This project is a Flask web application for predicting whether a given text is fake news or not. Here is a breakdown of the key components:

1. Imports:

* Flask: The main Flask module for creating the web application.
* jsonify, request, render\_template, redirect, url\_for: Various Flask utilities for handling HTTP requests, rendering templates, and redirects.
* PredictionModel: A custom class (not provided in the code snippet) for making predictions on text data.
* pandas: A library for data manipulation and analysis.
* randrange: A function from the random module for generating a random number within a specified range.
* OriginalTextForm: A custom form class (not provided in the code snippet) that likely inherits from Flask-WTF's FlaskForm for handling form data.

2. Flask App Initialization: An instance of the Flask application is created: app = Flask(\_\_name\_\_).

3. App Configuration: The Flask app is configured with a secret key: app.config['SECRET\_KEY'] = '4c99e0361905b9f941f17729187afdb9'. The secret key is essential for securing the forms in the application.

4. Routes:

* Home Route (/):
  + Handles both POST and GET requests.
  + Renders the 'home.html' template.
  + Generates a random text from a CSV file when the "Generate" button is clicked and displays it in the form.
  + Predicts the label for the provided text when the "Predict" button is clicked.
* Predict Route (/predict/<original\_text>):
  + Handles POST and GET requests.
  + Calls the PredictionModel to predict the label for the provided text.
  + Returns the prediction in JSON format.
* Random Route (/random):
  + Handles GET requests.
  + Retrieves a random entry from a CSV file and returns its title, text, and label in JSON format.

5. Form Handling: The code uses a form (presumably OriginalTextForm) to gather user input on the home route. The form likely includes fields for entering the original text and buttons for generating and predicting.

6. Running the App: The application is run when the script is executed directly (if \_\_name\_\_ == '\_\_main\_\_': app.run()).

Figure 1 depicts the graphical user interface (UI) of the proposed fake news prediction system. It provides an overview of how the application looks and the various components it contains. Figure 2 shows a scenario where a user has entered some text into an input or edit box within the application. After entering the text, the system predicts that the provided text is classified as real news. Figure 3 illustrates a situation where a user has inputted text into an edit box, and the system predicts that the provided text is classified as fake news. Figure 4 provides a step-by-step visual representation of how the proposed fake news prediction system works. It shows the transformation or processing of the original text and the subsequent prediction.

Figure 5 showcases a sample prediction generated by the system using random test data. It demonstrates the system's ability to make predictions on data that is not specifically presented in train or test data.

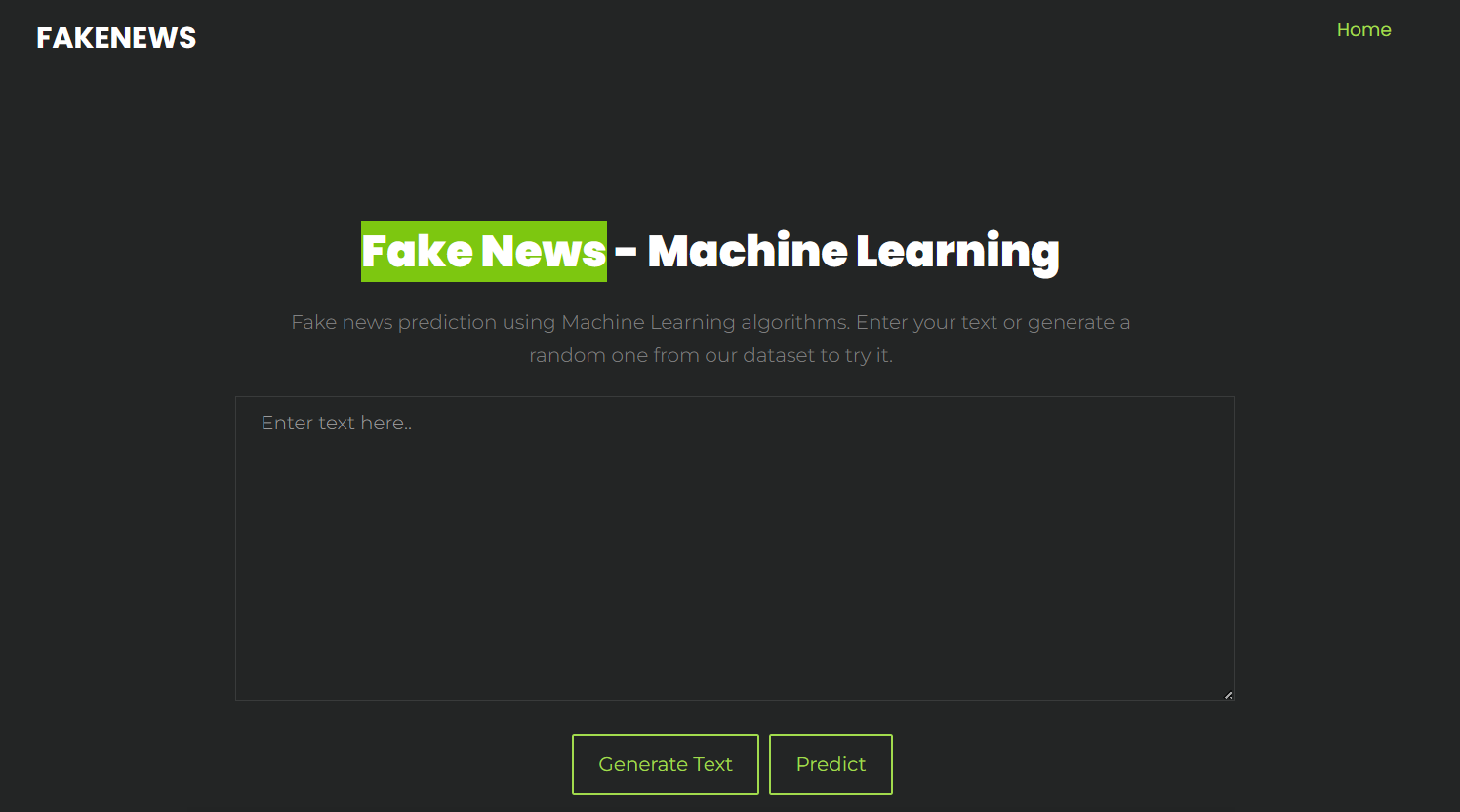


Figure 1: Sample UI application of proposed fake news prediction system.



Figure 2: User entered the text in given edit box and predicted as real news.



Figure 3: User entered the text in given edit box and predicted as fake news.

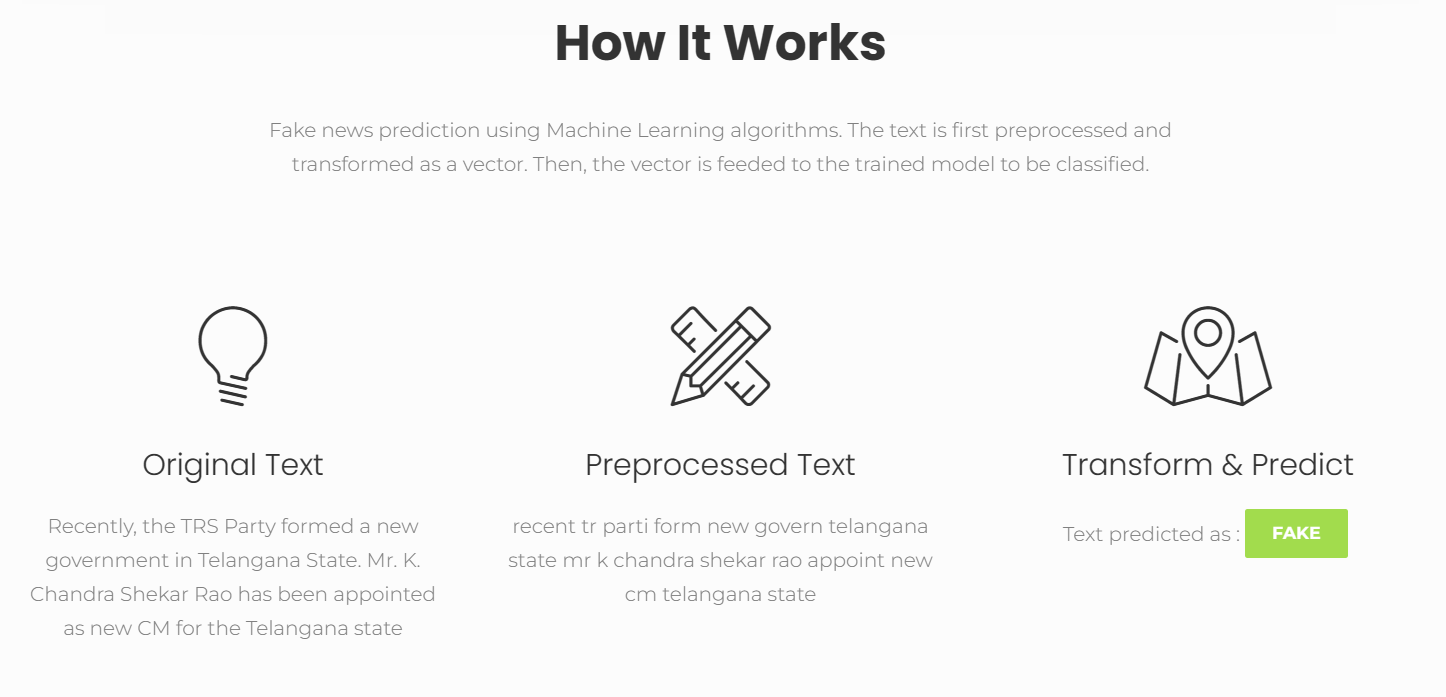
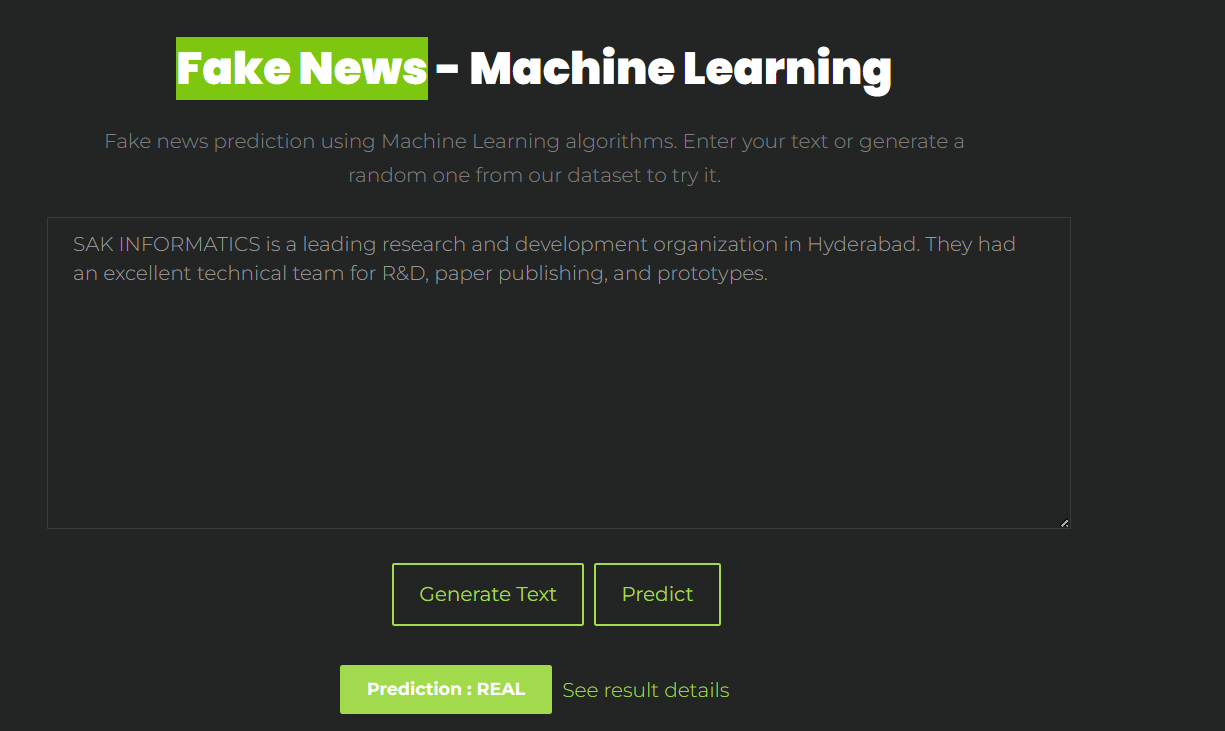


Figure 4: Displaying the working of proposed system from original text to transform and predict.



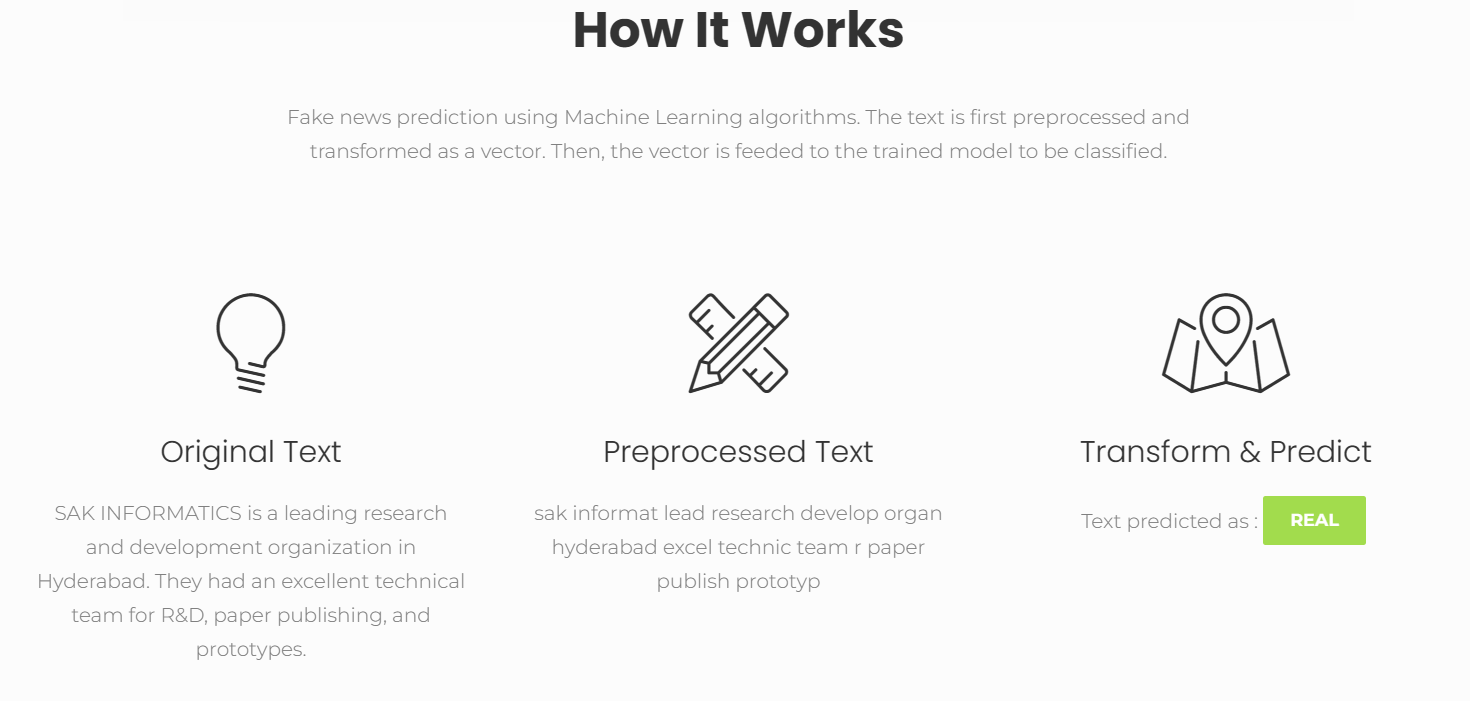


Figure 5: Sample prediction on random test data.