Faculty of Computers & Information Technology

Fake News Detection

(Pattern Recognition Project)

Under Supervision of :

Prof/ DR.Dina Khattab

TA/ ENG.Mohamed Mustafa Saad

Participated in this project :

1. Patrick Masry (21-01438)
2. Awsam Lotfallah (21-00534)
3. Mena Alkess (21-01067)
4. Sabry Adel (21-01526)
5. Thomas Zaki (21-01669)

Fake news detection

Introduction :

Welcome to our Pattern Recognition Project : Spotting Fake News! In this project, we're using special computer programs to tell real news from fake news. It's like having a truth detector for the internet!

With so much fake news circulating online, it's tough to know what to trust. Our project is all about training computers to sniff out the real stories from the phony ones. By analyzing lots of news articles, we're teaching computers to recognize the signs of fake news and steer you away from unreliable sources.

We're excited to show you how our project can make navigating the news safer and more reliable. Let's dive in and uncover the truth hidden within the digital world of news !

==========

Libraries Utilized :

- ***pandas*** : For efficient data manipulation and analysis.

- ***numpy*** : For numerical computations and array operations.

- ***sklearn.metrics*** : Provides a suite of metrics for evaluating model performance.

- ***pickle*** : For serializing and deserializing Python objects, particularly model objects.

- ***matplotlib.pyplot*** : For visualization of data and evaluation metrics.

- ***PIL.Image*** : For working with images, particularly to display them inline.

- ***nltk*** : The Natural Language Toolkit for text processing tasks such as tokenization and stemming.

- ***re*** : Regular expression operations for pattern matching and string manipulation.

- ***sklearn.feature_extraction.text.TfidfVectorizer*** : For converting text data into numerical feature vectors using the TF-IDF technique.

- ***sklearn.model_selection.train_test_split*** : For splitting the dataset into training and testing sets.

- ***sklearn.linear_model.LogisticRegression*** : Implementation of logistic regression classifier.

- ***sklearn.svm.SVC*** : Implementation of Support Vector Classifier.

- ***sklearn.ensemble.RandomForestClassifier*** : Implementation of Random Forest classifier.

==========

Pipeline Steps :

1. Importing Libraries : Necessary libraries are imported to support various tasks involved in text classification, including data manipulation, preprocessing, model building, and evaluation.

2. Text Preprocessing : The script employs NLTK for text preprocessing tasks such as removing stopwords, stemming, and basic text cleaning using regular expressions.

3. Feature Engineering : Text data is transformed into numerical feature vectors using the TF-IDF technique. This process assigns weights to words based on their frequency in the document and across the corpus.

4. Dataset Splitting : The dataset is split into training and testing subsets using train_test_split function from sklearn.model_selection module. This facilitates model training on one portion of the data and evaluation on another.

5. Model Training : Different classification algorithms including Logistic Regression, Support Vector Machine, and Random Forest are trained on the training dataset to learn patterns in the data.

6. Model Evaluation : The trained models are evaluated using various metrics such as accuracy score, ROC curve, confusion matrix, and classification report. These metrics provide insights into the performance of the models and help in selecting the best-performing one.

==========

```
dataset = pd.read_csv('News Detections_Training Part.csv')
df = pd.DataFrame(dataset)
df
```

1. ***Dataset Loading*** : The code reads a CSV file named 'News Detections_Training Part.csv' using the pandas read_csv() function. This function loads the data from the CSV file into pandas DataFrame object.

2. ***DataFrame Creation*** : The DataFrame object is stored in a variable named 'df'. This DataFrame contains the data from the CSV file, allowing for easy manipulation, analysis, and exploration of the dataset.

Output :

	news_url	source_domain	real	news	news_reply
0	http://toofab.com/2017/05/08/real-housewives-a...	toofab.com	1	Kandi Burruss Explodes Over Rape Accusation on...	42
1	https://www.today.com/style/see-people-s-choic...	www.today.com	1	People's Choice Awards 2018: The best red carp...	0
2	https://www.etonline.com/news/220806_sophia_bu...	www.etonline.com	1	Sophia Bush Sends Sweet Birthday Message to 'O...	63
3	https://www.dailymail.co.uk/news/article-33655...	www.dailymail.co.uk	1	Colombian singer Maluma sparks rumours of inap...	20
4	https://www.zerchoo.com/entertainment/gossip-g...	www.zerchoo.com	1	Gossip Girl 10 Years Later: How Upper East Sid...	38
...	...	...	...	...	...
16994	https://www.kqed.org/pop/28206/how-amber-rose-...	www.kqed.org	1	How Amber Rose Became an Unlikely Feminist Icon	67
16995	www.usmagazine.com/celebrity-moms/news/rihanna...	www.usmagazine.com	0	Rihanna Cradles Newborn Baby in Loving Twitter...	81
16996	https://www.longroom.com/discussion/710361/kou...	www.longroom.com	1	Kourtney Kardashian and Younes Bendjima Are Ma...	56
16997	https://www.mid-day.com/articles/emmy-awards-2...	www.mid-day.com	1	Emmy Awards 2017: Elisabeth Moss found out abo...	8
16998	gunrights.trendolizer.com/2017/10/jason-aldean...	gunrights.trendolizer.com	0	Jason Aldean Gig Canceled After He Sells Out T...	22

16999 rows × 5 columns

==========

dataset.head()

Displaying First Few Rows: The code uses the '.head()' method on the DataFrame 'dataset' to display the first few rows of the DataFrame.

Output :

	news_url	source_domain	real	news	news_reply
0	http://toofab.com/2017/05/08/real-housewives-a...	toofab.com	1	Kandi Burruss Explodes Over Rape Accusation on...	42
1	https://www.today.com/style/see-people-s-choic...	www.today.com	1	People's Choice Awards 2018: The best red carp...	0
2	https://www.etonline.com/news/220806_sophia_bu...	www.etonline.com	1	Sophia Bush Sends Sweet Birthday Message to 'O...	63
3	https://www.dailymail.co.uk/news/article-33655...	www.dailymail.co.uk	1	Colombian singer Maluma sparks rumours of inap...	20
4	https://www.zerchoo.com/entertainment/gossip-g...	www.zerchoo.com	1	Gossip Girl 10 Years Later: How Upper East Sid...	38

==========

dataset.columns

Output :

Index(['news_url', 'source_domain', 'real', 'news', 'news_reply'], dtype='object')

1. ***'news_url'*** : Represents the URL of the news article.
2. ***'source_domain'*** : Represents the domain of the news source.
3. ***'real'*** : Indicates whether the news is real (1) or not (0).
4. ***'news'*** : Contains the content of the news article.
5. ***'news_reply'*** : Possibly represents replies or comments related to the news article.

==========

dataset.isnull().sum()

1. ***dataset*** : Represents the dataset being analyzed.
2. ***.isnull()*** : Method used to create a boolean mask indicating missing values.
3. ***.sum()*** : Method applied to the boolean mask to sum up missing values along each column.

Output :

```
news_url        230
source_domain   230
real              0
news              0
news_reply        0
dtype: int64
```

==========

df = dataset.copy()

1. Copy the dataset to a new variable named df
2. This ensures that changes made to df do not affect the original dataset

==========

df['news_url'] = df['news_url'].fillna('')

Fill missing values in the 'news_url' column with empty strings

==========

df['source_domain'] = df['source_domain'].fillna('')

Fill missing values in the 'source_domain' column with empty strings

==========

missing_values_count = df.isnull().sum()

Calculate the number of missing values in each column of the DataFrame

==========

data_shape = df.shape

Get the shape of the DataFrame (number of rows and columns)

Output :

(16999, 5)

==========

df.isnull().sum()

Calculate the number of missing values in each column of the DataFrame

Output :

```
news_url         0
source_domain    0
real             0
news             0
news_reply       0
```

==========

features_dropped = ['news_url']

Define a list of features to be dropped from the DataFrame

df = df.drop(features_dropped, axis =1)

Drop the specified features from the DataFrame along the columns axis

==========

```
ps = PorterStemmer()
def wordopt(text):
    text = re.sub('[^a-zA-Z]', ' ',text)
    text = text.lower()
    text = text.split()
    text = [ps.stem(word) for word in text if not word in
stopwords.words('english')]
    text = ' '.join(text)
    return text
```

- Initialize Porter Stemmer.
- Define a function named `wordopt` which preprocesses text by:
- Removing non-alphabetic characters and replacing them with a space.
- Converting text to lowercase.
- Tokenizing text into words.
- Applying stemming using Porter Stemmer and removing stopwords.
- Joining the processed words back into a single string.
- The function takes a text input and returns the preprocessed text.

==========

```
df['news'] = df['news'].apply(wordopt)
```

Apply the wordopt function to preprocess the 'news' column of the DataFrame

==========

```
X = df['news']
Y = df['real']
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.20)
```

1. Assign the 'news' column from the DataFrame as the feature variable X.
2. Assign the 'real' column from the DataFrame as the target variable Y.
3. Split the data into training and test sets with a test size of 20% using train_test_split function.
4. x_train and y_train represent the feature and target variables for the training set respectively.
5. x_test and y_test represent the feature and target variables for the test set respectively.

==========

```
vectorization = TfidfVectorizer()
xv_train = vectorization.fit_transform(x_train)
xv_test = vectorization.transform(x_test)
```

1. Initialize a TfidfVectorizer object for text vectorization.
2. Fit and transform the training text data (x_train) using the vectorization object to convert text into TF-IDF vectors.
3. Transform the test text data (x_test) using the fitted vectorization object to maintain consistency between training and test data.
4. xv_train represents the TF-IDF vectors of the training text data.
5. xv_test represents the TF-IDF vectors of the test text data.

==========

```
LR_model = LogisticRegression()
LR_model.fit(xv_train,y_train)
lr_y_pred = LR_model.predict(xv_test)
score = accuracy_score(y_test,lr_y_pred)
print('Accuracy of LR model is ', score)
```

1. Initialize a Logistic Regression model.
2. Fit the training data (xv_train) and corresponding labels (y_train) to the model.
3. Predict the labels for the test data (xv_test) using the trained model.
4. Calculate the accuracy of the Logistic Regression model by comparing the predicted labels with the actual labels from the test set.
5. Print the accuracy score of the Logistic Regression model.

Accuracy of LR model is : 0.8297058823529412

==========

```
print (confusion_matrix(y_test,lr_y_pred))
print (classification_report(y_test,lr_y_pred))
```

1. Calculate the confusion matrix for Logistic Regression:
   - The confusion matrix is a table that summarizes the performance of a classification model. It presents the counts of true positive, false positive, true negative, and false negative predictions made by the model on the test data.

2. Calculate the classification report for Logistic Regression:
   - The classification report provides a comprehensive evaluation of the classification model's performance. It includes metrics such as precision, recall, F1-score, and support for each class, along with the overall accuracy.

Output :

```
[[ 353  485]
 [  94 2468]]
              precision    recall  f1-score   support

           0       0.79      0.42      0.55       838
           1       0.84      0.96      0.90      2562

    accuracy                           0.83      3400
   macro avg       0.81      0.69      0.72      3400
weighted avg       0.82      0.83      0.81      3400
```

==========

```
model_logistic = LogisticRegression()
model_logistic.fit(xv_train, y_train)
y_pred_logistic = model_logistic.decision_function(xv_test)
```

1. Create a Logistic Regression model object.
2. Fit the model to the training data (xv_train) and corresponding labels (y_train).
3. Use the trained model to predict the decision function scores for the test data (xv_test). The decision_function method returns the confidence scores (or decision function values) for each sample, indicating how likely they belong to each class. These scores can then be used to make predictions or perform further analysis.

==========

```
logistic_fpr, logistic_tpr, threshold = roc_curve(y_test, y_pred_logistic)
auc_logistic = auc(logistic_fpr, logistic_tpr)
plt.figure(figsize=(50,50), dpi=100)
plt.plot(logistic_fpr, logistic_tpr, marker='.', label='Logistic (auc = %0.3f)' % auc_logistic)
plt.xlabel('False Positive Rate -->')
plt.ylabel('True Positive Rate -->')
plt.legend()
plt.show()
```

1. Calculate the Receiver Operating Characteristic (ROC) curve for Logistic Regression :

- ROC curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. It plots the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.

- The roc_curve function computes the ROC curve from the true labels (y_test) and the predicted decision function scores (y_pred_logistic) of the logistic regression model.

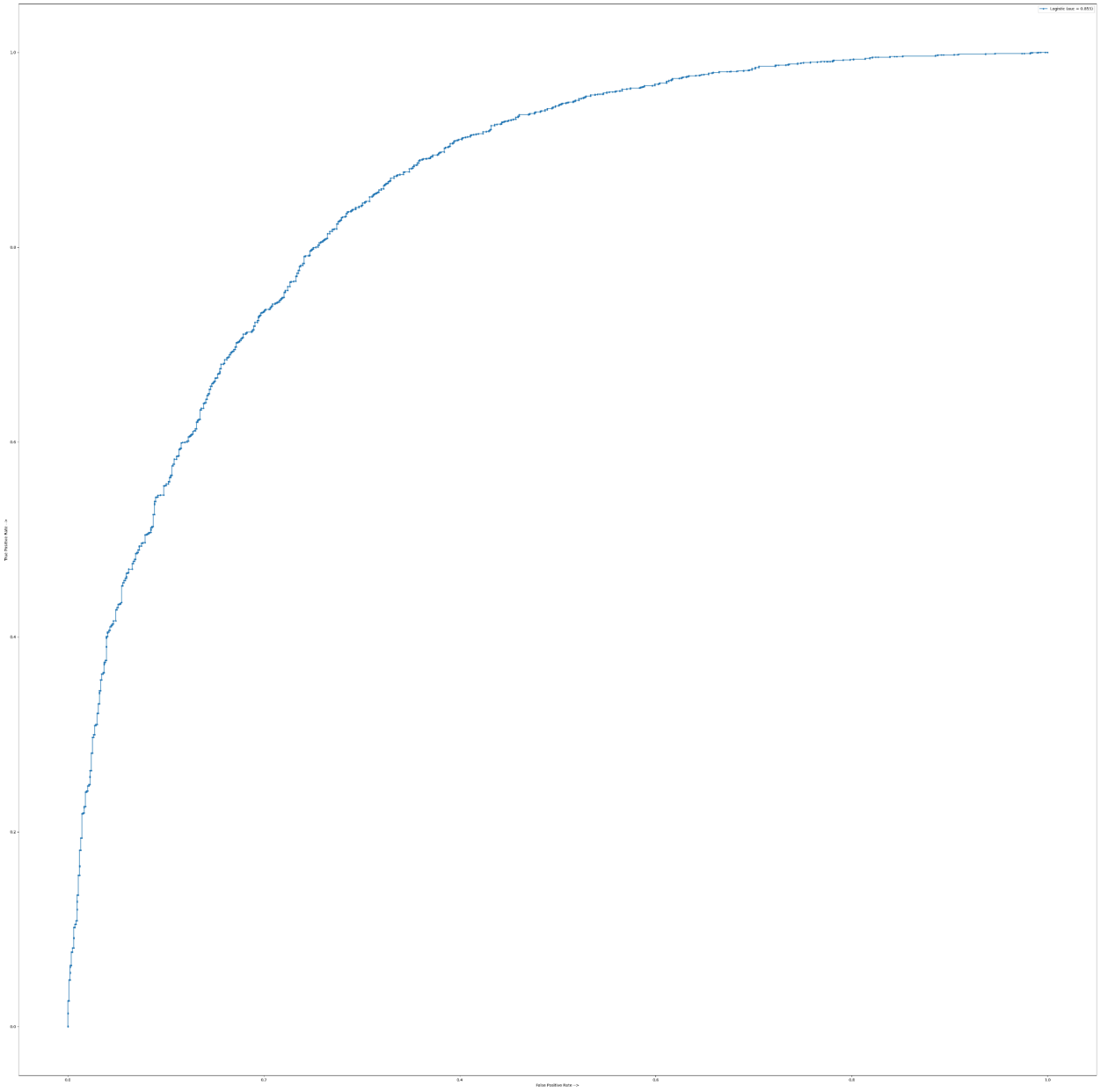
2.Calculate the Area Under the Curve (AUC) for Logistic Regression :

- AUC measures the entire two-dimensional area underneath the ROC curve. It provides an aggregate measure of performance across all possible classification thresholds.

- The auc function computes the AUC value from the false positive rate (logistic_fpr) and true positive rate (logistic_tpr) obtained from the ROC curve.

3. Plot the ROC curve :

- The plt.plot function is used to plot the ROC curve for Logistic Regression.
- The figure size and DPI settings are adjusted to ensure clear visualization.
- The xlabel and ylabel functions are used to label the x-axis and y-axis respectively.
- The legend function is used to display the label of the ROC curve along with the calculated AUC value.
- Finally, the plt.show function is called to display the plot.



==========

```
svm_model = SVC(kernel='linear')
svm_model.fit(xv_train,y_train)
svm_y_pred = svm_model.predict(xv_test)
score = accuracy_score(y_test,svm_y_pred)
print('Accuracy of SVM model is ', score)
```

1. Initialize a Support Vector Machine (SVM) model with a linear kernel:
 - SVM is a supervised learning algorithm used for classification and regression tasks. The 'linear' kernel specifies that the decision boundary will be linear.

2. Fit the training data (xv_train) and corresponding labels (y_train) to the SVM model:
 - The fit method trains the SVM model using the provided training data.

3. Predict the labels for the test data (xv_test) using the trained SVM model:
 - The predict method generates predictions for the test data based on the learned model.

4. Calculate the accuracy score of the SVM model:
 - The accuracy_score function calculates the accuracy of the SVM model by comparing the predicted labels (svm_y_pred) with the actual labels from the test set (y_test).
 - The accuracy score represents the proportion of correctly classified instances among all instances.
 - Finally, the accuracy score is printed to the console.

Accuracy of SVM model is : 0.8409411764705882

==========

```
model_SVC = SVC(kernel = 'rbf', random_state = 4)
model_SVC.fit(xv_train, y_train)
y_pred_svm = model_SVC.decision_function(xv_test)
```

1. Model Initialization: An SVC model is initialized with the parameters specified:
 - kernel = 'rbf': Specifies the radial basis function kernel for non-linear classification.
 - random_state = 4: Sets the random seed for reproducibility.

2. Model Training: The initialized SVC model is trained on the training data, represented by 'xv_train' (feature vectors) and 'y_train' (target labels), using the 'fit()' method.

3. Prediction Generation: The trained model is used to generate predictions for the test data, represented by 'xv_test', using the 'decision_function()' method. This method returns the signed distance of the samples to the separating hyperplane, which is then used to make predictions.

==========

```
svm_fpr, svm_tpr, threshold = roc_curve(y_test, y_pred_svm)
auc_svm = auc(svm_fpr, svm_tpr)

plt.figure(figsize=(50,50), dpi=100)
plt.plot(svm_fpr, svm_tpr, linestyle='-', label='SVM (auc = %0.3f)' % auc_svm)

plt.xlabel('False Positive Rate -->')
plt.ylabel('True Positive Rate -->')

plt.legend()

plt.show()
```

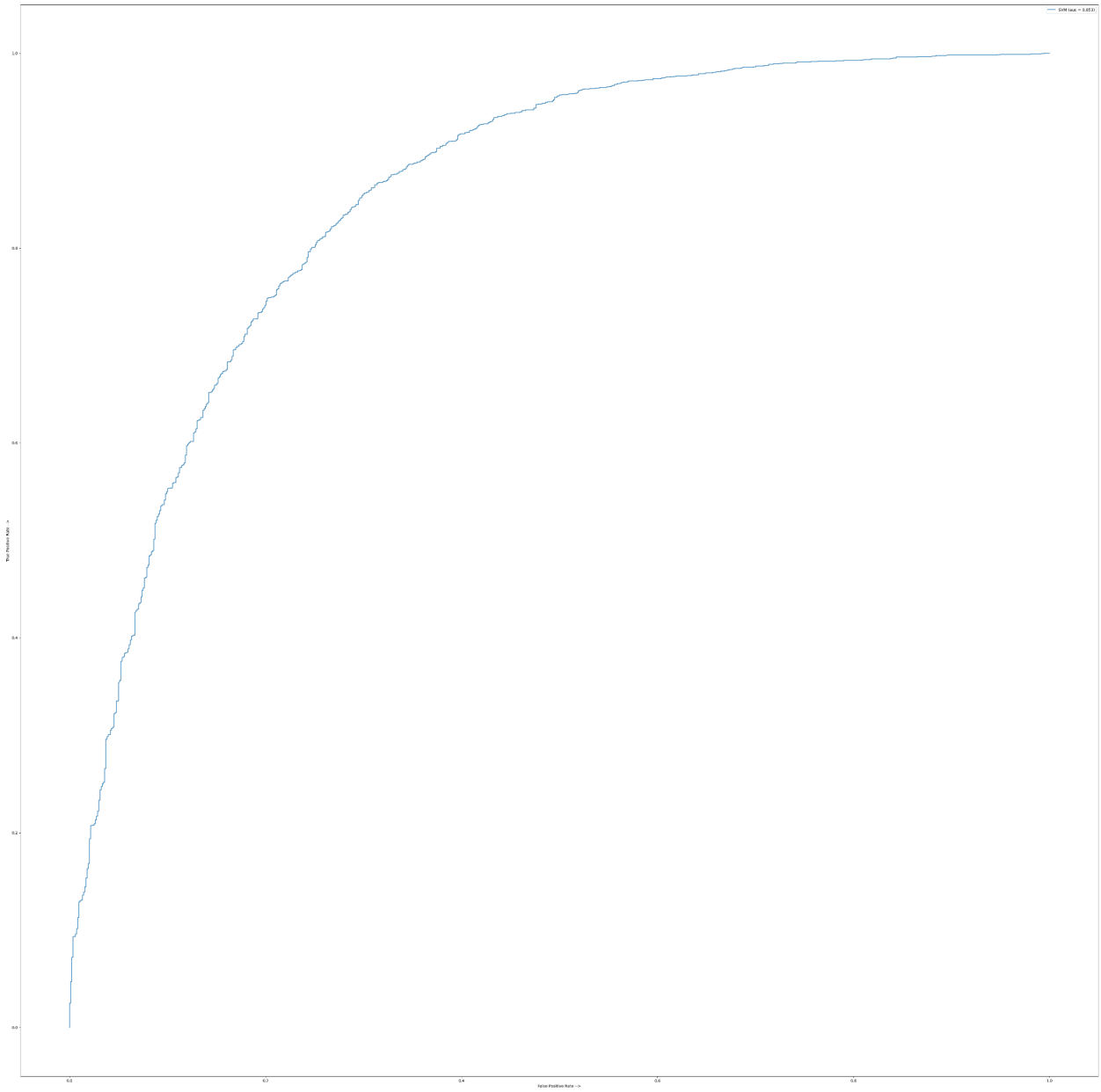
1. ROC Curve Calculation: The 'roc_curve()' function calculates the ROC curve by taking the true labels ('y_test') and predicted scores ('y_pred_svm') as input. It returns the false positive rates ('svm_fpr'), true positive rates ('svm_tpr'), and corresponding thresholds.

2. AUC Calculation: The 'auc()' function calculates the Area Under the Curve (AUC) for the ROC curve using the false positive rates ('svm_fpr') and true positive rates ('svm_tpr') obtained from the previous step.

3. Plotting the ROC Curve: The ROC curve along with its AUC value is plotted using the 'plt.plot()' function. The false positive rate ('svm_fpr') is plotted on the x-axis, and the true positive rate ('svm_tpr') is plotted on the y-axis. The linestyle is set to '-' for a solid line, and the label includes the AUC value.

4. Setting Labels and Legend: The x-axis label is set to 'False Positive Rate -->', and the y-axis label is set to 'True Positive Rate -->'. A legend is added to the plot to indicate the AUC value for the SVM model.

5. Displaying the Plot : The plot is displayed using the 'plt.show()' function.


SVM (auc = 0.853)
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate -->
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate -->

==========

```
print (confusion_matrix(y_test,svm_y_pred))
print (classification_report(y_test,svm_y_pred))
```

1. Confusion Matrix Calculation : The 'confusion_matrix()' function computes the confusion matrix by taking the true labels ('y_test') and predicted labels ('svm_y_pred') as input. It returns a 2x2 array representing the counts of true positive, true negative, false positive, and false negative predictions.

2. Displaying Confusion Matrix : The confusion matrix is printed using the 'print()' function.

3. Classification Report Generation : The 'classification_report()' function generates a text report comprising precision, recall, F1-score, and support for each class based on the true labels ('y_test') and predicted labels ('svm_y_pred').

4. Displaying Classification Report : The classification report is printed using the 'print()' function.

==========

```
RFC_model = RandomForestClassifier(random_state=0)

RFC_model.fit(xv_train, y_train)

rfc_y_pred = RFC_model.predict(xv_test)

score = accuracy_score(y_test,rfc_y_pred)
print('Accuracy of RFC model is ', score)
```

1. Random Forest Classifier Initialization : An RFC model is initialized with the parameter:
   - random_state=0: Sets the random seed for reproducibility.

2. Model Training : The initialized RFC model is trained on the training data ('xv_train' and 'y_train') using the 'fit()' method.

3. Prediction Generation : The trained model is used to predict the labels for the test data ('xv_test') using the 'predict()' method. The predicted labels are stored in the variable 'rfc_y_pred'.

4. Accuracy Calculation : The accuracy score of the RFC model is calculated by comparing the true labels ('y_test') with the predicted labels ('rfc_y_pred') using the 'accuracy_score()' function.

5. Displaying Accuracy : The accuracy score of the RFC model is printed using the 'print()' function.

Accuracy of RFC model is : 0.8279411764705882

==========

```
print (confusion_matrix(y_test,rfc_y_pred))
print (' ')
print (classification_report(y_test,rfc_y_pred))
```

1. Confusion Matrix Calculation: The 'confusion_matrix()' function computes the confusion matrix by taking the true labels ('y_test') and predicted labels ('rfc_y_pred') as input. It returns a 2x2 array representing the counts of true positive, true negative, false positive, and false negative predictions.

2. Displaying Confusion Matrix : The confusion matrix is printed using the 'print()' function.

3. Blank Line: A blank line is printed using the 'print()' function to improve readability in the output.

4. Classification Report Generation: The 'classification_report()' function generates a text report comprising precision, recall, F1-score, and support for each class based on the true labels ('y_test') and predicted labels ('rfc_y_pred').

5. Displaying Classification Report : The classification report is printed using the 'print()' function.

Output :

```
[[ 437  426]
 [ 149 2388]]
```

	precision	recall	f1-score	support
0	0.75	0.51	0.60	863
1	0.85	0.94	0.89	2537
accuracy			0.83	3400
macro avg	0.80	0.72	0.75	3400
weighted avg	0.82	0.83	0.82	3400

==========

```
RFC_fpr, RFC_tpr, threshold = roc_curve(y_test, rfc_y_pred)
auc_RFC = auc(RFC_fpr, RFC_tpr)

plt.figure(figsize=(40,40), dpi=100)
plt.plot(RFC_fpr, RFC_tpr , linestyle='-', label='RFC (auc = %0.3f)' % auc_RFC)

plt.xlabel('False Positive Rate -->')
plt.ylabel('True Positive Rate -->')

plt.legend()

plt.show()
```

1. **ROC Curve Calculation : The 'roc_curve()' function calculates the ROC curve by taking the true labels ('y_test') and predicted labels ('rfc_y_pred') as input. It returns the false positive rates ('RFC_fpr'), true positive rates ('RFC_tpr'), and corresponding thresholds.**

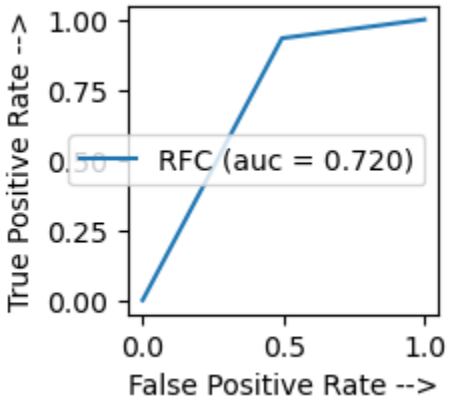
2. **AUC Calculation : The 'auc()' function calculates the Area Under the Curve (AUC) for the ROC curve using the false positive rates ('RFC_fpr') and true positive rates ('RFC_tpr') obtained from the previous step.**

3. Plotting the ROC Curve : The ROC curve along with its AUC value is plotted using the 'plt.plot()' function. The false positive rate ('RFC_fpr') is plotted on the x-axis, and the true positive rate ('RFC_tpr') is plotted on the y-axis. The linestyle is set to '-' for a solid line, and the label includes the AUC value.

4. Setting Labels and Legend : The x-axis label is set to 'False Positive Rate -->', and the y-axis label is set to 'True Positive Rate -->'. A legend is added to the plot to indicate the AUC value for the RFC model.

5. Displaying the Plot : The plot is displayed using the 'plt.show()' function.

Output :



==========

```
def fake_news_det(news):
    input_data = {"text":[news]}
    new_def_test = pd.DataFrame(input_data)
    new_def_test["text"] = new_def_test["text"].apply(wordopt)
    new_x_test = new_def_test["text"]
    print(new_x_test)
    vectorized_input_data = vectorization.transform(new_x_test)
    prediction = svm_model.predict(vectorized_input_data)

    if prediction == 1:
        print("Not a Fake News")
    else:
        print("Fake News")
```

1. Data Preparation : The input news article is converted into a DataFrame named 'new_def_test' with a single column 'text' containing the input news article.

2. Text Preprocessing: The 'wordopt' function (not provided) is applied to the 'text' column of the DataFrame to preprocess the text data. This function likely performs tasks such as tokenization, removing stopwords, and stemming.

3. Feature Vectorization: The preprocessed text data is transformed into numerical feature vectors using the 'vectorization' object's 'transform()' method. This step ensures that the input data is compatible with the machine learning model.

4. Prediction: The transformed feature vectors are passed to the 'svm_model' for prediction. The model predicts whether the input news article is fake (0) or not fake (1).

5. Output: Based on the prediction, the function prints either "Not a Fake News" or "Fake News" to indicate the classification result.

==========

Output of the model :

```
fake_news_det('Why All Ladies Crush on Angelina Jolie')
```

```
Fake News
```

```
fake_news_det('List of Who Beyonce Knowles Has Dated')
```

```
Not a Fake News
```

==========

Conclusion :

- The SVM model performs slightly better in terms of accuracy compared to Logistic Regression and RFC models.
- The ROC curves indicate good performance for all models, with AUC values close to 1.
- The function 'fake_news_det(news)' allows users to input a news article and determine whether it's fake or not using the trained SVM model.

Overall, the project demonstrates a comprehensive approach to fake news detection using machine learning techniques, offering a reliable tool for identifying potentially misleading information in news articles.

==========

Thank you for exploring our pattern recognition code! We hope this documentation has provided clarity on how to use and understand our implementation. Should you have any questions or feedback, feel free to reach out to our team. Happy coding!