

```
In [1]: #Import libraries
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
import warnings
warnings.simplefilter(action="ignore", category=FutureWarning)
```

```
In [2]: #Set global variables
fpath = "/Users/chuying/Documents/dataeng_assessment/q5/"
```

```
In [3]: #Change the current working directory
os.chdir(fpath)
print("Current working dir: ", os.getcwd())
```

Current working dir: /Users/chuying/Documents/dataeng_assessment/q5

```
In [4]: #Read data files
cars = pd.read_csv('car.data', sep=",")
print(cars)
```

	buying	maint	doors	persons	lug_boot	safety	car_class
0	vhigh	vhigh	2	2	small	low	unacc
1	vhigh	vhigh	2	2	small	med	unacc
2	vhigh	vhigh	2	2	small	high	unacc
3	vhigh	vhigh	2	2	med	low	unacc
4	vhigh	vhigh	2	2	med	med	unacc
...
1723	low	low	5more	more	med	med	good
1724	low	low	5more	more	med	high	vgood
1725	low	low	5more	more	big	low	unacc
1726	low	low	5more	more	big	med	good
1727	low	low	5more	more	big	high	vgood

[1728 rows x 7 columns]

In [5]:

```
## Step 1: Business Understanding
# Create a machine learning model to predict the buying price given the fo.

# Maintenance = High
# Number of doors = 4
# Lug Boot Size = Big
# Safety = High
# Class Value = Good
```

In [6]:

```
#Step 2: Data Understanding
#2.1 Data is structured
#2.2 Entity of interest - Buying Price
#2.3 1 row = 1 record
#2.4 1 column = 1 field
#2.5 Yes, there is a data column to identify my event
#2.6 Categorise my variables - all are categorical variables
#Training model : Log regression
```

In [7]:

```
cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1728 entries, 0 to 1727
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  -
 0   buying      1728 non-null   object
 1   maint       1728 non-null   object
 2   doors       1728 non-null   object
 3   persons     1728 non-null   object
 4   lug_boot    1728 non-null   object
 5   safety      1728 non-null   object
 6   car_class   1728 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

In [8]:

```
cars.head()
```

Out[8]:

	buying	maint	doors	persons	lug_boot	safety	car_class
0	vhigh	vhigh	2	2	small	low	unacc
1	vhigh	vhigh	2	2	small	med	unacc
2	vhigh	vhigh	2	2	small	high	unacc
3	vhigh	vhigh	2	2	med	low	unacc
4	vhigh	vhigh	2	2	med	med	unacc

```
In [9]: #Step3: Data Preparation
#Check for missing data
check_null = cars.isnull().sum()
print(check_null)
```

```
buying      0
maint       0
doors       0
persons     0
lug_boot    0
safety      0
car_class   0
dtype: int64
```

```
In [10]: cars.describe()
```

```
Out[10]:
```

	buying	maint	doors	persons	lug_boot	safety	car_class
count	1728	1728	1728	1728	1728	1728	1728
unique	4	4	4	3	3	3	4
top	vhigh	vhigh	2	2	small	low	unacc
freq	432	432	432	576	576	576	1210

```
In [11]: cars.dtypes
```

```
Out[11]: buying      object
maint      object
doors      object
persons    object
lug_boot   object
safety     object
car_class  object
dtype: object
```

```
In [12]: #Remove column not required for prediction model
cars.drop(columns=['persons'], inplace = True)
```

```
In [13]: #Convert buying price to codes, replace string values for doors
cars['buying'].replace(to_replace=['vhigh', 'high', 'med', 'low'], value=[1, 2, 3, 4])
cars['doors'].replace(to_replace=['5more'], value=[5], inplace=True)
print(cars)
```

	buying	maint	doors	lug_boot	safety	car_class
0	10	vhigh	2	small	low	unacc
1	10	vhigh	2	small	med	unacc
2	10	vhigh	2	small	high	unacc
3	10	vhigh	2	med	low	unacc
4	10	vhigh	2	med	med	unacc
...
1723	1	low	5	med	med	good
1724	1	low	5	med	high	vgood
1725	1	low	5	big	low	unacc
1726	1	low	5	big	med	good
1727	1	low	5	big	high	vgood

[1728 rows x 6 columns]

In [14]:

```
#Step 4: Build training model
#Convert categorical data into dummy or indicator variables
#X = predictor variable
X = cars[['maint', 'doors', 'lug_boot', 'safety', 'car_class']]
X = pd.get_dummies(data=X)
X.head()
```

Out[14]:

	maint_high	maint_low	maint_med	maint_vhigh	doors_5	doors_2	doors_3	doors_4
0	0	0	0	1	0	1	0	0
1	0	0	0	1	0	1	0	0
2	0	0	0	1	0	1	0	0
3	0	0	0	1	0	1	0	0
4	0	0	0	1	0	1	0	0

In [15]:

```
#Y = target variable
Y = cars['buying']
Y
```

Out[15]:

```
0      10
1      10
2      10
3      10
4      10
..
1723    1
1724    1
1725    1
1726    1
1727    1
Name: buying, Length: 1728, dtype: int64
```

```
In [16]: #Create train and test data  
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.4, ra  
print(X_train.shape)  
print(X_test.shape)  
print(y_train.shape)  
print(y_test.shape)
```

```
(1036, 18)  
(692, 18)  
(1036,)  
(692,)
```

```
In [17]: #Fit training data into linear regression model  
model = LinearRegression()  
model.fit(X_train,y_train)
```

```
Out[17]: LinearRegression()
```

```
In [18]: #print the intercept  
print(model.intercept_)
```

```
-585637056117938.8
```

```
In [19]: #Find coefficient for each predictors  
# A positive sign indicates that as the predictor variable increases, the  
# A negative sign indicates that as the predictor variable increases, the  
coeff_parameter = pd.DataFrame(model.coef_,X.columns,columns=['Coefficient  
coeff_parameter
```

Out [19]:

	Coefficient
maint_high	6.108197e+13
maint_low	6.108197e+13
maint_med	6.108197e+13
maint_vhigh	6.108197e+13
doors_5	-1.082787e+14
doors_2	-1.082787e+14
doors_3	-1.082787e+14
doors_4	-1.082787e+14
lug_boot_big	3.657272e+14
lug_boot_med	3.657272e+14
lug_boot_small	3.657272e+14
safety_high	3.208168e+14
safety_low	3.208168e+14
safety_med	3.208168e+14
car_class_acc	-5.371023e+13
car_class_good	-5.371023e+13
car_class_unacc	-5.371023e+13
car_class_vgood	-5.371023e+13

In [20]:

```
#Fit test data into model
predictions = model.predict(X_test)
predictions
```

Out [20]:

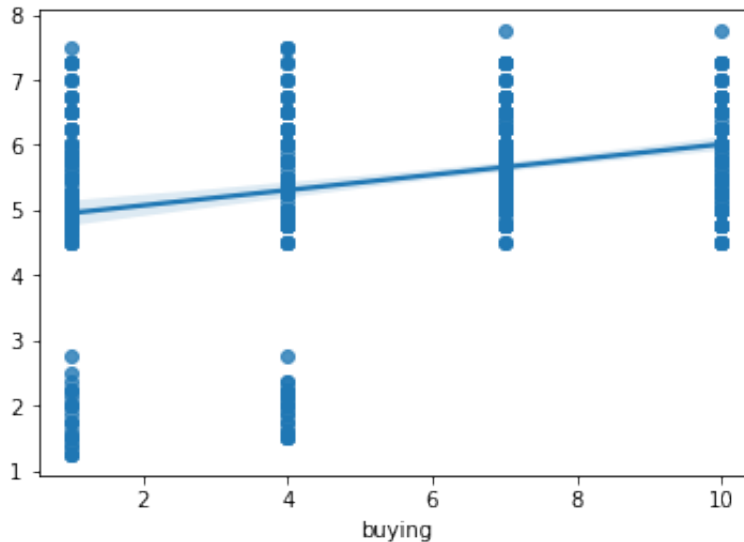
```
array([ 6.5 ,  6.75,  7.   ,  5.5 ,  5.5 ,  6.5 ,  6.5 ,  6.5 ,  7.5 ,
        5.75,  5.   ,  6.   ,  7.   ,  6.   ,  6.5 ,  5.875,  6.   ,  6.   ,
        5.   ,  7.   ,  6.   ,  6.25,  7.25,  6.75,  5.625,  4.5 ,  4.75 ,
        6.75,  6.25,  5.75,  5.25,  5.75,  6.   ,  5.   ,  6.   ,  5.   ,
        6.25,  6.5 ,  7.25,  4.5 ,  4.75,  7.25,  6.5 ,  5.   ,  4.75 ,
        5.75,  5.25,  5.   ,  5.25,  5.5 ,  4.5 ,  5.25,  6.75,  5.5 ,
        5.25,  5.25,  4.75,  6.   ,  7.75,  6.5 ,  5.5 ,  5.   ,  2.   ,
        6.   ,  5.375,  4.75,  5.875,  7.   ,  7.25,  6.5 ,  5.25,  6.   ,
        7.25,  5.5 ,  7.   ,  4.875,  6.   ,  1.5 ,  6.25,  6.25,  6.25 ,
        6.75,  5.25,  5.   ,  6.5 ,  6.5 ,  1.25,  4.5 ,  5.5 ,  5.25 ,
        7.75,  6.25,  2.   ,  6.25,  1.5 ,  6.   ,  6.25,  2.375,  4.5 ,
        6.75,  2.   ,  6.   ,  6.5 ,  5.25,  5.875,  4.5 ,  7.25,  6.   ,
        6.75,  1.875,  5.25,  5.75,  4.5 ,  5.25,  5.25,  6.5 ,  6.   ,
        5.75,  6.75,  5.25,  7.25,  4.75,  5.25,  6.75,  6.   ,  6.5 ,
        6.25,  2.   ,  5.25,  7.25,  6.375,  2.25,  5.5 ,  4.5 ,  5.375,
        7.25,  6.25,  5.5 ,  4.75,  4.75,  6.   ,  5.5 ,  5.   ,  1.25 ,
        4.75,  4.75,  5.5 ,  5.875,  5.25,  5.   ,  5.25,  5.375,  6.75 ,
        6.75,  6.75,  6.75 ,  5.125,  6.   ,  7.25 ,  6.375,  6.   ,  5.   ,
```

6.75 , 6. , 7.5 , 6.75 , 5.75 , 6.25 , 6.25 , 5.375, 6.5 ,
7.25 , 5.5 , 6.25 , 6. , 7.25 , 5.5 , 5.5 , 5.5 , 2.25 ,
6.75 , 6.5 , 5.75 , 5.75 , 5. , 6.25 , 6.5 , 5.875, 4.5 ,
5.25 , 4.5 , 6.25 , 5. , 4.5 , 6.25 , 4.75 , 5.875, 6.75 ,
5.875, 6.5 , 5.25 , 5.75 , 5.25 , 7. , 6.25 , 5. , 4.75 ,
4.75 , 5.25 , 5.75 , 5. , 6. , 5.375, 4.5 , 5.25 , 6.75 ,
5.75 , 4.75 , 5.25 , 5. , 7.25 , 6. , 7.25 , 6.75 , 6. ,
5.75 , 5.875, 5.5 , 5. , 6.75 , 2.375, 6.75 , 6.75 , 6. ,
1.75 , 6.75 , 4.5 , 5. , 6.5 , 2.125, 5.75 , 5.75 , 6.25 ,
5.5 , 6.25 , 7. , 6.5 , 5. , 5.25 , 5.75 , 5. , 5.875,
5.5 , 5.25 , 6.75 , 5. , 7. , 5.75 , 5.375, 5. , 6.25 ,
4.75 , 6. , 6.25 , 5.625, 5.125, 5.75 , 6.5 , 2.375, 5.75 ,
6. , 4.75 , 4.5 , 5.75 , 1.25 , 5.625, 6.5 , 6.25 , 6.75 ,
5. , 5. , 7.5 , 6.75 , 5.5 , 6.75 , 5.75 , 4.75 , 7.5 ,
6.25 , 5.5 , 4.75 , 4.75 , 7.25 , 5. , 7. , 5.875, 6.75 ,
4.5 , 4.5 , 4.75 , 6.5 , 6.25 , 7.25 , 5.25 , 5.5 , 1.875,
1.625, 1.5 , 6. , 6. , 6.25 , 5.5 , 5.5 , 4.875, 6.25 ,
5.75 , 5.875, 5.25 , 1.375, 6. , 4.875, 5.25 , 5.25 , 6.5 ,
5.5 , 1.5 , 5. , 6.25 , 6.25 , 6. , 5.75 , 6. , 5.75 ,
5.75 , 7. , 2.25 , 5.375, 6.75 , 6. , 6.25 , 5.5 , 6.75 ,
1.5 , 6. , 6.25 , 5.875, 7.25 , 5. , 4.5 , 5.25 , 5. ,
6. , 6.5 , 4.875, 7.25 , 5.75 , 5.625, 5.75 , 5.5 , 5.5 ,
5.5 , 6.5 , 5.5 , 5. , 7. , 6. , 4.625, 4.75 , 5. ,
5.75 , 5.5 , 5.75 , 5.375, 5.25 , 6.75 , 5.5 , 5.5 , 6.5 ,
6.5 , 4.75 , 5.5 , 5. , 5.25 , 6.25 , 6. , 5.75 , 6.75 ,
6.5 , 6.5 , 4.875, 5.5 , 7.25 , 5.5 , 5.625, 5. , 6.25 ,
5.75 , 5.125, 4.75 , 5.875, 4.75 , 6. , 6.5 , 6.75 , 6.25 ,
6.75 , 6.25 , 7. , 6.375, 1.75 , 7. , 5. , 5.25 , 1.5 ,
2.75 , 5.375, 6.25 , 4.75 , 6.75 , 2.75 , 6.75 , 5. , 7. ,
5.75 , 5. , 5. , 6.5 , 7.25 , 1.625, 6.5 , 5. , 5. ,
1.75 , 5. , 5.5 , 5.75 , 6.75 , 2. , 6.5 , 6.25 , 1.75 ,
4.75 , 5.75 , 5.5 , 4.75 , 5. , 5.75 , 5. , 5.625, 5.25 ,
5.625, 4.75 , 6.75 , 4.75 , 5.5 , 5.75 , 5.5 , 6. , 6. ,
4.625, 5.75 , 5.75 , 6. , 5. , 5.25 , 1.625, 5. , 4.75 ,
7. , 4.75 , 5.5 , 5. , 5.125, 6.25 , 5.75 , 5.25 , 5.25 ,
5.5 , 6. , 1.5 , 5. , 6.25 , 4.75 , 4.75 , 7. , 7.25 ,
5.25 , 4.75 , 5. , 4.75 , 7. , 2.25 , 5.25 , 4.5 , 1.25 ,
5.375, 7.25 , 5.75 , 6.5 , 5.125, 5.125, 2. , 6.25 , 6.25 ,
7. , 5.5 , 6.5 , 4.75 , 6. , 5. , 5.75 , 6.25 , 6.5 ,
6.25 , 6.5 , 6. , 7.5 , 6.5 , 6.5 , 6. , 5.75 , 6. ,
5.75 , 6.75 , 7.25 , 5.125, 7.25 , 5. , 4.625, 5. , 6.25 ,
6.5 , 5.75 , 5.5 , 6.25 , 1.5 , 6.25 , 4.5 , 6.5 , 7. ,
1.5 , 4.75 , 5.375, 5. , 5. , 4.5 , 6.75 , 4.5 , 6. ,
5. , 6.5 , 6.75 , 2.125, 4.75 , 4.75 , 6. , 5. , 4.75 ,
2. , 5.25 , 2.25 , 5.25 , 5.25 , 6.5 , 6.5 , 5.75 , 5.5 ,
5.5 , 2. , 6.75 , 6. , 2.125, 5.25 , 5. , 5.625, 6.5 ,
5.75 , 5.875, 4.75 , 5.125, 4.875, 6. , 5.75 , 5.5 , 7.25 ,
6.5 , 5.25 , 6.25 , 6.75 , 5.75 , 6.25 , 5.75 , 6.25 , 4.75 ,
5. , 5.625, 6.25 , 2.5 , 6.25 , 6. , 5.75 , 5. , 6.75 ,
6. , 4.875, 5.5 , 6. , 6.5 , 5.25 , 6.25 , 2.125, 5.5 ,
5.5 , 6. , 4.75 , 6.75 , 5.25 , 4.875, 4.75 , 5.375, 5. ,
4.5 , 5.25 , 1.875, 5.5 , 7.25 , 5. , 6.5 , 1.5 , 4.75 ,
5.75 , 6.25 , 5.75 , 6.5 , 6. , 5. , 5. , 5.125, 5. ,
6.25 , 7.25 , 6. , 5.75 , 5.5 , 5.5 , 5.25 , 5. , 6.25 ,
1.75 , 6.75 , 6. , 5.25 , 5.125, 5.125, 5.5 , 5. , 5.25 ,
5.625, 5. , 5.75 , 5.25 , 2. , 6.25 , 6. , 6.25 , 6.75 ,
5.5 , 2.25 , 1.375, 5. , 5.25 , 2. , 6.75 , 6.75 , 5.5 ,

```
5.    , 5.75 , 5.5  , 5.375, 5.5  , 5.    , 5.5  , 5.25 , 5.25 ,
5.5   , 5.    , 7.25 , 7.    , 5.75 , 5.25 , 6.25 , 5.75 ])
```

```
In [21]: #Plot graph for test data
sns.regplot(y_test,predictions)
```

```
Out[21]: <AxesSubplot:xlabel='buying'>
```



```
In [22]: #Predict output for specific input parameters
# Maintenance = High
# Number of doors = 4
# Lug Boot Size = Big
# Safety = High
# Class Value = Good

predicted_buying = model.predict([[1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0]])
predicted_buying #i.e. Predicted buying price is low
```

/usr/local/lib/python3.9/site-packages/sklearn/base.py:445: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names

```
Out[22]: warnings.warn(
array([1.75])
```

```
In [23]: #Find mean square error,the lower the MSE the higher the accuracy of prediction
y_pred = model.predict(X_test)
mean_squared_error(y_test,y_pred)
```

```
Out[23]: 10.070199602601155
```

```
In [24]: np.sqrt(mean_squared_error(y_test, y_pred))
```

```
Out[24]: 3.1733577804277213
```