In [1]:
```
import pandas as pd
import requests
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix, class
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/jayantjha/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[1]: True

In [2]:
```
df=pd.read_csv('dialogues_dataset_formatted_added_links.csv')
df
```

Out[2]:

	voice_id	num_caller	num_receiver	date	time	is_attack	dialog
0	4444444444	6094660035	9082456893	20050215	2050	False	This is mike smith from orange. I am a sales r...
1	5555555555	6094660036	9082456847	20050221	1254	False	Hi this is mike jones from orange. I am the sa...
2	3333333333	6094660038	9082456829	20050422	1315	True	I am john doe from apple computers where I am ...
3	2222222222	6094660035	9082456896	20050515	1150	True	This is john candy from apple. I am a computer...
4	1111111111	4879301873	9082456894	20050524	1433	False	hi this is moe sislack from moes tavern. I am ...
...	...	...	...	...	...	...	...
142	2666666666	6094660035	9082456830	20060416	842	True	hi this is jenny heights from notre dame. I am...
143	2777777777	6094660035	9082456867	20060518	1403	True	hi this is your sales rep samuel pitt from roc...
144	2999999999	6094660035	9082456890	20060520	1126	True	hello I am a sales rep from orange. oh sorry b...
145	2010101010	6094660035	9082456892	20060819	1638	True	Hi this is justin jones I am a sales rep from ...
146	2888888888	6094660035	9082456891	20061015	952	False	I am john doe from Apple where I am a sales re...

147 rows × 7 columns

In [3]:
```
pattern = r'(https?:\/\/(?:www\.)?[-a-zA-Z0-9@:%._+~#=]{1,256}\.[a-
```

In [4]:
```
df['link'] = df["dialog"].str.extract(pattern, expand=False).str.st
```

In [5]:
```
df_link_filtered=df[df['link'].notnull()]
```

In [6]:
```
df_link_filtered
```

Out[6]:

	voice_id	num_caller	num_receiver	date	time	is_attack	dialog	
67	4444444441	6094660036	9082456847	20070209	1023	True	Hi this is emily thirsty from purdue universit...	https://te
68	8888888881	6094660036	9082456847	20070211	1612	False	Hi this is claire foster from purdue universit...	
138	2555555555	4879301800	9082456836	20060209	1219	False	this is willard white from ww industries. I am...	https://te

In [7]:
```
!pip install requests
```

```
Requirement already satisfied: requests in /Users/jayantjha/opt/an
aconda3/lib/python3.9/site-packages (2.26.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /Users/jay
antjha/opt/anaconda3/lib/python3.9/site-packages (from requests) (
1.26.7)
Requirement already satisfied: charset-normalizer~=2.0.0 in /Users
/jayantjha/opt/anaconda3/lib/python3.9/site-packages (from request
s) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /Users/jayantjha/op
t/anaconda3/lib/python3.9/site-packages (from requests) (3.2)
Requirement already satisfied: certifi>=2017.4.17 in /Users/jayant
jha/opt/anaconda3/lib/python3.9/site-packages (from requests) (202
1.10.8)
```

In [8]:
```
# import requests
# df_link_filtered['status']=requests.get("https://wot-web-risk-and
# print(df_link_filtered)
```

In [9]:
```
import json
urldict=df_link_filtered['link'].to_dict()
url=[]
# for values in urldict.values():
#   url.append({"url": values})
# print(url)

print(json.dumps(url))
#url=json.dumps(url)
print(type(url))
```

```
[]
<class 'list'>
```

In [10]:
```
api_key='AIzaSyDxdEiqquLVWhEO-rWzKUMO4XZzDY0c1V0'
api_name='api key name 1'
```

In [11]:
```
client_Seecret='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'
client_ID='477452251798-bbbl25v45iqv2jnrocueraejlp29e4gg.apps.googl
client_name='Web client 1'
```

In [12]:
```
for i in urldict.values():
    print(i)
```

```
https://testsafebrowsing.appspot.com/s/malware.html
(https://testsafebrowsing.appspot.com/s/malware.html)
http://malware.testing.google.test
(http://malware.testing.google.test)
https://testsafebrowsing.appspot.com/s/malware.html
(https://testsafebrowsing.appspot.com/s/malware.html)
```

In [13]:
```
df['mal_check']=500
for index, i in urldict.items():
    print(index)
    header={"Content-Type": "application/json"}
    url="https://safebrowsing.googleapis.com/v4/threatMatches:find?
    data= {
        "client": {
          "clientId":client_Seecret,
          "clientVersion": "1.5.2"
        },
        "threatInfo": {
          "threatTypes":[ "SOCIAL_ENGINEERING","UNWANTED_SOFTWARE",
          "platformTypes":["ANY_PLATFORM"],
          "threatEntryTypes": ["URL"],
          "threatEntries": [ {"url":i}
    #
    #           urljson

    #         {"url": "https://testsafebrowsing.appspot.com/s/malwa
    #           "url":"http://malware.testing.google.test",
    #           "url": "https://testsafebrowsing.appspot.com/s/malw
```

```
                ]
            }
          }
        response = requests.post(url, headers=header, json=data)

        print("Status Code", response.status_code)
        print("JSON Response ", response.json())
        print(response.json()!={})
        if response.json()!={}:
            df['mal_check'][index]=1
        else:
            df['mal_check'][index]=400
        print(df.loc[[index]])
# df['mal_check'].fillna(0, inplace=True)
df
```

```
Status Code 200
JSON Response  {'matches': [{'threatType': 'MALWARE', 'platformTyp
e': 'ANY_PLATFORM', 'threat': {'url': 'https://testsafebrowsing.ap
pspot.com/s/malware.html'}, 'cacheDuration': '300s', 'threatEntryT
ype': 'URL'}]}
True
      voice_id  num_caller  num_receiver      date  time  is_attac
k  \
67  4444444441  6094660036    9082456847  20070209  1023       Tru
e

                                                   dialog  \
67  Hi this is emily thirsty from purdue universit...

                                                     link  mal_check
67  https://testsafebrowsing.appspot.com/s/malware...
(https://testsafebrowsing.appspot.com/s/malware...)          1
68

/var/folders/s0/rpqp729n2d73nk_4kzftjbp80000gn/T/ipykernel_689/236
```

In [14]:

```
# Randomize the dataset
# data_randomized = df.sample(frac=1, random_state=1)

# Calculate index for split
# training_test_index = round(len(data_randomized) * 0.8)

# Split into training and test sets
# training_set = data_randomized[:training_test_index].reset_index(
# test_set = data_randomized[training_test_index:].reset_index(drop

# print(training_set.shape)
# print(test_set.shape)
```

In [15]:
```
# training_set['dialog'] = training_set['dialog'].str.replace(
#     '\W', ' ') # Removes punctuation
# training_set['dialog'] = training_set['dialog'].str.lower()
# training_set.head(3)
```

In [16]:
```
df = df.drop(columns=["voice_id","num_caller","num_receiver","date"
# # #df = df.rename(columns={"v1":"Label", "v2":"Text"})
# # df.head()
```

In [17]:
```
df
```

Out[17]:

	is_attack	dialog	mal_check
0	False	This is mike smith from orange. I am a sales r...	500
1	False	Hi this is mike jones from orange. I am the sa...	500
2	True	I am john doe from apple computers where I am ...	500
3	True	This is john candy from apple. I am a computer...	500
4	False	hi this is moe sislack from moes tavern. I am ...	500
...	...	...	...
142	True	hi this is jenny heights from notre dame. I am...	500
143	True	hi this is your sales rep samuel pitt from roc...	500
144	True	hello I am a sales rep from orange. oh sorry b...	500
145	True	Hi this is justin jones I am a sales rep from ...	500
146	False	I am john doe from Apple where I am a sales re...	500

147 rows × 3 columns

In [18]:
```
stopset = set(stopwords.words("english"))
vectorizer = TfidfVectorizer(stop_words=stopset,binary=True)
vectorizer = TfidfVectorizer()
df['numClass'] = df['is_attack'].map({False:0, True:1})
df.drop(columns=["is_attack"], axis=0)
X= vectorizer.fit_transform(df.dialog,df.mal_check)
# Extract target column 'Class'
y = df.numClass
```

In [19]:
```
print(X.shape)
```

```
(147, 489)
```

In [20]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
print(y.value_counts())
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
1    78
0    69
Name: numClass, dtype: int64
(117, 489)
(117,)
(30, 489)
(30,)
```

In [21]:
```
print(X_train)
print(y_train)
```

```
  (0, 158)      0.2761292720563127
  (0, 467)      0.349590147648534
  (0, 25)       0.29665796669500416
  (0, 109)      0.349590147648534
  (0, 211)      0.3089425483812035
  (0, 384)      0.26736095423740935
  (0, 387)      0.25560057741762127
  (0, 55)       0.2761292720563127
  (0, 437)      0.14632170848624773
  (0, 230)      0.21953360502218566
  (0, 485)      0.12534533556685065
  (0, 478)      0.2035930042985445
  (0, 340)      0.22464590380364727
  (0, 347)      0.22464590380364727
  (0, 15)       0.1955389770146042
  (0, 145)      0.12992328025080843
  (1, 203)      0.2952237413030607
  (1, 221)      0.2180627335391244
  (1, 54)       0.27265559327318234
  (1, 82)       0.24422308930657607
  (1, 439)      0.2952237413030607
  (1, 460)      0.2180627335391244
  (1, 64)       0.2566432374210925
  (1, 367)      0.17948222965715627
  (1, 272)      0.17292643747936767
  :     :
  (116, 399)    0.30814090697832786
  (116, 413)    0.21980989544173635
  (116, 103)    0.21980989544173635
  (116, 138)    0.18305110243142814
  (116, 263)    0.18305110243142814
  (116, 171)    0.19421508282270394
  (116, 92)     0.200584522879018
  (116, 274)    0.200584522879018
  (116, 150)    0.24053709612541452
```

```
  (116, 194)    0.2618188936756173
  (116, 80)     0.2932285494844392
  (116, 60)     0.15947256548123437
  (116, 281)    0.21144542547701559
  (116, 282)    0.21549688037290668
  (116, 247)    0.13298071202610465
  (116, 56)     0.10157105621728274
  (116, 425)    0.10081592344971586
  (116, 174)    0.10081592344971586
  (116, 387)    0.18567216538512935
  (116, 485)    0.0910527672149866
  (116, 15)     0.07102125051183791
  (116, 145)    0.09437825619107365
  (116, 380)    0.22937311573413874
  (116, 200)    0.07397004675208253
  (116, 431)    0.07397004675208253
146    0
26     1
86     1
49     1
54     1
      ..
58     1
70     1
108    1
61     1
43     1
Name: numClass, Length: 117, dtype: int64
```

In [22]:
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier

from sklearn.metrics import f1_score
from sklearn.model_selection import learning_curve,validation_curve
from sklearn.model_selection import KFold

objects = ('Multi-NB', 'DTs', 'AdaBoost', 'KNN', 'RF')
```

In [23]:
```
def train_classifier(clf, X_train, y_train):
    clf.fit(X_train, y_train)

# function to predict features
def predict_labels(clf, features):
    return(clf.predict(features))
```

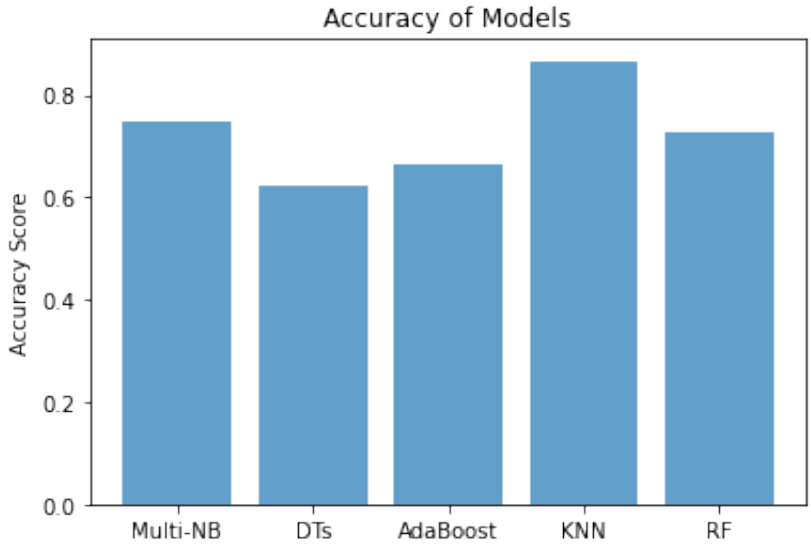
In [24]:
```
# Initialize the three models
A = MultinomialNB(alpha=1.0,fit_prior=True)
B = DecisionTreeClassifier(random_state=42)
C = AdaBoostClassifier(n_estimators=100)
D = KNeighborsClassifier(n_neighbors=1)
E = RandomForestClassifier(n_estimators=10, max_depth=None, min_sam
```

In [25]:
```
clf = [A,B,C,D,E]
pred_val = [0,0,0,0,0]

for a in range(0,5):
    train_classifier(clf[a], X_train, y_train)
    y_pred = predict_labels(clf[a],X_test)
    pred_val[a] = f1_score(y_test, y_pred)
    print(pred_val[a])
```

```
0.7500000000000001
0.6206896551724138
0.6666666666666666
0.8666666666666667
0.7272727272727272
```

In [26]:
```
# ploating data for F1 Score
import numpy as np
y_pos = np.arange(len(objects))
y_val = [ x for x in pred_val]
plt.bar(y_pos,y_val, align='center', alpha=0.7)
plt.xticks(y_pos, objects)
plt.ylabel('Accuracy Score')
plt.title('Accuracy of Models')
plt.show()
```



In [ ]:

In [ ]:

In [ ]: