The Dataset

1) Please download the 'housing.csv' dataset from moodle.

2) We will use location data (latitude and longitude).

3) We will cluster the houses by location.

In [ ]:

```
import pandas as pd
home_data = pd.read_csv('housing.csv', usecols = ['longitude', 'latitude', 'median_house_
home_data.head()
```

Out[13]:

	longitude	latitude	median_house_value
0	-122.23	37.88	452600.0
1	-122.22	37.86	358500.0
2	-122.24	37.85	352100.0
3	-122.25	37.85	341300.0
4	-122.25	37.85	342200.0

Visualize the Data

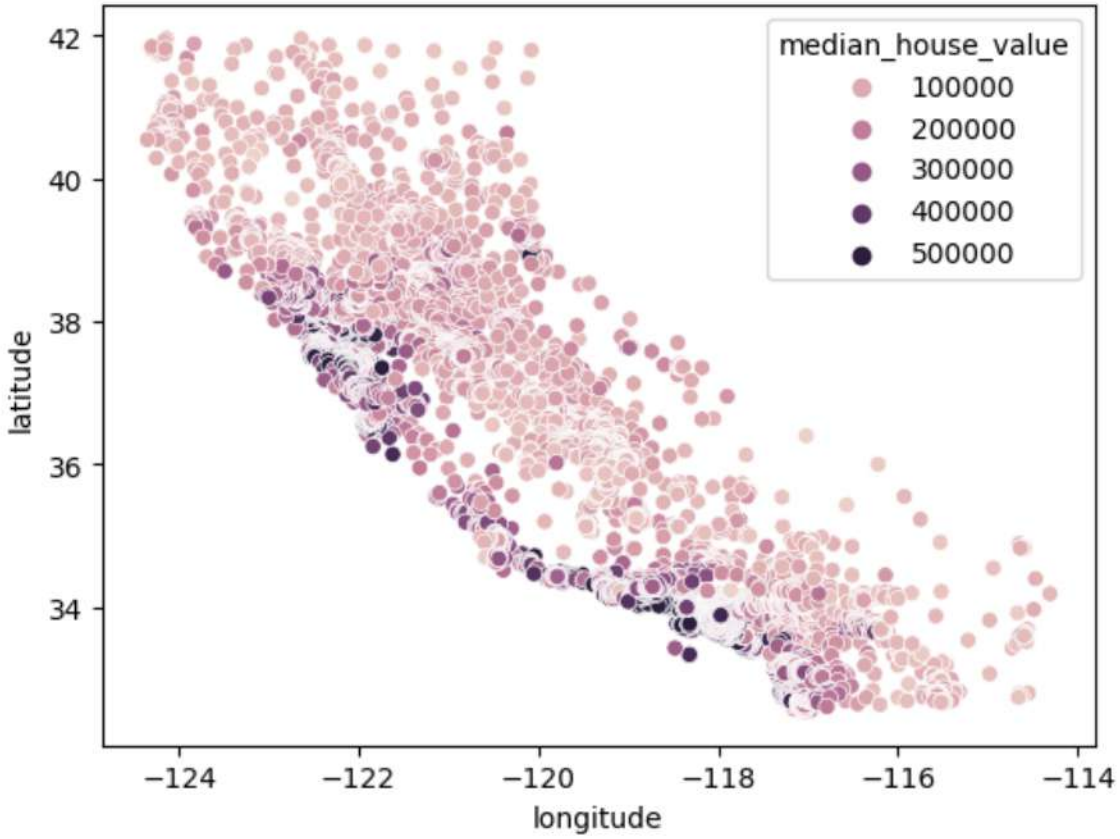
We look at the location data with a heatmap based on the median price in a block.

In [ ]:

```
import seaborn as sns
sns.scatterplot(data = home_data, x = 'longitude', y = 'latitude', hue = 'median_house_va
```

Out[14]:

```
<Axes: xlabel='longitude', ylabel='latitude'>
```



Normalizing the Data

1) When working with distance-based algorithms, like k-Means Clustering, we must normalize the data.

2) We first set up training and test splits using train_test_split from sklearn.

3) 33% of the data will be used for testing and 67% will be used for training.

4) random_state parameter is set to 0 to ensure that the same random split is generated each time the code is run.

In [ ]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(home_data[['latitude', 'longitude']],
```

We normalize the training and test data using the preprocessing.normalize() method from sklearn. This function scales the input data so that each sample (i.e., row) has a unit norm.

In [ ]:

```
from sklearn import preprocessing
X_train_norm = preprocessing.normalize(X_train)
X_test_norm = preprocessing.normalize(X_test)
```

Fitting and Evaluating the Model

1) We will arbitrarily choose a number of clusters (referred to as k) of 3.

2) We will create an instance of KMeans, define the number of clusters using the n_clusters attribute.

3) We can then fit the model to the normalized training data using the fit() method.

In [ ]:

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters = 3, random_state = 0, n_init='auto')
kmeans.fit(X_train_norm)
```

Out[17]:

```
KMeans(n_clusters=3, n_init='auto', random_state=0)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

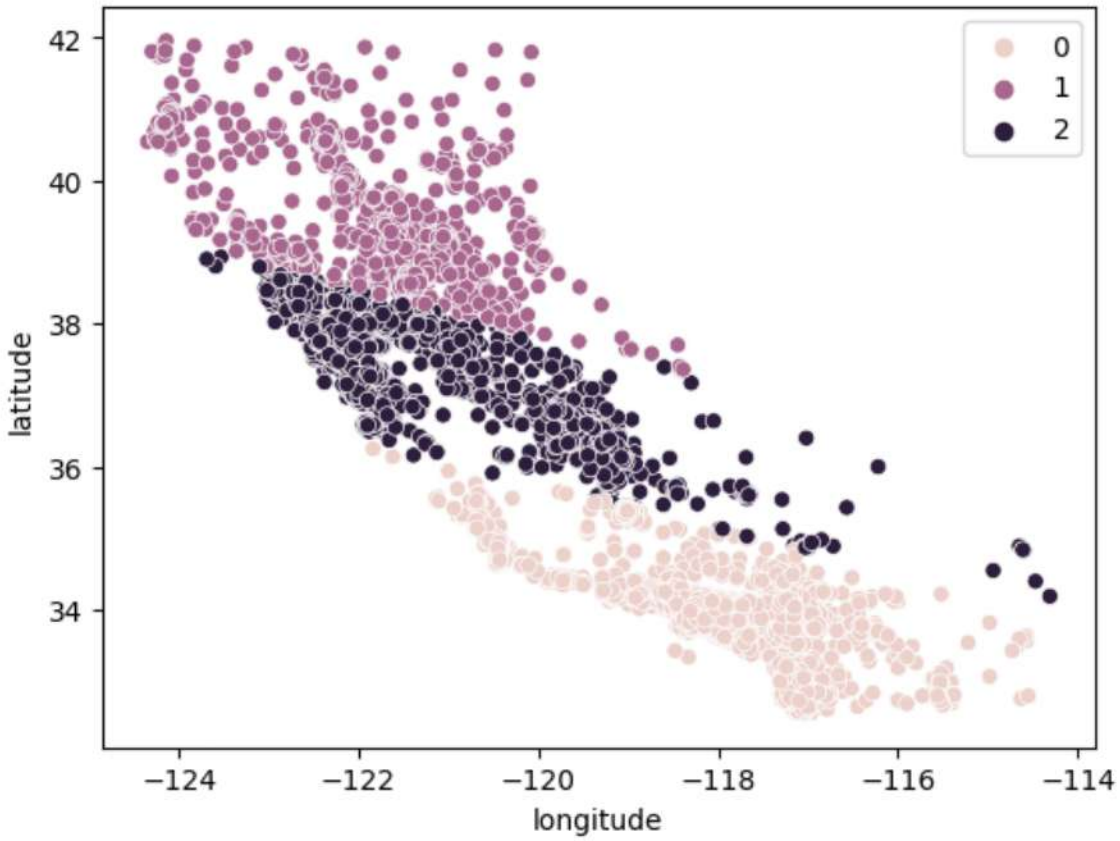
Below, we visualize the data we just fit. We see that the data are now clearly split into 3 distinct groups.

In [ ]:

```
sns.scatterplot(data = X_train, x = 'longitude', y = 'latitude', hue = kmeans.labels_)
```

Out[18]:

```
<Axes: xlabel='longitude', ylabel='latitude'>
```



Performance Evaluation

We can evaluate performance of the clustering algorithm using a Silhouette score which is a part of sklearn.metrics where a lower score represents a better fit.

In [ ]:

```
from sklearn.metrics import silhouette_score
silhouette_score(X_train_norm, kmeans.labels_, metric='euclidean')
```

Out[19]:

```
0.7499371920703546
```

Exercise : Choosing the best number of clusters

Try different values of k within the interval [2, 7] and identify the best k.