Отборочное задание Евраз. Исп.Бочаров А.М. скайп bam271074

Сталь обрабатывают в металлическом ковше вместимостью около 110 тонн. Чтобы ковш выдерживал высокие температуры, изнутри его облицовывают огнеупорным кирпичом. Расплавленную сталь заливают в ковш и подогревают до нужной температуры графитовыми электродами. Они установлены в крышке ковша. Из сплава выводится сера (десульфурация), добавлением примесей корректируется химический состав и отбираются пробы. Сталь легируют — изменяют её состав — подавая куски сплава из бункера для сыпучих материалов или проволоку через специальный трайб-аппарат (англ. tribe, «масса»). Перед тем как первый раз ввести легирующие добавки, измеряют температуру стали и производят её химический анализ. Потом температуру на несколько минут повышают, добавляют легирующие материалы и продувают сплав инертным газом. Затем его перемешивают и снова проводят измерения. Такой цикл повторяется до достижения целевого химического состава и оптимальной температуры плавки. Тогда расплавленная сталь отправляется на доводку металла или поступает в машину непрерывной разливки стали МНЛЗ. Оттуда готовый продукт выходит в виде заготовок-рельс, каждая из которых пилится на 4 отрезка - крата.

Задача Изучите данные физико-химического процесса легирования сталей. Создайте алгоритм определения химического состава шлака по исходным данным. Файл «Исходные данные.csv» (данные по плавкам о количестве добавок, расходе э/э, химическом анализе стали и хим анализ шлака, забираемых в начале плавки и в конце). Данные требуется очистить, в трети выборки показатели хим анализа отсутствуют, их нужно занулить. Зафиксировать самую популярную по плавкам марку стали и для нее спрогнозировать целевой состав шлака.

In [33]:

```
import pandas as pd  #for work with dataframes
from datetime import datetime
import matplotlib.pyplot as plt  # lib for plots
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error  #for metric of our model
from sklearn.model_selection import GridSearchCV
RS=42   #we will fix random state

import warnings
warnings.filterwarnings('ignore')
```

In [34]:

```
#let s build dataframe with our dataset
df_train = pd.read_csv('d:\\2021\\hackatones\\EVRAZ\\Исходные данные.csv',sep=';',
                       index_col=0,decimal=',')
df_train.tail()
```

Out[34]:

	nplv	DT	МАРКА	ПРОФИЛЬ	t вып-обр	t обработка	t под током	t продувка	
7036	r68386398<;	2071-04-21 19:50:00	Э90ХАФ	P65	41.0	42.333333	23.200000	42.166667	3
7037	r68386398<<	2071-04-21 20:45:00	Э90ХАФ	P65	36.0	46.533333	16.183333	46.383333	
7038	r68386398<=	2071-04-21 21:34:00	Э90ХАФ	P65	42.0	47.566667	23.016667	47.100000	2
7039	r68386398<>	2071-04-21 22:25:00	Э90ХАФ	P65	45.0	46.033333	17.533333	45.683333	3
7040	r68386398<?	2071-04-21 23:20:00	Э90ХАФ	P65	48.0	52.033333	21.700000	50.233333	3

5 rows × 84 columns

In [3]:

```
df_train.columns
```

Out[3]:

```
Index(['nplv', 'DT', 'МАРКА', 'ПРОФИЛЬ', 't вып-обр', 't обработка',
       't под током', 't продувка', 'ПСН гр.', 'чист расход C',
       'чист расход Cr', 'чист расход Mn', 'чист расход Si', 'чист расход
V',
       'температура первая', 'температура последняя', 'Ar (интенс.)',
       'N2 (интенс.)', 'эл. энергия (интенс.)', 'произв жидкая сталь',
       'произв количество обработок', 'произв  количество плавок',
       'произв количество плавок (цел)', 'расход газ Ar', 'расход газ  N
2',
       'расход C пров.', 'сыпуч известь РП', 'сыпуч кварцит',
       'сыпуч кокс пыль УСТК', 'сыпуч  кокс. мелочь (сух.)',
       'сыпуч  кокс. мелочь КМ1', 'сыпуч  шпат плав.', 'ферспл CaC2',
       'ферспл FeMo', 'ферспл FeSi-75', 'ферспл FeV азот.', 'ферспл  FeV-8
0',
       'ферспл  Mn5Si65Al0.5', 'ферспл  Ni H1 пласт.', 'ферспл SiMn18',
       'ферспл  ферванит', 'ферспл  фх850А', 'эл. энергия',
       'химсталь первый Al_1', 'химсталь первый C_1', 'химсталь первый Cr_
1',
       'химсталь первый Cu_1', 'химсталь первый Mn_1', 'химсталь первый Mo
_1',
       'химсталь первый N_1', 'химсталь первый Ni_1', 'химсталь первый P_
1',
       'химсталь первый S_1', 'химсталь первый Si_1', 'химсталь первый Ti_
1',
       'химсталь первый V_1', 'химсталь последний Al', 'химсталь последний
C',
       'химсталь последний Ca', 'химсталь последний Cr',
       'химсталь последний Cu', 'химсталь последний Mn',
       'химсталь последний Mo', 'химсталь последний N',
       'химсталь последний Ni', 'химсталь последний P', 'химсталь последни
й S',
       'химсталь последний Si', 'химсталь последний Ti',
       'химсталь последний V', 'химшлак первый Al2O3_1',
       'химшлак первый CaO_1', 'химшлак первый FeO_1', 'химшлак первый MgO
_1',
       'химшлак первый MnO_1', 'химшлак первый R_1', 'химшлак первый SiO2_
1',
       'химшлак последний Al2O3', 'химшлак последний CaO',
       'химшлак последний FeO', 'химшлак последний MgO',
       'химшлак последний MnO', 'химшлак последний R',
       'химшлак последний SiO2'],
      dtype='object')
```

In [35]:

```
df_train.describe(include='object').T
```

Out[35]:

	count	unique	top	freq
nplv	7041	7041	r6838637:9?	1
DT	7041	7041	2071-02-11 00:46:00	1
МАРКА	7041	18	Э76ХФ	4953
ПРОФИЛЬ	7041	18	Р65	6427

In [5]:

```
#let s see how many NaNs
df_train.isnull().sum()
```

Out[5]:

```
nplv                                0
DT                                  0
МАРКА                               0
ПРОФИЛЬ                             0
t вып-обр                         597
t обработка                       601
t под током                       601
t продувка                        597
ПСН гр.                          1792
чист расход C                     606
чист расход Cr                   1158
чист расход Mn                    610
чист расход Si                    610
чист расход V                     863
температура первая                605
температура последняя             605
Ar (интенс.)                      615
N2 (интенс.)                     5652
эл. энергия (интенс.)             601
произв жидкая сталь               597
произв количество обработок       597
произв  количество плавок         597
произв количество плавок (цел)    597
расход газ Ar                     615
расход газ  N2                   5652
расход C пров.                   6317
сыпуч известь РП                 2271
сыпуч кварцит                    7028
сыпуч кокс пыль УСТК              619
сыпуч  кокс. мелочь (сух.)       7034
                                  ...
химсталь первый Ti_1              597
химсталь первый V_1               597
химсталь последний Al             597
химсталь последний C              597
химсталь последний Ca             823
химсталь последний Cr             597
химсталь последний Cu             597
химсталь последний Mn             597
химсталь последний Mo             597
химсталь последний N              599
химсталь последний Ni             597
химсталь последний P              597
химсталь последний S              597
химсталь последний Si             597
химсталь последний Ti             597
химсталь последний V              597
химшлак первый Al2O3_1           2299
химшлак первый CaO_1              597
химшлак первый FeO_1              597
химшлак первый MgO_1              598
химшлак первый MnO_1              597
химшлак первый R_1                597
химшлак первый SiO2_1            2299
химшлак последний Al2O3          2299
химшлак последний CaO             597
химшлак последний FeO             597
химшлак последний MgO             598
химшлак последний MnO             597
```

```
химшлак последний R                597
химшлак последний SiO2            2299
Length: 84, dtype: int64
```

Провести EDA (исследовательский анализ данных) целевого химического состава шлака и предсказать: • химшлак последний Al2O3 • химшлак последний CaO • химшлак последний R • химшлак последний SiO2

In [36]:

```
df_train.shape
```

Out[36]:

```
(7041, 84)
```

In [38]:

```
# самая популярная марка Э76ХФ     встречается в датасете 4953 раз
df_best_marka = df_train.loc[df_train['МАРКА'] == 'Э76ХФ']
df_best_marka.shape
```

Out[38]:

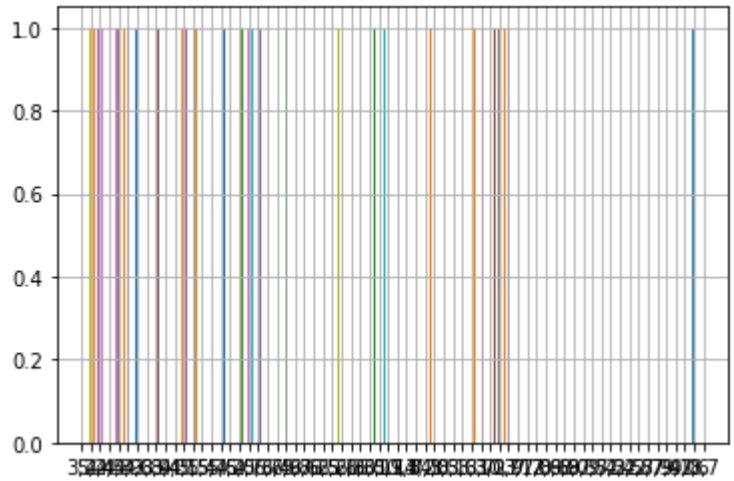
```
(4953, 84)
```

In [16]:

```
df_best_marka['химшлак последний Al2O3'].hist()
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xf38dd30>
```

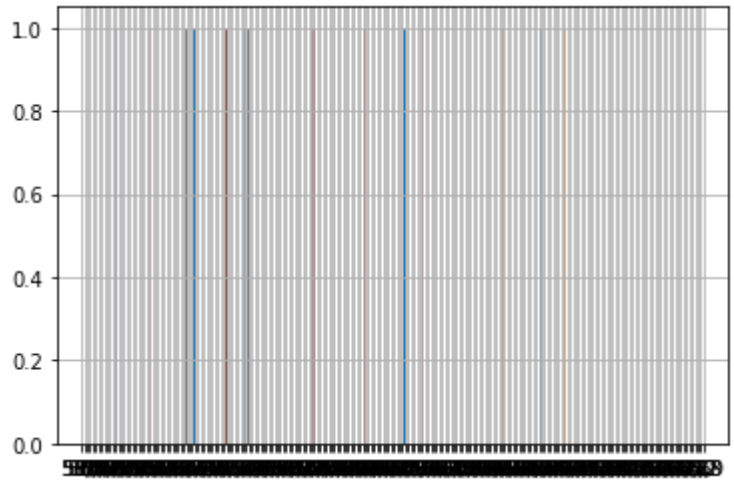


In [17]:

```
# let s see химшлак последний CaO
df_best_marka['химшлак последний CaO'].hist()
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x22d907f0>
```

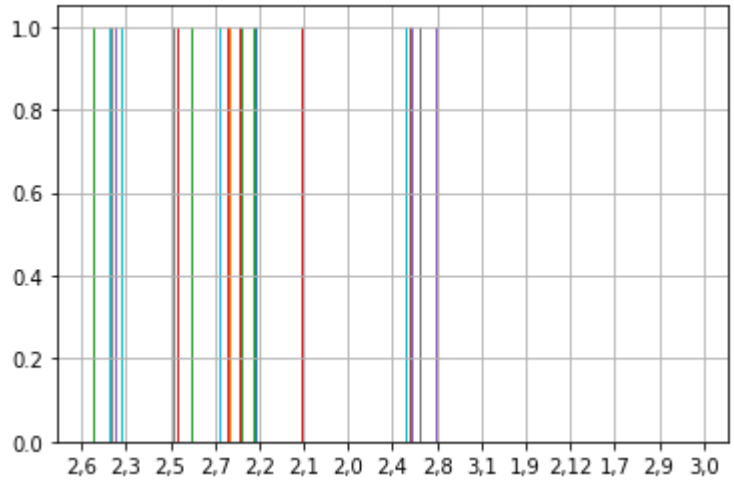


In [18]:

```
# let s see химшлак последний R
df_best_marka['химшлак последний R'].hist()
```

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x431f24e0>
```

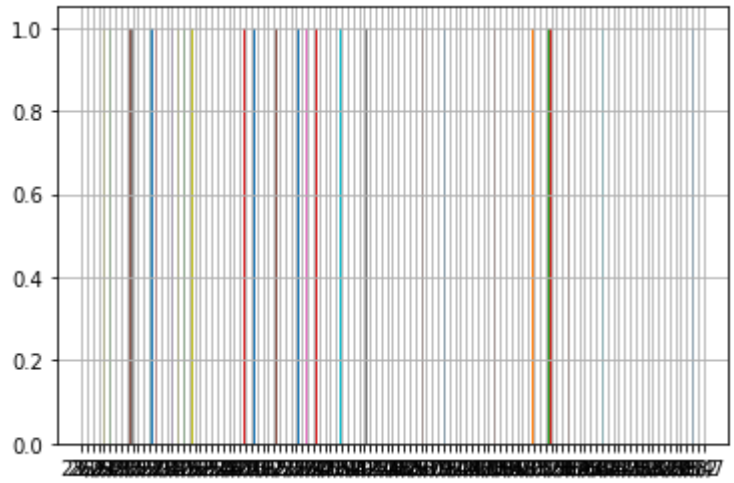


In [19]:

```
# let s see химшлак последний SiO2
df_best_marka['химшлак последний SiO2'].hist()
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x66dd90b8>
```



In [39]:

```
Target_columns=['химшлак последний Al2O3', 'химшлак последний CaO',
       'химшлак последний R','химшлак последний SiO2']
```

In [40]:

```
datetime_object_now = datetime.strptime('2071-02-11 00:46:00', '%Y-%m-%d %H:%M:%S')
df_best_marka['DT'] = \
list(map(lambda x: (datetime_object_now - datetime.strptime(x.replace('-', ' '),
                                                            '%Y %m %d %H:%M:%S')).days,
         df_best_marka['DT']))
df_best_marka.head()
```

Out[40]:

	nplv	DT	МАРКА	ПРОФИЛЬ	t вып-обр	t обработка	t под током	t продувка	ПСН гр.	ра
0	r6837?3<87?	235	Э76ХФ	Р65	29.0	45.366667	24.400000	41.033333	NaN	0.4
1	r683863<886	234	Э76ХФ	Р65	26.0	44.066667	13.866667	44.066667	NaN	0.3
2	r683863<887	234	Э76ХФ	Р65	24.0	43.350000	17.950000	43.350000	NaN	0.3
3	r683863<888	234	Э76ХФ	Р65	17.0	46.183333	19.816667	46.183333	NaN	0.3
4	r683863<889	234	Э76ХФ	Р65	20.0	48.500000	17.033333	48.500000	NaN	0.3

5 rows × 84 columns

In [41]:

```
df_best_marka.ffill(inplace=True)# заполнение соседними значениями
df_best_marka = df_best_marka.fillna(value=0.0) # заполнение Нанов
```

In [42]:

```
#let s see how many NaNs
df_best_marka.isnull().sum()
```

Out[42]:

```
nplv                                0
DT                                  0
МАРКА                               0
ПРОФИЛЬ                             0
t вып-обр                           0
t обработка                         0
t под током                         0
t продувка                          0
ПСН гр.                             0
чист расход C                       0
чист расход Cr                      0
чист расход Mn                      0
чист расход Si                      0
чист расход V                       0
температура первая                  0
температура последняя               0
Ar (интенс.)                        0
N2 (интенс.)                        0
эл. энергия (интенс.)               0
произв жидкая сталь                 0
произв количество обработок         0
произв  количество плавок           0
произв количество плавок (цел)      0
расход газ Ar                       0
расход газ  N2                      0
расход C пров.                      0
сыпуч известь РП                    0
сыпуч кварцит                       0
сыпуч кокс пыль УСТК                0
сыпуч  кокс. мелочь (сух.)          0
                                   ..
химсталь первый Ti_1                0
химсталь первый V_1                 0
химсталь последний Al               0
химсталь последний C                0
химсталь последний Ca               0
химсталь последний Cr               0
химсталь последний Cu               0
химсталь последний Mn               0
химсталь последний Mo               0
химсталь последний N                0
химсталь последний Ni               0
химсталь последний P                0
химсталь последний S                0
химсталь последний Si               0
химсталь последний Ti               0
химсталь последний V                0
химшлак первый Al2O3_1              0
химшлак первый CaO_1                0
химшлак первый FeO_1                0
химшлак первый MgO_1                0
химшлак первый MnO_1                0
химшлак первый R_1                  0
химшлак первый SiO2_1               0
химшлак последний Al2O3             0
химшлак последний CaO               0
химшлак последний FeO               0
химшлак последний MgO               0
химшлак последний MnO               0
```

```
химшлак последний R              0
химшлак последний SiO2           0
Length: 84, dtype: int64
```

In [43]:

```
#  let s make X dataframe for training model
X_columns=[ 'DT', 't вып-обр', 't обработка',
       't под током', 't продувка', 'ПСН гр.', 'чист расход C',
       'чист расход Cr', 'чист расход Mn', 'чист расход Si', 'чист расход V',
       'температура первая', 'температура последняя', 'Ar (интенс.)',
       'N2 (интенс.)', 'эл. энергия (интенс.)', 'произв жидкая сталь',
       'произв количество обработок', 'произв  количество плавок',
       'произв количество плавок (цел)', 'расход газ Ar', 'расход газ  N2',
       'расход C пров.', 'сыпуч известь РП', 'сыпуч кварцит',
       'сыпуч кокс пыль УСТК', 'сыпуч  кокс. мелочь (сух.)',
       'сыпуч  кокс. мелочь КМ1', 'сыпуч  шпат плав.', 'ферспл CaC2',
       'ферспл FeMo', 'ферспл FeSi-75', 'ферспл FeV азот.', 'ферспл  FeV-80',
       'ферспл  Mn5Si65Al0.5', 'ферспл  Ni H1 пласт.', 'ферспл SiMn18',
       'ферспл  ферванит', 'ферспл  фх850А', 'эл. энергия',
       'химсталь первый Al_1', 'химсталь первый C_1', 'химсталь первый Cr_1',
       'химсталь первый Cu_1', 'химсталь первый Mn_1', 'химсталь первый Mo_1',
       'химсталь первый N_1', 'химсталь первый Ni_1', 'химсталь первый P_1',
       'химсталь первый S_1', 'химсталь первый Si_1', 'химсталь первый Ti_1',
       'химсталь первый V_1', 'химсталь последний Al', 'химсталь последний C',
       'химсталь последний Ca', 'химсталь последний Cr',
       'химсталь последний Cu', 'химсталь последний Mn',
       'химсталь последний Mo', 'химсталь последний N',
       'химсталь последний Ni', 'химсталь последний P', 'химсталь последний S',
       'химсталь последний Si', 'химсталь последний Ti',
       'химсталь последний V', 'химшлак первый Al2O3_1',
       'химшлак первый CaO_1', 'химшлак первый FeO_1', 'химшлак первый MgO_1',
       'химшлак первый MnO_1', 'химшлак первый R_1', 'химшлак первый SiO2_1',
       'химшлак последний FeO', 'химшлак последний MgO',
       'химшлак последний MnO']
```

In [44]:

```
X_train, X_test, Y_train, Y_test = train_test_split(
    df_best_marka[X_columns],
    df_best_marka[Target_columns],
    test_size = 0.3,
    random_state = RS,
    shuffle=True)
```

In [45]:

```
net = MLPRegressor(hidden_layer_sizes=(10, 12, 4),
                   verbose=True,
                   early_stopping=True,
                   shuffle=False,
                   random_state=RS
)
```

In [46]:

```
net.fit(X_train,Y_train)
```

```
Iteration 1, loss = 50027.00354732
Validation score: -1865.567622
Iteration 2, loss = 7266.41982721
Validation score: -200.754995
Iteration 3, loss = 939.15758428
Validation score: -67.324499
Iteration 4, loss = 588.64867938
Validation score: -60.762390
Iteration 5, loss = 550.06645150
Validation score: -58.634277
Iteration 6, loss = 535.12108917
Validation score: -57.524775
Iteration 7, loss = 525.35638942
Validation score: -56.442226
Iteration 8, loss = 516.69904672
Validation score: -55.203828
Iteration 9, loss = 505.54837620
Validation score: -53.125211
Iteration 10, loss = 492.75891425
Validation score: -52.356678
Iteration 11, loss = 487.37209563
Validation score: -51.804624
Iteration 12, loss = 483.36835863
Validation score: -51.517969
Iteration 13, loss = 480.82429144
Validation score: -51.348614
Iteration 14, loss = 479.22605497
Validation score: -51.239959
Iteration 15, loss = 478.24685580
Validation score: -51.156874
Iteration 16, loss = 477.38263020
Validation score: -51.058906
Iteration 17, loss = 476.58936028
Validation score: -51.001589
Iteration 18, loss = 475.83908742
Validation score: -50.952900
Iteration 19, loss = 475.10470822
Validation score: -50.911144
Iteration 20, loss = 473.28071859
Validation score: -50.692680
Iteration 21, loss = 470.42689832
Validation score: -50.028863
Iteration 22, loss = 463.37228877
Validation score: -49.301263
Iteration 23, loss = 457.95194230
Validation score: -48.780108
Iteration 24, loss = 452.62558634
Validation score: -48.228303
Iteration 25, loss = 445.67156553
Validation score: -47.439140
Iteration 26, loss = 438.70150072
Validation score: -46.452341
Iteration 27, loss = 429.64970327
Validation score: -45.412518
Iteration 28, loss = 421.41763118
Validation score: -44.740220
Iteration 29, loss = 415.28646978
Validation score: -44.144675
Iteration 30, loss = 409.44020196
Validation score: -43.686861
Iteration 31, loss = 406.05947362
```

```
Validation score: -43.392151
Iteration 32, loss = 403.85346244
Validation score: -43.158333
Iteration 33, loss = 401.99970043
Validation score: -42.943296
Iteration 34, loss = 400.28068364
Validation score: -42.723334
Iteration 35, loss = 398.62975880
Validation score: -42.532153
Iteration 36, loss = 397.17721891
Validation score: -42.362558
Iteration 37, loss = 395.73783013
Validation score: -42.211077
Iteration 38, loss = 394.32719861
Validation score: -42.056448
Iteration 39, loss = 392.91859302
Validation score: -41.905252
Iteration 40, loss = 391.50213045
Validation score: -41.753963
Iteration 41, loss = 390.07231424
Validation score: -41.599578
Iteration 42, loss = 388.61486508
Validation score: -41.440579
Iteration 43, loss = 387.10524424
Validation score: -41.278907
Iteration 44, loss = 385.48310066
Validation score: -41.111681
Iteration 45, loss = 383.72132972
Validation score: -40.941427
Iteration 46, loss = 381.95702408
Validation score: -40.767554
Iteration 47, loss = 380.16677099
Validation score: -40.585158
Iteration 48, loss = 378.34745795
Validation score: -40.396090
Iteration 49, loss = 376.51417188
Validation score: -40.200858
Iteration 50, loss = 374.62472358
Validation score: -39.998811
Iteration 51, loss = 372.72771664
Validation score: -39.791335
Iteration 52, loss = 370.78523485
Validation score: -39.580753
Iteration 53, loss = 368.82523799
Validation score: -39.363805
Iteration 54, loss = 366.80247578
Validation score: -39.145387
Iteration 55, loss = 364.73823942
Validation score: -38.917600
Iteration 56, loss = 362.59842935
Validation score: -38.684393
Iteration 57, loss = 360.41602394
Validation score: -38.444741
Iteration 58, loss = 358.16371641
Validation score: -38.197921
Iteration 59, loss = 355.85706190
Validation score: -37.943562
Iteration 60, loss = 353.46884657
Validation score: -37.682668
Iteration 61, loss = 351.01936549
Validation score: -37.413707
```

```
Iteration 62, loss = 348.49750857
Validation score: -37.136678
Iteration 63, loss = 345.89997282
Validation score: -36.851262
Iteration 64, loss = 343.21452604
Validation score: -36.556772
Iteration 65, loss = 340.45042174
Validation score: -36.253049
Iteration 66, loss = 337.59855038
Validation score: -35.939712
Iteration 67, loss = 334.65584694
Validation score: -35.616555
Iteration 68, loss = 331.62010927
Validation score: -35.282896
Iteration 69, loss = 328.48547651
Validation score: -34.938349
Iteration 70, loss = 325.24918973
Validation score: -34.581903
Iteration 71, loss = 321.91035680
Validation score: -34.211237
Iteration 72, loss = 318.43726926
Validation score: -33.828258
Iteration 73, loss = 314.85857604
Validation score: -33.431708
Iteration 74, loss = 311.13846729
Validation score: -33.022702
Iteration 75, loss = 307.30918774
Validation score: -32.596946
Iteration 76, loss = 303.32068373
Validation score: -32.157099
Iteration 77, loss = 299.20116332
Validation score: -31.700640
Iteration 78, loss = 294.92617872
Validation score: -31.227799
Iteration 79, loss = 290.51243040
Validation score: -30.734599
Iteration 80, loss = 285.89515760
Validation score: -30.223867
Iteration 81, loss = 281.12394160
Validation score: -29.692537
Iteration 82, loss = 276.16144789
Validation score: -29.141273
Iteration 83, loss = 271.00884017
Validation score: -28.568390
Iteration 84, loss = 265.65408379
Validation score: -27.972939
Iteration 85, loss = 260.08682797
Validation score: -27.353807
Iteration 86, loss = 254.29729325
Validation score: -26.709802
Iteration 87, loss = 248.27197314
Validation score: -26.039931
Iteration 88, loss = 242.00634738
Validation score: -25.342651
Iteration 89, loss = 235.48630929
Validation score: -24.616996
Iteration 90, loss = 228.70123752
Validation score: -23.861922
Iteration 91, loss = 221.64252548
Validation score: -23.076381
Iteration 92, loss = 214.30156729
```

```
Validation score: -22.259518
Iteration 93, loss = 206.62612784
Validation score: -21.410121
Iteration 94, loss = 198.57533239
Validation score: -20.534939
Iteration 95, loss = 190.25619605
Validation score: -19.616574
Iteration 96, loss = 181.63727407
Validation score: -18.644923
Iteration 97, loss = 172.67667590
Validation score: -17.648191
Iteration 98, loss = 163.47884339
Validation score: -16.625412
Iteration 99, loss = 154.05155372
Validation score: -15.579531
Iteration 100, loss = 144.42814528
Validation score: -14.514416
Iteration 101, loss = 134.64760941
Validation score: -13.434930
Iteration 102, loss = 124.76118526
Validation score: -12.347542
Iteration 103, loss = 114.83384452
Validation score: -11.259781
Iteration 104, loss = 104.94547459
Validation score: -10.181141
Iteration 105, loss = 95.19092942
Validation score: -9.122736
Iteration 106, loss = 85.67163191
Validation score: -8.095682
Iteration 107, loss = 76.49275391
Validation score: -7.111635
Iteration 108, loss = 67.76055778
Validation score: -6.181973
Iteration 109, loss = 59.57441366
Validation score: -5.316924
Iteration 110, loss = 52.02007647
Validation score: -4.524855
Iteration 111, loss = 45.16377501
Validation score: -3.811880
Iteration 112, loss = 39.04759434
Validation score: -3.181095
Iteration 113, loss = 33.68717078
Validation score: -2.632579
Iteration 114, loss = 29.07218917
Validation score: -2.163754
Iteration 115, loss = 25.16777779
Validation score: -1.769931
Iteration 116, loss = 21.92171094
Validation score: -1.444220
Iteration 117, loss = 19.26607721
Validation score: -1.179361
Iteration 118, loss = 17.12900897
Validation score: -0.967032
Iteration 119, loss = 15.43300143
Validation score: -0.798919
Iteration 120, loss = 14.10518665
Validation score: -0.667457
Iteration 121, loss = 13.07813540
Validation score: -0.565669
Iteration 122, loss = 12.29273272
Validation score: -0.487594
```

```
Iteration 123, loss = 11.69799802
Validation score: -0.428160
Iteration 124, loss = 11.25138651
Validation score: -0.383173
Iteration 125, loss = 10.91835441
Validation score: -0.349251
Iteration 126, loss = 10.67144996
Validation score: -0.323773
Iteration 127, loss = 10.48893046
Validation score: -0.304561
Iteration 128, loss = 10.35427632
Validation score: -0.290076
Iteration 129, loss = 10.25502280
Validation score: -0.279107
Iteration 130, loss = 10.18164482
Validation score: -0.270730
Iteration 131, loss = 10.12704013
Validation score: -0.264258
Iteration 132, loss = 10.08687281
Validation score: -0.259329
Iteration 133, loss = 10.05631835
Validation score: -0.255353
Iteration 134, loss = 10.03258018
Validation score: -0.252116
Iteration 135, loss = 10.01388364
Validation score: -0.249431
Iteration 136, loss = 9.99880909
Validation score: -0.247153
Iteration 137, loss = 9.98633258
Validation score: -0.245175
Iteration 138, loss = 9.97572576
Validation score: -0.243418
Iteration 139, loss = 9.96645347
Validation score: -0.241882
Iteration 140, loss = 9.95815243
Validation score: -0.240475
Iteration 141, loss = 9.95056786
Validation score: -0.239156
Iteration 142, loss = 9.94351257
Validation score: -0.237905
Iteration 143, loss = 9.93685170
Validation score: -0.236708
Iteration 144, loss = 9.93047769
Validation score: -0.235554
Iteration 145, loss = 9.92431206
Validation score: -0.234438
Iteration 146, loss = 9.91829759
Validation score: -0.233357
Iteration 147, loss = 9.91238348
Validation score: -0.232310
Iteration 148, loss = 9.90735114
Validation score: -0.231287
Iteration 149, loss = 9.90277959
Validation score: -0.230310
Iteration 150, loss = 9.89766356
Validation score: -0.229382
Iteration 151, loss = 9.89230176
Validation score: -0.228494
Iteration 152, loss = 9.88680400
Validation score: -0.227642
Iteration 153, loss = 9.88123679
```

```
Validation score: -0.226828
Iteration 154, loss = 9.87553208
Validation score: -0.226040
Iteration 155, loss = 9.86975655
Validation score: -0.225295
Iteration 156, loss = 9.86411211
Validation score: -0.224584
Iteration 157, loss = 9.85876536
Validation score: -0.223903
Iteration 158, loss = 9.85341662
Validation score: -0.223246
Iteration 159, loss = 9.84804435
Validation score: -0.222612
Iteration 160, loss = 9.84280080
Validation score: -0.221995
Iteration 161, loss = 9.83752274
Validation score: -0.221392
Iteration 162, loss = 9.83227485
Validation score: -0.220801
Iteration 163, loss = 9.82709407
Validation score: -0.220222
Iteration 164, loss = 9.82199731
Validation score: -0.219652
Iteration 165, loss = 9.81699358
Validation score: -0.219091
Iteration 166, loss = 9.81208537
Validation score: -0.218540
Iteration 167, loss = 9.80728056
Validation score: -0.217999
Iteration 168, loss = 9.80258024
Validation score: -0.217465
Iteration 169, loss = 9.79798539
Validation score: -0.216939
Iteration 170, loss = 9.79348926
Validation score: -0.216421
Iteration 171, loss = 9.78909164
Validation score: -0.215910
Iteration 172, loss = 9.78478092
Validation score: -0.215427
Iteration 173, loss = 9.78056500
Validation score: -0.214984
Iteration 174, loss = 9.77643671
Validation score: -0.214546
Iteration 175, loss = 9.77239218
Validation score: -0.214115
Iteration 176, loss = 9.76842787
Validation score: -0.213691
Iteration 177, loss = 9.76454044
Validation score: -0.213272
Iteration 178, loss = 9.76073042
Validation score: -0.212861
Iteration 179, loss = 9.75699462
Validation score: -0.212455
Iteration 180, loss = 9.75333022
Validation score: -0.212056
Iteration 181, loss = 9.74973389
Validation score: -0.211664
Iteration 182, loss = 9.74620053
Validation score: -0.211276
Iteration 183, loss = 9.74272606
Validation score: -0.210895
```

```
Iteration 184, loss = 9.73931012
Validation score: -0.210518
Iteration 185, loss = 9.73595355
Validation score: -0.210146
Iteration 186, loss = 9.73265711
Validation score: -0.209782
Iteration 187, loss = 9.72941678
Validation score: -0.209422
Iteration 188, loss = 9.72622919
Validation score: -0.209068
Iteration 189, loss = 9.72309153
Validation score: -0.208720
Iteration 190, loss = 9.72000064
Validation score: -0.208376
Iteration 191, loss = 9.71696418
Validation score: -0.208039
Iteration 192, loss = 9.71398081
Validation score: -0.207708
Iteration 193, loss = 9.71104598
Validation score: -0.207382
Iteration 194, loss = 9.70858237
Validation score: -0.206984
Iteration 195, loss = 9.70571547
Validation score: -0.206648
Iteration 196, loss = 9.70261012
Validation score: -0.206344
Iteration 197, loss = 9.70004290
Validation score: -0.206027
Iteration 198, loss = 9.69739790
Validation score: -0.205716
Iteration 199, loss = 9.69473798
Validation score: -0.205412
Iteration 200, loss = 9.69215800
Validation score: -0.205110
```

Out[46]:

```
MLPRegressor(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.
9,
             beta_2=0.999, early_stopping=True, epsilon=1e-08,
             hidden_layer_sizes=(10, 12, 4), learning_rate='constant',
             learning_rate_init=0.001, max_iter=200, momentum=0.9,
             n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
             random_state=42, shuffle=False, solver='adam', tol=0.0001,
             validation_fraction=0.1, verbose=True, warm_start=False)
```

In [47]:

```
net.score(X_train,Y_train)
```

Out[47]:

```
-0.14286928478012748
```

In [48]:

```
net.score(X_test,Y_test)
```

Out[48]:

```
-0.10357021579147163
```

In [55]:

```
#let s see error on train and validation
print('Error on train is ',mean_absolute_error(net.predict(X_train),Y_train))
print('Error on validation is ',mean_absolute_error(net.predict(X_test),Y_test))
```

```
Error on train is  2.2031991484935545
Error on validation is  2.288107806077934
```

In [49]:

```
rf = RandomForestRegressor(
    n_estimators=150,
    max_depth=12,
    min_samples_split=7
)
```

In [50]:

```
rf.fit(X_train,Y_train)
```

Out[50]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=12,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=7,
                      min_weight_fraction_leaf=0.0, n_estimators=150,
                      n_jobs=None, oob_score=False, random_state=None,
                      verbose=0, warm_start=False)
```

In [51]:

```
rf.score(X_train,Y_train)
```

Out[51]:

```
0.9312987155361563
```

In [52]:

```
rf.score(X_test,Y_test)
```

Out[52]:

```
0.8389546138274998
```

In [54]:

```
#let s see error on train and validation
print('Error on train is ',mean_absolute_error(rf.predict(X_train),Y_train))
print('Error on validation is ',mean_absolute_error(rf.predict(X_test),Y_test))
```

```
Error on train is  0.6437125337583561
Error on validation is  0.9477296059540616
```