KNN

Import dataset

In [43]:
```
import sklearn
from sklearn.datasets import fetch_california_housing
# as_frame=True loads the data in a dataframe format, with other metadata besides i
california_housing = fetch_california_housing(as_frame=True)
# Select only the dataframe part and assign it to the df variable
df = california_housing.frame
```

In [44]:
```
import pandas as pd
df.head()
```

Out[44]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	Med
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	

Preprocessing Data for KNN Regression

In [45]:
```
y = df['MedHouseVal']
X = df.drop(['MedHouseVal'], axis = 1)
```

In [46]:
```
# .T transposes the results, transforming rows into columns
X.describe().T
```

Out[46]:

	count	mean	std	min	25%	50%	75%
MedInc	20640.0	3.870671	1.899822	0.499900	2.563400	3.534800	4.74325
HouseAge	20640.0	28.639486	12.585558	1.000000	18.000000	29.000000	37.00000
AveRooms	20640.0	5.429000	2.474173	0.846154	4.440716	5.229129	6.05238
AveBedrms	20640.0	1.096675	0.473911	0.333333	1.006079	1.048780	1.09952
Population	20640.0	1425.476744	1132.462122	3.000000	787.000000	1166.000000	1725.00000
AveOccup	20640.0	3.070655	10.386050	0.692308	2.429741	2.818116	3.28226
Latitude	20640.0	35.631861	2.135952	32.540000	33.930000	34.260000	37.71000
Longitude	20640.0	-119.569704	2.003532	-124.350000	-121.800000	-118.490000	-118.01000

Splitting Data into Train and Test Sets

In [47]:
```
from sklearn.model_selection import train_test_split

SEED = 42
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_st
```

In [48]:
```
print(len(X))       # 20640
print(len(X_train)) # 15480
print(len(X_test))  # 5160
```

```
20640
15480
5160
```

Feature Scaling for KNN Regression

In [49]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
# Fit only on X_train
scaler.fit(X_train)

# Scale both X_train and X_test
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

In [50]:
```
col_names=['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population', 'AveOccup',
scaled_df = pd.DataFrame(X_train, columns=col_names)
scaled_df.describe().T
```

Out[50]:

	count	mean	std	min	25%	50%	75%	max
MedInc	15480.0	2.172968e-16	1.000032	-1.774632	-0.688854	-0.175663	0.464450	5.842113
HouseAge	15480.0	-1.254954e-16	1.000032	-2.188261	-0.840224	0.032036	0.666407	1.855852
AveRooms	15480.0	-1.148163e-16	1.000032	-1.877586	-0.407008	-0.083940	0.257082	56.357392
AveBedrms	15480.0	1.239408e-16	1.000032	-1.740123	-0.205765	-0.108332	0.007435	55.925392
Population	15480.0	-7.874838e-17	1.000032	-1.246395	-0.558886	-0.227928	0.262056	29.971725
AveOccup	15480.0	2.672550e-17	1.000032	-0.201946	-0.056581	-0.024172	0.014501	103.737365
Latitude	15480.0	8.022581e-16	1.000032	-1.451215	-0.799820	-0.645172	0.971601	2.953905
Longitude	15480.0	2.169625e-15	1.000032	-2.380303	-1.106817	0.536231	0.785934	2.633738

Training and Predicting KNN Regression

In [51]:
```
from sklearn.neighbors import KNeighborsRegressor
regressor = KNeighborsRegressor(n_neighbors=5)
regressor.fit(X_train, y_train)
```

Out[51]:
```
KNeighborsRegressor()
```

In [52]:
```
y_pred = regressor.predict(X_test)
```

Evaluating the Algorithm for KNN Regression

In [53]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)

print(f'mae: {mae}')
print(f'mse: {mse}')
print(f'rmse: {rmse}')
```

```
mae: 0.4460739527131783
mse: 0.4316907430948294
rmse: 0.6570317671884894
```

In [54]:
```
regressor.score(X_test, y_test)
```

Out[54]:
```
0.6737569252627673
```

In [55]:
```
y.describe()
```

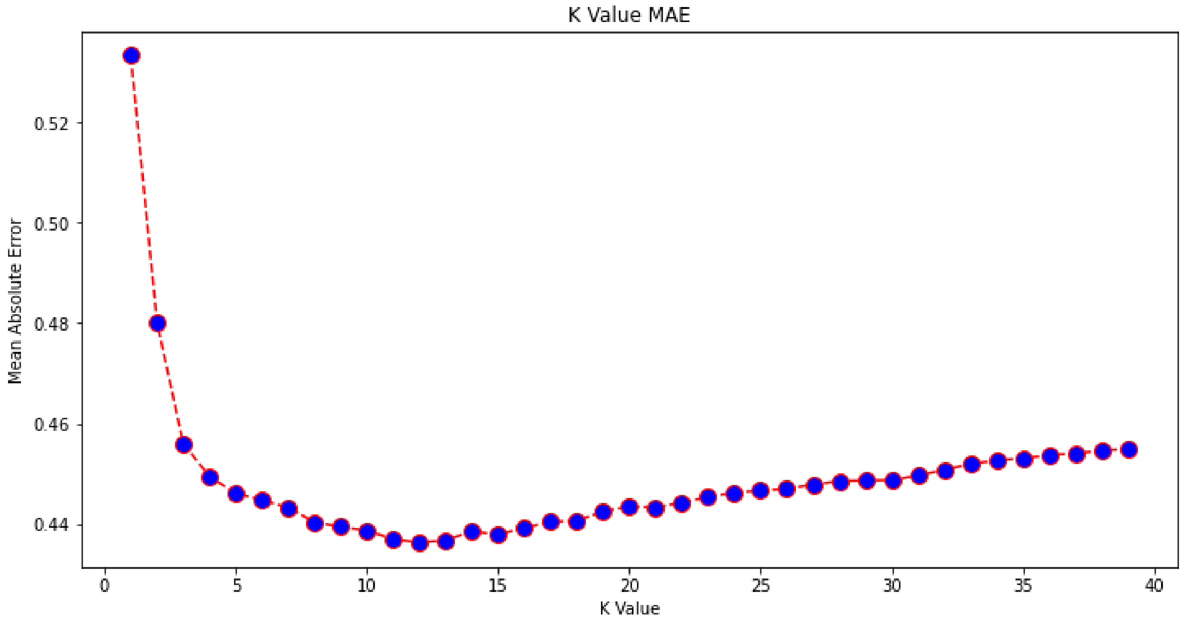
Out[55]:
```
count    20640.000000
mean         2.068558
std          1.153956
min          0.149990
25%          1.196000
50%          1.797000
75%          2.647250
max          5.000010
Name: MedHouseVal, dtype: float64
```

Finding the Best K for KNN Regression

In [56]:
```
error = []

# Calculating MAE error for K values between 1 and 39
for i in range(1, 40):
    knn = KNeighborsRegressor(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    mae = mean_absolute_error(y_test, pred_i)
    error.append(mae)
```

In [57]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
plt.plot(range(1, 40), error, color='red',
         linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)

plt.title('K Value MAE')
plt.xlabel('K Value')
plt.ylabel('Mean Absolute Error')
```

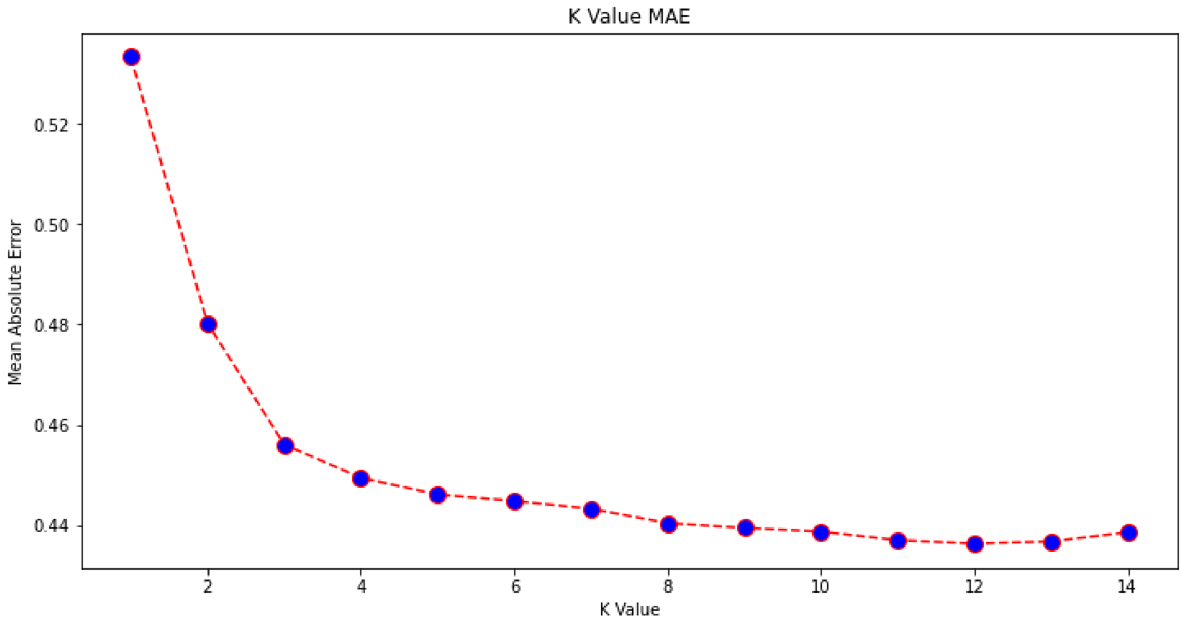
Out[57]: Text(0, 0.5, 'Mean Absolute Error')



Looking at the plot, it seems the lowest MAE value is when K is 12. Let's get a closer look at the plot to be sure by plotting less data

In [58]:
```
plt.figure(figsize=(12, 6))
plt.plot(range(1, 15), error[:14], color='red',
         linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('K Value MAE')
plt.xlabel('K Value')
plt.ylabel('Mean Absolute Error')
```

Out[58]: Text(0, 0.5, 'Mean Absolute Error')



In [59]:
```
import numpy as np

print(min(error))
print(np.array(error).argmin())
```

```
0.43631325936692505
11
```

KNN with 12 neighbours

In [60]:
```
knn_reg12 = KNeighborsRegressor(n_neighbors=12)
knn_reg12.fit(X_train, y_train)
y_pred12 = knn_reg12.predict(X_test)
r2 = knn_reg12.score(X_test, y_test)

mae12 = mean_absolute_error(y_test, y_pred12)
mse12 = mean_squared_error(y_test, y_pred12)
rmse12 = mean_squared_error(y_test, y_pred12, squared=False)
print(f'r2: {r2}, \nmae: {mae12} \nmse: {mse12} \nrmse: {rmse12}')
```

```
r2: 0.6887495617137436,
mae: 0.43631325936692505
mse: 0.4118522151025172
rmse: 0.6417571309323467
```

Classification using K-Nearest Neighbors with Scikit-Learn

Preprocessing Data for Classification

In [61]:
```
# Creating 4 categories and assigning them to a MedHouseValCat column
df["MedHouseValCat"] = pd.qcut(df["MedHouseVal"], 4, retbins=False, labels=[1, 2, 3
```

In [62]:
```
y = df['MedHouseValCat']
X = df.drop(['MedHouseVal', 'MedHouseValCat'], axis = 1)
```

Splitting Data into Train and Test Sets

In [63]:
```
from sklearn.model_selection import train_test_split

SEED = 42
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_st
```

Feature Scaling for Classification

In [64]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

Training and Predicting for Classification

In [65]:
```
from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier()
classifier.fit(X_train, y_train)
```

Out[65]:
```
KNeighborsClassifier()
```

In [66]:
```
y_pred = classifier.predict(X_test)
```

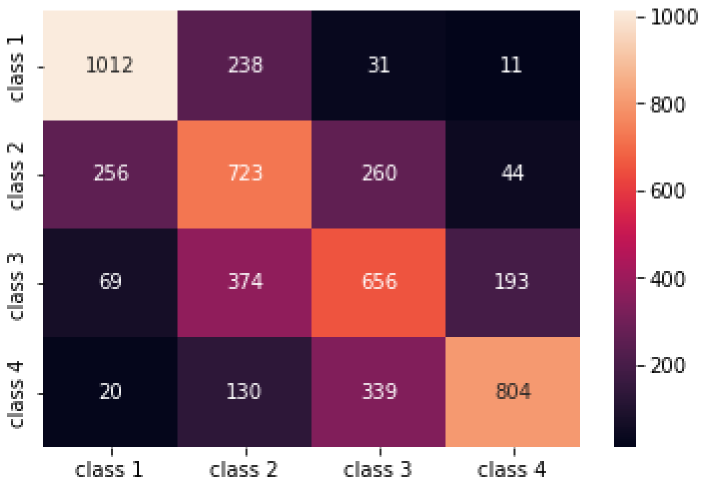
Evaluating KNN for Classification

In [67]:
```
acc =  classifier.score(X_test, y_test)
print(acc) # 0.6191860465116279
```

```
0.6191860465116279
```

In [68]:
```
from sklearn.metrics import classification_report, confusion_matrix
#importing Seaborn's to use the heatmap
import seaborn as sns

# Adding classes names for better interpretation
classes_names = ['class 1','class 2','class 3', 'class 4']
cm = pd.DataFrame(confusion_matrix(y_test, y_pred),
                  columns=classes_names, index = classes_names)

# Seaborn's heatmap to better visualize the confusion matrix
sns.heatmap(cm, annot=True, fmt='d');

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           1       0.75      0.78      0.76      1292
           2       0.49      0.56      0.53      1283
           3       0.51      0.51      0.51      1292
           4       0.76      0.62      0.69      1293

    accuracy                           0.62      5160
   macro avg       0.63      0.62      0.62      5160
weighted avg       0.63      0.62      0.62      5160
```



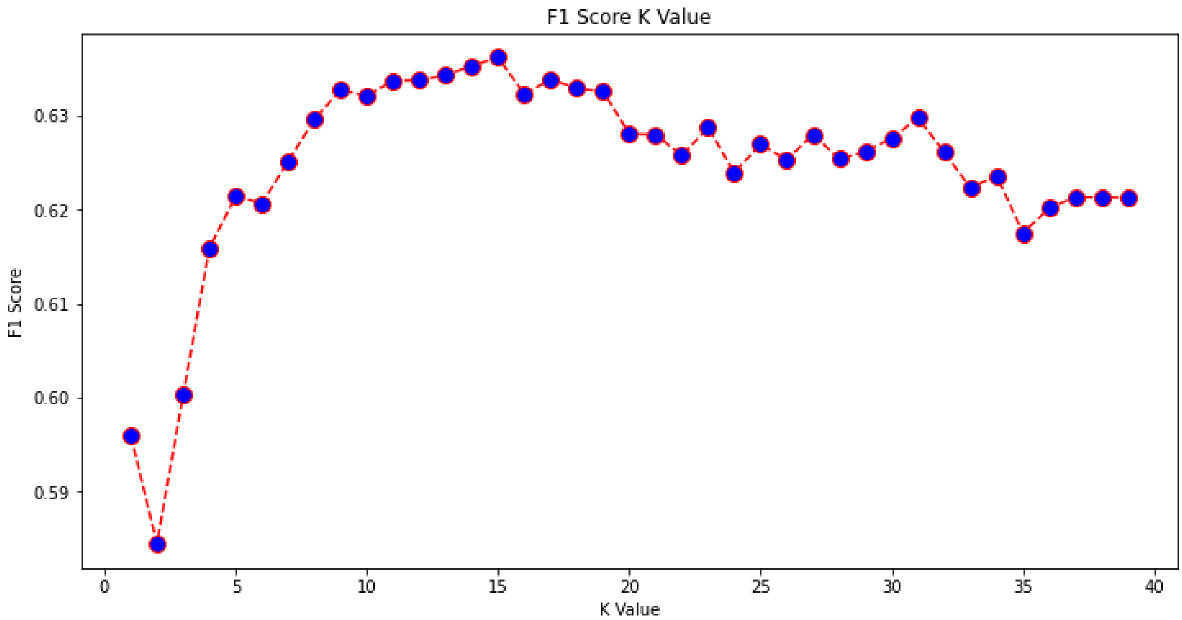
Finding the Best K for KNN Classification

In [69]:
```
from sklearn.metrics import f1_score

f1s = []

# Calculating f1 score for K values between 1 and 40
for i in range(1, 40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
```

```
    pred_i = knn.predict(X_test)
    # using average='weighted' to calculate a weighted average for the 4 classes
    f1s.append(f1_score(y_test, pred_i, average='weighted'))
```

In [70]:
```
plt.figure(figsize=(12, 6))
plt.plot(range(1, 40), f1s, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('F1 Score K Value')
plt.xlabel('K Value')
plt.ylabel('F1 Score')
```

Out[70]: Text(0, 0.5, 'F1 Score')



From the output, we can see that the f1-score is the highest when the value of the K is 15.

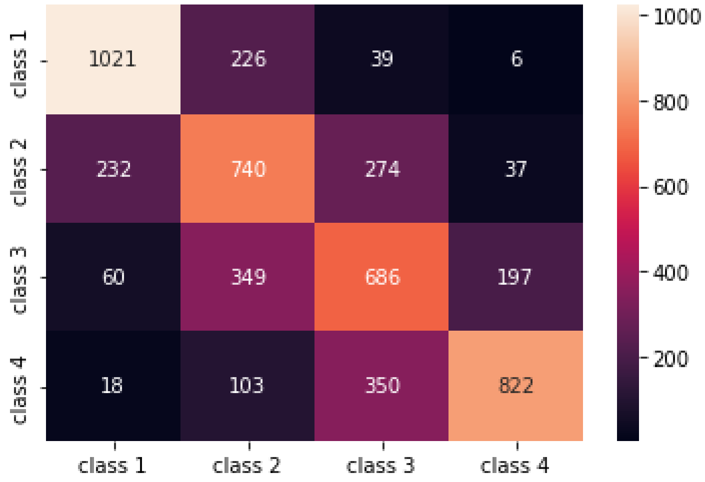
In [71]:
```
classifier15 = KNeighborsClassifier(n_neighbors=15)
classifier15.fit(X_train, y_train)
y_pred15 = classifier15.predict(X_test)
print(classification_report(y_test, y_pred15))
```

```
              precision    recall  f1-score   support

           1       0.77      0.79      0.78      1292
           2       0.52      0.58      0.55      1283
           3       0.51      0.53      0.52      1292
           4       0.77      0.64      0.70      1293

    accuracy                           0.63      5160
   macro avg       0.64      0.63      0.64      5160
weighted avg       0.64      0.63      0.64      5160
```

In [72]:
```
acc =  classifier.score(X_test, y_pred15)
print(acc)
```

0.7874031007751938

In [73]:
```
cm = pd.DataFrame(confusion_matrix(y_test, y_pred15),
                  columns=classes_names, index = classes_names)

sns.heatmap(cm, annot=True, fmt='d');
```



Implementing KNN for Outlier Detection with Scikit-Learn

In [74]:
```
from sklearn.neighbors import NearestNeighbors

nbrs = NearestNeighbors(n_neighbors = 5)
nbrs.fit(X_train)
# Distances and indexes of the 5 neighbors
distances, indexes = nbrs.kneighbors(X_train)
```

In [75]:
```
distances[:3], distances.shape
```

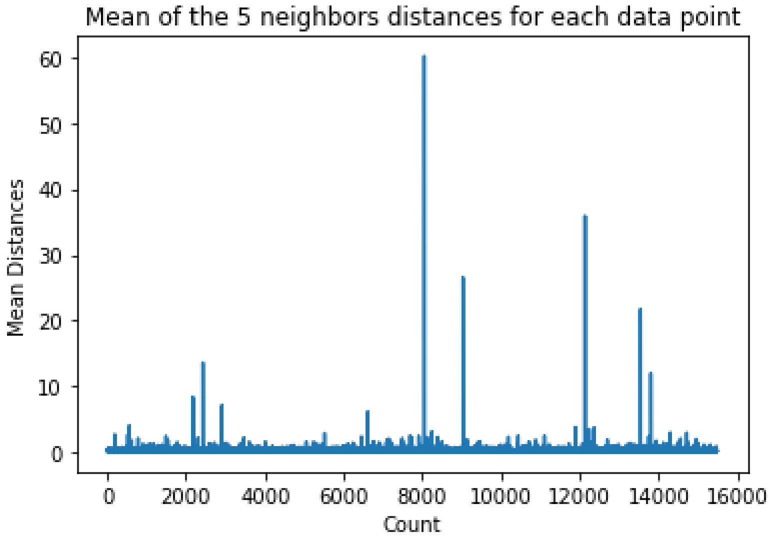
Out[75]:
```
(array([[0.        , 0.12998939, 0.15157687, 0.16543705, 0.17750354],
        [0.        , 0.25535314, 0.37100754, 0.39090243, 0.40619693],
        [0.        , 0.27149697, 0.28024623, 0.28112326, 0.30420656]]),
 (15480, 5))
```

In [76]:
```
indexes[:3], indexes[:3].shape
```

Out[76]:
```
(array([[    0,  8608, 12831,  8298,  2482],
        [    1,  4966,  5786,  8568,  6759],
        [    2, 13326, 13936,  3618,  9756]], dtype=int64),
 (3, 5))
```

In [77]:
```
dist_means = distances.mean(axis=1)
plt.plot(dist_means)
plt.title('Mean of the 5 neighbors distances for each data point')
plt.xlabel('Count')
plt.ylabel('Mean Distances')
```

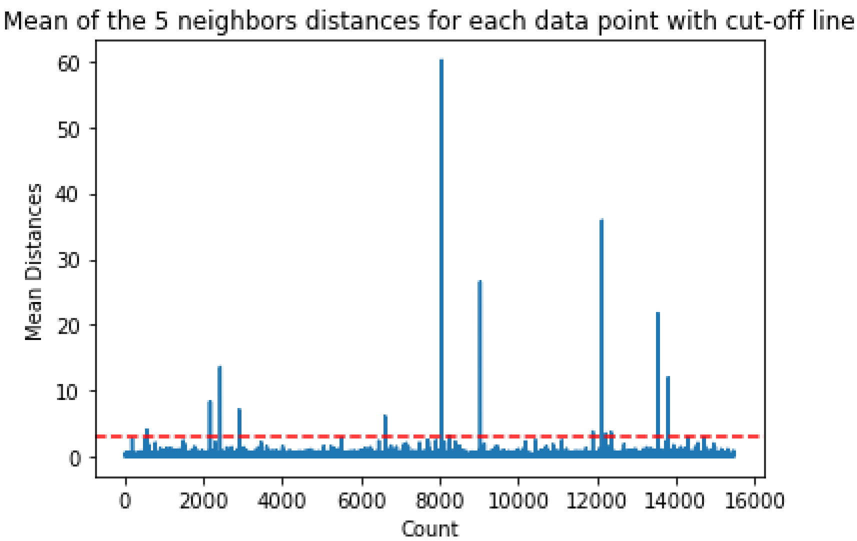
Out[77]:
```
Text(0, 0.5, 'Mean Distances')
```



mean distance is 3. Let's plot the graph again with a horizontal dotted line to be able to spot it

In [78]:
```
dist_means = distances.mean(axis=1)
plt.plot(dist_means)
plt.title('Mean of the 5 neighbors distances for each data point with cut-off line'
plt.xlabel('Count')
plt.ylabel('Mean Distances')
plt.axhline(y = 3, color = 'r', linestyle = '--')
```

Out[78]: <matplotlib.lines.Line2D at 0x20444b5a9a0>



In [79]:
```
import numpy as np

# Visually determine cutoff values > 3
outlier_index = np.where(dist_means > 3)
outlier_index
```

Out[79]:
```
(array([  564,  2167,  2415,  2902,  6607,  8047,  8243,  9029, 11892,
        12127, 12226, 12353, 13534, 13795, 14292, 14707], dtype=int64),)
```

In [80]:
```
# Filter outlier values
outlier_values = df.iloc[outlier_index]
outlier_values
```

Out[80]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
564	4.8711	27.0	5.082811	0.944793	1499.0	1.880803	37.75	-122.24
2167	2.8359	30.0	4.948357	1.001565	1660.0	2.597809	36.78	-119.83
2415	2.8250	32.0	4.784232	0.979253	761.0	3.157676	36.59	-119.44
2902	1.1875	48.0	5.492063	1.460317	129.0	2.047619	35.38	-119.02
6607	3.5164	47.0	5.970639	1.074266	1700.0	2.936097	34.18	-118.14
8047	2.7260	29.0	3.707547	1.078616	2515.0	1.977201	33.84	-118.17
8243	2.0769	17.0	3.941667	1.211111	1300.0	3.611111	33.78	-118.18
9029	6.8300	28.0	6.748744	1.080402	487.0	2.447236	34.05	-118.78
11892	2.6071	45.0	4.225806	0.903226	89.0	2.870968	33.99	-117.35
12127	4.1482	7.0	5.674957	1.106998	5595.0	3.235975	33.92	-117.25
12226	2.8125	18.0	4.962500	1.112500	239.0	2.987500	33.63	-116.92
12353	3.1493	24.0	7.307323	1.460984	1721.0	2.066026	33.81	-116.54
13534	3.7949	13.0	5.832258	1.072581	2189.0	3.530645	34.17	-117.33
13795	1.7567	8.0	4.485173	1.120264	3220.0	2.652389	34.59	-117.42
14292	2.6250	50.0	4.742236	1.049689	728.0	2.260870	32.74	-117.13
14707	3.7167	17.0	5.034130	1.051195	549.0	1.873720	32.80	-117.05

KNN With Outlier Removal

Import Libraries

In [94]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
```

Import Dataset

In [55]:
```
path_to_file = './housing.csv'
df = pd.read_csv(path_to_file)
```

In [56]:
```
df.head()
```

Out[56]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	Med
0	8.3252	41	6.984127	1.023810	322	2.555556	37.88	-122.23	
1	8.3014	21	6.238137	0.971880	2401	2.109842	37.86	-122.22	
2	7.2574	52	8.288136	1.073446	496	2.802260	37.85	-122.24	
3	5.6431	52	5.817352	1.073059	558	2.547945	37.85	-122.25	
4	3.8462	52	6.281853	1.081081	565	2.181467	37.85	-122.25	

Analysis of Data

In [57]:
```
df.shape
```

Out[57]: (20640, 9)

In [58]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   MedInc       20640 non-null  float64
 1   HouseAge     20640 non-null  int64  
 2   AveRooms     20640 non-null  float64
 3   AveBedrms    20640 non-null  float64
 4   Population   20640 non-null  int64  
 5   AveOccup     20640 non-null  float64
 6   Latitude     20640 non-null  float64
 7   Longitude    20640 non-null  float64
 8   MedHouseVal  20640 non-null  float64
dtypes: float64(7), int64(2)
memory usage: 1.4 MB
```

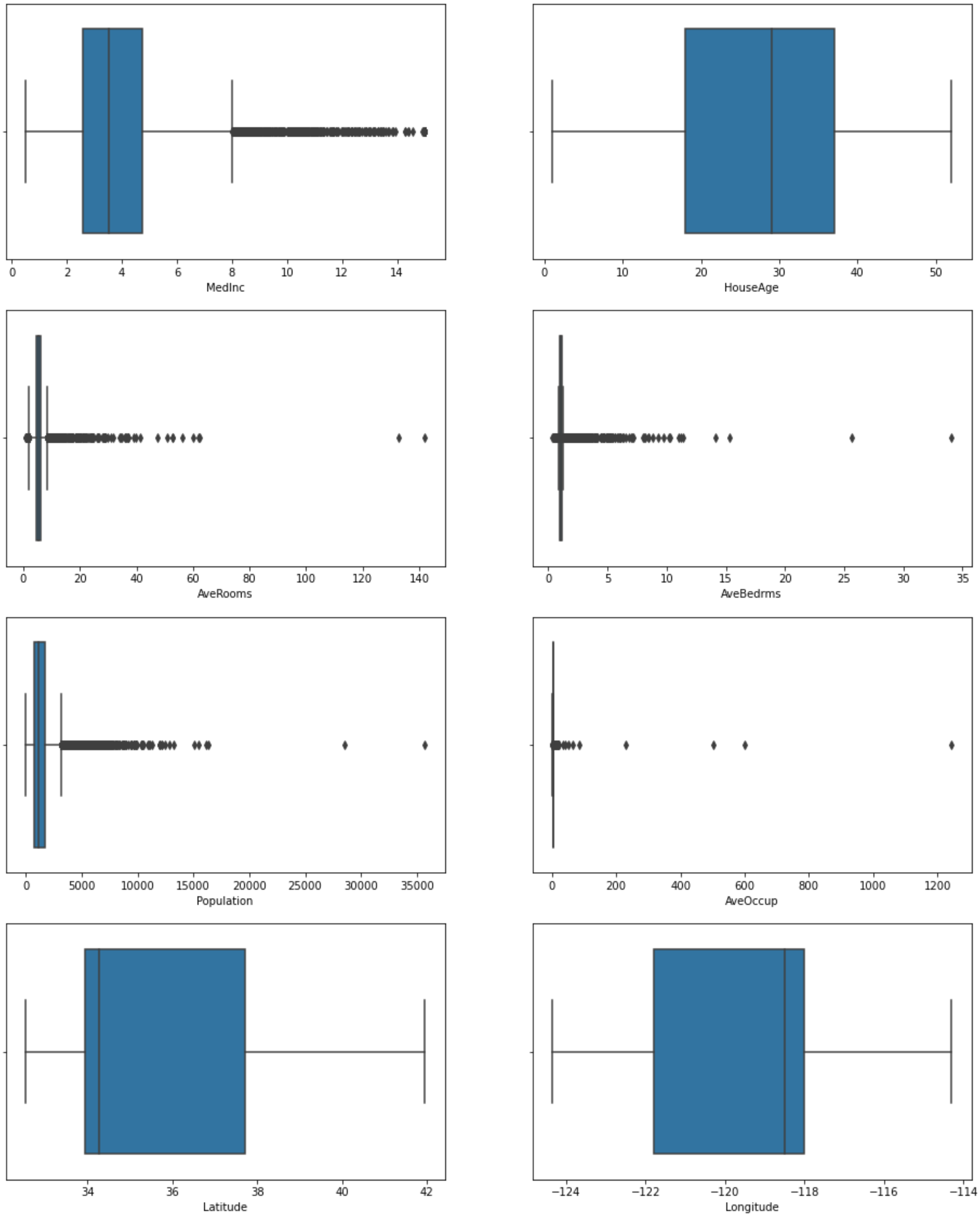
Inference

- There is not any null value.
- There is not any column with object type.

Outlier Removal

In [59]:
```
plt.figure(figsize=(16,20))
plt.subplot(4,2,1)
sns.boxplot(df['MedInc'])

plt.subplot(4,2,2)
sns.boxplot(df['HouseAge'])

plt.subplot(4,2,3)
sns.boxplot(df['AveRooms'])

plt.subplot(4,2,4)
sns.boxplot(df['AveBedrms'])

plt.subplot(4,2,5)
sns.boxplot(df['Population'])

plt.subplot(4,2,6)
sns.boxplot(df['AveOccup'])

plt.subplot(4,2,7)
sns.boxplot(df['Latitude'])

plt.subplot(4,2,8)
sns.boxplot(df['Longitude'])

plt.show()
```

```
G:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the f
ollowing variable as a keyword arg: x. From version 0.12, the only valid positiona
l argument will be `data`, and passing other arguments without an explicit keyword
will result in an error or misinterpretation.
  warnings.warn(
G:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the f
ollowing variable as a keyword arg: x. From version 0.12, the only valid positiona
l argument will be `data`, and passing other arguments without an explicit keyword
will result in an error or misinterpretation.
  warnings.warn(
G:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the f
ollowing variable as a keyword arg: x. From version 0.12, the only valid positiona
l argument will be `data`, and passing other arguments without an explicit keyword
will result in an error or misinterpretation.
  warnings.warn(
G:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the f
ollowing variable as a keyword arg: x. From version 0.12, the only valid positiona
l argument will be `data`, and passing other arguments without an explicit keyword
will result in an error or misinterpretation.
  warnings.warn(
G:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the f
ollowing variable as a keyword arg: x. From version 0.12, the only valid positiona
l argument will be `data`, and passing other arguments without an explicit keyword
will result in an error or misinterpretation.
  warnings.warn(
G:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the f
ollowing variable as a keyword arg: x. From version 0.12, the only valid positiona
l argument will be `data`, and passing other arguments without an explicit keyword
will result in an error or misinterpretation.
  warnings.warn(
G:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the f
ollowing variable as a keyword arg: x. From version 0.12, the only valid positiona
l argument will be `data`, and passing other arguments without an explicit keyword
will result in an error or misinterpretation.
  warnings.warn(
G:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the f
ollowing variable as a keyword arg: x. From version 0.12, the only valid positiona
l argument will be `data`, and passing other arguments without an explicit keyword
will result in an error or misinterpretation.
  warnings.warn(
```

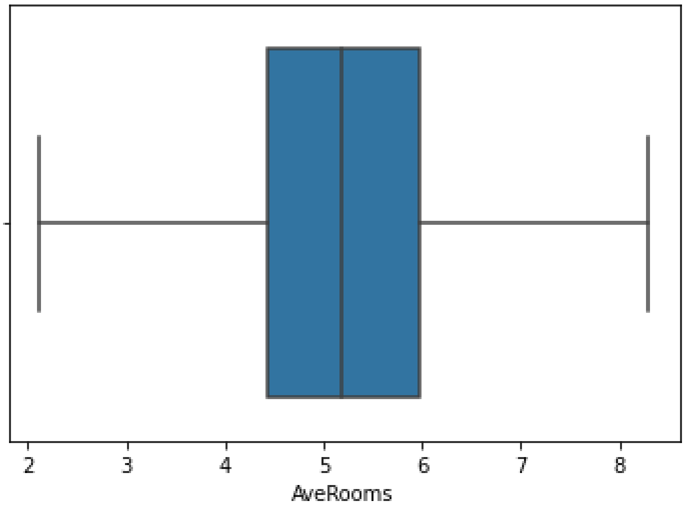


In [62]:
```python
q1 = df["AveRooms"].quantile(0.25)
q3 = df["AveRooms"].quantile(0.75)
iqr = q3-q1
low = q1 - 1.5*iqr
high = q3 + 1.5*iqr

df = df[~((df['AveRooms'] >= high) | (df['AveRooms']<= low))]
sns.boxplot(df['AveRooms'])
```

```
G:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[62]:
```
<AxesSubplot:xlabel='AveRooms'>
```

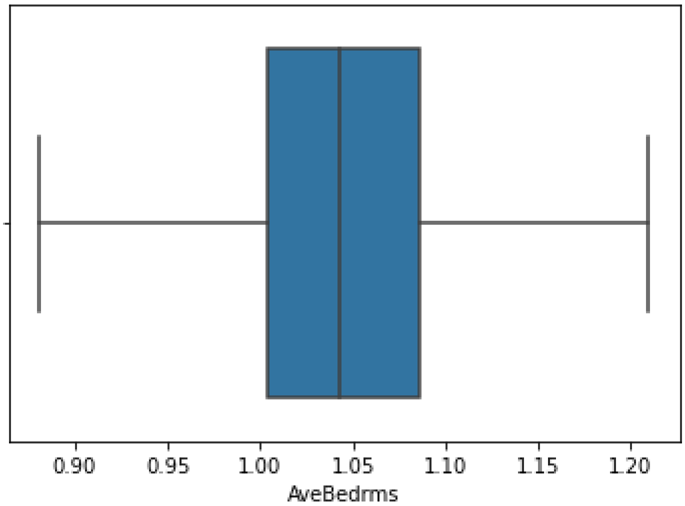
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



In [69]:
```
q1 = df["AveBedrms"].quantile(0.25)
q3 = df["AveBedrms"].quantile(0.75)
iqr = q3-q1
low = q1 - 1.5*iqr
high = q3 + 1.5*iqr

df = df[~((df['AveBedrms'] >= high) | (df['AveBedrms']<= low))]
sns.boxplot(df['AveBedrms'])
```

```
G:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[69]: `<AxesSubplot:xlabel='AveBedrms'>`

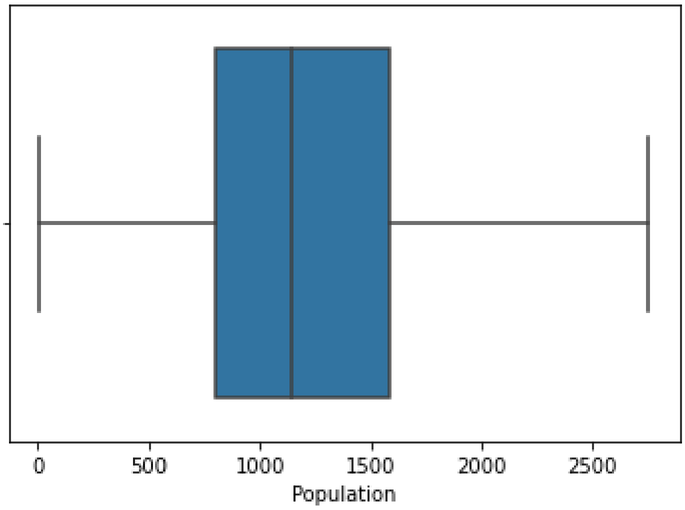


In [74]:
```
q1 = df["Population"].quantile(0.25)
q3 = df["Population"].quantile(0.75)
iqr = q3-q1
low = q1 - 1.5*iqr
high = q3 + 1.5*iqr

df = df[~((df['Population'] >= high) | (df['Population']<= low))]
sns.boxplot(df['Population'])
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
G:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the f
ollowing variable as a keyword arg: x. From version 0.12, the only valid positiona
l argument will be `data`, and passing other arguments without an explicit keyword
will result in an error or misinterpretation.
  warnings.warn(
```

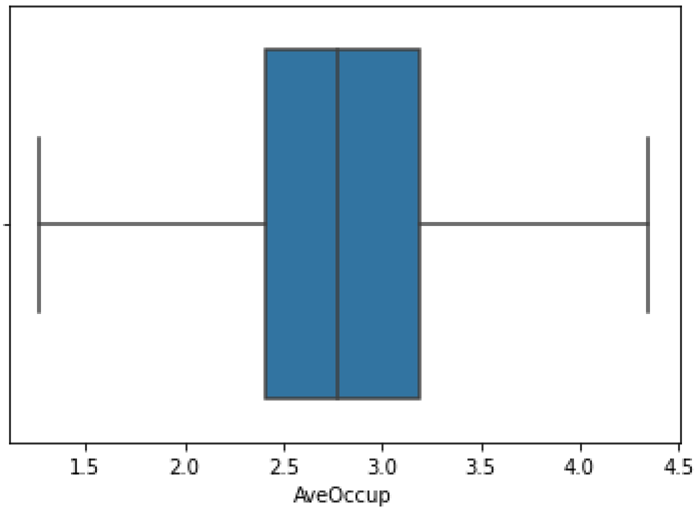
Out[74]: <AxesSubplot:xlabel='Population'>



In [80]:
```
q1 = df["AveOccup"].quantile(0.25)
q3 = df["AveOccup"].quantile(0.75)
iqr = q3-q1
low = q1 - 1.5*iqr
high = q3 + 1.5*iqr

df = df[~((df['AveOccup'] >= high) | (df['AveOccup']<= low))]
sns.boxplot(df['AveOccup'])
```

```
G:\anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the f
ollowing variable as a keyword arg: x. From version 0.12, the only valid positiona
l argument will be `data`, and passing other arguments without an explicit keyword
will result in an error or misinterpretation.
  warnings.warn(
```

Out[80]: <AxesSubplot:xlabel='AveOccup'>



Train Test Split

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

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

In [82]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_st
```

In [83]:
```
X_train
```

Out[83]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
10726	11.0138	16	7.306991	1.060790	868	2.638298	33.64	-117.81
9906	3.4543	12	4.801042	1.046875	2293	2.388542	38.32	-122.28
11947	4.6327	34	5.552817	0.957746	880	3.098592	33.93	-117.44
9134	4.8667	14	6.925743	1.136139	1236	3.059406	34.51	-118.07
6347	2.0156	44	4.076923	1.153846	502	4.290598	34.06	-117.75
...	...	...	...	...	...	...	...	...
17184	4.5625	21	4.667954	1.193050	801	3.092664	37.50	-122.49
6813	3.2361	28	3.654054	0.956757	543	2.935135	34.10	-118.07
981	6.8132	4	6.359838	0.998652	1895	2.553908	37.68	-121.85
20020	1.5893	17	4.244337	1.066343	1912	3.093851	36.07	-119.04
9242	2.5388	12	4.508816	0.954660	1399	3.523929	36.98	-120.07

12196 rows × 8 columns

In [84]:
```
X_test
```

Out[84]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
17095	3.9290	36	4.678241	1.002315	1117	2.585648	37.47	-122.24
13785	2.7028	29	4.828326	1.137339	1760	2.517883	34.03	-117.04
2880	1.3750	35	4.050847	1.031477	1041	2.520581	35.38	-118.97
8063	6.4468	43	5.948198	0.925676	1011	2.277027	33.83	-118.19
17648	6.0791	23	6.119910	1.015837	1180	2.669683	37.25	-121.89
...	...	...	...	...	...	...	...	...
8561	4.1818	22	4.426056	1.065141	1225	2.156690	33.93	-118.41
3895	3.2250	33	4.285714	1.072084	2118	2.775885	34.20	-118.53
19466	3.1625	16	5.992347	1.137755	1302	3.321429	37.68	-120.97
4689	2.3375	40	4.129252	1.013605	777	1.761905	34.07	-118.36
11168	4.1167	33	4.601179	0.933202	1367	2.685658	33.82	-117.99

4066 rows × 8 columns

Scaling Dataset

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

In [86]:
```
X_train
```

Out[86]:
```
array([[ 3.96529227, -1.12990404,  1.87674182, ..., -0.28905557,
        -0.95589406,  0.92501103],
       [-0.25974173, -1.45631436, -0.4062846 , ..., -0.71074472,
         1.23981314, -1.30709423],
       [ 0.39887062,  0.33894244,  0.27861465, ...,  0.48810571,
        -0.81983528,  1.10977142],
       ...,
       [ 1.61756056, -2.10913502,  1.01384548, ..., -0.43153909,
         0.93954549, -1.0923727 ],
       [-1.3020975 , -1.04830145, -0.91346664, ...,  0.48010197,
         0.18418468,  0.31080757],
       [-0.77141825, -1.45631436, -0.67251451, ...,  1.20624757,
         0.61112775, -0.20352541]])
```

In [87]:
```
col_names=['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population', 'AveOccup',
scaled_df = pd.DataFrame(X_train, columns=col_names)
scaled_df.describe()
```

Out[87]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup
count	1.219600e+04	1.219600e+04	1.219600e+04	1.219600e+04	1.219600e+04	1.219600e+04
mean	-7.778662e-17	1.611990e-16	1.572300e-16	-1.648949e-16	5.990798e-17	5.897764e-16
std	1.000041e+00	1.000041e+00	1.000041e+00	1.000041e+00	1.000041e+00	1.000041e+00
min	-1.890791e+00	-2.353943e+00	-2.836138e+00	-2.592897e+00	-2.144074e+00	-2.610157e+00
25%	-7.168692e-01	-8.034937e-01	-6.993266e-01	-6.817842e-01	-7.356713e-01	-6.824827e-01
50%	-1.590552e-01	9.413469e-02	-3.644057e-02	-3.713085e-02	-1.485411e-01	-6.414291e-02
75%	5.152344e-01	6.653528e-01	6.482085e-01	6.530779e-01	6.266127e-01	6.301155e-01
max	6.193251e+00	1.807789e+00	2.778775e+00	2.617318e+00	2.726800e+00	2.596455e+00

Training and Prediction For Regression

In [88]:
```
regressor = KNeighborsRegressor(n_neighbors=5)
regressor.fit(X_train, y_train)
```

Out[88]:
```
▾ KNeighborsRegressor
KNeighborsRegressor()
```

In [89]:
```
y_pred = regressor.predict(X_test)
```

In [90]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
                                    pred, squared=False)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
print(f'mae: {mae}')
print(f'mse: {mse}')
print(f'rmse: {rmse}')
```

```
mae: 0.4423957250368913
mse: 0.39249224151982975
rmse: 0.626492012335217
```

In [91]:
```
regressor.score(X_test, y_test)
```

Out[91]: 0.6889819935603496

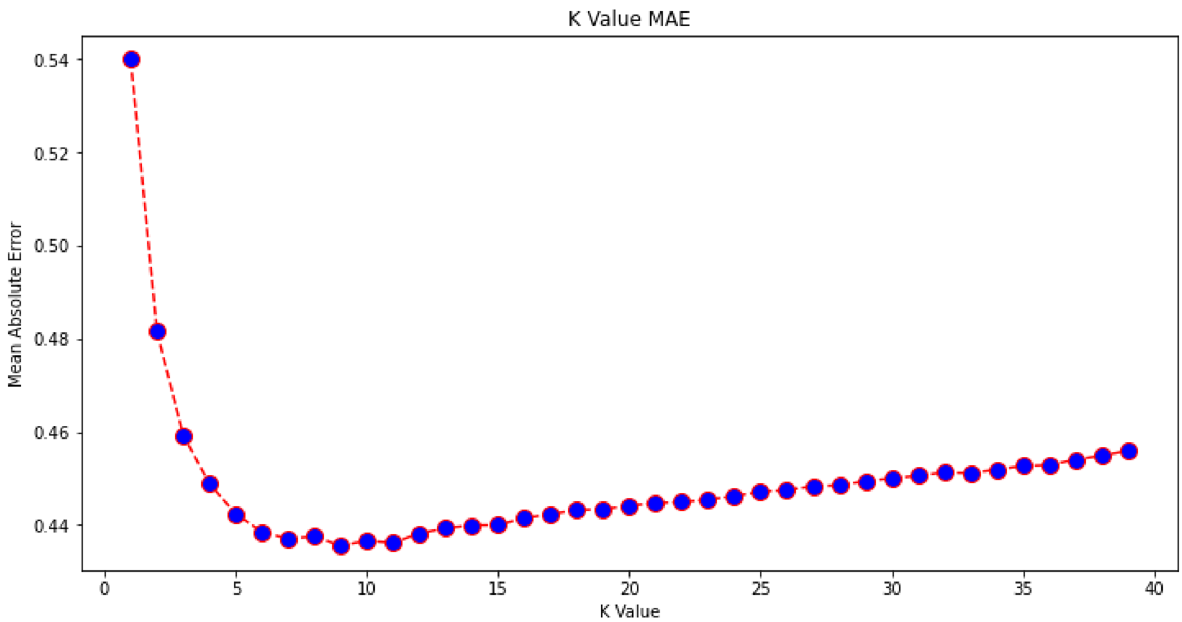
Tuning the parameters of KNN Regression

Best Value of K

In [92]:
```
error = []

# Calculating MAE error for K values between 1 and 39
for i in range(1, 40):
    knn = KNeighborsRegressor(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    mae = mean_absolute_error(y_test, pred_i)
    error.append(mae)
```

In [93]:
```
plt.figure(figsize=(12, 6))
plt.plot(range(1, 40), error, color='red',
         linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)

plt.title('K Value MAE')
plt.xlabel('K Value')
plt.ylabel('Mean Absolute Error')
```

Out[93]: Text(0, 0.5, 'Mean Absolute Error')



8

Inference

- Looking at the plot, it seems the lowest MAE value is when ***K*** is ***8***.

In [97]:
```
knn_reg8 = KNeighborsRegressor(n_neighbors=8)
knn_reg8.fit(X_train, y_train)
y_pred8 = knn_reg8.predict(X_test)
r2 = knn_reg8.score(X_test, y_test)

mae8 = mean_absolute_error(y_test, y_pred8)
mse8 = mean_squared_error(y_test, y_pred8)
rmse8 = mean_squared_error(y_test, y_pred8, squared=False)
print(f'r2: {r2}, \nmae: {mae8} \nmse: {mse8} \nrmse: {rmse8}')
```

```
r2: 0.702721456111149,
mae: 0.4375316087678307
mse: 0.37515359120960284
rmse: 0.6124978295550139
```

Conclusion For Regression

1. ***Observation before outlier removal***

- r2: 0.6887495617137436,
- mae: 0.43631325936692505
- mse: 0.4118522151025172
- rmse: 0.6417571309323467

2. ***Observation after outlier removal and k value selection***

- r2: 0.702721456111149
- mae: 0.4375316087678307
- mse: 0.37515359120960284
- rmse: 0.6124978295550139

Training and Prediction For Classification

In [98]:
```
df["MedHouseValCat"] = pd.qcut(df["MedHouseVal"], 4, retbins=False, labels=[1, 2, 3
```

In [100…
```
y = df['MedHouseValCat']
X = df.drop(['MedHouseVal'], axis = 1)
```

In [102…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_st
```

In [103…
```
scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [113…
```
classifier = KNeighborsClassifier(5)
```

```
classifier.fit(X_train, y_train)
```

Out[113]:

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [114…

```
y_pred = classifier.predict(X_test)
```

```
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

In [115…

```
acc =  classifier.score(X_test, y_test)
print(acc)
```

```
0.9795868175110674
```

```
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

Tuning the parameters of KNN Classification

Best Value of K

In [107…

```
from sklearn.metrics import f1_score

f1s = []

for i in range(1, 40):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    f1s.append(f1_score(y_test, pred_i, average='weighted'))
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
e of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
```

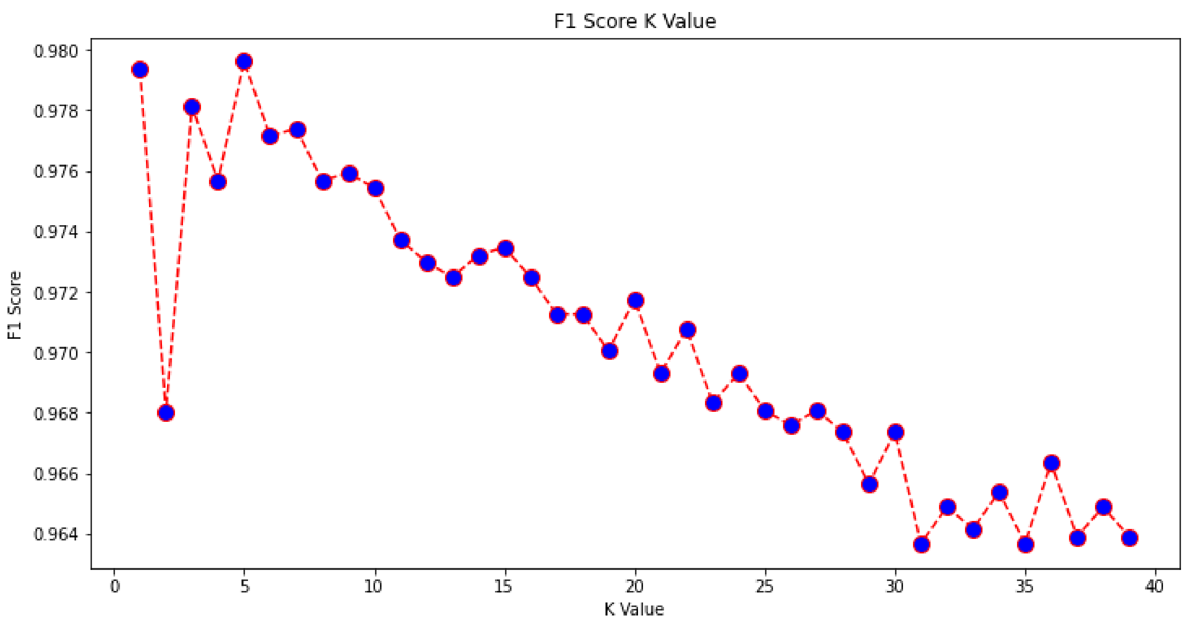
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

In [108…

```
plt.figure(figsize=(12, 6))
plt.plot(range(1, 40), f1s, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('F1 Score K Value')
plt.xlabel('K Value')
plt.ylabel('F1 Score')
```

Out[108]: Text(0, 0.5, 'F1 Score')



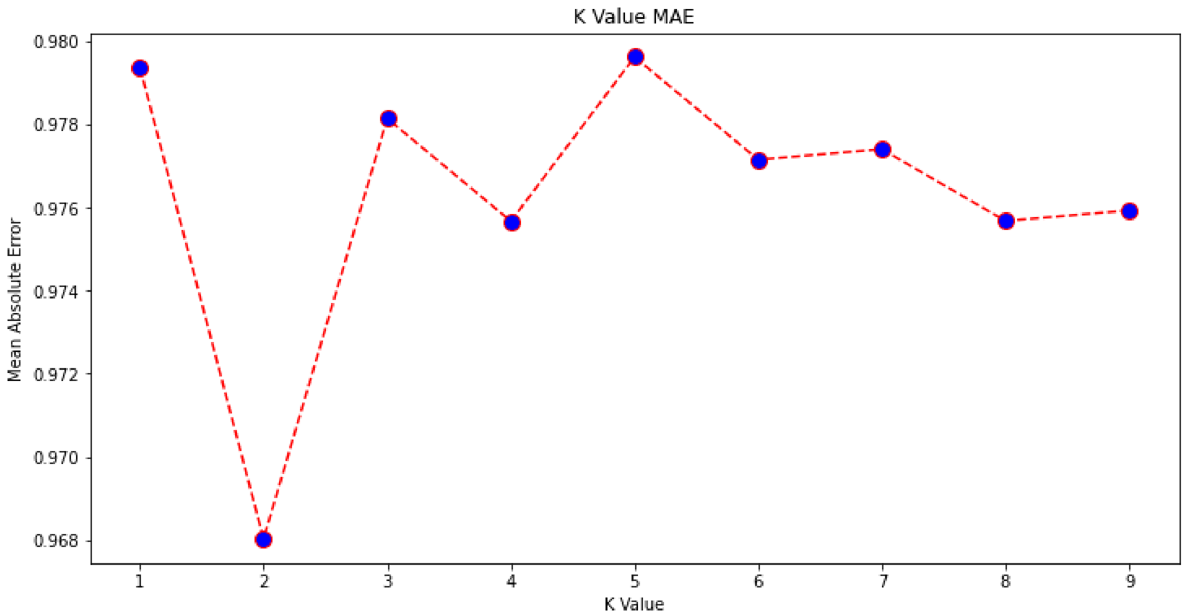
In [112…

```
plt.figure(figsize=(12, 6))
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
            markerfacecolor='blue', markersize=10)
plt.title('K Value MAE')
plt.xlabel('K Value')
plt.ylabel('Mean Absolute Error')
```

Out[112]: Text(0, 0.5, 'Mean Absolute Error')



Inference

- Looking at the plot, it seems the max f1s value is when ***K*** is ***5***.

In [116…

```
knn_class5 = KNeighborsRegressor(n_neighbors=5)
knn_class5.fit(X_train, y_train)
y_pred5 = knn_class5.predict(X_test)
acc =  classifier.score(X_test, y_test)
print(acc)
```

```
0.9795868175110674
```

```
G:\anaconda\lib\site-packages\sklearn\neighbors\_classification.py:237: FutureWarn
ing: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behav
ior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this be
havior will change: the default value of `keepdims` will become False, the `axis`
over which the statistic is taken will be eliminated, and the value None will no l
onger be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

Conclusion For Classification

1. ***Observation before outlier removal***

- 0.7874031007751938

2. ***Observation after outlier removal and k value selection***

- acc: 0.9795868175110674

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

Weighted KNN

Import dataset

In [18]:
```
import sklearn
from sklearn.datasets import fetch_california_housing
# as_frame=True loads the data in a dataframe format, with other metadata besides i
california_housing = fetch_california_housing(as_frame=True)
# Select only the dataframe part and assign it to the df variable
df = california_housing.frame
```

In [19]:
```
import pandas as pd
df.head()
```

Out[19]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	Med
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	

Preprocessing Data for KNN Regression

In [20]:
```
y = df['MedHouseVal']
X = df.drop(['MedHouseVal'], axis = 1)
```

In [21]:
```
# .T transposes the results, transforming rows into columns
X.describe().T
```

Out[21]:

	count	mean	std	min	25%	50%	75%
MedInc	20640.0	3.870671	1.899822	0.499900	2.563400	3.534800	4.74325
HouseAge	20640.0	28.639486	12.585558	1.000000	18.000000	29.000000	37.00000
AveRooms	20640.0	5.429000	2.474173	0.846154	4.440716	5.229129	6.05238
AveBedrms	20640.0	1.096675	0.473911	0.333333	1.006079	1.048780	1.09952
Population	20640.0	1425.476744	1132.462122	3.000000	787.000000	1166.000000	1725.00000
AveOccup	20640.0	3.070655	10.386050	0.692308	2.429741	2.818116	3.28226
Latitude	20640.0	35.631861	2.135952	32.540000	33.930000	34.260000	37.71000
Longitude	20640.0	-119.569704	2.003532	-124.350000	-121.800000	-118.490000	-118.01000

Splitting Data into Train and Test Sets

In [22]:
```
from sklearn.model_selection import train_test_split

SEED = 42
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_st
```

In [23]:
```
print(len(X))       # 20640
print(len(X_train)) # 15480
print(len(X_test))  # 5160
```

```
20640
15480
5160
```

Feature Scaling for KNN Regression

In [24]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
# Fit only on X_train
scaler.fit(X_train)

# Scale both X_train and X_test
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

In [25]:
```
col_names=['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms', 'Population', 'AveOccup',
scaled_df = pd.DataFrame(X_train, columns=col_names)
scaled_df.describe().T
```

Out[25]:

	count	mean	std	min	25%	50%	75%	max
MedInc	15480.0	2.172968e-16	1.000032	-1.774632	-0.688854	-0.175663	0.464450	5.842113
HouseAge	15480.0	-1.254954e-16	1.000032	-2.188261	-0.840224	0.032036	0.666407	1.855852
AveRooms	15480.0	-1.148163e-16	1.000032	-1.877586	-0.407008	-0.083940	0.257082	56.357392
AveBedrms	15480.0	1.239408e-16	1.000032	-1.740123	-0.205765	-0.108332	0.007435	55.925392
Population	15480.0	-7.874838e-17	1.000032	-1.246395	-0.558886	-0.227928	0.262056	29.971725
AveOccup	15480.0	2.672550e-17	1.000032	-0.201946	-0.056581	-0.024172	0.014501	103.737365
Latitude	15480.0	8.022581e-16	1.000032	-1.451215	-0.799820	-0.645172	0.971601	2.953905
Longitude	15480.0	2.169625e-15	1.000032	-2.380303	-1.106817	0.536231	0.785934	2.633738

Training and Predicting KNN Regression

In [26]:
```
from sklearn.neighbors import KNeighborsRegressor
regressor = KNeighborsRegressor(n_neighbors=5, weights="distance")
regressor.fit(X_train, y_train)
```

Out[26]:
```
KNeighborsRegressor(weights='distance')
```

In [27]:
```python
y_pred = regressor.predict(X_test)
```

Evaluating the Algorithm for KNN Regression

In [28]:
```python
from sklearn.metrics import mean_absolute_error, mean_squared_error

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)

print(f'mae: {mae}')
print(f'mse: {mse}')
print(f'rmse: {rmse}')
```

```
mae: 0.44330658993325084
mse: 0.4284245302766481
rmse: 0.6545414656663457
```

In [29]:
```python
regressor.score(X_test, y_test)
```

Out[29]:
```
0.6762253110912666
```

In [30]:
```python
y.describe()
```

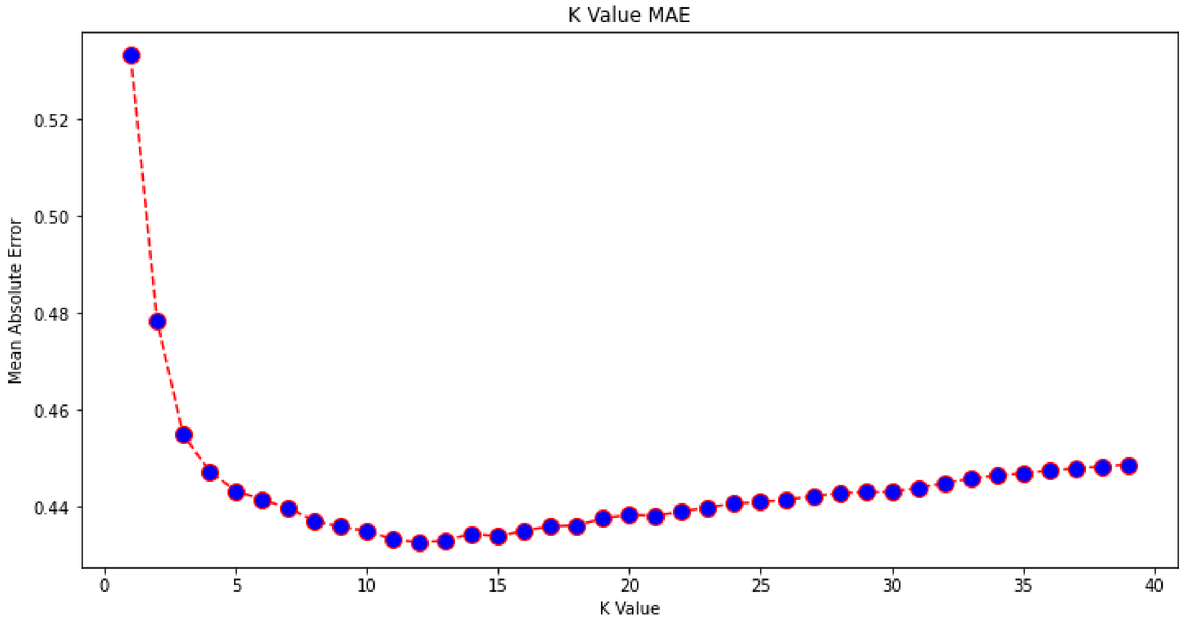
Out[30]:
```
count    20640.000000
mean         2.068558
std          1.153956
min          0.149990
25%          1.196000
50%          1.797000
75%          2.647250
max          5.000010
Name: MedHouseVal, dtype: float64
```

Finding the Best K for KNN Regression

In [31]:
```python
error = []

# Calculating MAE error for K values between 1 and 39
for i in range(1, 40):
    knn = KNeighborsRegressor(n_neighbors=i, weights="distance")
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    mae = mean_absolute_error(y_test, pred_i)
    error.append(mae)
```

In [32]:
```python
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
plt.plot(range(1, 40), error, color='red',
         linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)

plt.title('K Value MAE')
plt.xlabel('K Value')
plt.ylabel('Mean Absolute Error')
```

Out[32]: Text(0, 0.5, 'Mean Absolute Error')



In [33]:
```
import numpy as np

print(min(error))
print(np.array(error).argmin())
```

```
0.43265872078512396
11
```

KNN with 12 neighbours

In [34]:
```
knn_reg12 = KNeighborsRegressor(n_neighbors=12, weights="distance")
knn_reg12.fit(X_train, y_train)
y_pred12 = knn_reg12.predict(X_test)
r2 = knn_reg12.score(X_test, y_test)

mae12 = mean_absolute_error(y_test, y_pred12)
mse12 = mean_squared_error(y_test, y_pred12)
rmse12 = mean_squared_error(y_test, y_pred12, squared=False)
print(f'r2: {r2}, \nmae: {mae12} \nmse: {mse12} \nrmse: {rmse12}')
```

```
r2: 0.6925746041555878,
mae: 0.43265872078512396
mse: 0.40679084969140783
rmse: 0.6378015754852036
```

Classification using K-Nearest Neighbors with Scikit-Learn

Preprocessing Data for Classification

In [35]:
```
# Creating 4 categories and assigning them to a MedHouseValCat column
df["MedHouseValCat"] = pd.qcut(df["MedHouseVal"], 4, retbins=False, labels=[1, 2, 3
```

In [36]:
```
y = df['MedHouseValCat']
X = df.drop(['MedHouseVal', 'MedHouseValCat'], axis = 1)
```

Splitting Data into Train and Test Sets

In [37]:
```
from sklearn.model_selection import train_test_split

SEED = 42
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_st
```

Feature Scaling for Classification

In [38]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

Training and Predicting for Classification

In [39]:
```
from sklearn.neighbors import KNeighborsClassifier

classifier = KNeighborsClassifier(weights="distance")
classifier.fit(X_train, y_train)
```

Out[39]:
```
KNeighborsClassifier(weights='distance')
```

In [40]:
```
y_pred = classifier.predict(X_test)
```

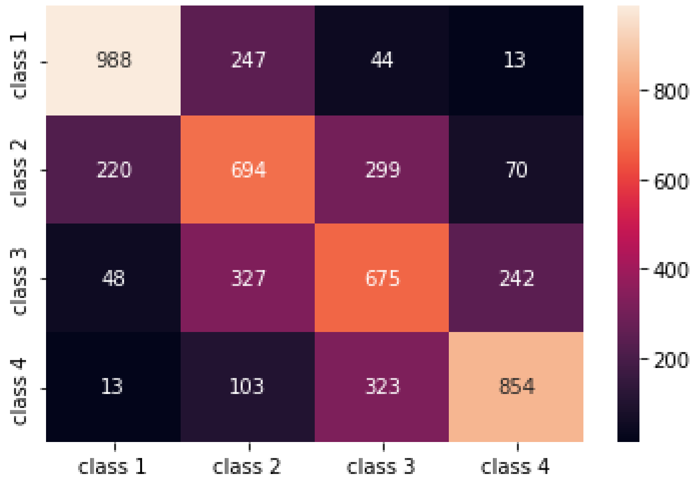
Evaluating KNN for Classification

In [41]:
```
acc =  classifier.score(X_test, y_test)
print(acc)
```

```
0.6222868217054264
```

In [42]:
```
from sklearn.metrics import classification_report, confusion_matrix
#importing Seaborn's to use the heatmap
import seaborn as sns

# Adding classes names for better interpretation
classes_names = ['class 1','class 2','class 3', 'class 4']
cm = pd.DataFrame(confusion_matrix(y_test, y_pred),
                  columns=classes_names, index = classes_names)

# Seaborn's heatmap to better visualize the confusion matrix
sns.heatmap(cm, annot=True, fmt='d');

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           1       0.78      0.76      0.77      1292
           2       0.51      0.54      0.52      1283
           3       0.50      0.52      0.51      1292
           4       0.72      0.66      0.69      1293

    accuracy                           0.62      5160
   macro avg       0.63      0.62      0.62      5160
weighted avg       0.63      0.62      0.62      5160
```

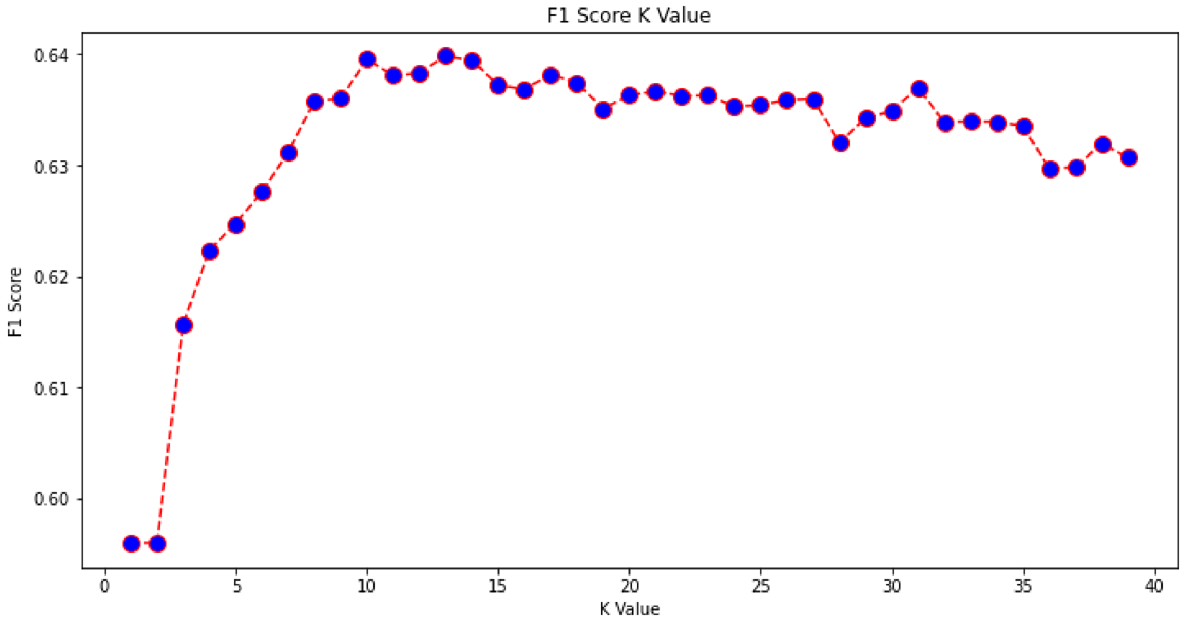


Finding the Best K for KNN Classification

In [43]:
```python
from sklearn.metrics import f1_score

f1s = []

# Calculating f1 score for K values between 1 and 40
for i in range(1, 40):
    knn = KNeighborsClassifier(n_neighbors=i, weights="distance")
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    # using average='weighted' to calculate a weighted average for the 4 classes
    f1s.append(f1_score(y_test, pred_i, average='weighted'))
```

In [44]:
```python
plt.figure(figsize=(12, 6))
plt.plot(range(1, 40), f1s, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('F1 Score K Value')
plt.xlabel('K Value')
plt.ylabel('F1 Score')
```

Out[44]:
```
Text(0, 0.5, 'F1 Score')
```



From the output, we can see that the f1-score is the highest when the value of the K is 10.

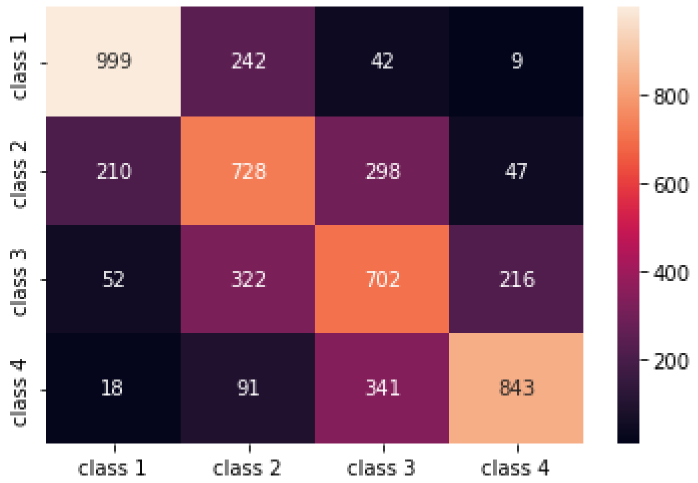
In [48]:
```
classifier15 = KNeighborsClassifier(n_neighbors=10, weights="distance")
classifier15.fit(X_train, y_train)
y_pred15 = classifier15.predict(X_test)
print(classification_report(y_test, y_pred15))
```

```
              precision    recall  f1-score   support

           1       0.79      0.77      0.78      1292
           2       0.53      0.56      0.54      1283
           3       0.51      0.55      0.53      1292
           4       0.74      0.67      0.70      1293

    accuracy                           0.64      5160
   macro avg       0.64      0.64      0.64      5160
weighted avg       0.64      0.64      0.64      5160
```

In [49]:
```
acc =  classifier.score(X_test, y_pred15)
print(acc)
```

```
0.8560077519379845
```

In [47]:
```
cm = pd.DataFrame(confusion_matrix(y_test, y_pred15),
                  columns=classes_names, index = classes_names)

sns.heatmap(cm, annot=True, fmt='d');
```



class 1
class 2
class 3
class 4
999
242
42
9
210
728
298
47
52
322
702
216
18
91
341
843
800
600
400
200

Recommendation

In [21]:
```
import pandas as pd
from scipy.sparse import csr_matrix
from sklearn.neighbors import NearestNeighbors
from fuzzywuzzy import process
```

In [5]:
```
movies = pd.read_csv("./movies.csv", usecols=['movieId', 'title']);
movies.head()
```

Out[5]:

	movieId	title
0	1	Toy Story (1995)
1	2	Jumanji (1995)
2	3	Grumpier Old Men (1995)
3	4	Waiting to Exhale (1995)
4	5	Father of the Bride Part II (1995)

In [6]:
```
ratings = pd.read_csv("./ratings.csv", usecols=['userId', 'movieId', 'rating']);
ratings.head()
```

Out[6]:

	userId	movieId	rating
0	1	1	4.0
1	1	3	4.0
2	1	6	4.0
3	1	47	5.0
4	1	50	5.0

In [7]:
```
movies.shape
```

Out[7]: (9742, 2)

In [9]:
```
ratings.shape
```

Out[9]: (100836, 3)

Create movies_users matrix

In [11]:
```
ratings.pivot(index='movieId', columns='userId', values='rating')
```

Out[11]:

userId	1	2	3	4	5	6	7	8	9	10	...	601	602	603	604
movieId															
1	4.0	NaN	NaN	NaN	4.0	NaN	4.5	NaN	NaN	NaN	...	4.0	NaN	4.0	3.0
2	NaN	NaN	NaN	NaN	NaN	4.0	NaN	4.0	NaN	NaN	...	NaN	4.0	NaN	5.0
3	4.0	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	3.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	3.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
193581	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
193583	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
193585	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
193587	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN
193609	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN

9724 rows × 610 columns

In [14]:
```
movies_users = ratings.pivot(index='movieId', columns='userId', values='rating').fi
movies_users
```

Out[14]:

userId	1	2	3	4	5	6	7	8	9	10	...	601	602	603	604	605	606	607
movieId																		
1	4.0	0.0	0.0	0.0	4.0	0.0	4.5	0.0	0.0	0.0	...	4.0	0.0	4.0	3.0	4.0	2.5	4.0
2	0.0	0.0	0.0	0.0	0.0	4.0	0.0	4.0	0.0	0.0	...	0.0	4.0	0.0	5.0	3.5	0.0	0.0
3	4.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	3.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
193581	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
193583	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
193585	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
193587	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0
193609	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0

9724 rows × 610 columns

In [16]:
```
mat_movies = csr_matrix(movies_users.values)
mat_movies
```

Out[16]:
```
<9724x610 sparse matrix of type '<class 'numpy.float64'>'
	with 100836 stored elements in Compressed Sparse Row format>
```

Create Model

In [22]:
```
model = NearestNeighbors(metric='cosine', algorithm='brute', n_neighbors=20)
model.fit(mat_movies)
```

Out[22]:
```
NearestNeighbors(algorithm='brute', metric='cosine', n_neighbors=20)
```

Item based recommandation

In [32]:
```
def recommender(movie_name, data, n):
    idx = process.extractOne(movie_name, movies['title'])[2]
    print('Movie Selected : ', movies['title'][idx], 'Index : ', idx)
    print("Searching for recommandation.............")
    distance, indices = model.kneighbors(data[idx], n_neighbors=n)
    # print(distance, indices)
    for i in indices:
        print(movies['title'][i].where(i!=idx))
```

In [34]:
```
recommender('iron man', mat_movies, 10)
```

```
Movie Selected :  Iron Man (2008) Index :  6743
Searching for recommandation.............
6743                                            NaN
7197                                 Garage (2007)
7195                       Merry Madagascar (2009)
7354                             A-Team, The (2010)
6726                         Superhero Movie (2008)
7137                        Thirst (Bakjwi) (2009)
7026                                Scorpio (1973)
7571                                Win Win (2011)
3880                    Look Who's Talking Now (1993)
6388    After the Wedding (Efter brylluppet) (2006)
Name: title, dtype: object
```