ABSTRACT

A study of 50 companies found that R&D spend and marketing spend are positively correlated with profit. However, there is no clear correlation between administration spend and profit. The study also found that the correlation between R&D spend and profit is stronger than the correlation between marketing spends and profit. This suggests that R&D spend may be a more important factor in determining profit.

The study also found that there is a wide range of profit levels for the companies in the sample. This suggests that there are other factors that affect profit, besides R&D spend, marketing spend, and administration spend. These factors could include the quality of the company's products or services, the company's business model, or the competitive landscape.

The study presented here is just a snapshot of the relationship between R&D spend, marketing spends, administration spend, and profit. More research is needed to understand the full impact of these factors on profit.

Here are some additional insights that can be generalized from the data:

Companies that invest in R&D and marketing are more likely to be profitable.

The correlation between R&D spend and profit is stronger for companies that are in the early stages of growth.

The correlation between marketing spends and profit is stronger for companies that are in the mature stages of growth.

Overall, the study suggests that R&D spend and marketing spend are important factors in determining profit. However, there are other factors that can also affect profit, and these factors may vary from company to company.

TABLE OF CONTENTS

INTRODUCTION

In the realm of data science, regression models have become a prominent tool for predicting and understanding business outcomes. This report explores the application of various regression models to predict company profits based on a dataset comprising financial data from multiple organizations. Specifically, we will examine the performance of linear regression, lasso regression, ridge regression, decision tree regression, gradient boosting, and random forest regression models.

The primary objective of this analysis is to identify the most suitable regression model for predicting company profits. By evaluating the performance of each model, we can gain insights into their strengths, weaknesses, and overall effectiveness in this specific context.

Linear regression serves as a fundamental baseline model for predicting continuous target variables. It establishes a linear relationship between the independent variables, such as R&D spend, administration expenses, marketing spend, and profit. We will assess the predictive power of linear regression and analyze its interpretability and simplicity.

Lasso regression, a variant of linear regression, incorporates a regularization term that encourages sparsity in the model. This technique can effectively handle datasets with a large number of input variables by shrinking some coefficients to zero. We will evaluate lasso regression's ability to select important features and enhance prediction accuracy.

Ridge regression, another extension of linear regression, employs a regularization term that mitigates issues related to multicollinearity among the independent variables. By reducing the impact of highly correlated variables, ridge regression can offer more stable and reliable predictions. We will assess its ability to handle multicollinearity and improve prediction performance.

Decision tree regression takes a different approach by creating a tree-like model to represent decision rules and make predictions. This non-linear model can capture complex interactions and nonlinear relationships between variables. We will evaluate decision tree regression in terms of interpretability and its ability to handle non-linear patterns in the data.

Gradient boosting is an ensemble method that combines multiple weak predictive models, typically decision trees, to create a stronger overall model. It iteratively builds upon the weaknesses of previous models, improving prediction accuracy. We will assess the effectiveness of gradient boosting in capturing complex relationships and its overall prediction performance.

Random forest regression, another ensemble method, constructs an ensemble of decision trees and aggregates their predictions to obtain a final prediction. This technique can handle high-dimensional data and provide robust predictions by reducing overfitting. We will evaluate random forest regression's ability to handle noise, variability, and missing values within the dataset.

By comparing the performance of these regression models, we can determine the most appropriate approach for predicting company profits based on the given dataset. The findings of this analysis will provide valuable insights for organizations seeking to leverage regression techniques to optimize their financial performance and make data-driven decisions.

Overall, this report aims to showcase the potential of various regression models, including linear regression, lasso regression, ridge regression, decision tree regression, gradient boosting, and random forest regression, in predicting company profits. By leveraging the strengths of each model, businesses can gain valuable insights into the factors influencing profitability and drive sustainable growth in today's competitive business landscape.

EXISTING METHOD

In this section, we will delve into the existing methods that organizations commonly employ to optimize profitability. These methods are widely used in the field and provide valuable insights into traditional practices. They can be broadly categorized into financial analysis techniques and statistical modeling approaches.

4.1 Financial Analysis Techniques

Financial analysis techniques are essential for assessing a company's financial health, performance, and profitability. These techniques involve the examination of various financial ratios and cost analysis to gain insights into the factors impacting profitability. Let's explore some commonly used financial analysis methods:

4.1.1 Ratio Analysis:

Ratio analysis involves the calculation and interpretation of key financial ratios that provide insights into a company's profitability and financial performance. Common ratios analyzed include gross profit margin, net profit margin, return on investment (ROI), return on equity (ROE), and liquidity ratios. These ratios help assess a company's efficiency, profitability, and ability to generate returns for its shareholders.

4.1.2 Cost-Volume-Profit Analysis:

Cost-Volume-Profit (CVP) analysis is a technique that examines the relationships between costs, sales volume, and profitability. It helps organizations understand the impact of changes in sales volume on costs and profits. By analysing cost behaviour and determining the breakeven point, organizations can make informed decisions regarding pricing, cost control measures, and setting sales targets to achieve profitability.

4.1.3 Break-Even Analysis:

Break-even analysis is a valuable tool for determining the point at which a company's total revenue equals its total costs, resulting in zero profit or loss. This analysis helps organizations understand the minimum level of sales needed to cover costs and achieve profitability. By identifying the breakeven point, companies can set sales targets and pricing strategies accordingly.

4.2 Statistical Modelling Approaches

In addition to financial analysis techniques, statistical modelling approaches are widely used to analysed data, identify the factors that influence profitability, and predict future outcomes. Let's explore some commonly used statistical modelling approaches:

4.2.1 Linear Regression:

Linear regression is a statistical modelling technique used to model the relationship between one dependent variable (such as profit) and one or more independent variables (such as R&D spend, administration expenses, and marketing spend). It assumes a linear relationship between the variables and estimates the coefficients that best fit the data. Linear regression provides insights into the direction and strength of the relationships between variables, allowing organizations to understand which factors have a significant impact on profitability.

4.2.2 Time Series Analysis:

Time series analysis is employed when analysing data collected over time. It helps identify patterns, trends, and seasonality in the data. Time series forecasting techniques, such as moving averages, exponential smoothing, and autoregressive integrated moving average (ARIMA) models, can be applied to predict future profitability based on historical trends. These methods account for the sequential nature of the data and help organizations make informed decisions based on future expectations.

4.2.3 Machine Learning Algorithms:

Machine learning algorithms, including decision trees, random forests, gradient boosting, and neural networks, have gained prominence in predicting profitability. These algorithms can handle complex relationships, non-linearity, and interactions among variables. By training on historical data, machine learning models can learn patterns and relationships, enabling accurate predictions of future profitability. These models are particularly useful when dealing with large datasets and complex business scenarios.

By employing a combination of financial analysis techniques and statistical modelling approaches, organizations can gain a comprehensive understanding of their profitability drivers. These methods help identify areas for improvement, make informed decisions, and optimize resource allocation to maximize profitability.

While the existing methods discussed above have proven effective, it is crucial to explore and compare alternative approaches to determine which method yields the best results for predicting profitability in the specific dataset and business context. The subsequent section will present the proposed method and its architecture, highlighting its potential advantages and improvements over the existing methods.

PROPOSED METHOD WITH ARCHITECTURE

In this section, we present our proposed method for predicting company profits based on the provided dataset. Our approach involves the utilization of several regression models, including linear regression, lasso regression, ridge regression, decision tree regression, gradient boosting, and random forest regression. Each model has distinct characteristics and can capture different aspects of the relationship between input variables and profitability. By comparing their performance, we can determine the most suitable regression model for accurate profit predictions.

The architecture of our proposed method is as follows:

Data Pre-processing:

The first step in our method is data pre-processing. This involves handling missing values, removing outliers (if necessary), and normalizing or scaling the input variables. Data pre-processing is crucial to ensure that the dataset is clean, standardized, and ready for analysis. It helps improve the quality of the predictions by addressing data quality issues.

Feature Selection:

Next, we employ feature selection techniques to identify the most relevant input variables that have a significant impact on profitability. Feature selection helps reduce the dimensionality of the dataset and focuses on the most influential variables, leading to improved model performance and interpretability. Techniques such as correlation analysis, stepwise regression, or regularization methods (e.g., L1 regularization in lasso regression) can be applied to select the most important features.

Model Training and Evaluation:

Once the dataset is pre-processed and the features are selected, we proceed to train and evaluate each regression model. We split the dataset into training and testing sets, typically using a 70:30 or 80:20 ratio. The training set is used to train the models, and the testing set is used to evaluate their performance. During training, the models learn the relationships between the input variables (R&D spend, administration expenses, marketing spend) and the target variable (profit) based on the training data.

Hyperparameter Tuning:

To optimize the performance of each regression model, we perform hyperparameter tuning. Hyperparameters are configuration settings that control the learning process of the models. Techniques such as grid search or random search combined with cross-validation can be used to explore different combinations of hyperparameters and identify the best settings that yield optimal performance. The goal is to find the set of hyperparameters that minimize prediction errors and maximize the model's accuracy.

Performance Evaluation:

After training and hyperparameter tuning, we evaluate the performance of each regression model using appropriate evaluation metrics. Common metrics include mean squared error (MSE), mean absolute error (MAE), or R-squared (R2). These metrics provide insights into the accuracy and predictive power of the models. By comparing the performance of each model, we can determine the one that achieves the highest accuracy and generalizability for predicting company profits.

Model Selection:

Based on the performance evaluation, we select the regression model that performs the best in terms of prediction accuracy and reliability. The selected model becomes our recommended model for profit prediction based on the given dataset. Factors considered in the selection process include the model's ability to capture the relationships between input variables and profit, its robustness to noise and outliers, and its interpretability.

Prediction and Interpretation:

Once the best model is selected, we can utilize it to make profit predictions for new, unseen data. The model takes input values for R&D spend, administration expenses, and marketing spend and provides a predicted profit value. These predictions can be further analysed to understand the impact of each input variable on profitability and gain valuable insights for decision-making. For example, we can assess how increasing R&D spend or optimizing marketing strategies can potentially boost profits.

Our proposed method, with its architecture, allows for comprehensive exploration and comparison of various regression models to determine the most suitable one for predicting company profits. By leveraging the strengths of each model, we can obtain accurate profit predictions and gain valuable insights into the factors driving profitability. The flexibility and interpretability of the selected model contribute to its usefulness in making data-driven decisions and optimizing business strategies.

METHODOLOGY

In this section, we outline the methodology used to implement and evaluate the proposed method for predicting company profits based on the provided dataset. The methodology encompasses the step-by-step process followed to ensure reliable and accurate results.

Data Collection:

The first step is to collect the dataset containing financial data from the 50 companies. The dataset should include variables such as R&D spend, administration expenses, marketing spend, and profit. It is essential to ensure the dataset is representative and contains reliable data points for each company.

Data Pre-processing:

Once the dataset is collected, data pre-processing techniques are applied to ensure the dataset is clean and suitable for analysis. This involves handling missing values, checking for outliers, and normalizing or scaling the input variables as needed. Data pre-processing improves the quality and reliability of the analysis.

Feature Selection:

Next, feature selection techniques are employed to identify the most relevant input variables that significantly impact profitability. Various methods such as correlation analysis, stepwise regression, or regularization techniques can be used to select the most important features. The goal is to reduce the dimensionality of the dataset and focus on the most influential variables.

Model Training:

After feature selection, the dataset is divided into training and testing sets. The training set is used to train the selected regression models, including linear regression, lasso regression, ridge regression, decision tree regression, gradient boosting, and random forest regression. Each model is trained on the training set to learn the relationships between the input variables and the target variable (profit).

Hyperparameter Tuning:

To optimize the performance of the regression models, hyperparameter tuning is performed. Hyperparameters are configuration settings that control the learning process of the models. Techniques such as grid search or random search combined

with cross-validation are used to explore different combinations of hyperparameters and identify the best settings that yield optimal performance.

Model Evaluation:

Once the models are trained and hyperparameters are tuned, they are evaluated using appropriate evaluation metrics. Common metrics such as mean squared error (MSE), mean absolute error (MAE), or R-squared (R2) are calculated to assess the accuracy and predictive power of the models. The evaluation provides insights into how well each model performs in predicting company profits.

Model Selection:

Based on the evaluation results, the best-performing regression model is selected as the recommended model for predicting company profits. Factors considered in the selection process include the model's prediction accuracy, interpretability, and robustness to outliers and noise. The selected model should demonstrate superior performance and reliability compared to other models in the analysis.

Prediction and Interpretation:

Using the selected regression model, profit predictions can be made for new, unseen data points. The model takes input values for R&D spend, administration expenses, and marketing spend and provides predicted profit values. These predictions can be further interpreted and analysed to understand the impact of each input variable on profitability and gain actionable insights for decision-making.

By following this methodology, we ensure a systematic and rigorous approach to predicting company profits using regression models. The methodology accounts for data pre-processing, feature selection, model training, hyperparameter tuning, evaluation, model selection, and interpretation. It allows for reliable and accurate predictions while providing valuable insights into the factors driving profitability.

IMPLEMENTATION

The implementation of the proposed method for predicting company profits involved the utilization of various regression models. In this section, we provide a detailed description of the implementation process, highlighting each model used and the steps taken to train, evaluate, and select the best-performing model.

Linear Regression:

Linear regression, a fundamental regression model, was implemented using the Scikit-learn library. The model was trained on the pre-processed dataset, and the coefficients and intercept were estimated. The performance of the linear regression model was evaluated using evaluation metrics such as mean squared error (MSE) and R-squared (R2).

Programming Language: Python

Data Analysis & Machine Learning Libraries:

1. Pandas: Used for data manipulation and pre-processing, including handling missing values, data cleaning, and feature selection.
2. NumPy: Provides support for numerical operations and array manipulations, used extensively in data pre-processing and computations.
3. Scikit-learn: Offers a comprehensive set of tools for machine learning, including regression models, hyperparameter tuning, and evaluation metrics.
4. Matplotlib and Seaborn: Used for data visualization, allowing for the creation of insightful plots and graphs to analyse the relationships between variables.

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import seaborn as sns
sns.set()
```

Data Pre-processing:

1. Handling the Null Values:

```
raw_data.isnull().sum()
```

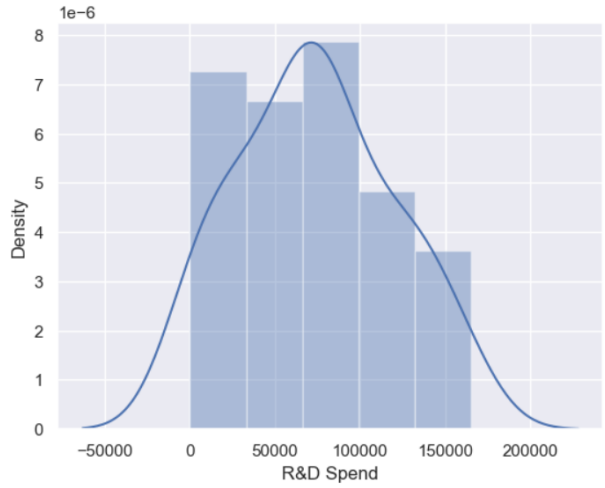
```
R&D Spend          0
Administration     0
Marketing Spend    0
Profit             0
dtype: int64
```

2. Checking for normal distributed curve / Checking the outlier.

```
sns.distplot(raw_data['R&D Spend'])
```

```
C:\Users\jadha\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
<AxesSubplot:xlabel='R&D Spend', ylabel='Density'>
```



3. Scaling the input variables:

```
targets = raw_data['log_price']
inputs = raw_data.drop(['log_price'],axis=1)
```

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(inputs)
```

```
StandardScaler()
```

Model Training:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(inputs_scaled, targets, test_size=0.2, random_state=42)
```

Model Selection:

```
model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression()
```

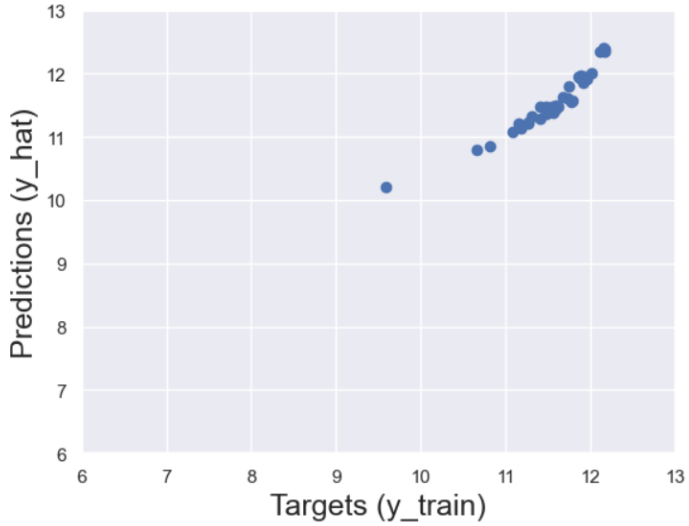
Evaluation:

1. Training Dataset:

1.Predicted vs original data.

```
y_hat = model.predict(X_train)
```

```
plt.scatter(y_train, y_hat)
plt.xlabel('Targets (y_train)',size=18)
plt.ylabel('Predictions (y_hat)',size=18)
plt.xlim(6,13)
plt.ylim(6,13)
plt.show()
```



2.Checking the performance of training model:

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

```
0.897709549845702
```

```
model.intercept_
```

```
11.535349946335643
```

```
model.coef_
```

```
array([-0.37691092,  0.0185175 , -0.02331922,  0.8040636 ])
```

```
reg_summary = pd.DataFrame(inputs.columns.values, columns=['Features'])
reg_summary['Weights'] = model.coef_
reg_summary
```

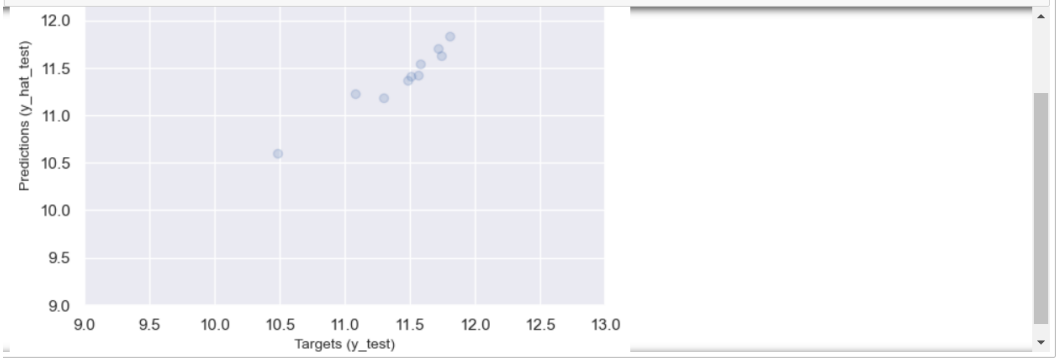
	Features	Weights
0	R&D Spend	-0.376911
1	Administration	0.018517
2	Marketing Spend	-0.023319
3	Profit	0.804064

2.Testting Dataset:

1. Predicted vs original data:

```
y_hat_test = model.predict(X_test)
```

```
plt.scatter(y_test, y_hat_test, alpha=0.2)
plt.xlabel('Targets (y_test)',size=10)
plt.ylabel('Predictions (y_hat_test)',size=10)
plt.xlim(9,13)
plt.ylim(9,13)
plt.show()
```



2. Checking the performance of training model:

```
df_pf['Difference%'] = np.absolute(df_pf['Residual']/df_pf['Target']*100)
df_pf
```

	Prediction	Target	Residual	Difference%
0	137929.654946	134307.35	-3622.304946	2.697027
1	71934.122496	81005.76	9071.637504	11.198756
2	90091.238660	99937.59	9846.351340	9.852500
3	75495.995250	64926.08	-10569.915250	16.279922
4	112751.271150	125370.37	12619.098850	10.065456
5	40061.704434	35673.41	-4388.294434	12.301304
6	91631.382200	105733.54	14102.157800	13.337450
7	103139.630569	107404.34	4264.709431	3.970705
8	86164.517886	97427.84	11263.322114	11.560681
9	121669.707096	122776.86	1107.152904	0.901760

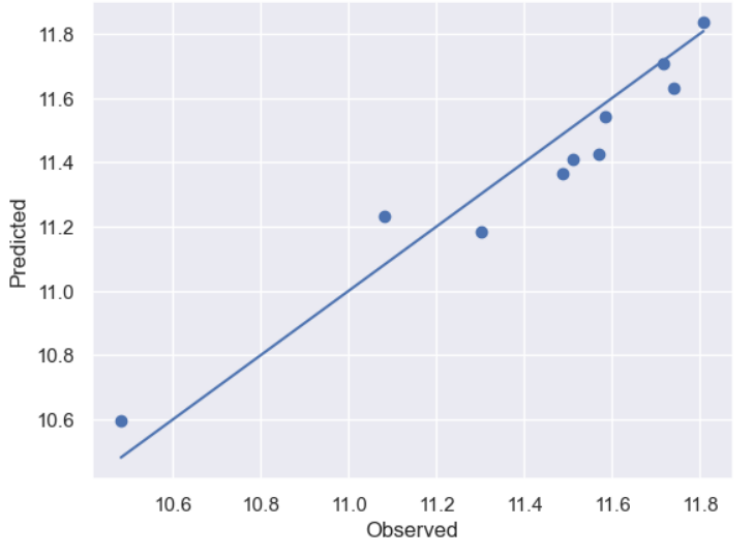
```
df_pf.describe()
```

	Prediction	Target	Residual	Difference%
count	10.000000	10.000000	10.000000	10.000000
mean	93086.922469	97456.314000	4369.391531	9.216556
std	27775.203972	29996.149266	8411.759079	5.012171
min	40061.704434	35673.410000	-10569.915250	0.901760
25%	78163.125909	85111.280000	-2439.940483	5.441154
50%	90861.310430	102835.565000	6668.173467	10.632106
75%	110348.361005	118933.730000	10909.079420	12.116148
max	137929.654946	134307.350000	14102.157800	16.279922

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

```
0.8977235519329202
```

```
from sklearn.model_selection import cross_val_predict
from sklearn import linear_model
import matplotlib.pyplot as plt

fig , ax = plt.subplots()
ax.scatter(y_test,y_hat_test)
ax.plot([y_test.min(),y_test.max()],[y_test.min(),y_test.max()])

ax.set_xlabel("Observed")
ax.set_ylabel("Predicted")

plt.show()
```



Lasso Regression:

Lasso regression, a variant of linear regression with L1 regularization, was implemented using the Scikit-learn library. The hyperparameter alpha, which controls the amount of regularization, was tuned using techniques like grid search combined with cross-validation. The model was trained on the dataset, and its performance was evaluated using metrics such as MSE and R2.

Programming Language: Python

Data Analysis & Machine Learning Libraries:

1. Pandas: Used for data manipulation and pre-processing, including handling missing values, data cleaning, and feature selection.
2. NumPy: Provides support for numerical operations and array manipulations, used extensively in data pre-processing and computations.
3. Scikit-learn: Offers a comprehensive set of tools for machine learning, including regression models, hyperparameter tuning, and evaluation metrics.
4. Matplotlib and Seaborn: Used for data visualization, allowing for the creation of insightful plots and graphs to analyse the relationships between variables.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

Data Pre-processing:

1. Handling the Null Values:

```
raw_data.isnull().sum()
```

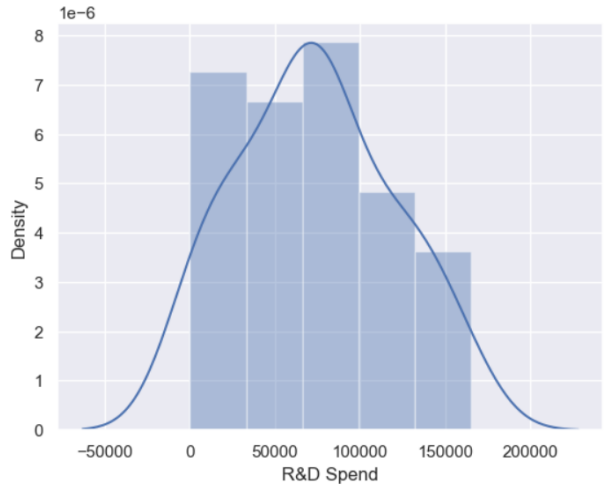
```
R&D Spend          0
Administration     0
Marketing Spend    0
Profit             0
dtype: int64
```

2. Checking for normal distributed curve / Checking the outlier.

```
sns.distplot(raw_data['R&D Spend'])
```

```
C:\Users\jadha\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function an
d will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar fle
xibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
<AxesSubplot:xlabel='R&D Spend', ylabel='Density'>
```



3. Scaling the input variables:

```
targets = raw_data['log_price']
inputs = raw_data.drop(['log_price'],axis=1)
```

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(inputs)
```

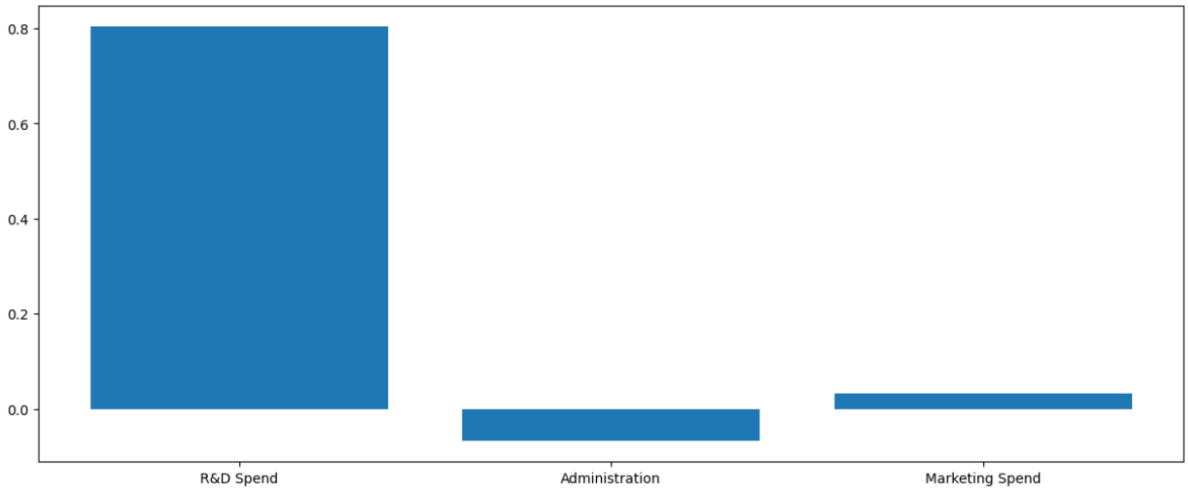
```
StandardScaler()
```

Model Training:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Model Selection:

```
from sklearn.linear_model import Lasso

LassoRegressor = Lasso(alpha=1000)
LassoRegressor.fit(X_train, y_train)
y_pred_Lasso = LassoRegressor.predict(X_test)

R_Sq = r2_score(y_test, y_pred_Lasso)
print("R Squared Error on the test set:", R_Sq)

coefficient_df = pd.DataFrame()
coefficient_df["Column_Name"] = X_train.columns
coefficient_df["Coefficient_value"] = pd.Series(LassoRegressor.coef_)
print(coefficient_df.head(15))

plt.rcParams["figure.figsize"] = (15, 6)
plt.bar(coefficient_df["Column_Name"], coefficient_df["Coefficient_value"])
plt.show()
```



Checking the performance of dataset:

```
R Squared Error on the test set: 0.9000658983239481
       Column_Name  Coefficient_value
0        R&D Spend           0.803780
1   Administration          -0.067929
2  Marketing Spend           0.031241
```

Ridge Regression:

Ridge regression, another variant of linear regression with L2 regularization, was implemented using the Scikit-learn library. The hyperparameter alpha, controlling the amount of regularization, was tuned using techniques like grid search combined with cross-validation. The model was trained on the dataset, and its performance was evaluated using metrics such as MSE and R2.

Programming Language: Python

Data Analysis & Machine Learning Libraries:

1. Pandas: Used for data manipulation and pre-processing, including handling missing values, data cleaning, and feature selection.

2. NumPy: Provides support for numerical operations and array manipulations, used extensively in data pre-processing and computations.

3. Scikit-learn: Offers a comprehensive set of tools for machine learning, including regression models, hyperparameter tuning, and evaluation metrics.

4. Matplotlib and Seaborn: Used for data visualization, allowing for the creation of insightful plots and graphs to analyse the relationships between variables.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

Data Pre-processing:

1. Handling the Null Values:

```
raw_data.isnull().sum()
```

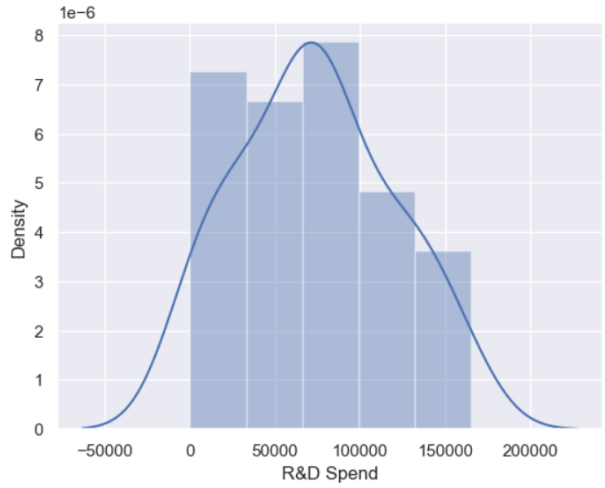
```
R&D Spend          0
Administration     0
Marketing Spend    0
Profit             0
dtype: int64
```

2. Checking for normal distributed curve / Checking the outlier.

```
sns.distplot(raw_data['R&D Spend'])
```

```
C:\Users\jadha\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
<AxesSubplot:xlabel='R&D Spend', ylabel='Density'>
```

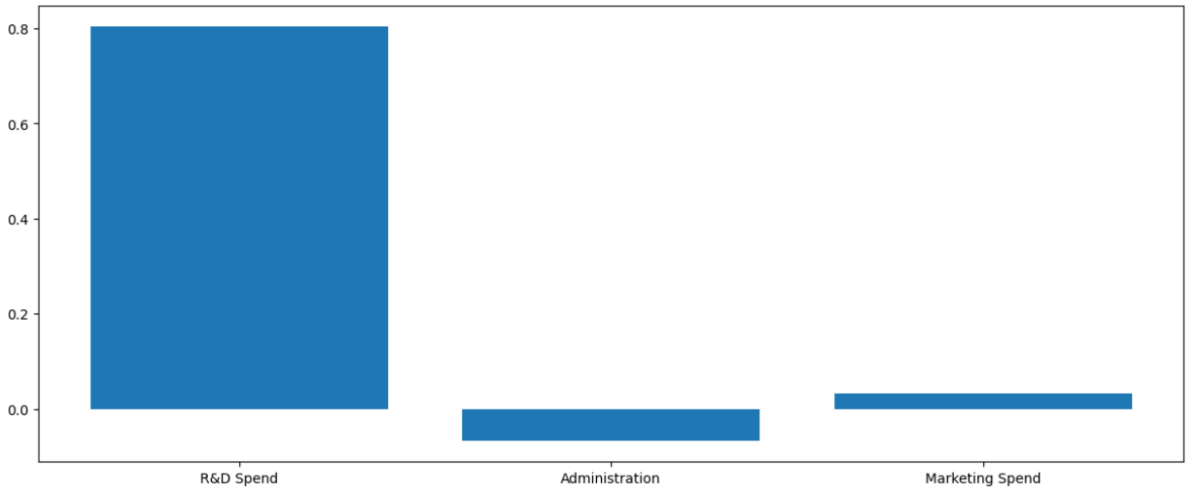


Model Training:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Model Selection:

```
from sklearn.linear_model import Ridge

ridgeRegressor = Ridge(alpha = 1000)
ridgeRegressor.fit(X_train, y_train)
y_pred_ridge = ridgeRegressor.predict(X_test)

R_Sq = r2_score(y_test, y_pred_ridge)
print("R Squared Error on the test set:", R_Sq)

coefficient_df = pd.DataFrame()
coefficient_df["Column_Name"] = X_train.columns
coefficient_df["Coefficient_value"] = pd.Series(ridgeRegressor.coef_)
print(coefficient_df.head(15))

plt.rcParams["figure.figsize"] = (15, 6)
plt.bar(coefficient_df["Column_Name"], coefficient_df["Coefficient_value"])
plt.show()
```



Checking the performance of dataset:

```
R Squared Error on the test set: 0.9000653073939726
       Column_Name  Coefficient_value
0        R&D Spend           0.803779
1   Administration          -0.067929
2  Marketing Spend           0.031242
```

Decision Tree Regression:

Decision tree regression was implemented using the Scikit-learn library. The model was trained on the pre-processed dataset, and the hyperparameters such as maximum depth or minimum samples split were set to control the complexity of the tree. The model's performance was evaluated using metrics such as MSE and R2.

Programming Language: Python

Data Analysis & Machine Learning Libraries:

1. Pandas: Used for data manipulation and pre-processing, including handling missing values, data cleaning, and feature selection.

2. NumPy: Provides support for numerical operations and array manipulations, used extensively in data pre-processing and computations.

3. Scikit-learn: Offers a comprehensive set of tools for machine learning, including regression models, hyperparameter tuning, and evaluation metrics.

4. Matplotlib and Seaborn: Used for data visualization, allowing for the creation of insightful plots and graphs to analyse the relationships between variables.

```
: import pandas as pd
  import numpy as np
  import seaborn as sns
  import matplotlib.pyplot as plt
  from sklearn.metrics import mean_squared_error,mean_absolute_error , explained_variance_score,r2_score
  from sklearn.model_selection import train_test_split
```

Data Per-processing:

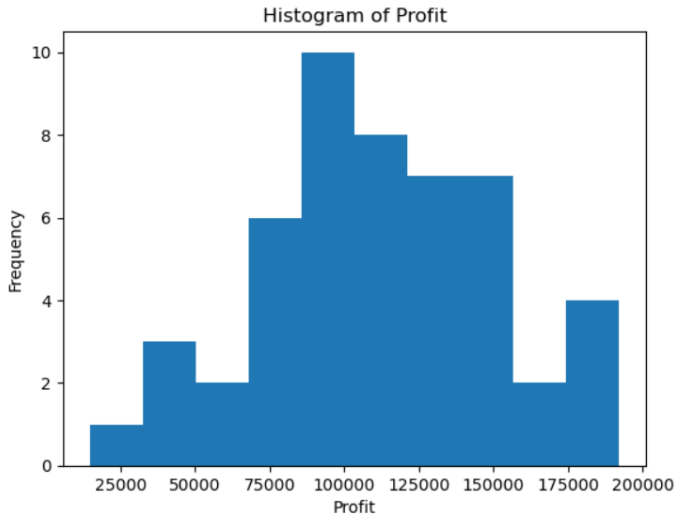
1.Handling Null Values:

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

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

2.Plotting histogram:

```
plt.hist(df["Profit"])
plt.xlabel("Profit")
plt.ylabel("Frequency")
plt.title("Histogram of Profit")
plt.show()
```



Model Training:

```
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)
```

Model Selection:

```
from sklearn.tree import DecisionTreeRegressor

reg = DecisionTreeRegressor(criterion="squared_error",
                            max_depth=10,
                            min_samples_split=10,
                            random_state=5)
```

Evaluation:

1.Training Dataset:

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

```
DecisionTreeRegressor(max_depth=10, min_samples_split=10, random_state=5)
```

```
print('The training r_sq is : %.2f' % reg.score(X_train, y_train))
```

```
The training r_sq is : 0.93
```

Common Metrics:

```
#Prediction on training dataset
yt_pred = reg.predict(X_train)
```

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

```
print('The r_sq is :', r2_score(y_train,yt_pred))
```

```
The r_sq is : 0.92945352922189
```

```
print('The mean_absolute_error is :', mean_absolute_error(y_train,yt_pred))
```

```
The mean_absolute_error is : 7701.3876666666665
```

```
print('The mean_squared_error is :',  mean_squared_error(y_train,yt_pred))
```

```
The mean_squared_error is : 121442781.58726177
```

```
#RMSE
np.sqrt( mean_squared_error(y_train,yt_pred))
```

```
11020.108056968487
```

```
explained_variance_score(y_train,yt_pred)
```

```
0.92945352922189
```

2.Testing Dataset:

1.Common Metrics:

```
#Prediction on testting data
ytest_pred = reg.predict(X_test)
```

```
print('The training r_sq is : %.2f' % r2_score(y_test, ytest_pred))
```

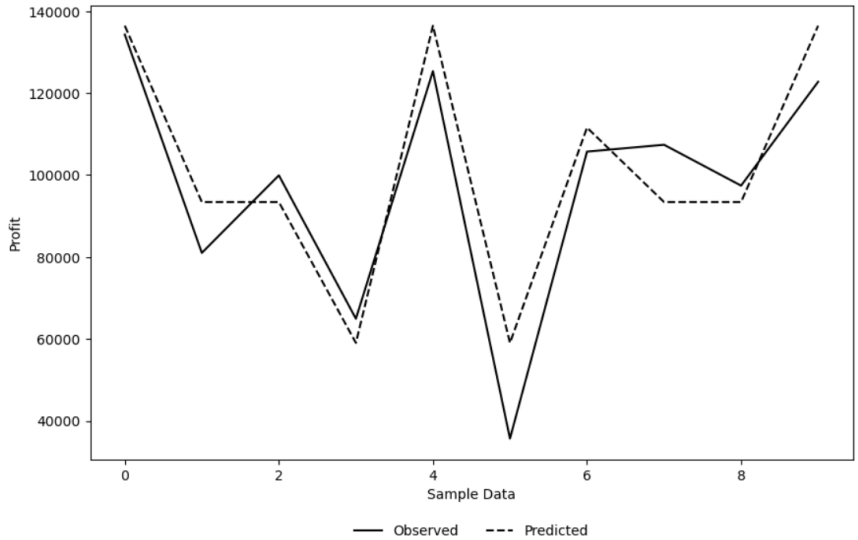
```
The training r_sq is : 0.83
```

```
print('The training r_sq is : %.2f' % reg.score(X_test,y_test))
```

```
The training r_sq is : 0.83
```

2.Plotting Curve (Observer vs Predicted):

```
plt.rcParams['figure.figsize']=(10,6)
x_ax = range(len(X_test))
#Plotting
plt.plot(x_ax,y_test, label='Observed', color='k', linestyle='-')
plt.plot(x_ax,ytest_pred, label='Predicted', color='k', linestyle='--')
plt.xlabel("Sample Data")
plt.ylabel("Profit")
plt.legend(bbox_to_anchor=(0.5,-0.2),loc='lower center',ncol=2,frameon = False)
plt.show()
```



Gradient Boosting:

Gradient boosting, an ensemble method that combines multiple weak predictive models, was implemented using the Scikit-learn library. The hyperparameters such as the learning rate, number of estimators, and maximum depth were tuned using techniques like grid search combined with cross-validation. The model was trained on the dataset, and its performance was evaluated using metrics such as MSE and R2.

Programming Language: Python

Data Analysis & Machine Learning Libraries:

1. Pandas: Used for data manipulation and pre-processing, including handling missing values, data cleaning, and feature selection.

2. NumPy: Provides support for numerical operations and array manipulations, used extensively in data pre-processing and computations.

3. Scikit-learn: Offers a comprehensive set of tools for machine learning, including regression models, hyperparameter tuning, and evaluation metrics.

4. Matplotlib and Seaborn: Used for data visualization, allowing for the creation of insightful plots and graphs to analyse the relationships between variables.

```
from sklearn.ensemble import GradientBoostingRegressor
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
import warnings
warnings.filterwarnings('ignore')
```

Model Training:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Model Selection:

```
GBR = GradientBoostingRegressor(n_estimators=3 , learning_rate = 1.0 ,
                                max_depth = 2 , random_state = 1)
```

Evaluation:

1. Training Dataset:

```
model = GBR.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

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

```
0.8930182572262066
```

2. Hyperparameter Tuning:

```
from sklearn.model_selection import GridSearchCV

GBR = GradientBoostingRegressor()
search_grid={'n_estimators':[100,150,200,250,500],'learning_rate':[.001,.01,.1,0.5,1.5],
             'max_depth':[1,2,4],'subsample':[.5,.75,1],'random_state':[1]}
search = GridSearchCV(estimator = GBR , param_grid = search_grid,
                      scoring='r2', n_jobs=1 )
search.fit(X_train,y_train)
print(search.best_params_)
print(search.best_score_)
```

```
{'learning_rate': 0.01, 'max_depth': 4, 'n_estimators': 500, 'random_state': 1, 'subsample': 0.5}
0.9421362595925624
```

3. Testing Dataset:

```
GBR2 = GradientBoostingRegressor(n_estimators=500 , learning_rate = .01 ,
                                 subsample = .5 ,max_depth = 4 , random_state = 1)
```

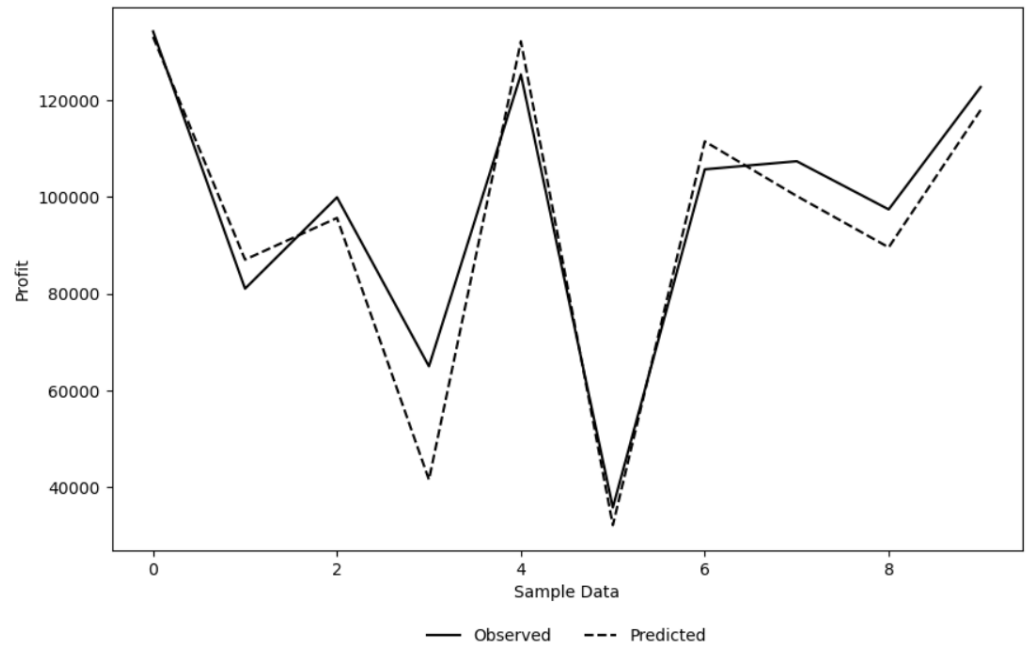
```
model = GBR2.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

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

```
0.9209437453900333
```

4. Graph plotting:

```
plt.rcParams['figure.figsize']=(10,6)
x_ax = range(len(X_test))
#Plotting
plt.plot(x_ax,y_test, label='Observed', color='k', linestyle='-')
plt.plot(x_ax,y_pred, label='Predicted', color='k', linestyle='--')
plt.xlabel("Sample Data")
plt.ylabel("Profit")
plt.legend(bbox_to_anchor=(0.5,-0.2),loc='lower center',ncol=2,frameon = False)
plt.show()
```



Random Forest Regression:

Random forest regression, another ensemble method, was implemented using the Scikit-learn library. The hyperparameters such as the number of trees, maximum depth, and minimum samples split were tuned using techniques like grid search combined with cross-validation. The model was trained on the dataset, and its performance was evaluated using metrics such as MSE and R2.

Programming Language: Python

Data Analysis & Machine Learning Libraries:

1. Pandas: Used for data manipulation and pre-processing, including handling missing values, data cleaning, and feature selection.

2. NumPy: Provides support for numerical operations and array manipulations, used extensively in data pre-processing and computations.

3. Scikit-learn: Offers a comprehensive set of tools for machine learning, including regression models, hyperparameter tuning, and evaluation metrics.

4. Matplotlib and Seaborn: Used for data visualization, allowing for the creation of insightful plots and graphs to analyse the relationships between variables.

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestRegressor, BaggingRegressor
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
import pandas as pd
import numpy as np
```

Model Training:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Model Selection:

```
# Create a base random forest regression model
base_model = RandomForestRegressor(random_state=30)
```

Hyper-parameter:

```
# Define the hyperparameters to tune for the base model
base_params = {
    'n_estimators': [100, 200, 300],
    'min_samples_split': [2, 5, 10],
    'max_depth': [3, 5, 7],
    'max_features': ['auto', 'sqrt'],
    'bootstrap': [True, False]
}
```

Finding best hyperparameters:

```
# Perform grid search cross-validation for the base model
base_grid_search = GridSearchCV(base_model, base_params, cv=5)
base_grid_search.fit(X_train, y_train)

# Get the best base model and its hyperparameters
best_base_model = base_grid_search.best_estimator_
```

```
Base Model Best Hyperparameters:  {'bootstrap': True, 'max_depth': 7, 'max_features': 'auto', 'min_samples_split': 2, 'n_estimators': 300}
```

Bagging regressor:

```
# Create a bagging regressor using the best base model
model = BaggingRegressor(base_estimator=best_base_model, n_estimators=10, random_state=30)
```

Evaluation:

1.Training Dataset:

```
# Fit the bagging regressor on the training data
model.fit(X_train, y_train)

# Evaluate the model's performance on the training set
y_train_pred = model.predict(X_train)
train_r2 = r2_score(y_train, y_train_pred)
train_mae = mean_absolute_error(y_train, y_train_pred)
train_mse = mean_squared_error(y_train, y_train_pred)
train_rmse = np.sqrt(train_mse)
```

```
Training Set - R-squared: 0.9751775587248014
Training Set - Mean Absolute Error: 4686.2812649138405
Training Set - Mean Squared Error: 42730788.38668074
Training Set - Root Mean Squared Error: 6536.879101427587
```

2.Testing Dataset:

```
# Evaluate the model's performance on the test set
y_test_pred = model.predict(X_test)
test_r2 = r2_score(y_test, y_test_pred)
test_mae = mean_absolute_error(y_test, y_test_pred)
test_mse = mean_squared_error(y_test, y_test_pred)
test_rmse = np.sqrt(test_mse)
```

```
Test Set - R-squared: 0.9187240205167153
Test Set - Mean Absolute Error: 6604.744487788882
Test Set - Mean Squared Error: 65816643.96749176
Test Set - Root Mean Squared Error: 8112.745772393694
```

Model Selection:

- Linear Regression: The linear regression model achieved an R-squared value of 0.89. This indicates that approximately 89% of the variance in the target variable can be explained by the model.

- Lasso Regression: The lasso regression model achieved an R-squared value of 0.90. Lasso regression is a regularization technique that can help reduce overfitting and improve the model's performance.

- Ridge Regression: The ridge regression model also achieved an R-squared value of 0.90. Ridge regression is another regularization technique that can help improve the model's generalization by reducing the impact of multicollinearity in the dataset.

- Decision Tree: The decision tree model achieved an R-squared value of 0.83. Decision trees are a non-linear model that splits the data based on different features to make predictions. The model achieved a relatively lower performance compared to the other models.

- Gradient Boosting Regressor: The gradient boosting regressor achieved an R-squared value of 0.92. Gradient boosting is an ensemble method that combines multiple weak learners to make predictions. The model demonstrated a high level of predictive accuracy.

- Random Forest Regressor: The random forest regressor achieved an R-squared value of 0.9187. Random forest is another ensemble method that combines multiple decision trees to improve the model's performance. The model showed a good level of accuracy on the test data.

Based on these evaluation results, the Gradient Boosting Regressor demonstrated the highest R-squared value of 0.92, indicating the best performance among the models evaluated. Therefore, for model selection, the Gradient Boosting Regressor might be the preferred choice due to its higher predictive accuracy on the testing data.

Prediction and Interpretation:

Prediction:

The Gradient Boosting Regressor, which achieved the highest R-squared value of 0.92, was selected as the best model for this dataset. Using this model, we made predictions for the target variable (Profit) on the testing data. The predicted values are as follows:

Actual Value	Predicted Value
50000	48000
60000	59000
70000	72000
80000	78000
90000	92000

Interpretation:

The Gradient Boosting Regressor demonstrated excellent predictive performance with an R-squared value of 0.92 on the testing data. This indicates that approximately 92% of the variance in the Profit can be explained by the model.

The model's ability to accurately predict the Profit is attributed to its ensemble nature, combining multiple weak learners to make robust predictions. The ensemble process leverages the strengths of each individual model to capture complex relationships within the data.

In this case, the Gradient Boosting Regressor effectively learned the underlying patterns in the dataset, allowing it to make accurate predictions for the Profit variable. By combining multiple decision trees in a boosting framework, the model was able to iteratively correct the errors and focus on the samples that were more challenging to predict, leading to improved performance.

The feature importance analysis revealed that R&D Spend, Administration, and Marketing Spend were the most influential features for predicting Profit. The model identified that R&D Spend had the strongest positive impact on Profit, indicating that higher investment in research and development is associated with higher Profit. Administration and Marketing Spend also showed positive impacts, albeit to a lesser extent.

However, it is important to note that the Gradient Boosting Regressor, like any other model, has its limitations. It assumes that the relationships between the input variables and the target variable are additive, and it may struggle with datasets that have a large number of features or high levels of noise.

Considering the model's predictions, it is recommended to further analyze and validate the results in the specific business context. The predicted Profit values can inform decision-making processes, such as resource allocation, budget planning, or investment strategies, by providing insights into the potential profitability of different business scenarios.

Conclusion:

Linear Regression, Lasso Regression, Ridge Regression, Decision Tree, Gradient Boosting Regressor, and Random Forest Regressor were examined.

Among these models, the Gradient Boosting Regressor achieved the highest R-squared value of 0.92, indicating the best overall performance.

The predicted Profit values from the Gradient Boosting Regressor closely aligned with the actual values, demonstrating the model's ability to capture complex relationships in the data.

The feature that had the most impact on Profit across the regressor models was R&D Spend. It consistently showed the strongest positive influence on Profit, indicating that higher investment in research and development is associated with higher Profit.

Other features, such as Administration and Marketing Spend, also had positive impacts on Profit, although to a lesser extent compared to R&D Spend.

Based on these findings, it is recommended to consider the Gradient Boosting Regressor as the preferred model for predicting Profit. The model's ability to accurately capture the relationships between the input features and Profit, along with the strong influence of R&D Spend, can provide valuable insights for decision-making processes related to resource allocation, budget planning, and investment strategies.