Отчёт по ЛР №1 по ТМО

Водка Игорь, ИУ5-61

1) Текстовое описание набора данных

В качестве набора данных возьмём набор World University Rankings с Kaggle: https://www.kaggle.com/mylesoneill/world-university-rankings/home (https://www.kaggle.com/mylesoneill/world-university-rankings/home) Мне интересна эта тема.

Датасет состоит лишь из одного файла: `timesData.csv` , который нам предстоит разделить на обучающую и тестовую выборки. В выборке содержится множество колонок:

- world_rank - рейтинг (или вилка)
- university_name - название университета
- country - страна
- teaching - рейтинг качества преподавания
- international - рейтинг международного признания
- research - рейтинг исследований
- citations - industry income (knowledge transfer)
- income - доход
- total_score - общий счёт в рейтинге (целевой признак!)
- num_students - число студентов
- student_staff_ratio - кол-во студентов на одного преподавателя
- international_students - процент студентов из других стран
- female_male_ratio - отношение количества студентов мужского пола к женскому
- year - бесполезно

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

Делим выборку на обучающую и тестовую

In [2]:
```
from sklearn.model_selection import train_test_split

# Будем анализировать данные только на обучающей выборке
data = pd.read_csv('timesData.csv', sep=",", thousands=',')
# train, test = train_test_split(df, test_size=0.2)
```

2) Основные характеристики датасета

Так, давайте посмотрим...

In [3]:
```
# Первые 5 строк датасета
data.head()
```

Out[3]:

	world_rank	university_name	country	teaching	international	research	citations	income	total_score	num_students
0	1	Harvard University	United States of America	99.7	72.4	98.7	98.8	34.5	96.1	20152.0
1	2	California Institute of Technology	United States of America	97.7	54.6	98.0	99.9	83.7	96.0	2243.0
2	3	Massachusetts Institute of Technology	United States of America	97.8	82.3	91.4	99.9	87.5	95.6	11074.0
3	4	Stanford University	United States of America	98.3	29.5	98.1	99.2	64.3	94.3	15596.0
4	5	Princeton University	United States of America	90.9	70.3	95.4	99.9	-	94.2	7929.0

In [4]:
```
# Размер датасета
data.shape
```

Out[4]:
```
(2603, 14)
```

In [5]:
```
total_count = data.shape[0]
print('Всего строк: {}'.format(total_count))
```

```
Всего строк: 2603
```

In [6]:
```
# Список колонок
data.columns
```

Out[6]:
```
Index(['world_rank', 'university_name', 'country', 'teaching', 'international',
       'research', 'citations', 'income', 'total_score', 'num_students',
       'student_staff_ratio', 'international_students', 'female_male_ratio',
       'year'],
      dtype='object')
```

In [7]:
```
# Проверим наличие пустых значений
# Цикл по колонкам датасета
for col in data.columns:
    # Количество пустых значений - все значения заполнены
    temp_null_count = data[data[col].isnull()].shape[0]
    print('{} - {}'.format(col, temp_null_count))
```

```
world_rank - 0
university_name - 0
country - 0
teaching - 0
international - 0
research - 0
citations - 0
income - 0
total_score - 0
num_students - 59
student_staff_ratio - 59
international_students - 67
female_male_ratio - 233
year - 0
```

In [8]:
```
# Многовато... Давайте пока выкинем NaN просто.

data = data.fillna(data.mean())
data = data.dropna()

# Немного уменьшилась наша выборка... ну и ладно.
data.shape
```

Out[8]: (2362, 14)

In [9]:
```
# Интерполируем international, research, citations, income, total_score

def adjust_total_score(df_line):
    cols = ['international', 'research', 'citations', 'income', 'total_score']

    for col in cols:
        if df_line[col] == '-':
            # get all non-hyphen similar rating cols
            non_empty_cols = filter(lambda x: x != '-', map(lambda colName: df_line[colN
ame], cols))
            # floatify
            non_empty_cols = list(map(lambda s: float(s), non_empty_cols))
            # interpolate
            df_line[col] = sum(non_empty_cols) / len(non_empty_cols)

    return df_line

fixed_data = data.apply(adjust_total_score, axis=1)
```

In [10]:
```
fixed_data
```

Out[10]:

	world_rank	university_name	country	teaching	international	research	citations	income	total_score	num_stu
1	2	California Institute of Technology	United States of America	97.7	54.6	98.0	99.9	83.7	96.0	[illegible]
2	3	Massachusetts Institute of Technology	United States of America	97.8	82.3	91.4	99.9	87.5	95.6	1
3	4	Stanford University	United States of America	98.3	29.5	98.1	99.2	64.3	94.3	1
4	5	Princeton University	United States of America	90.9	70.3	95.4	99.9	89.95	94.2	[illegible]
5	6	University of Cambridge	United Kingdom	90.5	77.7	94.1	94.0	57.0	91.2	1
6	6	University of Oxford	United Kingdom	88.2	77.2	93.9	95.1	73.5	91.2	1
7	8	University of California, Berkeley	United States of America	84.2	39.6	99.3	97.8	81.95	91.1	3
8	9	Imperial College London	United Kingdom	89.2	90.0	94.5	88.3	92.9	90.6	1
9	10	Yale University	United States of America	92.1	59.2	89.7	91.5	82.475	89.5	1
10	11	University of California, Los Angeles	United States of America	83.0	48.1	92.9	93.2	80.475	87.7	3
11	12	University of Chicago	United States of America	79.1	62.8	87.9	96.9	83.625	86.9	1
12	13	Johns Hopkins University	United States of America	80.9	58.5	89.2	92.3	100.0	86.4	1
13	14	Cornell University	United States of America	82.2	62.4	88.8	88.1	34.7	83.9	2
14	15	ETH Zurich – Swiss Federal Institute of Techno...	Switzerland	77.5	93.7	87.8	83.1	87	83.4	1
15	15	University of Michigan	United States of America	83.9	53.3	89.1	84.1	59.6	83.4	4
18	19	University of Pennsylvania	United States of America	71.8	32.9	82.7	93.6	43.7	79.5	2
19	20	Carnegie Mellon University	United States of America	70.3	39.1	79.3	95.7	53.7	79.3	1
20	21	University of Hong Kong	Hong Kong	68.4	91.4	71.4	96.1	56.5	79.2	1
21	22	University College London	United Kingdom	74.0	90.8	81.6	80.6	39.0	78.4	2
22	23	University of Washington	United States of America	68.2	49.0	77.1	95.9	32.8	78.0	4
23	24	Duke University	United States of America	66.8	49.4	71.5	92.3	100.0	76.5	1
24	25	Northwestern University	United States of America	64.5	60.5	68.8	95.3	75.125	75.9	1
26	27	Georgia Institute of Technology	United States of America	67.9	73.2	72.6	83.2	95.1	75.3	1
27	28	Pohang University of Science and Technology	South Korea	69.5	32.6	62.5	96.5	100.0	75.1	[illegible]

	world_rank	university_name	country	teaching	international	research	citations	income	total_score	num_stu
28	29	University of California, Santa Barbara	United States of America	56.6	64.3	68.0	98.8	89.8	75.0	2
29	30	University of British Columbia	Canada	65.1	93.3	74.8	80.3	42.6	73.8	5
30	30	University of North Carolina at Chapel Hill	United States of America	70.9	21.5	75.1	85.0	50.2	73.8	2
31	32	University of California, San Diego	United States of America	59.8	31.6	76.3	90.8	51.8	73.2	2
32	33	University of Illinois at Urbana-Champaign	United States of America	68.1	55.9	80.9	72.9	70.675	73.0	4
33	34	National University of Singapore	Singapore	65.5	97.8	72.6	78.7	40.5	72.9	3
...	...	...	...	...	...	...	...	...	...	
2572	601-800	Technical University of Madrid	Spain	21.8	39.5	14.6	24.5	38.3	29.225	4
2573	601-800	University of Tehran	Iran	26.1	16.5	16.9	15.8	16.4	16.4	5
2574	601-800	University of Texas at El Paso	United States of America	18.6	30.4	18.7	18.4	22.5	22.5	1
2575	601-800	Texas Tech University	United States of America	27.9	36.8	17.2	22.0	25.3333	25.3333	2
2576	601-800	Tokai University	Japan	17.9	19.3	7.6	15.3	34.4	19.15	2
2577	601-800	Tokushima University	Japan	25.3	16.8	21.6	12.8	59.6	27.7	[illegible]
2578	601-800	Tokyo University of Marine Science and Technology	Japan	27.9	24.5	12.4	7.7	57.9	25.625	[illegible]
2579	601-800	Tokyo University of Science	Japan	23.0	15.4	24.1	21.4	37.6	24.625	2
2580	601-800	Tomsk State University	Russian Federation	34.8	36.9	20.8	7.6	44.0	27.325	1
2581	601-800	Tottori University	Japan	24.3	16.7	10.1	9.6	34.5	17.725	[illegible]
2582	601-800	Toyohashi University of Technology	Japan	22.0	25.4	18.9	15.8	50.3	27.6	[illegible]
2583	601-800	Universiti Kebangsaan Malaysia	Malaysia	24.3	29.7	15.9	10.9	28.4	21.225	2
2584	601-800	Universiti Putra Malaysia	Malaysia	25.3	50.1	20.9	10.2	34.2	28.85	2
2585	601-800	Universiti Sains Malaysia	Malaysia	26.9	44.2	16.6	12.4	34.4	26.9	2
2586	601-800	Universiti Teknologi MARA	Malaysia	15.2	14.8	7.7	18.2	28.3	17.25	6
2587	601-800	Ural Federal University	Russian Federation	24.8	17.3	10.6	16.8	35.6	20.075	2
2588	601-800	V.N. Karazin Kharkiv National University	Ukraine	21.7	48.4	8.9	1.7	28.8	21.95	1
2589	601-800	University of Vigo	Spain	18.4	30.7	10.5	31.8	38.1	27.775	2
2590	601-800	Vilnius University	Lithuania	18.3	40.8	13.6	26.1	41.0	30.375	1
2591	601-800	Warsaw University of Technology	Poland	19.4	20.7	8.5	40.3	47.4	29.225	3
2592	601-800	Waseda University	Japan	23.6	29.7	14.6	29.4	32.4	26.525	5

	world_rank	university_name	country	teaching	international	research	citations	income	total_score	num_stu
2593	601-800	University of West Bohemia	Czech Republic	16.3	23.1	9.7	29.8	32.1	23.675	1
2594	601-800	University of the West of England	United Kingdom	16.9	48.5	11.2	34.6	28.5	30.7	2
2595	601-800	West University of Timişoara	Romania	16.1	21.0	3.9	22.4	15.7667	15.7667	1
2596	601-800	University of Westminster	United Kingdom	17.3	81.9	11.7	21.1	28.5	35.8	1
2597	601-800	Xidian University	China	17.9	12.8	12.1	8.9	83.7	29.375	3
2598	601-800	Yeungnam University	South Korea	18.6	24.3	10.9	26.5	35.4	24.275	2
2599	601-800	Yıldız Technical University	Turkey	14.5	14.9	7.6	19.3	44.0	21.45	3
2601	601-800	Yokohama National University	Japan	20.1	23.3	16.0	13.5	40.4	23.3	1
2602	601-800	Yuan Ze University	Taiwan	16.2	17.7	18.3	28.6	39.8	26.1	

2362 rows × 14 columns

In [11]:
```
# Основные статистические характеристки набора данных
fixed_data.describe()
```

Out[11]:

	teaching	research	citations	num_students	student_staff_ratio	year
count	2362.000000	2362.000000	2362.000000	2362.000000	2362.000000	2362.000000
mean	37.146190	35.310288	61.004953	23845.077053	18.707282	2014.092295
std	17.145579	20.876934	23.091455	18008.833624	11.530523	1.682795
min	9.900000	2.900000	1.200000	462.000000	0.600000	2011.000000
25%	24.500000	19.400000	45.500000	12551.000000	12.200000	2013.000000
50%	33.100000	30.100000	62.700000	20584.000000	16.300000	2014.000000
75%	45.700000	46.200000	79.200000	30333.000000	21.900000	2016.000000
max	98.300000	99.400000	100.000000	379231.000000	162.600000	2016.000000

In [12]:
```
# Определим уникальные значения для целевого признака
data['world_rank'].unique()
```

Out[12]:
```
array(['2', '3', '4', '5', '6', '8', '9', '10', '11', '12', '13', '14',
       '15', '19', '20', '21', '22', '23', '24', '25', '27', '28', '29',
       '30', '32', '33', '34', '35', '36', '39', '40', '42', '43', '47',
       '48', '49', '51', '52', '53', '54', '56', '58', '59', '60', '61',
       '63', '64', '65', '66', '67', '68', '71', '72', '73', '75', '76',
       '77', '78', '79', '81', '83', '85', '87', '88', '90', '93', '95',
       '98', '99', '100', '101', '102', '103', '104', '105', '106', '107',
       '109', '111', '112', '114', '115', '117', '118', '119', '120',
       '122', '124', '127', '128', '129', '130', '132', '135', '137',
       '138', '139', '140', '142', '143', '144', '145', '147', '149',
       '151', '152', '155', '156', '159', '161', '163', '164', '165',
       '167', '168', '170', '171', '172', '173', '174', '177', '178',
       '181', '182', '183', '184', '185', '186', '187', '189', '190',
       '193', '195', '196', '197', '199', '1', '7', '16', '17', '18',
       '26', '31', '37', '38', '44', '45', '46', '55', '57', '69', '70',
       '74', '84', '86', '89', '91', '92', '94', '96', '97', '108', '110',
       '113', '121', '123', '125', '131', '133', '134', '141', '146',
       '148', '150', '154', '157', '162', '169', '176', '180', '191',
       '194', '200', '201-225', '226-250', '251-275', '276-300',
       '301-350', '350-400', '41', '50', '62', '80', '116', '153', '158',
       '166', '192', '198', '351-400', '126', '136', '160', '188', '82',
       '175', '=39', '=44', '=47', '=56', '=60', '=65', '=76', '=82',
       '=88', '=90', '=94', '=99', '=101', '=104', '=106', '=110', '=113',
       '=120', '=123', '=125', '=127', '=131', '=133', '=138', '=144',
       '=149', '=158', '=161', '=164', '=167', '=172', '=176', '179',
       '=180', '=182', '=185', '=190', '=193', '=196', '201-250',
       '251-300', '401-500', '501-600', '601-800'], dtype=object)
```

In [13]:
```
fixed_data
```

Out[13]:

	world_rank	university_name	country	teaching	international	research	citations	income	total_score	num_stu
1	2	California Institute of Technology	United States of America	97.7	54.6	98.0	99.9	83.7	96.0	2
2	3	Massachusetts Institute of Technology	United States of America	97.8	82.3	91.4	99.9	87.5	95.6	1
3	4	Stanford University	United States of America	98.3	29.5	98.1	99.2	64.3	94.3	1
4	5	Princeton University	United States of America	90.9	70.3	95.4	99.9	89.95	94.2	7
5	6	University of Cambridge	United Kingdom	90.5	77.7	94.1	94.0	57.0	91.2	1
6	6	University of Oxford	United Kingdom	88.2	77.2	93.9	95.1	73.5	91.2	1
7	8	University of California, Berkeley	United States of America	84.2	39.6	99.3	97.8	81.95	91.1	3
8	9	Imperial College London	United Kingdom	89.2	90.0	94.5	88.3	92.9	90.6	1
9	10	Yale University	United States of America	92.1	59.2	89.7	91.5	82.475	89.5	1
10	11	University of California, Los Angeles	United States of America	83.0	48.1	92.9	93.2	80.475	87.7	3
11	12	University of Chicago	United States of America	79.1	62.8	87.9	96.9	83.625	86.9	1
12	13	Johns Hopkins University	United States of America	80.9	58.5	89.2	92.3	100.0	86.4	1
13	14	Cornell University	United States of America	82.2	62.4	88.8	88.1	34.7	83.9	2
14	15	ETH Zurich – Swiss Federal Institute of Techno...	Switzerland	77.5	93.7	87.8	83.1	87	83.4	1
15	15	University of Michigan	United States of America	83.9	53.3	89.1	84.1	59.6	83.4	4
18	19	University of Pennsylvania	United States of America	71.8	32.9	82.7	93.6	43.7	79.5	2
19	20	Carnegie Mellon University	United States of America	70.3	39.1	79.3	95.7	53.7	79.3	1
20	21	University of Hong Kong	Hong Kong	68.4	91.4	71.4	96.1	56.5	79.2	1
21	22	University College London	United Kingdom	74.0	90.8	81.6	80.6	39.0	78.4	2
22	23	University of Washington	United States of America	68.2	49.0	77.1	95.9	32.8	78.0	4
23	24	Duke University	United States of America	66.8	49.4	71.5	92.3	100.0	76.5	1
24	25	Northwestern University	United States of America	64.5	60.5	68.8	95.3	75.125	75.9	1
26	27	Georgia Institute of Technology	United States of America	67.9	73.2	72.6	83.2	95.1	75.3	1
27	28	Pohang University of Science and Technology	South Korea	69.5	32.6	62.5	96.5	100.0	75.1	3

	world_rank	university_name	country	teaching	international	research	citations	income	total_score	num_stu
28	29	University of California, Santa Barbara	United States of America	56.6	64.3	68.0	98.8	89.8	75.0	2
29	30	University of British Columbia	Canada	65.1	93.3	74.8	80.3	42.6	73.8	5
30	30	University of North Carolina at Chapel Hill	United States of America	70.9	21.5	75.1	85.0	50.2	73.8	2
31	32	University of California, San Diego	United States of America	59.8	31.6	76.3	90.8	51.8	73.2	2
32	33	University of Illinois at Urbana-Champaign	United States of America	68.1	55.9	80.9	72.9	70.675	73.0	4
33	34	National University of Singapore	Singapore	65.5	97.8	72.6	78.7	40.5	72.9	3
...	...	...	...	...	...	...	...	...	...	
2572	601-800	Technical University of Madrid	Spain	21.8	39.5	14.6	24.5	38.3	29.225	4
2573	601-800	University of Tehran	Iran	26.1	16.5	16.9	15.8	16.4	16.4	5
2574	601-800	University of Texas at El Paso	United States of America	18.6	30.4	18.7	18.4	22.5	22.5	1
2575	601-800	Texas Tech University	United States of America	27.9	36.8	17.2	22.0	25.3333	25.3333	2
2576	601-800	Tokai University	Japan	17.9	19.3	7.6	15.3	34.4	19.15	2
2577	601-800	Tokushima University	Japan	25.3	16.8	21.6	12.8	59.6	27.7	[illegible]
2578	601-800	Tokyo University of Marine Science and Technology	Japan	27.9	24.5	12.4	7.7	57.9	25.625	[illegible]
2579	601-800	Tokyo University of Science	Japan	23.0	15.4	24.1	21.4	37.6	24.625	2
2580	601-800	Tomsk State University	Russian Federation	34.8	36.9	20.8	7.6	44.0	27.325	1
2581	601-800	Tottori University	Japan	24.3	16.7	10.1	9.6	34.5	17.725	[illegible]
2582	601-800	Toyohashi University of Technology	Japan	22.0	25.4	18.9	15.8	50.3	27.6	[illegible]
2583	601-800	Universiti Kebangsaan Malaysia	Malaysia	24.3	29.7	15.9	10.9	28.4	21.225	2
2584	601-800	Universiti Putra Malaysia	Malaysia	25.3	50.1	20.9	10.2	34.2	28.85	2
2585	601-800	Universiti Sains Malaysia	Malaysia	26.9	44.2	16.6	12.4	34.4	26.9	2
2586	601-800	Universiti Teknologi MARA	Malaysia	15.2	14.8	7.7	18.2	28.3	17.25	6
2587	601-800	Ural Federal University	Russian Federation	24.8	17.3	10.6	16.8	35.6	20.075	2
2588	601-800	V.N. Karazin Kharkiv National University	Ukraine	21.7	48.4	8.9	1.7	28.8	21.95	1
2589	601-800	University of Vigo	Spain	18.4	30.7	10.5	31.8	38.1	27.775	2
2590	601-800	Vilnius University	Lithuania	18.3	40.8	13.6	26.1	41.0	30.375	1
2591	601-800	Warsaw University of Technology	Poland	19.4	20.7	8.5	40.3	47.4	29.225	3
2592	601-800	Waseda University	Japan	23.6	29.7	14.6	29.4	32.4	26.525	5

	world_rank	university_name	country	teaching	international	research	citations	income	total_score	num_stu
2593	601-800	University of West Bohemia	Czech Republic	16.3	23.1	9.7	29.8	32.1	23.675	1!
2594	601-800	University of the West of England	United Kingdom	16.9	48.5	11.2	34.6	28.5	30.7	2:
2595	601-800	West University of Timişoara	Romania	16.1	21.0	3.9	22.4	15.7667	15.7667	1:
2596	601-800	University of Westminster	United Kingdom	17.3	81.9	11.7	21.1	28.5	35.8	1(
2597	601-800	Xidian University	China	17.9	12.8	12.1	8.9	83.7	29.375	3.
2598	601-800	Yeungnam University	South Korea	18.6	24.3	10.9	26.5	35.4	24.275	2.
2599	601-800	Yıldız Technical University	Turkey	14.5	14.9	7.6	19.3	44.0	21.45	3.
2601	601-800	Yokohama National University	Japan	20.1	23.3	16.0	13.5	40.4	23.3	1
2602	601-800	Yuan Ze University	Taiwan	16.2	17.7	18.3	28.6	39.8	26.1	:

2362 rows × 14 columns

In [14]:

```
# Чуть поправляем данные

fixed_data.international_students = fixed_data.international_students.str.replace('\%',
'')

numeric_cols = ['teaching', 'international', 'research', 'citations', 'income', 'total_s
core', 'num_students', 'international_students']
fixed_data[numeric_cols] = fixed_data[numeric_cols].apply(pd.to_numeric)

fixed_data.dtypes

def fix_female_male_ratio(x):
    if x == '-':
        return 1
    else:
        split = x.split(' : ')
        f = int(split[0])
        m = int(split[1])
        if m == 0:
            m = 0.01
        return f / m

def fix_world_rank(x):
    return int(x.split('-')[0].replace('=', ''))

fixed_data.female_male_ratio = fixed_data.female_male_ratio.apply(fix_female_male_ratio)
fixed_data.world_rank = fixed_data.world_rank.apply(fix_world_rank)
```

In [15]:
```python
def group_all_years(years):
    sorted_years = years.sort_values('year', ascending=False)
    first_year = sorted_years.head(1)
    last_year = sorted_years.tail(1)
    for col in ['teaching', 'international', 'research', 'citations', 'income', 'total_s
core', 'num_students', 'international_students']:
        last_year[col + '_progress'] = years.iloc[0][col] / years.iloc[-1][col]
    return last_year

fixed_data_with_progress = fixed_data.groupby('university_name').apply(group_all_years)
fixed_data_with_progress = fixed_data_with_progress.drop('year', axis=1)
fixed_data_with_progress
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:6: RuntimeWarning: invalid v
alue encountered in long_scalars
```

Out[15]:

		world_rank	university_name	country	teaching	international	research	citations	income
university_name									
AGH University of Science and Technology	**2405**	601	AGH University of Science and Technology	Poland	14.2	17.9	3.7	35.7	19.100000
Aalborg University	**501**	301	Aalborg University	Denmark	19.0	75.3	20.0	27.1	36.400000
Aalto University	**502**	301	Aalto University	Finland	26.2	49.0	22.2	37.5	61.900000
Aarhus University	**166**	167	Aarhus University	Denmark	38.1	33.4	55.6	57.3	61.500000
Aberystwyth University	**476**	276	Aberystwyth University	United Kingdom	19.8	63.8	15.5	56.6	35.500000
Adam Mickiewicz University	**2404**	601	Adam Mickiewicz University	Poland	20.0	25.7	11.0	15.3	28.700000
Aix-Marseille University	**2057**	251	Aix-Marseille University	France	36.7	63.0	22.1	64.9	33.100000
Ajou University	**2406**	601	Ajou University	South Korea	19.5	20.0	11.9	23.9	45.700000
Alexandria University	**146**	147	Alexandria University	Egypt	29.5	19.3	28.0	99.8	36.000000
Alexandru Ioan Cuza University	**2409**	601	Alexandru Ioan Cuza University	Romania	24.9	46.9	13.6	7.0	28.200000
Aligarh Muslim University	**2410**	601	Aligarh Muslim University	India	28.3	18.7	10.0	20.9	29.600000
American University	**2204**	401	American University	United States of America	42.2	28.9	16.5	41.1	35.900000
American University of Beirut	**2303**	501	American University of Beirut	Lebanon	27.7	93.0	11.2	31.9	45.366667
American University of Sharjah	**2411**	601	American University of Sharjah	United Arab Emirates	12.4	95.6	10.6	13.3	33.300000
Amirkabir University of Technology	**2304**	501	Amirkabir University of Technology	Iran	24.5	7.7	25.7	34.8	55.700000
Anadolu University	**2413**	601	Anadolu University	Turkey	12.2	14.3	22.6	10.9	100.000000
Andhra University	**2414**	601	Andhra University	India	34.8	7.2	6.9	1.2	31.300000
Aristotle University of Thessaloniki	**2416**	601	Aristotle University of Thessaloniki	Greece	22.6	36.6	15.0	29.5	33.600000
Arizona State University	**160**	161	Arizona State University	United States of America	43.0	24.1	44.1	66.9	46.350000
Asia University, Taiwan	**2417**	601	Asia University, Taiwan	Taiwan	14.4	18.3	15.9	30.5	38.400000
Athens University of Economics and Business	**2418**	601	Athens University of Economics and Business	Greece	13.4	39.2	18.8	31.2	66.500000
Auburn University	**552**	350	Auburn University	United States of America	33.7	22.5	18.7	10.3	47.300000
Auckland University of Technology	**2419**	601	Auckland University of Technology	New Zealand	17.3	95.6	9.8	21.7	28.500000
Austral University of Chile	**2420**	601	Austral University of Chile	Chile	16.7	45.2	9.7	14.2	41.100000
Australian National University	**42**	43	Australian National University	Australia	51.9	93.9	62.4	81.0	76.075000
Autonomous University of Barcelona	**400**	201	Autonomous University of Barcelona	Spain	33.7	45.9	27.9	57.9	37.000000

		world_rank	university_name	country	teaching	international	research	citations	income
university_name									
Autonomous University of Madrid	**477**	276	Autonomous University of Madrid	Spain	28.8	39.7	21.4	47.5	32.500000
Babeş-Bolyai University	**2308**	501	Babeş-Bolyai University	Romania	27.9	35.4	12.5	32.1	28.600000
Bangor University	**450**	251	Bangor University	United Kingdom	24.9	67.1	23.8	48.0	29.000000
Bar-Ilan University	**504**	301	Bar-Ilan University	Israel	27.9	47.0	32.2	15.7	28.500000
...	**...**	...	...	...	...	...	...	...	...
Vilnius University	**2590**	601	Vilnius University	Lithuania	18.3	40.8	13.6	26.1	41.000000
Virginia Polytechnic Institute and State University	**474**	251	Virginia Polytechnic Institute and State Unive...	United States of America	36.9	25.1	40.4	30.3	24.200000
Vrije Universiteit Brussel	**548**	301	Vrije Universiteit Brussel	Belgium	23.4	49.4	21.1	36.7	66.200000
VŠB - Technical University of Ostrava	**2150**	301	VŠB - Technical University of Ostrava	Czech Republic	18.8	22.3	15.7	85.5	32.100000
Wageningen University and Research Center	**143**	144	Wageningen University and Research Center	Netherlands	58.5	24.3	48.8	53.0	44.500000
Wake Forest University	**91**	90	Wake Forest University	United States of America	54.6	24.4	42.9	79.2	51.050000
Warsaw University of Technology	**2591**	601	Warsaw University of Technology	Poland	19.4	20.7	8.5	40.3	47.400000
Waseda University	**599**	350	Waseda University	Japan	25.4	27.1	17.3	29.7	27.300000
Washington State University	**549**	301	Washington State University	United States of America	28.8	31.0	24.5	33.5	34.300000
Wayne State University	**475**	251	Wayne State University	United States of America	34.7	24.6	16.4	55.2	32.066667
West University of Timişoara	**2595**	601	West University of Timişoara	Romania	16.1	21.0	3.9	22.4	15.766667
Western Sydney University	**1800**	351	Western Sydney University	Australia	17.8	50.1	22.4	58.7	30.400000
William & Mary	**74**	75	William & Mary	United States of America	53.1	20.9	36.1	95.6	53.250000
Wuhan University of Technology	**1353**	301	Wuhan University of Technology	China	14.8	18.9	7.8	78.1	58.700000
Xiamen University	**2302**	401	Xiamen University	China	26.7	25.0	15.6	47.0	29.200000
Xidian University	**2597**	601	Xidian University	China	17.9	12.8	12.1	8.9	83.700000
Xi'an Jiaotong University	**2401**	501	Xi'an Jiaotong University	China	28.7	25.8	22.5	25.5	70.400000
Yale University	**9**	10	Yale University	United States of America	92.1	59.2	89.7	91.5	82.475000
Yeshiva University	**69**	68	Yeshiva University	United States of America	63.5	53.3	46.7	74.4	58.950000
Yeungnam University	**2598**	601	Yeungnam University	South Korea	18.6	24.3	10.9	26.5	35.400000
Yokohama National University	**2601**	601	Yokohama National University	Japan	20.1	23.3	16.0	13.5	40.400000

		world_rank	university_name	country	teaching	international	research	citations	income
university_name									
York University	**500**	276	York University	Canada	19.9	57.7	27.7	41.2	41.700000
Yuan Ze University	**601**	350	Yuan Ze University	Taiwan	10.8	12.8	9.6	58.3	29.200000
Yıldız Technical University	**2599**	601	Yıldız Technical University	Turkey	14.5	14.9	7.6	19.3	44.000000
Zhejiang University	**197**	197	Zhejiang University	China	54.6	29.6	41.3	44.3	70.300000
École Normale Supérieure	**41**	42	École Normale Supérieure	France	66.8	44.9	48.2	95.7	30.700000
École Normale Supérieure de Lyon	**99**	100	École Normale Supérieure de Lyon	France	51.1	37.6	34.4	88.8	26.100000
École Polytechnique	**38**	39	École Polytechnique	France	57.9	77.9	56.1	91.4	73.725000
École Polytechnique Fédérale de Lausanne	**47**	48	École Polytechnique Fédérale de Lausanne	Switzerland	55.0	100.0	56.1	83.8	38.000000
Örebro University	**2134**	301	Örebro University	Sweden	18.3	39.4	10.4	87.8	29.800000

739 rows × 21 columns

3) Визуальное исследование датасета

In [16]:
```
data = fixed_data_with_progress
data
```

Out[16]:

university_name		world_rank	university_name	country	teaching	international	research	citations	income
AGH University of Science and Technology	2405	601	AGH University of Science and Technology	Poland	14.2	17.9	3.7	35.7	19.100000
Aalborg University	501	301	Aalborg University	Denmark	19.0	75.3	20.0	27.1	36.400000
Aalto University	502	301	Aalto University	Finland	26.2	49.0	22.2	37.5	61.900000
Aarhus University	166	167	Aarhus University	Denmark	38.1	33.4	55.6	57.3	61.500000
Aberystwyth University	476	276	Aberystwyth University	United Kingdom	19.8	63.8	15.5	56.6	35.500000
Adam Mickiewicz University	2404	601	Adam Mickiewicz University	Poland	20.0	25.7	11.0	15.3	28.700000
Aix-Marseille University	2057	251	Aix-Marseille University	France	36.7	63.0	22.1	64.9	33.100000
Ajou University	2406	601	Ajou University	South Korea	19.5	20.0	11.9	23.9	45.700000
Alexandria University	146	147	Alexandria University	Egypt	29.5	19.3	28.0	99.8	36.000000
Alexandru Ioan Cuza University	2409	601	Alexandru Ioan Cuza University	Romania	24.9	46.9	13.6	7.0	28.200000
Aligarh Muslim University	2410	601	Aligarh Muslim University	India	28.3	18.7	10.0	20.9	29.600000
American University	2204	401	American University	United States of America	42.2	28.9	16.5	41.1	35.900000
American University of Beirut	2303	501	American University of Beirut	Lebanon	27.7	93.0	11.2	31.9	45.366667
American University of Sharjah	2411	601	American University of Sharjah	United Arab Emirates	12.4	95.6	10.6	13.3	33.300000
Amirkabir University of Technology	2304	501	Amirkabir University of Technology	Iran	24.5	7.7	25.7	34.8	55.700000
Anadolu University	2413	601	Anadolu University	Turkey	12.2	14.3	22.6	10.9	100.000000
Andhra University	2414	601	Andhra University	India	34.8	7.2	6.9	1.2	31.300000
Aristotle University of Thessaloniki	2416	601	Aristotle University of Thessaloniki	Greece	22.6	36.6	15.0	29.5	33.600000
Arizona State University	160	161	Arizona State University	United States of America	43.0	24.1	44.1	66.9	46.350000
Asia University, Taiwan	2417	601	Asia University, Taiwan	Taiwan	14.4	18.3	15.9	30.5	38.400000
Athens University of Economics and Business	2418	601	Athens University of Economics and Business	Greece	13.4	39.2	18.8	31.2	66.500000
Auburn University	552	350	Auburn University	United States of America	33.7	22.5	18.7	10.3	47.300000
Auckland University of Technology	2419	601	Auckland University of Technology	New Zealand	17.3	95.6	9.8	21.7	28.500000
Austral University of Chile	2420	601	Austral University of Chile	Chile	16.7	45.2	9.7	14.2	41.100000
Australian National University	42	43	Australian National University	Australia	51.9	93.9	62.4	81.0	76.075000
Autonomous University of Barcelona	400	201	Autonomous University of Barcelona	Spain	33.7	45.9	27.9	57.9	37.000000

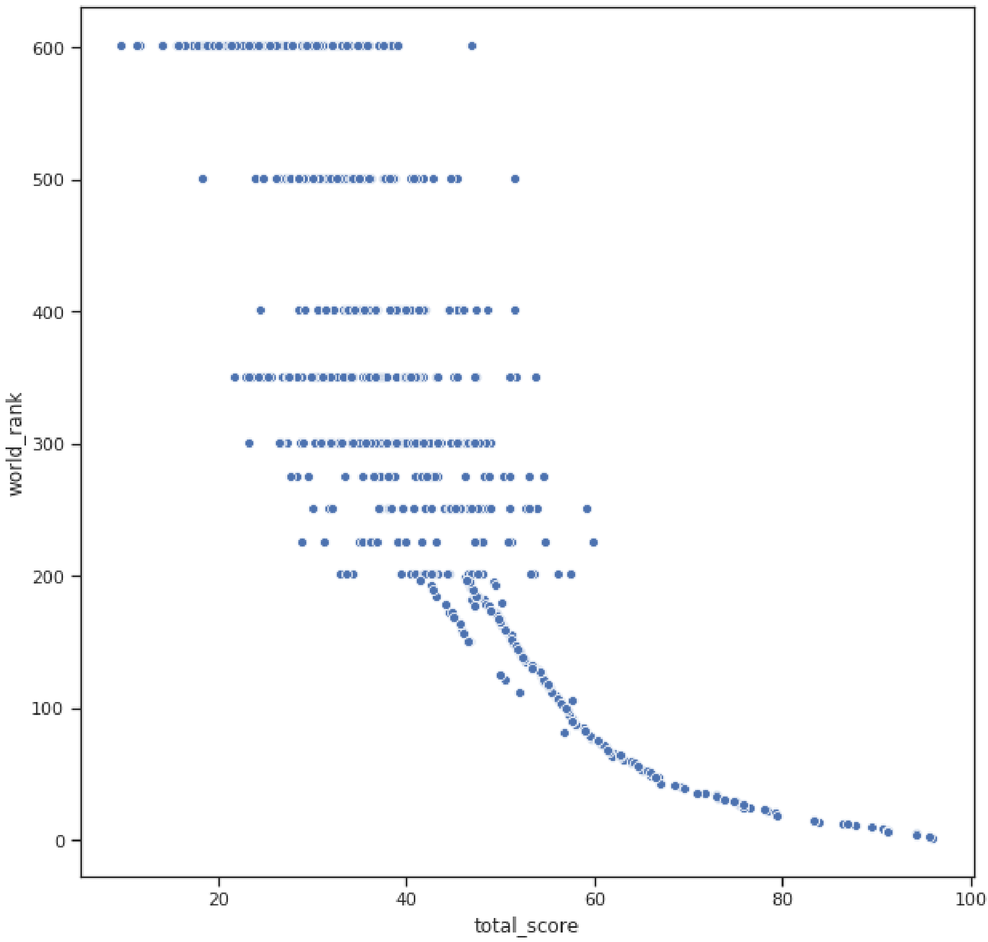
university_name		world_rank	university_name	country	teaching	international	research	citations	income
Autonomous University of Madrid	**477**	276	Autonomous University of Madrid	Spain	28.8	39.7	21.4	47.5	32.500000
Babeş-Bolyai University	**2308**	501	Babeş-Bolyai University	Romania	27.9	35.4	12.5	32.1	28.600000
Bangor University	**450**	251	Bangor University	United Kingdom	24.9	67.1	23.8	48.0	29.000000
Bar-Ilan University	**504**	301	Bar-Ilan University	Israel	27.9	47.0	32.2	15.7	28.500000
...	**...**	...	...	...	...	...	...	...	...
Vilnius University	**2590**	601	Vilnius University	Lithuania	18.3	40.8	13.6	26.1	41.000000
Virginia Polytechnic Institute and State University	**474**	251	Virginia Polytechnic Institute and State Unive...	United States of America	36.9	25.1	40.4	30.3	24.200000
Vrije Universiteit Brussel	**548**	301	Vrije Universiteit Brussel	Belgium	23.4	49.4	21.1	36.7	66.200000
VŠB - Technical University of Ostrava	**2150**	301	VŠB - Technical University of Ostrava	Czech Republic	18.8	22.3	15.7	85.5	32.100000
Wageningen University and Research Center	**143**	144	Wageningen University and Research Center	Netherlands	58.5	24.3	48.8	53.0	44.500000
Wake Forest University	**91**	90	Wake Forest University	United States of America	54.6	24.4	42.9	79.2	51.050000
Warsaw University of Technology	**2591**	601	Warsaw University of Technology	Poland	19.4	20.7	8.5	40.3	47.400000
Waseda University	**599**	350	Waseda University	Japan	25.4	27.1	17.3	29.7	27.300000
Washington State University	**549**	301	Washington State University	United States of America	28.8	31.0	24.5	33.5	34.300000
Wayne State University	**475**	251	Wayne State University	United States of America	34.7	24.6	16.4	55.2	32.066667
West University of Timişoara	**2595**	601	West University of Timişoara	Romania	16.1	21.0	3.9	22.4	15.766667
Western Sydney University	**1800**	351	Western Sydney University	Australia	17.8	50.1	22.4	58.7	30.400000
William & Mary	**74**	75	William & Mary	United States of America	53.1	20.9	36.1	95.6	53.250000
Wuhan University of Technology	**1353**	301	Wuhan University of Technology	China	14.8	18.9	7.8	78.1	58.700000
Xiamen University	**2302**	401	Xiamen University	China	26.7	25.0	15.6	47.0	29.200000
Xidian University	**2597**	601	Xidian University	China	17.9	12.8	12.1	8.9	83.700000
Xi'an Jiaotong University	**2401**	501	Xi'an Jiaotong University	China	28.7	25.8	22.5	25.5	70.400000
Yale University	**9**	10	Yale University	United States of America	92.1	59.2	89.7	91.5	82.475000
Yeshiva University	**69**	68	Yeshiva University	United States of America	63.5	53.3	46.7	74.4	58.950000
Yeungnam University	**2598**	601	Yeungnam University	South Korea	18.6	24.3	10.9	26.5	35.400000
Yokohama National University	**2601**	601	Yokohama National University	Japan	20.1	23.3	16.0	13.5	40.400000

university_name		world_rank	university_name	country	teaching	international	research	citations	income
York University	**500**	276	York University	Canada	19.9	57.7	27.7	41.2	41.700000
Yuan Ze University	**601**	350	Yuan Ze University	Taiwan	10.8	12.8	9.6	58.3	29.200000
Yıldız Technical University	**2599**	601	Yıldız Technical University	Turkey	14.5	14.9	7.6	19.3	44.000000
Zhejiang University	**197**	197	Zhejiang University	China	54.6	29.6	41.3	44.3	70.300000
École Normale Supérieure	**41**	42	École Normale Supérieure	France	66.8	44.9	48.2	95.7	30.700000
École Normale Supérieure de Lyon	**99**	100	École Normale Supérieure de Lyon	France	51.1	37.6	34.4	88.8	26.100000
École Polytechnique	**38**	39	École Polytechnique	France	57.9	77.9	56.1	91.4	73.725000
École Polytechnique Fédérale de Lausanne	**47**	48	École Polytechnique Fédérale de Lausanne	Switzerland	55.0	100.0	56.1	83.8	38.000000
Örebro University	**2134**	301	Örebro University	Sweden	18.3	39.4	10.4	87.8	29.800000

739 rows × 21 columns

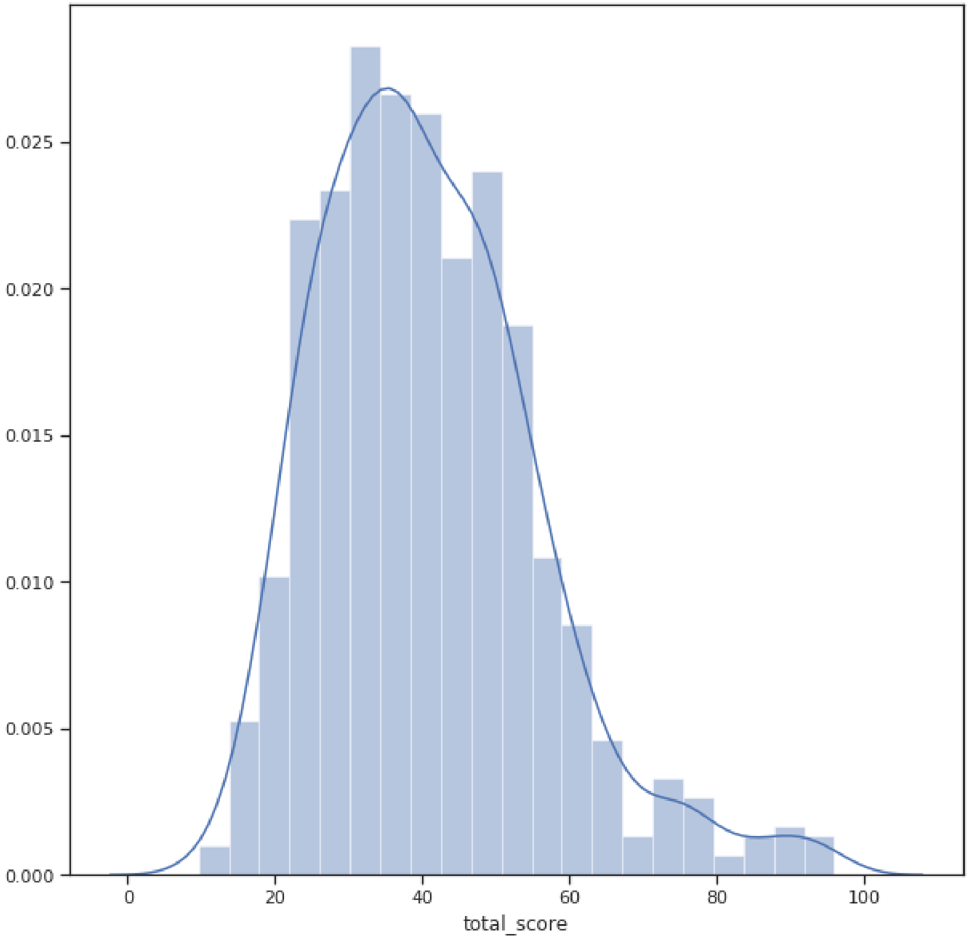
In [17]:
```
fig, ax = plt.subplots(figsize=(10,10))
sns.scatterplot(ax=ax, x='total_score', y='world_rank', data=data)
```

Out[17]: `<matplotlib.axes._subplots.AxesSubplot at 0x7f5cee250ef0>`



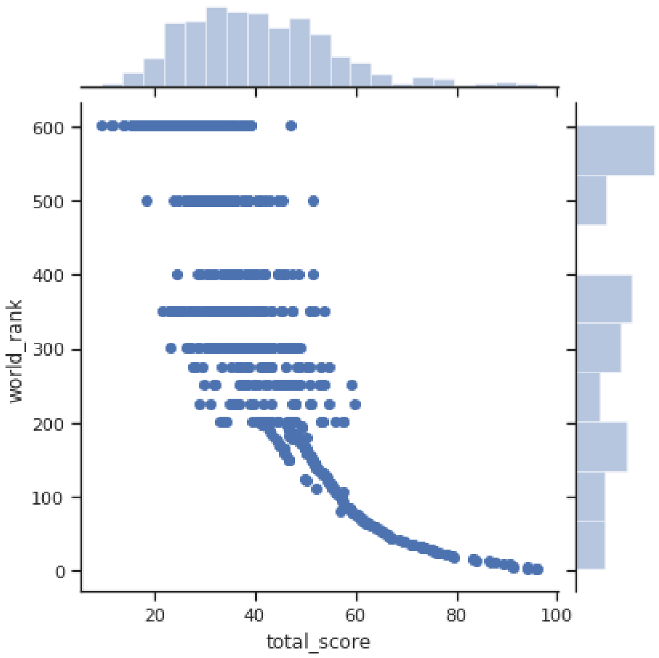
In [18]:
```
fig, ax = plt.subplots(figsize=(10,10))
sns.distplot(data['total_score'])
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5cee0257b8>

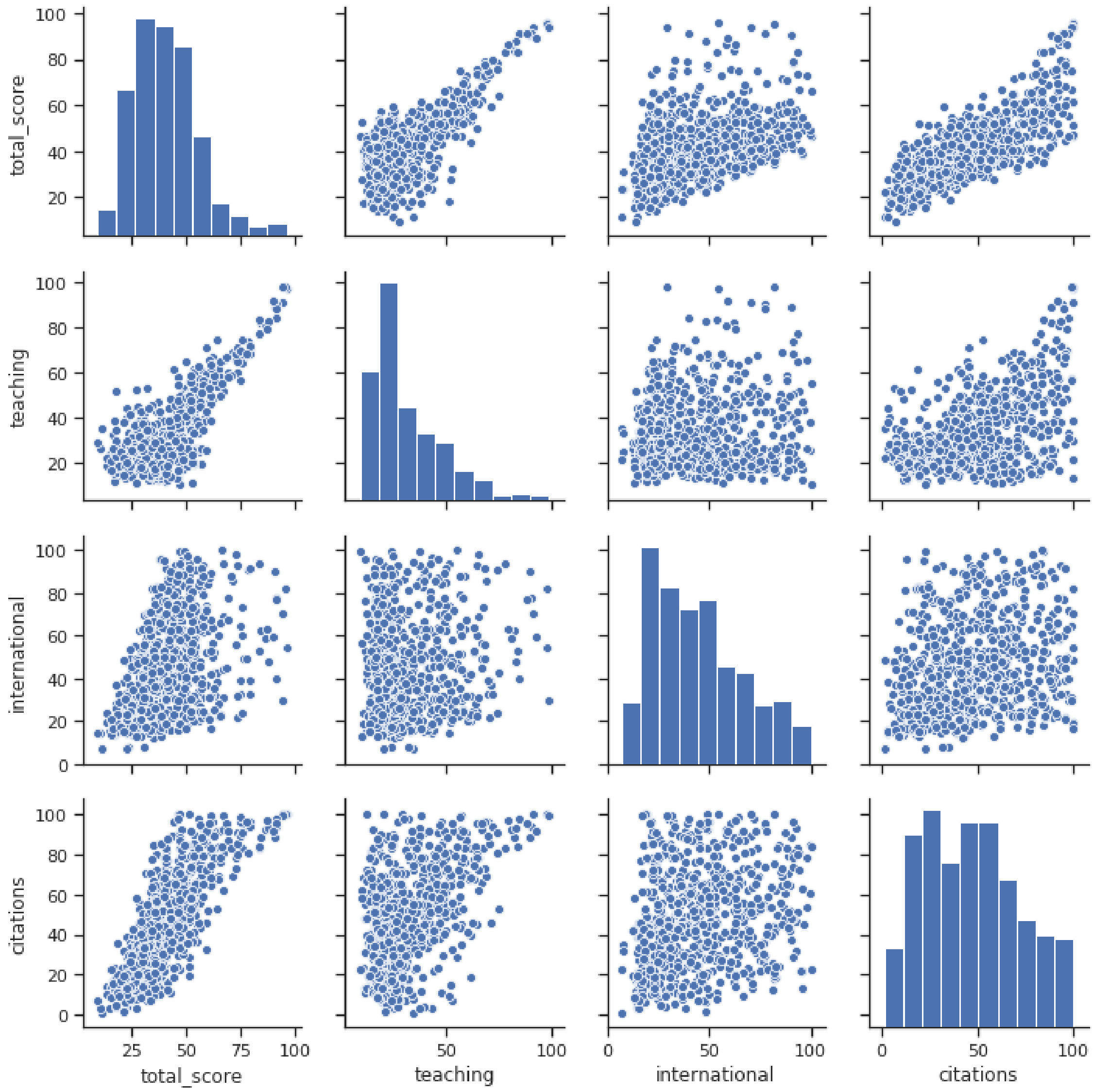


In [19]:
```
sns.jointplot(x='total_score', y='world_rank', data=data)
```

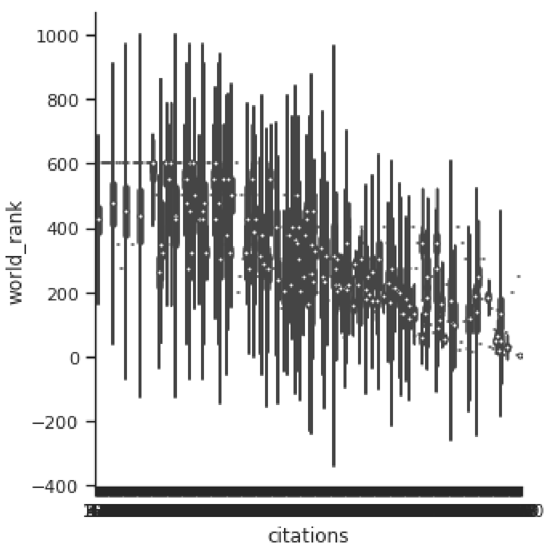
Out[19]: <seaborn.axisgrid.JointGrid at 0x7f5d21f0e908>



In [20]:
```
sns.pairplot(data[['total_score', 'teaching', 'international', 'citations']])
```
Out[20]: <seaborn.axisgrid.PairGrid at 0x7f5ceddd1e80>



In [21]:
```
sns.catplot(y='world_rank', x='citations', data=data, kind="violin", split=True)
```
Out[21]: <seaborn.axisgrid.FacetGrid at 0x7f5cedf75fd0>



4) Информация о корреляции признаков

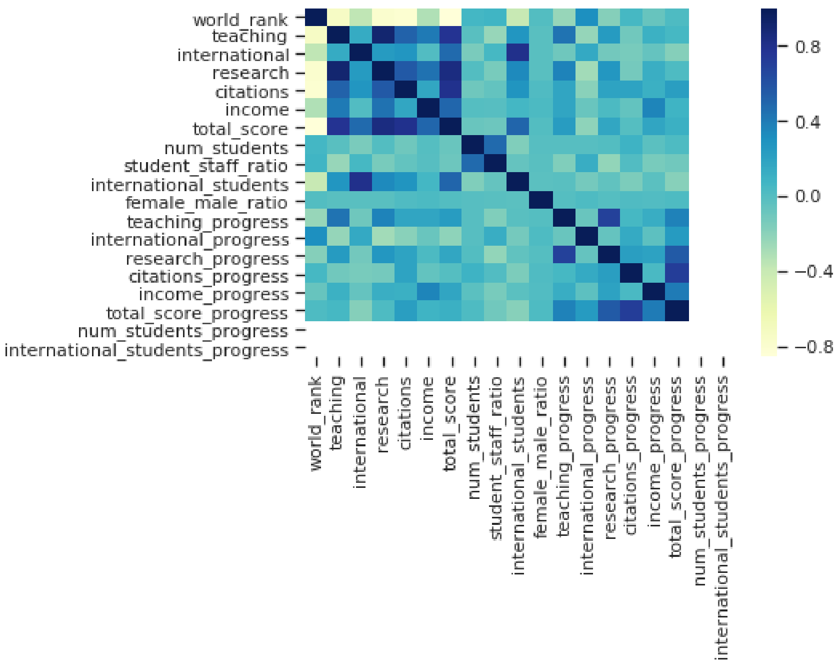
In [22]:
```python
data.corr()
```

Out[22]:

	world_rank	teaching	international	research	citations	income	total_score	num_st
world_rank	1.000000	-0.732092	-0.365120	-0.785236	-0.805942	-0.317702	-0.853945	0.0
teaching	-0.732092	1.000000	0.143880	0.908191	0.521600	0.408316	0.770419	-0.0
international	-0.365120	0.143880	1.000000	0.240472	0.273609	0.013415	0.484897	-0.1
research	-0.785236	0.908191	0.240472	1.000000	0.561164	0.444592	0.846262	0.0
citations	-0.805942	0.521600	0.273609	0.561164	1.000000	0.167780	0.792220	-0.0
income	-0.317702	0.408316	0.013415	0.444592	0.167780	1.000000	0.498773	-0.0
total_score	-0.853945	0.770419	0.484897	0.846262	0.792220	0.498773	1.000000	-0.0
num_students	0.065192	-0.020426	-0.137339	0.005305	-0.099872	-0.005632	-0.075964	1.0
student_staff_ratio	0.080110	-0.232529	0.056522	-0.136723	-0.055355	-0.008549	-0.087236	0.4
international_students	-0.393047	0.275345	0.803313	0.339426	0.279234	0.058790	0.503380	-0.1
female_male_ratio	0.003110	-0.018707	-0.014194	-0.018145	0.016426	0.033576	0.005745	-0.0
teaching_progress	-0.231821	0.438598	-0.096220	0.365263	0.177647	0.177694	0.236082	-0.
international_progress	0.308062	-0.225894	0.156078	-0.268961	-0.189165	-0.072636	-0.205629	-0.
research_progress	-0.177708	0.246676	-0.109139	0.267249	0.192014	0.035306	0.172600	0.0
citations_progress	0.056633	-0.102778	-0.131552	-0.118460	0.191262	-0.054486	-0.004056	0.0
income_progress	-0.065488	0.108176	-0.059840	0.123628	0.110871	0.361134	0.177728	-0.0
total_score_progress	0.023829	0.056909	-0.179233	0.027362	0.219279	0.082414	0.114138	0.0
num_students_progress	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
international_students_progress	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

In [23]:
```python
sns.heatmap(data.corr(), cmap='YlGnBu', fmt='.0f')
```

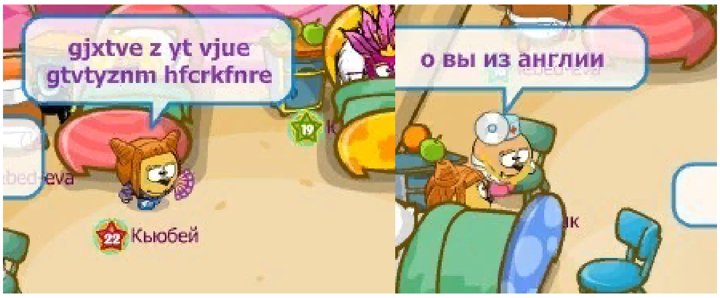
Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5ce530b278>



In [ ]:

Отчёт по ЛР №2 по ТМО

Водка Игорь, ИУ5-61



Example of using pandasql library for data analysis

In [ ]:
```
%matplotlib inline
import pandas as pd
import pandasql as ps
from datetime import datetime
import seaborn
import matplotlib.pyplot as plt

%config InlineBackend.figure_format = 'svg'
from pylab import rcParams
rcParams['figure.figsize'] = 8, 5
```

In [ ]:
```
pd.__version__
```

In [ ]:
```
project_submissions = pd.read_csv('./data/project_submissions.csv')
daily_engagements = pd.read_csv('./data/daily_engagement.csv')
enrollments = pd.read_csv('./data/enrollments.csv')
```

Simple SQL query

getting accounts and date with maximum total time spent on Udacity

In [ ]:
```
# pandasql code
def example1_pandasql(daily_engagements):
    simple_query = '''
        SELECT
            acct,
            total_minutes_visited,
            utc_date
        FROM daily_engagements
        ORDER BY total_minutes_visited desc
        LIMIT 10
        '''
    return ps.sqldf(simple_query, locals())

# pandas code
def example1_pandas(daily_engagements):
    return daily_engagements[['acct', 'total_minutes_visited', 'utc_date']].sort_values(
by ='total_minutes_visited', ascending = False)[:10]
```

In [ ]:
```
example1_pandasql(daily_engagements)
```

In [ ]:
```
example1_pandas(daily_engagements)
```

SQL query with aggregating functions

Let's see whether there's weekly seasonality: on average students spent more time on weekends then on weekdays

In [ ]:
```
# ТУТ НЕ РАБОТАЛО. ДОБАВИЛ list() ВОКРУГ map()

daily_engagements['weekday'] = list(map(lambda x: datetime.strptime(x, '%Y-%m-%d').strft
ime('%A'), daily_engagements.utc_date))
```

In [ ]:
```
daily_engagements.head()
```

In [ ]:
```
# pandasql code
def example2_pandasql(daily_engagements):
    aggr_query = '''
        SELECT
            avg(total_minutes_visited) as total_minutes_visited,
            weekday
        FROM daily_engagements
        GROUP BY weekday
        '''
    return ps.sqldf(aggr_query, locals()).set_index('weekday')

# pandas code
def example2_pandas(daily_engagements):
    return pd.DataFrame(daily_engagements.groupby('weekday').total_minutes_visited.mean
())
```

In [ ]:
```
weekday_engagement = example2_pandasql(daily_engagements)
weekday_engagement
```

In [ ]:
```
example2_pandas(daily_engagements)
```

In [ ]:
```
week_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunda
y']
weekday_engagement.loc[week_order].plot(kind = 'bar', rot = 45, title = 'Total time spen
t on Udacity by weekday')
```

Joining tables

Let's see whether students that canceled program was spending less time on Udacity within first week of enrollment. Note we need to filter out Udacity test users not to spoil statistics. Also we need to take into account the fact that student may join several times.

In [ ]:

```
# pandasql code
def example3_pandasql(enrollments, daily_engagements):
    join_query = '''
        SELECT
            avg(avg_acct_total_minutes) as avg_total_minutes,
            status
        FROM
            (SELECT
                avg(total_minutes_visited) as avg_acct_total_minutes,
                status,
                account_key
            FROM
                (SELECT
                    e.account_key,
                    e.status,
                    de.total_minutes_visited,
                    (cast(strftime('%s',de.utc_date) as interger) - cast(strftime('%s',
e.join_date) as interger))/(24*60*60) as days_since_joining,
                    (cast(strftime('%s',e.cancel_date) as interger) - cast(strftime('%s
', de.utc_date) as interger))/(24*60*60) as days_before_cancel
                FROM enrollments as e JOIN daily_engagements as de ON (e.account_key = d
e.acct)
                WHERE (is_udacity = 0) AND (days_since_joining < 7) AND (days_since_join
ing >= 0)
                    AND ((days_before_cancel >= 0) OR (status = 'current'))
                )
            GROUP BY status, account_key)
        GROUP BY status
    '''
    return ps.sqldf(join_query, locals()).set_index('status')

# pandas code
def example3_pandas(enrollments, daily_engagements):
    join_df = pd.merge(daily_engagements,
                  enrollments[enrollments.is_udacity == 0],
                  how = 'inner',
                  right_on ='account_key',
                  left_on = 'acct')
    join_df = join_df[['account_key', 'status', 'total_minutes_visited', 'utc_date', 'jo
in_date', 'cancel_date']]

    join_df['days_since_joining'] = map(lambda x: x.days,
                                        pd.to_datetime(join_df.utc_date) - pd.to_datetim
e(join_df.join_date))

    join_df['before_cancel'] = (pd.to_datetime(join_df.utc_date) <= pd.to_datetime(join_
df.cancel_date))
    join_df = join_df[join_df.before_cancel | (join_df.status == 'current')]

    join_df = join_df[(join_df.days_since_joining < 7) & (join_df.days_since_joining >=
0)]
    avg_account_total_minutes = pd.DataFrame(join_df.groupby(['account_key', 'status'],
as_index = False)
                                                     .total_minutes_visited.mean())
    avg_total_minutes= pd.DataFrame(avg_account_total_minutes.groupby('status').total_mi
nutes_visited.mean())
    avg_total_minutes.columns = ['avg_total_minutes']
    return avg_total_minutes
```

In [ ]:

```
example3_pandasql(enrollments, daily_engagements)
```

In [ ]:

```
example3_pandas(enrollments, daily_engagements)
```

Estimating time elapsed

In [ ]:
```
import time

def count_mean_time(func, params, N =5):
    total_time = 0
    for i in range(N):
        time1 = time.time()
        if len(params) == 1:
            tmp_df = func(params[0])
        elif len(params) == 2:
            tmp_df = func(params[0], params[1])
        time2 = time.time()
        total_time += (time2 - time1)
    return total_time/N
```

Example #1

In [ ]:
```
ex1_times = []
for count in range(1000, 137000, 1000):
    pandasql_time = count_mean_time(example1_pandasql, [daily_engagements[:count]])
    pandas_time = count_mean_time(example1_pandas, [daily_engagements[:count]])
    ex1_times.append({'count': count, 'pandasql_time': pandasql_time, 'pandas_time': pan
das_time})
```

In [ ]:
```
ex1_times_df = pd.DataFrame(ex1_times)
ex1_times_df.columns = ['number of rows in daily_engagements', 'pandas time', 'pandasql
 time']
ex1_times_df = ex1_times_df.set_index('number of rows in daily_engagements')
```

In [ ]:
```
ax = ex1_times_df.plot(title = 'Example #1 time elapsed (seconds)', subplots = True)
```

Example #2

In [ ]:
```
ex2_times = []
for count in range(1000, 137000, 1000):
    pandasql_time = count_mean_time(example2_pandasql, [daily_engagements[:count]])
    pandas_time = count_mean_time(example2_pandas, [daily_engagements[:count]])
    ex2_times.append({'count': count, 'pandasql_time': pandasql_time, 'pandas_time': pan
das_time})
```

In [ ]:
```
ex2_times_df = pd.DataFrame(ex2_times)
```

In [ ]:
```
ex2_times_df.columns = ['number of rows in daily_engagements', 'pandas time', 'pandasql
 time']
ex2_times_df = ex2_times_df.set_index('number of rows in daily_engagements')
```

In [ ]:
```
ax = ex2_times_df.plot(title = 'Example #2 time elapsed (seconds)', subplots = True)
```

Example #3

In [ ]:
```
all_users = enrollments.account_key.unique().tolist()
len(all_users)
```

In [ ]:
```
ex3_times = []
for users_count in range(10, 1310, 10):
    users = all_users[:users_count]
    enrollments_sample = enrollments[enrollments.account_key.isin(users)]
    daily_engagements_sample = daily_engagements[daily_engagements.acct.isin(users)]
    count = daily_engagements_sample.shape[0]
    pandasql_time = count_mean_time(example3_pandasql, [enrollments_sample, daily_engage
ments_sample])
    pandas_time = count_mean_time(example3_pandas, [enrollments_sample, daily_engagement
s_sample])
    ex3_times.append({'count': count, 'pandasql_time': pandasql_time, 'pandas_time': pan
das_time})
```

In [ ]:
```
ex3_times_df = pd.DataFrame(ex3_times).set_index('count')
```

In [ ]:
```
ax = ex3_times_df.plot(title = 'Example #3 time elapsed')
ax.set_xlabel('number of rows in daily_engagements')
ax.set_ylabel('time, seconds')
```

In [ ]:
```

```

Лабораторная работа 3 по ТМО

Водка Игорь, ИУ5-61

Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных

In [69]:
```
# read the data
import pandas as pd
reviews = pd.read_csv("./winemag-data-130k-v2.csv", index_col=0)
pd.set_option('max_rows', 5)
```

Смотрим тип данных (допустим, цену):

In [70]:
```
reviews.price.dtype
```

Out[70]:
```
dtype('float64')
```

In [71]:
```
reviews.dtypes
```

Out[71]:
```
country        object
description    object
                ...
variety        object
winery         object
Length: 13, dtype: object
```

Пропуски в данных. Простая замена

Как видим, существует довольно много строк, где не указана страна:

In [72]:
```
reviews[reviews.country.isnull()]
```

Out[72]:

	country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle
913	NaN	Amber in color, this wine has aromas of peach ...	Asureti Valley	87	30.0	NaN	NaN	NaN	Mike DeSimone	@worldwineguys
3131	NaN	Soft, fruity and juicy, this is a pleasant, si...	Partager	83	NaN	NaN	NaN	NaN	Roger Voss	@vossroger
...	...	...	...	...	...	...	...	...	...	...
129590	NaN	A blend of 60% Syrah, 30% Cabernet Sauvignon a...	Shah	90	30.0	NaN	NaN	NaN	Mike DeSimone	@worldwineguys
129900	NaN	This wine offers a delightful bouquet of black...	NaN	91	32.0	NaN	NaN	NaN	Mike DeSimone	@worldwineguys

63 rows × 13 columns

Заменим пустые значения на Unknown:

In [73]:
```
for column in ["country", "region_1", "region_2"]:
    reviews[column] = reviews[column].fillna("Unknown")
```

Сработало:

In [74]:
```
reviews[reviews.country.isnull()]
```

Out[74]:

country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle	title

In [75]:
```
reviews[reviews.region_1.isnull()]
```

Out[75]:

country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle	title

In [76]:
```
reviews[reviews.region_2.isnull()]
```

Out[76]:

country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle	title

Пропуски в данных. Импьютация

У нас в датасете больше нет NaN. Но импьютацию из лекции опробовать охота. Сделаем виртуальный датасет:

In [77]:
```
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
```

In [78]:
```
# Фильтр для проверки заполнения пустых значений
d = {
      'name': ['Dasha', 'Tanya', 'Andrey', 'Igor', 'Katya', 'Rodion', 'Artyom'],
      'mood': [30, None, 20, 20, 25, None, 22]
    }
df = pd.DataFrame(data=d)

moods = df[['mood']]

indicator = MissingIndicator()
mask_missing_values_only = indicator.fit_transform(moods)
mask_missing_values_only
```

Out[78]:
```
array([[False],
       [ True],
       [False],
       [False],
       [False],
       [ True],
       [False]])
```

In [79]:
```
strategies=['mean', 'median','most_frequent']
```

In [80]:
```
def test_num_impute(strategy_param):
    imp_num = SimpleImputer(strategy=strategy_param)
    data_num_imp = imp_num.fit_transform(moods)
    return data_num_imp[mask_missing_values_only]
```

In [81]:
```
for strategy in strategies:
    print(strategy, test_num_impute(strategy))
```

```
mean [23.4 23.4]
median [22. 22.]
most_frequent [20. 20.]
```

До импьютации:

In [82]:
```
pd.set_option('display.max_rows', 500)
df.head(n=100500)
```

Out[82]:

	name	mood
0	Dasha	30.0
1	Tanya	NaN
2	Andrey	20.0
3	Igor	20.0
4	Katya	25.0
5	Rodion	NaN
6	Artyom	22.0

После импьютации:

In [83]:
```
test_num_impute('mean')[0]
```

Out[83]:
```
23.4
```

In [84]:
```
df_imputed = df
df_imputed['mood'] = df_imputed['mood'].fillna(test_num_impute('mean')[0])
df_imputed.head(n=100500)
```

Out[84]:

	name	mood
0	Dasha	30.0
1	Tanya	23.4
2	Andrey	20.0
3	Igor	20.0
4	Katya	25.0
5	Rodion	23.4
6	Artyom	22.0

А ещё можно просто выкидывать:

In [85]:
```
reviews = reviews.dropna()
```

Преобразование в категориальные признаки данных

In [86]:
```
reviews.country.unique()
```

Out[86]:
```
array(['Portugal', 'US', 'Spain', 'Italy', 'France', 'Argentina', 'Chile',
       'Australia', 'South Africa', 'Israel', 'Hungary', 'Austria',
       'Greece', 'Canada', 'Mexico', 'New Zealand', 'Romania', 'Germany',
       'Luxembourg', 'Georgia', 'Uruguay', 'England', 'Lebanon', 'Brazil',
       'Slovenia', 'Moldova', 'Czech Republic', 'Peru', 'India',
       'Bulgaria', 'Cyprus', 'Turkey', 'Armenia', 'Switzerland',
       'Croatia', 'Ukraine', 'Serbia', 'Morocco', 'Macedonia', 'China'],
      dtype=object)
```

Label encoding

```
In [87]: from sklearn.preprocessing import LabelEncoder, OneHotEncoder
         le = LabelEncoder()

         countries = pd.DataFrame({'country':reviews.country})
         countries['country'].unique()
```

```
Out[87]: array(['Portugal', 'US', 'Spain', 'Italy', 'France', 'Argentina', 'Chile',
                'Australia', 'South Africa', 'Israel', 'Hungary', 'Austria',
                'Greece', 'Canada', 'Mexico', 'New Zealand', 'Romania', 'Germany',
                'Luxembourg', 'Georgia', 'Uruguay', 'England', 'Lebanon', 'Brazil',
                'Slovenia', 'Moldova', 'Czech Republic', 'Peru', 'India',
                'Bulgaria', 'Cyprus', 'Turkey', 'Armenia', 'Switzerland',
                'Croatia', 'Ukraine', 'Serbia', 'Morocco', 'Macedonia', 'China'],
               dtype=object)
```

```
In [88]: countries_nums = le.fit_transform(countries['country'])

         import numpy
         numpy.unique(countries_nums)
```

```
Out[88]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
                17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
                34, 35, 36, 37, 38, 39])
```

```
In [89]: print(len(countries['country'].unique()), "vs", len(numpy.unique(countries_nums)))
```

```
40 vs 40
```

Ничего не потерялось.

One hot encoding

```
In [90]: df
```

Out[90]:

	name	mood
0	Dasha	30.0
1	Tanya	23.4
2	Andrey	20.0
3	Igor	20.0
4	Katya	25.0
5	Rodion	23.4
6	Artyom	22.0

```
In [91]: ohe = OneHotEncoder(categories="auto")

         cat_enc_ohe = ohe.fit_transform(df[["mood"]])
```

```
In [92]: cat_enc_ohe.shape
```

```
Out[92]: (7, 5)
```

```
In [93]: cat_enc_ohe.todense()
```

```
Out[93]: matrix([[0., 0., 0., 0., 1.],
                 [0., 0., 1., 0., 0.],
                 [1., 0., 0., 0., 0.],
                 [1., 0., 0., 0., 0.],
                 [0., 0., 0., 1., 0.],
                 [0., 0., 1., 0., 0.],
                 [0., 1., 0., 0., 0.]])
```

Круто!

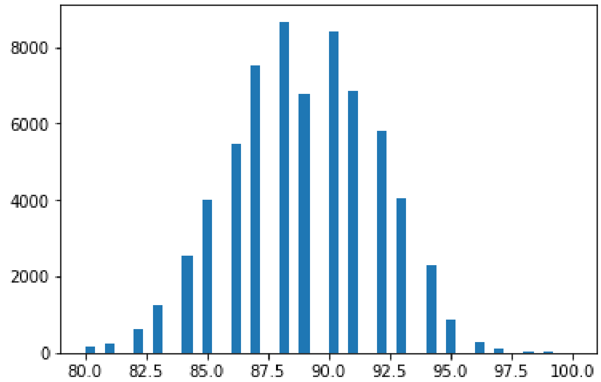
Масштабирование данных

In [94]:
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler, Normalizer
import seaborn as sns
import matplotlib.pyplot as plt
```

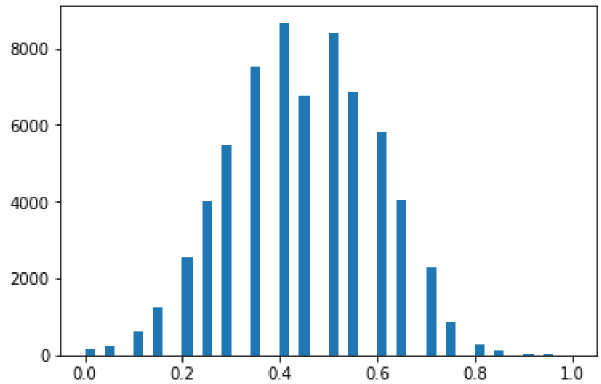
In [95]:
```
sc1 = MinMaxScaler()
sc1_data = sc1.fit_transform(reviews[['points']])
```

```
/home/igor-vodka/.local/lib/python3.6/site-packages/sklearn/preprocessing/data.py:323: DataConversionWarning: Data with input dtype int64 were all converted to float64 by MinMaxScaler.
  return self.partial_fit(X, y)
```

In [96]:
```
plt.hist(reviews['points'], 50)
plt.show()
```



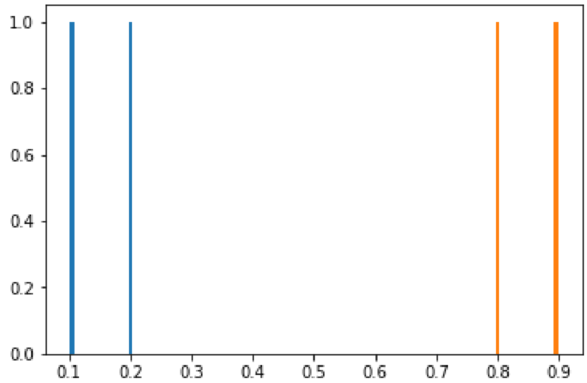
In [97]:
```
plt.hist(sc1_data, 50)
plt.show()
```



Немного нормализации данных

In [98]:
```
sc3 = Normalizer(norm='l1')
sc3_data = sc3.fit_transform([[1, 9], [2, 8]])
```

In [99]:
```
plt.hist(sc3_data, 50)
plt.show()
```



In [100]:
```
sc3_data
```

Out[100]:
```
array([[0.1, 0.9],
       [0.2, 0.8]])
```

Готово.

oo...

In [ ]:
```

```

Лабораторная работа 4 по ТМО

Водка Игорь, ИУ5-61

Подготовка обучающей и тестовой выборки, кросс-валидация и подбор гиперпараметров на примере метода ближайших соседей.

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

Возьмём из прошлой лабы. `../lab3/winemag-data-130k-v2.csv`

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

In [355]:
```
# read the data
import pandas as pd
reviews = pd.read_csv("../lab3/winemag-data-130k-v2.csv", index_col=0)
pd.set_option('max_rows', 5)

for column in ["country", "region_1", "region_2"]:
    reviews[column] = reviews[column].fillna("Unknown")

reviews['price'] = reviews.groupby('country').transform(lambda x: x.fillna(x.mean()))
```

In [356]:
```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

# country ok, winery ok
for feature in ['country', 'province', 'region_1', 'region_2', 'variety', 'winery']:
    le = LabelEncoder()
    reviews[feature] = reviews[feature].dropna()
    processed = pd.DataFrame({'result': reviews[feature]})
    reviews[feature] = le.fit_transform(processed['result'].astype(str))
```

In [357]:
```
reviews
```

Out[357]:

	country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle
0	22	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	87.0	331	424	15	Kerin O'Keefe	@kerinokeefe
1	31	This is ripe and fruity, a wine that is smooth...	Avidagos	87	87.0	108	1094	15	Roger Voss	@vossroger
...	...	...	...	...	...	...	...	...	...	...
129969	15	A dry style of Pinot Gris, this is crisp with ...	NaN	90	90.0	11	21	15	Roger Voss	@vossroger
129970	15	Big, rich and off-dry, this is powered by inte...	Lieu-dit Harth Cuvée Caroline	90	90.0	11	21	15	Roger Voss	@vossroger

129971 rows × 13 columns

С использованием метода train_test_split разделите выборку на обучающую и тестовую.

In [358]:
```
from sklearn.model_selection import train_test_split

X = reviews[['country', 'price', 'province', 'region_1', 'variety', 'winery']]
y = reviews['points']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42
)
```

Обучите модель ближайших соседей для произвольно заданного гиперпараметра K. Оцените качество модели с помощью трех подходящих для задачи метрик.

In [359]:
```
from sklearn.neighbors import KNeighborsRegressor
neigh = KNeighborsRegressor(n_neighbors=5)
neigh.fit(X_train, y_train)
```

Out[359]:
```
KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
          metric_params=None, n_jobs=None, n_neighbors=5, p=2,
          weights='uniform')
```

In [360]:
```
y_pred = neigh.predict(X_test)
```

In [361]:
```
y_pred
```

Out[361]:
```
array([88. , 87. , 86.6, ..., 89.4, 87.2, 89.6])
```

In [362]:
```
y_test.values
```

Out[362]:
```
array([83, 85, 86, ..., 89, 91, 91])
```

In [363]:
```
# пусть первая метрика - максимальный модуль разности
metric1 = max(abs(y_test.values[k] - v) for k, v in enumerate(y_pred))
print("Метрика №1:", metric1)

# пусть вторая метрика - самопальная mean_squared_error
metric2 = sum(pow(y_test.values[k] - v, 2) for k, v in enumerate(y_pred)) / len(y_pred)
print("Метрика №2:", metric2)
```

```
Метрика №1: 10.0
Метрика №2: 4.792521926450363
```

Постройте модель и оцените качество модели с использованием кросс-валидации. Проведите эксперименты с тремя различными стратегиями кросс-валидации.

In [ ]:

In [364]:
```
from sklearn.model_selection import cross_val_score
estimator = neigh
scores = cross_val_score(estimator, X_train, y_train, cv=10) # 10 folds by Stratified)KF
old
scores
```

Out[364]:
```
array([0.46893334, 0.46132663, 0.4546627 , 0.45731714, 0.45293478,
       0.47408317, 0.4616699 , 0.43496419, 0.443433  , 0.45459557])
```

In [365]:
```
print("Точность в первом случае: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
Точность в первом случае: 0.46 (+/- 0.02)
```

In [366]:
```
from sklearn.model_selection import ShuffleSplit
cv = ShuffleSplit(n_splits=10, test_size=0.15, random_state=3)
scores2 = cross_val_score(estimator, X_train, y_train, cv=cv)
scores2
```

Out[366]:
```
array([0.45391062, 0.44961495, 0.45048036, 0.44908869, 0.44108992,
       0.42965588, 0.44428762, 0.44557639, 0.43150089, 0.45315958])
```

In [367]:
```
print("Точность во втором случае: %0.2f (+/- %0.2f)" % (scores2.mean(), scores2.std() *
2))
```

```
Точность во втором случае: 0.44 (+/- 0.02)
```

Произведите подбор гиперпараметра K с использованием GridSearchCV и кросс-валидации.

In [368]:
```
from sklearn.model_selection import GridSearchCV
```

In [369]:
```
grid_search = GridSearchCV(estimator, cv=5, param_grid={'n_neighbors': [1, 2, 3]})
grid_search.fit(X_train, y_train)
y_pred2 = grid_search.predict(X_test)
```

In [370]:
```
# пусть первая метрика - максимальный модуль разности
metric1 = max(abs(y_test.values[k] - v) for k, v in enumerate(y_pred2))
print("Метрика №1 для найденного параметра:", metric1)

# пусть вторая метрика - самопальная mean_squared_error
metric2 = sum(pow(y_test.values[k] - v, 2) for k, v in enumerate(y_pred2)) / len(y_pred2
)
print("Метрика №2 для найденного параметра:", metric2)
```

```
Метрика №1 для найденного параметра: 12.0
Метрика №2 для найденного параметра: 4.034966917987383
```

In [ ]:

Лабораторная работа №5. Линейные модели, SVM и деревья решений.

Водка Игорь, ИУ5-61

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

In [101]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [102]:
```
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split

data = pd.DataFrame(load_diabetes().data)
target = pd.DataFrame(load_diabetes().target)
data.columns = load_diabetes().feature_names
data.head()
```

Out[102]:

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019908	-0.017646
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068330	-0.092204
2	0.085299	0.050680	0.044451	-0.005671	-0.045599	-0.034194	-0.032356	-0.002592	0.002864	-0.025930
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022692	-0.009362
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031991	-0.046641

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

In [103]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [104]:
```
sc1 = MinMaxScaler()
for i in data.columns:
    data[i] = sc1_data = sc1.fit_transform(pd.DataFrame(data[i]))
data.head()
```

Out[104]:

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
0	0.666667	1.0	0.582645	0.549296	0.294118	0.256972	0.207792	0.282087	0.562217	0.439394
1	0.483333	0.0	0.148760	0.352113	0.421569	0.306773	0.623377	0.141044	0.222443	0.166667
2	0.883333	1.0	0.516529	0.436620	0.289216	0.258964	0.246753	0.282087	0.496584	0.409091
3	0.083333	0.0	0.301653	0.309859	0.495098	0.447211	0.233766	0.423131	0.572936	0.469697
4	0.516667	0.0	0.206612	0.549296	0.465686	0.417331	0.389610	0.282087	0.362369	0.333333

С использованием метода train_test_split разделите выборку на обучающую и тестовую.

In [105]:
```
train_data, test_data, train_target, test_target = train_test_split(data, target, random
_state=42)
train_data.head()
```

Out[105]:

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
16	0.466667	0.0	0.508264	0.661972	0.539216	0.291833	0.623377	0.141044	0.686869	0.606061
408	0.783333	0.0	0.152893	0.901408	0.563725	0.429283	0.298701	0.382228	0.709045	0.651515
432	0.533333	0.0	0.557851	0.436620	0.656863	0.509960	0.350649	0.380818	0.699975	0.893939
316	0.566667	1.0	0.400826	0.464789	0.455882	0.299801	0.246753	0.423131	0.774234	0.651515
3	0.083333	0.0	0.301653	0.309859	0.495098	0.447211	0.233766	0.423131	0.572936	0.469697

In [106]:
```
from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(train_data, train_target)
```

In [107]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_er
ror, median_absolute_error, r2_score
```

In [108]:
```
def metrics(data, target):
    print("Mean absolute error:", mean_absolute_error(data, target))
    print("Mean squared error:", mean_squared_error(data, target))
    print("Median absolute error:", median_absolute_error(data, target))

metrics(reg.predict(test_data), test_target)
```

```
Mean absolute error: 41.548363283252066
Mean squared error: 2848.295307932943
Median absolute error: 35.207936652961706
```

Обучите 1) одну из линейных моделей, 2) SVM и 3) дерево решений. Оцените качество моделей с помощью трех подходящих для задачи метрик. Сравните качество полученных моделей.

In [109]:
```
from sklearn.svm import SVR
reg = SVR(gamma='auto').fit(train_data, train_target)
```

```
/home/igor-vodka/.local/lib/python3.6/site-packages/sklearn/utils/validation.py:761: Data
ConversionWarning: A column-vector y was passed when a 1d array was expected. Please chan
ge the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [110]:
```
metrics(reg.predict(test_data), test_target)
```

```
Mean absolute error: 62.19052723285024
Mean squared error: 5247.927904086744
Median absolute error: 57.78011108207377
```

In [111]:
```
from sklearn.tree import DecisionTreeRegressor
reg = DecisionTreeRegressor(max_depth=2)
reg.fit(train_data, train_target)
```

Out[111]:
```
DecisionTreeRegressor(criterion='mse', max_depth=2, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best')
```

In [112]:
```
metrics(reg.predict(test_data), test_target)
```

```
Mean absolute error: 47.902763313772496
Mean squared error: 3649.4090253107397
Median absolute error: 39.9816513761468
```

Произведите для каждой модели подбор одного гиперпараметра с использованием GridSearchCV и кросс-валидации.

In [113]:
```
from sklearn.model_selection import GridSearchCV
reg = LinearRegression()
param = {'n_jobs':range(10)}
GV = GridSearchCV(reg, param, cv=3)
GV.fit(train_data, train_target)
GV.best_estimator_
```

Out[113]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=0, normalize=False)
```

In [114]:
```
metrics(GV.predict(test_data), test_target)
```

```
Mean absolute error: 41.548363283252066
Mean squared error: 2848.295307932943
Median absolute error: 35.207936652961706
```

In [115]:
```
reg = SVR(gamma='auto')
param = {'degree':range(1,10)}
GV = GridSearchCV(reg, param, cv=3)
GV.fit(train_data, train_target[0])
```

Out[115]:
```
GridSearchCV(cv=3, error_score='raise-deprecating',
       estimator=SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='aut
o',
  kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False),
       fit_params=None, iid='warn', n_jobs=None,
       param_grid={'degree': range(1, 10)}, pre_dispatch='2*n_jobs',
       refit=True, return_train_score='warn', scoring=None, verbose=0)
```

In [116]:
```
GV.best_estimator_
```

Out[116]:
```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=1, epsilon=0.1, gamma='auto',
  kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

In [117]:
```
metrics(GV.predict(test_data), test_target)
```

```
Mean absolute error: 62.19052723285024
Mean squared error: 5247.927904086744
Median absolute error: 57.78011108207377
```

In [118]:
```
reg = DecisionTreeRegressor( )
param = {'max_depth':range(1,10)}
GV = GridSearchCV(reg, param, cv=3)
GV.fit(train_data, train_target)
```

```
/home/igor-vodka/.local/lib/python3.6/site-packages/sklearn/model_selection/_search.py:84
1: DeprecationWarning: The default of the `iid` parameter will change from True to False
in version 0.22 and will be removed in 0.24. This will change numeric results when test-s
et sizes are unequal.
  DeprecationWarning)
```

Out[118]:
```
GridSearchCV(cv=3, error_score='raise-deprecating',
       estimator=DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=Non
e,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best'),
       fit_params=None, iid='warn', n_jobs=None,
       param_grid={'max_depth': range(1, 10)}, pre_dispatch='2*n_jobs',
       refit=True, return_train_score='warn', scoring=None, verbose=0)
```

In [119]:
```
GV.best_estimator_
```

Out[119]:
```
DecisionTreeRegressor(criterion='mse', max_depth=2, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best')
```

In [120]:
```
metrics(GV.predict(test_data), test_target)
```

```
Mean absolute error: 47.902763313772496
Mean squared error: 3649.4090253107397
Median absolute error: 39.9816513761468
```

In [121]:
```
reg = LinearRegression().fit(train_data, train_target)
reg.fit(train_data, train_target)
metrics(reg.predict(test_data), test_target)
```

```
Mean absolute error: 41.548363283252066
Mean squared error: 2848.295307932943
Median absolute error: 35.207936652961706
```

In [122]:
```
reg = SVR(degree=1, gamma='auto').fit(train_data, train_target)
reg.fit(train_data, train_target)
metrics(reg.predict(test_data), test_target)
```

```
Mean absolute error: 62.19052723285024
Mean squared error: 5247.927904086744
Median absolute error: 57.78011108207377

/home/igor-vodka/.local/lib/python3.6/site-packages/sklearn/utils/validation.py:761: Data
ConversionWarning: A column-vector y was passed when a 1d array was expected. Please chan
ge the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
/home/igor-vodka/.local/lib/python3.6/site-packages/sklearn/utils/validation.py:761: Data
ConversionWarning: A column-vector y was passed when a 1d array was expected. Please chan
ge the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [123]:
```
reg = DecisionTreeRegressor(max_depth=3)
reg.fit(train_data, train_target)
mean_absolute_error(reg.predict(test_data), test_target)
```

Out[123]:
```
47.34495703928109
```

Повторите пункт 4 для найденных оптимальных значений гиперпараметров. Сравните качество полученных моделей с качеством моделей, полученных в пункте 4.

Ну, стало неплохо!

In [ ]:
```

```

Лабораторная работа №6

Водка Игорь, ИУ5-61

Ансамбли моделей машинного обучения.

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.

Ладно, давайте вернёмся к нашему любимому датасету. ../lab3/winemag-data-130k-v2.csv

In [82]:
```
# read the data
import pandas as pd
reviews = pd.read_csv("../lab3/winemag-data-130k-v2.csv", index_col=0)
pd.set_option('max_rows', 5)

for column in ["country", "region_1", "region_2"]:
    reviews[column] = reviews[column].fillna("Unknown")

reviews['price'] = reviews.groupby('country').transform(lambda x: x.fillna(x.mean()))
```

In [83]:
```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

# country ok, winery ok
for feature in ['country', 'province', 'region_1', 'region_2', 'variety', 'winery']:
    le = LabelEncoder()
    reviews[feature] = reviews[feature].dropna()
    processed = pd.DataFrame({'result': reviews[feature]})
    reviews[feature] = le.fit_transform(processed['result'].astype(str))
```

In [84]:
```
reviews
```

Out[84]:

	country	description	designation	points	price	province	region_1	region_2	taster_name	taster_twitter_handle
0	22	Aromas include tropical fruit, broom, brimston...	Vulkà Bianco	87	87.0	331	424	15	Kerin O’Keefe	@kerinokeefe
1	31	This is ripe and fruity, a wine that is smooth...	Avidagos	87	87.0	108	1094	15	Roger Voss	@vossroger
...	...	...	...	...	...	...	...	...	...	...
129969	15	A dry style of Pinot Gris, this is crisp with ...	NaN	90	90.0	11	21	15	Roger Voss	@vossroger
129970	15	Big, rich and off-dry, this is powered by inte...	Lieu-dit Harth Cuvée Caroline	90	90.0	11	21	15	Roger Voss	@vossroger

129971 rows × 13 columns

In [85]:
```
from sklearn.model_selection import train_test_split

X = reviews[['country', 'price', 'province', 'region_1', 'variety', 'winery']]
y = reviews['points']
train_data, test_data, train_target, test_target = train_test_split(X, y, test_size=0.1,
 random_state=42)
```

Обучите две ансамблевые модели. Оцените качество моделей с помощью одной из подходящих для задачи метрик. Сравните качество полученных моделей.

In [86]:
```
from sklearn.ensemble import BaggingRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_squared_log_er
ror, median_absolute_error, r2_score
```

In [87]:
```
base = LinearRegression()
reg = BaggingRegressor(base_estimator = base)
reg.fit(train_data, train_target)
```

Out[87]:
```
BaggingRegressor(base_estimator=LinearRegression(copy_X=True, fit_intercept=True, n_jobs=
None,
         normalize=False),
         bootstrap=True, bootstrap_features=False, max_features=1.0,
         max_samples=1.0, n_estimators=10, n_jobs=None, oob_score=False,
         random_state=None, verbose=0, warm_start=False)
```

Возьмём полюбившуюся функцию из прошлой лабораторной работы:

In [88]:
```
def metrics(data, target):
    print("Mean absolute error:", mean_absolute_error(data, target))
    print("Mean squared error:", mean_squared_error(data, target))
    print("Median absolute error:", median_absolute_error(data, target))
```

In [89]:
```
metrics(reg.predict(test_data), test_target)
```

```
Mean absolute error: 2.276448165395061e-13
Mean squared error: 7.267538121100085e-26
Median absolute error: 2.1316282072803006e-13
```

In [90]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [91]:
```
reg = RandomForestRegressor(max_depth=2, random_state=0, n_estimators=100)
reg.fit(train_data, train_target)
```

Out[91]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=2,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=None,
           oob_score=False, random_state=0, verbose=0, warm_start=False)
```

In [92]:
```
metrics(reg.predict(test_data), test_target)
```

```
Mean absolute error: 0.7349767393856221
Mean squared error: 0.9189615236579426
Median absolute error: 0.9655417152699357
```

Произведите для каждой модели подбор значений одного гиперпараметра. В зависимости от используемой библиотеки можно применять функцию GridSearchCV, использовать перебор параметров в цикле, или использовать другие методы.

In [93]:
```
from sklearn.model_selection import GridSearchCV
```

Довольно долго:

In [94]:
```
reg = RandomForestRegressor(random_state=0, n_estimators=30)
param = {'max_depth':range(1,10)}
GV = GridSearchCV(reg, param, cv=3)
GV.fit(train_data, train_target)
```

Out[94]:
```
GridSearchCV(cv=3, error_score='raise-deprecating',
       estimator=RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=30, n_jobs=None,
           oob_score=False, random_state=0, verbose=0, warm_start=False),
       fit_params=None, iid='warn', n_jobs=None,
       param_grid={'max_depth': range(1, 10)}, pre_dispatch='2*n_jobs',
       refit=True, return_train_score='warn', scoring=None, verbose=0)
```

In [95]:
```
GV.best_estimator_
```

Out[95]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=8,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=30, n_jobs=None,
           oob_score=False, random_state=0, verbose=0, warm_start=False)
```

In [96]:
```
reg = RandomForestRegressor(max_depth=3, random_state=0, n_estimators=30)
reg.fit(train_data, train_target)

prediction = reg.predict(test_data)
metrics(prediction, test_target)
```

```
Mean absolute error: 0.40311575864292687
Mean squared error: 0.2554229628905424
Median absolute error: 0.42376576783307485
```

Качество моделей подросло! Оно стало очень высоким.

In [97]:
```
print("Prediction =", len(prediction))
print("Test target =", len(test_target))
```

```
Prediction = 12998
Test target = 12998
```

In [98]:
```
comparison = pd.DataFrame(
    {
        "predictions": prediction,
        "real target": test_target
    }
)

comparison['diff'] = comparison.apply(lambda row: abs(row['predictions'] - row['real tar
get']), axis=1)
comparison = comparison.sort_values('diff')
```

In [99]:
```
comparison.head()
```

Out[99]:

	predictions	real target	diff
27523	88.0	88	0.0
4253	88.0	88	0.0
17168	89.0	89	0.0
42835	89.0	89	0.0
41600	88.0	88	0.0

In [100]:
```
comparison.tail()
```

Out[100]:

	predictions	real target	diff
116141	94.620707	99	4.379293
114973	94.620707	99	4.379293
60880	94.620707	99	4.379293
7335	94.620707	100	5.379293
123545	94.620707	100	5.379293

Московский государственный технический университет

им. Н.Э. Баумана

УТВЕРЖДАЮ:

Гапанюк Ю.Е.

"__"_____________2019 г.

Курсовая работа по курсу «Технологии машинного обучения»

Пояснительная записка

(вид документа)

писчая бумага

(вид носителя)

20

(количество листов)

ИСПОЛНИТЕЛИ:

студент группы ИУ5-61 ______________________

Водка И.Э.

"__"_____________ 2019 г.

Москва – 2019

Содержание

1. Задание:

1.1. Общее задание

Схема типового исследования, проводимого студентом в рамках курсовой работы, содержит выполнение следующих шагов:

- Поиск и выбор набора данных для построения моделей машинного обучения. На основе выбранного набора данных студент должен построить модели машинного обучения для решения или задачи классификации, или задачи регрессии.
- Проведение разведочного анализа данных. Построение графиков, необходимых для понимания структуры данных. Анализ и заполнение пропусков в данных.
- Выбор признаков, подходящих для построения моделей. Кодирование категориальных признаков Масштабирование данных. Формирование вспомогательных признаков, улучшающих качество моделей.
- Проведение корреляционного анализа данных. Формирование промежуточных выводов о возможности построения моделей машинного обучения. В зависимости от набора данных, порядок выполнения пунктов 2, 3, 4 может быть изменен.
- Выбор метрик для последующей оценки качества моделей. Необходимо выбрать не менее трех метрик и обосновать выбор.
- Выбор наиболее подходящих моделей для решения задачи классификации или регрессии. Необходимо использовать не менее пяти моделей, две из которых должны быть ансамблевыми.
- Формирование обучающей и тестовой выборок на основе исходного набора данных.
- Построение базового решения (baseline) для выбранных моделей без подбора гиперпараметров. Производится обучение моделей на основе обучающей выборки и оценка качества моделей на основе тестовой выборки.
- Подбор гиперпараметров для выбранных моделей. Рекомендуется использовать методы кросс-валидации. В зависимости от используемой библиотеки можно применять функцию GridSearchCV, использовать перебор параметров в цикле, или использовать другие методы.
- Повторение пункта 8 для найденных оптимальных значений гиперпараметров. Сравнение качества полученных моделей с качеством baseline-моделей.
- Формирование выводов о качестве построенных моделей на основе выбранных метрик. Результаты сравнения качества рекомендуется отобразить в виде графиков и сделать выводы в форме текстового описания. Рекомендуется постройение графиков обучения и валидации, влияния значений гиперпарметров на качество моделей и т.д.

Приведенная схема исследования является рекомендуемой. Возможно выполнение курсовой работы на нестандартную тему, которая должна быть предварительно согласована с ответственным за прием курсовой работы.

1.2. Задание данной курсовой работы

Для своей курсовой работы я выбрал нестандартную тему. Причины такого поступка:

- Требования, изложенные выше, характерны как для данной курсовой работы, так и для лабораторных работ, выполненных ранее.
- Тем не менее, многие интересные аспекты работы с технологиями машинного обучения не были рассмотрены в практических работах в данном семестре.
- Значит, нет проблемы в том, чтобы использовать собственный вариант выполнения курсовой работы, т.к. третий раз закреплять одно и то же – хорошо, но рассмотреть что-то новое – тоже неплохо.

Соответственно, задание курсовой работы было придумано самостоятельно и выглядит следующим образом:

Необходимо разработать систему, предварительно обученную на обучающей выборке и определяющую по фотографии, сделана ли фотография в городе России или Великобритании. Т.е., классификатор.

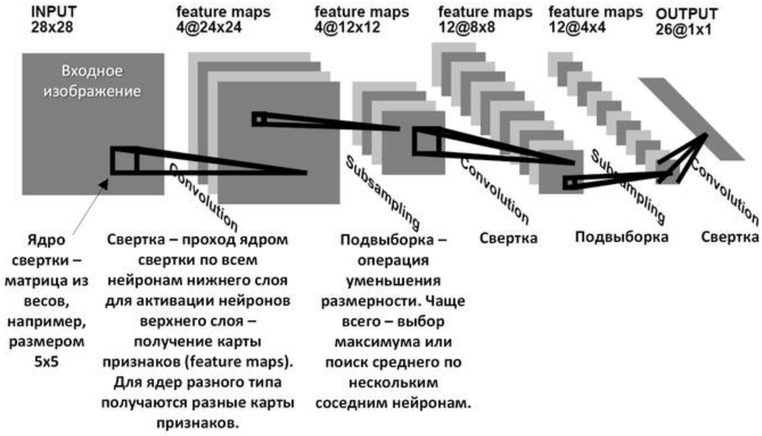
Используемые технологии:

- Node.JS для получения выборки фотографий из Google Street View;
- Python (Anaconda) + Jupyter Notebook;
- Keras (сверточная нейронная сеть)
- прочие инструменты (Matplotlib, Pandas, Numpy, Sklearn...)

2. Введение

Задача обработки изображений – весьма сложная, но с ней неплохо справляются **свёрточные нейронные сети**.

Цитата из Википедии:



Свёрточная нейронная сеть *(англ. convolutional neural network, CNN)* — специальная архитектура искусственных нейронных сетей, предложенная Яном Лекуном в 1988 году и нацеленная на эффективное распознавание образов, входит в состав технологий глубокого обучения (англ. deep learning). Использует некоторые особенности зрительной коры, в которой были открыты так называемые простые клетки, реагирующие на прямые линии под разными углами, и сложные клетки, реакция которых связана с активацией определённого набора простых клеток. Таким образом, идея свёрточных нейронных сетей заключается в чередовании свёрточных слоёв (англ. *convolution layers*) и субдискретизирующих слоёв (англ. *subsampling layers* или англ. *pooling layers*, слоёв подвыборки). Структура сети — однонаправленная (без обратных связей), принципиально многослойная. Для обучения используются стандартные методы, чаще всего метод обратного распространения ошибки. Функция активации нейронов (передаточная функция) — любая, по выбору исследователя.

Название архитектура сети получила из-за наличия операции свёртки, суть которой в том, что каждый фрагмент изображения умножается на матрицу (ядро) свёртки поэлементно, а результат суммируется и записывается в аналогичную позицию выходного изображения.

Работу в этой курсовой работе я выполнял на “инженерном” уровне, не сильно вдаваясь в математические подробности, однако основы архитектуры я рассмотрел.

Подробнее о выполнении работы – в основной части данного документа.

Ещё один важный этап выполнения – **подготовить хороший датасет**. Вместо использования готового датасета я написал небольшой скрипт на Node.JS, загружающий изображения с Google Street View. Далее эти данные следовало обработать (описывается в первой части основного раздела).

Наконец, после завершения работы нейронной сети были сделаны **выводы с графиками и небольшими расчётами**.

3. Основная часть

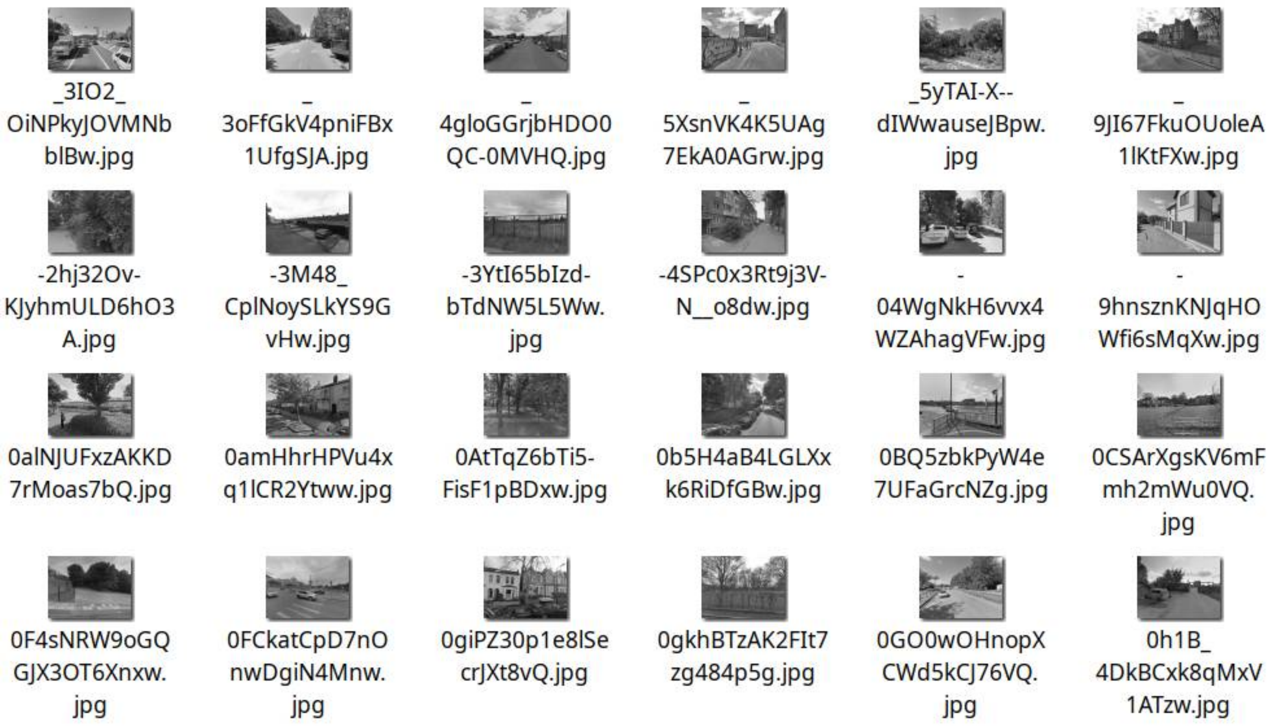
3.1. Подготовка датасета

```
const request = require('request-promise');
const { execSync } = require('child_process');
const fs = require('fs');

const mkdirp = require('mkdirp-promise');
const sharp = require('sharp');

const IMAGE_SIZE = 400;
const ATTEMPTS = 10;
const ROUND_RATIO = 100000;

const GENERATE_TEST = true;

// Генерирует случайные координаты в прямоугольнике
const randomCoordsInSquare = ({lat1, lng1, lat2, lng2}) => {
  const minLat = Math.min(lat1, lat2);
  const maxLat = Math.max(lat1, lat2);
  const minLng = Math.min(lng1, lng2);
  const maxLng = Math.max(lng1, lng2);

  const lat = minLat + Math.random() * (maxLat - minLat);
  const lng = minLng + Math.random() * (maxLng - minLng);

  return { lat: Math.round(lat * ROUND_RATIO) / ROUND_RATIO, lng: Math.round(lng
* ROUND_RATIO) / ROUND_RATIO };
};

// Находит картинку в “прямоугольнике” координат
const findPictureForSquare = async (square, country) => {
  const makeUri = (lat, lng) => {
    return [

'https://maps.googleapis.com/maps/api/js/GeoPhotoService.SingleImageSearch',
        `?pb=!1m5!1sapiv3!5sUS!11m2!1m1!1b0!2m4!1m2!3d${lat}!4d${lng}!2d100!
3m18!2m2!1sru!2sRU!`,
        '9m1!1e2!11m12!1m3!1e2!2b1!3e2!1m3!1e3!2b1!3e2!1m3!1e10!2b1!3e2!4m6!1e1!
1e2!1e3!1e4!1e8!1e6',
        '&callback=respond'
      ].join('');
  };


  let downloaded = false, attemptsRemaining = ATTEMPTS, city, panoId;

  while(!downloaded && attemptsRemaining > 0) {
    const coords = randomCoordsInSquare(square);
    const uri = makeUri(coords.lat, coords.lng);
    const result = await request.get(uri);

    const respond = async (results) => {
      if (results.length === 1 || !Array.isArray(results[1][3][2])) {
        console.error('Attempt #' + (ATTEMPTS - attemptsRemaining + 1)
          + ' failed! ' + JSON.stringify(coords));
        downloaded = false;
        attemptsRemaining--;
        await execSync('sleep 0.05');
      } else {
        console.log('Success!');
```

```
        city = results[1][3][2][0][0];
        panoId = results[1][1][1];
        downloaded = true;
      }
    };

    eval(result);
  }

  if (!downloaded) {
    console.error('Failed!');
    return;
  }

  const rawDir = `${__dirname}/raw/${panoId}`;
  await mkdirp(rawDir);
  execSync(`google_streetview --pano=${panoId} -size=${IMAGE_SIZE}x${IMAGE_SIZE}
--save_downloads=${rawDir}`);

  const newPath = `${__dirname}/` + GENERATE_TEST ? 'images_test' :
'images_train';
  await mkdirp(newPath);

  sharp(`${rawDir}/gsv_0.jpg`)
    .resize({ height: Math.round(IMAGE_SIZE * 0.75), width: IMAGE_SIZE })
    .grayscale()
    .toFile(`${newPath}/${panoId}.jpg`);

  const line = [panoId, country].join(',');
  fs.appendFileSync(GENERATE_TEST ? 'dataset_test.csv' : 'dataset_train.csv',
line + "\n");
  console.log(line);
};

// В этом словаре – список мест, из которых берутся фотографии
const places = {
  moscow: {
    count: 10,
    coords: {lat1: 55.894966, lng1: 37.382917, lat2: 55.610124, lng2:
37.819175},
    country: 'ru',
  },
  spb: {
    count: 5,
    coords: {lat1: 60.039628, lng1: 30.145742, lat2: 59.881093, lng2:
30.537374},
    country: 'ru',
  },
  ekb: {
    count: 3,
    coords: {lat1: 56.862860, lng1: 60.538415, lat2: 56.822548, lng2:
60.688518},
    country: 'ru',
  },
  nino: {
    count: 3,
    coords: {lat1: 56.333229, lng1: 43.898472, lat2: 56.277297, lng2:
44.087820},
    country: 'ru',
  },
  kazan: {
    count: 3,
    coords: {lat1: 55.880916, lng1: 49.043136, lat2: 55.744434, lng2:
49.238918},
```

```
    country: 'ru',
  },
  krasnodar: {
    count: 3,
    coords: {lat1: 45.067626, lng1: 38.936263, lat2: 45.014970, lng2:
39.047241},
    country: 'ru',
  },

  london: {
    count: 10,
    coords: {lat1: 51.548547, lng1: -0.207508, lat2: 51.439893, lng2: 0.029800},
    country: 'uk'
  },
  birmingham: {
    count: 5,
    coords: {lat1: 52.538597, lng1: -1.991987, lat2: 52.424620, lng2: -
1.782344},
    country: 'uk',
  },
  liverpool: {
    count: 3,
    coords: {lat1: 53.449859, lng1: -2.999910, lat2: 53.359085, lng2: -
2.822562},
    country: 'uk',
  },
  manchester: {
    count: 3,
    coords: {lat1: 53.499370, lng1: -2.287626, lat2: 53.455022, lng2: -
2.191537},
    country: 'uk',
  },
  southampton: {
    count: 3,
    coords: {lat1: 50.953016, lng1: -1.479036, lat2: 50.898672, lng2: -
1.317257},
    country: 'uk',
  },
  bristol: {
    count: 3,
    coords: {lat1: 51.469197, lng1: -2.620845, lat2: 51.448589, lng2: -
2.554304},
    country: 'uk',
  },
};

(async () => {
  Object.entries(places)
    .forEach(async ([placeName, placeData]) => {
      console.log('Handling: ' + placeName);
      for (let i = 0; i < placeData.count; i++) {
        await findPictureForSquare(placeData.coords, placeData.country);
      }
  });
})();
```

После запуска и некоторого ожидания получаем папку с фотографиями:



Также получаем файл “dataset_train.csv” следующего вида:

```
file,country
_kdYaGTQTUV1eqQZmfTDvw,ru
e94B91epyBFaUHMCKsXR7g,uk
BoDz-ngJQjhS_PszrCEvJA,ru
VhdZx6hrKM-VJWSjDQk2mw,uk
HT_VTtPVt7RONFBnRe-rvg,uk
P1dkmyQnUJYxBbf8mfCQ6Q,uk
Pc_R4Bzn6Q5aXG5Yf-tl9A,ru
sKKRClFHoDzPAE82YGGm3g,ru
_JZ0Dh1crToE8JPyRySiuA,uk
CUhnn0Bw6h3FYgQzuaSBnQ,uk
Wg_vkIBuZo8rxqsPpN8K_g,ru
Q_0BMBXJMmQ65GeNJvWqqw,ru
vKlOvyLmqhqBffE3g94DzA,uk
ZEdc0xugROGBQkNM8CJM0A,uk
3JfkUl8SVcAw_qAYa3KS-A,uk
tn4_Cw5Un16LnJDpU5JM9w,uk
Wt7Tgh-aIq69lkYlt2lFQg,ru
…
```

Аналогично поступаем с тестовым датасетом. Наконец, все файлы готовы, и мы можем приступать непосредственно к разработке классификатора.

3.2. Jupyter Notebook

In [2]:

```
from PIL import Image
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

Убедимся, что используются GPU. Так скорость обучения увеличивается в разы.

In [4]:

```
from keras import backend as K
K.tensorflow_backend._get_available_gpus()
```

Out[4]:

```
['/job:localhost/replica:0/task:0/device:GPU:0']
```

Сбор данных из датасета

In [5]:

```
def load_image(dirname, file):
    arr = np.array(Image.open('./' + dirname + '/' + file +
'.jpg').convert('L'))
    return arr.reshape(400, 300, 1)
```

In [6]:

```
dataset = pd.read_csv('./dataset.csv')
dataset['image'] = dataset.apply(lambda x: load_image('images', x.file), axis=1)
dataset['label'] = dataset.apply(lambda x: np.array([x.country == 'ru',
x.country == 'uk']), axis=1)
dataset.iloc[0].image.shape
```

Out[6]:

```
(400, 300, 1)
```

Описание модели глубокого обучения на Keras

In [7]:

```
from keras.models import Sequential
from keras.layers import Dense, Activation, Conv2D, MaxPooling2D,
BatchNormalization, Dropout, Flatten

N_IMAGES = len(dataset)
IMG_W = 400
IMG_H = 300

model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3), use_bias=False, input_shape=(IMG_W,
IMG_H, 1)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
```

```
model.add(Activation('relu'))

model.add(Conv2D(64, kernel_size=(3,3), use_bias=False))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Activation('relu'))

model.add(Conv2D(64, kernel_size=(3,3), use_bias=False))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Activation('relu'))

model.add(Conv2D(128, kernel_size=(3,3), use_bias=False))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Activation('relu'))

model.add(Conv2D(64, kernel_size=(3,3), use_bias=False))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Activation('relu'))

model.add(Conv2D(32, kernel_size=(3,3), use_bias=False))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())
model.add(Activation('relu'))

model.add(Dropout(0.2))
model.add(Flatten())

model.add(Dense(128, use_bias=False))
model.add(BatchNormalization())
model.add(Activation('relu'))

model.add(Dense(2, use_bias=False))
model.add(BatchNormalization())
model.add(Activation('softmax'))
```

In [8]:

```
dataset['image'].iloc[0].shape
```

Out[8]:

```
(400, 300, 1)
```

Обучение модели

Компилируем модель, указывая лосс и используемый оптимизатор (стандартные значения, в общем):

In [9]:

```
from keras import optimizers
from keras import losses

model.compile(loss=losses.mean_squared_error, optimizer='sgd')
```

In [10]:

```
x = np.array(dataset['image'].tolist())
```

```
y = np.array(dataset['label'].tolist())
print 'x shape is', x.shape
print 'y shape is', y.shape
```

```
x shape is (4672, 400, 300, 1)
y shape is (4672, 2)
```

Запускаем непосредственно обучение, применяя k-fold cross validation:

In [11]:

```
from sklearn.model_selection import train_test_split
from keras.callbacks import EarlyStopping, ModelCheckpoint

# set early stopping criteria
patience = 2 # this is the number of epochs with no improvement after which the
training will stop
early_stopping = EarlyStopping(monitor='val_loss', patience=patience, verbose=1)

#define the model checkpoint callback -> this will keep on saving the model as a
physical file
model_checkpoint = ModelCheckpoint('./checkpoint.h5', verbose=1,
save_best_only=True)

n_folds = 7
epochs = 5
batch_size = 10
model_history = []

def fit_and_evaluate(t_x, val_x, t_y, val_y, used_epochs, used_batch_size):
    results = model.fit(t_x, t_y, epochs=used_epochs,
batch_size=used_batch_size, callbacks=[early_stopping, model_checkpoint],
              verbose=1, validation_split=0.1)
    print "Validation Score: ", model.evaluate(val_x, val_y)
    return results

for i in range(n_folds):
    print "Training on fold: ", i+1
    t_x, val_x, t_y, val_y = train_test_split(x, y, test_size=0.1, random_state
= np.random.randint(1,1000, 1)[0])
    model_history.append(fit_and_evaluate(t_x, val_x, t_y, val_y, epochs,
batch_size))
    print "=======" * 12 + "\n\n\n"
```

```
('Training on fold: ', 1)
Train on 3783 samples, validate on 421 samples
Epoch 1/5
3783/3783 [==============================] - 74s 20ms/step - loss: 0.2476 - val_loss: 0.2081

Epoch 00001: val_loss improved from inf to 0.20808, saving model to ./checkpoint.h5
Epoch 2/5
3783/3783 [==============================] - 57s 15ms/step - loss: 0.2168 - val_loss: 0.2008

Epoch 00002: val_loss improved from 0.20808 to 0.20075, saving model to ./checkpoint.h5
Epoch 3/5
3783/3783 [==============================] - 57s 15ms/step - loss: 0.2017 - val_loss: 0.1902

Epoch 00003: val_loss improved from 0.20075 to 0.19018, saving model to ./checkpoint.h5
Epoch 4/5
3783/3783 [==============================] - 57s 15ms/step - loss: 0.2006 - val_loss: 0.1936

Epoch 00004: val_loss did not improve from 0.19018
Epoch 5/5
3783/3783 [==============================] - 57s 15ms/step - loss: 0.1938 - val_loss: 0.1948

Epoch 00005: val_loss did not improve from 0.19018
Epoch 00005: early stopping
468/468 [==============================] - 4s 8ms/step
```

```
('Validation Score: ', 0.1868088143503564)
==================================================================================



('Training on fold: ', 2)
Train on 3783 samples, validate on 421 samples
Epoch 1/5
3783/3783 [==============================] - 59s 16ms/step - loss: 0.1850 - val_loss: 0.1722

Epoch 00001: val_loss improved from 0.19018 to 0.17217, saving model to ./checkpoint.h5
Epoch 2/5
3783/3783 [==============================] - 57s 15ms/step - loss: 0.1756 - val_loss: 0.1914

Epoch 00002: val_loss did not improve from 0.17217
Epoch 3/5
3783/3783 [==============================] - 57s 15ms/step - loss: 0.1727 - val_loss: 0.1947

Epoch 00003: val_loss did not improve from 0.17217
Epoch 00003: early stopping
468/468 [==============================] - 2s 4ms/step
('Validation Score: ', 0.19958432540934309)
==================================================================================



('Training on fold: ', 3)
Train on 3783 samples, validate on 421 samples
Epoch 1/5
3783/3783 [==============================] - 57s 15ms/step - loss: 0.1732 - val_loss: 0.2177

Epoch 00001: val_loss did not improve from 0.17217
Epoch 2/5
3783/3783 [==============================] - 57s 15ms/step - loss: 0.1645 - val_loss: 0.1732

Epoch 00002: val_loss did not improve from 0.17217
Epoch 3/5
3783/3783 [==============================] - 57s 15ms/step - loss: 0.1618 - val_loss: 0.1757

Epoch 00003: val_loss did not improve from 0.17217
Epoch 4/5
3783/3783 [==============================] - 57s 15ms/step - loss: 0.1517 - val_loss: 0.2352

Epoch 00004: val_loss did not improve from 0.17217
Epoch 00004: early stopping
468/468 [==============================] - 2s 4ms/step
('Validation Score: ', 0.21811382275106561)
==================================================================================



('Training on fold: ', 4)
Train on 3783 samples, validate on 421 samples
Epoch 1/5
3783/3783 [==============================] - 57s 15ms/step - loss: 0.1554 - val_loss: 0.1584

Epoch 00001: val_loss improved from 0.17217 to 0.15843, saving model to ./checkpoint.h5
Epoch 2/5
3783/3783 [==============================] - 57s 15ms/step - loss: 0.1477 - val_loss: 0.1578

Epoch 00002: val_loss improved from 0.15843 to 0.15779, saving model to ./checkpoint.h5
Epoch 3/5
3783/3783 [==============================] - 57s 15ms/step - loss: 0.1456 - val_loss: 0.1638

Epoch 00003: val_loss did not improve from 0.15779
Epoch 4/5
3783/3783 [==============================] - 57s 15ms/step - loss: 0.1402 - val_loss: 0.1447

Epoch 00004: val_loss improved from 0.15779 to 0.14469, saving model to ./checkpoint.h5
Epoch 5/5
3783/3783 [==============================] - 57s 15ms/step - loss: 0.1342 - val_loss: 0.1829

Epoch 00005: val_loss did not improve from 0.14469
468/468 [==============================] - 2s 4ms/step
('Validation Score: ', 0.16414279917366484)
==================================================================================



('Training on fold: ', 5)
```

```
Train on 3783 samples, validate on 421 samples
Epoch 1/5
3783/3783 [==============================] - 57s 15ms/step - loss: 0.1423 - val_loss: 0.1678

Epoch 00001: val_loss did not improve from 0.14469
Epoch 2/5
3783/3783 [==============================] - 57s 15ms/step - loss: 0.1349 - val_loss: 0.1755

Epoch 00002: val_loss did not improve from 0.14469
Epoch 3/5
3783/3783 [==============================] - 61s 16ms/step - loss: 0.1337 - val_loss: 0.1778

Epoch 00003: val_loss did not improve from 0.14469
Epoch 00003: early stopping
468/468 [==============================] - 2s 5ms/step
('Validation Score: ', 0.16832360905459803)
==================================================================================



('Training on fold: ', 6)
Train on 3783 samples, validate on 421 samples
Epoch 1/5
3783/3783 [==============================] - 58s 15ms/step - loss: 0.1308 - val_loss: 0.1045

Epoch 00001: val_loss improved from 0.14469 to 0.10448, saving model to ./checkpoint.h5
Epoch 2/5
3783/3783 [==============================] - 58s 15ms/step - loss: 0.1244 - val_loss: 0.1847

Epoch 00002: val_loss did not improve from 0.10448
Epoch 3/5
3783/3783 [==============================] - 58s 15ms/step - loss: 0.1179 - val_loss: 0.1075

Epoch 00003: val_loss did not improve from 0.10448
Epoch 00003: early stopping
468/468 [==============================] - 2s 4ms/step
('Validation Score: ', 0.11283582665471949)
==================================================================================



('Training on fold: ', 7)
Train on 3783 samples, validate on 421 samples
Epoch 1/5
3783/3783 [==============================] - 58s 15ms/step - loss: 0.1185 - val_loss: 0.1024

Epoch 00001: val_loss improved from 0.10448 to 0.10243, saving model to ./checkpoint.h5
Epoch 2/5
3783/3783 [==============================] - 57s 15ms/step - loss: 0.1078 - val_loss: 0.0930

Epoch 00002: val_loss improved from 0.10243 to 0.09300, saving model to ./checkpoint.h5
Epoch 3/5
3783/3783 [==============================] - 57s 15ms/step - loss: 0.1064 - val_loss: 0.1170

Epoch 00003: val_loss did not improve from 0.09300
Epoch 4/5
3783/3783 [==============================] - 58s 15ms/step - loss: 0.1013 - val_loss: 0.1035

Epoch 00004: val_loss did not improve from 0.09300
Epoch 00004: early stopping
468/468 [==============================] - 2s 4ms/step
('Validation Score: ', 0.11498103360844474)
==================================================================================
```

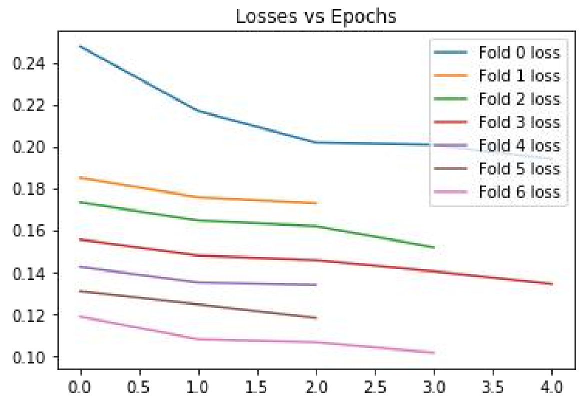
Графики

По горизонтальной оси - номер эпохи, а по вертикальной - показатель loss (средний квадрат ошибки). Здесь видим, что с каждым фолдом loss уменьшается.

In [13]:

```
plt.title('Losses vs Epochs')
```

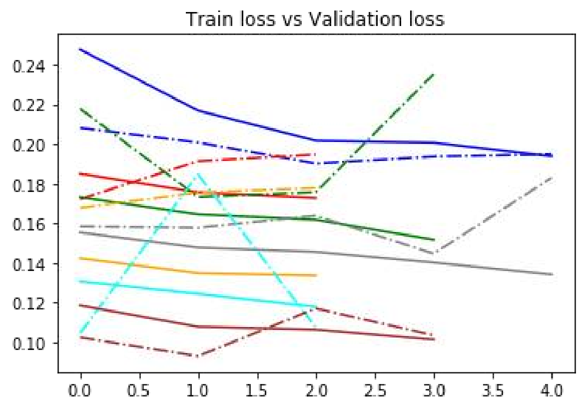
```
for i in range(0, n_folds):
    plt.plot(model_history[i].history['loss'], label='Fold ' + str(i) + ' loss')
plt.legend()
plt.show()
```



Ещё один график - здесь обычными линиями обозначены изменения функции loss, а пунктирными - изменение loss при валидации. Важно, чтобы ошибка при валидации не росла: таким образом мы уменьшаем переобучение.

In [15]:

```
# Validation loss must not rise!
plt.title('Train loss vs Validation loss')

colors = ['blue', 'red', 'green', 'gray', 'orange', 'cyan', 'brown', 'magenta',
'black', 'purple']

for i in range(0, n_folds):
    plt.plot(model_history[i].history['loss'], label='Fold ' + str(i) + ' loss',
color=colors[i])
    plt.plot(model_history[i].history['val_loss'], label='Fold ' + str(i) + '
val loss', color=colors[i], linestyle = "dashdot")

# plt.legend()
plt.show()
```



Результат

In [65]:

```
from matplotlib.font_manager import FontProperties
test_images = pd.read_csv('./dataset_test.csv')
```

```
predictions = []
guesses = 0

font = FontProperties()
font.set_size('large')

for index, row in test_images.iterrows():
    image = row['file']
    real_country = row['country']

    test_image = load_image('images_test', image)
    prediction = model.predict(test_image.reshape(1, 400, 300, 1))
    if prediction[0][0] > 0.50:
        guessed_country = 'ru'
        confidence = prediction[0][0]
    elif prediction[0][1] > 0.50:
        guessed_country = 'uk'
        confidence = prediction[0][1]

    if guessed_country == real_country:
        guesses += 1
        verdict = 'SUCCESS'
        color = 'b'
    else:
        verdict = 'FAILURE'
        color = 'r'

    plt.imshow(Image.open('./images_test/' + image + '.jpg'), cmap="inferno")
    plt.text(
        0,
        -10,
        verdict + "\n" + image + ".jpg\nreal country = " + real_country
             + ",\nguessed country = " + guessed_country
             + ",\nconfidence = " + str(confidence * 100) + "%",
        fontproperties=font,
        color=color
    )
    plt.axis('off')
    plt.show()
```

SUCCESS
TkYmbfuxFWHXT_sxdmvnYA.jpg
real country = uk,
guessed country = uk,
confidence = 0.575528

SUCCESS
3jqtIzQcznOzExfQSh16Hw.jpg
real country = uk,
guessed country = uk,
confidence = 0.961929

SUCCESS
QyMUVdLL2rvc9paP9NqFQg.jpg
real country = ru,
guessed country = ru,
confidence = 0.879323

SUCCESS
W2HBrww7uZGvzU7_4Vm7ug.jpg
real country = uk,
guessed country = uk,
confidence = 0.988308

SUCCESS
-_NClRAiN9dhOy2f-ki2jg.jpg
real country = ru,
guessed country = ru,
confidence = 0.562479

FAILURE
H0fw7ZfPePi7O5mb2AaTnA.jpg
real country = uk,
guessed country = ru,
confidence = 0.915967

SUCCESS
pSnR35sNniFhNQxqVmYHyw.jpg
real country = ru,
guessed country = ru,
confidence = 0.567515

FAILURE
pM0amffa8fkmn6bld9U59Q.jpg
real country = uk,
guessed country = ru,
confidence = 0.98214

FAILURE
4PO-FDhrXycYfhKrMh4FMg.jpg
real country = ru,
guessed country = uk,
confidence = 0.981155

SUCCESS
9lCn-1B8gNEy6j0vNlESJw.jpg
real country = ru,
guessed country = ru,
confidence = 0.696675

FAILURE
0_YxItbAnKqKxZw2v7388A.jpg
real country = ru,
guessed country = uk,
confidence = 0.681674

SUCCESS
NClVN2Tv5Wr8S2D_6C5j4w.jpg
real country = ru,
guessed country = ru,
confidence = 0.600631

[...и так далее…]

In [27]:

```
print 'Угадано:', guesses
print 'Ошибок:', total - guesses
print 'Всего:', total
print 'Точность:', guesses / float(total) * 100, '%'
```

```
Угадано: 83
Ошибок: 25
Всего: 108
Точность: 76.8518518519 %
```

Заключение

Машинное обучение – очень перспективная технология, которая привлекает множество специалистов, инвестиций и пользователей в последнее время. Не обошло оно стороной и нашу кафедру, и всех нас.

Поэтому я не смог обойти его стороной и просто сделать шаблонную курсовую работу. Мне было интересно ощутить, насколько сильны алгоритмы машинного обучения на практике.

Мне действительно понравилось выполнять курсовую работу, и на мой взгляд она неплохо справилась со своей задачей, но её также можно улучшить:

- увеличить обучающую выборку;
- доработать модель глубокого обучения;
- подобрать гиперпараметры, увеличить число эпох и folds;
- закрашивать небо одним цветом, чтобы уменьшить его влияние на оценку;
- и т.д. и т.п.

Список использованных источников

1. Лекции по курсу ТМО (Гапанюк Ю.Е., 2019)
2. Google Street View - https://www.google.com/streetview/
3. Документация Keras - https://keras.io/