ADVERTISING DATA

Analysis

OVERVIEW

- We conducted an analysis to understand how different combinations of advertising expenditures influence sales.

- We used the 'Advertising.csv' dataset to explore how combined advertising efforts impacts sales. The analysis involved:

  1.Creating Interactions Terms

  2.Train-Test Split

  3.Variable Interaction and Evaluation

  4.Visualization

  5. Data Augmentation

1. CREATING INTERACTION TERMS:

Interaction Terms :

- Variables created by multiplying two or more predictors
- They help capture how the effect of one predictor on the outcome variable depends on the level of another predictor.

Purpose:

- To model more complex relationships beyond simple additive effects.
- To identify if and how predictors interact to influence the outcome.

The interaction terms we are using in our model are below:

TV * Radio

TV * Newspaper

Radio * Newspaper

TV*Radio*Newspaper

Added a constant term to the model.

- The **intercept** is a constant term in a linear regression equation. It represents the value of the dependent variable when all the predictor variables are set to zero.
- Mathematical Representation
- In a simple linear regression model, the equation is typically written as:
- $Y=\beta0+\beta1X+\epsilon$
- Y: Dependent variable (e.g., sales)
- X: Independent Variable (e.g., Advertising budget)
- β0: Intercept
- B1: Coefficient for the independent variable
- ϵ : Error Term

2. TRAIN-TEST SPLIT

Refers to the process of dividing a dataset into two distinct subsets:

1. Training Set: Used to train the model.
2. Test Set: Used to evaluate the model's performance on unseen data.

Why Use Train-Test Split?

- Evaluate Model Performance: By splitting the data, you can assess how well the model performs on data it hasn't seen before, providing a better estimate of its performance on new, unseen data.
- Prevent Overfitting: Training the model on the entire dataset can lead to overfitting, where the model learns the training data too well, including noise and specifics that don't generalize to new data. Using a separate test set helps to check if the model generalizes well.
- Test Size
- In our model, different quantities of the data were reserved for testing.

3. VARIABLE INTERACTION

We fit four different types of variable interactions with varying sets of predictors:

- 1: TV, Radio, Newspaper
- 2: TV, Radio, Newspaper, TV * Radio
- 3: TV, Radio, Newspaper, TV * Newspaper
- 4: TV, Radio, Newspaper, Radio * Newspaper
- 5: TV, Radio, Newspaper, TV* Radio * Newspaper
- **To capture the combined effect of two or more variables on the target variable, sales.**

```
# Read the data and create interaction terms
advertising_df = pd.read_csv("Advertising.csv", index_col=0)
advertising_df['TV*radio'] = advertising_df['TV'] * advertising_df['radio']
advertising_df['TV*newspaper'] = advertising_df['TV'] * advertising_df['newspaper']
advertising_df['radio*newspaper'] = advertising_df['radio'] * advertising_df['newspaper']
advertising_df['TV*radio*newspaper'] = advertising_df['TV'] * advertising_df['radio'] * advertising_df['newspaper']
advertising_df['intercept'] = 1
```

MEASURE OF THE MODEL'S PREDICTION ACCURACY

Mean Squared Error (MSE): The MSE is calculated for both training and testing sets. The lower the MSE, the better the variable interaction.

Interpreting Results:

- Performance on Training Data:

  If a combination has a lower training MSE, it means it fits the training data better. However, this does not necessarily indicate that it will perform well on new data.

- Performance on Testing Data:

  The key metric to assess performance is the test MSE. The model with the lowest test MSE is considered to perform better because it generalizes better to unseen data.

- to identify the best model and test size combination that minimizes the Mean Squared Error (MSE) on the test data

DEFINING A FUNCTION TO CALCULATE MSE

```
# Define a function to calculate mean squared error (MSE) for train and test datasets
def get_train_test_mse_predictions(y_var, xvars, train_df, test_df):
    model = sm.OLS(train_df[y_var], train_df[xvars])  # Create an Ordinary Least Squares (OLS) model
    model_fit = model.fit()  # Fit the model to the training data
    train_y_hat = model_fit.predict(train_df[xvars])  # Predict values for the training data
    test_y_hat = model_fit.predict(test_df[xvars])  # Predict values for the test data
    train_mse = ((train_df[y_var] - train_y_hat) ** 2).mean()  # Calculate mean squared error for the training data
    test_mse = ((test_df[y_var] - test_y_hat) ** 2).mean()  # Calculate mean squared error for the test data

    print(xvars)
    print("Train MSE:", train_mse)
    print("Test MSE:", test_mse)
    print()
    return test_mse  # Return the test MSE for comparison
```

VARIABLES

- Each set includes the intercept and various combinations of the original variables (TV, radio, newspaper) and their interaction terms.

to test multiple models with different combinations of predictors to find the best one.

```
# Define target variable and predictor variables
y_var = 'sales'
x_vars_1 = ['intercept', 'TV', 'radio', 'newspaper']
x_vars_2 = ['intercept', 'TV', 'radio', 'newspaper', 'TV*radio']
x_vars_3 = ['intercept', 'TV', 'radio', 'newspaper', 'TV*newspaper']
x_vars_4 = ['intercept', 'TV', 'radio', 'newspaper', 'radio*newspaper']
x_vars_5 = ['intercept', 'TV', 'radio', 'newspaper', 'TV*radio*newspaper']
```

TRYING DIFFERENT TEST SIZES

```
# Try different test sizes to find the best model
best_model = None
best_test_mse = float('inf')  # Initialize the best test MSE to infinity
best_test_size = None
best_mse_for_best_model = None

# Define test sizes to evaluate
test_sizes = [0.05, 0.1, 0.15, 0.2, 0.25, 0.3]
```

EVALUATING MODELS WITH DIFFERENT TEST SIZES

```
for test_size in test_sizes:
    # Perform train-test split
    train_df, test_df = train_test_split(advertising_df, test_size=test_size, random_state=42)

    # Calculate test MSE for each model
    test_mse_1 = get_train_test_mse_predictions(y_var, x_vars_1, train_df, test_df)
    test_mse_2 = get_train_test_mse_predictions(y_var, x_vars_2, train_df, test_df)
    test_mse_3 = get_train_test_mse_predictions(y_var, x_vars_3, train_df, test_df)
    test_mse_4 = get_train_test_mse_predictions(y_var, x_vars_4, train_df, test_df)
    test_mse_5 = get_train_test_mse_predictions(y_var, x_vars_5, train_df, test_df)

    # Determine the best model for the current test size
    test_mses = [test_mse_1, test_mse_2, test_mse_3, test_mse_4, test_mse_5]
    min_test_mse = min(test_mses)
    min_model_index = test_mses.index(min_test_mse) + 1  # Model index is 1-based

    # If the current best model's test MSE is less than the previously recorded best, update the best model
    if min_test_mse < best_test_mse:
        best_test_mse = min_test_mse
        best_test_size = test_size
        best_model = min_model_index
        best_mse_for_best_model = min_test_mse

print(f'Best model: Model {best_model} with test size: {best_test_size} and Test MSE: {best_mse_for_best_model}')
```

OUTCOME

```
['intercept', 'TV', 'radio', 'newspaper']
Train MSE: 2.7902497056243574
Test MSE: 2.738496820284218

['intercept', 'TV', 'radio', 'newspaper', 'TV*radio']
Train MSE: 0.898814505551989
Test MSE: 0.35960347585455044

['intercept', 'TV', 'radio', 'newspaper', 'TV*newspaper']
Train MSE: 2.6238775926580464
Test MSE: 2.5224557710487288

['intercept', 'TV', 'radio', 'newspaper', 'radio*newspaper']
Train MSE: 2.776179600691102
Test MSE: 2.911746050177556

['intercept', 'TV', 'radio', 'newspaper', 'TV*radio*newspaper']
Train MSE: 2.0704851073315838
Test MSE: 2.751482901623296
```

```
['intercept', 'TV', 'radio', 'newspaper']
Train MSE: 2.7543155639500685
Test MSE: 3.1308020912380448

['intercept', 'TV', 'radio', 'newspaper', 'TV*radio']
Train MSE: 0.8661112504818059
Test MSE: 0.951547109426838

['intercept', 'TV', 'radio', 'newspaper', 'TV*newspaper']
Train MSE: 2.529338094121459
Test MSE: 3.522115176127172

['intercept', 'TV', 'radio', 'newspaper', 'radio*newspaper']
Train MSE: 2.752676905079901
Test MSE: 3.090602150515758

['intercept', 'TV', 'radio', 'newspaper', 'TV*radio*newspaper']
Train MSE: 2.0255151589794402
Test MSE: 2.836487953487888
```

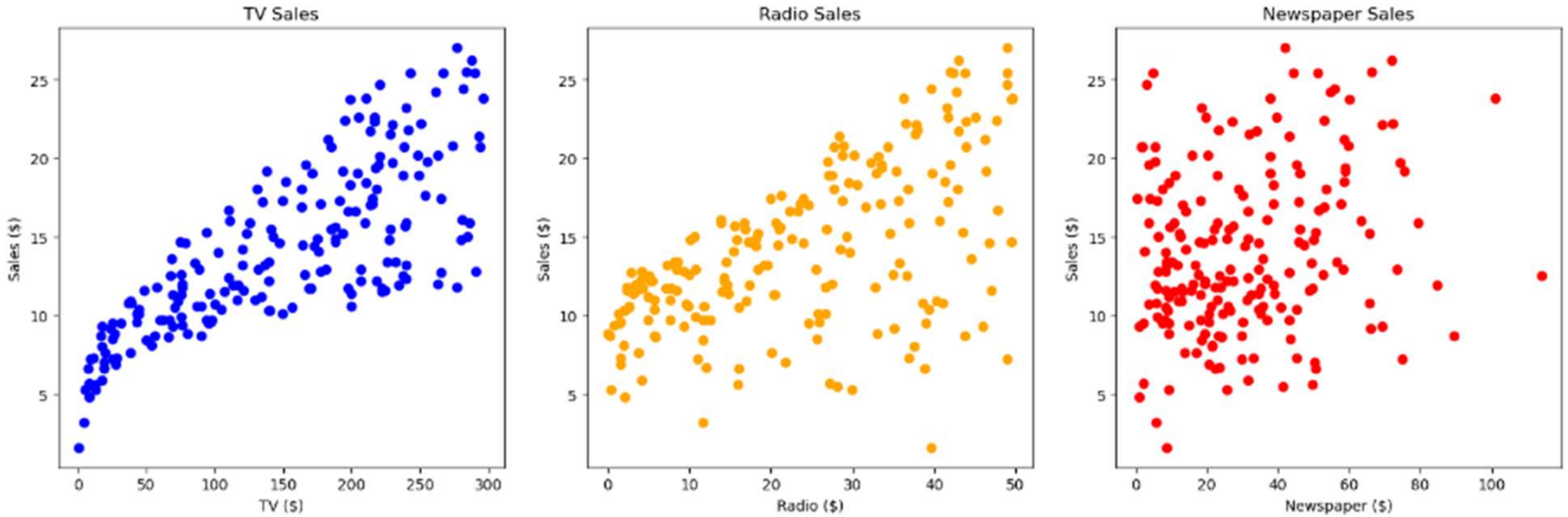
.....

```
Best model: Model 2 with test size: 0.05 and Test MSE: 0.35960347585455044
```

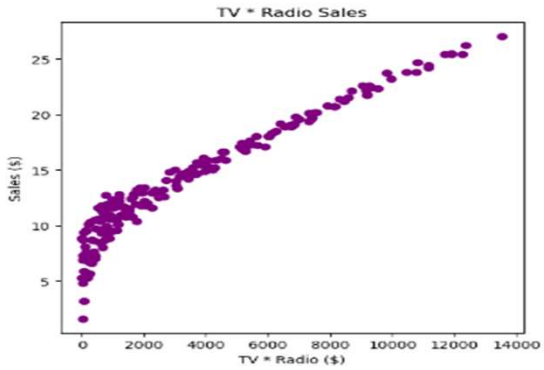
4. VISUALIZATION

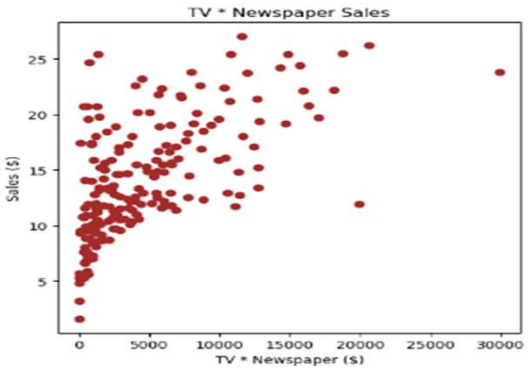
Scatter Plots of Predictor Variables Against Sales

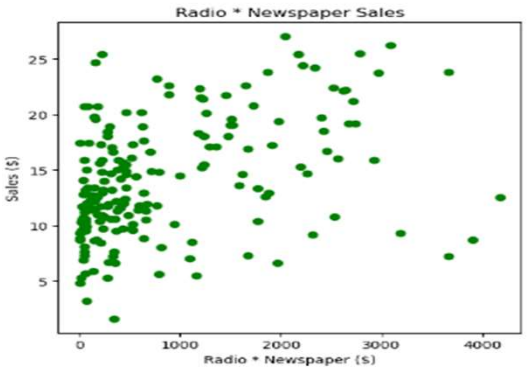
- Purpose: To visually inspect the relationship between each predictor variable and the sales outcome.

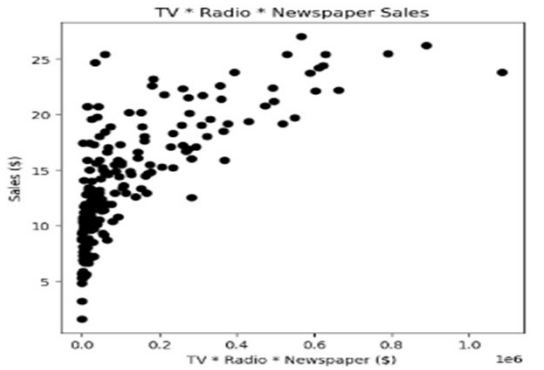


COMBINATION VARIABLES









```python
import matplotlib.pyplot as plt

# Create scatter plots
plt.figure(figsize=(15, 15))

# First row
plt.subplot(3, 3, 1)
plt.scatter(advertising_df['TV'], advertising_df['sales'], color='blue')
plt.title('TV Sales')
plt.xlabel('TV ($)')
plt.ylabel('Sales ($)')

plt.subplot(3, 3, 2)
plt.scatter(advertising_df['radio'], advertising_df['sales'], color='orange')
plt.title('Radio Sales')
plt.xlabel('Radio ($)')
plt.ylabel('Sales ($)')

plt.subplot(3, 3, 3)
plt.scatter(advertising_df['newspaper'], advertising_df['sales'], color='red')
plt.title('Newspaper Sales')
plt.xlabel('Newspaper ($)')
plt.ylabel('Sales ($)')

# Second row
plt.subplot(3, 3, 4)
plt.scatter(advertising_df['TV'] * advertising_df['radio'], advertising_df['sales'], color='purple')
plt.title('TV * Radio Sales')
plt.xlabel('TV * Radio ($)')
plt.ylabel('Sales ($)')

plt.subplot(3, 3, 5)
plt.scatter(advertising_df['TV'] * advertising_df['newspaper'], advertising_df['sales'], color='brown')
plt.title('TV * Newspaper Sales')
plt.xlabel('TV * Newspaper ($)')
plt.ylabel('Sales ($)')

plt.subplot(3, 3, 6)
plt.scatter(advertising_df['radio'] * advertising_df['newspaper'], advertising_df['sales'], color='green')
plt.title('Radio * Newspaper Sales')
plt.xlabel('Radio * Newspaper ($)')
plt.ylabel('Sales ($)')

# Third row: TV * Radio * Newspaper interaction
plt.subplot(3, 3, 7)
plt.scatter(advertising_df['TV'] * advertising_df['radio'] * advertising_df['newspaper'], advertising_df['sales'], color='black')
plt.title('TV * Radio * Newspaper Sales')
plt.xlabel('TV * Radio * Newspaper ($)')
plt.ylabel('Sales ($)')

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

5. DATA AUGMENTATION

Data augmentation is a technique used to artificially expand a dataset by generating new data points based on the existing data.

why data augmentation ?

- Reducing overfitting and underfitting
- Making the model more generalized and robust

How data augmentation works on a csv data ?

- Adding Gaussian noise
- Interpolation

- **Adding gaussian noise:**

  Adding Gaussian noise means making small, random changes to your data. These changes are like tiny, random adjustments that mostly stay close to the original values, similar to adding a bit of "fuzziness" or "randomness" to make the data less perfect or more like what you might see in the real world.

```
# 1. Adding Gaussian noise
def add_noise(data, noise_level=0.01):
    noise = np.random.normal(0, noise_level, data.shape)
    return data + noise

data_noisy = data.copy()
data_noisy[['TV', 'radio', 'newspaper', 'sales']] = add_noise(data[['TV', 'radio', 'newspaper', 'sales']], noise_level=0.05)
```

- **Interpolation :**

  Interpolation is a method used to estimate or create new data points within a range of known data points

  For example :

  Interpolation uses known temperature data to estimate unknown values. For example, if the temperature is 60°F at 8 AM and 75°F at 12 PM, you can estimate that the temperature at 10 AM is around 67.5°F, filling in the gap between the two measurements.

```python
# 2. Synthetic sample generation through interpolation
def interpolate_samples(data):
    new_data_list = []
    for i in range(len(data) - 1):
        new_sample = (data.iloc[i] + data.iloc[i + 1]) / 2
        new_data_list.append(new_sample)
    new_data = pd.DataFrame(new_data_list)
    return new_data

data_interpolated = interpolate_samples(data)
```

DATA AUGMENTATION RESULTS

```
['intercept', 'TV', 'radio', 'newspaper']
Train MSE: 2.310234366913194
Test MSE: 2.2253670228533866

['intercept', 'TV', 'radio', 'newspaper', 'TV*radio']
Train MSE: 0.9423183656945587
Test MSE: 0.6246881235196042

['intercept', 'TV', 'radio', 'newspaper', 'TV*newspaper']
Train MSE: 2.168237033595301
Test MSE: 2.208304012914105

['intercept', 'TV', 'radio', 'newspaper', 'radio*newspaper']
Train MSE: 2.308446377111834
Test MSE: 2.204121098178071
```

THANK YOU !