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)

Методы машинного обучения

Отчёт по лабораторной работе № 5

«Линейные модели, SVM и деревья решений»

Выполнил:
студент группы ИУ5 – 23М Крутов Т.Ю.

Преподаватель: Гапанюк Ю.Е.

2020г.

```
import pandas as pd
pd.set_option('display.max.columns', 100)
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_digits, load_wine, load_diabetes
from sklearn.model_selection import train_test_split
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 mean_absolute_error, mean_squared_error, mean_squared_log_erro
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import SGDRegressor
from sklearn.linear_model import SGDClassifier
from sklearn.svm import SVC, NuSVC, LinearSVC, OneClassSVM, SVR, NuSVR, LinearSVR
%matplotlib inline
sns.set(rc={'figure.figsize':(16,8)})


df = pd.read_csv('titanic.csv', error_bad_lines=False, comment='#')
#df.drop(columns='id', inplace = True);
df.drop(columns='Cabin', inplace= True)
for col in df.columns:
    count = df[df[col].isnull()].shape[0]
    print('{}  -  {}'.format(col, count))
print ('{} - размер датасета'.format(df.shape))
```

```
PassengerId  -  0
Survived  -  0
Pclass  -  0
Name  -  0
Sex  -  0
Age  -  177
SibSp  -  0
Parch  -  0
Ticket  -  0
Fare  -  0
Embarked  -  2
(891, 11) - размер датасета
```

```
df = df.dropna(axis=0, how='any')
(df.shape)
```

```
(712, 11)
```

```
#df.drop(columns=['Ticket', 'Name'], inplace= True)
df.reset_index(drop= True, inplace=True)
df
```

	PassengerId	Survived	Pclass	Name	Sex
0	1	0	3	Braund, Mr. Owen Harris	male
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female
2	3	1	3	Heikkinen, Miss. Laina	female
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female
4	5	0	3	Allen, Mr. William Henry	male
...	...	...	...	...	...
707	886	0	3	Rice, Mrs. William (Margaret Norton)	female
708	887	0	2	Montvila, Rev. Juozas	male
709	888	1	1	Graham, Miss. Margaret Edith	female
710	890	1	1	Behr, Mr. Karl Howell	male
711	891	0	3	Dooley, Mr. Patrick	male

Закодируем категориальные признаки

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
le = LabelEncoder()
ohe = OneHotEncoder()
sex = le.fit_transform(df['Sex'])
sex = pd.DataFrame({'Sex':sex.T})
Embarked = le.fit_transform(df['Embarked'])
Embarked = pd.DataFrame({'Embarked' : Embarked.T })
sex
```

Sex

```
del df['Sex']
del df['Embarked']
df = df.join(sex)
df = df.join(Embarked)
df
```

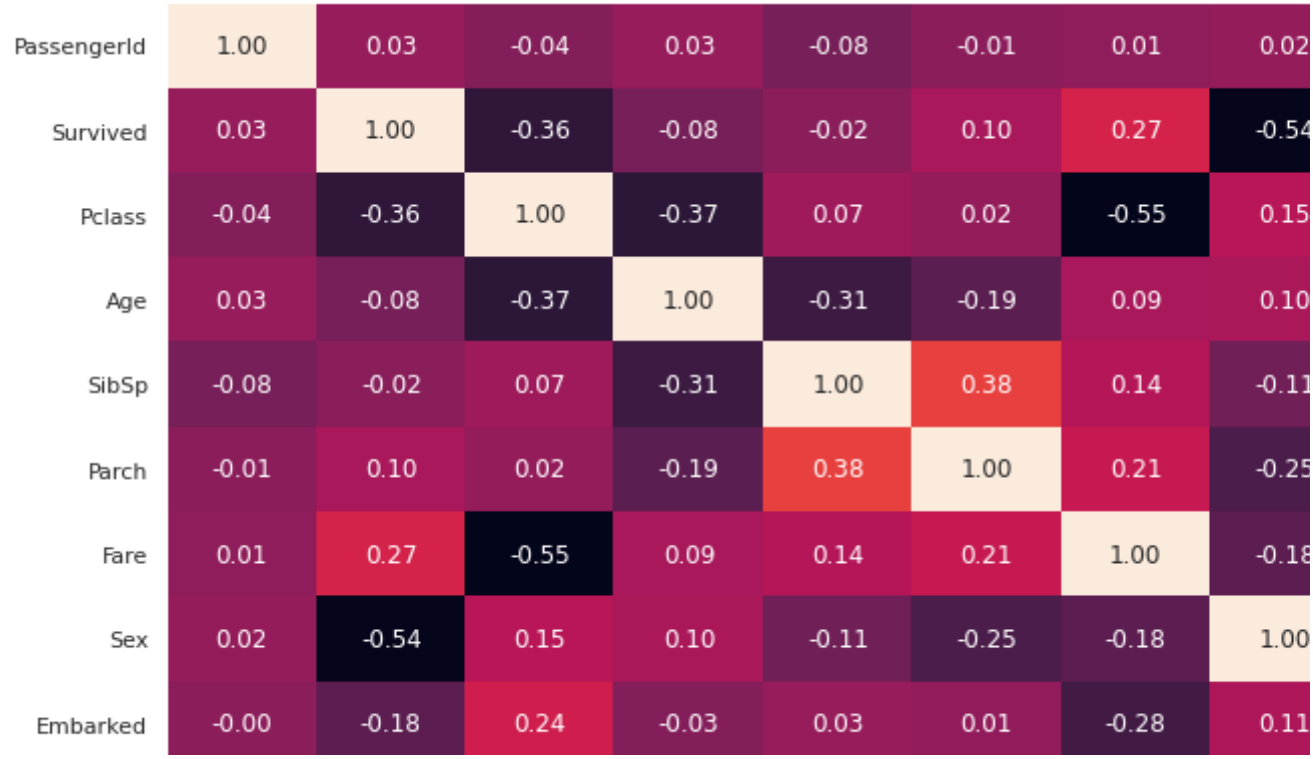
	PassengerId	Survived	Pclass	Name	Age
0	1	0	3	Braund, Mr. Owen Harris	22.0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	38.0
2	3	1	3	Heikkinen, Miss. Laina	26.0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	35.0
4	5	0	3	Allen, Mr. William Henry	35.0
...	...	...	...	...	...
707	886	0	3	Rice, Mrs. William (Margaret Norton)	39.0
708	887	0	2	Montvila, Rev. Juozas	27.0
709	888	1	1	Graham, Miss. Margaret Edith	19.0
710	890	1	1	Behr, Mr. Karl Howell	26.0
711	891	0	3	Dooley, Mr. Patrick	32.0

712 rows × 11 columns

```
df.drop(columns=['Ticket', 'Name'], inplace= True)
```

```
fig, ax = plt.subplots(figsize=(15,7))
sns.heatmap(df.corr(method='pearson'), ax=ax, annot=True, fmt='.2f')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f389d8e04e0>
```




```
X = df.drop(columns='Survived')
Y = df.Survived
```

Логистическая регрессия

```
from sklearn.model_selection import train_test_split, KFold, cross_val_score, GridSearchCV

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2)


from sklearn.linear_model import LogisticRegression
clf_lr = LogisticRegression(max_iter=100000)
clf_lr.fit(X_train, y_train)


clf_lr.score(X_train, y_train)
```

```
0.8031634446397188
```

```
from sklearn.metrics import accuracy_score as acuracy, precision_score as precision, recal

print('accuracy_score для обучающего набора:', acuracy(y_train, clf_lr.predict(X_train)))
print('accuracy_score для тестового набора :', acuracy(y_test, clf_lr.predict(X_test)), '\
print('precission_score для обучающего набора:', precision(y_train, clf_lr.predict(X_train
print('precission_score для тестового набора :', precision(y_test, clf_lr.predict(X_test))
print('recall_score для обучающего набора:', recall(y_train, clf_lr.predict(X_train)))
print('recall_score для тестового набора :', recall(y_test, clf_lr.predict(X_test)), '\n')
```

```
accuracy_score для обучающего набора: 0.8031634446397188
accuracy_score для тестового набора : 0.8181818181818182

precission_score для обучающего набора: 0.7788461538461539
precission_score для тестового набора : 0.8148148148148148

recall_score для обучающего набора: 0.7105263157894737
```

SVM

```
from sklearn.svm import SVC
clf_svc = SVC(kernel='rbf', C=1)
%time clf_svc.fit(X_train, y_train)
```

```
CPU times: user 17.6 ms, sys: 6 µs, total: 17.6 ms
Wall time: 22.2 ms
SVC(C=1, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
print('accuracy_score тдля обучающего набораrain:', acuracy(y_train, clf_svc.predict(X_tra
print('accuracy_score для тестового набора :', acuracy(y_test, clf_svc.predict(X_test)), '
print('precission_score для обучающего набора:', precision(y_train, clf_svc.predict(X_trai
print('precission_score для тестового набора :', precision(y_test, clf_svc.predict(X_test)
print('recall_score для обучающего набора:', recall(y_train, clf_svc.predict(X_train)))
print('recall_score для тестового набора :', recall(y_test, clf_svc.predict(X_test)), '\n'
```

```
accuracy train: 0.6467486818980668
accuracy test : 0.6293706293706294

precission train: 0.8
precission test : 0.7692307692307693

recall train: 0.15789473684210525
recall test : 0.16666666666666666
```

Деревья решений

```
from sklearn.tree import DecisionTreeClassifier
clf_dt = DecisionTreeClassifier(max_depth=25, min_samples_split=2, min_samples_leaf=1)
clf_dt.fit(X_train, y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=25, 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='deprecated',
                       random_state=None, splitter='best')
```

```
print('accuracy_score для обучающего набора:', acuracy(y_train, clf_dt.predict(X_train)))
```

```
print('accuracy_score для тестового набора :', acuracy(y_test, clf_dt.predict(X_test)), '\
print('precission_score для обучающего набора:', precision(y_train, clf_dt.predict(X_train
print('precission_score для тестового набора :', precision(y_test, clf_dt.predict(X_test))
print('recall_score для обучающего набора:', recall(y_train, clf_dt.predict(X_train)))
print('recall_score для тестового набора :', recall(y_test, clf_dt.predict(X_test)), '\n')
```

```
accuracy_score для обучающего набора: 1.0
accuracy_score для тестового набора : 0.7342657342657343

precission_score для обучающего набора: 1.0
precission_score для тестового набора : 0.6896551724137931

recall_score для обучающего набора: 1.0
recall_score для тестового набора : 0.6666666666666666
```

GridSearchCV

Используем функцию GridSearchCV для подбора гиперпараметров

```
from sklearn.model_selection import GridSearchCV, KFold
from sklearn.linear_model import LogisticRegression
params_l = {'max_iter' : [50000], 'C' : [x for x in 10.0**np.arange(-5, 5)]}
grsrch_lr = GridSearchCV(estimator=LogisticRegression(), param_grid=params_l, cv=KFold(ran
```

```
grsrch_lr.fit(X_train, y_train)
```

```
GridSearchCV(cv=KFold(n_splits=5, random_state=1, shuffle=True),
             error_score=nan,
             estimator=LogisticRegression(C=1.0, class_weight=None, dual=False,
                                          fit_intercept=True,
                                          intercept_scaling=1, l1_ratio=None,
                                          max_iter=100, multi_class='auto',
                                          n_jobs=None, penalty='l2',
                                          random_state=None, solver='lbfgs',
                                          tol=0.0001, verbose=0,
                                          warm_start=False),
             iid='deprecated', n_jobs=None,
             param_grid={'C': [1e-05, 0.0001, 0.001, 0.01, 0.1, 1.0, 10.0,
                               100.0, 1000.0, 10000.0],
                         'max_iter': [50000]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

```
grsrch_lr.best_params_
```

```
{'C': 1.0, 'max_iter': 50000}
```

```
print('accuracy_score для обучаещего набора:', acuracy(y_train, grsrch_lr.best_estimator_.
print('accuracy_score для тестового набора :', acuracy(y_test, grsrch_lr.best_estimator_.p
print('precission_score для обучаещего набора:', precision(y_train, grsrch_lr.best_estimat
```

```
print('precission_score для тестового набора :', precision(y_test, grsrch_lr.best_estimato
print('recall_score для обучаещего набора:', recall(y_train, grsrch_lr.best_estimator_.pre
print('recall_score для тестового набора :', recall(y_test, grsrch_lr.best_estimator_.pred
```

```
accuracy_score для обучаещего набора: 0.8031634446397188
accuracy_score для тестового набора : 0.8181818181818182

precission_score для обучаещего набора: 0.7788461538461539
precission_score для тестового набора : 0.8148148148148148

recall_score для обучаещего набора: 0.7105263157894737
recall_score для тестового набора : 0.7333333333333333
```

```
params_svc = {'kernel' : ['linear', 'poly']}
grsrch_svc = GridSearchCV(estimator=SVC(), param_grid=params_svc, cv=KFold(n_splits=5, ran
```

```
grsrch_svc.fit(X_train, y_train)
```

```
GridSearchCV(cv=KFold(n_splits=5, random_state=1, shuffle=True),
             error_score=nan,
             estimator=SVC(C=1.0, break_ties=False, cache_size=200,
                           class_weight=None, coef0=0.0,
                           decision_function_shape='ovr', degree=3,
                           gamma='scale', kernel='rbf', max_iter=-1,
                           probability=False, random_state=None, shrinking=True,
                           tol=0.001, verbose=False),
             iid='deprecated', n_jobs=None,
             param_grid={'kernel': ['linear', 'poly']}, pre_dispatch='2*n_jobs',
             refit=True, return_train_score=False, scoring=None, verbose=0)
```

```
print('accuracy_score для обучающего набора:', acuracy(y_train, grsrch_svc.best_estimator_
print('accuracy_score для тестового набора :', acuracy(y_test, grsrch_svc.best_estimator_.
print('precission_score для обучающего набора:', precision(y_train, grsrch_svc.best_estima
print('precission_score для тестового набора :', precision(y_test, grsrch_svc.best_estimat
print('recall_score для обучающего набора:', recall(y_train, grsrch_svc.best_estimator_.pr
print('recall_score для тестового набора :', recall(y_test, grsrch_svc.best_estimator_.pre
```

```
accuracy_score для обучающего набора: 0.7908611599297012
accuracy_score для тестового набора : 0.7762237762237763

precission_score для обучающего набора: 0.7738693467336684
precission_score для тестового набора : 0.7592592592592593

recall_score для обучающего набора: 0.6754385964912281
recall_score для тестового набора : 0.6833333333333333
```

```
params_dt = {'max_depth' : np.arange(10, 20, 2),
             'min_samples_split' : np.arange(2, 8, 1)}

grsrch_dt =  GridSearchCV(estimator=DecisionTreeClassifier(), param_grid=params_dt, cv=KFo
```

```
grsrch_dt.fit(X_train, y_train)
```

```
GridSearchCV(cv=KFold(n_splits=5, random_state=1, shuffle=True),
             error_score=nan,
             estimator=DecisionTreeClassifier(ccp_alpha=0.0, 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='deprecated',
                                              random_state=None,
                                              splitter='best'),
             iid='deprecated', n_jobs=None,
             param_grid={'max_depth': array([10, 12, 14, 16, 18]),
                         'min_samples_split': array([2, 3, 4, 5, 6, 7])},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

```
print('accuracy_score для обучающего набора:', acuracy(y_train, grsrch_dt.best_estimator_.
print('accuracy_score для тестового набора :', acuracy(y_test, grsrch_dt.best_estimator_.p
print('precission_score для обучающего набора:', precision(y_train, grsrch_dt.best_estimat
print('precission_score для тестового набора :', precision(y_test, grsrch_dt.best_estimato
print('recall_score для обучающего набора:', recall(y_train, grsrch_dt.best_estimator_.pre
print('recall_score для тестового набора :', recall(y_test, grsrch_dt.best_estimator_.pred
```

```
accuracy_score для обучающего набора: 0.9824253075571178
accuracy_score для тестового набора : 0.7412587412587412

precission_score для обучающего набора: 1.0
precission_score для тестового набора : 0.7090909090909091

recall_score для обучающего набора: 0.956140350877193
recall_score для тестового набора : 0.65
```

Как видно из значений метрик качества, после примения метода GridSea улучшилось

```
def plot_tree_regression(X_train, y_train, X_test):
    """
    Построение деревьев и вывод графиков для заданного датасета
    """

    # Обучение регрессионной модели
    regr_1 = DecisionTreeRegressor(max_depth=3)
    regr_2 = DecisionTreeRegressor(max_depth=10)
    regr_1.fit(X_train, y_train)
    regr_2.fit(X_train, y_train)

    # Предсказание
```

```
        y_1 = regr_1.predict(X_test)
        y_2 = regr_2.predict(X_test)

        # Вывод графика
        fig, ax = plt.subplots(figsize=(15,7))
        plt.scatter(X_train, y_train, s=20, edgecolor="black", c="darkorange", label="Данные")
        plt.plot(X_test, y_1, color="cornflowerblue", label="max_depth=3", linewidth=2)
        plt.plot(X_test, y_2, color="yellowgreen", label="max_depth=10", linewidth=2)
        plt.xlabel("Данные")
        plt.ylabel("Целевой признак")
        plt.title("Регрессия на основе дерева решений")
        plt.legend()
        plt.show()


    Image(get_png_tree(boston_tree_regr, df_boston.columns), height="500")
```