Лабораторная работа 5

Задание

Выберите набор данных (датасет) для решения задачи классификации или регресии.

В случае необходимости проведите удаление или заполнение пропусков и кодирование категориальных признаков.

С использованием метода train_test_split разделите выборку на обучающую и тестовую.

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.

Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.

Выбор и загрузка данных

В качестве датасета будем использовать набор данных, содержащий данные о трудоустройстве студентов. Данный набор доступен по адресу: https://www.kaggle.com/datasets/uciml/student-alcohol-consumption?select=student-por.csv

sl_no - серийный номер (номер в датасете);

gender - пол;

ssc_p - процент среднего образования;

ssc_b - Министерство образования (центральное или другое);

hsc_p - процент высшего образования;

hsc_b - Министерство образования (центральное или другое);

hsc_s - специализация полного среднего образования;

degree_p - процент выпустившихся;

degree_t - бакалавриат (сфера образования);

workex - опыт работы;

etest_p - процент теста на трудоустройство;

specialisation - специальность после выпуска;

mba_p - MBA процент;

status - статус трудоустройства (устроен или не устроен);

salary - запралата, которую предлагают кандидатам.

Импортируем библиотеки

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from google.colab import files
%matplotlib inline
sns.set(style="ticks")
```

Загружаем данные

```
files.upload()
```

```
Выбрать файлы Placement…lass.csv
• Placement_Data_Full_Class.csv(text/csv) - 19712 bytes, last modified: 11.04.2020 - 100% done
Saving Placement_Data_Full_Class.csv to Placement_Data_Full_Class.csv
{'Placement_Data_Full_Class.csv': b'sl_no,gender,ssc_p,ssc_b,hsc_p,hsc_b,hsc_s,degree
```

```
data = pd.read_csv('Placement_Data_Full_Class.csv')
```

Первичный анализ

Первые 5 строк датасета:

```
data.head()
```

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	wor
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	

Определим размер датасета:

```
data.shape
```

```
(215, 15)
```

Определим типы данных:

```
data.dtypes
```

```
sl_no              int64
gender            object
ssc_p            float64
ssc_b             object
hsc_p            float64
hsc_b             object
hsc_s             object
degree_p         float64
degree_t          object
workex            object
etest_p          float64
specialisation    object
mba_p            float64
status            object
salary           float64
dtype: object
```

Обработка пропусков

Проверим наличие пропусков:

```
data.isnull().sum()
```

```
sl_no             0
gender            0
ssc_p             0
ssc_b             0
hsc_p             0
hsc_b             0
hsc_s             0
degree_p          0
degree_t          0
workex            0
etest_p           0
specialisation    0
mba_p             0
status            0
salary           67
dtype: int64
```

Как мы видим, есть незаполненная зарплата, но поскольку она не заполнена из-за того, что данный человек не трудоустроен, то заполним поле зарплаты нулями.

```
data = data.fillna(0)
```

```
data.isnull().sum()
```

```
sl_no             0
gender            0
ssc_p             0
ssc_b             0
hsc_p             0
hsc_b             0
hsc_s             0
degree_p          0
degree_t          0
workex            0
etest_p           0
specialisation    0
mba_p             0
status            0
salary            0
dtype: int64
```

Теперь датасет выглядит следующим образом:

```
data.head()
```

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	wor
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	

Оптимизация данных

Для кодирования столбцов категорий будем использовать LabelEncoder:

```
from sklearn.preprocessing import LabelEncoder
```

```
legend = LabelEncoder()
legendarr = legend.fit_transform(data["gender"])
```

```
data["gender"] = legendarr
data = data.astype({"gender":"float"})

lerace = LabelEncoder()
leracearr = lerace.fit_transform(data["ssc_b"])
data["ssc_b"] = leracearr
data = data.astype({"ssc_b":"float"})

leeduc = LabelEncoder()
leeducarr = leeduc.fit_transform(data["hsc_b"])
data["hsc_b"] = leeducarr
data = data.astype({"hsc_b":"float"})

lelunch = LabelEncoder()
leluncharr = lelunch.fit_transform(data["hsc_s"])
data["hsc_s"] = leluncharr
data = data.astype({"hsc_s":"float"})

leprep = LabelEncoder()
lepreparr = leprep.fit_transform(data["degree_t"])
data["degree_t"] = lepreparr
data = data.astype({"degree_t":"float"})

lework = LabelEncoder()
leworkarr = leeduc.fit_transform(data["workex"])
data["workex"] = leworkarr
data = data.astype({"workex":"float"})

lespec = LabelEncoder()
lespecarr = lespec.fit_transform(data["specialisation"])
data["specialisation"] = lespecarr
data = data.astype({"specialisation":"float"})

lestatus = LabelEncoder()
lestatusarr = leprep.fit_transform(data["status"])
data["status"] = lestatusarr
data = data.astype({"status":"float"})
```

Проверим кодирование:

```
np.unique(legendarr), np.unique(leracearr), np.unique(leeducarr), np.unique(leluncharr), n
```

```
(array([0, 1]),
 array([0, 1]),
 array([0, 1]),
 array([0, 1, 2]),
 array([0, 1, 2]),
 array([0, 1]),
 array([0, 1]),
 array([0, 1]))
```

И замену в датасете:

```
data.head()
```

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	workex	etes
0	1	1.0	67.00	1.0	91.00	1.0	1.0	58.00	2.0	0.0	
1	2	1.0	79.33	0.0	78.33	1.0	2.0	77.48	2.0	1.0	
2	3	1.0	65.00	0.0	68.00	0.0	0.0	64.00	0.0	0.0	
3	4	1.0	56.00	0.0	52.00	0.0	2.0	52.00	2.0	0.0	
4	5	1.0	85.80	0.0	73.60	0.0	1.0	73.30	0.0	0.0	

Разделение выборки на обучающую и тестовую

Разделим выборку с помощью функции train_test_split:

```
from sklearn.model_selection import train_test_split

X = data.drop("salary", axis=1)
y = data["salary"]
```

Чтобы изменить содержимое ячейки, дважды нажмите на нее (или выберите "Ввод")

```
X_train, X_test, y_train, y_test = train_test_split(data, data["salary"], random_state=1)
```

Размеры обучающей выборки и тестовой выборки:

```
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
((161, 15), (161,), (54, 15), (54,))
```

Обучение ансамблевых моделей

Модель бэггинга

```
from sklearn.ensemble import BaggingRegressor
```

Обучим модель на 5 деревьях:

```
bagging_model = BaggingRegressor(n_estimators=5, oob_score=True, random_state=10)
```

```
bagging_model.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/_bagging.py:1164: UserWarning
  "Some inputs do not have OOB scores. "
BaggingRegressor(n_estimators=5, oob_score=True, random_state=10)
```

Сконвертируем объекты, которые были использованы в обучающей выборке каждого дерева, в двоичную матрицу (1 соответствует элементам, попавшим в обучающую выборку):

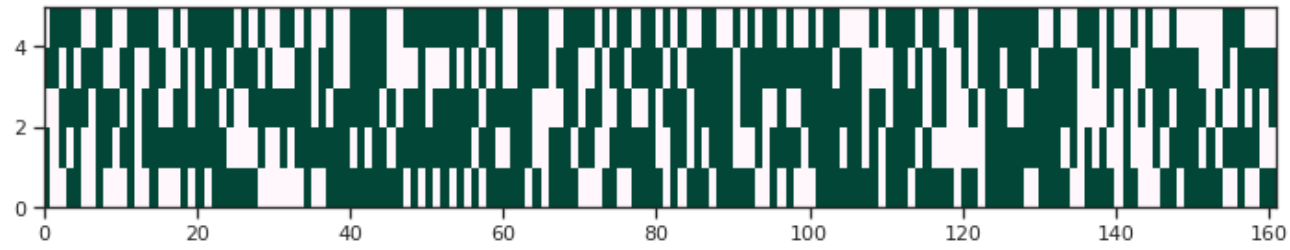
```
bin_array = np.zeros((5, X_train.shape[0]))
for i in range(5):
    for j in bagging_model.estimators_samples_[i]:
        bin_array[i][j] = 1
bin_array
```

```
array([[1., 0., 0., 1., 1., 0., 0., 1., 0., 0., 0., 1., 0., 0., 1., 0.,
        0., 0., 1., 0., 1., 0., 1., 1., 1., 1., 1., 1., 0., 0., 0., 0.,
        0., 0., 1., 0., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0.,
        1., 0., 1., 0., 1., 0., 1., 0., 1., 0., 1., 1., 1., 1., 1., 0.,
        1., 0., 1., 1., 1., 1., 0., 0., 0., 1., 1., 0., 0., 1., 1., 1.,
        1., 0., 1., 0., 1., 1., 1., 0., 1., 1., 1., 1., 1., 0., 0., 1.,
        0., 1., 0., 0., 0., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1.,
        1., 0., 1., 1., 1., 1., 1., 0., 1., 1., 0., 1., 1., 1., 1., 1.,
        0., 0., 1., 1., 1., 1., 0., 0., 1., 1., 1., 1., 0., 1., 1., 0.,
        0., 0., 1., 1., 0., 1., 1., 1., 1., 1., 0., 0., 1., 0., 0., 1.,
        1.],
       [1., 0., 1., 0., 1., 0., 0., 1., 1., 0., 1., 1., 0., 1., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 0., 0., 0., 0., 1., 0., 0., 1.,
        0., 1., 1., 1., 1., 1., 1., 1., 0., 1., 0., 1., 1., 0., 1., 1.,
        1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 0., 0., 1., 1., 1.,
        0., 0., 1., 1., 1., 0., 1., 1., 0., 0., 1., 1., 1., 1., 1., 1.,
        0., 0., 1., 1., 0., 1., 0., 1., 1., 1., 0., 0., 0., 1., 0., 1.,
        1., 1., 1., 0., 1., 1., 1., 1., 0., 1., 1., 0., 1., 0., 1., 1.,
        1., 1., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1.,
        1., 1., 1., 1., 1., 0., 1., 0., 1., 0., 1., 0., 0., 1., 0., 0.,
        0., 1., 0., 1., 1., 1., 1., 0., 1., 0., 1., 1., 1., 1., 1., 0.,
        0.],
       [0., 0., 1., 1., 1., 1., 0., 1., 1., 1., 0., 1., 0., 1., 1., 0.,
        0., 1., 1., 0., 1., 1., 1., 0., 1., 0., 0., 1., 1., 1., 1., 1.,
        1., 1., 1., 0., 1., 0., 1., 1., 1., 1., 1., 1., 0., 1., 0., 1.,
        1., 1., 0., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1.,
        0., 0., 0., 0., 1., 1., 0., 1., 1., 0., 0., 1., 1., 0., 0., 1.,
        0., 1., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1., 1., 1., 0., 0.,
        1., 0., 0., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 0., 1.,
        1., 1., 1., 0., 1., 1., 0., 0., 0., 1., 0., 1., 1., 1., 0., 0.,
        1., 1., 1., 1., 1., 1., 1., 0., 0., 0., 0., 1., 0., 1., 0., 1.,
        0., 1., 1., 0., 1., 1., 1., 1., 0., 1., 0., 0., 1., 0., 1., 0.,
        1.],
       [1., 1., 0., 1., 0., 1., 1., 1., 0., 0., 1., 1., 0., 0., 1., 1.,
        0., 0., 1., 0., 1., 0., 1., 1., 0., 1., 1., 1., 0., 1., 0., 0.,
        0., 1., 1., 0., 1., 1., 0., 0., 1., 1., 1., 1., 1., 0., 0., 0.,
        0., 1., 0., 0., 0., 1., 0., 1., 0., 1., 0., 1., 0., 0., 1., 1.,
        1., 1., 0., 1., 1., 1., 0., 0., 1., 0., 1., 1., 0., 1., 1., 1.,
        1., 0., 1., 0., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1.,
```

```
       1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 0., 0., 0., 0., 1.,
       1., 0., 1., 0., 1., 1., 0., 0., 1., 0., 1., 1., 1., 0., 1., 1.,
       1., 0., 1., 1., 1., 1., 1., 0., 0., 1., 0., 1., 1., 1., 0., 0.,
       1., 1., 1., 1., 1., 1., 1., 0., 0., 0., 1., 0., 1., 1., 1., 1.,
       1.],
      [0., 1., 1., 1., 1., 0., 0., 1., 1., 0., 0., 1., 1., 1., 1., 0.,
       0., 1., 0., 1., 1., 1., 1., 1., 1., 0., 1., 0., 1., 0., 0., 1.,
       1., 0., 1., 0., 0., 1., 0., 0., 1., 1., 1., 1., 1., 0., 0., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 0., 0., 1., 1.,
       1., 1., 0., 0., 1., 1., 1., 1., 0., 1., 0., 1., 1., 0., 1., 1.,
       0., 1., 0., 1., 0., 0., 1., 1., 0., 0., 1., 0., 1., 0., 1., 1.,
       0., 1., 0., 0., 1., 0., 1., 1., 0., 0., 0., 0., 1., 0., 1., 1.,
       1., 1., 0., 1., 1., 0., 0., 1., 1., 0., 1., 1., 1., 1., 1., 1.,
       1., 1., 0., 0., 1., 0., 0., 1., 1., 1., 0., 1., 0., 0., 1., 0.,
       1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 1., 1., 1., 0., 0., 0.,
       0.]])
```

Визуализируем эти данные:

```
fig, ax = plt.subplots(figsize=(12,2))
ax.pcolor(bin_array, cmap='PuBuGn')
plt.show()
```



Оценим Out-of-bag error, теоретическое значение 37%

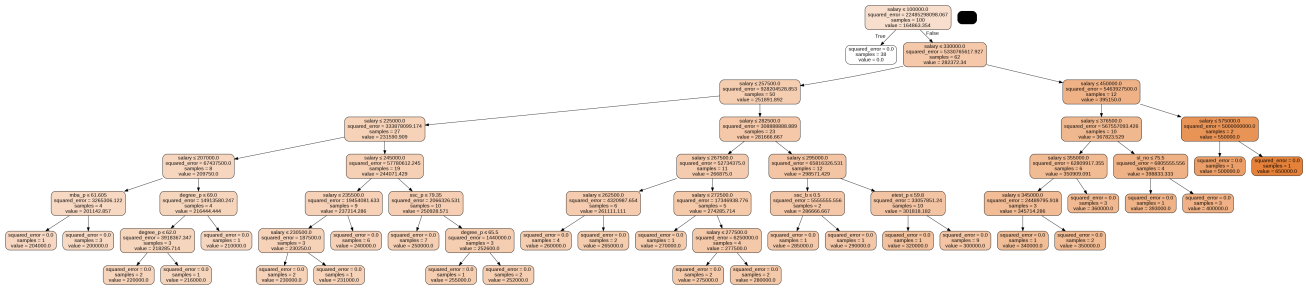
```
for i in range(5):
    cur_data = bin_array[i]
    len_cur_data = len(cur_data)
    sum_cur_data = sum(cur_data)
    (len(bin_array[0]) - sum(bin_array[0])) / len(bin_array[0])
    oob_i = (len_cur_data - sum_cur_data) / len_cur_data
    print('Для модели № {} размер OOB составляет {}%'.format(i+1, round(oob_i, 4)*100.0))
```

```
 Для модели № 1 размер OOB составляет 37.89%
 Для модели № 2 размер OOB составляет 35.4%
 Для модели № 3 размер OOB составляет 34.160000000000004%
 Для модели № 4 размер OOB составляет 36.65%
 Для модели № 5 размер OOB составляет 42.24%
```

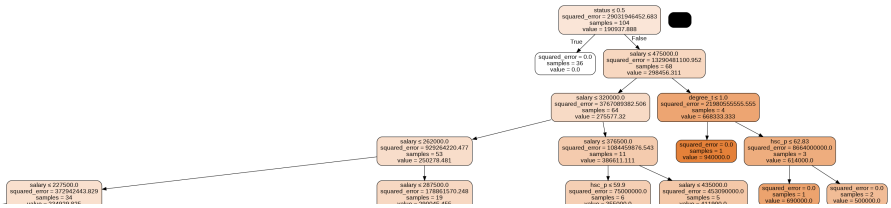
Визуализируем обученные деревья:

```
from io import StringIO
from IPython.display import Image
```

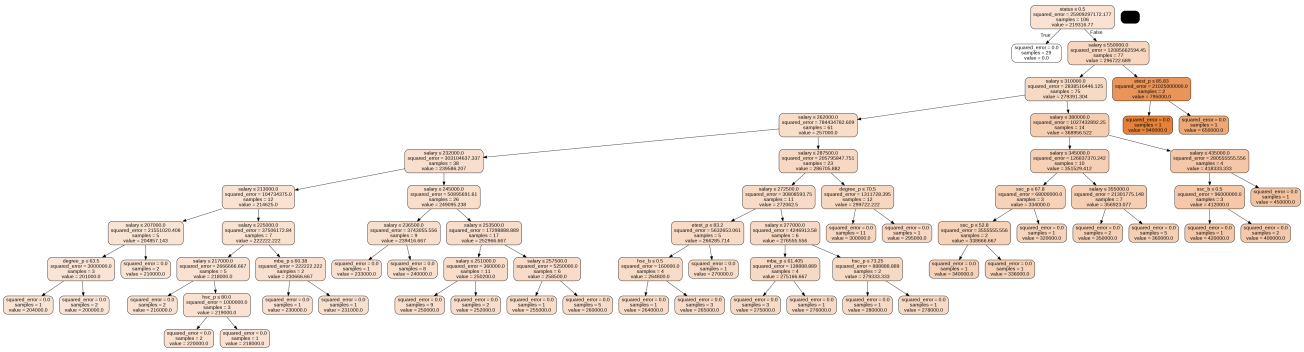
```
import graphviz
import pydotplus
from sklearn.tree import export_graphviz

def get_png_tree(tree_model_param, feature_names_param):
    dot_data = StringIO()
    export_graphviz(tree_model_param, out_file=dot_data, feature_names=feature_names_param
                    filled=True, rounded=True, special_characters=True)
    graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
    return graph.create_png()


Image(get_png_tree(bagging_model.estimators_[0], data.columns))
```



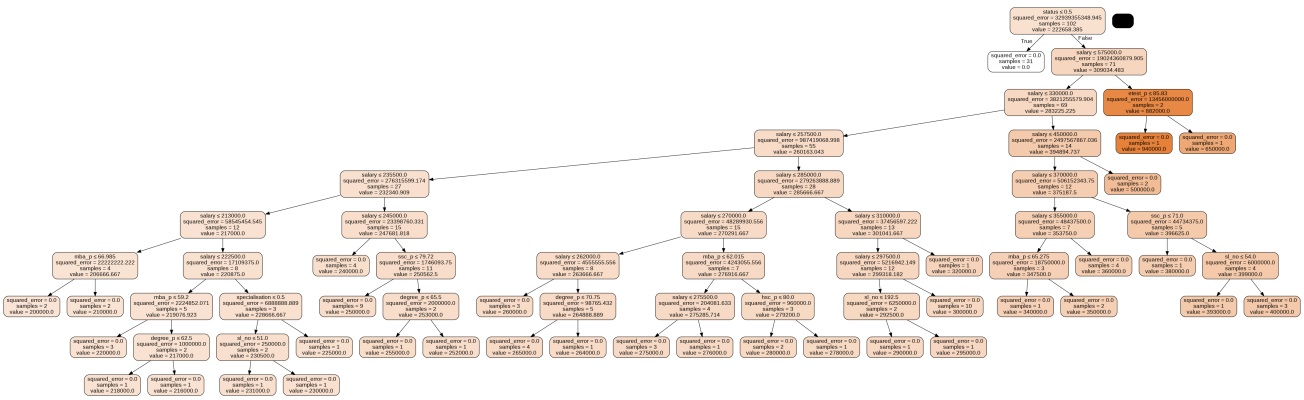
```
Image(get_png_tree(bagging_model.estimators_[1], data.columns))
```



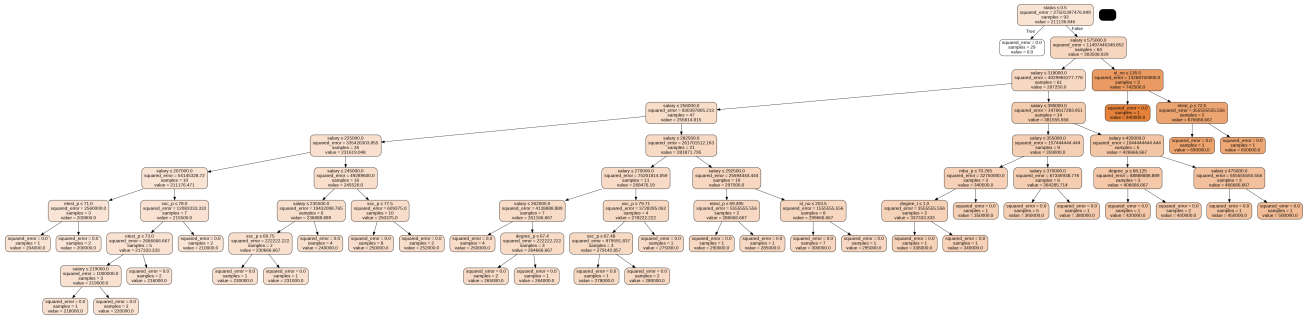
```
Image(get_png_tree(bagging_model.estimators_[2], data.columns))
```



```
Image(get_png_tree(bagging_model.estimators_[3], data.columns))
```



```
Image(get_png_tree(bagging_model.estimators_[4], data.columns))
```



Заметно, что деревья различны.

Визуализируем результаты регрессии:

```
from sklearn.tree import DecisionTreeRegressor


def make_meshgrid(x, y, h=.02):
    """Create a mesh of points to plot in

    Parameters
    ----------
    x: data to base x-axis meshgrid on
    y: data to base y-axis meshgrid on
    h: stepsize for meshgrid, optional

    Returns
    -------
    xx, yy : ndarray
    """
    x_min, x_max = x.min() - 1, x.max() + 1
    y_min, y_max = y.min() - 1, y.max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))
    return xx, yy

def plot_contours(ax, clf, xx, yy, **params):
    """Plot the decision boundaries for a classifier.

    Parameters
    ----------
```

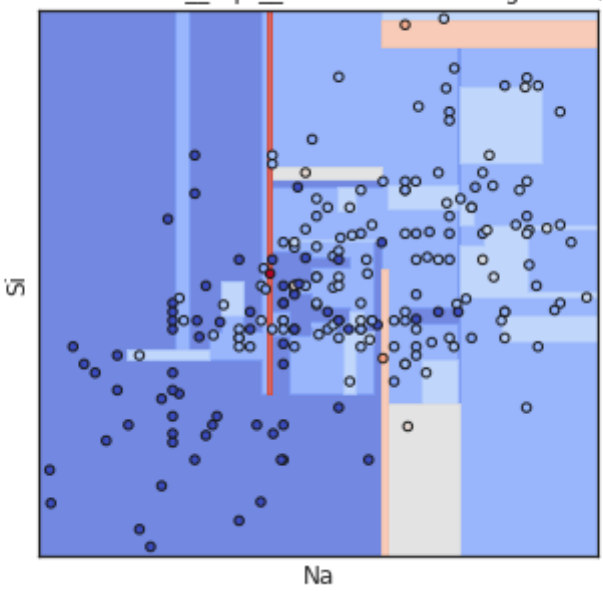
```
        ax: matplotlib axes object
        clf: a classifier
        xx: meshgrid ndarray
        yy: meshgrid ndarray
        params: dictionary of params to pass to contourf, optional
        """
        Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
        Z = Z.reshape(xx.shape)
        #Можно проверить все ли метки классов предсказываются
        #print(np.unique(Z))
        out = ax.contourf(xx, yy, Z, **params)
        return out


    def make_meshgrid(x, y, h=.02):
        """Create a mesh of points to plot in

        Parameters
        ----------
        x: data to base x-axis meshgrid on
        y: data to base y-axis meshgrid on
        h: stepsize for meshgrid, optional

        Returns
        -------
        xx, yy : ndarray
        """
        x_min, x_max = x.min() - 1, x.max() + 1
        y_min, y_max = y.min() - 1, y.max() + 1
        xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                             np.arange(y_min, y_max, h))
        return xx, yy

    def plot_contours(ax, clf, xx, yy, **params):
        """Plot the decision boundaries for a classifier.

        Parameters
        ----------
        ax: matplotlib axes object
        clf: a classifier
        xx: meshgrid ndarray
        yy: meshgrid ndarray
        params: dictionary of params to pass to contourf, optional
        """
        Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
        Z = Z.reshape(xx.shape)
        #Можно проверить все ли метки классов предсказываются
        #print(np.unique(Z))
        out = ax.contourf(xx, yy, Z, **params)
        return out


    def plot_cl(clf):
        title = clf.__repr__
        clf.fit(X2, y)
```

```
    fig, ax = plt.subplots(figsize=(5,5))
    X0, X1 = X2[:, 0], X2[:, 1]
    xx, yy = make_meshgrid(X0, X1)
    plot_contours(ax, clf, xx, yy, cmap=plt.cm.coolwarm, alpha=0.8)
    ax.scatter(X0, X1, c=y, cmap=plt.cm.coolwarm, s=20, edgecolors='k')
    ax.set_xlim(xx.min(), xx.max())
    ax.set_ylim(yy.min(), yy.max())
    ax.set_xlabel('Na')
    ax.set_ylabel('Si')
    ax.set_xticks(())
    ax.set_yticks(())
    ax.set_title(title)
    plt.show()
```

```
X2 = X[['ssc_p', 'hsc_p']].to_numpy()
```

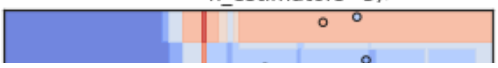
```
plot_cl(DecisionTreeRegressor(random_state=1))
```

```
<bound method BaseEstimator.__repr__ of DecisionTreeRegressor(random_state=1)>
```




```
plot_cl(BaggingRegressor(DecisionTreeRegressor(random_state=1), n_estimators=5))
```

```
<bound method BaseEstimator.__repr__ of BaggingRegressor(base_estimator=DecisionTreeRegressor(random_state=1),
                  n_estimators=5)>
```



Модель градиентного бустинга

```
from sklearn.ensemble import GradientBoostingRegressor
```

Обучим модель на 5 деревьях:

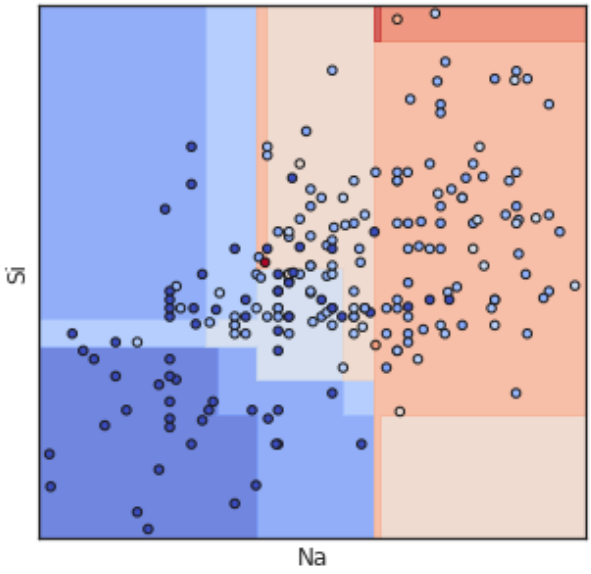
```
gradient_model = GradientBoostingRegressor(n_estimators=5)
gradient_model.fit(X_train, y_train)
```

```
GradientBoostingRegressor(n_estimators=5)
```

Для визуализации регрессии будем использовать функцию plot_cl из визуализации регрессии модели бэггинга:

```
plot_cl(GradientBoostingRegressor(random_state=1, n_estimators=5))
```

```
<bound method BaseEstimator.__repr__ of GradientBoostingRegressor(n_estimators=5, random_state=1)>
```



Модель стекинга

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
```

Реализуем модель стекинга через библиотеку heamy:

```
!pip install heamy
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
Collecting heamy
  Downloading heamy-0.0.7.tar.gz (30 kB)
Requirement already satisfied: scikit-learn>=0.17.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: pandas>=0.17.0 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: scipy>=0.16.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: numpy>=1.7.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dis
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages
Building wheels for collected packages: heamy
  Building wheel for heamy (setup.py) ... done
  Created wheel for heamy: filename=heamy-0.0.7-py2.py3-none-any.whl size=15366 sha25
  Stored in directory: /root/.cache/pip/wheels/f5/6c/da/55718ad26a9c8d3528b50edc2676f
Successfully built heamy
Installing collected packages: heamy
Successfully installed heamy-0.0.7
```

```
from heamy.estimator import Regressor
from heamy.pipeline import ModelsPipeline
from heamy.dataset import Dataset


dataset = Dataset(X_train, y_train, X_test)
```

Построим модели дерева, линейную модель и случайного леса для задачи регрессии:

```
model_tree = Regressor(dataset=dataset, estimator=DecisionTreeRegressor, name='tree')
model_lr = Regressor(dataset=dataset, estimator=LinearRegression, parameters={'normalize':
model_rf = Regressor(dataset=dataset, estimator=RandomForestRegressor, parameters={'n_esti
```

Определим их качество:

```
from sklearn.metrics import mean_absolute_error


def val_mae(model):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    result = mean_absolute_error(y_test, y_pred)
    print(model)
    print("MAE = {}".format(result))


for model in [
    LinearRegression(),
    DecisionTreeRegressor(),
    RandomForestRegressor(n_estimators=5)
]:
```

```
    val_mae(model)
    print()

     LinearRegression()
     MAE = 3.6458537766000704e-11

     DecisionTreeRegressor()
     MAE = 1314.8148148148148

     RandomForestRegressor(n_estimators=5)
     MAE = 1274.0740740740741
```

Оценка качества полученных моделей

Для оценки качества полученных моделей будем использовать метрику "Средняя абсолютная ошибка" (mean_absolute_error).

Чем ближе её значение к нулю, тем лучше качество регрессии.

Посчитаем метрику для всех моделей:

```
mean_absolute_error(y_test, bagging_model.predict(X_test))
```

```
1485.1851851851852
```

```
mean_absolute_error(y_test, gradient_model.predict(X_test))
```

```
66984.55795305151
```

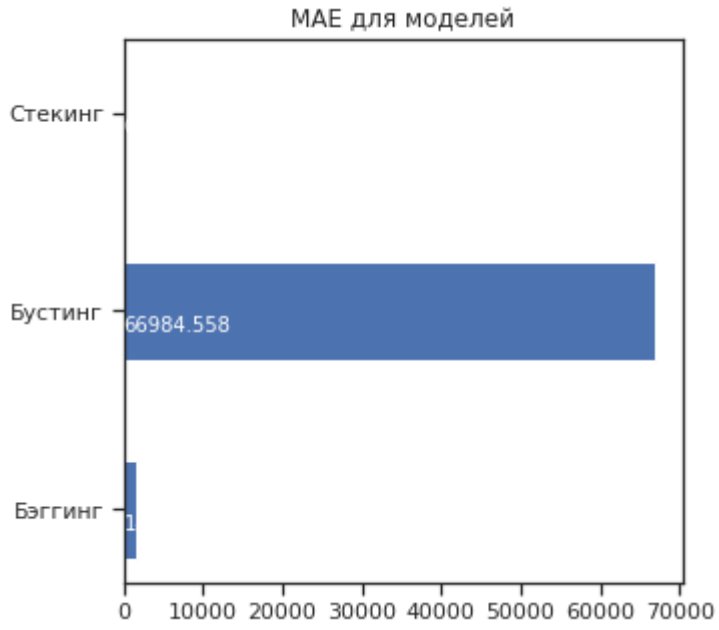
```
results = model_lr.validate(k=10, scorer=mean_absolute_error)
```

```
Metric: mean_absolute_error
Folds accuracy: [1.9548296938124766e-10, 1.8005682217953568e-10, 7.555599631955785e-1
Mean accuracy: 1.6042533553352403e-10
Standard Deviation: 5.2849837763945337e-11
Variance: 2.7931053516753428e-21
```

```
labels = ['Бэггинг','Бустинг', 'Стекинг']
mae = [1485.1851851851852, 66984.55795305151, 1.6042533553352403e-10]
```

```
def vis_models_quality(array_metric, array_labels, str_header, figsize=(5, 5)):
    fig, ax1 = plt.subplots(figsize=figsize)
    pos = np.arange(len(array_metric))
    rects = ax1.barh(pos, array_metric,
                     align='center',
                     height=0.5,
                     tick_label=array_labels)
    ax1.set_title(str_header)
    for a,b in zip(pos, array_metric):
        plt.text(0.2, a-0.1, str(round(b,2)), color='white')
```

```
            plt.text(0.2, a-0.1, str(round(b,3)), color= 'white' )
        plt.show()


vis_models_quality(mae, labels, 'MAE для моделей')
```



Самое лучшее качество регресии наблюдается у модели стекинга.

✓ 0 сек. выполнено в 17:24