```
#Assignment 4
#Amit pawar
#prn:25070126501
#Assignment 4: Ridge and Lasso Regression
```

Assignment Summary: Ridge and Lasso Regression

This assignment explores **Ridge and Lasso Regression**, which are regularization techniques used in linear regression to prevent overfitting and improve model performance, especially with multicollinear data or many features.

What to Use (Libraries and Tools):

To complete this assignment, you primarily need the following Python libraries:

- **`pandas`**: For data manipulation and analysis, especially for handling DataFrames (e.g., `pd.read_csv`, `pd.DataFrame`).
- **`numpy`**: For numerical operations, especially when working with arrays and generating ranges (e.g., `np.logspace`).
- **`matplotlib.pyplot`**: For creating static, interactive, and animated visualizations (e.g., `plt.figure`, `plt.plot`, `plt.show`).
- **`sklearn` (scikit-learn)**: The most crucial library for machine learning tasks, including:
    - `sklearn.model_selection.train_test_split`: To divide data into training and testing sets.
    - `sklearn.model_selection.GridSearchCV`: For hyperparameter tuning (finding the best `alpha` value for regularization).
    - `sklearn.linear_model.Ridge`: To implement Ridge Regression.
    - `sklearn.linear_model.Lasso`: To implement Lasso Regression.
    - `sklearn.metrics.mean_squared_error`: To evaluate model performance using Mean Squared Error.
    - `sklearn.metrics.r2_score`: To evaluate model performance using the R-squared score.

How to Run This Assignment:

The assignment typically follows these steps, executed sequentially in your notebook:

1. **Import Libraries**: Start by importing all necessary libraries (`pandas`, `numpy`, `matplotlib.pyplot`, and specific modules from `sklearn`).
2. **Load Data**: Load the provided dataset (`Advertising dataset.csv`) into a pandas DataFrame.
3. **Data Preparation**:
    - Separate your features (`x` - 'TV', 'Radio', 'Newspaper') from your target variable (`y` - 'Sales').
    - Split the data into training and testing sets using `train_test_split` (e.g., 80% for training, 20% for testing).
4. **Implement Ridge Regression**:
    - Initialize and train `Ridge` models with various `alpha` values to observe how coefficients change.
    - Visualize the coefficient shrinkage as `alpha` increases.
    - Use `GridSearchCV` with `Ridge` to find the optimal `alpha` that minimizes error (or maximizes a scoring metric like `neg_mean_squared_error`).
5. **Implement Lasso Regression**:
    - Similarly, initialize and train `Lasso` models with various `alpha` values, paying attention to coefficients potentially becoming zero.

   - Visualize the coefficient shrinkage for Lasso.
   - Use `GridSearchCV` with `Lasso` to find its optimal `alpha`.
6. **Evaluate Models**:
   - Make predictions on the test set using the best Ridge and Lasso models (found via `GridSearchCV`).
   - Calculate and print evaluation metrics such as Mean Squared Error (MSE) and R-squared (R2) score for both models.
7. **Visualize Results**:
   - Create scatter plots to compare actual sales values against predicted sales from both Ridge and Lasso models. An ideal fit would show points along a diagonal line.
   - You might also visualize the final coefficients of the best models.

By following these steps, you will perform a comprehensive analysis of Ridge and Lasso Regression, understand their impact on model coefficients, and compare their performance.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LinearRegression, Ridge, Lasso
```

```
df = pd.read_csv("/content/Advertising dataset.csv")
df.head(10)
```

	Unnamed: 0	TV	Radio	Newspaper	Sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9
5	6	8.7	48.9	75.0	7.2
6	7	57.5	32.8	23.5	11.8
7	8	120.2	19.6	11.6	13.2
8	9	8.6	2.1	1.0	4.8
9	10	199.8	2.6	21.2	10.6

Next steps: Generate code with df   New interactive sheet

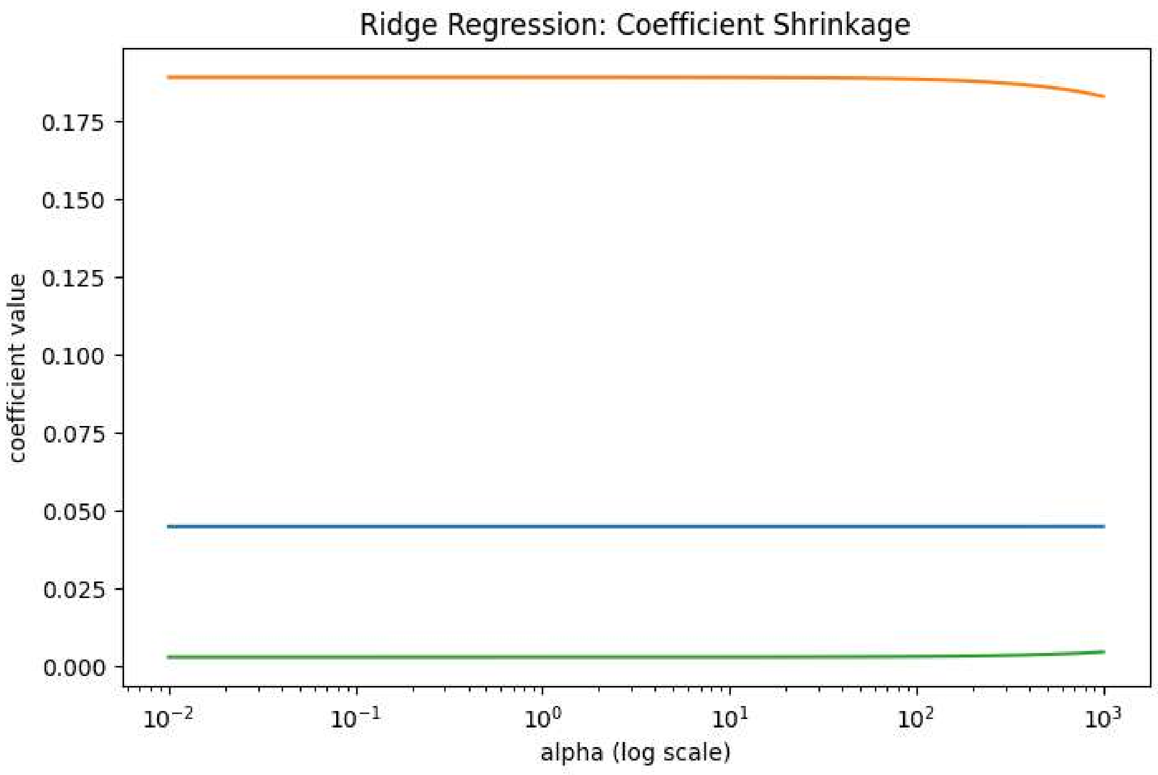
```
x = df[['TV', 'Radio','Newspaper']]
y = df['Sales']
```

```
X_train, X_test, y_train, y_test = train_test_split(
    x, y,
    test_size=0.2,
    random_state=42
)
```

```
alpha_values = [0, 0.1, 1, 10, 100]

print("RIDGE REGRESSION COEFFICIENTS\n")

for alpha_value in alpha_values:
    ridge_model = Ridge(alpha=alpha_value)
    ridge_model.fit(X_train, y_train)

    print(f"Alpha = {alpha_value}")
    print(pd.Series(ridge_model.coef_, index=x.columns))
    print("-" * 40)
```

```
RIDGE REGRESSION COEFFICIENTS

Alpha = 0
TV           0.044730
Radio        0.189195
Newspaper    0.002761
dtype: float64
----------------------------------------
Alpha = 0.1
TV           0.044730
Radio        0.189194
Newspaper    0.002761
dtype: float64
----------------------------------------
Alpha = 1
TV           0.044730
Radio        0.189189
Newspaper    0.002763
dtype: float64
----------------------------------------
Alpha = 10
TV           0.044730
Radio        0.189131
Newspaper    0.002779
dtype: float64
----------------------------------------
Alpha = 100
TV           0.044731
Radio        0.188561
Newspaper    0.002936
dtype: float64
----------------------------------------
```

```
alphas = np.logspace(-2, 3, 50)
coefficients_path = []

for alpha_value in alphas:
    ridge_model = Ridge(alpha=alpha_value)
    ridge_model.fit(X_train, y_train)
    coefficients_path.append(ridge_model.coef_)

coefficients_path = np.array(coefficients_path)

plt.figure(figsize=(8, 5))
plt.plot(alphas, coefficients_path)
plt.xscale("log")
plt.xlabel("alpha (log scale)")
plt.ylabel("coefficient value")
plt.title("Ridge Regression: Coefficient Shrinkage")
plt.show()
```



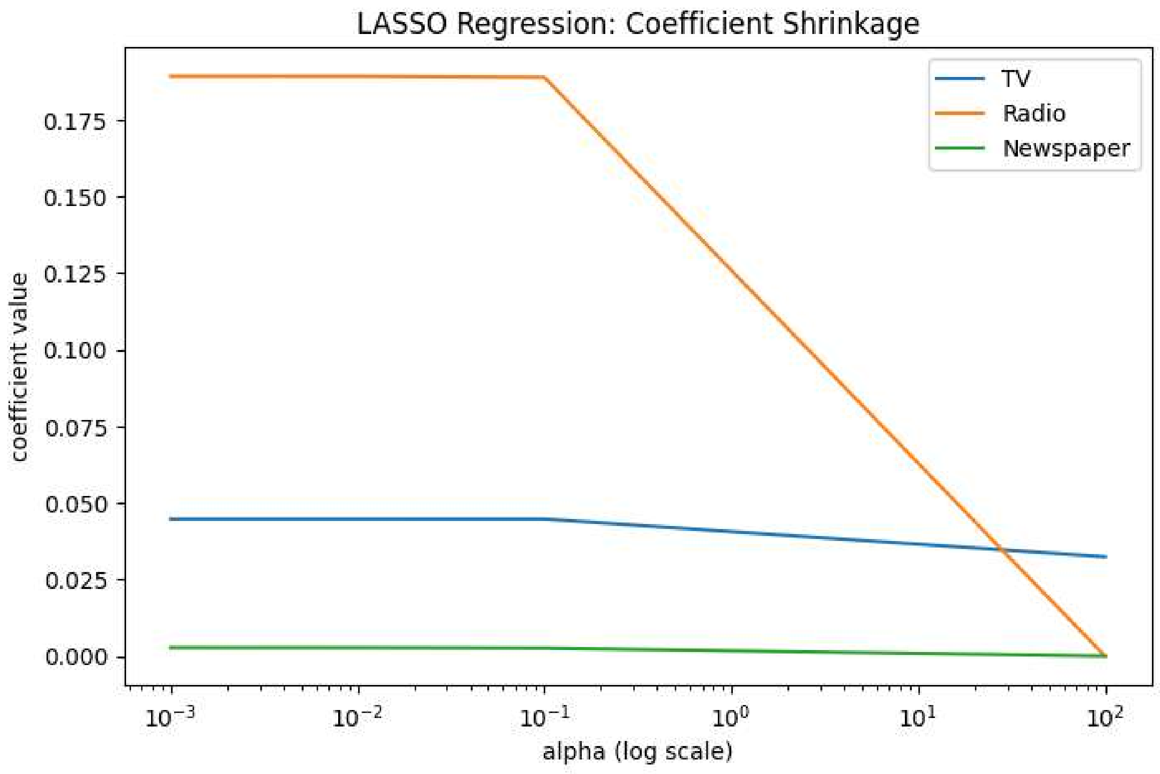
```
alpha_values = [0.001, 0.01, 0.1, 1]

print("RIDGE REGRESSION COEFFICIENTS\n")

for alpha_value in alpha_values:
    ridge_model = Ridge(alpha=alpha_value)
    ridge_model.fit(X_train, y_train)

    print(f"Alpha = {alpha_value}")
    print(pd.Series(ridge_model.coef_, index=x.columns))
    print("-" * 40)


print("\nLASSO REGRESSION COEFFICIENTS\n")

for alpha_value in alpha_values:
    lasso_model = Lasso(alpha=alpha_value, max_iter=5000)
    lasso_model.fit(X_train, y_train)

    print(f"Alpha = {alpha_value}")
    print(pd.Series(lasso_model.coef_, index=x.columns))
    print("-" * 40)
```

```
RIDGE REGRESSION COEFFICIENTS

Alpha = 0.001
TV           0.044730
Radio        0.189195
Newspaper    0.002761
dtype: float64
----------------------------------------
Alpha = 0.01
TV           0.044730
Radio        0.189195
Newspaper    0.002761
dtype: float64
----------------------------------------
```

```
Alpha = 0.1
TV           0.044730
Radio        0.189194
Newspaper    0.002761
dtype: float64
----------------------------------------
Alpha = 1
TV           0.044730
Radio        0.189189
Newspaper    0.002763
dtype: float64
----------------------------------------

LASSO REGRESSION COEFFICIENTS

Alpha = 0.001
TV           0.044729
Radio        0.189191
Newspaper    0.002760
dtype: float64
----------------------------------------
Alpha = 0.01
TV           0.044728
Radio        0.189157
Newspaper    0.002748
dtype: float64
----------------------------------------
Alpha = 0.1
TV           0.044720
Radio        0.188811
Newspaper    0.002627
dtype: float64
----------------------------------------
Alpha = 1
TV           0.044631
Radio        0.185350
Newspaper    0.001422
dtype: float64
----------------------------------------
```

```python
print("\nLASSO REGRESSION COEFFICIENTS\n")

alpha_values_lasso = [0.001, 0.01, 0.1, 100,]
coefficients_path_lasso = []

for alpha_value in alpha_values_lasso:
    lasso_model = Lasso(alpha=alpha_value, max_iter=5000)
    lasso_model.fit(X_train, y_train)

    print(f"Alpha = {alpha_value}")
    print(pd.Series(lasso_model.coef_, index=x.columns))
    print("-" * 40)
    coefficients_path_lasso.append(lasso_model.coef_)

coefficients_path_lasso = np.array(coefficients_path_lasso)

plt.figure(figsize=(8, 5))
plt.plot(alpha_values_lasso, coefficients_path_lasso)
plt.xscale("log")
plt.xlabel("alpha (log scale)")
plt.ylabel("coefficient value")
plt.title("LASSO Regression: Coefficient Shrinkage")
plt.legend(x.columns)
plt.show()
```

```
LASSO REGRESSION COEFFICIENTS

Alpha = 0.001
TV           0.044729
Radio        0.189191
Newspaper    0.002760
dtype: float64
----------------------------------------
Alpha = 0.01
TV           0.044728
Radio        0.189157
Newspaper    0.002748
dtype: float64
----------------------------------------
Alpha = 0.1
TV           0.044720
Radio        0.188811
Newspaper    0.002627
dtype: float64
----------------------------------------
Alpha = 100
TV           0.032409
Radio        0.000000
Newspaper    0.000000
dtype: float64
----------------------------------------
```



```
import numpy as np
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV

param_grid = {'alpha': np.logspace(-3, 2, 20)}

ridge_cv = GridSearchCV(
    Ridge(),
    param_grid,
    scoring='neg_mean_squared_error',
    cv=5
)

ridge_cv.fit(X_train, y_train)

print("Best Alpha (Ridge):", ridge_cv.best_params_['alpha'])
```

```
Best Alpha (Ridge): 0.001
```

```
from sklearn.linear_model import Lasso

param_grid_lasso = {'alpha': np.logspace(-3, 2, 20)}

lasso_cv = GridSearchCV(
    Lasso(max_iter=5000),
    param_grid_lasso,
    scoring='neg_mean_squared_error',
    cv=5
)

lasso_cv.fit(X_train, y_train)

print(f"Best Alpha (Lasso): {lasso_cv.best_params_['alpha']}")
```

```
Best Alpha (Lasso): 1.438449888287663
```

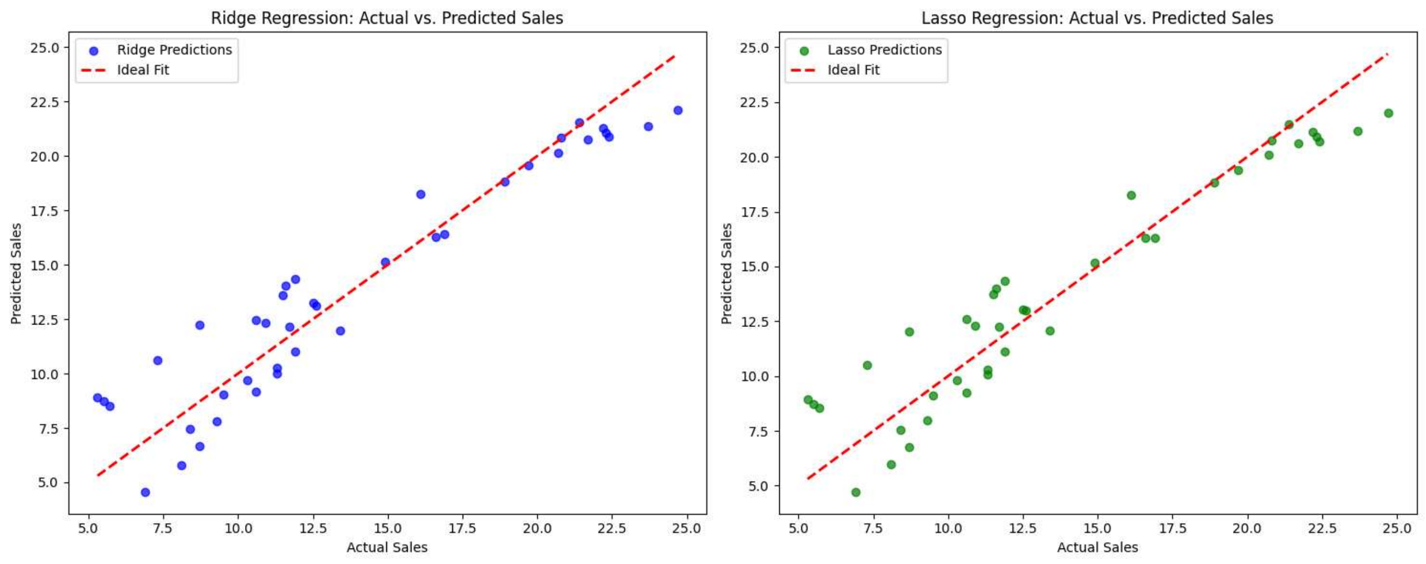
```
import numpy as np
from sklearn.linear_model import Ridge, Lasso
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error

param_grid_ridge = {'alpha': np.logspace(-3, 2, 20)}
ridge_cv = GridSearchCV(
    Ridge(),
    param_grid_ridge,
    scoring='neg_mean_squared_error',
    cv=5
)
ridge_cv.fit(X_train, y_train)

param_grid_lasso = {'alpha': np.logspace(-3, 2, 20)}
lasso_cv = GridSearchCV(
    Lasso(max_iter=5000),
    param_grid_lasso,
    scoring='neg_mean_squared_error',
    cv=5
)
lasso_cv.fit(X_train, y_train)

best_ridge = ridge_cv.best_estimator_
best_lasso = lasso_cv.best_estimator_

ridge_pred = best_ridge.predict(X_test)
lasso_pred = best_lasso.predict(X_test)

print("Ridge MSE:", mean_squared_error(y_test, ridge_pred))
print("Lasso MSE:", mean_squared_error(y_test, lasso_pred))
```

```
Ridge MSE: 3.1740973514255293
Lasso MSE: 3.1341162711720303
```

```
print("Final Ridge Coefficients:")
print(pd.Series(best_ridge.coef_, index=x.columns))

print("\nFinal Lasso Coefficients:")
print(pd.Series(best_lasso.coef_, index=x.columns))
```

```
Final Ridge Coefficients:
TV           0.044730
Radio        0.189195
Newspaper    0.002761
dtype: float64

Final Lasso Coefficients:
TV           0.044587
Radio        0.183663
Newspaper    0.000835
dtype: float64
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(15, 6))

plt.subplot(1, 2, 1)
plt.scatter(y_test, ridge_pred, color='blue', label="Ridge Predictions", alpha=0.7)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2, label='Ideal Fit')
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Ridge Regression: Actual vs. Predicted Sales')
plt.legend()

plt.subplot(1, 2, 2)
plt.scatter(y_test, lasso_pred, color='green', label="Lasso Predictions", alpha=0.7)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--', lw=2, label='Ideal Fit')
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Lasso Regression: Actual vs. Predicted Sales')
plt.legend()

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



```
from sklearn.metrics import r2_score
```

```python
ridge_r2 = r2_score(y_test, ridge_pred)
print(f"Ridge R2 Score: {ridge_r2}")

lasso_r2 = r2_score(y_test, lasso_pred)
print(f"Lasso R2 Score: {lasso_r2}")
```

```
Ridge R2 Score: 0.8994380241817195
Lasso R2 Score: 0.9007047075819106
```