**Project Plan**

**Project Overview**

* This big data project involves scraping products information sold on Amazon website, followed by data storage, cleaning/wrangling, processing, model building, machine learning, and application deploying (maybe an API, web application or dashboard) in the AWS cloud.
* The first step would be to **identify the products** to analyze and then use **web scraping** tools to extract price, description, selling rank, location, Amazon’s Choice, Manufacturer, Frequently bought together, customer reviews and ratings for each product. This would require developing a web scraper that can navigate the Amazon website and collect the desired data.
* Once the data is collected, **it would need to be cleaned**, preprocessed and transformed into a structured format suitable for analysis. This could involve removing irrelevant information such as duplicate reviews, formatting text data, and categorizing the reviews based on various criteria like sentiment, product features, and customer demographics.
* The next step would be to perform **exploratory data analysis** to identify trends, patterns, and correlations in the data. This could involve using **statistical techniques** like clustering, regression, and machine learning algorithms to identify relationships between different features of the products and customer reviews.
  + **Sentimental analysis** can be done from customer reviews for each product. <https://www.analyticsvidhya.com/blog/2022/07/sentiment-analysis-using-python/>

Diagram

Description automatically generated

* Finally, the insights obtained from the analysis could be used to:
  + Educate customers with better product choice,
  + Help individual Amazon seller to improve the product quality, pricing, marketing strategy, and customer service.
* The final delineable can be one of the following depends on timeline:
  + Results of the analysis could be presented in a user-friendly dashboard (PowerBI or Tableau with interaction) or report;
  + Build an API or package hosed in PyPi so other people can replicate our work in their computer as well. <https://www.freecodecamp.org/news/build-your-first-python-package/>
  + Build a Dash interactive application hosted in AWS cloud.

**Tools Involved**

* Git and GitHub will be used to do version control and collaboration.
* In each of the above steps, we will keep in mind of Scaling. We will explore using different big data best practise, like multiple threads and concurrency (we totally need this, because our data scrapping is going to be huge and the process is slow), using SparkSQL to store our data, using Dask and Parquet to do analysis.
* Additional things we can try are Hive, Cassandra, GraphX, Tensorflow.
* More details can be:
  + Web Scraping Frameworks: There are several web scraping frameworks available such as **Beautiful Soup**, Scrapy, and Selenium, which you can use to extract data from the Amazon website.
  + Data Storage: You can store the scraped data in a cloud-based database like **Amazon S3**, Amazon RDS, or Amazon DynamoDB.
  + Data Processing and Analysis: You can use cloud-based data processing and analysis tools like Amazon EMR, Amazon Redshift, and Amazon Athena to process and analyze the scraped data.
  + Machine Learning: You can use Amazon SageMaker, a cloud-based machine learning service, to build machine learning models that can analyze customer reviews and provide insights to improve the product quality, pricing, and marketing strategy.
  + Visualization: You can use tools like Amazon QuickSight or Tableau to create interactive visualizations and dashboards to present the insights
* We can use **AWS Elastic Beanstalk** to deploy Python packages and Dash applications. Elastic Beanstalk is a fully managed service that makes it easy to deploy and scale applications in multiple languages, including Python. To deploy your Python package and Dash application, you can follow these steps:
  + Create a Python package: Create a Python package for your application that includes all the required dependencies.
  + Create a virtual environment: Create a virtual environment for your application that includes all the required Python packages.
  + Create an Elastic Beanstalk environment: Create an Elastic Beanstalk environment for your application by selecting the Python platform and providing the necessary configuration details.
  + Upload your package: Upload your Python package and application code to the Elastic Beanstalk environment.
  + Configure the environment: Configure the environment by setting environment variables and other settings that your application requires.
  + Deploy your application: Deploy your application by starting the Elastic Beanstalk environment. You can use the AWS Management Console or the AWS Command Line Interface (CLI) to start the environment.