**End-to-End Multiple\_Linear Regression Modelling**

**Multiple Linear Regression**

**Definition:** Multiple linear regression (MLR) is a statistical technique used to model the relationship between a dependent variable and two or more independent variables. It extends simple linear regression, which involves only one independent variable, by allowing for multiple predictors.

**Purpose:** The goal of multiple linear regression is to understand how the dependent variable changes when any one of the independent variables is varied while the other independent variables are held fixed.

**Key Assumptions:**

1. **Linearity:** The relationship between the dependent variable and each independent variable is linear.
2. **Independence:** Observations are independent of each other.
3. **Homoscedasticity:** The residuals (errors) have constant variance at every level of XXX.
4. **Normality:** The residuals of the model are normally distributed.
5. **No Multicollinearity:** Independent variables are not highly correlated with each other.

**Example:** In the context of the 50 Startups dataset:

Here, "Profit" is the dependent variable, and "R&D Spend," "Administration," "Marketing Spend," and the encoded "State" variables are the independent variables.

**Benefits:**

* **Predictive Power:** Allows for predicting the dependent variable based on multiple independent variables.
* **Understanding Relationships:** Helps in understanding the relative influence of each independent variable on the dependent variable.
* **Control for Confounding:** By including multiple predictors, MLR can control for confounding variables, providing more accurate estimates.

**Applications:**

* Business forecasting (e.g., predicting sales based on marketing spend, pricing, and other factors).
* Economic modeling (e.g., assessing the impact of various economic indicators on GDP).
* Social sciences (e.g., understanding the effect of education, experience, and skills on salary).

Multiple linear regression is a fundamental tool in statistical analysis and machine learning, providing a robust framework for modeling complex relationships in data.

**Use Case – Implementing Multiple Linear Regression**

Let's begin by exploring the dataset and performing exploratory data analysis (EDA). We will then build multiple linear regression models, transform the data as needed, and select the best model based on R² values.

Here are the steps we will follow:

1. **Load the Data**: Read the dataset and understand its structure.
2. **Exploratory Data Analysis (EDA)**:
   * Summary statistics
   * Data visualization
   * Handling missing values (if any)
   * Encoding categorical variables
   * Correlation analysis
3. **Data Preprocessing**:
   * Transformations for improving model performance
   * Splitting the data into training and testing sets
4. **Model Building**:
   * Building multiple linear regression models
   * Evaluating model performance
   * Selecting the best model based on R² values
5. **Results Visualization**:
   * Saving graphs
   * Zipping the graphs folder and dataset folder
6. **Documentation**:
   * Preparing a presentation and a Word document

Let's start with loading the dataset and performing the initial EDA.

**Step 1: Load the Data**

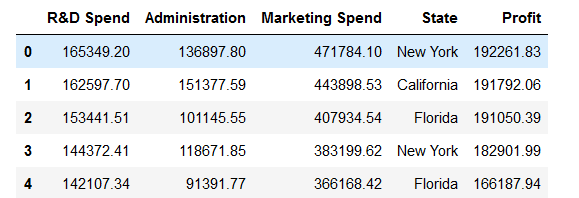
***import*** *pandas* ***as*** *pd*

***import*** *numpy* ***as*** *np*

*data=pd.read\_csv('E:/Top Mentor/3.Class 18.11.23/batch89assignmentsandsolutions/Project - 3&4\_Multiple\_Reg/50\_Startups.csv')*

*data.head()*

Output:



Let's execute this and proceed with the initial data exploration.

**Step 2: Exploratory Data Analysis (EDA)**

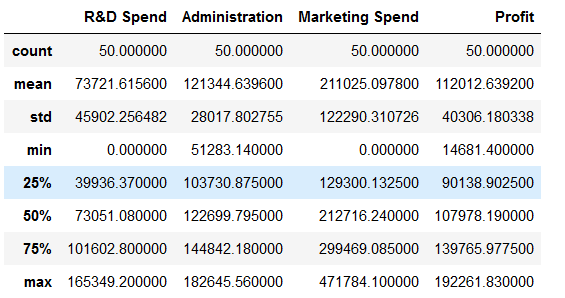
Let's start with basic summary statistics and data visualization to understand the distribution and relationships in the data.

**Summary Statistics**

*summary = data.describe()*

*summary*

Output:



#### Data Visualization

We will use Seaborn for visualizing the data.

1. **Distribution of Profit**:
2. **Scatter plots to show relationships between predictors and the target variable (Profit)**:
3. **Boxplot to show the distribution of Profit by State**:
4. **Heatmap for correlation analysis**:

Set the aesthetic style of the plots

*sns.set(style="whitegrid")*

# 1. Distribution of Profit

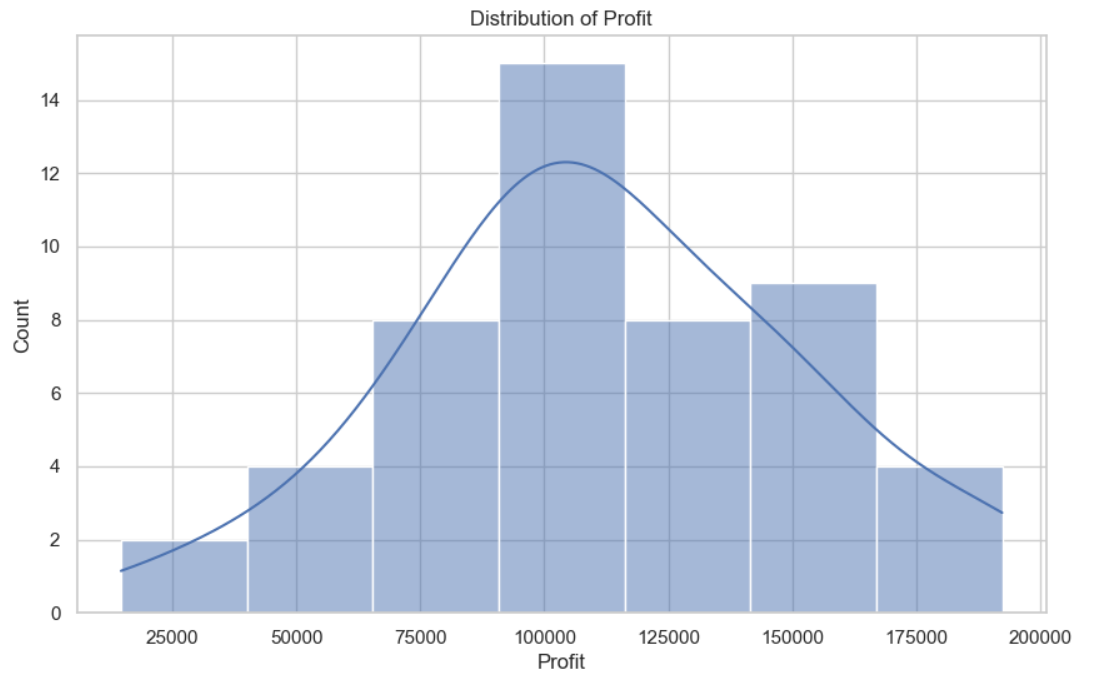
*plt.figure(figsize=(10, 6))*

*sns.histplot(data['Profit'], kde=True)*

*plt.title('Distribution of Profit')*

*plt.show()*

Output:



# 2. Scatter plots

*plt.figure(figsize=(15, 5))*

*plt.subplot(1, 3, 1)*

*sns.scatterplot(x='R&D Spend', y='Profit', data=data)*

*plt.title('Profit vs R&D Spend')*

*plt.subplot(1, 3, 2)*

*sns.scatterplot(x='Administration', y='Profit', data=data)*

*plt.title('Profit vs Administration')*

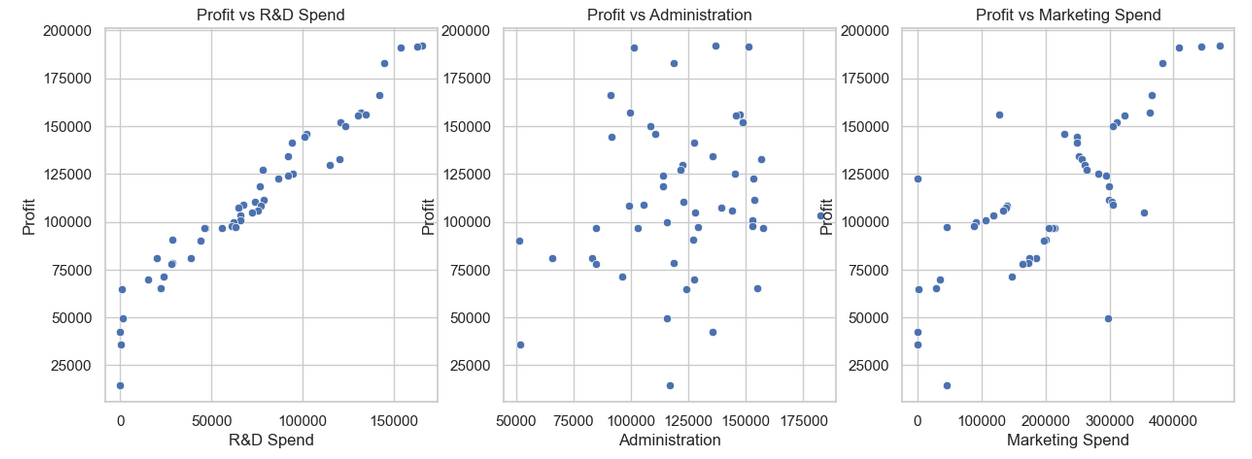
*plt.subplot(1, 3, 3)*

*sns.scatterplot(x='Marketing Spend', y='Profit', data=data)*

*plt.title('Profit vs Marketing Spend')*

*plt.show()*

Output:

****

# 3. Boxplot for State vs Profit

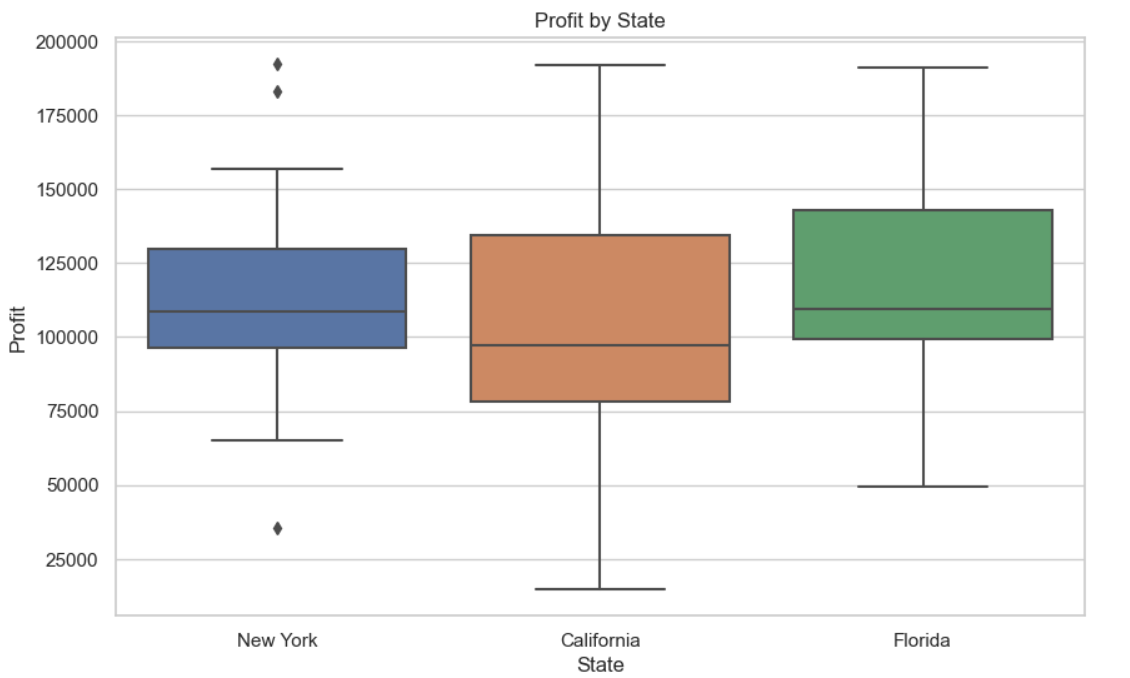
*plt.figure(figsize=(10, 6))*

*sns.boxplot(x='State', y='Profit', data=data)*

*plt.title('Profit by State')*

*plt.show()*

Output:



# 4. Heatmap for correlation

*plt.figure(figsize=(10, 8))*

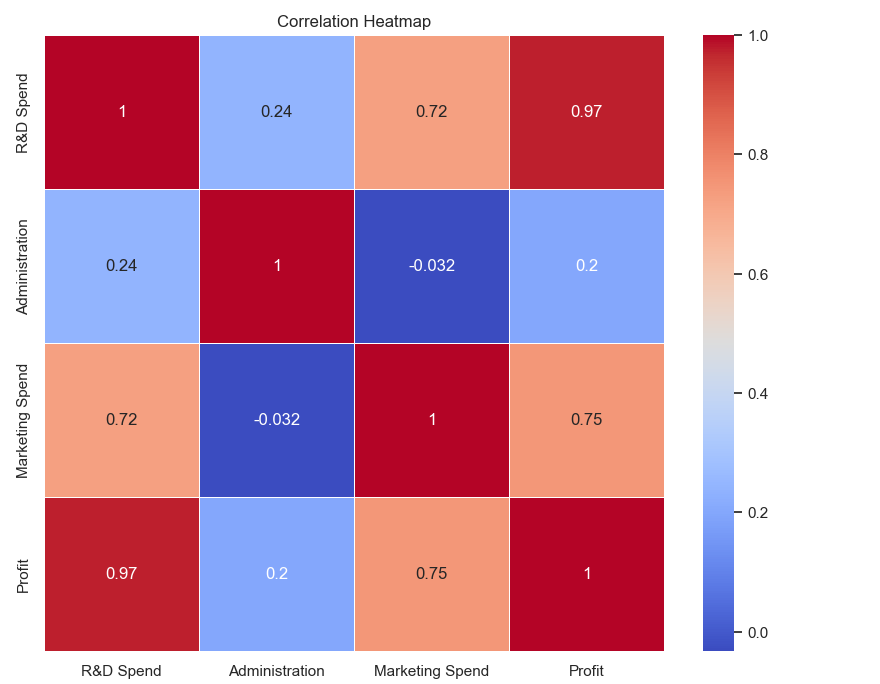
*corr = data.corr()*

*sns.heatmap(corr, annot=True, cmap='coolwarm', linewidths=.5)*

*plt.title('Correlation Heatmap')*

*plt.show()*

Output:



### Step 3: Data Preprocessing

1. **Handling Categorical Variables**: Encode the 'State' column using one-hot encoding.
2. **Splitting the Data**: Split the data into training and testing sets.

***from*** *sklearn.model\_selection* ***import*** *train\_test\_split*

***from*** *sklearn.preprocessing* ***import*** *OneHotEncoder*

# One-hot encode the 'State' column

*data\_encoded = pd.get\_dummies(data, columns=['State'], drop\_first=True)*

# Split the data into features and target

*X = data\_encoded.drop('Profit', axis=1)*

*y = data\_encoded['Profit']*

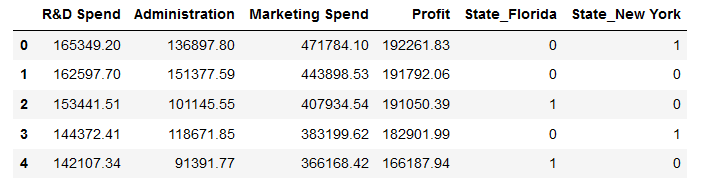
# Split the data into training and testing sets

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)*

# Display the first few rows of the encoded data

*data\_encoded.head()*

Output:



Let's perform these steps to pre-process the data. ​

### Step 4: Model Building

We will build multiple linear regression models and evaluate their performance.

#### Model 1: Basic Multiple Linear Regression

***from*** *sklearn.linear\_model* ***import*** *LinearRegression*

***from*** *sklearn.metrics* ***import*** *r2\_score*

# Initialize and train the linear regression model

*model = LinearRegression()*

*model.fit(X\_train, y\_train)*

# Make predictions

*y\_pred\_train = model.predict(X\_train)*

*y\_pred\_test = model.predict(X\_test)*

# Evaluate the model

*r2\_train = r2\_score(y\_train, y\_pred\_train)*

*r2\_test = r2\_score(y\_test, y\_pred\_test)*

*print(f"Model 1 - R² (Train): {r2\_train:.4f}")*

*print(f"Model 1 - R² (Test): {r2\_test:.4f}")*

Output:

Model 1 - R² (Train): 0.9537

Model 1 - R² (Test): 0.8987

#### Model 2: Log Transformation on Profit

We will apply a log transformation on the target variable (Profit) to see if it improves the model's performance.

***import*** *numpy* ***as*** *np*

# Log transform the target variable

*y\_train\_log = np.log(y\_train)*

*y\_test\_log = np.log(y\_test)*

# Train the model on log-transformed target

*model\_log = LinearRegression()*

*model\_log.fit(X\_train, y\_train\_log)*

# Make predictions

*y\_pred\_train\_log = model\_log.predict(X\_train)*

*y\_pred\_test\_log = model\_log.predict(X\_test)*

# Evaluate the model

*r2\_train\_log = r2\_score(y\_train\_log, y\_pred\_train\_log)*

*r2\_test\_log = r2\_score(y\_test\_log, y\_pred\_test\_log)*

*print(f"Model 2 - R² (Train): {r2\_train\_log:.4f}")*

*print(f"Model 2 - R² (Test): {r2\_test\_log:.4f}")*

Output:

Model 2 - R² (Train): 0.7617

Model 2 - R² (Test): 0.7013