# Project Title: Recommender Systems using Amazon Reviews.

# One sentence summary

One Sentence Summary Ex: This repository holds an attempt to apply LSTMs to Stock Market using data from "Get Rich" Kaggle challenge (provide link).

This project takes Amazon user data from Kaggle and utilizes collaborative filtering methods to create a quick and reliable recommender system. [Kaggle Challenge](https://www.kaggle.com/code/saurav9786/recommender-system-using-amazon-reviews/notebook)

# Abstract

This section could contain a short paragraph which include the following:

* In this challenge, my goal is to accurately predict which products users will buy based on user activity, alongside ratings which have been given by the user.
* Using a collaborative filtering system, this approach takes product IDs, compiles a list of users who scored them, and builds a similarity between all these users. From there it is possible to compare similar products and suggest those to other users.
* A model employed by other Kaggle users has netted a runtime of 1 minute and 13 seconds. Following a similar route, I intend to get under a minute with my specific method.

# Introduction

Create the context. Explain the domain. Plan for rest of the document.

Since the beginning of this semester, I have realized that there is still so much within my field I have no experience with. Taking this class has accelerated the pace at which I need to create familiarity with Machine Learning or Algorithms, but I believe in the process and wish to separate myself from merely knowing, to testing and accepting. This mindset has become the catalyst for a ruthless introspection, I now must pick a direction and stay there for a time. This direction is recommender systems, a topic I am familiar with because my Communications Minor has had me study them, however I lack technical expertise in order to further understand. Why not challenge what I think I know by researching and developing a counter knowledge, based on actual experience? If I want to be competent in this field, I shouldn’t hesitate to challenge ideas and counter them with results.

Motivation and Background

* This topic intrigues me because the methods of filtering can be used on different levels, so there is enough room for me to iterate as I see fit. This is the perfect challenge for me, as its structure and implementation are theoretically easy to comprehend but cannot be replicated with just the bigger picture in the same way a board game can be coded. With this said, a challenge like this will aid at a time where I am still an intermediate.
* This is also a unique issue because it is solely based on how well I can use algorithms to enhance the performance of a system. Given that this project is a success, there are numerous ways to iterate on this idea, like using deep learning.
* In this problem, I specifically am looking for a way to take implicit data and factor it alongside explicit data, in order to give more accurate representation of what people want. Implicit and explicit data includes things like whether you clicked a product versus if you liked or rated it 5 stars. One glaring problem that I intend to solve, is that implicit data is never recorded negatively. If you click a product, you show an arbitrary ‘interest’ in it and the recommender system factors this into its suggestions.
* This appears to be a structural and functional programming challenge, many of my previous works only loosely apply, but I have the skills to purvey this sort of information. Some references I have used are two Kaggle notebooks and an article, which I have linked below. All these references go into detail over the steps I will take to build a Recommender System using Python. Because provide a solution to the code. With that said, I don’t believe this is a new problem at all, nor is it one that has never been solved. There is existing code available, but I won’t necessarily be reproducing it. Any extensions I may make will likely come as a result of challenging myself further, like taking an Object-Oriented approach to some of the functions.

[An article which helped me understand what Implicit and Explicit rating systems are.](https://towardsdatascience.com/building-a-collaborative-filtering-recommender-system-with-clickstream-data-dffc86c8c65)

[A Kaggle challenge for Deep Learning using Recommender Systems.](https://www.kaggle.com/code/jamesloy/deep-learning-based-recommender-systems)

[The Data Set I will be using.](https://www.kaggle.com/datasets/gspmoreira/articles-sharing-reading-from-cit-deskdrop)

# Problem Formulation

## Dataset

* My Dataset consists of 73k instances of User Activity from CI&T DeskDrop. This data set contains 5 crucial features: Item Attributes (User info, Article URL), User context (Context for the implicit feedback like user visits), Logged user tracking, Implicit feedback, as well as the type of client from the user.
* The Article Data is 17.99 MB, and the User Interactions are 11.95MB

## Task Definition

* The goal is to reduce all the data provided to suggestions that may interest users.
* The type of algorithm that will power my recommender system is a Matrix Factorization technique. Because it merely is a mathematical framework, it can be supervised or unsupervised.
* The Inputs will be the explicit and implicit data that were collected, such as Likes/Thumbs ups and user article history. The output would be related items like Article links.
* Some libraries that can assist me are Surprise, SciPy, FastFM, and CaseRecommender. Surprise is specifically designed to implement Recommender Systems and comes pre-built to read data. SciPy contains modules for Matrices, and prepackaged algorithm implementation. FastFM is known as a Factorization Machine, also compiled in C so the performance is also great. CaseRecommender seems to contain modules to work with Implicit and Explicit data sets, and serves as a good ‘all around’ option. I doubt I will use them all at once, so for the time being CaseRecommender and SciPy seem most beginner friendly. For better performance down the line, FastFM and Surprise can be tested.

## Performance Evaluation

* Some metrics to look for are predictive successes and accuracy of article suggestions.
* Similar Kaggle challenges that rely solely on explicit sets run in just over a minute, but the Kaggle challenge I am using for its data runs in 191.3 seconds.
* The ‘training’ process doesn’t exactly apply, because no Deep Learning models are being employed, but it appears testing out the different Recommendation techniques will take up the majority of validity testing time. Using FastFM may give me an edge in performance further down the line because it is compiled instead of interpreted, so that is another aspect to explore.

# Planning

## GitHub Package

* What is the goal of the package you will provide? What could it look like.
* The vision I have for this package is a program that operates off of one button click. The program has a wide base of knowledge from articles shared through DeskDrop, in which suggestions for new and targeted articles are made with a simple button press. It would operate similarly to FireFox’s browser, which subtly runs a similar function based off of browser history, and places article links accordingly in a New Tab.

Goals

* Clearly define the goal for each stage.
  + Feasibility

In the feasibility stage, I will take my time understanding specifically which Recommendation Algorithm is needed, and whether I’ll make the approach to tie similarities between articles or between users (Item-to-Item VS User-to-User.) Next, I will make my way through the different libraries and see which package works best for the Algorithm I go with.

* + Prototype

In prototyping, most of the time will go into designing code to support Matrix Factorization techniques and creating features to take implicit/explicit data.

* + Production

The production stage revolves around performance after the code works as intended. For bonus points I would like to make an app that can run this program and suggest some articles to users.

## Workplan

* Outline a detailed 5-week workplan.
  + List tasks / goals for each week

Week 1: Accumulating Recommender Systems and understanding the difference between Recommender Systems. At this time, I will also build code to hold the data downloaded from Kaggle.

Week 2: Dabbling in matrix factorization and building a working framework in code. Ideally in this time, I’ll be starting to familiarize myself with the library features in SciPy or CaseRecommender.

Week 3: In the third week of development, I will be writing algorithms with the libraries I have worked with in the second week. By this point I expect to already have worked code that takes in the Data and outputs results based on the explicit rating system.

Week 4: In this phase, I expect to already have the explicit rating system working and this week will be devoted to integrating the implicit data. With this I’ll have both values factor into what recommendations are made.

Week 5: This final week will be devoted to condensing this code and trying to gauge performance. This will be the week to beat the Kaggle challenge time of 191.3 seconds.