**Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)**

Факультет «Информатика и вычислительная техника»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

# Отчет по лабораторной работе №5
# «Ансамбли моделей машинного обучения»

Выполнил:
студент группы ИУ5-63Б
Рыбина А.Д.

Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Подпись и дата:

Москва, 2022 г.

**Цель лабораторной работы:** изучение ансамблей моделей машинного обучения.

**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.

# Текст программы и результаты ее выполнения

## Лабораторная работа №5. Ансамбли моделей машинного обучения.

### 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:
   - одну из моделей группы бэггинга (бэггинг или случайный лес или сверхслучайные деревья);
   - одну из моделей группы бустинга;
   - одну из моделей группы стекинга.
5. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.

In [43]:
```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.linear_model import RidgeClassifier
import seaborn as sns
%matplotlib inline
sns.set(style="ticks")
```

In [44]:
```
# Выберем исходный набор данных и проведём разбиение на обучающую и тестовую выборки
iris = load_iris()

iris_X_train, iris_X_test, iris_y_train, iris_y_test = train_test_split(
    iris.data, iris.target, test_size=0.5, random_state=1)
```

In [45]:
```
# Обучение модели бэггинга
bg = BaggingClassifier(n_estimators=15, oob_score=True, random_state=1)
bg.fit(iris_X_train, iris_y_train)
```

Out[45]:
```
BaggingClassifier(n_estimators=15, oob_score=True, random_state=1)
```

In [46]:
```
# Оценка качества модели
accuracy_score(iris_y_test, bg.predict(iris_X_test))
```

Out[46]:
```
0.9466666666666667
```

In [47]:
```
# Обучение модели бустинга (градиентный спуск)
gb = GradientBoostingClassifier(n_estimators=15, random_state=1)
gb.fit(iris_X_train, iris_y_train)
```

Out[47]:
```
GradientBoostingClassifier(n_estimators=15, random_state=1)
```

In [48]:
```
# Оценка качества модели
accuracy_score(iris_y_test, gb.predict(iris_X_test))
```

Out[48]:
```
0.96
```

In [49]:
```
from heamy.estimator import Regressor
from heamy.pipeline import ModelsPipeline
from heamy.dataset import Dataset

dataset = Dataset(iris_X_train, iris_y_train, iris_X_test, iris_y_test)

model_tree = Regressor(dataset=dataset, estimator=DecisionTreeClassifier, name='tree')
model_lr = Regressor(dataset=dataset, estimator=RidgeClassifier,name='lr')
model_rf = Regressor(dataset=dataset, estimator=RandomForestClassifier, parameters={'n_estimators': 50},name='rf')

pipeline = ModelsPipeline(model_tree, model_lr, model_rf)
stack_ds = pipeline.stack(k=10, seed=1)
# модель второго уровня
stacker = Regressor(dataset=stack_ds, estimator=DecisionTreeClassifier)
results = stacker.validate(k=10,scorer=accuracy_score)
```

```
Metric: accuracy_score
Folds accuracy: [1.0, 0.75, 1.0, 0.875, 1.0, 1.0, 0.7142857142857143, 0.8571428571428571, 0.8571428571428571, 0.8571
428571428571]
Mean accuracy: 0.8910714285714286
Standard Deviation: 0.1011572074162039
Variance: 0.0102327806122449
```