Week 4 - 50.038 Computational Data Science

Advanced Regression Tasks & Temporal Dynamics

Foundations of Linear & Logistic Regression in Supervised Learning

Nizam Kadir

Framework
Applied ML with PyTorch

Overview

1 Part 1

Foundations of Linear Regression

- Introduction to predictive modeling and regression analysis
- Linear regression intuition and mathematical formulation
- Mean Squared Error (MSE) and gradient descent optimization
- California Housing case study with real data analysis
- PyTorch implementation and evaluation metrics

2 Part 2

Logistic Regression & Classification

- From regression to classification: adapting linear models
- Sigmoid function and probability estimation
- Softmax for multi-class classification problems
- Heart Disease prediction case study
- Confusion matrix and classification metrics

2	2	PyTorch
Parts	Case Studies	Implementation

Part 1

Foundations of Linear Regression

Understanding the fundamentals of supervised learning for continuous outcome prediction

SUPERVISED LEARNING FUNDAMENTALS

What is Regression?

Core Definition

Regression is a supervised learning technique used to predict continuous numerical outcomes by modeling the relationship between input features and a target variable.

Unlike classification which predicts categories, regression answers **"how much?"** questions.

Regression vs Classification

Regression

- Predicts continuous values
- Output: Real numbers
- Example: Predict house price ($350,000)

Classification

- Predicts discrete categories
- Output: Classes/labels
- Example: Classify as spam/not spam

Real-World Applications

Real Estate

Predict house prices based on size, location, age, and amenities

Business Analytics

Forecast sales, revenue, and customer demand trends

Weather & Climate

Predict temperature, rainfall, and energy consumption

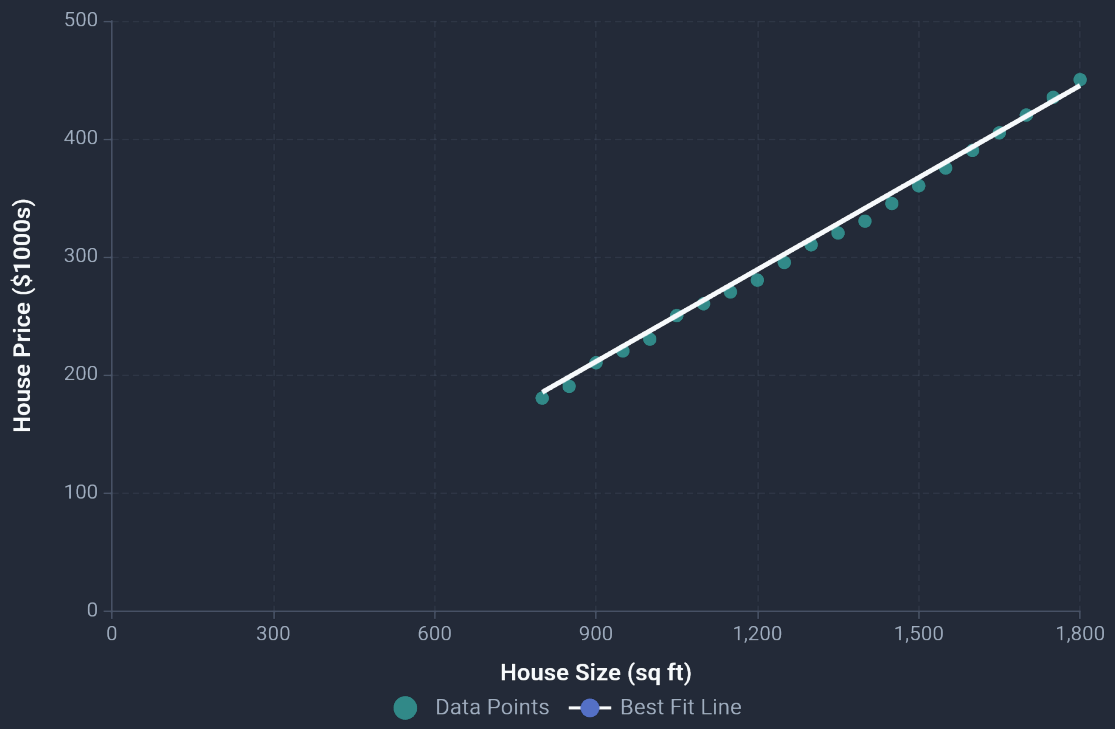
Healthcare

Estimate disease progression and drug dosage requirements

CORE CONCEPT

The Linear Regression Intuition

Visualizing the Concept



Core Idea

Linear regression fits a straight line (or hyperplane in multiple dimensions) through data points to best approximate the relationship between features and target.

The model learns **weights (w)** and a **bias (b)** that minimize prediction error across all data points.

Mathematical Model

$$y = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b$$

$$\hat{y} = w \cdot x + b \text{ (Vector Form)}$$

- y: Predicted value (target)
- $x_1 \ldots x_n$: Input features
- $w_1 \ldots w_n$: Weights (learned parameters)
- b: Bias (intercept term)

OPTIMIZATION GOAL

Finding the Best Fit

What Does "Best Fit" Mean?

The best fit line minimizes the overall prediction error across all data points. This error is measured by the vertical distance between actual values and predicted values.

$$y_i - \hat{y}_i = e_i$$

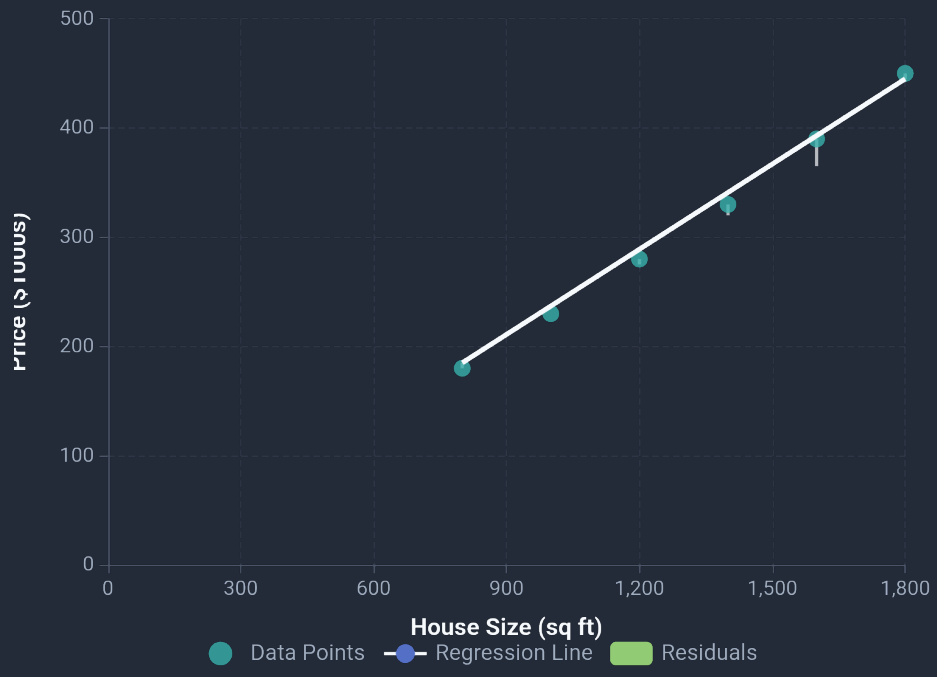
Actual − Predicted = Error

Residuals Explained

Residuals are the **vertical distances** from data points to the regression line. They represent what the model couldn't explain.

- **Positive residual:** Point above line (underpredicted)
- **Negative residual:** Point below line (overpredicted)
- **Zero residual:** Point on line (perfect prediction)

Visualizing Residuals



Goal: Find the line that makes the **sum of squared residuals** as small as possible.

COST FUNCTION

Mean Squared Error (MSE)

What is MSE?

Mean Squared Error (MSE) is the most common loss function for linear regression. It measures the **average of squared differences** between predicted and actual values.

The goal of training is to **minimize MSE** by finding optimal weights and bias.

Mathematical Formula

$$MSE = (1/N)\ \Sigma(y_i - \hat{y}_i)^2$$

Average of Squared Errors

N: Number of data points

y_i: Actual value for point i

$\hat{y}_i$: Predicted value for point i

Why Square the Errors?

Eliminates Negative Errors

Squaring ensures all errors are positive, preventing cancellation of positive and negative residuals.

Penalizes Large Errors

Squaring amplifies large errors more than small ones, pushing the model to minimize outliers.

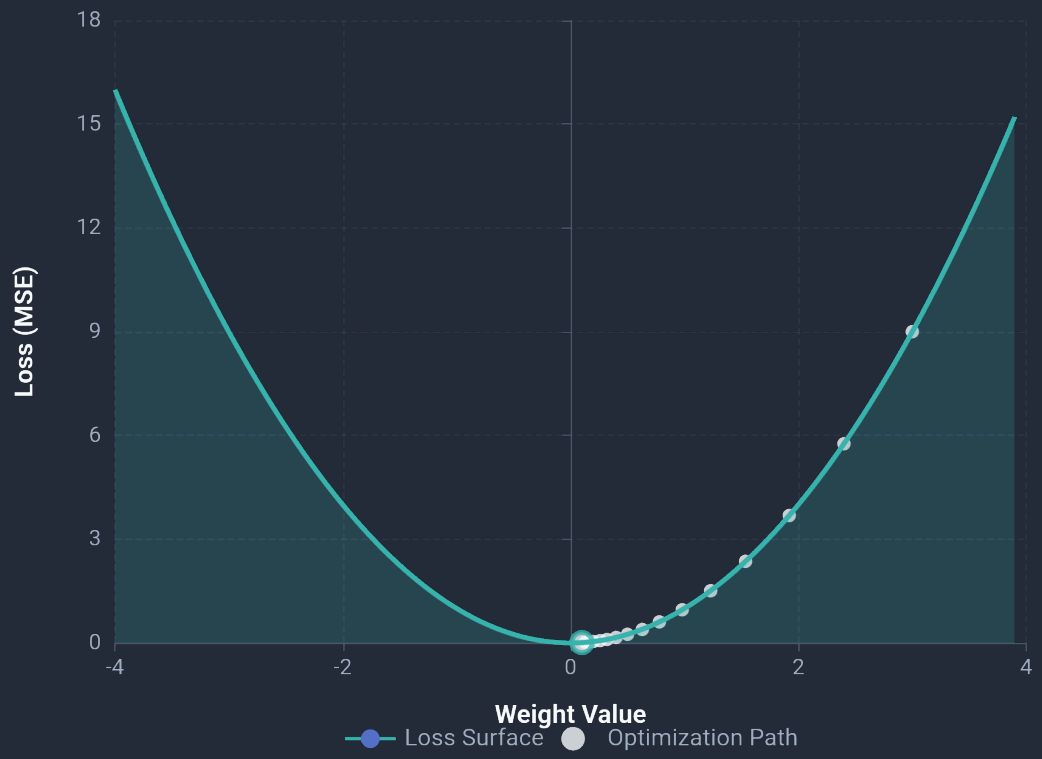
Differentiable & Smooth

The squared function is smooth and differentiable everywhere, making it suitable for gradient-based optimization.

Key Insight: Minimizing MSE finds the optimal balance between underfitting and overfitting.

OPTIMIZATION ALGORITHM

Gradient Descent: Walking Downhill



The Hiking Analogy

Imagine hiking down a mountain in thick fog. You can't see the bottom, but you can feel the slope beneath your feet.

At each step, you move in the **steepest downward direction** until you reach the valley.

How It Works

1. Start with random weights
2. Calculate the gradient (slope)
3. Move in opposite direction
4. Repeat until convergence

The gradient points in the direction of **steepest increase** — we move opposite to minimize loss.

OPTIMIZATION PARAMETER

The Learning Rate

Too Large: Overshooting

Large learning rate takes big steps, **overshooting the minimum** . The algorithm bounces around or diverges, never converging to the optimal solution.

Learning Rate: **0.5**

Too Small: Slow Convergence

Small learning rate takes tiny steps, **converging very slowly** . It may get stuck in local minima or take too long to reach the optimal solution.

Learning Rate: **0.001**

Just Right: Efficient Convergence

Optimal learning rate takes **appropriately sized steps** , converging smoothly and efficiently to the global minimum without overshooting.

Learning Rate: **0.01**

Modern Optimizers

Advanced optimizers like **Adam, RMSprop, and AdaGrad** automatically adjust the learning rate during training, often converging faster than standard SGD.

SGD **Baseline**

Adam **Adaptive**

Practical Tip: Start with learning rates between 0.01 and 0.0001. Use learning rate scheduling or adaptive optimizers for best results.

MODEL VALIDITY

Linear Regression Assumptions

1 Linearity

The relationship between features and target must be linear. The dependent variable should change proportionally with independent variables.

⚠ If violated: model underfits. Consider transformations or non-linear models.

2 Homoscedasticity

Residuals must have constant variance across all levels of independent variables. The spread of errors should be uniform.

⚠ If violated: confidence intervals unreliable. Use weighted regression or transformations.

3 Multivariate Normality

Residuals should follow a normal distribution. This enables valid hypothesis testing and confidence intervals for coefficients.

⚠ Check with Q-Q plots or histograms. Transformations may help.

4 No Multicollinearity

Independent variables should not be highly correlated (correlation < 0.8). High correlation makes coefficient estimates unstable.

⚠ Check VIF (>10 is problematic). Remove correlated features or use regularization.

CASE STUDY

California Housing Dataset

Dataset Overview

The California Housing dataset contains **20,640 examples** from the 1990 California census, representing block groups (600–3,000 people each).

20,640	8
Examples	Features

The Target Variable

Median House Value

In hundreds of thousands of dollars

The goal is to predict the median house value in each block group based on the 8 features. Values are **capped at $500,000** , which may affect model performance for high-value properties.

Feature Descriptions

1. **MedInc:** Median income (tens of thousands)
2. **HouseAge:** Median house age
3. **AveRooms:** Average rooms per house
4. **AveBedrms:** Average bedrooms per house
5. **Population:** Block group population
6. **AveOccup:** Average household size
7. **Latitude:** Geographic coordinate
8. **Longitude:** Geographic coordinate

INTERACTIVE DISCUSSION

Which Factors Drive House Prices?

Think-Pair-Share Activity

Based on your intuition and domain knowledge about real estate, which features do you expect to be **most important** in determining house prices?

Consider These Factors:

- **Median Income:** Higher income areas can afford more expensive homes
- **Number of Rooms:** More rooms typically mean larger, more valuable houses
- **Location (Lat/Long):** Coastal California areas tend to be more expensive
- **Population Density:** Crowded areas might be less desirable
- **House Age:** Newer homes might command premium prices

Discussion: Share your predictions with a partner and justify your reasoning.

Expected Relationships

↑ Positive Correlation

Income ↑, Price ↑

↑ Positive Correlation

Rooms ↑, Price ↑

↓ Negative Correlation

Population ↑, Price ↓

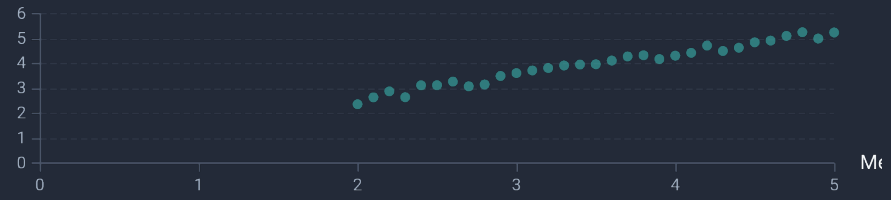
Connection to Domain Knowledge

Linear regression captures exactly these types of relationships. The model will learn **positive coefficients** for features that increase price, and **negative coefficients** for features that decrease price.

DATA EXPLORATION

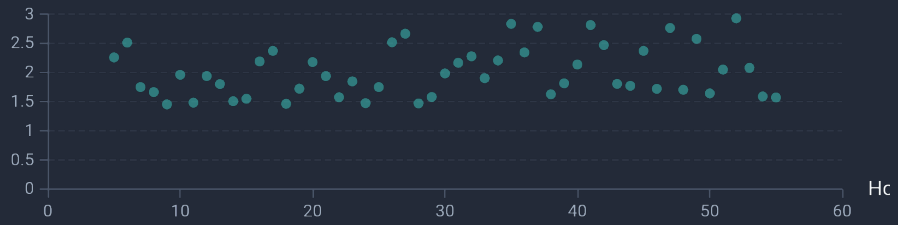
Feature-Target Relationships

Median Income vs Price



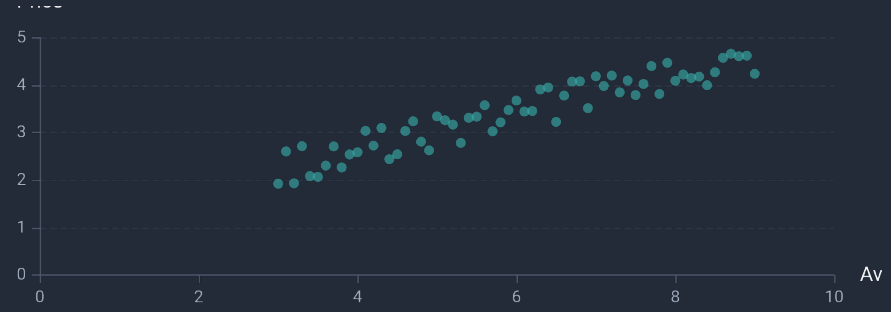
Observation: Clear positive linear trend — higher income areas have more expensive homes.

House Age vs Price



Observation: Weaker relationship — age alone doesn't strongly predict price.

Average Rooms vs Price



Observation: Strong positive relationship — more rooms means higher price.

Linear Assumption Check

Linear regression assumes a **straight-line relationship** between features and target.

- ✓ Income vs Price: **Linear ✓**
- ✓ Rooms vs Price: **Linear ✓**
- ? Age vs Price: **Weak relationship**

UNDERSTANDING THE MODEL

Model Interpretation

Interpreting Coefficients

In linear regression, **coefficients (weights) tell us how much the target variable changes** for a one-unit increase in each feature, **holding all other features constant** .

Example Interpretation

If the coefficient for "number of rooms" is **20,000** , then each additional room adds **$20,000** to the predicted price (assuming other features remain the same).

Positive Weight
Increasing the feature increases predicted price

Negative Weight
Increasing the feature decreases predicted price

Magnitude Matters

The **absolute value of the coefficient** indicates the feature's importance:

Weight = 50,000
Strong Impact

Weight = 5,000
Weak Impact

Example: California Housing Coefficients

Median Income	+65,000
Ave Rooms	+20,000
Ave Bedrms	+5,000
House Age	-200
Latitude	-3,000

Key Insight: Income and rooms are the strongest positive predictors.

MODEL CONSTRAINTS

Linear Regression Limitations

Linear Relationships Only

Linear regression cannot capture non-linear patterns. If the true relationship is curved, the model will underfit and produce systematic errors.

Solution: Use polynomial features, splines, or non-linear models like decision trees.

Sensitive to Outliers

A single extreme outlier can significantly skew the regression line, pulling it toward the outlier and affecting coefficient estimates.

Solution: Detect and handle outliers using robust regression (RANSAC, Huber).

Multicollinearity Issues

Highly correlated features make coefficient estimates unstable and difficult to interpret. The model can't distinguish individual feature contributions.

Solution: Remove correlated features or use regularization (Ridge/Lasso).

Assumes Feature Independence

Each feature's contribution is assumed independent, but in reality, features often interact (e.g., rooms × income may have synergistic effects).

Solution: Add interaction terms or use models that capture feature interactions.

DATA PREPROCESSING

Feature Scaling Importance

Why Scale Features?

Features with different scales cause problems for gradient-based optimization. Income (0–15) and house age (0–55) have vastly different ranges, leading to inefficient convergence.

Scaling brings all features to **similar numerical ranges**, ensuring each feature contributes appropriately.

Two Main Methods

Normalization (Min-Max)

$$x' = (x - \min) / (\max - \min)$$

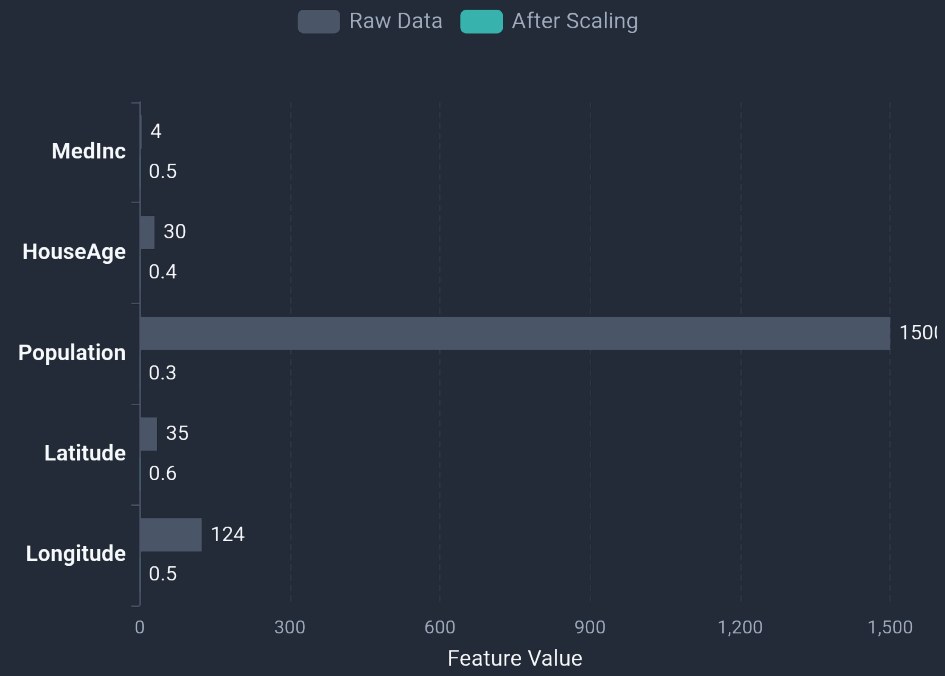
Scales to [0, 1] range. Sensitive to outliers.

Standardization (Z-Score)

$$x' = (x - \mu) / \sigma$$

Centers at 0, σ=1. Less sensitive to outliers.

Feature Scale Comparison



Raw Data: Features have vastly different scales

After Scaling: All features on similar scale ✓

IMPLEMENTATION PREVIEW

PyTorch Implementation

Model Definition

```
import torch
import torch.nn as nn
import torch.optim as optim
# Load and prepare data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
# Convert to PyTorch tensors
X_train_tensor = torch.tensor(X_train, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train, dtype=torch.float32).view(-1, 1)
# Define the model: single linear layer
model = nn.Linear(in_features=8, out_features=1)
# Loss function and optimizer
criterion = nn.MSELoss()
optimizer = optim.SGD(model.parameters(), lr=0.01)
```

Key Components

1. **nn.Linear(8, 1):** Single layer with 8 inputs, 1 output
2. **nn.MSELoss():** Mean Squared Error loss function
3. **optim.SGD:** Stochastic Gradient Descent optimizer

Training Loop

```
for epoch in range(100):
    optimizer.zero_grad()
    predictions = model(X_train)
    loss = criterion(predictions, y_train)
    loss.backward()
    optimizer.step()
```

PyTorch handles automatic differentiation and gradient computation.

TRAINING WORKFLOW

Training in Practice

1 Forward Pass

Feed training features to the model to get predictions.

$\hat{y} = w \cdot x + b$

2 Compute Loss

Compare predictions with actual values using MSE.

$\text{Loss} = \text{MSE}(y, \hat{y})$

3 Backward Pass

Compute gradients of loss with respect to weights.

$\partial \text{Loss} / \partial w$

4 Update Weights

Use optimizer to adjust weights and reduce loss.

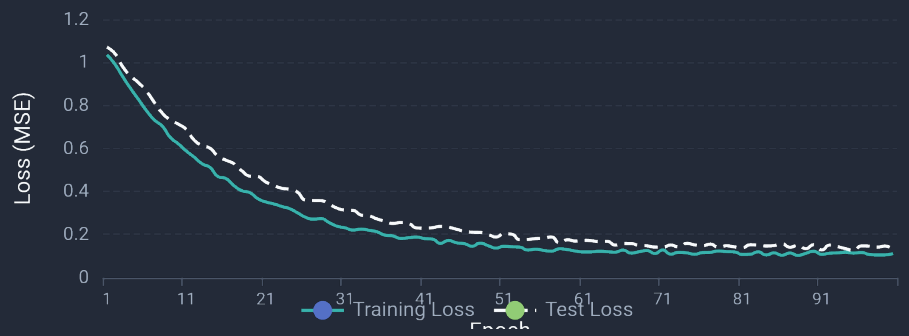
$w = w - lr \times \nabla w$

5 Repeat

Iterate for many epochs until convergence.

100–1000 epochs

Training vs Test Loss



Monitoring Training

Track both training and test loss to detect **overfitting**.

Good Training

Both losses decrease and converge.

Overfitting

Training loss ↓, test loss ↑.

MODEL PERFORMANCE

Evaluation Metrics for Regression

R^2 (Coefficient of Determination)

R^2 measures the proportion of variance in the target variable that is explained by the model. It's the most interpretable regression metric.

$$R^2 = 1 - (SS_{res} / SS_{tot})$$

Explained Variance / Total Variance

$R^2 = 1$: Perfect fit

$R^2 = 0.58$: 58% of variance explained

$R^2 = 0$: No predictive power

Interpreting R^2

$R^2 = 0.8$–1.0	Excellent model fit
$R^2 = 0.5$–0.8	Good model fit
$R^2 = 0.2$–0.5	Moderate model fit

Additional Metrics

MSE (Mean Squared Error)

Average of squared errors. Penalizes large errors heavily.

$$MSE = (1/N)\ \Sigma(y_i - \hat{y}_i)^2$$

RMSE (Root Mean Squared Error)

Square root of MSE. In same units as target variable.

$$RMSE = \sqrt{MSE}$$

MAE (Mean Absolute Error)

Average of absolute errors. Less sensitive to outliers.

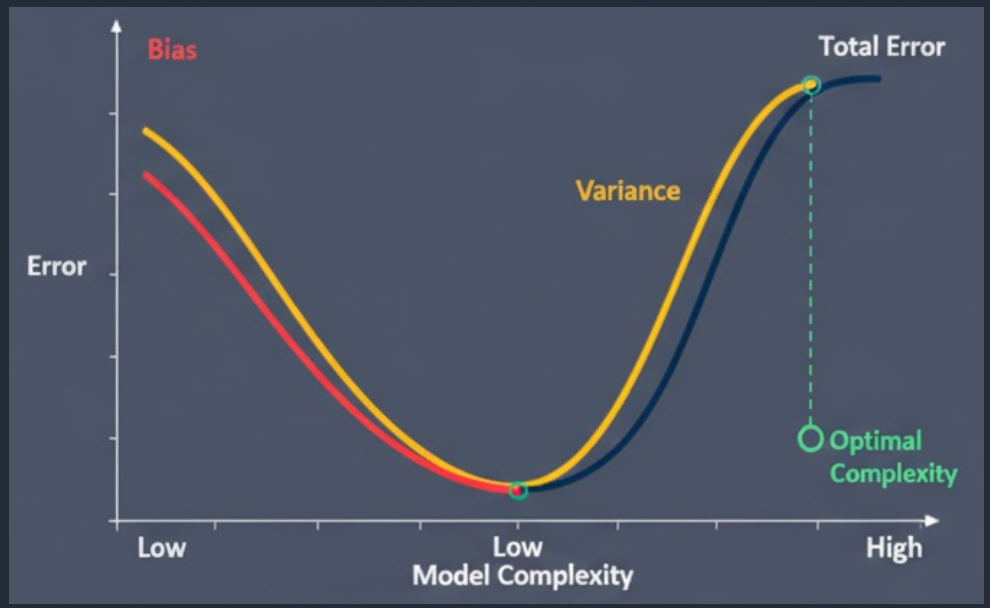
$$MAE = (1/N)\ \Sigma|y_i - \hat{y}_i|$$

Best Practice: Use multiple metrics for comprehensive evaluation.

MODEL COMPLEXITY

Overfitting vs Underfitting

The Bias-Variance Tradeoff



Underfitting

High Bias: Model too simple, fails to capture patterns.

Overfitting

High Variance: Model too complex, memorizes noise.

Symptoms & Solutions

Underfitting (High Bias)

Symptoms: High training & test error
Solutions: Add features, reduce regularization

Overfitting (High Variance)

Symptoms: Low training, high test error
Solutions: More data, regularization, simpler model

The Goal: Balance

Find the **optimal model complexity** that captures true patterns without memorizing noise.

Good Fit: Similar training and test performance.

LECTURE 1 SUMMARY

Key Takeaways

1 Linear Regression Definition

Linear regression fits a straight line through data to predict continuous outcomes. It's a foundational supervised learning technique for regression problems.

2 MSE as Loss Function

Mean Squared Error (MSE) measures average squared differences between predicted and actual values. Training aims to minimize MSE via gradient descent.

3 Gradient Descent Optimization

Gradient descent iteratively adjusts weights to reduce loss, like walking downhill. Learning rate controls step size for efficient convergence.

4 R^2 for Evaluation

R^2 (coefficient of determination) measures the proportion of variance explained by the model. R^2 = 0.6 means 60% of variance is captured.

5 California Housing Case Study

The California Housing dataset demonstrates practical application: median income and number of rooms are the strongest predictors. Feature scaling ensures efficient gradient descent convergence in PyTorch implementation.

Part 2

Logistic Regression & Classification Metrics

Adapting linear models for categorical outcome prediction and probabilistic classification

PROBLEM TRANSITION

From Regression to Classification

The Shift in Perspective

In Lecture 1, we predicted **continuous values** like house prices. Now we shift to predicting **discrete categories**.

Instead of asking **"how much?"**, we ask **"which category?"**

Why Not Linear Regression?

Linear regression is poorly suited for classification because it can output any real number, not just 0 or 1.

- ✗ Predicts values outside [0,1]
- ✗ No probabilistic interpretation
- ✗ Sensitive to outliers

Common Binary Classification Examples

Email Classification

Spam (1) vs Not Spam (0)

Medical Diagnosis

Disease Present (1) vs Absent (0)

Financial Risk

Loan Default (1) vs No Default (0)

Customer Behavior

Churn (1) vs Retained (0)

CLASSIFICATION SOLUTION

The Logistic Regression Solution

Core Innovation

Logistic regression uses the same linear combination of inputs ($z = w \cdot x + b$) but then applies a **Sigmoid function** to squash the output into the range [0, 1].

This output can be interpreted as a **probability** of belonging to the positive class.

Mathematical Formulation

Step 1: Linear Combination

$$z = w \cdot x + b$$

Step 2: Apply Sigmoid

$$p = \sigma(z) = 1 / (1 + e^{(-z)})$$

Key Advantages

Bounded Output
Output is always between 0 and 1, perfect for probabilities

Probabilistic Output
Gives confidence scores, not just hard classifications

S-shaped Curve
Smooth transition between classes with diminishing returns

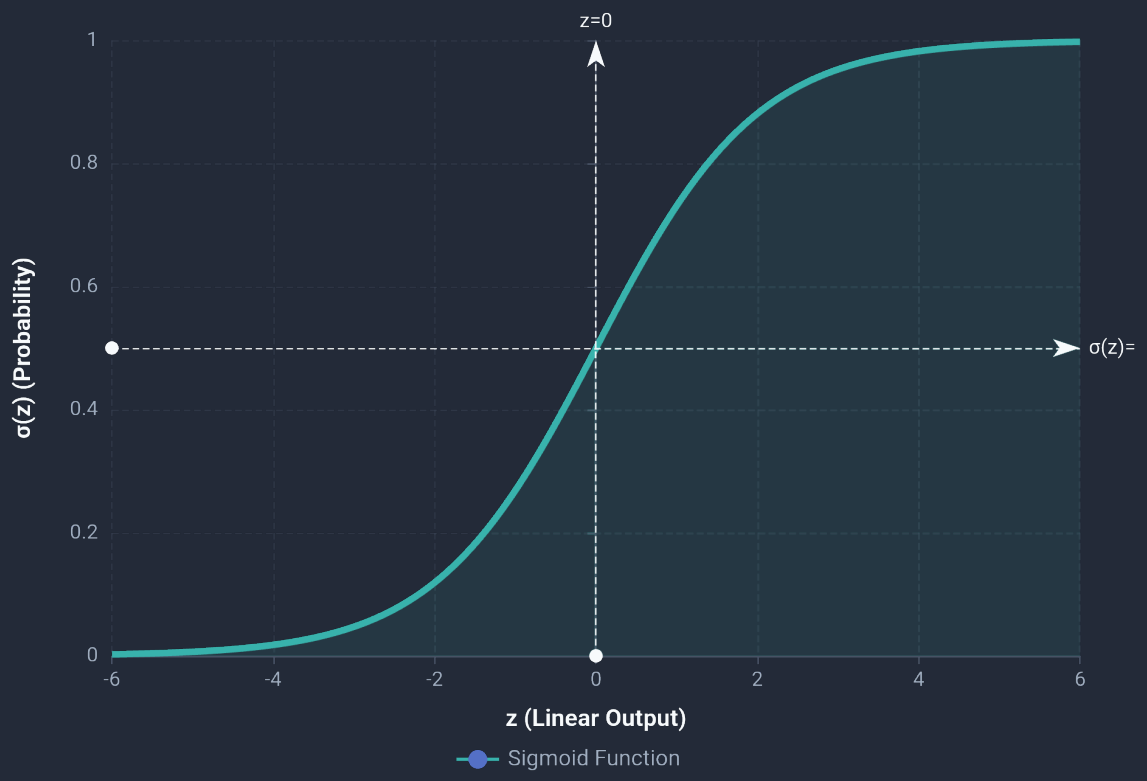
Interpretable Model
Coefficients indicate direction and strength of feature effects

Insight: Logistic regression is essentially linear regression with a probabilistic twist.

ACTIVATION FUNCTION

The Sigmoid Function

Visualizing the Sigmoid



Mathematical Definition

$$\sigma(z) = 1 / (1 + e^{(-z)})$$

Where e ≈ 2.718

Input z: Any real number (-∞ to +∞)

Output σ(z): Always between 0 and 1

Key Properties

At z = 0	$\sigma(0) = 0.5$
As z → +∞	$\sigma(z) \to 1$
As z → -∞	$\sigma(z) \to 0$

INTUITIVE UNDERSTANDING

Sigmoid Intuition

↑ Large Positive z

When the linear combination **z is large and positive**, the Sigmoid output approaches **1.0**.

The model is **very confident** this example belongs to the positive class.

↓ Large Negative z

When the linear combination **z is large and negative**, the Sigmoid output approaches **0.0**.

The model is **very confident** this example belongs to the negative class.

⚖ z Near Zero

When **z is close to zero**, the Sigmoid output is near **0.5**.

The model is **uncertain** — the example is on the decision boundary.

What the Model Learns

The model learns weights such that:

- **z > 0** for inputs likely to be class 1
- **z < 0** for inputs likely to be class 0

CLASSIFICATION DECISION

Decision Threshold

From Probability to Prediction

The Sigmoid function outputs a probability between 0 and 1. **To make a binary classification** , we need to convert this probability into a discrete prediction (0 or 1).

Default Decision Rule

If $p \geqslant 0.5 \rightarrow$ Predict Class 1

If $p < 0.5 \rightarrow$ Predict Class 0

Why 0.5 Threshold?

The 0.5 threshold represents **maximum uncertainty** in the Sigmoid function (when $z = 0$). It's the natural decision boundary:

p = 0.6

Slightly confident in class 1

p = 0.9

Very confident in class 1

Adjusting the Threshold

The 0.5 threshold can be **adjusted based on the cost of different errors** in your specific application.

Medical Diagnosis

Missing a disease (false negative) is very costly. Use a **lower threshold (e.g., 0.3)** to increase sensitivity.

If $p \geq 0.3 \rightarrow$ Predict Disease ✓

Spam Detection

False positives (marking important email as spam) are costly. Use a **higher threshold (e.g., 0.8)** to be more conservative.

If $p \geq 0.8 \rightarrow$ Predict Spam ✓

CASE STUDY

Heart Disease Dataset

Dataset Overview

The UCI Heart Disease dataset contains **303 patient records** with **13 medical attributes** plus the target variable indicating heart disease presence.

Target Variable

0	1
No Disease	Disease

Why This Dataset?

- ✓ **Binary outcome** perfect for logistic regression
- ✓ **Real medical data** with clinical relevance
- ✓ **Interpretable features** help identify risk factors
- ✓ **Balanced classes** (~54% have disease)

Feature Descriptions

1. **Age:** Patient age in years
2. **Sex:** 1 = male, 0 = female
3. **Chest Pain Type:** Type of chest pain (0-3)
4. **Resting BP:** Resting blood pressure (mm Hg)
5. **Cholesterol:** Serum cholesterol (mg/dl)
6. **Fasting Blood Sugar:** >120 mg/dl (1 = true; 0 = false)
7. **Resting ECG:** Resting ECG results (0-2)
8. **Max Heart Rate:** Maximum heart rate achieved
9. **Exercise-Induced Angina:** 1 = yes; 0 = no
10. **ST Depression:** Exercise-induced ST depression
11. **Slope of ST:** Slope of peak exercise ST segment
12. **Major Vessels:** Number (0-3) colored by fluoroscopy
13. **Thalassemia:** Blood disorder (0-3)

MODEL SELECTION

Why Logistic Regression for Medical Data?

Probabilistic Output

Outputs probability of heart disease (e.g., 85% risk), allowing doctors to assess risk levels rather than just binary predictions.

Interpretable Model

Coefficients show which factors increase/decrease risk. Positive weight on cholesterol means higher cholesterol increases heart disease probability.

Clinically Validated

Widely used and validated in medical research. Simple, robust, and well-understood by healthcare professionals.

Comparison with Other Models

Logistic Regression	✓ Interpretable
Neural Networks	- Black box
Random Forest	- Less interpretable

Clinical Applications

- Early disease detection
- Patient risk stratification
- Doctor-patient communication
- Healthcare resource allocation

DOMAIN KNOWLEDGE

Feature Insights

Expected Risk Factors

↑ Higher Cholesterol

Elevated cholesterol is a well-known cardiovascular risk factor. We expect a **positive coefficient**.

↑ Older Age

Heart disease risk increases with age. Age should have a **positive coefficient**.

↑ Exercise-Induced Angina

Chest pain during exercise indicates heart problems. Expect **positive coefficient**.

Protective Factors

↓ Higher Max Heart Rate

Achieving higher heart rate during exercise indicates better cardiac fitness. Expect **negative coefficient**.

↓ Normal Blood Pressure

Lower resting blood pressure is healthier. Should have **negative coefficient**.

↓ No Chest Pain

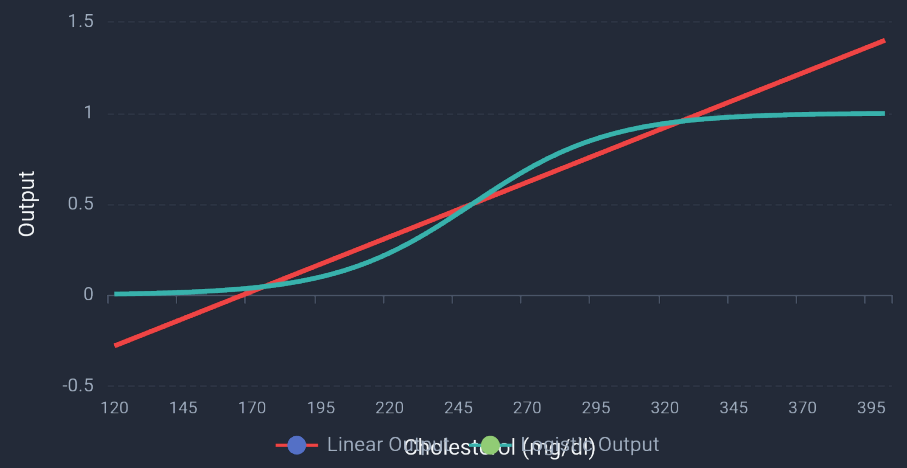
Absence of chest pain (cp=0) is protective. Should have **negative coefficient**.

Discussion: Which features do you think will be most predictive?

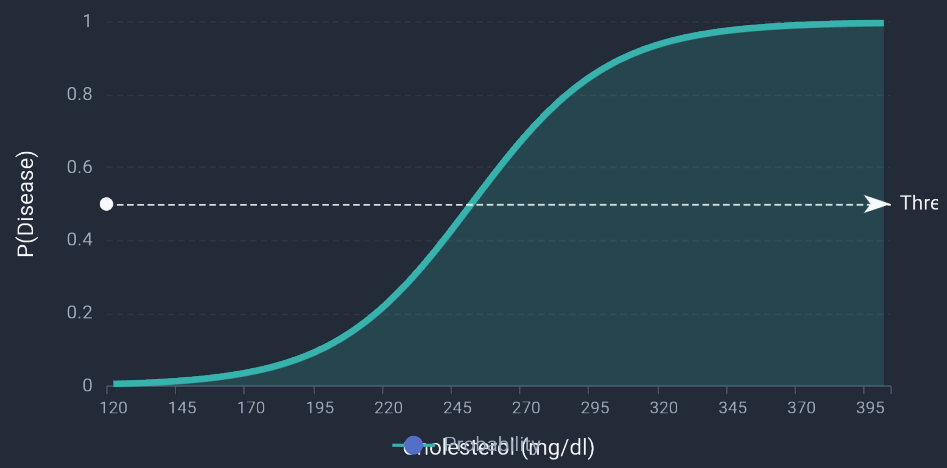
MODEL VISUALIZATION

Visualizing Logistic Regression

Linear vs Logistic Output



Probability vs Cholesterol



Linear Regression Issues

✗ Predicts values < 0 or > 1

✗ No probabilistic meaning

✗ Straight line doesn't fit

Logistic Solution

✓ Output always in [0,1]

✓ Clear probability interpretation

✓ S-shaped curve fits data

Key Insight

The Sigmoid function creates a **smooth transition** between classes, with diminishing returns at the extremes.

LOSS FUNCTION

Binary Cross-Entropy Loss

Why Not MSE for Classification?

MSE is poorly suited for classification because it **penalizes confident correct predictions** and doesn't work well with Sigmoid's non-linear output.

Instead, we use **binary cross-entropy** (log loss), which heavily penalizes **confident wrong predictions**.

Mathematical Formula

$$\text{Loss} = -[y \cdot \log(p) + (1-y) \cdot \log(1-p)]$$

Binary Cross-Entropy (Log Loss)

y: True label (0 or 1)

p: Predicted probability

Loss Examples

Good Prediction — Low Loss

True: 1, Predicted: 0.9 → Loss ≈ 0.1

Bad Prediction — High Loss

True: 1, Predicted: 0.1 → Loss ≈ 2.3

Confident Wrong — Very High Loss

True: 1, Predicted: 0.01 → Loss ≈ 4.6

Key Insight: Log loss punishes overconfident wrong predictions much more severely.

TRAINING PROCESS

Training Logistic Regression

PyTorch Implementation

```
import torch.nn as nn
import torch.optim as optim
# Define model: linear layer (no Sigmoid in model)
model = nn.Linear(in_features=13, out_features=1)
# Loss function: combines Sigmoid + BCE
criterion = nn.BCEWithLogitsLoss()  # Efficient!# Optimizer: Adam often works better than SGD
optimizer = optim.Adam(model.parameters(), lr=0.01)
# Training loopfor epoch in range(100):
    optimizer.zero_grad()
    logits = model(X_train)          # Raw scores
    loss = criterion(logits, y_train) # BCE with logits
    loss.backward()
    optimizer.step()
```

Training Parallels

The training process is **nearly identical to linear regression** with a few key differences:

1. Same gradient descent optimization
2. Different loss function (BCE vs MSE)
3. Sigmoid applied during evaluation

Key Differences

Loss Function

Binary cross-entropy instead of MSE

Output Transformation

Apply Sigmoid after training for probabilities

MULTI-CLASS CLASSIFICATION

Softmax for Multi-Class Classification

Beyond Binary Classification

Many real-world problems have **more than two classes**. Sigmoid only works for binary problems.

Softmax generalizes Sigmoid for multi-class classification, converting raw scores into probabilities across multiple categories.

Multi-Class Examples

- Handwritten digit recognition (0–9)
- News article classification
- Iris flower species classification

Softmax Formula & Properties

$$P(y=i|x) = e^{(z_i)} / \Sigma_j e^{(z_j)}$$

Exponentiate and normalize

Key Properties

- Outputs sum to 1: $\Sigma p_i = 1$
- Each output $\in [0, 1]$
- Preserves input ordering
- Differentiable everywhere

Example: 3 Classes

Raw scores: [2.0, 1.0, -1.0]

Softmax: [0.72, 0.26, 0.02]

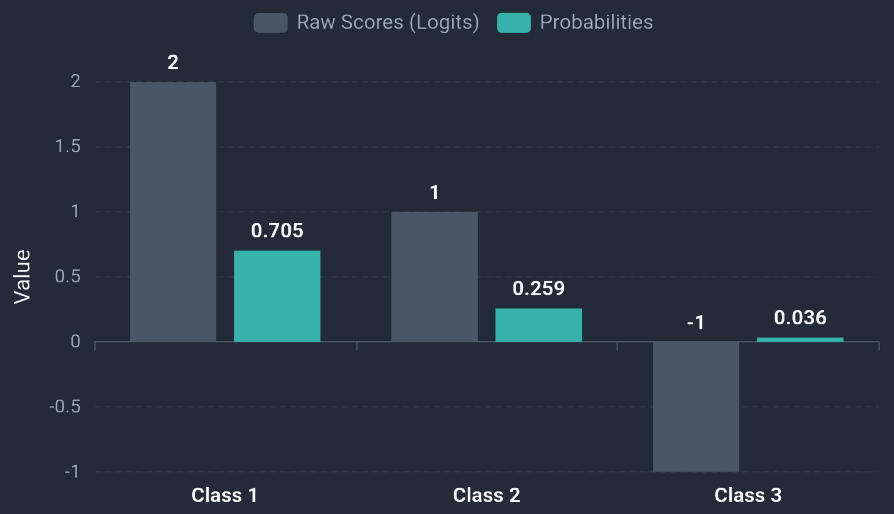
Prediction: Class 1 (72%)

Note: For 2 classes, Softmax reduces to Sigmoid.

MATHEMATICAL FOUNDATION

Softmax Formula & Properties

Step-by-Step Computation



Numerical Stability

Direct computation of softmax can cause **numerical overflow** when exponentiating large numbers.

Stable Computation

$$x' = x - \max(x)$$

Subtracting the maximum value before exponentiation keeps numbers in a safe range while preserving the result.

Comparison: Sigmoid vs Softmax

Sigmoid: Binary classification (2 classes)

Softmax: Multi-class classification (≥2 classes)

PyTorch Implementation

```
model = nn.Linear(10, 3)
criterion = nn.CrossEntropyLoss()
```

When to Use Which?

2 classes: Sigmoid or Softmax

>2 classes: Softmax only

EVALUATION METRICS

Classification Evaluation: Accuracy

What is Accuracy?

Accuracy is the fraction of correct predictions made by the model out of all predictions. It's the most intuitive classification metric.

Accuracy = Correct / Total

Fraction of correct predictions

Heart Disease Example

Correct Predictions	85
Incorrect Predictions	15

Accuracy = 85%

The Accuracy Trap

Accuracy can be **misleading with imbalanced classes**. If one class is much more frequent, high accuracy might hide poor performance on the minority class.

Dangerous Example

In a dataset with **95% healthy patients**, a model that always predicts "no disease" achieves:

95% Accuracy

But it **misses 100% of disease cases**!

⚠ **Lesson:** Always examine the confusion matrix, especially with imbalanced data.

DETAILED EVALUATION

The Confusion Matrix

2×2 Confusion Matrix

	Predicted No	Predicted Yes
Actual No	TN	FP
Actual Yes	FN	TP

True Positive (TP)
Correctly predicted disease when present

True Negative (TN)
Correctly predicted no disease when absent

False Positive (FP)
Incorrectly predicted disease when absent (Type I Error)

False Negative (FN)
Incorrectly predicted no disease when present (Type II Error)

Heart Disease Context

True Positive

Model correctly predicts heart disease when patient actually has it. Patient gets proper treatment.

True Negative

Model correctly predicts no heart disease when patient is healthy. No unnecessary treatment.

False Positive

Model predicts heart disease when patient is healthy. Causes anxiety and unnecessary tests.

False Negative

Model predicts no heart disease when patient has it. **Most dangerous error** — disease goes undetected.

WORKED EXAMPLE

Confusion Matrix Example

Scenario: 100 Patients

Suppose we have **100 patients**: 40 actually have heart disease, 60 do not. Our logistic regression model makes the following predictions:

30 TP (Correct Disease)	**10** FN (Missed Disease)
5 FP (False Alarm)	**55** TN (Correct No Disease)

Calculating Accuracy

$$\text{Accuracy} = (TP + TN) / \text{Total}$$

$$= (30 + 55) / 100 = 85\%$$

The model achieves **85% accuracy**, but more importantly, we can see the types of errors it makes.

Understanding the Errors

False Positives: 5 patients

Healthy patients incorrectly flagged as having heart disease. Causes unnecessary anxiety and additional testing.

False Negatives: 10 patients

Patients with heart disease that the model missed. **Much more dangerous** — disease goes undetected and untreated.

Clinical Insight: In medical diagnosis, we often want to minimize false negatives even at the cost of more false positives.

ADVANCED METRICS

Beyond Accuracy: Precision & Recall

Precision

Precision measures how accurate positive predictions are. Of all patients predicted to have heart disease, what fraction actually have it?

$$\text{Precision} = TP / (TP + FP)$$

Accuracy of positive predictions

High Precision: When model predicts disease, it's usually right (few false alarms)

Recall (Sensitivity)

Recall measures how many actual positives were detected. Of all patients who have heart disease, what fraction did the model correctly identify?

$$\text{Recall} = TP / (TP + FN)$$

Completeness of positive detection

High Recall: Model finds most disease cases (few missed cases)

Example Calculation

Using our previous example: TP=30, FP=5, FN=10, TN=55

Precision

$$30 / (30 + 5) = 0.857$$

85.7% of disease predictions were correct

Recall

$$30 / (30 + 10) = 0.75$$

75% of actual disease cases were found

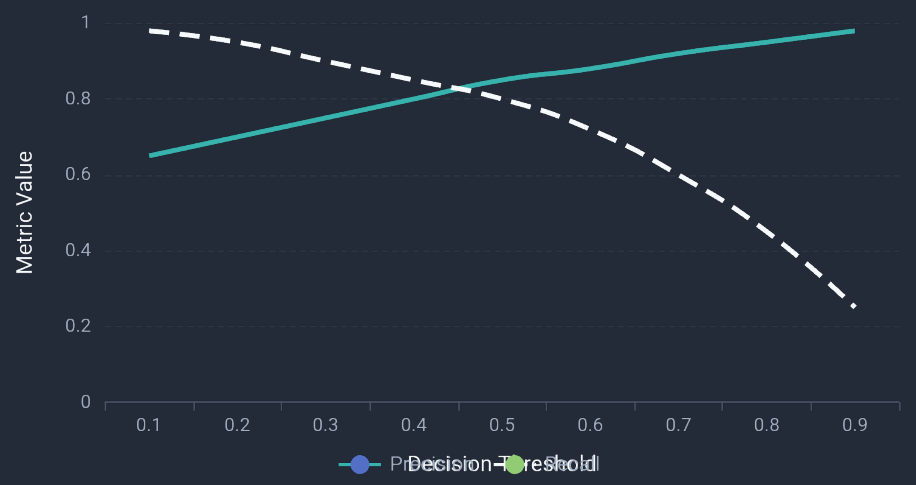
The Trade-off

Lowering the decision threshold **increases recall** (fewer missed cases) but **decreases precision** (more false alarms). Choose based on the cost of each error type.

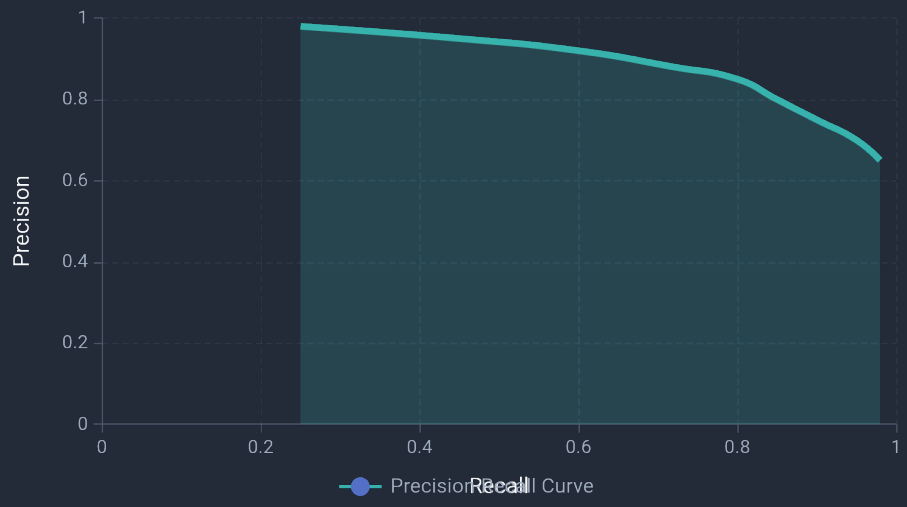
THRESHOLD OPTIMIZATION

Adjusting Decision Threshold

Threshold Effects



Precision-Recall Trade-off



Low Threshold (0.3)

✓**High Recall**: Few missed diseases

✗**Low Precision**: More false alarms

Use when **missing disease is very costly**

Default Threshold (0.5)

-**Balanced** Precision & Recall

- Good starting point

Use when **both errors are equally costly**

High Threshold (0.8)

✓**High Precision**: Few false alarms

✗**Low Recall**: More missed cases

Use when **false alarms are very costly**

SIDE-BY-SIDE COMPARISON

Linear vs Logistic Regression

Complete Comparison Table

Aspect	Linear Regression	Logistic Regression
Problem Type	Regression (continuous outcomes)	Classification (categorical outcomes)
Output	Real numbers (e.g., price: $350,000)	Probabilities (0-1), then classes
Activation Function	None (linear output)	Sigmoid (binary) or Softmax (multi-class)
Loss Function	Mean Squared Error (MSE)	Binary Cross-Entropy (Log Loss)
Primary Metrics	R^2, MSE, RMSE, MAE	Accuracy, Precision, Recall, F1, AUC
Output Interpretation	Direct prediction of target value	Probability, then class via threshold
Use Cases	Price prediction, sales forecasting	Spam detection, medical diagnosis
Similarities	Both use gradient descent, learn weights & bias, assume linear relationships, are baseline models	

SIDE-BY-SIDE COMPARISON

Linear vs Logistic Regression

Key Differences

- Output type and range
- Loss function formulation
- Evaluation metrics
- Activation function usage

Shared Foundation

- Linear combination of features
- Gradient-based optimization
- Weight and bias parameters
- Supervised learning approach

DECISION FRAMEWORK

When to Use Which?

Use Linear Regression When:

1 Predicting Continuous Values

Target variable is numerical and continuous: house prices, stock prices, temperature, sales revenue

2 Linear Relationship Exists

Features have approximately linear relationship with target. Check with scatter plots.

3 Need Interpretable Coefficients

Want to understand feature impact: "Each room adds $20k to house price"

Use Logistic Regression When:

1 Predicting Categories

Target variable is categorical with 2 or more classes: spam/not spam, disease/no disease

2 Need Probabilistic Output

Want confidence scores: "85% chance this patient has heart disease"

3 Interpretable Classification

Need to understand which features drive classification decisions

Best Practice: Start with the appropriate regression type based on your problem, then try more complex models if needed. Both linear and logistic regression serve as excellent baseline models.

INDUSTRY APPLICATIONS

Real-World Applications

Linear Regression Applications

Real Estate

Predict house prices, rental values, property valuations based on size, location, amenities

Finance & Economics

Stock price prediction, risk assessment, economic forecasting, portfolio optimization

Marketing & Sales

Sales forecasting, customer lifetime value, demand prediction, pricing optimization

Environmental Science

Temperature forecasting, pollution level prediction, climate modeling

Logistic Regression Applications

Email & Security

Spam detection, fraud detection, intrusion detection, malware classification

Healthcare

Disease diagnosis, patient risk assessment, treatment outcome prediction

Financial Services

Loan approval, credit scoring, insurance risk assessment, fraud detection

Customer Analytics

Customer churn prediction, conversion prediction, satisfaction classification

Industry Impact: These models power billions of predictions daily across finance, healthcare, technology, and countless other sectors. Understanding them opens doors to diverse career opportunities.

MODEL IMPROVEMENT

Feature Engineering Matters

Why Feature Engineering?

The features you create often matter more than the algorithm you choose. Raw data rarely contains the optimal representation for prediction.

Domain knowledge helps create features that capture underlying patterns that linear models alone cannot discover.

Common Techniques

Polynomial Features

Add x^2, x^3 terms to capture non-linear relationships. E.g., rooms^2 for diminishing returns.

Interaction Terms

Multiply features: income × rooms captures synergistic effects.

Example: Housing Features

Original Features

- Median Income: $50,000
- Number of Rooms: 6
- House Age: 20 years

Engineered Features

- Income per Room: $8,333
- Rooms^2: 36 (diminishing returns)
- Age Category: <10yr, 10-30yr, >30yr
- Location Score: Urban=1, Suburban=0.5

Impact

Feature engineering can improve R^2 from **0.6 to 0.8+** — capturing 20% more variance with the same model!

OVERFITTING PREVENTION

Regularization: Preventing Overfitting

What is Regularization?

Regularization prevents overfitting by penalizing large weights in the loss function. It encourages simpler models that generalize better to unseen data.

The model learns to fit the data while keeping coefficients small, reducing sensitivity to noise in training data.

L1 Regularization (Lasso)

$$\text{Loss} = \text{MSE} + \lambda \cdot \Sigma|w|$$

Adds penalty proportional to absolute weights

Effect: Can drive some weights to exactly zero, performing automatic feature selection.

L2 Regularization (Ridge)

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

Adds penalty proportional to weight magnitudes

λ (lambda): Regularization strength

Effect: Keeps all weights small

Choosing λ

λ = 0
No regularization — risk of overfitting

λ = optimal
Good balance — best generalization

λ = too high
Too much regularization — underfitting

COURSE SUMMARY

Key Takeaways

1 Foundational Models

Linear and logistic regression are **foundational supervised learning models** that serve as essential building blocks for understanding more complex algorithms.

2 Problem-Specific Choice

Linear for continuous outcomes , logistic for categorical. Choosing the right tool for your problem type is crucial for success.

3 Different Metrics

Regression uses **R^2 and MSE** ; classification uses **accuracy and confusion matrix** . Understanding evaluation is key.

4 Shared Fundamentals

Both models use **gradient descent optimization** and learn weights to minimize loss, sharing core concepts that apply to neural networks.

5 Simple Yet Powerful

Despite their simplicity, both models are **powerful baseline models** widely used in industry. They offer interpretability, fast training, and often provide competitive performance. Master these before moving to complex models.

YOUR ML JOURNEY

Next Steps in Your ML Journey

Decision Trees

Learn non-linear models that capture complex patterns through recursive splitting. Foundation for Random Forests and Gradient Boosting.

Neural Networks

Extend logistic regression to multiple layers. Build deep learning models for image recognition, NLP, and complex pattern recognition.

Ensemble Methods

Combine multiple models for better performance: Random Forests, Gradient Boosting (XGBoost, LightGBM), and stacking techniques.

Foundation for Advanced Topics

Understanding regression provides the foundation for learning more complex models. The concepts of loss functions, gradient descent, and evaluation metrics are universal.

- Loss function design
- Gradient-based optimization
- Model evaluation principles
- Bias-variance tradeoff

Continue Learning

Practice Implementation

Apply these concepts to new datasets and problems

Explore Extensions

Regularization, feature engineering, and model selection

Build Projects

Create end-to-end ML projects with real-world impact