Lab 5: Overfitting and Regularization in Linear Regression

Objective:

- Understand **overfitting** in machine learning models.
- Implement **L1 (Lasso) and L2 (Ridge) regularization** to **reduce overfitting**.
- Compare results to see how regularization **improves model performance**.

Prerequisites:

- Basic Python programming knowledge.
- Familiarity with **Pandas, NumPy, Matplotlib, Seaborn, and Scikit-learn**.
- Understanding of **linear regression and overfitting** concepts.

1. Import Required Libraries

```
# Import libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

2. Load and Preprocess the Dataset

```
# Read dataset
dataset = pd.read_csv('Melbourne_housing_FULL.csv')

# Display first 5 rows
dataset.head()

# Check unique values in each column
dataset.nunique()
```

3. Select Relevant Features

```
# let's use limited columns which makes more sense for serving our
purpose
cols_to_use = ['Suburb', 'Rooms', 'Type', 'Method', 'SellerG',
'Regionname', 'Propertycount',
               'Distance', 'CouncilArea', 'Bedroom2', 'Bathroom',
'Car', 'Landsize', 'BuildingArea', 'Price']
dataset = dataset[cols_to_use]

# Display first few rows
dataset.head()

# Check dataset shape
dataset.shape
```

```
# Check missing values
dataset.isna().sum()
```

4. Handle Missing Values

```
# Some feature's missing values can be treated as zero (another class
for NA values or absence of that feature)
# like 0 for Propertycount, Bedroom2 will refer to other class of NA
values
# like 0 for Car feature will mean that there's no car parking feature
with house
cols_to_fill_zero = ['Propertycount', 'Distance', 'Bedroom2',
'Bathroom', 'Car']
dataset[cols_to_fill_zero] = dataset[cols_to_fill_zero].fillna(0)

# other continuous features can be imputed with mean for faster results
since our focus is on Reducing overfitting
# using Lasso and Ridge Regression
dataset['Landsize'] =
dataset['Landsize'].fillna(dataset.Landsize.mean())
dataset['BuildingArea'] =
dataset['BuildingArea'].fillna(dataset.BuildingArea.mean())


dataset.dropna(inplace=True)


dataset.shape
```

5. Encode Categorical Features

```
dataset = pd.get_dummies(dataset, drop_first=True)

# Display dataset after encoding
dataset.head()
```

Let's bifurcate our dataset into train and test dataset

```
X = dataset.drop('Price', axis=1)
y = dataset['Price']
```

6. Split Data into Training and Testing Sets

```
from sklearn.model_selection import train_test_split
train_X, test_X, train_y, test_y = train_test_split(X, y,
test_size=0.3, random_state=2)
```

7. Train a Standard Linear Regression Model

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression().fit(train_X, train_y)


reg.score(test_X, test_y)

reg.score(train_X, train_y)
```

8. Apply Ridge (L2) Regularization

```
from sklearn.linear_model import Ridge
ridge_reg= Ridge(alpha=50, max_iter=100, tol=0.1)
ridge_reg.fit(train_X, train_y)

ridge_reg.score(test_X, test_y)

ridge_reg.score(train_X, train_y)
```

9. Apply Lasso (L1) Regularization

```
from sklearn import linear_model
lasso_reg = linear_model.Lasso(alpha=50, max_iter=100, tol=0.1)
lasso_reg.fit(train_X, train_y)


lasso_reg.score(test_X, test_y)

lasso_reg.score(train_X, train_y)
```

10.Visualizations

```
# Import necessary libraries
from sklearn.linear_model import Ridge, Lasso

# Train Ridge Regression (L2)
ridge_reg = Ridge(alpha=50, max_iter=100, tol=0.1)
ridge_reg.fit(train_X, train_y)

# Train Lasso Regression (L1)
lasso_reg = Lasso(alpha=50, max_iter=100, tol=0.1)
lasso_reg.fit(train_X, train_y)

# Store R² scores
lin_train_score = reg.score(train_X, train_y)
lin_test_score = reg.score(test_X, test_y)

ridge_train_score = ridge_reg.score(train_X, train_y)
ridge_test_score = ridge_reg.score(test_X, test_y)

lasso_train_score = lasso_reg.score(train_X, train_y)
lasso_test_score = lasso_reg.score(test_X, test_y)

# Create a dataframe for visualization
import pandas as pd
score_df = pd.DataFrame({
    "Model": ["Linear Regression", "Ridge", "Lasso"],
    "Train Score": [lin_train_score, ridge_train_score,
lasso_train_score],
    "Test Score": [lin_test_score, ridge_test_score, lasso_test_score]
})
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Print the R² scores for each model
print(f"Linear Regression - Train Score: {lin_train_score:.4f}")
print(f"Linear Regression - Test Score: {lin_test_score:.4f}\n")

print(f"Ridge Regression - Train Score: {ridge_train_score:.4f}")
print(f"Ridge Regression - Test Score: {ridge_test_score:.4f}\n")

print(f"Lasso Regression - Train Score: {lasso_train_score:.4f}")
print(f"Lasso Regression - Test Score: {lasso_test_score:.4f}")


import matplotlib.pyplot as plt
import seaborn as sns

# Set the style for plots
sns.set_style("whitegrid")

# Plot Train vs Test scores for different models
plt.figure(figsize=(10, 5))

# Bar plot for Train and Test scores
X_axis = ["Linear Regression", "Ridge", "Lasso"]
train_scores = [lin_train_score, ridge_train_score, lasso_train_score]
test_scores = [lin_test_score, ridge_test_score, lasso_test_score]

bar_width = 0.3  # Bar width for better visibility
index = range(len(X_axis))

plt.bar(index, train_scores, width=bar_width, label="Train Score",
color='royalblue', alpha=0.7)
plt.bar([i + bar_width for i in index], test_scores, width=bar_width,
label="Test Score", color='orange', alpha=0.7)

# Labels and title
plt.xlabel("Models", fontsize=12)
plt.ylabel("R² Score", fontsize=12)
plt.title("Train vs Test Scores of Different Regression Models",
fontsize=14)
plt.xticks([i + bar_width / 2 for i in index], X_axis)
plt.legend()
plt.show()
```

Tasks

1. **Data Preprocessing:**
    - How did you handle missing values in the dataset?
    - Why did we use get_dummies() for categorical variables?
2. **Model Training & Performance:**
    - What are the R^2 scores for the **Linear Regression model** on training and testing data?
    - What does the difference between the train and test scores indicate?
3. **Ridge (L2) Regularization:**
    - What are the train and test scores for Ridge Regression?
    - How does Ridge Regression help in reducing overfitting?
4. **Lasso (L1) Regularization:**
    - What are the train and test scores for Lasso Regression?
    - How does Lasso affect feature selection compared to Ridge?
5. **Comparison & Visualization:**
    - Compare the performances of Linear, Ridge, and Lasso Regression models.
    - Based on the visualizations, which model performed best and why?
6. **Regularization Impact:**
    - What happens when you increase the **alpha value** in Ridge and Lasso Regression?
    - If you had to choose one model for this dataset, which one would it be and why?