Lasso_Ridge_Class_Assignment

February 16, 2025

1 Ridge And Lasso Regression

[1]:
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, r2_score, mean_squared_error
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score
from sklearn.model_selection import GridSearchCV
```

[2]:
```python
df = pd.read_csv('Advertising.csv')
df.head()
```

[2]:

	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

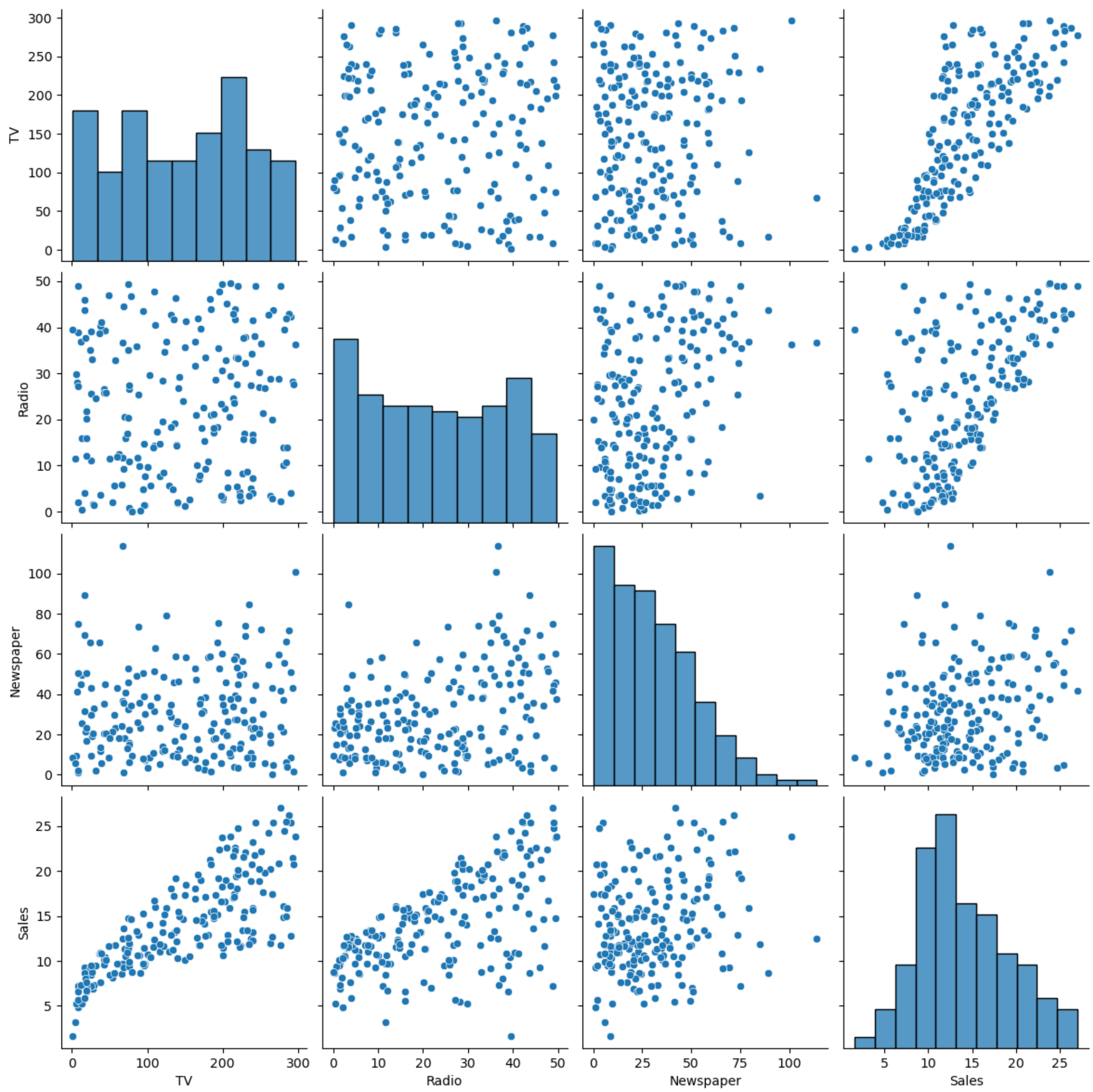
[3]:
```python
df = df.iloc[ : , 1: ]
df.head()
```

[3]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9

[4]:
```python
sns.pairplot(data = df , height=3)
```

[4]: <seaborn.axisgrid.PairGrid at 0x2b028cd7350>



[5]:
```python
x = df[['TV','Radio','Newspaper']]
y = df['Sales']
```

[6]:
```python
x.head()
```

[6]:

	TV	Radio	Newspaper
0	230.1	37.8	69.2
1	44.5	39.3	45.1
2	17.2	45.9	69.3
3	151.5	41.3	58.5
4	180.8	10.8	58.4

[7]:
```python
y.head()
```

[7]:
```
0    22.1
1    10.4
2     9.3
3    18.5
4    12.9
Name: Sales, dtype: float64
```

[8]:
```
X_train,X_test,y_train,y_test = train_test_split(x,y,test_size = 0.
↪20,random_state=42)
```

[9]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
def calc_vif(X):
    vif = pd.DataFrame()
    vif["VIF"] = [variance_inflation_factor(X.values,i) for i in range(X.
↪shape[1])]
    return(vif)
```

[10]:
```
X = df.iloc[:,:-1]
calc_vif(X)
```

[10]:

	VIF
0	2.486772
1	3.285462
2	3.055245

[11]:
```
alpha_values = np.linspace(-3, 3, 10)
print(alpha_values)
```

```
[-3.         -2.33333333 -1.66666667 -1.         -0.33333333  0.33333333
  1.          1.66666667  2.33333333  3.        ]
```

[12]:
```
from sklearn.linear_model import Ridge
# Ridge Regression with hyperparameter tuning
ridge = Ridge()
# Define the hyperparameter grid for alpha values (regularization strength)
alpha_values = { 'alpha': np.logspace(-3, 3, 10)} # 10 values from 10^-3 to 10^3

ridge_cv = GridSearchCV(ridge, alpha_values, cv=5,␣
↪scoring='neg_mean_squared_error')
ridge_cv.fit(X_train, y_train) # Train Ridge regression model with␣
↪cross-validation

GridSearchCV(cv=5, estimator=Ridge(), param_grid={'alpha': np.logspace(-3, 3,␣
↪10)}, scoring='neg_mean_squared_error')
```

[12]:
```
GridSearchCV(cv=5, estimator=Ridge(),
             param_grid={'alpha': array([1.00000000e-03, 4.64158883e-03,
2.15443469e-02, 1.00000000e-01,
```

```
        4.64158883e-01, 2.15443469e+00, 1.00000000e+01, 4.64158883e+01,
        2.15443469e+02, 1.00000000e+03])},
              scoring='neg_mean_squared_error')
```

[13]:
```
# Best Ridge Model
best_ridge = ridge_cv.best_estimator_
y_pred_ridge = best_ridge.predict(X_test) # Predictions on test data
```

[14]:
```
# Ridge RMSE
ridge_rmse = np.sqrt(mean_squared_error(y_test, y_pred_ridge))
r2_ridge = r2_score(y_test,y_pred_ridge)
print("Best ridge Alpha :",ridge_cv.best_params_['alpha'])
print("RMSE :", ridge_rmse)
print("R2 :",r2_ridge)
```

```
Best ridge Alpha : 0.001
RMSE : 1.7815996608176399
R2 : 0.8994380241817195
```

[15]:
```
from sklearn.linear_model import Lasso
# Lasso Regression with hyperparameter tuning
lasso = Lasso()
lasso_cv = GridSearchCV(lasso, alpha_values, cv=5,␣
 ↪scoring='neg_mean_squared_error')
# Define the hyperparameter grid for alpha values (regularization strength)
lasso_cv.fit(X_train, y_train) # Train Lasso regression model with␣
 ↪cross-validation
GridSearchCV(cv=5, estimator=Lasso(), param_grid={'alpha': np.logspace(-3, -3,␣
 ↪10)}, scoring='neg_mean_squared_error')
```

[15]:
```
GridSearchCV(cv=5, estimator=Lasso(),
             param_grid={'alpha': array([0.001, 0.001, 0.001, 0.001, 0.001,
0.001, 0.001, 0.001, 0.001,
       0.001])},
             scoring='neg_mean_squared_error')
```

[16]:
```
# Best Lasso Model
best_lasso = lasso_cv.best_estimator_
y_pred_lasso = best_lasso.predict(X_test) # Predictions on test data
```

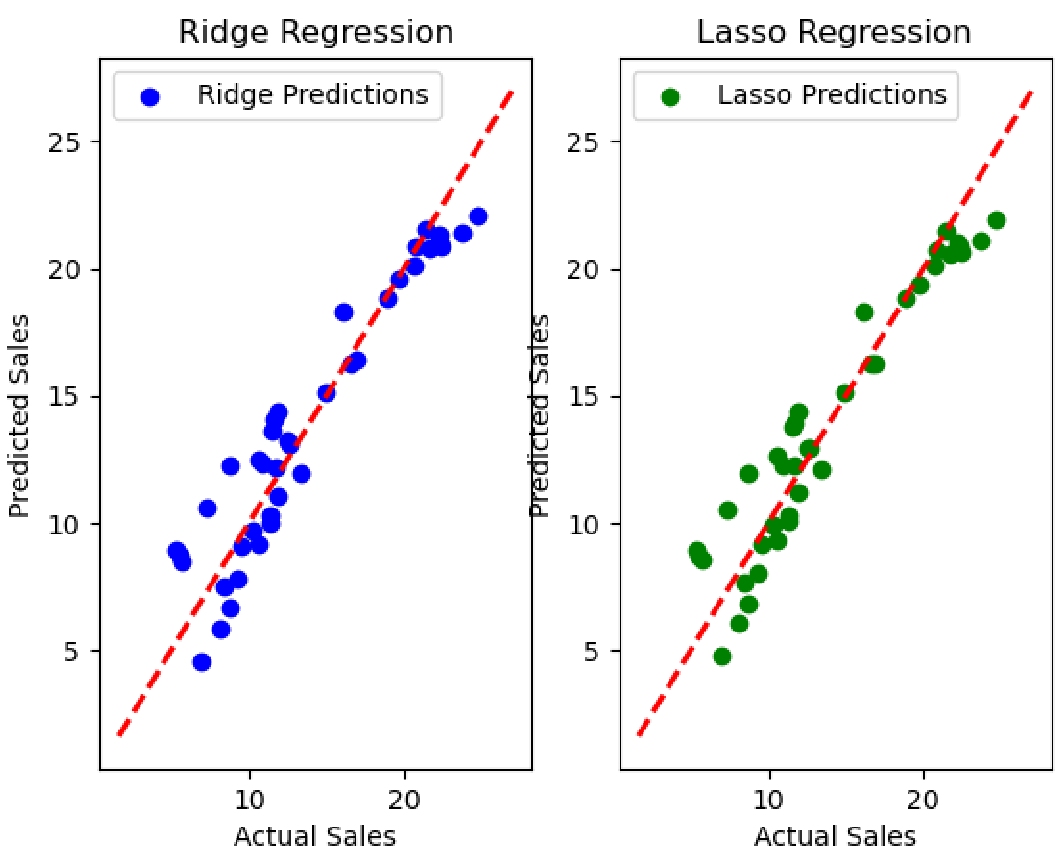
[17]:
```
# Lasso RMSE
lasso_rmse = np.sqrt(mean_squared_error(y_test, y_pred_lasso))
r2_lasso = r2_score(y_test,y_pred_lasso)
print("Best Lasso Alpha :",lasso_cv.best_params_['alpha'])
print("RMSE :", lasso_rmse)
print("R2 :",r2_lasso)
```

```
Best Lasso Alpha : 2.154434690031882
```

```
RMSE : 1.7677097690307533
R2 : 0.9009999351697155
```

[18]:
```python
# Ridge Regression Plot
plt.subplot(1,2,1)
plt.scatter(y_test,y_pred_ridge,color='blue',label="Ridge Predictions")
plt.plot([y.min(),y.max()],[y.min(),y.max()],'r--',lw=2)
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Ridge Regression')
plt.legend()

# Lasso Regression Plot
plt.subplot(1,2,2)
plt.scatter(y_test,y_pred_lasso,color='green',label="Lasso Predictions")
plt.plot([y.min(),y.max()],[y.min(),y.max()],'r--',lw=2)
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')
plt.title('Lasso Regression')
plt.legend()
plt.show()
```



[19]:

```
# Print the coefficients of the lasso with feature names
print("Lasso Coefficients :")
for feature, coef in zip(x.columns, best_lasso.coef_):
    print(f"feature : {feature}, Coefficient , {coef} ")
print("Ridge Coefficients :")
for feature, coef in zip(x.columns, best_lasso.coef_):
    print(f"feature : {feature}, Coefficient , {coef} ")
```

```
Lasso Coefficients :
feature : TV, Coefficient , 0.044516937884577404
feature : Radio, Coefficient , 0.1808414992503233
feature : Newspaper, Coefficient , 0.0
Ridge Coefficients :
feature : TV, Coefficient , 0.044516937884577404
feature : Radio, Coefficient , 0.1808414992503233
feature : Newspaper, Coefficient , 0.0
```