InTrAinZ Internship

Spam News Detection and Classification

Internship : Artificial Intelligence

Name : Y Pradeep Reddy College : AITS Tirupati Course : AIML

Topics / Agenda :-

- Spam News Detection
- Python libraries used
- True and Fake CSV data sets
- Training and Testing Program
- Conclusion

Spam News Detection:

What is spam news detection:

Spam Filter : A spam filter is a program used to detect unsolicited, unwanted and virus-infected emails and prevent those messages from getting to a user's inbox. Like other types of filtering programs, a spam filter looks for specific criteria on which to base its judgments.

--In spam News Detection we are training the machine first how true news looks like and how fake news looks like.

--The type of machine learning algorithm that would be used to train a system to detect spam in email messages or fake news is a supervised learning algorithm, specifically a classification algorithm.

--For Training the machine we have to take "True news" data as well as "Fake News " data in the form of ".csv" extension. which is edited in the MS Excel.
--After training the machine it can predicts the news whether the news is true or
fake.

Python libraries used:-

numpy (import numpy as np):

NumPy is a library for numerical computations in Python. It provides support for large, multi-dimensional arrays and matrices, along with mathematical functions to operate on these arrays.

pandas (import pandas as pd):

Pandas is a powerful data manipulation library. It provides data structures like DataFrame and Series, making it easy to manipulate, clean, and analyze structured data.
Statements used:
pd.read_csv("True_news.csv"): Reads a CSV file into a DataFrame.
pd.concat([dataset1, dataset2]): Concatenates two DataFrames.

nltk (import nltk):

NLTK (Natural Language Toolkit) is a library for working with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries.
Statements used:
nltk.download('wordnet'): Downloads the WordNet dataset

re (import re):

The re module is Python's regular expression module, allowing for pattern matching with strings.
Statements used:
re.sub('[^a-zA-Z]', ' ', row): Substitutes non-alphabetic characters with spaces.

stopwords (from nltk.corpus import stopwords):

NLTK's stopwords module provides a list of common words (e.g., "the," "and") that are often removed from text during natural language processing tasks.
Statements used:
stopwords.words('english'): Retrieves a list of English stopwords.

sklearn (from sklearn...):

Scikit-learn is a machine learning library in Python. It provides simple and efficient tools for data analysis and modeling, including various algorithms for classification, regression, clustering, and more.
Statements used:
from sklearn.feature_extraction.text import TfidfVectorizer: TF-IDF vectorizer for converting text data into numerical vectors.

from sklearn.model_selection import train_test_split: Splits data into training and testing sets.

from sklearn.naive_bayes import MultinomialNB: Multinomial Naive Bayes classifier.

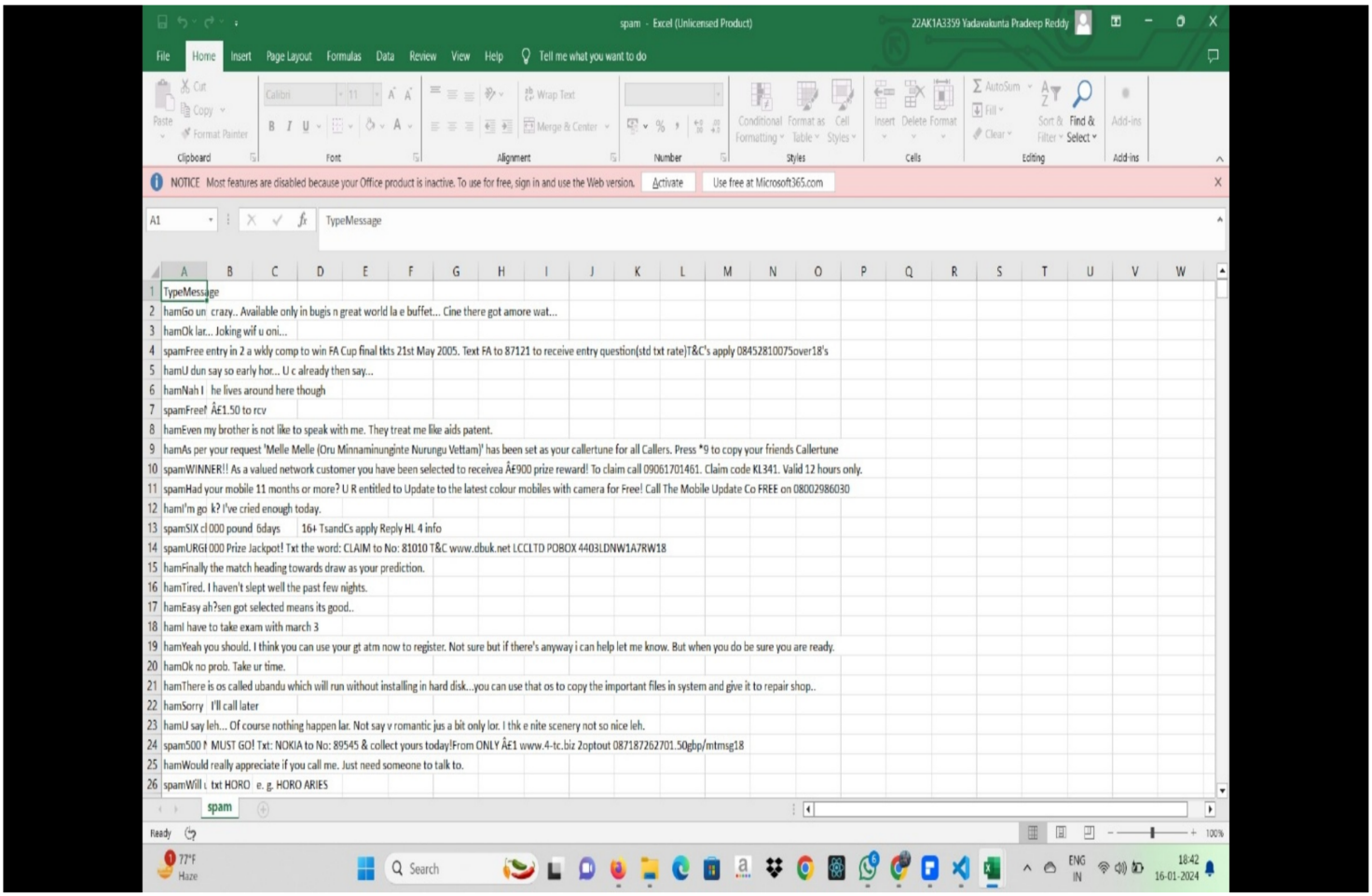
from sklearn.metrics import accuracy_score: Computes the accuracy of the classification.

These libraries are used to perform tasks such as handling, text processing, feature extraction, machine learning model training, and performance evaluation in the provided program.

True and Fake CSV data sets :-

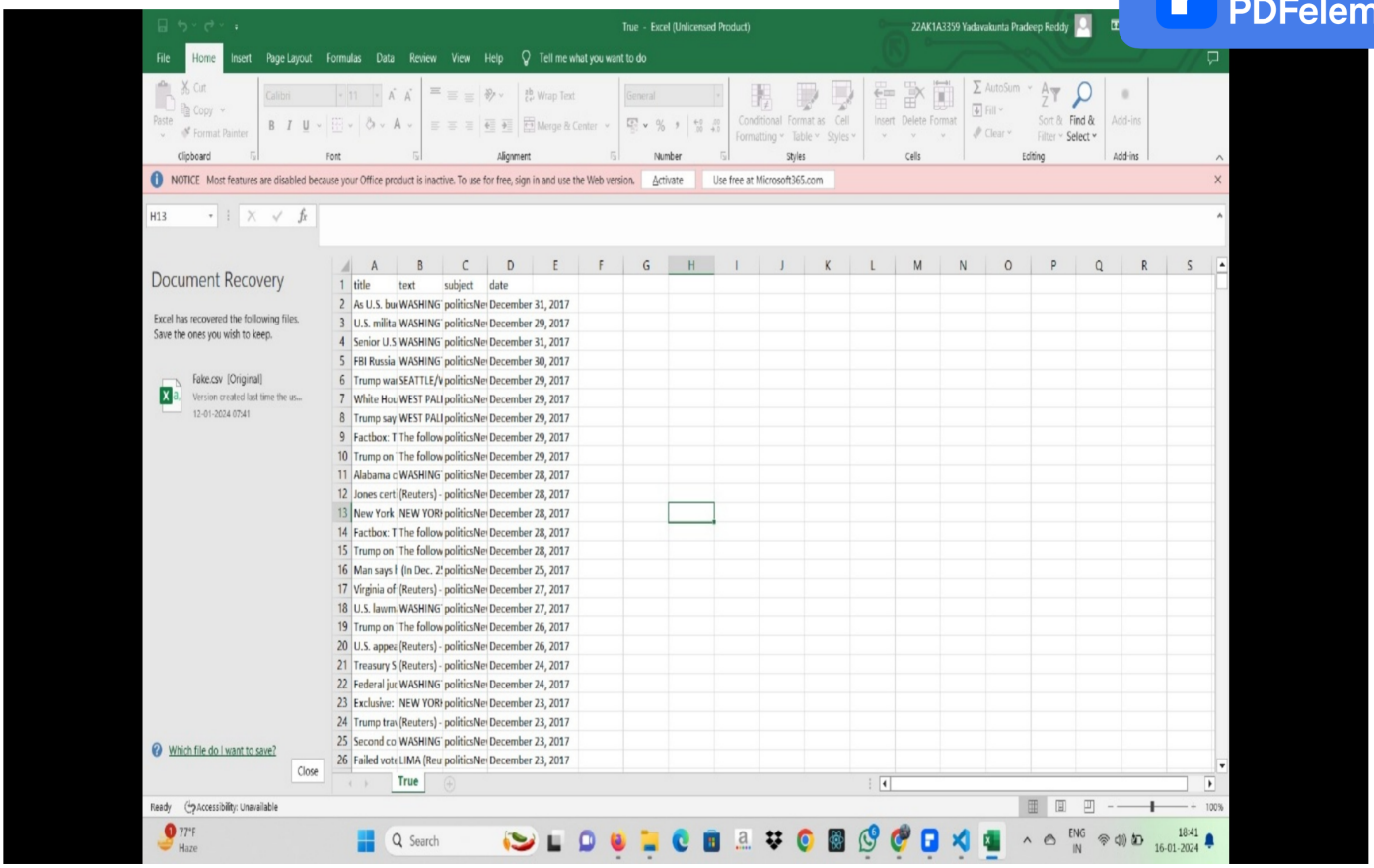
For True CSV data set:

Github link : https://github.com/ShyamPradeepReddy/Spam-News-Detection/blob/main/True.csv



For Fake CSV data set:

Github link : https://github.com/ShyamPradeepReddy/Spam-News-Detection/blob/main/Fake.csv



Why CSV files are used:-

CSV (Comma-Separated Values) files are a common plain-text format used for storing tabular data. Each line in a CSV file represents a row, and the values within each row are separated by commas (or other delimiters).

Advantages of using CSV files.

Simplicity: CSV files are simple and easy to understand. They are plain text, making them human-readable.

Common Use Cases: CSV files are widely used for data exchange between different programs, databases, and spreadsheet applications.

Character Encoding: CSV files are often encoded using UTF-8 or ASCII, ensuring compatibility across different systems.

*HeaderRow :*The first row often contains column names, making it easier to interpret the data.

*Quoting*Values containing special characters (like commas) are often enclosed in quotes.

Applications: CSV files are commonly used in data analysis, data science, and as an intermediate format for data import/export.

Training and Testing Program:-

```
import pandas as pd
import numpy as np
True_news = pd.read_csv("True_news.csv")
#loads the True_news data into True_news
Fake_news = pd.read_csv("Fake_news.csv")
#loads the Fake_news data into Fake_news
#Add label values
True_news['label']=0
Fake_news['label']=1
dataset1 = True_news[[ 'text' , 'label' ]]
dataset2 = Fake_news[[ 'text' , 'label']]
#Selecting only two columns and loading
into dataset1 and dataset2
dataset = pd.concat([ dataset1, dataset2])
#concatenating two datasets
#checking for null values
dataset.isnull().sum()          #returns
text =0 and label 0
#For shuffling the data
dataset = dataset.sample(frac = 1)
```

```
import nltk
import re
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer ps =
WordNetLemmatizer()
stopwords=stopwords.words('english') nltk.
download('wordnet')def clean_row(row):
row=row.lower()
row=re.sub('[^a-ZA-Z]', ' ',row)
token=row.split()
news=[ps.lemmatize(word) for word in token if
not word in stopwords] cleanned_news=''.join(
news)
return cleanned_news
dataset['text']=dataset['text'].apply(lambda x:
clean_row(x))
from sklearn.feature_extraction.text import
Tfidfvectorizer
vectorizer = Tfidfvectorizer(max_features =
50000, lowercase=False,
ngram_range=(1,2))
x=dataset.iloc[:35000,0]
y=dataset.iloc[:35000,1]
from sklearn.model_selection import train_test_
split
train_data,test_data,train_label,test_label=
train_test_split(x,y,test_size=0.2,rand om_state
=0)
vec_train_data=vectorizer.fit_transform(train_
data) vec_train_data=vec_test_data.toarray()
vec_test_data=vectorizer.fit_transform(test_data
) vec_test_data=vec_test_data.toarray()
```

```
training_data=pd.DataFrame(vec_train_data, columns
=vectorizer.get_feature_names()) testing_data=pd.
DataFrame(vec_#Model
from sklearn.naive_bayes import MultinomialNB
clf=MultinomialNB()
clf.fit(training_data,train_label)
y_pred=clf.predict(testing_data)
from sklearn.metrics import accuracy_score
accuracy_score(test_label,y_pred)    #returns
the accuracy of prediction y_pred_train=clf.
predict(training_data)
txt="Some news we will give here"
news=clean_row(txt)
pred=clf.predict(vectorizer.transform([news]).
toarray()) txt=input("Enter News:")
news=clean_row(str(txt))
pred=clf.predict(vectorizer.transform([news]).
toarray()) if pred == 0:
print("News is correct") else:
print("News is fake")
```

Steps involved in training the above model:-

Importthepandas and numpy libraries

```
import pandas as pd
import numpy as np
```

Read a CSV filenamed "True_news.csv" into a DataFramecalledTrue_news.

```
True_news = pd.read_csv("True_news.csv")
```

Read a CSV file named "Fake_news.csv" into a DataFrame called Fake_news.

```
Fake_news = pd.read_csv("Fake_news.csv")
```

Add labels to the news.

```
True_news['label']=0
Fake_news['label']=1
```

Select only the 'text' and 'label' columns from both True_news and Fake_news DataFrames and store them in dataset1 and dataset2 respectively.

```
dataset1 = True_news[['text', 'label']]
dataset2 = Fake_news[['text', 'label']]
```

Concatenate dataset1 and dataset2 along the rows (axis=0) to create a new DataFrame called dataset.

```
dataset = pd.concat([dataset1, dataset2])
```

Check for null values in the dataset DataFrame. Returns the count of null values for each column ('text' and 'label').

```
dataset.isnull().sum()
```

Shuffle the rows of the dataset DataFrame.

```
dataset = dataset.sample(frac=1)
```

Import the Natural Language Toolkit (nltk) library, regular expression library (re), and download the WordNet dataset.

```
import nltk
import re
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
ps = WordNetLemmatizer()
stopwords = stopwords.words('english')
nltk.download('wordnet')
```

Define a function clean_row that performs text cleaning on a given row.

```
def clean_row(row):
    row = row.lower()
    row = re.sub('[^a-zA-Z]', ' ', row)
    token = row.split()
    news = [ps.lemmatize(word) for word in token if not
word in stopwords]
    cleaned_news = ' '.join(news)
    return cleaned_news
```

Apply the clean_row function to the 'text' column of the dataset DataFrame

```
dataset['text'] = dataset['text'].apply(lambda x: clean_
row(x))
```

Import TfidfVectorizer from scikit-learn and create a TfidfVectorizer object with specified parameters.

```
from sklearn.feature_extraction.text import
TfidfVectorizer
vectorizer = TfidfVectorizer(max_features=50000, lowercase
=False, ngram_range=(1, 2))
```

Extract the 'text' column as input (x) and 'label' column as output (y) from the first 35,000 rows of the dataset DataFrame

```
x = dataset.iloc[:35000, 0]
y = dataset.iloc[:35000, 1]
```

```
Split the data into training and testing sets using the train_test_split functio

from sklearn.model_selection import train_test_split
train_data, test_data, train_label, test_label = train_test_
split(x, y, test_size=0.2, random_state=0)

Apply the vectorizer to the training data and convert it to a dense array

vec_train_data = vectorizer.fit_transform(train_data)
vec_train_data = vec_train_data.toarray()

Apply the vectorizer to the testing data and convert it to a dense array

vec_test_data = vectorizer.fit_transform(test_data)
vec_test_data = vec_test_data.toarray()

Create DataFrames training_data and testing_data with feature names
obtained from the vectorizer

training_data = pd.DataFrame(vec_train_data, columns=
vectorizer.get_feature_names())
testing_data = pd.DataFrame(vec_test_data, columns=vectorizer
.get_feature_names())

Import MultinomialNaiveBayes from scikit-learn, create a classifier object,
and fit it to the training data.

from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB()
clf.fit(training_data, train_label)

Predict the labels for the testing data using the trained classifier.

y_pred = clf.predict(testing_data)
```

Calculate and print the accuracy score for the predictions of the testing d

```
from sklearn.metrics import accuracy_score
accuracy_score(test_label, y_pred)
```

Clean a sample news text and make predictions using the trained classifi

```
txt = "Some news we will give here"
news = clean_row(txt)
pred = clf.predict(vectorizer.transform([news]).toarray())
```

For Knowing the result.
Take user input for news, clean it, and predict whether it is considered corr
fake based on the trained classifier. Print the result.

```
txt = input("Enter News:")
news = clean_row(str(txt))
pred = clf.predict(vectorizer.transform([news]).toarray())
if pred == 0:
    print("News is correct")
else:
    print("News is fake")
```

Conclusion:-

Spam news detection Python program processes two CSV files containing true and fake news data. It uses Pandas for data manipulation, NLTK for text preprocessing, and scikit-learn for text vectorization and classification. The code cleans and combines the datasets, shuffles the data, applies lemmatization and removal of stopwords to the text, and utilizes TF-IDF vectorization. A Multinomial Naive Bayes classifier is trained on the preprocessed text data to distinguish between true and fake news.

The program calculates and prints the accuracy of the classification on a test set. Additionally, it allows users to input news text, cleans it, and predicts whether the news is considered correct or fake based on the trained classifier. The code is an example of a simple text classification pipeline using machine learning techniques to differentiate between true and fake news.

Created by Pradeep
Thank you