МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Кафедра «Систем обработки информации и управления»

ОТЧЕТ

Лабораторная работа №__2__
по курсу «Методы машинного обучения»

Тема: «Изучение библиотек обработки данных»

ИСПОЛНИТЕЛЬ: Цитульский А.М.
ФИО

группа ИУ5-21М ____________________
подпись

"29"____03 ___2019 г.

ПРЕПОДАВАТЕЛЬ: ___Гапанюк Ю.Е.___
ФИО

____
подпись

"__"_________2019 г.

Москва - 2019

Assignment #1 (demo)

Exploratory data analysis with Pandas

In this task you should use Pandas to answer a few questions about the [Adult (https://archive.ics.uci.edu/ml/datasets/Adult)](https://archive.ics.uci.edu/ml/datasets/Adult) dataset. (You don't have to download the data – it's already in the repository). Choose the answers in the [web-form (https://docs.google.com/forms/d/1uY7MpI2trKx6FLWZte0uVh3ULV4Cm_tDud0VDFGCOKg)](https://docs.google.com/forms/d/1uY7MpI2trKx6FLWZte0uVh3ULV4Cm_tDud0VDFGCOKg).

Unique values of all features (for more information, please see the links above):

- `age` : continuous.
- `workclass` : Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
- `fnlwgt` : continuous.
- `education` : Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
- `education-num` : continuous.
- `marital-status` : Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
- `occupation` : Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
- `relationship` : Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
- `race` : White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
- `sex` : Female, Male.
- `capital-gain` : continuous.
- `capital-loss` : continuous.
- `hours-per-week` : continuous.
- `native-country` : United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.
- `salary` : >50K,<=50K

In [0]:
```
import numpy as np
import pandas as pd
pd.set_option('display.max.columns', 100)
# to draw pictures in jupyter notebook
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
# we don't like warnings
# you can comment the following 2 lines if you'd like to
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)

# os.listdir('/content/drive/My Drive/Colab Notebooks/')

# Будем анализировать данные только на обучающей выборке
data = pd.read_csv('/content/drive/My Drive/Colab Notebooks/adult.data.csv', sep=",")
```

```
Mounted at /content/drive
```

In [6]:
```
data.head()
```

Out[6]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

1. How many men and women (*sex* feature) are represented in this dataset?

In [7]:
```
data['sex'].value_counts()
```

Out[7]:
```
Male      21790
Female    10771
Name: sex, dtype: int64
```

2. What is the average age (*age* feature) of women?

In [8]:
```
data.loc[data['sex'] == 'Female', 'age'].mean()
```

Out[8]:
```
36.85823043357163
```

3. What is the percentage of German citizens (*native-country* feature)?

In [9]:
```
float((data['native-country'] == 'Germany').sum()) / data.shape[0]
```

Out[9]:
```
0.004207487485028101
```

4-5. What are the mean and standard deviation of age for those who earn more than 50K per year (*salary* feature) and those who earn less than 50K per year?

In [10]:
```
data.loc[data['salary'] == '<=50K', 'age'].mean()
```

Out[10]:
```
36.78373786407767
```

In [11]:
```
data.loc[data['salary'] == '>50K', 'age'].mean()
```

Out[11]:
```
44.24984058155847
```

6. Is it true that people who earn more than 50K have at least high school education? (*education – Bachelors, Prof-school, Assoc-acdm, Assoc-voc, Masters* or *Doctorate* feature)

In [12]:
```python
data.loc[data['salary'] == '>50K', 'education'].unique()
# No
```

Out[12]:
```
array(['HS-grad', 'Masters', 'Bachelors', 'Some-college', 'Assoc-voc',
       'Doctorate', 'Prof-school', 'Assoc-acdm', '7th-8th', '12th',
       '10th', '11th', '9th', '5th-6th', '1st-4th'], dtype=object)
```

7. Display age statistics for each race (*race* feature) and each gender (*sex* feature). Use *groupby()* and *describe()*. Find the maximum age of men of *Amer-Indian-Eskimo* race.

In [13]:
```python
f = data.loc[(data['race'] == 'Amer-Indian-Eskimo') & (data["sex"] == 'Male'),'age'].max()
print(f)
for (race, sex), sub_df in data.groupby(['race', 'sex']):
    print("Race: {0}, sex: {1}".format(race, sex))
    print(sub_df['age'].describe())
```

```
82
Race: Amer-Indian-Eskimo, sex: Female
count    119.000000
mean      37.117647
std       13.114991
min       17.000000
25%       27.000000
50%       36.000000
75%       46.000000
max       80.000000
Name: age, dtype: float64
Race: Amer-Indian-Eskimo, sex: Male
count    192.000000
mean      37.208333
std       12.049563
min       17.000000
25%       28.000000
50%       35.000000
75%       45.000000
max       82.000000
Name: age, dtype: float64
Race: Asian-Pac-Islander, sex: Female
count    346.000000
mean      35.089595
std       12.300845
min       17.000000
25%       25.000000
50%       33.000000
75%       43.750000
max       75.000000
Name: age, dtype: float64
Race: Asian-Pac-Islander, sex: Male
count    693.000000
mean      39.073593
std       12.883944
min       18.000000
25%       29.000000
50%       37.000000
75%       46.000000
max       90.000000
Name: age, dtype: float64
Race: Black, sex: Female
count    1555.000000
mean       37.854019
std        12.637197
min        17.000000
25%        28.000000
50%        37.000000
75%        46.000000
max        90.000000
Name: age, dtype: float64
Race: Black, sex: Male
count    1569.000000
mean       37.682600
std        12.882612
min        17.000000
25%        27.000000
50%        36.000000
75%        46.000000
max        90.000000
Name: age, dtype: float64
Race: Other, sex: Female
count    109.000000
mean      31.678899
std       11.631599
min       17.000000
25%       23.000000
50%       29.000000
75%       39.000000
max       74.000000
Name: age, dtype: float64
Race: Other, sex: Male
count    162.000000
mean      34.654321
std       11.355531
min       17.000000
25%       26.000000
50%       32.000000
75%       42.000000
max       77.000000
Name: age, dtype: float64
Race: White, sex: Female
count    8642.000000
mean       36.811618
std        14.329093
min        17.000000
25%        25.000000
50%        35.000000
75%        46.000000
max        90.000000
Name: age, dtype: float64
Race: White, sex: Male
count    19174.000000
mean        39.652498
std         13.436029
min         17.000000
25%         29.000000
50%         38.000000
75%         49.000000
max         90.000000
Name: age, dtype: float64
```

8. Among whom is the proportion of those who earn a lot (>50K) greater: married or single men (*marital-status* feature)? Consider as married those who have a *marital-status* starting with *Married* (Married-civ-spouse, Married-spouse-absent or Married-AF-spouse), the rest are considered bachelors.

In [14]:
```
data.loc[(data['sex'] == 'Male') &
     (data['marital-status'].isin(['Never-married',
                                   'Separated',
                                   'Divorced',
                                   'Widowed'])), 'salary'].value_counts()
```

Out[14]:
```
<=50K    7552
>50K      697
Name: salary, dtype: int64
```

In [15]:
```
data.loc[(data['sex'] == 'Male') &
     (data['marital-status'].str.startswith('Married')), 'salary'].value_counts()
```

Out[15]:
```
<=50K    7576
>50K     5965
Name: salary, dtype: int64
```

9. What is the maximum number of hours a person works per week (*hours-per-week* feature)? How many people work such a number of hours, and what is the percentage of those who earn a lot (>50K) among them?

In [16]:
```
max_load = data['hours-per-week'].max()
print("Максимально время - {0} час./неделя.".format(max_load))

num_workaholics = data[data['hours-per-week'] == max_load].shape[0]
print("Усердно работают {0}".format(num_workaholics))

rich_share = float(data[(data['hours-per-week'] == max_load)
                 & (data['salary'] == '>50K')].shape[0]) / num_workaholics
print("Из них богаты {0}%".format(int(100 * rich_share)))
```

```
Максимально время - 99 час./неделя.
Усердно работают 85
Из них богаты 29%
```

10. Count the average time of work (*hours-per-week*) for those who earn a little and a lot (*salary*) for each country (*native-country*). What will these be for Japan?

In [17]:
```
for (country, salary), sub_df in data.groupby(['native-country', 'salary']):
    print(country, salary, round(sub_df['hours-per-week'].mean(), 2))
```

```
? <=50K 40.16
? >50K 45.55
Cambodia <=50K 41.42
Cambodia >50K 40.0
Canada <=50K 37.91
Canada >50K 45.64
China <=50K 37.38
China >50K 38.9
Columbia <=50K 38.68
Columbia >50K 50.0
Cuba <=50K 37.99
Cuba >50K 42.44
Dominican-Republic <=50K 42.34
Dominican-Republic >50K 47.0
Ecuador <=50K 38.04
Ecuador >50K 48.75
El-Salvador <=50K 36.03
El-Salvador >50K 45.0
England <=50K 40.48
England >50K 44.53
France <=50K 41.06
France >50K 50.75
Germany <=50K 39.14
Germany >50K 44.98
Greece <=50K 41.81
Greece >50K 50.62
Guatemala <=50K 39.36
Guatemala >50K 36.67
Haiti <=50K 36.33
Haiti >50K 42.75
Holand-Netherlands <=50K 40.0
Honduras <=50K 34.33
Honduras >50K 60.0
Hong <=50K 39.14
Hong >50K 45.0
Hungary <=50K 31.3
Hungary >50K 50.0
India <=50K 38.23
India >50K 46.48
Iran <=50K 41.44
Iran >50K 47.5
Ireland <=50K 40.95
Ireland >50K 48.0
Italy <=50K 39.62
Italy >50K 45.4
Jamaica <=50K 38.24
Jamaica >50K 41.1
Japan <=50K 41.0
Japan >50K 47.96
Laos <=50K 40.38
Laos >50K 40.0
Mexico <=50K 40.0
Mexico >50K 46.58
Nicaragua <=50K 36.09
Nicaragua >50K 37.5
Outlying-US(Guam-USVI-etc) <=50K 41.86
Peru <=50K 35.07
Peru >50K 40.0
Philippines <=50K 38.07
Philippines >50K 43.03
Poland <=50K 38.17
Poland >50K 39.0
Portugal <=50K 41.94
Portugal >50K 41.5
Puerto-Rico <=50K 38.47
Puerto-Rico >50K 39.42
Scotland <=50K 39.44
Scotland >50K 46.67
South <=50K 40.16
South >50K 51.44
Taiwan <=50K 33.77
Taiwan >50K 46.8
Thailand <=50K 42.87
Thailand >50K 58.33
Trinadad&Tobago <=50K 37.06
Trinadad&Tobago >50K 40.0
United-States <=50K 38.8
United-States >50K 45.51
Vietnam <=50K 37.19
Vietnam >50K 39.2
Yugoslavia <=50K 41.6
Yugoslavia >50K 49.5
```

In [0]:
```
data1 = pd.read_csv('/content/drive/My Drive/Colab Notebooks/user_usage.csv', sep=",")
```

In [0]:
```
data2 = pd.read_csv('/content/drive/My Drive/Colab Notebooks/user_device.csv', sep=",")
```

In [0]:
```
data3 = pd.read_csv('/content/drive/My Drive/Colab Notebooks/android_devices.csv', sep=",")
```

In [21]:
```
data1.head()
```

Out[21]:

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id
0	21.97	4.82	1557.33	22787
1	1710.08	136.88	7267.55	22788
2	1710.08	136.88	7267.55	22789
3	94.46	35.17	519.12	22790
4	71.59	79.26	1557.33	22792

In [22]:
```
data2.head()
```

Out[22]:

	use_id	user_id	platform	platform_version	device	use_type_id
0	22782	26980	ios	10.2	iPhone7,2	2
1	22783	29628	android	6.0	Nexus 5	3
2	22784	28473	android	5.1	SM-G903F	1
3	22785	15200	ios	10.2	iPhone7,2	3
4	22786	28239	android	6.0	ONE E1003	1

In [23]:
```
data3.head()
```

Out[23]:

	Retail Branding	Marketing Name	Device	Model
0	NaN	NaN	AD681H	Smartfren Andromax AD681H
1	NaN	NaN	FJL21	FJL21
2	NaN	NaN	T31	Panasonic T31
3	NaN	NaN	hws7721g	MediaPad 7 Youth 2
4	3Q	OC1020A	OC1020A	OC1020A

In [24]:
```
import time
s_time = time.time()

result = pd.merge(data1,
                  data2[['use_id', 'platform', 'device']],
                  on='use_id')

result.head()
print("--- %s seconds ---" % (time.time() - s_time))
```

```
--- 0.024993896484375 seconds ---
```

Out[24]:

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	platform	device
0	21.97	4.82	1557.33	22787	android	GT-I9505
1	1710.08	136.88	7267.55	22788	android	SM-G930F
2	1710.08	136.88	7267.55	22789	android	SM-G930F
3	94.46	35.17	519.12	22790	android	D2303
4	71.59	79.26	1557.33	22792	android	SM-G361F

In [0]:
```
!pip install pandasql
```

```
Collecting pandasql
  Downloading https://files.pythonhosted.org/packages/6b/c4/ee4096ffa2eeeca0c749b26f0371bd26aa5c8b611c43de99a4f86d3de0a7/pandasql-0.7.3.tar.gz
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from pandasql) (1.14.6)
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-packages (from pandasql) (0.22.0)
Requirement already satisfied: sqlalchemy in /usr/local/lib/python3.6/dist-packages (from pandasql) (1.2.17)
Requirement already satisfied: pytz>=2011k in /usr/local/lib/python3.6/dist-packages (from pandas->pandasql) (2018.9)
Requirement already satisfied: python-dateutil>=2 in /usr/local/lib/python3.6/dist-packages (from pandas->pandasql) (2.5.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2->pandas->pandasql) (1.11.0)
Building wheels for collected packages: pandasql
  Building wheel for pandasql (setup.py) ... done
  Stored in directory: /root/.cache/pip/wheels/53/6c/18/b87a2e5fa8a82e9c026311de56210b8d1c01846e18a9607fc9
Successfully built pandasql
Installing collected packages: pandasql
Successfully installed pandasql-0.7.3
```

In [0]:
```python
import pandasql as ps

def example1_pandasql(data1, data2):
  simple_query = "SELECT * FROM data2 INNER JOIN data1 USING(use_id)"
  return ps.sqldf(simple_query, locals()).set_index('use_id')

t = time.time()
d = example1_pandasql(data2, data1)
print("time of exec: {0}s".format(time.time() - t))
d
```

time of exec: 0.028611183166503906s

Out[0]:

use_id	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	user_id	platform	platform_version	device	use_type_id
22787	21.97	4.82	1557.33	12921	android	4.3	GT-I9505	1
22788	1710.08	136.88	7267.55	28714	android	6.0	SM-G930F	1
22789	1710.08	136.88	7267.55	28714	android	6.0	SM-G930F	1
22790	94.46	35.17	519.12	29592	android	5.1	D2303	1
22792	71.59	79.26	1557.33	28217	android	5.1	SM-G361F	1
22793	71.59	79.26	1557.33	28217	android	5.1	SM-G361F	1
22794	71.59	79.26	519.12	28217	android	5.1	SM-G361F	1
22795	71.59	79.26	519.12	28217	android	5.1	SM-G361F	1
22799	30.92	22.77	3114.67	29643	android	6.0	ONEPLUS A3003	1
22801	69.80	14.70	25955.55	10976	android	4.4	GT-I9505	1
22804	554.41	150.06	3114.67	29645	android	6.0	SM-G935F	1
22805	189.10	24.08	519.12	29646	android	4.2	GT-I9195	1
22806	283.30	107.47	15573.33	21615	android	6.0	A0001	1
22808	324.34	92.52	519.12	29065	android	6.0	SM-G900F	1
22813	797.06	7.67	519.12	23415	android	4.4	HTC Desire 510	1
22814	797.06	7.67	15573.33	23415	android	4.4	HTC Desire 510	1
22815	797.06	7.67	15573.33	23415	android	4.4	HTC Desire 510	1
22816	797.06	7.67	15573.33	23415	android	4.4	HTC Desire 510	1
22817	797.06	7.67	15573.33	23415	android	4.4	HTC Desire 510	1
22819	78.80	327.33	10382.21	29651	android	4.4	HTC One mini 2	1
22820	78.80	327.33	15573.33	29651	android	4.4	HTC One mini 2	1
22822	78.80	327.33	15573.33	29651	android	4.4	HTC One mini 2	1
22823	164.10	192.64	3114.67	29652	android	6.0	SM-G900F	1
22824	208.26	91.76	5191.12	28953	android	6.0	SM-G900F	1
22829	681.44	47.35	1271.39	29653	ios	10.1	iPhone7,2	2
22830	324.27	91.50	519.12	29065	android	6.0	SM-G900F	1
22831	85.97	26.94	407.01	6541	android	4.1	GT-I8190N	1
22832	244.88	105.95	1557.33	29295	android	6.0	D5803	1
22833	135.09	42.02	5191.12	24847	android	6.0	E6653	1
22839	57.49	16.73	15573.33	29655	android	6.0	A0001	1
...	...	...	...	...	...	...	...	...
23002	322.33	86.39	3114.67	28898	android	6.0	SM-G920F	1
23003	124.70	4.64	11.68	29707	android	5.1	HUAWEI CUN-L01	1
23005	37.27	136.10	1557.33	26691	android	6.0	SM-G900F	1
23012	50.68	540.60	650.92	29711	ios	9.3	iPhone6,2	2
23013	28.74	29.52	3114.67	14268	android	6.0	SM-G900F	1
23015	87.76	140.61	1557.33	28945	android	6.0	SM-A300FU	1
23016	99.81	403.78	3114.67	29712	android	6.0	SM-G900F	1
23017	55.96	0.25	2076.45	29666	android	6.0	F3111	1
23018	101.59	84.41	5191.12	29454	android	6.0	Moto G (4)	1
23019	126.30	135.35	519.12	29713	android	5.1	SM-J320FN	1
23020	42.93	124.33	519.12	29714	android	5.1	SM-G361F	1
23021	63.56	26.87	9344.00	28220	android	6.0	SM-G930F	1
23023	157.33	8.87	1557.33	29647	android	7.0	ONEPLUS A3003	1
23024	70.34	18.00	212.64	28637	android	6.0	MotoE2(4G-LTE)	1
23026	532.98	44.36	2076.45	22763	android	6.0	ONE A2003	1
23027	60.08	261.33	12458.67	18108	android	4.4	X11	1
23028	92.52	162.39	1557.33	29716	android	5.1	C6603	1
23029	22.85	34.54	6577.12	29717	android	6.0	HTC One_M8	1
23030	227.13	76.94	0.00	27979	android	5.1	SM-J320FN	1
23031	227.13	76.94	1038.21	27979	android	5.1	SM-J320FN	1
23032	227.13	76.94	1038.21	27979	android	5.1	SM-J320FN	1
23036	57.66	62.85	1557.33	29719	android	5.1	VF-795	1
23039	180.18	17.49	2076.45	29721	android	5.1	SM-G531F	1
23040	12.85	58.32	74.40	29723	android	4.4	HTC Desire 620	1
23041	198.59	90.49	5191.12	28953	android	6.0	SM-G900F	1
23043	198.59	90.49	5191.12	28953	android	6.0	SM-G900F	1
23044	198.59	90.49	3114.67	28953	android	6.0	SM-G900F	1
23046	106.65	82.13	5191.12	29454	android	6.0	Moto G (4)	1
23049	344.53	20.53	519.12	29725	android	6.0	SM-G900F	1
23053	42.75	46.83	5191.12	20257	android	5.1	Vodafone Smart ultra 6	1

159 rows × 8 columns

In [0]:
```
t = time.time()
for (device), desc in result.groupby(['device']):
  print("Device: {0}, Value: {1}".format(device, desc['outgoing_sms_per_month'].max()))
print("time of exec: {0}s".format(time.time() - t))
```

```
Device: A0001, Value: 107.47
Device: C6603, Value: 162.39
Device: D2303, Value: 35.58
Device: D5503, Value: 48.67
Device: D5803, Value: 105.95
Device: D6603, Value: 14.19
Device: E6653, Value: 42.02
Device: EVA-L09, Value: 0.92
Device: F3111, Value: 0.47
Device: GT-I8190N, Value: 26.94
Device: GT-I9195, Value: 89.48
Device: GT-I9300, Value: 159.5
Device: GT-I9505, Value: 253.22
Device: GT-I9506, Value: 26.11
Device: GT-I9515, Value: 61.34
Device: GT-N7100, Value: 91.76
Device: HTC Desire 510, Value: 7.67
Device: HTC Desire 530, Value: 33.97
Device: HTC Desire 620, Value: 58.32
Device: HTC Desire 626, Value: 149.37
Device: HTC Desire 825, Value: 37.06
Device: HTC One M9, Value: 66.65
Device: HTC One S, Value: 150.59
Device: HTC One mini 2, Value: 327.33
Device: HTC One_M8, Value: 34.54
Device: HUAWEI CUN-L01, Value: 4.64
Device: HUAWEI VNS-L31, Value: 22.94
Device: LG-H815, Value: 10.14
Device: Lenovo K51c78, Value: 12.93
Device: Moto G (4), Value: 84.41
Device: MotoE2(4G-LTE), Value: 18.0
Device: Nexus 5X, Value: 15.38
Device: ONE A2003, Value: 44.36
Device: ONEPLUS A3003, Value: 153.35
Device: SM-A300FU, Value: 207.59
Device: SM-A310F, Value: 234.72
Device: SM-A500FU, Value: 138.28
Device: SM-G360F, Value: 69.2
Device: SM-G361F, Value: 124.33
Device: SM-G531F, Value: 17.49
Device: SM-G800F, Value: 47.4
Device: SM-G900F, Value: 403.78
Device: SM-G903F, Value: 52.47
Device: SM-G920F, Value: 435.29
Device: SM-G925F, Value: 11.5
Device: SM-G930F, Value: 136.88
Device: SM-G935F, Value: 274.76
Device: SM-J320FN, Value: 135.35
Device: SM-N9005, Value: 273.75
Device: SM-N910F, Value: 169.32
Device: VF-795, Value: 62.85
Device: Vodafone Smart ultra 6, Value: 46.83
Device: X11, Value: 262.47
Device: iPhone6,2, Value: 540.6
Device: iPhone7,2, Value: 47.35
time of exec: 0.0403900146484375s
```

In [0]:
```python
s_time = time.time()
# pandasql code
def example2_pandasql(d):
    aggr_query = '''
        SELECT MAX(outgoing_sms_per_month)
        FROM d
        GROUP BY device
        '''
    return ps.sqldf(aggr_query, locals())

k = example2_pandasql(d)
print("--- %s seconds ---" % (time.time() - s_time))
k
```

```
--- 0.023077011108398438 seconds ---
```

Out[0]:

	MAX(outgoing_sms_per_month)
0	107.47
1	162.39
2	35.58
3	48.67
4	105.95
5	14.19
6	42.02
7	0.92
8	0.47
9	26.94
10	89.48
11	159.50
12	253.22
13	26.11
14	61.34
15	91.76
16	7.67
17	33.97
18	58.32
19	149.37
20	37.06
21	66.65
22	150.59
23	327.33
24	34.54
25	4.64
26	22.94
27	10.14
28	12.93
29	84.41
30	18.00
31	15.38
32	44.36
33	153.35
34	207.59
35	234.72
36	138.28
37	69.20
38	124.33
39	17.49
40	47.40
41	403.78
42	52.47
43	435.29
44	11.50
45	136.88
46	274.76
47	135.35
48	273.75
49	169.32
50	62.85
51	46.83
52	262.47
53	540.60
54	47.35

In [5]:
```
!ipython nbconvert —to html "/content/drive/My Drive/Colab Notebooks/LR№2.ipynb"
```

```
[TerminalIPythonApp] WARNING | Subcommand `ipython nbconvert` is deprecated and will be removed in future versions.
[TerminalIPythonApp] WARNING | You likely want to use `jupyter nbconvert` in the future
[NbConvertApp] WARNING | pattern u'\u2014to' matched no files
[NbConvertApp] WARNING | pattern u'html' matched no files
[NbConvertApp] Converting notebook /content/drive/My Drive/Colab Notebooks/LR№2.ipynb to html
[NbConvertApp] Writing 345333 bytes to /content/drive/My Drive/Colab Notebooks/LR№2.html
```