import dataset

```
[ ] import pandas as pd
```

```
[ ] import nltk
    import re
    import string
    string.punctuation
    from nltk.stem.porter import PorterStemmer
    from nltk.corpus import stopwords

    def tokenization(text):
        tokens = text.split(' ')
        return tokens

    def remove_punctuation(text):
        punctuationfree=[]
        for t in text:
            punctuationfree.append(''.join(i for i in t if i not in string.punctuation))
        return punctuationfree


    nltk.download('stopwords')
    stopwords=stopwords.words('english')

    def remove_stopwords(text):
        output= [i for i in text if i not in stopwords]
        return output


    porter_stemmer = PorterStemmer()

    def stemming(text):
        stem_text = [porter_stemmer.stem(word) for word in text]
        return stem_text
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
[ ] """import json

    with open('dict.json') as json_file:
        data = json.load(json_file)"""
```

```
[ ] import warnings
    warnings.filterwarnings('ignore')
```

```
[ ] import pandas as pd

    df = pd.read_excel(r'Data.xlsx')
```

```
[ ] df['Processed Text']=df['Summary Text'].str.replace('\d+', '')
    df['Processed Text']= df['Processed Text'].apply(lambda x: x.lower())
    df['Processed Text']=df['Processed Text'].apply(lambda x: tokenization(x))
    df['Processed Text']=df['Processed Text'].apply(lambda x: remove_punctuation(x))
    df['Processed Text']=df['Processed Text'].apply(lambda x: remove_stopwords(x))
    df['Processed Text']=df['Processed Text'].apply(lambda x: ' '.join(x))
```

```
[ ] df.head(5)
```

	CVE ID Number	Vulnerability Type	Summary Text	Processed Text
0	CVE-2022-43766	Denial of Service	Apache IoTDB version 0.12.2 to 0.12.6, 0.13.0 ...	apache iotdb version vulnerable denial ser...
1	CVE-2022-43365	Denial of Service	IP-COM EW9 V15.11.0.14(9732) was discovered to...	ipcom ew v discovered contain buffer overflow ...
2	CVE-2022-43035	Denial of Service	An issue was discovered in Bento4 v1.6.0-639. ...	issue discovered bento v heapbufferoverflow ap...
3	CVE-2022-43033	Denial of Service	An issue was discovered in Bento4 1.6.0-639. T...	issue discovered bento bad free component aph...
4	CVE-2022-42969	Denial of Service	The py library through 1.11.0 for Python allow...	py library python allows remote attackers con...

```
[ ] df.shape
```

```
(175850, 4)
```

```
[ ] df.sample(10)
```

	CVE ID Number	Vulnerability Type	Summary Text	Processed Text
119368	CVE-2018-1000095	Cross Site Scripting (XSS)	oVirt version 4.2.0 to 4.2.2 contains a Cross ...	ovirt version contains cross site scripting ...
32398	CVE-2021-31761	Code Execute	Webmin 1.973 is affected by reflected Cross Si...	webmin affected reflected cross site scriptin...
75072	CVE-2020-8683	Overflow	Improper buffer restrictions in system driver ...	improper buffer restrictions system driver int...
9448	CVE-2017-2327	Denial of Service	A denial of service vulnerability in Juniper N...	denial service vulnerability juniper networks ...
91252	CVE-2005-3768	Overflow	Buffer overflow in the Internet Key Exchange v...	buffer overflow internet key exchange version ...
168597	CVE-2017-15645	CSRF	CSRF exists in Webmin 1.850. By sending a GET ...	csrf exists webmin sending get request atcrea...
36672	CVE-2020-7142	Code Execute	A eventinfo_content expression language inject...	eventinfocontent expression language injection...
50745	CVE-2014-5071	Code Execute	SQL injection vulnerability in the checkPasswo...	sql injection vulnerability checkpassword func...
34146	CVE-2021-1390	Code Execute	A vulnerability in one of the diagnostic test ...	vulnerability one diagnostic test cli commands...
142826	CVE-2018-4229	Bypass	An issue was discovered in certain Apple produ...	issue discovered certain apple products macos ...

Stratified Sampling of Data

```
[ ] df.sample(5)
```

	CVE ID Number	Vulnerability Type	Summary Text	Processed Text
33954	CVE-2021-3058	Code Execute	An OS command injection vulnerability in the P...	os command injection vulnerability palo alto n...
36300	CVE-2020-8983	Code Execute	An arbitrary file write issue exists in all ve...	arbitrary file write issue exists versions cit...
64541	CVE-2007-0189	Code Execute	** DISPUTED ** PHP remote file inclusion vuln...	disputed php remote file inclusion vulnerab...
68385	CVE-2005-0931	Code Execute	PHP remote file inclusion vulnerability in The...	php remote file inclusion vulnerability includ...
11039	CVE-2016-4181	Denial of Service	Adobe Flash Player before 18.0.0.366 and 19.x ...	adobe flash player x x windows os x linux a...

+ Code + Text

```
[ ] data=pd.DataFrame(df.columns=['CVE ID Number','Processed Text','Vulnerability Type'])
```

```
[ ] data.sample(5)
```

	CVE ID Number	Processed Text	Vulnerability Type
157334	CVE-2014-1483	mozilla firefox seamonkey allow remote attac...	Gain Information
56520	CVE-2010-2875	integer signedness error adobe shockwave playe...	Code Execute
164425	CVE-2010-2542	stackbased buffer overflow isgitdirectory func...	gain priviledge
105468	CVE-2012-3435	sql injection vulnerability frontendsphppopupb...	SQL Injection
13188	CVE-2015-3828	mpegextractorparsegppmetadata function mpegext...	Denial of Service

```
[ ] from sklearn.feature_extraction.text import CountVectorizer
    matrix = CountVectorizer(max_features=100)
    X = matrix.fit_transform(data['Processed Text']).toarray()
    y=df.iloc[:, 1]
```

```
[ ] X.shape,y.shape
```

```
((175850, 100), (175850,))
```

```
[ ] from sklearn.model_selection import train_test_split

    X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, stratify=y)
```

```
[ ] y_train.value_counts(normalize=True)
```

```
Code Execute                  0.244811
Denial of Service             0.159795
Cross Site Scripting (XSS)    0.130793
Overflow                      0.128519
Gain Information              0.071936
SQL Injection                 0.058857
Bypass                        0.048905
Memory Corruption             0.038385
gain priviledge               0.031845
Directory Traversal           0.031845
CSRF                          0.021609
Security Vulnerabilities      0.019903
File Inclusion                0.011942
Http Response Splitting       0.000853
Name: Vulnerability Type, dtype: float64
```

```
[ ] y_test.value_counts(normalize=True)
```

```
Code Execute                  0.244811
Denial of Service             0.159795
Cross Site Scripting (XSS)    0.130793
Overflow                      0.128519
Gain Information              0.071936
SQL Injection                 0.058857
Bypass                        0.048905
Memory Corruption             0.038385
gain priviledge               0.031845
Directory Traversal           0.031845
CSRF                          0.021609
Security Vulnerabilities      0.019903
File Inclusion                0.011942
Http Response Splitting       0.000853
Name: Vulnerability Type, dtype: float64
```

```
[ ] (y=='Code Execute').value_counts()
```

```
False    132800
True      43050
Name: Vulnerability Type, dtype: int64
```

```
[ ] (y_train=='Code Execute').value_counts()
```

```
False    106240
True      34440
Name: Vulnerability Type, dtype: int64
```

```
[ ] (y_test=='Code Execute').value_counts()
```

```
False    26560
True      8610
Name: Vulnerability Type, dtype: int64
```

```
[ ] 43050/132800
```

```
0.32417168674698793
```

```
[ ] 34440/106240
```

```
0.32417168674698793
```

Random Forest

```
[ ] from sklearn.ensemble import RandomForestClassifier
    import matplotlib.pyplot as plt

    forest = RandomForestClassifier(n_estimators = 300)

    forest.fit(X_train, y_train)
```

```
RandomForestClassifier(n_estimators=300)
```

```
[ ] feature_names = [f"feature {i}" for i in range(X_train.shape[1])]
```

```
[ ] X_train.shape
```

```
(140680, 100)
```

```
[ ] forest.feature_importances_
```

```
array([0.00449343, 0.00147889, 0.00261033, 0.00439644, 0.00391693,
       0.00650137, 0.00183622, 0.00173676, 0.00367923, 0.02394293,
       0.00445494, 0.00779272, 0.00245723, 0.02292777, 0.04837922,
       0.02611405, 0.0024106 , 0.00123107, 0.04818545, 0.01344744,
       0.00235361, 0.01684946, 0.00343736, 0.00571154, 0.0040113 ,
       0.0310896 , 0.00467536, 0.00397426, 0.05665518, 0.00181851,
       0.00192162, 0.01745411, 0.00207852, 0.00209276, 0.00552021,
       0.03826161, 0.01748833, 0.00186004, 0.00152939, 0.00803002,
       0.00789775, 0.00663171, 0.02043749, 0.01003362, 0.00205088,
       0.00299323, 0.02507167, 0.0112765 , 0.01964146, 0.00396726,
       0.00196339, 0.0020852 , 0.00492314, 0.00280066, 0.00244398,
       0.00214195, 0.00236235, 0.01301317, 0.00258131, 0.00313774,
       0.00303979, 0.01657015, 0.00215127, 0.03967541, 0.00912828,
       0.00469581, 0.00371087, 0.00433929, 0.00198493, 0.01262561,
       0.0029301 , 0.00434349, 0.00292386, 0.01154412, 0.0089453 ,
       0.0066378 , 0.03628123, 0.01598472, 0.00472267, 0.04227848,
       0.00239842, 0.00175322, 0.00235304, 0.0272808 , 0.00351741,
       0.00355793, 0.00350827, 0.00345196, 0.00465923, 0.00535447,
```

```
       0.00272903, 0.00282988, 0.00374574, 0.00750338, 0.00210916,
       0.00961512, 0.00211852, 0.00679338, 0.00338224, 0.06056538])
```

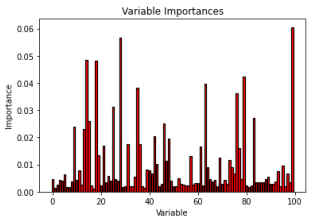
```
[ ] feature_importances = pd.DataFrame(list(zip(feature_names,forest.feature_importances_)),   columns=['feature','importance']).sort_values('importance', ascending=False)
```

```
[ ] feature_importances
```

	feature	importance
99	feature 99	0.060565
28	feature 28	0.056655
14	feature 14	0.048379
18	feature 18	0.048185
79	feature 79	0.042278
...	...	...
81	feature 81	0.001753
7	feature 7	0.001737
38	feature 38	0.001529
1	feature 1	0.001479
17	feature 17	0.001231

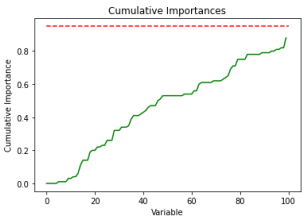
100 rows × 2 columns

```
[ ] x_values = list(range(len(forest.feature_importances_)))
    # Make a bar chart
    plt.bar(x_values, forest.feature_importances_, orientation = 'vertical', color = 'r', edgecolor = 'k', linewidth = 1.2)
    # Tick labels for x axis
    #plt.xticks(x_values, feature_names, rotation='vertical')
    # Axis labels and title
    plt.ylabel('Importance'); plt.xlabel('Variable'); plt.title('Variable Importances');
```




```
[ ] feature_importances = [(feature, round(importance, 2)) for feature, importance in zip(feature_names, forest.feature_importances_)]
```

```
[ ] import numpy as np

    sorted_importances = [importance[1] for importance in feature_importances]
    sorted_features = [importance[0] for importance in feature_importances]

    cumulative_importances = np.cumsum(sorted_importances)

    plt.plot(x_values, cumulative_importances, 'g-')
    plt.hlines(y = 0.95, xmin=0, xmax=len(sorted_importances), color = 'r', linestyles = 'dashed')

    #plt.xticks(x_values, sorted_features, rotation = 'vertical')

    plt.xlabel('Variable'); plt.ylabel('Cumulative Importance'); plt.title('Cumulative Importances');
```




```
[ ] from statistics import mean
    pivot1=mean(forest.feature_importances_)
```

```
[ ] forest.feature_importances_
```

```
array([0.00449343, 0.00147889, 0.00261033, 0.00439644, 0.00391693,
       0.00650137, 0.00183622, 0.00173676, 0.00367923, 0.02394293,
       0.00445494, 0.00779272, 0.00245723, 0.02292777, 0.04837922,
       0.02611405, 0.0024106 , 0.00123107, 0.04818545, 0.01344744,
       0.00235361, 0.01684946, 0.00343736, 0.00571154, 0.0040113 ,
       0.0310896 , 0.00467536, 0.00397426, 0.05665518, 0.00181851,
       0.00192162, 0.01745411, 0.00207852, 0.00209276, 0.00552021,
       0.03826161, 0.01748833, 0.00186004, 0.00152939, 0.00803002,
       0.00789775, 0.00663171, 0.02043749, 0.01003362, 0.00205088,
       0.00299323, 0.02507167, 0.0112765 , 0.01964146, 0.00396726,
       0.00196339, 0.0020852 , 0.00492314, 0.00280066, 0.00244398,
       0.00214195, 0.00236235, 0.01301317, 0.00258131, 0.00313774,
       0.00303979, 0.01657015, 0.00215127, 0.03967541, 0.00912828,
       0.00469581, 0.00371087, 0.00433929, 0.00198493, 0.01262561,
       0.0029301 , 0.00434349, 0.00292386, 0.01154412, 0.0089453 ,
       0.0066378 , 0.03628123, 0.01598472, 0.00472267, 0.04227848,
       0.00239842, 0.00175322, 0.00235304, 0.0272808 , 0.00351741,
       0.00355793, 0.00350827, 0.00345196, 0.00465923, 0.00535447,
       0.00272903, 0.00282988, 0.00374574, 0.00750338, 0.00210916,
       0.00961512, 0.00211852, 0.00679338, 0.00338224, 0.06056538])
```

```
[ ] feature_importances.sort(key=lambda x:x[1],reverse=True)
```

```
[ ] k=len(np.where(forest.feature_importances_ > 0.01)[0])
    print(k)
```

```
27
```

```
[ ] important_feature_names = [feature[0] for feature in feature_importances[0:k]]

    important_indices = [feature_names.index(feature) for feature in important_feature_names]

    Xtrain1 = X_train[:, important_indices]
    Xtest1 = X_test[:, important_indices]

    print('Important train features shape:', Xtrain1.shape)
    print('Important test features shape:', Xtest1.shape)
```

```
Important train features shape: (140680, 27)
```

```
Important test features shape: (35170, 27)
```

[ ]

[ ]

Testing

```
[ ] from sklearn.ensemble import RandomForestClassifier
    import matplotlib.pyplot as plt

    forest1 = RandomForestClassifier(n_estimators = 300)
    forest1.fit(Xtrain1, y_train)
```

```
RandomForestClassifier(n_estimators=300)
```

```
[ ] y_pred=forest.predict(X_test)
```

```
[ ] y_pred1=forest1.predict(Xtest1)
```

```
[ ] from sklearn.metrics import accuracy_score
```

```
[ ] p=accuracy_score(y_test,y_pred)
    p
```

```
0.5904179698606767
```

```
[ ] p1=accuracy_score(y_test,y_pred1)
    p1
```

```
0.6553596815467728
```

```
[ ] t=p1-p
    q='increase' if t>0 else 'decrease'
```

```
[ ] print('%.2f'%(100*(abs(p1-p)/p)),'%',q)
```

```
11.00 % increase
```

DEEP LEARNING LSTM

```
[ ] import nltk
    from nltk.corpus import stopwords
    import re
    import joblib
    from keras.preprocessing.text import Tokenizer
    import gensim
    from keras_preprocessing.sequence import pad_sequences
    from sklearn.preprocessing import LabelEncoder
    from keras.layers import Embedding
    from keras.models import Sequential
    from keras.layers import Dense,LSTM,Dropout
    from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
```

```
[ ] from sklearn.model_selection import train_test_split
    X,y=df['Processed Text'],df['Vulnerability Type']
    train_df,test_df,y_train,y_test=train_test_split(X,y,test_size=0.20,stratify=y)
```

```
[ ] documents = [text.split() for text in train_df]
    w2v_model = gensim.models.word2vec.Word2Vec(size=300,
                                                window=7,
                                                min_count=10,
                                                workers=8)
    w2v_model.build_vocab(documents)
    words = w2v_model.wv.vocab.keys()
    vocab_size = len(words)
    print("Vocab size", vocab_size)
    #w2v_model.train(documents, total_examples=len(documents), epochs=30)
```

```
Vocab size 10867
```

```
[ ] train_df.head()
```

```
118030    liferay portal  ga allows xss via journal arti...
173452    integer overflow dpautilities library emc data...
39368     command injection vulnerability allows attacke...
86450     array index error freetype  used mozilla firef...
138503    directory traversal vulnerability aol instant ...
Name: Processed Text, dtype: object
```

```
[ ] import numpy as np
    x=len(np.where(y=='Denial of Service')[0])
    y1=len(np.where(y_test=='Denial of Service')[0])
    print(x,y1,y1/x)
```

```
28100 5620 0.2
```

```
[ ] from tensorflow.keras.preprocessing.text import Tokenizer

    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(train_df)
```

```
[ ] tokenizer.word_index
```

```
{'via': 1,
 'remote': 2,
 'allows': 3,
 'attackers': 4,
 'vulnerability': 5,
 'arbitrary': 6,
 'cve': 7,
 'execute': 8,
 'code': 9,
 'service': 10,
 'denial': 11,
 'cause': 12,
 'x': 13,
 'crafted': 14,
 'parameter': 15,
 'sql': 16,
 'memory': 17,
 'allow': 18,
 'file': 19,
 'attacker': 20,
 'overflow': 21,
 'buffer': 22,
 'web': 23,
 'earlier': 24,
 'could': 25,
 'corruption': 26,
```

```
'windows': 27,
'users': 28,
'information': 29,
'xss': 30,
'scripting': 31,
'unspecified': 32,
'user': 33,
'issue': 34,
'crosssite': 35,
'injection': 36,
'commands': 37,
'server': 38,
'aka': 39,
'versions': 40,
'application': 41,
'affected': 42,
'function': 43,
'crash': 44,
'vectors': 45,
'execution': 46,
'local': 47,
'multiple': 48,
'sp': 49,
'possibly': 50,
'script': 51,
'access': 52,
'vulnerabilities': 53,
'html': 54,
'system': 55,
'microsoft': 56,
'privileges': 57,
'id': 58,
```

[ ]
```
vocab_size=len(tokenizer.word_index)+1
vocab_size
```

```
124526
```

[ ]
```
X_train = pad_sequences(tokenizer.texts_to_sequences(train_df), maxlen=300)
X_train
```

```
array([[    0,     0,     0, ...,   572, 62023, 62024],
       [    0,     0,     0, ...,    95,   340,   160],
       [    0,     0,     0, ...,  1437,   240,   450],
       ...,
       [    0,     0,     0, ...,   259,  4886,   366],
       [    0,     0,     0, ...,  1695,   377,   161],
       [    0,     0,     0, ...,     1,   434,   491]], dtype=int32)
```

[ ]
```
X_test = pad_sequences(tokenizer.texts_to_sequences(test_df), maxlen=300)
X_test
```

```
array([[    0,     0,     0, ...,   280,   282,   106],
       [    0,     0,     0, ...,     1,  2323,    15],
       [    0,     0,     0, ...,     1,  6992, 21219],
       ...,
       [    0,     0,     0, ...,   228,   314,    19],
       [    0,     0,     0, ...,   297,   190,   149],
       [    0,     0,     0, ...,     1,  3156,    15]], dtype=int32)
```

[ ]
```
y_train.head()
```

```
118030    Cross Site Scripting (XSS)
173452      Security Vulnerabilities
39368                    Code Execute
86450                        Overflow
138503            Directory Traversal
Name: Vulnerability Type, dtype: object
```

[ ]
```
y_test.head()
```

```
146881                Bypass
108359         SQL Injection
143665                Bypass
165746     gain priviledge
95927      Memory Corruption
Name: Vulnerability Type, dtype: object
```

[ ]
```
labelencoder = LabelEncoder()
y_train = labelencoder.fit_transform(y_train)
y_test=labelencoder.fit_transform(y_test)
```

[ ]
```
y_train.shape
```

```
(140680,)
```

[ ]
```
y_test.shape
```

```
(35170,)
```

[ ]
```
"""embedding_matrix = np.zeros((vocab_size, 300))
for word, i in tokenizer.word_index.items():
  if word in w2v_model.wv:
    embedding_matrix[i] = w2v_model.wv[word]
print(embedding_matrix.shape)"""
#weights=[embedding_matrix],
```

```
'embedding_matrix = np.zeros((vocab_size, 300))\nfor word, i in tokenizer.word_index.items():\n  if word in w2v_model.wv:\n    embedding_matrix[i] = w2v_model.wv[word]\nprint(embedding_matrix.shape)'
```

[ ]
```
embedding_layer = Embedding(vocab_size, 300, input_length=300)
```

[ ]
```
model = Sequential()
model.add(embedding_layer)
model.add(Dropout(0.5))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 300, 300)          37357800

dropout_1 (Dropout)          (None, 300, 300)          0

lstm_1 (LSTM)                (None, 100)               160400

dense_1 (Dense)              (None, 1)                 101

=================================================================
Total params: 37,518,301
Trainable params: 37,518,301
Non-trainable params: 0
_________________________________________________________________
```

[ ]
```
model.compile(loss='binary_crossentropy',
              optimizer="adam",
              metrics=['accuracy'])
```

[ ]
```
model.fit(X_train, y_train,batch_size=1000,epochs=1,validation_split=0.1,verbose=1)
```

```
127/127 [==============================] - 1761s 14s/step - loss: -62.7944 - accuracy: 0.0217 - val_loss: -99.8143 - val_accuracy: 0.0214
<keras.callbacks.History at 0x7fe10e143c40>
```

```
[ ] scores = model.predict(X_test, verbose=1, batch_size=1000)
```

```
[ ] scores
```

```
array([[1.],
       [1.],
       [1.],
       ...,
       [1.],
       [1.],
       [1.]], dtype=float32)
```

```
[ ] arr=[]
    for item in scores:
      arr.append(item[0])
    arr
```

```
[ ] y_test
```

```
[ ] del model
```

```
[ ] del model_history
```

```
[ ] del embedding_layer
```

LSTM 2

```
[ ] tweet = df['Processed Text'].values
```

```
[ ] tweet
```

```
array(['apache iotdb version     vulnerable denial service attack accepting untrusted patterns regexp queries java  users upgrade  addresses issue use later version java avoid',
       'ipcom ew v discovered contain buffer overflow formsetdebugcfg function vulnerability allows attackers cause denial service dos via crafted string',
       'issue discovered bento v heapbufferoverflow apdecatomapdecatom apdecatomcpp leading denial service dos demonstrated mpaac',
       ...,
       'sql injection vulnerability bolumphp teknoportal allows remote attackers execute arbitrary sql commands via id parameter note provenance information unknown details obtained solely third party information',
       'buffer overflow mpinfo  allows attackers execute arbitrary code via long command line argument  note mpinfo installed setuid setgid reasonable context issue might vulnerability',
       'crosssite scripting xss vulnerability aloginphp vizra allows remote attackers inject arbitrary web script html via message parameter'],
      dtype=object)
```

```
[ ] X_train.shape,y_train.shape
```

```
((140680, 100), (140680,))
```

```
[ ] from tensorflow.keras.preprocessing.text import Tokenizer
    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(X_train.text)
```

```
[ ] from tensorflow.keras.preprocessing.text import Tokenizer

    tokenizer = Tokenizer(num_words=5000)

    tokenizer.fit_on_texts(tweet)
```

```
[ ] from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import LSTM,Dense, Dropout, SpatialDropout1D
    from tensorflow.keras.layers import Embedding


    embedding_vector_length = 32
    model = Sequential()
    vocab_size=len(tokenizer.word_index) + 1
    model.add(Embedding(vocab_size, embedding_vector_length, input_length=100))
    model.add(SpatialDropout1D(0.25))
    model.add(LSTM(50, dropout=0.5, recurrent_dropout=0.5))
    model.add(Dropout(0.2))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy',optimizer='adam', metrics=['accuracy'])

    print(model.summary())
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 100, 32)           4452896

 spatial_dropout1d_1 (Spatia  (None, 100, 32)          0
 lDropout1D)

 lstm_1 (LSTM)               (None, 50)                16600

 dropout_1 (Dropout)         (None, 50)                0

 dense_1 (Dense)             (None, 1)                 51

=================================================================
Total params: 4,469,547
Trainable params: 4,469,547
Non-trainable params: 0
_________________________________________________________________
None
```

```
[ ] encoded_docs = tokenizer.texts_to_sequences(tweet)
```

```
[ ]

    from tensorflow.keras.preprocessing.sequence import pad_sequences

    padded_sequence = pad_sequences(X_train, maxlen=100)
```

```
[ ] sentiment_label=y_train.factorize()
    sentiment_label
```

```
(array([0, 1, 2, ..., 5, 0, 9]),
 Index(['Bypass', 'Memory Corruption', 'SQL Injection', 'Denial of Service',
        'Code Execute', 'Overflow', 'Directory Traversal', 'Gain Information',
        'gain priviledge', 'Cross Site Scripting (XSS)', 'CSRF',
        'Security Vulnerabilities', 'File Inclusion',
        'Http Response Splitting'],
       dtype='object'))
```

```
[ ] history = model.fit(X_train,sentiment_label[0], epochs=5, batch_size=1000)
```

```
[ ] res=model.predict(padded_sequence)
```

```
5496/5496 [==============================] - 174s 32ms/step
```

```
[ ] res
```

```
array([[0.8794238 ],
       [0.8850978 ],
       [0.89112335],
       ...,
       [0.8797933 ],
       [0.8810387 ],
       [0.8901106 ]], dtype=float32)
```

```
[ ] pivot=0.3
```

```
[ ] new_res=[1 if x>=pivot else 0 for x in res]
    print(new_res)
    new_res.count(1),new_res.count(0)
```

```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
(175850, 0)
```

```
[ ] list(sentiment_label[0]).count(1),list(sentiment_label[0]).count(0)
```

```
(43050, 28100)
```

```
[ ] from sklearn.metrics import accuracy_score
    print(accuracy_score(new_res,sentiment_label[0]))
```

```
0.24481091839636054
```

```
[ ] df1=pd.DataFrame(df,columns=['CVE ID Number','Vulnerability Type','Processed Text','DoS'])
```

```
[ ] df1['DoS1']=list(new_res)
```

```
[ ] df1.sample(5)
```

	CVE ID Number	Vulnerability Type	Processed Text	DoS	DoS1
29257	CVE-2022-28823	Code Execute	adobe framemaker versions u earlier u earlier ...	NaN	1
52629	CVE-2013-2145	Code Execute	cpansign verify functionality modulesignature ...	NaN	1
81411	CVE-2016-8699	Overflow	heapbased buffer overflow bmreadbodybmp functi...	NaN	1
72594	CVE-2021-46225	Overflow	buffer overflow gmfopenmesh function libmeshb ...	NaN	1
164333	CVE-2010-3716	gain priviledge	beusercreation task typo x x allows remote a...	NaN	1

```
[ ] df1.head(10)
```

	CVE ID Number	Vulnerability Type	Processed Text	DoS	DoS1
0	CVE-2022-43766	Denial of Service	apache iotdb version vulnerable denial ser...	NaN	1
1	CVE-2022-43365	Denial of Service	ipcom ew v discovered contain buffer overflow ...	NaN	1
2	CVE-2022-43035	Denial of Service	issue discovered bento v heapbufferoverflow ap...	NaN	1
3	CVE-2022-43033	Denial of Service	issue discovered bento bad free component aph...	NaN	1
4	CVE-2022-42969	Denial of Service	py library python allows remote attackers con...	NaN	1
5	CVE-2022-42300	Denial of Service	issue discovered veritas netbackup related ve...	NaN	1
6	CVE-2022-42299	Denial of Service	issue discovered veritas netbackup related ve...	NaN	1
7	CVE-2022-41715	Denial of Service	programs compile regular expressions untrusted...	NaN	1
8	CVE-2022-41665	Denial of Service	vulnerability identified sicam p versions v s...	NaN	1
9	CVE-2022-41556	Denial of Service	resource leak gwbackendc lighttpd could lead...	NaN	1

```
[ ] import matplotlib.pyplot as plt

    #plt.plot(history.history['accuracy'], label='acc')
    #plt.plot(history.history['val_accuracy'], label='val_acc')
    plt.legend()
    plt.show()

    plt.savefig("Accuracy plot.jpg")
```

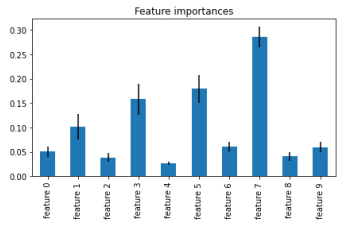
```
[ ] import nltk
    import pandas as pd
    from textblob import Word
    from nltk.corpus import stopwords
    from sklearn.preprocessing import LabelEncoder
    from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
    from keras.models import Sequential
    from keras.preprocessing.text import Tokenizer
    from keras_preprocessing.sequence import pad_sequences
    from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
    from sklearn.model_selection import train_test_split
    #Model Building
    model = Sequential()
    model.add(Embedding(500, 120, input_length = X.shape[1]))
    model.add(SpatialDropout1D(0.4))
    model.add(LSTM(704, dropout=0.2, recurrent_dropout=0.2))
    model.add(Dense(352, activation='LeakyReLU'))
    model.add(Dense(3, activation='softmax'))
    model.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics = ['accuracy'])
    print(model.summary())
    #Model Training
    #model.fit(X,y, epochs = 20, batch_size=32, verbose =1)
    #Model Testing
    #model.evaluate(X,y)
```

Impurity & Permutation Model

```
[ ] import time
    import numpy as np

    start_time = time.time()
    importances = forest.feature_importances_
    std = np.std([tree.feature_importances_ for tree in forest.estimators_], axis=0)
    elapsed_time = time.time() - start_time

    print(f"Elapsed time to compute the importances: {elapsed_time:.3f} seconds")
```

```
Elapsed time to compute the importances: 0.032 seconds
```

```
[ ] import pandas as pd
    import matplotlib.pyplot as plt

    forest_importances = pd.Series(importances, index=feature_names)

    fig, ax = plt.subplots()
    forest_importances.plot.bar(yerr=std, ax=ax)
    ax.set_title("Feature importances")
    fig.tight_layout()
```



[ ]
```
from sklearn.inspection import permutation_importance

start_time = time.time()
result = permutation_importance(
    forest, X_test, y_test, n_repeats=10, random_state=42, n_jobs=2
)
elapsed_time = time.time() - start_time
print(f"Elapsed time to compute the importances: {elapsed_time:.3f} seconds")

forest_importances = pd.Series(result.importances_mean, index=feature_names)
```

```
Elapsed time to compute the importances: 74.512 seconds
```

[ ]
```
fig, ax = plt.subplots()
forest_importances.plot.bar(yerr=result.importances_std, ax=ax)
ax.set_title("Feature importances using permutation on full model")
fig.tight_layout()
plt.show()
```