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

Лабораторная работа №5
по курсу «Методы машинного обучения»

«Линейные модели, SVM и деревья решений»

Выполнил: Ильин В.С.
Группа ИУ5-22М

Москва – 2019

Описание задания

Цель лабораторной работы: изучение линейных моделей, SVM и деревьев решений.

Задание

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 и 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.

Ход выполнения лабораторной работы

Выбор датасета

В качестве набора данных мы будем использовать набор данных по состоянию ходьбы человека - https://www.kaggle.com/vmalyi/run-or-walk. Датасет состоит из 88588 наборов значений взятых с акселерометра и гироскопа. Данные собирались на устройство iPhone 5c, который был закреплен на запястье человека(левое и правое). Информация о данных бралась каждые 10 секунд. Задача определения активности по электронным устройствам является актуальной для легкоатлетов.

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

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?clie

Enter your authorization code:
··········
Mounted at /content/drive
```

```
from google.colab import files
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

```
os.listdir()
data = pd.read_csv('drive/My Drive/mmo_datasets/row_dataset.csv', sep=",")
```

```
total_count = data.shape[0]
num_cols = []
for col in data.columns:
    # Количество пустых значений
    temp_null_count = data[data[col].isnull()].shape[0]
    dt = str(data[col].dtype)
    if temp_null_count>0:
        num_cols.append(col)
        temp_perc = round((temp_null_count / total_count) * 100.0, 2)
        print('Колонка {}. Тип данных {}. Количество пустых значений {}, {}%.'
              .format(col, dt, temp_null_count, temp_perc))

data_cleared = data
```

```
uniquevalues = np.unique(data_cleared['activity'].values)
uniquevalues
```

```
array([0, 1])
```

train_test_split

```
data_cleared = data_cleared.drop('date', axis=1)
data_cleared = data_cleared.drop('time', axis=1)
data_cleared = data_cleared.drop('username', axis=1)
```

```
target = data_cleared['activity']
data_cleared = data_cleared.drop('activity', axis=1)
```

```
data_cleared.head(10)
```

	wrist	acceleration_x	acceleration_y	acceleration_z	gyro_x	gyro_y	gy
0	0	0.2650	-0.7814	-0.0076	-0.0590	0.0325	-2.
1	0	0.6722	-1.1233	-0.2344	-0.1757	0.0208	0.
2	0	0.4399	-1.4817	0.0722	-0.9105	0.1063	-2.
3	0	0.3031	-0.8125	0.0888	0.1199	-0.4099	-2.
4	0	0.4814	-0.9312	0.0359	0.0527	0.4379	2.
5	0	0.4044	-0.8056	-0.0956	0.6925	-0.2179	2.
6	0	0.6320	-1.1290	-0.2982	0.0548	-0.1896	0.
7	0	0.6670	-1.3503	-0.0880	-0.8094	-0.7938	-1.
8	0	0.2704	-0.8633	0.1293	-0.4173	-0.1904	-2.
9	0	0.4690	-1.0740	0.0219	0.0388	1.1491	1.

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(
    data_cleared,
    target,
    test_size=0.2,
```

```
    random_state=1
)
```

```
X_train.shape, Y_train.shape
```

```
((70870, 7), (70870,))
```

```
X_test.shape, Y_test.shape
```

```
((17718, 7), (17718,))
```

Обучение

```
from sklearn.linear_model import SGDClassifier
from sklearn.svm import LinearSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

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

Стохастический градиентный спуск

```
sgd = SGDClassifier().fit(X_train, Y_train)
predicted_sgd = sgd.predict(X_test)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/stochastic_grad
  FutureWarning)
```

```
accuracy_score(Y_test, predicted_sgd)
```

```
0.8606501862512699
```

```
balanced_accuracy_score(Y_test, predicted_sgd)
```

```
0.86264084856474
```

```
(precision_score(Y_test, predicted_sgd, average='weighted'),
 recall_score(Y_test, predicted_sgd, average='weighted'))
```

```
(0.8754409896514297, 0.8606501862512699)
```

```
f1_score(Y_test, predicted_sgd, average='weighted')
```

```
0.8595997829328971
```

Линейный классификатор на основе SVM

```
svm = LinearSVC(C=1.0).fit(X_train, Y_train)
```

```
predicted_svm = svm.predict(X_test)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/svm/base.py:931: Convergence
  "the number of iterations.", ConvergenceWarning)
```

```
accuracy_score(Y_test, predicted_svm)
```

```
0.8641494525341461
```

```
balanced_accuracy_score(Y_test, predicted_svm)
```

```
0.865292833270138
```

```
(precision_score(Y_test, predicted_svm, average='weighted'),
 recall_score(Y_test, predicted_svm, average='weighted'))
```

```
(0.8693015182110753, 0.8641494525341461)
```

```
f1_score(Y_test, predicted_svm, average='weighted')
```

```
0.863866651257837
```

Дерево решений

```
dt = DecisionTreeClassifier(random_state=1).fit(X_train, Y_train)
predicted_dt = dt.predict(X_test)
```

```
accuracy_score(Y_test, predicted_dt)
```

```
0.9838582232757648
```

```
balanced_accuracy_score(Y_test, predicted_dt)
```

```
0.9838155393140757
```

```
(precision_score(Y_test, predicted_dt, average='weighted'),
 recall_score(Y_test, predicted_dt, average='weighted'))
```

```
(0.9838626269258699, 0.9838582232757648)
```

```
f1_score(Y_test, predicted_dt, average='weighted')
```

```
0.9838576028456452
```

Из двух представленных моделей с параметрами по умолчанию с задачей классификации на выбранном датасете лучше справляется линейный классификатор на основе SVM.

Подбор гиперпараметров

Стохастический градиентный спуск

```
n_range = np.array(range(0,100,5))
n_range = n_range / 100
tuned_parameters = [{'l1_ratio': n_range}]
tuned_parameters
```

```
[{'l1_ratio': array([0.  , 0.05, 0.1 , 0.15, 0.2 , 0.25, 0.3 , 0.35, 0.4 ,
        0.55, 0.6 , 0.65, 0.7 , 0.75, 0.8 , 0.85, 0.9 , 0.95])}]
```

```
import warnings
warnings.filterwarnings('ignore')

clf_gs_sgd = GridSearchCV(SGDClassifier(), tuned_parameters, cv=5,
                          scoring='accuracy')
clf_gs_sgd.fit(X_train, Y_train)
```

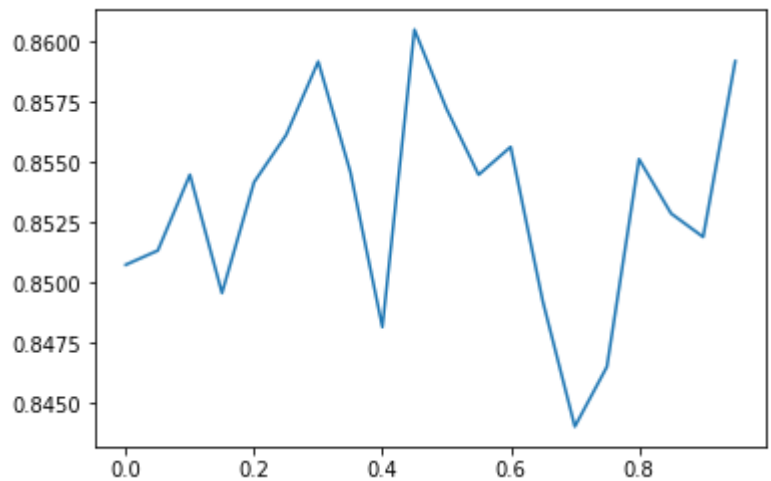
```
GridSearchCV(cv=5, error_score='raise-deprecating',
       estimator=SGDClassifier(alpha=0.0001, average=False, class_weight=No
       early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal', loss='hinge', max_iter=None,
       n_iter=None, n_iter_no_change=5, 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=[{'l1_ratio': array([0.  , 0.05, 0.1 , 0.15, 0.2 , 0.25,
       0.55, 0.6 , 0.65, 0.7 , 0.75, 0.8 , 0.85, 0.9 , 0.95])}],
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='accuracy', verbose=0)
```

```
clf_gs_sgd.best_params_
```

```
{'l1_ratio': 0.45}
```

```
plt.plot(n_range, clf_gs_sgd.cv_results_['mean_test_score'])
```

```
[<matplotlib.lines.Line2D at 0x7f7efc12f978>]
```



Линейный классификатор на основе SVM

```
n_range = np.array(range(1,20,1))
tuned_parameters = [{'C': n_range}]
```

```
tuned_parameters
```

```
[{'C': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 1
            18, 19])}]
```

```
clf_gs_svm = GridSearchCV(LinearSVC(), tuned_parameters, cv=3,
                      scoring='accuracy')
clf_gs_svm.fit(X_train, Y_train)
```

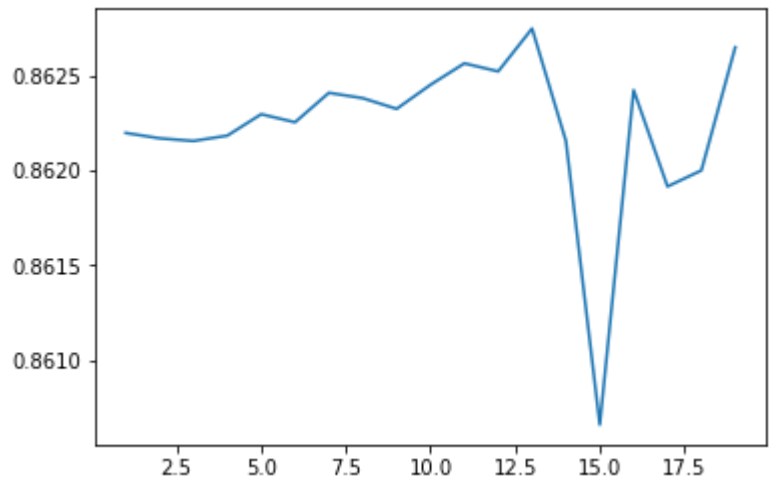
```
GridSearchCV(cv=3, error_score='raise-deprecating',
          estimator=LinearSVC(C=1.0, class_weight=None, dual=True, fit_interce
         intercept_scaling=1, loss='squared_hinge', max_iter=1000,
         multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
         verbose=0),
          fit_params=None, iid='warn', n_jobs=None,
          param_grid=[{'C': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11,
          18, 19])}],
          pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
          scoring='accuracy', verbose=0)
```

```
clf_gs_svm.best_params_
```

```
{'C': 13}
```

```
plt.plot(n_range, clf_gs_svm.cv_results_['mean_test_score'])
```

```
[<matplotlib.lines.Line2D at 0x7f7efc052048>]
```



Дерево решений

```
n_range = np.array(range(1,7,1))
tuned_parameters = [{'max_depth': n_range}]
tuned_parameters
```

```
[{'max_depth': array([1, 2, 3, 4, 5, 6])}]
```

```
clf_gs_dt = GridSearchCV(DecisionTreeClassifier(random_state=1), tuned_parameters
                         cv=5, scoring='accuracy')
clf_gs_dt.fit(X_train, Y_train)
```

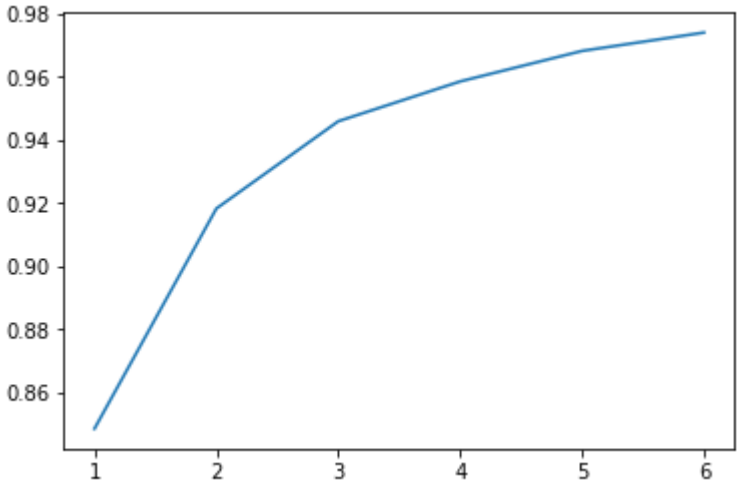
```
    GridSearchCV(cv=5, error_score='raise-deprecating',
           estimator=DecisionTreeClassifier(class_weight=None, criterion='gini'
                max_features=None, max_leaf_nodes=None,
                min_impurity_decrease=0.0, min_impurity_split=None,
                min_samples_leaf=1, min_samples_split=2,
                min_weight_fraction_leaf=0.0, presort=False, random_state=1,
                splitter='best'),
           fit_params=None, iid='warn', n_jobs=None,
           param_grid=[{'max_depth': array([1, 2, 3, 4, 5, 6])}],
```

```
clf_gs_dt.best_params_
```

```
{'max_depth': 6}
```

```
plt.plot(n_range, clf_gs_dt.cv_results_['mean_test_score'])
```

```
[<matplotlib.lines.Line2D at 0x7f7efc033128>]
```



Сравнение моделей после подбора гиперпараметров

Стохастический градиентный спуск

```
sgd_optimized = SGDClassifier(l1_ratio=clf_gs_sgd.best_params_['l1_ratio']).fit(X
predicted_sgd_opt = sgd_optimized.predict(X_test)
```

```
accuracy_score(Y_test, predicted_sgd_opt)
```

```
0.8634721751890733
```

```
balanced_accuracy_score(Y_test, predicted_sgd_opt)
```

```
0.8644777381396946
```

```
(precision_score(Y_test, predicted_sgd_opt, average='weighted'),
 recall_score(Y_test, predicted_sgd_opt, average='weighted'))
```

```
(0.8675323737236463, 0.8634721751890733)
```

```
f1_score(Y_test, predicted_sgd_opt, average='weighted')
```

```
0.8632659477399396
```

Линейный классификатор на основе SVM

```
svm_optimized = LinearSVC(C=clf_gs_svm.best_params_['C']).fit(X_train, Y_train)
predicted_svm_opt = svm_optimized.predict(X_test)
```

```
accuracy_score(Y_test, predicted_svm_opt)
```

```
0.8630206569590247
```

```
balanced_accuracy_score(Y_test, predicted_svm_opt)
```

```
0.8641374586768682
```

```
(precision_score(Y_test, predicted_svm_opt, average='weighted'),
 recall_score(Y_test, predicted_svm_opt, average='weighted'))
```

```
(0.8679448873889508, 0.8630206569590247)
```

```
f1_score(Y_test, predicted_svm_opt, average='weighted')
```

```
0.862751255238221
```

Дерево решений

```
dt_optimized = DecisionTreeClassifier(max_depth=clf_gs_dt.best_params_['max_depth
predicted_dt_opt = dt_optimized.predict(X_test)
```

```
accuracy_score(Y_test, predicted_dt_opt)
```

```
0.9755615757986229
```

```
balanced_accuracy_score(Y_test, predicted_dt_opt)
```

```
0.9757985697019342
```

```
(precision_score(Y_test, predicted_dt_opt, average='weighted'),
 recall_score(Y_test, predicted_dt_opt, average='weighted'))
```

```
(0.9758383679784371, 0.9755615757986229)
```

```
f1_score(Y_test, predicted_dt_opt, average='weighted')
```

```
0.9755642280485889
```

Подбор гиперпараметров позволил увеличить точность работы стохастического градиентного спуска и дерева решений. В случае с деревом решений, точность модели увеличилась

существенно и после подбора гиперпараметров именно эта модель предоставляет наибольшую точность.