

Project Report
On
Fake News Authentication using PySpark MLlib and NLP



Submitted
In partial fulfilment
For the award of the Degree of

PG-Diploma in Big Data Analytics

C-DAC, ACTS (Pune)

Guided By:

Mr. Shiva Karthik S.

Submitted By:

Jay Narendra Sharma (220940125014)

Ketkale Anmol Shrikant (220940125019)

Nagare Akash Shirish (220940125024)

Rishabh Tripathi (220940125036)

Rohan Jacob Alexander (220940125037)

Centre for Development of Advanced Computing

(C-DAC), ACTS (Pune- 411008)

Acknowledgement

This is to acknowledge our indebtedness to our Project Guide, **Mr. Shiva Karthik S**, C-DAC ACTS, Pune for her constant guidance and helpful suggestion for preparing this project **Fake News Authentication using PySpark MLlib and NLP**. We express our deep gratitude towards him for inspiration, personal involvement, constructive criticism that he provided us along with technical guidance during the course of this project.

We take this opportunity to thank Head of the department **Mr. Gaur Sunder** for providing us such a great infrastructure and environment for our overall development.

We express sincere thanks to **Mrs. Namrata Ailawar**, Process Owner, for their kind cooperation and extendible support towards the completion of our project.

It is our great pleasure in expressing sincere and deep gratitude towards **Mrs. Risha P R (Program Head)** and **Mrs. Srujana Bhamidi** (Course Coordinator, PG-DBDA) for their valuable guidance and constant support throughout this work and help to pursue additional studies.

Also, our warm thanks to **C-DAC ACTS Pune**, which provided us this opportunity to carry out, this prestigious Project and enhance our learning in various technical fields.

Jay Narendra Sharma (220940125014)

Ketkale Anmol Shrikant (220940125019)

Nagare Akash Shirish (220940125024)

Rishabh Tripathi (220940125036)

Rohan Jacob Alexander (220940125037)

ABSTRACT

The proliferation of fake news has become a major challenge in our current society. Misinformation and propaganda can have serious consequences on individuals, communities, and society as a whole. Therefore, the need to detect and classify fake news has become increasingly important. In this project, we present a system for detecting and classifying fake news articles using PySpark MLlib and NLP. The core feature extraction technique used in this system is Word2Vec, a widely used approach for text data analysis. The system is trained on a diverse dataset collected from reputable sources such as Kaggle, McIntire, Reuters, BuzzFeed Political, and other websites. This ensures that the system is reliable and accurate, and can identify different types of fake news articles. To collect data for model authentication, we used BeautifulSoup, a popular web scraping library in python that allows us to extract news articles from various websites. This ensures that the system is trained on a comprehensive dataset that covers different sources and types of news articles. The system is deployed using Streamlit, a user-friendly web application framework that allows users to interact with the system in a simple and intuitive way. The system can classify news articles as either "real" or "fake". Overall, this project provides a valuable contribution to the fight against the proliferation of fake news articles. By detecting and classifying fake news, the system can help prevent the spread of misinformation and promote more informed decision-making. The use of PySpark MLlib and NLP, along with Word2Vec as the core feature extraction technique, ensures that the system is reliable, accurate, and scalable.

Table of Contents

S. No	Title	Page No.
	Front Page	I
	Acknowledgement	II
	Abstract	III
	Table of Contents	IV
1	Introduction	01-02
1.1	Introduction	01
1.2	Objective and Specifications	02
2	Literature Review	03-04
3	Methodology/ Techniques	05-15
3.1	Machine Learning Models	05-07
3.2	Data Extraction	08-09
3.3	Model Deployment	10-15
4	Implementation	16-19
4.1	Implementation	16
5	Results	20-21
5.1	Results	20
6	Conclusion	22
6.1	Conclusion	22
7	References	23
7.1	References	23

Chapter 1

Introduction

1.1 Introduction

In recent years, the proliferation of fake news has become a major challenge in our society. Misinformation and propaganda can have serious consequences on individuals, communities, and society as a whole. Therefore, there is a pressing need to develop tools and systems that can detect and classify fake news articles. In this project, we present a system that can automatically detect and classify fake news articles using PySpark MLlib and NLP. PySpark MLlib is a powerful machine learning library that provides tools for building scalable and efficient machine learning models. By leveraging PySpark MLlib, we can process large datasets and build models that can analyze text data and classify news articles as either "real" or "fake." Natural Language Processing (NLP) is used to preprocess and extract features from the text data, and Word2Vec is used as the core feature extraction technique. The system is trained on a diverse dataset collected from reputable sources, including Kaggle, McIntire, Reuters, BuzzFeed Political, and other websites. To collect data for model authentication, we used BeautifulSoup, a popular web scraping library that allows us to extract news articles from various websites. The system is deployed using Streamlit, a user-friendly web application framework that provides a simple and intuitive user interface. Users can interact with the system by submitting news articles for classification, and the system provides detailed information about the article, including the source, date, and content. This system provides a valuable contribution to the fight against the proliferation of fake news articles. By using PySpark MLlib and NLP, along with Word2Vec as the core feature extraction technique, we can build a reliable and scalable system that can detect and classify fake news articles with high accuracy.

1.2 Objective

The objectives of the project work are -

- To develop a system that can detect and automatically classify fake news articles using PySpark MLlib and NLP.
- To train the system on a diverse dataset collected from reputable sources, including Kaggle, McIntire, Reuters, BuzzFeed Political, and other websites.
- To use BeautifulSoup to collect data for model authentication, by scraping news articles from various websites.
- To use Word2Vec as the core feature extraction technique, which can accurately capture the semantic relationships between words in the text data.
- To deploy the system using Streamlit, a user-friendly web application framework that provides a simple and intuitive user interface.
- To provide a valuable contribution to the fight against the proliferation of fake news articles, by developing a reliable and scalable system that can detect and classify fake news articles with high accuracy.

Chapter 2

LITERATURE REVIEW

Syafrial Fachri Pane et al. [1] The research proposes the use of the random forest model and sentiment analysis to detect COVID-19 fake news on Indonesian Tweets, and evaluates the performance of the sentiment analysis feature using the Spark Dataframe. The study divided the machine learning development stages into several steps, including data collection, sentiment analysis using Apache Spark, and classification using random forest with Spark MLlib. The results show that sentiment analysis does not significantly improve the prediction model, with the random forest model producing an accuracy of 0.787 for both models with and without sentiment analysis.

Tshegofatso Thate et al. [2] This research addresses the critical issue of fake news, which has become prevalent due to technological advancements and social media. The study benchmarks several Natural Language Processing classifiers against a common dataset and introduces natural language features to improve the classification of fake news. The findings indicate that the accuracy of classifying an article as fake news is based on previously seen truthful and fake news. The study recommends increasing the sample size to improve the results and highlights the importance of bespoke features in classifying fake news accurately. Overall, this research adds to the growing body of knowledge on detecting and combating the spread of fake news in the digital age.

Anjali Jain et al. [3] This paper addresses the issue of fake news circulating on social media platforms and presents a model for detecting it using machine learning and natural language processing techniques. The author highlights the importance of verifying news sources and preventing rumors from spreading in society. The proposed model uses Support Vector Machine to classify news articles as real or fake, and achieves a high accuracy rate of 93.6%. The study contributes to the field of fake news detection and offers a promising approach for addressing this important issue.

Abdullah-All-Tanvir et al. [4] This paper addresses the issue of detecting fake news on Twitter, which has become one of the dominant news sources in recent years. The authors propose a model to automatically identify fake news messages by predicting the accuracy ratings of Twitter posts. They compare the performance of five different machine learning algorithms, including Support Vector Machine, Naïve Bayes, Logistic Regression, and Recurrent Neural Network models, and find that SVM and Naïve Bayes classifiers outperform the others. The study highlights the importance of automating fake news detection to maintain a robust online media and social network. Overall, this paper provides valuable insights for researchers and practitioners working in the field of fake news detection and social media analytics.

Karishnu Poddar et al. [5] The paper presents a computational model using probabilistic and geometric machine learning models to tackle the issue of fake news. The study compares the scores of two different vectorizers and various classifiers to predict the fake news. The results show that Support Vector Machine (SVM) with the TF-IDF vectorizer provides the most accurate prediction. The use of English stop words was also shown to improve the scores. Overall, the study provides a valuable contribution to the field of fake news detection using machine learning techniques.

Chapter 3

Methodology and Techniques

MACHINE LEARNING MODELS

Logistic Regression:

Logistic regression is a widely used classification algorithm in machine learning. It is a type of supervised learning algorithm that can be used for binary classification problems, where the goal is to predict the probability of an input belonging to one of two classes.

In machine learning, logistic regression works by learning the relationship between the input features and the output label using a training dataset. The algorithm uses a logistic function to transform the output of a linear combination of the input features into a probability value between 0 and 1.

During training, the logistic regression model is fitted to the training data by minimizing a cost function, such as the cross-entropy loss. The parameters of the model, which include the weights for each input feature, are adjusted iteratively to find the values that minimize the cost function.

Once the model is trained, it can be used to predict the class label of new data points based on their input features. The logistic regression model assigns a probability value to each class, and the class with the highest probability is selected as the predicted label. Logistic regression has several advantages in machine learning, including its simplicity, interpretability, and ability to handle noise in the input data. However, it also has limitations, such as its assumption of a linear relationship between the input features and the output label, and its inability to handle non-linear decision boundaries. In practice, logistic regression is often used in conjunction with other machine learning algorithms, such as decision trees or neural networks, to improve performance on more complex tasks.

Random Forest Classifier:

Random forest is a popular machine learning algorithm used for classification and regression tasks. It is an ensemble learning method that builds a collection of decision trees and combines their predictions to obtain a more accurate and stable prediction.

In a random forest classifier, each tree is built using a random subset of the training data and a random subset of the input features. This helps to reduce the variance of the model and prevent overfitting to the training data. The algorithm then aggregates the predictions of each individual tree to obtain a final prediction.

The random forest algorithm has several advantages over other machine learning algorithms. It is robust to noise and outliers in the data, can handle large and high-dimensional datasets, and provides a measure of feature importance that can be used for feature selection.

Random forest classifiers are widely used in various applications, including image

recognition, text classification, and bioinformatics. They have been shown to be effective in a wide range of tasks and are often used as a baseline algorithm for comparison with more advanced methods.

However, random forests also have some limitations, such as their relatively high computational cost and difficulty in interpreting the results. Overall, random forests are a powerful and versatile machine learning algorithm that can be used for a variety of classification tasks.

Gradient-boosted Tree classifiers:

Gradient-boosted tree classifiers are a type of ensemble learning method used for classification and regression tasks. They combine multiple decision trees to obtain a more accurate and robust prediction.

The algorithm works by iteratively adding decision trees to the ensemble, with each subsequent tree correcting the errors made by the previous trees. The trees are constructed using a gradient descent algorithm, where the objective is to minimize a loss function that measures the difference between the predicted and actual values.

During training, the algorithm learns the optimal weights for each input feature that are used to make predictions. The weights are updated at each iteration based on the gradient of the loss function with respect to the predicted values. The learning rate parameter controls the step size of the gradient descent algorithm and determines how quickly the model converges to the optimal solution.

Once the model is trained, it can be used to predict the class label of new data points by aggregating the predictions of all the decision trees in the ensemble.

Gradient-boosted tree classifiers have several advantages over other machine learning algorithms. They are highly accurate and can handle complex nonlinear relationships between the input features and the output variable. They also provide a measure of feature importance that can be used for feature selection. Overall, gradient-boosted tree classifiers are a powerful and widely used machine learning algorithm for classification and regression tasks.

Linear Support Vector Machine:

Linear Support Vector Machine (SVM) is a popular machine learning algorithm used for classification tasks. It works by finding the optimal hyperplane that separates the input data into different classes. The algorithm tries to maximize the margin, which is the distance between the hyperplane and the closest data points of each class.

The linear SVM classifier works by transforming the input data into a high-dimensional space, where a linear hyperplane can be used to separate the classes. The SVM algorithm tries to find the optimal hyperplane that maximizes the margin in this high-dimensional space.

During training, the SVM algorithm tries to find the optimal values of the weight vector and bias term that define the hyperplane. This is done by minimizing a cost function, which penalizes misclassifications and encourages a large margin between the hyperplane and the data points.

Once the SVM model is trained, it can be used to predict the class label of new data

points based on their position relative to the hyperplane. Data points that are on the positive side of the hyperplane are classified as one class, while data points that are on the negative side are classified as the other class.

Linear SVM classifiers have several advantages in machine learning, including their ability to handle high-dimensional data, their robustness to noise and outliers, and their ability to handle nonlinearly separable data using kernel functions. They are widely used in various applications, including image recognition, text classification, and bioinformatics.

Factorization Machines Classifier:

Factorization machines (FM) are a type of machine learning algorithm used for classification and regression tasks. They are based on a linear model with additional low-rank interactions between the input features.

The FM algorithm works by learning a set of latent vectors for each input feature that capture the underlying relationships between the features. These latent vectors are used to compute the pairwise interactions between the features, which can capture complex nonlinear relationships that are not captured by a linear model.

During training, the FM algorithm learns the optimal values of the weight vector and the latent vectors that minimize a cost function, such as the log-loss function for classification tasks. This is done using an optimization algorithm, such as stochastic gradient descent.

Once the model is trained, it can be used to predict the class label of new data points based on their feature values and the learned latent vectors. The pairwise interactions between the features are computed using the dot product of their corresponding latent vectors.

FM classifiers have several advantages over other machine learning algorithms. They can capture complex nonlinear relationships between the input features, even in the presence of sparse and high-dimensional data. They are also computationally efficient and can handle large datasets.

DATA EXTRACTION

Beautiful Soup:

Beautiful Soup is a Python library used for web scraping and parsing HTML and XML documents. It provides a convenient way to extract data from web pages and manipulate the HTML or XML tree structure.

The library works by creating a parse tree from the input HTML or XML document. It then provides a set of methods for navigating and searching the parse tree to extract specific elements or attributes. The library also provides methods for modifying the parse tree and writing the modified tree back to a file.

Some of the key features of Beautiful Soup include:

1. Support for different parsers: Beautiful Soup supports various parsing libraries, including lxml, html5lib, and built-in Python parsers. This makes it flexible and able to handle different types of HTML and XML documents.
2. Navigable parse tree: Beautiful Soup creates a parse tree that can be navigated like a nested data structure, allowing users to easily access and extract specific elements and attributes.
3. Search and filtering: Beautiful Soup provides a powerful search and filtering mechanism that allows users to find specific elements based on various criteria, such as tag name, attribute value, or text content.
4. Encoding detection and conversion: Beautiful Soup can automatically detect the character encoding of the input document and convert it to Unicode, making it easy to work with text in different languages.
5. Robust error handling: Beautiful Soup is designed to handle malformed HTML and XML documents, and it provides robust error handling to prevent crashes and unexpected behavior.

Overall, Beautiful Soup is a powerful and flexible tool for web scraping and parsing HTML and XML documents. It is widely used in data science and web development to extract and manipulate data from web pages.

Requests:

Requests is a popular Python library used for making HTTP requests. It simplifies the process of sending HTTP requests and handling HTTP responses in Python.

With Requests, you can easily send GET, POST, PUT, DELETE, and other HTTP requests to web servers and APIs. The library provides an intuitive and easy-to-use API for making requests and handling responses, with features such as:

1. Request customization: Requests allows you to set headers, parameters, and other request options to customize your HTTP requests.
2. Response handling: Requests provides an easy-to-use interface for handling HTTP responses, including decoding JSON and other response data formats.
3. Session management: Requests supports session management, which allows you to persist cookies and other session data across multiple requests.
4. Authentication: Requests supports a variety of authentication methods, including Basic Auth, Digest Auth, and OAuth.
5. SSL verification: Requests provides SSL verification options to ensure secure connections to web servers and APIs.

Requests is a powerful and widely used library in the Python ecosystem for working with HTTP requests and web APIs. It is widely used in web development, data science, and other areas of Python programming.

DEPLOYMENT

Streamlit:

Streamlit is an open-source Python library used for building interactive web applications for data science and machine learning projects. It allows users to create web applications with simple Python scripts, without requiring expertise in web development or web frameworks.

With Streamlit, users can create interactive data visualizations, build custom user interfaces for machine learning models, and deploy their applications to the web. Streamlit provides a simple and intuitive API for building web applications, with features such as:

1. Customizable widgets: Streamlit provides a variety of widgets for building custom user interfaces, including sliders, text inputs, and dropdowns.
2. Interactive visualizations: Streamlit supports a variety of data visualization libraries, including Matplotlib, Plotly, and Altair, allowing users to create interactive and dynamic visualizations.
3. Real-time data updates: Streamlit automatically updates the web application in real-time as users interact with it, allowing for fast and responsive data exploration.
4. Seamless deployment: Streamlit provides an easy-to-use deployment platform, allowing users to deploy their applications to the web with a single command.

Streamlit is a powerful and flexible tool for building web applications for machine learning and data science projects. It is widely used in the Python ecosystem for creating interactive dashboards, data exploration tools, and machine learning model visualizations.

NATURAL LANGUAGE PROCESSING

Natural Language Processing:

Natural Language Processing (NLP) is a field of computer science and artificial intelligence that focuses on the interaction between computers and human language. It involves the use of computational techniques and algorithms to analyze, understand, and generate human language.

NLP is used in a wide range of applications, including machine translation, sentiment analysis, chatbots, and text summarization. NLP tasks can be broadly classified into two categories:

1. **Natural Language Understanding (NLU):** NLU involves the analysis of human language to extract meaning from it. This includes tasks such as part-of-speech tagging, named entity recognition, sentiment analysis, and text classification.
2. **Natural Language Generation (NLG):** NLG involves the generation of human language by computers. This includes tasks such as machine translation, text summarization, and dialogue generation.

NLP techniques are based on various mathematical and statistical models, including probability theory, machine learning, and deep learning. Some of the common techniques used in NLP include:

1. **Tokenization:** This involves breaking down text into smaller units, or tokens, such as words or phrases.
2. **Stemming and Lemmatization:** These techniques are used to reduce words to their base or root forms, in order to improve text analysis.
3. **Named Entity Recognition:** This involves identifying and classifying named entities in text, such as people, organizations, and locations.
4. **Sentiment Analysis:** This involves analyzing the tone and emotion of text to determine the sentiment of the author.
5. **Machine Translation:** This involves automatically translating text from one language to another.

NLP is a rapidly growing field with many applications in industries such as healthcare, finance, and marketing. It is an important area of research and development for creating

more human-like interactions between computers and people.

Word2Vec:

Word2Vec is a popular algorithm used for generating word embeddings, which are vector representations of words in a high-dimensional space. The algorithm was developed by Tomas Mikolov and his team at Google in 2013, and has since become a widely-used tool for natural language processing (NLP) tasks.

Word2Vec uses a neural network model to learn vector representations of words based on their context in a large corpus of text. The algorithm works by training the neural network on a large dataset of text, with the goal of predicting the probability of a word given its context (i.e., the words that appear before and after it in a sentence). During training, the network adjusts the weights of the connections between the neurons to improve its predictions, and these weights ultimately serve as the word embeddings.

There are two main architectures for Word2Vec: Continuous Bag of Words (CBOW) and Skip-Gram. In CBOW, the model tries to predict a target word based on its surrounding context words, while in Skip-Gram, the model tries to predict the context words given a target word. Both architectures have their own advantages and disadvantages depending on the specific use case.

Once trained, Word2Vec embeddings can be used to measure the similarity between words, to perform analogies (e.g., "king" - "man" + "woman" = "queen"), and for a variety of other NLP tasks such as text classification and clustering. The embeddings can also be visualized in a lower-dimensional space using techniques such as t-SNE, which can provide insights into the relationships between words in the high-dimensional space.

Overall, Word2Vec is a powerful and widely-used algorithm for generating word embeddings, and has contributed significantly to the field of natural language processing.

Dataset

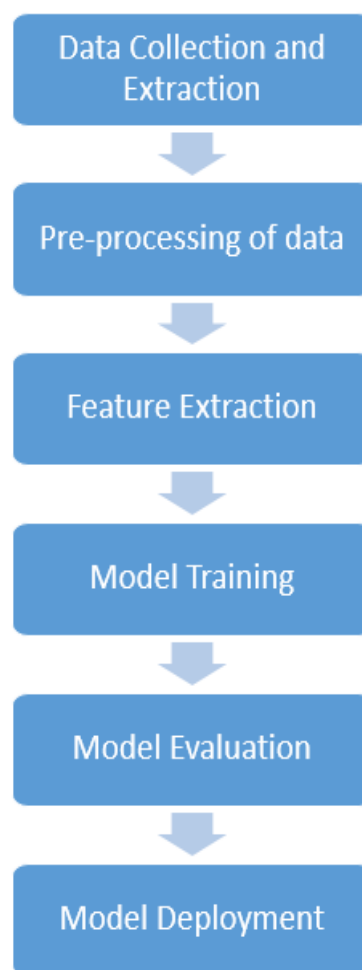
The IAM dataset has been used for training the model. The IAM database consists of handwritten English sentences. It is based on the Lancaster-Oslo/Bergen (LOB) corpus. The database serves as a basis for a variety of recognition tasks, particularly useful in recognition tasks where linguistic knowledge beyond the lexicon level is used, as this knowledge is automatically derived from the underlying corpus.

The IAM database also includes a few image-processing procedures for extracting the handwritten text from the forms and the segmentation of the text into lines and words. The training of the model is done using the IAM dataset along with the IAM dataset,

custom handwritten paragraph images collected from random people were used for testing. These custom images were captured under normal lighting with a 5MP camera with a resolution of range 800 to 1000 dpi.

Model Description:

Flowchart of the project



Data Collection and Extraction:

We have used the ISOT Fake News Dataset published by Ahmed H, Traore I, Saad S. for training our machine learning model. This dataset was collected from real world sources, the articles were scrapped from truthful articles at Reuters.com (News website). The size of Real-News articles is 21417 and for Fake-News is 23481 world news and political news articles. We have used another dataset WELFake: Word Embedding Over Linguistic Features For Fake News Detection by authors Pawan Kumar Verma, Prateek Agrawal, Ivone Amarim, and Radu Prodan. It contains 35028 real news and 37106 fake news articles taken from various web sources like Kaggle, McIntire, Reuters, Buzzfeed Political etc. to prevent overfitting of classifiers and to provide more text data for better ML Model training.

We have extracted data from various news websites like CNN, ABC News, The Times Of India, Indian Express and India Today for evaluating and authentication of our ML Model's accuracy.

Preprocessing of Data:

We have used Word Tokenization and Stop Words Removal techniques of Natural Language Processing for the preprocessing of our data. Word Tokenizer is used to break the sentence into separate words or tokens and array of words. That array of words is used as an input to the Stop Words Remover to eliminate unimportant words allowing our model to focus to the important information and provide more accuracy at the process of News Validation. We removed all the null rows because null values adversely affect the performance and accuracy of the machine learning algorithms. We used only Word Tokenization and Stop Words Removal techniques for training our model because we wanted our model to be more flexible and accurate models.

Feature Extraction:

In feature extraction, we take raw input data and transform it into a set of features that can be used to represent the data in a way that is more suitable for analysis and machine learning algorithms. In our project we have used Word2Vec NLP technique. We took raw input string and transformed it into vectors that can be used to represent the data. Word2Vec uses a neural network to learn word embeddings from a large corpus of text. The basic idea behind Word2Vec is to train a neural network to predict the context in which a word appears. The context of a word is defined by the words that appear around it in a sentence. Word2Vec can be trained using two different techniques: continuous bag of words (CBOW) and skip-gram.

In CBOW, the neural network is trained to predict a target word given its context. In skip-gram, the neural network is trained to predict the context words given a target word. Both techniques can generate word embeddings that can be used for a variety of NLP tasks, such as language modeling, machine translation, sentiment analysis, and many more.

Model Training:

We trained our model using the ISOT Fake News Dataset and WELFake Word Embedding Using Linguistic Features for Fake News Detection Datasets.

Model Evaluation:

For Model Evaluation, we thought of using articles from various news websites so that the model should be able to provide more accurate output when a news article is feed to it. We used BeautifulSoup Python Library for scrapping news articles from the news websites and then created CSV files of those articles using Pandas library and then feed those Test Datasets to our ML Models. BeautifulSoup is a Python library used for web scraping purposes to pull the data out of HTML and XML files. It creates a parse tree from page source code that can be used to extract data in a hierarchical and more readable manner. Requests is a Python library used to send HTTP requests and handle responses. It is often used in conjunction with other web scraping libraries like BeautifulSoup to make HTTP requests and retrieve the HTML content of web pages.

Model Deployment:

We have used Streamlit for our model deployment. It provides us a user interface to code. We can integrate text, images, animations and other objects in Streamlit. We do not need any external language like HTML to build the web application interface. Streamlit runs on top of Flask which is a micro web framework for Python Language and it is frequently used to build Python based web applications. Streamlit provides a set of pre-built components and widgets that can be used to create interactive user interfaces, such as sliders, checkboxes, and text inputs. It also provides easy integration with popular data science libraries such as Pandas, NumPy, and Matplotlib, allowing developers to easily visualize and explore data.

Chapter 4

Implementation

1. Use of **Google Colab, PySpark, Spark MLlib, Beautiful Soup, Streamlit.**
2. Hardware and Software Configuration:

Hardware Configuration:

- CPU: 8 GB RAM, Quad core processor
- GPU: 16GB RAM Nvidia's GTX 1080Ti

Software Required:

- **Google Colab:** is a cloud-based service provided by Google that allows users to write, run, and share Python code notebooks online. It provides free access to computing resources like GPUs and TPUs, making it an attractive option for machine learning and data science projects. With Colab, users can create, edit, and share Jupyter notebooks that contain live code, equations, visualizations, and narrative text. Colab notebooks can be easily shared with others, making it a popular tool for collaborative projects. Additionally, Colab integrates with Google Drive, making it easy to save and share your work with others.
- **PySpark:** It is the Python API for Apache Spark, an open-source big data processing engine. Apache Spark is designed to process large-scale data in a distributed computing environment, using a cluster of computers to perform computations in parallel. PySpark allows Python developers to leverage the power of Spark's distributed processing capabilities using the familiar syntax of Python. It provides an easy-to-use interface for working with big data and enables data scientists and analysts to perform tasks such as data cleaning, data processing, data analysis, and machine learning on large datasets. PySpark also supports a wide range of data sources, including Hadoop Distributed File System (HDFS), Apache Cassandra, and Amazon S3, among others.
- **Beautiful Soup:** Beautiful Soup is a Python library that is used for web scraping purposes to extract data from HTML and XML documents. It allows developers to parse the HTML and XML documents and extract the relevant information from them. With Beautiful Soup, developers can navigate the parse tree and search for tags, attributes, and contents of an HTML or XML document. It also provides a simple and Pythonic way to scrape web pages and extract the desired data. It is widely used in data science, web development, and machine learning projects. Beautiful Soup is designed to handle poorly-formed HTML and XML

documents and can be used to clean and normalize the data before further processing. It is an open-source library and can be easily installed using pip, the Python package installer.

- **Streamlit:** Streamlit is an open-source Python library that is used for building and deploying data science web applications. It enables data scientists and developers to create interactive and customizable web applications using Python scripts. Streamlit provides a simple and intuitive interface for creating data visualization, machine learning, and natural language processing applications. Developers can create web applications with minimal coding effort, as it comes with a wide range of pre-built components and widgets. It allows developers to write simple Python scripts that can be used to build complex and interactive web applications. Streamlit also provides real-time feedback, allowing developers to see the changes they make to their application in real-time.

Logistic Regression:

Logistic Regression

```
[ ] # Train a logistic regression model using the word vectors as input features
    lr = LogisticRegression(featuresCol="features", labelCol="label")
```

```
[ ] paramGrid = ParamGridBuilder() \
    .addGrid(lr.regParam, [0.0, 0.1, 0.01]) \
    .build()
```

```
[ ] crossval = CrossValidator(estimator=lr,
                             estimatorParamMaps=paramGrid,
                             evaluator=evaluator,
                             numFolds=3
                             )

    cvModel_lr = crossval.fit(train)
```

```
[ ] predictions_lr = cvModel_lr.transform(test)
```

```
[ ] auc_score_lr = evaluator.evaluate(predictions_lr)
    auc_score_lr
```

```
0.9881349207752821
```

Random Forest Classifier:

```
[ ] # Train a RandomForestClassifier model using the word vectors as input features
rf = RandomForestClassifier(featuresCol='features', labelCol='label')
```

```
[ ] paramGrid = ParamGridBuilder() \
    .addGrid(rf.maxDepth, [3,5,7]) \
    .build()
```

```
[ ] crossval = CrossValidator(estimator=rf,
    estimatorParamMaps=paramGrid,
    evaluator=evaluator,
    numFolds=3
    )

# Run cross-validation, and choose the best set of parameters.
cvModel_rf = crossval.fit(train)
```

```
[ ] predictions_rf = cvModel_rf.transform(test)
```

```
▶ predictions_rf.show(5)
```

label	features	rawPrediction	probability	prediction
0	[-0.1371433560002...	[16.6684073825216...	[0.83342036912608...	0.0
0	[-0.1092239692086...	[18.4674647347054...	[0.92337323673527...	0.0
0	[-0.1083032864480...	[9.10323102066358...	[0.45516155103317...	1.0
0	[-0.1036650367439...	[16.3077315905476...	[0.81538657952738...	0.0
0	[-0.1011851922911...	[18.0840999489321...	[0.90420499744660...	0.0

only showing top 5 rows

```
[ ] auc_score_rf = evaluator.evaluate(predictions_rf)
auc_score_rf

0.9570326602268212
```

```
[ ] accuracy_rf = predictions_rf.filter(predictions_rf.label == predictions_rf.prediction).count() / float(predictions_rf.count())
print("Accuracy : ",accuracy_rf)
```

Accuracy : 0.8898187580469447

GBT Classifier (Gradient-Boosted Trees):

GBTCClassifier(Gradient-Boosted Trees)

```
[ ] from pyspark.ml.classification import GBTCClassifier
```

```
[ ] # Train a GBTCClassifier model using the word vectors as input features
gbt = GBTCClassifier(labelCol="label", featuresCol="features")
```

```
[ ] paramGrid = ParamGridBuilder() \
    .addGrid(gbt.maxIter, [5,10,20]) \
    .addGrid(gbt.maxDepth, [3,5,7]) \
    .build()
```

```
▶ crossval = CrossValidator(estimator=gbt,
    estimatorParamMaps=paramGrid,
    evaluator=evaluator,
    numFolds=3
    )

# Run cross-validation, and choose the best set of parameters.
cvModel_gbt = crossval.fit(train)
```

```
[ ] predictions_gbt = cvModel_gbt.transform(test)
```

```
[ ] auc_score_gbt = evaluator.evaluate(predictions_gbt)
auc_score_gbt

0.9760034993521245
```

```
[ ] accuracy_gbt = predictions_gbt.filter(predictions_gbt.label == predictions_gbt.prediction).count() / float(predictions_gbt.count())
print("Accuracy : ",accuracy_gbt)
```

Accuracy : 0.9129939586015648

Linear Support Vector Machine:

Linear Support Vector Machine

```
[ ] from pyspark.ml.classification import LinearSVC

lsvc = LinearSVC(maxIter=10, regParam=0.1)

# Fit the model
lsvcModel = lsvc.fit(train)
```

```
predictions_lsvc = lsvcModel.transform(test)
predictions_lsvc.show(5)
```

```
+-----+-----+-----+-----+
|label|      features|rawPrediction|prediction|
+-----+-----+-----+-----+
|  0|[-0.1371433560002...|[2.75718516643774...|    0.0|
|  0|[-0.1092239692086...|[2.08530144669273...|    0.0|
|  0|[-0.1083032864480...|[-0.1462797930916...|    1.0|
|  0|[-0.1036650367439...|[1.80823344588825...|    0.0|
|  0|[-0.1011851922911...|[1.71753529894742...|    0.0|
+-----+-----+-----+-----+
only showing top 5 rows
```

```
[ ] auc_score_gbt = evaluator.evaluate(predictions_lsvc)
auc_score_gbt
```

0.9800935685649396

```
[ ] accuracy_lsvc = predictions_lsvc.filter(predictions_lsvc.label == predictions_lsvc.prediction).count() / float(predictions_lsvc.count())
print("Accuracy : ",accuracy_lsvc)
```

Accuracy : 0.9302763196989204

Factorization Machines Classifier

Factorization machines classifier

```
[ ] from pyspark.ml.classification import FMClassifier
```

```
[ ] fm = FMClassifier(labelCol="label", featuresCol="features", stepSize=0.001)

fm_model = fm.fit(train)
```

```
predictions_fm = fm_model.transform(test)
predictions_fm.show(5)
```

```
+-----+-----+-----+-----+
|label|      features|rawPrediction|probability|prediction|
+-----+-----+-----+-----+
|  0|[-0.1371433560002...|[0.36990052992999...|[0.59143494273837...|    0.0|
|  0|[-0.1092239692086...|[0.40592795530304...|[0.60011107818356...|    0.0|
|  0|[-0.1083032864480...|[-0.6221855522889...|[0.34928454501701...|    1.0|
|  0|[-0.1036650367439...|[0.04486402654030...|[0.51121412573335...|    0.0|
|  0|[-0.1011851922911...|[0.36701528862993...|[0.59073757054558...|    0.0|
+-----+-----+-----+-----+
only showing top 5 rows
```

```
[ ] auc_score_fm = evaluator.evaluate(predictions_fm)
auc_score_fm
```

0.9081041108348261

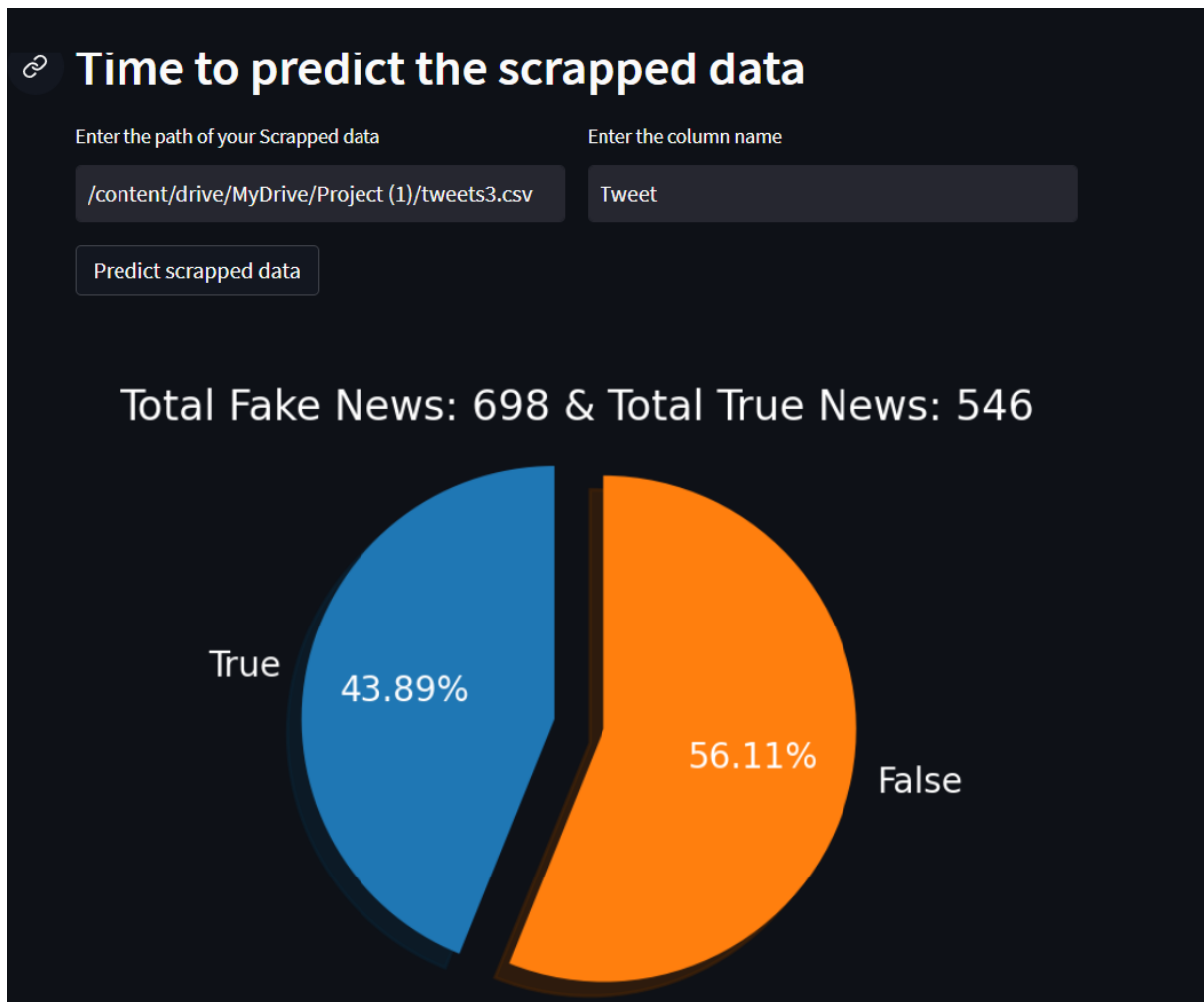
```
[ ] accuracy_fm = predictions_fm.filter(predictions_fm.label == predictions_fm.prediction).count() / float(predictions_fm.count())
print("Accuracy : ",accuracy_fm)
```

Accuracy : 0.7782014459740517


Chapter 5

Results

Scrapped data:



True News –




Time to predict the news

Enter the news article

A dozen politically active pastors came here for a private dinner Friday night to hear a conversion story u

Predict news article



Prediction result


False News –

Time to predict the news

Enter the news article

All we can say on this one is it s about time someone sued the Southern Poverty Law Center!On Tuesday

Predict news article



Prediction result

Chapter 6

Conclusion

6.1 Conclusion

- The project presents a valuable contribution to the ongoing fight against the proliferation of fake news articles. By leveraging PySpark MLlib and NLP, along with Word2Vec as the core feature extraction technique, we were able to build a system that can automatically detect and classify fake news articles with high accuracy.
- The system is trained on a diverse dataset collected from reputable sources and is deployed using Streamlit, a user-friendly web application framework that provides an intuitive user interface.
- The use of PySpark MLlib and NLP allows for the processing of large datasets, making the system scalable and efficient. The use of Word2Vec as the core feature extraction technique ensures that the system can accurately capture the semantic relationships between words in the text data. This, in turn, allows the system to classify news articles with high accuracy.
- The system can help prevent the spread of misinformation and promote more informed decision-making by identifying and flagging fake news articles. By making use of PySpark MLlib, NLP, and Word2Vec, we have built a powerful and reliable tool that can be used to combat the growing problem of fake news in our society.

6.2 Future Enhancement –

- Incorporate additional feature extraction techniques: While Word2Vec is a powerful technique for feature extraction, there are other techniques that can be used in combination with it to improve performance. For example, TF-IDF can be used to weight the importance of words in the text data, while Doc2Vec can be used to capture the context of words in the text data.
- Expand the dataset: While the dataset used in this project is diverse and comprehensive, it may still be limited in scope. Incorporating additional sources of news articles, or even incorporating non-news text data (such as social media posts or blog articles) could improve the performance of the system.
- Integrate more advanced machine learning algorithms: PySpark MLlib provides a wide range of machine learning algorithms, and experimenting with different algorithms could improve the accuracy and efficiency of the system.
- Incorporate user feedback: As users interact with the system and submit news articles for classification, the system could be improved by incorporating user feedback into the model. For example, users could be asked to rate the accuracy of the classification, which could be used to retrain the model and improve its performance.
- Implement real-time monitoring: To stay up-to-date with the latest news and ensure the system is always accurate and reliable, real-time monitoring could be implemented. This could involve monitoring news feeds or social media platforms for new articles, and automatically classifying them using the system.

Chapter 7

References

- [1] S. F. Pane, R. Prastya, A. G. Putrada, N. Alamsyah and M. N. Fauzan, "Reevaluating Synthesizing Sentiment Analysis on COVID-19 Fake News Detection using Spark Dataframe," 2022 International Conference on Information Technology Systems and Innovation (ICITSI), Bandung, Indonesia, 2022, pp. 269-274, doi: 10.1109/ICITSI56531.2022.9970773.
- [2] Tshegofatso Thate, Marié J. Hattingh and Vukosi. Marivate, Using Natural Language Processing to Detect Fake News. DOI: 10.5281/zenodo.5203129
- [3] A. Jain, A. Shakya, H. Khatter and A. K. Gupta, "A smart System for Fake News Detection Using Machine Learning," 2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), Ghaziabad, India, 2019, pp. 1-4, doi: 10.1109/ICICT46931.2019.8977659.
- [4] Abdullah-All-Tanvir, E. M. Mahir, S. Akhter and M. R. Huq, "Detecting Fake News using Machine Learning and Deep Learning Algorithms," 2019 7th International Conference on Smart Computing & Communications (ICSCC), Sarawak, Malaysia, 2019, pp. 1-5, doi: 10.1109/ICSCC.2019.8843612.
- [5] K. Poddar, G. B. Amali D. and K. S. Umadevi, "Comparison of Various Machine Learning Models for Accurate Detection of Fake News," 2019 Innovations in Power and Advanced Computing Technologies (i-PACT), Vellore, India, 2019, pp. 1-5, doi: 10.1109/i-PACT44901.2019.8960044.