Predicting Price of House using KNearestNeighbour Regressor

Import Libraries

In [1]:
```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split
```

load dataset

In [2]:
```
url ="https://raw.githubusercontent.com/edyoda/data-science-complete-tutorial/master/Data/house_rental_data.csv.txt"
```

1 . Use pandas to get some insights into the data.

In [3]:
```
df = pd.read_csv(url)
df
```

Out[3]:

	Unnamed: 0	Sqft	Floor	TotalFloor	Bedroom	Living.Room	Bathroom	Price
0	1	1177.698	2	7	2	2	2	62000
1	2	2134.800	5	7	4	2	2	78000
2	3	1138.560	5	7	2	2	1	58000
3	4	1458.780	2	7	3	2	2	45000
4	5	967.776	11	14	3	2	2	45000
...	...	...	...	...	...	...	...	...
640	644	1359.156	7	15	3	2	2	45000
641	645	377.148	4	10	1	1	1	24800
642	646	740.064	13	14	1	1	1	45000
643	647	1707.840	3	14	3	2	2	65000
644	648	1376.946	6	7	3	2	1	36000

645 rows × 8 columns

1.Use pandas to get some insights into the data.

In [4]:
```
df = df.drop('Unnamed: 0', axis=1)
```

In [5]:
```
df
```

Out[5]:

	Sqft	Floor	TotalFloor	Bedroom	Living.Room	Bathroom	Price
0	1177.698	2	7	2	2	2	62000
1	2134.800	5	7	4	2	2	78000
2	1138.560	5	7	2	2	1	58000
3	1458.780	2	7	3	2	2	45000
4	967.776	11	14	3	2	2	45000
...	...	...	...	...	...	...	...
640	1359.156	7	15	3	2	2	45000
641	377.148	4	10	1	1	1	24800
642	740.064	13	14	1	1	1	45000
643	1707.840	3	14	3	2	2	65000
644	1376.946	6	7	3	2	1	36000

645 rows × 7 columns

In [6]:
```
df.shape
```

Out[6]:
```
(645, 7)
```

In [7]:
```
df.dtypes
```

Out[7]:
```
Sqft           float64
Floor            int64
TotalFloor       int64
Bedroom          int64
Living.Room      int64
Bathroom         int64
Price            int64
dtype: object
```

In [8]:
```
df.isnull()
```

Out[8]:

	Sqft	Floor	TotalFloor	Bedroom	Living.Room	Bathroom	Price
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
640	False	False	False	False	False	False	False
641	False	False	False	False	False	False	False
642	False	False	False	False	False	False	False
643	False	False	False	False	False	False	False
644	False	False	False	False	False	False	False

645 rows × 7 columns

In [9]:
```
df.dropna()
```

Out[9]:

	Sqft	Floor	TotalFloor	Bedroom	Living.Room	Bathroom	Price
0	1177.698	2	7	2	2	2	62000
1	2134.800	5	7	4	2	2	78000
2	1138.560	5	7	2	2	1	58000
3	1458.780	2	7	3	2	2	45000
4	967.776	11	14	3	2	2	45000
...	...	...	...	...	...	...	...
640	1359.156	7	15	3	2	2	45000
641	377.148	4	10	1	1	1	24800
642	740.064	13	14	1	1	1	45000
643	1707.840	3	14	3	2	2	65000
644	1376.946	6	7	3	2	1	36000

645 rows × 7 columns

In [10]:
```
df1=df.drop_duplicates()
df1
```

Out[10]:

	Sqft	Floor	TotalFloor	Bedroom	Living.Room	Bathroom	Price
0	1177.698	2	7	2	2	2	62000
1	2134.800	5	7	4	2	2	78000
2	1138.560	5	7	2	2	1	58000
3	1458.780	2	7	3	2	2	45000
4	967.776	11	14	3	2	2	45000
...	...	...	...	...	...	...	...
639	2846.400	5	12	4	2	2	138888
640	1359.156	7	15	3	2	2	45000
641	377.148	4	10	1	1	1	24800
642	740.064	13	14	1	1	1	45000
644	1376.946	6	7	3	2	1	36000

579 rows × 7 columns

In [11]:
```
df1.describe(include='all')
```

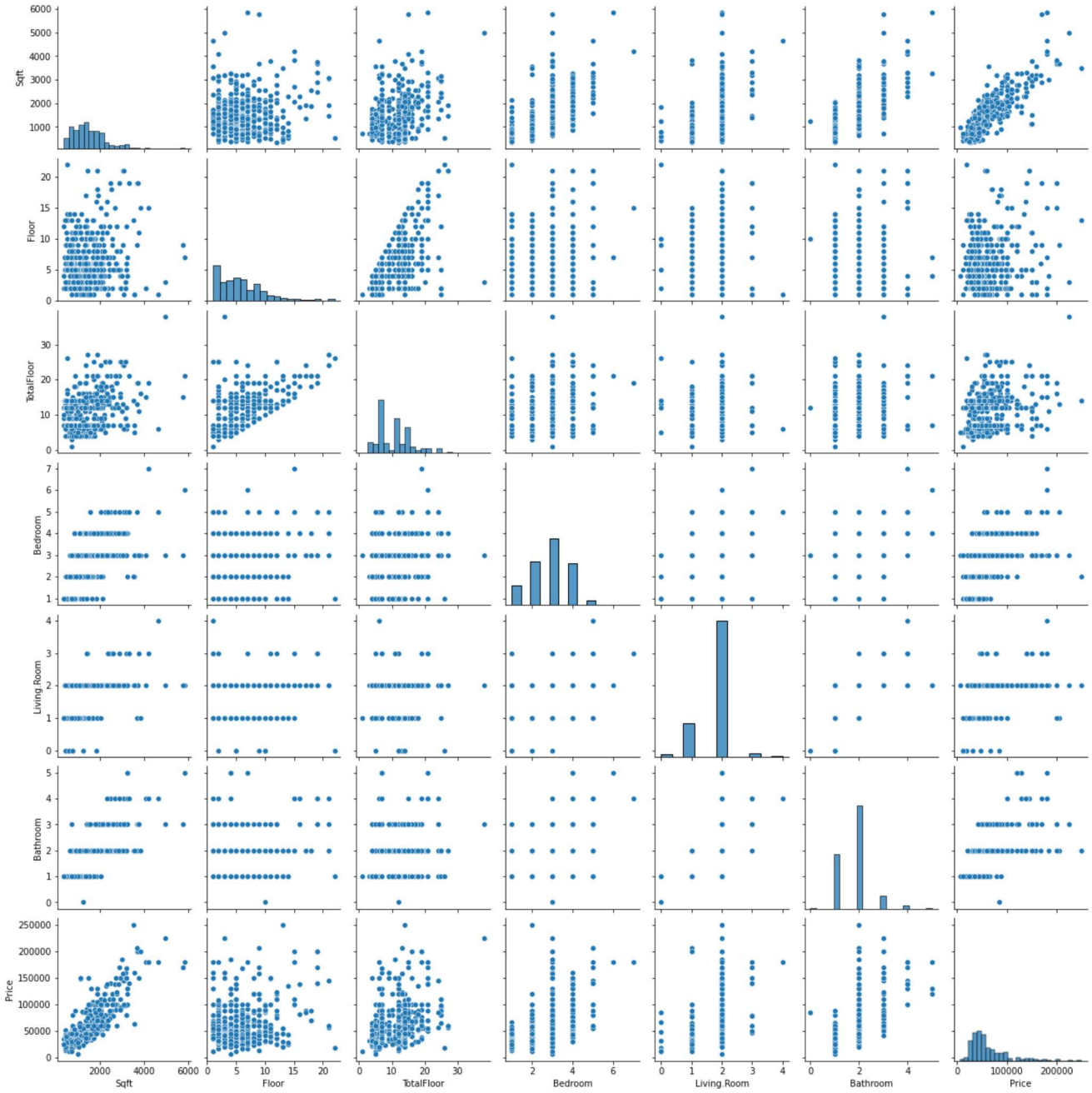
Out[11]:

	Sqft	Floor	TotalFloor	Bedroom	Living.Room	Bathroom	Price
count	579.000000	579.000000	579.000000	579.000000	579.000000	579.00000	579.000000
mean	1516.918580	5.908463	10.778929	2.825561	1.803109	1.80829	61383.614853
std	776.655587	3.893511	5.027210	1.018328	0.477077	0.68816	35764.023105
min	359.358000	1.000000	1.000000	1.000000	0.000000	0.00000	6100.000000
25%	925.080000	3.000000	7.000000	2.000000	2.000000	1.00000	38000.000000
50%	1419.642000	5.000000	11.000000	3.000000	2.000000	2.00000	50000.000000
75%	1891.077000	8.000000	14.000000	4.000000	2.000000	2.00000	75000.000000
max	5856.468000	22.000000	38.000000	7.000000	4.000000	5.00000	250000.000000

2. Show some interesting visualization of the data.

In [12]:
```
sns.pairplot(df1)
```

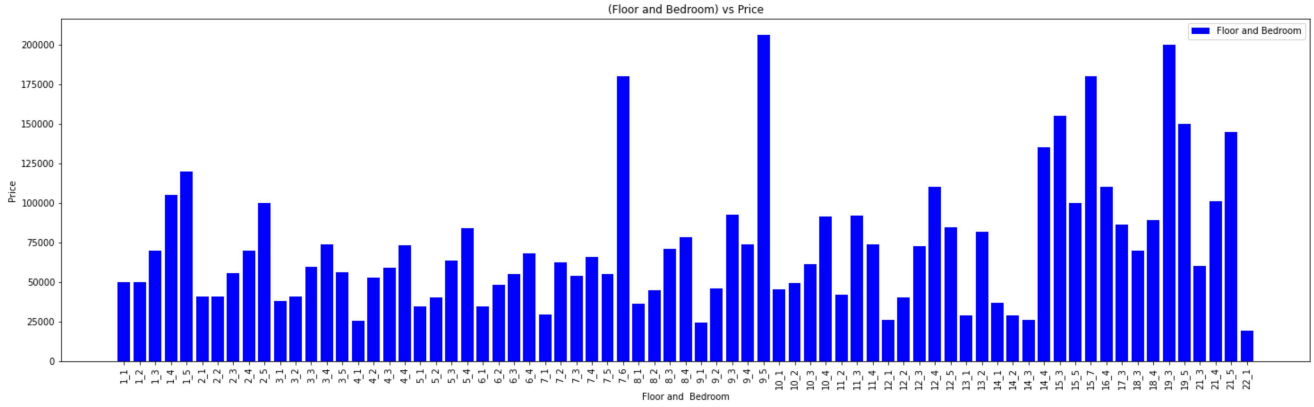
Out[12]:
```
<seaborn.axisgrid.PairGrid at 0x127b19d4910>
```



Making a different - 2 group of column for check price.

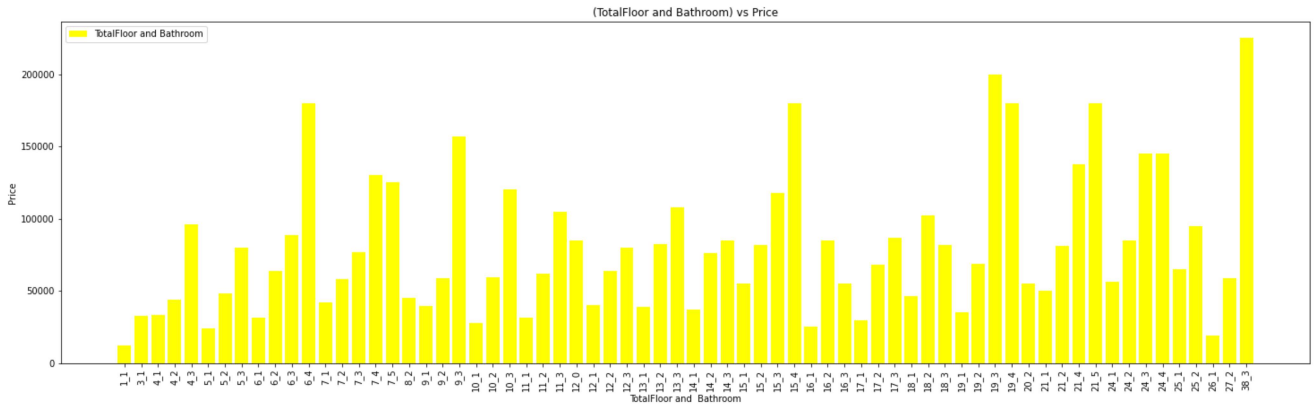
In [13]:
```
df_gp1 = df1.groupby(['Floor','Bedroom'], as_index = False)['Price'].mean()
plt.figure(figsize = (25,7))
plt.bar(df_gp1['Floor'].astype(str)+'_'+df_gp1['Bedroom'].astype(str),df_gp1['Price'],color='blue')
plt.xticks(rotation=90)
plt.title("(Floor and Bedroom) vs Price")
plt.xlabel('Floor and  Bedroom')
plt.ylabel('Price')
plt.legend(['Floor and Bedroom'])
```

Out[13]: <matplotlib.legend.Legend at 0x127b6a96a30>



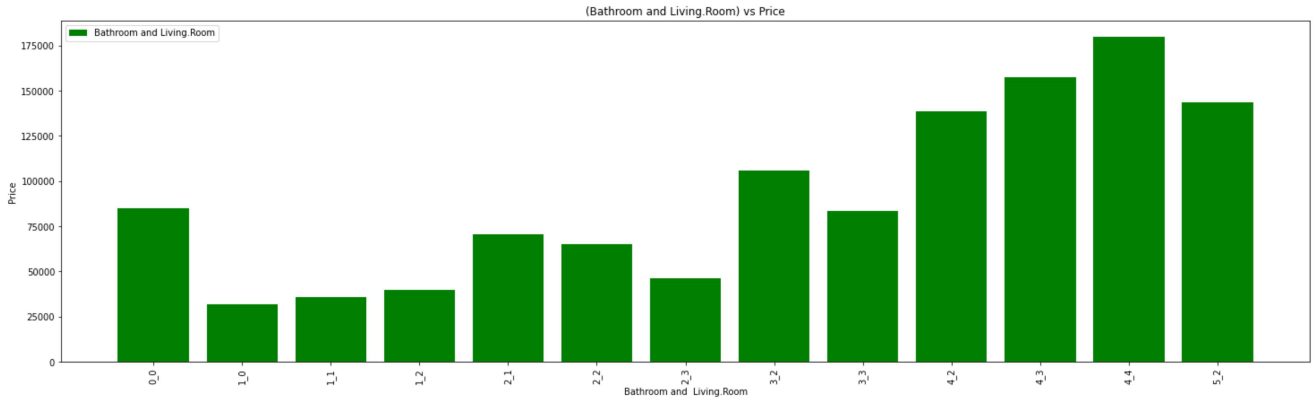
In [14]:
```
df_gp2 = df1.groupby(['TotalFloor','Bathroom'], as_index = False)['Price'].mean()
plt.figure(figsize = (25,7))
plt.bar(df_gp2['TotalFloor'].astype(str)+'_'+df_gp2['Bathroom'].astype(str),df_gp2['Price'],color='yellow')
plt.xticks(rotation=90)
plt.title("(TotalFloor and Bathroom) vs Price")
plt.xlabel('TotalFloor and  Bathroom')
plt.ylabel('Price')
plt.legend(['TotalFloor and Bathroom'])
```

Out[14]: <matplotlib.legend.Legend at 0x127b6ff4670>



In [15]:
```
df_gp3 = df1.groupby(['Bathroom','Living.Room'], as_index = False)['Price'].mean()
plt.figure(figsize = (25,7))
plt.bar(df_gp3['Bathroom'].astype(str)+'_'+df_gp3['Living.Room'].astype(str),df_gp3['Price'],color='green')
plt.xticks(rotation=90)
plt.title("(Bathroom and Living.Room) vs Price")
plt.xlabel('Bathroom and  Living.Room')
plt.ylabel('Price')
plt.legend(['Bathroom and Living.Room'])
```

Out[15]:
```
<matplotlib.legend.Legend at 0x127b6d5bb20>
```



3. Manage data for training & testing

Split dataset

In [16]:
```
X = df1.drop('Price', axis=1)
y = df1['Price']
```

In [17]:
```
#from sklearn import preprocessing
#X = preprocessing.StandardScaler().fit(X).transform(X.astype(float))
```

In [18]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 1)
```

In [19]:
```
print(X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)
```

```
(405, 6) (405,)
(174, 6) (174,)
```

In [20]:
```
X
```

Out[20]:

	Sqft	Floor	TotalFloor	Bedroom	Living.Room	Bathroom
0	1177.698	2	7	2	2	2
1	2134.800	5	7	4	2	2
2	1138.560	5	7	2	2	1
3	1458.780	2	7	3	2	2
4	967.776	11	14	3	2	2
...	...	...	...	...	...	...
639	2846.400	5	12	4	2	2
640	1359.156	7	15	3	2	2
641	377.148	4	10	1	1	1
642	740.064	13	14	1	1	1
644	1376.946	6	7	3	2	1

579 rows × 6 columns

In [21]:
```
y
```

Out[21]:
```
0       62000
1       78000
2       58000
3       45000
4       45000
        ...
639    138888
640     45000
641     24800
642     45000
644     36000
Name: Price, Length: 579, dtype: int64
```

In [22]:
```
y.info()
```

```
<class 'pandas.core.series.Series'>
Int64Index: 579 entries, 0 to 644
Series name: Price
Non-Null Count  Dtype
--------------  -----
579 non-null    int64
dtypes: int64(1)
memory usage: 9.0 KB
```

import the model

In [23]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
```

In [24]:
```
#Train Model and Predict
k = 1
neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
Pred_y = neigh.predict(X_test)
print("Accuracy of model at K = 1 is",metrics.accuracy_score(y_test, Pred_y))
```

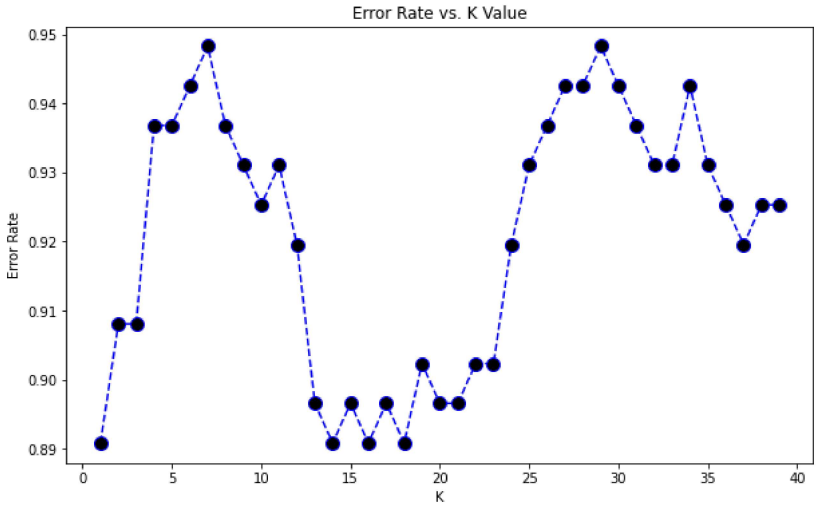
```
Accuracy of model at K = 1 is 0.10919540229885058
```

4 . Finding a better value of k.

In [25]:
```
error_rate = []
for i in range(1,40):
 knn = KNeighborsClassifier(n_neighbors=i)
 knn.fit(X_train,y_train)
 pred_i = knn.predict(X_test)
 error_rate.append(np.mean(pred_i != y_test))
```

In [26]:
```
plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate,color='blue', linestyle='dashed',
         marker='o',markerfacecolor='black', markersize=10)
plt.title('Error Rate vs. K Value')
plt.xlabel('K')
plt.ylabel('Error Rate')
print("Minimum error:-",min(error_rate),"at K =",error_rate.index(min(error_rate)))
```

```
Minimum error:- 0.8908045977011494 at K = 0
```



In [27]:
```
acc = []
# Will take some time
from sklearn import metrics
for i in range(1,40):
    neigh = KNeighborsClassifier(n_neighbors = i).fit(X_train,y_train)
    yhat = neigh.predict(X_test)
    acc.append(metrics.accuracy_score(y_test, yhat))

plt.figure(figsize=(10,6))
plt.plot(range(1,40),acc,color = 'blue',linestyle='dashed',
         marker='o',markerfacecolor='black', markersize=10)
plt.title('accuracy vs. K Value')
plt.xlabel('K')
plt.ylabel('Accuracy')
print("Maximum accuracy:-",max(acc),"at K =",acc.index(max(acc)))
```

```
Maximum accuracy:- 0.10919540229885058 at K = 0
```