AI-PROJECT

Project Report:

Plagiarism Checker

Submitted by:

FIZZA FAROOQ

SU92-BSDSM-F23-017

BSDS-3A

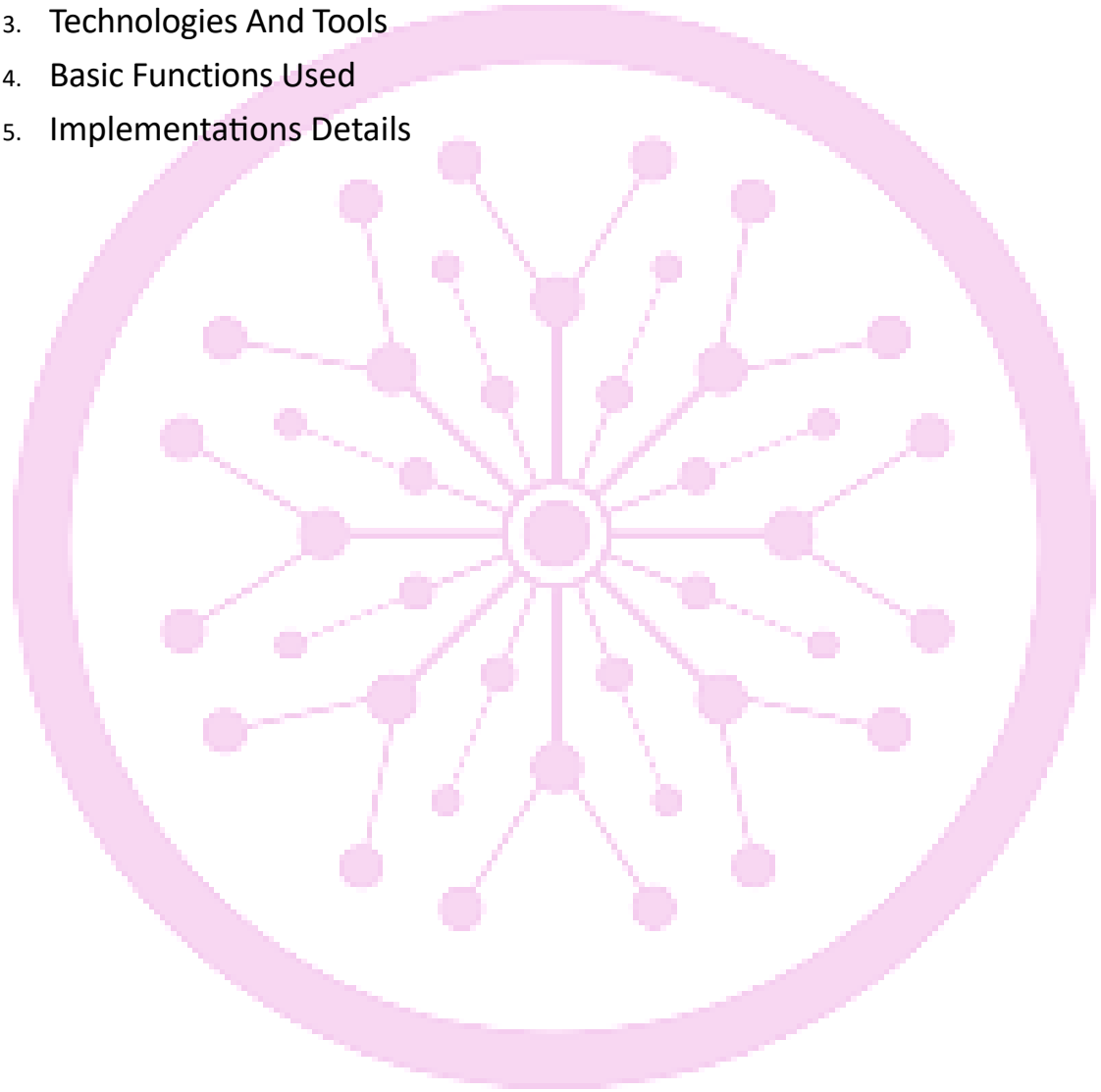
Submitted to:

Prof.RASIKH ALI

Program:

BS DATA SCIENCE

TABLE OF CONTENTS



PROJECT OVERVIEW

The Plagiarism Checker is a Python-based web application designed to detect plagiarism and AI-generated text in documents or user-provided text. This project uses Flask for the web interface and machine learning models for analysis. It leverages a TF-IDF vectorizer to process text and a pre-trained classification model to determine if the given text exhibits characteristics of plagiarism.

OBJECTIVES

1. **Detect Plagiarism:**

   Identify instances of text duplication or AI-generated text within the provided input.

2. **User-Friendly Interface:**

   Provide an intuitive web-based interface for users to input and test text for potential plagiarism.

3. **Scalability:**

   Ensure that it is extendable for future enhancements, including integration with larger datasets.

TECHNOLOGIES AND TOOLS USED

- **Programming Language:** Python
- **Web Framework:** Flask
- **Machine Learning:**
  - Pre-trained classification model (model.pkl)
  - TF-IDF Vectorizer (tfidf_vectorizer.pkl)
- **Frontend:** HTML templates rendered via Flask
- **Backend:** Flask routes and APIs
- **Dataset:** Provided dataset for reference and training the model.

BASIC FUNCTIONS USED

Libraries used:

- nltk - NLP toolkit
- cosine_similarity - Text similarity
- pandas - Data manipulation
- string - String operations
- stopwords - Common word filter
- joblib - Model serialization
- LogisticRegression - Binary classifier
- train_test_split - Data splitting
- accuracy_score - Performance metric
- TfidfVectorizer - Text vectorization

```
import nltk
nltk.download("popular")
from sklearn.metrics.pairwise import cosine_similarity
import pandas as pd
import string
from nltk.corpus import stopwords
import joblib
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,confusion_matrix
from sklearn.feature_extraction.text import TfidfVectorizer
```

Load Dataset:

```
data = pd.read_csv("dataset.csv")
data.head()
```

	Unnamed: 0	source_text	plagiarized_text	label
0	0	Researchers have discovered a new species of b...	Scientists have found a previously unknown but...	1
1	1	The moon orbits the Earth in approximately 27....	Our natural satellite takes around 27.3 days t...	1
2	2	Water is composed of two hydrogen atoms and on...	H2O consists of 2 hydrogen atoms and 1 oxygen ...	1
3	3	The history of Rome dates back to 753 BC.	Rome has a long history that can be traced bac...	1
4	4	Pluto was once considered the ninth planet in ...	In the past, Pluto was classified as the ninth...	1

Clean Text:

The **preprocess_text** function cleans text by removing punctuation, converting it to lowercase, and removing stopwords. It is applied to the **source_text** and **plagiarized_text** columns to prepare data for analysis.

	Unnamed: 0	source_text	plagiarized_text	label
0	0	researchers discovered new species butterfly a...	scientists found previously unknown butterfly ...	1
1	1	moon orbits earth approximately 273 days	natural satellite takes around 273 days comple...	1
2	2	water composed two hydrogen atoms one oxygen atom	h2o consists 2 hydrogen atoms 1 oxygen atom	1
3	3	history rome dates back 753 bc	rome long history traced back 753 bc	1
4	4	pluto considered ninth planet solar system	past pluto classified ninth planet suns planet...	1
...	...	...	...	...
365	397	playing musical instruments enhances creativity	creativity enhanced playing musical instruments	0
366	398	studying history helps understanding present	understanding present aided studying history	0
367	399	listening classical music improve focus	focus improved listening classical music	0
368	400	practicing yoga enhances physical flexibility	physical flexibility enhanced practicing yoga	0
369	401	volunteering fosters community spirit	community spirit fostered volunteering	0

370 rows × 4 columns

Train Test Split:

This code trains a **Logistic Regression model** using X_train and y_train, then predicts labels for X_test. It evaluates the model's performance with **accuracy**, a **classification report** (precision, recall, F1-score), and a **confusion matrix**.

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

Applying logisticRegression

```
model = LogisticRegression()
model.fit(X_train,y_train)

y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test,y_pred))
print("Classification ",classification_report(y_test,y_pred))
print("Confusion ", confusion_matrix(y_test,y_pred))
```

```
Accuracy: 0.8243243243243243
Classification                precision    recall  f1-score   support

           0       0.79      0.86      0.82        35
           1       0.86      0.79      0.83        39

    accuracy                           0.82        74
   macro avg       0.83      0.83      0.82        74
weighted avg       0.83      0.82      0.82        74

Confusion  [[30  5]
 [ 8 31]]
```

Random Forest Model:

This code uses a Random Forest Classifier to train on X_train and y_train, predict labels for X_test, and evaluate the model. It calculates accuracy, generates a classification report (precision, recall, F1-score), and displays a confusion matrix to analyze performance.

```
[32]
from sklearn.ensemble import RandomForestClassifier
# Instantiate the model
model = RandomForestClassifier(n_estimators=100, random_state=42)
# Fit the model
model.fit(X_train, y_train)
# Make predictions
y_pred = model.predict(X_test)
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
# Generate classification report
classification_rep = classification_report(y_test, y_pred)
# Generate confusion matrix
cm = confusion_matrix(y_test, y_pred)
# Print results
print("Accuracy:", accuracy)
print("Classification Report:")
print(classification_rep)
print("Confusion Matrix:")
print(cm)
```

```
Accuracy: 0.7972972972972973
Classification Report:
              precision    recall  f1-score   support

           0       0.71      0.97      0.82        35
           1       0.96      0.64      0.77        39

    accuracy                           0.80        74
   macro avg       0.83      0.81      0.79        74
weighted avg       0.84      0.80      0.79        74

Confusion Matrix:
[[34  1]
 [14 25]]
```

```
from sklearn.ensemble import RandomForestClassifier
# Instantiate the model
model = RandomForestClassifier(n_estimators=100, random_state=42)
# Fit the model
model.fit(X_train, y_train)
# Make predictions
y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test,y_pred))
print("Classification ",classification_report(y_test,y_pred))
print("Confusion ", confusion_matrix(y_test,y_pred))
```

```
Accuracy: 0.7972972972972973
Classification                precision    recall  f1-score   support

           0       0.71      0.97      0.82        35
           1       0.96      0.64      0.77        39

    accuracy                           0.80        74
   macro avg       0.83      0.81      0.79        74
weighted avg       0.84      0.80      0.79        74

Confusion  [[34  1]
 [14 25]]
```

Naiv Bays Model:

This code implements the Multinomial Naive Bayes algorithm, training it on X_train and y_train and predicting X_test. It evaluates the model's performance using accuracy, a classification report, and a confusion matrix.

```python
from sklearn.naive_bayes import MultinomialNB
# Instantiate the model
model = MultinomialNB()
# Fit the model
model.fit(X_train, y_train)
# Make predictions
y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test,y_pred))
print("Classification ",classification_report(y_test,y_pred))
print("Confusion ", confusion_matrix(y_test,y_pred))
```

```
Accuracy: 0.8648648648648649
Classification                precision    recall  f1-score   support

           0       0.86      0.86      0.86        35
           1       0.87      0.87      0.87        39

    accuracy                           0.86        74
   macro avg       0.86      0.86      0.86        74
weighted avg       0.86      0.86      0.86        74

Confusion  [[30  5]
 [ 5 34]]
```

SVM:

This code employs a Support Vector Machine (SVM) with a linear kernel to classify data. It trains the model on X_train and y_train, predicts labels for X_test, and evaluates performance using accuracy, a classification report, and a confusion matrix.

```python
from sklearn.svm import SVC

# Instantiate the model
model = SVC(kernel='linear', random_state=42)

model.fit(X_train, y_train)
# Make predictions
y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test,y_pred))
print("Classification ",classification_report(y_test,y_pred))
print("Confusion ", confusion_matrix(y_test,y_pred))
```

```
Accuracy: 0.8783783783783784
Classification                precision    recall  f1-score   support

           0       0.86      0.89      0.87        35
           1       0.89      0.87      0.88        39

    accuracy                           0.88        74
   macro avg       0.88      0.88      0.88        74
weighted avg       0.88      0.88      0.88        74

Confusion  [[31  4]
 [ 5 34]]
```

Save SVM and Vectorizer

```
import pickle

pickle.dump(model,open("model.pkl",'wb'))
pickle.dump(tfidf_vectorizer, open('tfidf_vectorizer.pkl','wb'))
```

Load Model And Vectorizer:

```
model = pickle.load(open('model.pkl','rb'))
tfidf_vectorizer = pickle.load(open('tfidf_vectorizer.pkl','rb'))
```

Detection System:

```
def detect(input_text):
    # vectorize text
    vectorized_text = tfidf_vectorizer.transform([input_text])
    # thwn will do prediction by model
    result = model.predict(vectorized_text)
    return "Plagiarim Detected" if result[0] == 1 else "No Plagiarism"
```

38]

```
# example ( it is a plagarized text)
input_text = 'Researchers have discovered a new species of butterfly in the Amazon rainforest.'
detect(input_text)
```

39]

```
'Plagiarim Detected'
```

```
# example ( it has no plagiarism)
input_text = 'Playing musical instruments enhances creativity.'
detect(input_text)
```

40]

```
'No Plagiarism'
```

```
# example ( it has no plagarism)
input_text = 'Practicing yoga enhances physical flexibility.'
detect(input_text)
```

```
'No Plagiarism'
```

```
# sklearn version
import sklearn
sklearn.__version__
```

```
'1.3.2'
```

IMPLEMENTATION DETAILS

1. Backend (app.py)

The core logic for the application resides in **app.py.** Key features include:

- **Text Input Handling:**

  Users provide input via a form on the webpage.

- **Plagiarism Detection:**
  - The input text is transformed using the TF-IDF vectorizer.
  - The vectorized data is passed to the pre-trained model to classify the text as "Plagiarism Detected" or "No Plagiarism."
- **Error Handling:**

  The application manages errors gracefully, such as missing form data, ensuring a smooth user experience.

```
def detect(input_text):
    vectorized_text = tfidf_vectorizer.transform([input_text])
    result = model.predict(vectorized_text)
    return "Plagiarism Detected" if result[0] == 1 else "No Plagiarism"
```

2. Web Interface

- **Home Page:** Displays a form where users can input text for analysis.
- **Results Page:** Shows the analysis results, indicating whether plagiarism was detected.

```python
@app.route('/')
def home():
    return render_template('index.html')
```

3. Machine Learning Components

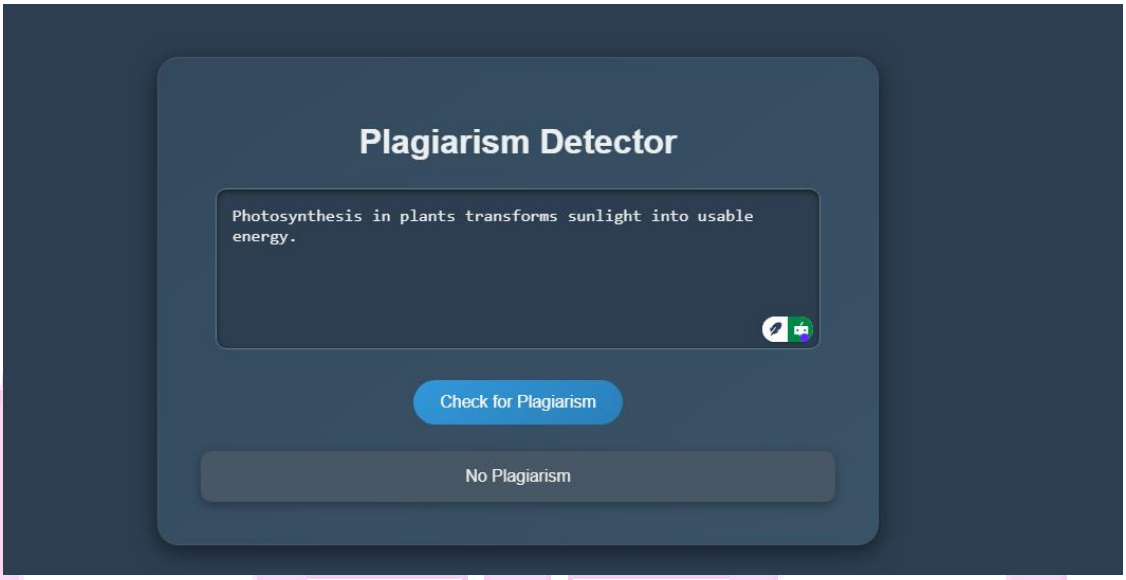
- **Model (model.pkl):** A trained classification model capable of identifying patterns indicative of plagiarism.
- **TF-IDF Vectorizer (tfidf_vectorizer.pkl):** Converts textual input into numerical form, enabling machine learning predictions.

Conclusion:

The Plagiarism Checker is a functional web-based application that efficiently detects plagiarism using machine learning. With its scalable architecture and intuitive interface, it serves as a foundational tool for text analysis and is well-suited for integration into educational and professional environments.

HOME PAGE

In this text “Photosynthesis in plants transforms sunlight into usable energy” plagiarism has been detected.



No Plagiarism