Import required Libraries

In [3]:

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

# import sklearn libraries for model building
from sklearn.model_selection import train_test_split, ShuffleSplit, GridSearchCV
from sklearn.linear_model import LinearRegression,Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor

import pickle
import json

import warnings
warnings.filterwarnings('ignore')
```

Loading Dataset

In [4]:

```
df = pd.read_csv(r"C:\Users\sayed\OneDrive\Desktop\TITANIC\50_Startups.csv")
```

In [5]:

```
df.head()
```

Out[5]:

	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 [6]:

```
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
```

Observations :

In [7]:

```
df.describe()
```

Out[7]:

	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

Observations :

1. The mean and median of the above columns are nearly equal so it is a Normally distributed dataset

Missing Value Count

In [8]:

```
df.isnull().sum()
```

Out[8]:

```
R&D Spend          0
Administration     0
Marketing Spend    0
State              0
Profit             0
dtype: int64
```

Observations :

1. All the columns are having zero missing values

Unique Values

In [9]:

```
df["State"].unique()
```

Out[9]:

```
array(['New York', 'California', 'Florida'], dtype=object)
```

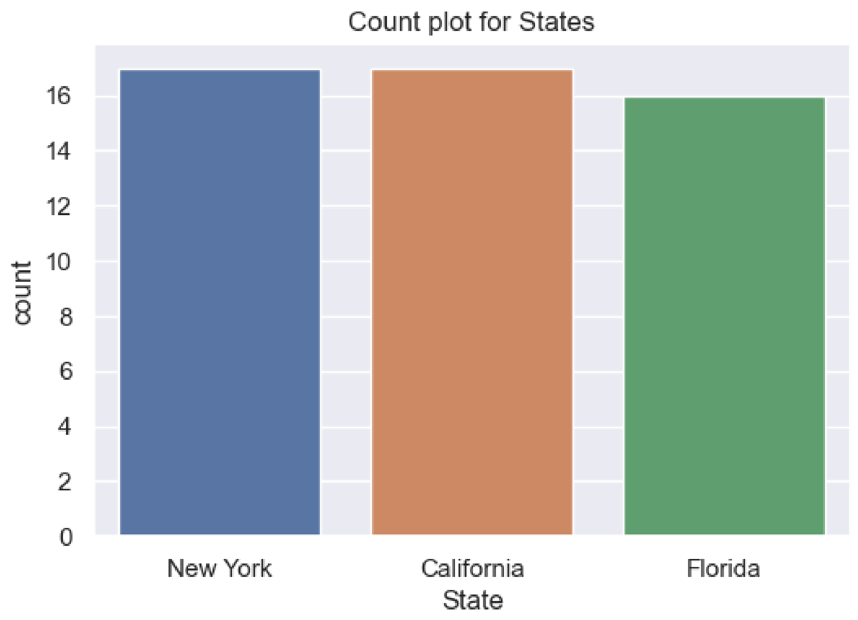
Distribution of Categorical variable

In [10]:

```
plt.figure(figsize = (6,4), dpi= 100)
sns.countplot(data = df, x = "State")
plt.title("Count plot for States")
```

Out[10]:

```
Text(0.5, 1.0, 'Count plot for States')
```



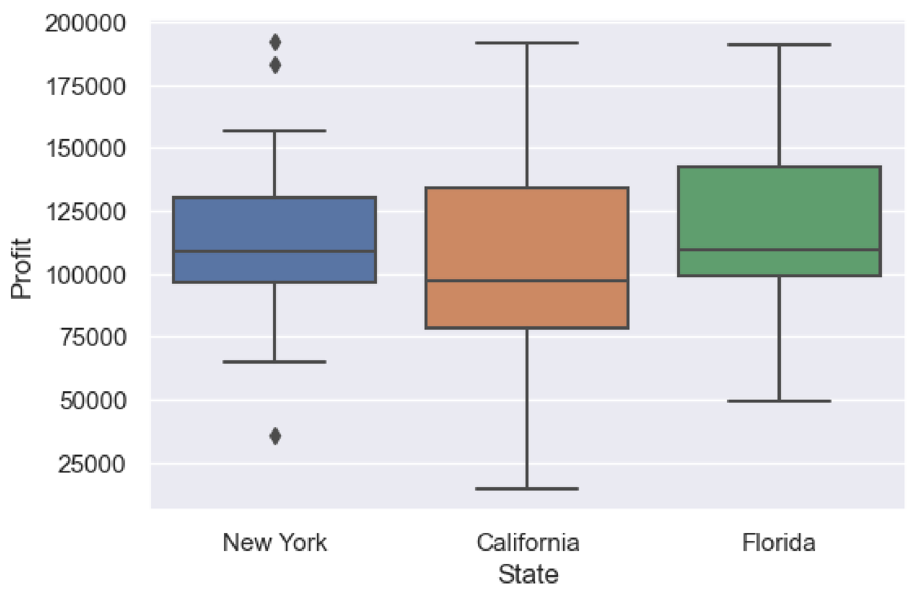
Relationship between categorical variable and target value

In [11]:

```
plt.figure(figsize = (6,4), dpi = 100)
sns.boxplot(data = df, x = "State", y = "Profit")
```

Out[11]:

```
<AxesSubplot:xlabel='State', ylabel='Profit'>
```

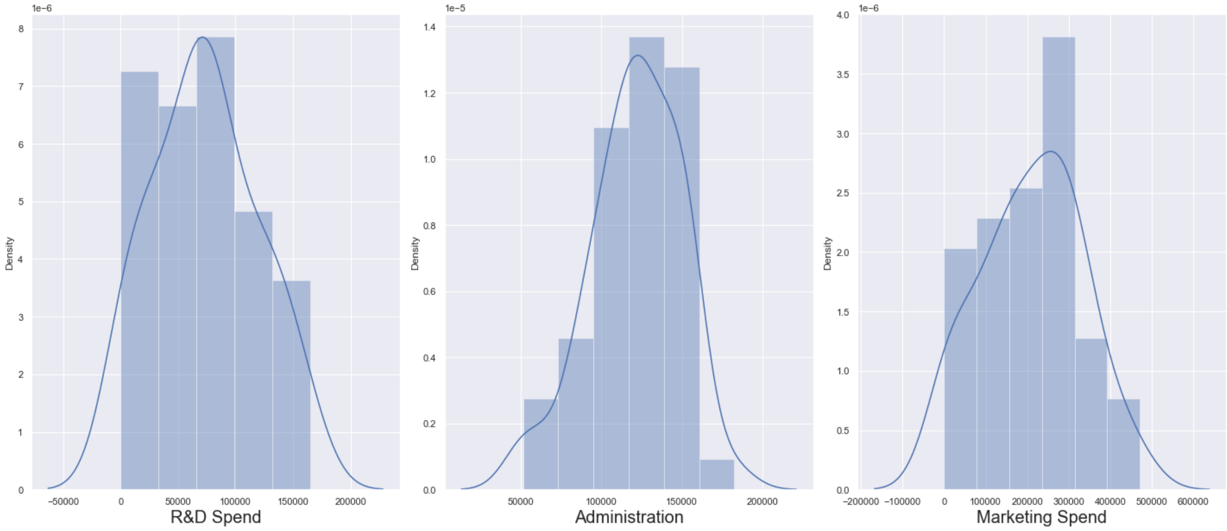


Distribution plot for numerical feature

In [12]:

```
plt.figure(figsize=(20,25))
plotnumber = 1

for column in df:
    if plotnumber<=3 :
        ax = plt.subplot(3,3, plotnumber)
        sns.distplot(df[column])
        plt.xlabel(column,fontsize=20)
    plotnumber+=1
plt.tight_layout()
```



Observations :

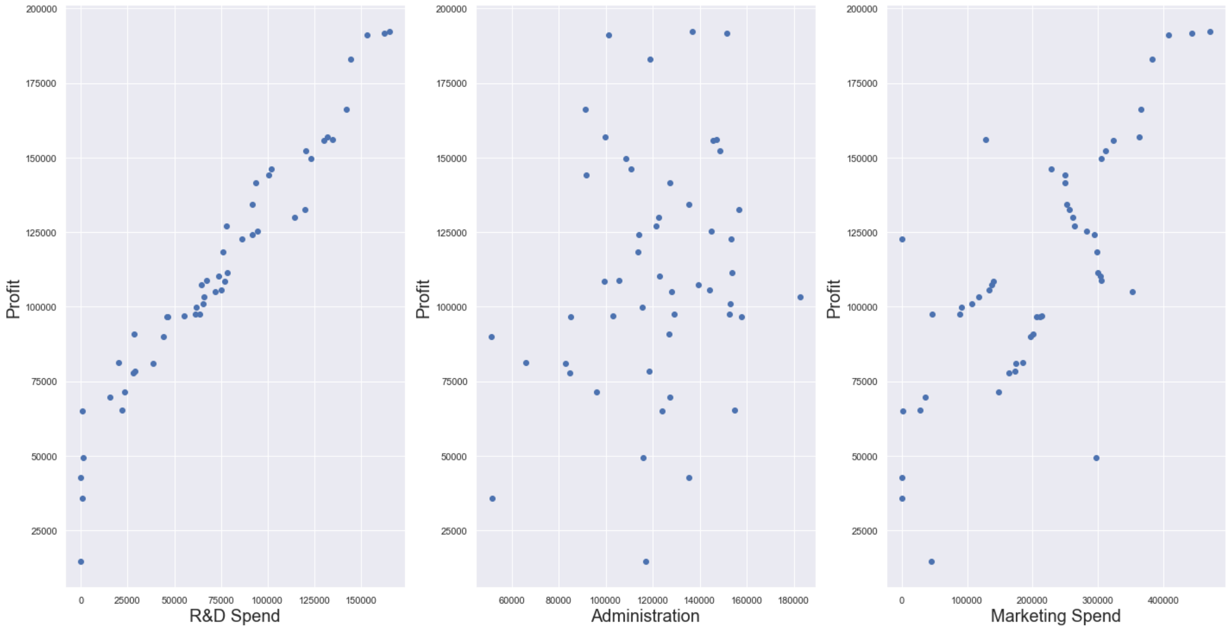
1. From the graph it is clearly visible that datset is Normally distributed

Scatter plot for Numerical and Target values

In [13]:

```
plt.figure(figsize=(20,30), facecolor='white')
plotnumber = 1

for column in df:
    if plotnumber<=3 :
        ax = plt.subplot(3,3,plotnumber)
        plt.scatter(df[column], df.Profit)
        plt.xlabel(column,fontsize=20)
        plt.ylabel('Profit',fontsize=20)
    plotnumber+=1
plt.tight_layout()
```



Observations :

1. From Scatter plot it is clear that all the numeric features are having linear relationship with profit

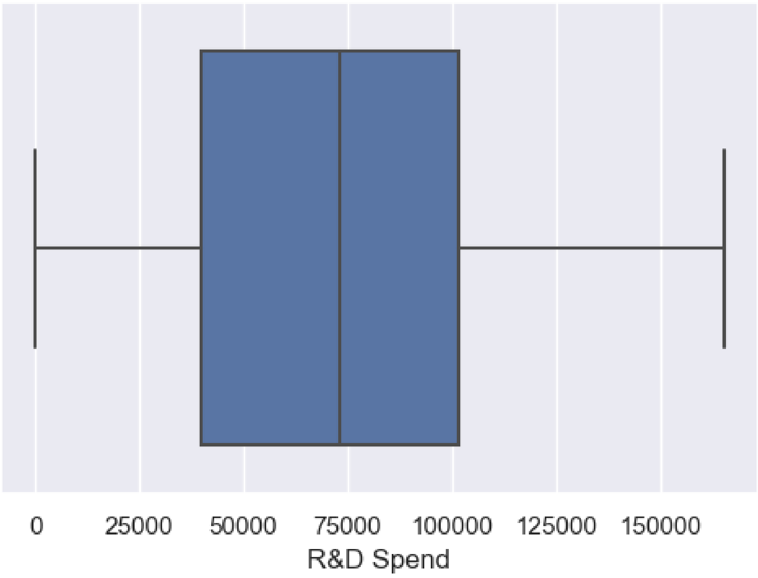
Boxplot for Numerical features

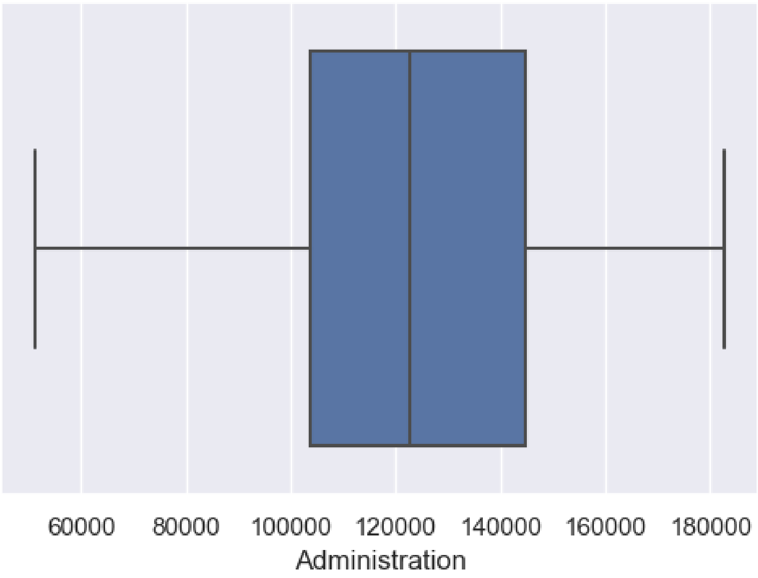
In [14]:

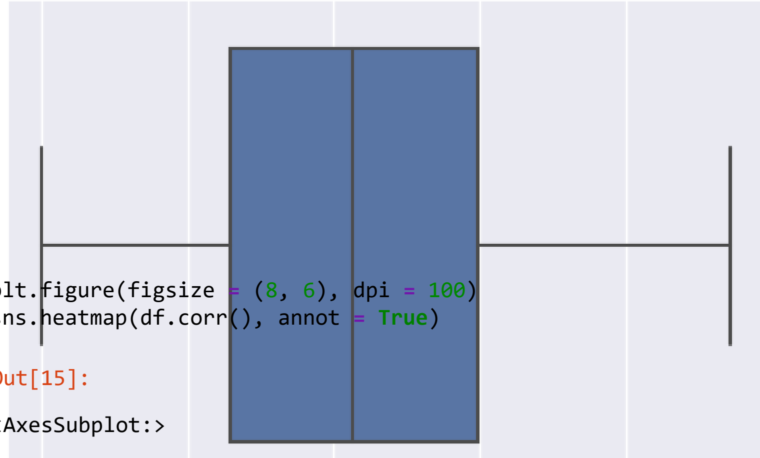
```
plt.figure(figsize = (6,4), dpi = 100)
sns.boxplot(data = df, x = "R&D Spend")
plt.figure(figsize = (6,4), dpi = 100)
sns.boxplot(data = df, x = "Administration")
plt.figure(figsize = (6,4), dpi = 100)
sns.boxplot(data = df, x = "Marketing Spend")
```

Out[14]:

```
<AxesSubplot:xlabel='Marketing Spend'>
```





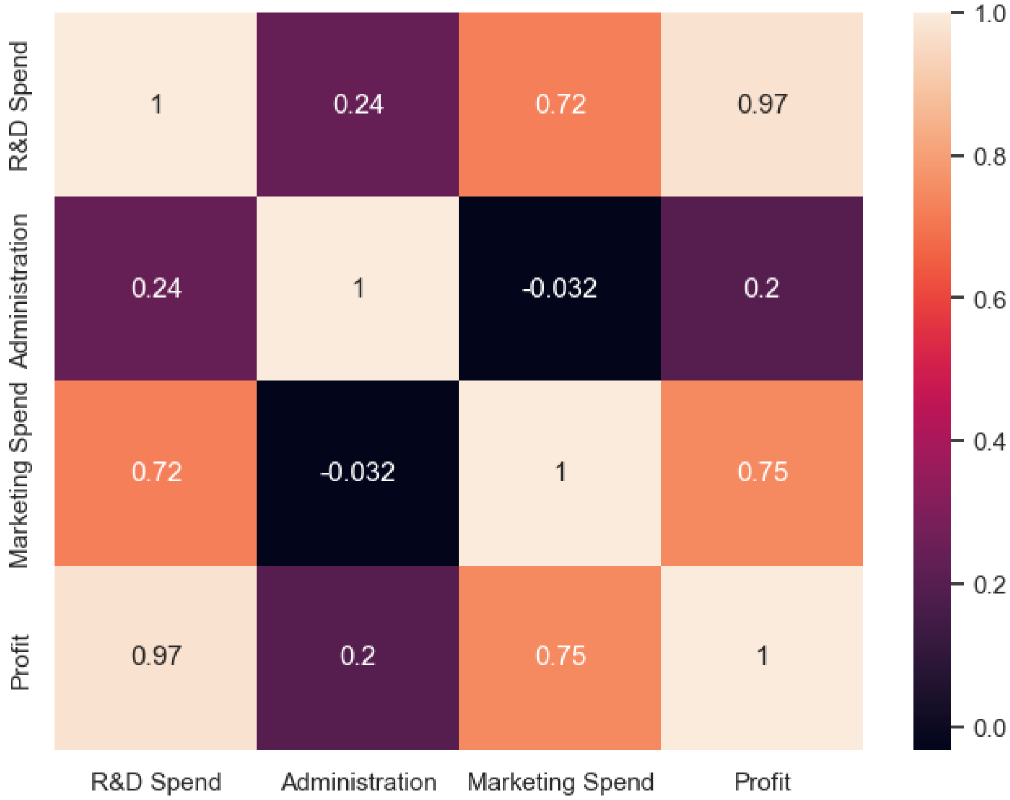


s

```
plt.figure(figsize = (8, 6), dpi = 100)
sns.heatmap(df.corr(), annot = True)
```

Out[15]:

<AxesSubplot:>



Feature Engineering

In [16]:

```
df_state_dummies = pd.get_dummies(df["State"],prefix="State",drop_first=True)
df = pd.concat([df, df_state_dummies], axis = 1)
df.head()
```

Out[16]:

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

Observations :

1. As we are having categorical feature called state column so we need to create dummy variable for that

In [17]:

```
df = df.drop("State", axis = 1)
df.head()
```

Out[17]:

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

Model selection

In [18]:

```
# Separate Dependent and Independent Variables
X = df.drop('Profit',axis=1)
y = df['Profit']
```

In [19]:

```
from sklearn.preprocessing import StandardScaler
```

In [20]:

```
scaler=StandardScaler()
```

In [21]:

```
X = scaler.fit_transform(X)
```

In [22]:

```
# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 50)
```

In [23]:

```
lm_model = LinearRegression()
lm_model.fit(X_train, y_train)
from sklearn.metrics import r2_score
y_hat_lm = lm_model.predict(X_test)
print("r2_score for Linear Regression Model is : ", r2_score(y_test, y_hat_lm))
```

```
r2_score for Linear Regression Model is :  0.9101473536069875
```

In [24]:

```
from sklearn.linear_model import Ridge, Lasso, ElasticNet
```

In [25]:

```
ridge_model = Ridge()
param_grid_ridge = {"alpha" : np.linspace(0, 10, 15),
                    "fit_intercept" : [True, False]}
```

In [26]:

```
grid_model = GridSearchCV(estimator = ridge_model, param_grid = param_grid_ridge, cv = 5, verbose = 3)
grid_model.fit(X_train, y_train)
y_hat_ridge = grid_model.predict(X_test)
```

```
[CV 3/5] END .....alpha=0.0, fit_intercept=True;, score=0.880 total time=   0.0s
[CV 4/5] END .....alpha=0.0, fit_intercept=True;, score=0.981 total time=   0.0s
[CV 5/5] END .....alpha=0.0, fit_intercept=True;, score=0.922 total time=   0.0s
[CV 1/5] END ...alpha=0.0, fit_intercept=False;, score=-2.888 total time=   0.0s
[CV 2/5] END ..alpha=0.0, fit_intercept=False;, score=-16.372 total time=   0.0s
[CV 3/5] END ..alpha=0.0, fit_intercept=False;, score=-27.948 total time=   0.0s
[CV 4/5] END ...alpha=0.0, fit_intercept=False;, score=-7.145 total time=   0.0s
[CV 5/5] END ...alpha=0.0, fit_intercept=False;, score=-9.689 total time=   0.0s
[CV 1/5] END alpha=0.7142857142857143, fit_intercept=True;, score=0.915 total time=   0.0s
[CV 2/5] END alpha=0.7142857142857143, fit_intercept=True;, score=0.965 total time=   0.0s
[CV 3/5] END alpha=0.7142857142857143, fit_intercept=True;, score=0.883 total time=   0.0s
[CV 4/5] END alpha=0.7142857142857143, fit_intercept=True;, score=0.980 total time=   0.0s
[CV 5/5] END alpha=0.7142857142857143, fit_intercept=True;, score=0.929 total time=   0.0s
[CV 1/5] END alpha=0.7142857142857143, fit_intercept=False;, score=-3.002 total time=   0.0s
[CV 2/5] END alpha=0.7142857142857143, fit_intercept=False;, score=-15.288 total time=   0.0s
[CV 3/5] END alpha=0.7142857142857143, fit_intercept=False;, score=-27.265 total time=   0.0s
[CV 4/5] END alpha=0.7142857142857143, fit_intercept=False;, score=-7.236 total time=   0.0s
[CV 5/5] END alpha=0.7142857142857143, fit_intercept=False;, score=-9.608 total time=   0.0s
[CV 1/5] END alpha=1.4285714285714286, fit_intercept=True;, score=0.904 total time=   0.0s
[CV 2/5] END alpha=1.4285714285714286, fit_intercept=True;, score=0.962 total time=   0.0s
```

In [27]:

```
lasso_model = Lasso()
param_grid_lasso = {"alpha" : np.linspace(0, 10, 15),
                   "fit_intercept" : [True, False],
                   "selection" : ["cyclic", "random"]}
```

In [28]:

```
grid_model_lasso = GridSearchCV(estimator = lasso_model, param_grid= param_grid_lasso, cv = 5, verbose = 3)
grid_model_lasso.fit(X_train, y_train)
y_hat_lasso = grid_model_lasso.predict(X_test)
```

```
Fitting 5 folds for each of 60 candidates, totalling 300 fits
[CV 1/5] END alpha=0.0, fit_intercept=True, selection=cyclic;, score=0.925 total time=   0.0s
[CV 2/5] END alpha=0.0, fit_intercept=True, selection=cyclic;, score=0.964 total time=   0.0s
[CV 3/5] END alpha=0.0, fit_intercept=True, selection=cyclic;, score=0.880 total time=   0.0s
[CV 4/5] END alpha=0.0, fit_intercept=True, selection=cyclic;, score=0.981 total time=   0.0s
[CV 5/5] END alpha=0.0, fit_intercept=True, selection=cyclic;, score=0.922 total time=   0.0s
[CV 1/5] END alpha=0.0, fit_intercept=True, selection=random;, score=0.925 total time=   0.0s
[CV 2/5] END alpha=0.0, fit_intercept=True, selection=random;, score=0.964 total time=   0.0s
[CV 3/5] END alpha=0.0, fit_intercept=True, selection=random;, score=0.880 total time=   0.0s
[CV 4/5] END alpha=0.0, fit_intercept=True, selection=random;, score=0.981 total time=   0.0s
[CV 5/5] END alpha=0.0, fit_intercept=True, selection=random;, score=0.922 total time=   0.0s
[CV 1/5] END alpha=0.0, fit_intercept=False, selection=cyclic;, score=-2.888 total time=   0.0s
[CV 2/5] END alpha=0.0, fit_intercept=False, selection=cyclic;, score=-16.372 total time=   0.0
s
[CV 3/5] END alpha=0.0, fit_intercept=False, selection=cyclic;, score=-27.948 total time=   0.0
s
[CV 4/5] END alpha=0.0, fit_intercept=False, selection=cyclic;, score=-7.145 total time=   0.0s
[CV 5/5] END alpha=0.0, fit_intercept=False, selection=cyclic;, score=-9.689 total time=   0.0s
[CV 1/5] END alpha=0.0, fit_intercept=False, selection=random;, score=-2.888 total time=   0.0s
```

In [29]:

```
elastic_model = ElasticNet()
para_grid_elastic = {"alpha" : np.linspace(0, 10, 15),
                    "l1_ratio" : np.linspace(0, 1, 15),
                    "fit_intercept" : [True, False],
                    "selection" : ["cyclic", "random"]}
```

In [30]:

```
grid_model_elastic = GridSearchCV(estimator = elastic_model, param_grid= para_grid_elastic, cv = 5, verbose = 3)
grid_model_elastic.fit(X_train, y_train)
```

```
re=0.880 total time=   0.0s
[CV 4/5] END alpha=0.0, fit_intercept=True, l1_ratio=0.3571428571428571, selection=cyclic;, sco
re=0.981 total time=   0.0s
[CV 5/5] END alpha=0.0, fit_intercept=True, l1_ratio=0.3571428571428571, selection=cyclic;, sco
re=0.922 total time=   0.0s
[CV 1/5] END alpha=0.0, fit_intercept=True, l1_ratio=0.3571428571428571, selection=random;, sco
re=0.925 total time=   0.0s
[CV 2/5] END alpha=0.0, fit_intercept=True, l1_ratio=0.3571428571428571, selection=random;, sco
re=0.964 total time=   0.0s
[CV 3/5] END alpha=0.0, fit_intercept=True, l1_ratio=0.3571428571428571, selection=random;, sco
re=0.880 total time=   0.0s
[CV 4/5] END alpha=0.0, fit_intercept=True, l1_ratio=0.3571428571428571, selection=random;, sco
re=0.981 total time=   0.0s
[CV 5/5] END alpha=0.0, fit_intercept=True, l1_ratio=0.3571428571428571, selection=random;, sco
re=0.922 total time=   0.0s
[CV 1/5] END alpha=0.0, fit_intercept=True, l1_ratio=0.42857142857142855, selection=cyclic;, sc
ore=0.925 total time=   0.0s
[CV 2/5] END alpha=0.0, fit_intercept=True, l1_ratio=0.42857142857142855, selection=cyclic;, sc
ore=0.964 total time=   0.0s
[CV 3/5] END alpha=0.0, fit_intercept=True, l1_ratio=0.42857142857142855, selection=cyclic;, sc
```

In [31]:

```
y_hat_elastic = grid_model_elastic.predict(X_test)
```

In [32]:

```
print("r2_score for Ridge Model is : ", r2_score(y_test, y_hat_ridge))
```

```
r2_score for Ridge Model is :  0.9101473536069874
```

In [33]:

```
print("r2_score for Ridge Model is : ", r2_score(y_test, y_hat_lasso))
```

```
r2_score for Ridge Model is :  0.9103811708859249
```

In [34]:

```
print("r2_score for Ridge Model is : ", r2_score(y_test, y_hat_elastic))
```

```
r2_score for Ridge Model is :  0.9103041588817918
```

In [35]:

```
decision_model = DecisionTreeRegressor()
param_grid_decision = {"criterion" : ["squared_error", "friedman_mse", "absolute_error", "poisson"],
             "splitter" : ["best", "random"],
             "max_features" : ["auto", "sqrt", "log2"]}
```

In [36]:

```
grid_decision = GridSearchCV(estimator = decision_model, param_grid=param_grid_decision)
grid_decision.fit(X_train, y_train)
y_hat_decision = grid_decision.predict(X_test)
```

In [37]:

```
print("r2_score for Ridge Model is : ", r2_score(y_test, y_hat_decision))
```

```
r2_score for Ridge Model is :  0.8593809123149713
```

In [38]:

```
random_model = RandomForestRegressor()
para_grid_random = {'n_estimators' : [int(x) for x in np.linspace(start = 100, stop = 400, num = 100)],
                'max_features': ['auto', 'sqrt'],
                'max_depth':[int(x) for x in np.linspace(10, 31, num = 11)],
                'min_samples_leaf':[1, 2]}
```

In [39]:

```
print("r2_score for Linear Regression Model is : ", r2_score(y_test, y_hat_lm))
print("r2_score for Ridge Model is : ", r2_score(y_test, y_hat_ridge))
print("r2_score for Lasso Model is : ", r2_score(y_test, y_hat_lasso))
print("r2_score for ElasticNet Model is : ", r2_score(y_test, y_hat_elastic))
print("r2_score for Decision Tree Model is : ", r2_score(y_test, y_hat_decision))
```

```
r2_score for Linear Regression Model is :  0.9101473536069875
r2_score for Ridge Model is :  0.9101473536069874
r2_score for Lasso Model is :  0.9103811708859249
r2_score for ElasticNet Model is :  0.9103041588817918
r2_score for Decision Tree Model is :  0.8593809123149713
```

Observations

In [40]:

```
# From above 5 models hieghst r2_score we got from linear model, so for model building we used linear regression
```

Model Building

In [43]:

```
model = LinearRegression()
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
model.score(X_test,y_test)
```

Out[43]:

```
0.9101473536069875
```

Summary

1. In the above dataset total we have 5 columns & 50 rows
2. In which 1 is categorical & others are continuous columns
3. In the dataset we dont have any null values
4. From the description for all columns mean and median is almost same so we consider data is normally distributed
5. From scatter plot all the continuous column having linear relationship
6. So we test our dataset on linear Regression, ridge model, Lasso & Elastic net, Decision tree regressor. We got highest r2_score from linear regression which is 91.0147% so we select Linear regression to build our model.