Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

Отчет по ЛР №1

по курсу «Технологии машинного обучения»

«Разведочный анализ данных. Исследование и визуализация данных»

ИСПОЛНИТЕЛЬ:
Аушева Л.И.
Группа ИУ5-61Б

_____

"__"_____2020 г.

ПРЕПОДАВАТЕЛЬ:
Гапанюк Ю.Е.

_____

"__"_____2020 г.

Москва 2020

Цель лабораторной работы:

Изучение различных методов визуализации данных.

Задание:

- Выбрать набор данных (датасет: marvel-wikia-data.csv).
- Создать ноутбук, который содержит следующие разделы:
  - Текстовое описание выбранного Вами набора данных.
  - Основные характеристики датасета.
  - Визуальное исследование датасета.
  - Информация о корреляции признаков.

Выполнение:

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

Загрузка данных

In [2]:
```
data = pandas.read_csv('marvel-wikia-data.csv')
```

In [3]:
```
data[:]
```

Out[3]:

	page_id	name	urlslug	ID	ALIGN	EYE	HAIR	SEX	GSM	ALIVE	APPEARANCES	F APPEARA
0	1678	Spider-Man (Peter Parker)	\/Spider-Man_(Peter_Parker)	Secret Identity	Good Characters	Hazel Eyes	Brown Hair	Male Characters	NaN	Living Characters	4043.0	A
1	7139	Captain America (Steven Rogers)	\/Captain_America_(Steven_Rogers)	Public Identity	Good Characters	Blue Eyes	White Hair	Male Characters	NaN	Living Characters	3360.0	M
2	64786	Wolverine (James \"Logan\" Howlett)	\/Wolverine_(James_%22Logan%22_Howlett)	Public Identity	Neutral Characters	Blue Eyes	Black Hair	Male Characters	NaN	Living Characters	3061.0	C
3	1868	Iron Man (Anthony \"Tony\" Stark)	\/Iron_Man_(Anthony_%22Tony%22_Stark)	Public Identity	Good Characters	Blue Eyes	Black Hair	Male Characters	NaN	Living Characters	2961.0	M
4	2460	Thor (Thor Odinson)	\/Thor_(Thor_Odinson)	No Dual Identity	Good Characters	Blue Eyes	Blond Hair	Male Characters	NaN	Living Characters	2258.0	N
...	...	...	...	...	...	...	...	...	...	...	...	
16371	657508	Ru'ach (Earth-616)	\/Ru%27ach_(Earth-616)	No Dual Identity	Bad Characters	Green Eyes	No Hair	Male Characters	NaN	Living Characters	NaN	
16372	665474	Thane (Thanos' son) (Earth-616)	\/Thane_(Thanos%27_son)_(Earth-616)	No Dual Identity	Good Characters	Blue Eyes	Bald	Male Characters	NaN	Living Characters	NaN	
16373	695217	Tinkerer (Skrull) (Earth-616)	\/Tinkerer_(Skrull)_(Earth-616)	Secret Identity	Bad Characters	Black Eyes	Bald	Male Characters	NaN	Living Characters	NaN	
16374	708811	TK421 (Spiderling) (Earth-616)	\/TK421_(Spiderling)_(Earth-616)	Secret Identity	Neutral Characters	NaN	NaN	Male Characters	NaN	Living Characters	NaN	
16375	673702	Yologarch (Earth-616)	\/Yologarch_(Earth-616)	NaN	Bad Characters	NaN	NaN	NaN	NaN	Living Characters	NaN	

16376 rows × 13 columns

In [4]:
```
data.head()
```

Out[4]:

	page_id	name	urlslug	ID	ALIGN	EYE	HAIR	SEX	GSM	ALIVE	APPEARANCES	FIRST APPEARANCE
0	1678	Spider-Man (Peter Parker)	\/Spider-Man_(Peter_Parker)	Secret Identity	Good Characters	Hazel Eyes	Brown Hair	Male Characters	NaN	Living Characters	4043.0	Aug-62
1	7139	Captain America (Steven Rogers)	\/Captain_America_(Steven_Rogers)	Public Identity	Good Characters	Blue Eyes	White Hair	Male Characters	NaN	Living Characters	3360.0	Mar-41
2	64786	Wolverine (James \"Logan\" Howlett)	\/Wolverine_(James_%22Logan%22_Howlett)	Public Identity	Neutral Characters	Blue Eyes	Black Hair	Male Characters	NaN	Living Characters	3061.0	Oct-74
3	1868	Iron Man (Anthony \"Tony\" Stark)	\/Iron_Man_(Anthony_%22Tony%22_Stark)	Public Identity	Good Characters	Blue Eyes	Black Hair	Male Characters	NaN	Living Characters	2961.0	Mar-63
4	2460	Thor (Thor Odinson)	\/Thor_(Thor_Odinson)	No Dual Identity	Good Characters	Blue Eyes	Blond Hair	Male Characters	NaN	Living Characters	2258.0	Nov-50

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16376 entries, 0 to 16375
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   page_id           16376 non-null  int64  
 1   name              16376 non-null  object 
 2   urlslug           16376 non-null  object 
 3   ID                12606 non-null  object 
 4   ALIGN             13564 non-null  object 
 5   EYE               6609 non-null   object 
 6   HAIR              12112 non-null  object 
 7   SEX               15522 non-null  object 
 8   GSM               90 non-null     object 
 9   ALIVE             16373 non-null  object 
 10  APPEARANCES       15280 non-null  float64
 11  FIRST APPEARANCE  15561 non-null  object 
 12  Year              15561 non-null  float64
dtypes: float64(2), int64(1), object(10)
memory usage: 1.6+ MB
```

In [6]:
```
data.shape
```

Out[6]:
```
(16376, 13)
```

In [7]:
```
data.columns
```

Out[7]:
```
Index(['page_id', 'name', 'urlslug', 'ID', 'ALIGN', 'EYE', 'HAIR', 'SEX',
       'GSM', 'ALIVE', 'APPEARANCES', 'FIRST APPEARANCE', 'Year'],
      dtype='object')
```

In [8]:
```
data.dtypes
```

Out[8]:
```
page_id               int64
name                 object
urlslug              object
ID                   object
ALIGN                object
EYE                  object
HAIR                 object
SEX                  object
GSM                  object
ALIVE                object
APPEARANCES         float64
FIRST APPEARANCE     object
Year                float64
dtype: object
```

In [9]:
```
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))
```

```
page_id - 0
name - 0
urlslug - 0
ID - 3770
ALIGN - 2812
EYE - 9767
HAIR - 4264
SEX - 854
GSM - 16286
ALIVE - 3
APPEARANCES - 1096
FIRST APPEARANCE - 815
Year - 815
```

In [10]:
```
data['ALIVE'].value_counts()
```

Out[10]:
```
Living Characters      12608
Deceased Characters     3765
Name: ALIVE, dtype: int64
```

In [11]:
```
data.describe()
```

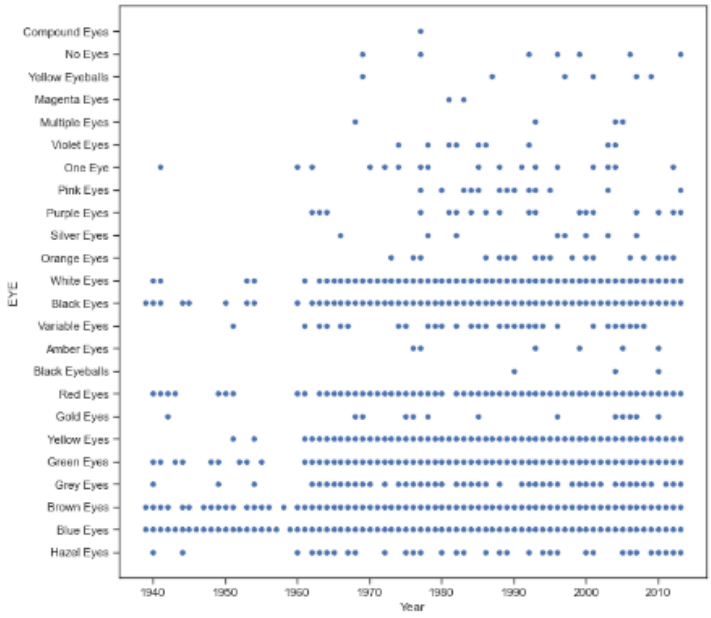
Out[11]:

	page_id	APPEARANCES	Year
count	16376.000000	15280.000000	15561.000000
mean	300232.082377	17.033377	1984.951803
std	253460.403399	96.372959	19.663571
min	1025.000000	1.000000	1939.000000
25%	28309.500000	1.000000	1974.000000
50%	282578.000000	3.000000	1990.000000
75%	509077.000000	8.000000	2000.000000
max	755278.000000	4043.000000	2013.000000

Диаграмма рассеяния

In [12]:
```
fig, ax = plt.subplots(figsize=(10,10))
sns.scatterplot(ax=ax, x='Year', y='EYE', data = data)
```

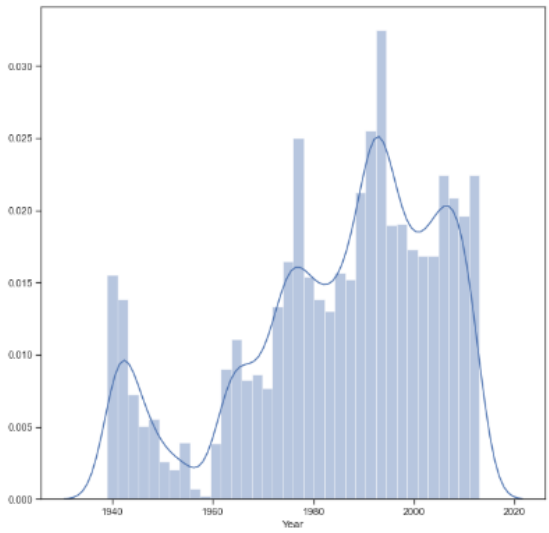
Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x207fe295ca0>
```



Гистограмма

In [13]:
```
fig, ax = plt.subplots(figsize=(10,10))
sns.distplot(data['Year'])
```

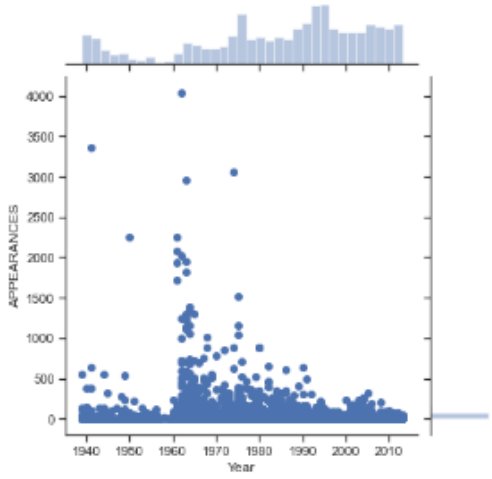
Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x207fdf1d8b0>
```



Jointplot

In [14]:
```
sns.jointplot(x='Year', y='APPEARANCES', data=data)
```

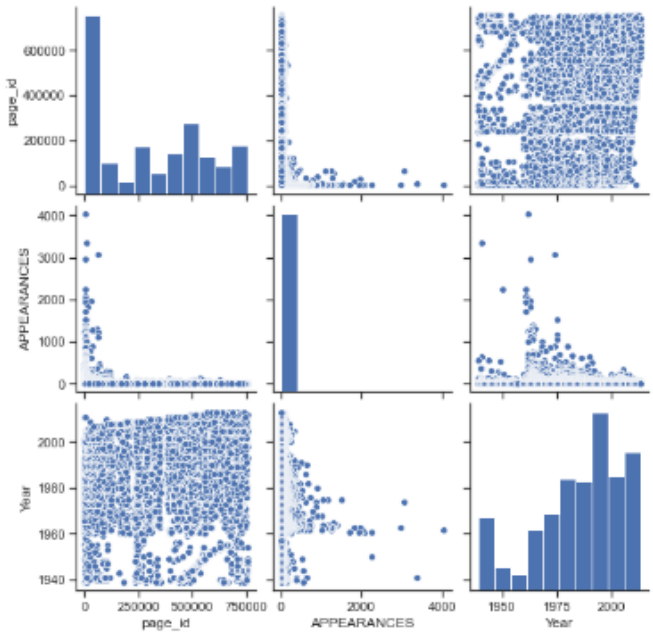
Out[14]:
```
<seaborn.axisgrid.JointGrid at 0x207fdd9ba90>
```



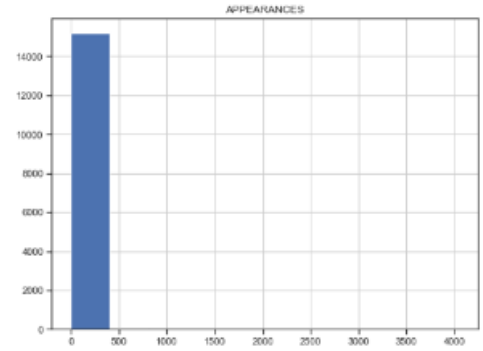
Парные диаграммы

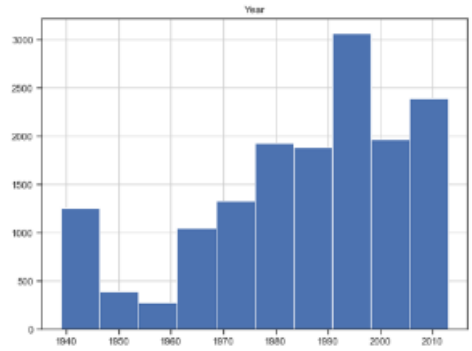
In [15]:
```
sns.pairplot(data)
```

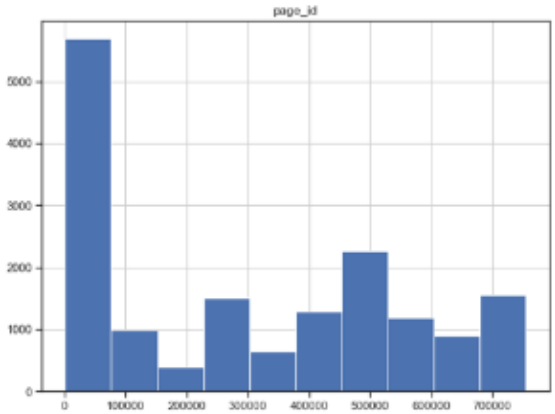
Out[15]:
```
<seaborn.axisgrid.PairGrid at 0x207fdfa42b0>
```



In [16]:
```
data.hist(bins=10, figsize= (20, 15))
plt.show()
```



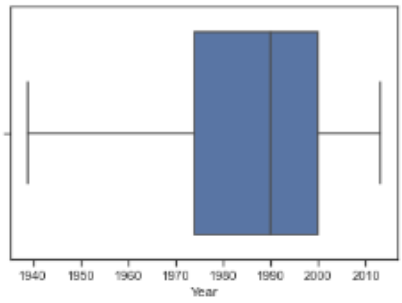




Ящик с усами

```
In [17]: sns.boxplot(x=data['Year'])
```

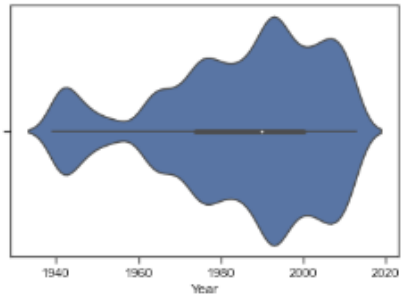
```
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x207fe642df0>
```



Violin plot

```
In [18]: sns.violinplot(x=data['Year'])
```

```
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x207fee5d9a0>
```



Информация о корреляции признаков

```
In [19]: data.corr()
```

Out[19]:

	page_id	APPEARANCES	Year
page_id	1.000000	-0.155810	-0.050244
APPEARANCES	-0.155810	1.000000	-0.082881
Year	-0.050244	-0.082881	1.000000

```
In [20]: data.corr(method='pearson')
```

Out[20]:

	page_id	APPEARANCES	Year
page_id	1.000000	-0.155810	-0.050244
APPEARANCES	-0.155810	1.000000	-0.082881
Year	-0.050244	-0.082881	1.000000

```
In [21]: data.corr(method='kendall')
```

Out[21]:

	page_id	APPEARANCES	Year
page_id	1.000000	-0.351101	0.065434
APPEARANCES	-0.351101	1.000000	0.003614
Year	0.065434	0.003614	1.000000

Вывод:

Изучила различные методы визуализации данных.