Data Cleansing and Text Analysis Challenge - Task 7

- Student Name: Michael Rideout
- Student Number: 225065259
- E-mail: s225065259@deakin.edu.au
- Student Course Code: SIT731

Introduction

This report presents an analysis of the Data Science Stack Exchange [1] community, investigating how this question and answer forum reflects the evolving field of data science. Stack Exchange communities provide vital community driven knowledge repositories where members share expertise, answer questions and solve problems. Through the analysis of the communities' data export, we an gain useful insights into trends, issues and common solutions faced by members of the community and the data science field in general.

This investigation focuses on the following aspects: * Popularity of languages and tools * User engagement * User location analysis

Data Preparation

This section documents the process of downloading, extracting and transforming the Data Science Stack Exchange data export. The result will be the transformation of the xml feeds to csv feeds.

The Data Science Stack Exchange data export at time of writing was produced on 7th April 2024. There are 137,433 users and 78,926 posts in the dataset.

Download and Extraction

Here we download the Data Science Stack Exchange internet archive [2] if the file hasn't been downloaded already. Then the files are extracted, if they don't already exist.

```
import os
import requests

!pip install py7zr
import py7zr

import pandas as pd
import xml.etree.ElementTree as ET
import csv
import re
import matplotlib.pyplot as plt
import plotly.express as px
import tabulate


!pip install wordcloud
from wordcloud import WordCloud
from urllib.parse import urlparse
import sys

!pip install pycountry
from pycountry import countries
from difflib import get_close_matches

%matplotlib inline
```

```
Requirement already satisfied: py7zr in /home/mick/bin/anaconda/envs/scratch/lib/python3.12/s
Requirement already satisfied: texttable in /home/mick/bin/anaconda/envs/scratch/lib/python3
Requirement already satisfied: pycryptodomex>=3.16.0 in /home/mick/bin/anaconda/envs/scratch/
Requirement already satisfied: pyzstd>=0.15.9 in /home/mick/bin/anaconda/envs/scratch/lib/pyt
Requirement already satisfied: pyppmd<1.2.0,>=1.1.0 in /home/mick/bin/anaconda/envs/scratch/l
Requirement already satisfied: pybcj<1.1.0,>=1.0.0 in /home/mick/bin/anaconda/envs/scratch/li
Requirement already satisfied: multivolumefile>=0.2.3 in /home/mick/bin/anaconda/envs/scratch
Requirement already satisfied: inflate64<1.1.0,>=1.0.0 in /home/mick/bin/anaconda/envs/scrat
Requirement already satisfied: brotli>=1.1.0 in /home/mick/bin/anaconda/envs/scratch/lib/pyth
Requirement already satisfied: psutil in /home/mick/bin/anaconda/envs/scratch/lib/python3.12/
Requirement already satisfied: wordcloud in /home/mick/bin/anaconda/envs/scratch/lib/python3
```

```
Requirement already satisfied: numpy>=1.6.1 in /home/mick/bin/anaconda/envs/scratch/lib/pytho
Requirement already satisfied: pillow in /home/mick/bin/anaconda/envs/scratch/lib/python3.12/
Requirement already satisfied: matplotlib in /home/mick/bin/anaconda/envs/scratch/lib/python3
Requirement already satisfied: contourpy>=1.0.1 in /home/mick/bin/anaconda/envs/scratch/lib/p
Requirement already satisfied: cycler>=0.10 in /home/mick/bin/anaconda/envs/scratch/lib/pytho
Requirement already satisfied: fonttools>=4.22.0 in /home/mick/bin/anaconda/envs/scratch/lib/
Requirement already satisfied: kiwisolver>=1.3.1 in /home/mick/bin/anaconda/envs/scratch/lib/
Requirement already satisfied: packaging>=20.0 in /home/mick/bin/anaconda/envs/scratch/lib/py
Requirement already satisfied: pyparsing>=2.3.1 in /home/mick/bin/anaconda/envs/scratch/lib/p
Requirement already satisfied: python-dateutil>=2.7 in /home/mick/bin/anaconda/envs/scratch/l
Requirement already satisfied: six>=1.5 in /home/mick/bin/anaconda/envs/scratch/lib/python3.1
Requirement already satisfied: pycountry in /home/mick/bin/anaconda/envs/scratch/lib/python3
```

```python
def download_file(url, output_path):
  """
    Check if output_path exists. If not download from url to output_path
  """
  if os.path.exists(output_path):
      print(f"File '{output_path}' already exists. Skipping download.")
      return

  print(f"Downloading file from {url}...")
  try:
      response = requests.get(url, stream=True)
      response.raise_for_status()

      with open(output_path, 'wb') as file:
          for chunk in response.iter_content(chunk_size=8192):
              file.write(chunk)
      print(f"Download completed: {output_path}")
  except requests.exceptions.RequestException as e:
      print(f"Failed to download file: {e}")


archive_file = "datascience.stackexchange.com.7z"
download_file("https://archive.org/download/stackexchange/datascience.stackexchange.com.7z",


# Extract the files to current directory
with py7zr.SevenZipFile(archive_file, 'r') as archive:
    for file in archive.getnames():
        if not os.path.exists(file):
```

```
            print(f"Extracting {file}...")
            archive.extract(targets=[file])
            archive.reset()
```

```
File 'datascience.stackexchange.com.7z' already exists. Skipping download.
```

Conversion of XML to CSV

Iterate through all XML files and convert them to CSV

```
def convert_all_xml_files_to_csv():
    """
    Iterate through all files in the current working directory ending with '.xml' and transfo
    """
    for filename in os.listdir():
        if filename.endswith('.xml'):
            csv_filename = filename.replace('.xml', '.csv')
            if os.path.exists(csv_filename):
                print(f"CSV file '{csv_filename}' already exists. Skipping conversion.")
                continue
            print(f"Converting {filename} to CSV...")
            tree = ET.parse(filename)
            root = tree.getroot()
            rows = root.findall('.//row')
            if rows:
                csv_data = []
                for item in rows:
                    row_data = {}
                    for attr in item.attrib:
                        row_data[attr] = item.attrib[attr]
                    csv_data.append(row_data)
                df = pd.DataFrame(csv_data)
                df.to_csv(csv_filename, index=False)
            else:
                print(f"No <row> tags found in {filename}. Skipping conversion.")

convert_all_xml_files_to_csv()
```

```
CSV file 'Tags.csv' already exists. Skipping conversion.
CSV file 'PostLinks.csv' already exists. Skipping conversion.
```

```
CSV file 'PostHistory.csv' already exists. Skipping conversion.
CSV file 'Votes.csv' already exists. Skipping conversion.
CSV file 'Comments.csv' already exists. Skipping conversion.
CSV file 'Badges.csv' already exists. Skipping conversion.
CSV file 'Posts.csv' already exists. Skipping conversion.
CSV file 'Users.csv' already exists. Skipping conversion.
```

Display of DataFrame Samples

Load badges

```
badges_df = pd.read_csv('Badges.csv')
badges_df.head()
```

	Id	UserId	Name	Date	Class	TagBased
0	28	26	Teacher	2014-05-14T00:00:58.947	3	False
1	29	36	Teacher	2014-05-14T00:00:58.947	3	False
2	46	34	Teacher	2014-05-14T00:17:25.757	3	False
3	55	22	Teacher	2014-05-14T00:54:48.590	3	False
4	56	51	Teacher	2014-05-14T00:54:48.590	3	False

Load the comments

```
comments_df = pd.read_csv('Comments.csv')
comments_df.head()
```

	Id	PostId	Score	Text	CreationDate	UserId
0	5	5	9	this is a super theoretical AI question. An in...	2014-05-14T00:23:15.437	34.0
1	6	7	4	List questions are usually not suited for Stac...	2014-05-14T00:38:19.510	51.0
2	9	7	3	This question appears to be off-topic because ...	2014-05-14T01:16:12.623	66.0
3	14	7	0	Fair enough regarding what constitutes a ”vali...	2014-05-14T02:35:50.090	36.0
4	80	7	1	@statsRus: Try posting a question like that to...	2014-05-15T21:08:13.933	158.0

Load the post history

```
post_history_df = pd.read_csv('PostHistory.csv', low_memory=False)
post_history_df.head()
```

	Id	PostHistoryTypeId	PostId	RevisionGUID	CreationDate	U
0	7.0	2	5	009bca93-fce2-44ed-a277-a8452650a627	2014-05-13T23:58:30.457	5
1	8.0	1	5	009bca93-fce2-44ed-a277-a8452650a627	2014-05-13T23:58:30.457	5
2	9.0	3	5	009bca93-fce2-44ed-a277-a8452650a627	2014-05-13T23:58:30.457	5
3	12.0	2	7	ea5a5642-ed30-43ea-9be5-8e8de0e1c660	2014-05-14T00:11:06.457	3
4	13.0	1	7	ea5a5642-ed30-43ea-9be5-8e8de0e1c660	2014-05-14T00:11:06.457	3

Load the Post Links

```
post_links_df = pd.read_csv('PostLinks.csv')
post_links_df.head()
```

	Id	CreationDate	PostId	RelatedPostId	LinkTypeId
0	1844230	2023-10-07T19:48:58.280	14324	123931	1
1	343588	2018-03-07T05:57:10.740	28719	26874	1
2	426221	2018-05-14T13:08:38.173	31636	28537	1
3	1317329	2021-05-06T22:21:14.737	88410	94071	1
4	1607825	2022-09-07T15:45:14.770	114212	54908	1

Load the Posts

```
posts_df = pd.read_csv('Posts.csv')
posts_df.head()
```

	Id	PostTypeId	CreationDate	Score	ViewCount	Body
0	5	1	2014-05-13T23:58:30.457	9	968.0	<p>I've always been interested in mach
1	7	1	2014-05-14T00:11:06.457	4	510.0	<p>As a researcher and instructor, I'm
2	9	2	2014-05-14T00:36:31.077	5	NaN	<p>Not sure if this fits the scope of th
3	10	2	2014-05-14T00:53:43.273	13	NaN	<p>One book that's freely available is
4	14	1	2014-05-14T01:25:59.677	26	1931.0	<p>I am sure data science as will be d

Load the Tags

```
tags_df = pd.read_csv('Tags.csv')
tags_df.head()
```

	Id	TagName	Count	ExcerptPostId	WikiPostId
0	1	definitions	39	105.0	104.0
1	2	machine-learning	11403	4909.0	4908.0
2	3	bigdata	456	66.0	65.0
3	5	data-mining	1181	80.0	79.0
4	6	databases	100	8960.0	8959.0

Load the Users

```
users_df = pd.read_csv('Users.csv')
users_df.head()
```

	Id	Reputation	CreationDate	DisplayName	LastAccessDate	WebsiteUrl
0	-1	1	2014-05-13T21:29:22.820	Community	2014-05-13T21:29:22.820	http://meta.stacke
1	1	101	2014-05-13T22:58:54.810	Adam Lear	2024-03-11T21:29:14.793	NaN
2	2	101	2014-05-13T22:59:19.787	Geoff Dalgas	2019-09-03T19:10:22.217	http://stackoverflo
3	3	101	2014-05-13T23:15:34.483	hichris123	2020-12-16T17:41:49.610	NaN
4	4	101	2014-05-13T23:16:09.937	Ben Collins	2014-08-04T15:25:54.810	http://benjamincol

Load the Votes

```
votes_df = pd.read_csv('Votes.csv')
votes_df.head()
```

	Id	PostId	VoteTypeId	CreationDate	UserId	BountyAmount
0	1	1	2	2014-05-13T00:00:00.000	NaN	NaN
1	2	1	2	2014-05-13T00:00:00.000	NaN	NaN
2	6	1	2	2014-05-13T00:00:00.000	NaN	NaN
3	20	1	2	2014-05-14T00:00:00.000	NaN	NaN
4	44	1	2	2014-05-14T00:00:00.000	NaN	NaN

Data Analysis

This section analyses the Data Science Stack Exchange data with a focus on the topics raised in the introduction. We will refer to Data Science Stack Exchange as DSSE from hereon in.

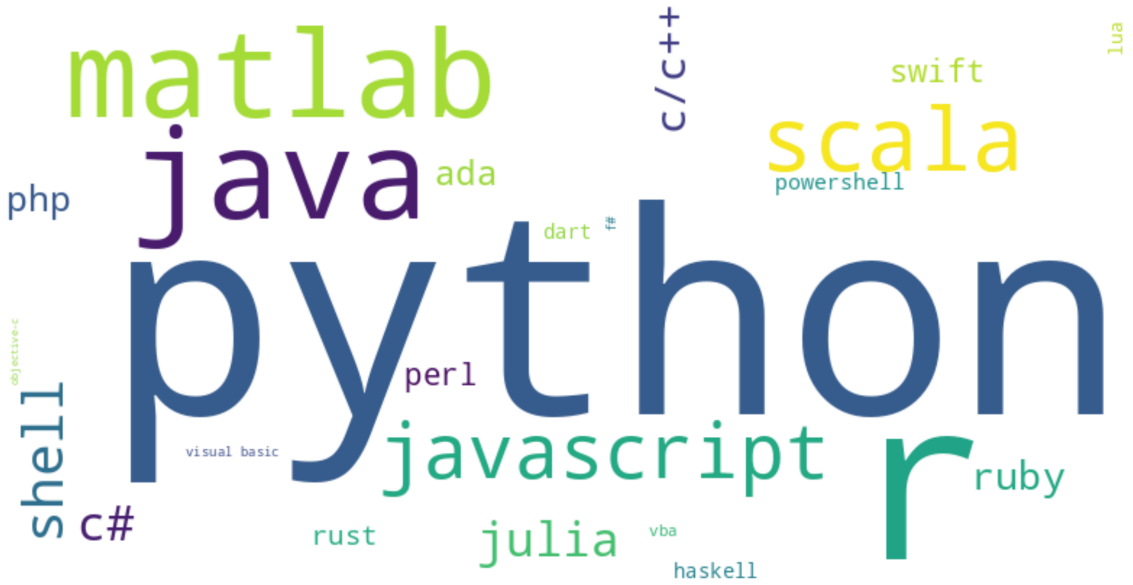
Languages Used in Data Science

The DSSE data presents an excellent opportunity to discover trends in the Data Science field. Here we investigate which computer languages are the most mentioned in DSSE posts. Regular expressions were employed to search posts for a set of predefined computer languages

```
def clean_body_text(body_text, regex_pattern):
    """
    Remove all strings from 'body_text' except those that match 'regex_pattern'.
    Separate all matches with ' & '
    """

    if not isinstance(body_text, str):
        return ''
    matches = re.findall(regex_pattern, body_text)
    return ' & '.join(matches)

def get_phrase_frequencies(regex_word_array, df: pd.DataFrame, post_year=None):
    """
    Search the Body column of the 'df' dataframe for anything matching 'regex_word_array'.
    Constrain the search to posts with the year 'post_year' if defined.
    """
    if post_year:
        df = df[df['CreationDate'].str.contains(post_year)]

    pattern = r'(?i)(?<=\s)(' + '|'.join(map(re.escape, regex_word_array)) + r')(?=\s)'
    all_text = df['Body'].str.cat(sep=' & ')
    language_text = clean_body_text(all_text, pattern)

    phrase_freq = {}
    for phrase in language_text.split(' & '):
        phrase = phrase.lower()
        if phrase in phrase_freq:
            phrase_freq[phrase] += 1
        else:
            phrase_freq[phrase] = 1
    return phrase_freq

def produce_word_cloud(regex_word_array, df: pd.DataFrame, post_year=None):
    """
    Display a word cloud for posts.
    """
    phrase_frequencies = get_phrase_frequencies(regex_word_array, df, post_year)
```

```
    # Display the word cloud
    wordcloud = WordCloud(width=800, height=400, background_color='white').generate_from_fre
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis("off")
    plt.show()

    # print top 10
    phrase_df = pd.DataFrame(list(phrase_frequencies.items()), columns=['Word', 'Frequency']
    phrase_df.set_index('Word', inplace=True)
    print(tabulate.tabulate(phrase_df, headers='keys', tablefmt='psql'))
```

```
# Sourced from https://pypl.github.io/PYPL.html
languages_list  = [
    "Python", "Java", "JavaScript", "C#", "C/C++", "PHP", "R", "TypeScript",
    "Swift", "Objective-C", "Kotlin", "Matlab", "VBA", "Rust", "Ruby",
    "Ada", "Scala", "Dart", "Perl", "Lua", "Visual Basic", "Julia", "Haskell",
    "Delphi/Object Pascal", "COBOL", "Shell", "PowerShell", "F#", "Assembly Language"
]

produce_word_cloud(languages_list, posts_df)
```




```
+------------+-------------+
```

Word	Frequency
python	4768
r	2822
java	296
matlab	277
scala	132
javascript	91
shell	42
c#	32
julia	32
c/c++	26

It can be observed from this that Python and R and the two most mentioned computer languages. They far exceed the number of mentions attributed to the third most mentioned language, Java. This indicates that the field has coalesced around these two languages.

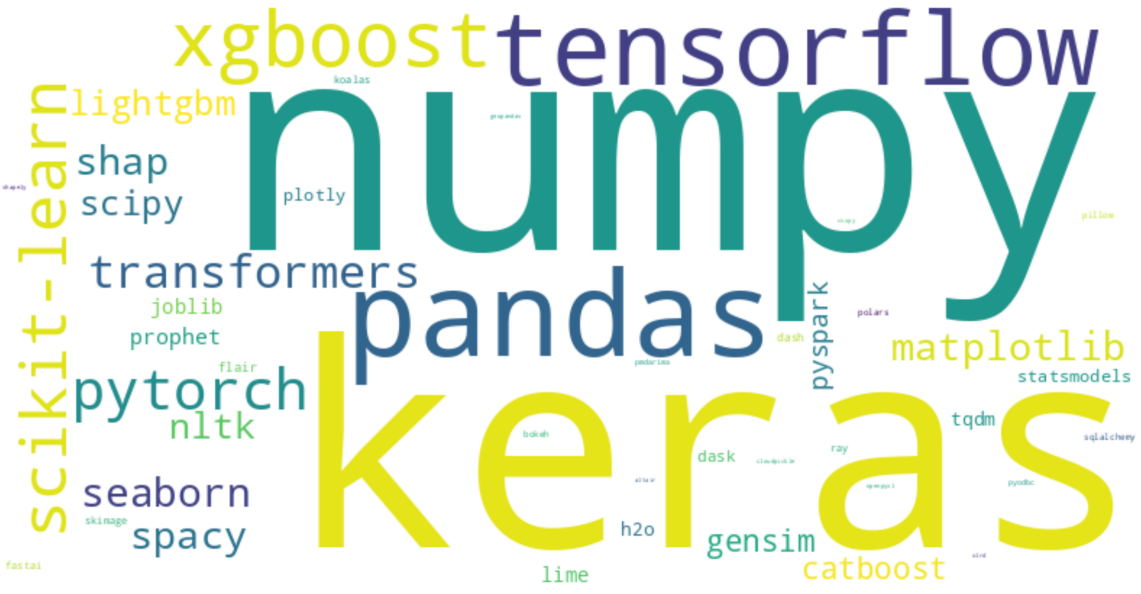
Popular Python Libraries

Python's popularity in Data Science can be attributed, in part, to its rich and varied ecosystem of libraries. The following word cloud and frequency table reveal exactly which python libraries are referenced most commonly in posts.

```
# Python datascience libraries produced via generative AI
data_science_libraries = [
    # General Purpose Libraries
    "numpy",       # Numerical computing
    "pandas",      # Data manipulation and analysis
    "matplotlib",  # Data visualization
    "seaborn",     # Statistical data visualization
    "scipy",       # Scientific computing
    "statsmodels", # Statistical modeling and testing

    # Machine Learning Libraries
    "scikit-learn",    # General-purpose machine learning
    "xgboost",         # Gradient boosting for structured data
    "lightgbm",        # Light Gradient Boosting Machine
    "catboost",        # Categorical data boosting
    "tensorflow",      # Deep learning
    "keras",           # High-level API for TensorFlow
    "pytorch",         # Deep learning
```

```
    "fastai",          # High-level deep learning library
    "h2o",             # Machine learning and AI platform

    # Natural Language Processing Libraries
    "nltk",            # Natural Language Toolkit
    "spacy",           # Industrial-strength NLP
    "gensim",          # Topic modeling and document similarity
    "transformers",    # State-of-the-art NLP (Hugging Face)
    "flair",           # Simple NLP library

    # Data Visualization Libraries
    "bokeh",           # Interactive visualizations
    "plotly",          # Interactive and web-based plots
    "altair",          # Declarative statistical visualization
    "dash",            # Web-based data dashboards

    # Data Handling and Transformation
    "pyjanitor",       # Data cleaning and preprocessing
    "pyarrow",         # In-memory columnar storage
    "dask",            # Parallel computing and large datasets
    "polars",          # Fast DataFrame library
    "modin",           # Accelerated pandas

    # Time Series Analysis
    "tslearn",         # Time series machine learning
    "prophet",         # Forecasting tool
    "pmdarima",        # Auto ARIMA for time series
    "statsforecast",   # Forecasting models

    # Specialized Libraries
    "cvxpy",           # Convex optimization
    "pulp",            # Linear programming
    "geopandas",       # Geospatial data analysis
    "pyproj",          # Geospatial projections
    "shapely",         # Geometric objects and operations

    # Evaluation and Interpretation
    "shap",            # Explain model predictions
    "lime",            # Model interpretability
    "yellowbrick",     # Model visualization

    # Data Input/Output
```

```
    "openpyxl",         # Excel file handling
    "xlrd",             # Excel file handling
    "pyodbc",           # Database connections
    "sqlalchemy",       # SQL toolkit and ORM

    # Distributed and Big Data Libraries
    "pyspark",          # Apache Spark interface
    "koalas",           # pandas API on Spark
    "ray",              # Distributed computing

    # Other Utilities
    "missingno",        # Visualization for missing data
    "skimage",          # Image processing
    "Pillow",           # Image manipulation
    "joblib",           # Parallel computing and job management
    "cloudpickle",      # Serialization for distributed systems
    "tqdm",             # Progress bars
]

produce_word_cloud(data_science_libraries, posts_df)
```



Word	Frequency

numpy	3205
keras	2979
pandas	2913
tensorflow	2571
xgboost	1295
scikit-learn	869
pytorch	685
transformers	449
seaborn	336
shap	306

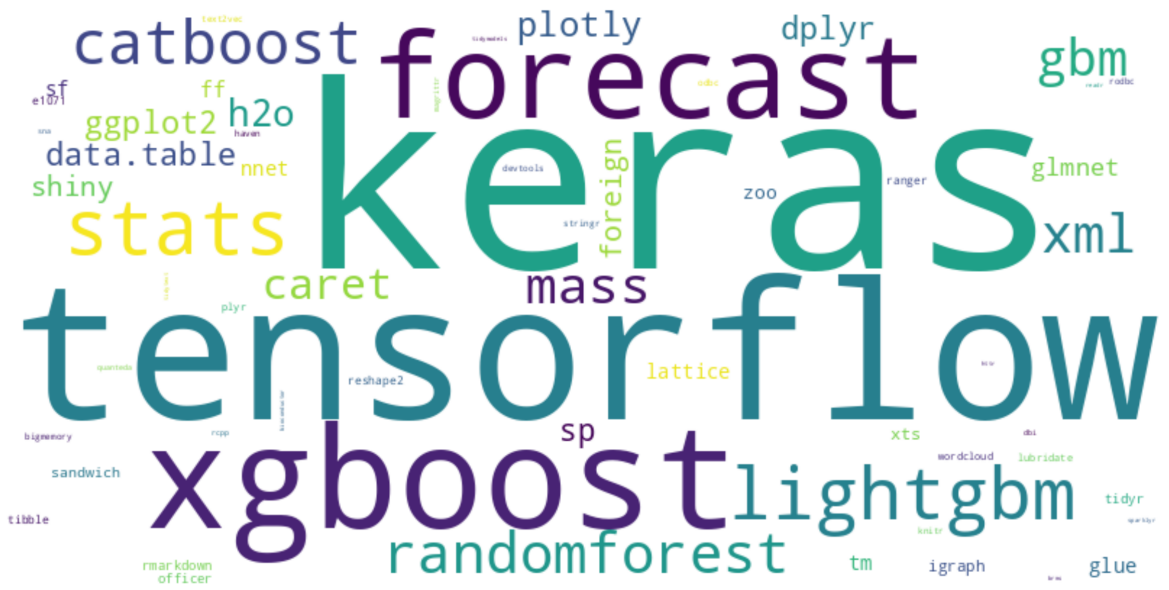
The most popular libraries are a mix of libraries used for data manipulation purposes and machine learning model creation.

Popular R DataScience Libraries

R is a popular choice in Data Science for statistical analysis and visualisation. The following word cloud and frequency table show which libraries are most referenced in posts.

```
# R library array produced via geneartive AI
r_data_science_libraries = [
    # Data Manipulation
    "dplyr", "data.table", "tidyr", "plyr", "reshape2", "readr", "tibble",

    # Visualization
    "ggplot2", "lattice", "plotly", "shiny", "leaflet", "highcharter",
    "visNetwork", "ggvis", "ggforce", "esquisse",

    # Statistical Modeling
    "stats", "lmtest", "sandwich", "caret", "lme4", "nlme", "randomForest",
    "e1071", "MASS", "nnet", "forecast", "fable", "xgboost", "glmnet",

    # Machine Learning
    "mlr3", "tidymodels", "caret", "e1071", "kernlab", "xgboost",
    "randomForest", "ranger", "lightgbm", "catboost", "gbm",
    "h2o", "tensorflow", "keras",

    # Time Series
    "forecast", "fable", "tseries", "zoo", "xts", "lubridate",

    # Text Analysis
```

```
    "tm", "quanteda", "tidytext", "text2vec", "RTextTools", "wordcloud",

    # Data Import/Export
    "readr", "readxl", "openxlsx", "haven", "foreign", "jsonlite",
    "httr", "XML", "DBI", "RSQLite", "RODBC", "odbc",

    # Big Data
    "sparklyr", "daskr", "bigmemory", "ff", "ffbase", "data.table",

    # Reporting and Dashboarding
    "shiny", "flexdashboard", "rmarkdown", "knitr", "officer",
    "rmdformats", "pagedown",

    # Geospatial Analysis
    "sf", "sp", "rgdal", "rgeos", "maptools", "leaflet", "tmap",

    # Bayesian Analysis
    "rstan", "brms", "coda", "BayesFactor",

    # Bioinformatics
    "Bioconductor", "Biobase", "GenomicRanges", "edgeR", "DESeq2",

    # Network Analysis
    "igraph", "ggraph", "networkD3", "statnet", "sna",

    # Utilities and Miscellaneous
    "magrittr", "stringr", "purrr", "lubridate", "glue", "forcats",
    "Rcpp", "devtools", "usethis"
]

produce_word_cloud(r_data_science_libraries, posts_df)
```



Word	Frequency
keras	2979
tensorflow	2571
xgboost	1295
forecast	847
stats	260
lightgbm	240
catboost	167
randomforest	145
xml	133
mass	129

Whilst Python had quite a few libraries dedicated to data manipulation, R does not have the same in its most popular referred to library list. This is due to most of the data manipulation features being baked into the R language itself. The most popular R libraries were mostly machine learning model building libraries.

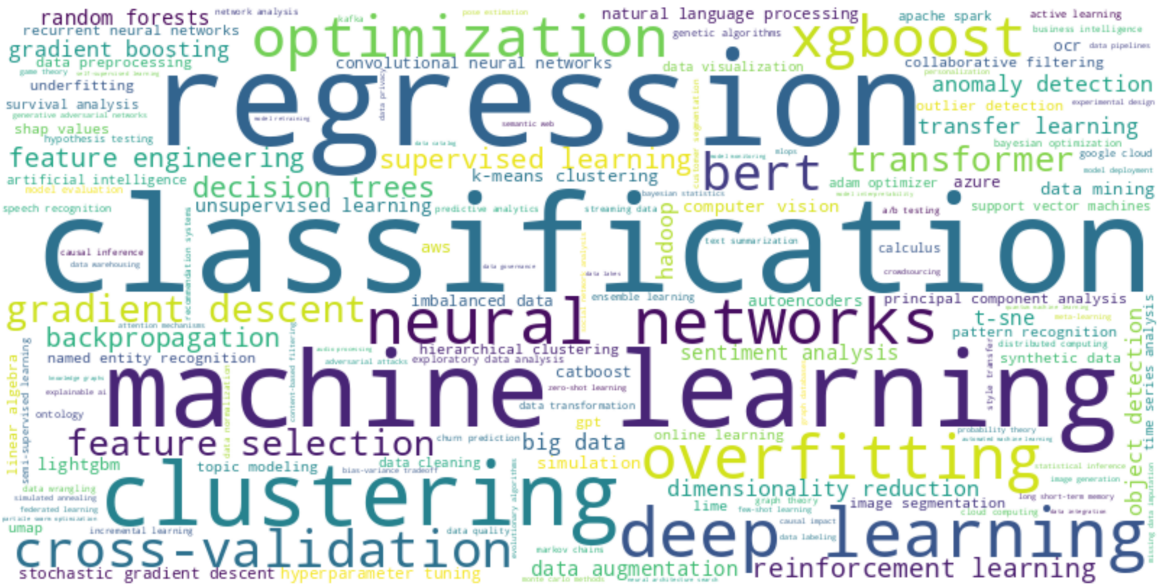
Popular Data Science Topics

Arguably one of the most useful analyses that can be performed on stack exchange datasets is determining popular topics under discussion. Utilising a predefined list of data science topics,

the most popular topics are discovered.

```
# Topics list produced using generative AI
data_science_topics = [
    "Machine Learning", "Deep Learning", "Artificial Intelligence", "Neural Networks",
    "Natural Language Processing", "Computer Vision", "Reinforcement Learning",
    "Supervised Learning", "Unsupervised Learning", "Semi-Supervised Learning",
    "Clustering", "Classification", "Regression", "Dimensionality Reduction",
    "Feature Engineering", "Feature Selection", "Data Preprocessing", "Data Cleaning",
    "Data Wrangling", "Data Visualization", "Exploratory Data Analysis", "Big Data",
    "Data Mining", "Predictive Analytics", "Prescriptive Analytics", "Descriptive Analytics"
    "Time Series Analysis", "Anomaly Detection", "Pattern Recognition", "Optimization",
    "Hyperparameter Tuning", "Model Evaluation", "Cross-Validation", "Bias-Variance Tradeoff
    "Overfitting", "Underfitting", "Ensemble Learning", "Random Forests", "Gradient Boosting
    "XGBoost", "LightGBM", "CatBoost", "Support Vector Machines", "Decision Trees",
    "K-Means Clustering", "Hierarchical Clustering", "Principal Component Analysis",
    "t-SNE", "UMAP", "Autoencoders", "Generative Adversarial Networks", "Transfer Learning",
    "Explainable AI", "Model Interpretability", "SHAP Values", "LIME", "A/B Testing",
    "Hypothesis Testing", "Statistical Inference", "Bayesian Statistics", "Probability Theory
    "Linear Algebra", "Calculus", "Optimization Algorithms", "Gradient Descent",
    "Stochastic Gradient Descent", "Adam Optimizer", "Backpropagation", "Convolutional Neural
    "Recurrent Neural Networks", "Long Short-Term Memory", "Transformer", "BERT",
    "GPT", "Attention Mechanisms", "Self-Supervised Learning", "Few-Shot Learning",
    "Zero-Shot Learning", "Edge AI", "Federated Learning", "Quantum Machine Learning",
    "Ethical AI", "AI Fairness", "Bias in AI", "Data Privacy", "GDPR Compliance",
    "Data Governance", "Data Lakes", "Data Warehousing", "ETL Pipelines", "ELT Pipelines",
    "Streaming Data", "Real-Time Analytics", "IoT Analytics", "Cloud Computing",
    "AWS", "Google Cloud", "Azure", "Distributed Computing", "Apache Spark", "Hadoop",
    "Kafka", "DataOps", "MLOps", "Model Deployment", "Model Monitoring", "Model Retraining",
    "CI/CD for ML", "Data Pipelines", "Feature Stores", "Model Serving", "APIs for ML",
    "RESTful APIs", "GraphQL", "Graph Databases", "Knowledge Graphs", "Ontology",
    "Semantic Web", "Data Ethics", "Data Security", "Cybersecurity in AI", "Adversarial Attac
    "Robust AI", "Causal Inference", "Counterfactual Analysis", "Experimental Design",
    "Simulation", "Monte Carlo Methods", "Markov Chains", "Game Theory", "Reinforcement Learn
    "Multi-Agent Systems", "Swarm Intelligence", "Evolutionary Algorithms", "Genetic Algorith
    "Particle Swarm Optimization", "Simulated Annealing", "Bayesian Optimization",
    "Active Learning", "Online Learning", "Incremental Learning", "Lifelong Learning",
    "Meta-Learning", "Neural Architecture Search", "Automated Machine Learning",
    "Data Labeling", "Crowdsourcing", "Data Annotation", "Synthetic Data", "Data Augmentation
    "Imbalanced Data", "Outlier Detection", "Missing Data Imputation", "Data Normalization",
    "Data Standardization", "Data Transformation", "Data Integration", "Data Fusion",
    "Data Lineage", "Data Catalog", "Data Quality", "Data Profiling", "Data Validation",
    "Data Versioning", "Data Storytelling", "Business Intelligence", "Dashboarding",
```

```
    "Geospatial Analysis", "Spatiotemporal Data", "Network Analysis", "Graph Theory",
    "Social Network Analysis", "Recommendation Systems", "Collaborative Filtering",
    "Content-Based Filtering", "Hybrid Recommenders", "Personalization", "Customer Segmentati
    "Churn Prediction", "Lifetime Value Prediction", "Sentiment Analysis", "Topic Modeling",
    "Text Summarization", "Named Entity Recognition", "Speech Recognition", "OCR",
    "Image Segmentation", "Object Detection", "Pose Estimation", "Style Transfer",
    "Image Generation", "Video Analysis", "Audio Processing", "Time-to-Event Analysis",
    "Survival Analysis", "Cohort Analysis", "A/B Testing", "Multi-Armed Bandits",
    "Causal Impact", "Root Cause Analysis", "Data-Driven Decision Making", "Digital Transfor
]

produce_word_cloud(data_science_topics, posts_df)
```



Word	Frequency
classification	8329
regression	7839
machine learning	4990
clustering	3156
deep learning	1861
neural networks	1601
overfitting	1457
optimization	1330

```
| xgboost          |        1295 |
| cross-validation |        1075 |
+------------------+-------------+
```

The most discussed and used topics on DSSE are mostly topics have been part of the data science field for quite some time and show the prominence of these topcics in the field. Those topics include classification, regression, machine learning, clustering, neural networks, overfitting and optimisation. There is one topic that have come into the data science venacular only recently, that being deep learning. This is a testiment to its rise to importance.

The topics of optimisation and cross validation also show the community's focus on improving model perfomance and reliability. The inclusion of the clustering topic in the top 10 topics indicates that unsupervised methods attract a lot of attention.

Popular Data Science Topics by Year

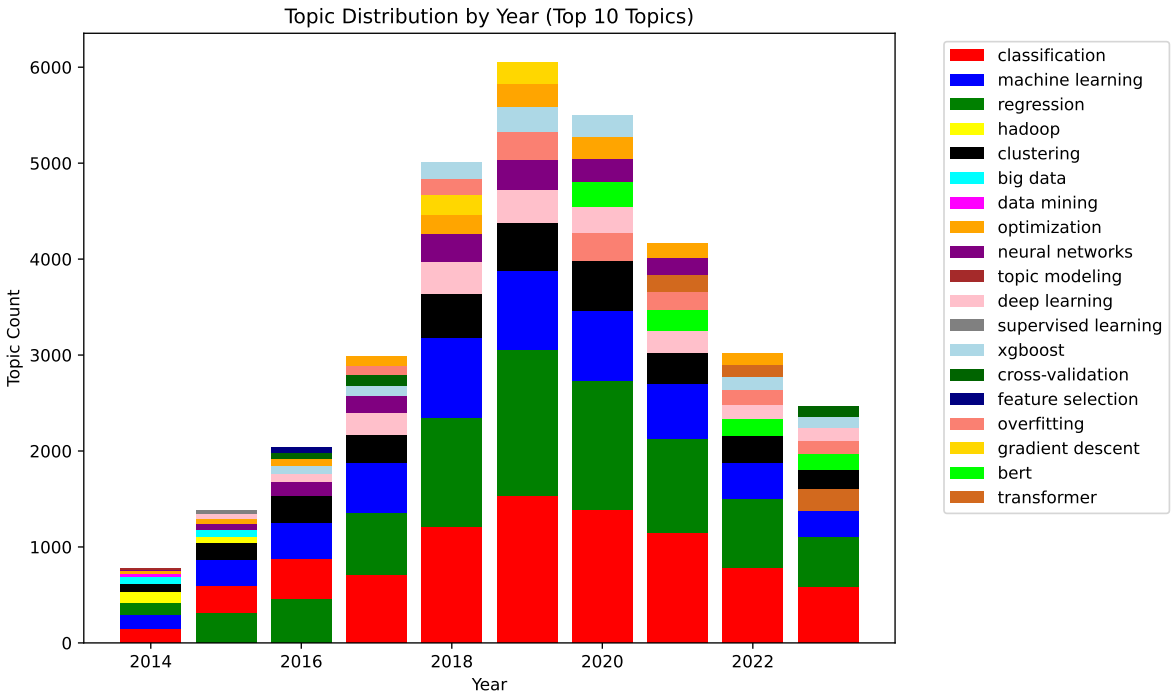
The top 10 data science topics of all time are interesting but they do not show temporal changes to the significance of data science topics. Producing an annual top 10 data science topics for each year between 2014 and 2023 will better allow use to view these changes. Analysising this change will provide an insight into the evolving focus of the field.

```
# get topic phase frequecies from 2014 to 2023

results = []

for year in range(2013, 2024):
    phrase_frequencies = get_phrase_frequencies(data_science_topics, posts_df, str(year))
    year_results = [{'Year': year, 'Topic': topic, 'Frequency': frequency} for topic, frequer
    results.extend(year_results)
results_df = pd.DataFrame(results)
```

```
years = sorted(results_df['Year'].unique())
cumulative_frequencies = {}
top_topics_per_year = {}

for year in years:
    top_topics = results_df[results_df['Year'] == year].nlargest(10, 'Frequency')['Topic'].to
    top_topics_per_year[year] = top_topics
    cumulative_frequencies[year] = {topic: 0 for topic in top_topics}
    bottom = 0
    for topic in top_topics:
        frequency = results_df[(results_df['Year'] == year) & (results_df['Topic'] == topic)]
```

```
        cumulative_frequencies[year][topic] = frequency + bottom
        bottom += frequency

topic_colours = {
    'classification': 'red',
    'machine learning': 'blue',
    'regression': 'green',
    'hadoop': 'yellow',
    'clustering': 'black',
    'big data': 'cyan',
    'data mining': 'magenta',
    'optimization': 'orange',
    'neural networks': 'purple',
    'topic modeling': 'brown',
    'deep learning': 'pink',
    'supervised learning': 'grey',
    'xgboost': 'lightblue',
    'cross-validation': 'darkgreen',
    'feature selection': 'navy',
    'overfitting': 'salmon',
    'gradient descent': 'gold',
    'bert': 'lime',
    'transformer': 'chocolate'
}


fig, ax = plt.subplots(figsize=(10, 6))

for year in years:
    bottom = 0
    for topic in top_topics_per_year[year]:
        if topic == '':
            continue
        ax.bar(year, cumulative_frequencies[year][topic] - bottom, label=topic, bottom=botto
        bottom = cumulative_frequencies[year][topic]

ax.set_xlabel('Year')
ax.set_ylabel('Topic Count')
ax.set_title('Topic Distribution by Year (Top 10 Topics)')
handles, labels = ax.get_legend_handles_labels()
unique_labels = list(dict.fromkeys(labels))
unique_handles = [handles[labels.index(label)] for label in unique_labels]
```

```
ax.legend(unique_handles, unique_labels, bbox_to_anchor=(1.05, 1), loc='upper left')

plt.tight_layout()
plt.show()
```

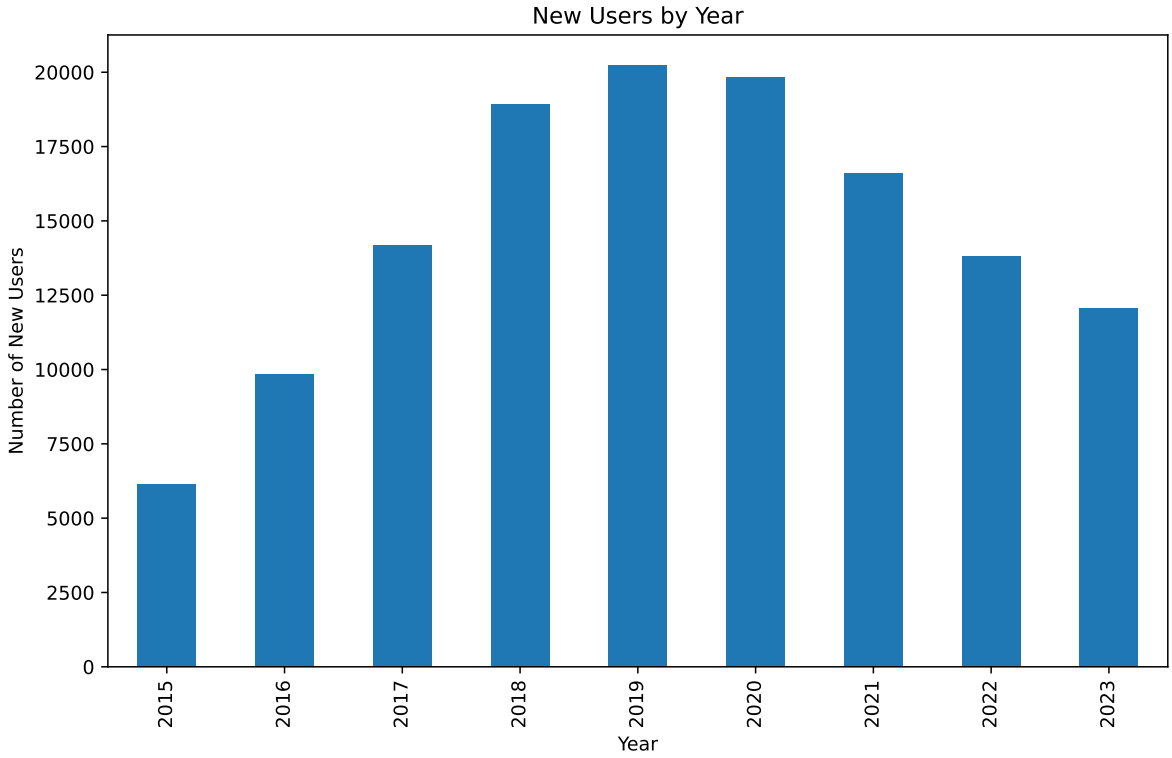


From the Tag Distribution by Year (Top 10 Topics) we can see that the topics of classification, regression and machine learning dominate the discussions on DSSE consistently throughout the years. This observation aligns witht he central role these methods play in data science applications. The rise of topics of deep learning and transformer in recent years illustrates the community's shift in discussions in response to technological advancements and emerging techniques. From 2020 onwards, BERT rose to prominence as a hot topic highlighting the field's advancements in Natural Language Processing (NLP).

New Users by Year

DSSE has been an invaluable space for users to share knowledge and experience. In this section we would like to determine if the site's popularity has increased or decreased. The number of new users per year will allow us to see a trend in either direction.

```
users_df['CreationDate'] = pd.to_datetime(users_df['CreationDate'])
filtered_users_df = users_df[(users_df['CreationDate'].dt.year >= 2015) & (users_df['Creatio
new_users_by_year = filtered_users_df.groupby(filtered_users_df['CreationDate'].dt.year)['Ac

new_users_by_year.plot(kind='bar', figsize=(10, 6))
plt.xlabel('Year')
plt.ylabel('Number of New Users')
plt.title('New Users by Year')
plt.show()
```



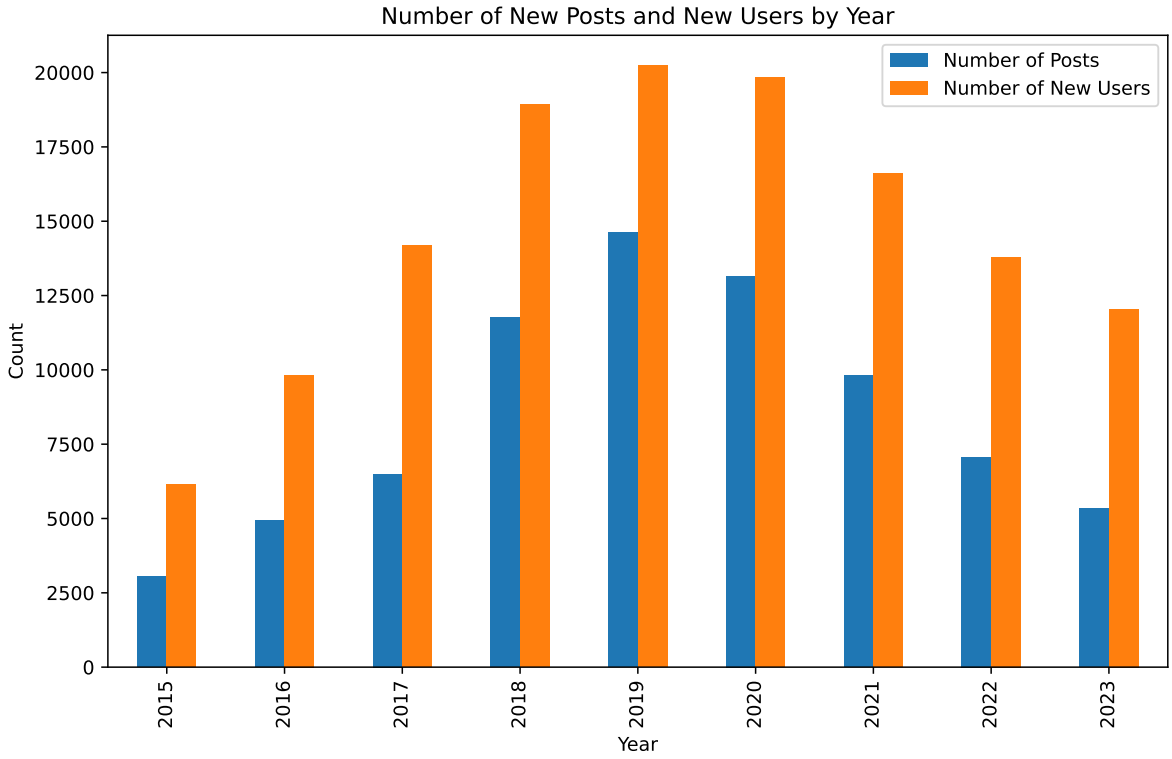
The bar chart of New Users by Year demonstrates the growth trajectory of the DSSE community. Between 2015 and 2020, the number of new users increased significantly, indicating hightened interest in data science. However from 2020 onwards there is a stark decline in growth in the number of new users. We will investigate this further.

Number of Posts by Year

As there was a decline in the number of new users from 2020 onwards, we would like to know if there was a decline in user engagement as well. User engagement can be determined from

the number of posts users are making over a certain period of time. We will analyse the post count per year to resolve this question.

```
posts_df['CreationDate'] = pd.to_datetime(posts_df['CreationDate'])
filtered_posts_df = posts_df[(posts_df['CreationDate'].dt.year >= 2015) & (posts_df['Creatio
posts_by_year = filtered_posts_df.groupby(filtered_posts_df['CreationDate'].dt.year).size()


# Combine the number of new posts and new users by year
combined_data = pd.concat([posts_by_year, new_users_by_year], axis=1)
combined_data.columns = ['Number of Posts', 'Number of New Users']

combined_data.plot(kind='bar', figsize=(10, 6))
plt.xlabel('Year')
plt.ylabel('Count')
plt.title('Number of New Posts and New Users by Year')
plt.legend()
plt.show()
```



In the bar chart Number of New Posts and New Users by Year, the number of new posts was clearly increasing from 2015 to 2019 and then clearnly declines from 2020 onwards. This is an

identical declining trend in new posts as there is in new users, both starting in 2020. There have been many theories postulated for this decline ranging from competing educational platforms to behavioural changes during the COVID pandemic. One theory backed by research [3] is that the introduction of Large Language Models, beginning with OpenAI's ChatGPT in 2022, have further increased the decline in user engagement in question and answering platforms as these models often provide accurate answers to question in less time that is required to search Q&A platforms.

World User Count HeatMap

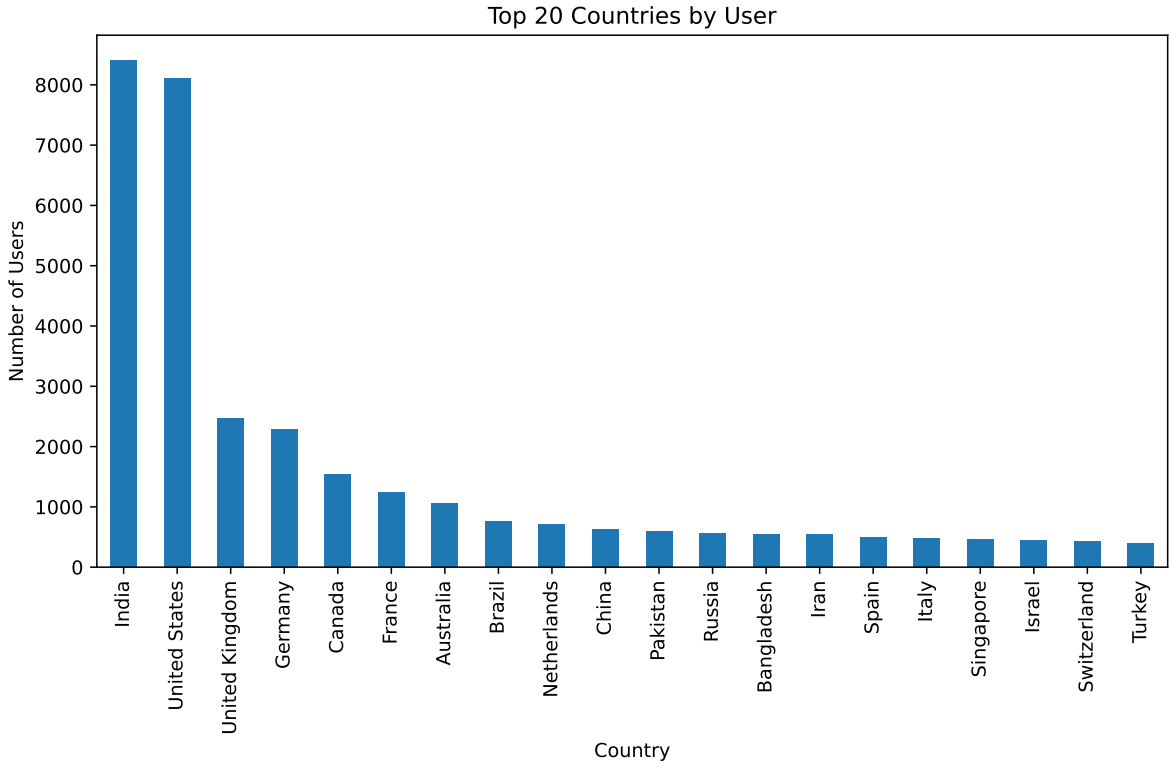
As DSSE has undergone a period of decline, segmentation analysis of its user population would be useful. One analysis that may be prudent is to understand the geographical dispersion of users.

```
def resolve_country_abbreviation(text):
    """
    Convert common country abbreviations to their full official names.
        Input - text (str): Input text containing country abbreviations
        Returns - str: Text with abbreviations replaced by full country names
    """
    country_abbreviations = {
        'UK': 'United Kingdom',
        'GB': 'United Kingdom',
        'US': 'United States',
        'USA': 'United States',
        'UAE': 'United Arab Emirates',
        'KSA': 'Saudi Arabia',
        'RSA': 'South Africa',
        'PRC': 'China',
        'ROK': 'South Korea',
        'DPRK': 'North Korea',
        'NZ': 'New Zealand',
        'UAE': 'United Arab Emirates',
        'KSA': 'Saudi Arabia',
        'RSA': 'South Africa',
    }

    def replace_abbreviation(match):
        """Helper function to handle case-sensitive replacement"""
        abbr = match.group(0)
        if abbr in country_abbreviations:
            return country_abbreviations[abbr]
```

```
        upper_abbr = abbr.upper()
        if upper_abbr in country_abbreviations:
            return country_abbreviations[upper_abbr]
        return abbr

    pattern = r'\b(' + '|'.join(map(re.escape, country_abbreviations.keys())) + r')\b'
    result = re.sub(pattern, replace_abbreviation, text, flags=re.IGNORECASE)
    return result


def extract_countries(text):
    """
    Extract country names from a given text string using regular expressions.
        Inputs - text (str): Input text to search for country names
        Returns - str: First country name found in the text (in its proper case), or None if
    """
    countries = {
        'Afghanistan', 'Albania', 'Algeria', 'Andorra', 'Angola', 'Antigua and Barbuda',
        'Argentina', 'Armenia', 'Australia', 'Austria', 'Azerbaijan', 'Bahamas', 'Bahrain',
        'Bangladesh', 'Barbados', 'Belarus', 'Belgium', 'Belize', 'Benin', 'Bhutan',
        'Bolivia', 'Bosnia and Herzegovina', 'Botswana', 'Brazil', 'Brunei', 'Bulgaria',
        'Burkina Faso', 'Burundi', 'Cabo Verde', 'Cambodia', 'Cameroon', 'Canada',
        'Central African Republic', 'Chad', 'Chile', 'China', 'Colombia', 'Comoros',
        'Congo', 'Costa Rica', 'Croatia', 'Cuba', 'Cyprus', 'Czech Republic', 'Denmark',
        'Djibouti', 'Dominica', 'Dominican Republic', 'Ecuador', 'Egypt', 'El Salvador',
        'Equatorial Guinea', 'Eritrea', 'Estonia', 'Eswatini', 'Ethiopia', 'Fiji',
        'Finland', 'France', 'Gabon', 'Gambia', 'Georgia', 'Germany', 'Ghana', 'Greece',
        'Grenada', 'Guatemala', 'Guinea', 'Guinea-Bissau', 'Guyana', 'Haiti', 'Honduras',
        'Hungary', 'Iceland', 'India', 'Indonesia', 'Iran', 'Iraq', 'Ireland', 'Israel',
        'Italy', 'Jamaica', 'Japan', 'Jordan', 'Kazakhstan', 'Kenya', 'Kiribati',
        'Kuwait', 'Kyrgyzstan', 'Laos', 'Latvia', 'Lebanon', 'Lesotho', 'Liberia',
        'Libya', 'Liechtenstein', 'Lithuania', 'Luxembourg', 'Madagascar', 'Malawi',
        'Malaysia', 'Maldives', 'Mali', 'Malta', 'Marshall Islands', 'Mauritania',
        'Mauritius', 'Mexico', 'Micronesia', 'Moldova', 'Monaco', 'Mongolia',
        'Montenegro', 'Morocco', 'Mozambique', 'Myanmar', 'Namibia', 'Nauru', 'Nepal',
        'Netherlands', 'New Zealand', 'Nicaragua', 'Niger', 'Nigeria', 'North Korea',
        'North Macedonia', 'Norway', 'Oman', 'Pakistan', 'Palau', 'Palestine', 'Panama',
        'Papua New Guinea', 'Paraguay', 'Peru', 'Philippines', 'Poland', 'Portugal',
        'Qatar', 'Romania', 'Russia', 'Rwanda', 'Saint Kitts and Nevis', 'Saint Lucia',
        'Saint Vincent and the Grenadines', 'Samoa', 'San Marino', 'Sao Tome and Principe',
        'Saudi Arabia', 'Senegal', 'Serbia', 'Seychelles', 'Sierra Leone', 'Singapore',
        'Slovakia', 'Slovenia', 'Solomon Islands', 'Somalia', 'South Africa', 'South Korea',
```

```
        'South Sudan', 'Spain', 'Sri Lanka', 'Sudan', 'Suriname', 'Sweden', 'Switzerland',
        'Syria', 'Taiwan', 'Tajikistan', 'Tanzania', 'Thailand', 'Timor-Leste', 'Togo',
        'Tonga', 'Trinidad and Tobago', 'Tunisia', 'Turkey', 'Turkmenistan', 'Tuvalu',
        'Uganda', 'Ukraine', 'United Arab Emirates', 'United Kingdom', 'United States',
        'Uruguay', 'Uzbekistan', 'Vanuatu', 'Vatican City', 'Venezuela', 'Vietnam',
        'Yemen', 'Zambia', 'Zimbabwe'
    }

    pattern = r'\b(' + '|'.join(countries) + r')\b'
    match = re.search(pattern, text, flags=re.IGNORECASE)

    if match:
        return match.group(0)
    else:
        return None

def country_to_iso3(country_name):
    """
    Convert a country name to its ISO alpha 3 code.
        Input - country_name - The full country name
        Returns - str: Three-letter ISO code if found, None if not found
    """
    try:
        country = countries.get(name=country_name)
        if country:
            return country.alpha_3
        country = countries.search_fuzzy(country_name)
        if country:
            return country[0].alpha_3
        country_names = [country.name for country in countries]
        matches = get_close_matches(country_name, country_names, n=1, cutoff=0.6)

        if matches:
            country = countries.get(name=matches[0])
            return country.alpha_3

        return None

    except (AttributeError, LookupError):
        return None
```

```python
#Get the city and country for each user
def get_user_country(location_text):
    if not isinstance(location_text, str):
        return None
    location_text = resolve_country_abbreviation(location_text)
    return extract_countries(location_text)


users_df['Country'] = users_df['Location'].apply(get_user_country)

top_countries = users_df['Country'].value_counts().nlargest(20)
top_countries.plot.bar(figsize=(10, 5))
plt.title('Top 20 Countries by User')
plt.xlabel('Country')
plt.ylabel('Number of Users')
plt.show()
```




```python
# Count users per country
users_df['Country'] = users_df['Country'].str.lower()
```

```
user_counts = users_df['Country'].value_counts().reset_index()


user_counts.columns = ['Country', 'Users']
user_counts['Iso'] = user_counts['Country'].apply(country_to_iso3)
user_counts.reset_index()
user_counts.dropna(inplace=True)


fig = px.choropleth(user_counts, locations="Iso",
                    color="Users", # lifeExp is a column of gapminder
                    hover_name="Country", # column to add to hover information
                    color_continuous_scale=px.colors.sequential.Plasma[::-1])

fig.update_layout(title_text='World Map of User Count by Country')


fig.show()
```

```
Unable to display output for mime type(s): text/html
```

```
Unable to display output for mime type(s): text/html
```

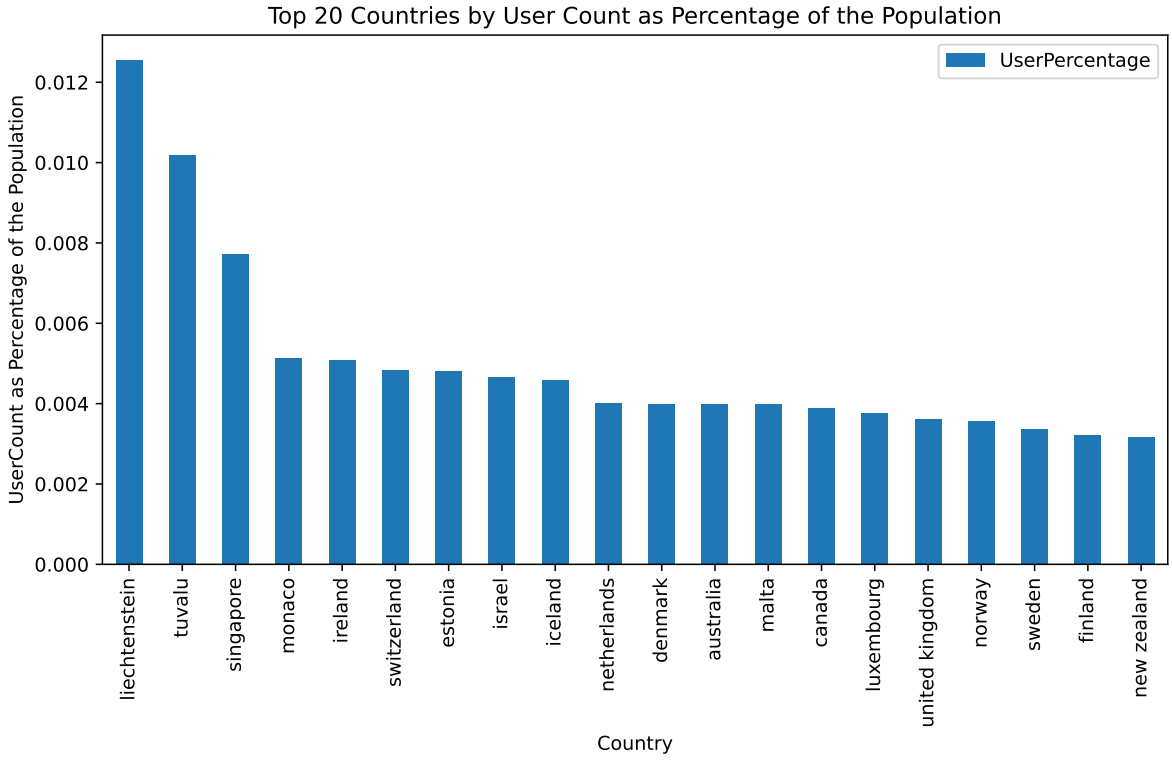
The Top 20 Countries by User shows that most users on the platform originate from India and the United States. These absolute numbers may not portray the full picture of how popular the DSSE platform is within a nation. It is more likely just a reflection of the nations that have the greatest populations.

World User HeatMap by Population Proportion

A more useful measure of the popularity of the DSSE platform within a nation to take the user count as percentage of population. This will allow us to identify which nations the platform is most popular and to compare those nations for any similarities.

```
world_bank_data = {
    "Country": [
        "ABW","AFE","AFG","AFW","AGO","ALB","AND","ARB","ARE","ARG","ARM","ASM","ATG","AUS",
    ],
    "Population": [
        107359,750503764,41454761,509398589,36749906,2745972,80856,481667539,10483751,455384
```

```
    ]
}

population_data = pd.DataFrame(world_bank_data)

merged_data = pd.merge(user_counts, population_data, left_on='Iso', right_on='Country')

merged_data['UserPercentage'] = merged_data['Users'] / merged_data['Population'] * 100

# Top 20 Countries by User Count as Percentage of Population
top_countries = merged_data.nlargest(20, 'UserPercentage')[['Country_x', 'UserPercentage']]
top_countries.plot.bar(x='Country_x', y='UserPercentage', figsize=(10, 5))
plt.title('Top 20 Countries by User Count as Percentage of the Population')
plt.xlabel('Country')
plt.ylabel('UserCount as Percentage of the Population')
plt.show()
```




```
# Draw heatmap
fig = px.choropleth(merged_data, locations="Iso",
```

```
                      color="UserPercentage",
                      hover_name="Country_x",
                      color_continuous_scale=px.colors.sequential.Plasma[::-1])

fig.update_layout(title_text='World Map of User Count as Percentage of the Population')

fig.show()
```

```
Unable to display output for mime type(s): text/html
```

The heatmap and bar chart of user count as a percentage of the population shows an interesting trend about engagement of smaller nations with the DSSE platform. Countries with smaller populations, such as Leichtenstein, Tuvalu and Monoco, show a disproportionately high user engagement relative to their size. This indicates a significant penetration of data science discuessions and platforms into these regions which may stem from higher internet connectivity or a focus on technology in education. Countries like Singapore standout in this regard as they are know to show a stong emphasis on data science and technolgoy within their education system. With only one exeception (Tuvalu), all counties that have a high user engagement relative to the total population are developed nations.

Although it has the highest numbers of users on the platform, India's user count a a percentage of the total population is ranked towards the lower end of the spectrum when compared to other nations. This shows that the importance of using the correct measure when determining metrics like user engagement and popularity.

Privacy and Ethical Considerations

This investigation has analyse data that involves data about people and as such, is subject to privacy and ethical concerns. A privacy concern is the potential for exposure of personal information. This analysis contains user locations, which could potentially be used to identify individuals, especially in smaller countries or regions (Tuvala is a good example of this). User creation dates and activity patterns could be potentially used to de-anonymise users when combined with other data. An ethical concern about the analysis is the bias in overrepresentation of developed nations. Language barriers may exclude non-English speaksers as DSSE is primarily English based.

Conclusion

This analysis of the DSSE platform highlights several significant insights into the global data science community. The trends observed in the lanauge usage and library preferences show the domincance of Python and R as programming languages of choice in this field.

A decline in the DSSE platform was observed from 2020 onwards. Whilst many external factors may be at play in this, further analysis and investigation would be prudent to determine what exactly those factors are and why they occured.

The global reach of the DSSE platform was also investigated. Whilst user from India and the United States dominate the platform in absolute terms, as a proportion to their populations, they are not the most represented. Nations that are highly developed are most present and represented on the platform.

The evolution and consistency of data science topics was also investigated. Fundamental concepts like classification and regression have consistently been discussed throughout time, whereas new topics like deep learning and transforms become very important and highly discussed issues in a short space of time. This represents the dynamic and evolving nature of the data science field.

Reference

1. Stack Exchange. (n.d.). Data Science Stack Exchange. Retrieved January 26, 2025, from https://datascience.stackexchange.com/
2. Internet Archive. (n.d.). Stack Exchange: Data Science [Data set]. Retrieved January 26, 2025, from https://archive.org/download/stackexchange/datascience.stackexchange.com.7z
3. del Rio-Chanona, M., Laurentsyeva, N., & Wachs, J. (2023). Are large language models a threat to digital public goods? Evidence from activity on Stack Overflow. arXiv. https://arxiv.org/abs/2307.07367