Statistics DRP Autumn 2025 Final Write Up
Introduction to Statistical Learning with Applications

Mentee: Duc Huy Nguyen
Mentor: Patrick Campbell

1 Regression

We predict a continuous values (e.g. prices, life expectancy, etc.)

1.1 Multiple Linear Regression (MLR)

We predict response variable Y based linear relationship with p predictors.

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$

where X_j represents the j^{th} predictor and β_j quantifies the relationship between Y and one unit increase in X_j. We interpret β_j as the average effect on Y of a one unit increase in X, assuming other variables fixed.

We fit MLR by minimize the sum of squared residuals (RSS):

$$\begin{aligned} RSS &= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \cdots - \hat{\beta}_p x_{ip})^2 \end{aligned}$$

1.2 Shrinkage Models

Reduce the linear coefficients to remove noisy data and overfitting.

Ridge Regression: shrink the coefficients of the model by adding shrinkage penalty of sum of squared coefficients to the optimization problem:

$$\underset{\hat{\beta}_R}{\text{minimize}} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j X_{ij} \right) + \lambda \sum_{j=1}^{p} \beta_j^2 = \text{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2 \quad \text{where } \lambda \geq 0 \text{ is a tuning parameter}$$

LASSO Regression: shrink the coefficients of the model by adding shrinkage penalty of sum of coefficients to the optimization problem:

$$\underset{\hat{\beta}_L}{\text{minimize}} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j X_{ij} \right) + \lambda \sum_{j=1}^{p} \beta_j = \text{RSS} + \lambda \sum_{j=1}^{p} \beta_j \quad \text{where } \lambda \geq 0 \text{ is a tuning parameter}$$

Standardization: since our optimization for Ridge and LASSO also consider magnitudes of coefficients so we should scale our predictors' values before fitting to improve performance:

$$z = \frac{x - \mu}{\sigma}$$

2 Classification

2.1 Logistic Regression

We model the response Y based on its probability of belonging to a particular category:

$$log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

And we fit then model by maximizing the likelihood function rather than minimizing RSS:

$$\ell(\beta_0, \beta_1, \cdots, \beta_p) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=1} (1 - p(x'_i))$$

2.2 Discriminant Analysis

The idea is that we model the distribution of predictors of X within each of the response classes using conditional probability. We then use Bayes' Theorem to flip this into estimates for $Pr(Y = k|X = x)$:

$$Pr(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{k} \pi_l f_l(x)}$$

Estimating the density function of X given k is problematic so we make simplifying assumptions about properties of X.

Linear Discriminant Analysis:
We assume that our predictors are normally distributed so we have the density functions of the Normal/Gaussian distributions for multiple predictors:

$$f(x) = \frac{1}{(2\pi)^{p/2} |\sum|^{1/2}} exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

Naive Bayes:
We assume that p predictors are conditionally independent within the k class:

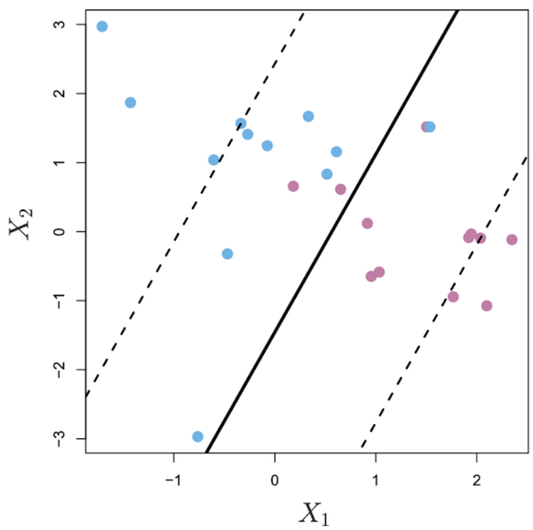
$$f_k(x) = \prod_{j=1}^{p} f_{jk}(x_j)$$

or in other words, given they are in the same class k the covariance matrix is diagonal with the non-zero elements being 1:

$$\Sigma_{X|Y=k} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 1 \end{pmatrix}$$

2.3 Support Vector Machine (SVM)

A different category of classifier is Support Vector Classifier in which we want to fit a hyperplane separating data into classes.



We can reduce our problem into a convex optimization problem:

$$\underset{\beta_0,\beta_1,\cdots,\beta_p,\epsilon_1,\cdots,\epsilon_n,M}{maximize} \quad M$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} \geq M(1 - \epsilon_i)$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C$$

where: M is the minimal distance between any points and the decision boundary
and C is our total budget for errors ϵ_i of how the point violates our margin

Solving the optimization problem gives us the solution: $f(x) = \beta_0 + \sum_{i \in S} \alpha_i \langle x, x_i \rangle$ in which $\langle x, x_i \rangle$ is the inner product between observations and predictions and S is the Support Vectors - data points that are closest to the decision boundary

Kernel: Our generalization of the inner product between prediction and actual. For instance, using the kernel $K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$ would gives us the same linear support vector classifier

Radial Kernel: Another kernel being widely used, would would produce a more circular decision boundary:

$$K(x_i, x_{i'}) = exp\left(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2\right)$$

where γ is one of our tuning parameter for the smoothness of the boundary. The intuition is that as predictions get further from actual observation, its importance to the kernel decrease.

2.4 Classification Model Metrics:

We also have several basic metrics to evaluate our models

Precision: measure how much of our positives are actually positives

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall: measures the proportion of the positives label that are actually classified

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

F1 Score: an metrics combine both Precision and Recall

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

2.5 Visualization of our Classifiers

The following graphic would help us visualize our decision boundary better:

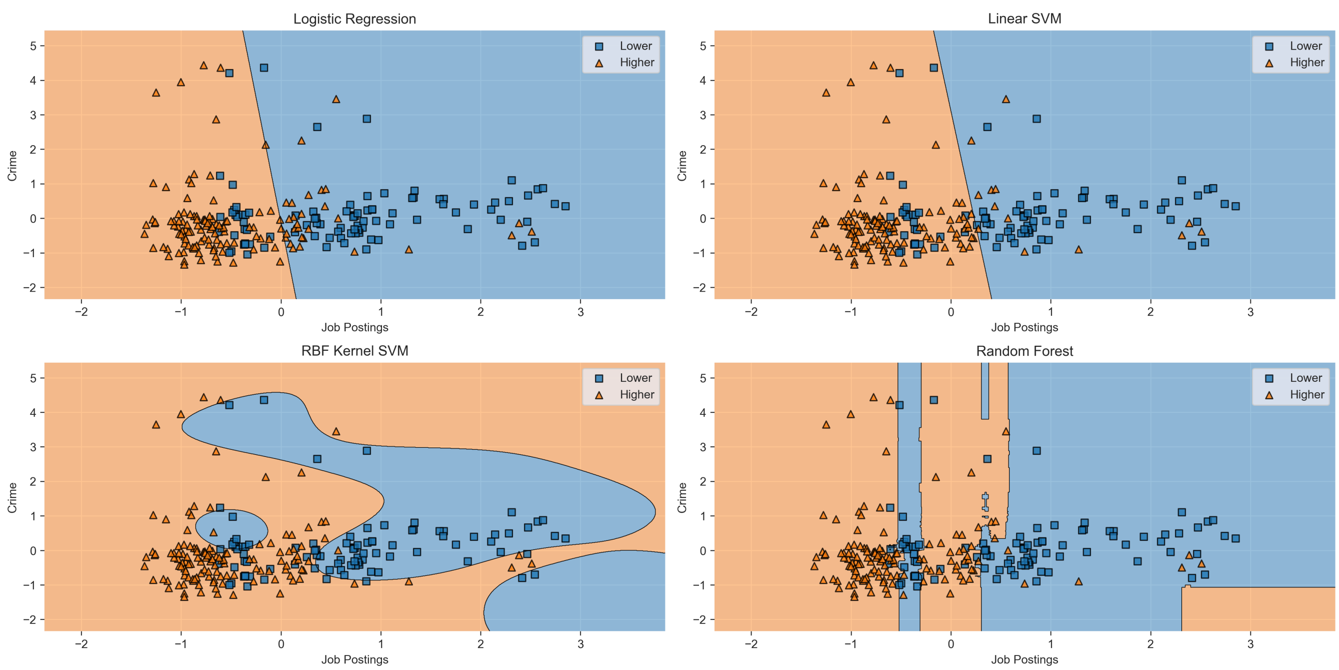


Figure 1: Plot of Different Classifiers

References

James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). *An introduction to statistical learning: With applications in python*. Springer International Publishing Springer.