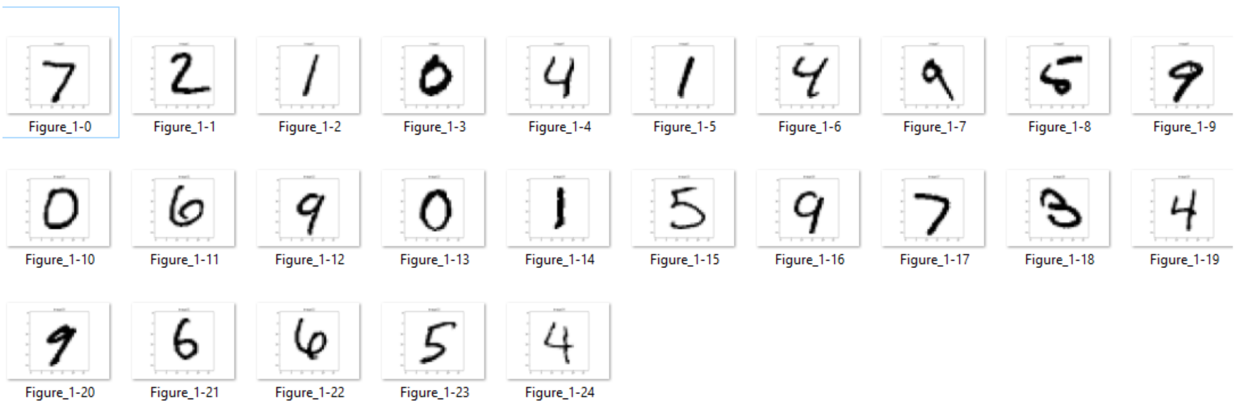
Homework 1

Question: 1

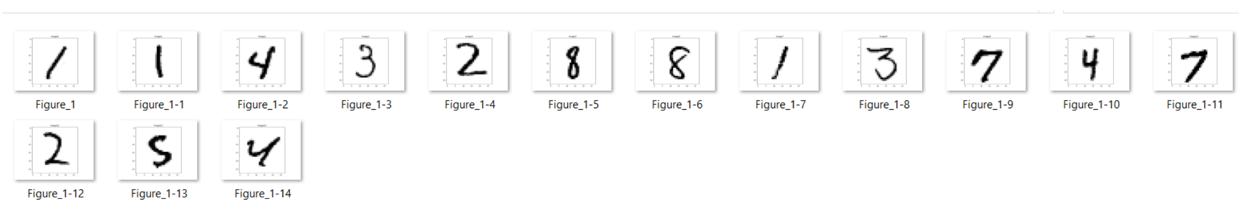
b)

i:



As we can clearly see that all images of 9 are not alike, some are comparatively straight while others are leaning either towards left or right.

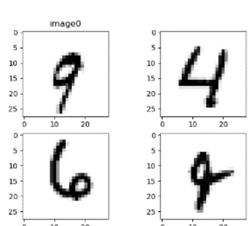
ii:



None of my guesses were wrong but I had a hard time guessing figure_1-13.

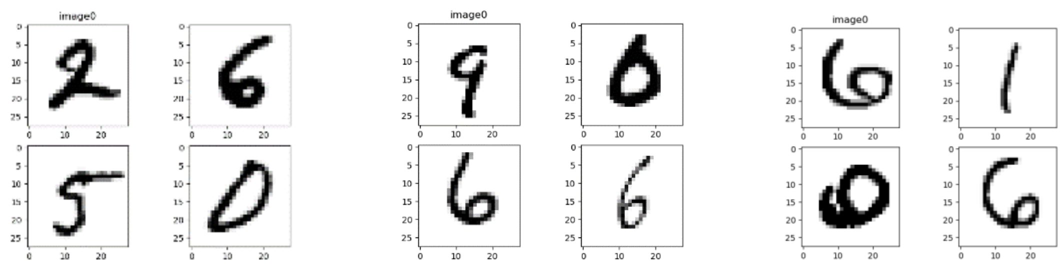
iii:

Two different digits that look alike:



4 looks like 9 in this sample.

Three samples of the same digit that don’t look alike:



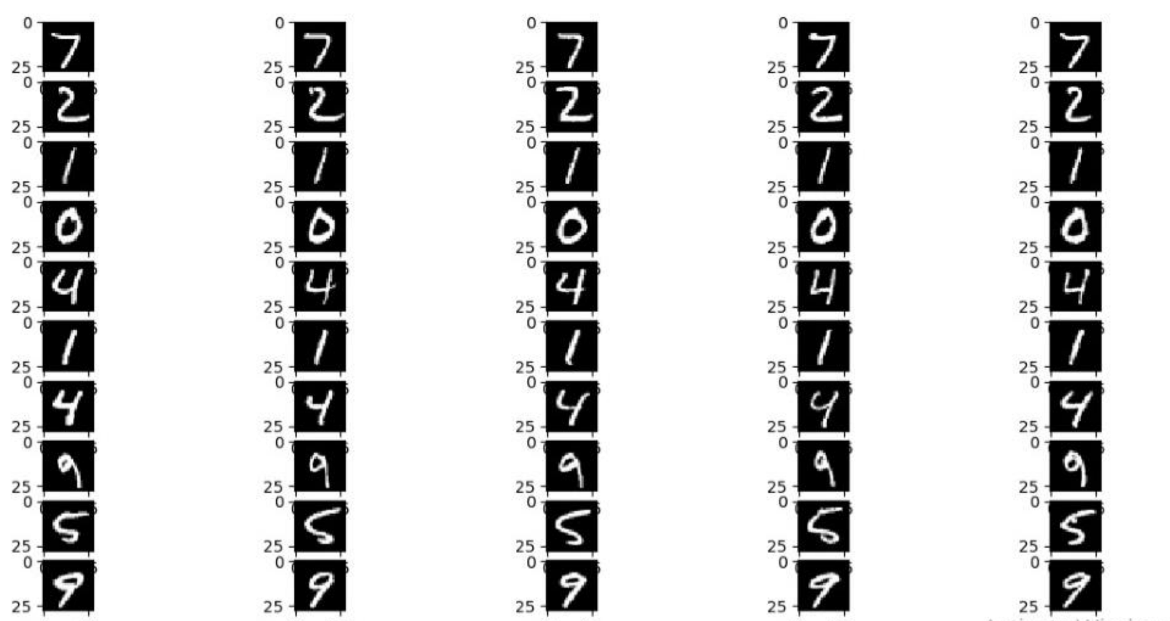
Zeros in these images look different.

c)

i:

Predictions are made for a new instance by searching through the entire training set for the K most similar instances which are called the neighbors.

ii:



Metric in KNNClassifier = Euclidean

Code:

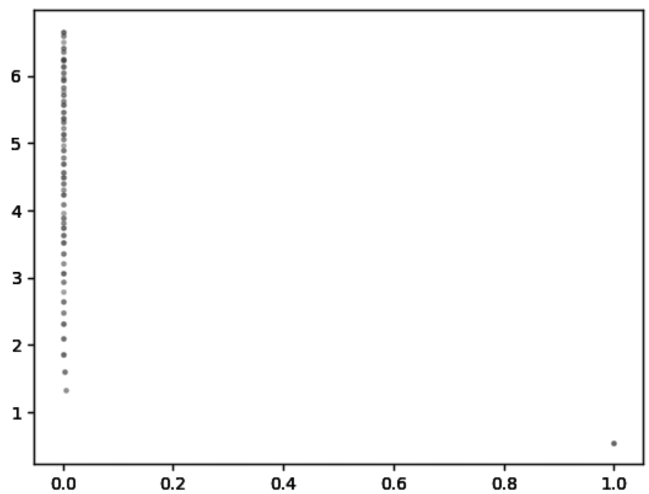
```
knn_classifier=KNeighborsClassifier(n_neighbors=5,metric='euclidean',algorithm='ball_tree')
```

```
knn_classifier.fit(x_train,y_train.ravel())

image_indices = []


for test_index in range(0,10):
    distances, indices = knn_classifier.kneighbors(x_test[test_index].reshape(1,-1))
    for test_data in indices:
        for index in test_data:
            image_indices.append(index)

image_plot_index = 0
for image_index in image_indices:
    image = x_train[image_index,:].reshape(28,28)
    image_plot_index = image_plot_index + 1
    plot.subplot(10, 5, image_plot_index)
    plot.imshow(image, cmap=plot.cm.gray)

plot.show()
```

iii:

```
import numpy as np
import mnist_loader_KNN as DS_loader
import matplotlib.pyplot as plt
import random
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from collections import defaultdict
import seaborn as sns
from struct import unpack

url_train_image = 'http://yann.lecun.com/exdb/mnist/train-images-idx3-
ubyte.gz'
url_train_labels = 'http://yann.lecun.com/exdb/mnist/train-labels-idx1-
ubyte.gz'
num_train_samples = 60000

x_train = DS_loader.try_download_x(url_train_image, url_train_labels,
num_train_samples)
y_train = DS_loader.try_download_y(url_train_image, url_train_labels,
num_train_samples)

url_test = 'http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz'
url_test_labels = 'http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz'
num_test_samples = 10000

temp=random.randrange(9999)

x_test = DS_loader.try_download_x(url_test_image, url_test_labels,
num_test_samples)
y_test = DS_loader.try_download_y(url_test_image, url_test_labels,
num_test_samples)

sample=x_test[temp,:].reshape(28,28)

prime_predictions = []
prime_knnclassifier =
KNeighborsClassifier(n_neighbors=1,algorithm='ball_tree')
index_error=1;
```

```
error_score=[]
inc = 1
while(inc <= 10001):
    knnclassifier=KNeighborsClassifier(n_neighbors=inc,algorithm='ball_tree')
    knnclassifier.fit(x_train,y_train.ravel())
    predictions= knnclassifier.predict(x_test)

    score=knnclassifier.score(x_test,y_test)
    error = metrics.mean_squared_error(y_test,predictions)
    error_score.append(error)
    plt.scatter(1/inc, error, 5, alpha=0.2)
    inc = inc + 200

plt.show()
```



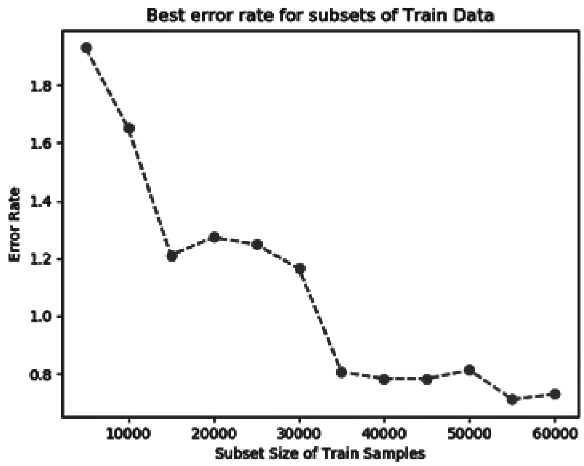
iv:

Using K = 1 or 1.2 gives the best result. So, I have used K = 1 in my classifier for this problem. For training, I have used increments of 5000 images.

Conclusion:

With the increase in training data, error reduces. Best error calculated is 0.7 when train sample is close to 55000.

Plot of best error rates on the dataset:



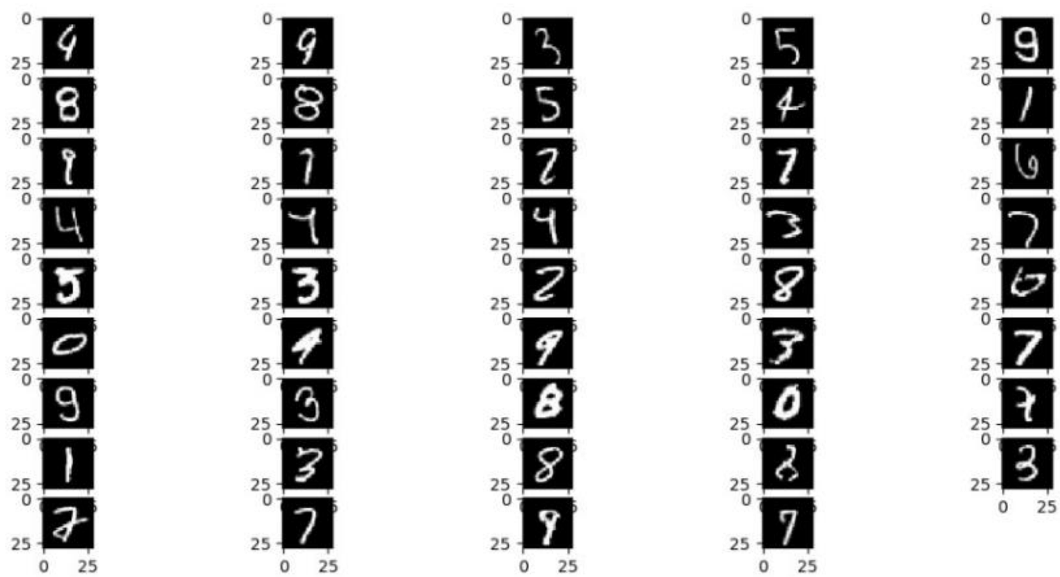
Code:

```
error_score=[]
index_error=1

knn_classifier=KNeighborsClassifier(n_neighbors=1,algorithm='ball_tree')


y_test_subset = y_test[:, :]
x_test_subset = x_test[:, :]



plot_x=[]



for index in range(1, 13):

    x_temp = x_train[:(index * 5000) + 1, :]

    y_temp = y_train[:(index * 5000) + 1, :]


    knn_classifier.fit(x_temp, y_temp.ravel())

    predictions = knn_classifier.predict(x_test_subset)

    score = knn_classifier.score(x_test_subset, y_test_subset)
    print(score)

    error = metrics.mean_squared_error(y_test_subset,predictions,
multioutput='raw_values')


    print(error)

    error_score.append(error)
```

```
    plot_x.append(((i-1)*5000)+5000)


plt.plot(plot_x, error_score, marker='o', linestyle='--', color='r', label='Best Error Rate')

plt.xlabel('Subset Size of Train Samples')

plt.ylabel('Error Rate')

plt.title('Best Error Rate')

plt.show()
```

d)

K nearest neighbors of some misclassified images:



In this image, first image is the expected output and the second is the prediction. For example, 4 was the expected output but prediction was 9, 3 was expected and 5 was the prediction and so on.

Code:

```
x_test_subset = x_test[:, :]


y_test_subset = y_test[:, :]


plot_x=[]
```

```
x_temp = x_train[:(15000)+1, :]

y_temp = y_train[:(15000)+1, :]
knnclassifier.fit(x_temp, y_temp.ravel())
predictions = knnclassifier.predict(x_test_subset)

prime_predictions = predictions

prime_knnclassifier = knnclassifier
miss_index = 0
misclassifiedIndexes = []
for label, predict in zip(y_test, prime_predictions):
    if label != predict:
        misclassifiedIndexes.append(miss_index)
    miss_index += 1

image_indices = []
for i in misclassifiedIndexes:
    distances, indices = prime_knnclassifier.kneighbors(x_test[i].reshape(1,-1))
    for test_data in indices:
        for index in test_data:
            image_indices.append(index)

plt.figure(figsize=(10,20))
img_plot_index = 0
for img_index in image_indices:
    img = x_train[img_index,:].reshape(28,28)
    img_plot_index = img_plot_index+1
    plt.subplot(10, 5, img_plot_index)
    plt.imshow(img,cmap=plt.cm.gray)
plt.show()
```

e)

i:

Minkowski Distance:

A. Manhattan with p = 1

```
knnclassifier=KNeighborsClassifier(n_neighbors=1,metric='minkowski',p=1,algorithm='bal
l_tree')
```

Accuracy: 0.9283

Error: 1.56666667

B. $\log_{10}(p)$

```
knnclassifier=KNeighborsClassifier(n_neighbors=1,metric='minkowski',p=log(.1),algorith
m='ball_tree')
knnclassifier=KNeighborsClassifier(n_neighbors=1,metric='minkowski',p=log(1.0),algorit
hm='ball_tree')
```

C. Chebyshev

```
knnclassifier=KNeighborsClassifier(n_neighbors=1,metric='chebyshev',algorithm='ball_tr
ee')
```

```
Accuracy: 0.7266666666666667
Error observed: 3.93833333
```

```
Results:
```

Metric	Minkowski, p=log(0.4)	Minkowski, p=log(0.2)	Minkowski, p=log(0.3)	Minkowski, p=log(0.5)	Minkowski, p=log(0.1)	Minkowski, p=log(0.6)
Error	8.59333333	0.89333333	8.59333333	8.59333333	0.745	8.59333333

Minkowski, p=log(0.8)	Minkowski, p=log(0.7)	Minkowski, p=log(0.9)
8.59333333	8.59333333	8.59333333

g)

Best error in *Minkowski* metric with $p=\log_{10}(0.1) =$ 0.745

Question: 2

b)

i:

Rows: 517

Columns: 13

Rows = Data points

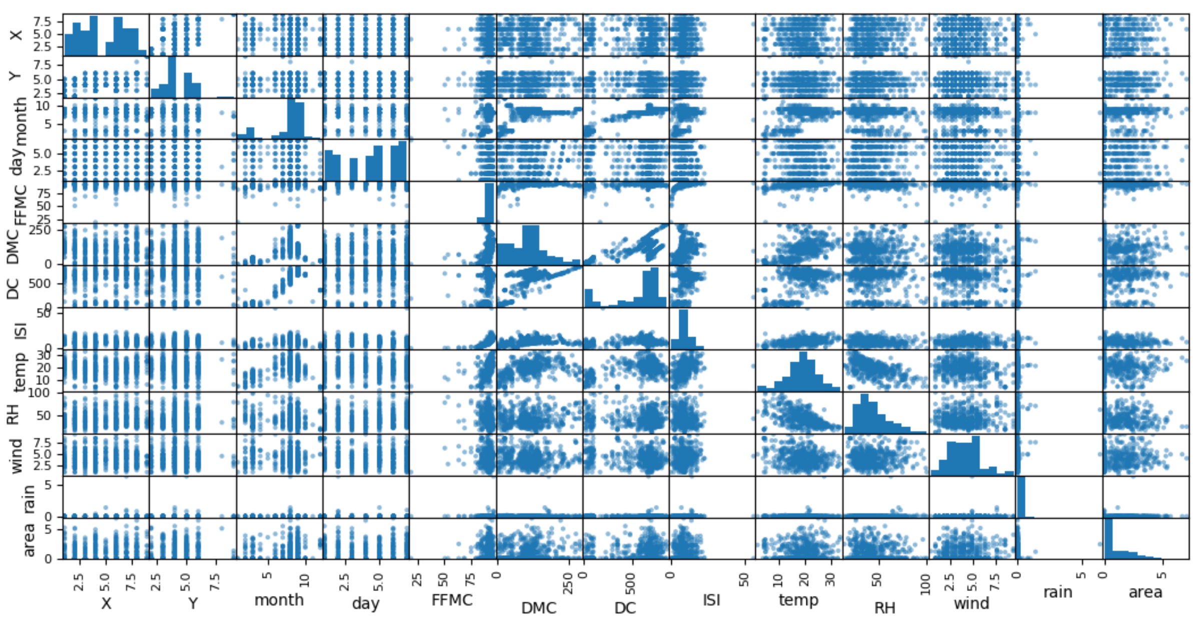
Columns = Predictors

ii:

Linear model has continuous data values whereas logistic has categorical data.

In my output, area is very skewed towards 0.0, thus logarithmic transformation is used.

iii: and iv:



Area does not change much depending upon predictors like FFMC and rain as it is accumulated only through one chunk of the plot.

Predictors like temp, RH, DMC and wind appear to be significant contributors. We shall observe new findings in further exercises.

Temp and RH also seem to have a good even distribution, it is far-fetched to say normal distribution just yet, but they provide signs of being co-related.

Code:

```
dataframe=pandas.read_csv("Forest_Fire/forestfires.csv")

# Encode Data

dataframe.month.replace(('jan','feb','mar','apr','may','jun','jul','aug','sep','oct','
nov','dec'),(1,2,3,4,5,6,7,8,9,10,11,12), inplace=True)


dataframe.day.replace(('mon','tue','wed','thu','fri','sat','sun'),(1,2,3,4,5,6,7),
inplace=True)


dataframe['area'] = np.log(dataframe['area']+1)


scatter_matrix(dataframe)
plt.show()
```

v:

Printing median	
X	4.00000
Y	4.00000
month	8.00000
day	5.00000
FFMC	91.60000
DMC	108.30000
DC	664.20000
ISI	8.40000
temp	19.30000
RH	42.00000
wind	4.00000
rain	0.00000
area	0.41871

Printing Range	
X	8.00000
Y	7.00000
month	11.00000
day	6.00000
FFMC	77.50000
DMC	290.20000
DC	852.70000
ISI	56.10000
temp	31.10000
RH	85.00000
wind	9.00000
rain	6.40000
area	6.99562

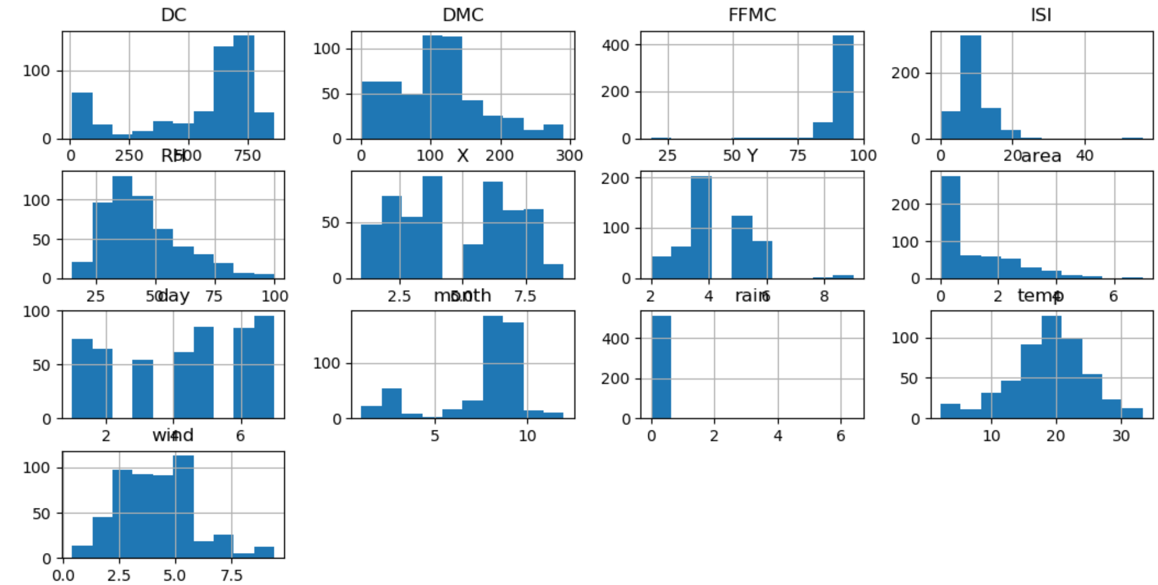
Printing mean	
X	4.669246
Y	4.299807
month	7.475822
day	4.259188
FFMC	90.644681
DMC	110.872340
DC	547.940039
ISI	9.021663
temp	18.889168
RH	44.288201
wind	4.017602
rain	0.021663
area	1.111026

1st quartile		3rd quartile		Inter-quartile Ranges	
X	3.0	X	7.000000	X	4.000000
Y	4.0	Y	5.000000	Y	1.000000
month	7.0	month	9.000000	month	2.000000
day	2.0	day	6.000000	day	4.000000
FFMC	90.2	FFMC	92.900000	FFMC	2.700000
DMC	68.6	DMC	142.400000	DMC	73.800000
DC	437.7	DC	713.900000	DC	276.200000
ISI	6.5	ISI	10.800000	ISI	4.300000
temp	15.5	temp	22.800000	temp	7.300000
RH	33.0	RH	53.000000	RH	20.000000
wind	2.7	wind	4.900000	wind	2.200000
rain	0.0	rain	0.000000	rain	0.000000
area	0.0	area	2.024193	area	2.024193

Code:

```
print('Median')

print(dataframe.median())

print('Range')

print(dataframe.max()-dataframe.min())


print('Mean')

print(dataframe.mean())

print('1st quantile')
print(dataframe.quantile(q=0.25, axis=0, numeric_only=True, interpolation='linear'))

print('3rd quantile')
print(dataframe.quantile(q=0.75, axis=0, numeric_only=True, interpolation='linear'))


print('Inter-quartile Ranges')

print(dataframe.quantile(q=0.75, axis=0, numeric_only=True, interpolation='linear')-
dataframe.quantile(q=0.25,axis=0,numeric_only=True,interpolation='linear'))
```

c)



Temp, RH are statistically significant while rain is not because temp has a near Gaussian distribution and RH also has close to Gaussian distribution.

Regression Results:

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.3147	0.178	7.387	0.000	0.965	1.664
I(**RH**)	-0.0046	0.004	-1.220	0.223	-0.012	0.003

----

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.1086	0.062	17.965	0.000	0.987	1.230
I(**rain**)	0.1101	0.208	0.529	0.597	-0.299	0.519

----

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.8677	0.209	4.144	0.000	0.456	1.279
I(**temp**)	0.0129	0.011	1.216	0.225	-0.008	0.034

ISI, rain and day do not have significant importance. *RH, wind, X, temp, DMC, DC, month* have significance.

Code:

```
plt.hist((dataframe.area))
```

```
dataframe.hist()
```

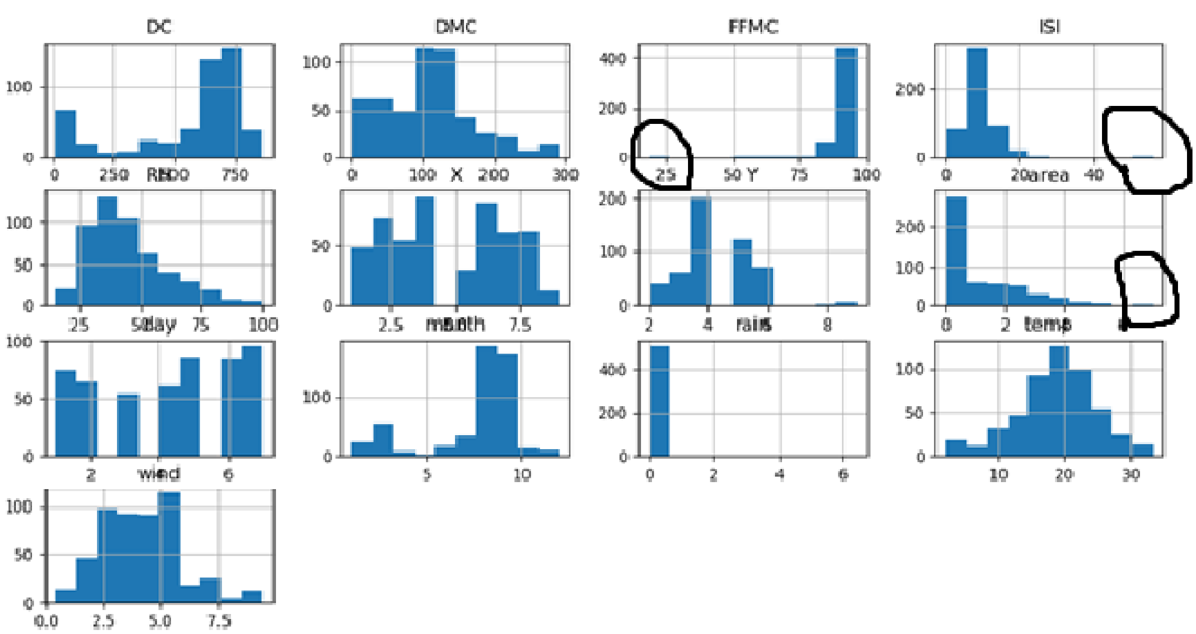
and for the p-values:

```
import numpy as np

import pandas

import copy

import statsmodels.formula.api as smf




# fix random seed for reproducibility

seed = 7

np.random.seed(seed)


# load the dataset

dataframe = pandas.read_csv("Forest_Fire/forestfires.csv")

data=copy.copy(dataframe)

# Encode Data

dataframe.month.replace(('jan','feb','mar','apr','may','jun','jul','aug','sep','oct','
nov','dec'),(1,2,3,4,5,6,7,8,9,10,11,12), inplace=True)

dataframe.day.replace(('mon','tue','wed','thu','fri','sat','sun'),(1,2,3,4,5,6,7),
inplace=True)

dataframe['area'] = np.log(dataframe['area']+1)



mod = smf.ols(formula='area~ I(ISI)', data=dataframe)
```

```
res = mod.fit()
print(res.summary())

mod = smf.ols(formula='area~ I(RH)', data=dataframe)
res = mod.fit()
print(res.summary())

mod = smf.ols(formula='area~ I(rain)', data=dataframe)
res = mod.fit()
print(res.summary())

mod = smf.ols(formula='area~ I(day)', data=dataframe)
res = mod.fit()
print(res.summary())

mod = smf.ols(formula='area~ I(month)', data=dataframe)
res = mod.fit()
print(res.summary())

mod = smf.ols(formula='area~ I(Y)', data=dataframe)
res = mod.fit()
print(res.summary())

mod = smf.ols(formula='area~ I(X)', data=dataframe)
res = mod.fit()
print(res.summary())

mod = smf.ols(formula='area~ I(wind)', data=dataframe)
res = mod.fit()
print(res.summary())

mod = smf.ols(formula='area~ I(temp)', data=dataframe)
```

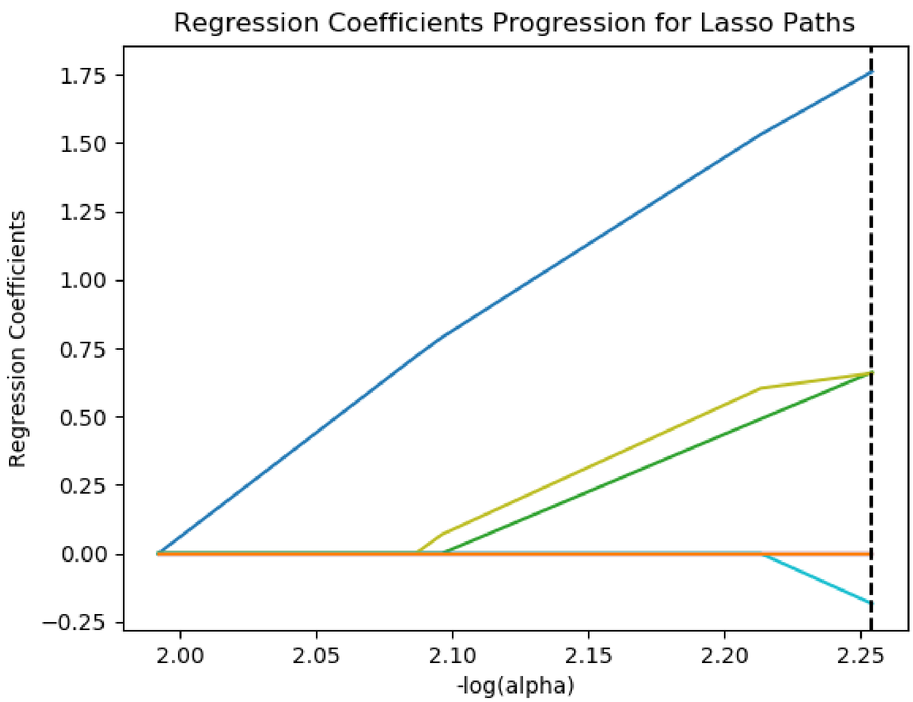
```
res = mod.fit()
print(res.summary())

mod = smf.ols(formula='area~ I(DMC)', data=dataframe)
res = mod.fit()
print(res.summary())

mod = smf.ols(formula='area~ I(DC)', data=dataframe)
res = mod.fit()
print(res.summary())

mod = smf.ols(formula='area~ I(FFMC)', data=dataframe)
res = mod.fit()
print(res.summary())
```

Outliers:



2(d)

Multi-variate linear regression:



Code:

```
my_labels=list(dataframe[0:13])

labels=my_labels[0:12]


pred_train, pred_test, tar_train, tar_test = train_test_split(X, Y,

                                                              test_size=.3,
random_state=123)



multi_model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)



print(dict(zip(labels[:12], multi_model.coef_)))
```

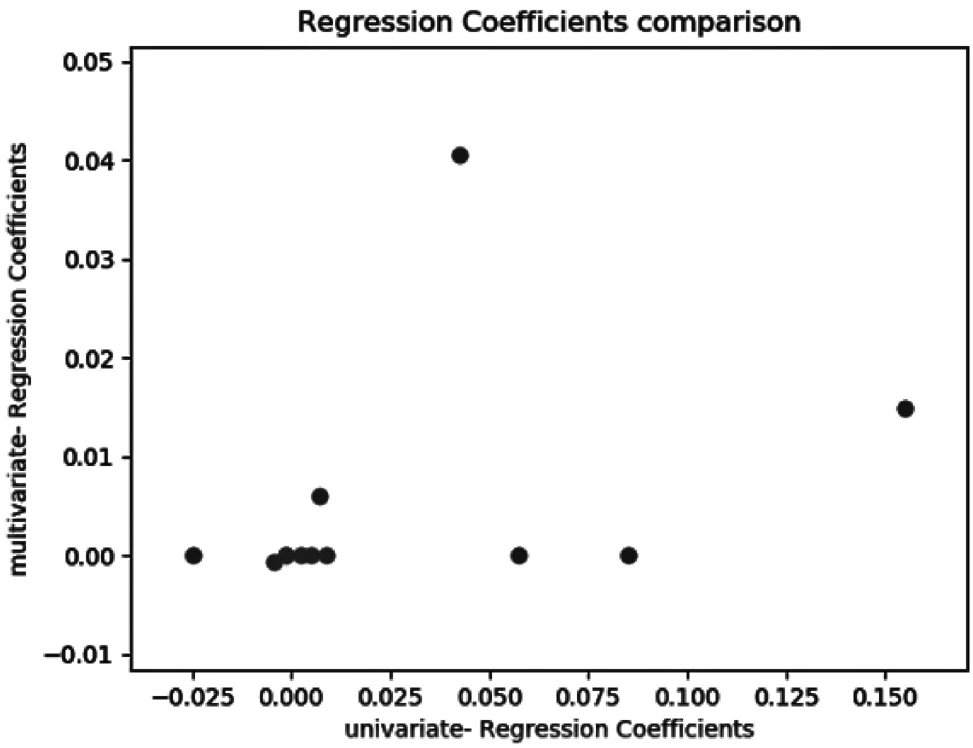
```
# plot coefficient progression

m_log_alphas = -np.log10(multi_model.alphas_)
-

ax = plt.gca()

plt.plot(m_log_alphas, multi_model.coef_path_.T)

plt.axvline(-np.log10(multi_model.alpha_), linestyle='--', color='k',

            label='alpha CV')

print(multi_model.coef_)

print(multi_model.coef_.shape)

print('Error in multivariate ')

plt.ylabel('Regression Coefficients')

plt.xlabel('-log(alpha)')

plt.title('Regression Coefficients Progression for Lasso Paths')

plt.show()
```

We can reject DC, wind and DMC because they are significant to the response. The coefficients are 0 for these but as in the plot in part 2(c), they contribute to the response.

Regression Coefficients:

```
{'X': 0.040603750619177384, 'Y': 0.0, 'month': 0.015004361282338776, 'day': 0.0,
'FFMC': 0.0, 'DMC': 0.0, 'DC': 0.0, 'ISI': 0.0, 'temp': 0.006010421743721822, 'RH': -
0.0005933952987996531, 'wind': 0.0, 'rain': 0.0}
```

e)



Code:

```
plt.scatter(coef_uni_vaale,multi_model.coef_
plt.ylabel('multivariate- Regression Coefficients')
plt.xlabel('univariate- Regression Coefficients')
plt.title('Regression Coefficients comparison')
plt.show()
```

f)

X, month and temp are significant

Non-linear model on all the predictors:

X:

```
C:\Users\Shanu\AppData\Roaming\Python\Python36\site-packages\sklearn\cross_validation.py:41: DeprecationWarning:
  "This module will be removed in 0.20.", DeprecationWarning)
                            OLS Regression Results
==============================================================================
Dep. Variable:                   area   R-squared:                       0.005
Model:                            OLS   Adj. R-squared:                  0.003
Method:                 Least Squares   F-statistic:                     2.653
Date:                Sat, 03 Feb 2018   Prob (F-statistic):              0.104
Time:                        11:11:55   Log-Likelihood:                -2879.1
No. Observations:                 517   AIC:                             5762.
Df Residuals:                     515   BIC:                             5771.
Df Model:                           1
Covariance Type:            nonrobust
========================================================================================
                           coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------
Intercept                8.4054      3.905      2.152      0.032       0.733      16.077
I(X + X ** 2 + X ** 3)   0.0213      0.013      1.629      0.104      -0.004       0.047
==============================================================================
Omnibus:                      982.556   Durbin-Watson:                   1.656
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           809290.384
Skew:                          12.774   Prob(JB):                         0.00
Kurtosis:                     195.135   Cond. No.                         418.
==============================================================================
```

DAY

```
C:\Users\Shanu\AppData\Roaming\Python\Python36\site-packages\sklearn\cross_validation.py:41: DeprecationWarning:
  "This module will be removed in 0.20.", DeprecationWarning)
                            OLS Regression Results
==============================================================================
Dep. Variable:                   area   R-squared:                       0.005
Model:                            OLS   Adj. R-squared:                  0.003
Method:                 Least Squares   F-statistic:                     2.653
Date:                Sat, 03 Feb 2018   Prob (F-statistic):              0.104
Time:                        11:11:55   Log-Likelihood:                -2879.1
No. Observations:                 517   AIC:                             5762.
Df Residuals:                     515   BIC:                             5771.
Df Model:                           1
Covariance Type:            nonrobust
========================================================================================
                           coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------
Intercept                8.4054      3.905      2.152      0.032       0.733      16.077
I(X + X ** 2 + X ** 3)   0.0213      0.013      1.629      0.104      -0.004       0.047
==============================================================================
Omnibus:                      982.556   Durbin-Watson:                   1.656
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           809290.384
Skew:                          12.774   Prob(JB):                         0.00
Kurtosis:                     195.135   Cond. No.                         418.
==============================================================================
```

:

Month:

```
  "This module will be removed in 0.20.", DeprecationWarning)
                            OLS Regression Results
==============================================================================
Dep. Variable:                   area   R-squared:                       0.002
Model:                            OLS   Adj. R-squared:                  0.000
Method:                 Least Squares   F-statistic:                     1.203
Date:                Sat, 03 Feb 2018   Prob (F-statistic):              0.273
Time:                        11:21:26   Log-Likelihood:                -2879.8
No. Observations:                 517   AIC:                             5764.
Df Residuals:                     515   BIC:                             5772.
Df Model:                           1
Covariance Type:            nonrobust
========================================================================================
                              coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------
Intercept                   7.4747      5.642      1.325      0.186      -3.610      18.559
I(month + month ** 2 + month ** 3)     0.0091      0.008      1.097      0.273      -0.007       0.026
==============================================================================
Omnibus:                      983.245   Durbin-Watson:                   1.645
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           807567.016
Skew:                          12.797   Prob(JB):                         0.00
Kurtosis:                     194.921   Cond. No.                     1.37e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

DMC:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   area   R-squared:                       0.002
Model:                            OLS   Adj. R-squared:                  0.000
Method:                 Least Squares   F-statistic:                     1.110
Date:                Sat, 03 Feb 2018   Prob (F-statistic):              0.293
Time:                        11:25:55   Log-Likelihood:                -2879.9
No. Observations:                 517   AIC:                             5764.
Df Residuals:                     515   BIC:                             5772.
Df Model:                           1
Covariance Type:            nonrobust
========================================================================================
                              coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------
Intercept                  10.9987      3.304      3.329      0.001       4.509      17.489
I(DMC + DMC ** 2 + DMC ** 3)  6.409e-07   6.08e-07      1.054      0.293   -5.54e-07    1.84e-06
==============================================================================
Omnibus:                      983.586   Durbin-Watson:                   1.650
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           812739.069
Skew:                          12.803   Prob(JB):                         0.00
Kurtosis:                     195.544   Cond. No.                     6.41e+06
==============================================================================
```

DC:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   area   R-squared:                       0.001
Model:                            OLS   Adj. R-squared:                 -0.001
Method:                 Least Squares   F-statistic:                    0.5276
Date:                Sat, 03 Feb 2018   Prob (F-statistic):              0.468
Time:                        11:26:56   Log-Likelihood:                -2880.2
No. Observations:                 517   AIC:                             5764.
Df Residuals:                     515   BIC:                             5773.
Df Model:                           1
Covariance Type:            nonrobust
===============================================================================================
                              coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------------
Intercept                   9.7937      5.051      1.939      0.053      -0.130      19.718
I(DC + DC ** 2 + DC ** 3) 1.226e-08   1.69e-08      0.726      0.468   -2.09e-08    4.54e-08
==============================================================================
Omnibus:                      983.288   Durbin-Watson:                   1.646
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           808494.764
Skew:                          12.797   Prob(JB):                         0.00
Kurtosis:                     195.033   Cond. No.                     5.40e+08
==============================================================================
```

RH:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   area   R-squared:                       0.004
Model:                            OLS   Adj. R-squared:                  0.002
Method:                 Least Squares   F-statistic:                     1.862
Date:                Sat, 03 Feb 2018   Prob (F-statistic):              0.173
Time:                        11:29:23   Log-Likelihood:                -2879.5
No. Observations:                 517   AIC:                             5763.
Df Residuals:                     515   BIC:                             5772.
Df Model:                           1
Covariance Type:            nonrobust
===============================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------------
Intercept                 16.0295      3.642      4.402      0.000       8.875      23.184
I(RH + RH ** 2 + RH ** 3) -2.483e-05  1.82e-05     -1.365      0.173   -6.06e-05    1.09e-05
==============================================================================
Omnibus:                      982.283   Durbin-Watson:                   1.644
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           804594.772
Skew:                          12.770   Prob(JB):                         0.00
Kurtosis:                     194.568   Cond. No.                     2.61e+05
==============================================================================
```

Wind:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   area   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                 -0.002
Method:                 Least Squares   F-statistic:                   0.03147
Date:                Sat, 03 Feb 2018   Prob (F-statistic):              0.859
Time:                        11:30:17   Log-Likelihood:                -2880.4
No. Observations:                 517   AIC:                             5765.
Df Residuals:                     515   BIC:                             5773.
Df Model:                           1
Covariance Type:            nonrobust
=====================================================================================================
                                       coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------------------
Intercept                           13.2559      3.627      3.654      0.000       6.130      20.382
I(wind + wind ** 2 + wind ** 3)     -0.0031      0.018     -0.177      0.859      -0.038       0.032
==============================================================================
Omnibus:                      983.644   Durbin-Watson:                   1.651
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           809915.938
Skew:                          12.807   Prob(JB):                         0.00
Kurtosis:                     195.202   Cond. No.                         265.
==============================================================================
```

Rain:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   area   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                 -0.002
Method:                 Least Squares   F-statistic:                  0.001631
Date:                Sat, 03 Feb 2018   Prob (F-statistic):              0.968
Time:                        11:31:08   Log-Likelihood:                -2880.4
No. Observations:                 517   AIC:                             5765.
Df Residuals:                     515   BIC:                             5773.
Df Model:                           1
Covariance Type:            nonrobust
=====================================================================================================
                                      coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------------------
Intercept                          12.8525      2.805      4.582      0.000       7.341      18.364
I(rain + rain ** 2 + rain ** 3)    -0.0083      0.206     -0.040      0.968      -0.413       0.396
==============================================================================
Omnibus:                      983.739   Durbin-Watson:                   1.649
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           810350.266
Skew:                          12.809   Prob(JB):                         0.00
Kurtosis:                     195.254   Cond. No.                         13.6
==============================================================================
```

Code:

```
import pandas as pd

import statsmodels.formula.api as smf



dataframe = pd.read_csv("Forest_Fire/forestfires.csv")

dataframe.month.replace(('jan','feb','mar','apr','may','jun','jul','aug','sep','oct','
nov','dec'),(1,2,3,4,5,6,7,8,9,10,11,12), inplace=True)

dataframe.day.replace(('mon','tue','wed','thu','fri','sat','sun'),(1,2,3,4,5,6,7),
inplace=True)
```

```
mod = smf.ols(formula='area~ I(rain+rain**2+rain**3)', data=dataframe)

res = mod.fit()

print(res.summary())
```

g)

Code:

```
import numpy as np

import pandas

import copy

import statsmodels.formula.api as smf


seed = 7

np.random.seed(seed)

dataframe = pandas.read_csv("Forest_Fire/forestfires.csv")

data=copy.copy(dataframe)
dataframe.month.replace(('jan','feb','mar','apr','may','jun','jul','aug','sep','oct','
nov','dec'),(1,2,3,4,5,6,7,8,9,10,11,12), inplace=True)

dataframe.day.replace(('mon','tue','wed','thu','fri','sat','sun'),(1,2,3,4,5,6,7),
inplace=True)

dataframe['area'] = np.log(dataframe['area']+1)


mod = smf.ols(formula='area~ I(temp*month+temp*wind+month*wind)', data=dataframe)

res = mod.fit()

print(res.summary())
```

Temp, month and wind look like significant contributors to the response.

Results of several predictor combinations are as follows:

```
  "This module will be removed in 0.20.", DeprecationWarning)
                            OLS Regression Results
==============================================================================
Dep. Variable:                   area   R-squared:                       0.007
Model:                            OLS   Adj. R-squared:                  0.005
Method:                 Least Squares   F-statistic:                     3.667
Date:                Sat, 03 Feb 2018   Prob (F-statistic):             0.0561
Time:                        12:10:16   Log-Likelihood:                -904.64
No. Observations:                 517   AIC:                             1813.
Df Residuals:                     515   BIC:                             1822.
Df Model:                           1
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
Intercept          0.9226      0.116      7.954      0.000       0.695       1.150
I(month * DMC)     0.0002      0.000      1.915      0.056   -5.46e-06       0.000
==============================================================================
Omnibus:                       91.501   Durbin-Watson:                   0.922
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              137.837
Skew:                           1.192   Prob(JB):                     1.17e-30
Kurtosis:                       3.844   Cond. No.                     2.00e+03
==============================================================================
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   area   R-squared:                       0.007
Model:                            OLS   Adj. R-squared:                  0.005
Method:                 Least Squares   F-statistic:                     3.667
Date:                Sat, 03 Feb 2018   Prob (F-statistic):             0.0561
Time:                        12:09:22   Log-Likelihood:                -904.64
No. Observations:                 517   AIC:                             1813.
Df Residuals:                     515   BIC:                             1822.
Df Model:                           1
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
Intercept         0.8812      0.135      6.538      0.000       0.616       1.146
I(month * DC)  5.011e-05   2.62e-05      1.915      0.056    -1.3e-06       0.000
==============================================================================
Omnibus:                       93.627   Durbin-Watson:                   0.921
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              142.615
Skew:                           1.205   Prob(JB):                     1.08e-31
Kurtosis:                       3.901   Cond. No.                     1.13e+04
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.13e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
  "This module will be removed in 0.20.", DeprecationWarning)
                            OLS Regression Results
==============================================================================
Dep. Variable:                   area   R-squared:                       0.005
Model:                            OLS   Adj. R-squared:                  0.003
Method:                 Least Squares   F-statistic:                     2.472
Date:                Sat, 03 Feb 2018   Prob (F-statistic):              0.116
Time:                        12:08:33   Log-Likelihood:                -905.23
No. Observations:                 517   AIC:                             1814.
Df Residuals:                     515   BIC:                             1823.
Df Model:                           1
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
Intercept       0.9801      0.103      9.473      0.000       0.777       1.183
I(DMC * DC)  1.829e-06   1.16e-06      1.572      0.116   -4.56e-07    4.11e-06
==============================================================================
Omnibus:                       92.846   Durbin-Watson:                   0.922
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              140.811
Skew:                           1.201   Prob(JB):                     2.65e-31
Kurtosis:                       3.874   Cond. No.                     1.50e+05
==============================================================================
```

```
C:\Users\Shanu\AppData\Roaming\Python\Python36\site-packages\sklearn\cross_validation.py:41: DeprecationWarning: This modu
  "This module will be removed in 0.20.", DeprecationWarning)
                            OLS Regression Results
==============================================================================
Dep. Variable:                   area   R-squared:                       0.012
Model:                            OLS   Adj. R-squared:                  0.010
Method:                 Least Squares   F-statistic:                     6.299
Date:                Sat, 03 Feb 2018   Prob (F-statistic):             0.0124
Time:                        12:13:40   Log-Likelihood:                -903.33
No. Observations:                 517   AIC:                             1811.
Df Residuals:                     515   BIC:                             1819.
Df Model:                           1
Covariance Type:            nonrobust
==================================================================================================================
                                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------------------------
Intercept                                        0.6927      0.178      3.901      0.000       0.344       1.042
I(temp * month + temp * wind + month * wind)     0.0017      0.001      2.510      0.012       0.000       0.003
==============================================================================
Omnibus:                       88.899   Durbin-Watson:                   0.920
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              132.189
Skew:                           1.177   Prob(JB):                     1.98e-29
Kurtosis:                       3.774   Cond. No.                         771.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   area   R-squared:                       0.008
Model:                            OLS   Adj. R-squared:                  0.006
Method:                 Least Squares   F-statistic:                     4.319
Date:                Sat, 03 Feb 2018   Prob (F-statistic):             0.0382
Time:                        12:12:15   Log-Likelihood:                -904.31
No. Observations:                 517   AIC:                             1813.
Df Residuals:                     515   BIC:                             1821.
Df Model:                           1
Covariance Type:            nonrobust
===================================================================================
                      coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------
Intercept           0.8139      0.156      5.233      0.000       0.508       1.120
I(temp * month)     0.0020      0.001      2.078      0.038       0.000       0.004
==============================================================================
Omnibus:                       90.075   Durbin-Watson:                   0.922
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              134.681
Skew:                           1.186   Prob(JB):                     5.68e-30
Kurtosis:                       3.794   Cond. No.                         403.
==============================================================================
```

h)

Code:

```
import numpy as np

import pandas

import copy

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error


seed = 7

np.random.seed(seed)
```

```
dataframe = pandas.read_csv("Forest_Fire/forestfires.csv")

data=copy.copy(dataframe)

dataframe.month.replace(('jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec'),(1,2,3,4,5,6,7,8,9,10,11,12), inplace=True)

dataframe.day.replace(('mon','tue','wed','thu','fri','sat','sun'),(1,2,3,4,5,6,7), inplace=True)

dataframe['area'] = np.log(dataframe['area']+1)


df_sample = dataframe.sample(frac=0.7)


X_train=df_sample.iloc[:,:12]

Y_train=df_sample.iloc[:,12]


df_rest = dataframe.loc[~dataframe.index.isin(df_sample.index)]


X_test = df_rest.iloc[:, :12]

Y_test = df_rest.iloc[:, 12]


X_train['new_pred']=X_train['RH']*X_train['temp']+X_train['temp']*X_train['month']+X_train['RH']*X_train['month']

X_test['new_pred']=X_test['RH']*X_test['temp']+X_test['temp']*X_test['month']+X_test['RH']*X_test['month']


plot_uni_model=LinearRegression()


res=plot_uni_model.fit(X_train,Y_train)

predictions=plot_uni_model.predict(X_test)

score=plot_uni_model.score(X_test,Y_test)

error = mean_squared_error(predictions, Y_test)
```

Train errors:

dc,month,dmc:

1.82856750304499

month,temp,wind:

1.8257893729464

RH,month,temp:

1.8272469364478

Test Error:

dc,month,dmc:

2.5402531452328

temp,month,wind:

2.55770011868637

RH,month,temp:

2.5061727601934

i)

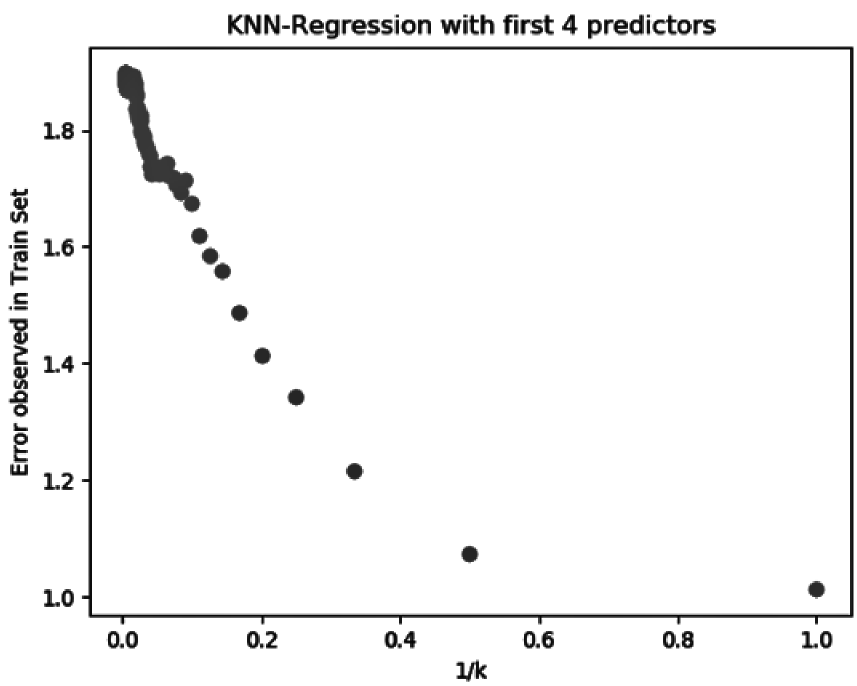
i: First 4 predictors:

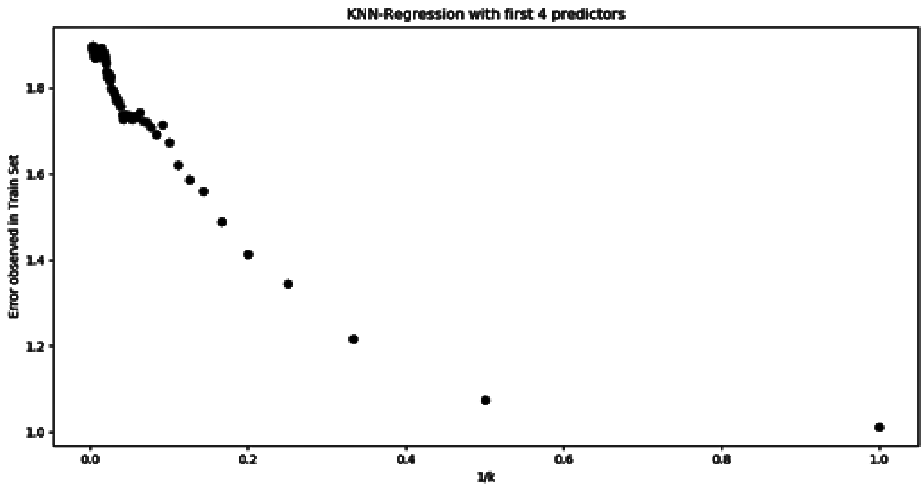
Best error rate when k=1

Code:

```
X_Knn_train=dataset[:400,0:4]
Y_Knn_train=dataset[:400,12]

X_Knn = dataset[400:, 0:4]
Y_Knn = dataset[400:, 12]

error_arr=[]
k_arr=[]
# knn regression
for k in range(1,350):
```

```
    neigh = KNeighborsRegressor(n_neighbors=k)
    neigh.fit(X_Knn_train, Y_Knn_train)
    predictions=neigh.predict(X_Knn_train)

    error= mean_squared_error(Y_Knn_train,predictions)
    score=neigh.score(X_Knn_train,Y_Knn_train)
    error_arr.append(error)
    k_arr.append((1/k))
    plt.plot(1/k, error, marker='o',linestyle='--', color='b')
    plt.xlabel('1/k')
    plt.ylabel('Error observed in Train Set')
    plt.title('KNN-Regression with first 4 predictors')
plt.show()
```



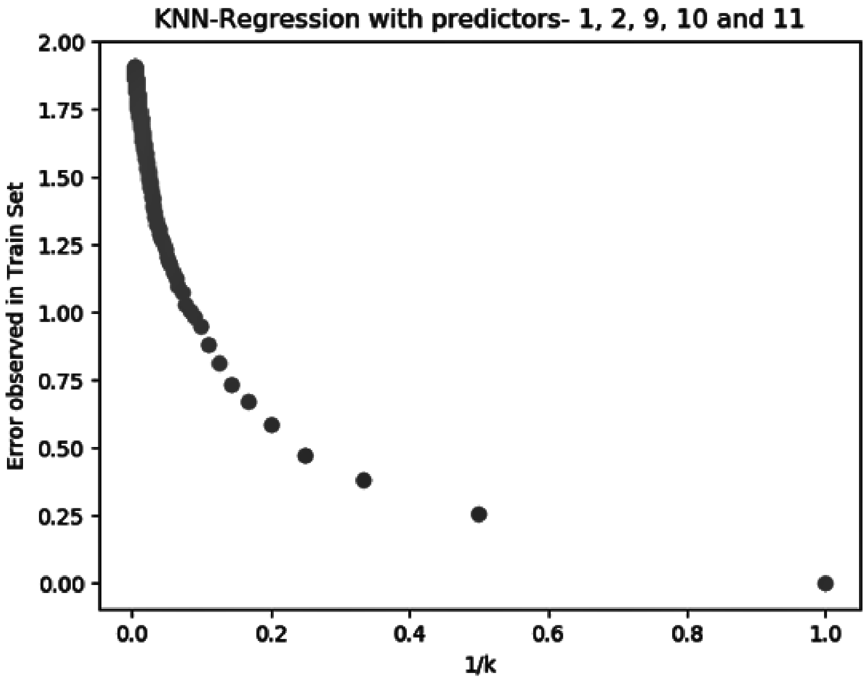


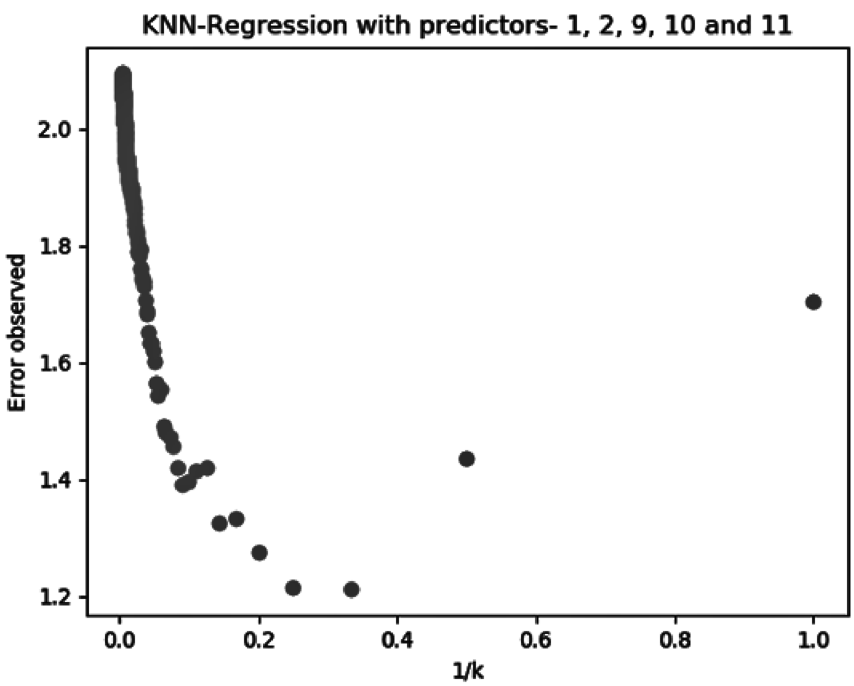
i)

iii KNN regression with predictors 1,2,9,10,11

K=1.1

```
dataset_kNN = copy.copy(dataframe)
dataset_kNN.drop(['day','FFMC','DMC','DC','ISI','rain'],axis=1, inplace=True)
X_Knn_train=dataset_kNN.iloc[:400,:]
Y_Knn_train=dataset[:400,12]

X_Knn = dataset_kNN.iloc[400:, :]
Y_Knn = dataset[400:, 12]

error_arr=[]
k_arr=[]
# knn regression
for k in range(1,350):
    neigh = KNeighborsRegressor(n_neighbors=k)
    neigh.fit(X_Knn_train, Y_Knn_train)
    predictions=neigh.predict(X_Knn_train)

    # print(predictions)
    error= mean_squared_error(Y_Knn_train,predictions)
    score=neigh.score(X_Knn_train,Y_Knn_train)
    error_arr.append(error)
    k_arr.append((1/k))
    plt.plot(1/k, error,marker='o', linestyle='--', color='b', label='vals')
             # , 16,c=1, alpha=0.5)
    plt.xlabel('1/k')
    plt.ylabel('Error observed in Train Set')
    plt.title('KNN-Regression with predictors- 1, 2, 9, 10 and 11')
plt.show()
```





ii)

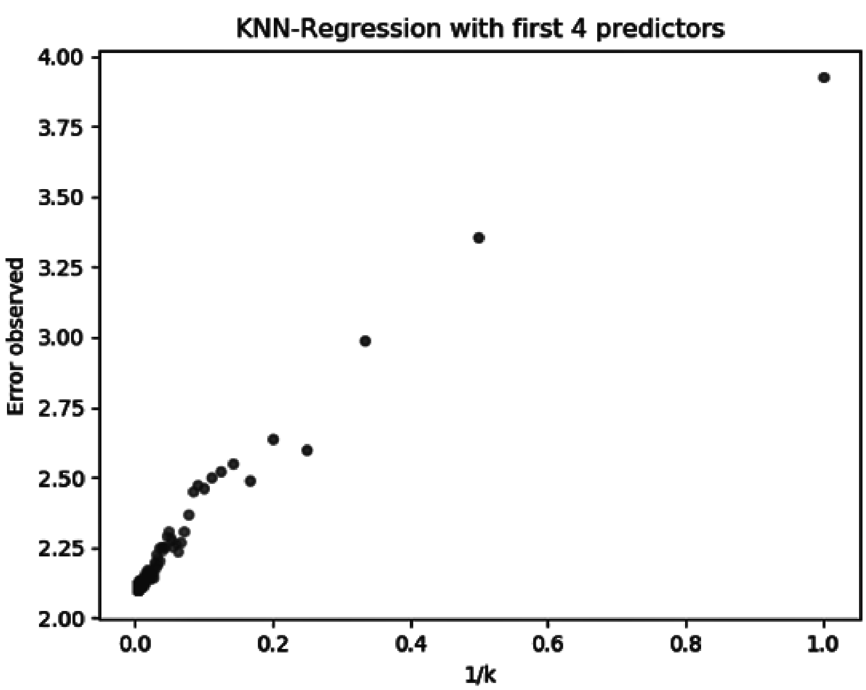
ii: Last 4 predictors

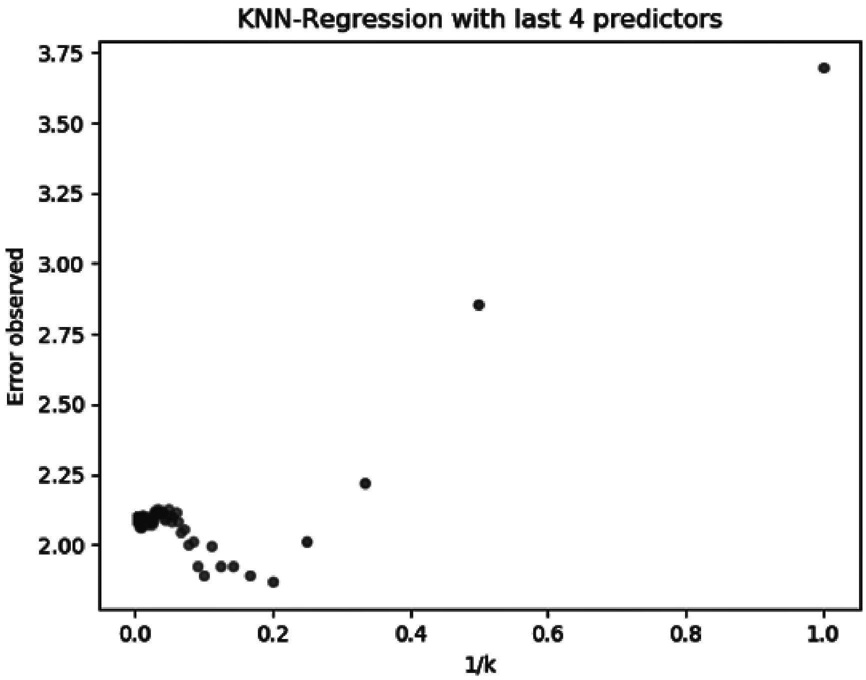
k=1

Code:

```
X_Knn_train=dataset[:400,8:12]
Y_Knn_train=dataset[:400,12]

X_Knn = dataset[400:, 8:12]
Y_Knn = dataset[400:, 12]

error_arr=[]
k_arr=[]
# knn regression
for k in range(1,350):
    neigh = KNeighborsRegressor(n_neighbors=k)
    neigh.fit(X_Knn_train, Y_Knn_train)
    predictions=neigh.predict(X_Knn_train)

    # print(predictions)
    error= mean_squared_error(Y_Knn_train,predictions)
    score=neigh.score(X_Knn_train,Y_Knn_train)
    error_arr.append(error)
    k_arr.append((1/k))
    plt.scatter(1/k, error, 16,c=16, alpha=0.5)
    plt.xlabel('1/k')
    plt.ylabel('Error observed in Train set')
    plt.title('KNN-Regression with last 4 predictors')
plt.show()
```

Test Error:





j)

Best error rate for linear regression was 0.75 which was lower but took longer time. Best error rate of KNN regression was around 1 but it is faster.

Code

Question: 1

```
import numpy as np
import mnist_loader_KNN as DS_loader
import matplotlib.pyplot as plt
import random
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from collections import defaultdict
import seaborn as sns
from struct import unpack
url_train_image = 'http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz'
url_train_labels = 'http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz'
num_train_samples = 60000


x_train = DS_loader.try_download_x(url_train_image, url_train_labels,
num_train_samples)
y_train= DS_loader.try_download_y(url_train_image, url_train_labels,
num_train_samples)

url_test_image = 'http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz'
```

```
url_test_labels = 'http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz'
num_test_samples = 10000

print("Downloading test data")
temp=random.randrange(9999)

x_test = DS_loader.try_download_x(url_test_image, url_test_labels, num_test_samples)
y_test = DS_loader.try_download_y(url_test_image, url_test_labels, num_test_samples)

print(y_test.shape)

sample=x_test[temp,:].reshape(28,28)
index_error=1;
error_score=[]
for i in range(1,10001):

knnclassifier=KNeighborsClassifier(n_neighbors=i,metrics='minkowski',p=1,algorithm='ba
ll_tree')
    knnclassifier.fit(x_train,y_train.ravel())
    predictions= knnclassifier.predict(x_test)

    score=knnclassifier.score(x_test,y_test)
    print(score)
    print(predictions)
    error_score[index_error]=metrics.mean_squared_error(y_test,predictions,
multioutput='raw_values')
    print(error_score[index_error])
    i=i+200
    index_error+=1
```

Question 1e:

```
import numpy as np
import mnist_loader_KNN as DS_loader
import matplotlib.pyplot as plt
import random
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from collections import defaultdict
import seaborn as sns
from struct import unpack
import plotly.plotly as py
import plotly.graph_objs as go

url_train_image = 'http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz'
url_train_labels = 'http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz'
num_train_samples = 60000

print("Downloading train data")

x_train = DS_loader.try_download_x(url_train_image, url_train_labels,
num_train_samples)
y_train = DS_loader.try_download_y(url_train_image, url_train_labels,
num_train_samples)
url_test_image = 'http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz'
url_test_labels = 'http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz'
num_test_samples = 10000

print("Downloading test data")
temp=random.randrange(9999)
```

```
x_test = DS_loader.try_download_x(url_test_image, url_test_labels, num_test_samples)
y_test = DS_loader.try_download_y(url_test_image, url_test_labels, num_test_samples)

print(y_test.shape)

sample=x_test[temp,:].reshape(28,28)

index_error=1
error_score=[]
x_test_subset = x_test[:600, :]
y_test_subset = y_test[:600, :]

plot_x=[]

x_temp = x_train[:25001, :]
y_temp = y_train[:25001, :]

for i in range(1,11):
    knnclassifier=KNeighborsClassifier(n_neighbors=1,metric='minkowski',p=pow(10,i),
algorithm='ball_tree')

    knnclassifier.fit(x_temp, y_temp.ravel())
    predictions = knnclassifier.predict(x_test_subset)

    score = knnclassifier.score(x_test_subset, y_test_subset)
    print(score)
    error = metrics.mean_squared_error(y_test_subset,predictions,
multioutput='raw_values')
    error_score.append(error)

trace = go.Table(

header=dict(values=['Minkowski(log)=0.1','Minkowski(log)=0.2','Minkowski(log)=0.3','Mi
nkowski(log)=0.4','Minkowski(log)=0.5','Minkowski(log)=0.6','Minkowski(log)=0.7','Mink
owski(log) 0.8','Minkowski(log)=0.9','Minkowski(log) 1.0' ]),
    cells=dict(values=error_score))

data = [trace]
py.iplot(data, filename = 'basic_table')
```

Question 1civ:

```
import numpy as np
import mnist_loader_KNN as DS_loader
import matplotlib.pyplot as plt
import random
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
from collections import defaultdict
import seaborn as sns
from struct import unpack
#!gzip -d data/*.gz


url_train_image = 'http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz'
url_train_labels = 'http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz'
num_train_samples = 60000

print("Downloading train data")

x_train = DS_loader.try_download_x(url_train_image, url_train_labels,
```

```
num_train_samples)
y_train = DS_loader.try_download_y(url_train_image, url_train_labels,
num_train_samples)

url_test_image = 'http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz'
url_test_labels = 'http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz'
num_test_samples = 10000

print("Downloading test data")
temp=random.randrange(9999)

x_test = DS_loader.try_download_x(url_test_image, url_test_labels, num_test_samples)
y_test = DS_loader.try_download_y(url_test_image, url_test_labels, num_test_samples)

print(y_test.shape)

sample=x_test[temp,:].reshape(28,28)
prime_knnclassifier = KNeighborsClassifier(n_neighbors=1,algorithm='ball_tree')

prime_predictions = []
index_error=1
error_score=[]
x_test_subset = x_test[:600, :]
y_test_subset = y_test[:600, :]


knnclassifier=KNeighborsClassifier(n_neighbors=1,algorithm='ball_tree')
knnclassifier.fit(x_train,y_train.ravel())
print('The Model is now set')
predictions= knnclassifier.predict(x_test_subset)
print('Prediction Complete!')
prime_predictions = predictions
prime_knnclassifier = knnclassifier
miss_index = 0
misclassifiedIndexes = []
for label, predict in zip(y_test_subset, prime_predictions):
    if label != predict:
        misclassifiedIndexes.append(miss_index)
    miss_index += 1

image_indices = []
print(len(misclassifiedIndexes))

for i in misclassifiedIndexes:
    distances, indices = prime_knnclassifier.kneighbors(x_test[i].reshape(1,-1))
    for test_data in indices:
        for index in test_data:
            image_indices.append(index)

plt.figure(figsize=(10,20))
img_plot_index = 0
neighbour_index=0
for img_index in misclassifiedIndexes:
    img = x_test_subset[img_index, :].reshape(28, 28)#this is the prediction
    img_plot_index = img_plot_index + 1
    plt.subplot(10, 5, img_plot_index)

    plt.imshow(img, cmap=plt.cm.gray)

    img = x_train[[image_indices[neighbour_index]],:].reshape(28,28)#this is the
neighbour
    img_plot_index = img_plot_index + 1
    plt.subplot(10, 5, img_plot_index)
```

```
    plt.imshow(img, cmap=plt.cm.gray)

    print("hello")
    neighbour_index+=1
plt.show()
```

Question 2b iii and iv:

```
import numpy as np
import pandas
import copy
from sklearn.feature_selection import RFE
from sklearn.ensemble import ExtraTreesRegressor
from scipy import stats
from scipy import stats
from sklearn.neighbors import KNeighborsRegressor



import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix


from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.metrics import explained_variance_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error


from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.utils import np_utils
from keras.constraints import maxnorm
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LassoLarsCV



# fix random seed for reproducibility
seed = 7
np.random.seed(seed)

# load the dataset
dataframe = pandas.read_csv("Forest_Fire/forestfires.csv")
```

```
data=copy.copy(dataframe)
# Encode Data
dataframe.month.replace(('jan','feb','mar','apr','may','jun','jul','aug','sep','oct','
nov','dec'),(1,2,3,4,5,6,7,8,9,10,11,12), inplace=True)
dataframe.day.replace(('mon','tue','wed','thu','fri','sat','sun'),(1,2,3,4,5,6,7),
inplace=True)
dataframe['area'] = np.log(dataframe['area']+1)

scatter_matrix(dataframe)
plt.show()
```

Question 2:

```
import numpy as np
import pandas
import copy
from sklearn.feature_selection import RFE
from sklearn.ensemble import ExtraTreesRegressor
from scipy import stats
from scipy import stats
from sklearn.neighbors import KNeighborsRegressor



import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix


from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.metrics import explained_variance_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error


from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.utils import np_utils
from keras.constraints import maxnorm
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LassoLarsCV



# fix random seed for reproducibility
seed = 7
```

```
np.random.seed(seed)

# load the dataset
dataframe = pandas.read_csv("Forest_Fire/forestfires.csv")
data=copy.copy(dataframe)
# Encode Data
dataframe.month.replace(('jan','feb','mar','apr','may','jun','jul','aug','sep','oct','
nov','dec'),(1,2,3,4,5,6,7,8,9,10,11,12), inplace=True)
dataframe.day.replace(('mon','tue','wed','thu','fri','sat','sun'),(1,2,3,4,5,6,7),
inplace=True)
dataframe['area'] = np.log(dataframe['area']+1)
print("Head:")
print(dataframe.head())
print("Statistical Description:")
print(dataframe.describe())
print("Shape:")
print(dataframe.shape)
print("Data Types:")
print(dataframe.dtypes)
print("Correlation:")
print(dataframe.corr(method='pearson'))
print('Median')
print(dataframe.median())
#
print('Range')
print(dataframe.max()-dataframe.min())

print('Mean')
print(dataframe.mean())

print('First Quartile')
print(dataframe.quantile(q=0.25, axis=0, numeric_only=True, interpolation='linear')

print(Third Quartile')
print(dataframe.quantile(q=0.75, axis=0, numeric_only=True, interpolation='linear')

print('Inter-quartile Ranges')
print(dataframe.quantile(q=0.75, axis=0, numeric_only=True, interpolation='linear')-
dataframe.quantile(q=0.25,axis=0,numeric_only=True,interpolation='linear'))
```

Questions 2 c:

```
import matplotlib.pyplot as plt
plt.hist((dataframe.area))
dataframe.hist()

import numpy as np

import pandas

import copy

import statsmodels.formula.api as smf






# fix random seed for reproducibility

seed = 7
```

```python
np.random.seed(seed)


# load the dataset

dataframe = pandas.read_csv("Forest_Fire/forestfires.csv")

data=copy.copy(dataframe)

# Encode Data

dataframe.month.replace(('jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec'),(1,2,3,4,5,6,7,8,9,10,11,12), inplace=True)

dataframe.day.replace(('mon','tue','wed','thu','fri','sat','sun'),(1,2,3,4,5,6,7), inplace=True)

dataframe['area'] = np.log(dataframe['area']+1)




mod = smf.ols(formula='area~ I(ISI)', data=dataframe)

res = mod.fit()

print(res.summary())


mod = smf.ols(formula='area~ I(RH)', data=dataframe)

res = mod.fit()

print(res.summary())


mod = smf.ols(formula='area~ I(rain)', data=dataframe)

res = mod.fit()

print(res.summary())


mod = smf.ols(formula='area~ I(wind)', data=dataframe)

res = mod.fit()

print(res.summary())


mod = smf.ols(formula='area~ I(temp)', data=dataframe)

res = mod.fit()

print(res.summary())
```

```
mod = smf.ols(formula='area~ I(DMC)', data=dataframe)

res = mod.fit()

print(res.summary())


mod = smf.ols(formula='area~ I(DC)', data=dataframe)

res = mod.fit()

print(res.summary())


mod = smf.ols(formula='area~ I(FFMC)', data=dataframe)

res = mod.fit()

print(res.summary())


mod = smf.ols(formula='area~ I(day)', data=dataframe)

res = mod.fit()

print(res.summary())


mod = smf.ols(formula='area~ I(month)', data=dataframe)

res = mod.fit()

print(res.summary())


mod = smf.ols(formula='area~ I(Y)', data=dataframe)

res = mod.fit()

print(res.summary())


mod = smf.ols(formula='area~ I(X)', data=dataframe)

res = mod.fit()

print(res.summary())
```

Question 2g:

```
import numpy as np
import pandas
import copy
from sklearn.feature_selection import RFE
from sklearn.ensemble import ExtraTreesRegressor
from scipy import stats
```

```
from scipy import stats
from sklearn.neighbors import KNeighborsRegressor



import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix


from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.metrics import explained_variance_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error


from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.utils import np_utils
from keras.constraints import maxnorm
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LassoLarsCV
import statsmodels.formula.api as smf

seed = 7
np.random.seed(seed)
dataframe = pandas.read_csv("Forest_Fire/forestfires.csv")
data=copy.copy(dataframe)
dataframe.month.replace(('jan','feb','mar','apr','may','jun','jul','aug','sep','oct','
nov','dec'),(1,2,3,4,5,6,7,8,9,10,11,12), inplace=True)
dataframe.day.replace(('mon','tue','wed','thu','fri','sat','sun'),(1,2,3,4,5,6,7),
inplace=True)
dataframe['area'] = np.log(dataframe['area']+1)
mod = smf.ols(formula='area~ I(temp*month+temp*wind+month*wind)', data=dataframe)
res = mod.fit()
print(res.summary())
```

Question 2h:

```
import numpy as np
import pandas
import copy
import random
from sklearn.feature_selection import RFE
from sklearn.ensemble import ExtraTreesRegressor
from scipy import stats
from scipy import stats
```

```
from sklearn.neighbors import KNeighborsRegressor



import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix


from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.metrics import explained_variance_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error


from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.utils import np_utils
from keras.constraints import maxnorm
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LassoLarsCV
import statsmodels.formula.api as smf


seed = 7
np.random.seed(seed)

dataframe = pandas.read_csv("Forest_Fire/forestfires.csv")
data=copy.copy(dataframe)

dataframe.day.replace(('mon','tue','wed','thu','fri','sat','sun'),(1,2,3,4,5,6,7),
inplace=True)
dataframe.month.replace(('jan','feb','mar','apr','may','jun','jul','aug','sep','oct','
nov','dec'),(1,2,3,4,5,6,7,8,9,10,11,12), inplace=True)
dataframe['area'] = np.log(dataframe['area']+1)

df_sample = dataframe.sample(frac=0.7)

X_train=df_sample.iloc[:,:12]
Y_train=df_sample.iloc[:,12]

df_rest = dataframe.loc[~dataframe.index.isin(df_sample.index)]

X_test = df_rest.iloc[:, :12]
Y_test = df_rest.iloc[:, 12]
#
```

```
X_train['new_pred']=X_train['wind']*X_train['temp']+X_train['temp']*X_train['month']+X
_train['RH']*X_train['month']
X_test['new_pred']=X_test['wind']*X_test['temp']+X_test['temp']*X_test['month']+X_test
['RH']*X_test['month']

model=LinearRegression()
res=model.fit(X_train,Y_train)
predictions=plot_uni_model.predict(X_test)
score=model.score(X_test,Y_test)
error = mean_squared_error(predictions, Y_test)
print(error)
print(score)
```

Question 2 remaining:

```
import numpy as np
import pandas
import copy
from sklearn.feature_selection import RFE
from sklearn.ensemble import ExtraTreesRegressor
from scipy import stats
from scipy import stats
from sklearn.neighbors import KNeighborsRegressor



import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix


from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.ensemble import BaggingRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.metrics import explained_variance_score
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error


from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.utils import np_utils
from keras.constraints import maxnorm
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LassoLarsCV


seed = 7
np.random.seed(seed)
```

```
dataframe = pandas.read_csv("Forest_Fire/forestfires.csv")
data=copy.copy(dataframe)
# Encode Data
dataframe.month.replace(('jan','feb','mar','apr','may','jun','jul','aug','sep','oct','
nov','dec'),(1,2,3,4,5,6,7,8,9,10,11,12), inplace=True)
dataframe.day.replace(('mon','tue','wed','thu','fri','sat','sun'),(1,2,3,4,5,6,7),
inplace=True)
dataframe['area'] = np.log(dataframe['area']+1)
print("Head:")
print(dataframe.head())
print("Statistical Description:")
print(dataframe.describe())
print("Shape:")
print(dataframe.shape)
print("Data Types:")
print(dataframe.dtypes)
print("Correlation:")
print(dataframe.corr(method='pearson'))
dataset = dataframe.values


X = dataset[:,0:12]
Y = dataset[:,12]

#Feature Selection
uni_model = ExtraTreesRegressor()
rfe = RFE(uni_model, 4)
fit = rfe.fit(X, Y)

print("Number of Features: ", fit.n_features_)
print("Selected Features: ", fit.support_)
print("Feature Ranking: ", fit.ranking_)

scatter_matrix(dataframe)
plt.show()


my_labels=list(dataframe[0:13])
print(my_labels)
labels=my_labels[0:12]
print(labels)
plot_uni_model=LinearRegression()
coef_uni_vaale=plot_uni_model.fit(X,Y).coef_

uni_pred=plot_uni_model.predict(X)
error=mean_squared_error(Y,uni_pred)
print(error)
ax = plt.gca()
plt.scatter(labels,coef_uni_vaale,16,alpha=0.5)
plt.ylabel('Regression Coefficients')
plt.xlabel('predictors')
plt.title('Regression Coefficients Progression for linear regression Paths')
plt.show()


pred_train, pred_test, tar_train, tar_test = train_test_split(X, Y,
                                                              test_size=.3,
random_state=123)

multi_model=LassoLarsCV(cv=10, precompute=False).fit(pred_train,tar_train)

print(dict(zip(labels[:12], multi_model.coef_)))
```

```
# plot coefficient progression
m_log_alphas = -np.log10(multi_model.alphas_)
ax = plt.gca()
plt.plot(m_log_alphas, multi_model.coef_path_.T)
plt.axvline(-np.log10(multi_model.alpha_), linestyle='--', color='k',
            label='alpha CV')
print(multi_model.coef_)
print(multi_model.coef_.shape)
print('Error in multivariate ')
plt.ylabel('Regression Coefficients')
plt.xlabel('-log(alpha)')
plt.title('Regression Coefficients Progression for Lasso Paths')
plt.show()

# 2c and 2d
ax = plt.gca()
plt.scatter(coef_uni_vaale,multi_model.coef_)

plt.ylabel('multivariate- Regression Coefficients')
plt.xlabel('univariate- Regression Coefficients')
plt.title('Regression Coefficients comparison')
plt.show()


print('in poly function now')

poly = PolynomialFeatures(degree=3)

poly_feature=dataframe['temp']
print('printing Y')
print(Y)
print('shape of Y')
print(Y.shape)
X_Knn_train=dataset[:400,0:12]
Y_Knn_train=dataset[:400,12]

X_Knn = dataset[400:, 0:12]
Y_Knn = dataset[400:, 12]

error_arr=[]
k_arr=[]
# knn regression
for k in range(1,350):
    neigh = KNeighborsRegressor(n_neighbors=k)
    neigh.fit(X_Knn_train, Y_Knn_train)
    predictions=neigh.predict(X_Knn)

    # print(predictions)
    error= mean_squared_error(Y_Knn,predictions)
    score=neigh.score(X_Knn,Y_Knn)
    error_arr.append(error)
    k_arr.append((1/k))
    plt.plot(1/k, error, marker='o',linestyle='--', color='b')
    plt.xlabel('1/k')
    plt.ylabel('Error in Test Set')
    plt.title('KNN-Regression with All predictors')
plt.show()
```