Progetto 21-22

June 6, 2022

1 Progetto DATA & WEB MINING 2021/2022

1.1 Zillow Prize Data set

Importiamo tutte le librerie e funzionalità utili

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import statsmodels.api as sm
import plotly.graph_objects as go
import warnings
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.feature_selection import RFECV
from plotly.subplots import make_subplots
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor
from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder,MinMaxScaler
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression
from matplotlib.ticker import MaxNLocator
from numpy import mean

warnings.filterwarnings("ignore")
plt.style.use("seaborn")
```

1.1.1 Processing dei dati

[ ]:
```
# importiamo i dataset
prop_data = pd.read_csv("properties_2016.csv")
train_data = pd.read_csv("train_2016_v2.csv")
```

Effettuiamo il merge delle properties e del train che ci è stato forinito per ottenere il trining dataset completo

[ ]:
```
# effettuiamo la fusione dei due dataset utilizzando parcelid come elemento␣
↪comune
df_train = pd.merge(train_data,prop_data, how="left", on="parcelid")
df_train.to_csv('completo.csv')
df_train
```

[ ]:

	parcelid	logerror	transactiondate	airconditioningtypeid	\
0	11016594	0.0276	2016-01-01	1.0	
1	14366692	-0.1684	2016-01-01	NaN	
2	12098116	-0.0040	2016-01-01	1.0	
3	12643413	0.0218	2016-01-02	1.0	
4	14432541	-0.0050	2016-01-02	NaN	
…	…	…	…	…	
90270	10774160	-0.0356	2016-12-30	1.0	
90271	12046695	0.0070	2016-12-30	NaN	
90272	12995401	-0.2679	2016-12-30	NaN	
90273	11402105	0.0602	2016-12-30	NaN	
90274	12566293	0.4207	2016-12-30	NaN	

	architecturalstyletypeid	basementsqft	bathroomcnt	bedroomcnt	\
0	NaN	NaN	2.0	3.0	
1	NaN	NaN	3.5	4.0	
2	NaN	NaN	3.0	2.0	
3	NaN	NaN	2.0	2.0	
4	NaN	NaN	2.5	4.0	
…	…	…	…	…	
90270	NaN	NaN	1.0	1.0	
90271	NaN	NaN	3.0	3.0	
90272	NaN	NaN	2.0	4.0	
90273	NaN	NaN	2.0	2.0	
90274	NaN	NaN	1.0	3.0	

	buildingclasstypeid	buildingqualitytypeid	…	numberofstories	\
0	NaN	4.0	…	NaN	
1	NaN	NaN	…	NaN	
2	NaN	4.0	…	NaN	
3	NaN	4.0	…	NaN	
4	NaN	NaN	…	2.0	
…	…	…	…	…	
90270	NaN	4.0	…	NaN	
90271	NaN	4.0	…	NaN	
90272	NaN	7.0	…	NaN	
90273	NaN	4.0	…	NaN	
90274	NaN	7.0	…	NaN	

	fireplaceflag	structuretaxvaluedollarcnt	taxvaluedollarcnt	\
0	NaN	122754.0	360170.0	

```
1                NaN                    346458.0           585529.0
2                NaN                     61994.0           119906.0
3                NaN                    171518.0           244880.0
4                NaN                    169574.0           434551.0
…                …                           …                  …
90270            NaN                     43800.0           191000.0
90271            NaN                    117893.0           161111.0
90272            NaN                     22008.0            38096.0
90273            NaN                    132991.0           165869.0
90274            NaN                     66258.0           163037.0

       assessmentyear  landtaxvaluedollarcnt  taxamount  taxdelinquencyflag  \
0              2015.0                237416.0    6735.88                 NaN
1              2015.0                239071.0   10153.02                 NaN
2              2015.0                 57912.0   11484.48                 NaN
3              2015.0                 73362.0    3048.74                 NaN
4              2015.0                264977.0    5488.96                 NaN
…               …                          …         …                  …
90270          2015.0                147200.0    2495.24                 NaN
90271          2015.0                 43218.0    1886.54                 NaN
90272          2015.0                 16088.0    1925.70                   Y
90273          2015.0                 32878.0    2285.57                 NaN
90274          2015.0                 96779.0    2560.96                 NaN

       taxdelinquencyyear  censustractandblock
0                     NaN         6.037107e+13
1                     NaN                  NaN
2                     NaN         6.037464e+13
3                     NaN         6.037296e+13
4                     NaN         6.059042e+13
…                     …                    …
90270                 NaN         6.037132e+13
90271                 NaN         6.037301e+13
90272                14.0         6.037433e+13
90273                 NaN         6.037601e+13
90274                 NaN         6.037544e+13

[90275 rows x 60 columns]
```

facciamo un check delle feature che abbiamo e dei rispettivi valori che gli sono associati

[ ]:
```python
def check_info(df):
    print(
        f'Il conteggio dei tipi associati alle colonne è il seguente :
↪\n\n{df_train.dtypes.value_counts()}')
    print('\nI valori unici associati ad ogni colonna sono i seguenti :\n')
    for f in df.columns:
```

```
        if df[f].dtypes == object :
            print(f, np.unique(df[f].astype(str).values))
        else:
            print(f, np.unique(df[f].values))


check_info(df_train)
```

```
Il conteggio dei tipi associati alle colonne è il seguente :

float64    53
object      6
int64       1
dtype: int64

I valori unici associati ad ogni colonna sono i seguenti :

parcelid [ 10711738  10711755  10711805 … 162960801 162960829 162960842]
logerror [-4.605 -4.51  -3.194 …  4.445  4.52   4.737]
transactiondate ['2016-01-01' '2016-01-02' '2016-01-03' '2016-01-04'
'2016-01-05'
 '2016-01-06' '2016-01-07' '2016-01-08' '2016-01-09' '2016-01-10'
 '2016-01-11' '2016-01-12' '2016-01-13' '2016-01-14' '2016-01-15'
 '2016-01-16' '2016-01-17' '2016-01-18' '2016-01-19' '2016-01-20'
 '2016-01-21' '2016-01-22' '2016-01-23' '2016-01-24' '2016-01-25'
 '2016-01-26' '2016-01-27' '2016-01-28' '2016-01-29' '2016-01-30'
 '2016-01-31' '2016-02-01' '2016-02-02' '2016-02-03' '2016-02-04'
 '2016-02-05' '2016-02-06' '2016-02-07' '2016-02-08' '2016-02-09'
 '2016-02-10' '2016-02-11' '2016-02-12' '2016-02-13' '2016-02-14'
 '2016-02-15' '2016-02-16' '2016-02-17' '2016-02-18' '2016-02-19'
 '2016-02-20' '2016-02-21' '2016-02-22' '2016-02-23' '2016-02-24'
 '2016-02-25' '2016-02-26' '2016-02-27' '2016-02-28' '2016-02-29'
 '2016-03-01' '2016-03-02' '2016-03-03' '2016-03-04' '2016-03-05'
 '2016-03-06' '2016-03-07' '2016-03-08' '2016-03-09' '2016-03-10'
 '2016-03-11' '2016-03-13' '2016-03-14' '2016-03-15' '2016-03-16'
 '2016-03-17' '2016-03-18' '2016-03-19' '2016-03-20' '2016-03-21'
 '2016-03-22' '2016-03-23' '2016-03-24' '2016-03-25' '2016-03-27'
 '2016-03-28' '2016-03-29' '2016-03-30' '2016-03-31' '2016-04-01'
 '2016-04-02' '2016-04-03' '2016-04-04' '2016-04-05' '2016-04-06'
 '2016-04-07' '2016-04-08' '2016-04-09' '2016-04-10' '2016-04-11'
 '2016-04-12' '2016-04-13' '2016-04-14' '2016-04-15' '2016-04-16'
 '2016-04-17' '2016-04-18' '2016-04-19' '2016-04-20' '2016-04-21'
 '2016-04-22' '2016-04-23' '2016-04-24' '2016-04-25' '2016-04-26'
 '2016-04-27' '2016-04-28' '2016-04-29' '2016-04-30' '2016-05-01'
 '2016-05-02' '2016-05-03' '2016-05-04' '2016-05-05' '2016-05-06'
 '2016-05-07' '2016-05-08' '2016-05-09' '2016-05-10' '2016-05-11'
 '2016-05-12' '2016-05-13' '2016-05-14' '2016-05-15' '2016-05-16'
```

```
 '2016-05-17' '2016-05-18' '2016-05-19' '2016-05-20' '2016-05-22'
 '2016-05-23' '2016-05-24' '2016-05-25' '2016-05-26' '2016-05-27'
 '2016-05-28' '2016-05-29' '2016-05-30' '2016-05-31' '2016-06-01'
 '2016-06-02' '2016-06-03' '2016-06-04' '2016-06-05' '2016-06-06'
 '2016-06-07' '2016-06-08' '2016-06-09' '2016-06-10' '2016-06-11'
 '2016-06-12' '2016-06-13' '2016-06-14' '2016-06-15' '2016-06-16'
 '2016-06-17' '2016-06-18' '2016-06-19' '2016-06-20' '2016-06-21'
 '2016-06-22' '2016-06-23' '2016-06-24' '2016-06-25' '2016-06-26'
 '2016-06-27' '2016-06-28' '2016-06-29' '2016-06-30' '2016-07-01'
 '2016-07-02' '2016-07-04' '2016-07-05' '2016-07-06' '2016-07-07'
 '2016-07-08' '2016-07-09' '2016-07-10' '2016-07-11' '2016-07-12'
 '2016-07-13' '2016-07-14' '2016-07-15' '2016-07-16' '2016-07-17'
 '2016-07-18' '2016-07-19' '2016-07-20' '2016-07-21' '2016-07-22'
 '2016-07-23' '2016-07-24' '2016-07-25' '2016-07-26' '2016-07-27'
 '2016-07-28' '2016-07-29' '2016-07-30' '2016-07-31' '2016-08-01'
 '2016-08-02' '2016-08-03' '2016-08-04' '2016-08-05' '2016-08-06'
 '2016-08-07' '2016-08-08' '2016-08-09' '2016-08-10' '2016-08-11'
 '2016-08-12' '2016-08-13' '2016-08-14' '2016-08-15' '2016-08-16'
 '2016-08-17' '2016-08-18' '2016-08-19' '2016-08-20' '2016-08-21'
 '2016-08-22' '2016-08-23' '2016-08-24' '2016-08-25' '2016-08-26'
 '2016-08-27' '2016-08-28' '2016-08-29' '2016-08-30' '2016-08-31'
 '2016-09-01' '2016-09-02' '2016-09-03' '2016-09-04' '2016-09-05'
 '2016-09-06' '2016-09-07' '2016-09-08' '2016-09-09' '2016-09-10'
 '2016-09-11' '2016-09-12' '2016-09-13' '2016-09-14' '2016-09-15'
 '2016-09-16' '2016-09-17' '2016-09-18' '2016-09-19' '2016-09-20'
 '2016-09-21' '2016-09-22' '2016-09-23' '2016-09-24' '2016-09-25'
 '2016-09-26' '2016-09-27' '2016-09-28' '2016-09-29' '2016-09-30'
 '2016-10-01' '2016-10-02' '2016-10-03' '2016-10-04' '2016-10-05'
 '2016-10-06' '2016-10-07' '2016-10-08' '2016-10-09' '2016-10-10'
 '2016-10-11' '2016-10-12' '2016-10-13' '2016-10-14' '2016-10-17'
 '2016-10-18' '2016-10-19' '2016-10-20' '2016-10-21' '2016-10-22'
 '2016-10-24' '2016-10-25' '2016-10-26' '2016-10-27' '2016-10-28'
 '2016-10-29' '2016-10-30' '2016-10-31' '2016-11-01' '2016-11-02'
 '2016-11-03' '2016-11-04' '2016-11-06' '2016-11-07' '2016-11-08'
 '2016-11-09' '2016-11-10' '2016-11-12' '2016-11-13' '2016-11-14'
 '2016-11-15' '2016-11-16' '2016-11-17' '2016-11-18' '2016-11-19'
 '2016-11-20' '2016-11-21' '2016-11-22' '2016-11-23' '2016-11-24'
 '2016-11-27' '2016-11-28' '2016-11-29' '2016-11-30' '2016-12-01'
 '2016-12-02' '2016-12-04' '2016-12-05' '2016-12-06' '2016-12-07'
 '2016-12-08' '2016-12-09' '2016-12-11' '2016-12-12' '2016-12-13'
 '2016-12-14' '2016-12-15' '2016-12-16' '2016-12-17' '2016-12-18'
 '2016-12-19' '2016-12-20' '2016-12-21' '2016-12-22' '2016-12-23'
 '2016-12-24' '2016-12-25' '2016-12-26' '2016-12-27' '2016-12-28'
 '2016-12-29' '2016-12-30']
airconditioningtypeid [ 1.  3.  5.  9. 11. 13. nan]
architecturalstyletypeid [ 2.  3.  7.  8. 10. 21. nan]
basementsqft [ 100.  162.  168.  184.  196.  198.  234.  238.  260.  312.  330.
485.
```

```
  493.  510.  515.  540.  557.  564.  579.  585.  608.  616.  671.  676.
  690.  700.  760.  771.  802.  814.  831.  913. 1048. 1210. 1312. 1350.
 1528. 1551. 1555.   nan]
bathroomcnt [ 0.   1.   1.5  2.   2.5  3.   3.5  4.   4.5  5.   5.5  6.   6.5
7.
  7.5  8.   8.5  9.  10.  11.  12.  15.  20. ]
bedroomcnt [ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 16.]
buildingclasstypeid [ 4. nan]
buildingqualitytypeid [ 1.  4.  6.  7.  8. 10. 11. 12. nan]
calculatedbathnbr [ 1.   1.5  2.   2.5  3.   3.5  4.   4.5  5.   5.5  6.   6.5
7.   7.5
  8.   8.5  9.  10.  11.  12.  15.  20.   nan]
decktypeid [66. nan]
finishedfloor1squarefeet [  44.   47.   49. … 6870. 7625.   nan]
calculatedfinishedsquarefeet [2.0000e+00 4.0000e+01 6.6000e+01 … 2.0013e+04
2.2741e+04        nan]
finishedsquarefeet12 [2.0000e+00 4.0000e+01 6.6000e+01 … 1.8577e+04 2.0013e+04
nan]
finishedsquarefeet13 [1056. 1152. 1248. 1344. 1392. 1416. 1440. 1464. 1536.
1566. 1584.   nan]
finishedsquarefeet15 [  560.   609.   646. …  8558. 22741.    nan]
finishedsquarefeet50 [  44.   47.   49. … 8195. 8352.   nan]
finishedsquarefeet6 [ 257.  300.  360.  384.  438.  480.  520.  529.  531.  572.
598.  605.
  624.  650.  666.  672.  684.  686.  700.  720.  722.  726.  728.  731.
  732.  735.  736.  754.  760.  765.  783.  784.  786.  800.  816.  820.
  824.  836.  837.  839.  842.  844.  852.  854.  856.  858.  873.  880.
  882.  899.  900.  922.  924.  931.  935.  944.  952.  954.  956.  960.
  967.  972.  980.  988.  991.  993. 1000. 1006. 1008. 1009. 1016. 1027.
 1029. 1036. 1044. 1070. 1074. 1078. 1080. 1082. 1092. 1102. 1104. 1106.
 1112. 1135. 1140. 1151. 1152. 1154. 1163. 1180. 1208. 1212. 1216. 1224.
 1228. 1248. 1256. 1280. 1290. 1316. 1320. 1326. 1328. 1346. 1360. 1376.
 1409. 1415. 1416. 1430. 1443. 1446. 1448. 1451. 1457. 1462. 1471. 1482.
 1484. 1488. 1492. 1508. 1517. 1518. 1531. 1541. 1553. 1556. 1589. 1590.
 1591. 1593. 1600. 1604. 1605. 1629. 1645. 1650. 1654. 1658. 1670. 1674.
 1679. 1682. 1712. 1730. 1750. 1759. 1776. 1777. 1780. 1790. 1794. 1809.
 1810. 1816. 1820. 1824. 1835. 1843. 1844. 1847. 1874. 1876. 1878. 1879.
 1880. 1886. 1888. 1904. 1920. 1921. 1926. 1968. 1975. 1986. 2028. 2041.
 2046. 2048. 2052. 2068. 2075. 2100. 2103. 2106. 2153. 2172. 2175. 2179.
 2180. 2191. 2193. 2194. 2195. 2202. 2224. 2235. 2237. 2241. 2252. 2260.
 2277. 2291. 2293. 2294. 2303. 2323. 2339. 2361. 2363. 2376. 2388. 2428.
 2430. 2460. 2467. 2486. 2490. 2498. 2517. 2530. 2537. 2556. 2564. 2576.
 2582. 2585. 2592. 2628. 2630. 2631. 2647. 2650. 2652. 2662. 2663. 2670.
 2686. 2743. 2754. 2760. 2859. 2870. 2898. 2921. 3000. 3004. 3032. 3055.
 3071. 3087. 3100. 3112. 3150. 3160. 3163. 3164. 3173. 3192. 3234. 3240.
 3278. 3300. 3355. 3368. 3377. 3380. 3388. 3423. 3431. 3432. 3462. 3469.
 3470. 3488. 3500. 3515. 3525. 3527. 3530. 3540. 3551. 3554. 3565. 3600.
 3603. 3621. 3681. 3707. 3718. 3722. 3728. 3732. 3750. 3752. 3762. 3780.
```

```
 3801. 3828. 3837. 3844. 3845. 3882. 3886. 3891. 3912. 3933. 3952. 3986.
 3988. 3996. 4076. 4087. 4102. 4174. 4184. 4206. 4209. 4225. 4249. 4273.
 4274. 4288. 4302. 4303. 4304. 4321. 4345. 4352. 4355. 4369. 4379. 4380.
 4414. 4442. 4446. 4452. 4520. 4553. 4600. 4692. 4699. 4773. 4788. 4790.
 4869. 4895. 4967. 4996. 5229. 5268. 5287. 5300. 6265. 6337. 6338. 7224.
   nan]
fips [6037. 6059. 6111.]
fireplacecnt [ 1.  2.  3.  4.  5. nan]
fullbathcnt [ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 15. 20. nan]
garagecarcnt [ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 13. 14. 24. nan]
garagetotalsqft [   0.  126.  150.  158.  160.  162.  168.  171.  175.  176.
180.  189.
  190.  192.  194.  195.  198.  200.  201.  204.  207.  208.  209.  210.
  211.  212.  213.  215.  216.  218.  219.  220.  222.  224.  225.  228.
  230.  231.  234.  235.  236.  237.  238.  239.  240.  241.  242.  246.
  247.  250.  252.  253.  256.  258.  259.  260.  261.  262.  263.  264.
  265.  266.  269.  270.  272.  273.  274.  276.  280.  281.  284.  285.
  286.  288.  289.  290.  292.  294.  295.  296.  297.  298.  299.  300.
  301.  304.  305.  306.  307.  308.  309.  310.  311.  312.  313.  315.
  320.  321.  322.  323.  324.  325.  327.  329.  330.  331.  333.  334.
  336.  338.  339.  340.  341.  342.  345.  346.  347.  348.  349.  350.
  351.  352.  355.  356.  357.  358.  359.  360.  361.  362.  364.  365.
  366.  367.  368.  369.  370.  371.  372.  373.  374.  375.  376.  377.
  378.  379.  380.  381.  382.  383.  384.  385.  386.  387.  388.  389.
  390.  391.  392.  393.  394.  395.  396.  397.  398.  399.  400.  401.
  402.  403.  404.  405.  406.  407.  408.  409.  410.  411.  412.  413.
  414.  415.  416.  417.  418.  419.  420.  421.  422.  423.  424.  425.
  426.  427.  428.  429.  430.  431.  432.  433.  434.  435.  436.  437.
  438.  439.  440.  441.  442.  443.  444.  445.  446.  447.  448.  449.
  450.  451.  452.  453.  454.  455.  456.  457.  458.  459.  460.  461.
  462.  463.  464.  465.  466.  467.  468.  469.  470.  471.  472.  473.
  474.  475.  476.  477.  478.  479.  480.  481.  482.  483.  484.  485.
  486.  487.  488.  489.  490.  491.  492.  493.  494.  495.  496.  497.
  498.  499.  500.  501.  502.  503.  504.  505.  506.  507.  508.  509.
  510.  511.  512.  513.  514.  515.  516.  517.  518.  519.  520.  521.
  522.  523.  524.  525.  526.  527.  528.  529.  530.  531.  532.  533.
  534.  535.  536.  537.  538.  539.  540.  541.  542.  543.  544.  545.
  546.  547.  548.  549.  550.  551.  552.  553.  554.  555.  556.  557.
  558.  559.  560.  561.  562.  563.  564.  565.  566.  567.  568.  569.
  570.  571.  572.  573.  574.  575.  576.  577.  578.  579.  580.  581.
  582.  583.  584.  585.  586.  587.  588.  589.  590.  591.  592.  593.
  594.  595.  596.  597.  598.  599.  600.  601.  602.  603.  604.  605.
  606.  607.  608.  609.  610.  611.  612.  613.  614.  615.  616.  617.
  618.  619.  620.  621.  622.  623.  624.  625.  626.  627.  628.  629.
  630.  631.  632.  633.  634.  635.  636.  637.  638.  639.  640.  641.
  642.  643.  644.  645.  646.  647.  648.  649.  650.  651.  652.  653.
  654.  655.  656.  657.  658.  659.  660.  661.  662.  663.  664.  665.
  666.  667.  668.  669.  670.  671.  672.  673.  674.  675.  676.  677.
```

```
  678.  679.  680.  681.  682.  683.  684.  685.  686.  687.  688.  689.
  690.  691.  692.  693.  694.  695.  696.  697.  698.  699.  700.  701.
  702.  703.  704.  705.  706.  707.  708.  709.  710.  711.  712.  713.
  714.  715.  716.  717.  718.  719.  720.  721.  722.  723.  724.  725.
  726.  727.  728.  729.  730.  731.  732.  733.  734.  735.  736.  737.
  738.  739.  740.  741.  742.  743.  744.  745.  746.  747.  748.  750.
  751.  752.  753.  754.  755.  756.  757.  758.  759.  760.  761.  762.
  763.  764.  765.  766.  767.  768.  769.  770.  771.  772.  773.  775.
  776.  777.  778.  779.  780.  781.  782.  783.  784.  785.  786.  787.
  788.  790.  791.  792.  793.  794.  795.  796.  797.  798.  799.  800.
  801.  802.  803.  804.  805.  806.  807.  808.  810.  811.  812.  813.
  814.  815.  816.  817.  818.  820.  821.  822.  823.  824.  825.  826.
  827.  828.  829.  830.  832.  833.  834.  835.  836.  837.  838.  839.
  840.  841.  842.  844.  845.  846.  847.  848.  850.  851.  852.  853.
  854.  856.  857.  858.  861.  862.  863.  864.  865.  868.  869.  870.
  871.  872.  873.  874.  875.  876.  877.  878.  879.  880.  882.  883.
  887.  888.  892.  893.  894.  895.  897.  898.  900.  902.  903.  904.
  905.  906.  908.  910.  913.  914.  917.  919.  920.  923.  924.  925.
  926.  928.  929.  930.  931.  934.  935.  936.  938.  939.  940.  942.
  944.  945.  946.  947.  949.  950.  951.  952.  954.  958.  959.  960.
  962.  963.  965.  966.  967.  968.  969.  971.  973.  974.  975.  976.
  978.  981.  982.  983.  986.  987.  988.  989.  992.  993.  995.  999.
 1000. 1001. 1002. 1004. 1005. 1007. 1008. 1009. 1011. 1012. 1014. 1016.
 1017. 1018. 1019. 1020. 1022. 1023. 1025. 1027. 1028. 1030. 1035. 1038.
 1040. 1045. 1046. 1048. 1050. 1051. 1054. 1056. 1057. 1062. 1063. 1064.
 1070. 1072. 1073. 1076. 1077. 1080. 1082. 1083. 1087. 1093. 1098. 1104.
 1109. 1110. 1111. 1113. 1116. 1120. 1124. 1132. 1135. 1138. 1141. 1142.
 1149. 1152. 1159. 1162. 1164. 1168. 1170. 1180. 1186. 1187. 1190. 1194.
 1202. 1215. 1216. 1217. 1223. 1225. 1229. 1234. 1239. 1247. 1251. 1252.
 1253. 1258. 1260. 1264. 1269. 1280. 1284. 1290. 1296. 1300. 1311. 1313.
 1320. 1323. 1350. 1353. 1354. 1362. 1365. 1373. 1380. 1392. 1410. 1411.
 1426. 1427. 1442. 1447. 1456. 1476. 1500. 1503. 1509. 1512. 1529. 1533.
 1536. 1539. 1568. 1586. 1624. 1630. 1674. 1686. 1740. 1790. 1794. 1831.
 1845. 1887. 1896. 1932. 2003. 2010. 2011. 2091. 2104. 2140. 2206. 2210.
 2243. 2293. 2352. 2441. 2446. 2535. 2685. 2700. 2762. 2766. 2777. 3045.
 3109. 3348. 3362. 4048. 4384. 7339.   nan]
hashottuborspa ['True' 'nan']
heatingorsystemtypeid [ 1.  2.  6.  7. 10. 11. 12. 13. 14. 18. 20. 24. nan]
latitude [33339295. 33340045. 33340134. … 34813287. 34813292. 34816009.]
longitude [-1.19447865e+08 -1.19447353e+08 -1.19447010e+08 … -1.17556200e+08
 -1.17555136e+08 -1.17554924e+08]
lotsizesquarefeet [1.670000e+02 4.350000e+02 4.380000e+02 … 3.589145e+06
6.971010e+06
          nan]
poolcnt [ 1. nan]
poolsizesum [  28.   40.   49.   91.  160.  164.  200.  207.  216.  242.  250.
254.
  264.  265.  270.  276.  277.  280.  288.  290.  291.  294.  295.  299.
```

```
  300.  304.  308.  310.  312.  319.  320.  321.  324.  325.  330.  333.
  336.  338.  340.  342.  343.  345.  350.  352.  356.  357.  360.  364.
  365.  366.  367.  368.  369.  370.  371.  372.  375.  377.  378.  379.
  380.  382.  384.  385.  386.  390.  392.  394.  395.  396.  397.  398.
  400.  401.  403.  404.  405.  406.  408.  411.  412.  413.  415.  416.
  418.  419.  420.  421.  422.  425.  426.  428.  429.  430.  432.  434.
  435.  440.  441.  442.  443.  444.  447.  448.  450.  451.  455.  456.
  460.  461.  462.  463.  465.  468.  471.  472.  475.  476.  477.  480.
  483.  485.  486.  487.  489.  490.  492.  495.  496.  500.  501.  503.
  504.  505.  506.  510.  512.  513.  514.  516.  518.  520.  521.  523.
  524.  525.  527.  528.  530.  531.  534.  535.  536.  537.  538.  539.
  540.  544.  546.  547.  550.  551.  554.  555.  556.  558.  560.  561.
  563.  564.  567.  568.  570.  572.  575.  576.  580.  581.  583.  584.
  585.  588.  589.  591.  592.  593.  594.  595.  600.  608.  610.  612.
  615.  620.  623.  624.  625.  626.  627.  629.  630.  631.  632.  634.
  640.  642.  645.  646.  647.  648.  649.  650.  653.  664.  665.  666.
  668.  670.  672.  680.  682.  684.  686.  690.  691.  700.  702.  704.
  705.  707.  714.  720.  722.  727.  736.  740.  745.  748.  750.  755.
  756.  759.  760.  775.  780.  795.  800.  810.  815.  820.  830.  836.
  838.  851.  855.  862.  870.  880.  893.  900.  908.  920.  948.  960.
  968.  971. 1000. 1020. 1052. 1125. 1220. 1749. 1750.   nan]
pooltypeid10 [ 1. nan]
pooltypeid2 [ 1. nan]
pooltypeid7 [ 1. nan]
propertycountylandusecode ['0' '010' '0100' '0101' '0102' '0103' '0104' '0108'
'0109' '010C' '010D'
 '010E' '010F' '010G' '010H' '010M' '010V' '0110' '0111' '0114' '012C'
 '012D' '012E' '0130' '0131' '01DC' '01DD' '01HC' '0200' '0201' '020G'
 '020M' '0210' '0300' '0301' '0303' '030G' '0400' '0401' '040A' '040V'
 '0700' '070D' '1' '100V' '1011' '1012' '1014' '105' '1110' '1111' '1112'
 '1116' '1117' '1128' '1129' '1200' '1210' '122' '1222' '1310' '1321'
 '1333' '135' '1410' '1420' '1421' '1432' '1720' '1722' '200' '34' '38'
 '6050' '73' '8800' '96' 'nan']
propertylandusetypeid [ 31.  47. 246. 247. 248. 260. 261. 263. 264. 265. 266.
267. 269. 275.]
propertyzoningdesc ['1NR1*' '1NR3*' 'AH RM-CD*' … 'WVRR' 'WVRR1-RPD1' 'nan']
rawcensustractandblock [60371011.101001 60371011.101002 60371011.101004 …
61110091.001011
 61110091.001012 61110091.00102 ]
regionidcity [  3491.   4406.   5465.   5534.   6021.   6285.   6395.   6822.
8384.
   9840.  10241.  10389.  10608.  10723.  10734.  10774.  10815.  11626.
  12292.  12447.  12520.  12773.  13091.  13150.  13232.  13311.  13693.
  13716.  14111.  14542.  14634.  14906.  15237.  15554.  16389.  16677.
  16764.  16961.  17150.  17597.  17686.  17882.  18098.  18874.  18875.
  19177.  20008.  21395.  21412.  21778.  22827.  24174.  24245.  24384.
  24435.  24797.  24812.  24832.  25218.  25271.  25458.  25459.  25468.
  25621.  25953.  25974.  26483.  26531.  26964.  26965.  27103.  27110.
```

```
  27183.  27491.  29189.  29712.  30187.  30267.  30399.  30908.  31134.
  32380.  32616.  32753.  32923.  32927.  33252.  33311.  33312.  33612.
  33727.  33836.  33837.  33840.  34037.  34278.  34543.  34636.  34780.
  36502.  37015.  37086.  37688.  37882.  38032.  38980.  39076.  39306.
  39308.  40009.  40081.  40110.  40227.  41673.  42091.  42150.  42967.
  44116.  44833.  45398.  45457.  45602.  45888.  46080.  46098.  46178.
  46298.  46314.  47019.  47198.  47547.  47568.  47695.  47762.  48424.
  50677.  50749.  51239.  51617.  51861.  52650.  52835.  52842.  53027.
  53162.  53571.  53636.  53655.  54053.  54212.  54299.  54311.  54352.
  54722.  54970.  55753.  56780. 113412. 113576. 114828. 114834. 116042.
 118217. 118225. 118694. 118875. 118878. 118880. 118895. 118914. 118994.
 272578. 396053. 396054. 396550. 396551. 396556.     nan]
regionidcounty [1286. 2061. 3101.]
regionidneighborhood [  6952.   7877.  11950.  13017.  13176.  13327.  19265.
19810.  21056.
  22655.  25449.  26134.  27080.  27328.  27431.  27484.  27987.  28119.
  30685.  30731.  31817.  32059.  32368.  33183.  33952.  34213.  36630.
  37739.  37835.  38888.  40215.  40548.  41131.  41466.  46736.  46795.
  47880.  47950.  48200.  48516.  48570.  51224.  51906.  54300. 113455.
 113688. 113713. 113749. 113886. 113910. 114246. 114808. 114909. 114914.
 115609. 115656. 115657. 115729. 115836. 116206. 116302. 116375. 116415.
 116430. 116646. 116692. 116774. 116828. 117023. 117148. 117183. 117954.
 118106. 118208. 118825. 118849. 118872. 118887. 118920. 135143. 155228.
 156012. 191484. 214079. 220669. 224568. 246825. 248165. 259315. 259606.
 259818. 260382. 260611. 262098. 265889. 266606. 267436. 267814. 268002.
 268007. 268010. 268050. 268055. 268057. 268097. 268103. 268118. 268132.
 268134. 268144. 268157. 268160. 268162. 268208. 268236. 268244. 268249.
 268253. 268268. 268269. 268288. 268291. 268293. 268316. 268323. 268334.
 268392. 268403. 268404. 268413. 268430. 268441. 268446. 268451. 268453.
 268454. 268473. 268482. 268496. 268509. 268540. 268546. 268548. 268551.
 268572. 268581. 268585. 268587. 268588. 268596. 268602. 268604. 268605.
 272170. 272725. 272732. 272737. 272912. 272933. 272968. 272969. 273041.
 273042. 273059. 273096. 273109. 273168. 273197. 273198. 273252. 273257.
 273263. 273313. 273339. 273350. 273400. 273486. 273539. 273567. 273572.
 273607. 273611. 273615. 273617. 273633. 273663. 273677. 273789. 273791.
 273825. 273834. 273837. 273866. 273868. 273890. 273930. 274049. 274262.
 274293. 274343. 274358. 274359. 274371. 274392. 274418. 274425. 274494.
 274513. 274514. 274517. 274580. 274582. 274587. 274684. 274695. 274705.
 274750. 274765. 274800. 274815. 274828. 274857. 274895. 274940. 274961.
 274995. 275024. 275067. 275078. 275111. 275129. 275130. 275207. 275210.
 275257. 275287. 275300. 275339. 275340. 275396. 275405. 275411. 275426.
 275428. 275470. 275496. 275512. 275567. 275695. 275738. 275778. 275784.
 275785. 275795. 275824. 275826. 275856. 275857. 275884. 275916. 275958.
 275979. 275989. 275994. 276023. 276061. 276119. 276121. 276157. 276256.
 276257. 276258. 276293. 276299. 276348. 276449. 276450. 276461. 276476.
 276486. 276514. 276581. 276606. 343505. 403127. 403142. 403183. 403184.
 403190. 403191. 403192. 403193. 403194. 403195. 403196. 403197. 411433.
 416302. 416303. 416304. 416305. 416306. 416307. 416308. 416309. 416310.
```

```
 416311. 416312. 416313. 416314. 416315. 416316. 416317. 416318. 416319.
 416320. 416329. 416330. 416331. 416332. 416333. 416334. 416335. 416336.
 416337. 416338. 416339. 416340. 416341. 416342. 416343. 416344. 416345.
 416346. 416347. 416349. 416350. 416860. 416873. 416963. 416964. 416965.
 416966. 416967. 417224. 417225. 417433. 623377. 623378. 623379. 623380.
 623381. 623382. 757352. 760999. 761000. 761001. 761055. 761090. 761094.
 761097. 761098. 761099. 761210. 761211. 761214. 761215. 761218. 761219.
 761220. 761221. 761222. 761223. 761541. 761542. 761543. 761544. 761545.
 761546. 761547. 762175. 762177. 762178. 762179. 762180. 762181. 762183.
 762184. 762185. 762186. 762187. 762188. 762189. 762190. 762191. 762192.
 762527. 762682. 762683. 762684. 762685. 762890. 762927. 762928. 762929.
 762930. 762931. 762932. 762934. 762935. 762937. 762938. 762939. 762940.
 762942. 762944. 762945. 762946. 762947. 762948. 762949. 762950. 762951.
 762952. 762953. 762955. 762956. 762957. 762959. 762960. 762961. 762962.
 762963. 763011. 763012. 763079. 763080. 763089. 763090. 763094. 763169.
 763170. 763171. 763172. 763185. 763217. 763218. 763219. 763220. 763527.
 763679. 763680. 763789. 763791. 764074. 764087. 764088. 764089. 764090.
 764093. 764094. 764095. 764096. 764097. 764098. 764099. 764101. 764102.
 764103. 764104. 764105. 764106. 764107. 764108. 764109. 764134. 764135.
 764136. 764138. 764139. 764140. 764141. 764142. 764143. 764144. 764145.
 764147. 764148. 764149. 764152. 764164. 764165. 764166. 764167.     nan]
regionidzip [ 95982.  95983.  95984.  95985.  95986.  95987.  95988.  95989.
95991.
  95992.  95993.  95994.  95995.  95996.  95997.  95998.  95999.  96000.
  96001.  96002.  96003.  96004.  96005.  96006.  96007.  96008.  96009.
  96010.  96012.  96013.  96014.  96015.  96016.  96017.  96018.  96019.
  96020.  96021.  96022.  96023.  96024.  96025.  96026.  96027.  96028.
  96029.  96030.  96034.  96037.  96038.  96039.  96040.  96042.  96043.
  96044.  96045.  96046.  96047.  96048.  96049.  96050.  96058.  96072.
  96083.  96086.  96087.  96088.  96090.  96091.  96092.  96095.  96097.
  96100.  96101.  96102.  96103.  96104.  96105.  96106.  96107.  96109.
  96110.  96111.  96113.  96116.  96117.  96119.  96120.  96121.  96122.
  96123.  96124.  96125.  96126.  96127.  96128.  96129.  96133.  96134.
  96135.  96136.  96137.  96148.  96149.  96150.  96151.  96152.  96159.
  96160.  96161.  96162.  96163.  96169.  96170.  96171.  96172.  96173.
  96174.  96180.  96181.  96183.  96185.  96186.  96190.  96192.  96193.
  96197.  96201.  96203.  96204.  96206.  96207.  96208.  96210.  96212.
  96213.  96215.  96216.  96217.  96218.  96220.  96221.  96222.  96225.
  96226.  96228.  96229.  96230.  96234.  96236.  96237.  96238.  96239.
  96240.  96241.  96242.  96244.  96245.  96246.  96247.  96265.  96267.
  96268.  96270.  96271.  96273.  96275.  96278.  96280.  96282.  96284.
  96289.  96291.  96292.  96293.  96294.  96295.  96296.  96320.  96321.
  96322.  96323.  96324.  96325.  96326.  96327.  96329.  96330.  96336.
  96337.  96338.  96339.  96341.  96342.  96346.  96349.  96351.  96352.
  96354.  96355.  96356.  96361.  96364.  96366.  96368.  96369.  96370.
  96371.  96373.  96374.  96375.  96377.  96378.  96379.  96383.  96384.
  96385.  96387.  96389.  96393.  96395.  96398.  96401.  96403.  96410.
  96411.  96412.  96414.  96415.  96420.  96424.  96426.  96433.  96434.
```

```
  96436.  96437.  96438.  96446.  96447.  96449.  96450.  96451.  96452.
  96464.  96465.  96467.  96469.  96473.  96474.  96475.  96478.  96479.
  96480.  96485.  96486.  96488.  96489.  96490.  96492.  96494.  96496.
  96497.  96500.  96505.  96506.  96507.  96508.  96510.  96513.  96514.
  96515.  96517.  96522.  96523.  96524.  96525.  96531.  96533.  96939.
  96940.  96941.  96943.  96946.  96947.  96948.  96951.  96952.  96954.
  96956.  96957.  96958.  96959.  96961.  96962.  96963.  96964.  96965.
  96966.  96967.  96969.  96971.  96973.  96974.  96975.  96978.  96979.
  96980.  96981.  96982.  96983.  96985.  96986.  96987.  96989.  96990.
  96993.  96995.  96996.  96998.  97001.  97003.  97004.  97005.  97006.
  97007.  97008.  97016.  97018.  97020.  97021.  97023.  97024.  97025.
  97026.  97027.  97035.  97037.  97039.  97040.  97041.  97043.  97047.
  97048.  97050.  97051.  97052.  97059.  97063.  97064.  97065.  97066.
  97067.  97068.  97078.  97079.  97081.  97083.  97084.  97089.  97091.
  97094.  97097.  97098.  97099.  97101.  97104.  97106.  97107.  97108.
  97109.  97111.  97113.  97116.  97118.  97119.  97298.  97316.  97317.
  97318.  97319.  97323.  97324.  97328.  97329.  97330.  97331.  97344.
 399675.     nan]
roomcnt [ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 18.]
storytypeid [ 7. nan]
threequarterbathnbr [ 1.  2.  3.  4. nan]
typeconstructiontypeid [ 4.  6. 13. nan]
unitcnt [  1.   2.   3.   4.   5.   6.   9.  11.  70. 143.  nan]
yardbuildingsqft17 [  25.   28.   30.   31.   36.   37.   40.   41.   42.   44.
45.   48.
   49.   50.   54.   55.   56.   57.   60.   63.   64.   65.   66.   67.
   68.   70.   72.   75.   77.   78.   79.   80.   81.   83.   84.   85.
   87.   88.   90.   91.   92.   93.   95.   96.   97.   98.   99.  100.
  102.  104.  105.  106.  108.  109.  110.  111.  112.  113.  114.  115.
  116.  117.  119.  120.  121.  122.  123.  124.  125.  126.  127.  128.
  129.  130.  132.  133.  134.  135.  136.  137.  140.  141.  142.  143.
  144.  147.  148.  149.  150.  152.  153.  154.  155.  156.  157.  158.
  159.  160.  161.  162.  164.  165.  166.  167.  168.  170.  171.  172.
  173.  174.  175.  176.  177.  178.  179.  180.  181.  182.  183.  184.
  185.  186.  187.  188.  189.  190.  192.  195.  196.  198.  200.  201.
  203.  204.  205.  206.  207.  208.  209.  210.  215.  216.  217.  218.
  219.  220.  221.  222.  224.  225.  226.  227.  228.  229.  230.  231.
  232.  233.  234.  235.  236.  238.  239.  240.  241.  242.  243.  244.
  245.  247.  248.  250.  251.  252.  253.  254.  255.  256.  258.  259.
  260.  262.  263.  264.  265.  266.  267.  268.  269.  270.  272.  273.
  274.  275.  276.  277.  280.  284.  285.  286.  287.  288.  289.  290.
  292.  293.  294.  296.  297.  298.  299.  300.  302.  303.  304.  305.
  306.  307.  308.  310.  311.  312.  313.  314.  315.  316.  317.  318.
  320.  321.  322.  323.  324.  325.  326.  327.  328.  329.  330.  333.
  334.  336.  337.  338.  340.  341.  342.  343.  344.  345.  346.  347.
  348.  350.  351.  352.  353.  356.  357.  358.  359.  360.  361.  363.
  364.  366.  367.  368.  370.  372.  373.  374.  375.  378.  379.  380.
  382.  384.  385.  388.  389.  390.  391.  392.  394.  395.  396.  400.
```

```
  402.  405.  407.  408.  409.  410.  412.  413.  415.  416.  418.  419.
  420.  421.  423.  424.  425.  426.  427.  428.  429.  430.  431.  432.
  433.  435.  438.  440.  441.  442.  444.  445.  446.  447.  448.  449.
  450.  451.  452.  454.  456.  457.  458.  459.  460.  462.  464.  465.
  467.  468.  469.  470.  471.  472.  474.  475.  476.  477.  479.  480.
  484.  485.  486.  488.  490.  492.  495.  500.  501.  503.  504.  508.
  509.  510.  511.  512.  513.  514.  515.  516.  518.  519.  520.  525.
  526.  528.  529.  530.  531.  532.  533.  534.  535.  536.  540.  542.
  543.  544.  546.  547.  548.  550.  552.  553.  554.  555.  558.  560.
  564.  568.  570.  571.  574.  575.  576.  584.  588.  589.  592.  594.
  596.  597.  598.  600.  602.  608.  609.  610.  612.  616.  620.  623.
  624.  625.  626.  628.  630.  631.  632.  635.  638.  640.  644.  645.
  647.  650.  657.  659.  660.  661.  662.  663.  666.  668.  672.  675.
  676.  677.  680.  681.  683.  686.  687.  688.  690.  692.  694.  696.
  698.  700.  701.  702.  704.  706.  711.  714.  715.  720.  725.  728.
  732.  733.  734.  736.  740.  742.  750.  753.  758.  760.  764.  767.
  768.  770.  772.  776.  781.  782.  783.  794.  798.  800.  801.  805.
  806.  808.  816.  819.  824.  825.  828.  832.  836.  839.  840.  843.
  848.  850.  854.  859.  862.  864.  870.  875.  883.  886.  888.  892.
  895.  900.  910.  924.  926.  929.  932.  938.  947.  962.  971.  973.
  976.  980.  982.  984.  992.  996. 1000. 1017. 1018. 1030. 1032. 1045.
 1071. 1077. 1107. 1120. 1147. 1158. 1176. 1185. 1186. 1208. 1257. 1262.
 1272. 1354. 1374. 1397. 1405. 1410. 1502. 1535. 1608. 1652. 1700. 1804.
 1911. 2571. 2678.   nan]
yardbuildingsqft26 [  18.   33.   34.   36.   37.   41.   48.   49.   54.   55.
56.   60.
   62.   72.   78.   88.   90.   96.  100.  104.  108.  110.  119.  120.
  125.  126.  133.  136.  144.  150.  156.  159.  160.  168.  180.  192.
  200.  204.  230.  231.  235.  240.  252.  264.  268.  276.  288.  290.
  308.  310.  322.  400.  408.  477.  480.  504.  525.  550.  627.  642.
  648.  846.  864.  943.  960. 1000. 1050. 1100. 1182. 1197. 1200. 1248.
 1366.   nan]
yearbuilt [1885. 1886. 1887. 1888. 1890. 1891. 1892. 1893. 1894. 1895. 1896.
1897.
 1898. 1899. 1900. 1901. 1902. 1903. 1904. 1905. 1906. 1907. 1908. 1909.
 1910. 1911. 1912. 1913. 1914. 1915. 1916. 1917. 1918. 1919. 1920. 1921.
 1922. 1923. 1924. 1925. 1926. 1927. 1928. 1929. 1930. 1931. 1932. 1933.
 1934. 1935. 1936. 1937. 1938. 1939. 1940. 1941. 1942. 1943. 1944. 1945.
 1946. 1947. 1948. 1949. 1950. 1951. 1952. 1953. 1954. 1955. 1956. 1957.
 1958. 1959. 1960. 1961. 1962. 1963. 1964. 1965. 1966. 1967. 1968. 1969.
 1970. 1971. 1972. 1973. 1974. 1975. 1976. 1977. 1978. 1979. 1980. 1981.
 1982. 1983. 1984. 1985. 1986. 1987. 1988. 1989. 1990. 1991. 1992. 1993.
 1994. 1995. 1996. 1997. 1998. 1999. 2000. 2001. 2002. 2003. 2004. 2005.
 2006. 2007. 2008. 2009. 2010. 2011. 2012. 2013. 2014. 2015.   nan]
numberofstories [ 1.  2.  3.  4. nan]
fireplaceflag ['True' 'nan']
structuretaxvaluedollarcnt [1.000000e+02 1.010000e+02 1.040000e+02 …
7.838271e+06 9.948100e+06
```

```
          nan]
taxvaluedollarcnt [2.200e+01 1.044e+03 1.928e+03 … 2.650e+07 2.775e+07
nan]
assessmentyear [2015.]
landtaxvaluedollarcnt [2.20000e+01 2.78000e+02 3.70000e+02 … 1.78019e+07
2.45000e+07
         nan]
taxamount [4.9080000e+01 5.1400000e+01 6.4000000e+01 … 3.1106407e+05
3.2193609e+05
          nan]
taxdelinquencyflag ['Y' 'nan']
taxdelinquencyyear [ 6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 99. nan]
censustractandblock [6.03710111e+13 6.03710111e+13 6.03710111e+13 …
6.11100910e+13
 6.11100910e+13            nan]
```

Abbiamo deciso di gestire subito la feature 'transactiondate' ricavandone altre di utili e che fossero utilizzabili successivamente per lavorare col 'parcelid'

[ ]:
```
# metodo con mese acquisto e tempo da età ad acquisto
def add_new_buyed_eta_features(df):
    # suddivide transactiondate in anno-mese-giorno
    df['transactiondate'] = pd.to_datetime(df.transactiondate,␣
 ↪format='%Y-%m-%d')
    # nuova feature mese acquisto
    df['month_buyed'] = df['transactiondate'].dt.month
    # nuova feature giorno acquisto
    df['day_buyed'] = df['transactiondate'].dt.day
    # nuova feature conteggio giorni da costruzione a vendita
    df['builded_buyed'] = (
        df['transactiondate'].dt.day
        + (df['transactiondate'].dt.month
        + ((df['transactiondate'].dt.year - df['yearbuilt'] ) * 12)) * 30)
    df = df.drop(['transactiondate','yearbuilt'], axis = 1)
    return df

df_train = add_new_buyed_eta_features(df_train)

df_train
```

[ ]:

	parcelid	logerror	airconditioningtypeid	architecturalstyletypeid	\
0	11016594	0.0276	1.0	NaN	
1	14366692	-0.1684	NaN	NaN	
2	12098116	-0.0040	1.0	NaN	
3	12643413	0.0218	1.0	NaN	
4	14432541	-0.0050	NaN	NaN	
…	…	…	…	…	
90270	10774160	-0.0356	1.0	NaN	

```
90271  12046695    0.0070                    NaN                         NaN
90272  12995401   -0.2679                    NaN                         NaN
90273  11402105    0.0602                    NaN                         NaN
90274  12566293    0.4207                    NaN                         NaN

       basementsqft  bathroomcnt  bedroomcnt  buildingclasstypeid  \
0               NaN          2.0         3.0                  NaN
1               NaN          3.5         4.0                  NaN
2               NaN          3.0         2.0                  NaN
3               NaN          2.0         2.0                  NaN
4               NaN          2.5         4.0                  NaN
…             …          …         …                  …
90270           NaN          1.0         1.0                  NaN
90271           NaN          3.0         3.0                  NaN
90272           NaN          2.0         4.0                  NaN
90273           NaN          2.0         2.0                  NaN
90274           NaN          1.0         3.0                  NaN

       buildingqualitytypeid  calculatedbathnbr  …  taxvaluedollarcnt  \
0                        4.0                2.0  …          360170.0
1                        NaN                3.5  …          585529.0
2                        4.0                3.0  …          119906.0
3                        4.0                2.0  …          244880.0
4                        NaN                2.5  …          434551.0
…                      …                  …  …               …
90270                    4.0                1.0  …          191000.0
90271                    4.0                3.0  …          161111.0
90272                    7.0                2.0  …           38096.0
90273                    4.0                2.0  …          165869.0
90274                    7.0                1.0  …          163037.0

       assessmentyear  landtaxvaluedollarcnt  taxamount  taxdelinquencyflag  \
0              2015.0               237416.0    6735.88                 NaN
1              2015.0               239071.0   10153.02                 NaN
2              2015.0                57912.0   11484.48                 NaN
3              2015.0                73362.0    3048.74                 NaN
4              2015.0               264977.0    5488.96                 NaN
…               …                     …        …                  …
90270          2015.0               147200.0    2495.24                 NaN
90271          2015.0                43218.0    1886.54                 NaN
90272          2015.0                16088.0    1925.70                   Y
90273          2015.0                32878.0    2285.57                 NaN
90274          2015.0                96779.0    2560.96                 NaN

       taxdelinquencyyear  censustractandblock  month_buyed  day_buyed  \
0                     NaN         6.037107e+13            1          1
1                     NaN                  NaN            1          1
```

```
2                    NaN         6.037464e+13            1           1
3                    NaN         6.037296e+13            1           2
4                    NaN         6.059042e+13            1           2
…                    …                    …            …           …
90270                NaN         6.037132e+13           12          30
90271                NaN         6.037301e+13           12          30
90272               14.0         6.037433e+13           12          30
90273                NaN         6.037601e+13           12          30
90274                NaN         6.037544e+13           12          30

       builded_buyed
0            20551.0
1              751.0
2            27391.0
3            10472.0
4            12632.0
…                …
90270        13710.0
90271        18750.0
90272        33510.0
90273        12990.0
90274        25230.0

[90275 rows x 61 columns]
```

Dopo l'aggiunta di nuove feature per la gestione della data di acquisto siamo andati a controllare la presenza di eventuali parcelid duplicati, ed abbiamo rimosso quelli con data di acquisto più vecchia (nel nostro caso con feature 'builded_buyed' più bassa)

[ ]:
```python
#conta quante volte compare uno stesso parcelid
def count_dup(df,feature):
    print("\nId duplicati : " + str(df[feature].duplicated().sum()))
    for id,count in df[feature].value_counts().iteritems():
        if count > 1:
            print('L\'id : ' + str(id) +' compare '+ str(count) +' volte.' )

count_dup(df_train, 'parcelid')

# eliminiamo le righe con il parcelid duplicato tenendo quella con l'ultima␣
 ↪vendita effettuata
df_train = df_train.sort_values("builded_buyed").
 ↪drop_duplicates("parcelid",keep="last")

print('\nElimino righe con Id duplicati...')

count_dup(df_train, 'parcelid')
```

```
Id duplicati : 125
L'id : 11842707 compare 3 volte.
L'id : 11633771 compare 2 volte.
L'id : 11742566 compare 2 volte.
L'id : 14322378 compare 2 volte.
L'id : 12097956 compare 2 volte.
L'id : 11554091 compare 2 volte.
L'id : 13037293 compare 2 volte.
L'id : 12448490 compare 2 volte.
L'id : 11735136 compare 2 volte.
L'id : 10961914 compare 2 volte.
L'id : 11188497 compare 2 volte.
L'id : 11913710 compare 2 volte.
L'id : 12715657 compare 2 volte.
L'id : 13850164 compare 2 volte.
L'id : 12678472 compare 2 volte.
L'id : 11866315 compare 2 volte.
L'id : 14667297 compare 2 volte.
L'id : 14607531 compare 2 volte.
L'id : 14621246 compare 2 volte.
L'id : 11477350 compare 2 volte.
L'id : 17151530 compare 2 volte.
L'id : 14500952 compare 2 volte.
L'id : 11729067 compare 2 volte.
L'id : 11822007 compare 2 volte.
L'id : 12457291 compare 2 volte.
L'id : 14753974 compare 2 volte.
L'id : 10796614 compare 2 volte.
L'id : 10815854 compare 2 volte.
L'id : 11561632 compare 2 volte.
L'id : 11121105 compare 2 volte.
L'id : 11845988 compare 2 volte.
L'id : 12032773 compare 2 volte.
L'id : 13966342 compare 2 volte.
L'id : 13074547 compare 2 volte.
L'id : 10790468 compare 2 volte.
L'id : 14683772 compare 2 volte.
L'id : 12978851 compare 2 volte.
L'id : 12081817 compare 2 volte.
L'id : 10911172 compare 2 volte.
L'id : 11230482 compare 2 volte.
L'id : 14086119 compare 2 volte.
L'id : 14141531 compare 2 volte.
L'id : 12042403 compare 2 volte.
L'id : 13858886 compare 2 volte.
L'id : 12465776 compare 2 volte.
L'id : 13969515 compare 2 volte.
```

L'id : 14530899 compare 2 volte.
L'id : 11183209 compare 2 volte.
L'id : 10799924 compare 2 volte.
L'id : 13003771 compare 2 volte.
L'id : 10858080 compare 2 volte.
L'id : 12689560 compare 2 volte.
L'id : 14367791 compare 2 volte.
L'id : 13923841 compare 2 volte.
L'id : 13092608 compare 2 volte.
L'id : 12276495 compare 2 volte.
L'id : 10736972 compare 2 volte.
L'id : 17164212 compare 2 volte.
L'id : 14485861 compare 2 volte.
L'id : 12613442 compare 2 volte.
L'id : 12512227 compare 2 volte.
L'id : 11526663 compare 2 volte.
L'id : 12764062 compare 2 volte.
L'id : 11419032 compare 2 volte.
L'id : 12545874 compare 2 volte.
L'id : 14175744 compare 2 volte.
L'id : 11135845 compare 2 volte.
L'id : 13041169 compare 2 volte.
L'id : 11514606 compare 2 volte.
L'id : 11061551 compare 2 volte.
L'id : 14294516 compare 2 volte.
L'id : 14079655 compare 2 volte.
L'id : 11226939 compare 2 volte.
L'id : 11863325 compare 2 volte.
L'id : 14316410 compare 2 volte.
L'id : 13927813 compare 2 volte.
L'id : 14022812 compare 2 volte.
L'id : 12206101 compare 2 volte.
L'id : 14678446 compare 2 volte.
L'id : 17237150 compare 2 volte.
L'id : 12304797 compare 2 volte.
L'id : 11146377 compare 2 volte.
L'id : 11639269 compare 2 volte.
L'id : 13991964 compare 2 volte.
L'id : 14613416 compare 2 volte.
L'id : 11005771 compare 2 volte.
L'id : 11122560 compare 2 volte.
L'id : 14677191 compare 2 volte.
L'id : 11663348 compare 2 volte.
L'id : 14490188 compare 2 volte.
L'id : 14057417 compare 2 volte.
L'id : 12921348 compare 2 volte.
L'id : 12518077 compare 2 volte.
L'id : 11503158 compare 2 volte.

```
L'id : 11121274 compare 2 volte.
L'id : 10821829 compare 2 volte.
L'id : 11656774 compare 2 volte.
L'id : 12023571 compare 2 volte.
L'id : 17128287 compare 2 volte.
L'id : 10798910 compare 2 volte.
L'id : 14306028 compare 2 volte.
L'id : 13859028 compare 2 volte.
L'id : 14672826 compare 2 volte.
L'id : 12596069 compare 2 volte.
L'id : 14615311 compare 2 volte.
L'id : 14376552 compare 2 volte.
L'id : 12584665 compare 2 volte.
L'id : 17290242 compare 2 volte.
L'id : 11747818 compare 2 volte.
L'id : 17148603 compare 2 volte.
L'id : 11513960 compare 2 volte.
L'id : 11484014 compare 2 volte.
L'id : 11602482 compare 2 volte.
L'id : 11555342 compare 2 volte.
L'id : 10818076 compare 2 volte.
L'id : 10883535 compare 2 volte.
L'id : 14368021 compare 2 volte.
L'id : 14326166 compare 2 volte.
L'id : 14336872 compare 2 volte.
L'id : 11105038 compare 2 volte.
L'id : 12779635 compare 2 volte.
L'id : 12551240 compare 2 volte.
L'id : 14444102 compare 2 volte.
L'id : 11476515 compare 2 volte.

Elimino righe con Id duplicati…

Id duplicati : 0
```

Abbiamo provato ad fare un clustering con KMeans eliminando dal dataset alcune feature come latitude, longitude ed idcounty, per vedere se inserendo un K=3 il risultato fosse simile a quello ottenibile inserendo direttamente l'idcounty, confermando che i valori delle case erano bne disposti per zona.

[ ]:
```
# prepariamo dataframe provvisorio per creare la nuova classificazione
df_pos = df_train.drop(['latitude','longitude','regionidcounty',
                        'regionidzip', 'regionidcity','fips',
                        'propertyzoningdesc', 'parcelid',␣
 ↪'regionidneighborhood',
                        'propertycountylandusecode', 'taxdelinquencyflag',
                        'propertylandusetypeid'], axis=1)
```

```
# riempiamo con la mediana i valori NaN
df_pos = df_pos.fillna(df_pos.median())

#applichiamo algoritmo di clustering KMeans
KM = KMeans(n_clusters=3).fit(df_pos)
labels = KM.labels_

#aggiungiamo nuova feature calcolata da KMeans al dataset
df_train['county'] = labels.tolist()

regionid = { 3101.0 : 'Los Angeles',
             2061.0 : 'Ventura',
             1286.0 : 'Orange',
}

fig = go.Figure()
fig = make_subplots(
    rows=1, cols=2, subplot_titles=(
        'plot county per lat x long',
        'plot clustering-county per lat x long'
    ),
    specs=[[{"type": "mapbox"}, {"type": "mapbox"}]],
)

fig.add_trace(
    go.Scattermapbox(
        lat = df_train["latitude"]/1e6, # coordinate y, viene diviso per 1e6␣
 ↪per convertire i valori in coordinate realistiche
        lon = df_train["longitude"]/1e6, # coorinate x
        text = 'Contea : ' + df_train['regionidcounty'].replace(regionid),
        marker=dict(
            size = 4,
            color = df_train['regionidcounty'],
        )
), 1, 1)

# secondo plot divisione fatta da kmeans
fig.add_trace(
    go.Scattermapbox(
        lat = df_train["latitude"]/1e6,
        lon = df_train["longitude"]/1e6,
        text = 'Contea : ' + (df_train['regionidcounty'].replace(regionid)
                              + '\n Cluster : ' + df_train['county'].
 ↪astype(str)),
        marker=dict(
            size = 4,
            color = df_train['county'],
```

```
        )
), 1, 2)

fig.update_mapboxes(
    style="open-street-map",
    center={
        "lat": df_train['latitude'].mean()/1e6, # coordinate centro mappa
        "lon": df_train['longitude'].mean()/1e6 # coordinate centro mappa
    },
    zoom = 7,
)
fig.update_layout(
    margin=dict(l=30, r=30, t=30, b=20),
)
fig.show()
```

Il risultato mostra come effettivamente le istanze sono ben disposte nel dataset a parte qualche eccezione. Abbiamo provato a stamparci un dataset contenente solo i punti in questione che veniva clusterizzati in maniera strana ma non abbiamo trovato particolari evidenze che potessero spiegare il fenomeno.

[ ]:
```
strange_county = df_train.loc[(df_train['regionidcounty'] == 2061) &␣
 ↪(df_train['county'] != 1)]
strange_county = strange_county.append(df_train.loc[(df_train['regionidcounty']␣
 ↪== 1286) & (df_train['county'] != 2)])
print(f'numero di istanze clusterizzate in maniera "anomala" : {strange_county.
 ↪shape[0]}')
strange_county.to_csv('strange_county.csv')
```

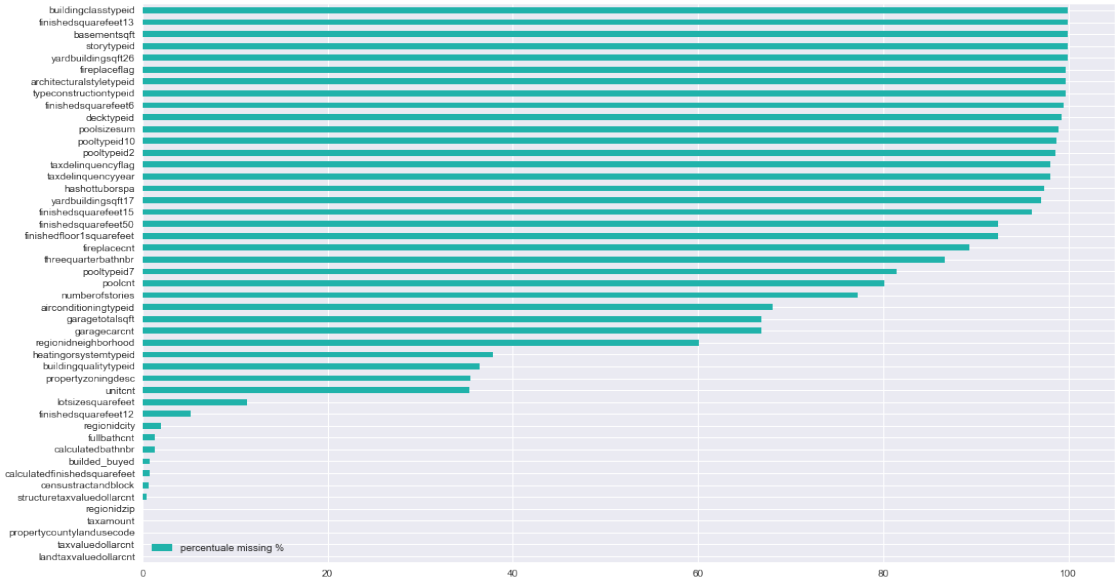
```
numero di istanze clusterizzate in maniera "anomala" : 253
```

1.1.2 Analisi e gestione dei missing values e feature engeneering

In questa fase siamo andati inizialmente a visualizzarci le percentuali di valori mancanti all'interno delle diverse feature del nostro dataset.

[ ]:
```
def missing_values_table(df):
        if df.isnull().sum().sum() != 0:
                na_df = (df.isnull().sum() / len(df)) * 100
                na_df = na_df.drop(na_df[na_df == 0].index).
 ↪sort_values(ascending=True)
                missing_data = pd.DataFrame({'percentuale missing %' :na_df})
                missing_data.plot(kind='barh', color='lightseagreen',␣
 ↪figsize=(17,10))
                plt.show()
                return missing_data
        else:
                print('Nessun NaN trovato')
```

```
missing_table = missing_values_table(df_train)
```



Abbiamo poi deciso di rimuovere le feature aventi una percentuale di valori mancanti superiore al 50%. Alcune sono state invece mantenute per le fasi successive nonostante il largo numero di valori mancanti.

[ ]:
```
# eliminiamo le feature con una percentuale di missing troppo elevata,␣
 ↪escludendo
# alcune che utilizzeremo comunque successivamente
def drop_feature(percentage, table, not_drop, df):
    drop = list()
    for f,p in table.iloc[:,0].iteritems() :
        if p > percentage : drop.append(f)
    # togliamo dalla lista le feature che non volgiamo eliminare
    for f in not_drop:
        drop.remove(f)
    # eliminiamo le feature selezionate
    print(drop)
    df.drop(drop,
            axis=1,
            inplace= True,
            errors='ignore')

# rimuovo alcune delle feature che nonostante la percentuale di missing values␣
 ↪saranno utili
# successivamente
```

```
useful_f = [
    'hashottuborspa', 'fireplacecnt',
    'threequarterbathnbr','poolcnt',
    'poolsizesum','airconditioningtypeid',
    'garagetotalsqft','garagecarcnt'
]

to_drop = drop_feature(50, missing_table, useful_f, df_train) # drop con␣
 ↪missing > 50%
```

```
['regionidneighborhood', 'numberofstories', 'pooltypeid7',
'finishedfloor1squarefeet', 'finishedsquarefeet50', 'finishedsquarefeet15',
'yardbuildingsqft17', 'taxdelinquencyyear', 'taxdelinquencyflag', 'pooltypeid2',
'pooltypeid10', 'decktypeid', 'finishedsquarefeet6', 'typeconstructiontypeid',
'architecturalstyletypeid', 'fireplaceflag', 'yardbuildingsqft26',
'storytypeid', 'basementsqft', 'finishedsquarefeet13', 'buildingclasstypeid']
```

Creazione "nuove" feature In questa fase siamo andati a creare alcune feature nuove basandoci su quelle preesistenti. Inoltre, abbiamo provato ad utilizzare le feature che si era deciso di non scartare dalla fase di rimozione per alta percentuale di valori mancanti.

1. recupero dei valori mancanti da altre feature

[ ]:
```
# se non è presente la dimensione della casa completata sostituiamo con quello␣
 ↪della
# casa completata
df_train['house_dim'] = df_train['finishedsquarefeet12'].fillna(
    df_train['calculatedfinishedsquarefeet'], inplace=False)

# se non è presente la censustractandblock della casa sostituiamo con quello
# della colonna rawcensustractandblock rendendola comparabile alle altre␣
 ↪presenti
df_train['censustractandblock'] = df_train['censustractandblock'].fillna(
    df_train['rawcensustractandblock'] * 1000000, inplace=False)
```

2. inserimento di features binarie che segnalano la presenza o meno di un determinato valore

[ ]:
```
# se il conteggio dei garage non era presente inserisce un 1, sarà utile per la
# nuova feature 'room_count'
df_train['missed_room'] = df_train['garagecarcnt'].isna().astype(int)

# era presente la dimensione della casa oppure è stato recuperato con quella␣
 ↪prevista
df_train['pre_house_dim'] = df_train['finishedsquarefeet12'].isna().astype(int)
```

3. Applicazione del Label Encoder per la gestione delle due feature di tipo Object (categoriali)

```
[ ]: # label encoder per sostituire alle due colonne il valore numerico associato␣
      ↪alle precedenti stringhe
     df_train[['propertyzoningdesc', 'propertycountylandusecode']] = (
         df_train[['propertyzoningdesc', 'propertycountylandusecode']
                  ].apply(LabelEncoder().fit_transform))
```

4. Creazione nuove feature utilizzando quelle preesistenti

```
[ ]: # beni di 'lusso' presenti
     df_train['luxury_goods'] = (
         (df_train['hashottuborspa'] == True).astype(int) * 5
         + df_train['poolcnt'].fillna(0)
         * (np.where(df_train['poolsizesum'].fillna(0, inplace=True) == 0, 1,
                     df_train['poolsizesum']/df_train['poolsizesum'].mean()))
         + (df_train['airconditioningtypeid'].fillna(0) > 0).astype(int) * 0.5
         + df_train['fireplacecnt'].fillna(0) * 0.5)

     # stima del valore della casa sulla base delle tasse sulla struttura
     df_train['ValueProperty'] = (df_train['structuretaxvaluedollarcnt']
                                  / df_train['landtaxvaluedollarcnt'])

     # media valore proprietà in base al valore stimato della proprietà
     df_train['avg_dimxzip'] = df_train['ValueProperty'].groupby(
         df_train['regionidzip']).transform('mean')

     # media costo spesa per zona da suddivisione del clustering
     df_train['avg_taxzip'] = df_train['taxamount'].groupby(
         df_train['regionidzip']).transform('mean')

     # media logerror acquisti per giorno di acquisto
     df_train['avg_day'] = df_train['logerror'].groupby(
         df_train['day_buyed']).transform('mean')

     # media logerror acquisti per mese di acquisto
     df_train['avg_month'] = df_train['logerror'].groupby(
         df_train['month_buyed']).transform('mean')

     # numero di stanze approssimativo assegnando un peso maggiore in presenza di␣
      ↪bagni completi e garage di dimensione maggiore
     df_train['room_count'] = (
         (df_train['bathroomcnt'].fillna(df_train['bathroomcnt'].median())
         - df_train['fullbathcnt'].fillna(0)
         - df_train['threequarterbathnbr'].fillna(0))
         + df_train['fullbathcnt'].fillna(0) * 1.5
         + df_train['threequarterbathnbr'].fillna(0) * 1.25
         + df_train['garagecarcnt'].fillna(0) * (
         np.where(df_train['garagetotalsqft'].fillna(0, inplace=True) == 0, 1,
```

```
        df_train['garagetotalsqft']/df_train['garagetotalsqft'].mean()))
        + df_train['bedroomcnt'].fillna(
            df_train['bedroomcnt'].median())
        + (np.where((df_train['roomcnt']
                     - (df_train['bathroomcnt']
                        + df_train['bedroomcnt'])) > 0,df_train['roomcnt']
                    - (df_train['bathroomcnt'] + df_train['bedroomcnt']), 0)))
```

5. Rimozione feature che non avranno più un utilità per allenare i modelli

```
[ ]: #eliminiamo alcune feature che sono state già utilizzate in precedenza per␣
      ↪creare altre features
    to_remove = [
        'fullbathcnt', 'calculatedbathnbr', 'garagetotalsqft',
        'hashottuborspa', 'poolcnt', 'poolsizesum','roomcnt',
        'threequarterbathnbr','fireplacecnt','airconditioningtypeid',
        'bathroomcnt', 'garagecarcnt', 'bedroomcnt',
        'calculatedfinishedsquarefeet','county',
        'finishedsquarefeet12','rawcensustractandblock'
    ]

    df_train = df_train.drop(to_remove,axis=1, errors = 'ignore')
```

Gestione dei missing values Abbiamo deciso poi di sostituire tutti i missing values rimanenti rimpiazzandoli con la media per quanto riguarda le features numeriche, mentre utilizzando la mediana per quanto riguarda le feature categoriche

```
[ ]: # controlliamo le numerical e categorical feature, abbiamo deciso di impostare␣
      ↪un ipotetico 35 come valore per distinguerle
    # visto l'ingente numero di valori che assumono le righe, inoltre usiamo una␣
      ↪regex per cercare alcune feature che essendo
    # comunque categoriche presentano troppi valori distinti per essere inseriti␣
      ↪solo col conteggio dei valori unici
    def count_catnum_feat(df) :
        cat = np.array( [ bool(
            re.search(r'[i][d]|[p][r]', col) or len(np.unique(df[col]))<=35 ) for␣
      ↪col in df ] )
        cat_idx = np.flatnonzero(cat)
        num_idx = np.flatnonzero(cat == False)
        print ('numercial features :\n',df.columns[num_idx])
        print ('categorical features :\n',df.columns[cat_idx])
        print ('Numero di numerical features:', sum(cat == False))
        print ("Numero di categorical features:", sum(cat))
        return cat_idx, num_idx

    categorical, numerical = count_catnum_feat(df_train)
```

```
df_train.to_csv('prova.csv')

# trasformazione delle numercial a floats
for col in df_train.columns[numerical]:
    df_train[col].apply(pd.to_numeric, errors='coerce')
```

```
numercial features :
 Index(['logerror', 'latitude', 'longitude', 'lotsizesquarefeet',
       'structuretaxvaluedollarcnt', 'taxvaluedollarcnt',
       'landtaxvaluedollarcnt', 'taxamount', 'censustractandblock',
       'builded_buyed', 'house_dim', 'luxury_goods', 'ValueProperty',
       'avg_dimxzip', 'avg_taxzip', 'room_count'],
      dtype='object')
categorical features :
 Index(['parcelid', 'buildingqualitytypeid', 'fips', 'heatingorsystemtypeid',
       'propertycountylandusecode', 'propertylandusetypeid',
       'propertyzoningdesc', 'regionidcity', 'regionidcounty', 'regionidzip',
       'unitcnt', 'assessmentyear', 'month_buyed', 'day_buyed', 'missed_room',
       'pre_house_dim', 'avg_day', 'avg_month'],
      dtype='object')
Numero di numerical features: 16
Numero di categorical features: 18
```

[ ]:
```
#rempiamo i missung values con media e mediana
def replace_missing(num,cat,df):
    for col in df.columns[num]:
        df[col].fillna(df[col].mean(axis=0), inplace=True)
    for col in df.columns[cat]:
        df[col].fillna(df[col].median(axis=0), inplace=True)
    print(missing_values_table(df))

replace_missing(numerical, categorical, df_train)
```

```
Nessun NaN trovato
None
```

1.1.3 Test dell'importanza delle feature e preparazione finale del dataset

In questa fase siamo andati a capire quali delle feature rimaste all'interno del dataset risultassero utili per allenare poi i modelli. Per verificarlo siamo andati ad utilizzare l'algoritmo RFECV passandogli come modello un RandomForestRegressor, che in automatico ci dirà quali feature rimuovere e quali invece sono significative e vanno preservate.

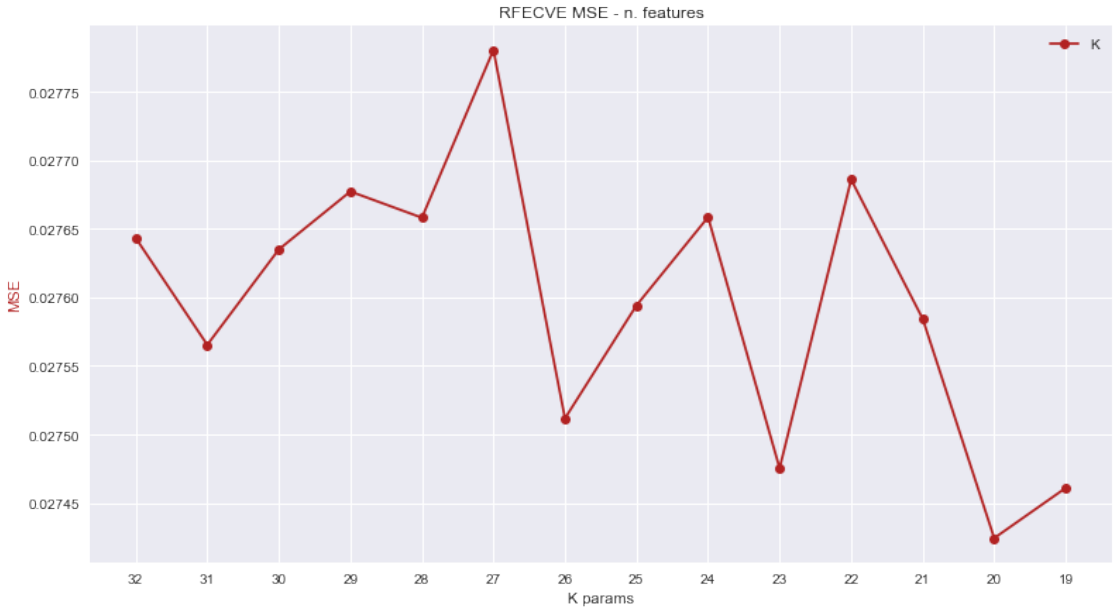
[ ]:
```
# feature importance usando RFECV e passandogli una random forest come modello
def feature_imp_RFECV(df):
    x = df.copy()
    y = x['logerror'].values
    x = x.drop(['parcelid', 'logerror'],axis=1)
```

```
    # assegniamo modello random forest con n alberi
    rf = RandomForestRegressor(n_estimators=100)
    selector = RFECV(rf,
                     step=1, # feature da rimuovere ad ogni passo
                     cv=5, # cross validation
                     scoring='neg_mean_squared_error',
                     min_features_to_select=19, # feature min da inserire
                     n_jobs = -1 # lavoro parallelo con tutti i core
                 )
    selector.fit(x,y)

    # nuovo dataframe in base ai risultati di RFECV
    df_imp = x.loc[:, selector.support_]
    df_imp = df_imp.join(df[['logerror','parcelid']])

    return selector,df_imp

Rfecv,df_train = feature_imp_RFECV(df_train)
```

[ ]:
```
def plot_MSE_param(x,y):

    fig, ax1 = plt.subplots(figsize=(13,7))

    ax1.plot(x, y, '-o', label='mean squared error', color = 'firebrick')
    ax1.set_xlabel('K params')
    ax1.set_ylabel('MSE', color='firebrick')
    ax1.invert_xaxis()
    ax1.xaxis.set_major_locator(MaxNLocator(integer=True))

    plt.xticks(x)
    plt.legend('K')
    plt.grid(True)
    plt.title('RFECVE MSE - n. features')
    fig.show()

test_scores_MSE = abs(Rfecv.cv_results_['mean_test_score'])
test_scores_VAR = pow(Rfecv.cv_results_['std_test_score'], 2)

# stampiamo il plot dell'andamento per feature rimossa
plot_MSE_param(range(
    Rfecv.min_features_to_select,
    Rfecv.n_features_in_ + Rfecv.step, Rfecv.step), test_scores_MSE
)

print('Ottenuto il miglior MSE {} con {} features'.format(
    abs(Rfecv.cv_results_['mean_test_score'][
```

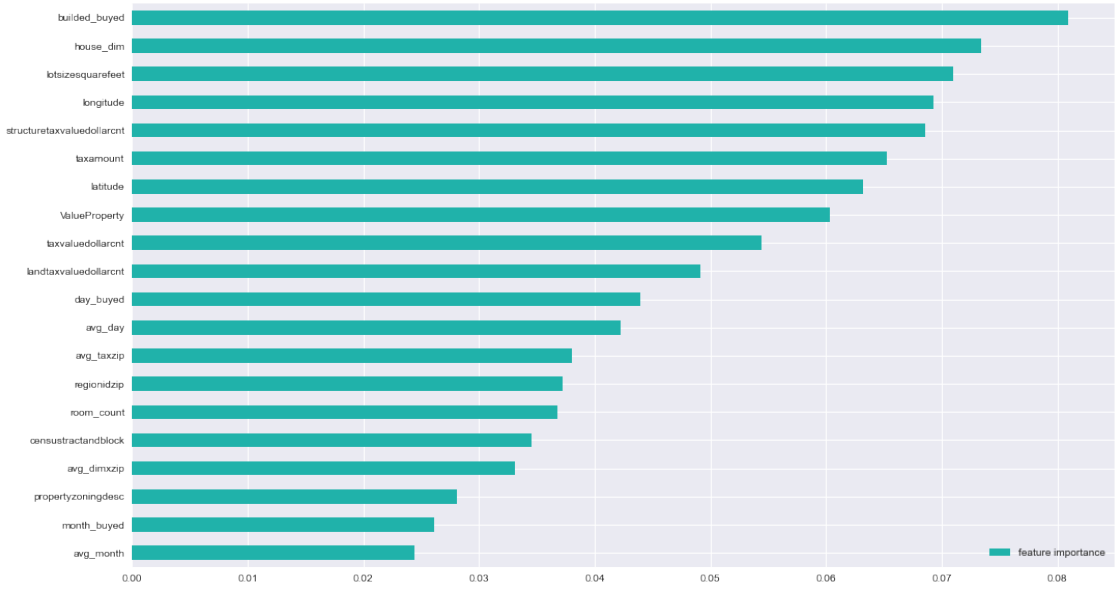
```
        int((Rfecv.n_features_ - Rfecv.min_features_to_select)
        / Rfecv.step)]),Rfecv.n_features_))
```

```
Ottenuto il miglior MSE 0.027424458426010895 con 20 features
```



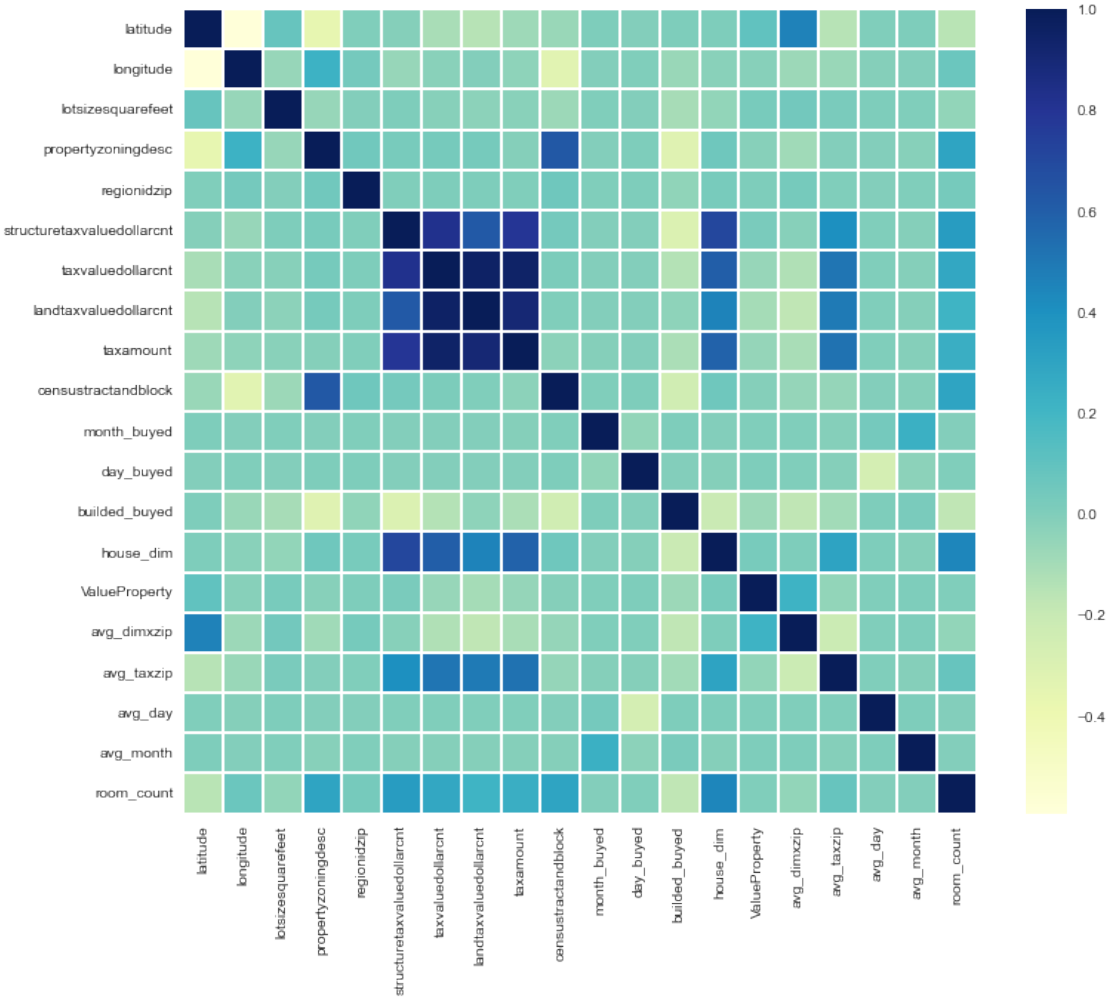
Per una miglior compresione abbiamo anche stampato il grafico della significatività di ogni feature presente ora nel dataset.

[ ]:
```
def plot_feature_imp(df):
    x = df.copy()
    y = x['logerror'].values
    x = x.drop(['parcelid', 'logerror'],axis=1)

    rf = RandomForestRegressor(n_estimators=100).fit(x,y)

    f_imp = pd.DataFrame(rf.feature_importances_,
                         list(x),
                         columns=['feature importance'])
    f_imp.sort_values('feature importance', ascending=True, inplace=True)

    f_imp.plot(kind='barh', color='lightseagreen', figsize=(17,10))

# stampiamo plot delle features importance del dataframe usando random forest
plot_feature_imp(df_train)
```



[ ]:
```
f, ax = plt.subplots(figsize =(12, 10))
corr_df = df_train.drop(['parcelid','logerror'], axis=1)

sns.heatmap(corr_df.corr(), ax = ax, cmap ="YlGnBu",
            linewidths = 0.1, yticklabels = corr_df.columns.values,
                              xticklabels = corr_df.columns.values)
```

[ ]: <AxesSubplot:>



1.2 Traning dei modelli

Una volta sistemati i dati siamo passati alla fase di allenamento dei modelli di predizione

Divisione in Train e Test dei dati Inizialmente abbiamo diviso il dataset in due parti, la Y composta della sola variabile risposta, ossia il 'logerror', e la X composta invece dei diversi predittori o feature. A sua volta poi abbiamo separato i set in Train e Test, così da poter allenare i modelli su di una parte ed effettuare poi la predizione finale sulla seconda.

[ ]:
```
def train_test(df):

    #mettiamo la featrue logerror come y e facciamo il drop da X
    y = df.logerror.values
    x = df.copy()
    x = (x.drop(columns=["parcelid","logerror"]))
```

```
    # usiamo MinMaxScaler per riscalare i valori delle variabili
    scale = MinMaxScaler()
    x[x.columns] = scale.fit_transform(x)

    #split in train e test
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20,␣
 ↪random_state=42)

    return x_train, x_test, y_train, y_test

x_train, x_test, y_train, y_test = train_test(df_train)
```

[ ]:

	latitude	longitude	lotsizesquarefeet	propertyzoningdesc	\
42787	0.512606	0.448991	0.000473	0.427856	
9019	0.447822	0.862079	0.013670	0.516032	
29870	0.357834	0.806381	0.004154	1.000000	
89047	0.575835	0.535063	0.001138	0.305110	
21492	0.638201	0.563654	0.000758	0.305110	
…	…	…	…	…	
51588	0.869561	0.663855	0.000935	0.715932	
42296	0.234255	0.824335	0.000837	1.000000	
3354	0.362746	0.800535	0.000795	1.000000	
16847	0.244969	0.905983	0.004154	1.000000	
55208	0.481952	0.570469	0.001195	0.307615	

	regionidzip	structuretaxvaluedollarcnt	taxvaluedollarcnt	\
42787	0.000474	0.021326	0.025495	
9019	0.001722	0.021904	0.015242	
29870	0.003484	0.029642	0.014774	
89047	0.001409	0.013988	0.007808	
21492	0.001271	0.016974	0.009215	
…	…	…	…	
51588	0.004435	0.026226	0.013477	
42296	0.003214	0.009348	0.025655	
3354	0.003484	0.005273	0.014590	
16847	0.003194	0.023124	0.019150	
55208	0.000112	0.017829	0.016614	

	landtaxvaluedollarcnt	taxamount	censustractandblock	month_buyed	\
42787	0.020213	0.029335	0.009458	0.454545	
9019	0.008366	0.015531	0.004089	0.090909	
29870	0.004694	0.014919	0.296452	0.272727	
89047	0.003160	0.008588	0.000365	1.000000	
21492	0.003541	0.010170	0.000041	0.181818	
…	…	…	…	…	
51588	0.004611	0.017281	0.010947	0.454545	
42296	0.025258	0.022579	0.297164	0.454545	

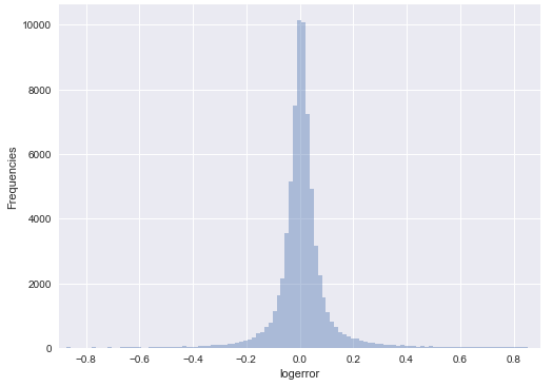
```
3354                0.014381   0.016640             0.296451     0.000000
16847               0.012297   0.026541             0.297009     0.181818
55208               0.011575   0.017269             0.001564     0.545455

       day_buyed  builded_buyed  house_dim  ValueProperty  avg_dimxzip  \
42787   0.166667       0.685009   0.041251       0.000435     0.013993
9019    0.333333       0.215278   0.056335       0.001080     0.048444
29870   0.900000       0.071369   0.079907       0.002604     0.016270
89047   0.266667       0.566337   0.095739       0.001826     0.026490
21492   1.000000       0.231663   0.084788       0.001977     0.056744
…            …              …          …              …            …
51588   0.966667       0.088009   0.161749       0.002345     0.121352
42296   0.066667       0.416832   0.105633       0.000153     0.019047
3354    0.600000       0.536546   0.052553       0.000151     0.016270
16847   0.500000       0.032174   0.067021       0.000776     0.019725
55208   0.400000       0.187871   0.063899       0.000635     0.016005

       avg_taxzip   avg_day  avg_month  room_count
42787    0.297950  0.622368   0.039397    0.001606
9019     0.130293  0.000000   0.772992    0.004176
29870    0.077916  0.543499   0.000000    0.003212
89047    0.143760  0.490739   1.000000    0.005140
21492    0.078279  0.520223   0.272601    0.004818
…             …         …          …           …
51588    0.097107  0.505219   0.039397    0.007067
42296    0.133720  0.538855   0.039397    0.010081
3354     0.077916  0.297843   0.769319    0.005552
16847    0.262391  0.598703   0.272601    0.004337
55208    0.245517  0.586925   0.402991    0.004176

[72120 rows x 20 columns]
```

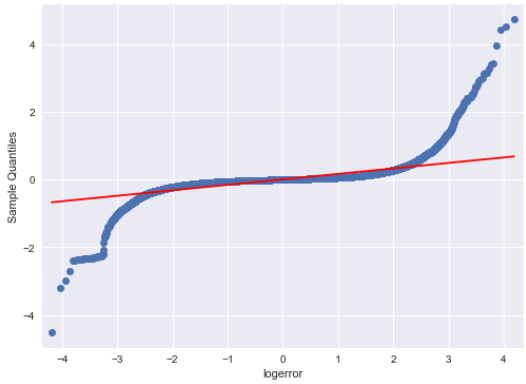
Prima di effettuare i test con i modelli abbiamo pensato di provare a vedere la distribuzione del 'logerror' e rimuovere eventualmente alcuni degli outliers presenti nel traning set.

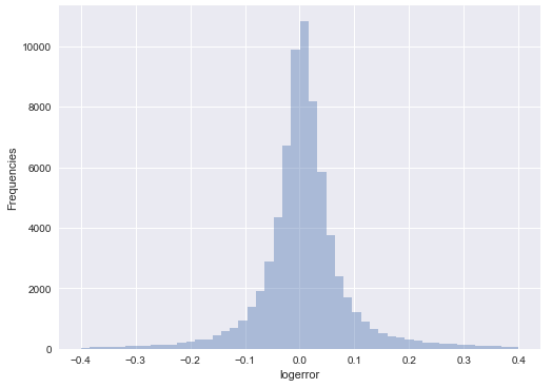
[ ]:
```
def remove_outliers(x_train,y_train):

    y = x_train.copy()
    y['logerror'] = y_train

    # definizione grafico distribuzione e qqplot
    fig, (ax1, ax2) = plt.subplots(1,2, figsize=(18,6))
    sns.distplot(y.logerror, bins=550, kde=False, ax = ax1)
    ax1.set_xlim(-0.9, 0.9)
    ax1.set(xlabel='logerror', ylabel='Frequencies')

    sm.qqplot(y.logerror, line ='s', ax = ax2, )
```

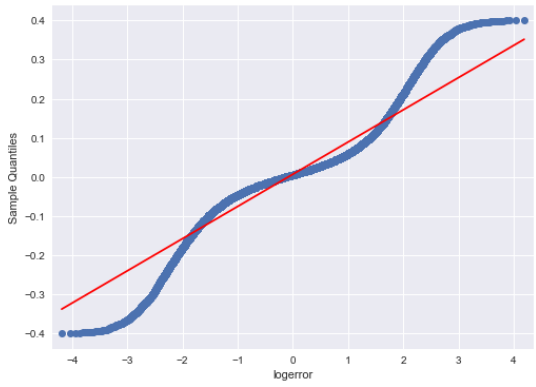
```
    ax2.set(xlabel='logerror')

    fig.show()

    print('Sono presenti {} istanze del logerror < -0.4 e {} istanze > 0.4'.
↪format(
        y[ y.logerror < -0.4 ].logerror.count(),
        y[ y.logerror > 0.4  ].logerror.count()))

    # rimozione outliers
    y = y[ y.logerror > -0.4 ]
    y = y[ y.logerror < 0.4 ]

    # plot della distirbuzione del logerror dopo rimozione aoutliers
    fig, (ax1, ax2) = plt.subplots(1,2, figsize=(18,6))
    sns.distplot(y.logerror, bins=50, kde=False, ax = ax1)
    ax1.set(xlabel='logerror', ylabel='Frequencies')

    sm.qqplot(y.logerror, line ='s', ax = ax2)
    ax2.set(xlabel='logerror')

    fig.show()

    x = y.drop('logerror',axis=1)

    return y.logerror, x

y_train, x_train = remove_outliers(x_train,y_train)
```

```
Sono presenti 545 istanze del logerror < -0.4 e 949 istanze > 0.4
```









Una volta rimossi gli outliers da y_train, quindi la porzione che sarà utilizzata per allenare i modelli, possiamo andare a costruire i nostri modelli.

1.2.1 Parameter tuning ed allenamento dei modelli

Abbiamo deciso di utilizzare la funzione GridSearchCV per effettuare il tuning dei parametri, così da capire prima sul validation set quali parametri fossero i migliori da applicare ai modelli prima di effettuare la vera predizione sui nuovi dati, ossia sul Test set. Come modelli da testare abbiamo scelto: Linear regression, Knn, Random Forest ed SVM.

[ ]:
```
model_params = {
    'AdaBoost':{
        'model': AdaBoostRegressor(),
        'params': {
            'n_estimators':  range(100,600,100),
            'learning_rate': [.001, .01]
        }
    },
    'Random forest':{
        'model': RandomForestRegressor(),
        'params': {
            'n_estimators':  range(300,700,100),
            'max_depth': [40, 60],
        }
    },
    'Linear regression':{
        'model': LinearRegression(),
        'params': {
        }
    },
    'Knn': {
        'model': KNeighborsRegressor(),
        'params': {
            'n_neighbors': range(100, 1000, 100),
```

```
        }
    },
    'SVR':{
        'model': SVR(),
        'params': {
            'C': np.arange(.01, .1, .02)
        }
    }
}

def grid(model_params):
    scores=[]
    models = [];
    for model_name, mp in model_params.items():
        clf = GridSearchCV(mp['model'], mp['params'], cv=5,␣
 ↪return_train_score=False, scoring='neg_mean_squared_error', n_jobs=-1)
        clf.fit(x_train, y_train)
        scores.append({
            'model': model_name,
            'best Mean Squared Error (MSE)': abs(clf.
 ↪cv_results_['mean_test_score'].mean()),
            'best Variance': abs(mean(clf.cv_results_['std_test_score'] ** 2)),
            'mean exec time': clf.cv_results_['mean_score_time'].mean(),
            'best params': clf.best_params_
        })
        models.append(clf)
    Train_scores = pd.DataFrame(
    scores,
    index=list(model_params.keys()),
    columns=['best Mean Squared Error (MSE)','best Variance','mean exec␣
 ↪time','best params']
    )
    Train_scores.index.name = "models"
    Train_scores.columns.name = 'Train scores'
    return models,Train_scores

clf,Train_scores =  grid(model_params)

Train_scores.style.highlight_min(['best Mean Squared Error (MSE)','best␣
 ↪Variance','mean exec time'],color='teal')
```

1.2.2 Utilizzo modelli per predizioni sul test set

Ottenuti i risultati con i diversi parametri sul validation set, abbiamo preso e passato quelli migliori ai vari modelli per effettuare le predizioni sul Test set, illustrando le differenze tra i risultati ottenuti nella precedente fase.

Inizialmente abbiamo anche definito due funzioni per stamparci i risultati ottenuti con la GridSearch

al variare dei parametri sul training set

[ ]:
```
# funzione per stampare il grafico dei risultati ottenuti sul train test con␣
 ↪singolo parametro
# passato alla GridSearch
def single_plot(Cv_results, model_name) :

    params = Cv_results['params']
    params_str = list(params[0].keys())[0]
    time = Cv_results['mean_score_time']
    MSE = abs(Cv_results['mean_test_score'])
    VAR = Cv_results['std_test_score']**2
    val = []

    fig, (ax1, ax3) = plt.subplots(1,2, figsize=(23,7))

    f = lambda p,ls : [ls.append(list(x.values())) for x in p]

    f(params, val)
    ax1.plot(val, MSE, '-o', label='mean squared error', color = 'firebrick')
    ax1.set_xlabel(params_str, fontsize=13)
    ax1.set_ylabel('MSE', color='firebrick', fontsize=13)
    ax1.tick_params(axis='y')
    plt.title('GridSearchCV time - ' + params_str, fontsize=15)

    ax2 = ax1.twinx()

    ax2.plot(val, VAR, '-o', label='mean squred error', color = 'cadetblue')
    ax2.set_ylabel('Variance', color='cadetblue', fontsize=13)
    ax2.tick_params(axis='y')

    ax3.plot(val, time, '-o', label='mean execution time', color =␣
 ↪'darkmagenta')
    ax3.set_ylabel('mean execution time', color='purple', fontsize=13)
    ax3.set_xlabel(params_str, fontsize=13)
    ax3.tick_params(axis='y')
    plt.title('GridSearchCV MSE/Var - ' + params_str, fontsize=15)

    fig.suptitle(model_name, fontsize=30)
    plt.grid(False)
    plt.show()
```

[ ]:
```
# funzione per stampare il grafico dei risultati ottenuti sul train test con␣
 ↪doppio parametro
# passato alla GridSearch
def double_plot(Cv_results, Cv_results_name) :
```

```
    params = Cv_results['params']
    time = Cv_results['mean_score_time']
    MSE = abs(Cv_results['mean_test_score'])
    VAR = Cv_results['std_test_score']**2
    l = len(params)
    legend_str = list(params[0].keys())[0]
    param_str = list(params[1].keys())[1]
    par = []
    leg = []

    f = lambda p,ls,e : [ls.append(x[list(x.keys())[e]]) for x in p]

    fig, (ax1, ax3) = plt.subplots(1,2, figsize=(23,7))

    f(params,par,1)
    f(params,leg,0)
    ax1.plot(par[0:int(l/2)], MSE[0:int(l/2)], '-o', label='mean squared␣
↪error', color = '#AA5042')
    ax1.plot(par[int(l/2):l], MSE[int(l/2):l], '-o', label='mean squred error',␣
↪color = 'salmon')
    ax1.set_xlabel(param_str, fontsize=13)
    ax1.set_ylabel('MSE', color='firebrick', fontsize=13)
    ax1.tick_params(axis='y')
    plt.title('GridSearchCV time - ' + param_str, fontsize=15)

    ax2 = ax1.twinx()

    ax2.plot(par[0:int(l/2)], VAR[0:int(l/2)], '-o', label='variance', color =␣
↪'#8FD4E3')
    ax2.plot(par[int(l/2):l], VAR[int(l/2):l], '-o', label='variance', color =␣
↪'#8FA1E3')
    ax2.set_ylabel('Variance', color='cadetblue', fontsize=13)
    ax2.tick_params(axis='y')

    ax2.legend(np.unique(leg), title=legend_str, loc = 1, frameon=True,␣
↪fontsize=11).get_frame().set_facecolor('white')
    ax1.legend(np.unique(leg), title=legend_str, loc = 2, frameon=True,␣
↪fontsize=11).get_frame().set_facecolor('white')

    ax3.plot(par[0:int(l/2)], time[0:int(l/2)], '-o', label='mean execution␣
↪time', color = 'darkmagenta')
    ax3.plot(par[int(l/2):l], time[int(l/2):l], '-o', label='mean execution␣
↪time', color = 'plum')
    ax3.set_ylabel('mean execution time', color='purple', fontsize=13)
    ax3.set_xlabel(param_str, fontsize=13)
    ax3.tick_params(axis='y')
```

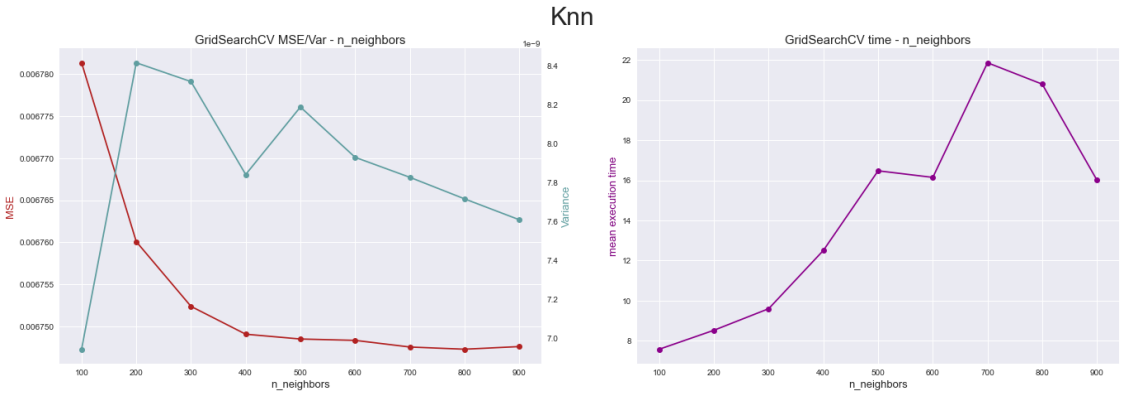
```
    ax3.legend(np.unique(leg), title=legend_str, loc = 0, frameon=True,␣
 ↪fontsize=11).get_frame().set_facecolor('white')
    plt.title('GridSearchCV MSE/Var - ' + param_str, fontsize=15)

    fig.suptitle(Cv_results_name, fontsize=20)
    plt.grid(False)
    plt.show()
```

Per ogni modello abbiamo deciso di stampare i risultati ottenuti sul test set utilizzando i migliori parametri, ed i grafici dei risultati della GridSearch

[ ]:
```
def best_model_test(Train_scores, Test_scores, model, x_test, y_test):
    test_res  = mean_squared_error(
        y_true=y_test,
        y_pred=model.predict(x_test)
    )
    Test_scores.loc[Train_scores.name] = {
        'Mean Squared Error (MSE) Validation' : Train_scores['Best MSE'],
        'Mean Squared Error (MSE) Test' : test_res,
        'mean exec time' : Train_scores['mean exec time'],
        'best params' : Train_scores['best params']
    }

Test_scores = pd.DataFrame(
    index=model_params.keys(),
    columns=['Mean Squared Error (MSE) Validation',
             'Mean Squared Error (MSE) Test',
             'mean exec time', 'best params']
)
Test_scores.columns.name = 'Test scores'
Train_scores.index.name = 'models'


model_name = Train_scores.loc['Linear regression'].name
model = LinearRegression(**Train_scores.loc[model_name, 'best params']).
 ↪fit(x_train,y_train)

best_model_test(Train_scores.loc[model_name],Test_scores, model, x_test,y_test)

Test_scores.loc[[model_name]]
```

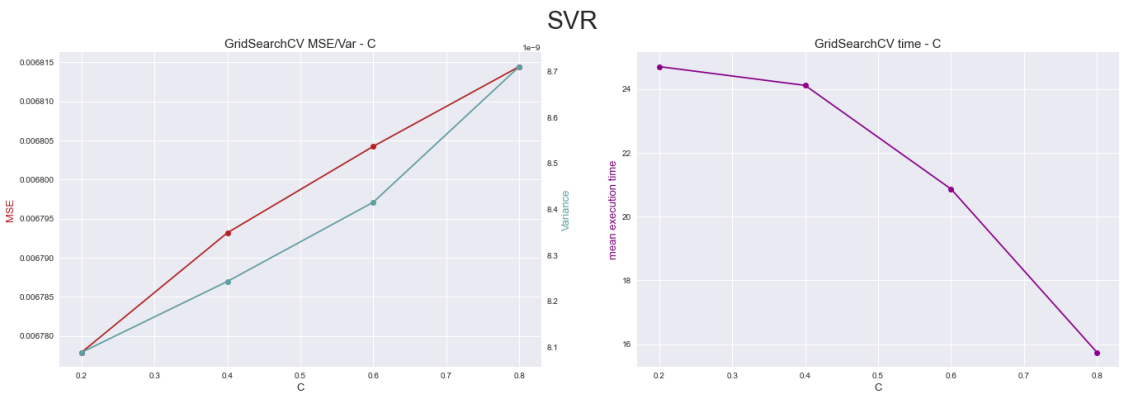
[ ]:
```
Test scores       Mean Squared Error (MSE) Validation  \
Linear regression                                  NaN

Test scores       Mean Squared Error (MSE) Test mean exec time best params
Linear regression                      0.024572       0.003571          {}
```

[ ]:
```
model_name = Train_scores.loc['Knn'].name
model = KNeighborsRegressor(**Train_scores.loc[model_name, 'best params']).
  ↪fit(x_train,y_train)

best_model_test(Train_scores.loc[model_name],Test_scores, model, x_test,y_test)

single_plot(clf[3].cv_results_, model_name)

Test_scores.loc[[model_name]]
```



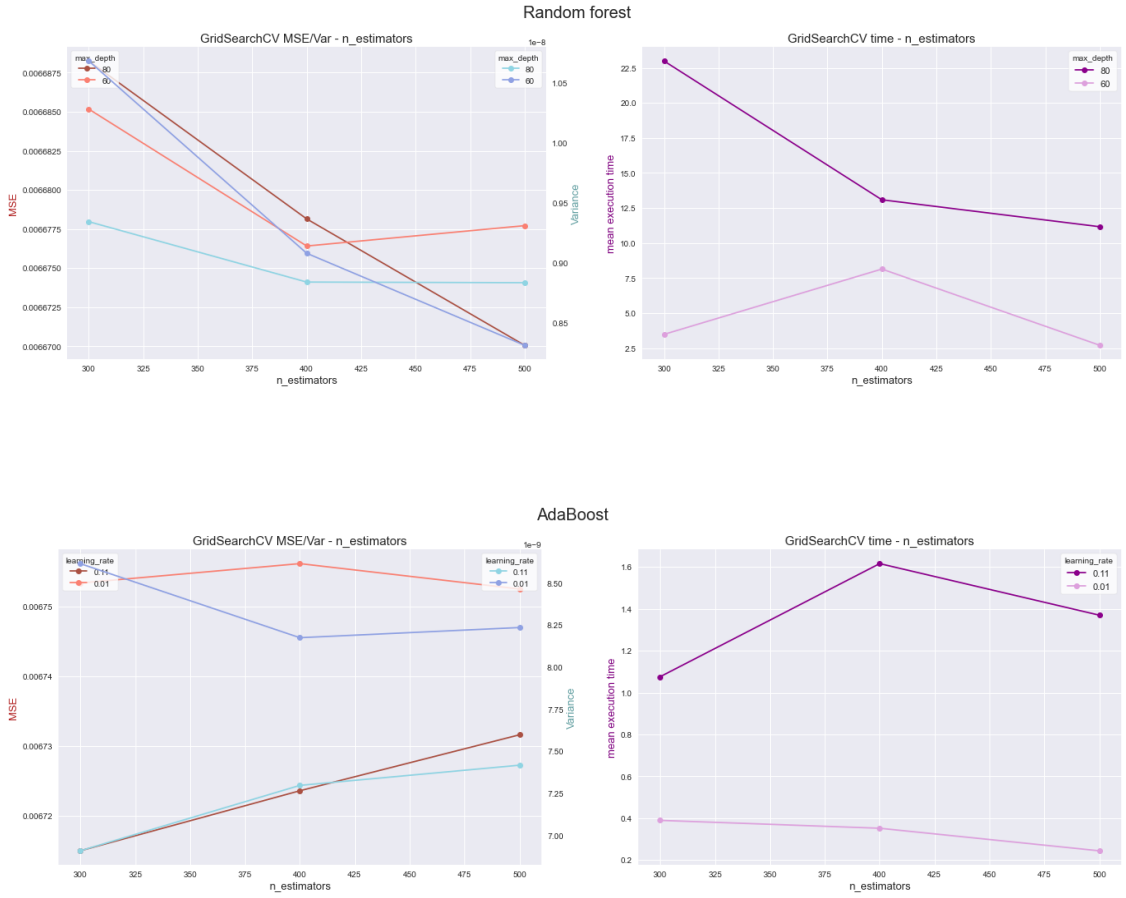
[ ]:
```
Test scores Mean Squared Error (MSE) Validation Mean Squared Error (MSE) Test  \
Knn                                                 NaN                       0.024687

Test scores mean exec time           best params
Knn                14.387109  {'n_neighbors': 800}
```

[ ]:
```
model_name = Train_scores.loc['SVR'].name
model = SVR(**Train_scores.loc[model_name, 'best params']).fit(x_train,y_train)

best_model_test(Train_scores.loc[model_name],Test_scores, model, x_test,y_test)

single_plot(clf[4].cv_results_, model_name)

Test_scores.loc[[model_name]]
```



[ ]:
```
Test scores Mean Squared Error (MSE) Validation Mean Squared Error (MSE) Test  \
SVR                                                NaN                       0.02468

Test scores mean exec time best params
SVR                21.35248  {'C': 0.2}
```

[ ]:
```
model_name1 = Train_scores.loc['Random forest'].name
model = RandomForestRegressor(**Train_scores.loc[model_name1, 'best params']).
 ↪fit(x_train,y_train)

best_model_test(Train_scores.loc[model_name1],Test_scores, model, x_test,y_test)

model_name2 = Train_scores.loc['AdaBoost'].name
model = AdaBoostRegressor(**Train_scores.loc[model_name2, 'best params']).
 ↪fit(x_train,y_train)

best_model_test(Train_scores.loc[model_name2],Test_scores, model, x_test,y_test)

double_plot(clf[1].cv_results_,'Random forest')
double_plot(clf[0].cv_results_,'AdaBoost')

comp = pd.concat([Test_scores.loc[[model_name1]], Test_scores.
 ↪loc[[model_name2]]])
comp.style.highlight_min([
    'Mean Squared Error (MSE) Validation',
    'Mean Squared Error (MSE) Test',
    'mean exec time'], color='teal')
```



[ ]: <pandas.io.formats.style.Styler at 0x22d87c7f910>

[ ]:
```
def prova(df):
    df = df.style.highlight_min([
        'Mean Squared Error (MSE) Validation',
        'Mean Squared Error (MSE) Test',
        'mean exec time'], color='teal')
    display(df)

print('\nRisultati finali : \n')
prova(Test_scores)
```

```
Risultati finali :
```

<pandas.io.formats.style.Styler at 0x22d87437bb0>