

▼ Пет-проект: Хоккей. Число Лемтюгова

▼ Цель проекта:

Создать аналог числа Эрдёша — Бэйкона для отечественных хоккеистов, называемый числом, чтобы измерять "близость" хоккеистов к Николаю Лемтюгову через команды, в которых они играли

Задачи проекта:

1. Собрать данные об отечественных хоккеистах и командах, в которых они играли.
2. Рассчитать число Лемтюгова для каждого хоккеиста.
3. Построить дэшборд или отчет с графами, отображающими связи между хоккеистами и количеством команд, через которые они были связаны.

План:

1. Описание данных
2. Загрузка данных и подготовка их к анализу
 - Общая информация по датафрейму
 - Проверить корректность наименований колонок
 - Исследовать соответствие типов
 - Проверка данных на наличие дубликатов
3. Расчет числа Лемтюгова
4. Визуализация графа
5. Проверка статистических гипотез
6. Выводы

▼ Описание данных

Датасет players2 - информация о переходах игроков:

- player_link - ссылка на команду
- player - имя игрока
- team - команда
- start_date - дата начала
- end_date - дата конца

Датасет stat2_new - статистика по игрокам и играм:

- player_link - ссылка на игрока
- player - имя игрока
- position - игровое амплуа
- born - дата рождения
- age - возраст
- country - страна рождения
- hight - рост
- weight - вес
- shoot - захват клюшки
- GP - количество игр
- G - количество голов
- Assists - количество передач
- PTS - количество очков, присуждаемое в команде
- +/- - очки +/-
- '+' - очки +

- '-' - очки минус

Статистика игроков по амплуа

- PIM - штрафное время
- ESG - шайбы в равенстве
- PPG - шайбы в большинстве
- SHG - шайбы в меньшинстве
- OTG - шайбы в овертайме
- GWG - победные голы
- SDS - решающие броски в булитах
- SOG - броски по воротам
- %SOG - % реализованных
- S/G - среднее количество бросков в ворота
- FO - вбрасывания
- FOW - выигранные вбрасывания
- %FO - % выигранных вбрасываний
- TOI/G - среднее время пребывания на льду/в игре
- SFT/G - среднее количество смен за игру
- TIE/G - среднее время игры на площадке при игре в равных составах за игру
- SFTE/G - среднее количество смен при игре в равных составах за игру
- TIPP/G - среднее время игры на площадке при игре в большинстве за игру
- SFTPP/G - среднее количество смен при игре в большинстве за игру
- TISH/G - среднее время игры на площадке при игре в меньшинстве за игру
- SFTSH/G - среднее количество смен при игре в меньшинстве за игру
- HITS - силовые приемы
- BLS - заблокированные броски
- FOA - фолы против
- TKA - отборы шайбы

для вратарей:

- W - выигрыши
- L - проигрыши
- SOP - игры с буллитными сериями
- SOG - броски
- GA - пропущено шайб
- Sv - отраженные броски
- %Sv - % отраженных бросков
- GAA - коэффициент надежности
- SO - сухие игры

▼ Загрузка данных и подготовка их к анализу

```
!pip install scikit-image --quiet
!pip install datashader --quiet
!pip install holoviews==1.16 --quiet
!pip install bokeh==3.1.1 --quiet
```

```
import pandas as pd
import numpy as np
from datetime import datetime, timedelta as dt
```

```
#Библиотека для парсинга json
import json
```

```
#Основные графические библиотеки
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
```

```
#Библиотека визуализации bokeh и фронтенд для вывода - holoviews
import holoviews as hv
import bokeh
from holoviews import opts, dim
import holoviews.plotting.bokeh
from bokeh.plotting import show, output_file
hv.extension('bokeh')
```

```

#request - чтобы подтягивать информацию с других сайтов - используется в функции отрисовки sankey чарта
import requests

#Для отображения прогресса - используется в функции отрисовки sankey чарта
from tqdm import tqdm

#Библиотека для работы с очередями - для прохода по графу
from queue import Queue

#Для работы с матрицей вместо графа
import scipy
from scipy.sparse import csr_matrix

#Для работы с графами и их визуализации
import networkx as nx

# для гипотез
from scipy import stats as st

```



```

#загрузка данных
players2 = pd.read_csv(
    'https://docs.google.com/spreadsheets/d/e/2PACX-1vT3TtrPT83dPIbAetp5tjhpHm1lTxYjFhJ62lThCeI9yPMeYwMCHPq9J5uoAL-Zcx3HvxMlXEeE18iT/pub?

#загрузка данных
stat2_new = pd.read_csv('https://docs.google.com/spreadsheets/d/e/2PACX-1vT411kwpQk6lUrapB4bnsC1--cUkQMxWieItZ0sfIoYdmsO50Lu_hWws-r_UBfav

```

▼ Общая информация по датафрейму

```
list_data = [players2, stat2_new] #сохраним названия датафреймов в список
```

```
players2.info()#выведем общую информацию
```


```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8179 entries, 0 to 8178
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   player_link  8179 non-null   object
1   player       8179 non-null   object
2   team         8179 non-null   object
3   start_date   8179 non-null   datetime64[ns]
4   end_date     8179 non-null   datetime64[ns]
dtypes: datetime64[ns](2), object(3)
memory usage: 319.6+ KB

```

8179 строк. Пропусков нет

```
players2.head(1)
```

	player_link	player	team	start_date	end_date	
0	https://en.khl.ru/players/16785/?idplayer=16785...	Juhamatti Aaltonen	Jokerit	2014-09-04	2016-03-02	

```
stat2_new.info()#выведем общую информацию
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3717 entries, 0 to 3716
Data columns (total 49 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   player_link  3717 non-null   object
1   player       3717 non-null   object
2   position     3717 non-null   object
3   born         3717 non-null   object
4   age          3717 non-null   object
5   country      3717 non-null   object
6   hight        3717 non-null   int64
7   weight       3717 non-null   int64
8   shoot       3717 non-null   object
9   GP           3717 non-null   int64
10  G            3717 non-null   int64
11  Assists      3717 non-null   int64
12  PTS          3226 non-null   float64

```

```
13 +/-      3226 non-null float64
14 +        3226 non-null float64
15 -        3226 non-null float64
16 PIM      3717 non-null int64
17 ESG      3226 non-null float64
18 PPG      3226 non-null float64
19 SHG      3226 non-null float64
20 OTG      3226 non-null float64
21 GWG      3226 non-null float64
22 SDS      3226 non-null float64
23 SOG      3717 non-null int64
24 %SOG     3226 non-null object
25 S/G      3226 non-null object
26 FO       3226 non-null float64
27 FOW      3226 non-null float64
28 %FO      3226 non-null object
29 TOI/G     3227 non-null object
30 SFT/G     3226 non-null object
31 TIE/G     3226 non-null object
32 SFTE/G   3226 non-null object
33 TIPP/G   3226 non-null object
34 SFTPP/G  3226 non-null object
35 TISH/G   3226 non-null object
36 SFTSH/G  3226 non-null object
37 HITS     3226 non-null float64
38 BLS      3226 non-null float64
39 FOA      3226 non-null float64
40 TkA      3226 non-null float64
41 W        491 non-null float64
42 L        491 non-null float64
43 SOP      491 non-null float64
44 GA       491 non-null float64
45 Sv       491 non-null float64
46 %Sv      491 non-null object
47 GAA      491 non-null object
48 SO       491 non-null float64
dtypes: float64(22), int64(7), object(20)
memory usage: 1.4+ MB
```

3717 строк. Есть пропуски. Информация не внесена, если для амплуа игрока она не предусмотрена

stat2_new.head(1)

	player_link	player	position	born	age	country	hight	weight	shoot	GP	...	FOA	TkA	W	L	SOP
0	https://en.khl.ru/players/16785/?idplayer=1678...	Juhamatti Aaltonen	forward	4-июн.-85	38	Finland	184	89	right	245	...	22.0	0.0	NaN	NaN	NaN



1 rows × 49 columns

Пропуски: количество и в % выражении



#Код ревьюера - смотрим количество и долю пропусков/функция

```
def isna_data(df):
    display(pd.concat(
        [
            df.isna().sum(),
            df.isna().mean().apply('{:.2%}'.format)
        ], axis=1,
        keys=['nan_count', 'nan_share']
    ).sort_values(by='nan_count', ascending=False)
    )
```

isna_data(stat2_new)

	nan_count	nan_share	
SO	3226	86.79%	
GAA	3226	86.79%	
%Sv	3226	86.79%	
Sv	3226	86.79%	
GA	3226	86.79%	
SOP	3226	86.79%	
L	3226	86.79%	
W	3226	86.79%	
SFTPP/G	491	13.21%	
FOW	491	13.21%	
%FO	491	13.21%	
SFT/G	491	13.21%	
TIE/G	491	13.21%	
SFTE/G	491	13.21%	
TIPP/G	491	13.21%	
BLS	491	13.21%	
TISH/G	491	13.21%	
SFTSH/G	491	13.21%	
HITS	491	13.21%	
S/G	491	13.21%	
FOA	491	13.21%	
TkA	491	13.21%	
FO	491	13.21%	
%SOG	491	13.21%	
+/-	491	13.21%	
SHG	491	13.21%	
PTS	491	13.21%	
+	491	13.21%	
-	491	13.21%	
PPG	491	13.21%	
ESG	491	13.21%	
OTG	491	13.21%	
GWG	491	13.21%	
SDS	491	13.21%	
TOI/G	490	13.18%	
weight	0	0.00%	
position	0	0.00%	
born	0	0.00%	
age	0	0.00%	

isna_data(players2)

	nan_count	nan_share	
player_link	0	0.00%	
player	0	0.00%	
team	0	0.00%	
start_date	0	0.00%	
end_date	0	0.00%	

В этом разделе добавим новый столбец link с id игрока, извлеченного из 'player_link', во всех датасетах.

```
players2.info()#выведем общую информацию
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8179 entries, 0 to 8178
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   player_link  8179 non-null   object
1   player       8179 non-null   object
2   team         8179 non-null   object
3   start_date   8179 non-null   datetime64[ns]
4   end_date     8179 non-null   datetime64[ns]
dtypes: datetime64[ns](2), object(3)
memory usage: 319.6+ KB
```

С типом данных все в порядке

```
len(players2['player'].unique())# найдем количество уникальных игроков по имени
```

```
3712
```

```
#выделим id игрока из ссылки на профиль
players2['link'] = players2['player_link'].str.extract(r'/(\\d+)/')
players2.head(1)
```

	player_link	player	team	start_date	end_date	link
0	https://en.khl.ru/players/16785/?idplayer=1678...	Juhamatti Aaltonen	Jokerit	2014-09-04	2016-03-02	16785

```
len(players2['link'].unique())# найдем количество уникальных игроков по id
```

```
3720
```

Получили на 8 игроков больше, в данных есть тетки.

```
stat2_new.info() #выведем общую информацию
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3717 entries, 0 to 3716
Data columns (total 49 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   player_link  3717 non-null   object
1   player       3717 non-null   object
2   position     3717 non-null   object
3   born         3717 non-null   object
4   age          3717 non-null   object
5   country      3717 non-null   object
6   hight        3717 non-null   int64
7   weight       3717 non-null   int64
8   shoot        3717 non-null   object
9   GP           3717 non-null   int64
10  G            3717 non-null   int64
11  Assists      3717 non-null   int64
12  PTS          3226 non-null   float64
13  +/-          3226 non-null   float64
14  +            3226 non-null   float64
15  -            3226 non-null   float64
16  PIM          3717 non-null   int64
17  ESG          3226 non-null   float64
18  PPG          3226 non-null   float64
19  SHG          3226 non-null   float64
20  OTG          3226 non-null   float64
21  GWG          3226 non-null   float64
22  SDS          3226 non-null   float64
23  SOG          3717 non-null   int64
24  %SOG         3226 non-null   object
25  S/G          3226 non-null   object
26  FO           3226 non-null   float64
27  FOW          3226 non-null   float64
28  %FO          3226 non-null   object
29  TOI/G        3227 non-null   object
30  SFT/G        3226 non-null   object
31  TIE/G        3226 non-null   object
32  SFTE/G       3226 non-null   object
33  TIPP/G       3226 non-null   object
```

```

34 SFTPP/G      3226 non-null object
35 TISH/G       3226 non-null object
36 SFTSH/G      3226 non-null object
37 HITS         3226 non-null float64
38 BLS          3226 non-null float64
39 FOA          3226 non-null float64
40 TkA          3226 non-null float64
41 W            491 non-null float64
42 L            491 non-null float64
43 SOP          491 non-null float64
44 GA           491 non-null float64
45 Sv           491 non-null float64
46 %Sv          491 non-null object
47 GAA          491 non-null object
48 SO           491 non-null float64
dtypes: float64(22), int64(7), object(20)
memory usage: 1.4+ MB

```

Есть несоответствие типов содержимому в столбцах. Изменим при необходимости в ходе анализа.

```

len(stat2_new['player'].unique()) #найдем кооличество игроков в датасете

3709

```

```

#выделим id игрока из ссылки на профиль
stat2_new['link'] = stat2_new['player_link'].str.extract(r'/(\\d+)/')
stat2_new.head(1)

```

	player_link	player	position	born	age	country	hight	weight	shoot	GP	...	TkA	W	L	SOP	GA
0	https://en.khl.ru/players/16785/?idplayer=1678...	Juhamatti Aaltonen	forward	4-июн.-85	38	Finland	184	89	right	245	...	0.0	NaN	NaN	NaN	NaN

1 rows × 50 columns

```

len(stat2_new['link'].unique())#найдем кооличество игроков в датасете по id

3717

```

```

#найдем трех игроков, по которым нет мнформации о переходах
uniq = stat2_new['link'].unique()#список игроков в stat2_new uniq
players2.query('link not in @uniq')# срез по списку uniq

```

	player_link	player	team	start_date	end_date	link	
4591	https://en.khl.ru/players/16439/?idplayer=1643...	Adam Munro	Sibir	2010-01-16	2010-03-07	16439	
4848	https://en.khl.ru/players/15624/?idplayer=1562...	Alexander Novikov	Vityaz	2009-01-27	2009-02-26	15624	
6415	https://en.khl.ru/players/15246/?idplayer=1524...	Igor Shvedov	Dinamo Mn	2009-01-27	2009-02-26	15246	

Для 3 игроков данных в players2 (о переходах) нет

▼ Проверка данных на наличие дубликатов

```

players2.duplicated().sum() #найдем явные дубликаты

0

```

Явных дубликатов нет.

```

stat2_new.duplicated().sum() #найдем явные дубликаты

0

```

Явных дубликатов нет.

▼ Анализ данных

▼ Построим график переходов игроков в команды

Sankey Chart: визуализируем переходы между командами Чтобы строить sankey, у вас должна быть готова таблица с информацией по трансферам игроков - даты здесь не так важны, как сами переходы из команды в команду - т.е. эту визуализацию можем строить практически сразу.

Для построения Sankey чатра воспользуемся тьюториалом отсюда: <https://habr.com/ru/articles/566568/> и документацией plotly по Sankey: <https://plotly.com/python/sankey-diagram/>

```
players2.head()# выведем первые 5 строк
```

	player_link	player	team	start_date	end_date	link	
0	https://en.khl.ru/players/16785/?idplayer=1678...	Juhamatti Aaltonen	Jokerit	2014-09-04	2016-03-02	16785	
1	https://en.khl.ru/players/16785/?idplayer=1678...	Juhamatti Aaltonen	Metallurg Mg	2010-09-09	2012-03-22	16785	
2	https://en.khl.ru/players/17585/?idplayer=1758...	Miro Aaltonen	Vityaz	2021-09-02	2022-01-11	17585	
3	https://en.khl.ru/players/17585/?idplayer=1758...	Miro Aaltonen	SKA	2019-12-17	2021-02-27	17585	
4	https://en.khl.ru/players/17585/?idplayer=1758...	Miro Aaltonen	Vityaz	2016-08-27	2019-12-09	17585	

Для построения графика перехода выделим 10 команд, с наибольшим количеством игроков

```
top_teams = players2.groupby('team').agg(
    num_players=('link', 'count')).sort_values(by='num_players', ascending=False)[:10].index # группируем по командам, сортируем по сумме
```

```
top_teams
```

```
Index(['Spartak', 'Neftekhimik', 'Amur', 'Torpedo', 'Vityaz', 'Severstal',
      'Avtomobilist', 'Dinamo Mn', 'CSKA', 'Avangard'],
      dtype='object', name='team')
```

```
players2_top10_team = players2.query('team in @top_teams').copy() #выделим датасет с командами в top_teams
#players2_top10_team
```

Добавим необходимые столбцы в датасете:

```
def add_features(df):
```

```
    """Функция генерации новых столбцов для исходной таблицы
```

```
    Args:
```

```
        df (pd.DataFrame): исходная таблица.
```

```
    Returns:
```

```
        pd.DataFrame: таблица с новыми признаками.
```

```
    """
```

```
    # сортируем по id и времени
```

```
    sorted_df = players2_top10_team.sort_values(by=['link', 'start_date']).copy()
```

```
    # добавляем шаги событий
```

```
    sorted_df['step'] = sorted_df.groupby('link').cumcount() + 1
```

```
    # добавляем узлы-источники и целевые узлы
```

```
    # узлы-источники - это сами события
```

```
    sorted_df['source'] = sorted_df['team']
```

```
    # добавляем целевые узлы
```

```
    sorted_df['target'] = sorted_df.groupby('link')['source'].shift(-1)
```

```
    # возврат таблицы без имени событий
```

```
    return sorted_df.drop(['team'], axis=1)
```

```
# преобразуем таблицу
```

```
players_step = add_features(players2_top10_team)
```

```
players_step.head()
```



```
df_comp = players_step[players_step['step'] <= 7].copy().reset_index(drop=True)# оставим в датасете информацию о первых 7ми шагах
3885 https://en.khl.ru/players/1/?idplayer=1&PAGE_N_1=1 Evgeny Lobanov 2011-09-14 2014-09-30 1 2 Avtomobilist NaN

def get_source_index(df):

    """Функция генерации индексов source

    Args:
        df (pd.DataFrame): исходная таблица с признаками step, source, target.
    Returns:
        dict: словарь с индексами, именами и соответствиями индексов именам source.
    """

    res_dict = {}

    count = 0
    # получаем индексы источников
    for no, step in enumerate(df['step'].unique().tolist()):
        # получаем уникальные наименования для шага
        res_dict[no+1] = {}
        res_dict[no+1]['sources'] = df[df['step'] == step]['source'].unique().tolist()
        res_dict[no+1]['players'] = df[df['step'] == step]['player'].unique().tolist()
        res_dict[no+1]['sources_index'] = []
        for i in range(len(res_dict[no+1]['sources'])):
            res_dict[no+1]['sources_index'].append(count)
            count += 1

    # соединим списки
    for key in res_dict:
        res_dict[key]['sources_dict'] = {}
        for name, no in zip(res_dict[key]['sources'], res_dict[key]['sources_index']):
            res_dict[key]['sources_dict'][name] = no
    return res_dict
source_indexes = get_source_index(df_comp)
#source_indexes

def generate_random_color():

    """Случайная генерация цветов rgba

    Args:

    Returns:
        str: Строка со сгенерированными параметрами цвета
    """

    # сгенерим значение для каждого канала
    r, g, b = np.random.randint(255, size=3)
    return f'rgba({r}, {g}, {b}, 1)'

def colors_for_sources(mode):

    """Генерация цветов rgba

    Args:
        mode (str): сгенерировать случайные цвета, если 'random', а если 'custom' -
            использовать заранее подготовленные
    Returns:
        dict: словарь с цветами, соответствующими каждому индексу
    """
    # словарь, в который сложим цвета в соответствии с индексом
    colors_dict = {}

    if mode == 'random':
        # генерим случайные цвета
        for label in df_comp['source'].unique():
            r, g, b = np.random.randint(255, size=3)
            colors_dict[label] = f'rgba({r}, {g}, {b}, 1)'

    elif mode == 'custom':
        # присваиваем ранее подготовленные цвета
        colors = requests.get('https://raw.githubusercontent.com/rusantsovsv/senkey_tutorial/main/json/colors_senkey.json').json()
        for no, label in enumerate(df_comp['source'].unique()):
            colors_dict[label] = colors['custom_colors'][no]

    return colors_dict
```

```

# генерию цвета из своего списка
colors_dict = colors_for_sources(mode='custom')

def percent_users(sources, targets, values):

    """
    Расчет уникальных id в процентах (для вывода в hover text каждого узла)

    Args:
        sources (list): список с индексами source.
        targets (list): список с индексами target.
        values (list): список с "объемами" потоков.

    Returns:
        list: список с "объемами" потоков в процентах
    """

    # объединим источники и метки и найдем пары
    zip_lists = list(zip(sources, targets, values))

    new_list = []

    # подготовим список словарь с общим объемом трафика в узлах
    unique_dict = {}

    # проходим по каждому узлу
    for source, target, value in zip_lists:
        if source not in unique_dict:
            # находим все источники и считаем общий трафик
            unique_dict[source] = 0
            for sr, tg, vl in zip_lists:
                if sr == source:
                    unique_dict[source] += vl

    # считаем проценты
    for source, target, value in zip_lists:
        new_list.append(round(100 * value / unique_dict[source], 1))

    return new_list

def lists_for_plot(source_indexes=source_indexes, colors=colors_dict, frac=0):

    """
    Создаем необходимые для отрисовки диаграммы переменные списков и возвращаем
    их в виде словаря

    Args:
        source_indexes (dict): словарь с именами и индексами source.
        colors (dict): словарь с цветами source.
        frac (int): ограничение на минимальный "объем" между узлами.

    Returns:
        dict: словарь со списками, необходимыми для диаграммы.
    """

    sources = []
    targets = []
    values = []
    labels = []
    link_color = []
    link_text = []

    # проходим по каждому шагу
    for step in tqdm(sorted(df_comp['step'].unique()), desc='Шаг'):
        if step + 1 not in source_indexes:
            continue

        # получаем индекс источника
        temp_dict_source = source_indexes[step]['sources_dict']

        # получаем индексы цели
        temp_dict_target = source_indexes[step+1]['sources_dict']

        # проходим по каждой возможной паре, считаем количество таких пар
        for source, index_source in tqdm(temp_dict_source.items()):
            for target, index_target in temp_dict_target.items():
                # делаем срез данных и считаем количество id
                temp_df = df_comp[(df_comp['step'] == step)&(df_comp['source'] == source)&(df_comp['target'] == target)]
                value = len(temp_df)
                # проверяем минимальный объем потока и добавляем нужные данные

```

```

        if value > frac:
            sources.append(index_source)
            targets.append(index_target)
            values.append(value)
            # делаем поток прозрачным для лучшего отображения
            link_color.append(colors[source].replace(',', '1'), ', ', 0.2)))

labels = []
colors_labels = []
for key in source_indexes:
    for name in source_indexes[key]['sources']:
        labels.append(name)
        colors_labels.append(colors[name])

# посчитаем проценты всех потоков
perc_values = percent_users(sources, targets, values)

# добавим значения процентов для howertext
link_text = []
for perc in perc_values:
    link_text.append(f"{perc}%")

# возвратим словарь с вложенными списками
return {'sources': sources,
        'targets': targets,
        'values': values,
        'labels': labels,
        'colors_labels': colors_labels,
        'link_color': link_color,
        'link_text': link_text}

# создаем словарь
data_for_plot = lists_for_plot()

War: 0%|          | 0/7 [00:00<?, ?it/s]
0%|          | 0/10 [00:00<?, ?it/s]
40%|██████    | 4/10 [00:00<00:00, 35.95it/s]
100%|██████████| 10/10 [00:00<00:00, 44.22it/s]
War: 14%|███    | 1/7 [00:00<00:01, 4.11it/s]
0%|          | 0/10 [00:00<?, ?it/s]
100%|██████████| 10/10 [00:00<00:00, 54.33it/s]
War: 29%|█████  | 2/7 [00:00<00:01, 4.67it/s]
0%|          | 0/10 [00:00<?, ?it/s]
100%|██████████| 10/10 [00:00<00:00, 70.39it/s]
War: 43%|██████  | 3/7 [00:00<00:00, 5.35it/s]
0%|          | 0/10 [00:00<?, ?it/s]
100%|██████████| 10/10 [00:00<00:00, 58.50it/s]
War: 57%|███████ | 4/7 [00:00<00:00, 5.42it/s]
0%|          | 0/10 [00:00<?, ?it/s]
100%|██████████| 10/10 [00:00<00:00, 76.51it/s]
War: 71%|███████ | 5/7 [00:00<00:00, 5.93it/s]
100%|██████████| 8/8 [00:00<00:00, 101.72it/s]
War: 100%|██████████| 7/7 [00:01<00:00, 6.93it/s]

def plot_sankey_diagram(title, data_dict=data_for_plot, width=1300, height=700):

    """
    Функция для генерации объекта диаграммы Сенкей

    Args:
        data_dict (dict): словарь со списками данных для построения.

    Returns:
        plotly.graph_objs._figure.Figure: объект изображения.
    """

    fig = go.Figure(data=[go.Sankey(
        domain = dict(
            x = [0,1],
            y = [0,1]
        ),
        orientation = "h",
        valueformat = ".0f",
        node = dict(
            pad=50,
            thickness=15,
            line=dict(color = "black", width = 0.1),
            label=data_dict['labels'],
            color=data_dict['colors_labels'],
            #customdata=data_dict['players'],
            hovertemplate='Всего переходов у команды %{label}: %{value}<extra></extra>'
        ),
        link = dict(
            arrowlen=15,

```

```

source=data_dict['sources'],
target=data_dict['targets'],
value=data_dict['values'],
label=data_dict['link_text'],
color=data_dict['link_color'],
customdata=data_dict['link_text'],
hovertemplate='Из команды %{source.label}<br />'+
'в команду %{target.label}<br /> перешло игроков: %{value}'+
', что составляет %{customdata}<br />',
)))
fig.update_layout(title_text=title, font_size=10, width=width, height=height)

```

```

# возвращаем объект диаграммы
return fig

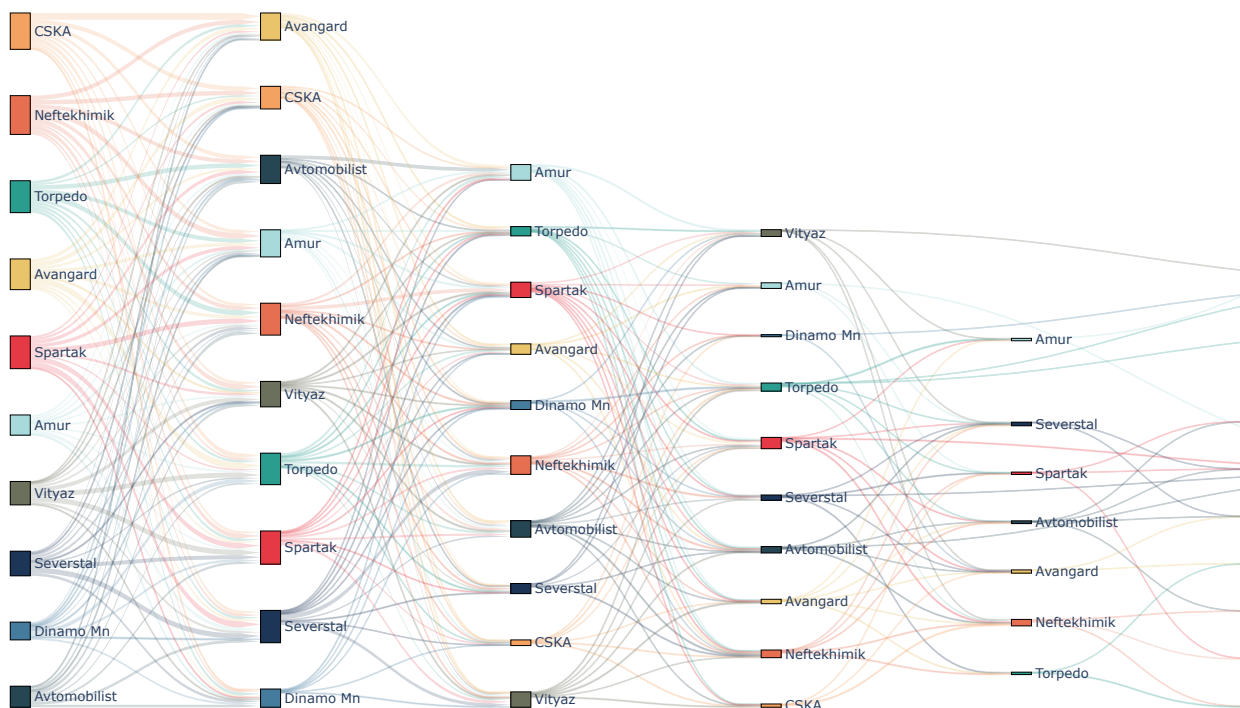
```

```

# строим диаграмму
senkey_diagram = plot_senkey_diagram('Переходы топ-10 команд игроков между командами КХЛ')
senkey_diagram.show()

```

Переходы топ-10 команд игроков между командами КХЛ



Присоединим датасет players2 сам к себе для нахождения одноклубников

```

merged_p12= players2.merge(players2,left_on='team', right_on='team', how='left' )# соединяем 2 датасета
merged_p12.head(10)

```

	player_link_x	player_x	team	start_date_x	end_date_x	link_x	player_link_y	player_y	start_date_y
0	https://en.khl.ru/players/16785/?idplayer=16785	Juhamatti Aaltonen	Jokerit	2014-09-04	2016-03-02	16785	https://en.khl.ru/players/16785/?idplayer=16785	Juhamatti Aaltonen	2014-09-04
1	https://en.khl.ru/players/16785/?idplayer=16785	Juhamatti Aaltonen	Jokerit	2014-09-04	2016-03-02	16785	https://en.khl.ru/players/19127/?idplayer=19127	Niclas Andersen	2017-11-17
2	https://en.khl.ru/players/16785/?idplayer=16785	Juhamatti Aaltonen	Jokerit	2014-09-04	2016-03-02	16785	https://en.khl.ru/players/20970/?idplayer=20970	Marko Anttila	2016-08-25
3	https://en.khl.ru/players/16785/?idplayer=16785	Juhamatti Aaltonen	Jokerit	2014-09-04	2016-03-02	16785	https://en.khl.ru/players/22344/?idplayer=22344	Semir Ben-Amor	2014-09-04

удалим строки, пересечения игроков с самим собой:

```
merged_p12 = merged_p12.loc[(merged_p12['link_x']!= merged_p12['link_y'])].copy()
```

Найдем одноклубников, совместим игроков по датам

```
# установим флаг для игроков, игравших в командах в одно время
merged_p12['flag'] = np.where((merged_p12['start_date_x']<=merged_p12['end_date_y'])&(merged_p12['end_date_x']>=merged_p12['start_date_y']
0      1
1      0
2      0
3      0
merged_p12.head(1)
```

	player_link_x	player_x	team	start_date_x	end_date_x	link_x	player_link_y	player_y	start_date_y
1	https://en.khl.ru/players/16785/?idplayer=16785	Juhamatti Aaltonen	Jokerit	2014-09-04	2016-03-02	16785	https://en.khl.ru/players/19127/?idplayer=19127	Niclas Andersen	2017-11-17

```
merged_p12['flag'].value_counts()#выведем частоту значений 1 и 0
```

```
0    1758350
1     417922
Name: flag, dtype: int64
```

Получили: 417922 одноклубников играли вместе.

Посмотрим, у каких игроков больше одноклубников

```
# группируем по имени и id, находим количество уникальных одноклубников
t = merged_p12.query('flag ==1').groupby(['player_x', 'link_x'])['link_y'].nunique().sort_values(ascending=False)[:15]
t
```

```
player_x    link_x
Evgeny Lapenkov    4351    475
Gennady Stolyarov    548    467
Denis Kazionov    14299    464
Vladimir Galuzin    14815    454
Denis Parshin     494    446
Alexander Lazushin  14674    445
Yakov Rylov       10546    443
Mikhail Grigoryev   14867    437
Ilya Proskuryakov   13871    436
Nikita Tochitsky    15846    430
Alexei Kruchinin    16355    423
Mikhail Zhukov      13679    418
Ilya Krikunov       125    416
Zakhar Arzamastsev  16220    413
Stanislav Galimov   14426    409
Name: link_y, dtype: int64
```

Больше всего одноклубников у : Евгения Лапенкова - 475, Геннадия Столярова - 467 и Дениса Казионова - 464.

▼ Число Лемтюгова

Для каждого хоккеиста необходимо рассчитать число Лемтюгова.

Число Лемтюгова можно определить как минимальное количество команд, через которые данный хоккеист связан с Николаем Лемтюговым. Одноклубниками считаются хоккеисты, которые были участниками одной команды в одно время (с учетом переходов игроков). Учитываются одноклубники по лиге КХЛ.

Словарь link : players

```
# создадим словарь, link (id) будет соответствовать имя игрока
id_dic = players2[['link', 'player']].to_dict('records')
players_dict = {}
for i in id_dic:
    for k, v in i.items():
        players_dict.update({i['link']: i['player']})

search_name = 'Nikolai Lemtyugov'# найдем id Лемтюгова
[link for link, name in players_dict.items() if name == search_name]

['13705']
```

```
players2.loc[(players2['player']== 'Nikolai Lemtyugov')].head(1)
```

	player_link	player	team	start_date	end_date	link	
3746	https://en.khl.ru/players/13705/?idplayer=1370...	Nikolai Lemtyugov	Avangard	2015-08-25	2018-03-10	13705	

Выделим датасет с игроками и их одноклубниками, которые пересекались по времени:

```
play_together = merged_p12[merged_p12['flag']==1].copy()
play_together = play_together[['link_x', 'link_y']]# выведем 2 столбца датасета
```

```
play_together.head()# выведем первые строки
```

	link_x	link_y	
3	16785	22344	
11	16785	22345	
12	16785	22372	
14	16785	19382	
15	16785	22353	

Создадим словарь, в котором ключом будет игрок, а значениями - его одноклубники.

```
dicto = play_together.groupby('link_x').agg({'link_y': 'unique'}).to_dict()
```

```
#dicto
```

```
#Убираем индекс (link_y)- перекладываем значения в новый словарь
newd = {}
for k, v in dicto.items():
    newd.update(v)
```

```
#newd
```

```
#Конвертируем значения из массива в список
for k, v in newd.items():
    newd[k] = v.tolist()
```

```
#newd
```

Функция обхода графа:

Напишем функцию для обхода графа и поиска кратчайшего пути между его вершинами.

Будем использовать такой тип данных, как очередь, и сохранять результаты обхода в словарь.

прикладываю ссылку на источник мудрости:

<https://www.pythonforbeginners.com/data-structures/shortest-path-length-from-a-vertex-to-other-vertices-in-a-graph>

И еще один источник мудрости: https://pimiento.github.io/python_graphs.html - если заходим выводить не просто число, а путь до игрока, можем использовать мануалы с этой страницы.

```
def calculate_distance(input_graph, source):
    Q = Queue()
    distance_dict = {k: 999999999 for k in input_graph.keys()}
    visited_vertices = list()
    Q.put(source)
    visited_vertices.append(source)
    while not Q.empty():
        vertex = Q.get()
        if vertex == source:
            distance_dict[vertex] = 0
        for u in input_graph[vertex]:
            if u not in visited_vertices:
                # update the distance
                if distance_dict[u] > distance_dict[vertex] + 1:
                    distance_dict[u] = distance_dict[vertex] + 1
                Q.put(u)
            visited_vertices.append(u)
    return distance_dict

search_name = 'Nikolai Lemtyugov'
[link for link, name in players_dict.items() if name == search_name]

['13705']
```

#Рассчитываем число Лемтюгова - чем число больше, тем дальше игроки друг от друга, если число равно 999999999, то игроки не связаны

```
distances = calculate_distance(newd,'13705')
```

```
# Преобразуем словарь в датафрейм
lemtyugov_number = pd.DataFrame.from_dict(distances, orient='index').reset_index().rename(columns={'index': 'link', 0: 'number'})

# Добавляем имя игрока из словаря с id игроков
lemtyugov_number['player'] = lemtyugov_number['link'].map(players_dict)

#Меняем порядок столбцов и сортируем по числу в порядке убывания
lemtyugov_number = lemtyugov_number[['link', 'player', 'number']]
lemtyugov_number.sort_values(by='number', ascending=False)
```

	link	player	number	
2706	28636	Ty Schultz	3	
2878	32570	Rasmus Lahnaviik	3	
2873	32393	Yegor V. Guskov	3	
3312	39895	Alex Lintuniemi	3	
3026	3496	Marat Davydov	3	
...	
2149	22606	Alexei Zubov	1	
1476	18818	Vadim Shchegolkov	1	
2155	22669	Derek Roy	1	
1472	18793	David Nosek	1	
116	13705	Nikolai Lemtyugov	0	

3720 rows × 3 columns

```
lemtyugov_number['number'].agg(['mean', 'median'])# найдем среднее и медианное значение числа Лемтюгова
```

```
mean      1.923656
median    2.000000
Name: number, dtype: float64
```

Среднее и медианное значения числа Лемтюгова почти равны.

```
players2.loc[(players2['link']== '13705')].sort_values(by= 'start_date')
```

	player_link	player	team	start_date	end_date	link	
3756	https://en.khl.ru/players/13705/?idplayer=1370...	Nikolai Lemtyugov	Severstal	2008-12-24	2010-01-12	13705	
3755	https://en.khl.ru/players/13705/?idplayer=1370...	Nikolai Lemtyugov	Ak Bars	2010-01-14	2011-10-28	13705	
3754	https://en.khl.ru/players/13705/?idplayer=1370...	Nikolai Lemtyugov	Metallurg Mg	2011-11-04	2011-12-27	13705	
3753	https://en.khl.ru/players/13705/?idplayer=1370...	Nikolai Lemtyugov	Neftekhimik	2012-01-05	2012-02-26	13705	
3752	https://en.khl.ru/players/13705/?idplayer=1370...	Nikolai Lemtyugov	Atlant	2012-09-05	2012-12-23	13705	
3751	https://en.khl.ru/players/13705/?idplayer=1370...	Nikolai Lemtyugov	Traktor	2013-01-04	2013-04-15	13705	
3750	https://en.khl.ru/players/13705/?idplayer=1370...	Nikolai Lemtyugov	Sibir	2013-09-05	2013-10-03	13705	

stat2_new.loc[(stat2_new['link']== '13705')]

	player_link	player	position	born	age	country	height	weight	shoot	GP	...	TkA	W	L	SOP	
1643	https://en.khl.ru/players/13705/?idplayer=1370...	Nikolai Lemtyugov	forward	15-январь-86	37	Russia	184	96	left	416	...	0.0	NaN	NaN	NaN	NaN

1 rows × 50 columns

Н. Лемтюгов 1986 года рождения, играет с конца 2008 года. Найдём для сравнения "число Лемтюгова" для других игроков.

Посчитаем тоже самое для другого игрока, например для Ty Schultz id - 28636

```
#Рассчитываем число Лемтюгова - чем число больше, тем дальше игроки друг от друга, если число равно 999999999, то игроки не связаны
distances_Schultz = calculate_distance(newdw, '28636')

# Преобразуем словарь в датафрейм
Schultz_number = pd.DataFrame.from_dict(distances_Schultz, orient='index').reset_index().rename(columns={'index': 'link', 0: 'number'})

# Добавляем имя игрока из словаря с id игроков
Schultz_number['player'] = lemtyugov_number['link'].map(players_dict)

#Меняем порядок столбцов и сортируем по числу в порядке убывания
Schultz_number = Schultz_number[['link', 'player', 'number']]
Schultz_number.sort_values(by='number', ascending=False)
```

	link	player	number	
1860	20930	Mark Katic	3	
2210	23318	Chad Kolarik	3	
2213	23334	Artyom Korepanov	3	
2214	23335	Mark Owuya	3	
2215	23337	Georgy Kuznetsov	3	
...	
3041	35045	Garet Hunt	1	
3012	34832	Jason Fram	1	
1077	16740	Tomas Jurco	1	
2626	26926	Brandon Yip	1	
2706	28636	Ty Schultz	0	

3720 rows × 3 columns

```
Schultz_number['number'].agg(['mean', 'median'])

mean      2.61586
median    3.00000
Name: number, dtype: float64
```

```
players2.loc[(players2['link']== '28636')]
```

	player_link	player	team	start_date	end_date	link	
6042	https://en.khl.ru/players/28636/?idplayer=2863...	Ty Schultz	Kunlun RS	2021-10-02	2022-12-23	28636	

```
stat2_new.loc[(stat2_new['link']== '28636')]
```


	player_link	player	position	born	age	country	hight	weight	shoot	GP	...	TkA	W	L	SOP	GA
2630	https://en.khl.ru/players/28636/?idplayer=28636	Ty Schultz	defense	5-mar-97	26	China	185	91	right	46	...	2.0	NaN	NaN	NaN	NaN

1 rows × 17 columns

Тy Schultz так же связан со всеми одноклубниками через 3 человека, но медианное и среднее значения "числа Лемтюгова" выше. Это можно объяснить , тем что игрок достаточно молод и пока играл только в 1 команде.


Игрок Evgeny Lapenkov, id - 4351

```
#Рассчитываем число Лемтюгова - чем число больше, тем дальше игроки друг от друга, если число равно 999999999, то игроки не связаны
distances_Lapenkov = calculate_distance(newd, '4351')

# Преобразуем словарь в датафрейм
Lapenkov_number = pd.DataFrame.from_dict(distances_Lapenkov, orient='index').reset_index().rename(columns={'index': 'link', 0: 'number'})

# Добавляем имя игрока из словаря с id игроков
Lapenkov_number['player'] = Lapenkov_number['link'].map(players_dict)

#Меняем порядок столбцов и сортируем по числу в порядке убывания
Lapenkov_number = Lapenkov_number[['link', 'player', 'number']]
Lapenkov_number.sort_values(by='number', ascending=False)
```

	link	player	number	
2263	23560	Alexander Melikhov	3	
3465	42245	Michael Chaput	3	
1299	17663	Roman Rachinsky	3	
1588	19098	David Printz	3	
1293	17651	Kamil Kreps	3	
...	
1560	19036	Vladislav Voropayev	1	
1566	19050	Andrei Yerofeyev	1	
1569	19058	Anatoly Golyshev	1	
3719	99	Evgeny Skachkov	1	
3482	4351	Evgeny Lapenkov	0	

3720 rows × 5 columns

```
Lapenkov_number['number'].agg(['mean', 'median'])

mean      1.888978
median    2.000000
Name: number, dtype: float64

players2.loc[(players2['link']== '4351')].sort_values(by= 'start_date')
```

player linkplayerteam start dateend date link

stat2_new.loc[(stat2_new['link']== '4351')]

player_linkplayerpositionbornagecountryheightweightshootGP...TkAWLSONPGA

3622

<https://en.khl.ru/players/4351/?idplayer=4351&...>

Evgeny Lapenkov

forward

1-
abr-84

38

Russia

192

100

left

569

...

0.0

NaN

NaN

NaN

NaN

1 rows × 50 columns

<

У Лапенкова среднее "число Лемтюгова" даже ниже, чем у самого Лемтюгова. При этом он тоже начал играть с конца 2008 года.

Визуализация графа

```
3651 https://en.khl.ru/players/4351/?idplayer=4351& Evgeny Lapenkov Neftekhimik 2015-08-26 2015-10-20 4351
play_together.groupby('link_x').agg({'link_y': 'unique'})# сгруппируем по игроку, найдем список уникальных одноклубников для каждого
```

link_x	link_y
1	[13873, 13490, 9435, 13923, 15889, 16119, 1929...
10162	[23494, 23973, 24048, 20083, 16074, 28452, 254...
10176	[23249, 39290, 26339, 16672, 17609, 39158, 272...
10427	[14763, 14876, 4512, 13806, 14478, 6303, 13226...
10541	[19232, 14929, 15948, 13923, 22519, 15129, 151...
...	...
9859	[27280, 25049, 13806, 19530, 14822, 23514, 211...
9860	[13873, 13490, 9435, 3739, 16119, 13061, 3607,...
9862	[25268, 27236, 22027, 25206, 19624, 27228, 189...
9863	[18961, 19232, 14929, 3737, 16165, 14551, 1594...
99	[19232, 15948, 22519, 9863, 10541, 19201, 1770...
3720 rows × 1 columns	

```
# добавим имена игроков
play_together_names = pd.DataFrame()
play_together_names['link_x'] = play_together['link_x']
play_together_names['player_x'] = play_together['link_x'].map(players_dict)
play_together_names['link_y'] = play_together['link_y']
play_together_names['player_y'] = play_together['link_y'].map(players_dict)
play_together_names
```

	link_x	player_x	link_y	player_y
3	16785	Juhamatti Aaltonen	22344	Semir Ben-Amor
11	16785	Juhamatti Aaltonen	22345	Ryan Gunderson
12	16785	Juhamatti Aaltonen	22372	Frank Gymer
14	16785	Juhamatti Aaltonen	19382	Niklas Hagman
15	16785	Juhamatti Aaltonen	22353	Riku Hahl
...
2185574	15734	Maxim Zyuzynkin	13136	Vitaly Vishnevsky
2185576	15734	Maxim Zyuzynkin	4089	Alexander Vyukhin
2185578	15734	Maxim Zyuzynkin	15604	Artyom Yarchuk
2185584	15734	Maxim Zyuzynkin	15972	Richard Zednik
2185585	15734	Maxim Zyuzynkin	4808	Sergei P. Zhukov
417922 rows × 4 columns				

```
#выделим топ-30 игроков
top_id = merged_p12[merged_p12['flag']==1].groupby(['link_x'])['link_y'].nunique().sort_values(ascending=False)[:30].index.to_list()
top_ids = play_together_names.query('link_x in @top_id')
```

#Создаем лист для изменения расчета расстояний между элементами - он нам понадобится, если захотим применить метод расчета расстояний shap

```
rp = top_ids.groupby('player x').agg({'player y': 'unique'})['player y'].to dict()
```

```

for k, v in rp.items():
    rp[k] = v.tolist()

lst = []
for i, v in zip(top_id, rp.values()):
    lst.append([i])
    lst.append(v)

#Создаем граф из таблицы pandas - у библиотеки Networks два метода работы с датафреймами pandas, разберем from_pandas_edgelist
G = nx.from_pandas_edgelist(top_ids, 'player_x', 'player_y')

#G = nx.from_pandas_edgelist(real_peers_names, 'player_x', 'player_y', create_using=nx.MultiGraph())

#Можем изменить внешний вид визуализации, если поменяем способ расчета расстояний
pos = nx.spring_layout(G)
#pos = nx.layout.shell_layout(G, lst)

#Добавляем атрибут с информацией по расположению узлов (можно задать через set_attribute)
for node in G.nodes:
    G.nodes[node]['pos'] = list(pos[node])

```

Материалы: возьмем код с Plotly: <https://plotly.com/python/network-graphs/>

```

edge_x = []
edge_y = []
for edge in G.edges():
    x0, y0 = G.nodes[edge[0]]['pos']
    x1, y1 = G.nodes[edge[1]]['pos']
    edge_x.append(x0)
    edge_x.append(x1)
    edge_x.append(None)
    edge_y.append(y0)
    edge_y.append(y1)
    edge_y.append(None)

edge_trace = go.Scatter(
    x=edge_x, y=edge_y,
    line=dict(width=0.5, color='#888'),
    hoverinfo='none',
    mode='lines')

node_x = []
node_y = []
for node in G.nodes():
    x, y = G.nodes[node]['pos']
    node_x.append(x)
    node_y.append(y)

node_trace = go.Scatter(
    x=node_x, y=node_y,
    mode='markers',
    hoverinfo='text',
    marker=dict(
        showscale=True,
        # colorscale options
        #'Greys' | 'YlGnBu' | 'Greens' | 'YlOrRd' | 'Bluered' | 'RdBu' |
        #'Reds' | 'Blues' | 'Picnic' | 'Rainbow' | 'Portland' | 'Jet' |
        #'Hot' | 'Blackbody' | 'Earth' | 'Electric' | 'Viridis' |
        colorscale='Hot',
        reversescale=True,
        color=[],
        size=10,
        colorbar=dict(
            thickness=15,
            title='Node Connections',
            xanchor='left',
            titleside='right'
        ),
    ),
    line_width=2))

node_adjacencies = []
node_text = []
for node, adjacencies in enumerate(G.adjacency()):
    node_adjacencies.append(len(adjacencies[1]))
    node_text.append('player: '+str(adjacencies[0])+ ' ' + '# of connections: '+str(len(adjacencies[1])))

node_trace.marker.color = node_adjacencies
node_trace.text = node_text

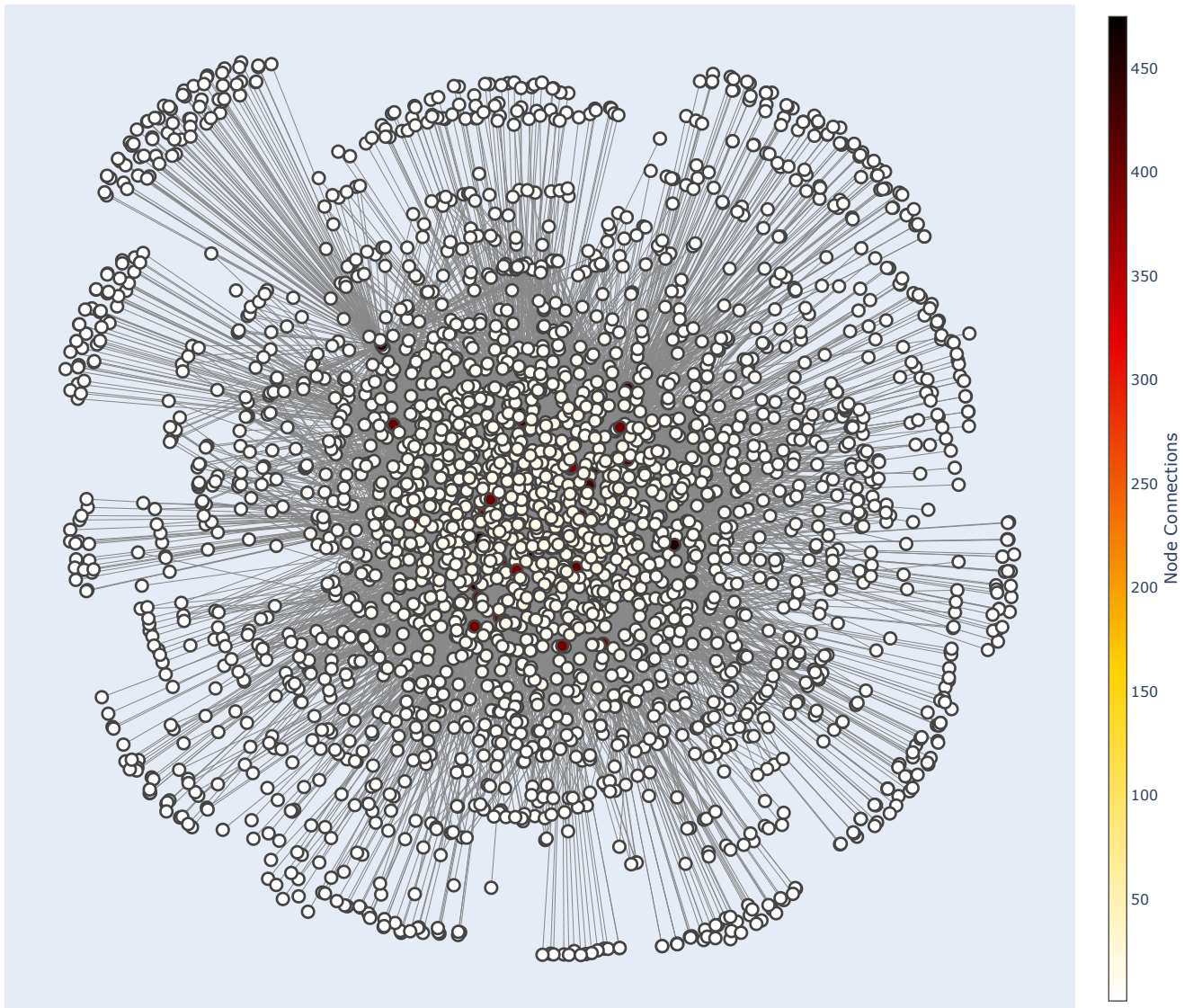
```

```

fig = go.Figure(data=[edge_trace, node_trace],
               layout=go.Layout(width=1000, height=900,
                                title='Граф топ-30 игроков и их связей',
                                titlefont_size=16,
                                showlegend=False,
                                hovermode='closest',
                                margin=dict(b=20,l=5,r=5,t=40),
                                annotations=[ dict(
                                    text="",
                                    showarrow=False,
                                    xref="paper", yref="paper",
                                    x=0.005, y=-0.002 ) ],
                                xaxis=dict(showgrid=False, zeroline=False, showticklabels=False),
                                yaxis=dict(showgrid=False, zeroline=False, showticklabels=False))
fig.show()

```

Граф топ-30 игроков и их связей



Визуализация графов с помощью Holoview:

https://holoviews.org/user_guide/Network_Graphs.html

```

kwargs = dict(width=800, height=800, xaxis=None, yaxis=None)
opts.defaults(opts.Nodes(**kwargs), opts.Graph(**kwargs))

colors = ['#000000']+hv.Cycle('Category20').values

graph = hv.Graph.from_networkx(G, pos)

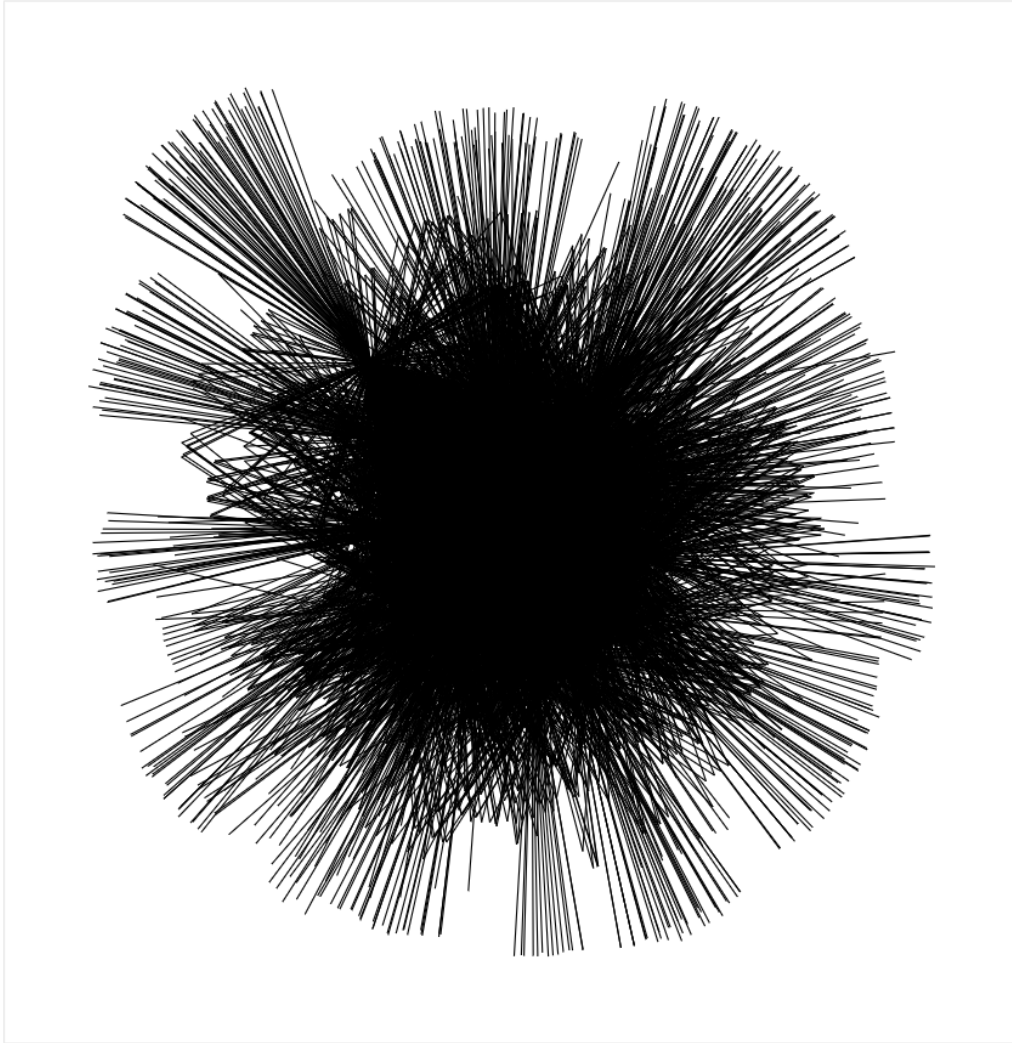
graph.opts(cmap=colors, node_size=5, edge_line_width=1,
           node_line_color='gray')

```

```

hv.extension('bokeh')
hv.output(size=100)
output_file('test.html')
show(hv.render(graph))

```



▼ Проверка гипотез

Проверка гипотез

- проверить гипотезу о том, что Лемтюгов – самый популярный хоккеист в лиге, то есть обладающий самым большим количеством одноклубников. Если это не так, составить список наиболее популярных игроков.
- правда ли, что игроки с низким значением числа Лемтюгова играли в среднем в большем количестве команд, чем игроки, у которых это значение больше?
- есть ли игроки, у которых отсутствует число Лемтюгова (нет связей через других игроков)?
- каково медианное значение числа Лемтюгова?
- проверить, есть ли связь между количеством переходов и ростом/весом игрока (предполагаем, что игроки, у которых рост выше, лучше играют в хоккей и соответственно чаще переходят из клуба в клуб).
- проверить, верна ли гипотеза о том, что более успешные хоккеисты рождаются в январе-марте.

Проверить гипотезу о том, что Лемтюгов – самый популярный хоккеист в лиге, то есть обладающий

- ▼ самым большим количеством одноклубников. Если это не так, составить список наиболее популярных игроков.

```
merged_p12.query('flag ==1').groupby(['player_x', 'link_x'])['link_y'].nunique().sort_values(ascending=False)[:20]# найдем количество од-
```

player_x	link_x	
Evgeny Lapenkov	4351	475
Gennady Stolyarov	548	467

Denis Kazionov	14299	464
Vladimir Galuzin	14815	454
Denis Parshin	494	446
Alexander Lazushin	14674	445
Yakov Rylov	10546	443
Mikhail Grigoryev	14867	437
Ilya Proskuryakov	13871	436
Nikita Tochitsky	15846	430
Alexei Kruchinin	16355	423
Mikhail Zhukov	13679	418
Ilya Krikunov	125	416
Zakhar Arzamastsev	16220	413
Stanislav Galimov	14426	409
Yegor Milovzorov	14257	408
Mikhail Yunkov	595	407
Maxim A. Goncharov	14612	406
Konstantin Glazachev	266	406
Andrei Sergeyev	15416	402

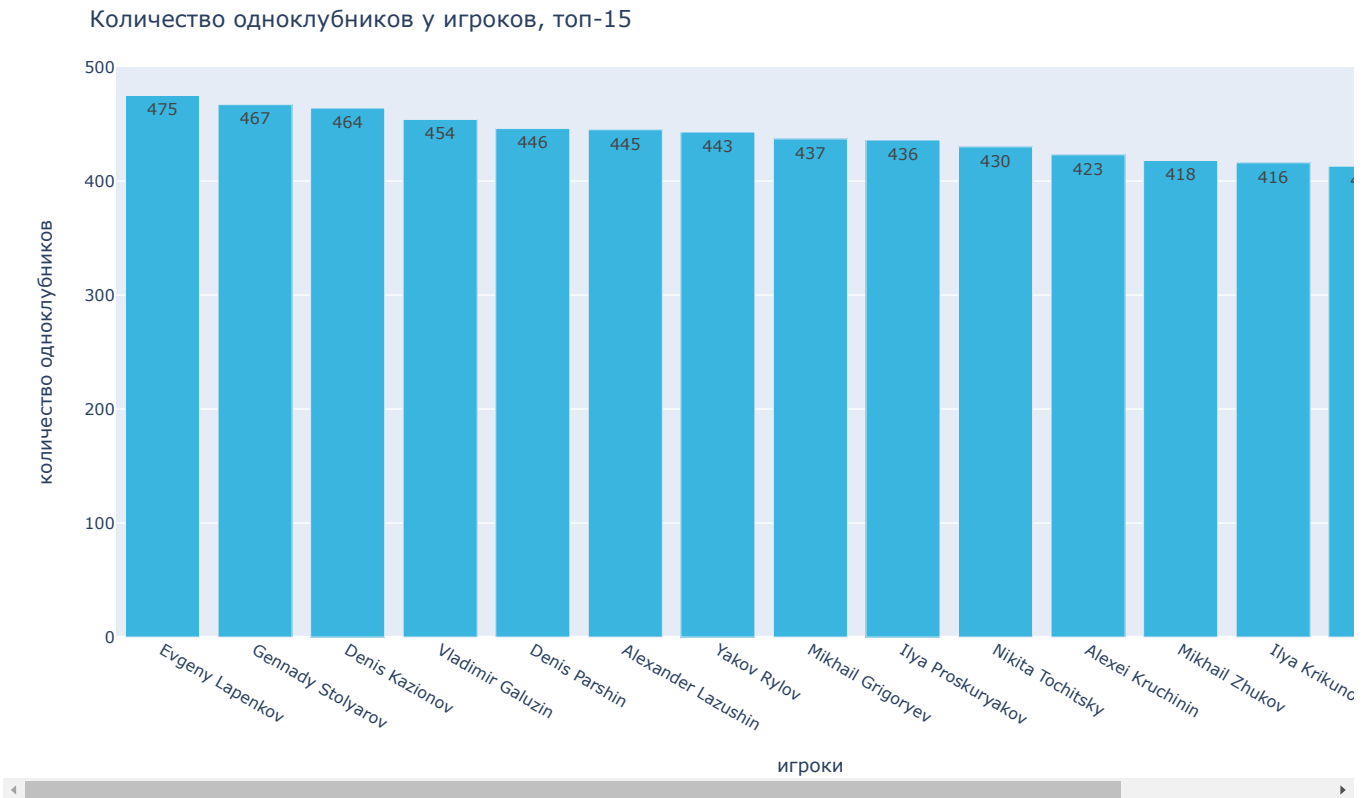
Name: link_y, dtype: int64

Посмотрим на графике самых популярных игроков:

```
data = merged_pl2.query('flag ==1').groupby(['player_x', 'link_x'],as_index = False)['link_y'].nunique().sort_values(by='link_y', ascending=False)

fig = px.bar(data, x = 'player_x', y = 'link_y', \
             color_discrete_sequence=[ '#39B5E0', '#4C6793'],\
             text='link_y',height=600, width=1200
             )
fig.update_layout(title =f'Количество одноклубников у игроков, топ-15', xaxis_title= 'игроки',\
                  yaxis_title= 'количество одноклубников',\
                  xaxis={'categoryorder':'total descending'})

fig.show()
```



```
data_L = merged_pl2.query('flag ==1').\
groupby(['player_x', 'link_x'],as_index = False)['link_y'].nunique().sort_values(by='link_y', ascending=False)
```

Лемтюгов не вошел в топ-15. Найдем количество его одноклубников.

```
data_L.loc[(data_L['player_x']== 'Nikolai Lemtyugov')]\# выведем строку со значением в столбце player_x - Nikolai Lemtyugov
```

	player_x	link_x	link_y	
2588	Nikolai Lemtyugov	13705	357	il

Правда ли, что игроки с низким значением числа Лемтюгова играли в среднем в большем количестве команд, чем игроки, у которых это значение больше?

```
merged_pl2.head(1)
```

	player_link_x	player_x	team	start_date_x	end_date_x	link_x	player_link_y	player_y	start_date_y
1	https://en.khl.ru/players/16785/?idplayer=16785...	Juhamatti Aaltonen	Jokerit	2014-09-04	2016-03-02	16785	https://en.khl.ru/players/19127/?idplayer=19127...	Niclas Andersen	2017-11-17

```
# посчитаем количество команд у игроков
```

```
data_team = merged_pl2.query('flag ==1').groupby(['player_x', 'link_x'], as_index = False)['team'].nunique().sort_values(by='team', ascending=False).head(1)
```

	player_x	link_x	team
942	Denis Kazionov	14299	11

Соединим два датасета с информацией о числе Лемтюгова и количеством команд

```
lemtyugov_number.sort_values(by='number', ascending=False).head(1)
```

	link	player	number
2706	28636	Ty Schultz	3

```
merged_lemtyugov_team = data_team.merge(lemtyugov_number, left_on='link_x', right_on='link', how='left') # соединим датасеты data_team и le
```

```
merged_lemtyugov_team.head()
```

	player_x	link_x	team	link	player	number
0	Denis Kazionov	14299	11	14299	Denis Kazionov	2
1	Alexander Ryazantsev	12990	11	12990	Alexander Ryazantsev	2
2	Gennady Stolyarov	548	11	548	Gennady Stolyarov	2
3	Yakov Rylov	10546	10	10546	Yakov Rylov	2
4	Mikhail Zhukov	13679	10	13679	Mikhail Zhukov	1

H0: Статистически значимой разницы в количестве команд, в которых играли игроки с низким значением числа Лемтюгова и игроки, у которых это значение больше, нет.

H1: Статистически значимая разница в количестве команд, в которых играли игроки с низким значением числа Лемтюгова и игроки, у которых это значение больше, есть.

```
sample_1 = merged_lemtyugov_team.loc[(merged_lemtyugov_team['number'] == 1)]['team'] # набор данных, условие число Лемтюгова = 1
sample_2 = merged_lemtyugov_team.loc[(merged_lemtyugov_team['number'] > 1)]['team'] # набор данных, условие число Лемтюгова > 1
```

```
alpha = 0.05 # уровень статистической значимости
# если p-value окажется меньше него, отвергнем гипотезу
```

```
results = st.ttest_ind(sample_1, sample_2)
```

```
print(results.statistic, 'p-значение:', results.pvalue)
```

```
if results.pvalue < alpha:
    print('Отвергаем нулевую гипотезу')
else:
    print('Не получилось отвергнуть нулевую гипотезу')
```

```
22.800506898172316 p-значение: 8.17254670675204e-108
Отвергаем нулевую гипотезу
```

Количество команд, в которых играли игроки с низким значением числа Лемтюгова, отличается от количества команд, в которых играли игроки с высоким значением числа Лемтюгова. Средние различаются на 23%.

▼ Есть ли игроки, у которых отсутствует число Лемтюгова (нет связей через других игроков)?

```
#Код ревьюера - проверяем игроков без связей с Лемтюговым
lemtyugov_number.loc[(lemtyugov_number['number'] == 999999999)]
```

link	player	number

Таких игроков нет

▼ Каково медианное значение числа Лемтюгова?

```
lemtyugov_number['number'].median()#найдем медиану числа Лемтюгова
2.0
```

Медианное значение числа Лемтюгова равно 2.

▼ Проверить, есть ли связь между количеством переходов и ростом/весом игрока (предполагаем, что игроки, у которых рост выше, лучше играют в хоккей и соответственно чаще переходят из клуба в клуб).

```
players2.head()
```

	player_link	player	team	start_date	end_date	link
0	https://en.khl.ru/players/16785/?idplayer=1678...	Juhamatti Aaltonen	Jokerit	2014-09-04	2016-03-02	16785
1	https://en.khl.ru/players/16785/?idplayer=1678...	Juhamatti Aaltonen	Metallurg Mg	2010-09-09	2012-03-22	16785
2	https://en.khl.ru/players/17585/?idplayer=1758...	Miro Aaltonen	Vityaz	2021-09-02	2022-01-11	17585
3	https://en.khl.ru/players/17585/?idplayer=1758...	Miro Aaltonen	SKA	2019-12-17	2021-02-27	17585
4	https://en.khl.ru/players/17585/?idplayer=1758...	Miro Aaltonen	Vityaz	2016-08-27	2019-12-09	17585

```
# найдем количество переходов у игроков
transition = players2.groupby(['link', 'player'],as_index = False).agg(
    num_teams=('team', 'count')).sort_values(by='num_teams', ascending=False)
transition
```

	link	player	num_teams
3482	4351	Evgeny Lapenkov	14
108	13679	Mikhail Zhukov	14
3601	5433	Gleb Klimenko	13
221	14299	Denis Kazionov	13
3605	548	Gennady Stolyarov	12
...
2219	23356	Jeff Deslauriers	1
872	16125	Alexander Zakirov	1
2217	23351	Calvin Heeter	1
874	16134	Vadim Mitryakov	1
1860	20930	Mark Katic	1

3720 rows × 3 columns

```
merged_pl2['link_x'].nunique()# проверим
3720
```

```
stat_transition = stat2_new[['link','player', 'hight', 'weight']]# выделим некоторые поля stat2_new в отдельный датасет
stat_transition
```


	link	player	hight	weight	
0	16785	Juhamatti Aaltonen	184	89	
1	17585	Miro Aaltonen	177	84	
2	13041	Ruslan Abdrakhmanov	178	77	
3	38736	Jindrich Abdul	185	85	
4	24998	Ilnur Abdulkhakov	187	84	
...	
3712	19061	Viktor Zakharov	194	85	
3713	16028	Ignat Zemchenko	189	97	
3714	15765	Stepan Zhdanov	170	76	
3715	16593	Nikolai Zhilin	187	88	
3716	40564	Nikita A. Zimin	188	81	

3717 rows x 4 columns

```
stat_transition['link'].nunique()
```

3717

#соединим таблицы с количеством переходов и данных о весе/росте игроков

```
merged_stat_transition = stat_transition.merge(transition,left_on='link', right_on='link', how='inner' )
merged_stat_transition#.head(30)
```

	link	player_x	hight	weight	player_y	num_teams
0	16785	Juhamatti Aaltonen	184	89	Juhamatti Aaltonen	2
1	17585	Miro Aaltonen	177	84	Miro Aaltonen	3
2	13041	Ruslan Abdrakhmanov	178	77	Ruslan Abdrakhmanov	1
3	38736	Jindrich Abdul	185	85	Jindrich Abdul	1
4	24998	Ilnur Abdulkhakov	187	84	Ilnur Abdulkhakov	1
...
3712	19061	Viktor Zakharov	194	85	Viktor Zakharov	1
3713	16028	Ignat Zemchenko	189	97	Ignat Zemchenko	5
3714	15765	Stepan Zhdanov	170	76	Stepan Zhdanov	1
3715	16593	Nikolai Zhilin	187	88	Nikolai Zhilin	1
3716	40564	Nikita A. Zimin	188	81	Nikita A. Zimin	1

3717 rows x 6 columns

```
mergedd = stat2_new.merge(transition,left_on='link', right_on='link', how='inner' )
```

```
mergedd['link'].count()
```

3717

Найдем медианные значения роста и вес игроков

```
weight_median = merged_stat_transition['weight'].median()
weight_median
```

87.0

H0: нет статистически значимой разницы в количестве переходов у игроков с разным ростом.

H1: есть статистически значимая разница в количестве переходов у игроков с разным ростом.

```
hight_median = merged_stat_transition['hight'].median()# найдем медиану роста игроков
# выделим два набора данных, больше и меньше медианного роста
sample_3 = merged_stat_transition.loc[(merged_stat_transition['hight']<= hight_median)][ 'num_teams' ]
sample_4 = merged_stat_transition.loc[(merged_stat_transition['hight']> hight_median)][ 'num_teams' ]
```

```
alpha = 0.05 # уровень статистической значимости
# если p-value окажется меньше него, отвергнем гипотезу
```

```
results = st.ttest_ind(sample_3, sample_4, equal_var=False)
```

```
print(results.statistic, 'p-значение:', results.pvalue)

if results.pvalue < alpha:
    print('Отвергаем нулевую гипотезу')
else:
    print('Не получилось отвергнуть нулевую гипотезу')

-1.6938623805254616 p-значение: 0.09038008649736448
Не получилось отвергнуть нулевую гипотезу
```

Гипотезу о связи между количеством переходов и ростом игрока отвергаем.

H0: нет статистически значимой разницы в количестве переходов у игроков с разным весом.

H1: есть статистически значимая разница в количестве переходов у игроков с разным весом.

```
weight_median = merged_stat_transition['weight'].median()# найдем медиану веса игроков
# выделим два набора данных, больше и меньше медианного веса

sample_5 = merged_stat_transition.loc[(merged_stat_transition['weight']<= weight_median)][ 'num_teams']
sample_6 = merged_stat_transition.loc[(merged_stat_transition['weight']> weight_median)][ 'num_teams']

alpha = 0.05 # уровень статистической значимости
# если p-value окажется меньше него, отвергнем гипотезу

results = st.ttest_ind(sample_5, sample_6, equal_var=False)

print(results.statistic, 'p-значение:', results.pvalue)

if results.pvalue < alpha:
    print('Отвергаем нулевую гипотезу')
else:
    print('Не получилось отвергнуть нулевую гипотезу')

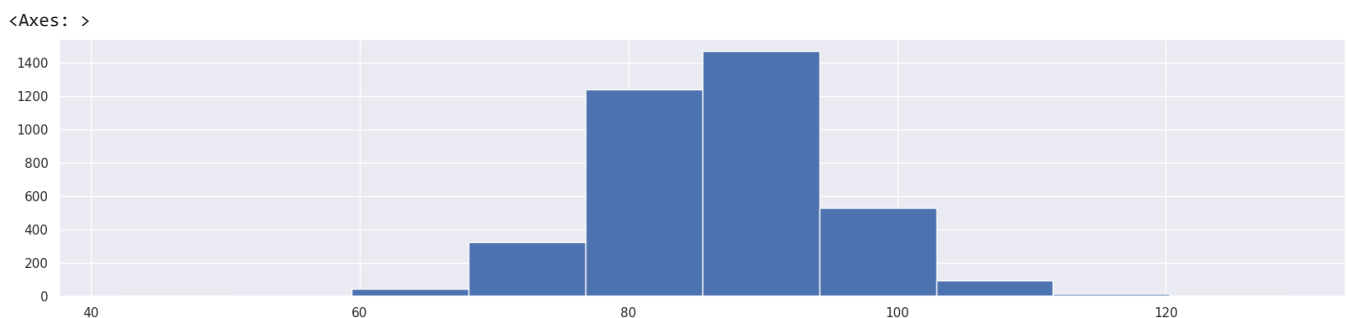
-11.156710563343138 p-значение: 2.2064160613571706e-28
Отвергаем нулевую гипотезу
```

Гипотезу о связи количества переходов и весом игрока принимаем.

Посмотрим на графиках:

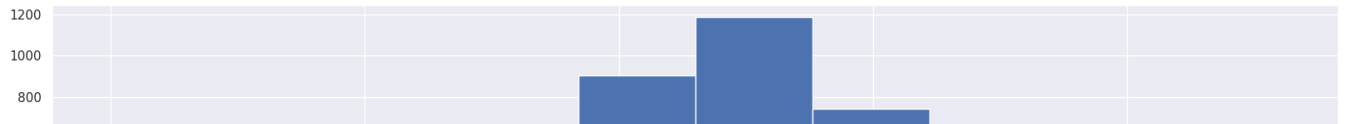
Связь количества переходов и веса игрока

```
merged_stat_transition['weight'].hist()#value_counts()
```



```
merged_stat_transition['hight'].hist()
```

<Axes: >



Plotting the KDE Plot

```
sns.kdeplot(merged_stat_transition.loc[(merged_stat_transition['weight']<= weight_median),'num_teams'], color='r', fill=True)#, Label=87)
```

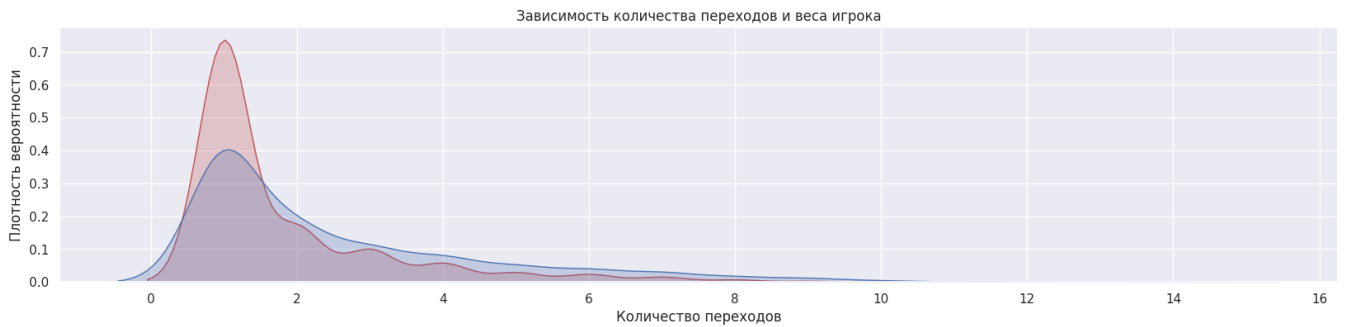
```
sns.kdeplot(merged_stat_transition.loc[(merged_stat_transition['weight']> weight_median),'num_teams'], color='b', fill=True)#, Label=87)
```

```
plt.title('Зависимость количества переходов и веса игрока')
```

```
plt.xlabel('Количество переходов')
```

```
plt.ylabel('Плотность вероятности')
```

```
plt.show()
```



Plotting the KDE Plot

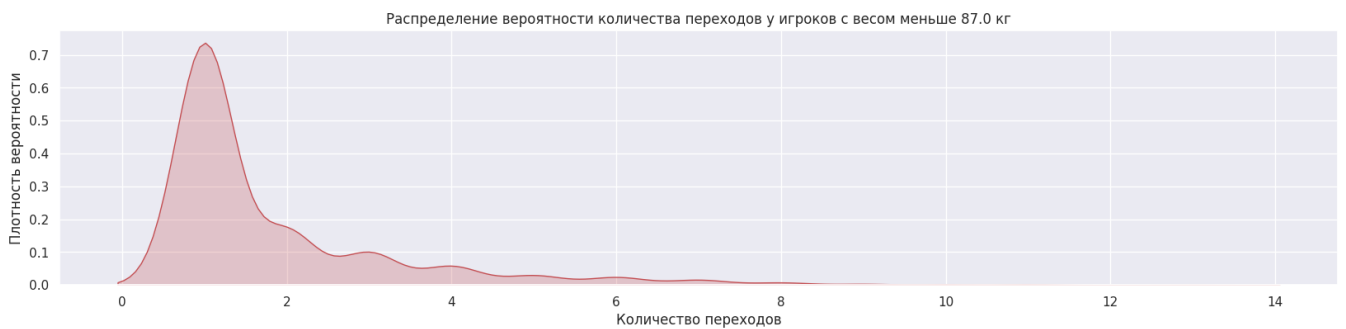
```
sns.kdeplot(merged_stat_transition.loc[(merged_stat_transition['weight']<= weight_median),'num_teams'], color='r', fill=True)#, Label=87)
```

```
plt.title(f'Распределение вероятности количества переходов у игроков с весом меньше {weight_median} кг')
```

```
plt.xlabel('Количество переходов')
```

```
plt.ylabel('Плотность вероятности')
```

```
plt.show()
```



Plotting the KDE Plot

```
sns.kdeplot(merged_stat_transition.loc[(merged_stat_transition['weight']> weight_median),'num_teams'], color='b', fill=True)#, Label=87)
```

```
plt.title(f'Распределение вероятности количества переходов у игроков с весом больше {weight_median} кг')
```

```
plt.xlabel('Количество переходов')
```

```
plt.ylabel('Плотность вероятности');
```



```
# Plotting the KDE Plot
sns.kdeplot(merged_stat_transition.loc[(merged_stat_transition['hight']<= hight_median),'num_teams'], color='r', fill=True)#, Label=87)

sns.kdeplot(merged_stat_transition.loc[(merged_stat_transition['hight']> hight_median),'num_teams'], color='b', fill=True)#, Label=87)
plt.title('Зависимость количества переходов и роста игрока')
plt.xlabel('Количество переходов')
plt.ylabel('Плотность вероятности');
```



Исследуем эту зависимость:

```
!pip install phik
```

```
Requirement already satisfied: phik in /usr/local/lib/python3.10/dist-packages (0.12.3)
Requirement already satisfied: numpy>=1.18.0 in /usr/local/lib/python3.10/dist-packages (from phik) (1.23.5)
Requirement already satisfied: scipy>=1.5.2 in /usr/local/lib/python3.10/dist-packages (from phik) (1.10.1)
Requirement already satisfied: pandas>=0.25.1 in /usr/local/lib/python3.10/dist-packages (from phik) (1.5.3)
Requirement already satisfied: matplotlib>=2.2.3 in /usr/local/lib/python3.10/dist-packages (from phik) (3.7.1)
Requirement already satisfied: joblib>=0.14.1 in /usr/local/lib/python3.10/dist-packages (from phik) (1.3.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.2.3->phik) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.2.3->phik) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.2.3->phik) (4.42.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.2.3->phik) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.2.3->phik) (23.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.2.3->phik) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.2.3->phik) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=2.2.3->phik) (2.8.
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.25.1->phik) (2023.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib>=2.2.3->p
```

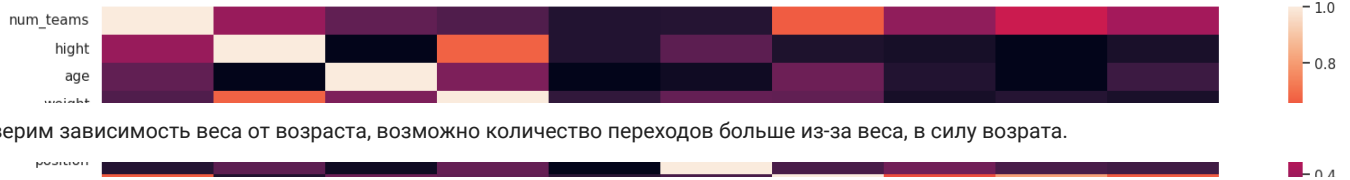
```
import phik
from phik.report import plot_correlation_matrix
from phik import report
```

```
data=stat2_new[['hight','age','weight','country', 'position','GP','G', 'Assists', 'PTS']]
```

```
data2=mergedd[['num_teams', 'hight','age','weight','country', 'position','GP','G', 'Assists', 'PTS']]
```

```
phik_overview = data2.phik_matrix()
sns.heatmap(phik_overview.round(2))
sns.set(rc = {'figure.figsize':(20,4)})
```

interval columns not set, guessing: ['num_teams', 'hight', 'weight', 'GP', 'G', 'Assists', 'PTS']



```
stat2_new['age'].unique()
```

```
array(['38', '30', '27', '24', '29', '43', '25', '28', '22', '21', '33',  
      '31', '35', '42', '50', '23', '34', '47', '46', '32', '37', '41',  
      '20', '40', '36', '7-сент.-11', '39', '53', '45', '52', '19', '44',  
      '26', '17', '49', '48', '51', '18', '13-окт.-08', '11-дек.-20',  
      '15-июл.-18', '65', '12-сент.-11', '58', '25-мая-23', '54',  
      '1-апр.-14', '11-авг.-22', '6-июл.-10', '5-окт.-19', '15-февр.-15',  
      '24-июн.-18', '26-сент.-20', '24-июл.-21', '22-июн.-23',  
      '7-январ.-16', '5-нояб.-16', '14-июн.-23', '16-июл.-20',  
      '21-дек.-15', '31-мар.-13', '11-мая-21', '59', '3-нояб.-21'],  
      dtype=object)
```

```
list_a = ['13-окт.-08', '11-дек.-20',  
         '15-июл.-18', '12-сент.-11', '25-мая-23',  
         '1-апр.-14', '11-авг.-22', '6-июл.-10', '5-окт.-19', '15-февр.-15',  
         '24-июн.-18', '26-сент.-20', '24-июл.-21', '22-июн.-23',  
         '7-январ.-16', '5-нояб.-16', '14-июн.-23', '16-июл.-20',  
         '21-дек.-15', '31-мар.-13', '11-мая-21', '3-нояб.-21', '7-сент.-11']
```

Записи возраста полной датой ('13-окт.-08') удалим. Так дана информация о дате смерти игрока.

```
stat2_new2 = stat2_new.query('age not in @list_a')  
stat2_new2['age'] = stat2_new2['age'].astype(int)  
stat2_new2.age.unique()
```

<ipython-input-336-4e9540e1cb08>:2: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus

```
array([38, 30, 27, 24, 29, 43, 25, 28, 22, 21, 33, 31, 35, 42, 50, 23, 34,  
      47, 46, 32, 37, 41, 20, 40, 36, 39, 53, 45, 52, 19, 44, 26, 17, 49,  
      48, 51, 18, 65, 58, 54, 59])
```

```
stat2_new2['age'].unique()
```

```
array([38, 30, 27, 24, 29, 43, 25, 28, 22, 21, 33, 31, 35, 42, 50, 23, 34,  
      47, 46, 32, 37, 41, 20, 40, 36, 39, 53, 45, 52, 19, 44, 26, 17, 49,  
      48, 51, 18, 65, 58, 54, 59])
```

Выделим два дата сета в зависимости от веса игрока:

```
weight_median_up = merged_stat_transition.loc[(merged_stat_transition['weight'] > weight_median), 'link']  
weight_median_down = merged_stat_transition.loc[(merged_stat_transition['weight'] <= weight_median), 'link']
```

```
w = stat2_new2.query('link in @weight_median_up')  
w2 = stat2_new2.query('link in @weight_median_down')
```

Найдем медианные значения возраста игроков больше 87 кг и меньше 87 кг.

```
w['age'].median(), w['age'].mean()
```

```
(34.0, 34.242476851851855)
```

Медианный возраст игроков, тяжелее 87 кг - 34 года

```
w2['age'].median(), w2['age'].mean()
```

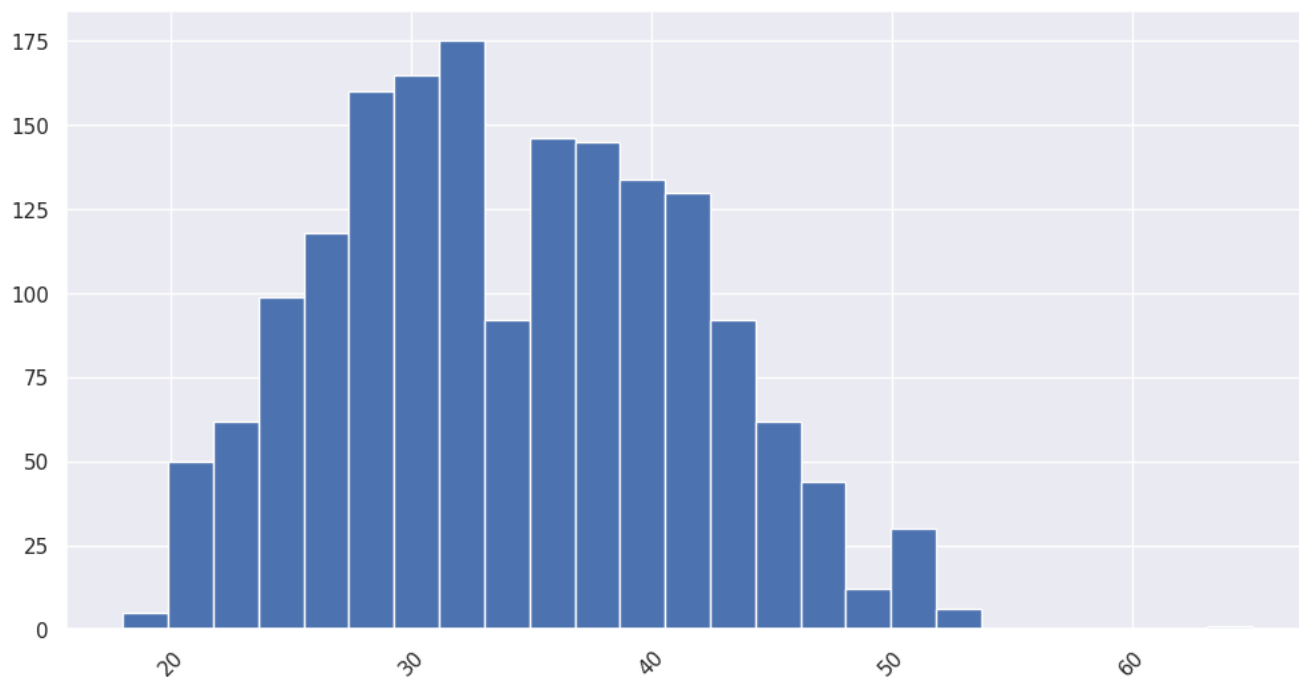
```
(30.0, 30.36382322713258)
```

Медианный возраст игроков, легче 87 кг - 30 лет

```
plt.figure(figsize=(12, 6))
w['age'].hist(bins=25)

plt.xticks(rotation=45) # Изменить наклон подписей на 45 градусов

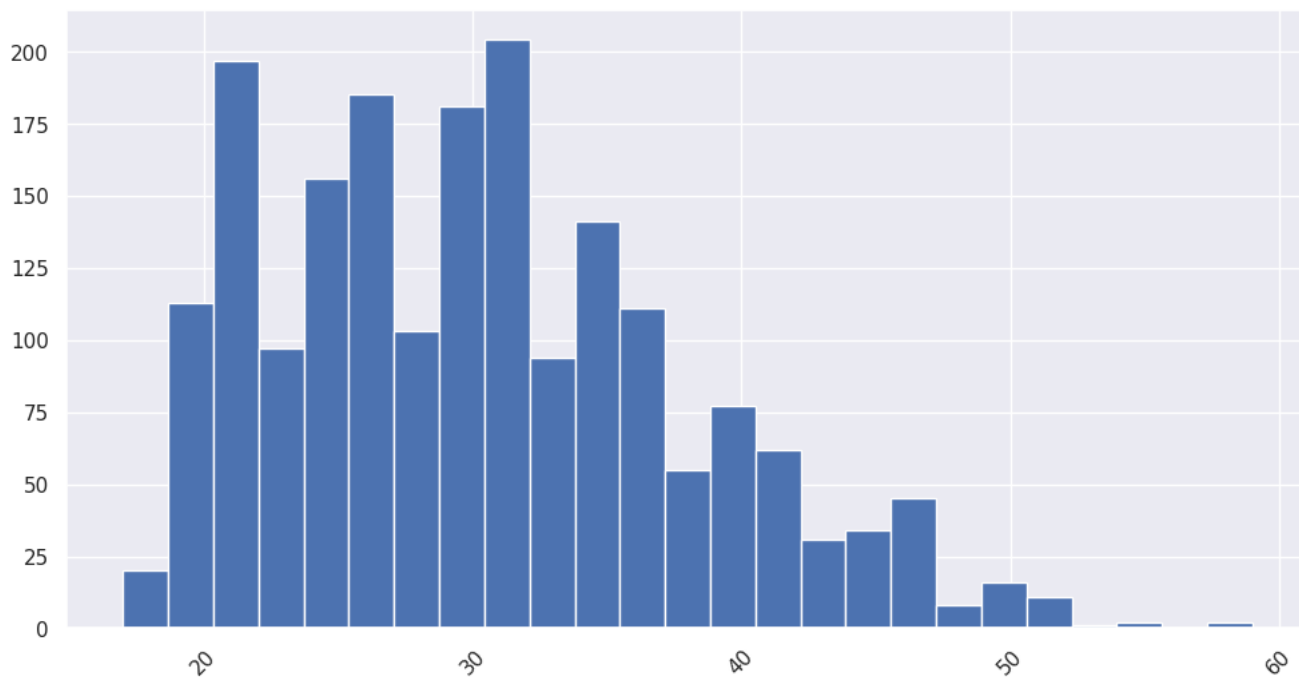
plt.show()
```



```
plt.figure(figsize=(12, 6))
w2['age'].hist(bins=25)

plt.xticks(rotation=45) # Изменить наклон подписей на 45 градусов

plt.show()
```



Можем сделать вывод, что с возрастом игрок становится тяжелее. Поэтому получили такую зависимость - вес влияет на количество переходов. Т.е. больший вес по медиане у игроков старше, которые дольше играют, а значит переходили из команды в команду, чаще.

▼ Проверить, верна ли гипотеза о том, что более успешные хоккеисты рождаются в январе-марте.

Возьмем для анализа нападающих. Проверим влияет ли месяц рождения на успешность - количество очков, заработанных в команде.

```
#удалим лишние столбцы
statistic = stat2_new.loc[(stat2_new['position']=='defense')]\
.drop(columns=['W','L','SOP','GA','Sv','%Sv','GAA','SO','hight','weight','shoot','player_link','TkA','\
'FOA','BLS','HITS','SFTSH/G','TISH/G','SFTE/G','TIIP/G','SFTPP/G','TIE/G','SFT/G','S/G','age','position'])
statistic.head()
```

	player	born	country	GP	G	Assists	PTS	+/-	+	-	...	OTG	GW
8	Kirill Ablayev	1- мар.-96	Russia	17	0	0	0.0	-2.0	1.0	3.0	...	0.0	0
13	Roman Abrosimov	31- июл.-94	Russia	235	10	23	33.0	-1.0	110.0	111.0	...	0.0	2
15	Kirill Adamchuk	24- мая-94	Russia	188	8	24	32.0	-4.0	88.0	92.0	...	0.0	0
18	Maxim A. Afonasov	11- авг.-98	Russia	92	2	11	13.0	8.0	39.0	31.0	...	0.0	0

```
# преобразуем дату рождения в формат data
statistic['born'] = statistic['born'].str.replace('янв.', '01', regex=True)
statistic['born'] = statistic['born'].str.replace('февр.', '02', regex=True)
statistic['born'] = statistic['born'].str.replace('мар.', '03', regex=True)
statistic['born'] = statistic['born'].str.replace('апр.', '04', regex=True)
statistic['born'] = statistic['born'].str.replace('мая', '05', regex=True)
statistic['born'] = statistic['born'].str.replace('июн.', '06', regex=True)
statistic['born'] = statistic['born'].str.replace('июл.', '07', regex=True)
statistic['born'] = statistic['born'].str.replace('авг.', '08', regex=True)
statistic['born'] = statistic['born'].str.replace('сент.', '09', regex=True)
statistic['born'] = statistic['born'].str.replace('окт.', '10', regex=True)
statistic['born'] = statistic['born'].str.replace('нояб.', '11', regex=True)
statistic['born'] = statistic['born'].str.replace('дек.', '12', regex=True)
statistic['born'] = pd.to_datetime(statistic['born'], format='%d-%m-%y')
```

```
# выделим месяц
statistic['mon'] = statistic['born'].dt.month
```

```
statistic.columns #расшифровка столбцов

Index(['player', 'born', 'country', 'GP', 'G', 'Assists', 'PTS', '+/-', '+',
      '-', 'PIM', 'ESG', 'PPG', 'SHG', 'OTG', 'GWG', 'SDS', 'SOG', '%SOG',
      'FO', 'FOW', '%FO', 'TOI/G', 'link', 'mon'],
      dtype='object')
```

- 'GP'-количество игр
- 'G'- количество голов
- 'Assists' - передачи
- 'PTS' - очки
- '+' - +/- очки
- '+' - очки
- '-' - очки
- 'PIM' - штрафное время
- 'ESG' - шайбы, заброшенные в равернске
- 'PPG' - шайбы, заброшенные в большинстве
- 'SHG' - шайбы, заброшенные в меньшенстве
- 'OTG' - шайбы, заброшенные в овертайме
- 'GWG' - победные голы
- 'SDS' - решающие буллиты
- 'SOG' - броски по воротам

'%SOG' - % реализованных бросков по воротам

'FO' - вбрасывания

'FOW' - выигранные вбрасывания

'%FO' - % выигранных вбрасываний

```
#statistic.head(50)
```

Зависимость от даты рождения очков игрока PTS:

Гипотеза H0: различий в очках PTS между игроками, которые родились с января по март, и игроками, которые родились с апреля по декабрь, нет.

Гипотеза H1: различия в очках PTS между игроками, которые родились с января по март, и игроками, которые родились с апреля по декабрь, есть.

```
sample_7 = statistic.query('mon==3 or mon==2 or mon==1')['PTS']
sample_8 = statistic.query('mon!=3 and mon!=2 and mon!=1')['PTS']
```

```
alpha = 0.05 # уровень статистической значимости
# если p-value окажется меньше него, отвергнем гипотезу
```

```
results = st.ttest_ind(sample_7, sample_8, equal_var=False)
```

```
print(results.statistic, 'p-значение:', results.pvalue)
```

```
if results.pvalue < alpha:
    print('Отвергаем нулевую гипотезу')
else:
    print('Не получилось отвергнуть нулевую гипотезу')
```

```
0.9097720790527537 p-значение: 0.36337034932180456
Не получилось отвергнуть нулевую гипотезу
```

Посмотрим на зависимость от даты рождения % реализованных бросков по воротам у игрока

Гипотеза H0: различий в % реализованных бросков по воротам между игроками, которые родились с января по март, и игроками, которые родились с апреля по декабрь, нет.

Гипотеза H1: различия в % реализованных бросков по воротам между игроками, которые родились с января по март, и игроками, которые родились с апреля по декабрь, есть.

```
statistic2 = statistic.loc[(statistic['%SOG']!= '-').copy()]#уберем из датасета данные с прочерками
```

```
statistic2['%SOG'] = statistic2['%SOG'].apply(lambda x: float(x.split()[0].replace(',','.')))#преобразуем тип столбца в числовой
```

```
sample_9 = statistic2.query('mon==3 or mon==2 or mon==1')['%SOG']
sample_10 = statistic2.query('mon!=3 and mon!=2 and mon!=1')['%SOG']
```

```
alpha = 0.05 # уровень статистической значимости
# если p-value окажется меньше него, отвергнем гипотезу
```

```
results = st.ttest_ind(sample_9, sample_10, equal_var=False)
```

```
print(results.statistic, 'p-значение:', results.pvalue)
```

```
if results.pvalue < alpha:
    print('Отвергаем нулевую гипотезу')
else:
    print('Не получилось отвергнуть нулевую гипотезу')
```

```
-0.8579308002989287 p-значение: 0.391269028175505
Не получилось отвергнуть нулевую гипотезу
```

Нельзя сказать, что есть зависимость % реализованных бросков по воротам от месяца рождения

Посмотрим в какие месяца родилось больше всего игроков (нападающие):

```
statistic['mon'].value_counts()
```



```

5      124
1      122
7      112
4      108
3      104
2      103
6       97
8       92
9       86
10      76
11      73
12      68
Name: mon, dtype: int64

```

Посмотрим месяц рождения всех игроков:

```

mon_stat = stat2_new

mon_stat['born'] = mon_stat['born'].str.replace('янв.', '01', regex=True)
mon_stat['born'] = mon_stat['born'].str.replace('февр.', '02', regex=True)
mon_stat['born'] = mon_stat['born'].str.replace('мар.', '03', regex=True)
mon_stat['born'] = mon_stat['born'].str.replace('апр.', '04', regex=True)
mon_stat['born'] = mon_stat['born'].str.replace('мая', '05', regex=True)
mon_stat['born'] = mon_stat['born'].str.replace('июн.', '06', regex=True)
mon_stat['born'] = mon_stat['born'].str.replace('июл.', '07', regex=True)
mon_stat['born'] = mon_stat['born'].str.replace('авг.', '08', regex=True)
mon_stat['born'] = mon_stat['born'].str.replace('сент.', '09', regex=True)
mon_stat['born'] = mon_stat['born'].str.replace('окт.', '10', regex=True)
mon_stat['born'] = mon_stat['born'].str.replace('нояб.', '11', regex=True)
mon_stat['born'] = mon_stat['born'].str.replace('дек.', '12', regex=True)
mon_stat['born'] = pd.to_datetime(mon_stat['born'], format='%d-%m-%y')

# выделим месяц
mon_stat['mon'] = mon_stat['born'].dt.month

```

```

mon_stat['mon'].value_counts()

1      422
3      363
4      356
2      354
5      353
6      329
7      326
8      285
9      256
10     235
11     233
12     205
Name: mon, dtype: int64

```

```

mon_stat.loc[(mon_stat['position']=='forward')]['mon'].value_counts()

1      252
2      206
3      200
5      188
4      188
6      185
7      171
8      159
9      137
10     131
11     130
12     114
Name: mon, dtype: int64

```

```

mon_stat.loc[(mon_stat['position']=='goaltender')]['mon'].value_counts()

4      60
3      59
1      48
6      47
2      45
7      43
5      41
8      34
9      33
11     30

```

```
10    28
12    23
Name: mon, dtype: int64
```

Нападающие, чаще всего рождались в мае и январе, защитники с января по март, вратари в марте и апреле. По всем игрокам видим, что наибольшее число родились в месяца первого полугодия. Возможно это из-за того, что при отборе детей в команды по возрасту, у детей которые родились раньше на несколько месяцев своих сверстников больше шансов, они старше, более развиты физически.

▼ Вывод:

- Мы обработали данные, полученные с сайта <https://en.khl.ru/>.
- Выделили одноклубников. Самым популярным хоккеистом оказался Евгений Лапенков, у него 475 одноклубников. У Николая Лемтюгова 357, он не вошел в топ-15 популярных игроков.
 - Рассчитали с помощью математического модели графа длину пути каждого игрока до Николая Лемтюгова. Каждый игрок знаком с ним максимум через 3 игроков. Нашли медианное и среднее значение числа Лемтюгова - 2 и 1.9 соответственно. У более молодых игроков среднее и медианное значения "числа Лемтюгова" больше, что логично, они в начале карьеры.
 - Проверка гипотез показала, что, чем меньше число Лемтюгова, тем в большем количестве команд играл хоккеист. Нет зависимости между количеством переходов и ростом игрока, но есть зависимость от веса игрока. Более тяжелые игроки чаще переходят из команды в команду. Так же не подтвердилась гипотеза о том, что игроки, рожденные с января по март более успешные.

✓ 0 сек. выполнено в 23:04



Не удается связаться с сервисом reCAPTCHA. Проверьте подключение к Интернету и перезагрузите страницу.