In [1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```python
df=pd.read_csv(r'C:\Users\sharma17\Downloads\ml\regression dataset\Startups.csv')
```

In [3]:
```python
df.head()
```

Out[3]:

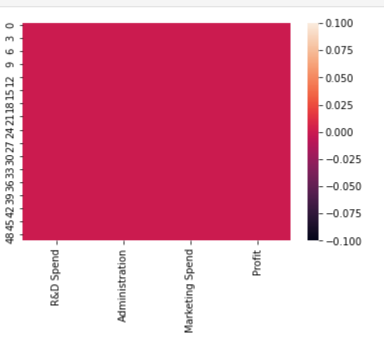
| | R&D Spend | Administration | Marketing Spend | State | Profit |
|---|---|---|---|---|---|
| 0 | 165349.20 | 136897.80 | 471784.10 | New York | 192261.83 |
| 1 | 162597.70 | 151377.59 | 443898.53 | California | 191792.06 |
| 2 | 153441.51 | 101145.55 | 407934.54 | Florida | 191050.39 |
| 3 | 144372.41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

In [5]:
```python
df.drop('State',inplace=True,axis=1)
```

In [6]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 4 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   Profit           50 non-null     float64
dtypes: float64(4)
memory usage: 1.7 KB
```

In [7]:
```python
sns.heatmap(df.isnull())
plt.show()
```

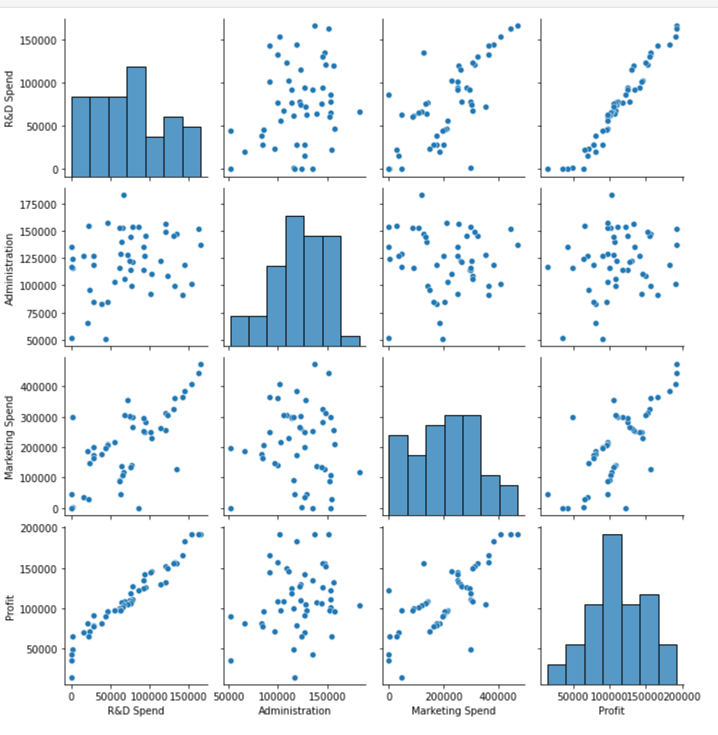



In [8]:
```python
df.describe()
```

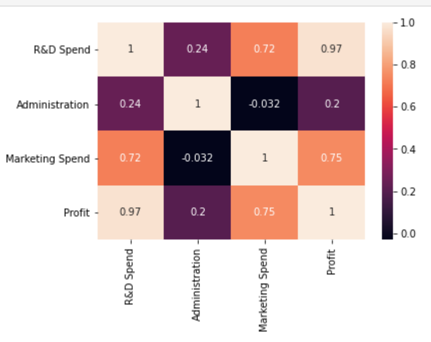
Out[8]:

| | R&D Spend | Administration | Marketing Spend | 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 [9]:
```python
sns.pairplot(data=df)
plt.show()
```

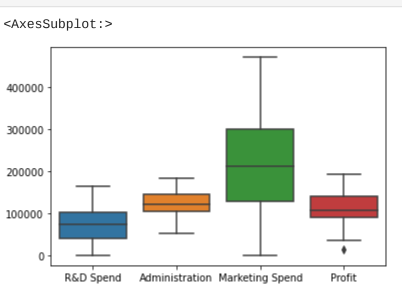



In [10]:
```python
sns.heatmap(df.corr(),annot=True)
plt.show()
```




In [11]:
```python
sns.boxplot(data=df)
```

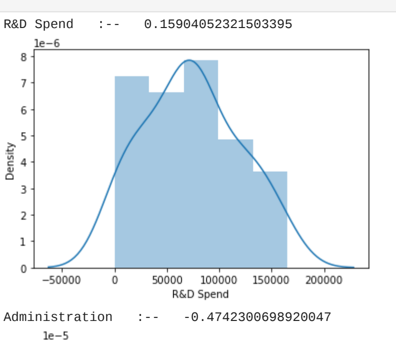
Out[11]: <AxesSubplot:>



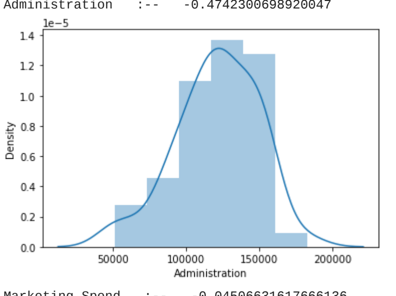

In [12]:
```python
from scipy.stats import skew
```

In [13]:
```python
for col in df:

    print (col,'  :--   ',end='' )
    print(skew(df[col]))
    sns.distplot(df[col])
    plt.show()
```

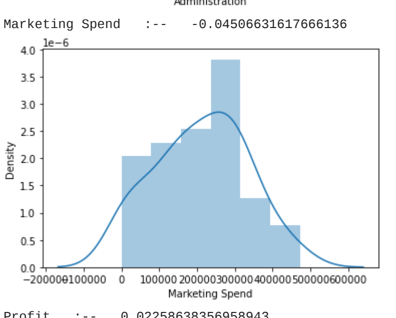
R&D Spend   :--   0.15904052321503395



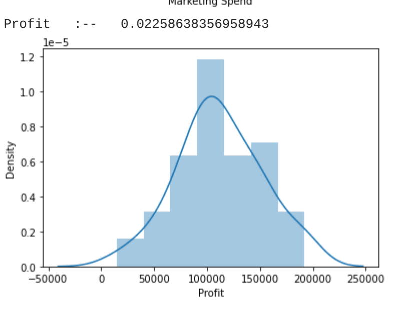

Administration   :--   -0.4742300698920047




Marketing Spend   :--   -0.04506631617666136




Profit   :--   0.02258638356958943




In [64]:
```python
col=df.drop('Profit',axis=1).columns
x = df.iloc[:,:-1].values
y = df.iloc[:,-1].values
```

In [65]:
```python
from sklearn.linear_model import LinearRegression ,Lasso,Ridge
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor ,AdaBoostRegressor,GradientBoostingRegressor
from xgboost import XGBRegressor
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import r2_score,mean_squared_error ,mean_absolute_error
from sklearn.model_selection import train_test_split ,cross_val_score
```

In [66]:
```python
models=[('Linear Regression',LinearRegression()),
        ('Decision Tree Regressor',DecisionTreeRegressor()),
        ('Lasso Regression  ',Lasso()),
        ('Ridge Regression  ',Ridge()),
        ('Support vector Regression  ',SVR(kernel='linear')),
        ('K Nearest Neighbors Regresor',KNeighborsRegressor()),
        ('Random Forest Regressor',RandomForestRegressor()),
        ('AdaBoost Regressor',AdaBoostRegressor()),
        ('Gradient Boosting Regressor',GradientBoostingRegressor()),
        ('Xtreme Gradient Boosting Regressor',XGBRegressor())
]
```

In [67]:
```python
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=.20,random_state=0)
```

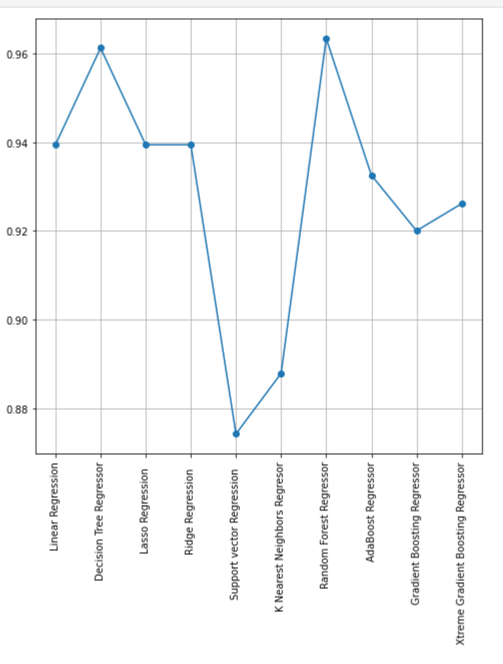
In [68]:
```python
accuracy=[]

for name,model in models:

    print(name,  ':--  ')
    print()
    model.fit(xtrain,ytrain)
    ypred=model.predict(xtest)
    print('       mean absolute error            :-- ',mean_absolute_error(ytest,ypred))
    print()

    print('       mean squared error             :--  ',mean_squared_error(ytest,ypred))

    print()


    print('       squar root  mean squared error :--  ',np.sqrt(mean_squared_error(ytest,ypred)))


    print()
    print('       accuracy                       :-  ',r2_score(ytest,ypred)*100)

    print()

    accuracy.append(r2_score(ytest,ypred))
```

```
Linear Regression :--

       mean absolute error            :--  7320.441614848123

       mean squared error             :--   77506468.16885388

       squar root  mean squared error :--   8803.77579046933

       accuracy                       :-   93.93955917820573

Decision Tree Regressor :--

       mean absolute error            :--  5232.012000000004

       mean squared error             :--   49607308.50916002

       squar root  mean squared error :--   7043.245594834814

       accuracy                       :-   96.12107009064994

Lasso Regression   :--

       mean absolute error            :--  7320.441565311366

       mean squared error             :--   77506467.1863158

       squar root  mean squared error :--   8803.775734667246

       accuracy                       :-   93.93955925503303

Ridge Regression   :--

       mean absolute error            :--  7320.4416150478255

       mean squared error             :--   77506468.17418559

       squar root  mean squared error :--   8803.775790772139

       accuracy                       :-   93.93955917778882

Support vector Regression   :--

       mean absolute error            :--  11583.344547538087

       mean squared error             :--   160811542.7753044

       squar root  mean squared error :--   12681.149110995597

       accuracy                       :-   87.42570960235275

K Nearest Neighbors Regresor :--

       mean absolute error            :--  10447.029400000007

       mean squared error             :--   143358751.24623257

       squar root  mean squared error :--   11973.251490143877

       accuracy                       :-   88.79039067653898

Random Forest Regressor :--

       mean absolute error            :--  5382.992020000025

       mean squared error             :--   46897136.1520224

       squar root  mean squared error :--   6848.148373978356

       accuracy                       :-   96.33298581298052

AdaBoost Regressor :--

       mean absolute error            :--  8058.5556000000015

       mean squared error             :--   86273981.35834625

       squar root  mean squared error :--   9288.378833701081

       accuracy                       :-   93.25400355820945

Gradient Boosting Regressor :--

       mean absolute error            :--  7843.152119035384

       mean squared error             :--   102277207.79570016

       squar root  mean squared error :--   10113.219457507099

       accuracy                       :-   92.0026679074859

Xtreme Gradient Boosting Regressor :--

       mean absolute error            :--  7916.362203125002

       mean squared error             :--   94474686.90564486

       squar root  mean squared error :--   9719.808995327267

       accuracy                       :-   92.61276816404741
```

In [69]:
```python
print('this is the  mean accuracy all models we have used  :--  ',np.array(accuracy).mean()*100)
```

this is the  mean accuracy all models we have used  :--   92.83582734232925

In [70]:
```python
model_names=[]
for name,model in models:
    model_names.append(name)

plt.figure(figsize=(8,8))
plt.plot(model_names,accuracy,marker='o')
plt.grid()
plt.xticks(rotation=90)
plt.show()
```




In [71]:
```python
Random=RandomForestRegressor()
Random.fit(xtrain,ytrain)
ypred=Random.predict(xtest)

print('training  error                      :--  ',Random.score(xtrain,ytrain)*100)
print('testing  error                       :--  ',Random.score(xtest,ytest)*100)
print('square root of mean squared error    :--  ', np.sqrt(mean_squared_error(ytest,ypred)))
print('mean absolute error                  :--  ', mean_absolute_error(ytest,ypred))
```

```
training  error                      :--   98.81498969206113
testing  error                       :--   95.96759194779739
square root of mean squared error    :--   7181.234404696621
mean absolute error                  :--   5614.063820000034
```

In [72]:
```python
accuracies = cross_val_score(estimator = RandomForestRegressor(), X = xtrain, y = ytrain, cv = 5,verbose = 1)
accuracies
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.4s finished
```

Out[72]: array([0.7840862 , 0.88940862, 0.89773132, 0.88858691, 0.93683515])

In [73]:
```python
print('this is the minimum accuracy we can get from this model  :-   ',accuracies.mean()*100)
```

this is the minimum accuracy we can get from this model  :-    87.93296392918835

In [74]:
```python
Random.feature_importances_
```

Out[74]: array([0.92768124, 0.00629134, 0.06602742])

In [75]:
```python
pd.DataFrame(Random.feature_importances_,col,columns=['co-efficient'])
```

Out[75]:

| | co-efficient |
|---|---|
| R&D Spend | 0.927681 |
| Administration | 0.006291 |
| Marketing Spend | 0.066027 |

## Above interpretation tell as about

In [ ]:
```
setting all values fixed if we increase  1 unit of R&D Spend, then it will increase 0.927681 unit  in Profit

setting all values fixed if we increase  1 unit of Administration, then it will increase 0.006291 unit  in Profit

setting all values fixed if we increase  1 unit of Marketing Spend, then it will increase 0.066027 unit  in Profit
```

In [ ]: