Report

EP2300 Project - Network Management Estimating Conformance to Service Level Agreements (SLAs) using Machine Learning

Submitted by: Aman Kumar Gulia

Task 1:

1. Compute the following statistics for each feature of X and target of Y: mean, maximum, minimum, 25th percentile, 90th percentile, and standard deviation. Give no more than two digits after decimal point.

Solution:

Describe() method can be used to derive the statistics for the dataset X and Y individually. But since describe method displays extra attributes by default like 'count', we need to eliminate columns that are not required by using iloc indexer. Moreover, we need 90th percentile. To incorporate this, we can specify 0.9 as an attribute to 'percentile' parameter in describe(). The values can be rounded to 2 decimal digits using round().

TimeStamp column is removed as it is not a feature of X and Y.

Result:

Statistics of X features:

	plist-sz	totsck	ldavg-1	pgfree/s	proc/s	all_%%usr	file-nr	\
mean	875.69	484.52	73.30	150234.32	8.00	86.22	2580.47	
std	267.28	132.24	48.20	30798.85	9.03	18.15	184.89	
min	404.00	241.00	1.79	180.00	0.00	1.96	2112.00	
25%	611.00	354.00	20.62	131677.25	0.00	74.73	2400.00	
50%	837.00	469.00	65.01	148475.50	6.00	96.88	2592.00	
90%	1253.20	672.00	135.70	185566.70	21.00	97.50	2832.00	

	cswch/s	%%memused	runq-sz
mean	52519.94	13.27	63.41
std	20576.24	1.86	37.08
min	2730.00	6.19	0.00
25%	29381.00	12.23	21.00
50%	63336.00	13.00	66.00
90%	72283.70	15.83	111.00

Statistics of Y features:

	DispFrames
mean	18.88
std	5.46
min	0.00
25%	13.39
50%	20.27
90%	24.00

2. Compute the following quantities of X:

(a) The number of observations with CPU utilization ("all %%usr") smaller than 90% and memory utilization ("%%memused") smaller than 50%.

Solution:

& operator is used to specify conditions and the result is again stored in a data-set X. The concatenation operation count() will count the number of entries.

RESULT:

```
Number of observations with CPU utilization (“all %%usr”) smaller than 90% and
memory utilization (“%%memused”) smaller than 50% :  1114
```

(b) The average number of used sockets (“totsck”) for observations with less than 60 000 context switches per seconds (“cswch/s”).

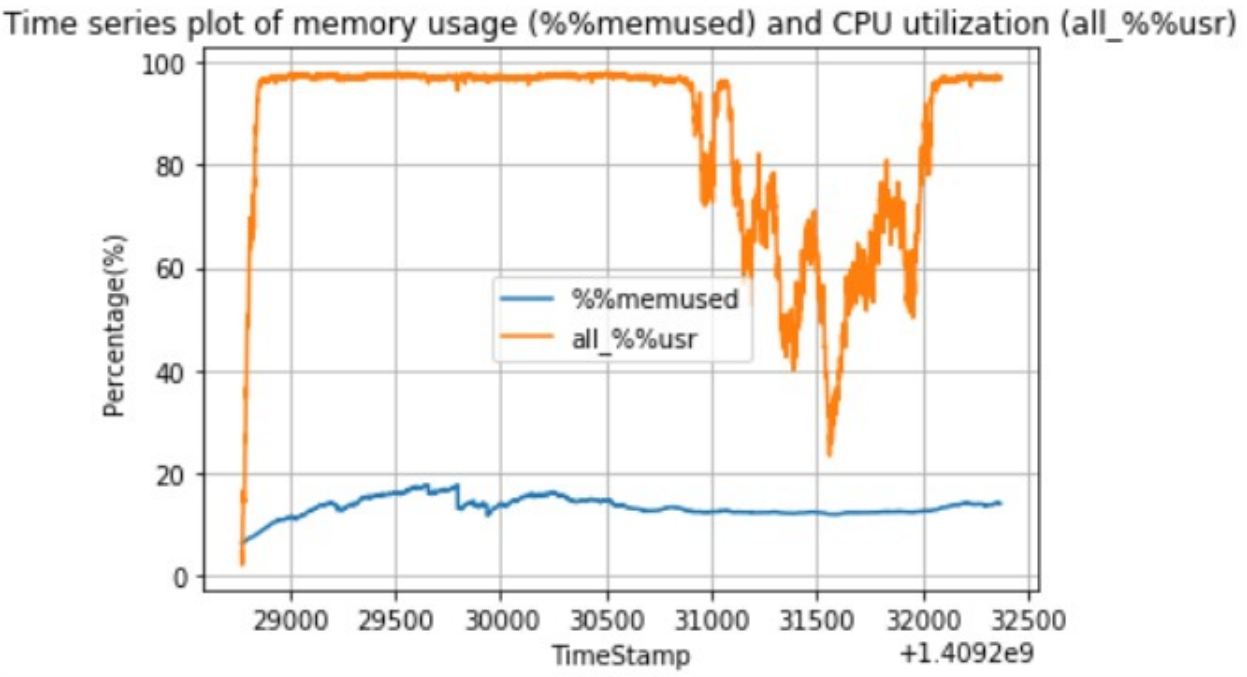
Solution:
Comparison operator < will display the observations with less than 60000 context switches per second.

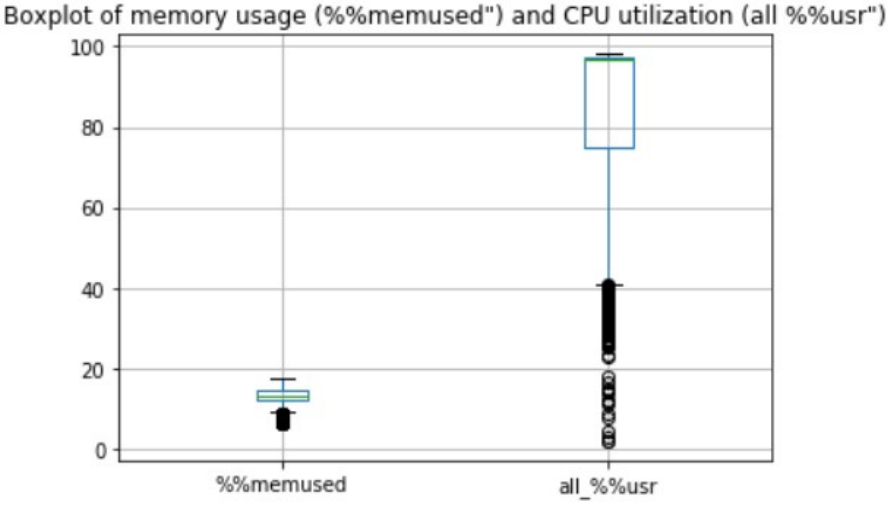
Result:

```
 The average number of used sockets (“totsck”) for observations with less than
60 000 context switches per seconds (“cswch/s”) :  356.1242564441507
```

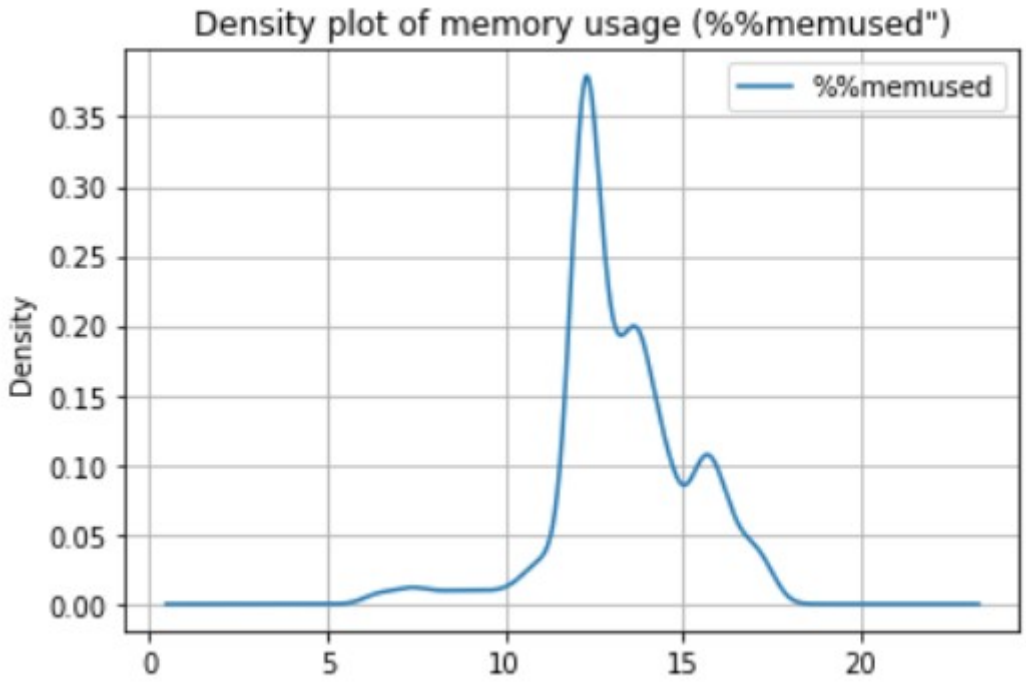
3. Produce the following plots:
(a) Time series of memory usage (“%%memused”) and CPU utilization (“all %%usr”), both curves in a single plot. Box plot of both features in a single plot.

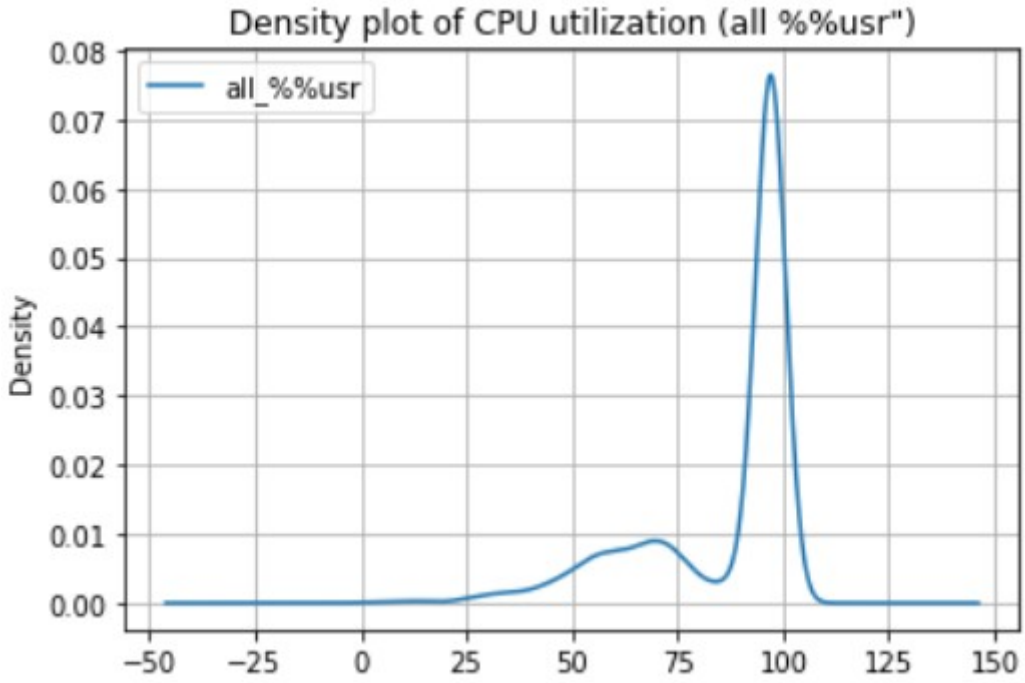
Solution:
matplotlib.pyplot provides method ‘plot()’ to plot time series by providing parameters for x axis (TimeStamp), and y axis(%%memused and all_%%usr).
DataFrame.box.plot() plots boxplot which takes only y axis parameters(%%memused and all_%%usr)

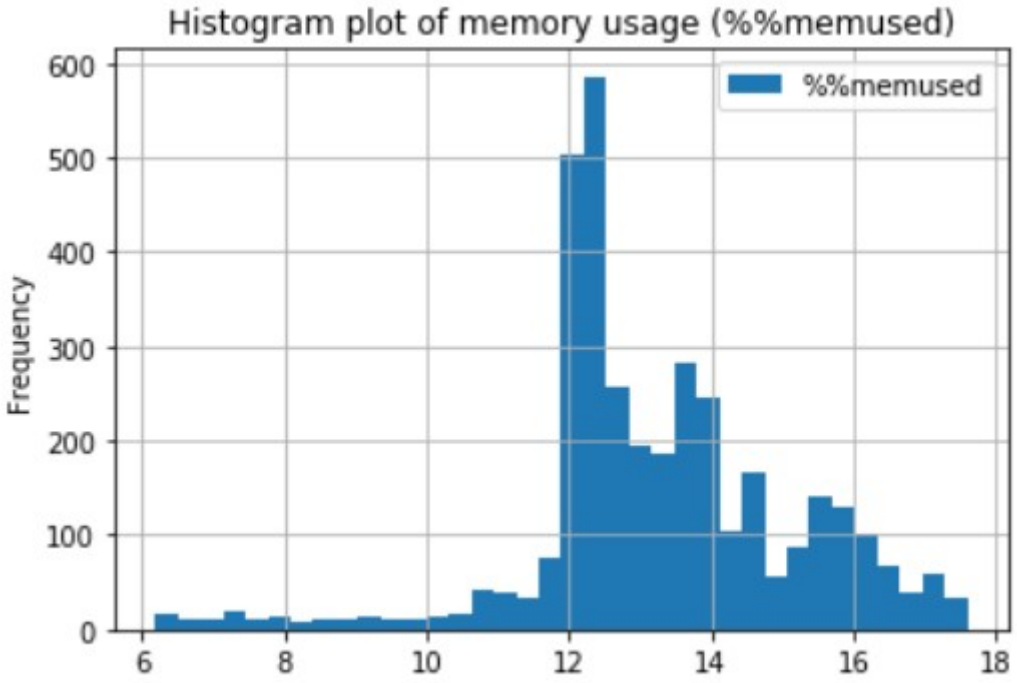


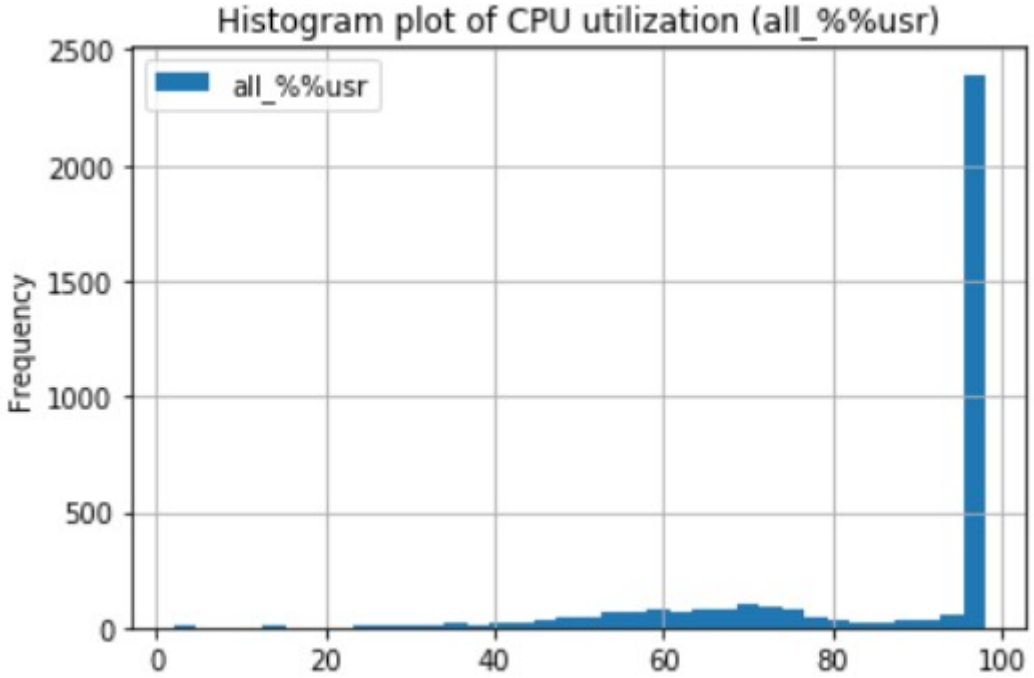


(b) Density plots of memory usage (“%%memused”) and CPU utilization (“all %%usr”), Histograms of both these features (choose a bin size of 1%), four plots in all.






Histogram plot of memory usage (%%memused)
%%memused
Frequency
600
500
400
300
200
100
0
6
8
10
12
14
16
18


Histogram plot of CPU utilization (all_%%usr)
all_%%usr
Frequency
2500
2000
1500
1000
500
0
0
20
40
60
80
100

Task 2 :

1(a)Model Training - use linear regression to train a model M with the training set. Provide the coefficients (Θ 0 , ..., Θ 10) of your model M . (Θ 0 is the offset.) Give no more than three significant digits.

Solution:
Timestamp column is dropped from the data set X and Y.
Datasets X and Y are then split into training and test sets.(X_train, X_test, y_train, y_test)
Sklearn's LinearRegression function will create an object for regression.
Using this object(regr_obj), we train the model using fit() with arguments as X_train and y_train.
Attribute coef_ will then provide us the coefficients.
Panda function set_printoptions(precision=2) lets us print only 3 significant digits in the output for displayng the coefficients.

Result:

```
Coefficients limited to 3 significant digits :

  [-1.43e-02 -1.35e-03  1.91e-03 -5.07e-06 -5.96e-03  8.96e-02 -3.38e-03
 -7.67e-05  4.25e-01 -1.41e-02]


Intercept in the linear regression model :  33.04886018304596
```

1(b) Accuracy of Model M - compute the estimation error of M on the test set. We define the estimation error as the Normalized Mean Absolute Error.
As a baseline for M , use a naïve method which relies on Y values only. For each x ∈ X it predicts a constant value ȳ which is the sample mean of the samples y i in the training set. Compute ȳ for the naïve method for the training set and compute the NMAE for the test set.

Solution:
Result from Mean_absolute_error(y_test,y_predict) is divided with mean of y_test observations to find Normalised Mean absolute error/ Estimation Error.
Result:

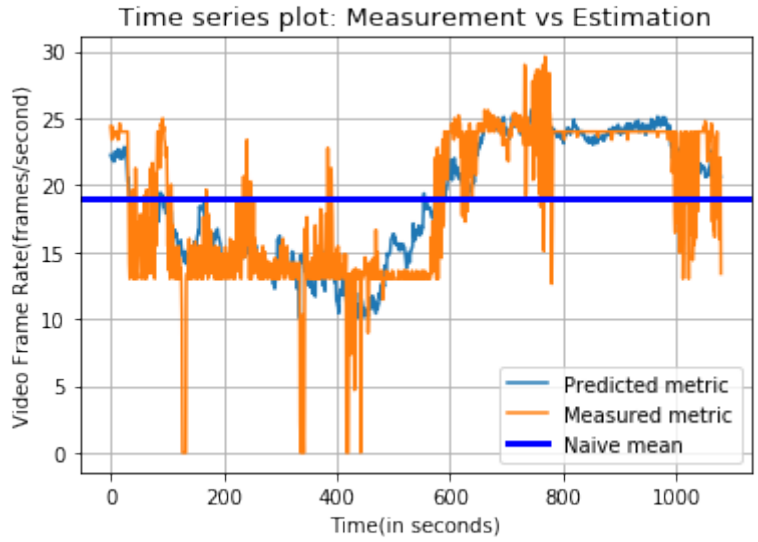
```
Mean Absolute error: 2.01
Mean of Y test set observations :  18.61513
Normalised Mean Absolute Error :  0.10784
```

From the problem stated, the naive method will require change in the mean to calculate the normalised mean absolute error. Mean (naive_mean) will now will be calculated using the Y_train observations instead of Y_test as in previous case. Also, the NMAE will be calculated with arguments y_train and naive_mean

```
NAIVE METHOD RESULTS:
Mean Absolute error: 5.00
Mean of Y train set observations :  18.99962
Normalised Mean Absolute Error for Naive method :  0.26327
```

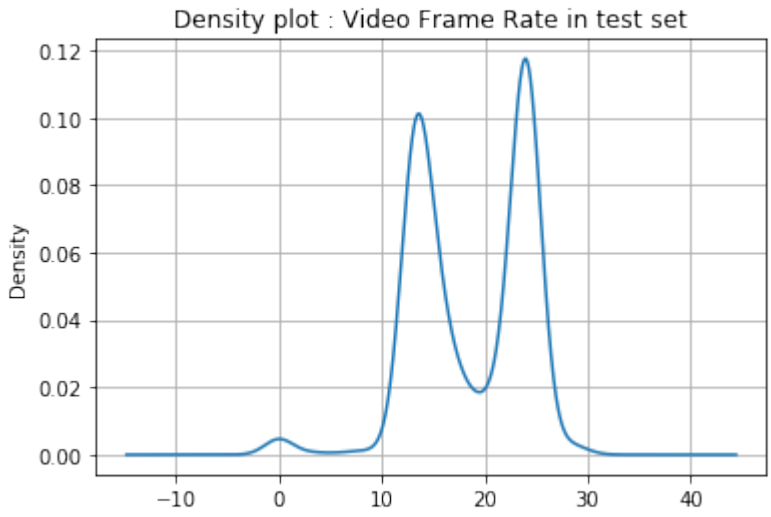
1(c)Produce a time series plot that shows both the measurements and the model estimations for M for the Video Frame Rate values in the test set (see example of such a plot in Figure 4(a) of [1]). Show also the prediction of the a naïve method.

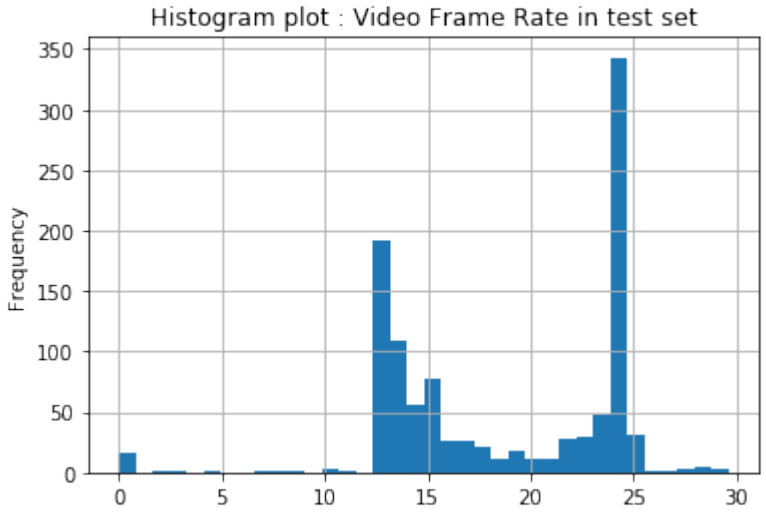
Solution:
To prepare the data-sets y_predict and y_test for plotting, both these data-sets are sorted according to the timestamp.
matplotlib's plot method will plot measured metric and prediction metric values as line graph.
Axhline will be used to add a horizontal line which denotes the naive mean value calculated in the previous question.



1(d)Produce a density plot and a histogram for the Video Frame Rate values in the test set. Set the bin size of the histogram to 1 frame.

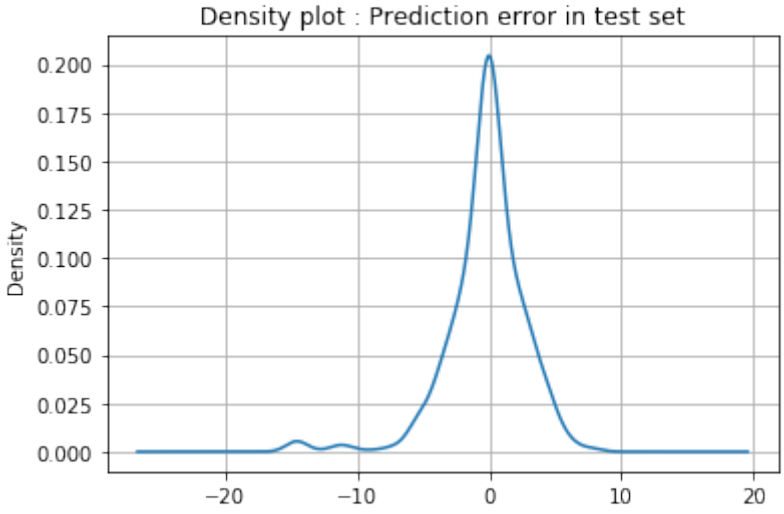
Solution:
Density plot is plotted using DataFrame.plot.kde() where y axis is DispFrames feature from y_test.
Histogram plot is plotted using DataFrame.plot.hist() where y axis is DispFrames feature from y_test along with bin size 36(1% of total samples 3600).





1(e) Produce a density plot for the prediction errors y i − ŷ i in the test set.

Solution:
The difference in y(i)- ŷ(i) is stored in another DataFrame p_err.
Kde() is then used to plot density points.



1(f) Based on the above figures and graphs, discuss the accuracy of estimating the Video Frame Rate.

Solution:
We employ linear regression modeling to estimate the Video Frame Rates by dividing the sample set X(Device Statistics) and Y(Service Statistics) into training set and test set.
The larger size of the training set ensures that the model M is trained enough times so that when the resulting trained model M is used to predict the values of Y from the test sample of X, the result is as close as possible to measured values of test sample Y.

In order to accurately predict the values of test sample, coefficients are required to create a model the linear regression problem as per the feature sets of X.
Training the model with training set x and y provides the coefficients.
The number of features considered plays an important role in determining the accuracy of the model. An optimal number of features which affect the video frame rate will give accurate measure

of the Y values in Test set when trained in Training set as compared to training suboptimal features set of X.

We employ normalised mean absolute error (NMAE) to determine the estimation error. This refers to the difference in values of predicted Y compared to measured Y in Test Set.
Another technique that provides fairly better accuracy is normalised mean square error.

Moreover, multi-variate regression models might provide further improved accuracy compared to linear regression model as it trains the model using more than one predictor variable. This may prove to be more tolerant to handling outliers than linear regression model.

2(a)From the above training set with 2520 observations, create six training sets by selecting uniformly at random 50, 100, 200, 500, 1000, and 2520 observations (which is the original set).

Solution:
Importing train_test_split from sklearn.model_selection will enable splitting the previously created training sets X_train and y_train into 6 new sets. The argument test_size will determine the number of observations to be included in the test set.

2(b)Train a linear model and compute the N M AE for each model for the original test set with 1080 observations.

Solution:
A user-defined function estimationerror() is defined in order to train linear regression model. This function is defined similar to 2.1(a). This function contains 4 arguments namely x_train,x_test,y_train and y_test. The values from newly created subsets will be passed to this function, which then initialises regression object regr_obj and fits the x_train and y_train values. y_predict is the result of the predict(x_test) method. The result of y_predict is then passed as an argument to mean_absolute_error(). Mean is calculated using y_test observations. mean_absolute_error divded by mean of y_test gives the NMAE for the particular subset.

X training set length : 50 NMAE : 0.1277717665318912

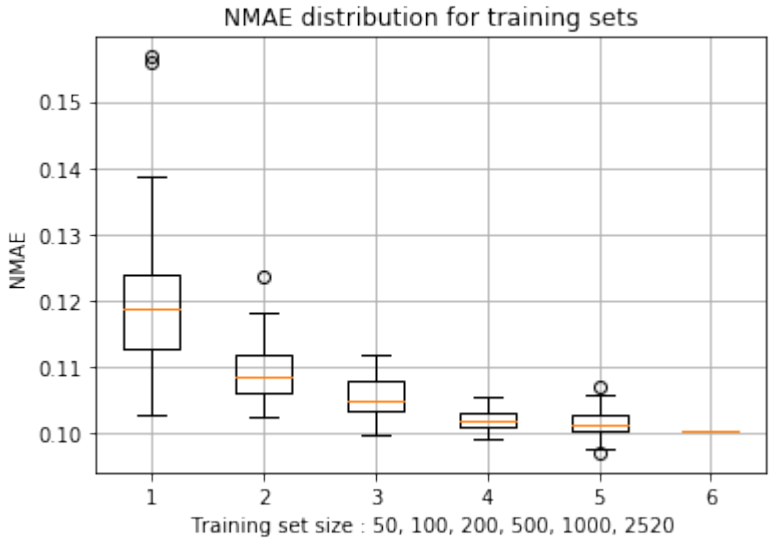
```
X training set length :  100    NMAE :  0.11800874812916437

X training set length :  200    NMAE :  0.10385837292491437

X training set length :  500    NMAE :  0.10173348752176319

X training set length :  1000   NMAE :  0.10209026346740915

X training set length :  2520   NMAE :  0.10028481932127198
```

2(c)Perform the above 50 times, so you train models for 50 different subsets of a given size.

Solution:
To train each subset 50 times, *for* loop is created to iteratively append output NMAE from estimationerror2() method to the subsequent DataFrame namely NMAE_50, NMAE_100,etc.

2(d)Produce a plot that shows NMAE for M against the size of the training set. Use error bars or box plots to show the range of the NMAE values for a given set size.

Solution:
The whisker box plot shows NMAE for training sets of 50,100,200,500,1000 and 2520 observation samples.



2(e)Based on the above, discuss the relationship between the accuracy of the model estimations and the training set.

Solution:
As shown by results in 2(b), we can observe that NMAE is seen to decrease towards 0 as we increase the training set sample length from 50 to 2520. This might lead an observer to an assumption that increasing the length of the training set sample length will automatically result in improved learning model and thereby reducing normalised Mean Absolute Error(NMAE). However, the estimation error observed here is only for 1 iteration of learning model M.

In 2(c), we train the Linear Regression Model M 50 times for 50 different subsets for a given size. This drastically improves the results of the model M as compared to only 1 iteration done previously.
As a result of this, the whisker box plot generated displays the outliers, maximum values, minimum values, median, lower quartile and upper quartile.

The median values for NMAE decrease as the observation set sample size increases.
Also, the gap between the 25^{th} percentile and 75^{th} percentile reduces proportionally as the observation set size increases.
However, comparing the outlier in the sample size 500 and 1000 provides interesting observation. We observe that the gap between the outlier(3/2 times of upper quartile) decreases for sample set size 50 to 500, but increases for sample set 500 with 1000.

Task 3:

1. Model Training - use Logistic Regression to train a classifier C with the training set. Provide the coefficients (Θ 0 , ..., Θ 10) of your model C. (Θ 0 is the offset.) Give no more than three significant digits.

Solution:

The data sets X and Y are trimmed by removing the TimeStamp column.
Both dataframes are split into training(2520 samples) and test set(1080 samples).
Y_train is classified as 0 and 1 as per the threshold value 18 Frames / sec. (Y_train_classifier)
1- Values greater than 18
0- Values less than or equal to 18
sklearn's linear model lets import Logistic regression method which will help to train the classifier model C. The model learns with the fit() method where the y_train_classifier and X_train are passed as arguments.
coef_ attribute returns the coefficients generated by the model according the features of X_train.
set_printoptions lets one return coefficients with 3 significant digits.

```
Coefficients limited to 3 significant digits :
  [[-1.16e-02 -3.94e-03 -4.54e-03  3.03e-06 -2.13e-04  5.71e-03  5.30e-03
  -3.24e-05  1.36e-03 -6.06e-03]]
Intercept in logistic model is :  [5.08e-05]
```

2. Accuracy of the Classifiers C - Compute the classification error (ERR) on the test set for C. Use confusion matrix plot to show True Positives (TP), True Negatives (TN), False Positives (FN), and False Negatives (FN).

Solution:

First step is to compute the y_predict with X_test sample. This is done using the trained classifier model C from task 3.1.
Confusion_matrix method from sklearn.metrics library lets us compare the results obtained from y_predict with true values in y_test_classifier.
This generates Tp, Tn, Fp and Fn. Results from one of the program execution gave the following values -

```
tp :  544
fn :  37
fp :  47
tn :  452
```

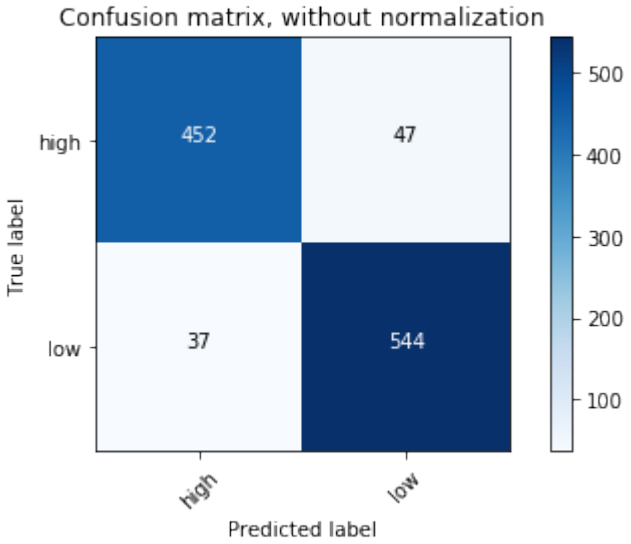
Classification error is calculated by the formula Err= 1 − (TP+T N)/m, whereby m is the number of observations in the test set. For the above values of confusion matrix, I obtained the following classification error.

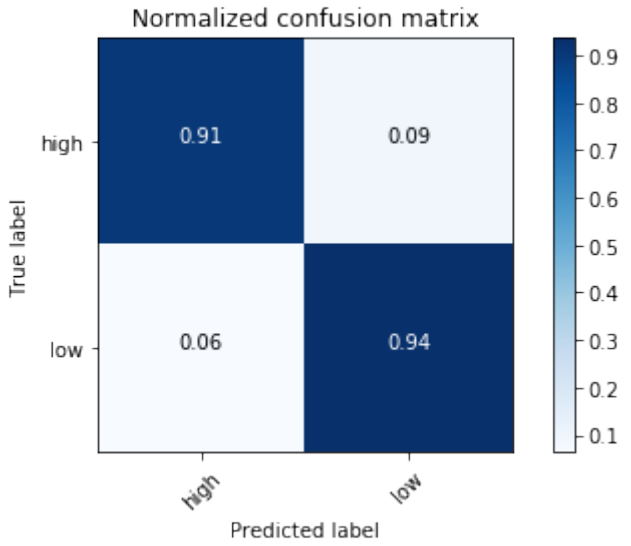
```
Classificiation error :  0.0778
```

Mathplot library is used to plot the confusion matrix.
User defined function plot_confusion_matrix() takes the arguments : confusion matrix, classes and title.
I have created two confusion matrix(normalized and without normalization).
Itertools method product() allows mapping of cartesian product with values in confusion matrix.

```
Confusion matrix, without normalization
[[452  47]
```

```
 [ 37 544]]
Normalized confusion matrix
[[0.91 0.09]
 [0.06 0.94]]
```





3. As a baseline for C, use a naive method which relies on Y values only, as follows. For each x ∈ X, the naive classifier predicts a value True with probability ‘p’ and False with probability 1 − p. p is the fraction of Y values that conform with the SLA. Compute ‘p’ on the training set and the classification error for the naive classifier on the test set.

Solution:

First step is to calculate the probability of 0’s and 1’s in Y_train_classifier, i.e. the probability with which samples conform the SLA theshold of 18 (True=1) or not(False=0). Let the probability ‘p’ resemble to True value of the naive Y trained set.
The naive method predicts the probability of 0’s and 1’s in the Y_test set with the probability ‘p’ calculated in Y_train_classifier.
Now, we need to find the total samples with 0’s(Y_test_total_0) and 1’s(Y_test_total_1) in Y_test set.

Now the classification error will be (1- (Y_test_total_0*(1-p)+ Y_test_total_1*p)/1080)

```
NAIVE METHOD :

Total positives in Y Test :  581
Total negatives in Y Test :  499
Classification error on test set with naive method :  0.4978
```

4. Build a new classifier by extending extend the linear regression function developed in Task II with a check on the output, i.e., the Video Frame Rate. If the frame rate for a given X is above the SLA threshold, then the Y label of the classifier is set to conformance, otherwise to violation. Compute the new classifier on the training set and the classification error for this new classifier on the test set.

Solution:

LinearRegression() method in the sklearn library is called to fit the X_train values with Y_train and train the model.
This model predicts the Y_test values with X_test samples,named y_predict_lin.
Values predicted greater than 18 are classified as 1 in the y_predict_lin_classifier dataframe and vice versa. This classifier is then fed into the confusion matrix function to provide Tn,Tp,Fn and Fp values.
With these values, we can now calculate the error for extended version of linear regression with the formula:
Err= 1-(Tp+Tn)/n, where n is the total number of samples in Y test.

```
CONFUSION MATRIX WITH LINEAR REGRESSION :
tp :  550
fn :  31
fp :  49
tn :  450
Classificiation error with linear regression :  0.0741
```

5. Formulate your observations and conclusions based on the above work.

Solution:

Classification modeling is used to used to segregate the samples into specific set of labels. It helps to distinguish between objects of different classes. Here, since we have binary classifier model, the labels are:
- Samples conforming with frame rate of 18
- Samples that have frame rate less than 18

As compared to the linear regression, this model does not try to fit the equation of a line to the points. Logistic Regression model fits the samples from training set of X with training set of Y by mapping them 0 or 1.
Mathematically, this involves the implementation of sigmoid function but conceptually it still represents the curve that depicts the likelihood of a sample belonging to a certain class given a linear combination of inputs.

Table 1 tabulates the results from execution of the program.

Regression Models	Classification	Naive	Extended Linear Regression
Error(in %)	8.52	49.78	7.31

Table 1 : Errors with different regression models

We observe that naive method achieves the most error for this data set. Extended linear regression outperforms Classification model. Classification model fairs second worst.

In our case, Naive is defeated the classifier model. This is due to the fact that naive method estimates the probability of X train as a probability of 1's and 0's and then uses this a benchmark to find tp and tn.

Extended linear regression achieves less error than classification model. This might be a result of the threshold chosen is good for linear regression convergence. It is observed that the classification error increases for extended linear regression when the threshold is set to 15 Frames per second.

```
Classificiation error :  0.0241
Classificiation error with linear regression :  0.0889
```

Similarly, when the threshold is set to 21.5 Frames per second, logistic regression fairs better than extended linear regression.
This shows that classification model in our case performs better when thresholds are set away from the mean value(18.8 Frames/sec).
Hence, 18 threshold might be an exceptional case where linear regression model trained to classify Y test performs better than logistic regression trained model.

Task 4:

1. Construct a training set and a test set from the trace as above.

Solution:

The data sets X and Y are trimmed by removing the TimeStamp column.
Both dataframes are split into training(2520 samples) and test set(1080 samples).

2. Method 1 (Optimal method; this method is appropriate if X contains a small number of features): Build all subsets of the feature set X. Compute a linear regression model for each of these subsets of the training set. For each model compute the error (NMAE) on the test set. Draw histograms of these errors. In addition, list the device statistics (i.e., features) of the model with the smallest error. Produce a plot that contains 10 box plots, one box plot each with the errors (NMAE) for all models that contain 1 feature, 2 features, ..., 10 features.

Solution:
All combination of feature subsets can be built with the combinations() method provided in the itertools library. It takes the dataframe and number of columns as arguments and returns the name of the columns.
Temporary data frames x_subset_train and x_subset_test inside the nested 'for' loop are created for calculating the NMAE for corresponding features. The NMAE are appended in a new dataframe nmae_cum_dataframe where we can sort the table based on descending values of NMAE.
The top 10 subsets with least NMAE are displayed below:

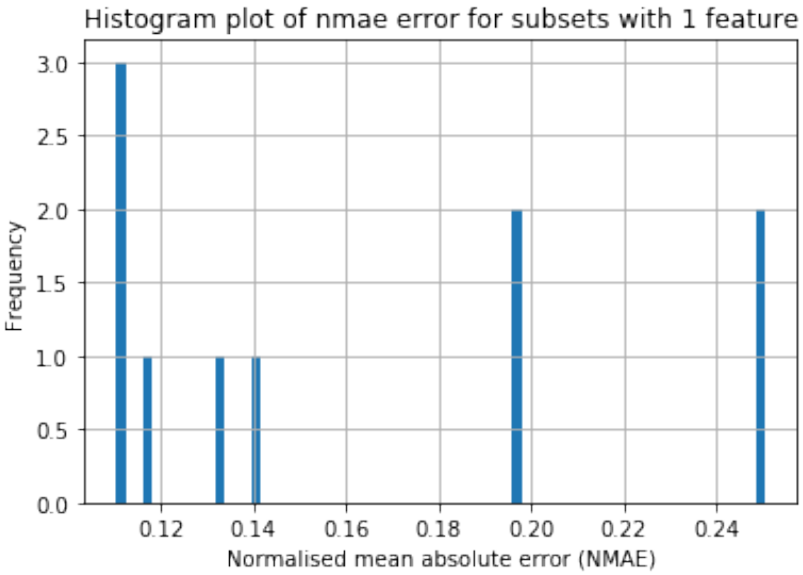
Results:

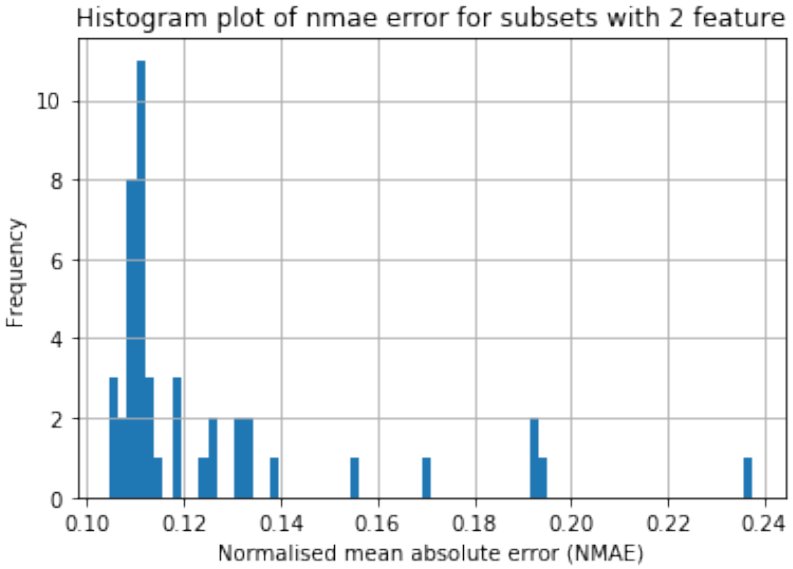
```
TASK 4.2 - Optimal method
Top 10 subsets with least NMAE in ascending order :
                                              FEATURE SET      NMAE
983   (totsck, ldavg-1, pgfree/s, all_%%usr, file-nr...  0.099119
867   (totsck, pgfree/s, all_%%usr, file-nr, cswch/s...  0.099128
985   (plist-sz, totsck, ldavg-1, all_%%usr, file-nr...  0.099147
847   (totsck, ldavg-1, all_%%usr, file-nr, cswch/s,...  0.099150
1015  (totsck, ldavg-1, pgfree/s, proc/s, all_%%usr,...  0.099153
1006  (totsck, pgfree/s, proc/s, all_%%usr, file-nr,...  0.099154
973   (plist-sz, ldavg-1, pgfree/s, all_%%usr, file-...  0.099155
865   (plist-sz, pgfree/s, all_%%usr, file-nr, cswch...  0.099157
935   (plist-sz, ldavg-1, all_%%usr, file-nr, cswch/...  0.099174
1000  (totsck, ldavg-1, proc/s, all_%%usr, file-nr, ...  0.099179
```

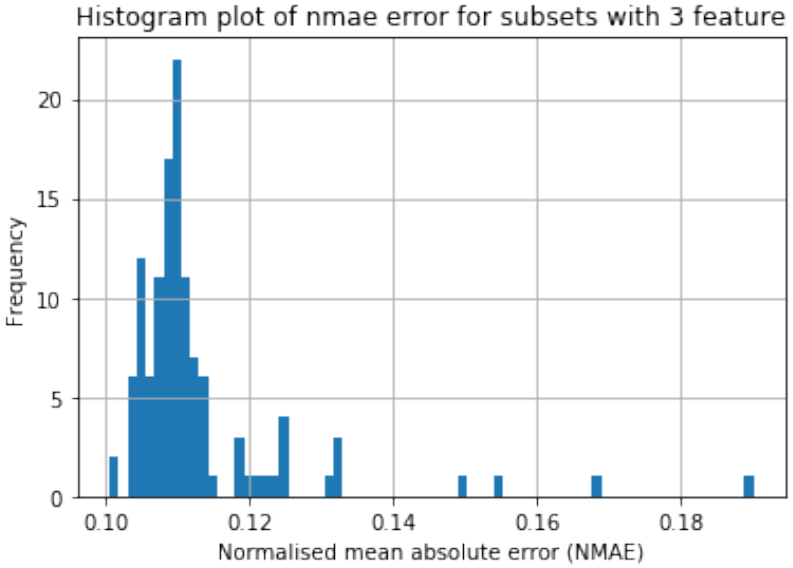
The feature list with least NMAE is the first row of this dataframe.

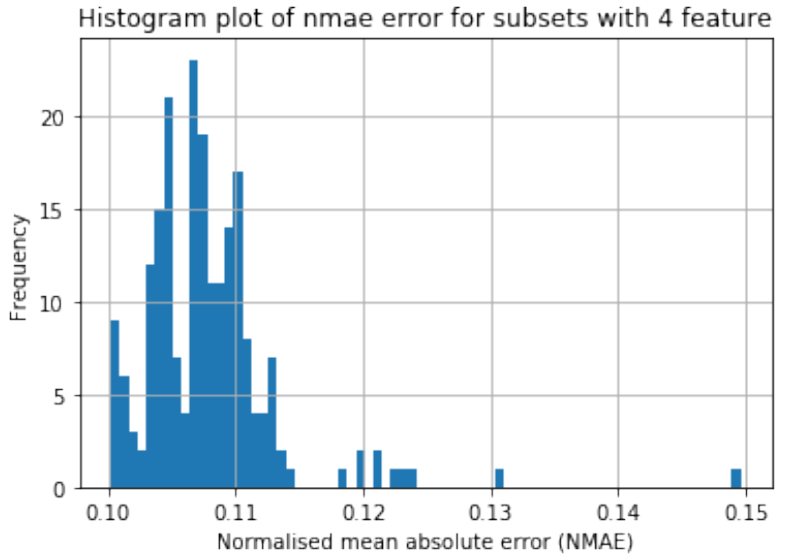
```
Feature list of the least NMAE linear regression model :
 ('totsck', 'ldavg-1', 'pgfree/s', 'all_%%usr', 'file-nr', 'cswch/s', '%
%memused', 'runq-sz')
```

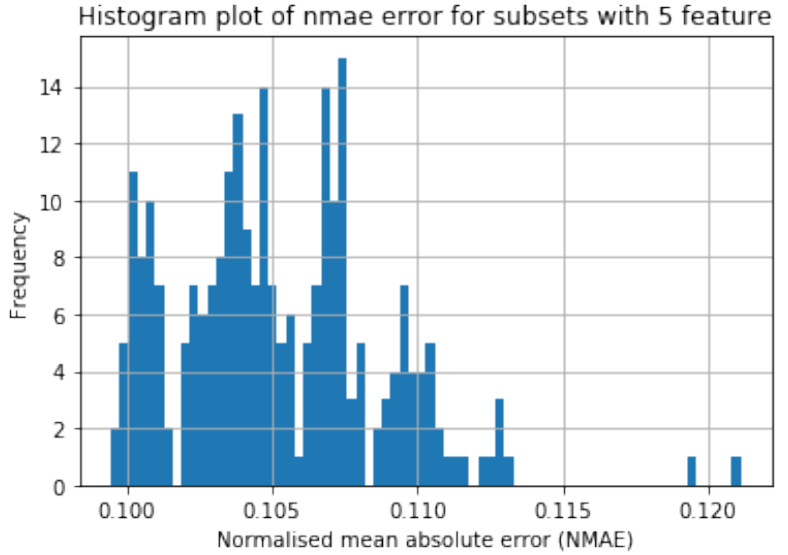
An user defined function similar to above process is used here with the exception of passing just one argument, i.e. number of features. This function calculates the NMAE of each combination of features for that particular iteration and then plots the histogram. The plot obtained below shows the frequency of unique NMAE for all combinations of the subsets possible.

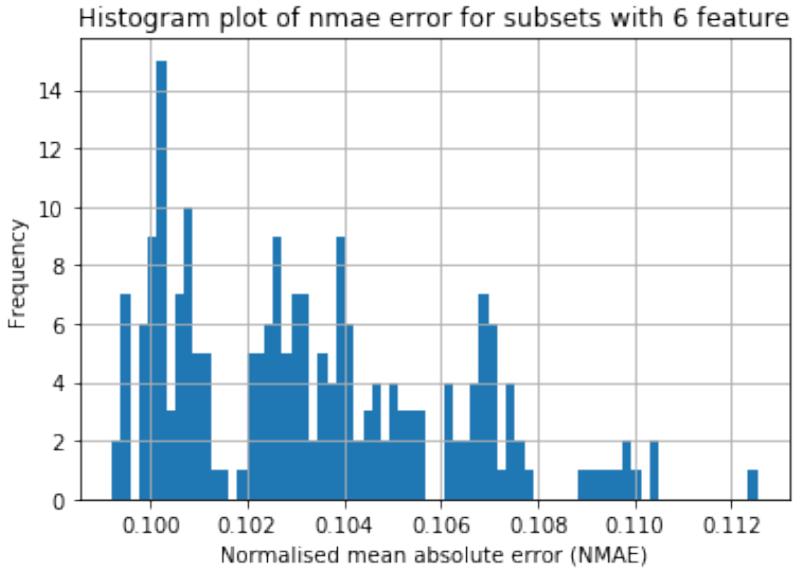

Histogram plot of nmae error for subsets with 1 feature
Frequency
Normalised mean absolute error (NMAE)

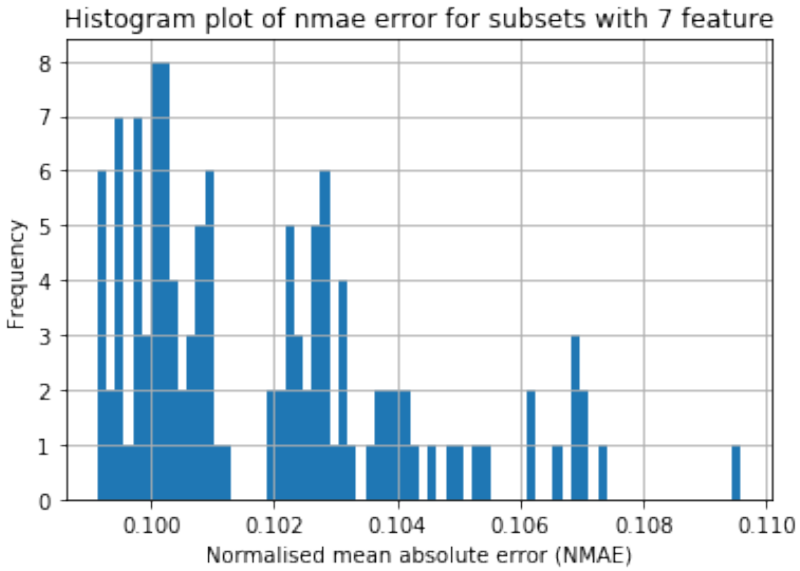

Histogram plot of nmae error for subsets with 2 feature
Frequency
Normalised mean absolute error (NMAE)

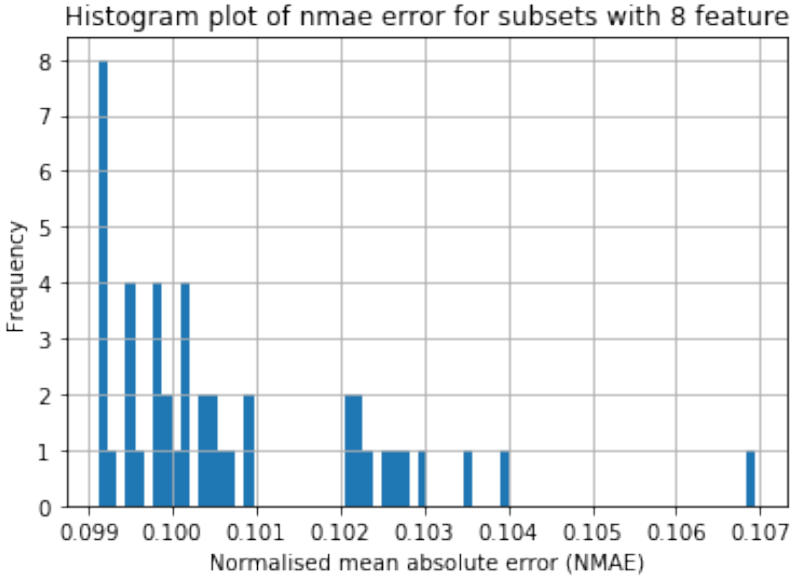

Histogram plot of nmae error for subsets with 3 feature
Frequency
Normalised mean absolute error (NMAE)

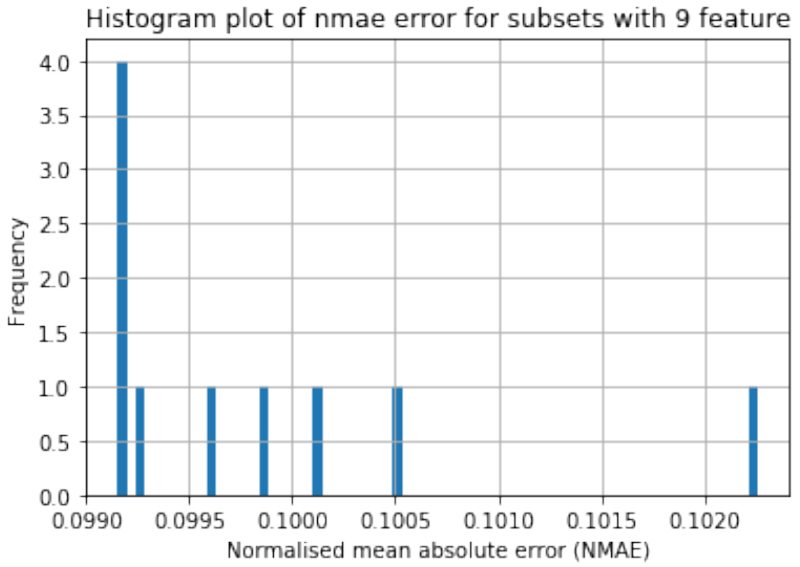

Histogram plot of nmae error for subsets with 4 feature
Frequency
0
5
10
15
20
0.10
0.11
0.12
0.13
0.14
0.15
Normalised mean absolute error (NMAE)

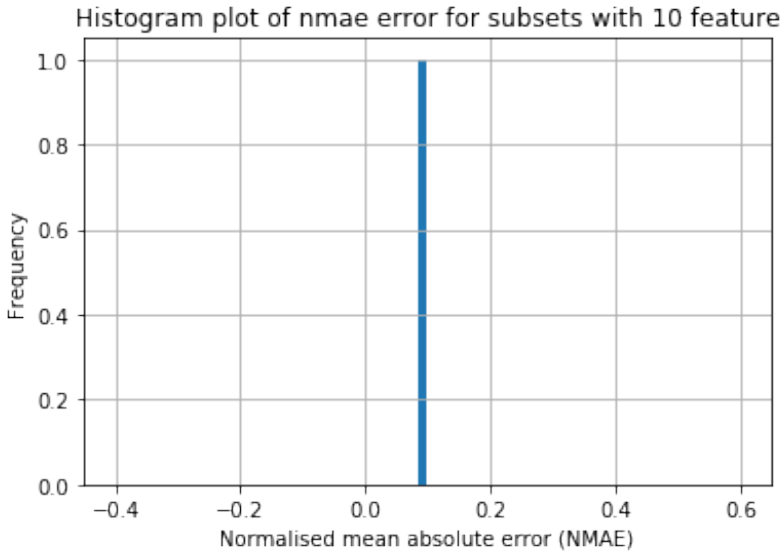

Histogram plot of nmae error for subsets with 5 feature
Frequency
0
2
4
6
8
10
12
14
0.100
0.105
0.110
0.115
0.120
Normalised mean absolute error (NMAE)


Histogram plot of nmae error for subsets with 6 feature
Frequency
0
2
4
6
8
10
12
14
0.100
0.102
0.104
0.106
0.108
0.110
0.112
Normalised mean absolute error (NMAE)

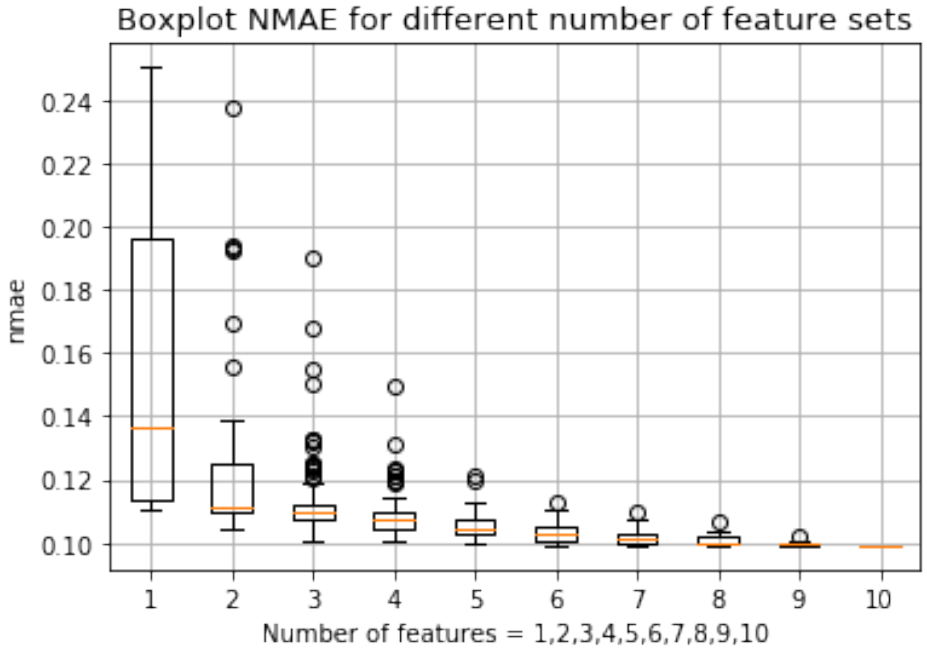

Histogram plot of nmae error for subsets with 7 feature
Frequency
0
1
2
3
4
5
6
7
8
0.100
0.102
0.104
0.106
0.108
0.110
Normalised mean absolute error (NMAE)


Histogram plot of nmae error for subsets with 8 feature
Frequency
0
1
2
3
4
5
6
7
8
0.099
0.100
0.101
0.102
0.103
0.104
0.105
0.106
0.107
Normalised mean absolute error (NMAE)


Histogram plot of nmae error for subsets with 9 feature
Frequency
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0.0990
0.0995
0.1000
0.1005
0.1010
0.1015
0.1020
Normalised mean absolute error (NMAE)



Below is the box plot of NMAE against the feature subsets(1-10). We can observe the reducing NMAE with increase in number of features considered in the subset.



3. Method 2 (Heuristic method): Linear univariate feature selection. Take each feature of X and compute the sample correlation of the feature with P the Y value on the training set.
Rank the features according to the square of the correlation values; the top feature has the highest value. Build ten feature sets composed of the top k features, k = 1, ..., 10. The first feature set contains the top feature, the second feature set contains the top two features, etc. For each feature set, compute the linear regression model on the training set and compute the error (NMAE) on the test set. Produce a plot that shows the error value in function of the set size k.

Plot a heat map of the correlation matrix whose vertical axis and horizontal axis is the same vector (x 1 , x 2 , ..., x 1 0, y).

Solution:

Firstly, I calculate the standard deviation on Y train using the std() method. Simultaneously, Y_mae calculates the relative difference between y(i) and y(mean)
For calculating the correlative error, nested ‘for’ loop is applied to calculate Standard deviation, relative mean error of X train values, as well as the correlation value of each feature in X.
The correlation squares are then fed into a list corr_squ with feature name and the correlation value. This list is sorted to rank the features is decreasing order of correlation.

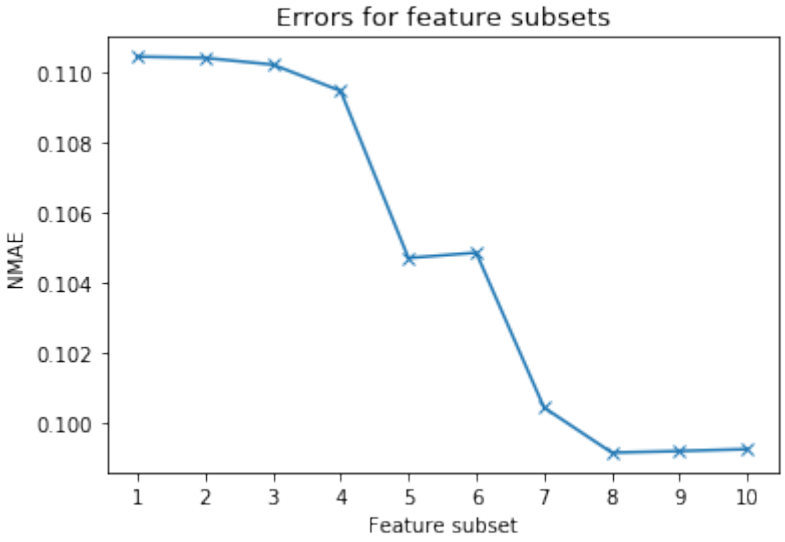
```
TASK 4.3 - Linear univariate feature selection(Heuristic method)

Feature : plist-sz
Correlation        :  -0.829575
Correlation Square :   0.688195

Feature : totsck
Correlation        :  -0.82849
Correlation Square :   0.686396

Feature : ldavg-1
Correlation        :  -0.807848
Correlation Square :   0.652619

Feature : pgfree/s
Correlation        :  0.212777
Correlation Square :   0.045274

Feature : proc/s
Correlation        :  -0.234172
Correlation Square :   0.054837

Feature : all_%%usr
Correlation        :  -0.554649
Correlation Square :   0.307636

Feature : file-nr
Correlation        :  -0.747179
Correlation Square :   0.558276

Feature : cswch/s
Correlation        :  -0.734353
Correlation Square :   0.539274

Feature : %%memused
Correlation        :  -0.534111
Correlation Square :   0.285274

Feature : runq-sz
Correlation        :  -0.799614
Correlation Square :   0.639383


Execution time (in seconds) :  1.0904359999999542
Correlation square of features in descending order
        NAME  CORRELATION SQUARE
0   plist-sz            0.688195
1     totsck            0.686396
2    ldavg-1            0.652619
```

```
9    runq-sz            0.639383
6    file-nr            0.558276
7    cswch/s            0.539274
5  all_%%usr            0.307636
8  %%memused            0.285274
4     proc/s            0.054837
3   pgfree/s            0.045274
```

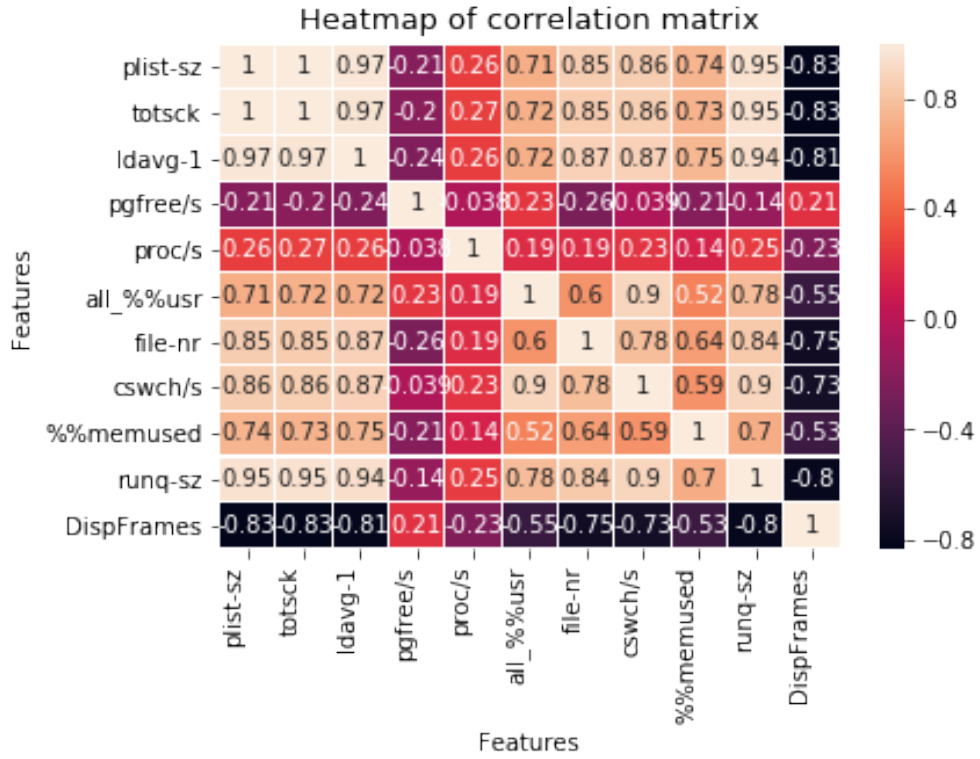
After obtaining the feature ranking, this dataframe can now be fed to a nest while() loop to calculate the NMAE with increasing number of subsets from 1 to 10. The results of the reducing NMAE can be observed below:

```
NMAE :
['plist-sz'] : 0.110445
['plist-sz', 'totsck'] : 0.110408
['plist-sz', 'totsck', 'ldavg-1'] : 0.110216
['plist-sz', 'totsck', 'ldavg-1', 'runq-sz'] : 0.109467
['plist-sz', 'totsck', 'ldavg-1', 'runq-sz', 'file-nr'] : 0.104699
['plist-sz', 'totsck', 'ldavg-1', 'runq-sz', 'file-nr', 'cswch/s'] : 0.104852
['plist-sz', 'totsck', 'ldavg-1', 'runq-sz', 'file-nr', 'cswch/s', 'all_%
%usr'] : 0.100433
['plist-sz', 'totsck', 'ldavg-1', 'runq-sz', 'file-nr', 'cswch/s', 'all_%%usr',
'%%memused'] : 0.099147
['plist-sz', 'totsck', 'ldavg-1', 'runq-sz', 'file-nr', 'cswch/s', 'all_%%usr',
'%%memused', 'proc/s'] : 0.099193
['plist-sz', 'totsck', 'ldavg-1', 'runq-sz', 'file-nr', 'cswch/s', 'all_%%usr',
'%%memused', 'proc/s', 'pgfree/s'] : 0.099247
```

The Error graph depicts the sizable drop in NMAE with increase in subsets according the ranking of correlation square.



The heat map depicts the correlation values of each feature against the other.

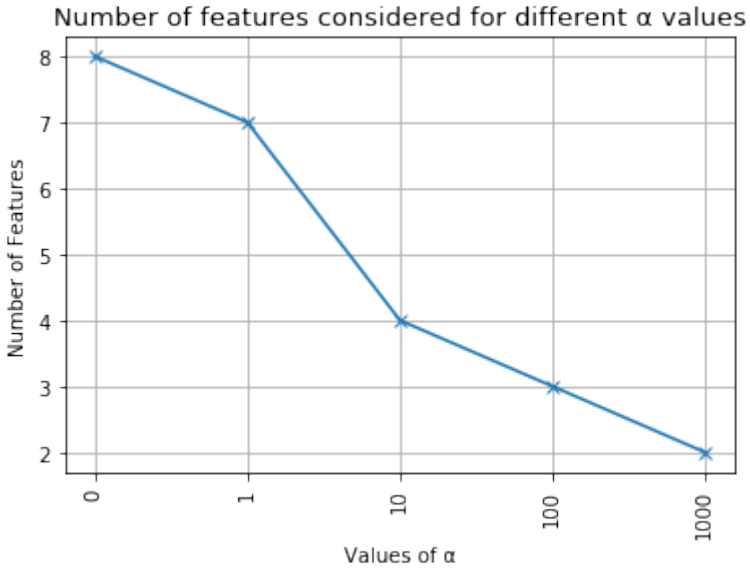


4. Method 3 (Optional): Linear regression with L1 Regularization (Lasso).
Compute and evaluate models for different values of α (α = 10 −1 , ..., 10 + 3). As a result of this evaluation, draw a graph with α on the horizontal axis (use logarithmic scale) and the number of features with value not zero (#features != 0) on the vertical axis. Also, draw a table with two columns: the first column is #features != 0 and the second column is the NMAE of the model on the test set.

Solution:
An user defined function calculates the NMAE for different values of α.
features_used and NMAE calculated from the call_lasso() method are added to the list and then sorted with descending order.

Result:

```
TASK 4.4 - Linear regression with L1 regularization
   #features!=0      NMAE
0             8  0.096843
1             7  0.097066
2             4  0.099828
3             3  0.104064
4             2  0.122040
```



5. Compare these three methods against each other and explain the advantages and disadvantages of each method in terms of required computing time and achieved accuracy. Which method produces a small feature set whose model has an error very close to the smallest possible error?

Solution:

Execution time:

```
Optimal Method (in seconds)   :  3.7993

Heuristic Method (in seconds) :  1.0904

Lasso Method (in seconds)     :  0.0323
```

Accuracy achieved:

```
Optimal Method   :  0.099119

Heuristic Method :  0.099147

Lasso Method     :  0.098928
```

Lasso method takes the least amount of execution time with most accuracy. This is due to the fact that the cost function of Lasso method includes the control parameter which helps to reduce the number of features considered for NMAE when the |θ i | becomes zero.
Optimal method requires iteration through all combination of features subsets to evaluate the least NMAE. Although it achieves comparable accuracy as Lasso method, it consumes large amount of resources and has high complexity wrt space and time.

Heuristic method significantly achieves better time performance than Optimal method as it calculates the correlation factor of various features initially and ranks the features. By ranking the features first and thereafter creating subsets allows quick execution of algorithm. But it is not the most efficient method to reduce the feature sets as we can observe from the findings that heuristic model of 10 features achieves less accuracy than optimal method with 8 features.

With lasso, execution time is 35 times less than Heuristic. Although it considers 9 features instead of 8 as in Optimal method, Lasso surpasses the other two methods if we wish obtain the most efficient model in terms of execution time and achieving accuracy.

Task 5

1. Use a random forest regressor to train a model M on the training set. As above, compute the prediction error (NMAE) on the test set.

Solution: Sklearn's ensemble library contains RandomForestRegressor that allows to perform prediction error as performed similarly for linear regression.

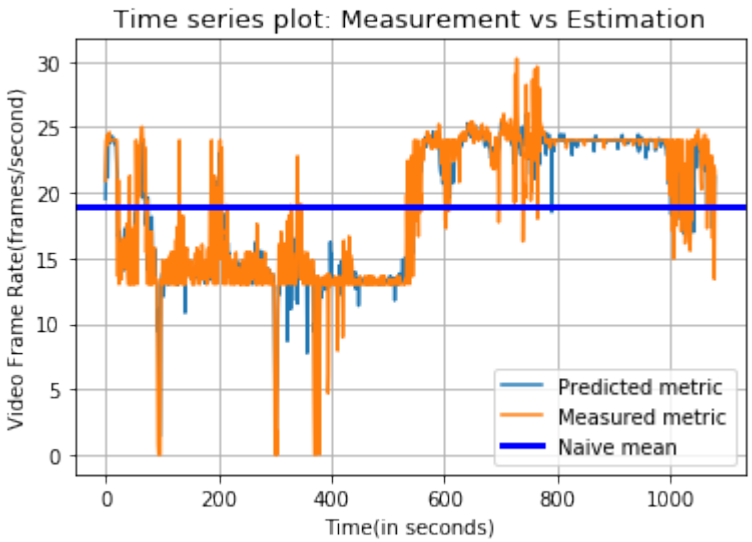
Results:

```
Mean Absolute error: 1.20
Mean of Y test set observations :  18.90341
Normalised Mean Absolute Error :  0.06368
```

2. Produce a time-series plot that shows both the measurements and the model estimation for M for the Video Frame Rate values on the test set. Show also the prediction of the naïve method on the same plot. (See an example of such a plot in Figure 4(a) of [1]).

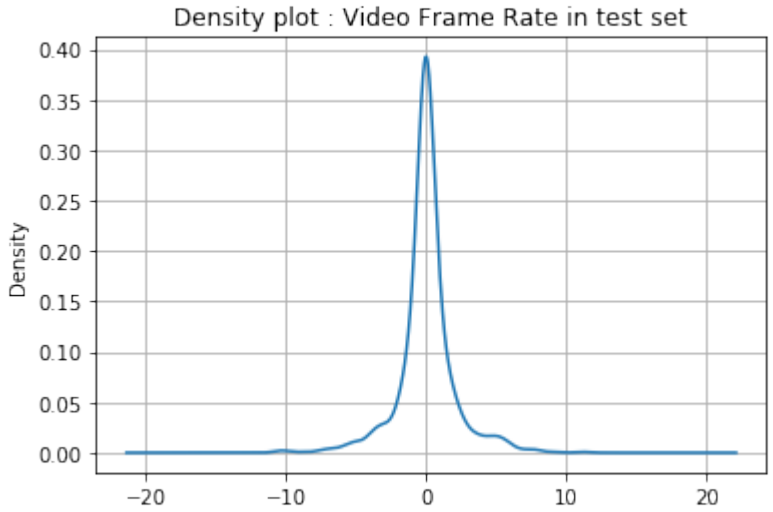
Solution:
Naive method is calculated on the Y_train set.
The plot() method is used to plot Y_test, Y_predict and naive prediction. For this the datasets are sorted according to the timestamp.

```
NAIVE METHOD RESULTS:
Mean Absolute error: 4.99
Mean of Y train set observations :  18.87608
Normalised Mean Absolute Error for Naive method :  0.26448
```



3. Produce a density plot for the prediction error $y_i - \hat{y}_i$ on the test set.

Solution:



4. Perform feature selection using the feature importances attribute of the random-forest model from sklearn. This gives a ranking of k features with respect to M on the training set, for k = 1, ..1, 0. For each k, create a random-forest model models with the top k features. Compare the results with those of the three methods you evaluated in Task IV.

Solution:
feature_improtances_ attribute allows to rank the features. This ranking is then combined with another list which contains the feature list. Then it is sorted to display as below:

	Features	Importance
1	totsck	0.400498
0	plist-sz	0.381976
2	ldavg-1	0.058069
8	%%memused	0.054796
9	runq-sz	0.024069
3	pgfree/s	0.018951
7	cswch/s	0.017981
6	file-nr	0.017823
5	all_%%usr	0.013533
4	proc/s	0.012304

This ranking data-frame is then fed to the user-defined method which calculates the feature-subsets according the reducing importance factor. Individual subsets with features(k=1 to 10) are used to model different RandomForestRegression models and subsequently calculate NMAE for each subset as shown below:

```
NMAE
['totsck'] : 0.092814
['totsck', 'plist-sz'] : 0.09164
['totsck', 'plist-sz', 'ldavg-1'] : 0.066203
['totsck', 'plist-sz', 'ldavg-1', '%%memused'] : 0.064236
['totsck', 'plist-sz', 'ldavg-1', '%%memused', 'runq-sz'] : 0.066151
['totsck', 'plist-sz', 'ldavg-1', '%%memused', 'runq-sz', 'pgfree/s'] : 0.065559
['totsck', 'plist-sz', 'ldavg-1', '%%memused', 'runq-sz', 'pgfree/s', 'cswch/s']
: 0.065911
['totsck', 'plist-sz', 'ldavg-1', '%%memused', 'runq-sz', 'pgfree/s', 'cswch/s',
'file-nr'] : 0.064772
['totsck', 'plist-sz', 'ldavg-1', '%%memused', 'runq-sz', 'pgfree/s', 'cswch/s',
'file-nr', 'all_%%usr'] : 0.066653
['totsck', 'plist-sz', 'ldavg-1', '%%memused', 'runq-sz', 'pgfree/s', 'cswch/s',
'file-nr', 'all_%%usr', 'proc/s'] : 0.065051
```

With 10 feature subset, it provides an error estimation of 6.5% which is less than the least error estimation among the previously discussed methods
It is important to note that this method produces more accurate result with only one feature than any of the previous discussed methods namely Optimal, Heuristic and Lasso.

5. Considering the full features set (k = 10), compute the error (NMAE) for the training set and for the test set. The errors will be different. Try to explain this difference using the concepts of bias and variance in model prediction. Compute the error for linear regression for the same two data sets. Compare random forest with linear regression with respect to bias and variance in prediction.

Solution:
Y_train is used to train the model for calculating NMAE for the training set as well as test set in order to obtain training error as well as cross-validation error.
For calculating NMAE on test set, the predict() is used with X_test argument.
For calculating NMAE on training set, the predict() is used with X_train argument.
MAE on test set is calculated with y_test and y_predict
MAE on training set is calculated with y_train and y_predict.
This gives the following NMAE:

Results:
```
Random Forest
NMAE on test set :  0.06481
NMAE on training set :  0.02546
Linear Regression
NMAE on test set:  0.10284
NMAE on training set :  0.10228
```

Here we observe that the NMAE for linear regression on test set and training set are almost equal. Moreover, the error in training set is high(10%). We can hence infer that this is the BIAS problem.

For Random Forest, NMAE for test set is very large than training set. The estimation error in training set is really low(2%). This tells us that the model is overfitting, hence, a VARIANCE problem.

6. Based on the results of this task, discuss the accuracy of estimating the Video Frame Rate. Compare Random Forest with the methods used in previous tasks.

Solution:

Although the first impression of Random Forest depicts that it achieves very high accuracy, it has been found that the model is overfitting. That is why it was able to predict with better accuracy with only 1 feature, whereas, other methods like Optimal, Heuristic and Lasso require at least 8 features to predict with the similar level of accuracy.

Execution time for RandomForest is close to 850 milliseconds which is more than Lasso but less than heuristic and optimal.
Since it lacks on its scale of accuracy by overfitting the graph, it is wise to not use Random Forest for depicting Video Frame Rates with the given set of feature list. However, Random Forest might prove to be accurate and time saving when the sample size is significantly huge.