Importing the Libraries

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

Import Dataset ¶

In [2]:
```
dataset=pd.read_csv('C:/Users/navna/Music/datasets/Salary_Data.csv')
dataset
```

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0
5	2.9	56642.0
6	3.0	60150.0
7	3.2	54445.0
8	3.2	64445.0
9	3.7	57189.0
10	3.9	63218.0
11	4.0	55794.0
12	4.0	56957.0
13	4.1	57081.0
14	4.5	61111.0
15	4.9	67938.0
16	5.1	66029.0
17	5.3	83088.0
18	5.9	81363.0
19	6.0	93940.0
20	6.8	91738.0
21	7.1	98273.0
22	7.9	101302.0
23	8.2	113812.0
24	8.7	109431.0

	YearsExperience	Salary
25	9.0	105582.0
26	9.5	116969.0
27	9.6	112635.0
28	10.3	122391.0
29	10.5	121872.0

In [3]:
```
dataset.shape
```

```
(30, 2)
```

In [4]:
```
#Independent Variable
X=dataset.iloc[:,:-1].values

#Dependent Variable
y=dataset.iloc[:,-1].values
```

In [5]:
```
X
```

```
array([[ 1.1],
       [ 1.3],
       [ 1.5],
       [ 2. ],
       [ 2.2],
       [ 2.9],
       [ 3. ],
       [ 3.2],
       [ 3.2],
       [ 3.7],
       [ 3.9],
       [ 4. ],
       [ 4. ],
       [ 4.1],
       [ 4.5],
       [ 4.9],
       [ 5.1],
       [ 5.3],
       [ 5.9],
       [ 6. ],
       [ 6.8],
       [ 7.1],
       [ 7.9],
       [ 8.2],
       [ 8.7],
       [ 9. ],
       [ 9.5],
       [ 9.6],
       [10.3],
       [10.5]])
```

In [6]:
```
y
```

```
array([ 39343.,  46205.,  37731.,  43525.,  39891.,  56642.,  60150.,
        54445.,  64445.,  57189.,  63218.,  55794.,  56957.,  57081.,
        61111.,  67938.,  66029.,  83088.,  81363.,  93940.,  91738.,
        98273., 101302., 113812., 109431., 105582., 116969., 112635.,
       122391., 121872.])
```

Splitting the dataset (Training and Testing)

In [7]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train,y_test =train_test_split(X,y,test_size=1/3, random_state=0)
```

In [8]:
```
X_train
```

```
array([[ 2.9],
       [ 5.1],
       [ 3.2],
       [ 4.5],
       [ 8.2],
       [ 6.8],
       [ 1.3],
       [10.5],
       [ 3. ],
       [ 2.2],
       [ 5.9],
       [ 6. ],
       [ 3.7],
       [ 3.2],
       [ 9. ],
       [ 2. ],
       [ 1.1],
       [ 7.1],
       [ 4.9],
       [ 4. ]])
```

Training the Simple Regression Model

In [9]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(X_train, y_train)
```

```
LinearRegression()
```

Prediction of Testing dataset

In [10]:
```
y_pred=reg.predict(X_test)
```

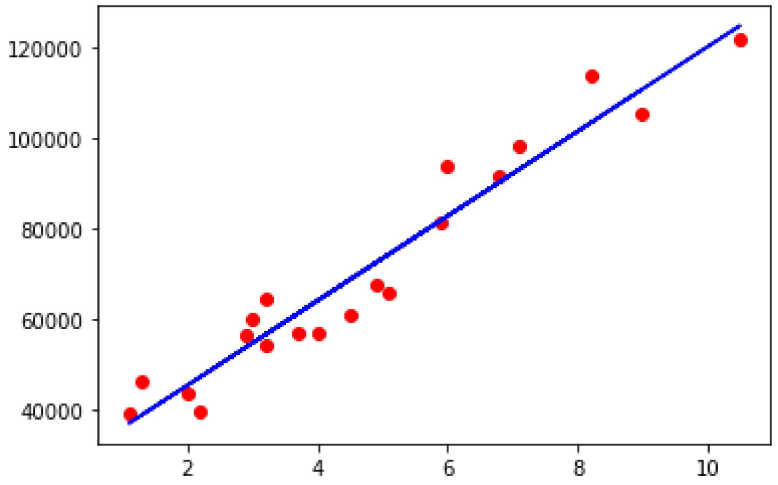
In [11]:
```
y_pred
```

```
array([ 40835.10590871, 123079.39940819,  65134.55626083,  63265.36777221,
       115602.64545369, 108125.8914992 , 116537.23969801,  64199.96201652,
        76349.68719258, 100649.1375447 ])
```

Visualising the results

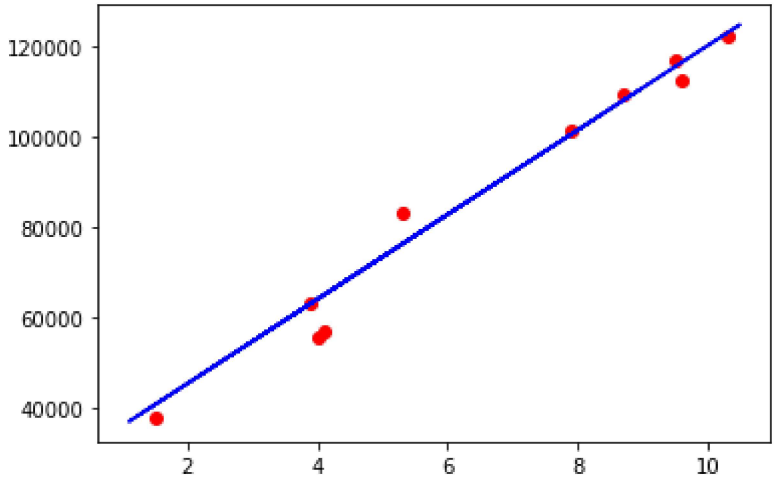
In [12]:
```
plt.scatter(X_train,y_train,color='red')
plt.plot(X_train, reg.predict(X_train), color='blue')
```

```
[<matplotlib.lines.Line2D at 0x1cb4c8b7790>]
```



In [13]:
```
plt.scatter(X_test,y_test,color='red')
plt.plot(X_train, reg.predict(X_train), color='blue')
```

```
[<matplotlib.lines.Line2D at 0x1cb4c9b7640>]
```



In [14]:
```
#Coefficient
b=reg.coef_
```

In [15]:
```

b
```

```
array([9345.94244312])
```

In [16]:
```
#Intercept
a=reg.intercept_
```

In [17]:
```
a
```

```
26816.19224403119
```

In [18]:
```
reg.predict([[13]])
```

```
array([148313.44400462])
```

In [19]:
```
#MSE
from sklearn import metrics
```

In [20]:
```
metrics.mean_squared_error(y_test,y_pred)
```

```
21026037.329511296
```

In [21]:
```
import statsmodels.api as sm
```

In [22]:
```
X_stat = sm.add_constant(X_train)
Summ=sm.OLS(y_train,X_stat).fit()
```

In [23]:

```
Summ.summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.938
Model:	OLS	**Adj. R-squared:**	0.935
Method:	Least Squares	**F-statistic:**	273.2
Date:	Sat, 06 Aug 2022	**Prob (F-statistic):**	2.51e-12
Time:	16:57:08	**Log-Likelihood:**	-202.60
No. Observations:	20	**AIC:**	409.2
Df Residuals:	18	**BIC:**	411.2
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	2.682e+04	3033.148	8.841	0.000	2.04e+04	3.32e+04
x1	9345.9424	565.420	16.529	0.000	8158.040	1.05e+04

Omnibus:	2.688	**Durbin-Watson:**	2.684
Prob(Omnibus):	0.261	**Jarque-Bera (JB):**	1.386
Skew:	0.305	**Prob(JB):**	0.500
Kurtosis:	1.864	**Cond. No.**	11.7

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [24]:
```
Summ.rsquared_adj
```

```
0.9347561124721737
```

In [25]:
```
Summ.rsquared
```

```
0.9381900012894278
```

Multiple Linear Regression.

In [26]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [27]:
```
dataset = pd.read_csv('C:/Users/navna/Music/datasets/50_Startups.csv')
```

In [28]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   R&D Spend        50 non-null     float64
 1   Administration   50 non-null     float64
 2   Marketing Spend  50 non-null     float64
 3   State            50 non-null     object 
 4   Profit           50 non-null     float64
dtypes: float64(4), object(1)
memory usage: 2.1+ KB
```

In [29]:
```
dataset.shape
```

```
(50, 5)
```

In [30]:
```
dataset.head()
```

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

In [31]:
```
X=dataset.iloc[:,:-1].values
y=dataset.iloc[:,-1].values
```

In [32]:
```
print(X)
```

```
[[165349.2 136897.8 471784.1 'New York']
 [162597.7 151377.59 443898.53 'California']
 [153441.51 101145.55 407934.54 'Florida']
 [144372.41 118671.85 383199.62 'New York']
 [142107.34 91391.77 366168.42 'Florida']
 [131876.9 99814.71 362861.36 'New York']
 [134615.46 147198.87 127716.82 'California']
 [130298.13 145530.06 323876.68 'Florida']
 [120542.52 148718.95 311613.29 'New York']
 [123334.88 108679.17 304981.62 'California']
 [101913.08 110594.11 229160.95 'Florida']
 [100671.96 91790.61 249744.55 'California']
 [93863.75 127320.38 249839.44 'Florida']
 [91992.39 135495.07 252664.93 'California']
 [119943.24 156547.42 256512.92 'Florida']
 [114523.61 122616.84 261776.23 'New York']
 [78013.11 121597.55 264346.06 'California']
 [94657.16 145077.58 282574.31 'New York']
 [91749.16 114175.79 294919.57 'Florida']
 [86419.7 153514.11 0.0 'New York']
 [76253.86 113867.3 298664.47 'California']
 [78389.47 153773.43 299737.29 'New York']
 [73994.56 122782.75 303319.26 'Florida']
 [67532.53 105751.03 304768.73 'Florida']
 [77044.01 99281.34 140574.81 'New York']
 [64664.71 139553.16 137962.62 'California']
 [75328.87 144135.98 134050.07 'Florida']
 [72107.6 127864.55 353183.81 'New York']
 [66051.52 182645.56 118148.2 'Florida']
 [65605.48 153032.06 107138.38 'New York']
 [61994.48 115641.28 91131.24 'Florida']
 [61136.38 152701.92 88218.23 'New York']
 [63408.86 129219.61 46085.25 'California']
 [55493.95 103057.49 214634.81 'Florida']
 [46426.07 157693.92 210797.67 'California']
 [46014.02 85047.44 205517.64 'New York']
 [28663.76 127056.21 201126.82 'Florida']
 [44069.95 51283.14 197029.42 'California']
```

```
[20229.59 65947.93 185265.1 'New York']
[38558.51 82982.09 174999.3 'California']
[28754.33 118546.05 172795.67 'California']
[27892.92 84710.77 164470.71 'Florida']
[23640.93 96189.63 148001.11 'California']
[15505.73 127382.3 35534.17 'New York']
[22177.74 154806.14 28334.72 'California']
[1000.23 124153.04 1903.93 'New York']
[1315.46 115816.21 297114.46 'Florida']
[0.0 135426.92 0.0 'California']
[542.05 51743.15 0.0 'New York']
[0.0 116983.8 45173.06 'California']]
```

In [33]:

```
print(y)
```

```
[192261.83 191792.06 191050.39 182901.99 166187.94 156991.12 156122.51
 155752.6  152211.77 149759.96 146121.95 144259.4  141585.52 134307.35
 132602.65 129917.04 126992.93 125370.37 124266.9  122776.86 118474.03
 111313.02 110352.25 108733.99 108552.04 107404.34 105733.54 105008.31
 103282.38 101004.64  99937.59  97483.56  97427.84  96778.92  96712.8
  96479.51  90708.19  89949.14  81229.06  81005.76  78239.91  77798.83
  71498.49  69758.98  65200.33  64926.08  49490.75  42559.73  35673.41
  14681.4 ]
```

Encoding categorical Data

In [34]:

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [3])], remainder='passthrough')
X = np.array(ct.fit_transform(X))
```

In [35]:
```
X
```

```
array([[0.0, 0.0, 1.0, 165349.2, 136897.8, 471784.1],
       [1.0, 0.0, 0.0, 162597.7, 151377.59, 443898.53],
       [0.0, 1.0, 0.0, 153441.51, 101145.55, 407934.54],
       [0.0, 0.0, 1.0, 144372.41, 118671.85, 383199.62],
       [0.0, 1.0, 0.0, 142107.34, 91391.77, 366168.42],
       [0.0, 0.0, 1.0, 131876.9, 99814.71, 362861.36],
       [1.0, 0.0, 0.0, 134615.46, 147198.87, 127716.82],
       [0.0, 1.0, 0.0, 130298.13, 145530.06, 323876.68],
       [0.0, 0.0, 1.0, 120542.52, 148718.95, 311613.29],
       [1.0, 0.0, 0.0, 123334.88, 108679.17, 304981.62],
       [0.0, 1.0, 0.0, 101913.08, 110594.11, 229160.95],
       [1.0, 0.0, 0.0, 100671.96, 91790.61, 249744.55],
       [0.0, 1.0, 0.0, 93863.75, 127320.38, 249839.44],
       [1.0, 0.0, 0.0, 91992.39, 135495.07, 252664.93],
       [0.0, 1.0, 0.0, 119943.24, 156547.42, 256512.92],
       [0.0, 0.0, 1.0, 114523.61, 122616.84, 261776.23],
       [1.0, 0.0, 0.0, 78013.11, 121597.55, 264346.06],
       [0.0, 0.0, 1.0, 94657.16, 145077.58, 282574.31],
       [0.0, 1.0, 0.0, 91749.16, 114175.79, 294919.57],
       [0.0, 0.0, 1.0, 86419.7, 153514.11, 0.0],
       [1.0, 0.0, 0.0, 76253.86, 113867.3, 298664.47],
       [0.0, 0.0, 1.0, 78389.47, 153773.43, 299737.29],
       [0.0, 1.0, 0.0, 73994.56, 122782.75, 303319.26],
       [0.0, 1.0, 0.0, 67532.53, 105751.03, 304768.73],
       [0.0, 0.0, 1.0, 77044.01, 99281.34, 140574.81],
       [1.0, 0.0, 0.0, 64664.71, 139553.16, 137962.62],
       [0.0, 1.0, 0.0, 75328.87, 144135.98, 134050.07],
       [0.0, 0.0, 1.0, 72107.6, 127864.55, 353183.81],
       [0.0, 1.0, 0.0, 66051.52, 182645.56, 118148.2],
       [0.0, 0.0, 1.0, 65605.48, 153032.06, 107138.38],
       [0.0, 1.0, 0.0, 61994.48, 115641.28, 91131.24],
       [0.0, 0.0, 1.0, 61136.38, 152701.92, 88218.23],
       [1.0, 0.0, 0.0, 63408.86, 129219.61, 46085.25],
       [0.0, 1.0, 0.0, 55493.95, 103057.49, 214634.81],
       [1.0, 0.0, 0.0, 46426.07, 157693.92, 210797.67],
       [0.0, 0.0, 1.0, 46014.02, 85047.44, 205517.64],
       [0.0, 1.0, 0.0, 28663.76, 127056.21, 201126.82],
       [1.0, 0.0, 0.0, 44069.95, 51283.14, 197029.42],
```

```
[0.0, 0.0, 1.0, 20229.59, 65947.93, 185265.1],
[1.0, 0.0, 0.0, 38558.51, 82982.09, 174999.3],
[1.0, 0.0, 0.0, 28754.33, 118546.05, 172795.67],
[0.0, 1.0, 0.0, 27892.92, 84710.77, 164470.71],
[1.0, 0.0, 0.0, 23640.93, 96189.63, 148001.11],
[0.0, 0.0, 1.0, 15505.73, 127382.3, 35534.17],
[1.0, 0.0, 0.0, 22177.74, 154806.14, 28334.72],
[0.0, 0.0, 1.0, 1000.23, 124153.04, 1903.93],
[0.0, 1.0, 0.0, 1315.46, 115816.21, 297114.46],
[1.0, 0.0, 0.0, 0.0, 135426.92, 0.0],
[0.0, 0.0, 1.0, 542.05, 51743.15, 0.0],
[1.0, 0.0, 0.0, 0.0, 116983.8, 45173.06]], dtype=object)
```

Splitting the dataset

In [36]:
```
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 )
```

Training the Multiple Linear Regression Model

In [37]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(X_train, y_train)
```

```
LinearRegression()
```

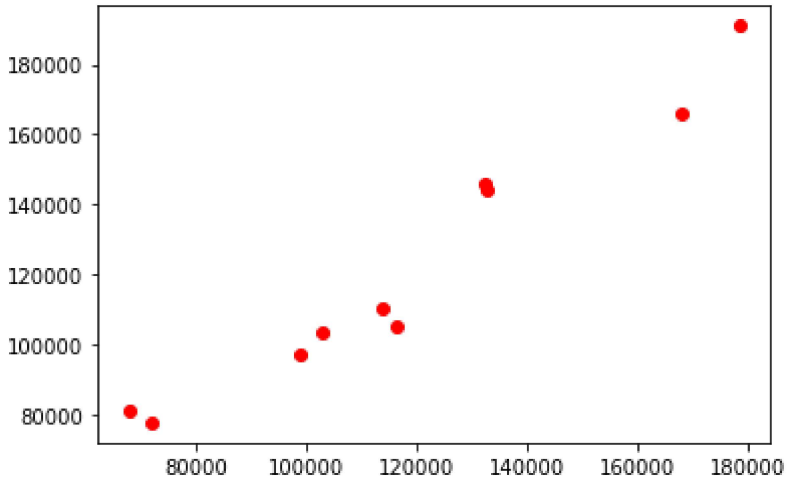
Prediction

In [38]:
```
y_pred=reg.predict(X_test)
y_pred
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

```
[[103015.2  103282.38]
 [132582.28 144259.4 ]
 [132447.74 146121.95]
 [ 71976.1   77798.83]
 [178537.48 191050.39]
 [116161.24 105008.31]
 [ 67851.69  81229.06]
 [ 98791.73  97483.56]
 [113969.44 110352.25]
 [167921.07 166187.94]]
```

In [39]:
```
plt.scatter(y_pred, y_test, color = 'red')
```

```
<matplotlib.collections.PathCollection at 0x1cb4ec16520>
```



Plynomial Regression

In [40]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [41]:
```
dataset = pd.read_csv('C:/Users/navna/Music/datasets/Salary_Data.csv')#D:/Test/Position_Salaries.csv
```

In [42]:
```
dataset.describe
```

```
<bound method NDFrame.describe of     YearsExperience    Salary
0               1.1   39343.0
1               1.3   46205.0
2               1.5   37731.0
3               2.0   43525.0
4               2.2   39891.0
5               2.9   56642.0
6               3.0   60150.0
7               3.2   54445.0
8               3.2   64445.0
9               3.7   57189.0
10              3.9   63218.0
11              4.0   55794.0
12              4.0   56957.0
13              4.1   57081.0
14              4.5   61111.0
15              4.9   67938.0
16              5.1   66029.0
17              5.3   83088.0
18              5.9   81363.0
19              6.0   93940.0
20              6.8   91738.0
21              7.1   98273.0
22              7.9  101302.0
23              8.2  113812.0
24              8.7  109431.0
25              9.0  105582.0
26              9.5  116969.0
27              9.6  112635.0
28             10.3  122391.0
29             10.5  121872.0>
```

In [43]:
```
x = dataset.iloc[:,1:-1].values
y = dataset.iloc[:,-1].values
```

In [44]:
```
x
```

```
array([], shape=(30, 0), dtype=float64)
```

In [45]:
```
y
```

```
array([ 39343.,  46205.,  37731.,  43525.,  39891.,  56642.,  60150.,
        54445.,  64445.,  57189.,  63218.,  55794.,  56957.,  57081.,
        61111.,  67938.,  66029.,  83088.,  81363.,  93940.,  91738.,
        98273., 101302., 113812., 109431., 105582., 116969., 112635.,
       122391., 121872.])
```

Training for Linear Regression

In [46]:

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(x,y)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [46], in <cell line: 3>()
      1 from sklearn.linear_model import LinearRegression
      2 lin_reg = LinearRegression()
----> 3 lin_reg.fit(x,y)

File C:\Anaconda3\lib\site-packages\sklearn\linear_model\_base.py:662, in LinearRegression.fit(self, X, y, sample_weight)
    658 n_jobs_ = self.n_jobs
    660 accept_sparse = False if self.positive else ["csr", "csc", "coo"]
--> 662 X, y = self._validate_data(
    663     X, y, accept_sparse=accept_sparse, y_numeric=True, multi_output=True
    664 )
    666 if sample_weight is not None:
    667     sample_weight = _check_sample_weight(sample_weight, X, dtype=X.dtype)

File C:\Anaconda3\lib\site-packages\sklearn\base.py:581, in BaseEstimator._validate_data(self, X, y, reset, validate_separately, **check_p
arams)
    579         y = check_array(y, **check_y_params)
    580     else:
--> 581         X, y = check_X_y(X, y, **check_params)
    582     out = X, y
    584 if not no_val_X and check_params.get("ensure_2d", True):

File C:\Anaconda3\lib\site-packages\sklearn\utils\validation.py:964, in check_X_y(X, y, accept_sparse, accept_large_sparse, dtype, order,
 copy, force_all_finite, ensure_2d, allow_nd, multi_output, ensure_min_samples, ensure_min_features, y_numeric, estimator)
    961 if y is None:
    962     raise ValueError("y cannot be None")
--> 964 X = check_array(
    965     X,
    966     accept_sparse=accept_sparse,
    967     accept_large_sparse=accept_large_sparse,
    968     dtype=dtype,
```

```
    969     order=order,
    970     copy=copy,
    971     force_all_finite=force_all_finite,
    972     ensure_2d=ensure_2d,
    973     allow_nd=allow_nd,
    974     ensure_min_samples=ensure_min_samples,
    975     ensure_min_features=ensure_min_features,
    976     estimator=estimator,
    977 )
    979 y = _check_y(y, multi_output=multi_output, y_numeric=y_numeric)
    981 check_consistent_length(X, y)

File C:\Anaconda3\lib\site-packages\sklearn\utils\validation.py:814, in check_array(array, accept_sparse, accept_large_sparse, dtype, orde
r, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, estimator)
    812     n_features = array.shape[1]
    813     if n_features < ensure_min_features:
--> 814         raise ValueError(
    815             "Found array with %d feature(s) (shape=%s) while"
    816             " a minimum of %d is required%s."
    817             % (n_features, array.shape, ensure_min_features, context)
    818         )
    820 if copy and np.may_share_memory(array, array_orig):
    821     array = np.array(array, dtype=dtype, order=order)

ValueError: Found array with 0 feature(s) (shape=(30, 0)) while a minimum of 1 is required.
```

In [ ]:

In [ ]: