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 (линейную или полиномиальную регрессию при решении задачи регрессии, логистическую регрессию при решении задачи классификации); SVM; дерево решений. Оцените качество моделей с помощью двух подходящих для задачи метрик. Сравните качество полученных моделей. Постройте график, показывающий важность признаков в дереве решений. Визуализируйте дерево решений или выведите правила дерева решений в текстовом виде.

In [ ]:

```
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 [ ]:

```
# 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[ ]:

False

In [ ]:

```
diab_df.describe()
```

Out[ ]:

	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	-3.634285e-16	1.308343e-16	-8.045349e-16	1.281655e-16	-8.835316e-17	1.327024e-16	-4.574646e-16	3.777301e-16	-3.830854e-16	-3.412882e-16	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	-1.123996e-01	-1.267807e-01	-1.156131e-01	-1.023071e-01	-7.639450e-02	-1.260974e-01	-1.377672e-01	25.00
25%	-3.729927e-02	-4.464164e-02	-3.422907e-02	-3.665645e-02	-3.424784e-02	-3.035840e-02	-3.511716e-02	-3.949338e-02	-3.324879e-02	-3.317903e-02	87.00
50%	5.383060e-03	-4.464164e-02	-7.283766e-03	-5.670611e-03	-4.320866e-03	-3.819065e-03	-6.584468e-03	-2.592262e-03	-1.947634e-03	-1.077698e-03	140.50
75%	3.807591e-02	5.068012e-02	3.124802e-02	3.564384e-02	2.835801e-02	2.984439e-02	2.931150e-02	3.430886e-02	3.243323e-02	2.791705e-02	211.50
max	1.107267e-01	5.068012e-02	1.705552e-01	1.320442e-01	1.539137e-01	1.987880e-01	1.811791e-01	1.852344e-01	1.335990e-01	1.356118e-01	346.00

In [ ]:

```
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 [ ]:

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

Out[ ]:

```
(array([   0.        ,  -29.04480455,  522.18766076,  185.22299625,
          -0.        ,   -0.        , -142.29451879,    0.        ,
         430.33034168,    0.        ]), 151.85913333692827)
```

In [ ]:

```
# 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[ ]:

(0.4224527596544878, 43.95088508641399)

In [ ]:

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

Out[ ]:

```
array([0.66666667, 1.        , 0.58264463, 0.54929577, 0.29411765,
       0.25697211, 0.20779221, 0.28208745, 0.56221737, 0.43939394])
```

In [ ]:

```
# 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[ ]:

```
SVR(kernel='poly')
```

In [ ]:

```
# 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[ ]:

```
(0.4092781221129439, 45.28303482270574)
```

In [ ]:

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

Out[ ]:

```
DecisionTreeRegressor(criterion='poisson', max_depth=5)
```

In [ ]:

```
# 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[ ]:

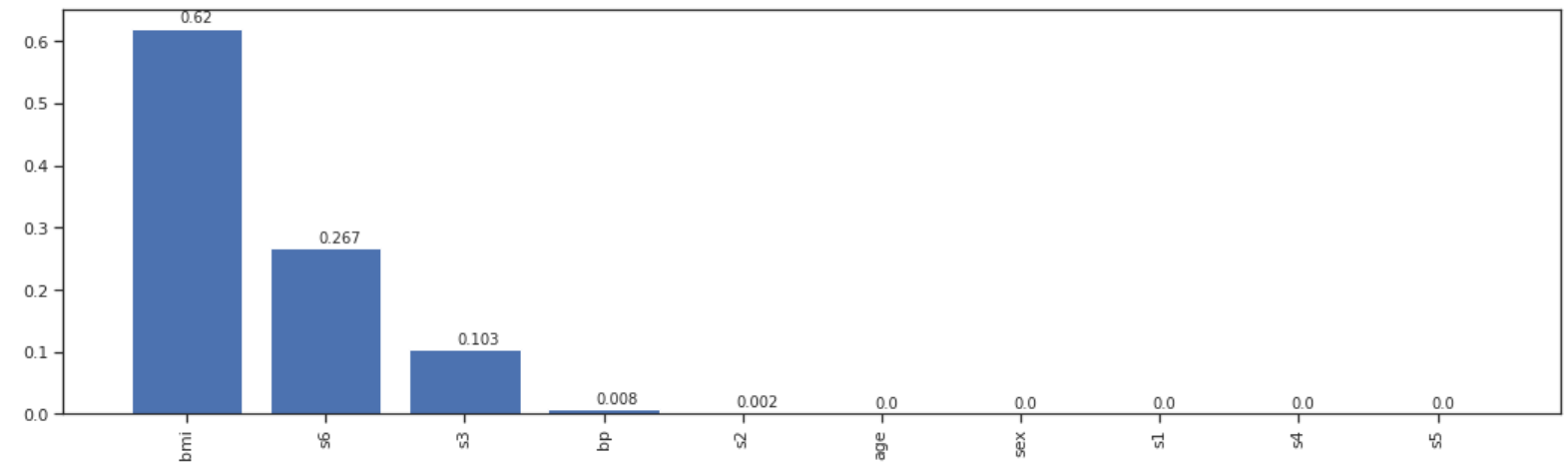
```
(0.08654337289278313, 59.64016853932585)
```

In [ ]:

```
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[ ]:

```
(['bmi', 's6', 's3', 'bp', 's2', 'age', 'sex', 's1', 's4', 's5'],
 [0.6203009857476233,
  0.2667450072003657,
  0.10310974300456148,
  0.007720911683706442,
  0.002123352363743064,
  0.0,
  0.0,
  0.0,
  0.0,
  0.0])
```

In [ ]:

```
from IPython.core.display import HTML
from sklearn.tree import export_text
tree_rules = export_text(dtr, feature_names=list(diab['feature_names']))
HTML('<pre>' + tree_rules + '</pre>')
```

Out[ ]:

```
|--- s3 <= -0.10
|   |--- value: [341.00]
|--- s3 >  -0.10
|   |--- s6 <= 0.13
|   |   |--- bmi <= 0.10
|   |   |   |--- bmi <= 0.10
|   |   |   |   |--- bmi <= 0.10
|   |   |   |   |   |--- value: [146.60]
|   |   |   |   |--- bmi >  0.10
|   |   |   |   |   |--- value: [274.00]
|   |   |   |--- bmi >  0.10
|   |   |   |   |--- value: [275.00]
|   |   |--- bmi >  0.10
|   |   |   |--- s6 <= 0.06
|   |   |   |   |--- bp <= 0.09
|   |   |   |   |   |--- value: [262.62]
|   |   |   |   |--- bp >  0.09
|   |   |   |   |   |--- value: [308.00]
|   |   |   |--- s6 >  0.06
|   |   |   |   |--- value: [336.00]
|   |--- s6 >  0.13
|   |   |--- bp <= -0.01
|   |   |   |--- value: [317.00]
|   |   |--- bp >  -0.01
|   |   |   |--- bmi <= 0.07
|   |   |   |   |--- s2 <= 0.01
|   |   |   |   |   |--- value: [244.00]
|   |   |   |   |--- s2 >  0.01
|   |   |   |   |   |--- value: [281.00]
|   |   |   |--- bmi >  0.07
|   |   |   |   |--- value: [310.00]
```