**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 №5

«Ансамбли моделей машинного обучения.»

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 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: одну из моделей группы бэггинга (бэггинг или случайный лес или сверхслучайные деревья); одну из моделей группы бустинга; одну из моделей группы стекинга.
5. (+1 балл на экзамене) Дополнительно к указанным моделям обучите еще две модели: Модель многослойного персептрона. По желанию, вместо библиотеки scikit-learn возможно использование библиотек TensorFlow, PyTorch или других аналогичных библиотек. Модель МГУА с использованием библиотеки - https://github.com/kvoyager/GmdhPy (или аналогичных библиотек). Найдите такие параметры запуска модели, при которых она будет по крайней мере не хуже, чем одна из предыдущих ансамблевых моделей.
6. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.

Ход работы import numpy as np  
import pandas as pd  
from typing import Dict, Tuple  
from scipy import stats  
from IPython.display import Image  
from io import StringIO   
from IPython.display import Image  
import graphviz   
import pydotplus  
from sklearn.datasets import load\_diabetes  
from sklearn.model\_selection import cross\_val\_score  
from sklearn.model\_selection import train\_test\_split  
from sklearn.model\_selection import GridSearchCV, RandomizedSearchCV  
from sklearn.metrics import accuracy\_score, balanced\_accuracy\_score  
from sklearn.metrics import precision\_score, recall\_score, f1\_score, classification\_report  
from sklearn.metrics import confusion\_matrix  
from sklearn.tree import DecisionTreeRegressor, export\_graphviz  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.ensemble import ExtraTreesRegressor  
from sklearn.ensemble import GradientBoostingRegressor  
from sklearn.ensemble import BaggingRegressor  
from sklearn.linear\_model import LinearRegression  
from sklearn.ensemble import AdaBoostRegressor  
from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, mean\_squared\_log\_error, median\_absolute\_error, r2\_score   
from sklearn.metrics import roc\_curve, roc\_auc\_score  
import seaborn as sns  
import matplotlib.pyplot as plt  
from heamy.estimator import Regressor, Classifier  
from heamy.pipeline import ModelsPipeline  
from heamy.dataset import Dataset  
from sklearn.neural\_network import MLPRegressor  
%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

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

def get\_mean\_absolute\_error(model):   
 target1 = model.predict(X\_test)  
 return mean\_absolute\_error(y\_test, target1)

# Boosting

ab1 = AdaBoostRegressor(n\_estimators=5, random\_state=10)  
ab1.fit(X\_train, y\_train)  
  
bst\_error = get\_mean\_absolute\_error(ab1)  
  
ls = [bst\_error]  
print(ls)  
bst\_error

[53.761369406388894]

53.761369406388894

# Bagging

bag1 = BaggingRegressor(n\_estimators=5, oob\_score=True, random\_state=10)  
bag1.fit(X\_train, y\_train)  
  
bag\_error = get\_mean\_absolute\_error(bag1)  
  
ls.append(bag\_error)  
print(ls)  
get\_mean\_absolute\_error(bag1)

[53.761369406388894, 54.43728813559323]

C:\Users\prite\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9\_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\sklearn\ensemble\\_bagging.py:1253: UserWarning: Some inputs do not have OOB scores. This probably means too few estimators were used to compute any reliable oob estimates.  
 warn(

54.43728813559323

# Stacking

dataset = Dataset(X\_train, y\_train, X\_test)

# Первый уровень - две модели: дерево и линейная регрессия  
# Второй уровень: линейная регрессия  
  
model\_tree = Regressor(dataset=dataset, estimator=DecisionTreeRegressor, name='tree')  
pipeline = ModelsPipeline(model\_tree)  
stack\_ds = pipeline.stack(k=10, seed=1)  
# модель второго уровня  
stacker = Regressor(dataset=stack\_ds, estimator=LinearRegression)  
results = stacker.validate(k=10,scorer=mean\_absolute\_error)  
print(ls)

Metric: mean\_absolute\_error  
Folds accuracy: [66.51167877786833, 69.3507471513273, 48.40985503104768, 61.796529468775475, 54.27968045168703, 40.43980193427986, 48.07677534665627, 38.735365752869015, 57.848093689165, 77.01094737191416]  
Mean accuracy: 56.245947497559015  
Standard Deviation: 11.958857389319805  
Variance: 143.0142700580889  
[53.761369406388894, 54.43728813559323]

# MLP

mlp1 = MLPRegressor()  
mlp1.fit(X\_train, y\_train)  
  
mlp1\_error = get\_mean\_absolute\_error(mlp1)  
  
ls.append(mlp1\_error)  
  
print(ls)  
mlp1\_error

C:\Users\prite\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9\_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\sklearn\neural\_network\\_multilayer\_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.  
 warnings.warn(

[53.761369406388894, 54.43728813559323, 145.53921997372854]

145.53921997372854

# Gmdh

from gmdhpy import gmdh  
gmdh1 = gmdh.Regressor()  
  
gmdh1.fit(X\_train, y\_train)  
  
gmdh1\_error = get\_mean\_absolute\_error(gmdh1)  
  
ls.append(gmdh1\_error)  
print(ls)  
gmdh1\_error

train layer0 in 0.04 sec  
train layer1 in 0.18 sec  
train layer2 in 0.16 sec  
train layer3 in 0.15 sec  
train layer4 in 0.16 sec  
train layer5 in 0.16 sec  
train layer6 in 0.20 sec  
train layer7 in 0.15 sec  
train layer8 in 0.16 sec  
train layer9 in 0.16 sec  
train layer10 in 0.15 sec  
train layer11 in 0.15 sec  
[53.761369406388894, 54.43728813559323, 145.53921997372854, 50.95011646397348]

50.95011646397348

x = np.arange(len(ls))  
plt.bar(x, height=ls)  
plt.xticks(x, ['Boosting','Bagging', 'MLP', 'GmdhPy']);  
ls

[53.761369406388894, 54.43728813559323, 145.53921997372854, 50.95011646397348]

