http://www.mtaeducation.in/images/logo.png

**Project Report**

**Summer Industrial Training - 2018**

**Domain of Project : Machine Learning using Python**

**Topic : Accuracy Prediction using Naïve Bayes**

**Classification of user knowledge modeling**

**Clustering of user knowledge modeling**

**Evaluation of reviews in twitter data**

**Chat Bots inter conversation**

**Duration : 1 month**

**Date of Submission: 04/06/2018**

**Report Submitted by**

**Shubham Banerjee**

**College - KIIT**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Name of the Topic** | **Page** |
|  |  |  |
|  |  |  |
|  |  |  |

Index

**I would like to express my deepest appreciation to all those who provided me the possibility to complete this report.  A special gratitude I give to our Instructor, Mr. Gagandeep Thakur, whose contribution in stimulating suggestions and encouragement,  helped me to coordinate my project especially in writing this report.**

**Signature of Teacher**

Acknowledgement

Classification (Naïve Bayes)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Adult Data Set**  *Download*: [Data Folder](http://archive.ics.uci.edu/ml/machine-learning-databases/adult/), [Data Set Description](http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.names)  **Abstract**: Predict whether income exceeds $50K/yr based on census data. Also known as "Census Income" dataset. | http://archive.ics.uci.edu/ml/assets/MLimages/Large2.jpg |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Data Set Characteristics:** | Multivariate | **Number of Instances:** | 48842 | **Area:** | Social | | **Attribute Characteristics:** | Categorical, Integer | **Number of Attributes:** | 14 | **Date Donated** | 1996-05-01 | | **Associated Tasks:** | Classification | **Missing Values?** | Yes | **Number of Web Hits:** | 1185948 |   **Source:**  Donor:   Ronny Kohavi and Barry Becker  Data Mining and Visualization  Silicon Graphics.  e-mail: ronnyk '@' live.com for questions.  **Data Set Information:**  Extraction was done by Barry Becker from the 1994 Census database. A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1)&& (HRSWK>0))   Prediction task is to determine whether a person makes over 50K a year.  **Attribute Information:**  Listing of attributes:   >50K, <=50K.   age: continuous.  workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.  fnlwgt: continuous.  education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.  education-num: continuous.  marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.  occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.  relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.  race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.  sex: Female, Male.  capital-gain: continuous.  capital-loss: continuous.  hours-per-week: continuous.  native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands. |

**Python Script**

**# -\*- coding: utf-8 -\*-**

**"""**

**Created on Mon May 28 16:26:23 2018**

**@author: Shubham Banerjee**

**"""**

**#Importing Libraries**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import pandas as pd**

**#importing dataset**

**dataset = pd.read\_csv('adult\_data.csv')**

**X = dataset.iloc[:,:-1].values**

**y = dataset.iloc[:,14].values**

**#replacing categorical data**

**from sklearn.preprocessing import LabelEncoder,OneHotEncoder**

**labenc\_X = LabelEncoder()**

**X[:,1] = labenc\_X.fit\_transform(X[:,1])**

**X[:,3] = labenc\_X.fit\_transform(X[:,3])**

**X[:,5] = labenc\_X.fit\_transform(X[:,5])**

**X[:,6] = labenc\_X.fit\_transform(X[:,6])**

**X[:,7] = labenc\_X.fit\_transform(X[:,7])**

**X[:,8] = labenc\_X.fit\_transform(X[:,8])**

**X[:,9] = labenc\_X.fit\_transform(X[:,9])**

**X[:,13] = labenc\_X.fit\_transform(X[:,13])**

**labenc\_y = LabelEncoder()**

**y = labenc\_y.fit\_transform(y)**

**#removing missing values**

**from sklearn.preprocessing import Imputer**

**imp = Imputer(missing\_values="NaN", strategy="mean",axis=0)**

**imp.fit(X)**

**X = imp.transform(X)**

**#splitting datasets into training and testing sets**

**from sklearn.cross\_validation import train\_test\_split**

**X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,y,train\_size=0.75,random\_state=0)**

**#feature scaling**

**from sklearn.preprocessing import StandardScaler**

**sc\_X = StandardScaler()**

**X\_train = sc\_X.fit\_transform(X\_train)**

**X\_test = sc\_X.fit\_transform(X\_test)**

**#creating the naivebayes model**

**from sklearn.naive\_bayes import GaussianNB**

**classifier = GaussianNB()**

**classifier.fit(X\_train,y\_train)**

**#predicting the results**

**y\_pred = classifier.predict(X\_test)**

**#creating the confusion matrix**

**from sklearn.metrics import confusion\_matrix**

**cm = confusion\_matrix(y\_test,y\_pred)**

**from sklearn.metrics import accuracy\_score**

**percentage\_accuracy = accuracy\_score(y\_test,y\_pred)\*100**

CM output

|  |  |
| --- | --- |
| **181** | **28** |
| **42** | **78** |

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | |  |  | |

**Classification**

|  |  |
| --- | --- |
| **User Knowledge Modeling Data Set**  *Download*: [Data Folder](https://archive.ics.uci.edu/ml/machine-learning-databases/00257/), [Data Set Description](https://archive.ics.uci.edu/ml/datasets/User+Knowledge+Modeling)  **Abstract**: It is the real dataset about the students' knowledge status about the subject of Electrical DC Machines. The dataset had been obtained from Ph.D. Thesis. |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data Set Characteristics:** | Multivariate | **Number of Instances:** | 403 | **Area:** | Computer |
| **Attribute Characteristics:** | Integer | **Number of Attributes:** | 5 | **Date Donated** | 2013-06-26 |
| **Associated Tasks:** | Classification, Clustering | **Missing Values?** | N/A | **Number of Web Hits:** | 81627 |

**Source:**

-- Creators: Hamdi Tolga Kahraman (htolgakahraman **'@'** yahoo.com)   
-- Institution: Faculty of Technology, Department of Software Engineering, Karadeniz Technical University, Trabzon, Turkiye   
-- Creators: Ilhami Colak (icolak **'@'** gazi.edu.tr)   
-- Institution: Faculty of Technology, Department of Electrical and Electronics Engineering, Gazi University, Ankara, Turkiye   
-- Creators: Seref Sagiroglu (ss **'@'** gazi.edu.tr)   
-- Institution: Faculty of Technology, Department of Computer Engineering, Gazi University, Ankara, Turkiye   
  
-- Donor: undergraduate students of Department of Electrical Education of Gazi University in the 2009 semester   
-- Date: October, 2009

**Data Set Information:**

-- The users' knowledge class were classified by the authors   
using intuitive knowledge classifier (a hybrid ML technique of k-NN and meta-heuristic exploring methods), k-nearest neighbor algorithm.   
See article for more details on how the users' data was collected and evaluated by the user modeling server.   
  
H. T. Kahraman, Sagiroglu, S., Colak, I., Developing intuitive knowledge classifier and modeling of users' domain dependent data in web,   
Knowledge Based Systems, vol. 37, pp. 283-295, 2013.

**Attribute Information:**

STG (The degree of study time for goal object materails), (input value)   
SCG (The degree of repetition number of user for goal object materails) (input value)   
STR (The degree of study time of user for related objects with goal object) (input value)   
LPR (The exam performance of user for related objects with goal object) (input value)   
PEG (The exam performance of user for goal objects) (input value)   
UNS (The knowledge level of user) (target value)   
Very Low: 50   
Low:129   
Middle: 122   
High 130

#### Python Script

**# -\*- coding: utf-8 -\*-**

**"""**

**Created on Wed May 30 12:34:30 2018**

**@author: Shubham Banerjee**

**"""**

**#Importing Libraries**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import pandas as pd**

**#importing dataset**

**dataset = pd.read\_csv('user\_knowledge.csv').iloc[:,:6]**

**x = dataset.iloc[:,:-1].values**

**y = dataset.iloc[:,5].values**

**#applying encoding to the categorical datas**

**from sklearn.preprocessing import LabelEncoder,OneHotEncoder**

**labenc\_y = LabelEncoder()**

**y = labenc\_y.fit\_transform(y)**

**#splitting datasets into training and testing sets**

**from sklearn.cross\_validation import train\_test\_split**

**x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,train\_size=0.80,random\_state=0)**

**#feature scaling**

**from sklearn.preprocessing import StandardScaler**

**sc\_x = StandardScaler()**

**x\_train = sc\_x.fit\_transform(x\_train)**

**x\_test = sc\_x.fit\_transform(x\_test)**

**#applying the logostic regression model**

**from sklearn.linear\_model import LogisticRegression**

**classifier\_lr = LogisticRegression(random\_state=0)**

**classifier\_lr.fit(x\_train,y\_train)**

**#creating the SVM model**

**#This model however achieved even higher score using linear kernel**

**from sklearn.svm import SVC**

**classifier\_lsvm = SVC(kernel = 'linear', random\_state = 0)**

**classifier\_lsvm.fit(x\_train,y\_train)**

**#creating the SVM model**

**#This model however achieved even higher score using linear kernel**

**from sklearn.svm import SVC**

**classifier\_svm = SVC(kernel = 'rbf', random\_state = 0)**

**classifier\_svm.fit(x\_train,y\_train)**

**#using the k-nn classifier**

**from sklearn.neighbors import KNeighborsClassifier**

**classifier\_kn = KNeighborsClassifier(n\_neighbors=5,metric='minkowski',p=2)**

**classifier\_kn.fit(x\_train,y\_train)**

**#creating the DecisionTree model**

**from sklearn.tree import DecisionTreeClassifier**

**classifier\_dt = DecisionTreeClassifier(criterion = "entropy", random\_state = 0)**

**classifier\_dt.fit(x\_train,y\_train)**

**#creating the Random Forest model**

**from sklearn.ensemble import RandomForestClassifier**

**classifier\_rf = RandomForestClassifier(n\_estimators = 10, criterion = "entropy", random\_state = 0)**

**classifier\_rf.fit(x\_train,y\_train)**

**#predicting the results**

**y\_pred\_lr = classifier\_lr.predict(x\_test)**

**y\_pred\_lsvm = classifier\_lsvm.predict(x\_test)**

**y\_pred\_svm = classifier\_svm.predict(x\_test)**

**y\_pred\_kn = classifier\_kn.predict(x\_test)**

**y\_pred\_dt = classifier\_dt.predict(x\_test)**

**y\_pred\_rf = classifier\_rf.predict(x\_test)**

**#creating the confusion matrix**

**from sklearn.metrics import confusion\_matrix**

**cm\_lr = confusion\_matrix(y\_test,y\_pred)**

**cm\_lsvm = confusion\_matrix(y\_test,y\_pred)**

**cm\_svm = confusion\_matrix(y\_test,y\_pred)**

**cm\_kn = confusion\_matrix(y\_test,y\_pred)**

**cm\_dt = confusion\_matrix(y\_test,y\_pred)**

**cm\_rf = confusion\_matrix(y\_test,y\_pred)**

**#percentage accuracy score**

**from sklearn.metrics import accuracy\_score**

**percentage\_accuracy\_lr = accuracy\_score(y\_test,y\_pred\_lr)\*100**

**percentage\_accuracy\_lsvm = accuracy\_score(y\_test,y\_pred\_lsvm)\*100**

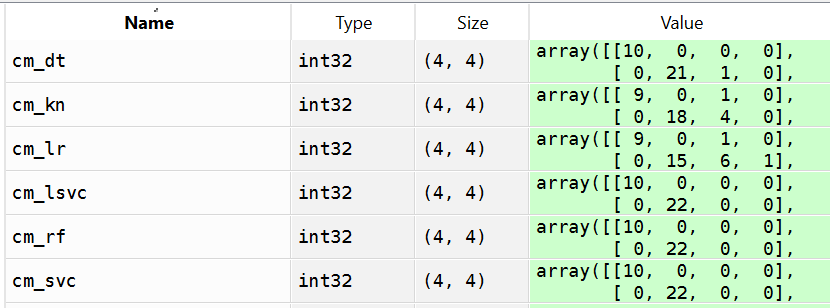
**percentage\_accuracy\_svm = accuracy\_score(y\_test,y\_pred\_svm)\*100**

**percentage\_accuracy\_kn = accuracy\_score(y\_test,y\_pred\_kn)\*100**

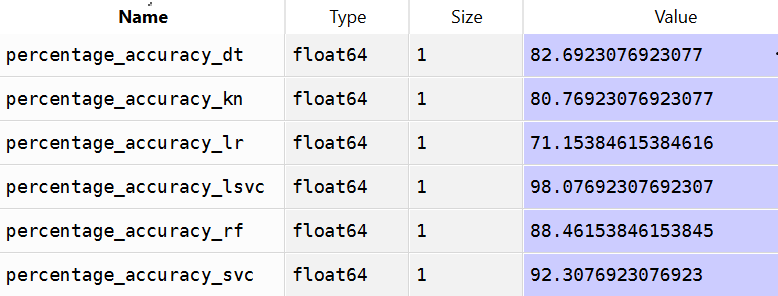
**percentage\_accuracy\_dt = accuracy\_score(y\_test,y\_pred\_dt)\*100**

**percentage\_accuracy\_rf = accuracy\_score(y\_test,y\_pred\_rf)\*100**

## CM Output



##### Percentage accuracy



**Clustering**

|  |  |
| --- | --- |
| **User Knowledge Modeling Data Set**  *Download*: [Data Folder](https://archive.ics.uci.edu/ml/machine-learning-databases/00257/), [Data Set Description](https://archive.ics.uci.edu/ml/datasets/User+Knowledge+Modeling)  **Abstract**: It is the real dataset about the students' knowledge status about the subject of Electrical DC Machines. The dataset had been obtained from Ph.D. Thesis. |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data Set Characteristics:** | Multivariate | **Number of Instances:** | 403 | **Area:** | Computer |
| **Attribute Characteristics:** | Integer | **Number of Attributes:** | 5 | **Date Donated** | 2013-06-26 |
| **Associated Tasks:** | Classification, Clustering | **Missing Values?** | N/A | **Number of Web Hits:** | 81627 |

**Source:**

-- Creators: Hamdi Tolga Kahraman (htolgakahraman **'@'** yahoo.com)   
-- Institution: Faculty of Technology, Department of Software Engineering, Karadeniz Technical University, Trabzon, Turkiye   
-- Creators: Ilhami Colak (icolak **'@'** gazi.edu.tr)   
-- Institution: Faculty of Technology, Department of Electrical and Electronics Engineering, Gazi University, Ankara, Turkiye   
-- Creators: Seref Sagiroglu (ss **'@'** gazi.edu.tr)   
-- Institution: Faculty of Technology, Department of Computer Engineering, Gazi University, Ankara, Turkiye   
  
-- Donor: undergraduate students of Department of Electrical Education of Gazi University in the 2009 semester   
-- Date: October, 2009

**Data Set Information:**

-- The users' knowledge class were classified by the authors   
using intuitive knowledge classifier (a hybrid ML technique of k-NN and meta-heuristic exploring methods), k-nearest neighbor algorithm.   
See article for more details on how the users' data was collected and evaluated by the user modeling server.   
  
H. T. Kahraman, Sagiroglu, S., Colak, I., Developing intuitive knowledge classifier and modeling of users' domain dependent data in web,   
Knowledge Based Systems, vol. 37, pp. 283-295, 2013

**Attribute Information:**

STG (The degree of study time for goal object materails), (input value)   
SCG (The degree of repetition number of user for goal object materails) (input value)   
STR (The degree of study time of user for related objects with goal object) (input value)   
LPR (The exam performance of user for related objects with goal object) (input value)   
PEG (The exam performance of user for goal objects) (input value)   
UNS (The knowledge level of user) (target value)   
Very Low: 50   
Low:129   
Middle: 122   
High 130

#### Python Script

**# -\*- coding: utf-8 -\*-**

**"""**

**Created on Wed May 30 12:11:42 2018**

**@author: Shubham Banerjee**

**"""**

**#Importing Libraries**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import pandas as pd**

**#importing dataset**

**dataset = pd.read\_csv('user\_knowledge.csv').iloc[:,:6]**

**x = dataset.values**

**#if imputer is required**

**flag = dataset.isnull().sum()**

**#applying encoding to the categorical datas**

**from sklearn.preprocessing import LabelEncoder,OneHotEncoder**

**labenc\_x = LabelEncoder()**

**x[:,5] = labenc\_x.fit\_transform(x[:,5])**

**oneHotEnc = OneHotEncoder(categorical\_features=[5])**

**x = oneHotEnc.fit\_transform(x).toarray()**

**#removing the dummy variable trap**

**x = x[:,1:]**

**#plotting the elbow**

**from sklearn.cluster import KMeans**

**wcss=[]**

**for i in range(1,11):**

**kmeans = KMeans(n\_clusters = i, init = 'k-means++', n\_init = 10,**

**max\_iter = 300, random\_state = 42)**

**kmeans.fit(x)**

**wcss.append(kmeans.inertia\_)**

**plt.plot(range(1,11),wcss)**

**plt.xlabel('No. of clusters -->')**

**plt.ylabel('wcss -->')**

**plt.title('Elbow curve')**

**plt.show()**

**#fitting our kmeans with 5 clusters and predicting the clusters**

**kmeans = KMeans(n\_clusters = 4, init = 'k-means++', n\_init = 10,**

**max\_iter = 300, random\_state = 42)**

**y\_kmeans = kmeans.fit\_predict(x)**

#### Screenshot 2018-06-03 17.49.46

**Evaluation using NLP**

#### 

## Data Set description :

###### Contributors evaluated tweets for belief in the existence of global warming or climate change. The possible answers were “Yes” if the tweet suggests global warming is occurring, “No” if the tweet suggests global warming is not occurring, and “I can’t tell” if the tweet is ambiguous or unrelated to global warming. We also provide a confidence score for the classification of each tweet.

**There are two columns in this data set in Tab separated format especially designed for the usage of Natural Language Processing to decipher the sentiments in a given tweet for the presence of the reference of Global warming and its sentiment.**

**Python Script**

**# -\*- coding: utf-8 -\*-**

**"""**

**Created on Thu May 31 17:36:35 2018**

**@author: Shubham Banerjee**

**"""**

**#Importing Libraries**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import pandas as pd**

**import json**

**from difflib import get\_close\_matches**

**data=json.load(open("data.json")).keys()**

**def correct\_word(word):**

**if word in data:**

**return word**

**elif len(get\_close\_matches(word,data))>0:**

**return get\_close\_matches(word,data)[0].lower()**

**else:**

**return word**

**#importing the data set**

**dataset = pd.read\_csv('twitter\_globalwarm.csv')**

**dataset['existence'] = dataset['existence'].replace('Y','Yes')**

**dataset['existence'] = dataset['existence'].replace('N','No')**

**#null datas in the dataset**

**nulls = dataset.isnull().sum()**

**#cleaning the dataset for useful information**

**tweet = []**

**ex = []**

**for i in range(0,len(dataset)):**

**if str(dataset['existence'][i]) != 'nan':**

**tweet.append([dataset['tweet'][i]])**

**ex.append([dataset['existence'][i]])**

**import re**

**import nltk**

**from nltk.corpus import stopwords**

**from nltk.stem.porter import PorterStemmer**

**ps = PorterStemmer()**

**corpus = []**

**for i in range(0,len(tweet)):**

**review = re.sub('[^a-zA-Z]',' ',str(tweet[i]))**

**review = review.lower()**

**review = review.split()**

**review = [ps.stem(word) for word in review if word not in set(stopwords.words('english'))]**

**review = ' '.join(review)**

**review = review.rstrip('link')**

**corpus.append(review)**

**corpus\_new = []**

**for i in range(0,len(corpus)):**

**w = corpus[i].split()**

**w = [correct\_word(word) for word in w]**

**corpus\_new.append(w)**

**corpus = corpus\_new**

**#Creating our bag of words model**

**from sklearn.feature\_extraction.text import CountVectorizer**

**cv = CountVectorizer(max\_features = 8150)**

**x = cv.fit\_transform(corpus).toarray()**

**y = ex**

**#splitting datasets into training and testing sets**

**from sklearn.cross\_validation import train\_test\_split**

**x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,train\_size=0.90,random\_state=0)**

**#feature scaling**

**from sklearn.preprocessing import StandardScaler**

**sc\_x = StandardScaler()**

**x\_train = sc\_x.fit\_transform(x\_train)**

**x\_test = sc\_x.fit\_transform(x\_test)**

**#creating the Naive bayes model**

**from sklearn.naive\_bayes import GaussianNB**

**classifier\_nb = GaussianNB()**

**classifier\_nb.fit(x\_train,y\_train)**

**#creating the DecisionTree model**

**from sklearn.tree import DecisionTreeClassifier**

**classifier\_dt = DecisionTreeClassifier(criterion = "entropy", random\_state = 0)**

**classifier\_dt.fit(x\_train,y\_train)**

**#creating the Random Forest model**

**from sklearn.ensemble import RandomForestClassifier**

**classifier\_rf = RandomForestClassifier(n\_estimators = 10, criterion = "entropy", random\_state = 0)**

**classifier\_rf.fit(x\_train,y\_train)**

**#predicting the results**

**y\_pred\_dt = classifier\_dt.predict(x\_test)**

**y\_pred\_rf = classifier\_rf.predict(x\_test)**

**y\_pred\_nb = classifier\_nb.predict(x\_test)**

**#creating the confusion matrix**

**from sklearn.metrics import confusion\_matrix**

**cm\_nb = confusion\_matrix(y\_test,y\_pred)**

**cm\_dt = confusion\_matrix(y\_test,y\_pred)**

**cm\_rf = confusion\_matrix(y\_test,y\_pred)**

**#percentage accuracy score**

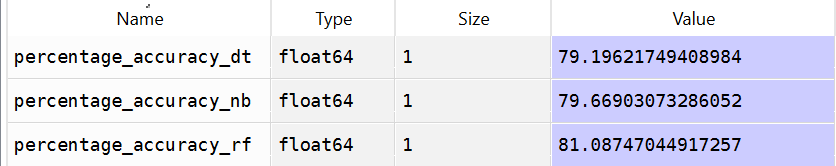
**from sklearn.metrics import accuracy\_score**

**percentage\_accuracy\_nb = accuracy\_score(y\_test,y\_pred\_nb)\*100**

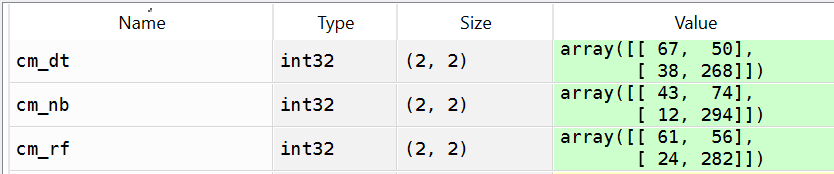
**percentage\_accuracy\_dt = accuracy\_score(y\_test,y\_pred\_dt)\*100**

**percentage\_accuracy\_rf = accuracy\_score(y\_test,y\_pred\_rf)\*100**

**Percentage accuracy**



**Confusion Matrix**



**Chatbots inter-communication**

**Data set description:**

**The data set contains 200 sets of basic question and their replies.**

**Python Script**

**# -\*- coding: utf-8 -\*-**

**"""**

**Created on Sat Jun 2 15:46:50 2018**

**@author: Shubham Banerjee**

**"""**

**import pandas as pd**

**import re**

**import time**

**from sklearn.feature\_extraction.text import TfidfVectorizer**

**from sklearn.metrics.pairwise import cosine\_similarity**

**dataset = pd.read\_csv('chatdata.csv',encoding='latin-1')**

**convo = dataset.iloc[:,1]**

**clist = []**

**def qa\_pairs(x):**

**cpairs = re.findall(": (.\*?)(?:$|\\n)",x)**

**clist.extend(list(zip(cpairs,cpairs[1:])))**

**convo.map(qa\_pairs)**

**convo\_frame = pd.Series(dict(clist)).to\_frame().reset\_index()**

**convo\_frame.columns = ['q','a']**

**vectorizer = TfidfVectorizer(ngram\_range=(1,3))**

**vec = vectorizer.fit\_transform(convo\_frame['q'])**

**class Bots:**

**bot\_count=0**

**creator = 'Master Shubham Banerjee'**

**def \_\_init\_\_(self,name):**

**self.name = name**

**Bots.bot\_count+=1**

**#vectorizer = TfidfVectorizer(ngram\_range=(1,3))**

**#vec = vectorizer.fit\_transform(convo\_frame['q'])**

**self.thisbot = {'what is your name?':self.name,**

**'who is your creator?':Bots.creator,**

**Bots.creator:'Oh! he created me too... I love him'}**

**def introduce(self):**

**return 'Hi, my name is '+str(self.name)+'\n'+Bots.creator+' created me.'**

**def get\_response(self,q):**

**if q in self.thisbot:**

**return thisbot[q]**

**my\_q = vectorizer.transform([q])**

**cs = cosine\_similarity(my\_q, vec)**

**rs = pd.Series(cs[0]).sort\_values(ascending = False)**

**return convo\_frame.iloc[rs.index[0]]['a']**

**sean = Bots('Sean')**

**april = Bots('April')**

**bot=[sean,april]**

**def bot\_chat(bot):**

**while True:**

**i=0**

**msg = bot[i].introduce()**

**while True:**

**print(bot[i%Bots.bot\_count].name+' : '+msg)**

**msg = bot[i%Bots.bot\_count].get\_response(msg)**

**i=i+1**

**time.sleep(3)**

**if i==chat\_count:**

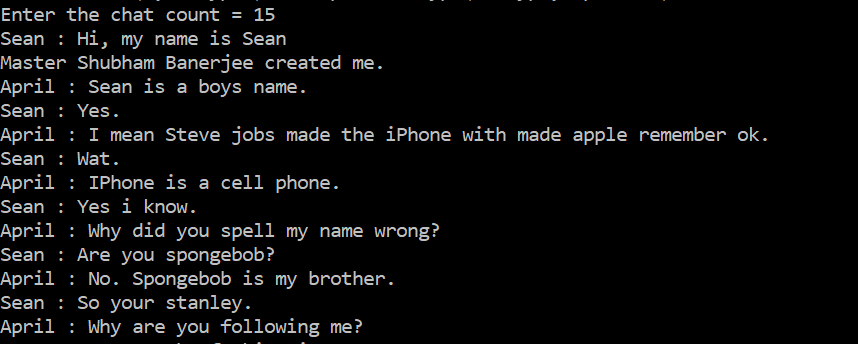
**break**

**break**

**chat\_count = int(input('Enter the chat count = '))**

**bot\_chat(bot)**

**Bots conversation**

****