Exploring the Use of Machine Learning and Natural Language Processing Techniques for Fake News Detection on Reddit

Dhrumil Shah

Department of Applied Data Science, San Jose State University

DATA270: Data Analytics Process

Dr. Linesy Pang

December 04, 2023

Model Development

Model Proposals

Fake news dissemination is a growing concern in online platforms, with Reddit being a prominent source of information sharing so for our project, A supervised machine learning model was used to predict fake information in Reddit user posts through binary classification. Support Vector Machines (SVM) is an commonly applied supervised approach for classification and regression tasks. SVM is well-suited for binary classification problems that involve categorizing data points into one of two labels.

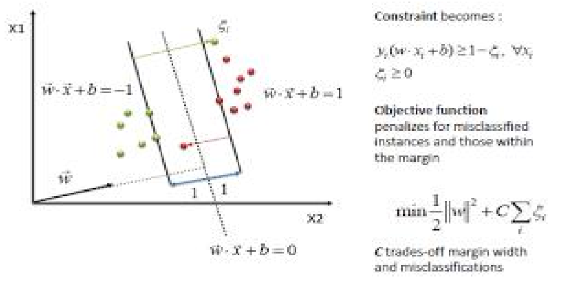
Also, The study conducted by Khanam et al. (2021) focuses on identifying fake news in the political sphere using machine learning approaches applied to Twitter data. The researchers implemented a systematic process that began with acquiring data in tab-separated format, which was then converted to comma-separated format. Subsequent steps involved preprocessing to remove noise from the dataset. This involved the use of natural language processing and NLTK libraries for tasks like stemming, lemmatization and tokenization.

Feature extraction represented a key aspect of the research. Lexical features such as word count and length were selected. The authors extracted unigram and bigram features using the Tiff Vectorizer function, leading to TF-IDF n-gram features being generated. The dataset was then divided into 70% for training and 30% for testing to facilitate evaluation using various classifiers including Linear Regression, Random Forests, XGBoost, Naive Bayes, K-nearest neighbors, Decision Trees, and Support Vector Machine.

SVM performs classification by selecting the optimal boundary or decision surface to separate labeled data points. This boundary, known as a hyperplane in high-dimensional space, aims to maximize the gap between the two classes of data points. As a result, all data bearing the same label will be located on one side of this hyperplane. A key concept is that the hyperplane represents an n-dimensional dividing line used to segregate data, where 'n' equals the number of features in the dataset. The hyperplane serves as the optimal decision boundary between the two classes (Gandhi,2018).

Figure1

SVM Workflow


X1
X2
ξ_i
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=1$
$\vec{w}\cdot\vec{x}+b=0$
$\vec{w}$
1
1
Constraint becomes :
$y_i(w\cdot x_i+b)\geq 1-\xi_i,\ \forall x_i$
$\xi_i\geq 0$
Objective function
penalizes for misclassified instances and those within the margin
$\min\frac{1}{2}\|w\|^2+C\sum_i\xi_i$
C trades-off margin width and misclassifications

For our project Let X represent the feature matrix containing linguistic attributes of Reddit posts, Let y denote the binary class labels (0 for fake news, 1 for genuine news), Let n be the number of training examples.The SVM model aims to find the hyperplane defined by

Decision Function

$$w \cdot x - b = 0 \quad (1)$$

that maximally separates the two classes, subject to the constraint

Objective Function

$$\text{Minimize} \quad \text{-->} \quad 1/2||w||^2 \quad (2)$$

The objective function for SVM aims to maximize the margin of separation between classes while minimizing classification errors. In (2) where y_i is the class label (either +1 or -1) for the i-th data point.

$$y_i(w * x_i - b) \geq 1 \text{ for i=1,2,...,n} \quad (3)$$

In other words form (3), the SVM seeks an optimal hyperplane w·x - b = 0 to perform maximum-margin classification of the training data points x_i with corresponding labels y_i for i=1 to n. The hyperplane aims to correctly classify future unseen examples based on the separation achieved during model training.

Lagrange Multipliers:

SVM uses Lagrange multipliers to solve the optimization problem subject to the constraints. The Lagrangian is formed, and the dual problem is derived.

Kernel Trick:

For non-linearly separable data, SVM employs the kernel trick. The decision function is transformed to include a kernel function K(xi,xj), allowing SVM to operate in a higher-dimensional space without explicitly computing the transformations as seen from figure(2).

Figure2

$$\begin{aligned} \text{Linear :}\ & K(w,b) = w^T x + b \\ \text{Polynomial :}\ & K(w,x) = (\gamma w^T x + b)^N \\ \text{Gaussian RBF:}\ & K(w,x) = \exp(-\gamma||x_i - x_j||^n \\ \text{Sigmoid :}& K(x_i,x_j) = \tanh(\alpha x_i^T x_j + b) \end{aligned}$$

Different Kernels.

Support Vectors:

Support vectors are the data points that lie closest to the hyperplane and influence its position.These are crucial in defining the margin and ultimately the decision boundary.

Final Formulation:

To handle non-linearly separable data, SVM uses a kernel trick. The decision function is transformed to include a kernel function K(x_i, x_j), allowing operations in a higher-dimensional space:

$$f(x) = \sum_{i=1}^{n} [\alpha_i \cdot y_i \cdot K(x, x_i) - b] \tag{4}$$

Here

a_i is are the Lagrange multipliers,

y_i is the class label,

x_i is the support vector,

In summary, as seen from (4) the objective is to define a maximum-margin hyperplane for classification while improving generalization capability through use of the kernel trick for non-linear datasets. The output is a decision function that can be used to predict class labels for new test points.

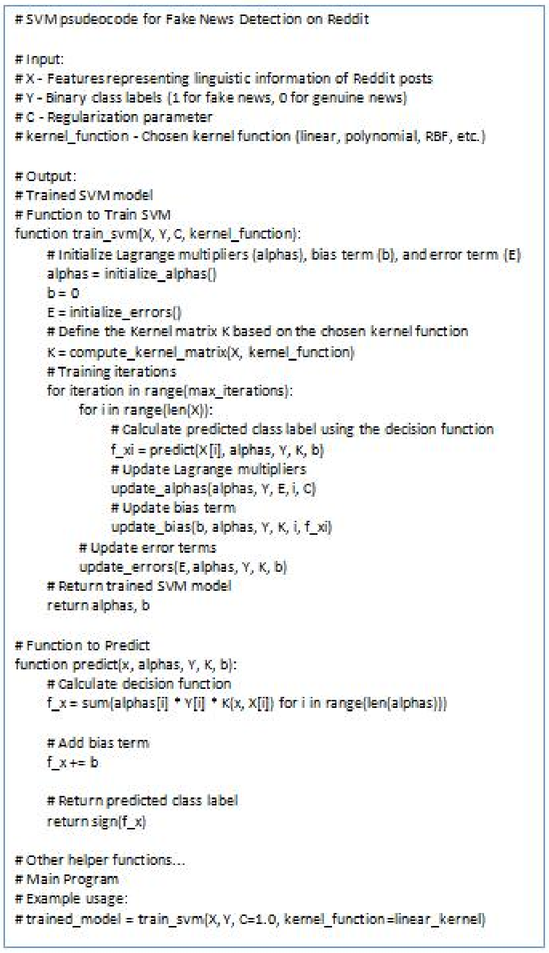
Similar to logistic regression, text-based posts undergo conversion into numerical features through methods like Count Vectorization or TF-IDF. SVM operates by applying a kernel function to transform these features into a higher-dimensional space, subsequently identifying the hyperplane that optimally separates data points of distinct classes, (Gandhi, 2018). The SMO algorithm is utilized for iterative training of the SVM model, adjusting the position of misclassified points towards the decision boundary and selecting the most effective hyperplane to maximize the margin.

SVM's ability to handle high-dimensional data, find clear decision boundaries, and generalize well makes it a popular choice for various classification tasks, including the detection of fake news on platforms like Reddit, as outlined in our project proposal.

Corresponding SVM psuedocode for our project:

Figure3

Psudeocode of SVM workflow



```
# SVM psudeocode for Fake News Detection on Reddit

# Input:
# X - Featuresrepresenting linguistic information of Reddit posts
# Y - Binary class labels (1 for fake news, 0 for genuine news)
# C - Regularization parameter
# kernel_function - Chosen kernel function (linear, polynomial, RBF, etc.)

# Output:
# Trained SVM model
# Function to Train SVM
function train_svm(X, Y, C, kernel_function):
    # Initialize Lagrange multipliers (alphas), bias term (b), and error term (E)
    alphas = initialize_alphas()
    b = 0
    E = initialize_errors()
    # Define the Kernel matrix K based on the chosen kernel function
    K = compute_kernel_matrix(X, kernel_function)
    # Training iterations
    for iteration in range(max_iterations):
        for i in range(len(X)):
            # Calculate predicted class label using the decision function
            f_xi = predict(X[i], alphas, Y, K, b)
            # Update Lagrange multipliers
            update_alphas(alphas, Y, E, i, C)
            # Update bias term
            update_bias(b, alphas, Y, K, i, f_xi)
        # Update error terms
        update_errors(E, alphas, Y, K, b)
    # Return trained SVM model
    return alphas, b

# Function to Predict
function predict(x, alphas, Y, K, b):
    # Calculate decision function
    f_x = sum(alphas[i] * Y[i] * K(x, X[i]) for i in range(len(alphas)))

    # Add bias term
    f_x += b

    # Return predicted class label
    return sign(f_x)

# Other helper functions...
# Main Program
# Example usage:
# trained_model = train_svm(X, Y, C=1.0, kernel_function=linear_kernel)
```

Input: X - Features representing linguistic information of Reddit posts. Y - Binary class labels (1 for fake news, 0 for genuine news). C - Regularization parameter. kernel_function - Chosen kernel function (linear, polynomial, RBF, etc.). Output: Trained SVM model with Lagrange multipliers (α) and bias term (b). Initialization: Initialize Lagrange multipliers (α), bias term (b), and error term (E). K is the Kernel matrix calculated based on the chosen kernel function as seen from figure(3).

Training Iterations:

Iterative process to update Lagrange multipliers and bias term. Outer loop iterates for a predefined number of iterations.Inner loop processes each data point in the dataset.For each data point, it calculates the predicted class label using the decision function.

Update Lagrange Multipliers (α): Update the Lagrange multipliers based on the error terms. αi is updated using the Sequential Minimal Optimization (SMO) algorithm.Update

Bias Term (b): Update the bias term based on Lagrange multipliers, error terms, and kernel matrix. Adjust b to improve the model's performance.Update Error Terms (E): Update the error terms based on the Lagrange multipliers, bias term, and kernel matrix. Helps in tracking misclassifications during training.

Return Trained Model: After training iterations, return the trained SVM model with updated Lagrange multipliers (α) and bias term (b).

Prediction Function: A function to predict the class label of a new data point using the trained SVM model. Computes the decision function and adds the bias term to determine the predicted class label.

Main Program: Example usage demonstrates how to train the SVM model with the specified dataset (X, Y), regularization parameter (C), and kernel function.

Model Supports

The project is executed using a combination of AWS services, Excel, Python, Jupyter notebook, Google Colab, and visualization tools. For our project: Instance Type: Specifically, the project makes use of the c7g.2xlarge computer-optimized EC2 instance type. Instance Specifications: The chosen instance type is equipped with 8 virtual CPUs and 16GB of memory.

Storage Integration: The project integrates Elastic Block Store (EBS) storage for efficient data storage and retrieval.

In the AWS environment, AWS EC2 instances, Lambda, and EventBridge are employed to automate the extraction of data from Reddit using the PRAW API. This ensures an efficient and streamlined workflow.

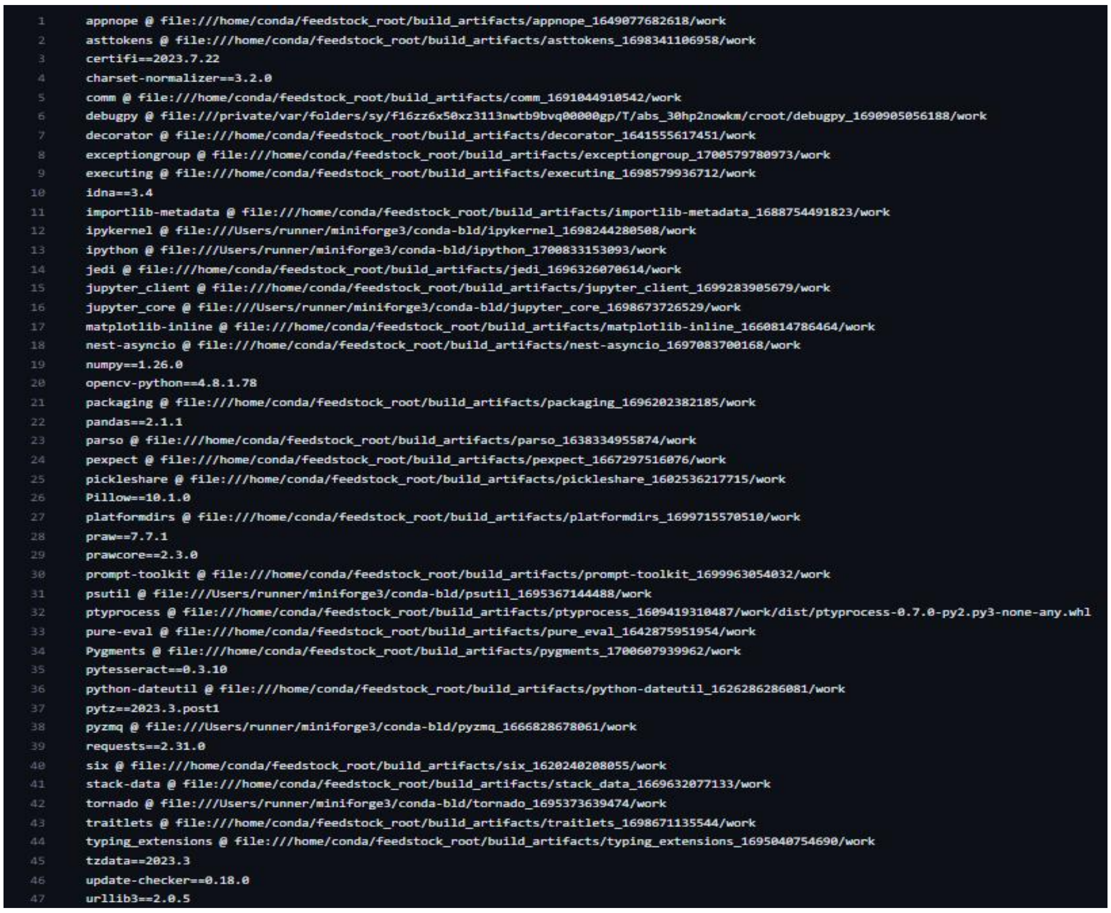
The data requirements suggests that the source and metadata information of the user posts could be gathered through a Reddit API called PRAW which stands for “Python Reddit

API Wrapper”. It is a python package that provides simple access to Reddit API. The user posts that are scrapped from Reddit can be from several news-based subreddits such as “news”, “Politics”, “WorldNews”, “fakenews” and many more. The scarped data is saved in comma-separated file with approx. 70,000 rows and 10 columns. Out of the ten columns, we will retain a few columns 5 such as “title”, “upvote_ratio”, “num_comments”, “url” etc are shown in Table 1 and we will discard the rest of the columns.

Python's robust ecosystem played a pivotal role in supporting various stages of the research pipeline, encompassing data collection, preprocessing, model training, testing, and overall model construction (Rastogi, 2022). The picture below provides a comprehensive overview of all the libraries employed throughout the research process as shown in the figure4,5. The requirements for the project were

Figure4:

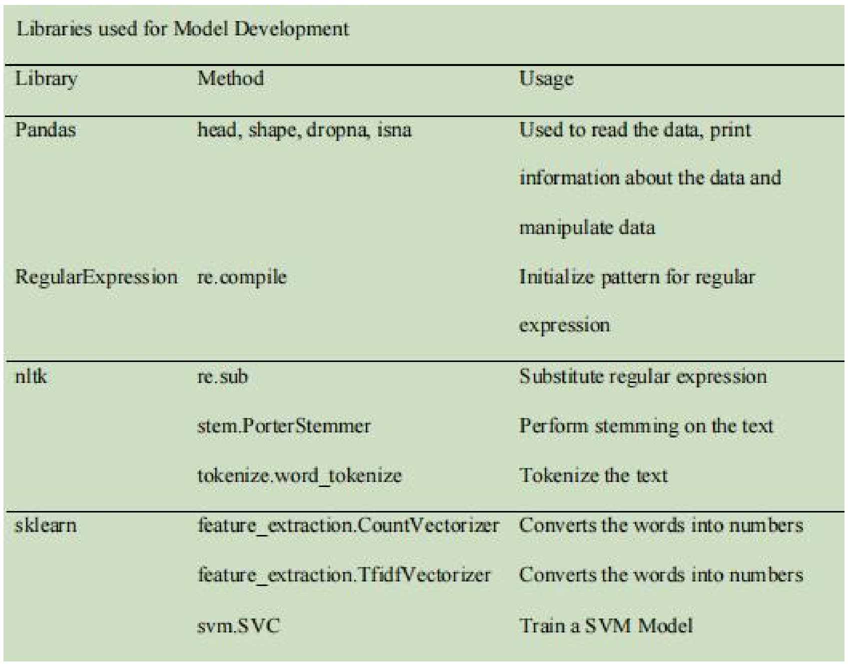
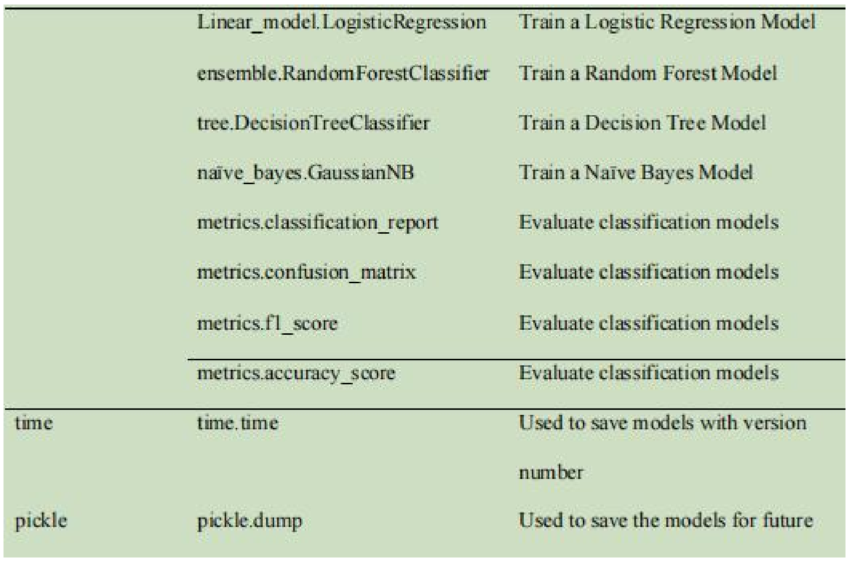
Project requirements



```
appnope @ file:///home/conda/feedstock_root/build_artifacts/appnope_1649077682618/work
asttokens @ file:///home/conda/feedstock_root/build_artifacts/asttokens_1698341106958/work
certifi==2023.7.22
charset-normalizer==3.2.0
comm @ file:///home/conda/feedstock_root/build_artifacts/comm_1691044910542/work
debugpy @ file:///private/var/folders/sy/f16zz6x50xz3113nwtb9bvq00000gp/T/abs_30hp2nowkm/croot/debugpy_1690905056188/work
decorator @ file:///home/conda/feedstock_root/build_artifacts/decorator_1641555617451/work
exceptiongroup @ file:///home/conda/feedstock_root/build_artifacts/exceptiongroup_1700579780973/work
executing @ file:///home/conda/feedstock_root/build_artifacts/executing_1698579936712/work
idna==3.4
importlib-metadata @ file:///home/conda/feedstock_root/build_artifacts/importlib-metadata_1688754491823/work
ipykernel @ file:///Users/runner/miniforge3/conda-bld/ipykernel_1698244280508/work
ipython @ file:///Users/runner/miniforge3/conda-bld/ipython_1700833153093/work
jedi @ file:///home/conda/feedstock_root/build_artifacts/jedi_1696326070614/work
jupyter_client @ file:///home/conda/feedstock_root/build_artifacts/jupyter_client_1699283905679/work
jupyter_core @ file:///Users/runner/miniforge3/conda-bld/jupyter_core_1698673726529/work
matplotlib-inline @ file:///home/conda/feedstock_root/build_artifacts/matplotlib-inline_1660814786464/work
nest-asyncio @ file:///home/conda/feedstock_root/build_artifacts/nest-asyncio_1697083700168/work
numpy==1.26.0
opencv-python==4.8.1.78
packaging @ file:///home/conda/feedstock_root/build_artifacts/packaging_1696202382185/work
pandas==2.1.1
parso @ file:///home/conda/feedstock_root/build_artifacts/parso_1638334955874/work
pexpect @ file:///home/conda/feedstock_root/build_artifacts/pexpect_1667297516076/work
pickleshare @ file:///home/conda/feedstock_root/build_artifacts/pickleshare_1602536217715/work
Pillow==10.1.0
platformdirs @ file:///home/conda/feedstock_root/build_artifacts/platformdirs_1699715570510/work
praw==7.7.1
prawcore==2.3.0
prompt-toolkit @ file:///home/conda/feedstock_root/build_artifacts/prompt-toolkit_1699963054032/work
psutil @ file:///Users/runner/miniforge3/conda-bld/psutil_1695367144488/work
ptyprocess @ file:///home/conda/feedstock_root/build_artifacts/ptyprocess_1609419310487/work/dist/ptyprocess-0.7.0-py2.py3-none-any.whl
pure-eval @ file:///home/conda/feedstock_root/build_artifacts/pure_eval_1642875951954/work
Pygments @ file:///home/conda/feedstock_root/build_artifacts/pygments_1700607939962/work
pytesseract==0.3.10
python-dateutil @ file:///home/conda/feedstock_root/build_artifacts/python-dateutil_1626286286081/work
pytz==2023.3.post1
pyzmq @ file:///Users/runner/miniforge3/conda-bld/pyzmq_1666828678061/work
requests==2.31.0
six @ file:///home/conda/feedstock_root/build_artifacts/six_1620240208055/work
stack-data @ file:///home/conda/feedstock_root/build_artifacts/stack_data_1669632077133/work
tornado @ file:///Users/runner/miniforge3/conda-bld/tornado_1695373639474/work
traitlets @ file:///home/conda/feedstock_root/build_artifacts/traitlets_1698671135544/work
typing_extensions @ file:///home/conda/feedstock_root/build_artifacts/typing_extensions_1695040754690/work
tzdata==2023.3
update-checker==0.18.0
urllib3==2.0.5
```

Figure5

List of all Libraries used for Model Deployment

Libraries used for Model Development

Library	Method	Usage
Pandas	head, shape, dropna, isna	Used to read the data, print information about the data and manipulate data
RegularExpression	re.compile	Initialize pattern for regular expression
nltk	re.sub	Substitute regular expression
	stem.PorterStemmer	Perform stemming on the text
	tokenize.word_tokenize	Tokenize the text
sklearn	feature_extraction.CountVectorizer	Converts the words into numbers
	feature_extraction.TfidfVectorizer	Converts the words into numbers
	svm.SVC	Train a SVM Model
	Linear_model.LogisticRegression	Train a Logistic Regression Model
	ensemble.RandomForestClassifier	Train a Random Forest Model
	tree.DecisionTreeClassifier	Train a Decision Tree Model
	naïve_bayes.GaussianNB	Train a Naïve Bayes Model
	metrics.classification_report	Evaluate classification models
	metrics.confusion_matrix	Evaluate classification models
	metrics.f1_score	Evaluate classification models
	metrics.accuracy_score	Evaluate classification models
time	time.time	Used to save models with version number
pickle	pickle.dump	Used to save the models for future

Model Architecture and Flow

We commence by formulating our problem statement and delving into the background of fake news detection. Given the significant impact of fake news on society, it is imperative to develop a robust solution. Recognizing the potential harm caused by the rapid spread of misinformation, we revisit existing research to identify gaps that our paper aims to address.

The dataset is acquired using PRAW, a Reddit API, ensuring an equal representation of fake and real data. Subsequently, we categorize each data point as either fake or real. To assess data quality, we scrutinize its richness in context and ensure the absence of null or special characters that might pose challenges for our model. Acknowledging that the model's efficacy is contingent on data quality, we transition from data comprehension to preparation by eliminating irrelevant data points. Additionally, we generate embeddings as inputs for certain machine learning models, (Tompkins, 2019). The training data will be used for model training, the validation set will be employed to assess performance and make improvements, while the test set will serve as an evaluation on unseen data. In the evaluation phase, we will compare the performance of our base models and ensembles on the test set.

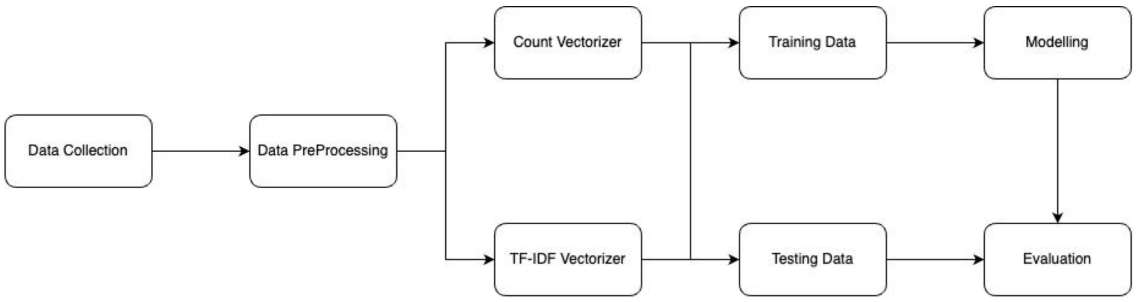
Project in depth, the initial phase of the study focused on optimizing the dataset for machine learning models through meticulous preprocessing steps, including text conversions such as converting to lowercase, removing unnecessary characters, and eliminating URLs to streamline the dataset. The dataset was then divided into training and test sets using an 80-20 split, enabling the development, evaluation, and validation of machine learning models. To enable numerical analysis, the processed text underwent transformation using Count Vectorizer and Tf-Idf Vectorizer techniques, exploring their impact on model performance (Reddy,2021).

This iterative process facilitated a comprehensive comparison of models and their responsiveness to different text representations. The research flow, illustrated in Figure6, depicts the sequential progression of raw text data through preprocessing, data splitting, text-to-vector transformation, model construction, and final evaluation and can be seen in figure(6).

Flow of the model building process

Figure6

Model Architecture and Support Diagram



Model Comparison and Justification

Figure 7

Model Comparison

Models	Advantages	Disadvantages
Logistic Regression	1. Logistic Regression provides clear interpretations of the coefficients, making it easy to understand the impact of each feature on the predicted outcome. 2. Training and prediction in Logistic Regression are computationally efficient, particularly for large datasets. 3. Logistic Regression models provide probabilities, allowing for a nuanced understanding of the certainty of predictions.	1. Logistic Regression assumes a linear relationship between the features and the log-odds of the response variable. 2. It may not capture complex relationships in the data if the decision boundary is not linear. 3. Logistic Regression can be sensitive to outliers, which may impact the model's performance.

Naive Bayes:	1. Naive Bayes models are computationally efficient and have fast training and prediction times. 2. Performs well in high-dimensional spaces, making it suitable for text classification tasks with a large number of features. 3. Naive Bayes is simple to understand and implement, making it a good choice for baseline models.	1. The model assumes independence between features, which may not hold true in real-world scenarios. 2. Naive Bayes can be sensitive to irrelevant or redundant features. 3. The model may not capture interactions between features well.
Decision Tree:	1. Decision Trees are easy to interpret and visualize, making them suitable for understanding model decisions. 2. Decision Trees do not require feature scaling, making them less sensitive to the scale of input features.	1. Decision Trees can be biased toward classes that are dominant in the dataset. 2. Decision Trees are prone to overfitting, especially on small datasets or with deep trees.
Random Forest:	1. Random Forests often provide high accuracy on various types of data & can handle missing data and maintain predictive accuracy with handling a large number of input feautres identifying crucial ones.	1. While each individual tree is interpretable,interpreting the overall Random Forest .can be expensive and may require more resources during training.
Support	1. SVMs can model non-linear	1. SVMs can be

Vector Machines (SVM):	relationships using various kernel functions. 2. SVMs are less prone to overfitting, especially in high-dimensional spaces. 3. SVMs perform well in high-dimensional feature spaces, making them suitable for text classification.	computationally intensive, particularly with large datasets and take a long time to run. 2. SVMs are not as interpretable as simpler models like logistic regression or decision trees.

Support Vector Machines (SVM) have proven highly advantageous for the task of fake news detection in this specific scenario. The elevated accuracy, precision, recall, and F1 score collectively indicate that SVM excels in minimizing both false positives (misclassifying real news as fake) and false negatives (misclassifying fake news as real). SVM showcases a well-balanced performance across all metrics, highlighting its resilience in handling the intricacies of language and patterns present in Reddit data. This reliability holds critical significance in the realm of fake news detection, where achieving precision and recall equilibrium is crucial for accurate identification without overlooking potential instances.

SVM has demonstrated significant advantages in the realm of fake news detection within this specific scenario utilizing the TF-IDF Vectorizer. The noteworthy accuracy, precision, recall, and F1 score collectively indicate that SVM adeptly achieves a delicate balance, minimizing both false positives (misclassifying real news as fake) and false negatives (misclassifying fake news as real). SVM's ability to consistently maintain high scores across all metrics underscores its robustness in deciphering the intricacies of language and patterns inherent in Reddit data. This reliability is paramount in the domain of fake news detection, highlighting SVM's capability to ensure accurate identification without overlooking potential instances.

Model Evaluation Methods

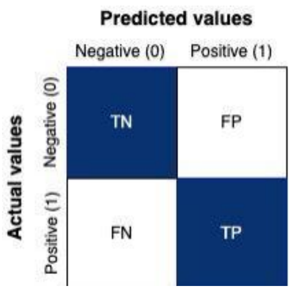
We start with binary classification models to classify news-related user posts from Reddit as fake or real. We evaluate these models on metrics such as accuracy, recall, precision, F1 score, and confusion matrix. The following subsection explains each of these metrics.

Confusion Matrix

The confusion matrix is a core technique for evaluating classification models as it provides a comprehensive overview of a model's performance. From figure8 Predictions are broken down into four categories: true positives (correctly predicted positive instances), true negatives (correctly predicted negative instances), false positives (incorrectly predicted as positive), and false negatives (incorrectly predicted as negative). This matrix is highly useful for assessing a model's strengths and weaknesses by facilitating the calculation of different performance metrics such as , precision, recall, and the F1 score, (Orellana, 2021).

Figure8

Confusion Matrix for Binary Classification



In the context of our project "Exploring the Use of Machine Learning and Natural Language Processing Techniques for Fake News Detection on Reddit," a confusion matrix plays a pivotal role by employing machine learning algorithms and natural language processing techniques, the goal is to identify and classify news articles on Reddit as either genuine or fake. The confusion matrix allows us to analyze the model's performance by quantifying how well it distinguishes between true and false information.

Precision

Precision is a key performance metric in the realm of machine learning and classification algorithms. It measures the accuracy of positive predictions made by a model, indicating the proportion of correctly identified positive instances among all instances predicted as positive. A high precision value signifies that when the model asserts a positive classification, it is more likely to be correct, minimizing the occurrence of false positives. .

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{5}$$

In this formula (5): True Positives (TP) are the instances correctly predicted as positive by the model. False Positives (FP) are the instances incorrectly predicted as positive by the model. The precision value ranges from 0 to 1, with higher values indicating better precision. It is an essential metric, especially in situations where the cost or consequences of false positives are significant, as it provides a clear understanding of the model's ability to make accurate positive predictions.

Recall

Recall is a performance metric that measures the ability of a model to correctly identify and retrieve all relevant instances of a particular class within a dataset. Also known as sensitivity or true positive rate.Recall is particularly important in scenarios where missing positive instances could have serious consequences, such as in medical diagnoses or fraud detection, where the goal is to ensure that as many relevant cases as possible are correctly identified. A high recall value indicates that the model is effective in capturing a significant portion of the positive instances, minimizing the risk of false negatives.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{6}$$

Accuracy

Accuracy is a fundamental performance metric in machine learning and statistical modeling, representing the proportion of correctly classified instances among the total predictions made by a model. It is calculated by dividing the sum of true positives and true negatives by the total number of predictions. In essence, accuracy provides a clear and intuitive measure of how well a model is performing overall.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{7}$$

In the confusion matrix, the number of correct predictions to total number of predictions is accuracy, here the sum of True positives and true negatives to the total number of predictions is the Accuracy.

F1 score

The F1 score is a metric commonly used in machine learning and statistical analysis to assess the overall performance of a classification model. The F1 score is calculated as the harmonic mean of precision and recall, emphasizing the importance of both false positives and false negatives in its computation. It is calculated using the following formula:

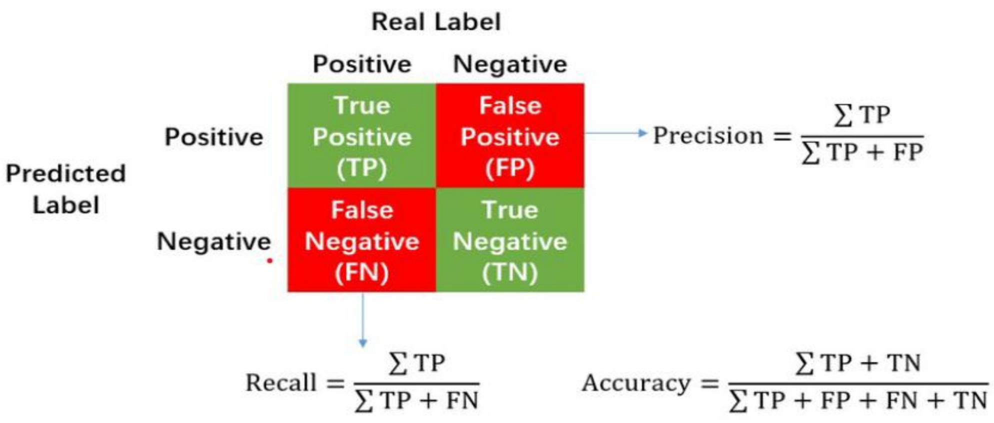
$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{8}$$

A high F1 score indicates a well-balanced model that effectively manages both precision and recall, while a lower score suggests an imbalance between false positives and false negatives.

Given below in figure9 is a confusion matrix which shows how we can derive all other metrics from it namely, Precision, Recall, Accuracy and F1 score.

Figure9

Calculation of all metrics from Confusion Matrix

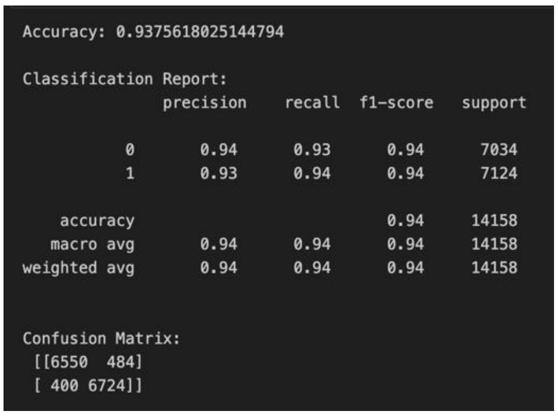

Real Label
Positive
Negative
Predicted Label
Positive
Negative
True Positive (TP)
False Positive (FP)
False Negative (FN)
True Negative (TN)
$\text{Precision} = \frac{\sum TP}{\sum TP + FP}$
$\text{Recall} = \frac{\sum TP}{\sum TP + FN}$
$\text{Accuracy} = \frac{\sum TP + TN}{\sum TP + FP + FN + TN}$

Model Validation and Evaluation Results

The baseline model for the Support Vector Machine is created using SVC from the Scikit-learn library. The default parameter kernel is set to 'linear' and used to train the model on Count and TF-IDF vectorized data. Figures 10,11,12,13,show the evaluation metrics for the baseline model using Count Vectorizer and TF-IDF Vectorizer, respectively.

Figure 10,11

Evaluation Metrics for SVM Model with Count Vectorizer



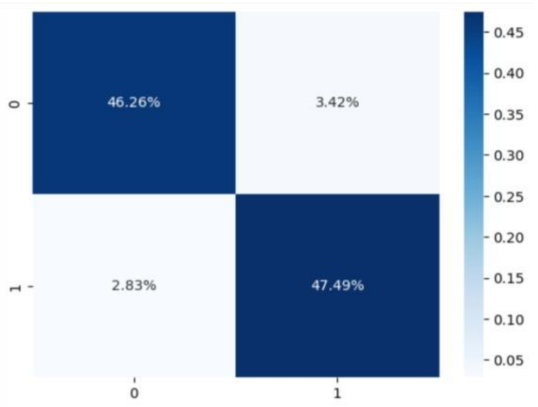
```
Accuracy: 0.9375618025144794

Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.93      0.94      7034
           1       0.93      0.94      0.94      7124

    accuracy                           0.94     14158
   macro avg       0.94      0.94      0.94     14158
weighted avg       0.94      0.94      0.94     14158


Confusion Matrix:
 [[6550  484]
 [ 400 6724]]
```



In our project, the use of Count Vectorizer is a crucial component in the application of machine learning and natural language processing techniques for Fake News Detection on

Reddit. Count Vectorizer is a text preprocessing technique that transforms a collection of text documents into a numerical format suitable for machine learning algorithms. It works by converting a text corpus into a matrix of token counts, representing the frequency of each word in the document, (Reddy, 2021).

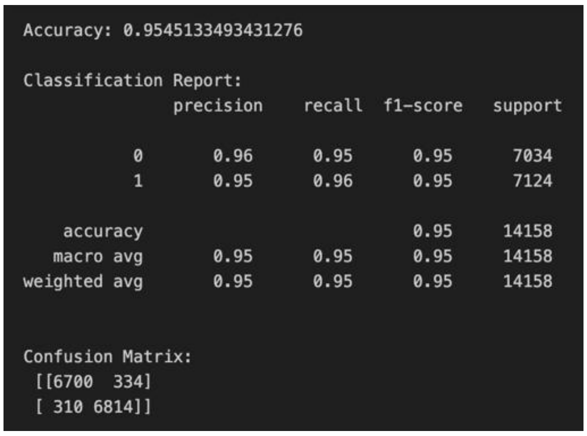
In the context of our Support Vector Machine (SVM) model, the Count Vectorizer likely played a vital role in converting the textual data from Reddit posts into a format that SVM can comprehend. The effectiveness of Count Vectorizer in achieving an accuracy of 0.9375 is notable. The confusion matrix further illustrates the model's performance, with 6550 true negatives, 6724 true positives, 484 false positives, and 400 false negatives. The high number of true positives and true negatives indicates that the SVM model, with the aid of Count Vectorizer, is proficient at correctly classifying both genuine and fake news posts on Reddit as seen from figure 10,11.

Support Vector Machines (SVM) also demonstrated strong performance, achieving an accuracy of 93.75%. With precision, recall, and F1 score consistently at 94%, SVM proves its efficacy in handling high-dimensional feature spaces and delineating clear decision boundaries.

In conclusion, the use of Count Vectorizer in conjunction with SVM has proven to be effective in our Fake News Detection project on Reddit. The high accuracy and the balanced distribution of true positives and true negatives in the confusion matrix signify that the model is robust and capable of distinguishing between real and fake news with a high degree of accuracy, (Rastogi, 2022). Further analysis and refinement of the model could potentially enhance its performance and contribute to more accurate fake news detection on the Reddit platform.

Figure 12,13

Evaluation Metrics for SVM Model with TF-IDF Vectorizer



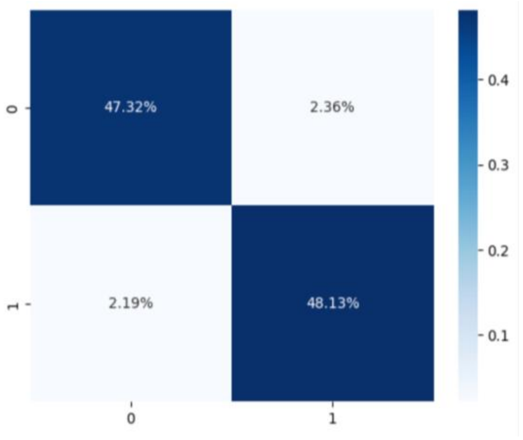
```
Accuracy: 0.9545133493431276

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.95      0.95      7034
           1       0.95      0.96      0.95      7124

    accuracy                           0.95     14158
   macro avg       0.95      0.95      0.95     14158
weighted avg       0.95      0.95      0.95     14158


Confusion Matrix:
 [[6700  334]
 [ 310 6814]]
```



In the project the TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer played a crucial role in transforming the raw text data into a numerical format suitable for machine learning algorithms, particularly Support Vector Machines (SVM). TF-IDF is a statistical measure that evaluates the importance of a word in a document relative to a collection of documents, (Reddy, 2021). The vectorizer assigns higher weights to words that are frequent within a specific document but rare across the entire dataset.

The effectiveness of TF-IDF in this context is evident from the achieved accuracy of 95.45% and the corresponding confusion matrix. The high accuracy indicates that the SVM model, when trained on TF-IDF-transformed data, demonstrated a strong ability to discern between genuine and fake news on Reddit. The confusion matrix further reveals that the model made 6700 true positive predictions (correctly identifying genuine news) and 6814 true negative predictions (correctly identifying fake news). The low values in the off-diagonal elements (334 false positives and 310 false negatives) demonstrate the model's ability to minimize both types of errors.

Among the models, Random Forest and Support Vector Machines emerged as the top performers, achieving high accuracy rates of 95.31% and 95.45%, respectively. These models demonstrated remarkable consistency in precision, recall, and F1 score, all consistently reaching 95%.

In conclusion, the use of the TF-IDF vectorizer in conjunction with SVM proved highly effective for fake news detection on Reddit. The model's impressive accuracy and the balanced distribution of true positives and true negatives in the confusion matrix signify a robust performance (Rastogi, 2022). This success suggests that the combination of TF-IDF vectorization and SVM is a promising approach for identifying misleading information within the context of online content on platforms like Reddit.

The persistent and impressive performance of SVM, as emphasized by its metrics, establishes it as a standout model for fake news detection when utilizing the Count Vectorizer and TF-IDF Vectorizer (Gandhi, 2018). In conclusion, SVM, coupled with the chosen vectorization technique, emerges as a potent and dependable choice for precisely identifying fake news on the Reddit platform in the specified project.

References

Khanam, Z., Alwasel, B. N., Sirafi, H., & Rashid, M. (2021). Fake news detection using machine learning approaches. IOP Conference Series: Materials Science and Engineering, 1099(1),

012040. https://doi.org/10.1088/1757-899x/1099/1/012040

Rastogi, S., & Bansal, D. (2022). A review on fake news detection 3T's: Typology, time of detection, taxonomies. International Journal of Information Security, 22(1), 177–212.

https://doi.org/10.1007/s10207-022-00625-3

Tompkins Jillian. (2019). Disinformation Detection: A review of linguistic feature selection and classification models in news veracity assessments.

https://doi.org/10.48550/arXiv.1910.12073

Gandhi, R. (2018, July 5). Support Vector Machine - introduction to machine learning algorithms. Medium. Retrieved February 17, 2022, from https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

Reddy, A. V. (2021, July 30). NLP using NLTK Library | NLTK Library for Natural Language Processing. Analytics Vidhya. Retrieved April 28, 2022, from https://www.analyticsvidhya.com/blog/2021/07/getting-started-with-nlp-using-nltk-librar