**Exposys Data Labs**

**Data Science Internship**

Arisha Akhtar

**Importing Libraries**

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

**Data Loading**

```
data=pd.read_csv('/content/50_Startups.csv')
data.head(5)
```

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

This dataset appears to represent financial information for 50 startup companies.

R&D Spend: This column likely represents the amount of money each startup has spent on research and development (R&D) activities. R&D spending typically includes expenses related to creating new products, improving existing ones, or conducting scientific research to support innovation.

Administration: This column probably indicates the administration or operational expenses of each startup.

Marketing Spend: This column most likely represents the amount of money each startup has allocated towards marketing activities.

Profit: This column presumably indicates the profit earned by each startup. Profit is the amount of money remaining after subtracting expenses (such as R&D, administration, and marketing spend) from total revenue. It's a key metric for assessing the financial performance and viability of a business.

```
data.shape[0]
```

```
50
```

```
data.columns
```

```
Index(['R&D Spend', 'Administration', 'Marketing Spend', 'Profit'], dtype='object')
```

```
data.isnull().sum()
```

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

**No Null Values present in the dataset**

**Descriptive Statistics**

```
summary_stats = data.describe()

print(summary_stats)
```

```
            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
```

**R&D Spend:**

Mean:73,721.62: On average, startup companies spend approximately 73,721.62 on research and development.

Standard Deviation: 45,902.26: The expenditure on R&D varies by approximately 45,902.26 around the mean.

Minimum:0.00: The minimum R&D spend in the dataset is 0.00, indicating that some companies did not invest anything in R&D.

Maximum:165,349.20: The maximum R&D spend is 165,349.20, suggesting significant investment in innovation by some companies.

**Administration:**

Mean: 121,344.64: The average administration cost for these startups is approximately 121,344.64.

Standard Deviation: 28,017.80: Administration costs have relatively less variability compared to R&D spending.

Minimum: 51,283.14: The lowest administration cost is 51,283.14.

Maximum: 182,645.56: The highest administration cost is 182,645.56.

**Marketing Spend:**

Mean: 211,025.10: The average marketing expenditure is 211,025.10.

Standard Deviation: 122,290.31: Marketing spend varies considerably across the dataset, with a standard deviation of 122,290.31.

Minimum: 0.00: Some companies have not allocated any funds to marketing activities.

Maximum: 471,784.10: The maximum marketing spend is 471,784.10, indicating significant investment in marketing by certain companies.

**Profit:**

Mean: 112,012.64: The average profit earned by these startups is approximately 112,012.64.

Standard Deviation: 40,306.18: Profitability varies across the dataset, with a standard deviation of 40,306.18.

Minimum: 14,681.40: The lowest profit recorded is 14,681.40, indicating potential financial challenges for some companies.

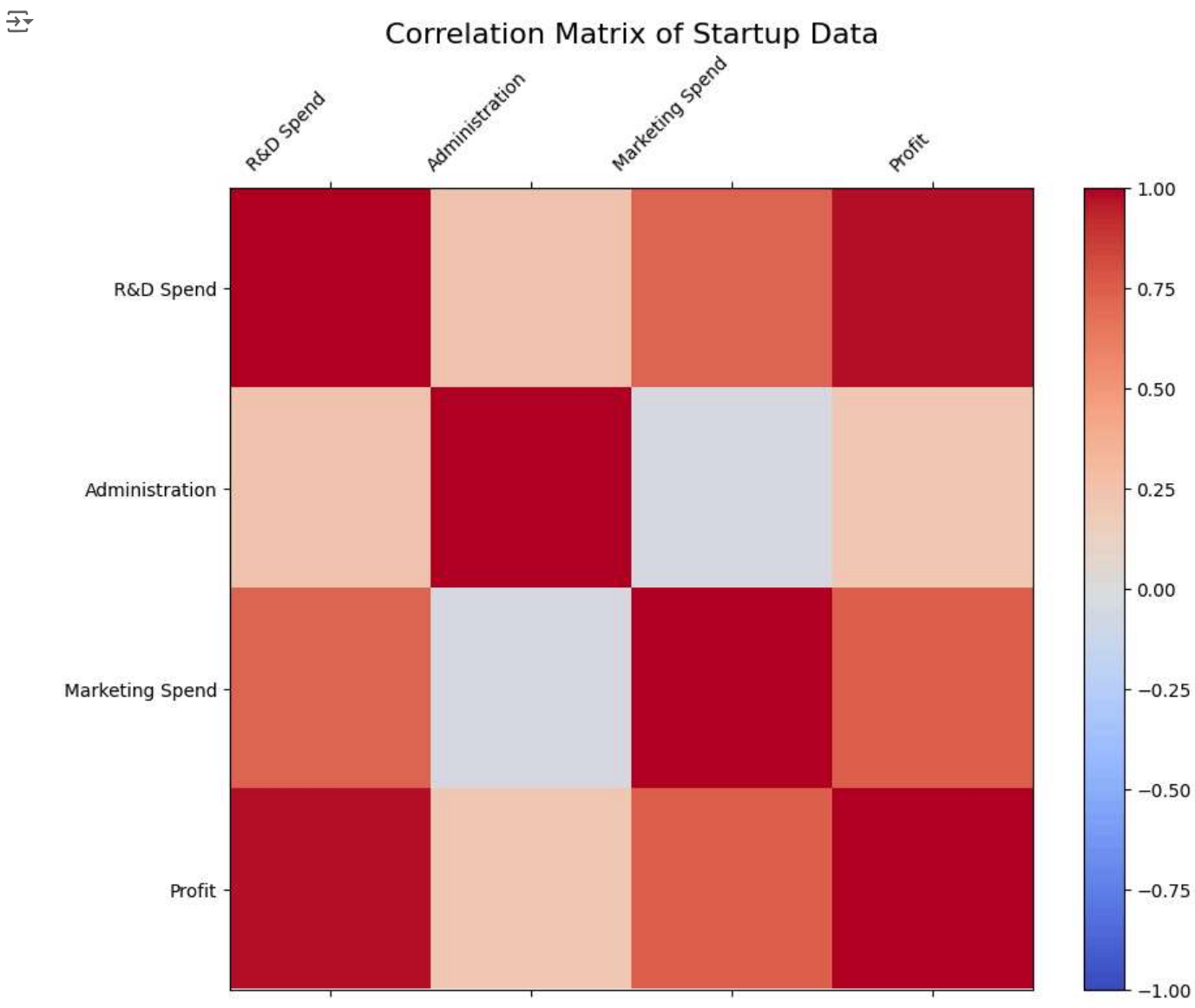
Maximum: 192,261.83: The highest profit recorded is 192,261.83, suggesting strong performance by certain companies.

**Correlation Analysis**

```
correlation_matrix = data.corr()

print(correlation_matrix)
```

```
                 R&D Spend  Administration  Marketing Spend    Profit
R&D Spend         1.000000        0.241955         0.724248  0.972900
Administration    0.241955        1.000000        -0.032154  0.200717
Marketing Spend   0.724248       -0.032154         1.000000  0.747766
Profit            0.972900        0.200717         0.747766  1.000000
```

```
# Create a heatmap
fig, ax = plt.subplots(figsize=(10, 8))
cax = ax.matshow(correlation_matrix, cmap='coolwarm', vmin=-1, vmax=1)
fig.colorbar(cax)

# Set labels for axes and title
ax.set_xticks(np.arange(len(correlation_matrix.columns)))
ax.set_yticks(np.arange(len(correlation_matrix.columns)))
ax.set_xticklabels(correlation_matrix.columns, rotation=45, ha='right')
ax.set_yticklabels(correlation_matrix.columns)
plt.title('Correlation Matrix of Startup Data', fontsize=16)

# Display the heatmap
plt.show()
```




R&D Spend vs. Profit:

Correlation coefficient: 0.973

Explanation: There is a very strong positive correlation of 0.973 between R&D Spend and Profit. This suggests a close relationship between R&D spending and the profitability of the startup companies. As R&D spending increases, profits tend to increase as well, indicating that R&D investment is a significant factor contributing to profitability.

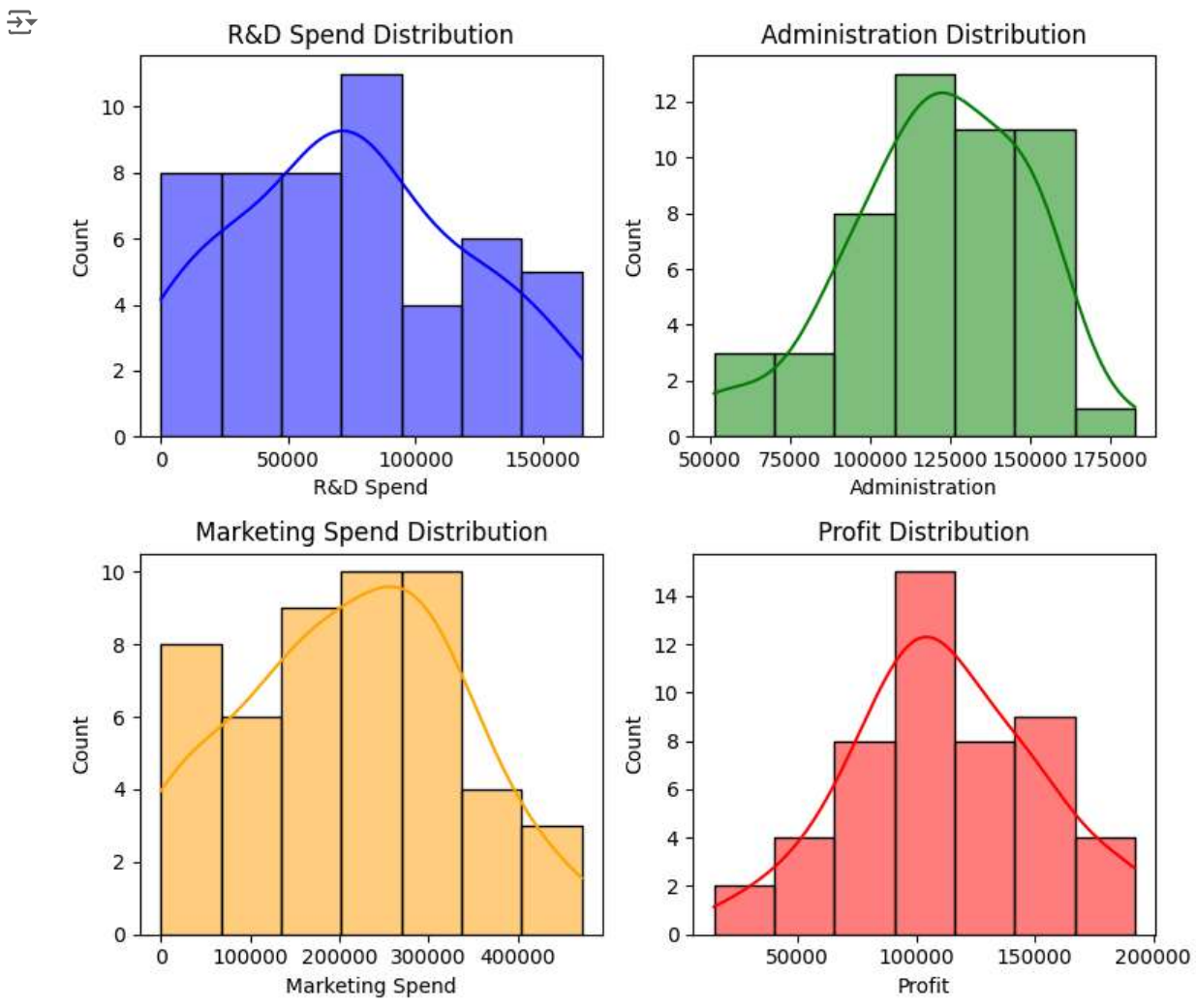
**Data Visualization**

Histogram:Interpreting histograms involves understanding the distribution of the data within each variable

Boxplot:Interpreting boxplots involves understanding whether the data is having outliers or not.

Scatter Plot:Interpreting scatter plots involves understanding the relationship between two variables.

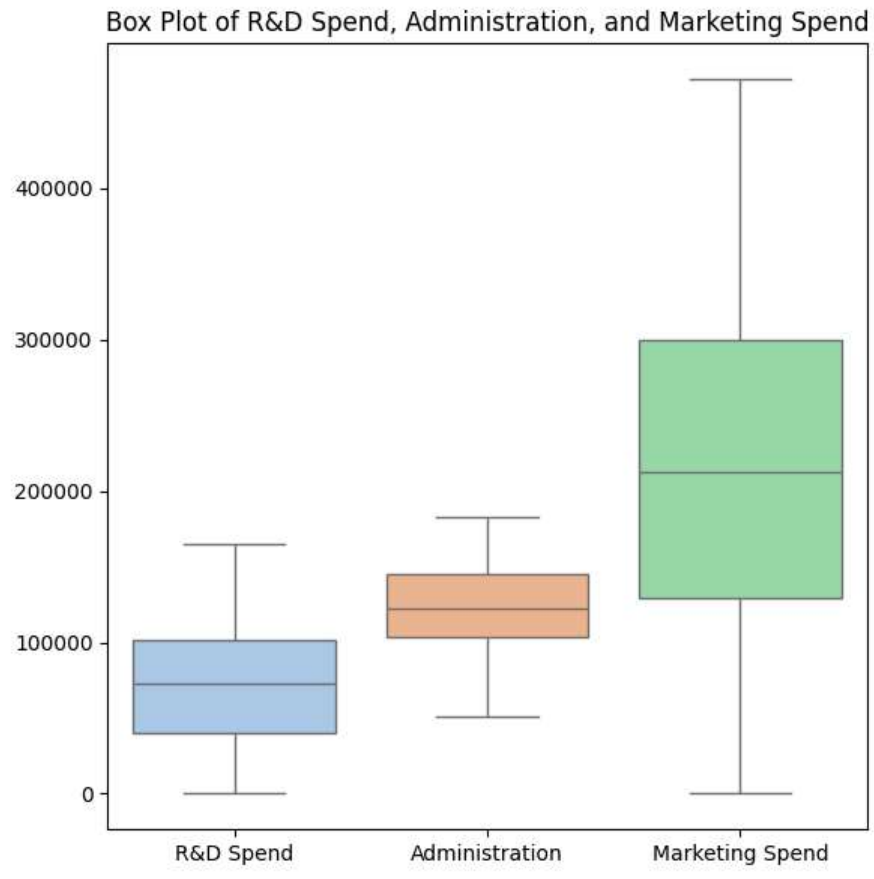
```
plt.figure(figsize=(8, 7))

# Plot histograms for each column
plt.subplot(2, 2, 1)
sns.histplot(data['R&D Spend'], kde=True, color='blue')
plt.title('R&D Spend Distribution')

plt.subplot(2, 2, 2)
sns.histplot(data['Administration'], kde=True, color='green')
plt.title('Administration Distribution')

plt.subplot(2, 2, 3)
sns.histplot(data['Marketing Spend'], kde=True, color='orange')
plt.title('Marketing Spend Distribution')

plt.subplot(2, 2, 4)
sns.histplot(data['Profit'], kde=True, color='red')
plt.title('Profit Distribution')

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

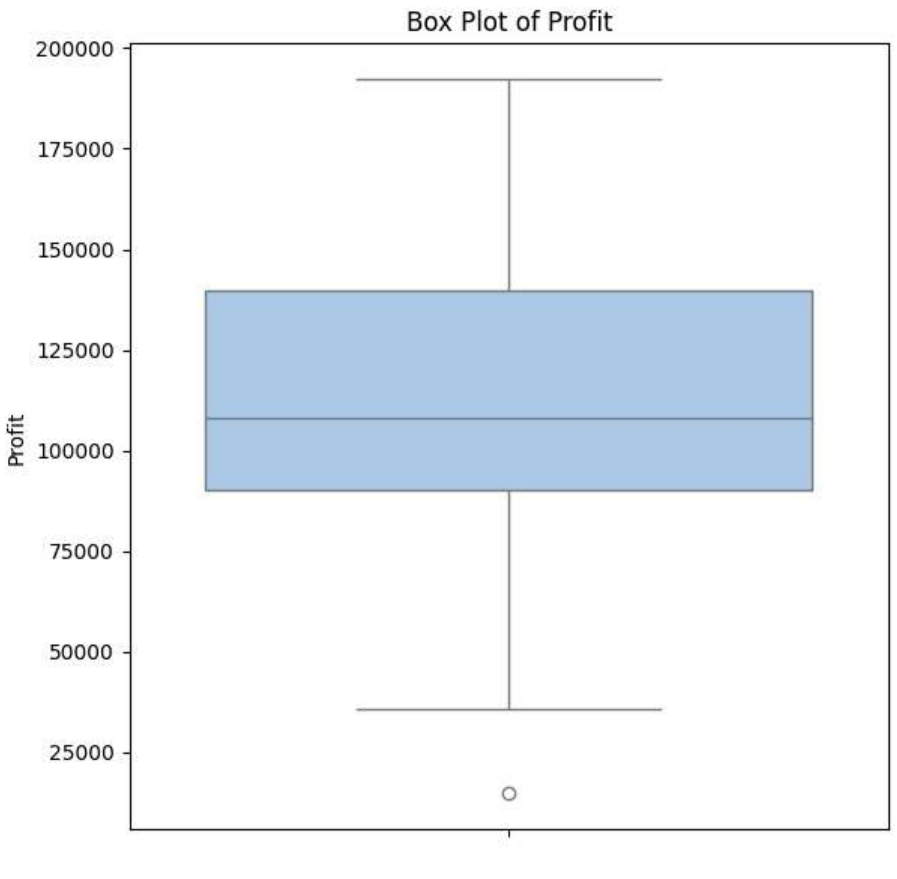



A symmetric distribution suggests a more even distribution of profits across companies, indicating that companies have similar levels of profitability.

```
# Create box plots for each column
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
sns.boxplot(data=data[['R&D Spend', 'Administration', 'Marketing Spend']], palette='pastel')
plt.title('Box Plot of R&D Spend, Administration, and Marketing Spend')

plt.subplot(1, 2, 2)
sns.boxplot(data=data['Profit'], palette='pastel', color='pink')
plt.title('Box Plot of Profit')

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

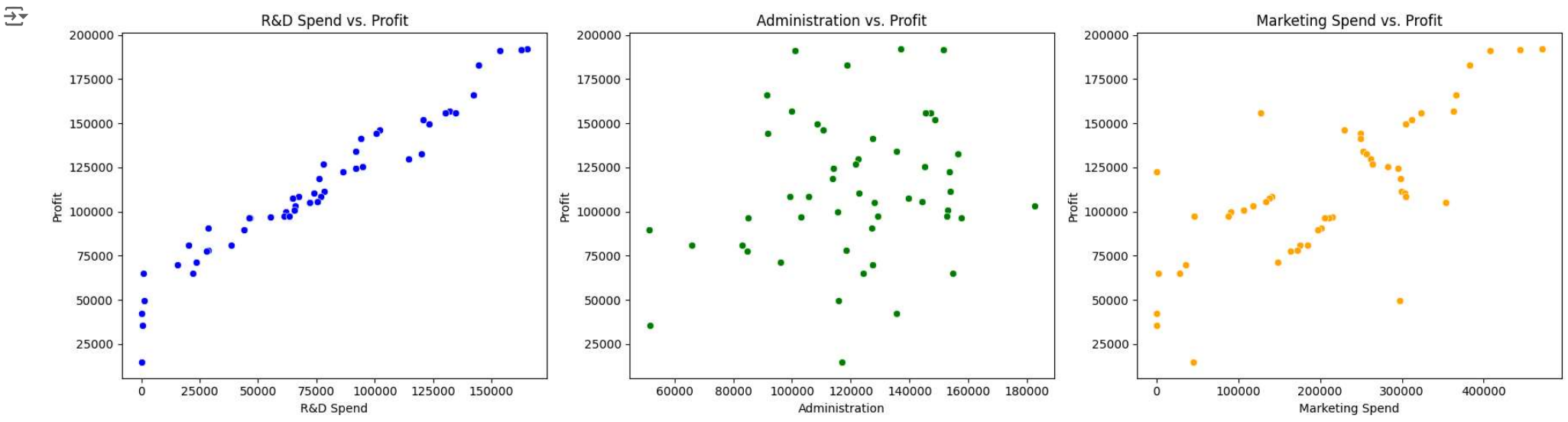
```
<ipython-input-9-331c42ab6438>:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(data=data['Profit'], palette='pastel', color='pink')
```




From the box plot we can visualize that there is no Outlier present in the data

```
plt.figure(figsize=(18, 5))

# Scatter plot between R&D Spend and Profit
plt.subplot(1, 3, 1)
sns.scatterplot(x='R&D Spend', y='Profit', data=data, color='blue')
plt.title('R&D Spend vs. Profit')

# Scatter plot between Administration and Profit
plt.subplot(1, 3, 2)
sns.scatterplot(x='Administration', y='Profit', data=data, color='green')
plt.title('Administration vs. Profit')

# Scatter plot between Marketing Spend and Profit
plt.subplot(1, 3, 3)
sns.scatterplot(x='Marketing Spend', y='Profit', data=data, color='orange')
plt.title('Marketing Spend vs. Profit')

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




The scatter plot shows the relationship between R&D spending and the profitability of the startup companies.

There is a positive correlation this indicates that as R&D spending increases, profits tend to increase as well.

**Identifying Dependent and Independent Variables**

```
# Separate independent variables (features) and dependent variable (target)
X = data[['R&D Spend', 'Administration', 'Marketing Spend']]  # Independent variables (features)
y = data['Profit']  # Dependent variable (target)

print("Independent Variables (Features):\n", X)
print("\nDependent Variable (Target):\n", y)
```

```
Independent Variables (Features):
     R&D Spend  Administration  Marketing Spend
0   165349.20       136897.80        471784.10
1   162597.70       151377.59        443898.53
2   153441.51       101145.55        407934.54
3   144372.41       118671.85        383199.62
4   142107.34        91391.77        366168.42
5   131876.90        99814.71        362861.36
```

```
6   134615.46       147198.87       127716.82
7   130298.13       145530.06       323876.68
8   120542.52       148718.95       311613.29
9   123334.88       108679.17       304981.62
10  101913.08       110594.11       229160.95
11  100671.96        91790.61       249744.55
12   93863.75       127320.38       249839.44
13   91992.39       135495.07       252664.93
14  119943.24       156547.42       256512.92
15  114523.61       122616.84       261776.23
16   78013.11       121597.55       264346.06
17   94657.16       145077.58       282574.31
18   91749.16       114175.79       294919.57
19   86419.70       153514.11            0.00
20   76253.86       113867.30       298664.47
21   78389.47       153773.43       299737.29
22   73994.56       122782.75       303319.26
23   67532.53       105751.03       304768.73
24   77044.01        99281.34       140574.81
25   64664.71       139553.16       137962.62
26   75328.87       144135.98       134050.07
27   72107.60       127864.55       353183.81
28   66051.52       182645.56       118148.20
29   65605.48       153032.06       107138.38
30   61994.48       115641.28        91131.24
31   61136.38       152701.92        88218.23
32   63408.86       129219.61        46085.25
33   55493.95       103057.49       214634.81
34   46426.07       157693.92       210797.67
35   46014.02        85047.44       205517.64
36   28663.76       127056.21       201126.82
37   44069.95        51283.14       197029.42
38   20229.59        65947.93       185265.10
39   38558.51        82982.09       174999.30
40   28754.33       118546.05       172795.67
41   27892.92        84710.77       164470.71
42   23640.93        96189.63       148001.11
43   15505.73       127382.30        35534.17
44   22177.74       154806.14        28334.72
45    1000.23       124153.04         1903.93
46    1315.46       115816.21       297114.46
47       0.00       135426.92            0.00
48     542.05        51743.15            0.00
49       0.00       116983.80        45173.06

Dependent Variable (Target):
 0     192261.83
1     191792.06
2     191050.39
```

```
X.count()
y.count()
```

```
50
```

### Dividing the data into train and test

```
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=42)
```

### Linear Regression

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

```
LinearRegression
LinearRegression()
```

```
reg.score(X_train,y_train)
```

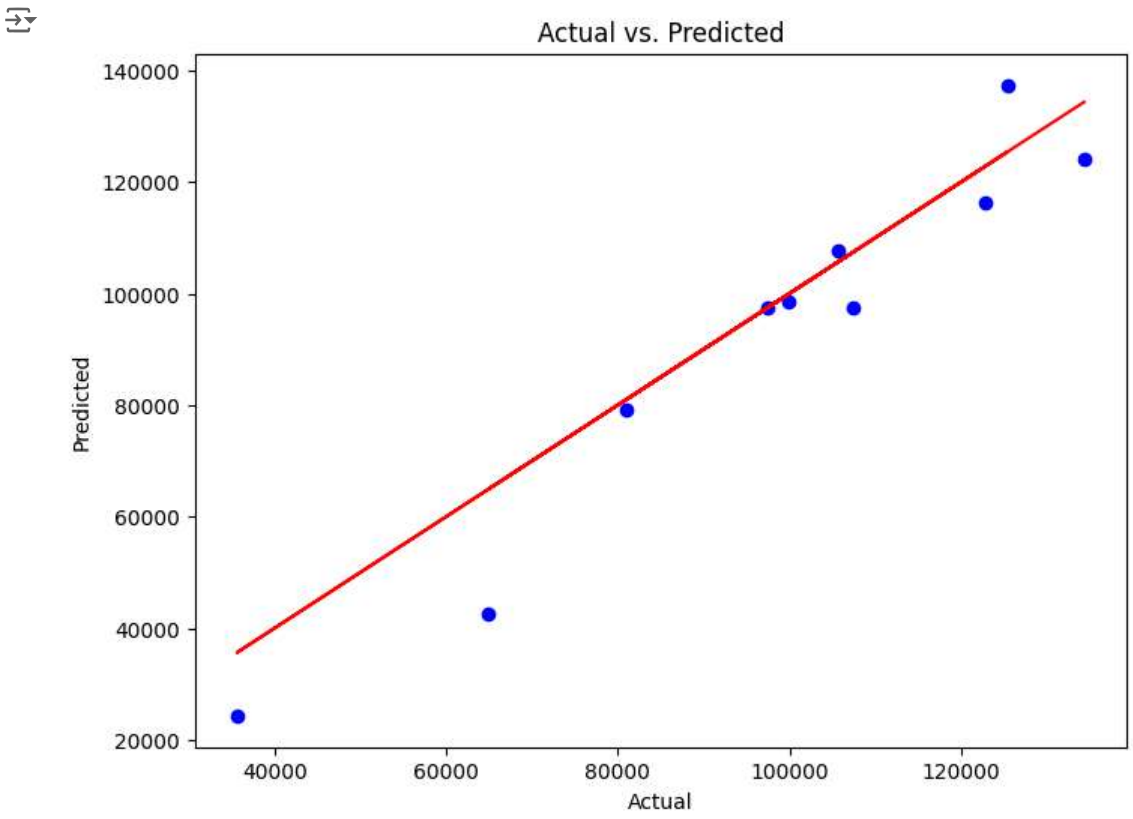
```
0.9535927757257411
```

```
y_pred=reg.predict(X_test)
```

```
df=pd.DataFrame({"Actual":y_test,"Predicted":y_pred})
df
```

| | Actual | Predicted |
|---|---|---|
| **13** | 134307.35 | 126703.027165 |
| **39** | 81005.76 | 84894.750816 |
| **30** | 99937.59 | 98893.418160 |
| **45** | 64926.08 | 46501.708150 |
| **17** | 125370.37 | 129128.397344 |
| **48** | 35673.41 | 50992.694863 |
| **26** | 105733.54 | 109016.553658 |
| **25** | 107404.34 | 100878.464145 |
| **32** | 97427.84 | 97700.596386 |
| **19** | 122776.86 | 113106.152922 |

```
plt.figure(figsize=(8, 6))
plt.scatter(df["Actual"], df["Predicted"], color='blue')
plt.plot(df["Actual"], df["Actual"], color='red')  # Add a diagonal line for comparison
plt.title('Actual vs. Predicted')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.show()
```




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

```
80926321.22295158
```

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

```
6979.152252370402
```

```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

```
0.9000653083037321
```

## Ridge Regression model

```
from sklearn.linear_model import Ridge

# Initialize the Ridge Regression model
ridge_model = Ridge(alpha=1.0)
ridge_model.fit(X_train,y_train)
```

```
Ridge
Ridge()
```

```
ridge_model.score(X_train,y_train)
```

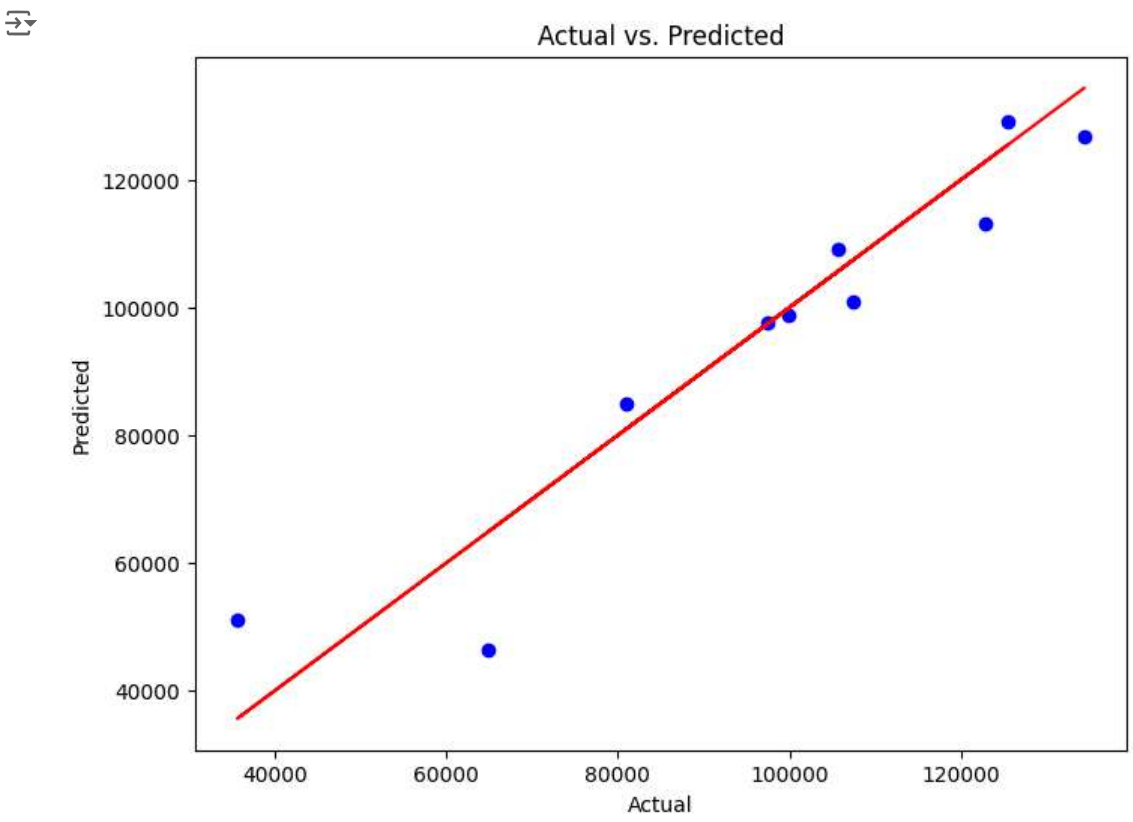
```
0.9535927757257411
```

```
y_pred1=ridge_model.predict(X_test)
```

```
df1=pd.DataFrame({"Actual":y_test,"Predicted":y_pred1})
df1
```

| | Actual | Predicted |
|---|---|---|
| **13** | 134307.35 | 126703.027165 |
| **39** | 81005.76 | 84894.750816 |
| **30** | 99937.59 | 98893.418159 |
| **45** | 64926.08 | 46501.708150 |
| **17** | 125370.37 | 129128.397344 |
| **48** | 35673.41 | 50992.694862 |
| **26** | 105733.54 | 109016.553657 |
| **25** | 107404.34 | 100878.464145 |
| **32** | 97427.84 | 97700.596385 |
| **19** | 122776.86 | 113106.152921 |

```
plt.figure(figsize=(8, 6))
plt.scatter(df1["Actual"], df1["Predicted"], color='blue')
plt.plot(df1["Actual"], df1["Actual"], color='red')  # Add a diagonal line for comparison
plt.title('Actual vs. Predicted')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.show()
```




```
from sklearn.metrics import mean_squared_error
Ridge_regression=mean_squared_error(y_test,y_pred1)
Ridge_regression
```

```
80926321.22368833
```

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test,y_pred1)
```

```
6979.152252428612
```

```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred1)
```

```
0.9000653083028223
```

## Lasso Regression

```
from sklearn.linear_model import Lasso

# Initialize the Lasso Regression model
lasso_model = Lasso(alpha=1.0)
lasso_model.fit(X_train,y_train)
```

```
▾ Lasso
Lasso()
```

```
lasso_model.score(X_train,y_train)
```

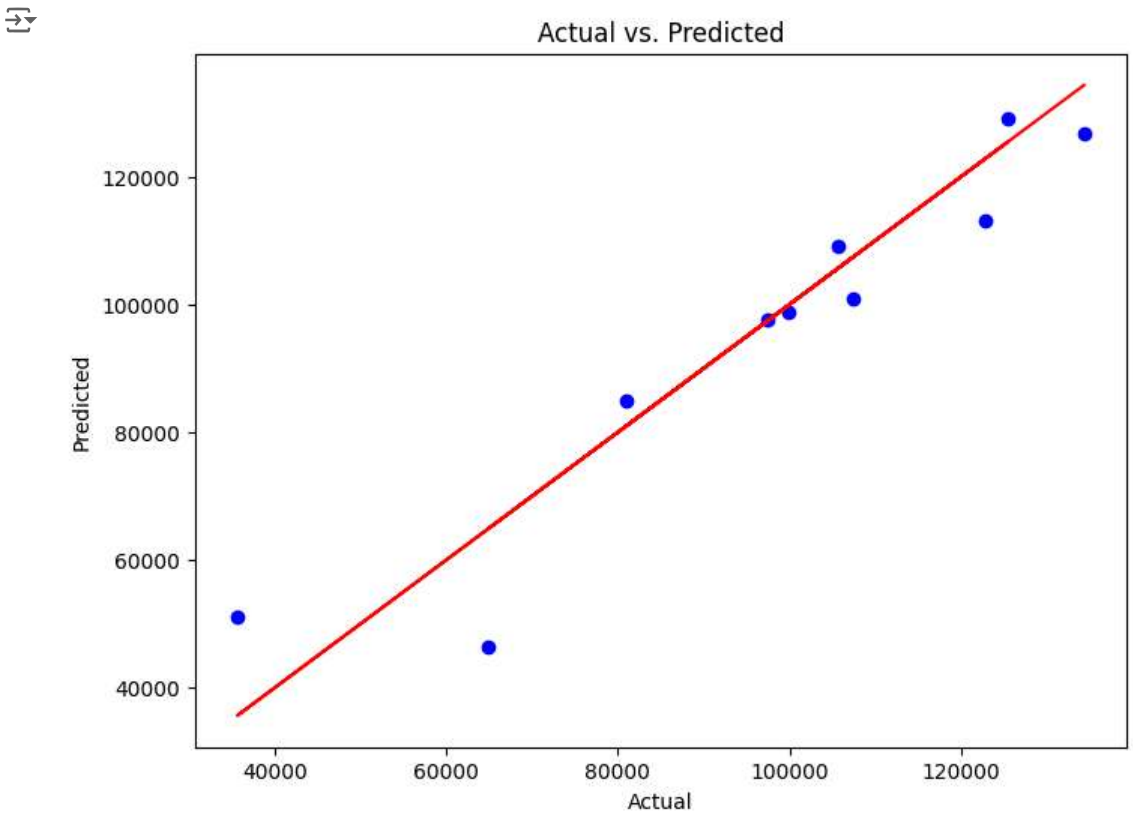
```
0.9535927757257411
```

```
y_pred2=lasso_model.predict(X_test)
```

```
df2=pd.DataFrame({"Actual":y_test,"Predicted":y_pred2})
df2
```

| | Actual | Predicted |
|---|---|---|
| **13** | 134307.35 | 126703.027183 |
| **39** | 81005.76 | 84894.750771 |
| **30** | 99937.59 | 98893.418184 |
| **45** | 64926.08 | 46501.708163 |
| **17** | 125370.37 | 129128.397364 |
| **48** | 35673.41 | 50992.694812 |
| **26** | 105733.54 | 109016.553705 |
| **25** | 107404.34 | 100878.464180 |
| **32** | 97427.84 | 97700.596436 |
| **19** | 122776.86 | 113106.153023 |

```
plt.figure(figsize=(8, 6))
plt.scatter(df2["Actual"], df2["Predicted"], color='blue')
plt.plot(df2["Actual"], df2["Actual"], color='red')  # Add a diagonal line for comparison
plt.title('Actual vs. Predicted')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.show()
```




```
from sklearn.metrics import mean_squared_error
Lasso_Regression=mean_squared_error(y_test,y_pred2)
Lasso_Regression
```

```
80926320.76116839
```

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test,y_pred2)
```

```
6979.152235475121
```

```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred2)
```

```
0.9000653088739812
```

## Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor

# Initialize the Random Forest Regression model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train,y_train)
```

```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

```
rf_model.score(X_train,y_train)
```

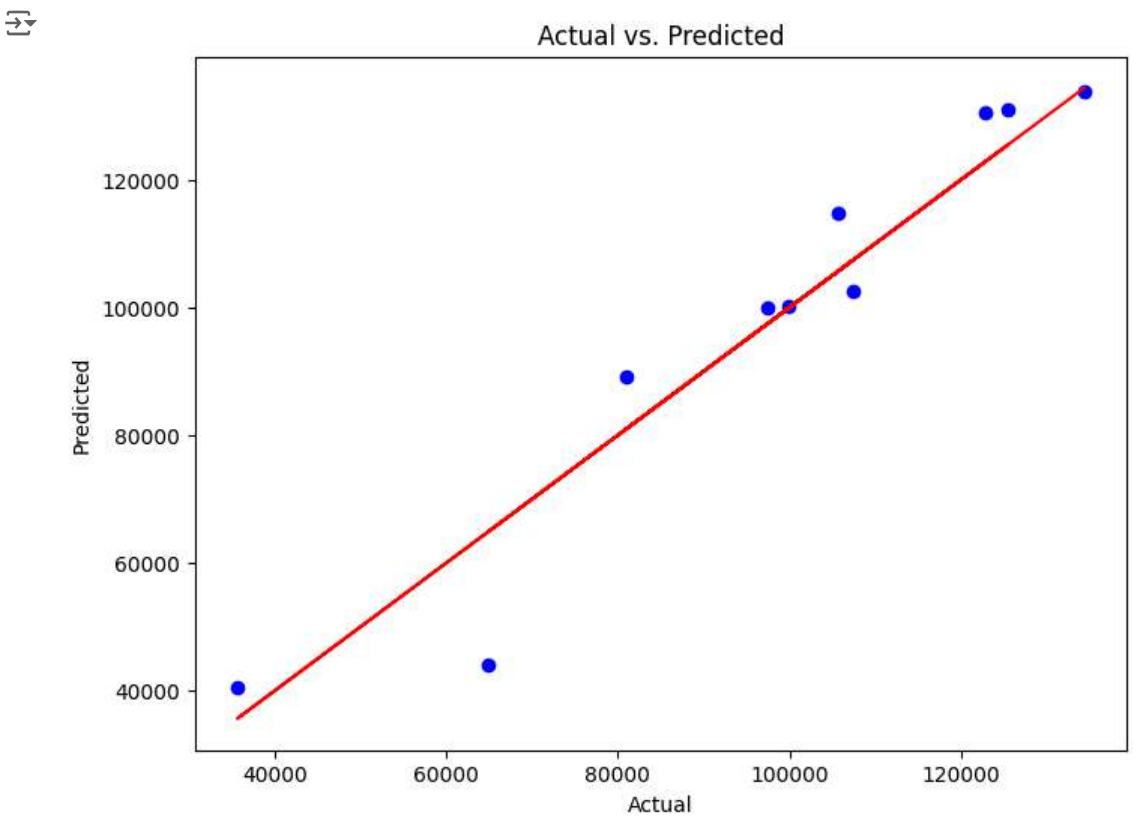
```
0.9909353043705511
```

```
y_pred3=rf_model.predict(X_test)
```

```
df3=pd.DataFrame({"Actual":y_test,"Predicted":y_pred3})
df3
```

| | Actual | Predicted |
|---|---|---|
| **13** | 134307.35 | 133831.6871 |
| **39** | 81005.76 | 89192.0856 |
| **30** | 99937.59 | 100239.4889 |
| **45** | 64926.08 | 44157.3419 |
| **17** | 125370.37 | 131047.2193 |
| **48** | 35673.41 | 40492.4908 |
| **26** | 105733.54 | 114781.5865 |
| **25** | 107404.34 | 102560.8152 |
| **32** | 97427.84 | 99888.9697 |
| **19** | 122776.86 | 130570.5808 |

```
plt.figure(figsize=(8, 6))
plt.scatter(df3["Actual"], df3["Predicted"], color='blue')
plt.plot(df3["Actual"], df3["Actual"], color='red')  # Add a diagonal line for comparison
plt.title('Actual vs. Predicted')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.show()
```




```
from sklearn.metrics import mean_squared_error
rf=mean_squared_error(y_test,y_pred3)
rf
```

```
72625008.62306513
```

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test,y_pred3)
```

```
6437.497739999977
```

```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred3)
```

```
0.9103164738430438
```

## SVM

```
# prompt: support vector machine apply odel

from sklearn.svm import SVR

# Initialize the SVR model
svr_model = SVR(kernel='linear')
svr_model.fit(X_train,y_train)
```

```
SVR(kernel='linear')
```

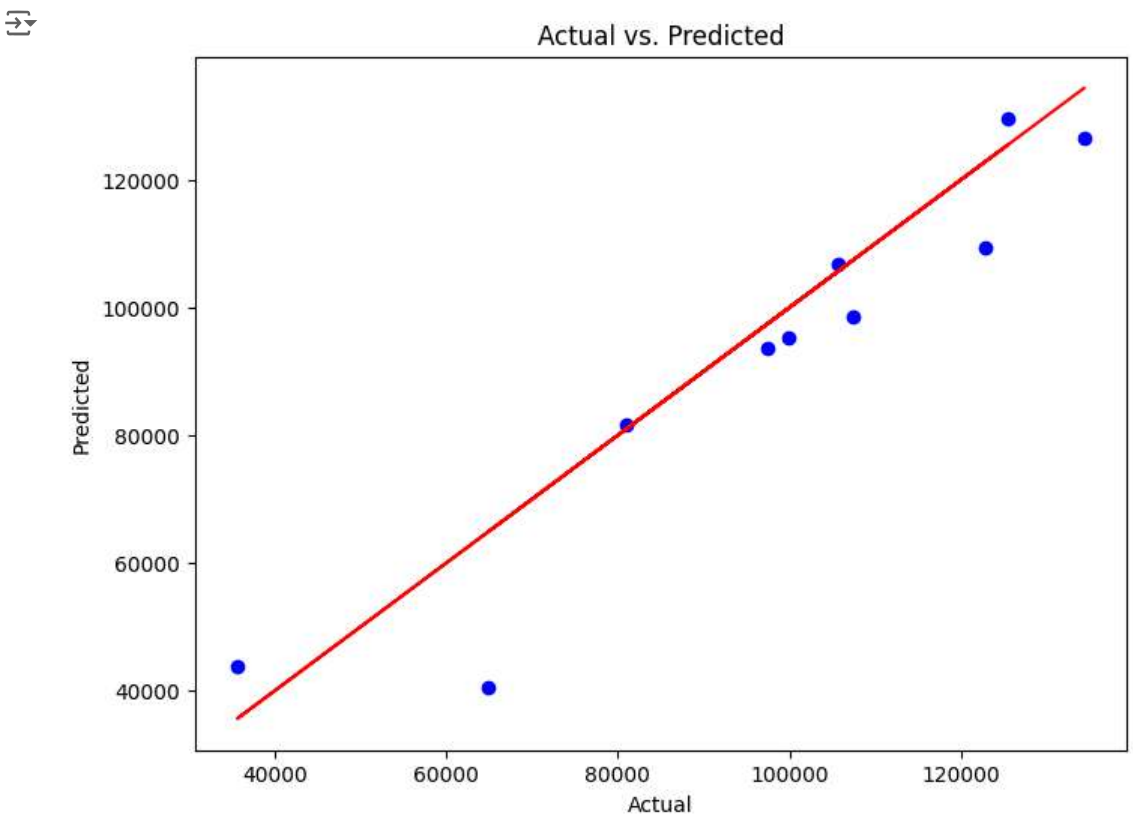
```
svr_model.score(X_train,y_train)
```

```
0.9493135484875507
```

```
y_pred4=svr_model.predict(X_test)
df4=pd.DataFrame({"Actual":y_test,"Predicted":y_pred4})
df4
```

| | Actual | Predicted |
|---|---|---|
| **13** | 134307.35 | 126550.837667 |
| **39** | 81005.76 | 81640.943593 |
| **30** | 99937.59 | 95354.687719 |
| **45** | 64926.08 | 40518.231950 |
| **17** | 125370.37 | 129640.765783 |
| **48** | 35673.41 | 43802.709072 |
| **26** | 105733.54 | 106859.511922 |
| **25** | 107404.34 | 98493.962987 |
| **32** | 97427.84 | 93721.453985 |
| **19** | 122776.86 | 109275.503904 |

```
plt.figure(figsize=(8, 6))
plt.scatter(df4["Actual"], df4["Predicted"], color='blue')
plt.plot(df4["Actual"], df4["Actual"], color='red')  # Add a diagonal line for comparison
plt.title('Actual vs. Predicted')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.show()
```




```
mean_squared_error(y_test,y_pred4)
```

```
103832131.00714561
```

```
mean_absolute_error(y_test,y_pred4)
```

```
7702.623215893979
```

```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred4)
```

```
0.8717792697906213
```

### Decision Tree

```
# prompt: Decision tree

from sklearn import tree

# Initialize the Decision Tree Regression model
dt_model = tree.DecisionTreeRegressor(max_depth=3)
dt_model.fit(X_train,y_train)
```

```
DecisionTreeRegressor(max_depth=3)
```

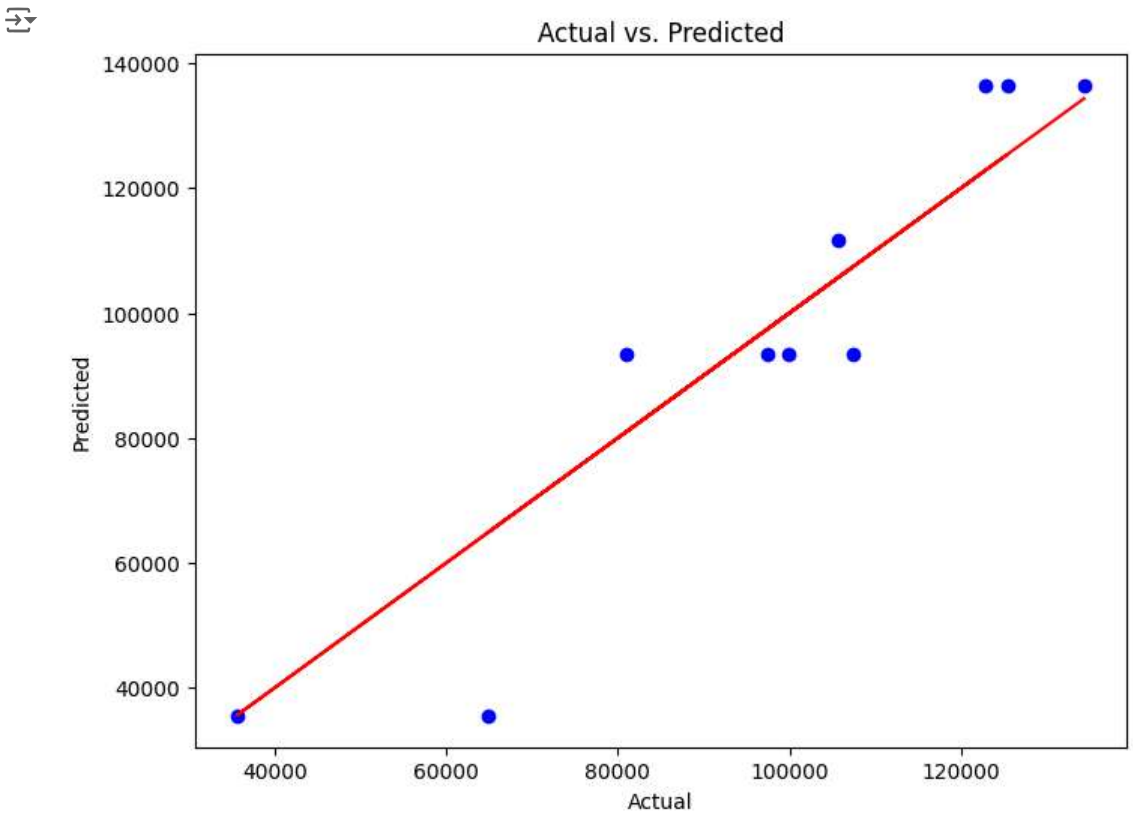
```
dt_model.score(X_train,y_train)
```

```
0.9686293734459094
```

```
y_pred5=dt_model.predict(X_test)
df5=pd.DataFrame({"Actual":y_test,"Predicted":y_pred5})
df5
```

| | Actual | Predicted |
|---|---|---|
| **13** | 134307.35 | 136458.910000 |
| **39** | 81005.76 | 93419.583750 |
| **30** | 99937.59 | 93419.583750 |
| **45** | 64926.08 | 35577.293333 |
| **17** | 125370.37 | 136458.910000 |
| **48** | 35673.41 | 35577.293333 |
| **26** | 105733.54 | 111588.618750 |
| **25** | 107404.34 | 93419.583750 |
| **32** | 97427.84 | 93419.583750 |
| **19** | 122776.86 | 136458.910000 |

```
plt.figure(figsize=(8, 6))
plt.scatter(df5["Actual"], df5["Predicted"], color='blue')
plt.plot(df5["Actual"], df5["Actual"], color='red')  # Add a diagonal line for comparison
plt.title('Actual vs. Predicted')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.show()
```




```
decision_tree=mean_squared_error(y_test,y_pred5)
decision_tree
```

```
161865283.74642056
```

```
mean_absolute_error(y_test,y_pred5)
```

```
9914.697458333338
```

```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred5)
```

```
0.8001150060564004
```

```
# prompt: xgboost classifier

from xgboost import XGBRegressor

# Initialize the XGBoost Regressor model
xgb_model = XGBRegressor(n_estimators=100, random_state=42)
xgb_model.fit(X_train,y_train)
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=100, n_jobs=None,
             num_parallel_tree=None, random_state=42, ...)
```

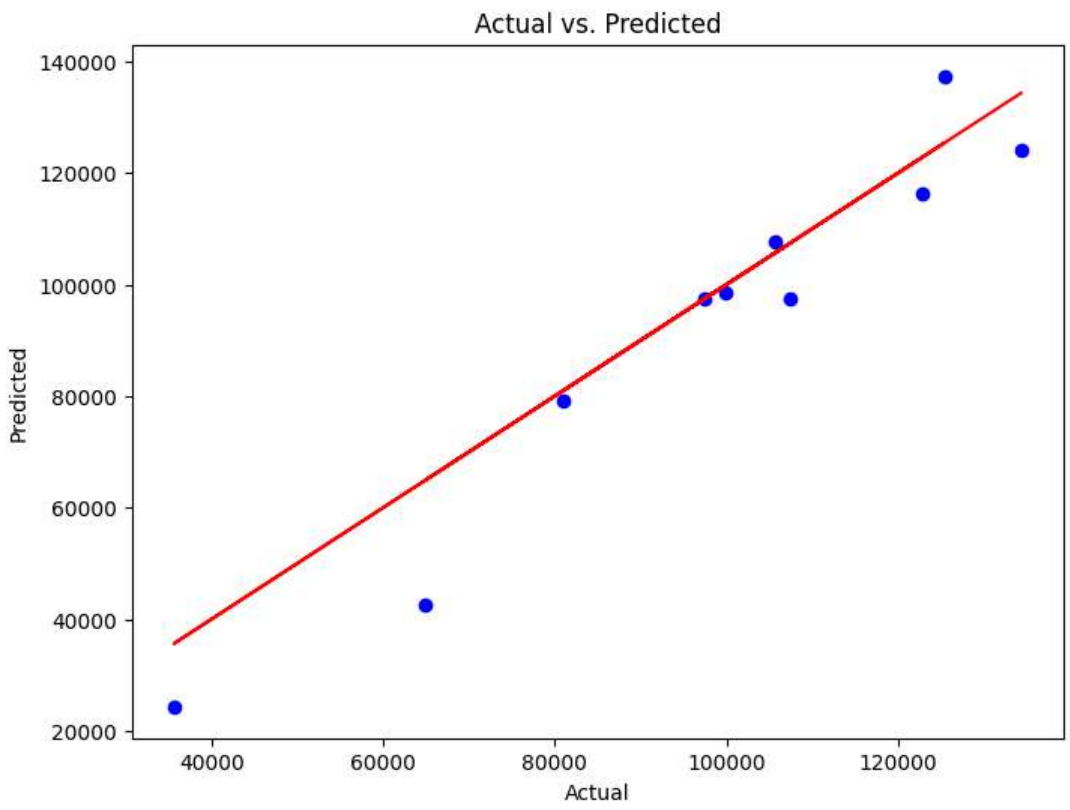
```
xgb_model.score(X_train,y_train)
```

```
0.9999999999985478
```

```
y_pred6=xgb_model.predict(X_test)
df6=pd.DataFrame({"Actual":y_test,"Predicted":y_pred6})
df6
```

| | Actual | Predicted |
|---|---|---|
| **13** | 134307.35 | 123975.867188 |

```
plt.figure(figsize=(8, 6))
plt.scatter(df6["Actual"], df6["Predicted"], color='blue')
plt.plot(df6["Actual"], df6["Actual"], color='red')  # Add a diagonal line for comparison
plt.title('Actual vs. Predicted')
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.show()
```




```
mean_squared_error(y_test,y_pred6)
```

```
102829917.00381728
```

```
mean_absolute_error(y_test,y_pred6)
```

```
7757.660484375001
```

```
r2_score(y_test,y_pred6)
```

```
0.8730168887250143
```

**Comparison Table**

```
final_data=pd.DataFrame({'Models':['LR','DT','RF','Ridge','SVM','LASSO','XgBoost'],
             "ACC":[reg.score(X_train,y_train)*100,
                    dt_model.score(X_train,y_train)*100,
                    rf_model.score(X_train,y_train)*100,
                    ridge_model.score(X_train,y_train)*100,
                    svr_model.score(X_train,y_train)*100,
                    lasso_model.score(X_train,y_train)*100,
                    xgb_model.score(X_train,y_train)*100],
             "R2_score":[r2_score(y_test,y_pred)*100,
                         r2_score(y_test,y_pred5)*100,
                         r2_score(y_test,y_pred3)*100,
                         r2_score(y_test,y_pred1)*100,
                         r2_score(y_test,y_pred4)*100,
                         r2_score(y_test,y_pred2)*100,
                         r2_score(y_test,y_pred6)*100],
                "MSE":[mean_squared_error(y_test,y_pred)*100,
                       mean_squared_error(y_test,y_pred5)*100,
                       mean_squared_error(y_test,y_pred3)*100,
                       mean_squared_error(y_test,y_pred1)*100,
                       mean_squared_error(y_test,y_pred4)*100,
                       mean_squared_error(y_test,y_pred2)*100,
                       mean_squared_error(y_test,y_pred6)*100],
                 "MAE":[mean_absolute_error(y_test,y_pred)*100,
                        mean_absolute_error(y_test,y_pred5)*100,
                        mean_absolute_error(y_test,y_pred3)*100,
                        mean_absolute_error(y_test,y_pred1)*100,
                        mean_absolute_error(y_test,y_pred4)*100,
                        mean_absolute_error(y_test,y_pred2)*100,
                         mean_absolute_error(y_test,y_pred6)*100
                        ]})
```

```
final_data
```

| | Models | ACC | R2_score | MSE | MAE |
|---|---|---|---|---|---|
| **0** | LR | 95.359278 | 90.006531 | 8.092632e+09 | 697915.225237 |
| **1** | DT | 96.862937 | 80.011501 | 1.618653e+10 | 991469.745833 |
| **2** | RF | 99.093530 | 91.031647 | 7.262501e+09 | 643749.774000 |
| **3** | Ridge | 95.359278 | 90.006531 | 8.092632e+09 | 697915.225243 |
| **4** | SVM | 94.931355 | 87.177927 | 1.038321e+10 | 770262.321589 |
| **5** | LASSO | 95.359278 | 90.006531 | 8.092632e+09 | 697915.223548 |