💰 A Comparative Study of Ridge and Lasso Regularization Based on MSE Performance on Advertising Data

This example explores how **L1 (Lasso)** and **L2 (Ridge)** regularization techniques help improve prediction accuracy and avoid overfitting when building a regression model to predict **Sales Revenue** from various **advertising spending channels**.

🔍 Scenario: Predicting Sales from Ad Spend

You have the following features:

- **TV Advertising Spend**
- **Radio Advertising Spend**
- **Newspaper Advertising Spend**
- **Social Media Advertising Spend**
- **Influencer Marketing Spend**
- **Search Engine Marketing Spend**

Your linear regression model aims to predict:

```
Sales Revenue = w1*TV + w2*Radio + w3*Newspaper + w4*Social Media + w5*Influencer + w6*Search Engine + b
```

⚠️ Without Regularization (Basic Linear Regression)

A plain linear regression model might **overfit** the data if:

- Some channels don't strongly influence sales
- There's overlap or multicollinearity between ad channels
- Data contains noise or outliers

Overfitting means the model performs well on training data but poorly on new/unseen data.

🧮 L2 Regularization (Ridge)

Suppose Ridge regression estimates:

- `w1 = 3.2` (TV)
- `w2 = 2.0` (Radio)
- `w3 = 1.1` (Newspaper)
- `w4 = 1.8` (Social Media)
- `w5 = 0.9` (Influencer)
- `w6 = 2.5` (Search Engine)

Ridge **shrinks** all coefficients but **retains all features**, assuming each has some relevance.

✅ **Best when all ad channels contribute to sales performance**.

✂️ L1 Regularization (Lasso)

Now suppose Lasso regression results in:

- `w1 = 3.5` (TV)
- `w2 = 1.9` (Radio)
- `w3 = 0` (Newspaper)
- `w4 = 2.0` (Social Media)
- `w5 = 0` (Influencer)
- `w6 = 2.7` (Search Engine)

Lasso sets some coefficients to **zero**, removing channels that don't significantly contribute to sales.

✅ **Great when you want simpler models and automatic feature selection**.

🧠 Intuition Recap

Regularization	What it does	When to use
Ridge (L2)	Shrinks all coefficients (keeps all features)	When all ad channels matter to some extent
Lasso (L1)	Eliminates unimportant features (zero weights)	When simplifying the model or selecting top ads

With out Regulation

```
import pandas as pd
```

```
# Load the dataset
data = pd.read_excel("/content/Sales.xlsx")
```

```
data
```

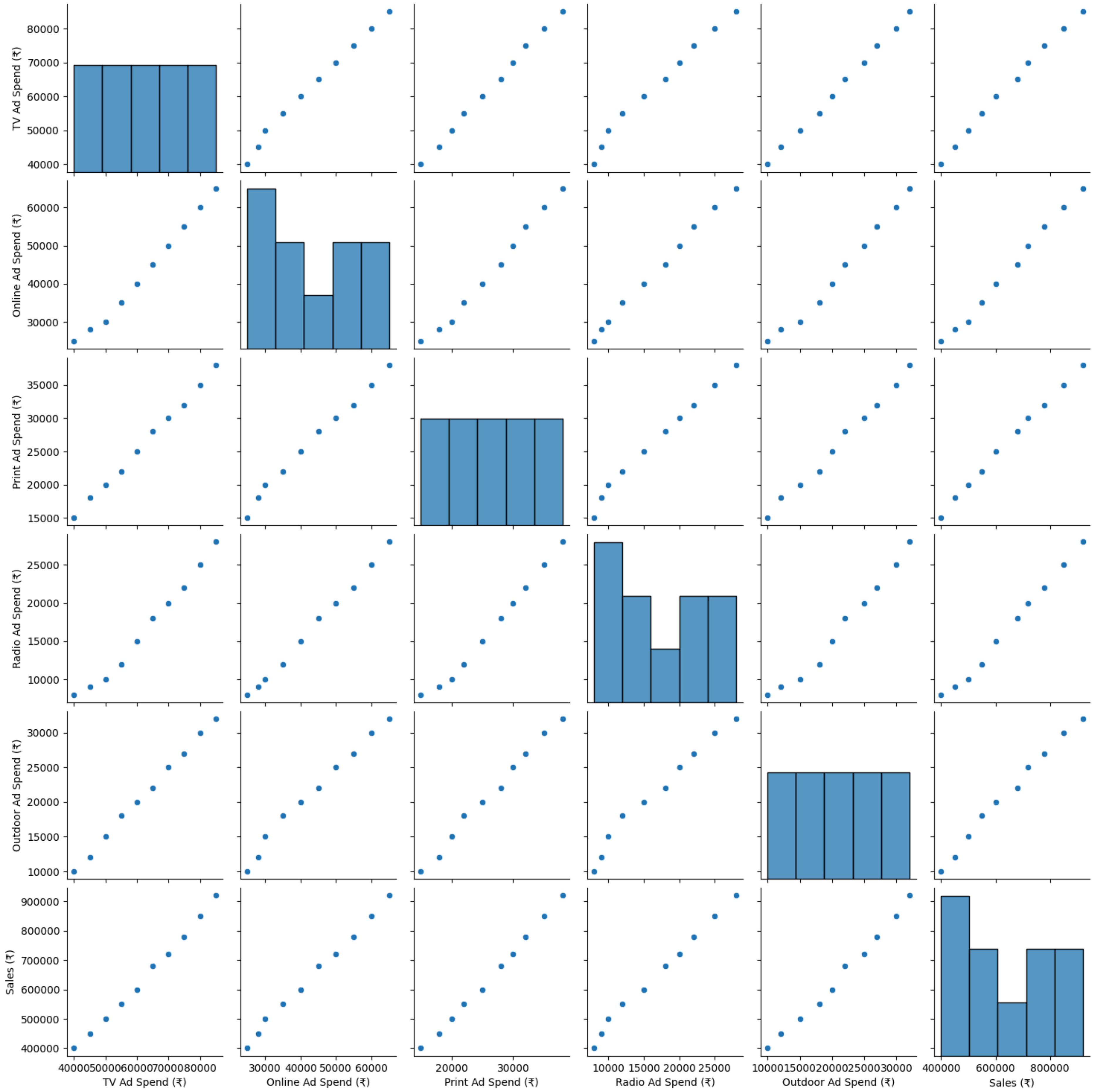
	TV Ad Spend (₹)	Online Ad Spend (₹)	Print Ad Spend (₹)	Radio Ad Spend (₹)	Outdoor Ad Spend (₹)	Sales (₹)
0	50000	30000	20000	10000	15000	500000
1	60000	40000	25000	15000	20000	600000
2	70000	50000	30000	20000	25000	720000
3	40000	25000	15000	8000	10000	400000
4	80000	60000	35000	25000	30000	850000
5	55000	35000	22000	12000	18000	550000
6	65000	45000	28000	18000	22000	680000
7	75000	55000	32000	22000	27000	780000
8	45000	28000	18000	9000	12000	450000
9	85000	65000	38000	28000	32000	920000

```
data.columns
```

```
Index(['TV Ad Spend (₹)', 'Online Ad Spend (₹)', 'Print Ad Spend (₹)',
       'Radio Ad Spend (₹)', 'Outdoor Ad Spend (₹)', 'Sales (₹)'],
      dtype='object')
```

```
import seaborn as sn
sn.pairplot(data)
```

<seaborn.axisgrid.PairGrid at 0x7aa44b3047d0>




```
# Features and target
X = data[['TV Ad Spend (₹)', 'Online Ad Spend (₹)', 'Print Ad Spend (₹)', 'Radio Ad Spend (₹)', 'Outdoor Ad Spend (₹)']]
y = data["Sales (₹)"]
```

```
# Split the dataset
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
# Linear Regression (no regularization)
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
print("Linear Regression Coefficients:", lr.coef_)
```

```
Linear Regression Coefficients: [ 4.82058824 -0.62058824 11.63235294  8.36764706 -5.        ]
```

```
from sklearn.metrics import mean_squared_error
y_pred = lr.predict(X_test)
mse_No_regulation = mean_squared_error(y_test, y_pred)
mse_No_regulation
```

```
85104913.49480934
```

L1 Regulation

```
# Linear Regression (L1 regularization)
from sklearn.linear_model import Lasso
lasso = Lasso(alpha=10000)

lasso.fit(X_train, y_train)
```

/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_coordinate_descent.py:695: ConvergenceWarning: Objective did not converge. You might want to increase the number of itera
  model = cd_fast.enet_coordinate_descent(

▾ Lasso

```
print("Lasso Coefficients:", lasso.coef_)
```

Lasso Coefficients: [9.42227539   1.40354614 11.0651829   5.02269524 -14.46198805]

```
from sklearn.metrics import mean_squared_error
y_pred = lasso.predict(X_test)
mse_L1_regulation = mean_squared_error(y_test, y_pred)
mse_L1_regulation
```

129776832.71697903

L2 Regulation Ridge

```
# Linear Regression (L2 regularization)
from sklearn.linear_model import Ridge
ridge = Ridge(alpha=1.0)
ridge.fit(X_train, y_train)
```

▾ Ridge

Ridge()

```
print("Lasso Coefficients:", ridge.coef_)
```

Lasso Coefficients: [4.82058229 -0.62058668 11.6323446   8.36764528 -4.99998065]

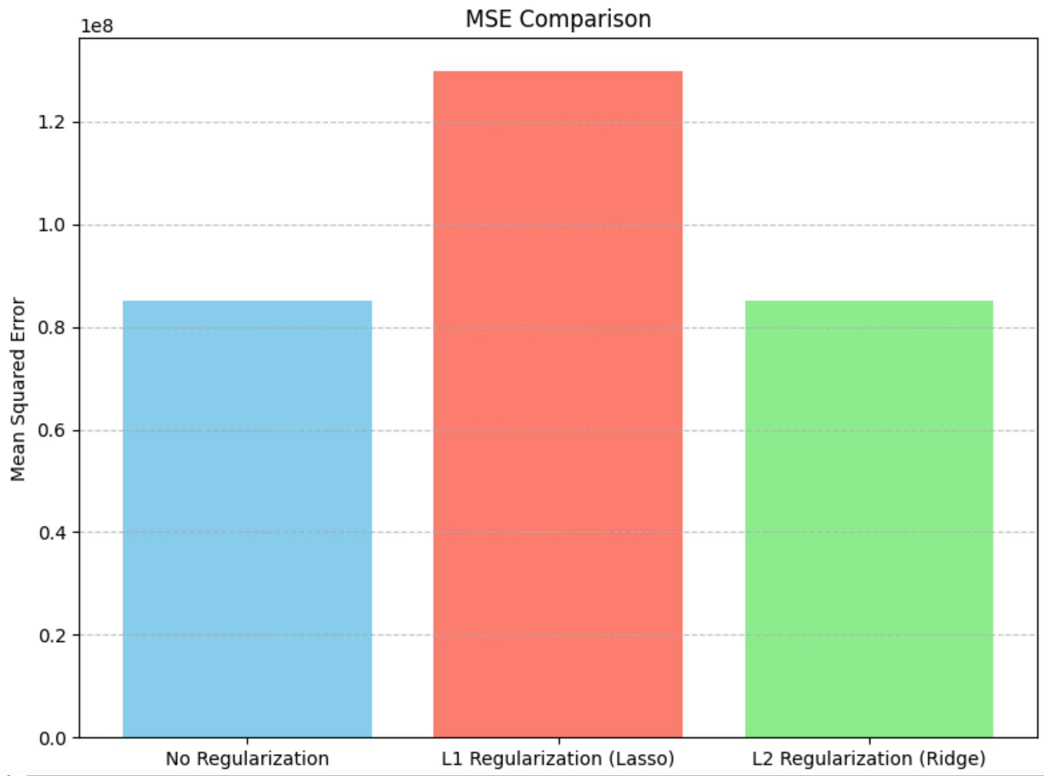
```
from sklearn.metrics import mean_squared_error
y_pred = ridge.predict(X_test)
mse_L2_regulation = mean_squared_error(y_test, y_pred)
mse_L2_regulation
```

85104793.54359482

```
import matplotlib.pyplot as plt

mse_values = {
    "No Regularization": mse_No_regulation,
    "L1 Regularization (Lasso)": mse_L1_regulation,
    "L2 Regularization (Ridge)": mse_L2_regulation
}

plt.figure(figsize=(8, 6))
plt.bar(mse_values.keys(), mse_values.values(), color=["skyblue", "salmon", "lightgreen"])
plt.title("MSE Comparison")
plt.ylabel("Mean Squared Error")
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.