FAKE NEWS PREDICTION USING MACHINE LEARNING

by

Akanksha Vejendla (MST03-0035)

Submitted to Scifor Technologies

Meta Scifor Technologies

Script. Sculpt. Socialize

UNDER GUIDIANCE OF

Urooj Khan

TABLE OF CONTENTS

ABSTRACT

Fake news is a big problem on the internet. Fake news has become a pervasive issue in the digital age, with potentially severe consequences for society. In this report, we investigate the use of machine learning algorithms to tackle the problem of fake news detection. We leverage various algorithms and techniques to analyse and classify news articles as either real or fake. Our goal is to make it easier for people to know if what they're reading online is true or not. we aim to develop robust solutions for identifying and mitigating the spread of misinformation.

INTRODUCTION

The Rise of Fake News

In today's digital age, the internet has become a primary source of information for millions of people worldwide. While this connectivity has undoubtedly brought many benefits, it has also facilitated the rapid spread of misinformation and fake news. Fake news refers to false or misleading information presented as factual news content. It can range from fabricated stories and conspiracy theories to deliberately misleading headlines and doctored images.

Impact and Consequences

The proliferation of fake news poses significant challenges to society, democracy, and public discourse. Misinformation can have far-reaching consequences, including:

- Undermining trust in traditional media sources
- Influencing public opinion and decision-making processes
- Compromising the integrity of democratic institutions
- Potentially inciting violence or social unrest

Importance of Fake News Detection

Given the potential harm caused by fake news, there is a growing recognition of the importance of developing effective strategies for detecting and combating misinformation. Traditional methods of fact-checking and verification are often time-consuming and resource-intensive, making them impractical for addressing the scale and speed of fake news dissemination on digital platforms.

Role of Technology and Machine Learning

Advancements in technology, particularly in the fields of artificial intelligence and machine learning, offer promising solutions to the problem of fake news detection. By leveraging algorithms and data analysis techniques, it is possible to develop automated systems capable of identifying patterns and indicators of fake news content. These systems can help augment human efforts in verifying information and contribute to a more informed and resilient society

TECHNOLOGY USED

The technology stack used in this project are mentioned below:

1. Programming Language: Python
2. Libraries:
   - numpy – numpy is a Python library for numerical computing that provides high-performance multidimensional array objects and tools for working with these arrays.
   - pandas - pandas is a Python library for data manipulation and analysis that provides data structures.
   - matplotlib – data visualization library
   - seaborn – data visualization library
   - scikit-learn - Scikit-learn is a Python library for machine learning featuring various algorithms for classification, regression, clustering, and dimensionality reduction, with tools for model selection and evaluation.
   - streamlit - streamlit is a Python library for creating web applications with minimal effort, allowing users to build interactive and customizable data-driven applications using simple Python scripts.
   - PIL - PIL (Python Imaging Library) is a library in Python used for opening, manipulating, and saving many different image file formats.

DATASET INFORMATION

The dataset consists of two CSV files, "fake.csv" and "true.csv", each containing 21,418 rows and four columns: title, subject, date, and text. These files are structured to store information about news articles, with "fake.csv" presumably containing articles classified as fake or misleading, while "true.csv" holds articles verified as true or accurate. The "title" column likely holds the headline or title of each article, providing a brief summary of its content. The "subject" column likely categorizes the articles into different topics or domains, such as politics, health, or technology. The "date" column likely records the date when each article was published or last updated, facilitating temporal analysis. Lastly, the "text" column likely contains the main body of the article, providing the detailed information and content.

METHODOLOGY

Importing Libraries:

The project starts by importing essential libraries required for building the web application and performing machine learning tasks. This includes Streamlit for creating the user interface, pandas for data manipulation, and scikit-learn for implementing machine learning models.

Loading Data:

The project loads two CSV files containing datasets of fake and true news articles using pandas' `read_csv()` function. These datasets are then concatenated into a single DataFrame to facilitate model training and evaluation.

Preprocessing Data:

Preprocessing involves adding a target column ('fake' or 'true') to each dataset, which indicates the authenticity of the news articles. No further preprocessing, such as text cleaning or feature engineering, is performed at this stage.

Model Definition:

Four classification models (Naive Bayes, Logistic Regression, Decision Tree, and Random Forest) are defined using scikit-learn's Pipeline class. Each model is encapsulated within a pipeline consisting of three steps: CountVectorizer, TfidfTransformer, and the respective classifier.

CountVectorizer converts text documents into a matrix of token counts, while TfidfTransformer applies TF-IDF transformation to the token count matrix. The classifier (e.g., Multinomial Naive Bayes, Logistic Regression) is responsible for training and prediction.

Sidebar Input:

A Streamlit sidebar is created to enable user interaction. Users can input the text they want to classify and select the model they wish to use for prediction. This is achieved using Streamlit's sidebar components such as text input and dropdown selection.

Model Prediction:

Upon receiving user input, the selected model is trained on the training data, which comprises both fake and true news articles. Grid search and hyperparameter tuning, as shown in the initial code, are excluded in this version for simplicity.
The trained model is then used to predict the authenticity ('fake' or 'true') of the input text provided by the user. The predicted label is displayed on the main app interface along with the name of the selected model.

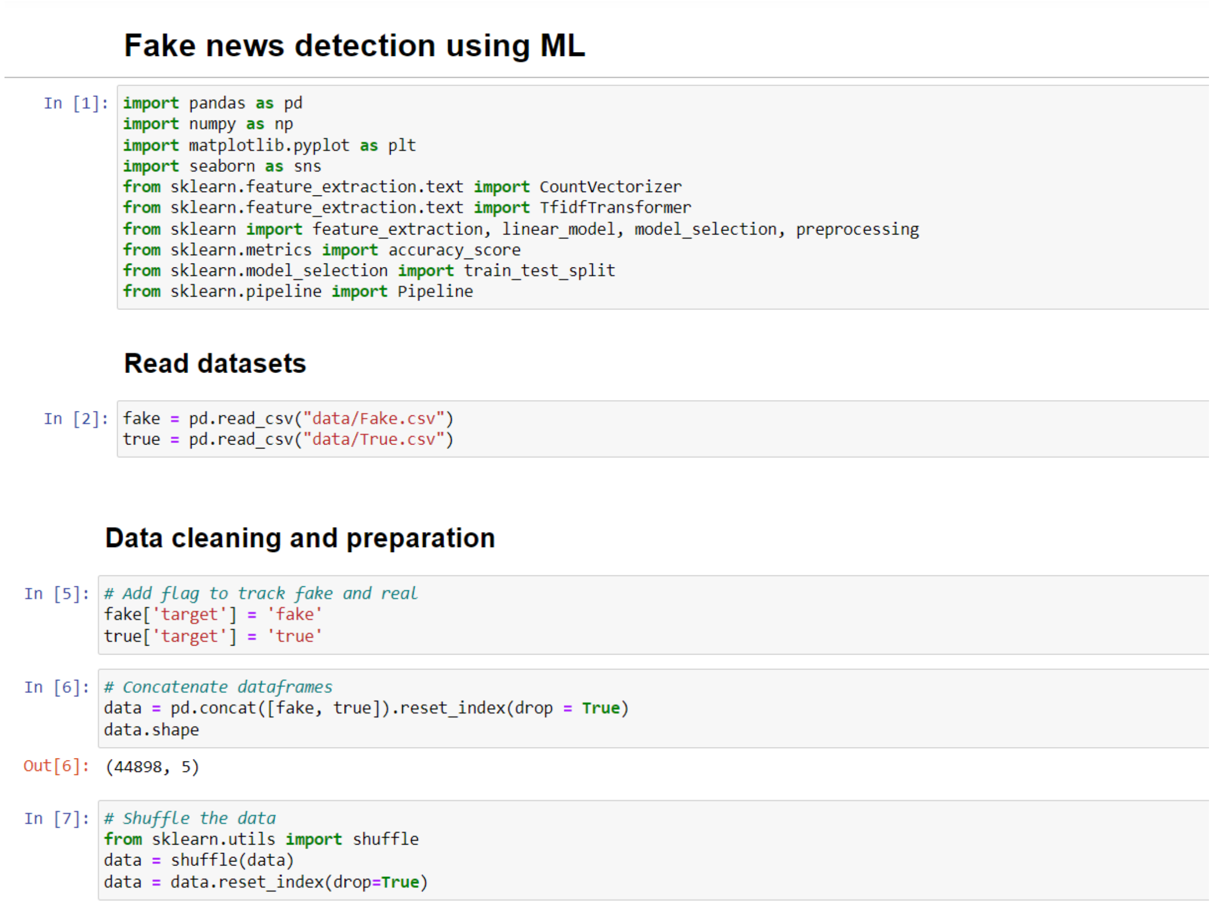
Image Display:

To enhance the user experience, associated images representing fake and true news are displayed in the sidebar based on the predicted label. If the model predicts the input text as 'fake', an image representing fake news is shown, and vice versa.
The inclusion of images serves as visual cues, reinforcing the classification result and making the web application more intuitive and engaging for users.

Custom Styling:

Custom CSS styling is applied to the web application using HTML and Streamlit's `markdown()` function. This allows for the customization of font styles, colors, and layout to enhance the visual appeal and user experience of the application.

CODE SNIPPET



Fake news detection using ML

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.feature_extraction.text import TfidfTransformer
        from sklearn import feature_extraction, linear_model, model_selection, preprocessing
        from sklearn.metrics import accuracy_score
        from sklearn.model_selection import train_test_split
        from sklearn.pipeline import Pipeline
```

Read datasets

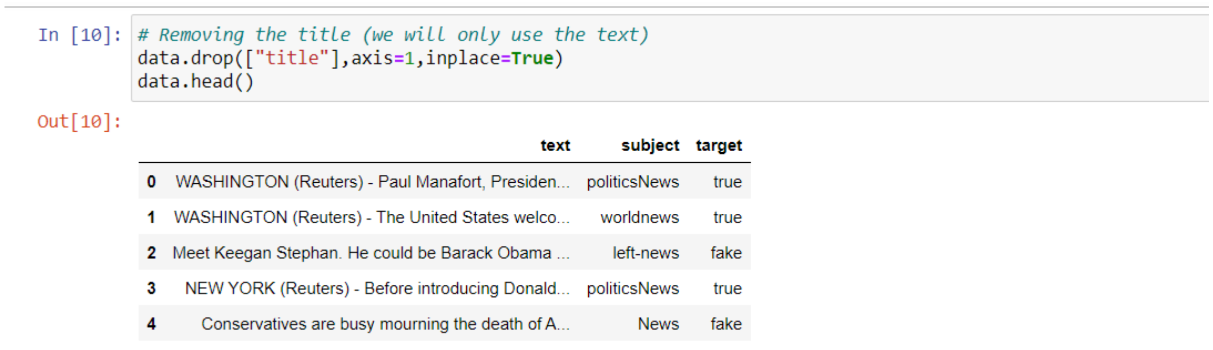
```
In [2]: fake = pd.read_csv("data/Fake.csv")
        true = pd.read_csv("data/True.csv")
```

Data cleaning and preparation

```
In [5]: # Add flag to track fake and real
        fake['target'] = 'fake'
        true['target'] = 'true'
```

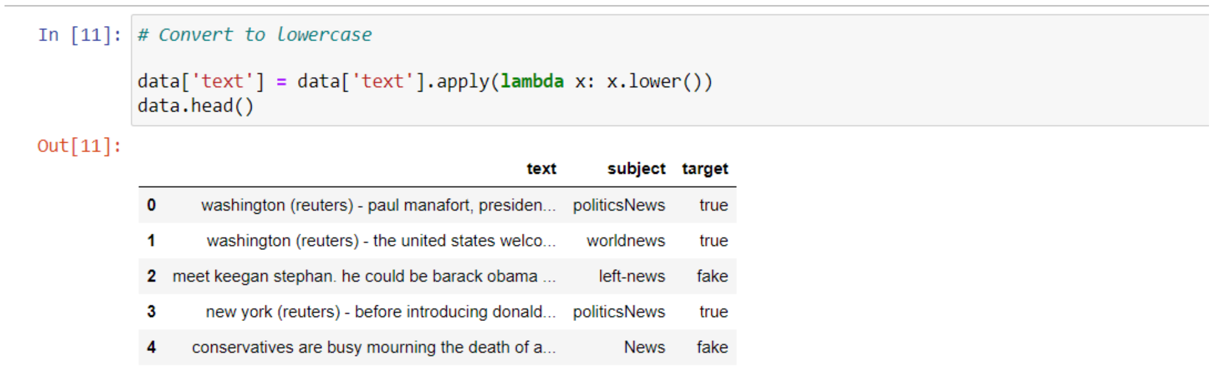
```
In [6]: # Concatenate dataframes
        data = pd.concat([fake, true]).reset_index(drop = True)
        data.shape

Out[6]: (44898, 5)
```

```
In [7]: # Shuffle the data
        from sklearn.utils import shuffle
        data = shuffle(data)
        data = data.reset_index(drop=True)
```



```
In [10]: # Removing the title (we will only use the text)
         data.drop(["title"],axis=1,inplace=True)
         data.head()

Out[10]:
```

	text	subject	target
0	WASHINGTON (Reuters) - Paul Manafort, Presiden...	politicsNews	true
1	WASHINGTON (Reuters) - The United States welco...	worldnews	true
2	Meet Keegan Stephan. He could be Barack Obama ...	left-news	fake
3	NEW YORK (Reuters) - Before introducing Donald...	politicsNews	true
4	Conservatives are busy mourning the death of A...	News	fake



```
In [11]: # Convert to lowercase

         data['text'] = data['text'].apply(lambda x: x.lower())
         data.head()

Out[11]:
```

	text	subject	target
0	washington (reuters) - paul manafort, presiden...	politicsNews	true
1	washington (reuters) - the united states welco...	worldnews	true
2	meet keegan stephan. he could be barack obama ...	left-news	fake
3	new york (reuters) - before introducing donald...	politicsNews	true
4	conservatives are busy mourning the death of a...	News	fake

Naive Bayes

]:

```python
dct = dict()

from sklearn.naive_bayes import MultinomialNB

NB_classifier = MultinomialNB()
pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', NB_classifier)])

model = pipe.fit(X_train, y_train)
prediction = model.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))

dct['Naive Bayes'] = round(accuracy_score(y_test, prediction)*100,2)
```

```
accuracy: 95.46%
```

Logistic regression

3]:

```python
# Vectorizing and applying TF-IDF
from sklearn.linear_model import LogisticRegression

pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', LogisticRegression())])

# Fitting the model
model = pipe.fit(X_train, y_train)

# Accuracy
prediction = model.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
dct['Logistic Regression'] = round(accuracy_score(y_test, prediction)*100,2)
```

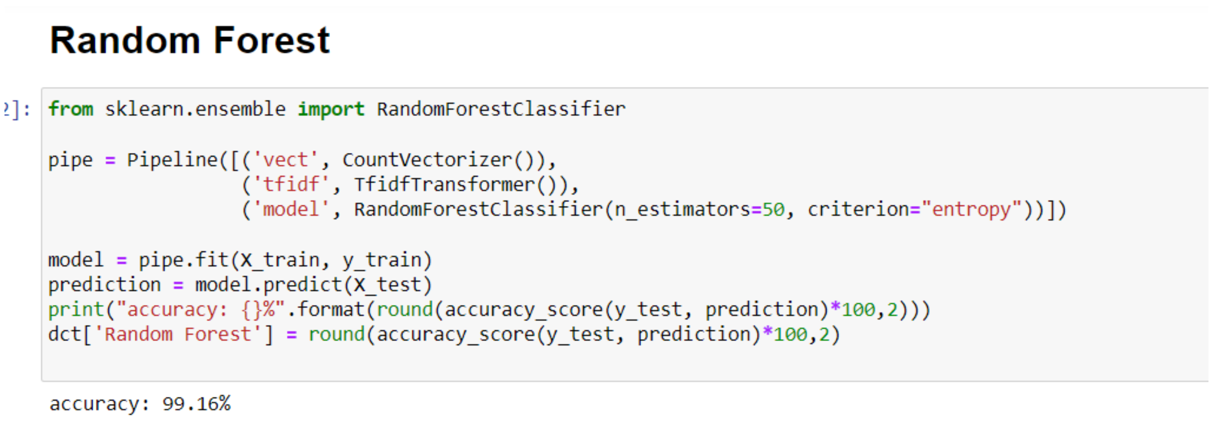
```
accuracy: 98.89%
```

Decision Tree

30]:

```python
from sklearn.tree import DecisionTreeClassifier

# Vectorizing and applying TF-IDF
pipe = Pipeline([('vect', CountVectorizer()),
                 ('tfidf', TfidfTransformer()),
                 ('model', DecisionTreeClassifier(criterion= 'entropy',
                                                  max_depth = 20,
                                                  splitter='best',
                                                  random_state=42))])
# Fitting the model
model = pipe.fit(X_train, y_train)

# Accuracy
prediction = model.predict(X_test)
print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
dct['Decision Tree'] = round(accuracy_score(y_test, prediction)*100,2)
```

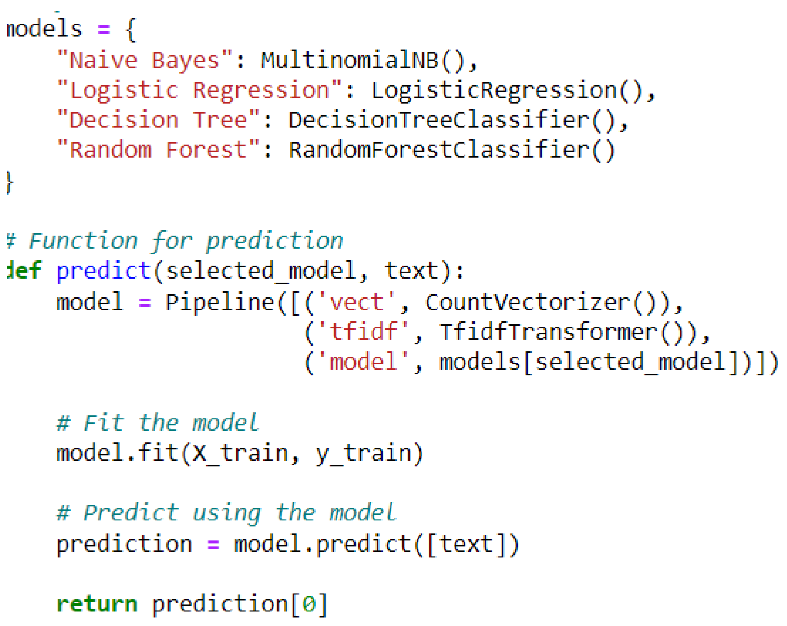
```
accuracy: 99.62%
```



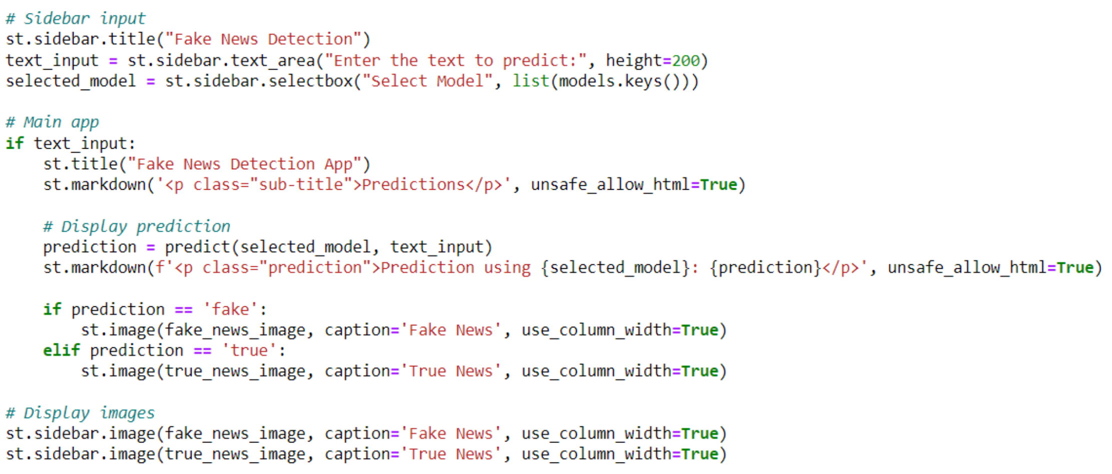
Random Forest

```
]: from sklearn.ensemble import RandomForestClassifier

   pipe = Pipeline([('vect', CountVectorizer()),
                    ('tfidf', TfidfTransformer()),
                    ('model', RandomForestClassifier(n_estimators=50, criterion="entropy"))])

   model = pipe.fit(X_train, y_train)
   prediction = model.predict(X_test)
   print("accuracy: {}%".format(round(accuracy_score(y_test, prediction)*100,2)))
   dct['Random Forest'] = round(accuracy_score(y_test, prediction)*100,2)

   accuracy: 99.16%
```

STREAMLIT CODE:

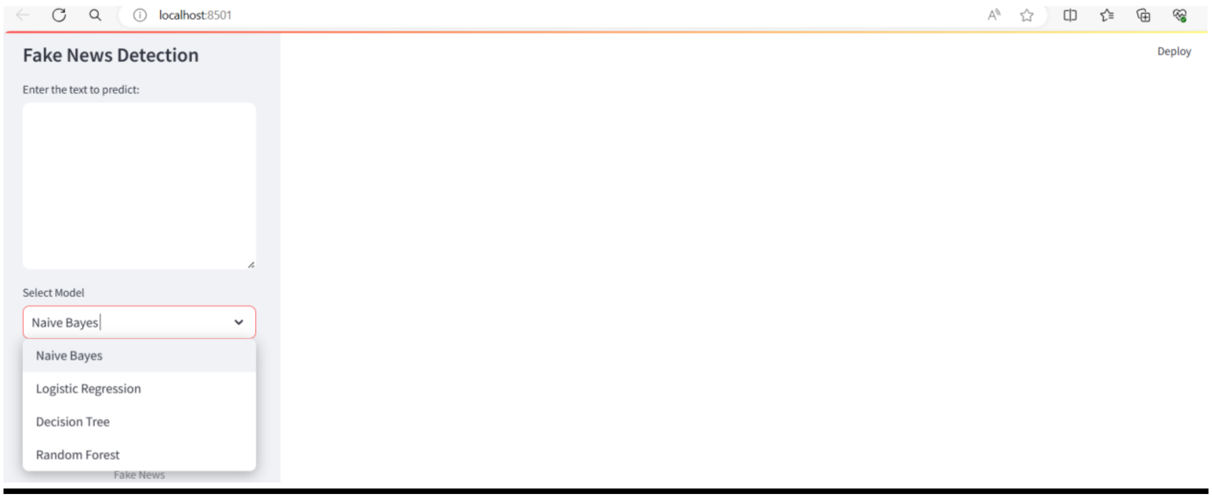


```
nodels = {
    "Naive Bayes": MultinomialNB(),
    "Logistic Regression": LogisticRegression(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier()
}

# Function for prediction
lef predict(selected_model, text):
    model = Pipeline([('vect', CountVectorizer()),
                      ('tfidf', TfidfTransformer()),
                      ('model', models[selected_model])])

    # Fit the model
    model.fit(X_train, y_train)

    # Predict using the model
    prediction = model.predict([text])

    return prediction[0]
```



```
# Sidebar input
st.sidebar.title("Fake News Detection")
text_input = st.sidebar.text_area("Enter the text to predict:", height=200)
selected_model = st.sidebar.selectbox("Select Model", list(models.keys()))

# Main app
if text_input:
    st.title("Fake News Detection App")
    st.markdown('<p class="sub-title">Predictions</p>', unsafe_allow_html=True)

    # Display prediction
    prediction = predict(selected_model, text_input)
    st.markdown(f'<p class="prediction">Prediction using {selected_model}: {prediction}</p>', unsafe_allow_html=True)

    if prediction == 'fake':
        st.image(fake_news_image, caption='Fake News', use_column_width=True)
    elif prediction == 'true':
        st.image(true_news_image, caption='True News', use_column_width=True)

# Display images
st.sidebar.image(fake_news_image, caption='Fake News', use_column_width=True)
st.sidebar.image(true_news_image, caption='True News', use_column_width=True)
```

RESULTS AND DISCUSSION



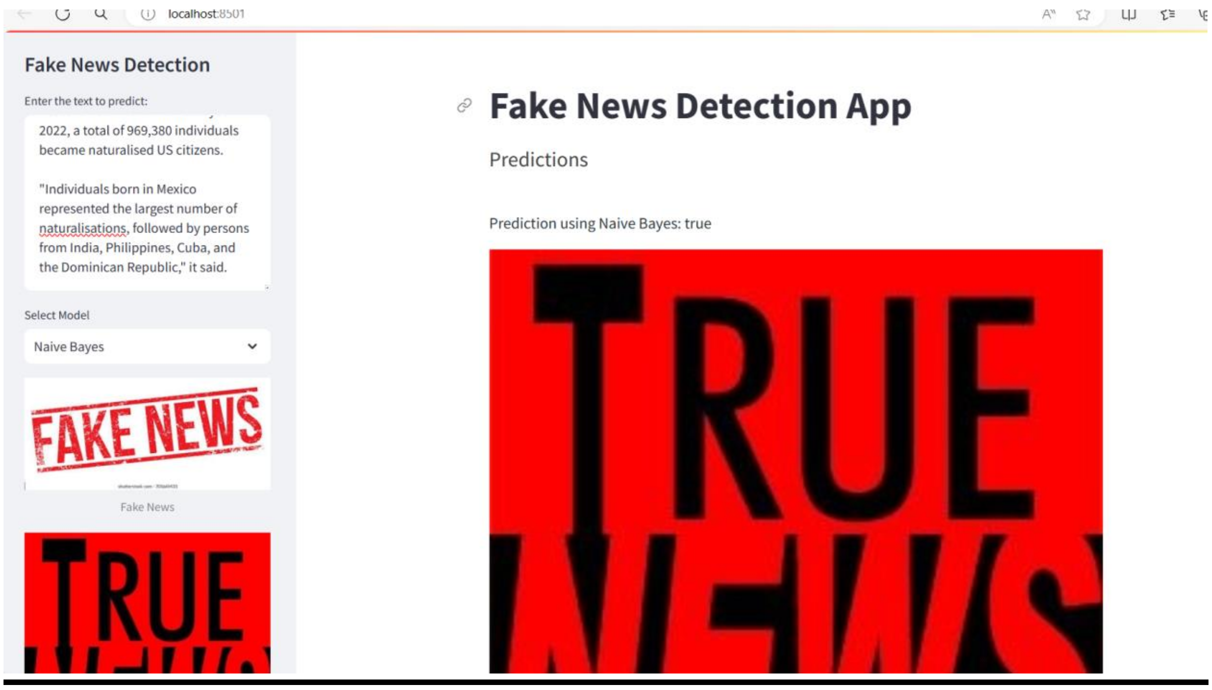
Fake News Detection

Enter the text to predict:

Select Model

Naive Bayes

- Naive Bayes
- Logistic Regression
- Decision Tree
- Random Forest



Fake News Detection

Enter the text to predict:

2022, a total of 969,380 individuals became naturalised US citizens.

"Individuals born in Mexico represented the largest number of naturalisations, followed by persons from India, Philippines, Cuba, and the Dominican Republic," it said.

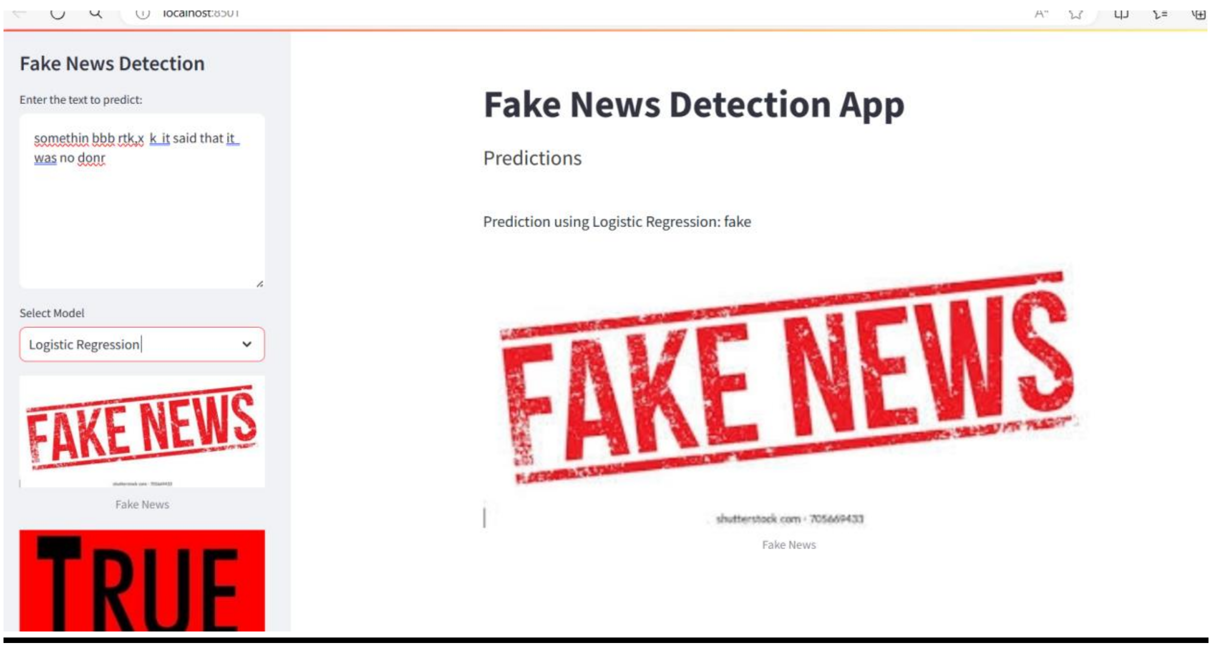
Select Model

Naive Bayes

Fake News Detection App

Predictions

Prediction using Naive Bayes: true



localhost:8501

Fake News Detection

Enter the text to predict:

somethin bbb rtk,x k it said that it was no donr

Select Model

Logistic Regression

Fake News Detection App

Predictions

Prediction using Logistic Regression: fake

CONCLUSION

In wrapping up our project, we've made significant strides in developing a tool that helps identify fake news articles. While there's still work to be done, our model shows promise in distinguishing between real and fake news, offering a potential solution to combat misinformation online.

With continued dedication and innovation, we can work towards a future where misinformation is less prevalent, and individuals are empowered to make informed decisions about the content they consume. Our project represents just the beginning of this journey towards a more truthful and trustworthy online landscape.

This conclusion emphasizes the progress made in the project, the potential impact of the model, and the importance of ongoing efforts to combat misinformation. Adjustments can be made based on the specific findings and outcomes of your project.

REFERENCES

https://www.learnpython.org/

https://www.geeksforgeeks.org/machine-learning-algorithms/

https://scikit-learn.org/stable/tutorial/index.html

https://www.javatpoint.com/how-to-check-the-accuracy-of-your-machine-learning-model