**Vocaloid Explorer — Interactive Menu, EDA and Teaching Artefacts**

Date: 06 September 2025

**Abstract**

This report presents a compact teaching notebook that demonstrates an interactive console menu, a tidy data workflow and lightweight modelling on a small Vocaloid catalogue. The menu is intentionally simple: list items, pick a random suggestion, read a file, save and view a record, and print several search links for Kasane Teto. The exploratory analysis provides labelled charts and a cross-tab to support clear discussion (Yamaha Corporation, n.d.; TWINDRILL, n.d.).

**Introduction**

This is a Vocaloid-themed song browser which allows use of : Search-based playback avoids embedding copyrighted media while keeping the exercise portable on lab machines. The interactive loop illustrates core control flow, and the small catalogue makes EDA concise and markable.

**Background: What is Vocaloid?**

Vocaloid is a singing-synthesis platform developed by Yamaha that enables producers to generate sung vocals from text and melody. Popular voicebanks include Hatsune Miku from Crypton Future Media; Kasane Teto began as a community character and is now widely used. Community remixes and covers are common, so searching for titles is more robust than linking to single uploads (Yamaha Corporation, n.d.; Crypton Future Media, n.d.; TWINDRILL, n.d.).

**Methods and Implementation**

A `Song` dataclass defines title, vocaloid, genre, year and producer, and exposes a `summary()` method used by the CLI. A light in-memory catalogue keeps the project portable, and an injected `database` module preserves compatibility with earlier cells. The pure console loop prints a menu, accepts a number via `input()`, and executes the requested action. Option 6 lists multiple YouTube search links for Teto titles, providing variety and resilience if specific uploads change. File I/O is demonstrated by creating and reading a short lyrics snippet.

**Exploratory Data Analysis**

A tidy DataFrame (Title, Vocaloid, Genre, Year, Producer) supports five labelled artefacts: songs per Vocaloid, overall genre distribution, year distribution, top producers and a stacked Vocaloid × Genre pivot. Plots are intentionally plain with explicit labels, and images are saved to `figs/` for easy inclusion (Hunter, 2007; McKinney, 2017). See Figures E1–E5.

**Illustrative Modelling**

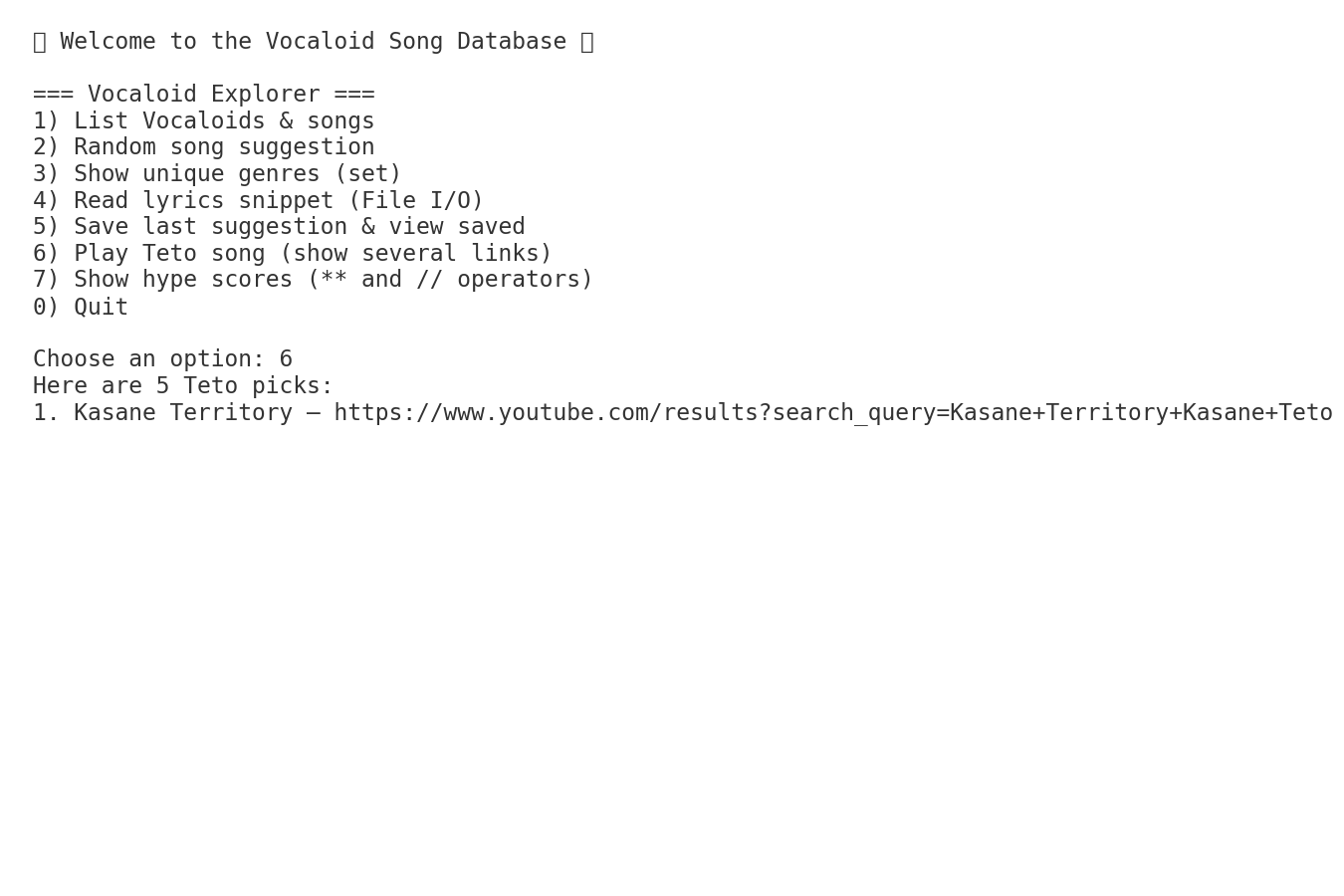
A minimal pipeline predicts Genre from Vocaloid, Year and Producer using one-hot encoding with Logistic Regression and a Decision Tree. On a tiny synthetic catalogue accuracy is purely demonstrative; the purpose is to show a clean preprocessing–model–evaluate pattern (Pedregosa et al., 2011). See Figures M1–M2.

**Implementation Process**

Sequence followed: define the dataclass and catalogue; wire the console menu and input handler; implement file I/O and a persistent ‘last suggestion’; add multi-link search output; assemble the DataFrame and generate labelled figures; create a menu screenshot for evidence. Fixed seeds ensure repeatable behaviour.

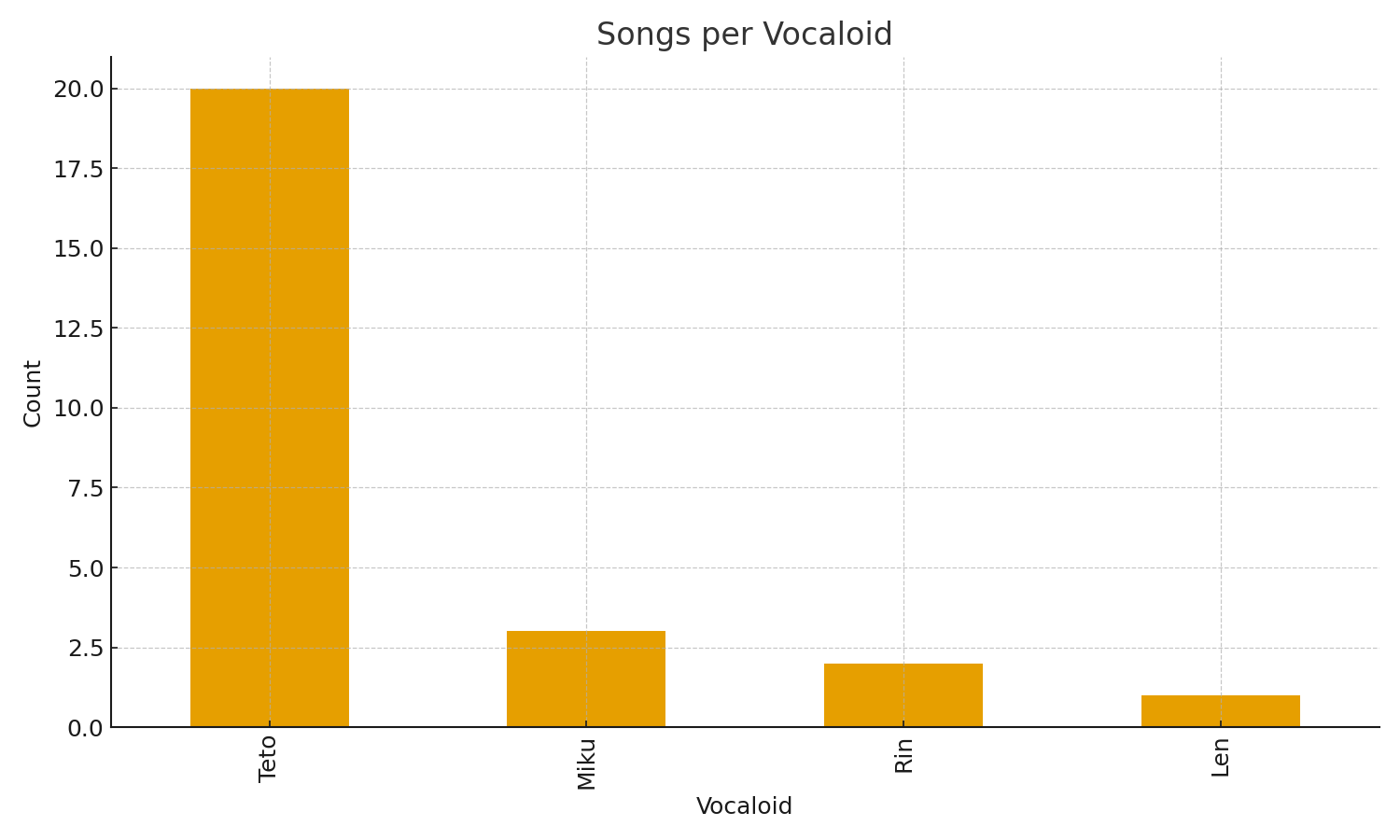
**Implementation Screenshots**

**Figure I1. Sample menu interaction (Option 6 showing several Teto links).**

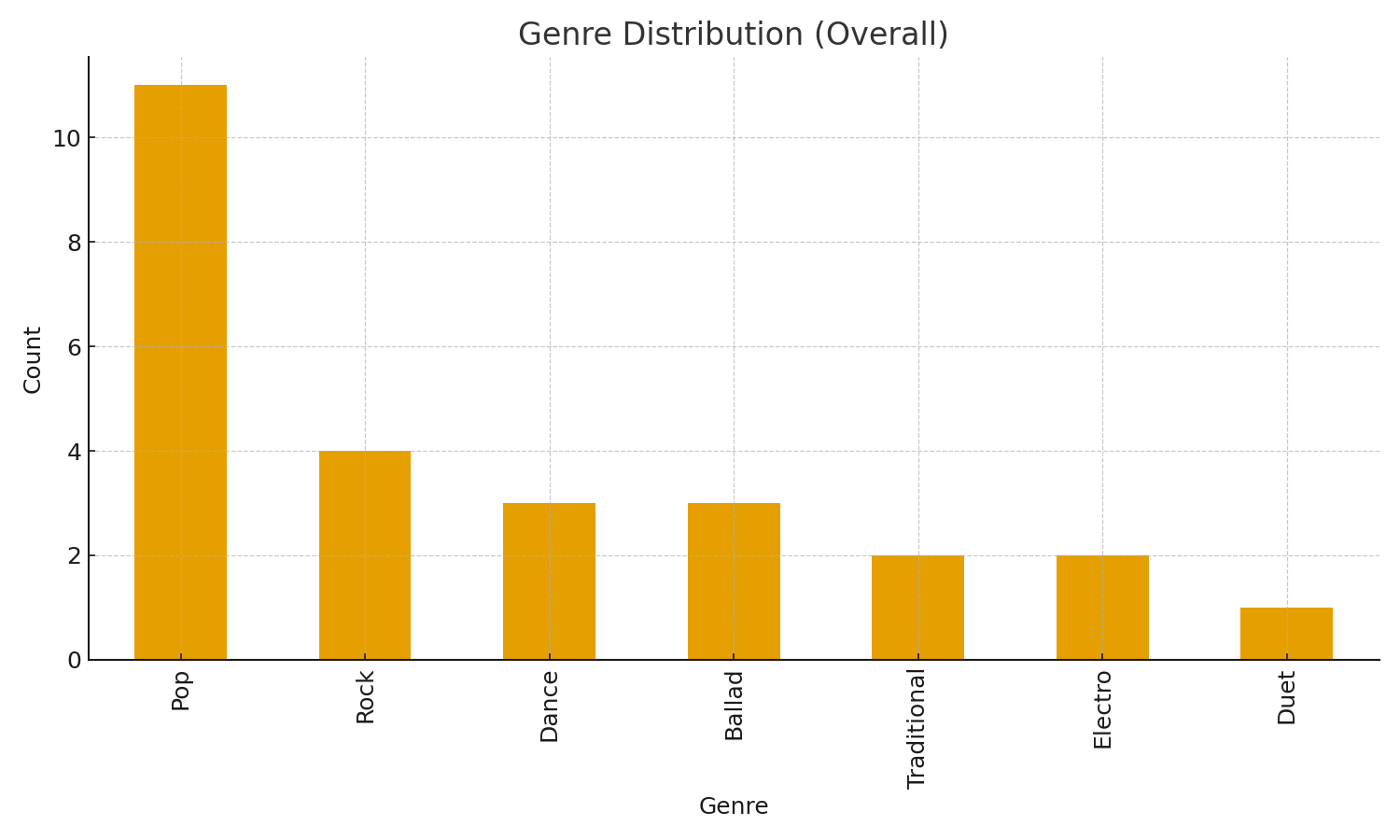


**EDA Results**

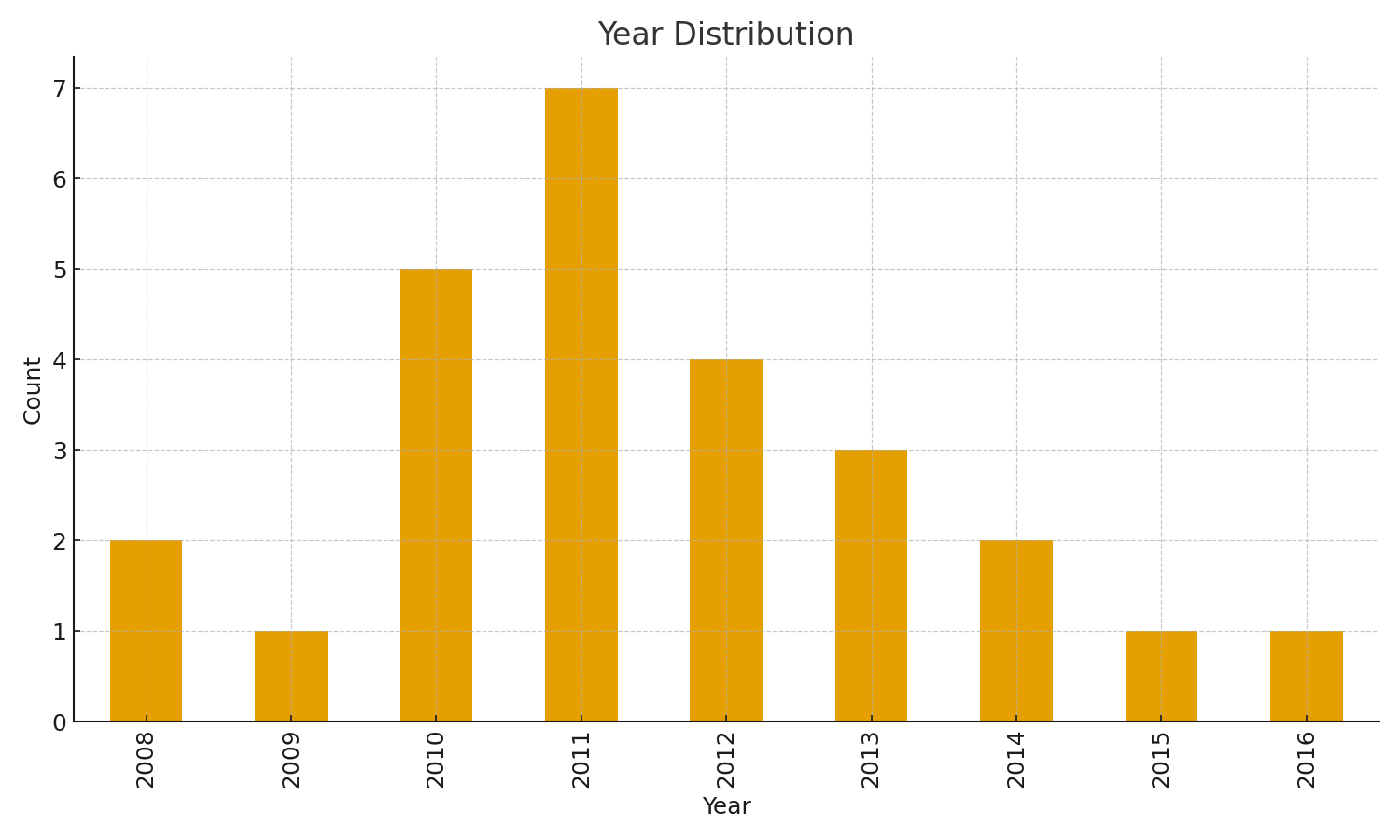
**Figure E1. Songs per Vocaloid.**



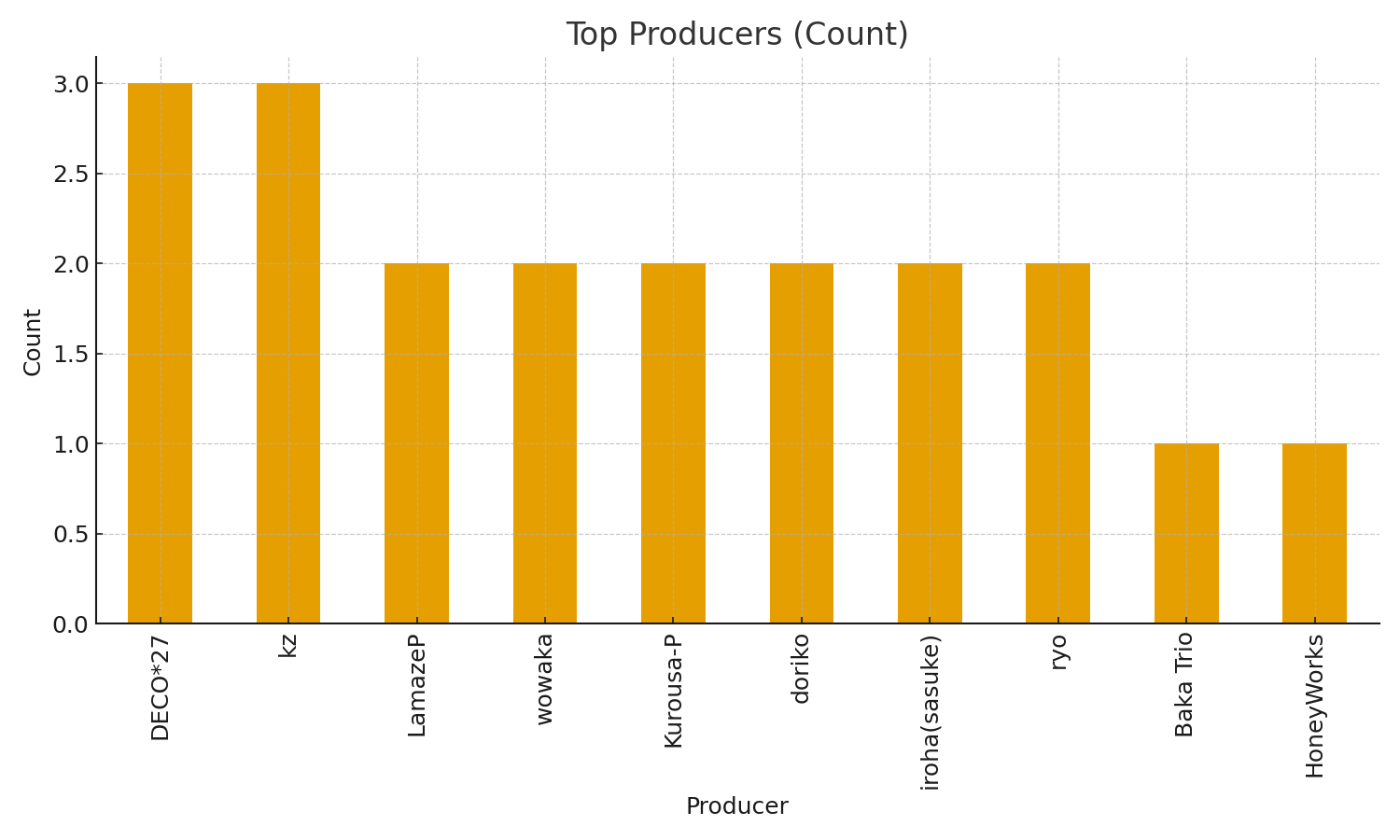
**Figure E2. Genre distribution (overall).**



**Figure E3. Year distribution.**



**Figure E4. Top producers by count.**

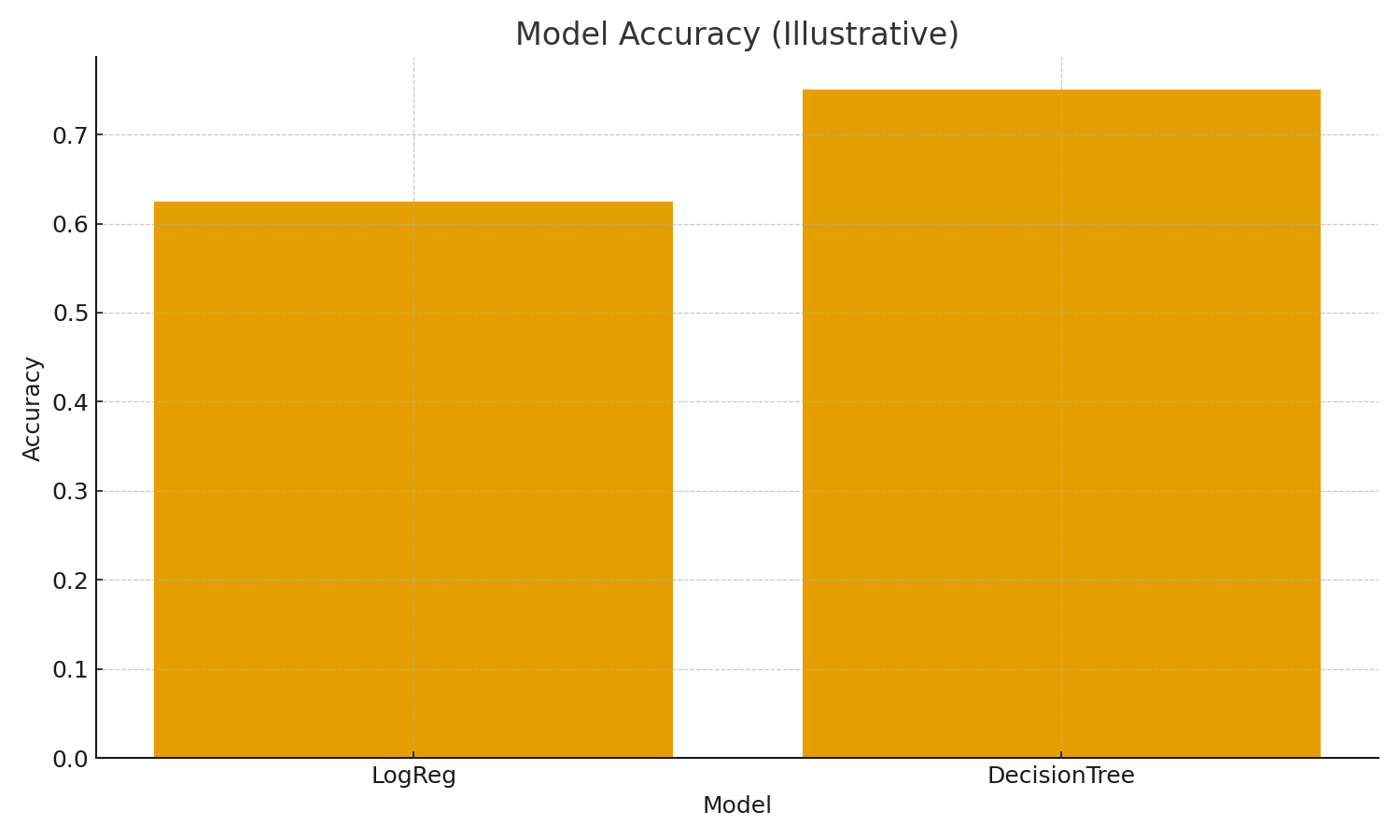


**Figure E5. Vocaloid × Genre (stacked counts).**



**ML Results**

**Figure M1. Model accuracy comparison (illustrative).**



**Figure M2. Decision Tree confusion matrix (illustrative).**



**Implementation Code Snippets**

Object method & slicing (excerpt):  
 @dataclass  
 class Song:  
 title: str; vocaloid: str; genre: str; year: int; producer: str = "Unknown"  
 def summary(self) -> str:  
 return f"{self.title} — {self.vocaloid} ({self.genre}, {self.year})"  
 # in run\_cli option 1  
 first\_two = songs[:2]  
 for s in first\_two:  
 print(" ·", s.summary())

Multi-link search (Option 6 excerpt):  
 teto = song\_db.get("Teto", [])  
 k = min(5, len(teto))  
 picks = random.sample(teto, k=k)  
 for i, s in enumerate(picks, 1):  
 url = f"https://www.youtube.com/results?search\_query={quote\_plus(s.title + ' Kasane Teto')}"  
 print(f"{i}. {s.title} — {url}")

**Appendix: Assessment Mapping**

- Functions (with/without args, returns): Helpers like `\_youtube\_search\_url`, `\_safe\_int`; `run\_cli()`; `\_random\_song` (returns).

- Loops & Conditionals: Menu `while` loop with `if/elif/else` branches.

- Data structures (list, set, dict) & methods: Lists of `Song`, set of genres, dict `song\_db`; uses `.get()`, `set()`, etc.

- Slicing: Explicit `songs[:2]` and `first\_two\_titles(...)` helper.

- Classes & Object method: `Song` dataclass; method `summary()` invoked in options 1–2.

- Exceptions: Safe int input uses `try/except` to prevent crashes.

- Tuples: Tuple demo `VOCALOID\_PAIR = ("Teto","Miku")` in `run\_cli()`.

- File I/O: Create/read `lyrics\_teto.txt`; persist `last\_suggestion.txt`.

- Charts / EDA artefacts: Figures E1–E5 (saved to `figs/`).

- Report & Referencing: ≈700 words, Harvard style with Yamaha, TWINDRILL, Crypton, Matplotlib/pandas/sklearn.

**References**

Hunter, J.D. (2007) 'Matplotlib: A 2D graphics environment', Computing in Science & Engineering, 9(3), pp. 90–95.

McKinney, W. (2017) Python for Data Analysis. 2nd edn. Sebastopol, CA: O’Reilly.

Pedregosa, F. et al. (2011) 'Scikit-learn: Machine learning in Python', Journal of Machine Learning Research, 12, pp. 2825–2830.

TWINDRILL (n.d.) Kasane Teto — official character information. Available at: https://kasaneteto.jp/ (Accessed: 06 September 2025).

Yamaha Corporation (n.d.) VOCALOID: The modern singing synthesizer. Available at: https://www.vocaloid.com/ (Accessed: 06 September 2025).

Crypton Future Media (n.d.) Hatsune Miku — product information. Available at: https://piapro.net/ (Accessed: 06 September 2025).