Artificial Intelligence

Semester Project

Submitted to:

Dr. Pervez Akhtar

Submitted by:

Fatima Rasool (082)

Maryeem Rasool (069)

Khadija Maham (068)

BS CS 6th Semester

Department of Computer Science

National University of Modern Languages

Faisalabad Campus

Implementing 5 machine learning algorithms on a dataset (niche: sports) created by us.

KNN ALGORITHM

import pandas as pd

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

df = pd.read_excel('C:/Users/DELL/Desktop/university/BSCS 6TH/AI/sportsDataset.xlsx')

vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)

X = vectorizer.fit_transform(df['Title']).toarray()

label_encoder = LabelEncoder()

df['Source'] = label_encoder.fit_transform(df['Source'])

y = df['Source']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

```
In [8]: import pandas as pd
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.model_selection import train_test_split
        from sklearn.preprocessing import LabelEncoder
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

In [9]: df = pd.read_excel('C:/Users/DELL/Desktop/university/BSCS 6TH/AI/sportsDataset.xlsx')

In [10]: vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)
         X = vectorizer.fit_transform(df['Title']).toarray()

         label_encoder = LabelEncoder()
         df['Source'] = label_encoder.fit_transform(df['Source'])
         y = df['Source']

         # Split the data
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

conf_matrix = confusion_matrix(y_test, y_pred)

class_report = classification_report(y_test, y_pred)

```
In [11]: # Train the KNN model
         knn = KNeighborsClassifier(n_neighbors=5)
         knn.fit(X_train, y_train)

         # Evaluate the model
         y_pred = knn.predict(X_test)
         accuracy = accuracy_score(y_test, y_pred)
         conf_matrix = confusion_matrix(y_test, y_pred)
         class_report = classification_report(y_test, y_pred)
```

print(f'Accuracy: {accuracy * 100:.2f}%')

print("Confusion Matrix:")

print(conf_matrix)

print("\nClassification Report:")

print(class_report)

```
In [12]: print(f'Accuracy: {accuracy * 100:.2f}%')
         print("Confusion Matrix:")
         print(conf_matrix)
         print("\nClassification Report:")
         print(class_report)

Accuracy: 78.00%
Confusion Matrix:
[[47  7]
 [15 31]]

Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.87      0.81        54
           1       0.82      0.67      0.74        46

    accuracy                           0.78       100
   macro avg       0.79      0.77      0.77       100
weighted avg       0.78      0.78      0.78       100
```

Checking the model by giving news title by dawn news and it says “real”:

```
def predict_new_title(title):
    title_tfidf = vectorizer.transform([title]).toarray()
    prediction = knn.predict(title_tfidf)
```

return label_encoder.inverse_transform(prediction)[0]

Predict for a new title

new_title = "T20 World Cup: Inspired Afghanistan stun Australia, keep semis hopes alive"

predicted_source = predict_new_title(new_title)

print(f"The title '{new_title}' is predicted to be: {predicted_source}")

```
In [14]: def predict_new_title(title):
             title_tfidf = vectorizer.transform([title]).toarray()
             prediction = knn.predict(title_tfidf)
             return label_encoder.inverse_transform(prediction)[0]

         # Predict for a new title
         new_title = "T20 World Cup: Inspired Afghanistan stun Australia, keep semis hopes alive"
         predicted_source = predict_new_title(new_title)
         print(f"The title '{new_title}' is predicted to be: {predicted_source}")

         The title 'T20 World Cup: Inspired Afghanistan stun Australia, keep semis hopes alive' is predicted to be: real
```

Checking the model by giving news title by ChatGPT and it says “AI”:

def predict_new_title(title):

title_tfidf = vectorizer.transform([title]).toarray()

prediction = knn.predict(title_tfidf)

return label_encoder.inverse_transform(prediction)[0]

Predict for a new title

new_title = "Local Team Clinches Championship Title in Thrilling Last-Minute Victory"

predicted_source = predict_new_title(new_title)

print(f"The title '{new_title}' is predicted to be: {predicted_source}")

```
In [15]: def predict_new_title(title):
             title_tfidf = vectorizer.transform([title]).toarray()
             prediction = knn.predict(title_tfidf)
             return label_encoder.inverse_transform(prediction)[0]

         # Predict for a new title
         new_title = "Local Team Clinches Championship Title in Thrilling Last-Minute Victory"
         predicted_source = predict_new_title(new_title)
         print(f"The title '{new_title}' is predicted to be: {predicted_source}")

         The title 'Local Team Clinches Championship Title in Thrilling Last-Minute Victory' is predicted to be: AI
```

Conclusion:

KNN model is predicting correctly because our dataset was not too large and it was balanced.

ANN ALGORITHM

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, Dense, Dropout
from tensorflow.keras.utils import to_categorical
import numpy as np
df = pd.read_excel('C:/Users/DELL/Desktop/university/BSCS 6TH/AI/sportsDataset.xlsx')
# Preprocess the data
vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)
X = vectorizer.fit_transform(df['Title']).toarray()

label_encoder = LabelEncoder()
df['Source'] = label_encoder.fit_transform(df['Source'])
y = to_categorical(df['Source'])
```

In [1]:
```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, Dense, Dropout
from tensorflow.keras.utils import to_categorical
import numpy as np
```

In [2]:
```
df = pd.read_excel('C:/Users/DELL/Desktop/university/BSCS 6TH/AI/sportsDataset.xlsx')
```

In [3]:
```
# Preprocess the data
vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)
X = vectorizer.fit_transform(df['Title']).toarray()

label_encoder = LabelEncoder()
df['Source'] = label_encoder.fit_transform(df['Source'])
y = to_categorical(df['Source'])
```

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = Sequential()

model.add(Input(shape=(X.shape[1],)))

model.add(Dense(512, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(256, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(2, activation='softmax'))

In [4]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [5]:
```
model = Sequential()
model.add(Input(shape=(X.shape[1],)))  # Define the input shape in the Input layer
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))
```

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test), verbose=2)

In [6]:
```
# Compile the model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test), verbose=2)
```

```
Epoch 1/10
13/13 - 1s - 97ms/step - accuracy: 0.5825 - loss: 0.6875 - val_accuracy: 0.7100 - val_loss: 0.6773
Epoch 2/10
13/13 - 0s - 13ms/step - accuracy: 0.8425 - loss: 0.6370 - val_accuracy: 0.8400 - val_loss: 0.6239
Epoch 3/10
13/13 - 0s - 12ms/step - accuracy: 0.9675 - loss: 0.4921 - val_accuracy: 0.8600 - val_loss: 0.4657
Epoch 4/10
13/13 - 0s - 13ms/step - accuracy: 0.9950 - loss: 0.2081 - val_accuracy: 0.8900 - val_loss: 0.2878
Epoch 5/10
13/13 - 0s - 13ms/step - accuracy: 0.9975 - loss: 0.0436 - val_accuracy: 0.9000 - val_loss: 0.2308
Epoch 6/10
13/13 - 0s - 13ms/step - accuracy: 1.0000 - loss: 0.0085 - val_accuracy: 0.8800 - val_loss: 0.2086
Epoch 7/10
13/13 - 0s - 13ms/step - accuracy: 1.0000 - loss: 0.0026 - val_accuracy: 0.8900 - val_loss: 0.2084
Epoch 8/10
13/13 - 0s - 16ms/step - accuracy: 1.0000 - loss: 0.0015 - val_accuracy: 0.9000 - val_loss: 0.2061
Epoch 9/10
13/13 - 0s - 23ms/step - accuracy: 1.0000 - loss: 9.0874e-04 - val_accuracy: 0.9000 - val_loss: 0.2054
Epoch 10/10
13/13 - 0s - 13ms/step - accuracy: 1.0000 - loss: 9.9004e-04 - val_accuracy: 0.9100 - val_loss: 0.2056
```

loss, accuracy = model.evaluate(X_test, y_test, verbose=0)

print(f'Accuracy: {accuracy * 100:.2f}%')

In [7]:
```
# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

```
Accuracy: 91.00%
```

Checking the model by giving news title by dawn news and it says “real”:

```
def predict_new_title(title):
    title_tfidf = vectorizer.transform([title]).toarray()
    prediction = model.predict(title_tfidf)
    predicted_class = np.argmax(prediction, axis=1)
    return label_encoder.inverse_transform(predicted_class)[0]


# Predict for a new title
new_title = "T20 World Cup: Inspired Afghanistan stun Australia, keep semis hopes alive"
predicted_source = predict_new_title(new_title)
print(f"The title '{new_title}' is predicted to be: {predicted_source}")
```

```
In [11]: def predict_new_title(title):
             title_tfidf = vectorizer.transform([title]).toarray()
             prediction = model.predict(title_tfidf)
             predicted_class = np.argmax(prediction, axis=1)
             return label_encoder.inverse_transform(predicted_class)[0]

         # Predict for a new title
         new_title = "T20 World Cup: Inspired Afghanistan stun Australia, keep semis hopes alive"
         predicted_source = predict_new_title(new_title)
         print(f"The title '{new_title}' is predicted to be: {predicted_source}")

         1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
         The title 'T20 World Cup: Inspired Afghanistan stun Australia, keep semis hopes alive' is predicted to be: real
```

Checking the model by giving news title by ChatGPT and it says “real”:

```
def predict_new_title(title):
    title_tfidf = vectorizer.transform([title]).toarray()
    prediction = model.predict(title_tfidf)
    predicted_class = np.argmax(prediction, axis=1)
    return label_encoder.inverse_transform(predicted_class)[0]


# Predict for a new title
new_title = "Local Team Clinches Championship Title in Thrilling Last-Minute Victory"
predicted_source = predict_new_title(new_title)
print(f"The title '{new_title}' is predicted to be: {predicted_source}")
```

```
In [12]: def predict_new_title(title):
             title_tfidf = vectorizer.transform([title]).toarray()
             prediction = model.predict(title_tfidf)
             predicted_class = np.argmax(prediction, axis=1)
             return label_encoder.inverse_transform(predicted_class)[0]

         # Predict for a new title
         new_title = "Local Team Clinches Championship Title in Thrilling Last-Minute Victory"
         predicted_source = predict_new_title(new_title)
         print(f"The title '{new_title}' is predicted to be: {predicted_source}")

         1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 238ms/step
         The title 'Local Team Clinches Championship Title in Thrilling Last-Minute Victory' is predicted to be: real
```

Conclusion:

The ANN model is NOT predicting correctly due to the small dataset size of 500 rows, limiting its ability to learn effectively and generalize reliably as for the ANN model the dataset should be large.

K-MEANS ALGORITHM

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
import numpy as np
df = pd.read_excel('C:/Users/DELL/Desktop/university/BSCS 6TH/AI/sportsDataset.xlsx')
vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)
X = vectorizer.fit_transform(df['Title']).toarray()
kmeans = KMeans(n_clusters=2, random_state=42)  # Example with 2 clusters
kmeans.fit(X)
cluster_labels = kmeans.labels_
```

```
In [1]: import pandas as pd
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.cluster import KMeans
        import numpy as np

In [2]: df = pd.read_excel('C:/Users/DELL/Desktop/university/BSCS 6TH/AI/sportsDataset.xlsx')

In [3]: vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)
        X = vectorizer.fit_transform(df['Title']).toarray()

In [4]: kmeans = KMeans(n_clusters=2, random_state=42)  # Example with 2 clusters
        kmeans.fit(X)
        cluster_labels = kmeans.labels_
```

Checking the model by giving news title by dawn news and it says “real”:

```
def predict_source_from_title(title):
    title_tfidf = vectorizer.transform([title]).toarray()
    cluster_label = kmeans.predict(title_tfidf)[0]
    # Assume you have some logic to determine source based on cluster label
    # For example, if cluster_label == 0, predict 'real'; if cluster_label == 1, predict 'ai'
    predicted_source = 'real' if cluster_label == 0 else 'ai'
    return predicted_source
new_title ="T20 World Cup: Inspired Afghanistan stun Australia, keep semis hopes alive"
```

predicted_source = predict_source_from_title(new_title)

print(f"The title '{new_title}' is predicted to be: {predicted_source}")

```
In [5]: # Function to predict source based on new title
def predict_source_from_title(title):
    title_tfidf = vectorizer.transform([title]).toarray()
    cluster_label = kmeans.predict(title_tfidf)[0]
    # Assume you have some logic to determine source based on cluster label
    # For example, if cluster_label == 0, predict 'real'; if cluster_label == 1, predict 'ai'
    predicted_source = 'real' if cluster_label == 0 else 'ai'
    return predicted_source
```

```
In [7]: # Predict for a new title
new_title ="T20 World Cup: Inspired Afghanistan stun Australia, keep semis hopes alive"
predicted_source = predict_source_from_title(new_title)
print(f"The title '{new_title}' is predicted to be: {predicted_source}")

The title 'T20 World Cup: Inspired Afghanistan stun Australia, keep semis hopes alive' is predicted to be: real
```

Checking the model by giving news title by ChatGPT and it says “real”:

def predict_source_from_title(title):

    title_tfidf = vectorizer.transform([title]).toarray()

    cluster_label = kmeans.predict(title_tfidf)[0]

    # Assume you have some logic to determine source based on cluster label

    # For example, if cluster_label == 0, predict 'real'; if cluster_label == 1, predict 'ai'

    predicted_source = 'real' if cluster_label == 0 else 'ai'

    return predicted_source

new_title ="Local Team Clinches Championship Title in Thrilling Last-Minute Victory"

predicted_source = predict_source_from_title(new_title)

print(f"The title '{new_title}' is predicted to be: {predicted_source}")

```
In [5]: # Function to predict source based on new title
def predict_source_from_title(title):
    title_tfidf = vectorizer.transform([title]).toarray()
    cluster_label = kmeans.predict(title_tfidf)[0]
    # Assume you have some logic to determine source based on cluster label
    # For example, if cluster_label == 0, predict 'real'; if cluster_label == 1, predict 'ai'
    predicted_source = 'real' if cluster_label == 0 else 'ai'
    return predicted_source
```

```
In [8]: # Predict for a new title
new_title ="Local Team Clinches Championship Title in Thrilling Last-Minute Victory"
predicted_source = predict_source_from_title(new_title)
print(f"The title '{new_title}' is predicted to be: {predicted_source}")

The title 'Local Team Clinches Championship Title in Thrilling Last-Minute Victory' is predicted to be: real
```

Conclusion:

The k-Means model is NOT predicting correctly because the TF-IDF method, which converts words into numerical values to help the clustering algorithm understand similarities between news titles, might not be detailed enough to distinguish well.

DECISION TREE ALGORITHM

import pandas as pd

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

df = pd.read_excel('C:/Users/DELL/Desktop/university/BSCS 6TH/AI/sportsDataset.xlsx')

vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)

X = vectorizer.fit_transform(df['Title']).toarray()

y = df['Source']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

```
In [1]: import pandas as pd
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.model_selection import train_test_split
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

In [2]: df = pd.read_excel('C:/Users/DELL/Desktop/university/BSCS 6TH/AI/sportsDataset.xlsx')

In [3]: vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)
        X = vectorizer.fit_transform(df['Title']).toarray()
        y = df['Source']

In [4]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

model = DecisionTreeClassifier(random_state=42)

model.fit(X_train, y_train)

```
In [5]: model = DecisionTreeClassifier(random_state=42)
        model.fit(X_train, y_train)

Out[5]: DecisionTreeClassifier
        DecisionTreeClassifier(random_state=42)
```

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f'Accuracy: {accuracy * 100:.2f}%')

```
In [6]: y_pred = model.predict(X_test)
```

```
In [7]: accuracy = accuracy_score(y_test, y_pred)
        print(f'Accuracy: {accuracy * 100:.2f}%')

        Accuracy: 93.00%
```

print('Confusion Matrix:')

print(confusion_matrix(y_test, y_pred))

print('\nClassification Report:')

print(classification_report(y_test, y_pred))

```
In [8]: print('Confusion Matrix:')
        print(confusion_matrix(y_test, y_pred))
        print('\nClassification Report:')
        print(classification_report(y_test, y_pred))

        Confusion Matrix:
        [[48  6]
         [ 1 45]]

        Classification Report:
                      precision    recall  f1-score   support

                  AI       0.98      0.89      0.93        54
                real       0.88      0.98      0.93        46

            accuracy                           0.93       100
           macro avg       0.93      0.93      0.93       100
        weighted avg       0.93      0.93      0.93       100
```

Checking the model by giving news title by ChatGPT and it says “real”:

new_title = "Local Team Clinches Championship Title in Thrilling Last-Minute Victory"

new_title_tfidf = vectorizer.transform([new_title]).toarray()

predicted_source = model.predict(new_title_tfidf)

print(f"The title '{new_title}' is predicted to be: {predicted_source[0]}")

```
In [11]: new_title = "Local Team Clinches Championship Title in Thrilling Last-Minute Victory"
         new_title_tfidf = vectorizer.transform([new_title]).toarray()
         predicted_source = model.predict(new_title_tfidf)
         print(f"The title '{new_title}' is predicted to be: {predicted_source[0]}")

         The title 'Local Team Clinches Championship Title in Thrilling Last-Minute Victory' is predicted to be: real
```

Checking the model by giving news title by dawn news and it says “real”:

new_title = "T20 World Cup: Inspired Afghanistan stun Australia, keep semis hopes alive"

new_title_tfidf = vectorizer.transform([new_title]).toarray()

predicted_source = model.predict(new_title_tfidf)

print(f"The title '{new_title}' is predicted to be: {predicted_source[0]}")

```
In [12]: new_title = "T20 World Cup: Inspired Afghanistan stun Australia, keep semis hopes alive"
         new_title_tfidf = vectorizer.transform([new_title]).toarray()
         predicted_source = model.predict(new_title_tfidf)
         print(f"The title '{new_title}' is predicted to be: {predicted_source[0]}")

         The title 'T20 World Cup: Inspired Afghanistan stun Australia, keep semis hopes alive' is predicted to be: real
```

Conclusion:

The decision tree model is NOT predicting correctly. The decision tree classifier could mistakenly label both 'real' and 'ai' news titles as 'real' because their TF-IDF features are too similar to distinguish between them accurately.

NAÏVE BAYES ALGORITHM

import pandas as pd

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

df = pd.read_excel('C:/Users/DELL/Desktop/university/BSCS 6TH/AI/sportsDataset.xlsx')

vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)

X = vectorizer.fit_transform(df['Title']).toarray()

y = df['Source']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

```
In [1]: import pandas as pd
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.model_selection import train_test_split
        from sklearn.naive_bayes import MultinomialNB
        from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

In [2]: df = pd.read_excel('C:/Users/DELL/Desktop/university/BSCS 6TH/AI/sportsDataset.xlsx')

In [4]: vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)
        X = vectorizer.fit_transform(df['Title']).toarray()
        y = df['Source']

In [5]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

model = MultinomialNB()

model.fit(X_train, y_train)

```
In [6]: model = MultinomialNB()
        model.fit(X_train, y_train)

Out[6]: MultinomialNB
        MultinomialNB()
```

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f'Accuracy: {accuracy * 100:.2f}%')

```
In [7]: y_pred = model.predict(X_test)

In [8]: accuracy = accuracy_score(y_test, y_pred)
        print(f'Accuracy: {accuracy * 100:.2f}%')

        Accuracy: 84.00%
```

print('Confusion Matrix:')

print(confusion_matrix(y_test, y_pred))

print('\nClassification Report:')

print(classification_report(y_test, y_pred))

```
In [9]: print('Confusion Matrix:')
        print(confusion_matrix(y_test, y_pred))
        print('\nClassification Report:')
        print(classification_report(y_test, y_pred))

        Confusion Matrix:
        [[51  3]
         [13 33]]

        Classification Report:
                      precision    recall  f1-score   support

                  AI       0.80      0.94      0.86        54
                real       0.92      0.72      0.80        46

            accuracy                           0.84       100
           macro avg       0.86      0.83      0.83       100
        weighted avg       0.85      0.84      0.84       100
```

Checking the model by giving news title by dawn news and it says “real”:

new_title = "T20 World Cup: Inspired Afghanistan stun Australia, keep semis hopes alive"

new_title_tfidf = vectorizer.transform([new_title]).toarray()

predicted_source = model.predict(new_title_tfidf)

print(f"The title '{new_title}' is predicted to be: {predicted_source[0]}")

```
In [11]: new_title = "T20 World Cup: Inspired Afghanistan stun Australia, keep semis hopes alive"
         new_title_tfidf = vectorizer.transform([new_title]).toarray()
         predicted_source = model.predict(new_title_tfidf)
         print(f"The title '{new_title}' is predicted to be: {predicted_source[0]}")

         The title 'T20 World Cup: Inspired Afghanistan stun Australia, keep semis hopes alive' is predicted to be: real
```

Checking the model by giving news title by ChatGPT and it says “AI”:

new_title = "Local Team Clinches Championship Title in Thrilling Last-Minute Victory"

new_title_tfidf = vectorizer.transform([new_title]).toarray()

predicted_source = model.predict(new_title_tfidf)

print(f"The title '{new_title}' is predicted to be: {predicted_source[0]}")

```
In [12]: new_title = "Local Team Clinches Championship Title in Thrilling Last-Minute Victory"
         new_title_tfidf = vectorizer.transform([new_title]).toarray()
         predicted_source = model.predict(new_title_tfidf)
         print(f"The title '{new_title}' is predicted to be: {predicted_source[0]}")

         The title 'Local Team Clinches Championship Title in Thrilling Last-Minute Victory' is predicted to be: AI
```

Conclusion:

This naïve Bayes classifier is predicting correctly due to a balanced dataset.