Московский государственный технический университет
им. Н.Э. Баумана
(МГТУ им. Н.Э. Баумана)
Радиотехнический факультет (РТ)

Отчёт по лабораторной работе №6

По дисциплине

«Технологии машинного обучения»

Проверил:
Преподаватель кафедры ИУ-5
Гапанюк Ю.Е.
Подпись: _____
«___» _____ 2020 г.

Выполнил:
студент группы РТ5-61Б
Ануров Н.С.
Подпись: _____
«___» _____ 2020 г.

Москва, 2020

In [7]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
from sklearn.metrics import accuracy_score, balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score, classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_error, median_absolute_error, r2_score
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, LinearSVR
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, export_graphviz
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.ensemble import ExtraTreesClassifier, ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingClassifier, GradientBoostingRegressor

%matplotlib inline
```

In [10]:
```
df_test.head()
```

Out[10]:

	league	year	h_a	xG	xGA	npxG	npxGA	deep	deep_allowed	scored	...	ppda_coef	ppda_att	ppda_def	oppda_coef	oppda_att	oppd
0	Ligue_1	2019	a	1.514670	1.124190	0.754575	1.124190	3	2	2	...	14.727273	324	22	10.952381	230	
1	Ligue_1	2019	h	2.561580	1.170330	1.801490	1.170330	11	2	0	...	8.827586	256	29	10.227273	225	
2	Ligue_1	2019	a	0.683431	0.785876	0.683431	0.785876	5	1	1	...	10.297297	381	37	6.700000	134	
3	Ligue_1	2019	h	1.891860	1.042850	1.131760	1.042850	7	2	1	...	14.368421	273	19	17.705882	301	
4	Ligue_1	2019	a	0.767321	0.825226	0.767321	0.825226	6	3	0	...	12.714286	267	21	12.115385	315	

5 rows × 29 columns

In [11]:
```
df_train.head()
```

Out[11]:

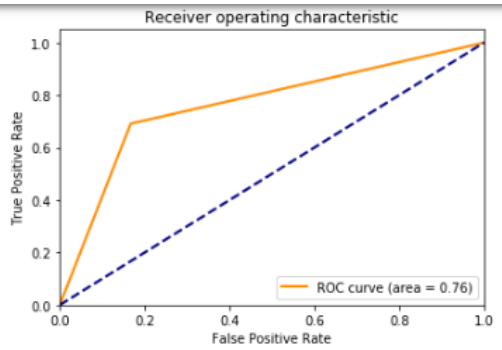
	league	year	h_a	xG	xGA	npxG	npxGA	deep	deep_allowed	scored	...	ppda_coef	ppda_att	ppda_def	oppda_coef	oppda_att	oppd
0	Bundesliga	2014	h	2.57012	1.198420	2.57012	1.198420	5	4	2	...	9.625000	231	24	21.850000	437	
1	Bundesliga	2014	a	1.50328	1.307950	1.50328	1.307950	10	1	1	...	4.756098	195	41	17.695652	407	
2	Bundesliga	2014	h	1.22987	0.310166	1.22987	0.310166	13	3	2	...	5.060606	167	33	16.961538	441	
3	Bundesliga	2014	a	1.03519	0.203118	1.03519	0.203118	6	2	0	...	4.423077	115	26	9.446809	444	
4	Bundesliga	2014	h	3.48286	0.402844	3.48286	0.402844	23	2	4	...	4.250000	170	40	44.800000	448	

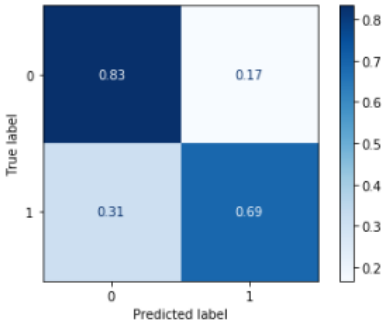
5 rows × 29 columns

In [90]:
```
clas_models = {'LogR': LogisticRegression(),
               'KNN_5':KNeighborsClassifier(n_neighbors=5),
               'SVC':SVC(),
               'Tree':DecisionTreeClassifier(),
               'RF':RandomForestClassifier(),
               'GB':GradientBoostingClassifier()}
```

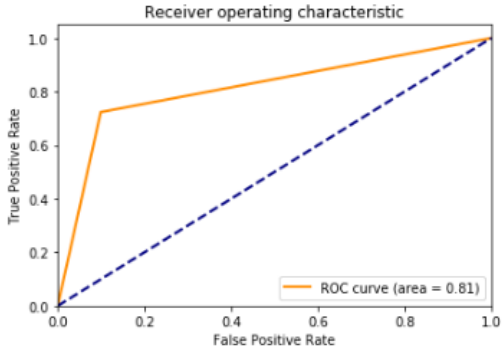
In [95]:
```
clasMetricLogger = MetricLogger()

def clas_train_model(model_name, model, clasMetricLogger):
    model.fit(X_train, Y_train)
    Y_pred = model.predict(X_test)
    precision = precision_score(Y_test.values, Y_pred)
    recall = recall_score(Y_test.values, Y_pred)
    f1 = f1_score(Y_test.values, Y_pred)
    roc_auc = roc_auc_score(Y_test.values, Y_pred)

    clasMetricLogger.add('precision', model_name, precision)
    clasMetricLogger.add('recall', model_name, recall)
    clasMetricLogger.add('f1', model_name, f1)
    clasMetricLogger.add('roc_auc', model_name, roc_auc)

    print('************************************************')
    print(model)
    print('************************************************')
    draw_roc_curve(Y_test.values, Y_pred)

    plot_confusion_matrix(model, X_test, Y_test.values,
                          display_labels=['0','1'],
                          cmap=plt.cm.Blues, normalize='true')
    plt.show()
```

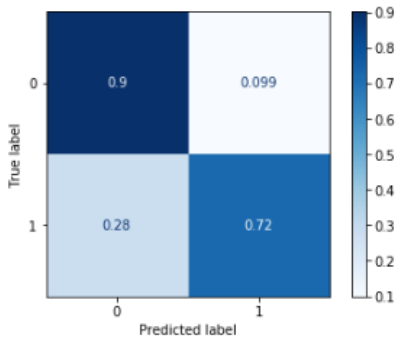
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
**********************************************************
```






```
**********************************************************
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)
**********************************************************
```




0.9
0.099
0.28
0.72
0
1
True label
0
1
Predicted label
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1