```
#import the basics libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

```
df=pd.read_csv("/content/50_Startups.csv")
```

```
df
```

18	91749.16	114175.79	294919.57	Florida	124266.90
19	86419.70	153514.11	0.00	New York	122776.86
20	76253.86	113867.30	298664.47	California	118474.03
21	78389.47	153773.43	299737.29	New York	111313.02
22	73994.56	122782.75	303319.26	Florida	110352.25
23	67532.53	105751.03	304768.73	Florida	108733.99
24	77044.01	99281.34	140574.81	New York	108552.04
25	64664.71	139553.16	137962.62	California	107404.34
26	75328.87	144135.98	134050.07	Florida	105733.54
27	72107.60	127864.55	353183.81	New York	105008.31
28	66051.52	182645.56	118148.20	Florida	103282.38
29	65605.48	153032.06	107138.38	New York	101004.64
30	61994.48	115641.28	91131.24	Florida	99937.59
31	61136.38	152701.92	88218.23	New York	97483.56

```
df.head()
```

	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

```
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.describe()
```

	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

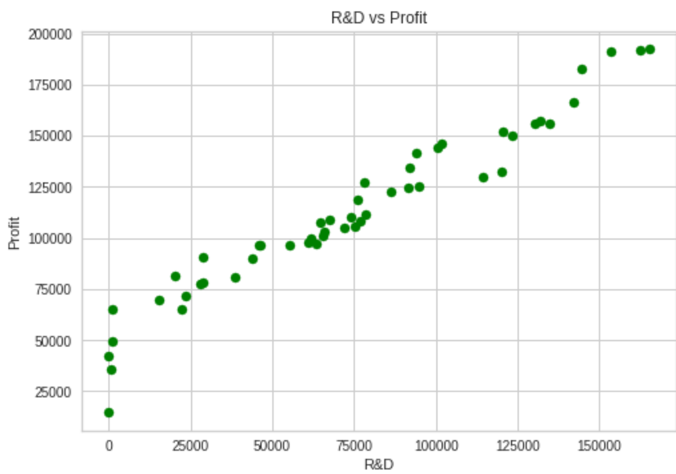
```
df.corr()
```

	R&D Spend	Administration	Marketing Spend	Profit
R&D Spend	1.000000	0.241955	0.724248	0.972900

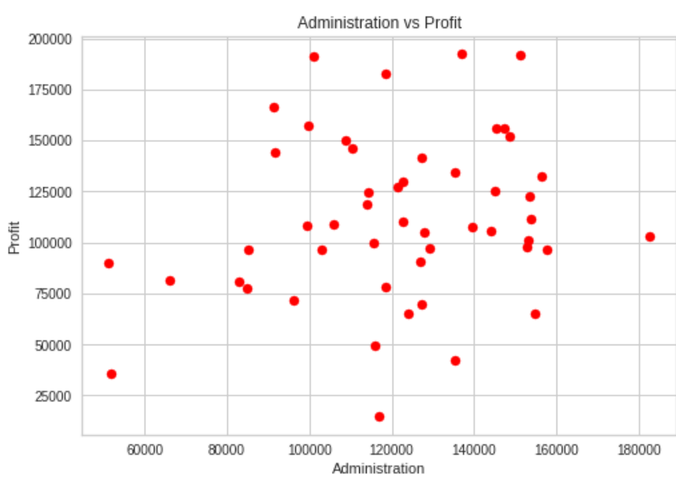
```
df.isnull().sum()
```

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

```
#Plot R&D vs Profit...........
x1 = df.iloc[:, 0].values
y1 = df.iloc[:, -1].values
plt.scatter(x1,y1,color='Green',s=50)
plt.xlabel('R&D')
plt.ylabel('Profit')
plt.title('R&D vs Profit')
plt.show()
```

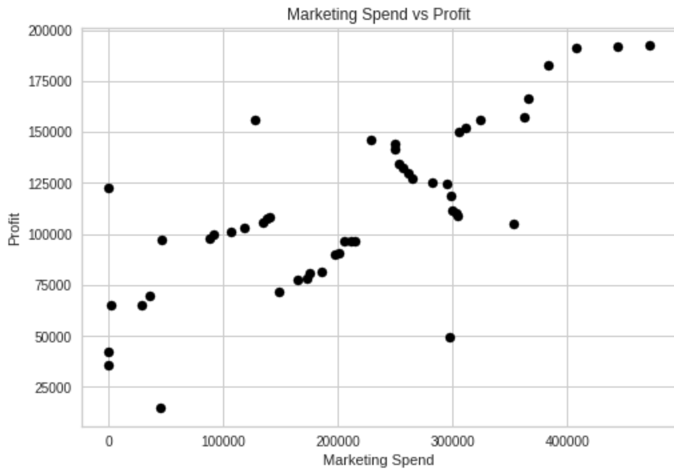



```
#Plot Administration vs Profit
x1 = df.iloc[:, 1].values
y1 = df.iloc[:, -1].values
plt.scatter(x1,y1,color='Red',s=50)
plt.xlabel('Administration')
plt.ylabel('Profit')
plt.title('Administration vs Profit')
plt.show()
```

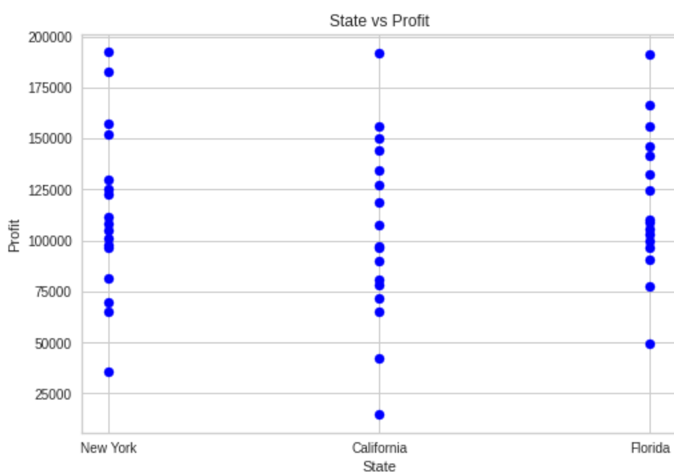



```
#Plot Marketing Spend vs Profit
x1 = df.iloc[:, 2].values
y1 = df.iloc[:, -1].values
plt.scatter(x1,y1,color='Black',s=50)
```

```
plt.xlabel('Marketing Spend')
plt.ylabel('Profit')
plt.title('Marketing Spend vs Profit')
plt.show()
```




```
#High correlation between Marketing Spend and Profit.
#Plot State vs Profit
x1 = df.iloc[:, 3].values
y1 = df.iloc[:, -1].values
plt.scatter(x1,y1,color='Blue',s=50)
plt.xlabel('State')
plt.ylabel('Profit')
plt.title('State vs Profit')
plt.show()
```

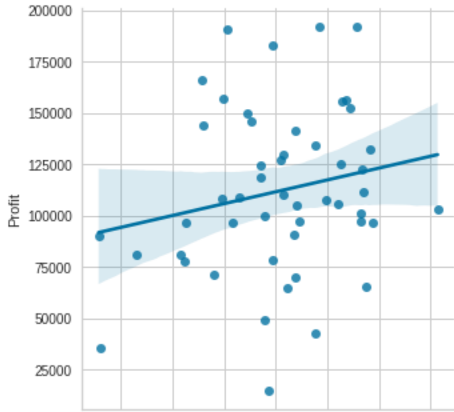



```
df.head()
```

	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

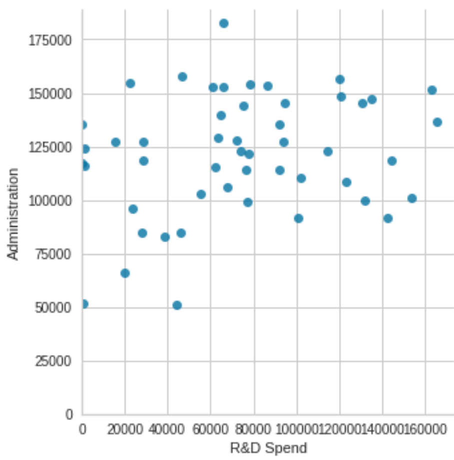
```
# Recommended way
sns.lmplot(x='Administration', y='Profit', data=df)

# Alternative way
# sns.lmplot(x=df.Administration, y=df.Profit)
```

<seaborn.axisgrid.FacetGrid at 0x7f14058b8c10>



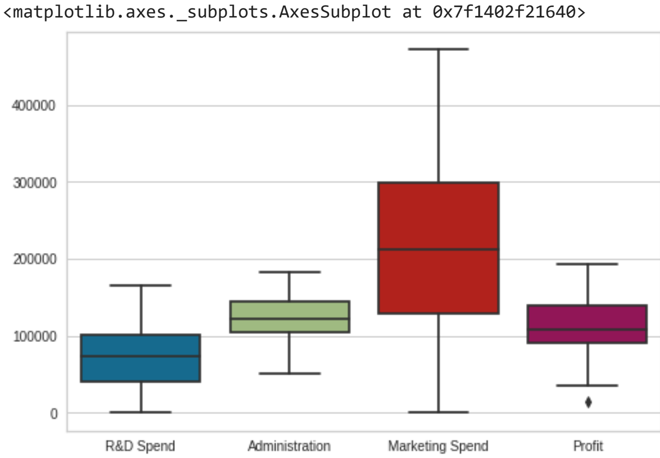

```
# Plot using Seaborn
sns.lmplot(x='R&D Spend', y='Administration', data=df, fit_reg=False)

# Tweak using Matplotlib
plt.ylim(0, None)
plt.xlim(0, None)
```

(0.0, 173616.66)



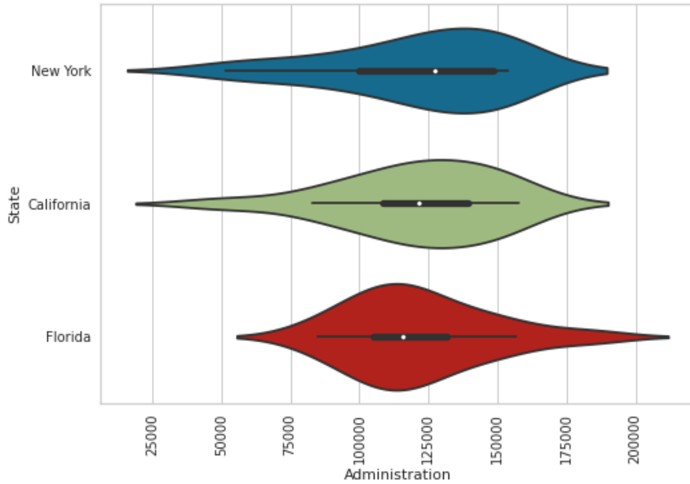

```
# Boxplot
sns.boxplot(data=df)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f1402f21640>




```
# Set theme
sns.set_style('whitegrid')

# Violin plot
sns.violinplot(x='Administration', y='State', data=df)
plt.xticks(rotation=90)
```

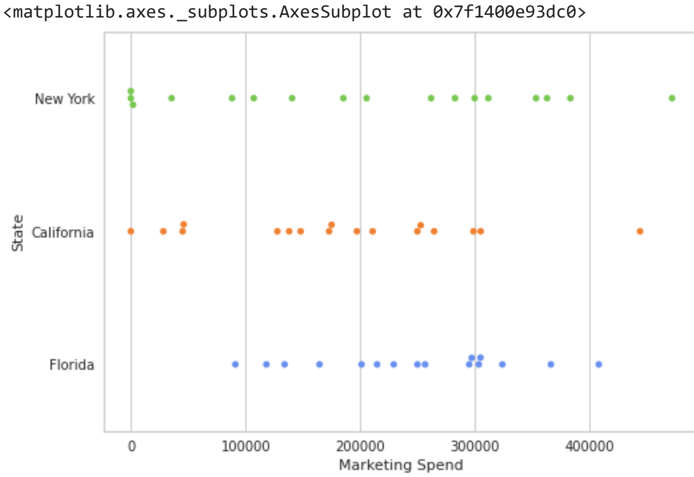
```
(array([     0.,  25000.,  50000.,  75000., 100000., 125000., 150000.,
        175000., 200000., 225000.]),
 <a list of 10 Text major ticklabel objects>)
```




```
pkmn_type_colors = ['#78C850',  # Grass
                    '#F08030',  # Fire
                    '#6890F0',  # Water
                    '#A8B820',  # Bug
                    '#A8A878',  # Normal
                    '#A040A0',  # Poison
                    '#F8D030',  # Electric
                    '#E0C068',  # Ground
                    '#EE99AC',  # Fairy
                    '#C03028',  # Fighting
                    '#F85888',  # Psychic
                    '#B8A038',  # Rock
                    '#705898',  # Ghost
                    '#98D8D8',  # Ice
                    '#7038F8',  # Dragon
                   ]
```

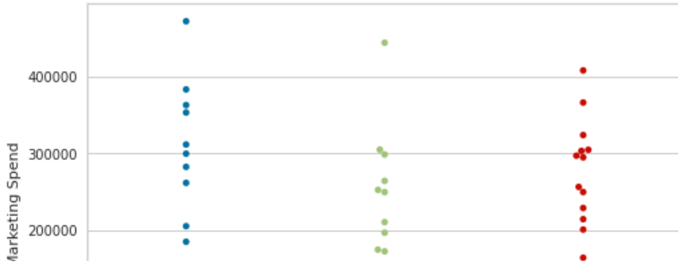
```
# Swarm plot with Pokemon color palette
sns.swarmplot(x='Marketing Spend', y='State', data=df,
              palette=pkmn_type_colors)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1400e93dc0>
```



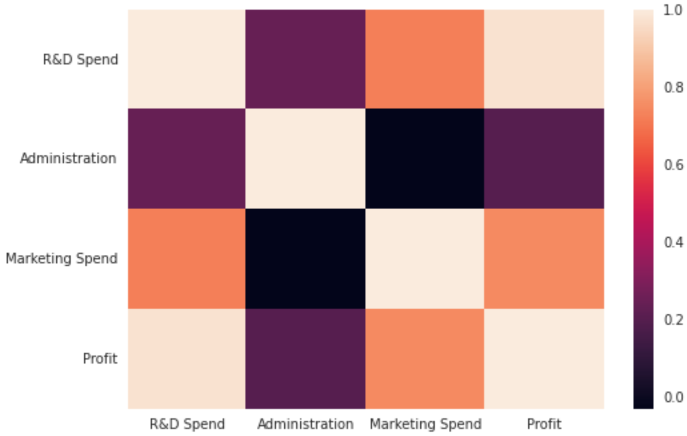

```
# Swarmplot with melted_df
sns.swarmplot(x='State', y='Marketing Spend', data=df)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f1400e11370>



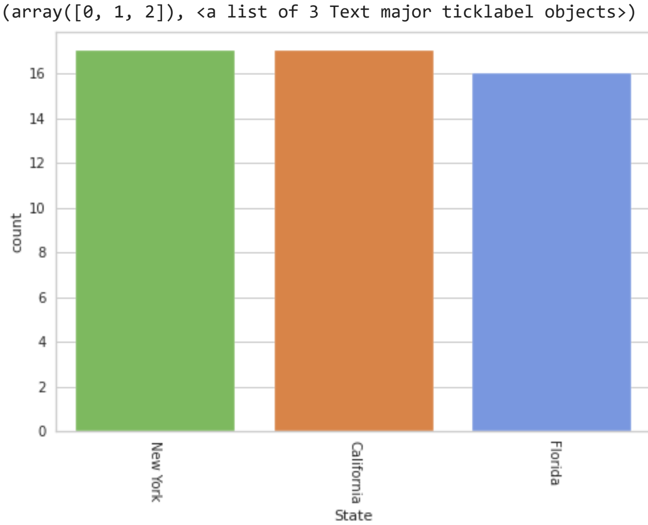

```
# Calculate correlations
corr = df.corr()

# Heatmap
sns.heatmap(corr)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f1400dcaaf0>



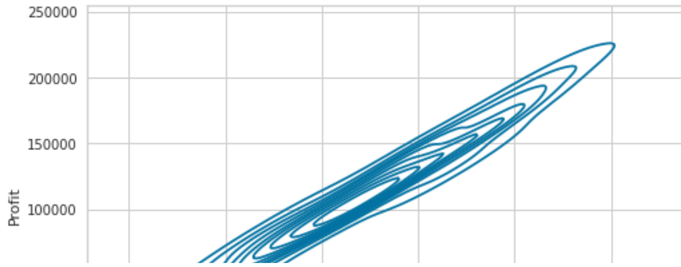

```
# Count Plot (a.k.a. Bar Plot)
sns.countplot(x='State', data=df, palette=pkmn_type_colors)

# Rotate x-labels
plt.xticks(rotation=-90)
```

(array([0, 1, 2]), <a list of 3 Text major ticklabel objects>)



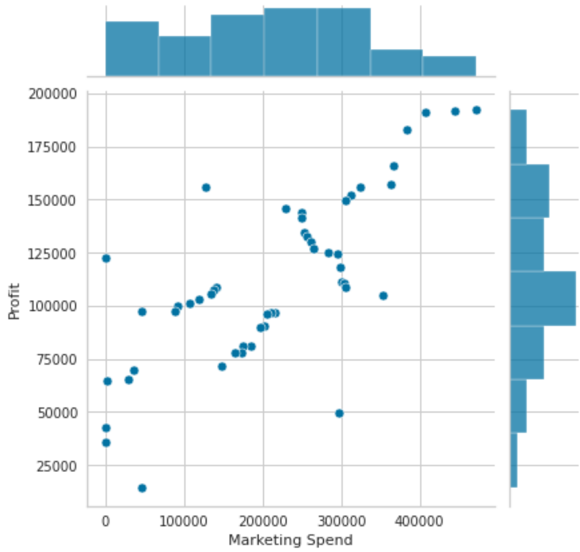

```
sns.kdeplot(df["R&D Spend"], df["Profit"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1400cb7730>
```




```
# Joint Distribution Plot
sns.jointplot(x='Marketing Spend', y='Profit', data=df)
```

```
<seaborn.axisgrid.JointGrid at 0x7f1402eb44f0>
```



>Seperation of dependent and independent variables bold text

```
X = df.iloc[:, :-1].values
print(X)
```

```
[[165349.2 136897.8 471784.1 'New York']
 [162597.7 151377.59 443898.53 'California']
 [153441.51 101145.55 407934.54 'Florida']
 [144372.41 118671.85 383199.62 'New York']
 [142107.34 91391.77 366168.42 'Florida']
 [131876.9 99814.71 362861.36 'New York']
 [134615.46 147198.87 127716.82 'California']
 [130298.13 145530.06 323876.68 'Florida']
 [120542.52 148718.95 311613.29 'New York']
 [123334.88 108679.17 304981.62 'California']
 [101913.08 110594.11 229160.95 'Florida']
 [100671.96 91790.61 249744.55 'California']
 [93863.75 127320.38 249839.44 'Florida']
 [91992.39 135495.07 252664.93 'California']
 [119943.24 156547.42 256512.92 'Florida']
 [114523.61 122616.84 261776.23 'New York']
 [78013.11 121597.55 264346.06 'California']
 [94657.16 145077.58 282574.31 'New York']
 [91749.16 114175.79 294919.57 'Florida']
 [86419.7 153514.11 0.0 'New York']
 [76253.86 113867.3 298664.47 'California']
 [78389.47 153773.43 299737.29 'New York']
 [73994.56 122782.75 303319.26 'Florida']
 [67532.53 105751.03 304768.73 'Florida']
 [77044.01 99281.34 140574.81 'New York']
 [64664.71 139553.16 137962.62 'California']
 [75328.87 144135.98 134050.07 'Florida']
 [72107.6 127864.55 353183.81 'New York']
```

```
 [66051.52 182645.56 118148.2 'Florida']
 [65605.48 153032.06 107138.38 'New York']
 [61994.48 115641.28 91131.24 'Florida']
 [61136.38 152701.92 88218.23 'New York']
 [63408.86 129219.61 46085.25 'California']
 [55493.95 103057.49 214634.81 'Florida']
 [46426.07 157693.92 210797.67 'California']
 [46014.02 85047.44 205517.64 'New York']
 [28663.76 127056.21 201126.82 'Florida']
 [44069.95 51283.14 197029.42 'California']
 [20229.59 65947.93 185265.1 'New York']
 [38558.51 82982.09 174999.3 'California']
 [28754.33 118546.05 172795.67 'California']
 [27892.92 84710.77 164470.71 'Florida']
 [23640.93 96189.63 148001.11 'California']
 [15505.73 127382.3 35534.17 'New York']
 [22177.74 154806.14 28334.72 'California']
 [1000.23 124153.04 1903.93 'New York']
 [1315.46 115816.21 297114.46 'Florida']
 [0.0 135426.92 0.0 'California']
 [542.05 51743.15 0.0 'New York']
 [0.0 116983.8 45173.06 'California']]
```

```
y = df.iloc[:, 4].values
print(y)
```

```
[192261.83 191792.06 191050.39 182901.99 166187.94 156991.12 156122.51
 155752.6  152211.77 149759.96 146121.95 144259.4  141585.52 134307.35
 132602.65 129917.04 126992.93 125370.37 124266.9  122776.86 118474.03
 111313.02 110352.25 108733.99 108552.04 107404.34 105733.54 105008.31
 103282.38 101004.64  99937.59  97483.56  97427.84  96778.92  96712.8
  96479.51  90708.19  89949.14  81229.06  81005.76  78239.91  77798.83
  71498.49  69758.98  65200.33  64926.08  49490.75  42559.73  35673.41
  14681.4 ]
```

```
# Encoding categorical data
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
X[:, 3] = labelencoder.fit_transform(X[:, 3])
```

```
# Avoiding the Dummy Variable Trap
X = X[:, 1:]
```

```
print(X)
```

```
[[136897.8 471784.1 2]
 [151377.59 443898.53 0]
 [101145.55 407934.54 1]
 [118671.85 383199.62 2]
 [91391.77 366168.42 1]
 [99814.71 362861.36 2]
 [147198.87 127716.82 0]
 [145530.06 323876.68 1]
 [148718.95 311613.29 2]
 [108679.17 304981.62 0]
 [110594.11 229160.95 1]
 [91790.61 249744.55 0]
 [127320.38 249839.44 1]
 [135495.07 252664.93 0]
 [156547.42 256512.92 1]
 [122616.84 261776.23 2]
 [121597.55 264346.06 0]
 [145077.58 282574.31 2]
 [114175.79 294919.57 1]
 [153514.11 0.0 2]
 [113867.3 298664.47 0]
 [153773.43 299737.29 2]
 [122782.75 303319.26 1]
 [105751.03 304768.73 1]
 [99281.34 140574.81 2]
 [139553.16 137962.62 0]
 [144135.98 134050.07 1]
 [127864.55 353183.81 2]
 [182645.56 118148.2 1]
 [153032.06 107138.38 2]
 [115641.28 91131.24 1]
```

```
 [152701.92 88218.23 2]
 [129219.61 46085.25 0]
 [103057.49 214634.81 1]
 [157693.92 210797.67 0]
 [85047.44 205517.64 2]
 [127056.21 201126.82 1]
 [51283.14 197029.42 0]
 [65947.93 185265.1 2]
 [82982.09 174999.3 0]
 [118546.05 172795.67 0]
 [84710.77 164470.71 1]
 [96189.63 148001.11 0]
 [127382.3 35534.17 2]
 [154806.14 28334.72 0]
 [124153.04 1903.93 2]
 [115816.21 297114.46 1]
 [135426.92 0.0 0]
 [51743.15 0.0 2]
 [116983.8 45173.06 0]]
```

```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

BUILDING OF MODEL

```
# Fitting Multiple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

```
LinearRegression()
```

```
# Predicting the Test set results
y_pred = regressor.predict(X_test)
```

```
#evaluate the model
from sklearn.metrics import r2_score
```

```
r2_score(y_test,y_pred)
```

```
0.3161625677198352
```

Evaluate the result:-

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predicting the Test set results
y_pred = regressor.predict(X_test)
```

```
#evaluate the model
from sklearn.metrics import r2_score
```

```
r2_score(y_test,y_pred)
```

```
0.3161625677198352
```

```
df.head()
```

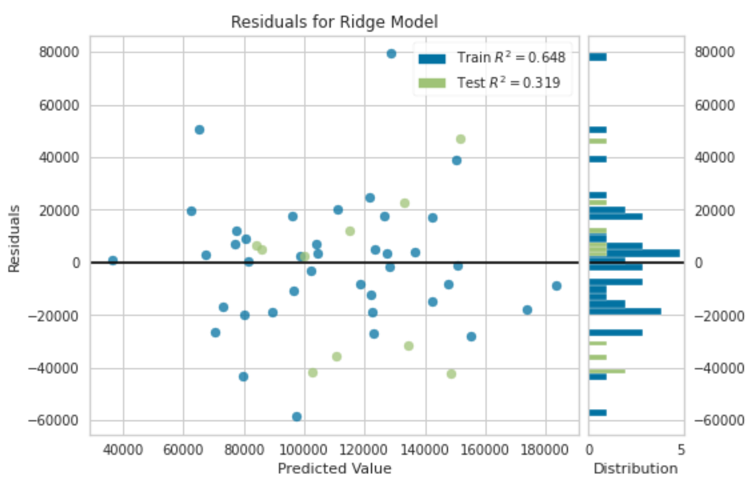
	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83

```
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split

from yellowbrick.datasets import load_concrete
from yellowbrick.regressor import ResidualsPlot

# Load a regression dataset
X, y =

# Create the train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Instantiate the linear model and visualizer
model = Ridge()
visualizer = ResidualsPlot(model)

visualizer.fit(X_train, y_train)  # Fit the training data to the visualizer
visualizer.score(X_test, y_test)  # Evaluate the model on the test data
visualizer.show()                 # Finalize and render the figure
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-45-fa09cf5dd279> in <module>
      6
      7 # Load a regression dataset
----> 8 X, y = df()
      9
     10 # Create the train and test data

TypeError: 'DataFrame' object is not callable
```

SEARCH STACK OVERFLOW

```
# Instantiate the linear model and visualizer # Insta
model = Ridge()
visualizer = ResidualsPlot(model)

visualizer.fit(X_train, y_train)  # Fit the training data to the visualizer
visualizer.score(X_test, y_test)  # Evaluate the model on the test data
g = visualizer.poof()             # Draw/show/poof the data
```




```
from sklearn.metrics import mean_absolute_error

mean_absolute_error(y_test,y_pred)
```

    24725.68122329431

```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test,y_pred)
```

    874553943.1239169

```
np.sqrt(mean_squared_error(y_test,y_pred))
```

```
29572.858217019148
```

```
from sklearn.metrics import r2_score
```

```
r2_score(y_test,y_pred)
```

```
0.3161625677198352
```

✓ 0s completed at 18:44