

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn import linear_model
from collections import Counter
import statsmodels.api as sm

df=pd.read_excel('/content/FINALPROJECT1.xlsx')

df.columns

Index(['Manufacturing Price', 'Sale Price', 'Gross Sales', 'Discounts',
      ' Sales', 'COGS', 'Profit', 'PROFIT_LOSS'],
      dtype='object')

x=df[[' Sales', 'COGS']]

y=df[['PROFIT_LOSS']]

profit_count = (df['PROFIT_LOSS'] == 'yes').sum()
loss_count = (df['PROFIT_LOSS'] == 'no').sum()

print(f'Profit Count: {profit_count}')
print(f'Loss Count: {loss_count}')
```

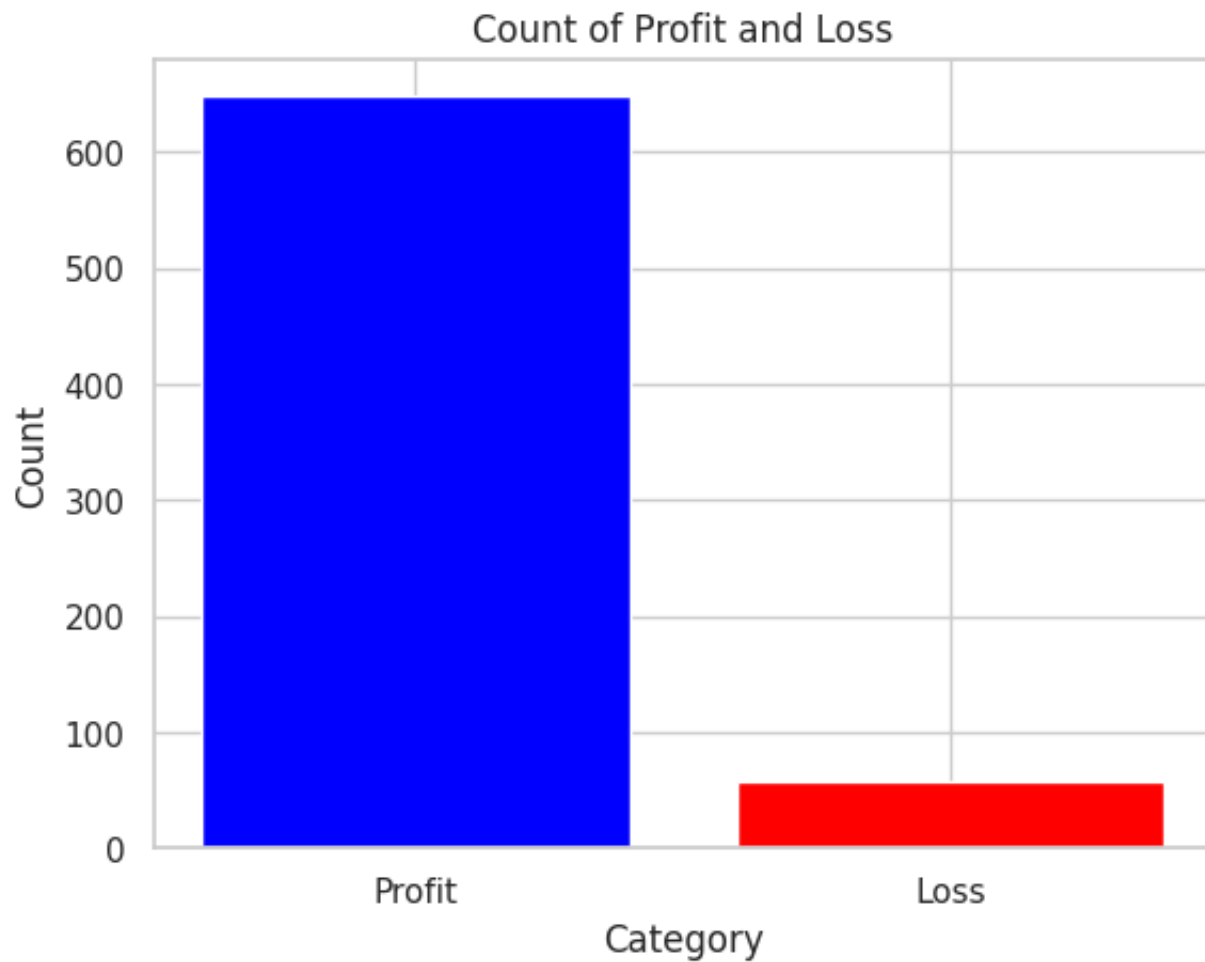
Profit Count: 642  
Loss Count: 58

```
import matplotlib.pyplot as plt



# Counts
profit_count = 648
loss_count = 58

# Create a bar chart
categories = ['Profit', 'Loss']
counts = [profit_count, loss_count]

plt.bar(categories, counts, color=['blue', 'red'])
plt.xlabel('Category')
plt.ylabel('Count')
plt.title('Count of Profit and Loss')
plt.show()
```



X

	Sales	COGS	
0	32370.00	16185.0	
1	26420.00	13210.0	
2	32670.00	21780.0	
3	13320.00	8880.0	
4	37050.00	24700.0	
...	...	...	
695	631125.00	618750.0	
696	139230.00	136500.0	
697	8139.60	6840.0	
698	4301.85	3615.0	
699	18421.20	5418.0	

700 rows x 2 columns

y

	PROFIT_LOSS	
0	yes	
1	yes	
2	yes	
3	yes	
4	yes	
...	...	
695	yes	
696	yes	
697	yes	
698	yes	
699	yes	

700 rows x 1 columns

x.shape

(700, 2)

y.shape

(700, 1)

```
df.describe()
```

	Manufacturing Price	Sale Price	Gross Sales	Sales	COGS	Pr
count	700.000000	700.000000	7.000000e+02	7.000000e+02	700.000000	700.00
mean	96.477143	118.428571	1.827594e+05	1.696091e+05	145475.211429	24133.86
std	108.602612	136.775515	2.542623e+05	2.367263e+05	203865.506118	42760.62
min	3.000000	7.000000	1.799000e+03	1.655080e+03	918.000000	-40617.50
25%	5.000000	12.000000	1.739175e+04	1.592800e+04	7490.000000	2805.96
50%	10.000000	20.000000	3.798000e+04	3.554020e+04	22506.250000	9242.20
75%	250.000000	300.000000	2.790250e+05	2.610775e+05	245607.500000	22662.00
max	260.000000	350.000000	1.207500e+06	1.159200e+06	950625.000000	262200.00

```
df.corr()
```

<ipython-input-151-2f6f6606aa2c>:1: FutureWarning: The default value of numeri

df.corr()

	Manufacturing Price	Sale Price	Gross Sales	Sales	COGS	Profit
Manufacturing Price	1.000000	0.070786	0.049852	0.051549	0.046857	0.061985
Sale Price	0.070786	1.000000	0.808250	0.805878	0.799335	0.650495
Gross Sales	0.049852	0.808250	1.000000	0.998174	0.994519	0.784509
Sales	0.051549	0.805878	0.998174	1.000000	0.992244	0.805462
COGS	0.046857	0.799335	0.994519	0.992244	1.000000	0.725545
Profit	0.061985	0.650495	0.784509	0.805462	0.725545	1.000000





```
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)
```

```
from sklearn.linear_model import LogisticRegression
lrclf = LogisticRegression()
lrclf.fit(x_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: Data
  y = column_or_1d(y, warn=True)
```



```
▼ LogisticRegression
LogisticRegression()
```

x\_train

	Sales	COGS	
420	19383.00	5325.0	
160	15940.98	4108.5	
551	490952.00	414440.0	
54	34095.60	22960.0	
461	202950.00	216480.0	
...	...	...	
198	21801.60	15140.0	
613	29979.60	17430.0	
480	116640.00	108000.0	
557	429660.00	362700.0	
559	15928.00	9050.0	


560 rows x 2 columns

x\_test

	Sales	COGS	
42	534450.00	397020.0	
428	21359.52	5868.0	
614	18035.92	14980.0	
522	49929.00	28050.0	
137	76146.00	64750.0	
...	...	...	
645	354277.00	306020.0	
504	246708.00	205920.0	
376	10291.12	7990.0	
549	15928.00	9050.0	
395	18721.08	14535.0	

140 rows x 2 columns


y\_train

	PROFIT_LOSS	
420	yes	
160	yes	
551	yes	
54	yes	
461	no	
...	...	
198	yes	
613	yes	
480	yes	
557	yes	
559	yes	

560 rows x 1 columns




y\_test

PROFIT_LOSS		
42	yes	
428	yes	
614	yes	
522	yes	
137	yes	
...	...	
645	yes	
504	yes	
376	yes	
549	yes	
395	yes	



140 rows x 1 columns

x\_test

	Sales	COGS	
42	534450.00	397020.0	
428	21359.52	5868.0	
614	18035.92	14980.0	
522	49929.00	28050.0	
137	76146.00	64750.0	
...	...	...	
645	354277.00	306020.0	
504	246708.00	205920.0	
376	10291.12	7990.0	
549	15928.00	9050.0	
395	18721.08	14535.0	

140 rows x 2 columns

y\_test

PROFIT_LOSS		
42	yes	
428	yes	
614	yes	
522	yes	
137	yes	
...	...	
645	yes	
504	yes	
376	yes	
549	yes	
395	yes	

140 rows x 1 columns

lrclf.predict(x\_test)

```
array(['yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'no', 'yes',
      'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'no',
      'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
      'no', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
      'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
      'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
      'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
      'yes', 'yes', 'yes', 'yes', 'no', 'yes', 'yes', 'yes', 'yes', 'no',
      'yes', 'no', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
      'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
      'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
      'no', 'yes', 'yes', 'no', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
      'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'no', 'yes', 'no',
      'yes', 'yes', 'yes', 'yes', 'no', 'yes', 'yes', 'yes', 'yes',
      'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes', 'yes',
      'yes', 'yes'], dtype=object)
```

```
lrclf.score(x_test,y_test)
```

```
1.0
```