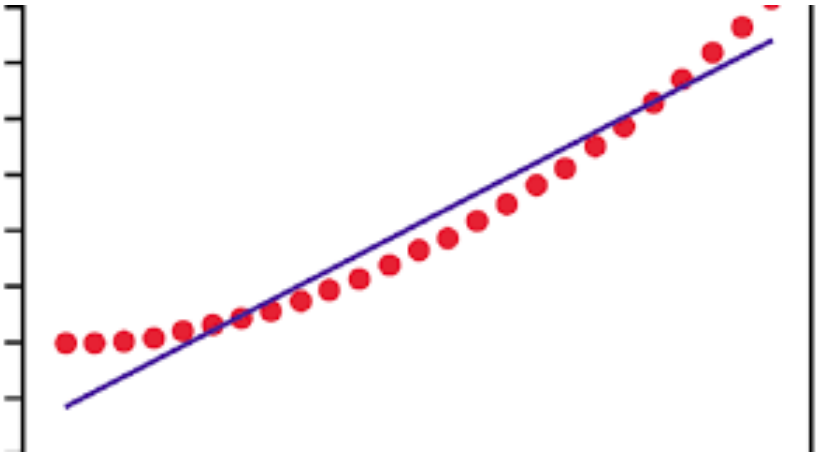
PROJECT 4
REPORT

Regression



Fabrice Harel-Canada (705221880)
Ganesha Srivallabha Durbha (405291327)
Boyuan He (004791432)

2021.02.26
21W-ECENGR219-1

Q1

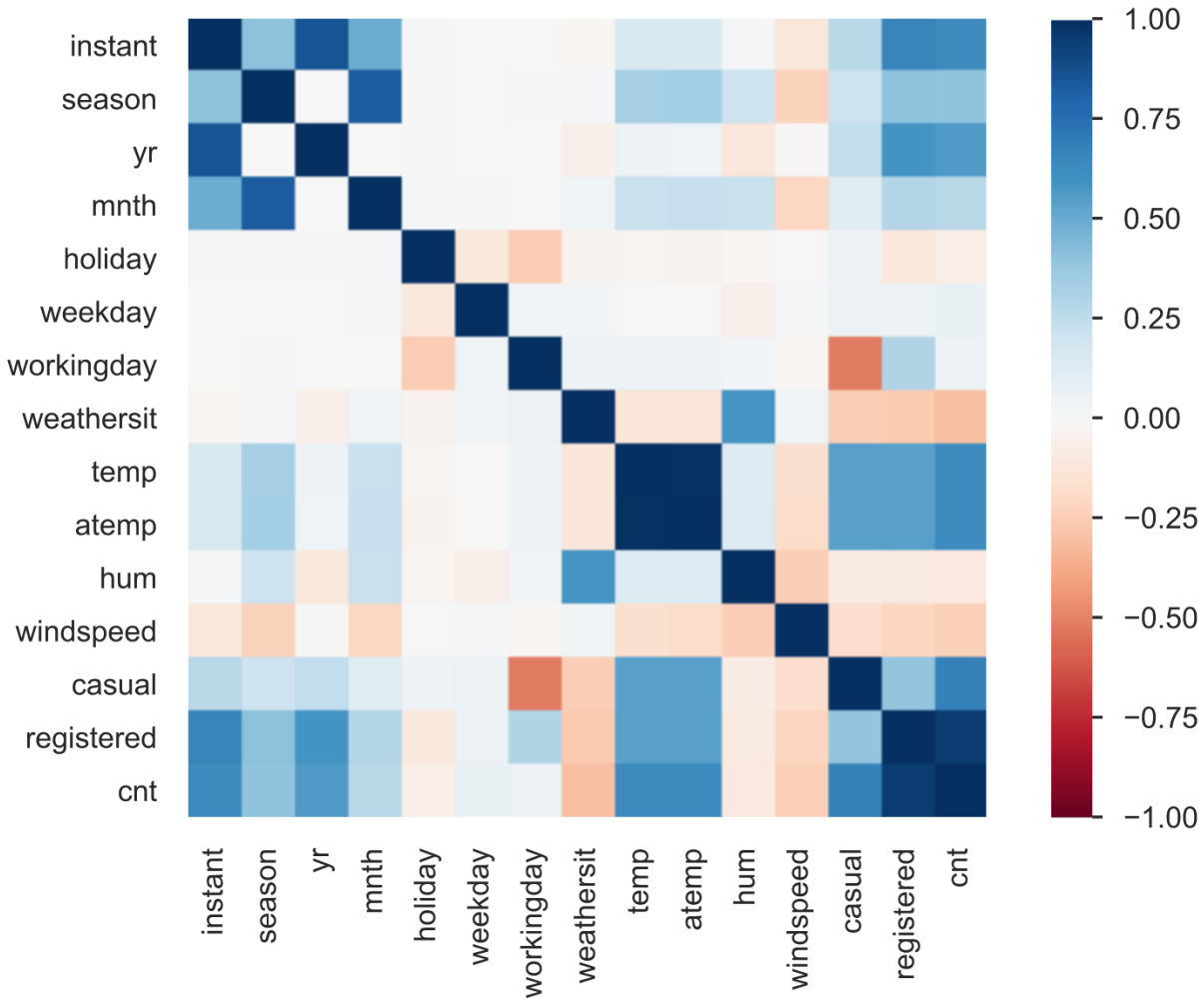
Plot a heatmap of Pearson correlation matrix of dataset columns. Report which features have the highest absolute correlation with the target variable and what that implies.

ANSWER

Bike Sharing

atemp has the highest correlation with each of the target variables. This implies that the hotter the temperature feels, the more bike share rentals are made. However, since the correlation is only mid-range, it suggests that getting too hot may suppress rentals.

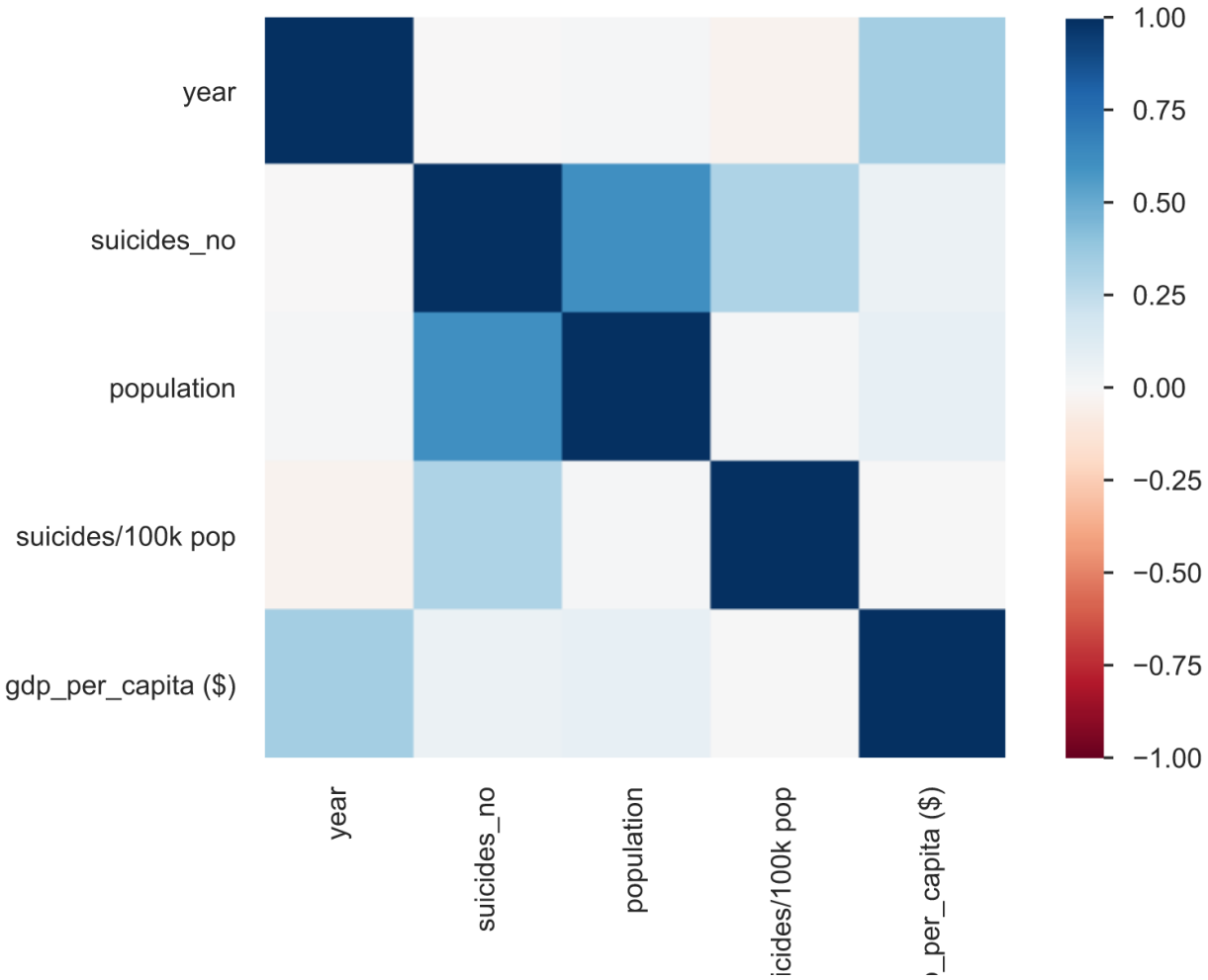
	casual	registered	cnt
season	0.2104	0.4116	0.4061
yr	0.2485	0.5942	0.5667
mnth	0.1230	0.2935	0.2800
holiday	0.0543	-0.1087	-0.0683
weekday	0.0599	0.0574	0.0674
workingday	-0.5180	0.3039	0.0612
weathersit	-0.2474	-0.2604	-0.2974
temp	0.5433	0.5400	0.6275
atemp	**0.5439**	**0.5442**	**0.6311**
hum	-0.0770	-0.0911	-0.1007
windspeed	-0.1676	-0.2174	-0.2345


instant
season
yr
mnth
holiday
weekday
workingday
weathersit
temp
atemp
hum
windspeed
casual
registered
cnt
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Suicide Rate

population has the highest correlation for **suicides_no** and **year** has the highest (negative) correlation with **suicides/100k pop**. Perhaps unsurprisingly, the larger the population, the larger the total count of people who can / do commit suicide, although this correlation is virtually non-existent when considering the normalized figure per 100k. Then for **year** we have a small negative correlation, suggesting that, if anything fewer people are committing suicide (per 100k) as time goes on.

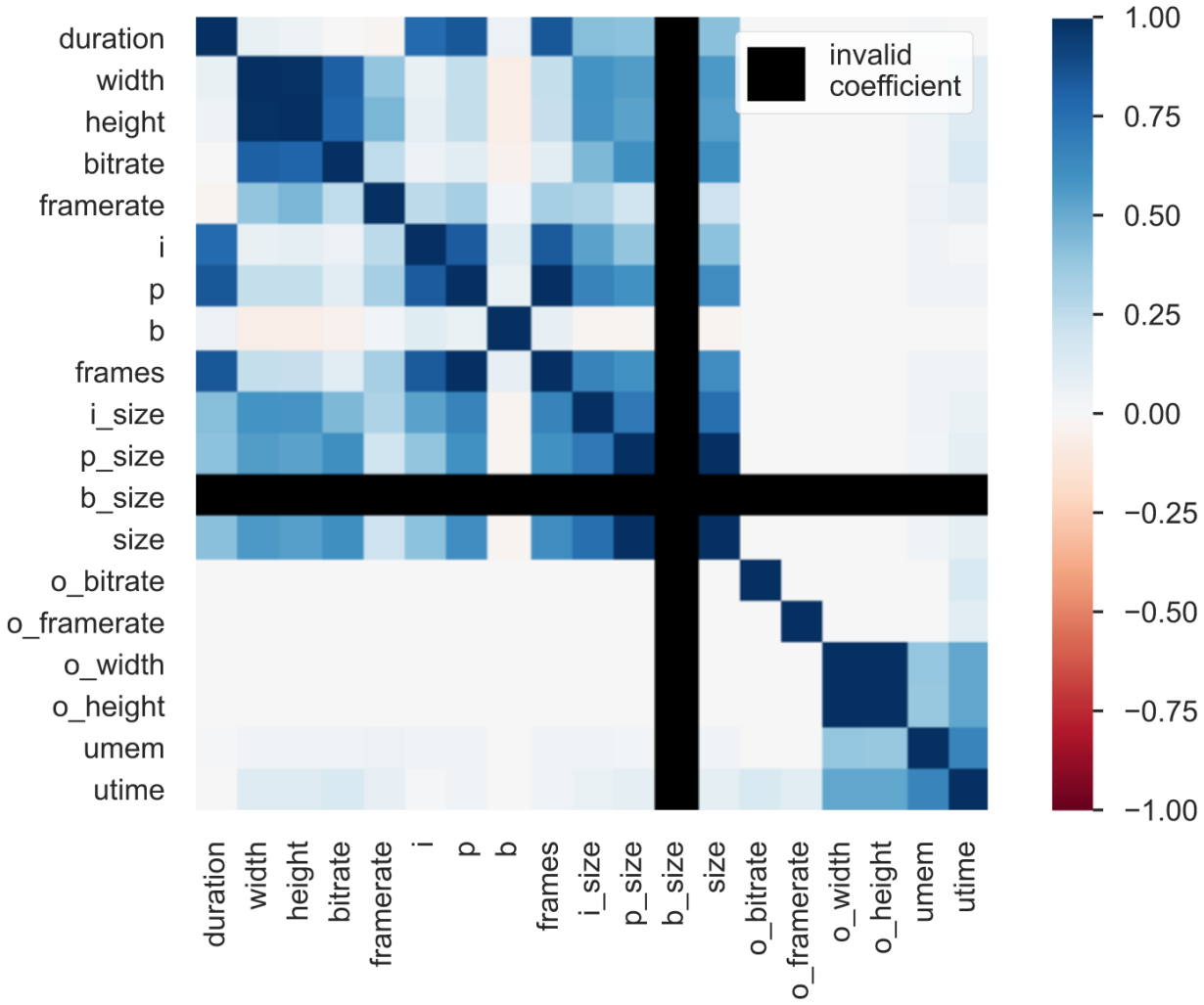
	suicides_no	suicides/100k pop
year	-0.0045	**-0.0390**
population	**0.6162**	0.0083
gdp_for_year ($)	0.4301	0.0252
gdp_per_capita ($)	0.0613	0.0018



Video Transcoding Time

umem is correlated most strongly with **utime**, however, it is unclear if this is supposed to be a feature or target vector. Excluding that, the features with the highest correlation are **o_width** and **o_height**. This suggests that the larger the output pixels, the more time it will take to transcode the video.

	utime
duration	0.0055
width	0.1299
height	0.1285
bitrate	0.1552
framerate	0.0793
i	0.0185
p	0.0332
b	0.0051
frames	0.0331
i_size	0.0647
p_size	0.0976
b_size	
size	0.0971
o_bitrate	0.1555
o_framerate	0.1040
o_width	0.5234
o_height	0.5196
umem	**0.6633**


duration
width
height
bitrate
framerate
i
p
b
frames
i_size
p_size
b_size
size
o_bitrate
o_framerate
o_width
o_height
umem
utime
invalid coefficient
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

Q2

Plot the histogram of numerical features. What preprocessing can be done if the distribution of a feature has high skewness?

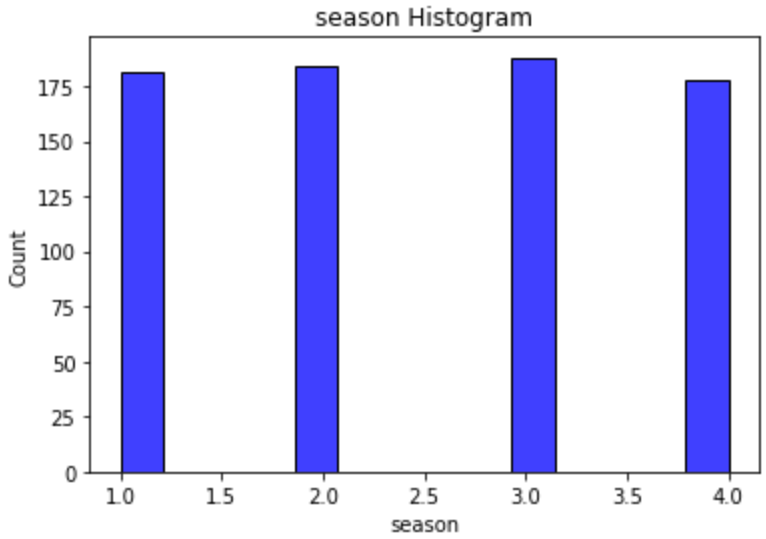
ANSWER

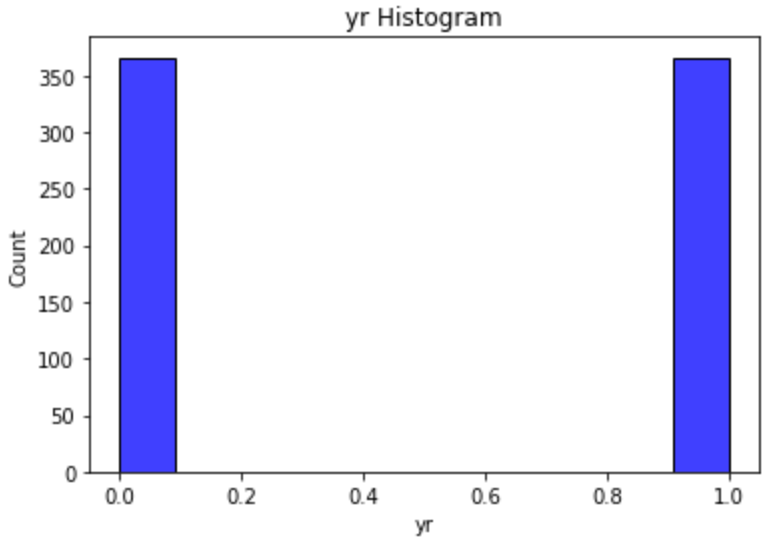
The following techniques can be used to address feature skew:

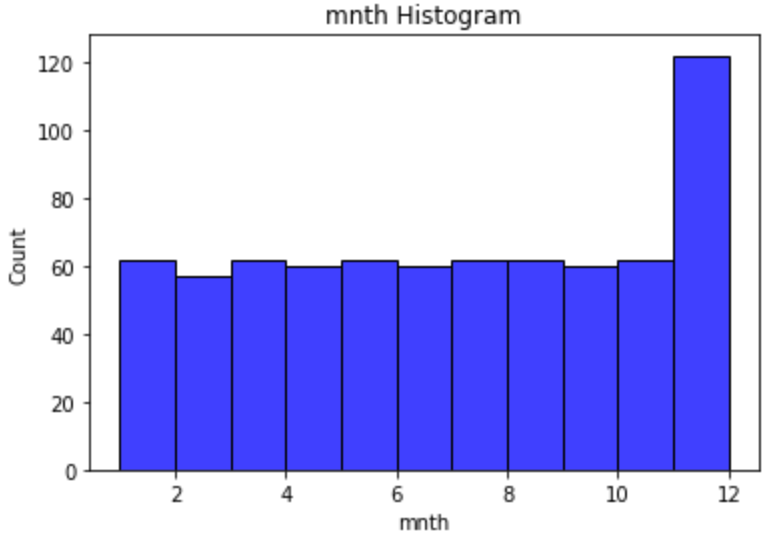
- **log transform:** the logarithm - $log_a \mid a \in \{2, 10, e, ...\}$ of x, is a strong transformation and can be used to reduce right skewness.
- **boxcox transform:** a transformation of a non-normal dependent variables into a normal shape.
- **root transform (square, cube, etc.):** fairly strong transformation for negatively skewed data with a substantial effect on distribution shape, but is weaker than the logarithm. Can be applied to negative and zero values too.
- **power transform (square, cube, etc.):** $x \rightarrow x^n \mid n \in \{2, 3, ...\}$has a moderate effect on distribution shape and it could be used to reduce left skewness.

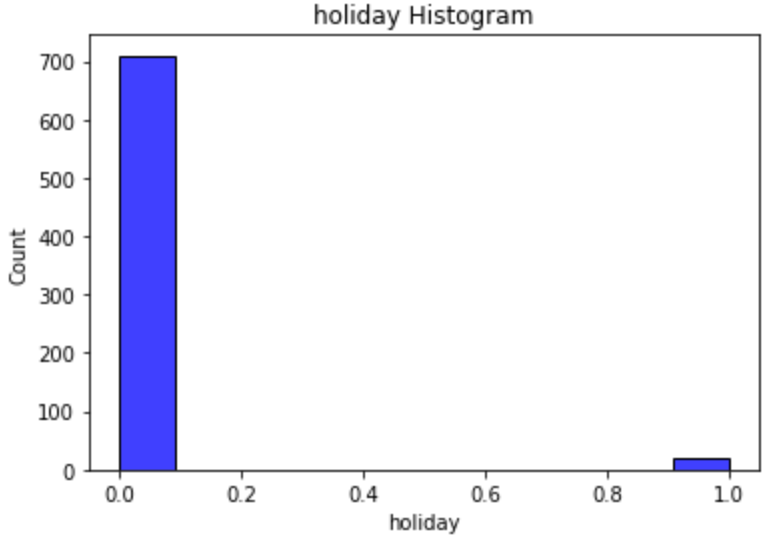
NOTE: for the histograms, we plotted some numerically encoded categorical features like season and holiday because we thought it was helpful to visualize them even if the question's intention was to plot numerical continuous features only.

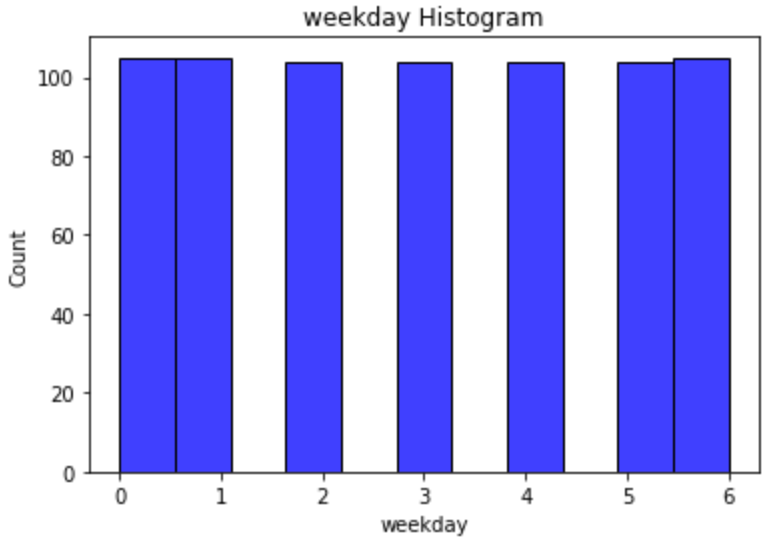
Bike Sharing

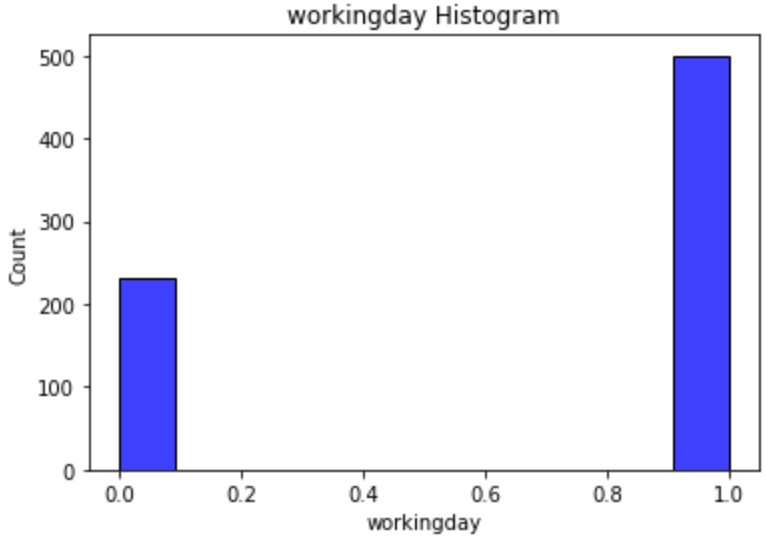


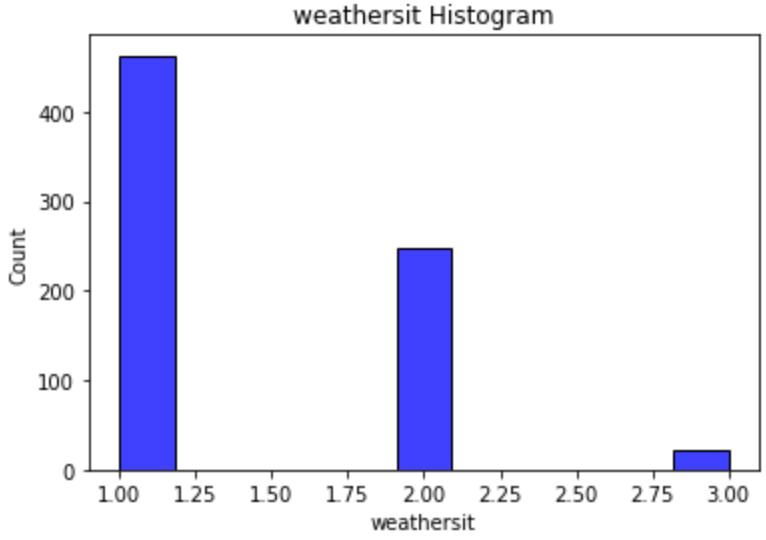

yr Histogram
Count
yr
0
50
100
150
200
250
300
350
0.0
0.2
0.4
0.6
0.8
1.0

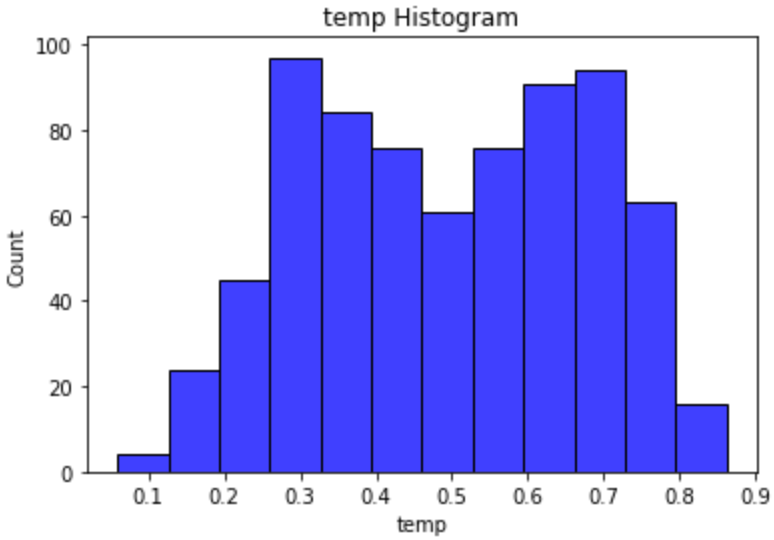

mnth Histogram
Count
mnth
0
20
40
60
80
100
120
2
4
6
8
10
12

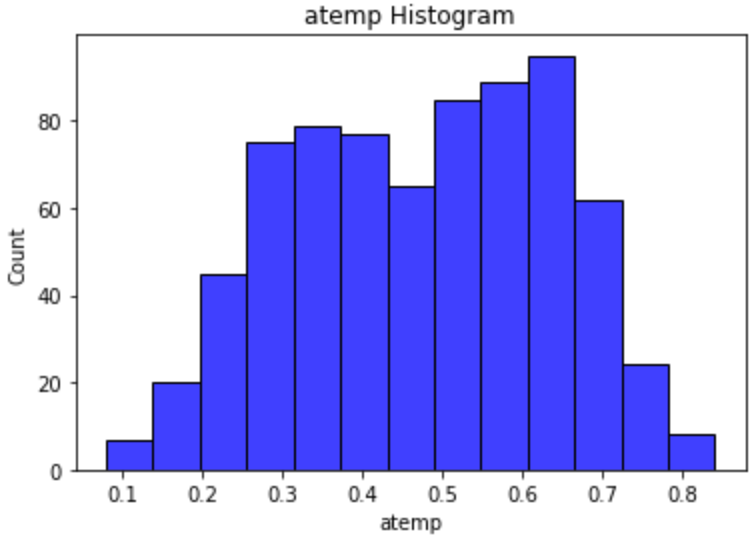

holiday Histogram
Count
holiday
0
100
200
300
400
500
600
700
0.0
0.2
0.4
0.6
0.8
1.0

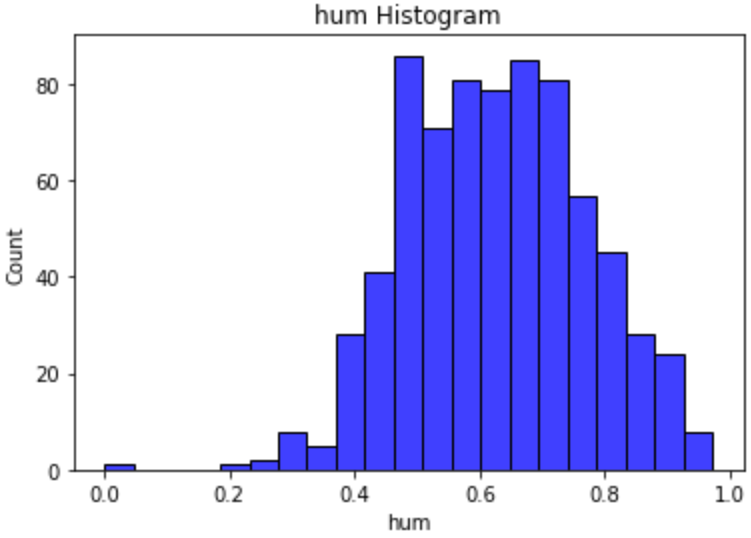

weekday Histogram
Count
weekday
0
20
40
60
80
100
0
1
2
3
4
5
6

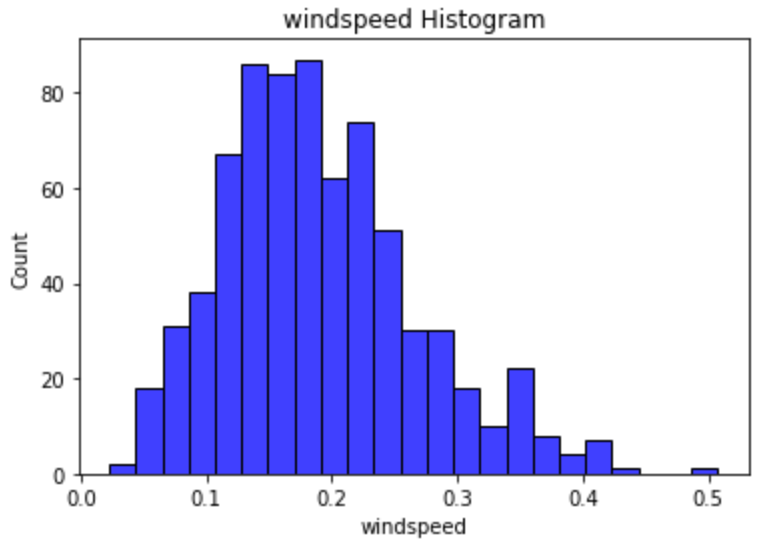

workingday Histogram
Count
workingday
0
100
200
300
400
500
0.0
0.2
0.4
0.6
0.8
1.0

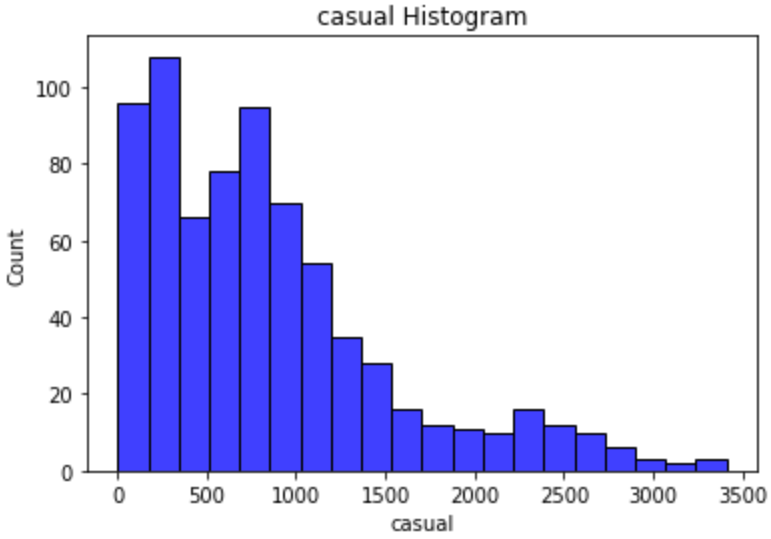

weathersit Histogram
Count
weathersit
0
100
200
300
400
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00

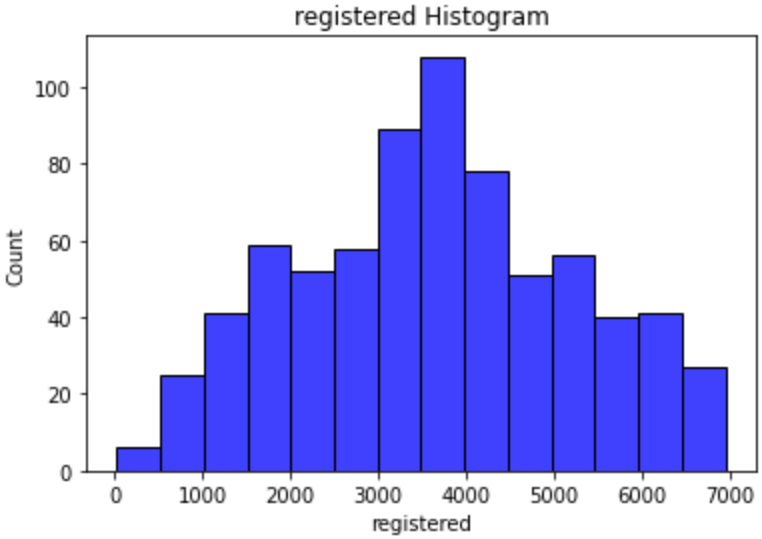

temp Histogram
Count
temp
0
20
40
60
80
100
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

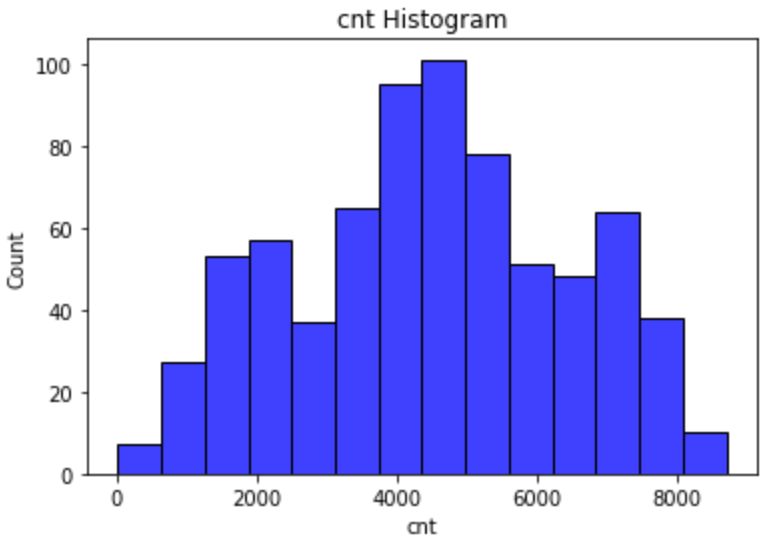

atemp Histogram
Count
atemp
0
20
40
60
80
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8


hum Histogram
Count
hum
0
20
40
60
80
0.0
0.2
0.4
0.6
0.8
1.0

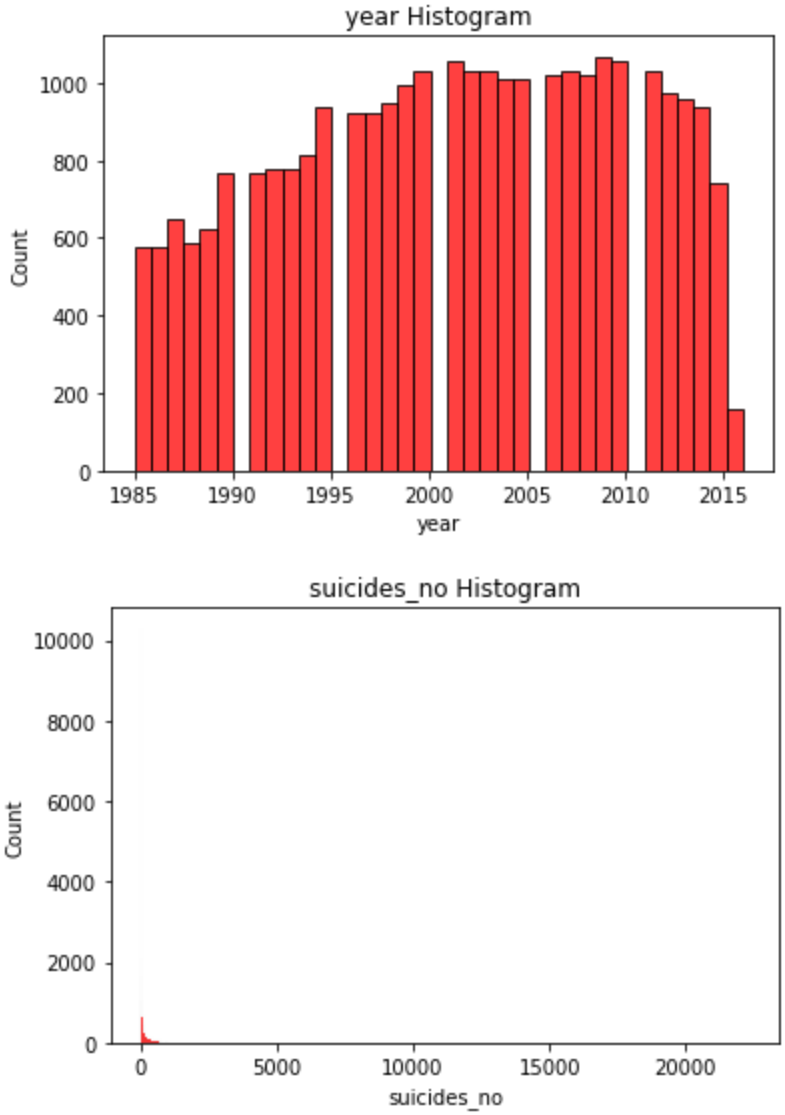

windspeed Histogram
Count
windspeed
0
20
40
60
80
0.0
0.1
0.2
0.3
0.4
0.5

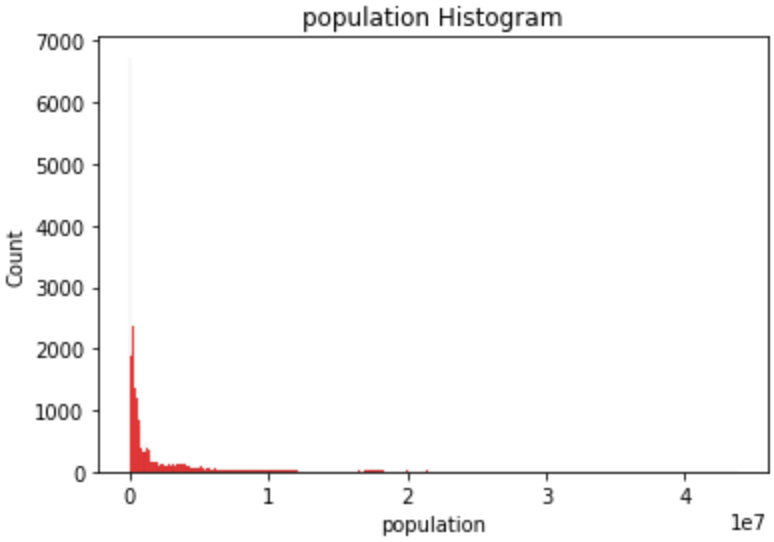

casual Histogram
Count
casual
0
20
40
60
80
100
0
500
1000
1500
2000
2500
3000
3500

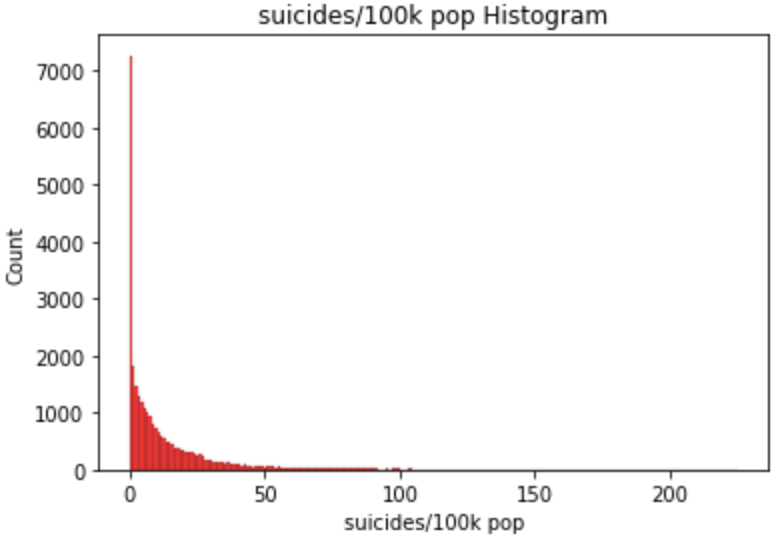

registered Histogram
Count
registered
0
20
40
60
80
100
0
1000
2000
3000
4000
5000
6000
7000

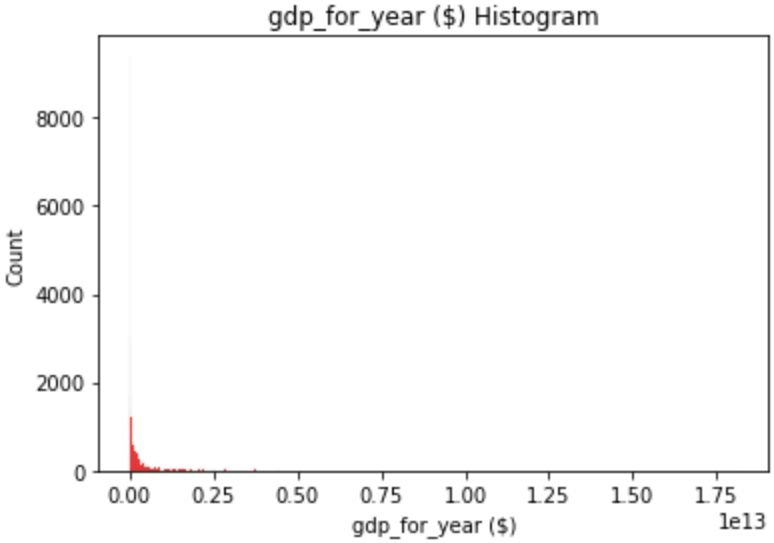

cnt Histogram
Count
cnt
0
20
40
60
80
100
0
2000
4000
6000
8000

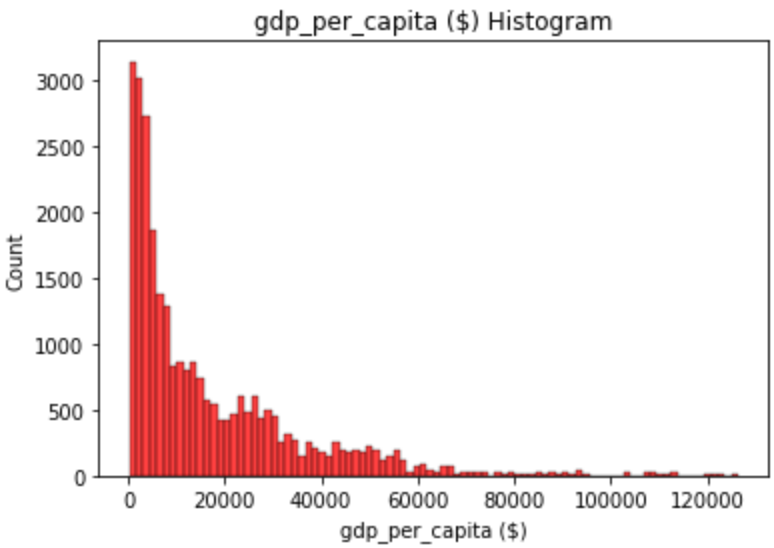
Suicide Rate



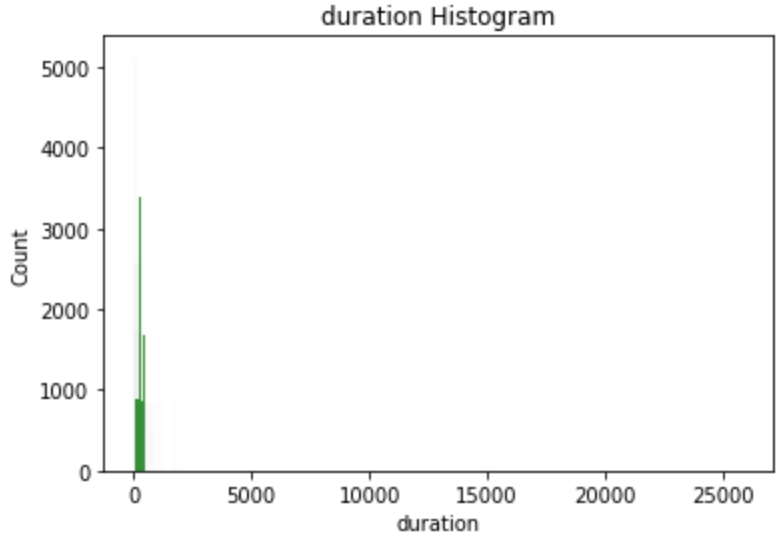

population Histogram
Count
7000
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
population
1e7

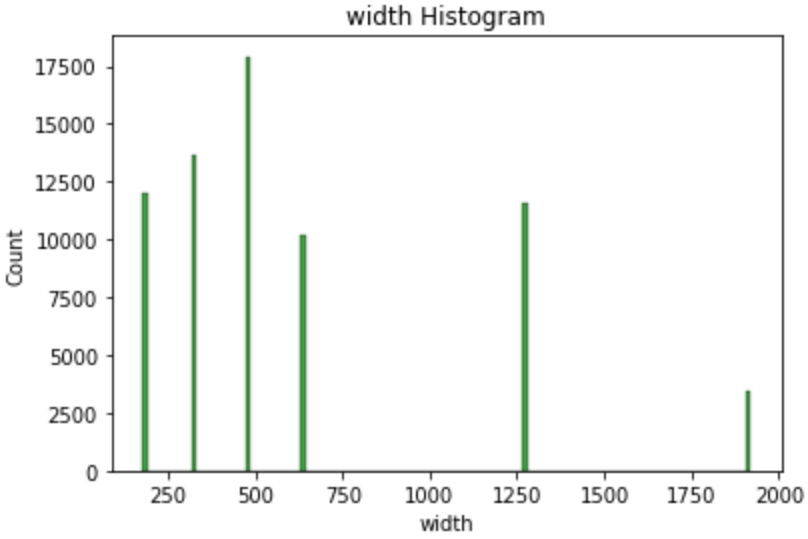

suicides/100k pop Histogram
Count
7000
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
suicides/100k pop

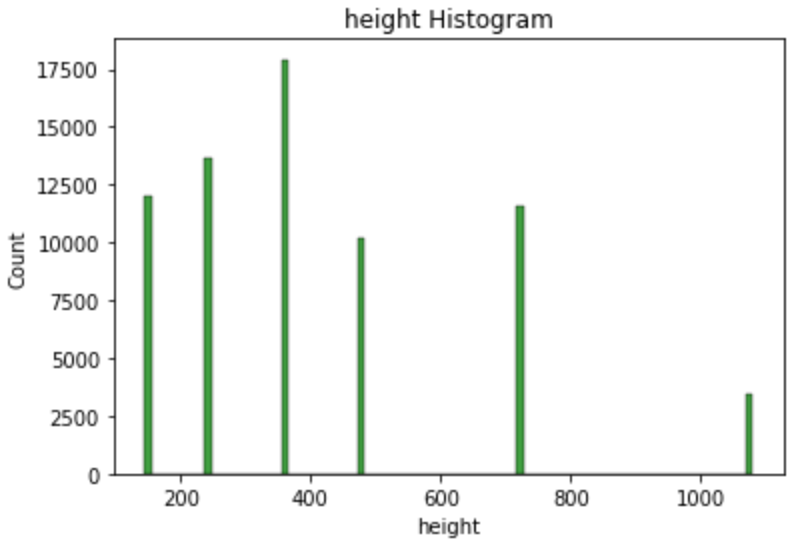

gdp_for_year ($) Histogram
Count
8000
6000
4000
2000
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
gdp_for_year ($)
1e13

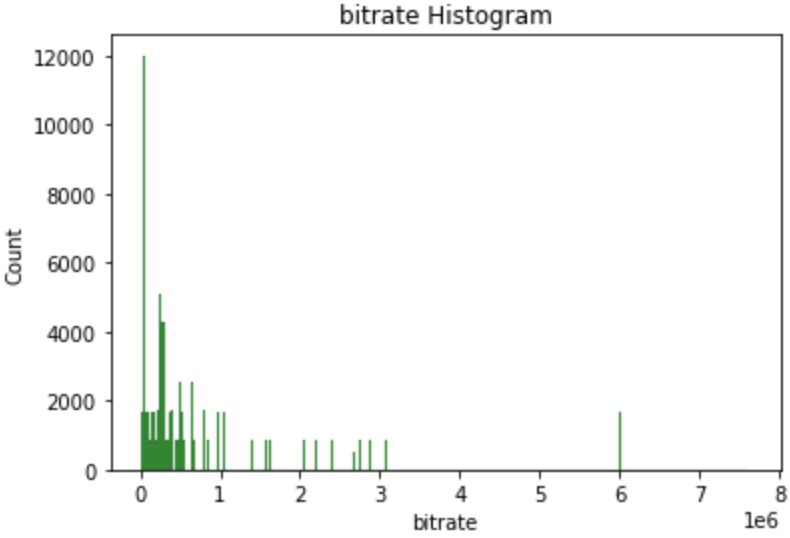

gdp_per_capita ($) Histogram
Count
3000
2500
2000
1500
1000
500
0
0
20000
40000
60000
80000
100000
120000
gdp_per_capita ($)

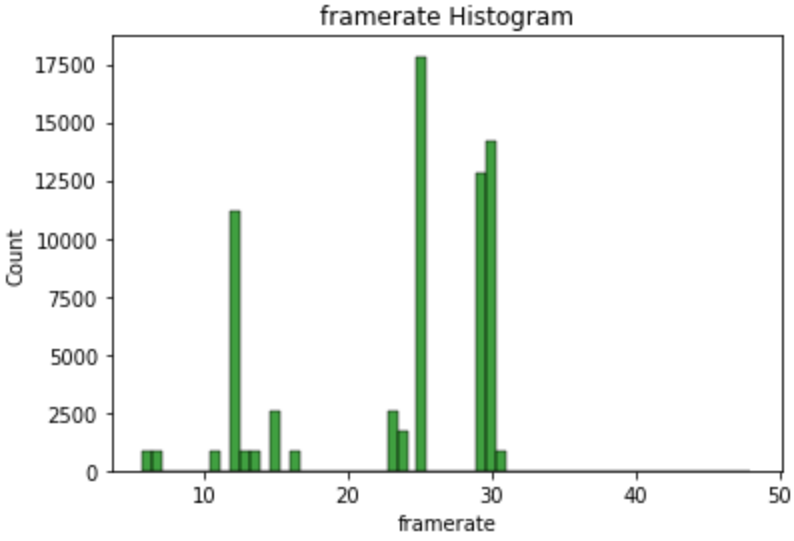
Video Transcoding Time

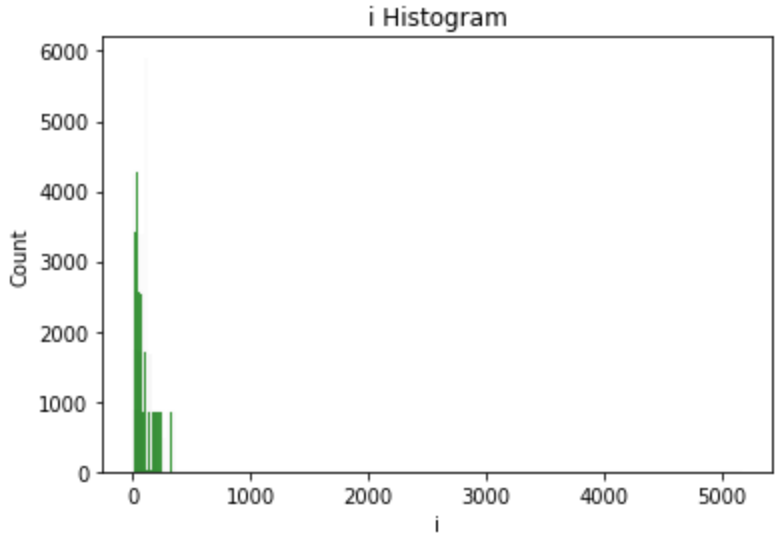


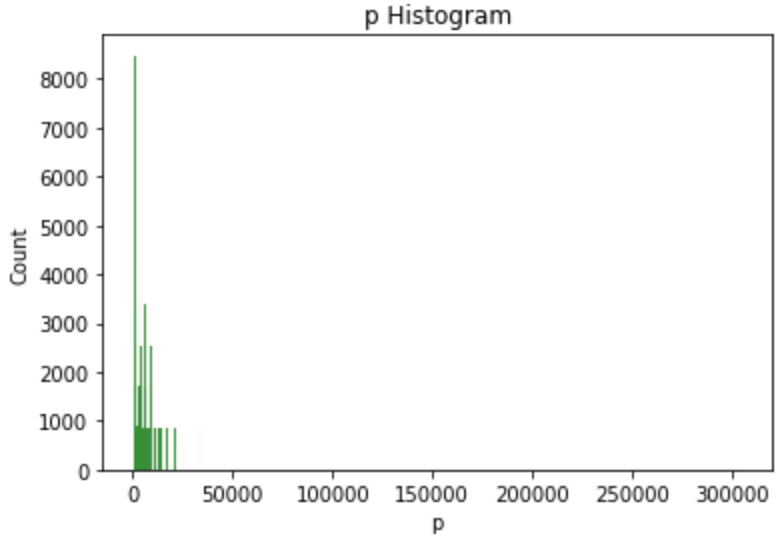


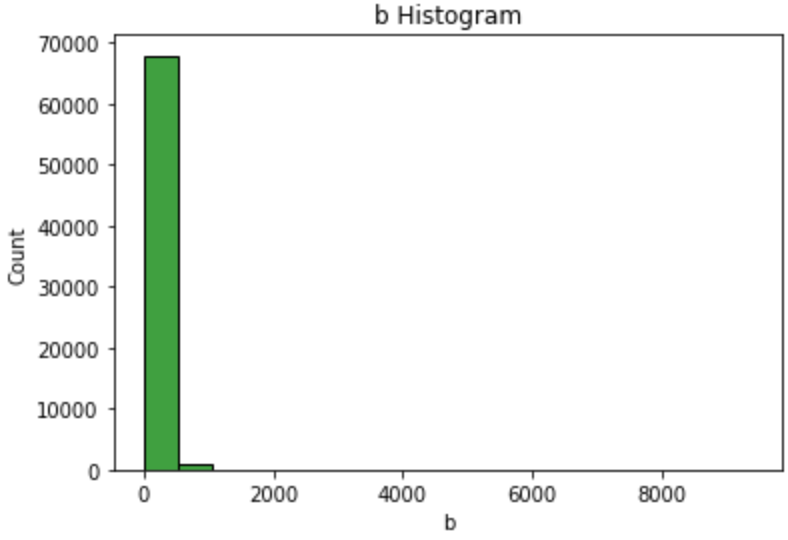

height Histogram
Count
height
0
2500
5000
7500
10000
12500
15000
17500
200
400
600
800
1000

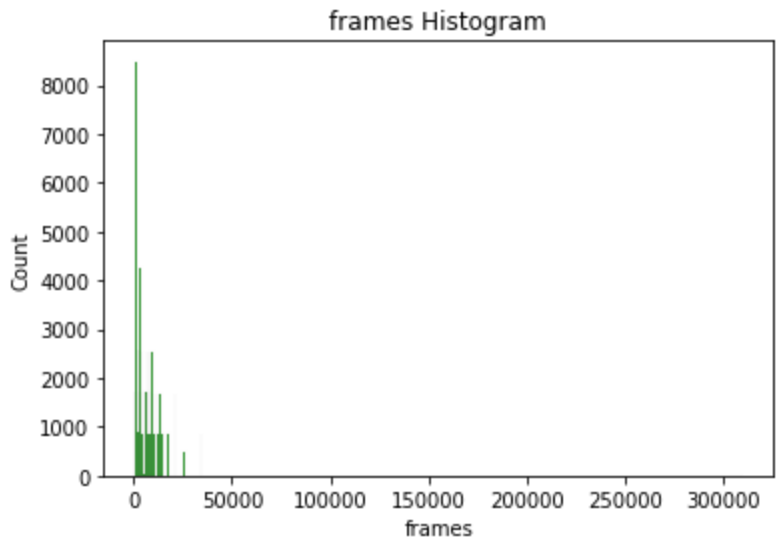

bitrate Histogram
Count
bitrate
1e6
0
2000
4000
6000
8000
10000
12000
0
1
2
3
4
5
6
7
8

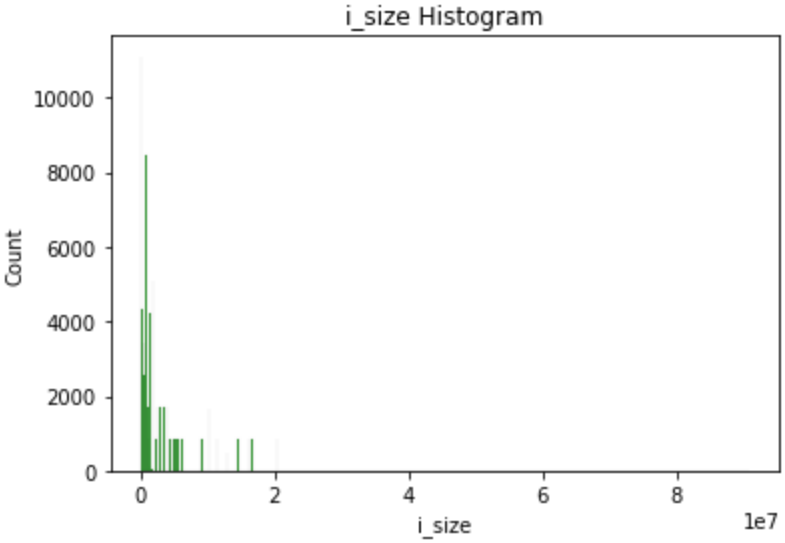

framerate Histogram
Count
framerate
0
2500
5000
7500
10000
12500
15000
17500
10
20
30
40
50

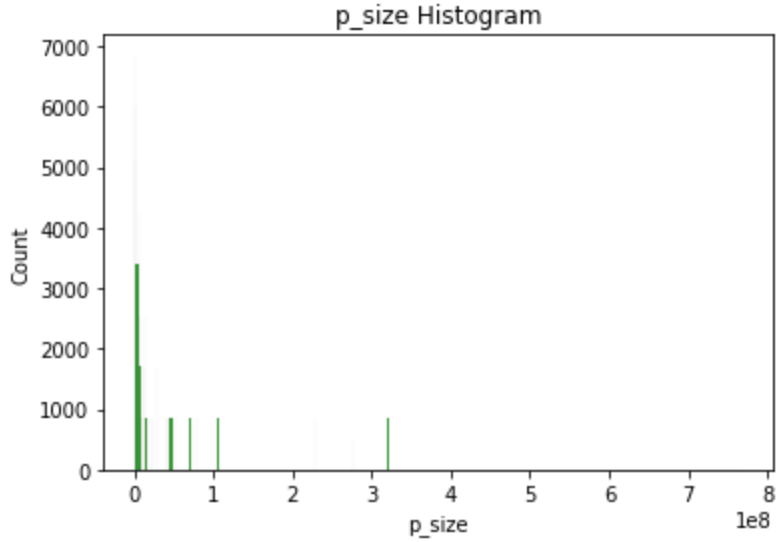

i Histogram
Count
i
0
1000
2000
3000
4000
5000
6000

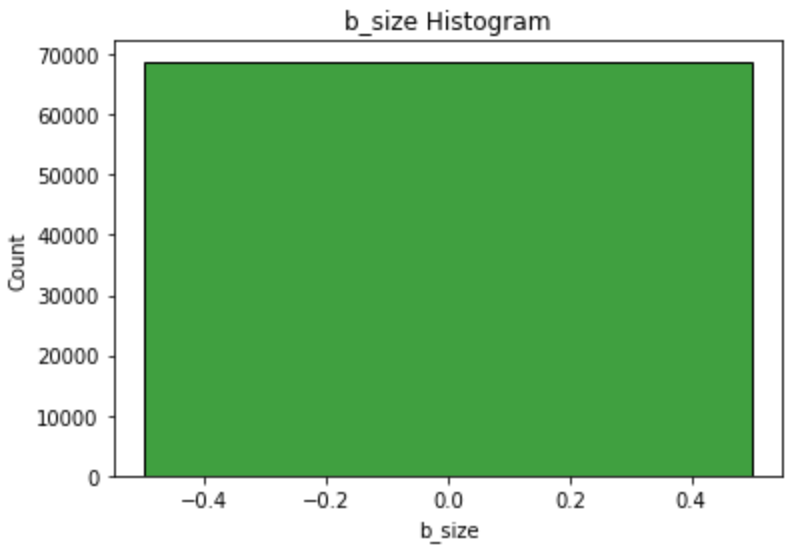

p Histogram
Count
p
0
50000
100000
150000
200000
250000
300000
1000
2000
3000
4000
5000
6000
7000
8000

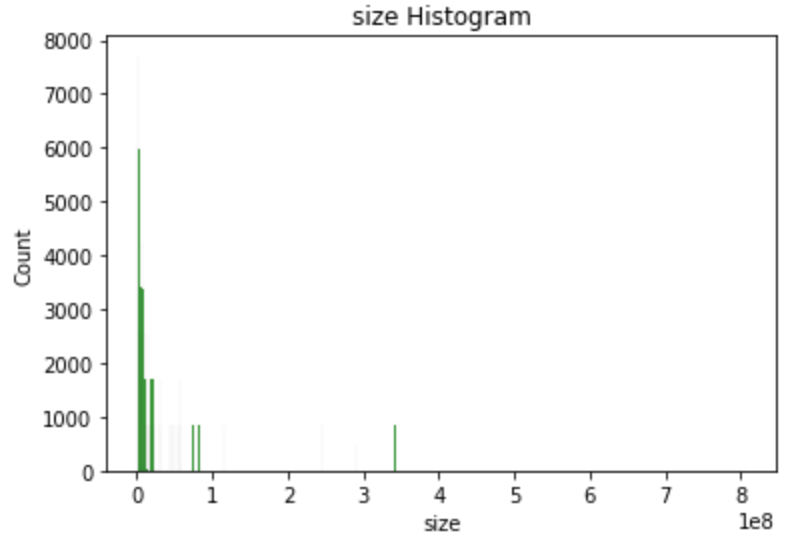

b Histogram
Count
b
0
2000
4000
6000
8000
10000
20000
30000
40000
50000
60000
70000

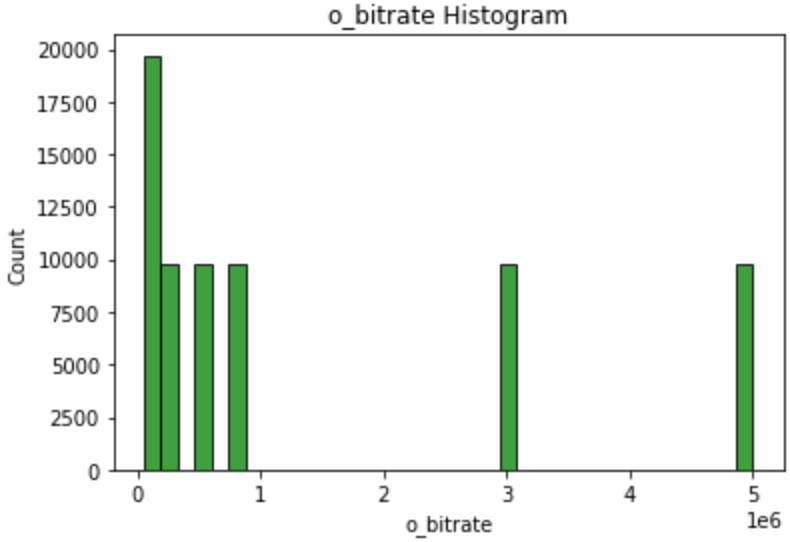

frames Histogram
Count
frames
0
1000
2000
3000
4000
5000
6000
7000
8000
0
50000
100000
150000
200000
250000
300000

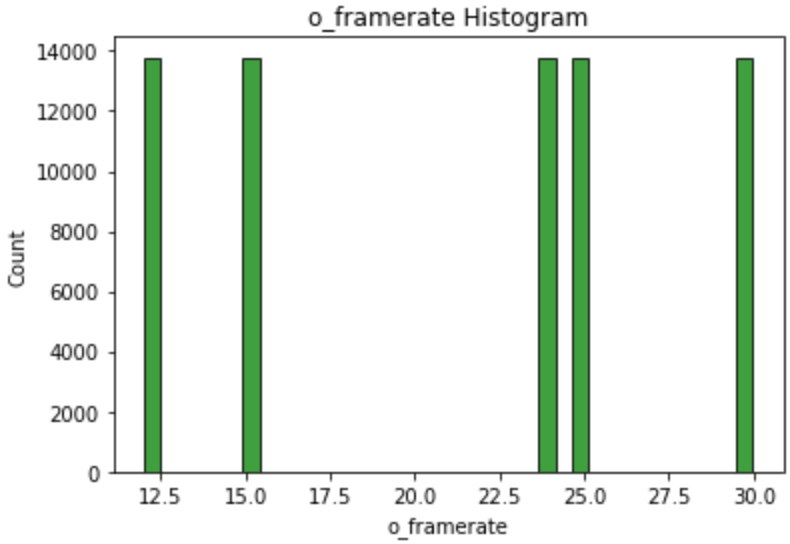

i_size Histogram
Count
i_size
1e7
0
2000
4000
6000
8000
10000
0
2
4
6
8

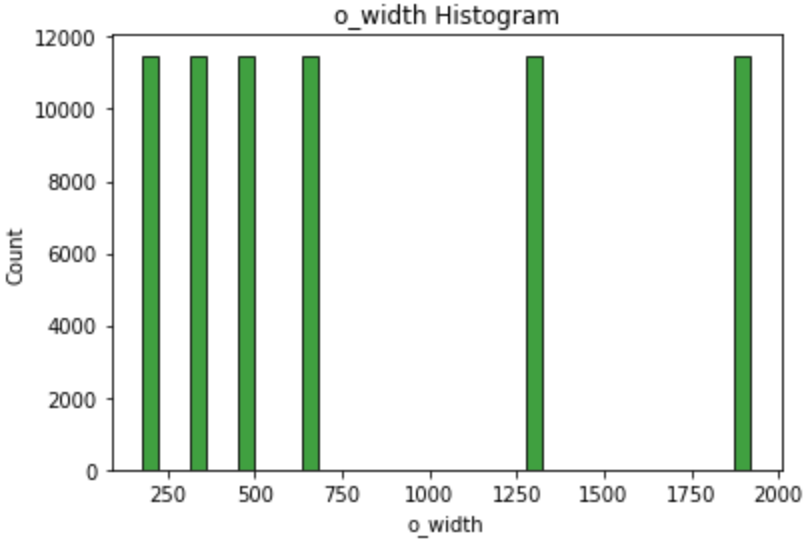

p_size Histogram
Count
p_size
1e8
0
1000
2000
3000
4000
5000
6000
7000
0
1
2
3
4
5
6
7
8

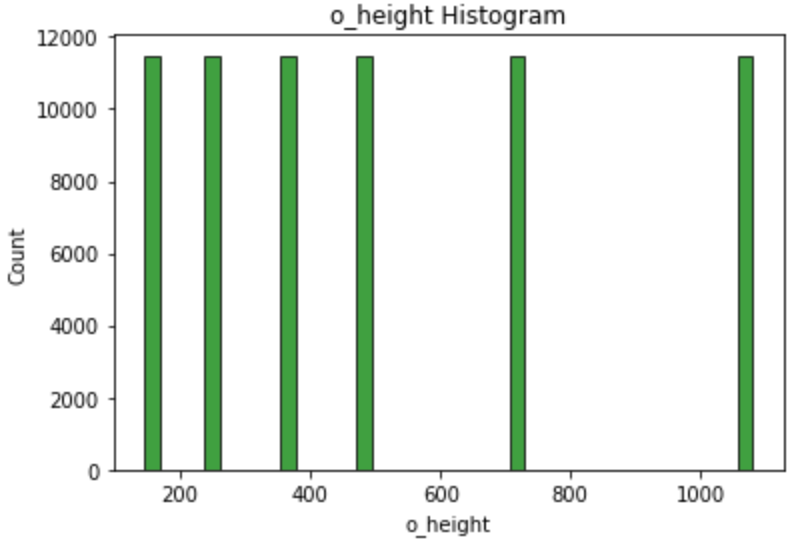

b_size Histogram
Count
b_size
70000
60000
50000
40000
30000
20000
10000
0
−0.4
−0.2
0.0
0.2
0.4

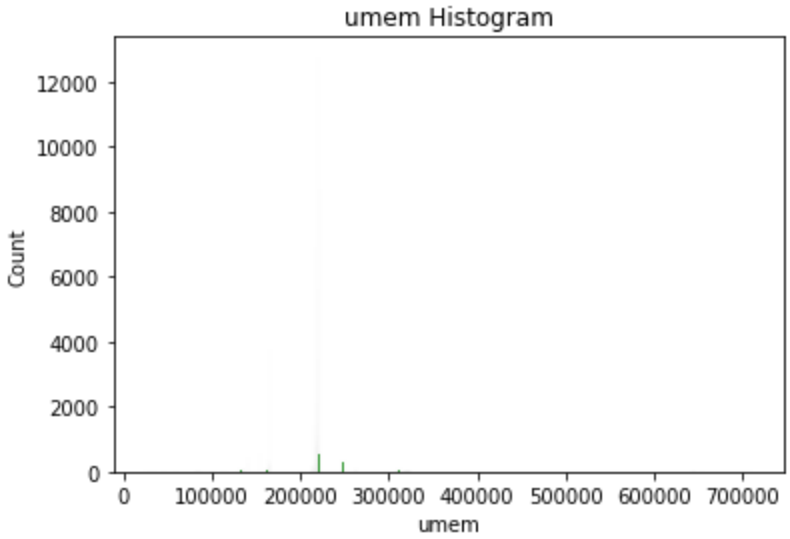

size Histogram
Count
size
1e8
8000
7000
6000
5000
4000
3000
2000
1000
0
0
1
2
3
4
5
6
7
8

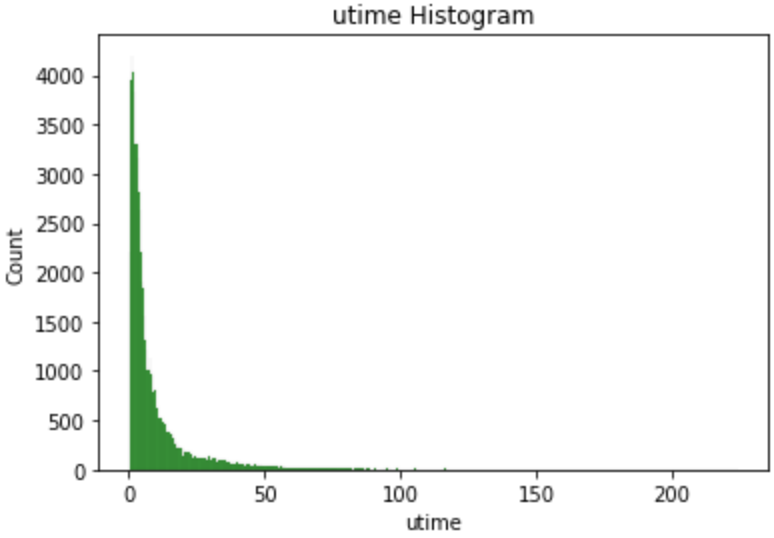

o_bitrate Histogram
Count
o_bitrate
1e6
20000
17500
15000
12500
10000
7500
5000
2500
0
0
1
2
3
4
5


o_framerate Histogram
Count
o_framerate
14000
12000
10000
8000
6000
4000
2000
0
12.5
15.0
17.5
20.0
22.5
25.0
27.5
30.0


o_width Histogram
Count
o_width
12000
10000
8000
6000
4000
2000
0
250
500
750
1000
1250
1500
1750
2000


o_height Histogram
Count
o_height
12000
10000
8000
6000
4000
2000
0
200
400
600
800
1000


umem Histogram
Count
umem
0
2000
4000
6000
8000
10000
12000
0
100000
200000
300000
400000
500000
600000
700000

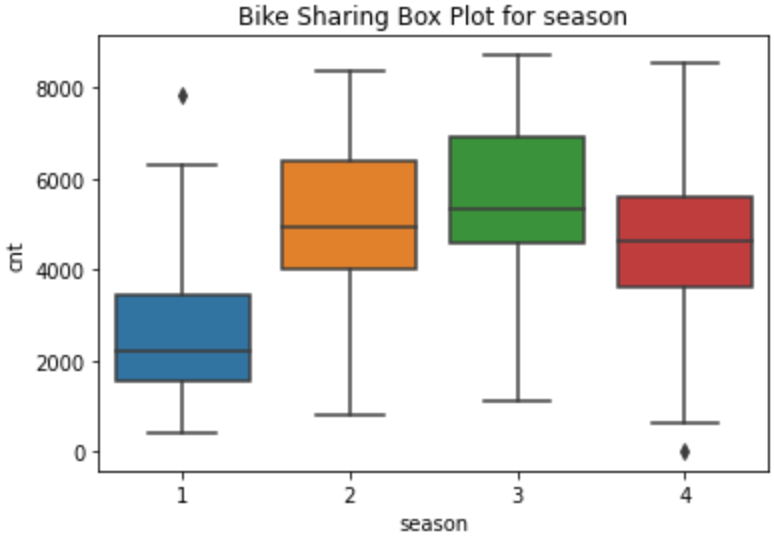

utime Histogram
Count
utime
0
500
1000
1500
2000
2500
3000
3500
4000
0
50
100
150
200

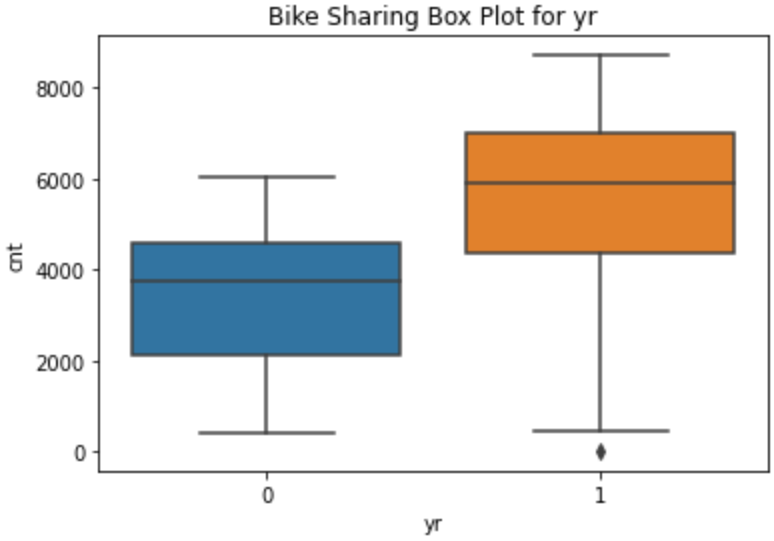
Q3

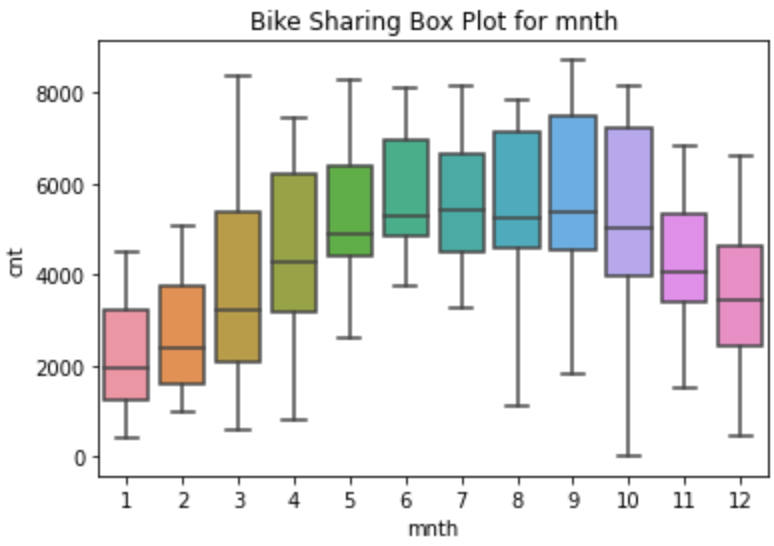
ANSWER

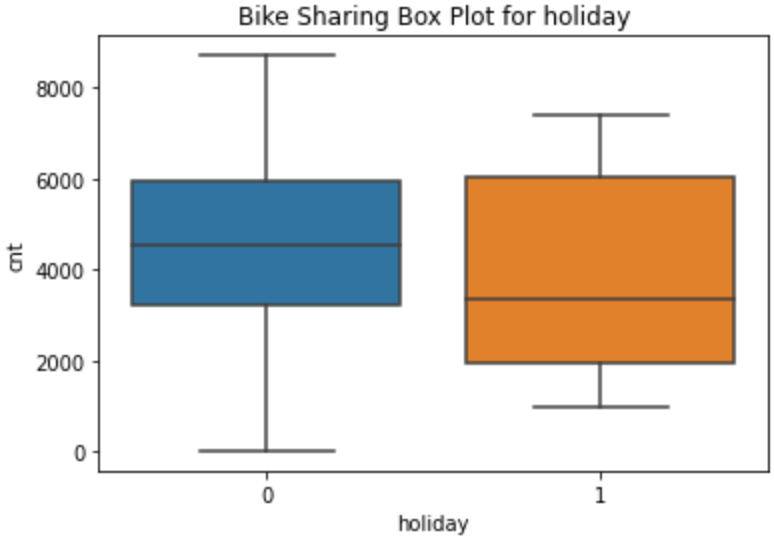
Intuitions on categorical variables:

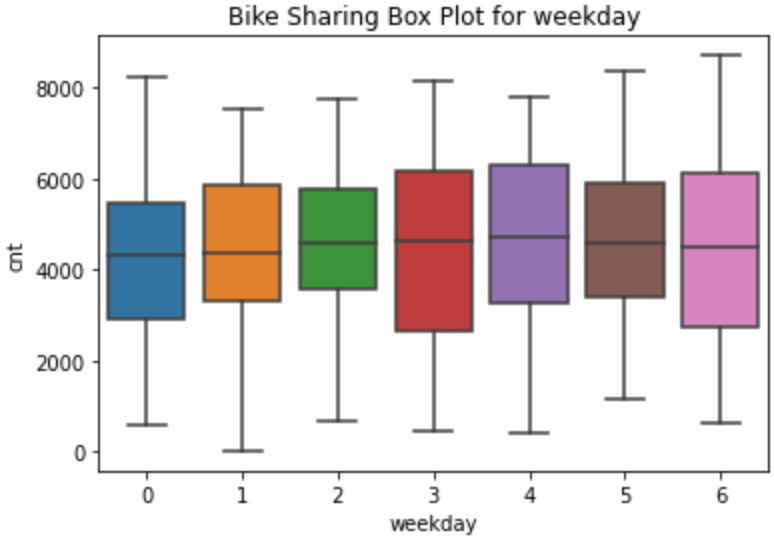
- **Bike Sharing**
  - **season**: shows that spring and summer have the highest rental rates (unsurprising)
  - **yr**: suggests that the company whose data we have is growing (i.e. the latter year has higher overall rentals)
  - **mnth**: shows a more fine grained view of the seasonality seen earlier (i.e. warmer months have higher rentals)
  - **holiday**: shows that on average, rentals are lower for holidays; maybe due to more competing leisurely alternatives
  - **weekday**: shows no clear peaks on particular weekdays, although there seems to be a slight boost mid-week
  - **workingday**: same as above, no clear trends favoring working days over non-working
  - **weathersit**: shows strong decrease in rentals when the weather is inclement (unsurprising)
- **Suicides**
  - **country**: shows that a lot of countries connected to the former USSR have higher suicide rates
  - **year**: shows that suicides peaked around 1995 and were decreasing until about 2015
  - **sex**: shows that men generally commit suicide at higher rates than women
  - **age**: shows that the older someone is, the more likely they are to commit suicide
  - **generation**: shows a similar trend to the age feature
- **Video Transcoding**
  - **codec**: shows relatively consistent transcoding time between codec options
  - **o_codec**: shows h264 and vp8 have both longer and more varied transcoding times; flv and mpeg are more consistent

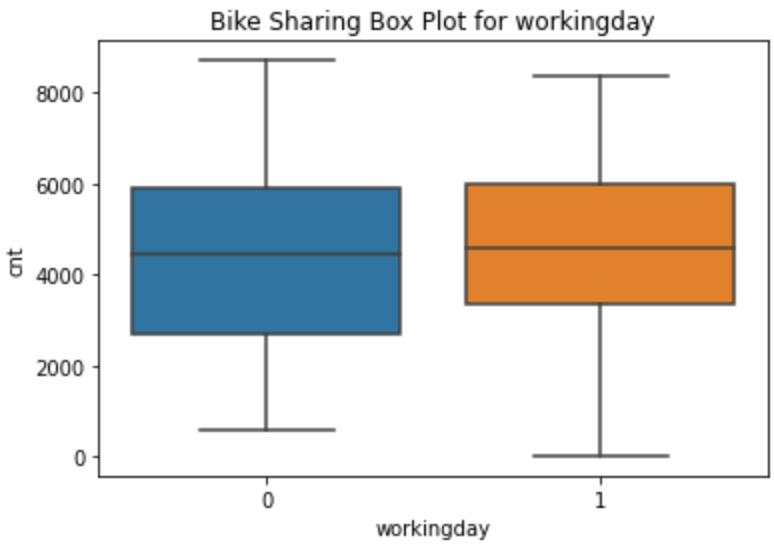

Bike Sharing Box Plot for season
cnt
8000
6000
4000
2000
0
1
2
3
4
season

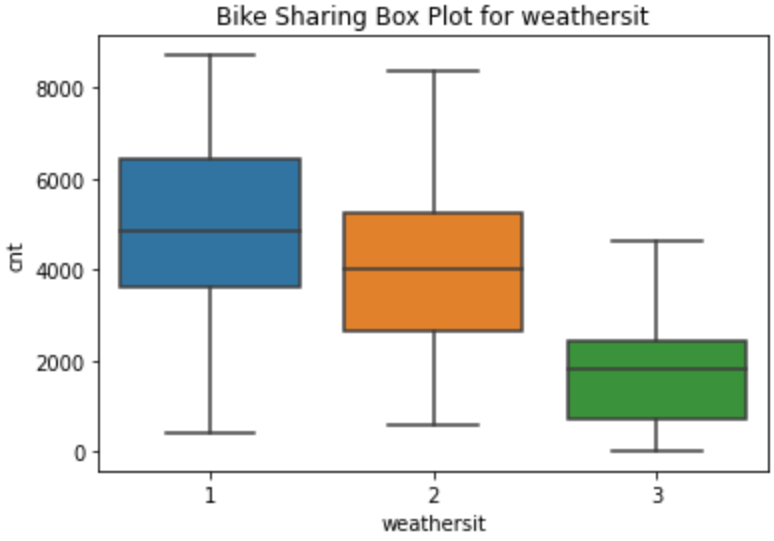

Bike Sharing Box Plot for yr
cnt
8000
6000
4000
2000
0
0
1
yr


Bike Sharing Box Plot for mnth
cnt
mnth
0
2000
4000
6000
8000
1
2
3
4
5
6
7
8
9
10
11
12

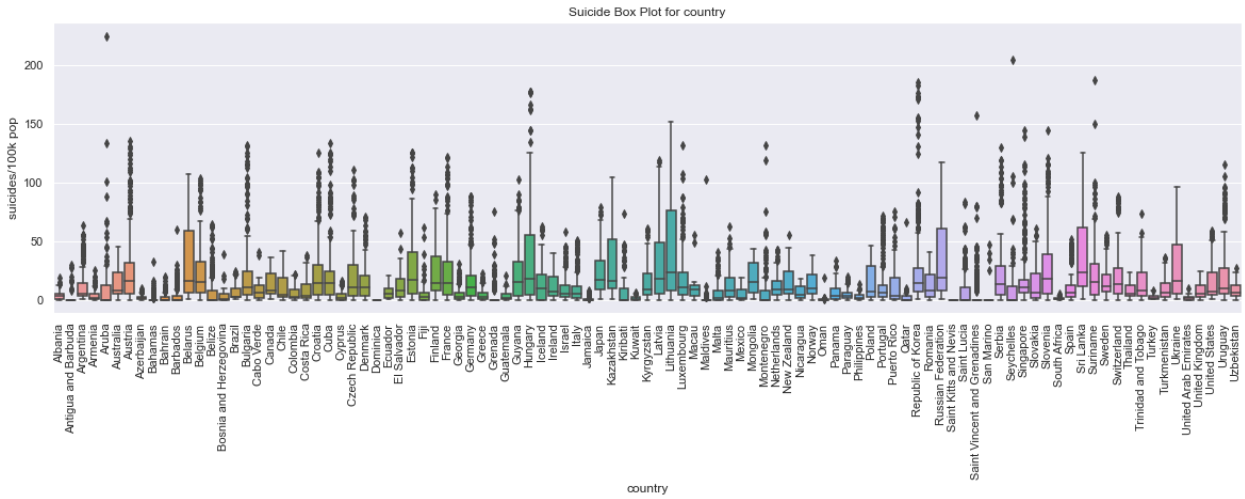

Bike Sharing Box Plot for holiday
cnt
holiday
0
2000
4000
6000
8000
0
1

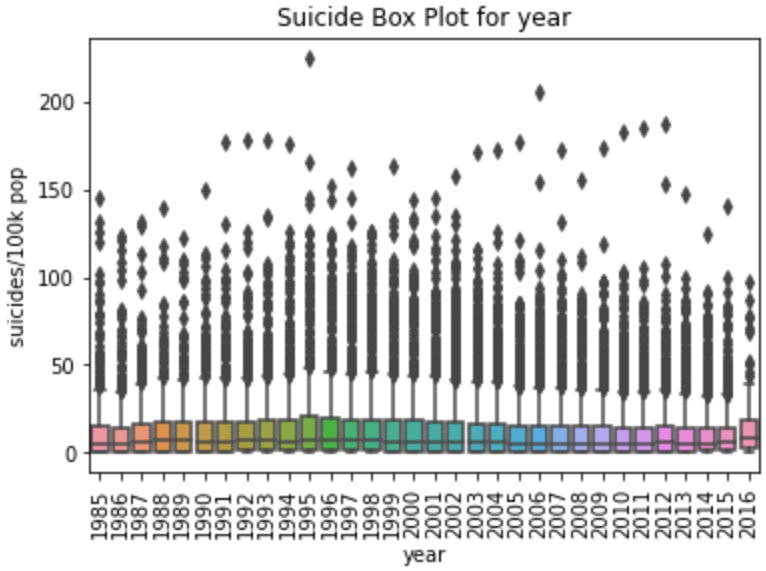

Bike Sharing Box Plot for weekday
cnt
weekday
0
2000
4000
6000
8000
0
1
2
3
4
5
6

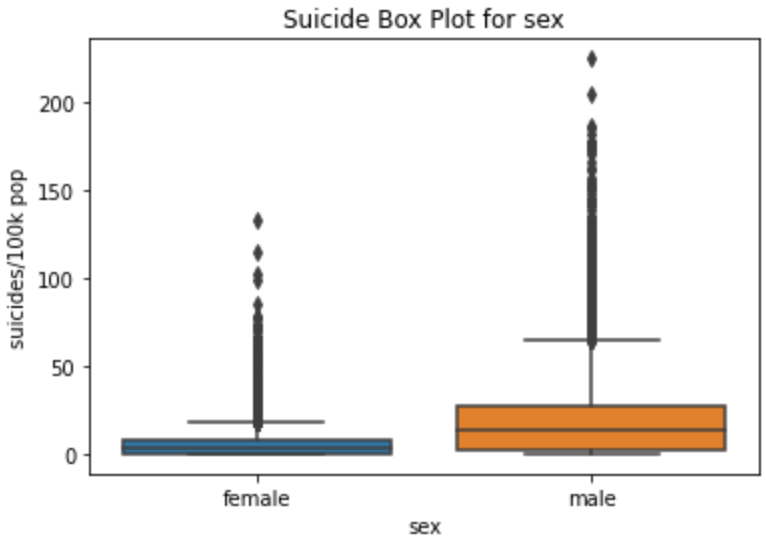

Bike Sharing Box Plot for workingday
cnt
8000
6000
4000
2000
0
0
1
workingday

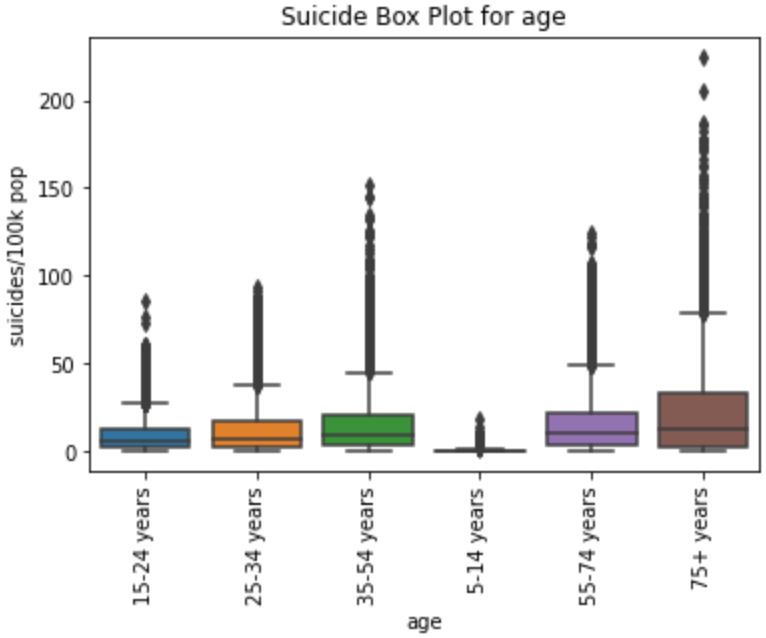

Bike Sharing Box Plot for weathersit
cnt
8000
6000
4000
2000
0
1
2
3
weathersit

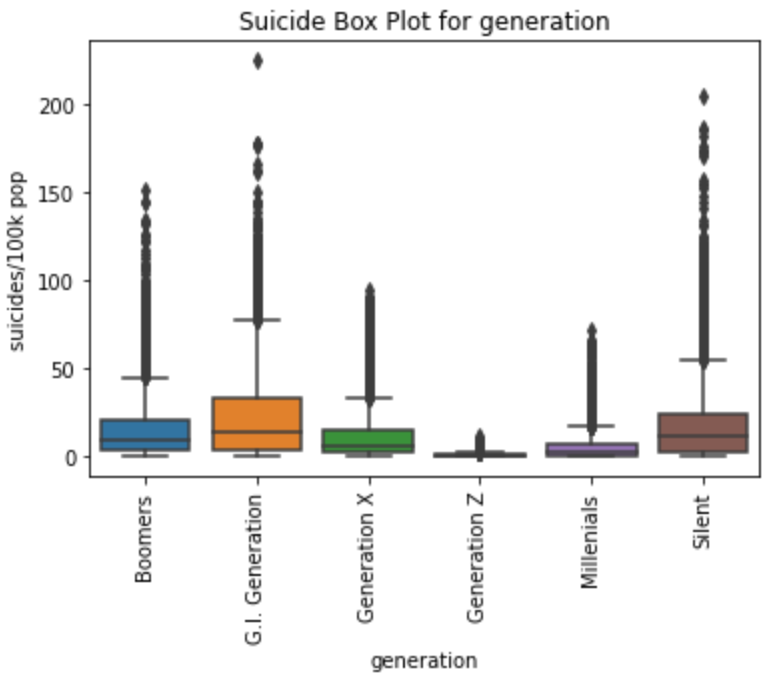
Suicide Rate



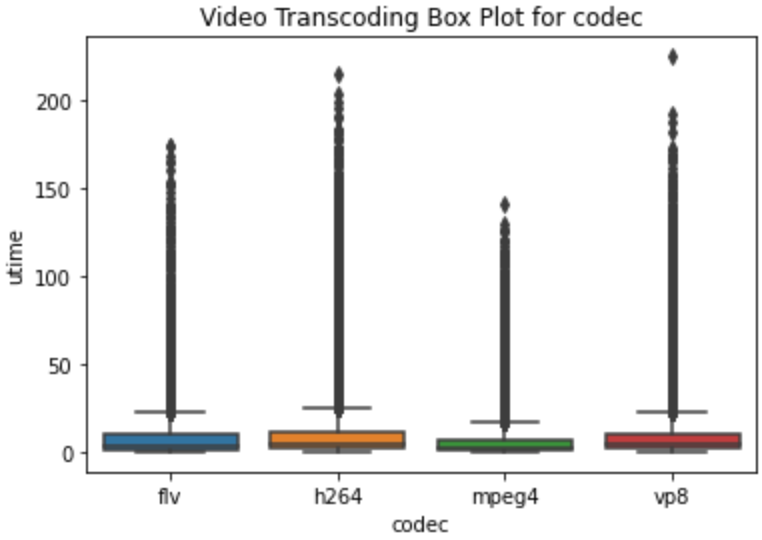


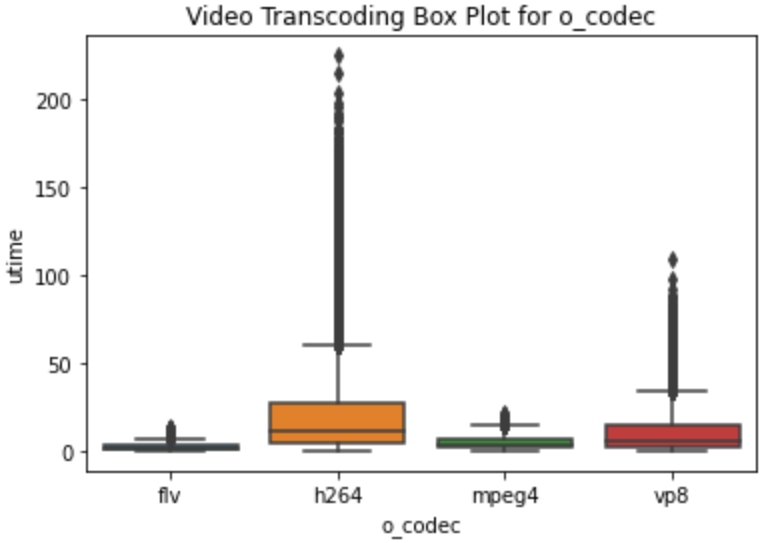

Suicide Box Plot for sex
suicides/100k pop
0
50
100
150
200
female
male
sex


Suicide Box Plot for age
suicides/100k pop
0
50
100
150
200
15-24 years
25-34 years
35-54 years
5-14 years
55-74 years
75+ years
age


Suicide Box Plot for generation
suicides/100k pop
200
150
100
50
0
Boomers
G.I. Generation
Generation X
Generation Z
Millenials
Silent
generation

Video Transcoding Time



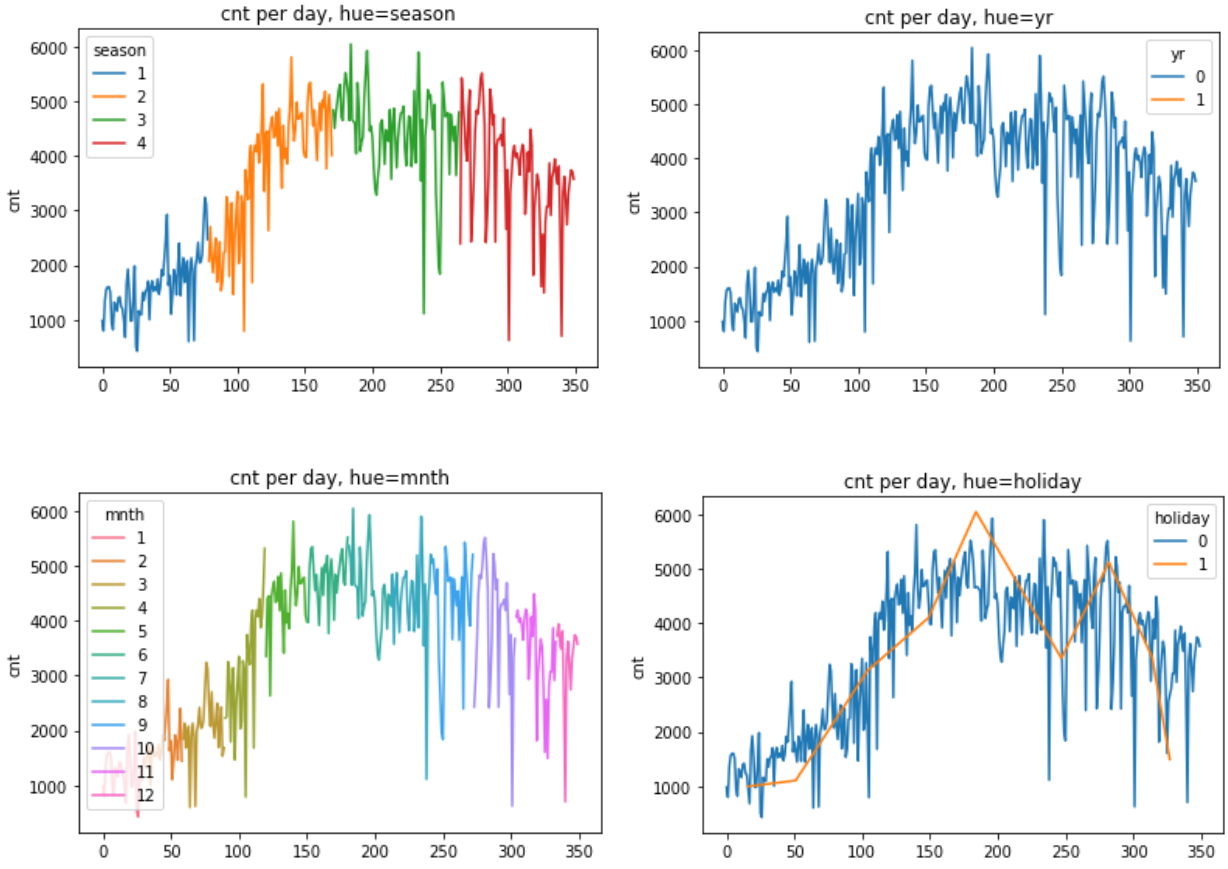


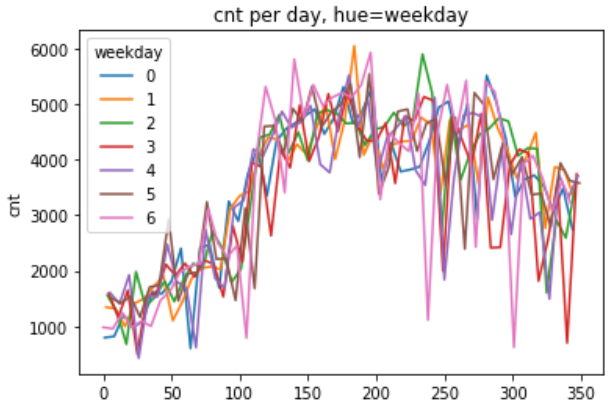
Q4

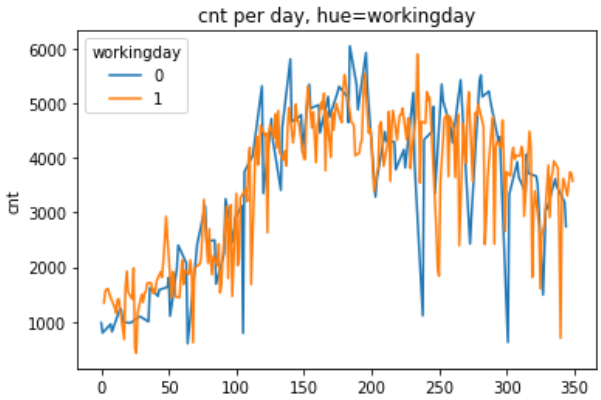
For bike sharing dataset, plot the count number per day for a few months. Can you identify any repeating patterns in every month?

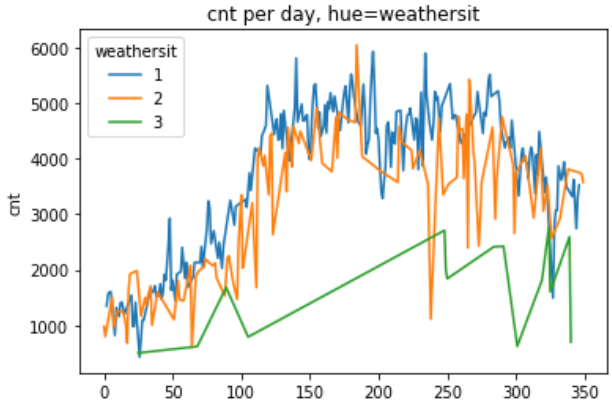
ANSWER

Saturdays and Sundays typically have the highest rental rates, followed by Mondays. The weekend use is probably for casual leisure and the Monday rentals might be more likely to be for work commutes.




cnt per day, hue=weekday
weekday
0
1
2
3
4
5
6
cnt
6000
5000
4000
3000
2000
1000
0
50
100
150
200
250
300
350

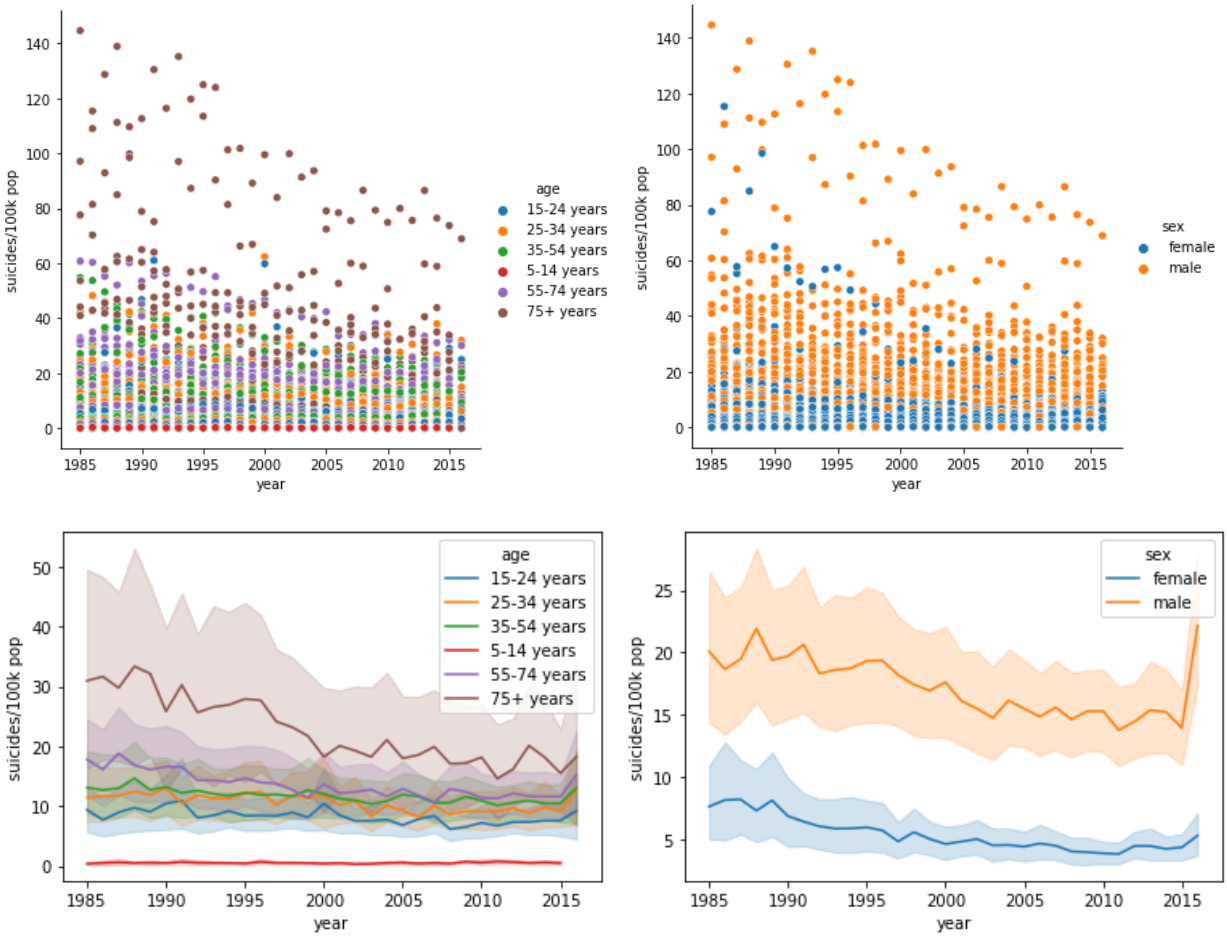

cnt per day, hue=workingday
workingday
0
1
cnt
6000
5000
4000
3000
2000
1000
0
50
100
150
200
250
300
350


cnt per day, hue=weathersit
weathersit
1
2
3
cnt
6000
5000
4000
3000
2000
1000
0
50
100
150
200
250
300
350

Q5

For the suicide rate dataset, pick the top 10 countries that have the longest time-span of records (in terms of years). Plot the suicide rate ("suicides/100k pop") against time for different age groups and gender, and explain your observations.**ANSWER**

- Age
    - The elderly (75+ yo) are the most likely to commit suicide and the youngest (5-14 yo) are the least likely to kill commit suicide.
    - There is a strong correlation between age group and suicide rate: the older someone is the more likely they are to commit suicide.
- Sex
    - Men commit suicide at a much higher rate than women.
    - There was a significant spike in male suicides around 2015-2016

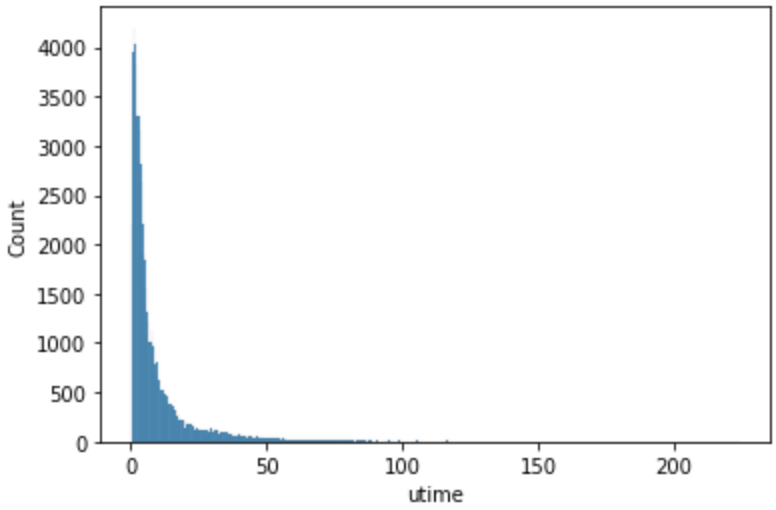


Q6

For video transcoding time dataset, plot the distribution of video transcoding times, what can you observe? Report mean and median transcoding times.

ANSWER

There is a significant left-skew in the utime data where the vast majority of observations are lower than 15ms. A long tail of longer utimes extends but there are very few instances of them in our dataset.




```
mean utime:  10.00ms
median utime: 4.41ms
```

Q7

Can you explain a trade-off here [between one-hot encoding and scalar encoding]? (Hint: let us assume we perform linear regression, what information does one-hot encoding discard, and what assumption should hold strongly if we perform the scalar encoding instead?)

ANSWER

- One-hot encoding (OHE)
    - Do not use on features with ordinal structure.
    - Discards similarities between feature values (hamming distance is always 1 between all values).
- Scalar encoding
    - Do not use when feature values are unrelated.
    - Preserves / creates a sense of distance between feature values (which is undesirable if there is no corresponding relationship).

In the context of linear regression, the two encoding approaches are utilized by the model in different ways. With scalar encoding, the model has only 1 weight to multiply by n different features, which limits its degrees of freedom and forces the lumping of unrelated features together on one side of the decision boundary. With OHE, the model has n weights and can express any sum/disjunction of the possible values.

For the Suicide dataset, we used `country_converter` and `pycountry_convert` to map the noisy country names to standardized ISO names and then to continents. 100% mapping was possible using this approach, whereas using just `pycountry_convert` alone lead to many countries having an “N/A” continent assignment.

Q8

Standardize feature columns and prepare them for training.

ANSWER

We normalized the continuous variables by subtracting the mean and dividing by the standard deviation.

Bike Sharing: N/A - all features appear to be normalized already

Suicide

The following columns were normalized:

- 'population'
- 'gdp_for_year ($)'
- 'gdp_per_capita ($)'

Video Transcoding

The following columns were normalized:

- 'duration',
- 'width',
- 'height',
- 'bitrate',
- 'framerate',
- 'i',
- 'p',
- 'frames',
- 'i_size',
- 'p_size',
- 'size',
- 'o_bitrate',
- 'o_framerate',
- 'o_width',
- 'o_height'

Q9

You may use these functions (i.e. sklearn.feature_selection.mutual_info_regression and sklearn.feature_selection.f_regression) to select most important features. How does this step affect the performance of your models in terms of test RMSE?

ANSWERS

For this task, I generated 2 additional datasets based on the top 10 features identified via `f_regression` and `mutual_info_regression`. I then trained 3 `SVR` models on the original dataset (containing all features) as well as the two feature selected datasets. In all cases, training on the top 10 most informative features resulted in a better RMSE than training on the full dataset. In most cases, `f_regression` showed the best performance, except on the Bike Sharing dataset.

Bike Sharing
RMSE: 1990.2930609006144
RMSE_ft: 1965.789840162959
RMSE_mi: 1960.62387277009

Suicide
RMSE: 20.04310681876698
RMSE_ft: 14.51722896683381
RMSE_mi: 15.64100924581367

Video Transcoding
RMSE: 14.511271543913297
RMSE_ft: 6.349714266154858
RMSE_mi: 14.350365906992282

Q10

What is the loss function? Explain how each regularization scheme affects the learned hypotheses.

ANSWER:

The objective function for the regression problems dealt with in this report is the mean squared error. The aim of the model is to reduce the squared deviation between the target variable and the predictions.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

MSE = mean squared error
n = number of data points
Y_i = observed values
$\hat{Y}_i$ = predicted values

Without any regularization, ie, ordinary least squares is a convex problem and is usually solved with the pseudo-inverse method. It fits the training data without any impetus to generalizability. Least squares is an unbiased estimator.

Ridge regression (aka L2 regularization) improves the generalizability of the learnt model by imposing a penalty factor on the vector 2 norm of the weights vector. It is given by:

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p}\beta_j^2$$

It effectively acts to shrink the weights associated with the features to zero, giving a more robust model.

Lasso regression (aka L1 regularization) imposes a penalty factor on the 1-norm of the weights vector. It is given by:

$$\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

It effectively acts as a feature selector since it could push low coefficients to zero, producing sparse models.

Q11

Report your choice of the best regularization scheme along with the optimal penalty parameter and briefly explain how it can be computed.

ANSWER:

Regularization increases the bias by reducing the variance. The best regularization scheme is obtained when the regularization penalty is such that both the bias and the variance are low. Such a case can be achieved when the test error and the training error are the lowest they can be against (the kink in the approximately U-shaped curves).

The best validation RMSE (with regularization) for the BIKE dataset was 763, for SUICIDE dataset was 15, and 11 for the VIDEO dataset. Recursive feature selection is performed as the last part of this question.

An elaborate process was undertaken to get the best linear model. Models were run on the complete set of features. The effect of dropping features based on the Fscores and mutual information index (MI) was also studied. The preprocessing, the results, and the analysis are explained below:

Below are the MI and F scores for each feature. Categorical variables were scalar encoded (only for the purpose of scoring features) since feature scores cannot be computed on one-hot encoded datasets:

Bike Sharing

	feature	mi	f_reg	pvals
0	season	0.465988	0.298407	2.133997e-30
1	yr	0.592423	0.714867	2.483540e-63
2	mnth	0.810982	0.128519	1.243112e-14
3	holiday	0.024801	0.007092	6.475936e-02
4	weekday	0.094525	0.006904	6.839081e-02
5	workingday	0.053017	0.005673	9.849496e-02
6	weathersit	0.142324	0.146603	2.150976e-16
7	temp	0.834838	0.981381	2.810622e-81
8	atemp	1.000000	1.000000	1.854504e-82
9	hum	0.098948	0.015467	6.454143e-03
10	windspeed	0.120754	0.087962	1.359959e-10

Suicide

	feature	mi	f_reg	pvals
0	year	0.000000	0.008431	7.353434e-11
1	sex	0.227132	1.000000	0.000000e+00
2	age	0.458164	0.200661	7.739320e-218
3	population	1.000000	0.000379	1.670210e-01
4	gdp_for_year ($)	0.374965	0.003522	2.550492e-05
5	gdp_per_capita ($)	0.323343	0.000018	7.659053e-01
6	generation	0.261683	0.013746	9.218445e-17
7	continent	0.149859	0.011214	5.891869e-14

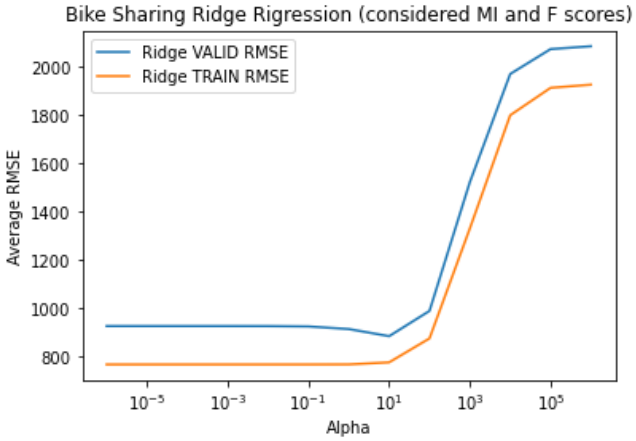
Video Transcoding

	feature	mi	f_reg	pvals
0	duration	1.000000	0.000081	1.467846e-01
1	codec	0.189577	0.000900	1.351842e-06
2	width	0.435609	0.045465	2.213683e-256
3	height	0.431167	0.044485	5.955923e-251
4	bitrate	0.953942	0.065419	0.000000e+00
5	framerate	0.534009	0.016789	1.889575e-96
6	i	0.875387	0.000906	1.238180e-06
7	p	0.962952	0.002925	3.047665e-18
8	b	0.002121	0.000070	1.776859e-01
9	frames	0.956993	0.002910	3.718494e-18
10	i_size	0.940567	0.011146	9.874803e-65
11	p_size	0.953539	0.025514	2.541518e-145
12	size	0.952545	0.025226	1.043438e-143
13	o_codec	0.824045	0.013240	1.631317e-76
14	o_bitrate	0.057788	0.065660	0.000000e+00
15	o_framerate	0.047011	0.029005	8.053878e-165
16	o_width	0.908783	1.000000	0.000000e+00
17	o_height	0.917212	0.980497	0.000000e+00

Drop features that have both MI and D scores less than 0.1. This implies dropping ['holiday', 'weekday', 'workingday'] from the bike dataset, dropping ['year'] from the suicide dataset and dropping ['b', 'o_bitrate', 'o_framerate'] from the video dataset. The average train and validation RMSE after performing 10-fold CV are:

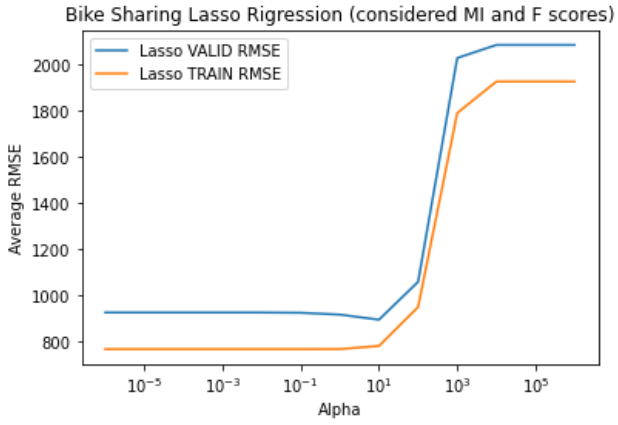
BIKE SHARING:
ORDINARY LEAST SQUARES:
Avg Train RMSE = 768.539
Avg Valid RMSE = 027.327

Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.000001	768.538669	927.326826
0.000010	768.538669	927.326684
0.000100	768.538669	927.325272
0.001000	768.538669	927.311151
0.010000	768.538703	927.170357
0.100000	768.542026	925.802692
1.000000	768.797490	915.066580
10.000000	777.294616	885.549747
100.000000	876.362275	990.153714
1000.000000	1331.296623	1522.979277
10000.000000	1799.550737	1970.179438
100000.000000	1912.901202	2073.149906
1000000.000000	1926.011959	2084.981847



For this case, best alpha = 10 (from the plots)

Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.000001	768.538669	927.326834
0.000010	768.538669	927.326766
0.000100	768.538669	927.326235
0.001000	768.538669	927.318901
0.010000	768.538733	927.225098
0.100000	768.543595	926.093562
1.000000	768.882529	918.118447
10.000000	782.255640	896.119430
100.000000	951.128394	1060.376043
1000.000000	1791.133860	2029.505532
10000.000000	1927.491983	2086.316569
100000.000000	1927.491983	2086.316569
1000000.000000	1927.491983	2086.316569

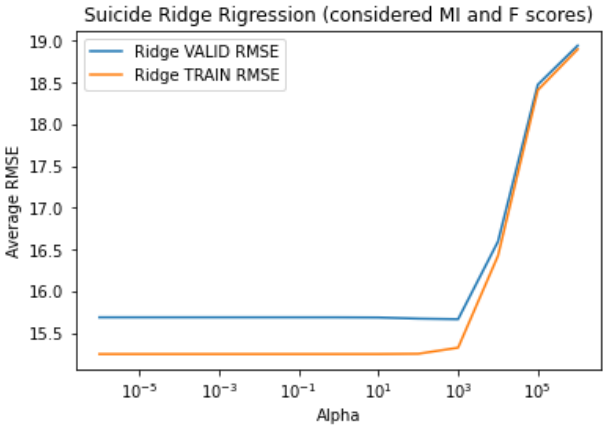


For LASSO too, best alpha = 10 (from the plots)

SUICIDE_DATASET:

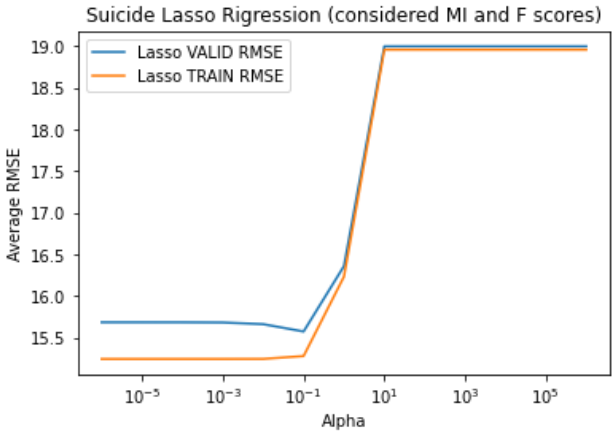
Ordinary Least Squares:
Avg Train RMSE: 15.247
Avg Test RMSE: 15.687

Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.000001	15.247070	15.686851
0.000010	15.247070	15.686851
0.000100	15.247070	15.686851
0.001000	15.247070	15.686850
0.010000	15.247070	15.686848
0.100000	15.247070	15.686828
1.000000	15.247071	15.686629
10.000000	15.247124	15.684726
100.000000	15.250336	15.671479
1000.000000	15.323062	15.664045
10000.000000	16.422457	16.593604
100000.000000	18.407356	18.473529
1000000.000000	18.896928	18.940543



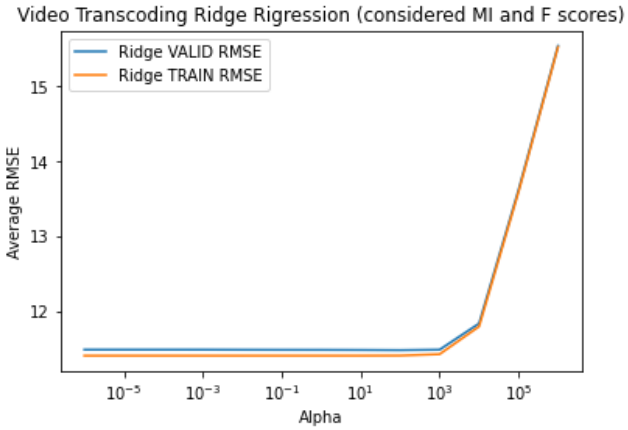
We see from the plots that the best alpha = 10^3

Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.000001	15.247070	15.686848
0.000010	15.247070	15.686826
0.000100	15.247070	15.686607
0.001000	15.247080	15.684504
0.010000	15.247734	15.664367
0.100000	15.281677	15.576465
1.000000	16.235252	16.361834
10.000000	18.959137	18.999758
100.000000	18.959137	18.999758
1000.000000	18.959137	18.999758
10000.000000	18.959137	18.999758
100000.000000	18.959137	18.999758
1000000.000000	18.959137	18.999758



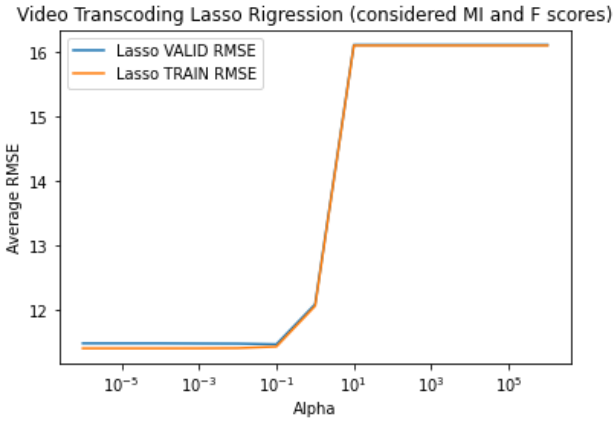
From the above plots, best alpha = 10^-1

VIDEO DATASET:



Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.000001	11.401138	11.481688
0.000010	11.401138	11.481685
0.000100	11.401138	11.481659
0.001000	11.401138	11.481422
0.010000	11.401143	11.480359
0.100000	11.401168	11.479431
1.000000	11.401187	11.478905
10.000000	11.401352	11.477571
100.000000	11.402624	11.475173
1000.000000	11.420896	11.481379
10000.000000	11.790868	11.831773
100000.000000	13.591885	13.617214
1000000.000000	15.529429	15.544868

Best alpha = 10^3



Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.000001	11.401176	11.479348
0.000010	11.401179	11.479294
0.000100	11.401188	11.479153
0.001000	11.401610	11.476482
0.010000	11.404468	11.474262
0.100000	11.427839	11.463634
1.000000	12.059405	12.086116
10.000000	16.106800	16.116994
100.000000	16.106800	16.116994
1000.000000	16.106800	16.116994
10000.000000	16.106800	16.116994
100000.000000	16.106800	16.116994
1000000.000000	16.106800	16.116994

Best alpha = 10^-1

The above results have to be treated with care, since they include categorical features which were scalar encoded for the purposes of feature selection. Let us remove the categorical features and observe the relationship w.r.t only the continuous features.

Bike Sharing

	Continuous features	mi	f_reg	P_values
0	temp	0.836662	0.981381	2.810622e-81
1	atemp	1.000000	1.000000	1.854504e-82
2	hum	0.099821	0.015467	6.454143e-03
3	windspeed	0.119986	0.087962	1.359959e-10

Suicide

	Continuous features	mi	f_reg	P_values
0	population	1.000000	0.107688	0.167021
1	gdp_for_year ($)	0.375296	1.000000	0.000026
2	gdp_per_capita ($)	0.327576	0.004999	0.765905

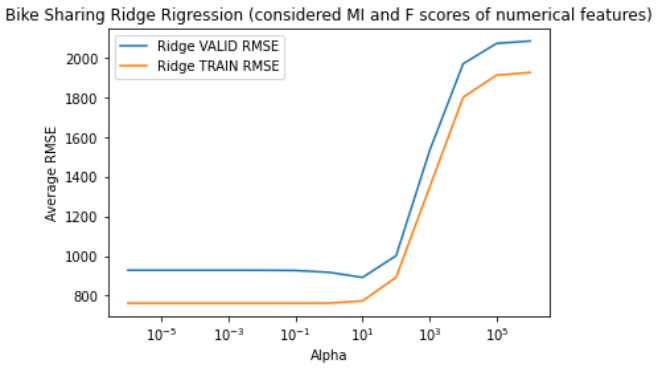
```
Video Transcoding
```

	Continuous features	mi	f_reg	P_values
0	duration	0.972518	0.000081	1.467846e-01
1	width	0.442772	0.045465	2.213683e-256
2	height	0.453664	0.044485	5.955923e-251
3	bitrate	0.987996	0.065419	0.000000e+00
4	framerate	0.556470	0.016789	1.889575e-96
5	i	0.906506	0.000906	1.238180e-06
6	p	1.000000	0.002925	3.047665e-18
7	b	0.009551	0.000070	1.776859e-01
8	frames	0.995296	0.002910	3.718494e-18
9	i_size	0.975284	0.011146	9.874803e-65
10	p_size	0.985954	0.025514	2.541518e-145
11	size	0.984795	0.025226	1.043438e-143
12	o_bitrate	0.059304	0.065660	0.000000e+00
13	o_framerate	0.040801	0.029005	8.053878e-165
14	o_width	0.944968	1.000000	0.000000e+00
15	o_height	0.951485	0.980497	0.000000e+00

BIKE DATASET:

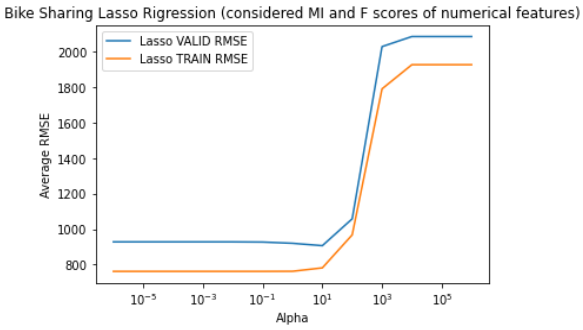
Bike Sharing Least Squares (considered MI and F scores of numerical features) Avg Train RMSE 761.819

Bike Sharing Least Squares (considered MI and F scores of numerical features) Avg Test RMSE 928.172



Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.000001	761.819063	928.171603
0.000010	761.819063	928.171484
0.000100	761.819063	928.170292
0.001000	761.819063	928.158381
0.010000	761.819105	928.039560
0.100000	761.823168	926.879972
1.000000	762.144492	917.539138
10.000000	773.448215	891.846448
100.000000	892.374408	1001.318995
1000.000000	1347.123881	1531.893523
10000.000000	1800.954195	1970.283525
100000.000000	1913.012915	2073.115256
1000000.000000	1926.022777	2084.977833

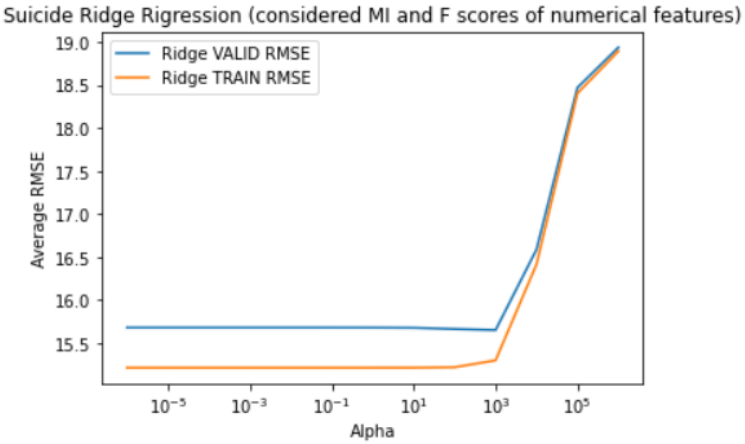
Best alpha = 10

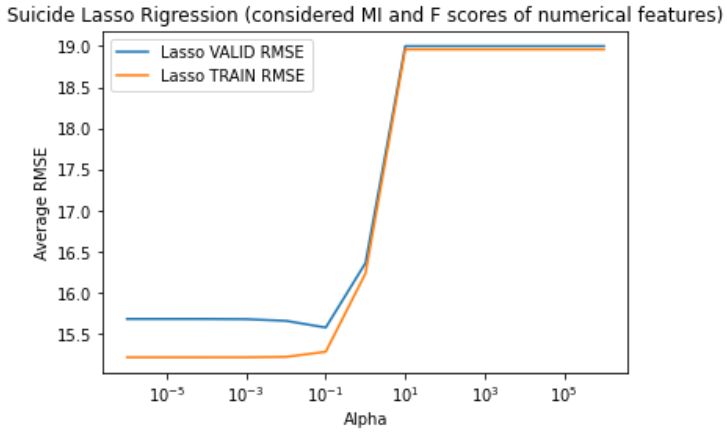


Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.000001	761.819063	928.171609
0.000010	761.819063	928.171546
0.000100	761.819063	928.170906
0.001000	761.819063	928.162908
0.010000	761.819134	928.084505
0.100000	761.825957	927.062447
1.000000	762.259632	920.149507
10.000000	781.560947	906.748109
100.000000	967.650054	1058.198619
1000.000000	1791.133860	2029.505532
10000.000000	1927.491983	2086.316569
100000.000000	1927.491983	2086.316569
1000000.000000	1927.491983	2086.316569

Best alpha = 10

SUICIDE DATASET:





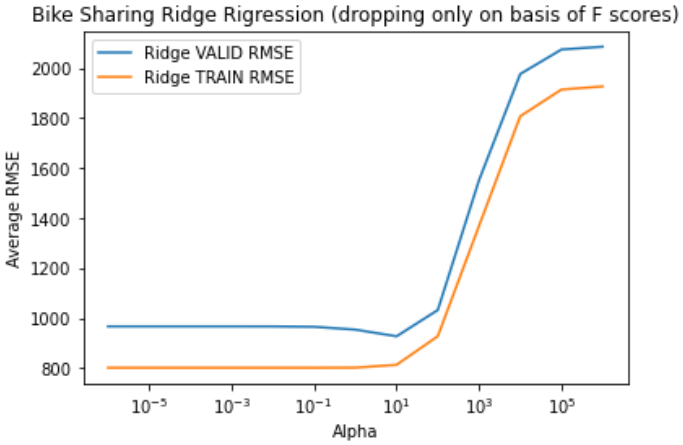
We noticed that for the Suicide dataset, the MI scores and F scores were very different. So, we tested by dropping features only on basis of F scores and the results briefly are:

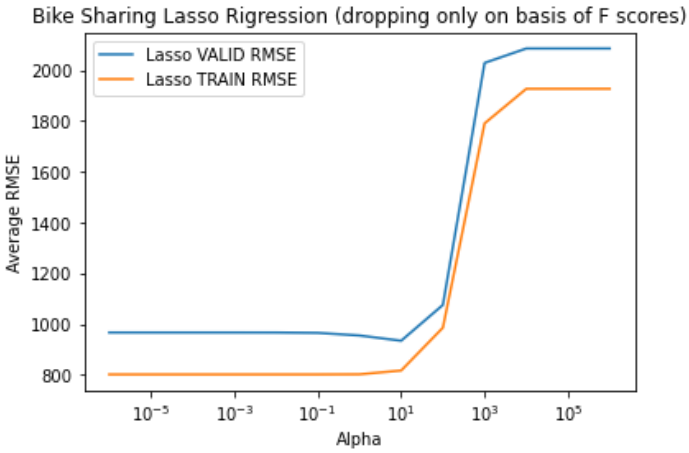
For **BIKE dataset**, dropped: ['holiday', 'weekday','workingday','hum','windspeed']

Bike Sharing Least Squares (dropping only on basis of F scores) Avg Train RMSE 801.513

Bike Sharing Least Squares (dropping only on basis of F scores) Avg Test RMSE 966.878

Similar curves to previous case:

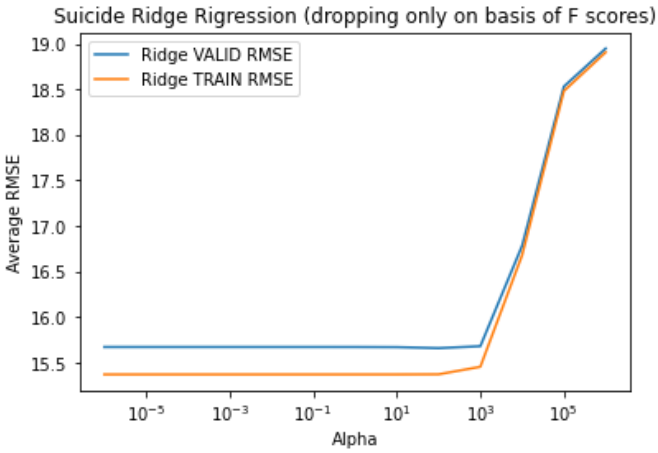


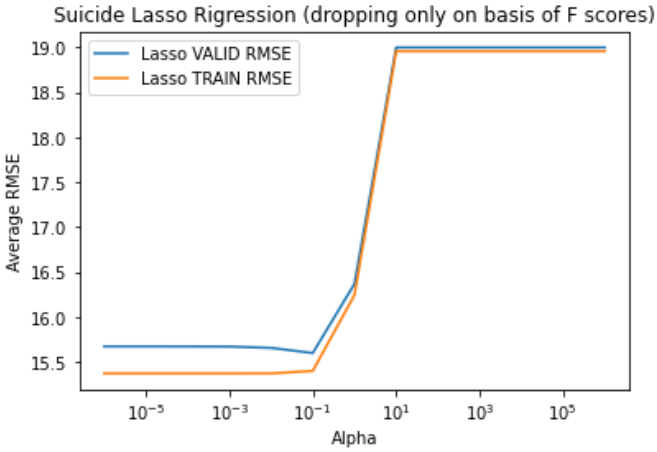


SUICIDE dataset

Dropped: ['year', 'population','gdp_for_year ($)','gdp_per_capita ($)','generation']

Still, similar results as in the above scenario were obtained.

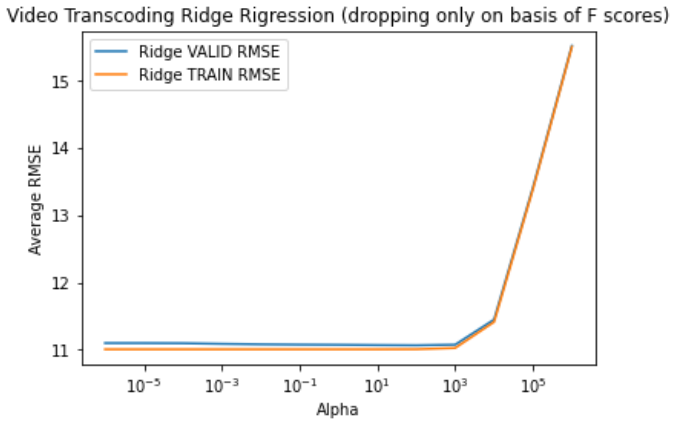


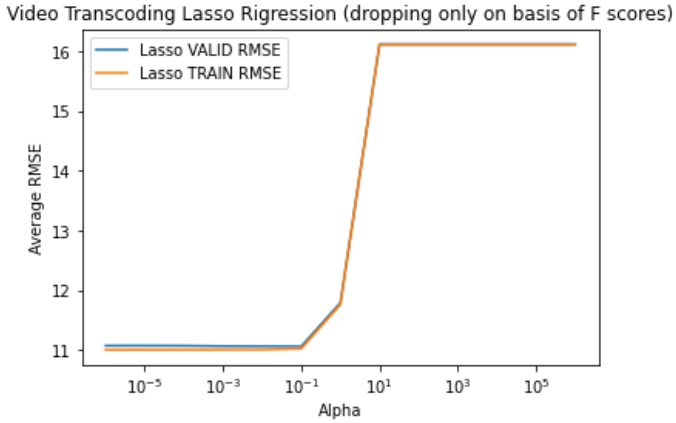


VIDEO dataset:

Dropped: ['duration', 'codec', 'i', 'b']. Still obtained good results:

Video Transcoding Least Squares (dropping only on basis of F scores) Avg Train RMSE 11.008
Video Transcoding Least Squares (dropping only on basis of F scores) Avg Test RMSE 11.101



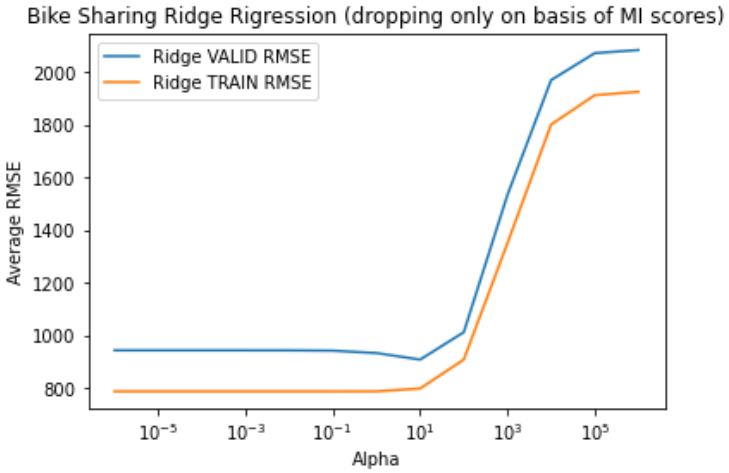


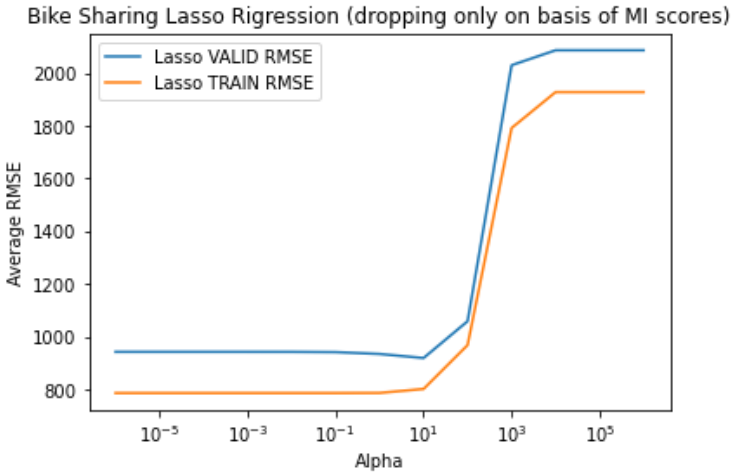
A similar experiment, but by dropping features only on the basis of MI scores (those with MI < 0.1)

BIKE dataset:

Dropping: ['holiday', 'weekday','workingday','hum']

Bike Sharing Least Squares (dropping only on basis of MI scores) Avg Train RMSE 785.720
Bike Sharing Least Squares (dropping only on basis of MI scores) Avg Test RMSE 941.971

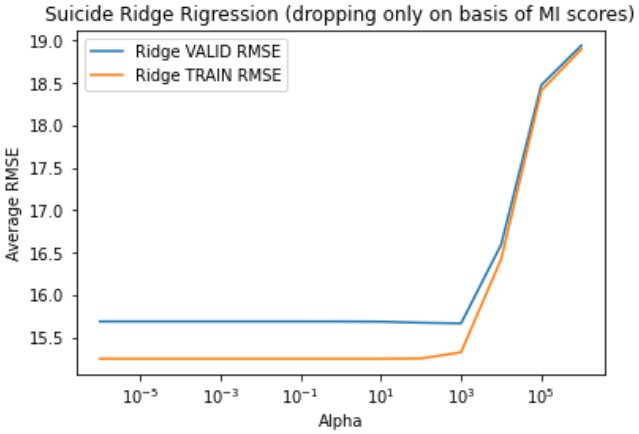


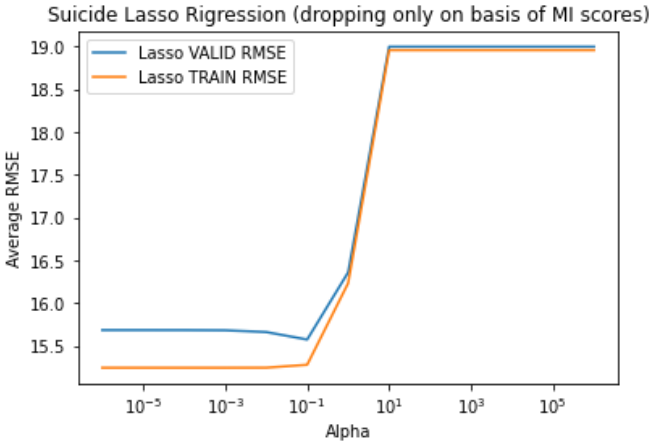


SUICIDE dataset:

Dropping ['year']

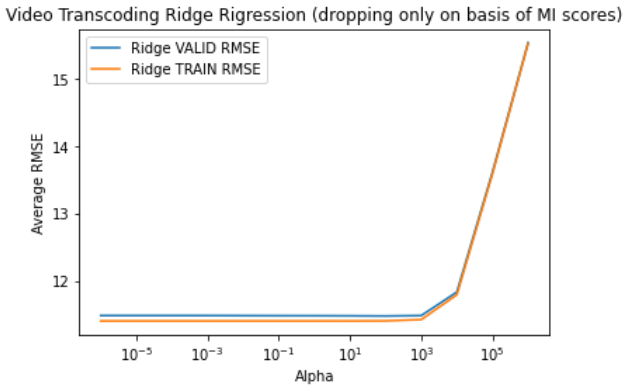
Suicide Least Squares (dropping only on basis of MI scores) Avg Train RMSE 15.247
Suicide Least Squares (dropping only on basis of MI scores) Avg Test RMSE 15.687

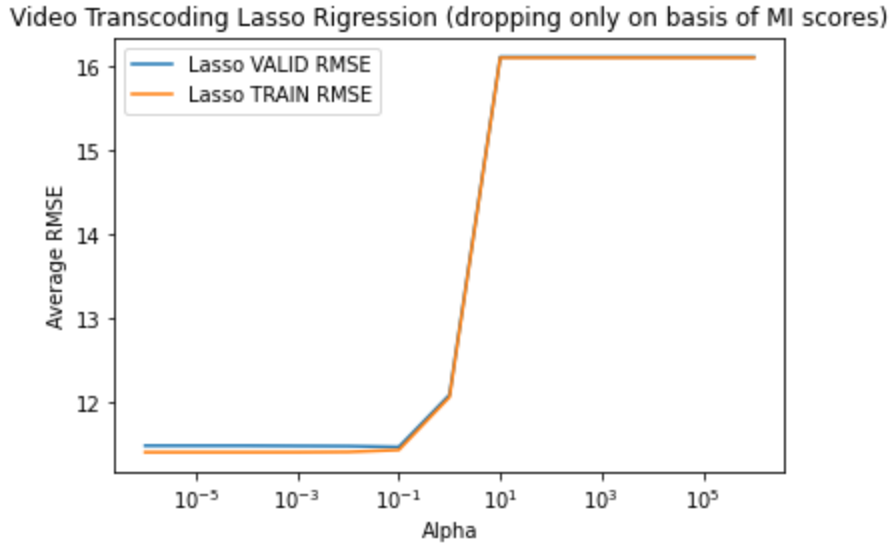




VIDEO dataset:

Dropping ['b', 'o_bitrate', 'o_framerate']





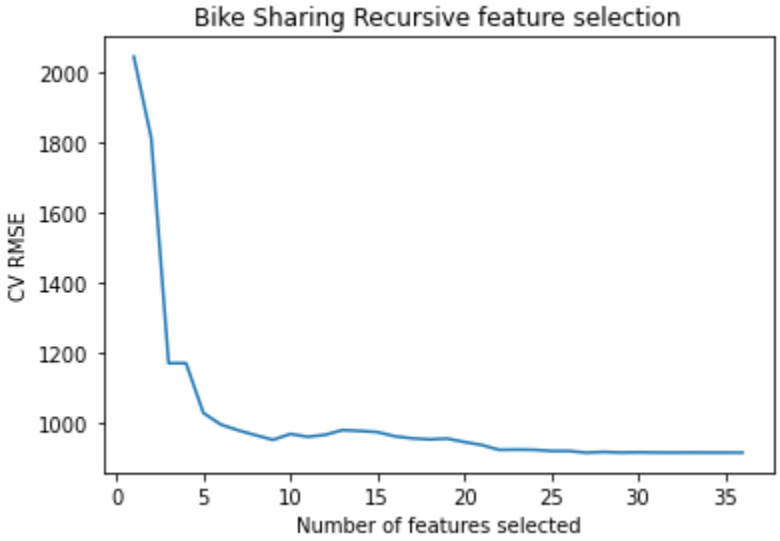
As a final experiment, recursive selection of features on the full dataset was performed, to give:

Recursive selection of the best features

BIKE DATASET:

We see that the optimal number of features is around 5.

Minimum CV rmse at number of features : 27
CV rmse: 914.856



The lowest CV rmse features are:
temp, atemp,hum, windspeed, season, month, holiday, weekday, workingday, weathersit.

One hot encoded:
['temp', 'atemp', 'hum', 'windspeed', 'season_1', 'season_4', 'yr_0', 'yr_1', 'mnth_1', 'mnth_2', 'mnth_5', 'mnth_6', 'mnth_7', 'mnth_9', 'mnth_11', 'mnth_12', 'holiday_0', 'holiday_1', 'weekday_0', 'weekday_1', 'weekday_2', 'weekday_6', 'workingday_0', 'workingday_1', 'weathersit_1', 'weathersit_2', 'weathersit_3']

SUICIDE DATASET:

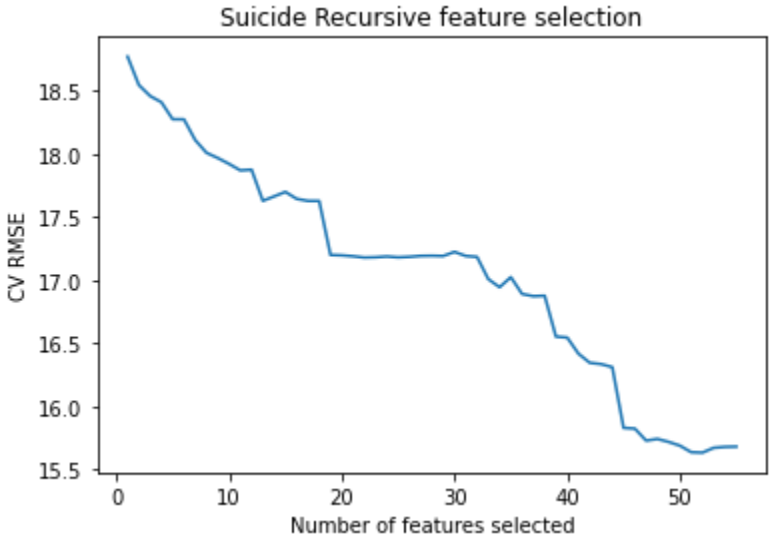
Lowest validation RMSE, number of features : 52

CV rmse: 15.633

Features selected at lowest CV RMSE:
year, sex,age,generation, and continent

['year_1985', 'year_1986', 'year_1987', 'year_1988', 'year_1989', 'year_1990', 'year_1991', 'year_1992', 'year_1993', 'year_1994', 'year_1995', 'year_1996', 'year_1997', 'year_1998', 'year_1999', 'year_2000', 'year_2001', 'year_2002', 'year_2003', 'year_2004', 'year_2005', 'year_2006', 'year_2007', 'year_2008', 'year_2009', 'year_2010', 'year_2011', 'year_2012', 'year_2013', 'year_2014', 'year_2015', 'year_2016', 'sex_female', 'sex_male', 'age_15-24 years', 'age_25-34 years', 'age_35-54 years', 'age_5-14 years', 'age_55-74 years', 'age_75+ years', 'generation_Boomers', 'generation_G.I. Generation', 'generation_Generation X', 'generation_Generation Z', 'generation_Millenials', 'generation_Silent', 'continent_AF', 'continent_AS', 'continent_EU', 'continent_NA', 'continent_OC', 'continent_SA'],

second best features:
'gdp_per_capita ($)'



Above plot shows that the suicide dataset requires a lot of features.

VIDEO DATASET

At validation RMSE, number of features : 24

CV rmse: 11.083

Optimal features selected at lowest Validation RMSE:

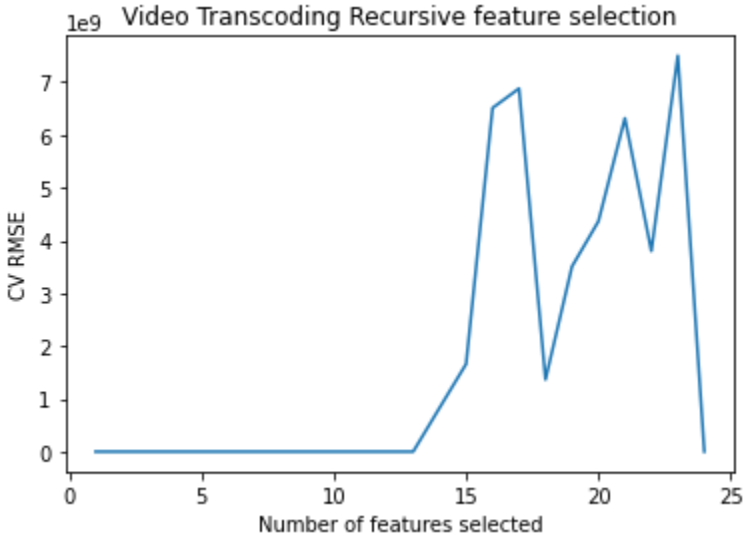
['duration', 'width', 'height', 'bitrate', 'framerate', 'i', 'p', 'b',

'frames', 'i_size', 'p_size', 'size', 'o_bitrate', 'o_framerate',

'o_width', 'o_height', 'codec_flv', 'codec_h264', 'codec_mpeg4',

'codec_vp8', 'o_codec_flv', 'o_codec_h264', 'o_codec_mpeg4',

'O_codec_vp8']



which shows that an optimal number would be about 5 features.

Q12 Does feature scaling play any role (in the cases with and without regularization)? Justify your answer.

Feature scaling is required and determines the kind of model obtained when we apply regularization. This is because a penalty is imposed on the magnitude of the weights and that depends on the associated features.

However, without regularization, feature scaling does not affect the model in case of Ordinary least squares since any scaling can be learnt by a suitable coefficient.

Q13 Some linear regression packages return p-values for different features. Interpret them.

The p values indicate the believability of the null hypothesis , given the sample statistic. The null hypothesis in this case would be that there is no relationship between the feature and the response. Thus low p-values implies that our observed sample statistic indicates a very low probability that the null was true. Thus, low p-values indicate that there is some relationship between the feature and the target and hence that feature must be included.

Q14 Look up for the most salient features and interpret them

For each dataset, features with degree 2 and 3 were computed before training the model. From this, the best degree was chosen. The MI and F scores of features were computed to get the salient features with as (decreasing magnitude of the scores):

BIKE DATASET:
Third degree polynomial features best explained this dataset (results in the next question)
Column name:

	col_name	col_rep
0	season	x0
1	yr	x1
2	mnth	x2
3	holiday	x3
4	weekday	x4
5	workingday	x5
6	weathersit	x6
7	temp	x7
8	atemp	x8
9	hum	x9
10	windspeed	x10

	feature	MI
151	x1^2 x8	1.000000
30	x1 x8	0.990671
150	x1^2 x7	0.960507
29	x1 x7	0.956465
24	x1 x2	0.950856
...	...	...
274	x3 x8^2	0.000000
273	x3 x7 x10	0.000000
272	x3 x7 x9	0.000000
271	x3 x7 x8	0.000000
0	1	0.000000

	feature	f_reg	pvals
9	x8	NaN	1.854504e-82
8	x7	NaN	2.810622e-81
89	x0 x1^2	NaN	7.678823e-76
13	x0 x1	NaN	7.678823e-76
2	x1	NaN	2.483540e-63
...	...	...	...
260	x3 x5 x6	NaN	NaN
261	x3 x5 x7	NaN	NaN
262	x3 x5 x8	NaN	NaN
263	x3 x5 x9	NaN	NaN
264	x3 x5 x10	NaN	NaN

This data shows that interactions between x1 and x8,ie, a combination of the year and the atemp value in that year is an important feature. Also important is the interaction between x0 and x1, ie, the season of a particular year.

SUICIDE DATASET:
Second degree features best explained this model (results in next question)

Column key:

	col_name	col_rep
0	year	x0
1	sex	x1
2	age	x2
3	population	x3
4	gdp_for_year ($)	x4
5	gdp_per_capita ($)	x5
6	generation	x6
7	continent	x7

	feature	MI
4	x3	1.000000
25	x2 x3	0.818274
30	x3^2	0.768640
34	x3 x7	0.763826
33	x3 x6	0.675059
19	x1 x3	0.585638
18	x1 x2	0.484141
3	x2	0.464899
26	x2 x4	0.461452
24	x2^2	0.459720
28	x2 x6	0.454680
29	x2 x7	0.448323
38	x4 x7	0.423222
31	x3 x4	0.422790
37	x4 x6	0.407669
20	x1 x4	0.396524
21	x1 x5	0.381208

	feature	f_reg	pvals
22	x1 x6	NaN	0.000000e+00
2	x1	NaN	0.000000e+00
24	x2^2	NaN	0.000000e+00
23	x1 x7	NaN	0.000000e+00
18	x1 x2	NaN	0.000000e+00
17	x1^2	NaN	0.000000e+00
10	x0 x1	NaN	0.000000e+00
3	x2	NaN	7.739320e-218
11	x0 x2	NaN	1.336458e-71
29	x2 x7	NaN	1.440259e-53
28	x2 x6	NaN	1.056041e-49
44	x7^2	NaN	1.630703e-22
43	x6 x7	NaN	3.815148e-20
7	x6	NaN	9.218445e-17
9	x0^2	NaN	1.819911e-16
8	x7	NaN	5.891869e-14
1	x0	NaN	7.353434e-11

Best features include interaction of other variables with x1 or x3, 0.ie , the various factors considered with the sex and population were cruicial features. X2, ie, Age was another important factor that interacts with x6 and x7,ie, with the generation and the continent.

VIDEO DATASET:
Degree 2 features were the best for this dataset.
Column key:

	col_name	col_rep
0	duration	x0
1	codec	x1
2	width	x2
3	height	x3
4	bitrate	x4
5	framerate	x5
6	i	x6
7	p	x7
8	b	x8
9	frames	x9
10	i_size	x10
11	p_size	x11
12	size	x12
13	o_codec	x13
14	o_bitrate	x14
15	o_framerate	x15
16	o_width	x16
17	o_height	x17

	feature	MI
97	x4 x16	1.000000
98	x4 x17	0.989692
160	x10 x16	0.973940
161	x10 x17	0.964778
168	x11 x17	0.959194
...	...	...
114	x6 x8	0.001267
136	x8 x9	0.001206
125	x7 x8	0.000500
60	x2 x8	0.000000
138	x8 x11	0.000000

	feature	f_reg	pvals
189	x17^2	NaN	0.000000
15	x14	NaN	0.000000
183	x14 x17	NaN	0.000000
178	x13 x16	NaN	0.000000
53	x1 x17	NaN	0.000000
...	...	...	...
158	x10 x14	NaN	0.914047
19	x0^2	NaN	0.915946
25	x0 x6	NaN	0.949275
24	x0 x5	NaN	0.962540
0	1	NaN	NaN

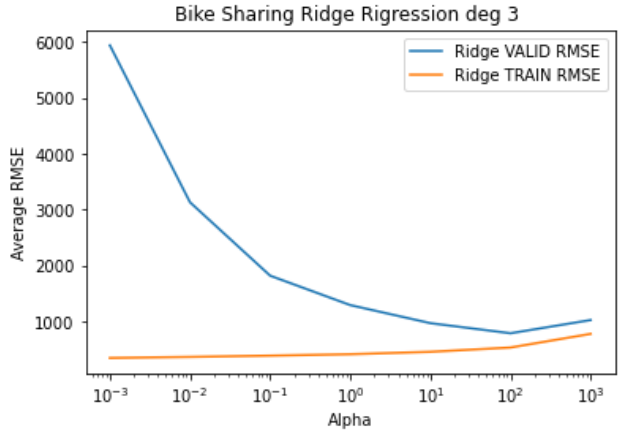
The best features were x4 x16 and x4 x17. This is highly informative as it says that the bitrate (x4) considered with the width and height (x 16 and x 17) are important features. Also important are x10 x16 and x10 x17 (showing the interaction of i_size with width and height)

Q15 What degree of polynomial is best? What reasons would stop us from too much increase of the polynomial degree? How do you choose that?

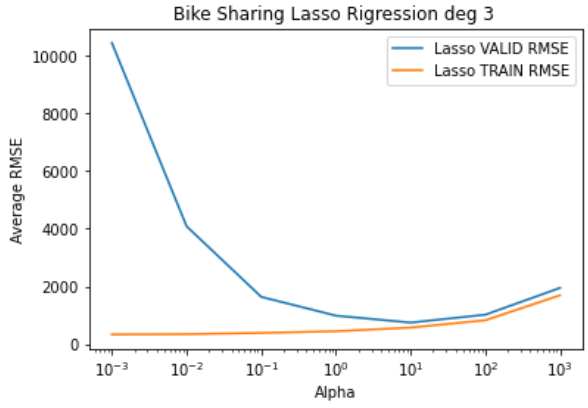
The best degree was found out by experimenting across degrees 2 and 3 and finding the model that results in the lowest CV average RMSE.

It is not advisable to increase the polynomial degree to a large number because it tends to overfit the training data (due to less bias present in the model)

BIKE DATASET:
For the BIKE dataset, Ridge regression's best model occurs at about alpha = 100 with degree 3 of polynomial features.
For Lasso, the best model was at degree 3 and alpha = 10

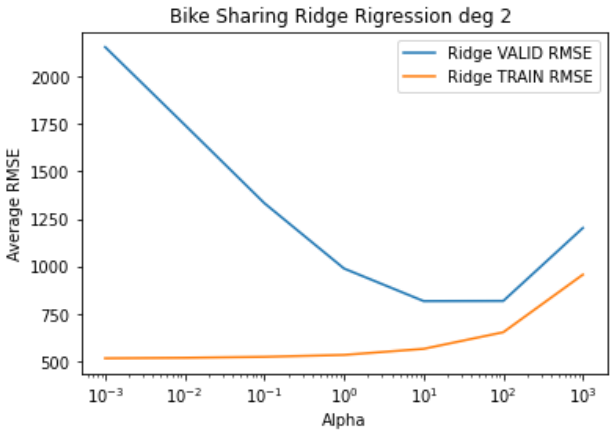


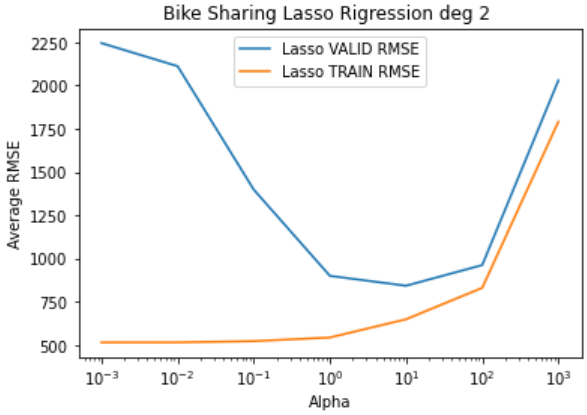
Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.001	342.731147	5943.150076
0.010	361.170004	3131.310821
0.100	383.727208	1816.218095
1.000	409.351020	1289.168173
10.000	452.490100	965.629355
100.000	529.271463	784.794129
1000.000	773.974237	1021.565979



Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.001	337.258718	10442.636292
0.010	346.922525	4086.819409
0.100	390.142132	1639.195859
1.000	448.612421	988.685071
10.000	573.901469	748.794123
100.000	829.449240	1022.074151
1000.000	1696.076213	1951.616949

The BIKE dataset's RMSE for degree 2 features:

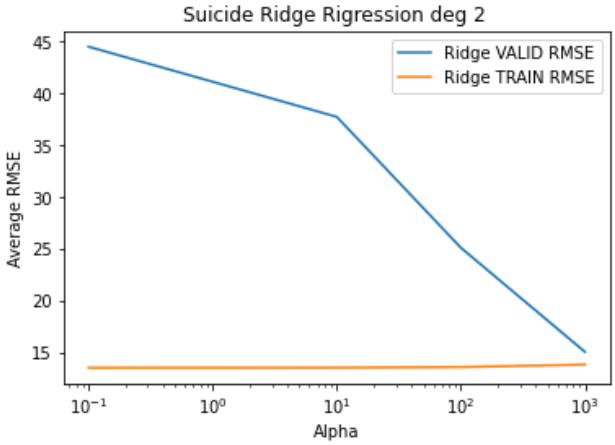




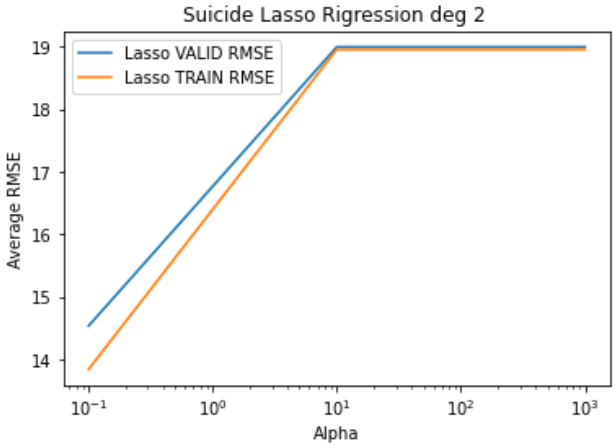
SUICIDE DATASET:

For Suicide dataset, RIdge regression's best model appears at alpha = 1000, degree 2.

Lasso: alpha = 0.1 , degree 2

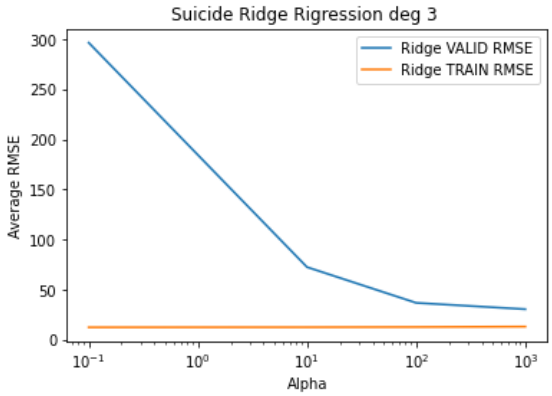


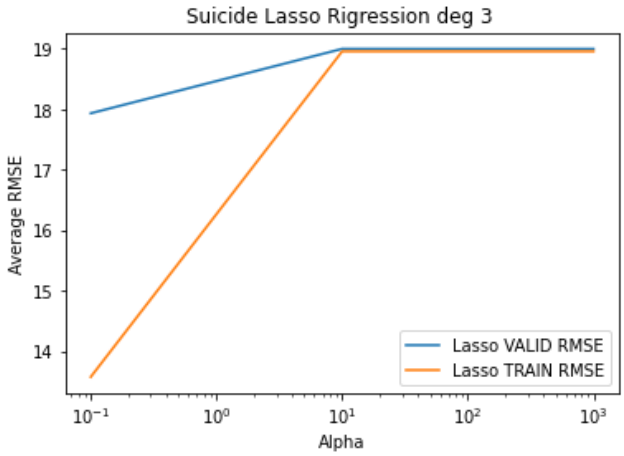
Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.1	13.492286	44.507807
10.0	13.501272	37.728411
100.0	13.559748	25.085427
1000.0	13.796183	15.018599



Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.1	13.842205	14.537597
10.0	18.959137	18.999758
100.0	18.959137	18.999758
1000.0	18.959137	18.999758

The SUICIDE datasets RMSE for degree 3 features was:

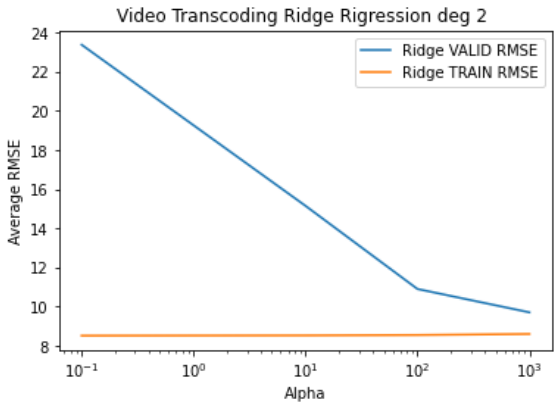


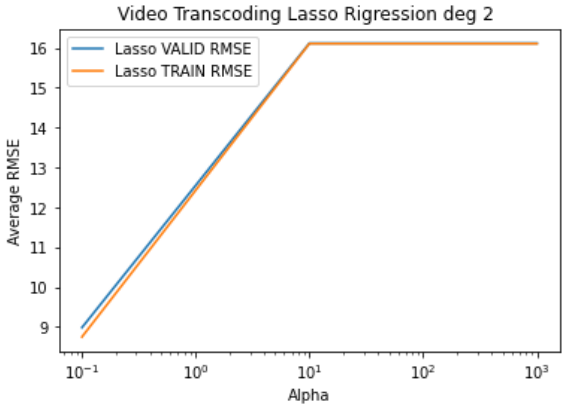


VIDEO DATASET:

For VIDEO dataset, Ridge regression: alpha = 1000 and deg = 2

Lasso: alpha = 0.1 and deg = 2

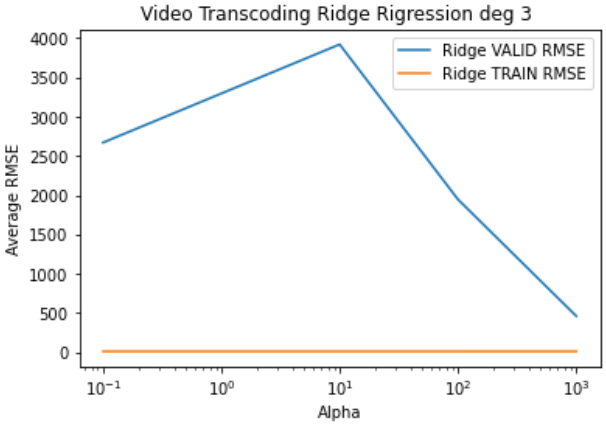


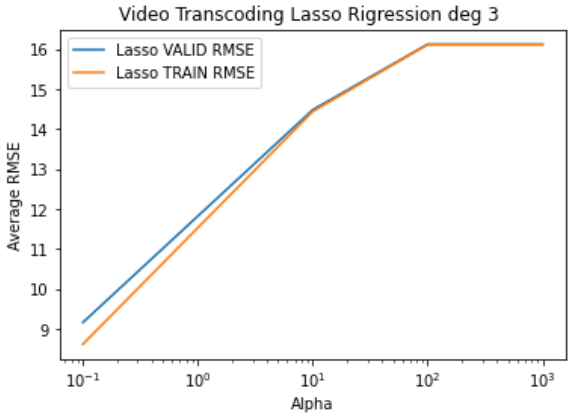


Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.1	8.519758	23.364192
10.0	8.531022	15.143170
100.0	8.551798	10.900243
1000.0	8.604859	9.709025

Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.1	8.751143	8.985448
10.0	16.106800	16.116994
100.0	16.106800	16.116994
1000.0	16.106800	16.116994

VIDEO datasets RMSE for degree 3 was:





Q16 For the transcoding dataset it might make sense to craft inverse of certain features such that you get features such as xixj/ xk , etc. Explain why this might make sense and check if doing so will boost accuracy

Numerical features could be inverted. The reciprocal transformation might give beatures that interact well with the rest of the features by forming xi * xj/ xk

For example, Frame rate could be inverted to seconds per frame. The new feature multiplied with i frames, p frames, b frames gives the duration of iframes ,p frames and b frames in the video, which might give a better understanding of the total transcoding time. The result of **inverting framerate:**

Video Transcoding Ridge Rigression deg 2

	Ridge Train RMSE	Ridge Valid RMSE
0	6.333415	14.185797

Video Transcoding Lasso Rigression deg 2

	Lasso Train RMSE	Lasso Valid RMSE
0	6.584652	7.288184

Similarly, if we invert the number of frames, 'frames' feature, then we could get features like iframes/ frames, p frames/frames, b frames/frames which give fraction of frames of each type present. The result of **inverting 'frames':**

Video Transcoding Ridge Rigression deg 2

	Ridge Train RMSE	Ridge Valid RMSE
0	6.322121	21.227671

Video Transcoding Lasso Rigression deg 2

	Lasso Train RMSE	Lasso Valid RMSE
0	6.563849	7.484193

An experiment was tried where both ‘frames’ and ‘framerate’ were inverted to give results:

Video Transcoding Ridge Rigression deg 2

	Ridge Train RMSE	Ridge Valid RMSE
0	6.32069	8.311974

Video Transcoding Lasso Rigression deg 2

	Lasso Train RMSE	Lasso Valid RMSE
0	6.569238	7.134968

Q17 Why does it do much better than linear regression?

Neural networks have a non-linear transformation at each hidden layer that enables them to learn non-linear relationships from while training. Hence, they do better than linear regression.

Q18 Adjust your network size (number of hidden neurons and depth), and weight decay as regularization. Find a good hyper-parameter set systematically

Neural networks' performance depends on the hyperparameters. It is general practice to run a model several times to decide on its true nature. The ReLU activation was used for each hidden layer and adam was the optimizer used to perform gradient descent.
For these datasets, a range of hyperparameters were tested: the number of hidden layers and units in each layer, the learning rate, and the weight decay. It must be noted that a 10 fold cross validation was performed before reporting the errors.

BIKE DATASET:
The best set of hyper parameters are as follows (listed from best model first):

Hidden Size	Avg Valid RMSE	Avg Train RMSE	LR	Alpha
[200, 500, 500, 200]	790.181225	478.767594	0.005	0.1
[300, 600, 600, 600, 200]	798.274542	490.637614	0.0001	100
[300, 600, 600, 600, 200]	808.869003	521.34453	0.0001	1000
[300, 600, 600, 600, 200]	811.689381	513.925067	0.005	0.1
[300, 600, 600, 600, 200]	812.559547	470.328962	0.0001	0.1
[300, 600, 600, 600, 200]	814.254005	441.076267	0.005	1000
[300, 600, 600, 600, 200]	814.950618	508.141826	0.0001	10
[300, 600, 600, 600, 200]	820.352491	450.234021	0.005	10
[200, 500, 500, 200]	832.639317	679.320955	0.0001	1000
[300, 500, 200]	836.028609	501.176772	0.1	100
[200, 500, 500, 200]	843.677566	678.420703	0.0001	100
[300, 500, 200]	846.941875	646.715115	0.005	1000
[200, 500, 500, 200]	850.742015	631.340909	0.1	10
[300, 500, 200]	851.401551	653.799133	0.005	100
[200, 500, 500, 200]	857.497725	554.731027	0.005	10
[300, 500, 200]	862.294441	700.119869	0.005	10
[200, 500, 500, 200]	867.975918	513.090648	0.005	100
[300, 500, 200]	868.017783	460.029352	0.1	1000
[300, 600, 600, 600, 200]	871.172024	440.256064	0.005	100
[200, 500, 500, 200]	872.668926	709.27799	0.0001	10
[300, 500, 200]	890.704954	757.110922	0.0001	1000
[200, 500, 500, 200]	892.504273	901.425112	0.0001	0.1
[200, 500, 500, 200]	898.274294	592.051691	0.005	1000
[300, 500, 200]	901.896137	754.942178	0.1	10
[300, 500, 200]	910.431761	765.50343	0.0001	100
[300, 500, 200]	913.062202	658.613518	0.005	0.1
[200, 500, 500, 200]	928.239549	722.819133	0.1	0.1
[200, 500, 500, 200]	930.446682	545.482235	0.1	1000
[200, 500, 500, 200]	942.211655	589.915409	0.1	100
[300, 500, 200]	945.991572	771.994056	0.0001	0.1
[300, 600, 600, 600, 200]	964.630918	552.493932	0.1	1000
[300, 600, 600, 600, 200]	1010.349786	623.197229	0.1	100
[300, 500, 200]	1066.346358	856.686661	0.0001	10
[300, 500, 200]	1236.097119	1201.043316	0.1	0.1
[300, 600, 600, 600, 200]	1262.282642	842.981638	0.1	10
[300, 600, 600, 600, 200]	1884.979581	1528.835868	0.1	0.1

VIDEO dataset:

The best hyperparameters are (best model first):

[300, 500, 200]	5.303775	3.919869	0.005	100
[300, 600, 600, 600, 200]	5.326107	3.463364	0.005	100
[200, 500, 500, 200]	5.47252	3.574046	0.005	100
[300, 500, 200]	5.474537	4.048715	0.0001	100
[200, 500, 500, 200]	5.587379	2.989653	0.005	10
[300, 600, 600, 600, 200]	5.612401	3.017464	0.005	10
[300, 500, 200]	5.666926	2.946742	0.005	10
[300, 600, 600, 600, 200]	5.68787	2.651244	0.005	0.1
[300, 500, 200]	5.711178	2.671341	0.005	0.1
[200, 500, 500, 200]	5.759734	2.665932	0.005	0.1
[200, 500, 500, 200]	5.803174	3.409466	0.0001	100
[300, 600, 600, 600, 200]	5.890392	2.751719	0.0001	10
[300, 600, 600, 600, 200]	5.989316	5.198434	0.0001	1000
[300, 600, 600, 600, 200]	6.001243	3.174738	0.0001	100
[200, 500, 500, 200]	6.010435	2.701074	0.0001	0.1
[300, 600, 600, 600, 200]	6.018228	2.665176	0.0001	0.1
[300, 600, 600, 600, 200]	6.047226	5.496345	0.005	1000
[300, 500, 200]	6.048427	3.027596	0.0001	10
[200, 500, 500, 200]	6.057106	2.843946	0.0001	10
[200, 500, 500, 200]	6.285845	5.648885	0.0001	1000
[300, 500, 200]	6.286149	2.390788	0.0001	0.1
[200, 500, 500, 200]	6.344535	5.80822	0.005	1000
[300, 500, 200]	6.742069	6.162259	0.1	100
[300, 500, 200]	6.825335	6.490337	0.005	1000
[300, 500, 200]	6.839218	6.413917	0.0001	1000
[300, 500, 200]	7.117106	5.862018	0.1	0.1
[300, 500, 200]	7.741897	8.28193	0.1	1000
[300, 500, 200]	8.565896	7.027464	0.1	10

[200, 500, 500, 200]	9.565461	9.334768	0.1	1000
[200, 500, 500, 200]	10.925547	10.528532	0.1	0.1
[200, 500, 500, 200]	13.17706	13.116358	0.1	100
[200, 500, 500, 200]	13.599626	13.408736	0.1	10
[300, 600, 600, 600, 200]	15.485387	15.217684	0.1	0.1
[300, 600, 600, 600, 200]	15.815059	15.684552	0.1	100
[300, 600, 600, 600, 200]	15.913271	15.720486	0.1	10
[300, 600, 600, 600, 200]	16.060417	16.162939	0.1	1000

SUICIDE dataset:

Suicide MLP

	Hidden Size	Avg Valid RMSE	Avg Train RMSE	LR	Alpha
0	[200, 500, 500, 200]	158.709368	109.620041	0.005	0.01
1	[200, 500, 500, 200]	154.103722	106.100199	0.005	10.00
2	[200, 500, 500, 200]	142.941868	104.676267	0.005	100.00
3	[200, 500, 500, 200]	180.350828	184.202084	0.005	1000.00
4	[300, 600, 600, 600, 200]	146.117978	67.098693	0.005	0.01
5	[300, 600, 600, 600, 200]	135.497625	79.403298	0.005	10.00
6	[300, 600, 600, 600, 200]	140.504993	110.779732	0.005	100.00
7	[300, 600, 600, 600, 200]	171.475260	173.471465	0.005	1000.00

Q19 What activation function should be used for the output? You may use none

Sklearn implementation for this report does not use any activation,ie, the activation is an identity transformation for the output layer. It is general practice to use ReLU for each hidden layer. Other popular activation functions are tanh and logistic (used especially for multi-class classification in the output layer)

Q20 What reasons would stop us from too much increase of the depth of the network?

As the depth of the network increases we come across the problem of gradient vanishing during back propagation which could prevent any sort of training from happening. Batch normalization and Residual networks are some techniques that alleviate this problem.

Q21

Apply a random forest regression model on datasets, and answer the following. Fine-tune your model. Explain how these hyper-parameters affect the overall performance? Do some of them have regularization effect?

ANSWER

For each of the dataset, I first use BayesSearchCV to find the optimal parameter values, and then run tests around that parameter value.

For a parameter test, I kept other parameters to optimal, and changed the current parameter to see how it affected RMSE score. For instance, if I want to test max_features, I would keep n_estimators and max_depth to the optimal value.

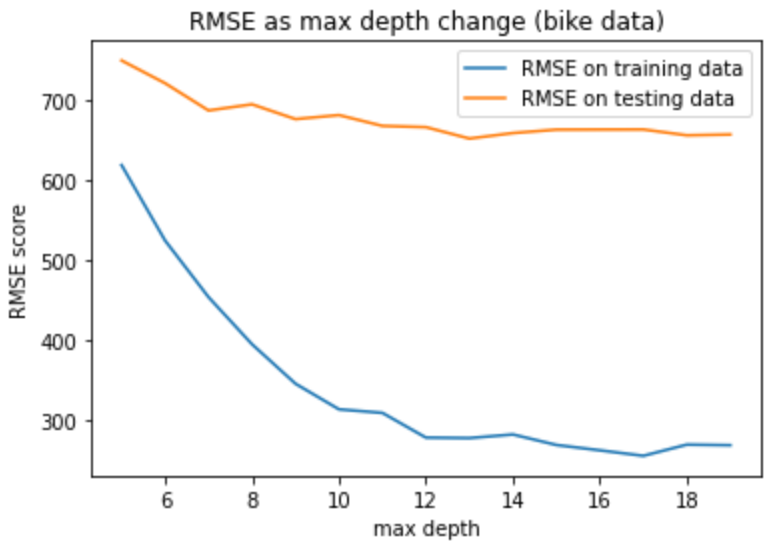
we draw the effect of each value out, on the graphs below. x-axis are the values for parameters and y-axis are RMSE scores. We have 3 parameters so 3 graphs for each dataset, here we only show the graph for the bike dataset below. only 3 lines of code change need to draw other datasets.

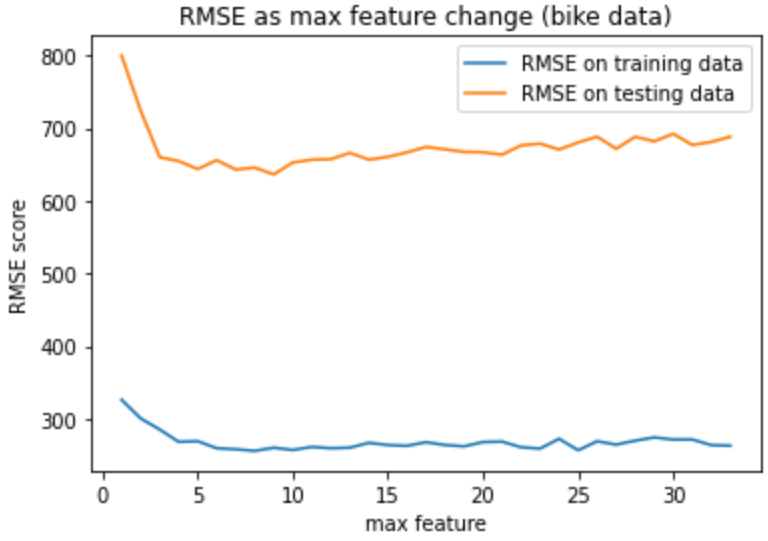
How parameters affect performance: overall, if we start at a small value for a parameter and slowly increase it, the performance would improve. However, the improvement would stop at some point (as shown in graphs below). Around that point, we would get the best result. Best parameter for each dataset and corresponding RMSE shown below

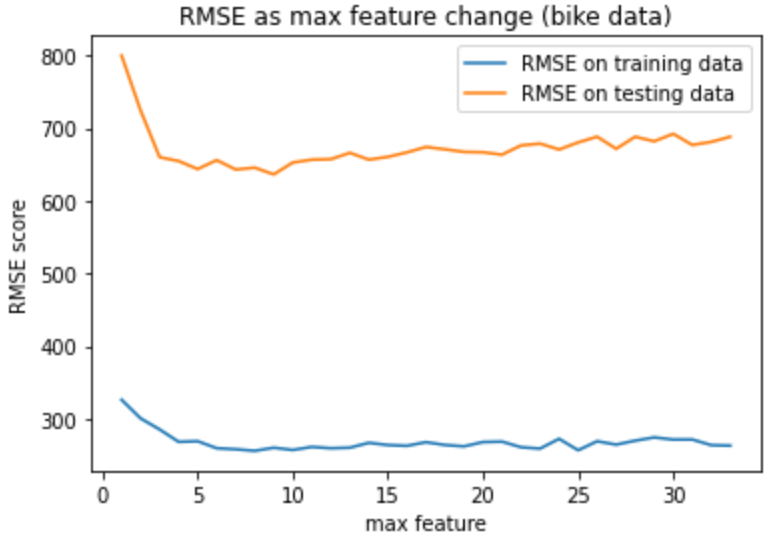
	number of estimators	max depth	max features	RMSE
Bike dataset	106	18	11	812.71
Suicide dataset	81	10	5	13.85
Video dataset	136	11	18	4.26

table for best parameter value corresponding to each dataset

Regularization effect: max depth have the greatest regularization effect, since as we can see from RMSE as max depth change plots, smaller max depth would bring train and test RMSE closer


RMSE as max depth change (bike data)
RMSE on training data
RMSE on testing data
RMSE score
max depth
700
600
500
400
300
6
8
10
12
14
16
18


RMSE as max feature change (bike data)
RMSE on training data
RMSE on testing data
RMSE score
max feature
800
700
600
500
400
300
0
5
10
15
20
25
30


RMSE as max feature change (bike data)
RMSE on training data
RMSE on testing data
RMSE score
max feature
800
700
600
500
400
300
0
5
10
15
20
25
30

Q22

Why does random forest perform well?

ANSWER

Random forest is like doing feature selection by forming feature subsets, and this can be very effective when we have many features in the data. Furthermore, random forest uses multiple decision trees, which can better capture patterns in data. Also it searches a random subset of features instead of the best feature, which adds randomness into the model.

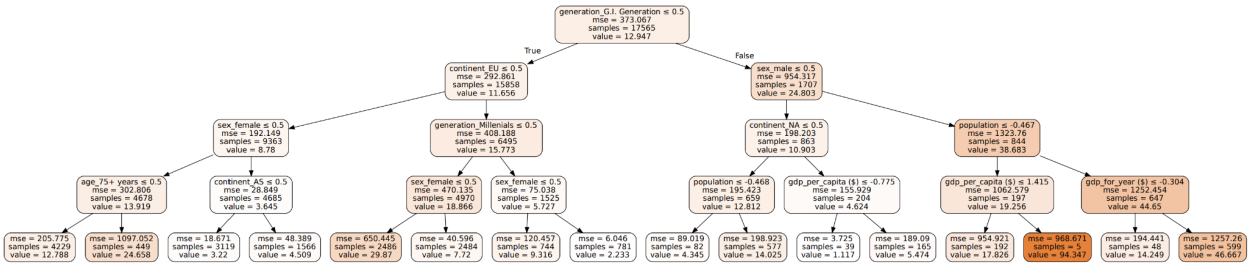
Q23

Randomly pick a tree in your random forest model (with maximum depth of 4) and plot its structure. Which feature is selected for branching at the root node? What can you infer about the importance of features? Do the important features match what you got in part 3.2.1?

ANSWER

G.I. Generation is selected at root node, which means this feature is probably very important if not the most important. Feature such as continent EU/NA/AS, population, GDPper capita, GDP for year, sex male/female, age 75+ are also selected. These features are probably more important among all the features.

Since the tree is so wide, please zoom in to see the details



This concurs with the results from section 3.2.1 where at the end of question 11 in this report, important features for each dataset were listed (by regression via sklearn recursive feature selection) among **sex and generation, and continent** for the SUICIDE dataset. This agrees with the features selected at the root node and also at the branches closest to the root, confirming our results.

Q24

Read the documentation of LightGBM and CatBoost and experiment on the picked dataset to determined the important hyperparameters along with a proper search space for the tuning of these parameters

ANSWER

We choose video transcoding time dataset for Q 24, 25, 26
For LightGBM
We pick important parameters base on the parameter tuning section suggestions, here are the parameter we choice

1. **num_leaves** -- the main parameter that controls the complexity of the model
2. **max_depth** -- max depth of the tree
3. we also throw in **n-estimators** since it does have large effect base on our experience

For CatBoost
We pick important parameters base on the parameter tuning section suggestions, here are the parameter we choice

1. **iterations** -- number of tree used, determine overfitting and underfitting
2. **learning_rate** -- learning rate of the model
3. **depth** -- the depth of each tree

Q25

Apply Bayesian optimization using skopt.BayesSearchCV from scikit-optmize to search good hyperparameter combinations in your search space. Report the best hyperparameter found and the corresponding RMSE, for both algorithms.

ANSWER

The best hyper parameter for LightGBM is show below (RMSE: 3.72)

max depth	number of estimator	number of leaves
8	220	28

The best hyper parameter for CatBoost is show below (RMSE: 4.79)

depth	iterations	learning rate
6	12	1

Q26

Interpret the effect of the hyperparameters using the Bayesian optimization results: Which of them helps with performance? Which helps with regularization (shrinks the generalization gap)? Which affects the fitting efficiency? Endorse your interpretation with numbers and visualizations.

ANSWER

Performance:

LightGBM:
To determine which parameters are important, I rank all the tests that BayesSearch did by RMSE score (shown in next page). Max depth and number of leaves seem to be two biggest influencing factors.

- Max depth: 9 out of 10 top performing models have max depth of 8, and all least performing models have max depth of 4 which is also the lowest max depth number. (mark in red)
- Number of leaves: 7 out of 10 top performing models have 28 leaves, and 9 out of 10 least performing models have 30+ leaves. (mark in blue)

Catboost:
To determine which parameters are important, I rank all the tests that BayesSearch did by RMSE score (shown in next page, after removing redundant combinations, only 24 unique left). Learning rate is the biggest influencing factor.

- Learning rate: all top performing models have a learning rate of 1. All least performing model have learning rate of 0.01

LightGBM Performance Ranking

rank	max depth	number of estimators	number of leaves
1	8	220	28
2	8	220	28
3	8	200	28
4	8	240	28
5	6	240	30
6	8	280	28
7	8	220	28
8	8	220	28
9	8	200	24
10	8	200	30
11	8	220	28
12	8	220	22
13	8	200	26
14	-1	240	28
15	8	200	28
15	8	200	28
17	8	240	30
18	6	240	30
19	6	240	30
20	-1	220	32
21	8	220	28
22	6	240	30
23	6	260	30
24	8	240	26
25	-1	220	26
26	8	200	20
27	8	240	36
28	6	240	32
29	8	200	40
30	6	240	30
31	6	220	28
32	6	260	30
33	-1	280	24
34	8	220	24
35	8	280	22
36	6	240	30
37	8	280	34
38	6	280	38
39	-1	300	30
40	6	240	30
41	-1	200	36
42	-1	260	38
43	8	260	38
44	6	240	30
45	6	240	36
46	8	300	32
47	4	260	32
48	4	260	38
49	4	200	22
50	4	220	36

Catboost Performance Ranking

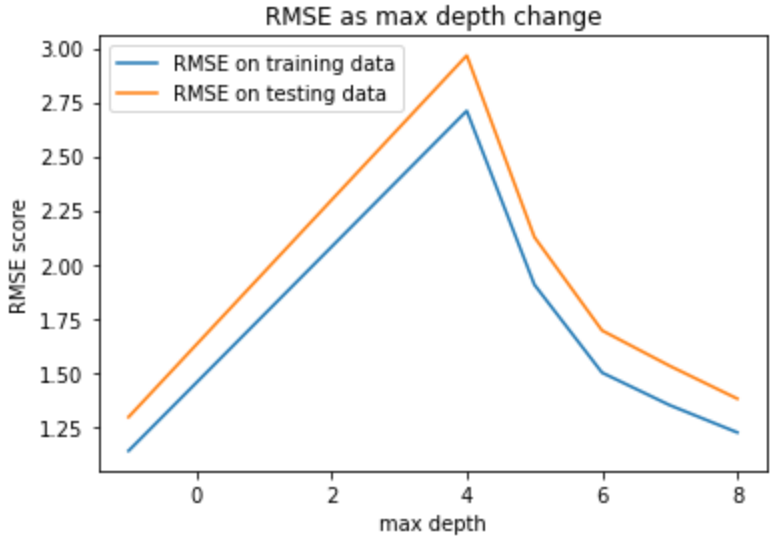
rank	depth	iterations	learning rate
1	6	12	1
2	6	16	1
3	6	14	1
4	8	14	1
5	6	10	1
6	8	10	1
7	8	12	1
8	8	16	1
9	6	20	1
10	6	18	1
11	8	18	1
12	7	16	1
13	7	14	1
14	4	12	1
15	7	18	0.1
16	7	18	0.1
17	8	16	0.1
18	6	18	0.1
19	7	16	0.1
20	7	12	0.1
21	4	16	0.1
22	6	20	0.01
23	4	18	0.01
24	7	14	0.01

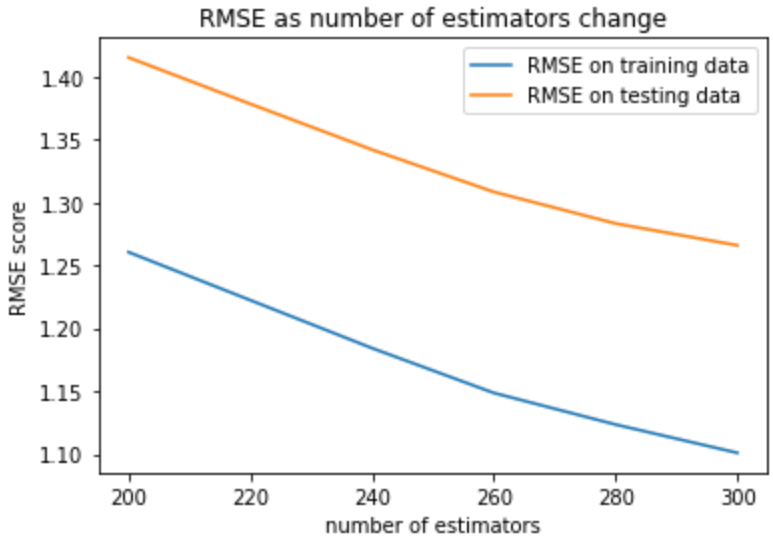
Regularization

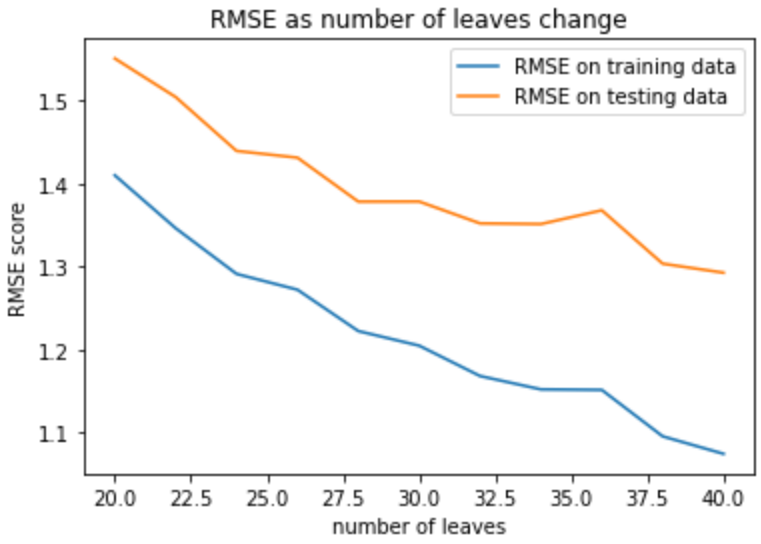
I split the total data into training and testing dataset. To test the regularization effect of a parameter, I kept other parameters to optimal, and changed the current parameter to see how it affected RMSE score difference between test dataset and training dataset. For instance, if I want to test max_depth, I would keep n_estimators and num_leavesto the optimal value. The parameter values and their corresponding scores are shown in the graphs below

LightGBM

As we can see, the training and testing score difference changes the most with the number of leaves. Less leave means less score difference, which means better generalization

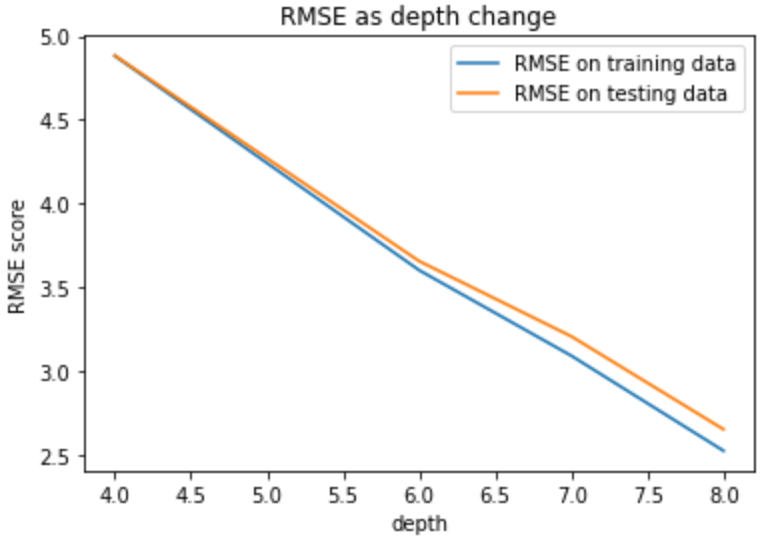


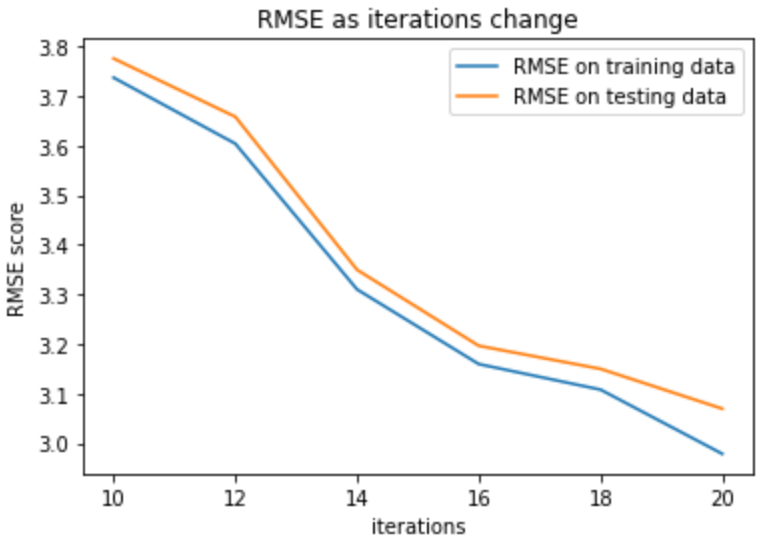


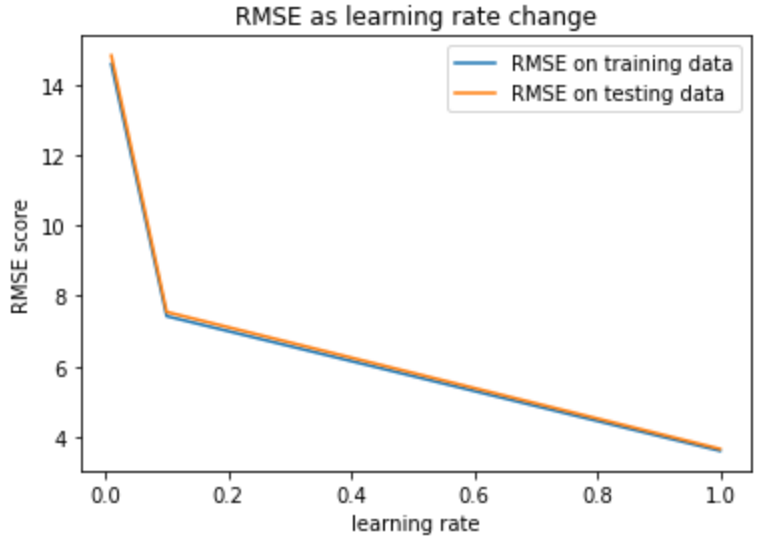


CatBoost

As we can see, the training and testing score difference changes the most with the depth. Less depth means less score difference, which means better generalization







(next page for fitting efficiency analysis)

Fitting efficiency

LightGBM:
To determine which parameters are important, I rank all the tests that BayesSearch did by mean fitting time (shown in next page). Number of estimators and number of leaves seem to be two biggest influencing factors.

- Number of estimators: all fast models models have 200 or 220 estimators (least among estimator tested), and all slow models have above 260 estimators (mark in red)
- Number of leaves: 8 out of 10 fastest models have less than 30 leaves, all slow models have more than 30 leaves. (mark in blue)

CatBoost:
To determine which parameters are important, I rank all the tests that BayesSearch did by mean fitting time (shown in next page). Depth and iterations seem to be two biggest influencing factors.

- Depth: all model with depth of 4 (red) are in top 10 fast, and most of model with depth of 8 (blue) are slow (3 model ranked slowest 4)
- Iterations : most model with few iterations (10 or 12, red) are fast, and most of model with large iterations (18 or 20, blue) are slow (4 model ranked slowest 6)

LightGBM Speed Ranking

time rank	max depth	number of estimators	number of leaves
1	8	200	24
2	8	200	20
3	4	200	22
4	8	200	28
5	8	200	28
6	4	220	36
7	8	200	26
8	8	220	22
9	8	220	24
10	8	200	30
11	-1	200	36
12	8	220	28
13	-1	220	26
14	8	200	28
15	8	220	28
16	-1	220	32
17	6	220	28
18	8	200	40
19	8	220	28
20	4	260	38
21	-1	240	28
22	8	220	28
23	4	260	32
24	8	220	28
25	8	240	26
26	6	240	30
27	6	240	30
28	6	240	30
29	8	240	28
30	8	220	28
31	6	240	30
32	8	240	30
33	6	240	32
34	6	240	30
35	8	280	22
36	6	240	30
37	6	240	30
38	-1	280	24
39	6	240	30
40	8	240	36
41	8	280	28
42	6	240	36
43	6	260	30
44	8	260	38
45	6	260	30
46	-1	300	30
47	8	300	32
48	-1	260	38
49	6	280	38
50	8	280	34

Catboost Speed Ranking

rank	depth	iterations	learning rate
1	4	12	1
2	6	10	1
3	6	12	1
4	4	16	0.1
5	6	14	1
6	4	18	0.01
7	7	12	0.1
8	6	16	1
9	7	14	1
10	8	10	1
11	7	14	0.01
12	6	18	0.1
13	6	18	1
14	8	12	1
15	7	16	0.1
16	7	16	1
17	6	20	1
18	8	14	1
19	7	18	0.1
20	7	18	0.1
21	8	16	0.1
22	8	16	1
23	8	18	1
24	6	20	0.01

Q27

Perform 10-fold cross-validation and measure average RMSE errors for training and validation sets. Why is the training RMSE different from that of validation set?

ANSWER

Because overfitting. The model is too closely fit with the training set. It captures the characteristics of the training set that cannot be generalized.

Q28

For random forest model, measure “Out-of-Bag Error” (OOB) as well. Explain what OOB error and R2 score means given this link.

ANSWER

Here Out-of-Bag score for each dataset are obtained using best parameter

- Bike dataset Out-of-Bag score 0.88
- Suicide dataset Out-of-Bag score 0.64
- Video dataset Out-of-Bag score 0.97

Out-of-Bag Error is calculated using R2 error, but it will have Out-of-Bag data instances. For each instance d, it would: 1. pick tree that’s not trained by d. 2.make prediction using that tree. 3. calculate R2 error

Code

Following pages contain jupyter notebook code for the project, which is roughly divided into 4 parts. Each part has detailed explanations and comments.

In [ ]:
```
from google.colab import drive
drive.mount("/content/drive")
#drive.mount("/content/drive", force_remount=True)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive",
force_remount=True).
```

In [ ]:
```
from IPython.display import display
from tqdm.notebook import tqdm
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd

tqdm.pandas()
```

```
/usr/local/lib/python3.7/dist-packages/tqdm/std.py:658: FutureWarning: The Panel class is removed from pand
as. Accessing it from the top-level namespace will also be removed in the next version
  from pandas import Panel
```

In [ ]:
```
from sklearn.preprocessing import OrdinalEncoder
from sklearn.feature_selection import f_classif
from sklearn.feature_selection import mutual_info_regression
from sklearn.feature_selection import f_regression
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_validate
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import SCORERS
from sklearn.model_selection import KFold
from sklearn.feature_selection import RFECV
from sklearn.preprocessing import PolynomialFeatures
import warnings
warnings.filterwarnings("ignore")
```

In [ ]:
```
def convert_categorical(df, cols):
    for c in cols:
        df[c] = pd.Categorical(df[c])
    return df
```

In [ ]:
```
def enc_df(df, cat):
    X = df.copy(deep = True)
    enc = OrdinalEncoder()
    enc.fit(X[cat])
    X[cat] = enc.transform(X[cat])

    #enc is returned so as to transform any test data, if needed.
    return X, enc
```

In [ ]:
```
df_bik = pd.read_csv('df_bik.csv')

df_sui = pd.read_csv('df_sui.csv',keep_default_na=False)

df_vid = pd.read_csv('df_vid.csv')
df_vid = df_vid.drop(columns=['umem'])
```

In [ ]:
```
y_bik = 'cnt'
y_sui = 'suicides/100k pop'
y_vid = 'utime'
```

```
ys = [y_bik, y_sui, y_vid]

dfs = [df_bik, df_sui, df_vid]

datasets = ['Bike Sharing', 'Suicide', 'Video Transcoding']
```

In [ ]:
```
cat_bik = ['season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday', 'weathersit']
cat_bik_mask = np.array([1,1,1,1,1,1,1,0,0,0,0])

cat_sui = ['year', 'sex', 'age', 'generation','continent']
cat_sui_mask = np.array([1,1,1,0,0,0,1,1])

cat_vid = ['codec', 'o_codec']
cat_vid_mask = np.array([0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0])

cat_masks = [cat_bik_mask, cat_sui_mask, cat_vid_mask]
cat_names = [cat_bik, cat_sui, cat_vid]
```

In [ ]:
```
df_bik = convert_categorical(df_bik, cat_bik)
df_sui = convert_categorical(df_sui, cat_sui)
df_vid = convert_categorical(df_vid, cat_vid)

dfs = [df_bik, df_sui, df_vid]
```

In [ ]:
```
df_bik_oh = pd.get_dummies(df_bik)
df_sui_oh = pd.get_dummies(df_sui)
df_vid_oh = pd.get_dummies(df_vid)

dfs_oh = [df_bik_oh, df_sui_oh, df_vid_oh]
```

In [ ]:
```
df_bikenc, enc_bik = enc_df(df_bik, cat_bik)
df_suienc, enc_sui = enc_df(df_sui, cat_sui)
df_videnc, enc_vid = enc_df(df_vid, cat_vid)

dfs_enc = [df_bikenc, df_suienc, df_videnc]
```

In [ ]:
```
#Checking for any null values
print(df_bik.isnull().sum().sum())
print(df_sui.isnull().sum().sum())
print(df_vid.isnull().sum().sum())
```

```
0
0
0
```

In [ ]:
```
def Poly_Reg_twoal(d, df, Y, drp,alphas,deg,title):
    X = df.drop(columns = [Y] + drp)
    y = df[Y]
    X = pd.get_dummies(X)

    cols = X.columns
    print("No of features", len(cols))
    feat_names = ['x'+str(i) for i in range(len(cols))]
    dfpoly = pd.DataFrame({'col_name':cols, 'col_rep':feat_names})
    display(dfpoly)

    poly = PolynomialFeatures(deg)
    X = poly.fit_transform(X)

    alphar, alphal = alphas

    rid_scores={}
    las_scores={}
```

```
    rid_train_rmse = []
    rid_test_rmse = []
    las_train_rmse = []
    las_test_rmse = []

    rid_reg = Ridge(alpha = alphar)
    scores = cross_validate(rid_reg, X, y, scoring="neg_mean_squared_error", cv=10, return_train_score=Tr
    rid_train_rmse.append(np.sqrt(np.mean(-scores['train_score'])))
    rid_test_rmse.append(np.sqrt(np.mean(-scores['test_score'])))

    las_reg = Lasso(alpha = alphal)
    scores = cross_validate(las_reg, X, y, scoring="neg_mean_squared_error", cv=10, return_train_score=Tr
    las_train_rmse.append(np.sqrt(np.mean(-scores['train_score'])))
    las_test_rmse.append(np.sqrt(np.mean(-scores['test_score'])))


    dframe = pd.DataFrame({'Ridge Train RMSE':rid_train_rmse, 'Ridge Valid RMSE':rid_test_rmse})
    print(d+" Ridge Rigression deg {} ".format(deg)+title)
    display(dframe)

    dframe = pd.DataFrame({'Lasso Train RMSE':las_train_rmse, 'Lasso Valid RMSE':las_test_rmse})
    print(d+" Lasso Rigression deg {} ".format(deg)+title)
    display(dframe)
```

Invert Frame rate and see regression performance

In [ ]:

```
inv_frate = (df_vid['framerate'])**-1
X = df_vid.drop(columns = 'framerate')
X['inv_fr'] = inv_frate
```

In [ ]:

```
deg = 2
Poly_Reg_twoal(datasets[2],X, ys[2],drp = [],alphas = (1000,0.1), deg = deg, title = "")
```

No of features 24

	col_name	col_rep
0	duration	x0
1	width	x1
2	height	x2
3	bitrate	x3
4	i	x4
5	p	x5
6	b	x6
7	frames	x7
8	i_size	x8
9	p_size	x9
10	size	x10
11	o_bitrate	x11
12	o_framerate	x12
13	o_width	x13
14	o_height	x14
15	inv_fr	x15
16	codec_flv	x16
17	codec_h264	x17

	col_name	col_rep
18	codec_mpeg4	x18
19	codec_vp8	x19
20	o_codec_flv	x20
21	o_codec_h264	x21
22	o_codec_mpeg4	x22
23	o_codec_vp8	x23

```
Video Transcoding Ridge Rigression deg 2
```

	Ridge Train RMSE	Ridge Valid RMSE
0	6.333415	14.185797

```
Video Transcoding Lasso Rigression deg 2
```

	Lasso Train RMSE	Lasso Valid RMSE
0	6.584652	7.288184

Invert number of frames:

In [ ]:

```
inv_fr = (df_vid['frames'])**-1.0
X = df_vid.drop(columns = ['frames'])
X['inv_fr'] = inv_fr
deg = 2
Poly_Reg_twoal(datasets[2],X, ys[2],drp = [],alphas = (1000,0.1), deg = deg, title = "")
```

```
No of features 24
```

	col_name	col_rep
0	duration	x0
1	width	x1
2	height	x2
3	bitrate	x3
4	framerate	x4
5	i	x5
6	p	x6
7	b	x7
8	i_size	x8
9	p_size	x9
10	size	x10
11	o_bitrate	x11
12	o_framerate	x12
13	o_width	x13
14	o_height	x14
15	inv_fr	x15
16	codec_flv	x16
17	codec_h264	x17
18	codec_mpeg4	x18
19	codec_vp8	x19

	col_name	col_rep
20	o_codec_flv	x20
21	o_codec_h264	x21
22	o_codec_mpeg4	x22
23	o_codec_vp8	x23

Video Transcoding Ridge Rigression deg 2

	Ridge Train RMSE	Ridge Valid RMSE
0	6.322121	21.227671

Video Transcoding Lasso Rigression deg 2

	Lasso Train RMSE	Lasso Valid RMSE
0	6.563849	7.484193

In [ ]:

```
inv_fr = (df_vid['frames'])**-1.0
inv_frate = (df_vid['framerate'])**-1.0
X = df_vid.drop(columns = ['frames', 'framerate'])
X['inv_fr'] = inv_fr
X['inv_frate'] = inv_frate
deg = 2
Poly_Reg_twoal(datasets[2],X, ys[2],drp = [],alphas = (1000,0.1), deg = deg, title = "")
```

No of features 24

	col_name	col_rep
0	duration	x0
1	width	x1
2	height	x2
3	bitrate	x3
4	i	x4
5	p	x5
6	b	x6
7	i_size	x7
8	p_size	x8
9	size	x9
10	o_bitrate	x10
11	o_framerate	x11
12	o_width	x12
13	o_height	x13
14	inv_fr	x14
15	inv_frate	x15
16	codec_flv	x16
17	codec_h264	x17
18	codec_mpeg4	x18
19	codec_vp8	x19
20	o_codec_flv	x20
21	o_codec_h264	x21

	col_name	col_rep
22	o_codec_mpeg4	x22
23	o_codec_vp8	x23

Video Transcoding Ridge Rigression deg 2

	Ridge Train RMSE	Ridge Valid RMSE
0	6.32069	8.311974

Video Transcoding Lasso Rigression deg 2

	Lasso Train RMSE	Lasso Valid RMSE
0	6.569238	7.134968

In [ ]:

```
def mlp(d,df,Y,hid_lay_size,alpha,lr,batch_size):

    X = df.drop(columns = Y)
    X = pd.get_dummies(X)
    y = df[Y]

    nn_reg = MLPRegressor(hid_lay_size, activation = 'relu', alpha = alpha, max_iter=300,learning_rate_ini
    scores = cross_validate(nn_reg, X, y, scoring="neg_mean_squared_error", cv=10, return_train_score=Tru

    return np.sqrt(np.mean(-scores['test_score'])), np.sqrt(np.mean(-scores['train_score']))
```

In [ ]:

```
#hid_size_list = [[200,500,500,200],[300,600,600,600,200],[500,1000,1000,500]]
#al_list = [1e-1,1e1,1e2,1e3]
#lr_list = [1e-4, 5e-3, 1e-1]
hid_size_list = [[300,600,600,600,200]]
al_list = [1e2]
lr_list = [1e-3]
d = datasets[1]
df = dfs[1]
Y = ys[1]

nn_list = []
print(d+" MLP")
for hid_size in hid_size_list:
    for al in al_list:
        for lr in lr_list:
            val_score, tr_score = mlp(d,df,Y,hid_size, al,lr,1024)
            nn_list.append([hid_size, val_score, tr_score, lr,al])
nn_df = pd.DataFrame(nn_list, columns = ['Hidden Size', 'Avg Valid RMSE', 'Avg Train RMSE', 'LR','Alpha'])
display(nn_df)
```

Suicide MLP

	Hidden Size	Avg Valid RMSE	Avg Train RMSE	LR	Alpha
0	[300, 600, 600, 600, 200]	14.768253	12.141768	0.001	100.0

In [ ]:

```
hid_size_list = [[300, 500, 200],[200,500,500,200],[300,600,600,600,200]]
al_list = [1e-1,1e1,1e2,1e3]
lr_list = [1e-4, 5e-3, 1e-1]
for d, df, Y in zip(datasets, dfs, ys):
    nn_list = []
    print(d+" MLP")
    for hid_size in hid_size_list:
        for al in al_list:
            for lr in lr_list:
                val_score, tr_score = mlp(d,df,Y,hid_size, al,lr)
                nn_list.append([hid_size, val_score, tr_score, lr])
    nn_df = pd.DataFrame(nn_list, columns = ['Hidden Size', 'Avg Valid RMSE', 'Avg Train RMSE', 'LR'])
    display(nn_df)
```

Bike Sharing MLP

	Hidden Size	Avg Valid RMSE	Avg Train RMSE	LR
0	[300, 500, 200]	945.991572	771.994056	0.0001
1	[300, 500, 200]	913.062202	658.613518	0.0050
2	[300, 500, 200]	1236.097119	1201.043316	0.1000
3	[300, 500, 200]	1066.346358	856.686661	0.0001
4	[300, 500, 200]	862.294441	700.119869	0.0050
5	[300, 500, 200]	901.896137	754.942178	0.1000
6	[300, 500, 200]	910.431761	765.503430	0.0001
7	[300, 500, 200]	851.401551	653.799133	0.0050
8	[300, 500, 200]	836.028609	501.176772	0.1000
9	[300, 500, 200]	890.704954	757.110922	0.0001
10	[300, 500, 200]	846.941875	646.715115	0.0050
11	[300, 500, 200]	868.017783	460.029352	0.1000
12	[200, 500, 500, 200]	892.504273	901.425112	0.0001
13	[200, 500, 500, 200]	790.181225	478.767594	0.0050
14	[200, 500, 500, 200]	928.239549	722.819133	0.1000
15	[200, 500, 500, 200]	872.668926	709.277990	0.0001
16	[200, 500, 500, 200]	857.497725	554.731027	0.0050
17	[200, 500, 500, 200]	850.742015	631.340909	0.1000
18	[200, 500, 500, 200]	843.677566	678.420703	0.0001
19	[200, 500, 500, 200]	867.975918	513.090648	0.0050
20	[200, 500, 500, 200]	942.211655	589.915409	0.1000
21	[200, 500, 500, 200]	832.639317	679.320955	0.0001
22	[200, 500, 500, 200]	898.274294	592.051691	0.0050
23	[200, 500, 500, 200]	930.446682	545.482235	0.1000
24	[300, 600, 600, 600, 200]	812.559547	470.328962	0.0001
25	[300, 600, 600, 600, 200]	811.689381	513.925067	0.0050
26	[300, 600, 600, 600, 200]	1884.979581	1528.835868	0.1000
27	[300, 600, 600, 600, 200]	814.950618	508.141826	0.0001
28	[300, 600, 600, 600, 200]	820.352491	450.234021	0.0050
29	[300, 600, 600, 600, 200]	1262.282642	842.981638	0.1000
30	[300, 600, 600, 600, 200]	798.274542	490.637614	0.0001
31	[300, 600, 600, 600, 200]	871.172024	440.256064	0.0050
32	[300, 600, 600, 600, 200]	1010.349786	623.197229	0.1000
33	[300, 600, 600, 600, 200]	808.869003	521.344530	0.0001
34	[300, 600, 600, 600, 200]	814.254005	441.076267	0.0050
35	[300, 600, 600, 600, 200]	964.630918	552.493932	0.1000

Suicide MLP

In [ ]:
```
def mlp_noCV(d,df,Y,hid_lay_size,alpha,batch_size,lr):
```

```
    X = df.drop(columns = Y)
    X = pd.get_dummies(X)
    y = df[Y]
    X_train, X_test, y_train, y_test = train_test_split(X, y,random_state=1)
    nn_reg = MLPRegressor(hid_lay_size, activation = 'relu', alpha = alpha, max_iter=300,learning_rate_ini
    nn_reg.fit(X_train, y_train)
    train_pred = nn_reg.predict(X_train)
    test_pred = nn_reg.predict(X_test)

    return mean_squared_error(y_train,train_pred), mean_squared_error(y_test,test_pred)
```

In [ ]:

```
hid_size_list = [[300,600,600,600,200]]
al_list = [1e3]
lr_list = [1e-3]
d = datasets[1]
df = dfs[1]
Y = ys[1]

nn_list = []
print(d+" MLP")
for hid_size in hid_size_list:
    for al in al_list:
        for lr in lr_list:
            tr_score, val_score = mlp_noCV(d,df,Y,hid_size, al,512,lr)
            nn_list.append([hid_size, val_score, tr_score, lr,al])
nn_df = pd.DataFrame(nn_list, columns = ['Hidden Size', 'Avg Valid RMSE', 'Avg Train RMSE', 'LR','Alpha'])
display(nn_df)
```

```
Suicide MLP
```

	Hidden Size	Avg Valid RMSE	Avg Train RMSE	LR	Alpha
0	[300, 600, 600, 600, 200]	166.740852	167.730007	0.001	1000.0

In [ ]:

```
hid_size_list = [[200,500,500,200],[300,600,600,600,200]]
al_list = [1e-2,1e1,1e2,1e3]
lr_list = [ 5e-3]
d = datasets[1]
df = dfs[1]
Y = ys[1]

nn_list = []
print(d+" MLP")
for hid_size in hid_size_list:
    for al in al_list:
        for lr in lr_list:
            tr_score, val_score = mlp_noCV(d,df,Y,hid_size, al,512,lr)
            nn_list.append([hid_size, val_score, tr_score, lr,al])
nn_df = pd.DataFrame(nn_list, columns = ['Hidden Size', 'Avg Valid RMSE', 'Avg Train RMSE', 'LR','Alpha'])
display(nn_df)
```

```
Suicide MLP
```

	Hidden Size	Avg Valid RMSE	Avg Train RMSE	LR	Alpha
0	[200, 500, 500, 200]	158.709368	109.620041	0.005	0.01
1	[200, 500, 500, 200]	154.103722	106.100199	0.005	10.00
2	[200, 500, 500, 200]	142.941868	104.676267	0.005	100.00
3	[200, 500, 500, 200]	180.350828	184.202084	0.005	1000.00
4	[300, 600, 600, 600, 200]	146.117978	67.098693	0.005	0.01
5	[300, 600, 600, 600, 200]	135.497625	79.403298	0.005	10.00
6	[300, 600, 600, 600, 200]	140.504993	110.779732	0.005	100.00
7	[300, 600, 600, 600, 200]	171.475260	173.471465	0.005	1000.00

In [ ]:
```
df_vid.columns
```

Out[ ]:
```
Index(['duration', 'codec', 'width', 'height', 'bitrate', 'framerate', 'i',
       'p', 'b', 'frames', 'i_size', 'p_size', 'size', 'o_codec', 'o_bitrate',
       'o_framerate', 'o_width', 'o_height', 'utime'],
      dtype='object')
```

In [ ]:
```

```

In [15]:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from skopt import BayesSearchCV
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score, cross_validate, train_test_split
from sklearn.metrics import mean_squared_error
```

Bike data baye search

In [19]:

```
df_bik = pd.read_csv('df_bik.csv')
X = df_bik.drop(columns='cnt')
y = df_bik['cnt']

grid_pipline = Pipeline([('model', RandomForestRegressor())])

param_grid = [
    {
        'model__n_estimators': [x for x in range(1, 200, 5)],
        'model__max_depth': [x for x in range(1, 30)],
        'model__max_features': [x for x in range(1, 34)],
    }
]

grid = BayesSearchCV(grid_pipline,param_grid, cv=10, n_jobs=-1, scoring="neg_root_mean_squared_error")
grid.fit(X, y)
df = pd.DataFrame(grid.cv_results_)
df.to_csv('random_forest_bike.csv')
```

```
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
-1097.280422033911
```

In [33]:

```
df_sui = pd.read_csv('df_sui.csv')
X = df_sui.drop(columns='suicides/100k pop')
y = df_sui['suicides/100k pop']

grid_pipline = Pipeline([('model', RandomForestRegressor())])

param_grid = [
    {
        'model__n_estimators': [x for x in range(1, 200, 5)],
        'model__max_depth': [x for x in range(1, 30)],
        'model__max_features': [x for x in range(1, 24)],
    }
]

grid = BayesSearchCV(grid_pipline,param_grid, cv=10, n_jobs=-1, scoring="neg_root_mean_squared_error")
grid.fit(X, y)
df = pd.DataFrame(grid.cv_results_)
df.to_csv('random_forest_suicide.csv')
```

In [51]:
```python
df_vid = pd.read_csv('df_vid.csv')
X = df_vid.drop(columns='utime')
y = df_vid['utime']

grid_pipline = Pipeline([('model', RandomForestRegressor())])

param_grid = [
    {
        'model__n_estimators': [x for x in range(1, 200, 5)],
        'model__max_depth': [x for x in range(1, 30)],
        'model__max_features': [x for x in range(10, 24)],
    }
]

grid = BayesSearchCV(grid_pipline,param_grid, cv=10, n_jobs=-1, scoring="neg_root_mean_squared_error")
grid.fit(X, y)
df = pd.DataFrame(grid.cv_results_)
df.to_csv('random_forest_video.csv')
```

```
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
```

```
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

Load the dataset you want to plot by uncomment any of these load, and draw the graph of rmse score vs parameter value in the next cel

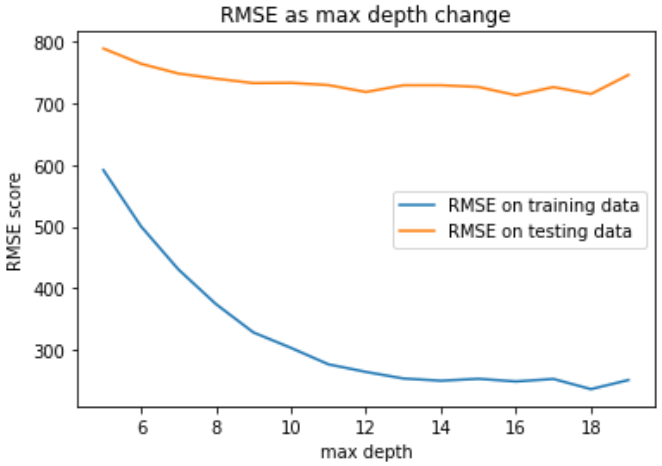
In [59]:

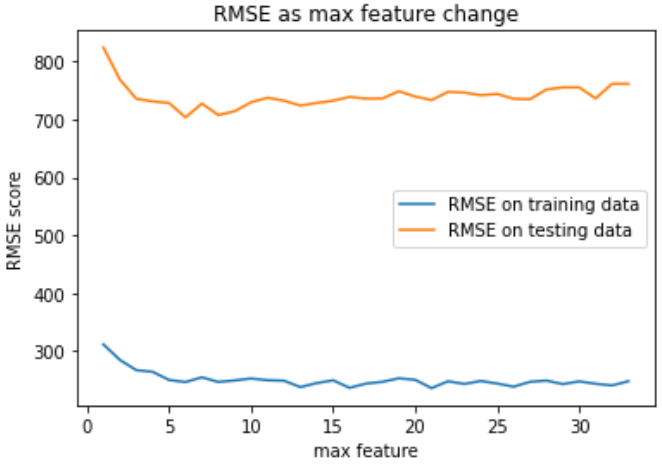
```
df_bik = pd.read_csv('df_bik.csv')
X = df_bik.drop(columns='cnt')
y = df_bik['cnt']

# # load suicide dataset, uncomment to use
# df_sui = pd.read_csv('df_sui.csv')
# X = df_sui.drop(columns='suicides/100k pop')
# y = df_sui['suicides/100k pop']

# # load video dataset, uncomment to use
# df_vid = pd.read_csv('df_vid.csv')
# X = df_vid.drop(columns='utime')
# y = df_vid['utime']
```

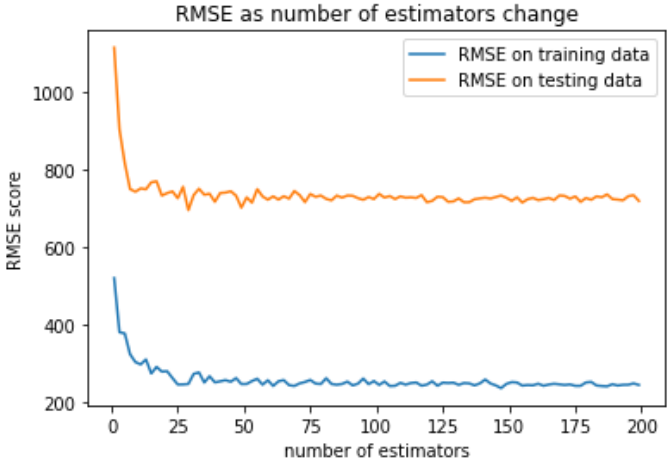
In [31]:

```
max_depth = [x for x in range(5, 20)]
```

```
max_features = [x for x in range(1, 34)]
n_estimators = [x for x in range(1, 200, 2)]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

train_score = []
test_score = []
for md in max_depth:
    rf = RandomForestRegressor(max_features=11, n_estimators=106, max_depth=md)
    rf.fit(X_train, y_train)
    pred = rf.predict(X_train)
    train_score.append(np.sqrt(mean_squared_error(y_train, pred)))
    pred = rf.predict(X_test)
    test_score.append(np.sqrt(mean_squared_error(y_test, pred)))

plt.plot(max_depth, train_score, label = "RMSE on training data")
plt.plot(max_depth, test_score, label = "RMSE on testing data")
plt.xlabel("max depth")
plt.ylabel("RMSE score")
plt.title("RMSE as max depth change")
plt.legend()
plt.show()

max_features = [x for x in range(1, 34)]
train_score = []
test_score = []
for mf in max_features:
    rf = RandomForestRegressor(max_features=mf, n_estimators=106, max_depth=18)
    rf.fit(X_train, y_train)
    pred = rf.predict(X_train)
    train_score.append(np.sqrt(mean_squared_error(y_train, pred)))
    pred = rf.predict(X_test)
    test_score.append(np.sqrt(mean_squared_error(y_test, pred)))

plt.plot(max_features, train_score, label = "RMSE on training data")
plt.plot(max_features, test_score, label = "RMSE on testing data")
plt.xlabel("max feature")
plt.ylabel("RMSE score")
plt.title("RMSE as max feature change")
plt.legend()
plt.show()

n_estimators = [x for x in range(1, 200, 2)]
train_score = []
test_score = []
for ne in n_estimators:
    rf = RandomForestRegressor(max_features=11, n_estimators=ne, max_depth=18)
    rf.fit(X_train, y_train)
    pred = rf.predict(X_train)
    train_score.append(np.sqrt(mean_squared_error(y_train, pred)))
    pred = rf.predict(X_test)
    test_score.append(np.sqrt(mean_squared_error(y_test, pred)))

plt.plot(n_estimators, train_score, label = "RMSE on training data")
plt.plot(n_estimators, test_score, label = "RMSE on testing data")
plt.xlabel("number of estimators")
plt.ylabel("RMSE score")
plt.title("RMSE as number of estimators change")
plt.legend()
plt.show()
```







Draw the tree

In [49]:

```
from sklearn import tree
import graphviz

df_sui = pd.read_csv('df_sui.csv')
X = df_sui.drop(columns='suicides/100k pop')
y = df_sui['suicides/100k pop']

feature_names = ["year","population","gdp_for_year ($)","gdp_per_capita ($)","sex_female","sex_male","age_
rf = RandomForestRegressor(max_features=5, n_estimators=81, max_depth=4)
rf.fit(X, y)
my_tree = rf.estimators_[1]
dot_data = tree.export_graphviz(my_tree, out_file=None, feature_names=feature_names,
                                class_names="charges",   filled=True, rounded=True,  special_characters='
graph = graphviz.Source(dot_data)
graph.render('random_forest', view=False)
```

Out[49]: 'random_forest.pdf'

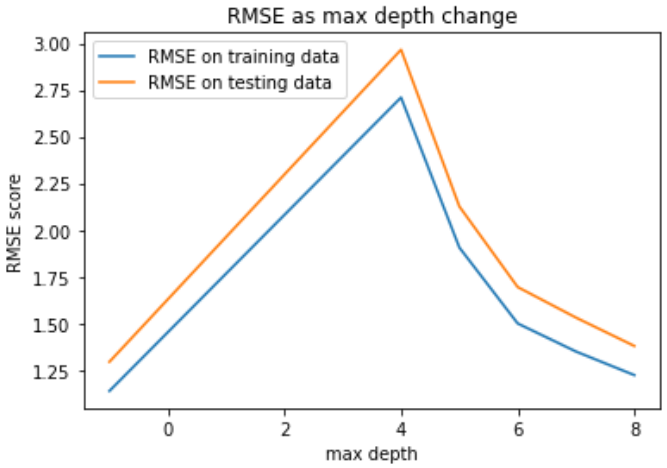
In [64]:
```
from lightgbm import LGBMRegressor

param_grid = {
    'n_estimators': [200, 220, 240, 260, 280, 300],
    'max_depth': [-1, 4, 6, 8],
    'min_data_in_leaf': [10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30]
}


grid = BayesSearchCV(LGBMRegressor(), param_grid, cv=10, n_jobs=-1, scoring="neg_root_mean_squared_error")
grid.fit(X, y)
df = pd.DataFrame(grid.cv_results_)
df.to_csv('LightGBM.csv')
```

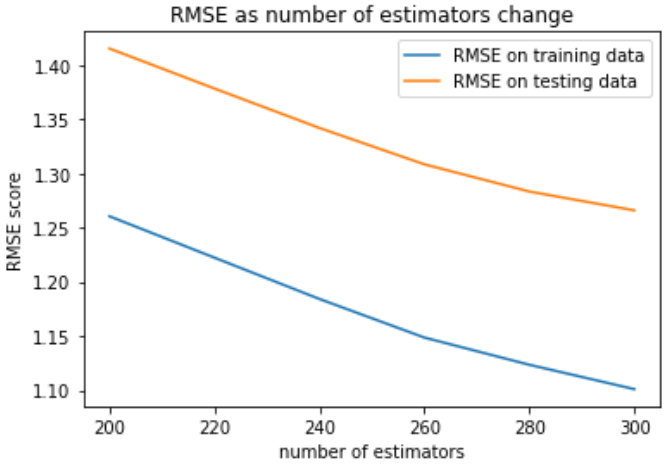
```
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

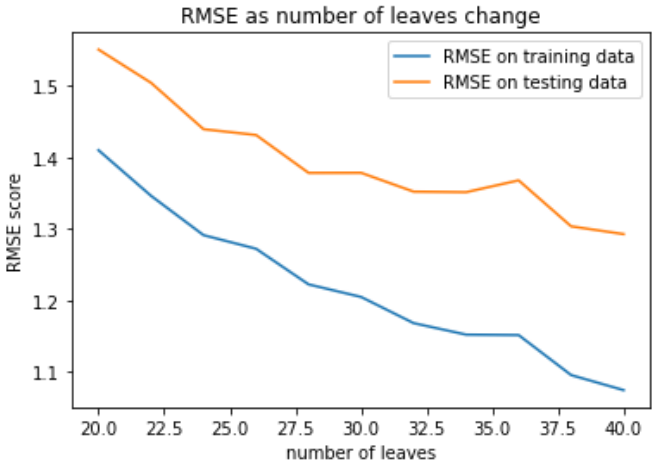
test generalization

In [69]:
```
max_depth = [-1, 4,5, 6,7, 8]
n_estimators = [200, 220, 240, 260, 280, 300]
num_leaves = [20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40]


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

train_score = []
test_score = []
for md in max_depth:
    rf = LGBMRegressor(max_depth=md, n_estimators=220, num_leaves=28)
    rf.fit(X_train, y_train)
    pred = rf.predict(X_train)
    train_score.append(np.sqrt(mean_squared_error(y_train, pred)))
    pred = rf.predict(X_test)
    test_score.append(np.sqrt(mean_squared_error(y_test, pred)))

plt.plot(max_depth, train_score, label = "RMSE on training data")
plt.plot(max_depth, test_score, label = "RMSE on testing data")
plt.xlabel("max depth")
plt.ylabel("RMSE score")
plt.title("RMSE as max depth change")
plt.legend()
plt.show()

train_score = []
test_score = []
for ne in n_estimators:
    rf = LGBMRegressor(max_depth=8, n_estimators=ne, num_leaves=28)
    rf.fit(X_train, y_train)
    pred = rf.predict(X_train)
    train_score.append(np.sqrt(mean_squared_error(y_train, pred)))
    pred = rf.predict(X_test)
    test_score.append(np.sqrt(mean_squared_error(y_test, pred)))

plt.plot(n_estimators, train_score, label = "RMSE on training data")
plt.plot(n_estimators, test_score, label = "RMSE on testing data")
plt.xlabel("number of estimators")
plt.ylabel("RMSE score")
plt.title("RMSE as number of estimators change")
plt.legend()
plt.show()


train_score = []
test_score = []
```

```
for nl in num_leaves:
    rf = LGBMRegressor(max_depth=8, n_estimators=220, num_leaves=nl)
    rf.fit(X_train, y_train)
    pred = rf.predict(X_train)
    train_score.append(np.sqrt(mean_squared_error(y_train, pred)))
    pred = rf.predict(X_test)
    test_score.append(np.sqrt(mean_squared_error(y_test, pred)))

plt.plot(num_leaves, train_score, label = "RMSE on training data")
plt.plot(num_leaves, test_score, label = "RMSE on testing data")
plt.xlabel("number of leaves")
plt.ylabel("RMSE score")
plt.title("RMSE as number of leaves change")
plt.legend()
plt.show()
```







In [65]:

```python
from catboost import CatBoostRegressor

param_grid = {
    'learning_rate': [1, 0.1, 0.01],
    'depth': [4, 6, 7, 8],
    'iterations': [10, 12, 14, 16, 18, 20]
}

grid = BayesSearchCV(CatBoostRegressor(silent=True), param_grid, cv=10, n_jobs=-1, scoring="neg_root_mean
grid.fit(X, y)
df = pd.DataFrame(grid.cv_results_)
df.to_csv('CatBoost.csv')
```

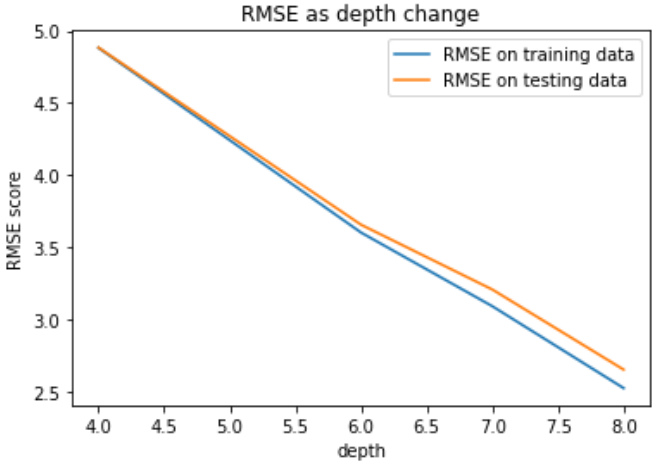
```
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
```

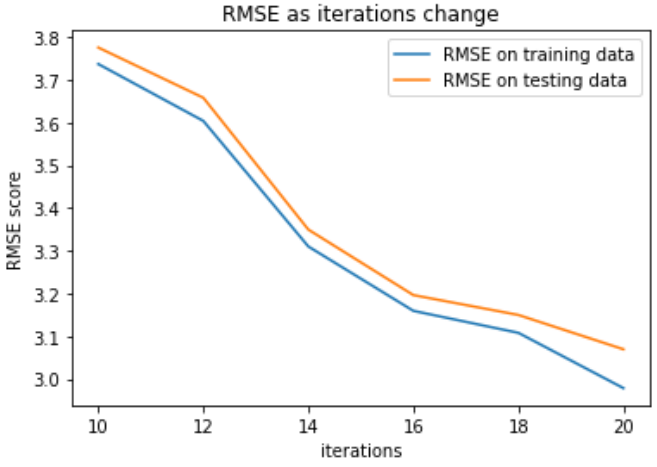
```
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
/home/vagrant/.local/lib/python3.8/site-packages/skopt/optimizer/optimizer.py:449: UserWarning: The objecti
ve has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

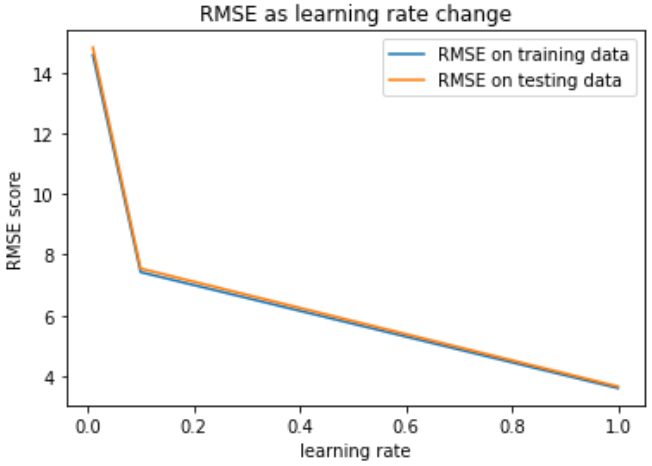
In [71]:

```
depth=[4, 6, 7, 8]
iterations=[10, 12, 14, 16, 18, 20]
learning_rate=[1, 0.1, 0.01]


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

train_score = []
test_score = []
for d in depth:
    rf = CatBoostRegressor(silent=True, depth=d, iterations=12, learning_rate=1)
    rf.fit(X_train, y_train)
    pred = rf.predict(X_train)
    train_score.append(np.sqrt(mean_squared_error(y_train, pred)))
    pred = rf.predict(X_test)
    test_score.append(np.sqrt(mean_squared_error(y_test, pred)))

plt.plot(depth, train_score, label = "RMSE on training data")
plt.plot(depth, test_score, label = "RMSE on testing data")
plt.xlabel("depth")
plt.ylabel("RMSE score")
plt.title("RMSE as depth change")
plt.legend()
plt.show()

train_score = []
test_score = []
for i in iterations:
    rf = CatBoostRegressor(silent=True, depth=6, iterations=i, learning_rate=1)
    rf.fit(X_train, y_train)
    pred = rf.predict(X_train)
    train_score.append(np.sqrt(mean_squared_error(y_train, pred)))
    pred = rf.predict(X_test)
    test_score.append(np.sqrt(mean_squared_error(y_test, pred)))

plt.plot(iterations, train_score, label = "RMSE on training data")
plt.plot(iterations, test_score, label = "RMSE on testing data")
plt.xlabel("iterations")
plt.ylabel("RMSE score")
plt.title("RMSE as iterations change")
plt.legend()
plt.show()


train_score = []
test_score = []
for lr in learning_rate:
    rf = CatBoostRegressor(silent=True, depth=6, iterations=12, learning_rate=lr)
    rf.fit(X_train, y_train)
    pred = rf.predict(X_train)
    train_score.append(np.sqrt(mean_squared_error(y_train, pred)))
    pred = rf.predict(X_test)
    test_score.append(np.sqrt(mean_squared_error(y_test, pred)))
```

```
plt.plot(learning_rate, train_score, label = "RMSE on training data")
plt.plot(learning_rate, test_score, label = "RMSE on testing data")
plt.xlabel("learning rate")
plt.ylabel("RMSE score")
plt.title("RMSE as learning rate change")
plt.legend()
plt.show()
```







In [75]:

```
df_bik = pd.read_csv('df_bik.csv')
X = df_bik.drop(columns='cnt')
y = df_bik['cnt']

rf = RandomForestRegressor(max_features=11, n_estimators=106, max_depth=18, oob_score=True)
rf.fit(X, y)
print("bike dataset oob score")
```

```
print(rf.oob_score_)

df_sui = pd.read_csv('df_sui.csv')
X = df_sui.drop(columns='suicides/100k pop')
y = df_sui['suicides/100k pop']

rf = RandomForestRegressor(max_features=5, n_estimators=81, max_depth=10, oob_score=True)
rf.fit(X, y)
print("suicide dataset oob score")
print(rf.oob_score_)

df_vid = pd.read_csv('df_vid.csv')
X = df_vid.drop(columns='utime')
y = df_vid['utime']

rf = RandomForestRegressor(max_features=18, n_estimators=136, max_depth=11, oob_score=True)
rf.fit(X, y)
print("video dataset oob score")
print(rf.oob_score_)
```

```
bike dataset oob score
0.8811218588395673
suicide dataset oob score
0.6402305383530402
video dataset oob score
0.9765867598563309
```

In [ ]:

Project 4: Regression

ECE 219: Large-Scale Data Mining: Models and Algorithms [Winter 2021]

Prof. Vwani Roychowdhury

UCLA, Department of ECE

Due: 2021.03.19 11:59PM PT

In [1]:
```
# !pip install pandas-profiling[notebook]
```

In [2]:
```
from pandas_profiling import ProfileReport
from IPython.display import display
from tqdm.notebook import tqdm
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd

tqdm.pandas()
```

```
C:\Users\fabri\Anaconda3\envs\python38\lib\site-packages\tqdm\std.py:670: FutureWarning: The Panel class is
removed from pandas. Accessing it from the top-level namespace will also be removed in the next version
  from pandas import Panel
```

In [9]:
```
BIKE_SHARING_DATA = "data/bike_sharing/day.csv"
SUICIDE_DATA = "data/suicides/master.csv"
VIDEO_TRANSCODING_DATA = "data/video_transcoding/transcoding_mesurment.tsv"
```

In [10]:
```
Ys_bik = ['casual', 'registered', 'cnt']
df_bik = pd.read_csv(BIKE_SHARING_DATA)
df_bik.drop(columns=['instant', 'dteday'], inplace=True)
```

In [11]:
```
Xs_sui = [
    'country',
    'year',
    'sex',
    'age',
    'population',
    ' gdp_for_year ($) ',
    'gdp_per_capita ($)',
    'generation'
]
Ys_sui = ['suicides_no', 'suicides/100k pop']
df_sui = pd.read_csv(SUICIDE_DATA, usecols=Xs_sui+Ys_sui)
df_sui.reset_index(drop=True, inplace=True)
df_sui.columns = [col.strip() for col in df_sui.columns]
Xs_sui = [col.strip() for col in Xs_sui]
df_sui['gdp_for_year ($)'] = df_sui['gdp_for_year ($)'].map(lambda x: int(x.replace(',', "")))
```

In [12]:
```
Ys_vid = ['utime']
df_vid = pd.read_csv(VIDEO_TRANSCODING_DATA, sep='\t')
df_vid.drop(columns=['id', 'b_size'], inplace=True)
```

In [13]:
```
dfs = [df_bik, df_sui, df_vid]
Ys = [Ys_bik, Ys_sui, Ys_vid]
```

```
datasets = ['Bike Sharing', 'Suicide', 'Video Transcoding']
colors = ['blue', 'red', 'green']
```

Q1

Plot a heatmap of Pearson correlation matrix of dataset columns. Report which features have the highest absolute correlation with the target variable and what that implies.

ANSWER

See report directly (link).

In [9]:
```
for d, df, Y in zip(datasets, dfs, Ys):
    print(d)
    df_corr = df.corr(method='pearson').loc[Y].T
    display(df_corr.sort_values(by=df_corr.columns[0], ascending=False))
```

Bike Sharing

	casual	registered	cnt
casual	1.000000	0.395282	0.672804
cnt	0.672804	0.945517	1.000000
atemp	0.543864	0.544192	0.631066
temp	0.543285	0.540012	0.627494
registered	0.395282	1.000000	0.945517
yr	0.248546	0.594248	0.566710
season	0.210399	0.411623	0.406100
mnth	0.123006	0.293488	0.279977
weekday	0.059923	0.057367	0.067443
holiday	0.054274	-0.108745	-0.068348
hum	-0.077008	-0.091089	-0.100659
windspeed	-0.167613	-0.217449	-0.234545
weathersit	-0.247353	-0.260388	-0.297391
workingday	-0.518044	0.303907	0.061156

Suicide

	suicides_no	suicides/100k pop
suicides_no	1.000000	0.306604
population	0.616162	0.008285
gdp_for_year ($)	0.430096	0.025240
suicides/100k pop	0.306604	1.000000
gdp_per_capita ($)	0.061330	0.001785
year	-0.004546	-0.039037

Video Transcoding

	utime
utime	1.000000
umem	0.663301

	utime
o_width	0.523388
o_height	0.519649
o_bitrate	0.155479
bitrate	0.155200
width	0.129861
height	0.128479
o_framerate	0.104043
p_size	0.097644
size	0.097096
framerate	0.079336
i_size	0.064711
p	0.033201
frames	0.033115
i	0.018489
duration	0.005533
b	0.005140

In [61]:
```
ProfileReport(df_bik, explorative=True).to_notebook_iframe()
```

Overview

Dataset statistics

Number of variables	16
Number of observations	731
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	133.6 KiB
Average record size in memory	187.2 B

Variable types

Numeric	10
Categorical	6

Warnings

`dteday` has a high cardinality: 731 distinct values	**High cardinality**
`temp` is highly correlated with `atemp`	**High correlation**
`atemp` is highly correlated with `temp`	**High correlation**
`registered` is highly correlated with `cnt`	**High correlation**

In [62]:
```
ProfileReport(df_sui, explorative=True).to_notebook_iframe()
```

Overview

Dataset statistics

Number of variables	10
Number of observations	27820
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	8.2 MiB
Average record size in memory	309.7 B

Variable types

Categorical	4
Numeric	6

Warnings

`country` has a high cardinality: 101 distinct values	**High cardinality**
`sex` is uniformly distributed	**Uniform**
`age` is uniformly distributed	**Uniform**
`suicides_no` has 4281 (15.4%) zeros	**Zeros**

In [63]:
```
ProfileReport(df_vid, explorative=True).to_notebook_iframe()
```

Overview

Dataset statistics

Number of variables	22
Number of observations	68784
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	22.4 MiB
Average record size in memory	341.6 B

Variable types

Categorical	5
Numeric	17

Warnings

`b_size` has constant value "0"	**Constant**
`id` has a high cardinality: 1099 distinct values	**High cardinality**
`width` is highly correlated with `height`	**High correlation**
`height` is highly correlated with `width`	**High correlation**

In [64]:

```
for d, df, Y in zip(datasets, dfs, Ys):
    p = ProfileReport(
        df,
        title="Pandas Profiling Report for " + d + " Dataset",
        explorative=True
    )
    # p.to_notebook_iframe()
    p.to_file("reports/" + d + " Report.html")
```

Q2

Plot the histogram of numerical features. What preprocessing can be done if the distribution of a feature has high skewness?

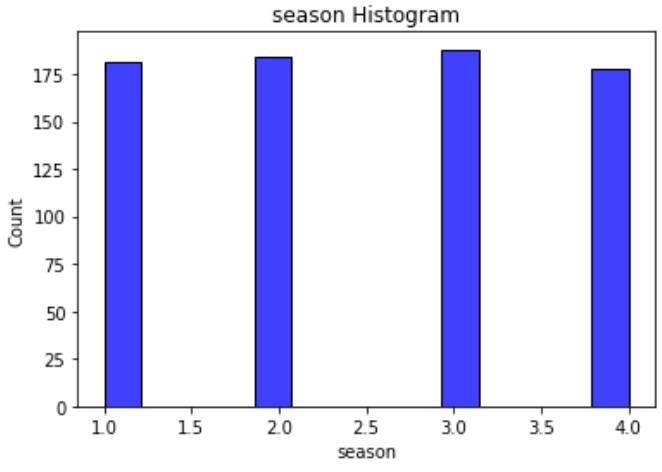
ANSWER

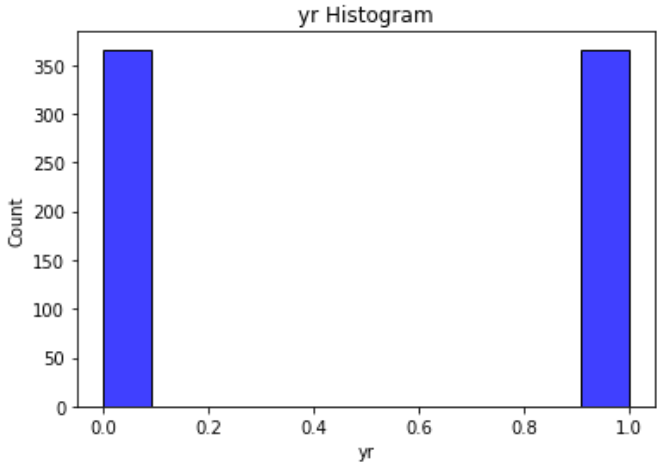
The following techniques can be used to address feature skew:

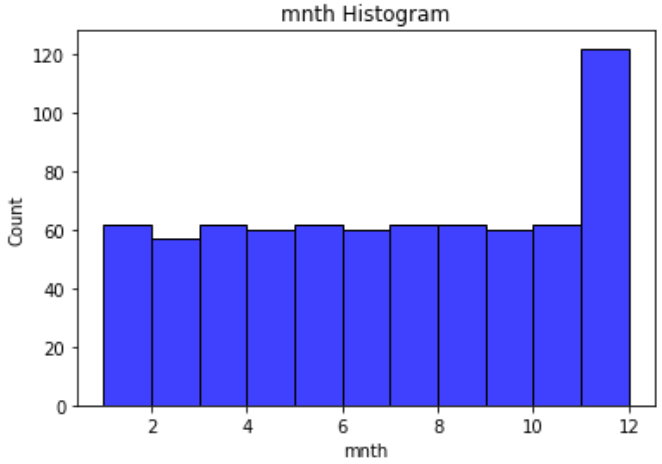
- log transform: the logarithm, x to log_{10} or log_{e} or log_{2} of x, is a strong transformation and can be used to reduce right skewness.
- boxcox transform: a transformation of a non-normal dependent variables into a normal shape.
- root transform (square, cube, etc.): fairly strong transformation for negatively skewed data with a substantial effect on distribution shape, but is weaker than the logarithm. Can be applied to negative and zero values too.
- power transform (square, cube, etc.): x to x^n, has a moderate effect on distribution shape and it could be used to reduce left skewness.

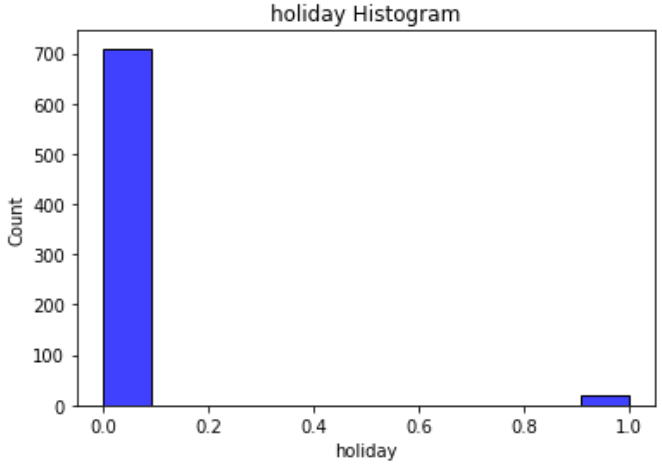
In [10]:
```
def get_numeric_cols(df):
    return df.select_dtypes(include=[np.number]).columns
```

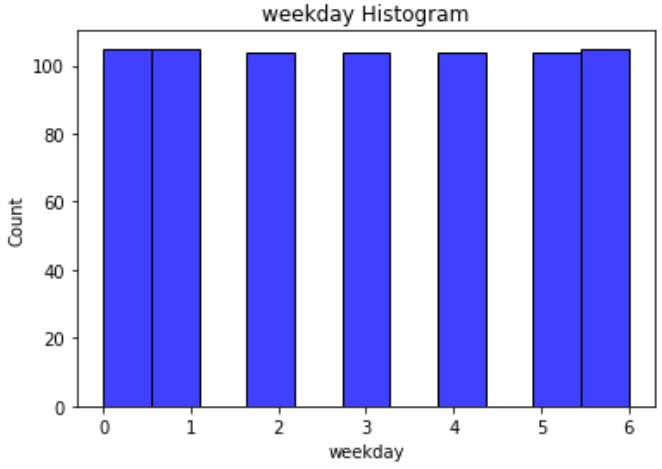
In [11]:
```
for d, df, Y, c in zip(datasets, dfs, Ys, colors):
    for col in get_numeric_cols(df):
        g = sns.histplot(df[col], color=c)
        g.set_title(col + " Histogram")
        plt.show()
```

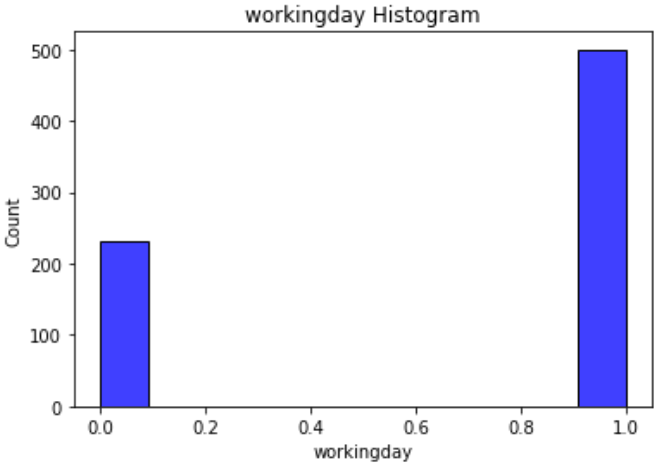


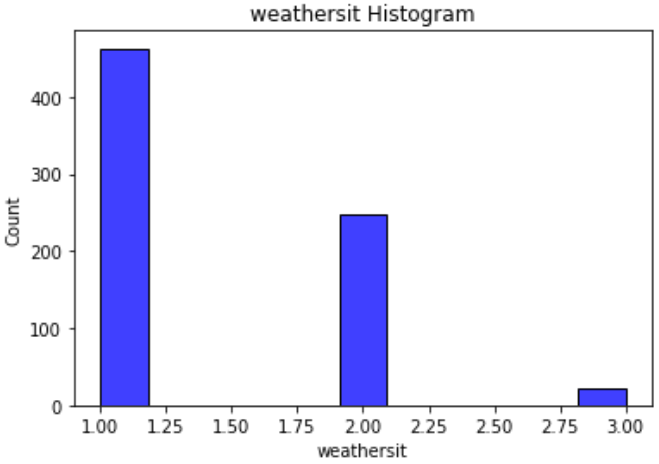

yr Histogram
Count
yr
0
50
100
150
200
250
300
350
0.0
0.2
0.4
0.6
0.8
1.0

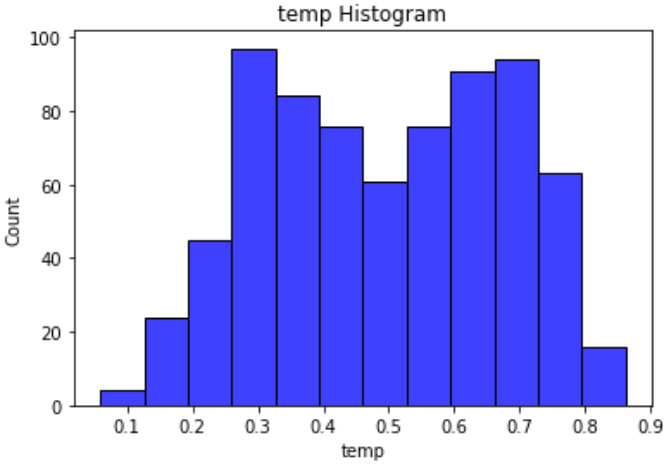

mnth Histogram
Count
mnth
0
20
40
60
80
100
120
2
4
6
8
10
12

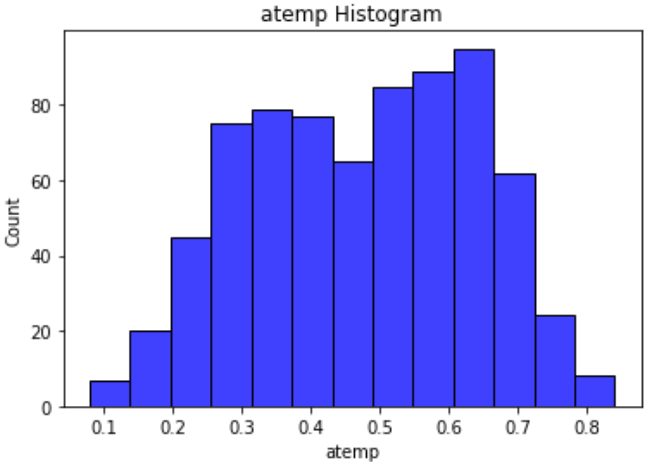

holiday Histogram
Count
holiday
0
100
200
300
400
500
600
700
0.0
0.2
0.4
0.6
0.8
1.0

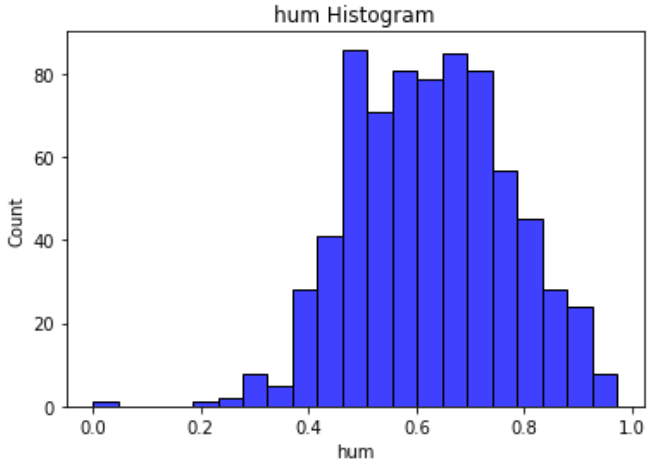

weekday Histogram
Count
weekday
0
20
40
60
80
100
0
1
2
3
4
5
6

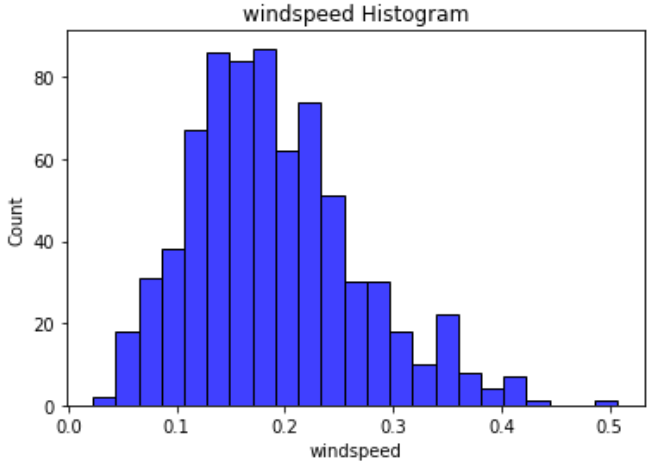

workingday Histogram
Count
workingday
0
100
200
300
400
500
0.0
0.2
0.4
0.6
0.8
1.0

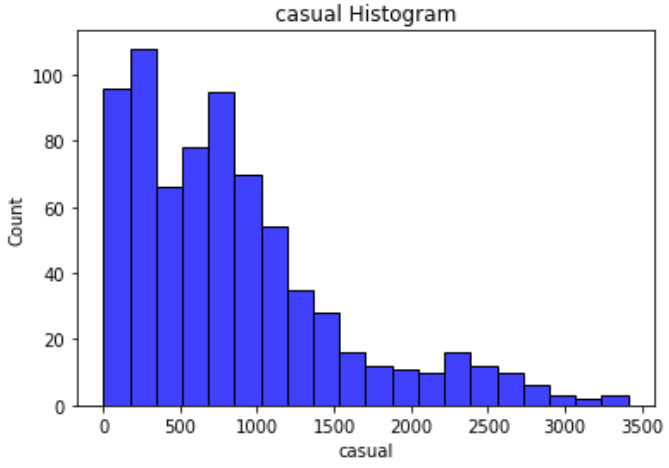

weathersit Histogram
Count
weathersit
0
100
200
300
400
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00

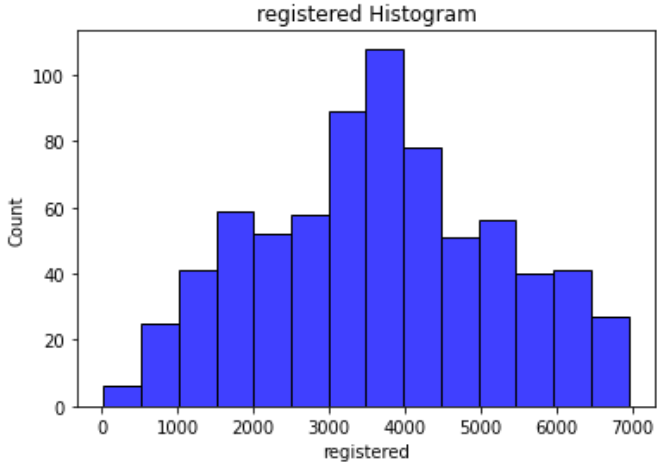

temp Histogram
Count
temp
0
20
40
60
80
100
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

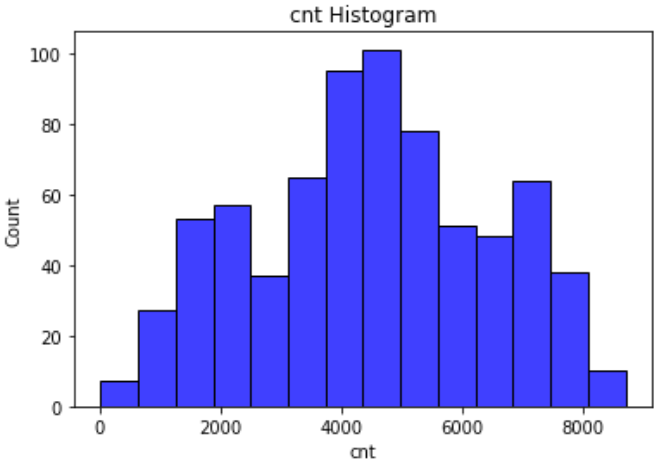

atemp Histogram
Count
atemp
0
20
40
60
80
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

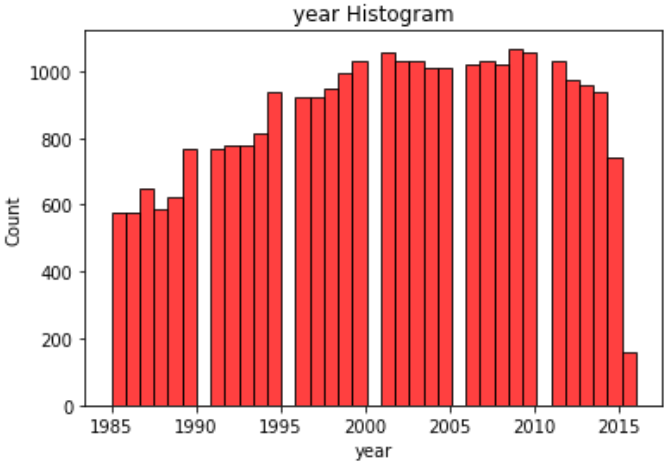

hum Histogram
Count
hum
0
20
40
60
80
0.0
0.2
0.4
0.6
0.8
1.0

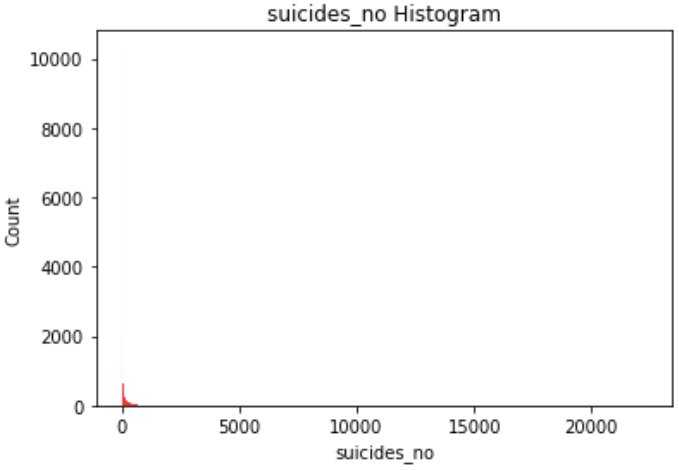

windspeed Histogram
Count
windspeed
0
20
40
60
80
0.0
0.1
0.2
0.3
0.4
0.5

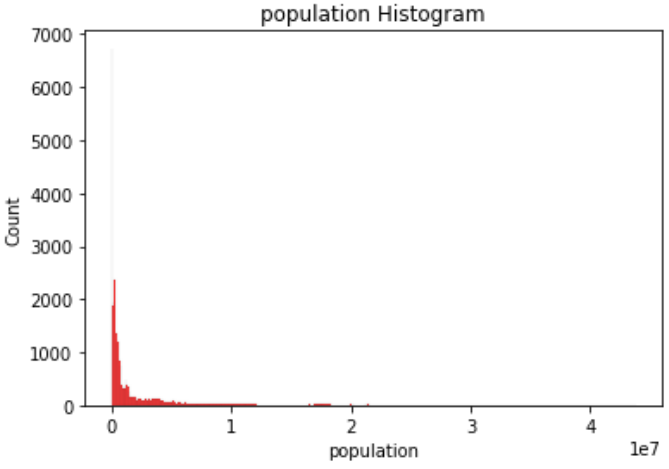

casual Histogram
Count
casual
0
20
40
60
80
100
0
500
1000
1500
2000
2500
3000
3500

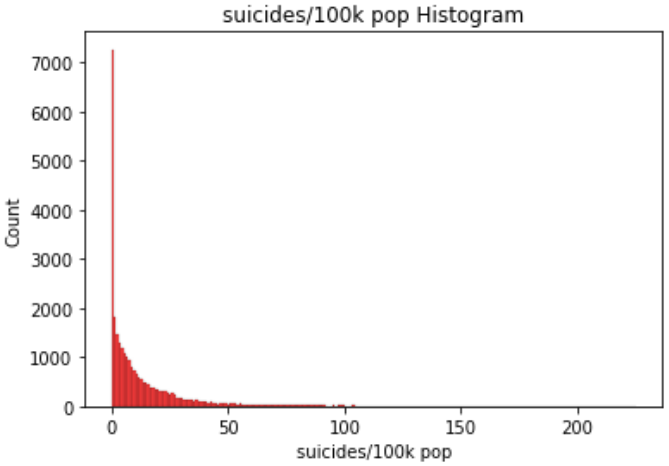

registered Histogram
Count
registered
0
20
40
60
80
100
0
1000
2000
3000
4000
5000
6000
7000

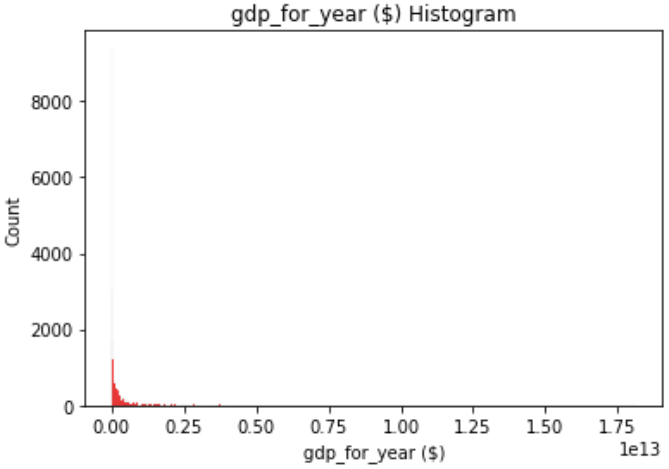

cnt Histogram
Count
cnt
0
20
40
60
80
100
0
2000
4000
6000
8000

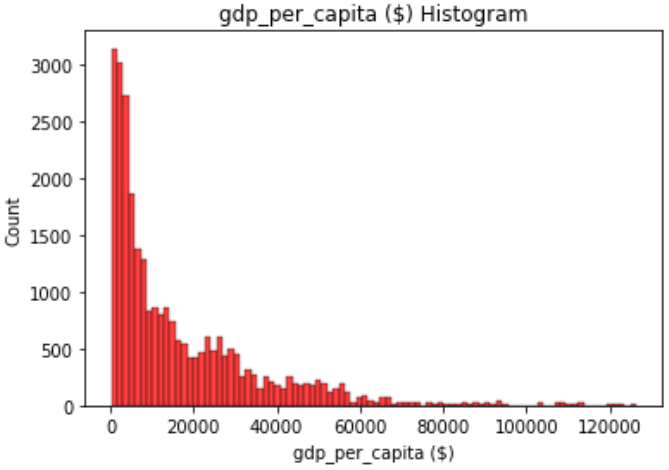

year Histogram
Count
year
0
200
400
600
800
1000
1985
1990
1995
2000
2005
2010
2015

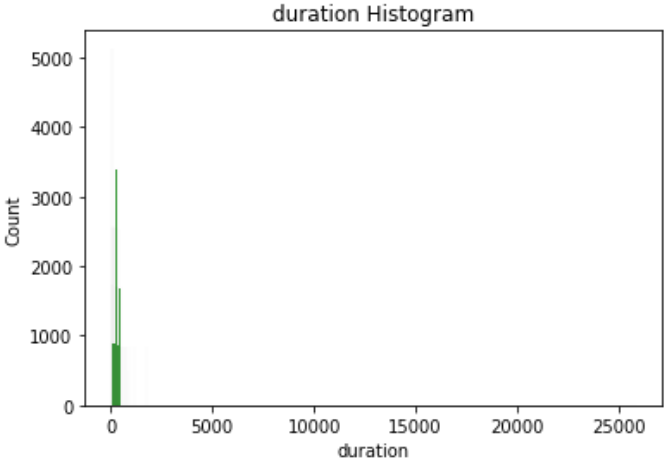

suicides_no Histogram
Count
suicides_no
0
2000
4000
6000
8000
10000
0
5000
10000
15000
20000

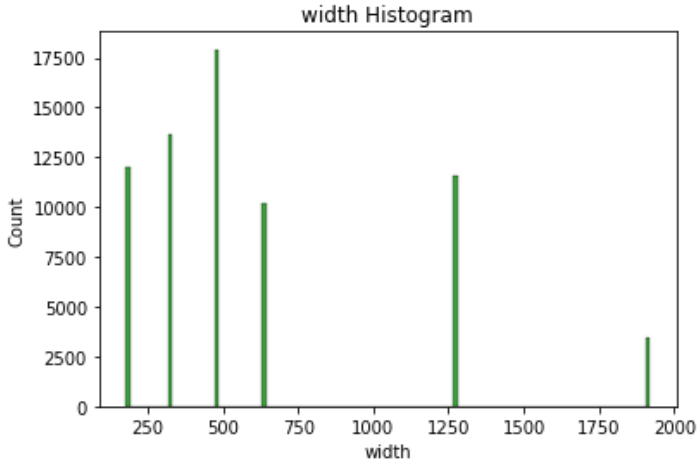

population Histogram
Count
population
1e7
0
1000
2000
3000
4000
5000
6000
7000
0
1
2
3
4

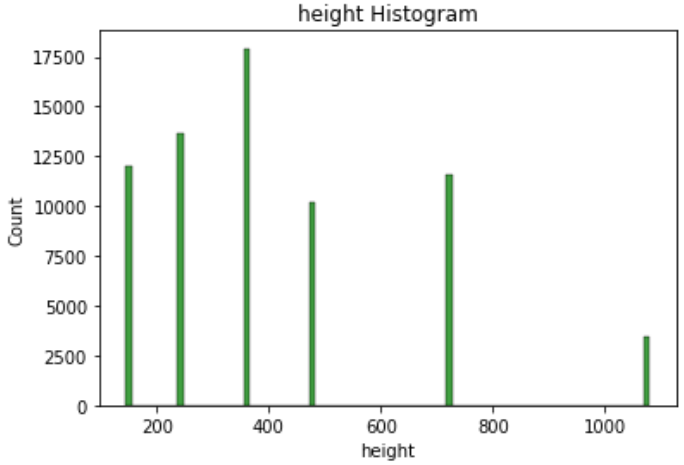

suicides/100k pop Histogram
Count
suicides/100k pop
0
1000
2000
3000
4000
5000
6000
7000
0
50
100
150
200

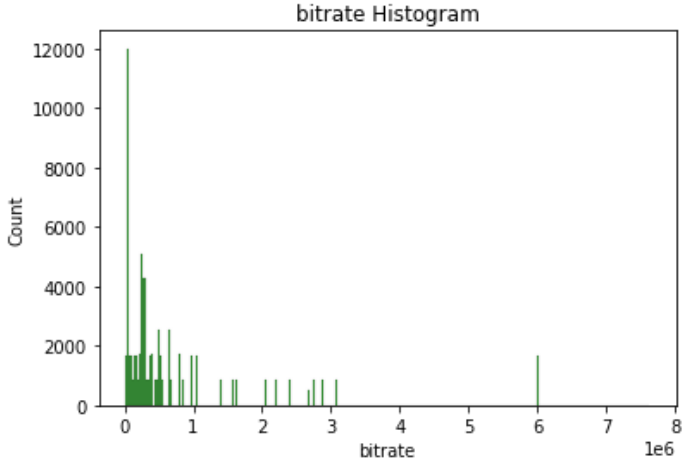

gdp_for_year ($) Histogram
Count
gdp_for_year ($)
1e13
0
2000
4000
6000
8000
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75

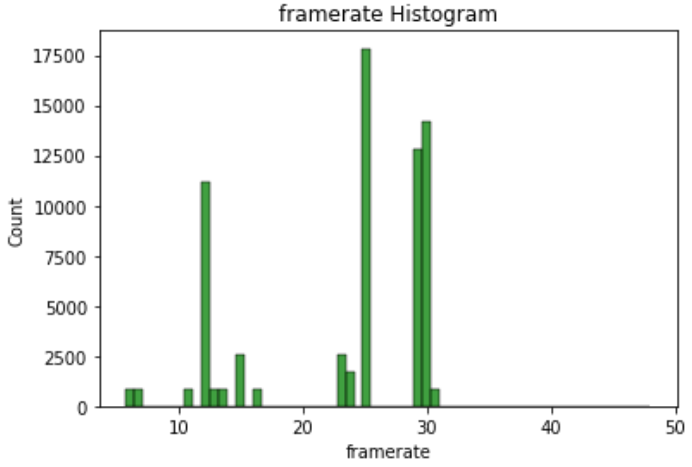

gdp_per_capita ($) Histogram
Count
gdp_per_capita ($)
0
500
1000
1500
2000
2500
3000
0
20000
40000
60000
80000
100000
120000

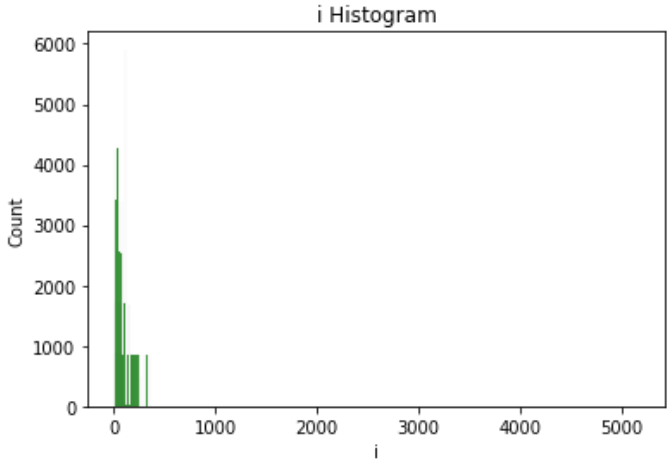

duration Histogram
Count
duration
0
1000
2000
3000
4000
5000
0
5000
10000
15000
20000
25000

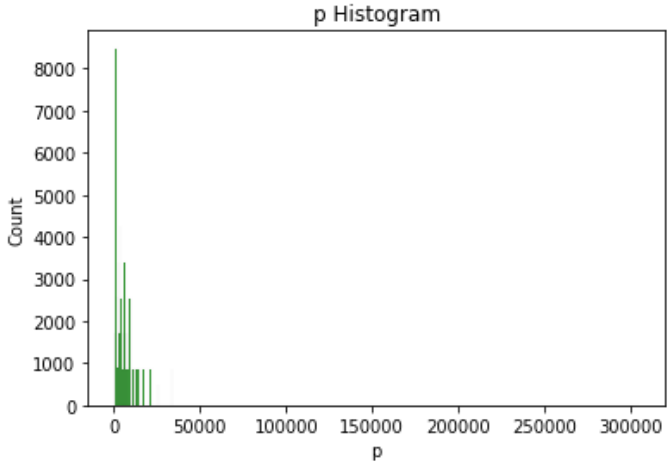

width Histogram
Count
width
0
2500
5000
7500
10000
12500
15000
17500
250
500
750
1000
1250
1500
1750
2000

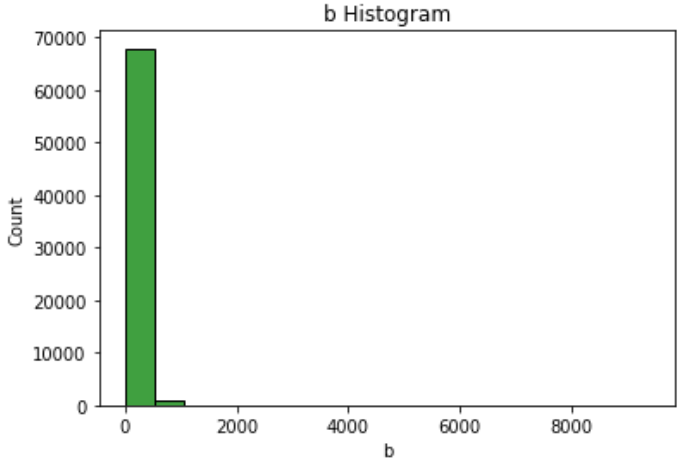

height Histogram
Count
height
0
2500
5000
7500
10000
12500
15000
17500
200
400
600
800
1000

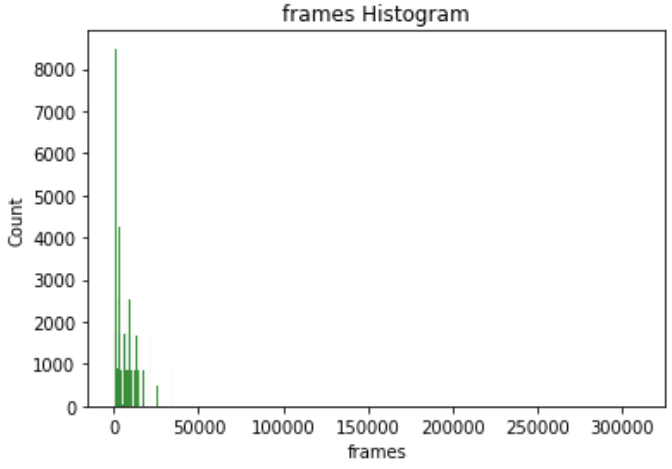

bitrate Histogram
Count
bitrate
1e6
0
2000
4000
6000
8000
10000
12000
0
1
2
3
4
5
6
7
8

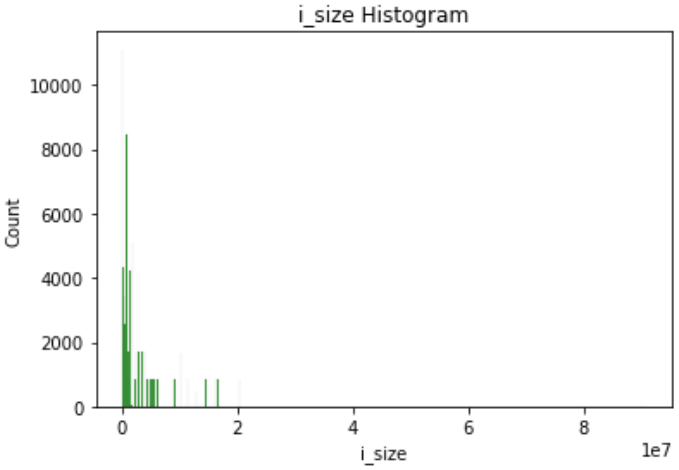

framerate Histogram
Count
framerate
0
2500
5000
7500
10000
12500
15000
17500
10
20
30
40
50

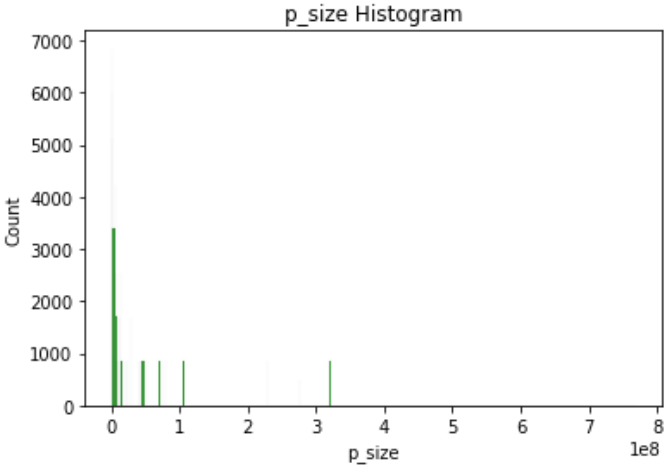

i Histogram
Count
i
0
1000
2000
3000
4000
5000
6000

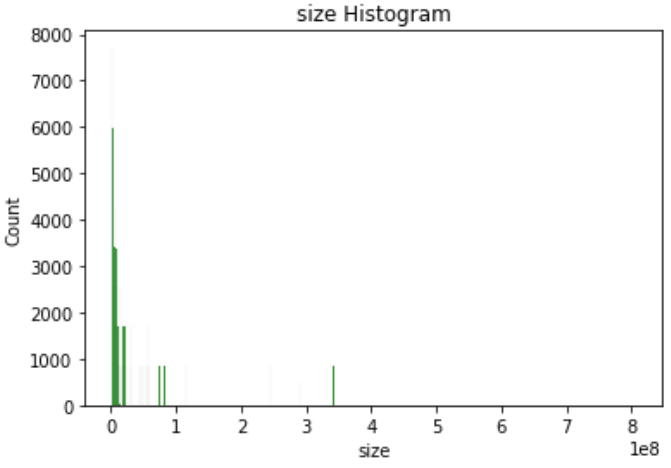

p Histogram
Count
p
0
50000
100000
150000
200000
250000
300000
1000
2000
3000
4000
5000
6000
7000
8000

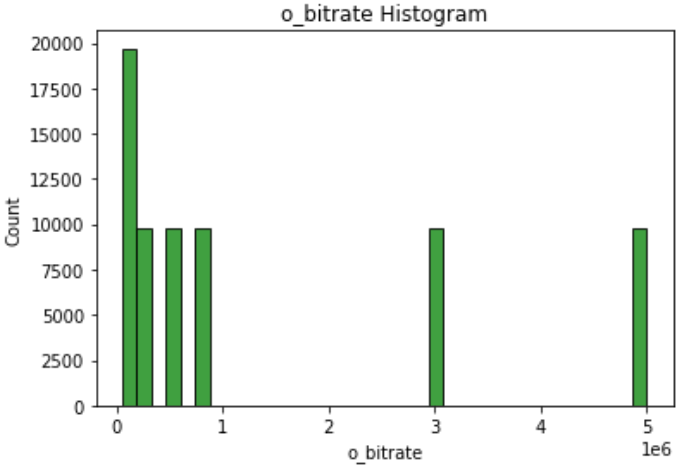

b Histogram
Count
b
0
2000
4000
6000
8000
10000
20000
30000
40000
50000
60000
70000

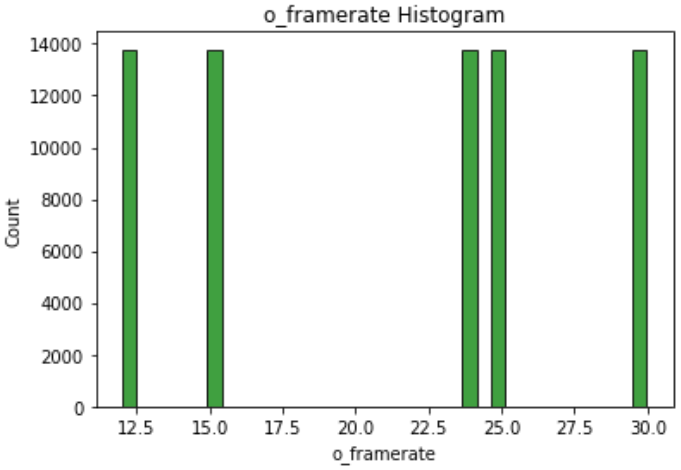

frames Histogram
Count
frames
0
50000
100000
150000
200000
250000
300000
1000
2000
3000
4000
5000
6000
7000
8000

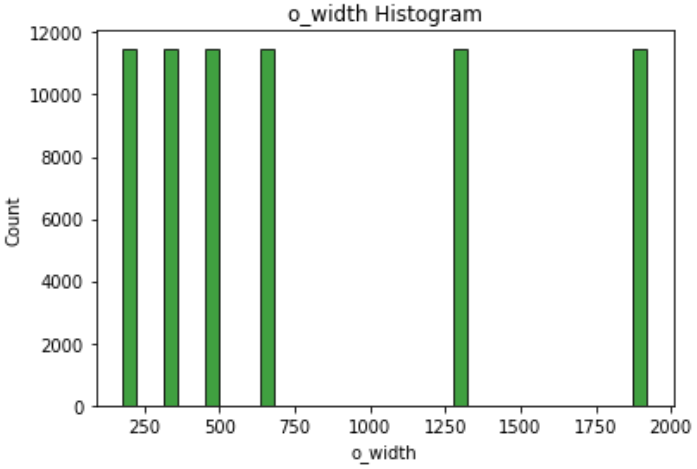

i_size Histogram
Count
i_size
1e7

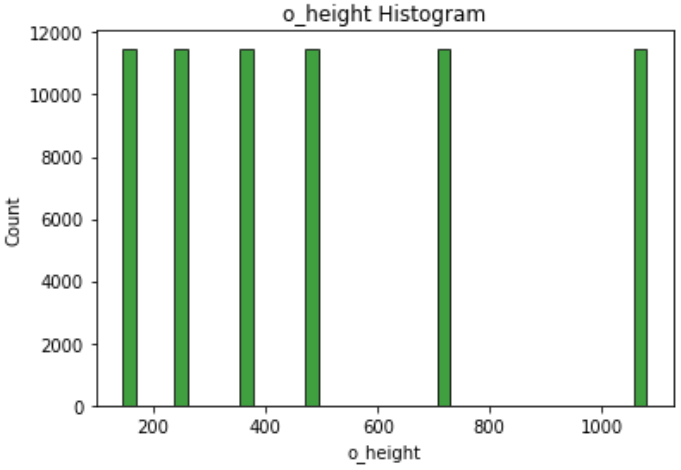

p_size Histogram
Count
p_size
1e8

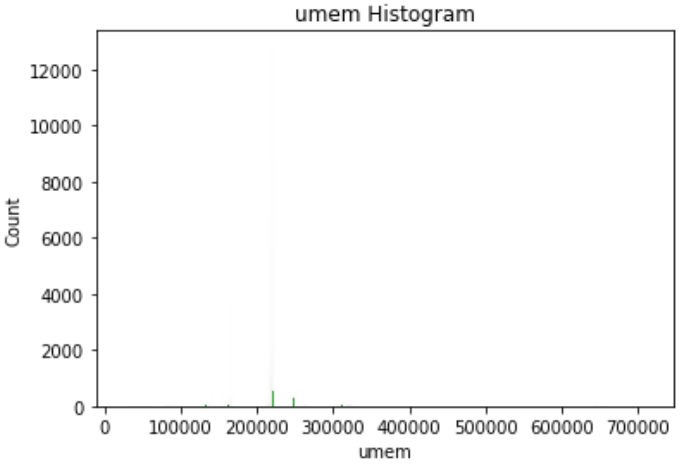

size Histogram
Count
size
1e8

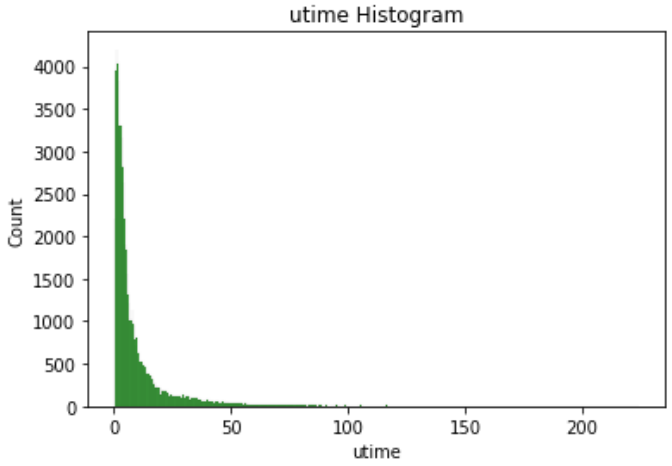

o_bitrate Histogram
Count
o_bitrate
1e6


o_framerate Histogram
Count
o_framerate
0
2000
4000
6000
8000
10000
12000
14000
12.5
15.0
17.5
20.0
22.5
25.0
27.5
30.0


o_width Histogram
Count
o_width
0
2000
4000
6000
8000
10000
12000
250
500
750
1000
1250
1500
1750
2000


o_height Histogram
Count
o_height
0
2000
4000
6000
8000
10000
12000
200
400
600
800
1000


umem Histogram
Count
umem
0
2000
4000
6000
8000
10000
12000
0
100000
200000
300000
400000
500000
600000
700000



Q3

Inspect box plot of categorical features vs target variable. What intuition do you get?

ANSWER

Intuitions on categorical variables:

- Bike Sharing
  - season: shows that spring and summer have the highest rental rates (unsurprising)
  - yr: suggests that the company who's data we have is growing (i.e. the latter year has higher overall rentals)
  - mnth: shows a more fine grained view of the seasonality seen earlier (i.e. warmer months have higher rentals)
  - holiday: shows that on average, rentals are lower for holidays; maybe due to more competing liesurely alternatives
  - weekday: shows no clear peaks on particular weekdays, although there seems to be a slight boost mid-week
  - workingday: same as above, no clear trends favoring working days over non-working
  - weathersit: shows strong decrease in rentals when the weather is inclement (unsurprising)
- Suicides
  - country: shows that a lot of countries connected to the former USSR have higher suicide rates
  - year: shows that suicides peaked around 1995 and were decreasing until about 2015
  - sex: shows that men generally commit suicide at higher rates than women
  - age: shows that the older someone is, the more likely they are to commit suicide
  - generation: shows a similar trend to the age feature
- Video Transcoding
  - codec: shows relatively consistent transcoding time between codec options
  - o_codec: shows h264 and vp8 have both longer and more varied transcoding times; flv and mpeg are more consistent

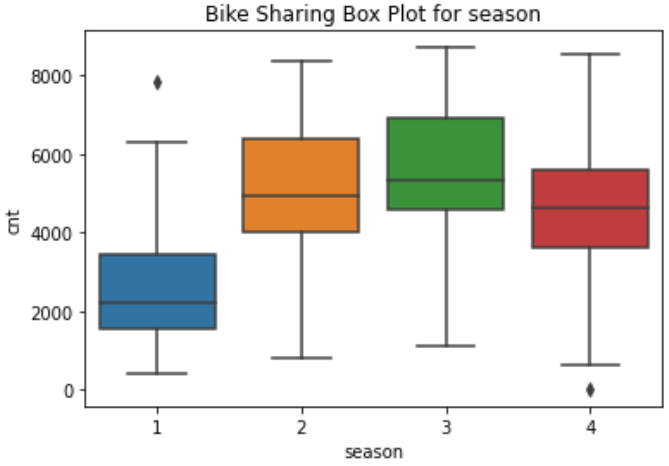
In [37]:
```
def convert_categorical(df, cols):
    for c in cols:
        df[c] = pd.Categorical(df[c])
    return df
```

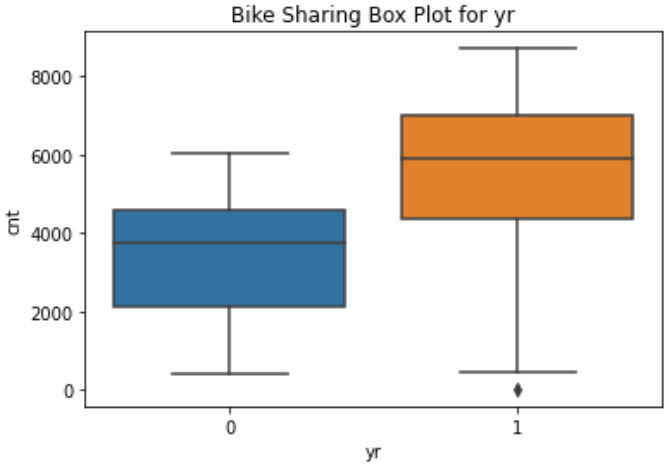
In [38]:
```
cat_bik = ['season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday', 'weathersit']
df_bik = convert_categorical(df_bik, cat_bik)
```

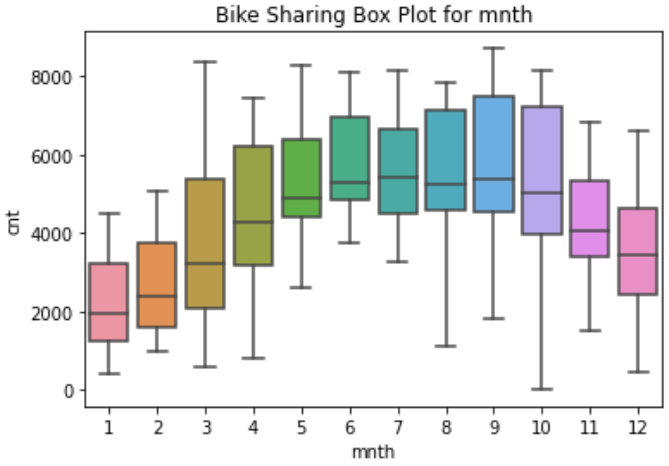
In [40]:
```
cat_sui = ['country', 'year', 'sex', 'age', 'generation']
df_sui = convert_categorical(df_sui, cat_sui)
```

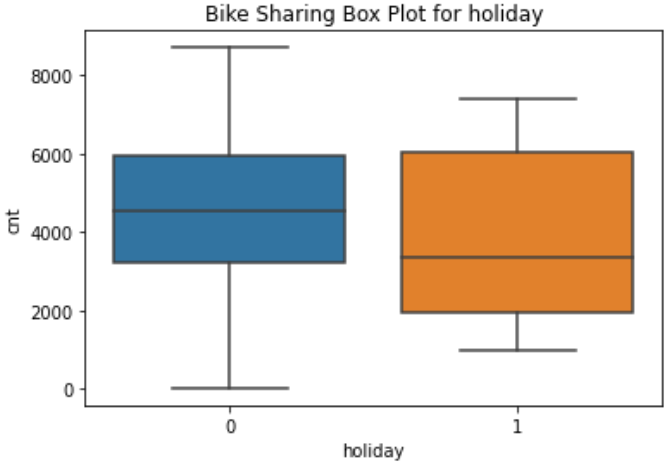
In [44]:
```
cat_vid = ['codec', 'o_codec']
df_vid = convert_categorical(df_vid, cat_vid)
```

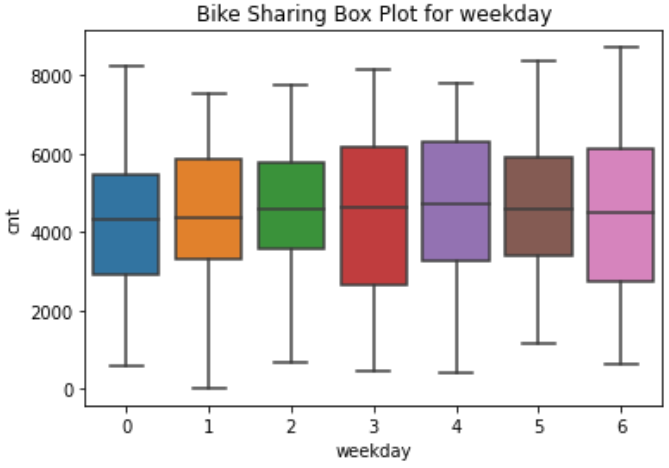
In [32…
```
y_bik = 'cnt'
y_sui = 'suicides/100k pop'
y_vid = 'utime'

dfs = [df_bik, df_sui, df_vid]
cats = [cat_bik, cat_sui, cat_vid]
ys = [y_bik, y_sui, y_vid]


for d, df, cat, y in zip(datasets, dfs, cats, ys):
    for c in cat:
        g = sns.boxplot(data=df, x=c, y=y, orient='v')
        if len(g.get_xticklabels()) > 12 or c == 'age' or c == 'generation':
            g.set_xticklabels(g.get_xticklabels(),rotation=90)
        g.set_title(d + ' Box Plot for ' + c)
        plt.show()
```

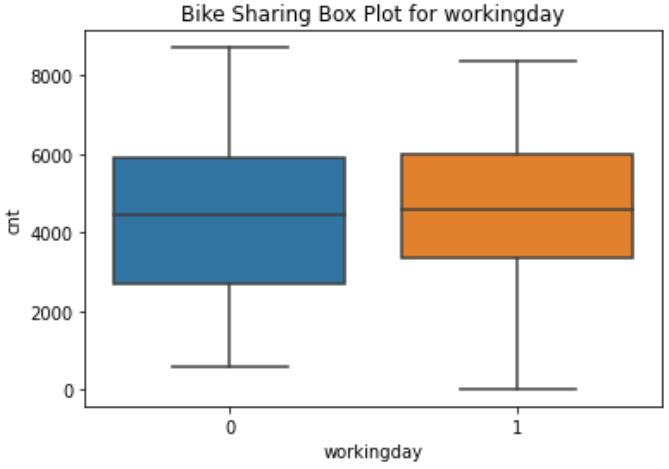


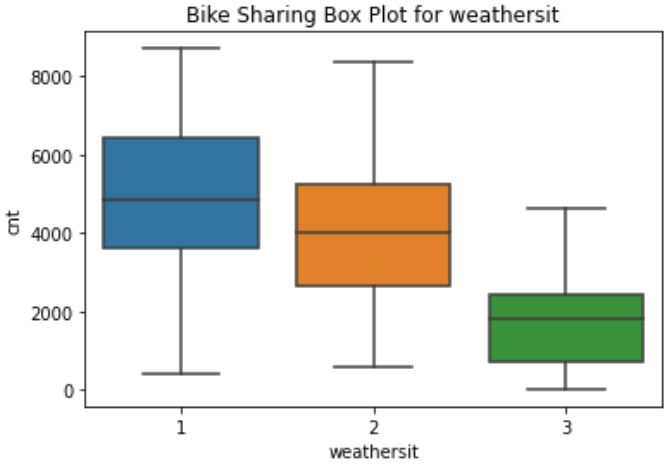


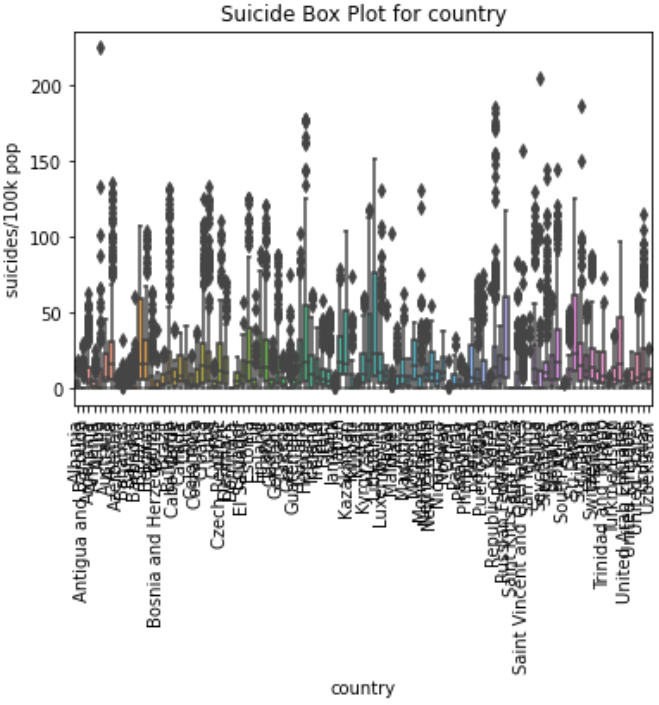

Bike Sharing Box Plot for mnth
cnt
mnth
0
2000
4000
6000
8000
1
2
3
4
5
6
7
8
9
10
11
12

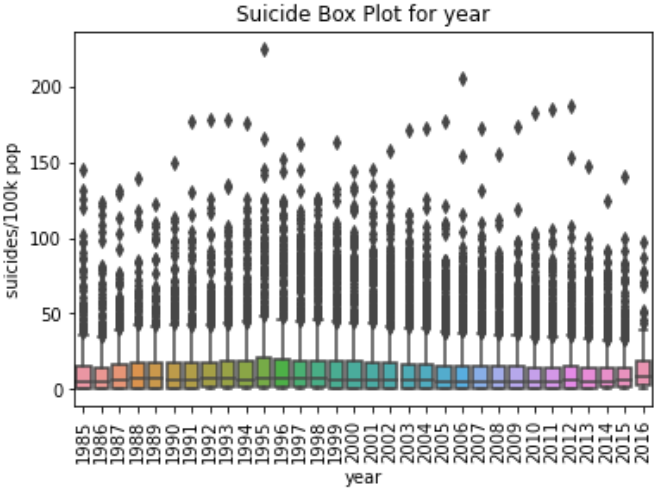

Bike Sharing Box Plot for holiday
cnt
holiday
0
2000
4000
6000
8000
0
1

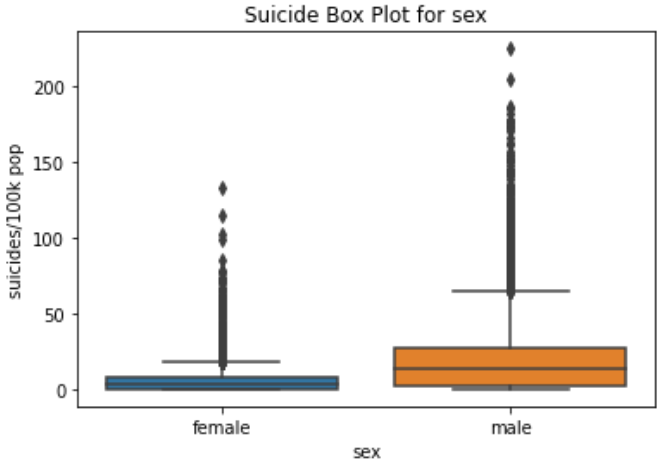

Bike Sharing Box Plot for weekday
cnt
weekday
0
2000
4000
6000
8000
0
1
2
3
4
5
6

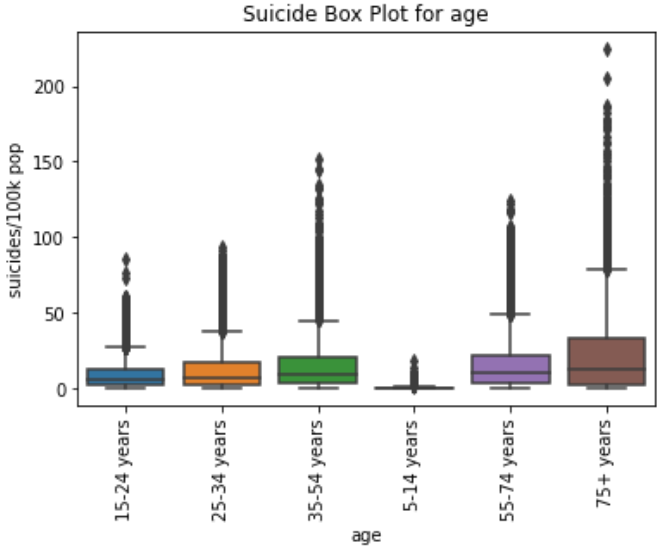

Bike Sharing Box Plot for workingday
cnt
workingday
0
2000
4000
6000
8000
0
1

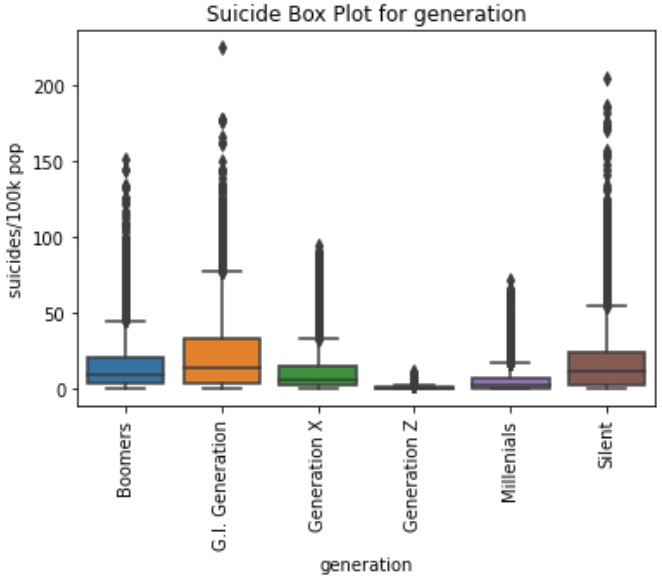

Bike Sharing Box Plot for weathersit
cnt
8000
6000
4000
2000
0
1
2
3
weathersit

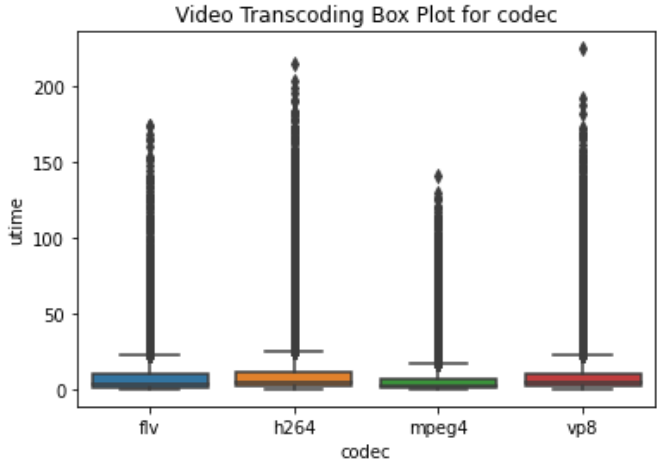

Suicide Box Plot for country
suicides/100k pop
200
150
100
50
0
country

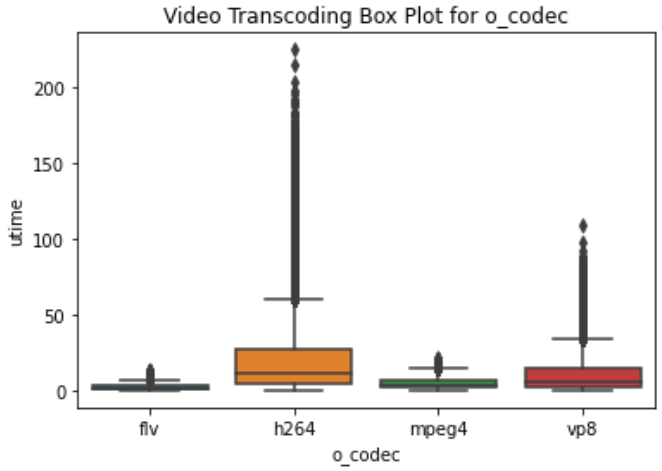

Suicide Box Plot for year
suicides/100k pop
200
150
100
50
0
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
year


Suicide Box Plot for sex
suicides/100k pop
0
50
100
150
200
female
male
sex


Suicide Box Plot for age
suicides/100k pop
0
50
100
150
200
15-24 years
25-34 years
35-54 years
5-14 years
55-74 years
75+ years
age

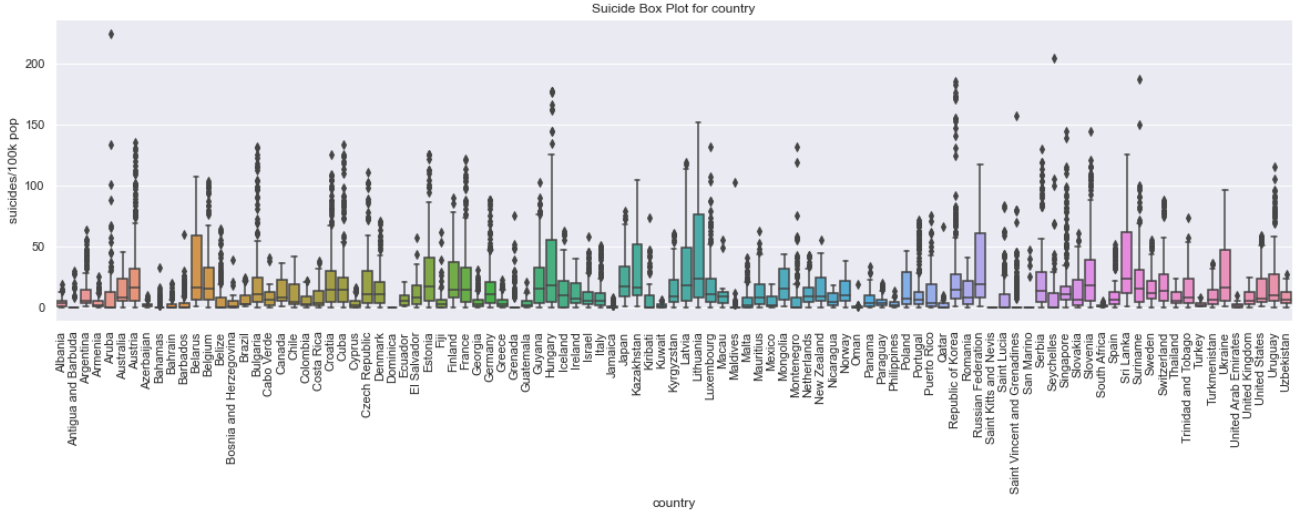

Suicide Box Plot for generation
suicides/100k pop
0
50
100
150
200
Boomers
G.I. Generation
Generation X
Generation Z
Millenials
Silent
generation





In [32…

```
sns.set(rc={'figure.figsize':(20,5)})
g = sns.boxplot(data=df_sui, x='country', y=y_sui)
g.set_xticklabels(df_sui['country'].unique(), rotation=90)
g.set_title('Suicide Box Plot for country')
plt.show()
sns.reset_orig()
```



Q4

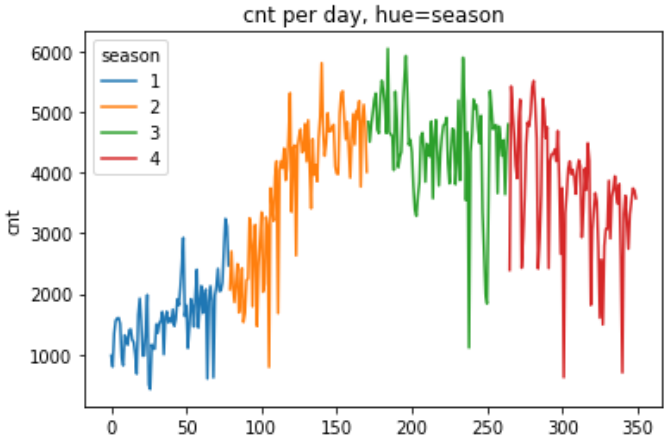
For bike sharing dataset, plot the count number per day for a few months. Can you identify any repeating patterns in every month?

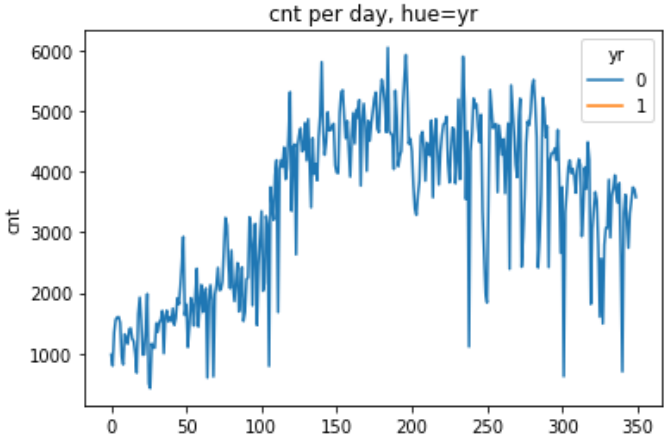
ANSWER

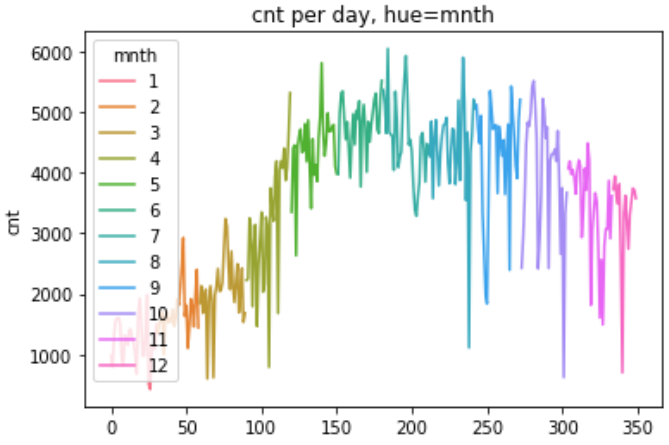
Saturdays and Sundays typically have the highest rental rates, followed by Mondays. The weekend use is probably for casual liesure and the Monday rentals might be more likely to be for work commutes.

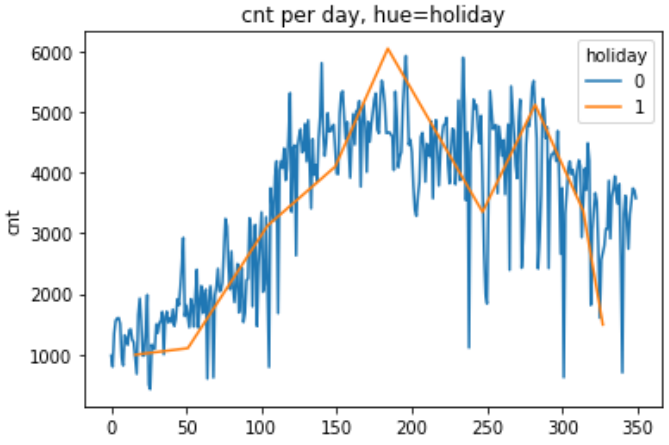
In [99]:
```
def plot_bik(df, plot_type='scatter', first_n=0, target='cnt'):
    if plot_type == 'scatter':
        plot = sns.scatterplot
    elif plot_type == 'line':
        plot = sns.lineplot
    else:
        raise ValueError("Unsupported plot type. Choose one: 'line', 'scatter'")
    if first_n > 0:
        df = df.head(first_n)
    for c in df.select_dtypes('category').columns:
        g = plot(data=df, x=df.index, y=target, hue=c)
        g.set_title(target + ' per day, hue=' + c)
        plt.show()
```

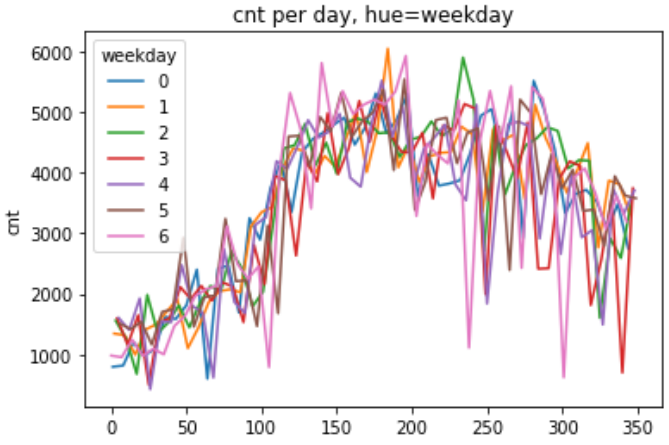
In [11…
```
plot_bik(df_bik, plot_type='line', target='cnt', first_n=350)
```

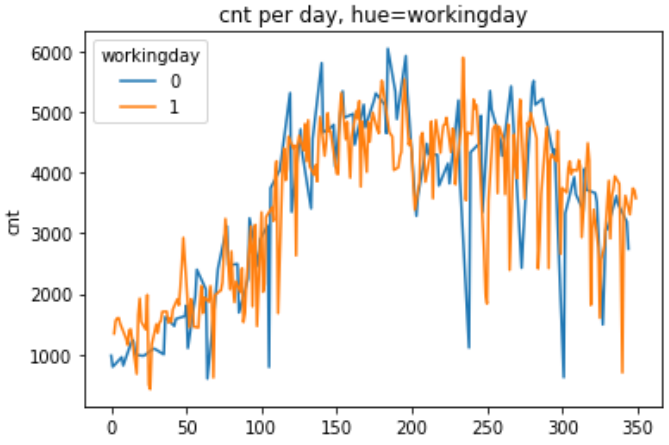


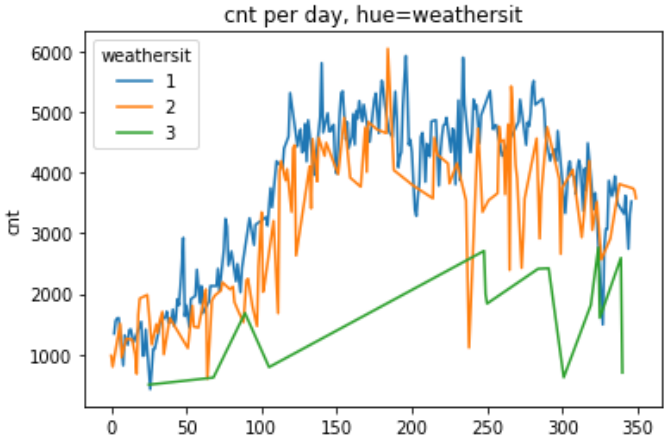



cnt per day, hue=mnth
mnth
1
2
3
4
5
6
7
8
9
10
11
12
cnt
6000
5000
4000
3000
2000
1000
0
50
100
150
200
250
300
350


cnt per day, hue=holiday
holiday
0
1
cnt
6000
5000
4000
3000
2000
1000
0
50
100
150
200
250
300
350

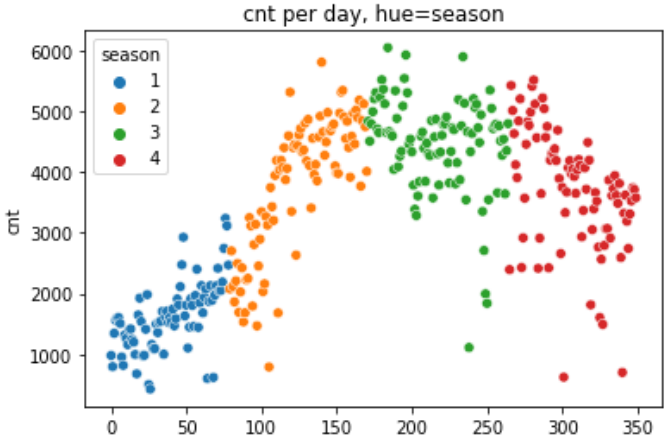

cnt per day, hue=weekday
weekday
0
1
2
3
4
5
6
cnt
6000
5000
4000
3000
2000
1000
0
50
100
150
200
250
300
350

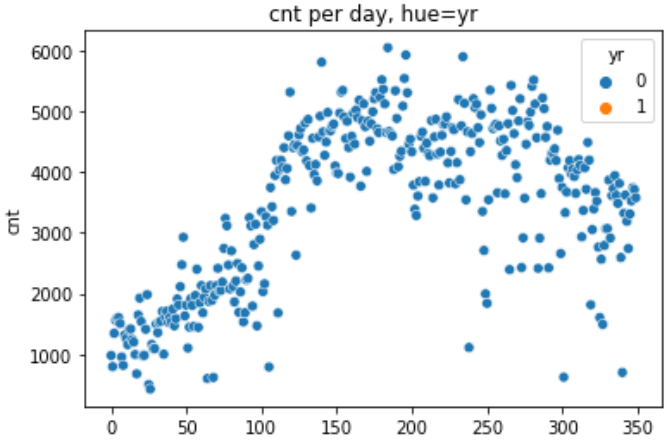

cnt per day, hue=workingday
workingday
0
1
cnt
6000
5000
4000
3000
2000
1000
0
50
100
150
200
250
300
350

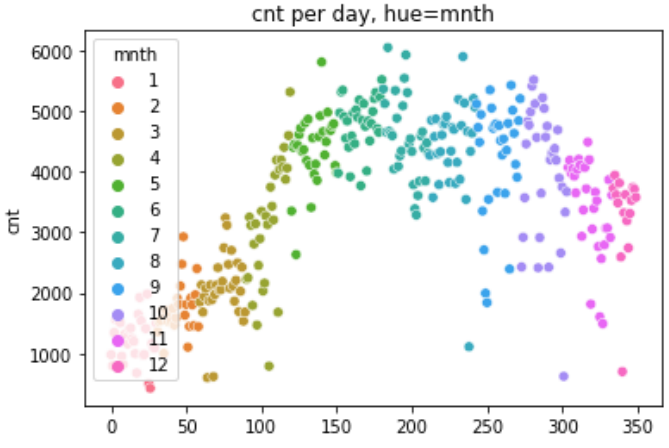


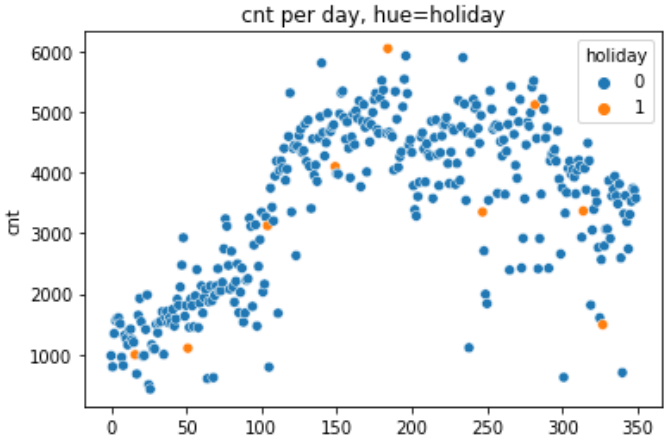
In [11…

```
plot_bik(df_bik, plot_type='scatter', target='cnt', first_n=350)
```








cnt per day, hue=holiday
cnt
holiday
0
1
6000
5000
4000
3000
2000
1000
0
50
100
150
200
250
300
350

Project 4: Regression

ECE 219: Large-Scale Data Mining: Models and Algorithms [Winter 2021]

Prof. Vwani Roychowdhury

UCLA, Department of ECE

Due: 2021.03.19 11:59PM PT

In [11…

```
#!pip install pandas-profiling[notebook]
```

In [55]:

```
from pandas_profiling import ProfileReport
from IPython.display import display
from tqdm.notebook import tqdm
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd

tqdm.pandas()
```

Q 10 - 20

scoring in recursive feature selection

Data-loading and pre-processing

In [11…

```
BIKE_SHARING_DATA = "C:\\Work\\UCLA\\Winter 2021\\219 Large Scale Data Mining Models and Algorithms\\Proje
SUICIDE_DATA = "C:\\Work\\UCLA\\Winter 2021\\219 Large Scale Data Mining Models and Algorithms\\Project_4\
VIDEO_TRANSCODING_DATA = "C:\\Work\\UCLA\\Winter 2021\\219 Large Scale Data Mining Models and Algorithms\\
```

In [ ]:

```
Ys_bik = ['cnt']
df_bik = pd.read_csv(BIKE_SHARING_DATA)
df_bik.drop(columns=['instant', 'dteday','casual', 'registered'], inplace=True)
```

In [11…

```
Xs_sui = [
    'country',
    'year',
    'sex',
    'age',
    'population',
    ' gdp_for_year ($) ',
    'gdp_per_capita ($)',
    'generation'
]
Ys_sui = ['suicides/100k pop']
df_sui = pd.read_csv(SUICIDE_DATA, usecols=Xs_sui+Ys_sui)
df_sui.reset_index(drop=True, inplace=True)
df_sui.columns = [col.strip() for col in df_sui.columns]
Xs_sui = [col.strip() for col in Xs_sui]
df_sui['gdp_for_year ($)'] = df_sui['gdp_for_year ($)'].map(lambda x: int(x.replace(',', "")))
df_sui.dtypes
```

Out[119]:

```
country              object
year                  int64
sex                  object
age                  object
population            int64
```

```
suicides/100k pop     float64
gdp_for_year ($)        int64
gdp_per_capita ($)      int64
generation             object
dtype: object
```

In [12…

```
Ys_vid = ['utime']
df_vid = pd.read_csv(VIDEO_TRANSCODING_DATA, sep='\t')
df_vid.drop(columns=['id', 'b_size'], inplace=True)
df_vid.dtypes
```

Out[120]:

```
duration       float64
codec           object
width            int64
height           int64
bitrate          int64
framerate      float64
i                int64
p                int64
b                int64
frames           int64
i_size           int64
p_size           int64
size             int64
o_codec         object
o_bitrate        int64
o_framerate    float64
o_width          int64
o_height         int64
umem             int64
utime          float64
dtype: object
```

In [12…

```
dfs = [df_bik, df_sui, df_vid]
Ys = [Ys_bik, Ys_sui, Ys_vid]

datasets = ['Bike Sharing', 'Suicide', 'Video Transcoding']
colors = ['blue', 'red', 'green']
```

Changing countries to continents in df_sui

In [11…

```
import pycountry_convert as pc
import country_converter as coco
```

In [11…

```
def country2continent(country_name):
    try:
        country_code = coco.convert(names=country_name, to='ISO2') # alpha2 == ISO2
        return pc.country_alpha2_to_continent_code(country_code)
    except:
        return "N/A"
```

In [ ]:

```
df_sui['continent'] = df_sui['country'].progress_apply(lambda x: country2continent(x)) #Already ran this
```

In [ ]:

```
df_sui = df_sui.drop(columns=['country'])
```

In [ ]:

```
def normalize_df(df, columns):
    df[columns] = (df[columns]-df[columns].mean())/df[columns].std()
    return df
```

In [ ]:

```
norm_bik = ['temp','atemp', 'hum', 'windspeed']
df_bik = normalize_df(df_bik, norm_bik)
```

In [ ]:
```
norm_sui = ['population', 'gdp_for_year ($)', 'gdp_per_capita ($)']
df_sui = normalize_df(df_sui, norm_sui)
```

In [ ]:
```
norm_vid = [
    'duration',
    'width',
    'height',
    'bitrate',
    'framerate',
    'i',
    'p',
    'frames',
    'i_size',
    'p_size',
    'size',
    'o_bitrate',
    'o_framerate',
    'o_width',
    'o_height'
]
df_vid = normalize_df(df_vid, norm_vid)
```

In [ ]:
```
norm_names = [norm_bik, norm_sui, norm_vid]
```

In [ ]:
```
#df_bik.to_csv('C:\\Work\\UCLA\\Winter 2021\\219 Large Scale Data Mining Models and Algorithms\\Project_4\
#df_sui.to_csv('C:\\Work\\UCLA\\Winter 2021\\219 Large Scale Data Mining Models and Algorithms\\Project_4\
#df_vid.to_csv('C:\\Work\\UCLA\\Winter 2021\\219 Large Scale Data Mining Models and Algorithms\\Project_4\
```

In [57]:
```
df_bik = pd.read_csv('C:\\Work\\UCLA\\Winter 2021\\219 Large Scale Data Mining Models and Algorithms\\Proj

df_sui = pd.read_csv('C:\\Work\\UCLA\\Winter 2021\\219 Large Scale Data Mining Models and Algorithms\\Proj

df_vid = pd.read_csv('C:\\Work\\UCLA\\Winter 2021\\219 Large Scale Data Mining Models and Algorithms\\Proj
df_vid = df_vid.drop(columns=['umem'])
```

In [58]:
```
y_bik = 'cnt'
y_sui = 'suicides/100k pop'
y_vid = 'utime'
ys = [y_bik, y_sui, y_vid]
```

In [59]:
```
dfs = [df_bik, df_sui, df_vid]
```

In [60]:
```
datasets = ['Bike Sharing', 'Suicide', 'Video Transcoding']
```

In [61]:
```
cat_bik = ['season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday', 'weathersit']
cat_bik_mask = np.array([1,1,1,1,1,1,1,0,0,0,0])

cat_sui = ['year', 'sex', 'age', 'generation','continent']
cat_sui_mask = np.array([1,1,1,0,0,0,1,1])

cat_vid = ['codec', 'o_codec']
cat_vid_mask = np.array([0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0])

cat_masks = [cat_bik_mask, cat_sui_mask, cat_vid_mask]
cat_names = [cat_bik, cat_sui, cat_vid]
```

In [62]:
```
#To see the names of categorical features
for d, df, y, cmask in zip(datasets, dfs, ys, cat_masks):
```

```
    print(d)
    X = df.drop(columns=y)
    print(X.columns[cmask == 1])
```

```
Bike Sharing
Index(['season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday',
       'weathersit'],
      dtype='object')
Suicide
Index(['year', 'sex', 'age', 'generation', 'continent'], dtype='object')
Video Transcoding
Index(['codec', 'o_codec'], dtype='object')
```

In [63]:
```
def convert_categorical(df, cols):
    for c in cols:
        df[c] = pd.Categorical(df[c])
    return df
```

In [64]:
```
df_bik = convert_categorical(df_bik, cat_bik)
df_sui = convert_categorical(df_sui, cat_sui)
df_vid = convert_categorical(df_vid, cat_vid)
```

In [65]:
```
dfs = [df_bik, df_sui, df_vid]
```

Feature selection

In [66]:
```
from sklearn.preprocessing import OrdinalEncoder
```

In [67]:
```
from sklearn.feature_selection import f_classif
from sklearn.feature_selection import mutual_info_regression
from sklearn.feature_selection import f_regression
```

In [68]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_validate
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import train_test_split
```

In [69]:
```
from sklearn.metrics import SCORERS
from sklearn.model_selection import KFold
from sklearn.feature_selection import RFECV
from sklearn.preprocessing import PolynomialFeatures
import warnings
warnings.filterwarnings("ignore")
```

In [70]:
```
print(df_bik.isnull().sum().sum())
print(df_sui.isnull().sum().sum())
print(df_vid.isnull().sum().sum())
```

```
0
0
0
```

Scalar encoding the categorical features before computing F scores and MI

In [71]:
```
def enc_df(df, cat):
    X = df.copy(deep = True)
    enc = OrdinalEncoder()
```

```
    enc.fit(X[cat])
    X[cat] = enc.transform(X[cat])

    #enc is returned so as to transform any test data, if needed.
    return X, enc
```

In [72]:
```
df_bikenc, enc_bik = enc_df(df_bik, cat_bik)
df_suienc, enc_sui = enc_df(df_sui, cat_sui)
df_videnc, enc_vid = enc_df(df_vid, cat_vid)
```

In [73]:
```
dfs_enc = [df_bikenc, df_suienc, df_videnc]
obj_enc = [enc_bik, enc_sui, enc_vid]
```

In [13…
```
results = {}

for d, df, y, cmask in zip(datasets, dfs_enc, ys, cat_masks):
    print(d)

    X = df.drop(columns=y)
    y = df[y]

    # mutual information
    mi = mutual_info_regression(X, y, discrete_features = cmask)
    mi /= np.max(mi)

    #fregression
    f_test, pvals= f_regression(X,y)
    f_test /= np.max(f_test)

    df_f = pd.DataFrame(list(zip(X.columns, mi, f_test, pvals)), columns=['feature', 'mi','f_reg','pvals']

    results[d] = df_f

    display(df_f)
```

Bike Sharing

	feature	mi	f_reg	pvals
0	season	0.465988	0.298407	2.133997e-30
1	yr	0.592423	0.714867	2.483540e-63
2	mnth	0.810982	0.128519	1.243112e-14
3	holiday	0.024801	0.007092	6.475936e-02
4	weekday	0.094525	0.006904	6.839081e-02
5	workingday	0.053017	0.005673	9.849496e-02
6	weathersit	0.142324	0.146603	2.150976e-16
7	temp	0.834838	0.981381	2.810622e-81
8	atemp	1.000000	1.000000	1.854504e-82
9	hum	0.098948	0.015467	6.454143e-03
10	windspeed	0.120754	0.087962	1.359959e-10

Suicide

	feature	mi	f_reg	pvals
0	year	0.000000	0.008431	7.353434e-11
1	sex	0.227132	1.000000	0.000000e+00
2	age	0.458164	0.200661	7.739320e-218

	feature	mi	f_reg	pvals
3	population	1.000000	0.000379	1.670210e-01
4	gdp_for_year ($)	0.374965	0.003522	2.550492e-05
5	gdp_per_capita ($)	0.323343	0.000018	7.659053e-01
6	generation	0.261683	0.013746	9.218445e-17
7	continent	0.149859	0.011214	5.891869e-14

Video Transcoding

	feature	mi	f_reg	pvals
0	duration	1.000000	0.000081	1.467846e-01
1	codec	0.189577	0.000900	1.351842e-06
2	width	0.435609	0.045465	2.213683e-256
3	height	0.431167	0.044485	5.955923e-251
4	bitrate	0.953942	0.065419	0.000000e+00
5	framerate	0.534009	0.016789	1.889575e-96
6	i	0.875387	0.000906	1.238180e-06
7	p	0.962952	0.002925	3.047665e-18
8	b	0.002121	0.000070	1.776859e-01
9	frames	0.956993	0.002910	3.718494e-18
10	i_size	0.940567	0.011146	9.874803e-65
11	p_size	0.953539	0.025514	2.541518e-145
12	size	0.952545	0.025226	1.043438e-143
13	o_codec	0.824045	0.013240	1.631317e-76
14	o_bitrate	0.057788	0.065660	0.000000e+00
15	o_framerate	0.047011	0.029005	8.053878e-165
16	o_width	0.908783	1.000000	0.000000e+00
17	o_height	0.917212	0.980497	0.000000e+00

Asumming a cut-off of 0.1, train on the dataset after dropping features that had **both** MI and Fscore less than 0.1

Suicide dataset shows a lot of difference between MI and F scores for each feature.

In [74]:
```
drop_bikmf = ['holiday', 'weekday', 'workingday']
drop_suimf = ['year']
drop_vidmf = ['b', 'o_bitrate', 'o_framerate']
drop_featmf = [drop_bikmf, drop_suimf, drop_vidmf]
```

In [75]:
```
#df_bik_oh.columns
```

In [76]:
```
def plot_regression(scores,title):
    plt.plot(-scores['test_score'], label = 'Avg VALID MSE = {:.4f}'.format(np.mean(-scores['test_score']]
    plt.plot(-scores['train_score'],label = 'Avg TRAIN MSE = {:.4f}'.format(np.mean(-scores['train_score'
    plt.ylabel('MSE')
    plt.xlabel('Fold')
    plt.title(title)
    plt.legend()
    plt.show()
```

In [77]:
```
def Reg_CV_plot(d, df, Y, drp,alphas, title):
    X = df.drop(columns = [Y] + drp)
    X = pd.get_dummies(X)
    y = df[Y]

    lin_reg = LinearRegression()
    lin_reg_score = cross_validate(lin_reg, X, y, scoring="neg_mean_squared_error", cv=10, return_train_sc
    plot_regression(lin_reg_score,d +' Least Squares'+ title)

    rid_scores={}
    las_scores={}

    for alpha in alphas:

        rid_reg = Ridge(alpha = alpha)
        rid_scores[str(alpha)] = cross_validate(rid_reg, X, y, scoring="neg_mean_squared_error", cv=10, re
        plot_regression(rid_scores[str(alpha)],d +' Ridge'+title+' alpha={:0.4f} '.format(alpha))

        las_reg = Lasso(alpha = alpha)
        las_scores[str(alpha)] = cross_validate(las_reg, X, y, scoring="neg_mean_squared_error", cv=10, re
        plot_regression(las_scores[str(alpha)],d +' Lasso'+title+' alpha={:0.4f} '.format(alpha))
```

In [78]:
```
def Reg_CV_rmse(d, df, Y, drp,alphas, title):
    X = df.drop(columns = [Y] + drp)
    X = pd.get_dummies(X)
    y = df[Y]

    lin_reg = LinearRegression()
    scores = cross_validate(lin_reg, X, y, scoring="neg_mean_squared_error", cv=10, return_train_score=Tr
    print(d+" Least Squares "+title+" Avg Train RMSE {:0.3f}".format(np.sqrt(np.mean(-scores['train_score
    print(d+" Least Squares "+title+" Avg Test RMSE {:0.3f}".format(np.sqrt(np.mean(-scores['test_score']

    rid_scores={}
    las_scores={}

    rid_train_rmse = []
    rid_test_rmse = []
    las_train_rmse = []
    las_test_rmse = []

    for alpha in alphas:

        rid_reg = Ridge(alpha = alpha,max_iter = 1e5)
        scores = cross_validate(rid_reg, X, y, scoring="neg_mean_squared_error", cv=10, return_train_score
        rid_train_rmse.append(np.sqrt(np.mean(-scores['train_score'])))
        rid_test_rmse.append(np.sqrt(np.mean(-scores['test_score'])))

        las_reg = Lasso(alpha = alpha, max_iter = 1e5)
        scores = cross_validate(las_reg, X, y, scoring="neg_mean_squared_error", cv=10, return_train_score
        las_train_rmse.append(np.sqrt(np.mean(-scores['train_score'])))
        las_test_rmse.append(np.sqrt(np.mean(-scores['test_score'])))

    dframe = pd.DataFrame({'Ridge Train RMSE':rid_train_rmse, 'Ridge Valid RMSE':rid_test_rmse}, index = a
    dframe.index.name = 'Alpha'
    print(d+" Ridge Rigression "+title)
    display(dframe)

    dframe = pd.DataFrame({'Lasso Train RMSE':las_train_rmse, 'Lasso Valid RMSE':las_test_rmse}, index = a
    dframe.index.name = 'Alpha'
    print(d+" Lasso Rigression "+title)
    display(dframe)

    plt.plot(alphas, rid_test_rmse, label = 'Ridge VALID RMSE')
    plt.plot(alphas, rid_train_rmse, label = 'Ridge TRAIN RMSE')
    plt.ylabel('Average RMSE')
    plt.xlabel('Alpha')
    plt.xscale("log")
    plt.title(d+" Ridge Rigression "+title)
    plt.legend()
```

```
    plt.show()

    plt.plot(alphas, las_test_rmse, label = 'Lasso VALID RMSE')
    plt.plot(alphas, las_train_rmse, label = 'Lasso TRAIN RMSE')
    plt.ylabel('Average RMSE')
    plt.xlabel('Alpha')
    plt.xscale("log")
    plt.title(d+" Lasso Rigression "+title)
    plt.legend()
    plt.show()
```

In [14…

```
#BIKE dataset
alphas = np.logspace(-6, 6, 13)
Reg_CV_rmse(datasets[0],dfs[0], ys[0], drop_featmf[0], alphas, title = "(considered MI and F scores)")
```

```
Bike Sharing Least Squares (considered MI and F scores) Avg Train RMSE 768.539
Bike Sharing Least Squares (considered MI and F scores) Avg Test RMSE 927.327
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5666766.
634627461, tolerance: 199790.9642304414
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1497539
7.380360126, tolerance: 255245.10859148938
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1047121
8.04866147, tolerance: 271504.2992972644
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1494892
9.96444273, tolerance: 266874.350268845
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1457775
5.484047055, tolerance: 252850.1612019757
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5878363.
967967868, tolerance: 259618.5997721885
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 8264977.
2122491, tolerance: 230120.4946808511
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1413768
8.88105452, tolerance: 215148.17779042554
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5464024.
677106023, tolerance: 248149.695762462
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5666788.
498111308, tolerance: 199790.9642304414
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1497542
5.509460986, tolerance: 255245.10859148938
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1047124
1.244420707, tolerance: 271504.2992972644
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1494896
0.302278101, tolerance: 266874.350268845
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1457778
6.02229768, tolerance: 252850.1612019757
  model = cd_fast.enet_coordinate_descent(
```

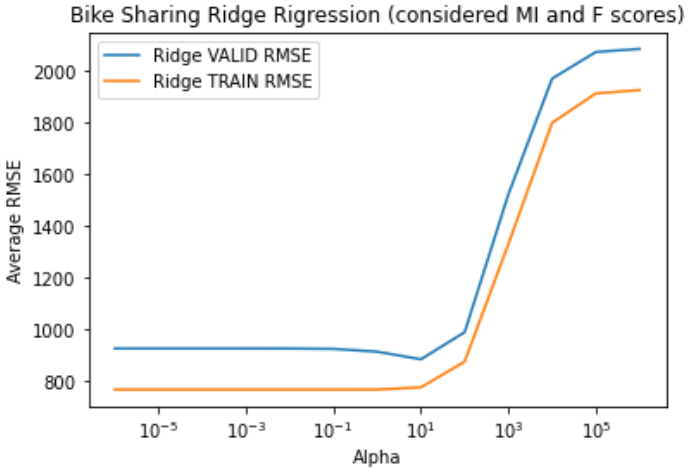
```
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5878386.
899279833, tolerance: 259618.5997721885
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 8265008.
043797553, tolerance: 230120.4946808511
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1413771
6.566514552, tolerance: 215148.17779042554
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5464047.
78382349, tolerance: 248149.695762462
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5666994.
191078067, tolerance: 199790.9642304414
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1494925
1.430134892, tolerance: 266874.350268845
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1457807
1.683276415, tolerance: 252850.1612019757
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5878602.
499028921, tolerance: 259618.5997721885
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1342002
0.973685622, tolerance: 230120.4946808511
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1413797
6.066423893, tolerance: 215148.17779042554
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5464266.
752822936, tolerance: 248149.695762462
  model = cd_fast.enet_coordinate_descent(
```

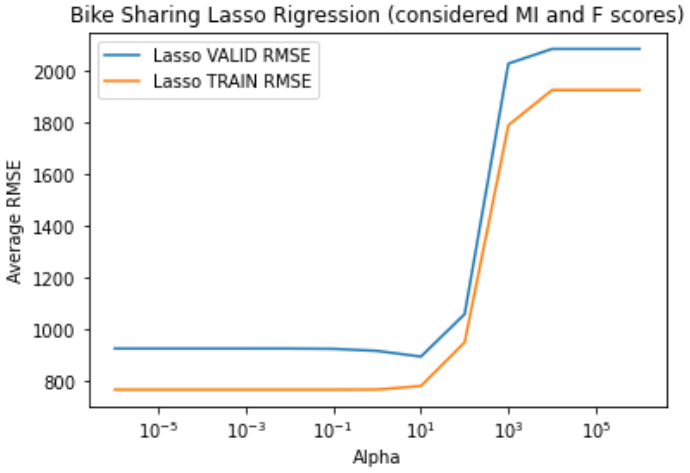
Bike Sharing Ridge Rigression (considered MI and F scores)

Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.000001	768.538669	927.326826
0.000010	768.538669	927.326684
0.000100	768.538669	927.325272
0.001000	768.538669	927.311151
0.010000	768.538703	927.170357
0.100000	768.542026	925.802692
1.000000	768.797490	915.066580
10.000000	777.294616	885.549747
100.000000	876.362275	990.153714
1000.000000	1331.296623	1522.979277
10000.000000	1799.550737	1970.179438
100000.000000	1912.901202	2073.149906
1000000.000000	1926.011959	2084.981847

Bike Sharing Lasso Rigression (considered MI and F scores)

Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.000001	768.538669	927.326834
0.000010	768.538669	927.326766
0.000100	768.538669	927.326235
0.001000	768.538669	927.318901
0.010000	768.538733	927.225098
0.100000	768.543595	926.093562
1.000000	768.882529	918.118447
10.000000	782.255640	896.119430
100.000000	951.128394	1060.376043
1000.000000	1791.133860	2029.505532
10000.000000	1927.491983	2086.316569
100000.000000	1927.491983	2086.316569
1000000.000000	1927.491983	2086.316569





In [14…

```
#SUI dataset
alphas = np.logspace(-6, 6, 13)
Reg_CV_rmse(datasets[1],dfs[1], ys[1], drop_featmf[1], alphas, title = "(considered MI and F scores)")
```

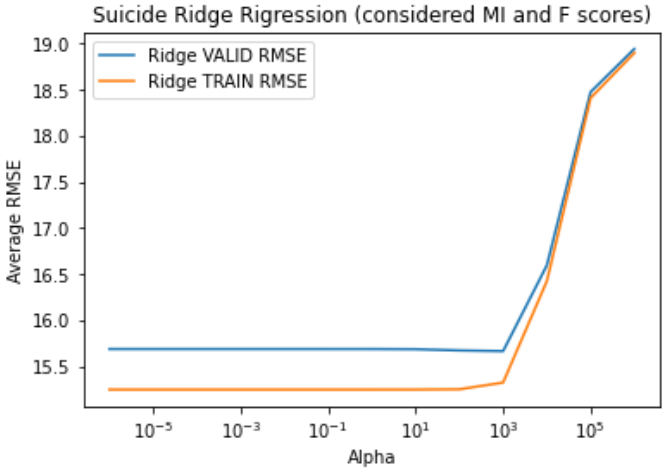
```
Suicide Least Squares (considered MI and F scores) Avg Train RMSE 15.247
Suicide Least Squares (considered MI and F scores) Avg Test RMSE 15.687
```

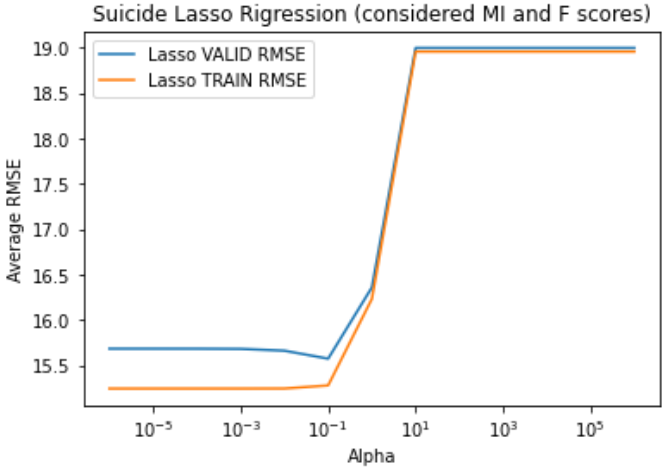
```
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 41263.52
7881586924, tolerance: 902.9057210731917
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 582450.0
231578457, tolerance: 866.7469129119216
  model = cd_fast.enet_coordinate_descent(
Suicide Ridge Rigression (considered MI and F scores)
```

Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.000001	15.247070	15.686851
0.000010	15.247070	15.686851
0.000100	15.247070	15.686851
0.001000	15.247070	15.686850
0.010000	15.247070	15.686848
0.100000	15.247070	15.686828
1.000000	15.247071	15.686629
10.000000	15.247124	15.684726
100.000000	15.250336	15.671479
1000.000000	15.323062	15.664045
10000.000000	16.422457	16.593604
100000.000000	18.407356	18.473529
1000000.000000	18.896928	18.940543

Suicide Lasso Rigression (considered MI and F scores)

Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.000001	15.247070	15.686848
0.000010	15.247070	15.686826
0.000100	15.247070	15.686607
0.001000	15.247080	15.684504
0.010000	15.247734	15.664367
0.100000	15.281677	15.576465
1.000000	16.235252	16.361834
10.000000	18.959137	18.999758
100.000000	18.959137	18.999758
1000.000000	18.959137	18.999758
10000.000000	18.959137	18.999758
100000.000000	18.959137	18.999758
1000000.000000	18.959137	18.999758





In [14…

```
#VID dataset
alphas = np.logspace(-6, 6, 13)
Reg_CV_rmse(datasets[2],dfs[2], ys[2], drop_featmf[2], alphas, title = "(considered MI and F scores)")
```

```
Video Transcoding Least Squares (considered MI and F scores) Avg Train RMSE 11.401
Video Transcoding Least Squares (considered MI and F scores) Avg Test RMSE 5193532805.567
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3711203.
398203249, tolerance: 1591.512660257359
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3134066.
6020303946, tolerance: 1603.301628732056
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3681942.
413496883, tolerance: 1604.4482794073392
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3532341.
222939875, tolerance: 1586.9882526084136
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3800284.
27967036, tolerance: 1601.1905426026985
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3568890.
637001399, tolerance: 1562.2192674560872
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3868785.
673490963, tolerance: 1634.9649856439673
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
```

```
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3680.157
062070444, tolerance: 1606.2653608586756
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3868495.
322104753, tolerance: 1676.73797704552
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 15749.11
2767972052, tolerance: 1592.4802455801141
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1095506.
2692707893, tolerance: 1591.512660257359
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 333365.9
278662009, tolerance: 1604.4482794073392
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1732897.
378295518, tolerance: 1601.1905426026985
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 332213.0
0509078987, tolerance: 1562.2192674560872
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1605307.
8708834993, tolerance: 1634.9649856439673
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 863029.6
842512321, tolerance: 1676.73797704552
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 15774.34
6628539264, tolerance: 1592.4802455801141
  model = cd_fast.enet_coordinate_descent(
```

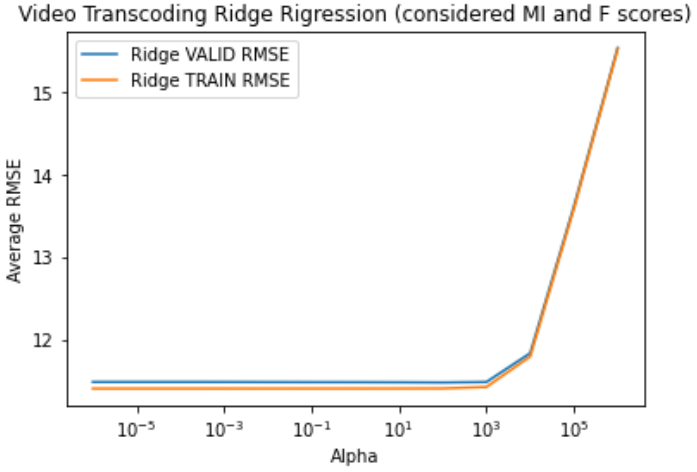
Video Transcoding Ridge Rigression (considered MI and F scores)

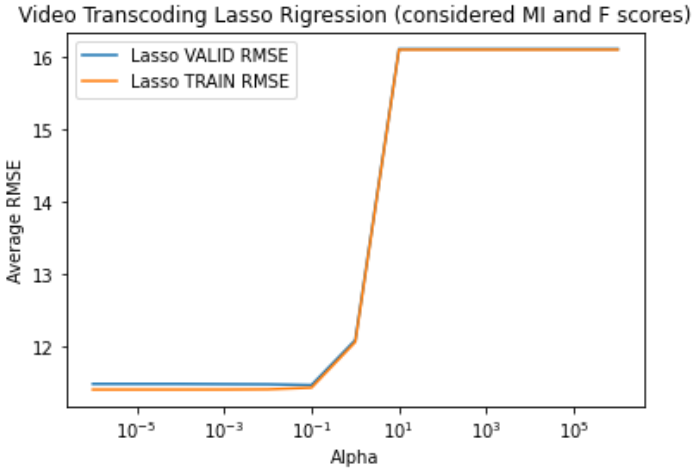
Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.000001	11.401138	11.481688
0.000010	11.401138	11.481685
0.000100	11.401138	11.481659
0.001000	11.401138	11.481422
0.010000	11.401143	11.480359
0.100000	11.401168	11.479431
1.000000	11.401187	11.478905
10.000000	11.401352	11.477571
100.000000	11.402624	11.475173
1000.000000	11.420896	11.481379
10000.000000	11.790868	11.831773
100000.000000	13.591885	13.617214
1000000.000000	15.529429	15.544868

Video Transcoding Lasso Rigression (considered MI and F scores)

Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.000001	11.401176	11.479348

Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.000010	11.401179	11.479294
0.000100	11.401188	11.479153
0.001000	11.401610	11.476482
0.010000	11.404468	11.474262
0.100000	11.427839	11.463634
1.000000	12.059405	12.086116
10.000000	16.106800	16.116994
100.000000	16.106800	16.116994
1000.000000	16.106800	16.116994
10000.000000	16.106800	16.116994
100000.000000	16.106800	16.116994
1000000.000000	16.106800	16.116994





The above results have to be treated with care, since they include categorical features which were scalar encoded for the purposes of feature selection. Let us remove the categorical features and observe the relationship w.r.t only the continuous features.

In [14…

```
#Computing Fscores for numerical features alone
results_cont = {}

for d, df, y, cnames in zip(datasets, dfs, ys, cat_names):
```

```
    print(d)

    X = df.drop(columns=[y]+cnames)
    y = df[y]

    # mutual information
    mi = mutual_info_regression(X, y)
    mi /= np.max(mi)

    #fregression
    f_test, pvalues = f_regression(X,y)
    f_test /= np.max(f_test)


    df_f = pd.DataFrame(list(zip(X.columns, mi, f_test, pvalues)), columns=['Continuous features', 'mi','f

    results_cont[d] = df_f

    display(df_f)
```

Bike Sharing

	Continuous features	mi	f_reg	P_values
0	temp	0.836662	0.981381	2.810622e-81
1	atemp	1.000000	1.000000	1.854504e-82
2	hum	0.099821	0.015467	6.454143e-03
3	windspeed	0.119986	0.087962	1.359959e-10

Suicide

	Continuous features	mi	f_reg	P_values
0	population	1.000000	0.107688	0.167021
1	gdp_for_year ($)	0.375296	1.000000	0.000026
2	gdp_per_capita ($)	0.327576	0.004999	0.765905

Video Transcoding

	Continuous features	mi	f_reg	P_values
0	duration	0.972518	0.000081	1.467846e-01
1	width	0.442772	0.045465	2.213683e-256
2	height	0.453664	0.044485	5.955923e-251
3	bitrate	0.987996	0.065419	0.000000e+00
4	framerate	0.556470	0.016789	1.889575e-96
5	i	0.906506	0.000906	1.238180e-06
6	p	1.000000	0.002925	3.047665e-18
7	b	0.009551	0.000070	1.776859e-01
8	frames	0.995296	0.002910	3.718494e-18
9	i_size	0.975284	0.011146	9.874803e-65
10	p_size	0.985954	0.025514	2.541518e-145
11	size	0.984795	0.025226	1.043438e-143
12	o_bitrate	0.059304	0.065660	0.000000e+00
13	o_framerate	0.040801	0.029005	8.053878e-165
14	o_width	0.944968	1.000000	0.000000e+00
15	o_height	0.951485	0.980497	0.000000e+00

Let us only drop numeric columns which do not seem to contribute much. Criteria: drop a feature if it has both Fscore and MI less than 0.1

In [15…

```
drop_bik2mf = ['hum']
drop_sui2mf = []
#drop_vid2 = ['b', 'o_bitrate', 'o_framerate']

drop_feat2mf = [drop_bik2mf, drop_sui2mf]
```

In [15…

```
#BIKE dataset (dropping only numerical features)
alphas = np.logspace(-6, 6, 13)
Reg_CV_rmse(datasets[0],dfs[0], ys[0], drop_feat2mf[0], alphas, title = "(considered MI and F scores of nu
```

```
Bike Sharing Least Squares (considered MI and F scores of numerical features) Avg Train RMSE 761.819
Bike Sharing Least Squares (considered MI and F scores of numerical features) Avg Test RMSE 928.172
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5549815.
650009513, tolerance: 199790.9642304414
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1475903
2.130912244, tolerance: 255245.10859148938
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1428053
0.876301706, tolerance: 271504.2992972644
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1436021
7.852361262, tolerance: 266874.350268845
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1435363
2.599874258, tolerance: 252850.1612019757
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 8582475.
2042948, tolerance: 259618.5997721885
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1352108
8.064303368, tolerance: 245012.33036170216
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1325707
4.201948196, tolerance: 230120.4946808511
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1408141
1.068472683, tolerance: 215148.17779042554
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5425736.
417455733, tolerance: 248149.695762462
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5549842.
539135516, tolerance: 199790.9642304414
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1475906
5.203077853, tolerance: 255245.10859148938
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1428056
4.313158214, tolerance: 271504.2992972644
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1436025
3.925693393, tolerance: 266874.350268845
  model = cd_fast.enet_coordinate_descent(
```

```
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1435366
6.568000078, tolerance: 252850.1612019757
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5807326.
927026033, tolerance: 259618.5997721885
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1352112
0.51316315, tolerance: 245012.33036170216
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1325711
2.275432497, tolerance: 230120.4946808511
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1408144
2.3953318, tolerance: 215148.17779042554
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5425763.
45553416, tolerance: 248149.695762462
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5550093.
878501177, tolerance: 199790.9642304414
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1475937
6.161370456, tolerance: 255245.10859148938
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1428086
2.418976426, tolerance: 271504.2992972644
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1436060
0.480954826, tolerance: 266874.350268845
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1435399
2.835641503, tolerance: 252850.1612019757
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5807577.
069760323, tolerance: 259618.5997721885
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1325747
4.678876251, tolerance: 230120.4946808511
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1411048.
8701751232, tolerance: 215148.17779042554
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5426021.
773596704, tolerance: 248149.695762462
  model = cd_fast.enet_coordinate_descent(
Bike Sharing Ridge Rigression (considered MI and F scores of numerical features)
```

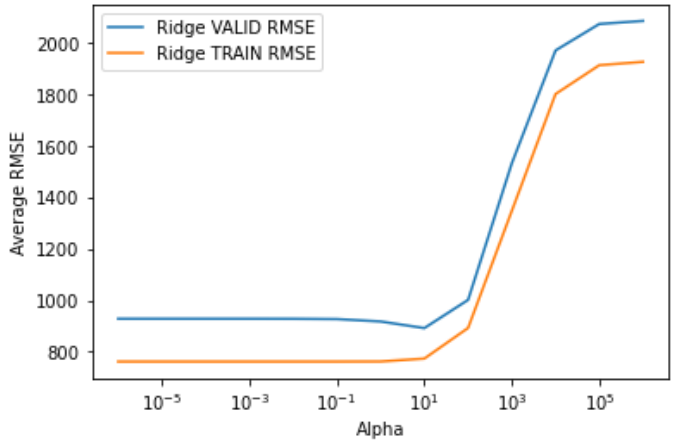
Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.000001	761.819063	928.171603
0.000010	761.819063	928.171484
0.000100	761.819063	928.170292
0.001000	761.819063	928.158381
0.010000	761.819105	928.039560
0.100000	761.823168	926.879972

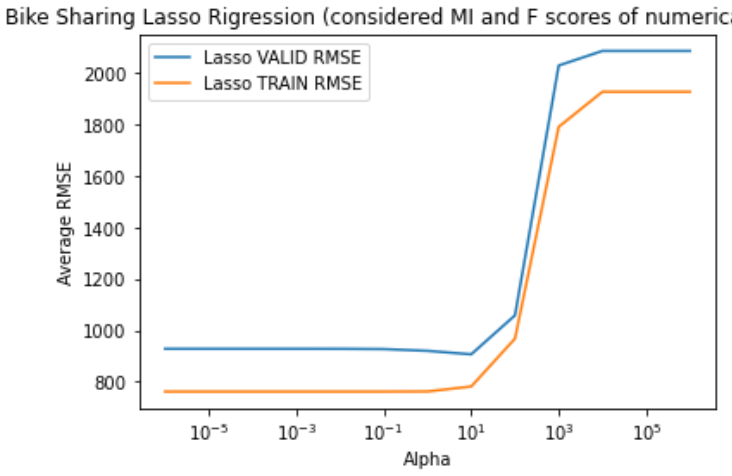
Alpha	Ridge Train RMSE	Ridge Valid RMSE
1.000000	762.144492	917.539138
10.000000	773.448215	891.846448
100.000000	892.374408	1001.318995
1000.000000	1347.123881	1531.893523
10000.000000	1800.954195	1970.283525
100000.000000	1913.012915	2073.115256
1000000.000000	1926.022777	2084.977833

Bike Sharing Lasso Rigression (considered MI and F scores of numerical features)

Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.000001	761.819063	928.171609
0.000010	761.819063	928.171546
0.000100	761.819063	928.170906
0.001000	761.819063	928.162908
0.010000	761.819134	928.084505
0.100000	761.825957	927.062447
1.000000	762.259632	920.149507
10.000000	781.560947	906.748109
100.000000	967.650054	1058.198619
1000.000000	1791.133860	2029.505532
10000.000000	1927.491983	2086.316569
100000.000000	1927.491983	2086.316569
1000000.000000	1927.491983	2086.316569

Bike Sharing Ridge Rigression (considered MI and F scores of numerical features)





In [15…

```
#SUI dataset (dropping only numerical features)
alphas = np.logspace(-6, 6, 13)
Reg_CV_rmse(datasets[1],dfs[1], ys[1], drop_feat2mf[1], alphas, title = "(considered MI and F scores of nu
```

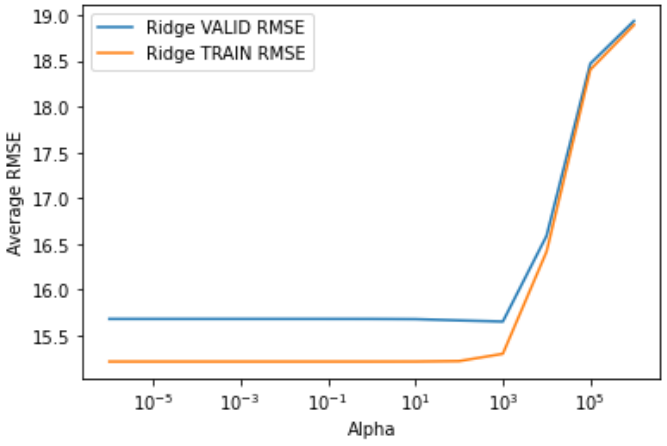
```
Suicide Least Squares (considered MI and F scores of numerical features) Avg Train RMSE 15.214
Suicide Least Squares (considered MI and F scores of numerical features) Avg Test RMSE 15.681
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 603506.1
211688416, tolerance: 928.0534336397959
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 41108.68
286894448, tolerance: 902.9057210731917
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 580407.6
820527837, tolerance: 866.7469129119216
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 38218.35
147104133, tolerance: 861.6330013969156
  model = cd_fast.enet_coordinate_descent(
Suicide Ridge Rigression (considered MI and F scores of numerical features)
```

Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.000001	15.214334	15.680975
0.000010	15.214334	15.680975
0.000100	15.214334	15.680975
0.001000	15.214334	15.680975
0.010000	15.214334	15.680973
0.100000	15.214334	15.680949
1.000000	15.214337	15.680721
10.000000	15.214581	15.678696
100.000000	15.219545	15.664560
1000.000000	15.299454	15.652099
10000.000000	16.416726	16.589899
100000.000000	18.406502	18.472879
1000000.000000	18.896840	18.940475

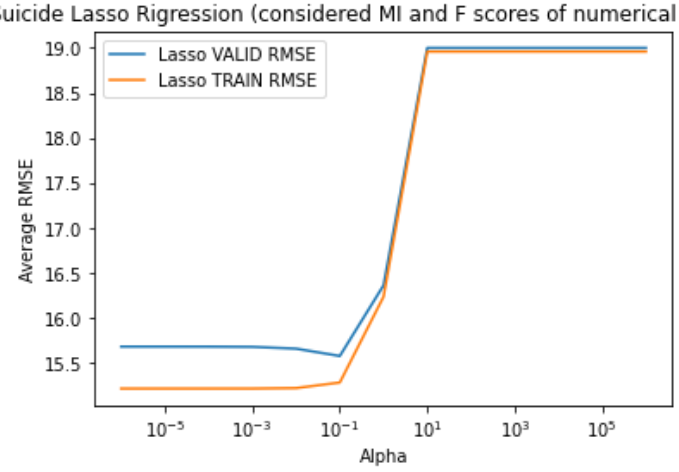
```
Suicide Lasso Rigression (considered MI and F scores of numerical features)
```

Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.000001	15.214334	15.680972
0.000010	15.214334	15.680944
0.000100	15.214335	15.680667
0.001000	15.214434	15.677997
0.010000	15.219571	15.657915
0.100000	15.281677	15.576465
1.000000	16.235252	16.361834
10.000000	18.959137	18.999758
100.000000	18.959137	18.999758
1000.000000	18.959137	18.999758
10000.000000	18.959137	18.999758
100000.000000	18.959137	18.999758
1000000.000000	18.959137	18.999758

Suicide Ridge Rigression (considered MI and F scores of numerical features)



Suicide Lasso Rigression (considered MI and F scores of numerical features)



Dropping features if Fscore < 0.1

For VID dataset, dropping features if pvalue is > 1e-6

In [15…

```
drop_bikf = ['holiday', 'weekday','workingday','hum','windspeed']
drop_suif = ['year', 'population','gdp_for_year ($)','gdp_per_capita ($)','generation']
```

```
drop_vidf = [
    'duration',
    'codec',
    'i',
    'b'
]

drop_featf = [drop_bikf, drop_suif, drop_vidf]
```

In [15…

```
#BIKE dataset (dropping only on basis of F scores)
alphas = np.logspace(-6, 6, 13)
Reg_CV_rmse(datasets[0],dfs[0], ys[0], drop_featf[0], alphas, title = "(dropping only on basis of F scores
```

```
Bike Sharing Least Squares (dropping only on basis of F scores) Avg Train RMSE 801.513
Bike Sharing Least Squares (dropping only on basis of F scores) Avg Test RMSE 966.878
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 6140986.
39256084, tolerance: 199790.9642304414
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1644173
9.858892322, tolerance: 255245.10859148938
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1615084
0.89150399, tolerance: 271504.2992972644
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1609953
3.203085482, tolerance: 266874.350268845
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1577877
2.704669416, tolerance: 252850.1612019757
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 6362351.
669636071, tolerance: 259618.5997721885
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1488937
5.564325273, tolerance: 230120.4946808511
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1529415
3.561452687, tolerance: 215148.17779042554
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5928045.
502282858, tolerance: 248149.695762462
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 6141011.
161694944, tolerance: 199790.9642304414
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1644177
0.788832963, tolerance: 255245.10859148938
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1615087
1.049491882, tolerance: 271504.2992972644
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1609956
6.324915707, tolerance: 266874.350268845
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1577880
4.61026895, tolerance: 252850.1612019757
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 6362377.
```

649617612, tolerance: 259618.5997721885
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model_coordinate_descent.py:530: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1488941
0.985424638, tolerance: 230120.4946808511
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model_coordinate_descent.py:530: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1529418
6.305452049, tolerance: 215148.17779042554
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model_coordinate_descent.py:530: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5928071.
344408095, tolerance: 248149.695762462
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model_coordinate_descent.py:530: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations. Duality gap: 6141247.
402817965, tolerance: 199790.9642304414
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model_coordinate_descent.py:530: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1644206
5.953319788, tolerance: 255245.10859148938
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model_coordinate_descent.py:530: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1609988
4.98759216, tolerance: 266874.350268845
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model_coordinate_descent.py:530: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations. Duality gap: 6362624.
732319355, tolerance: 259618.5997721885
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model_coordinate_descent.py:530: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1488974
8.612031877, tolerance: 230120.4946808511
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model_coordinate_descent.py:530: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1529449
5.055318534, tolerance: 215148.17779042554
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model_coordinate_descent.py:530: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5928315.
783423364, tolerance: 248149.695762462
  model = cd_fast.enet_coordinate_descent(
Bike Sharing Ridge Rigression (dropping only on basis of F scores)

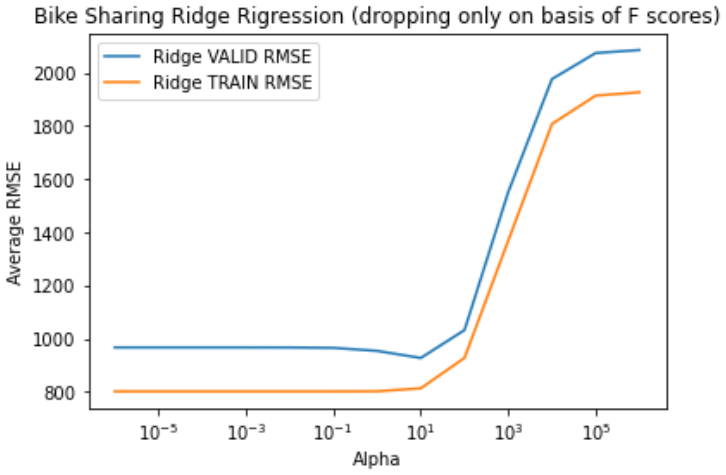
Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.000001	801.512770	966.878428
0.000010	801.512770	966.878266
0.000100	801.512770	966.876641
0.001000	801.512770	966.860400
0.010000	801.512822	966.698670
0.100000	801.517827	965.145954
1.000000	801.879478	953.866578
10.000000	813.159740	927.168662
100.000000	927.530778	1032.085098
1000.000000	1368.353809	1551.045025
10000.000000	1806.906604	1975.868526
100000.000000	1913.732419	2073.793313
1000000.000000	1926.096233	2085.047084

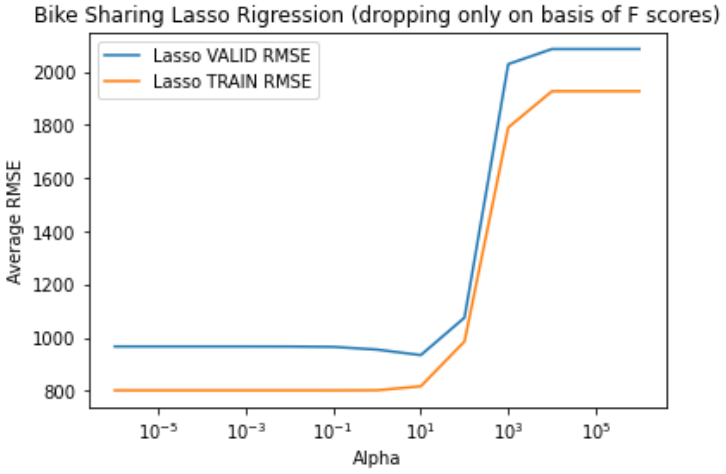
Bike Sharing Lasso Rigression (dropping only on basis of F scores)

	Lasso Train RMSE	Lasso Valid RMSE

Alpha	Lasso Train RMSE	Lasso Valid RMSE

Alpha		
0.000001	801.512770	966.878434
0.000010	801.512770	966.878326
0.000100	801.512770	966.877209
0.001000	801.512770	966.864835
0.010000	801.512843	966.742209
0.100000	801.519166	965.352169
1.000000	802.005700	955.054877
10.000000	816.977436	934.330780
100.000000	985.685709	1075.448264
1000.000000	1791.133860	2029.505532
10000.000000	1927.491983	2086.316569
100000.000000	1927.491983	2086.316569
1000000.000000	1927.491983	2086.316569





In [15…

```
#SUI dataset (dropping only on basis of F scores)
alphas = np.logspace(-6, 6, 13)
Reg_CV_rmse(datasets[1],dfs[1], ys[1], drop_featf[1], alphas, title = "(dropping only on basis of F scores
```

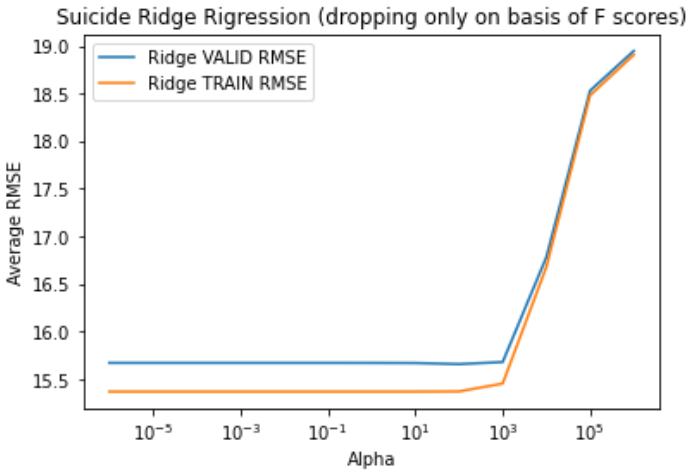
```
Suicide Least Squares (dropping only on basis of F scores) Avg Train RMSE 15.391
```

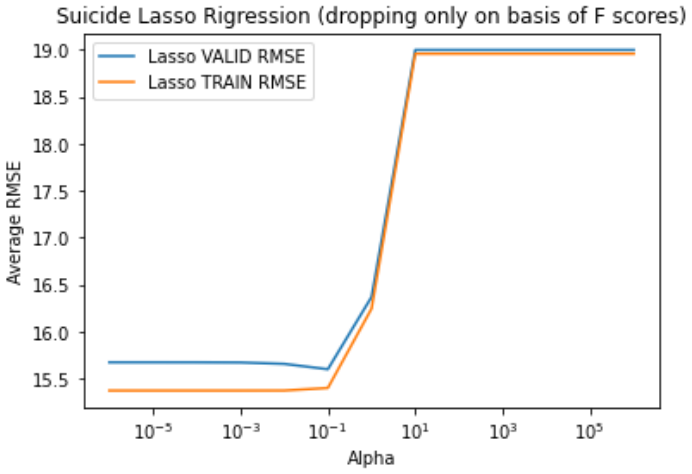
```
Suicide Least Squares (dropping only on basis of F scores) Avg Test RMSE 15.661
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations. Duality gap: 589517.544945607, tolerance: 866.7469129119216
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations. Duality gap: 38985.161411485635, tolerance: 861.6330013969156
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarning: Objective did not converge. You might want to increase the number of iterations. Duality gap: 611437.5506174704, tolerance: 962.8427222214124
  model = cd_fast.enet_coordinate_descent(
Suicide Ridge Rigression (dropping only on basis of F scores)
```

Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.000001	15.371207	15.672340
0.000010	15.371207	15.672340
0.000100	15.371207	15.672340
0.001000	15.371207	15.672340
0.010000	15.371207	15.672339
0.100000	15.371207	15.672325
1.000000	15.371207	15.672182
10.000000	15.371221	15.670790
100.000000	15.372530	15.659862
1000.000000	15.455062	15.680952
10000.000000	16.678948	16.783074
100000.000000	18.480526	18.529835
1000000.000000	18.905374	18.946899

Suicide Lasso Rigression (dropping only on basis of F scores)

Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.000001	15.371207	15.672339
0.000010	15.371207	15.672323
0.000100	15.371207	15.672166
0.001000	15.371211	15.670601
0.010000	15.371609	15.655665
0.100000	15.399047	15.598492
1.000000	16.244231	16.368981
10.000000	18.959137	18.999758
100.000000	18.959137	18.999758
1000.000000	18.959137	18.999758
10000.000000	18.959137	18.999758
100000.000000	18.959137	18.999758
1000000.000000	18.959137	18.999758





In [15…

```
#VID dataset (dropping only on basis of F scores)
alphas = np.logspace(-6, 6, 13)
Reg_CV_rmse(datasets[2],dfs[2], ys[2], drop_featf[2], alphas, title = "(dropping only on basis of F scores
```

```
Video Transcoding Least Squares (dropping only on basis of F scores) Avg Train RMSE 11.008
Video Transcoding Least Squares (dropping only on basis of F scores) Avg Test RMSE 11.101
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3598364.
047907798, tolerance: 1591.512660257359
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3125572.
365122967, tolerance: 1603.301628732056
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 2957444.
8884534827, tolerance: 1604.4482794073392
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3444486.
9039269313, tolerance: 1586.9882526084136
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3499518.
0847310424, tolerance: 1601.1905426026985
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3500302.
224462295, tolerance: 1562.2192674560872
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3371354.
6213527382, tolerance: 1634.9649856439673
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
```

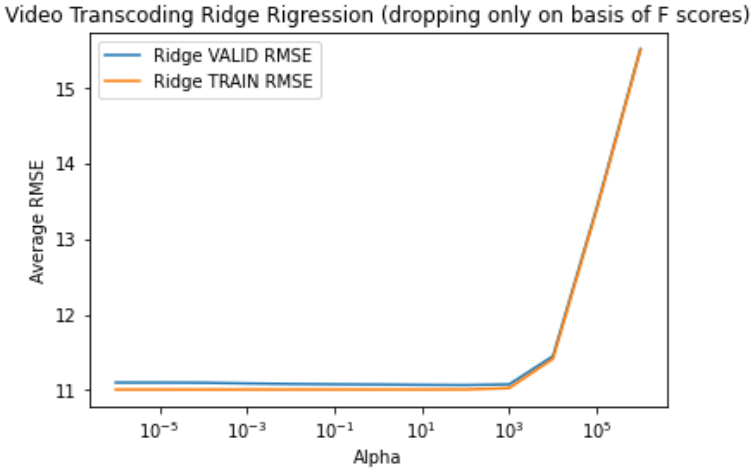
```
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3397468.
594383608, tolerance: 1606.2653608586756
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3808648.
2031367994, tolerance: 1676.73797704552
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3329741.
279571014, tolerance: 1592.4802455801141
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 2566158.
1192809576, tolerance: 1591.512660257359
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 731602.0
858222293, tolerance: 1586.9882526084136
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1206875.
6085407273, tolerance: 1601.1905426026985
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 2015792.
79403474, tolerance: 1562.2192674560872
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 80084.53
500898927, tolerance: 1606.2653608586756
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3259547.
301192263, tolerance: 1676.73797704552
  model = cd_fast.enet_coordinate_descent(
Video Transcoding Ridge Rigression (dropping only on basis of F scores)
```

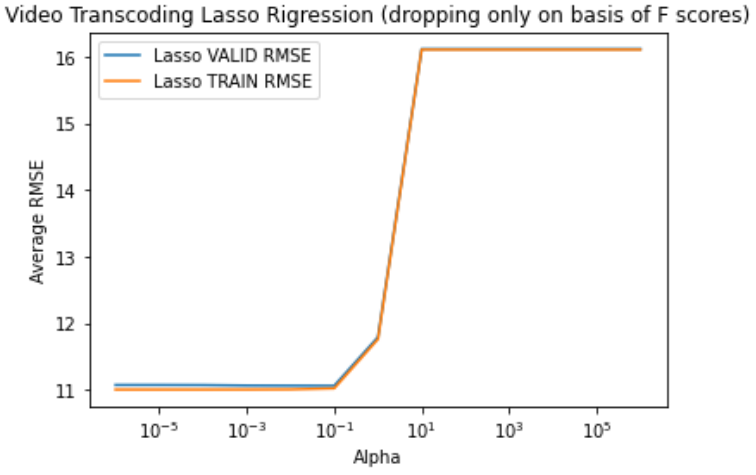
Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.000001	11.008188	11.100982
0.000010	11.008188	11.100750
0.000100	11.008188	11.098686
0.001000	11.008198	11.089620
0.010000	11.008238	11.082505
0.100000	11.008337	11.078306
1.000000	11.008401	11.075602
10.000000	11.008610	11.071089
100.000000	11.009903	11.068000
1000.000000	11.027936	11.077188
10000.000000	11.414257	11.450014
100000.000000	13.386633	13.410761
1000000.000000	15.497384	15.512488

```
Video Transcoding Lasso Rigression (dropping only on basis of F scores)
```

Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.000001	11.008375	11.077660
0.000010	11.008380	11.077473
0.000100	11.008405	11.076206

Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.001000	11.008945	11.069033
0.010000	11.011135	11.067147
0.100000	11.026481	11.064975
1.000000	11.763790	11.794118
10.000000	16.106800	16.116994
100.000000	16.106800	16.116994
1000.000000	16.106800	16.116994
10000.000000	16.106800	16.116994
100000.000000	16.106800	16.116994
1000000.000000	16.106800	16.116994





Dropping features on basis of MI scores alone.

Drop features is MI < 0.1

In [15…

```
drop_bikm = ['holiday', 'weekday','workingday','hum']
drop_suim = ['year']
drop_vidm = [
    'b',
    'o_bitrate',
    'o_framerate'
]
```

```
drop_featm = [drop_bikm, drop_suim, drop_vidm]
```

In [15…

```
#BIKE dataset (dropping only on basis of MI scores)
alphas = np.logspace(-6, 6, 13)
Reg_CV_rmse(datasets[0],dfs[0], ys[0], drop_featm[0], alphas, title = "(dropping only on basis of MI score
```

```
Bike Sharing Least Squares (dropping only on basis of MI scores) Avg Train RMSE 785.720
Bike Sharing Least Squares (dropping only on basis of MI scores) Avg Test RMSE 941.971
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5905041.
252309978, tolerance: 199790.9642304414
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1575105
7.26000452, tolerance: 255245.10859148938
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1542509
5.317616582, tolerance: 271504.2992972644
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1545068
8.48313123, tolerance: 266874.350268845
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1513571
8.151868582, tolerance: 252850.1612019757
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 6126659.
293267131, tolerance: 259618.5997721885
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3567021
7.098873764, tolerance: 245012.33036170216
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1422628
1.465982735, tolerance: 230120.4946808511
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1476156
2.195263207, tolerance: 215148.17779042554
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5703574.
9524437785, tolerance: 248149.695762462
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5905066.
218712568, tolerance: 199790.9642304414
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1575108
7.959168255, tolerance: 255245.10859148938
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1542512
4.731168509, tolerance: 271504.2992972644
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1545072
1.542671502, tolerance: 266874.350268845
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1513575
0.11130941, tolerance: 252850.1612019757
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 6126685.
134329498, tolerance: 259618.5997721885
  model = cd_fast.enet_coordinate_descent(
```

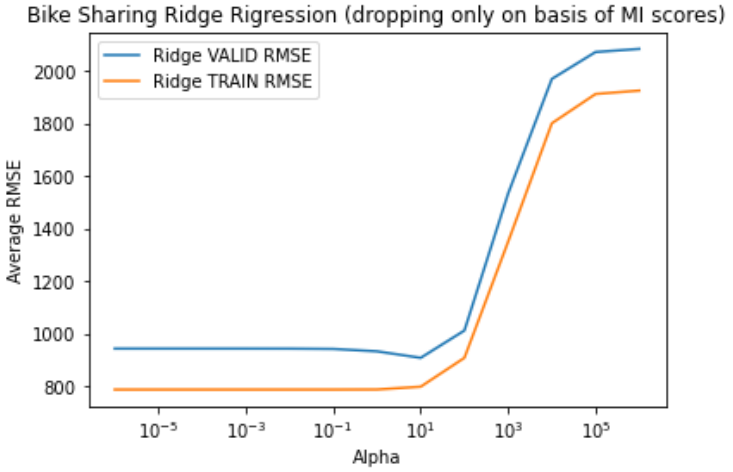
```
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3567025
7.38410357, tolerance: 245012.33036170216
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1422631
6.378231943, tolerance: 230120.4946808511
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5703600.
093001902, tolerance: 248149.695762462
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5905300.
527900338, tolerance: 199790.9642304414
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1575137
9.078811884, tolerance: 255245.10859148938
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1542540
4.554082334, tolerance: 271504.2992972644
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1513605
4.183032334, tolerance: 252850.1612019757
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 6126927.
308772385, tolerance: 259618.5997721885
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1422665
0.445593119, tolerance: 230120.4946808511
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 5703836.
042867482, tolerance: 248149.695762462
  model = cd_fast.enet_coordinate_descent(
Bike Sharing Ridge Rigression (dropping only on basis of MI scores)
```

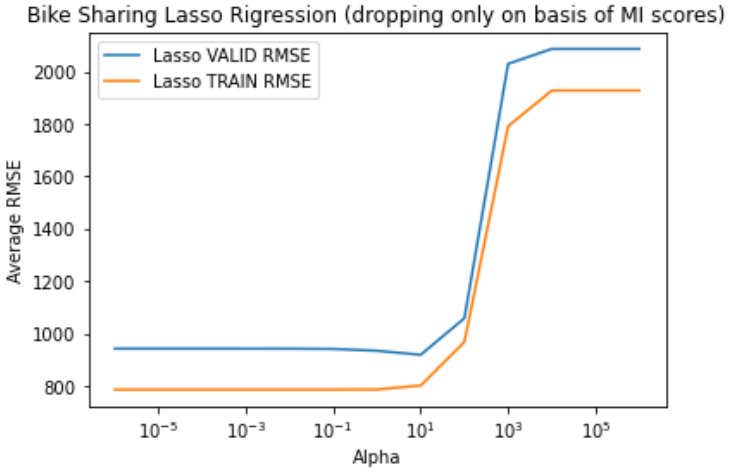
Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.000001	785.720253	941.970602
0.000010	785.720253	941.970485
0.000100	785.720253	941.969314
0.001000	785.720253	941.957612
0.010000	785.720294	941.840896
0.100000	785.724260	940.703228
1.000000	786.036387	931.605106
10.000000	796.702738	906.766714
100.000000	906.887643	1011.021831
1000.000000	1349.621617	1533.346014
10000.000000	1801.286247	1970.492499
100000.000000	1913.049337	2073.139015
1000000.000000	1926.026460	2084.980245

Bike Sharing Lasso Rigression (dropping only on basis of MI scores)

Alpha	Lasso Train RMSE	Lasso Valid RMSE

Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.000001	785.720253	941.970609
0.000010	785.720253	941.970551
0.000100	785.720253	941.969994
0.001000	785.720253	941.960945
0.010000	785.720326	941.873129
0.100000	785.726576	940.909806
1.000000	786.131003	933.924083
10.000000	801.124145	918.537063
100.000000	967.650054	1058.198619
1000.000000	1791.133860	2029.505532
10000.000000	1927.491983	2086.316569
100000.000000	1927.491983	2086.316569
1000000.000000	1927.491983	2086.316569





In [15…

```
#SUI dataset (dropping only on basis of MI scores)
alphas = np.logspace(-6, 6, 13)
Reg_CV_rmse(datasets[1],dfs[1], ys[1], drop_featm[1], alphas, title = "(dropping only on basis of MI score
```

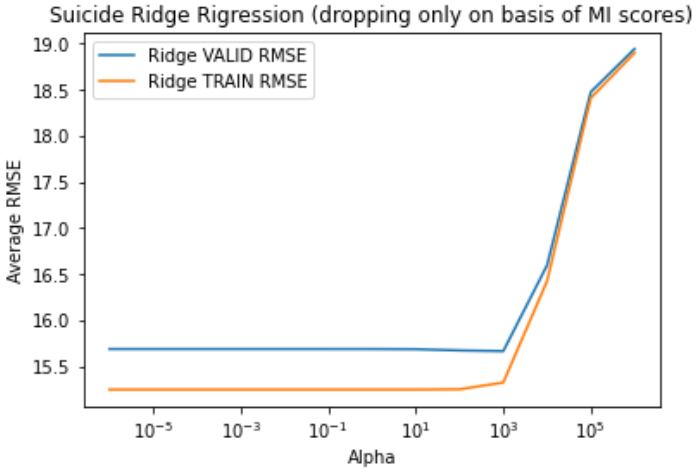
```
Suicide Least Squares (dropping only on basis of MI scores) Avg Train RMSE 15.247
Suicide Least Squares (dropping only on basis of MI scores) Avg Test RMSE 15.687
```

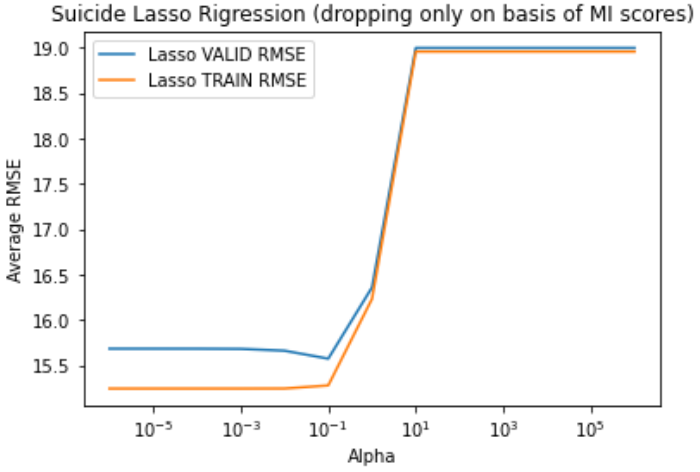
```
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 41263.52
7881586924, tolerance: 902.9057210731917
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 582450.0
231578457, tolerance: 866.7469129119216
  model = cd_fast.enet_coordinate_descent(
Suicide Ridge Rigression (dropping only on basis of MI scores)
```

Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.000001	15.247070	15.686851
0.000010	15.247070	15.686851
0.000100	15.247070	15.686851
0.001000	15.247070	15.686850
0.010000	15.247070	15.686848
0.100000	15.247070	15.686828
1.000000	15.247071	15.686629
10.000000	15.247124	15.684726
100.000000	15.250336	15.671479
1000.000000	15.323062	15.664045
10000.000000	16.422457	16.593604
100000.000000	18.407356	18.473529
1000000.000000	18.896928	18.940543

```
Suicide Lasso Rigression (dropping only on basis of MI scores)
```

Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.000001	15.247070	15.686848
0.000010	15.247070	15.686826
0.000100	15.247070	15.686607
0.001000	15.247080	15.684504
0.010000	15.247734	15.664367
0.100000	15.281677	15.576465
1.000000	16.235252	16.361834
10.000000	18.959137	18.999758
100.000000	18.959137	18.999758
1000.000000	18.959137	18.999758
10000.000000	18.959137	18.999758
100000.000000	18.959137	18.999758
1000000.000000	18.959137	18.999758





In [16…

```
#VID dataset (dropping only on basis of MI scores)
alphas = np.logspace(-6, 6, 13)
Reg_CV_rmse(datasets[2],dfs[2], ys[2], drop_featm[2], alphas, title = "(dropping only on basis of MI score
```

```
Video Transcoding Least Squares (dropping only on basis of MI scores) Avg Train RMSE 11.401
Video Transcoding Least Squares (dropping only on basis of MI scores) Avg Test RMSE 5193532805.567
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3711203.
398203249, tolerance: 1591.512660257359
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3134066.
6020303946, tolerance: 1603.301628732056
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3681942.
413496883, tolerance: 1604.4482794073392
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3532341.
222939875, tolerance: 1586.9882526084136
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3800284.
27967036, tolerance: 1601.1905426026985
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3568890.
637001399, tolerance: 1562.2192674560872
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3868785.
673490963, tolerance: 1634.9649856439673
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
```

```
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3680.157
062070444, tolerance: 1606.2653608586756
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 3868495.
322104753, tolerance: 1676.73797704552
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 15749.11
2767972052, tolerance: 1592.4802455801141
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1095506.
2692707893, tolerance: 1591.512660257359
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 333365.9
278662009, tolerance: 1604.4482794073392
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1732897.
378295518, tolerance: 1601.1905426026985
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 332213.0
0509078987, tolerance: 1562.2192674560872
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 1605307.
8708834993, tolerance: 1634.9649856439673
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 863029.6
842512321, tolerance: 1676.73797704552
  model = cd_fast.enet_coordinate_descent(
C:\Users\sriva\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:530: ConvergenceWarn
ing: Objective did not converge. You might want to increase the number of iterations. Duality gap: 15774.34
6628539264, tolerance: 1592.4802455801141
  model = cd_fast.enet_coordinate_descent(
Video Transcoding Ridge Rigression (dropping only on basis of MI scores)
```

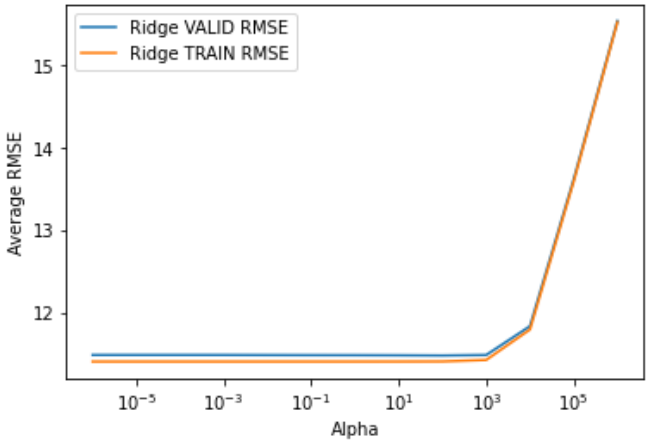
Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.000001	11.401138	11.481688
0.000010	11.401138	11.481685
0.000100	11.401138	11.481659
0.001000	11.401138	11.481422
0.010000	11.401143	11.480359
0.100000	11.401168	11.479431
1.000000	11.401187	11.478905
10.000000	11.401352	11.477571
100.000000	11.402624	11.475173
1000.000000	11.420896	11.481379
10000.000000	11.790868	11.831773
100000.000000	13.591885	13.617214
1000000.000000	15.529429	15.544868

Video Transcoding Lasso Rigression (dropping only on basis of MI scores)

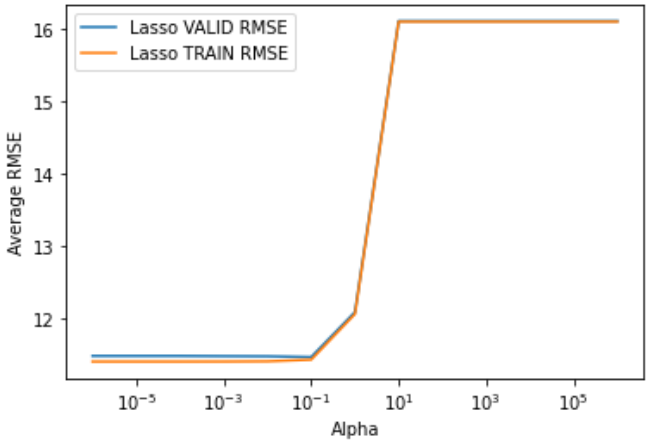
Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.000001	11.401176	11.479348

Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.000010	11.401179	11.479294
0.000100	11.401188	11.479153
0.001000	11.401610	11.476482
0.010000	11.404468	11.474262
0.100000	11.427839	11.463634
1.000000	12.059405	12.086116
10.000000	16.106800	16.116994
100.000000	16.106800	16.116994
1000.000000	16.106800	16.116994
10000.000000	16.106800	16.116994
100000.000000	16.106800	16.116994
1000000.000000	16.106800	16.116994

Video Transcoding Ridge Rigression (dropping only on basis of MI scores)



Video Transcoding Lasso Rigression (dropping only on basis of MI scores)



Trying recursive feature elimination

In [16…

```
aa = np.array([True, False,True,True,False, True,False,False,False,False,False,False])
print(dfs[0].columns)
print(dfs[0].columns[aa])
```

```
Index(['season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday',
       'weathersit', 'temp', 'atemp', 'hum', 'windspeed', 'cnt'],
```

```
      dtype='object')
Index(['season', 'mnth', 'holiday', 'workingday'], dtype='object')
```

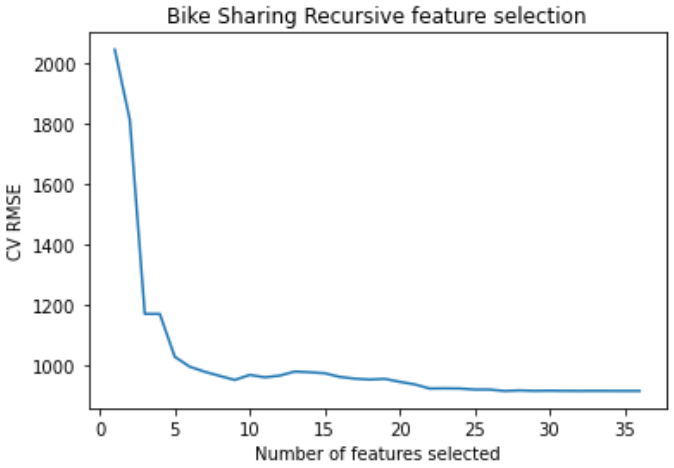
In [16…

```
def Recursive_Linreg(d, df, Y):
    lin_reg = LinearRegression()
    min_features_to_select = 1
    rfecv = RFECV(estimator=lin_reg, step=1, cv=KFold(10),
                  scoring='neg_mean_squared_error',
                  min_features_to_select=min_features_to_select)
    X = df.drop(columns = Y)
    X = pd.get_dummies(X)
    y = df[Y]
    rfecv.fit(X, y)
    cv_rmse_scores = np.sqrt(-rfecv.grid_scores_)

    print("Optimal number of features : %d" % rfecv.n_features_)
    print("CV rmse: {:.3f}".format(np.nanmin(cv_rmse_scores)))
    print("Optimal features selected:")
    print(X.columns[rfecv.ranking_==1])
    print("second best features:")
    print(X.columns[rfecv.ranking_==2])

    # Plot number of features VS. cross-validation scores
    plt.figure()
    plt.xlabel("Number of features selected")
    plt.ylabel("CV RMSE")
    plt.title(d+" Recursive feature selection")
    plt.plot(range(min_features_to_select,
                   len(rfecv.grid_scores_) + min_features_to_select),
             cv_rmse_scores)
    plt.show()
```

In [16…

```
#BIKE
Recursive_Linreg(datasets[0],dfs[0], ys[0])
```

```
Optimal number of features : 27
CV rmse: 914.856
Optimal features selected:
Index(['temp', 'atemp', 'hum', 'windspeed', 'season_1', 'season_4', 'yr_0',
       'yr_1', 'mnth_1', 'mnth_2', 'mnth_5', 'mnth_6', 'mnth_7', 'mnth_9',
       'mnth_11', 'mnth_12', 'holiday_0', 'holiday_1', 'weekday_0',
       'weekday_1', 'weekday_2', 'weekday_6', 'workingday_0', 'workingday_1',
       'weathersit_1', 'weathersit_2', 'weathersit_3'],
      dtype='object')
second best features:
Index(['mnth_3'], dtype='object')
```

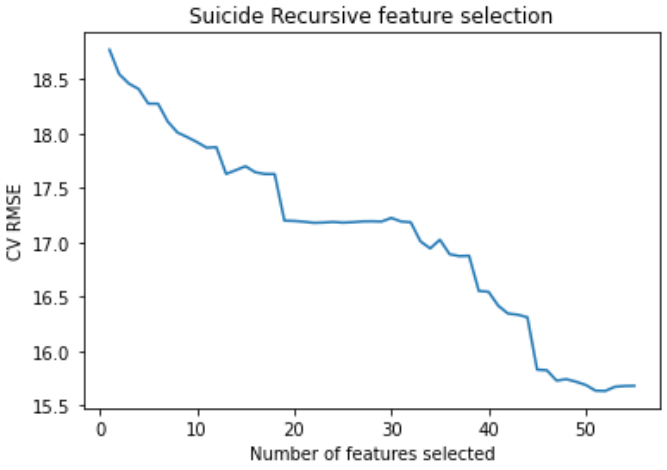


In [16…

```
#SUI
Recursive_Linreg(datasets[1],dfs[1], ys[1])
```

```
Optimal number of features : 52
```

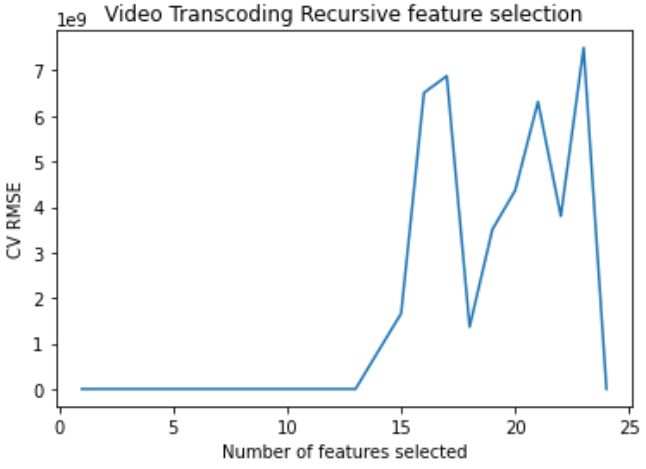
```
CV rmse: 15.633
Optimal features selected:
Index(['year_1985', 'year_1986', 'year_1987', 'year_1988', 'year_1989',
       'year_1990', 'year_1991', 'year_1992', 'year_1993', 'year_1994',
       'year_1995', 'year_1996', 'year_1997', 'year_1998', 'year_1999',
       'year_2000', 'year_2001', 'year_2002', 'year_2003', 'year_2004',
       'year_2005', 'year_2006', 'year_2007', 'year_2008', 'year_2009',
       'year_2010', 'year_2011', 'year_2012', 'year_2013', 'year_2014',
       'year_2015', 'year_2016', 'sex_female', 'sex_male', 'age_15-24 years',
       'age_25-34 years', 'age_35-54 years', 'age_5-14 years',
       'age_55-74 years', 'age_75+ years', 'generation_Boomers',
       'generation_G.I. Generation', 'generation_Generation X',
       'generation_Generation Z', 'generation_Millenials', 'generation_Silent',
       'continent_AF', 'continent_AS', 'continent_EU', 'continent_NA',
       'continent_OC', 'continent_SA'],
      dtype='object')
second best features:
Index(['gdp_per_capita ($)'], dtype='object')
```



In [16…

```
#VID
Recursive_Linreg(datasets[2],dfs[2], ys[2])
```

```
Optimal number of features : 24
CV rmse: 11.083
Optimal features selected:
Index(['duration', 'width', 'height', 'bitrate', 'framerate', 'i', 'p', 'b',
       'frames', 'i_size', 'p_size', 'size', 'o_bitrate', 'o_framerate',
       'o_width', 'o_height', 'codec_flv', 'codec_h264', 'codec_mpeg4',
       'codec_vp8', 'o_codec_flv', 'o_codec_h264', 'o_codec_mpeg4',
       'o_codec_vp8'],
      dtype='object')
second best features:
Index([], dtype='object')
```



Polynomial Regression

One hot vectorize the categorical features of the dataset

In [79]:
```
df_bik_oh = pd.get_dummies(df_bik)
df_sui_oh = pd.get_dummies(df_sui)
df_vid_oh = pd.get_dummies(df_vid)
```

In [80]:
```
dfs_oh = [df_bik_oh, df_sui_oh, df_vid_oh]
```

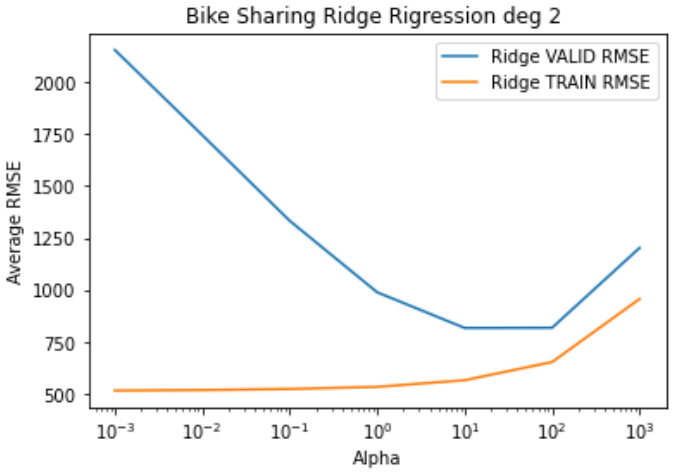
In [20…
```
def Poly_Reg_CV_rmse(d, df, Y, drp,alphas,deg,title = ""):
    X = df.drop(columns = [Y] + drp)
    y = df[Y]
    X = pd.get_dummies(X)

    poly = PolynomialFeatures(deg)
    X = poly.fit_transform(X)

    #lin_reg = LinearRegression()
    #scores = cross_validate(lin_reg, X, y, scoring="neg_mean_squared_error", cv=10, return_train_score=Tr
    #print(d+" Least Squares deg {} ".format(deg)+title+" Avg Train RMSE {:0.3f}".format(np.sqrt(np.mean(-
    #print(d+" Least Squares deg {} ".format(deg)+title+" Avg Test RMSE {:0.3f}".format(np.sqrt(np.mean(-s

    rid_scores={}
    las_scores={}

    rid_train_rmse = []
    rid_test_rmse = []
    las_train_rmse = []
    las_test_rmse = []

    for alpha in alphas:

        rid_reg = Ridge(alpha = alpha,max_iter = 1e5)
        scores = cross_validate(rid_reg, X, y, scoring="neg_mean_squared_error", cv=10, return_train_score
        rid_train_rmse.append(np.sqrt(np.mean(-scores['train_score'])))
        rid_test_rmse.append(np.sqrt(np.mean(-scores['test_score'])))

        las_reg = Lasso(alpha = alpha, max_iter = 1e5)
        scores = cross_validate(las_reg, X, y, scoring="neg_mean_squared_error", cv=10, return_train_score
        las_train_rmse.append(np.sqrt(np.mean(-scores['train_score'])))
        las_test_rmse.append(np.sqrt(np.mean(-scores['test_score'])))

    plt.plot(alphas, rid_test_rmse, label = 'Ridge VALID RMSE')
    plt.plot(alphas, rid_train_rmse, label = 'Ridge TRAIN RMSE')
    plt.ylabel('Average RMSE')
    plt.xlabel('Alpha')
    plt.xscale("log")
    plt.title(d+" Ridge Rigression deg {} ".format(deg)+title)
    plt.legend()
    plt.show()

    plt.plot(alphas, las_test_rmse, label = 'Lasso VALID RMSE')
    plt.plot(alphas, las_train_rmse, label = 'Lasso TRAIN RMSE')
    plt.ylabel('Average RMSE')
    plt.xlabel('Alpha')
    plt.xscale("log")
    plt.title(d+" Lasso Rigression deg {} ".format(deg)+title)
    plt.legend()
    plt.show()

    dframe = pd.DataFrame({'Ridge Train RMSE':rid_train_rmse, 'Ridge Valid RMSE':rid_test_rmse}, index = a
    dframe.index.name = 'Alpha'
    print(d+" Ridge Rigression deg {} ".format(deg)+title)
    display(dframe)

    dframe = pd.DataFrame({'Lasso Train RMSE':las_train_rmse, 'Lasso Valid RMSE':las_test_rmse}, index = a
    dframe.index.name = 'Alpha'
```

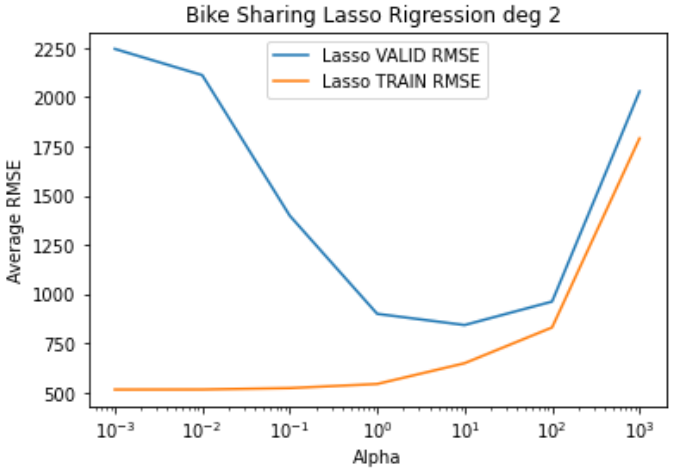
```
    print(d+" Lasso Rigression deg {} ".format(deg)+title)
    display(dframe)
```

In [ ]:

In [20…

```
#BIKE dataset degree = 2
deg = 2
alphas = np.logspace(-3, 3, 7)
Poly_Reg_CV_rmse(datasets[0],dfs[0], ys[0], drop_featmf[0], alphas, deg)
```





Bike Sharing Ridge Rigression deg 2

Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.001	515.665880	2155.546296
0.010	518.304905	1745.641630
0.100	523.418891	1332.969570
1.000	533.562158	988.780246
10.000	565.533582	817.118449
100.000	653.030491	817.952204
1000.000	956.661019	1202.634055

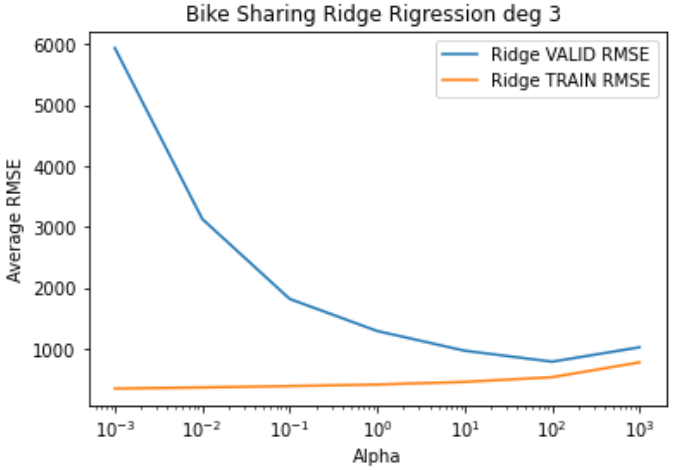
Bike Sharing Lasso Rigression deg 2

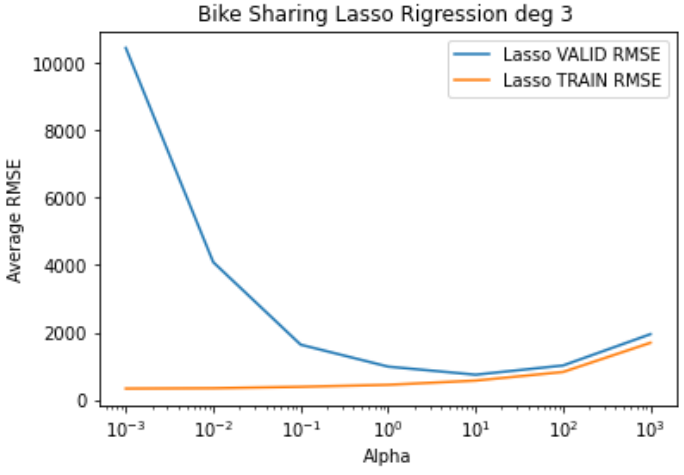
Alpha	Lasso Train RMSE	Lasso Valid RMSE

Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.001	515.264843	2245.189613
0.010	515.649650	2111.947010
0.100	522.739994	1398.493194
1.000	543.667733	899.912517
10.000	649.221121	843.033716
100.000	830.861938	962.315833
1000.000	1791.133860	2029.505532

In [20…

```
#BIKE dataset degree = 3
deg = 3
alphas = np.logspace(-3, 3, 7)
Poly_Reg_CV_rmse(datasets[0],dfs[0], ys[0], drop_featmf[0], alphas, deg)
```





Bike Sharing Ridge Rigression deg 3

Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.001	342.731147	5943.150076
0.010	361.170004	3131.310821
0.100	383.727208	1816.218095
1.000	409.351020	1289.168173

Alpha	Ridge Train RMSE	Ridge Valid RMSE
10.000	452.490100	965.629355
100.000	529.271463	784.794129
1000.000	773.974237	1021.565979

Bike Sharing Lasso Rigression deg 3

Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.001	337.258718	10442.636292
0.010	346.922525	4086.819409
0.100	390.142132	1639.195859
1.000	448.612421	988.685071
10.000	573.901469	748.794123
100.000	829.449240	1022.074151
1000.000	1696.076213	1951.616949

For the BIKE dataset, Ridge regression's best model occured at about alpha = 100 with degree 3 of polynomial features.

For Lasso, the best model was at degree 3 and alpha = 10

In [81]:
```
def Poly_feat_scores(d, df, y, deg):
    print(d)
    results = {}

    X = df.drop(columns=y)
    y = df[y]
    X = pd.get_dummies(X)

    cols = X.columns
    print("No of features", len(cols))
    feat_names = ['x'+str(i) for i in range(len(cols))]
    dfpoly = pd.DataFrame({'col_name':cols, 'col_rep':feat_names})
    display(dfpoly)

    poly = PolynomialFeatures(deg)
    X = poly.fit_transform(X)

    # mutual information
    mi = mutual_info_regression(X, y)
    mi /= np.max(mi)

    df_f = pd.DataFrame(list(zip(poly.get_feature_names(),mi)), columns=['feature','MI'])
    results[d+' MI'] = df_f.sort_values(by = ['MI'], ascending = False)
    display(results[d+' MI'])

    #fregression
    f_test, pvals= f_regression(X,y)
    f_test /= np.max(f_test)

    df_f = pd.DataFrame(list(zip(poly.get_feature_names(),f_test, pvals)), columns=['feature','f_reg','pva
    results[d+' F'] = df_f.sort_values(by = ['pvals'])
    display(results[d+' F'])
```

In [50]:
```
#BIKE dataset feature scores for best model, deg 3
Poly_feat_scores(datasets[0], dfs_enc[0], ys[0], 3)
```

Bike Sharing

No of features 11

	col_name	col_rep
0	season	x0
1	yr	x1
2	mnth	x2
3	holiday	x3
4	weekday	x4
5	workingday	x5
6	weathersit	x6
7	temp	x7
8	atemp	x8
9	hum	x9
10	windspeed	x10

	feature	MI
151	x1^2 x8	1.000000
30	x1 x8	0.990671
150	x1^2 x7	0.960507
29	x1 x7	0.956465
24	x1 x2	0.950856
...	...	...
274	x3 x8^2	0.000000
273	x3 x7 x10	0.000000
272	x3 x7 x9	0.000000
271	x3 x7 x8	0.000000
0	1	0.000000

364 rows × 2 columns

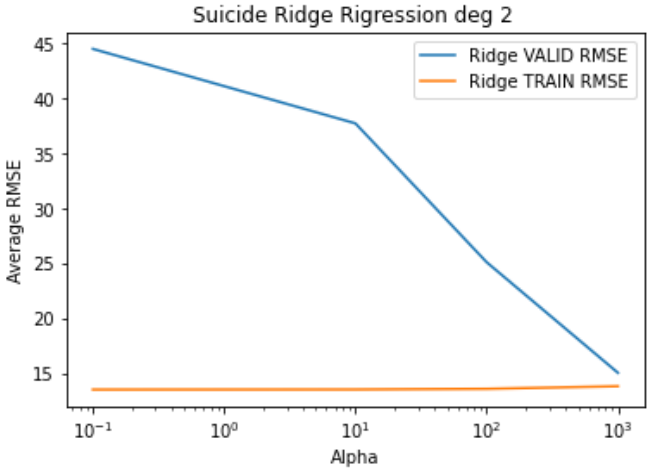
	feature	f_reg	pvals
9	x8	NaN	1.854504e-82
8	x7	NaN	2.810622e-81
89	x0 x1^2	NaN	7.678823e-76
13	x0 x1	NaN	7.678823e-76
2	x1	NaN	2.483540e-63
...	...	...	...
260	x3 x5 x6	NaN	NaN
261	x3 x5 x7	NaN	NaN
262	x3 x5 x8	NaN	NaN
263	x3 x5 x9	NaN	NaN
264	x3 x5 x10	NaN	NaN

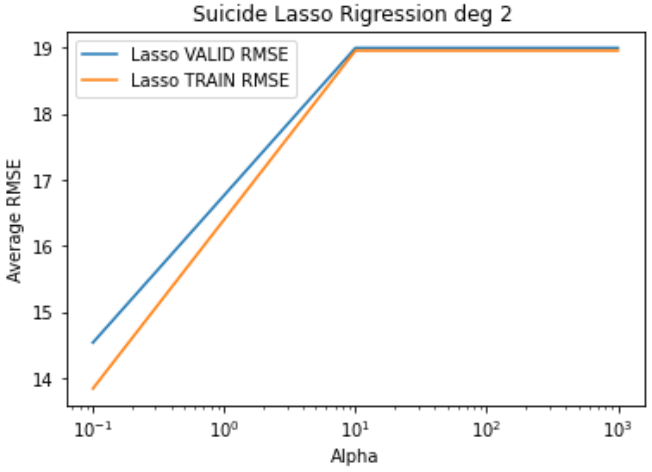
364 rows × 3 columns

In [ ]:

In [20…

```
#SUI dataset degree = 2
deg = 2
#alphas = np.logspace(-3, 3, 7)
alphas = [1e-1,1e1,1e2,1e3]
Poly_Reg_CV_rmse(datasets[1],dfs[1], ys[1], drop_featmf[1], alphas, deg)
```





Suicide Ridge Rigression deg 2

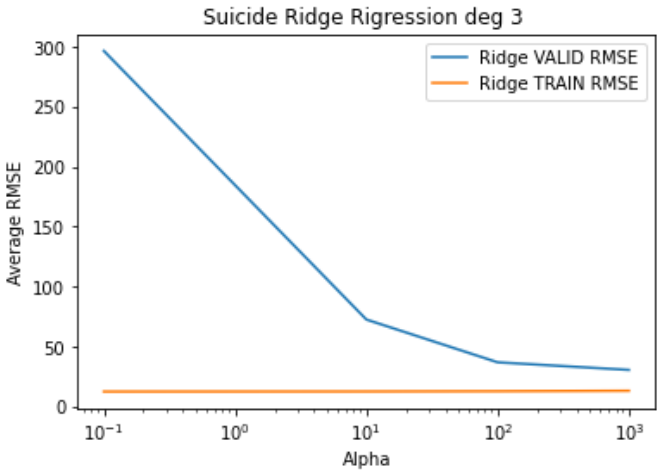
Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.1	13.492286	44.507807
10.0	13.501272	37.728411
100.0	13.559748	25.085427
1000.0	13.796183	15.018599

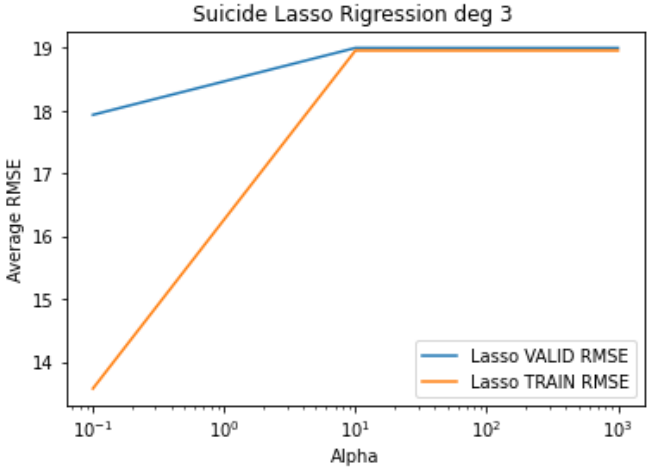
Suicide Lasso Rigression deg 2

Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.1	13.842205	14.537597
10.0	18.959137	18.999758
100.0	18.959137	18.999758
1000.0	18.959137	18.999758

In [21…

```
#SUI dataset degree = 3
deg = 3
#alphas = np.logspace(-3, 3, 7)
alphas = [1e-1,1e1,1e2,1e3]
Poly_Reg_CV_rmse(datasets[1],dfs[1], ys[1], drop_featmf[1], alphas, deg)
```





Suicide Ridge Rigression deg 3

Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.1	12.502354	296.313013
10.0	12.594297	72.522844
100.0	12.787766	36.886910
1000.0	13.190576	30.570586

Suicide Lasso Rigression deg 3

Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.1	13.573795	17.934474
10.0	18.958874	19.000564
100.0	18.959137	18.999758
1000.0	18.959137	18.999758

For Suicide dataset, RIdge regression's best model appears at alpha = 1000, degree 2. Lasso: alpha = 0.1 , degree 2

In [51]:
```
#SUI dataset feature scores for best model, deg 2
Poly_feat_scores(datasets[1], dfs_enc[1], ys[1], 2)
```

```
Suicide
No of features 8
```

	col_name	col_rep
0	year	x0
1	sex	x1
2	age	x2
3	population	x3
4	gdp_for_year ($)	x4
5	gdp_per_capita ($)	x5
6	generation	x6
7	continent	x7

	feature	MI
4	x3	1.000000
25	x2 x3	0.818274
30	x3^2	0.768640
34	x3 x7	0.763826
33	x3 x6	0.675059
19	x1 x3	0.585638
18	x1 x2	0.484141
3	x2	0.464899
26	x2 x4	0.461452
24	x2^2	0.459720
28	x2 x6	0.454680
29	x2 x7	0.448323
38	x4 x7	0.423222
31	x3 x4	0.422790
37	x4 x6	0.407669
20	x1 x4	0.396524
21	x1 x5	0.381208
5	x4	0.377263
41	x5 x7	0.366007
35	x4^2	0.344472
36	x4 x5	0.342829
6	x5	0.327209
23	x1 x7	0.323210
14	x0 x5	0.317748
39	x5^2	0.304424
13	x0 x4	0.301391
22	x1 x6	0.275643

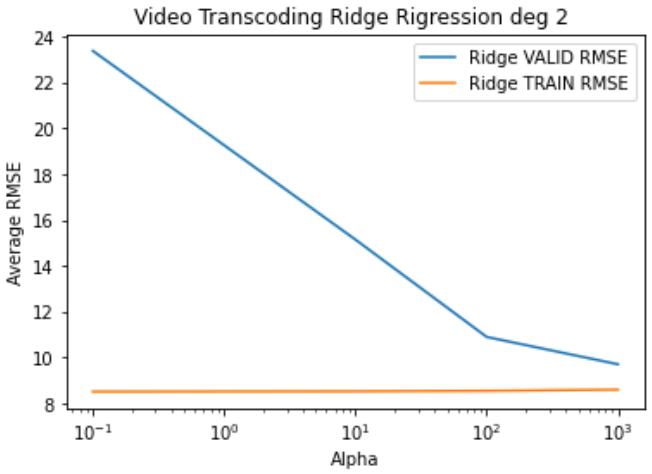
	feature	MI
7	x6	0.265780
42	x6^2	0.264016
43	x6 x7	0.258342
40	x5 x6	0.244298
27	x2 x5	0.238872
17	x1^2	0.236130
2	x1	0.236007
11	x0 x2	0.235375
10	x0 x1	0.217463
32	x3 x5	0.209627
12	x0 x3	0.201230
15	x0 x6	0.181832
8	x7	0.145537
44	x7^2	0.144076
16	x0 x7	0.071824
0	1	0.003412
1	x0	0.000000
9	x0^2	0.000000

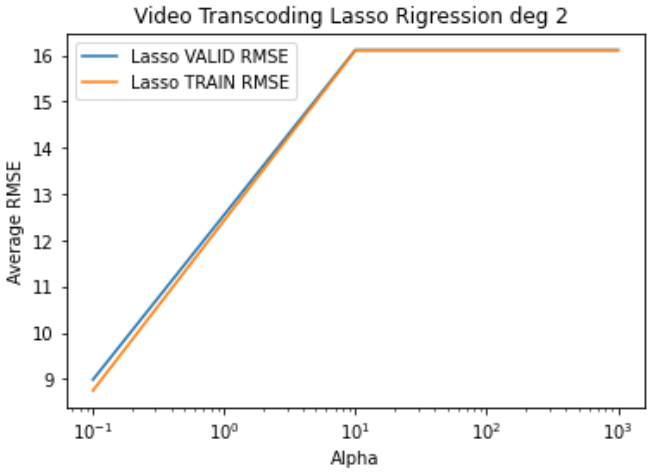
	feature	f_reg	pvals
22	x1 x6	NaN	0.000000e+00
2	x1	NaN	0.000000e+00
24	x2^2	NaN	0.000000e+00
23	x1 x7	NaN	0.000000e+00
18	x1 x2	NaN	0.000000e+00
17	x1^2	NaN	0.000000e+00
10	x0 x1	NaN	0.000000e+00
3	x2	NaN	7.739320e-218
11	x0 x2	NaN	1.336458e-71
29	x2 x7	NaN	1.440259e-53
28	x2 x6	NaN	1.056041e-49
44	x7^2	NaN	1.630703e-22
43	x6 x7	NaN	3.815148e-20
7	x6	NaN	9.218445e-17
9	x0^2	NaN	1.819911e-16
8	x7	NaN	5.891869e-14
1	x0	NaN	7.353434e-11
16	x0 x7	NaN	2.308113e-09
15	x0 x6	NaN	4.048636e-09
26	x2 x4	NaN	2.720250e-06

	feature	f_reg	pvals
5	x4	NaN	2.550492e-05
20	x1 x4	NaN	7.084222e-05
25	x2 x3	NaN	7.124447e-05
37	x4 x6	NaN	4.148544e-04
41	x5 x7	NaN	6.523963e-04
13	x0 x4	NaN	1.646318e-03
34	x3 x7	NaN	4.471774e-03
32	x3 x5	NaN	6.415495e-03
39	x5^2	NaN	7.166056e-03
30	x3^2	NaN	3.243545e-02
36	x4 x5	NaN	3.357656e-02
38	x4 x7	NaN	3.490003e-02
21	x1 x5	NaN	5.202615e-02
14	x0 x5	NaN	5.424841e-02
12	x0 x3	NaN	9.037717e-02
27	x2 x5	NaN	1.564885e-01
4	x3	NaN	1.670210e-01
35	x4^2	NaN	1.874319e-01
31	x3 x4	NaN	3.287313e-01
40	x5 x6	NaN	3.851062e-01
19	x1 x3	NaN	3.920588e-01
33	x3 x6	NaN	6.667680e-01
6	x5	NaN	7.659053e-01
42	x6^2	NaN	7.929940e-01
0	1	NaN	NaN

In [23…

```
#VID dataset degree = 2
deg = 2
#alphas = np.logspace(-3, 3, 7)
alphas = [1e-1,1e1,1e2,1e3]
Poly_Reg_CV_rmse(datasets[2],dfs[2], ys[2], drop_featmf[2], alphas, deg)
```





Video Transcoding Ridge Rigression deg 2

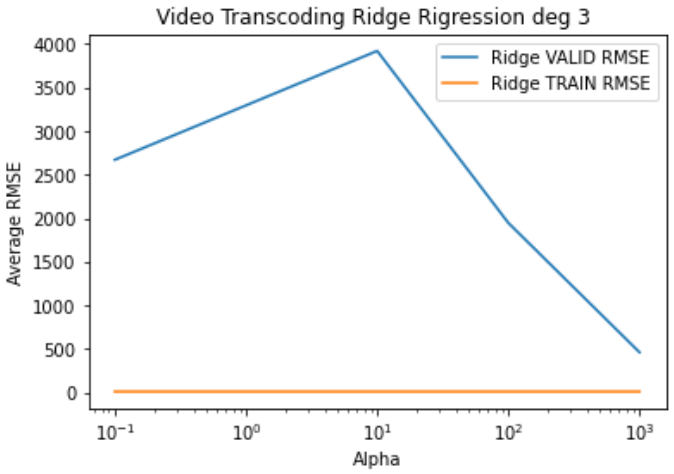
Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.1	8.519758	23.364192
10.0	8.531022	15.143170
100.0	8.551798	10.900243
1000.0	8.604859	9.709025

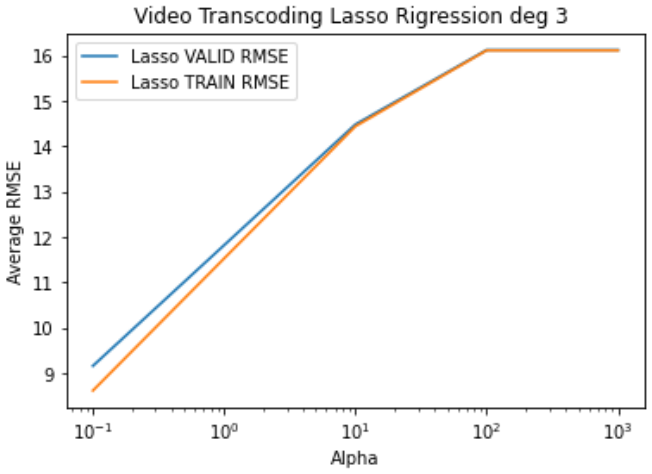
Video Transcoding Lasso Rigression deg 2

Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.1	8.751143	8.985448
10.0	16.106800	16.116994
100.0	16.106800	16.116994
1000.0	16.106800	16.116994

In [23…

```
#VID dataset degree = 3
deg = 3
#alphas = np.logspace(-3, 3, 7)
alphas = [1e-1,1e1,1e2,1e3]
Poly_Reg_CV_rmse(datasets[2],dfs[2], ys[2], drop_featmf[2], alphas, deg)
```





Video Transcoding Ridge Rigression deg 3

Alpha	Ridge Train RMSE	Ridge Valid RMSE
0.1	8.078538	2670.531117
10.0	8.123090	3921.609979
100.0	8.166838	1945.703544
1000.0	8.254043	459.014726

Video Transcoding Lasso Rigression deg 3

Alpha	Lasso Train RMSE	Lasso Valid RMSE
0.1	8.625249	9.168273
10.0	14.436610	14.478908
100.0	16.106800	16.116994
1000.0	16.106800	16.116994

For VIDEO dataset, Ridge regression: alpha = 1000 and deg = 2

Lasso: alpha = 0.1 and deg = 2

In [52]:
```
#VID dataset feature scores for best model, deg 2
Poly_feat_scores(datasets[2], dfs_enc[2], ys[2], 2)
```

```
Video Transcoding
No of features 18
```

	col_name	col_rep
0	duration	x0
1	codec	x1
2	width	x2
3	height	x3
4	bitrate	x4
5	framerate	x5
6	i	x6
7	p	x7
8	b	x8
9	frames	x9
10	i_size	x10
11	p_size	x11
12	size	x12
13	o_codec	x13
14	o_bitrate	x14
15	o_framerate	x15
16	o_width	x16
17	o_height	x17

	feature	MI
97	x4 x16	1.000000
98	x4 x17	0.989692
160	x10 x16	0.973940
161	x10 x17	0.964778
168	x11 x17	0.959194
...	...	...
114	x6 x8	0.001267
136	x8 x9	0.001206
125	x7 x8	0.000500
60	x2 x8	0.000000
138	x8 x11	0.000000

190 rows × 2 columns

	feature	f_reg	pvals
189	x17^2	NaN	0.000000
15	x14	NaN	0.000000
183	x14 x17	NaN	0.000000
178	x13 x16	NaN	0.000000
53	x1 x17	NaN	0.000000
...	...	...	...

	feature	f_reg	pvals
158	x10 x14	NaN	0.914047
19	x0^2	NaN	0.915946
25	x0 x6	NaN	0.949275
24	x0 x5	NaN	0.962540
0	1	NaN	NaN

190 rows × 3 columns

In [22]:

```
def Poly_Recursive(d, df, Y, alphas, deg):

    alphar, alphal = alphas

    X = df.drop(columns = Y)
    X = pd.get_dummies(X)
    y = df[Y]

    cols = X.columns
    feat_names = ['x'+str(i) for i in range(len(cols))]
    print('')
    print(list(zip(cols, feat_names)))

    poly = PolynomialFeatures(deg)
    X = poly.fit_transform(X)

    print("RIDGE REGRESSION")
    rid_reg = Ridge(alpha = alphar,max_iter = 1e5)
    min_features_to_select = 1
    rfecv = RFECV(estimator=rid_reg, step=1, cv=KFold(10),
                  scoring='neg_mean_squared_error',
                  min_features_to_select=min_features_to_select)

    rfecv.fit(X, y)
    cv_rmse_scores = np.sqrt(-rfecv.grid_scores_)

    print("Optimal number of features : %d" % rfecv.n_features_)
    print("CV rmse: {:.3f}".format(np.amin(cv_rmse_scores)))
    print("Optimal features selected:")
    print(rfecv.ranking_[rfecv.ranking_==1])
    #print("second best features:")
    #print(X.columns[rfecv.ranking_==2])

    # Plot number of features VS. cross-validation scores
    plt.figure()
    plt.xlabel("Number of features selected")
    plt.ylabel("CV RMSE")
    plt.title(d+" Recursive feature selection")
    plt.plot(range(min_features_to_select,
                   len(rfecv.grid_scores_) + min_features_to_select),
             cv_rmse_scores)
    plt.show()

    print("**************************************************************************")
    print("LASSO REGRESSION")
    las_reg = Lasso(alpha = alphal,max_iter = 1e5)
    rfecv = RFECV(estimator=rid_reg, step=1, cv=KFold(10),
                  scoring='neg_mean_squared_error',
                  min_features_to_select=min_features_to_select)

    rfecv.fit(X, y)
    cv_rmse_scores = np.sqrt(-rfecv.grid_scores_)

    print("Optimal number of features : %d" % rfecv.n_features_)
    print("CV rmse: {:.3f}".format(np.amin(cv_rmse_scores)))
    print("Optimal features selected:")
```

```
    print(rfecv.ranking_[rfecv.ranking_==1])
    #print(X.columns[rfecv.ranking_==1])
    #print("second best features:")
    #print(X.columns[rfecv.ranking_==2])

    # Plot number of features VS. cross-validation scores
    plt.figure()
    plt.xlabel("Number of features selected")
    plt.ylabel("CV RMSE")
    plt.title(d+" Recursive feature selection")
    plt.plot(range(min_features_to_select,
                   len(rfecv.grid_scores_) + min_features_to_select),
             cv_rmse_scores)
    plt.show()
```

Numerical features could be inverted.

Frame rate could be inverted to seconds per frame. The new feature multiplied with i frames, p frames, b frames gives the duration of iframes ,p frames and b frames in the video, which might give a better understanding of the total transcoding time.

In [19…

```
poly = PolynomialFeatures(2)
X = dfs[0]
print(list(zip(X.columns,np.arange())
poly.fit_transform(dfs[0])
poly.get_feature_names()
```

Out[199]:

```
['1',
 'x0',
 'x1',
 'x2',
 'x3',
 'x4',
 'x5',
 'x6',
 'x7',
 'x8',
 'x9',
 'x10',
 'x11',
 'x0^2',
 'x0 x1',
 'x0 x2',
 'x0 x3',
 'x0 x4',
 'x0 x5',
 'x0 x6',
 'x0 x7',
 'x0 x8',
 'x0 x9',
 'x0 x10',
 'x0 x11',
 'x1^2',
 'x1 x2',
 'x1 x3',
 'x1 x4',
 'x1 x5',
 'x1 x6',
 'x1 x7',
 'x1 x8',
 'x1 x9',
 'x1 x10',
 'x1 x11',
 'x2^2',
 'x2 x3',
 'x2 x4',
 'x2 x5',
 'x2 x6',
 'x2 x7',
 'x2 x8',
 'x2 x9',
```

```
'x2 x10',
'x2 x11',
'x3^2',
'x3 x4',
'x3 x5',
'x3 x6',
'x3 x7',
'x3 x8',
'x3 x9',
'x3 x10',
'x3 x11',
'x4^2',
'x4 x5',
'x4 x6',
'x4 x7',
'x4 x8',
'x4 x9',
'x4 x10',
'x4 x11',
'x5^2',
'x5 x6',
'x5 x7',
'x5 x8',
'x5 x9',
'x5 x10',
'x5 x11',
'x6^2',
'x6 x7',
'x6 x8',
'x6 x9',
'x6 x10',
'x6 x11',
'x7^2',
'x7 x8',
'x7 x9',
'x7 x10',
'x7 x11',
'x8^2',
'x8 x9',
'x8 x10',
'x8 x11',
'x9^2',
'x9 x10',
'x9 x11',
'x10^2',
'x10 x11',
'x11^2']
```

In [19…

```
type(aa)
```

Out[198]: method

In [ ]:

```
#Computing Fscores for numerical features alone
results_cont = {}

for d, df, y, cnames in zip(datasets, dfs, ys, cat_names):
    print(d)

    X = df.drop(columns=[y]+cnames)
    y = df[y]

    # mutual information
    mi = mutual_info_regression(X, y)
    mi /= np.max(mi)

    #fregression
    f_test, pvalues = f_regression(X,y)
    f_test /= np.max(f_test)


    df_f = pd.DataFrame(list(zip(X.columns, mi, f_test, pvalues)), columns=['Continuous features', 'mi','f
```

```
    results_cont[d] = df_f

    display(df_f)
```

Neural Networks

In [53]:
```
def mlp(d,df,Y,hid_lay_size,alpha,lr,batch_size):

    X = df.drop(columns = Y)
    X = pd.get_dummies(X)
    y = df[Y]

    nn_reg = MLPRegressor(hid_lay_size, activation = 'relu', alpha = alpha, max_iter=500,learning_rate_ini
    scores = cross_validate(nn_reg, X, y, scoring="neg_mean_squared_error", cv=10, return_train_score=Tru

    return np.sqrt(np.mean(-scores['test_score'])), np.sqrt(np.mean(-scores['train_score']))
```

In [ ]:
```
hid_size_list = [[300, 500, 200],[200,500,500,200],[300,600,600,600,200]]
al_list = [1e-1,1,1e2,1e3]
lr_list = [1e-4, 5e-3, 1e-1]
for d, df, Y in zip(datasets, dfs, ys):
    nn_list = []
    print(d+" MLP")
    for hid_size in hid_size_list:
        for al in al_list:
            for lr in lr_list:
                val_score, tr_score = mlp(d,df,Y,hid_size, al,lr,512)
                nn_list.append([hid_size, val_score, tr_score, lr])
nn_df = pd.DataFrame(nn_list, columns = ['Hidden Size', 'Avg Valid RMSE', 'Avg Train RMSE', 'LR'])
display(nn_df)
```

In [54]:
```
hid_size_list = [[300, 500, 200],[200,500,500,200],[300,600,600,600,200]]
al_list = [1e-1,1e1,1e2,1e3]
lr_list = [1e-4, 5e-3, 1e-1]
d = datasets[2]
df = dfs[2]
Y = ys[2]
#for d, df, Y in zip(datasets, dfs, ys):
nn_list = []
print(d+" MLP")
for hid_size in hid_size_list:
    for al in al_list:
        for lr in lr_list:
            val_score, tr_score = mlp(d,df,Y,hid_size, al,lr,512)
            nn_list.append([hid_size, val_score, tr_score, lr])
nn_df = pd.DataFrame(nn_list, columns = ['Hidden Size', 'Avg Valid RMSE', 'Avg Train RMSE', 'LR'])
display(nn_df)
```

```
Video Transcoding MLP
```

	Hidden Size	Avg Valid RMSE	Avg Train RMSE	LR
0	[300, 500, 200]	6.286149	2.390788	0.0001
1	[300, 500, 200]	5.711178	2.671341	0.0050
2	[300, 500, 200]	7.117106	5.862018	0.1000
3	[300, 500, 200]	6.048427	3.027596	0.0001
4	[300, 500, 200]	5.666926	2.946742	0.0050
5	[300, 500, 200]	8.565896	7.027464	0.1000
6	[300, 500, 200]	5.474537	4.048715	0.0001
7	[300, 500, 200]	5.303775	3.919869	0.0050

	Hidden Size	Avg Valid RMSE	Avg Train RMSE	LR
8	[300, 500, 200]	6.742069	6.162259	0.1000
9	[300, 500, 200]	6.839218	6.413917	0.0001
10	[300, 500, 200]	6.825335	6.490337	0.0050
11	[300, 500, 200]	7.741897	8.281930	0.1000
12	[200, 500, 500, 200]	6.010435	2.701074	0.0001
13	[200, 500, 500, 200]	5.759734	2.665932	0.0050
14	[200, 500, 500, 200]	10.925547	10.528532	0.1000
15	[200, 500, 500, 200]	6.057106	2.843946	0.0001
16	[200, 500, 500, 200]	5.587379	2.989653	0.0050
17	[200, 500, 500, 200]	13.599626	13.408736	0.1000
18	[200, 500, 500, 200]	5.803174	3.409466	0.0001
19	[200, 500, 500, 200]	5.472520	3.574046	0.0050
20	[200, 500, 500, 200]	13.177060	13.116358	0.1000
21	[200, 500, 500, 200]	6.285845	5.648885	0.0001
22	[200, 500, 500, 200]	6.344535	5.808220	0.0050
23	[200, 500, 500, 200]	9.565461	9.334768	0.1000
24	[300, 600, 600, 600, 200]	6.018228	2.665176	0.0001
25	[300, 600, 600, 600, 200]	5.687870	2.651244	0.0050
26	[300, 600, 600, 600, 200]	15.485387	15.217684	0.1000
27	[300, 600, 600, 600, 200]	5.890392	2.751719	0.0001
28	[300, 600, 600, 600, 200]	5.612401	3.017464	0.0050
29	[300, 600, 600, 600, 200]	15.913271	15.720486	0.1000
30	[300, 600, 600, 600, 200]	6.001243	3.174738	0.0001
31	[300, 600, 600, 600, 200]	5.326107	3.463364	0.0050
32	[300, 600, 600, 600, 200]	15.815059	15.684552	0.1000
33	[300, 600, 600, 600, 200]	5.989316	5.198434	0.0001
34	[300, 600, 600, 600, 200]	6.047226	5.496345	0.0050
35	[300, 600, 600, 600, 200]	16.060417	16.162939	0.1000

In [ ]: