In [9]: %matplotlib notebook
import numpy as np
import matplotlib
import matplotlib.animation
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import pandas as pd

Dataset and visualization

The goal for this notebook is to show you some data, define terms of supervised learning, and give you confidence to go out and grab data from the wild world. Also, the first rule of machine learning: **LOOK AT YOUR DATA**.

I cannot emphasize this maxim enough: **LOOK AT YOUR DATA**

1. Housing prices are one of the most popular datasets on Kaggle—and its classical. We’re going to use the [Ames set](http://jse.amstat.org/v19n3/decock/DataDocumentation.txt). This is an updated dataset for the famous “Boston Housing Dataset”, which has been used for many years in stats classes.

2. I also want to point out the [UCI machine learning datasets](https://archive.ics.uci.edu/ml/index.php), which are amazing for ML datasets. Could be valuable for your projects!

In [2]: import pandas as pd

url="https://www.openintro.org/stat/data/ames.csv"
df=pd.read_csv(url)
data=df.values
df[1:10]

Out[2]:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>526350040</td>
<td>20</td>
<td>RH</td>
<td>80.0</td>
<td>11622</td>
<td>Pave</td>
<td>NaN</td>
<td>Reg</td>
<td>Lvl</td>
<td>0</td>
<td>NaN</td>
<td>NaN</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>526351010</td>
<td>20</td>
<td>RL</td>
<td>81.0</td>
<td>14267</td>
<td>Pave</td>
<td>NaN</td>
<td>IR1</td>
<td>Lvl</td>
<td>0</td>
<td>NaN</td>
<td>NaN</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>526353030</td>
<td>20</td>
<td>RL</td>
<td>93.0</td>
<td>11160</td>
<td>Pave</td>
<td>NaN</td>
<td>Reg</td>
<td>Lvl</td>
<td>0</td>
<td>NaN</td>
<td>NaN</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>527105010</td>
<td>60</td>
<td>RL</td>
<td>74.0</td>
<td>13830</td>
<td>Pave</td>
<td>NaN</td>
<td>IR1</td>
<td>Lvl</td>
<td>0</td>
<td>NaN</td>
<td>NaN</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>527105030</td>
<td>60</td>
<td>RL</td>
<td>78.0</td>
<td>9978</td>
<td>Pave</td>
<td>NaN</td>
<td>IR1</td>
<td>Lvl</td>
<td>0</td>
<td>NaN</td>
<td>NaN</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>527127150</td>
<td>120</td>
<td>RL</td>
<td>41.0</td>
<td>4820</td>
<td>Pave</td>
<td>NaN</td>
<td>Reg</td>
<td>Lvl</td>
<td>0</td>
<td>NaN</td>
<td>NaN</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>527145080</td>
<td>120</td>
<td>RL</td>
<td>43.0</td>
<td>5005</td>
<td>Pave</td>
<td>NaN</td>
<td>IR1</td>
<td>HLS</td>
<td>0</td>
<td>NaN</td>
<td>NaN</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>527146030</td>
<td>120</td>
<td>RL</td>
<td>39.0</td>
<td>5389</td>
<td>Pave</td>
<td>NaN</td>
<td>IR1</td>
<td>Lvl</td>
<td>0</td>
<td>NaN</td>
<td>NaN</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>527162130</td>
<td>60</td>
<td>RL</td>
<td>60.0</td>
<td>7500</td>
<td>Pave</td>
<td>NaN</td>
<td>Reg</td>
<td>Lvl</td>
<td>0</td>
<td>NaN</td>
<td>NaN</td>
<td></td>
</tr>
</tbody>
</table>

9 rows × 82 columns
```python
n = 50
# This says "let's look at two story houses sold in 2010"
new_house = (df['MS.SubClass'] == 60) & (df['Yr.Sold'] == 2010)
price = df['SalePrice'][new_house]
lot = df['Lot.Frontage'][new_house]
bedroom = df['Bedroom.AbvGr'][new_house]
fig = plt.figure()
ax = fig.add_subplot(111)
ax.scatter(lot, price)
ax.scatter(bedroom, price)
plt.show()
```

```python
fig = plt.figure()
plt.scatter(bedroom, price)
plt.xlabel("Bedrooms")
plt.ylabel("Price")
plt.show()
```
There does intuitively seem to be a correlation, but we could do a bit better if we added in another feature.

A much simpler, but still real linear relationship!

Here is another dataset, in which we do. Let’s look at miles per gallon in the city to Highway Miles per Gallon of old cars. The hypothesis is that there should be a linear relationship, because some cars are just more efficient than others.
Now that looks like a line! Let's recall how to fit it!