In [1]:

```
# In[1]:


#Import environments:

import pandas as pd
import numpy as np
import sklearn
import os
from sklearn import linear_model
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, roc_curve
import seaborn as sns
import matplotlib.pyplot as plt
```

In [6]:

```
# In[ ]:


#Load Data:

airport = pd.read_excel(os.path.expanduser("/Users/carlosalvarez/Desktop/12Months.x
```

In [7]:

```
# In[ ]:


airport.shape
```

Out[7]:

```
(100000, 20)
```

In [8]:

```
# In[ ]:


airport.isnull().values.any()
```

Out[8]:

```
True
```

In [9]:

```
# In[ ]:


airport.isnull().sum()
```

Out[9]:

```
YEAR                    0
QUARTER                 0
MONTH                   0
DAY_OF_MONTH            0
ORIGIN                  0
ORIGIN_CITY_NAME        0
ORIGIN_STATE_ABR        0
DEST                    0
DEST_CITY_NAME          0
DEST_STATE_ABR          0
CRS_DEP_TIME            0
DEP_TIME             1258
DEP_DELAY            1511
DEP_DEL15            1511
CRS_ARR_TIME            0
ARR_TIME             1350
ARR_DELAY            1639
ARR_DEL15            1639
CANCELLED               0
DISTANCE                0
dtype: int64
```

In [10]:

```
# In[ ]:


columns_to_drop = ["ORIGIN_STATE_ABR","DEST_CITY_NAME","DEST_STATE_ABR","ORIGIN_CITY
```

In [11]:

```
# In[ ]:


airport.fillna(value=0, inplace=True)
airport.isnull().values.any()
```

Out[11]:

```
False
```

In [ ]:

```

```

In [12]:

```

# In[ ]:


airport = pd.get_dummies(airport, columns=['ORIGIN','DEST'])
airport.head()
```

Out[12]:

	YEAR	QUARTER	MONTH	DAY_OF_MONTH	ORIGIN_CITY_NAME	ORIGIN_STATE_ABR	DEST_
0	2018	1	1	27	Fort Lauderdale, FL	FL	
1	2018	1	1	27	Seattle, WA	WA	San F
2	2018	1	1	27	Washington, DC	VA	
3	2018	1	1	27	Los Angeles, CA	CA	
4	2018	1	1	27	Jacksonville, FL	FL	

5 rows × 403 columns

In [13]:

```
# In[ ]:

airport.drop(labels=columns_to_drop, axis=1, inplace=True)
train_x, test_x, train_y, test_y = train_test_split(airport.drop('ARR_DEL15', axis=
```

In [14]:

```

# In[ ]:


train_x.shape
```

Out[14]:

```
(80000, 398)
```

In [15]:

```
# In[ ]:


test_x.shape
```

Out[15]:

```
(20000, 398)
```

In [271]:

```
# In[ ]:

gboost = GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.085, loss='deviance', max_depth=5,
              max_features=8, 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='auto', random_state=None,
              subsample=1.0, tol=0.0001, validation_fraction=0.1,
              verbose=0, warm_start=False)
gboost.fit(train_x, train_y)
```

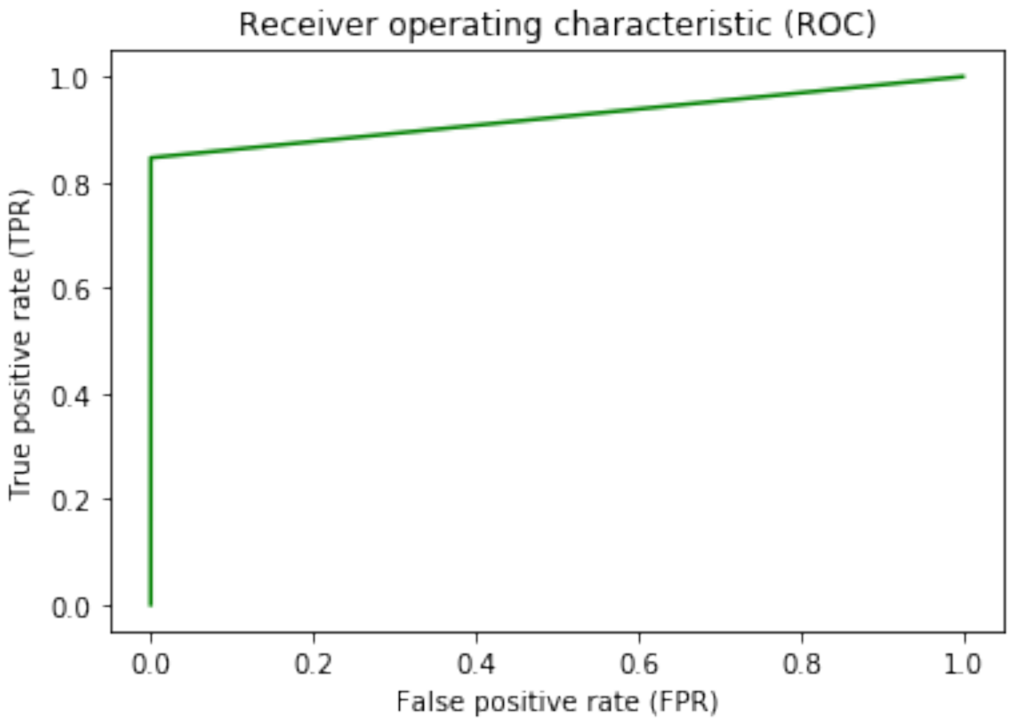
Out[271]:

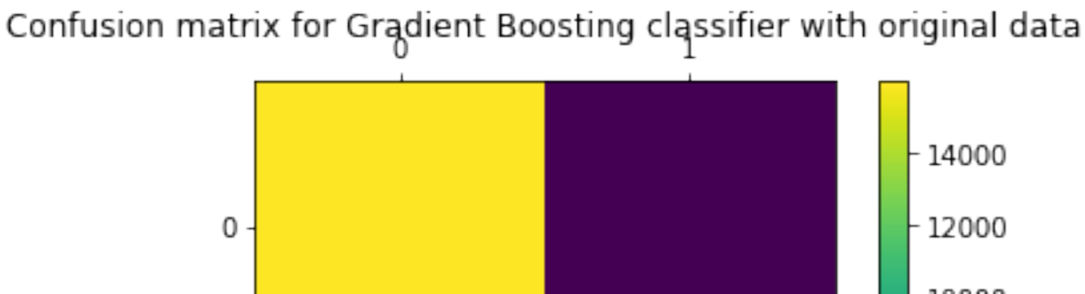
```
GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.085, loss='deviance', max_depth=5,
              max_features=8, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, n_estimators=100,
              n_iter_no_change=None, presort='auto', random_state=None
,
              subsample=1.0, tol=0.0001, validation_fraction=0.1,
              verbose=0, warm_start=False)
```

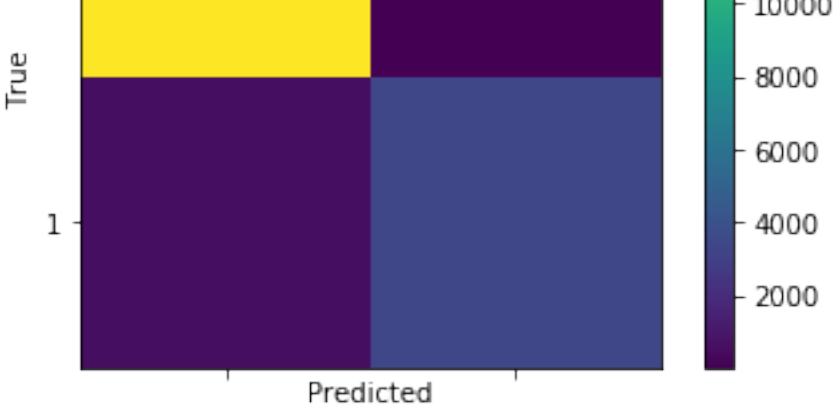
In [272]:

```
# In[ ]:


y_gboost_pred = gboost.predict(test_x)

labels = [0, 1]
cm = confusion_matrix(test_y, y_gboost_pred,labels)

gboost_accuracy = str(np.round(100*float(cm[0][0]+cm[1][1])/float((cm[0][0]+cm[1][1
gboost_recall = str(np.round(100*float((cm[1][1]))/float((cm[1][0]+cm[1][1])),2))

print('Accuracy: ' + gboost_accuracy +'%')
print('Recall: ' + gboost_recall +'%')
print('Confusion matrix:')
print(cm)

fpr, tpr, _ = roc_curve(test_y, y_gboost_pred)
auc = np.trapz(fpr,tpr)
print('Area under the ROC curve: ' + str(auc))

fig = plt.figure(1)
plt.plot(fpr,tpr,color='green')
plt.xlabel('False positive rate (FPR)')
plt.ylabel('True positive rate (TPR)')
plt.title('Receiver operating characteristic (ROC)')

fig = plt.figure(2)
ax = fig.add_subplot(111)
cax = ax.matshow(cm)
plt.title('Confusion matrix for Gradient Boosting classifier with original data')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
Accuracy: 96.88%
Recall: 84.63%
Confusion matrix:
[[15951     2]
 [  622  3425]]
Area under the ROC curve: 0.07690973133009896
```







In [269]:

```
# In[ ]:


dtree = DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=15,
            max_features=5, 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=False, random_state=None,
            splitter='best')
dtree.fit(train_x, train_y)
```

Out[269]:

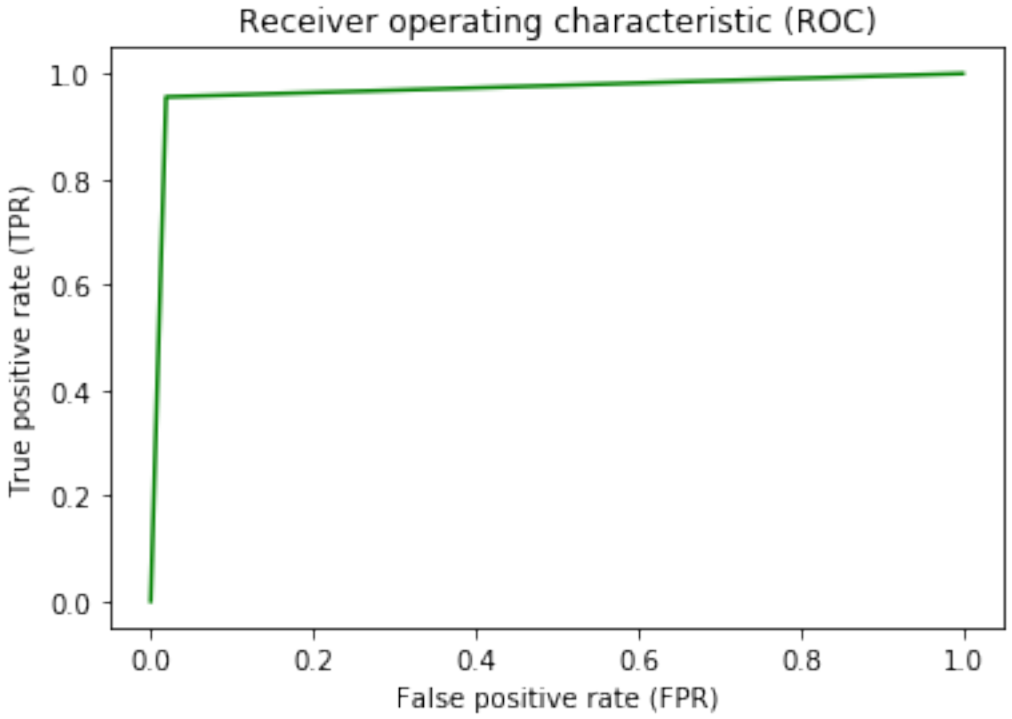
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=
15,
            max_features=5, max_leaf_nodes=None, min_impurity_decrease
=0.0,
            min_impurity_split=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None, splitter='best')
```

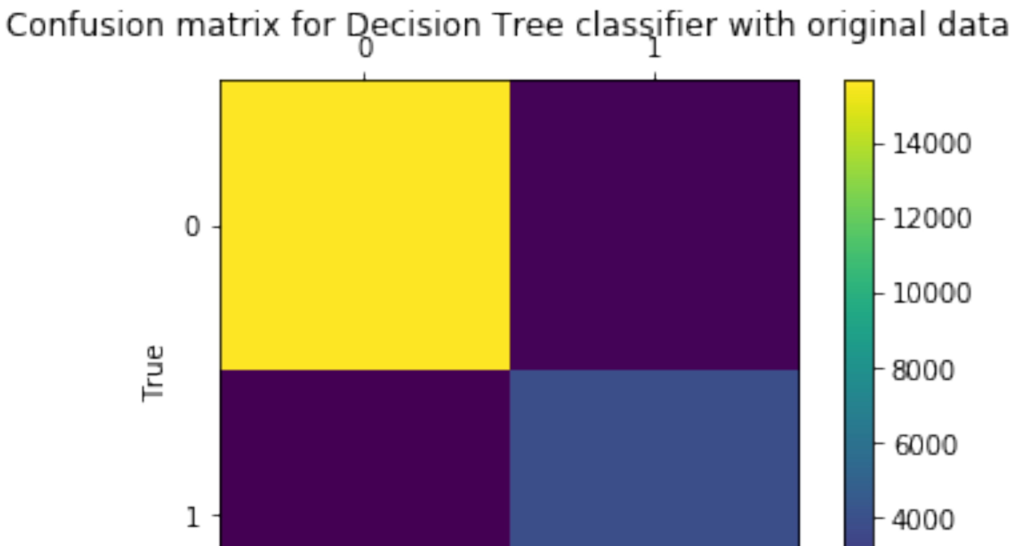
In [270]:

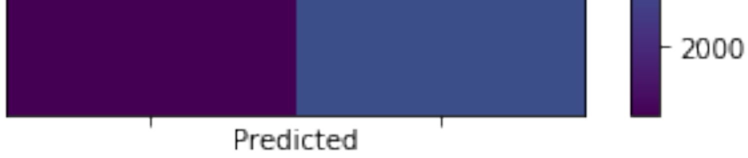
```
# In[ ]:


y_dtree_pred = dtree.predict(test_x)

labels = [0, 1]
cm = confusion_matrix(test_y, y_dtree_pred,labels)

dtree_accuracy = str(np.round(100*float(cm[0][0]+cm[1][1])/float((cm[0][0]+cm[1][1
dtree_recall = str(np.round(100*float((cm[1][1]))/float((cm[1][0]+cm[1][1])),2))

print('DT Accuracy: ' + dtree_accuracy +'%')
print('Recall: ' + dtree_recall +'%')
print('Confusion matrix:')
print(cm)

fpr, tpr, _ = roc_curve(test_y, y_dtree_pred)
auc = np.trapz(fpr,tpr)
print('Area under the ROC curve: ' + str(auc))
```

```
fig = plt.figure(1)
plt.plot(fpr,tpr,color='green')
plt.xlabel('False positive rate (FPR)')
plt.ylabel('True positive rate (TPR)')
plt.title('Receiver operating characteristic (ROC)')

fig = plt.figure(2)
ax = fig.add_subplot(111)
cax = ax.matshow(cm)
plt.title('Confusion matrix for Decision Tree classifier with original data')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
DT Accuracy: 97.6%
Recall: 95.58%
Confusion matrix:
[[15652   301]
 [  179  3868]]
Area under the ROC curve: 0.031549109286636734
```







In [228]:

```
# In[ ]:


rforest = RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini
            max_depth=6, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=4,
            min_weight_fraction_leaf=0.0, n_estimators=5, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
rforest.fit(train_x, train_y)
```

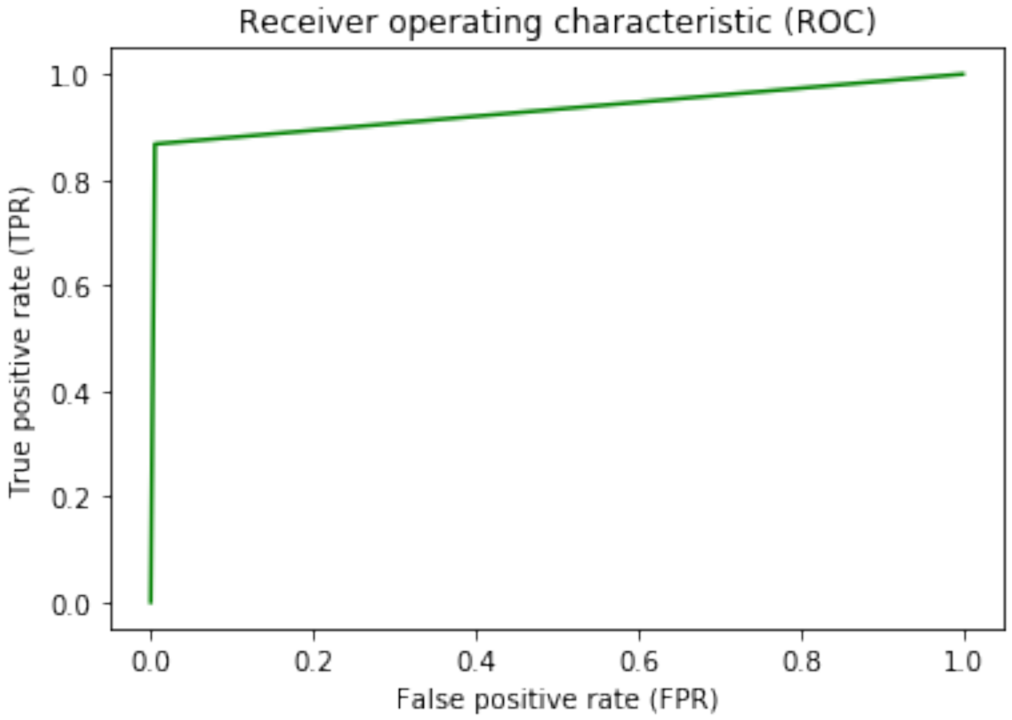
Out[228]:

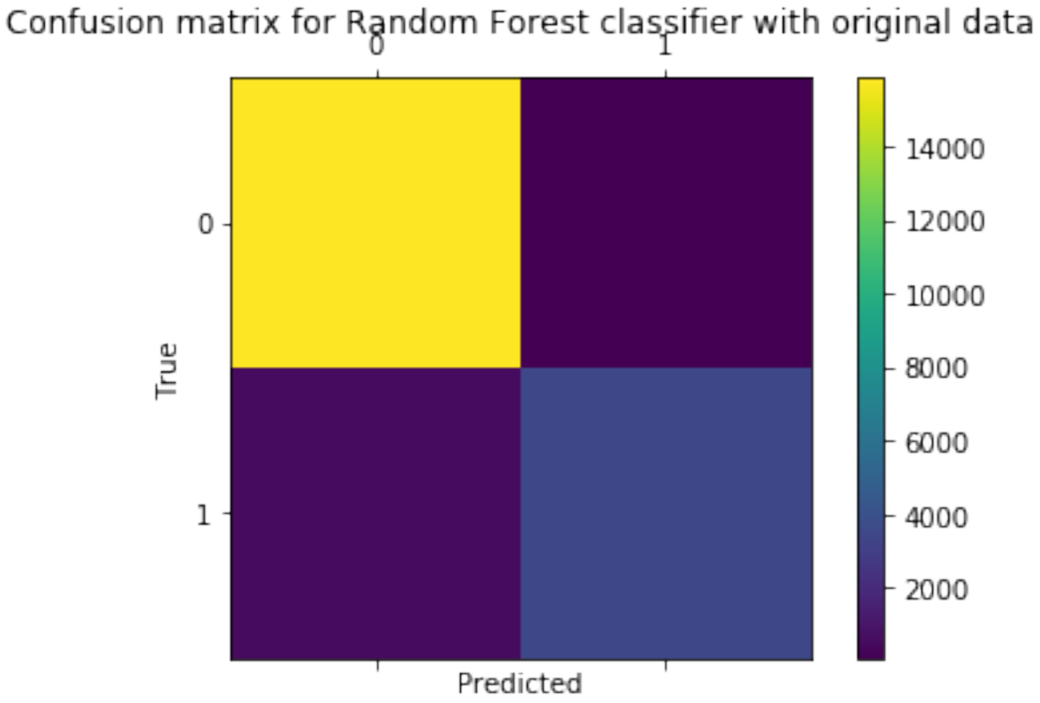
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='g
ini',
            max_depth=6, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=4,
            min_weight_fraction_leaf=0.0, n_estimators=5, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [232]:

```
# In[ ]:


y_rforest_pred = rforest.predict(test_x)

labels = [0, 1]
cm = confusion_matrix(test_y, y_rforest_pred,labels)

rforest_accuracy = str(np.round(100*float(cm[0][0]+cm[1][1])/float((cm[0][0]+cm[1][
rforest_recall = str(np.round(100*float((cm[1][1]))/float((cm[1][0]+cm[1][1])),2))

print('RF Accuracy: ' + rforest_accuracy +'%')
print('Recall: ' + rforest_recall +'%')
print('Confusion matrix:')
print(cm)

fpr, tpr, _ = roc_curve(test_y, y_rforest_pred)
auc = np.trapz(fpr,tpr)
print('Area under the ROC curve: ' + str(auc))

fig = plt.figure(1)
plt.plot(fpr,tpr,color='green')
plt.xlabel('False positive rate (FPR)')
plt.xlabel('False positive rate (FPR)')
plt.ylabel('True positive rate (TPR)')
plt.title('Receiver operating characteristic (ROC)')

fig = plt.figure(2)
ax = fig.add_subplot(111)
cax = ax.matshow(cm)
plt.title('Confusion matrix for Random Forest classifier with original data')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
RF Accuracy: 96.91%
Recall: 86.76%
Confusion matrix:
[[15871    82]
 [  536  3511]]
Area under the ROC curve: 0.06879194228053555
```





In [22]:

```
reg = linear_model.LogisticRegression(C=1.0, class_weight=None, dual=True, fit_inter
          intercept_scaling=10, max_iter=20, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=2, solver='warn',
          tol=0.008, verbose=90, warm_start=True)
reg.fit(train_x, train_y)
```

```
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.p
y:433: FutureWarning: Default solver will be changed to 'lbfgs' in 0.2
2. Specify a solver to silence this warning.
  FutureWarning)

[LibLinear]

/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:922: Conver
genceWarning: Liblinear failed to converge, increase the number of ite
rations.
  "the number of iterations.", ConvergenceWarning)
```

Out[22]:

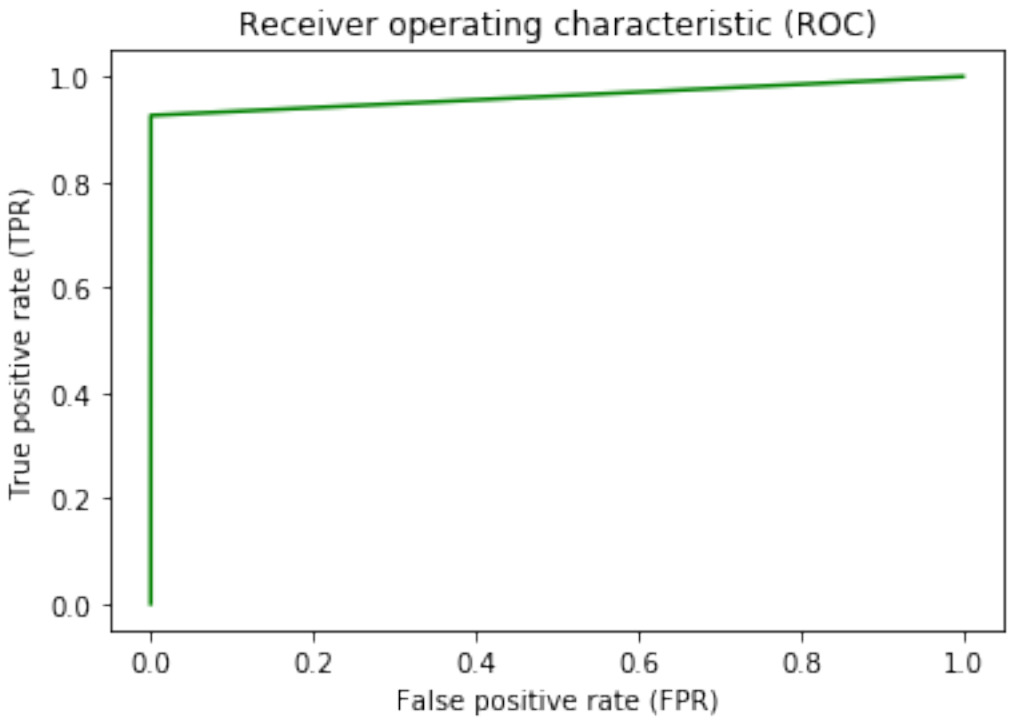
```
LogisticRegression(C=1.0, class_weight=None, dual=True, fit_intercept=
True,
          intercept_scaling=10, max_iter=20, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=2, solver='warn',
          tol=0.008, verbose=90, warm_start=True)
```

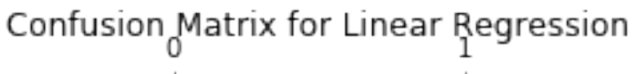
In [23]:

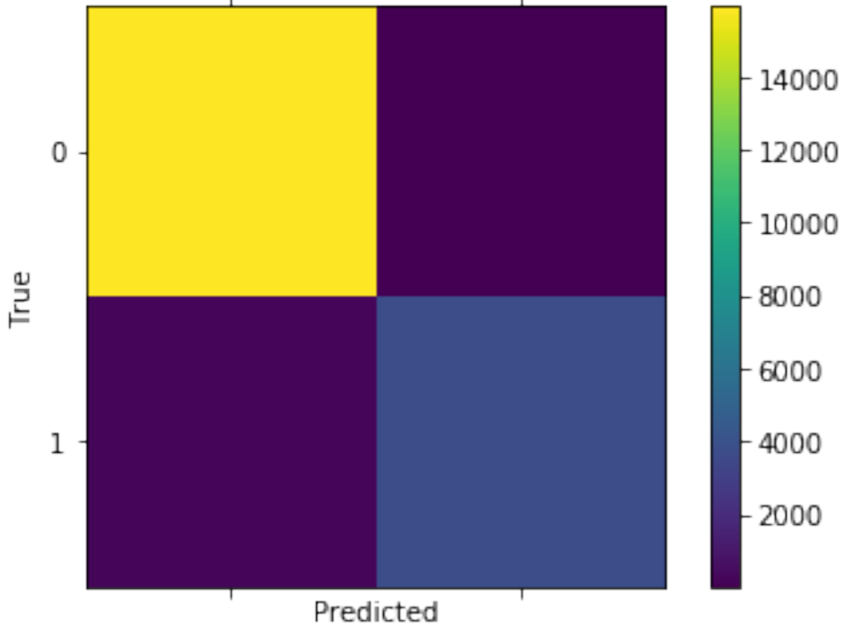
```


reg = linear_model.LogisticRegression(C=1.0, class_weight=None, dual=True, fit_inter
          intercept_scaling=10, max_iter=20, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=2, solver='warn',
          tol=0.008, verbose=90, warm_start=True)
reg.fit(train_x, train_y)

y_reg_pred = reg.predict(test_x)

labels = [0, 1]
cm = confusion_matrix(test_y, y_reg_pred,labels)

reg_accuracy = str(np.round(100*float(cm[0][0]+cm[1][1])/float((cm[0][0]+cm[1][1] +
reg_recall = str(np.round(100*float((cm[1][1]))/float((cm[1][0]+cm[1][1])),2))

print('Accuracy: ' + reg_accuracy +'%')
print('Recall: ' + reg_recall +'%')
print('Confusion matrix:')
print(cm)

fpr, tpr, _ = roc_curve(test_y, y_reg_pred)
auc = np.trapz(fpr,tpr)
print('Area under the ROC curve: ' + str(auc))
```

```
fig = plt.figure(1)
plt.plot(fpr,tpr,color='green')
plt.xlabel('False positive rate (FPR)')
plt.ylabel('True positive rate (TPR)')
plt.title('Receiver operating characteristic (ROC)')

fig = plt.figure(2)
ax = fig.add_subplot(111)
cax = ax.matshow(cm)
plt.title('Confusion Matrix for Linear Regression')
fig.colorbar(cax)
ax.set_xticklabels([''] + labels)
ax.set_yticklabels([''] + labels)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

```
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic.p
y:433: FutureWarning: Default solver will be changed to 'lbfgs' in 0.2
2. Specify a solver to silence this warning.
  FutureWarning)

[LibLinear]

/anaconda3/lib/python3.7/site-packages/sklearn/svm/base.py:922: Conver
genceWarning: Liblinear failed to converge, increase the number of ite
rations.
  "the number of iterations.", ConvergenceWarning)

Accuracy: 98.5%
Recall: 92.61%
Confusion matrix:
[[15951     2]
 [  299  3748]]
Area under the ROC curve: 0.03700362804371396
```







In [ ]:

In [ ]:

In [ ]:

In [1]:

In [6]:

In [7]:

Out[7]:

```
(100000, 20)
```

In [8]:

Out[8]:

```
True
```