Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчет по лабораторной работе № 4
«Линейные модели, SVM и деревья решений»

Выполнил:

студент группы ИУ5-65Б

Герасименко А.В.

Подпись и дата:

Проверил:

преподаватель каф. ИУ5

Гапанюк Ю.Е.

Подпись и дата:

г. Москва, 2021 г.

Лабораторная работа №4

Линейные модели, SVM и деревья решений

Цель лабораторной работы

Изучение линейных моделей, SVM и деревьев 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;
   - SVM;
   - дерево решений.
5. Оцените качество моделей с помощью двух подходящих для задачи метрик. Сравните качество полученных моделей.

Дополнительные задания

- Проведите эксперименты с важностью признаков в дереве решений.
- Визуализируйте дерево решений.

Ход выполнения лабораторной работы

In [1]:
```
import pandas as pd
import seaborn as sns
import numpy as np
from typing import Tuple, Dict
import matplotlib.pyplot as plt
from operator import itemgetter
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import f1_score, r2_score, mean_squared_error
from sklearn.metrics import mean_absolute_error, accuracy_score, precisio
from sklearn.svm import LinearSVR, SVR
from sklearn.tree import DecisionTreeRegressor, plot_tree
from sklearn.preprocessing import MinMaxScaler
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [2]:
```
col_list = ['Pelvic_incidence',
            'Pelvic_tilt',
            'Lumbar_lordosis_angle',
```

```
                'Sacral_slope',
                'Pelvic_radius',
                'Degree_spondylolisthesis',
                'Pelvic_slope',
                'Direct_tilt',
                'Thoracic_slope',
                'Cervical_tilt',
                'Sacrum_angle',
                'Scoliosis_slope',
                'Class_att',
                'To_drop']
data = pd.read_csv('Dataset_spine.csv', names=col_list, header=1, sep=",")
```

In [3]:

Out[3]:

	Pelvic_incidence	Pelvic_tilt	Lumbar_lordosis_angle	Sacral_slope	Pelvic_radius	Degree_spon
0	39.056951	10.060991	25.015378	28.995960	114.405425	
1	68.832021	22.218482	50.092194	46.613539	105.985135	
2	69.297008	24.652878	44.311238	44.644130	101.868495	
3	49.712859	9.652075	28.317406	40.060784	108.168725	
4	40.250200	13.921907	25.124950	26.328293	130.327871	

In [4]:

Out[4]:
```
Pelvic_incidence            0
Pelvic_tilt                 0
Lumbar_lordosis_angle       0
Sacral_slope                0
Pelvic_radius               0
Degree_spondylolisthesis    0
Pelvic_slope                0
Direct_tilt                 0
Thoracic_slope              0
Cervical_tilt               0
Sacrum_angle                0
Scoliosis_slope             0
Class_att                   0
dtype: int64
```

Пропуски данных отсутствуют.

In [5]:

Разделим выборку на обучающую и тестовую:

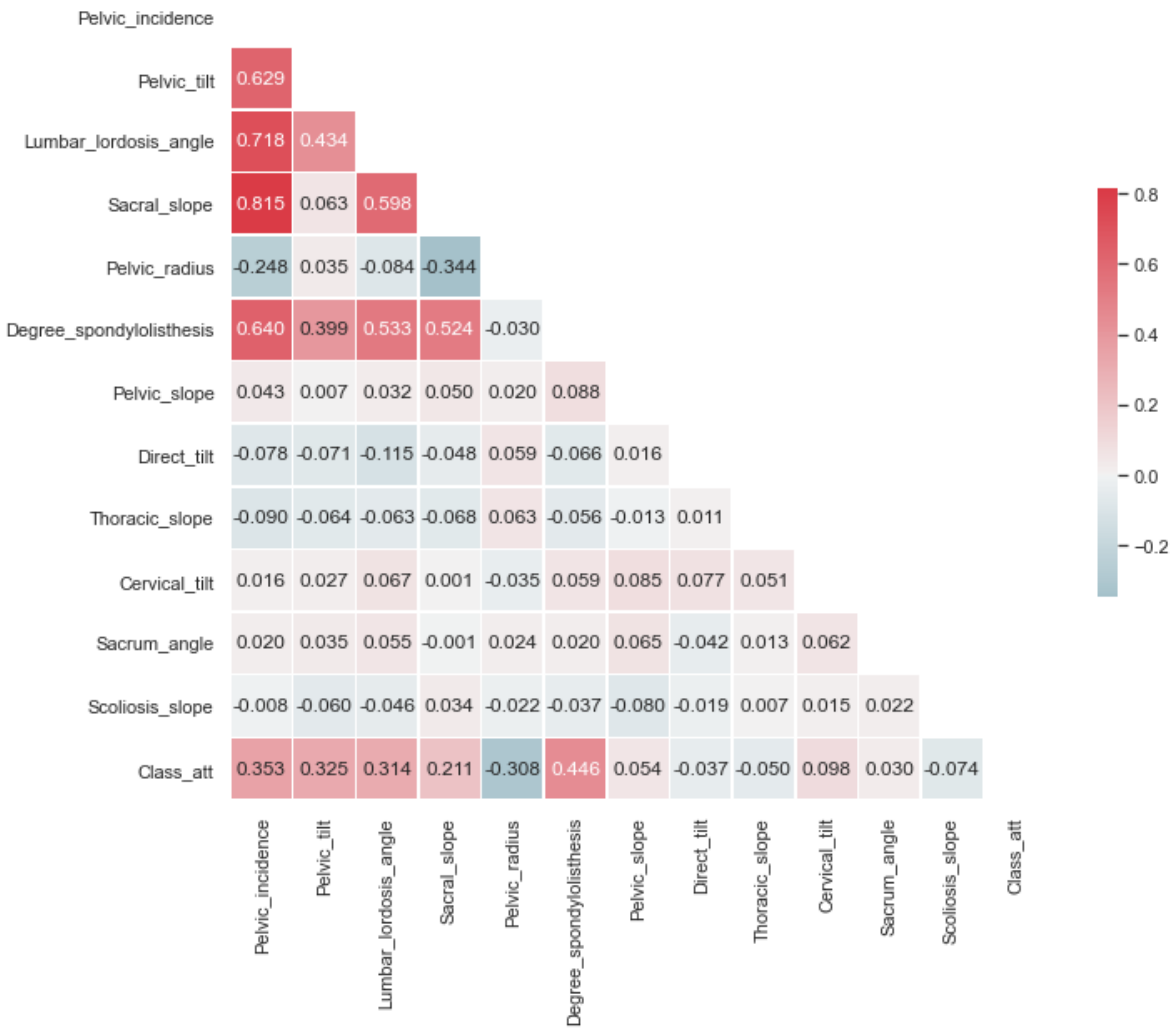
In [6]:
```
# Разделим данные на целевой столбец и признаки
X = data.drop("Class_att", axis=1)
```

In [7]:
```
# С использованием метода train_test_split разделим выборку на обучающую
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25,
```

Out[7]:
```
((231, 12), (78, 12), (231,), (78,))
```

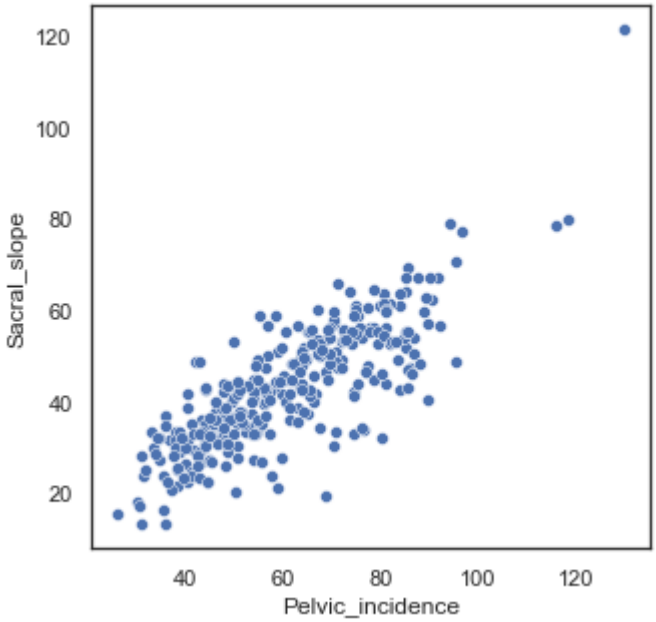
Линейная модель

In [8]:
```
#Построим корреляционную матрицу
sns.set(style="white")
corr = data.corr(method='pearson')
mask = np.zeros_like(corr, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
f, ax = plt.subplots(figsize=(11, 9))
cmap = sns.diverging_palette(220, 10, as_cmap=True)
g=sns.heatmap(corr, mask=mask, cmap=cmap, center=0, annot=True, fmt='.3f'
```



In [9]:
```
fig, ax = plt.subplots(figsize=(5,5))
```

Out[9]: <AxesSubplot:xlabel='Pelvic_incidence', ylabel='Sacral_slope'>



In [10]:
```
x_array = data['Pelvic_incidence'].values
```

In [11]:
```
# Аналитическое вычисление коэффициентов регрессии
def analytic_regr_coef(x_array : np.ndarray,
                       y_array : np.ndarray) -> Tuple[float, float]:
    x_mean = np.mean(x_array)
    y_mean = np.mean(y_array)
    var1 = np.sum([(x-x_mean)**2 for x in x_array])
    cov1 = np.sum([(x-x_mean)*(y-x_mean) for x, y in zip(x_array, y_array
    b1 = cov1 / var1
    b0 = y_mean - b1*x_mean
```

In [12]:
```
b0, b1 = analytic_regr_coef(x_array, y_array)
```

Out[12]: (4.565546113493056, 0.634770752628697)

In [13]:
```
# Вычисление значений y на основе x для регрессии
def y_regr(x_array : np.ndarray, b0: float, b1: float) -> np.ndarray:
    res = [b1*x+b0 for x in x_array]
```

In [14]:
```
```

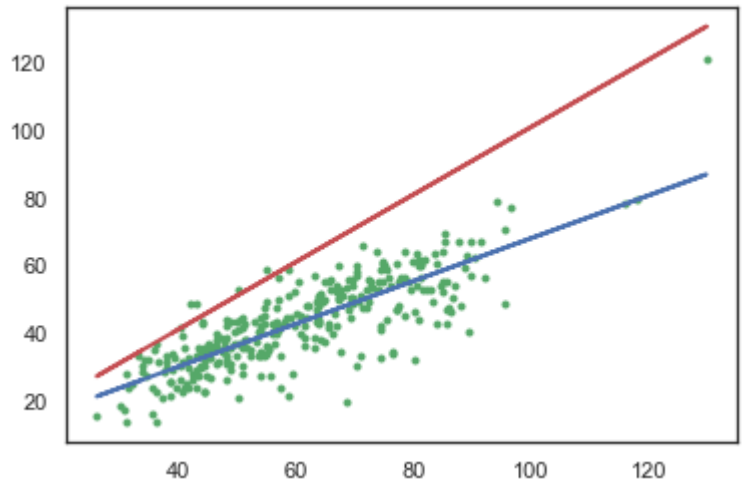
In [15]:
```
# Простейшая реализация градиентного спуска
def gradient_descent(x_array : np.ndarray,
                     y_array : np.ndarray,
                     b0_0 : float,
                     b1_0 : float,
                     epochs : int,
                     learning_rate : float = 0.001
                    ) -> Tuple[float, float]:
    # Значения для коэффициентов по умолчанию
    b0, b1 = b0_0, b1_0
    k = float(len(x_array))
    for i in range(epochs):
```

```
        # Вычисление новых предсказанных значений
        # используется векторизованное умножение и сложение для вектора и
        y_pred = b1 * x_array + b0
        # Расчет градиентов
        # np.multiply - поэлементное умножение векторов
        dL_db1 = (-2/k) * np.sum(np.multiply(x_array, (y_array - y_pred))
        dL_db0 = (-2/k) * np.sum(y_array - y_pred)
        # Изменение значений коэффициентов:
        b1 = b1 - learning_rate * dL_db1
        b0 = b0 - learning_rate * dL_db0
    # Результирующие значения
    y_pred = b1 * x_array + b0
```

In [16]:
```
def show_gradient_descent(epochs, b0_0, b1_0):
    grad_b0, grad_b1, grad_y_pred = gradient_descent(x_array, y_array, b0
    print('b0 = {} - (теоретический), {} - (градиентный спуск)'.format(b0
    print('b1 = {} - (теоретический), {} - (градиентный спуск)'.format(b1
    print('MSE = {}'.format(mean_squared_error(y_array_regr, grad_y_pred)
    plt.plot(x_array, y_array, 'g.')
    plt.plot(x_array, y_array_regr, 'b', linewidth=2.0)
    plt.plot(x_array, grad_y_pred, 'r', linewidth=2.0)
```

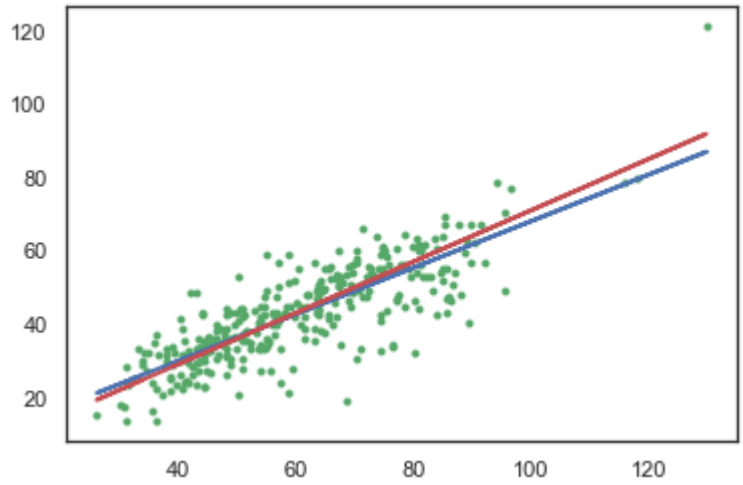
In [17]:
```
# Примеры использования градиентного спуска
```

```
b0 = 4.565546113493056 - (теоретический), 1 - (градиентный спуск)
b1 = 0.634770752628697 - (теоретический), 1 - (градиентный спуск)
MSE = 382.86303871346706
```



In [18]:

```
b0 = 4.565546113493056 - (теоретический), 1 - (градиентный спуск)
b1 = 0.634770752628697 - (теоретический), 0.7 - (градиентный спуск)
MSE = 1.4084724961222774
```



In [19]:

```
# Обучим линейную регрессию и сравним коэффициенты с рассчитанными ранее
reg1 = LinearRegression().fit(x_array.reshape(-1, 1), y_array.reshape(-1,
```

Out[19]:

```
((0.634770752628697, array([[0.63477075]])),
 (4.565546113493056, array([4.56554611])))
```

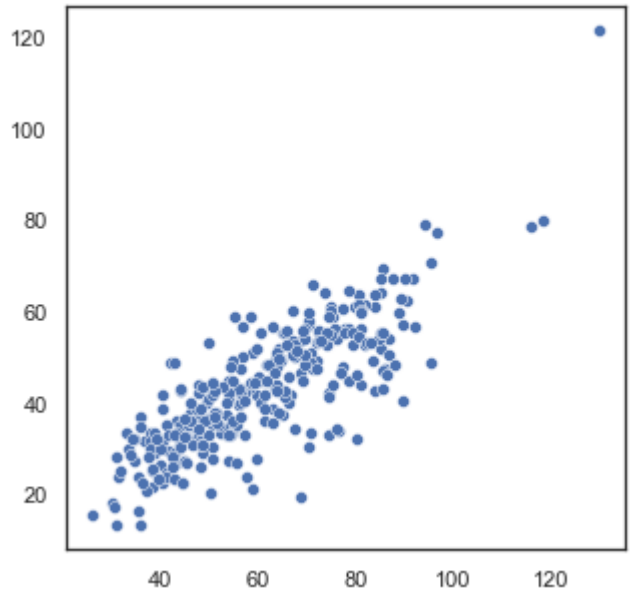
SVM

In [20]:

```
fig, ax = plt.subplots(figsize=(5,5))
```

Out[20]:

```
<AxesSubplot:>
```



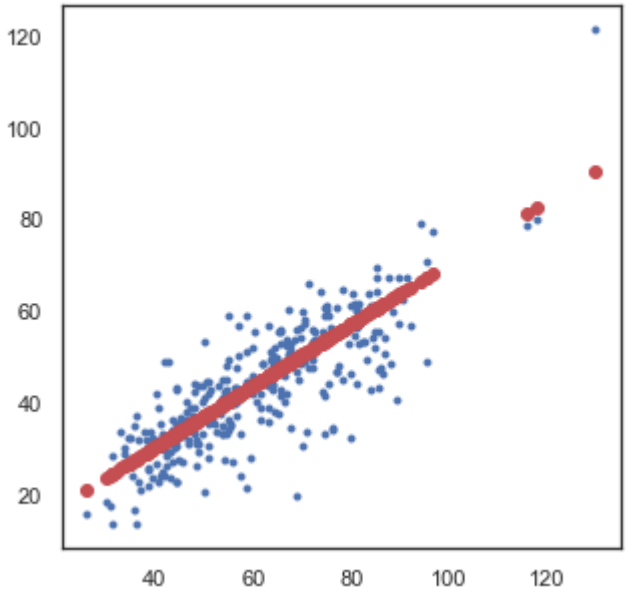
In [21]:

```
def plot_regr(clf):
    title = clf.__repr__
    clf.fit(x_array.reshape(-1, 1), y_array)
    y_pred = clf.predict(x_array.reshape(-1, 1))
    fig, ax = plt.subplots(figsize=(5,5))
```

```
    ax.set_title(title)
    ax.plot(x_array, y_array, 'b.')
    ax.plot(x_array, y_pred, 'ro')
```

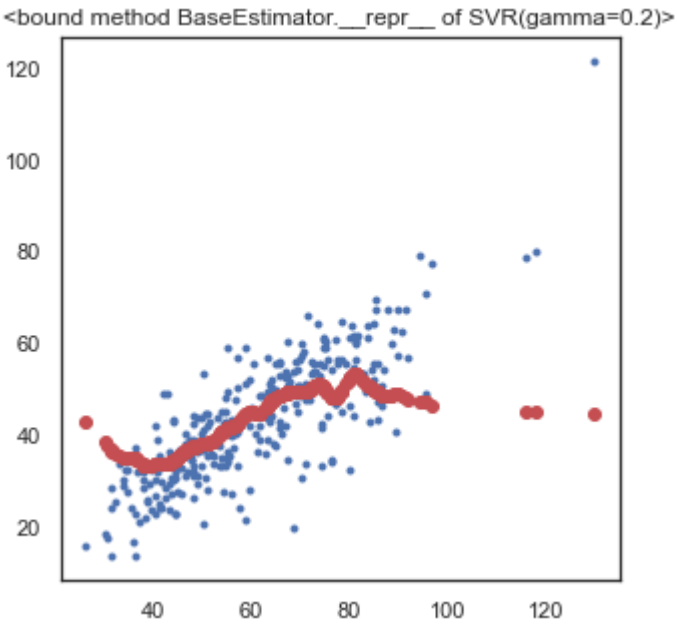
In [22]:

```
<bound method BaseEstimator.__repr__ of LinearSVR(max_iter=100000)>
```



In [23]:

```
<bound method BaseEstimator.__repr__ of SVR(gamma=0.2)>
```



Дерево решений

In [24]:

```
# Обучим дерево на всех признаках
tree = DecisionTreeRegressor(random_state=1)
```

Out[24]:

```
DecisionTreeRegressor(random_state=1)
```

In [25]:

```
# Важность признаков
```

Out[25]:

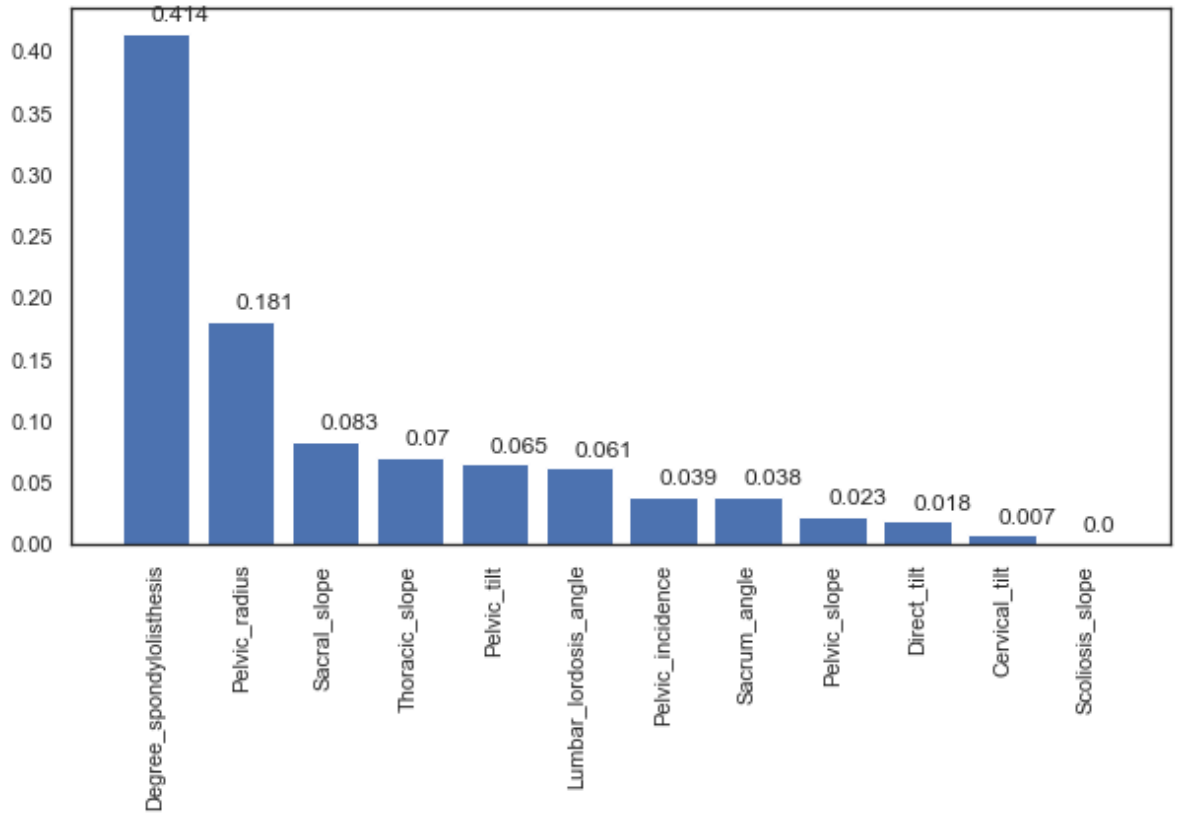
```
[('Pelvic_incidence', 0.03881985535831688),
 ('Pelvic_tilt', 0.0648640652612829),
 ('Lumbar_lordosis_angle', 0.06149200087661627),
 ('Sacral_slope', 0.08263904265020883),
 ('Pelvic_radius', 0.181238439623055),
 ('Degree_spondylolisthesis', 0.41411903317709764),
 ('Pelvic_slope', 0.023065828402366866),
 ('Direct_tilt', 0.01842735042735041),
 ('Thoracic_slope', 0.07000532439464523),
```

In [26]:
```
# Важность признаков в сумме дает единицу
```

Out[26]: 1.0

In [27]:
```
def draw_feature_importances(tree_model, X_dataset, figsize=(10,5)):
    """
    Вывод важности признаков в виде графика
    """
    # Сортировка значений важности признаков по убыванию
    list_to_sort = list(zip(X_dataset.columns.values, tree_model.feature_
    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()
```

In [28]:



In [29]:
```
# Список признаков, отсортированный на основе важности, и значения важнос
```

Out[29]:
```
(['Degree_spondylolisthesis',
  'Pelvic_radius',
  'Sacral_slope',
  'Thoracic_slope',
  'Pelvic_tilt',
  'Lumbar_lordosis_angle',
  'Pelvic_incidence',
  'Sacrum_angle',
  'Pelvic_slope',
  'Direct_tilt',
  'Cervical_tilt',
  'Scoliosis_slope'],
 [0.41411903317709764,
  0.181238439623055,
  0.08263904265020883,
  0.07000532439464523,
  0.0648640652612829,
  0.06149200087661627,
  0.03881985535831688,
  0.038183760683760684,
  0.023065828402366866,
  0.01842735042735041,
  0.007145299145299295,
  0.0])
```

In [30]:

Out[30]:

	Pelvic_incidence	Pelvic_tilt	Lumbar_lordosis_angle	Sacral_slope	Pelvic_radius	Degree_sp

	Pelvic_incidence	Pelvic_tilt	Lumbar_lordosis_angle	Sacral_slope	Pelvic_radius	Degree_sp
291	34.756738	2.631740	29.504381	32.124998	127.139849	
17	38.697912	13.444749	31.000000	25.253163	123.159251	
110	84.998956	29.610098	83.352194	55.388858	126.912990	
228	43.426451	10.095743	36.032224	33.340707	137.439694	

In [31]:
```
# Пересортируем признаки на основе важности
X_train_sorted = X_train[tree_fl]
```

Out[31]:

	Degree_spondylolisthesis	Pelvic_radius	Sacral_slope	Thoracic_slope	Pelvic_tilt	Lumbar_l
291	-0.460894	127.139849	32.124998	11.2762	2.631740	
17	1.429186	123.159251	25.253163	17.9575	13.444749	
110	71.321175	126.912990	55.388858	9.0119	29.610098	
228	-3.114451	137.439694	33.340707	11.0132	10.095743	
125	27.810148	103.008354	48.972496	14.8568	21.704402	

In [32]:
```
```

In [33]:
```
```

Out[33]: 0.11538461538461539

In [34]:
```
# Обучим дерево и предскажем результаты на пяти лучших признаках
tree_2 = DecisionTreeRegressor(random_state=1).fit(X_train[tree_fl[0:5]],
```

In [35]:
```
```

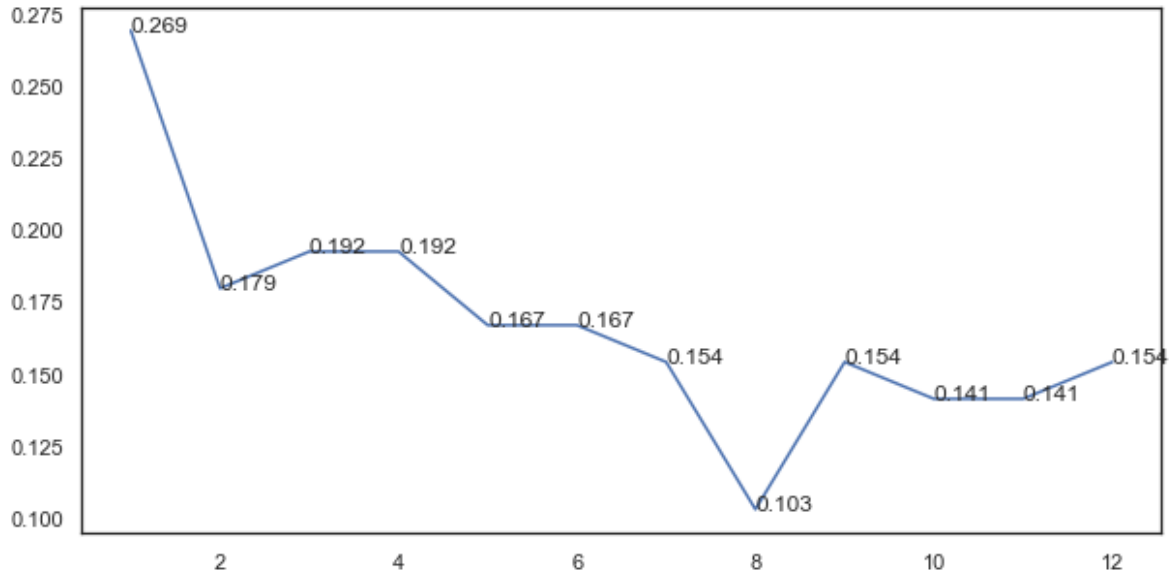
Out[35]: 0.16666666666666666

In [36]:
```
# Исследуем, как изменяется ошибка при добавлении признаков в порядке зна
X_range = list(range(1, len(X_train.columns)+1))
```

Out[36]: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

In [37]:
```
mae_list = []
for i in X_range:
    # Обучим дерево и предскажем результаты на заданном количестве призна
    tree_3 = DecisionTreeRegressor(random_state=1).fit(X_train[tree_fl[0:
    Y_test_predict_3 = tree_3.predict(X_test[tree_fl[0:i]])
    temp_mae = mean_absolute_error(Y_test, Y_test_predict_3)
```

In [38]:
```
plt.subplots(figsize=(10,5))
plt.plot(X_range, mae_list)
for a,b in zip(X_range, mae_list):
    plt.text(a, b, str(round(b,3)))
```



Оценка качества моделей

Дерево решений

In [39]:
```
print("r2_score:", r2_score(Y_test, tree.predict(X_test)))
```

```
r2_score: 0.4701886792452832
mean_squared_error: 0.11538461538461539
```

Линейная регрессия

In [40]:
```
pred = reg1.predict(x_array.reshape(-1, 1))
print("r2_score:", r2_score(y_array, pred))
```

```
r2_score: 0.664423352506976
mean_squared_error 60.45739674813066
```

Метод опорных векторов

In [41]:
```
svr = SVR(kernel='rbf')
svr.fit(X_train, Y_train)
print("r2_score:", r2_score(Y_test, svr.predict(X_test)))
```

```
r2_score: 0.5993410854165502
mean_squared_error 0.0872572422457382
```

Последние две модели являются приемлемыми, т.к. коэффициент детерминации для всех трех моделей больше 50%.

Если учитывать показатели обеих метрик, наилучший результат показал метод опорных векторов.