Load the data

In [2]:
```
import pandas as pd
df = pd.read_csv('california_housing.csv')
df
```

Out[2]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	L
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	
...	...	...	...	...	...	...	...	
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	

20640 rows × 9 columns

data preparation

seperate the x and y

In [3]:
```
x=df.drop('MedHouseVal',axis=1)# features
y=df['MedHouseVal']     # target values
print(x)
print(y)
```

```
       MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  \
0      8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88
1      8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86
2      7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85
3      5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85
4      3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85
...       ...       ...       ...        ...         ...       ...       ...
20635  1.5603      25.0  5.045455   1.133333       845.0  2.560606     39.48
20636  2.5568      18.0  6.114035   1.315789       356.0  3.122807     39.49
20637  1.7000      17.0  5.205543   1.120092      1007.0  2.325635     39.43
20638  1.8672      18.0  5.329513   1.171920       741.0  2.123209     39.43
20639  2.3886      16.0  5.254717   1.162264      1387.0  2.616981     39.37

       Longitude
0        -122.23
1        -122.22
2        -122.24
3        -122.25
4        -122.25
...          ...
20635    -121.09
20636    -121.21
20637    -121.22
20638    -121.32
20639    -121.24

[20640 rows x 8 columns]
0        4.526
1        3.585
2        3.521
3        3.413
4        3.422
         ...
20635    0.781
20636    0.771
20637    0.923
20638    0.847
20639    0.894
Name: MedHouseVal, Length: 20640, dtype: float64
```

data spliting

In [4]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.2,random_state=10
```

In [5]:
```
X_train # 80 % traning data .
```

Out[5]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	L
3278	3.3929	13.0	8.580645	1.790323	214.0	3.451613	39.14	
16630	3.2226	11.0	4.927273	1.062121	1814.0	2.748485	35.32	
18748	3.3500	18.0	5.710638	1.018440	1731.0	2.455319	40.49	
14961	5.2741	7.0	6.855372	0.979339	862.0	3.561983	32.76	
1740	2.3382	19.0	4.059891	1.052632	1438.0	2.609800	37.97	
...	...	...	...	...	...	...	...	
16304	10.0088	15.0	7.738854	1.003185	1016.0	3.235669	38.01	
79	2.0114	38.0	4.412903	1.135484	344.0	2.219355	37.80	
12119	5.6409	3.0	7.837746	1.083262	8437.0	3.602477	33.97	
14147	2.3812	35.0	6.289474	1.109649	753.0	3.302632	32.74	
5640	4.6250	48.0	5.152632	1.015789	1098.0	2.889474	33.75	

16512 rows × 8 columns

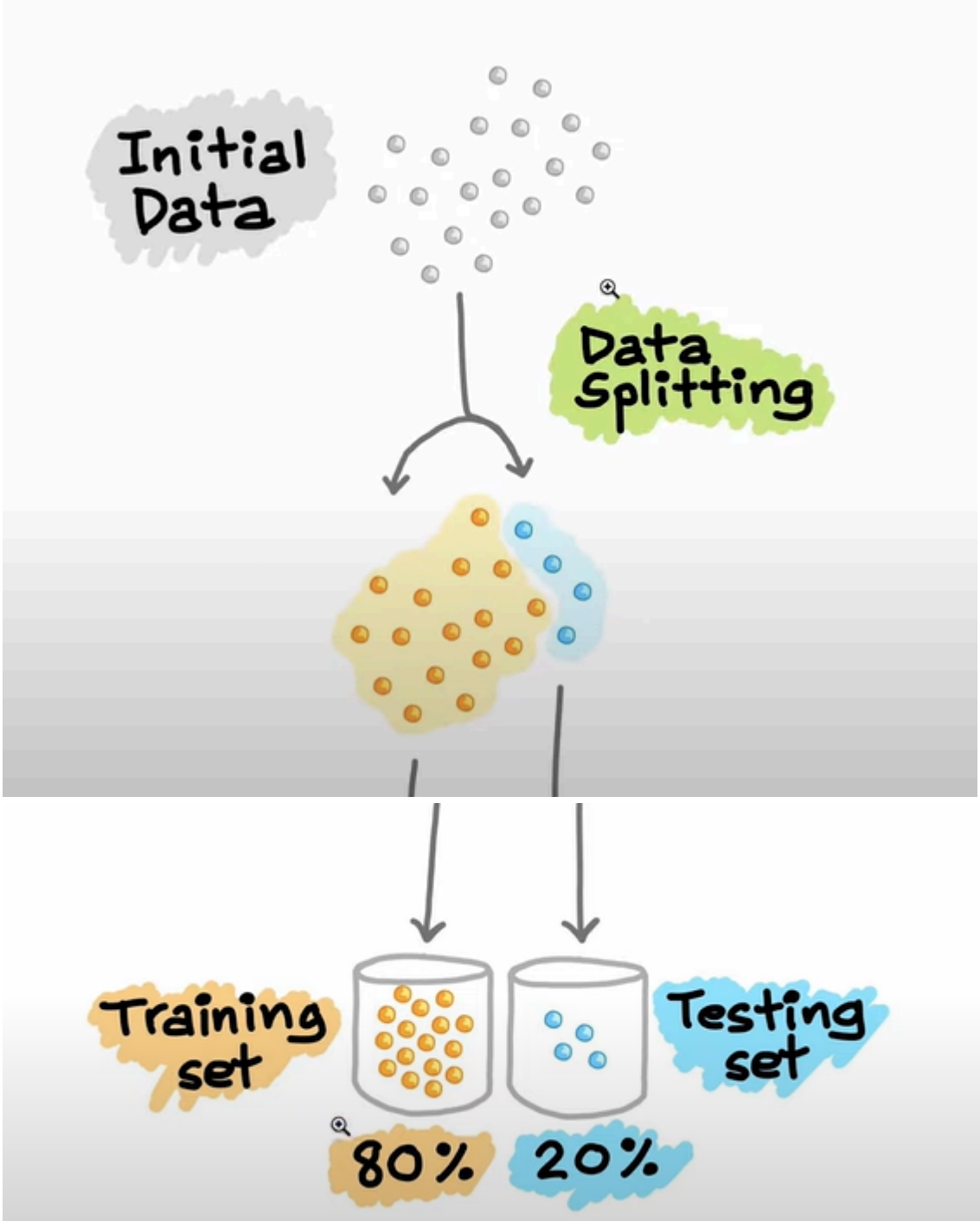
In [6]:
```
X_test # 20 % of the testing data
```

Out[6]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	L
8151	3.7031	36.0	6.276836	1.039548	444.0	2.508475	33.81	
53	1.2475	52.0	4.075000	1.140000	1162.0	2.905000	37.82	
3039	4.8266	13.0	6.746647	1.062593	2170.0	3.233979	35.37	
9484	2.8833	19.0	6.750000	1.348684	424.0	2.789474	39.31	
9307	2.8903	31.0	4.477459	1.073087	2962.0	2.023224	37.98	
...	...	...	...	...	...	...	...	
16733	2.1154	8.0	4.288660	1.247423	936.0	9.649485	35.47	
5264	11.2866	14.0	7.271898	1.041971	2926.0	2.669708	34.09	
12374	3.3799	6.0	10.860423	2.036020	4176.0	1.880234	33.78	
19662	1.7227	52.0	4.954023	1.011494	922.0	2.649425	37.50	
11942	4.1528	35.0	4.997167	0.920680	1094.0	3.099150	33.94	

4128 rows × 8 columns

In [7]:
```
from IPython.display import display
from PIL import Image
img1=Image.open(r"C:\Users\bhanuprasad\OneDrive\Pictures\Screenshots\Screenshot
img2=Image.open(r"C:\Users\bhanuprasad\OneDrive\Pictures\Screenshots\Screenshot
display(img1,img2)
#display(img2)
```



- **typically we use the training data set using to built a model**

Model Building

Linear regression

- **Training the model**

In [8]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train, Y_train)
```

Out[8]:
```
LinearRegression()
```

- **Applying the model to make preditions**

In [9]:
```
y_lr_train_pred=lr.predict(X_train)
y_lr_test_pred=lr.predict(X_test)
```

In [10]:
```
print(y_lr_train_pred)
```

```
[1.72441662 2.34061463 0.80938884 ... 1.95650644 1.54009365 2.79819134]
```

In [11]:
```
print(y_lr_test_pred)
```

```
[2.07653781 1.57976836 2.08731697 ... 1.04278469 1.12583146 1.9625772 ]
```

Evaluate the performance

In [12]:
```
Y_train
# actual traing data
```

Out[12]:
```
3278     1.083
16630    1.832
18748    1.184
14961    2.494
1740     1.207
         ...
16304    2.420
79       1.313
12119    1.977
14147    1.351
5640     2.734
Name: MedHouseVal, Length: 16512, dtype: float64
```

In [13]:
```
y_lr_train_pred
# predicted traing data
```

Out[13]:
```
array([1.72441662, 2.34061463, 0.80938884, ..., 1.95650644, 1.54009365,
       2.79819134])
```

In [14]:
```
from sklearn.metrics import mean_squared_error,r2_score
# training sets
lr_train_mse=mean_squared_error(Y_train,y_lr_train_pred)
lr_train_r2=r2_score(Y_train,y_lr_train_pred )
#testing sets
lr_test_mse=mean_squared_error(Y_test,y_lr_test_pred)
lr_test_r2=r2_score(Y_test,y_lr_test_pred )
```

In [15]:
```
lr_train_mse
```

Out[15]:
```
0.5282593680413493
```

In [16]:
```
lr_train_r2
```

Out[16]:
```
0.6020775708086616
```

In [17]:
```
lr_test_mse
```

Out[17]:
```
0.5088933351158945
```

In [18]:
```
lr_test_r2
```

Out[18]:
```
0.622313810729529
```

In [19]:
```
print('LR MSE (Train) :',lr_train_mse)
print('LR R2 (Train)  :',lr_train_r2)
print('LR MSE (test)  :',lr_test_mse)
print('LR R2 (test)   :',lr_test_r2)
```

```
LR MSE (Train) : 0.5282593680413493
LR R2 (Train)  : 0.6020775708086616
LR MSE (test)  : 0.5088933351158945
LR R2 (test)   : 0.622313810729529
```

In [20]:
```
import pandas as pd
lr_results=pd.DataFrame(['Linear regression',lr_train_mse,lr_train_r2,lr_test_ms
lr_results
```

Out[20]:

	0	1	2	3	4
0	Linear regression	0.528259	0.602078	0.508893	0.622314

In [21]:
```
lr_results.columns=['method','Training MSE','Training R2','Test MSE','Test R2' ]
lr_results
```

Out[21]:

	method	Training MSE	Training R2	Test MSE	Test R2
0	Linear regression	0.528259	0.602078	0.508893	0.622314

In [ ]:
```

```

In [ ]:
```

```

Random Forest

traning the model

In [22]:
```
from sklearn.ensemble import RandomForestRegressor
rf=RandomForestRegressor(max_depth=2,random_state=100)
rf.fit(X_train,Y_train)
```

Out[22]:
```
RandomForestRegressor
RandomForestRegressor(max_depth=2, random_state=100)
```

Applying the model to make a predition

In [23]:
```
y_rf_train_pred=rf.predict(X_train)
y_rf_test_pred=rf.predict(X_test)
```

Evaluate model preformance

In [24]:
```
from sklearn.metrics import mean_squared_error,r2_score
# training sets
rf_train_mse=mean_squared_error(Y_train,y_rf_train_pred)
rf_train_r2=r2_score(Y_train,y_rf_train_pred )
#testing sets
rf_test_mse=mean_squared_error(Y_test,y_rf_test_pred)
rf_test_r2=r2_score(Y_test,y_rf_test_pred )
```

In [25]:
```
import pandas as pd
rf_results=pd.DataFrame(['Random Forest',rf_train_mse,rf_train_r2,rf_test_mse,rf
rf_results.columns=['method','Training MSE','Training R2','Test MSE','Test R2' ]
rf_results
```

Out[25]:

	method	Training MSE	Training R2	Test MSE	Test R2
0	Random Forest	0.715754	0.460844	0.706795	0.475437

Model comparison

In [26]:
```
import pandas as pd
df_models=pd.concat([lr_results,rf_results],axis=0)
df_models
```

Out[26]:

	method	Training MSE	Training R2	Test MSE	Test R2
0	Linear regression	0.528259	0.602078	0.508893	0.622314
0	Random Forest	0.715754	0.460844	0.706795	0.475437

In [27]:
```
df_models.reset_index(drop=True)
```

Out[27]:

	method	Training MSE	Training R2	Test MSE	Test R2
0	Linear regression	0.528259	0.602078	0.508893	0.622314
1	Random Forest	0.715754	0.460844	0.706795	0.475437

Data visulaization of predition results

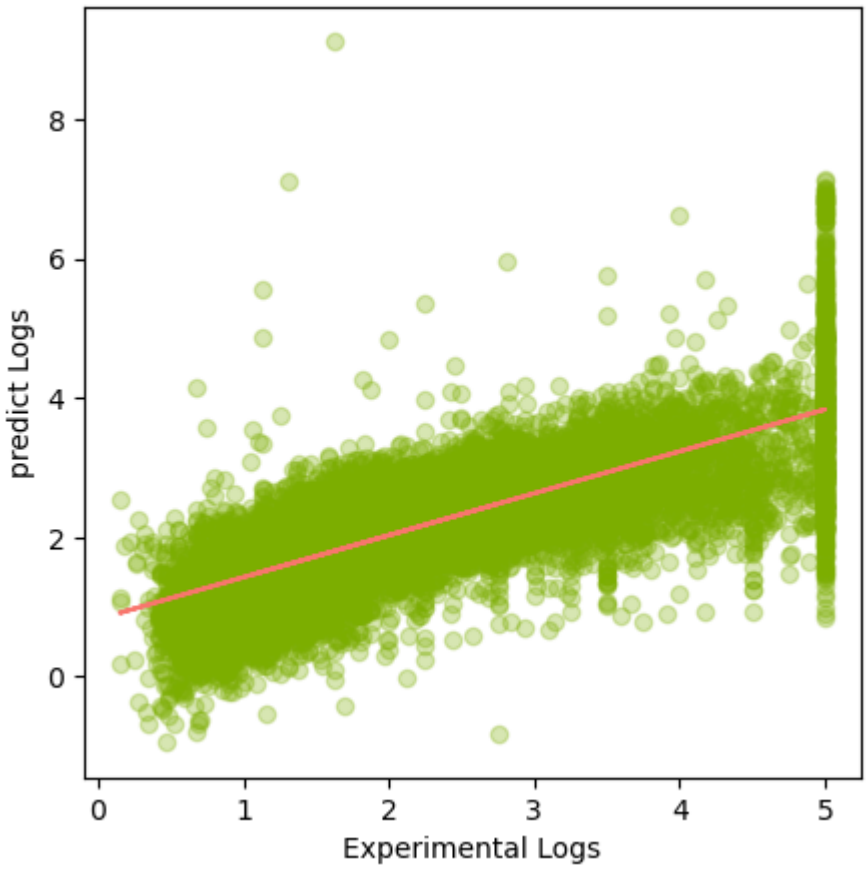
In [29]:
```
import matplotlib.pyplot as plt
import numpy as np

plt.figure(figsize=(5,5))
plt.scatter(x=Y_train,y=y_lr_train_pred,c='#7CAE00',alpha=0.3)

z=np.polyfit(Y_train,y_lr_train_pred,1)
p=np.poly1d(z)
```

```
plt.plot(Y_train,p(Y_train),'#F8766D')
plt.ylabel('predict Logs')
plt.xlabel('Experimental Logs')
```

Out[29]: Text(0.5, 0, 'Experimental Logs')



In [ ]: