SPPU

MACHINE LEARNING

UNIT 3

Supervised Learning : Regression

Contents

- Bias
- Variance
- Generalization
- Underfitting
- Overfitting
- Linear regression
- Regression: Lasso regression, Ridge regression, Gradient descent algorithm.
- Evaluation Metrics: MAE, RMSE, R2

Bias

Bias = Error due to wrong assumptions in the model.

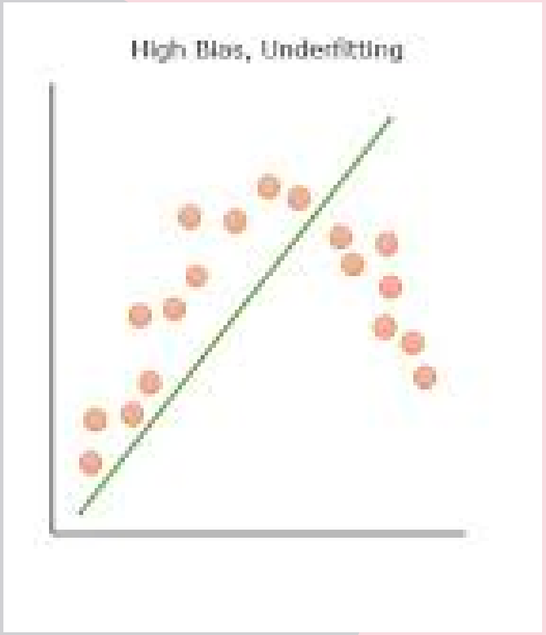
A high-bias model is too simple, doesn’t learn enough patterns, and underfits.

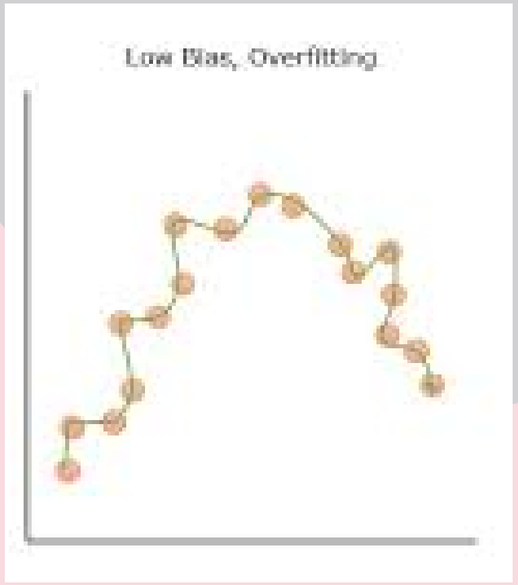
Example

- You want to predict house prices using only one feature: size of the house.
- But house price also depends on:
- location
- number of rooms
- condition
- demand

Your model is too simple → so predictions are far from real values.

This is high bias.





Technical Understanding

A high-bias model:

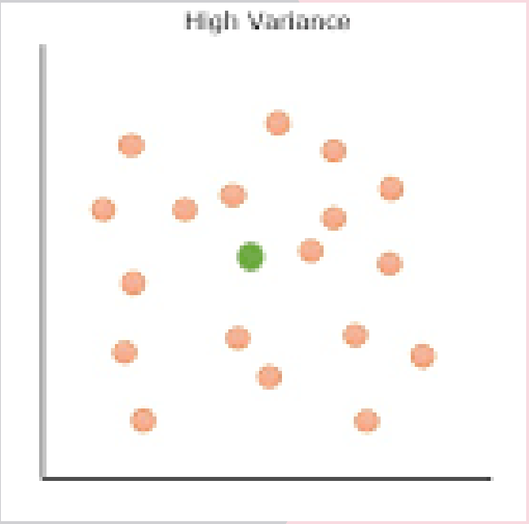
- ignores important features
- oversimplifies the relationship
- cannot capture complexity
- assumes data follows a simple pattern even if it doesn’t

Variance

Variance means the model is too sensitive to training data, so it gives inconsistent answers.

It happens when the model is too complex and tries to memorize everything.

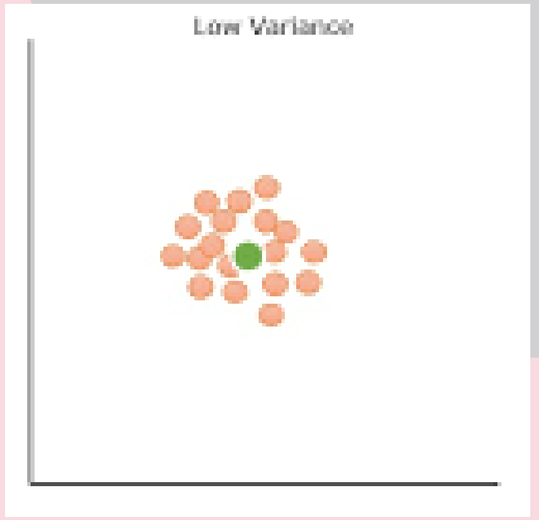
- Variance is the error that comes because the model changes a lot depending on training data.
- The model learns noise instead of patterns.
- So predictions become unstable:
- sometimes very high, sometimes very low.
- This leads to overfitting → model performs great on training data but fails on new data.



Technical Understanding

A high-variance model:

- is too complex
- memorizes training data
- does not generalize
- reacts strongly to even small changes in data



Bias–Variance Tradeoff

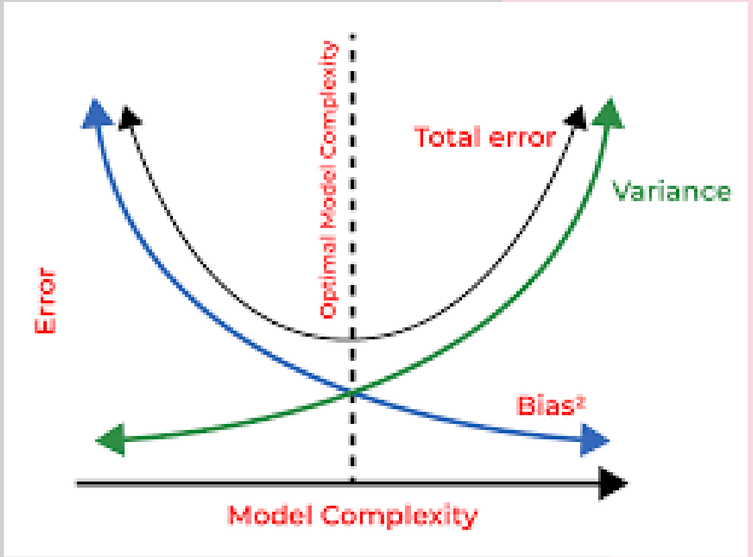
The Bias–Variance Tradeoff means:

To make a good model, you must balance bias (too simple) and variance (too complex).

If:

- Bias is high → model underfits
- Variance is high → model overfits

A good model keeps both low, not just one.



Why is it a “Tradeoff”?

Because:

- When you reduce bias,

  the model becomes more complex → variance increases.
- When you reduce variance,

  the model becomes simpler → bias increases.

So you must find the middle point (the sweet spot).

Generalization, Underfitting, Overfitting

Generalization is the model's ability to perform well on new, unseen data — not just the data it was trained on.

A model that generalizes well:

- learns the true pattern in data,
- not the noise or unnecessary details

Important?

- Real-world data will always be different from training data.
- A good ML model must perform correctly on future unseen data.
- Too simple → underfits.
- Too complex → overfits.
- Good generalization lies in the middle

Example : A model trained to identify cats should correctly recognize:

- different breeds,
- different angles,
- different backgrounds.

Not just the exact images from the training set.

To Improve Generalization

- Use more diverse training data – Helps the model learn patterns that work for many different real-world situations.
- Use regularization (L1, L2, dropout) – Prevents the model from becoming too complex and reduces overfitting.
- Use cross-validation – Ensures the model performs well on multiple splits of the data, not just one.
- Use simpler model architecture – Reduces unnecessary complexity so the model focuses on important patterns.
- Early stopping during training – Stops training before the model starts memorizing the data.
- Data augmentation (images/audio) – Creates more varied versions of existing data to help the model learn better general patterns.

Underfitting happens when a model is too simple and fails to learn the real pattern in the data.

Characteristics

- High bias, low variance
- Poor performance on training data
- Poor performance on test data
- Model is not powerful enough

ML Example

Model becomes too simple → underfits.

Example

Using a straight line (linear model) to fit data that clearly follows a curve.

Example: Predicting house prices using only square-feet and ignoring:

- location
- number of rooms
- age of house
- demand

Model becomes too simple → underfits.

To Reduce Underfitting

1. Increase model complexity – Allows the model to learn more detailed and complex patterns in the data.
2. Use deeper neural networks – Adds more layers so the model can capture higher-level relationships.
3. Use polynomial features – Helps the model fit curved or non-linear patterns that a straight line can't capture.
4. Train for more epochs – Gives the model more time to learn patterns that it initially missed.
5. Add more meaningful features – Provides the model with additional useful information to better understand the data.
6. Reduce regularization strength – Prevents the model from being oversimplified by strong regularization.
7. Improve data quality and representation – Better, cleaner data helps the model learn accurate relationships.

Overfitting happens when a model learns the training data too well, including:

- noise
- mistakes
- random patterns

It memorizes the data instead of learning the true pattern.

Characteristics

- Low bias, high variance
- Very high accuracy on training data
- Poor accuracy on test or new data
- Model is too complex

Example

A neural network with too many layers and parameters memorizes every training image.

Another example:

Using a 20-degree polynomial to fit 10 data points → curve becomes very wiggly.

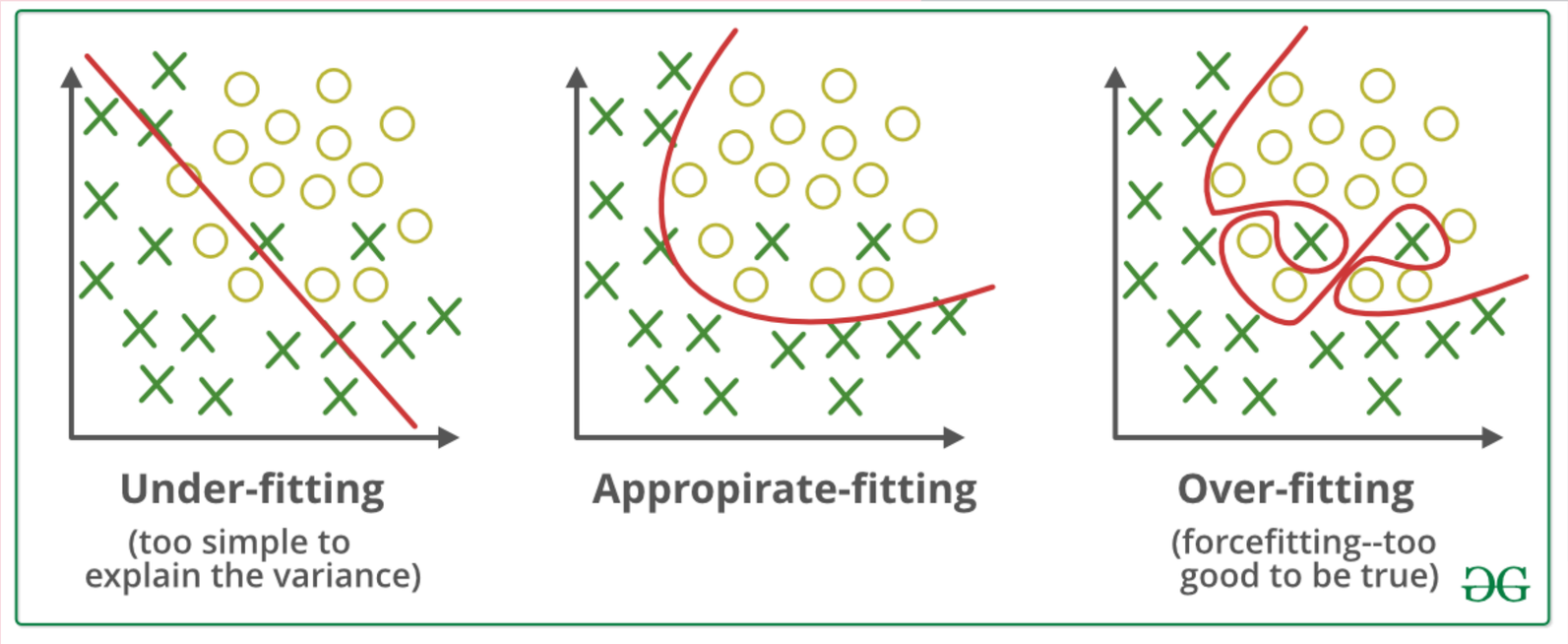
Model performs:

- excellent on training
- terrible on new data

This is overfitting.

To Reduce Overfitting

- Regularization (L1/L2) – Adds a penalty to large weights to prevent the model from becoming too complex.
- Dropout – Randomly ignores some neurons during training to make the model less dependent on specific paths.
- Use more training data – Helps the model learn true patterns instead of memorizing noise.
- Simplify the model (reduce layers/parameters) – Reduces complexity so the model focuses on essential patterns.
- Early stopping – Stops training when performance on validation data stops improving, preventing memorization.



Linear Regression

Linear Regression is a supervised machine learning algorithm used to predict a continuous numerical value based on one or more input features.

- The model assumes a linear relationship between input(s) and output.
- Formula (Simple Linear Regression with one feature):

$$y=b0+b1x$$

Where:

- y = predicted value (dependent variable)
- x = input feature (independent variable)
- b0 = intercept (value of y when x = 0)
- b1= slope (how much y changes for 1 unit change in x)

For Multiple Linear Regression (more than one feature):

$$y=b0+b1x1+b2x2+...+bnxn$$

The table below shows the number of grams of carbohydrates, X and the number of Calories, Y of six different foods. Find linear regression equation for this dataset. **[8]**

Carbohydrates (X)	8	9.5	10	6	7	4
Calories (Y)	112	138	147	88	108	62

Also find the value of Y for X = 12

Goal:
Find the linear equation $Y = b_0 + b_1X$ and predict Calories for X = 12.

Step 1: Averages

$\bar{X} = 7.4167$

$\bar{Y} = 109.167$

Step 2: Differences & Products

X	Y	$X - \bar{X}$	$Y - \bar{Y}$	$(X - \bar{X})(Y - \bar{Y})$	$(X - \bar{X})^2$
8	112	0.5833	2.833	1.652	0.34
9.5	138	2.0833	28.833	60.062	4.34
10	147	2.5833	37.833	97.684	6.676
6	88	-1.4167	-21.167	29.993	2.007
7	108	-0.4167	-1.167	0.486	0.174
4	62	-3.4167	-47.167	161.02	11.674

$$b_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}, \qquad b_0 = \bar{y} - b_1\bar{x}$$

Step 3: Summations

$\Sigma(X - \bar{X})(Y - \bar{Y}) = 350.897$

$\Sigma(X - \bar{X})^2 = 25.211$

For X = 12:

$Y = 5.947 + 13.91 \times 12$

$Y \approx 172.867$

Predicted Calories: ≈ 173

Find the Equation of linear Regression line using following data : **[6]**

X	Y
1	3
2	4
3	5
4	7

Data points:

(1, 3), (2, 4), (3, 5), (4, 7)

We want the best-fit line y=b0+b1x .

Use the usual formulas:

$$\bar{x} = \frac{1+2+3+4}{4} = 2.5, \qquad \bar{y} = \frac{3+4+5+7}{4} = 4.75$$

$$b_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}, \qquad b_0 = \bar{y} - b_1\bar{x}$$

x	y	$x - \bar{x}$	$y - \bar{y}$	$(x - \bar{x})(y - \bar{y})$	$(x - \bar{x})^2$
1	3	-1.5	-1.75	2.625	2.25
2	4	-0.5	-0.75	0.375	0.25
3	5	0.5	0.25	0.125	0.25
4	7	1.5	2.25	3.375	2.25
Sum				**6.50**	**5.00**

$$b_1 = \frac{6.50}{5.00} = 1.30$$

$$b_0 = \bar{y} - b_1\bar{x} = 4.75 - 1.30 \times 2.5 = 4.75 - 3.25 = 1.50$$

$$\boxed{y = 1.50 + 1.30\,x}$$

With following data of shows company expenditure.

x(month)	1	2	3	4	5
y(expenditure)	12	19	29	37	45

using regression model predict expenditure of 6th month.

Try it on your own and share your answer in the comments!

Lasso Reggression

- Lasso (Least Absolute Shrinkage and Selection Operator) is a linear regression technique that uses L1 regularization.
- It adds a penalty equal to the absolute value of coefficients to the loss function.
- Normal Linear Regression Loss Function

$$\text{Loss} = \text{SSE} = \sum (y - \hat{y})^2$$

- Lasso Regression Loss Function

$$\text{Loss} = \sum (y - \hat{y})^2 + \lambda \sum |w|$$

- ✓ SSE = Sum of Squared Errors
- ✓ λ\lambdaλ = Regularization strength
- ✓ ∣w∣|w|∣w∣ = absolute value of model coefficients

Why do we use Lasso Regression?

Because Lasso can:

- Reduce overfitting : By shrinking coefficients.
- Perform feature selection (unique!) : Lasso can make coefficients exactly zero, removing irrelevant features automatically.
- Handle high-dimensional data : Very useful when you have many features but only some are important.

Example: Predicting student marks

You have data about:

Feature	Useful?
Hours studied	✓ Yes
Sleep hours	✓ Ok
Number of pens	✗ No
Bag weight	✗ No
Favourite chocolate	✗ No

A normal regression model may give small weights to useless features.

But Lasso Regression will shrink them to 0:

Marks =6(Hours studied)+1(Sleep)+0(Pens)+0(Bag weight)+0(Chocolate)

Ridge Regression

Ridge Regression is a linear regression technique that reduces overfitting by adding a penalty to large coefficients.

It does NOT remove features.

Instead, it shrinks the coefficients (makes them smaller).

Formula Penalty:

$$\text{Loss} = \text{MSE} + \lambda \sum w^2$$

- (lambda) = regularization strength
- Large λ → coefficients shrink more
- Useful when dataset has:
  - Multicollinearity (features correlated)
  - Many features
  - Overfitting

Predict House Price

- Features:
- Size (sq ft)
- Number of rooms
- Distance to city
- Age of house
- Color of walls (NOT useful)

What Ridge does:
It keeps all features, even useless ones.
But reduces coefficient size so the model becomes stable.

Example trained coefficients:
Feature Coefficient
Size 38
Rooms 20
Distance -9
Age -3
Color of walls 0.5 (not removed but very small)
✓ Lasso sets useless feature to zero
✓ Ridge keeps it but shrinks it a LOT
Final Ridge Regression Model:
y =38(Size)+20(Rooms)−9(Distance)−3(Age)+0.5(Color)

Gradient Descent Algorithm

Gradient Descent is an optimization algorithm used in ML to find the best parameters of a model (like slope & intercept in linear regression).
It works by minimizing the cost function step by step.

In linear regression, we minimize:

$$J(w,b) = \frac{1}{2m}\sum(y-\hat{y})^2$$

Gradient Descent updates parameters:

$$w = w - \alpha\frac{\partial J}{\partial w}$$

$$b = b - \alpha\frac{\partial J}{\partial b}$$

Where:

- **α (alpha)** = learning rate
- **w** = weight (slope)
- **b** = bias (intercept)

Steps of Gradient Descent

- Start with random values of w and b
- Predict Y using linear regression
- Compute cost (error)
- Compute gradients (direction of steepest increase)
- Update parameters in the opposite direction (downhill)
- Repeat until convergence (cost becomes minimum)

Example: Learning cooking

If you make a dish too salty:

- You reduce salt in the next trial

If it's too bland:

- You increase salt

This is gradient descent:

- Compare error
- Adjust
- Improve
- Repeat

MAE (Mean Absolute Error)

MAE measures the average magnitude of errors in predictions, without considering their direction. It's the average of the absolute differences between predicted and actual values.

Formula:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Where:

- yi = actual value
- y^i = predicted value
- n = number of data points

Intuition:

Lower MAE → better model.

Gives equal weight to all errors.

Example:

- Actual: [3, 5, 2]
- Predicted: [2, 5, 4]
- MAE=3 ∣ 3−2 ∣ + ∣ 5−5 ∣ + ∣ 2−4 ∣ =31+0+2=1

RMSE (Root Mean Squared Error)

RMSE measures the square root of the average of squared errors. It penalizes larger errors more than MAE.

Formula:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Sensitive to large errors (outliers).

- Lower RMSE → better model.
- If errors are normally distributed, RMSE gives a good estimate of the standard deviation of errors.

Example:

Using the same data:

$$\text{RMSE} = \sqrt{\frac{(3-2)^2 + (5-5)^2 + (2-4)^2}{3}} = \sqrt{\frac{1+0+4}{3}} = \sqrt{\frac{5}{3}} \approx 1.29$$

R² (R-Squared / Coefficient of Determination)

R² measures how well the model explains the variability of the data. It ranges from 0 to 1 (sometimes negative if the model is worse than just predicting the mean).

Formula:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Intuition:

- R² = 1 → perfect prediction
- R² = 0 → model predicts no better than mean
- Higher R² → better model fit

Example:

Actual: [3, 5, 2]

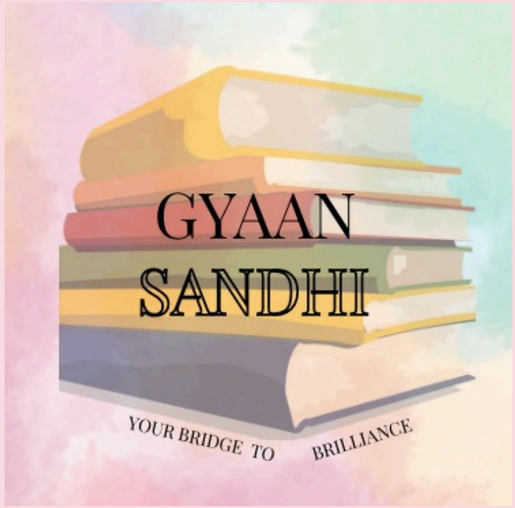
Predicted: [2, 5, 4]

Mean: $\bar{y}=3+5+23=3.33$

$$SS_{\text{res}} = (3-2)^2 + (5-5)^2 + (2-4)^2 = 1 + 0 + 4 = 5$$

$$SS_{\text{tot}} = (3-3.33)^2 + (5-3.33)^2 + (2-3.33)^2 \approx 0.11 + 2.78 + 1.77 = 4.66$$

$$R^2 = 1 - \frac{5}{4.66} \approx -0.07$$



THANK YOU

Share