Mobile Price Prediction

Machine Learning Course Project

Arkaprava Jas

Student ID: B2530072

Ramakrishna Mission Vivekananda Educational & Research Institute
Department of Computer Science

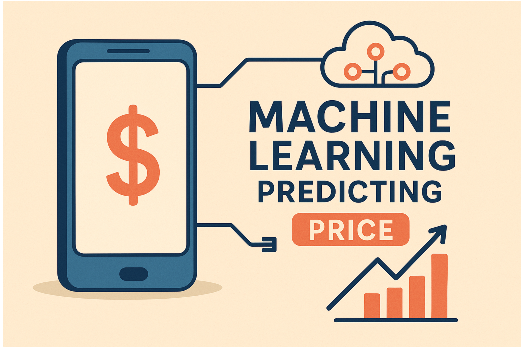
22,November 2025

Outline

Motivation

Motivation

“Accurate price prediction empowers customers, manufacturers, and markets — and machine learning makes it possible.”



Problem Statement

Objective

Predict the **price range** of mobile phones based on technical specifications

Input Features:

- RAM (GB)
- Storage (GB)
- Screen Size (inches)
- Camera (MP)
- Battery (mAh)
- Brand (encoded)

Goal:

- Compare multiple regression models
- Evaluate using MSE, MAE, R^2
- Select best-performing approach

Dataset Overview

Source

Kaggle Mobile Price Classification Dataset (preprocessed)

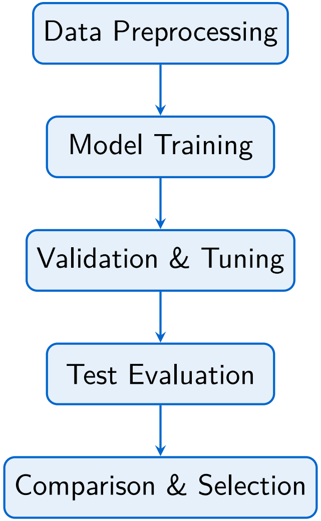
Data Split

- 70 % Training
- 20 % Validation
- 10 % Testing

Dataset Dimensions:

- **381 samples**
- **8 features**
- 1 target variable (Price)

Experimental Workflow



Models Implemented

Analytical Methods:

- Linear Regression (OLS)
- Normal Equation
- Singular Value Decomposition

Iterative Methods:

- Batch Gradient Descent
- Stochastic Gradient Descent
- Mini-Batch Gradient Descent

Regularization:

- Ridge Regression
- Lasso Regression
- Elastic Net

Polynomial Models:

- Degree 2, 3, 4

Validation Performance

Model	Val MSE	Val MAE	Val R^2
Linear Regression	34,624.11	129.29	0.676
Polynomial (deg 2)	36,389.72	127.35	0.659
Batch GD	**31,495.72**	**122.00**	**0.705**
Normal Eq / SVD	32,689.21	123.08	0.694
Polynomial (deg 3)	**193,689.16**	**186.12**	**-0.814**

Key Observation

High-degree polynomials (≥ 3) exhibit **severe overfitting** with negative R^2 values

K-Fold Cross Validation (k=5)

Model	MSE	MAE	R^2
Linear / Ridge / Lasso / EN	**25,872.11**	**118.66**	**0.724**
Polynomial (Degree 2)	27,329.29	106.81	0.716
Polynomial (Degree 3)	**127,306.14**	**146.80**	**-0.300**

Winner

Linear models with regularization achieve **best stability and accuracy**

Test Set Performance

Model	Test MSE	Test MAE	Test R^2
Linear Regression	17,119.27	106.30	0.731
Polynomial (deg 2)	15,730.50	92.00	0.753
Batch GD	**14,903.55**	**100.94**	**0.766**
Normal Eq / SVD	15,701.96	102.55	0.753

Champion Model

Batch Gradient Descent

MSE: 14,903.55 | R^2: 0.766

Computational Efficiency

Model	**Training Time (seconds)**
SVD	**0.000479**
Normal Equation	**0.001411**
Linear Regression (OLS)	0.002824
Batch Gradient Descent	0.084035
Stochastic Gradient Descent	0.408012
Mini-Batch Gradient Descent	**1.580085**

Fastest

Analytical methods
(SVD & Normal Eq)

Slowest

Mini-Batch GD
(iterative updates)

Comparative Analysis

Best Accuracy: Batch Gradient Descent (lowest test MSE, highest R^2)

Most Stable: Linear Regression, Ridge, Lasso, Elastic Net

Fastest: SVD and Normal Equation — ideal for small/medium datasets

Best Generalization: Ridge (slight improvement over OLS)

Worst Performance: Polynomial (degree ≥ 3) — severe overfitting

Recommendation

Use **Linear/SVD/Normal Eq** for speed and reliability
Use **Batch GD** for large datasets where matrix inversion is expensive

Why Analytical Solvers Excel Here

Advantages:

- **Orders of magnitude faster**
- Numerically stable
- Exact solution (no convergence issues)
- Optimized BLAS routines

Perfect for this dataset:

- Small size (381×8)
- Matrix inversion is trivial
- No computational bottleneck

Performance

R^2**:** 0.72–0.75

Time: $\sim$0.001 sec

Winner!

Batch Gradient Descent Performance

Key Result

Best test MSE (**14,903.55**) despite longer training time

Why it works well for this dataset:

- Smooth convergence with stable updates
- Gradient computation cost: $O(mn) = O(381 \times 8)$ — extremely cheap
- Training time (0.084s) still practical for this scale

Complexity Analysis

$$m = 381 \text{ (samples)}$$

$$n = 8 \text{ (features)}$$

$$\text{Cost per iteration} = O(mn) \approx 3000 \text{ operations}$$

Scalability: When Does Batch GD Become Necessary?

Analytical Methods:

Normal Equation:

$$O(n^3)$$

SVD Decomposition:

$$O(mn^2)$$

Problems at scale:

- × Matrix inversion infeasible
- × Memory requirements explode
- × Numerical instability

Batch Gradient Descent:

Gradient Computation:

$$O(mn)$$

Advantages at scale:

- Linear scaling
- No matrix inversion
- Memory efficient
- Parallelizable

Conclusion

For **large datasets**, Batch GD is the **only practical optimization method**

Key Takeaways

1. **For this dataset:** Analytical solvers (SVD, Normal Eq) are optimal
   - Fast, accurate, and stable
   - No hyperparameter tuning needed
2. **Batch GD** achieved best test accuracy but requires more time
3. **Regularization** (Ridge, Lasso) improves generalization slightly
4. **High-degree polynomials** must be avoided — severe overfitting
5. **For large-scale problems:** Gradient-based methods become necessary

References

- Kaggle.*Mobile Price Classification Dataset*.
- Murphy, K. P. (2022). *Probabilistic Machine Learning: An Introduction*. MIT Press.

Thank You