**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Радиотехнический»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №4

«Линейные модели, SVM и деревья решений.»

Выполнил: Проверил:

студент группы РТ5-61Б преподаватель каф. ИУ5

Агеев Алексей Гапанюк Ю.Е.

Подпись и дата: Подпись и дата:

Москва, 2023 г

# 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.

# Ход работы

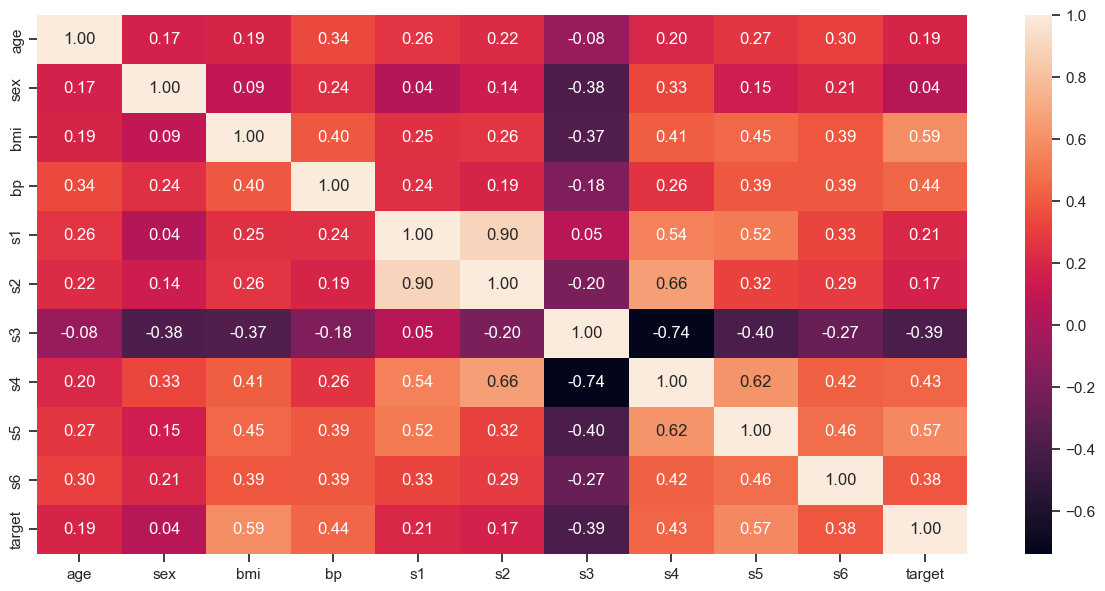
from IPython.display import Image  
import numpy as np  
import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
from sklearn.datasets import load\_diabetes  
from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, mean\_squared\_log\_error, median\_absolute\_error, r2\_score  
from sklearn.linear\_model import LinearRegression  
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor, export\_graphviz  
import matplotlib.pyplot as plt  
from IPython.display import Image  
from sklearn.svm import SVC  
import pydotplus  
import graphviz   
from io import StringIO   
import seaborn as sns  
%matplotlib inline   
sns.set(style="ticks")

diabetes = load\_diabetes()  
df\_diabetes = pd.DataFrame(diabetes.data,columns=diabetes.feature\_names)  
df\_diabetes['target'] = pd.Series(diabetes.target)  
df\_diabetes.head()

age sex bmi bp s1 s2 s3   
0 0.038076 0.050680 0.061696 0.021872 -0.044223 -0.034821 -0.043401 \  
1 -0.001882 -0.044642 -0.051474 -0.026328 -0.008449 -0.019163 0.074412   
2 0.085299 0.050680 0.044451 -0.005670 -0.045599 -0.034194 -0.032356   
3 -0.089063 -0.044642 -0.011595 -0.036656 0.012191 0.024991 -0.036038   
4 0.005383 -0.044642 -0.036385 0.021872 0.003935 0.015596 0.008142   
  
 s4 s5 s6 target   
0 -0.002592 0.019907 -0.017646 151.0   
1 -0.039493 -0.068332 -0.092204 75.0   
2 -0.002592 0.002861 -0.025930 141.0   
3 0.034309 0.022688 -0.009362 206.0   
4 -0.002592 -0.031988 -0.046641 135.0

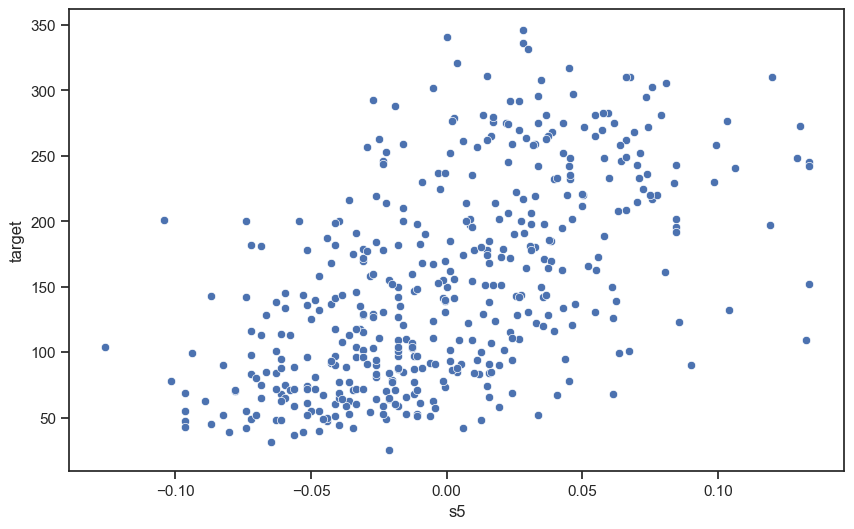
#Построим корреляционную матрицу  
fig, ax = plt.subplots(figsize=(15,7))  
sns.heatmap(df\_diabetes.corr(method='pearson'), ax=ax, annot=True, fmt='.2f')

<Axes: >



fig, ax = plt.subplots(figsize=(10,6))   
sns.scatterplot(ax=ax, x='s5', y='target', data=df\_diabetes)

<Axes: xlabel='s5', ylabel='target'>



## Линейная модель

X\_train, X\_test, y\_train, y\_test = train\_test\_split(  
 diabetes.data, diabetes.target, test\_size=0.8, random\_state=1)

# Обучим линейную регрессию и сравним коэффициенты с рассчитанными ранее  
reg1 = LinearRegression().fit(X\_train, y\_train)  
target1 = reg1.predict(X\_test)

mean\_absolute\_error(y\_test, target1)

48.12036867045504

r2\_score(y\_test, target1)

0.36123801479756923

## SVM

#SVM  
svm = SVC(kernel='linear', C=1E10)  
svm.fit(X\_train, y\_train)  
target3 = svm.predict(X\_test)

mean\_absolute\_error(y\_test, target3)

65.5

r2\_score(y\_test, target1)

0.36123801479756923

## Дерево решений

tree = DecisionTreeRegressor(random\_state=1).fit(X\_train, y\_train)  
target2 = tree.predict(X\_test)

mean\_absolute\_error(y\_test, target2)

63.5819209039548

r2\_score(y\_test, target2)

-0.1389533158261982

# Важность признаков  
list(zip(df\_diabetes.columns.values, tree.feature\_importances\_))

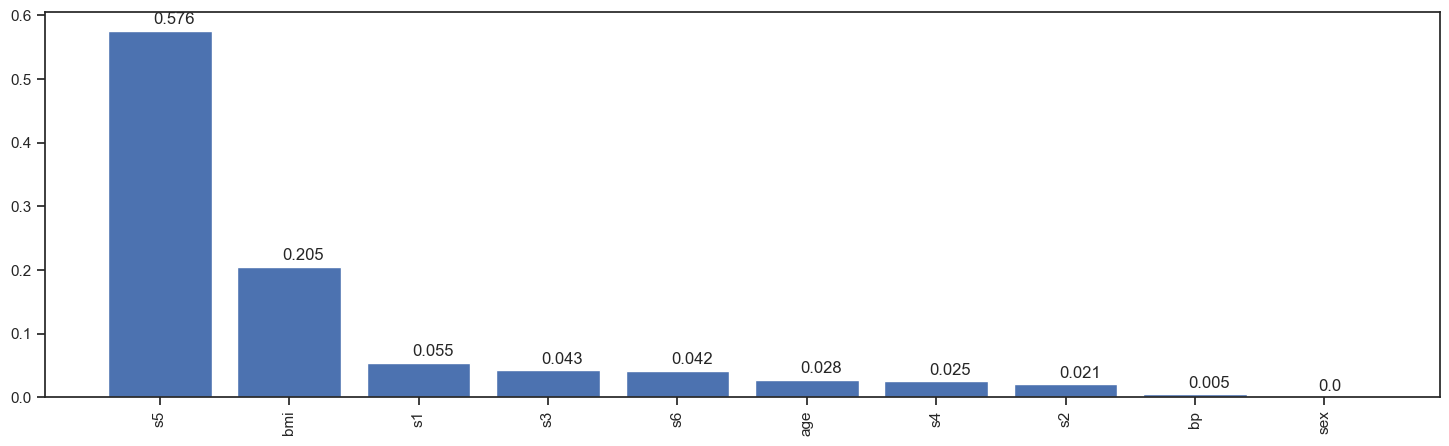
[('age', 0.027573806970594715),  
 ('sex', 0.0004258510680318088),  
 ('bmi', 0.20523697536156485),  
 ('bp', 0.004954211390648432),  
 ('s1', 0.05459571598761492),  
 ('s2', 0.021091077379320294),  
 ('s3', 0.04270292643445246),  
 ('s4', 0.025465854604880798),  
 ('s5', 0.5761425902388766),  
 ('s6', 0.04181099056401502)]

# Важность признаков в сумме дает единицу  
sum(tree.feature\_importances\_)

1.0

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(tree, df\_diabetes)



(['s5', 'bmi', 's1', 's3', 's6', 'age', 's4', 's2', 'bp', 'sex'],  
 [0.5761425902388766,  
 0.20523697536156485,  
 0.05459571598761492,  
 0.04270292643445246,  
 0.04181099056401502,  
 0.027573806970594715,  
 0.025465854604880798,  
 0.021091077379320294,  
 0.004954211390648432,  
 0.0004258510680318088])

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()

from IPython.core.display import HTML  
from sklearn.tree import export\_text  
tree\_rules = export\_text(tree, feature\_names=list(diabetes.feature\_names))  
HTML('<pre>' + tree\_rules + '</pre>')

<IPython.core.display.HTML object>