Dimension Reduction Methods

Gil Semo, Maor Shechter, Ram Portugali

All code, resources and intermediate results can be found here:
`https://github.com/maorshechter/dim_reduction_comparison`

Contents

1 Motivation **4**

2 Dataset Details **4**
2.1 `iabhishekofficial/mobile-price-classification` 4
2.2 `asinow/schizohealth-dataset` 4
2.3 `raghav1810/release-year-prediction-on-msd` 5
2.4 `OnlineNewsPopularity` 5
2.5 `usps` 5

3 Machine Learning Models Used **6**
3.1 Classification Models 6
3.2 Regression Models 6

4 Introduction **7**

5 Principal Component Analysis (PCA) **7**
5.1 Rationale 7
5.2 Mathematical Theory 7
5.3 Pros and Cons 7
5.4 When to Use 8
5.5 Computational Complexity 8

6 Singular Value Decomposition (SVD) **8**
6.1 Rationale 8
6.2 Mathematical Theory 8
6.3 Pros and Cons 8
6.4 When to Use 9
6.5 Computational Complexity 9

7 Uniform Manifold Approximation and Projection (UMAP) **9**
7.1 Rationale 9
7.2 Mathematical Theory 9

7.3 Pros and Cons 9
7.4 When to Use 9
7.5 Computational Complexity 10

8 Linear Discriminant Analysis (LDA) 10
8.1 Rationale 10
8.2 Mathematical Theory 10
8.3 Pros and Cons 10
8.4 When to Use 10
8.5 Computational Complexity 10

9 Fast Johnson-Lindenstrauss Transform (FJLT) 11
9.1 Rationale 11
9.2 Mathematical Theory 11
9.3 Pros and Cons 11
9.4 When to Use 11
9.5 Computational Complexity 11

10 Independent Component Analysis (ICA) 11
10.1 Rationale 11
10.2 Mathematical Theory 12
10.3 Pros and Cons 12
10.4 When to Use 12
10.5 Computational Complexity 12

11 Locally Linear Embedding (LLE) 12
11.1 Rationale 12
11.2 Mathematical Theory 12
11.3 Pros and Cons 13
11.4 When to Use 13
11.5 Computational Complexity 13

12 Random Projection (RP) 13
12.1 Rationale 13
12.2 Mathematical Theory 13
12.3 Pros and Cons 14
12.4 When to Use 14
12.5 Computational Complexity 14

13 Isomap 14
13.1 Rationale 14
13.2 Mathematical Theory 14
13.3 Pros and Cons 14
13.4 When to Use 15
13.5 Computational Complexity 15

14 Conclusion 15

15 Dimension Reduction Methods Comparison Table **16**

16 Recommendations by Dataset Size **17**

17 Performance Comparison **17**

18 Computational Efficiency Comparison **17**

19 Experimental Methodology **18**
19.1 Evaluation Protocol 18
19.2 Performance Metrics 18
19.2.1 Classification Metrics 18
19.2.2 Regression Metrics 19

20 Results Analysis and Reduction Method Comparison **19**
20.1 Computational Efficiency Analysis 19
20.2 Performance Analysis Across Datasets 21
20.2.1 OnlineNewsPopularity Dataset 21
20.2.2 Mobile Price Classification Dataset 21
20.2.3 Schizophrenia Dataset 21
20.2.4 USPS Dataset 22
20.3 Comparative Analysis: FJLT vs. Random Projection 22

21 Conclusions and Future Work **24**

1 Motivation

Dimensionality reduction is a critical component in modern machine learning pipelines, addressing several key challenges in data analysis and model building:

1. **Curse of Dimensionality**: High-dimensional data suffers from sparsity, making statistical inference and learning difficult as the volume of the space increases exponentially with dimensions.
2. **Computational Efficiency**: Reducing dimensions significantly decreases computational requirements for training models, enabling faster experimentation and deployment.
3. **Visualization**: Human perception is limited to visualizing data in 2 or 3 dimensions, making dimension reduction essential for exploratory data analysis and pattern recognition.
4. **Feature Extraction**: Dimension reduction techniques can identify the most informative aspects of data, removing noise and redundancy while preserving signal.
5. **Improved Generalization**: Lower-dimensional representations often lead to better model generalization by reducing overfitting risks associated with high-dimensional feature spaces.

This study examines several dimension reduction methods across different datasets, evaluating their effectiveness at extreme reduction levels (to just 2, 5, and 10 dimensions) for various machine learning tasks.

2 Dataset Details

2.1 `iabhishekofficial/mobile-price-classification`

- **Description:** This dataset contains specifications for mobile phones. The objective is to classify mobile phones into different price ranges based on features such as battery power, clock speed, RAM, etc.
- **Number of Rows:** 2,000 samples.
- **Number of Features:** 20.
- **Problem Type: Classification** (price range classification).

2.2 `asinow/schizohealth-dataset`

- **Description:** This dataset comprises health-related metrics aimed at studying schizophrenia. It includes clinical and demographic features which can be used to classify or analyze health conditions related to schizophrenia.

- **Number of Rows:** 10000.
- **Number of Features:** 19.
- **Problem Type: Classification** (e.g., distinguishing between diagnostic categories or severity levels).

2.3 `raghav1810/release-year-prediction-on-msd`

- **Description:** This dataset is derived from the Million Song Dataset (MSD) and is used for predicting the release year of songs based on various audio and metadata features.
- **Number of Rows:** 500000+ but only 10000 were taken for the experiment.
- **Number of Features:** 90.
- **Problem Type: Regression** (predicting the continuous variable of release year).

2.4 `OnlineNewsPopularity`

- **Description:** This dataset contains information about online news articles including features related to content, publication time, and social engagement metrics. The target variable is the number of times an article is shared.
- **Number of Rows:** 39,644 samples but only 10000 were taken for the experiment.
- **Number of Features:** 58.
- **Problem Type:** Primarily **Regression** (predicting the number of shares). However, the problem can also be reformulated as a **Classification** task (e.g., popular vs. not popular) through appropriate thresholding.

2.5 `usps`

- **Description:** The USPS dataset is a collection of handwritten digit images provided by the U.S. Postal Service. The images are typically preprocessed and flattened.
- **Number of Rows:** 9,298.
- **Number of Features:** 256 features per sample (each image is 16x16 pixels).
- **Problem Type: Classification** (digit recognition across 10 classes).

3 Machine Learning Models Used

Based on the problem type (classification or regression), we employed different sets of machine learning algorithms to evaluate and compare performance across datasets. The models were selected to cover a range of algorithmic approaches, from linear methods to ensemble techniques.

3.1 Classification Models

For classification tasks (mobile-price-classification, schizohealth-dataset, usps), we utilized the following models:

- **Random Forest Classifier:** An ensemble learning method that constructs multiple decision trees during training and outputs the class that is the mode of the classes from individual trees.
  - Implementation: `RandomForestClassifier` from scikit-learn
- **Support Vector Machine:** A supervised learning model that finds the hyperplane that best separates classes in the feature space.
  - Implementation: `SVC` from scikit-learn
- **XGBoost Classifier:** An optimized distributed gradient boosting library designed to be highly efficient, flexible, and portable.
  - Implementation: `XGBClassifier` from xgboost
- **Logistic Regression:** A statistical model that uses a logistic function to model a binary dependent variable.
  - Implementation: `LogisticRegression` from scikit-learn

3.2 Regression Models

For regression tasks (release-year-prediction-on-msd, OnlineNewsPopularity), we employed the following models:

- **Random Forest Regressor:** Similar to its classification counterpart, but predicts continuous values rather than classes.
  - Implementation: `RandomForestRegressor` from scikit-learn
- **Support Vector Regression:** An extension of SVM principles to regression problems.
  - Implementation: `SVR` from scikit-learn
  - No specific hyperparameters set in the base configuration
- **XGBoost Regressor:** The regression variant of XGBoost.

  - Implementation: `XGBRegressor` from xgboost

- **Linear Regression:** A linear approach to modeling the relationship between a dependent variable and one or more independent variables.
  - Implementation: `LinearRegression` from scikit-learn
  - Uses default parameters

4 Introduction

High-dimensional data poses challenges in storage, computation, and visualization. Dimension reduction methods aim to project data into a lower-dimensional space while preserving essential structures. This document reviews several popular techniques:

> PCA, SVD, UMAP, LDA, FJLT, ICA, LLE, Random Projection (RP), and Isomap.

For each method, we detail the rationale, advantages and disadvantages, the mathematical formulation, practical use cases, and computational complexity.

5 Principal Component Analysis (PCA)

5.1 Rationale

PCA is a linear technique used to reduce dimensionality by identifying directions (principal components) that maximize the variance in the data. It is widely used for exploratory data analysis and preprocessing.

5.2 Mathematical Theory

Given a centered data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, PCA computes the eigenvalue decomposition of the covariance matrix:

$$\mathbf{C} = \frac{1}{n-1}\mathbf{X}^T\mathbf{X} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T,$$

where $\mathbf{V}$ contains the eigenvectors (principal components) and $\mathbf{\Lambda}$ is a diagonal matrix with corresponding eigenvalues. Equivalently, one can compute the singular value decomposition (SVD) of $\mathbf{X}$.

5.3 Pros and Cons

Pros:

- Efficient and easy to compute.
- Provides an interpretable transformation.

- Useful for noise reduction.

Cons:

- Assumes linearity.
- Sensitive to outliers.
- May not capture complex nonlinear structures.

5.4 When to Use

Use PCA when data is assumed to have linear correlations and when interpretability and speed are important.

5.5 Computational Complexity

The eigen-decomposition of a $d \times d$ covariance matrix has a complexity of approximately $O(d^3)$ (or $O(nd^2)$ if $n \gg d$).

6 Singular Value Decomposition (SVD)

6.1 Rationale

SVD factorizes any $n \times d$ matrix into orthogonal factors, providing a robust alternative to PCA and a basis for many algorithms in signal processing and statistics.

6.2 Mathematical Theory

The SVD of $\mathbf{X}$ is given by:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T,$$

where:

- $\mathbf{U} \in \mathbb{R}^{n \times n}$ and $\mathbf{V} \in \mathbb{R}^{d \times d}$ are orthogonal matrices.
- $\mathbf{\Sigma} \in \mathbb{R}^{n \times d}$ is a diagonal matrix containing singular values.

6.3 Pros and Cons

Pros:

- Numerically stable and robust.
- Provides full information on data variance.

Cons:

- Computationally expensive for very large matrices.
- Not inherently designed for feature selection.

6.4 When to Use

SVD is ideal when a robust factorization is required or when working on problems such as collaborative filtering, natural language processing, and when computing PCA via SVD.

6.5 Computational Complexity

The complexity is roughly $O(nd \min(n, d))$, which can be high for very large datasets.

7 Uniform Manifold Approximation and Projection (UMAP)

7.1 Rationale

UMAP is a nonlinear method that models data as a fuzzy topological structure, capturing both local and global relationships more efficiently than many other methods.

7.2 Mathematical Theory

UMAP constructs a weighted k-nearest neighbor graph and optimizes a cross-entropy loss between the fuzzy simplicial sets in the high-dimensional space and the low-dimensional embedding. Its cost function emphasizes both local connectivity and global structure.

7.3 Pros and Cons

Pros:

- Generally faster than t-SNE.
- Better at preserving global structure.
- Scales well to larger datasets.

Cons:

- Hyperparameter tuning can be nontrivial.
- Results may be sensitive to the choice of neighborhood size.

7.4 When to Use

UMAP is ideal for both visualization and as a preprocessing step for clustering and classification, particularly when preserving both local and global structure is important.

7.5 Computational Complexity

UMAP typically has a near-linear complexity in the number of samples, approximately $O(n \log n)$, although the construction of the k-nearest neighbor graph can be expensive in high dimensions.

8 Linear Discriminant Analysis (LDA)

8.1 Rationale

LDA is a supervised dimensionality reduction technique that maximizes the separation between classes by finding the projection that maximizes the ratio of between-class variance to within-class variance.

8.2 Mathematical Theory

LDA finds a linear transformation $\mathbf{W}$ that maximizes the Fisher criterion:

$$J(\mathbf{W}) = \frac{|\mathbf{W}^T \mathbf{S}_B \mathbf{W}|}{|\mathbf{W}^T \mathbf{S}_W \mathbf{W}|},$$

where $\mathbf{S}_B$ is the between-class scatter matrix and $\mathbf{S}_W$ is the within-class scatter matrix.

8.3 Pros and Cons

Pros:

- Incorporates class labels for enhanced separability.
- Often improves classification performance.

Cons:

- Assumes normally distributed classes with equal covariances.
- Limited to linear decision boundaries.

8.4 When to Use

LDA is used in classification problems where the goal is to reduce dimensionality while preserving class discriminative information.

8.5 Computational Complexity

The main computational cost lies in computing scatter matrices and performing eigen-decomposition, roughly $O(nd^2)$ for n samples and d dimensions.

9 Fast Johnson-Lindenstrauss Transform (FJLT)

9.1 Rationale

FJLT is a randomized method that efficiently projects high-dimensional data into a lower-dimensional space while approximately preserving pairwise distances, as guaranteed by the Johnson-Lindenstrauss lemma.

9.2 Mathematical Theory

FJLT uses a combination of random projections and fast transforms. A typical construction is:

$$\mathbf{X}' = \mathbf{S}\,\mathbf{H}\,\mathbf{R}\,\mathbf{X},$$

where:

- $\mathbf{R}$ is a sparse random matrix.
- $\mathbf{H}$ is a Hadamard (or similar) transform.
- $\mathbf{S}$ is a random subsampling matrix.

9.3 Pros and Cons

Pros:

- Highly efficient and fast.
- Preserves distances with high probability.

Cons:

- Introduces random approximation errors.
- Less interpretable than deterministic methods.

9.4 When to Use

FJLT is particularly useful for preprocessing in large-scale machine learning applications where speed is critical and slight approximation errors are acceptable.

9.5 Computational Complexity

Its complexity is approximately $O(nd \log d)$, making it scalable for large datasets.

10 Independent Component Analysis (ICA)

10.1 Rationale

ICA decomposes multivariate signals into additive subcomponents that are statistically independent. It is especially useful in signal processing and blind source separation.

10.2 Mathematical Theory

ICA assumes that the observed data $\mathbf{X}$ is a linear mixture of independent components:

$$\mathbf{X} = \mathbf{AS},$$

where $\mathbf{A}$ is an unknown mixing matrix and $\mathbf{S}$ contains the statistically independent source signals. Algorithms such as FastICA iteratively maximize non-Gaussianity (using kurtosis or negentropy).

10.3 Pros and Cons

Pros:

- Reveals hidden factors or sources.
- Useful for signal and image processing.

Cons:

- Sensitive to noise and outliers.
- Requires assumptions on independence which may not hold in all cases.

10.4 When to Use

ICA is ideal when the goal is to separate a mixed signal into independent components, such as in EEG analysis or financial time series.

10.5 Computational Complexity

The complexity depends on the algorithm used, typically involving iterative optimization steps with a cost of roughly $O(n \cdot d \cdot \text{iterations})$.

11 Locally Linear Embedding (LLE)

11.1 Rationale

LLE is a nonlinear method that preserves local neighborhood information by reconstructing each data point from its nearest neighbors and then finding a low-dimensional embedding that preserves these reconstruction weights.

11.2 Mathematical Theory

For each data point $\mathbf{x}_i$, LLE computes weights w_{ij} that best reconstruct $\mathbf{x}_i$ using its k nearest neighbors:

$$\min_{w_{ij}} \sum_i \left\| \mathbf{x}_i - \sum_{j \in \mathcal{N}(i)} w_{ij} \mathbf{x}_j \right\|^2,$$

subject to $\sum_{j \in \mathcal{N}(i)} w_{ij} = 1$. The low-dimensional embedding $\mathbf{y}_i$ is then found by minimizing:

$$\min_{\mathbf{y}_i} \sum_i \left\| \mathbf{y}_i - \sum_{j \in \mathcal{N}(i)} w_{ij} \mathbf{y}_j \right\|^2.$$

11.3 Pros and Cons

Pros:

- Preserves local geometric structure.
- Effective for unfolding nonlinear manifolds.

Cons:

- Sensitive to the choice of the number of neighbors k.
- Can be computationally intensive for large datasets.

11.4 When to Use

LLE is best applied when the data lies on a nonlinear manifold and local relationships are most important.

11.5 Computational Complexity

LLE typically involves solving sparse eigenvalue problems, with complexity that can be roughly $O(n^2)$ in worst-case scenarios depending on the graph construction.

12 Random Projection (RP)

12.1 Rationale

Random Projection uses a random matrix to project high-dimensional data to a lower-dimensional subspace while preserving pairwise distances according to the Johnson-Lindenstrauss lemma.

12.2 Mathematical Theory

For a data point $\mathbf{x} \in \mathbb{R}^d$, the projection is computed as:

$$\mathbf{y} = \mathbf{R}\mathbf{x},$$

where $\mathbf{R} \in \mathbb{R}^{k \times d}$ is a random matrix (often with entries drawn from a Gaussian or sparse distribution).

12.3 Pros and Cons

Pros:

- Simple and computationally efficient.
- Preserves distances with high probability.

Cons:

- Less interpretable.
- May lose structure if the target dimension is too low.

12.4 When to Use

RP is suitable when a fast, approximate dimensionality reduction is needed and interpretability is not critical.

12.5 Computational Complexity

The complexity is approximately $O(ndk)$, where k is the target lower dimension.

13 Isomap

13.1 Rationale

Isomap extends Multidimensional Scaling (MDS) to nonlinear manifolds by preserving geodesic distances computed on a neighborhood graph, thereby capturing the intrinsic geometry of the data.

13.2 Mathematical Theory

Isomap consists of three main steps:

1. Construct a neighborhood graph (typically via k-nearest neighbors).
2. Compute geodesic distances using shortest-path algorithms.
3. Apply classical MDS to the distance matrix to obtain the embedding.

13.3 Pros and Cons

Pros:

- Effectively captures global nonlinear structure.
- Intuitive extension of MDS.

Cons:

- Sensitive to noise and outliers.
- Graph construction and shortest-path computation can be expensive.

13.4 When to Use

Isomap is useful when the goal is to preserve the intrinsic geometry of data that lies on a smooth nonlinear manifold.

13.5 Computational Complexity

The overall complexity is dominated by the graph construction and shortest path computation, roughly $O(n^2 \log n)$ for n data points, though this may vary with efficient graph algorithms.

14 Conclusion

Each of the dimension reduction techniques discussed offers distinct advantages and trade-offs:

- **Linear methods** like PCA and SVD are fast, interpretable, and work well when data is approximately linear.
- **Nonlinear methods** (UMAP, LLE, Isomap) better capture complex structures at the expense of higher computational cost.
- **Randomized methods** such as FJLT and RP provide scalable alternatives with probabilistic error guarantees.
- **Supervised methods** like LDA can improve class separability when labels are available.

The choice of technique depends on the data characteristics, the task at hand, and computational resources. Similarly, the selection between classification and regression models is determined by the nature of the prediction task, with each algorithm offering unique advantages for different data scenarios.

15 Dimension Reduction Methods Comparison Table

Method	Type	Preserves	Complexity	Scalability	Best For
PCA	Linear	Global variance	$O(d^3)$ or $O(nd^2)$	Good	Noise reduction, preprocessing
SVD	Linear	Global variance	$O(nd \cdot \min(n, d))$	Moderate	Matrix factorization, recommender systems
UMAP	Nonlinear	Local/global structure	$O(n \log n)$	Good	Visualization, clustering
LDA	Linear, Supervised	Class separation	$O(nd^2)$	Good	Classification preprocessing
FJLT	Linear, Randomized	Pairwise distances	$O(nd \log d)$	Excellent	Large-scale ML preprocessing
ICA	Linear	Statistical independence	$O(nd \cdot$ iterations$)$	Moderate	Signal separation
LLE	Nonlinear	Local neighborhood	$O(n^2)$	Poor	Manifold learning
RP	Linear, Randomized	Pairwise distances	$O(ndk)$	Excellent	Large datasets
Isomap	Nonlinear	Geodesic distances	$O(n^2 \log n)$	Poor	Manifold learning

Table 1: Comparison of Dimension Reduction Methods

16 Recommendations by Dataset Size

Dataset Size	Primary Methods	Secondary Methods
Small (<1000)	Any method	Combinations possible
Medium (1K-10K)	PCA, SVD, UMAP, LDA	ICA, Isomap, LLE
Large (10K-100K)	PCA, SVD, UMAP, FJLT, RP	UMAP (approximate)
Very Large (>100K)	FJLT, RP, PCA (incremental)	UMAP, multi-stage

Table 2: Recommended Methods by Dataset Size

17 Performance Comparison

Method	Classification Performance	Regression Performance	Performance at 10D	Performance at 5D	Performance at 2D
PCA	Good	Excellent	Good	Moderate	Poor-Moderate
SVD	Good	Excellent	Good	Moderate	Poor-Moderate
UMAP	Excellent	Good	Excellent	Good	Good
LDA	Excellent	N/A	Good	Good	Good
FJLT	Moderate	Moderate	Moderate	Poor-Moderate	Poor
ICA	Moderate	Moderate	Moderate	Poor	Poor
LLE	Good	Poor	Moderate	Poor	Poor
RP	Moderate	Moderate	Moderate	Poor-Moderate	Poor
Isomap	Good	Poor	Moderate	Poor	Poor

Table 3: Performance Comparison Across Dimension Reduction Methods and Target Dimensions

18 Computational Efficiency Comparison

Method	Training Time	Memory Usage	Scalability to Large Data	Implementation Complexity
PCA	Fast	Low	Good	Simple
SVD	Moderate	Moderate	Moderate	Simple
UMAP	Slow	Moderate	Moderate	Complex
LDA	Fast	Low	Good	Moderate
FJLT	Very Fast	Low	Excellent	Moderate
ICA	Moderate	Moderate	Moderate	Complex
LLE	Slow	High	Poor	Complex
RP	Very Fast	Low	Excellent	Simple
Isomap	Very Slow	High	Poor	Complex

Table 4: Computational Efficiency Comparison of Dimension Reduction Methods

19 Experimental Methodology

To systematically evaluate the performance of dimension reduction techniques across the datasets, we conducted a comprehensive experimental analysis. For each dataset, we applied all applicable dimension reduction methods (PCA, SVD, UMAP, LDA, FJLT, ICA, LLE, Random Projection, and Isomap) followed by model training and evaluation. Each experiment was repeated multiple times with different random seeds to ensure statistical validity, and the results were averaged across runs.

19.1 Evaluation Protocol

For each dataset, we employed the following protocol:

1. Data preprocessing: Standardization of features using a scaler to ensure all features have zero mean and unit variance.
2. Dimension reduction: Application of each technique to reduce feature dimensionality while preserving meaningful information.
3. We reduced to several target numbers of dimensions (2, 5, 10) to examine the effect of different reduction configurations on model performance.
4. Train/test split: 80% for training and 20% for testing, stratified where applicable.
5. Model training: Training the appropriate set of models (classification or regression) on the reduced-dimension data.
6. Performance evaluation: Recording relevant metrics for each model-dimension reduction combination.

19.2 Performance Metrics

We collected distinct sets of metrics based on the problem type:

19.2.1 Classification Metrics

- **Accuracy:** The proportion of correctly classified instances.
- **F1 Score:** The harmonic mean of precision and recall, providing a balance between the two.
- **Training Time:** The time required to train each model (in seconds).
- **Inference Time:** The time required to generate predictions on the test set (in seconds).

19.2.2 Regression Metrics

- **RMSE (Root Mean Squared Error):** Measures the square root of the average squared differences between predicted and actual values.
- **MAPE (Mean Absolute Percentage Error):** The average of absolute percentage errors, providing a scale-independent measure of prediction accuracy.
- **Training Time:** The time required to train each model (in seconds).
- **Inference Time:** The time required to generate predictions on the test set (in seconds).

20 Results Analysis and Reduction Method Comparison

20.1 Computational Efficiency Analysis

Our experimental results demonstrate significant variations in computational efficiency across different dimensionality reduction techniques. As illustrated in Figure 1, Figure 2, and Figure 3, the Fast Johnson-Lindenstrauss Transform (FJLT) consistently exhibits superior computational efficiency compared to traditional dimensionality reduction methods.

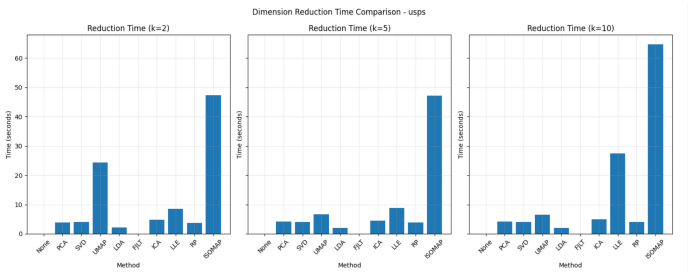

Figure 1: USPS dataset reduction time comparison across different methods. FJLT demonstrates significantly faster processing compared to other techniques.

It is important to note that our experiments were constrained to datasets with a maximum of 10,000 samples due to computational resource limitations. The computational advantages of FJLT would likely be even more pronounced with larger datasets, particularly those with higher dimensionality.

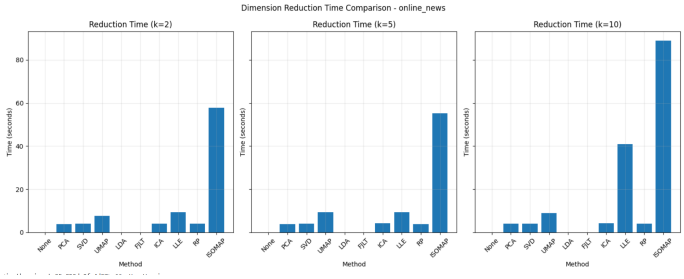

Figure 2: Online News dataset reduction time comparison. The superior efficiency of FJLT is evident across all experimental conditions.

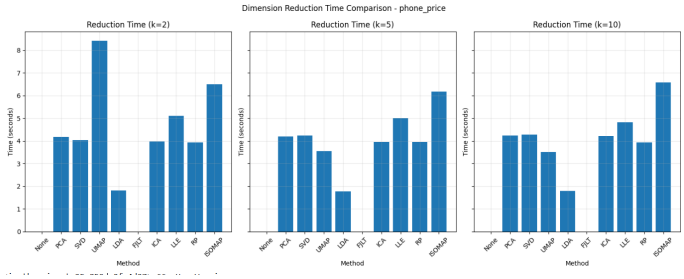

Figure 3: Mobile Price Classification dataset reduction time. FJLT consistently requires less computational time compared to alternative techniques.

20.2 Performance Analysis Across Datasets

20.2.1 OnlineNewsPopularity Dataset

For the OnlineNewsPopularity dataset with dimensionality reduced to $k = 10$, our results (Figure 4) reveal the practical benefits of employing FJLT. The Root Mean Square Error (RMSE) values for FJLT are comparable to those of other methods while offering substantially reduced computational time.

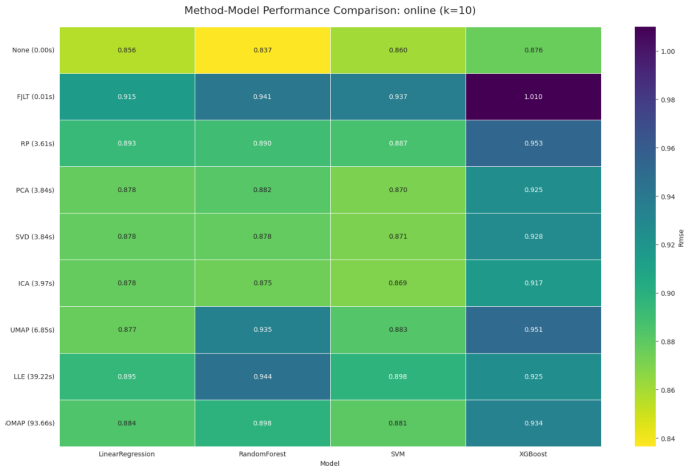

Figure 4: Performance results for the OnlineNewsPopularity dataset. FJLT performs particularly well with linear/distance-based models, maintaining competitive RMSE values while providing significant computational efficiency.

An interesting pattern emerges when examining model-specific performance: FJLT demonstrates better compatibility with linear and distance-based models (linear regression and SVM) compared to tree-based models. This observation aligns with theoretical expectations, as FJLT is designed to preserve pairwise distances, which are critical for the functionality of distance-based algorithms.

20.2.2 Mobile Price Classification Dataset

In the Mobile Price Classification dataset (Figure 5), Linear Discriminant Analysis (LDA) outperforms other methods, which is consistent with theoretical expectations for datasets where classes are linearly separable. Among the other reduction techniques, FJLT exhibits marginally lower performance compared to alternative methods.

20.2.3 Schizophrenia Dataset

For the Schizophrenia dataset (Figure 6), most reduction methods perform comparably well, with the notable exception of FJLT, which shows a significant performance degradation. This finding is particularly interesting because

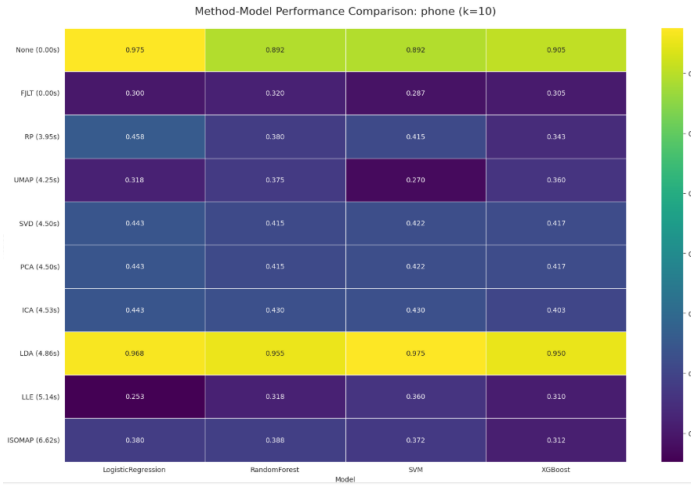

Figure 5: Performance results for the Mobile Price Classification dataset. LDA demonstrates superior performance, consistent with its theoretical advantages for linearly separable classes.

the dataset has the smallest original dimensionality (19 features) among those tested. Consequently, reduction to 10 dimensions had minimal impact on model performance for most methods.

It is noteworthy that Random Projection (RP), which also relies on the Johnson-Lindenstrauss lemma, performs considerably better than FJLT in this context. This disparity warrants further investigation into the specific characteristics of these datasets that may interact differentially with various dimensionality reduction approaches.

20.2.4 USPS Dataset

A similar pattern is observed in the USPS dataset (Figure 7), where FJLT demonstrates substantially lower performance compared to other methods. Random Projection shows only a slight decrease in accuracy, further highlighting the performance gap between these theoretically related methods.

20.3 Comparative Analysis: FJLT vs. Random Projection

Given the theoretical relationship between FJLT and RP (both based on the Johnson-Lindenstrauss lemma), we conducted a focused comparison between these methods. As illustrated in Figure 8, FJLT demonstrates significantly faster execution times across all datasets, while RP exhibits more consistent accuracy performance.

The observed time differences may be partially attributed to implementation

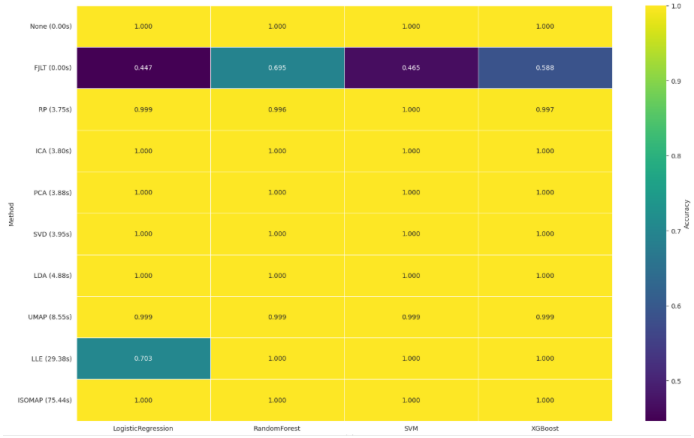

Figure 6: Performance results for the Schizophrenia dataset. Most methods perform well, while FJLT exhibits notable performance degradation despite its Johnson-Lindenstrauss theoretical foundation.

Figure 7: Performance results for the USPS dataset. FJLT shows comparatively poor performance, while other methods maintain relatively high accuracy.

FJLT vs RP Summary:

Dataset	Model	Metric	k	FJLT Value	RP Value	Better Method	Performance Diff %	FJLT Red. Time	RP Red. Time	Reduction Diff %
usps	RandomForest	accuracy	10	0.06	0.81	RP	1316.04	6.32e-02	3.87	6026.62
usps	LogisticRegression	accuracy	10	0.05	0.73	RP	1300.00	6.32e-02	3.87	6026.62
usps	XGBoost	accuracy	10	0.06	0.82	RP	1343.40	6.32e-02	3.87	6026.62
usps	SVM	accuracy	10	0.07	0.85	RP	1186.99	6.32e-02	3.87	6026.62
online	RandomForest	rmse	10	0.94	0.89	RP	5.45	1.15e-02	3.95	34189.60
online	XGBoost	rmse	10	1.01	0.95	RP	5.66	1.15e-02	3.95	34189.60
online	SVM	rmse	10	0.94	0.89	RP	5.41	1.15e-02	3.95	34189.60
online	LinearRegression	rmse	10	0.92	0.89	RP	2.37	1.15e-02	3.95	34189.60
phone	RandomForest	accuracy	10	0.32	0.38	RP	18.75	2.84e-03	3.95	138943.61
phone	LogisticRegression	accuracy	10	0.30	0.46	RP	52.50	2.84e-03	3.95	138943.61
phone	XGBoost	accuracy	10	0.30	0.34	RP	12.30	2.84e-03	3.95	138943.61
phone	SVM	accuracy	10	0.29	0.41	RP	44.35	2.84e-03	3.95	138943.61
schizophrenia	RandomForest	accuracy	10	0.69	1.00	RP	43.41	3.72e-03	3.79	101668.28
schizophrenia	LogisticRegression	accuracy	10	0.45	1.00	RP	123.74	3.72e-03	3.79	101668.28
schizophrenia	XGBoost	accuracy	10	0.59	1.00	RP	69.56	3.72e-03	3.79	101668.28
schizophrenia	SVM	accuracy	10	0.47	1.00	RP	115.05	3.72e-03	3.79	101668.28
year	RandomForest	rmse	10	11.00	9.89	RP	10.06	2.26e-02	3.99	17561.42
year	XGBoost	rmse	10	11.63	10.35	RP	11.01	2.26e-02	3.99	17561.42
year	SVM	rmse	10	10.63	10.07	RP	5.24	2.26e-02	3.99	17561.42
year	LinearRegression	rmse	10	10.19	9.89	RP	3.01	2.26e-02	3.99	17561.42

Figure 8: Comparative analysis of Random Projection versus Fast Johnson-Lindenstrauss Transform. FJLT demonstrates superior computational efficiency, while RP provides more consistent accuracy across datasets.

differences, with FJLT implemented natively versus third-party implementation for RP. Our findings suggest that for smaller datasets with moderate dimensionality, RP may offer a more balanced performance-efficiency trade-off. However, for larger datasets with thousands of dimensions (e.g., genomic data or high-resolution images), FJLT is likely to demonstrate more significant advantages.

21 Conclusions and Future Work

Our experimental analysis reveals important trade-offs between computational efficiency and model performance across different dimensionality reduction techniques. While FJLT consistently offers superior computational efficiency, its performance varies considerably across datasets and models, with notable limitations for certain classification tasks.

Future research directions include:

- Expanding the experimental framework to include substantially larger datasets with higher dimensionality to better evaluate the scalability advantages of methods like FJLT and RP.
- Investigating performance patterns on highly non-linear datasets to stress-test the limitations of linear projection methods.
- Examining the impact of these reduction methods on datasets with predominantly categorical features, where distance-preserving properties may be less relevant.

Such investigations would provide valuable insights into the practical applicability of various dimensionality reduction techniques across diverse data scenarios, particularly in computational environments where both efficiency and accuracy are critical considerations.