Project Report: Data Analysis and Machine Learning Model for Fake News Detection

This project, based on a Kaggle dataset, involves the analysis and classification of news articles to determine their authenticity using a combination of data manipulation, text processing, and machine learning techniques. The necessary libraries include pandas for data manipulation, matplotlib.pyplot for visualizations, and re for text processing with regular expressions. Key modules from Scikit-learn are utilized, such as train_test_split and cross_val_score for model selection, TfidfVectorizer for feature extraction, and machine learning algorithms including LogisticRegression, RandomForestClassifier, and KNeighborsClassifier. The evaluation of these models is performed using metrics like accuracy_score, confusion_matrix, classification_report, roc_curve, and auc to ensure a comprehensive assessment of the models' performance. This setup facilitates the creation of a robust fake news detection system by leveraging standard tools and methodologies for text classification.

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import re

from sklearn.model_selection import train_test_split,cross_val_score

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy_score,confusion_matrix, classification_report,roc_curve, auc
```

This code block imports additional libraries necessary for various aspects of data analysis and visualization. The numpy library is imported as np for efficient numerical operations and handling of arrays. The seaborn library, imported as sns, is used for creating enhanced statistical visualizations and plots. The string module is imported to provide a collection of string operations, particularly useful for text processing tasks. These libraries complement the data manipulation and machine learning processes by enabling advanced numerical computations, visualizations, and text handling.

In [2]:
```
import numpy as np
import seaborn as sns
import string
```

The code below reads two CSV files, "Fake.csv" and "True.csv", from the specified file paths using the pd.read_csv() function provided by the pandas library. The data from these files is loaded into two pandas DataFrame objects named df_fake and df_true, respectively. These DataFrames are expected to contain data related to fake news articles and true news articles, which will be used for analysis and classification in the project.

In [3]:
```
df_fake = pd.read_csv("C:/Users/Collins PC/Downloads/Fake News Detection Project/Fake.csv")
df_true = pd.read_csv("C:/Users/Collins PC/Downloads/Fake News Detection Project/True.csv")
```

This part returns the first five entries of the fake news articles to get a quick glimpse of the data I am working with

In [4]:
```
df_fake.head()
```

Out[4]:

	title	text	subject	date
0	Donald Trump Sends Out Embarrassing New Year'...	Donald Trump just couldn t wish all Americans ...	News	December 31, 2017
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	December 30, 2017
3	Trump Is So Obsessed He Even Has Obama's Name...	On Christmas day, Donald Trump announced that ...	News	December 29, 2017
4	Pope Francis Just Called Out Donald Trump Dur...	Pope Francis used his annual Christmas Day mes...	News	December 25, 2017

This part returns the first five entries of the true or real news articles to get a quick understanding of the data I am working with

In [5]:
```
df_true.head()
```

Out[5]:

	title	text	subject	date
0	As U.S. budget fight looms, Republicans flip t...	WASHINGTON (Reuters) - The head of a conservat...	politicsNews	December 31, 2017
1	U.S. military to accept transgender recruits o...	WASHINGTON (Reuters) - Transgender people will...	politicsNews	December 29, 2017
2	Senior U.S. Republican senator: 'Let Mr. Muell...	WASHINGTON (Reuters) - The special counsel inv...	politicsNews	December 31, 2017
3	FBI Russia probe helped by Australian diplomat...	WASHINGTON (Reuters) - Trump campaign adviser ...	politicsNews	December 30, 2017
4	Trump wants Postal Service to charge 'much mor...	SEATTLE/WASHINGTON (Reuters) - President Donal...	politicsNews	December 29, 2017

The code below adds a new column named "class" to both DataFrames, df_fake and df_true, and assigns a value of 0 to all entries in the "class" column of df_fake and a value of 1 to all entries in the "class" column of df_true. This step is crucial for supervised machine learning tasks where we need to define a target variable (in this case, the authenticity of the news articles) that the models will learn to predict based on the features of the data. By labeling the data in this manner, we establish the ground truth for classification, with "0" representing fake news and "1" representing true news.

In [6]:
```
#Inserting the column class as a target feature
df_fake["class"] = 0
df_true["class"] = 1
```

The code below provides the dimensions (number of rows and columns) of the DataFrames df_fake and df_true using the shape attribute. The output consists of two tuples, where the first element represents the number of rows and the second element represents the number of columns in each DataFrame. This information is useful for understanding the size of the datasets and ensuring data integrity throughout the analysis and modeling processes.

In [7]:
```
df_fake.shape, df_true.shape
```

Out[7]:
```
((23481, 5), (21417, 5))
```

The code below displays concise summary information about the DataFrames df_true and df_fake using the info() method in pandas. The summary includes the total number of entries in each column, the data type of each column, and the number of non-null values in each column. This summary helps in understanding the structure of the dataset, identifying any missing values, and assessing the data types of the features, which is crucial for data preprocessing and analysis.

In [8]:
```
df_true.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21417 entries, 0 to 21416
Data columns (total 5 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   title    21417 non-null  object
 1   text     21417 non-null  object
 2   subject  21417 non-null  object
 3   date     21417 non-null  object
 4   class    21417 non-null  int64 
dtypes: int64(1), object(4)
memory usage: 836.7+ KB
```

In [9]:
```
df_fake.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23481 entries, 0 to 23480
Data columns (total 5 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   title    23481 non-null  object
 1   text     23481 non-null  object
 2   subject  23481 non-null  object
 3   date     23481 non-null  object
 4   class    23481 non-null  int64 
dtypes: int64(1), object(4)
memory usage: 917.4+ KB
```

The code snippet below removes the last 10 rows from the DataFrames df_fake and df_true for manual testing purposes. It creates new DataFrames named df_fake_manual_testing and df_true_manual_testing containing these last 10 rows, which will be used for manual evaluation of the trained models.

The removal is done by iterating over a range of index values corresponding to the rows to be removed, and using the drop() method with the axis=0 parameter to drop rows based on their index. This process ensures that the original DataFrames are modified accordingly, removing the specified rows from the dataset.

However, it's worth noting that directly modifying a DataFrame within a loop can sometimes lead to unexpected behavior due to indexing changes. It's advisable to use caution when modifying DataFrames in place, especially within loops. Alternatively, you can create a copy of the DataFrame and modify the copy to avoid modifying the original data.

In [10]:
```
# Removing last 10 rows for manual testing
df_fake_manual_testing = df_fake.tail(10)
for i in range(23480,23470,-1):
    df_fake.drop([i], axis = 0, inplace = True)


df_true_manual_testing = df_true.tail(10)
for i in range(21416,21406,-1):
    df_true.drop([i], axis = 0, inplace = True)
```

In [11]:
```
df_fake.shape, df_true.shape
```

Out[11]:
```
((23471, 5), (21407, 5))
```

The code below labels the manual testing data by assigning the value "0" to the "class" column of the DataFrame df_fake_manual_testing and the value "1" to the "class" column of the DataFrame df_true_manual_testing. This process is essential for establishing the ground truth, indicating whether each news article in the manual testing datasets is fake or true. By using the .loc[] accessor to select all rows and the "class" column, followed by the assignment operation, we ensure that each news article is correctly labeled according to its authenticity. These class labels serve as reference points for evaluating the performance of machine learning models during manual testing, providing valuable insights into the models' ability to distinguish between fake and true news articles.

In [12]:
```
df_fake_manual_testing.loc[:, "class"] = 0
df_true_manual_testing.loc[:, "class"] = 1
```

In [13]:
```
df_fake_manual_testing.head(10)
```

Out[13]:

	title	text	subject	date	class
23471	Seven Iranians freed in the prisoner swap have...	21st Century Wire says This week, the historic...	Middle-east	January 20, 2016	0
23472	#Hashtag Hell & The Fake Left	By Dady Chery and Gilbert MercierAll writers ...	Middle-east	January 19, 2016	0
23473	Astroturfing: Journalist Reveals Brainwashing ...	Vic Bishop Waking TimesOur reality is carefull...	Middle-east	January 19, 2016	0
23474	The New American Century: An Era of Fraud	Paul Craig RobertsIn the last years of the 20t...	Middle-east	January 19, 2016	0
23475	Hillary Clinton: 'Israel First' (and no peace ...	Robert Fantina CounterpunchAlthough the United...	Middle-east	January 18, 2016	0
23476	McPain: John McCain Furious That Iran Treated ...	21st Century Wire says As 21WIRE reported earl...	Middle-east	January 16, 2016	0
23477	JUSTICE? Yahoo Settles E-mail Privacy Class-ac...	21st Century Wire says It s a familiar theme. ...	Middle-east	January 16, 2016	0
23478	Sunnistan: US and Allied 'Safe Zone' Plan to T...	Patrick Henningsen 21st Century WireRemember ...	Middle-east	January 15, 2016	0
23479	How to Blow $700 Million: Al Jazeera America F...	21st Century Wire says Al Jazeera America will...	Middle-east	January 14, 2016	0
23480	10 U.S. Navy Sailors Held by Iranian Military ...	21st Century Wire says As 21WIRE predicted in ...	Middle-east	January 12, 2016	0

In [14]:
```
df_true_manual_testing.head(10)
```

Out[14]:

	title	text	subject	date	class
21407	Mata Pires, owner of embattled Brazil builder ...	SAO PAULO (Reuters) - Cesar Mata Pires, the ow...	worldnews	August 22, 2017	1
21408	U.S., North Korea clash at U.N. forum over nuc...	GENEVA (Reuters) - North Korea and the United ...	worldnews	August 22, 2017	1
21409	U.S., North Korea clash at U.N. arms forum on ...	GENEVA (Reuters) - North Korea and the United ...	worldnews	August 22, 2017	1
21410	Headless torso could belong to submarine journ...	COPENHAGEN (Reuters) - Danish police said on T...	worldnews	August 22, 2017	1
21411	North Korea shipments to Syria chemical arms a...	UNITED NATIONS (Reuters) - Two North Korean sh...	worldnews	August 21, 2017	1
21412	'Fully committed' NATO backs new U.S. approach...	BRUSSELS (Reuters) - NATO allies on Tuesday we...	worldnews	August 22, 2017	1
21413	LexisNexis withdrew two products from Chinese ...	LONDON (Reuters) - LexisNexis, a provider of l...	worldnews	August 22, 2017	1
21414	Minsk cultural hub becomes haven from authorities	MINSK (Reuters) - In the shadow of disused Sov...	worldnews	August 22, 2017	1
21415	Vatican upbeat on possibility of Pope Francis ...	MOSCOW (Reuters) - Vatican Secretary of State ...	worldnews	August 22, 2017	1
21416	Indonesia to buy $1.14 billion worth of Russia...	JAKARTA (Reuters) - Indonesia will buy 11 Sukh...	worldnews	August 22, 2017	1

The code below concatenates the DataFrames df_fake_manual_testing and df_true_manual_testing along the rows (axis=0) to create a single DataFrame named df_manual_testing. This combined DataFrame contains the manual testing data for both fake and true news articles.

Subsequently, the to_csv() method is used to export the df_manual_testing DataFrame to a CSV file named "manual_testing.csv". This file will contain the manual testing data, which can be used for further analysis or evaluation purposes.

By saving the manual testing data to a CSV file, it becomes easily accessible for future reference and can be seamlessly integrated into the overall workflow of the project.

In [15]:
```
df_manual_testing = pd.concat([df_fake_manual_testing,df_true_manual_testing], axis = 0)
df_manual_testing.to_csv("manual_testing.csv")
```

The code below concatenates the DataFrames df_fake and df_true along the rows to create a single DataFrame named df_merge. The resulting DataFrame contains both fake and true news articles combined.

By using the head(10) method, the code then displays the first 10 rows of the merged DataFrame df_merge. This provides a quick overview of the combined dataset, showing the structure and content of the data.

Concatenating the two DataFrames allows for a unified dataset that can be used for further analysis, feature engineering, or model training in the fake news detection project.

In [16]:
```
df_merge = pd.concat([df_fake, df_true], axis =0 )
df_merge.head(10)
```

Out[16]:

	title	text	subject	date	class
0	Donald Trump Sends Out Embarrassing New Year'...	Donald Trump just couldn t wish all Americans ...	News	December 31, 2017	0
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017	0
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	December 30, 2017	0
3	Trump Is So Obsessed He Even Has Obama's Name...	On Christmas day, Donald Trump announced that ...	News	December 29, 2017	0
4	Pope Francis Just Called Out Donald Trump Dur...	Pope Francis used his annual Christmas Day mes...	News	December 25, 2017	0
5	Racist Alabama Cops Brutalize Black Boy While...	The number of cases of cops brutalizing and ki...	News	December 25, 2017	0
6	Fresh Off The Golf Course, Trump Lashes Out A...	Donald Trump spent a good portion of his day a...	News	December 23, 2017	0
7	Trump Said Some INSANELY Racist Stuff Inside ...	In the wake of yet another court decision that...	News	December 23, 2017	0
8	Former CIA Director Slams Trump Over UN Bully...	Many people have raised the alarm regarding th...	News	December 22, 2017	0
9	WATCH: Brand-New Pro-Trump Ad Features So Muc...	Just when you might have thought we d get a br...	News	December 21, 2017	0

In [17]:
```
df_merge.columns
```

Out[17]:
```
Index(['title', 'text', 'subject', 'date', 'class'], dtype='object')
```

In [18]:
```
df = df_merge.drop(["title", "subject","date"], axis = 1)
```

The code below calculates the sum of missing values (NaN) for each column in the DataFrame df using the isnull() method, which returns a DataFrame of Boolean values indicating whether each element is missing or not, and then the sum() method, which calculates the sum of True values (i.e., missing values) along each column.

The output provides the number of missing values for each column in the DataFrame, helping to identify any areas of missing data that may require further investigation or handling before proceeding with analysis or modeling tasks.

In [19]:
```
df.isnull().sum()
```

Out[19]:
```
text     0
class    0
dtype: int64
```

The code below randomly shuffles the rows of the DataFrame df using the sample() method with the frac parameter set to 1. By setting frac to 1, it means that the entire DataFrame will be sampled (i.e., all rows will be included), and since no replacement is specified, each row will be selected exactly once.

Shuffling the rows of the DataFrame can be useful for various purposes, such as randomizing the order of data for training machine learning models or ensuring that the data is not biased by any particular order. It helps in creating more robust and reliable models by preventing any patterns or biases that may exist due to the original order of the data.

In [20]:
```
df = df.sample(frac = 1)
```

In [21]:
```
df.head()
```

Out[21]:

	text	class
13297	So what do you think the transparent Hillary...	0
21137	The results of this town s gun ownership manda...	0
885	Proving that the biggest obstacle to his presi...	0
17982	Leftist agitators showed up tonight at the ope...	0
21835	The laws don t apply to the Clinton s they re ...	0

The code below resets the index of the DataFrame df to the default integer index and drops the original index column.

The reset_index() method resets the index of the DataFrame, and the inplace=True parameter ensures that the changes are applied to the DataFrame df directly. Then, the drop() method is used to remove the original index column, specified by the label "index", along the columns (axis=1). Again, inplace=True ensures that the changes are applied to the DataFrame df directly. This process ensures that the DataFrame df has a clean, sequential integer index without retaining any previous index information, which can be helpful for further analysis or modeling tasks.

In [22]:
```
df.reset_index(inplace = True)
df.drop(["index"], axis = 1, inplace = True)
```

In [23]:
```
df.columns
```

Out[23]:
```
Index(['text', 'class'], dtype='object')
```

In [24]:
```
df.head()
```

Out[24]:

	text	class
0	So what do you think the transparent Hillary...	0
1	The results of this town s gun ownership manda...	0
2	Proving that the biggest obstacle to his presi...	0
3	Leftist agitators showed up tonight at the ope...	0
4	The laws don t apply to the Clinton s they re ...	0

The wordopt function preprocesses text data by applying various cleaning operations. It first converts all text to lowercase for consistency. Then, it removes content within square brackets, non-word characters, URLs, HTML tags, punctuation, newline characters, and words containing numbers. This comprehensive cleaning process aims to standardize the text data and remove any irrelevant or noisy elements, preparing it for further analysis or modeling in natural language processing tasks.

In [25]:
```
def wordopt(text):
    text = text.lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub("\\W"," ",text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text
```

The below code snippet applies the wordopt function to the "text" column of the DataFrame df using the apply() method. By applying the wordopt function to each entry in the "text" column, it preprocesses the text data within the DataFrame. This preprocessing step ensures that the text data is cleaned and standardized according to the operations defined in the wordopt function, making it suitable for downstream analysis or modeling tasks in natural language processing.

In [26]:
```
df["text"] = df["text"].apply(wordopt)
```

The code below sets the X and y features in preparation for further processing using machine learning models

In [27]:
```
x = df["text"]
y = df["class"]
```

The code snippet below splits the dataset into two subsets: one for training the model and the other for testing its performance. It uses the train_test_split function from scikit-learn to achieve this, with the test_size parameter specifying the proportion of data to be allocated to the testing set (in this case, 25%). The resulting subsets, x_train, x_test, y_train, and y_test, correspond to the training and testing sets for both the input features (x) and the target variable (y). This split enables the evaluation of the model's ability to generalize to unseen data, aiding in the detection of potential overfitting issues.

In [28]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25)
```

The provided code below is part of a machine learning project focused on text data preprocessing and transformation using TF-IDF (Term Frequency-Inverse Document Frequency) vectorization. The goal is to convert textual data into a numerical format suitable for machine learning models.

In [29]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

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

In this segment of the machine learning project, a Logistic Regression model from the scikit-learn library is being utilized to perform supervised learning. The LogisticRegression object, denoted as LR, is first instantiated, representing the model to be trained. The fit method is then called with the previously transformed training data (xv_train) and the corresponding target labels (y_train). This step involves training the Logistic Regression model by finding the optimal weights that best map the TF-IDF features of the training data to their respective target labels, thus enabling the model to make predictions on new, unseen data.

In [30]:
```
from sklearn.linear_model import LogisticRegression

LR = LogisticRegression()
LR.fit(xv_train,y_train)
```

Out[30]:
```
LogisticRegression
LogisticRegression()
```

In [31]:
```
pred_lr=LR.predict(xv_test)
```

The score method is used to evaluate the performance of the trained Logistic Regression model by calculating its accuracy on the test data (xv_test) against the true labels (y_test).

In [32]:
```
LR.score(xv_test, y_test)
```

Out[32]:
```
0.9873440285204991
```

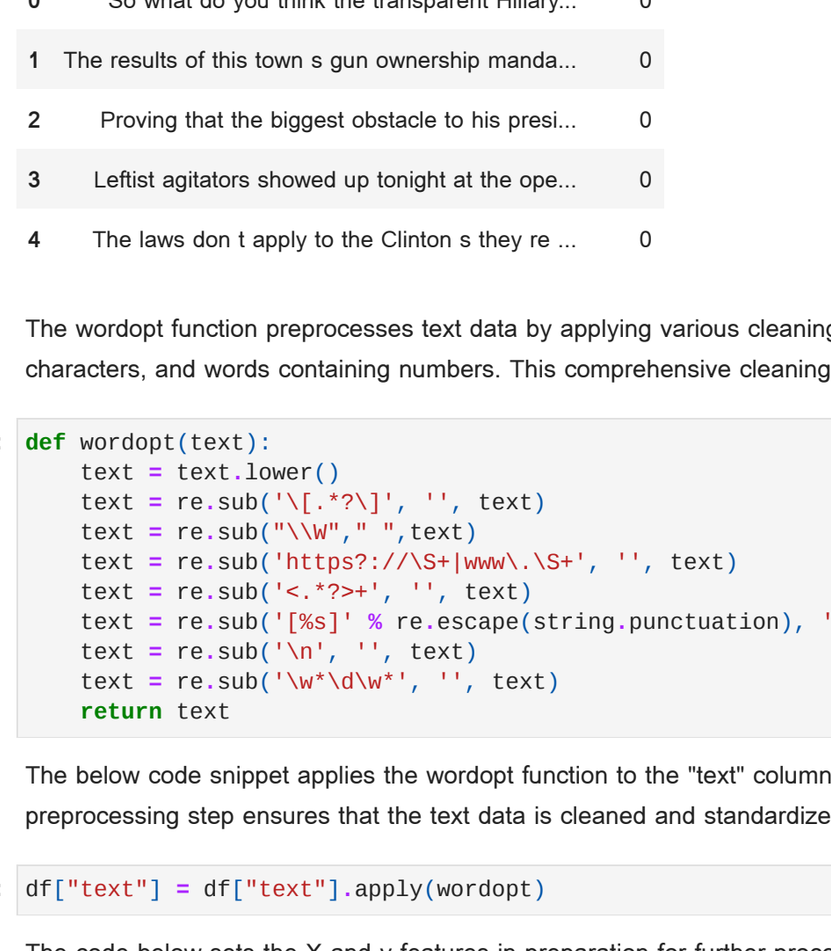
The print(classification_report(y_test, pred_lr)) statement generates and displays a detailed classification report, including precision, recall, F1-score, and support for each class, based on the true labels (y_test) and the predicted labels (pred_lr).

In [33]:
```
print(classification_report(y_test, pred_lr))
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      5891
           1       0.98      0.99      0.99      5329

    accuracy                           0.99     11220
   macro avg       0.99      0.99      0.99     11220
weighted avg       0.99      0.99      0.99     11220
```

The code below calculates and visualizes the Receiver Operating Characteristic (ROC) curve, a crucial metric for evaluating the performance of a binary classification model. By computing the false positive rate (FPR) and true positive rate (TPR) using the roc_curve function, and the area under the curve (AUC) with the auc function, the code assesses the model's ability to distinguish between the two classes. The ROC curve plot, generated with matplotlib, shows the trade-off between the FPR and TPR at various threshold settings, with the AUC value providing a single scalar to summarize the model's overall performance. An AUC closer to 1 indicates a better performing model.

In [34]:
```
#Compute ROC curve and ROC area
fpr, tpr, _ = roc_curve(y_test,pred_lr)
roc_auc = auc(fpr, tpr)

# Plot the ROC Curve
plt.figure()
lw = 2
plt.plot (fpr, tpr, color= "darkorange",
         lw=lw, label = "ROC curve (area = %0.2f)" %roc_auc)
plt.plot ([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc = "lower right")
plt.show()
```



In this part of the machine learning project, a Decision Tree Classifier from the scikit-learn library is being employed for supervised learning. A DecisionTreeClassifier object, denoted as DT, is instantiated and subsequently trained using the fit method with the TF-IDF transformed training data (xv_train) and the corresponding target labels (y_train). This process involves building a decision tree model that learns to map the input features to the target labels by recursively splitting the data based on feature values, thus enabling the model to make predictions on new data.

In [35]:
```
from sklearn.tree import DecisionTreeClassifier

DT = DecisionTreeClassifier()
DT.fit(xv_train, y_train)
```

Out[35]:
```
DecisionTreeClassifier
DecisionTreeClassifier()
```

In [36]:
```
pred_dt = DT.predict(xv_test)
```

In [37]:
```
DT.score(xv_test, y_test)
```

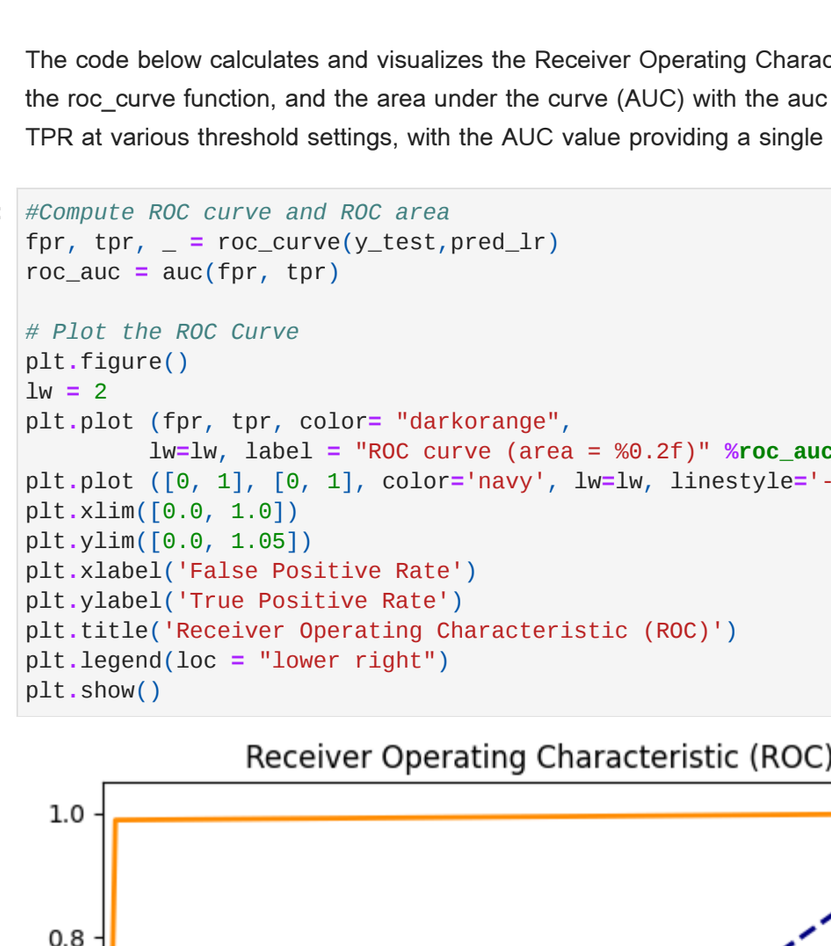
Out[37]:
```
0.995632798573975
```

In [38]:
```
print(classification_report(y_test, pred_dt))
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      5891
           1       1.00      0.99      1.00      5329

    accuracy                           1.00     11220
   macro avg       1.00      1.00      1.00     11220
weighted avg       1.00      1.00      1.00     11220
```

This code segment is used to compute and visualize the Receiver Operating Characteristic (ROC) curve for the predictions made by a Decision Tree Classifier. The false positive rate (FPR) and true positive rate (TPR) are calculated using the roc_curve function based on the true labels (y_test) and the predicted labels (pred_dt). The area under the curve (AUC) is then computed with the auc function, providing a summary metric of the classifier's performance. The ROC curve is plotted using matplotlib, with the FPR on the x-axis and the TPR on the y-axis, illustrating the trade-offs between sensitivity and specificity at various threshold levels. The AUC value is included in the plot's legend to indicate the overall discriminative ability of the classifier, with values closer to 1 indicating better performance.

In [44]:
```
#Compute ROC curve and ROC area
fpr, tpr, _ = roc_curve(y_test,pred_dt)
roc_auc = auc(fpr, tpr)

# Plot the ROC Curve
plt.figure()
lw = 2
plt.plot (fpr, tpr, color= "darkorange",
         lw=lw, label = "ROC curve (area = %0.2f)" %roc_auc)
plt.plot ([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc = "lower right")
plt.show()
```



In this part of the machine learning project, a Gradient Boosting Classifier from the scikit-learn library is employed for supervised learning. The GradientBoostingClassifier object, referred to as GBC, is instantiated with a specified random state for reproducibility. The fit method is then called with the TF-IDF transformed training data (xv_train) and the corresponding target labels (y_train). This process trains the Gradient Boosting model, which builds an ensemble of weak prediction models, typically decision trees, in a sequential manner. Each subsequent model focuses on correcting the errors made by the previous ones, thereby improving the overall predictive performance of the ensemble on the training data.

In [39]:
```
from sklearn.ensemble import GradientBoostingClassifier

GBC = GradientBoostingClassifier(random_state=0)
GBC.fit(xv_train, y_train)
```

Out[39]:
```
GradientBoostingClassifier
GradientBoostingClassifier(random_state=0)
```

In [45]:
```
pred_gbc = GBC.predict(xv_test)
```

In [46]:
```
GBC.score(xv_test, y_test)
```

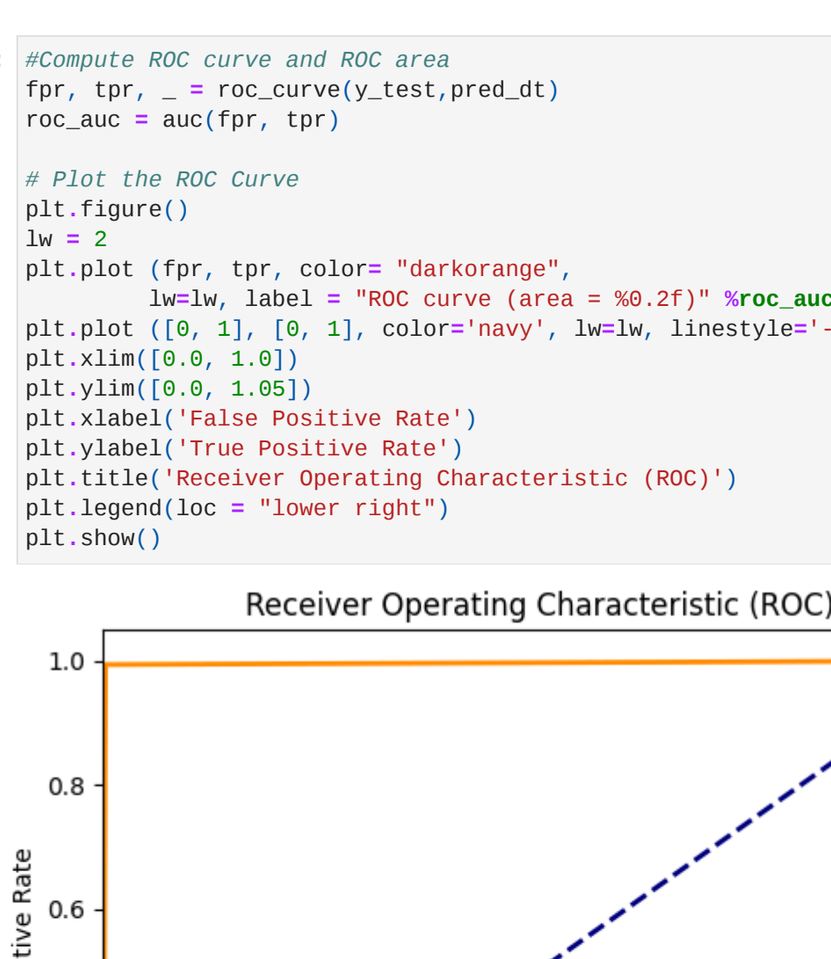
Out[46]:
```
0.9957219251336898
```

In [47]:
```
print(classification_report(y_test, pred_gbc))
```

```
              precision    recall  f1-score   support

           0       1.00      0.99      1.00      5891
           1       0.99      1.00      1.00      5329

    accuracy                           1.00     11220
   macro avg       1.00      1.00      1.00     11220
weighted avg       1.00      1.00      1.00     11220
```

This code segment is designed to compute and visualize the Receiver Operating Characteristic (ROC) curve for the predictions made by a Gradient Boosting Classifier. The roc_curve function is used to calculate the false positive rate (FPR) and true positive rate (TPR) based on the true labels (y_test) and the predicted labels (pred_gbc). The area under the curve (AUC) is then computed using the auc function, providing a single value summary of the model's overall performance, where a higher AUC indicates better discriminative ability. The ROC curve is plotted using matplotlib, with the FPR on the x-axis and the TPR on the y-axis, showing the trade-off between the true positive rate and false positive rate at various threshold levels. The plot includes a diagonal line representing a random classifier, and the ROC curve is annotated with the AUC value in the legend, helping to visually assess the model's performance.

In [49]:
```
#Compute ROC curve and ROC area
fpr, tpr, _ = roc_curve(y_test,pred_gbc)
roc_auc = auc(fpr, tpr)

# Plot the ROC Curve
plt.figure()
lw = 2
plt.plot (fpr, tpr, color= "darkorange",
         lw=lw, label = "ROC curve (area = %0.2f)" %roc_auc)
plt.plot ([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc = "lower right")
plt.show()
```



In this part of the machine learning project, a Random Forest Classifier from the scikit-learn library is employed for supervised learning. The RandomForestClassifier object, denoted as RFC, is instantiated with a specified random state to ensure reproducibility. The fit method is called with the TF-IDF transformed training data (xv_train) and the corresponding target labels (y_train). This process involves training the Random Forest model, which is an ensemble method that builds multiple decision trees and merges their results to improve predictive accuracy and control overfitting. Each tree is trained on a random subset of the data and features, and the final prediction is made by averaging the outputs (regression) or majority voting (classification) from all the individual trees.

In [48]:
```
from sklearn.ensemble import RandomForestClassifier

RFC = RandomForestClassifier(random_state=0)
RFC.fit(xv_train, y_train)
```

Out[48]:
```
RandomForestClassifier
RandomForestClassifier(random_state=0)
```

In [50]:
```
pred_rfc = RFC.predict(xv_test)
```

In [51]:
```
RFC.score(xv_test, y_test)
```

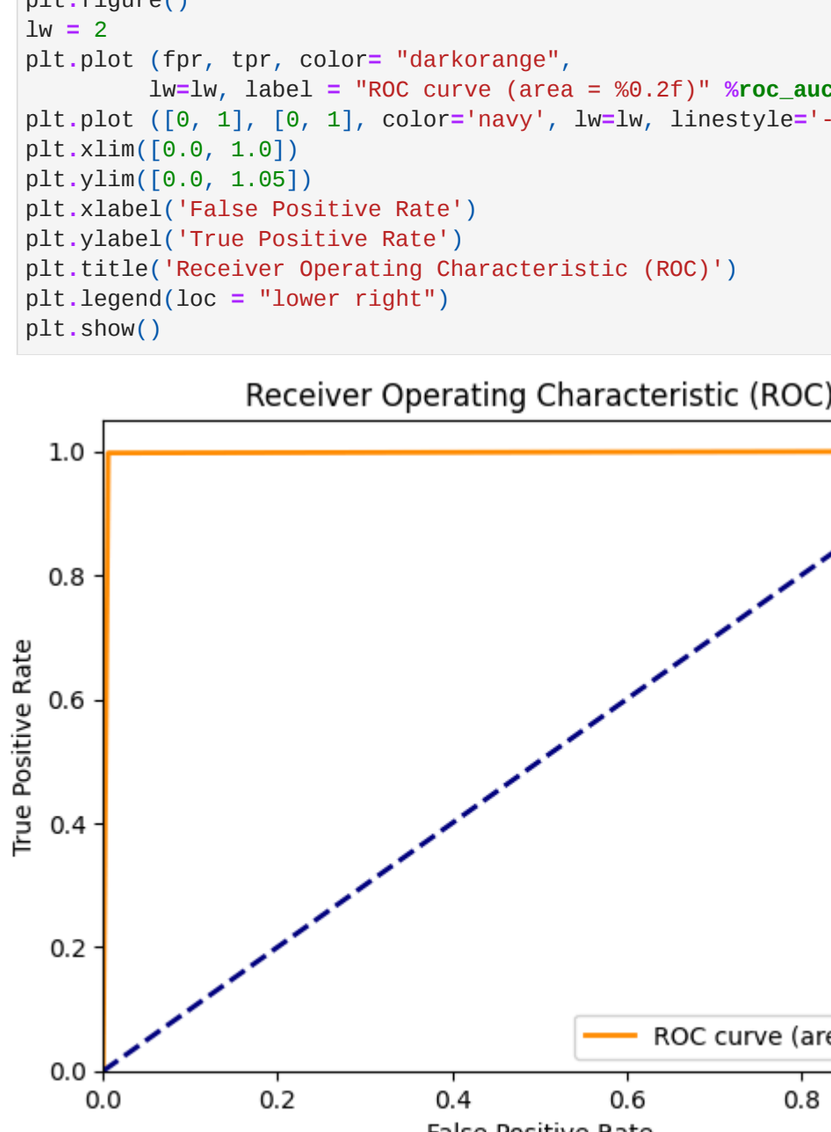
Out[51]:
```
0.9888591800356507
```

In [52]:
```
print(classification_report(y_test, pred_rfc))
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      5891
           1       0.99      0.98      0.99      5329

    accuracy                           0.99     11220
   macro avg       0.99      0.99      0.99     11220
weighted avg       0.99      0.99      0.99     11220
```

This code segment is aimed at computing and visualizing the Receiver Operating Characteristic (ROC) curve for the predictions made by a Random Forest Classifier. It utilizes the roc_curve function to calculate the false positive rate (FPR) and true positive rate (TPR) based on the true labels (y_test) and the predicted labels (pred_rfc). The area under the curve (AUC) is then computed using the auc function, serving as a summary metric for the classifier's overall performance. The ROC curve is plotted using matplotlib, with the FPR on the x-axis and the TPR on the y-axis, illustrating the trade-off between sensitivity and specificity at various threshold levels. The plot includes a diagonal line representing a random classifier, and the ROC curve is annotated with the AUC value in the legend, aiding in the visual assessment of the model's performance

In [53]:
```
#Compute ROC curve and ROC area
fpr, tpr, _ = roc_curve(y_test,pred_rfc)
roc_auc = auc(fpr, tpr)

# Plot the ROC Curve
plt.figure()
lw = 2
plt.plot (fpr, tpr, color= "darkorange",
         lw=lw, label = "ROC curve (area = %0.2f)" %roc_auc)
plt.plot ([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc = "lower right")
plt.show()
```



This function, manual_testing(news), facilitates manual testing of news articles by predicting their authenticity using trained machine learning models. It takes a news article as input, preprocesses it, and then uses the trained models (Logistic Regression, Decision Tree, Gradient Boosting, and Random Forest) to predict whether the news is fake or not. The function returns the predictions made by each model for the provided news article. The output_lable(n) function maps the model predictions (n) to meaningful labels indicating whether the news is classified as fake or not.

In [54]:
```
def output_lable(n):
    if n == 0:
        return "This News is Fake"
    elif n == 1:
        return "NOT Fake News"

def manual_testing(news):
    testing_news = {"text":[news]}
    new_def_test = pd.DataFrame(testing_news)
    new_def_test["text"] = new_def_test["text"].apply(wordopt)
    new_x_test = new_def_test["text"]
    new_xv_test = vectorization.transform(new_x_test)
    pred_LR = LR.predict(new_xv_test)
    pred_DT = DT.predict(new_xv_test)
    pred_GBC = GBC.predict(new_xv_test)
    pred_RFC = RFC.predict(new_xv_test)

    return print("\n\nLR Prediction: {} \nDT Prediction: {} \nGBC Prediction: {} \nRFC Prediction: {}".format(output_lable(pred_LR[0]),
                                                                                                              output_lable(pred_GBC[0]),
                                                                                                              output_lable(pred_RFC[0])))
```

Feel free to input a news article, and I'll run it through the trained models to predict its authenticity.

In [55]:
```
news = str(input())
manual_testing(news)
```

```
LR Prediction: NOT Fake News
DT Prediction: NOT Fake News
GBC Prediction: NOT Fake News
RFC Prediction: NOT Fake News
```

In [56]:
```
news = str(input())
manual_testing(news)
```

```
LR Prediction: This News is Fake
DT Prediction: This News is Fake
GBC Prediction: This News is Fake
RFC Prediction: This News is Fake
```

In [57]:
```
news = str(input())
manual_testing(news)
```

```
LR Prediction: NOT Fake News
DT Prediction: NOT Fake News
GBC Prediction: NOT Fake News
RFC Prediction: NOT Fake News
```

In [58]:
```
news = str(input())
manual_testing(news)
```

```
LR Prediction: This News is Fake
DT Prediction: This News is Fake
GBC Prediction: This News is Fake
RFC Prediction: This News is Fake
```

In [61]:
```
news = str(input())
manual_testing(news)
```

```
LR Prediction: NOT Fake News
DT Prediction: This News is Fake
GBC Prediction: This News is Fake
RFC Prediction: NOT Fake News
```

In [62]:
```
news = str(input())
manual_testing(news)
```

```
LR Prediction: NOT Fake News
DT Prediction: This News is Fake
GBC Prediction: This News is Fake
RFC Prediction: NOT Fake News
```

In [ ]: