Untitled67

December 10, 2025

[32]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
from scipy.stats import shapiro
from statsmodels.stats.diagnostic import het_breuschpagan
from statsmodels.stats.stattools import durbin_watson
```

[2]:
```python
data = pd.read_csv(r"C:\Users\USER\Downloads\Startups.csv")
data.head()
```

[2]:
```
   R&D Expenditure  Administration Expenditure  Marketing Expenditure  \
0        165349.20                   136897.80              471784.10
1        162597.70                   151377.59              443898.53
2        153441.51                   101145.55              407934.54
3        144372.41                   118671.85              383199.62
4        142107.34                    91391.77              366168.42

     State     Profit
0  Florida  192261.83
1  Florida  191792.06
2  Florida  191050.39
3  Florida  182901.99
4  Florida  166187.94
```

[3]:
```python
data.shape
```

[3]:
```
(50, 5)
```

[4]:
```python
data.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 Expenditure                50 non-null     float64
 1   Administration Expenditure     50 non-null     float64
 2   Marketing Expenditure          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
```

[5]:
```
data.duplicated().sum()
```

[5]:
```
np.int64(0)
```

[6]:
```
data.isnull().sum()
```

[6]:
```
R&D Expenditure               0
Administration Expenditure    0
Marketing Expenditure         0
State                         0
Profit                        0
dtype: int64
```

[7]:
```
data.describe()
```

[7]:
```
       R&D Expenditure  Administration Expenditure  Marketing Expenditure  \
count        50.000000                   50.000000              50.000000
mean      73721.615600               121344.639600          211025.097800
std       45902.256482                28017.802755          122290.310726
min           0.000000                51283.140000               0.000000
25%       39936.370000               103730.875000          129300.132500
50%       73051.080000               122699.795000          212716.240000
75%      101602.800000               144842.180000          299469.085000
max      165349.200000               182645.560000          471784.100000

              Profit
count      50.000000
mean   112012.639200
std     40306.180338
min     14681.400000
25%     90138.902500
50%    107978.190000
75%    139765.977500
max    192261.830000
```

[8]:
```
y = data['Profit']
x = data[['Administration Expenditure','Marketing Expenditure','R&D␣
↪Expenditure']]
```

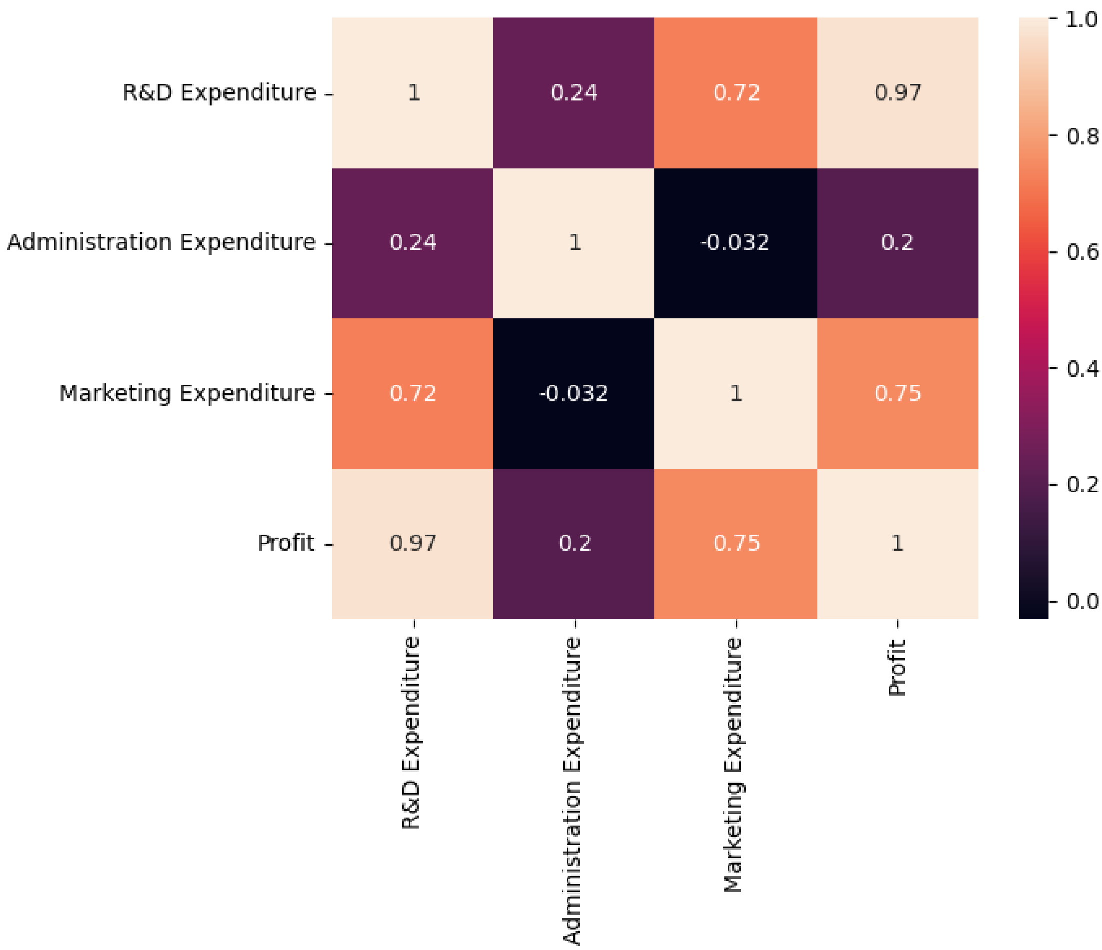
[9]:
```
df = data.select_dtypes(include=['number'])
correlation = df.corr()
print(correlation)
```

```
                            R&D Expenditure  Administration Expenditure  \
R&D Expenditure                    1.000000                    0.241955
Administration Expenditure         0.241955                    1.000000
Marketing Expenditure              0.724248                   -0.032154
Profit                             0.972900                    0.200717

                            Marketing Expenditure    Profit
R&D Expenditure                          0.724248  0.972900
Administration Expenditure              -0.032154  0.200717
Marketing Expenditure                    1.000000  0.747766
Profit                                   0.747766  1.000000
```

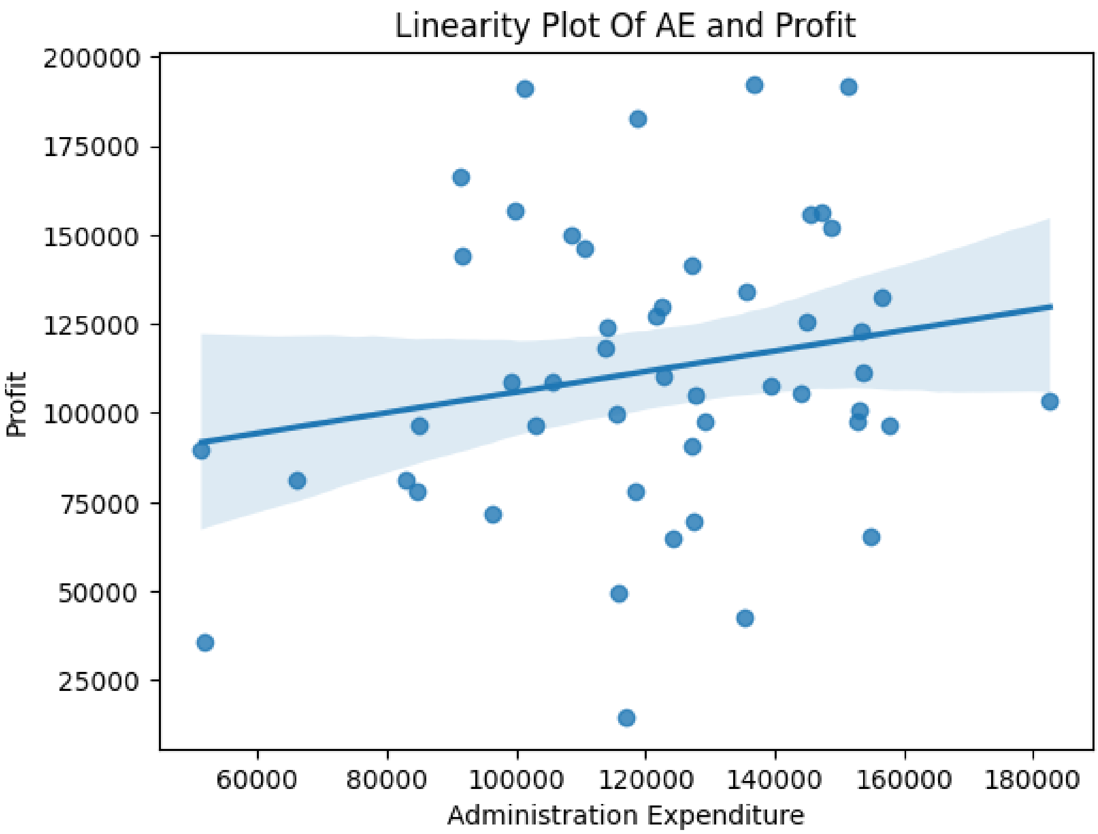
[10]:
```
Heatmap = sns.heatmap(correlation, annot= True)
plt.show()
```

```
<IPython.core.display.Javascript object>
```



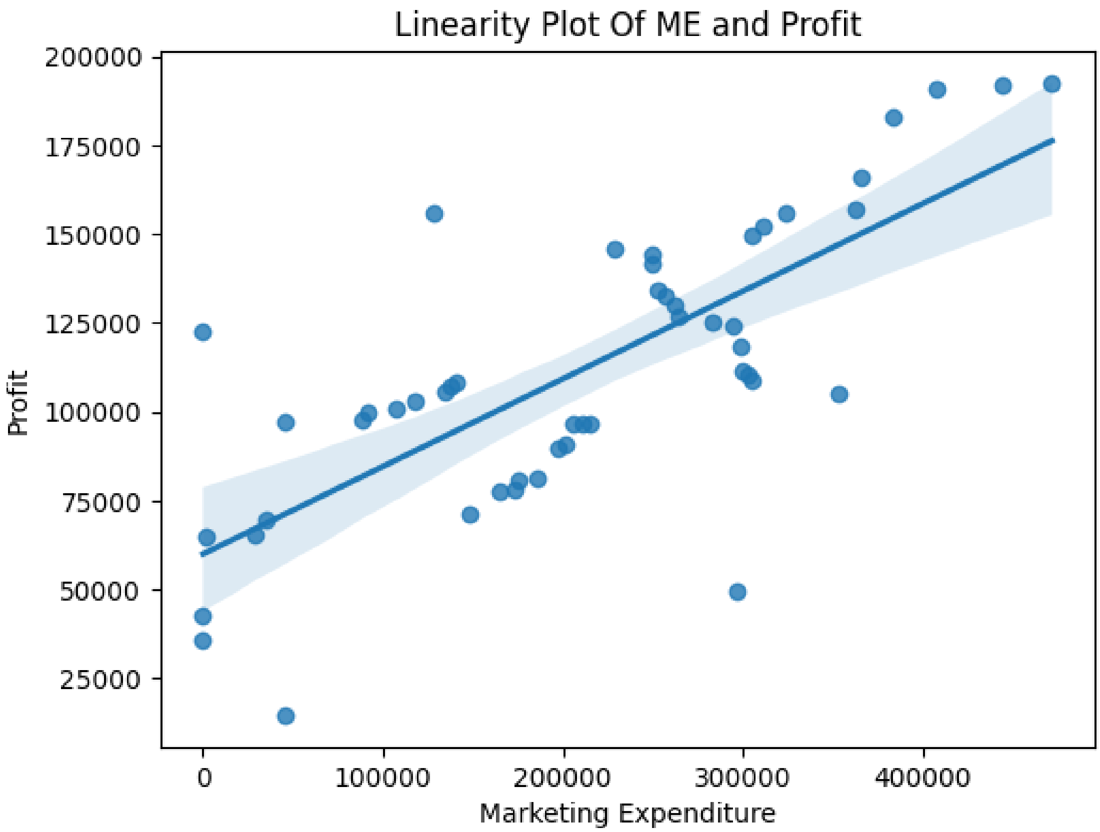
[11]:
```
sns.regplot(x='Administration Expenditure', y='Profit', data=data)
plt.title('Linearity Plot Of AE and Profit')
plt.show()
```

<IPython.core.display.Javascript object>



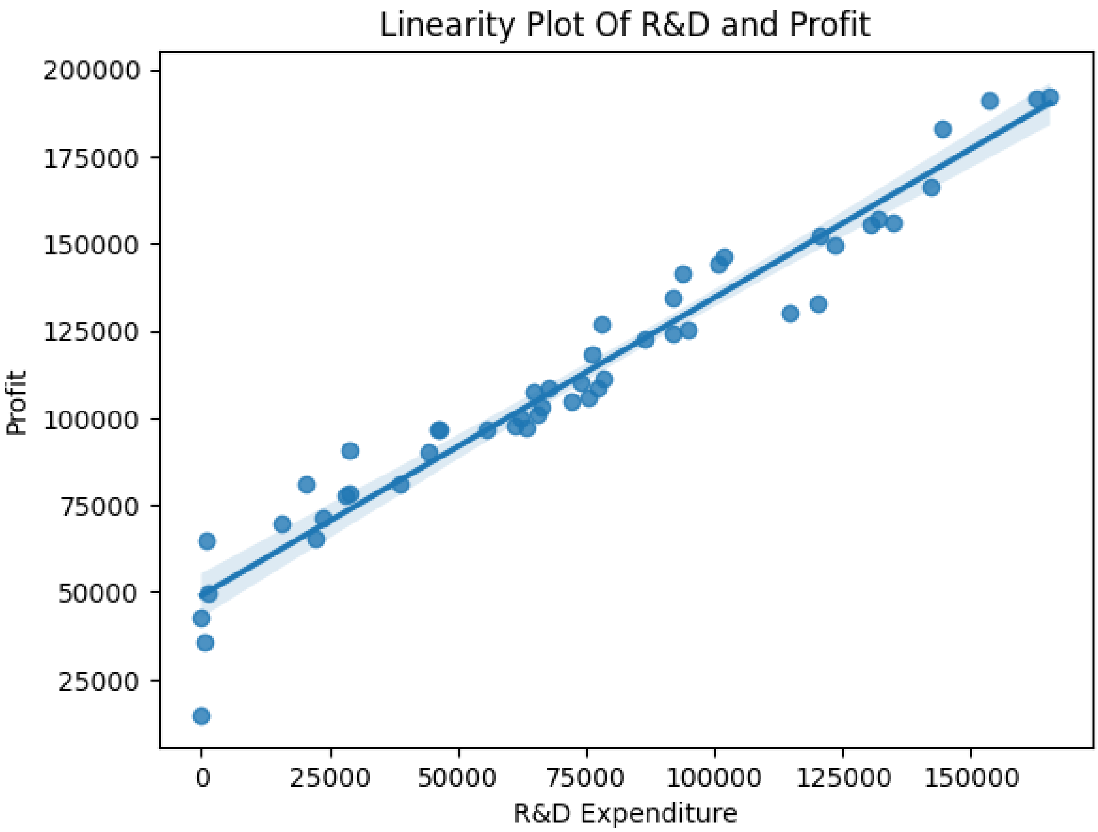
[12]:
```
sns.regplot(x='Marketing Expenditure', y='Profit', data=data)
plt.title('Linearity Plot Of ME and Profit')
plt.show()
```

<IPython.core.display.Javascript object>



[13]:
```
sns.regplot(x='R&D Expenditure', y='Profit', data=data)
plt.title('Linearity Plot Of R&D and Profit')
plt.show()
```

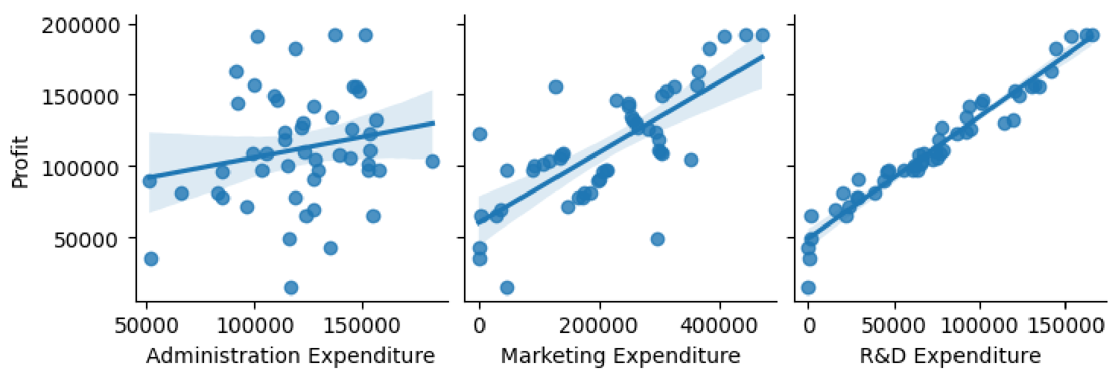
```
<IPython.core.display.Javascript object>
```



[17]:
```
sns.pairplot(df, x_vars=x.columns, y_vars='Profit', kind='reg')
```

```
<IPython.core.display.Javascript object>
```

[17]:
```
<seaborn.axisgrid.PairGrid at 0x17d1a15d940>
```



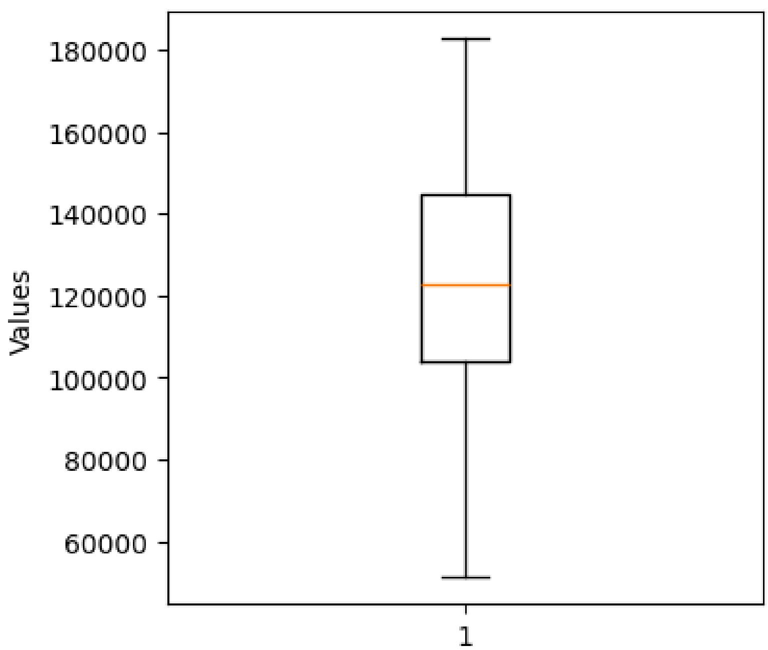
[19]:

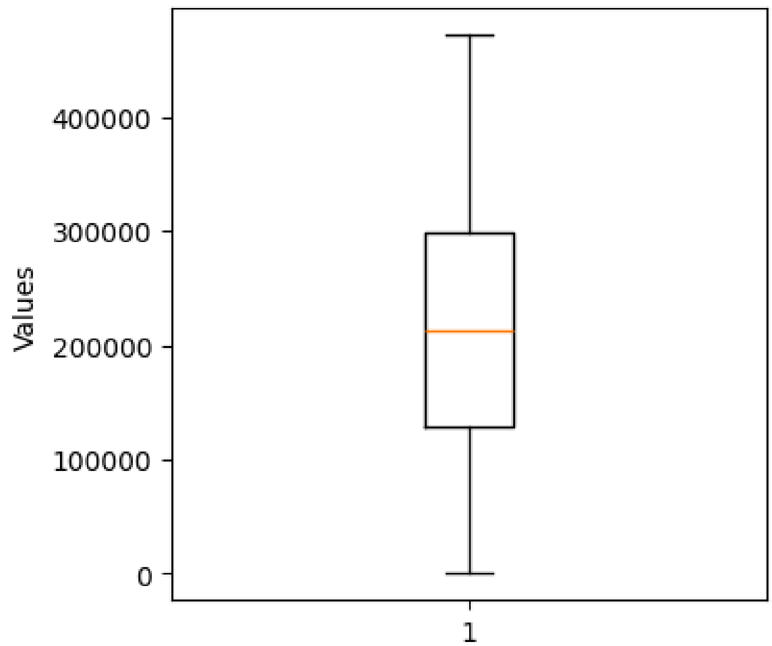
```
X_const = sm.add_constant(x)
vif = pd.DataFrame({
    'Feature': X_const.columns,
    'VIF': [variance_inflation_factor(X_const.values, i)
            for i in range(X_const.shape[1])]
})
print(vif)
```

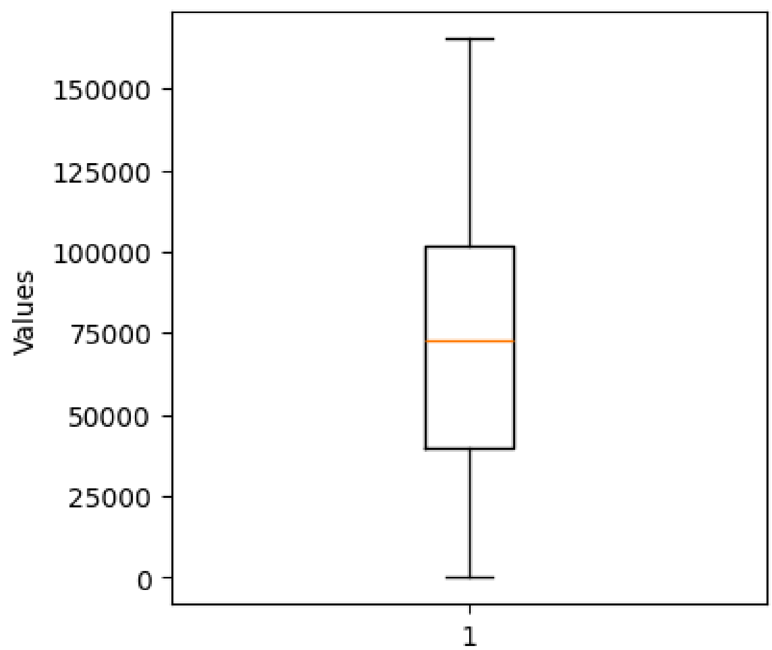
```
                    Feature        VIF
0                     const  25.338950
1  Administration Expenditure   1.175091
2       Marketing Expenditure   2.326773
3             R&D Expenditure   2.468903
```

[20]:

```
for col in x.columns:
    plt.figure(figsize=(4, 4))
    plt.boxplot(x[col], vert=True)
    plt.ylabel('Values')
    plt.show()
```







[21]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.
↪2,random_state=42)
```

[22]:
```
model = LinearRegression()
model.fit(x_train,y_train)
```

[22]: LinearRegression()

[24]:
```
y_pred_train = model.predict(x_train)
y_pred_test = model.predict(x_test)
```

[27]:
```
print(r2_score(y_test, y_pred_test))
print(mean_squared_error(y_test, y_pred_test))
print(mean_absolute_error(y_test, y_pred_test))
```

```
0.900065308303732
80926321.22295165
6979.1522523704025
```

[28]:
```
residuals_train = y_train - y_pred_train
residuals_test = y_test - y_pred_test
```

[31]:
```
stat, p = shapiro(residuals_train)
p
```

[31]: np.float64(0.013342603561094615)

[34]:
```
dw = durbin_watson(residuals_train)
dw
```

[34]: np.float64(1.7593850382807832)

[36]:
```
X_train_const = sm.add_constant(x_train)
bp = het_breuschpagan(residuals_train, X_train_const)
bp
```

[36]:
```
(np.float64(2.301709808738015),
 np.float64(0.5121934321518875),
 np.float64(0.7326729558482282),
 np.float64(0.5393370968568404))
```

[ ]:
```

```