CSCI 6515 Machine Learning for Big Data Final Project

Data Preprocessing

In [10]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
```

After importing the necessary python libraries for Data Preprocessing, the retrieved dataset is converted from CSV into a pandas Dataframe. This is usually used in machine learning for viewing any dataset and allows easy preprocessing of data.

In [11]:
```
df = pd.read_csv("housing_train.csv")
print(df)
```

```
                id                                                url  \
0       7039061606  https://bham.craigslist.org/apa/d/birmingham-h...
1       7041970863  https://bham.craigslist.org/apa/d/birmingham-w...
2       7041966914  https://bham.craigslist.org/apa/d/birmingham-g...
3       7041966936  https://bham.craigslist.org/apa/d/birmingham-f...
4       7041966888  https://bham.craigslist.org/apa/d/birmingham-2...
...            ...                                                ...
265185  7050851033  https://columbus.craigslist.org/apa/d/columbus...
265186  7050887997  https://columbus.craigslist.org/apa/d/grove-ci...
265187  7044801015  https://columbus.craigslist.org/apa/d/columbus...
265188  7050885800  https://columbus.craigslist.org/apa/d/newark-1...
265189  7050884586  https://columbus.craigslist.org/apa/d/columbus...

            region                         region_url  price       type  sqfeet  \
0       birmingham    https://bham.craigslist.org   1195  apartment    1908
1       birmingham    https://bham.craigslist.org   1120  apartment    1319
2       birmingham    https://bham.craigslist.org    825  apartment    1133
3       birmingham    https://bham.craigslist.org    800  apartment     927
4       birmingham    https://bham.craigslist.org    785  apartment    1047
...            ...                            ...    ...        ...     ...
265185    columbus  https://columbus.craigslist.org      0  apartment    1061
265186    columbus  https://columbus.craigslist.org   1069  apartment    1020
265187    columbus  https://columbus.craigslist.org   1507  apartment    1660
265188    columbus  https://columbus.craigslist.org   1001  apartment    1220
265189    columbus  https://columbus.craigslist.org   1164  townhouse    1300

        beds  baths  cats_allowed  ...  wheelchair_access  \
0          3    2.0             1  ...                  0
1          3    2.0             1  ...                  0
2          1    1.5             1  ...                  0
3          1    1.0             1  ...                  0
4          2    1.0             1  ...                  0
...      ...    ...           ...  ...                ...
265185     2    2.0             1  ...                  0
265186     2    1.5             1  ...                  0
265187     2    1.5             1  ...                  0
265188     3    1.5             1  ...                  0
265189     2    2.5             1  ...                  0

        electric_vehicle_charge  comes_furnished  laundry_options  \
0                             0                0  laundry on site
1                             0                0  laundry on site
2                             0                0  laundry on site
3                             0                0  laundry on site
4                             0                0  laundry on site
...                         ...              ...              ...
265185                        0                0       w/d in unit
265186                        0                0       w/d hookups
265187                        0                0              NaN
265188                        0                0       w/d hookups
265189                        0                0       w/d hookups

           parking_options                                          image_url  \
0           street parking  https://images.craigslist.org/00L0L_80pNkyDeG0...
1         off-street parking  https://images.craigslist.org/00707_uRrY9CsNMC...
2           street parking  https://images.craigslist.org/00h0h_b7Bdj1NLBi...
3           street parking  https://images.craigslist.org/00808_6ghZ8tSRQs...
4           street parking  https://images.craigslist.org/00y0y_21c0FOvUXm...
...                    ...                                                ...
265185     detached garage  https://images.craigslist.org/00m0m_8wP9hX96TZ...
265186     detached garage  https://images.craigslist.org/00N0N_1oCvFNCv24...
265187                 NaN  https://images.craigslist.org/00j0j_5EIRNrdHCV...
265188  off-street parking  https://images.craigslist.org/00J0J_6VSTqFJb86...
265189     attached garage  https://images.craigslist.org/00w0w_h1L3gxb8rG...

                                               description      lat     long  \
0        Apartments In Birmingham AL Welcome to 100 Inv...  33.4226 -86.7065
1        Find Your Way to Haven Apartment Homes Come ho...  33.3755 -86.8045
2        Apartments In Birmingham AL Welcome to 100 Inv...  33.4226 -86.7065
3        Apartments In Birmingham AL Welcome to 100 Inv...  33.4226 -86.7065
4        Apartments In Birmingham AL Welcome to 100 Inv...  33.4226 -86.7065
...                                                    ...      ...      ...
265185   &[?][?] BERKLEY HOUSE &[?][?]   Voted #1 BEST Communit...  40.0495 -83.0669
265186   !!!!Tour today and receive $5 Starbucks gift c...  39.8408 -83.0804
265187   The Commons at Olentangy4765 Blairfield Dr, Co...  40.0564 -83.0417
265188   www.McMillenWoods.com  www.mcmillenwoods.com  ...  40.0451 -82.4564
265189   Park Club apartments is offering some of the B...      NaN      NaN

       state
0         al
1         al
2         al
3         al
4         al
...      ...
265185    oh
265186    oh
265187    oh
265188    oh
265189   NaN

[265190 rows x 22 columns]
```

The above dataset consists of data about different types of rental units in various states of the US. For the first step of preprocessing the data, a thorough analysis of the dataset was made and unnecessary columns, columns that will not be useful to the purpose of this project, were dropped. These columns included URLs from which the information displayed in the other column was taken and also the description column was removed. This latter was as the aim of this project does not involve analysing textual data to predict the rent of the house. The aim of this project is to use other factors that are detrimental to a person to choose a rental unit as a living space. Some of those factors include the number of bedrooms and bathrooms, the type of the unit, the area and other available amenities as shown in the retained columns in the dataframe below.

In [12]:
```
df = df.drop(['url', 'region_url', 'image_url', 'description'], axis=1)
print(df)
```

```
                id      region  price       type  sqfeet  beds  baths  \
0       7039061606  birmingham   1195  apartment    1908     3    2.0
1       7041970863  birmingham   1120  apartment    1319     3    2.0
2       7041966914  birmingham    825  apartment    1133     1    1.5
3       7041966936  birmingham    800  apartment     927     1    1.0
4       7041966888  birmingham    785  apartment    1047     2    1.0
...            ...         ...    ...        ...     ...   ...    ...
265185  7050851033    columbus      0  apartment    1061     2    2.0
265186  7050887997    columbus   1069  apartment    1020     2    1.5
265187  7044801015    columbus   1507  apartment    1660     2    1.5
265188  7050885800    columbus   1001  apartment    1220     3    1.5
265189  7050884586    columbus   1164  townhouse    1300     2    2.5

        cats_allowed  dogs_allowed  smoking_allowed  wheelchair_access  \
0                  1             1                1                  0
1                  1             1                1                  0
2                  1             1                1                  0
3                  1             1                1                  0
4                  1             1                1                  0
...              ...           ...              ...                ...
265185             1             1                1                  0
265186             1             1                1                  0
265187             1             1                1                  0
265188             1             1                1                  0
265189             1             1                0                  0

        electric_vehicle_charge  comes_furnished  laundry_options  \
0                             0                0  laundry on site
1                             0                0  laundry on site
2                             0                0  laundry on site
3                             0                0  laundry on site
4                             0                0  laundry on site
...                         ...              ...              ...
265185                        0                0      w/d in unit
265186                        0                0      w/d hookups
265187                        0                0              NaN
265188                        0                0      w/d hookups
265189                        0                0      w/d hookups

           parking_options      lat     long state
0           street parking  33.4226 -86.7065    al
1       off-street parking  33.3755 -86.8045    al
2           street parking  33.4226 -86.7065    al
3           street parking  33.4226 -86.7065    al
4           street parking  33.4226 -86.7065    al
...                    ...      ...      ...   ...
265185     detached garage  40.0495 -83.0669    oh
265186     detached garage  39.8408 -83.0804    oh
265187                 NaN  40.0564 -83.0417    oh
265188  off-street parking  40.0451 -82.4564    oh
265189     attached garage      NaN      NaN   NaN

[265190 rows x 18 columns]
```

The next step was to remove rows for which any of its corresponding column value is null. Null values decrease the meaningfullness of the data. I have removed them because. I am considering all of the columns, the factors, to predict the price of a given rental unit. Having one of them as Null or replaced with 0 will not be fair in assesing the dataset by a machine learning model

In [13]:
```
for column in df.columns:
    df = df[df[column].notna()]
print(df)
```

```
                id      region  price       type  sqfeet  beds  baths  \
0       7039061606  birmingham   1195  apartment    1908     3    2.0
1       7041970863  birmingham   1120  apartment    1319     3    2.0
2       7041966914  birmingham    825  apartment    1133     1    1.5
3       7041966936  birmingham    800  apartment     927     1    1.0
4       7041966888  birmingham    785  apartment    1047     2    1.0
...            ...         ...    ...        ...     ...   ...    ...
265183  7049194586    columbus    870  apartment     933     2    2.0
265184  7050888256    columbus    929  apartment     728     1    1.0
265185  7050851033    columbus      0  apartment    1061     2    2.0
265186  7050887997    columbus   1069  apartment    1020     2    1.5
265188  7050885800    columbus   1001  apartment    1220     3    1.5

        cats_allowed  dogs_allowed  smoking_allowed  wheelchair_access  \
0                  1             1                1                  0
1                  1             1                1                  0
2                  1             1                1                  0
3                  1             1                1                  0
4                  1             1                1                  0
...              ...           ...              ...                ...
265183             1             1                1                  0
265184             1             1                1                  1
265185             1             1                1                  0
265186             1             1                1                  0
265188             1             1                1                  0

        electric_vehicle_charge  comes_furnished  laundry_options  \
0                             0                0  laundry on site
1                             0                0  laundry on site
2                             0                0  laundry on site
3                             0                0  laundry on site
4                             0                0  laundry on site
...                         ...              ...              ...
265183                        0                0      w/d hookups
265184                        0                0      w/d in unit
265185                        0                0      w/d in unit
265186                        0                0      w/d hookups
265188                        0                0      w/d hookups

           parking_options      lat     long state
0           street parking  33.4226 -86.7065    al
1       off-street parking  33.3755 -86.8045    al
2           street parking  33.4226 -86.7065    al
3           street parking  33.4226 -86.7065    al
4           street parking  33.4226 -86.7065    al
...                    ...      ...      ...   ...
265183      street parking  39.8971 -82.8957    oh
265184  off-street parking  39.9709 -82.9241    oh
265185     detached garage  40.0495 -83.0669    oh
265186     detached garage  39.8408 -83.0804    oh
265188  off-street parking  40.0451 -82.4564    oh

[164284 rows x 18 columns]
```

This step involves removing rows for which the rent is less than five hundred dollars. This was based on a real world understanding of the data that most rental units atleast cost five hundred. This was used to limit the number of rows further to avoid memory errors as well.

In [14]:
```
df = df[df['price'] >= 500]
print(df)
```

```
                id      region  price       type  sqfeet  beds  baths  \
0       7039061606  birmingham   1195  apartment    1908     3    2.0
1       7041970863  birmingham   1120  apartment    1319     3    2.0
2       7041966914  birmingham    825  apartment    1133     1    1.5
3       7041966936  birmingham    800  apartment     927     1    1.0
4       7041966888  birmingham    785  apartment    1047     2    1.0
...            ...         ...    ...        ...     ...   ...    ...
265182  7050888285    columbus   1719  apartment    1630     3    2.5
265183  7049194586    columbus    870  apartment     933     2    2.0
265184  7050888256    columbus    929  apartment     728     1    1.0
265186  7050887997    columbus   1069  apartment    1020     2    1.5
265188  7050885800    columbus   1001  apartment    1220     3    1.5

        cats_allowed  dogs_allowed  smoking_allowed  wheelchair_access  \
0                  1             1                1                  0
1                  1             1                1                  0
2                  1             1                1                  0
3                  1             1                1                  0
4                  1             1                1                  0
...              ...           ...              ...                ...
265182             1             1                1                  1
265183             1             1                1                  0
265184             1             1                1                  1
265186             1             1                1                  0
265188             1             1                1                  0

        electric_vehicle_charge  comes_furnished  laundry_options  \
0                             0                0  laundry on site
1                             0                0  laundry on site
2                             0                0  laundry on site
3                             0                0  laundry on site
4                             0                0  laundry on site
...                         ...              ...              ...
265182                        0                0      w/d in unit
265183                        0                0      w/d hookups
265184                        0                0      w/d in unit
265186                        0                0      w/d hookups
265188                        0                0      w/d hookups

           parking_options      lat     long state
0           street parking  33.4226 -86.7065    al
1       off-street parking  33.3755 -86.8045    al
2           street parking  33.4226 -86.7065    al
3           street parking  33.4226 -86.7065    al
4           street parking  33.4226 -86.7065    al
...                    ...      ...      ...   ...
265182  off-street parking  39.9709 -82.9241    oh
265183      street parking  39.8971 -82.8957    oh
265184  off-street parking  39.9709 -82.9241    oh
265186     detached garage  39.8408 -83.0804    oh
265188  off-street parking  40.0451 -82.4564    oh

[160147 rows x 18 columns]
```

Another major step in data preprocessing is ensuring all the columns are in a format that can be easily comprehended by a machine learning algorithm. This usually means ensuring that the columns with textual content is either vectorized or encoded in a numeric form. This step includes selecting all categorical columns and storing them in a list.

In [15]:
```
textList = df.select_dtypes(exclude=[np.number]).columns.tolist()
print(textList)
```

```
['region', 'type', 'laundry_options', 'parking_options', 'state']
```

This list contains all the categorical (textual) columns present in the final dataset. For each of the columns, Label Encoding is performed. Label Encoding is an encoding technique in the sklearn library in python. It is mostly used for categorical columns. It is used to assign a unique numeric label to each string available in that column.

In [16]:
```
for column in textList:
    le = LabelEncoder()
    le = le.fit(df[column])
    df[column] = le.fit_transform(df[column])
print(df)
```

```
                id  region  price  type  sqfeet  beds  baths  cats_allowed  \
0       7039061606      21   1195     0    1908     3    2.0             1
1       7041970863      21   1120     0    1319     3    2.0             1
2       7041966914      21    825     0    1133     1    1.5             1
3       7041966936      21    800     0     927     1    1.0             1
4       7041966888      21    785     0    1047     2    1.0             1
...            ...     ...    ...   ...     ...   ...    ...           ...
265182  7050888285      52   1719     0    1630     3    2.5             1
265183  7049194586      52    870     0     933     2    2.0             1
265184  7050888256      52    929     0     728     1    1.0             1
265186  7050887997      52   1069     0    1020     2    1.5             1
265188  7050885800      52   1001     0    1220     3    1.5             1

        dogs_allowed  smoking_allowed  wheelchair_access  \
0                  1                1                  0
1                  1                1                  0
2                  1                1                  0
3                  1                1                  0
4                  1                1                  0
...              ...              ...                ...
265182             1                1                  1
265183             1                1                  0
265184             1                1                  1
265186             1                1                  0
265188             1                1                  0

        electric_vehicle_charge  comes_furnished  laundry_options  \
0                             0                0                1
1                             0                0                1
2                             0                0                1
3                             0                0                1
4                             0                0                1
...                         ...              ...              ...
265182                        0                0                4
265183                        0                0                3
265184                        0                0                4
265186                        0                0                3
265188                        0                0                3

        parking_options      lat     long  state
0                     5  33.4226 -86.7065      1
1                     4  33.3755 -86.8045      1
2                     5  33.4226 -86.7065      1
3                     5  33.4226 -86.7065      1
4                     5  33.4226 -86.7065      1
...                 ...      ...      ...    ...
265182                4  39.9709 -82.9241     35
265183                5  39.8971 -82.8957     35
265184                4  39.9709 -82.9241     35
265186                2  39.8408 -83.0804     35
265188                4  40.0451 -82.4564     35

[160147 rows x 18 columns]
```

After all these steps are completed, the data has been preprocessed and is now ready for application with machine learning algorithms. For easy readability the final clean datframe is converted into a separate CSV file.

In [17]:
```
df.to_csv("Cleaned_Rent_Dataset_Final.csv", index=False)
```

Clustering

After preprocessing is done, Clustering needs to be done to group the data into similar clusters and to observe how the data is related to each other. All necessary libraries need to accomplish this task are imported below. For the purpose of this project I have chosen the KMeans clustering algorithm.

In [18]:
```
import pandas as pd
from sklearn import metrics
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

The cleaned CSV dataset is again read as a Pandas Dataframe to facilitate easy processing of the data in the further tasks to be accomplished for clustering and later for the application of machine learning.

In [19]:
```
df = pd.read_csv("Cleaned_Rent_Dataset_Final.csv")
print(df)
```

```
                id  region  price  type  sqfeet  beds  baths  cats_allowed  \
0       7039061606      21   1195     0    1908     3    2.0             1
1       7041970863      21   1120     0    1319     3    2.0             1
2       7041966914      21    825     0    1133     1    1.5             1
3       7041966936      21    800     0     927     1    1.0             1
4       7041966888      21    785     0    1047     2    1.0             1
...            ...     ...    ...   ...     ...   ...    ...           ...
160142  7050888285      52   1719     0    1630     3    2.5             1
160143  7049194586      52    870     0     933     2    2.0             1
160144  7050888256      52    929     0     728     1    1.0             1
160145  7050887997      52   1069     0    1020     2    1.5             1
160146  7050885800      52   1001     0    1220     3    1.5             1

        dogs_allowed  smoking_allowed  wheelchair_access  \
0                  1                1                  0
1                  1                1                  0
2                  1                1                  0
3                  1                1                  0
4                  1                1                  0
...              ...              ...                ...
160142             1                1                  1
160143             1                1                  0
160144             1                1                  1
160145             1                1                  0
160146             1                1                  0

        electric_vehicle_charge  comes_furnished  laundry_options  \
0                             0                0                1
1                             0                0                1
2                             0                0                1
3                             0                0                1
4                             0                0                1
...                         ...              ...              ...
160142                        0                0                4
160143                        0                0                3
160144                        0                0                4
160145                        0                0                3
160146                        0                0                3

        parking_options      lat     long  state
0                     5  33.4226 -86.7065      1
1                     4  33.3755 -86.8045      1
2                     5  33.4226 -86.7065      1
3                     5  33.4226 -86.7065      1
4                     5  33.4226 -86.7065      1
...                 ...      ...      ...    ...
160142                4  39.9709 -82.9241     35
160143                5  39.8971 -82.8957     35
160144                4  39.9709 -82.9241     35
160145                2  39.8408 -83.0804     35
160146                4  40.0451 -82.4564     35

[160147 rows x 18 columns]
```

As the dataset is huge, some of the sklearn algorithms do not work with the complete dataset and cause a memory error. To avoid this but to still achieve the right results , i am limiting the number of rows considered to around 5000 rows.

In [20]:
```
cdf = df.iloc[:5000, :]
```

I have now created a model to perform clustering using KMeans algorithm. Kmeans algorithms tries to sample the data into a number of groups where the data is of equal variance. I have now clustered the entire dataset into three groups i.e., three clusters. The data is first fitted and transformed from the pandas dataframe into the Kmeans Model. A metric to evaluate the score is chosen from sklearn. The silhouette score computes the mean silhouette coefficeint of all the given data in the dataframe based on a given metric. For the purpose of this experiment I've given the metric to be euclidean as it is more well used metric in data science.

In [21]:
```
model = KMeans(n_clusters=3,  random_state=0)
clusters = model.fit_transform(cdf)
score = metrics.silhouette_score(cdf, model.labels_, metric='euclidean')
```

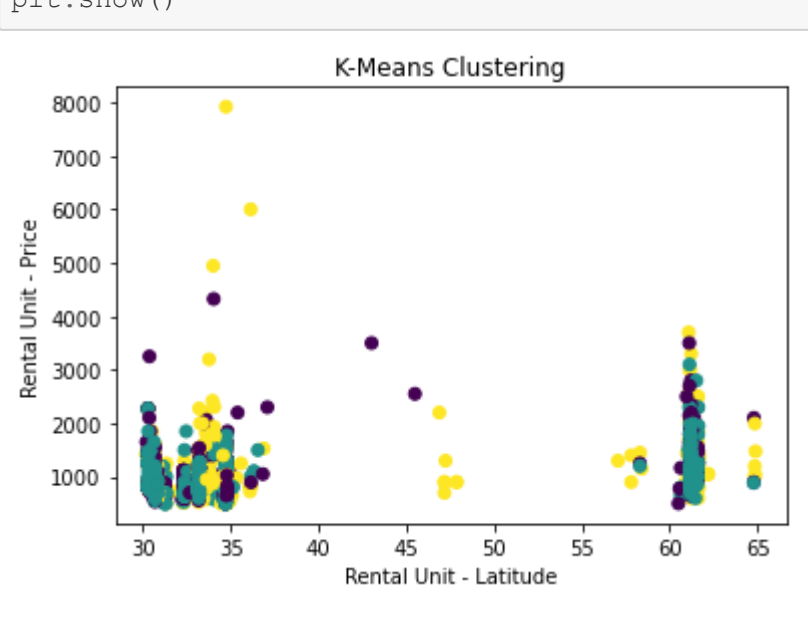
The data from the applied clustering model is stored in dictionary which shows the model name, the computed score and the final clusters formed. Each cluster holds the variance of each of the samples sent to the algorithm.

In [22]:
```
cluster_results = dict(model=model, score=score, clusters=clusters)
print(cluster_results)
```

```
{'model': KMeans(n_clusters=3, random_state=0), 'score': 0.5614956807146259, 'clusters': array([[ 494
8655.96872848, 16047541.21055858,  5275928.05623124],
       [ 7857912.88952276, 18956798.21742058,  2366670.99916824],
       [ 7853963.88545198, 18952849.28814831,  2370619.98714598],
       ...,
       [ 7482498.1310704 ,  3616394.45183072, 17707081.98249638],
       [14690243.12632027,  3591366.48211123, 24914826.98110697],
       [ 8667907.1296843 ,  2430989.62759449, 18892490.98132102]])}
```

To visualize these clusters with respect to the price of the rental unit and the lattitude of the rental unit, which shows that location is one of the salient features for rental unit prediction, a matplotlib graph has been plotted as illustrated below. Here each color denotes values from each cluster group. There are three colors in this graph, each denoting the three clusters that the trained model has clustered.

In [23]:
```
labels = pd.DataFrame(model.labels_)
fig = plt.figure()
ax = fig.add_subplot(111)
scatter = ax.scatter(cdf['lat'], cdf['price'], c=labels)
ax.set_title('K-Means Clustering')
ax.set_xlabel('Rental Unit - Latitude')
ax.set_ylabel('Rental Unit - Price')
plt.show()
```



Relavance of columns in the Dataset

The columns that have been retained as a result of preprocessing and clustering are those that are deemed as potentially useful for the machine learning models to predict the rental unit's price. To analyse the relevance between each of these columns a heatmap was plotted with some of the major column factors as shown below. The correlation matrix projected on this heatmaps holds the values of how these columns are connected to each other

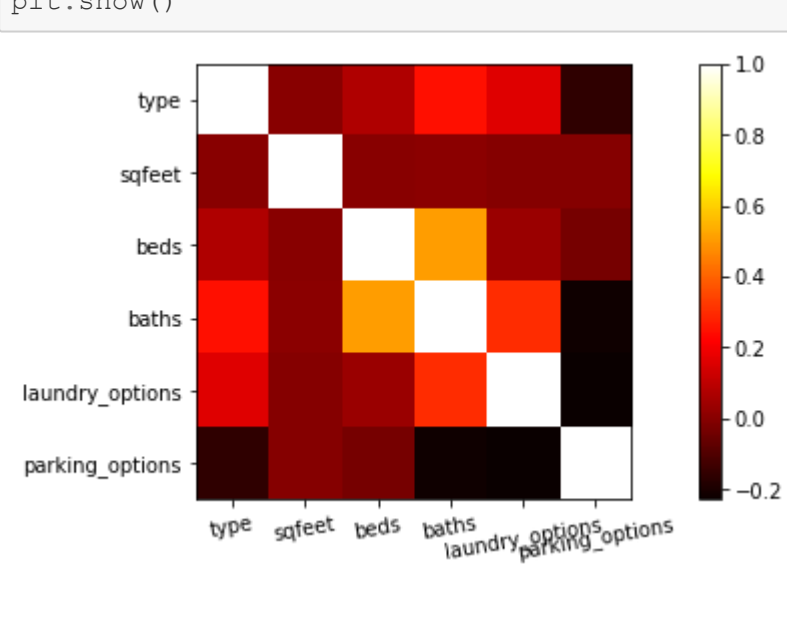
In [24]:
```
cmdf = df.drop(df.columns.difference(['beds','type','sqfeet', 'baths', 'laundry_options', 'parking_opti
ons']), axis=1)
correlation_matrix = cmdf.corr()
print(correlation_matrix)
```

```
                     type    sqfeet      beds     baths  laundry_options  \
type             1.000000  0.004396  0.074024  0.248673         0.162209
sqfeet           0.004396  1.000000  0.002024  0.006236        -0.002874
beds             0.074024  0.002024  1.000000  0.509909         0.039661
baths            0.248673  0.006236  0.509909  1.000000         0.303795
laundry_options  0.162209 -0.002874  0.039661  0.303795         1.000000
parking_options -0.157893 -0.003288 -0.030670 -0.215376        -0.229021

                 parking_options
type                   -0.157893
sqfeet                 -0.003288
beds                   -0.030670
baths                  -0.215376
laundry_options        -0.229021
parking_options         1.000000
```

The correlation matrix displayed above from few of the chosen columns, features, are now plotted as a heatmap using the matplotlib library in python

In [25]:
```
fig, ax = plt.subplots()
ax.set_yticks(np.arange(0, len(cmdf.columns), 1))
ax.set_xticks(np.arange(0, len(cmdf.columns), 1))
ax.set_xticklabels(cmdf.columns, rotation=10)
ax.set_yticklabels(cmdf.columns, fontsize=10)
heatmap = plt.imshow(correlation_matrix, cmap='hot')
plt.colorbar(heatmap, pad=0.1)
plt.show()
```



Machine Learning Models

After clustering and preprocessing, the data needs to be run through a machine learning model to predict that rental price of any asked unit. Regression is one of the major machine learning models that is chosen for prediction of continous group of values, in other words numeric data. Classification is used for prediction of categorical data or textual data. Since all the columns, factors or features influencing the rent of a unit, is numeric, I decided to explore a few numeric algorithms available on sklearn to predict the rental cost of a given unit and also determine the model's corresponding accuracy. I investigated five regression models, each of which is illustrated below with its working. All necessary libraries for these machine learning models are imported below.

In [26]:
```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline
```

The dataframe processed in the above steps is now being prepared to split it for prediction. Here X denotes the dataset that is needed for the model and y denotes the column that is being predicted, in this case the price column.

In [27]:
```
X, y = df.iloc[:, :], df.iloc[:, 2]
```

Each of this data is then split into Training and Testing datasets. The train_test_split function of sklearn is used to achieve this. The testing size of the data is taken as 20 percent of the original dataframe. Both the X and y parts of the data to be used for prediction is split for training and testing.

In [28]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

Machine Learning Model : Decision Forest Regression

Passing the data through a decision tree DecisionTreeRegressor algorithm. The data is first fit into the decision tree regressor model imported from skelearn. The training data is fit into the model to allow the model to align the data and construct the decision tree from that data. Then the testing dataframe is sent to predict the price of the unit. The final results are stored in a dictionary.

In [29]:
```
model = DecisionTreeRegressor()
model.fit(X_train, y_train)
y_predict = model.predict(X_test)
score = model.score(X_test, y_test)
```

The score the decison tree is pretty good with 91% without much need to fine tune the parameters of the model, the test prediction array shows a list of the possible rental prices for the test data that was passed to the model.

In [30]:
```
print(dict(model=model, score=score, test_prediction=y_predict))
```

```
{'model': DecisionTreeRegressor(), 'score': 0.9142911525778237, 'test_prediction': array([1975., 117
5.,  721., ..., 1075., 1650.,  759.])}
```

Machine Learning Model : Random Forest Regression

Passing the data through a RandomForestRegressor Algorithm. The data is first fit into the random forest regressor model imported from skelearn and then the testing dataframe is sent to predict the price of the unit. The final results are stored in a dictionary. A Random Forest a combination of multiple decision trees and it chooses the best score from that tree.

In [31]:
```
modelA = RandomForestRegressor()
modelA.fit(X_train, y_train)
y_predict = modelA.predict(X_test)
score = modelA.score(X_test, y_test)
print(dict(model=modelA, score=score, test_prediction=y_predict))
```

```
{'model': RandomForestRegressor(), 'score': 0.9681751410010944, 'test_prediction': array([1975., 117
5.,  721., ..., 1075., 1650.,  759.])}
```

Machine Learning Model : Linear Regression

Passing the data through a LinearRegression Algorithm. This algorithm predicts the values mathematically with respect to the linear equation constructed from the fitted training dataset and gives a value that satisfies this equation and is part of near to that line. The data is first fit linear regressor model imported from skelearn. Next, the testing dataframe is sent to predict the price of the unit. The final results are stored in a dictionary.

In [32]:
```
modelB = LinearRegression()
modelB.fit(X_train, y_train)
y_predict = modelB.predict(X_test)
score = modelB.score(X_test, y_test)
print(dict(model=modelB, score=score, test_prediction=y_predict))
```

```
{'model': LinearRegression(), 'score': 1.0, 'test_prediction': array([1975.        , 1175.        ,
 720.99999999, ..., 1075.        ,
       1650.        ,  759.        ])}
```

Machine Learning Model : Polynomial Regression

Passing the data through a PolynomialRegression Algorithm. This is the same as linear regression except that first linear regression is done and then an input holding the required number of polynomial features to build a pipeline for the regressor is built. The data is then fitted and tested to predict the price. The results are stored in a dictionary and it can be seen that the polynomial regressor produces a good accuracy.

In [33]:
```
Input = [('polynomial', PolynomialFeatures(degree=2)), ('modal', LinearRegression())]
pipe = Pipeline(Input)
pipe.fit(X_train, y_train)
poly_pred = pipe.predict(X_test)
score = pipe.score(X_test, y_test)
print(dict(model=pipe, score=score, test_prediction=poly_pred))
```

```
{'model': Pipeline(steps=[('polynomial', PolynomialFeatures()),
                ('modal', LinearRegression())]), 'score': 0.9999690426702051, 'test_prediction': arra
y([1976.44293239, 1175.04751195,  722.28708953, ..., 1073.90638462,
       1651.29027306,  759.54268337])}
```

Machine Learning Model : Neural Networks - CHOSEN PROJECT MODEL

Neural Networks is one of the emerging fields in machine learning. Hence I am taking this as the chosen method to implement this rent prediction problem despite the fact that the above models have a better accuracy. There are various ways to implement neural networks such as using Keras with Tensorflow, Pytorch etc. I have chosen to use the neural networks regressor available with sklearn as it much easier to comprehend and apply. Also, the other models that I have implemented also are from sklearn and I would like to learn and observe the difference in the working of various regression models for prediction by sklearn. The necessary files have been imported

In [119]:
```
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import train_test_split
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
```

The data, as done for the above regression models, is split into training and testing datasets and also into X and y prior to that. Here X denotes the entire rows of data to be trained from and y denotes the column that is to be predicted as a result, in this case the rental unit's price. The data is also scaled using the MinMaxScaler to normalize it.

In [120]:
```
X, y = df.iloc[:, :], df.iloc[:, 2]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)

scaler = MinMaxScaler()
X_train_norm = scaler.fit_transform(X_train)
X_test_norm = scaler.fit_transform(X_test)
```

Passing the data through a Neural Networks machine learning model. The various parameters of the neural networks model were adjusted to ensure that the neural network provides this accuracy. This is a supervised learning model where the data is divided as multiple layers which connect with each other. Each layer learns from the previous layer. The forecasters, the factors determining the predictors form the top most layer and the predicted values are the last layer. These emulate the way neurons behave in our nervous system in the human body.

In [121]:
```
regr = MLPRegressor(hidden_layer_sizes = [2], solver='adam', activation='identity', shuffle=False, lea
rning_rate='adaptive')
```

The data after being fit into the MLPRegressor model, the sklearn neural network regressor, is then fitted with the training data. Once the model has been trained the test data is sent to it to retrieve the predicted value of the house along with the score of the trained model's accuracy to predict.

In [122]:
```
regr.fit(X_train, y_train)
value = regr.predict(X_test)
score = regr.score(X_test, y_test)
```

The predicted value and score is then printed to see the results. At every level of the neural network a weight gets added to the outcome. Assuming the weight to be 800, a look of how the actual predicted rent price will be displayed as well.

In [126]:
```
print("Predicted Value excluding weights", value)
print("Score ", score)
weight =800
print("Predicted Value exluding weights ", value-weight)
```

```
Predicted Value excluding weights [2425.43543345 3345.90271336 1468.37958795 ... 2622.84428734 1284.9
9652178
 1594.08553619]
Score  0.9994853993499084
Predicted Value exluding weights  [1625.43543345 2545.90271336  668.37958795 ... 1822.84428734  484.9
9652178
  794.08553619]
```

The Predicted value with reference to the trained data is predicted in a scatter plot as show below. The graph shows the price with respect to one of the factors, the area of the rental unit. The blue dots indicate the estimated values and the black dots indicate the predicted values.

In [132]:
```
value = pd.Series(value)
plt.scatter(value.head()-weight,  X_test['sqfeet'].head(),color = 'black')
print(value.head()-weight)

plt.scatter(X_test['price'].head(), X_test['sqfeet'].head(), color = 'blue')
plt.title('Neural Networks')
plt.xlabel('Rental Price')
plt.ylabel('Rental Unit Area')
plt.show()
```

```
0    1625.435433
1    2545.902713
2     668.379588
3    1220.505145
4     197.616248
dtype: float64
```