In [1]:
```
!pip install sklearn
```

```
  (requirements.txt, setup.py, setup.cfg, Pipfile, etc ...)
  - if the 'sklearn' package is used by one of your dependencies,
    it would be great if you take some time to track which package uses
    'sklearn' instead of 'scikit-learn' and report it to their issue track
er
  - as a last resort, set the environment variable
    SKLEARN_ALLOW_DEPRECATED_SKLEARN_PACKAGE_INSTALL=True to avoid this er
ror

  More information is available at
  https://github.com/scikit-learn/sklearn-pypi-package (https://github.co
m/scikit-learn/sklearn-pypi-package)
  [end of output]

  note: This error originates from a subprocess, and is likely not a probl
em with pip.
error: metadata-generation-failed

Encountered error while generating package metadata.
```

In [2]:
```
import sklearn
print(sklearn.__version__)
```

```
1.4.2
```

In [3]:
```
pip install --upgrade pip
```

```
Requirement already satisfied: pip in c:\users\jyoth\anaconda3\lib\site-packa
ges (23.3.1)
Collecting pip
  Using cached pip-24.0-py3-none-any.whl.metadata (3.6 kB)
Using cached pip-24.0-py3-none-any.whl (2.1 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 23.3.1
    Uninstalling pip-23.3.1:
      Successfully uninstalled pip-23.3.1
Successfully installed pip-24.0
Note: you may need to restart the kernel to use updated packages.
```

In [4]:
```
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\jyoth\anaconda3\lib\s
ite-packages (1.4.2)
Requirement already satisfied: numpy>=1.19.5 in c:\users\jyoth\anaconda3\lib
\site-packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in c:\users\jyoth\anaconda3\lib\s
ite-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.2.0 in c:\users\jyoth\anaconda3\lib
\site-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\jyoth\anacond
a3\lib\site-packages (from scikit-learn) (2.2.0)
Note: you may need to restart the kernel to use updated packages.
```

In [5]:
```
pip install pandas
```

```
Requirement already satisfied: pandas in c:\users\jyoth\anaconda3\lib\site-pa
ckages (2.1.4)
Requirement already satisfied: numpy<2,>=1.23.2 in c:\users\jyoth\anaconda3\l
ib\site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\jyoth\anaco
nda3\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\jyoth\anaconda3\lib\s
ite-packages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\jyoth\anaconda3\lib
\site-packages (from pandas) (2023.3)
Requirement already satisfied: six>=1.5 in c:\users\jyoth\anaconda3\lib\site-
packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [6]:
```
import pandas as pd

# Load the dataset
df = pd.read_csv('honeyproduction.csv')  # Replace 'your_dataset.csv' with you
```

In [7]:
```
# Display the first 5 rows of the dataset
df.head()
```

Out[7]:

	state	numcol	yieldpercol	totalprod	stocks	priceperlb	prodvalue	year
0	AL	16000.0	71	1136000.0	159000.0	0.72	818000.0	1998
1	AZ	55000.0	60	3300000.0	1485000.0	0.64	2112000.0	1998
2	AR	53000.0	65	3445000.0	1688000.0	0.59	2033000.0	1998
3	CA	450000.0	83	37350000.0	12326000.0	0.62	23157000.0	1998
4	CO	27000.0	72	1944000.0	1594000.0	0.70	1361000.0	1998

In [8]:
```
# Display basic information about the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 626 entries, 0 to 625
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   state        626 non-null    object 
 1   numcol       626 non-null    float64
 2   yieldpercol  626 non-null    int64  
 3   totalprod    626 non-null    float64
 4   stocks       626 non-null    float64
 5   priceperlb   626 non-null    float64
 6   prodvalue    626 non-null    float64
 7   year         626 non-null    int64  
dtypes: float64(5), int64(2), object(1)
memory usage: 39.3+ KB
```

In [20]:
```
# insert the cell
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,r2_score
```

In [13]:
```
# Load the dataset
df = pd.read_csv('honeyproduction.csv')

# Display the first few rows of the dataset
print(df.head())
```

```
  state    numcol  yieldpercol   totalprod      stocks  priceperlb  \
0    AL   16000.0           71   1136000.0    159000.0        0.72   
1    AZ   55000.0           60   3300000.0   1485000.0        0.64   
2    AR   53000.0           65   3445000.0   1688000.0        0.59   
3    CA  450000.0           83  37350000.0  12326000.0        0.62   
4    CO   27000.0           72   1944000.0   1594000.0        0.70   

    prodvalue  year  
0    818000.0  1998  
1   2112000.0  1998  
2   2033000.0  1998  
3  23157000.0  1998  
4   1361000.0  1998  
```

In [29]:
```
df=df.drop(['state'],axis=1)
df
```

Out[29]:

	numcol	yieldpercol	totalprod	stocks	priceperlb	prodvalue	year
0	16000.0	71	1136000.0	159000.0	0.72	818000.0	1998
1	55000.0	60	3300000.0	1485000.0	0.64	2112000.0	1998
2	53000.0	65	3445000.0	1688000.0	0.59	2033000.0	1998
3	450000.0	83	37350000.0	12326000.0	0.62	23157000.0	1998
4	27000.0	72	1944000.0	1594000.0	0.70	1361000.0	1998
...	...	...	...	...	...	...	...
621	4000.0	41	164000.0	23000.0	3.77	618000.0	2012
622	62000.0	41	2542000.0	1017000.0	2.38	6050000.0	2012
623	6000.0	48	288000.0	95000.0	2.91	838000.0	2012
624	60000.0	69	4140000.0	1863000.0	2.05	8487000.0	2012
625	50000.0	51	2550000.0	459000.0	1.87	4769000.0	2012

626 rows × 7 columns

In [31]:
```
#check the null value
df.isnull().sum()
```

Out[31]:
```
numcol         0
yieldpercol    0
totalprod      0
stocks         0
priceperlb     0
prodvalue      0
year           0
dtype: int64
```

In [33]:
```
#define features and target variable
X=df.drop(['totalprod'],axis=1)
y=df['totalprod']
```

In [34]:
```
#split the data into test & train sets
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_stat
```

In [35]:
```
scaler=StandardScaler()
X_train_scaled=scaler.fit_transform(X_train)
X_test_scaled=scaler.transform(X_test)
```

In [36]:
```
#train model

model=LinearRegression()
model.fit(X_train_scaled,y_train)
```

Out[36]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [37]:
```
#evaluate the model
y_predict=model.predict(X_test_scaled)
```

In [38]:
```
mse=mean_squared_error(y_test,y_predict)
r2=r2_score(y_test,y_predict)
print(mse)
print(r2)
```

```
2291037707861.2847
0.9371642699573703
```