12.Write a python program where you have to check that an article is AI generated or Human generated which gives us at least 70 % accuracy. Compare results for TF-IDF, Word2Vec, and GloVe embeddings for feature extraction over neural networks for classification. Use dataset https://www.kaggle.com/datasets/sunilthite/llm-detect-ai-generated-text-dataset

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
import numpy as np

import re
import nltk
from nltk.tokenize import word_tokenize
from sklearn.model_selection import train_test_split

data=pd.read_csv("/content/drive/MyDrive/Colab Notebooks/NLP/Exp-12/Training_Essay_Data.csv")
print("Total text in df: ", data.shape)
data.tail()
```

```
Total text in df:  (29145, 2)
```

	text	generated
29140	There has been a fuss about the Elector Colleg...	0
29141	Limiting car usage has many advantages. Such a...	0
29142	There's a new trend that has been developing f...	0
29143	As we all know cars are a big part of our soci...	0
29144	Cars have been around since the 1800's and hav...	0

```
y = data['generated'].values
# data.drop(['label'], axis=1, inplace=True)

X_train, X_test, y_train, y_test = train_test_split(data, y, test_size=0.3, stratify=y)

print("Train data:",  X_train.shape, y_train.shape)
print("Test data:",  X_test.shape, y_test.shape)
```

```
Train data: (20401, 2) (20401,)
Test data: (8744, 2) (8744,)
```

```
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.stem import WordNetLemmatizer

# Download required NLTK resources
nltk.download('stopwords')
nltk.download('punkt_tab')
nltk.download('wordnet')
nltk.download('punkt_tab')
# Load stopwords and modify
stop_words = set(stopwords.words('english'))
if 'not' in stop_words:
    stop_words.remove('not')

lemmatizer = WordNetLemmatizer()

def data_preprocessing(text):
    if not isinstance(text, str) or text.strip() == "":  # Ensure valid string input
        return ""

    text = text.lower()
    text = re.sub(r'<.*?>', '', text)  # Remove HTML tags
    text = re.sub(r'\s+', ' ', text).strip()
    text = re.sub(r'[^a-z0-9. ]', '', text)

    sentences = sent_tokenize(text)
    processed_sentences = []

    for sentence in sentences:
        words = word_tokenize(sentence)
        words = [word for word in words if word not in stop_words]
        words = [lemmatizer.lemmatize(word) for word in words]
```

```
            processed_sentences.append(' '.join(words))

        cleaned_text = '. '.join(processed_sentences)

        return cleaned_text.strip()
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
```

```
X_train['text']=X_train['text'].apply(lambda x: data_preprocessing(x))
X_test['text']=X_test['text'].apply(lambda x: data_preprocessing(x))
```

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from sklearn.feature_extraction.text import TfidfVectorizer

# Convert text into TF-IDF features
vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = vectorizer.fit_transform(X_train['text'])
X_train_tfidf = X_train_tfidf.toarray()
```

```
y_train_binary = np.array(X_train['generated'])

# Define input dimensions
input_dim = X_train_tfidf.shape[1]
# 1. Define the model
model = keras.Sequential([
    keras.layers.Dense(128, activation='relu', input_shape=(input_dim,)),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])

# 2. Compile the model
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# 3. Train the model
model.fit(X_train_tfidf, y_train_binary, epochs=10, batch_size=32)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` arg
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/10
638/638 ━━━━━━━━━━━━━━━━━━━━ 8s 9ms/step - accuracy: 0.9299 - loss: 0.1596
Epoch 2/10
638/638 ━━━━━━━━━━━━━━━━━━━━ 7s 12ms/step - accuracy: 0.9988 - loss: 0.0048
Epoch 3/10
638/638 ━━━━━━━━━━━━━━━━━━━━ 6s 10ms/step - accuracy: 1.0000 - loss: 5.8848e-04
Epoch 4/10
638/638 ━━━━━━━━━━━━━━━━━━━━ 7s 11ms/step - accuracy: 1.0000 - loss: 1.3421e-04
Epoch 5/10
638/638 ━━━━━━━━━━━━━━━━━━━━ 9s 9ms/step - accuracy: 1.0000 - loss: 5.1374e-05
Epoch 6/10
638/638 ━━━━━━━━━━━━━━━━━━━━ 8s 12ms/step - accuracy: 1.0000 - loss: 3.0425e-05
Epoch 7/10
638/638 ━━━━━━━━━━━━━━━━━━━━ 9s 10ms/step - accuracy: 1.0000 - loss: 1.7970e-05
Epoch 8/10
638/638 ━━━━━━━━━━━━━━━━━━━━ 7s 11ms/step - accuracy: 1.0000 - loss: 1.1882e-05
Epoch 9/10
638/638 ━━━━━━━━━━━━━━━━━━━━ 6s 10ms/step - accuracy: 1.0000 - loss: 8.2872e-06
Epoch 10/10
638/638 ━━━━━━━━━━━━━━━━━━━━ 7s 12ms/step - accuracy: 1.0000 - loss: 5.2688e-06
<keras.src.callbacks.history.History at 0x7d536cd238d0>
```

```
X_test_tfidf = vectorizer.transform(X_test['text'])
X_test_tfidf = X_test_tfidf.toarray()
y_pred = model.predict(X_test_tfidf)
```

```
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step
```

```
y_pred = (y_pred > 0.5).astype(int)
y_pred
```

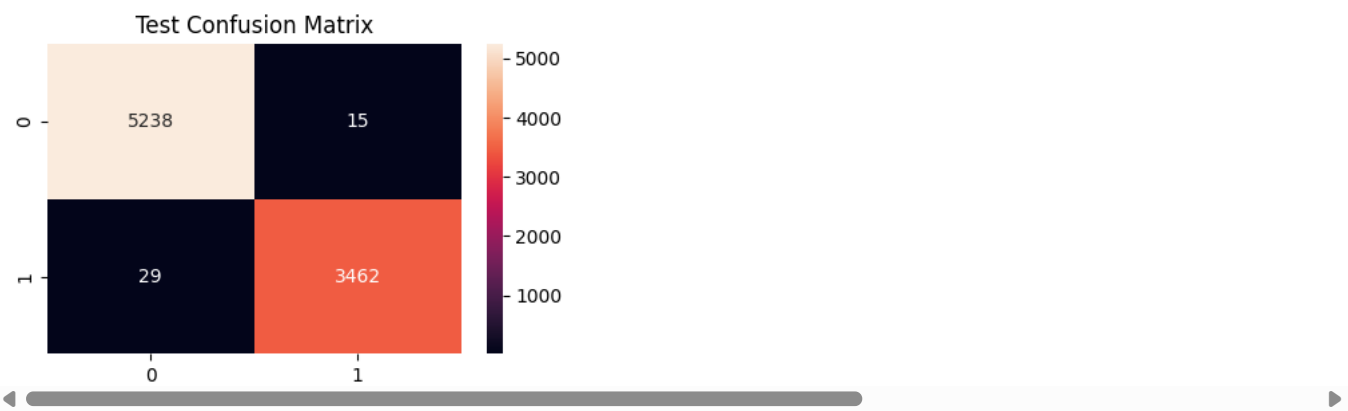
```
array([[1],
       [1],
       [1],
       ...,
       [1],
       [0],
       [0]])
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(X_test['generated'], y_pred)
print(f"Accuracy: {accuracy:.4f}\n")
```

```
Accuracy: 0.9950
```

```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

cm = confusion_matrix(X_test['generated'], y_pred)
plt.figure(figsize=(5, 3))
sns.heatmap(cm, annot=True, fmt='d')
plt.title('Test Confusion Matrix')
plt.show()
```




```
!pip install gensim
```

```
Collecting gensim
  Downloading gensim-4.3.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (8.1 kB)
Collecting numpy<2.0,>=1.18.5 (from gensim)
  Downloading numpy-1.26.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (61 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 61.0/61.0 kB 3.9 MB/s eta 0:00:00
Collecting scipy<1.14.0,>=1.7.0 (from gensim)
  Downloading scipy-1.13.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (60 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 60.6/60.6 kB 3.7 MB/s eta 0:00:00
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.11/dist-packages (from gensim) (7.1.0)
Requirement already satisfied: wrapt in /usr/local/lib/python3.11/dist-packages (from smart-open>=1.8.1->gensim) (1.17.2)
Downloading gensim-4.3.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (26.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 26.7/26.7 MB 66.1 MB/s eta 0:00:00
Downloading numpy-1.26.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (18.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 18.3/18.3 MB 69.1 MB/s eta 0:00:00
Downloading scipy-1.13.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (38.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 38.6/38.6 MB 12.8 MB/s eta 0:00:00
Installing collected packages: numpy, scipy, gensim
  Attempting uninstall: numpy
    Found existing installation: numpy 2.0.2
    Uninstalling numpy-2.0.2:
      Successfully uninstalled numpy-2.0.2
  Attempting uninstall: scipy
    Found existing installation: scipy 1.14.1
    Uninstalling scipy-1.14.1:
      Successfully uninstalled scipy-1.14.1
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the sou
thinc 8.3.6 requires numpy<3.0.0,>=2.0.0, but you have numpy 1.26.4 which is incompatible.
Successfully installed gensim-4.3.3 numpy-1.26.4 scipy-1.13.1
```

```
import gensim
from gensim.models import Word2Vec

X_train_tokenized = X_train['text'].apply(lambda x: x.split()).tolist()
X_test_tokenized = X_test['text'].apply(lambda x: x.split()).tolist()

model_w2v = Word2Vec(sentences=X_train_tokenized, vector_size=200, window=5, min_count=1, workers=4)
```

```
def get_doc_vector(text, model):
    words = text.split()
    vectors = [model.wv[word] for word in words if word in model.wv]
    if vectors:
        return np.mean(vectors, axis=0)
    else:
        return np.zeros(model.vector_size)

X_train_w2v = np.array([get_doc_vector(text, model_w2v) for text in X_train['text']])
X_test_w2v = np.array([get_doc_vector(text, model_w2v) for text in X_test['text']])


y_train_binary = np.array(X_train['generated'])

# Define input dimensions
input_dim = X_train_w2v.shape[1]
# 1. Define the model
model = keras.Sequential([
    keras.layers.Dense(128, activation='relu', input_shape=(input_dim,)),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])

# 2. Compile the model
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# 3. Train the model
model.fit(X_train_w2v, y_train_binary, epochs=10, batch_size=32)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` arg
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/10
638/638 ━━━━━━━━━━━━━━━━━━━━ 9s 6ms/step - accuracy: 0.9571 - loss: 0.1360
Epoch 2/10
638/638 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.9830 - loss: 0.0563
Epoch 3/10
638/638 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.9843 - loss: 0.0478
Epoch 4/10
638/638 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.9893 - loss: 0.0364
Epoch 5/10
638/638 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.9895 - loss: 0.0301
Epoch 6/10
638/638 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.9904 - loss: 0.0280
Epoch 7/10
638/638 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.9913 - loss: 0.0261
Epoch 8/10
638/638 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.9940 - loss: 0.0196
Epoch 9/10
638/638 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.9928 - loss: 0.0200
Epoch 10/10
638/638 ━━━━━━━━━━━━━━━━━━━━ 3s 5ms/step - accuracy: 0.9941 - loss: 0.0181
<keras.src.callbacks.history.History at 0x7d536c6b5a10>
```

```
y_pred = model.predict(X_test_w2v)
```

```
274/274 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
```

```
y_pred = (y_pred > 0.5).astype(int)
y_pred
```

```
array([[1],
       [1],
       [1],
       ...,
       [1],
       [0],
       [0]])
```

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(X_test['generated'], y_pred)
print(f"Accuracy: {accuracy:.4f}\n")
```

```
Accuracy: 0.9911
```

```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

cm = confusion_matrix(X_test['generated'], y_pred)
```

```
plt.figure(figsize=(5, 3))
sns.heatmap(cm, annot=True, fmt='d')
plt.title('Test Confusion Matrix')
plt.show()
```