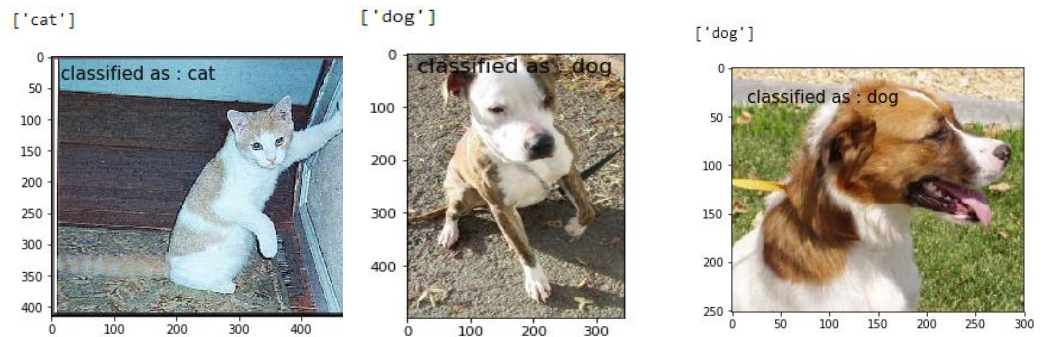
mlp= MLPClassifier (activation= 'logistic', solver= 'sgd')
cat_data,cat_label = convert(cpath)
dog_data,dog_label = convert(dpath)
main_data = cat_data+dog_data
main_label = cat_label+dog_label
main_data = np.asarray(main_data)
main_label = np.asarray(main_label)
main_label= main_label.reshape(len(main_label), -1)
print(main_data.shape)
print("started")

mlp.fit(main_data, main_label)

t_data = []
t = r'E:\Research\IC-18\cat-and-dog\test_set\cats\cat.4001.jpg'
te = Image.open(t)
tes = te.convert(mode="RGB")
test = tes.resize((100,100))
test = img_to_array(test)/255
ta = test
testa = test.reshape(3*100*100)
t_data.append(testa)
```

```
t_data = np.asarray(t_data)
pred= mlp.predict(t_data)
image = mpimg.imread(t)
print(pred)
tr= ("classified as: {}".format(str(pred[0])))
plt.text (15, 35, tr, fontsize=15)
plt.imshow(image)
plt.show()
```

■ **Results:**



- **Interpretation:** The dataset containing around 8000 images of cats and dogs was used to train the image classifier model and for testing a separate dataset was used containing approximately 2000 images of cats and dogs. The model was executed three times during testing with different image paths. The results show that the model was fairly able to recognize cats and dogs separately. If the size of dataset was reduced, as an experiment by the researchers, the accuracy of the model was affected in a negative way.

■ ***Stock Market Opening Price Prediction***

- **What?** A machine learning application built in python for predicting opening prices of a stock in the stock market.
- **Why?** As the name suggests, this application is used for predicting prices of shares of a particular company listed in the stock market based on past data of one year and ten months
- **Where?** This application is currently not being used anywhere. For testing purposes, the authors tried in this research paper to design and implement such a model using python, and with accurate data this model can be used for predicting opening prices of a particular stock in the stock market.
- **Implementation using Python?** Scikit-learn library was used for importing the model required. Matplotlib was used for making graphs and pandas library was used for reading data from CSV file. Then the dates and adjusted closing prices column from given dataset were parsed and reshaped into desired format and the model was trained over this data. Then as per the

requirements, this model was used to predict approximate opening prices of stocks for the next month, with an aim of achieving maximum accuracy.

- Model Used: DecisionTreeRegressor from scikit-learn, numpy, pandas
- Code Snippet:

```

dates = []
prices= []
adjacent_close = []
with open ('rel22train.csv', 'r') as csvfile:
    csvFileReader = csv.reader(csvfile)
    next(csvFileReader)
    for row in csvFileReader:
        if row [1] == "null":
            continue
        dates.append(''.join(row[0].split('-')))
        prices.append(float(row[1]))
        adjacent_close.append(float(row[5]))

new_data = []
common= []
new_prices= []
for i in range(len(dates)-1):
    new_data.append([dates[i],adjacent_close[i]])
    new_prices.append(prices[i+1])
new_data = np.asarray(new_data,dtype='float32')
new_data = np.reshape(new_data,(len(new_data), 2))
new_prices = np.asarray(new_prices,dtype='float32')
new_prices = np.reshape(new_prices,(len(new_prices), 1))

dtr= DecisionTreeRegressor()
dtr.fit(new_data,new_prices)
test_dates = []
test_prices = []
test_adj = []
actual_date = []
t = 0
with open ('reltestapril.csv', 'r') as csvfile:
    csvFileReader = csv.reader(csvfile)
    next(csvFileReader)
    for row in csvFileReader:
        if row [1] == "null":
            continue
        if t>0:
            actual_date.append(row[0])
            test_dates.append(''.join(row[0].split('-')))
            test_prices.append(float(row[1]))

```



```

        test_adj.append(float(row[5]))
        t+=1
    test_new_data = []
    test_new_prices = []
    for i in range(len(test_dates)-1):
        test_new_data.append([test_dates[i],test_adj[i]])
        test_new_prices.append(test_prices[i+1])
    test_new_data=np.asarray(test_new_data,dtype='float32')
    test_new_data = np.reshape(test_new_data,(len(test_new_data), 2))
    test_new_prices = np.reshape(test_new_prices,(len(test_new_prices), 1))
    pred = dtr.predict(test_new_data)
    data = [i for i in zip(actual_date,test_new_prices,pred)]
    df = pd.DataFrame(data,columns=['Date','Actual','Predicted'])
    print(df)
    ac = dtr.score(test_new_data,test_new_prices)
    print ("Accuracy: ", ac*100,"%")

```

■ Results:

April	>>>		
	===== RESTART: E:\Research\IC-1		
	\stockprediction1.py =====		
	Date	Actual	Predicted
	0 2018-04-03	[891.0]	893.950012
	1 2018-04-04	[904.700012]	895.799988
	2 2018-04-05	[905.099976]	893.950012
	3 2018-04-06	[908.0]	890.599976
	4 2018-04-09	[912.549988]	909.250000
	5 2018-04-10	[918.400024]	923.000000
	6 2018-04-11	[921.799988]	923.000000
	7 2018-04-12	[929.849976]	931.900024
	8 2018-04-13	[932.700012]	928.150024
	9 2018-04-16	[934.450012]	939.000000
	10 2018-04-17	[940.0]	939.000000
	11 2018-04-18	[945.25]	939.000000
	12 2018-04-19	[939.099976]	939.000000
	13 2018-04-20	[938.0]	939.000000
	14 2018-04-23	[930.0]	928.150024
	15 2018-04-24	[935.799988]	928.799988
	16 2018-04-25	[973.650024]	981.750000
	17 2018-04-26	[973.0]	981.750000
	18 2018-04-27	[989.799988]	981.750000
	19 2018-04-30	[982.0]	981.750000
	Accuracy: 93.51395499643739 %		

May	>>>		
	===== RESTART: E:\Research\IC-18		
	Date	Actual	Predicted
	0 2018-05-03	[977.900024]	975.000000
	1 2018-05-04	[962.849976]	963.250000
	2 2018-05-07	[958.849976]	943.000000
	3 2018-05-08	[977.5]	975.000000
	4 2018-05-09	[965.900024]	963.250000
	5 2018-05-10	[980.0]	975.000000
	6 2018-05-11	[981.25]	981.750000
	7 2018-05-14	[991.950012]	981.750000
	8 2018-05-15	[983.049988]	981.750000
	9 2018-05-16	[974.0]	981.750000
	10 2018-05-17	[959.349976]	943.000000
	11 2018-05-18	[945.599976]	939.000000
	12 2018-05-21	[930.75]	928.799988
	13 2018-05-22	[933.150024]	928.799988
	14 2018-05-23	[926.0]	925.000000
	15 2018-05-24	[914.0]	909.000000
	16 2018-05-25	[916.0]	923.000000
	17 2018-05-28	[924.900024]	921.799988
	18 2018-05-29	[921.5]	921.799988
	19 2018-05-30	[914.849976]	923.000000
	20 2018-05-31	[921.0]	923.000000
	Accuracy: 93.49089719386347 %		
June	>>>		
	===== RESTART: E:\Research\IC-18		
	Date	Actual	Predicted
	0 2018-06-04	[933.400024]	931.900024
	1 2018-06-05	[942.799988]	939.000000
	2 2018-06-06	[947.0]	943.000000
	3 2018-06-07	[954.900024]	943.000000
	4 2018-06-08	[966.349976]	975.000000
	5 2018-06-11	[987.099976]	981.750000
	6 2018-06-12	[983.400024]	981.750000
	7 2018-06-13	[1000.0]	981.750000
	8 2018-06-14	[997.450012]	981.750000
	9 2018-06-15	[1006.0]	981.750000
	10 2018-06-18	[1008.799988]	981.750000
	11 2018-06-19	[1013.900024]	981.750000
	12 2018-06-20	[996.0]	981.750000
	13 2018-06-21	[1023.450012]	981.750000
	14 2018-06-22	[1028.400024]	981.750000
	15 2018-06-25	[1011.400024]	981.750000
	16 2018-06-26	[1006.0]	981.750000
	17 2018-06-27	[978.5]	981.750000
	18 2018-06-28	[962.0]	975.000000
	19 2018-06-29	[949.099976]	943.000000
	Accuracy: 43.12324988768421 %		

July	>>>		
	===== RESTART: E:\Research\IC-18		
	Date	Actual	Predicted
	0 2018-07-03	[965.0]	967.900024
	1 2018-07-04	[971.799988]	975.000000
	2 2018-07-05	[997.0]	981.750000
	3 2018-07-06	[964.75]	975.000000
	4 2018-07-09	[987.099976]	981.750000
	5 2018-07-10	[1002.75]	981.750000
	6 2018-07-11	[1025.0]	981.750000
	7 2018-07-12	[1044.349976]	981.750000
	8 2018-07-13	[1080.5]	981.750000
	9 2018-07-16	[1099.800049]	981.750000
	10 2018-07-17	[1078.300049]	981.750000
	11 2018-07-18	[1098.400024]	981.750000
	12 2018-07-19	[1093.300049]	981.750000
	13 2018-07-20	[1113.400024]	981.750000
	14 2018-07-23	[1129.800049]	981.750000
	15 2018-07-24	[1122.0]	981.750000
	16 2018-07-25	[1110.0]	981.750000
	17 2018-07-26	[1110.099976]	981.750000
	18 2018-07-27	[1119.5]	981.750000
	19 2018-07-30	[1130.0]	981.750000
	20 2018-07-31	[1151.0]	981.750000
	Accuracy: -193.90237668445502 %		

- Interpretation: The model was given a large dataset (CSV file) containing the stock market data of Reliance Industries Ltd. (listed on NSE) for the past twenty two and a half years, ranging from January 1, 1996 to March 31, 2018, downloaded from Yahoo! Finance. The model was then trained to recognize patterns between adjusted closing price of the stock on previous day and the opening stock price for the next day with respect to all the dates in the data and then it was tested to predict the value of stock for each date of the month of April, 2018 first, following the same for the months of May, June, and July 2018 as well. The output depicted the date, predicted value of stock and actual value of stock as shown in the results. As visible from the results, the accuracy of the model was maximum when it was used to predict data for the consecutive months that occurred immediately after the training data ended i.e., for April and May, 2018, it was most accurate with an accuracy rate of 93% approximately, and then it decreased to 43% approximately, and even went towards a highly negative level of -193% as the gap between time periods of training and testing datasets was increased.

Conclusion

The concept of Machine Learning is being adopted and implemented in a wide variety of applications in this new tech savvy age of technocrats. Its understanding and use has accomplished the unthinkable. People can literally talk to their smart devices and get an answer in their own language. They are being used to distinguish between objects and also identify various objects based on their features. The machine has already started behaving like a human being. This plethora of opportunities was the actual motivation behind conducting this study.

Available on SSRN-Elsevier

In this research paper, the researchers tried to establish an elementary understanding of supervised machine learning algorithms and related concepts by building a few fundamental applications using python. Coding these algorithms in Python for this research became simple because of Python's vast set of APIs and modules that came as a handy tool for a programmer and helped to focus more on creating logic. Moreover, Python's syntax was also very easy to code and implement. Thus, Python was chosen as a preferable development platform.

Four of the developed models i.e., Spam Filter, Chat bot, Image Classifier and Sentiment Analyzer were found to be accurate enough to be deployed in real world applications. Moreover, their accuracy could be improved further by training them on more amounts of data and adjusting their parameters accordingly for example, learning rate, optimizers, etc.

Fifth model of the study i.e., the Stock Market Opening Price Prediction produced high levels of accuracy when trained upon a raw dataset of adjusted closing prices and opening stock prices of Reliance Industries Ltd. and tested on the data of immediate consecutive months but its accuracy decreased drastically as the gap between the time period of training and test dataset was increased. This model was developed out of curiosity and willingness of learning, only for the purpose of this research and wasn't found suitable as a solution for the real world problem of predicting accurate stock prices for better returns on investments, for the sole reason that stock prices are multidimensional in nature and get affected by various micro and macro economic factors as well as other factors which were not explored in this research. Therefore, the researchers do not recommend using this model in the real world as it was only meant to be for experimentation purposes and did not take into account all the factors and adjustments that actually need to be considered while predicting stock market opening prices. These factors could be explored and used in future researches in the same dimension to gain more accuracy and develop a model that could be actually implemented in real life to predict stock market functioning.

To conclude, while conducting and learning from this research, the authors came to a common consensus that, "Right amount of data in the Right Format given to the Right Algorithm by using Right Parameters is the key to Success rate of Machine Learning...!"

References

Brownlee, J. (2015, December 25). *Basic Concepts in Machine Learning*. Retrieved January 4, 2018, from Machine Learning Mastery: <https://machinelearningmastery.com/basic-concepts-in-machine-learning/>

- Brownlee, J. (2016, August 16). *What is Deep Learning?* Retrieved December 30, 2017, from Machine Learning Mastery: <https://machinelearningmastery.com/what-is-deep-learning/>
- Chen, S. (2017, June 16). *A Basic Machine Learning Workflow In Production*. Retrieved January 5, 2018, from Medium: <https://medium.com/eliza-effect/how-machines-learn-d9e9a3e6f97c>
- Dey, A. (2016). Machine Learning Algorithms: A Review. *International Journal of Computer Science and Information Technologies (IJCSIT)* , 7 (3), 1174-1179.
- Dimitrov, K. (2017, February 26). *Tokyo Azure Meetup #11 Introduction to Azure Machine Learning*. Retrieved January 4, 2018, from SlideShare: <https://www.slideshare.net/TokyoAzureMeetup/tokyo-azure-meetup-11-introduction-to-azure-machine-learning>
- Gupta, B., Negi, M., Vishwakarma, K., Rawat, G., & Badhani, P. (2017). Study of Twitter Sentiment Analysis using Machine Learning Algorithms on Python. *International Journal of Computer Applications* , 165 (9), 30-34.
- Haffner, P. (2016, July 7). *What is Machine Learning – and Why is it Important?* Retrieved January 4, 2018, from interactions: <https://www.interactions.com/machine-learning-important/>
- Harrington, P. (2012). *Machine Learning in Action*. New York, United States of America: Manning Publications.
- Malhotra, P. (2016, June 24). Why Machine Learning is being given so much Importance? Delhi, Delhi, India.
- Marr, B. (2016, February 19). *A Short History of Machine Learning- Every Manager Should Read*. Retrieved December 28, 2017, from Forbes: <https://www.forbes.com/sites/bernardmarr/2016/02/19/a-short-history-of-machine-learning-every-manager-should-read/#3cb3d02e15e7>
- Mostafa, Y. A. (2012, April 3). Learning From Data. Pasadena, California, USA. Retrieved January 5, 2018, from <https://www.youtube.com/watch?v=mbyG85GZoPI&list=PLD63A284B7615313A>
- Pacheco, V. G. (2016, November 8). *A Brief History of Machine Learning*. Retrieved December 28, 2017, from Synergic Partners: <http://www.synergicpartners.com/en/espanol-una-breve-historia-del-machine-learning/>
- Priyadharshini. (2017, December 15). *Machine Learning: What it is and Why it Matters*. Retrieved December 29, 2017, from Simpli Learn: <https://www.simplilearn.com/what-is-machine-learning-and-why-it-matters-article>
- Puget, J. F. (2016, April 27). *Machine Learning Algorithm != Learning Machine (IT Best Kept Secret is Optimization)*. Retrieved January 4, 2018, from IBM Developer Works: https://www.ibm.com/developerworks/community/blogs/jfp/entry/Machine_Learning_Learning_Machine?lang=en
- Puget, J. F. (2016, May 18). *What is Machine Learning? (IT Best Kept Secret is Optimization)*. Retrieved December 30, 2017, from IBM Developer Works: https://www.ibm.com/developerworks/community/blogs/jfp/entry/What_Is_Machine_Learning?lang=en
- Samuel, A. L. (1959, July). Some Studies in Machine Learning using the Game of Checkers. *IBM Journal of Research and Development* , 210-229.
- Tagliaferri, L. (2017, September 28). *An Introduction to Machine Learning*. Retrieved December 29, 2017, from DigitalOcean: <https://www.digitalocean.com/community/tutorials/an-introduction-to-machine-learning>

- Xin, T. K. (2016, April 16). Why is Python so Popular in Machine Learning? Munich, Bavaria, Germany.
- Taiwo Oladipupo Ayodele (2010). Types of Machine Learning Algorithms, New Advances in Machine Learning, Yagang Zhang (Ed.), InTech, DOI: 10.5772/9385. Available from: <https://www.intechopen.com/books/new-advances-in-machine-learning/types-of-machine-learning-algorithms>