Introduction to Data Science
WS25/26
Assignment Part 2

Report

Group Members:

Zexiao Xu (405919)
Yiwen Yang (422686)
Haoran Zhang(441340)

January 21, 2026

Contents

Question 1: Market Basket Analysis

(a) Product Frequencies

A support of at least 1% corresponds to a product being ordered in at least 648.64 orders. The mean support count across all products is 21.59, while the median support count is 4, indicating that most products occur in only very few orders.

(b) Most Frequent Products

The ten most frequent product IDs together with their absolute and relative frequencies are shown in Table 1. The maximum support that an itemset of size three can achieve is 0.08448, corresponding to the products with IDs 24852, 13176, and 21137.

Table 1: *Top-10 most frequent products with absolute and relative support.*

Product ID	Absolute Support	Relative Support
24852	9376	0.14455
13176	7701	0.11873
21137	5480	0.08448
21903	4818	0.07428
47626	4096	0.06315
47766	3670	0.05658
47209	3605	0.05558
16797	3199	0.04932
26209	3017	0.04651
27966	2728	0.04206

(c) Co-occurrence of the Size-3 Itemset

The itemset consisting of products *Banana*, *Bag of Organic Bananas*, and *Organic Strawberries* occurs in only **3** from all orders despite the high individual frequencies. This itemset is rare not because the products are unpopular, but because: Market-basket logic is about complements, not similarity. And strong rules would usually link different-use items (e.g., bananas $\rightarrow$ yogurt)

(d) Product Categories

Figure 1 shows the total number of sold products per category. Figure 2 shows the mean and median number of items per order for each category, with categories ordered alphabetically.

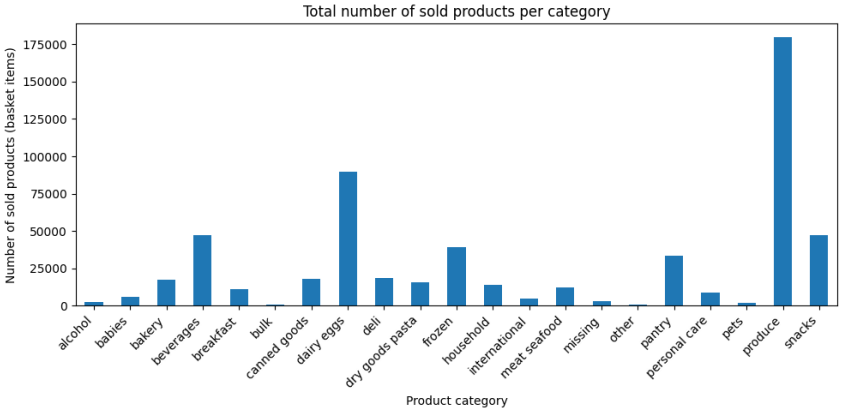

Figure 1: *Total number of sold products per category.*

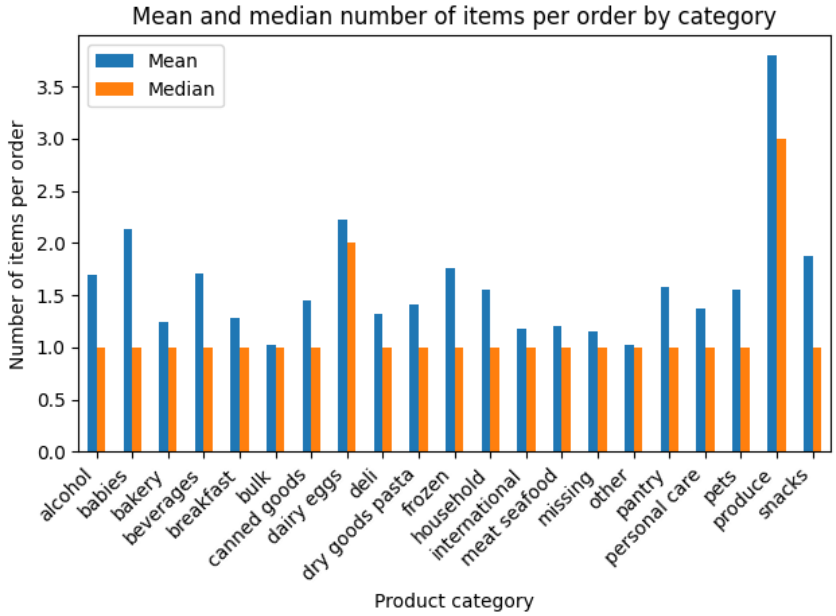

Figure 2: *Mean and median number of items per order by product category.*

(e) Beverage Purchase Behavior

For beverages, the mean number of items per order is noticeably higher than the median, indicating a right-skewed distribution. This suggests that while most orders contain only one beverage item, a smaller number of orders include multiple beverages, consistent with bulk purchases.

(f) Frequent Itemsets

The largest frequent itemset contains four products:

- Bag of Organic Bananas
- Organic Strawberries
- Organic Hass Avocado

- Organic Raspberries

, There are 1 017 frequent itemsets of size two or larger, corresponding to approximately 51.74% of all frequent itemsets. The product Banana appears in the most frequent itemsets, occurring in 168 frequent itemsets. while 794 products appear in only a single frequent itemset.

(g) Category Sets of Frequent Itemsets

Figure 3 shows the frequency of category sets derived from frequent itemsets of size two or larger, ordered by descending frequency.

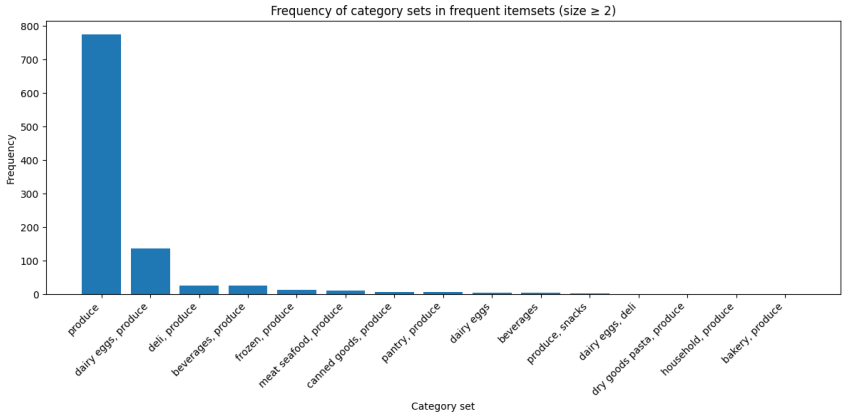

Figure 3: *Frequency of category sets for frequent itemsets.*

(h) Interpretation of Category Set Observations

Observation 1: This is not unexpected because produce items have the highest individual support and are purchased very frequently across many orders. Since frequent itemsets are constrained by minimum support, combinations of high-support produce items are much more likely to remain frequent than combinations involving less frequently purchased categories.

Observation 2: Produce products appear in most orders and co-occur with many other product categories, which makes them natural "connectors" in frequent itemsets. As a result, they survive the support threshold even when combined with less frequent products, explaining their near-ubiquitous presence.

Observation 3: This is not surprising because beverage purchases, while less common overall, often involve buying multiple beverage items together (e.g., several drinks at once), leading to strong co-occurrence within that category. This implies that the sold beverages exhibit high intra-category diversity, with multiple beverage products being frequently bought together even if beverages are absent from many other orders.

(i) Association Rules

Across all association rules, the mean lift is 3.95 and the median lift is 2.83. 228 of rules contain at least one of the top-10 most frequent products in the antecedent while 430 in the consequent.

(j) Rule with Highest Confidence

$$\{\text{Organic Hass Avocado}, \text{Organic Strawberries}, \text{Organic Raspberries}\} \Rightarrow \{\text{Bag of Organic Bananas}\}$$

Interpretation of the highest-confidence rule The association rule has the highest confidence, meaning that when the products in the antecedent are purchased, the consequent product is bought with very high probability. This indicates a strong conditional relationship, although it does not necessarily imply a strong increase in purchase likelihood beyond the product's overall popularity.

Why the most frequent product appears as the consequent It is not unexpected that the most frequent product appears as the consequent because highly popular products already occur in a large fraction of orders, which naturally leads to high conditional probabilities. Placing such a product in the antecedent would instead restrict the rule to many orders where the consequent is not present, lowering confidence. Therefore, frequent products tend to maximize confidence when used as consequents rather than antecedents.

(k) Rules Without Top-10 Products

The association rules that do not contain any of the top-10 most frequent products have a dramatically higher lift (mean: 10.90) than the overall set of rules (mean: 2.83), with both mean and median lift being an order of magnitude larger. This is not surprising because rules involving less frequent products often capture strong, specific co-occurrence patterns, which leads to high lift despite lower absolute support. In contrast, highly frequent products tend to co-occur with many items by chance, which inflates confidence but suppresses lift.

(l) Bi-directional Sequential Rules

Table 2 reports the support counts for the ordered subsequences $\langle a, b\rangle$ and $\langle b, a\rangle$ for all bi-directional association rules.

Table 2: *Support counts for ordered subsequences of bi-directional rules.*

Product a	Product b	Support$_{\langle a,b\rangle}$	Support$_{\langle b,a\rangle}$
4957	33754	53	57
28465	36865	59	61
33754	33787	69	47
13176	21137	1149	404

(m) Impossibility of Lift Below One

Lift is defined as $\text{lift}(A \Rightarrow B) = \frac{\text{confidence}(A \Rightarrow B)}{\text{support}(B)}$. From task (b), we know that no product in the dataset has a support larger than 0.2. Given the constraint $\text{confidence}(A \Rightarrow B) > 0.2$, this implies that

$$\text{lift}(A \Rightarrow B) = \frac{\text{confidence}(A \Rightarrow B)}{\text{support}(B)} > 1$$

for all possible rules. Therefore, it is not possible to find an association rule that simultaneously satisfies a support count.

Question 2: Process Mining

(a) Process Discovery with the Inductive Miner

(a1) With which activities can the process start? Based on the discovered Petri net, the process can only start with the activity *Create Fine*. This is the only visible transition enabled by the initial marking of the model.

(a2) Which activities must be executed in a trace containing an appeal to a judge and in what order? From the Petri net, it follows that every trace containing an *Appeal to Judge* must execute *Create Fine* beforehand. The model enforces the following causal order:

$$\text{Create Fine} \prec \text{Appeal to Judge}.$$

Apart from this ordering, the Petri net does not impose any additional mandatory activities that must occur in every trace containing an appeal to a judge. Other activities may occur optionally or in different orders.

(a3) Which activities can be executed more than once? The Petri net contains a loop on the payment branch, which allows the activity *Payment* to be executed multiple times within the same case.

(a4) Is it possible that the credit collection is involved without the fine being sent to the offender via post? Yes. The Petri net allows executions in which *Send for Credit Collection* is performed without the activity *Send Fine* having occurred beforehand, since *Send Fine* is not on every path leading to credit collection.

(b) Behavior Allowed by the Model but Rare in the Log

According to our results, there are

- 1013 cases with Payment eventually followed by Send for Credit Collection
- 0 cases with Add penalty eventually followed by Send Fine
- 10 cases with Payment eventually followed by (Prefecture-appeal activity OR Appeal to Judge)
- 5 cases with Notify Result but NO appeal action

(c) Variant Analysis

Figure 4 shows the cumulative variant frequency. A very small number of variants already covers the majority of cases, as the curve rises steeply at the end. This indicates that the process is highly structured, with a few dominant execution patterns and many rare variants.
The five most frequent variants and their case counts are:

1. **26,872 cases:**
   Create Fine → Send Fine
   → Insert Fine Notification
   → Add Penalty
   → Send for Credit Collection
2. **22,078 cases:**
   Create Fine → Payment
3. **9,631 cases:**
   Create Fine → Send Fine
4. **4,507 cases:**
   Create Fine → Send Fine → Insert Fine Notification → Add Penalty → Payment
5. **1,806 cases:**
   Create Fine → Send Fine → Insert Fine Notification

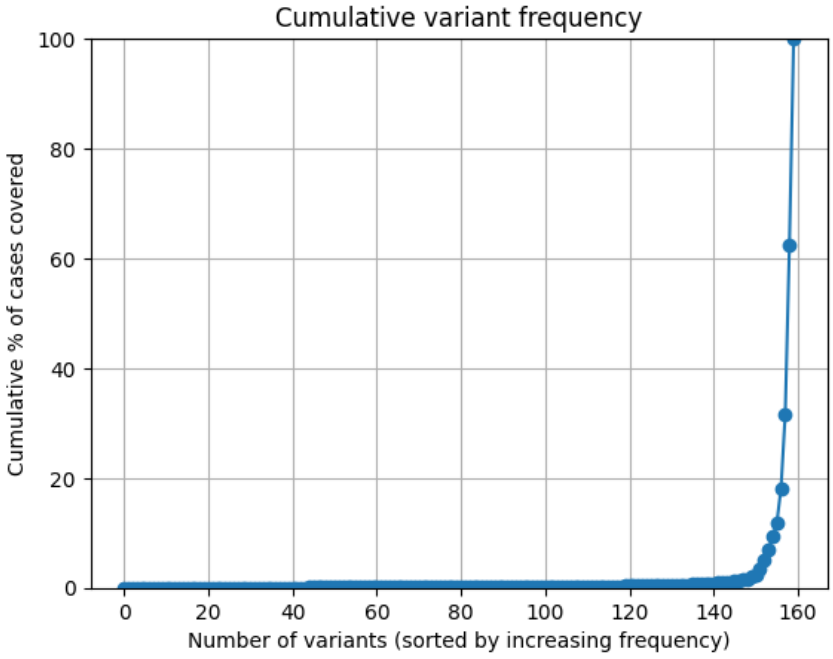

Figure 4: *Cumulative variant frequency of the event log.*

(d) Case Outcomes

Figure 5 shows the distribution of case outcomes. Most cases are closed either by full payment or by being sent to credit collection, while only a small fraction of cases are dismissed or remain open.

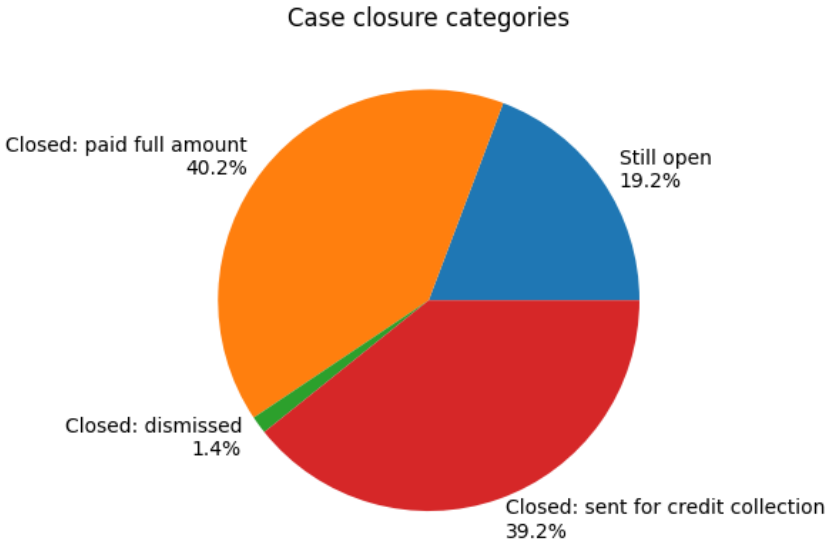

Figure 5: *Distribution of case outcomes.*

(e) Filtered Model of Closed Cases

Compared to the Petri net discovered from the full event log (part a)), the model based on the five most frequent closed-case variants is substantially simpler and more structured. Only the core activities of the fine-handling process are retained, while many infrequent and exceptional activities, in particular those related to appeals, are absent. Payments occur in a more clearly defined and consistent position late in the process, whereas in the full-log model they can appear at multiple stages. Moreover, the relation between making a payment and sending for credit collection is tighter: credit collection is mostly an end-of-process step in the filtered model, while in the full model it can be reached through several alternative paths.

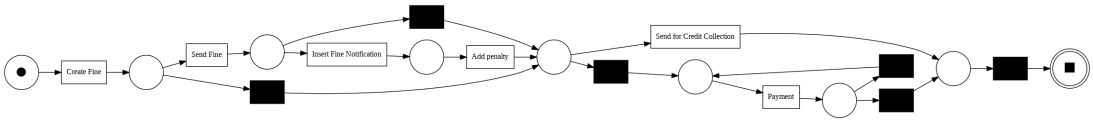

Figure 6: *Petri net discovered from the filtered log of closed cases.*

(f) Conformance Checking

Applying token-based replay of the full log on the model discovered from the filtered log results in 80.26% perfectly fitting traces and a log fitness of 0.97. The fitness value is substantially higher than the percentage of perfectly fitting traces because many traces deviate only slightly from the model and require few missing or remaining tokens, which still yields a high overall fitness score.

Question 3: Natural Language Processing

(a) Word Frequencies Without Preprocessing

Figure 7 shows the 25 most frequent words obtained by splitting the text by whitespace without any preprocessing. The histogram is dominated by function words such as "the", "is", "to", and "and", which occur frequently but carry little semantic meaning.
Two main problems are visible: first, the approach is case-sensitive, resulting in duplicate counts for words such as "and" and "And"; second, frequent function words obscure more informative content words. Both issues can be mitigated by normalizing case and removing stop words.

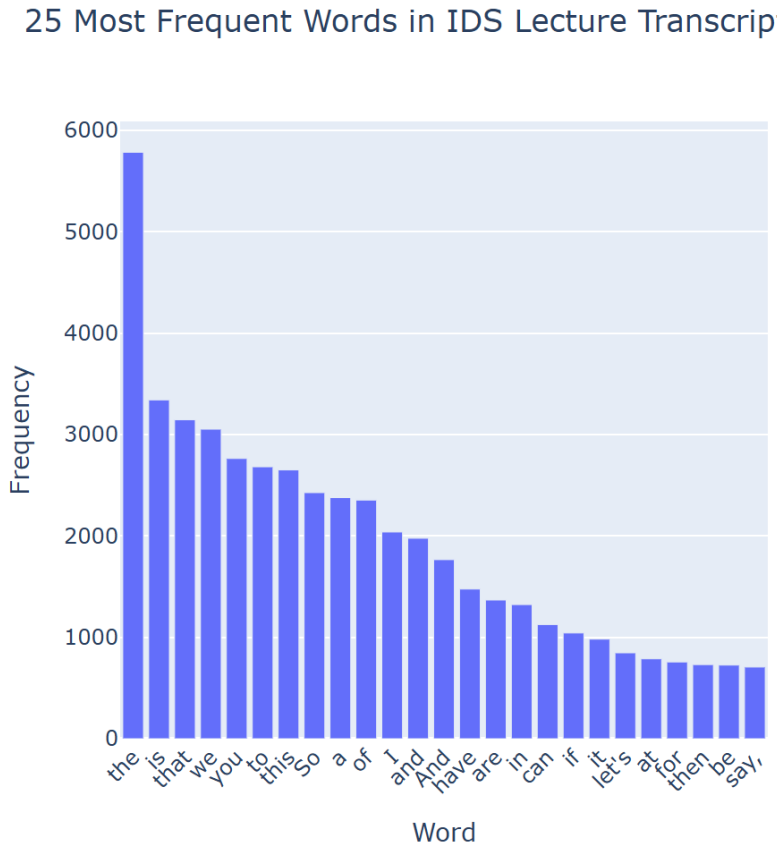

Figure 7: *25 most frequent words without preprocessing.*

(b) Word Frequencies After Preprocessing

After lowercasing, removing punctuation, tokenizing, and filtering stop words, Figure 8 shows the 25 most frequent tokens in the lecture "01-introduction". Compared to part (a), the remaining tokens are more content-bearing, such as "data", "lecture", "course", and "topics", making the distribution more informative for analysis.

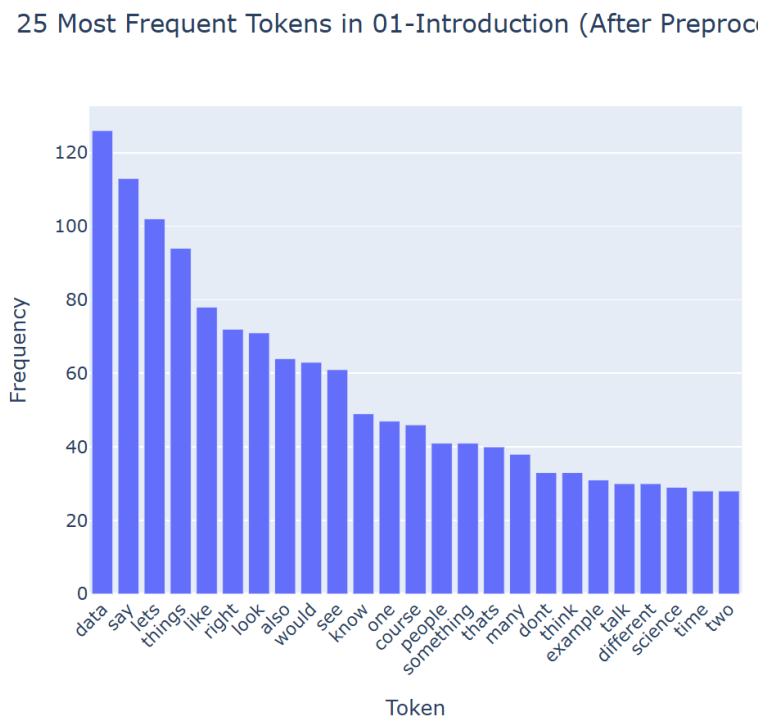

Figure 8: *25 most frequent tokens in "01-introduction" after preprocessing.*

(c) Token Frequencies Across Lectures

Figure 9 shows the stacked frequencies of selected tokens across all lectures. The token "data" appears frequently in almost every lecture, indicating that data-related concepts are central to the course. Tokens such as "decision" and "predict" are concentrated in lectures on decision trees and regression, while "derivative" and "network" mainly occur in the neural networks lectures. More specific terms such as "database" appear almost exclusively in the corresponding lecture, reflecting their specialized nature.

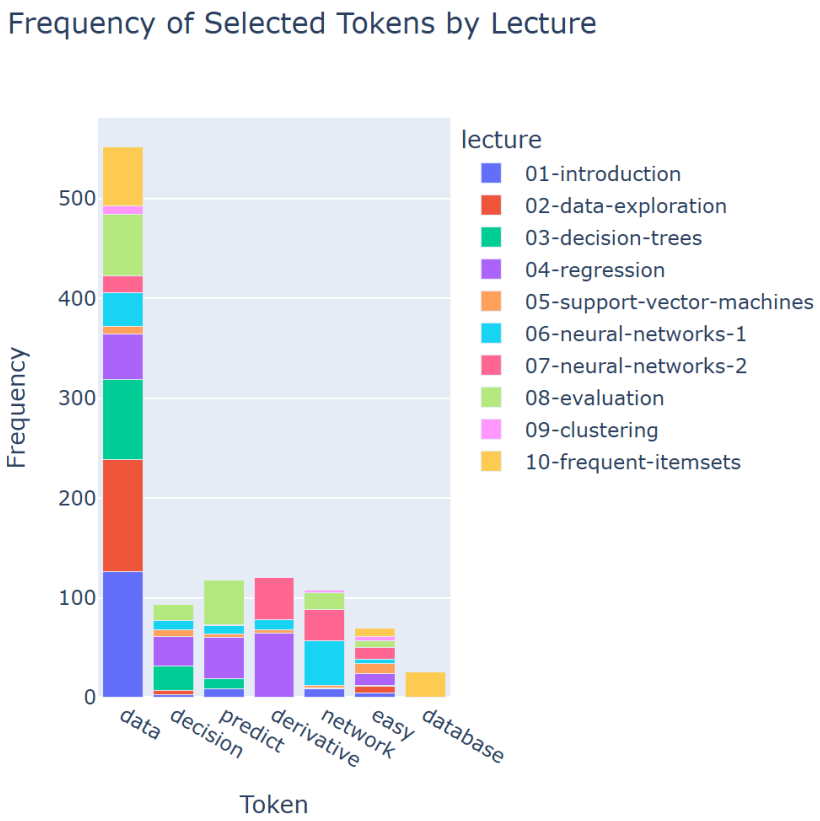

Figure 9: *Frequency of selected tokens by lecture (stacked).*

(d) n-gram Language Models

Using an n-gram language model trained on the transcript text, the following text was generated for the seed phrase "introduction to data":

n = 2: `introduction to data set on the or regression model prediction model prediction model prediction model prediction model prediction`

n = 3: `introduction to data science master and its not shown so again this other one is not possible and so on the part that is possible`

n = 4: `introduction to data science so this lecture will be a lecture on regression they will be provided to you so the question`

n = 5: `introduction to data`

For $n = 2$, the generated text is highly repetitive and quickly falls into loops because the model conditions on only one previous word. For $n = 3$ and $n = 4$, the text becomes more coherent and lecture-like, as additional context improves prediction. For $n = 5$, generation stops almost immediately because longer contexts are rare in the dataset, leading to unseen contexts.

(e) Hierarchical TF-IDF Retrieval

Query: Gradient Descent Approach

- **Lecture 04 – Regression** (lecture score: 0.0756)
  - Segment [3324.14, 3353.17] (score: 0.2849): The decision boundary induced by certain models is not smooth or continuous, which leads to a non-continuous error surface. As a result, standard gradient descent cannot be directly applied without modifications.
  - Segment [1907.18, 1935.92] (score: 0.2806): Fixing one parameter (e.g., w_1) guarantees the existence of at least one optimal value for another parameter (e.g., w_0). However, a very flat error surface can result in extremely small gradients.
- **Lecture 06 – Neural Networks I** (lecture score: 0.0301)
  - Segment [2272.58, 2301.78] (score: 0.4496): The derivative of the sigmoid function, $\sigma(x)(1 - \sigma(x))$, has a convenient analytical form, which makes it well suited for gradient descent optimization.
  - Segment [1921.86, 1952.67] (score: 0.3124): Gradient descent is generalized to large-scale models such as neural networks, where weight updates are expressed using more compact and mathematical notation.

Comment: The retrieval correctly ranks 04-regression highest, and the top segments explicitly discuss why plain gradient descent fails on a non-smooth/non-continuous error surface and then introduce how to modify the approach. The second-best lecture (06-neural-networks-1) is also a good match: its top segment connects gradient descent to having a "nice" derivative (sigmoid derivative) and weight updates, which fits the optimization context

Query: Beer and Diapers

- **Lecture 10 – Frequent Itemsets** (lecture score: 0.0285)
  - Segment [1521.46, 1549.01] (score: 0.4175): Adding items to an itemset cannot increase its frequency; support is monotonic and always decreases or remains unchanged when the itemset grows.
  - Segment [857.25, 886.69] (score: 0.3388): Transaction data is modeled as a multiset of itemsets, where only the presence or absence of items matters, not their quantities.
- **Lecture 01 – Introduction** (lecture score: 0.0030)
  - Segment [3881.79, 3904.34] (score: 0.2300): Pattern mining aims to discover meaningful and potentially surprising co-occurrence patterns in customer purchase behavior, such as customers buying beer also buying diapers.

Comment: The top lecture (10-frequent-itemsets) is exactly the expected match: both top segments directly mention the classic "beer and diapers" co-occurrence example and relate it to itemsets and frequency monotonicity. The second match (01-introduction) is much weaker and more general—talking about pattern mining and "people who buy beer also buy chips"—which explains the very low lecture score and why it's ranked far below the frequent-itemsets lecture.

Question 4: Time Series Analysis

(a) Exploring the data

(a.i) Dataset overview

We load `time-series/air_traffic.csv`. The dataset spans from **January 2003** to **September 2023**. In total, **249 months** are tracked, and the overall number of flights (sum of `Flt` over the full dataset) is 1.921002×10^8.

(a.ii) Time series plot: flights per month

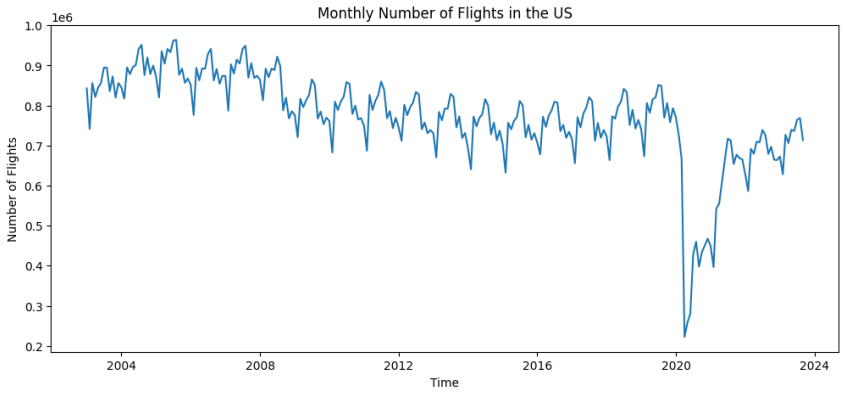

Figure 10: *Monthly overall number of flights (`Flt`).*

The monthly number of flights shows a clear repeating seasonal pattern with yearly peaks and troughs. From 2003 to around 2008 there is a slight increase, followed by a gradual decline and then a relatively stable period. A sharp drop appears in early 2020 (COVID-19), after which flights recover but remain below pre-pandemic levels by the end of the observation window.

(a.iii) Passengers vs. flights

Figure 11: *Comparison of passengers (`Pax`) and flights (`Flt`) over time.*

The number of flights and the number of passengers follow similar long-term and seasonal trends, indicating a strong correlation between the two variables. However, passenger counts show larger relative fluctuations, especially during the COVID-19 period, where

passenger numbers dropped more sharply than flights. This suggests that while the number of flights was reduced, aircraft occupancy decreased even more significantly.

(b) In-depth time series analysis

(b.i) Yearly seasonal plots

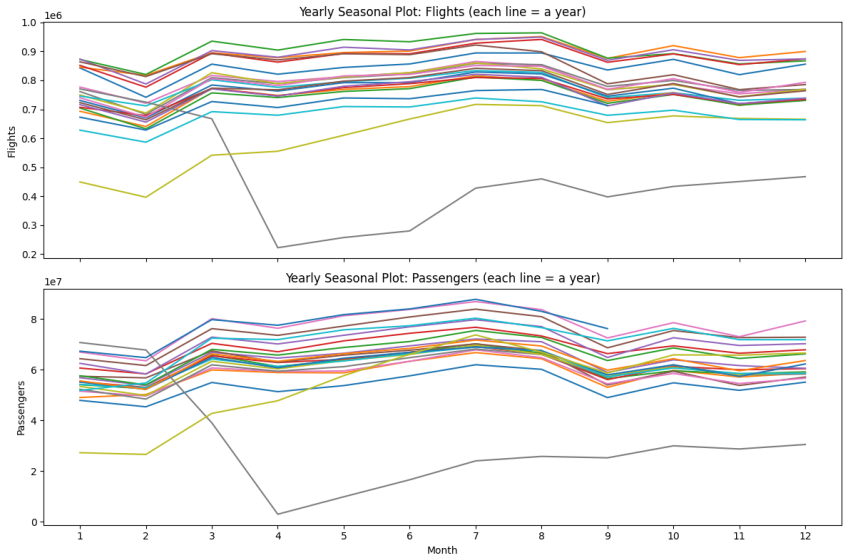

Figure 12: *Yearly seasonal plots for flights (`Flt`) and passengers (`Pax`).*

The yearly seasonal plots show a strong and consistent seasonal pattern for both flights and passengers, with lowest values typically occurring in winter months and peaks during the summer (around July–August). This seasonal structure is stable across most years, indicating persistent annual travel behavior. A clear exception is the year **2020** (and partially 2021), where the usual seasonal pattern is strongly disrupted due to the COVID-19 pandemic.

(b.ii) Correlation between passengers and flights

The correlation coefficient between `Pax` and `Flt` is

$$\rho(\texttt{Pax}, \texttt{Flt}) \approx 0.57,$$

indicating a *moderate positive* relationship. Months with more flights tend to have more passengers, but the relationship is not perfectly linear. This matches the visual comparison in (a.iii): both series move together most of the time, yet they diverge during structural breaks such as the COVID-19 period.

(b.iii) Correlograms with differencing $d \in \{0, 1, 2\}$ (lags up to 24)

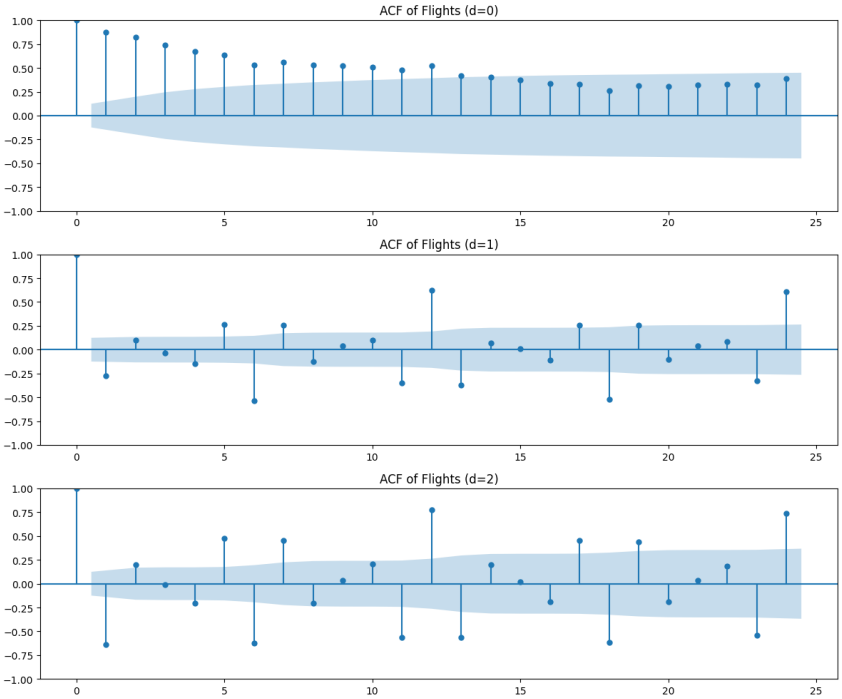

Figure 13: *Correlograms for flights with 0/1/2 differencing steps (lags up to 24).*

Without differencing ($d = 0$), the ACF decays slowly and many lags are significant, suggesting non-stationarity and strong seasonal structure. After first differencing ($d = 1$), short-term autocorrelation is reduced, but seasonal spikes remain. After second differencing ($d = 2$), most autocorrelations are further reduced, although seasonal effects are still visible. In all correlograms, the most significant non-zero autocorrelation occurs at **lag 12** (also visible at **lag 24**), reflecting strong annual seasonality in monthly flights.

(b.iv) STL decomposition (period 12) and residual stationarity

The STL residual fluctuates around zero without a clear trend, which suggests approximate stationarity. However, there are large outliers during the COVID-19 period, indicating that the residual distribution is affected by extreme shocks/structural breaks.

(c) Forecasting

(c.i) Suitability for forecasting

Both the flights and passengers time series are suitable for forecasting because they exhibit clear temporal structure, including long-term trends and strong annual seasonality. However, structural breaks (most notably during COVID-19) introduce non-stationarity and make forecasting harder; models should account for both trend and seasonality to perform well.

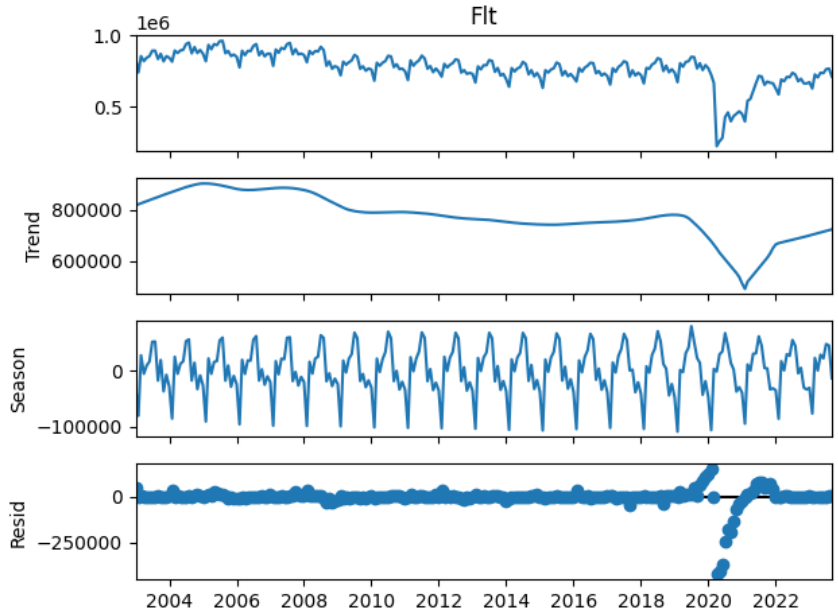

Figure 14: *STL decomposition of flights with period 12 (trend, seasonal, residual).*

(c.ii) Naive vs. ARIMA forecasting performance

We fit a `NaiveForecaster` (mean strategy) and a `StatsModelsARIMA` model in `sktime` to forecast `Flt`. The best ARIMA configuration found is:

$$(p, d, q) = (4, 2, 8).$$

Model	RMSE	MAE	MAPE
NaiveForecaster (mean)	51192.09	36663.86	0.049
ARIMA(4,2,8)	25099.49	19890.06	0.026

Table 3: *Forecasting performance on `flights_test.csv` (lower is better).*

The ARIMA(4,2,8) model outperforms the mean-baseline in all three metrics (RMSE, MAE, and MAPE).

(c.iii) Comment on the prediction plot

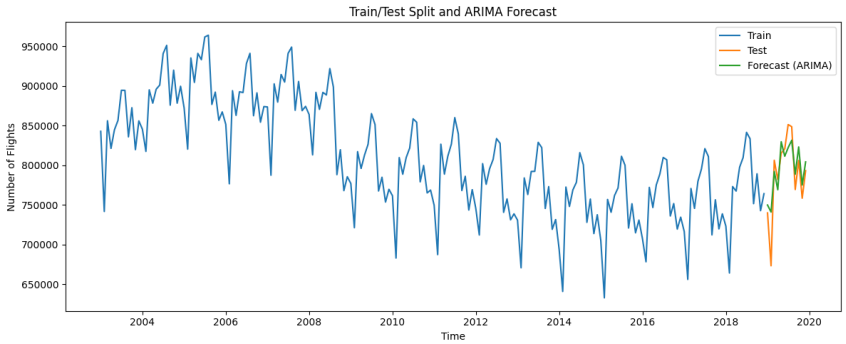

Figure 15: *Forecast visualization: ARIMA prediction vs. observed test flights.*

The ARIMA forecast captures the overall level and short-term dynamics of the flights series reasonably well, with predictions lying within the observed test range. It follows the general seasonal oscillations but smooths extreme fluctuations, slightly underestimating peaks and overestimating troughs. Overall, it improves over the naive baseline, but it still cannot fully capture variability during rapidly changing periods.

Question 5: Distributed Data Processing

(a) Computing a Directly Follows Graph with MapReduce

Input. Each CSV line (no header) encodes one event

$$(c, a, t) \in C \times A \times T,$$

and the raw MapReduce input is

$$K \times V = \mathbb{N}_0 \times S$$

Goal. Compute the Directly Follows Graph (DFG) as edge counts

$$\{((a, b), \text{count}_{a,b}) \mid a, b \in A\}.$$

Job 1: Group events by case and emit directly-follows pairs

Map_1.

$$\text{map}_1 : \mathbb{N}_0 \times S \to \mathcal{P}(C \times (A \times T))$$

$$\text{map}_1(n, \text{line}) = \{(c, (a, t))\}.$$

Explanation: Parse the CSV line into (c, a, t) and key the event by its case id c.

Reduce_1.

$$\text{reduce}_1 : C \times \mathcal{P}(A \times T) \to \mathcal{P}((A \times A) \times \mathbb{N}_0)$$

Conceptually, for each case c, sort events by timestamp to obtain $(a_{(1)}, \ldots, a_{(m)})$ and emit one count for each consecutive pair:

$$\text{reduce}_1(c, \{(a_i, t_i)\}) = \{((a_{(i)}, a_{(i+1)}), 1) \mid i = 1, \ldots, m-1\}.$$

Explanation: Within each case, convert the ordered trace into directly-follows edges.

Job 2: Aggregate counts of directly-follows pairs

Map_2.

$$\text{map}_2 : (A \times A) \times \mathbb{N}_0 \to \mathcal{P}((A \times A) \times \mathbb{N}_0)$$

$$\text{map}_2((a, b), 1) = \{((a, b), 1)\}.$$

Explanation: Identity mapper to forward the emitted edge occurrences.

Reduce$_2$.

$$\text{reduce}_2 : (A \times A) \times \mathcal{P}(\mathbb{N}_0) \to \mathcal{P}((A \times A) \times \mathbb{N}_0)$$

$$\text{reduce}_2((a,b), \{1, \ldots, 1\}) = \{((a,b), \sum 1)\}.$$

Explanation: Sum up all occurrences to obtain the final DFG edge weights.

(b) Computing a DFG with Apache Spark

To compute the DFG using Apache Spark, the event log is loaded as an RDD and processed using a sequence of transformations.
First, `textFile` is used to load the dataset, followed by `map` to parse each line into structured event tuples. Next, `groupByKey` groups events by case id, and `mapValues` sorts the events of each case by timestamp and extracts directly-following activity pairs. The `flatMap` transformation emits all such pairs across cases. Finally, `reduceByKey` aggregates the counts for each activity pair.
Figure 16 shows the resulting DFG computed from the full event log.

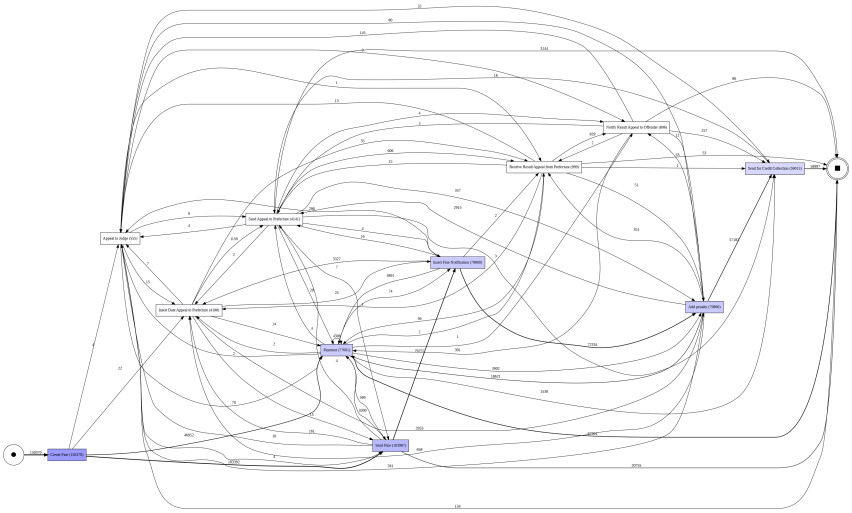

Figure 16: *Directly Follows Graph computed from the full event log using Apache Spark.*

(c) Filtering Low-Frequency Relations

(c.1) MapReduce Filtering

To remove arcs with a frequency below 100 in the MapReduce solution, the filtering can be applied after the aggregation step by discarding all (a, b) pairs whose count is less than 100.

(c.2) Spark Filtering

In the Spark implementation, this is achieved by applying a `filter` transformation after `reduceByKey`, keeping only directly-follows relations with a count of at least 100.

(c.3) Filtered DFG

Figure 17 shows the DFG after removing low-frequency arcs. The filtered graph is significantly simpler and highlights the dominant process behavior.

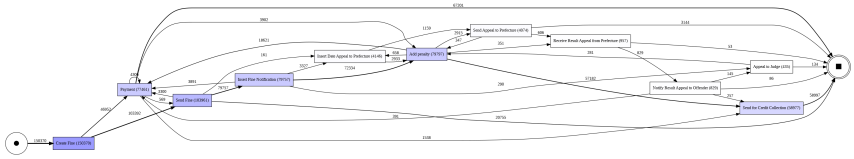

Figure 17: *Filtered Directly Follows Graph with a minimum frequency of 100.*

(d) Most Common Resource per Activity

Transformation functions used

- `map1`: Parses each log line and extracts the activity and resource.

$$\texttt{map1} : \text{String} \rightarrow (\text{activity}, \text{resource})$$

- `map2`: Converts each event into a countable key–value pair.

$$\texttt{map2} : (\text{activity}, \text{resource}) \rightarrow ((\text{activity}, \text{resource}), 1)$$

- `reduceByKey`: Aggregates counts to compute how often each resource executes a given activity.

$$\texttt{reduceByKey} : ((a, r), 1) \rightarrow ((a, r), \text{count})$$

- `map3`: Reorganizes the data by activity.

$$\texttt{map3} : ((a, r), c) \rightarrow (a, (r, c))$$

- `reduceByKey`: For each activity, selects the resource with the highest execution count.

Results Table 4 shows, for each activity, the resource that most commonly executes it, along with the corresponding execution count.

Activity	Most common resource	Count
Add penalty	MISSING	79,860
Appeal to Judge	R-0	555
Create Fine	R-538	8,608
Insert Date Appeal to Prefecture	MISSING	4,188
Insert Fine Notification	MISSING	79,860
Notify Result Appeal to Offender	MISSING	896
Payment	MISSING	77,601
Receive Result Appeal from Prefecture	MISSING	999
Send Appeal to Prefecture	MISSING	4,141
Send Fine	MISSING	103,987
Send for Credit Collection	MISSING	59,013

Table 4: *Most common resource per activity*