Multiple Linear Regression

import required package

In [1]:

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

read data from source and describing

In [2]:

```
df = pd.read_csv('50_Startups.csv')
df.head()
```

Out[2]:

	RnD	Administration	Marketing	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 [3]:

```
print(df.columns)
```

```
Index(['RnD', 'Administration', 'Marketing', 'State', 'Profit'], dtype='object')
```

In [4]:

```
print(df.describe())
```

```
                 RnD  Administration      Marketing         Profit
count      50.000000       50.000000      50.000000      50.000000
mean    73721.615600   121344.639600  211025.097800  112012.639200
std     45902.256482    28017.802755  122290.310726   40306.180338
min         0.000000    51283.140000       0.000000   14681.400000
25%     39936.370000   103730.875000  129300.132500   90138.902500
50%     73051.080000   122699.795000  212716.240000  107978.190000
75%    101602.800000   144842.180000  299469.085000  139765.977500
max    165349.200000   182645.560000  471784.100000  192261.830000
```

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   RnD             50 non-null     float64
 1   Administration  50 non-null     float64
 2   Marketing       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
```

check the relation between variables

In [6]:

```
## state value is excluded as it is categorical
corr = df.corr()
corr.style.background_gradient(cmap = 'Greens')
```

Out[6]:

	RnD	Administration	Marketing	Profit
RnD	1.000000	0.241955	0.724248	0.972900
Administration	0.241955	1.000000	-0.032154	0.200717
Marketing	0.724248	-0.032154	1.000000	0.747766
Profit	0.972900	0.200717	0.747766	1.000000

data cleansing and normilasation

In [7]:

```
## since state values are categorical it would be replaced with other values
```

In [8]:

```
state_values = df['State'].unique()
print(state_values)

unique_values = [1,2,3]

df.replace(state_values, unique_values, inplace= True)
print(df.head())
```

```
['New York' 'California' 'Florida']
         RnD  Administration  Marketing  State     Profit
0  165349.20       136897.80  471784.10      1  192261.83
1  162597.70       151377.59  443898.53      2  191792.06
2  153441.51       101145.55  407934.54      3  191050.39
3  144372.41       118671.85  383199.62      1  182901.99
4  142107.34        91391.77  366168.42      3  166187.94
```

EDA

In [9]:

```
corr = df.corr()
corr.style.background_gradient(cmap = 'Greens')
```

Out[9]:

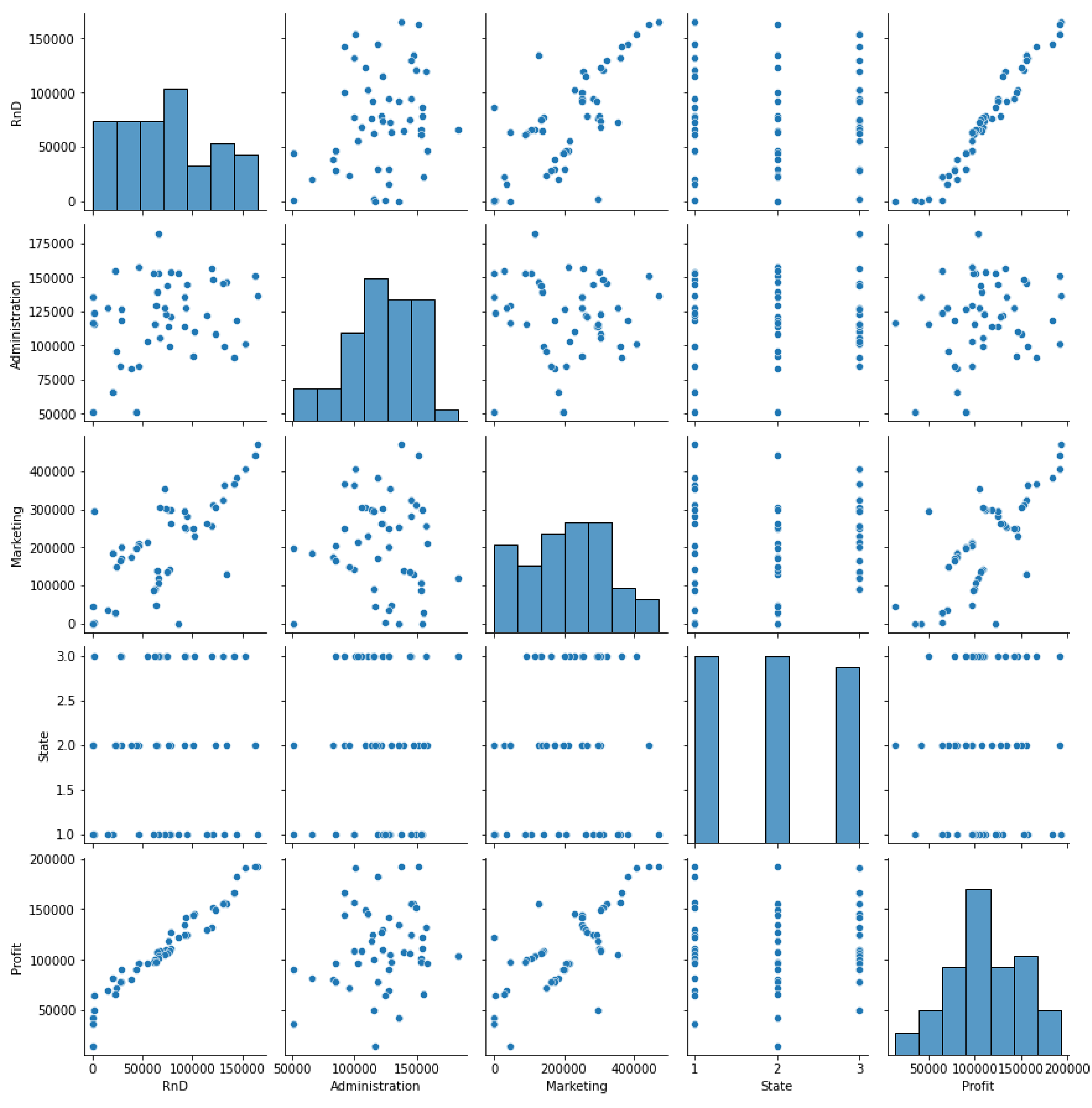
	RnD	Administration	Marketing	State	Profit
RnD	1.000000	0.241955	0.724248	0.037930	0.972900
Administration	0.241955	1.000000	-0.032154	0.003026	0.200717
Marketing	0.724248	-0.032154	1.000000	0.137777	0.747766
State	0.037930	0.003026	0.137777	1.000000	0.048471
Profit	0.972900	0.200717	0.747766	0.048471	1.000000

In [10]:

```
sns.pairplot(df)
```

Out[10]:

```
<seaborn.axisgrid.PairGrid at 0x1e9290817f0>
```



select input and op variable

In [11]:

```
x = df.drop(['State', 'Profit'], axis=1)
y = df['Profit']
```

split the data

In [12]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state= 123456, train_
```

creating a model

In [13]:

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

In [14]:

```
## fit the data
model.fit(x_train, y_train)
```

Out[14]:

```
LinearRegression()
```

evaluate the model

In [15]:

```
score = model.score(x_train, y_train)
print(score)
```

```
0.9601722295771402
```

parameters to fine tune model

In [16]:

```
print(model.get_params())
```

```
{'copy_X': True, 'fit_intercept': True, 'n_jobs': None, 'normalize': False, 'positiv
e': False}
```

predict the values for unseen data

In [17]:

```
y_prediction = model.predict(x_test)
print(y_prediction)
```

```
[ 67329.47406593 117366.3802916   91360.40144635 154304.50169146
  76117.37130998  98755.52129056  50827.79389197  99500.68282807
 129036.46220559 103456.53045388]
```

evaluation of loss functions

In [18]:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

MAE = mean_absolute_error(y_test,y_prediction)
MSE = mean_squared_error(y_test, y_prediction)
RMSE = np.sqrt(MSE)
R2 = r2_score(y_test, y_prediction)

print(MAE)
print(MSE)
print(RMSE)
print(R2)
```

```
7046.237337953069
149018716.7626929
12207.322260131126
0.8813666867792831
```

visualization only possible for 2d values

In [ ]: