ДОДАТОК А

Лістинг коду

```
import numpy as np

import pandas as pd

from sklearn.metrics import mean_squared_error,
accuracy_score, recall_score, precision_score, f1_score,
confusion_matrix

from sklearn.model_selection import train_test_split as split


from keras.layers import LSTM, Input, Dense

from keras.models import Model


from deap import base, creator, tools, algorithms

from scipy.stats import bernoulli

from bitstring import BitArray


np.random.seed(1120)

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.preprocessing import StandardScaler

df = pd.read_csv('/kaggle/input/diplomadataset/train.csv',
on_bad_lines='warn')

# Concatenate 'title', 'author', and 'text' into a single
string

df['content'] = df[['title', 'author', 'text']].apply(lambda
x: ' '.join(x.dropna()), axis=1)


# Convert 'label' column to numeric type

df['label'] = pd.to_numeric(df['label'], errors='coerce')


# Drop rows with NaN 'label'

df = df.dropna(subset=['label'])
```

```
# Convert 'label' to integer type
df['label'] = df['label'].astype(int)


# Split data into training and validation sets
X_train, X_val, y_train, y_val =
train_test_split(df['content'], df['label'], test_size=0.2,
random_state=42)


# Convert text data into numerical format using TF-IDF
vectorizer = TfidfVectorizer(max_features=2000)  # Limit to
2000 most frequent words
X_train_tfidf =
vectorizer.fit_transform(X_train).toarray().astype('float32')
X_val_tfidf =
vectorizer.transform(X_val).toarray().astype('float32')


# Scale data


scaler = StandardScaler()
X_train_tfidf_scaled = scaler.fit_transform(X_train_tfidf)
X_val_tfidf_scaled = scaler.transform(X_val_tfidf)


# Reshape input to be 3D [samples, timesteps, features]
X_train_tfidf_reshaped =
X_train_tfidf_scaled.reshape((X_train_tfidf_scaled.shape[0],
X_train_tfidf_scaled.shape[1], 1))
X_val_tfidf_reshaped =
X_val_tfidf_scaled.reshape((X_val_tfidf_scaled.shape[0],
X_val_tfidf_scaled.shape[1], 1))
from keras.callbacks import ModelCheckpoint,EarlyStopping
from keras.models import Model
from keras.initializers import Orthogonal


def orthogonal_initializer(ΓAin=1.0, seed=None):
    return Orthogonal(ΓAin=ΓAin, seed=seed)
```

```
def train_evaluate(ГА_individual_solution):
    num_units_bits = BitArray(ГА_individual_solution)
    num_of_units = min(num_units_bits.uint, 50)
    print('\nNum of Units: ', num_of_units)

    if num_of_units == 0:
        return 100,

    input_ph = Input(shape=(X_train_tfidf_reshaped.shape[1],
1))
    x = LSTM(num_of_units,
kernel_initializer=orthogonal_initializer())(input_ph)
    predicted_values = Dense(1, activation='sigmoid')(x)
    model = Model(inputs=input_ph, outputs=predicted_values)
    model.compile(optimizer='adam',
loss='binary_crossentropy', metrics=['accuracy'])

    early_stopping = EarlyStopping(monitor='val_loss',
patience=3, restore_best_weights=True)
    checkpoint = ModelCheckpoint('model-{epoch:03d}.keras',
verbose=1, monitor='val_loss', save_best_only=True,
mode='auto')

    model.fit(X_train_tfidf_reshaped, y_train, epochs=10,
batch_size=10, validation_split=0.2, shuffle=True,
callbacks=[early_stopping, checkpoint], verbose=1)

    _, accuracy = model.evaluate(X_val_tfidf_reshaped, y_val)
    print('Validation Accuracy: ', accuracy, '\n')

    model.save('my_model.h5')

    return accuracy,
population_size = 6
num_generations = 6
```

```
gene_length = 10


creator.create('FitnessMax', base.Fitness, weights=(1.0,))

creator.create('Individual', list, fitness=creator.FitnessMax)


toolbox = base.Toolbox()

toolbox.register('binary', bernoulli.rvs, 0.5)

toolbox.register('individual', tools.initRepeat,
creator.Individual, toolbox.binary, n=gene_length)

toolbox.register('population', tools.initRepeat, list,
toolbox.individual)


toolbox.register('mate', tools.cxTwoPoint)

toolbox.register('mutate', tools.mutFlipBit, indpb=0.05)

toolbox.register('select', tools.selTournament, tournsize=3)

toolbox.register('evaluate', train_evaluate)


population = toolbox.population(n=population_size)

r = algorithms.eaSimple(population, toolbox, cxpb=0.4,
mutpb=0.2, ngen=num_generations, verbose=True)

# Load the model

from keras.models import load_model

model = load_model('my_model.h5')

# Get the best individual

best_ind = tools.selBest(population, k=1)[0]

print('Best Individual: ', best_ind)

print('Number of Units: ', int(BitArray(best_ind).uint))

optimal_individuals_data = tools.selBest(population,k = 1)
#select top 1 solution

optimal_window_size = None

optimal_num_units = None


for bi in optimal_individuals_data:
```

```
    window_size_bits = BitArray(bi[0:6])
    num_units_bits = BitArray(bi[6:])
    optimal_window_size = window_size_bits.uint
    optimal_num_units = num_units_bits.uint
    print('\n Best Window Size: ', optimal_window_size, ',
Best Num of Units: ', optimal_num_units)
# Load the test data
df_test = pd.read_csv('/kaggle/input/diplomadataset/test.csv',
on_bad_lines='warn')


# Concatenate 'title', 'author', and 'text' into a single
string
df_test['content'] = df_test[['title', 'author',
'text']].apply(lambda x: ' '.join(x.dropna()), axis=1)


# Convert text data into numerical format using TF-IDF
X_test_tfidf =
vectorizer.transform(df_test['content']).toarray().astype('flo
at32')


# Scale data
X_test_tfidf_scaled = scaler.transform(X_test_tfidf)


# Reshape input to be 3D [samples, timesteps, features]
X_test_tfidf_reshaped =
X_test_tfidf_scaled.reshape((X_test_tfidf_scaled.shape[0],
X_test_tfidf_scaled.shape[1], 1))


# Make predictions on the test data
y_pred = model.predict(X_test_tfidf_reshaped)
# Convert predictions to a numpy array
y_pred_array = np.array(y_pred)


# Print the first 10 predictions
print(y_pred_array[:10])
```

```
import matplotlib.pyplot as plt

# Create a histogram of the predictions
plt.hist(y_pred_array, bins=20)
plt.xlabel('Predicted Values')
plt.ylabel('Frequency')
plt.show()
threshold = 0.5
y_pred_labels = (y_pred > threshold).astype(int)
y_pred_labels = y_pred_labels[:len(y_val)]
y_pred_labels = y_pred_labels.flatten()
print(y_val.shape)
print(y_pred_labels.shape)
# Розрахунок точності
accuracy = accuracy_score(y_val, y_pred_labels)

# Розрахунок виклику
recall = recall_score(y_val, y_pred_labels)

# Розрахунок точності
precision = precision_score(y_val, y_pred_labels)

# Розрахунок F1-оцінки
f1 = f1_score(y_val, y_pred_labels)

# Розрахунок матриці помилок
conf_matrix = confusion_matrix(y_val, y_pred_labels)

# Виведення метрик
print(f'Accuracy: {accuracy}')
print(f'Recall: {recall}')
print(f'Precision: {precision}')
```

```
print(f'F1 Score: {f1}')

print(f'Confusion Matrix:\n{conf_matrix}')

def predict_text(text):

    # Convert text to TF-IDF

    text_tfidf =
vectorizer.transform([text]).toarray().astype('float32')


    # Scale data

    text_tfidf_scaled = scaler.transform(text_tfidf)


    # Reshape input to be 3D [samples, timesteps, features]

    text_tfidf_reshaped =
text_tfidf_scaled.reshape((text_tfidf_scaled.shape[0],
text_tfidf_scaled.shape[1], 1))

    # Make prediction

    prediction = model.predict(text_tfidf_reshaped)

    return prediction

def predict_text(text):

    # Convert text to TF-IDF

    text_tfidf =
vectorizer.transform([text]).toarray().astype('float32')

    # Scale data

    text_tfidf_scaled = scaler.transform(text_tfidf)

    # Reshape input to be 3D [samples, timesteps, features]

    text_tfidf_reshaped =
text_tfidf_scaled.reshape((text_tfidf_scaled.shape[0],
text_tfidf_scaled.shape[1], 1))

    # Make prediction

    prediction = model.predict(text_tfidf_reshaped)


    return prediction

new_text = ""

prediction = predict_text(new_text)

print("The predicted label for the new text is: ", prediction)
```

ДОДАТОК Б

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	93 с.
	Інші документи	
2.	Презентаційні матеріали	20 плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Лобанов А.Д.			Еволюційні алгоритми в оптимізації параметрів штучних нейронних мереж	Шифр групи	Код напр./спец.
Перевірив		Філатов В.О.				СШІм-22-1	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Філатов В.О.					