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 вычислительная техника»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчет по лабораторной работе №4

«Линейные модели, SVM и деревья решений»

Выполнил:
студент группы ИУ5-63Б

Рыбина А.Д.
Подпись и дата:

Проверил:
преподаватель каф. ИУ5

Гапанюк Ю.Е.
Подпись и дата:

Москва, 2022 г.

Цель лабораторной работы

изучение линейных моделей, SVM и деревьев решений.

Описание задания

1. Выберите набор данных (датасет) для решения задачи классификации или регрессии.
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 следующие модели:

    - одну из линейных моделей (линейную или полиномиальную регрессию при решении задачи регрессии, логистическую регрессию при решении задачи классификации);
    - SVM;
    - дерево решений.

5. Оцените качество моделей с помощью двух подходящих для задачи метрик. Сравните качество полученных моделей.
6. Постройте график, показывающий важность признаков в дереве решений.
7. Визуализируйте дерево решений или выведите правила дерева решений в текстовом виде.

Текст программы и результаты ее выполнение

Лабораторная работа №4. Линейные модели, SVM и деревья решений.

Задание

1. Выберите набор данных (датасет) для решения задачи классификации или регрессии.
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 следующие модели:
   - одну из линейных моделей (линейную или полиномиальную регрессию при решении задачи регрессии, логистическую регрессию при решении задачи классификации);
   - SVM;
   - дерево решений.
5. Оцените качество моделей с помощью двух подходящих для задачи метрик. Сравните качество полученных моделей.
6. Постройте график, показывающий важность признаков в дереве решений.
7. Визуализируйте дерево решений или выведите правила дерева решений в текстовом виде.

In [14]:
```
from IPython.display import Image
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_diabetes
from sklearn.metrics import mean_absolute_error, r2_score
from sklearn.linear_model import Lasso
from sklearn.preprocessing import MinMaxScaler
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

In [15]:
```
# 1.Формирование обучающей и тестовой выборки
diab = load_diabetes()

diab_df = pd.DataFrame(data = np.c_[diab['data'], diab['target']],
                       columns = diab['feature_names'] + ['target'])

diab_df.isnull().any().any()
```

Out[15]: False

In [16]:
```
diab_df.describe()
```

Out[16]:

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6	t
count	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	4.420000e+02	442.00
mean	-2.511817e-19	1.230790e-17	-2.245564e-16	-4.797570e-17	-1.381499e-17	3.918434e-17	-5.777179e-18	-9.042540e-18	9.293722e-17	1.130318e-17	152.13
std	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	4.761905e-02	77.09
min	-1.072256e-01	-4.464164e-02	-9.027530e-02	-1.123988e-01	-1.267807e-01	-1.156131e-01	-1.023071e-01	-7.639450e-02	-1.260971e-01	-1.377672e-01	25.00
25%	-3.729927e-02	-4.464164e-02	-3.422907e-02	-3.665608e-02	-3.424784e-02	-3.035840e-02	-3.511716e-02	-3.949338e-02	-3.324559e-02	-3.317903e-02	87.00
50%	5.383060e-03	-4.464164e-02	-7.283766e-03	-5.670422e-03	-4.320866e-03	-3.819065e-03	-6.584468e-03	-2.592262e-03	-1.947171e-03	-1.077698e-03	140.50
75%	3.807591e-02	5.068012e-02	3.124802e-02	3.564379e-02	2.835801e-02	2.984439e-02	2.931150e-02	3.430886e-02	3.243232e-02	2.791705e-02	211.50
max	1.107267e-01	5.068012e-02	1.705552e-01	1.320436e-01	1.539137e-01	1.987880e-01	1.811791e-01	1.852344e-01	1.335973e-01	1.356118e-01	346.00

In [17]:
```
diab_X_train, diab_X_test, diab_y_train, diab_y_test = train_test_split(
    diab.data, diab.target, test_size=0.2, random_state=1)
```

In [18]:
```
# 2. Обучение линейной модели
reg = Lasso(alpha=0.3)
res = reg.fit(diab_X_train, diab_y_train)
res.coef_, res.intercept_
```

Out[18]:
```
(array([   0.        ,  -29.04276213,  522.19211551,  185.22048746,
          -0.        ,   -0.        , -142.29250916,    0.        ,
         430.32958915,    0.        ]),
 151.85914486120444)
```

In [19]:
```
# Оценка качества модели
r2_score(diab_y_test, res.predict(diab_X_test)), mean_absolute_error(diab_y_test, res.predict(diab_X_test))
```

Out[19]: (0.4224494036251901, 43.950963695524834)

In [20]:
```
# 3. Обучение SVM (SVR)
# Масштабирование данных
sc = MinMaxScaler()
sc_data = sc.fit_transform(diab.data)
sc_data[0]
```

Out[20]:
```
array([0.66666667, 1.        , 0.58264463, 0.54929577, 0.29411765,
       0.25697211, 0.20779221, 0.28208745, 0.562217  , 0.43939394])
```

In [21]:
```
# Разделение на тестовую и обучающие выборки
diab_X_train1, diab_X_test1, diab_y_train1, diab_y_test1 = train_test_split(
    sc_data, diab.target, test_size=0.5, random_state=1)

# Обучение SVR
svr = SVR(kernel='poly')
svr.fit(diab_X_train1, diab_y_train1)
```

Out[21]: SVR(kernel='poly')

In [22]:
```
# Оценка качества модели
r2_score(diab_y_test1, svr.predict(diab_X_test1)), mean_absolute_error(diab_y_test1, svr.predict(diab_X_test1))
```

Out[22]: (0.4092880494475172, 45.28288000789277)

In [23]:
```
# 4. Обучение дерева решений
dtr = DecisionTreeRegressor(max_depth=5, criterion='poisson')
dtr.fit(diab_X_train, diab_y_train)
```

Out[23]: DecisionTreeRegressor(criterion='poisson', max_depth=5)

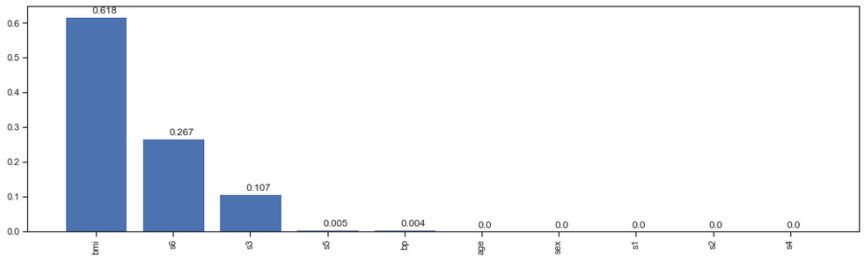
In [24]:
```
# Оценка качества модели
r2_score(diab_y_test, dtr.predict(diab_X_test)), mean_absolute_error(diab_y_test, dtr.predict(diab_X_test))
```

Out[24]: (0.08654337289278313, 59.64016853932585)

In [25]:
```
from operator import itemgetter

def draw_feature_importances(tree_model, X_dataset, figsize=(18,5)):
    """
    Вывод важности признаков в виде графика
    """
    # Сортировка значений важности признаков по убыванию
    list_to_sort = list(zip(X_dataset.columns.values, tree_model.feature_importances_))
    sorted_list = sorted(list_to_sort, key=itemgetter(1), reverse = True)
    # Названия признаков
    labels = [x for x,_ in sorted_list]
    # Важности признаков
    data = [x for _,x in sorted_list]
    # Вывод графика
    fig, ax = plt.subplots(figsize=figsize)
    ind = np.arange(len(labels))
    plt.bar(ind, data)
    plt.xticks(ind, labels, rotation='vertical')
    # Вывод значений
    for a,b in zip(ind, data):
        plt.text(a-0.05, b+0.01, str(round(b,3)))
    plt.show()
    return labels, data

draw_feature_importances(dtr, diab_df)
```



Out[25]:
```
(['bmi', 's6', 's3', 's5', 'bp', 'age', 'sex', 's1', 's2', 's4'],
 [0.6177242303081946,
  0.266745007200364,
  0.10686039054222497,
  0.004700107803172038,
  0.003970264146044441,
  0.0,
  0.0,
  0.0,
  0.0,
  0.0])
```

In [26]:
```
from io import StringIO
from sklearn.tree import export_graphviz
import pydotplus

def get_png_tree(tree_model_param, feature_names_param):
    dot_data = StringIO()
    export_graphviz(tree_model_param, out_file=dot_data, feature_names=feature_names_param,
                    filled=True, rounded=True, special_characters=True)
    graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
    return graph.create_png()

Image(get_png_tree(dtr, diab_df[diab['feature_names']].columns), height='70%')
```

Out[26]: