Problem statement

predicting the house price in USA.To create a model to help him estimate of what the house would sell for.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv("Vande Bharat")
```

To display top 10 rows

In [3]:

```
df.head(10)
```

Out[3]:

	Sr. No.	Train Name	Train Number	Originating City	Originating Station	Terminal City	Terminal Station	Operator	No. of Cars	Frequency	Distance	T
0	1	New Delhi - Varanasi Vande Bharat Express	22435/22436	Delhi	New Delhi	Varanasi	Varanasi Junction	NR	16	Except Thursdays	759 km (472 mi)	
1	2	New Delhi - Shri Mata Vaishno Devi Katra Vande...	22439/22440	Delhi	New Delhi	Katra	Shri Mata Vaishno Devi Katra	NR	16	Except Tuesdays	655 km (407 mi)	
2	3	Mumbai Central - Gandhinagar Capital Vande Bha...	20901/20902	Mumbai	Mumbai Central	Gandhinagar	Gandhinagar Capital	WR	16	Except Wednesdays	522 km (324 mi)	
3	4	New Delhi - Amb Andaura Vande Bharat Express	22447/22448	Delhi	New Delhi	Andaura	Amb Andaura	NR	16	Except Fridays	412 km (256 mi)	
4	5	MGR Chennai Central - Mysuru Vande Bharat Express	20607/20608	Chennai	Chennai Central	Mysuru	Mysore Junction	SR	16	Except Wednesdays	496 km (308 mi)	
5	6	Bilaspur - Nagpur Vande Bharat Express	20825/20826	Bilaspur, Chhattisgarh	Bilaspur Junction	Nagpur	Nagpur Junction	SECR	8	Except Saturdays	412 km (256 mi)	
6	7	Howrah - New Jalpaiguri Vande Bharat Express	22301/22302	Kolkata	Howrah Junction	Siliguri	New Jalpaiguri Junction	ER	16	Except Wednesdays	565 km (351 mi)	
7	8	Visakhapatnam - Secunderabad Vande Bharat Express	20833/20834	Visakhapatnam	Visakhapatnam Junction	Hyderabad	Secunderabad Junction	ECoR	16	Except Sundays	698 km (434 mi)	
8	9	Mumbai CSMT - Solapur Vande Bharat Express	22225/22226	Mumbai	Chhatrapati Shivaji Terminus	Solapur	Solapur	CR	16	Except Wednesdays (22225) , Except Thursdays (...	452 km (281 mi)	

	Sr. No.	Train Name	Train Number	Originating City	Originating Station	Terminal City	Terminal Station	Operator	No. of Cars	Frequency	Distance	T
9	10	Mumbai CSMT - Sainagar Shirdi Vande Bharat Exp...	22223/22224	Mumbai	Chhatrapati Shivaji Terminus	Shirdi	Sainagar Shirdi	CR	16	Except Tuesdays	339 km (211 mi)	

Data Cleaning And Pre-Processing

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26 entries, 0 to 25
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   Sr. No.              26 non-null     int64 
 1   Train Name           26 non-null     object
 2   Train Number         26 non-null     object
 3   Originating City     26 non-null     object
 4   Originating Station  26 non-null     object
 5   Terminal City        26 non-null     object
 6   Terminal Station     26 non-null     object
 7   Operator             26 non-null     object
 8   No. of Cars          26 non-null     int64 
 9   Frequency            26 non-null     object
 10  Distance             26 non-null     object
 11  Travel Time          26 non-null     object
 12  Speed                26 non-null     object
 13  Average Speed        26 non-null     object
 14  Inauguration         26 non-null     object
 15  Average occupancy    26 non-null     object
dtypes: int64(2), object(14)
memory usage: 3.4+ KB
```

In [5]:
```python
# Display the statistical summary
df.describe()
```
Out[5]:

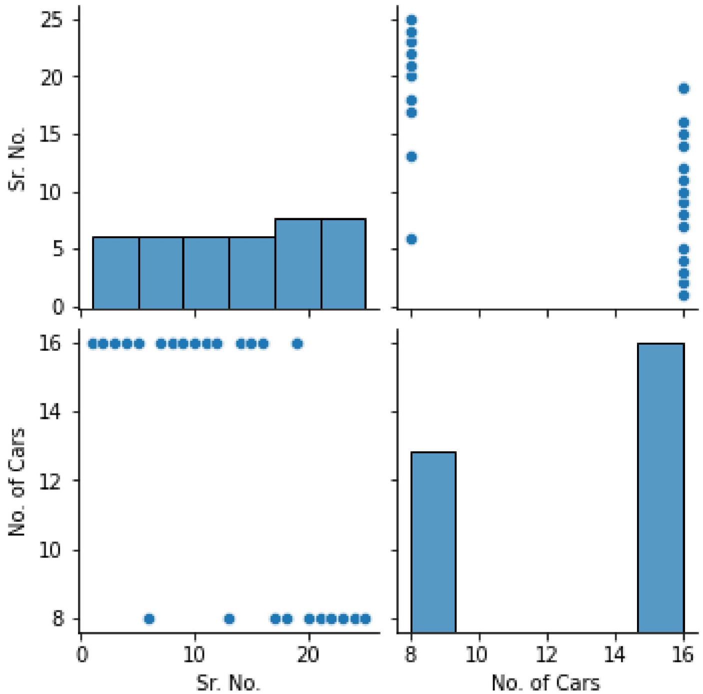
	Sr. No.	No. of Cars
count	26.000000	26.000000
mean	13.230769	12.923077
std	7.306478	3.969112
min	1.000000	8.000000
25%	7.250000	8.000000
50%	13.500000	16.000000
75%	19.000000	16.000000
max	25.000000	16.000000

In [6]:
```python
# To display the col headings
df.columns
```
Out[6]:
```
Index(['Sr. No.', 'Train Name', 'Train Number', 'Originating City',
       'Originating Station', 'Terminal City', 'Terminal Station', 'Operator',
       'No. of Cars', 'Frequency', 'Distance', 'Travel Time', 'Speed',
       'Average Speed', 'Inauguration', 'Average occupancy'],
      dtype='object')
```
In [7]:
```python
cols=df.dropna(axis=1)
```
In [8]:
```python
cols.columns
```
Out[8]:
```
Index(['Sr. No.', 'Train Name', 'Train Number', 'Originating City',
       'Originating Station', 'Terminal City', 'Terminal Station', 'Operator',
       'No. of Cars', 'Frequency', 'Distance', 'Travel Time', 'Speed',
       'Average Speed', 'Inauguration', 'Average occupancy'],
      dtype='object')
```

EDA and Visualization

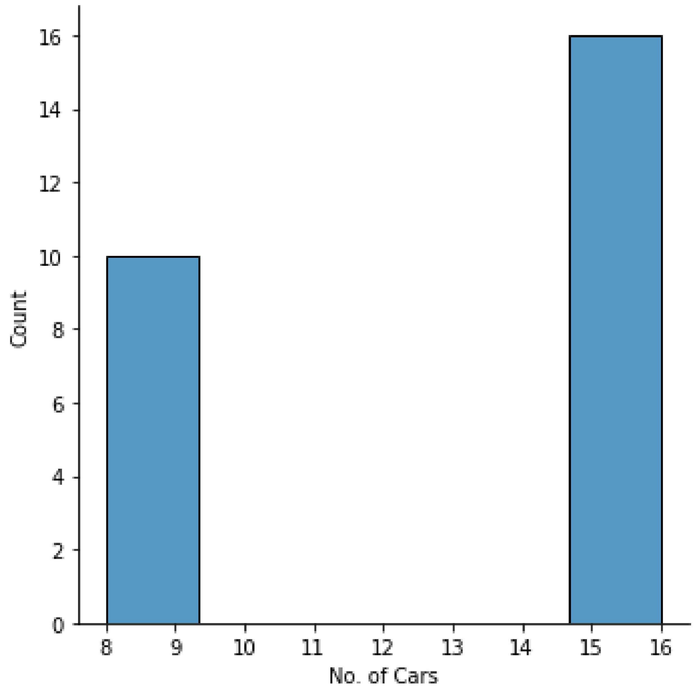
In [9]:
```
sns.pairplot(cols)
```

Out[9]: <seaborn.axisgrid.PairGrid at 0x1e963a001f0>



In [11]:
```
sns.displot(df['No. of Cars'])
```

Out[11]: <seaborn.axisgrid.FacetGrid at 0x1e9629f4790>



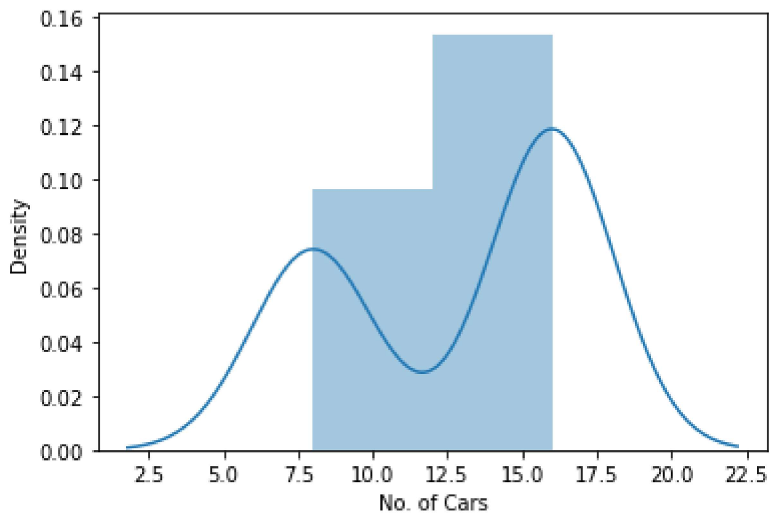
In [12]:

```python
# We use displot in older version we get distplot use displot
sns.distplot(df['No. of Cars'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a dep
recated function and will be removed in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[12]:

```
<AxesSubplot:xlabel='No. of Cars', ylabel='Density'>
```



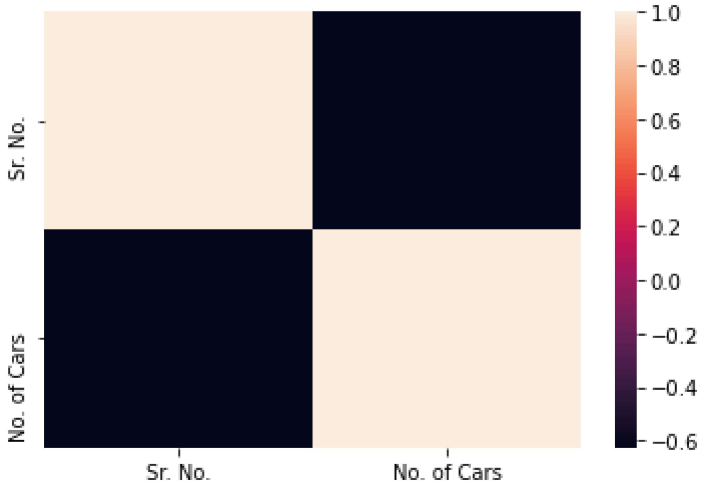
In [13]:

```
df1=cols[['Sr. No.', 'No. of Cars',]]
df1
```

Out[13]:

	Sr. No.	No. of Cars
0	1	16
1	2	16
2	3	16
3	4	16
4	5	16
5	6	8
6	7	16
7	8	16
8	9	16
9	10	16
10	11	16
11	12	16
12	13	8
13	14	16
14	15	16
15	16	16
16	17	8
17	18	8
18	19	16
19	19	16
20	20	8
21	21	8
22	22	8
23	23	8
24	24	8
25	25	8

In [14]:
```
sns.heatmap(df1.corr())
```
Out[14]: <AxesSubplot:>



To train the model - MODEL BUILD

Going to train linear regression model;We split our data into 2 variables x and y where x is independent var(input) and y is dependent on x(output), we could ignore address col as it is not required for our model

In [15]:
```
x=df1[['Sr. No.', 'No. of Cars',]]
y=df1[['No. of Cars']]
```

To split the dataset into test data

In [16]:
```
# importing lib for splitting test data
from sklearn.model_selection import train_test_split
```

In [17]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [18]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[18]:
```
LinearRegression()
```

In [19]:
```
print(lr.intercept_)
```

```
[3.55271368e-15]
```

In [20]:
```
print(lr.score(x_test,y_test))
```

```
1.0
```

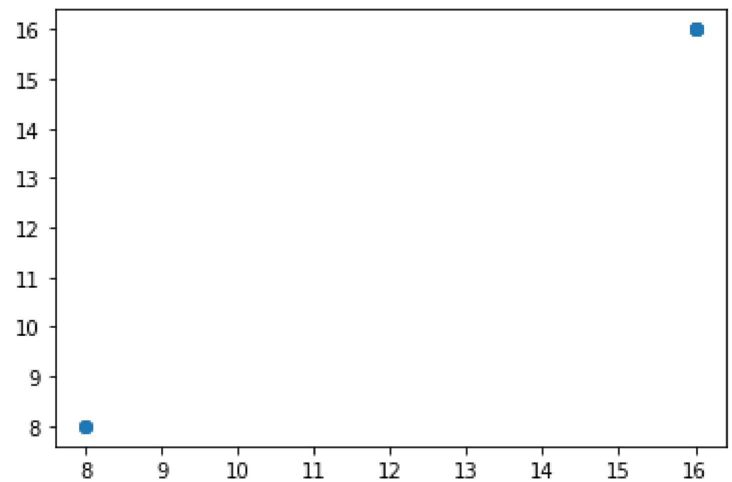
In [21]:
```
coeff=pd.DataFrame(lr.coef_)
coeff
```

Out[21]:

	0	1
0	-2.607996e-16	1.0

In [22]:
```
pred = lr.predict(x_test)
plt.scatter(y_test,pred)
```

Out[22]: <matplotlib.collections.PathCollection at 0x1e965f3c100>



In [23]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [24]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[24]: Ridge(alpha=10)

In [25]:
```
rr.score(x_test,y_test)
```

Out[25]: 0.9995007346948294

In [26]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[26]: Lasso(alpha=10)

In [27]:
```
la.score(x_test,y_test)
```

Out[27]: 0.567380671276775

ELASTIC NET

In [28]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[28]:
```
ElasticNet()
```

In [29]:
```
print(en.coef_)
```

```
[-0.00738509  0.93085556]
```

In [30]:
```
print(en.intercept_)
```

```
[0.99007318]
```

In [31]:
```
prediction=en.predict(x_test)
prediction
```

Out[31]:
```
array([15.84683669, 15.85422179, 15.86899197,  8.28183071, 15.78037086,
       15.81729632,  8.25229034,  8.26706053])
```

In [32]:
```
print(en.score(x_test,y_test))
```

```
0.997017572713016
```

EVALUATION METRICS

In [33]:
```
from sklearn import metrics
```

In [34]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 0.20418299315749788
```

In [35]:

```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 0.04473640930475922
```

In [36]:

```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root Mean Squared Error: 0.21150983264321124
```

MODEL SAVING

In [37]:

```
import pickle
```

In [38]:

```
filename='prediction5'
pickle.dump(lr,open(filename,'wb'))
```

In [ ]:

```

```