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

Отчет
Лабораторная работа № 5

По курсу «Технологии машинного обучения»

«Линейные модели, SVM и деревья решений»

ИСПОЛНИТЕЛЬ:

Сергеев И.В.
Группа ИУ5-64Б

_____

"__"_____2020 г.

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю.Е.

_____

"__"_____2020 г.

Москва 2020

Описание задания

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

Задание:

- Выберите набор данных (датасет) для решения задачи классификации или регрессии.
- В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.
- С использованием метода train_test_split разделите выборку на обучающую и тестовую.
- Обучите следующие модели: одну из линейных моделей; SVM; дерево решений.
- Оцените качество моделей с помощью двух подходящих для задачи метрик. Сравните качество полученных моделей.

Текст программы

Программа разрабатывалась в IDE PyCharm. Ниже приведён полный листинг программы:

```
#%%

from sklearn.model_selection import train_test_split
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
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import cross_val_score, cross_validate, KFold,
LeaveOneOut, GridSearchCV,learning_curve,validation_curve
%matplotlib inline
sns.set(style="ticks")
import mglearn

#%%

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
X, y = mglearn.datasets.make_forge()
X

#%%

fig, axes = plt.subplots(1, 2, figsize=(10, 3))

for model, ax in zip([SVC(), LogisticRegression()], axes):
```

```
 clf = model.fit(X, y)
 mglearn.plots.plot_2d_separator(clf, X, fill=False, eps=0.5,
 ax=ax, alpha=.7)
 mglearn.discrete_scatter(X[:, 0], X[:, 1], y, ax=ax)
 ax.set_title("{}".format(clf.__class__.__name__))
 ax.set_xlabel("Признак 0")
 ax.set_ylabel("Признак 1")
axes[0].legend()

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,
random_state=42)

#%%

logreg = LogisticRegression().fit(X_train, y_train)
score_train = logreg.score(X_train, y_train)
score_test = logreg.score(X_test, y_test)
print("Правильность на обучающем наборе {}".format(score_train))
print("Правильность на тестовом наборе {}".format(score_test))

#%%

decTree = DecisionTreeClassifier(max_depth = 4).fit(X_train, y_train)
score_train_tree = decTree.score(X_train, y_train)
score_test_tree = decTree.score(X_test, y_test)
print("Правильность на обучающем наборе {}".format(score_train_tree))
print("Правильность на тестовом наборе {}".format(score_test_tree))

#%%

linSVM = SVC().fit(X_train, y_train)
score_train_SVM = linSVM.score(X_train, y_train)
score_test_SVM = linSVM.score(X_test, y_test)
print("Правильность на обучающем наборе {}".format(score_train_SVM))
print("Правильность на тестовом наборе {}".format(score_test_SVM))

#%%

print("Логистическая регрессия \n")
for method in [accuracy_score,confusion_matrix, classification_report]:
    print("Метод {}".format(method.__name__))
    print(method(y_test,logreg.predict(X_test)))

#%%

print("SVM \n")
for method in [accuracy_score,confusion_matrix, classification_report]:
    print("Метод {}".format(method.__name__))
    print(method(y_test,linSVM.predict(X_test)))

#%%

print("Tree \n")
for method in [accuracy_score,confusion_matrix, classification_report]:
    print("Метод {}".format(method.__name__))
    print(method(y_test,decTree.predict(X_test)))

#%%

CTP = [{'C':[0.001,0.01,0.1,1,10,100,1000]}]
GammaTP = [{'gamma':[0.001,0.01,0.1,1,10,100,1000]}]
DepthTP = [{'max_depth':[1,2,3,4,5,6,7,8,9,10]}]
```

```
bestParams=[]

for model,tuned_parameters,koef_name in
zip([LogisticRegression(solver='liblinear'),SVC(),DecisionTreeClassifier()],[CTP,Gamm
aTP,DepthTP],['C','gamma','max_depth']):
    modelGS = GridSearchCV(model, tuned_parameters, cv=LeaveOneOut(),
scoring='accuracy')
    modelGS.fit(X_train,y_train)
    print(modelGS.best_params_.get(koef_name))
    bestParams.append(modelGS.best_params_.get(koef_name))

#%%

print('Подобранные гиперпараметры:')
print('    Логистическая регрессия: параметр C: ',bestParams[0])
print('    Метод Опорных Векторов: параметр gamma: ',bestParams[1])
print('    Дерево решений: параметр глубина: ',bestParams[2])

#%%

LogRegBest = LogisticRegression(C=bestParams[0], solver='liblinear')
SVCBest = SVC(gamma = bestParams[1])
DecTreeBest = DecisionTreeClassifier(max_depth = bestParams[2])

#%%

for model in [LogRegBest,SVCBest,DecTreeBest]:
    model.fit(X_train,y_train)

#%%

print("Логистическая регрессия \n")
for method in [accuracy_score,confusion_matrix, classification_report]:
    print("Метод {}".format(method.__name__))
    print(method(y_test,LogRegBest.predict(X_test)))

#%%

print("SVC \n")
for method in [accuracy_score,confusion_matrix, classification_report]:
    print("Метод {}".format(method.__name__))
    print(method(y_test,SVCBest.predict(X_test)))

#%%

print("Дерево решений \n")
for method in [accuracy_score,confusion_matrix, classification_report]:
    print("Метод {}".format(method.__name__))
    print(method(y_test,DecTreeBest.predict(X_test)))

#%%
```

Примеры выполнения программы

Выполнение программы, а также наглядная демонстрация входных и выходных данных (таблиц, графиков и тд) осуществлялась на базе Jupyter Notebook, сервер которого запускался из-под PyCharm. Ниже приведены скриншоты, отражающие работу программы:

jupyter lab5 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

In [1]:
```
from sklearn.model_selection import train_test_split
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
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import cross_val_score, cross_validate, KFold, LeaveOneOut, GridSearchCV,learning_curve,validation_c
%matplotlib inline
sns.set(style="ticks")
import mglearn
```

In [2]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
X, y = mglearn.datasets.make_forge()
X
```

```
c:\users\is-st\appdata\local\programs\python\python37-32\lib\site-packages\sklearn\utils\deprecation.py:86: FutureWarning: Func
tion make_blobs is deprecated; Please import make_blobs directly from scikit-learn
  warnings.warn(msg, category=FutureWarning)
```

Out[2]:
```
array([[ 9.96346605,  4.59676542],
       [11.0329545 , -0.16816717],
       [11.54155807,  5.21116083],
       [ 8.69289001,  1.54322016],
       [ 8.1062269 ,  4.28695977],
       [ 8.30988863,  4.80623966],
       [11.93027136,  4.64866327],
       [ 9.67284681, -0.20283165],
       [ 8.34810316,  5.13415623],
       [ 8.67494727,  4.47573059],
       [ 9.17748385,  5.09283177],
```

jupyter lab5 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

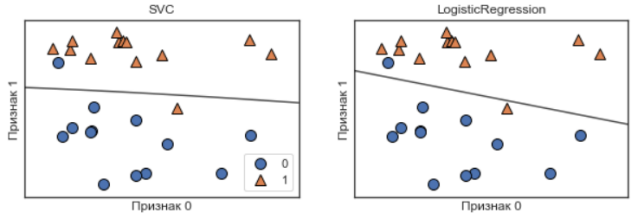
Not Trusted | Python 3

Run Code

In [3]:
```
fig, axes = plt.subplots(1, 2, figsize=(10, 3))

for model, ax in zip([SVC(), LogisticRegression()], axes):
 clf = model.fit(X, y)
 mglearn.plots.plot_2d_separator(clf, X, fill=False, eps=0.5,
 ax=ax, alpha=.7)
 mglearn.discrete_scatter(X[:, 0], X[:, 1], y, ax=ax)
 ax.set_title("{}".format(clf.__class__.__name__))
 ax.set_xlabel("Признак 0")
 ax.set_ylabel("Признак 1")
axes[0].legend()

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, random_state=42)
```



In [4]:
```
logreg = LogisticRegression().fit(X_train, y_train)
score_train = logreg.score(X_train, y_train)
score_test = logreg.score(X_test, y_test)
print("Правильность на обучающем наборе {}".format(score_train))
print("Правильность на тестовом наборе {}".format(score_test))
```

```
Правильность на обучающем наборе 0.9473684210526315
Правильность на тестовом наборе 0.8571428571428571
```

jupyter lab5 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

In [5]:
```
decTree = DecisionTreeClassifier(max_depth = 4).fit(X_train, y_train)
score_train_tree = decTree.score(X_train, y_train)
score_test_tree = decTree.score(X_test, y_test)
print("Правильность на обучающем наборе {}".format(score_train_tree))
print("Правильность на тестовом наборе {}".format(score_test_tree))
```

```
Правильность на обучающем наборе 1.0
Правильность на тестовом наборе 0.8571428571428571
```

In [6]:
```
linSVM = SVC().fit(X_train, y_train)
score_train_SVM = linSVM.score(X_train, y_train)
score_test_SVM = linSVM.score(X_test, y_test)
print("Правильность на обучающем наборе {}".format(score_train_SVM))
print("Правильность на тестовом наборе {}".format(score_test_SVM))
```

```
Правильность на обучающем наборе 0.9473684210526315
Правильность на тестовом наборе 0.8571428571428571
```

In [7]:
```
print("Логистическая регрессия \n")
for method in [accuracy_score,confusion_matrix, classification_report]:
    print("Метод {}".format(method.__name__))
    print(method(y_test,logreg.predict(X_test)))
```

```
Логистическая регрессия

Метод accuracy_score
0.8571428571428571
Метод confusion_matrix
[[4 0]
 [1 2]]
Метод classification_report
              precision    recall  f1-score   support

           0       0.80      1.00      0.89         4
```

jupyter lab5 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

In [7]:
```
print("Логистическая регрессия \n")
for method in [accuracy_score,confusion_matrix, classification_report]:
    print("Метод {}".format(method.__name__))
    print(method(y_test,logreg.predict(X_test)))
```

```
Логистическая регрессия

Метод accuracy_score
0.8571428571428571
Метод confusion_matrix
[[4 0]
 [1 2]]
Метод classification_report
              precision    recall  f1-score   support

           0       0.80      1.00      0.89         4
           1       1.00      0.67      0.80         3

    accuracy                           0.86         7
   macro avg       0.90      0.83      0.84         7
weighted avg       0.89      0.86      0.85         7
```

In [8]:
```
print("SVM \n")
for method in [accuracy_score,confusion_matrix, classification_report]:
    print("Метод {}".format(method.__name__))
    print(method(y_test,linSVM.predict(X_test)))
```

```
SVM

Метод accuracy_score
0.8571428571428571
Метод confusion_matrix
[[4 0]
 [1 2]]
Метод classification_report
```

jupyter lab5 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

In [8]:
```
print("SVM \n")
for method in [accuracy_score,confusion_matrix, classification_report]:
    print("Метод {}".format(method.__name__))
    print(method(y_test,linSVM.predict(X_test)))
```

```
SVM

Метод accuracy_score
0.8571428571428571
Метод confusion_matrix
[[4 0]
 [1 2]]
Метод classification_report
              precision    recall  f1-score   support

           0       0.80      1.00      0.89         4
           1       1.00      0.67      0.80         3

    accuracy                           0.86         7
   macro avg       0.90      0.83      0.84         7
weighted avg       0.89      0.86      0.85         7
```

In [9]:
```
print("Tree \n")
for method in [accuracy_score,confusion_matrix, classification_report]:
    print("Метод {}".format(method.__name__))
    print(method(y_test,decTree.predict(X_test)))
```

```
Tree

Метод accuracy_score
0.8571428571428571
Метод confusion_matrix
[[4 0]
 [1 2]]
Метод classification_report
```

jupyter lab5 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

```
    print(method(y_test,decTree.predict(X_test)))
```

Tree

Метод accuracy_score
0.8571428571428571
Метод confusion_matrix
[[4 0]
 [1 2]]
Метод classification_report
              precision   recall f1-score   support

           0   0.80   1.00   0.89   4
           1   1.00   0.67   0.80   3

    accuracy   0.86   7
   macro avg   0.90   0.83   0.84   7
weighted avg   0.89   0.86   0.85   7

In [10]:
```
CTP = [{'C':[0.001,0.01,0.1,1,10,100,1000]}]
GammaTP = [{'gamma':[0.001,0.01,0.1,1,10,100,1000]}]
DepthTP = [{'max_depth':[1,2,3,4,5,6,7,8,9,10]}]
bestParams=[]

for model,tuned_parameters,koef_name in zip([LogisticRegression(solver='liblinear'),SVC(),DecisionTreeClassifier()],[CTP,GammaTP,
    modelGS = GridSearchCV(model, tuned_parameters, cv=LeaveOneOut(), scoring='accuracy')
    modelGS.fit(X_train,y_train)
    print(modelGS.best_params_.get(koef_name))
    bestParams.append(modelGS.best_params_.get(koef_name))
```

```
0.1
0.01
1
```

jupyter lab5 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

In [11]:
```
print('Подобранные гиперпараметры:')
print('    Логистическая регрессия: параметр C: ',bestParams[0])
print('    Метод Опорных Векторов: параметр gamma: ',bestParams[1])
print('    Дерево решений: параметр глубина: ',bestParams[2])
```

```
Подобранные гиперпараметры:
    Логистическая регрессия: параметр C:  0.1
    Метод Опорных Векторов: параметр gamma:  0.01
    Дерево решений: параметр глубина:  1
```

In [12]:
```
LogRegBest = LogisticRegression(C=bestParams[0], solver='liblinear')
SVCBest = SVC(gamma = bestParams[1])
DecTreeBest = DecisionTreeClassifier(max_depth = bestParams[2])
```

In [13]:
```
for model in [LogRegBest,SVCBest,DecTreeBest]:
    model.fit(X_train,y_train)
```

In [14]:
```
print("Логистическая регрессия \n")
for method in [accuracy_score,confusion_matrix, classification_report]:
    print("Метод {}".format(method.__name__))
    print(method(y_test,LogRegBest.predict(X_test)))
```

```
Логистическая регрессия

Метод accuracy_score
0.8571428571428571
Метод confusion_matrix
[[4 0]
 [1 2]]
Метод classification_report
              precision    recall  f1-score   support

           0       0.80      1.00      0.89         4
           1       1.00      0.67      0.80         3
```

jupyter lab5 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

In [15]:
```
print("SVC \n")
for method in [accuracy_score,confusion_matrix, classification_report]:
    print("Метод {}".format(method.__name__))
    print(method(y_test,SVCBest.predict(X_test)))
```

```
SVC

Метод accuracy_score
0.8571428571428571
Метод confusion_matrix
[[4 0]
 [1 2]]
Метод classification_report
              precision    recall  f1-score   support

           0       0.80      1.00      0.89         4
           1       1.00      0.67      0.80         3

    accuracy                           0.86         7
   macro avg       0.90      0.83      0.84         7
weighted avg       0.89      0.86      0.85         7
```

In [16]:
```
print("Дерево решений \n")
for method in [accuracy_score,confusion_matrix, classification_report]:
    print("Метод {}".format(method.__name__))
    print(method(y_test,DecTreeBest.predict(X_test)))
```

```
Дерево решений

Метод accuracy_score
0.8571428571428571
Метод confusion_matrix
[[4 0]
 [1 2]]
```

jupyter lab5 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3

Run Code

```
weighted avg       0.89      0.86      0.85         7
```

In [16]:

```
print("Дерево решений \n")
for method in [accuracy_score,confusion_matrix, classification_report]:
    print("Метод {}".format(method.__name__))
    print(method(y_test,DecTreeBest.predict(X_test)))
```

```
Дерево решений

Метод accuracy_score
0.8571428571428571
Метод confusion_matrix
[[4 0]
 [1 2]]
Метод classification_report
              precision    recall  f1-score   support

           0       0.80      1.00      0.89         4
           1       1.00      0.67      0.80         3

    accuracy                           0.86         7
   macro avg       0.90      0.83      0.84         7
weighted avg       0.89      0.86      0.85         7
```

In [16]: