Part 1: Part One: Distance and Similarity Metrics

1.1 Load the Ames housing data. Create a matrix with the following variables and standardize the: Sale_Price, Lot_Area, Year_Built, Gr_Liv_Area, Total_Bsmt_SF, and Full_Bath. Randomly select five (5) rows from the matrix and print them. (5 pts)

In [705]:
```
import numpy       as np
import pandas      as pd
import sklearn     as sk
import statistics as st
import pprint      as pp
import math        as mth
import seaborn     as sns
import matplotlib as mpl
import os          as os
import scipy       as sp
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear_model  import LinearRegression
from sklearn.linear_model  import LogisticRegression
from sklearn.linear_model  import LogisticRegressionCV
from sklearn.linear_model  import Ridge
from sklearn.linear_model  import Lasso

from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_validate
from sklearn.model_selection import train_test_split

from sklearn.metrics          import mean_squared_error
from sklearn.metrics          import mean_absolute_error
from sklearn.metrics          import confusion_matrix

from sklearn.metrics          import accuracy_score
from sklearn.metrics          import recall_score
from sklearn.metrics          import precision_score
from sklearn.metrics          import roc_curve
from sklearn.metrics          import roc_auc_score
from sklearn.metrics.pairwise import cosine_similarity

from sklearn.neighbors import NearestNeighbors
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split


from scipy.sparse.linalg     import norm
from scipy.spatial.distance  import cosine

#from   tensorflow import keras
%matplotlib inline
```

In [706]:
```
#Reading CSV Data File #1
data_folder = "C:/Users/HP/Google Drive/Education/Ashish/MCAS-UND/Data sets/"
file_name = data_folder + "ames.csv"
ames_ds = pd.read_csv(file_name)
```

In [707]:
```
ames_ds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2930 entries, 0 to 2929
Data columns (total 81 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   MS_SubClass         2930 non-null   object
 1   MS_Zoning           2930 non-null   object
 2   Lot_Frontage        2930 non-null   int64
 3   Lot_Area            2930 non-null   int64
 4   Street              2930 non-null   object
 5   Alley               2930 non-null   object
 6   Lot_Shape           2930 non-null   object
 7   Land_Contour        2930 non-null   object
 8   Utilities           2930 non-null   object
 9   Lot_Config          2930 non-null   object
 10  Land_Slope          2930 non-null   object
 11  Neighborhood        2930 non-null   object
 12  Condition_1         2930 non-null   object
 13  Condition_2         2930 non-null   object
 14  Bldg_Type           2930 non-null   object
 15  House_Style         2930 non-null   object
 16  Overall_Qual        2930 non-null   object
 17  Overall_Cond        2930 non-null   object
 18  Year_Built          2930 non-null   int64
 19  Year_Remod_Add      2930 non-null   int64
 20  Roof_Style          2930 non-null   object
 21  Roof_Matl           2930 non-null   object
 22  Exterior_1st        2930 non-null   object
 23  Exterior_2nd        2930 non-null   object
 24  Mas_Vnr_Type        2930 non-null   object
 25  Mas_Vnr_Area        2930 non-null   int64
 26  Exter_Qual          2930 non-null   object
 27  Exter_Cond          2930 non-null   object
 28  Foundation          2930 non-null   object
 29  Bsmt_Qual           2930 non-null   object
 30  Bsmt_Cond           2930 non-null   object
 31  Bsmt_Exposure       2930 non-null   object
 32  BsmtFin_Type_1      2930 non-null   object
 33  BsmtFin_SF_1        2930 non-null   int64
 34  BsmtFin_Type_2      2930 non-null   object
 35  BsmtFin_SF_2        2930 non-null   int64
 36  Bsmt_Unf_SF         2930 non-null   int64
 37  Total_Bsmt_SF       2930 non-null   int64
 38  Heating             2930 non-null   object
 39  Heating_QC          2930 non-null   object
 40  Central_Air         2930 non-null   object
 41  Electrical          2930 non-null   object
 42  First_Flr_SF        2930 non-null   int64
 43  Second_Flr_SF       2930 non-null   int64
 44  Low_Qual_Fin_SF     2930 non-null   int64
 45  Gr_Liv_Area         2930 non-null   int64
 46  Bsmt_Full_Bath      2930 non-null   int64
 47  Bsmt_Half_Bath      2930 non-null   int64
 48  Full_Bath           2930 non-null   int64
 49  Half_Bath           2930 non-null   int64
 50  Bedroom_AbvGr       2930 non-null   int64
```

```
 51  Kitchen_AbvGr       2930 non-null   int64
 52  Kitchen_Qual        2930 non-null   object
 53  TotRms_AbvGrd       2930 non-null   int64
 54  Functional          2930 non-null   object
 55  Fireplaces          2930 non-null   int64
 56  Fireplace_Qu        2930 non-null   object
 57  Garage_Type         2930 non-null   object
 58  Garage_Finish       2930 non-null   object
 59  Garage_Cars         2930 non-null   int64
 60  Garage_Area         2930 non-null   int64
 61  Garage_Qual         2930 non-null   object
 62  Garage_Cond         2930 non-null   object
 63  Paved_Drive         2930 non-null   object
 64  Wood_Deck_SF        2930 non-null   int64
 65  Open_Porch_SF       2930 non-null   int64
 66  Enclosed_Porch      2930 non-null   int64
 67  Three_season_porch  2930 non-null   int64
 68  Screen_Porch        2930 non-null   int64
 69  Pool_Area           2930 non-null   int64
 70  Pool_QC             2930 non-null   object
 71  Fence               2930 non-null   object
 72  Misc_Feature        2930 non-null   object
 73  Misc_Val            2930 non-null   int64
 74  Mo_Sold             2930 non-null   int64
 75  Year_Sold           2930 non-null   int64
 76  Sale_Type           2930 non-null   object
 77  Sale_Condition      2930 non-null   object
 78  Sale_Price          2930 non-null   int64
 79  Longitude           2930 non-null   float64
 80  Latitude            2930 non-null   float64
dtypes: float64(2), int64(33), object(46)
memory usage: 1.8+ MB
```

In [708]:
```
col_list = ['Lot_Area', 'Year_Built', 'Gr_Liv_Area', 'Total_Bsmt_SF',
'Full_Bath', 'Sale_Price']
```

In [709]:
```
ames_df = ames_ds[col_list].dropna()
ames_df = pd.get_dummies(ames_df, drop_first = True)
ames_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2930 entries, 0 to 2929
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Lot_Area       2930 non-null   int64
 1   Year_Built     2930 non-null   int64
 2   Gr_Liv_Area    2930 non-null   int64
 3   Total_Bsmt_SF  2930 non-null   int64
 4   Full_Bath      2930 non-null   int64
 5   Sale_Price     2930 non-null   int64
dtypes: int64(6)
memory usage: 160.2 KB
```

In [710]:
```
X = ames_df[col_list[0:5]].values
y = ames_df[['Sale_Price']].values
```

In [711]:
```
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)
obs = X[0:5,]
```

In [712]:
```
obs
```

Out[712]:
```
array([[ 2.74438073, -0.37553701,  0.30926506,  0.06519583, -1.024792
89],
       [ 0.18709726, -0.34246845, -1.19442705, -0.38389297, -1.024792
89],
       [ 0.5228137 , -0.44167415, -0.33771825,  0.62995901, -1.024792
89],
       [ 0.12845795, -0.11098849,  1.20752324,  2.40136483,  0.784028
3 ],
       [ 0.46734751,  0.84799991,  0.25584442, -0.27955921,  0.784028
3 ]])
```

1.2 Create distance matrices for these five (5) observations with the following metrics.

Print each distance matrix. Which points are closest and farthest from each other? Are they the same with each metric? (15 pts)

Euclidean distance

Manhattan distance

Minkowski distance with p=0.5

In [713]:
```
p_vals = [2, 1, 0.5]
p_names = ['Euclidean', 'Manhattan', 'Minowski = 0.5']
```

In [714]:
```
def p_root(arg_value, arg_root):
    root_value = 1 / float(arg_root)
    ret_value = round (float(arg_value) ** float(root_value), 3)
    return (ret_value)
```

In [715]:
```
# ------------------------------------------------------------------
# Functions for Minowski Distance for calculating Distances
# ------------------------------------------------------------------
def f_minowski_distance(x, y, p_value):
    s = sum ( pow(abs(a-b), p_value) for a, b in zip(x, y) )
    ret_value = p_root( s , p_value)
    return (ret_value)
```

In [716]:
```
# Creating a 3 dimensional matrix of 3,5,5

n, p = obs.shape
dist_mtx = np.vectorize(float)(np.arange(3*n*p).reshape(3,n,p))
dist_mtx[:] = np.NaN
```

In [717]:
```
# Evaluating min and max distances for all distance types
for i, (p_val, p_name) in enumerate(zip(p_vals,p_names)):
    for j in range(n):
        for k in range(p):
            # Calculating distance between all observations for a disti
ance type
            if j != k: dist_mtx[i][j][k] = f_minowski_distance(obs[j],
obs[k], p_val)

        #Identifying the base observation for measuring distances with
other observations
        obs_base = j + 1

        #Calculating shortest distance and closest observation
        obs_closest  = np.nanargmin(dist_mtx[i,j]) + 1
        dist_min     = np.nanmin(dist_mtx[i,j])

        #Calculating largest distance and farthest observation
        obs_farhtest = np.nanargmax(dist_mtx[i,j]) + 1
        dist_max     = np.nanmax(dist_mtx[i,j])

        # Displaying Result Findings
        msg = p_names[i] + ' Distance: For Observation ' + str(obs_bas
e) + ', closest observation is ' + str(obs_closest) + ' with distance
= ' + str(dist_min) + '.'
        print(msg)
        msg = p_names[i] + ' Distance: For Observation ' + str(obs_bas
e) + ', farhtest observation is ' + str(obs_farhtest) + ' with distanc
e = ' + str(dist_max) + '.'
        print(msg)
```

```
Euclidean Distance: For Observation 1, closest observation is 3 with
distance = 2.383.
Euclidean Distance: For Observation 1, farhtest observation is 4 with
distance = 4.056.
Euclidean Distance: For Observation 2, closest observation is 3 with
distance = 1.373.
Euclidean Distance: For Observation 2, farhtest observation is 4 with
distance = 4.106.
Euclidean Distance: For Observation 3, closest observation is 2 with
distance = 1.373.
Euclidean Distance: For Observation 3, farhtest observation is 4 with
distance = 3.01.
Euclidean Distance: For Observation 4, closest observation is 3 with
distance = 3.01.
Euclidean Distance: For Observation 4, farhtest observation is 2 with
distance = 4.106.
Euclidean Distance: For Observation 5, closest observation is 3 with
distance = 2.473.
Euclidean Distance: For Observation 5, farhtest observation is 1 with
distance = 3.174.
Manhattan Distance: For Observation 1, closest observation is 3 with
distance = 3.499.
Manhattan Distance: For Observation 1, farhtest observation is 4 with
distance = 7.924.
Manhattan Distance: For Observation 2, closest observation is 3 with
distance = 2.305.
Manhattan Distance: For Observation 2, farhtest observation is 4 with
distance = 7.286.
Manhattan Distance: For Observation 3, closest observation is 2 with
distance = 2.305.
Manhattan Distance: For Observation 3, farhtest observation is 4 with
distance = 5.851.
Manhattan Distance: For Observation 4, closest observation is 5 with
distance = 4.93.
Manhattan Distance: For Observation 4, farhtest observation is 1 with
distance = 7.924.
Manhattan Distance: For Observation 5, closest observation is 3 with
distance = 4.657.
Manhattan Distance: For Observation 5, farhtest observation is 1 with
distance = 5.708.
Minowski = 0.5 Distance: For Observation 1, closest observation is 3
with distance = 10.913.
Minowski = 0.5 Distance: For Observation 1, farhtest observation is 4
with distance = 35.437.
Minowski = 0.5 Distance: For Observation 2, closest observation is 3
with distance = 7.991.
Minowski = 0.5 Distance: For Observation 2, farhtest observation is 4
with distance = 27.952.
Minowski = 0.5 Distance: For Observation 3, closest observation is 2
with distance = 7.991.
Minowski = 0.5 Distance: For Observation 3, farhtest observation is 4
with distance = 26.235.
Minowski = 0.5 Distance: For Observation 4, closest observation is 5
with distance = 17.425.
Minowski = 0.5 Distance: For Observation 4, farhtest observation is 1
with distance = 35.437.
```

```
Minowski = 0.5 Distance: For Observation 5, closest observation is 4
with distance = 17.425.
Minowski = 0.5 Distance: For Observation 5, farhtest observation is 1
with distance = 22.832.
```

In [718]:
```
dist_mtx
```

Out[718]:
```
array([[[   nan,  3.001,  2.383,  4.056,  3.174],
        [ 3.001,    nan,  1.373,  4.106,  2.623],
        [ 2.383,  1.373,    nan,  3.01 ,  2.473],
        [ 4.056,  4.106,  3.01 ,    nan,  3.021],
        [ 3.174,  2.623,  2.473,  3.021,    nan]],

       [[   nan,  4.543,  3.499,  7.924,  5.708],
        [ 4.543,    nan,  2.305,  7.286,  4.834],
        [ 3.499,  2.305,    nan,  5.851,  4.657],
        [ 7.924,  7.286,  5.851,    nan,  4.93 ],
        [ 5.708,  4.834,  4.657,  4.93 ,    nan]],

       [[   nan, 13.523, 10.913, 35.437, 22.832],
        [13.523,    nan,  7.991, 27.952, 20.184],
        [10.913,  7.991,    nan, 26.235, 19.715],
        [35.437, 27.952, 26.235,    nan, 17.425],
        [22.832, 20.184, 19.715, 17.425,    nan]]])
```

1.3 Create similarity matrices for these five (5) observations with the following kernels.

Print each similarity matrix. Which points are most similar to each other?

Are they the same with each kernel?

Are they the same points that were closest in question two (2)? (15 pts)

1.3.1 Gaussian kernel (use gamma = 1.0)

1.3.2 Laplace kernel (use gamma = 1.0)

1.3.3 Cosine kernel

In [719]:
```
def f_kernel(arg_kernel_type, v1, v2, y):

    kernel_type = arg_kernel_type.upper()
    if kernel_type == 'RBF':
        power = 2
        s = sum ( abs(a - b)**power for a, b in zip(v1, v2) )
        ret_value = mth.exp(-1 * y * s)
    elif kernel_type == 'LAPLACIAN':
        power = 1
        s = sum ( abs(a - b)**power for a, b in zip(v1, v2) )
        ret_value = mth.exp(-1 * y * s)
    elif kernel_type == 'GAUSSIAN':
        ret_value = np.exp(-np.linalg.norm(v1-v2, 2)**2/(2.*y**2))
    elif kernel_type == 'COSINE':
        ret_value =  1 - cosine(v1, v2)
    else:
        ret_value = None
    return (ret_value)
```

In [720]:
```
round(f_kernel('RBF', obs[0], obs[1], 1), 4)
```

Out[720]: 0.0001

In [721]:
```
round(f_kernel('GAUSSIAN', obs[0], obs[1], 1), 4)
```

Out[721]: 0.0111

In [722]:
```
round(f_kernel('LAPLACIAN', obs[0], obs[1], 1), 4)
```

Out[722]: 0.0106

In [723]:
```
round(f_kernel('COSINE', obs[0], obs[1], 1),4)
```

Out[723]: 0.2622

In [724]:
```
# Creating a 3 dimensional matrix of 3,5,5
sim_mtx = np.vectorize(float)(np.arange(3*n*p).reshape(3,n,p))
sim_mtx[:] = np.NaN
```

In [725]:
```
p_vals = [1, 1, 1]
p_names = ['GAUSSIAN', 'LAPLACIAN', 'COSINE']
```

In [726]:
```
sim_value = 0
X = p_vals[0]
Y = p_names[0]
for i, (p_val, p_name) in enumerate(zip(p_vals, p_names)):
    for j in range(n):
        for k in range(p):
            # Calculating Similarity between all observations for a Sim
ilarity type
            if j != k: sim_mtx[i][j][k] = round(f_kernel(p_name, obs
[j], obs[k], p_val), 5)
            #print(i, p_name, 'Distance between Observations : ', j+1,
' and ',k+1, ' is ', sim_mtx[i][j][k])

        #Identifying the base observation for measuring similarity wit
h other observations
        obs_base = j + 1

        #Calculating shortest similarity and closest observation
        obs_closest  = np.nanargmin(sim_mtx[i,j]) + 1
        dist_min     = np.nanmin(sim_mtx[i,j])

        #Calculating largest distance and farthest observation
        obs_farhtest = np.nanargmax(sim_mtx[i,j]) + 1
        dist_max     = np.nanmax(sim_mtx[i,j])

        # Displaying Result Findings
        msg = p_names[i] + ' Kernel: For Observation ' + str(obs_base)
+ ', closest observation is ' + str(obs_closest) + ' with similarity =
' + str(dist_min) + '.'
        print(msg)
        msg = p_names[i] + ' Kernel: For Observation ' + str(obs_base)
+ ', farhtest observation is ' + str(obs_farhtest) + ' with similarity
= ' + str(dist_max) + '.'
        print(msg)
```

```
GAUSSIAN Kernel: For Observation 1, closest observation is 4 with sim
ilarity = 0.00027.
GAUSSIAN Kernel: For Observation 1, farhtest observation is 3 with si
milarity = 0.05851.
GAUSSIAN Kernel: For Observation 2, closest observation is 4 with sim
ilarity = 0.00022.
GAUSSIAN Kernel: For Observation 2, farhtest observation is 3 with si
milarity = 0.38977.
GAUSSIAN Kernel: For Observation 3, closest observation is 4 with sim
ilarity = 0.01077.
GAUSSIAN Kernel: For Observation 3, farhtest observation is 2 with si
milarity = 0.38977.
GAUSSIAN Kernel: For Observation 4, closest observation is 2 with sim
ilarity = 0.00022.
GAUSSIAN Kernel: For Observation 4, farhtest observation is 3 with si
milarity = 0.01077.
GAUSSIAN Kernel: For Observation 5, closest observation is 1 with sim
ilarity = 0.00649.
GAUSSIAN Kernel: For Observation 5, farhtest observation is 3 with si
milarity = 0.04694.
LAPLACIAN Kernel: For Observation 1, closest observation is 4 with si
milarity = 0.00036.
LAPLACIAN Kernel: For Observation 1, farhtest observation is 3 with s
imilarity = 0.03021.
LAPLACIAN Kernel: For Observation 2, closest observation is 4 with si
milarity = 0.00068.
LAPLACIAN Kernel: For Observation 2, farhtest observation is 3 with s
imilarity = 0.09971.
LAPLACIAN Kernel: For Observation 3, closest observation is 4 with si
milarity = 0.00288.
LAPLACIAN Kernel: For Observation 3, farhtest observation is 2 with s
imilarity = 0.09971.
LAPLACIAN Kernel: For Observation 4, closest observation is 1 with si
milarity = 0.00036.
LAPLACIAN Kernel: For Observation 4, farhtest observation is 5 with s
imilarity = 0.00722.
LAPLACIAN Kernel: For Observation 5, closest observation is 1 with si
milarity = 0.00332.
LAPLACIAN Kernel: For Observation 5, farhtest observation is 3 with s
imilarity = 0.00949.
COSINE Kernel: For Observation 1, closest observation is 4 with simil
arity = 0.01449.
COSINE Kernel: For Observation 1, farhtest observation is 3 with simi
larity = 0.61148.
COSINE Kernel: For Observation 2, closest observation is 4 with simil
arity = -0.66444.
COSINE Kernel: For Observation 2, farhtest observation is 3 with simi
larity = 0.61539.
COSINE Kernel: For Observation 3, closest observation is 5 with simil
arity = -0.64478.
COSINE Kernel: For Observation 3, farhtest observation is 2 with simi
larity = 0.61539.
COSINE Kernel: For Observation 4, closest observation is 2 with simil
arity = -0.66444.
COSINE Kernel: For Observation 4, farhtest observation is 3 with simi
larity = 0.10452.
```

```
COSINE Kernel: For Observation 5, closest observation is 3 with simil
arity = -0.64478.
COSINE Kernel: For Observation 5, farhtest observation is 4 with simi
larity = 0.05974.
```

In [727]:
```
sim_mtx.round(3)
```

Out[727]:
```
array([[[   nan,  0.011,  0.059,  0.   ,  0.006],
        [ 0.011,    nan,  0.39 ,  0.   ,  0.032],
        [ 0.059,  0.39 ,    nan,  0.011,  0.047],
        [ 0.   ,  0.   ,  0.011,    nan,  0.01 ],
        [ 0.006,  0.032,  0.047,  0.01 ,    nan]],

       [[   nan,  0.011,  0.03 ,  0.   ,  0.003],
        [ 0.011,    nan,  0.1  ,  0.001,  0.008],
        [ 0.03 ,  0.1  ,    nan,  0.003,  0.009],
        [ 0.   ,  0.001,  0.003,    nan,  0.007],
        [ 0.003,  0.008,  0.009,  0.007,    nan]],

       [[   nan,  0.262,  0.611,  0.014,  0.057],
        [ 0.262,    nan,  0.615, -0.664, -0.555],
        [ 0.611,  0.615,    nan,  0.105, -0.645],
        [ 0.014, -0.664,  0.105,    nan,  0.06 ],
        [ 0.057, -0.555, -0.645,  0.06 ,    nan]]])
```

Part Two: K-NN and Kernel Smoothing Methods

For this problem we will fit a K-nearest neighbor regression model and kernel smoothing regression model to predict the sales price (Sale_Price) of a home from the following variables: Lot_Area, Year_Built, Gr_Liv_Area, Total_Bsmt_SF, Full_Bath

In [728]:
```
ames_df[col_list].head()
```

Out[728]:

	Lot_Area	Year_Built	Gr_Liv_Area	Total_Bsmt_SF	Full_Bath	Sale_Price
0	31770	1960	1656	1080	1	215000
1	11622	1961	896	882	1	105000
2	14267	1958	1329	1329	1	172000
3	11160	1968	2110	2110	2	244000
4	13830	1997	1629	928	2	189900

2.1 Create a data matrix for the explanatory variables and vector for the response. Print the first few rows of the feature matrix. (5 pts)

In [729]:
```
X = ames_df[col_list[0:5]].values
y = ames_df[['Sale_Price']].values
```

In [730]:
```
X[0:5, :]
```

Out[730]:
```
array([[31770,  1960,  1656,  1080,     1],
       [11622,  1961,   896,   882,     1],
       [14267,  1958,  1329,  1329,     1],
       [11160,  1968,  2110,  2110,     2],
       [13830,  1997,  1629,   928,     2]], dtype=int64)
```

2.2 Randomly split the data into 70% training and 30% test data. Print the dimensions of each matrix/vector. (5 pts)

In [731]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.
3,  random_state=1)
```

In [732]:
```
X_train.shape
```

Out[732]:
```
(2051, 5)
```

In [733]:
```
y_train.shape
```

Out[733]:
```
(2051, 1)
```

In [734]:
```
X_test.shape
```

Out[734]:
```
(879, 5)
```

In [735]:
```
y_test.shape
```

Out[735]:
```
(879, 1)
```

In [736]:
```
y_train = y_train.ravel()
```

In [737]:
```
y_test = y_test.ravel()
```

2.3 Standardize the training data and apply the same transformation to the test data. Print the first few rows of the training feature matrix. (10 pts)

Note: You do not need to standardize the response variable, Sale_Price.

In [738]:
```
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
scaler.fit(X_test)
X_test = scaler.transform(X_test)
```

In [739]:
```
X_train[0:5,]
```

Out[739]:
```
array([[ 0.08940132, -0.20356345, -0.68571107,  0.20488523, -1.069805
98],
       [-0.21291426,  0.75372232, -0.06057701, -1.48213657,  0.740567
01],
       [ 0.13689057, -2.05211527, -0.45323934, -0.97383909, -1.069805
98],
       [-0.29308297,  0.75372232, -0.49231022,  0.38673303,  0.740567
01],
       [ 0.14753893, -0.36861272,  0.13868448,  0.32319585, -1.069805
98]])
```

In [740]:
```
X_test[0:5,]
```

Out[740]:
```
array([[-0.25112638,  0.81702292, -1.01532502, -0.09892942, -0.927073
38],
       [-0.2898994 , -1.58062832,  1.18568481,  0.19496866, -0.927073
38],
       [-0.43001415,  1.18333075, -0.24354235,  0.8425407 ,  0.891894
26],
       [-0.48894354,  1.3165336 , -0.26395988,  0.81763408,  0.891894
26],
       [-0.21837228,  0.91692506,  0.71403985, -0.22097185,  0.891894
26]])
```

2.4 Fit a K-nearest neighbor regression model on the training data using Euclidean distance. Use 5-fold cross validation to choose the best K in terms of RSME. Report the CV error for each choice of K considered. (20 pts)

Hint: Consider various values of K between 5 and 20.

In [741]:
```
folds = 5
cv = KFold(folds, shuffle = True)
```

In [742]:
```
k_list = list(range(5,21,2))
k_count = len(k_list)
k_list
```

Out[742]:
```
[5, 7, 9, 11, 13, 15, 17, 19]
```

In [743]:
```
train_KNN_RMSEs       = np.empty((k_count, folds))
```

In [744]:
```
# ------------------------------------------------------------
#   Looping Through 5 folds for KNN Model
# ------------------------------------------------------------
for i, k in enumerate(k_list):
    for j, (train_index, test_index) in enumerate(cv.split(X_train)):
        knn = KNeighborsClassifier(n_neighbors = k, p = 2)

        # Fitting the kNN Models based on features
        knn.fit(X_train[train_index], y_train[train_index])
        knn_score_dict[k] = knn.score(X_train[train_index], y_train[tr
ain_index])

        # Doing predictions on training and testing data-sets for each
fold
        y_train_pred = knn.predict(X_train[train_index])

        # Calculating RMSE for training and testing data-sets for each
fold
        train_KNN_RMSEs[i,j]       = mth.sqrt(mean_squared_error(y_tra
in[train_index], y_train_pred))

        print('For neighbors k = ', k, 'and fold =', j + 1, ', score i
s', round(train_KNN_RMSEs[i,j] , 6))
```

```
For neighbors k =  5 and fold = 1 , score is 49984.055582
For neighbors k =  5 and fold = 2 , score is 49671.318317
For neighbors k =  5 and fold = 3 , score is 47490.233056
For neighbors k =  5 and fold = 4 , score is 51161.234243
For neighbors k =  5 and fold = 5 , score is 50374.794934
For neighbors k =  7 and fold = 1 , score is 55054.493093
For neighbors k =  7 and fold = 2 , score is 52859.009956
For neighbors k =  7 and fold = 3 , score is 56242.014397
For neighbors k =  7 and fold = 4 , score is 52985.341683
For neighbors k =  7 and fold = 5 , score is 56496.166476
For neighbors k =  9 and fold = 1 , score is 55637.808715
For neighbors k =  9 and fold = 2 , score is 58873.904634
For neighbors k =  9 and fold = 3 , score is 60214.903152
For neighbors k =  9 and fold = 4 , score is 56390.541752
For neighbors k =  9 and fold = 5 , score is 60607.391322
For neighbors k =  11 and fold = 1 , score is 61887.114025
For neighbors k =  11 and fold = 2 , score is 55741.887763
For neighbors k =  11 and fold = 3 , score is 60278.346271
For neighbors k =  11 and fold = 4 , score is 60750.92953
For neighbors k =  11 and fold = 5 , score is 61745.672095
For neighbors k =  13 and fold = 1 , score is 60746.498621
For neighbors k =  13 and fold = 2 , score is 58946.669303
For neighbors k =  13 and fold = 3 , score is 62138.703042
For neighbors k =  13 and fold = 4 , score is 60559.36603
For neighbors k =  13 and fold = 5 , score is 61351.011217
For neighbors k =  15 and fold = 1 , score is 61064.917788
For neighbors k =  15 and fold = 2 , score is 61764.153404
For neighbors k =  15 and fold = 3 , score is 60391.436915
For neighbors k =  15 and fold = 4 , score is 60491.519368
For neighbors k =  15 and fold = 5 , score is 61908.318072
For neighbors k =  17 and fold = 1 , score is 62115.410826
For neighbors k =  17 and fold = 2 , score is 58455.044366
For neighbors k =  17 and fold = 3 , score is 59815.302634
For neighbors k =  17 and fold = 4 , score is 64881.230392
For neighbors k =  17 and fold = 5 , score is 60251.760981
For neighbors k =  19 and fold = 1 , score is 58375.623591
For neighbors k =  19 and fold = 2 , score is 60617.725171
For neighbors k =  19 and fold = 3 , score is 61377.680849
For neighbors k =  19 and fold = 4 , score is 62269.607197
For neighbors k =  19 and fold = 5 , score is 61718.100696
```

In [745]:
```
#Collecting Minimum Score Information
RMSE_KNN_min = train_KNN_RMSEs.min()
RMSE_KNN_min_loc = np.argmin(train_KNN_RMSEs)

k_KNN_min = k_list[int(RMSE_KNN_min_loc/k_count)]
fold_KNN_min = (RMSE_KNN_min_loc%k_count) + 1
msg = 'Lowest RMSE Score ' + str(round(RMSE_KNN_min, 4)) + ' was found
for k = ' + str(k_KNN_min) + ' during fold = ' + str(fold_KNN_min) +
'.'
print(msg)
```

```
Lowest RMSE Score 47490.2331 was found for k = 5 during fold = 3.
```

In [746]:
```
for i, k  in enumerate(k_list):
    print('For k =', k, 'Lowest RMSE was', round(train_KNN_RMSEs[i,:].
mean(),4))
```

```
For k = 5 Lowest RMSE was 49736.3272
For k = 7 Lowest RMSE was 54727.4051
For k = 9 Lowest RMSE was 58344.9099
For k = 11 Lowest RMSE was 60080.7899
For k = 13 Lowest RMSE was 60748.4496
For k = 15 Lowest RMSE was 61124.0691
For k = 17 Lowest RMSE was 61103.7498
For k = 19 Lowest RMSE was 60871.7475
```

2.5 Implement and fit a kernel smoothed regression model using the Laplace kernel. Use 5-fold cross validation to choose the best tuning parameter gamma in terms of RSME. Report the average CV error for each choice of gamma considered. (20 pts)

Hint: Consider various values of gamma between 0.10 and 10.0.

You do not need to tune the number of neighbors; choose a large value for this and let gamma do the work.

In [747]:
```
g_list = np.arange(0.10 ,10.1, 0.10)
g_list = g_list.round(1)
g_count = len(g_list)
g_list
```

Out[747]:
```
array([ 0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9,  1. ,
1.1,
        1.2,  1.3,  1.4,  1.5,  1.6,  1.7,  1.8,  1.9,  2. ,  2.1,
2.2,
        2.3,  2.4,  2.5,  2.6,  2.7,  2.8,  2.9,  3. ,  3.1,  3.2,
3.3,
        3.4,  3.5,  3.6,  3.7,  3.8,  3.9,  4. ,  4.1,  4.2,  4.3,
4.4,
        4.5,  4.6,  4.7,  4.8,  4.9,  5. ,  5.1,  5.2,  5.3,  5.4,
5.5,
        5.6,  5.7,  5.8,  5.9,  6. ,  6.1,  6.2,  6.3,  6.4,  6.5,
6.6,
        6.7,  6.8,  6.9,  7. ,  7.1,  7.2,  7.3,  7.4,  7.5,  7.6,
7.7,
        7.8,  7.9,  8. ,  8.1,  8.2,  8.3,  8.4,  8.5,  8.6,  8.7,
8.8,
        8.9,  9. ,  9.1,  9.2,  9.3,  9.4,  9.5,  9.6,  9.7,  9.8,
9.9,
       10. ])
```

In [748]:
```
train_KER_scores      = np.empty((g_count, folds))
train_KER_RMSEs       = np.empty((g_count, folds))
train_KER_scores.shape
```

Out[748]:
```
(100, 5)
```

In [749]:
```
def laplacian(gamma):
    def kernel(distance):
        return np.exp(-gamma*distance**1)
    return (kernel)
```

In [750]:
```
# ------------------------------------------------------------
#   Looping Through 5 folds for Smoothed Kernel Model
# ------------------------------------------------------------
for i, g in enumerate(g_list):
    for j, (train_index, test_index) in enumerate(cv.split(X_train)):
        ks = KNeighborsClassifier(n_neighbors = 50, weights = laplacia
n(gamma = g))

        # Fitting the kNN Models based on features
        ks.fit(X_train[train_index], y_train[train_index])

        # Doing predictions on training and testing data-sets for each
fold
        y_train_pred = ks.predict(X_train[train_index])

        # Calculating RMSE for training and testing data-sets for each
fold
        train_KER_RMSEs[i,j]       = mth.sqrt(mean_squared_error(y_tra
in[train_index], y_train_pred))

        print('For gamma = ', g, 'and fold =', j + 1, ', RMSE is', rou
nd(train_KER_RMSEs[i,j] , 6))
```

```
For gamma =  0.1 and fold = 1 , RMSE is 43414.936357
For gamma =  0.1 and fold = 2 , RMSE is 43973.739165
For gamma =  0.1 and fold = 3 , RMSE is 44364.861951
For gamma =  0.1 and fold = 4 , RMSE is 43563.436463
For gamma =  0.1 and fold = 5 , RMSE is 42146.436115
For gamma =  0.2 and fold = 1 , RMSE is 40645.652885
For gamma =  0.2 and fold = 2 , RMSE is 40820.081316
For gamma =  0.2 and fold = 3 , RMSE is 42051.752764
For gamma =  0.2 and fold = 4 , RMSE is 39936.278755
For gamma =  0.2 and fold = 5 , RMSE is 39219.803026
For gamma =  0.3 and fold = 1 , RMSE is 38809.605005
For gamma =  0.3 and fold = 2 , RMSE is 39158.955104
For gamma =  0.3 and fold = 3 , RMSE is 40698.225875
For gamma =  0.3 and fold = 4 , RMSE is 39304.141733
For gamma =  0.3 and fold = 5 , RMSE is 39551.57505
For gamma =  0.4 and fold = 1 , RMSE is 37329.426854
For gamma =  0.4 and fold = 2 , RMSE is 35712.378178
For gamma =  0.4 and fold = 3 , RMSE is 38021.863468
For gamma =  0.4 and fold = 4 , RMSE is 36389.981106
For gamma =  0.4 and fold = 5 , RMSE is 35767.638519
For gamma =  0.5 and fold = 1 , RMSE is 34941.303477
For gamma =  0.5 and fold = 2 , RMSE is 35327.326551
For gamma =  0.5 and fold = 3 , RMSE is 35714.017809
For gamma =  0.5 and fold = 4 , RMSE is 35395.360304
For gamma =  0.5 and fold = 5 , RMSE is 34525.774586
For gamma =  0.6 and fold = 1 , RMSE is 33208.661562
For gamma =  0.6 and fold = 2 , RMSE is 35082.050588
For gamma =  0.6 and fold = 3 , RMSE is 33066.050194
For gamma =  0.6 and fold = 4 , RMSE is 33179.485707
For gamma =  0.6 and fold = 5 , RMSE is 32994.875704
For gamma =  0.7 and fold = 1 , RMSE is 31956.466182
For gamma =  0.7 and fold = 2 , RMSE is 32986.394805
For gamma =  0.7 and fold = 3 , RMSE is 30673.050228
For gamma =  0.7 and fold = 4 , RMSE is 32970.844569
For gamma =  0.7 and fold = 5 , RMSE is 31876.236903
For gamma =  0.8 and fold = 1 , RMSE is 31520.50537
For gamma =  0.8 and fold = 2 , RMSE is 32973.302447
For gamma =  0.8 and fold = 3 , RMSE is 31346.802229
For gamma =  0.8 and fold = 4 , RMSE is 29111.547637
For gamma =  0.8 and fold = 5 , RMSE is 28914.301709
For gamma =  0.9 and fold = 1 , RMSE is 28707.856094
For gamma =  0.9 and fold = 2 , RMSE is 29150.525935
For gamma =  0.9 and fold = 3 , RMSE is 27079.126351
For gamma =  0.9 and fold = 4 , RMSE is 28293.826567
For gamma =  0.9 and fold = 5 , RMSE is 27905.688135
For gamma =  1.0 and fold = 1 , RMSE is 27105.399843
For gamma =  1.0 and fold = 2 , RMSE is 27768.935675
For gamma =  1.0 and fold = 3 , RMSE is 26379.903305
For gamma =  1.0 and fold = 4 , RMSE is 26500.590307
For gamma =  1.0 and fold = 5 , RMSE is 27985.911767
For gamma =  1.1 and fold = 1 , RMSE is 27107.855445
For gamma =  1.1 and fold = 2 , RMSE is 26431.209305
For gamma =  1.1 and fold = 3 , RMSE is 26305.743318
For gamma =  1.1 and fold = 4 , RMSE is 25612.216708
For gamma =  1.1 and fold = 5 , RMSE is 26051.965276
For gamma =  1.2 and fold = 1 , RMSE is 25060.466441
```

```
For gamma =  1.2 and fold = 2 , RMSE is 25476.782702
For gamma =  1.2 and fold = 3 , RMSE is 25104.500698
For gamma =  1.2 and fold = 4 , RMSE is 24983.664134
For gamma =  1.2 and fold = 5 , RMSE is 25155.6799
For gamma =  1.3 and fold = 1 , RMSE is 24454.713175
For gamma =  1.3 and fold = 2 , RMSE is 24000.424644
For gamma =  1.3 and fold = 3 , RMSE is 24124.917888
For gamma =  1.3 and fold = 4 , RMSE is 23560.761146
For gamma =  1.3 and fold = 5 , RMSE is 25095.227719
For gamma =  1.4 and fold = 1 , RMSE is 23828.767889
For gamma =  1.4 and fold = 2 , RMSE is 23766.343184
For gamma =  1.4 and fold = 3 , RMSE is 23947.680304
For gamma =  1.4 and fold = 4 , RMSE is 22926.293086
For gamma =  1.4 and fold = 5 , RMSE is 22759.800565
For gamma =  1.5 and fold = 1 , RMSE is 22295.428324
For gamma =  1.5 and fold = 2 , RMSE is 24227.839827
For gamma =  1.5 and fold = 3 , RMSE is 22967.954798
For gamma =  1.5 and fold = 4 , RMSE is 22814.853721
For gamma =  1.5 and fold = 5 , RMSE is 23814.532497
For gamma =  1.6 and fold = 1 , RMSE is 22786.460924
For gamma =  1.6 and fold = 2 , RMSE is 22471.106887
For gamma =  1.6 and fold = 3 , RMSE is 22385.665715
For gamma =  1.6 and fold = 4 , RMSE is 21655.305669
For gamma =  1.6 and fold = 5 , RMSE is 21810.397475
For gamma =  1.7 and fold = 1 , RMSE is 22768.259406
For gamma =  1.7 and fold = 2 , RMSE is 22461.266957
For gamma =  1.7 and fold = 3 , RMSE is 20534.582286
For gamma =  1.7 and fold = 4 , RMSE is 21463.442359
For gamma =  1.7 and fold = 5 , RMSE is 21609.09271
For gamma =  1.8 and fold = 1 , RMSE is 20488.674465
For gamma =  1.8 and fold = 2 , RMSE is 21790.465454
For gamma =  1.8 and fold = 3 , RMSE is 19085.88302
For gamma =  1.8 and fold = 4 , RMSE is 21938.54851
For gamma =  1.8 and fold = 5 , RMSE is 20699.565759
For gamma =  1.9 and fold = 1 , RMSE is 20528.288149
For gamma =  1.9 and fold = 2 , RMSE is 21253.791193
For gamma =  1.9 and fold = 3 , RMSE is 18903.562768
For gamma =  1.9 and fold = 4 , RMSE is 21089.065683
For gamma =  1.9 and fold = 5 , RMSE is 20509.634121
For gamma =  2.0 and fold = 1 , RMSE is 19125.066547
For gamma =  2.0 and fold = 2 , RMSE is 19573.222305
For gamma =  2.0 and fold = 3 , RMSE is 19927.088499
For gamma =  2.0 and fold = 4 , RMSE is 21052.267523
For gamma =  2.0 and fold = 5 , RMSE is 20809.743239
For gamma =  2.1 and fold = 1 , RMSE is 20227.483495
For gamma =  2.1 and fold = 2 , RMSE is 19998.756778
For gamma =  2.1 and fold = 3 , RMSE is 18858.159531
For gamma =  2.1 and fold = 4 , RMSE is 19453.956574
For gamma =  2.1 and fold = 5 , RMSE is 19341.913306
For gamma =  2.2 and fold = 1 , RMSE is 18930.708873
For gamma =  2.2 and fold = 2 , RMSE is 17841.356126
For gamma =  2.2 and fold = 3 , RMSE is 18304.195694
For gamma =  2.2 and fold = 4 , RMSE is 18858.948126
For gamma =  2.2 and fold = 5 , RMSE is 17888.389777
For gamma =  2.3 and fold = 1 , RMSE is 17280.344576
For gamma =  2.3 and fold = 2 , RMSE is 16895.357545
```

```
For gamma =  2.3 and fold = 3 , RMSE is 19589.265173
For gamma =  2.3 and fold = 4 , RMSE is 18334.015663
For gamma =  2.3 and fold = 5 , RMSE is 17767.83967
For gamma =  2.4 and fold = 1 , RMSE is 17797.19417
For gamma =  2.4 and fold = 2 , RMSE is 18106.546462
For gamma =  2.4 and fold = 3 , RMSE is 17136.99376
For gamma =  2.4 and fold = 4 , RMSE is 16515.276723
For gamma =  2.4 and fold = 5 , RMSE is 17076.437619
For gamma =  2.5 and fold = 1 , RMSE is 15158.870961
For gamma =  2.5 and fold = 2 , RMSE is 17671.605734
For gamma =  2.5 and fold = 3 , RMSE is 16931.79411
For gamma =  2.5 and fold = 4 , RMSE is 17744.203271
For gamma =  2.5 and fold = 5 , RMSE is 15975.675743
For gamma =  2.6 and fold = 1 , RMSE is 16327.183953
For gamma =  2.6 and fold = 2 , RMSE is 16778.273273
For gamma =  2.6 and fold = 3 , RMSE is 16761.977818
For gamma =  2.6 and fold = 4 , RMSE is 15437.294242
For gamma =  2.6 and fold = 5 , RMSE is 16097.975589
For gamma =  2.7 and fold = 1 , RMSE is 15852.011289
For gamma =  2.7 and fold = 2 , RMSE is 16438.011359
For gamma =  2.7 and fold = 3 , RMSE is 15397.97973
For gamma =  2.7 and fold = 4 , RMSE is 16570.665614
For gamma =  2.7 and fold = 5 , RMSE is 15812.523574
For gamma =  2.8 and fold = 1 , RMSE is 13456.538796
For gamma =  2.8 and fold = 2 , RMSE is 15616.133144
For gamma =  2.8 and fold = 3 , RMSE is 15856.46782
For gamma =  2.8 and fold = 4 , RMSE is 14586.793564
For gamma =  2.8 and fold = 5 , RMSE is 15674.464832
For gamma =  2.9 and fold = 1 , RMSE is 14201.101898
For gamma =  2.9 and fold = 2 , RMSE is 14593.375015
For gamma =  2.9 and fold = 3 , RMSE is 15881.125436
For gamma =  2.9 and fold = 4 , RMSE is 14252.85639
For gamma =  2.9 and fold = 5 , RMSE is 14318.403279
For gamma =  3.0 and fold = 1 , RMSE is 14384.932554
For gamma =  3.0 and fold = 2 , RMSE is 12522.213753
For gamma =  3.0 and fold = 3 , RMSE is 15581.007278
For gamma =  3.0 and fold = 4 , RMSE is 14659.672418
For gamma =  3.0 and fold = 5 , RMSE is 15069.85466
For gamma =  3.1 and fold = 1 , RMSE is 13096.30223
For gamma =  3.1 and fold = 2 , RMSE is 14410.557844
For gamma =  3.1 and fold = 3 , RMSE is 13325.449264
For gamma =  3.1 and fold = 4 , RMSE is 14893.496788
For gamma =  3.1 and fold = 5 , RMSE is 13342.440942
For gamma =  3.2 and fold = 1 , RMSE is 13038.172466
For gamma =  3.2 and fold = 2 , RMSE is 12394.885974
For gamma =  3.2 and fold = 3 , RMSE is 14196.359598
For gamma =  3.2 and fold = 4 , RMSE is 11260.91754
For gamma =  3.2 and fold = 5 , RMSE is 14367.212852
For gamma =  3.3 and fold = 1 , RMSE is 11170.588364
For gamma =  3.3 and fold = 2 , RMSE is 13315.349891
For gamma =  3.3 and fold = 3 , RMSE is 12019.26898
For gamma =  3.3 and fold = 4 , RMSE is 12080.236818
For gamma =  3.3 and fold = 5 , RMSE is 13041.155813
For gamma =  3.4 and fold = 1 , RMSE is 9752.293011
For gamma =  3.4 and fold = 2 , RMSE is 12837.625972
For gamma =  3.4 and fold = 3 , RMSE is 12039.723828
```

```
For gamma =  3.4 and fold = 4 , RMSE is 14039.524708
For gamma =  3.4 and fold = 5 , RMSE is 11076.983158
For gamma =  3.5 and fold = 1 , RMSE is 12066.551433
For gamma =  3.5 and fold = 2 , RMSE is 13113.980514
For gamma =  3.5 and fold = 3 , RMSE is 12014.908909
For gamma =  3.5 and fold = 4 , RMSE is 10090.087791
For gamma =  3.5 and fold = 5 , RMSE is 11358.037671
For gamma =  3.6 and fold = 1 , RMSE is 10672.053396
For gamma =  3.6 and fold = 2 , RMSE is 12828.799241
For gamma =  3.6 and fold = 3 , RMSE is 12174.596446
For gamma =  3.6 and fold = 4 , RMSE is 9725.841026
For gamma =  3.6 and fold = 5 , RMSE is 11864.125263
For gamma =  3.7 and fold = 1 , RMSE is 11209.544266
For gamma =  3.7 and fold = 2 , RMSE is 11086.442273
For gamma =  3.7 and fold = 3 , RMSE is 11804.256111
For gamma =  3.7 and fold = 4 , RMSE is 11540.717171
For gamma =  3.7 and fold = 5 , RMSE is 12202.201171
For gamma =  3.8 and fold = 1 , RMSE is 8814.981171
For gamma =  3.8 and fold = 2 , RMSE is 11057.808324
For gamma =  3.8 and fold = 3 , RMSE is 11776.104004
For gamma =  3.8 and fold = 4 , RMSE is 10838.566722
For gamma =  3.8 and fold = 5 , RMSE is 10980.173161
For gamma =  3.9 and fold = 1 , RMSE is 10414.014088
For gamma =  3.9 and fold = 2 , RMSE is 12530.909728
For gamma =  3.9 and fold = 3 , RMSE is 9492.052743
For gamma =  3.9 and fold = 4 , RMSE is 10429.377526
For gamma =  3.9 and fold = 5 , RMSE is 10577.04388
For gamma =  4.0 and fold = 1 , RMSE is 9979.444955
For gamma =  4.0 and fold = 2 , RMSE is 10889.95348
For gamma =  4.0 and fold = 3 , RMSE is 10992.625435
For gamma =  4.0 and fold = 4 , RMSE is 9070.915199
For gamma =  4.0 and fold = 5 , RMSE is 10861.531723
For gamma =  4.1 and fold = 1 , RMSE is 10866.615529
For gamma =  4.1 and fold = 2 , RMSE is 11692.327937
For gamma =  4.1 and fold = 3 , RMSE is 9607.154965
For gamma =  4.1 and fold = 4 , RMSE is 10281.451199
For gamma =  4.1 and fold = 5 , RMSE is 8342.565043
For gamma =  4.2 and fold = 1 , RMSE is 11061.720268
For gamma =  4.2 and fold = 2 , RMSE is 10032.045278
For gamma =  4.2 and fold = 3 , RMSE is 9097.078587
For gamma =  4.2 and fold = 4 , RMSE is 10007.385911
For gamma =  4.2 and fold = 5 , RMSE is 10276.973746
For gamma =  4.3 and fold = 1 , RMSE is 9373.384306
For gamma =  4.3 and fold = 2 , RMSE is 8846.081564
For gamma =  4.3 and fold = 3 , RMSE is 10782.253951
For gamma =  4.3 and fold = 4 , RMSE is 11303.99515
For gamma =  4.3 and fold = 5 , RMSE is 7788.995468
For gamma =  4.4 and fold = 1 , RMSE is 9598.97805
For gamma =  4.4 and fold = 2 , RMSE is 9906.547794
For gamma =  4.4 and fold = 3 , RMSE is 10097.542643
For gamma =  4.4 and fold = 4 , RMSE is 8674.294537
For gamma =  4.4 and fold = 5 , RMSE is 7894.861584
For gamma =  4.5 and fold = 1 , RMSE is 9982.855106
For gamma =  4.5 and fold = 2 , RMSE is 9614.046601
For gamma =  4.5 and fold = 3 , RMSE is 7858.475486
For gamma =  4.5 and fold = 4 , RMSE is 6841.434326
```

```
For gamma =  4.5 and fold = 5 , RMSE is 8366.383924
For gamma =  4.6 and fold = 1 , RMSE is 10520.782176
For gamma =  4.6 and fold = 2 , RMSE is 7434.53887
For gamma =  4.6 and fold = 3 , RMSE is 7933.362496
For gamma =  4.6 and fold = 4 , RMSE is 6857.452812
For gamma =  4.6 and fold = 5 , RMSE is 9654.92287
For gamma =  4.7 and fold = 1 , RMSE is 8839.366365
For gamma =  4.7 and fold = 2 , RMSE is 9551.079589
For gamma =  4.7 and fold = 3 , RMSE is 7850.802615
For gamma =  4.7 and fold = 4 , RMSE is 8009.510664
For gamma =  4.7 and fold = 5 , RMSE is 8517.193308
For gamma =  4.8 and fold = 1 , RMSE is 7764.323352
For gamma =  4.8 and fold = 2 , RMSE is 6853.011144
For gamma =  4.8 and fold = 3 , RMSE is 7951.097485
For gamma =  4.8 and fold = 4 , RMSE is 7773.003206
For gamma =  4.8 and fold = 5 , RMSE is 9169.058501
For gamma =  4.9 and fold = 1 , RMSE is 7509.663789
For gamma =  4.9 and fold = 2 , RMSE is 9403.943327
For gamma =  4.9 and fold = 3 , RMSE is 8010.309573
For gamma =  4.9 and fold = 4 , RMSE is 7506.403173
For gamma =  4.9 and fold = 5 , RMSE is 7718.056828
For gamma =  5.0 and fold = 1 , RMSE is 8484.091128
For gamma =  5.0 and fold = 2 , RMSE is 6116.191765
For gamma =  5.0 and fold = 3 , RMSE is 9009.731056
For gamma =  5.0 and fold = 4 , RMSE is 7824.627344
For gamma =  5.0 and fold = 5 , RMSE is 6051.843759
For gamma =  5.1 and fold = 1 , RMSE is 7280.858055
For gamma =  5.1 and fold = 2 , RMSE is 6261.726847
For gamma =  5.1 and fold = 3 , RMSE is 5597.385542
For gamma =  5.1 and fold = 4 , RMSE is 6955.261801
For gamma =  5.1 and fold = 5 , RMSE is 6410.745328
For gamma =  5.2 and fold = 1 , RMSE is 5431.47351
For gamma =  5.2 and fold = 2 , RMSE is 5906.675886
For gamma =  5.2 and fold = 3 , RMSE is 6557.319156
For gamma =  5.2 and fold = 4 , RMSE is 5467.315398
For gamma =  5.2 and fold = 5 , RMSE is 5353.667414
For gamma =  5.3 and fold = 1 , RMSE is 4917.855852
For gamma =  5.3 and fold = 2 , RMSE is 6345.826886
For gamma =  5.3 and fold = 3 , RMSE is 6016.23897
For gamma =  5.3 and fold = 4 , RMSE is 6097.113032
For gamma =  5.3 and fold = 5 , RMSE is 4901.274359
For gamma =  5.4 and fold = 1 , RMSE is 5598.125911
For gamma =  5.4 and fold = 2 , RMSE is 5778.859688
For gamma =  5.4 and fold = 3 , RMSE is 5725.273309
For gamma =  5.4 and fold = 4 , RMSE is 5519.074885
For gamma =  5.4 and fold = 5 , RMSE is 5017.378905
For gamma =  5.5 and fold = 1 , RMSE is 6208.411139
For gamma =  5.5 and fold = 2 , RMSE is 4904.536783
For gamma =  5.5 and fold = 3 , RMSE is 6047.580559
For gamma =  5.5 and fold = 4 , RMSE is 4753.511733
For gamma =  5.5 and fold = 5 , RMSE is 4979.839705
For gamma =  5.6 and fold = 1 , RMSE is 5447.951845
For gamma =  5.6 and fold = 2 , RMSE is 6036.42052
For gamma =  5.6 and fold = 3 , RMSE is 4638.936938
For gamma =  5.6 and fold = 4 , RMSE is 4787.561386
For gamma =  5.6 and fold = 5 , RMSE is 5822.477454
```

```
For gamma =  5.7 and fold = 1 , RMSE is 5165.969374
For gamma =  5.7 and fold = 2 , RMSE is 5078.557807
For gamma =  5.7 and fold = 3 , RMSE is 4763.742391
For gamma =  5.7 and fold = 4 , RMSE is 5878.416097
For gamma =  5.7 and fold = 5 , RMSE is 6142.60363
For gamma =  5.8 and fold = 1 , RMSE is 4437.497952
For gamma =  5.8 and fold = 2 , RMSE is 4984.298849
For gamma =  5.8 and fold = 3 , RMSE is 5835.13488
For gamma =  5.8 and fold = 4 , RMSE is 4359.982035
For gamma =  5.8 and fold = 5 , RMSE is 4065.695114
For gamma =  5.9 and fold = 1 , RMSE is 4696.004072
For gamma =  5.9 and fold = 2 , RMSE is 5516.650092
For gamma =  5.9 and fold = 3 , RMSE is 5037.00045
For gamma =  5.9 and fold = 4 , RMSE is 4787.1694
For gamma =  5.9 and fold = 5 , RMSE is 5426.354413
For gamma =  6.0 and fold = 1 , RMSE is 5078.03763
For gamma =  6.0 and fold = 2 , RMSE is 5042.714516
For gamma =  6.0 and fold = 3 , RMSE is 4865.824837
For gamma =  6.0 and fold = 4 , RMSE is 4670.193037
For gamma =  6.0 and fold = 5 , RMSE is 4857.766439
For gamma =  6.1 and fold = 1 , RMSE is 5222.266718
For gamma =  6.1 and fold = 2 , RMSE is 5197.074463
For gamma =  6.1 and fold = 3 , RMSE is 3980.830633
For gamma =  6.1 and fold = 4 , RMSE is 5072.732934
For gamma =  6.1 and fold = 5 , RMSE is 5116.224042
For gamma =  6.2 and fold = 1 , RMSE is 5156.898377
For gamma =  6.2 and fold = 2 , RMSE is 4138.792069
For gamma =  6.2 and fold = 3 , RMSE is 4324.310045
For gamma =  6.2 and fold = 4 , RMSE is 4625.887969
For gamma =  6.2 and fold = 5 , RMSE is 4560.452296
For gamma =  6.3 and fold = 1 , RMSE is 4708.224171
For gamma =  6.3 and fold = 2 , RMSE is 4782.383027
For gamma =  6.3 and fold = 3 , RMSE is 4510.664576
For gamma =  6.3 and fold = 4 , RMSE is 4661.809011
For gamma =  6.3 and fold = 5 , RMSE is 3741.871161
For gamma =  6.4 and fold = 1 , RMSE is 2809.98043
For gamma =  6.4 and fold = 2 , RMSE is 4574.233292
For gamma =  6.4 and fold = 3 , RMSE is 4195.952741
For gamma =  6.4 and fold = 4 , RMSE is 4608.481803
For gamma =  6.4 and fold = 5 , RMSE is 4930.614584
For gamma =  6.5 and fold = 1 , RMSE is 4308.281267
For gamma =  6.5 and fold = 2 , RMSE is 4107.777572
For gamma =  6.5 and fold = 3 , RMSE is 4463.724335
For gamma =  6.5 and fold = 4 , RMSE is 4527.605196
For gamma =  6.5 and fold = 5 , RMSE is 4007.285735
For gamma =  6.6 and fold = 1 , RMSE is 4218.666545
For gamma =  6.6 and fold = 2 , RMSE is 4866.303664
For gamma =  6.6 and fold = 3 , RMSE is 3205.867332
For gamma =  6.6 and fold = 4 , RMSE is 4563.654025
For gamma =  6.6 and fold = 5 , RMSE is 4388.237393
For gamma =  6.7 and fold = 1 , RMSE is 4083.601622
For gamma =  6.7 and fold = 2 , RMSE is 2900.955929
For gamma =  6.7 and fold = 3 , RMSE is 4618.480045
For gamma =  6.7 and fold = 4 , RMSE is 4464.320081
For gamma =  6.7 and fold = 5 , RMSE is 3853.047152
For gamma =  6.8 and fold = 1 , RMSE is 4051.158903
```

```
For gamma =  6.8 and fold = 2 , RMSE is 3202.812413
For gamma =  6.8 and fold = 3 , RMSE is 3757.580751
For gamma =  6.8 and fold = 4 , RMSE is 3794.052352
For gamma =  6.8 and fold = 5 , RMSE is 3632.805705
For gamma =  6.9 and fold = 1 , RMSE is 3439.297367
For gamma =  6.9 and fold = 2 , RMSE is 4414.690814
For gamma =  6.9 and fold = 3 , RMSE is 2937.503794
For gamma =  6.9 and fold = 4 , RMSE is 3029.971262
For gamma =  6.9 and fold = 5 , RMSE is 3935.553257
For gamma =  7.0 and fold = 1 , RMSE is 3296.105385
For gamma =  7.0 and fold = 2 , RMSE is 2450.541152
For gamma =  7.0 and fold = 3 , RMSE is 2951.757784
For gamma =  7.0 and fold = 4 , RMSE is 3355.137153
For gamma =  7.0 and fold = 5 , RMSE is 3875.17104
For gamma =  7.1 and fold = 1 , RMSE is 3335.107721
For gamma =  7.1 and fold = 2 , RMSE is 3089.424752
For gamma =  7.1 and fold = 3 , RMSE is 3315.774603
For gamma =  7.1 and fold = 4 , RMSE is 2700.203064
For gamma =  7.1 and fold = 5 , RMSE is 4141.848617
For gamma =  7.2 and fold = 1 , RMSE is 3319.168655
For gamma =  7.2 and fold = 2 , RMSE is 2729.953507
For gamma =  7.2 and fold = 3 , RMSE is 3742.565829
For gamma =  7.2 and fold = 4 , RMSE is 3186.671944
For gamma =  7.2 and fold = 5 , RMSE is 3212.597913
For gamma =  7.3 and fold = 1 , RMSE is 2680.986205
For gamma =  7.3 and fold = 2 , RMSE is 3068.148225
For gamma =  7.3 and fold = 3 , RMSE is 3291.762488
For gamma =  7.3 and fold = 4 , RMSE is 3858.494294
For gamma =  7.3 and fold = 5 , RMSE is 3201.812355
For gamma =  7.4 and fold = 1 , RMSE is 2882.710376
For gamma =  7.4 and fold = 2 , RMSE is 3539.724058
For gamma =  7.4 and fold = 3 , RMSE is 3220.289352
For gamma =  7.4 and fold = 4 , RMSE is 3191.027456
For gamma =  7.4 and fold = 5 , RMSE is 2970.740067
For gamma =  7.5 and fold = 1 , RMSE is 3269.154275
For gamma =  7.5 and fold = 2 , RMSE is 3214.528817
For gamma =  7.5 and fold = 3 , RMSE is 3723.736675
For gamma =  7.5 and fold = 4 , RMSE is 3139.72514
For gamma =  7.5 and fold = 5 , RMSE is 2766.440839
For gamma =  7.6 and fold = 1 , RMSE is 2779.364713
For gamma =  7.6 and fold = 2 , RMSE is 2319.788931
For gamma =  7.6 and fold = 3 , RMSE is 3646.781023
For gamma =  7.6 and fold = 4 , RMSE is 3320.470425
For gamma =  7.6 and fold = 5 , RMSE is 3616.190972
For gamma =  7.7 and fold = 1 , RMSE is 3018.094038
For gamma =  7.7 and fold = 2 , RMSE is 2849.205138
For gamma =  7.7 and fold = 3 , RMSE is 3712.658465
For gamma =  7.7 and fold = 4 , RMSE is 3104.083655
For gamma =  7.7 and fold = 5 , RMSE is 3182.932243
For gamma =  7.8 and fold = 1 , RMSE is 3721.841887
For gamma =  7.8 and fold = 2 , RMSE is 2695.417266
For gamma =  7.8 and fold = 3 , RMSE is 3220.172956
For gamma =  7.8 and fold = 4 , RMSE is 3265.233632
For gamma =  7.8 and fold = 5 , RMSE is 3152.058367
For gamma =  7.9 and fold = 1 , RMSE is 2267.322862
For gamma =  7.9 and fold = 2 , RMSE is 3022.610182
```

```
For gamma =  7.9 and fold = 3 , RMSE is 3298.963274
For gamma =  7.9 and fold = 4 , RMSE is 3139.939193
For gamma =  7.9 and fold = 5 , RMSE is 2949.75283
For gamma =  8.0 and fold = 1 , RMSE is 2817.73004
For gamma =  8.0 and fold = 2 , RMSE is 3090.618262
For gamma =  8.0 and fold = 3 , RMSE is 3120.607971
For gamma =  8.0 and fold = 4 , RMSE is 2960.90075
For gamma =  8.0 and fold = 5 , RMSE is 3151.516196
For gamma =  8.1 and fold = 1 , RMSE is 3202.151843
For gamma =  8.1 and fold = 2 , RMSE is 2827.62286
For gamma =  8.1 and fold = 3 , RMSE is 3734.608451
For gamma =  8.1 and fold = 4 , RMSE is 2844.248798
For gamma =  8.1 and fold = 5 , RMSE is 2457.602418
For gamma =  8.2 and fold = 1 , RMSE is 3080.309814
For gamma =  8.2 and fold = 2 , RMSE is 3054.475078
For gamma =  8.2 and fold = 3 , RMSE is 3459.979355
For gamma =  8.2 and fold = 4 , RMSE is 2963.472141
For gamma =  8.2 and fold = 5 , RMSE is 2479.192859
For gamma =  8.3 and fold = 1 , RMSE is 2696.051761
For gamma =  8.3 and fold = 2 , RMSE is 3223.350662
For gamma =  8.3 and fold = 3 , RMSE is 2471.097582
For gamma =  8.3 and fold = 4 , RMSE is 2641.019245
For gamma =  8.3 and fold = 5 , RMSE is 3356.564443
For gamma =  8.4 and fold = 1 , RMSE is 3548.63351
For gamma =  8.4 and fold = 2 , RMSE is 2263.652713
For gamma =  8.4 and fold = 3 , RMSE is 2150.720489
For gamma =  8.4 and fold = 4 , RMSE is 2671.344685
For gamma =  8.4 and fold = 5 , RMSE is 3591.063331
For gamma =  8.5 and fold = 1 , RMSE is 3121.059567
For gamma =  8.5 and fold = 2 , RMSE is 2571.503156
For gamma =  8.5 and fold = 3 , RMSE is 2961.040304
For gamma =  8.5 and fold = 4 , RMSE is 2751.561482
For gamma =  8.5 and fold = 5 , RMSE is 2637.53319
For gamma =  8.6 and fold = 1 , RMSE is 2377.548564
For gamma =  8.6 and fold = 2 , RMSE is 3029.317318
For gamma =  8.6 and fold = 3 , RMSE is 2529.926325
For gamma =  8.6 and fold = 4 , RMSE is 3447.820906
For gamma =  8.6 and fold = 5 , RMSE is 2893.672076
For gamma =  8.7 and fold = 1 , RMSE is 2973.476145
For gamma =  8.7 and fold = 2 , RMSE is 3337.135068
For gamma =  8.7 and fold = 3 , RMSE is 3511.682883
For gamma =  8.7 and fold = 4 , RMSE is 2516.863279
For gamma =  8.7 and fold = 5 , RMSE is 1916.299216
For gamma =  8.8 and fold = 1 , RMSE is 2704.974193
For gamma =  8.8 and fold = 2 , RMSE is 2494.962293
For gamma =  8.8 and fold = 3 , RMSE is 3387.225698
For gamma =  8.8 and fold = 4 , RMSE is 2078.363475
For gamma =  8.8 and fold = 5 , RMSE is 2937.141215
For gamma =  8.9 and fold = 1 , RMSE is 3389.740822
For gamma =  8.9 and fold = 2 , RMSE is 2199.773302
For gamma =  8.9 and fold = 3 , RMSE is 3112.607764
For gamma =  8.9 and fold = 4 , RMSE is 3151.874404
For gamma =  8.9 and fold = 5 , RMSE is 1661.412247
For gamma =  9.0 and fold = 1 , RMSE is 2986.825966
For gamma =  9.0 and fold = 2 , RMSE is 2083.046897
For gamma =  9.0 and fold = 3 , RMSE is 3174.030154
```

```
For gamma =  9.0 and fold = 4 , RMSE is 2872.200439
For gamma =  9.0 and fold = 5 , RMSE is 2704.170371
For gamma =  9.1 and fold = 1 , RMSE is 2335.161067
For gamma =  9.1 and fold = 2 , RMSE is 2885.725669
For gamma =  9.1 and fold = 3 , RMSE is 1990.527879
For gamma =  9.1 and fold = 4 , RMSE is 2892.561723
For gamma =  9.1 and fold = 5 , RMSE is 3056.273209
For gamma =  9.2 and fold = 1 , RMSE is 2421.687914
For gamma =  9.2 and fold = 2 , RMSE is 2369.876826
For gamma =  9.2 and fold = 3 , RMSE is 2963.941918
For gamma =  9.2 and fold = 4 , RMSE is 2925.841143
For gamma =  9.2 and fold = 5 , RMSE is 3272.738834
For gamma =  9.3 and fold = 1 , RMSE is 3242.881864
For gamma =  9.3 and fold = 2 , RMSE is 1676.245471
For gamma =  9.3 and fold = 3 , RMSE is 3028.093354
For gamma =  9.3 and fold = 4 , RMSE is 2838.098057
For gamma =  9.3 and fold = 5 , RMSE is 2705.906604
For gamma =  9.4 and fold = 1 , RMSE is 3223.340671
For gamma =  9.4 and fold = 2 , RMSE is 2922.56735
For gamma =  9.4 and fold = 3 , RMSE is 2528.202173
For gamma =  9.4 and fold = 4 , RMSE is 2379.351651
For gamma =  9.4 and fold = 5 , RMSE is 2012.698524
For gamma =  9.5 and fold = 1 , RMSE is 2183.407909
For gamma =  9.5 and fold = 2 , RMSE is 3152.496561
For gamma =  9.5 and fold = 3 , RMSE is 2932.767309
For gamma =  9.5 and fold = 4 , RMSE is 1932.717067
For gamma =  9.5 and fold = 5 , RMSE is 2432.487328
For gamma =  9.6 and fold = 1 , RMSE is 3088.674991
For gamma =  9.6 and fold = 2 , RMSE is 2177.986765
For gamma =  9.6 and fold = 3 , RMSE is 2640.398023
For gamma =  9.6 and fold = 4 , RMSE is 2537.964215
For gamma =  9.6 and fold = 5 , RMSE is 2021.634882
For gamma =  9.7 and fold = 1 , RMSE is 2711.185588
For gamma =  9.7 and fold = 2 , RMSE is 2510.761028
For gamma =  9.7 and fold = 3 , RMSE is 3186.054858
For gamma =  9.7 and fold = 4 , RMSE is 2790.8261
For gamma =  9.7 and fold = 5 , RMSE is 1855.473991
For gamma =  9.8 and fold = 1 , RMSE is 2418.417996
For gamma =  9.8 and fold = 2 , RMSE is 3039.742119
For gamma =  9.8 and fold = 3 , RMSE is 2294.843415
For gamma =  9.8 and fold = 4 , RMSE is 3155.2524
For gamma =  9.8 and fold = 5 , RMSE is 2467.689125
For gamma =  9.9 and fold = 1 , RMSE is 2342.237499
For gamma =  9.9 and fold = 2 , RMSE is 2973.328167
For gamma =  9.9 and fold = 3 , RMSE is 2426.319321
For gamma =  9.9 and fold = 4 , RMSE is 2424.637911
For gamma =  9.9 and fold = 5 , RMSE is 2313.35587
For gamma =  10.0 and fold = 1 , RMSE is 2233.93068
For gamma =  10.0 and fold = 2 , RMSE is 2009.354094
```

In [751]:
```
#Collecting Minimum Score Information
RMSE_KER_min = train_KER_RMSEs.min()
RMSE_KER_min_loc = np.argmin(train_KER_RMSEs)

g_KER_min = g_list[int(RMSE_KER_min_loc/g_count)]
fold_KER_min = (RMSE_KER_min_loc%g_count) + 1
msg = 'Lowest RMSE Score ' + str(round(RMSE_KER_min, 4)) + ' was found
for gamma = ' + str(g_KER_min) + ' during fold = ' + str(fold_KER_min)
+ '.'
print(msg)
```

```
Lowest RMSE Score 1661.4122 was found for gamma = 0.5 during fold = 4
5.
```

In [752]:
```
train_KER_avg_RMSEs = list()
for i, g  in enumerate(g_list):
    val = round(train_KER_RMSEs[i,:].mean(),4)
    train_KER_avg_RMSEs.append(val)
    print('For g =', g, 'Average RMSE was', val)
```

```
For g = 0.1 Average RMSE was 43492.682
For g = 0.2 Average RMSE was 40534.7137
For g = 0.3 Average RMSE was 39504.5006
For g = 0.4 Average RMSE was 36644.2576
For g = 0.5 Average RMSE was 35180.7565
For g = 0.6 Average RMSE was 33506.2248
For g = 0.7 Average RMSE was 32092.5985
For g = 0.8 Average RMSE was 30773.2919
For g = 0.9 Average RMSE was 28227.4046
For g = 1.0 Average RMSE was 27148.1482
For g = 1.1 Average RMSE was 26301.798
For g = 1.2 Average RMSE was 25156.2188
For g = 1.3 Average RMSE was 24247.2089
For g = 1.4 Average RMSE was 23445.777
For g = 1.5 Average RMSE was 23224.1218
For g = 1.6 Average RMSE was 22221.7873
For g = 1.7 Average RMSE was 21767.3287
For g = 1.8 Average RMSE was 20800.6274
For g = 1.9 Average RMSE was 20456.8684
For g = 2.0 Average RMSE was 20097.4776
For g = 2.1 Average RMSE was 19576.0539
For g = 2.2 Average RMSE was 18364.7197
For g = 2.3 Average RMSE was 17973.3645
For g = 2.4 Average RMSE was 17326.4897
For g = 2.5 Average RMSE was 16696.43
For g = 2.6 Average RMSE was 16280.541
For g = 2.7 Average RMSE was 16014.2383
For g = 2.8 Average RMSE was 15038.0796
For g = 2.9 Average RMSE was 14649.3724
For g = 3.0 Average RMSE was 14443.5361
For g = 3.1 Average RMSE was 13813.6494
For g = 3.2 Average RMSE was 13051.5097
For g = 3.3 Average RMSE was 12325.32
For g = 3.4 Average RMSE was 11949.2301
For g = 3.5 Average RMSE was 11728.7133
For g = 3.6 Average RMSE was 11453.0831
For g = 3.7 Average RMSE was 11568.6322
For g = 3.8 Average RMSE was 10693.5267
For g = 3.9 Average RMSE was 10688.6796
For g = 4.0 Average RMSE was 10358.8942
For g = 4.1 Average RMSE was 10158.0229
For g = 4.2 Average RMSE was 10095.0408
For g = 4.3 Average RMSE was 9618.9421
For g = 4.4 Average RMSE was 9234.4449
For g = 4.5 Average RMSE was 8532.6391
For g = 4.6 Average RMSE was 8480.2118
For g = 4.7 Average RMSE was 8553.5905
For g = 4.8 Average RMSE was 7902.0987
For g = 4.9 Average RMSE was 8029.6753
For g = 5.0 Average RMSE was 7497.297
For g = 5.1 Average RMSE was 6501.1955
For g = 5.2 Average RMSE was 5743.2903
For g = 5.3 Average RMSE was 5655.6618
For g = 5.4 Average RMSE was 5527.7425
For g = 5.5 Average RMSE was 5378.776
For g = 5.6 Average RMSE was 5346.6696
```

```
For g = 5.7 Average RMSE was 5405.8579
For g = 5.8 Average RMSE was 4736.5218
For g = 5.9 Average RMSE was 5092.6357
For g = 6.0 Average RMSE was 4902.9073
For g = 6.1 Average RMSE was 4917.8258
For g = 6.2 Average RMSE was 4561.2682
For g = 6.3 Average RMSE was 4480.9904
For g = 6.4 Average RMSE was 4223.8526
For g = 6.5 Average RMSE was 4282.9348
For g = 6.6 Average RMSE was 4248.5458
For g = 6.7 Average RMSE was 3984.081
For g = 6.8 Average RMSE was 3687.682
For g = 6.9 Average RMSE was 3551.4033
For g = 7.0 Average RMSE was 3185.7425
For g = 7.1 Average RMSE was 3316.4718
For g = 7.2 Average RMSE was 3238.1916
For g = 7.3 Average RMSE was 3220.2407
For g = 7.4 Average RMSE was 3160.8983
For g = 7.5 Average RMSE was 3222.7171
For g = 7.6 Average RMSE was 3136.5192
For g = 7.7 Average RMSE was 3173.3947
For g = 7.8 Average RMSE was 3210.9448
For g = 7.9 Average RMSE was 2935.7177
For g = 8.0 Average RMSE was 3028.2746
For g = 8.1 Average RMSE was 3013.2469
For g = 8.2 Average RMSE was 3007.4858
For g = 8.3 Average RMSE was 2877.6167
For g = 8.4 Average RMSE was 2845.0829
For g = 8.5 Average RMSE was 2808.5395
For g = 8.6 Average RMSE was 2855.657
For g = 8.7 Average RMSE was 2851.0913
For g = 8.8 Average RMSE was 2720.5334
For g = 8.9 Average RMSE was 2703.0817
For g = 9.0 Average RMSE was 2764.0548
For g = 9.1 Average RMSE was 2632.0499
For g = 9.2 Average RMSE was 2790.8173
For g = 9.3 Average RMSE was 2698.2451
For g = 9.4 Average RMSE was 2613.2321
For g = 9.5 Average RMSE was 2526.7752
For g = 9.6 Average RMSE was 2493.3318
For g = 9.7 Average RMSE was 2610.8603
For g = 9.8 Average RMSE was 2675.189
For g = 9.9 Average RMSE was 2495.9758
For g = 10.0 Average RMSE was 2516.5292
```

In [753]:
```
loc = np.argmin(train_KER_avg_RMSEs)
g_KER_min = round((loc+1)*0.1,1)

print(loc, g_KER_min, train_KER_avg_RMSEs[loc])
#train_KER_avg_RMSEs[int(g_KER_min*10)]
#min(train_KER_avg_RMSEs), g_KER_min
```

```
95 9.6 2493.3318
```

2.6 Use the best K-NN model and the best kernel smoothed model to make predictions on the test data and report the RMSE and mean absolute error (MAE) from each. Which model would you prefer? Consider other constraints beyond prediction error. (5 pts)

In [754]:
```
#Best kNN k = 5
#Best Kernel Smoothed Model gamma = 0.5
print(k_KNN_min, g_KER_min)
```

```
5 9.6
```

In [755]:
```
knn = KNeighborsClassifier(n_neighbors = k_KNN_min, p = 2)

# Fitting the kNN Models based on features for best performing k value
knn.fit(X_train[train_index], y_train[train_index])

# Doing predictions on  testing data-sets for best k value
y_test_pred = knn.predict(X_train[test_index])

# Calculating RMSE and MAE for testing data-sets
knn_RMSE = mth.sqrt(mean_squared_error(y_train[test_index], y_test_pre
d))
knn_MAE  = mean_absolute_error(y_train[test_index], y_test_pred)
```

In [756]:
```
ks = KNeighborsClassifier(n_neighbors = 50, weights = laplacian(gamma
= g))

# Fitting the kNN Models based on features
ks.fit(X_train[train_index], y_train[train_index])

# Doing predictions on training and testing data-sets for each fold
y_train_pred = ks.predict(X_train[train_index])

# Calculating RMSE for training and testing data-sets for each fold
train_KER_RMSEs[i,j]       = mth.sqrt(mean_squared_error(y_train[train
_index], y_train_pred))
```

In [757]:
```
ker = KNeighborsClassifier(n_neighbors = 50, weights = laplacian(gamma
= g_KER_min))

# Fitting the ker Models based on features for best performing k value
ker.fit(X_train[train_index], y_train[train_index])

# Doing predictions on  testing data-sets for best k value
y_test_pred = ker.predict(X_train[test_index])

# Calculating RMSE and MAE for testing data-sets
ker_RMSE = mth.sqrt(mean_squared_error(y_train[test_index], y_test_pre
d))
ker_MAE  = mean_absolute_error(y_train[test_index], y_test_pred)
```

In [758]:
```
print('For best performing KNN Model, RMSE value is', round(knn_RMSE,
4) , 'and MAE is ', round(knn_MAE,4))
print('For best performing Kernel Model, RMSE value is', round(ker_RMS
E,4) , 'and MAE is ', round(ker_MAE,4))
```

```
For best performing KNN Model, RMSE value is 57025.3715 and MAE is  3
7497.2293
For best performing Kernel Model, RMSE value is 39998.0375 and MAE is
26650.5171
```

Between these 2 models for doing predictions, I will prefer to use Smoothed Kernel Algorithm due to its lower RMSE and MAE Error Value.

I would prefer Smoothed Kernel Method over KNN as in KNN only k neighbors are considered for predicting the response value where in former, all observations are considered.