

**Московский государственный технический  
университет им. Н. Э. Баумана**

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №4

Выполнил:  
Сайфутдинов Р.И.  
группа ИУ5-62Б

Проверил:  
Гапанюк Ю.Е.

Дата: 09.04.25

Дата:

Подпись:

Подпись:

Москва, 2025 г.

**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:
  - a. одну из линейных моделей (линейную или полиномиальную регрессию при решении задачи регрессии, логистическую регрессию при решении задачи классификации);
  - b. SVM;
  - c. 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 выполнения:**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
```

## Загрузка и предобработка данных

```
from sklearn.datasets import load_breast_cancer
```

```
data = load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target)
```

## 1. проверим наличие пропусков

```
X.isnull().sum()
```

```
mean radius      0
mean texture     0
mean perimeter   0
mean area        0
mean smoothness  0
mean compactness 0
mean concavity   0
mean concave points 0
mean symmetry    0
mean fractal dimension 0
radius error     0
texture error    0
perimeter error  0
area error       0
smoothness error 0
compactness error 0
concavity error  0
concave points error 0
symmetry error   0
fractal dimension error 0
worst radius     0
worst texture    0
worst perimeter  0
worst area       0
worst smoothness 0
...
worst concavity  0
worst concave points 0
worst symmetry   0
worst fractal dimension 0
dtype: int64
```

Output is truncated. View as a [scrollable element](#) or open in a [text editor](#). Adjust cell output [settings...](#)

## 2. масштабирование признаков

[+ Code](#)

```
scaler = StandardScaler()  
X_scaled = scaler.fit_transform(X)
```

## 3. Разделение на обучающую и тестовую выборки

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)
```

# Обучение моделей

- логистическая регрессия
- SVM
- дерево решений

## 1. логистическая регрессия

```
log_reg = LogisticRegression(max_iter=1000)  
log_reg.fit(X_train, y_train)  
y_pred_log = log_reg.predict(X_test)
```

## 2. SVM

```
svm = SVC(kernel='linear')  
svm.fit(X_train, y_train)  
y_pred_svm = svm.predict(X_test)
```

## 3. дерево решений

```
tree = DecisionTreeClassifier(max_depth=3, random_state=42)  
tree.fit(X_train, y_train)  
y_pred_tree = tree.predict(X_test)
```

# Оценка качества моделей

accuracy и F1-score

```
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix
```



```
print("Logistic Regression:")
print("Accuracy:", accuracy_score(y_test, y_pred_log))
print("F1-score:", f1_score(y_test, y_pred_log))

print("\nSVM:")
print("Accuracy:", accuracy_score(y_test, y_pred_svm))
print("F1-score:", f1_score(y_test, y_pred_svm))

print("\nDecision Tree:")
print("Accuracy:", accuracy_score(y_test, y_pred_tree))

print("F1-score:", f1_score(y_test, y_pred_tree))
```

```
... Logistic Regression:
Accuracy: 0.9824561403508771
F1-score: 0.986046511627907

SVM:
Accuracy: 0.9766081871345029
F1-score: 0.9814814814814815

Decision Tree:
Accuracy: 0.9649122807017544
F1-score: 0.9724770642201835
```

## Важность признаков в дереве решений

[16]

```
plt.figure(figsize=(12, 6))
plt.barh(data.feature_names, tree.feature_importances_)
plt.xlabel("Feature Importance")
plt.ylabel("Feature")
plt.title("Decision Tree Feature Importance")
plt.show()
```

...

