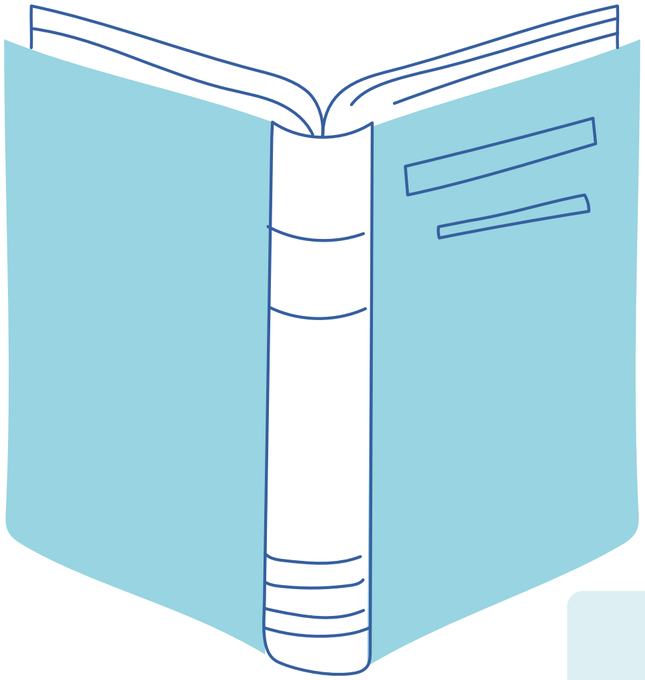
ASTANA IT UNIVERSITY



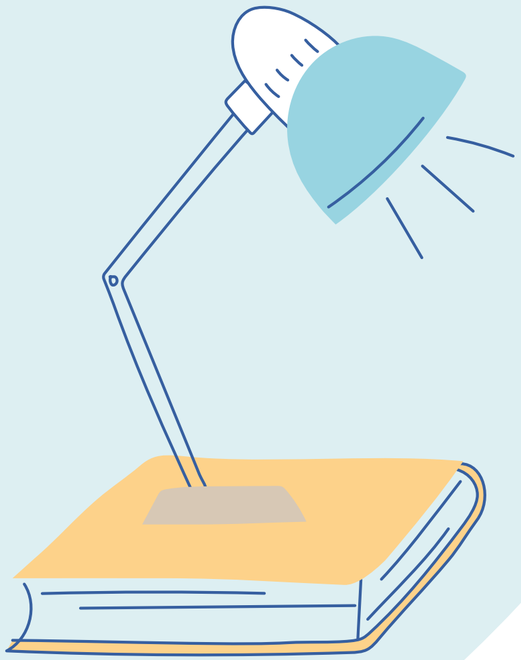
Final Project Presentation

Academic Reference Quality Scoring System Using Machine Learning

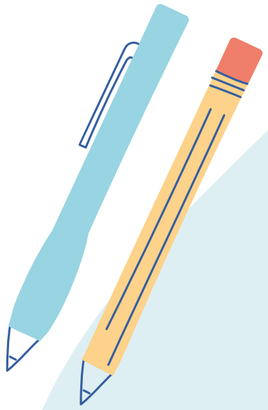
By S.Faizullayev &
M.Kamal Zada
Astana IT University

Research Project IP 2218

Instructor: Assiya Karatay



Overview



Introduction

We propose an automated system that assigns quality scores to academic references using machine learning.

Problem & Motivation

Why This Project Matters

- Academic papers rely on references to support claims
- Many references are outdated, weak, or from low-quality sources
- Manual checking of references is slow and subjective
- Need for an automatic system to score reference quality

Project objectives

01

Build a system that predicts the quality score of academic references

02

Use reference metadata (year, citations, publisher, authors, etc.)

03

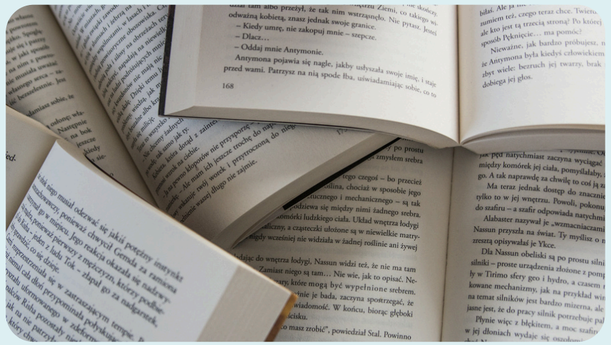
Compare machine learning models

04

Provide a usable tool for students and researchers

Dataset & Data

Sources: CrossRef, Google Scholar, Semantic Scholar



Features

- *Publication year*
- *Citation count*
- *Publisher / journal*
- *Number of authors*
- *Document type*

Labels

Continuous quality score based on predefined rules

Split

- *80% training*
- *20% testing*

Methodology

System Architecture

Flow

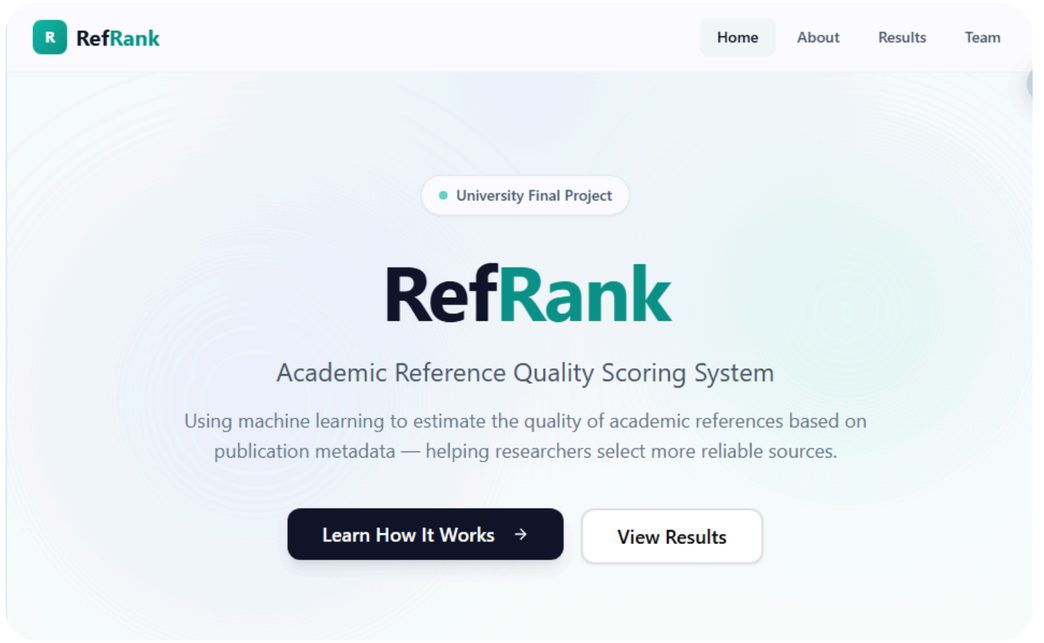
Input Reference

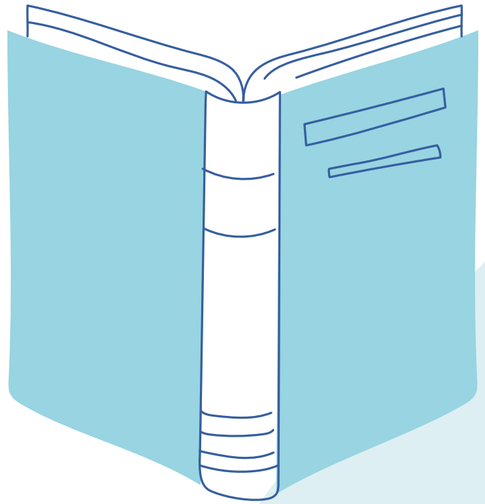
Preprocessing

Feature Extraction

Predicted Quality Score

ML Model





Methodology

Machine Learning Model

Neural network regression model

02 **Regularization: Dropout + L2**

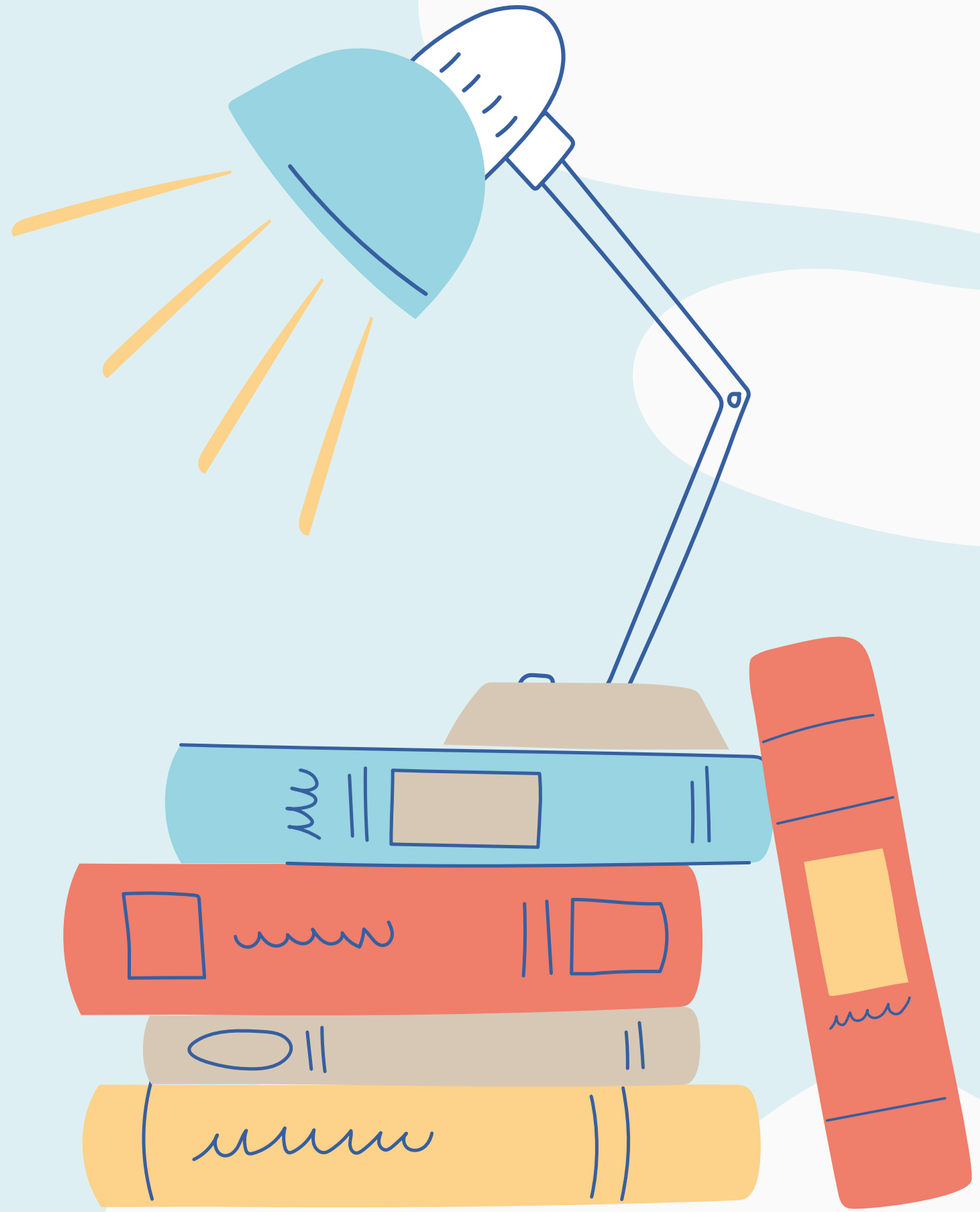
Input: reference metadata

04 **Activation: GELU**

Hidden layers:

- 128 neurons
- 64 neurons

06 **Output: continuous quality score**



Training Procedure

Optimizer: Adam

Loss: Mean Squared Error

Metrics:

- RMSE
- MAE

Hyperparameters:

- *Learning rate search*
- *Batch size tuning*

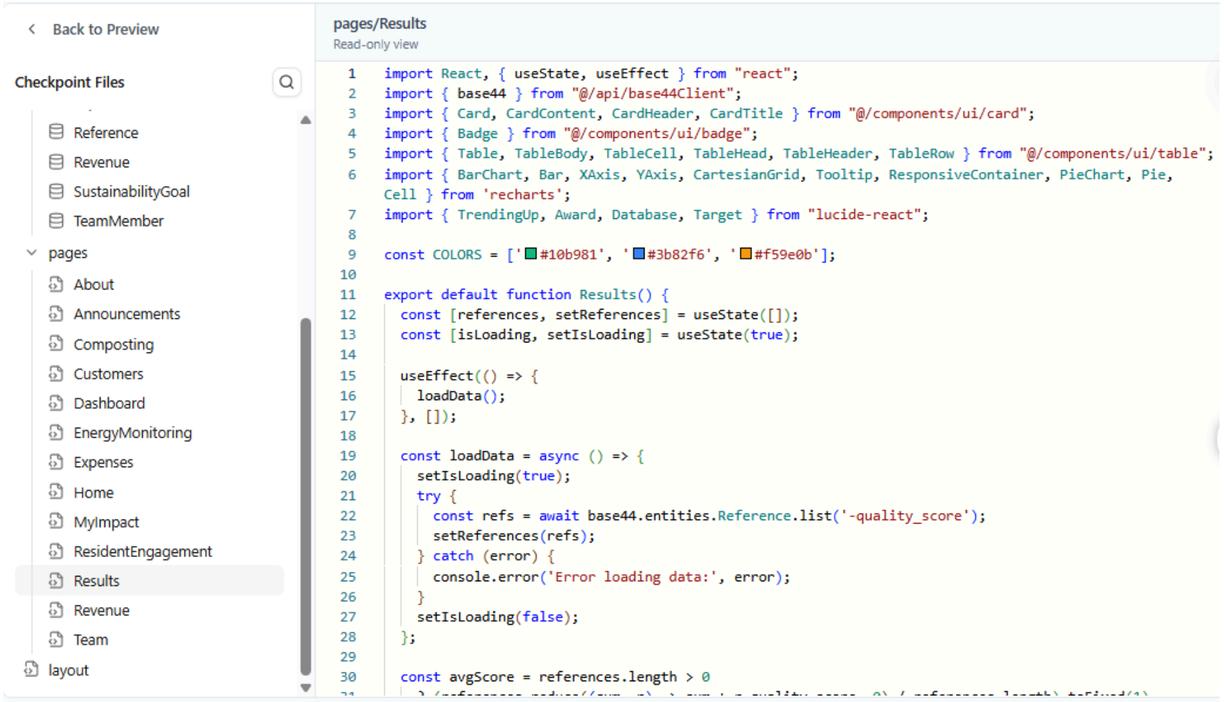
Experiments

01

1:Learning rate tuning

2: Model comparison

- Baseline
- Linear regression
- Random Forest
- Neural Network



Results

02

- Neural network achieved lowest RMSE and MAE
- Better performance on difficult cases
- Stable predictions across score range
- $R^2 > 0.85$

Discussion

01

- ***Model can separate high-quality and low-quality references***
- ***Metadata alone works reasonably well***

02

Useful for:

- *Academic writing tools*
- *Reference managers*
- *Peer review support*

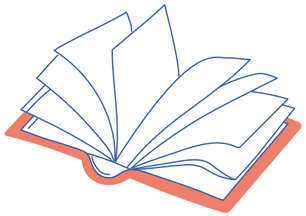
Limitations & Future Work

03

Points:

- *Labels are based on rules, may include bias*
- *No full-text analysis yet*
- *Dataset may not cover all research fields*

04



Future:

- *Add abstract and text analysis*
- *Use NLP for context relevance*
- *Expand dataset*

Conclusions

01

Built an automated reference quality scoring system

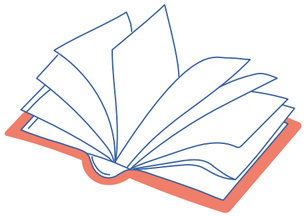
02

Machine learning improves reliability of reference selection

03

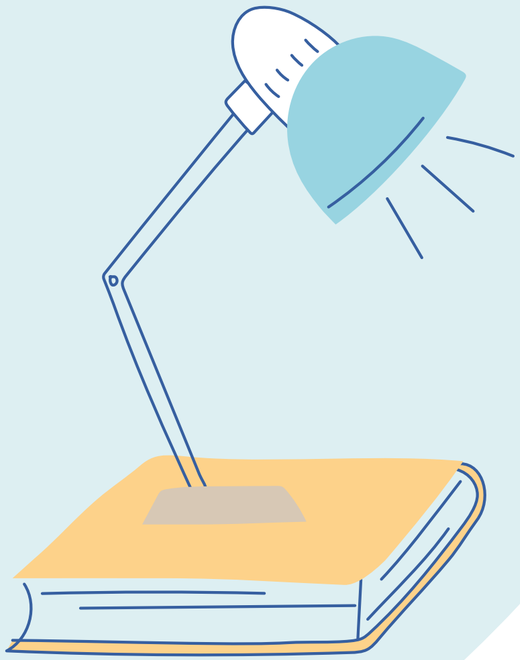
Can help researchers choose better sources

04



Ready to be integrated into tools and platforms

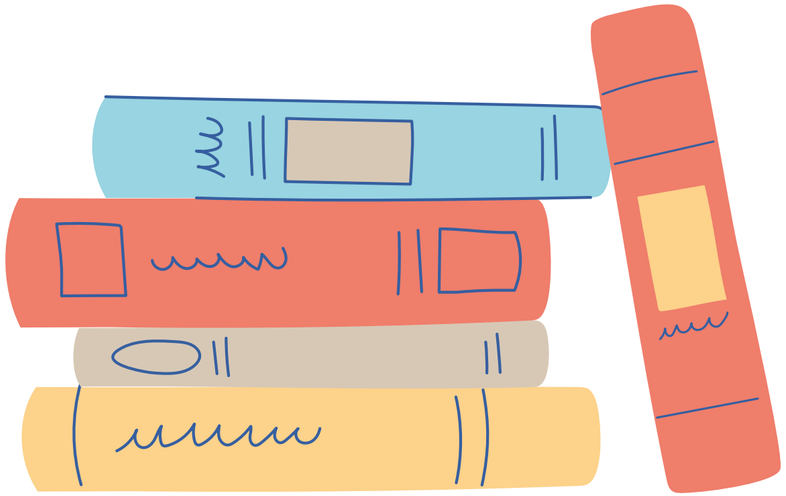
App Link:



Reference

- 1. Garfield, E. (1972). Citation analysis as a tool in journal evaluation. Science, 178(4060), 471–479. https://doi.org/10.1126/science.178.4060.471
- 2. Bornmann, L., & Daniel, H. D. (2008). What do citation counts measure? A review of studies on citing behavior. Journal of Documentation, 64(1), 45–80. https://doi.org/10.1108/00220410810844150
- 3. Moed, H. F. (2005). Citation Analysis in Research Evaluation. Springer. https://doi.org/10.1007/1-4020-3714-7
- 4. Waltman, L. (2016). A review of the literature on citation impact indicators. Journal of Informetrics, 10(2), 365–391. https://doi.org/10.1016/j.joi.2016.02.007
- 5. Hirsch, J. E. (2005). An index to quantify an individual's scientific research output. Proceedings of the National Academy of Sciences, 102(46), 16569–16572. https://doi.org/10.1073/pnas.0507655102
- 6. Sugimoto, C. R., & Larivière, V. (2018). Measuring Research: What Everyone Needs to Know. Oxford University Press

The End



Thank you

Question !

SCAN ME

Research Project: Academic Reference Quality Scoring System

Overview

This project implements a machine learning pipeline for evaluating the quality of academic references using bibliographic metadata.
The system leverages real publication data obtained from the **Semantic Scholar Academic Graph API** and predicts a continuous **reference quality score**.

The project focuses on:

- metadata-based feature extraction
- supervised regression modeling
- quantitative evaluation using standard ML metrics

How to Run This Notebook

1. Environment

This notebook is designed to run in **Google Colab**.

No manual dataset upload is required — data is fetched automatically from a public API.

---

2. Install & Import Dependencies

All required libraries are standard in Google Colab:

- pandas
- numpy
- scikit-learn
- requests
- (optional) tensorflow / keras for neural network experiments

No additional installation steps are needed.

---

3. Data Collection

The dataset is obtained programmatically from the **Semantic Scholar Academic Graph API** using a keyword-based search (e.g., *"machine learning"*).

Extracted metadata includes:

- publication year
- citation count
- number of authors
- venue availability

These attributes are commonly used in scientometric analysis.

---

4. Feature Engineering & Label Construction

A continuous **quality score (0–1)** is constructed using transparent, rule-based weighting of metadata features:

- citation impact
- publication recency
- authorship
- venue presence

This enables regression-based learning and comparison across models.

---

5. Model Training

The notebook supports multiple models:

- **Random Forest Regressor** (baseline, no epochs required)
- **Neural Network Regressor** (optional, trained with epochs)

Data is split into training and testing subsets (80/20) and standardized prior to training.

---

6. Evaluation Metrics

Model performance is evaluated using:

- **RMSE** (Root Mean Squared Error)
- **MAE** (Mean Absolute Error)
- **R^2 Score**

Additionally, an optional categorical accuracy is computed by mapping scores to:

- High / Medium / Low quality classes

7. Output

The processed dataset is saved locally as: semantic_scholar_dataset.csv This allows reproducibility and reuse in further experiments.

Notes

- The system validates the **methodology and ML pipeline**, not absolute scholarly quality.
- Full-text and NLP-based analysis are considered future work.
- The approach is suitable for early-stage research and academic tooling prototypes.

Authors:

Mustafa Kamal Zada, Salim Faizullayev

IP2218 Research Project

Instructor: Assiya Karatay

Academic Reference Quality Scoring System

Astana IT University

```
import time
import numpy as np
import pandas as pd
import requests

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.ensemble import RandomForestRegressor

np.random.seed(42)
```

```
BASE_URL = "https://api.semanticscholar.org/graph/v1/paper/search"

def fetch_papers(query: str, total: int = 800, batch: int = 100, sl
    out = []
    offset = 0

    fields = "title,year,citationCount,authors,venue"

    while len(out) < total:
        limit = min(batch, total - len(out))
        params = {
            "query": query,
            "limit": limit,
            "offset": offset,
            "fields": fields
        }
        r = requests.get(BASE_URL, params=params, timeout=60)

        if r.status_code == 429:
            # rate limit; wait and retry
            time.sleep(max(3.0, sleep_s) * 2)
            continue

        r.raise_for_status()
        data = r.json()

        batch_data = data.get("data", [])
        if not batch_data:
            break

        out.extend(batch_data)
        offset += limit
        time.sleep(sleep_s)

    return out

papers = fetch_papers(query="machine learning", total=800, batch=10
print("Fetched papers:", len(papers))
papers[0].keys()
```

```
Fetched papers: 800
dict_keys(['paperId', 'title', 'venue', 'year', 'citationCount',
'authors'])
```

```
def safe_num_authors(authors):
    if not authors:
        return 1
    return max(1, len(authors))

rows = []
for p in papers:
    rows.append({
        "year": p.get("year"),
        "citationCount": p.get("citationCount", 0),
        "num_authors": safe_num_authors(p.get("authors")),
        "venue": p.get("venue")
    })

df = pd.DataFrame(rows)

df["citationCount"] = pd.to_numeric(df["citationCount"], errors="co
df = df.dropna(subset=["year"])
df = df[(df["year"] >= 1900) & (df["year"] <= 2030)]
df["has_venue"] = df["venue"].notna().astype(int)

print("Rows after cleaning:", len(df))
df.head()
```

```
Rows after cleaning: 797
```

	year	citationCount	num_authors	venue	has_venue
0	2021.0	5377	6	Nature Reviews Physics	1
1	2021.0	3950	1	SN Computer Science	1
2	2017.0	10082	3	arXiv.org	1
3	2019.0	5323	5	ACM Computing Surveys	1

Proxy Reference Quality Score

The reference quality score is defined using a rule-based composite formulation inspired by established bibliometric evaluation frameworks, including the Article Influence Score and Reference Quality Scoring Systems (RQSS) used in scholarly databases and knowledge graphs to assess credibility and trustworthiness of references [1], [2]. These approaches evaluate reference quality by combining multiple normalized indicators rather than relying on raw citation counts alone. In line with these principles, the proposed score integrates available reference metadata, including citation count as a measure of scholarly influence, publication year to capture recency effects, number of authors as an indicator of collaborative strength, and venue availability as a basic credibility signal. The resulting composite score provides a continuous target variable suitable for supervised learning experiments in automated reference quality assessment.

References

[1] C. T. Bergstrom, J. D. West, and M. A. Wiseman, “The Eigenfactor™ metrics,” Journal of Neuroscience, vol. 28, no. 45, pp. 11433–11434, 2008. https://doi.org/10.1523/JNEUROSCI.0003-08.2008

[2] C. R. Sugimoto and V. Larivière, Measuring Research: What Everyone Needs to Know. Oxford, UK: Oxford University Press, 2018, pp. 1–152. https://www.researchgate.net/publication/346866471_Measuring_Research_What_Everyone_Needs_to_KnowRWhat_Everyone_Needs_to_KnowR

```
year_norm = (df["year"] - df["year"].min()) / (df["year"].max() - df
cit_norm  = np.log1p(df["citationCount"]) / (np.log1p(df["citationCo
auth_norm = (df["num_authors"] - 1) / (df["num_authors"].max() - 1 +

df["quality_score"] = (
    0.45 * cit_norm +
    0.35 * year_norm +
    0.15 * auth_norm +
    0.05 * df["has_venue"] +
    np.random.normal(0, 0.03, size=len(df))
).clip(0, 1)

df[["year", "citationCount", "num_authors", "has_venue", "quality_sc
```

	year	citationCount	num_authors	has_venue	quality_score
0	2021.0	5377	6	1	0.744217
1	2021.0	3950	1	1	0.707090
2	2017.0	10082	3	1	0.747557
3	2019.0	5323	5	1	0.762145
4	2018.0	7706	1	1	0.713754

```
X = df[["year", "citationCount", "num_authors", "has_venue"]]
y = df["quality_score"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

scaler = StandardScaler()
X_train_s = scaler.fit_transform(X_train)
X_test_s  = scaler.transform(X_test)
```

```
model = RandomForestRegressor(
    n_estimators=300,
    random_state=42,
    n_jobs=-1
)
model.fit(X_train_s, y_train)
```

```
RandomForestRegressor(n_estimators=300, n_jobs=-1, random_state=42)
```

```
y_pred = model.predict(X_test_s)

rmse = np.sqrt(mean_squared_error(y_test, y_pred))
mae  = mean_absolute_error(y_test, y_pred)
r2   = r2_score(y_test, y_pred)

print("Research Project — Model Evaluation")
print(f"RMSE: {rmse:.4f}")
print(f"MAE:  {mae:.4f}")
print(f"R²:   {r2:.4f}")
```

```
Research Project — Model Evaluation
RMSE: 0.0326
MAE:  0.0262
R²:   0.7568
```

```
def to_bin(v):
    if v >= 0.75:
        return "High"
    if v >= 0.40:
        return "Medium"
    return "Low"

true_bin = y_test.apply(to_bin).values
pred_bin = pd.Series(y_pred).apply(to_bin).values

acc = (true_bin == pred_bin).mean()
print(f"Binned Accuracy: {acc:.2%}")
```

```
Binned Accuracy: 98.75%
```

```
df.to_csv("semantic_scholar_dataset.csv", index=False)
print("Saved: semantic_scholar_dataset.csv")
```

```
Saved: semantic_scholar_dataset.csv
```