Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ**

Отчет по лабораторной работе № 5
«Линейные модели, SVM и деревья решений»
по курсу “Технологии машинного обучения”

Исполнитель:
Студент группы ИУ5-63
Желанкина А.С.
___________ 17.04.2018

Москва, 2019

Задание лабораторной работы

1. Выберите набор данных (датасет) для решения задачи классификации или регресии.
2. В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.
3. С использованием метода train_test_split разделите выборку на обучающую и тестовую.
4. Обучите:
   1) одну из линейных моделей,
   2) SVM,
   3) дерево решений.
   Оцените качество моделей с помощью трех подходящих для задачи метрик. Сравните качество полученных моделей.
5. Произведите для каждой модели подбор одного гиперпараметра с использованием GridSearchCV и кросс-валидации.
6. Повторите пункт 4 для найденных оптимальных значений гиперпараметров. Сравните качество полученных моделей с качеством моделей, полученных в пункте 4.

Экранные формы с текстом программы и примерами её выполнения

jupyter lab5 Last Checkpoint: несколько секунд назад (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [63]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

from sklearn.linear_model import SGDClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV

%matplotlib inline
sns.set(style="ticks")
```

Был взят датасет из прошлой лабораторной работы

In [64]:
```
data = pd.read_csv('heart.csv', sep=",")
data.head()
```

Out[64]:

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

In [65]:
```
data.describe()
```

Out[65]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpea
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.00000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.03960
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.16107
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.00000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.00000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.80000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.60000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.20000

In [66]:
```
data.corr()
```

Out[66]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.398522	0.096801	0.210013	-0.168814
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.044020	0.141664	0.096093	-0.030711
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.295762	-0.394280	-0.149230	0.119717
trestbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.046698	0.067616	0.193216	-0.121475
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.009940	0.067023	0.053952	-0.004038
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.008567	0.025665	0.005747	-0.059894
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.044123	-0.070733	-0.058770	0.093045
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.000000	-0.378812	-0.344187	0.386784
exang	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.378812	1.000000	0.288223	-0.257748
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.344187	0.288223	1.000000	-0.577537
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.386784	-0.257748	-0.577537	1.000000
ca	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.213177	0.115739	0.222682	-0.080155
thal	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.096439	0.206754	0.210244	-0.104764
target	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.421741	-0.436757	-0.430696	0.345877

Деление датасета на обучающую и тестовую выборки

In [67]:
```
X_train, X_test, y_train, y_test = train_test_split(
    data, data['target'], test_size=0.2, random_state=1)
```

Обучение моделей: линейной, SVM и дерево решений

In [68]:
```
sgd = SGDClassifier().fit(X_train, y_train)
```

```
C:\Anaconda\lib\site-packages\sklearn\linear_model\stochastic_gradient.py:166: FutureWarning: max_ite
r and tol parameters have been added in SGDClassifier in 0.19. If both are left unset, they default t
o max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=1000. From 0.21, default
max_iter will be 1000, and default tol will be 1e-3.
  FutureWarning)
```

In [69]:
```
svm_svc = SVC(gamma='auto').fit(X_train, y_train)
```

In [70]:
```
decision_tree = DecisionTreeClassifier(random_state=1, max_depth=0.75).fit(X_train, y_train)
```

Предсказание

In [71]:
```
target_sgd = sgd.predict(X_test)
```

In [72]:
```
target_svm_svc = svm_svc.predict(X_test)
```

In [73]:
```
target_decision_tree = decision_tree.predict(X_test)
```

Оценка качества стохастического градиентного спуска

In [74]:
```
accuracy_score(y_test, target_sgd), \
precision_score(y_test, target_sgd),  \
recall_score(y_test, target_sgd)
```

Out[74]:
```
(0.5081967213114754, 1.0, 0.03225806451612903)
```

Оценка качества SVM

```
In [75]: accuracy_score(y_test, target_svm_svc), \
         precision_score(y_test, target_svm_svc),  \
         recall_score(y_test, target_svm_svc)

Out[75]: (0.5081967213114754, 0.5081967213114754, 1.0)
```

Оценка качества дерева принятия решений

```
In [76]: accuracy_score(y_test, target_decision_tree), \
         precision_score(y_test, target_decision_tree),  \
         recall_score(y_test, target_decision_tree)

Out[76]: (0.5081967213114754, 0.5081967213114754, 1.0)
```

Подбор одного гиперпараметра с использованием GridSearchCV и кросс-валидации для каждой модели

```
In [77]: scores_sgd = cross_val_score(SGDClassifier(),
                                      X_train, y_train, cv=2)
         scores_sgd

Out[77]: array([0.60330579, 0.55371901])

In [78]: scores_svm_svc = cross_val_score(SVC(gamma='auto'),
                                      X_train, y_train, cv=2)
         scores_svm_svc

Out[78]: array([0.55371901, 0.55371901])

In [79]: scores_decision_tree = cross_val_score(DecisionTreeClassifier(),
                                      X_train, y_train, cv=2)
         scores_decision_tree

Out[79]: array([1., 1.])

In [95]: parameters = {'alpha':[0.5,0.4,0.3,0.2,0.1]}
         clf_gs_sgd = GridSearchCV(SGDClassifier(), parameters, cv=2, scoring='accuracy')
         clf_gs_sgd.fit(X_train, y_train)

Out[95]: GridSearchCV(cv=2, error_score='raise-deprecating',
                estimator=SGDClassifier(alpha=0.0001, average=False, class_weight=None,
                early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
                l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=None,
                n_iter=None, n_iter_no_change=5, n_jobs=None, penalty='l2',
                power_t=0.5, random_state=None, shuffle=True, tol=None,
                validation_fraction=0.1, verbose=0, warm_start=False),
                fit_params=None, iid='warn', n_jobs=None,
                param_grid={'alpha': [0.5, 0.4, 0.3, 0.2, 0.1]},
                pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                scoring='accuracy', verbose=0)

In [96]: clf_gs_sgd.best_params_

Out[96]: {'alpha': 0.5}

In [82]: parameters = {'gamma':[0.9,0.8,0.7,0.6,0.5,0.4,0.3,0.2,0.1]}
         clf_gs_svm_svc = GridSearchCV(SVC(), parameters, cv=2, scoring='accuracy')
         clf_gs_svm_svc.fit(X_train, y_train)

Out[82]: GridSearchCV(cv=2, error_score='raise-deprecating',
                estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
           decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
           kernel='rbf', max_iter=-1, probability=False, random_state=None,
           shrinking=True, tol=0.001, verbose=False),
                fit_params=None, iid='warn', n_jobs=None,
                param_grid={'gamma': [0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1]},
                pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                scoring='accuracy', verbose=0)

In [83]: clf_gs_svm_svc.best_params_

Out[83]: {'gamma': 0.9}

In [84]: parameters = {'min_impurity_decrease':[0.9,0.8,0.7,0.6,0.5,0.4,0.3,0.2,0.1]}
         clf_gs_decision_tree = GridSearchCV(DecisionTreeClassifier(), parameters, cv=2, scoring='accuracy')
         clf_gs_decision_tree.fit(X_train, y_train)
```

Out[84]: GridSearchCV(cv=2, error_score='raise-deprecating',
```
       estimator=DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            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, presort=False, random_state=None,
            splitter='best'),
       fit_params=None, iid='warn', n_jobs=None,
       param_grid={'min_impurity_decrease': [0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='accuracy', verbose=0)
```

In [85]:
```
clf_gs_decision_tree.best_params_
```

Out[85]: {'min_impurity_decrease': 0.4}

Обучение моделей: линейной, SVM и дерево решений с использованием оптимальных значений гиперпараметров

In [97]:
```
sgd_new = SGDClassifier(alpha=0.5).fit(X_train, y_train)
```

In [87]:
```
svm_svc_new = SVC(gamma=0.9).fit(X_train, y_train)
```

In [88]:
```
decision_tree_new = DecisionTreeClassifier(random_state=1, min_impurity_decrease=0.4, max_depth=0.75).
```

Предсказание

In [98]:
```
target_sgd_new = sgd_new.predict(X_test)
```

In [90]:
```
target_svm_svc_new = svm_svc_new.predict(X_test)
```

In [91]:
```
target_decision_tree_new = decision_tree_new.predict(X_test)
```

Оценка качества

In [99]:
```
accuracy_score(y_test, target_sgd_new), \
precision_score(y_test, target_sgd_new),  \
recall_score(y_test, target_sgd_new)
```

Out[99]: (0.5409836065573771, 0.5294117647058824, 0.8709677419354839)

In [93]:
```
accuracy_score(y_test, target_svm_svc_new), \
precision_score(y_test, target_svm_svc_new),  \
recall_score(y_test, target_svm_svc_new)
```

Out[93]: (0.5081967213114754, 0.5081967213114754, 1.0)

In [94]:
```
accuracy_score(y_test, target_decision_tree_new), \
precision_score(y_test, target_decision_tree_new),  \
recall_score(y_test, target_decision_tree_new)
```

Out[94]: (0.5081967213114754, 0.5081967213114754, 1.0)