<u>Data 228: Big Data Technologies And Applications</u>
<u>Group - 8</u>
<u>Final Group Project - Technical Report</u>

Submitted By:
Aafrin Shehnaz(016801837),
Shivram Sriramulu(017439916),
Shreenithi Sivakumar(017473534),
Yogavarshni Ramachandran(017404036).

Classification of Subreddit Posts

1. Abstract:

In the expansive landscape of social media, Reddit stands out as a prolific platform where users engage in diverse discussions across thousands of communities known as subreddits. Each subreddit is dedicated to a particular theme or interest, ranging from general topics to niche hobbies. As the platform continues to grow, efficiently categorizing the vast array of posts becomes crucial for enhancing user experience, facilitating content discovery, and ensuring targeted engagement. This project leverages the capabilities of Apache Spark and its machine learning library, MLlib, to address the challenge of classifying Reddit posts into their corresponding subreddits. Utilizing a dataset sourced from Pushshift.io, which includes a wide range of post attributes such as text content, metadata, and interaction metrics, we conducted thorough exploratory data analysis to understand underlying patterns and inform our feature engineering processes. Our approach involved preprocessing the data to improve quality, transforming text data using advanced natural language processing techniques, and employing several machine learning models to predict the subreddit category for each post. The models were evaluated based on their accuracy and performance metrics.

Keywords: *Apache Spark, MLlib, Reddit, Machine Learning, Natural Language Processing, Big Data, Subreddit Classification, Social Media Analytics, Text Mining, Data Preprocessing, Feature Engineering.*

2. Introduction:

In the digital age, platforms like Reddit have become central hubs for sharing information, discussing topics, and building communities around interests both broad and niche. With millions of posts generated daily across thousands of subreddits, the task of managing and organizing this vast trove of content poses a significant challenge. Each subreddit, a dedicated virtual space for a specific topic, attracts discussions that range from general knowledge to specialized interests, like underwater basket weaving or high-level technology discussions. The sheer volume, variety, and velocity of data generated on Reddit epitomize the challenges associated with 'big data'. These characteristics not only complicate content management but also affect user engagement and content discoverability. As users navigate through this extensive network, the ability to efficiently categorize posts according to their relevant subreddits becomes crucial. Proper classification enhances user experience by facilitating easier content discovery and ensuring that discussions are targeted and relevant.

3. Problem Statement:

Our project addresses the critical challenge of accurately classifying Reddit posts into their corresponding subreddits. Effective categorization is essential for optimizing the user experience, facilitating content discovery, and fostering targeted engagement. Leveraging advanced machine learning techniques and big data analytics, our goal is to develop an automated model that can efficiently and accurately perform this classification. By enhancing the Reddit ecosystem, we aim to contribute to a more organized and accessible platform where users can easily find or stumble upon content that resonates with their interests or needs.

4. Motivation & Objective:

Utilizing the power of Apache Spark and its MLlib for machine learning, we will process and analyze Reddit post data to train and evaluate models capable of this classification. Our approach will cover extensive exploratory data analysis, feature engineering, model training, and evaluation, ensuring a robust system that understands the intricacies and nuances of Reddit's diverse content. By the end of this project, we aim to deliver a machine learning solution that not only categorizes Reddit posts with high accuracy but also enhances the overall moderation system, helping moderators redirect posts to the most appropriate subreddits and improving the discoverability of content across the platform.

5. Dataset overview:

Our dataset comprises a comprehensive collection of posts across multiple subreddits. We have considered the following 5 different subreddits for this project:

1. personalfinance
2. travel
3. Cooking
4. plants
5. programming

Each post in the dataset is represented by several attributes that capture the essence and context of the content. Key attributes include:

Column name	Column details
Author	The username of the post creator.

Title	The headline or title of the post.
Selftext	The main body of the post
Num_comments	The number of comments the post has received
Score	The net upvotes received by the post.
Subreddit	The subreddit to which the post belongs.
Created	The timestamp when the post was created.

Additionally, metadata such as the number of subscribers at the time of the post, the pageviews per month for the subreddit, and interaction metrics like upvote ratios are included to provide further insights into the engagement levels of each post.

6. Data Extraction

- **Source:** Raw data was sourced from the official monthly update of the PushShift.io Reddit data dump for February 2024.
- **Initial Format and Size:** The data initially totaled 22.84 GB in zst_blocks format.
- **Data Conversion:** Converted from zst_blocks to zst_files.
- **Data Focus:** Extracted Submissions data from selected subreddits (plants, cooking, personal finance, travel, programming), with a total data size of 14.5 MB.
- **File Conversion:** Used a batch processing script to convert zst files into CSV files, accommodating large file sizes.
- **Resulting Files:** Produced five CSV files, totaling 58.7 MB with 35,444 rows and 118 columns.
- **File Merging:** Merged these files into a single data file for further processing.

.

7. Data Preprocessing

Our preprocessing involved several steps to refine the dataset:

7.1 Column Selection and Reduction:

- Initial columns of interest included: author, created, domain, downs, link_flair_text, num_comments, over_18, permalink, score, selftext, subreddit_subscribers, title, ups, upvote_ratio, url, and subreddit.
- We eliminated the 'permalink' and 'url' columns due to redundancy.

7.2 Statistical Analysis:

- A correlation matrix was created to identify the relationships among numeric variables.

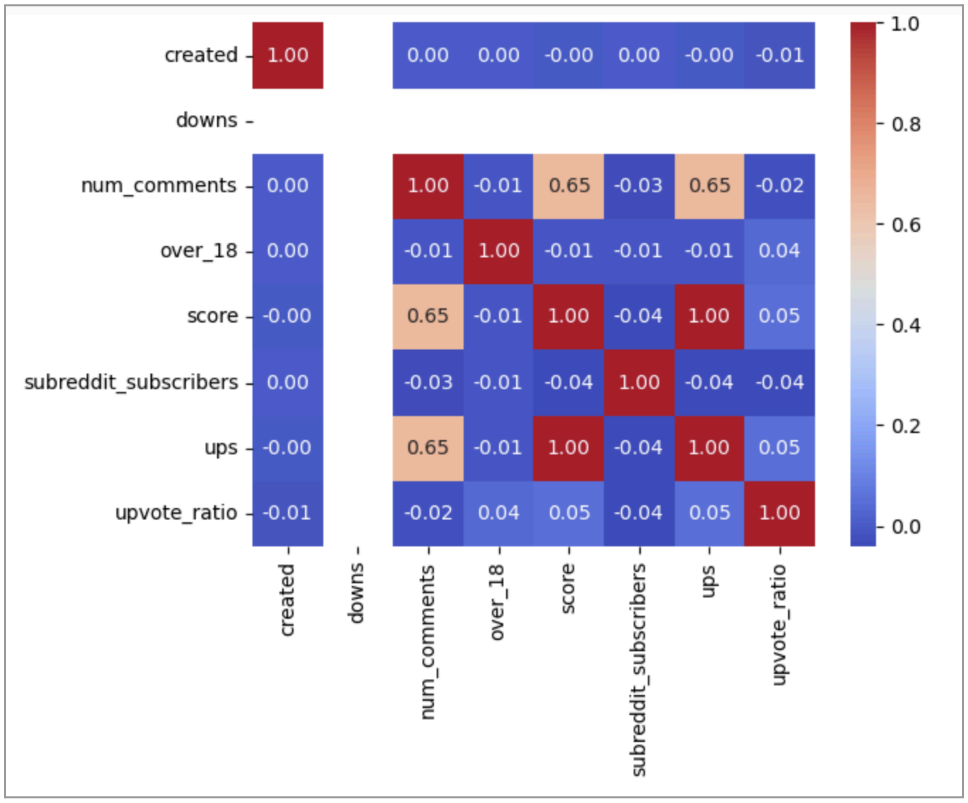


Fig. Correlation matrix for the numerical columns

- Based on this, the 'ups' and 'downs' columns were removed as 'ups' is highly correlated with 'score' and 'downs' provided no additional value.

7.3 Data Cleaning:

- Rows containing deleted or removed 'selftext' were excluded from the dataset.
- Nan values in the 'selftext' column were also dropped.
- We examined and ultimately dropped the 'domain' column, considering it redundant or irrelevant.

7.4 Subreddit Data Analysis:

- We analyzed the count of entries per subreddit and decided to drop the 'programming' subreddit due to its disproportionately small data size (147 rows).

7.5 Column Transformations:

- The 'over_18' column, which contained Boolean values, was converted to binary format (1 for True, 0 for False).

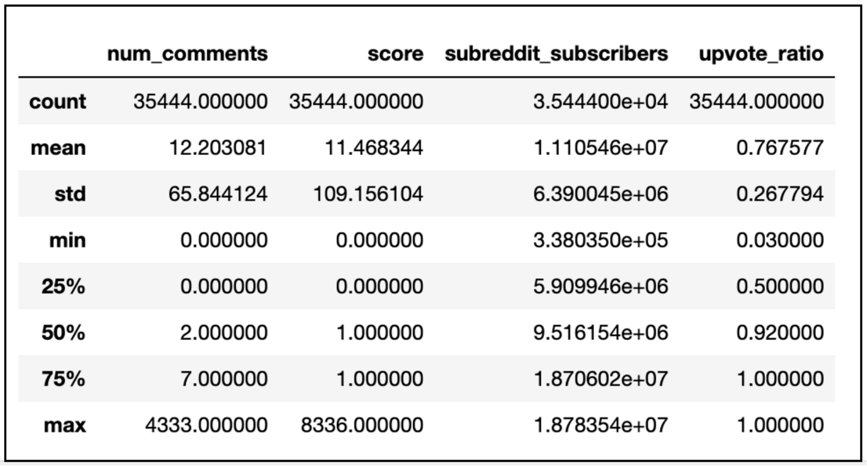
- The 'link_flair_text' column, primarily consisting of one or two words, contained some nan values. We decided to label these as 'Unknown' for the time being and planned to encode them later in the machine learning pipeline.

8. **Exploratory Data Analysis:**

To better comprehend the dataset, Exploratory Data Analysis (EDA) is vital for subreddit classification, as it reveals data patterns, detects outliers, and clarifies feature distributions.

8.1 Summary Statistics

By analyzing summary statistics and visualizing the data, it is useful to make informed decisions on preprocessing steps like normalization and feature selection.



	num_comments	score	subreddit_subscribers	upvote_ratio
count	35444.000000	35444.000000	3.544400e+04	35444.000000
mean	12.203081	11.468344	1.110546e+07	0.767577
std	65.844124	109.156104	6.390045e+06	0.267794
min	0.000000	0.000000	3.380350e+05	0.030000
25%	0.000000	0.000000	5.909946e+06	0.500000
50%	2.000000	1.000000	9.516154e+06	0.920000
75%	7.000000	1.000000	1.870602e+07	1.000000
max	4333.000000	8336.000000	1.878354e+07	1.000000

Fig. Summary statistics for the numerical columns

Significantly Different Ranges: Features have varying ranges, e.g., score averages 11.47 (max 8336), while subreddit_subscribers range from 338,035 to over 18 million. This necessitates normalization for consistent scaling.

Feature Statistics:

- num_comments: Avg. 12.20, Std. 65.84, Median 2.
- score: Avg. 11.47, Std. 109.16, highly skewed.
- subreddit_subscribers: Avg. 11.11M, Std. 6.39M, wide variation.
- upvote_ratio: Avg. 0.77, generally more upvotes than downvotes.

Given the varying feature ranges and distributions, normalization is essential to ensure the features are on a comparable scale for subsequent modeling.

8.2 Numerical Analysis

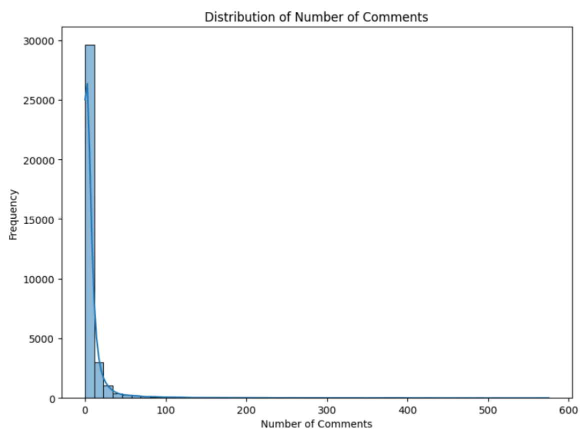


Fig. Distribution of No. of Comments

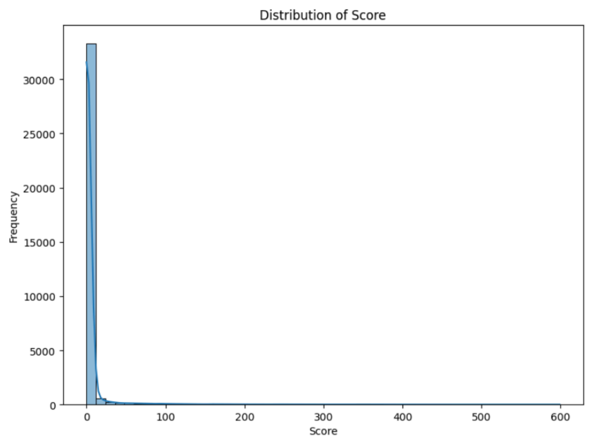


Fig. Distribution of Score

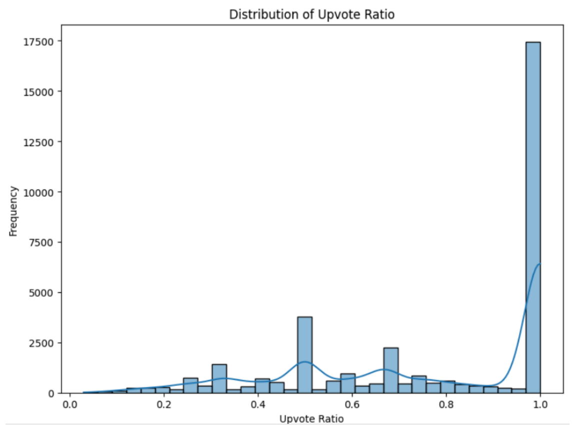


Fig. Distribution of Upvote Ratio

- The dataset has a few important numerical columns regarding the Reddit post. For num_comments, a post gets an average of 12.20 comments with a standard deviation of 65.84. The distribution is right-skewed since most posts do not get any comments; in fact, the median is 2.
- A similar reflection is made on the score column, which has an average value of 11.47 and a standard deviation of 109.16, while most posts contain a low score (median 1).
- The average upvote_ratio is 0.77. Most of the posts have a ratio very close to 1, which means they usually get more upvotes than downvotes.

These analyses further, agree on the point that various steps, such as normalization and feature engineering, are highly necessary to effect subreddit classification.

8.3 Correlation Analysis

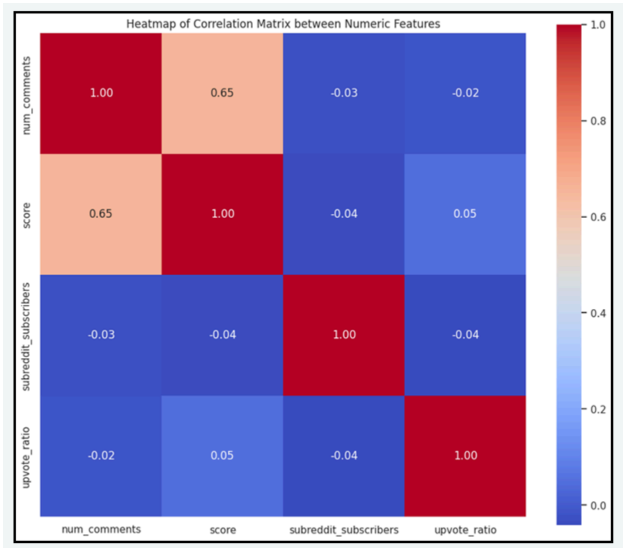


Fig. Correlation matrix for the selected numerical columns

- The positive correlation between num_comments and score, which is 0.65, suggests that posts with more comments are often the ones that receive higher scores.
- The number of subscribers doesn't necessarily predict how many comments a post will receive or its score, as indicated by the weak correlation between subreddit_subscribers and both num_comments (-0.03) and score (-0.04).
- The upvote_ratio does not seem to have a strong linear relationship with either the number of comments or the score of the posts, with correlations of -0.02 and 0.05, respectively.

8.4 Date and Time Analysis

To gain comprehensive insights into the dynamics of posts over time, it's essential to explore all available columns before finalizing the features for modeling. The graphs below were derived from the 'created' column to understand how post behavior changes with time.

Daily Number of Posts: There is no clear upward or downward trend, suggesting a relatively stable overall posting frequency.

- The top graph illustrates the number of posts made each day, fluctuating between 1100 to 1400 posts.

Daily Average Score: The majority of days cluster around a certain average score range, with a few outliers.

- The middle graph presents the daily average score per post, which remains relatively stable around 15-20 but shows notable spikes around 2024-02-17.

Daily Average Number of Comments: There's a fluctuating pattern in the average number of comments, indicating variability in user engagement over time.

- The bottom graph shows the average number of comments per post daily. Fluctuations are pronounced.

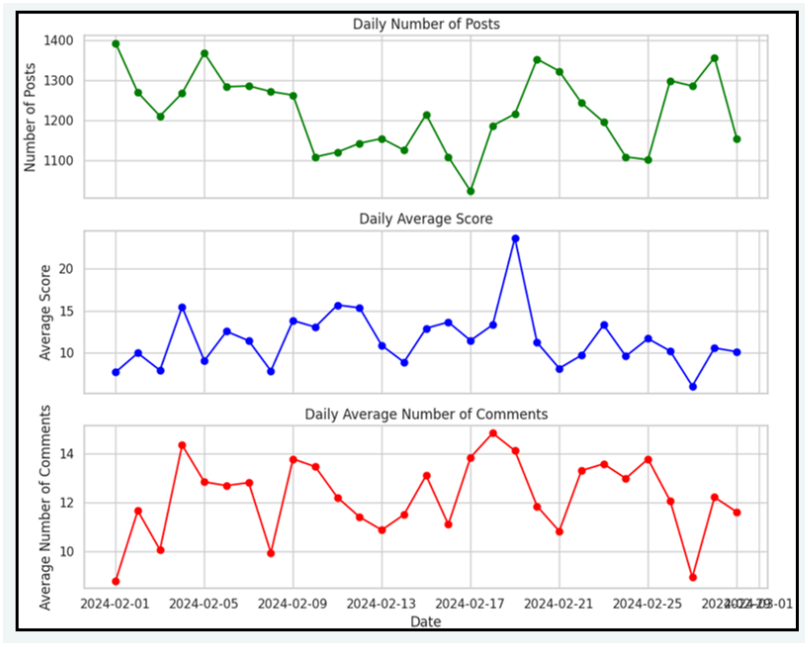


Fig Date and Time Analysis on the 'Created' column

8.5 Categorical Analysis

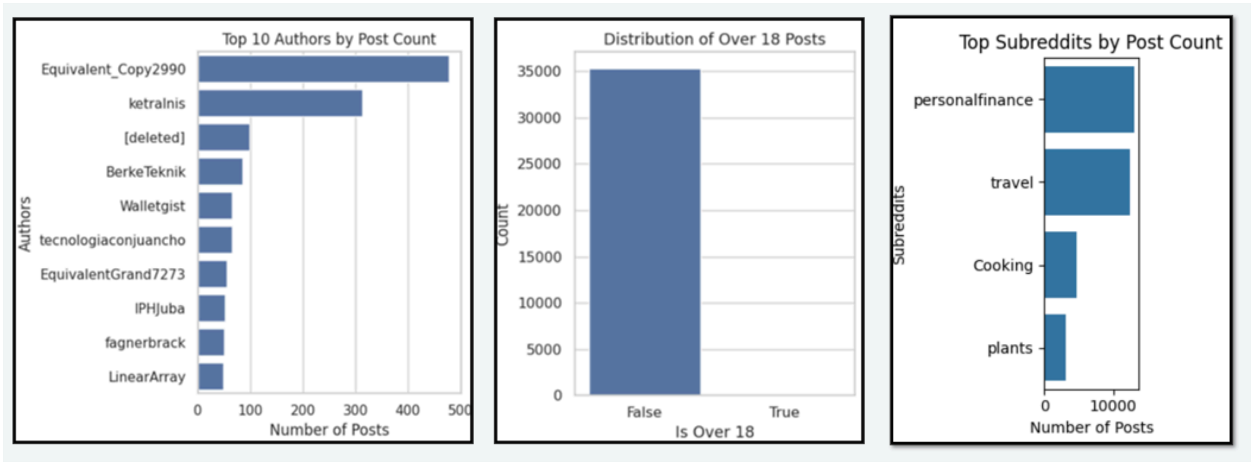


Fig. Count of 'Authors', 'Over 18', and 'Subreddit'

In exploring the dataset, we analyzed the categorical columns to gain deeper insights into user behavior and posting patterns.

- The first visualization, depicting the top authors by post count, reveals that Equivalent_Copy2990 is the most prolific author with close to 500 posts, followed by Ketralnis with approximately 250 posts. Understanding the contribution patterns of these top authors provides valuable insights into engagement strategies and the identification of potential spam or bot-like activity.
- The second visualization illustrates the distribution of over-18 posts, showing that the vast majority of posts over 35,000 are marked as "False," indicating they are not over-18. This minority presence of over-18 content highlights its rarity in the dataset.
- Finally, the top subreddits by post count indicate that ‘personalfinance’ dominates with 12983 posts, next in the row is ‘travel’ with 12326 posts, ‘cooking’ with 4658 posts, and ‘plants’ with 3058 posts represent a diverse range of interests.

8.6 Text Data Analysis

The two primary text columns in the Reddit dataset are ‘selftext’ and ‘title’, which together contain most of the content crucial for classification tasks. Analyzing these columns provides insights into their distribution and potential impact on the overall analysis.

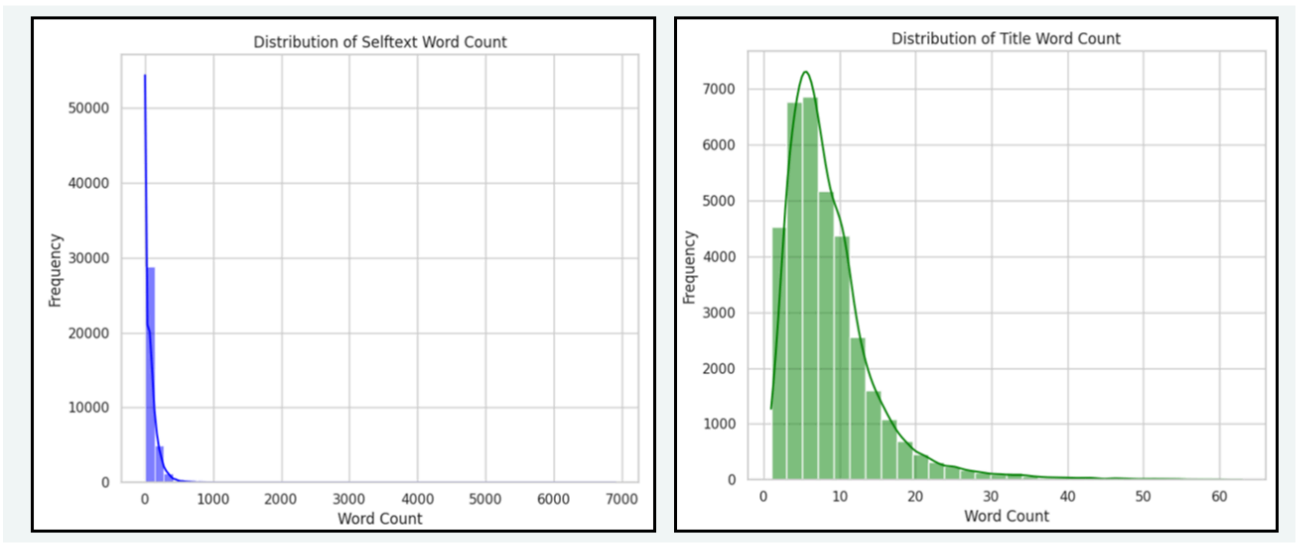


Fig. Distribution of ‘Selftext’ and ‘ Title’ column

- **‘Selftext’ column**: It contains the main body of the post content, and shows a highly skewed distribution where most posts have fewer than 100 words. Despite the skewed distribution, the tail extends up to 7000 words, indicating outliers with exceptionally long posts indicating the rich source of information and it will be critical for classification.

- **'title' column**: The word count distribution is also left-skewed, with most titles containing fewer than 10 words and a smaller tail extending up to 60 words. Despite their brevity, titles provide crucial context, often summarizing the most succinctly.

Both columns offer valuable insights for classification, suggesting feature engineering opportunities like extracting word counts and analyzing keywords or sentiment.

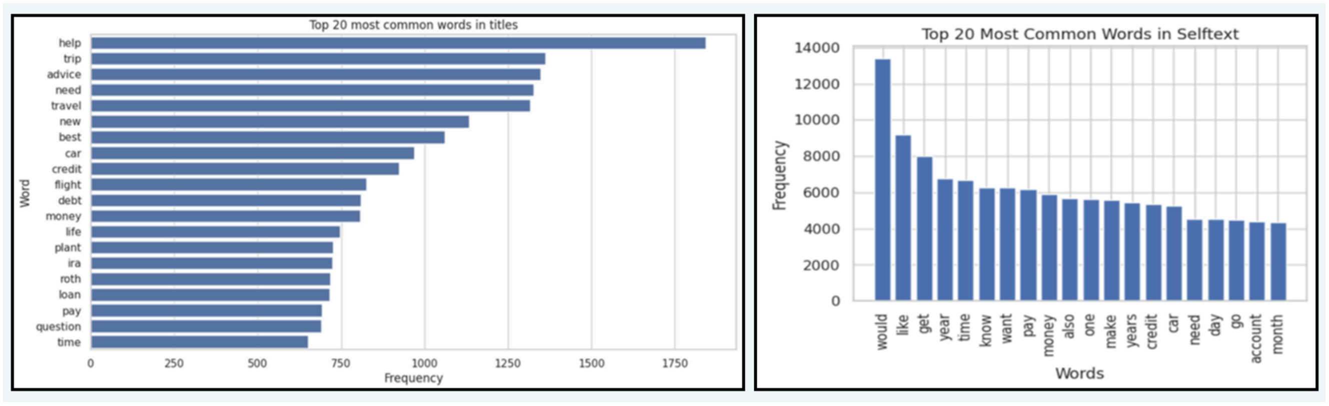
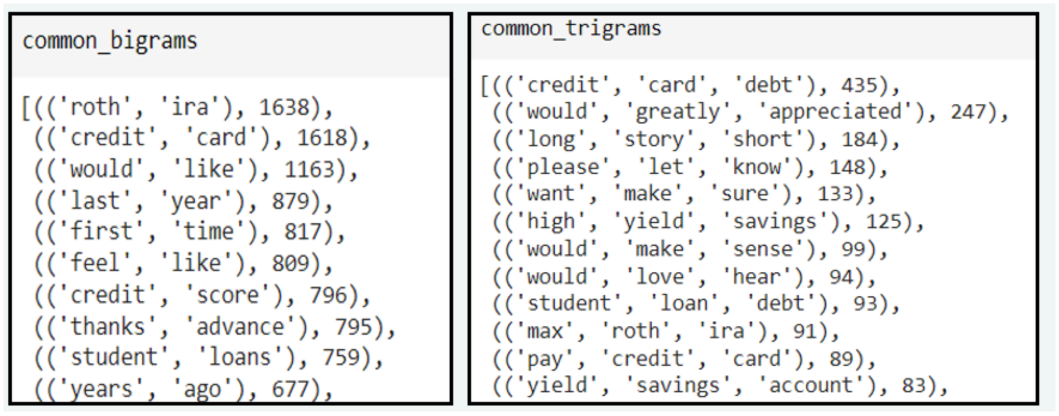


Fig. Top 20 most common words in the 'Selftext' and 'Title' column

Top Words in Titles:
The most common words include "help," "trip," "advice," and "travel," indicating the main topics of interest. Other frequent terms like "new," "credit," "money," and "life" offer additional context. For Example: "travel" Indicates discussions related to travel, itineraries, and trip planning, often found in subreddits like travel

Top Words in Self-Text:
The most frequent words, "would," "like," "get," "year," and "know," reflect similar themes to those in the titles. Words like "money," "pay," and "credit" highlight financial concerns, while "travel" and "plant" indicate popular leisure topics.

The below chart provides a basic exploration of common n-grams in the "selftext" column. By using bigram and trigram analyses, you can discover key phrases, relationships, and patterns to enhance the understanding of the dataset. A bigram is a combination of two consecutive words. A trigram is a combination of three consecutive words. It will display like "first word combination": Occurs [N] times.



```
common_bigrams

[(('roth', 'ira'), 1638),
 (('credit', 'card'), 1618),
 (('would', 'like'), 1163),
 (('last', 'year'), 879),
 (('first', 'time'), 817),
 (('feel', 'like'), 809),
 (('credit', 'score'), 796),
 (('thanks', 'advance'), 795),
 (('student', 'loans'), 759),
 (('years', 'ago'), 677),
```

```
common_trigrams

[(('credit', 'card', 'debt'), 435),
 (('would', 'greatly', 'appreciated'), 247),
 (('long', 'story', 'short'), 184),
 (('please', 'let', 'know'), 148),
 (('want', 'make', 'sure'), 133),
 (('high', 'yield', 'savings'), 125),
 (('would', 'make', 'sense'), 99),
 (('would', 'love', 'hear'), 94),
 (('student', 'loan', 'debt'), 93),
 (('max', 'roth', 'ira'), 91),
 (('pay', 'credit', 'card'), 89),
 (('yield', 'savings', 'account'), 83),
```

Fig. Few 'ngrams' from 'Selftext' column

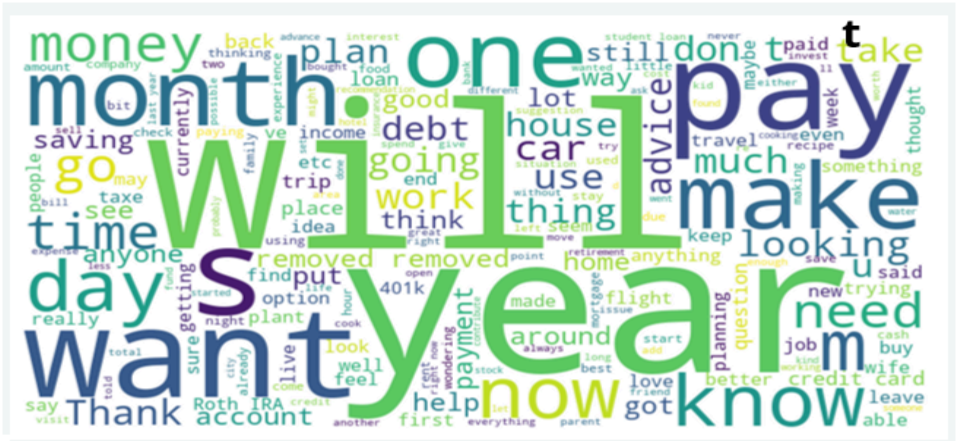


Fig. Word Cloud for Self-Texts

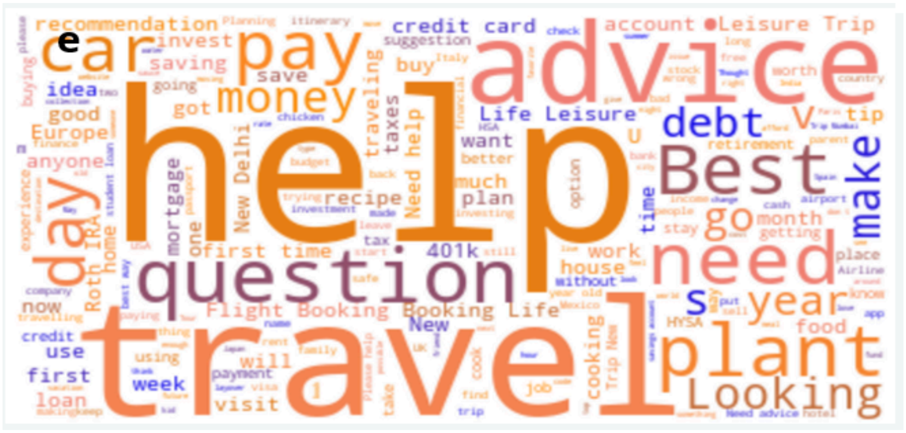


Fig. Word Cloud for Titles

This report aims to analyze the most frequently used words and phrases in a collection of Reddit posts, distinguishing between the titles and the self-texts of these posts. The word clouds reveal that the Reddit community is predominantly interested in seeking advice on topics related to travel, personal finance, and lifestyle. The differences between the title and self-text word clouds are highlighted:

- Titles often focus on attracting attention with requests for "help," "advice," or posing "questions."
- Self-texts delve deeper into the specifics of timeframes (year, month, day), financial concerns, and detailed advice or travel plans**.**

Both selftext and title data should be vectorized for inclusion in machine learning models. Comprehensive text processing and feature extraction will ensure accurate and reliable classification models that leverage these crucial text columns.

Key Insights:

- **created:** Time-based features may provide some signal if posting patterns are different across subreddits, though this is likely to be less informative than the text itself for our classification task.
- **author:** Authors may post frequently to specific subreddits, it will not be a potential feature for our classification.
- **num_comments, score, upvote_ratio**: These columns are the outcomes of the subreddit, post hoc characteristics rather than predictive features, However, in some instances, they might capture the level of engagement or popularity that is typical for certain subreddits.
- **selftext-** The main body of the post is the most informative feature for this task.
- **title -** It summarizes the post and is rich in keywords that can be indicative of the subreddit. Able to extract features from this text using the NLP technique

9. **Model Building and Hyperparameter Tuning (Using GridSearch Cross Validation)**

9.1 Setup and Preprocessing

Before constructing the models, we executed several preprocessing steps using the sklearn library:

- **Label Encoding:** Converted the target label from string format to numerical values.
- **Text Processing:** Utilized the nltk library for removing stop words and performing lemmatization on the selftext and title text columns.
- **Feature Transformation:** Transformed text columns into numerical vectors using TF-IDF.
- **Scaling of numeric columns:** Standardized all numeric columns with StandardScaler to have zero mean and unit variance.

- **Encoding of Categorical column:** Encoded the link_flair_text column using OrdinalEncoder.
- Finally, we dropped all non-essential columns for model building.

9.2 Data Splitting
We divided the dataset into training, validation, and test sets. Due to the highly imbalanced nature of the data, the stratify option was used to maintain consistent proportions across these subsets. These datasets were then saved for direct loading into PySpark.

9.3 Model Training and Evaluation with GridSearchCV
We constructed three different models—Random Forest, Logistic Regression, and Decision Tree and performed hyperparameter tuning using GridSearchCV with the following outcomes:

- **Best Parameters for Random Forest**

Parameter	Value
classifier__max_depth	None
classifier__min_samples_split	2
classifier__n_estimators	400

- **Best Parameters for Logistic Regression**

Parameter	Value
classifier__C	1
classifier__class_weight	balanced
classifier__max_iter	200
classifier__penalty	l2
classifier__solver	liblinear

- **Best Parameters for Decision Tree**

Parameter	Value
classifier__max_depth	None
classifier__min_samples_leaf	1

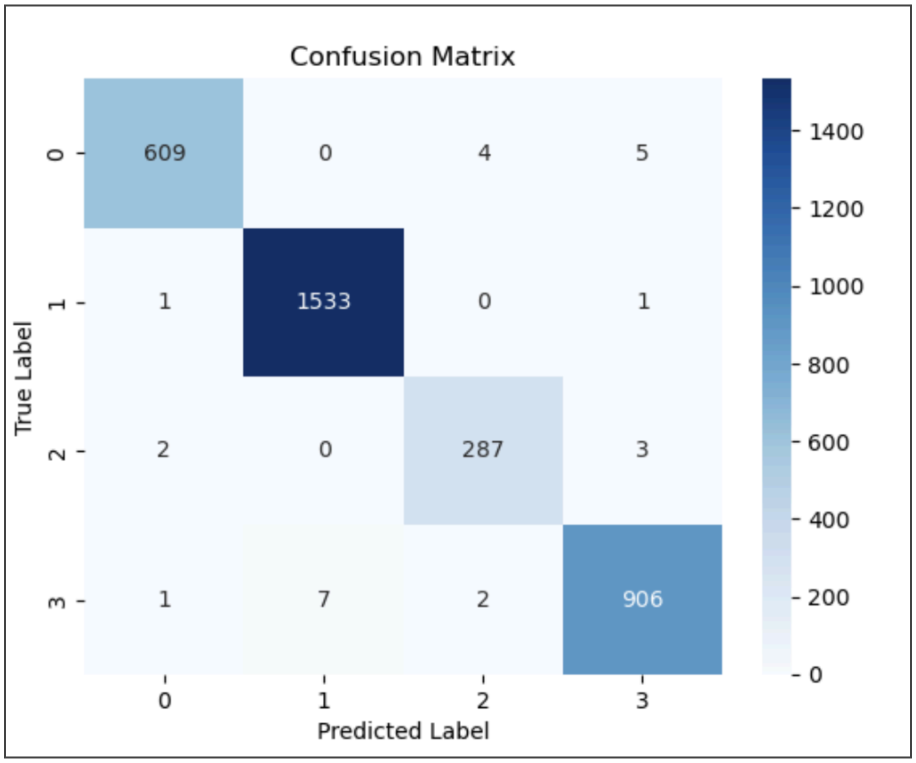
classifier__min_samples_split	2

- **Best Score**

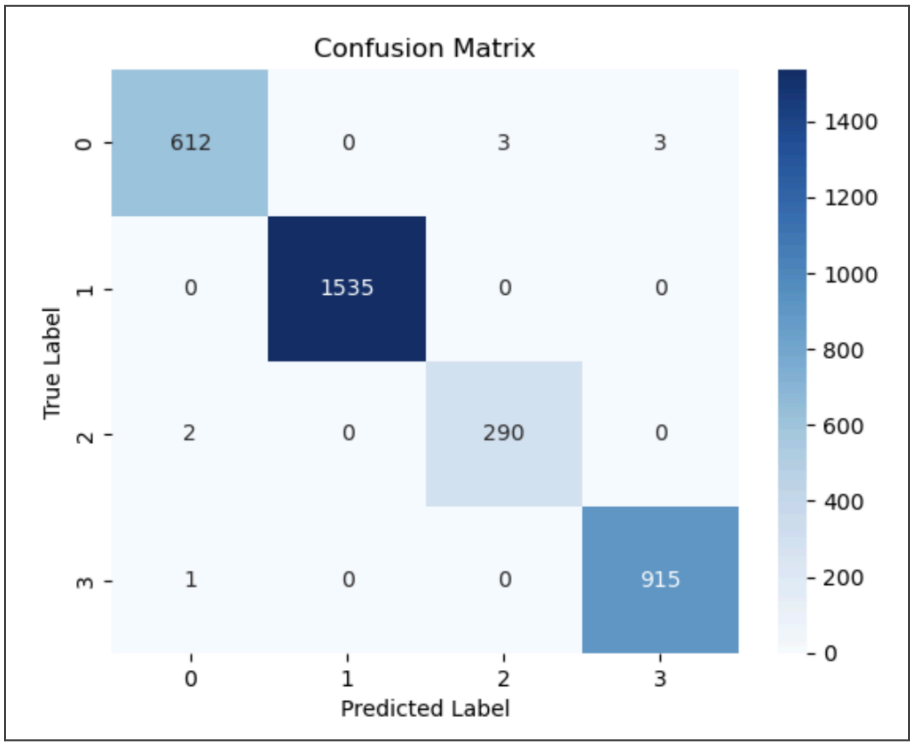
Model	Best Score
Random Forest	0.9947
Logistic Regression	0.9977
Decision Tree	1

Confusion Matrix For Validation Dataset:

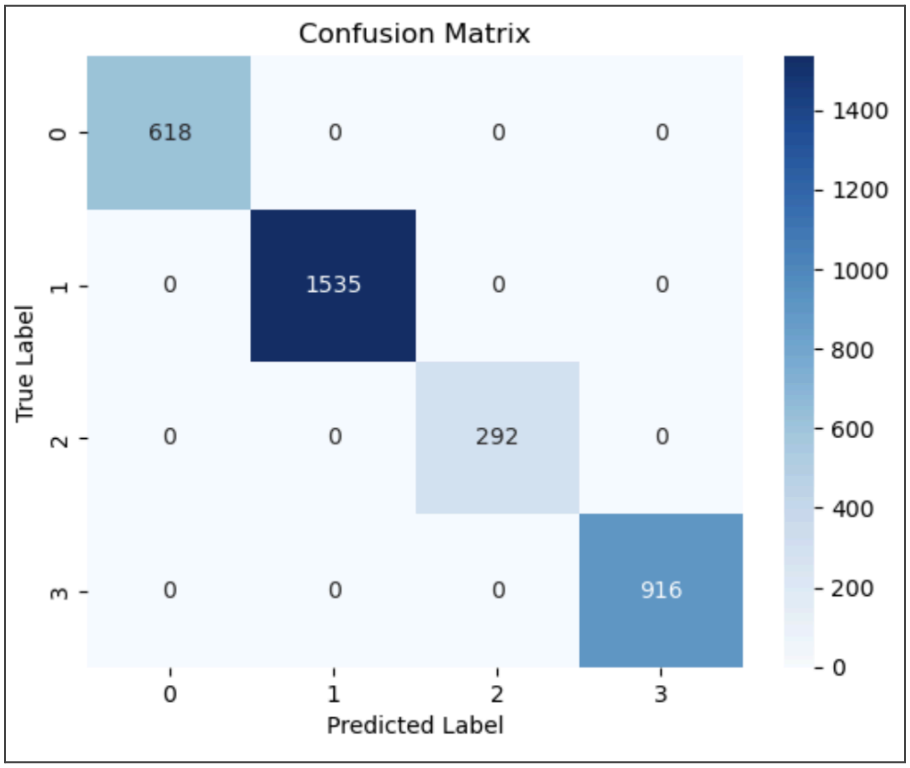
- **Random Forest**



- **Logistic Regression:**



- **Decision Tree**



10. PySpark Integration

10.1 Data Loading and Initial Transformation

We began by encoding the target variable outside the Spark Pipeline using StringIndexer, which handles invalid entries effectively by retaining them (handleInvalid="keep"). This step was performed separately for the training, validation, and test datasets to ensure the label consistency across different datasets.

10.2 Data Transformation:

- **Dynamic Text Processing Pipeline:**
  To efficiently handle text columns, we dynamically created pipeline stages for each text column using a custom function. This function incorporates four key steps:

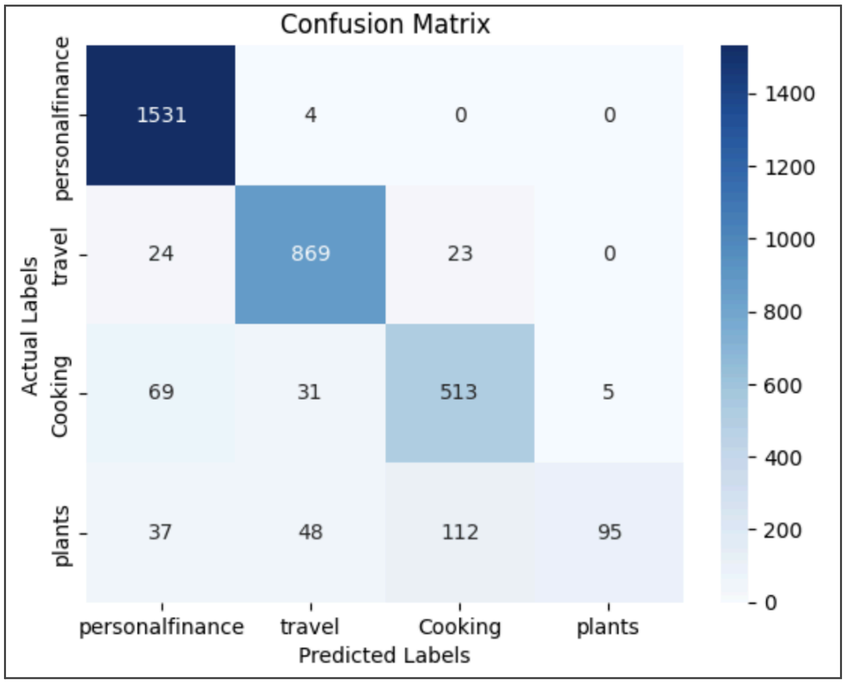
  - **RegexTokenizer** to split text into words.
  - **StopWordsRemover** to eliminate common stopwords.
  - **HashingTF** to convert the filtered text into feature vectors.
  - **IDF** to compute the Inverse Document Frequency (IDF) of each word.

- **Categorical Data Transformation:**
  For categorical columns, we employed **StringIndexer** to transform these into numeric indices, again using handleInvalid="keep" to manage any missing values.

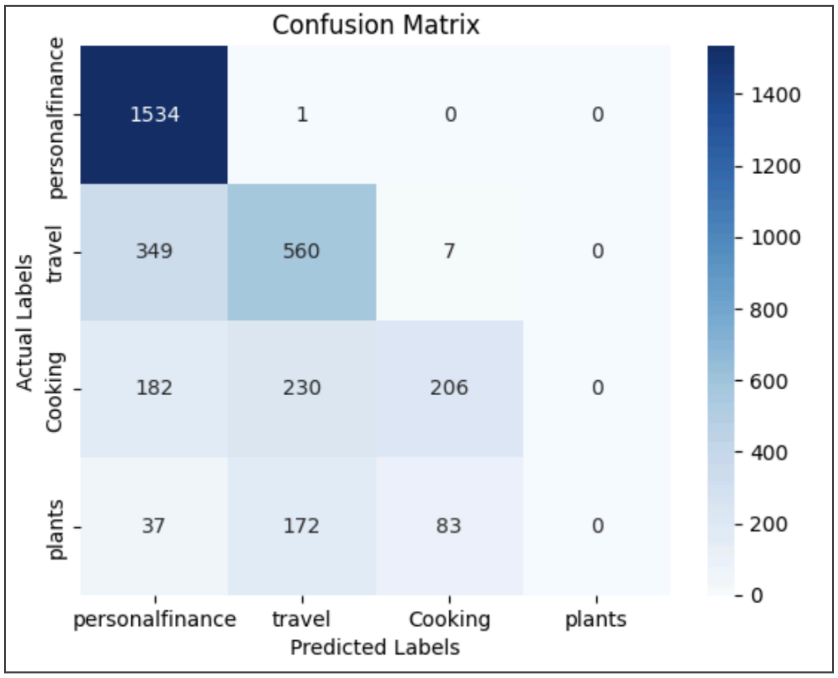
- **Numeric Data Transformation:**
  Numeric columns were assembled into a single vector using VectorAssembler and then scaled using StandardScaler to ensure data standardization.

Confusion Matrix For Validation Dataset:

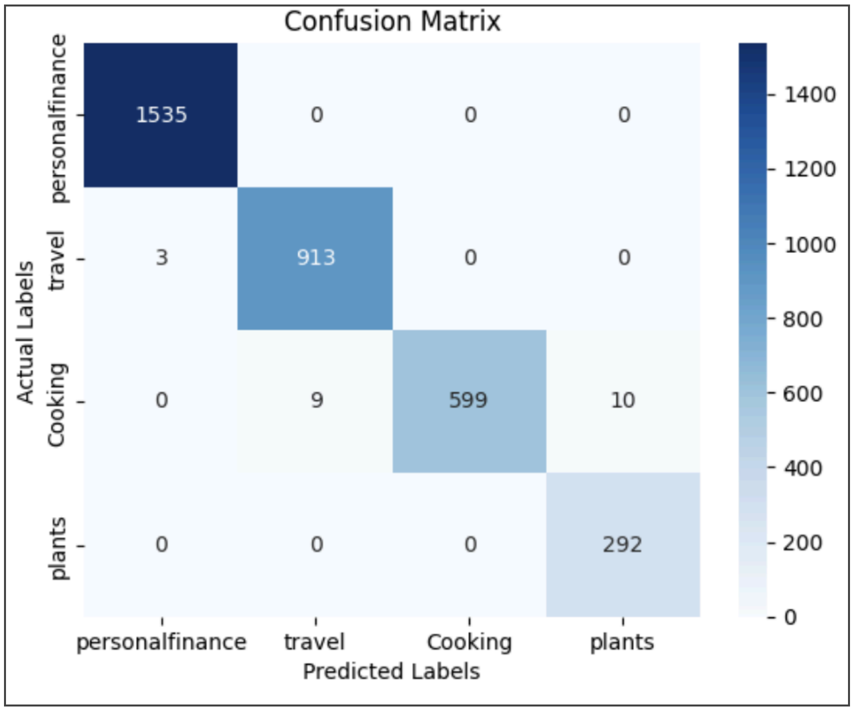
- **Random Forest**



- **Logistic Regression:**



- **Decision Tree:**



10.3 Feature Assembly:
All preprocessed features (TF-IDF vectors for text, indexed categorical values, and scaled numeric features) were combined into a single feature vector using VectorAssembler. This comprehensive feature vector is critical for training the machine learning model.

10.4 Model Building and Pipeline Configuration:
We set up a Spark Pipeline that includes:

- Text processing stages generated dynamically for each text column.
- Categorical data indexers.
- Numeric feature assembler and scaler.
- A Random Forest classifier with predefined parameters (numTrees=400, maxBins=40), leveraging the best practices identified from sklearn.

10.5 Pipeline Execution:
The pipeline was fitted to the indexed training dataset, and then used to transform both the validation and test datasets. This approach ensures that the same preprocessing and modeling steps are applied consistently across all data splits.

10.6 Evaluation:

Predictions made on the validation and test datasets were evaluated to measure the model's performance. Metrics such as accuracy, precision, recall, and F1-score were calculated to assess the effectiveness of the model under real-world conditions. The table below shows the evaluation results for each model:

Model	Validation Accuracy	Test Accuracy	F1 Score	Precision	Recall
Random Forest	0.89	0.89	0.8782	0.8938	0.8914
Logistic Regression	0.68	0.67	0.6306	0.6197	0.6843
Decision Tree	0.99	0.99	0.9932	0.9933	0.9932

10.7 Model Evaluation Differences: scikit-learn vs. PySpark.ml
The key distinctions affecting model evaluations between scikit-learn and PySpark.ml are:

- **Implementation Variances:** Differences in core algorithms and optimization techniques can lead to varying results.
- **Data Handling:** PySpark.ml distributes data across multiple nodes, while scikit-learn processes it on a single node, impacting efficiency and scalability.
- **Default Settings:** Variations in default parameters such as regularization and iterations influence model performance.
- **Preprocessing:** Each platform uses different data scaling and normalization techniques, affecting the input data's representation.
- **Random Processes:** Handling of random processes like data shuffling and initialization differs, influencing consistency and reproducibility.

11. Model Building with only textual features

- Tokenizer is applied to tokenize the "title" column into individual words.
- Each title is split into a list of words, creating a new column named "words" in the DataFrame.
- HashingTF is applied to convert the tokenized words into feature vectors.
- IDF (Inverse Document Frequency) is applied to scale the raw features obtained from HashingTF.
- Later we apply one hot encoding for our target feature subReddit.
- VectorAssembler is used to assemble the TF-IDF features and labels (one-hot encoded) into a single feature vector.

- This encoded assembly vector is randomly split,then loaded into the classification model to obtain accuracy,precision and F1 score.

11.1 Model Evaluation For Text Only Models:

Validation Metrics:

Models	**Precision**	**Recall**	**F1 Score**	**Accuracy**
Logistic Regression	0.7394	0.8475	0.7843	0.8475
Random Forest	0.9805	0.9798	0.9782	0.9798
Naive Bayes	0.9427	0.9402	0.9182	0.9402

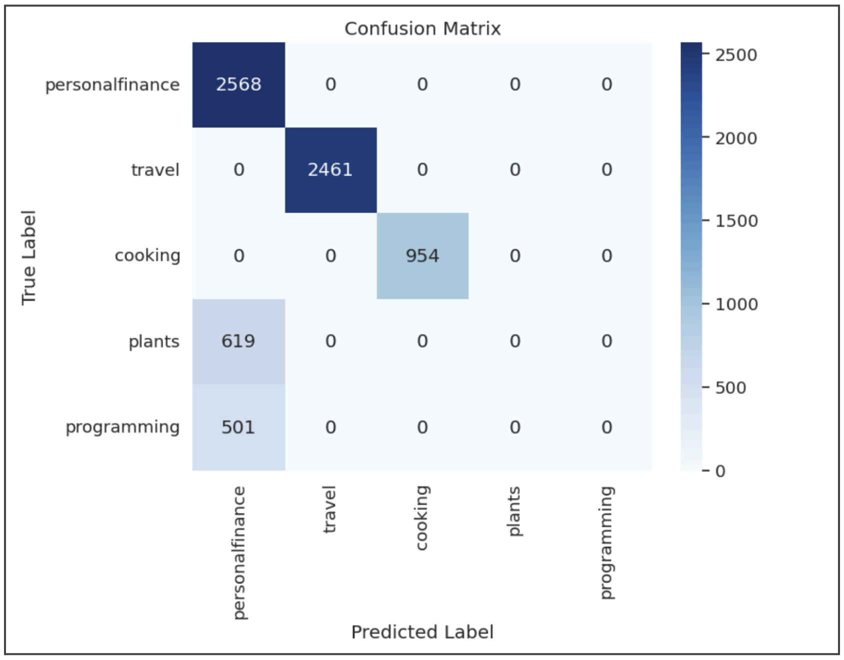
Test Metrics:

Models	**Precision**	**Recall**	**F1 Score**	**Accuracy**
Logistic Regression	0.7325	0.8423	0.7775	0.8423
Random Forest	0.9913	0.9912	0.9910	0.9812
Naive Bayes	0.9391	0.9392	0.9159	0.9392

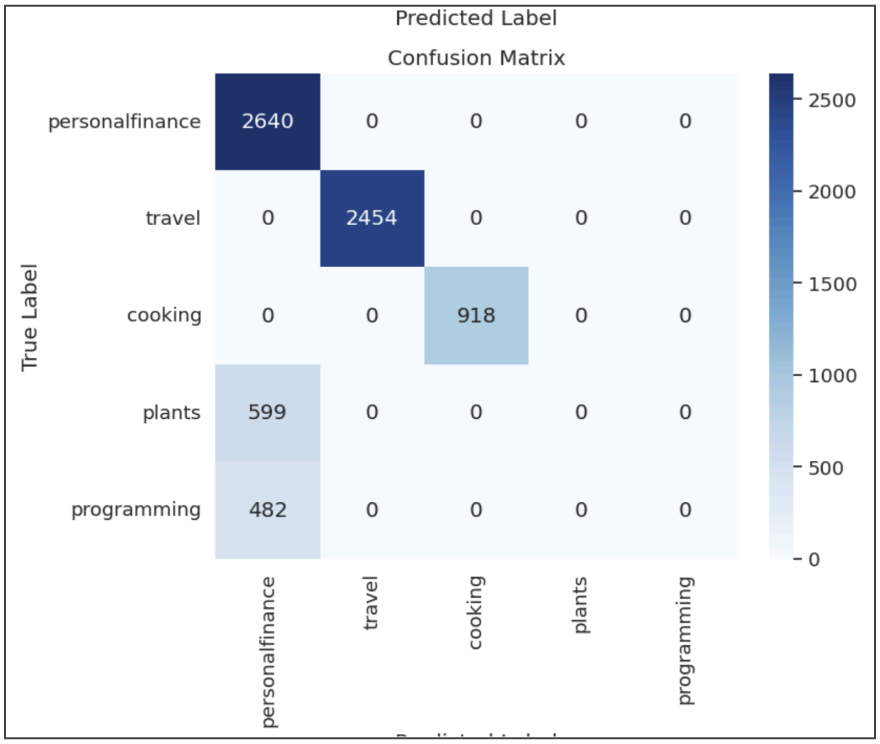
We can observe that Random forest with the highest accuracy 98% and logistic regression with the lowest accuracy of 84%.

Logistic Regression:

- **Confusion matrix for Validation data:**



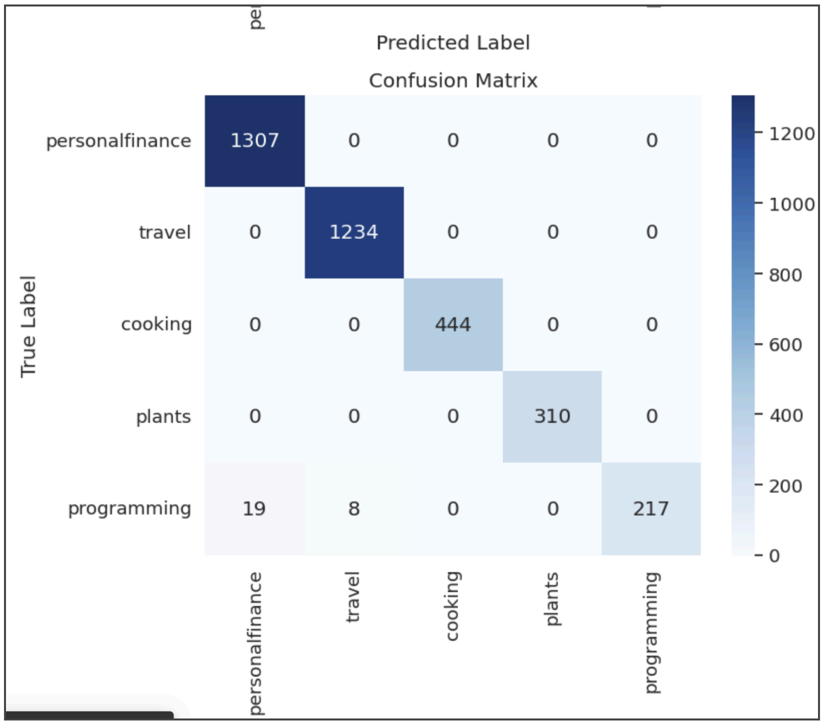
- **Confusion matrix for Testing data**



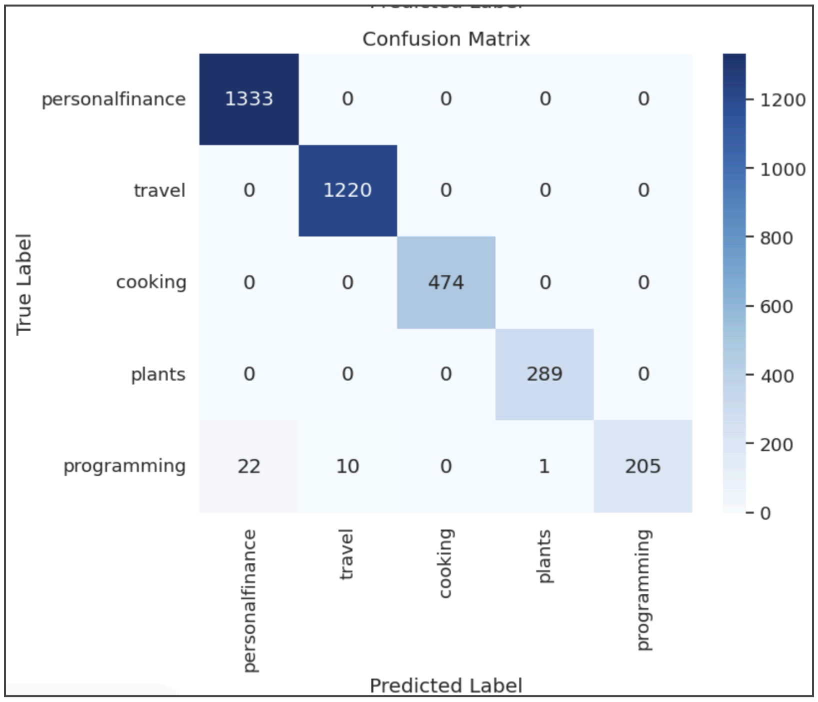
The testing and validation datasets show that the accuracy of the classification model varies depending on the class. The model achieves perfect precision and recall with excellent classification accuracy for the personal finance, travel, and cookery categories in the validation data. In particular, the model correctly identified 2,568 cases related to personal finance, 2,461 instances related to travel, and 954 instances related to cooking, yielding strong F1-scores for these categories.

Random Forest

- **Confusion matrix for Validation data:**

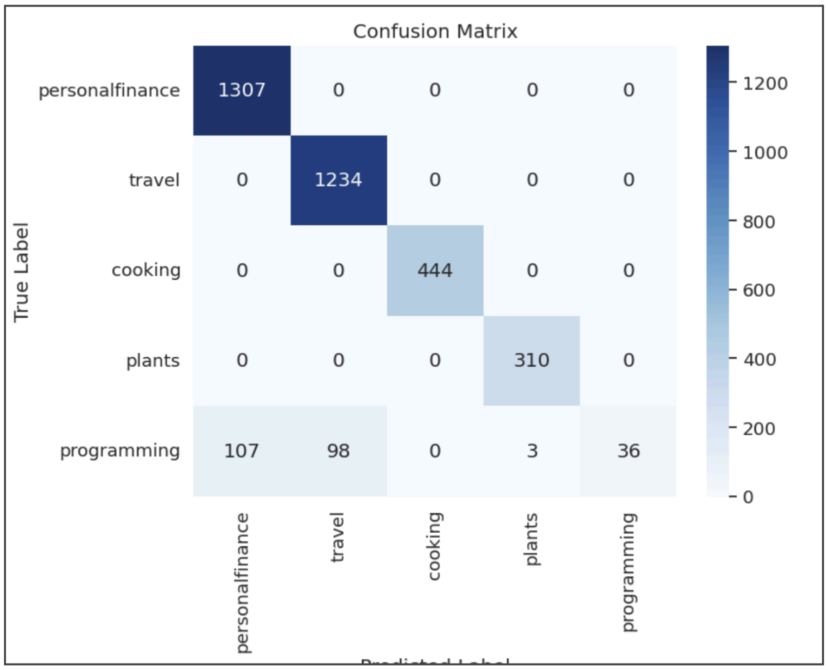


- **Confusion matrix for Test data**

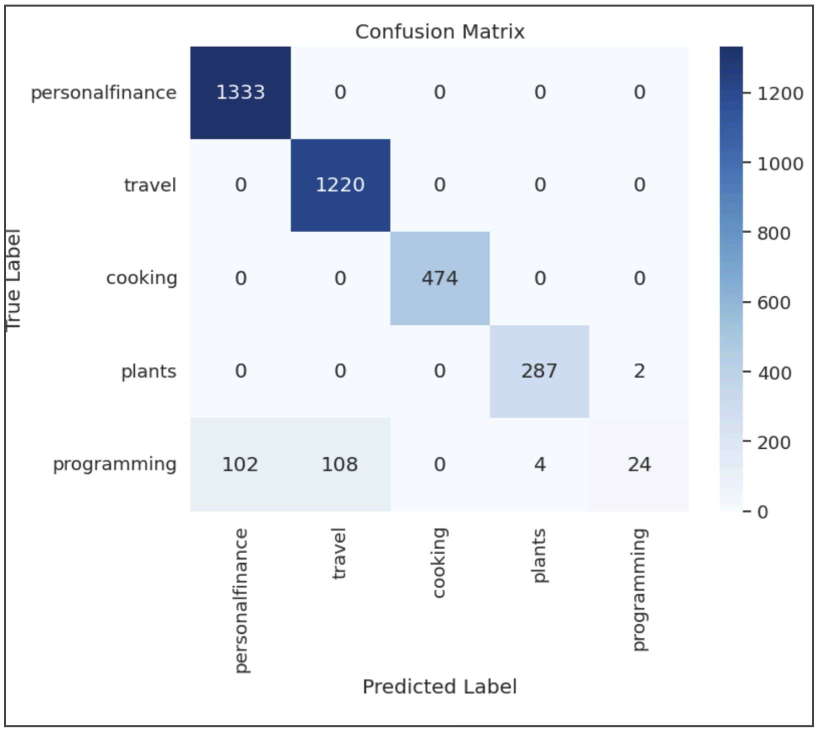


Naive Bayes

- **Confusion matrix for validation data**



- **Confusion matrix for test data:**



Comparative Analysis
The evaluation also included a comparative analysis between the models processed in scikit-learn and their counterparts in PySpark. This comparison was crucial to ensure that despite the differences in processing architectures and data handling between the two platforms, the models maintained their efficacy. The Random Forest model consistently showed superior performance on both platforms, highlighting its robustness and adaptability.

Conclusion:
This project successfully demonstrated the application of machine learning techniques and Apache Spark to classify Reddit posts into relevant subreddits effectively. By leveraging a comprehensive dataset and employing models such as Random Forest, Logistic Regression, and Decision Trees, we achieved high accuracy and robust performance in automating the categorization process. The implementation showcased the potential of big data technologies in enhancing content management and user experience on large digital platforms like Reddit. The challenges of data preprocessing and model optimization provided valuable insights, highlighting the importance of thorough data analysis and the careful integration of diverse technologies. Future enhancements could focus on real-time classification to further support dynamic content management and improve user engagement. Through this

project, we have established a foundation for more sophisticated content management strategies that could significantly benefit digital community interactions and platform navigability.