```
import pandas as pd
import numpy as np
import  matplotlib.pyplot as plt
```

```
data=pd.read_csv("/content/CyberSecurity Data.csv")
```

```
data.head()
```

	id	having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol	double_s
0	1	-1	1	1	1	
1	2	1	1	1	1	
2	3	1	0	1	1	
3	4	1	0	1	1	
4	5	1	0	-1	1	

```
data.drop(["id"], axis = 1, inplace = True)
```

```
data.columns
```

```
Index(['having_IP_Address', 'URL_Length', 'Shortining_Service',
       'having_At_Symbol', 'double_slash_redirecting', 'Prefix_Suffix',
       'having_Sub_Domain', 'SSLfinal_State', 'Domain_registeration_length',
       'Favicon', 'port', 'HTTPS_token', 'Request_URL', 'URL_of_Anchor',
       'Links_in_tags', 'SFH', 'Submitting_to_email', 'Abnormal_URL',
       'Redirect', 'on_mouseover', 'RightClick', 'popUpWidnow', 'Iframe',
       'age_of_domain', 'DNSRecord', 'web_traffic', 'Page_Rank',
       'Google_Index', 'Links_pointing_to_page', 'Statistical_report',
       'Result'],
      dtype='object')
```

```
data.shape
```

```
(11055, 31)
```

```
data.isnull().values.any()
```

```
False
```

```
data.isnull().sum()
```

```
having_IP_Address              0
URL_Length                     0
Shortining_Service             0
having_At_Symbol               0
double_slash_redirecting       0
Prefix_Suffix                  0
```

```
having_Sub_Domain              0
SSLfinal_State                 0
Domain_registeration_length    0
Favicon                        0
port                           0
HTTPS_token                    0
Request_URL                    0
URL_of_Anchor                  0
Links_in_tags                  0
SFH                            0
Submitting_to_email            0
Abnormal_URL                   0
Redirect                       0
on_mouseover                   0
RightClick                     0
popUpWidnow                    0
Iframe                         0
age_of_domain                  0
DNSRecord                      0
web_traffic                    0
Page_Rank                      0
Google_Index                   0
Links_pointing_to_page         0
Statistical_report             0
Result                         0
dtype: int64
```

```
data=data.dropna()
```

```
y=data.Result
y.shape
```

```
(11055,)
```

```
x=data.drop('Result',axis=1)
x.shape
```

```
(11055, 30)
```

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

```
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(8844, 30)
(8844,)
(2211, 30)
(2211,)
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics

lr=LogisticRegression(random_state = 0)


lr.fit(x_train,y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=0, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
y_predict=lr.predict(x_test)


print("Train Accuracy : ",100*lr.score(x_train,y_train))
print("Test Accuracy : ",100*lr.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train Accuracy :  92.83129805517866
Test Accuracy :  92.85391225689733
              precision    recall  f1-score   support

          -1       0.93      0.91      0.92       975
           1       0.93      0.94      0.94      1236

    accuracy                           0.93      2211
   macro avg       0.93      0.93      0.93      2211
weighted avg       0.93      0.93      0.93      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier


rb=GradientBoostingClassifier()


rb.fit(x_train,y_train)
```

```
GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='deviance', max_depth=3,
                           max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=100,
                           n_iter_no_change=None, presort='deprecated',
                           random_state=None, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)
```

```
y_predict=rb.predict(x_test)
```

```
print("Train Accuracy : ",100*rb.score(x_train,y_train))
print("Test Accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
     Train Accuracy :  95.3527815468114
     Test Accuracy :  95.07010402532791
                   precision    recall  f1-score   support

              -1       0.96      0.93      0.94       975
               1       0.95      0.97      0.96      1236

        accuracy                           0.95      2211
       macro avg       0.95      0.95      0.95      2211
    weighted avg       0.95      0.95      0.95      2211
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
dt=DecisionTreeClassifier(random_state=0)
```

```
dt.fit(x_train,y_train)
```

```
     DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                            max_depth=None, max_features=None, max_leaf_nodes=None,
                            min_impurity_decrease=0.0, min_impurity_split=None,
                            min_samples_leaf=1, min_samples_split=2,
                            min_weight_fraction_leaf=0.0, presort='deprecated',
                            random_state=0, splitter='best')
```

```
y_predict=dt.predict(x_test)
```

```
print("Train Accuracy : ",100*dt.score(x_train,y_train))
print("Test Accuracy : ",100*dt.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
     Train Accuracy :  98.9484396200814
     Test Accuracy :  96.56264133876074
                   precision    recall  f1-score   support

              -1       0.96      0.96      0.96       975
               1       0.97      0.97      0.97      1236

        accuracy                           0.97      2211
       macro avg       0.97      0.96      0.97      2211
    weighted avg       0.97      0.97      0.97      2211
```

```
from sklearn.ensemble import AdaBoostClassifier
```

```
adc=AdaBoostClassifier()
```

```
adc.fit(x_train,y_train)
```

```
AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=1.0,
                   n_estimators=50, random_state=None)
```

```
print("Train Accuracy : ",100*adc.score(x_train,y_train))
print("Test Accuracy : ",100*adc.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train Accuracy :  93.66802351876979
Test Accuracy :  93.80370872908186
              precision    recall  f1-score   support

          -1       0.96      0.96      0.96       975
           1       0.97      0.97      0.97      1236

    accuracy                           0.97      2211
   macro avg       0.97      0.96      0.97      2211
weighted avg       0.97      0.97      0.97      2211
```

```
models=[lr,rb,dt,adc]
a=["Logistic Regression","Regular Boosting","Decision Tree","AdaBoost Classifire"]
names=["LR","RB","DT","ADC"]
```

```
test=[]
train=[]
for model in models:
    model.fit(x_train,y_train)
    train.append(model.score(x_train,y_train))
    test.append(model.score(x_test,y_test))
```

```
results = pd.DataFrame({ 'ML Model': a,
    'Train Accuracy': train,
    'Test Accuracy': test})
```

```
results
```

	ML Model	Train Accuracy	Test Accuracy
0	Logistic Regression	0.928313	0.928539
1	Regular Boosting	0.953528	0.950701
2	Decision Tree	0.989484	0.965626
3	AdaBoost Classifire	0.936680	0.938037

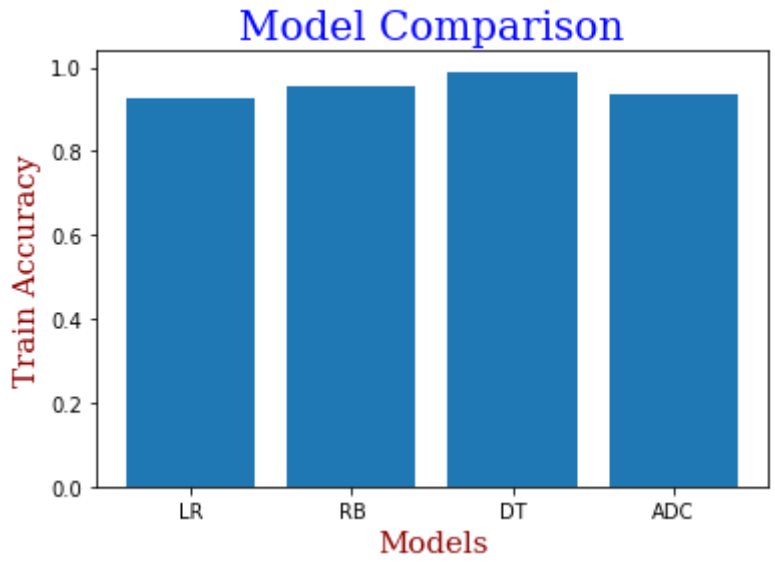
```
results.sort_values(by=['Train Accuracy','Test Accuracy'], ascending=False)
```

	ML Model	Train Accuracy	Test Accuracy
2	Decision Tree	0.989484	0.965626
1	Regular Boosting	0.953528	0.950701
3	AdaBoost Classifire	0.936680	0.938037

```
results.sort_values(by=['Test Accuracy', 'Train Accuracy'], ascending=False)
```

	ML Model	Train Accuracy	Test Accuracy
2	Decision Tree	0.989484	0.965626
1	Regular Boosting	0.953528	0.950701
3	AdaBoost Classifire	0.936680	0.938037
0	Logistic Regression	0.928313	0.928539

```
fig=plt.figure()
ax=fig.add_subplot(111)
font1 = {'family':'serif','color':'blue','size':20}
font2 = {'family':'serif','color':'darkred','size':15}
plt.xlabel('Models',fontdict = font2)
plt.ylabel('Train Accuracy',fontdict = font2)
plt.title('Model Comparison',fontdict = font1)
plt.bar(names,train)
plt.show()
```




```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.ensemble import StackingClassifier
from numpy import mean
from numpy import std


level0 = list()
level0.append(('lr', LogisticRegression()))
```

```
level0.append(('dt', DecisionTreeClassifier()))
level0.append(('adc', AdaBoostClassifier()))
level0.append(('rb', GradientBoostingClassifier()))
level1 = LogisticRegression()
model = StackingClassifier(estimators=level0, final_estimator=level1, cv=5)

models = dict()
models['Logistic'] = LogisticRegression()
models['Decision'] = DecisionTreeClassifier()
models['AdaBoost'] = AdaBoostClassifier()
models['Regular'] = GradientBoostingClassifier()
models['Stacking'] = model


def evaluate_model(model, x_train, y_train):
 cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
 scores = cross_val_score(model, x_train, y_train, scoring='accuracy', cv=cv, n_jobs=-1, e
 return scores


result, names = list(), list()
for name, model in models.items():
 scores = evaluate_model(model, x_train, y_train)
 result.append(scores)
 names.append(name)
 print('>%s %.3f (%.3f)' % (name, mean(scores), std(scores)))
```

```
>Logistic 0.927 (0.010)
>Decision 0.959 (0.007)
>AdaBoost 0.937 (0.008)
>Regular 0.949 (0.009)
>Stacking 0.964 (0.006)
```

✓ 3m 26s completed at 1:54 PM