**Machine Learning Case Study**

M. Ruthvik Reddy

CB.SC.P2AIE25013

**Regression**

Regression is a type of supervised learning (in machine learning) and a statistical method (in statistics) used to model and analyze the relationship between one dependent variable (target) and one or more independent variables (features). The goal of regression is to predict continuous numerical values.

The dependent variable (Y) is what we want to predict. The independent variables (X) are the inputs/features that influence Y. And the mapping function given as:

Y≈f(X)+ϵ

**Case Study**

**Dataset and processing:** In this case study, Advertising Dataset was used. This dataset describes about how advertising in TV, Radio, Newspaper impacted the Sales. And the features in the dataset are:

| ['TV', 'Radio', 'Newspaper', 'Sales'] |
| --- |

* 'Sales' is the dependent feature, and others ['TV', 'Radio', 'Newspaper'] are the independent features.
* The data is normalized using sklearn.preprocessing.MinMaxScaler(), where it scales down the data in [0, 1] range, as this is efficient process.
* Split the data into X\_train, X\_val, y\_train, y\_val, in this 85% of data was considered for training and other 15% was considered for validation (or testing).

**Linear Regression**

Linear Regression is a supervised learning algorithm used to model the relationship between a dependent variable (Y) and one or more independent variables (X) by fitting a straight line (or hyperplane).

Y=β0​+β1​X1​+β2​X2​+⋯+βn​Xn​+ϵ

* To perform Linear Regression, sklearn.linear\_model.LinearRegression() was used.

Fitting data and predicting values:

| LinearRegressionV1 = LinearRegression().fit(X\_train, y\_train) y\_perd = LinearRegressionV1.predict(X\_val) Y\_perd  array([[0.27135673],  [0.55888557],  [0.2409764 ],  [0.50745078],  [0.60851327],  [0.91680504],  [0.77621641],  [0.51592903],  [0.72178291],  ........  [0.65064665]]) |
| --- |

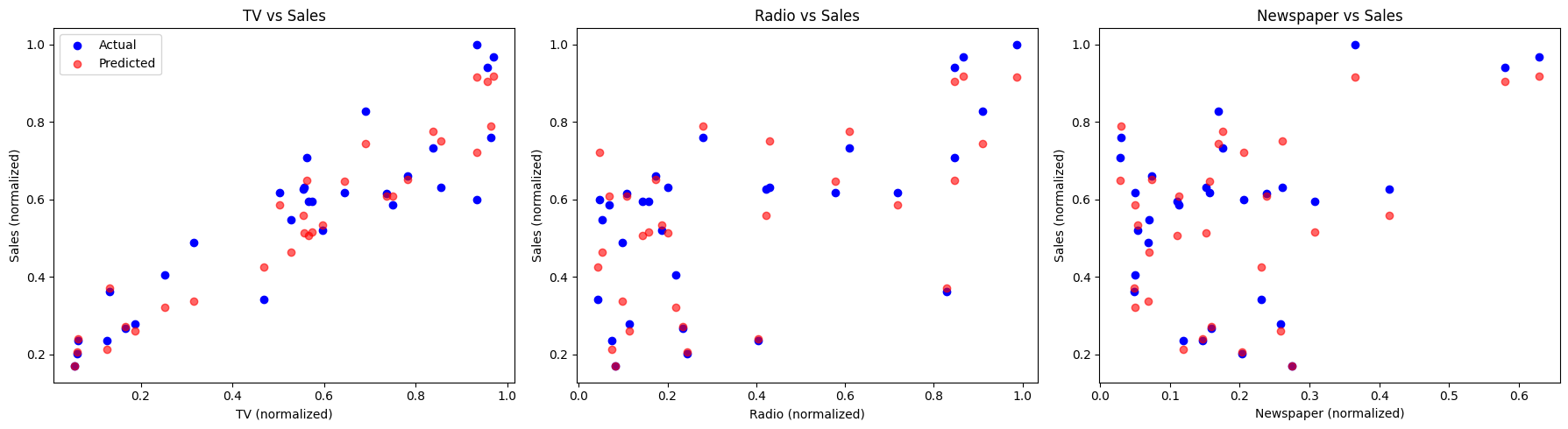
R2 score and Mean-Squared Error (MSE):

| from sklearn.metrics import mean\_squared\_error, r2\_score  mse = mean\_squared\_error(y\_val, y\_perd) r2 = r2\_score(y\_val, y\_perd)  mse, r2 |
| --- |

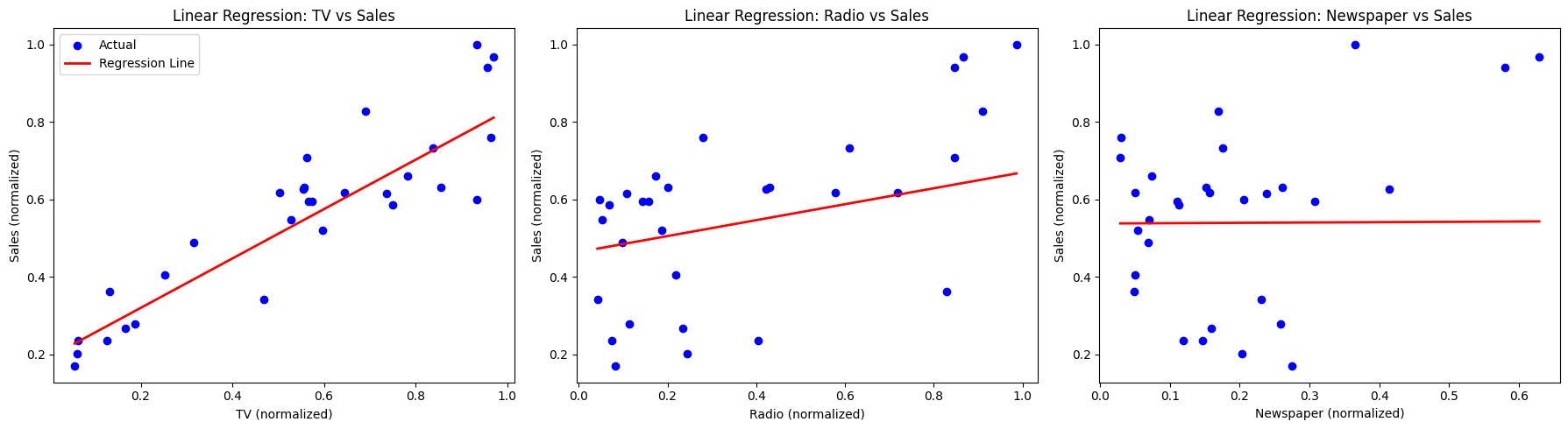
| (0.004435455043519201, 0.9100791189160852) |
| --- |

And plots for actual and predicted values:

| fig, axs = plt.subplots(1, 3, figsize=(18, 5))  feature\_names = ['TV', 'Radio', 'Newspaper'] for i, ax in enumerate(axs):  ax.scatter(X\_val[:, i], y\_val, color='blue', label='Actual')  ax.scatter(X\_val[:, i], y\_perd, color='red', label='Predicted', alpha=0.6)  ax.set\_xlabel(f'{feature\_names[i]} (normalized)')  ax.set\_ylabel('Sales (normalized)')  ax.set\_title(f'{feature\_names[i]} vs Sales')  if i == 0:  ax.legend()  plt.tight\_layout() plt.show() |
| --- |



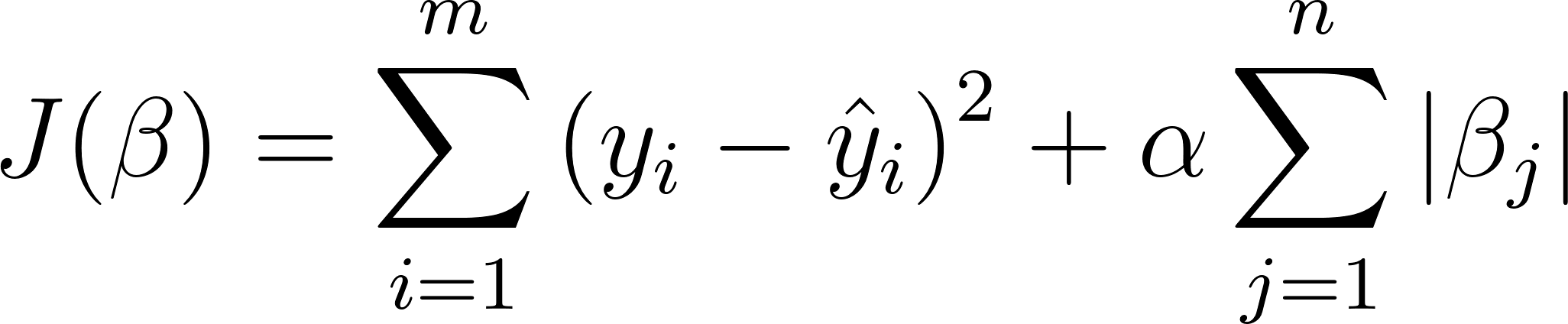
And Plot for projection line:



**Lasso Regression (L1 Norm)**

Lasso (Least Absolute Shrinkage and Selection Operator) Regression adds a penalty term on the sum of absolute coefficients.

Equation:



* For Lasso Regression, we are using same dataset and same features as the linear regression.
* And to perform sklearn.linear\_models.Lasso is used.
* But before doing Lasso regression, best alpha value should be found, which decides the best regularization rate.
* And to that sklearn.linear\_models.LassoCV is used

| from sklearn.linear\_model import LassoCV LassoCV = LassoCV(alphas=[0.01, 0.1, 1, 10], cv=5, random\_state=0) LassoCV.fit(X\_train, y\_train)  LassoCV.alpha\_ np.float64(0.01) |
| --- |

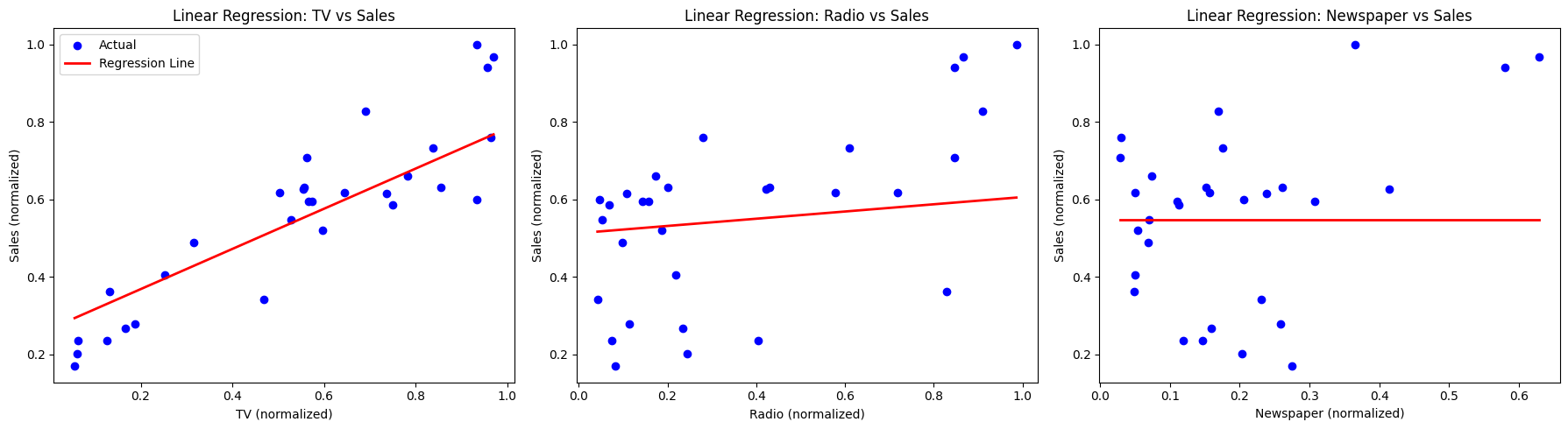
Now to perform Lasso Regression with alpha=0.01:

| from sklearn.linear\_model import Lasso LassoV1 = Lasso(alpha=0.01, random\_state=42) LassoV1.fit(X\_train, y\_train)  y\_pred = LassoV1.predict(X\_val) y\_pred array([0.33888146, 0.55707519, 0.30155842, 0.53798583, 0.62569814,  0.80671941, 0.72162626, 0.54245638, 0.718821 , 0.5385256 ,  0.50884591, 0.62261875, 0.33866586, 0.81439244, 0.71437211,  0.67353758, 0.47792977, 0.61835967, 0.75691718, 0.28495102,  0.37589655, 0.38208071, 0.26729182, 0.6007479 , 0.55874738,  0.30335393, 0.40377797, 0.55719503, 0.80550162, 0.6524718 ]) |
| --- |

R2 score and MSE:

| from sklearn.metrics import mean\_squared\_error, r2\_score mse = mean\_squared\_error(y\_val, y\_pred) r2 = r2\_score(y\_val, y\_pred) mse, r2 (0.007454552537285793, 0.848872342147059) |
| --- |

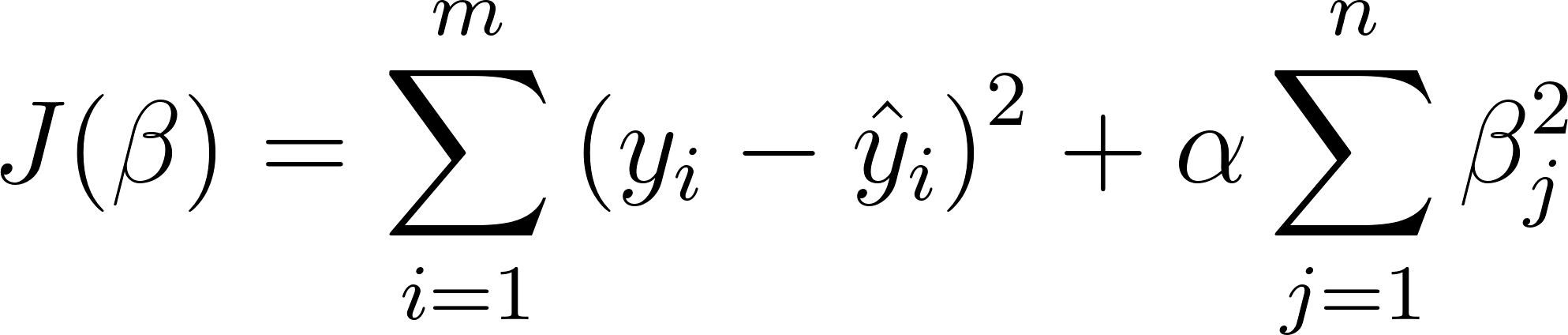
And Plots for Projection Line:



**Ridge Regression (L2 Norm)**

Ridge Regression is a regularized version of Linear Regression that adds a penalty term on the sum of squared coefficients to prevent overfitting.

Equation:



* For Ridge Regression, we are using same dataset and same features as the linear regression.
* And to perform sklearn.linear\_models.Ridge is used, and same regularization rate as Lasso Regression is used.

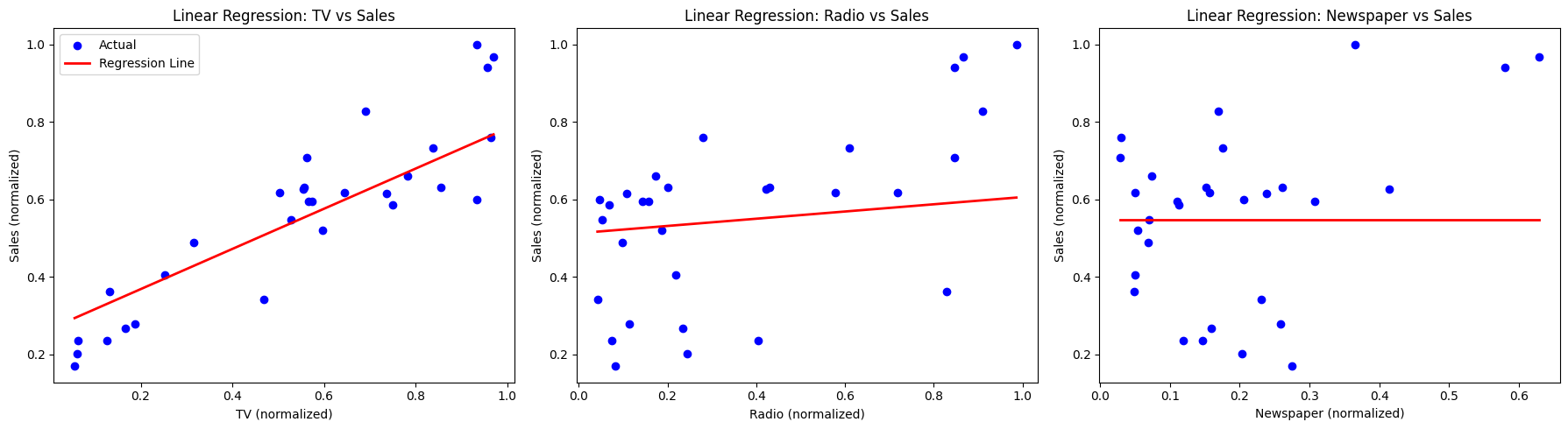
Now to perform RidgeRegression with alpha=0.01:

| from sklearn.linear\_model import Ridge RidgeV1 = Ridge(alpha=0.01, random\_state=42) RidgeV1.fit(X\_train, y\_train) y\_pred = RidgeV1.predict(X\_val) y\_pred array([0.27152969, 0.55887675, 0.24116862, 0.50745202, 0.60844345,  0.91653341, 0.77602962, 0.51594252, 0.72164146, 0.51380441,  0.46304964, 0.60942461, 0.26130163, 0.91742744, 0.75152639,  0.74424124, 0.42555005, 0.64624832, 0.78820264, 0.20657371,  0.37017876, 0.32218526, 0.1708243 , 0.6479026 , 0.58474135,  0.21303253, 0.33817437, 0.53459925, 0.90424105, 0.65054256]) |
| --- |

R2 score and MSE:

| from sklearn.metrics import mean\_squared\_error, r2\_score mse = mean\_squared\_error(y\_val, y\_pred) r2 = r2\_score(y\_val, y\_pred) mse, r2  (0.004434287872727083, 0.9101027812066445) |
| --- |

And Plots for Projection Line:



**Conclusion:** Ridge Regression performed well, and following with Linear Regression and Lasso Regression. The Lasso and Ridge change the learning based on the alpha value, so each has its own best alpha value. But all the projections were similar (close).