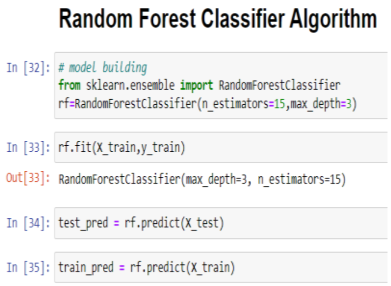
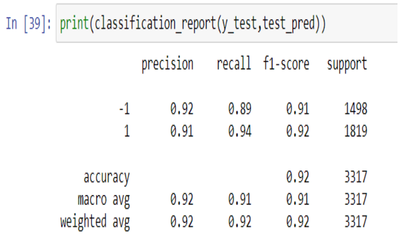
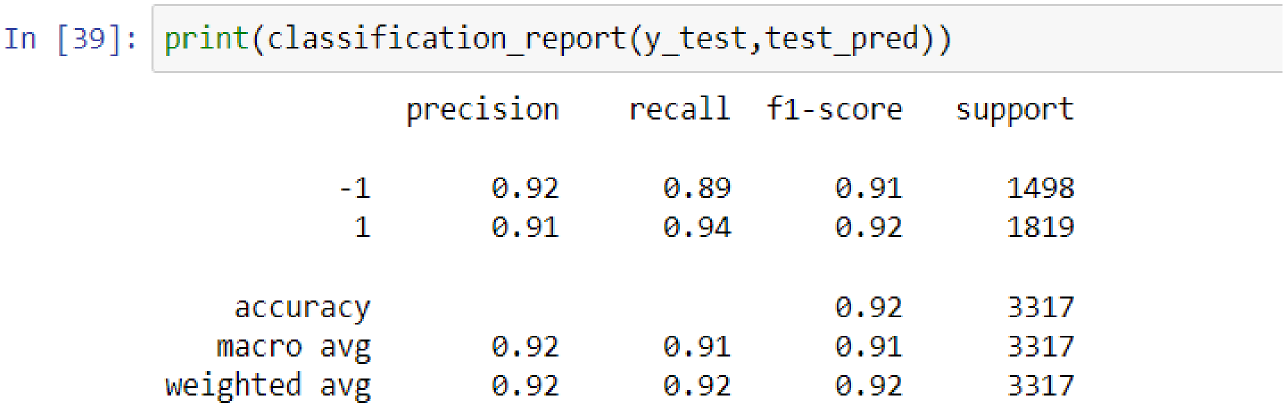
PERFORMANCE TESTING

Project Name	WEB PHISHING DETECTION
Team ID	PNT2022TMID33847

Model Performance Testing:

S. No	Parameter	Values	Screenshot
1.	Metrics	**Classification Model:** Random Forest Classifier. **Evaluation Metrics:** Confusion Matrix - Accuracy Score- & Classification Report.	 Random Forest Classifier Algorithm In [32]: # model building from sklearn.ensemble import RandomForestClassifier rf=RandomForestClassifier(n_estimators=15,max_depth=3) In [33]: rf.fit(X_train,y_train) Out[33]: RandomForestClassifier(max_depth=3, n_estimators=15) In [34]: test_pred = rf.predict(X_test) In [35]: train_pred = rf.predict(X_train)  In [39]: print(classification_report(y_test,test_pred)) precision recall f1-score support -1 0.92 0.89 0.91 1498 1 0.91 0.94 0.92 1819 accuracy 0.92 3317 macro avg 0.92 0.91 0.91 3317 weighted avg 0.92 0.92 0.92 3317

EVALUATION METRICS:



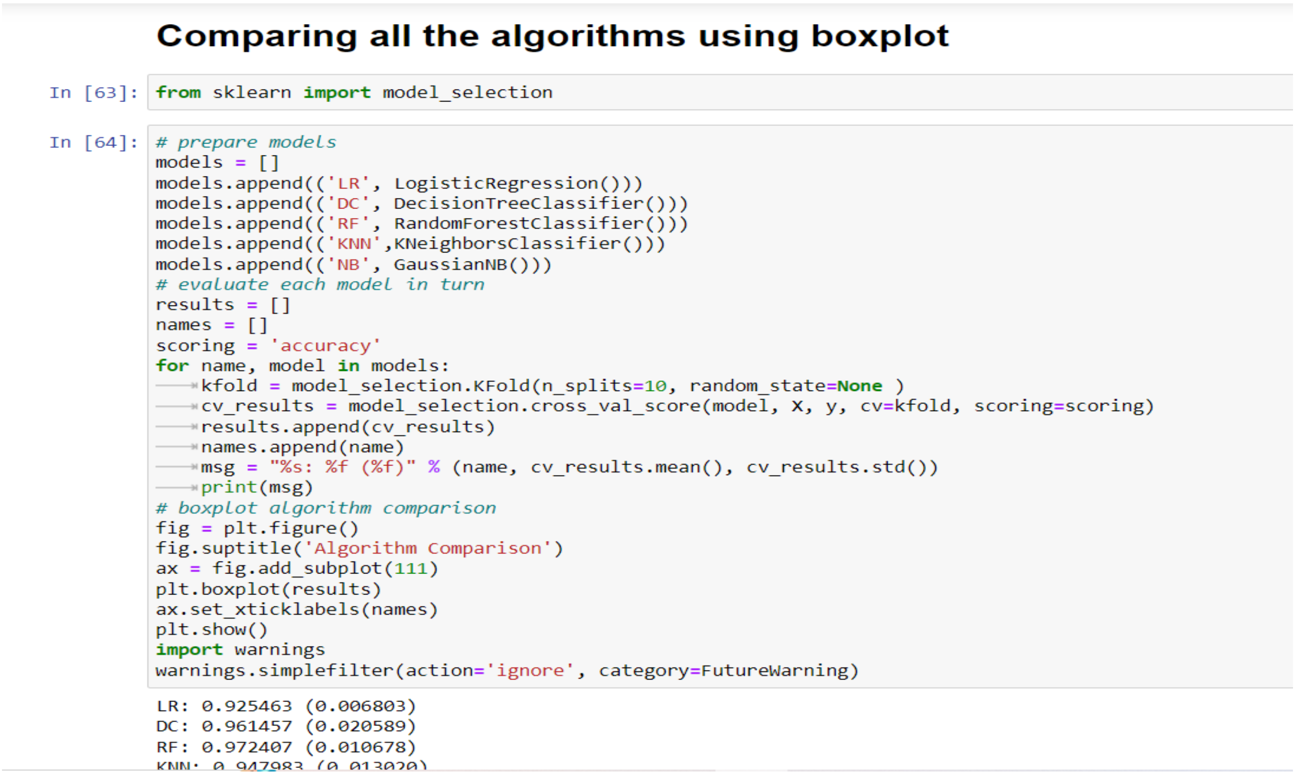
```
In [39]: print(classification_report(y_test,test_pred))

              precision    recall  f1-score   support

          -1       0.92      0.89      0.91      1498
           1       0.91      0.94      0.92      1819

    accuracy                           0.92      3317
   macro avg       0.92      0.91      0.91      3317
weighted avg       0.92      0.92      0.92      3317
```

COMPARING PERFORMANCE OF DIFFERENT ML MODELS:

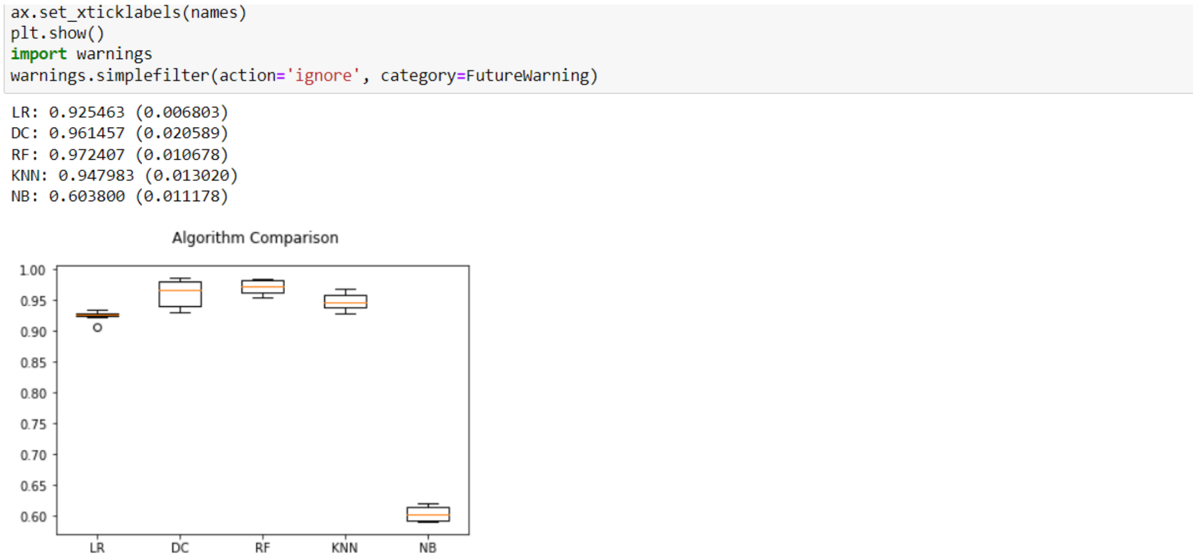


Comparing all the algorithms using boxplot

In [63]:
```
from sklearn import model_selection
```

In [64]:
```
# prepare models
models = []
models.append(('LR', LogisticRegression()))
models.append(('DC', DecisionTreeClassifier()))
models.append(('RF', RandomForestClassifier()))
models.append(('KNN',KNeighborsClassifier()))
models.append(('NB', GaussianNB()))
# evaluate each model in turn
results = []
names = []
scoring = 'accuracy'
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=None )
    cv_results = model_selection.cross_val_score(model, X, y, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
# boxplot algorithm comparison
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

```
LR: 0.925463 (0.006803)
DC: 0.961457 (0.020589)
RF: 0.972407 (0.010678)
KNN: 0.947983 (0.013020)
```



```
ax.set_xticklabels(names)
plt.show()
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

```
LR: 0.925463 (0.006803)
DC: 0.961457 (0.020589)
RF: 0.972407 (0.010678)
KNN: 0.947983 (0.013020)
NB: 0.603800 (0.011178)
```

FROM BOX PLOT RESULTS IT IS CLEAR THAT 'RANDOM FOREST CLASSIFIER' HAS THE GREATEST ACCURACY. 'RANDOM FOREST CLASSIFIER' ALGORITHM IS THE BEST MACHINE LEARNING ALGORITHM. FITTING THE BEST MODEL.