Regularization -Bias Variance Trade-off (VVIMP for Interview)

Asked in ~~almost~~ every interview.

- Only 1% people actually understand this topic
- Can be applied to ALL the ML as well as Deep Learning algorithms.

Bias (Underfitting)

Variance (Overfitting)

> 💡 **Regularization Reduces Overfitting (Variance)**

Bias-Variance Trade-off

Bias and variance are two sources of error in a model:

- **High Bias (Underfitting):** Model is too simple, fails to capture patterns.
- **High Variance (Overfitting):** Model is too complex, captures noise instead of patterns.

💡 **Regularization helps balance bias & variance** by adding a penalty term to the loss function.

Impact of Regularization

Aspect	Effect on Bias	Effect on Variance
No Regularization	**Low Bias**: Model fits training data well.	**High Variance**: Overfits noise in training data.

Aspect	Effect on Bias	Effect on Variance
Moderate Regularization	**Balanced**: Maintains model flexibility while reducing overfitting.	**Reduced**: Penalizes complexity, generalizes better.
High Regularization	**High Bias**: Oversimplifies the model (underfitting).	**Low Variance**: Model becomes rigid and less sensitive to data fluctuations.

For adjusting the variance and bias we use:

1. **Lasso (L1)**
2. **Ridge (L2)**
3. **Elastic Net**

Regularization	Bias	Variance	Effect
No Regularization	Low	High	Overfits
L1 (Lasso)	Medium	Medium	Feature selection + reduced overfitting
L2 (Ridge)	Medium	Lower	Handles collinearity + reduces overfitting
L1 + L2 (Elastic Net)	Medium	Medium	Best of both worlds

Problem:

- You have to predict result for the entire population from sample data.
- The WILL BE some error.
- You can’t do an accurate prediction due to this error.
- So, we try to find out best estimate.
- If we have data for the entire population, the equation will be $y = f(x)$
- The prediction is → $f’(x)$= $\hat{y}$
- $f(x) - f’(x)$ → **Reducible Error**
- You can reduce this 👆 error.

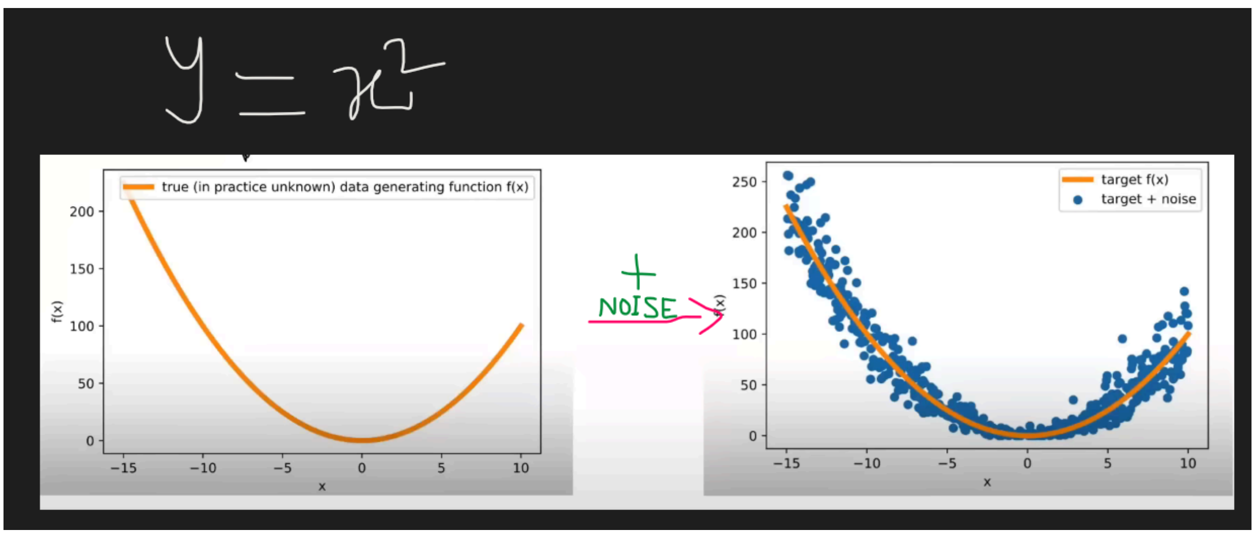
- **You cannot reduce the irreducible error.**

$$ReducibleError = Bias^2 + Variance$$

Bias Variance Trade-off

- We'll go reverse:
    - from Population → to Sample

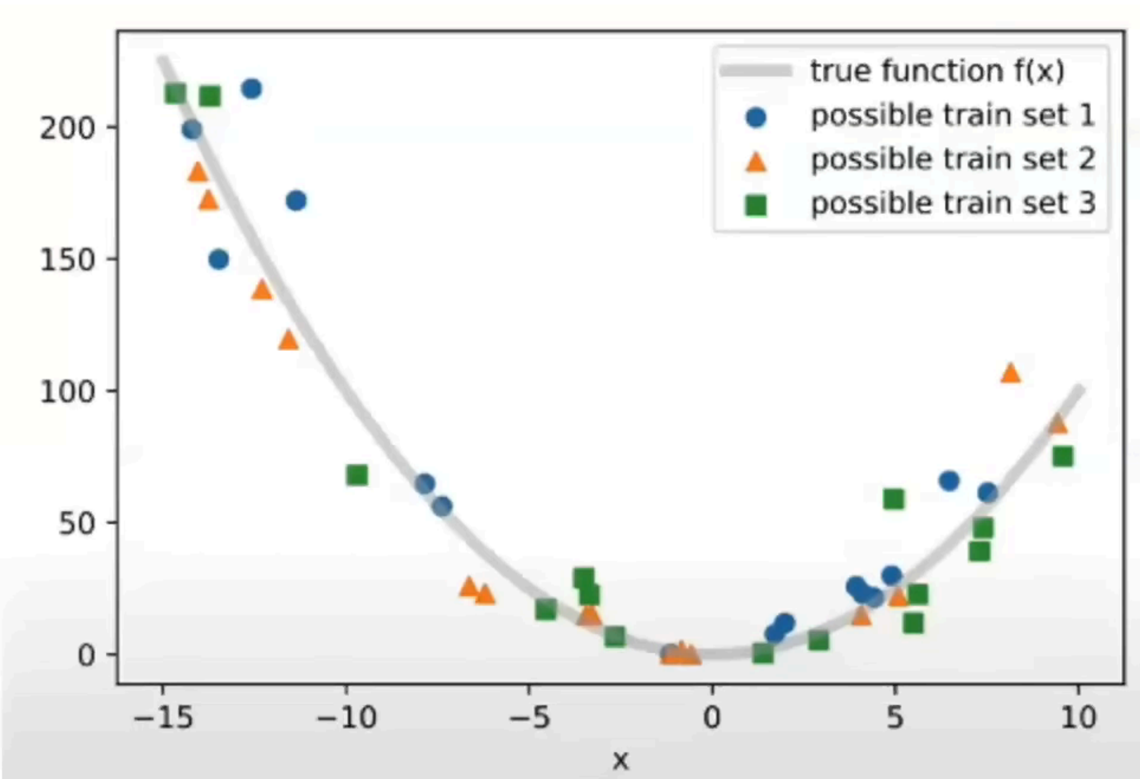
  (In real world, you will not have population data)
- We'll add an error/noise to this



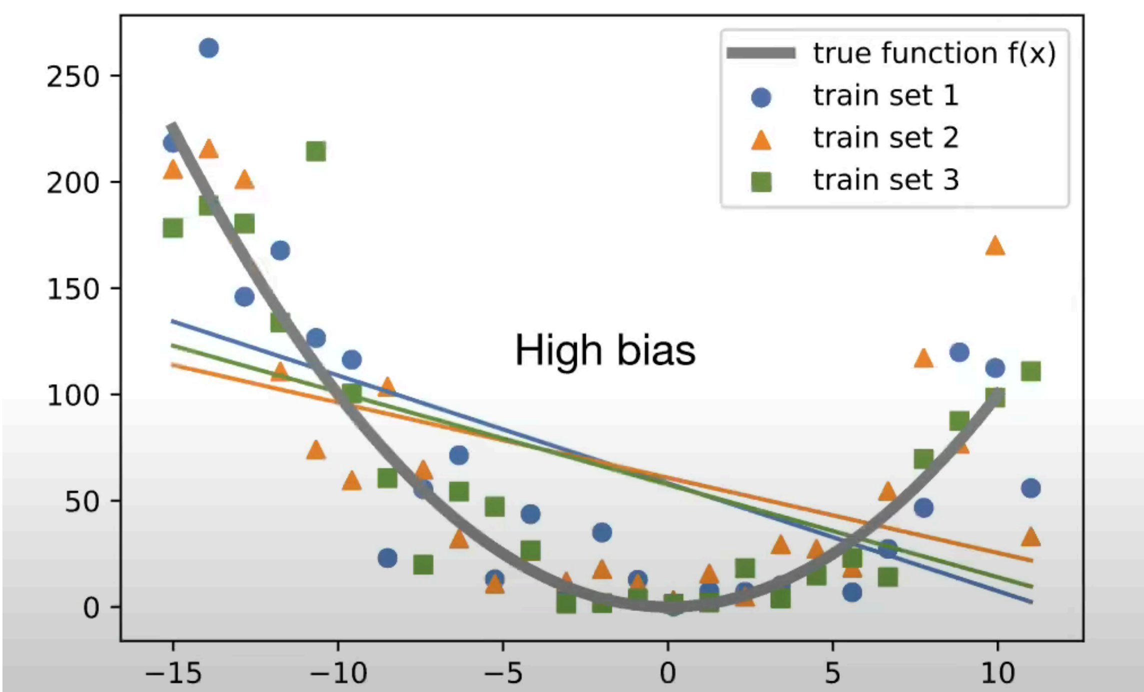
Population data=

$$\boxed{y = x^2 \quad + error}$$
$$(-15, 10)$$

- Draw 3 random **samples** from the above data👆



- We try to fit a linear regression from the sample data:



💡 **BIAS: The inability of a machine learning model to fit the training data**

Underfitting

- Above graph is **HIGH BIAS.**

High Bias = Underfitting

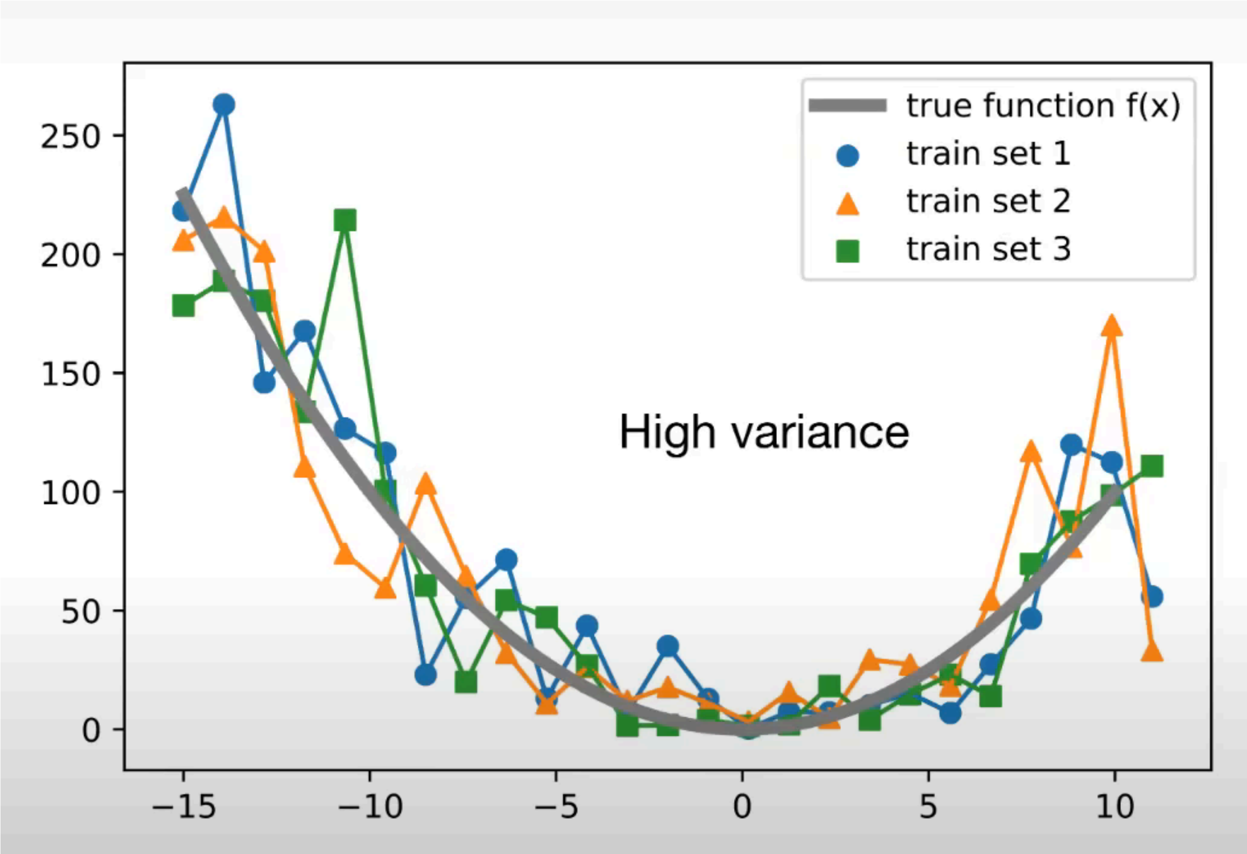
- As the bias decreases, the data starts fitting

💡 **Variance: Change in ML model when data is changed.**

Overfitting👆

- Above graph is **LOW VARIANCE.**
- **Variance** in machine learning refers to the model's **sensitivity to small changes in the training data**.
- A model with high variance overfits, meaning it **captures noise and random fluctuations** in the training data, leading to **poor generalization** to new data.

Now, apply polynomial regression ti the above data.



- The models are Low Bias (less Underfitting)
    - The training data is fitting very well.
- But the results of all three models it's varying from each other.
    - Therefore it's a **high variance** model.

💡 **High variance is closely related to overfitting.**

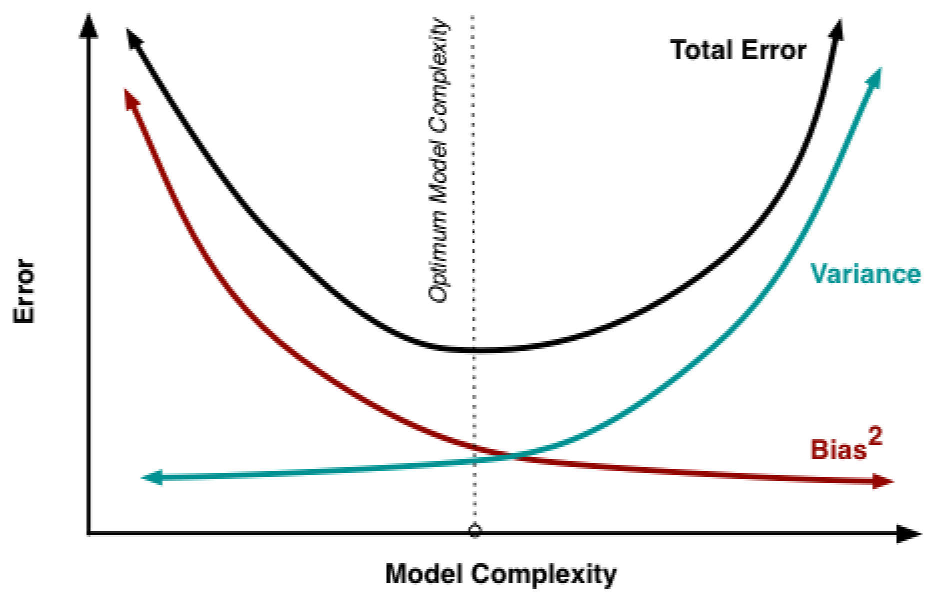
💡 **Underfitting is closely related to high bias.**

Ideal Situation:

Low variance-Low Bias

> **Meaning→ Your data is fitting test data properly. And when you get another training data, the model doesn't drastically change.**

- **Problem → Bias and Variance are inversely proportional to each other.**



- As you increase the complexity, bias decreases
- BUT variance starts to increase.
- **AIM : To find the middle point**

Expected Value and Variance

- Expected value represents the average outcome of a random variable over a large number of trials or experiments.

- We roll a die 1 Lac times
  - Mean will be →3.5
  - This is EXPECTED VALUE

Expected Value $E[X]$ = Population Mean

$Var(X)$ = Variance of Population

$$\begin{aligned}
\operatorname{Var}(X) &= \operatorname{E}\left[(X - \operatorname{E}[X])^2\right] \\
&= \operatorname{E}\left[X^2 - 2X\operatorname{E}[X] + \operatorname{E}[X]^2\right] \\
&= \operatorname{E}\left[X^2\right] - 2\operatorname{E}[X]\operatorname{E}[X] + \operatorname{E}[X]^2 \\
&= \operatorname{E}\left[X^2\right] - 2\operatorname{E}[X]^2 + \operatorname{E}[X]^2 \\
&= \operatorname{E}\left[X^2\right] - \operatorname{E}[X]^2
\end{aligned}$$

Bias and Variance Mathematically?

Bias:

$$\text{Bias}\left(\underline{f'(x)}\right) = E\left[f'(x)\right] - \underline{f(x)}$$

$f(x)$→ Population mean (Expected value of the population)

$f'(x)$ → Sample mean

- If difference between them is zero → Our model is unbiased

> 💡 **If we draw 100 samples & find out the mean→ It will be close to the population mean.**

Variance:

- Variance refers to the amount by which the prediction of our model will change if we used a different training data set.
- In other words, it measures how much the predictions for a given point vary between different realizations of the model.

$$\boxed{Var(f'(x)) = E\left[\left(f'(x) - E[f'(x)]\right)^2\right]}$$

- **If this is high→ Upon changing the data, the accuracy, R2 score, etc will change a lot.**
- Because the model is **OVERFITTING**

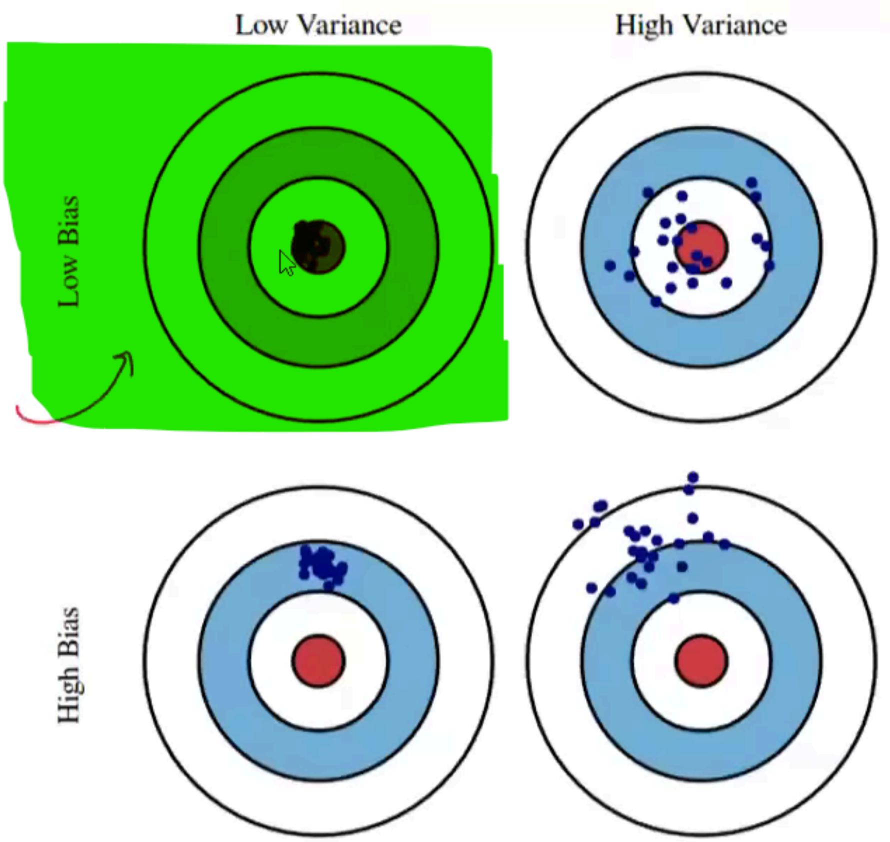


Fig. 1 Graphical illustration of bias and variance.

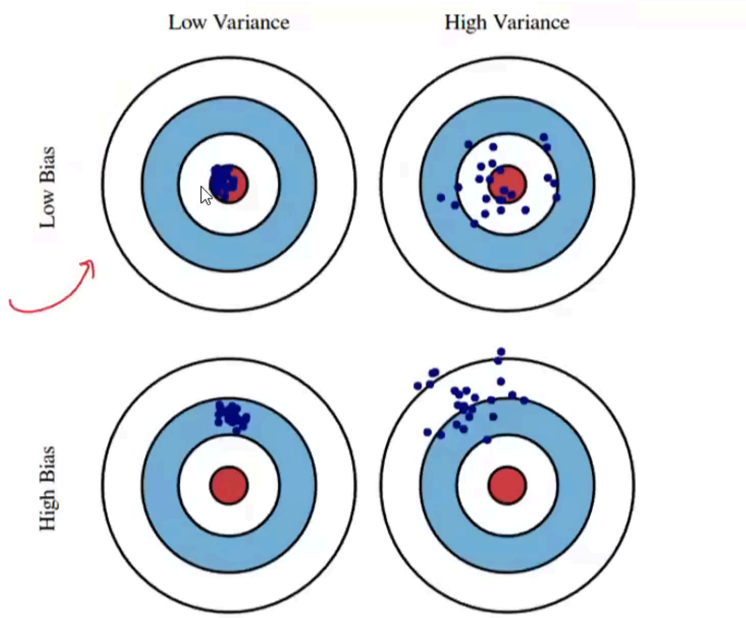


Fig. 1 Graphical illustration of bias and variance.

Bias Variance Decomposition (VIMP)

- It divides the loss (eg. MSE) into 3 parts:
  1. Bias
  2. Variance
  3. Irreducible Error

Total Error = Bias² + Variance + Irreducible Error

> 💡 **VIMP NOTE: Here, we're DECOMPOSING. First we get the loss. Then we split it into these 3.**

- **Irreducible Error (ε)** → Moving target/Noise

Bias² will cancel out the negative values.

- **Bias + Variance → Reducible Errors**
- **High Bias**: Simplified model → Underfitting → Low accuracy.
- **High Variance**: Complex model → Overfitting → Poor generalization.

Code Example

```
from mlxtend.evaluate import bias_variance_decomp
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression
from mlxtend.data import boston_housing_data
from sklearn.model_selection import train_test_split


X, y = boston_housing_data()
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.3,
                                                    random_state=123,
                                                    shuffle=True)
```

```
lr = LinearRegression()

avg_expected_loss, avg_bias, avg_var = bias_variance_decomp(
        lr, X_train, y_train, X_test, y_test,
        loss='mse',
        random_seed=123)

print('Average expected loss: %.3f' % avg_expected_loss)
```

```
print('Average bias: %.3f' % avg_bias)
print('Average variance: %.3f' % avg_var)
```

Output:

Average **expected loss**: 29.891   #**(MSE)**

Average **bias**: 28.609 $\#Bias^2$

Average **variance**: 1.282

`bias_variance_decomp(...)`:

- This function is used to compute the **bias, variance, and expected loss** of the model on a given dataset.
  - `avg_expected_loss`: The average error on the test set (MSE).
  - `avg_bias`: The average bias of the model (how far off the model's predictions are from the true values).
  - `avg_var`: The variance of the model's predictions (how much the predictions fluctuate across different data subsets).

Total Error = Bias² + Variance + Irreducible Error

- Linear Regression gives **high bias** (predictions are away from actual value) &
- **Low variance** (It's precise when ran multiple times)
- `DecisionTreeRegressor` is opposite of linear regression

Let's apply `DecisionTreeRegressor` on same dataset:

```
dt = DecisionTreeRegressor(random_state=123)

avg_expected_loss, avg_bias, avg_var = bias_variance_decomp(
```

```
        dt, X_train, y_train, X_test, y_test,
        loss='mse',
        random_seed=123)

print('Average expected loss: %.3f' % avg_expected_loss)
print('Average bias: %.3f' % avg_bias)
print('Average variance: %.3f' % avg_var)
```

Output:

Average expected loss: 31.536
Average bias: 14.096
Average variance: 17.440

- Here, bias has reduced but variance is increased as compared to linear regression

When to use Regularization?

1. **Prevent Overfitting**
    - Use when your model performs well on training data but poorly on validation/test data.
2. **High Dimensionality (Many Features, Few Samples)**
    - Use L1/L2 to reduce model complexity and avoid overfitting (e.g., text/gene data).
3. **Multicollinearity (Correlated Features)**
    - Use **Ridge (L2)** to stabilize coefficients and distribute weights among correlated features.
4. **Feature Selection**
    - Use **Lasso (L1)** to shrink irrelevant features' coefficients to zero, retaining only important ones.
5. **Improve Interpretability**

- Simplify models by reducing feature count (L1) or shrinking coefficients (L2).

6. **Boost Model Performance**
    - Apply regularization to enhance out-of-sample performance, even if overfitting isn't evident.