Машинное обучение

Практическое задание 1 посвящено изучению основных библиотек для анализа данных, а также линейных моделей и методов их обучения. Вы научитесь:

- применять библиотеки NumPy и Pandas для осуществления желаемых преобразований;
- подготавливать данные для обучения линейных моделей;
- обучать линейную, Lasso и Ridge-регрессии при помощи модуля scikit-learn;
- реализовывать обычный и стохастический градиентные спуски;
- обучать линейную регрессию для произвольного функционала качества.

Библиотеки для анализа данных

NumPy

Во всех заданиях данного раздела запрещено использовать циклы и list comprehensions. Под вектором и матрицей в данных заданиях понимается одномерный и двумерный numpy.array соответственно.

In [50]:

```
import numpy as np
```

Реализуйте функцию, возвращающую максимальный элемент в векторе x среди элементов, перед которыми стоит нулевой. Для x = np.array([6, 2, 0, 3, 0, 0, 5, 7, 0]) ответом является 5. Если нулевых элементов нет, функция должна возвращать None.

In [51]:

```
x = np.array([6, 2, 0, 3, 0, 0, 5, 7, 0])
def max_element(arr):
    # Your code here
    zero = arr == 0
    print(arr[1:][zero[:-1]].max())
max_element(x)
```

```
5
```

Реализуйте функцию, принимающую на вход матрицу и некоторое число и возвращающую ближайший к числу элемент матрицы. Например: для X = np.arange(0,10).reshape((2, 5)) и v = 3.6 ответом будет 4.

In [52]:

```
X = np.arange(0,10).reshape((2, 5))
v = 3.6
def nearest_value(X, v):
    # Your code here
    X = X.ravel()
    Xv = np.abs(X - v).argmin()
    return X[Xv]
print(nearest_value(X, v))
```

4

Реализуйте функцию scale(X), которая принимает на вход матрицу и масштабирует каждый ее столбец (вычитает выборочное среднее и делит на стандартное отклонение). Убедитесь, что в функции не будет происходить деления на ноль. Протестируйте на случайной матрице (для её генерации можно использовать, например, функцию numpy.random.randint (http://docs.scipy.org/doc/numpy/reference/generated/numpy.random.randint.html)).

In [53]:

```
def scale(X):
    # Your code here
    for i in range(len(X[0])):
        for j in range(len(X)):
            try:
                X[j,i]=(X[j,i]-np.mean(X[:,i]))/np.std(X[:,i])
            except:
                X[j,i] = 0
    print(X)
scale(np.random.randint(10, 20, size=(4, 4)))
```

```
[[0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]
 [1 1 1 1]]
```

Реализуйте функцию, которая для заданной матрицы находит:

- определитель
- след
- наименьший и наибольший элементы
- норму Фробениуса
- собственные числа
- обратную матрицу

Для тестирования сгенерируйте матрицу с элементами из нормального распределения $\mathcal{N}(10,1)$

In [54]:

```
N = np.random.normal(10,1,size=(2,2))
def get_stats(X):
    # Your code here
    print(np.linalg.det(X))
    print(np.linalg.matrix_rank(N))
    print(str(np.min(X))+'<-->'+str(np.max(X)))
    print(np.linalg.norm(X,ord='fro'))
    print(np.linalg.eig(X))
    print(np.linalg.inv(X))
    print(np.trace(X))
get_stats(N)
```

```
-1.5204339136460594
2
8.639517405342799<-->10.359178516323917
18.920864236485258
(array([-0.08069224, 18.84238196]), array([[-0.74369835, -0.73307913],
       [ 0.66851535, -0.68014336]]))
[[-6.26824228  6.81330403]
 [ 5.68227091 -6.07145204]]
18.76168972378538
```

Повторите 100 раз следующий эксперимент: сгенерируйте две матрицы размера 10×10 из стандартного нормального распределения, перемножьте их (как матрицы) и найдите максимальный элемент. Какое среднее значение по экспериментам у максимальных элементов? 95-процентная квантиль?

In [55]:

```
for exp_num in range(100):
    # Your code here
    N1 = np.random.normal(0,1,size=(10,10))
    N2 = np.random.normal(1,0,size=(10,10))
N = np.dot(N1,N2)
A = np.min(N)
B = np.max(N)
print(np.quantile(N,0.95))
```

```
3.3340584972788925
```

Pandas

Ответьте на вопросы о данных по авиарейсам в США за январь-апрель 2008 года.

Данные (https://www.dropbox.com/s/dvfitn93obn0rql/2008.csv?dl=0) и их описание (http://stat-computing.org/dataexpo/2009/the-data.html)

In [56]:

```
import pandas as pd
%matplotlib inline
df = pd.read_csv('2008.csv')
da = pd.read_csv('airports.csv')
```

Какая из причин отмены рейса (`CancellationCode`) была самой частой? (расшифровки кодов можно найти в описании данных)

In [57]:

```
# Your code here
print('A = carrier, B = weather, C = NAS, D = security')
print(df.groupby('CancellationCode')['Cancelled'].count())
```

```
A = carrier, B = weather, C = NAS, D = security
CancellationCode
A    563
B    549
C    299
Name: Cancelled, dtype: int64
```

Найдите среднее, минимальное и максимальное расстояние, пройденное самолетом.

In [58]:

```
# Your code here
df1 = df['Distance']
print(df1.sum()/len(df1))
print(df1.mean())
print(df1.min())
print(df1.max())
```

```
724.5082571428571
724.5082571428571
31
4962
```

Не выглядит ли подозрительным минимальное пройденное расстояние? В какие дни и на каких рейсах оно было? Какое расстояние было пройдено этими же рейсами в другие дни?

In [59]:

```
# Your code here
df2 = df[df.Distance == 31][['Year','Month','DayofMonth','FlightNum']]
print(df2)
print(df[(df.FlightNum == 64)|(df.FlightNum == 65)][['DayofMonth','Month','Year','Di
stance']].sort_values(by=['DayofMonth','Month','Year']))
```

	Year	Month	DayofMonth	FlightNum
1116	2008	12	30	65
6958	2008	12	26	65
17349	2008	8	18	64
27534	2008	3	11	64
46082	2008	8	9	65
48112	2008	2	28	64

	DayofMonth	Month	Year	Distance
26109	1	4	2008	571
3869	1	7	2008	82
48020	1	11	2008	1005
39438	2	5	2008	571
13155	2	7	2008	1747
57822	3	5	2008	1747
64319	3	5	2008	414
68338	3	8	2008	2454
9615	4	1	2008	533
30053	4	1	2008	82
64203	4	1	2008	82
65662	4	3	2008	123
69305	5	1	2008	1005
7891	6	4	2008	2381
12980	6	8	2008	82
54909	6	11	2008	581
68264	7	9	2008	386
33769	7	12	2008	1747
7977	9	7	2008	1747
46082	9	8	2008	31
52459	9	12	2008	581
8448	10	2	2008	123
66042	10	2	2008	372
1517	10	7	2008	680
47716	11	1	2008	281
33211	11	2	2008	1005
27534	11	3	2008	31
44810	12	2	2008	82
6778	12	7	2008	359
1389	13	3	2008	680
...	...	...	...	...
47168	19	7	2008	581
501	20	3	2008	533
24750	21	8	2008	680
66529	21	12	2008	82
63028	22	1	2008	1747
67172	22	3	2008	533
15173	22	10	2008	1005
32173	23	2	2008	1747
29801	23	4	2008	123
2619	23	5	2008	2381
45031	23	8	2008	82
32242	23	9	2008	82
10113	24	7	2008	571
43132	24	7	2008	123
50184	25	1	2008	372
59015	26	2	2008	1005
6958	26	12	2008	31
37350	27	5	2008	82
48112	28	2	2008	31
32956	28	5	2008	2454
52618	29	4	2008	680
4466	29	6	2008	123

```
43353          29      8  2008       571
14646          29     11  2008      2454
31375          30      3  2008      1005
41044          30      3  2008      1747
50888          30      9  2008        82
1116           30     12  2008        31
10833          31      3  2008       372
55053          31      3  2008       123

[78 rows x 4 columns]
```

Из какого аэропорта было произведено больше всего вылетов? В каком городе он находится?

In [60]:

```
# Your code here
a = df.groupby('Origin').count()[['Year']].sort_values(by=['Year'], ascending=False)
print(a.iloc[0])
print(da[da.iata == 'ATL'])
```

```
Year    4134
Name: ATL, dtype: int64
    iata                          airport     city state country
lat  \
880  ATL  William B Hartsfield-Atlanta Intl  Atlanta    GA     USA  33.6
40444

          long
880 -84.426944
```

In [61]:

```
# Your code here
print(df[['Origin','AirTime']].groupby('Origin').mean().sort_values(by=['AirTime'],
ascending=False).head(1))
```

```
        AirTime
Origin
SJU       205.2
```

Найдите аэропорт, у которого наибольшая доля задержанных (`DepDelay > 0`) рейсов. Исключите при этом из рассмотрения аэропорты, из которых было отправлено меньше 1000 рейсов (используйте функцию `filter` после `groupby`).

In [62]:

```
# Your code here
b = df[['Origin', 'DepDelay']].groupby(by=['Origin']).count().sort_values(by=['DepDelay'], ascending=False)
print(b[b.DepDelay > 1000])
```

```
        DepDelay
Origin
ATL         4079
ORD         3391
DFW         2730
DEN         2353
LAX         2064
PHX         2011
LAS         1773
IAH         1770
DTW         1588
SFO         1374
EWR         1343
MCO         1324
SLC         1313
MSP         1233
CLT         1229
JFK         1172
BOS         1128
SEA         1114
LGA         1114
BWI         1018
PHL         1007
```

Линейная регрессия

В этой части мы разберемся с линейной регрессией, способами её обучения и измерением качества ее прогнозов.

Будем рассматривать датасет из предыдущей части задания для предсказания времени задержки отправления рейса в минутах (DepDelay). Отметим, что под задержкой подразумевается не только опоздание рейса относительно планируемого времени вылета, но и отправление до планируемого времени.

Подготовка данных

12. (0.5 балла) Считайте выборку из файла при помощи функции pd.read_csv и ответьте на следующие вопросы:

- Имеются ли в данных пропущенные значения?
- Сколько всего пропущенных элементов в таблице "объект-признак"?
- Сколько объектов имеют хотя бы один пропуск?
- Сколько признаков имеют хотя бы одно пропущенное значение?

In [63]:

```
# Your code here
import pandas as pd
df = pd.read_csv('2008.csv')
print(df.isnull().values.any())
d = df.isnull().sum()
print('Сумма пропущенных значений =',d.sum())
print(np.count_nonzero(df.isnull().values.sum(axis=1)))
print(np.count_nonzero(df.isnull().values.sum(axis=0)))
```

```
True
Сумма пропущенных значений = 355215
70000
16
```

Как вы понимаете, также не имеет смысла рассматривать при решении поставленной задачи объекты с пропущенным значением целевой переменной. В связи с этим ответьте на следующие вопросы и выполните соответствующие действия:

- Имеются ли пропущенные значения в целевой переменной?
- Проанализируйте объекты с пропущенными значениями целевой переменной. Чем вызвано это явление? Что их объединяет? Можно ли в связи с этим, на ваш взгляд, исключить какие-то признаки из рассмотрения? Обоснуйте свою точку зрения.

Исключите из выборки объекты **с пропущенным значением целевой переменной и со значением целевой переменной, равным 0**, а также при необходимости исключите признаки в соответствии с вашим ответом на последний вопрос из списка и выделите целевую переменную в отдельный вектор, исключив её из матрицы "объект-признак".

In [64]:

```
# Your code here
print('Пропущенные значения в целевой переменной: ',df['DepDelay'].values.any())
nan = df[(df.DepDelay.isnull() == True)|(df.DepDelay != 0)]['DepDelay'].sum()
dfd = df['DepDelay']
df = df.drop(['DepDelay'], axis='columns')
df = df.fillna(0)
dfd = dfd.fillna(0)
```

```
Пропущенные значения в целевой переменной:  True
```

13. (0.5 балла) Обратите внимание, что признаки DepTime, CRSDepTime, ArrTime, CRSArrTime приведены в формате hhmm, в связи с чем будет не вполне корректно рассматривать их как вещественные.

Преобразуйте каждый признак FeatureName из указанных в пару новых признаков FeatureName_Hour, FeatureName_Minute, разделив каждое из значений на часы и минуты. Не забудьте при этом исключить исходный признак из выборки. В случае, если значение признака отсутствует, значения двух новых признаков, его заменяющих, также должны отсутствовать.

Например, признак DepTime необходимо заменить на пару признаков DepTime_Hour, DepTime_Minute. При этом, например, значение 155 исходного признака будет преобразовано в значения 1 и 55 признаков DepTime_Hour, DepTime_Minute соответственно.

In [65]:

```
# Your code here
for a in ['DepTime','CRSDepTime','ArrTime','CRSArrTime']:
    df[a+'_Hour'] = df[a]//100
    df[a+'_Minute'] = df[a]%100
df = df.drop(['DepTime','CRSDepTime','ArrTime','CRSArrTime'],axis='columns')
```

14. (0.5 балла) Некоторые из признаков, отличных от целевой переменной, могут оказывать чересчур значимое влияние на прогноз, поскольку по своему смыслу содержат большую долю информации о значении целевой переменной. Изучите описание датасета и исключите признаки, сильно коррелирующие с ответами. Ваш выбор признаков для исключения из выборки обоснуйте. Кроме того, исключите признаки TailNum и Year.

In [66]:

```
# Your code here
#Исключаются потому что имеют много пропусков
print(df.drop(['CancellationCode','CarrierDelay','WeatherDelay','NASDelay','Security
Delay','LateAircraftDelay'], 1))
#Исключаются по заданию
print(df.drop(['TailNum','Year'],1))
```

	Year	Month	DayofMonth	DayOfWeek	UniqueCarrier	FlightNum	TailNum
0	2008	6	18	3	WN	242	N699SW
1	2008	6	4	3	XE	2380	N15980
2	2008	8	3	7	WN	1769	N464WN
3	2008	1	23	3	OO	3802	N465SW
4	2008	5	4	7	WN	399	N489WN
5	2008	1	3	4	B6	834	N640JB
6	2008	3	31	1	WN	1589	N387SW
7	2008	4	22	2	DL	617	N3765
8	2008	3	4	2	WN	454	N633SW
9	2008	10	6	1	UA	270	N421UA
10	2008	11	16	7	WN	3596	N325SW
11	2008	3	6	4	AA	484	0
12	2008	12	23	2	FL	52	N316AT
13	2008	7	15	2	EV	4186	N722EV
14	2008	1	29	2	WN	3589	N510SW
15	2008	11	13	4	XE	2271	N14177
16	2008	7	5	6	UA	532	N826UA
17	2008	7	8	2	US	1632	N957UW
18	2008	8	17	7	WN	101	N910WN
19	2008	5	2	5	AA	2300	N5EMAA
20	2008	5	19	1	NW	1405	N765NC
21	2008	8	2	6	FL	15	N299AT
22	2008	1	14	1	CO	1015	N18611
23	2008	5	19	1	US	1528	N532AU
24	2008	5	24	6	AA	33	N336AA
25	2008	7	19	6	AA	1430	N503AA
26	2008	9	23	2	US	989	N438US
27	2008	1	22	2	HA	335	N481HA
28	2008	3	24	1	YV	7230	N37208
29	2008	3	24	1	YV	7293	N715

```
   SF
   ...     ...    ...         ...         ...           ...        ...
   ...
   69970  2008      7          10           4            DL       1735   N70
   4X
   69971  2008     12           7           7            WN       1047  N730
   SW
   69972  2008      4           9           3            US        183  N652
   AW
   69973  2008      3          22           6            HA        179  N481
   HA
   69974  2008      8          19           2            XE       2476  N175
   07
   69975  2008      4          20           7            WN       2194  N733
   SA
   69976  2008     12          31           3            EV       5060  N917
   EV
   69977  2008      4           6           7            US       1159  N443
   US
   69978  2008     12           9           2            9E       2276  8888
   9E
   69979  2008     12           2           2            EV       5008  N823
   AS
   69980  2008      5          12           1            AA        413  N5DC
   AA
   69981  2008      1           2           3            OH       5353  N917
   CA
   69982  2008      6          13           5            NW       1272  N329
   NB
   69983  2008      9           2           2            MQ       4506  N844
   AE
   69984  2008     10           4           6            MQ       3110  N848
   MQ
   69985  2008     12          11           4            UA         73  N820
   UA
   69986  2008     12           4           4            HA        392  N484
   HA
   69987  2008     11          15           6            XE       2463  N231
   39
   69988  2008     12           7           7            OH       6581  N916
   CA
   69989  2008      7          24           4            CO        284  N473
   32
   69990  2008     12          23           2            MQ       3092  N837
   MQ
   69991  2008      4          24           4            NW        241  N515
   US
   69992  2008     11           4           2            UA        885  N219
   UA
   69993  2008      2          28           4            MQ       3547  N939
   AE
   69994  2008      1          26           6            OH       5218  N655
   CA
   69995  2008      5          12           1            DL        794  N988
   DL
   69996  2008      5          11           7            OO       6159  N776
   SK
   69997  2008      9          24           3            YV       7058  N773
   31
   69998  2008      2          18           1            NW        641  N318
   US
```

```
69999  2008     12           6           6            WN        510  N786
SW

       ActualElapsedTime  CRSElapsedTime  AirTime  ...  Cancelled Divert
ed  \
0                   57.0            65.0     46.0  ...          0
0
1                  124.0           138.0    108.0  ...          0
0
2                  138.0           155.0    125.0  ...          0
0
3                  102.0           111.0     79.0  ...          0
0
4                  148.0           160.0    136.0  ...          0
0
5                  171.0           164.0    153.0  ...          0
0
6                   74.0            75.0     55.0  ...          0
0
7                  342.0           371.0    302.0  ...          0
0
8                  174.0           180.0    164.0  ...          0
0
9                  199.0           210.0    177.0  ...          0
0
10                 173.0           175.0    146.0  ...          0
0
11                   0.0           155.0      0.0  ...          1
0
12                 233.0           225.0    218.0  ...          0
0
13                  94.0            98.0     69.0  ...          0
0
14                  62.0            70.0     50.0  ...          0
0
15                 129.0           129.0    101.0  ...          0
0
16                 141.0           141.0    117.0  ...          0
0
17                 132.0            81.0     59.0  ...          0
0
18                 178.0           190.0    163.0  ...          0
0
19                 171.0           180.0    145.0  ...          0
0
20                  81.0            76.0     55.0  ...          0
0
21                 320.0           314.0    294.0  ...          0
0
22                  79.0            68.0     47.0  ...          0
0
23                  95.0           106.0     71.0  ...          0
0
24                 371.0           360.0    340.0  ...          0
0
25                 134.0           145.0    121.0  ...          0
0
26                 128.0           121.0    101.0  ...          0
0
27                  35.0            34.0     21.0  ...          0
0
```

```
28                105.0           115.0     86.0  ...          0
0
29                 66.0            90.0     47.0  ...          0
0
...                 ...             ...      ...  ...        ...
...
69970             122.0           128.0     92.0  ...          0
0
69971              87.0            95.0     70.0  ...          0
0
69972             248.0           270.0    226.0  ...          0
0
69973              30.0            29.0     18.0  ...          0
0
69974             117.0           122.0     86.0  ...          0
0
69975              49.0            55.0     41.0  ...          0
0
69976              71.0            85.0     55.0  ...          0
0
69977             146.0           142.0    120.0  ...          0
0
69978               0.0            74.0      0.0  ...          0
1
69979              47.0            50.0     31.0  ...          0
0
69980             264.0           255.0    243.0  ...          0
0
69981             100.0           108.0     84.0  ...          0
0
69982             130.0           139.0    108.0  ...          0
0
69983              61.0            75.0     41.0  ...          0
0
69984              52.0            45.0     23.0  ...          0
0
69985              90.0           105.0     63.0  ...          0
0
69986              51.0            52.0     37.0  ...          0
0
69987             132.0           120.0    100.0  ...          0
0
69988             115.0           113.0     97.0  ...          0
0
69989             202.0           229.0    187.0  ...          0
0
69990              47.0            45.0     26.0  ...          0
0
69991             201.0           191.0    169.0  ...          0
0
69992             273.0           278.0    251.0  ...          0
0
69993              51.0            55.0     32.0  ...          0
0
69994             110.0           113.0     85.0  ...          0
0
69995             128.0           129.0     86.0  ...          0
0
69996              47.0            49.0     30.0  ...          0
0
69997              80.0            80.0     63.0  ...          0
```

```
0
69998              234.0           219.0    192.0  ...          0
0
69999               60.0            65.0     46.0  ...          0
0

      DepTime_Hour  DepTime_Minute  CRSDepTime_Hour  CRSDepTime_Minute
\
0             21.0            11.0               20                 55
1             14.0            26.0               14                 10
2             11.0            43.0               11                 45
3             11.0            41.0               11                 44
4              8.0            15.0                8                 20
5             13.0            49.0               13                 25
6             13.0            59.0               14                  0
7             18.0            15.0               18                 20
8              6.0            26.0                6                 30
9             19.0             5.0               19                  7
10            20.0            42.0               20                  0
11             0.0             0.0               16                  5
12            11.0             1.0               10                 50
13            13.0            48.0               13                 55
14            15.0            28.0               15                 10
15            16.0            51.0               17                  0
16             5.0            57.0                6                  0
17            19.0            51.0               19                 50
18            20.0             5.0               19                 45
19            10.0            44.0               10                 45
20             9.0             6.0                9                 13
21             9.0            42.0                9                 45
22            14.0            42.0               14                 45
23            20.0            32.0               20                  0
24             7.0            41.0                7                 45
25             9.0             1.0                9                  0
26             7.0            52.0                8                  0
27            16.0            39.0               16                 40
28            16.0            45.0               16                 45
29             8.0            11.0                8                 15
...             ...             ...              ...                ...
69970          9.0            37.0                9                 35
69971         22.0             1.0               21                 50
69972          8.0            55.0                8                 45
69973         13.0             3.0               13                  6
69974         12.0            59.0               13                  0
69975         18.0            40.0               18                 10
69976         20.0            11.0               20                 12
69977          8.0            40.0                8                 35
69978          7.0            26.0                7                 31
69979          9.0             8.0                9                 15
69980         11.0            55.0               11                 40
69981         11.0            22.0               10                 48
69982         13.0            51.0               14                  0
69983         13.0            51.0               13                 55
69984          6.0             5.0                6                 15
69985          6.0            15.0                6                 18
69986         19.0             1.0               19                  5
69987          6.0            51.0                6                 59
69988         13.0            24.0               13                 25
69989          0.0            28.0                0                 30
69990         16.0            27.0               16                 30
69991          9.0            14.0                9                 20
```

69992	7.0	54.0	8	1
69993	14.0	10.0	14	0
69994	9.0	20.0	9	25
69995	18.0	29.0	18	40
69996	11.0	49.0	11	25
69997	10.0	12.0	10	12
69998	19.0	6.0	19	0
69999	8.0	59.0	9	0

	ArrTime_Hour	ArrTime_Minute	CRSArrTime_Hour	CRSArrTime_Minute
0	23.0	8.0	23	0
1	17.0	30.0	17	28
2	15.0	1.0	15	20
3	13.0	23.0	13	35
4	12.0	43.0	13	0
5	16.0	40.0	16	9
6	14.0	13.0	14	15
7	20.0	57.0	21	31
8	8.0	20.0	8	30
9	0.0	24.0	0	37
10	0.0	35.0	23	55
11	0.0	0.0	19	40
12	15.0	54.0	15	35
13	15.0	22.0	15	33
14	16.0	30.0	16	20
15	19.0	0.0	19	9
16	9.0	18.0	9	21
17	22.0	3.0	21	11
18	1.0	3.0	0	55
19	15.0	35.0	15	45
20	9.0	27.0	9	29
21	12.0	2.0	11	59
22	16.0	1.0	15	53
23	22.0	7.0	21	46
24	10.0	52.0	10	45
25	12.0	15.0	12	25
26	10.0	0.0	10	1
27	17.0	14.0	17	14
28	19.0	30.0	19	40
29	9.0	17.0	9	45
...	...	...	...	...
69970	10.0	39.0	10	43
69971	23.0	28.0	23	25
69972	16.0	3.0	16	15
69973	13.0	33.0	13	35
69974	14.0	56.0	15	2
69975	19.0	29.0	19	5
69976	21.0	22.0	21	37
69977	11.0	6.0	10	57
69978	0.0	0.0	7	45
69979	10.0	55.0	11	5
69980	14.0	19.0	13	55
69981	13.0	2.0	12	36
69982	17.0	1.0	17	19
69983	13.0	52.0	14	10
69984	6.0	57.0	7	0
69985	7.0	45.0	8	3
69986	19.0	52.0	19	57
69987	9.0	3.0	8	59
69988	14.0	19.0	14	18
69989	5.0	50.0	6	19

```
69990          17.0            14.0                17                15
69991          12.0            35.0                12                31
69992          10.0            27.0                10                39
69993          15.0             1.0                14                55
69994          11.0            10.0                11                18
69995          21.0            37.0                21                49
69996          13.0            36.0                13                14
69997          11.0            32.0                11                32
69998          22.0             0.0                21                39
69999           9.0            59.0                10                 5

[70000 rows x 26 columns]
       Month  DayofMonth  DayOfWeek UniqueCarrier  FlightNum  \
0          6          18          3             WN        242
1          6           4          3             XE       2380
2          8           3          7             WN       1769
3          1          23          3             OO       3802
4          5           4          7             WN        399
5          1           3          4             B6        834
6          3          31          1             WN       1589
7          4          22          2             DL        617
8          3           4          2             WN        454
9         10           6          1             UA        270
10        11          16          7             WN       3596
11         3           6          4             AA        484
12        12          23          2             FL         52
13         7          15          2             EV       4186
14         1          29          2             WN       3589
15        11          13          4             XE       2271
16         7           5          6             UA        532
17         7           8          2             US       1632
18         8          17          7             WN        101
19         5           2          5             AA       2300
20         5          19          1             NW       1405
21         8           2          6             FL         15
22         1          14          1             CO       1015
23         5          19          1             US       1528
24         5          24          6             AA         33
25         7          19          6             AA       1430
26         9          23          2             US        989
27         1          22          2             HA        335
28         3          24          1             YV       7230
29         3          24          1             YV       7293
...      ...         ...        ...            ...        ...
69970      7          10          4             DL       1735
69971     12           7          7             WN       1047
69972      4           9          3             US        183
69973      3          22          6             HA        179
69974      8          19          2             XE       2476
69975      4          20          7             WN       2194
69976     12          31          3             EV       5060
69977      4           6          7             US       1159
69978     12           9          2             9E       2276
69979     12           2          2             EV       5008
69980      5          12          1             AA        413
69981      1           2          3             OH       5353
69982      6          13          5             NW       1272
69983      9           2          2             MQ       4506
69984     10           4          6             MQ       3110
69985     12          11          4             UA         73
69986     12           4          4             HA        392
```

69987	11	15	6	XE	2463
69988	12	7	7	OH	6581
69989	7	24	4	CO	284
69990	12	23	2	MQ	3092
69991	4	24	4	NW	241
69992	11	4	2	UA	885
69993	2	28	4	MQ	3547
69994	1	26	6	OH	5218
69995	5	12	1	DL	794
69996	5	11	7	OO	6159
69997	9	24	3	YV	7058
69998	2	18	1	NW	641
69999	12	6	6	WN	510

	ActualElapsedTime	CRSElapsedTime	AirTime	ArrDelay	Origin	... \
0	57.0	65.0	46.0	8.0	MDW	...
1	124.0	138.0	108.0	2.0	IAH	...
2	138.0	155.0	125.0	-19.0	MDW	...
3	102.0	111.0	79.0	-12.0	SLC	...
4	148.0	160.0	136.0	-17.0	LAS	...
5	171.0	164.0	153.0	31.0	PBI	...
6	74.0	75.0	55.0	-2.0	ABQ	...
7	342.0	371.0	302.0	-34.0	JFK	...
8	174.0	180.0	164.0	-10.0	MCO	...
9	199.0	210.0	177.0	-13.0	DEN	...
10	173.0	175.0	146.0	40.0	MDW	...
11	0.0	155.0	0.0	0.0	DFW	...
12	233.0	225.0	218.0	19.0	BWI	...
13	94.0	98.0	69.0	-11.0	ATL	...
14	62.0	70.0	50.0	10.0	PHL	...
15	129.0	129.0	101.0	-9.0	MCI	...
16	141.0	141.0	117.0	-3.0	DEN	...
17	132.0	81.0	59.0	52.0	CLT	...
18	178.0	190.0	163.0	8.0	LAS	...
19	171.0	180.0	145.0	-10.0	LAX	...
20	81.0	76.0	55.0	-2.0	DTW	...
21	320.0	314.0	294.0	3.0	ATL	...
22	79.0	68.0	47.0	8.0	IAH	...
23	95.0	106.0	71.0	21.0	CLT	...
24	371.0	360.0	340.0	7.0	JFK	...
25	134.0	145.0	121.0	-10.0	DFW	...
26	128.0	121.0	101.0	-1.0	CLT	...
27	35.0	34.0	21.0	0.0	OGG	...
28	105.0	115.0	86.0	-10.0	ORD	...
29	66.0	90.0	47.0	-28.0	IAD	...
...	...	...	...	...	...	...
69970	122.0	128.0	92.0	-4.0	SLC	...
69971	87.0	95.0	70.0	3.0	SAN	...
69972	248.0	270.0	226.0	-12.0	PHX	...
69973	30.0	29.0	18.0	-2.0	KOA	...
69974	117.0	122.0	86.0	-6.0	EWR	...
69975	49.0	55.0	41.0	24.0	ONT	...
69976	71.0	85.0	55.0	-15.0	ATL	...
69977	146.0	142.0	120.0	9.0	DCA	...
69978	0.0	74.0	0.0	0.0	CHA	...
69979	47.0	50.0	31.0	-10.0	MGM	...
69980	264.0	255.0	243.0	24.0	ORD	...
69981	100.0	108.0	84.0	26.0	ROC	...
69982	130.0	139.0	108.0	-18.0	BZN	...
69983	61.0	75.0	41.0	-18.0	DTW	...

69984	52.0	45.0	23.0	-3.0	SAN	...
69985	90.0	105.0	63.0	-18.0	SAN	...
69986	51.0	52.0	37.0	-5.0	HNL	...
69987	132.0	120.0	100.0	4.0	EWR	...
69988	115.0	113.0	97.0	1.0	CVG	...
69989	202.0	229.0	187.0	-29.0	OAK	...
69990	47.0	45.0	26.0	-1.0	SAN	...
69991	201.0	191.0	169.0	4.0	FLL	...
69992	273.0	278.0	251.0	-12.0	ORD	...
69993	51.0	55.0	32.0	6.0	DFW	...
69994	110.0	113.0	85.0	-8.0	CVG	...
69995	128.0	129.0	86.0	-12.0	ORD	...
69996	47.0	49.0	30.0	22.0	ORD	...
69997	80.0	80.0	63.0	0.0	DAY	...
69998	234.0	219.0	192.0	21.0	RSW	...
69999	60.0	65.0	46.0	-6.0	LAS	...

	SecurityDelay	LateAircraftDelay	DepTime_Hour	DepTime_Minute	\
0	0.0	0.0	21.0	11.0	
1	0.0	0.0	14.0	26.0	
2	0.0	0.0	11.0	43.0	
3	0.0	0.0	11.0	41.0	
4	0.0	0.0	8.0	15.0	
5	0.0	11.0	13.0	49.0	
6	0.0	0.0	13.0	59.0	
7	0.0	0.0	18.0	15.0	
8	0.0	0.0	6.0	26.0	
9	0.0	0.0	19.0	5.0	
10	0.0	0.0	20.0	42.0	
11	0.0	0.0	0.0	0.0	
12	0.0	0.0	11.0	1.0	
13	0.0	0.0	13.0	48.0	
14	0.0	0.0	15.0	28.0	
15	0.0	0.0	16.0	51.0	
16	0.0	0.0	5.0	57.0	
17	0.0	0.0	19.0	51.0	
18	0.0	0.0	20.0	5.0	
19	0.0	0.0	10.0	44.0	
20	0.0	0.0	9.0	6.0	
21	0.0	0.0	9.0	42.0	
22	0.0	0.0	14.0	42.0	
23	0.0	21.0	20.0	32.0	
24	0.0	0.0	7.0	41.0	
25	0.0	0.0	9.0	1.0	
26	0.0	0.0	7.0	52.0	
27	0.0	0.0	16.0	39.0	
28	0.0	0.0	16.0	45.0	
29	0.0	0.0	8.0	11.0	
...	...	...	...	...	
69970	0.0	0.0	9.0	37.0	
69971	0.0	0.0	22.0	1.0	
69972	0.0	0.0	8.0	55.0	
69973	0.0	0.0	13.0	3.0	
69974	0.0	0.0	12.0	59.0	
69975	1.0	23.0	18.0	40.0	
69976	0.0	0.0	20.0	11.0	
69977	0.0	0.0	8.0	40.0	
69978	0.0	0.0	7.0	26.0	
69979	0.0	0.0	9.0	8.0	
69980	0.0	0.0	11.0	55.0	
69981	0.0	0.0	11.0	22.0	

69982	0.0	0.0	13.0	51.0
69983	0.0	0.0	13.0	51.0
69984	0.0	0.0	6.0	5.0
69985	0.0	0.0	6.0	15.0
69986	0.0	0.0	19.0	1.0
69987	0.0	0.0	6.0	51.0
69988	0.0	0.0	13.0	24.0
69989	0.0	0.0	0.0	28.0
69990	0.0	0.0	16.0	27.0
69991	0.0	0.0	9.0	14.0
69992	0.0	0.0	7.0	54.0
69993	0.0	0.0	14.0	10.0
69994	0.0	0.0	9.0	20.0
69995	0.0	0.0	18.0	29.0
69996	0.0	22.0	11.0	49.0
69997	0.0	0.0	10.0	12.0
69998	0.0	6.0	19.0	6.0
69999	0.0	0.0	8.0	59.0

	CRSDepTime_Hour	CRSDepTime_Minute	ArrTime_Hour	ArrTime_Minute
0	20	55	23.0	8.0
1	14	10	17.0	30.0
2	11	45	15.0	1.0
3	11	44	13.0	23.0
4	8	20	12.0	43.0
5	13	25	16.0	40.0
6	14	0	14.0	13.0
7	18	20	20.0	57.0
8	6	30	8.0	20.0
9	19	7	0.0	24.0
10	20	0	0.0	35.0
11	16	5	0.0	0.0
12	10	50	15.0	54.0
13	13	55	15.0	22.0
14	15	10	16.0	30.0
15	17	0	19.0	0.0
16	6	0	9.0	18.0
17	19	50	22.0	3.0
18	19	45	1.0	3.0
19	10	45	15.0	35.0
20	9	13	9.0	27.0
21	9	45	12.0	2.0
22	14	45	16.0	1.0
23	20	0	22.0	7.0
24	7	45	10.0	52.0
25	9	0	12.0	15.0
26	8	0	10.0	0.0
27	16	40	17.0	14.0
28	16	45	19.0	30.0
29	8	15	9.0	17.0
...	...	...	...	...
69970	9	35	10.0	39.0
69971	21	50	23.0	28.0
69972	8	45	16.0	3.0
69973	13	6	13.0	33.0
69974	13	0	14.0	56.0
69975	18	10	19.0	29.0
69976	20	12	21.0	22.0
69977	8	35	11.0	6.0
69978	7	31	0.0	0.0

69979	9	15	10.0	55.0
69980	11	40	14.0	19.0
69981	10	48	13.0	2.0
69982	14	0	17.0	1.0
69983	13	55	13.0	52.0
69984	6	15	6.0	57.0
69985	6	18	7.0	45.0
69986	19	5	19.0	52.0
69987	6	59	9.0	3.0
69988	13	25	14.0	19.0
69989	0	30	5.0	50.0
69990	16	30	17.0	14.0
69991	9	20	12.0	35.0
69992	8	1	10.0	27.0
69993	14	0	15.0	1.0
69994	9	25	11.0	10.0
69995	18	40	21.0	37.0
69996	11	25	13.0	36.0
69997	10	12	11.0	32.0
69998	19	0	22.0	0.0
69999	9	0	9.0	59.0

	CRSArrTime_Hour	CRSArrTime_Minute
0	23	0
1	17	28
2	15	20
3	13	35
4	13	0
5	16	9
6	14	15
7	21	31
8	8	30
9	0	37
10	23	55
11	19	40
12	15	35
13	15	33
14	16	20
15	19	9
16	9	21
17	21	11
18	0	55
19	15	45
20	9	29
21	11	59
22	15	53
23	21	46
24	10	45
25	12	25
26	10	1
27	17	14
28	19	40
29	9	45
...	...	...
69970	10	43
69971	23	25
69972	16	15
69973	13	35
69974	15	2
69975	19	5
69976	21	37

```
69977          10          57
69978           7          45
69979          11           5
69980          13          55
69981          12          36
69982          17          19
69983          14          10
69984           7           0
69985           8           3
69986          19          57
69987           8          59
69988          14          18
69989           6          19
69990          17          15
69991          12          31
69992          10          39
69993          14          55
69994          11          18
69995          21          49
69996          13          14
69997          11          32
69998          21          39
69999          10           5

[70000 rows x 30 columns]
```

15. (1 балл) Приведем данные к виду, пригодному для обучения линейных моделей. Для этого вещественные признаки надо отмасштабировать, а категориальные — привести к числовому виду. Также надо устранить пропуски в данных.

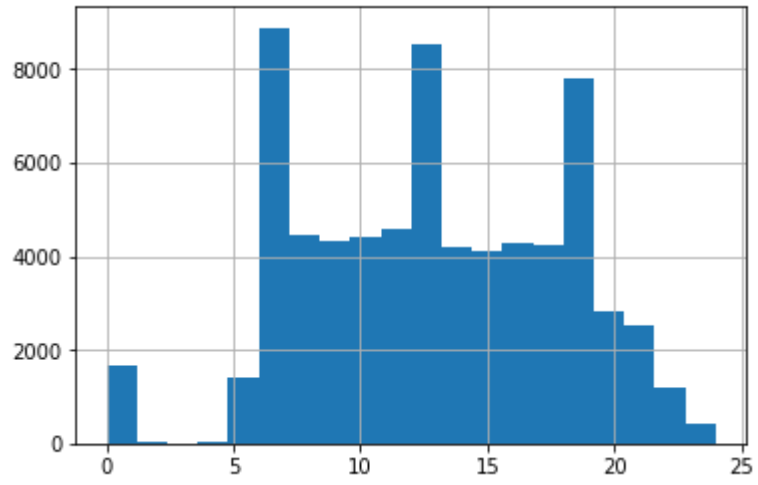
В первую очередь поймем, зачем необходимо применять масштабирование. Следующие ячейки с кодом построят гистограммы для 3 вещественных признаков выборки.

In [67]:

```
df['DepTime_Hour'].hist(bins=20)
```

Out[67]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x20717c71710>
```

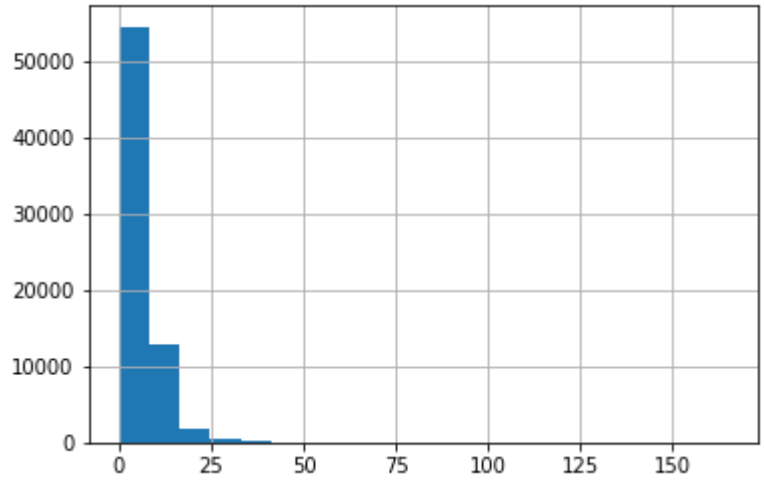


In [68]:

```
df['TaxiIn'].hist(bins=20)
```

Out[68]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x20716e21c18>
```

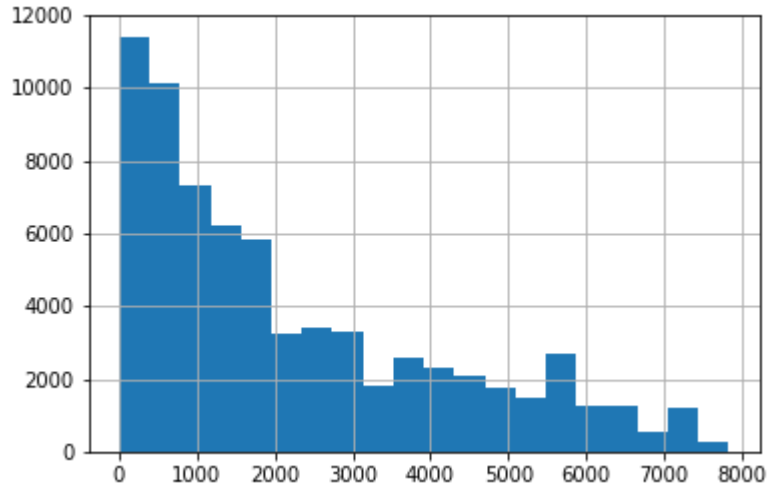


In [69]:

```
df['FlightNum'].hist(bins=20)
```

Out[69]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x20717c29cc0>
```



Какую проблему вы наблюдаете на этих графиках? Как масштабирование поможет её исправить?

Разный разброс единиц измерения.

Некоторые из признаков в нашем датасете являются категориальными. Типичным подходом к работе с ними является бинарное, или one-hot-кодирование (https://en.wikipedia.org/wiki/One-hot).

Реализуйте функцию transform_data, которая принимает на вход DataFrame с признаками и выполняет следующие шаги:

1. Замена пропущенных значений на нули для вещественных признаков и на строки 'nan' для категориальных.
2. Масштабирование вещественных признаков с помощью StandardScaler (http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html).
3. One-hot-кодирование категориальных признаков с помощью DictVectorizer (http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.DictVectorizer.html) или функции pd.get_dummies (http://pandas.pydata.org/pandas-docs/stable/generated/pandas.get_dummies.html).

Метод должен возвращать преобразованный DataFrame, который должна состоять из масштабированных вещественных признаков и закодированных категориальных (исходные признаки должны быть исключены из выборки).

In [70]:

```
from sklearn.preprocessing import StandardScaler as ss
def transform_data(data):
    # Your code here
    for c in data.columns:
        if data[c].dtype.name == 'object':
            data[c].fillna('nan')
        else:
            data[c].fillna(0)
    data = pd.get_dummies(data)
    return data
```

Примените функцию transform_data к данным. Сколько признаков получилось после преобразования?

In [71]:

```
# Your code here
df = transform_data(df)
```

16. (0.5 балла) Разбейте выборку и вектор целевой переменной на обучение и контроль в отношении 70/30 (для этого можно использовать, например, функцию train_test_split (http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.train_test_split.html)).

In [72]:

```
# Your code here
from sklearn.model_selection import train_test_split as tts
df_train, df_test, dfd_train, dfd_test = tts(df,dfd,test_size=0.3, random_state=42)
```

Scikit-learn

Теперь, когда мы привели данные к пригодному виду, попробуем решить задачу при помощи метода наименьших квадратов. Напомним, что данный метод заключается в оптимизации функционала MSE:

$$MSE(X, y) = \frac{1}{l}\sum_{i=1}^{l}(< w, x_i > -y_i)^2 \to \min_w,$$

где $\{(x_i, y_i)\}_{i=1}^{l}$ — обучающая выборка, состоящая из l пар объект-ответ.

Заметим, что решение данной задачи уже реализовано в модуле sklearn в виде класса LinearRegression (http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.Linea

17. (0.5 балла) Обучите линейную регрессию на 1000 объектах из обучающей выборки и выведите значения MSE и R^2 на этой подвыборке и контрольной выборке (итого 4 различных числа). Проинтерпретируйте полученный результат — насколько качественные прогнозы строит полученная модель? Какие проблемы наблюдаются в модели?

Подсказка: изучите значения полученных коэффициентов w, сохраненных в атрибуте coef_ объекта LinearRegression.

In [73]:

```
# Your code here
from sklearn.linear_model import LinearRegression as lr
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import r2_score as r2
df_train2, df_test2, dfd_train2, dfd_test2 = df_train[:1000], df_test[:1000], dfd_tr
ain[:1000], dfd_test[:1000]
reg = lr().fit(df_train2, dfd_train2)
print(reg.score(df_train2,dfd_train2))
print(reg.intercept_)
print(reg.coef_)
df_pred = reg.predict(df_train2)
tst_pred = reg.predict(df_test2)
print(mse(dfd_train2, df_pred), r2(dfd_train2, df_pred))
print(mse(dfd_test2, tst_pred), r2(dfd_test2, tst_pred))
```

```
1.0
-4.277973879678958
[-1.16701229e-03 -5.09156273e-01 -9.47968736e-03 ... -5.43689487e+00
 -7.18277818e+00  5.01179481e+00]
6.460137412647939e-23 1.0
175.5460895096415 0.905406190627801
```

Для решения описанных вами в предыдущем пункте проблем используем L1- или L2-регуляризацию, тем самым получив Lasso и Ridge регрессии соответственно и изменив оптимизационную задачу одним из следующих образов:

$$MSE_{L1}(X, y) = \frac{1}{l}\sum_{i=1}^{l}(< w, x_i > -y_i)^2 + \alpha||w||_1 \to \min_w,$$

$$MSE_{L2}(X, y) = \frac{1}{l}\sum_{i=1}^{l}(< w, x_i > -y_i)^2 + \alpha||w||_2^2 \to \min_w,$$

где α — коэффициент регуляризации. Один из способов его подбора заключается в переборе некоторого количества значений и оценке качества на кросс-валидации для каждого из них, после чего выбирается значение, для которого было получено наилучшее качество.

18. (0.5 балла) Обучите линейные регрессии с L1- и L2-регуляризатором, подобрав лучшее значение параметра регуляризации из списка alpha_grid при помощи кросс-валидации с 5 фолдами на тех же 1000 объектах, что и в п.17. Выведите значения MSE и R^2 на обучающей и контрольной выборках. Удалось ли решить указанные вами ранее проблемы?

Для выполнения данного задания вам могут понадобиться реализованные в библиотеке объекты LassoCV (http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LassoCV.html), RidgeCV (http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeCV.html) и KFold (http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.KFold.html).

In [74]:

```
# Your code here
from sklearn.linear_model import LassoCV
from sklearn.linear_model import RidgeCV
from sklearn.model_selection import KFold
cv = KFold(n_splits=2)
r = LassoCV(cv=2, random_state=42).fit(df_train2, dfd_train2)
print(r.score(df_train2, dfd_train2))
print(r.predict(df_train2))
clf = RidgeCV(alphas=[1, 1, 1, 1]).fit(df_train2, dfd_train2)
print(clf.score(df_train2, dfd_train2))
```

```
0.9462942091415065
[ 1.15829627e+02 -7.00511086e+00 -7.55324428e+00 -1.33025612e+01
  7.95707463e-01 -3.66960114e-02 -4.13391257e+00  6.04455784e+00
  2.34295769e+01 -1.60448652e+00  7.30400994e+01  3.30761804e+01
  9.28975507e+00  1.01635558e+02  1.59843164e+01 -2.55639622e+00
  1.92393752e+00  6.61891913e+00 -6.98780927e-01  8.72741906e+00
 -4.63226907e+00 -2.43018952e+00 -5.78189806e+00  5.63993980e+00
 -3.70150237e+00 -2.46170583e+00 -7.04190020e+00 -4.39843828e+00
 -2.20234640e+00 -1.56572392e+00  7.10504779e-01  1.47261157e+00
 -4.21298893e+00 -5.00369229e+00 -3.13854057e+00 -3.41864127e-01
  9.18000104e+00 -7.76344264e+00 -2.75044153e+00  1.85464937e-01
  8.48979853e+01  9.98122800e+00  4.19459941e+01  1.57052589e+00
  1.22766300e+01 -4.68902530e+00  1.29890890e+00  2.63262311e+02
 -7.53687481e+00 -5.43187544e+00 -2.46124267e+00  2.55671475e+01
  6.79120633e+00  2.76467348e+01 -2.34010670e+00  8.08835540e+00
 -3.74677317e+00 -5.46056300e+00  1.03762917e+02 -5.56005205e+00
 -1.02388060e+00  4.50020159e+01 -7.06873388e+00  4.09539499e+01
  1.41592647e+01 -1.85183467e+00 -4.19235494e-01  2.53792140e+01
 -7.12895595e+00  3.46562107e+01  3.48538389e+01 -1.46157296e+00
  8.84334316e+00 -2.95079781e+00 -7.53848947e+00  3.35212054e+00
 -4.15457144e-01 -1.71520128e+00 -9.51769050e-01  3.67200709e+01
  4.40864636e+01  1.47828183e+00  1.48610240e+01 -1.22722435e+00
  1.34413353e+01 -6.72658652e+00  7.32141585e+00 -2.74131947e+00
  2.22845581e+01  1.89713718e+01 -2.08670514e+00  2.59382288e+01
  8.57171728e+00  3.21217015e+00  7.48914990e+01  1.65357185e+01
  3.89108613e+01 -1.70815720e-01  5.99496631e-01  7.01307546e+00
  9.39288981e-01 -5.26897567e+00  1.42193180e+01  2.14501182e+01
  7.37314402e+00 -7.21467964e+00  1.07375442e+01  9.78687495e+01
 -1.74542632e+00  1.65953127e+01  2.30693461e+01 -8.77975155e+00
 -7.01435085e+00  1.82164709e+01  1.20825286e+00 -3.24268581e+00
 -6.47101364e+00 -9.99278177e-01 -2.50457571e+00  1.10221118e+01
  6.31916136e+01  1.58261809e+01  9.96720830e-01 -3.53997879e+00
 -9.12523452e-01 -3.57279654e+00  7.55389755e+01 -7.25898200e+00
 -8.74416258e+00  6.24627647e-01 -1.08503748e+01  1.15336150e+00
 -2.24954956e+00  8.52657359e+00 -2.85683202e+00 -4.80882866e+00
 -5.64911761e+00 -8.54265914e+00  9.81431616e+01  1.66676557e+00
 -2.05446437e+00 -4.53723775e+00  4.36259677e-01  6.96659347e+00
  5.53629406e+00 -2.21101113e+00  1.26518928e+00  4.76785436e+00
  2.15111232e+01 -2.27265747e+00  6.11237088e+00 -1.47239067e+00
  2.08077913e+01 -2.41450191e+00 -2.47572414e+00  1.33269994e+00
 -4.54928080e+00  4.16105701e+01 -1.53604201e+00  5.95452763e+00
 -4.33377610e+00 -7.69115306e+00  3.63458239e-01 -1.63244737e+00
  8.37511698e-01 -9.07202426e+00 -3.93286277e+00 -1.02137693e+00
 -7.16536140e+00  1.43364867e+01 -1.26670982e+00  8.35677483e-01
 -5.01789086e+00 -4.57935517e+00  1.06141686e+02  8.35253108e+00
 -3.93075129e+00  1.46076576e+00  2.00402244e+00  1.58759720e+00
  1.20940058e+02 -2.55847395e+00  2.21745232e-01 -7.67703251e+00
  6.00277549e+01 -4.03575703e+00  9.12317952e+01 -1.93043381e+00
  5.28696357e+01 -2.81305814e+00 -8.64320403e+00  1.94068599e+01
  5.80440747e+00  4.04277092e+01  2.60392927e+01 -3.35946373e+00
  3.76653080e+00 -6.58852294e-01 -6.78571399e+00 -2.32654143e+00
 -2.35081978e+00  5.90749316e-01 -5.27689840e+00  7.60469962e+00
  3.14614308e+00 -1.16802365e+00 -4.84505763e+00 -4.70534691e+00
  3.11996719e+00 -3.13815697e+00 -5.33224637e+00  2.04370993e+01
  9.03441847e-02 -5.81751379e+00  2.07834460e+01 -6.86295147e+00
 -5.38647917e+00  1.25140433e+00  9.07155419e+00 -2.45023883e+00
  2.34787092e+00  3.01082519e+01  3.29514699e+00  4.12294730e+01
 -2.13022372e-01 -7.44059425e+00  2.52286505e+00  1.45153021e+01
 -3.22537598e+00 -3.39197881e+00 -3.74510917e+00  7.49935238e+00
  3.82010365e+00 -2.62689605e+00 -6.72906146e-01 -1.09463921e+01
 -6.28931521e+00  5.87811851e+00 -1.20806544e+01  6.29329959e+00
```

```
-3.51002205e+00  7.56646784e-01  4.06813472e+00  4.46699715e+00
-5.26175346e-01 -4.22128079e+00  6.10659232e+00  1.02346160e+00
 4.24078915e-01  3.40915692e+00 -1.16720411e+01  3.48283596e+01
-2.64497418e+00  2.11019436e+01  6.62821352e+01  8.81543461e+00
-2.28066931e+00  1.52231898e+00 -5.42864057e+00  8.09970360e+01
 4.82574041e+00 -5.35609304e-01 -3.73905730e+00  3.85279522e+00
-2.33593741e+00  5.66974209e+00 -8.89605006e+00 -4.39358784e+00
 2.18883676e+01 -2.20536562e+00 -7.86288394e+00  1.09331503e+01
-5.90405232e+00  5.87816883e+00  5.40264080e-01  4.06062325e+01
 8.03205174e-01  1.00666670e+01  6.93429381e+01 -5.42230294e+00
 6.96511984e+01  5.25680616e-01 -2.02960908e+00 -8.45962557e+00
-7.39821310e+00 -6.91805293e-01 -1.75516123e+00  3.90203760e+01
 1.74239683e+01  7.08446746e+01 -3.91963231e+00  5.06117300e+01
-2.67767997e+00  2.17963078e+02  1.65069236e+00 -2.12088171e+00
 8.74080901e+01  4.43972171e+00  6.40597044e-02  8.26842501e+01
-4.39068013e+00 -3.74890491e+00  9.54533612e+01  1.14308743e+00
-6.99195502e+00 -4.65194882e+00  2.74382546e+01 -7.31305401e-01
-2.77518033e-01 -8.95978586e+00  1.89643327e+02 -7.27369365e+00
 3.96470174e+00 -2.76464043e+00  3.09717564e-01 -3.41846562e+00
 9.18001358e+01  5.32540497e+00 -8.37075603e+00  1.83477881e+00
-6.05001779e+00 -5.09794512e+00  6.87397239e+01 -6.65153587e+00
 7.00050277e+00 -6.31809216e+00  5.37509785e-01 -1.38161786e+00
 6.33988935e+00  9.27880042e-01  7.70954043e+00 -2.23471198e+00
 6.75377957e+01  6.22304547e-01 -1.48534770e+00  2.54512702e-02
 9.45130068e+01 -4.93943884e+00 -1.37336123e+00  7.59799199e+01
-6.39518661e+00 -6.11450569e+00  2.90954955e+00 -1.03826571e+01
-3.80040946e+00  2.03740891e+01  1.72922878e+00 -6.04044903e+00
 1.69242190e+01 -1.30234364e+01  4.04157302e+01  1.05451836e+01
-2.31762168e+00 -6.43188290e+00 -9.56555490e+00 -1.81613989e+00
-3.26195410e-01  3.37891335e+00 -1.80968459e+00 -5.77956306e+00
-4.52632999e+00 -5.05664408e+00  1.13925677e+01  3.13385344e+01
 5.31424463e+01  2.55171683e+00  1.05052006e+00  2.09910820e+01
-3.02314868e+00 -7.58517613e+00  4.49953661e+00 -3.76874539e+00
-1.82585853e+00 -1.46004768e-01 -6.24591963e+00 -7.13058035e-02
-1.28831825e+00 -4.46411653e-01  6.34184767e+00 -3.14023154e+00
 6.09955508e+00  9.71616807e-01 -2.33256642e+00 -8.71208396e-01
 3.90523769e+01  3.81918334e+00  1.69457286e+00 -4.14311920e+00
 1.73238013e+01  2.02392529e+01  1.69704119e+01 -7.79170767e+00
-4.31268712e+00  5.90915030e+00 -7.72585743e-01 -3.78526086e+00
 6.52897741e+00  2.29316848e+01 -1.59082918e+00 -5.83537410e+00
 3.65845718e+00 -1.11175070e+00 -1.09611895e+00 -4.38754157e+00
-7.19531032e+00 -1.38181305e+00  2.47306215e+01 -1.00002234e+00
 2.34236818e+01 -7.40776274e+00  2.83304458e+01 -1.17350552e+01
-3.16833354e+00 -6.73593570e+00 -2.56539914e+00 -7.82483796e+00
-7.64144957e-01 -4.75852269e+00  2.40080054e+01  3.73204405e+01
 2.18084674e+00  1.92524387e+01  6.21778439e+00 -8.81054794e+00
-1.79894317e+00 -9.83016404e+00  1.95303628e+00  1.53949638e+02
 5.74471363e+00  1.87395679e+01 -1.48414699e+00  5.84985740e+00
 5.22323815e+00 -1.38945998e+00 -8.44455833e-01 -6.61002171e+00
 3.13413180e+01 -1.72378630e+00 -3.20041009e+00  1.75110174e+01
 3.51912253e+01 -1.41570701e+00 -4.45786153e+00 -2.06286294e+00
-4.69398578e+00  5.09064495e+00 -5.65784254e+00  9.56672209e+01
-7.01071159e+00  4.43498620e+00 -7.36955578e+00  7.44495118e+00
 3.30699149e+00  5.30073140e+00 -2.75284728e+00  3.50336025e+01
-5.10703385e+00  2.62412736e+00 -1.05688742e+00  3.17251152e+00
 2.81762787e+00 -4.22347067e+00 -1.84165660e+00 -1.09618262e+00
-1.03981981e+00  2.07131511e+01  5.53915930e+00 -5.63959613e-02
-8.61385240e+00  1.11735461e+01 -3.12334261e+00 -2.29931406e+00
-1.53078605e+00 -2.15405720e+00  6.38956684e+01  1.09235411e+00
 5.07513515e+00  1.04606779e+01  6.50558303e+00  1.17120735e+01
 5.53920805e+00 -2.95095791e+00 -5.59145826e+00  8.43248939e+01
```

```
 1.97841343e+01 -1.34505961e+00 -1.41904097e+00  2.46519267e+00
-2.38130557e+00  7.57796699e+00 -5.03799324e+00 -2.30683659e+00
-1.45886006e-01  1.84589034e+01  1.56481687e+00 -9.23922393e-01
-4.27748845e+00 -3.73490022e+00  2.92563662e+00  9.95866665e-01
 1.01124956e+01  5.96678239e+01  1.72529828e+00  9.58360033e+00
 5.87956008e+01 -8.91930034e+00 -8.57179528e-01 -9.17066087e-01
-1.09821546e+01 -3.76516734e+00 -3.87394232e+00 -1.74947625e+00
 7.92544015e-01  2.04324193e+01  4.94443738e+00 -6.57775525e-01
-4.37121109e+00  7.72306376e+00  6.66007230e+01  1.91708348e-01
 9.53504815e+00  3.67149660e+00  2.52257664e+00  8.94585488e+00
-5.53042190e+00 -5.93544295e+00  3.62478319e+00 -3.49834803e+00
-2.04142232e+00 -7.20015372e+00 -2.36228029e+00 -5.69899320e+00
-1.48738560e-01 -3.86064432e+00 -2.87751919e+00  4.08217774e+01
-9.53821714e+00 -6.75886081e-01  2.16802574e+01  9.88012001e-02
-6.67570284e-01  2.33016266e+01 -5.16781278e+00  8.80269191e+00
-2.79802155e+00  6.75406586e+00 -3.35123424e+00 -1.26995006e+00
 7.17417576e+01 -1.57574686e+00  1.77186106e+00 -2.46271185e+00
-4.22121828e+00  5.61064286e+01  1.29329734e+01 -1.56733138e-02
 2.35592675e+00  1.38031310e+01 -1.89454459e+00  7.57002543e-01
 5.89904620e+00 -3.67587994e+00 -3.94818612e+00 -4.94679744e-01
-1.63837831e+00  1.87481331e+01 -1.03384315e+00  3.39629748e-01
-6.85337515e+00 -2.20231408e+00  4.09982726e+00  3.70480995e+00
-3.80410733e-01  8.51067693e+00 -5.27496429e+00 -3.11354017e+00
-9.73046261e+00  4.25683689e+00 -8.04842444e+00 -4.72516032e+00
 1.18781864e+02 -5.17239572e+00 -6.68250841e-01 -4.01050049e-01
 4.75486031e+00 -2.07146326e+00  1.75288935e+01 -3.12001525e+00
-1.28980432e-01 -9.02971989e-01  2.80543083e+01 -5.28921174e+00
 1.81354962e-01 -9.64713020e-01  3.91661388e+00  3.58006356e+00
-6.89133800e+00  3.71791790e+01  4.62162465e-01  3.92372741e+00
-4.21102795e+00  9.43853004e+01  1.84891670e+00  7.66501875e+00
 1.75020703e+00 -1.15809193e+00  1.27362044e+00  5.73177406e+01
 1.49951659e+01 -1.58610692e+01 -8.83595928e+00  8.23846867e+01
-5.57247030e+00 -6.21627074e-01  1.77467702e+00 -6.11265176e+00
-1.08920705e+00 -2.64346720e+00 -6.17977869e-01 -1.54093837e+00
-6.01949331e+00  1.54955860e+02 -1.65264728e-03  1.98752832e+00
-1.88080667e+00  1.92254336e+01 -6.37022460e+00  8.38071964e+00
-1.72323597e+00 -4.14803641e+00 -3.11764007e-01 -3.73153391e+00
 1.60974562e+02  4.85607023e-01 -2.99240410e+00 -5.14874248e+00
-1.59968321e+00  6.19177130e+01  2.52370697e+00 -5.59321310e+00
 1.58721077e+00 -1.35148741e+00  9.90750764e+01  3.88708421e+00
 2.67329633e+00  1.44109767e+00  4.54619257e+00  5.10438916e+02
-6.32693667e-01 -7.62427064e+00  2.37903619e+00  3.58548404e+00
 1.02688517e+00 -4.37673636e+00 -4.93119816e+00  1.01745022e+01
 4.67227124e+00  5.03658145e+01  1.68074051e-01 -5.94981289e+00
-6.29855668e+00  7.31708777e+00 -1.09117932e+01  1.50917050e+00
 3.64151031e+00 -2.22267976e+00 -3.26909771e+00  3.71614926e+01
-5.19942128e+00  2.92924388e+00  5.83700511e+01  2.89521307e+00
 1.59826631e-01  9.88393403e+00  4.17966986e+00  1.72288726e+01
-5.90956916e+00 -1.06013671e+00 -1.70874345e+00 -6.69973173e+00
 6.80029812e+00  1.99853193e+01 -8.04012841e+00 -1.24945487e+01
-4.81008235e+00 -4.78475177e-01 -6.64274973e+00  2.13975138e+01
-2.69441438e+00  2.62145870e+01 -3.16550362e+00  9.62506065e-01
 7.97465414e+00 -9.09099401e-01 -5.49624748e+00 -2.12047608e+00
 4.97022042e+00 -3.73070540e+00 -6.49672217e+00  1.90944668e+01
 6.33714709e+00 -1.87525606e+00 -5.95442131e-02 -3.30000106e+00
-6.16427723e+00  6.77337783e+00 -4.28922730e+00  3.27771148e+00
-4.34056282e+00  1.37218922e+00 -2.98271717e+00 -6.61193964e+00
-7.21757129e+00 -3.96307694e-01 -5.51042546e-01 -5.77479295e+00
-6.09343711e+00  3.97866634e-01 -3.73594913e+00 -2.75631376e+00
 2.14431321e+00  4.64785192e+00  8.85396965e+01  6.49070756e+00
-2.89662160e+00 -1.15696643e+01  2.04000468e+01  2.62341831e+00
```

```
 3.82927251e+00 -5.51544727e+00 -2.21487733e+00 -1.00267198e+00
 4.85102363e-01 -7.06677062e+00 -6.16863241e+00  4.73912899e+00
 4.01428502e+01  1.74015278e+00 -7.55051386e+00 -3.14485132e+00
 1.32116799e+01  5.16550707e-01  1.09130248e+00  2.03877339e-01
 2.42657415e+01 -1.81551598e+00 -5.30799250e+00  2.01511145e+01
 4.80354863e-01 -2.68387614e+00 -3.69426458e+00 -3.38744068e+00
-4.04384344e+00  3.64392202e+01  1.30209278e+01 -5.56629000e-01
-1.78843474e+00  1.32402512e+01 -6.03576479e+00  4.53837383e+01
-2.09819636e+00 -2.44546899e+00 -7.55762500e+00 -3.03517375e+00
-1.96957321e+00 -8.35449565e-01 -7.86445164e+00 -2.49343540e+00
 9.51155153e+00 -2.52079630e+00 -2.88861112e+00 -2.69180971e+00
-4.35647273e-01  2.53735371e+01 -1.70995253e+00  1.06182909e+01
 3.37304434e+01 -1.76487091e+00  5.49586682e+01  2.52458301e+01
-3.28104291e+00 -6.32315866e-01 -4.20685746e+00 -2.45032873e+00
 8.43670587e-01  3.33790319e+00  7.03902387e-01  7.48691340e+00
-3.00204577e+00 -1.10069045e-01 -8.69450707e+00 -8.96114778e-01
 1.02902118e+02  3.88581649e-01  2.06084565e+00 -6.79088053e-01
-8.03324609e-01  6.98164953e+01  8.50690997e+01  4.11967337e+01
-2.18309526e-01  5.35190753e-01 -7.87568163e+00 -1.88660345e+00
-8.36579255e+00  1.37013478e+01  3.68194120e+01  2.24770508e+00
 6.87212475e+00 -1.13581690e+00  4.40794065e+00 -5.32265154e-01
 1.78917595e+01  8.26761801e-01 -1.52189964e+00 -2.51340071e+00
-1.73813437e+00  2.69083805e+00  1.05928019e+01  9.35486919e+00
 7.13806228e+00  1.10797967e+01 -3.43434467e+00  1.01120897e+02
-7.39673872e+00 -1.69016536e+00  9.90329755e+00 -3.39957824e+00
 1.15496501e+02 -5.99764603e+00  1.88283885e+01 -3.08617874e+00
 5.53027705e+00 -8.19939647e+00  2.16085378e+01  6.92589101e+01
 5.45195754e+00 -9.15130857e+00 -3.18954155e+00  1.89577026e+01
 1.23376755e+02 -1.14637430e+00  5.80897688e+00  4.62093600e+00
 6.38384959e+01  5.48920804e-01 -1.01021240e+01 -5.54182286e+00
 4.44846212e-01  1.07050700e+01 -2.77086862e+00 -2.87451617e+00
 1.72169920e+01 -3.40617346e+00 -3.52763418e+00  4.70778179e+00
 3.74410710e+01  5.30619825e+00  8.78268445e-01 -2.32579117e+00
 2.99145338e-01 -3.05725667e+00  5.74007319e+00  4.40195041e+01
 3.35174520e+00 -7.21968063e+00  6.44730689e+00  1.19465438e+00
-2.05048982e+00  1.55136392e+01  7.86152422e+00 -2.12260679e+00
 8.90497080e+00  2.84729396e+01  4.46967489e+00 -3.25492265e+00
-1.96165356e+00 -5.25078078e+00  1.98626884e+01 -1.20707909e+00
-1.60705120e+00 -3.28054009e+00 -1.26237287e+01 -1.29378327e+00
 1.22184507e+01 -1.93768590e+00 -3.04959167e+00 -5.57677465e+00
 2.43635669e+01 -1.39843716e+00 -2.45879664e+00  6.90830861e+00
 2.47367685e-01  1.05630979e+01  1.46297986e+01  4.71736432e+01
 4.15501536e+01  1.69154639e+01  1.33569123e+01  2.54354691e+01
 8.93415161e+01  4.15504116e+00  2.34217996e+01  6.13780354e+01
-6.85401420e+00  4.87373085e+01  8.14421109e+00  1.06136890e+00
 4.47966848e+01  1.75226565e+01 -4.73034493e+00  1.26393652e+00
-2.93747641e+00  4.50415159e+01  3.48035015e+01 -1.67843101e+00
 1.61603334e+00  3.98831765e+00 -2.36966161e+00 -6.73802408e+00
-1.36553710e+01 -1.00559354e+01  2.98946161e+01 -4.55753407e+00
-6.20825718e+00 -4.76277781e+00  5.63718843e-01 -9.43908575e-01
 4.63435332e+00 -7.29773209e+00 -3.48187258e+00  3.94877742e+01
 5.91859262e+00 -3.13362629e+00  7.57009255e+00  1.49660879e+00
-3.27318488e-01  1.00790149e+01  1.00161133e+01 -5.97538001e+00
 7.24359458e+00  1.24938529e+00  2.63846324e+00  5.06729727e+00
-2.47081613e+00 -2.02080176e+00 -1.39042970e+00  4.97502108e+00
-7.40832240e+00 -1.96474608e+00 -4.48354872e+00  3.05828324e+01
-2.12067439e+00 -1.75047739e-01 -3.10261213e+00 -1.14072344e+01
 4.53914283e+01 -6.03957204e+00 -2.82499363e+00 -2.49318875e+00
 9.51395639e-01  1.02408175e+01  1.18577674e+01 -2.53887832e+00
 8.86317420e+01  8.49059781e-01 -2.21289172e+00  1.36778897e-01
 3.79743596e+00  9.51626488e+00 -9.32845012e-01 -7.69497710e+00
```

```
 -4.31297437e+00 -1.13140051e+01  2.33547269e+01  5.72271628e+01
 -3.99726112e+00  5.51302633e-01  3.29486405e+01  1.63008754e+01
 -4.81735170e+00  5.39533413e+01 -9.66091626e+00 -6.46468585e-01
 -1.10649308e+00 -9.98509908e-02 -5.92498701e-01  1.22689412e+01
 -7.58572647e+00 -7.78328033e+00 -6.34476198e+00 -1.35436993e+00
 -4.27755888e+00  4.63415735e+01 -3.72077440e+00  2.79974039e+00
  2.30019692e+00 -3.26694853e+00 -1.74948323e+00  1.36024857e+00]
0.9903576962542506
```

Градиентный спуск

В предыдущем разделе мы использовали существующие реализации методов обучения линейной регрессии с регуляризацией и без. Тем не менее, подобные реализации, как правило, имеются лишь для ограниченного набора стандартных методов. В частности, при выходе функционала качества за пределы стандартного множества необходимо самостоятельно реализовывать составляющие процесса решения оптимизационной задачи. Именно этому и посвящен данный раздел задания.

Пусть необходимо минимизировать следующий функционал (Mean Square Percentage Error — модифицированный RMSPE (https://www.kaggle.com/c/rossmann-store-sales/details/evaluation)):

$$MSPE(\{x_i, y_i\}_{i=1}^l,\ w) = \frac{1}{l}\sum_{i=1}^{l}\left(\frac{y_i - \langle w, x_i\rangle}{y_i}\right)^2,$$

где $\{x_i, y_i\}_{i=1}^l$ — обучающая выборка, w — вектор весов линейной модели. Будем также рассматривать функционал $MSPE$ с L2-регуляризацией:

$$MSPE(\{x_i, y_i\}_{i=1}^l,\ w) = \frac{1}{l}\sum_{i=1}^{l}\left(\frac{y_i - \langle w, x_i\rangle}{y_i}\right)^2 + ||w||_2^2.$$

19. (0 баллов) Добавьте к объектам обеих выборок из п. 16 единичный признак.

In [75]:

```
# Your code here
df['sign']=1
dfd['sign']=1
```

20. (1 балл) Реализуйте функции, которые вычисляют:

- прогнозы линейной модели;
- функционал $MSPE$ и его градиент;
- регуляризованный $MSPE$ и его градиент.

In [76]:

```
# возвращает вектор прогнозов линейной модели с вектором весов w для выборки X
def make_pred(X, w):
    return np.dot(X,w)
    pass
```

In [77]:

```
# возвращает значение функционала MSPE для выборки (X, y) и вектора весов w
def get_func(w, X, y):
    MSPE = (y - make_pred(X, w)) / y
    return np.dot(MSPE, MSPE) / y.shape[0]
    pass
```

In [78]:

```
# возвращает градиент функционала MSPE для выборки (X, y) и вектора весов w
def get_grad(w, X, y):
    M = (y - make_pred(X, w)) / y
    S= np.dot(X.T, M / y)
    return -2 * S / y.shape[0]
    pass
```

In [79]:

```
# возвращает значение регуляризованного функционала MSPE для выборки (X, y) и вектор
а весов w
def get_reg_func(w, X, y):
    L = (y - make_pred(X, w)) / y
    return np.dot(L, L) / y.shape[0] + np.dot(w, w)
    pass
```

In [80]:

```
# возвращает градиент регуляризованного функционала MSPE для выборки (X, y) и вектор
а весов w
def get_reg_grad(w, X, y):
    return get_grad(w, X, y) + 2 * w
    pass
```

21. (1 балл) Реализуйте метод градиентного спуска для описанных функционалов ($MSPE$ и его регуляризованный вариант). Функция должна принимать следующие параметры:

- X — матрица "объект-признак";
- y — вектор целевой переменной;
- w0 — начальное значение вектора весов;
- step_size — значение темпа обучения;
- max_iter — максимальное число итераций;
- eps — значение, используемое в критерии останова;
- is_reg — бинарный параметр, принимает значение True в случае наличия регуляризации функционала, False — в противном случае.

Процесс должен быть остановлен, если выполнено хотя бы одно из следующих условий:

- было выполнено заданное количество итераций max_iter;
- евклидова норма разности векторов w на соседних итерациях стала меньше, чем eps.

Функция должна возвращать полученный в результате оптимизации вектор w и список значений функционала на каждой итерации.

In [81]:

```
def grad_descent(X, y, w0, step_size, max_iter, eps, is_reg):
    # Your code here
    arr = []
    for i in range(0, max_iter, 1):
        if np.linalg.norm(w0) > eps:
            if is_reg:
                w0 = w0 - step_size * get_reg_grad(w0, X, y)
                arr.append(get_reg_func(w0, X, y))
            else:
                w0 = w0 - step_size * get_grad(w0, X, y)
                arr.append(get_func(w_new, X, y))
    return w0, arr
```

Обучите линейную регрессию с функционалом $MSPE$ на обучающей выборке при помощи метода градиентного спуска и изобразите кривые зависимости значения функционала от номера итерации для различных:

- значений размера шага из набора [0.001, 1, 10];
- способов начальной инициализации вектора весов (нули, случайные веса).

Проанализируйте полученные результаты — влияют ли данные параметры на скорость сходимости и итоговое качество? Если да, то как?

In [82]:

```
# Your code here
df_train2 = df_train[:5768]
dfd_train2 = dfd_train[:5768]
w1, a = grad_descent(df_train2, dfd_train2, np.zeros(df_train2.shape[1]), 1, 1, 0.00
1, False)
print(w1, a)
```

```
[0. 0. 0. ... 0. 0. 0.] []
```

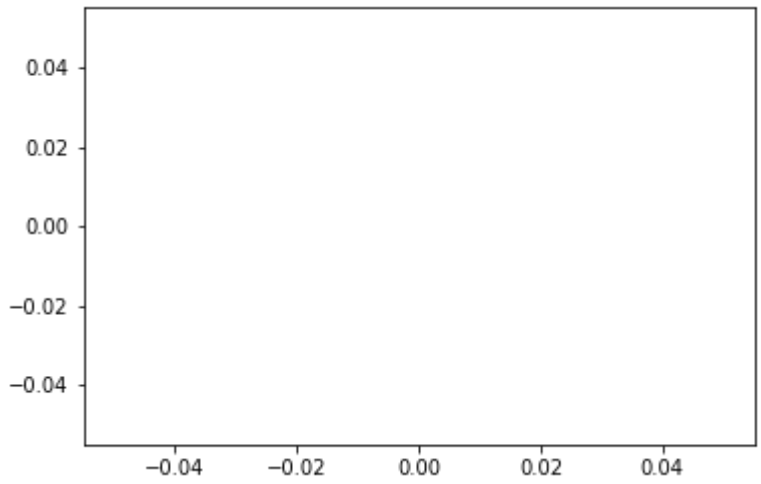
22. (0.5 балла) Обучите линейную регрессию с функционалом MSPE и его регуляризованным вариантом на обучающей выборке при помощи метода градиентного спуска и изобразите кривые зависимости значения функционала от номера итерации. Исследуйте зависимость скорости сходимости от наличия регуляризации. Обоснуйте, почему так происходит.

In [83]:

```
# Your code here
import matplotlib.pyplot as plt
plt.figure()
w2, L = grad_descent(df_train2, dfd_train2, np.zeros(df_train2.shape[1]), 1, 1, 0.1,
False)
plt.plot(range(len(L)), L, 'rs')
```

Out[83]:

```
[<matplotlib.lines.Line2D at 0x207156b7278>]
```



Метод градиентного спуска может быть весьма трудозатратен в случае большого размера обучающей выборки. Поэтому часто используют метод стохастического градиентного спуска, где на каждой итерации выбирается случайный объект из обучающей выборки и обновление весов происходит только по этому объекту.

23. (1 доп. балл) Реализуйте метод стохастического градиентного спуска (SGD) для описанных функционалов ($MSPE$ и его регуляризованный вариант). Функция должна иметь параметры и возвращаемое значение, аналогичные оным функции grad_descent из п.21. Кроме того, должен использоваться аналогичный критерий останова.

In [84]:

```
def sgd(X, y, w0, step_size, max_iter, eps, is_reg):
    # Your code here
    X0 = X.iloc[0].values.reshape(1, -1)
    y0 = np.asarray(y.iloc[0]).reshape(1, -1)[0,:]
    arr2 = []
    for i in range(max_iter):
        if np.linalg.norm(w0) > eps:
            ii = np.random.randint(0, high=X.shape[0], size=1)[0]
            X1 = X.iloc[ii].values.reshape(1, -1)
            y1 = np.asarray(y.iloc[ii]).reshape(1, -1)[0,:]
            if is_reg:
                w0 = w0 - step_size * get_reg_grad(w0, X1, y1)
                arr2.append(get_reg_func(w0, X1, y1))
            else:
                w0 = w0 - step_size * get_grad(w0, X1, y1)
                arr2.append(get_func(w0, X1, y1))
    return w0, arr2
```

Обучите линейную регрессию с функционалом $MSPE$ и его регуляризованным вариантом на обучающей выборке при помощи метода стохастического градиентного спуска, подобрав при этом размер шага, при котором метод будет сходиться. Нарисуйте график сходимости. Выведите значения $MSPE, MSE, R^2$ на контрольной выборке.

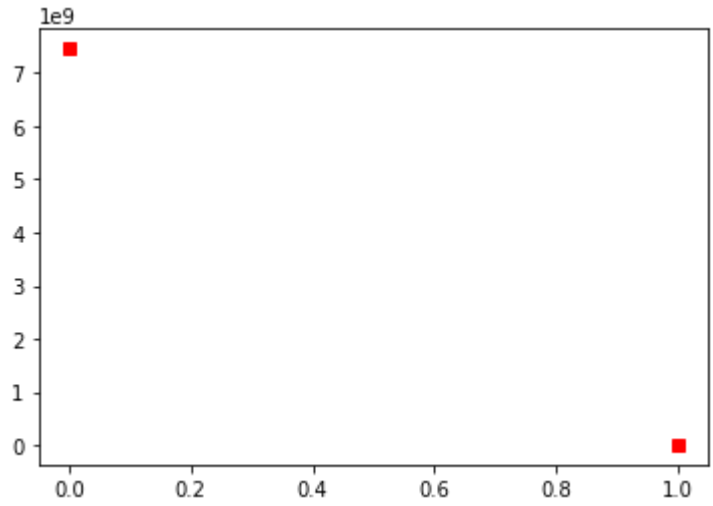
In [85]:

```
# Your code here
ww = np.random.rand(df_train.shape[1])
ff = plt.figure()
aa = ff.add_subplot(1, 1, 1)
ww, L2r = sgd(df_train, dfd_train, ww, 0.0003, 20000, 0.0001, True)
plt.plot(range(len(L2r)), L2r, 'rs')
ww, L2 = sgd(df_train, dfd_train, ww, 0.008, 20000, 0.0001, False)
plt.plot(range(len(L2)), L2, 'rs')
preds = make_pred(df_test, ww)
preds
```

```
c:\users\hp\appdata\local\programs\python\python35\lib\site-packages\ipy
kernel_launcher.py:3: RuntimeWarning: divide by zero encountered in true
_divide
  This is separate from the ipykernel package so we can avoid doing impo
rts until
```

Out[85]:

```
array([nan, nan, nan, ..., nan, nan, nan])
```



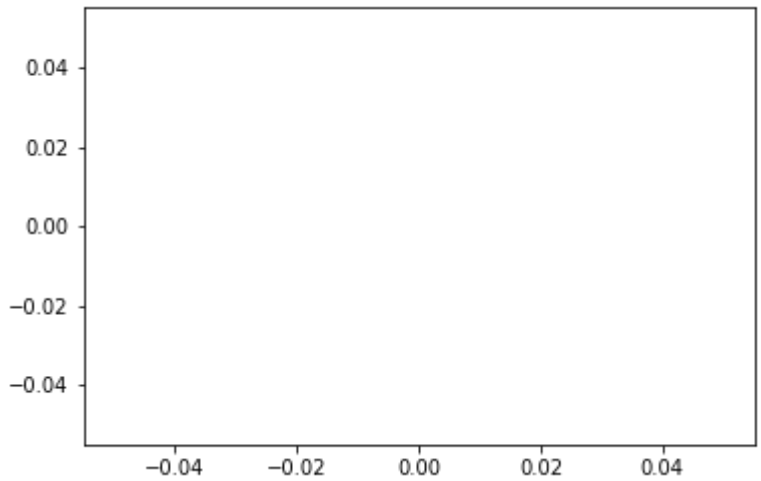
24. (0.5 доп. балла) Аналогично п.22 исследуйте зависимость скорости сходимости метода SGD от наличия регуляризации. Обоснуйте, почему так происходит.

In [86]:

```
# Your code here
fff = plt.figure()
ww, L3 = sgd(df_train, dfd_train, ww, 0.0003, 20000, 0.0001, False)
plt.plot(range(len(L3)), L3, 'rs')
w_opt, Loss = sgd(df_train, dfd_train, ww, 0.008, 20000, 0.0001, True)
plt.plot(range(len(L3)), L3, 'rs')
```

Out[86]:

```
[<matplotlib.lines.Line2D at 0x207156e63c8>]
```



25. (0.5 балла) Обучите стандартную линейную регрессию с функционалом качества MSE на обучающей выборке и выведите значение MSPE полученного решения на контрольной выборке. Как оно соотносится с аналогичным результатом для решения, полученного в п.22? Почему?

In [88]:

```
# Your code here
from sklearn.linear_model import LinearRegression
linreg = LinearRegression()
linreg.fit(df_train, dfd_train)
c = linreg.coef_
p = linreg.predict(df_test)
print(c)
print(p)
```

```
[ 3.77454339e-10  4.43931907e-03 -1.14896391e-03 ... -2.69452889e+07
 -2.69452865e+07 -2.69452852e+07]
[ 22.69580966  86.87362266  40.86627477 ... 269.50595593   4.00586587
  37.37045759]
```

Здесь вы можете поделиться своими мыслями по поводу этого задания.

In [ ]:

```

```

А здесь — вставить вашу любимую картинку.

In [ ]:

```

```