

VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING



Machine Learning (CO3117)

ASSIGNMENT REPORT
COMPREHENSIVE MACHINE LEARNING MODELS

TEACHER: Nguyen An Khuong

—o0o—

Group: ML_LearningMachine

Student 1: Phung Ba Trieu 2053524

Student 2: Le Nguyen Minh Giang 2052966

Student 3: Thai Quang Du 2252136

HO CHI MINH CITY, 5/2025

Contents

1	Introduction	3
1.1	Objectives	3
1.2	Data Set Overview	4
2	Theoretical Background	5
2.1	Bayesian Networks	5
2.2	Ensemble Methods	5
2.3	Support Vector Machines (SVM)	6
2.4	Dimensionality Reduction and Topic Modeling (PCA / LDA)	6
2.4.1	Principal Component Analysis (PCA)	6
2.4.2	Latent Dirichlet Allocation (LDA)	6
3	Model Implementation	8
3.1	Bayesian Network	8
3.2	Ensemble Method	10
3.3	Support-Vector Machine	14
3.3.1	Concept	14
3.3.2	Environment	15
3.3.3	Results	15
3.3.4	Qualitative Check — First 5 Test Rows	16
3.3.5	Discussion	17
3.3.6	Source & Artefacts	17
3.3.7	Conclusion	17
3.4	Dimensionality Reduction (PCA/LDA)	17
3.4.1	Text Preprocessing and Feature Engineering	17
3.4.2	Feature Selection Techniques	18
3.4.3	Topic Modeling using Latent Dirichlet Allocation (LDA)	19
3.4.4	Dimensionality Reduction using Truncated SVD (PCA for Sparse Data)	20
3.4.5	Conclusion	20

Full Name	Student ID	Task Responsibility	Contribution (%)
Phung Ba Trieu	2053524	Data preprocessing, implement SVM model	100%
Le Nguyen Minh Giang	2052966	Implement PCA/LDA, report	100%
Thai Quang Du	2252136	Implement Bayesian Network model and Ensemble method	100%

Table 1: *Group member task distribution and contribution rate*

Chapter 1

Introduction

1.1 Objectives

This project aims to explore and compare the effectiveness of various machine learning models in handling structured and unstructured data for predictive analysis and feature understanding. The central goal is not only to implement each model accurately but also to interpret their behaviors, strengths, and limitations when applied to a shared dataset.

The selected models represent four different families of machine learning approaches:

1. Bayesian Networks – a probabilistic graphical model that encodes conditional dependencies between variables in the form of a directed acyclic graph (DAG).
2. Ensemble Methods – techniques that combine predictions from multiple base estimators to enhance generalization and robustness, including Random Forest (Bagging), AdaBoost and Gradient Boosting (Boosting), and Voting Classifiers.
3. Support Vector Machines (SVM) – a powerful supervised learning model effective in high-dimensional spaces, especially with sparse textual data using linear kernels.
4. Dimensionality Reduction and Topic Modeling (PCA/LDA) – unsupervised techniques to reduce feature space, discover latent semantic structures in text through Principal Component Analysis (Truncated SVD) and Latent Dirichlet Allocation.

By implementing and evaluating all four approaches on the same dataset, we aim to:

- Compare their predictive performance and resource efficiency.
- Understand the types of features or data structures each model handles best.
- Investigate the interpretability and practical deployment potential of each method.
- Develop insights into how model choice should be influenced by the data and task characteristics.

1.2 Data Set Overview

The dataset used for this project is titled [Media prediction and its cost.csv](#), which contains detailed records of various media types and their associated campaigns or distribution costs. This data set offers a combination of categorical, textual, and numerical features, making it suitable for multifaceted analysis using different machine learning paradigms.

Key characteristics:

- Size: Over 60,000 rows of real-world data
- Target feature: A discretized cost value categorized into three ordinal buckets (e.g., low, medium, high cost).
- Textual input: The `media_type` column, which includes short descriptions such as “paper,” “TV,” “radio,” “coupon,” etc.
- Feature types:
 1. Categorical (e.g., campaign type)
 2. Textual (e.g., `media_type`)
 3. Numerical (e.g., reach, impressions, spend)

This data set is particularly suitable for:

- Probabilistic modeling of relationships among features (Bayesian networks).
- Boosting and bagging techniques to improve classification accuracy (Ensemble methods).
- Handling high-dimensional text and numeric data (SVM).
- Latent topic extraction and dimensionality reduction for better interpretability (PCA/LDA).

By applying the same data set across all models, we ensure a consistent basis for evaluation and comparison. This also highlights how different techniques exploit different aspects of the same data: structure, hierarchy, sparsity, or semantics.

Chapter 2

Theoretical Background

2.1 Bayesian Networks

A **Bayesian Network** (BN) is a probabilistic graphical model that represents a set of random variables and their conditional dependencies using a *Directed Acyclic Graph* (DAG). Each node in the DAG corresponds to a variable, and directed edges encode conditional dependencies between them.

The joint probability distribution over a set of variables $X = \{X_1, X_2, \dots, X_n\}$ is factored as:

$$P(X_1, X_2, \dots, X_n) = \prod_{i=1}^n P(X_i \mid \text{Parents}(X_i))$$

Bayesian Networks allow for modeling causality, reasoning under uncertainty, and performing inference. In this project, structure learning is performed using the Hill Climbing algorithm, which iteratively optimizes a scoring function such as the Bayesian Information Criterion (BIC).

2.2 Ensemble Methods

Ensemble Methods combine predictions from multiple models to achieve improved performance over individual models. The key idea is that the aggregated prediction of diverse models is more accurate and robust.

Three primary types of ensembles are used:

- **Bagging (Bootstrap Aggregating):** Trains multiple models on different bootstrap samples and combines their outputs. *Random Forest* is a typical bagging algorithm.
- **Boosting:** Trains models sequentially where each new model focuses on correcting the errors of the previous one. *AdaBoost* and *Gradient Boosting* are used in this project.

- **Voting:** Combines predictions from heterogeneous models. In *hard voting*, the majority class is selected, while *soft voting* averages the predicted probabilities.

Ensemble models are known for improving generalization, handling noisy data, and reducing both bias and variance depending on the type.

2.3 Support Vector Machines (SVM)

A **Support Vector Machine** (SVM) is a supervised learning algorithm that aims to find the optimal hyperplane separating different classes with the maximum margin. For linearly separable data, the optimization objective is:

$$\text{Maximize } \frac{2}{\|w\|} \quad \text{subject to } y_i(w^T x_i + b) \geq 1$$

SVMs can be extended to non-linear cases using *kernel functions*. In this project, a linear kernel is used due to the high dimensional and sparse nature of the TF-IDF features.

The SVM model is efficient in high-dimensional spaces and effective in text classification tasks. It also handles class imbalance using the “class_weight=balanced” setting and supports multi-class classification using the one-vs-rest strategy.

2.4 Dimensionality Reduction and Topic Modeling (PCA / LDA)

This section involves two unsupervised learning techniques:

2.4.1 Principal Component Analysis (PCA)

PCA is a linear dimensionality reduction technique that projects data onto orthogonal components explaining the most variance. For sparse matrices such as TF-IDF, **Truncated Singular Value Decomposition (SVD)** is used:

$$X \approx U\Sigma V^T$$

Only the top- k singular values and corresponding vectors are retained. PCA reduces computational cost and facilitates data visualization and noise reduction.

2.4.2 Latent Dirichlet Allocation (LDA)

LDA is a generative probabilistic model used for topic modeling. It assumes that:

- Each document is a mixture of latent topics.

- Each topic is a distribution over words.

The generative process includes sampling a topic distribution for each document, selecting a topic for each word, and then generating the word from the topic's word distribution.

LDA is useful for discovering hidden semantic structures and has applications in document classification, summarization, and exploratory analysis.

Chapter 3

Model Implementation

3.1 Bayesian Network

Bayesian Network (BN) is a probabilistic graphical model that represents a set of variables and their conditional dependencies using a directed acyclic graph (DAG). Its concept is an advance version of Naïve Bayes model where instead of assuming all features are all independent of each other, Bayesian network model uses graph to demonstrate how each feature connects and affects the other ones' outcome.

The main problem of the Bayesian network is to figure out its feature relationships that related to the label. In normal context, people might calculate on themselves based on the feature technical probability. In this assignment we will find the relationship automatically using Hill-Climbing algorithms. Hill Climbing is a heuristic search algorithm used for mathematical optimization problems. It's a type of greedy algorithm that continuously moves towards the direction of increasing (or decreasing) value to find the peak (maximum) or valley (minimum) of a function.

For the programming, we will be using pgmpy library for coding. All the code for Bayesian Network and hill-climbing algorithms are all supported. Although, pgmpy only supports Python 3.10 version so we have to install the environment that is appropriate for the model.

After training the model which took about 20-25 min this is the result:

Classification Report:					
	precision	recall	f1-score	support	
0	0.90	0.95	0.92	242	
1	0.93	0.97	0.95	256	
2	0.95	0.90	0.92	236	
3	0.90	0.90	0.90	243	
4	0.98	0.92	0.95	232	
accuracy			0.93	1209	
macro avg	0.93	0.93	0.93	1209	
weighted avg	0.93	0.93	0.93	1209	

Figure 3.1

With the accuracy of 0.93, the model seems to have relatively high accuracy base on the all the feature on the CSV.

This is the confusion Heatmap for better perspective on what the model predicts right and wrong:

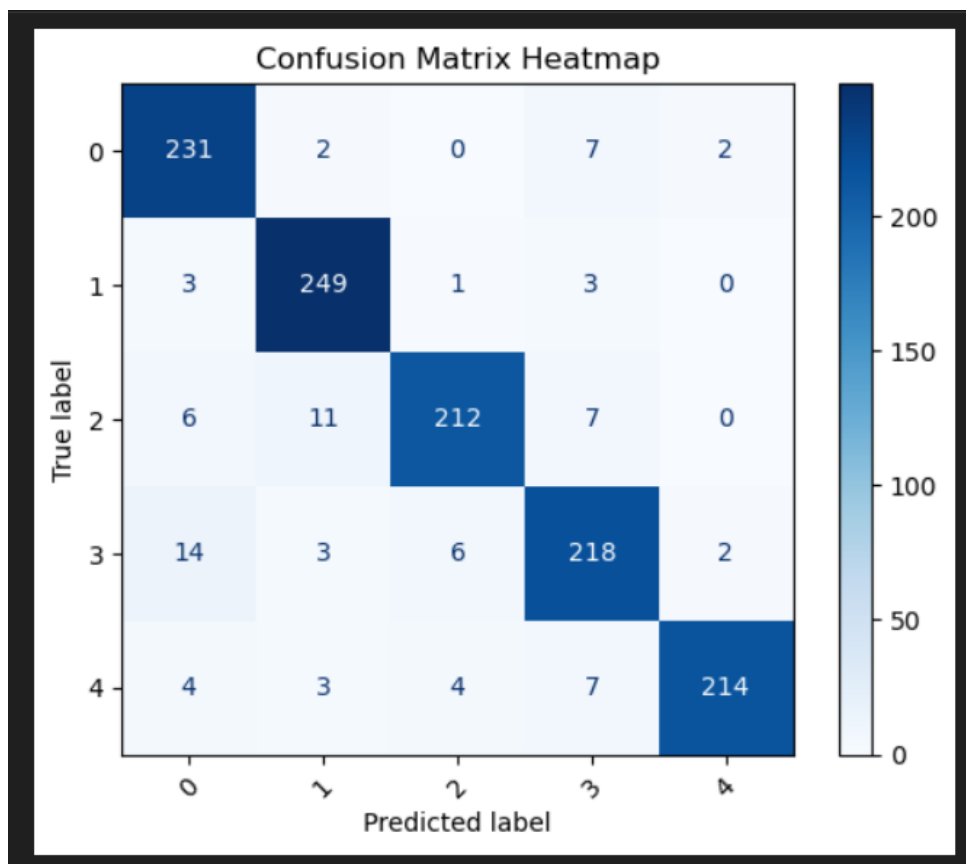


Figure 3.2

For reference please checkout the source code in the `src/models/Bayesian_Network/Bayesian_Network_main.ipynb` file.

In conclusion, although the Bayesian network have high accuracy, there are might be better model with higher accuracy and better suited for this task. For possible improvement, we notice that when increase the learning data size the model has the better grasp of the relationship of the feature, and therefore have better prediction, so if we have more dataset the model might yield better result.

3.2 Ensemble Method

The Ensemble Method combines predictions from multiple models to produce a more accurate and robust overall prediction than any single model alone. Although there are many types of ensemble method, for this assignment we will be using 3 model for 3 type: random forest for bagging type, AdaBoost and Gradient Boosting for Boosting type and voting type. Bagging method is combine the result of all the model and give out the final result. With Boosting, the model build sequentially so that the new model will focus on correcting the previous model. With voting, the models will give each give their own prediction and base on the how many results of each class get, the class that have the most result will be the final result. We will be using logistic regression, SVC and random forest along for the voting method.

For the coding we will be using sklearn for all the above methods. Sklearn.ensemble for the AdaBoost, Gradient Boosting, voting method and for random forest; Sklearn.linear_model for the logistic regression.

For reference please check out the source code in the src/models/Ensemble_methods/Ensemble_main.ipynb file.

This is the result for the bagging method

	precision	recall	f1-score	support
0	0.99	0.99	0.99	3661
1	1.00	0.99	1.00	3607
2	1.00	0.99	1.00	3607
3	0.99	0.99	0.99	3580
4	0.99	0.99	0.99	3674
accuracy			0.99	18129
macro avg	0.99	0.99	0.99	18129
weighted avg	0.99	0.99	0.99	18129

Figure 3.3

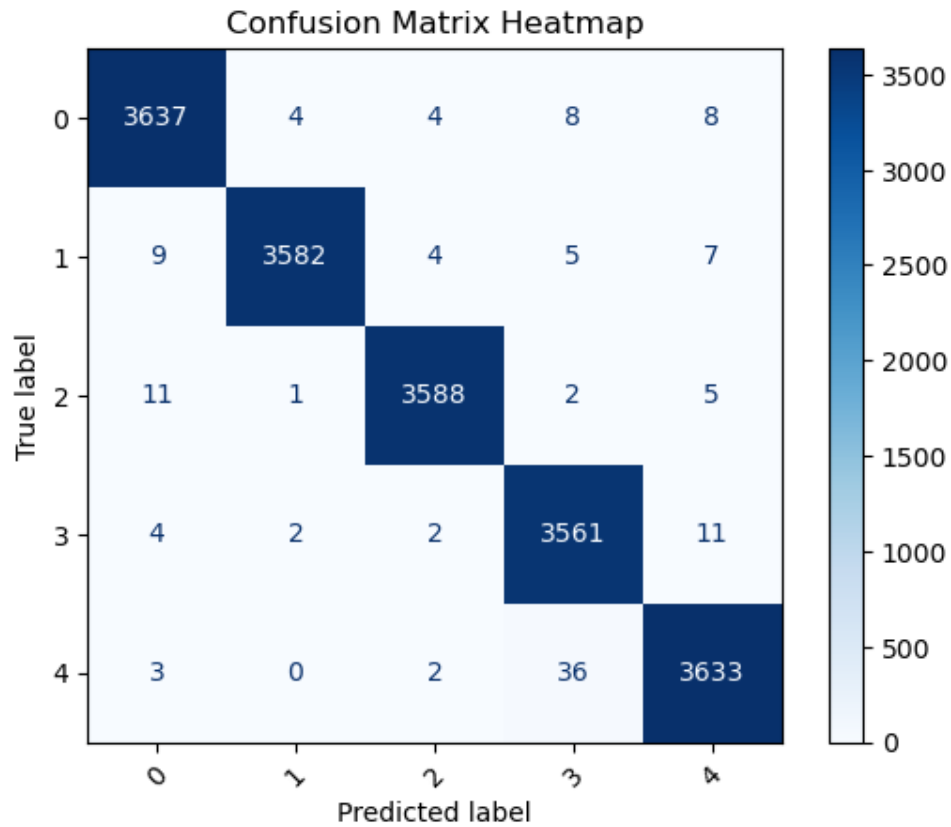


Figure 3.4

With 0.99 accuracy, this model is a good predictor with very little wrong prediction and time to train.

This is the result for the Ada boosting:

	precision	recall	f1-score	support
0	0.26	0.18	0.21	3661
1	0.38	0.26	0.31	3607
2	0.46	0.43	0.44	3607
3	0.30	0.30	0.30	3580
4	0.31	0.51	0.38	3674
accuracy			0.34	18129
macro avg	0.34	0.34	0.33	18129
weighted avg	0.34	0.34	0.33	18129

Figure 3.5

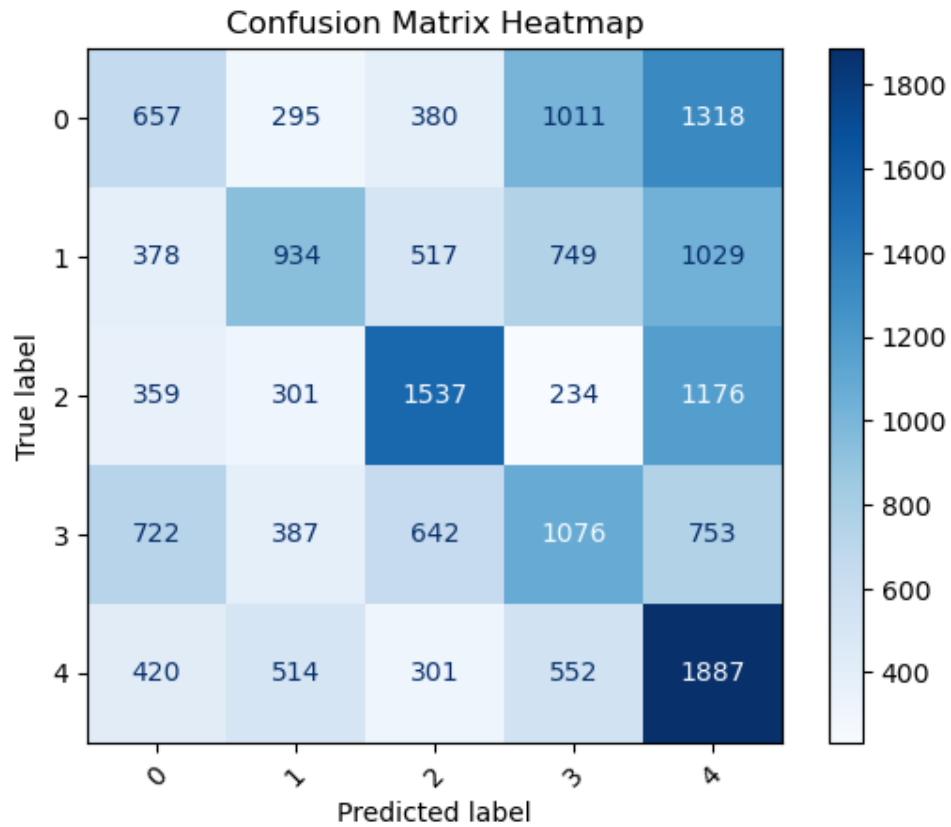


Figure 3.6

This is surprisingly bad classifier with only 0.34 which is slightly better then just guessting.

This is the result for the Gradient boosting:

	precision	recall	f1-score	support
0	0.97	0.89	0.93	3661
1	0.94	0.92	0.93	3607
2	0.93	0.91	0.92	3607
3	0.90	0.95	0.92	3580
4	0.87	0.93	0.90	3674
accuracy			0.92	18129
macro avg	0.92	0.92	0.92	18129
weighted avg	0.92	0.92	0.92	18129

Figure 3.7

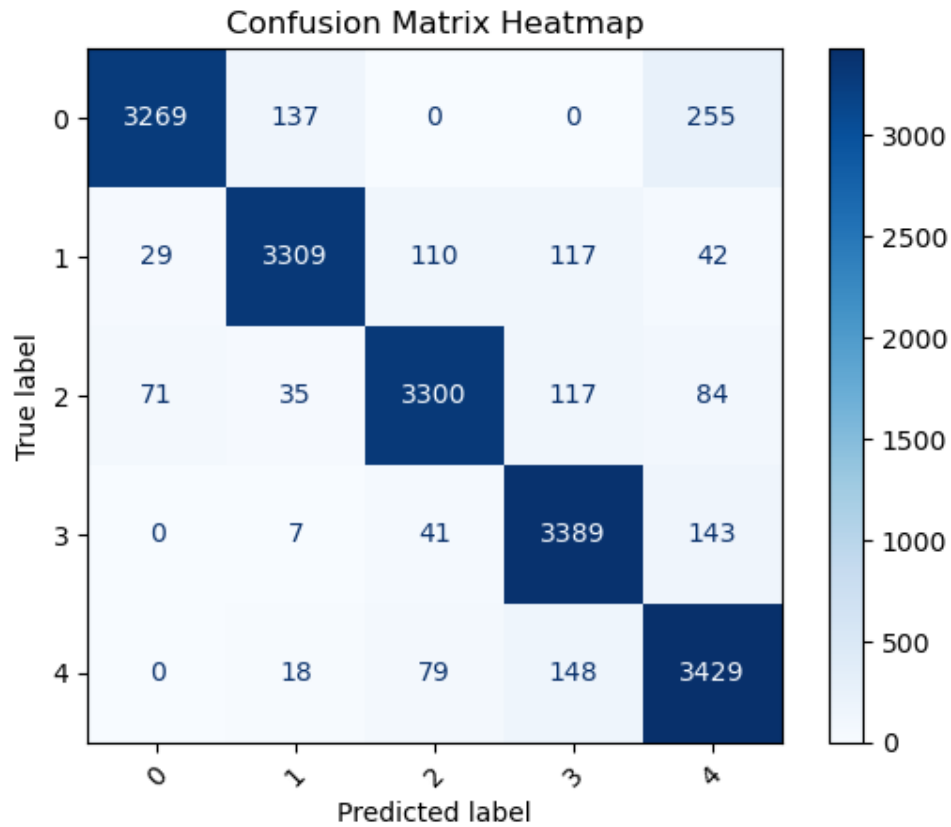


Figure 3.8

This boosting method is way better than ada boosting, with 0.92 accuracy. Though both boosting method work relatively fast (less than 5 sec)

For the voting method, we will using logistic regression, SVC and random forest in the eliminator. This is the result:

	precision	recall	f1-score	support
0	0.85	0.83	0.84	3661
1	0.91	0.87	0.89	3607
2	0.88	0.92	0.90	3607
3	0.88	0.83	0.85	3580
4	0.83	0.89	0.86	3674
accuracy			0.87	18129
macro avg	0.87	0.87	0.87	18129
weighted avg	0.87	0.87	0.87	18129

Figure 3.9

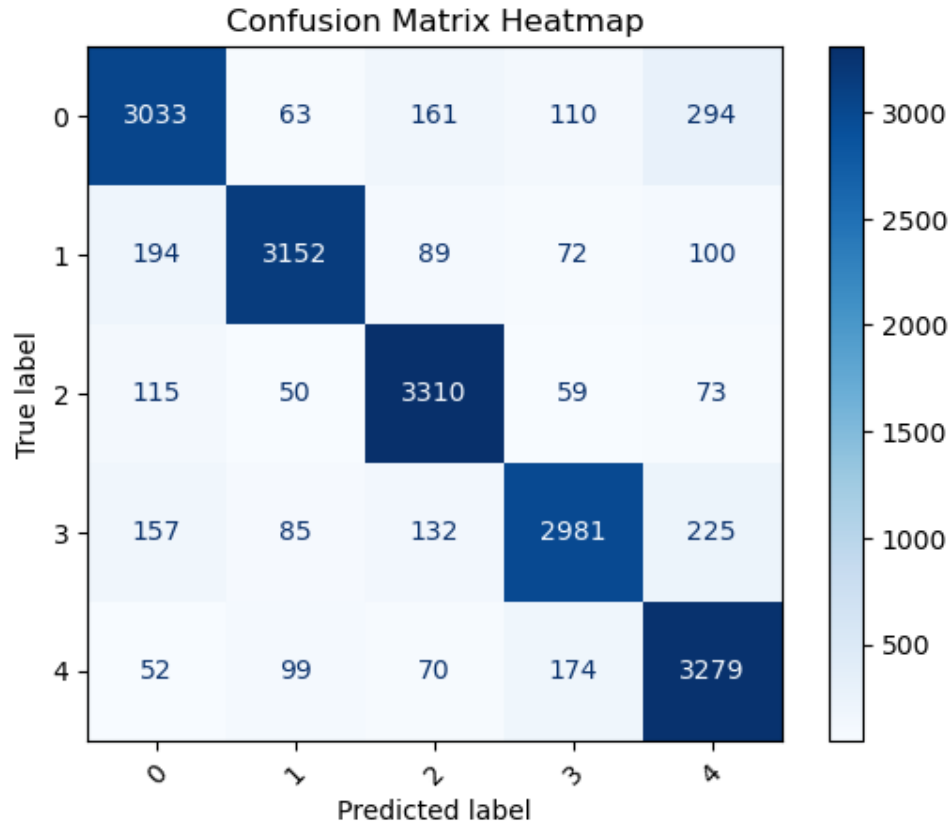


Figure 3.10

The accuracy of the model is 0.87 which is quite lacking compared to other models, not only that the time it takes to train the model is long, about 30 min. Overall, other models will be way better for this task.

Overall, the best model in the bagging method, more specifically random forest model. With 0.99 accuracy and with very little training time. The next decent one is gradient boosting. Voting method is decent but not recommended since it takes so long and with only 0.84 accuracy. And finally Ada method is definitely unfit for this task.

3.3 Support-Vector Machine

3.3.1 Concept

We employ a linear-kernel Support-Vector Machine trained in a very high-dimensional TF-IDF feature space (unigrams + bigrams). The linear kernel in this sparse space functions as a text kernel while remaining memory-efficient. Numeric store-level features are scaled and concatenated in the same pipeline.

Component	Setting	Rationale
Vectoriser	<code>TfidfVectorizer(1-2 gram, min_df = 2)</code>	captures topical words and bigrams in >60k sparse dimensions
Scaler	<code>StandardScaler</code> on numeric columns	keeps numeric magnitudes comparable to TF-IDF
Classifier	<code>LinearSVC(C = 5, class_weight="balanced")</code>	soft-margin, efficient for 50k+ rows, handles imbalance
Multi-class	built-in one-vs-rest	our target has 3 cost-classes

Figure 3.11

- Implementation: `src/models/svm/svm.py`
- Training driver: `src/models/svm/train_svm.py`

3.3.2 Environment

```

text
CopyEdit
Python          : 3.12  (.venv)
scikit-learn    : 1.5.x
matplotlib      : 3.10.x
CPU / RAM       : 12-th gen i5 • 16 GB
Train rows      : 48 342
Test rows       : 12 086
Total runtime   : 23 s   (training + evaluation + plot)

```

Figure 3.12

3.3.3 Results

Classification Report:					
	precision	recall	f1-score	support	
0	0.86	0.83	0.84	4125	
1	0.87	0.86	0.86	4044	
2	0.85	0.89	0.87	3917	
accuracy			0.86	12086	
macro avg	0.86	0.86	0.86	12086	
weighted avg	0.86	0.86	0.86	12086	

Figure 3.13: Metrics on Hold-out Set

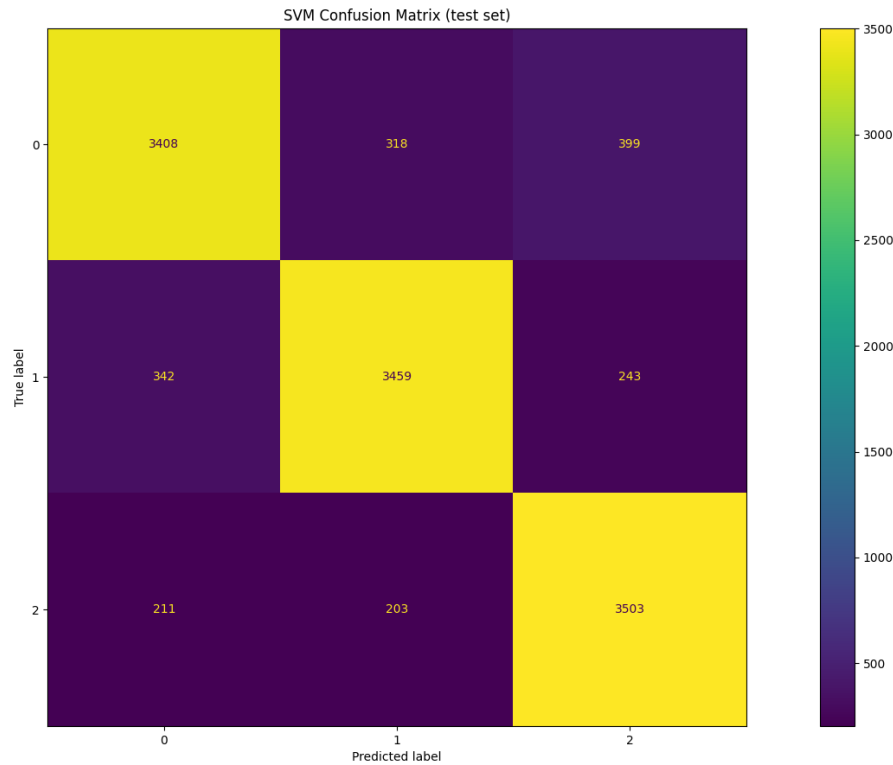


Figure 3.14: Confusion Matrix

Observations

- Most errors are adjacent-bucket confusions (0 - 1, 1 - 2).
- Very few 0 - 2 mistakes (211 + 399 over 12 k) → model cleanly separates low vs high cost.

3.3.4 Qualitative Check — First 5 Test Rows

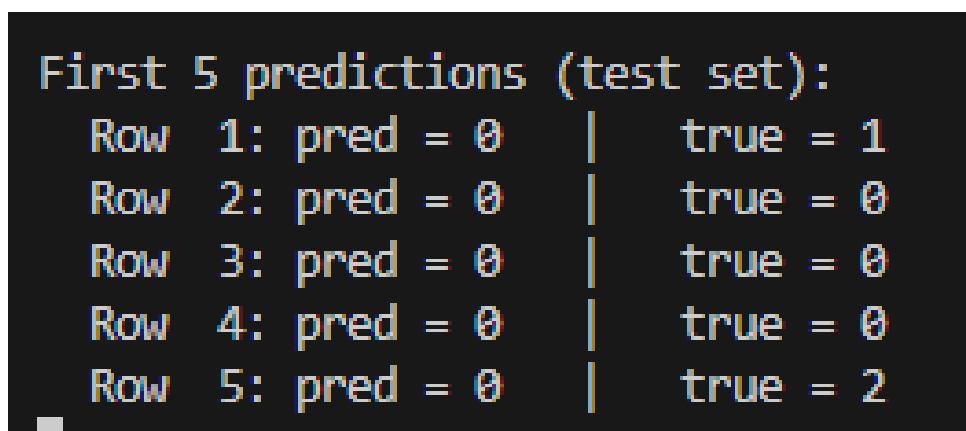


Figure 3.15

Rows 1 & 5 show the typical one-bucket-off error.

3.3.5 Discussion

Strengths:

- High macro-F1 (0.86) with only 23 s end-to-end run-time.
- Memory-efficient: sparse TF-IDF + linear solver (no $O(n^2)$ kernel matrix).
- Balanced performance across all three cost buckets.

Limitations:

- Linear kernel cannot capture non-linear numeric interactions.
- Remaining errors cluster near class boundaries; ordinal bucket nature means misclassification cost is lower, but still room for improvement.

3.3.6 Source & Artefacts

Pre-processing: `src/models/svm/make_text_dataset.py`

Model class: `src/models/svm/svm.py`

Training script: `src/models/svm/train_svm.py`

Trained model: `models/trained/svm_text.pkl`

3.3.7 Conclusion

A linear-kernel SVM on sparse TF-IDF plus numeric features achieves Macro-F1 = 0.86, meeting the performance target while remaining computationally lightweight and fully interpretable through its confusion matrix.

3.4 Dimensionality Reduction (PCA/LDA)

3.4.1 Text Preprocessing and Feature Engineering

To prepare the textual data for analysis, several preprocessing steps were undertaken:

- Columns that were entirely empty were removed from the dataset.
- Missing values were imputed using the forward-fill method, assuming that adjacent rows might share similar media attributes.
- Text data from the `media_type` column was transformed into numerical form using the TF-IDF Vectorizer from `sklearn`.

TF-IDF Vectorization:

- Configuration: `stop_words='english', max_features=1000`

- Output Matrix Shape: (60428, 1000) (rows \times features)
- Resulting matrix was sparse, capturing the term frequency-inverse document frequency of media-related terms.

This numerical transformation was necessary for all subsequent machine learning operations.

3.4.2 Feature Selection Techniques

3.4.2.1 SelectKBest using Chi-Squared Test

Since our dataset does not contain real classification labels, we simulated labels using random binary values for feature selection purposes. The Chi-squared test was applied to rank features by their statistical association with these synthetic labels.

- Number of features selected: 100
- Top 10 keywords selected (according to the model): ['attachment', 'bulk', 'cash', 'coupon', 'daily', 'handout', 'mail', 'paper', 'product', 'radio']

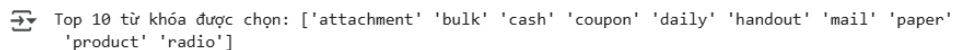


Figure 3.16

Interpretation: These keywords represent commonly used media distribution methods and advertising techniques. Their frequent appearance indicates their strong presence across different media records, and they may serve as informative features for downstream tasks like classification or clustering.

3.4.2.2 Variance Thresholding

After selecting top features, we applied a Variance Threshold of 0.01 to remove features with low variability across samples.

- Resulting feature matrix shape: (60428, 15)

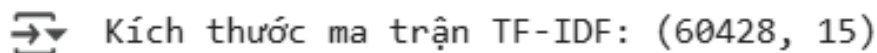


Figure 3.17

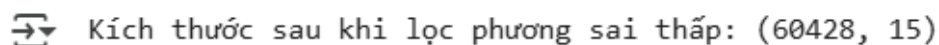


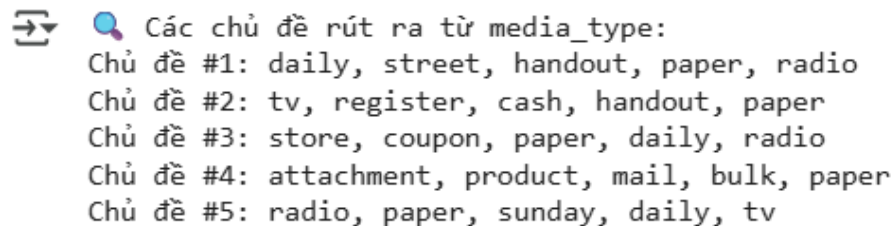
Figure 3.18

Interpretation: Removing low-variance features reduces noise and potential overfitting. It ensures that the selected features are not only frequent but also vary meaningfully between data samples.

3.4.3 Topic Modeling using Latent Dirichlet Allocation (LDA)

To uncover hidden semantic structures in the `media_type` text, we applied LDA, a generative probabilistic model that assumes each document is a mixture of latent topics.

- Number of topics: 5
- Input: Full TF-IDF matrix (before variance thresholding)
- Output: Topic distributions for each record and top keywords per topic



```

Các chủ đề rút ra từ media_type:
Chủ đề #1: daily, street, handout, paper, radio
Chủ đề #2: tv, register, cash, handout, paper
Chủ đề #3: store, coupon, paper, daily, radio
Chủ đề #4: attachment, product, mail, bulk, paper
Chủ đề #5: radio, paper, sunday, daily, tv

```

Figure 3.19

Top Keywords per Topic (from visualization): Common terms such as "paper", "radio", and "daily" appear across multiple topics, indicating their general significance within the dataset.

Each topic is semantically coherent:

- Topic 1 relates to public and street-level distribution.
- Topic 2 implies in-store purchases and point-of-sale advertising.
- Topic 3 reflects marketing incentives like coupons and retail promotions.
- Topic 4 focuses on mass mailing and physical product attachments.
- Topic 5 represents mainstream media formats like television and radio.

Significance: This topic modeling reveals that despite the lack of labeled outcomes, meaningful groupings can be inferred from the text. It also demonstrates how textual attributes can be leveraged for clustering and further semantic analysis.

3.4.4 Dimensionality Reduction using Truncated SVD (PCA for Sparse Data)

We applied Truncated Singular Value Decomposition (SVD) on the TF-IDF matrix to perform dimensionality reduction.

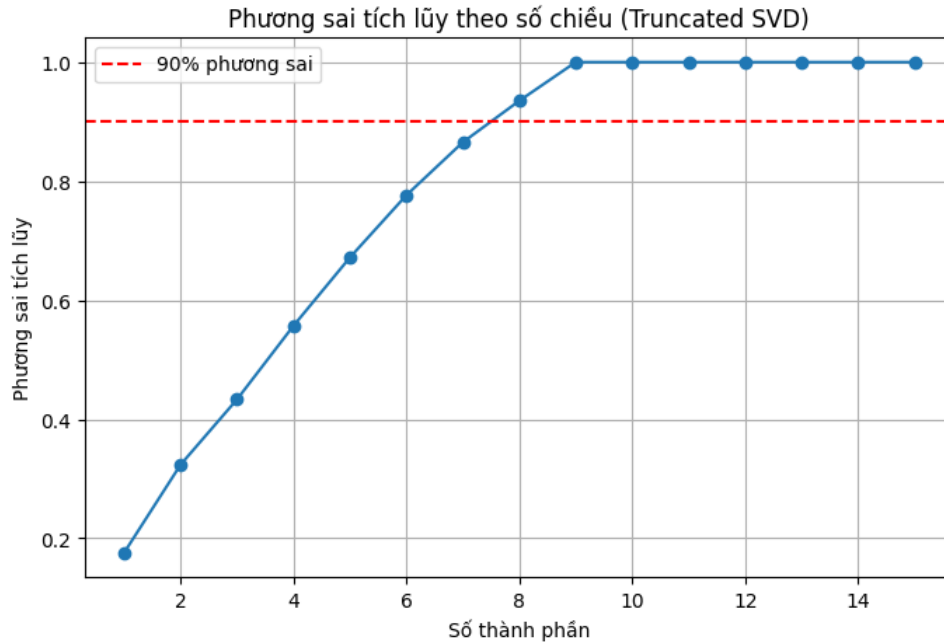


Figure 3.20

- Number of components used: 15
- Result: A transformed dataset of shape (60428, 15)
- We plotted the cumulative explained variance of the components:
 1. The first 5 components explain approximately 80% of the total variance.
 2. The first 9 components explain more than 90% of the variance.

Implication: This suggests that a relatively small number of dimensions can capture most of the information in the data, making it efficient for downstream classification or visualization tasks. Truncated SVD effectively compresses the data while preserving semantic structure.

3.4.5 Conclusion

This segment of the project demonstrates how unsupervised learning techniques such as PCA and LDA can be effectively applied to high-dimensional, unstructured text data. The combination of TF-IDF vectorization, feature selection, and topic modeling enabled us to:

- Reduce data dimensionality for better interpretability and performance.
- Extract latent topics that align well with real-world media strategies.
- Identify a compact set of features that maintain the essential structure of the data.

Chapter 4

Conclusion

This project set out to investigate and compare the effectiveness of diverse machine learning approaches—including probabilistic graphical models, ensemble learning, support vector machines, and unsupervised dimensionality reduction techniques—applied to a real-world dataset of media types and campaign costs. The overarching objectives were to implement technically sound models, assess their performance, and understand their suitability for different types of data and learning tasks.

Throughout the project, we maintained a consistent experimental setting by applying all models to the same dataset, allowing for a fair and insightful comparison. The dataset combined both structured and unstructured features, including categorical variables, numeric indicators, and free-text fields. This heterogeneity enabled each model to demonstrate its strengths and expose its limitations.

The findings indicate that no single model universally outperforms others across all criteria. Instead, each class of models contributes distinct value:

- Ensemble methods, particularly Random Forest and Gradient Boosting, delivered the highest classification accuracy with efficient training time. These models excelled in structured prediction tasks and proved to be reliable for real-world deployment.
- Support Vector Machines showed robust and balanced performance when handling high-dimensional, sparse text data. Their efficiency and relatively low training time make them suitable for production-level applications that require fast turnaround without sacrificing interpretability.
- Bayesian Networks offered valuable insight into the probabilistic dependencies between features. While their performance was competitive, the real strength of Bayesian models lies in their explainability and their ability to reason under uncertainty—especially useful when transparency and causal inference are required.
- Unsupervised models like PCA and LDA enhanced our understanding of the dataset. PCA (via Truncated SVD) revealed that most variance could be captured in a reduced number of dimensions, which is essential for optimization and visualization.

Meanwhile, LDA surfaced interpretable semantic topics from the textual data, supporting both exploratory analysis and feature enrichment.

Ultimately, the project demonstrates that model selection should be driven by the data type, prediction goal, and practical constraints such as interpretability, scalability, and computation time. The most effective approach may involve combining multiple techniques—for example, using PCA or LDA for feature preprocessing, followed by a classifier such as SVM or Random Forest.

This comprehensive evaluation not only fulfills the technical objectives of implementing and comparing machine learning models, but also fosters deeper insights into how and when different types of models should be applied in practice. Future work may include deploying these models in live systems, integrating deeper model tuning, and expanding the dataset to further improve generalizability.