**Search Engine On Amazon Product Dataset Project:**

**Code Explanation**

**Importing Libraries**

The code starts with importing necessary libraries such as pandas, numpy, nltk, SnowballStemmer, TfidfVectorizer, cosine\_similarity, and streamlit.

**Loading Dataset**

Then, it loads a dataset named "amazon\_product\_dataset.csv" using the pandas library.

**Removing Unnecessary Columns**

Next, the code drops an unnecessary column 'id' from the dataset using the "drop()" function of pandas.

**Tokenization and Stemming**

After removing the unnecessary columns, the code defines a tokenizer and stemmer using the SnowballStemmer library from nltk. Then, it defines a function named "tokenize\_and\_stem" that tokenizes and stems the given text using the previously defined tokenizer and stemmer.

**Creating Stemmed Tokens Column**

The code applies the "tokenize\_and\_stem" function on each row of the dataset's 'Title' and 'Description' columns, concatenates them, and saves the stemmed tokens in a new column 'stemmed\_tokens'.

**TF-IDF Vectorizer and Cosine Similarity**

Then, the code defines a TF-IDF vectorizer using TfidfVectorizer from sklearn. It also defines a function named "cosine\_sim" that takes two texts and returns their cosine similarity using the previously defined TF-IDF vectorizer.

**Search Function**

Next, the code defines a search function named "search\_products" that takes a query, tokenizes and stems it using the "tokenize\_and\_stem" function, and then calculates cosine similarity between the query and each row of the dataset's 'stemmed\_tokens' column using the "cosine\_sim" function. It then sorts the dataset based on similarity and returns the top 10 relevant results.

import pandas as pd  
import numpy as np  
import nltk  
from nltk.stem.snowball import SnowballStemmer  
from sklearn.feature\_extraction.text import TfidfVectorizer  
from sklearn.metrics.pairwise import cosine\_similarity  
import streamlit as st  
  
# Load the dataset  
data = pd.read\_csv('amazon\_product\_dataset.csv')  
  
# Remove unnecessary columns  
data = data.drop('id'**,** axis=**1**)  
  
# Define tokenizer and stemmer  
stemmer = SnowballStemmer('english')  
def tokenize\_and\_stem(text):  
 tokens = nltk.word\_tokenize(text)  
 stems = [stemmer.stem(t) for t in tokens]  
 return stems  
  
# Create stemmed tokens column  
data['stemmed\_tokens'] = data.apply(lambda row: tokenize\_and\_stem(row['Title'] + ' ' + row['Description'])**,** axis=**1**)  
  
# Define TF-IDF vectorizer and cosine similarity function  
tfidf\_vectorizer = TfidfVectorizer(tokenizer=tokenize\_and\_stem)  
def cosine\_sim(text1**,** text2):  
 tfidf\_matrix = tfidf\_vectorizer.fit\_transform([text1**,** text2])  
 return cosine\_similarity(tfidf\_matrix)[**0**][**1**]  
  
# Define search function  
def search\_products(query):  
 query\_stemmed = tokenize\_and\_stem(query)  
 data['similarity'] = data['stemmed\_tokens'].apply(lambda x: cosine\_sim(query\_stemmed**,** x))  
 results = data.sort\_values(by=['similarity']**,** ascending=False).head(**10**)[['Title'**,** 'Description'**,** 'Category']]  
 return results  
  
# Create Streamlit app  
st.title('Amazon Product Search')  
  
# Create search box and button  
query = st.text\_input('Enter a product name')  
search\_button = st.button('Search')  
  
# Perform search and display results  
if search\_button:  
 results = search\_products(query)  
 st.write(results)

**Streamlit App**

Finally, the code creates a Streamlit app titled "Amazon Product Search" and a text input field named "Enter a product name" and a "Search" button. When the user clicks the "Search" button, it calls the "search\_products" function with the entered query, retrieves the top 10 relevant results, and displays them.

**Report**

**Project Description**

The project aims to build a search engine that allows users to search for products in the Amazon Product Dataset using a query. It uses natural language processing techniques such as tokenization, stemming, and TF-IDF vectorization to convert product titles and descriptions into a numerical format and calculate cosine similarity between the query and the products.

**Dataset**

The dataset used in this project is the "Amazon Product Dataset" available on Kaggle. It contains product information such as title, description, price, category, etc., for over 1.3 million products. For this project, we only used the 'Title', 'Description', and 'Category' columns.

**Implementation**

The implementation of the project involved several steps. First, we loaded the dataset using pandas and removed the 'id' column. Then, we defined a tokenizer and stemmer using nltk's SnowballStemmer and created a function to tokenize and stem the product titles and descriptions.

Next, we used TfidfVectorizer from sklearn to convert the stemmed tokens into a TF-IDF matrix. We also defined a function to calculate cosine similarity between two texts using the TF-IDF matrix.

Finally, we created a Streamlit app that allows users to enter a query and retrieves the top 10 relevant products using the search function. The app displays the product title, description, and category for each result.

**Conclusion**

In conclusion, we successfully built a search engine that allows users to search for products in the Amazon Product Dataset using a query. The engine uses natural language processing techniques to convert product titles and descriptions into a numerical format and