4.2 Source Code

In [1]:

```
import re
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from urllib.parse import urlparse
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, ExtraTreesClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.naive_bayes import GaussianNB
from tld import get_tld, is_tld
```

In [2]:

```
data=pd.read_csv("C:\\Users\\2011c\\Downloads\\ad\\malicious_phish.csv")
data.head()
```

Out[2]:

	url	type
0	br-icloud.com.br	phishing
1	mp3raid.com/music/krizz_kaliko.html	benign
2	bopsecrets.org/rexroth/cr/1.htm	benign
3	http://www.garage-pirenne.be/index.php?option=...	defacement
4	http://adventure-nicaragua.net/index.php?optio...	defacement

Meta information of Dataframe

In [3]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 651191 entries, 0 to 651190
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype 
---  ------  --------------   ----- 
 0   url     651191 non-null  object
 1   type    651191 non-null  object
dtypes: object(2)
memory usage: 9.9+ MB
```

Checking for NaN values

In [4]:

```
data.isna().sum()
```

Out[4]:

```
url     0
type    0
dtype: int64
```

In [5]:

```
count = data.type.value_counts()
count
```

Out[5]:

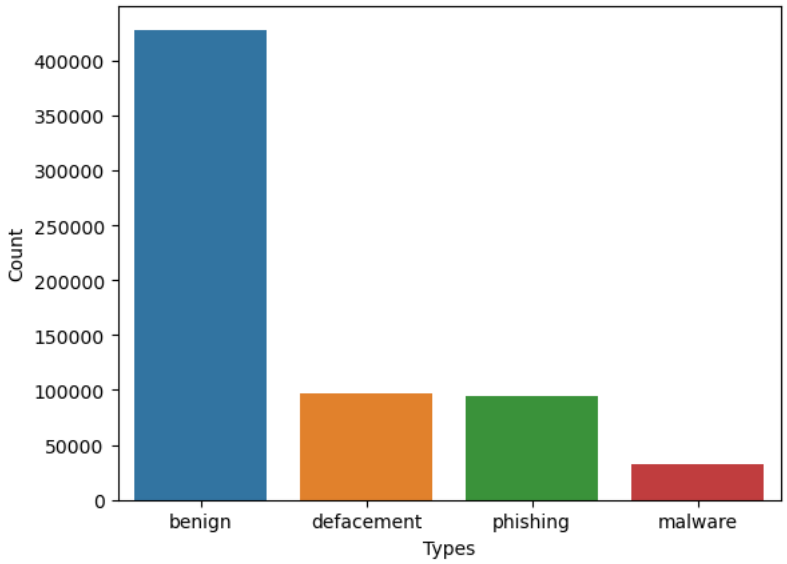
```
benign        428103
defacement     96457
phishing       94111
malware        32520
Name: type, dtype: int64
```

In [6]:

```
sns.barplot(x=count.index, y=count)
plt.xlabel('Types')
plt.ylabel('Count')
```

Out[6]:

```
Text(0, 0.5, 'Count')
```



#first have to omit the (www.) from the URL which is in fact a sub domain in itself.

In [7]:

```
data['url'] = data['url'].replace('www.', '', regex=True)
data
```

Out[7]:

	url	type
0	br-icloud.com.br	phishing
1	mp3raid.com/music/krizz_kaliko.html	benign
2	bopsecrets.org/rexroth/cr/1.htm	benign
3	http://garage-pirenne.be/index.php?option=com_...	defacement
4	http://adventure-nicaragua.net/index.php?optio...	defacement
...	...	...
651186	xbox360.ign.com/objects/850/850402.html	phishing
651187	games.teamxbox.com/xbox-360/1860/Dead-Space/	phishing
651188	gamespot.com/xbox360/action/deadspace/	phishing
651189	en.wikipedia.org/wiki/Dead_Space_(video_game)	phishing
651190	angelfire.com/goth/devilmaycrytonite/	phishing

651191 rows × 2 columns

In [10]:

```
data.head()
```

Out[10]:

	url	type	Category
0	br-icloud.com.br	phishing	2
1	mp3raid.com/music/krizz_kaliko.html	benign	0
2	bopsecrets.org/rexroth/cr/1.htm	benign	0
3	http://garage-pirenne.be/index.php?option=com_...	defacement	1
4	http://adventure-nicaragua.net/index.php?optio...	defacement	1

In [11]:

```
rem = {"Category": {"benign": 0, "defacement": 1, "phishing":2, "malware":3}}
data['Category'] = data['type']
data = data.replace(rem)
```

Feature Extraction

In [12]:

```
data['url_len'] = data['url'].apply(lambda x: len(str(x)))
```

In [13]:

```
def process_tld(url):
    try:
        res = get_tld(url, as_object = True, fail_silently=False,fix_protocol=True)
        pri_domain= res.parsed_url.netloc
    except :
        pri_domain= None
    return pri_domain
```

In [14]:

```
data['domain'] = data['url'].apply(lambda i: process_tld(i))
```

In [15]:

```
data.head()
```

Out[15]:

	url	type	Category	url_len	domain
0	br-icloud.com.br	phishing	2	16	br-icloud.com.br
1	mp3raid.com/music/krizz_kaliko.html	benign	0	35	mp3raid.com
2	bopsecrets.org/rexroth/cr/1.htm	benign	0	31	bopsecrets.org
3	http://garage-pirenne.be/index.php?option=com_...	defacement	1	84	garage-pirenne.be
4	http://adventure-nicaragua.net/index.php?optio...	defacement	1	235	adventure-nicaragua.net

In [16]:

```
feature = ['@','?','-','=','.','#','%','+','$','!','*',',','//']
for a in feature:
    data[a] = data['url'].apply(lambda i: i.count(a))
```

In [17]:

```
data
```

Out[17]:

	url	type	Category	url_len	domain	@	?	-	=	.	#	%	+	$	!	*	,	//
0	br-icloud.com.br	phishing	2	16	br-icloud.com.br	0	0	1	0	2	0	0	0	0	0	0	0	0
1	mp3raid.com/music/krizz_kaliko.html	benign	0	35	mp3raid.com	0	0	0	0	2	0	0	0	0	0	0	0	0
2	bopsecrets.org/rexroth/cr/1.htm	benign	0	31	bopsecrets.org	0	0	0	0	2	0	0	0	0	0	0	0	0
3	http://garage-pirenne.be/index.php?option=com_...	defacement	1	84	garage-pirenne.be	0	1	1	4	2	0	0	0	0	0	0	0	1
4	http://adventure-nicaragua.net/index.php?optio...	defacement	1	235	adventure-nicaragua.net	0	1	1	3	2	0	0	0	0	0	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
651186	xbox360.ign.com/objects/850/850402.html	phishing	2	39	xbox360.ign.com	0	0	0	0	3	0	0	0	0	0	0	0	0
651187	games.teamxbox.com/xbox-360/1860/Dead-Space/	phishing	2	44	games.teamxbox.com	0	0	2	0	2	0	0	0	0	0	0	0	0
651188	gamespot.com/xbox360/action/deadspace/	phishing	2	38	gamespot.com	0	0	0	0	1	0	0	0	0	0	0	0	0
651189	en.wikipedia.org/wiki/Dead_Space_(video_game)	phishing	2	45	en.wikipedia.org	0	0	0	0	2	0	0	0	0	0	0	0	0
651190	angelfire.com/goth/devilmaycrytonite/	phishing	2	37	angelfire.com	0	0	0	0	1	0	0	0	0	0	0	0	0

651191 rows × 18 columns

In [18]:

```
def abnormal_url(url):
    hostname = urlparse(url).hostname
    hostname = str(hostname)
    match = re.search(hostname, url)
    if match:
        # print match.group()
        return 1
    else:
        # print 'No matching pattern found'
        return 0
```

In [19]:

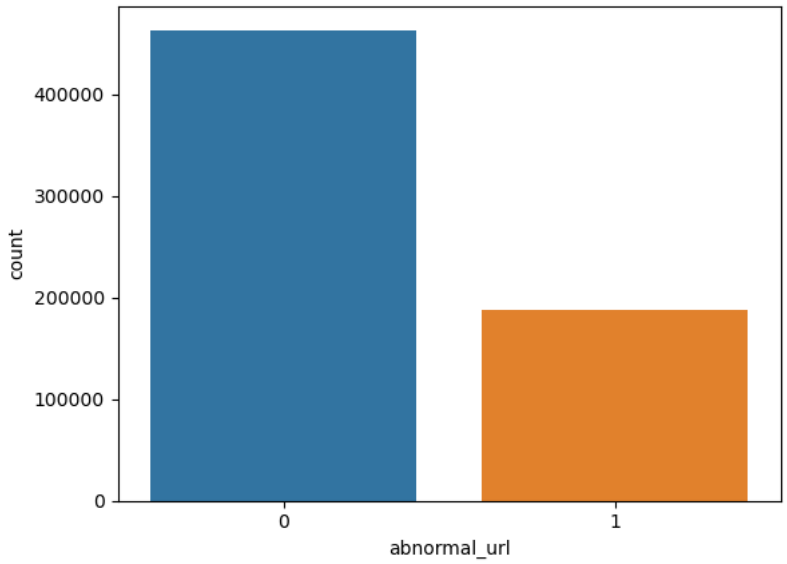
```
data['abnormal_url'] = data['url'].apply(lambda i: abnormal_url(i))
```

In [20]:

```
sns.countplot(x='abnormal_url', data=data)
```

Out[20]:

```
<AxesSubplot:xlabel='abnormal_url', ylabel='count'>
```



In [21]:

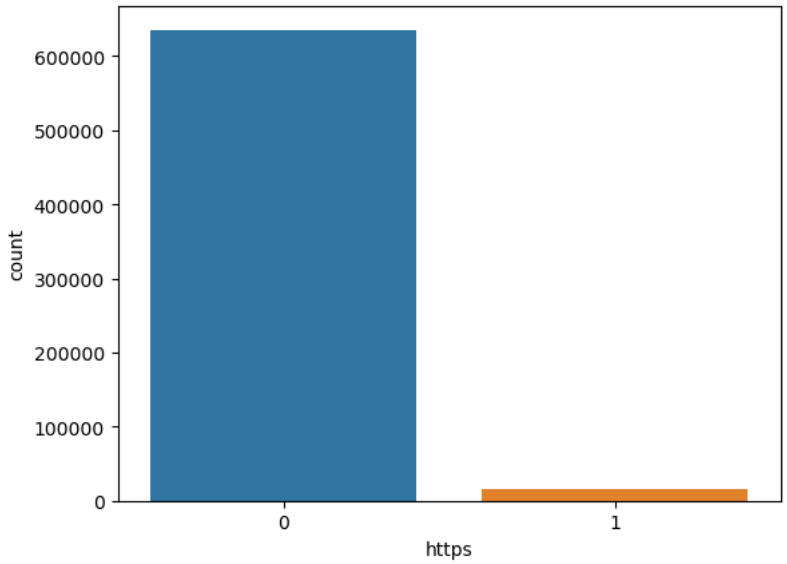
```
def httpSecure(url):
    htp = urlparse(url).scheme
    match = str(htp)
    if match=='https':
        # print match.group()
        return 1
    else:
        # print 'No matching pattern found'
        return 0
```

In [22]:

```
data['https'] = data['url'].apply(lambda i: httpSecure(i))
```

In [23]:

```
sns.countplot(x='https', data=data);
```



Counts the number of digit characters in a URL

In [24]:

```
def digit_count(url):
    digits = 0
    for i in url:
        if i.isnumeric():
            digits = digits + 1
    return digits
```

In [25]:

```
data['digits']= data['url'].apply(lambda i: digit_count(i))
```

Counts the number of letter characters in a URL

In [26]:

```
def letter_count(url):
    letters = 0
    for i in url:
        if i.isalpha():
            letters = letters + 1
    return letters
```

In [27]:

```
data['letters']= data['url'].apply(lambda i: letter_count(i))
```

Checks to see whether URL contains a shortening service

In [28]:

```
import re
def Shortining_Service(url):
    match = re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'
                      'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'
                      'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'
                      'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'
                      'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'
                      'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|'
                      'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|'
                      'tr\.im|link\.zip\.net',
                      url)
    if match:
        return 1
    else:
        return 0
```

In [29]:

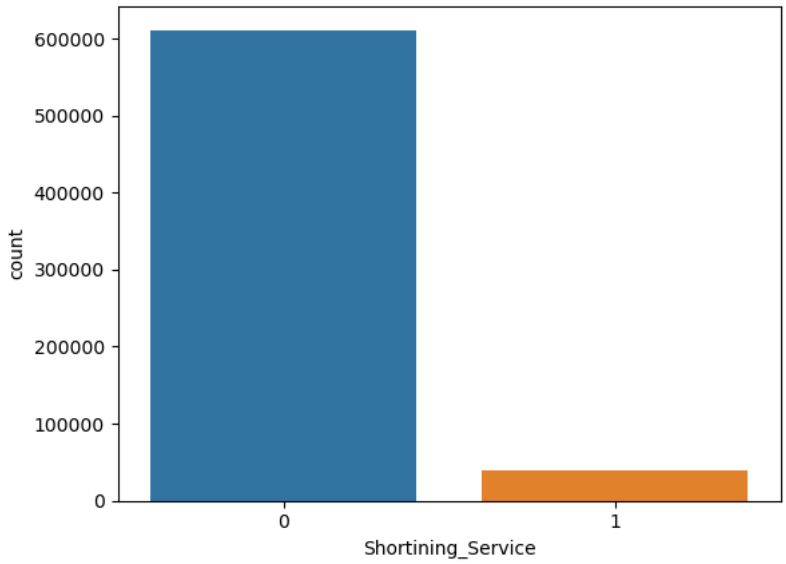
```
data['Shortining_Service'] = data['url'].apply(lambda x: Shortining_Service(x))
```

In [29]:

```
sns.countplot(x='Shortining_Service', data=data)
```

Out[29]:

```
<AxesSubplot:xlabel='Shortining_Service', ylabel='count'>
```



In [30]:

```
def having_ip_address(url):
    match = re.search(
        '(([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.'
        '([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\/)|'  # IPv4
        '(([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.'
        '([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\/)|'  # IPv4 with port
        '((0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\/)' # IPv4 in hexadecimal
        '(?:[a-fA-F0-9]{1,4}:){7}[a-fA-F0-9]{1,4}|'
        '([0-9]+(?:\.[0-9]+){3}:[0-9]+)|'
        '((?:(?:\d|[01]?\d\d|2[0-4]\d|25[0-5])\.){3}(?:25[0-5]|2[0-4]\d|[01]?\d\d|\d)(?:\/\d{1,2})?)', url)  # Ipv6
    if match:
        return 1
    else:
        return 0
```

In [31]:

```
data['having_ip_address'] = data['url'].apply(lambda i: having_ip_address(i))
```

In [32]:

```
data['having_ip_address'].value_counts()
```

Out[32]:

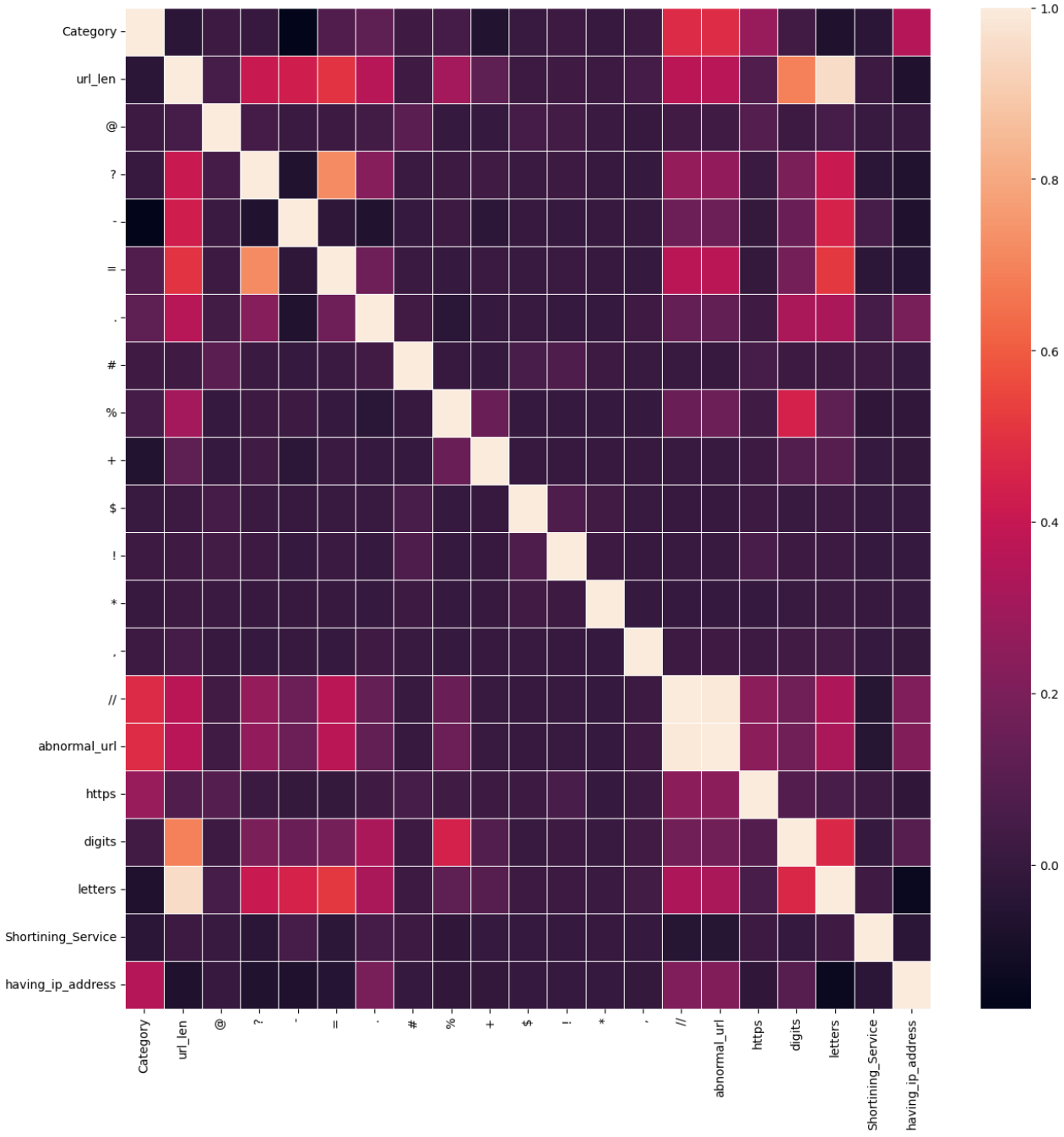
```
0    638703
1     12488
Name: having_ip_address, dtype: int64
```

In [33]:

```
plt.figure(figsize=(15, 15))
sns.heatmap(data.corr(), linewidths=.5)
```

Out[33]:

```
<AxesSubplot:>
```



In [34]:

```
X = data.drop(['url','type','Category','domain'],axis=1)#,'type_code'
y = data['Category']
```

Train & Test Split

In [35]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=2)
```

Training models

In [36]:

```
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import plot_roc_curve
```

In [ ]:

```
models = [DecisionTreeClassifier,RandomForestClassifier,AdaBoostClassifier,KNeighborsClassifier,SGDClassifier,
         ExtraTreesClassifier,GaussianNB]
accuracy_test=[]
for m in models:
    print('##############################################')
    print('######-Model =>\033[07m {} \033[0m'.format(m))
    model_ = m()
    model_.fit(X_train, y_train)
    pred = model_.predict(X_test)
    acc = accuracy_score(pred, y_test)
    accuracy_test.append(acc)
    print('Test Accuracy :\033[32m \033[01m {:.2f}% \033[30m \033[0m'.format(acc*100))
    print('\033[01m              Classification_report \033[0m')
    print(classification_report(y_test, pred))
    print('\033[01m             Confusion_matrix \033[0m')
    cf_matrix = confusion_matrix(y_test, pred)
    plot_ = sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True,fmt= '0.3%')
    plt.show()
    print('\033[31m####################- End -####################\033[0m')
```

```
##############################################
######-Model => <class 'sklearn.tree._classes.DecisionTreeClassifier'>
Test Accuracy :  90.74%
              Classification_report
              precision    recall  f1-score   support

           0       0.92      0.97      0.94    128461
           1       0.93      0.96      0.94     28882
           2       0.79      0.56      0.66     28171
           3       0.94      0.90      0.92      9844

    accuracy                           0.91    195358
   macro avg       0.89      0.85      0.87    195358
weighted avg       0.90      0.91      0.90    195358

             Confusion_matrix
```

FINAL REPORT

In [38]:

```
output = pd.DataFrame({"Model":['Decision Tree Classifier','Random Forest Classifier',
                                'AdaBoost Classifier','KNeighbors Classifier','SGD Classifier',
                                'Extra Trees Classifier','Gaussian NB'],
                       "Accuracy":accuracy_test})
```

In [39]:

```
output
```

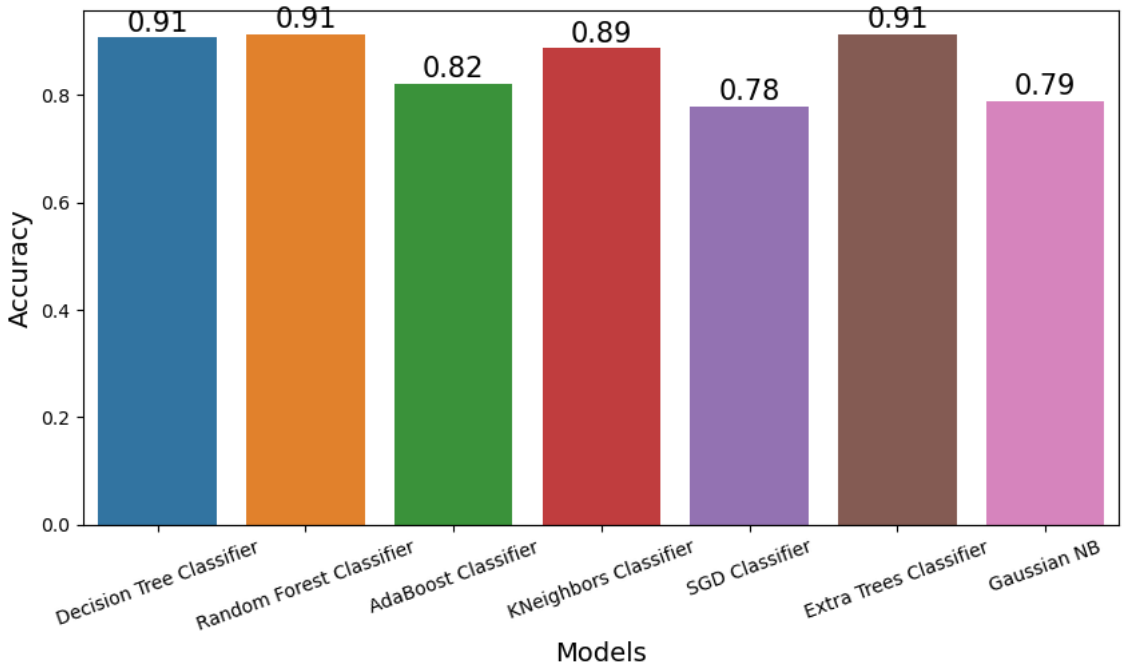
Out[39]:

	Model	Accuracy
0	Decision Tree Classifier	0.907222
1	Random Forest Classifier	0.913518
2	AdaBoost Classifier	0.821707
3	KNeighbors Classifier	0.887125
4	SGD Classifier	0.777859
5	Extra Trees Classifier	0.913323
6	Gaussian NB	0.789822

In [40]:

```
plt.figure(figsize=(10, 5))
plots = sns.barplot(x='Model', y='Accuracy', data=output)
for bar in plots.patches:
    plots.annotate(format(bar.get_height(), '.2f'),
                   (bar.get_x() + bar.get_width() / 2,
                    bar.get_height()), ha='center', va='center',
                   size=15, xytext=(0, 8),
                   textcoords='offset points')

plt.xlabel("Models", size=14)
plt.xticks(rotation=20);
plt.ylabel("Accuracy", size=14)
plt.show()
```



In [ ]: