Защищено: Гапанюк Ю.Е.

" " 2022 г.

Демонстрация: Гапанюк Ю.Е.

" " 2022 г.

**Отчет по лабораторной работе № 5 по курсу Технологии машинного обучения**

**ГУИМЦ**

**Тема работы: " Ансамбли моделей машинного обучения. "**

10

(количество листов) Вариант № **3**

ИСПОЛНИТЕЛЬ:

студент группы ИУ5Ц-84Б

(подпись)

Семенова А.А.

" " 2022 г.

Москва, МГТУ - 2022

# Цель лабораторной работы

Изучение ансамблей моделей машинного обучения

# Задание

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

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

#### С использованием метода train\_test\_split разделите выборку на обучающую и тестовую.

1. 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.

1. **(+1 балл на экзамене)** Дополнительно к указанным моделям обучите еще две модели:

#### Модель [многослойного персептрона.](https://scikit-learn.org/stable/modules/neural_networks_supervised.html) По желанию, вместо библиотеки scikit-learn возможно использование библиотек [TensorFlow](https://www.tensorflow.org/), [PyTorch](https://pytorch.org/) 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.

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

# Ход лабораторной работы

## Текстовое описание набора данных

В качестве набора данных используется dataset рейтингов университетов мира на основании трёх рейтингов. Датасет доступен по адресу: [https://www.kaggle.com/mylesoneill/world-university-ranking](https://www.kaggle.com/mylesoneill/world-university-rankings)s

Из набора данных будет рассматриваться только файл cwurData.csv . Описание столбцов:

world\_rank - мировой рейтинг университета

institution - название университета

country - страна, в которой расположен университет national\_rank - рейтинг университета в стране его нахождения quality\_of\_education - рейтинг качества образования

quality\_of\_faculty - рейтинг качества профессорско-преподавательского состава

publications - рейтинг публикаций

infuence - рейтинг влияния

citations - количество студентов в университете

#### broad\_impact - рейтинг за широкое влияние (предоставлен только за 2014 и 2015 гг. Остальное - пропуски)

patents - рейтинг за патенты

#### score - общий балл, используемый для определения мирового рейтинга

year - год рейтинга (с 2012 по 2015 год)

## Основные характеристики набора данных

Подключаем все необходимые библиотеки In [1]: **import** numpy **as** np

**import** pandas **as** pd

**import** seaborn **as** sns **import** matplotlib **import** matplotlib\_inline

**import** matplotlib.pyplot **as** plt

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

**%matplotlib** inline sns**.**set(style**=**"ticks") **from** io **import** StringIO

**from** IPython.display **import** Image

**import** graphviz

**import** pydotplus

**from** sklearn.metrics **import** mean\_absolute\_error

#### Подключаем Dataset

Inda[2t]a:

**=** pd**.**read\_csv('cwurData.csv', sep**=**",")

Размер набора данных In [3]: data**.**shape

Out[3]:(2200, 14)

#### Типы колонок

In [4]: data**.**dtypes

Out[4]:world\_rank int64

institution object

country object

national\_rank int64 quality\_of\_education int64 alumni\_employment int64

quality\_of\_faculty int64

publications int64

influence int64

citations int64

broad\_impact float64

patents int64

score float64

year int64

dtype: object

#### Проверяем, есть ли пропущенные значения

In [5]: data**.**isnull()**.**sum()

|  |  |
| --- | --- |
| Out[5]:world\_rank | 0 |
| institution | 0 |
| country | 0 |
| national\_rank | 0 |
| quality\_of\_education | 0 |
| alumni\_employment | 0 |
| quality\_of\_faculty | 0 |
| publications | 0 |
| influence | 0 |
| citations | 0 |
| broad\_impact | 200 |
| patents | 0 |
| score | 0 |
| year | 0 |
| dtype: int64 |  |

Первые 5 строк датасета In [6]: data**.**head()

Out[6]: **world\_rank institution**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **country** | **national\_rank** | **quality\_of\_education** | **alumni\_employment** | **quality\_of\_faculty** | **publications** | **influence** |
| USA | 1 | 7 | 9 | 1 | 1 | 1 |
| USA | 2 | 9 | 17 | 3 | 12 | 4 |
| USA | 3 | 17 | 11 | 5 | 4 | 2 |

* 1. 1 Harvard University

Massachusetts

* 1. 2 Institute of Technology
  2. 3 Stanford University
  3. 4 University of Cambridge

California

United Kingdom

1 10 24 4 16 16

* 1. 5 Institute of Technology

USA 4 2 29 7 37 22



In [7]: total\_count **=** data**.**shape[0]

print('Всего строк: {}'**.**format(total\_count))

Всего строк: 2200

Процент пропусков в broad\_impact

In [8]: (200 **/** 2200) **\*** 100

Out[8]:9.090909090909092

#### Настройка отображения графиков

In [9]: *# Задание формата графиков для сохранения высокого качества PNG* **from** IPython.display **import** set\_matplotlib\_formats matplotlib\_inline**.**backend\_inline**.**set\_matplotlib\_formats("retina")

*# Задание ширины графиков, чтобы они помещались на A4*

pd**.**set\_option("display.width", 70)

## Обработка пропусков данных

### Очистка строк

#### Можно очистить строки, содержащие пропуски. При этом останутся данные только за 2014 и 2015 гг (см. описание датасета)

In [10]: *# Удаление строк, содержащих пустые значения* data\_no\_null **=** data**.**dropna(axis**=**0, how**=**'any') (data**.**shape, data\_no\_null**.**shape)

Out[10]:((2200, 14), (2000, 14))

#### Выведем первые 11 строк, чтобы убедиться, что данные в national\_rank числовые (Jupyter Lab в предпросмотре CSV показывает не совсем верно)

In [11]: data\_no\_null**.**head(11)

**institution**

|  |  |  |
| --- | --- | --- |
| Out[11]: | **200** | **world\_rank**  1 |
|  | **201** | 2 |
|  | **202** | 3 |
|  | **203** | 4 |
|  | **204** | 5 |
|  | **205** | 6 |
|  | **206** | 7 |
|  | **207** | 8 |
|  | **208** | 9 |
|  | **209** | 10 |
|  | **210** | 11 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **country**  USA | **national\_rank**  1 | **quality\_of\_education**  1 | **alumni\_employment**  1 | **quality\_of\_faculty**  1 | **publications**  1 | **influenc** |
| USA | 2 | 11 | 2 | 4 | 5 |  |
| USA | 3 | 3 | 11 | 2 | 15 |  |
| United Kingdom | 1 | 2 | 10 | 5 | 10 |  |
| United | 2 | 7 | 12 | 10 | 11 |  |

Harvard University

Stanford University

Massachusetts Institute of Technology

University of Cambridge

University of

Oxford

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Kingdom  USA | 4 | 13 | 8 | 9 | 14 |
| USA | 5 | 4 | 22 | 6 | 7 |
| USA | 6 | 10 | 14 | 8 | 17 |
| USA | 7 | 5 | 16 | 3 | 70 |
| USA | 8 | 9 | 25 | 11 | 18 |
| USA | 9 | 12 | 18 | 19 | 23 |

Columbia University

University of California, berkeley

University of

Chicago

Princeton University

yale University

Cornell University



In [12]: total\_count **=** data\_no\_null**.**shape[0]

print('Всего строк: {}'**.**format(total\_count))

Всего строк: 2000

## Кодирование категориальных признаков

#### Преобразуем названия стран, городов, ... в числовые зеачения (label encoding)

In [13]: **from** sklearn.preprocessing **import** LabelEncoder, OneHotEncoder

In [14]: le **=** LabelEncoder()

*# "institution"* le**.**fit(data\_no\_null**.**institution**.**drop\_duplicates()) data\_no\_null**.**institution **=** le**.**transform(data\_no\_null**.**institution)

*# "country"* le**.**fit(data\_no\_null["country"]**.**drop\_duplicates()) data\_no\_null["country"] **=** le**.**transform(data\_no\_null["country"])

/tmp/ipykernel\_156/4210865855.py:4: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame. Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#re turning-a-view-versus-a-copy

data\_no\_null.institution = le.transform(data\_no\_null.institution)

/tmp/ipykernel\_156/4210865855.py:7: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame. Try using .loc[row\_indexer,col\_indexer] = value instead

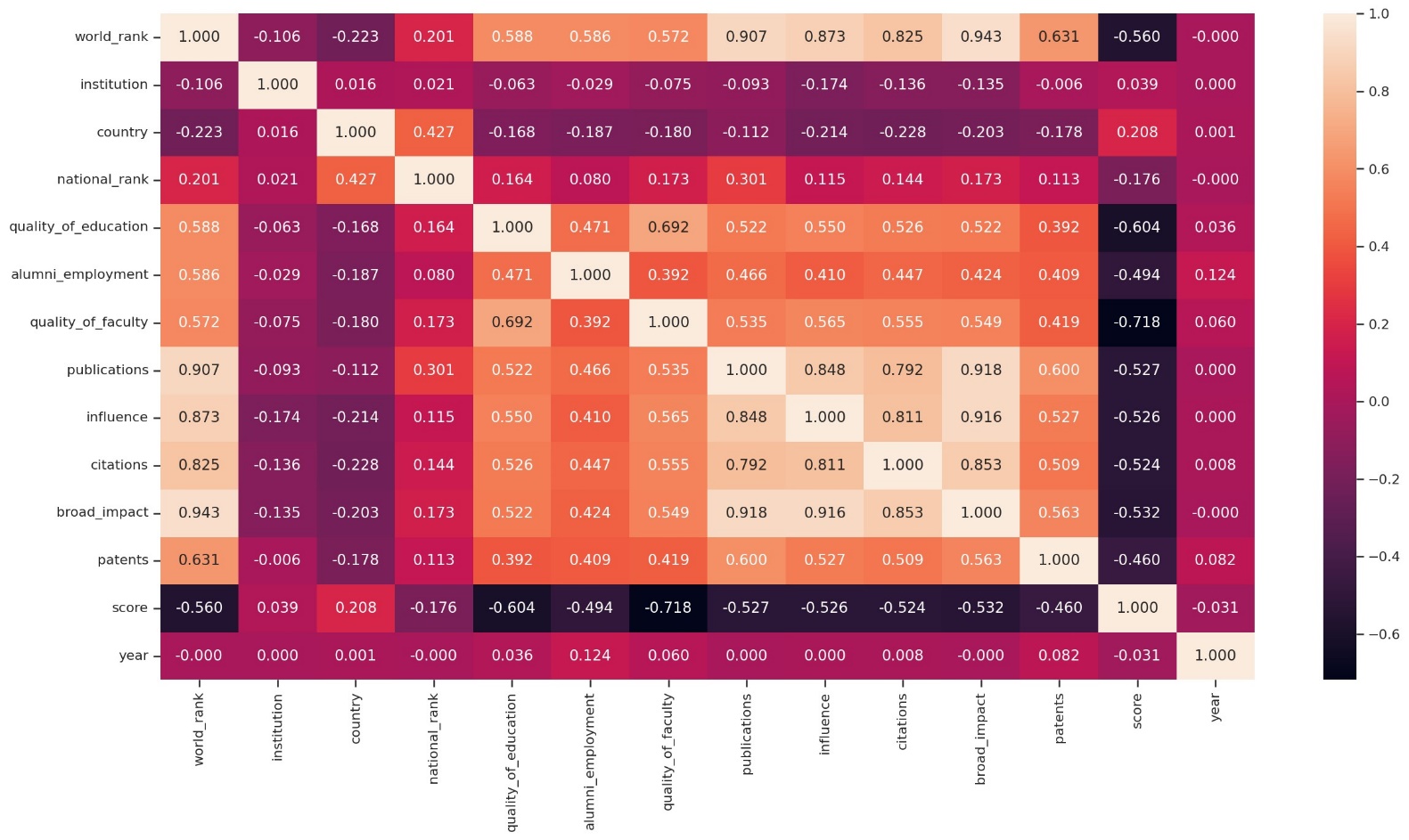
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#re turning-a-view-versus-a-copy

data\_no\_null["country"] = le.transform(data\_no\_null["country"])

#### Построим кореляционную матрицу

In [15]: ig, ax **=** plt**.**subplots(figsize**=**(20,10)) sns**.**heatmap(data\_no\_null**.**corr(method**=**'pearson'), ax**=**ax, annot**=True**, fmt**=**'.3f')

Out[15]:<AxesSubplot:>



|  |
| --- |
|  |
|  |
|  |

## Предсказание целевого признака

Предскажем значение целевого признака world\_rank по broad\_impact , influence и publications , поскольку их значения кореляции ближе всего к 1

### Разбиение выборки на обучающую и тестовую

In [16]: X **=** data\_no\_null[["broad\_impact", "publications", "influence"]] Y **=** data\_no\_null["world\_rank"]

print('Входные данные:\n\n', X**.**head(), '\n\nВыходные данные:\n\n', Y**.**head())

Входные данные:

broad\_impact publications influence

|  |  |  |  |
| --- | --- | --- | --- |
| 200 | 1.0 | 1 | 1 |
| 201 | 4.0 | 5 | 3 |
| 202 | 2.0 | 15 | 2 |
| 203 | 13.0 | 10 | 9 |
| 204 | 12.0 | 11 | 12 |

Выходные данные:

200 1

201 2

202 3

203 4

204 5

Name: world\_rank, dtype: int64

#### Разделим выборку на обучающую и тестовую

In [17]: X\_train, X\_test, Y\_train, Y\_test **=** train\_test\_split(X, Y, random\_state **=** 2022, test\_size **=** 0.1)

Входные параметры обучающей выборки In [18]: X\_train**.**head()

|  |  |  |  |
| --- | --- | --- | --- |
| Out[18]: | **broad\_impact** | **publications** | **influence** |
|  | **2164** 932.0 | 875 | 832 |
|  | **1710** 590.0 | 576 | 792 |
|  | **428** 164.0 | 200 | 149 |
|  | **1389** 164.0 | 233 | 251 |
|  | **2089** 932.0 | 675 | 775 |

Входные параметры тестовой выборки In [19]: X\_test**.**head()

|  |  |  |  |
| --- | --- | --- | --- |
| Out[19]: | **broad\_impact** | **publications** | **influence** |
|  | **1218** 14.0 | 3 | 20 |
|  | **1495** 265.0 | 236 | 235 |
|  | **843** 703.0 | 943 | 599 |
|  | **2042** 850.0 | 803 | 933 |
|  | **1869** 606.0 | 701 | 658 |

Выходные параметры обучающей выборки In [20]: Y\_train**.**head()

Out[20]:2164 965

|  |  |
| --- | --- |
| 1710 | 511 |
| 428 | 229 |
| 1389 | 190 |
| 2089 | 890 |

Name: world\_rank, dtype: int64

#### Выходные параметры тестовой выборки

In [21]: Y\_test**.**head()

Out[21]:1218 19

|  |  |
| --- | --- |
| 1495 | 296 |
| 843 | 644 |
| 2042 | 843 |
| 1869 | 670 |

Name: world\_rank, dtype: int64

### Построение модели сверхслучайных деревьев (группа бэггинга)

In [22]: *# Визуализация дерева*

**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()

#### Обучим регрессор на 3 деревьях

In [23]: **from** sklearn.ensemble **import** RandomForestRegressor, ExtraTreesRegressor

**from** sklearn.tree **import** DecisionTreeClassifier, DecisionTreeRegressor, export\_graphviz

In [24]: tree1 **=** RandomForestRegressor(n\_estimators**=**5, oob\_score**=True**, random\_state**=**2022) tree1**.**fit(X, Y)

/home/alexandr/.local/lib/python3.10/site-packages/sklearn/ensemble/\_forest.py:560: UserWarning: Some inputs d o not have OOB scores. This probably means too few trees were used to compute any reliable OOB estimates.

warn(

Out[24]:RandomForestRegressor(n\_estimators=5, oob\_score=True, random\_state=2022)

#### Out-of-bag error, возвращаемый регрессором

In [25]: tree1**.**oob\_score\_, 1**-**tree1**.**oob\_score\_

Out[25]:(0.4357507476207476, 0.5642492523792524)

In [26]: tree1**.**oob\_prediction\_[55:70]

Out[26]:array([ 0. , 18.33333333, 52. , 61.5 ,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 83. | , 22. | , 100.5 | , 72. | , |
| 157. | , 188.75 | , 75. | , 0. | , |
| 36. | , 77.5 | , 68. | ]) |  |

#### Выведем полученные деревья

In [27]: Image(get\_png\_tree(tree1**.**estimators\_[0], X**.**columns[:3]), width**=**"100%") dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.310891 to fit

#### Out[27]:



In [28]: Image(get\_png\_tree(tree1**.**estimators\_[1], X**.**columns[:3]), width**=**"100%") dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.318068 to fit

#### Out[28]:



In [29]: Image(get\_png\_tree(tree1**.**estimators\_[2], X**.**columns[:3]), width**=**"100%") dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.293917 to fit

#### Out[29]:



Посмотрим важность признаков в каждом из деревьев In [30]: **from** operator **import** itemgetter

**def** draw\_feature\_importances(tree\_model, X\_dataset, figsize**=**(10,5)): """

Вывод важности признаков в виде графика """

*# Сортировка значений важности признаков по убыванию*

list\_to\_sort **=** list(zip(X\_dataset**.**columns**.**values, tree\_model**.**feature\_importances\_)) sorted\_list **=** sorted(list\_to\_sort, key**=**itemgetter(1), reverse **= True**)

*# Названия признаков*

labels **=** [x **for** x,\_ **in** sorted\_list]

*# Важности признаков*

data **=** [x **for** \_,x **in** sorted\_list]

*# Вывод графика*

fig, ax **=** plt**.**subplots(figsize**=**figsize) ind **=** np**.**arange(len(labels)) plt**.**bar(ind, data)

plt**.**xticks(ind, labels, rotation**=**'vertical')

*# Вывод значений*

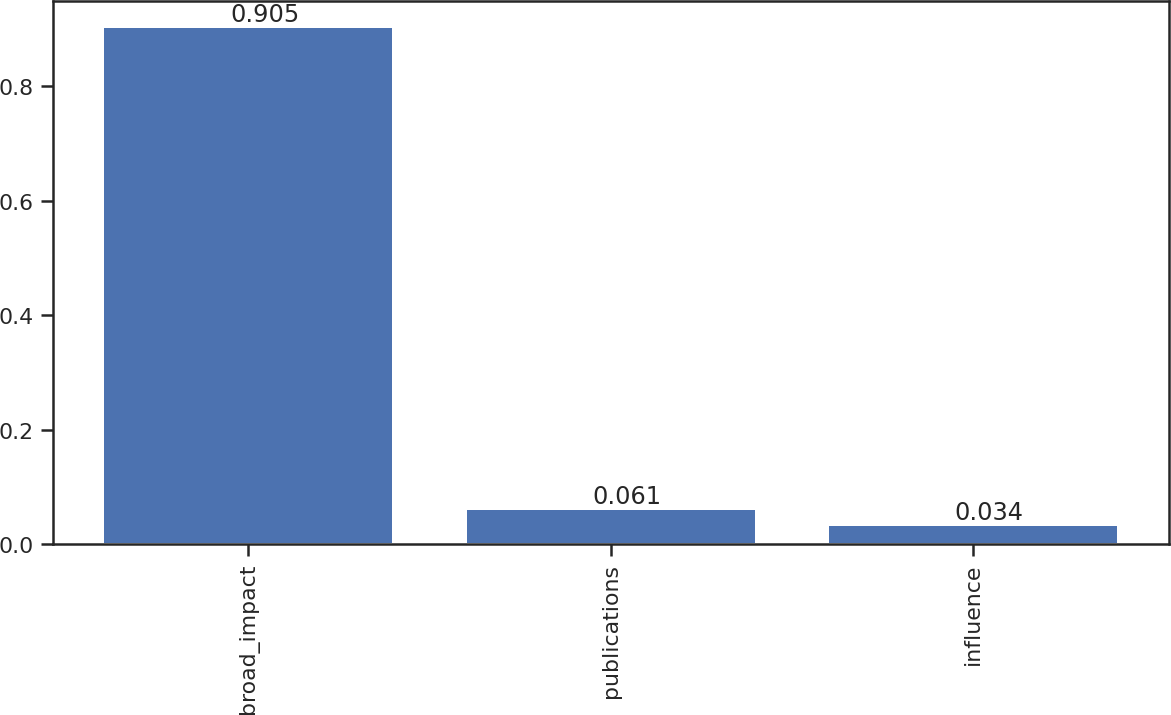
**for** a,b **in** zip(ind, data):

plt**.**text(a**-**0.05, b**+**0.01, str(round(b,3))) plt**.**show()

**return** labels, data

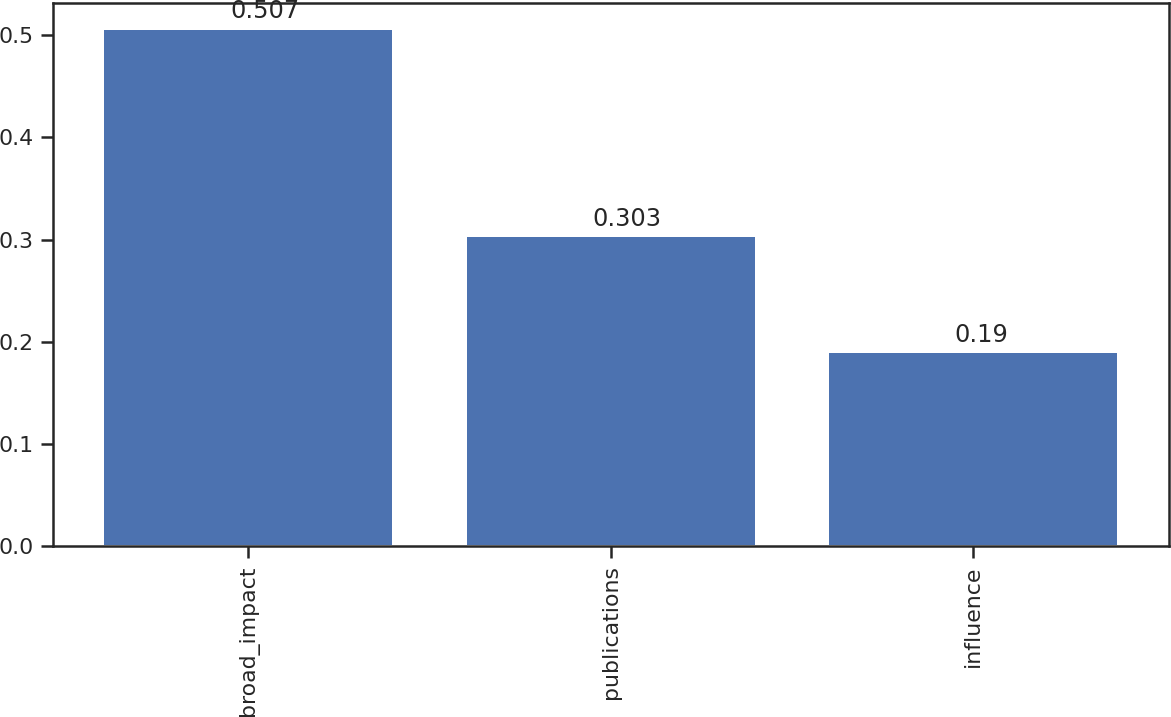
In [31]: data\_rf\_reg **=** RandomForestRegressor(random\_state**=**2022) data\_rf\_reg**.**fit(X, Y)

\_,\_ **=** draw\_feature\_importances(data\_rf\_reg, X)



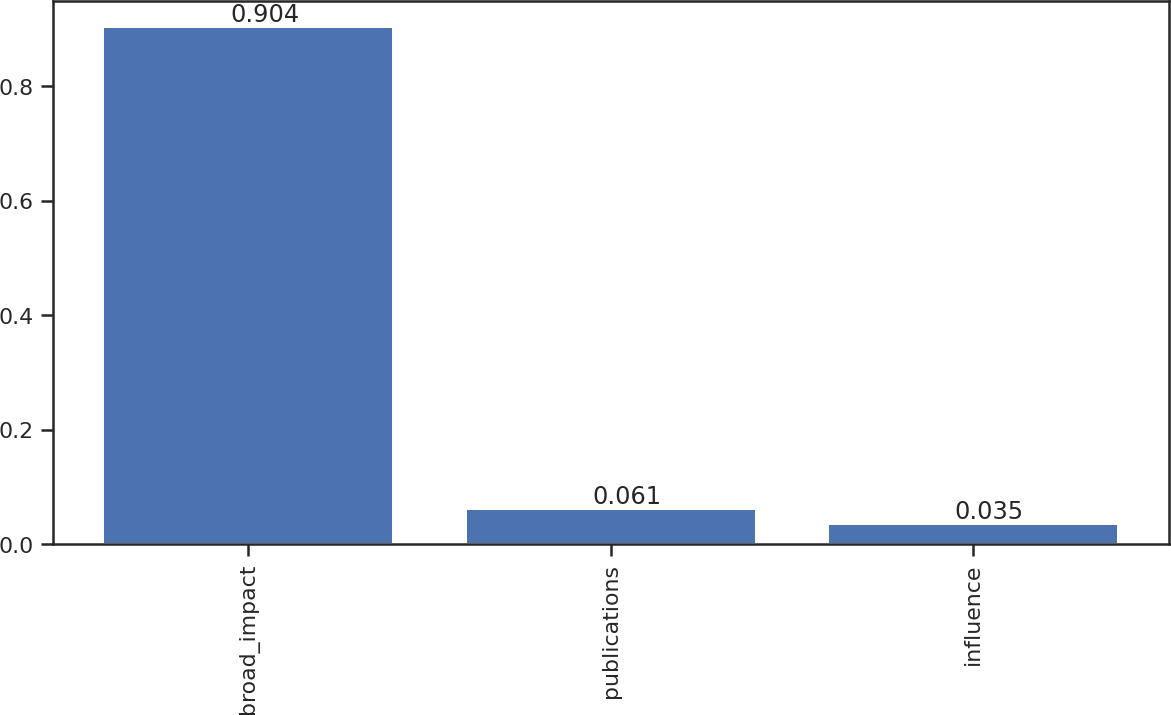
|  |
| --- |
|  |
|  |
|  |

In [32]: data\_xtree\_reg **=** ExtraTreesRegressor(random\_state**=**2022) data\_xtree\_reg**.**fit(X, Y)

\_, \_ **=** draw\_feature\_importances(data\_xtree\_reg, X)

|  |
| --- |
|  |
|  |
|  |

In [33]: data\_tree\_reg **=** DecisionTreeRegressor(random\_state**=**2022) data\_tree\_reg**.**fit(X, Y)

\_,\_ **=** draw\_feature\_importances(data\_tree\_reg, X)

|  |
| --- |
|  |
|  |
|  |

In [49]: y\_pred1 **=** tree1**.**predict(X\_test)

In [50]: print('Средняя абсолютная ошибка:', mean\_absolute\_error(Y\_test, y\_pred1)) Средняя абсолютная ошибка: 24.033999999999995

### AdaBoost (бустинг)

In [35]: **from** sklearn.ensemble **import** AdaBoostRegressor

#### Обучим регрессор на 3 деревьях

In [36]: ab1 **=** AdaBoostRegressor(n\_estimators**=**3, random\_state**=**2022) ab1**.**fit(X, Y)

Out[36]:AdaBoostRegressor(n\_estimators=3, random\_state=2022)

#### Выведем используемые деревья

In [37]: Image(get\_png\_tree(ab1**.**estimators\_[0], X**.**columns[:3]), width**=**'40%')

#### Out[37]:

|  |
| --- |
|  |
|  |
|  |

In [38]: Image(get\_png\_tree(ab1**.**estimators\_[1], X**.**columns[:3]), width**=**'40%')

#### Out[38]:

|  |
| --- |
|  |
|  |
|  |

In [39]: Image(get\_png\_tree(ab1**.**estimators\_[2], X**.**columns[:3]), width**=**'40%')

#### Out[39]:

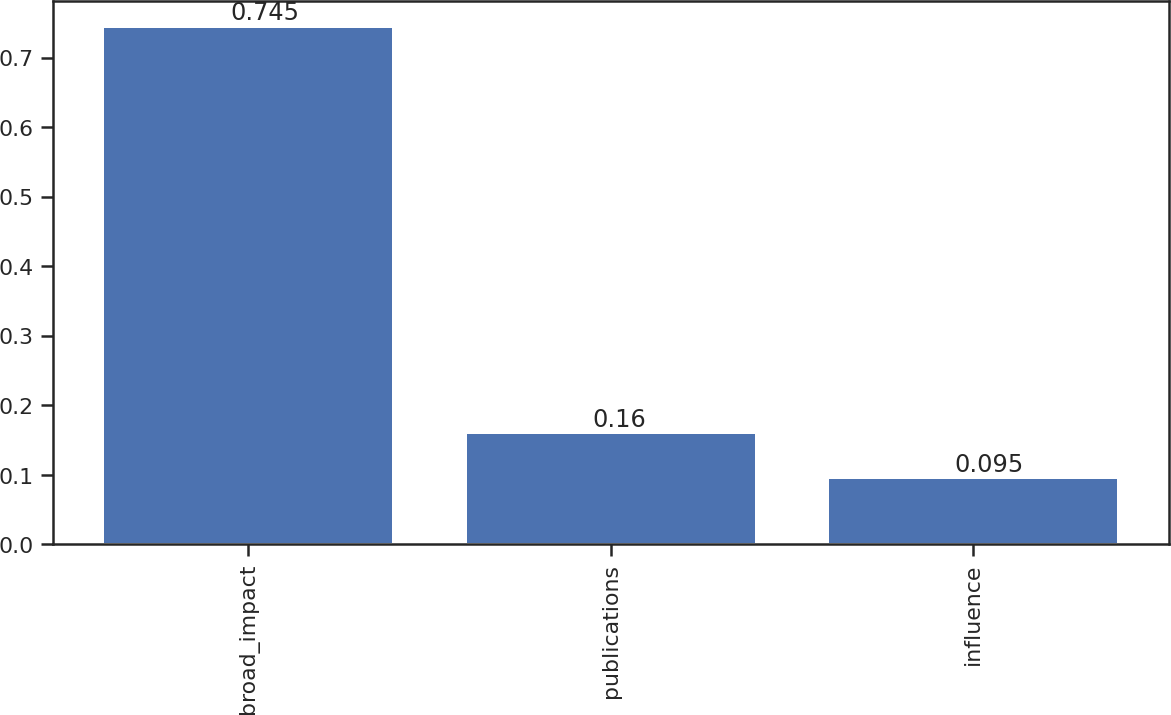
|  |
| --- |
|  |
|  |
|  |

Получим веса деревьев в ансамбле In [40]: ab1**.**estimator\_weights\_

Out[40]:array([2.236239 , 1.77461772, 1.20861837])

#### Проверим важность признаков в модели

In [41]: ab2 **=** AdaBoostRegressor(random\_state**=**2022) ab2**.**fit(X, Y)

\_,\_ **=** draw\_feature\_importances(ab2, X)

|  |
| --- |
|  |
|  |
|  |

In [52]: y\_pred2 **=** ab1**.**predict(X\_test)

In [53]: print('Средняя абсолютная ошибка:', mean\_absolute\_error(Y\_test, y\_pred2)) Средняя абсолютная ошибка: 68.50896702287369

### Стекинг

In [42]: **from** heamy.estimator **import** Regressor, Classifier

**from** heamy.pipeline **import** ModelsPipeline

**from** heamy.dataset **import** Dataset

In [43]: *# Качество отдельных моделей*

**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))

#### Проверим точность на отдельных моделях

In [44]: **from** sklearn.linear\_model **import** LinearRegression

In [45]: **for** model **in** [

LinearRegression(), DecisionTreeRegressor(), RandomForestRegressor(n\_estimators**=**50)

]:

val\_mae(model) print('==========================')

print()

LinearRegression() MAE=55.13824819458175

==========================

DecisionTreeRegressor() MAE=71.065

==========================

RandomForestRegressor(n\_estimators=50) MAE=60.00380000000001

==========================

In [46]: dataset **=** Dataset(X\_train, Y\_train, X\_test)

#### С использованием библиотеки heamy сделаем стек из трёх уровней

In [47]: *# Первый уровень - две модели: дерево и линейная регрессия*

model\_tree **=** Regressor(dataset**=**dataset, estimator**=**DecisionTreeRegressor, name**=**'tree')

model\_lr **=** Regressor(dataset**=**dataset, estimator**=**LinearRegression, parameters**=**{'normalize': **True**},name**=**'l model\_rf **=** Regressor(dataset**=**dataset, estimator**=**RandomForestRegressor, parameters**=**{'n\_estimators': 50},n

In [48]: pipeline **=** ModelsPipeline(model\_tree, model\_lr) stack\_ds **=** pipeline**.**stack(k**=**3, seed**=**1)

*# модель второго уровня - линейная регрессия*

stacker **=** Regressor(dataset**=**stack\_ds, estimator**=**LinearRegression) results **=** stacker**.**validate(k**=**3,scorer**=**mean\_absolute\_error)

Metric: mean\_absolute\_error

Folds accuracy: [55.89716660590352, 62.120348985954415, 61.202675679339954]

Mean accuracy: 59.74006375706597

Standard Deviation: 2.743042709623444

Variance: 7.524283306818326

## Выводы

#### Лучшей оказалась модель случайного леса со средней абсолютной ошибкой 24% In [ ]: