TECHNOHACKS : Data Science

Use a dataset that includes information about housing prices and featurs like squre footage,number of bedrooms etc. to train a model that can predict the price of a new house.

Author: Sujata Gaikwad

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
data=pd.read_csv(r"C:\Users\HP\Downloads\kc_house_data.csv.zip")
data
```

Out[2]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_b
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180	0	1
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170	400	1
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770	0	1
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050	910	1
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
21608	263000018	20140521T000000	360000.0	3	2.50	1530	1131	3.0	0	0	...	8	1530	0	2
21609	6600060120	20150223T000000	400000.0	4	2.50	2310	5813	2.0	0	0	...	8	2310	0	2
21610	1523300141	20140623T000000	402101.0	2	0.75	1020	1350	2.0	0	0	...	7	1020	0	2
21611	291310100	20150116T000000	400000.0	3	2.50	1600	2388	2.0	0	0	...	8	1600	0	2
21612	1523300157	20141015T000000	325000.0	2	0.75	1020	1076	2.0	0	0	...	7	1020	0	2

21613 rows × 21 columns

In [3]:
```
data.head()
```

Out[3]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_built
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180	0	1955
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170	400	1951
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770	0	1933
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050	910	1965
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680	0	1987

5 rows × 21 columns

In [4]:
```
data.shape
```

Out[4]:
```
(21613, 21)
```

In [5]:
```
data.dtypes
```

Out[5]:
```
id                int64
date             object
price           float64
bedrooms          int64
bathrooms       float64
sqft_living       int64
sqft_lot          int64
floors          float64
waterfront        int64
view              int64
condition         int64
grade             int64
sqft_above        int64
sqft_basement     int64
yr_built          int64
yr_renovated      int64
zipcode           int64
lat             float64
long            float64
sqft_living15     int64
sqft_lot15        int64
dtype: object
```

In [6]:
```
data.describe()
```

Out[6]:

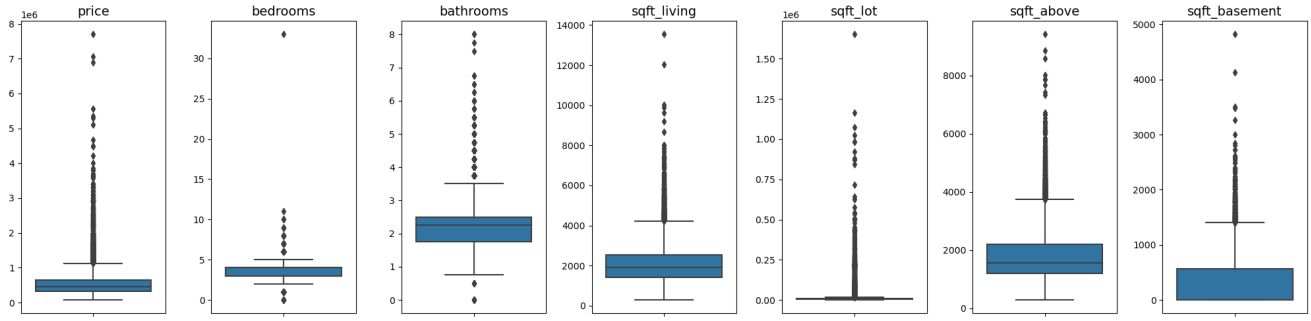
	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	g
count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000	2.161300e+04	21613.000000	21613.000000	21613.000000	21613.000000	21613.00
mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736	1.510697e+04	1.494309	0.007542	0.234303	3.409430	7.65
std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897	4.142051e+04	0.539989	0.086517	0.766318	0.650743	1.17
min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000	5.200000e+02	1.000000	0.000000	0.000000	1.000000	1.00
25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	1.000000	0.000000	0.000000	3.000000	7.00
50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	0.000000	0.000000	3.000000	7.00
75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04	2.000000	0.000000	0.000000	4.000000	8.00
max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.000000	4.000000	5.000000	13.00

In [7]:
```
data.isnull().sum()
```

Out[7]:
```
id               0
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
grade            0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
zipcode          0
lat              0
long             0
sqft_living15    0
sqft_lot15       0
dtype: int64
```

Exploratory Data Analysis

In [8]:
```
import seaborn as sns
import matplotlib.pyplot as plt
cols=["price","bedrooms","bathrooms","sqft_living","sqft_lot","sqft_above","sqft_basement"]
fig,axes=plt.subplots(1,len(cols),figsize=(20,5))
for i, col in enumerate(cols):
    sns.boxplot(y=col,data=data,ax=axes[i])
    axes[i].set_title(col,fontsize=14)
    axes[i].set_ylabel("")
plt.tight_layout()
plt.show()
```

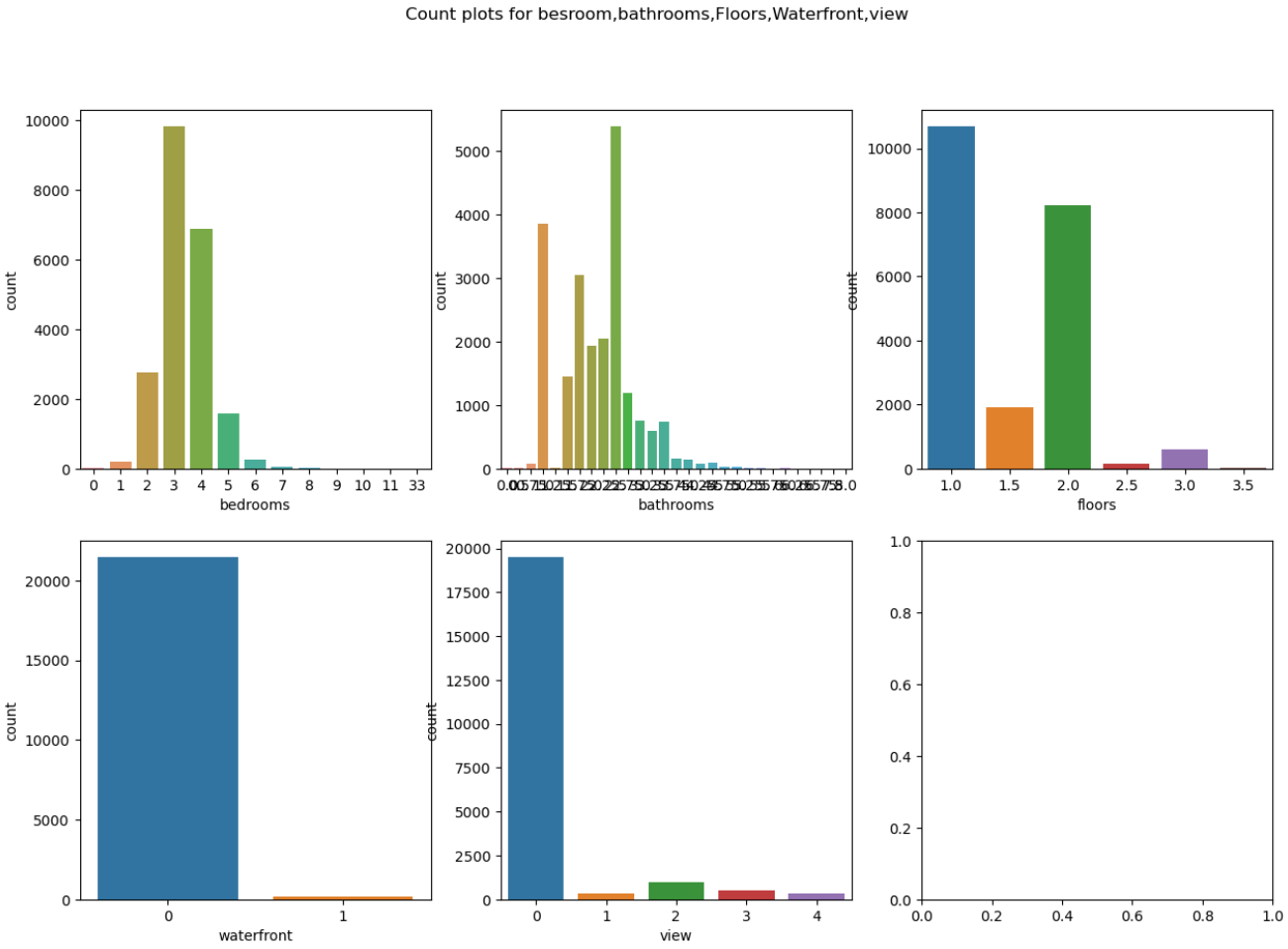


In [9]:
```
## Remove outliers for price column
ul=5000000
data=data[data["price"]<=ul]
## Remove outlierd for bedroom column
ul1=8
data1=data[data["bedrooms"]<=ul1]
## Remove outlierd for bathrooms column
ul2=6
data2=data[(data["bathrooms"]<ul2)&data["bathrooms"]>=1]
## Remove outlierd for sqft_living column
ul3=8500
data3=data[data["sqft_living"]<=ul3]
##Remove outlierd for sqft_lot column
ul4=60000
data4=data[data["sqft_lot"]<=ul4]
##Remove outlierd for sqft_above column
ul5=7000
data5=data[data["sqft_above"]<=ul5]
##Remove outlierd for sqft_basement column
ul5=3000
data5=data[data["sqft_lot"]<=ul5]
data.describe().transpose()
```

Out[9]:

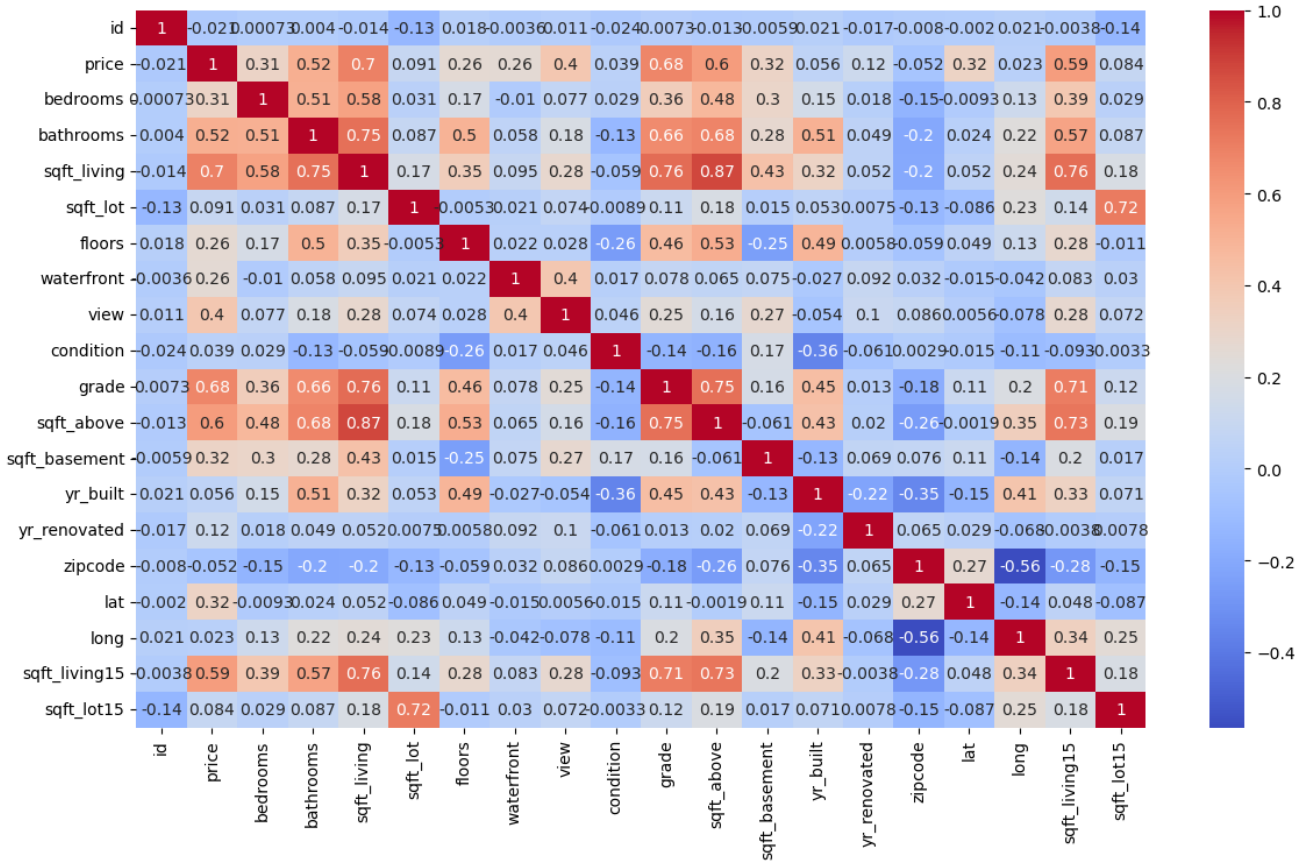
	count	mean	std	min	25%	50%	75%	max
id	21606.0	4.579658e+09	2.876254e+09	1.000102e+06	2.123049e+09	3.904926e+09	7.308600e+09	9.900000e+09
price	21606.0	5.382739e+05	3.526472e+05	7.500000e+04	3.215000e+05	4.500000e+05	6.450000e+05	4.668000e+06
bedrooms	21606.0	3.370175e+00	9.294319e-01	0.000000e+00	3.000000e+00	3.000000e+00	4.000000e+00	3.300000e+01
bathrooms	21606.0	2.113487e+00	7.667170e-01	0.000000e+00	1.750000e+00	2.250000e+00	2.500000e+00	8.000000e+00
sqft_living	21606.0	2.077585e+03	9.091442e+02	2.900000e+02	1.422750e+03	1.910000e+03	2.550000e+03	1.354000e+04
sqft_lot	21606.0	1.510142e+04	4.142587e+04	5.200000e+02	5.040000e+03	7.616500e+03	1.067850e+04	1.651359e+06
floors	21606.0	1.494122e+00	5.399672e-01	1.000000e+00	1.000000e+00	1.500000e+00	2.000000e+00	3.500000e+00
waterfront	21606.0	7.405350e-03	8.573711e-02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00
view	21606.0	2.334074e-01	7.643990e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	4.000000e+00
condition	21606.0	3.409470e+00	6.507938e-01	1.000000e+00	3.000000e+00	3.000000e+00	4.000000e+00	5.000000e+00
grade	21606.0	7.655374e+00	1.172624e+00	1.000000e+00	7.000000e+00	7.000000e+00	8.000000e+00	1.300000e+01
sqft_above	21606.0	1.786701e+03	8.225467e+02	2.900000e+02	1.190000e+03	1.560000e+03	2.210000e+03	9.410000e+03
sqft_basement	21606.0	2.908833e+02	4.410386e+02	0.000000e+00	0.000000e+00	0.000000e+00	5.600000e+02	4.820000e+03
yr_built	21606.0	1.971003e+03	2.937099e+01	1.900000e+03	1.951000e+03	1.975000e+03	1.997000e+03	2.015000e+03
yr_renovated	21606.0	8.424502e+01	4.013219e+02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	2.015000e+03
zipcode	21606.0	9.807795e+04	5.350590e+01	9.800100e+04	9.803300e+04	9.806500e+04	9.811800e+04	9.819900e+04
lat	21606.0	4.756003e+01	1.385794e-01	4.715590e+01	4.747082e+01	4.757180e+01	4.767800e+01	4.777760e+01
long	21606.0	-1.222139e+02	1.408490e-01	-1.225190e+02	-1.223280e+02	-1.222305e+02	-1.221250e+02	-1.213150e+02
sqft_living15	21606.0	1.985885e+03	6.844564e+02	3.990000e+02	1.490000e+03	1.840000e+03	2.360000e+03	6.210000e+03
sqft_lot15	21606.0	1.276452e+04	2.730722e+04	6.510000e+02	5.100000e+03	7.620000e+03	1.008000e+04	8.712000e+05

In [10]:
```
## Define the fig and suplots
fig,axes=plt.subplots(nrows=2,ncols=3,figsize=(15,10))

## Create count plots for each variable and add them to the subplots
sns.countplot(x="bedrooms",data=data,ax=axes[0,0])
sns.countplot(x="bathrooms",data=data,ax=axes[0,1])
sns.countplot(x="floors",data=data,ax=axes[0,2])
sns.countplot(x="waterfront",data=data,ax=axes[1,0])
sns.countplot(x="view",data=data,ax=axes[1,1])

# Set the title for the entire plot
fig.suptitle("Count plots for besroom,bathrooms,Floors,Waterfront,view")
plt.show()
```



In [11]:
```
## Create a correlation matrix for numerical variables
corr_matrix=data.corr()
## Visualize the correlation matrix using a heatmap
fig,ax=plt.subplots(figsize=(14,8))
sns.heatmap(corr_matrix,annot=True,cmap="coolwarm",ax=ax)
```

Out[11]: <AxesSubplot:>



In [12]:
```
data=data.drop(["zipcode","yr_built","condition"],axis=1)
data.head()
```

Out[12]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	grade	sqft_above	sqft_basement	yr_renovate
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	7	1180	0	
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	7	2170	400	199
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	6	770	0	
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	7	1050	910	
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	8	1680	0	

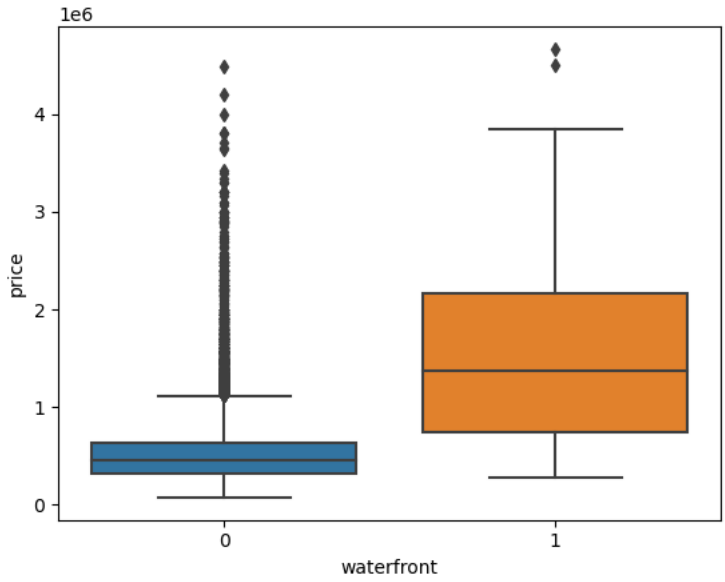
In [13]:
```
Floor_value_counts=data["floors"].value_counts().to_frame()
Floor_value_counts
```

Out[13]:

	floors
1.0	10680
2.0	8235
1.5	1910
3.0	613
2.5	160
3.5	8

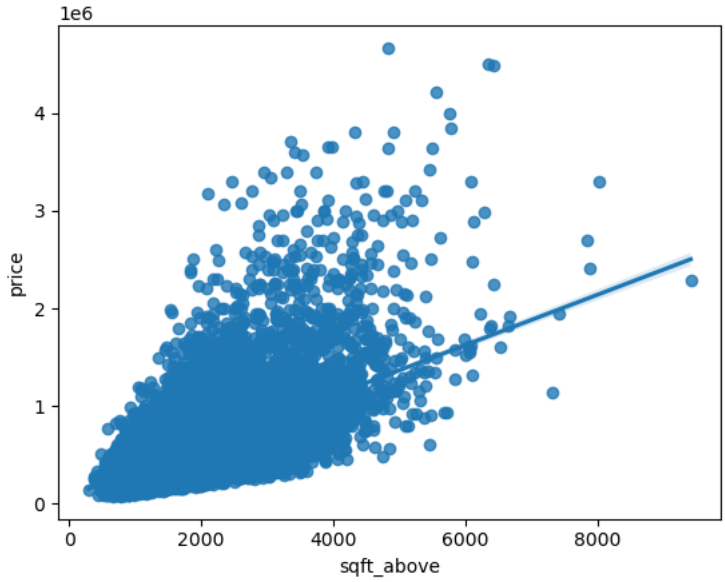
In [14]:
```
sns.boxplot(x="waterfront",y="price",data=data)
```

Out[14]: <AxesSubplot:xlabel='waterfront', ylabel='price'>



In [15]:
```
sns.regplot(x="sqft_above",y="price",data=data)
```

Out[15]: <AxesSubplot:xlabel='sqft_above', ylabel='price'>



Model Development and Evaluation

In [16]:
```
x=data[["long"]]
y=data[["price"]]
```

In [17]:
```
from sklearn.linear_model import LinearRegression
l1=LinearRegression()
l1.fit(x,y)
l1.score(x,y)
```

Out[17]: 0.0005517180183860493

In [18]:
```
x=data[["sqft_living"]]
y=data["price"]
l2=LinearRegression()
l2.fit(x,y)
l1.score(x,y)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:493: FutureWarning: The feature names should match those that were p
assed during fit. Starting version 1.2, an error will be raised.
Feature names unseen at fit time:
- sqft_living
Feature names seen at fit time, yet now missing:
- long

  warnings.warn(message, FutureWarning)
```

Out[18]:
```
-157359.95819891948
```

In [19]:
```
features=data[["floors","waterfront","lat","bedrooms","sqft_basement","view","bathrooms","sqft_living15","sqft_above","grade","sc
l3=LinearRegression()
l3.fit(features,y)
```

Out[19]:
```
LinearRegression()
```

In [20]:
```
l3.score(features,y)*100
```

Out[20]:
```
66.24153459947514
```

Thank you

In [ ]:
```

```