Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Лабораторная работа №6
по дисциплине
«Методы машинного обучения»
на тему
«Ансамбли моделей машинного обучения»

Выполнил:
студент группы ИУ5-23М
Богомолов Д.Н.

Москва — 2020 г.

ЛР6 по курсу Методы машинного обучения

Ансамбли моделей машинного обучения

Цель лабораторной работы: изучение ансамблей моделей машинного обучения. Задание:

1. Выберите набор данных (датасет) для решения задачи классификации или регресии.
2. В случае необходимости проведите удаление или заполнение пропусков и кодирован
3. С использованием метода train_test_split разделите выборку на обучающую и тестову
4. Обучите две ансамблевые модели. Оцените качество моделей с помощью одной из п качество полученных моделей.
5. Произведите для каждой модели подбор значений одного гиперпараметра. В зависи применять функцию GridSearchCV, использовать перебор параметров в цикле, или ис
6. Повторите пункт 4 для найденных оптимальных значений гиперпараметров. Сравнит качеством моделей, полученных в пункте 4.

Задание 1. Выберите набор данных (датасет) для решения зада регресии.

Для ЛР был использован набор данных о выявлении заболеваний сердца **heart-disease**. Д

- age
- sex
- chest pain type (4 values)
- resting blood pressure
- serum cholestoral in mg/dl
- fasting blood sugar > 120 mg/dl
- resting electrocardiographic results (values 0,1,2)
- maximum heart rate achieved
- exercise induced angina
- oldpeak = ST depression induced by exercise relative to rest
- the slope of the peak exercise ST segment
- number of major vessels (0-3) colored by flourosopy
- thal: 3 = normal; 6 = fixed defect; 7 = reversable defect

```
1  import numpy as np
2  import pandas as pd
3  import seaborn as sns
4  import matplotlib.pyplot as plt
5  %matplotlib inline
6  sns.set(style="ticks")
```

```
data = pd.read_csv( /content/drive/My Drive/MMO_Datasets/heart.csv ,sep= , )
data.head(10)
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	tha
0	63	1	3	145	233	1	0	150	0	2.3	0	0	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

Задание 2. В случае необходимости проведите удаление пропусков и кодирование категориальных признаков.

```
data.shape
```

```
(303, 14)
```

```
# Проверка на пустые значения
data.isnull().sum()
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

Масштабирование

```
from sklearn.preprocessing import MinMaxScaler
import warnings
warnings.filterwarnings('ignore')

```

```
# Create the scaler object with a range of 0-1
scaler = MinMaxScaler(feature_range=(0, 1))
# Fit on data, transform  data
scaler.fit_transform(data)

```

```
array([[0.70833333, 1.        , 1.        , ..., 0.        , 0.33333333,
        1.        ],
       [0.16666667, 1.        , 0.66666667, ..., 0.        , 0.66666667,
        1.        ],
       [0.25      , 0.        , 0.33333333, ..., 0.        , 0.66666667,
        1.        ],
       ...,
       [0.8125    , 1.        , 0.        , ..., 0.5       , 1.        ,
        0.        ],
       [0.58333333, 1.        , 0.        , ..., 0.25      , 1.        ,
        0.        ],
       [0.58333333, 0.        , 0.33333333, ..., 0.25      , 0.66666667,
        0.        ]])
```

```
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, auc
import pylab as pl
```

```
# Пустых значений нет
# Перейдем к разделению выборки на обучающую и тестовую.
X = data.drop('target',axis = 1).values
y = data['target'].values
```

Задание 3. С использованием метода train_test_split разделите тестовую.

```
from sklearn.model_selection import train_test_split
kfold = 5 #количество подвыборок для валидации
itog_val = {} #список для записи результатов кросс валидации разных алгоритмов
```

Задание 4. Обучите две ансамблевые модели. Оцените качеств из подходящих для задачи метрик. Сравните качество получен

```
ROCtrainTRN, ROCtestTRN, ROCtrainTRG, ROCtestTRG = train_test_split(X, y, test_size=0
```

```
model_rfc = RandomForestClassifier(n_estimators = 70) #в параметре передаем кол-во де
model_knc = KNeighborsClassifier(n_neighbors = 18) #в параметре передаем кол-во сосед
model_lr = LogisticRegression(penalty='l2', tol=0.01)
model_svc = svm.SVC() #по умолчанию kernek='rbf'
```

```
from sklearn.model_selection import cross_val_score
```

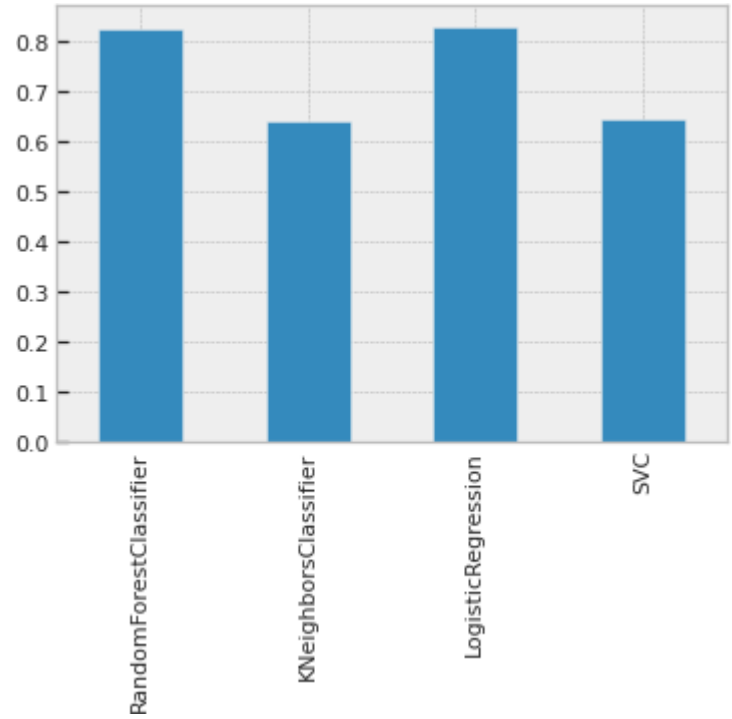
SVM - метод опорных векторов(SVC): Суть работы “Машин” Опорных Векторов проста: алго которая разделяет данные на классы. Метод k-ближайших соседей(KNeighborsClassifier) Ra Логистическая регрессия (LogisticRegression)

```
from sklearn.ensemble import ExtraTreesClassifier, ExtraTreesRegressor
```

```
scores = cross_val_score(model_rfc, X, y, cv = kfold)
itog_val['RandomForestClassifier'] = scores.mean()
scores = cross_val_score(model_knc, X, y, cv = kfold)
itog_val['KNeighborsClassifier'] = scores.mean()
scores = cross_val_score(model_lr, X, y, cv = kfold)
itog_val['LogisticRegression'] = scores.mean()
scores = cross_val_score(model_svc, X, y, cv = kfold)
itog_val['SVC'] = scores.mean()
```

```
import matplotlib.pyplot as plt
plt.style.use('bmh')
data.from_dict(data = itog_val, orient='index').plot(kind='bar', legend=False)
```

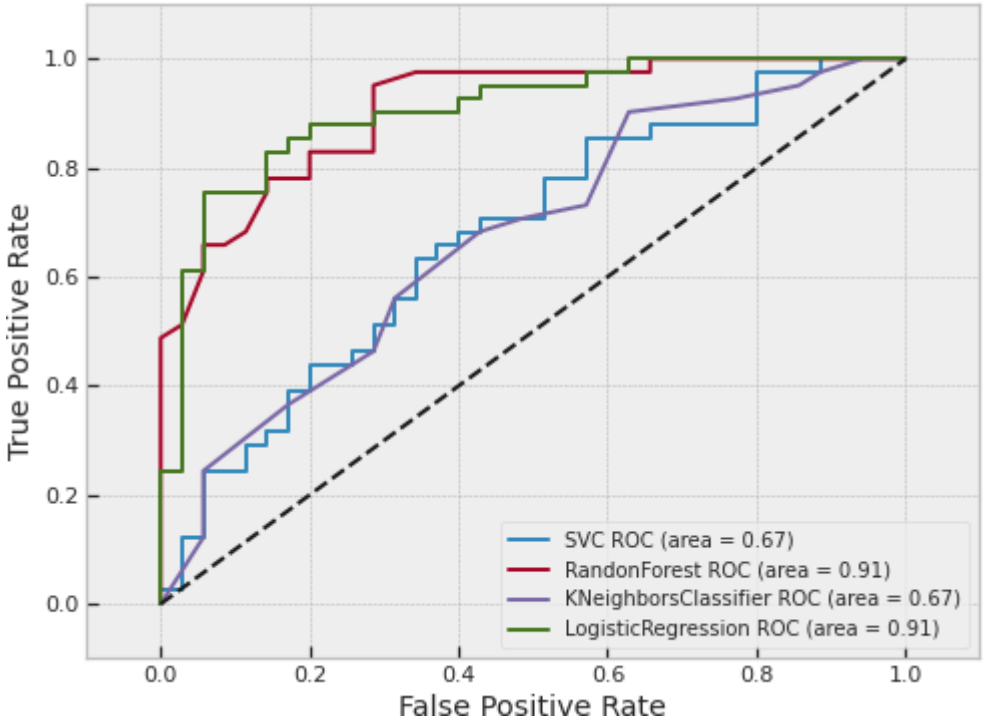
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa04d5e8fd0>
```




```
pl.clf()
plt.figure(figsize=(8,6))
#SVC
model_svc.probability = True
probas = model_svc.fit(ROCtrainTRN, ROCtrainTRG).predict_proba(ROCtestTRN)
fpr, tpr, thresholds = roc_curve(ROCtestTRG, probas[:, 1])
roc_auc  = auc(fpr, tpr)
pl.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % ('SVC', roc_auc))
#RandomForestClassifier
```

```
probas = model_rfc.fit(ROCtrainTRN, ROCtrainTRG).predict_proba(ROCtestTRN)
fpr, tpr, thresholds = roc_curve(ROCtestTRG, probas[:, 1])
roc_auc  = auc(fpr, tpr)
pl.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % ('RandonForest',roc_auc))
#KNeighborsClassifier
probas = model_knc.fit(ROCtrainTRN, ROCtrainTRG).predict_proba(ROCtestTRN)
fpr, tpr, thresholds = roc_curve(ROCtestTRG, probas[:, 1])
roc_auc  = auc(fpr, tpr)
pl.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % ('KNeighborsClassifier',roc_auc))
#LogisticRegression
probas = model_lr.fit(ROCtrainTRN, ROCtrainTRG).predict_proba(ROCtestTRN)
fpr, tpr, thresholds = roc_curve(ROCtestTRG, probas[:, 1])
roc_auc  = auc(fpr, tpr)
pl.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % ('LogisticRegression',roc_auc))
pl.plot([0, 1], [0, 1], 'k--')
pl.xlim([-0.1, 1.1])
pl.ylim([-0.1, 1.1])
pl.xlabel('False Positive Rate')
pl.ylabel('True Positive Rate')
pl.legend(loc=0, fontsize='small')
pl.show()
```

```
<Figure size 432x288 with 0 Axes>
```



Задание 5. Произведите для каждой модели подбор значений о зависимости от используемой библиотеки можно применять ф использовать перебор параметров в цикле, или использовать

```
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import balanced_accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.35, random_state=1)
```

Масштабирование

```
from sklearn.preprocessing import MinMaxScaler
import warnings
warnings.filterwarnings('ignore')
scaler = MinMaxScaler(feature_range=(0, 1))
scaler.fit_transform(X)
scaler.fit_transform(X_train)
scaler.fit_transform(X_test)
```

```
array([[0.77777778, 0.        , 0.        , ..., 0.        , 0.75      ,
        1.        ],
       [0.61111111, 1.        , 0.33333333, ..., 1.        , 0.        ,
        1.        ],
       [0.38888889, 1.        , 0.        , ..., 1.        , 0.5       ,
        1.        ],
       ...,
       [0.52777778, 1.        , 0.        , ..., 0.        , 0.        ,
        1.        ],
       [0.66666667, 1.        , 0.33333333, ..., 0.5       , 1.        ,
        1.        ],
       [0.38888889, 1.        , 0.33333333, ..., 0.        , 0.        ,
        1.        ]])
```

```
rfc = RandomForestClassifier().fit(X_train, y_train)
predicted_rfc = rfc.predict(X_test)
```

```
accuracy_score(y_test, predicted_rfc)
```

```
0.7476635514018691
```

```
balanced_accuracy_score(y_test, predicted_rfc)
```

```
0.7471929824561403
```

```
(precision_score(y_test, predicted_rfc, average='weighted'),
 recall_score(y_test, predicted_rfc, average='weighted'))
```

```
(0.748059504175502, 0.7476635514018691)
```

```
f1_score(y_test, predicted_rfc, average='weighted')
```

```
0.7477962087830651
```

```
abc = AdaBoostClassifier().fit(X_train, y_train)
predicted_abc = abc.predict(X_test)
```

```
accuracy_score(y_test, predicted_abc)
```

```
0.7289719626168224
```

```
balanced_accuracy_score(y_test, predicted_abc)
```

```
0.7284210526315789
```

```
(precision_score(y_test, predicted_abc, average='weighted'),
 recall_score(y_test, predicted_abc, average='weighted'))
```

```
(0.7293842770753162, 0.7289719626168224)
```

```
f1_score(y_test, predicted_abc, average='weighted')
```

```
0.7291144464706996
```

```
rfc_n_range = np.array(range(5,100,5))
rfc_tuned_parameters = [{'n_estimators': rfc_n_range}]
rfc_tuned_parameters
```

```
[{'n_estimators': array([ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75,
         90, 95])}]
```

GridSearchCV

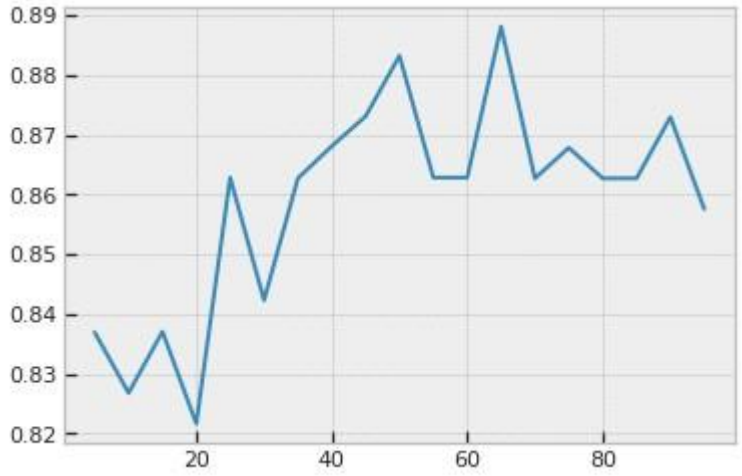
```
import warnings
from sklearn.model_selection import GridSearchCV
warnings.filterwarnings('ignore')

gs_rfc = GridSearchCV(RandomForestClassifier(), rfc_tuned_parameters, cv=5,
                      scoring='accuracy')
gs_rfc.fit(X_train, y_train)
```

```
1  gs_rfc.best_params_
```

```
{'n_estimators': 65}
```

```
1  plt.plot(rfc_n_range, gs_rfc.cv_results_['mean_test_score'])
```

```
[<matplotlib.lines.Line2D at 0x7fa04d40cd68>]
```




```
1  abc_n_range = np.array(range(5,100,5))
2  abc_tuned_parameters = [{'n_estimators': abc_n_range}]
3  abc_tuned_parameters
```

```
[{'n_estimators': array([ 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75,
         90, 95])}]
```

```
1  gs_abc = GridSearchCV(AdaBoostClassifier(), abc_tuned_parameters, cv=5,
2                        scoring='accuracy')
3  gs_abc.fit(X_train, y_train)
```

```
GridSearchCV(cv=5, error_score=nan,
             estimator=AdaBoostClassifier(algorithm='SAMME.R',
                                          base_estimator=None,
                                          learning_rate=1.0, n_estimators=50,
                                          random_state=None),
             iid='deprecated', n_jobs=None,
             param_grid=[{'n_estimators': array([ 5, 10, 15, 20, 25, 30, 35, 40, 45,
       90, 95])}],
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring='accuracy', verbose=0)
```

```
1  gs_abc.best_params_
```

```
{'n_estimators': 25}
```

```
1  plt.plot(abc_n_range, gs_abc.cv_results_['mean_test_score'])
```

```
[<matplotlib.lines.Line2D at 0x7fa04d36a438>]
```

Задание 6. Повторите пункт 4 для найденных оптимальных зна Сравните качество полученных моделей с качеством моделей,

```
1  rfc_optimized = RandomForestClassifier(n_estimators=gs_rfc.best_params_['n_estimators
2  predicted_rfc_opt = rfc_optimized.predict(X_test)
```

```
1  accuracy_score(y_test, predicted_rfc_opt)
```

```
0.719626168224299
```

```
1  balanced_accuracy_score(y_test, predicted_rfc_opt)
```

```
0.7196491228070175
```

```
1  (precision_score(y_test, predicted_rfc_opt, average='weighted'),
2   recall_score(y_test, predicted_rfc_opt, average='weighted'))
```

```
(0.7206195673485394, 0.719626168224299)
```

```
1  f1_score(y_test, predicted_rfc_opt, average='weighted')
```

```
0.7198715934972119
```

```
1  abc_optimized = RandomForestClassifier(n_estimators=gs_abc.best_params_['n_estimators
2  predicted_abc_opt = abc_optimized.predict(X_test)
```

```
1  accuracy_score(y_test, predicted_abc_opt)
```

```
0.719626168224299
```

```
1  balanced_accuracy_score(y_test, predicted_abc_opt)
```

```
0.7221052631578948
```

```
1  (precision_score(y_test, predicted_abc_opt, average='weighted'),
2   recall_score(y_test, predicted_abc_opt, average='weighted'))
```

```
(0.724456137595225, 0.719626168224299)
```

```
1   f1_score(y_test, predicted_abc_opt, average='weighted')
```

```
0.7197731146770117
```

Список литературы

[1] Гапанюк Ю. Е. Лабораторная работа «Ансамбли моделей машинного обучения.» [Электронный ресурс] // GitHub. – 2019. – Режим досту- па: https://github.com/ugapanyuk/ml_course_2020/wiki/LAB_MMO_ENSEMBLES (дата обращения: 22.05.2020).

[2] Scikit-learn. Boston house-prices dataset [Electronic resource] // Scikit-learn. — 2018. — Access mode: https://scikit-learn.org/0.20/modules/generated/sklearn.datasets.load_boston.html (online; accessed: 18.02.2020).

[3] Team The IPython Development. IPython 7.3.0 Documentation [Electronic resource] // Read the Docs. — 2019. — Access mode: https://ipython.readthedocs.io/en/ stable/ (online; accessed: 20.02.2020).

[4] Waskom M. seaborn 0.9.0 documentation [Electronic resource] // PyData. — 2018. — Access mode: https://seaborn.pydata.org/ (online; accessed: 20.02.2020).