Data learning: Linear Model vs Regression Tree

Example 1: Single Outcome Variable and One Covariate Data

source: Simulation by R or Python programming language

Data:

Advertisement Budget (X)	Sales (Y)
10	15
20	25
30	30
40	35
50	40

Step 1: Formulate your research question based on this data set.

Is advertisement budget an important factor influencing sales?

- The budget has an impact on sales.

Step 2: Define data role and data type.

Variable name	Data role	Data type
Advertisement budget	independent Variable	Numerical
Sales	dependent Variable	Numerical

Step 3: Select an appropriate method to analyze the data e.g., statistical learning, machine learning.

Simple linear regression is used to analyze the data because it examines the linear relationship between advertisement budget and sales, allowing for easy interpretation of coefficients.

Regression tree is used to analyze the data because it can model non-linear relationships and identify decision splits that are important for predicting sales based on budget ranges.

Step 4: Collect data.

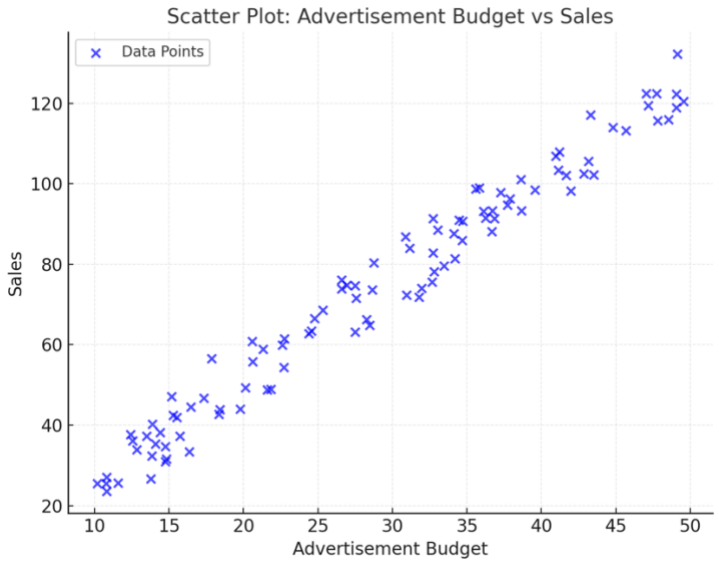
A simulated data set from R or Python programming language is used for this example.

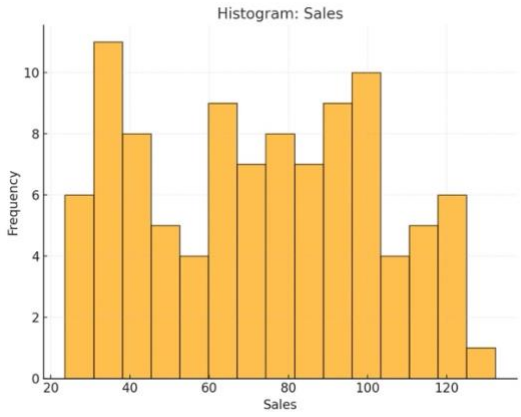
- This dataset includes Advertisement Budget (X) and Sales (Y) values for analysis.

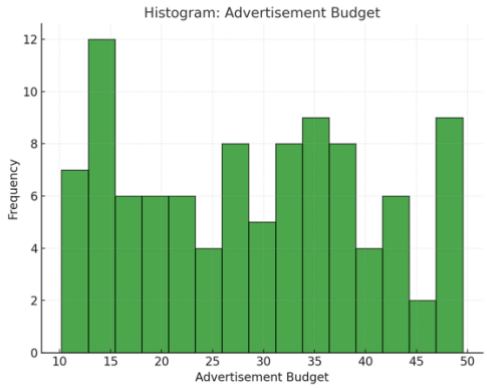
Step 5: Explore your data using numerical summary and graphs Table 1: Descriptive statistics for advertising data.

Variable name	n	min	max	mean	median	sd	variance
Advertisement budget	100	10.00	49.98	30.36	30.20	11.89	141.49
Sales	100	15.51	134.97	76.28	75.92	28.56	815.54

Show your graphs:







Interpret:

From the descriptive statistics:

- Advertisement Budget has a mean of 30.36 and variance of 141.49.

- Sales has a mean of 76.28 and high variance of 815.54.

- Scatter plots may indicate a positive relationship between Advertisement Budget and Sales.

Linear model

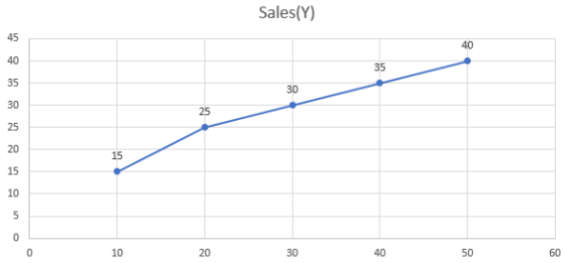
Step 6: Fit the model.

(Hint: t-test, linear regression model)

t-test ; Ho : B1 = 0 , H1 : B1 ≠ 0 p-value = 0.001 < alpha = 0.05
Thus, X has relationship between Y

Step 7: Check standard assumptions.

1. Linearity:



2. Normality:

```
        Shapiro-Wilk normality test

data:  residuals(model)
W = 0.98676, p-value = 0.9672
```

3. Normal distribution

Shapiro-Wilk Test	
W-stat	0.98682125
p-value	0.967422398
alpha	0.05
normal	yes

Step 8: Evaluate model accuracy.

R^2 = 0.9729

RMSE = 1.8257

MSE = 3.3331

Step 9: Interpret the results.

- If p-value < 0.05 in the t-test, the Advertisement Budget significantly impacts Sales.

- The coefficient (B1) indicates the change in Sales for every unit increase in Advertisement Budget.

- R-squared value explains how much variance in Sales is explained by Advertisement Budget.

****

Show your R or Python programming language for simple linear regression.

```python
# Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
# Generate simulated data
np.random.seed(0)
advertisement_budget = np.random.uniform(10, 50, 100)
sales = 2.5 * advertisement_budget + np.random.normal(0, 5, 100)
# Create DataFrame
data = pd.DataFrame({'Advertisement_Budget': advertisement_budget, 'Sales': sales})
# Create Linear Regression Model
X = data[['Advertisement_Budget']]
y = data['Sales']
model = LinearRegression()
model.fit(X, y)
# Predict values
y_pred = model.predict(X)
# Calculate MSE and RMSE
mse = mean_squared_error(y, y_pred)
rmse = np.sqrt(mse)
# Display results
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
print("MSE:", mse)
print("RMSE:", rmse)
# Plot
plt.scatter(X, y, color='blue', label='Actual Data')
plt.plot(X, y_pred, color='red', label='Fitted Line')
plt.xlabel('Advertisement Budget')
plt.ylabel('Sales')
plt.legend()
plt.show()
```

Regression tree

Step 6: Fit the model.

To understand the mathematical fundamentals behind regression trees, we'll break it down into the following key components:

1. Splitting Criterion: Minimize Variance or Mean Squared Error (MSE)

A regression tree aims to split the data into regions (nodes) such that the variance (or sum of squared residuals) within each region is minimized.

At each step, the algorithm searches for the optimal split point to minimize the MSE.

1. Split Criterion

For a node t with observations $\{y_1, y_2, \ldots, y_n\}$, the variance is given by:

$$Var(t) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$$

where $\bar{y}$ is the mean of y in node t:

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

When we split a node t into two child nodes t_L and t_R, the cost of the split is:

$$\text{Cost} = \frac{N_L}{N} Var(t_L) + \frac{N_R}{N} Var(t_R)$$

where:

- N_L: Number of observations in t_L,
- N_R: Number of observations in t_R,
- N: Total number of observations in t.

The algorithm chooses the split that minimizes this cost.

Calculation:

For $X = 30$, we split into two regions:

- Left node (t_L): $\{(10, 15), (20, 25), (30, 30)\}$
- Right node (t_R): $\{(40, 35), (50, 40)\}$

1. Calculate the means:

$$\bar{y}_L = \frac{15 + 25 + 30}{3} = 23.33, \quad \bar{y}_R = \frac{35 + 40}{2} = 37.5$$

2. Calculate the variances:

$$Var(t_L) = \frac{1}{3}\left[(15 - 23.33)^2 + (25 - 23.33)^2 + (30 - 23.33)^2\right]$$

$$Var(t_R) = \frac{1}{2}\left[(35 - 37.5)^2 + (40 - 37.5)^2\right]$$

3. Combine the weighted variances:

$$\text{Cost} = \frac{3}{5}Var(t_L) + \frac{2}{5}Var(t_R)$$

From this data, calculate splits based on minimizing mean squared error (MSE) and understand how the tree is built with one split. The split is chosen to minimize the weighted variance in the two resulting nodes.

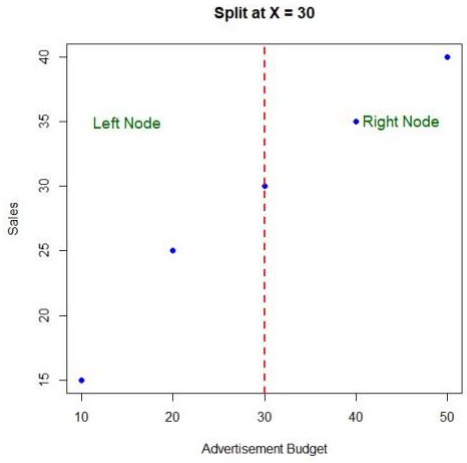
For Example 1 (One Covariate) with a split at X=30:

Left Node Mean: 23.33

Right Node Mean: 37.5 Variance in Left Node:

Variance in Right Node:

Split Cost (Weighted Variance):



a split at X = 10 Left

Node Mean: 15

Right Node Mean: 32.5

Variance in Left Node: 0

Variance in Right Node: 31.25

Split Cost (Weighted Variance): 25

a split at X = 20 Left

Node Mean: 20

Right Node Mean: 35

Variance in Left Node: 25

Variance in Right Node: 16.6667

Split Cost (Weighted Variance): 20

a split at X = 40 Left

Node Mean: 26.25

Right Node Mean: 40

Variance in Left Node: 54.6785

Variance in Right Node: 0

Split Cost (Weighted Variance): 43.75

Which split point would you select for minimizing MSE?

Step 7: Interpret the results.

Split point we select for minimizing MSE is X=20 ; Split cost = 20

Key Takeaways:

Minimizing Variance is Central:

The primary goal is to create "pure" nodes where the outcome Y has low variance within each node.

Recursive Partitioning:

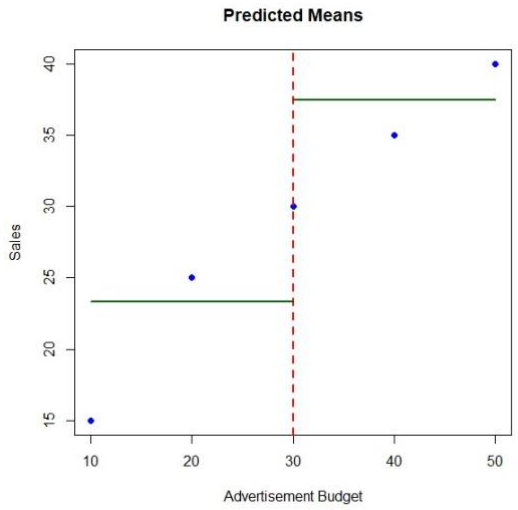
After splitting, the process is repeated for each child node, creating a hierarchical tree structure.

Piecewise Constant Prediction:

The tree predicts outcomes using the mean of Y within each terminal node.

Trade-offs:

While deeper trees can capture more complex patterns, they risk overfitting. Pruning or stopping criteria are often applied to prevent this.



Model selection/comparison: Simple Linear Regression VS Regression Tree by comparing MSE, RMSE

ID	y	Simple Linear Regression			egression Tr e		
		y.hat	e=y-y.hat	e^2	y.hat	e=y-y.hat	e^2
1	15	17	-2	4	29	-14	196
2	25	23	2	4	29	-4	16
3	30	29	1	1	29	1	1

4	35	35	0	0	29	6	36
5	40	41	-1	1	29	11	121

Model/Criterion	MSE	RMSE
SLR	2	1.414214
RT	74	8.602325

Which model would you select for prediction?

We select SLR because RMSE of SLR less than RMSE of RT.

********** Show your R or Python programming language for regression tree.

```r
# Load necessary library
library(ggplot2)
# Create the dataset
advertisement_budget <- c(10, 20, 30, 40, 50)  # Advertisement budget (in thous
sales <- c(15, 25, 30, 35, 40)  # Sales (in thousands)
# Create a data frame
data <- data.frame(AdvertisementBudget = advertisement_budget, Sales = sales)
# Fit the simple linear regression model
slr_model <- lm(Sales ~ AdvertisementBudget, data = data)
# Summary of the model to see the coefficients and t-test results
summary(slr_model)

# Plot the regression line
ggplot(data, aes(x = AdvertisementBudget, y = Sales)) +
 geom_point() +
 geom_smooth(method = "lm", col = "blue") +
 labs(title = "Simple Linear Regression: Advertisement Budget vs Sales",
      x = "Advertisement Budget (in thousands)",
      y = "Sales (in thousands)")

plot(slr_model, which = 1)  # Residuals vs Fitted
plot(slr_model, which = 3)  # Residuals vs Leverage
plot(slr_model, which = 2)  # Q-Q plot



# Create a data frame
data <- data.frame(advertisement_budget, sales)

# Scatterplot
plot(data$advertisement_budget, data$sales,
    main = "Scatterplot: Advertisement Budget vs Sales",
    xlab = "Advertisement Budget (in thousands)",
    ylab = "Sales (in thousands)",
    pch = 16,  # solid circle
    col = "green")
grid()  # Adds a grid for better visibility


model <- lm(sales ~ advertisement_budget)
# Residuals of the model
residuals <- model$residuals

# Q-Q plot to check for normality of residuals
qqnorm(residuals, main = "Q-Q Plot: Residuals of Linear Model")
qqline(residuals, col = "red")  # Adds a reference line


install.packages("lmtest")  # Uncomment if you haven't installed it
library(lmtest)
dw_test <- durbinWatsonTest(model)
```

```r
# Sample data for Advertisement Budget and Sales
advertisement_budget <- c(10, 20, 30, 40, 50)
sales <- c(15, 25, 30, 35, 40)
# Create a linear model
model <- lm(sales ~ advertisement_budget)
# Extract residuals from the model
residuals <- model$residuals
# Perform Shapiro-Wilk test for normality
shapiro_test <- shapiro.test(residuals)
# Print the result
print(shapiro_test)

# Sample data for Advertisement Budget and Sales
advertisement_budget <- c(10, 20, 30, 40, 50)
sales <- c(15, 25, 30, 35, 40)

# Create a linear model
model <- lm(sales ~ advertisement_budget)

# Predict values using the model
predictions <- predict(model)

# Compute residuals
residuals <- sales - predictions

# Compute Mean Squared Error (MSE)
mse <- mean(residuals^2)
print(paste("Mean Squared Error (MSE):", mse))



# Compute Root Mean Squared Error (RMSE)
rmse <- sqrt(mse)
print(paste("Root Mean Squared Error (RMSE):", rmse))

# R-squared (Coefficient of Determination)
rsq <- summary(model)$r.squared
print(paste("R-squared (R²):", rsq))

# Adjusted R-squared
adj_rsq <- summary(model)$adj.r.squared
print(paste("Adjusted R-squared:", adj_rsq))

# Plotting residuals
plot(predictions, residuals,
    main = "Residuals vs Fitted Values",
    xlab = "Fitted Values", ylab = "Residuals",
    pch = 16, col = "blue")
abline(h = 0, col = "red")  # Horizontal line at 0 (no residuals)
```