In [1]:

```python
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
import numpy as np
import pandas as pd
import seaborn as sns
import statsmodels.api as sn
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
```

In [2]:

```python
df = pd.read_csv('D:/reseach article/articleshort.csv',header =0)
dff=df.dropna()
#basic insights
df.head()
```

Out[2]:

	Q207AAYes	Q201Yes	Q208	Q412AYes	Q412CYes	Q501QNo	Q510Yes
count	5771.000000	5771.000000	5771.000000	5771.000000	5771.000000	5771.000000	5771.000000
mean	0.288858	0.993415	3.798129	0.910241	0.755848	0.204471	0.263559
std	0.453271	0.080885	2.492269	0.285861	0.429620	0.403349	0.440601
min	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	1.000000	2.000000	1.000000	1.000000	0.000000	0.000000
50%	0.000000	1.000000	3.000000	1.000000	1.000000	0.000000	0.000000
75%	1.000000	1.000000	5.000000	1.000000	1.000000	0.000000	1.000000
max	1.000000	1.000000	16.000000	1.000000	1.000000	1.000000	1.000000

In [3]:

```python
X = df.drop("Q207AAYes",axis=1)
Y = df["Q207AAYes"]
```

In [4]:
```
validation_size = 0.20
seed = 72
from sklearn.model_selection import train_test_split
X_train, X_validation, Y_train, Y_validation = train_test_split(X, Y,test_size=va
print(Y_train)
```

```
4378    0
140     0
4746    1
421     0
920     0
       ..
3951    0
2885    0
3941    0
5166    0
4568    0
Name: Q207AAYes, Length: 4616, dtype: int64
```

In [5]:
```
from sklearn.svm import LinearSVC
from sklearn.neighbors import  RadiusNeighborsClassifier
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.naive_bayes import BernoulliNB
from sklearn.tree import ExtraTreeClassifier
models = []
models.append(('LR', LogisticRegression(C=0.23357214690901212, penalty='l1', solv
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
models.append(('BNB', BernoulliNB()))
models.append(('passive',PassiveAggressiveClassifier()))
models.append(('RNC', RadiusNeighborsClassifier(radius=8.0)))
models.append(('ETC',ExtraTreeClassifier()))
```

In [6]:

```
num_folds = 100
seed = 72
scoring = 'accuracy'
results = []
names = []
print('Mean and Standard Deviation accuracy with 10 folds')
for name, model in models:
    kfold = KFold(n_splits=num_folds,shuffle=True,random_state=seed)
    cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring=scori
    results.append(cv_results)
    names.append(name)
    print('{}: {} ({})'.format(name, cv_results.mean(), cv_results.std()))
```

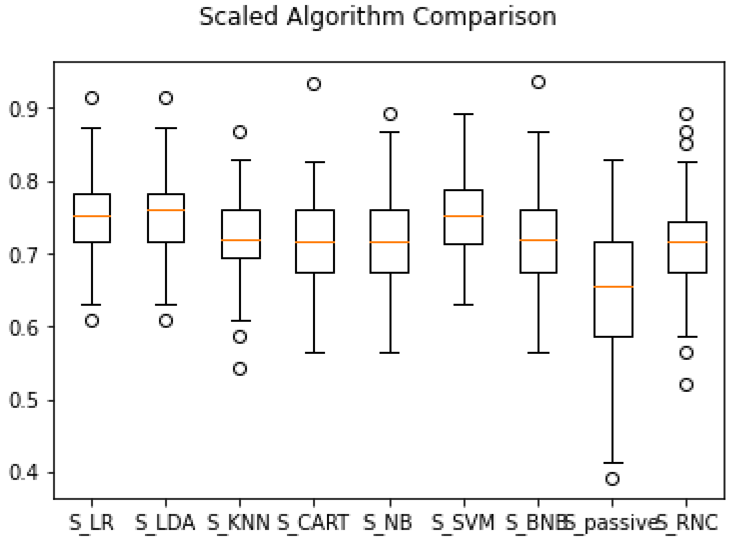
```
Mean and Standard Deviation accuracy with 10 folds
LR: 0.7497271045328401 (0.06448195086190095)
LDA: 0.7510360777058278 (0.0658319506768492)
KNN: 0.7280619796484735 (0.06500783987364349)
CART: 0.7198519888991675 (0.06976973229184091)
NB: 0.7176827012025904 (0.06468331104405067)
SVM: 0.7473311748381128 (0.06319533642140379)
BNB: 0.712918593894542 (0.06742017261495158)
passive: 0.6738020351526365 (0.12144265795856705)
RNC: 0.7139916743755782 (0.06979522417234571)
ETC: 0.7204995374653099 (0.07059194924149655)
```

In [19]:

```
pipelines = []
pipelines.append(( 'S_LR' , Pipeline([( 'Scaler' , StandardScaler()),( 'LR' ,
    LogisticRegression())])))
pipelines.append(( 'S_LDA' , Pipeline([( 'Scaler' , StandardScaler()),( 'LDA' ,
    LinearDiscriminantAnalysis())])))
pipelines.append(( 'S_KNN' , Pipeline([( 'Scaler' , StandardScaler()),( 'KNN' ,
    KNeighborsClassifier())])))
pipelines.append(( 'S_CART' , Pipeline([( 'Scaler' , StandardScaler()),( 'CART' ,
    DecisionTreeClassifier())])))
pipelines.append(( 'S_NB' , Pipeline([( 'Scaler' , StandardScaler()),( 'NB' ,
    GaussianNB())])))
pipelines.append(( 'S_SVM' , Pipeline([( 'Scaler' , StandardScaler()),( 'SVM' ,
    SVC())])))
pipelines.append(( 'S_BNB' , Pipeline([( 'Scaler' , StandardScaler()),( 'BNB' , B
pipelines.append(( 'S_passive' , Pipeline([( 'Scaler' , StandardScaler()),( 'pass
pipelines.append(( 'S_RNC' , Pipeline([( 'Scaler' , StandardScaler()),( 'RNC' , R
results = []
names = []
print("Mean and Standard Deviation Accuracy with 10 folds ")
for name, model in pipelines:
    kfold = KFold(n_splits=num_folds)
    cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring=scori
    results.append(cv_results)
    names.append(name)
    print('{}: {} ,  {}'.format(name, cv_results.mean(), cv_results.std()))
```

```
Mean and Standard Deviation Accuracy with 10 folds
S_LR: 0.7503885291396856 ,  0.06183769556077344
S_LDA: 0.7516790009250696 ,  0.06065625874021685
S_KNN: 0.7246808510638298 ,  0.057735599542594246
S_CART: 0.7181036077705827 ,  0.06052500758106497
S_NB: 0.7185013876040705 ,  0.06716102450070766
S_SVM: 0.747571692876966 ,  0.059222374386819154
S_BNB: 0.7187511563367253 ,  0.06610181695036804
S_passive: 0.6500000000000001 ,  0.09317727752640827
S_RNC: 0.7126965772432934 ,  0.06344139618691282
```

In [20]:
```
from matplotlib import pyplot
fig = pyplot.figure()
fig.suptitle( 'Scaled Algorithm Comparison' )
ax = fig.add_subplot(111)
pyplot.boxplot(results)
ax.set_xticklabels(names)
pyplot.show()
```

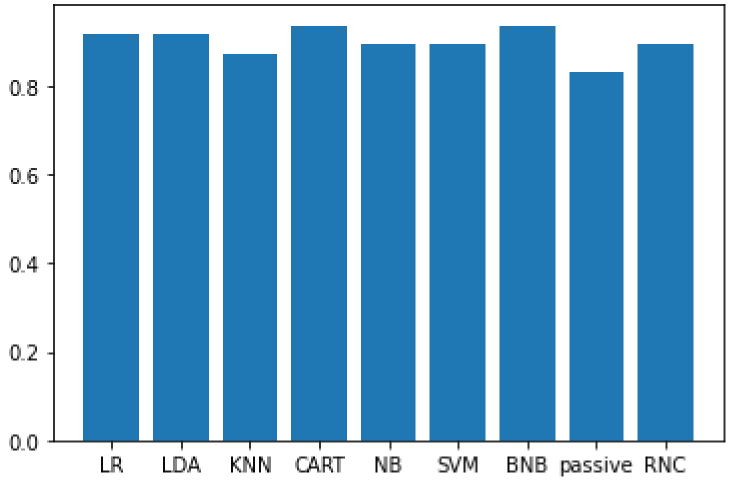


In [22]:
```
aa=list(map(max,results))
list(map(max,results))
```

Out[22]:
```
[0.9148936170212766,
 0.9148936170212766,
 0.8695652173913043,
 0.9347826086956522,
 0.8936170212765957,
 0.8936170212765957,
 0.9361702127659575,
 0.8297872340425532,
 0.8936170212765957]
```

In [23]:

```
# Libraries
import numpy as np
import matplotlib.pyplot as plt

# Make a random dataset:

bars = ('LR', 'LDA', 'KNN', 'CART', 'NB','SVM','BNB','passive','RNC')
y_pos = np.arange(len(bars))

# Create bars
plt.bar(y_pos, aa)

# Create names on the x-axis
plt.xticks(y_pos, bars)

# Show graphic
plt.show()
```


LR
LDA
KNN
CART
NB
SVM
BNB
passive
RNC