**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика с системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №5

# «Ансамбли моделей машинного обучения[.](https://github.com/ugapanyuk/courses_current/wiki/LAB_TMO__MISSING)»

Выполнил: Проверил:

студент группы РТ5-61Б преподаватель каф. ИУ5

Мицкевич В.Б. Гапанюк Ю.Е.

Подпись и дата: Подпись и дата:

Москва, 2023 г.

# Ансамбли моделей машинного обучения

# Цель лабораторной работы: изучение ансамблей 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:
5. одну из моделей группы бэггинга (бэггинг или случайный лес или сверхслучайные деревья);  
     
   одну из моделей группы бустинга;  
     
   одну из моделей группы стекинга.
6. (+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.
7. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.
9. import pandas as pd  
   import numpy as np  
   import seaborn as sns  
   import matplotlib.pyplot as plt  
   from sklearn.datasets import \*  
   from sklearn.preprocessing import MinMaxScaler  
   from sklearn.model\_selection import train\_test\_split  
   from sklearn.linear\_model import LogisticRegression  
   from sklearn import svm, tree  
   from sklearn.tree import DecisionTreeClassifier  
   from sklearn.ensemble import BaggingClassifier  
   from sklearn.ensemble import GradientBoostingClassifier  
   from sklearn.ensemble import StackingClassifier  
   from sklearn.neural\_network import MLPClassifier  
   from sklearn.metrics import accuracy\_score  
   from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay  
   from operator import itemgetter  
     
     
   def make\_dataframe(ds\_function):  
    ds = ds\_function()  
    df = pd.DataFrame(data= np.c\_[ds['data'], ds['target']],  
    columns= list(ds['feature\_names']) + ['target'])  
    return df  
     
   wine = load\_wine()  
     
   df = make\_dataframe(load\_wine)
10. df.info()
11. <class 'pandas.core.frame.DataFrame'>  
    RangeIndex: 178 entries, 0 to 177  
    Data columns (total 14 columns):  
     # Column Non-Null Count Dtype   
    --- ------ -------------- -----   
     0 alcohol 178 non-null float64  
     1 malic\_acid 178 non-null float64  
     2 ash 178 non-null float64  
     3 alcalinity\_of\_ash 178 non-null float64  
     4 magnesium 178 non-null float64  
     5 total\_phenols 178 non-null float64  
     6 flavanoids 178 non-null float64  
     7 nonflavanoid\_phenols 178 non-null float64  
     8 proanthocyanins 178 non-null float64  
     9 color\_intensity 178 non-null float64  
     10 hue 178 non-null float64  
     11 od280/od315\_of\_diluted\_wines 178 non-null float64  
     12 proline 178 non-null float64  
     13 target 178 non-null float64  
    dtypes: float64(14)  
    memory usage: 19.6 KB
12. def count\_nan(data):  
     for col in data.columns:  
     count\_nan = data[data[col].isnull()].shape[0]  
     print('{} имеет NAN: {}'.format(col, count\_nan))  
    count\_nan(df)
13. alcohol имеет NAN: 0  
    malic\_acid имеет NAN: 0  
    ash имеет NAN: 0  
    alcalinity\_of\_ash имеет NAN: 0  
    magnesium имеет NAN: 0  
    total\_phenols имеет NAN: 0  
    flavanoids имеет NAN: 0  
    nonflavanoid\_phenols имеет NAN: 0  
    proanthocyanins имеет NAN: 0  
    color\_intensity имеет NAN: 0  
    hue имеет NAN: 0  
    od280/od315\_of\_diluted\_wines имеет NAN: 0  
    proline имеет NAN: 0  
    target имеет NAN: 0

# Разделение на тестовую и обучающую выборки

1. y = df['target']  
   x = df.drop('target', axis = 1)  
     
   scaler = MinMaxScaler()  
   scaled\_data = scaler.fit\_transform(x)  
     
   x\_train, x\_test, y\_train, y\_test = train\_test\_split(scaled\_data, y, test\_size = 0.3, random\_state = 1)

## Про ансамбли

1. <https://habr.com/ru/articles/561732/>
2. <https://academy.yandex.ru/handbook/ml/article/ansambli-v-mashinnom-obuchenii>

# Бэггинг

1. bc = BaggingClassifier(n\_estimators=8, oob\_score=True, random\_state=10)  
   bc\_model = bc.fit(x\_train, y\_train)  
   bc\_predict = bc\_model.predict(x\_test)
2. /home/vlad/anaconda3/lib/python3.9/site-packages/sklearn/ensemble/\_bagging.py:640: UserWarning: Some inputs do not have OOB scores. This probably means too few estimators were used to compute any reliable oob estimates.  
    warn("Some inputs do not have OOB scores. "  
   /home/vlad/anaconda3/lib/python3.9/site-packages/sklearn/ensemble/\_bagging.py:644: RuntimeWarning: invalid value encountered in true\_divide  
    oob\_decision\_function = (predictions /

# Градиентный бустинг

1. gb = GradientBoostingClassifier(random\_state=0)  
   gb\_model = gb.fit(x\_train, y\_train)  
   gb\_predict = gb\_model.predict(x\_test)

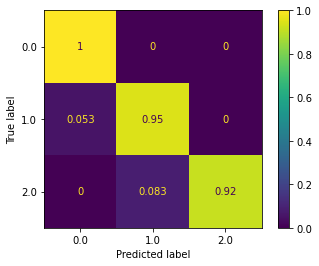
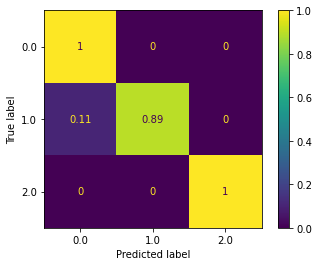
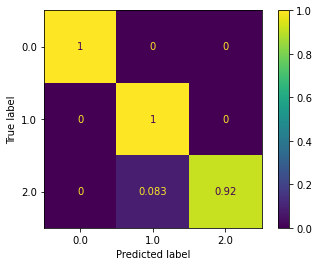
# Стекинг

1. # Качество отдельных моделей  
   def val\_mae(model):  
    st\_prediction = model.fit(x\_train, y\_train).predict(x\_test)  
    print(model)  
    print('Accuracy score={}'.format(accuracy\_score(y\_test, st\_prediction)))
2. # Точность на отдельных моделях  
   for model in [  
    LogisticRegression(random\_state=1),  
    svm.SVC(random\_state=1),  
    DecisionTreeClassifier(random\_state=1)  
   ]:  
    val\_mae(model)  
    print('==========================')  
    print()
3. LogisticRegression(random\_state=1)  
   Accuracy score=0.9814814814814815  
   ==========================  
     
   SVC(random\_state=1)  
   Accuracy score=0.9814814814814815  
   ==========================  
     
   DecisionTreeClassifier(random\_state=1)  
   Accuracy score=0.9444444444444444  
   ==========================
4. estimators = [  
    ('lg', LogisticRegression(random\_state=0)),  
    ('svc', svm.SVC(random\_state=0))  
   ]  
   LogisticRegression  
   sc\_class = StackingClassifier(  
    estimators=estimators, final\_estimator=DecisionTreeClassifier()  
   )  
   sc\_model = sc\_class.fit(x\_train, y\_train)  
   sc\_predict = sc\_model.predict(x\_test)

## Модель МГУА (метод группового учёта аргументов)

1. mgua = Classifier()  
   mgua\_prediction = mgua.fit(x\_train, y\_train).predict(x\_test)  
   accuracy\_score(y\_test, mgua\_prediction)
2. ---------------------------------------------------------------------------  
   NameError Traceback (most recent call last)  
   /tmp/ipykernel\_43159/1357388246.py in <module>  
   ----> 1 mgua = Classifier()  
    2 mgua\_prediction = mgua.fit(x\_train, y\_train).predict(x\_test)  
    3 accuracy\_score(y\_test, mgua\_prediction)  
     
   NameError: name 'Classifier' is not defined

# Оценка качества решений

1. print("Бэггинг: ", accuracy\_score(y\_test, bc\_predict))  
   print("Градиентный бустинг: ", accuracy\_score(y\_test, gb\_predict))  
   print("Стекинг (дерево и метод опорных векторов + логистическая регрессия): ", accuracy\_score(y\_test, sc\_predict))
2. Бэггинг: 0.9629629629629629  
   Градиентный бустинг: 0.9629629629629629  
   Стекинг (дерево и метод опорных векторов + логистическая регрессия): 0.9814814814814815
3. print("Бэггинг")  
     
   cm = confusion\_matrix(y\_test, bc\_predict, labels=np.unique(df.target), normalize='true')  
   disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=np.unique(df.target))  
   disp.plot()
4. Бэггинг
5. <sklearn.metrics.\_plot.confusion\_matrix.ConfusionMatrixDisplay at 0x7f3951eff580>
6. 
7. print("Градиентный бустинг")  
     
   cm = confusion\_matrix(y\_test, gb\_predict, labels=np.unique(df.target), normalize='true')  
   disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=np.unique(df.target))  
   disp.plot()
8. Градиентный бустинг
9. <sklearn.metrics.\_plot.confusion\_matrix.ConfusionMatrixDisplay at 0x7f3950fa4bb0>
10. 
11. print("Стекинг (логистическая регрессия дерево и метод опорных векторов + дерево)")  
      
    cm = confusion\_matrix(y\_test, sc\_predict, labels=np.unique(df.target), normalize='true')  
    disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=np.unique(df.target))  
    disp.plot()
12. Стекинг (логистическая регрессия дерево и метод опорных векторов + дерево)
13. <sklearn.metrics.\_plot.confusion\_matrix.ConfusionMatrixDisplay at 0x7f3950ec5820>
    1. ****