**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет Радиотехнический

Кафедра РТ5

Курс «Технологии машинного обучения»

Отчет по лабораторной работе №5

«Ансамбли машинного обучения. Часть 1.»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Руководитель: |
| студент группы РТ5-61Б: |  | преподаватель каф. ИУ5 |
| Бабасанова Н. С. |  | Гапанюк Ю.Е. |

Москва, 2025г.

**Задание**

1. Выберите набор данных (датасет) для решения задачи классификации или регресии.
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 ансамблевые модели:
   * две модели группы бэггинга (бэггинг или случайный лес или сверхслучайные деревья);
   * AdaBoost;
   * градиентный бустинг.
5. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.

**Текст программы**

import numpy as np

import pandas as pd

import sklearn as sk

import seaborn as sns

import matplotlib.pyplot as plt

from mlxtend.plotting import plot\_decision\_regions

from sklearn.preprocessing import MinMaxScaler

from sklearn.datasets import load\_breast\_cancer

from sklearn.tree import DecisionTreeClassifier

from sklearn.svm import SVC

from sklearn.ensemble import BaggingClassifier, AdaBoostClassifier, RandomForestClassifier, GradientBoostingClassifier

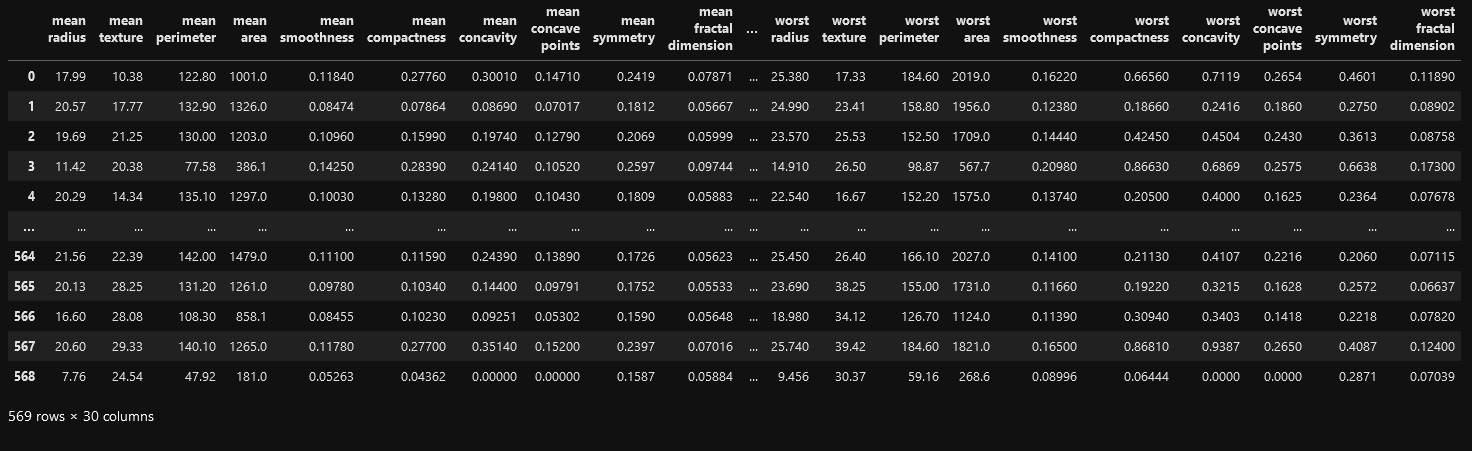
from sklearn.model\_selection import train\_test\_split

from sklearn.utils.class\_weight import compute\_sample\_weightdata = load\_breast\_cancer()

data\_df = pd.DataFrame(data = data.data,

columns = data.feature\_names)

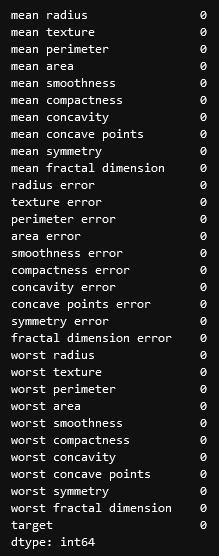
data\_df



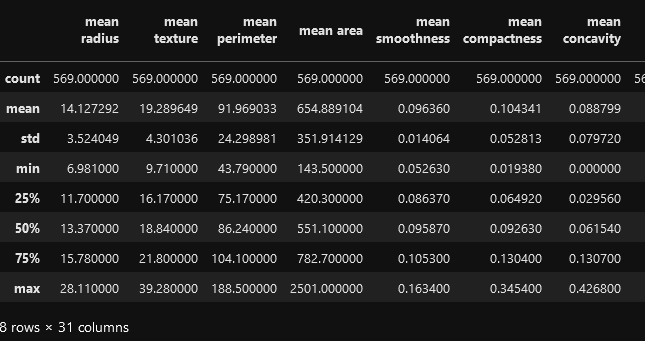
data\_df["target"] = data.target

data\_df

data\_df.isnull().sum()

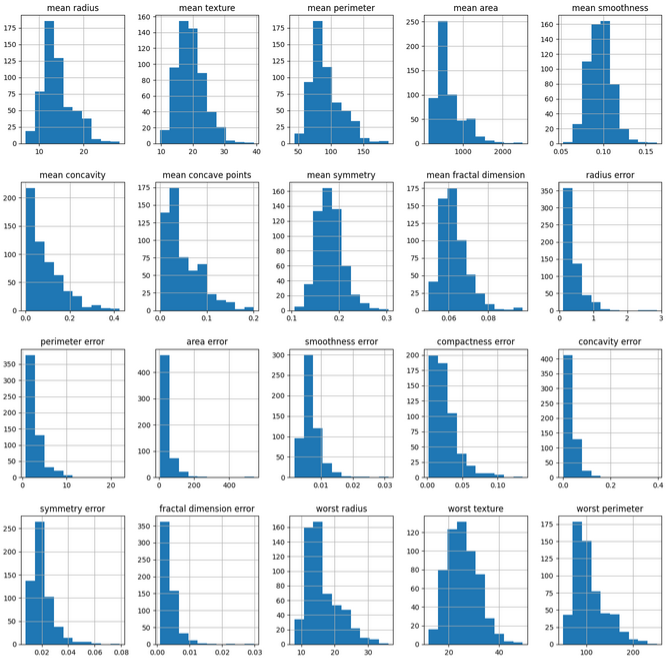


data\_df.describe()



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,

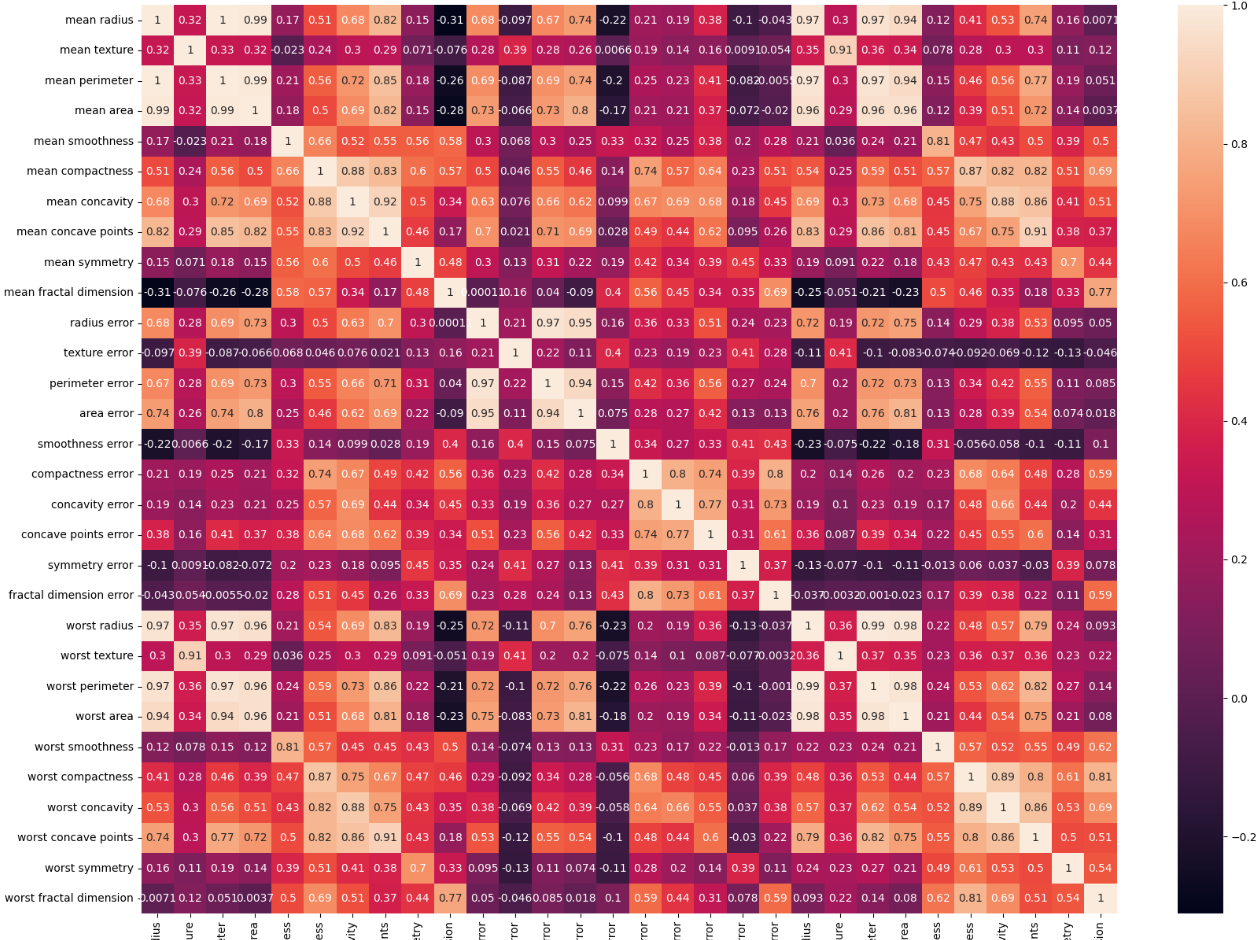
columns=data\_df.columns[:-1])

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)]

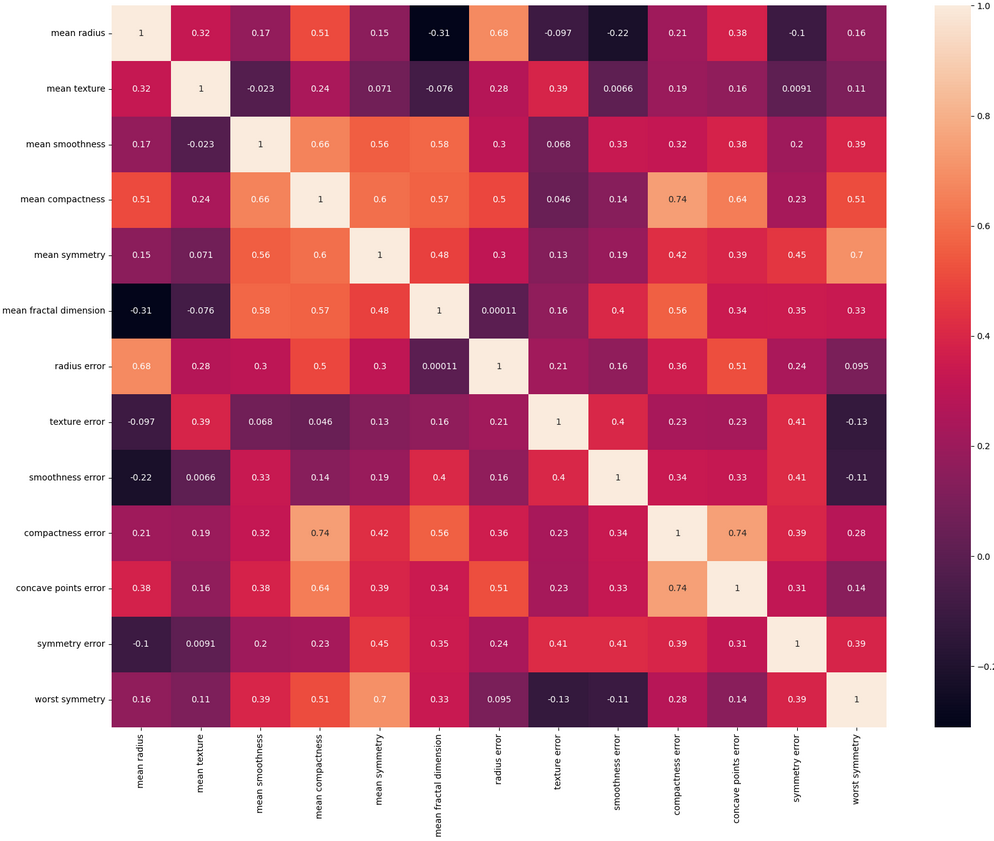
df\_reduced = df\_feat.drop(columns=to\_drop)

df\_reduced

plt.figure(figsize=(20,15))

ax = sns.heatmap(df\_reduced.corr(),annot=True)

plt.show()



X = df\_reduced

y = data\_df["target"]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42, stratify=y)

Бэггинг

model = BaggingClassifier(

estimator=SVC(class\_weight='balanced', random\_state=42),

n\_estimators=50,

max\_samples=0.8,

bootstrap=True,

random\_state=42,

n\_jobs=-1

)

classifiers = model.fit(X\_train, y\_train)

Случайный лес

clf\_rf = RandomForestClassifier(max\_depth=2, random\_state=0)

clf\_rf.fit(X, y)

AdaBoost

base\_estimator = DecisionTreeClassifier(

max\_depth=1,

class\_weight='balanced'

)

clf = AdaBoostClassifier(

estimator=base\_estimator,

n\_estimators=50,

random\_state=0

)

clf.fit(X, y)

Градиентный бустинг

# grad = GradientBoostingClassifier(n\_estimators=100, learning\_rate=0.1, max\_depth=1, random\_state=0).fit(X\_train, y\_train)

sample\_weights = compute\_sample\_weight(class\_weight='balanced', y=y\_train)

grad = GradientBoostingClassifier(

n\_estimators=50,

learning\_rate=0.1,

max\_depth=3,

random\_state=0

).fit(X\_train, y\_train, sample\_weight=sample\_weights)

Оценка моделей

bag\_pred = model.predict(X\_test)

clf\_rf\_pred = clf\_rf.predict(X\_test)

ada\_boost\_pred = clf.predict(X\_test)

grad\_boost\_pred = grad.predict(X\_test)

print(f"Бэггинг f1: {f1\_score(bag\_pred, y\_test, average='weighted')}, accuracy: {accuracy\_score(bag\_pred, y\_test)}")



print(f"RanForrest f1: {f1\_score(clf\_rf\_pred, y\_test, average='weighted')}, accuracy: {accuracy\_score(clf\_rf\_pred, y\_test)}")



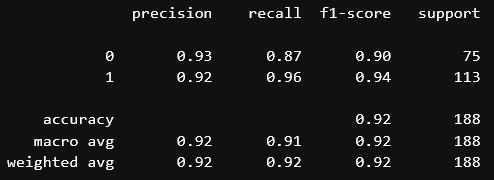
print(f"AdaBoost f1: {f1\_score(ada\_boost\_pred, y\_test, average='weighted')}, accuracy: {accuracy\_score(ada\_boost\_pred, y\_test)}")



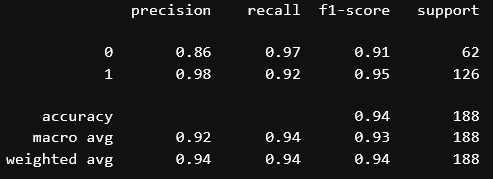
print(f"Градиентный бустинг f1: {f1\_score(grad\_boost\_pred, y\_test, average='weighted')}, accuracy: {accuracy\_score(grad\_boost\_pred, y\_test)}")



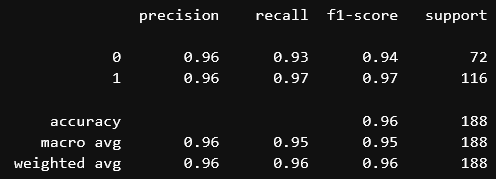
print(classification\_report(bag\_pred, y\_test))



print(classification\_report(clf\_rf\_pred, y\_test))



print(classification\_report(ada\_boost\_pred, y\_test))



print(classification\_report(grad\_boost\_pred, y\_test))

