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

Лабораторная работа №5
по дисциплине
# «Методы машинного обучения»
на тему
# «Линейные модели, SVM и деревья решений»

Выполнил:
студент группы ИУ5-22М
Ромичева Е.

Москва — 2019 г.

## 0.1. 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.

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        from sklearn.model_selection import GridSearchCV
        from sklearn.model_selection import learning_curve, validation_curve
        from sklearn.model_selection import KFold, RepeatedKFold, LeaveOneOut,
        from sklearn.model_selection import cross_val_score, cross_validate
        from sklearn.metrics import roc_curve,confusion_matrix, roc_auc_score,

        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import classification_report
        from sklearn.svm import SVC
        from sklearn.model_selection import cross_val_score
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.linear_model import LinearRegression

        import warnings

        warnings.filterwarnings('ignore')
        plt.style.use('ggplot')
```

### 0.1.1. Загрузка данных

```
In [ ]: data = pd.read_csv('diabetes.csv')
        data.head()

In [4]: data.dtypes

Out[4]: Pregnancies                 int64
        Glucose                     int64
        BloodPressure               int64
        SkinThickness               int64
        Insulin                     int64
        BMI                       float64
        DiabetesPedigreeFunction  float64
        Age                         int64
```

```
Outcome                       int64
dtype: object
```

```
In [5]: for col in data.columns:
            print('{} - {}'.format(col, data[data[col].isnull()].shape[0]))
```

```
Pregnancies - 0
Glucose - 0
BloodPressure - 0
SkinThickness - 0
Insulin - 0
BMI - 0
DiabetesPedigreeFunction - 0
Age - 0
Outcome - 0
```

```
In [6]: data.shape
```

```
Out[6]: (768, 9)
```

```
In [8]: CLASS = 'Outcome'
        RANDOM_STATE = 17
        TEST_SIZE = 0.3

        X = data.drop(CLASS, axis=1).values
        Y = data[CLASS].values

        X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=TE
        print('X_train: {}'.format(X_train.shape))
        print('X_test: {}'.format(X_test.shape))
```

```
X_train: (537, 8)
X_test: (231, 8)
```

## 0.2. Обучение

### 0.2.1. Метод опрных векторов

```
In [9]: clf = SVC(gamma='auto')
        clf.fit(X_train, Y_train)
        clf.score(X_test, Y_test)
```

```
Out[9]: 0.6493506493506493
```

```
In [10]: Y_pred = clf.predict(X_test)
         print(classification_report(Y_test, Y_pred))
```

```
              precision    recall  f1-score   support

           0       0.65      1.00      0.79       150
```

```
           1       0.00      0.00      0.00        81

   micro avg       0.65      0.65      0.65       231
   macro avg       0.32      0.50      0.39       231
weighted avg       0.42      0.65      0.51       231
```

#### 0.2.2. Дерево решений

```
In [11]: tree = DecisionTreeClassifier(random_state=0)
         tree.fit(X_train, Y_train)
         tree.score(X_test, Y_test)
```

```
Out[11]: 0.6926406926406926
```

```
In [12]: Y_pred = tree.predict(X_test)
         print(classification_report(Y_test, Y_pred))
```

```
              precision    recall  f1-score   support

           0       0.78      0.73      0.75       150
           1       0.55      0.63      0.59        81

   micro avg       0.69      0.69      0.69       231
   macro avg       0.67      0.68      0.67       231
weighted avg       0.70      0.69      0.70       231
```

#### 0.2.3. Линейная регрессия

```
In [13]: lin = LinearRegression()
         lin.fit(X_train, Y_train)
         lin.score(X_test, Y_test)
```

```
Out[13]: 0.26790683521502
```

### 0.3. Подбор гиперпараметра с использованием GridSearchCV и кросс-валидации

#### 0.3.1. Метод опрных векторов

```
In [14]: CROSS_VALIDATOR_GENERATOR = 5
         PARAMETER_TAG = 'C'
         PARAMETER_MAX_VALUE = 3

         param_grid = {PARAMETER_TAG : np.arange(0.01, PARAMETER_MAX_VALUE, 0.
         clf = SVC(gamma='auto')

         clf_cv = GridSearchCV(clf, param_grid, cv = CROSS_VALIDATOR_GENERATOR
         clf_cv.fit(X_train,Y_train)
         clf_cv.best_score_
```

```
Out[14]: 0.6517690875232774

In [15]: clf_cv.best_params_

Out[15]: {'C': 0.01}

In [16]: clf = SVC(gamma='auto', C = clf_cv.best_params_[PARAMETER_TAG])
         clf.fit(X_train, Y_train)
         clf.score(X_test, Y_test)

Out[16]: 0.6493506493506493

In [17]: Y_pred = clf.predict(X_test)
         print(classification_report(Y_test, Y_pred))

              precision    recall  f1-score   support

           0       0.65      1.00      0.79       150
           1       0.00      0.00      0.00        81

   micro avg       0.65      0.65      0.65       231
   macro avg       0.32      0.50      0.39       231
weighted avg       0.42      0.65      0.51       231
```

### 0.3.2. Дерево решений

```
In [18]: PARAMETER_TAG = 'min_impurity_decrease'

         param_grid = {PARAMETER_TAG : np.arange(0.01, PARAMETER_MAX_VALUE, 0.
         tree = DecisionTreeClassifier(random_state=0)

         tree_cv = GridSearchCV(tree, param_grid, cv = CROSS_VALIDATOR_GENERAT
         tree_cv.fit(X_train,Y_train)
         tree_cv.best_score_

Out[18]: 0.7374301675977654

In [19]: tree_cv.best_params_

Out[19]: {'min_impurity_decrease': 0.01}

In [20]: tree = DecisionTreeClassifier(random_state=0, min_impurity_decrease =
         tree.fit(X_train, Y_train)
         tree.score(X_test, Y_test)

Out[20]: 0.7489177489177489

In [21]: Y_pred = tree.predict(X_test)
         print(classification_report(Y_test, Y_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.79      0.80       150
           1       0.64      0.67      0.65        81

   micro avg       0.75      0.75      0.75       231
   macro avg       0.73      0.73      0.73       231
weighted avg       0.75      0.75      0.75       231
```

### 0.3.3. Линейная регрессия

```
In [22]: PARAMETER_TAG = 'n_jobs'

         param_grid = {PARAMETER_TAG : np.arange(1, 100)}
         lin = LinearRegression()


         lin_cv = GridSearchCV(lin, param_grid, cv = CROSS_VALIDATOR_GENERATOR
         lin_cv.fit(X_train, Y_train)
         lin_cv.best_score_

Out[22]: 0.27434664756854726

In [23]: lin_cv.best_params_

Out[23]: {'n_jobs': 1}

In [24]: lin = LinearRegression(n_jobs = lin_cv.best_params_[PARAMETER_TAG])
         lin.fit(X_train, Y_train)
         lin.score(X_test, Y_test)

Out[24]: 0.26790683521502
```

### 0.3.4. Результаты

Наилиучший вариант: дерево решений.
Наихудший вариант: линейная регрессия