```
import time
st = time.time()
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn as skl
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import KNNImputer, SimpleImputer
from sklearn.model_selection import train_test_split
```

```
df = pd.read_csv('/content/drive/MyDrive/most_subscribed_youtube_channels.csv', index_col=
df
```

rank	Youtuber	subscribers	video views	video count	cate
1	T-Series	222,000,000	198,459,090,822	17,317	M
2	YouTube Movies	154,000,000	0	0	Film & Anim
3	Cocomelon - Nursery Rhymes	140,000,000	135,481,339,848	786	Educ
4	SET India	139,000,000	125,764,252,686	91,271	S
5	Music	116,000,000	0	0	
...	...	...	...	...	
996	JP Plays	10,900,000	4,609,300,218	3,528	Ga
997	TrapMusicHDTV	10,900,000	4,070,521,973	690	M
998	Games EduUu	10,900,000	3,093,784,767	1,006	Ga
999	Hueva	10,900,000	3,040,301,750	831	Ga
1000	Dobre Brothers	10,900,000	2,808,411,693	590	People &

1000 rows × 6 columns

```
df = df.drop(['Youtuber'],axis=1)
df
```

rank	subscribers	video views	video count	category	started
1	222,000,000	198,459,090,822	17,317	Music	2006
2	154,000,000	0	0	Film & Animation	2015
3	140,000,000	135,481,339,848	786	Education	2006
4	139,000,000	125,764,252,686	91,271	Shows	2006
5	116,000,000	0	0	NaN	2013
...	...	...	...	...	...
996	10,900,000	4,609,300,218	3,528	Gaming	2014

```
df.isnull().sum()
```

```
subscribers     0
video views     0
video count     0
category       27
started         0
dtype: int64
```

```
encoder = LabelEncoder()
df['category'] = encoder.fit_transform(df[['category']])
df
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/preprocessing/_label.py:115: DataConv
  y = column_or_1d(y, warn=True)
```

rank	subscribers	video views	video count	category	started
1	222,000,000	198,459,090,822	17,317	8	2006
2	154,000,000	0	0	4	2015
3	140,000,000	135,481,339,848	786	2	2006
4	139,000,000	125,764,252,686	91,271	14	2006
5	116,000,000	0	0	18	2013
...	...	...	...	...	...
996	10,900,000	4,609,300,218	3,528	5	2014
997	10,900,000	4,070,521,973	690	8	2013
998	10,900,000	3,093,784,767	1,006	5	2011
999	10,900,000	3,040,301,750	831	5	2012
1000	10,900,000	2,808,411,693	590	11	2017

1000 rows × 5 columns

```
imputer = SimpleImputer(strategy='mean')
df['category'] = imputer.fit_transform(df[['category']])
df
```

rank	subscribers	video views	video count	category	started
1	222,000,000	198,459,090,822	17,317	8.0	2006
2	154,000,000	0	0	4.0	2015
3	140,000,000	135,481,339,848	786	2.0	2006
4	139,000,000	125,764,252,686	91,271	14.0	2006
5	116,000,000	0	0	18.0	2013
...	...	...	...	...	...
996	10,900,000	4,609,300,218	3,528	5.0	2014
997	10,900,000	4,070,521,973	690	8.0	2013
998	10,900,000	3,093,784,767	1,006	5.0	2011
999	10,900,000	3,040,301,750	831	5.0	2012
1000	10,900,000	2,808,411,693	590	11.0	2017

1000 rows × 5 columns

```
cols_to_change = ['subscribers','video views','video count']
for col in cols_to_change:
  df[col] = df[col].str.replace(',', '')
```

```
df
```

rank	subscribers	video views	video count	category	started
1	222000000	198459090822	17317	8.0	2006
4	139000000	125764252686	91271	14.0	2006

```
# for col in cols_to_change:
#   df[col] = df[col].str.strip('.')
```

```
df
```

rank	subscribers	video views	video count	category	started
1	222000000	198459090822	17317	8.0	2006
2	154000000	0	0	4.0	2015
3	140000000	135481339848	786	2.0	2006
4	139000000	125764252686	91271	14.0	2006
5	116000000	0	0	18.0	2013
...	...	...	...	...	...
996	10900000	4609300218	3528	5.0	2014
997	10900000	4070521973	690	8.0	2013
998	10900000	3093784767	1006	5.0	2011
999	10900000	3040301750	831	5.0	2012
1000	10900000	2808411693	590	11.0	2017

1000 rows × 5 columns

```
df.isnull().sum()
```

```
subscribers    0
video views    0
video count    0
category       0
started        0
dtype: int64
```

```
column_name = df.columns
column_name
```

```
Index(['subscribers', 'video views', 'video count', 'category', 'started'],
dtype='object')
```

```
print(df.dtypes)
```

```
subscribers     object
video views     object
video count     object
category       float64
started          int64
dtype: object
```

```
df['subscribers'] = df['subscribers'].astype(str).astype(float)
df['video views'] = df['video views'].astype(str).astype(int)
df['video count'] = df['video count'].astype(str).astype(int)
```

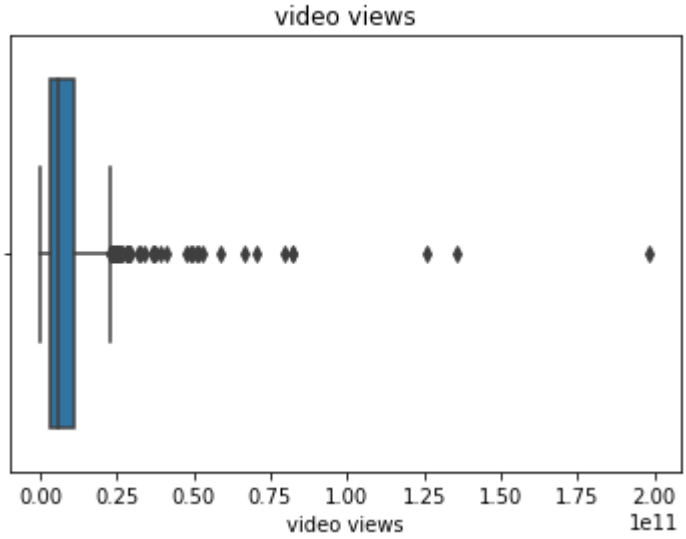
```
print(df.dtypes)
```

```
subscribers    float64
video views      int64
video count      int64
category       float64
started          int64
dtype: object
```

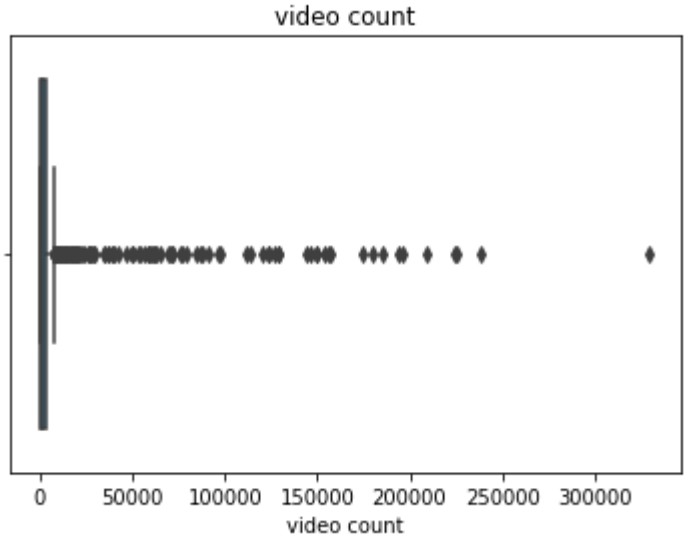
```
print(np.dtype(df['video count']))
```

```
int64
```

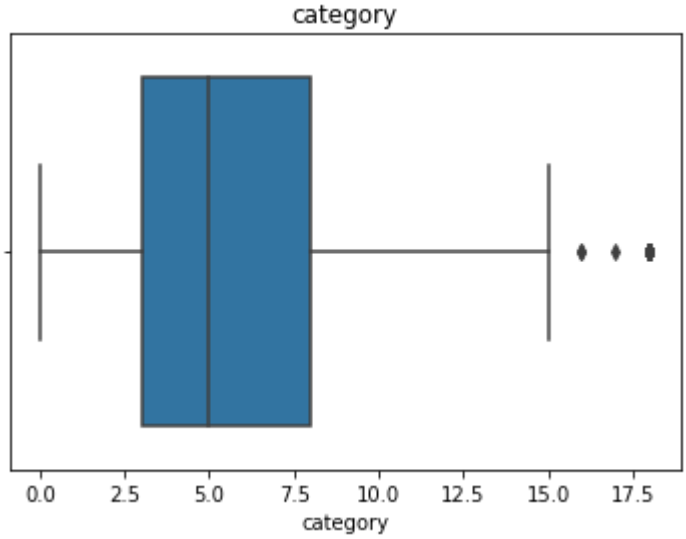
```
for boxcol in column_name:
  box = sns.boxplot(df[boxcol])
  plt.title(boxcol)
  plt.show(box)
  # fig, ax = plt.subplots(figsize = (18,10))
  # plt.title(boxcol)
  # ax.scatter(df[boxcol], df['video views'])
  # plt.show()
```



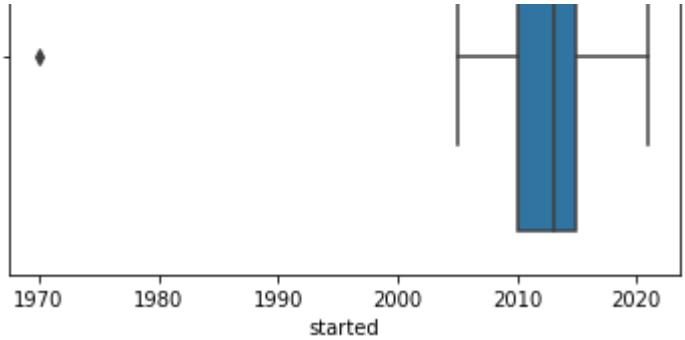

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
  FutureWarning
```




```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
  FutureWarning
```




```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
  FutureWarning
```




```
categoryoutl = np.where(df['category']>15)
startedoutl = np.where(df['started']<2000)
suboutl = np.where(df['subscribers']>0.5)
vwoutl = np.where(df['video views']>0.5)
print('Category')
print(categoryoutl)
print('Started')
print(startedoutl)
print('Subscribers')
print(suboutl)
print('Video Views')
print(vwoutl)
```

```
240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252,
253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265,
266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278,
279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291,
292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304,
305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317,
318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330,
332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344,
345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357,
358, 359, 360, 361, 362, 363, 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, 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,
749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761,
762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774,
775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787,
788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800,
801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813,
814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826,
827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839,
840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852,
853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865,
866, 867, 868, 869, 871, 872, 873, 874, 875, 876, 877, 878, 879,
880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892,
893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905,
906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918,
919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931,
932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944,
945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957,
958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970,
971, 972, 973, 974, 975, 976, 978, 979, 980, 981, 982, 983, 984,
985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997,
998, 999]),)
```

```
df.shape
```

```
(1000, 5)
```

```
for f in categoryoutl:
  df.drop(f, inplace=True)
for g in startedoutl:
  df.drop(g, inplace=True)
for j in suboutl:
  try:
    df.drop(j, inplace=True)
  except:
    pass
for k in vwoutl:
  try:
    df.drop(k, inplace=True)
  except:
    pass
```

```
df.shape
```

```
(969, 5)
```

```
y = df['video views']
y = pd.DataFrame(y)
y
```

rank	video views
1	198459090822
2	0
3	135481339848
5	0
6	28469458228
...	...
996	4609300218
997	4070521973
998	3093784767
999	3040301750

```
x = df.drop(['video views'], axis=1)
x
```

rank	subscribers	video count	category	started
1	222000000.0	17317	8.0	2006
2	154000000.0	0	4.0	2015
3	140000000.0	786	2.0	2006
5	116000000.0	0	18.0	2013
6	111000000.0	4497	5.0	2010
...	...	...	...	...
996	10900000.0	3528	5.0	2014
997	10900000.0	690	8.0	2013
998	10900000.0	1006	5.0	2011
999	10900000.0	831	5.0	2012
1000	10900000.0	590	11.0	2017

969 rows × 4 columns

```
x_tr, x_ts, y_tr, y_ts = train_test_split(x, y, train_size=.8, shuffle=True)
```

```
from sklearn.svm import SVR,LinearSVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, VotingRegressor, BaggingRegressor, Gra
```

```
from sklearn.metrics import r2_score
from sklearn.preprocessing import StandardScaler ,RobustScaler
from sklearn.linear_model import LinearRegression

x_tr.shape, x_ts.shape, y_tr.shape, y_ts.shape
```

```
((775, 4), (194, 4), (775, 1), (194, 1))
```

```
def my_modelfit(my_model,my_x_train,my_y_train):
  my_model.fit(my_x_train, my_y_train.values.ravel())
def my_predict(my_model,my_x_test):
  y_pred = my_model.predict(my_x_test)
  return y_pred
def my_r2_score(my_y_test,my_y_pred):
  r2 = r2_score(my_y_test, my_y_pred)
  return r2


def cr_scaler(my_scaler,my_x_train,my_x_test):
  my_scaler.fit_transform(my_x_train)
  my_scaler.fit_transform(my_x_test)


clf1 = SVR(kernel= 'linear',C=25)
clf2 = KNeighborsRegressor(n_neighbors=25)
clf3 = DecisionTreeRegressor()
clf4 = RandomForestRegressor()
clf5 = SVR(kernel= 'rbf',C=25)
clf5 = LinearSVR()
clf6 = LinearRegression()
clf7 = VotingRegressor(
                       estimators=[('knr', clf2), ('rfr', clf3), ('svr2', clf4), ('lsvr',
                      )
clf8 = BaggingRegressor(clf3, n_estimators=10, max_samples=.8, n_jobs=-1) #For all cpu use
clf9 = GradientBoostingRegressor()
Sscaler = StandardScaler()
Rscaler = RobustScaler()


cr_scaler(Sscaler, x_tr, x_ts)

my_modelfit(clf2, x_tr, y_tr)
y_pred_res2 = my_predict(clf2,x_ts)
print(my_r2_score(y_ts,y_pred_res2))

my_modelfit(clf3, x_tr, y_tr)
y_pred_res3 = my_predict(clf3,x_ts)
print(my_r2_score(y_ts,y_pred_res3))

my_modelfit(clf4, x_tr, y_tr)
y_pred_res4 = my_predict(clf4,x_ts)
print(my_r2_score(y_ts,y_pred_res4))

my_modelfit(clf5, x_tr, y_tr)
y_pred_res5 = my_predict(clf5,x_ts)
```

```
print(my_r2_score(y_ts,y_pred_res5))

my_modelfit(clf6, x_tr, y_tr)
y_pred_res6 = my_predict(clf6,x_ts)
print(my_r2_score(y_ts,y_pred_res6))

my_modelfit(clf7, x_tr, y_tr)
y_pred_res7 = my_predict(clf7,x_ts)
print(my_r2_score(y_ts,y_pred_res7))

my_modelfit(clf8, x_tr, y_tr)
y_pred_res8 = my_predict(clf8,x_ts)
print(my_r2_score(y_ts,y_pred_res8))

my_modelfit(clf9, x_tr, y_tr)
y_pred_res9 = my_predict(clf9,x_ts)
print(my_r2_score(y_ts,y_pred_res9))
```

```
0.29983962893168603
0.7869952522225225
0.6291883810751792
0.18643703968158876
0.6570039205040821
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning
  ConvergenceWarning,
0.5799135644991298
0.5826941083973319
0.6152831631347144
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning
  ConvergenceWarning,
```

```
cr_scaler(Rscaler, x_tr, x_ts)

my_modelfit(clf2, x_tr, y_tr)
y_pred_res2 = my_predict(clf2,x_ts)
print(my_r2_score(y_ts,y_pred_res2))

my_modelfit(clf3, x_tr, y_tr)
y_pred_res3 = my_predict(clf3,x_ts)
print(my_r2_score(y_ts,y_pred_res3))

my_modelfit(clf4, x_tr, y_tr)
y_pred_res4 = my_predict(clf4,x_ts)
print(my_r2_score(y_ts,y_pred_res4))

my_modelfit(clf5, x_tr, y_tr)
y_pred_res5 = my_predict(clf5,x_ts)
print(my_r2_score(y_ts,y_pred_res5))

my_modelfit(clf6, x_tr, y_tr)
y_pred_res6 = my_predict(clf6,x_ts)
print(my_r2_score(y_ts,y_pred_res6))

my_modelfit(clf7, x_tr, y_tr)
```

```
y_pred_res7 = my_predict(clf7,x_ts)
print(my_r2_score(y_ts,y_pred_res7))

my_modelfit(clf8, x_tr, y_tr)
y_pred_res8 = my_predict(clf8,x_ts)
print(my_r2_score(y_ts,y_pred_res8))

my_modelfit(clf9, x_tr, y_tr)
y_pred_res9 = my_predict(clf9,x_ts)
print(my_r2_score(y_ts,y_pred_res9))
```

```
0.29983962893168603
0.7858212691890757
0.5961335960439986
-0.1732071620050024
0.6570039205040821
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning
  ConvergenceWarning,
0.6610220485474642
0.5937072837125926
0.5504085993521658
/usr/local/lib/python3.7/dist-packages/sklearn/svm/_base.py:1208: ConvergenceWarning
  ConvergenceWarning,
```

```
end = time.time()
print((end-st)/60)
```

```
0.053175270557403564
```

✓ 0 sn. tamamlanma zamanı: 01:11