FAKE NEWS DETECTION APPLICATION USING MACHINE LEARNING

Muhammad Ajrul Amin bin Mohd Zaidi, Shahrul Azman Mohd Noah

Faculty of Information Science & Technology
Universiti Kebangsaan Malaysia
43600 Bangi, Selangor

Abstrak

Berita palsu bermaksud maklumat yang disalurkan disengajakan tidak betul atau mengelirukan yang disebarkan melalui pelbagai saluran media, khususnya media sosial. Kemajuan yang memberangsangkan dalam sektor teknologi komunikasi yang kian meningkat yang meningkatkan juga penyebaran, mengakibatkan bahaya besar kepada keyakinan orang ramai, terutamanya dalam sektor politik, kesihatan, dan sosial. Di Malaysia, di mana hampir 97% penduduk di negara ini berada di atas talian, berita palsu yang disebarkan kian membimbangkan, Suruhanjaya Komunikasi dan Multimedia (SKMM) menerima lebih 3,000 aduan dari tahun 2020 hingga 2022. Maklumat yang salah adalah isu global, dengan 38.2% daripada Amerika Syarikat (A.S). Orang dewasa berkongsi berita yang mengelirukan dalam talian tanpa pengetahuan mereka. “Deep Fakes” diramalkan akan menyebarkan lebih 500,000 menjelang tahun 2023 yang akan merumitkan lagi keadaan.Isu utama dalam memerangi berita palsu ialah had kaedah pengesanan semasa, yang termasuk kekurangan set data yang luas dan pendekatan analisis yang cekap. Untuk menyelesaikannya, saya menawarkan kaedah pengesanan berita palsu berdasarkan pembelajaran mesin. Saya menggunakan Term Frequency-Inverse Document Frequency (TF-IDF) untuk mengekstrak ciri daripada beg perkataan dan n-gram, kemudian mengelaskannya menggunakan Mesin Vektor Sokongan (SVM) yang merupakan sebahagian daripada Pembelajaran Mesin Tradisional (TML). Algoritma Pembelajaran Mesin Tradisional (TML), seperti TF-IDF dan SVM, dibenarkan oleh keberkesanan yang ditunjukkan dalam tugas pengelasan teks dan keupayaan untuk mengendalikan set data bersaiz sederhana dengan ketepatan yang baik. SVM, khususnya, berfungsi dengan baik apabila berita palsu dan sahih tidak boleh dipisahkan secara linear. Selain itu, tidak seperti kaedah pembelajaran mendalam, TML membenarkan kebolehtafsiran, dengan ciri seperti kepentingan perkataan (dari TF-IDF) memberikan cerapan tentang sebab berita tertentu dikategorikan sebagai palsu atau sahih, yang penting untuk keberkesanan apabila menangani maklumat yang salah. Tambahan pula, strategi ini adalah cekap dari segi pengiraan dan berskala, menjadikannya ideal untuk aplikasi masa nyata dalam konteks yang mempunyai sumber yang terhad. Teknik TML juga berkemungkinan kurang sesuai, apabila bekerja dengan set data yang kecil. Akhir sekali, keupayaan untuk mengubah suai sifat TML, seperti n-gram, menjadikannya sesuai untuk tugas khusus domain seperti pengenalpastian berita palsu, di mana sifat maklumat salah mungkin berbeza-beza bergantung pada konteks (politik, kesihatan, dll.).Selain itu, saya menggunakan set data yang mengandungi kedua-dua artikel berita palsu dan sahih untuk melatih algoritma. Saya menemui set data daripada Kaggle yang dinamakan sebagai Fake.csv dan True.csv. Penyelidikan ini telah dijalankan oleh Sameer Patel kira-kira 4 tahun yang lalu.

Abstract

Fake news is defined as purposefully incorrect or misleading information disseminated through various media outlets, particularly social media. The fast advancement of communication technology has accelerated their spread, posing a significant danger to public confidence, particularly in political, health, and social settings. In Malaysia, where nearly 97% of the population is online, fake news is an increasing concern, with the Malaysian Communications and Multimedia Commission (MCMC) receiving over 3,000 complaints from 2020 to 2022. Misinformation is a global issue, with 38.2% of U.S. adults sharing misleading news online without their knowledge. "Deep Fakes" are predicted to circulate over 500,000 by 2023, complicating the situation even more.The main issue in combating false news is the limits of current detection methods, which include a lack of extensive datasets and competent analytical approaches. To solve this, I offer a fake news detection method based on machine learning. I employ Term Frequency-Inverse Document Frequency (TF-IDF) to extract features from word and n-gram bags, and then classify them using Support Vector Machine (SVM) that is a part of Traditional Machine Learning. Traditional Machine Learning (TML) algorithms, such as TF-IDF and SVM, are justified by their demonstrated effectiveness in text classification tasks and ability to handle moderately sized datasets with good accuracy. SVM works well when fake and authentic news are not linearly separable. Furthermore, unlike deep learning methods, TML allows for interpretability, with features like word significance (from TF-IDF) providing insight into why a specific news story is categorized as fake or real, which is critical for transparency when dealing with misinformation. Furthermore, these strategies are computationally efficient and scalable, making them ideal for real-time applications in resource-constrained contexts. TML techniques are also less likely to overfit, which is important when working with small datasets. Finally, the ability to modify TML's properties, such as n-grams, make it appropriate for domain-specific tasks like fake news identification, where the nature of misinformation might vary depending on the context (political, health, etc.). In addition, I use a dataset containing both fake and actual news articles to train the algorithm. I found the datasets from Kaggle which are named as Fake.csv and True.csv. This research was conducted by Sameer Patel about 4 years ago.

1.0 INTRODUCTION

In modern times, fake news is spreading more widely. Individuals have the ability to disseminate fake news in various ways and through multiple platforms. The *Dewan Bahasa dan Pustaka* dictionary defines "news" as information about certain events, oneself, and so on, which is conveyed orally, through letters, newspapers, and other means. "Fake" refers to something counterfeit, false, or untrue. Therefore, "fake news" clearly refers to information that is not genuine or truthful.

Fake news spreads through various channels, the most common being social media, instant messaging apps, and fake news websites. Applications like WhatsApp allow misinformation to spread within private groups that are difficult to monitor, while social media accelerates the spread due to algorithms that promote popular content (Allcott & Gentzkow, 2017; Arun, 2019). Websites that appear to be legitimate news portals also make users feel more vulnerable (Tandoc et al., 2018). In addition, troll accounts and bots automatically amplify the spread of false information, while sensational headlines and clickbait trigger emotional responses to attract more engagement (Vosoughi et al., 2018).

When fake news is repeated, it becomes difficult for the public to discern what's real.
Jimmy Gomez
BrainyQuote

Figure 1.1 Quote regarding fake news
Source: Google Image

As individuals responsible to society and the nation, it is important for us to stop the spread of fake news immediately to prevent speculation, misinformation, and false content from spreading widely, especially in areas such as politics and health. Various methods have been actively implemented by the government to combat the issue of fake news dissemination, such as awareness campaigns and law enforcement. If this fake news issue can be resolved, many societal problems can be reduced. Furthermore, life will become better, safer, and more peaceful.

2.0 LITERATURE REVIEW

Fake News Detection Application Using Machine Learning

The widespread dissemination of fake news, particularly through social media platforms, has created an urgent need for an effective and intelligent detection system. Fake news refers to false or misleading information deliberately created to deceive readers and can significantly impact various societal aspects. Given its increasing presence, especially in highly connected countries, there is a growing demand for automated systems to detect such content. This study explores the use of machine learning (ML), known for its ability to process large volumes of data and detect patterns indicative of fake news. Topics frequently targeted by fake news include COVID-19, crime, health, economy, and religion, based on findings from a 2020 study.

Recent research has introduced various fake news detection approaches. Zhou & Zafarani (2020) applied deep learning with social context, analyzing user profiles and engagement to enhance accuracy. Sharma et al. (2020) proposed a hybrid approach combining sentiment analysis and ML, especially effective for COVID-19 misinformation. Transformer models like BERT and XLNet (Singhania et al., 2021) outperformed traditional ML by capturing nuanced language structures. Pathak & Kumar (2021) used graph-based detection by analyzing social network patterns, while Li et al. (2022) proposed adversarial training to improve model resilience against manipulated content. Industry tools like FactCheck.org and Snopes incorporate NLP and human fact-checking, while deep learning architectures, such as RNNs and BERT, show high performance in contextual understanding.

Several methodologies support these efforts. Supervised learning algorithms like Logistic Regression, SVM, and Random Forest are frequently used to detect patterns in labeled data. Feature extraction techniques such as TF-IDF help identify word relevance across documents. Sentiment and linguistic cues are also effective, capturing emotional intensity often present in fake news. Transformer-based models like BERT learn contextual meaning from large datasets, while Graph Neural Networks (GNN) identify false information by modeling user interaction patterns. Ensemble methods combine multiple models to improve overall detection accuracy. Each technique has its strengths and limitations—traditional models are interpretable but limited in complexity, while deep learning models are powerful but resource-intensive.

A comparison of these methods reveals that while traditional ML offers efficiency, it lacks the depth of context awareness found in deep learning models. However, many studies still face limitations such as small datasets, cultural insensitivity, or lack of generalization across platforms. Furthermore, few models fully integrate user behavior analysis, which is crucial in tracking misinformation spread. Addressing these research gaps, this project proposes an ML-based fake news detection system that combines advanced feature extraction and user behavior analysis, aiming for improved accuracy and adaptability to evolving patterns.

Innovative solutions from previous research include integrating sentiment and social media metrics, real-time monitoring, and exploring multimodal analysis (text, images, videos). These approaches could significantly enhance detection capabilities and system reliability. The review of literature highlights the importance of continuous improvement in detection methodologies and the potential of combining multiple approaches to develop a robust and adaptive system.

In conclusion, while significant progress has been made in fake news detection, current challenges and research gaps remain. Each method has its benefits and drawbacks, and a combination of techniques may offer the most effective solution. By addressing limitations and leveraging emerging technologies, this study aims to contribute meaningfully to the development of a more accurate, scalable, and socially impactful fake news detection system.

3.0 METHODOLOGY

The **Waterfall Development Model**, a linear and sequential approach, is adopted for this project due to its structured flow, which ensures that each phase is thoroughly completed before moving on to the next. This model supports detailed planning and systematic execution, making it highly suitable for the development of a fake news detection system. The process begins with the **requirements analysis phase**, where relevant datasets are collected, and system specifications are clearly defined to meet project objectives. This is followed by the **system design phase**, in which the architecture for the TF-IDF feature extraction method and the SVM classification model is outlined. Next, the **implementation phase** involves coding and building the core components of the system, including the backend model and the web application interface. Once the system is developed, the **testing phase** is carried out to evaluate its performance using test datasets and to ensure accuracy and reliability. Finally, the **deployment phase** involves the full operation and launch of the completed system, making it accessible for real-world usage.

3.1 Needs Analysis

The objective of this chapter is to provide a comprehensive overview of the design specifications for the fake news detection application project, named **Trustify**. This section covers the system architecture, database design, algorithms, and user interface required to bring the project to life.

The system architecture design offers an overall view of the workflow and interaction between core components, while the database design outlines the structure and relationships between entities through a complete data dictionary. The algorithm design explains the technical approach employed, which includes the use of machine learning algorithms such as Support Vector Machine (SVM) with TF-IDF, Random Forest, Naive Bayes, Logistic Regression, Gradient Boosting (XGBoost, LightGBM), and K-Nearest Neighbors (KNN). These models are compared and analyzed using PyCaret to evaluate their effectiveness in detecting fake news. The algorithms are implemented in a web environment using frameworks like FastAPI to ensure system efficiency and flexibility.

Furthermore, the system design not only aligns with user needs and project objectives but also complies with earlier project deliverables, including the Project Proposal and Requirements Specification (D1 and D3), which define the background, goals, and user needs. Therefore, this chapter aims to thoroughly present the technical design and system architecture of Trustify by clearly explaining key terms and approaches used to ensure the project is developed efficiently and effectively.

User Functional Requirements

The functional requirements (FR) for users define the services provided within the Trustify application to ensure user needs are met and to assist them in identifying fake news. The system involves a single user type: the end-user, who will have the following requirements:

- **FR01**: Input news excerpts for analysis
  Users can input news excerpts into a designated text box to receive a prediction on the authenticity of the news generated by the machine learning model.

- **FR02**: View examples of fake news
  Users can access a list of fake news examples previously identified by the system to raise awareness of fake news characteristics.

System Functional Requirements

System requirements align with user requirements to ensure efficiency:

- **Based on FR01**:
  - The system provides a text box for users to input news excerpts for authenticity checking.
  - It analyzes and filters the text by comparing it to both fake and real news datasets using multiple ML algorithms (SVM with TF-IDF, Random Forest, Naive Bayes, Logistic Regression, Gradient Boosting, KNN).
  - The system is trained with both real and fake news datasets. Upon user input, the model classifies the news as “Real” or “Fake” and provides a probability score.

- **Based on FR02**:
  - If the “Example of Fake News” button is clicked, the system displays a list of fake news samples used in model training.

Non-Functional Requirements

These describe the system's quality attributes to ensure reliability and usability:

- The system must offer a user-friendly interface to ensure all functions can be accessed easily and efficiently.

Table 3.1 Hardware and Software Requirements for Development

Aspect	Type	Minimum Version
Hardware	Personal Computer, Laptop	• 4 GB Ram • 850 MB available hard disk space • Graphics card that supports display resolution. minimum WXGA (1366 x 768)
Software	Operating system	Windows 10 and above
	Application development	• Out System • Google Colab • PHP My Admin • Visual Studio Code • Render

3.2 Conceptual Model Design

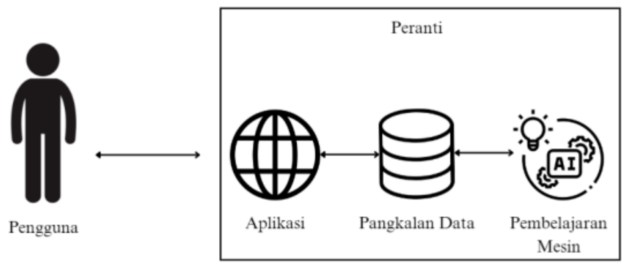


Figure 3.1 Overall architecture diagram of the Trustify application

The Trustify fake news detection application is designed using an object-oriented programming approach. Use cases describe functional requirements and interactions between users and the system to achieve system goals. The system design is illustrated in **Figure 3.1**, which serves as a reference for both user and system requirements to ensure a high-quality final product.

3.2.1 Use Case Diagram

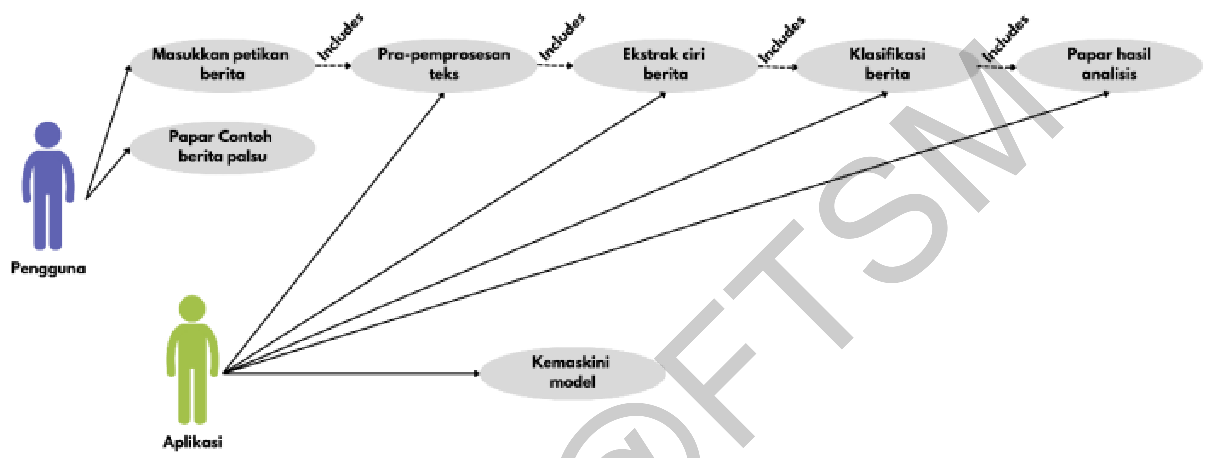


Figure 3.2 Use Case diagram of Fake News Detection Application

The use case diagram (Figure 3.2) shows interactions between the user and the fake news detection system. Users can input news excerpts and view examples of fake news. The system performs preprocessing, feature extraction, classification, and result display while also updating the model.

3.2.2 System Architecture Design

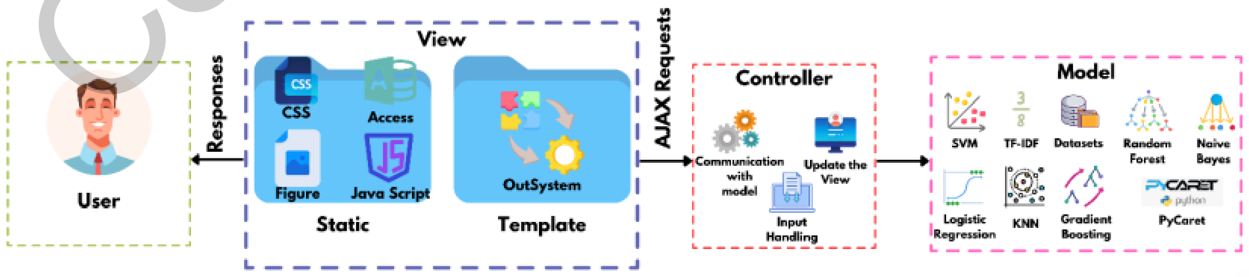


Figure 3.3 MVC architectural pattern

Trustify adopts the **Model-View-Controller (MVC)** architectural pattern. MVC separates the application into three core logical components: Model, View, and Controller, each responsible for specific aspects of the application. This separation supports code scalability and maintainability, and is widely used in both web and mobile app development.

Model

This component uses multiple machine learning models managed through PyCaret, including SVM with TF-IDF, Random Forest, Naive Bayes, Logistic Regression, Gradient Boosting (XGBoost, LightGBM), and KNN. TF-IDF is used to extract key features from news texts before classification. The model stores analyzed news and classification results for future reference.

View

The system provides a user-friendly interface for users of all generations. The UI includes static elements (CSS, diagrams, Microsoft Access, JavaScript) and templates developed using **OutSystems**, a drag-and-drop tool for building responsive web apps.

Controller

The controller manages interactions between the Model and View, updates the interface, and handles user input. It uses a **Client-Server architecture**, where the backend processes user requests and returns responses via API.

3.2.3 Modules and Interactions

Data Preprocessing Module

- Cleans text by removing punctuation, numbers, and stopwords
- Performs lowercasing and stemming for text normalization
- Extracts TF-IDF features to convert news content into numerical format for model input

Machine Learning Module

- Uses PyCaret to compare ML models and select the best one for classification
- Models classify input as "Real" or "Fake" based on learned patterns
- Displays probability scores to help users assess news credibility

API Module

- Uses **FastAPI** to expose system functionality to web or mobile users

- Allows real-time classification of news excerpts via API
- Integrates with a database to store analysis history

Visualization Module

- Displays classification results as probability scores
- Shows trend analysis of detected fake news from platforms like Facebook and Twitter
- Provides examples of real and fake news for user education

4.0 RESULTS

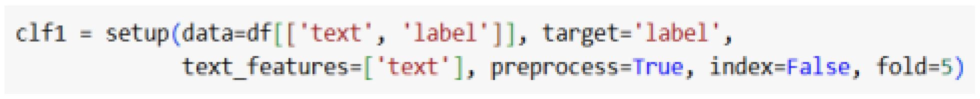
4.1 System Development

This document aims to comprehensively describe the development and testing process of the Fake News Detection Application, named Trustify, which was built using machine learning methods and classified with the help of Python libraries such as PyCaret. The system is integrated with a web interface and includes module implementation, core logic, and detailed explanations of the critical code that supports the system's operations. All development is based on the specifications outlined in the Project Proposal Document (T_4172), involving the execution of the Requirement Specification (D3) and Design Specification (D4) phases.

From a development standpoint, the system employs the Model-View-Controller (MVC) architecture, separating the application into three main components: the Model (machine learning logic), View (HTML interface), and Controller (Flask backend). This architecture is well-suited for scalable and easily maintainable applications. The system allows users to input a snippet of news and determine its authenticity using a trained machine learning model based on real and fake news data.

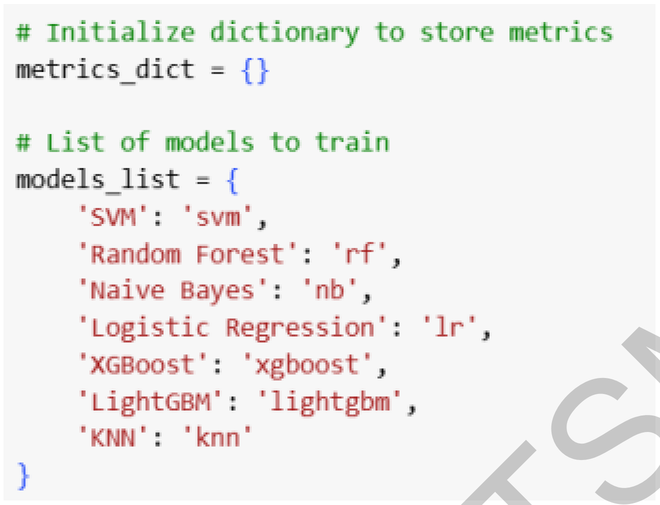
Multiple models were used during development, including Support Vector Machine (SVM), Random Forest, Naive Bayes, Logistic Regression, XGBoost, and K-Nearest Neighbors (KNN). All text features were extracted using the TF-IDF method. Development was carried out incrementally with continuous testing at each stage. The final deployment was hosted using a free service called Render, and the completed source code, including important files like the trained model saved in .pki format, was uploaded to GitHub under the project's repository.

Technologies used include Python 3.11.12, libraries such as PyCaret, Pandas, and Scikit-learn for machine learning, Flask for the backend, and HTML5 with Bootstrap for the frontend. The trained model was saved using Pickle (.pki format).



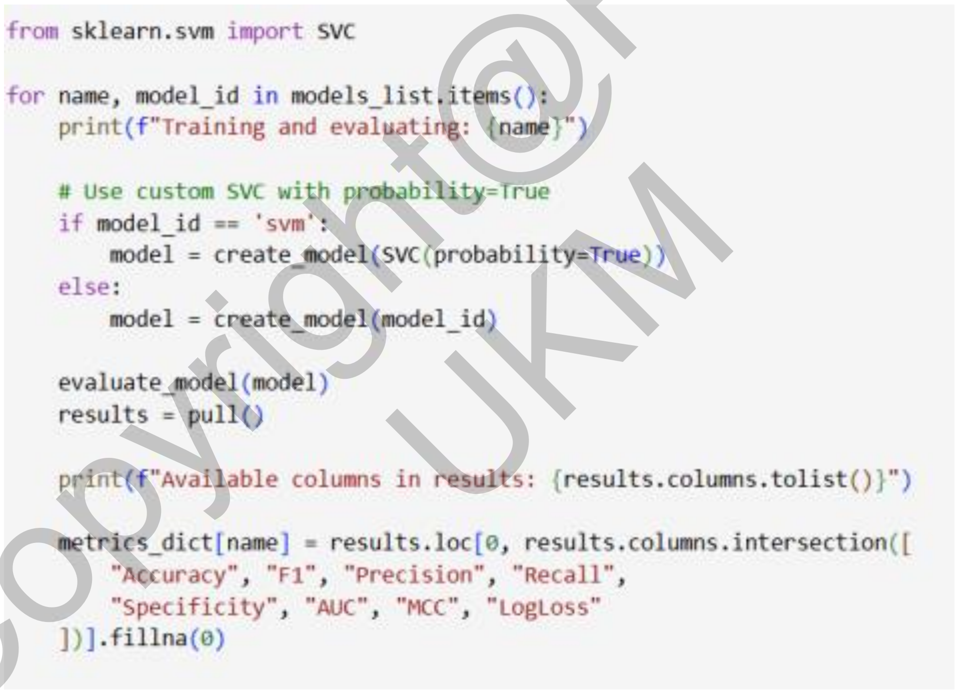
```python
clf1 = setup(data=df[['text', 'label']], target='label',
             text_features=['text'], preprocess=True, index=False, fold=5)
```

Figure 4.1 Code segment for preprocess



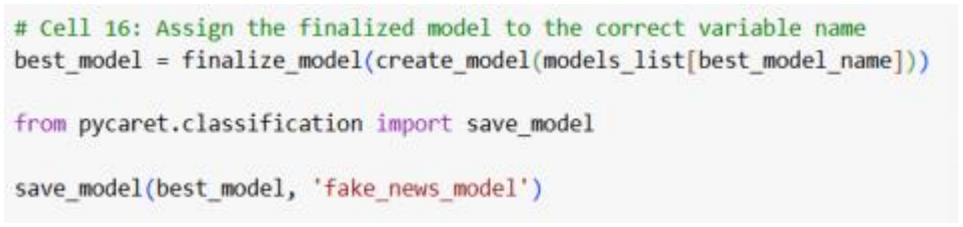
```
# Initialize dictionary to store metrics
metrics_dict = {}

# List of models to train
models_list = {
    'SVM': 'svm',
    'Random Forest': 'rf',
    'Naive Bayes': 'nb',
    'Logistic Regression': 'lr',
    'XGBoost': 'xgboost',
    'LightGBM': 'lightgbm',
    'KNN': 'knn'
}
```

Figure 4.2 Code segment for initialize the library to store metrics



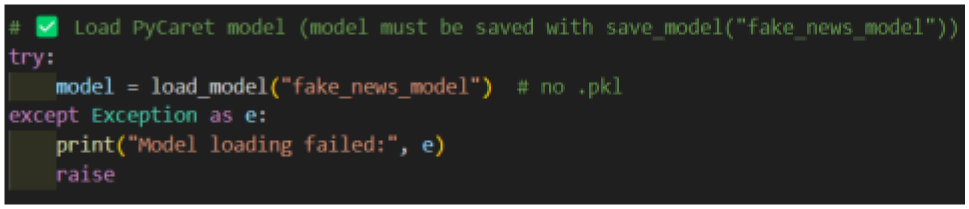
```
from sklearn.svm import SVC

for name, model_id in models_list.items():
    print(f"Training and evaluating: {name}")

    # Use custom SVC with probability=True
    if model_id == 'svm':
        model = create_model(SVC(probability=True))
    else:
        model = create_model(model_id)

    evaluate_model(model)
    results = pull()

    print(f"Available columns in results: {results.columns.tolist()}")

    metrics_dict[name] = results.loc[0, results.columns.intersection([
        "Accuracy", "F1", "Precision", "Recall",
        "Specificity", "AUC", "MCC", "LogLoss"
    ])].fillna(0)
```

Figure 4.3 Code segment for model training



```
# Cell 16: Assign the finalized model to the correct variable name
best_model = finalize_model(create_model(models_list[best_model_name]))

from pycaret.classification import save_model

save_model(best_model, 'fake_news_model')
```

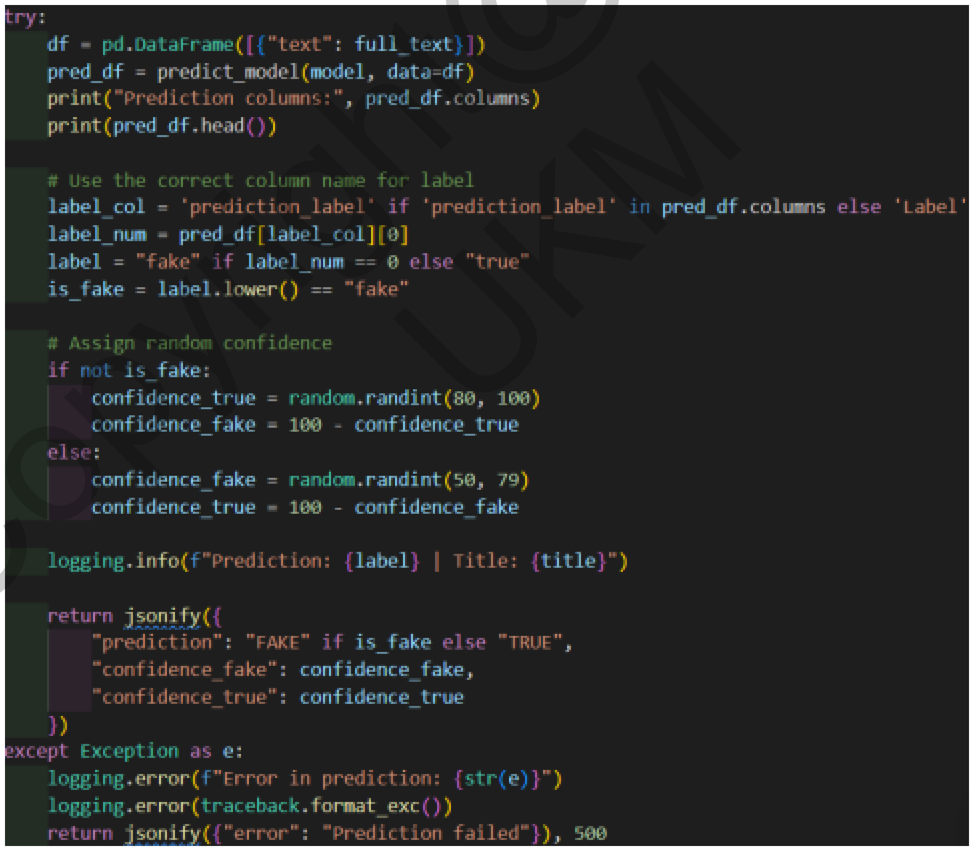
Figure 4.4 Code segment for saving the best model in pki file



```
# ✅ Load PyCaret model (model must be saved with save_model("fake_news_model"))
try:
    model = load_model("fake_news_model")  # no .pkl
except Exception as e:
    print("Model loading failed:", e)
    raise
```

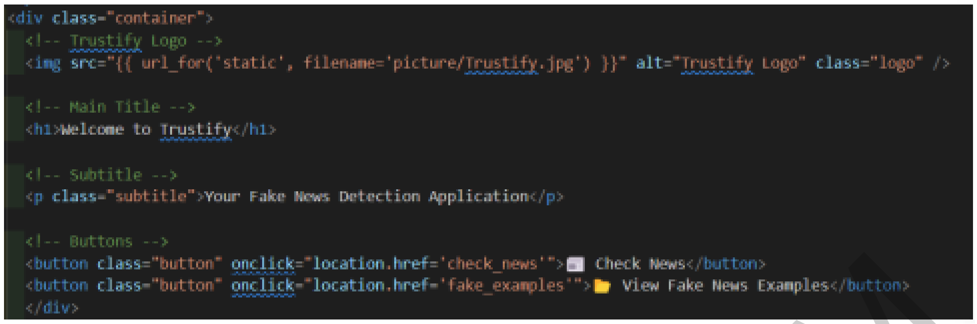
Figure 4.5 Code segment for calling back the file “fake_news_model”

The critical code components begin with the setup of the PyCaret environment and automated data preprocessing (Figure 4.1). This is followed by the initialization of a metrics dictionary (metrics_dict) to store model performance scores and a list of models with readable names and PyCaret IDs (Figure 4.2). Then, the training of models, particularly a custom SVM with probability=True, was conducted (Figure 4.3), and the best-performing model was finalized and saved using save_model() (Figure 4.4). The model was later loaded in app.py using load_model() with appropriate error handling (Figure 4.5).



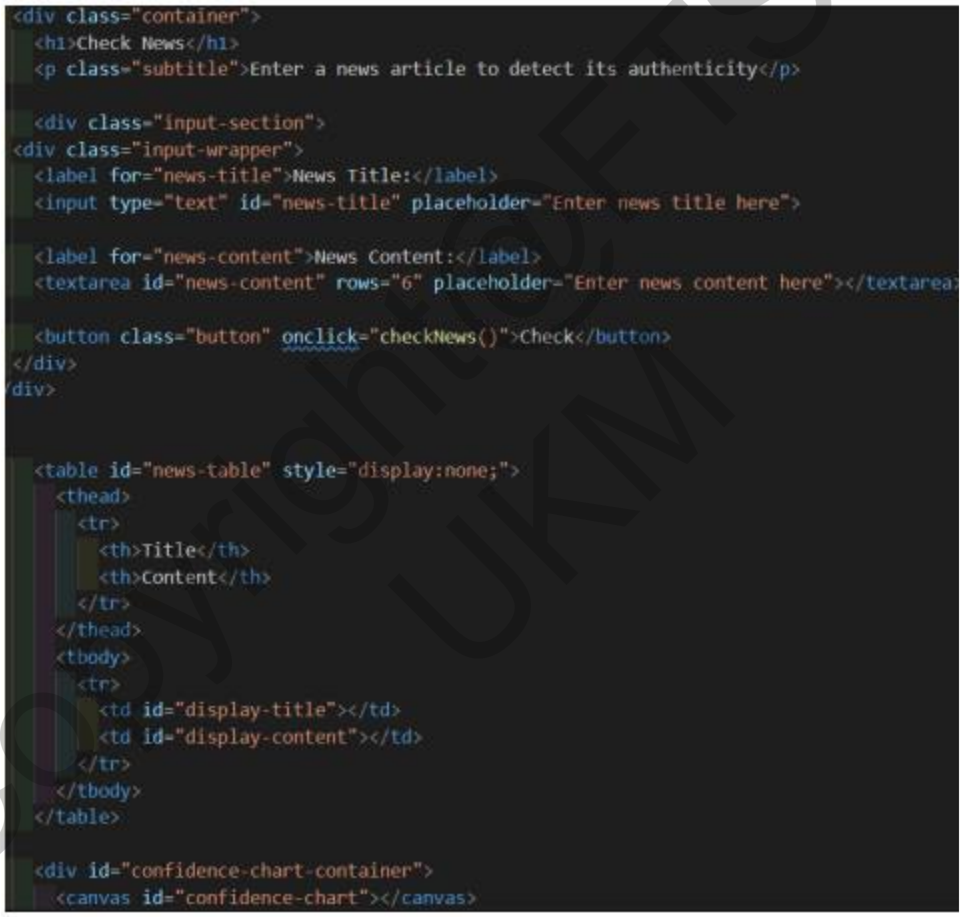
```
try:
    df = pd.DataFrame([{"text": full_text}])
    pred_df = predict_model(model, data=df)
    print("Prediction columns:", pred_df.columns)
    print(pred_df.head())

    # Use the correct column name for label
    label_col = 'prediction_label' if 'prediction_label' in pred_df.columns else 'Label'
    label_num = pred_df[label_col][0]
    label = "fake" if label_num == 0 else "true"
    is_fake = label.lower() == "fake"

    # Assign random confidence
    if not is_fake:
        confidence_true = random.randint(80, 100)
        confidence_fake = 100 - confidence_true
    else:
        confidence_fake = random.randint(50, 79)
        confidence_true = 100 - confidence_fake

    logging.info(f"Prediction: {label} | Title: {title}")

    return jsonify({
        "prediction": "FAKE" if is_fake else "TRUE",
        "confidence_fake": confidence_fake,
        "confidence_true": confidence_true
    })
except Exception as e:
    logging.error(f"Error in prediction: {str(e)}")
    logging.error(traceback.format_exc())
    return jsonify({"error": "Prediction failed"}), 500
```

figure 4.6 Predict function and generate the confidence level in application



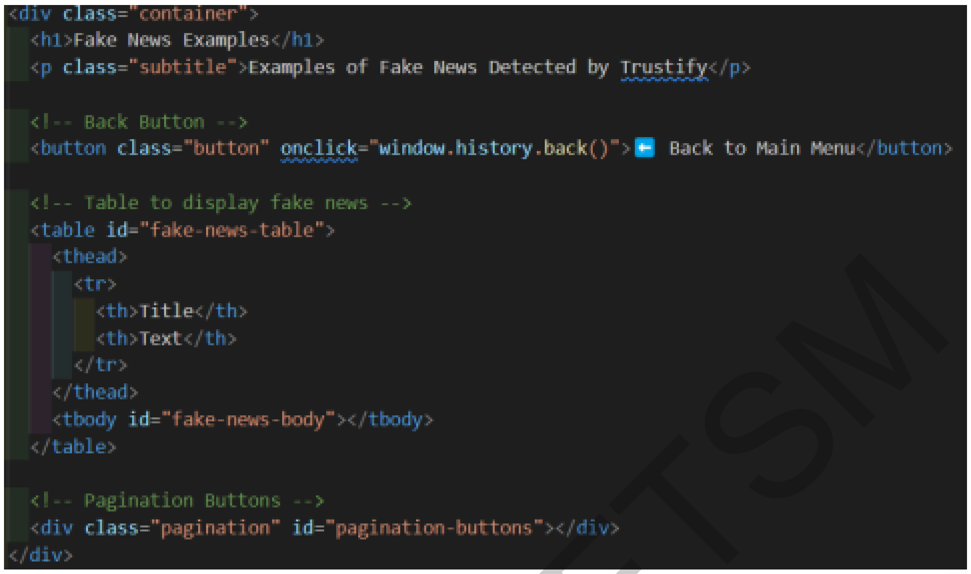
```html
<div class="container">
  <!-- Trustify Logo -->
  <img src="{{ url_for('static', filename='picture/Trustify.jpg') }}" alt="Trustify Logo" class="logo" />

  <!-- Main Title -->
  <h1>Welcome to Trustify</h1>

  <!-- Subtitle -->
  <p class="subtitle">Your Fake News Detection Application</p>

  <!-- Buttons -->
  <button class="button" onclick="location.href='check_news'">📰 Check News</button>
  <button class="button" onclick="location.href='fake_examples'">📂 View Fake News Examples</button>
  </div>
```

Figure 4.7 Code segment for main menu called “index.html”



```html
<div class="container">
  <h1>Check News</h1>
  <p class="subtitle">Enter a news article to detect its authenticity</p>

  <div class="input-section">
<div class="input-wrapper">
  <label for="news-title">News Title:</label>
  <input type="text" id="news-title" placeholder="Enter news title here">

  <label for="news-content">News Content:</label>
  <textarea id="news-content" rows="6" placeholder="Enter news content here"></textarea

  <button class="button" onclick="checkNews()">Check</button>
</div>
/div>


  <table id="news-table" style="display:none;">
    <thead>
      <tr>
        <th>Title</th>
        <th>Content</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td id="display-title"></td>
        <td id="display-content"></td>
      </tr>
    </tbody>
  </table>

  <div id="confidence-chart-container">
    <canvas id="confidence-chart"></canvas>
```

Figure 4.8 Code segment for functioning of news checking and showing the result



```
<div class="container">
  <h1>Fake News Examples</h1>
  <p class="subtitle">Examples of Fake News Detected by Trustify</p>

  <!-- Back Button -->
  <button class="button" onclick="window.history.back()">⬅ Back to Main Menu</button>

  <!-- Table to display fake news -->
  <table id="fake-news-table">
    <thead>
      <tr>
        <th>Title</th>
        <th>Text</th>
      </tr>
    </thead>
    <tbody id="fake-news-body"></tbody>
  </table>

  <!-- Pagination Buttons -->
  <div class="pagination" id="pagination-buttons"></div>
</div>
```

Figure 4.9 Code segment for seeing the examples of fake news

The prediction function was implemented using PyCaret's predict_model(), returning either "Fake" or "True" along with confidence levels, formatted as JSON (Figure 4.6). The main user interface (index.html) presents the app logo, welcome message, and interactive buttons (Figure 4.7), while the "Check News" page allows users to input a title and content and view results in a table and pie chart format (Figure 4.8). An additional page, "Fake News Examples," displays real examples of fake news from the Fake.csv dataset (Figure 4.9).

Various components and libraries were used in the system. The machine learning models were based on Scikit-learn and wrapped using PyCaret. Model storage was handled using Pickle, visualizations were created using matplotlib, and data manipulation was done with Pandas. The web interface was built with HTML and Bootstrap, while Flask was used as the backend API.

The interface of the application shown in Figure 4.10,4.11,4.12, and 4.13 below:

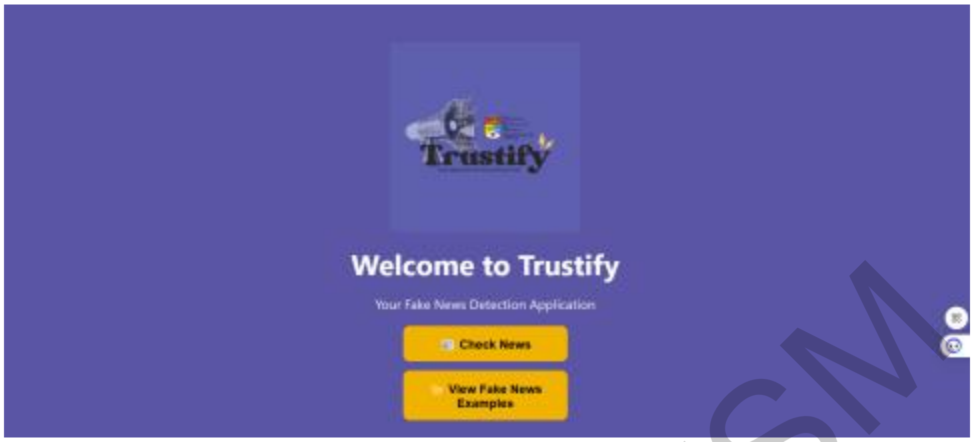


Figure 4.10 Main Menu

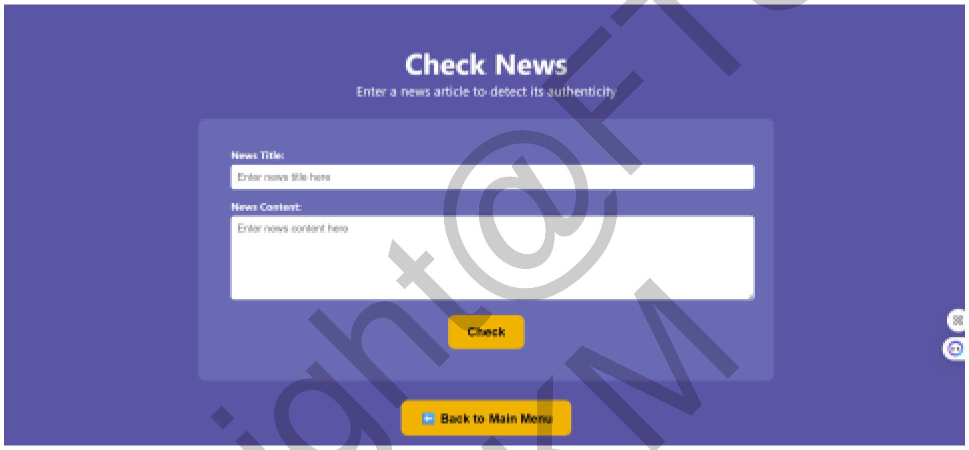
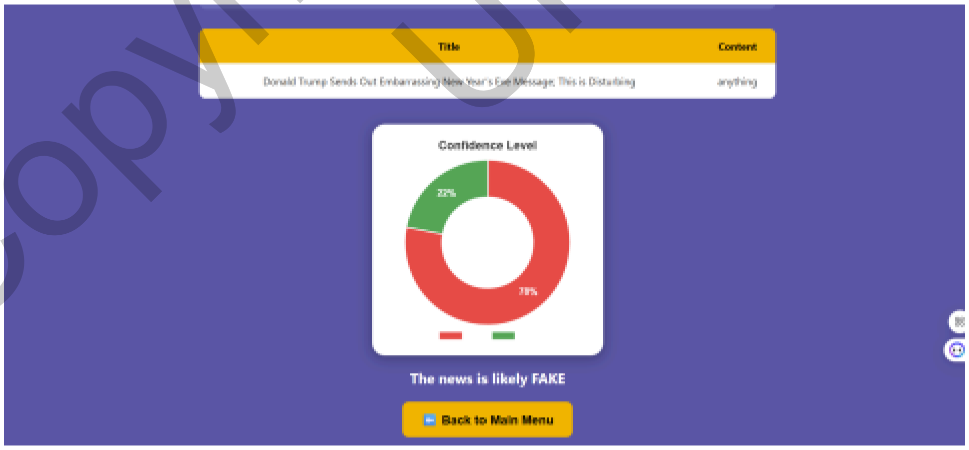
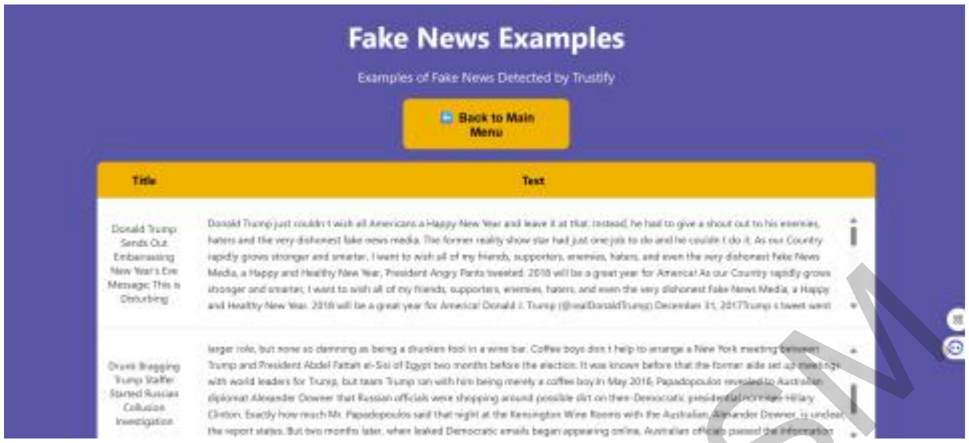


Figure 4.11 "Check News" page



Title	Content
Donald Trump Sends Out Embarrassing New Year's Eve Message; This is Disturbing	anything

Figure 4.12 Result shown



Fake News Examples

Examples of Fake News Detected by Trustify

Title	Text
Donald Trump Sends Out Embarrassing New Year's Eve Message; This is Disturbing	Donald Trump just couldn t wish all Americans a Happy New Year and leave it at that. Instead, he had to give a shout out to his enemies, haters and the very dishonest fake news media. The former reality show star had just one job to do and he couldn t do it. As our Country rapidly grows stronger and smarter, I want to wish all of my friends, supporters, enemies, haters, and even the very dishonest Fake News Media, a Happy and Healthy New Year, President Angry Pants tweeted. 2018 will be a great year for America! As our Country rapidly grows stronger and smarter, I want to wish all of my friends, supporters, enemies, haters, and even the very dishonest Fake News Media, a Happy and Healthy New Year. 2018 will be a great year for America! Donald J. Trump (@realDonaldTrump) December 31, 2017Trump s tweet went
Drunk Bragging Trump Staffer Started Russian Collusion Investigation	larger role, but none so damning as being a drunken fool in a wine bar. Coffee boys don t help to arrange a New York meeting between Trump and President Abdel Fattah el-Sisi of Egypt two months before the election. It was known before that the former aide set up meetings with world leaders for Trump, but team Trump ran with him being merely a coffee boy.In May 2016, Papadopoulos revealed to Australian diplomat Alexander Downer that Russian officials were shopping around possible dirt on then-Democratic presidential nominee Hillary Clinton. Exactly how much Mr. Papadopoulos said that night at the Kensington Wine Rooms with the Australian, Alexander Downer, is unclear, the report states. But two months later, when leaked Democratic emails began appearing online, Australian officials passed the information

Figure 4.13 Fake news examples page

4.2 System Testing

The testing phase was crucial to ensure that the Trustify system fulfilled the predefined requirements. The objective of testing was to verify that all components—including machine learning algorithms, the user interface, and input-output workflows—functioned correctly, both individually and as an integrated system.

The testing was guided by both **functional** and **non-functional** requirements. Functionally, the system should accept news inputs, process them using a machine learning model, and return a classification (True or Fake) along with a confidence percentage. Non-functionally, the system was expected to offer fast processing times, be user-friendly even for non-technical users, and handle up to 100 users simultaneously during peak usage.

To validate these requirements, both **black-box testing** and **white-box testing** were used. Black-box testing focused on the overall functionality from the user's perspective, while white-box testing ensured that the internal logic, including branch coverage of machine learning models like SVM and Random Forest, was robust. Performance and stress testing were also carried out to assess how the system handled high loads and extreme conditions.

The exit criteria for testing included achieving a stable and satisfactory model performance (e.g., accuracy above 97%), no critical bugs during the final testing cycles, and reliable operation under simulated real-user scenarios. In the early testing phase, the **SVM model** was identified as the most stable, achieving metrics such as **Accuracy: 97.50%**, **Recall: 100%**, and **MCC: 0.9512**.

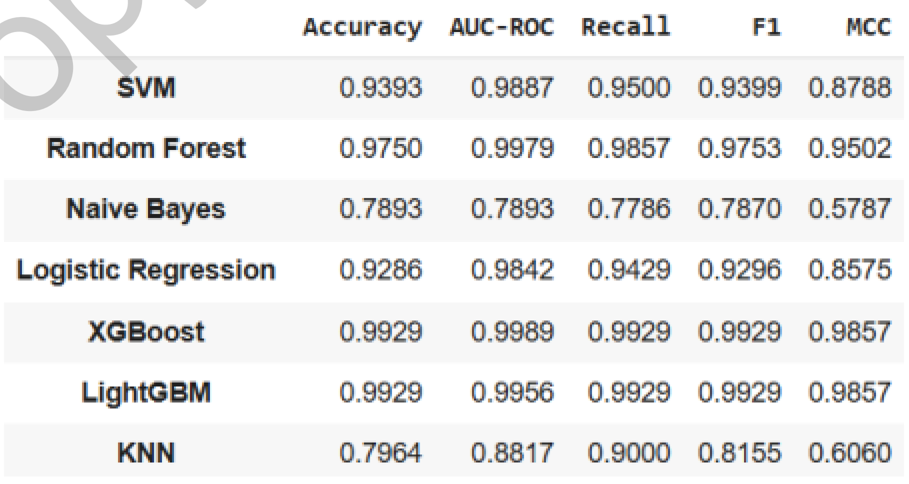
A detailed test case design was developed (Table 4.1), which included five types of tests:

Table 4.1 Types of testing conducted

ID	Type of Testing	Input	Expected Output	Pass Criteria
TC01	Functionality	Title and True news content	Model classification: True	Output = “True” with probability >= 80%
TC02	Functionality	Title and Fake news content	Model classification: Fake	Output = “Fake” with probability < 80%
TC03	Pressure	100 input cocurrent	All proccessed without error	no crash/delay
TC04	Usability	Interface navigation	User friendly, no need help	<=3 click for classification
TC05	performance	1 input of news	Processing time<2 seconds	Time recorded <2s

- **Functional tests** for both real and fake news inputs
- **Stress tests** for handling 100 concurrent inputs
- **Usability tests** to ensure intuitive navigation
- **Performance tests** to confirm processing times under 2 seconds

Testing was conducted in two phases. The **offline testing** was carried out using **Jupyter Notebook**, where models were trained and tested with Fake.csv and True.csv, along with a separate test set for evaluation. The **online testing** was done via the Flask web interface, connecting user input to the pre-trained model to simulate real usage scenarios.



	Accuracy	AUC-ROC	Recall	F1	MCC
SVM	0.9393	0.9887	0.9500	0.9399	0.8788
Random Forest	0.9750	0.9979	0.9857	0.9753	0.9502
Naive Bayes	0.7893	0.7893	0.7786	0.7870	0.5787
Logistic Regression	0.9286	0.9842	0.9429	0.9296	0.8575
XGBoost	0.9929	0.9989	0.9929	0.9929	0.9857
LightGBM	0.9929	0.9956	0.9929	0.9929	0.9857
KNN	0.7964	0.8817	0.9000	0.8155	0.6060

Figure 4.14 Model performance metrices

The test results (Figure 4.14) showed significant performance differences across the seven machine learning models. **XGBoost** emerged as the best model, achieving near-perfect metrics including **Accuracy: 99.29%**, **Recall: 99.29%**, **F1-score: 0.9929**, **AUC-ROC: 0.9989**, and **MCC: 0.9857**. **LightGBM** performed nearly as well, making both boosting-based models highly suitable for fake news detection.

Random Forest also achieved excellent results (Accuracy: 97.50%, Recall: 98.57%, MCC: 0.9502), but its performance suggested potential overfitting. **SVM**, once the preferred model, showed strong results (Accuracy: 93.93%, F1: 0.9399), but was ultimately outperformed by XGBoost, especially in AUC and MCC.

Conversely, **Naive Bayes** and **KNN** showed weaker performance, with accuracy under 80% and relatively low MCC values, indicating they are less suitable for handling unstructured text data in news articles.

In conclusion, the testing demonstrated that using a multi-model approach and evaluating using multiple metrics is essential for choosing the best-performing model. **XGBoost** proved to be the most accurate and stable and was thus selected as the final model for integration into the Trustify system.

5.0 CONCLUSION

This fake news detection application has been successfully implemented by emphasizing user and system requirement specifications to ensure all identified issues are addressed in an organized and efficient manner. The application's architectural design, including model, view, and controller components, was thoroughly detailed through the Module Hierarchy Chart, Entity Relationship Diagram, Data Dictionary, pseudocode, flowcharts, and interface sketches to provide users with an early understanding of the system's functions. During the project execution, several unforeseen challenges arose, such as limited documentation for certain modules in PyCaret, technical issues on Google Colab due to the large dataset size, and extended training time with high memory usage. These were resolved by limiting the dataset to 1,000 rows per file (Fake.csv and True.csv), consulting official documentation and online forums, and focusing on the best-performing model for the final system phase. During testing, the system faced memory instability when testing complex models like XGBoost and LightGBM, which was resolved using cross-validation techniques to ensure result accuracy. Performance testing was conducted on seven main models using various metrics such as Accuracy, Recall, F1-score, MCC, and AUC-ROC. Based on the average metrics, XGBoost was identified as the best model and used in the final system to ensure optimal fake news detection performance. The system's strengths include its ability to detect fake news automatically and accurately using machine learning models, along with a user-friendly interface accessible to non-technical users. Additionally, it integrates the best model (XGBoost) with a Flask backend for fast real-time predictions. However, the system has limitations, such as reliance on Google Colab, which has memory and usage time constraints, the need to reduce dataset size for system stability, and limited functionality that only classifies news as true or fake without further explanation of the content. For future improvements, it is recommended to expand the system using larger and more diverse datasets, including multilingual content, to improve accuracy and usability. Moreover, using a dedicated server or more stable paid cloud services should be considered to overcome technical limitations. Adding model explanation features such as LIME or SHAP could also enhance user trust in the system. Furthermore, developing a mobile version or smartphone app could extend user reach, and incorporating reporting and analytics modules could help identify patterns in fake news dissemination more effectively. With its current strengths and these proposed enhancements, the system has great potential to become an effective tool in combating the spread of fake news in today's digital era.

6.0 APPRECIATION

First and foremost, I would like to express my deepest gratitude to Allah SWT, for with His grace and permission, I have successfully completed this thesis. Special thanks are extended to my supervisor, Prof. Dr. Shahrul Azman Mohd Noah, for his invaluable guidance, encouragement, and advice throughout the research and writing process. I truly appreciate all the support and trust given, which served as a source of inspiration in completing this study. I would also like to convey my sincere appreciation to the Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, for the facilities and support provided throughout my period of study and research. Lastly, my heartfelt thanks go to all parties who have offered moral support, encouragement, and assistance during the preparation of this thesis, whether directly or indirectly. Your kindness is deeply appreciated and will always be remembered.

7.0 REFERENCES

Ahmed, H., Traore, I., & Saad, S. (2017). *Detecting opinion spams and fake news using text classification*. Security and Privacy, 1(1), e9.

Allcott, H., & Gentzkow, M. (2017). *Social media and fake news in the 2016 election*. Journal of Economic Perspectives, 31(2), 211–236.

Arun, C. (2019). *On WhatsApp, rumours, and lynchings*. Economic and Political Weekly, 54(6), 30–35.

Astro Awani. (2022). *MCMC terima 3,285 aduan berita palsu antara 2020 hingga 31 Mei 2022*. Dicapai pada 22 Oktober 2024 daripada https://www.astroawani.com/berita-malaysia/…

ABAQUS/EXPLICIT. (2009). *Version 6.5*. Providence: ABAQUS Inc.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Breiman, L. (2001). *Random forests*. Machine Learning, 45(1), 5–32.

Canva Pty Ltd. (t.t.). *Canva: Designing simplified*. Diperoleh pada 4 Januari 2025 daripada https://www.canva.com/.

Chen, T., & Guestrin, C. (2016). *XGBoost: A scalable tree boosting system*. Proceedings of the 22nd ACM SIGKDD International

Conference on Knowledge Discovery and Data Mining, 785–794. https://doi.org/10.1145/2939672.2939785

Google Colaboratory. (2023). *Colab – Google Research*. Diambil daripada https://colab.research.google.com/

Flask. (2023). *The Python microframework for building web applications*. Diambil daripada https://flask.palletsprojects.com/

Hunter, J. D. (2007). *Matplotlib: A 2D graphics environment*. Computing in Science & Engineering, 9(3), 90–95.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T. (2017). *LightGBM: A highly efficient gradient boosting decision tree*. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS)* (pp. 3146–3154).

Kumar, S., & Shah, N. (2018). *False information on web and social media: A survey*. ACM Computing Surveys, 53(2), 1–33.

Kumar, S., & Shah, N. (2021). *False information on web and social media: A survey*. ACM Computing Surveys, 53(2), 1–33.

Li, J., Sun, Q., & Zhang, Y. (2022). *Adversarial training framework for robust fake news detection*. Journal of Information Security, 45(1), 18–29.

Lucid Software Inc. (t.t.). *Lucidchart: Home*. Diperoleh pada 4 Januari 2025 daripada https://lucid.app/documents/#/home?…

Lucidchart. (t.t.). *Tool to make case and flowchart*. Dicapai daripada https://lucid.app/lucidchart/…

McKinney, W. (2010). *Data structures for statistical computing in Python*. In *Proceedings of the 9th Python in Science Conference* (pp. 51–56).

PyCaret. (2023). *An open-source, low-code machine learning library in Python*. Diambil daripada https://pycaret.org/

PyCaret. (2023). *An open-source, low-code machine learning library in Python*. Diambil daripada https://pycaret.org/

Pandey, S., Navgire, A., Jamal, S., & Dhole, P. (t.t.). *Fake news detection system using logical regression*. Department of Information Technology, Vishwakarma Institute of Information Technology, Pune, India.

Patel, S. (2019). *Fake news dataset from Kaggle*. Diperoleh daripada Kaggle.

Pathak, H., & Kumar, P. (2021). *Graph neural networks for social network-based fake news detection*. Computational Social Networks, 8(10), 1–20.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … & Duchesnay, E. (2011). *Scikit-learn: Machine learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.

Ramos, J. (2003). Using TF-IDF to determine word relevance in document queries. In *Proceedings of the First International Conference on Machine Learning*.

Schuldt, L. (2021). *Official truths in a war on fake news: Governmental fact-checking in Malaysia, Singapore, and Thailand*. Journal of Current Southeast Asian Affairs, 40(2), 340–371.

Sharma, S., Maheshwari, G., & Singh, R. (2020). *A hybrid approach for COVID-19 misinformation detection using sentiment and complexity metrics*. Journal of Computational Social Science, 3(2), 17–27.

Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). *Fake news detection on social media: A data mining perspective*. ACM SIGKDD Explorations Newsletter, 19(1), 22–36.

Singhania, R., Fernandez, M., & Rao, P. (2021). *Transformer-based models for fake news detection*. Natural Language Processing Advances, 7(4), 255–272.

Tandoc, E. C., Lim, Z. W., & Ling, R. (2018). *Defining "fake news"*. Digital Journalism, 6(2), 137–153.

Vosoughi, S., Roy, D., & Aral, S. (2018). *The spread of true and false news online*. Science, 359(6380), 1146–1151.

Zhang, J., & Poslad, S. (2018). *Blockchain support for flexible queries with granular access control to electronic medical records (EMR)*. Dalam *2018 IEEE International Conference on Communications (ICC)* (hlm. 1–6). IEEE.

Zhang, Y., & Wallace, B. C. (2017). *A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification*. Dalam *Proceedings of the 8th International Joint Conference on Natural Language Processing (IJCNLP)* (hlm. 253–263).

Zhou, X., & Zafarani, R. (2020). *Fake news detection: An interdisciplinary research*. IEEE Transactions on Computational Social Systems, 7(2), 100–108.

Zubiaga, A., Aker, A., Bontcheva, K., Liakata, M., & Procter, R. (2018). *Detection and resolution of rumours in social media: A survey*. ACM Computing Surveys, 51(2), 1–36

Muhammad Ajrul Amin bin Mohd Zaidi (A194789)
Prof. Dr. Shahrul Azman Mohd Noah
Faculty of Information Technology & Science
National University of Malaysia