**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика с системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №4

# «Линейные модели, SVM и деревья решений[.](https://github.com/ugapanyuk/courses_current/wiki/LAB_TMO__MISSING)»

Выполнил: Проверил:

студент группы РТ5-61Б преподаватель каф. ИУ5

Мицкевич В.Б. Гапанюк Ю.Е.

Подпись и дата: Подпись и дата:

Москва, 2023 г.

# **Цель лабораторной работы:**

# изучение линейных моделей, SVM и деревьев решений.

## Задание:[¶](http://localhost:8888/notebooks/LAB4/Untitled.ipynb" \l "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.

# Лабораторная работа

# Линейные модели, SVM и деревья решений.

# Цель лабораторной работы: изучение линейных моделей, SVM и деревьев решений.

## 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.

import pandas as pd  
import numpy as np  
import seaborn as sns  
import matplotlib.pyplot as plt  
from sklearn.datasets import \*  
from sklearn.preprocessing import MinMaxScaler  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LogisticRegression  
from sklearn import svm, tree  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.metrics import accuracy\_score  
from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay  
import operator

def make\_dataframe(ds\_function):  
 ds = ds\_function()  
 df = pd.DataFrame(data= np.c\_[ds['data'], ds['target']],  
 columns= list(ds['feature\_names']) + ['target'])  
 return df  
  
wine = load\_wine()  
  
df = make\_dataframe(load\_wine)

df.head()

alcohol malic\_acid ash alcalinity\_of\_ash magnesium total\_phenols \  
0 14.23 1.71 2.43 15.6 127.0 2.80   
1 13.20 1.78 2.14 11.2 100.0 2.65   
2 13.16 2.36 2.67 18.6 101.0 2.80   
3 14.37 1.95 2.50 16.8 113.0 3.85   
4 13.24 2.59 2.87 21.0 118.0 2.80   
  
 flavanoids nonflavanoid\_phenols proanthocyanins color\_intensity hue \  
0 3.06 0.28 2.29 5.64 1.04   
1 2.76 0.26 1.28 4.38 1.05   
2 3.24 0.30 2.81 5.68 1.03   
3 3.49 0.24 2.18 7.80 0.86   
4 2.69 0.39 1.82 4.32 1.04   
  
 od280/od315\_of\_diluted\_wines proline target   
0 3.92 1065.0 0.0   
1 3.40 1050.0 0.0   
2 3.17 1185.0 0.0   
3 3.45 1480.0 0.0   
4 2.93 735.0 0.0

df.describe()

alcohol malic\_acid ash alcalinity\_of\_ash magnesium \  
count 178.000000 178.000000 178.000000 178.000000 178.000000   
mean 13.000618 2.336348 2.366517 19.494944 99.741573   
std 0.811827 1.117146 0.274344 3.339564 14.282484   
min 11.030000 0.740000 1.360000 10.600000 70.000000   
25% 12.362500 1.602500 2.210000 17.200000 88.000000   
50% 13.050000 1.865000 2.360000 19.500000 98.000000   
75% 13.677500 3.082500 2.557500 21.500000 107.000000   
max 14.830000 5.800000 3.230000 30.000000 162.000000   
  
 total\_phenols flavanoids nonflavanoid\_phenols proanthocyanins \  
count 178.000000 178.000000 178.000000 178.000000   
mean 2.295112 2.029270 0.361854 1.590899   
std 0.625851 0.998859 0.124453 0.572359   
min 0.980000 0.340000 0.130000 0.410000   
25% 1.742500 1.205000 0.270000 1.250000   
50% 2.355000 2.135000 0.340000 1.555000   
75% 2.800000 2.875000 0.437500 1.950000   
max 3.880000 5.080000 0.660000 3.580000   
  
 color\_intensity hue od280/od315\_of\_diluted\_wines proline \  
count 178.000000 178.000000 178.000000 178.000000   
mean 5.058090 0.957449 2.611685 746.893258   
std 2.318286 0.228572 0.709990 314.907474   
min 1.280000 0.480000 1.270000 278.000000   
25% 3.220000 0.782500 1.937500 500.500000   
50% 4.690000 0.965000 2.780000 673.500000   
75% 6.200000 1.120000 3.170000 985.000000   
max 13.000000 1.710000 4.000000 1680.000000   
  
 target   
count 178.000000   
mean 0.938202   
std 0.775035   
min 0.000000   
25% 0.000000   
50% 1.000000   
75% 2.000000   
max 2.000000

df.target.unique()

array([0., 1., 2.])

df.dtypes

alcohol float64  
malic\_acid float64  
ash float64  
alcalinity\_of\_ash float64  
magnesium float64  
total\_phenols float64  
flavanoids float64  
nonflavanoid\_phenols float64  
proanthocyanins float64  
color\_intensity float64  
hue float64  
od280/od315\_of\_diluted\_wines float64  
proline float64  
target float64  
dtype: object

def count\_nan(data):  
 for col in data.columns:  
 count\_nan = data[data[col].isnull()].shape[0]  
 print('{} имеет NAN: {}'.format(col, count\_nan))  
count\_nan(df)

alcohol имеет NAN: 0  
malic\_acid имеет NAN: 0  
ash имеет NAN: 0  
alcalinity\_of\_ash имеет NAN: 0  
magnesium имеет NAN: 0  
total\_phenols имеет NAN: 0  
flavanoids имеет NAN: 0  
nonflavanoid\_phenols имеет NAN: 0  
proanthocyanins имеет NAN: 0  
color\_intensity имеет NAN: 0  
hue имеет NAN: 0  
od280/od315\_of\_diluted\_wines имеет NAN: 0  
proline имеет NAN: 0  
target имеет NAN: 0

# В данном датасете много значений которые варируютсяв разных чилосвых диапазонах, а тк мы используем линейные можели нужно выполнить нормализацию

x\_data = df.drop('target', axis = 1)  
scaler = MinMaxScaler()  
x\_scaled\_data = scaler.fit\_transform(x\_data)

# Разделение на тестовую и обучающую выборки

y\_target = df['target']  
X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(x\_scaled\_data, y\_target, test\_size = 0.3, random\_state = 1)

## Логическая регрессия <https://academy.yandex.ru/handbook/ml/article/linear-models>

lr = LogisticRegression(random\_state=1)  
lr\_model = lr.fit(X\_train, Y\_train)  
lr\_predict = lr\_model.predict(X\_test)

## SVM <https://www.helenkapatsa.ru/mietod-opornykh-viektorov/>

svc = svm.SVC(random\_state=1)  
svc\_model = svc.fit(X\_train, Y\_train)  
svc\_predict = svc\_model.predict(X\_test)

## Дерево решений <https://academy.yandex.ru/handbook/ml/article/reshayushchiye-derevya>

dt = DecisionTreeClassifier(random\_state=1)  
dt\_model = dt.fit(X\_train, Y\_train)  
dt\_predict = dt\_model.predict(X\_test)

# Оценим качество трех моделей

## 1) accuracy\_score

print("Logistic regression: ", accuracy\_score(Y\_test, lr\_predict))  
print("SVM: ", accuracy\_score(Y\_test, svc\_predict))  
print("Decision tree: ", accuracy\_score(Y\_test, dt\_predict))

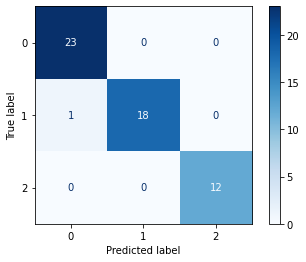
Logistic regression: 0.9814814814814815  
SVM: 0.9814814814814815  
Decision tree: 0.9444444444444444

## 2) Матрица ошибок или Confusion Matrix

### Логистическая регрессия

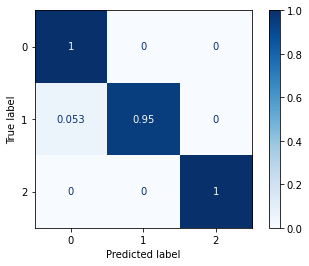
cm = confusion\_matrix(Y\_test, lr\_predict)  
ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=['0', '1', '2']).plot(cmap="Blues")

<sklearn.metrics.\_plot.confusion\_matrix.ConfusionMatrixDisplay at 0x7fe6c3cbd3d0>



cm = confusion\_matrix(Y\_test, lr\_predict, normalize='true')  
ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=['0', '1', '2']).plot(cmap="Blues")

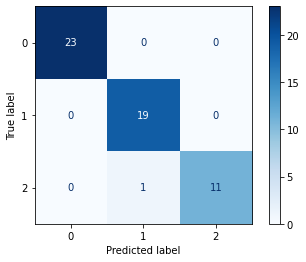
<sklearn.metrics.\_plot.confusion\_matrix.ConfusionMatrixDisplay at 0x7fe6c3fbbb50>



### SVM

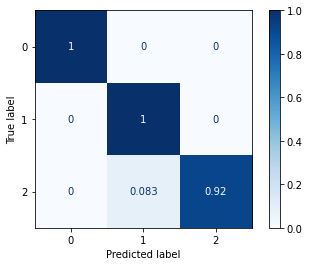
cm = confusion\_matrix(Y\_test, svc\_predict)  
ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=['0', '1', '2']).plot(cmap="Blues")

<sklearn.metrics.\_plot.confusion\_matrix.ConfusionMatrixDisplay at 0x7fe6c1426f10>



cm = confusion\_matrix(Y\_test, svc\_predict, normalize='true')  
ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=['0', '1', '2']).plot(cmap="Blues")

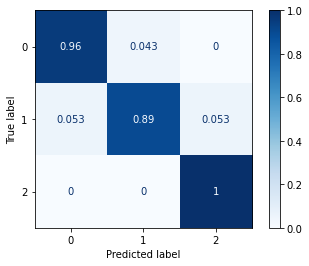
<sklearn.metrics.\_plot.confusion\_matrix.ConfusionMatrixDisplay at 0x7fe6c13acc40>



### Дерево решений

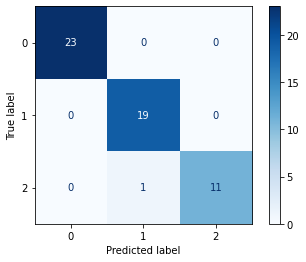
cm = confusion\_matrix(Y\_test, dt\_predict, normalize='true')  
ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=['0', '1', '2']).plot(cmap="Blues")

<sklearn.metrics.\_plot.confusion\_matrix.ConfusionMatrixDisplay at 0x7fe6c13e8160>



cm = confusion\_matrix(Y\_test, svc\_predict)  
ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=['0', '1', '2']).plot(cmap="Blues")

<sklearn.metrics.\_plot.confusion\_matrix.ConfusionMatrixDisplay at 0x7fe6c11c1700>



# Постройте график, показывающий важность признаков в дереве решений. Визуализируйте дерево решений или выведите правила дерева решений в текстовом виде.

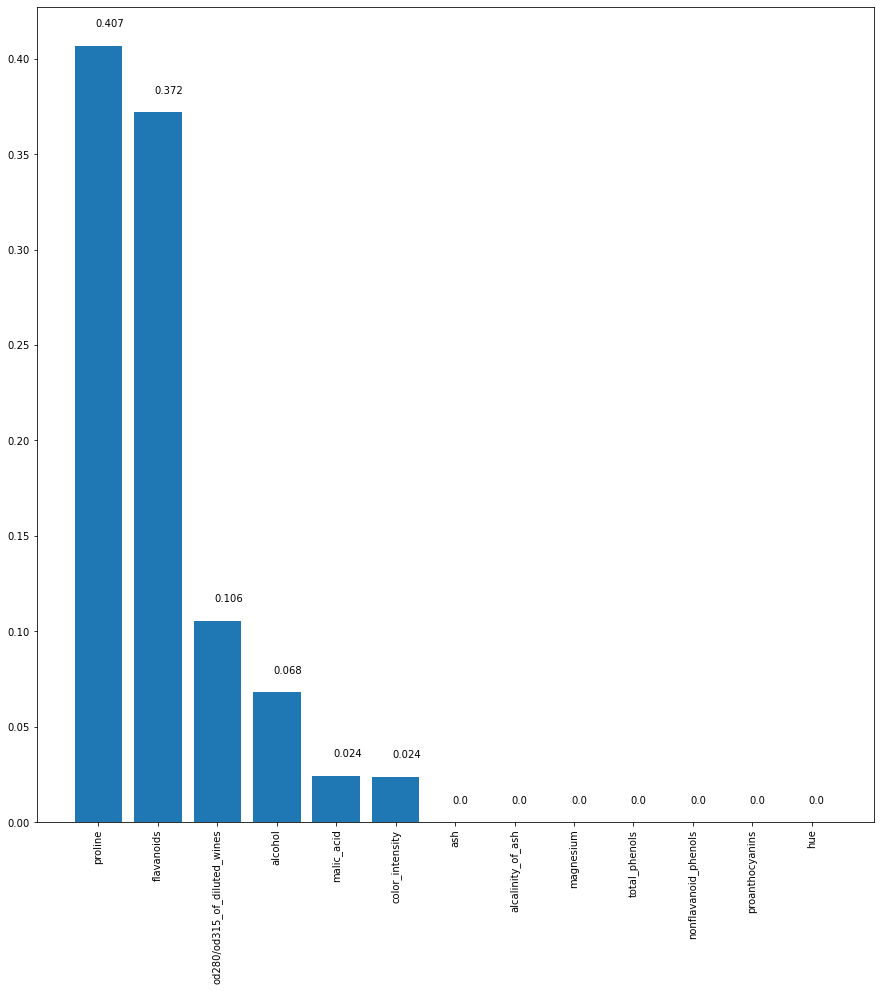
## Важность признаков в дереве

list(zip(x\_data.columns.values, dt.feature\_importances\_))

[('alcohol', 0.0679824561403509),  
 ('malic\_acid', 0.024111111111111114),  
 ('ash', 0.0),  
 ('alcalinity\_of\_ash', 0.0),  
 ('magnesium', 0.0),  
 ('total\_phenols', 0.0),  
 ('flavanoids', 0.37183383991894625),  
 ('nonflavanoid\_phenols', 0.0),  
 ('proanthocyanins', 0.0),  
 ('color\_intensity', 0.023647992530345427),  
 ('hue', 0.0),  
 ('od280/od315\_of\_diluted\_wines', 0.1055713952247566),  
 ('proline', 0.40685320507448963)]

import pandas as pd  
  
def visual\_feature\_importances(values, model, heigth, width):  
 feature\_importances = list(zip(x\_data.columns.values, dt.feature\_importances\_))  
 sort\_feature\_importances = sorted(feature\_importances, key=operator.itemgetter(1), reverse = True)  
 labels = [name for name,value in sort\_feature\_importances]  
 values = [value for name,value in sort\_feature\_importances]  
   
 fig, ax = plt.subplots(figsize=(heigth, width))  
 ind = np.arange(len(labels))  
 plt.bar(ind, values)  
 plt.xticks(ind, labels, rotation='vertical')  
 for a,b in zip(ind, values):  
 plt.text(a-0.05, b+0.01, str(round(b,3)))  
 plt.show()  
 return labels, values

dt\_labels, dt\_values = visual\_feature\_importances(x\_data.columns, dt.feature\_importances\_, 15, 15)



## Визуализация дерева решений

fig = plt.figure(figsize=(25,20))  
\_ = tree.plot\_tree(dt\_model,   
 feature\_names=wine.feature\_names,   
 class\_names=wine.target\_names,  
 filled=True)

