```
#Amit Sharad pawar
#PRN:25070126501
#Multiple Linear Regression
#PracticeMLR50_Startups dataset use
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
import seaborn as sns
```

```
df=pd.read_csv('/content/PracticeMLR50_Startups.csv')
```

```
df.head(20)
```

	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
5	131876.90	99814.71	362861.36	New York	156991.12
6	134615.46	147198.87	127716.82	California	156122.51
7	130298.13	145530.06	323876.68	Florida	155752.60
8	120542.52	148718.95	311613.29	New York	152211.77
9	123334.88	108679.17	304981.62	California	149759.96
10	101913.08	110594.11	229160.95	Florida	146121.95
11	100671.96	91790.61	249744.55	California	144259.40
12	93863.75	127320.38	249839.44	Florida	141585.52
13	91992.39	135495.07	252664.93	California	134307.35
14	119943.24	156547.42	256512.92	Florida	132602.65
15	114523.61	122616.84	261776.23	New York	129917.04
16	78013.11	121597.55	264346.06	California	126992.93
17	94657.16	145077.58	282574.31	New York	125370.37
18	91749.16	114175.79	294919.57	Florida	124266.90
19	86419.70	153514.11	0.00	New York	122776.86

Next steps: Generate code with df New interactive sheet

```
df.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
```

```
df=df.iloc[:,1:]
```

```
df.head()
```

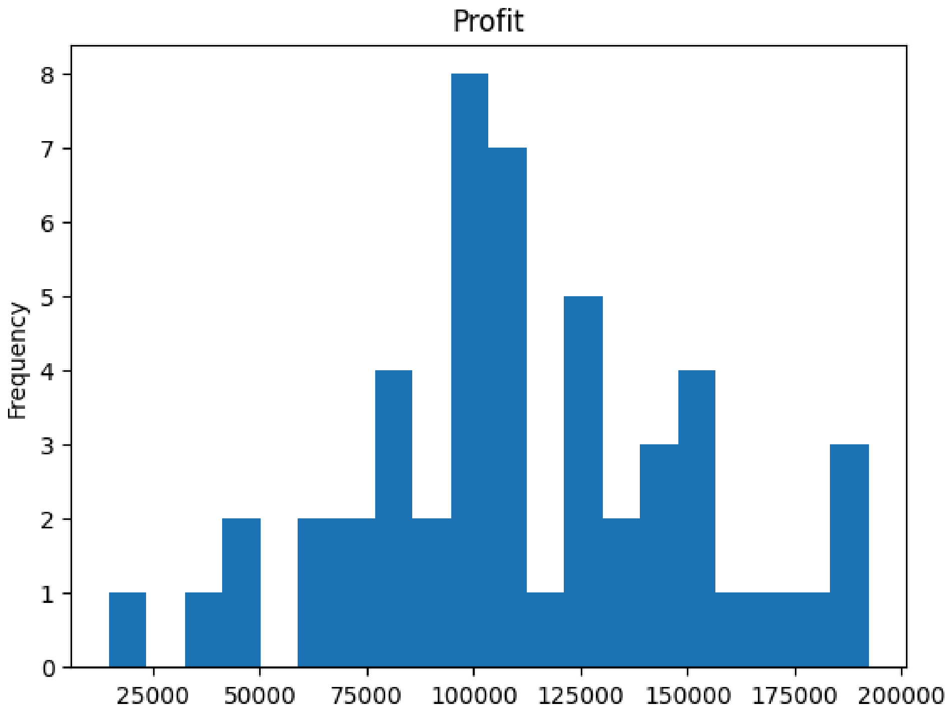
	Administration	Marketing Spend	State	Profit
0	136897.80	471784.10	New York	192261.83
1	151377.59	443898.53	California	191792.06
2	101145.55	407934.54	Florida	191050.39
3	118671.85	383199.62	New York	182901.99
4	91391.77	366168.42	Florida	166187.94

Next steps: Generate code with df New interactive sheet

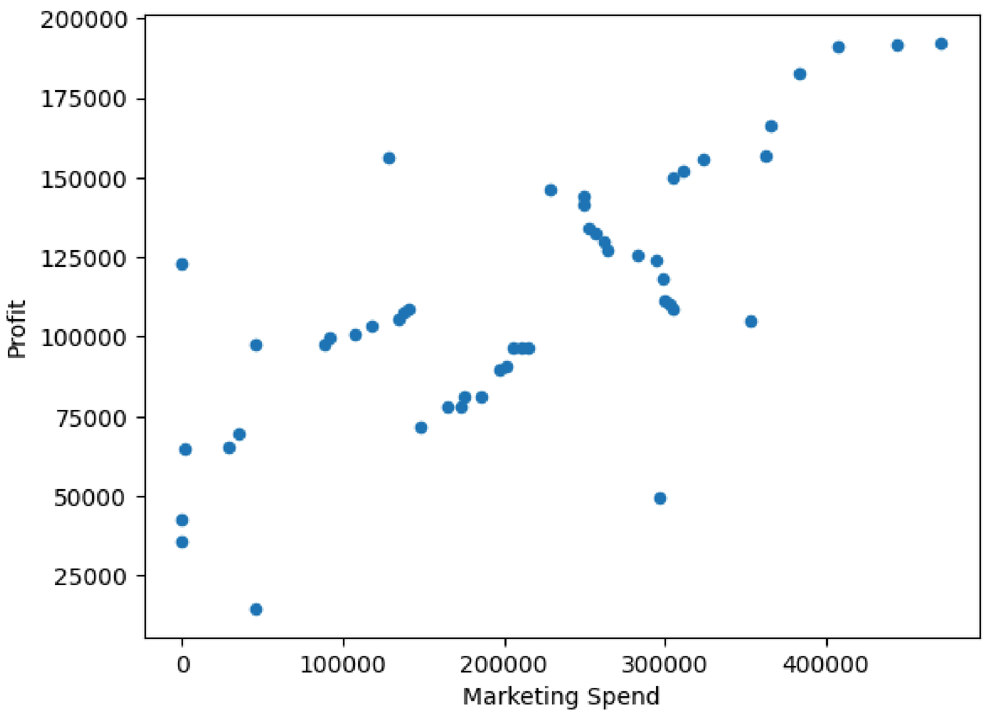
```
df.shape
```

```
(50, 4)
```

```
df['Profit'].plot(kind='hist',bins=20,title='Profit')
plt.show()
```

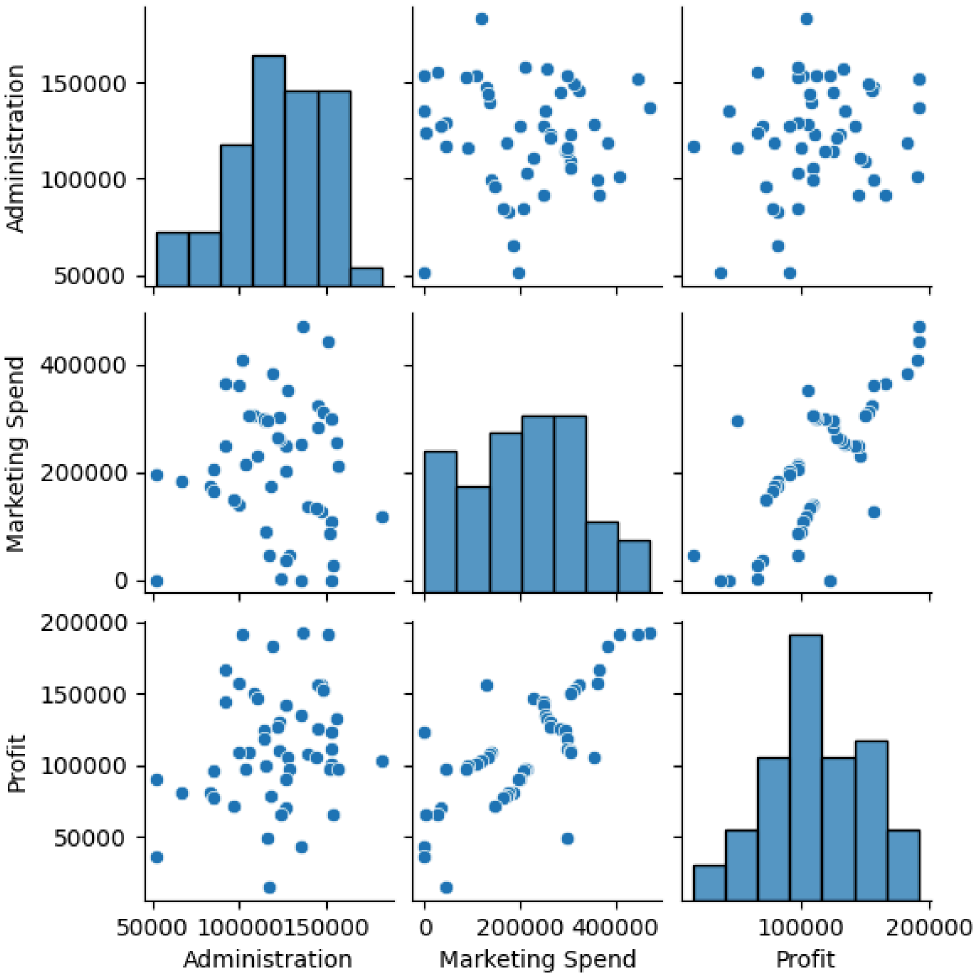


```
df.plot(kind='scatter',x='Marketing Spend',y='Profit',alpha=1)
plt.show()
```



```
sns.pairplot(data=df,height=2)
```

```
<seaborn.axisgrid.PairGrid at 0x7b1d00b93ec0>
```



We check VIF before moving further

,VIF starts from 1 and no upper limit, if vif=1 then no correlation

- if above 5 or 10 then remove

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

Double-click (or enter) to edit

```
def calc_vif(X):
  vif=pd.DataFrame()
  vif["VIF"] = [variance_inflation_factor(X.values,i)
                    for i in range(X.shape[1])]
  vif.index=X.columns # directly assign column names
  return vif
```

```
X = df.iloc[:,:-1]
# Drop the 'State' column as VIF is typically calculated for numerical features
X_numeric = X.drop('State', axis=1)
calc_vif(X_numeric)
```

	VIF
Administration	3.43653
Marketing Spend	3.43653

Setting values for independent x and dependent y variable

```
x = df[['Administration','Marketing Spend']]
y = df['Profit']
```

```
x.head()
```

	Administration	Marketing Spend
0	136897.80	471784.10
1	151377.59	443898.53
2	101145.55	407934.54
3	118671.85	383199.62
4	91391.77	366168.42

Next steps: Generate code with x   New interactive sheet

```
y.head()
```

	Profit
0	192261.83
1	191792.06
2	191050.39
3	182901.99
4	166187.94

dtype: float64

```
from sklearn.model_selection import train_test_split
x_train,xtest,ytrain,ytest = train_test_split(x,y,test_size = 0.20,random_state = 42)
```

Implement Linear Model

```
from sklearn.linear_model import LinearRegression
mlr = LinearRegression()
mlr.fit(x_train,ytrain)
```

```
▾ LinearRegression  i ?
LinearRegression()
```

```
print("Intercept: ",mlr.intercept_)
print("Coefficients: ",mlr.coef_)
list(zip(x,mlr.coef_))
```

```
Intercept:  15816.945810641628
Coefficients:  [0.26096614 0.28938036]
[('Administration', np.float64(0.26096613654763695)),
 ('Marketing Spend', np.float64(0.2893803557601718))]
```

Prediction on test set

```
#prediction of test set
y_pred_mlr=mlr.predict(xtest)
print("Prediction for test set: {}".format(y_pred_mlr))
```

```
Prediction for test set: [124292.83808131  88113.82093237  72366.99452973  48767.64494083
 135448.73571941  29320.15575895  92223.01259503  92159.26687606
  62875.05423883  55878.93000289]
```

Combine Actual Values and Predicted value

```
mlr_diff=pd.DataFrame({'Actual Value': ytest, 'Predicted Value': y_pred_mlr})
mlr_diff.head()
```

	Actual Value	Predicted Value
13	134307.35	124292.838081
39	81005.76	88113.820932
30	99937.59	72366.994530
45	64926.08	48767.644941
17	125370.37	135448.735719

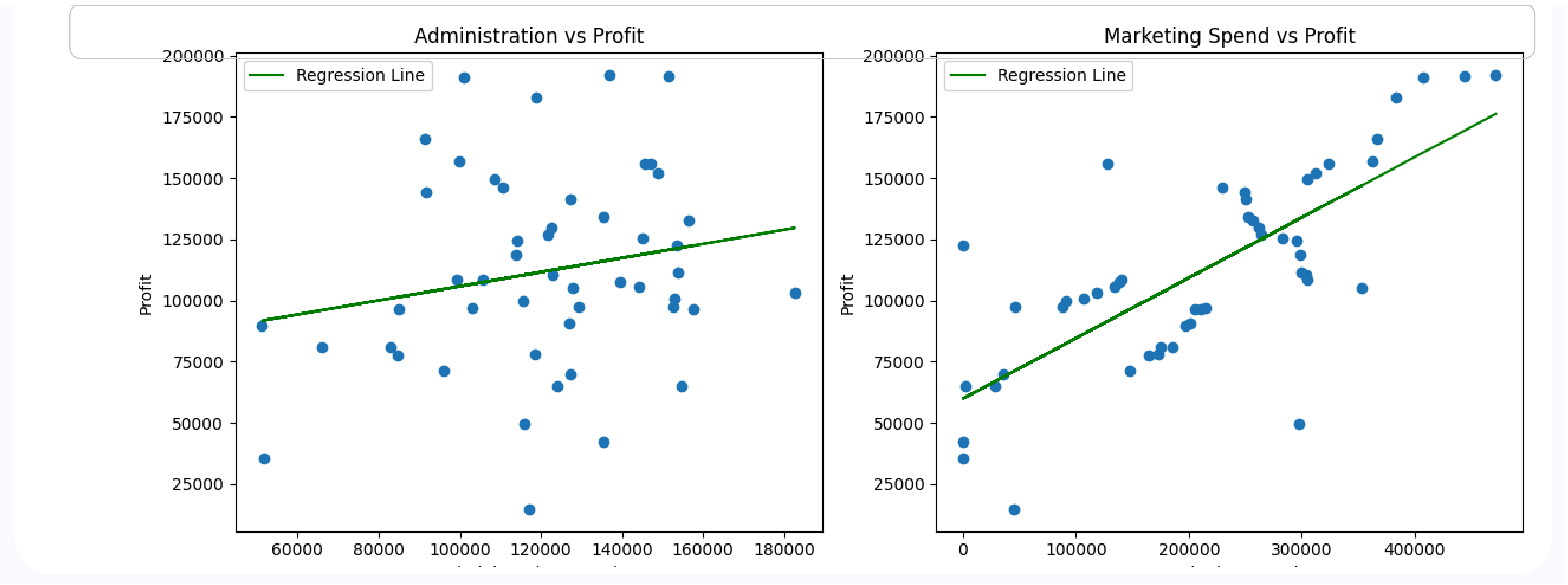
Next steps: Generate code with `mlr_diff` New interactive sheet

Evaluating the Model

```
import numpy as np
from sklearn import metrics
meanAbsErr = metrics.mean_absolute_error(ytest,y_pred_mlr)
meanSqErr = metrics.mean_squared_error(ytest,y_pred_mlr)
rootMeanSqErr = np.sqrt(metrics.mean_squared_error(ytest,y_pred_mlr))
print("R square: {:.2f}".format(mlr.score(x,y)*100))
print("Mean Absolute Error: ",meanAbsErr)
print("Mean Square Error: ",meanSqErr)
print("Root Mean Square Error: ",rootMeanSqErr)
```

```
R square: 58.48
Mean Absolute Error:  20748.953962815332
Mean Square Error:  739815967.943038
Root Mean Square Error:  27199.55823065952
```

```
fig, axes = plt.subplots(1, 2, figsize=(12, 5)) # Changed to 2 subplots as we have two numerical independent variables

def plot_scatter_with_regression(ax, x_data, y_data, x_label, title):
  ax.scatter(x_data, y_data, alpha=1)

  # calculating reg line
  slope, intercept = np.polyfit(x_data, y_data, 1)
  regression_line = slope * x_data + intercept

  ax.plot(x_data, regression_line, color='green', label='Regression Line')
  ax.set_title(title)
  ax.set_xlabel(x_label)
  ax.set_ylabel('Profit') # Changed y-label to 'Profit'
  ax.legend()


# Scatter plot for Administration vs Profit
plot_scatter_with_regression(axes[0], df['Administration'], df['Profit'], 'Administration Spend', 'Administration vs Profit')
# Scatter plot for Marketing Spend vs Profit
plot_scatter_with_regression(axes[1], df['Marketing Spend'], df['Profit'], 'Marketing Spend', 'Marketing Spend vs Profit')

plt.tight_layout()
plt.show()
```



Administration vs Profit

Marketing Spend vs Profit