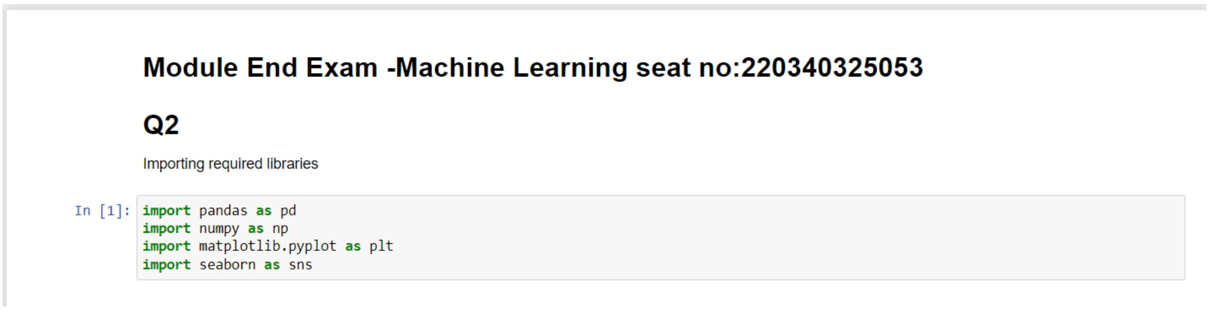
200340325053



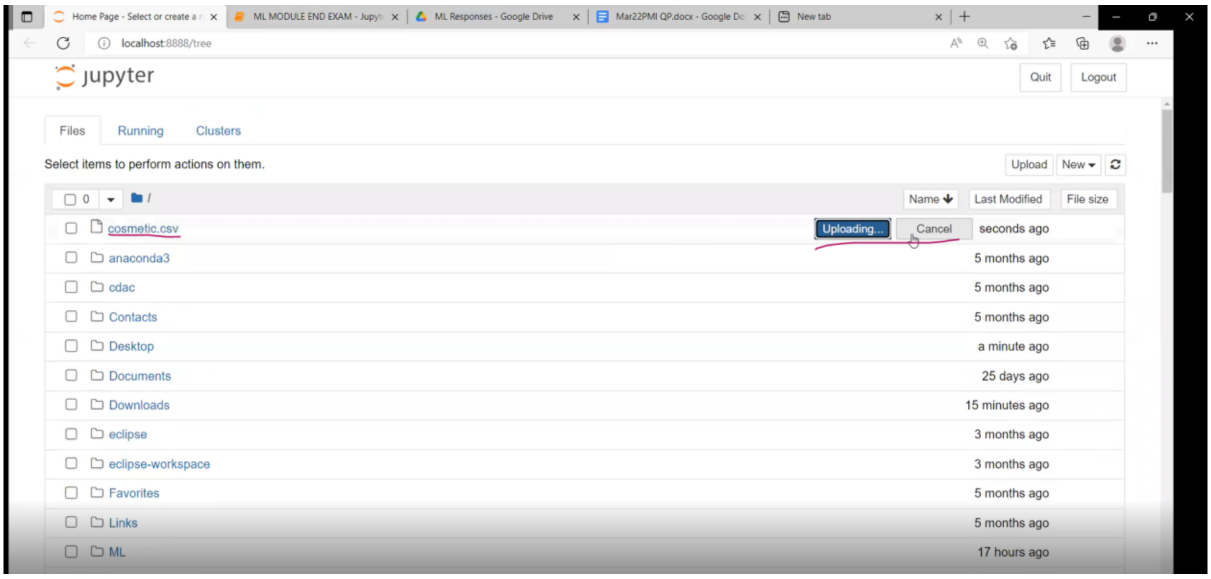
Module End Exam -Machine Learning seat no:220340325053

Q2

Importing required libraries

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

Upload csv file in jupyter

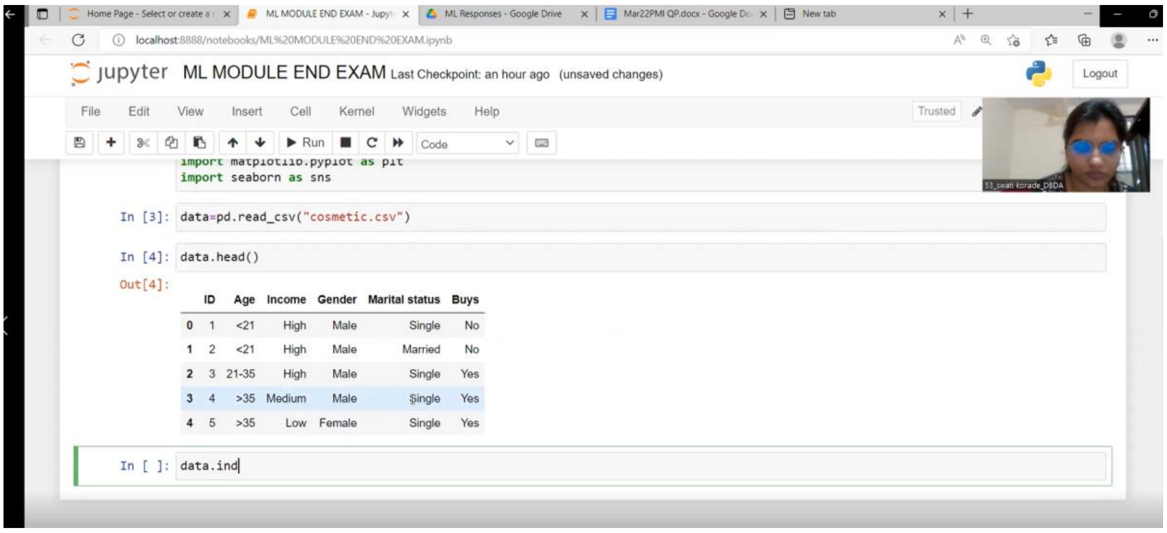


Select items to perform actions on them.

Name	Last Modified
cosmetic.csv	seconds ago
anaconda3	5 months ago
cdac	5 months ago
Contacts	5 months ago
Desktop	a minute ago
Documents	25 days ago
Downloads	15 minutes ago
eclipse	3 months ago
eclipse-workspace	3 months ago
Favorites	5 months ago
Links	5 months ago
ML	17 hours ago

EDA

#Loading and show some recoreds in data



localhost:8888/notebooks/ML%20MODULE%20END%20EXAM.ipynb
jupyter
ML MODULE END EXAM Last Checkpoint: an hour ago (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Run
Code
53_swati korade_DBDA

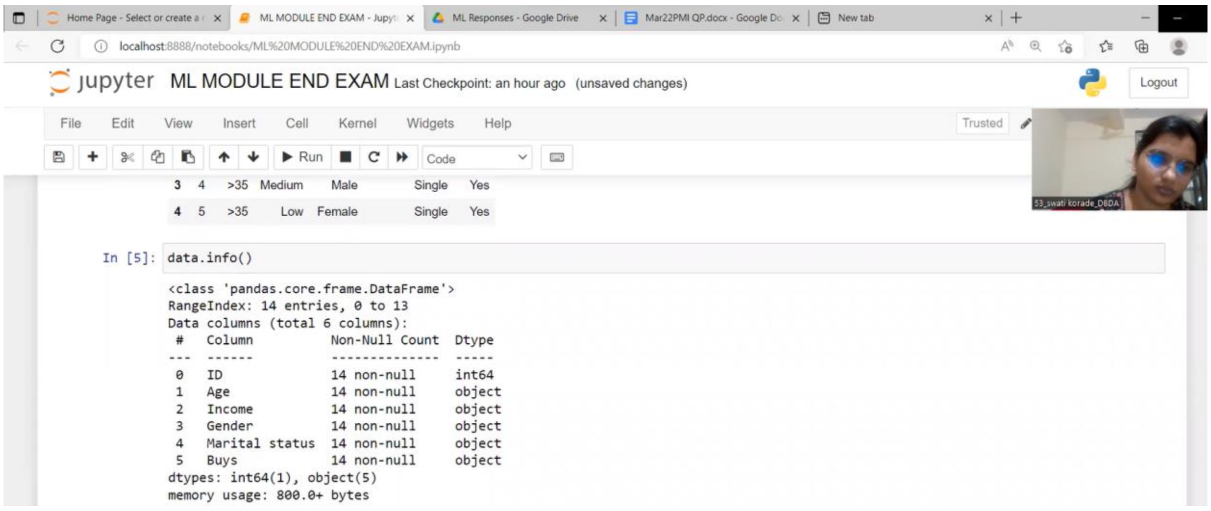
```
        import matplotlib.pyplot as plt
        import seaborn as sns

In [3]: data=pd.read_csv("cosmetic.csv")

In [4]: data.head()

Out[4]:
```

	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

```
In [ ]: data.ind
```

200340325053

#show info about data



```
3   4   >35   Medium   Male     Single   Yes
4   5   >35   Low      Female   Single   Yes
```

```
In [5]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14 entries, 0 to 13
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   ID              14 non-null     int64 
 1   Age             14 non-null     object
 2   Income          14 non-null     object
 3   Gender          14 non-null     object
 4   Marital status  14 non-null     object
 5   Buys            14 non-null     object
dtypes: int64(1), object(5)
memory usage: 800.0+ bytes
```

#drop unneccceracy column and check missing values

```
data=data.drop('ID',axis=1)                     #id column is not required so i will drop it
data.head()
```

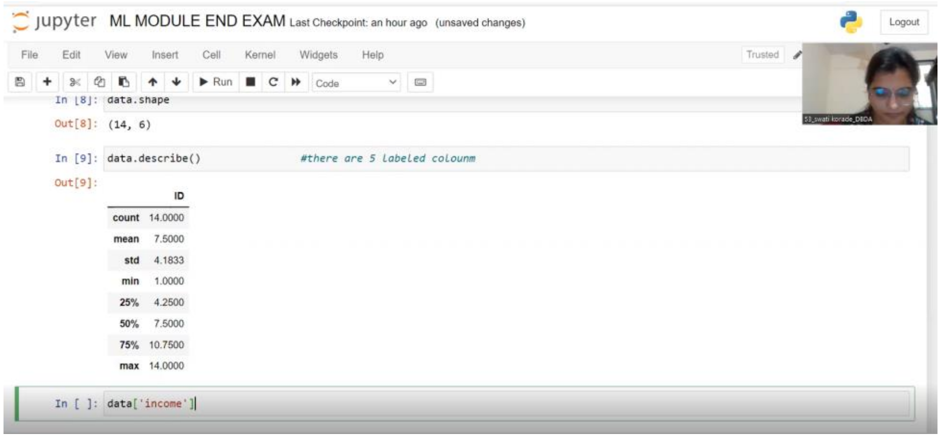
	Age	Income	Gender	Marital status	Buys
0	<21	High	Male	Single	No
1	<21	High	Male	Married	No
2	21-35	High	Male	Single	Yes
3	>35	Medium	Male	Single	Yes
4	>35	Low	Female	Single	Yes

```
data.isnull().sum()                     # checking null values in the data set or not
```

```
Age               0
Income            0
Gender            0
Marital status    0
Buys              0
dtype: int64
```

from above we can say that there is no null values in the data

# check statistics about data



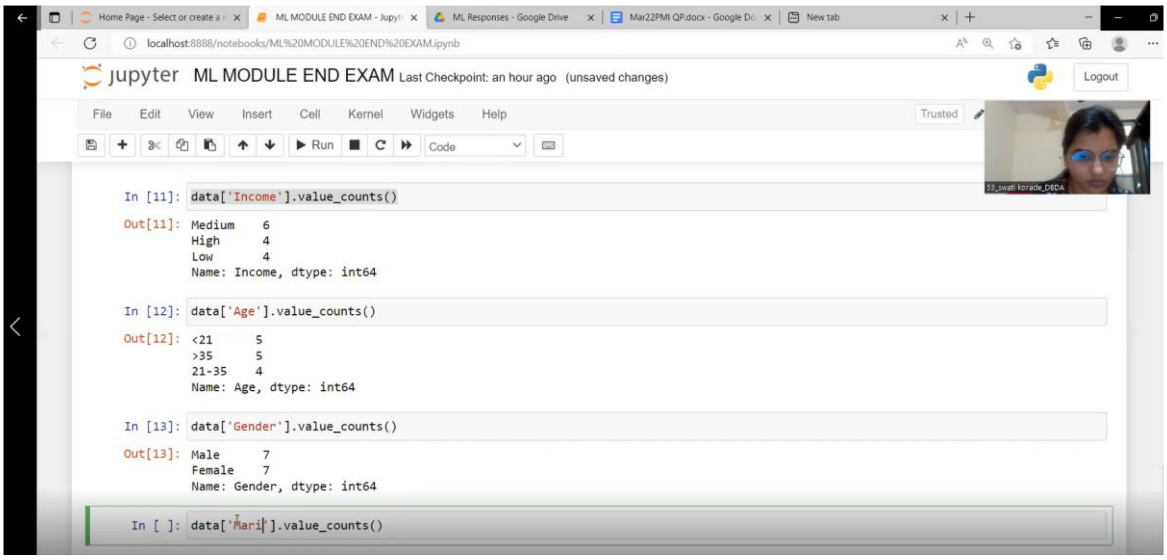
```
In [8]: data.shape
Out[8]: (14, 6)

In [9]: data.describe()                     #there are 5 labeled colounm
Out[9]:
```

	ID
count	14.0000
mean	7.5000
std	4.1833
min	1.0000
25%	4.2500
50%	7.5000
75%	10.7500
max	14.0000

```
In [ ]: data['income']
```

But here all data in terms of categorical form so need to label it.

# count the number of observation in each group of variables



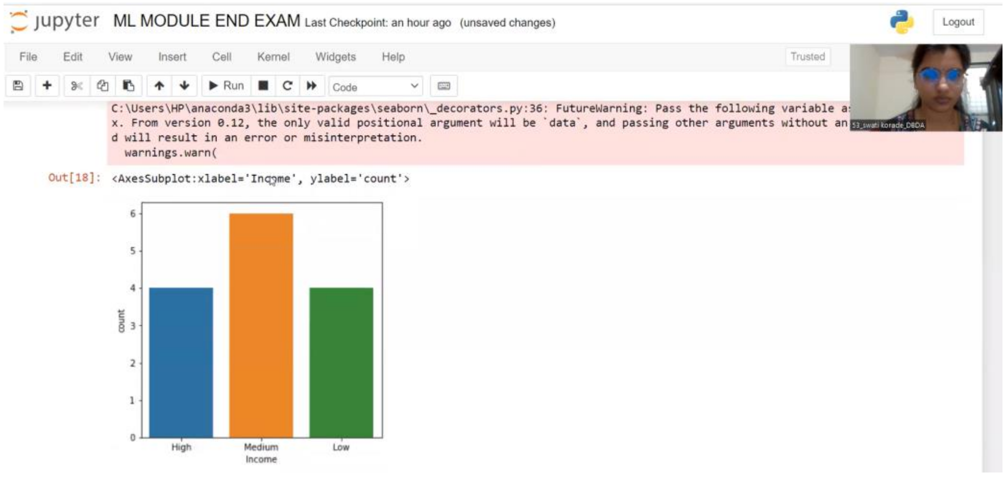
```
In [11]: data['Income'].value_counts()
Out[11]: Medium    6
         High      4
         Low       4
         Name: Income, dtype: int64

In [12]: data['Age'].value_counts()
Out[12]: <21      5
         >35      5
         21-35    4
         Name: Age, dtype: int64

In [13]: data['Gender'].value_counts()
Out[13]: Male      7
         Female    7
         Name: Gender, dtype: int64

In [ ]: data['Mari'].value_counts()
```

Target Variable

```
: data['Buys'].value_counts()                        # buys is target binary variable
: Yes    9
  No     5
  Name: Buys, dtype: int64
```

Countplot for each categorical variable

```
: plt.figure(figsize=(5,5))
  sns.countplot(data['Income'])
  C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
  x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keywor
  d will result in an error or misinterpretation.
    warnings.warn(
: <AxesSubplot:xlabel='Income', ylabel='count'>
```



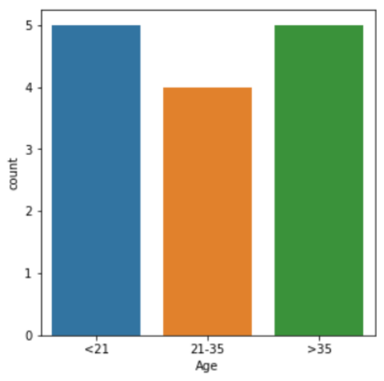
```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable a
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an
d will result in an error or misinterpretation.
  warnings.warn(
Out[18]: <AxesSubplot:xlabel='Income', ylabel='count'>
```

Conclusion : most of the people having medium income.

```
plt.figure(figsize=(5,5))
sns.countplot(data['Age'])
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```

```
<AxesSubplot:xlabel='Age', ylabel='count'>
```

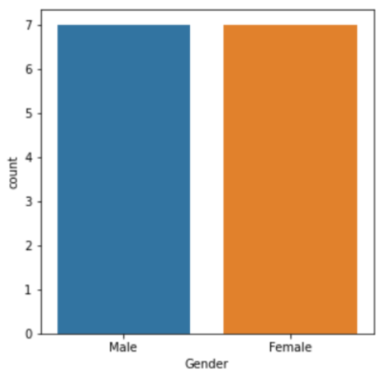


Conclusion: There are most of the peoplea are having age less than 21 and greater than 35.

```
: plt.figure(figsize=(5,5))
  sns.countplot(data['Gender'])
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```

```
: <AxesSubplot:xlabel='Gender', ylabel='count'>
```



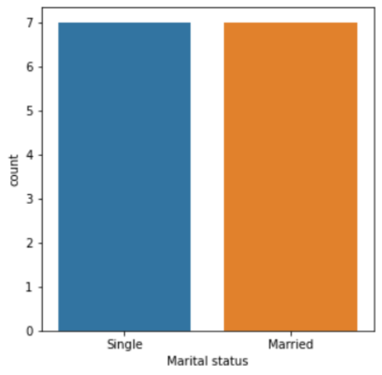
Conclusion: In the data set equal no of male and female.

200340325053

```
plt.figure(figsize=(5,5))
sns.countplot(data['Marital status'])
```

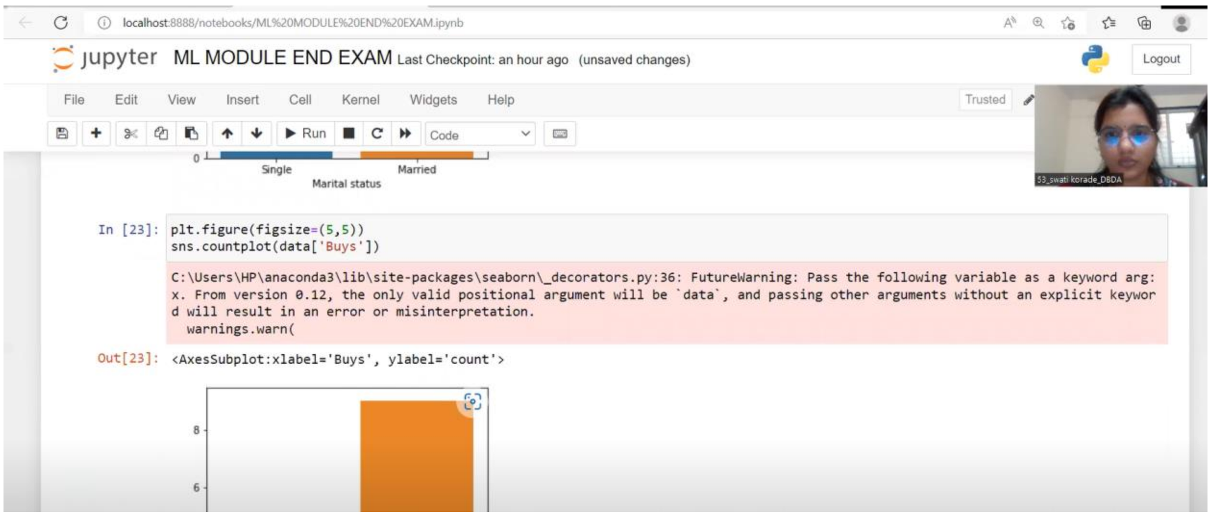
```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```

```
<AxesSubplot:xlabel='Marital status', ylabel='count'>
```



Conclusion: same no of married and unmarried peoples.

# checking Imbalance data


localhost:8888/notebooks/ML%20MODULE%20END%20EXAM.ipynb
jupyter ML MODULE END EXAM Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Run
Code
0
Single
Married
Marital status
53_swati korade_DBDA
8
6

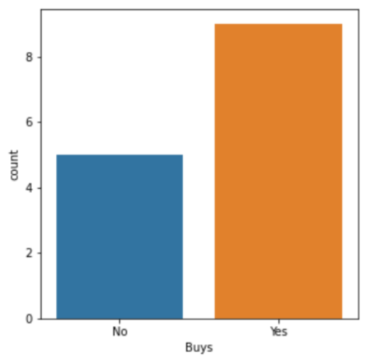
```
In [23]: plt.figure(figsize=(5,5))
         sns.countplot(data['Buys'])
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```

```
Out[23]: <AxesSubplot:xlabel='Buys', ylabel='count'>
```

checking of Balance data

```
n [8]: plt.figure(figsize=(5,5))
       sns.countplot(data['Buys'])

       C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
       x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keywor
       d will result in an error or misinterpretation.
         warnings.warn(

ut[8]: <AxesSubplot:xlabel='Buys', ylabel='count'>
```

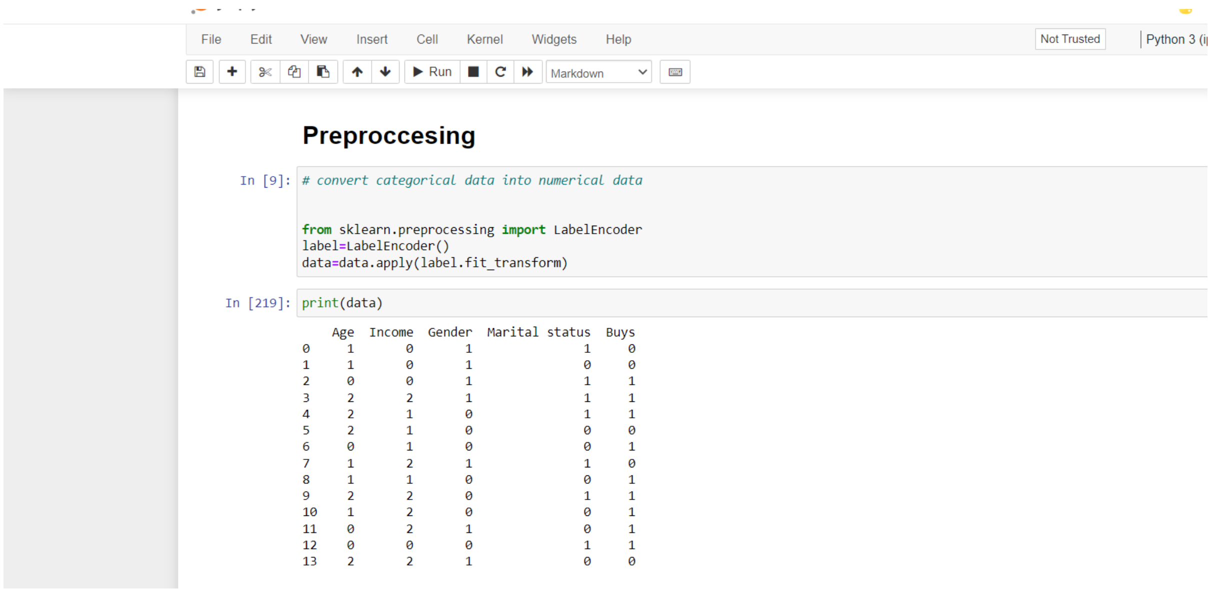


Here we can say that count of buys cosmetics is greater than not buying.

here buys is our target variable and praprotion of both is not balance it may be possible that data is imbalance in some extend 64 % data in yes category and remaining in no category.

PREPROCESSING:

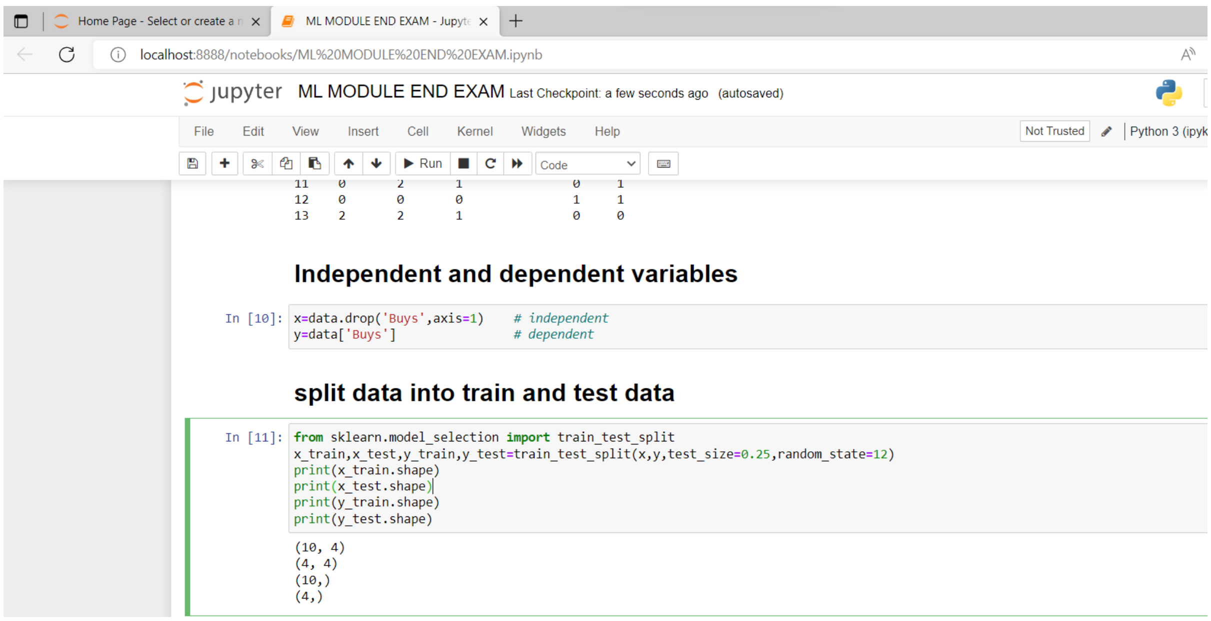
Label the data



Preproccesing

```
In [9]: # convert categorical data into numerical data

        from sklearn.preprocessing import LabelEncoder
        label=LabelEncoder()
        data=data.apply(label.fit_transform)

In [219]: print(data)

    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
10    1       2       0               0     1
11    0       2       1               0     1
12    0       0       0               1     1
13    2       2       1               0     0
```

200340325053


Home Page - Select or create a
ML MODULE END EXAM - Jupyt
localhost:8888/notebooks/ML%20MODULE%20END%20EXAM.ipynb
jupyter ML MODULE END EXAM Last Checkpoint: a few seconds ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipyk
Run
Code

```
11   0     2     1           0    1
12   0     0     0           1    1
13   2     2     1           0    0
```

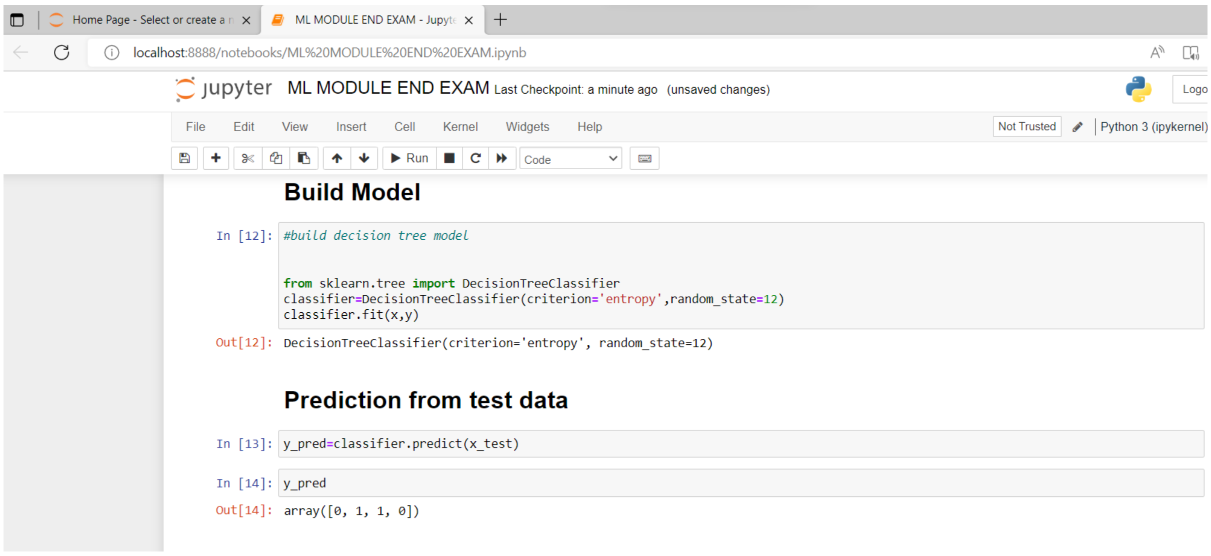
Independent and dependent variables

In [10]:
```
x=data.drop('Buys',axis=1)     # independent
y=data['Buys']                 # dependent
```

split data into train and test data

In [11]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=12)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(10, 4)
(4, 4)
(10,)
(4,)
```


Home Page - Select or create a
ML MODULE END EXAM - Jupyt
localhost:8888/notebooks/ML%20MODULE%20END%20EXAM.ipynb
jupyter ML MODULE END EXAM Last Checkpoint: a minute ago (unsaved changes)
Logo
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

Build Model

In [12]:
```
#build decision tree model


from sklearn.tree import DecisionTreeClassifier
classifier=DecisionTreeClassifier(criterion='entropy',random_state=12)
classifier.fit(x,y)
```

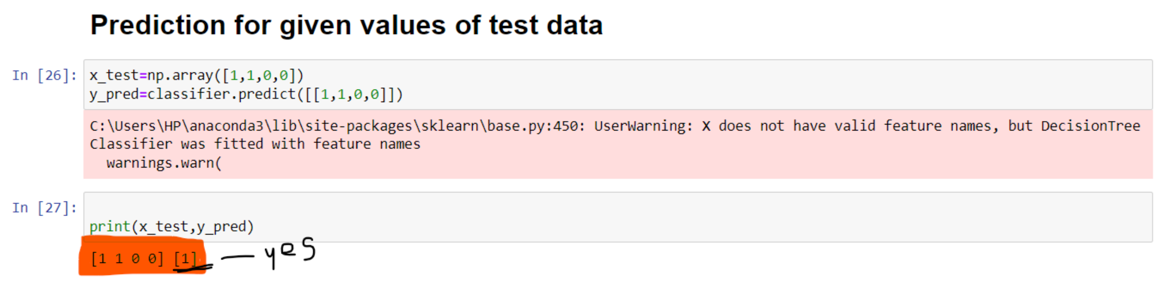
Out[12]:
```
DecisionTreeClassifier(criterion='entropy', random_state=12)
```

Prediction from test data

In [13]:
```
y_pred=classifier.predict(x_test)
```

In [14]:
```
y_pred
```

Out[14]:
```
array([0, 1, 1, 0])
```



Prediction for given values of test data

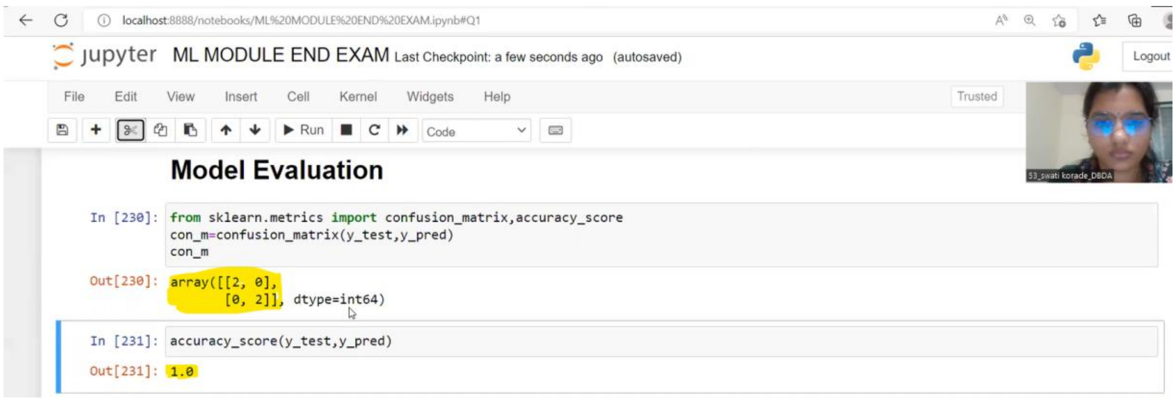
In [26]:
```
x_test=np.array([1,1,0,0])
y_pred=classifier.predict([[1,1,0,0]])
```

```
C:\Users\HP\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but DecisionTree
Classifier was fitted with feature names
  warnings.warn(
```

In [27]:
```
print(x_test,y_pred)
```

```
[1 1 0 0] [1]
```
— yes

for given data [Age < 21, Income = Low, Gender = Female, Marital Status = Married] prection is buying cosmetic

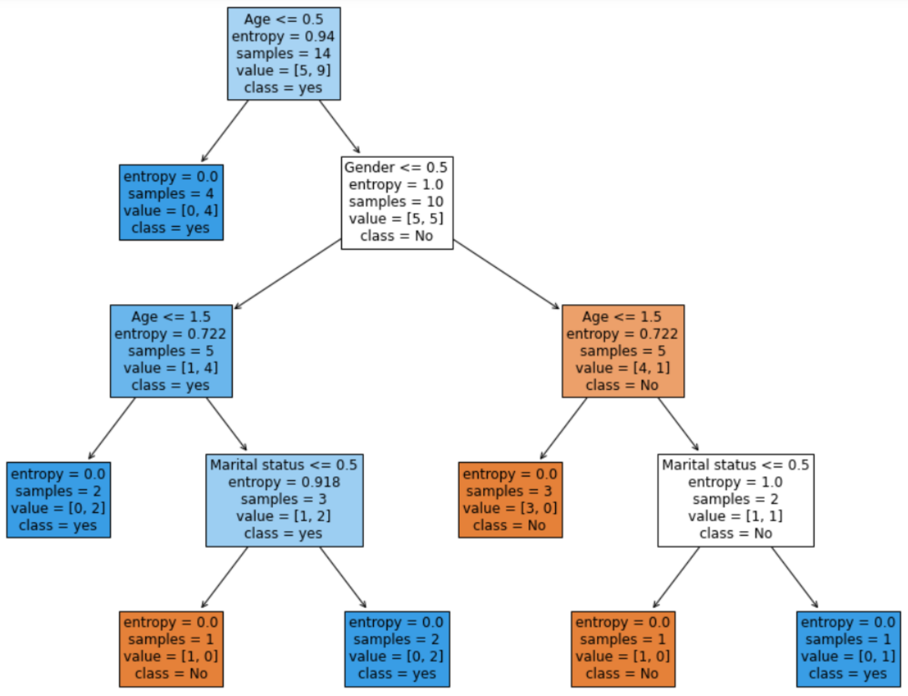


localhost:8888/notebooks/ML%20MODULE%20END%20EXAM.ipynb#Q1
jupyter ML MODULE END EXAM Last Checkpoint: a few seconds ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Run
Code
53_swati korade_DBDA

Model Evaluation

```
In [230]: from sklearn.metrics import confusion_matrix,accuracy_score
          con_m=confusion_matrix(y_test,y_pred)
          con_m

Out[230]: array([[2, 0],
                 [0, 2]], dtype=int64)

In [231]: accuracy_score(y_test,y_pred)

Out[231]: 1.0
```

```
In [242]: from sklearn.tree import plot_tree

          import matplotlib.pyplot as plt
          fig = plt.figure(figsize=(16,12))
          a = plot_tree(classifier, feature_names=x.columns, fontsize=12, filled=True,
                        class_names=['No', 'yes'])
```

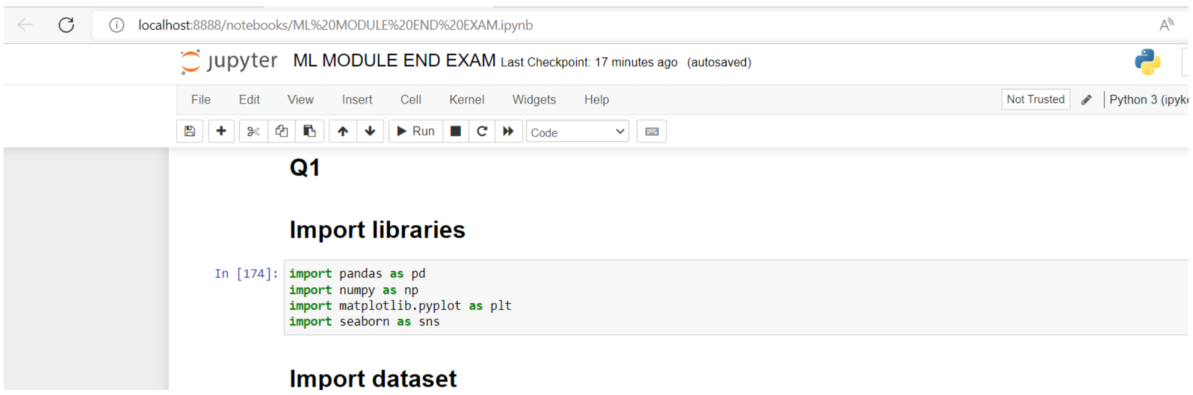


Overall conclusion:

```
from the conclusion matrix and accuracy score we can say that th
ere all obseravtions are correctly predicted. But there might be
possibility of overfitting due to some imbalace data
```

200340325053

Q.1



localhost:8888/notebooks/ML%20MODULE%20END%20EXAM.ipynb

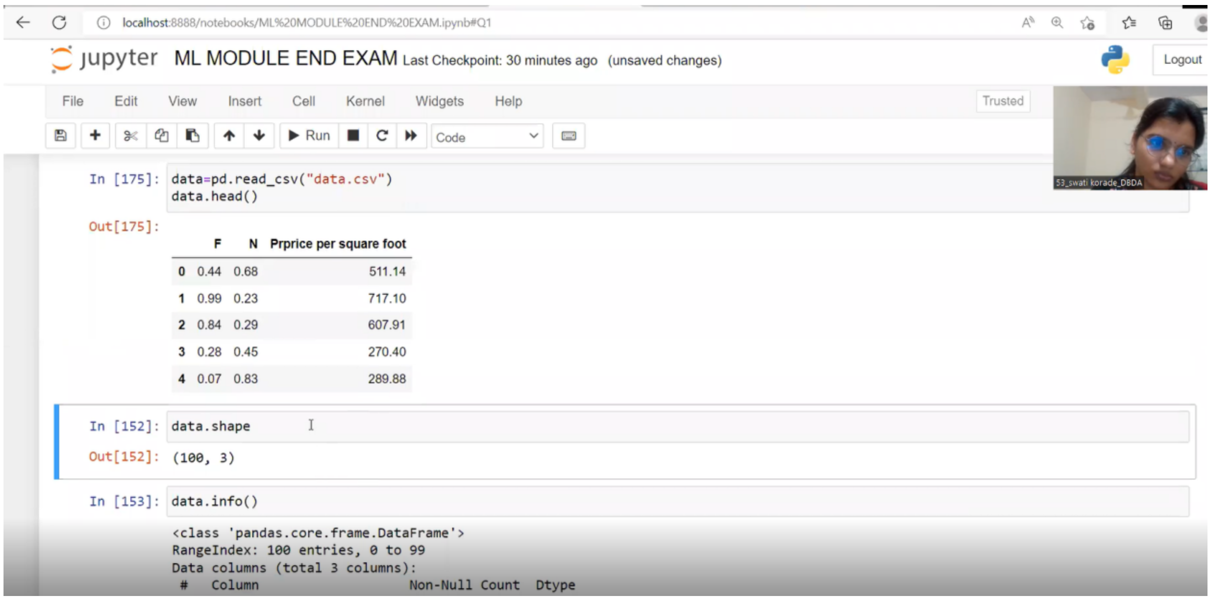
jupyter
ML MODULE END EXAM Last Checkpoint: 17 minutes ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipyk
Run
Code

Q1

Import libraries

```
In [174]: import pandas as pd
          import numpy as np
          import matplotlib.pyplot as plt
          import seaborn as sns
```

Import dataset



localhost:8888/notebooks/ML%20MODULE%20END%20EXAM.ipynb#Q1

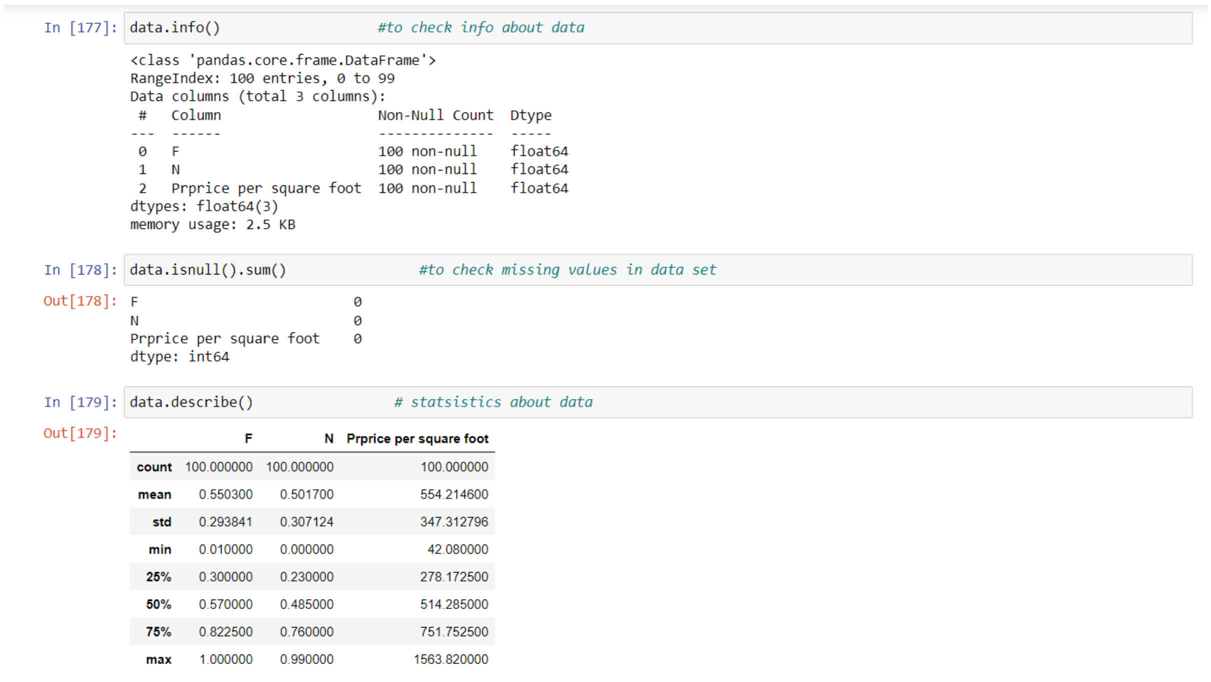
jupyter
ML MODULE END EXAM Last Checkpoint: 30 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Run
Code
53_swati korade_DBDA

```
In [175]: data=pd.read_csv("data.csv")
          data.head()
Out[175]:
```

	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

```
In [152]: data.shape
Out[152]: (100, 3)

In [153]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 3 columns):
 #   Column                   Non-Null Count  Dtype
```

200340325053



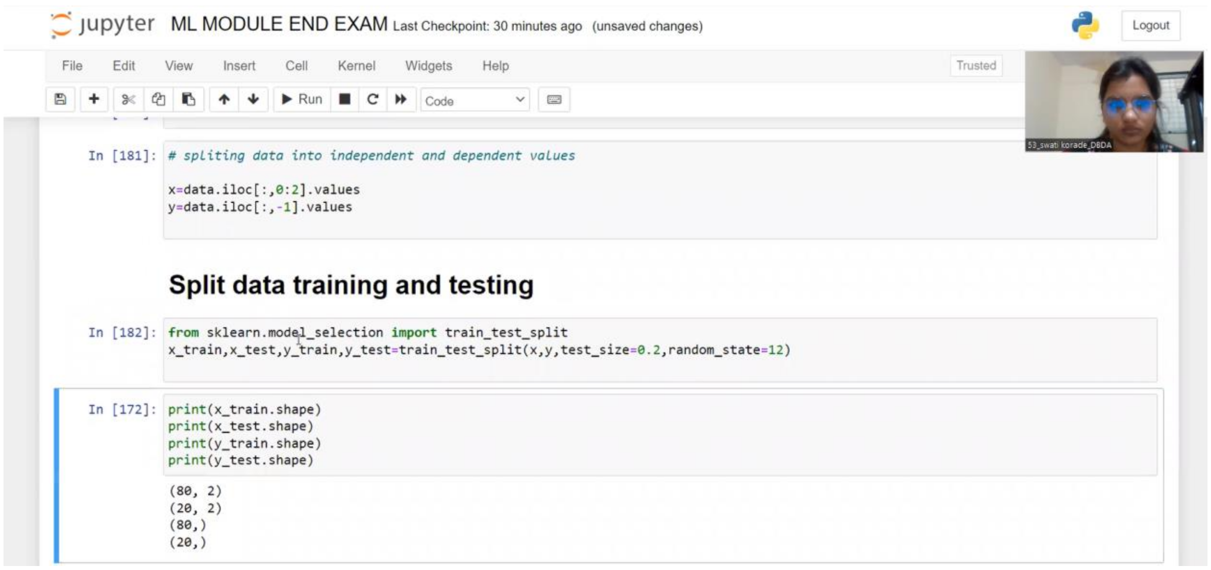
```
In [177]: data.info()                    #to check info about data

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 3 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   F                       100 non-null    float64
 1   N                       100 non-null    float64
 2   Prprice per square foot 100 non-null    float64
dtypes: float64(3)
memory usage: 2.5 KB
```

```
In [178]: data.isnull().sum()                 #to check missing values in data set

Out[178]: F                          0
N                          0
Prprice per square foot    0
dtype: int64
```

```
In [179]: data.describe()                  # statsistics about data
```

Out[179]:

	F	N	Prprice per square foot
count	100.000000	100.000000	100.000000
mean	0.550300	0.501700	554.214600
std	0.293841	0.307124	347.312796
min	0.010000	0.000000	42.080000
25%	0.300000	0.230000	278.172500
50%	0.570000	0.485000	514.285000
75%	0.822500	0.760000	751.752500
max	1.000000	0.990000	1563.820000

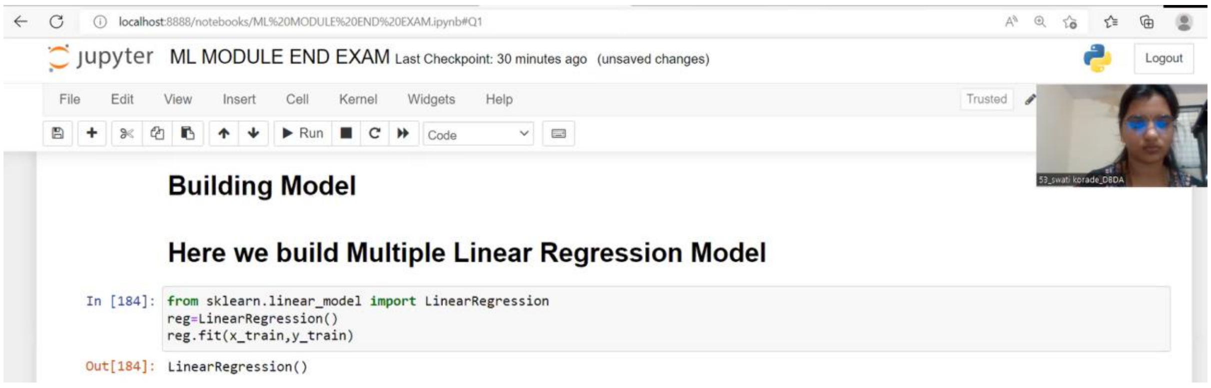

jupyter ML MODULE END EXAM Last Checkpoint: 30 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Run
Code
53_swati korade_DBDA

```
In [181]: # spliting data into independent and dependent values

          x=data.iloc[:,0:2].values
          y=data.iloc[:,-1].values
```

Split data training and testing

```
In [182]: 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=12)
```

```
In [172]: print(x_train.shape)
          print(x_test.shape)
          print(y_train.shape)
          print(y_test.shape)

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

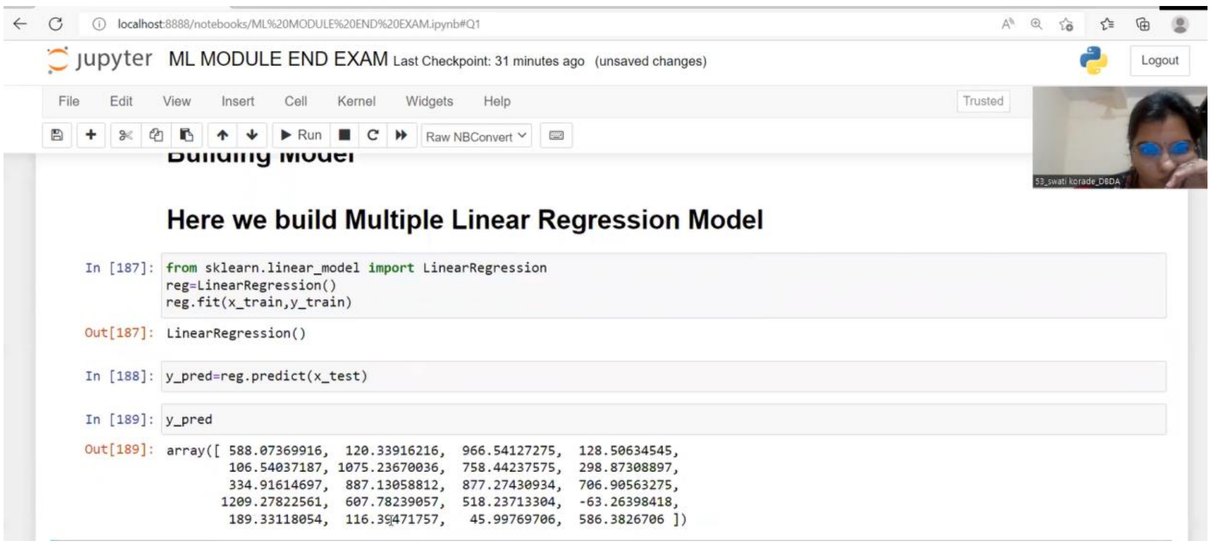

localhost:8888/notebooks/ML%20MODULE%20END%20EXAM.ipynb#Q1
jupyter ML MODULE END EXAM Last Checkpoint: 30 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Run
Code
53_swati korade_DBDA

Building Model

Here we build Multiple Linear Regression Model

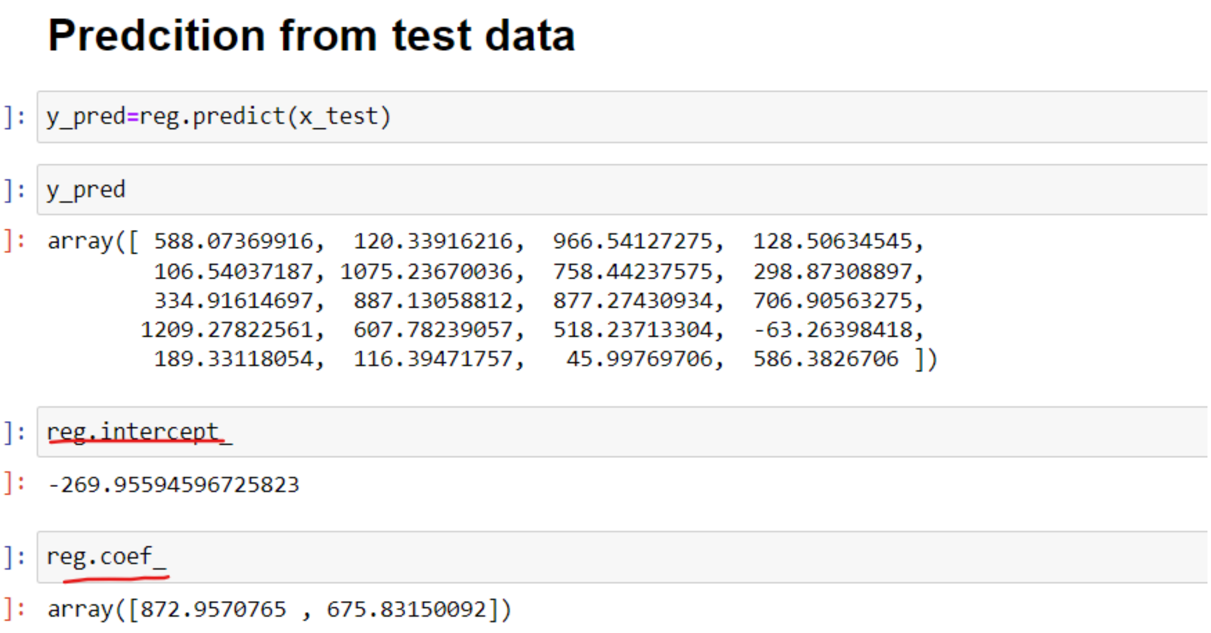
```
In [184]: from sklearn.linear_model import LinearRegression
          reg=LinearRegression()
          reg.fit(x_train,y_train)

Out[184]: LinearRegression()
```

200340325053


localhost:8888/notebooks/ML%20MODULE%20END%20EXAM.ipynb#Q1
jupyter ML MODULE END EXAM Last Checkpoint: 31 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Run
Raw NBConvert
53_swati korade_DBDA

Here we build Multiple Linear Regression Model

```
In [187]: from sklearn.linear_model import LinearRegression
          reg=LinearRegression()
          reg.fit(x_train,y_train)

Out[187]: LinearRegression()

In [188]: y_pred=reg.predict(x_test)

In [189]: y_pred

Out[189]: array([ 588.07369916,  120.33916216,  966.54127275,  128.50634545,
                  106.54037187, 1075.23670036,  758.44237575,  298.87308897,
                  334.91614697,  887.13058812,  877.27430934,  706.90563275,
                 1209.27822561,  607.78239057,  518.23713304,  -63.26398418,
                  189.33118054,  116.39471757,   45.99769706,  586.3826706 ])
```



Predcition from test data

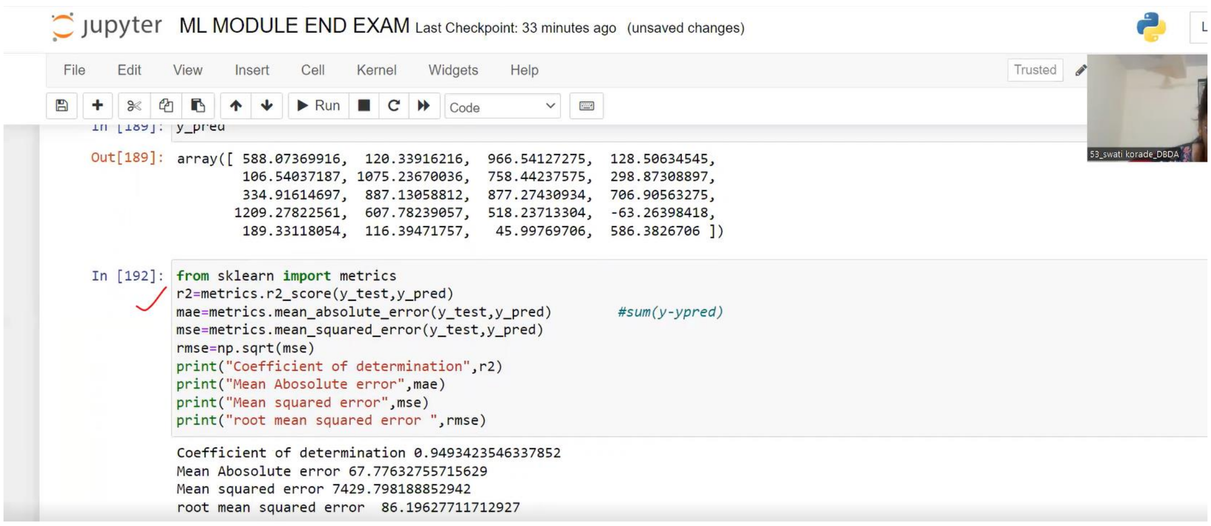
```
]: y_pred=reg.predict(x_test)

]: y_pred

]: array([ 588.07369916,  120.33916216,  966.54127275,  128.50634545,
            106.54037187, 1075.23670036,  758.44237575,  298.87308897,
            334.91614697,  887.13058812,  877.27430934,  706.90563275,
           1209.27822561,  607.78239057,  518.23713304,  -63.26398418,
            189.33118054,  116.39471757,   45.99769706,  586.3826706 ])

]: reg.intercept_

]: -269.95594596725823

]: reg.coef_

]: array([872.9570765 , 675.83150092])
```

Multiple linear equation is :

Perpreice square foot: y

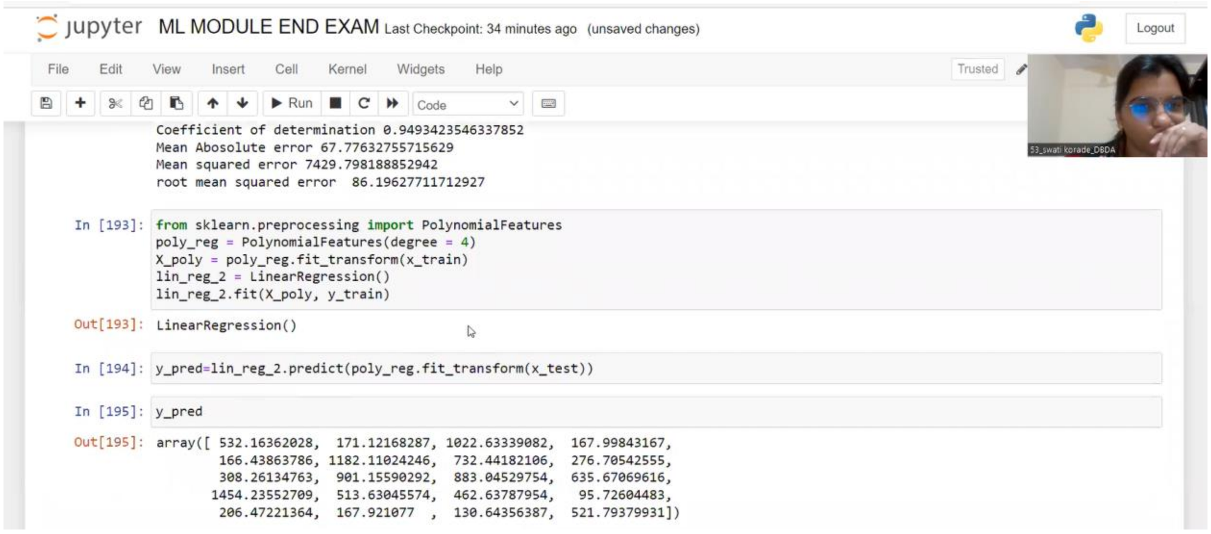
Y= -269.9559+(872.9570*F)+(675.8315*N)

200340325053

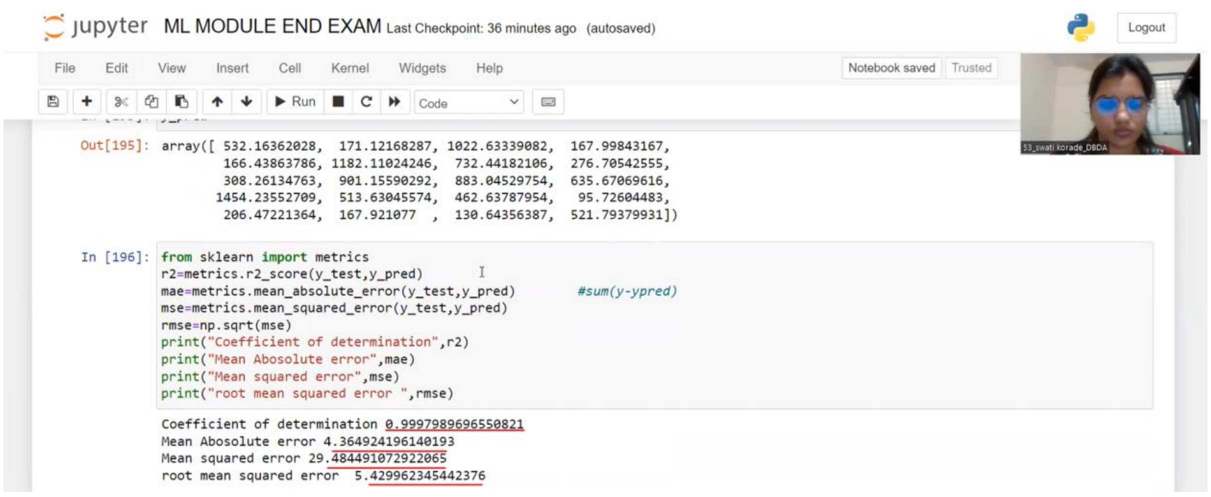

jupyter ML MODULE END EXAM Last Checkpoint: 33 minutes ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Run
Code
53_swati korade_DBDA

```
In [189]: y_pred

Out[189]: array([ 588.07369916,  120.33916216,  966.54127275,  128.50634545,
                  106.54037187, 1075.23670036,  758.44237575,  298.87308897,
                  334.91614697,  887.13058812,  877.27430934,  706.90563275,
                 1209.27822561,  607.78239057,  518.23713304,  -63.26398418,
                  189.33118054,  116.39471757,   45.99769706,  586.3826706 ])

In [192]: from sklearn import metrics
          r2=metrics.r2_score(y_test,y_pred)
          mae=metrics.mean_absolute_error(y_test,y_pred)        #sum(y-ypred)
          mse=metrics.mean_squared_error(y_test,y_pred)
          rmse=np.sqrt(mse)
          print("Coefficient of determination",r2)
          print("Mean Abosolute error",mae)
          print("Mean squared error",mse)
          print("root mean squared error ",rmse)

          Coefficient of determination 0.9493423546337852
          Mean Abosolute error 67.77632755715629
          Mean squared error 7429.798188852942
          root mean squared error  86.19627711712927
```

If we see r2 is good but error is larger so we try to fit another model polynomial regression for better accuracy and minimize the error


jupyter ML MODULE END EXAM Last Checkpoint: 34 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Run
Code
53_swati korade_DBDA

```
          Coefficient of determination 0.9493423546337852
          Mean Abosolute error 67.77632755715629
          Mean squared error 7429.798188852942
          root mean squared error  86.19627711712927

In [193]: from sklearn.preprocessing import PolynomialFeatures
          poly_reg = PolynomialFeatures(degree = 4)
          X_poly = poly_reg.fit_transform(x_train)
          lin_reg_2 = LinearRegression()
          lin_reg_2.fit(X_poly, y_train)

Out[193]: LinearRegression()

In [194]: y_pred=lin_reg_2.predict(poly_reg.fit_transform(x_test))

In [195]: y_pred

Out[195]: array([ 532.16362028,  171.12168287, 1022.63339082,  167.99843167,
                  166.43863786, 1182.11024246,  732.44182106,  276.70542555,
                  308.26134763,  901.15590292,  883.04529754,  635.67069616,
                 1454.23552709,  513.63045574,  462.63787954,   95.72604483,
                  206.47221364,  167.921077  ,  130.64356387,  521.79379931])
```


jupyter ML MODULE END EXAM Last Checkpoint: 36 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Notebook saved Trusted
Run
Code
53_swati korade_DBDA

```
Out[195]: array([ 532.16362028,  171.12168287, 1022.63339082,  167.99843167,
                  166.43863786, 1182.11024246,  732.44182106,  276.70542555,
                  308.26134763,  901.15590292,  883.04529754,  635.67069616,
                 1454.23552709,  513.63045574,  462.63787954,   95.72604483,
                  206.47221364,  167.921077  ,  130.64356387,  521.79379931])

In [196]: from sklearn import metrics
          r2=metrics.r2_score(y_test,y_pred)
          mae=metrics.mean_absolute_error(y_test,y_pred)        #sum(y-ypred)
          mse=metrics.mean_squared_error(y_test,y_pred)
          rmse=np.sqrt(mse)
          print("Coefficient of determination",r2)
          print("Mean Abosolute error",mae)
          print("Mean squared error",mse)
          print("root mean squared error ",rmse)

          Coefficient of determination 0.9997989696550821
          Mean Abosolute error 4.364924196140193
          Mean squared error 29.484491072922065
          root mean squared error  5.429962345442376
```

Overall conclusion :

```
if we compare both the data set then accuracy for polynomial reg
ression is better than multiple linear regression aslo mean abso
lute error also reduce much in case of polynomial resgression



so from the model we can suggest that anil has to choose polynom
ial regression for his prediction
```