Solution for 1 que

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
df = pd.read_csv("data.csv")
```

```
df.shape
```

```
(100, 3)
```

```
df.head(30)
```

	F	N	Prprice per square foot
0	0.44	0.68	511.14
1	0.99	0.23	717.10
2	0.84	0.29	607.91
3	0.28	0.45	270.40
4	0.07	0.83	289.88
5	0.66	0.80	830.85
6	0.73	0.92	1038.09
7	0.57	0.43	455.19
8	0.43	0.89	640.17
9	0.27	0.95	511.06
10	0.43	0.06	177.03
11	0.87	0.91	1242.52
12	0.78	0.69	891.37
13	0.90	0.94	1339.72
14	0.41	0.06	169.88
15	0.52	0.17	276.05
16	0.47	0.66	517.43
17	0.65	0.43	522.25
18	0.85	0.64	932.21
19	0.93	0.44	851.25
20	0.41	0.93	640.11
21	0.36	0.43	308.68
22	0.78	0.85	1046.05
23	0.69	0.07	332.40
24	0.04	0.52	171.85
25	0.17	0.15	109.55

	F	N	Prprice per square foot
26	0.68	0.13	361.97
27	0.84	0.60	872.21
28	0.38	0.40	303.70
29	0.12	0.65	256.38

```
df.info
```

```
<bound method DataFrame.info of        F     N  Prprice per square foot
0   0.44  0.68                   511.14
1   0.99  0.23                   717.10
2   0.84  0.29                   607.91
3   0.28  0.45                   270.40
4   0.07  0.83                   289.88
..   ...   ...                      ...
95  0.99  0.13                   636.22
96  0.28  0.46                   272.12
97  0.87  0.36                   696.65
98  0.23  0.87                   434.53
99  0.77  0.36                   593.86

[100 rows x 3 columns]>
```

```
# Preprocessing
df.isnull().sum()
```

```
F                          0
N                          0
Prprice per square foot    0
dtype: int64
```

```
# Independent and Dependent variables

X = df.iloc[:,:-1].values

y = df.iloc[:,-1].values
```

```
X.shape , y.shape
```

```
((100, 2), (100,))
```

```
X
```

```
array([[0.44, 0.68],
       [0.99, 0.23],
       [0.84, 0.29],
       [0.28, 0.45],
       [0.07, 0.83],
```

```
[0.66, 0.8 ],
[0.73, 0.92],
[0.57, 0.43],
[0.43, 0.89],
[0.27, 0.95],
[0.43, 0.06],
[0.87, 0.91],
[0.78, 0.69],
[0.9 , 0.94],
[0.41, 0.06],
[0.52, 0.17],
[0.47, 0.66],
[0.65, 0.43],
[0.85, 0.64],
[0.93, 0.44],
[0.41, 0.93],
[0.36, 0.43],
[0.78, 0.85],
[0.69, 0.07],
[0.04, 0.52],
[0.17, 0.15],
[0.68, 0.13],
[0.84, 0.6 ],
[0.38, 0.4 ],
[0.12, 0.65],
[0.62, 0.17],
[0.79, 0.97],
[0.82, 0.04],
[0.91, 0.53],
[0.35, 0.85],
[0.57, 0.69],
[0.52, 0.22],
[0.31, 0.15],
[0.6 , 0.02],
[0.99, 0.91],
[0.48, 0.76],
[0.3 , 0.19],
[0.58, 0.62],
[0.65, 0.17],
[0.6 , 0.69],
[0.95, 0.76],
[0.47, 0.23],
[0.15, 0.96],
[0.01, 0.03],
[0.26, 0.23],
[0.01, 0.11],
[0.45, 0.87],
[0.09, 0.97],
[0.96, 0.25],
```

```
       [0.63, 0.58],
       [0.06, 0.42],
       [0.1 , 0.24],
       [0.26, 0.62],
       [0.41, 0.15],
       [0.91, 0.95],
       [0.83, 0.64],
       [0.44, 0.64],
       [0.2 , 0.4 ],
       [0.43, 0.12],
       [0.21, 0.22],
       [0.88, 0.4 ],
       [0.31, 0.87],
       [0.99, 0.99],
       [0.23, 0.26],
       [0.79, 0.12],
       [0.02, 0.28],
       [0.89, 0.48],
       [0.02, 0.56],
       [0.92, 0.03],
       [0.72, 0.34],
       [0.3 , 0.99],
       [0.86, 0.66],
       [0.47, 0.65],
       [0.79, 0.94],
       [0.82, 0.96],
       [0.9 , 0.42],
       [0.19, 0.62],
       [0.7 , 0.57],
       [0.7 , 0.61],
       [0.69, 0.  ],
       [0.98, 0.3 ],
       [0.3 , 0.08],
       [0.85, 0.49],
       [0.73, 0.01],
       [1.  , 0.23],
       [0.42, 0.94],
       [0.49, 0.98],
       [0.89, 0.68],
       [0.22, 0.46],
       [0.34, 0.5 ],
       [0.99, 0.13],
       [0.28, 0.46],
       [0.87, 0.36],
       [0.23, 0.87],
       [0.77, 0.36]])
```

```
y
```

```
array([ 511.14,  717.1 ,  607.91,  270.4 ,  289.88,  830.85, 1038.09,
        455.19,  640.17,  511.06,  177.03, 1242.52,  891.37, 1339.72,
        169.88,  276.05,  517.43,  522.25,  932.21,  851.25,  640.11,
        308.68, 1046.05,  332.4 ,  171.85,  109.55,  361.97,  872.21,
        303.7 ,  256.38,  341.2 , 1194.63,  408.6 ,  895.54,  518.25,
        638.75,  301.9 ,  163.38,  240.77, 1449.05,  609.  ,  174.59,
        593.45,  355.96,  671.46, 1193.7 ,  278.88,  411.4 ,   42.08,
        166.19,   58.62,  642.45,  368.14,  702.78,  615.74,  143.79,
        109.  ,  328.28,  205.16, 1360.49,  905.83,  487.33,  202.76,
        202.01,  148.87,  745.3 ,  503.04, 1563.82,  165.21,  438.4 ,
         98.47,  819.63,  174.44,  483.13,  534.24,  572.31,  957.61,
        518.29, 1143.49, 1211.31,  784.74,  283.7 ,  684.38,  719.46,
        292.23,  775.68,  130.77,  801.6 ,  323.55,  726.9 ,  661.12,
        771.11, 1016.14,  237.69,  325.89,  636.22,  272.12,  696.65,
        434.53,  593.86])
```

```
#Train and Test Data

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=0
```

```
X_train.shape , X_test.shape , y_train.shape , y_test.shape
```

```
((80, 2), (20, 2), (80,), (20,))
```

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X_train,y_train)
```

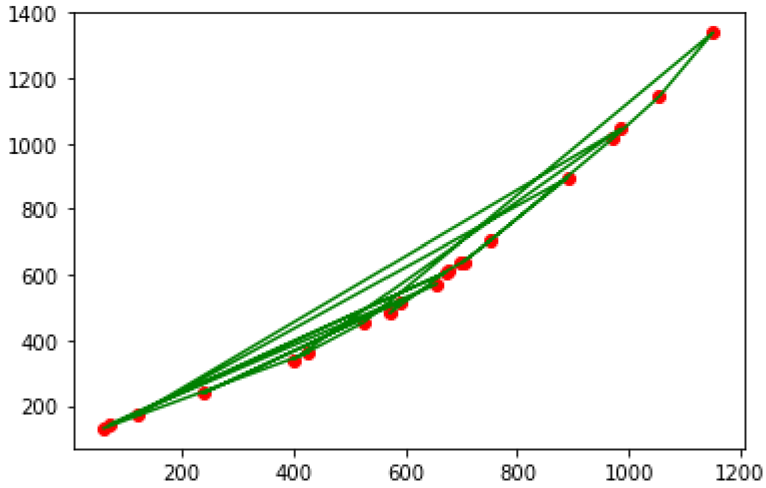
```
LinearRegression()
```

```
y_pred = reg.predict(X_test)
y_pred
```

```
array([ 426.09146539,   58.82595194,  671.86958179,   70.59925474,
        655.72394234,  237.66554823,  588.89726222,  571.74180841,
        677.2517328 ,  698.94769006,  751.25407187,  971.69207474,
       1054.21622072, 1151.03617141,  526.05107197,  399.51785749,
        986.38053984,  118.5888131 ,  890.9059235 ,  704.55433467])
```

```
plt.scatter(y_pred,y_test,color='red')
plt.plot(y_pred,y_test,color='green')
```

```
[<matplotlib.lines.Line2D at 0x1e9c3ef2760>]
```




```
reg.predict([[0.44,0.68]])
```

array([575.61045828])

```
reg.predict([[0.43,0.06]])
```

array([160.13089288])

```
reg.predict([[0.90,0.94]])
```

array([1151.03617141])

Polynomial Regression Model

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 3)
X_poly = poly_reg.fit_transform(X_train)
lg1 = LinearRegression()
lg1.fit(X_poly,y_train)
```

LinearRegression()

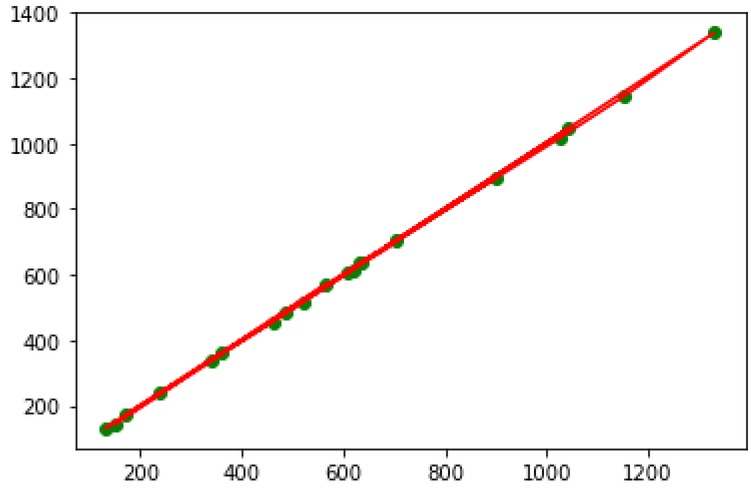
```
y_pred1 = lg1.predict(poly_reg.fit_transform(X_test))
```

```
y_pred1
```

```
array([ 359.50360287,  131.23834372,  609.16930978,  149.98412841,
        566.72101623,  237.76303341,  520.60934004,  485.30770519,
        619.83089854,  632.50577664,  703.21376307, 1024.98728581,
       1151.24558258, 1330.85054638,  460.70580041,  340.84136488,
       1041.64246977,  170.75998438,  901.57838952,  636.04094675])
```

```
plt.scatter(y_pred1,y_test,color='green')
plt.plot(y_pred1,y_test,color='red')
```

[<matplotlib.lines.Line2D at 0x1e9c4f31520>]



Conclusion

==Polynomial Regression is a form of Linear regression known as a special case of Multiple linear regression which estimates the relationship as an nth degree polynomial. Polynomial Regression is sensitive to outliers so the presence of one or two outliers can also badly affect the performance. ==There are 100 rows and 3 columns.I used two models for this dataset. 1.Multiple Linear regression model and 2.Polynomial Regression model.

== Polynomial regression model gives best regression line this came under observation as i plottes scatter graph for both

So as per my observation for these Data ,Polynomial Regression model is better than Multiple Linear regression model.

Solution for 2 que

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
df = pd.read_csv("sales.csv")
```

```
df
```

	ID	Age	Income	Gender	Marital Status	Buys
0	1	<21	High	Male	Single	No
1	2	<21	High	Male	Married	No
2	3	21-35	High	Male	Single	Yes
3	4	>35	Medium	Male	Single	Yes
4	5	>35	Low	Female	Single	Yes
5	6	>35	Low	Female	Married	No
6	7	21-35	Low	Female	Married	Yes
7	8	<21	Medium	Male	Single	No

	ID	Age	Income	Gender	Marital Status	Buys
8	9	<21	Low	Female	Married	Yes
9	10	>35	Medium	Female	Single	Yes
10	11	<21	Medium	Female	Married	Yes
11	12	21-35	Medium	Male	Married	Yes
12	13	21-35	High	Female	Single	Yes
13	14	>35	Medium	Male	Married	No

```
df.shape
```

```
(14, 6)
```

```
df.head(14)
```

	ID	Age	Income	Gender	Marital Status	Buys
0	1	<21	High	Male	Single	No
1	2	<21	High	Male	Married	No
2	3	21-35	High	Male	Single	Yes
3	4	>35	Medium	Male	Single	Yes
4	5	>35	Low	Female	Single	Yes
5	6	>35	Low	Female	Married	No
6	7	21-35	Low	Female	Married	Yes
7	8	<21	Medium	Male	Single	No
8	9	<21	Low	Female	Married	Yes
9	10	>35	Medium	Female	Single	Yes
10	11	<21	Medium	Female	Married	Yes
11	12	21-35	Medium	Male	Married	Yes
12	13	21-35	High	Female	Single	Yes
13	14	>35	Medium	Male	Married	No

```
df["Buys"].value_counts()
```

```
Yes    9
No     5
Name: Buys, dtype: int64
```

```
df.head()
```

	ID	Age	Income	Gender	Marital Status	Buys
0	1	<21	High	Male	Single	No
1	2	<21	High	Male	Married	No
2	3	21-35	High	Male	Single	Yes
3	4	>35	Medium	Male	Single	Yes

	ID	Age	Income	Gender	Marital Status	Buys
4	5	>35	Low	Female	Single	Yes

```
# Preprocessing

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df = df.apply(le.fit_transform)
```

```
df
```

	ID	Age	Income	Gender	Marital Status	Buys
0	0	1	0	1	1	0
1	1	1	0	1	0	0
2	2	0	0	1	1	1
3	3	2	2	1	1	1
4	4	2	1	0	1	1
5	5	2	1	0	0	0
6	6	0	1	0	0	1
7	7	1	2	1	1	0
8	8	1	1	0	0	1
9	9	2	2	0	1	1
10	10	1	2	0	0	1
11	11	0	2	1	0	1
12	12	0	0	0	1	1
13	13	2	2	1	0	0

```
df = df.drop(['ID'],axis =1)
```

```
df
```

	Age	Income	Gender	Marital Status	Buys
0	1	0	1	1	0
1	1	0	1	0	0
2	0	0	1	1	1
3	2	2	1	1	1
4	2	1	0	1	1
5	2	1	0	0	0
6	0	1	0	0	1
7	1	2	1	1	0
8	1	1	0	0	1
9	2	2	0	1	1

	Age	Income	Gender	Marital Status	Buys
10	1	2	0	0	1
11	0	2	1	0	1
12	0	0	0	1	1
13	2	2	1	0	0

```
df.head()
```

	Age	Income	Gender	Marital Status	Buys
0	1	0	1	1	0
1	1	0	1	0	0
2	0	0	1	1	1
3	2	2	1	1	1
4	2	1	0	1	1

```
#Independent and Dependent variables
x = df.iloc[:,:-1].values

y = df.iloc[:,-1].values
```

```
x
```

```
array([[1, 0, 1, 1],
       [1, 0, 1, 0],
       [0, 0, 1, 1],
       [2, 2, 1, 1],
       [2, 1, 0, 1],
       [2, 1, 0, 0],
       [0, 1, 0, 0],
       [1, 2, 1, 1],
       [1, 1, 0, 0],
       [2, 2, 0, 1],
       [1, 2, 0, 0],
       [0, 2, 1, 0],
       [0, 0, 0, 1],
       [2, 2, 1, 0]])
```

```
y
```

```
array([0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0])
```

```
#Train and Test Data

from sklearn.model_selection import train_test_split
X_train , X_test , y_train , y_test = train_test_split(X,y,train_size=0.25,random_state
```

```
#Model Building

from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion='entropy' , random_state=0)
classifier.fit(X_train,y_train)
```

```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

```
y_pred = classifier.predict(X_test)
```

```
y_pred
```

```
array([0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0])
```

```
from sklearn.metrics import confusion_matrix , accuracy_score

#Confusion Matrix
cm = confusion_matrix(y_test , y_pred)
cm
```

```
array([[3, 0],
       [5, 3]], dtype=int64)
```

```
#Accuracy Score
accuracy_score(y_test , y_pred)
```

```
0.5454545454545454
```

```
classifier.predict([[0,0,1,1]])
```

```
array([1])
```

```
classifier.predict([[1,1,0,0]])
```

```
array([0])
```

conclusion

```
#after analyzing data i concluded that root node is age for decision tree.
#According to the decision tree, you have made from the previous Training data set, wha
#[Age < 21, Income = Low, Gender = Female, Marital Status = Married] so the prediction
```

```
pip install jovian --upgrade
```

```
Requirement already satisfied: jovian in c:\users\shubham\anaconda3\lib\site-packages
(0.2.41)
Requirement already satisfied: pyyaml in c:\users\shubham\anaconda3\lib\site-packages
```

```
(from jovian) (6.0)
Requirement already satisfied: click in c:\users\shubham\anaconda3\lib\site-packages
(from jovian) (8.0.3)
Requirement already satisfied: requests in c:\users\shubham\anaconda3\lib\site-packages
(from jovian) (2.26.0)
Requirement already satisfied: uuid in c:\users\shubham\anaconda3\lib\site-packages
(from jovian) (1.30)
Requirement already satisfied: colorama in c:\users\shubham\anaconda3\lib\site-packages
(from click->jovian) (0.4.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\shubham\anaconda3\lib\site-
packages (from requests->jovian) (3.2)
Requirement already satisfied: charset-normalizer~=2.0.0 in
c:\users\shubham\anaconda3\lib\site-packages (from requests->jovian) (2.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
c:\users\shubham\anaconda3\lib\site-packages (from requests->jovian) (1.26.7)
Requirement already satisfied: certifi>=2017.4.17 in
c:\users\shubham\anaconda3\lib\site-packages (from requests->jovian) (2021.10.8)
Note: you may need to restart the kernel to use updated packages.
```

```python
import jovian
```

```python
jovian.commit()
```

```python

```