In [1]:
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
```

In [59]:
```python
df = pd.read_csv('C:\\Users\\sahil\\OneDrive\\Desktop\\Fall 2020\\Security Analytics\\wc1-roc.csv')
pd.set_option('display.max_columns', None)
df.head()
#1 -> Alexa and 0 -> phish
```

Out[59]:

	inline_count	external_count	onclick_count	onload_count	onchange_count	avg_inline_script_block	avg_external_script_block	avg_onc
0	21.0	23.0	1	131.0	0.0	0.0	662.062500	
1	13.0	30.0	2	4.0	1.0	0.0	55.777778	
2	0.0	3.0	1	0.0	0.0	0.0	207.000000	
3	21.0	11.0	1	10.0	1.0	0.0	104.800000	
4	10.0	5.0	1	0.0	0.0	0.0	473.000000	

In [3]:
```python
X_train, X_test, y_train, y_test = train_test_split(
    df.drop('type', axis=1), df['type'],
    test_size=0.33, random_state=133)
```

In [37]:
```python
# Initialize and train classifier model
clf1 = LogisticRegression().fit(X_train, y_train)

# Make predictions on test set
y_pred1 = clf1.predict(X_test)
y_score1 = clf1.predict_proba(X_test)[:,1]

print(accuracy_score(y_pred1, y_test))
print(confusion_matrix(y_test, y_pred1))
```

```
0.8105539577341503
[[ 950 1242]
 [ 273 5532]]
```

```
C:\Users\sahil\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: ConvergenceWarning:
lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

In [38]:
```python
#coef. of LR
df.columns
print(clf1.coef_[0])
```

```
[ 9.37131471e-03  2.87062772e-02 -5.23397237e-02  7.16068137e-03
 -3.94834331e-02 -3.49383416e-04  8.35559335e-04 -1.08404788e-05
 -3.71928654e-02  9.66807556e-02  1.64175196e-02  1.06295850e-01]
```

In [14]:
```python
clf2=RandomForestClassifier(n_estimators=200)
clf2.fit(X_train,y_train)

# Make predictions on test set
y_pred2 = clf2.predict(X_test)

print(accuracy_score(y_pred2, y_test))
print(confusion_matrix(y_test, y_pred2))
```

```
0.9200950356383644
[[1833  359]
 [ 280 5525]]
```

In [15]:
```python
#Important features found using Random Forest Classifier
clf2.feature_importances_
```

Out[15]:
```
array([0.08393345, 0.10609362, 0.02089688, 0.03343237, 0.01115627,
       0.00672596, 0.15811858, 0.25810467, 0.05272282, 0.13002643,
       0.06346408, 0.07532486])
```

In [21]:
```python
from sklearn.neural_network import MLPClassifier

clf3 = MLPClassifier(max_iter=600) #This setting will be used.
clf3.fit(X_train,y_train)

y_pred3 = clf3.predict(X_test)

print(accuracy_score(y_pred3, y_test))
print(confusion_matrix(y_test, y_pred3))
```

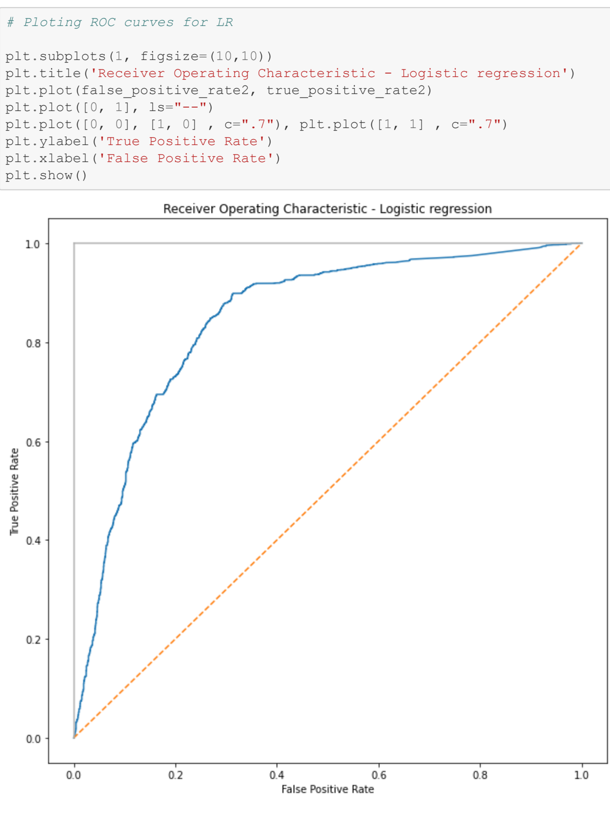
```
0.853445041890709
[[1403  789]
 [ 383 5422]]
```

In [24]:
```python
"""1.1 ROC curve for LR"""
false_positive_rate2, true_positive_rate2, threshold2 = roc_curve(y_test, y_score1)
print('roc_auc_score for Logistic Regression: ', roc_auc_score(y_test, y_score1))
```

```
roc_auc_score for Logistic Regression:  0.844101092690042
```

In [25]:
```python
# Ploting ROC curves for LR

plt.subplots(1, figsize=(10,10))
plt.title('Receiver Operating Characteristic - Logistic regression')
plt.plot(false_positive_rate2, true_positive_rate2)
plt.plot([0, 1], ls="--")
plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



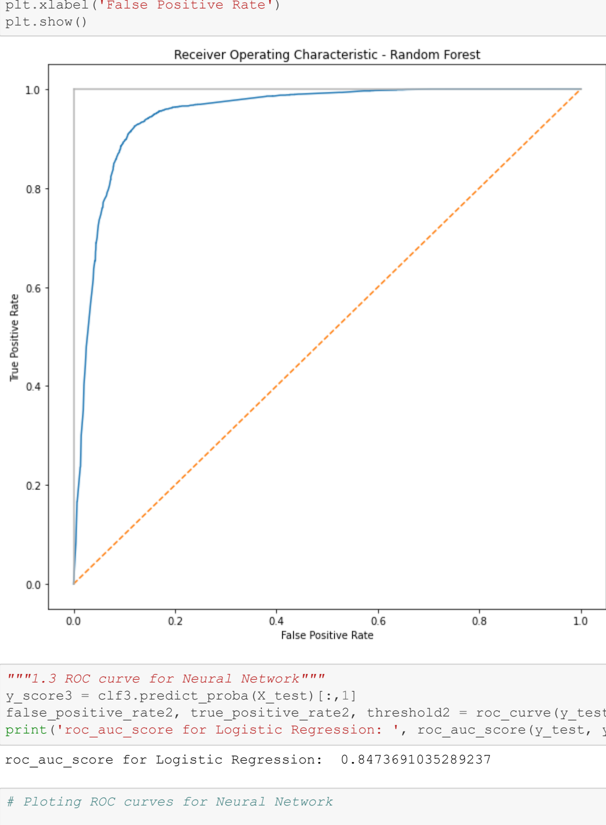
In [26]:
```python
"""1.2 ROC curve for Random Forest"""
y_score2 = clf2.predict_proba(X_test)[:,1]
false_positive_rate2, true_positive_rate2, threshold2 = roc_curve(y_test, y_score2)
print('roc_auc_score for Logistic Regression: ', roc_auc_score(y_test, y_score2))

"""Gives the best result"""
```

```
roc_auc_score for Logistic Regression:  0.9496205762713996
```

In [27]:
```python
# Ploting ROC curves for Random Forest

plt.subplots(1, figsize=(10,10))
plt.title('Receiver Operating Characteristic - Random Forest')
plt.plot(false_positive_rate2, true_positive_rate2)
plt.plot([0, 1], ls="--")
plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

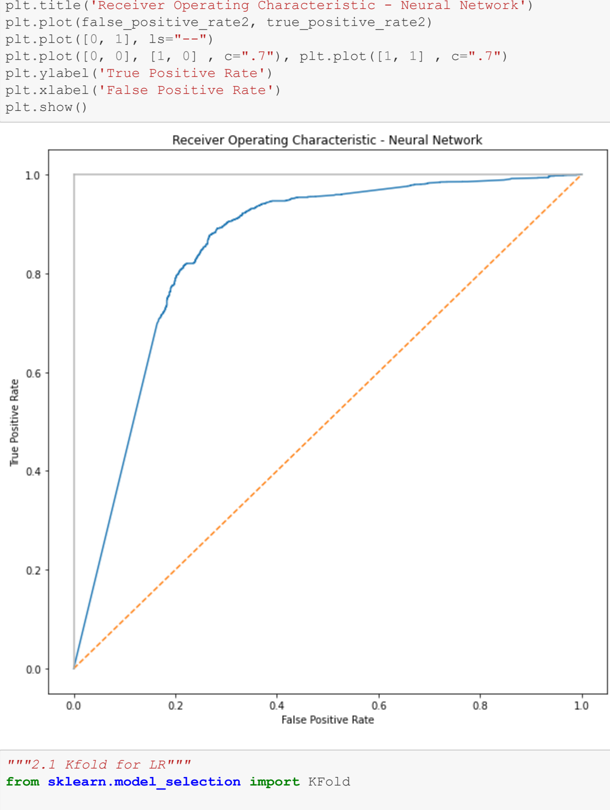


In [28]:
```python
"""1.3 ROC curve for Neural Network"""
y_score3 = clf3.predict_proba(X_test)[:,1]
false_positive_rate2, true_positive_rate2, threshold2 = roc_curve(y_test, y_score3)
print('roc_auc_score for Logistic Regression: ', roc_auc_score(y_test, y_score3))
```

```
roc_auc_score for Logistic Regression:  0.8473691035289237
```

In [29]:
```python
# Ploting ROC curves for Neural Network

plt.subplots(1, figsize=(10,10))
plt.title('Receiver Operating Characteristic - Neural Network')
plt.plot(false_positive_rate2, true_positive_rate2)
plt.plot([0, 1], ls="--")
plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



In [34]:
```python
"""2.1 Kfold for LR"""
from sklearn.model_selection import KFold

model_accuracy = []
kf = KFold(n_splits=10)
#split train data to train and validation
for train, val in kf.split(X_train, y_train):
    clf = LogisticRegression(max_iter=10000).fit(X_train.iloc[train], y_train.iloc[train])
    y_pred = clf.predict(X_train.iloc[val])
    model_accuracy.append({'model': clf, 'acc': accuracy_score(y_pred, y_train.iloc[val])})
    print(accuracy_score(y_pred, y_train.iloc[val]))

#choosing the highest accuracy model
clf_best = max(model_accuracy, key = lambda x: x['acc'])['model']

#run it on the test set
y_test_pred = clf_best.predict(X_test)
print('test accuracy:',accuracy_score(y_test, y_test_pred))
```

```
0.8442118226600985
0.8251231527093597
0.8349753694581281
0.8257389162561576
0.8275862068965517
0.8330252618607517
0.8170055452865065
0.8280961182994455
0.8299445471349353
0.8373382624768947
test accuracy: 0.817806677504064
```

In [35]:
```python
"""2.2 Kfold for RF"""
model_accuracy = []
kf = KFold(n_splits=10)
#split train data to train and validation
for train, val in kf.split(X_train, y_train):
    clf = RandomForestClassifier(n_estimators=200).fit(X_train.iloc[train], y_train.iloc[train])
    y_pred = clf.predict(X_train.iloc[val])
    model_accuracy.append({'model': clf, 'acc': accuracy_score(y_pred, y_train.iloc[val])})
    print(accuracy_score(y_pred, y_train.iloc[val]))

#choosing the highest accuracy model
clf_best = max(model_accuracy, key = lambda x: x['acc'])['model']

#run it on the test set
y_test_pred = clf_best.predict(X_test)
print('test accuracy:',accuracy_score(y_test, y_test_pred))
"""Gives the best result"""
```

```
0.9334975369458128
0.9144088669950738
0.9261083743842364
0.9131773399014779
0.9217980295566502
0.9242144177449169
0.906962415280345
0.9217498459642637
0.9180529882932841
0.9168207024029574
test accuracy: 0.9170939102163311
```

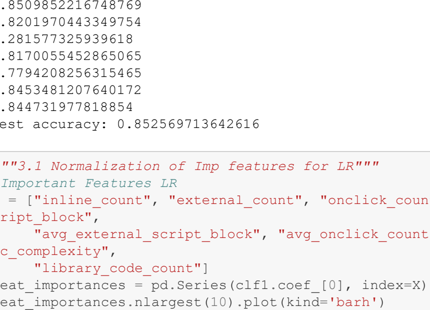
In [36]:
```python
"""2.3 Kfold for NN"""
model_accuracy = []
kf = KFold(n_splits=10)
#split train data to train and validation
for train, val in kf.split(X_train, y_train):
    clf = MLPClassifier(max_iter=600).fit(X_train.iloc[train], y_train.iloc[train])
    y_pred = clf.predict(X_train.iloc[val])
    model_accuracy.append({'model': clf, 'acc': accuracy_score(y_pred, y_train.iloc[val])})
    print(accuracy_score(y_pred, y_train.iloc[val]))

#choosing the highest accuracy model
clf_best = max(model_accuracy, key = lambda x: x['acc'])['model']

#run it on the test set
y_test_pred = clf_best.predict(X_test)
print('test accuracy:',accuracy_score(y_test, y_test_pred))
```

```
0.8620689655172413
0.833743842364532
0.8639162561576355
0.8509852216748769
0.8201970443349754
0.281577325939618
0.8170055452865065
0.7794208256315465
0.8453481207640172
0.844731977818854
test accuracy: 0.852569713642616
```

In [40]:
```python
"""3.1 Normalization of Imp features for LR"""
#Important Features LR
X = ["inline_count", "external_count", "onclick_count", "onload_count", "onchange_count", "avg_inline_s
cript_block",
     "avg_external_script_block", "avg_onclick_count", "avg_onload_count", "avg_onchange_count", "avg_c
yc_complexity",
     "library_code_count"]
feat_importances = pd.Series(clf1.coef_[0], index=X)
feat_importances.nlargest(10).plot(kind='barh')
plt.show()
```



In [66]:
```python
cols_to_normalize = ["external_count", "avg_onload_count", "avg_cyc_complexity",
                     "avg_onchange_count", "library_code_count"]
df1=df
df1[cols_to_normalize]=(df1[cols_to_normalize]-df1[cols_to_normalize].min())/(df1[cols_to_normalize].ma
x()-df1[cols_to_normalize].min())
df1.sample(10)
```

Out[66]:

	inline_count	external_count	onclick_count	onload_count	onchange_count	avg_inline_script_block	avg_external_script_block	avg
9183	3.0	0.011385	1	0.0	0.0	0.0	34.000000	
15964	15.0	0.028463	1	0.0	0.0	0.0	3285.000000	
2147	6.0	0.015180	2	0.0	1.0	0.0	434.250000	
22583	2.0	0.010436	2	0.0	0.0	0.0	664.166667	
23289	14.0	0.043643	1	0.0	11.0	0.0	316.833333	
22177	1.0	0.007590	2	0.0	0.0	0.0	60.000000	
5906	9.0	0.013283	2	4.0	0.0	0.0	194.800000	
22098	8.0	0.026565	2	0.0	0.0	0.0	132.625000	
11664	8.0	0.020873	1	1.0	3.0	0.0	1593.500000	
5032	1.0	0.013283	2	0.0	0.0	0.0	98.250000	

In [67]:
```python
# Split dataset up into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    df1.drop('type', axis=1), df1['type'],
    test_size=0.33, random_state=17)

# Initialize and train classifier model
clf = LogisticRegression(max_iter=3000).fit(X_train, y_train)

# Make predictions on test set
y_pred = clf.predict(X_test)
y_score2 = clf.predict_proba(X_test)[:,1]

# Compare test set predictions with ground truth labels
print(accuracy_score(y_pred, y_test))
print(confusion_matrix(y_test, y_pred))

"""Observed increase in the accuracy of the model"""
```

```
0.820057521570589
[[1232  945]
 [ 494 5326]]
```

In [76]:
```python
"""3.2 RF using only imp features"""
#Random Forest using only Important feaatures
feat_labels = ["inline_count", "external_count", "onclick_count", "onload_count", "onchange_count", "av
g_inline_script_block",
     "avg_external_script_block", "avg_onclick_count", "avg_onload_count", "avg_onchange_count", "avg_c
yc_complexity",
     "library_code_count"]

for feature in zip(feat_labels, clf2.feature_importances_):
    print(feature)
```

```
('inline_count', 0.08393345283504508)
('external_count', 0.10609361982432518)
('onclick_count', 0.02089688071224046)
('onload_count', 0.0334323729651518)
('onchange_count', 0.011156272556183396)
('avg_inline_script_block', 0.006725956566494705)
('avg_external_script_block', 0.1581185827778195)
('avg_onclick_count', 0.2581046677364175)
('avg_onload_count', 0.05272282170785226)
('avg_onchange_count', 0.1300264297686832)
('avg_cyc_complexity', 0.06346407755049038)
('library_code_count', 0.0753248649992966)
```

In [77]:
```python
from sklearn.feature_selection import SelectFromModel

sfm = SelectFromModel(clf2, threshold=0.05).fit(X_train, y_train)
```

In [78]:
```python
feat_labels = ["inline_count", "external_count", "onclick_count", "onload_count", "onchange_count", "av
g_inline_script_block",
     "avg_external_script_block", "avg_onclick_count", "avg_onload_count", "avg_onchange_count", "avg_c
yc_complexity",
     "library_code_count"]
for feature_list_index in sfm.get_support(indices=True):
    print(feat_labels[feature_list_index])
```

```
inline_count
external_count
avg_external_script_block
avg_onclick_count
avg_onload_count
avg_onchange_count
avg_cyc_complexity
library_code_count
```

In [83]:
```python
# Transform the data to create a new dataset containing only the most important features
X_important_train = sfm.transform(X_train)
X_important_test = sfm.transform(X_test)

clf_important = RandomForestClassifier(n_estimators=200).fit(X_important_train, y_train)

y_important_pred = clf_important.predict(X_important_test)
accuracy_score(y_test, y_important_pred)
"""The accuracy of the new RF model is less than but close to the accuracy of the old model but the cos
t of the new model is
much lesser than that of the old model as the number of features in the second model are only 8 whereas
 the older model had 12
features"""
```

Out[83]:
```
0.9189696136051019
```