SHRI VAISHNAV VIDYAPEETH VISHAWAVIDYALAYA

SHRI VAISHNAV INSTITUTE OF INFORMATION TECHNOLOGY

Department of Computer Science and Engineering

Practical File

Introduction to Data Science

BTIBM505

III YEAR V SEM

SECTION – M

Submitted To:
Prof. Omkant Sharma

Submitted By:
Aryak Tomar
20100BTCSDSI07264

Parth Shrivastava

20100BTCSDSI07283

SPAM HAM

- Nowadays, it's likely that everyone knows what Spam means, in the context of e-mail. The use of the word "Ham", on the other hand, is relatively new and sometimes confusing.
- "Ham" is *e-mail that is not Spam*. In other words, "non-spam", or "good mail". It should be considered a shorter, snappier synonym for "non-spam".
- Its usage is particularly common among anti-spam software developers, and not widely known elsewhere; in general, it is probably better to use the term "non-spam", instead.

Why do people send out spam email?

- Often, spam email is sent for commercial purposes. While some people view it as unethical, many businesses still use spam. The cost per email is incredibly low, and businesses can send out mass quantities consistently. Spam email can also be a malicious attempt to gain access to your computer.

How do you stop spam email?

- Spam email can be difficult to stop, as it can be sent from botnets. Botnets are a network of previously infected computers. As a result, the original spammer can be difficult to trace and stop.
- If you receive a message that appears to be spam--for example, if you don’t recognize the sender--mark the message as spam in your email application. Don't click any links or attached files, including opt-out or unsubscribe links. Spammers sometimes include these links to confirm that your email address is legitimate, or the links may trigger malicious webpages or downloads.

Is spam email dangerous?

- Spam email can be dangerous. It can include malicious links that can infect your computer with malware. Do not click links in spam. Dangerous spam emails often sound urgent, so you feel the need to act. Keep reading to learn about some of the basic spam types.

Common types of spam

Commercial advertisements

Whether an email message is spam or a legitimate advertisement, in the United States it's subject to the guidelines in the CAN-SPAM act.

When businesses capture your email address, they often subscribe you to their newsletter by default, as a low-cost way to sell their products. Whenever you fill out an online form, look for a checkbox to opt into or out of marketing email. While these emails can be pesky, most are harmless, and by law they must have a visible opt-out or unsubscribe option.

If you unsubscribe and continue to receive spam, update your email settings to filter messages from the sender's address out of your inbox.

Antivirus warnings

Ironically, antivirus warnings are a common spam tactic. These emails warn you about a computer virus infection and offer a solution--often an antivirus scan--to fix the alleged cyber threat. But taking the bait and clicking the link can grant the hacker access to your system or may download a malicious file.

If you suspect that your computer is infected, do not click a random email link. Instead, pursue legitimate cybersecurity software solutions to protect your endpoints.

Email spoofing

Why are phishing email scams often effective? Because the spam emails masterfully mimic legitimate corporate messages to get you to act. In a spoofing attack a spammer picks a company brand victims will trust, such as a bank or an employer, then uses the company's exact formatting and logos.

Before you reply or click anything, check the From line to make sure that the sender's email address (not just the alias) is legitimate. When in doubt, contact the company to verify whether the email is real.

Sweepstakes winners

Spammers often send emails claiming that you have won a sweepstakes or a prize. They urge you to respond quickly to collect your prize, and may ask you to click a link or submit some personal information. If you don't recognize the competition, or if the email address seems dubious, don't click any links or reply with any personal details.

Money scams

Unfortunately, spammers prey on people's goodwill.The spammer fabricates a story about needing funds for a family emergency or a tragic life event. Some scams, like the Nigerian prince scheme, promise to give you money if you just send your bank account information or pay a small processing fee. Always be cautious about providing personal information or sending money.

What is Ham?

The term ‘ham’ was originally coined by SpamBayes sometime around 2001and is currently defined and understood to be “E-mail that is generally desired and isn't considered spam.”

Desired? You may be saying to yourself “I do not desire this mail, how is this ham and why am I getting it? “The answer is you requested it.

There are two ways you could have signed up for this email.

- **Directly**– While downloading free software such as a browser or a game or signing up for a new online service you were required to agree to and check the box agreeing to their Terms of Service (TOS). Below or above the TOS were other checkboxes. One said “Yes! I would like to receive information and offers from you and your partners.” If you checked this box, then legally you asked for this email.
- **Indirectly**– This is the same scenario as Directly signing up except, the box for the information and offers is pre-checked, leaving it for you to uncheck the box if you do not want to be on their mail lists.

Either way, once you are on a bulk mail list, they can legally send you the offers (and rarely any information worth anything) as long as they follow RFC Regulations.

The good news is that if they follow RFC Rules then it is easy to stop these emails. All you have to do is to simply “click to unsubscribe” and the mail stops. That is if they follow rules.

Malicious spammers especially will take advantage of this and offer the same format at the bottom of their emails linking the unsubscribe link to malicious downloads and/or tracking cookies; Etc…

Email fraud: an INFOSEC case study

This case study looks at the possible consequences of an email scam.

Themes covered include:

- posting personal information online
- poor awareness of spoofing email
- agency protection against spoofing email.

Scenario – what happened

Amy, an agency head from a small government organisation receives an email message from someone she believes is a Ministry of Foreign Affairs (MFAT) colleague.

The colleague’s email address looks genuine because at first glance it features her colleague’s name and his organisation in the usual way, eg, Joe.Smith@mfat.x.co.nz

The email asks Amy to make payments on behalf of the organisation for routine administration expenses and that she click on a weblink, taking her to another website, as part of the payment process.

Amy, being busy and unfamiliar with the capabilities of the SEEmail system that protects agency-to-agency email, clicks the weblink and forwards the payment request to finance.

Finance immediately notes that the email address is not an official government email address (it should have been Joe.Smith@mfat.govt.nz) and confirms that the email sender is, indeed, a fraudster who has obtained the name and organisation of Amy's colleague from his LinkedIn profile.

The weblink Amy clicked on infects her workstation and the entire network with malware, resulting in a widespread compromise of information.

It also presents a threat to the other agencies and contractors that were networked with the agency, undermining official and public confidence in its reliability.

Lessons learned – what should have happened

Amy, Amy's colleague Joe, and her agency made a couple of errors.

Amy's colleague Joe should have:

Posted as little personal information online as possible

Employees working for government, especially those in possession of a national security clearance, should take care to post as little personal information on the internet about themselves as possible as their identity could be fraudulently used to obtain access to information, resources or assets.

Amy should have:

Been more aware of spoofing email and known what to do

Spoofing emails, which may be motivated by financial, criminal or political gain, attempt to appear legitimate by using content, templates and email and web addresses that look very similar to official or legitimate ones. They often contain an active web address directing personnel to a malicious website to either obtain illicit information or infect their work station with malicious code.

Employees should verify the source, be adequately trained to detect and react to malicious or suspicious-looking emails and never send or click active weblinks at work or in official emails.

Amy's agency should have:

Taken steps to avoid spoofing threats

Agencies should configure their email infrastructure to avoid spoofing threats.

MACHINE LEARNING

Machine learning is a field of inquiry devoted to understanding and building methods that 'learn', that is, methods that leverage data to improve performance on some set of tasks. It is seen as a part of artificial intelligence.

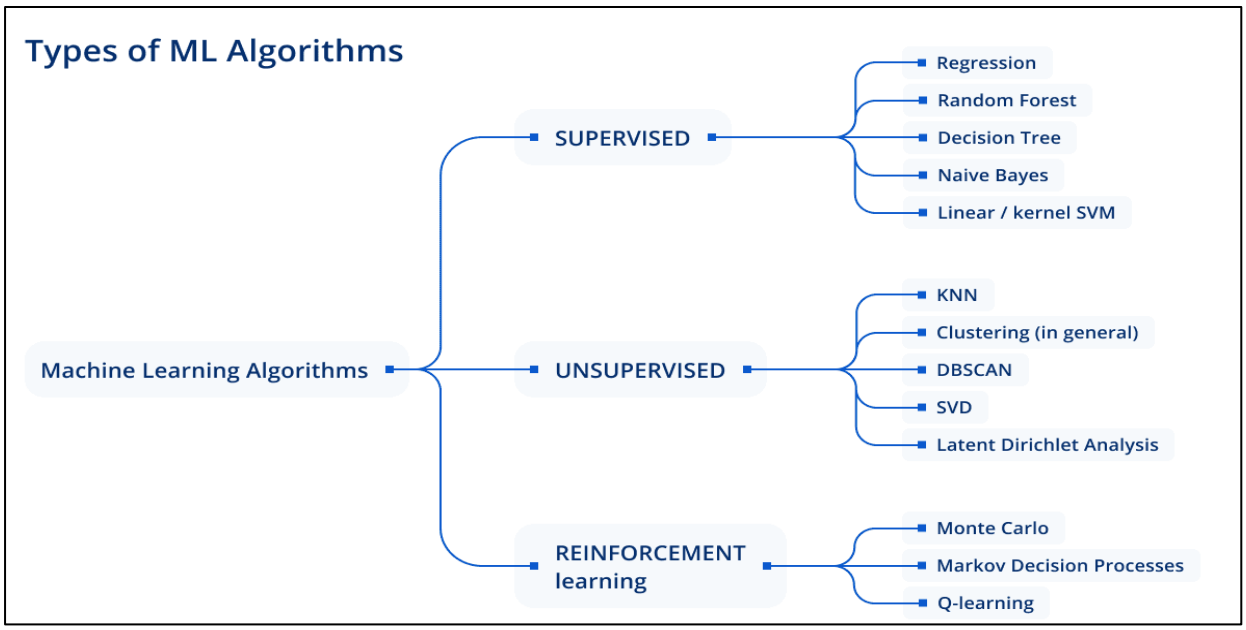
ML Algorithm

A machine learning algorithm is the method by which the AI system conducts its task, generally predicting output values from given input data. The two main processes of machine learning algorithms are classification and regression.

Categorization of ML Algorithms

SUPERVISED LEARNING is a type of Machine learning in which the machine needs external supervision to learn. The supervised learning models are trained using the labeled dataset. Once the training and processing are done, the model is tested by providing a sample test data to check whether it predicts the correct output.

UNSUPERVISED LEARNING is a type of machine learning in which the machine does not need any external supervision to learn from the data, hence called unsupervised learning. The unsupervised models can be trained using the unlabeled dataset that is not classified, nor categorized, and the algorithm needs to act on that data without any supervision.



Data set used from Kaggle - Spam_Ham_Dataset.csv

Basic Libraries used - Pandas, NumPy, Matplotlib, Seaborn, Sklearn

ML Algorithms used –

(i) Logistic Regression

(ii) Decision Tree

(iii) Random Forest

(iv) Decision Tree Classifier

(v) Random Forest Classifier

Libraries In Python –

- Pandas- Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

- NumPy - is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It is open-source software.

- Matplotlib- is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

- Seaborn- Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on the top of matplotlib library and also closely integrated to the data structures from pandas.

- Scikit-learn- (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, *k*-means and DBSCAN.

ML Algorithms Used –

1. Logistic Regression

Logistic regression aims to solve classification problems. It does this by predicting categorical outcomes, unlike linear regression that predicts a continuous outcome. In the simplest case there are two outcomes, which is called binomial, an example of which is predicting if a tumour is malignant or benign.

2. Decision Tree

A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value.

3. Random Forest

Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as bagging.

4. Decision Tree Classifier

DecisionTreeClassifier is a class capable of performing multi-class classification on a dataset. In case that there are multiple classes with the same and highest probability, the classifier will predict the class with the lowest index amongst those classes.

5. Random Forest Classifier

The Random Forest classifier creates a set of decision trees from a randomly selected subset of the training set. It is basically a set of decision trees (DT) from a randomly selected subset of the training set and then It collects the votes from different decision trees to decide the final prediction.

Importing Libraries and Dataset

```
[9]
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
[3]
df = pd.read_csv("spam_ham_dataset.csv")
df
```

...

Now we will perform data analytics on the data set

```
[4]
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5171 entries, 0 to 5170
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Unnamed: 0  5171 non-null   int64 
 1   label       5171 non-null   object
 2   text        5171 non-null   object
 3   label_num   5171 non-null   int64 
dtypes: int64(2), object(2)
memory usage: 161.7+ KB
```

```
[13]
df.shape
```

```
(5171, 4)
```

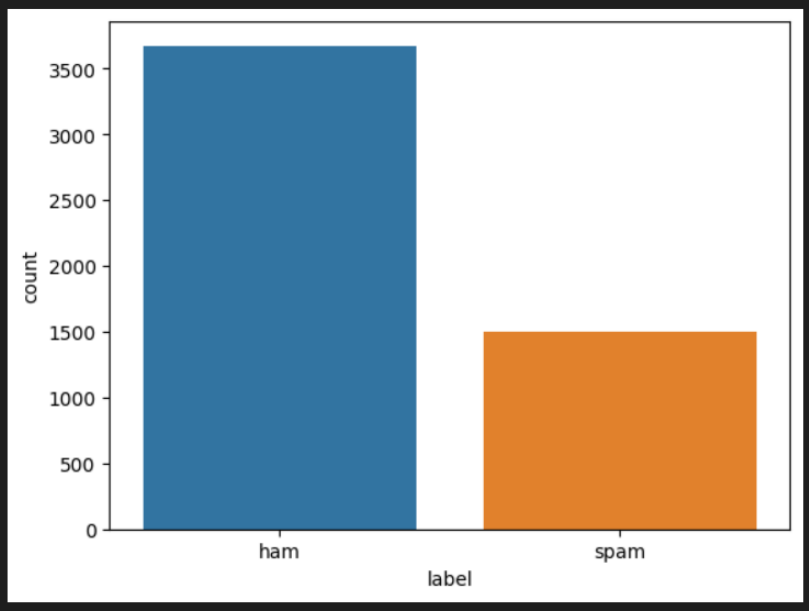
```
[14]
# To calculate the number of unique values in the data set.
df.nunique()
```

```
Unnamed: 0    5171
label            2
text          4993
label_num        2
dtype: int64
```

```
[17]
df["label"].value_counts()
```

```
ham     3672
spam    1499
Name: label, dtype: int64
```

```
[12]
# Here we used count plot for visualizing the amount of spam and ham type emails in the Label column.
sns.countplot(x = df["label"])
plt.show()
```



Model Implementation

```
[34]
# Removing stopping words.
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords

c_message = c_message.split()
print(c_message)
```

```
['subject', 'hpl', 'nom', 'for', 'january', 'see', 'attached', 'file', 'hplnol', 'xls', 'hplnol', 'xls']

[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Parth\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
[39]
# Lemmantization
import nltk as nlp
nltk.download('omw-1.4')
nltk.download('wordnet')
lemma= nlp.WordNetLemmatizer()
#eg.
c_message=[lemma.lemmatize(word) for word in c_message]
c_message= " ".join(c_message)
c_message
```

```
[40]
import string
print(string.punctuation)
```

```
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

```
[41]
str.maketrans('XYZ', 'abc')
```

```
{88: 97, 89: 98, 90: 99}
```

```
[42]
def remove_punctuation(text):
    translator = str.maketrans("","", string.punctuation)
    return text.translate(translator)
```

```
[43]
df["text"] = df["text"].apply(remove_punctuation)
df.head(7)
```

```
[44]
sw = set(stopwords.words('english'))
```

```
[45]
print(sw)
```

```
{'ve', "wasn't", 'why', 'all', 'and', 'in', 'a', 'won', 'but', 'once', 'at', 'into', 'most', 'are', 'some', 'couldn', 'to', 'again', "shan't", 'it', 'do', 'not', 'wouldn', "needn't",
'nor', 'hadn', 'she', 'd', "you've", 'this', "couldn't", 'will', 'of', 'before', 'from', 'needn', 'whom', 'isn', 'them', 'him', 'no', 'on', 'were', 'few', 'than', 'so', 'our',
'shouldn', 'because', 'these', 'he', 'or', 'against', 'their', "isn't", 'doing', 'll', "weren't", 'my', 'herself', 'for', 'shan', 'ain', 'yourselves', "hadn't", 'when', 'how', 'his',
'had', 'himself', 'under', 'myself', 'ourselves', 'have', 'such', "wouldn't", 'which', 'too', 'now', 'her', 'has', "you're", 'its', 'itself', 'wasn', 'you', "you'd", 'by', 'be',
"aren't", 'didn', 'hers', "should've", 'above', 'is', 'off', 'doesn', 'just', 'your', 'more', 'they', 'an', "don't", 'over', 'until', 'that', 'o', 'don', 'out', "doesn't", "you'll",
'does', 'any', 'while', 'can', 'should', 'm', 'those', 'with', 'we', 'where', 'between', 'each', 'then', 'after', 'same', 'here', "mustn't", 'as', 'other', 't', 'further', 'hasn',
'about', 'did', 'during', "mightn't", 'weren', 'mustn', 'if', 'haven', 'having', 'me', 'down', 'been', 'through', 's', 'being', 'own', 'i', 'y', "she's", 'ours', 'both', 'theirs',
'yourself', 'what', "that'll", 'only', 'yours', 'was', "shouldn't", "hasn't", 'there', "didn't", 'the', 'very', 'mightn', 'ma', "won't", 'who', 'themselves', 'am', "haven't", 're',
"it's", 'aren', 'below', 'up'}
```

```
[47]
nltk.download('punkt')
msg_list=[]
for i in df['text']:
    txt=re.sub('[^a-zA-Z]',' ',i)            # remove punctuation
    txt=txt.lower()                          # convert to lower
    txt=nltk.word_tokenize(txt)              # tokenize the words
    txt=[i for i in txt if not i in sw]      # stop words removal
    lemma = nlp.WordNetLemmatizer()
    txt=[lemma.lemmatize(word) for word in txt]
    txt= ' '.join(txt)
    msg_list.append(txt)
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Parth\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
```

Models-

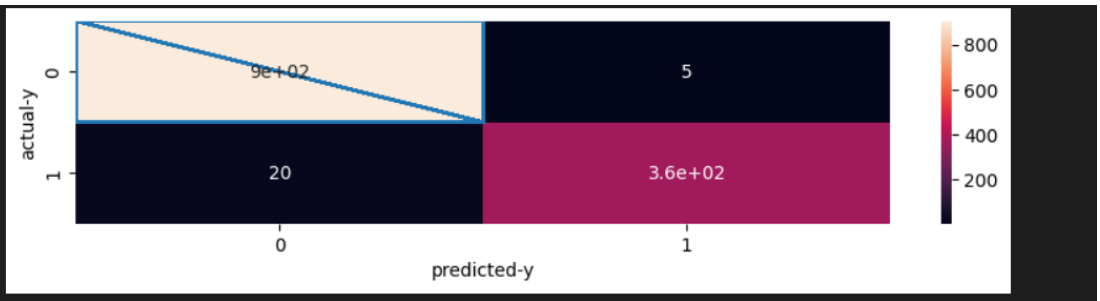
1) Logistic Regression

Logistic Regression

```
[63]
from sklearn.linear_model import LogisticRegression
c=LogisticRegression()
print(c.fit(x_train,y_train.values))
y_pred=c.predict(x_test)
print(f'predicted-y',y_pred[:5],'actual-y',y_test.values[:5])
print(f'predicted-y_shape',y_pred.shape,'actual-y-shape',y_test.values.shape)
```

```
LogisticRegression()
predicted-y [0 1 0 0 1] actual-y [0 1 0 0 1]
predicted-y_shape (1293,) actual-y-shape (1293,)
```

```
[64]
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
cm=confusion_matrix(y_pred,y_test)
plt.figure(figsize=(10,2))
sns.heatmap(cm,annot=True)
plt.plot(y_pred,y_test)
plt.xlabel('predicted-y')
plt.ylabel('actual-y')
plt.show()
```




```
print(classification_report(y_pred,y_test))
print('accuracy-score',accuracy_score(y_pred,y_test))
print('Model score',c.score(x_test,y_test))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       909
           1       0.99      0.95      0.97       384

    accuracy                           0.98      1293
   macro avg       0.98      0.97      0.98      1293
weighted avg       0.98      0.98      0.98      1293

accuracy-score 0.9806651198762568
Model score 0.9806651198762568
```

2) Decision Tree

Decision Tree

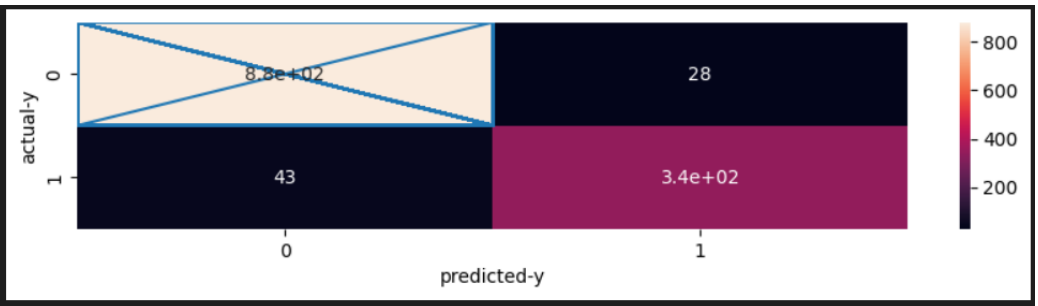
```
from sklearn.tree import DecisionTreeClassifier
c=DecisionTreeClassifier()
print(c.fit(x_train,y_train))
y_pred=c.predict(x_test)
print(f'predicted-y',y_pred[:5],'actual-y',y_test.values[:5])
print(f'predicted-y_shape',y_pred.shape,'actual-y-shape',y_test.values.shape)
```

[66]

```
DecisionTreeClassifier()
predicted-y [0 1 0 0 1] actual-y [0 1 0 0 1]
predicted-y_shape (1293,) actual-y-shape (1293,)
```

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
cm=confusion_matrix(y_pred,y_test)
plt.figure(figsize=(10,2))
sns.heatmap(cm,annot=True)
plt.plot(y_pred,y_test)
plt.xlabel('predicted-y')
plt.ylabel('actual-y')
plt.show()
```

[67]




```
print(classification_report(y_pred,y_test))
print('accuracy-score',accuracy_score(y_pred,y_test))
print('Model score',c.score(x_test,y_test))
```

[68]

```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       909
           1       0.92      0.89      0.91       384

    accuracy                           0.95      1293
   macro avg       0.94      0.93      0.93      1293
weighted avg       0.94      0.95      0.94      1293

accuracy-score 0.9450889404485692
Model score 0.9450889404485692
```

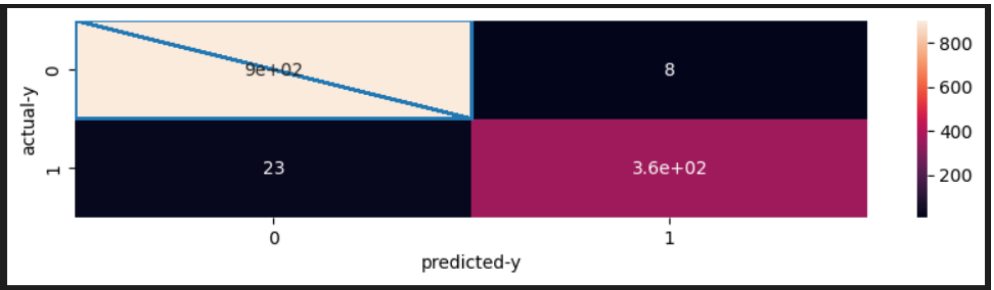
3) Random Forest

Random Forest

```
[69]
from sklearn.ensemble import RandomForestClassifier
c=RandomForestClassifier()
print(c.fit(x_train,y_train))
y_pred=c.predict(x_test)
print(f'predicted-y',y_pred[:5],'actual-y',y_test.values[:5])
print(f'predicted-y_shape',y_pred.shape,'actual-y-shape',y_test.values.shape)
```

```
RandomForestClassifier()
predicted-y [0 1 0 0 1] actual-y [0 1 0 0 1]
predicted-y_shape (1293,) actual-y-shape (1293,)
```

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
cm=confusion_matrix(y_pred,y_test)
plt.figure(figsize=(10,2))
sns.heatmap(cm,annot=True)
plt.plot(y_pred,y_test)
plt.xlabel('predicted-y')
plt.ylabel('actual-y')
plt.show()
```




```
[71]
print(classification_report(y_pred,y_test))
print('accuracy-score',accuracy_score(y_pred,y_test))
print('Model score',c.score(x_test,y_test))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       909
           1       0.98      0.94      0.96       384

    accuracy                           0.98      1293
   macro avg       0.98      0.97      0.97      1293
weighted avg       0.98      0.98      0.98      1293

accuracy-score 0.9760247486465584
Model score 0.9760247486465584
```

4) Decision Tree Classifier

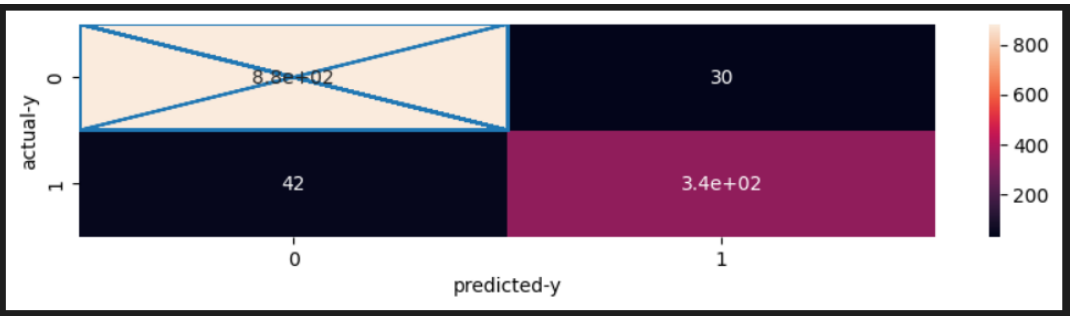
Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
c=DecisionTreeClassifier()
print(c.fit(x_train,y_train))
y_pred=c.predict(x_test)
print(f'predicted-y',y_pred[:5],'actual-y',y_test.values[:5])
print(f'predicted-y_shape',y_pred.shape,'actual-y-shape',y_test.values.shape)
```

[72]

```
DecisionTreeClassifier()
predicted-y [0 1 0 0 1] actual-y [0 1 0 0 1]
predicted-y_shape (1293,) actual-y-shape (1293,)
```

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
cm=confusion_matrix(y_pred,y_test)
plt.figure(figsize=(10,2))
sns.heatmap(cm,annot=True)
plt.plot(y_pred,y_test)
plt.xlabel('predicted-y')
plt.ylabel('actual-y')
plt.show()
```




```
print(classification_report(y_pred,y_test))
print('accuracy-score',accuracy_score(y_pred,y_test))
print('Model score',c.score(x_test,y_test))
```

```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       912
           1       0.92      0.89      0.90       381

    accuracy                           0.94      1293
   macro avg       0.94      0.93      0.93      1293
weighted avg       0.94      0.94      0.94      1293

accuracy-score 0.9443155452436195
Model score 0.9443155452436195
```

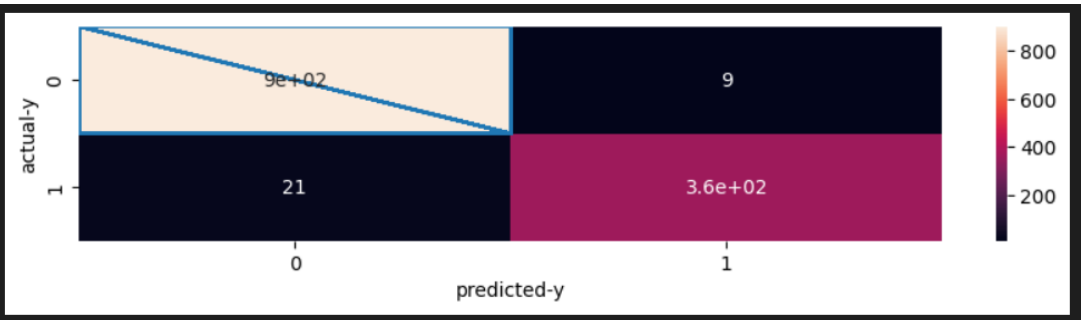
5) Random Forest Classifier

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
c=RandomForestClassifier()
print(c.fit(x_train,y_train))
y_pred=c.predict(x_test)
print(f'predicted-y',y_pred[:5],'actual-y',y_test.values[:5])
print(f'predicted-y_shape',y_pred.shape,'actual-y-shape',y_test.values.shape)
```

```
RandomForestClassifier()
predicted-y [0 1 0 0 1] actual-y [0 1 0 0 1]
predicted-y_shape (1293,) actual-y-shape (1293,)
```

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
cm=confusion_matrix(y_pred,y_test)
plt.figure(figsize=(10,2))
sns.heatmap(cm,annot=True)
plt.plot(y_pred,y_test)
plt.xlabel('predicted-y')
plt.ylabel('actual-y')
plt.show()
```




```
print(classification_report(y_pred,y_test))
print('accuracy-score',accuracy_score(y_pred,y_test))
print('Model score',c.score(x_test,y_test))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       912
           1       0.98      0.94      0.96       381

    accuracy                           0.98      1293
   macro avg       0.98      0.97      0.97      1293
weighted avg       0.98      0.98      0.98      1293

accuracy-score 0.9767981438515081
Model score 0.9767981438515081
```

Accuracy Scores-

Logistic Regression – 0.9806651198762568

Decision Tree – 0.9450889404485692

Random Forest – 0.9760247486465584

Decision Tree Classifier – 0.9443155452436195

Random Forest Classifier – 0.9767981438515081

Conclusion-

After analysing the predictions done by every model and checking the accuracy in different models, we can conclude that the best fit machine learning model for the data set of Spam and Ham is LOGISTIC REGRESSION.

It gave the best model accuracy score of 0.9806651198762568.

Followed by the score of Random Forest Classifier with model accuracy score of 0.9767981438515081.

TEAM MATE 1 - ARYAK TOMAR [20100BTCSDSI07264]

TEAM MATE 2 - PARTH SHRIVASTAVA [20100BTCSDSI07283]

PROJECT TOPIC - SPAM HAM CLASSIFIER

UNDER GUIDENCE OF - PROF. OMKANT SHARMA SIR

Data set used from Kaggle - Spam_Ham_Dataset.csv

Basic Libraries used - Pandasm, Numpy, Matlplotlib, Seaborn, Sklearn

ML Algortihms used -

(i) Logistic Regression

(ii) Decision Tree

(iii) Random Forest

(iv) Decision Tree Classifier

(v) Random Forest Classifier

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
C:\Users\Parth\AppData\Local\Programs\Python\Python310\lib\site-packages\numpy\_distri
butor_init.py:30: UserWarning: loaded more than 1 DLL from .libs:
C:\Users\Parth\AppData\Local\Programs\Python\Python310\lib\site-packages\numpy\.libs\l
ibopenblas.EL2C6PLE4ZYW3ECEVIV3OXXGRN2NRFM2.gfortran-win_amd64.dll
C:\Users\Parth\AppData\Local\Programs\Python\Python310\lib\site-packages\numpy\.libs\l
ibopenblas.FB5AE2TYXYH2IJRDKGDGQ3XBKLKTF43H.gfortran-win_amd64.dll
  warnings.warn("loaded more than 1 DLL from .libs:"
```

In [3]:
```
df = pd.read_csv("spam_ham_dataset.csv")
df
```

Out[3]:

	Unnamed: 0	label	text	label_num
0	605	ham	Subject: enron methanol ; meter # : 988291\r\n...	0
1	2349	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	3624	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	4685	spam	Subject: photoshop , windows , office . cheap ...	1
4	2030	ham	Subject: re : indian springs\r\nthis deal is t...	0
...	...	...	...	...
5166	1518	ham	Subject: put the 10 on the ft\r\nthe transport...	0
5167	404	ham	Subject: 3 / 4 / 2000 and following noms\r\nhp...	0
5168	2933	ham	Subject: calpine daily gas nomination\r\n>\r\n...	0
5169	1409	ham	Subject: industrial worksheets for august 2000...	0
5170	4807	spam	Subject: important online banking alert\r\ndea...	1

5171 rows × 4 columns

FIRST STEP - Data Analytics

Now we will perform data analytics on the data set

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5171 entries, 0 to 5170
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Unnamed: 0  5171 non-null   int64 
 1   label       5171 non-null   object
 2   text        5171 non-null   object
 3   label_num   5171 non-null   int64 
dtypes: int64(2), object(2)
memory usage: 161.7+ KB
```

In [5]:
```
df.describe()
```

Out[5]:

	Unnamed: 0	label_num
count	5171.000000	5171.000000
mean	2585.000000	0.289886
std	1492.883452	0.453753
min	0.000000	0.000000
25%	1292.500000	0.000000
50%	2585.000000	0.000000
75%	3877.500000	1.000000
max	5170.000000	1.000000

In [6]:
```
df.shape
```

Out[6]:
```
(5171, 4)
```

In [7]:
```
# To calculate the number of unique values in the data set.
df.nunique()
```

Out[7]:
```
Unnamed: 0    5171
label            2
text          4993
label_num        2
dtype: int64
```

Getting the total number of values from each columns.

In [8]:
```
df["label"].value_counts()
```

Out[8]:
```
ham     3672
spam    1499
Name: label, dtype: int64
```

In [9]:
```
df["text"].value_counts()
```

Out[9]:
```
Subject: calpine daily gas nomination\r\n>\r\nricky a . archer\r\nfuel supply\r\n700 l
ouisiana , suite 2700\r\nhouston , texas 77002\r\n713 - 830 - 8659 direct\r\n713 - 830
- 8722 fax\r\n- calpine daily gas nomination 1 . doc
20
Subject: \r\n
16
Subject: \r\nthis week only : f . ree gen . erlc vlag . ra\r\ncover the shipping , and
we ' ll send youour product at no cost to prove its\r\neffectiveness .\r\nwon ' t last
. . . hurry\r\nstop promos .\r\n
3
Subject: we ' ve found a school for you !\r\n
3
Subject: you can be smart !\r\n
3


..
Subject: re : driscoll ranch # 3 gas pricing and interconnect estimate\r\ncan you help
me out on this darren ? mjj\r\n- - - - - - - - - - - - - - - - - - - - - - - forwarded b
y mary jo johnson / hou / ect on 11 / 09 / 2000\r\n10 : 04 am - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - -\r\n" john daugherty " on 11 / 08 / 2000 04 : 38 : 37 pm
\r\nto :\r\ncc :\r\nsubject : re : driscoll ranch # 3 gas pricing and interconnect est
imate\r\nmary jo ,\r\nthanks for the update . regarding the notice provision of 6 busi
ness days\r\nprior to the close of business on the last business day of the month prio
r\r\nto selected month , does that mean we need to give you notice for december\r\nby
tuesday , november 21 st at 5 : 00 pm or monday , november 20 th at 5 : 00 pm\r\nassum
ing the 23 rd and 24 th are holidays ?\r\njohn daugherty\r\n- - - - - original message
- - - - -\r\nfrom :\r\nto :\r\ncc : ; ;\r\n; ;\r\n; ;\r\n;\r\nsent : wednesday , novem
ber 08 , 2000 5 : 12 pm\r\nsubject : re : driscoll ranch # 3 gas pricing and interconn
ect estimate
1
Subject: jordyn , there is nothing like a dream to create the future .\r\nfor the sake
of one good action a hundred evil ones should be forgotten\r\nto accomplish more , red
irect your mental energy by continuously reminding yourself of all the things you do r
ight\r\n
1
Subject: from raymond bowen , jr . , exec . v . p . , finance & treasurer\r\nto : all
enron employees\r\nfrom : raymond bowen , jr .\r\nevp , finance & treasurer\r\nsubject
: update on employee expense reimbursement\r\npost - petition expenses\r\neffective im
mediately , the processing of expense reimbursement for business expenses incurred by
employees after enron corp . ' s chapter 11 filing ( after december 2 nd ) has returne
d to normal . expenses will be reimbursed promptly upon submission . employees should
conduct business travel in the ordinary course of business as approved by direct super
visors . new travel and entertainment guidelines regarding business travel and other e
mployee expenses will be forthcoming . in the meantime , please use discretion as you
incur expenses . all expenditures should be made with consideration given to enron ' s
current financial situation .\r\npre - petition expenses\r\nwe will immediately begin
to reimburse pre - petition expenses for current and former employees ( incurred prior
to the december 2 nd chapter 11 filing ) up to a maximum of $ 5 , 000 . we understand
that a number of employees have pre - petition expenses in excess of $ 5 , 000 . howev
er , under the first day orders of the bankruptcy court , there is a $ 15 , 000 allowa
nce per employee for unpaid pre - petition compensation , benefits , and related emplo
yee expenses . this $ 15 , 000 cap was imposed without input from enron management and
counsel . we have sought clarification from the bankruptcy court to assure that this $
15 , 000 cap does not include the reimbursement of medical , dental , or vision relate
d expenses .\r\nwe have commenced a manual review of every individual with pending pre
- petition expenses in excess of $ 5 , 000 . even prior to clarification from the cour
t regarding the medical / dental / vision expense issue , we will begin to reimburse p
re - petition expenses in excess of $ 5 , 000 and up to $ 15 , 000 as soon as we can v
erify that the aggregate reimbursements to an individual employee have not exceeded th
e $ 15 , 000 cap . unfortunately , since this will be a manual process , it will take
```

```
some time to work through . employees with pre - petition business expenses in excess
of $ 5 , 000 can assist us with this manual review process by providing a brief email
summary of their situation , including the total amount of pending pre - petition medi
cal / dental / vision claims , to expensereport @ enron . com .\r\nthank you for your
patience .     1
Subject: re : indutrial report\r\nrobert ,\r\nthis is the file that i referenced in my
last email . please get this file\r\ngoing again . thanks , pat\r\ndaren j farmer @ ec
t\r\n02 / 25 / 2000 04 : 52 pm\r\nto : robert e lloyd / hou / ect @ ect\r\ncc : pat cl
ynes / corp / enron @ enron\r\nsubject : indutrial report\r\nrobert ,\r\nken developed
an industrial report before he left . it can be found at\r\no / logistics / kenseaman
/ industrialsmonthly / . . . there is one file for each\r\nmonth of 2000 . i need you
to update this for march . this will need to be\r\ndistributed to gas control , logist
ics , and myself . let me know if you have\r\nany questions .\r\nd
1
Subject: important online banking alert\r\ndear valued citizensr bank member ,\r\ndue
to concerns , for the safety and integrity of the online banking community we have iss
ued the following warning message .\r\nit has come to our attention that your citizens
r bank account information needs to be updated as part of our continuing commitment to
protect your account and to reduce the instance of fraud on our website . if you could
please take 5 - 10 minutes out of your online experience and renew your records you wi
ll not run into any future problems with the online service . however , failure to con
firm your records may result in your account suspension .\r\nonce you have confirmed y
our account records your internet banking service will not be interrupted and will con
tinue as normal .\r\nto confirm your bank account records please click here .\r\nnote
:\r\nthis e - mail was sent on behalf of the online banking community , if you do not
have an online banking account with charterr one then this message does not apply to y
ou and you may ignore this message .\r\nthank you for your time ,\r\ncitizensr financi
al group .\r\n
1
Name: text, Length: 4993, dtype: int64
```

In [10]:
```
df.groupby("label").count()
```

Out[10]:

label	Unnamed: 0	text	label_num
ham	3672	3672	3672
spam	1499	1499	1499

In [11]:
```
df.groupby("text").count()
```

Out[11]:

Subject: \r\n(envelope - from 20040929124340 . cca 972659112757918286 . 39382 qi @ earthlink . co
com\r\nnotification : no\r\nreply - to : matilda cox qi @ earthlink . com\r\ndate : thu , 10 feb 2005 03 : 13 : 03 + 02
demokritos . gr\r\nsubject : paliourg\r\nmime - version : 1 . 0\r\ncontent - type : text / html ; charset = us - ascii\r\n
/ www . accucast . com)\r\nhtml\r\nbrbr\r\nmicros 0 ft for pennies\r\nbrbr\r\na href = http : / / degassing . jnalf
waddle duress hate gun salem buckboard sarasota blustery transposable willowy wadi appointe philolog
insensitive cranny gnu polio incommensurable

Subject: \r\n2\r\nx - message - info :\r\ns 63 h 5 ja 567 vssd 2 p 26 n 61 wl 67 hzvlnqxjn\r\nreceived : from mimbdz
with\r\nmicrosoft smtpsvc (5 . 0 . 2195 . 6824) ;\r\ntue , 23 mar 2004 00 : 38 : 43 - 0100\r\nreceived : from characte
with smtp\r\nid 619255 c 738 ox\r\n(authid : mitchelforeman) ;\r\nmon , 22 mar 2004 23 : 37 : 43 - 0200\r\nfrom :
paliourg @ iit . demokritos . gr\r\n2\r\nsubject : [spam ?] fwd : doctors not needed . secure shopping . vcodln . v
19 : 39 : 43 - 0600\r\nmessage - id :\r\nmime - version : 1 . 0\r\ncontent - type : mult
20807747445880214 97\r\ncontent - type : text / html ;\r\ncontent - transfer - encoding : quoted - printable\r\n! doct
/ www =\r\n. w 3 . org / tr / html 4 / loose . dtd\r\nhtml\r\nhead\r\ntitleall the medications you will ever need / title\r\r
src = 3 dhttp : / / www . populardrug =\r\n

Subject: \r\n80 %\r\n?????????\r\n???\r\n?????? ?????????\r\n????????\r\n????????\r\n??? ?????? ???
\r\n??????\

Subject: \r\n9\r\nreceived : from 55 . 240 . 132 . 191 by 217 . 129 . 64 . 201 ; tue , 23 mar 2004 20 : 33 : 35 - 0400\r\n
rogers . com\r\nreply - to : rhonda madrid mklhbi @ rogers . com\r\nto : vangelis @ iit . demokritos . gr\r\ncc : pa
@ iit . demokritos . gr\r\n8\r\nsubject : [spam ?] fwd : better approved than hospitals . on - time delivery . v : @ g
mar 2004 05 : 33 : 35 + 0500\r\nx - mailer : aol 5 . 0 for windows us sub 510\r\nmime - versio
5482930460129096\r\nx - priority : 5\r\nx - msmail - priority : low\r\nx - ip : 22 . 168 . 158 . 193\r\n- - - - 5482
encoding : quoted - printable\r\n! doctype html public - / / w 3 c / / dtd html 4 . 01 transitional / / en http : / / www =
medications you will ever need / title\r\n/ head\r\nbody\r\na href = 3 dhttp : / / www . populardrugs . bizimg src = 3

Subject: your username and password\r\ndear daren farmer ,\r\nhere is your username and password that you re

Subject: your women will be happy !\r\nhow does viagra professional work ?\r\nviagra is a prescription drug used
. since it first became available in 1998 , the prescription drug viagra\r\nhas helped about 16 million men aro
dispensed every second worldwide . and no other therapy or prescription\r\ndrug for erectile dysfunction ha
experience as viagra since its launch . it ' s no wonder more than 600 , 000 doctors\r\nhave chosen th
taken\r\ncorrectly , viagra works for most men . studies show that it works for up to 4 out of 5 men\r\n(versus 1 c
matter how long\r\nthey have had ed , what caused it , how often they have it , or how old they are .\r\nif
unopened pack . all you have to do is send them back , and we will immediately refun

Subject: yvette ooto\r\ni will be on vacation monday , april 17 th thru friday , april 21 st . elizabeth\r\nsoto wil
the\r\nremainder of the week .\r\nif you have any questions regarding brenda ' s calendar , need to make change
villanueva at\r\next 3 . 6279 .\r\nif you need assistance with

Subject: zdrive 1 . 5 gb usb 2 . 0 portable storage @ 138. 00\r\nzdrive1. 5gbusb2. 0portable\r\nstorage\r\n
storage capacity - 4 mbps media\r\ntransfer rate - 40 mbps transfer speed (read / write) - usb plug\r\nrotates 90
macintosh compatible - small , portable\r\ndesign fits in your pocket\r\nthe 1 . 5 gb zdrive is a\r\nhigh - capacity
customers to carry 30 cds worth of mp 3\r\nmusic , two hours of vhs - quality mpeg digital video , 650 4 megapixe
of\r\ninformation ! users can now easily carry data wherever they go and\r\nplug the storage device into any
bandwidth for multimedia and\r\nstorage applications .\r\nvisit : http : / / www . computron - me . com for deals
www . computron - me . com\r\nfor latest clearance sale listing contact our\r\nsales department .\r\nfor further
aeor contact
com\r\ndell\r\nintel\r\niomega\r\nepson\r\naopen\r\ncreative\r\ntoshiba\r\napc\r\ncisco\r\nus\r\nrobotics\r\nmicros
- - - - - - and lots more\r\n! ! !\r\nif you have any\r\ncomplaints / suggestions contact : customerservice @ c
dollars , ex - works ,\r\nfax + 971 4\r\n8834454\r\njebel ali duty free zone\r\nwww . computron - me . com\r\n
.\r\nwithout\r\nnotice .\r\nto receive our special offers\r\nin plain\r\ntext format reply to this\r\nmail with the
spam as long as we include : contact\r\ninformation remove instructions . this message is intended for deal
error , or\r\nfor any other reason would like to be removed , please reply with " remove\r\n" in the subject line of
with the federal legislation for commercial e - mail\r\n(h . r . 4176 - section 101 paragraph (e) (1) (a) and bill s
and\r\ntrademarks are the property of their respective\r\nowners\r\nproducts may not be exactly as shown\r\nabo
:\r

	Subject: zero path termination in path manager\r\nmatt / ben :\r\ni am responding to your request for comments c termination process to include zero volume paths on the last day of the month as a way to speed up unify respon including aep and calgary and our recent discussions , we find a legitimate business need to allow zero volu wellhead gas) . following is a recap of what is proposed to be done regarding zero path termination .\r\n1 . all pat below) will terminate on that day and will no longer roll into the following month .\r\nthis path termination out fun will zero out future month ' s paths unless a volume\r\nis on that path in the future month .\r\n2 . zero paths mark to month until changed to some other nomination option . it appears that hpl\r\n(736 paths) and nigas (11 special one time handling of these paths will be coordinated with the\r\nscheduler when you are ready .\r\ management - do not send " function will cease to do so . this was agree to by patti in regards to\r\npat

4993 rows × 3 columns

In [12]:
```
df.columns
```

Out[12]:
```
Index(['Unnamed: 0', 'label', 'text', 'label_num'], dtype='object')
```

Splitting the original data set into the desired columns for further processing.

In [13]:
```
new_df = df[["label", "text"]]
new_df
```

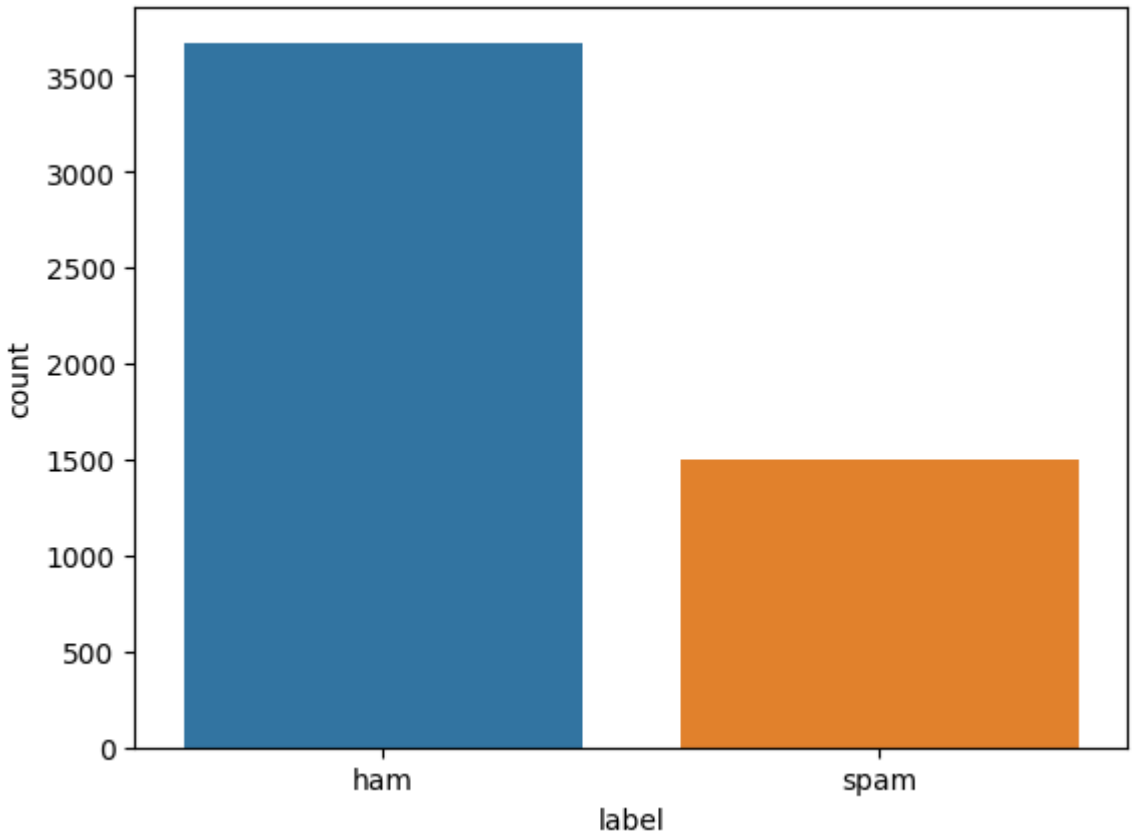
Out[13]:

	label	text
0	ham	Subject: enron methanol ; meter # : 988291\r\n...
1	ham	Subject: hpl nom for january 9 , 2001\r\n(see...
2	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...
3	spam	Subject: photoshop , windows , office . cheap ...
4	ham	Subject: re : indian springs\r\nthis deal is t...
...	...	...
5166	ham	Subject: put the 10 on the ft\r\nthe transport...
5167	ham	Subject: 3 / 4 / 2000 and following noms\r\nhp...
5168	ham	Subject: calpine daily gas nomination\r\n>\r\n...
5169	ham	Subject: industrial worksheets for august 2000...
5170	spam	Subject: important online banking alert\r\ndea...

5171 rows × 2 columns

Visualization of the data after data analytics.

In [14]:
```
sns.countplot(x = df["label"])
plt.show()
```



Importing the necessary libraries and modules for predicition.

In [15]:
```
from sklearn.preprocessing import LabelEncoder
lb = LabelEncoder()
df["label"] = lb.fit_transform(df["label"])
df["label"][:5]
```

Out[15]:
```
0    0
1    0
2    0
3    1
4    0
Name: label, dtype: int32
```

In [16]:
```
# Checkinf for NULL values.
df.isnull().sum()
```

Out[16]:
```
Unnamed: 0    0
label         0
text          0
label_num     0
dtype: int64
```

Removing unwanted elements from the data set.

In [17]:
```
# unwanted elements like symbols, numbers etc, so for removing them we use tokenizer.
from nltk.tokenize import RegexpTokenizer
tn = RegexpTokenizer(r'[a-zA-Z0-9]')
```

In [18]:
```
# Now for converting the string to lower case we wil use sub method of re library.
import re
sample_message = df["text"][1]
print(sample_message)
c_message = re.sub('[^a-zA-Z]', ' ', sample_message)
c_message = c_message.lower()
c_message
```

```
Subject: hpl nom for january 9 , 2001
( see attached file : hplnol 09 . xls )
- hplnol 09 . xls
```

Out[18]:
```
'subject  hpl nom for january             see attached file   hplnol      xls      hpl
nol      xls'
```

Removing stopping words.

In [19]:
```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords

c_message = c_message.split()
print(c_message)
```

```
['subject', 'hpl', 'nom', 'for', 'january', 'see', 'attached', 'file', 'hplnol', 'xl
s', 'hplnol', 'xls']

[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Parth\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Performing Lemmantization for grouping together of different forms of the same word.

In [20]:
```
# Lemmantization
import nltk as nlp
nltk.download('omw-1.4')
nltk.download('wordnet')
lemma= nlp.WordNetLemmatizer()
#eg.
c_message=[lemma.lemmatize(word) for word in c_message]
c_message= " ".join(c_message)
c_message
```

```
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\Parth\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Parth\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[20]:
```
'subject hpl nom for january see attached file hplnol xl hplnol xl'
```

In [21]:
```
import string
print(string.punctuation)
```

```
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

In [22]:
```
str.maketrans('XYZ', 'abc')
```

Out[22]:
```
{88: 97, 89: 98, 90: 99}
```

Removing punctuations for filtering the data.

In [23]:
```
def remove_punctuation(text):
    translator = str.maketrans("","", string.punctuation)
    return text.translate(translator)
```

In [24]:
```
df["text"] = df["text"].apply(remove_punctuation)
df.head(7)
```

Out[24]:

	Unnamed: 0	label	text	label_num
0	605	0	Subject enron methanol meter 988291\r\nthis...	0
1	2349	0	Subject hpl nom for january 9 2001\r\n see at...	0
2	3624	0	Subject neon retreat\r\nho ho ho we re aroun...	0
3	4685	1	Subject photoshop windows office cheap mai...	1
4	2030	0	Subject re indian springs\r\nthis deal is to ...	0
5	2949	0	Subject ehronline web address change\r\nthis m...	0
6	2793	0	Subject spring savings certificate take 30 o...	0

In [25]:
```
sw = set(stopwords.words('english'))
```

In [26]:
```
print(sw)
```

```
{'few', 'have', 'down', 'it', 'up', 'can', 'why', 'our', 'should', 'this', 'theirs',
'when', "mustn't", 'do', 'more', 'as', 'to', "you've", 'until', "couldn't", 'here', 'm
y', 'some', 'did', 'into', 'does', 'above', "wasn't", 'doing', 'other', 'once', "tha
t'll", 'most', 'having', "don't", 've', 'you', 'who', 'yourself', 'only', 'doesn', 'mu
stn', 'so', 'ma', 'that', 'while', "she's", 'in', 'were', 'these', 'being', 'just', "s
houldn't", "won't", 'hasn', 'through', 'had', 'haven', 'weren', 'didn', 'now', 'but',
'himself', 'y', 'then', 'own', 'him', 'than', 'between', 'he', 'them', 'after', 'is',
"needn't", 'before', 'the', "isn't", "you're", 'themselves', 'on', 'not', 'are', 'her
s', "it's", 'an', 'shan', 'further', 'all', 'will', 'needn', 'd', 'am', 're', "sha
n't", 's', 'she', 'below', 'any', 'ourselves', 'ours', "hasn't", 'very', 'her', 'i',
'from', 'be', 'at', "weren't", 'there', "mightn't", "wouldn't", 'its', 'wasn', 'abou
t', 'nor', 'those', 'o', "should've", "haven't", 'wouldn', 'for', 'hadn', 'me', 'you
r', 'or', 'out', 'what', "didn't", 'off', 'of', 'over', 'aren', 'if', 'his', 'during',
'whom', 'because', 'shouldn', 'with', 'ain', 'couldn', 'both', 'mightn', 'yours', 'do
n', 'where', 'itself', 'yourselves', 'myself', 't', 'their', 'm', 'a', 'again', 'll',
"aren't", 'has', 'each', 'too', 'was', 'isn', "you'll", 'we', 'such', 'against', "yo
u'd", 'no', 'same', 'and', 'how', 'herself', 'they', 'under', 'won', "doesn't", 'bee
n', 'which', "hadn't", 'by'}
```

In [27]:
```
nltk.download('punkt')
msg_list=[]
for i in df['text']:
    txt=re.sub('[^a-zA-Z]',' ',i)            # remove punctuation
    txt=txt.lower()                          # convert to lower
    txt=nltk.word_tokenize(txt)              # tokenize the words
    txt=[i for i in txt if not i in sw]      # stop words removal
    lemma = nlp.WordNetLemmatizer()
    txt=[lemma.lemmatize(word) for word in txt]
    txt= ' '.join(txt)
    msg_list.append(txt)
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Parth\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

In [28]:
```
msg_list
```

Out[28]:
```
['subject enron methanol meter follow note gave monday preliminary flow data provid
ed daren please override pop daily volume presently zero reflect daily activity obt
ain gas control change needed asap economics purpose',
 'subject hpl nom january see attached file hplnol xl hplnol xl',
 'subject neon retreat ho ho ho around wonderful time year neon leader retreat time
know time year extremely hectic tough think anything past holiday life go past week
december january like think minute calender handed beginning fall semester retreat
scheduled weekend january youth minister conference brad dustin connected week goin
g change date following weekend january come part need think think agree important
u get together time recharge battery get far spring semester lot trouble difficult
u get away without kid etc brad came potential alternative get together weekend let
know prefer first option would retreat similar done past several year year could go
heartland country inn www com outside brenham nice place bedroom bedroom house side
side country real relaxing also close brenham one hour minute golf shop antique cra
ft store brenham eat dinner together ranch spend time meet saturday return sunday m
orning like done past second option would stay houston dinner together nice restaur
ant dessert time visiting recharging one home saturday evening might easier trade w
ould much time together let decide email back would preference course available wee
kend democratic process prevail majority vote rule let hear soon possible preferabl
```

In [29]:
```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer()
```

In [30]:
```
# Converting the list to array.
x_cv = cv.fit_transform(msg_list)
x_cv
```

Out[30]:
```
<5171x43198 sparse matrix of type '<class 'numpy.int64'>'
        with 313566 stored elements in Compressed Sparse Row format>
```

In [31]:
```
x_cv.toarray()
```

Out[31]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

Splitting the data set into train and test .

In [32]:
```
# Splitting the data set into train and test.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_cv, df['label'], test_size = .25)
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Out[32]:
```
((3878, 43198), (1293, 43198), (3878,), (1293,))
```

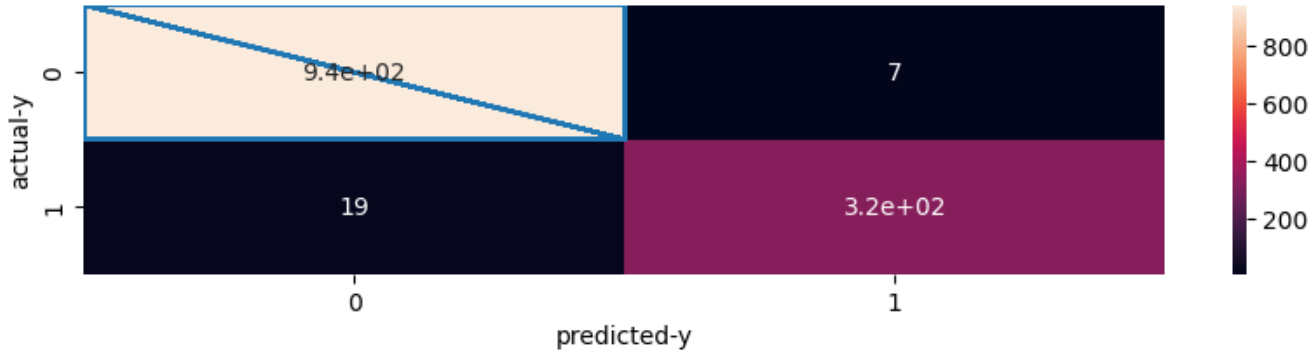
STEP 2 - Implementing Models for best predictions.

Logistic Regression

In [33]:
```
from sklearn.linear_model import LogisticRegression
c=LogisticRegression()
print(c.fit(x_train,y_train.values))
y_pred=c.predict(x_test)
print(f'predicted-y',y_pred[:5],'actual-y',y_test.values[:5])
print(f'predicted-y_shape',y_pred.shape,'actual-y-shape',y_test.values.shape)
```

```
LogisticRegression()
predicted-y [0 0 0 0 0] actual-y [0 0 0 0 0]
predicted-y_shape (1293,) actual-y-shape (1293,)
```

In [34]:
```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
cm=confusion_matrix(y_pred,y_test)
plt.figure(figsize=(10,2))
sns.heatmap(cm,annot=True)
plt.plot(y_pred,y_test)
plt.xlabel('predicted-y')
plt.ylabel('actual-y')
plt.show()
```



In [35]:
```
print(classification_report(y_pred,y_test))
print('accuracy-score',accuracy_score(y_pred,y_test))
print('Model score',c.score(x_test,y_test))
```

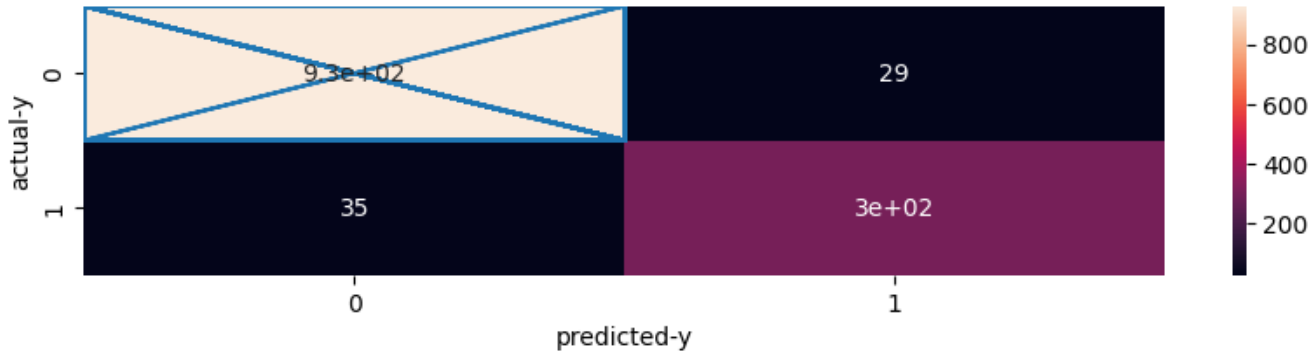
```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       950
           1       0.98      0.94      0.96       343

    accuracy                           0.98      1293
   macro avg       0.98      0.97      0.97      1293
weighted avg       0.98      0.98      0.98      1293

accuracy-score 0.979891724671307
Model score 0.979891724671307
```

Decision Tree

In [36]:
```
from sklearn.tree import DecisionTreeClassifier
c=DecisionTreeClassifier()
print(c.fit(x_train,y_train))
y_pred=c.predict(x_test)
print(f'predicted-y',y_pred[:5],'actual-y',y_test.values[:5])
print(f'predicted-y_shape',y_pred.shape,'actual-y-shape',y_test.values.shape)
```

```
DecisionTreeClassifier()
predicted-y [0 0 0 0 0] actual-y [0 0 0 0 0]
predicted-y_shape (1293,) actual-y-shape (1293,)
```

In [37]:
```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
cm=confusion_matrix(y_pred,y_test)
plt.figure(figsize=(10,2))
sns.heatmap(cm,annot=True)
plt.plot(y_pred,y_test)
plt.xlabel('predicted-y')
plt.ylabel('actual-y')
plt.show()
```



In [38]:
```
print(classification_report(y_pred,y_test))
print('accuracy-score',accuracy_score(y_pred,y_test))
print('Model score',c.score(x_test,y_test))
```

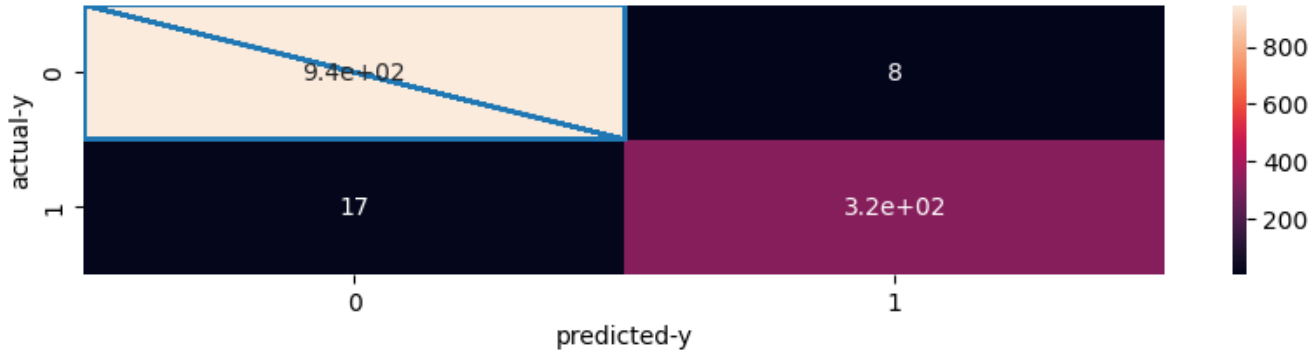
```
              precision    recall  f1-score   support

           0       0.96      0.97      0.97       956
           1       0.91      0.90      0.90       337

    accuracy                           0.95      1293
   macro avg       0.94      0.93      0.94      1293
weighted avg       0.95      0.95      0.95      1293

accuracy-score 0.9505027068832174
Model score 0.9505027068832174
```

Random Forest

In [39]:
```
from sklearn.ensemble import RandomForestClassifier
c=RandomForestClassifier()
print(c.fit(x_train,y_train))
y_pred=c.predict(x_test)
print(f'predicted-y',y_pred[:5],'actual-y',y_test.values[:5])
print(f'predicted-y_shape',y_pred.shape,'actual-y-shape',y_test.values.shape)
```

```
RandomForestClassifier()
predicted-y [0 0 0 0 0] actual-y [0 0 0 0 0]
predicted-y_shape (1293,) actual-y-shape (1293,)
```

In [40]:
```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
cm=confusion_matrix(y_pred,y_test)
plt.figure(figsize=(10,2))
sns.heatmap(cm,annot=True)
plt.plot(y_pred,y_test)
plt.xlabel('predicted-y')
plt.ylabel('actual-y')
plt.show()
```



In [41]:
```
print(classification_report(y_pred,y_test))
print('accuracy-score',accuracy_score(y_pred,y_test))
print('Model score',c.score(x_test,y_test))
```

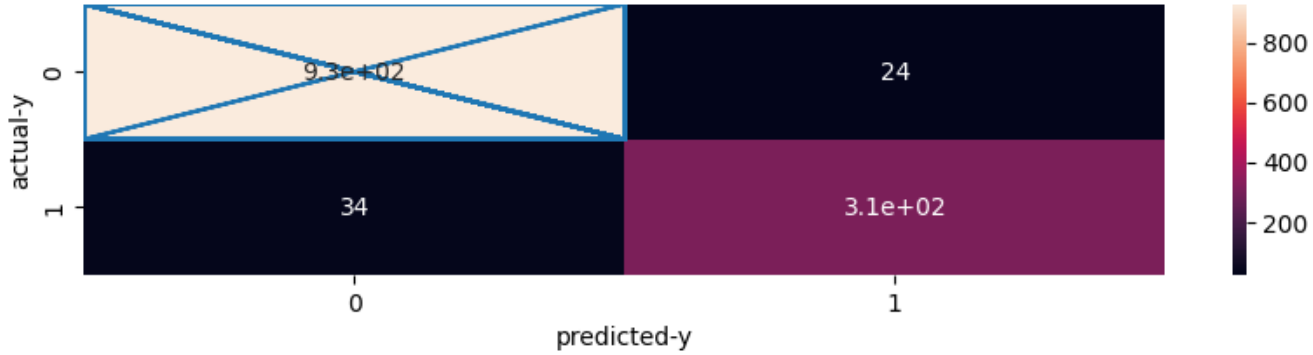
```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       953
           1       0.98      0.95      0.96       340

    accuracy                           0.98      1293
   macro avg       0.98      0.97      0.97      1293
weighted avg       0.98      0.98      0.98      1293

accuracy-score 0.9806651198762568
Model score 0.9806651198762568
```

Decision Tree Classifier

In [42]:
```
from sklearn.tree import DecisionTreeClassifier
c=DecisionTreeClassifier()
print(c.fit(x_train,y_train))
y_pred=c.predict(x_test)
print(f'predicted-y',y_pred[:5],'actual-y',y_test.values[:5])
print(f'predicted-y_shape',y_pred.shape,'actual-y-shape',y_test.values.shape)
```

```
DecisionTreeClassifier()
predicted-y [0 0 0 0 0] actual-y [0 0 0 0 0]
predicted-y_shape (1293,) actual-y-shape (1293,)
```

In [43]:
```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
cm=confusion_matrix(y_pred,y_test)
plt.figure(figsize=(10,2))
sns.heatmap(cm,annot=True)
plt.plot(y_pred,y_test)
plt.xlabel('predicted-y')
plt.ylabel('actual-y')
plt.show()
```



In [44]:
```
print(classification_report(y_pred,y_test))
print('accuracy-score',accuracy_score(y_pred,y_test))
print('Model score',c.score(x_test,y_test))
```

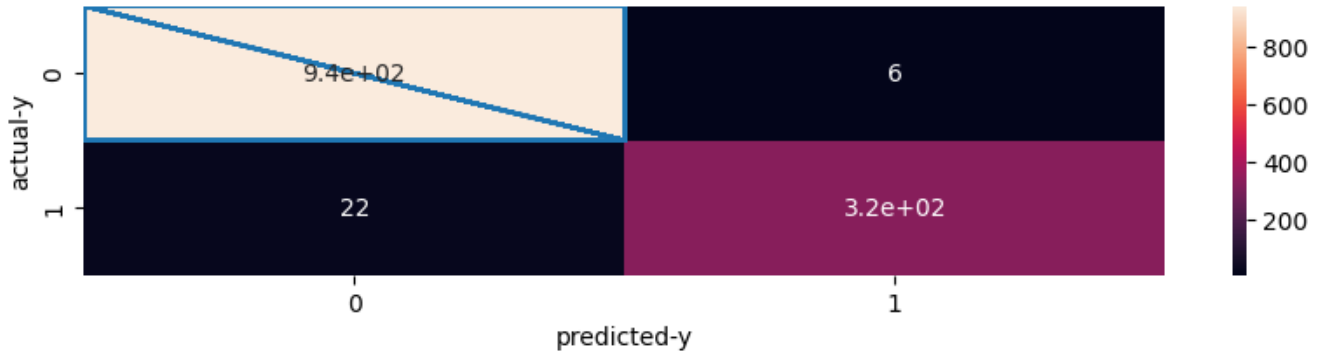
```
              precision    recall  f1-score   support

           0       0.96      0.97      0.97       952
           1       0.93      0.90      0.91       341

    accuracy                           0.96      1293
   macro avg       0.95      0.94      0.94      1293
weighted avg       0.95      0.96      0.95      1293

accuracy-score 0.9551430781129157
Model score 0.9551430781129157
```

Random Forest Classifier

In [45]:
```
from sklearn.ensemble import RandomForestClassifier
c=RandomForestClassifier()
print(c.fit(x_train,y_train))
y_pred=c.predict(x_test)
print(f'predicted-y',y_pred[:5],'actual-y',y_test.values[:5])
print(f'predicted-y_shape',y_pred.shape,'actual-y-shape',y_test.values.shape)
```

```
RandomForestClassifier()
predicted-y [0 0 0 0 0] actual-y [0 0 0 0 0]
predicted-y_shape (1293,) actual-y-shape (1293,)
```

In [46]:
```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
cm=confusion_matrix(y_pred,y_test)
plt.figure(figsize=(10,2))
sns.heatmap(cm,annot=True)
plt.plot(y_pred,y_test)
plt.xlabel('predicted-y')
plt.ylabel('actual-y')
plt.show()
```



In [47]:
```
print(classification_report(y_pred,y_test))
print('accuracy-score',accuracy_score(y_pred,y_test))
print('Model score',c.score(x_test,y_test))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       946
           1       0.98      0.94      0.96       347

    accuracy                           0.98      1293
   macro avg       0.98      0.97      0.97      1293
weighted avg       0.98      0.98      0.98      1293

accuracy-score 0.9783449342614076
Model score 0.9783449342614076
```

=> After checking accuracy in different models, atlast we can conclude that the best fit model for this data set is Logisitic Regression.

In [ ]:
```
```

SHRI VAISHNAV VIDYAPEETH VISHAWAVIDYALAYA

Introduction to Data Science

BTIBM505

III YEAR V SEM
SECTION – M

Submitted To:
Prof. Omkant Sharma

Submitted By:
Aryak Tomar (20100BTCSDSI07264)
Parth Shrivastava (20100BTCSDSI07283)

SPAM HAM

MACHINE LEARNING

- Machine learning is a field of inquiry devoted to understanding and building methods that 'learn', that is, methods that leverage data to improve performance on some set of tasks. It is seen as a part of artificial intelligence.

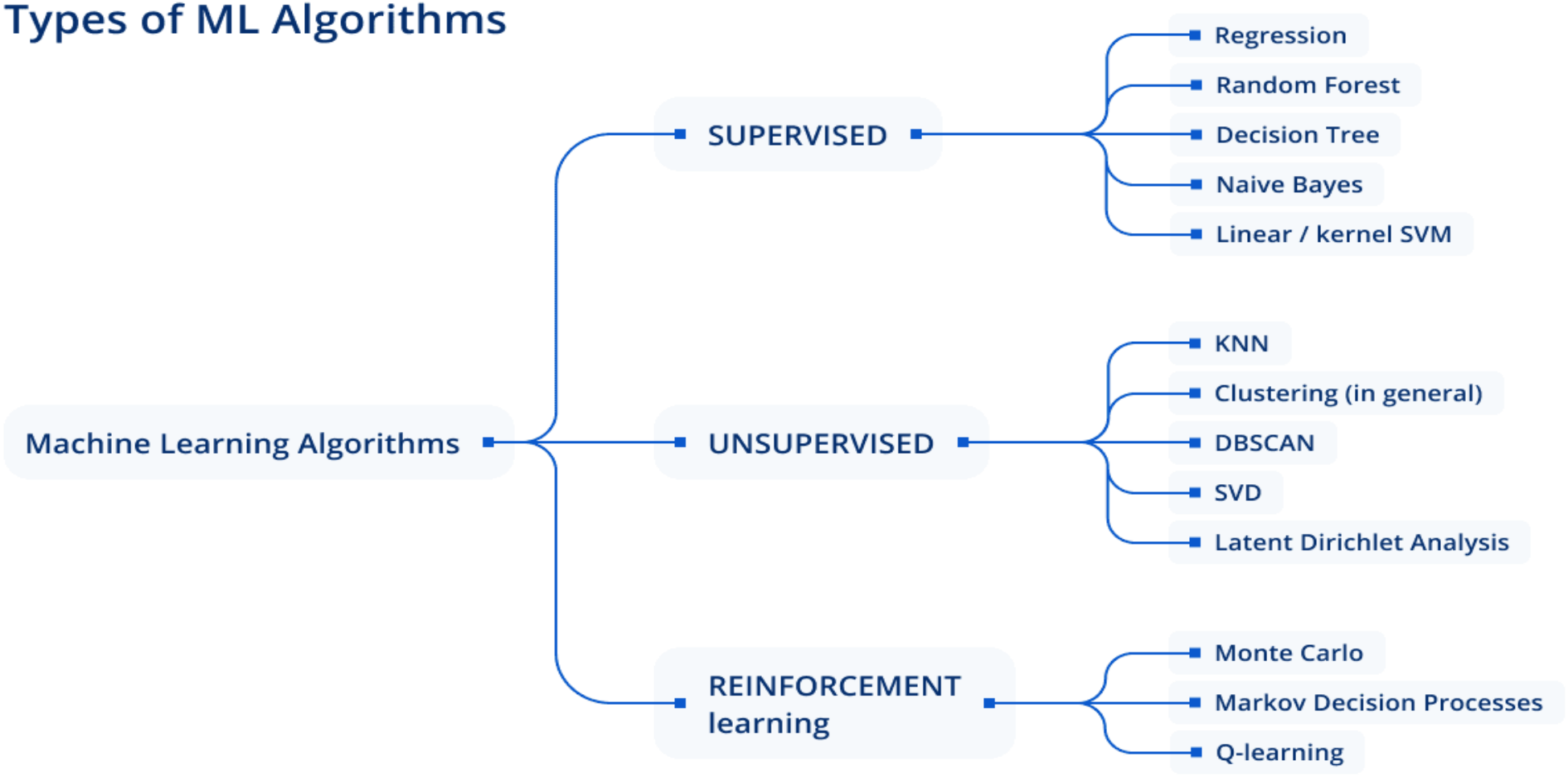
ML Algorithm

- A machine learning algorithm is the method by which the AI system conducts its task, generally predicting output values from given input data. The two main processes of machine learning algorithms are classification and regression.

Categorization of ML Algorithms

- **SUPERVISED LEARNING** is a type of Machine learning in which the machine needs external supervision to learn. The supervised learning models are trained using the labeled dataset. Once the training and processing are done, the model is tested by providing a sample test data to check whether it predicts the correct output.
- **UNSUPERVISED LEARNING** is a type of machine learning in which the machine does not need any external supervision to learn from the data, hence called unsupervised learning. The unsupervised models can be trained using the unlabeled dataset that is not classified, nor categorized, and the algorithm needs to act on that data without any supervision.

Types of ML Algorithms



Data set used from Kaggle - Spam_Ham_Dataset.csv

Basic Libraries used - Pandas, NumPy, Matplotlib, Seaborn, Sklearn

ML Algorithms used –
(i) Logistic Regression
(ii) Decision Tree
(iii) Random Forest
(iv) Decision Tree Classifier
(v) Random Forest Classifier

Libraries In Python -

- Pandas- Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.
- NumPy - is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It is open-source software.
- Matplotlib- is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.
- Seaborn- Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on the top of matplotlib library and also closely integrated to the data structures from pandas.
- Scikit-learn- (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, *k*-means and DBSCAN.

ML Algorithms - Logistic Regression

- Logistic regression aims to solve classification problems. It does this by predicting categorical outcomes, unlike linear regression that predicts a continuous outcome. In the simplest case there are two outcomes, which is called binomial, an example of which is predicting if a tumour is malignant or benign.

ML Algorithms - Decision Tree

- A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value.

ML Algorithms - Random Forest

- Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as bagging.

ML Algorithms - Decision Tree Classifier

- DecisionTreeClassifier is a class capable of performing multi-class classification on a dataset. In case that there are multiple classes with the same and highest probability, the classifier will predict the class with the lowest index amongst those classes.

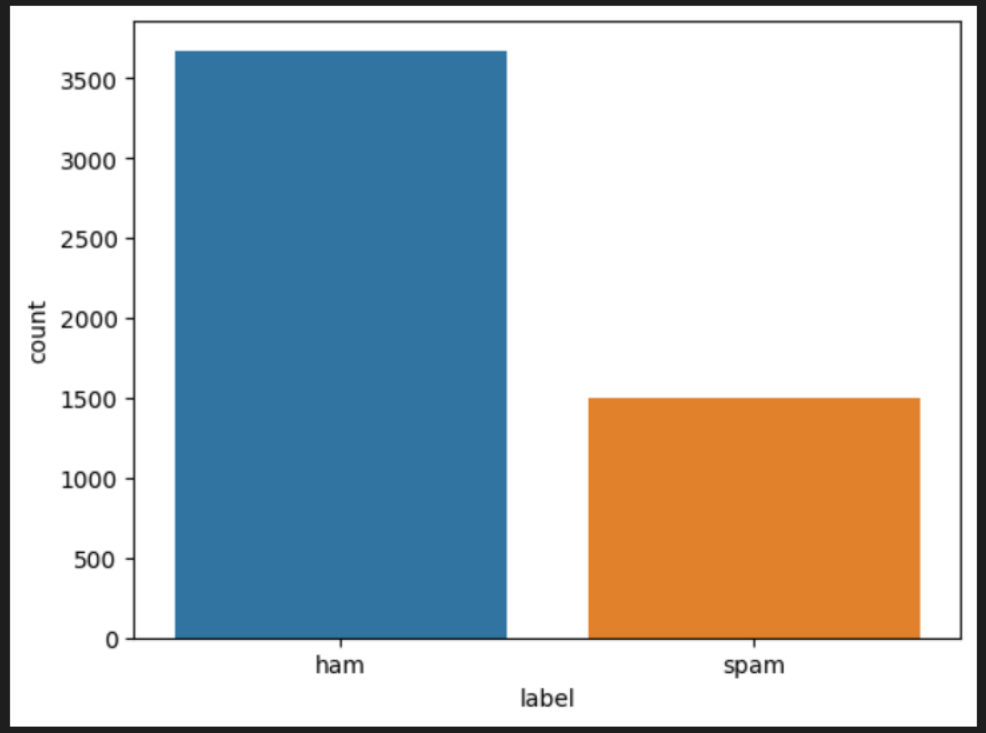
ML Algorithms - Random Forest Classifier

- The Random forest classifier creates a set of decision trees from a randomly selected subset of the training set. It is basically a set of decision trees (DT) from a randomly selected subset of the training set and then It collects the votes from different decision trees to decide the final prediction.

SPAM HAM

- Nowadays, it's likely that everyone knows what Spam means, in the context of e-mail. The use of the word "Ham", on the other hand, is relatively new and sometimes confusing.
- "Ham" is *e-mail that is not Spam*. In other words, "non-spam", or "good mail". It should be considered a shorter, snappier synonym for "non-spam".
- Its usage is particularly common among anti-spam software developers, and not widely known elsewhere; in general it is probably better to use the term "non-spam", instead.

```
[12]
# Here we used count plot for visualizing the amount of spam and ham type emails in the Label column.
sns.countplot(x = df["label"])
plt.show()
```



Logistic Regression

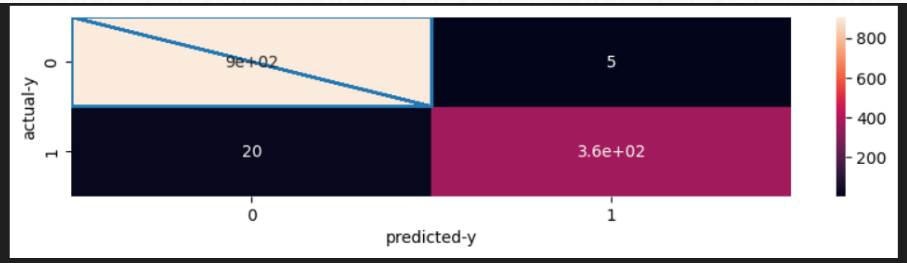
```
from sklearn.linear_model import LogisticRegression
c=LogisticRegression()
print(c.fit(x_train,y_train.values))
y_pred=c.predict(x_test)
print(f'predicted-y',y_pred[:5],'actual-y',y_test.values[:5])
print(f'predicted-y_shape',y_pred.shape,'actual-y-shape',y_test.values.shape)
```

[63]

```
LogisticRegression()
predicted-y [0 1 0 0 1] actual-y [0 1 0 0 1]
predicted-y_shape (1293,) actual-y-shape (1293,)
```

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
cm=confusion_matrix(y_pred,y_test)
plt.figure(figsize=(10,2))
sns.heatmap(cm,annot=True)
plt.plot(y_pred,y_test)
plt.xlabel('predicted-y')
plt.ylabel('actual-y')
plt.show()
```

[64]




```
print(classification_report(y_pred,y_test))
print('accuracy-score',accuracy_score(y_pred,y_test))
print('Model score',c.score(x_test,y_test))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       909
           1       0.99      0.95      0.97       384

    accuracy                           0.98      1293
   macro avg       0.98      0.97      0.98      1293
weighted avg       0.98      0.98      0.98      1293

accuracy-score 0.9806651198762568
Model score 0.9806651198762568
```

It is used **to calculate or predict the probability of a binary (yes/no) event occurring**.
In this data set we will predict whether a mail is spam or not, this can be done using logistic regression.

Decision Tree

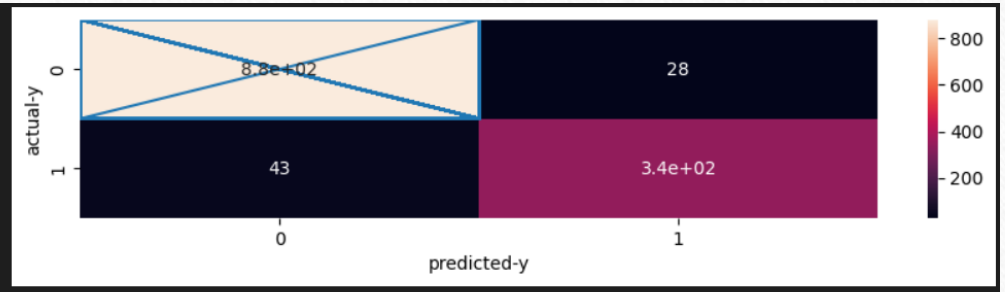
```
from sklearn.tree import DecisionTreeClassifier
c=DecisionTreeClassifier()
print(c.fit(x_train,y_train))
y_pred=c.predict(x_test)
print(f'predicted-y',y_pred[:5],'actual-y',y_test.values[:5])
print(f'predicted-y_shape',y_pred.shape,'actual-y-shape',y_test.values.shape)
```

[66]

```
DecisionTreeClassifier()
predicted-y [0 1 0 0 1] actual-y [0 1 0 0 1]
predicted-y_shape (1293,) actual-y-shape (1293,)
```

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
cm=confusion_matrix(y_pred,y_test)
plt.figure(figsize=(10,2))
sns.heatmap(cm,annot=True)
plt.plot(y_pred,y_test)
plt.xlabel('predicted-y')
plt.ylabel('actual-y')
plt.show()
```

[67]




```
print(classification_report(y_pred,y_test))
print('accuracy-score',accuracy_score(y_pred,y_test))
print('Model score',c.score(x_test,y_test))
```

[68]

```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       909
           1       0.92      0.89      0.91       384

    accuracy                           0.95      1293
   macro avg       0.94      0.93      0.93      1293
weighted avg       0.94      0.95      0.94      1293

accuracy-score 0.9450889404485692
Model score 0.9450889404485692
```

The main benefits of using a decision tree in machine learning is **its simplicity, as the decision-making process is easy to visualise and understand**. We are using quantitative data with categorical columns that makes Decision Tree fit for use.

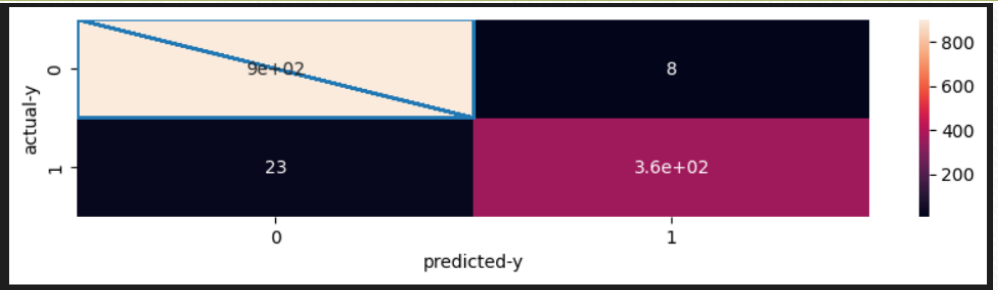
Random Forest

```
from sklearn.ensemble import RandomForestClassifier
c=RandomForestClassifier()
print(c.fit(x_train,y_train))
y_pred=c.predict(x_test)
print(f'predicted-y',y_pred[:5],'actual-y',y_test.values[:5])
print(f'predicted-y_shape',y_pred.shape,'actual-y-shape',y_test.values.shape)
```

[69]

```
RandomForestClassifier()
predicted-y [0 1 0 0 1] actual-y [0 1 0 0 1]
predicted-y_shape (1293,) actual-y-shape (1293,)
```

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
cm=confusion_matrix(y_pred,y_test)
plt.figure(figsize=(10,2))
sns.heatmap(cm,annot=True)
plt.plot(y_pred,y_test)
plt.xlabel('predicted-y')
plt.ylabel('actual-y')
plt.show()
```




```
print(classification_report(y_pred,y_test))
print('accuracy-score',accuracy_score(y_pred,y_test))
print('Model score',c.score(x_test,y_test))
```

[71]

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       909
           1       0.98      0.94      0.96       384

    accuracy                           0.98      1293
   macro avg       0.98      0.97      0.97      1293
weighted avg       0.98      0.98      0.98      1293

accuracy-score 0.9760247486465584
Model score 0.9760247486465584
```

One of the most important features of the Random Forest Algorithm is that **it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification**. In this data set we used quantitative dataset and with help of random forest the prediction can be made.

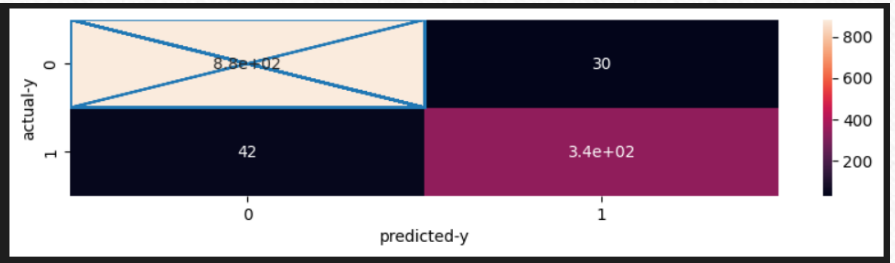
Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
c=DecisionTreeClassifier()
print(c.fit(x_train,y_train))
y_pred=c.predict(x_test)
print(f'predicted-y',y_pred[:5],'actual-y',y_test.values[:5])
print(f'predicted-y_shape',y_pred.shape,'actual-y-shape',y_test.values.shape)
```

[72]

```
DecisionTreeClassifier()
predicted-y [0 1 0 0 1] actual-y [0 1 0 0 1]
predicted-y_shape (1293,) actual-y-shape (1293,)
```

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
cm=confusion_matrix(y_pred,y_test)
plt.figure(figsize=(10,2))
sns.heatmap(cm,annot=True)
plt.plot(y_pred,y_test)
plt.xlabel('predicted-y')
plt.ylabel('actual-y')
plt.show()
```




```
print(classification_report(y_pred,y_test))
print('accuracy-score',accuracy_score(y_pred,y_test))
print('Model score',c.score(x_test,y_test))
```

```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       912
           1       0.92      0.89      0.90       381

    accuracy                           0.94      1293
   macro avg       0.94      0.93      0.93      1293
weighted avg       0.94      0.94      0.94      1293

accuracy-score 0.9443155452436195
Model score 0.9443155452436195
```

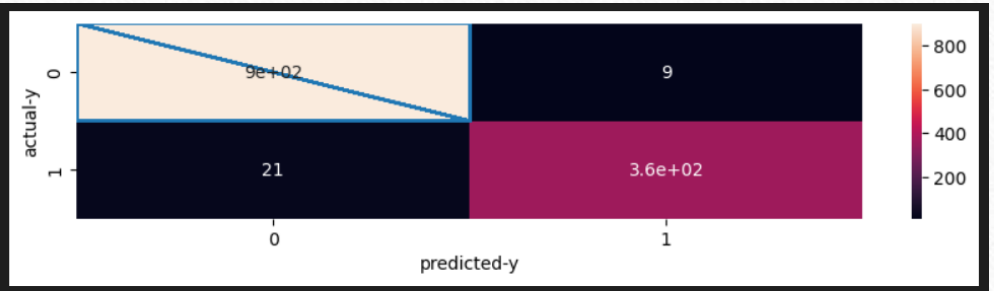
The main benefits of using a decision tree in machine learning is **its simplicity, as the decision-making process is easy to visualise and understand**.

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
c=RandomForestClassifier()
print(c.fit(x_train,y_train))
y_pred=c.predict(x_test)
print(f'predicted-y',y_pred[:5],'actual-y',y_test.values[:5])
print(f'predicted-y_shape',y_pred.shape,'actual-y-shape',y_test.values.shape)
```

```
RandomForestClassifier()
predicted-y [0 1 0 0 1] actual-y [0 1 0 0 1]
predicted-y_shape (1293,) actual-y-shape (1293,)
```

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
cm=confusion_matrix(y_pred,y_test)
plt.figure(figsize=(10,2))
sns.heatmap(cm,annot=True)
plt.plot(y_pred,y_test)
plt.xlabel('predicted-y')
plt.ylabel('actual-y')
plt.show()
```




```
print(classification_report(y_pred,y_test))
print('accuracy-score',accuracy_score(y_pred,y_test))
print('Model score',c.score(x_test,y_test))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       912
           1       0.98      0.94      0.96       381

    accuracy                           0.98      1293
   macro avg       0.98      0.97      0.97      1293
weighted avg       0.98      0.98      0.98      1293

accuracy-score 0.9767981438515081
Model score 0.9767981438515081
```

Random forest classifier performs better results for classification problems.

After analysing the predictions done by every model and checking the accuracy in different models we can conclude that the best fit machine learning model for the data set of Spam and Ham is LOGISTIC REGRESSION.

It gave the best model accuracy score of 0.9806651198762568.

Followed by the score of Random Forest Classifier with model accuracy score of 0.9767981438515081.