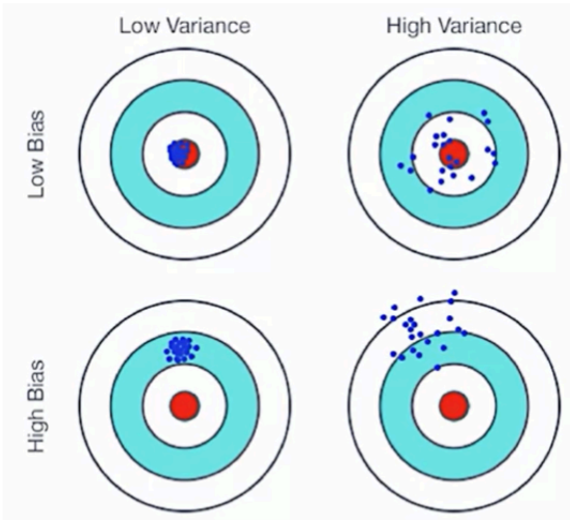
SUPERVISED MACHINE LEARNING: REGRESSION

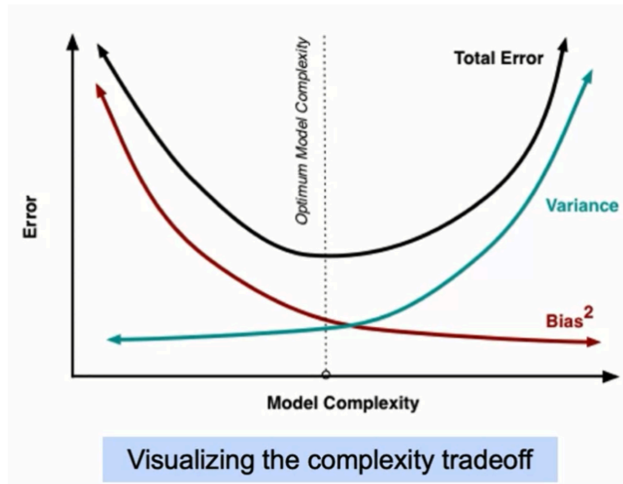
MODULE 4:

BIAS VARIANCE TRACE OFF AND REGULARIZATION TECHNIQUES: RIDGE, LASSO AND ELASTIC NET

TABLE OF CONTENTS

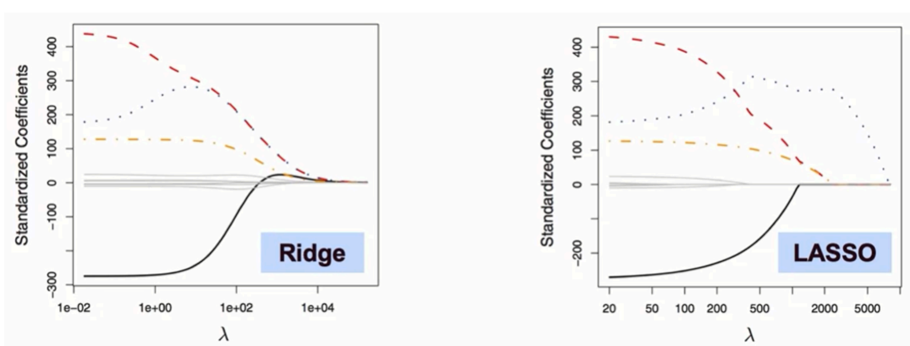
Bias–Variance Trade-Off





- Increasing regularization (larger alpha):
  - Increases **bias**
  - Decreases **variance**
- Too little regularization → overfitting
- Too much regularization → underfitting
- The best model strikes a balance between the two.

Regularized Linear Models



Regularization reduces overfitting by adding a **penalty term** to the cost function, keeping coefficients small and improving generalization.

$$\boxed{\textit{Minimize (Loss+Penalty)}}$$

Ridge Regression (L2 Regularization)

Adds a squared penalty:

$$L(\beta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \underbrace{\boxed{\lambda \sum_{j=1}^{p} |\beta_j|}}_{\text{Regularization}}$$

- Shrinks all coefficients but keeps them nonzero.
- Useful when all predictors contribute slightly.

```
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error

ridge = Ridge(alpha=1.0)
ridge.fit(X, y)                          # Train model
print("Ridge Coefficients:", ridge.coef_)
```

Lasso Regression (L1 Regularization)

Adds an absolute penalty:

$$L(\beta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \underbrace{\boxed{\lambda \sum_{j=1}^{p} \beta_j^2}}_{\text{Regularization}}$$

- Some coefficients become exactly zero → **automatic feature selection**.
- Best when only a few features are important.

```
from sklearn.linear_model import Lasso

lasso = Lasso(alpha=0.05)
lasso.fit(X, y)
print("Lasso Coefficients:", lasso.coef_)
```

Elastic Net (Combination of L1 and L2)

Combines Ridge and Lasso penalties:

$$L(\beta) = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \underbrace{\boxed{\lambda_1 \sum_{j=1}^{p}|\beta_j| + \lambda_2 \sum_{j=1}^{p}\beta_j^2}}_{\text{Regularization}}$$

- Balances Lasso’s sparsity and Ridge’s stability.
- Works well for correlated features.

```
from sklearn.linear_model import ElasticNet

elastic = ElasticNet(alpha=0.5, l1_ratio=0.5)
elastic.fit(X, y)
print("Elastic Net Coefficients:", elastic.coef_)
```

Selecting the Best Alpha with Cross-Validation

- Scikit-learn provides built-in models that automatically choose the best alpha (regularization strength) using K-Fold Cross-Validation.

    - **RidgeCV**

```
from sklearn.linear_model import RidgeCV

# Define candidate alpha values
alphas = [0.01, 0.1, 1.0, 10.0]

# Automatically select best alpha via 5-fold CV
ridge_cv = RidgeCV(alphas=alphas, cv=5)
ridge_cv.fit(X, y)

print("Best alpha (RidgeCV):", ridge_cv.alpha_)
print("R² Score:", ridge_cv.score(X, y))
```

- **LassoCV**

```
from sklearn.linear_model import LassoCV

# Automatically tunes alpha using cross-validation
lasso_cv = LassoCV(alphas=[0.001, 0.01, 0.1, 1.0], cv=5,
random_state=42)
lasso_cv.fit(X, y)

print("Best alpha (LassoCV):", lasso_cv.alpha_)
print("Selected Coefficients:", lasso_cv.coef_)
print("R² Score:", lasso_cv.score(X, y))
```

- **ElsticNetCV**

```
from sklearn.linear_model import ElasticNetCV

# Elastic Net tunes both alpha and l1_ratio
elastic_cv = ElasticNetCV(
    alphas=[0.001, 0.01, 0.1, 1.0],
    l1_ratio=[0.2, 0.5, 0.8],
    cv=5,
    random_state=42
)
elastic_cv.fit(X, y)

print("Best alpha (ElasticNetCV):", elastic_cv.alpha_)
print("Best l1_ratio:", elastic_cv.l1_ratio_)
print("R² Score:", elastic_cv.score(X, y))
```

Key Takeaways

- **Regularization**
  Adds penalties to limit model flexibility and reduce overfitting.
- **Ridge (L2):** Shrinks all coefficients smoothly.
- **Lasso (L1):** Performs feature selection.
- **Elastic Net:** Balances sparsity and stability.
- **Cross-Validation + Regularization:**
  Use CV (`RidgeCV`, `LassoCV`, `ElasticNetCV`) to pick the best alpha for optimal bias–variance balance.
- **Core Insight:**
  Combining **Cross-Validation** and **Regularization** produces models that are **accurate, interpretable, and generalizable**.