Московский государственный технический

университет им. Н.Э. Баумана

Факультет Радиотехнический

Кафедра РТ5

Курс «Технологии машинного обучения»

Отчет по лабораторной работе №4

«Линейные модели, SVM и деревья решений»

Выполнил:

студент группы РТ5-61Б:

Бабасанова Н. С.

Руководитель:

преподаватель каф. ИУ5

Гапанюк Ю.Е.

Москва, 2025г.

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 (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.

Текст программы

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

from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import MinMaxScaler
from sklearn.svm import LinearSVC

from sklearn.datasets import load_breast_cancer
data = load_breast_cancer()
data_df = pd.DataFrame(data = data.data,
                       columns = data.feature_names)
data_df
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	...	25.380
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	24.990
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	23.570
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	14.910
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	22.540
...	...	...	...	...	...	...	...	...	...	...	...	...
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	25.450
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	23.690
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	18.980
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	25.740
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...	9.456

569 rows × 30 columns

```
data_df["target"] = data.target
data_df
data_df.isnull().sum()
```

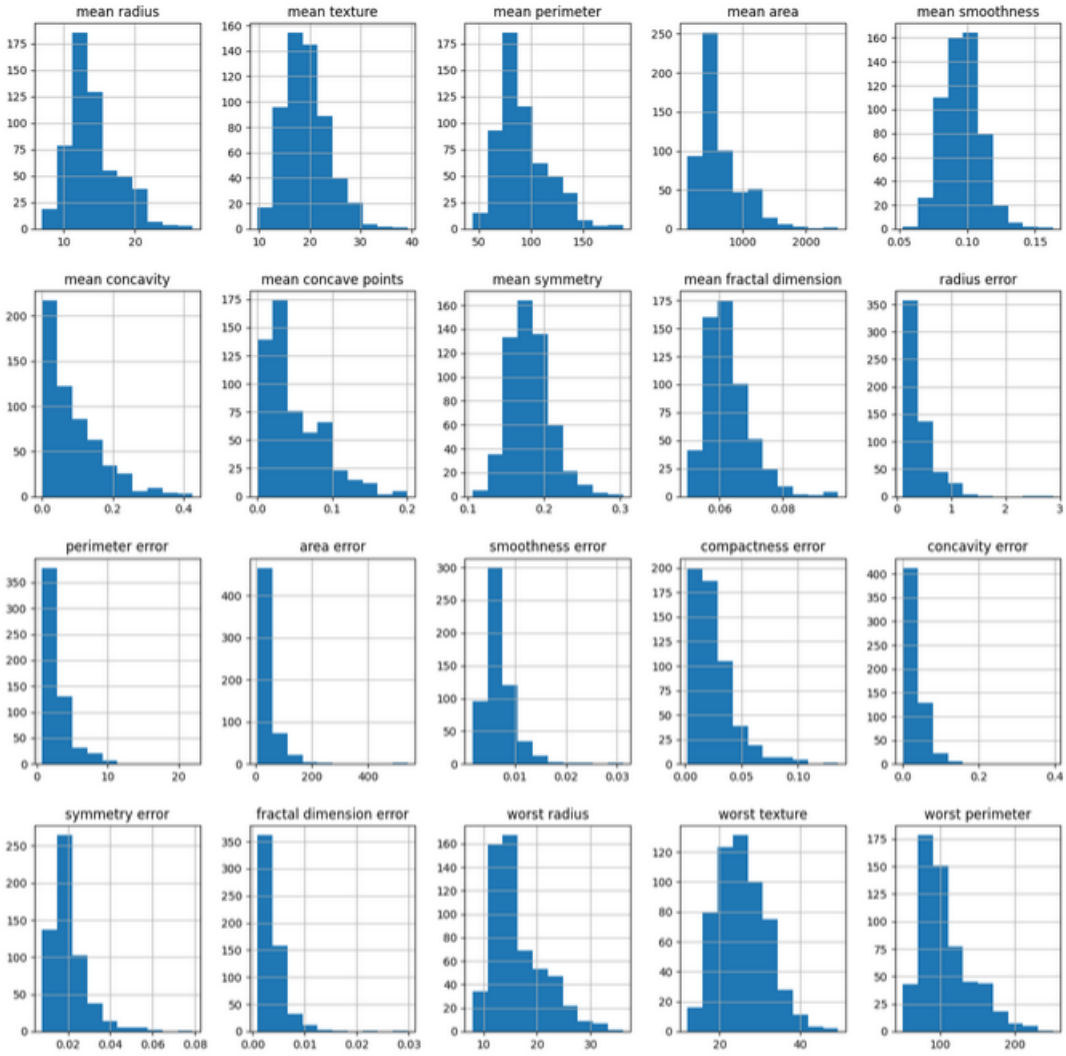
```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius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

```
data_df.describe()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	56
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	

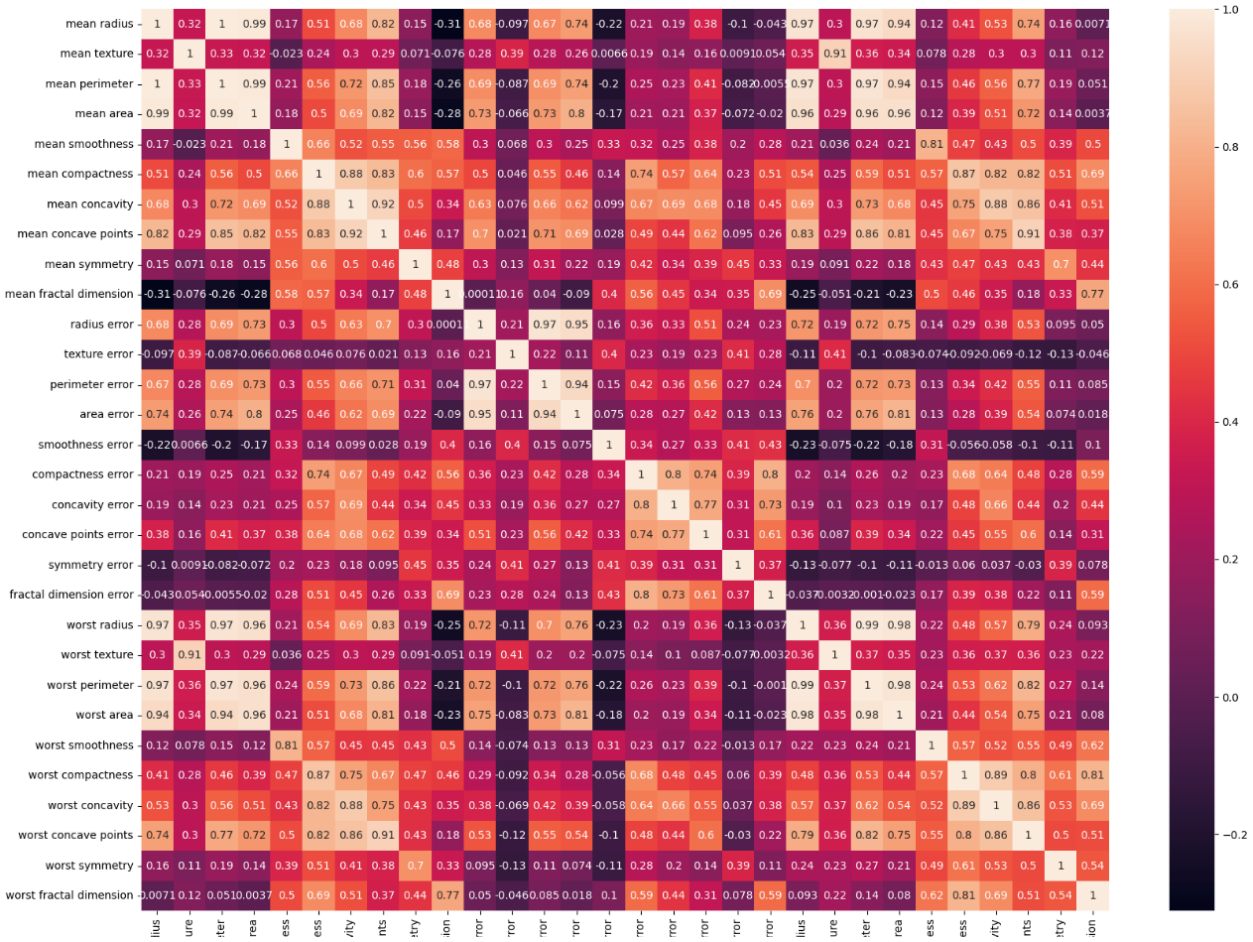
8 rows × 31 columns

```
data_df.hist(figsize=(20,25))
plt.show()
```

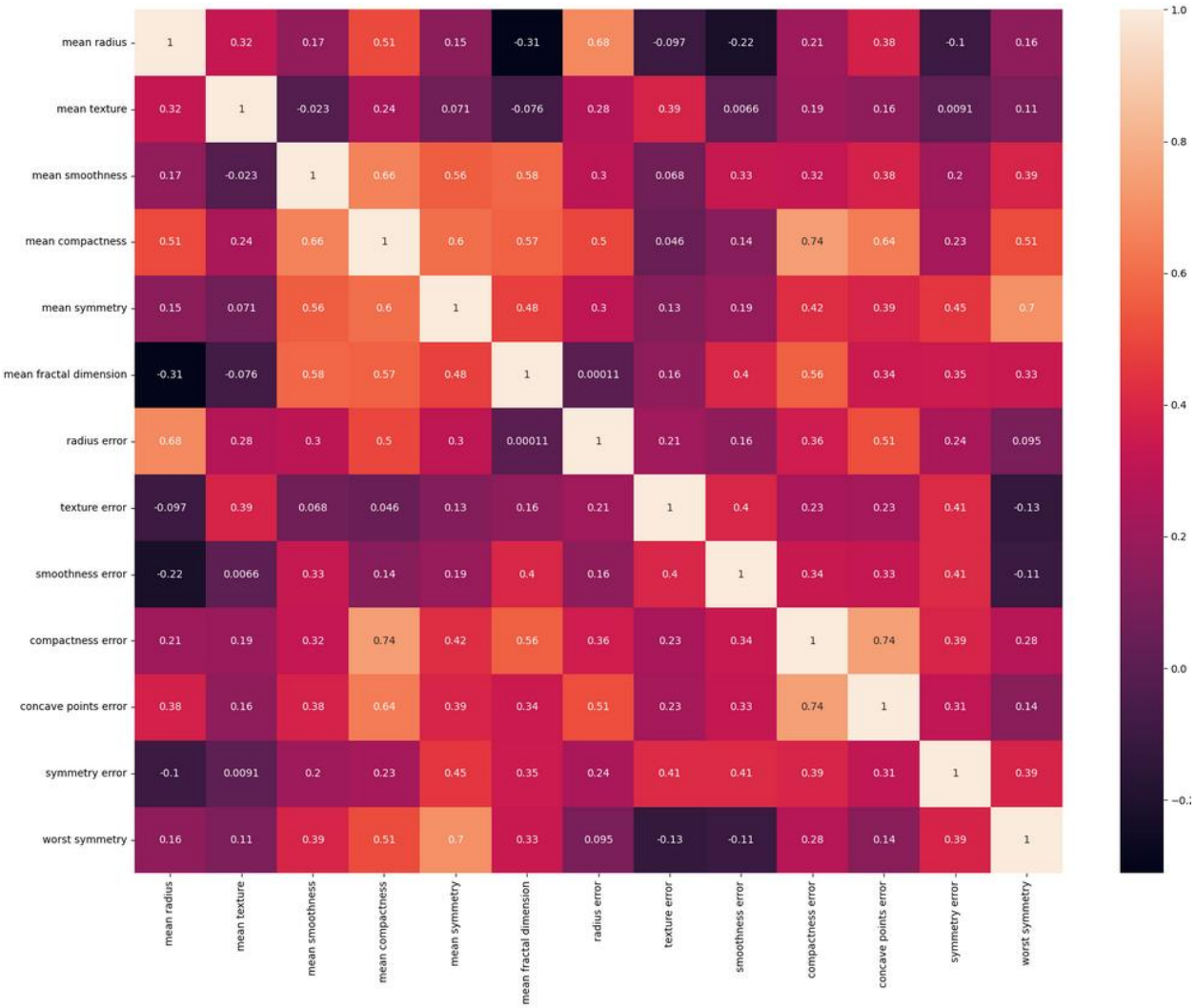



```
scaler = MinMaxScaler()

scaler.fit(data_df.drop('target', axis=1))
scaled_features = scaler.transform(data_df.drop('target',
```

```
                                                    axis=1))

        df_feat = pd.DataFrame(scaled_features,
                                    columns=data_df.columns[:-1])
        import seaborn as sns
        plt.figure(figsize=(20,15))
        ax = sns.heatmap(df_feat.corr(),annot=True)
        plt.show()
```




```
        corr_matrix = df_feat.corr().abs()
        # Верхний треугольник матрицы, чтобы не учитывать дубли
        upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape),
k=1).astype(bool))

        to_drop = [column for column in upper.columns if any(upper[column] > 0.8)]

        df_reduced = df_feat.drop(columns=to_drop)
        df_reduced
plt.figure(figsize=(20,15))
ax = sns.heatmap(df_reduced.corr(),annot=True)
plt.show()
```

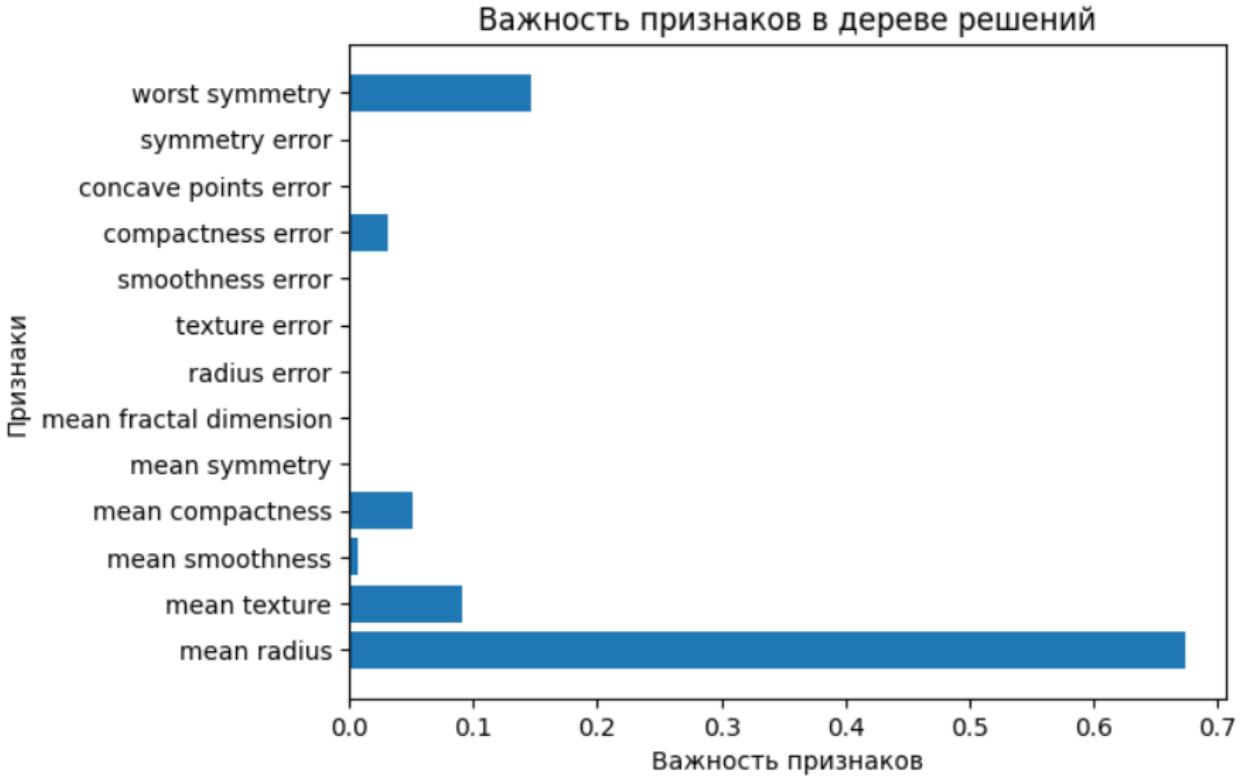



```
X = df_reduced
y = data_df["target"]
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
random_state=42)
# ------ Логрег
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression(random_state=42)
logreg.fit(X_train, y_train)
y_pred_logreg = logreg.predict(X_test)

# ------ SVM
from sklearn.svm import LinearSVC, SVC
# X = X.iloc[:, :2]
# svm = LinearSVC(C=1.0, loss='hinge')
svm = SVC(kernel='rbf', C=1.0, gamma='scale')
svm.fit(X, y)
y_pred_svm = svm.predict(X_test)
```

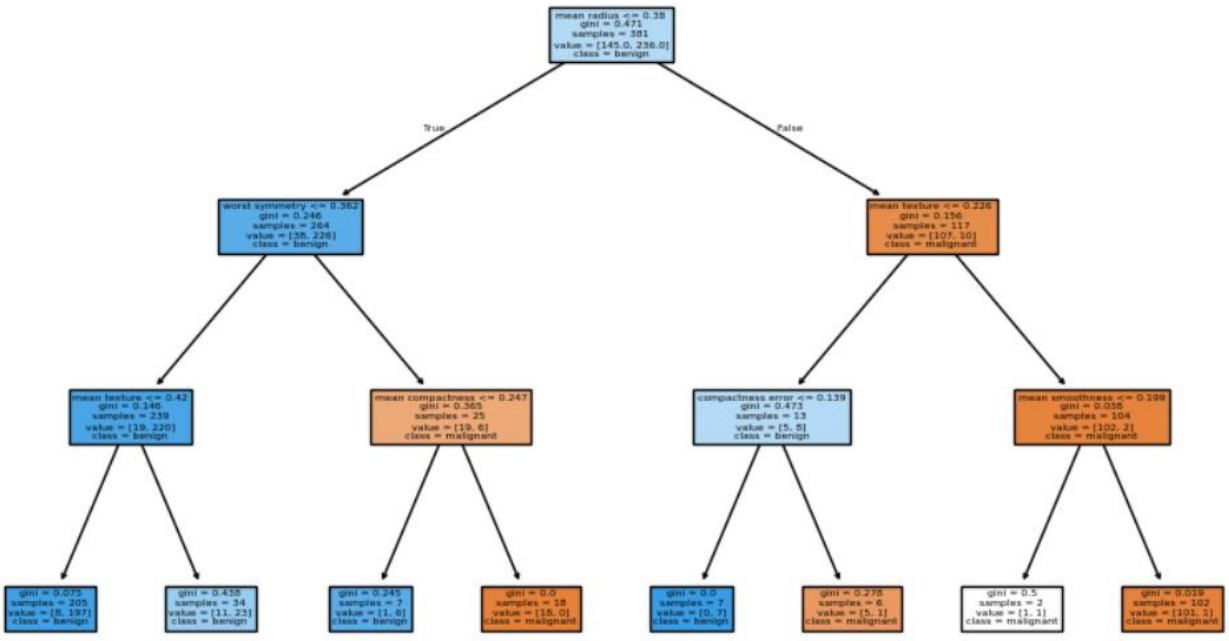
```
# ------ Дерево решений
from sklearn.tree import DecisionTreeClassifier, plot_tree

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

from sklearn.metrics import accuracy_score, classification_report
# print("Логрег accuracy:", accuracy_score(y_test, y_pred_logreg))
# print("SVM accuracy:", accuracy_score(y_test, y_pred_svm))
# print("Дерево решений accuracy:", accuracy_score(y_test, y_pred_tree))

print("\nЛогрег репорт\n", classification_report(y_test, y_pred_logreg,
digits=4))
print("\nSVM репорт\n", classification_report(y_test, y_pred_svm, digits=4))
print("\nДерево решений репорт\n", classification_report(y_test, y_pred_tree,
digits=4))
```

```
Логрег репорт
              precision    recall  f1-score   support

           0     0.9516    0.8806    0.9147        67
           1     0.9365    0.9752    0.9555       121

    accuracy                         0.9415       188
   macro avg     0.9441    0.9279    0.9351       188
weighted avg     0.9419    0.9415    0.9409       188


SVM репорт
              precision    recall  f1-score   support

           0     0.9552    0.9552    0.9552        67
           1     0.9752    0.9752    0.9752       121

    accuracy                         0.9681       188
   macro avg     0.9652    0.9652    0.9652       188
weighted avg     0.9681    0.9681    0.9681       188


Дерево решений репорт
              precision    recall  f1-score   support

           0     0.8971    0.9104    0.9037        67
           1     0.9500    0.9421    0.9461       121

    accuracy                         0.9309       188
   macro avg     0.9235    0.9263    0.9249       188
weighted avg     0.9311    0.9309    0.9310       188
```

```
# Важность признаков в дереве решений
feature_importances = tree.feature_importances_
plt.barh(X.columns, feature_importances)
plt.xlabel("Важность признаков")
plt.ylabel("Признаки")
plt.title("Важность признаков в дереве решений")
plt.show()
```




```
# Визуализация дерева решений
plt.figure(figsize=(10, 6))
plot_tree(tree, feature_names=X.columns, class_names=["malignant", "benign"],
filled=True)
plt.show()
```



```
# Правила дерева решений в текстовом виде
from sklearn.tree import export_text
tree_rules = export_text(tree, feature_names=list(X.columns))
```

```
print("Правила решений дерева в текстовом виде:")
print(tree_rules)
```

```
Правила решений дерева в текстовом виде:
|--- mean radius <= 0.38
|   |--- worst symmetry <= 0.36
|   |   |--- mean texture <= 0.42
|   |   |   |--- class: 1
|   |   |--- mean texture >  0.42
|   |   |   |--- class: 1
|   |--- worst symmetry >  0.36
|   |   |--- mean compactness <= 0.25
|   |   |   |--- class: 1
|   |   |--- mean compactness >  0.25
|   |   |   |--- class: 0
|--- mean radius >  0.38
|   |--- mean texture <= 0.23
|   |   |--- compactness error <= 0.14
|   |   |   |--- class: 1
|   |   |--- compactness error >  0.14
|   |   |   |--- class: 0
|   |--- mean texture >  0.23
|   |   |--- mean smoothness <= 0.20
|   |   |   |--- class: 0
|   |   |--- mean smoothness >  0.20
|   |   |   |--- class: 0
```