Міністерство освіти і науки України

Національний технічний університет України

"Київський політехнічний інститут імені Ігоря Сікорського"

Фізико-технічний інститут

WEB-АНАЛІТИКА

Лабораторна робота

Робота №4

Виконав студент гр. ФБ-31мп:

Шевченко С. Ш.

Київ – 2024

[Репозиторій](https://github.com/dedkuzmich/web-analytics-2024) Github з кодом.

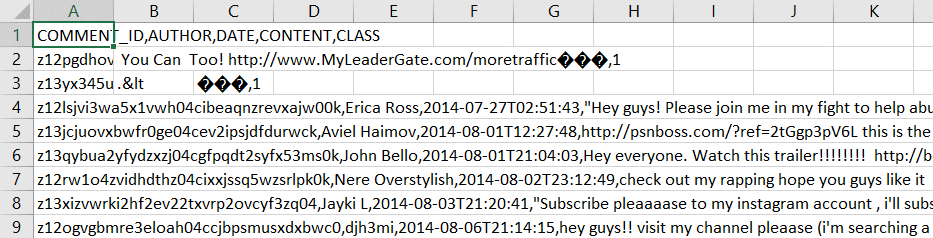
# Завдання:

На основі наданого викладачем датасету (або будь-якого, знайденого самостійно) збудувати систему прогнозування / класифікації (в залежності від датасету та типу даних) із застосуванням методів машинного навчання.

# Виконання завдань:

**Отже, будемо дектувати спамові коментарі на YouTube за домогою рекурентної нейронної мережі (RNN - Recurrent Neural Network).**

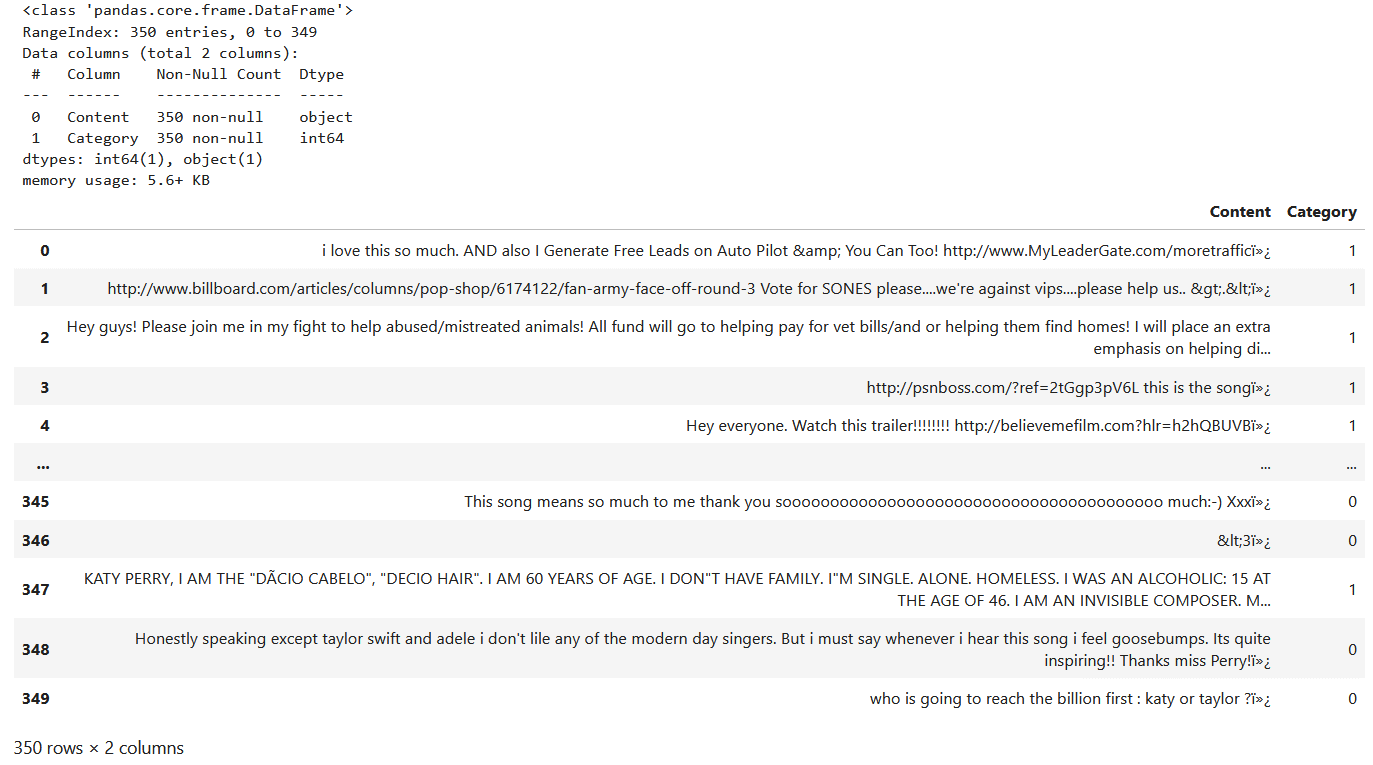
В якості датасету оберемо цю [колекцію](https://www.kaggle.com/datasets/lakshmi25npathi/images) коментарів. Серед всіх таблиць колекції оберемо Youtube02-KatyPerry.csv:



Спамові коментарі кодуються в 1, не спамові – в 0.

Зчитуємо датасет, залишаємо тільки контент (вміст коментаря) та категорію (спам / не спам):

# RECURRENT NEURAL NETWORK (RNN)  
# Load dataset: https://www.kaggle.com/datasets/lakshmi25npathi/images  
file\_dataset = "input/Youtube02-KatyPerry.csv"  
data = pd.read\_csv(file\_dataset, encoding = "latin-1")  
data = data.drop("COMMENT\_ID", axis = 1)  
data = data.drop("AUTHOR", axis = 1)  
data = data.drop("DATE", axis = 1)  
feature\_names = {  
 "CONTENT": "Content",  
 "CLASS": "Category"  
}  
data = data.rename(columns = feature\_names)  
data.info()  
display(data)  
unique\_vals = data["Category"].unique()  
print(unique\_vals)



Препроцесинг коментарів включає в себе:

1) Видалення пунктуації

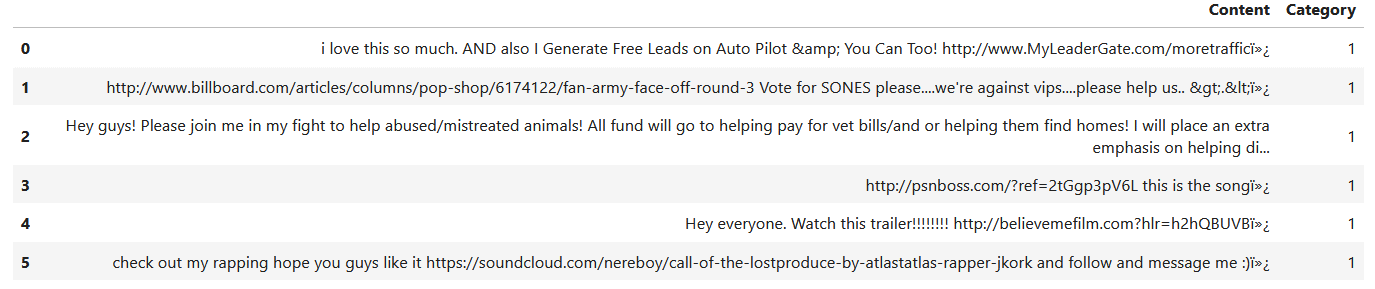
2) Конвертацію в lowcase

3) Видалення стоп слів (сполучники, вигуки, артиклі та інші часто повторювані слова)

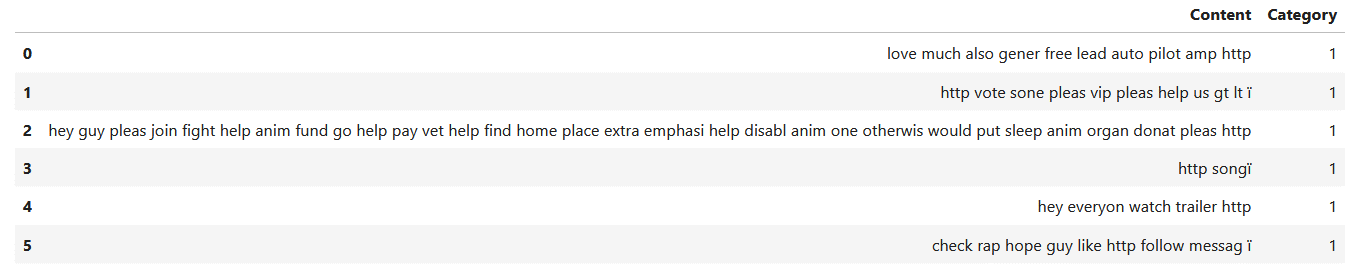
4) Стеммінг, тобто відкидання суфікса та закінчення

# Preprocess texts  
*def* clean\_text(*text*):  
 tokens = word\_tokenize(*text*) # Split text into tokens  
  
 # Remove punctuation, convert to lower case, clean stop words  
 stop\_words = stopwords.words("english")  
 words = []  
 *for* token *in* tokens:  
 *if* token.isalpha():  
 word = token.lower()  
 *if* word *not in* stop\_words:  
 words.append(word)  
  
 # Perform stemming  
 stemmer = PorterStemmer()  
 stemmed\_words = []  
 *for* word *in* words:  
 stemmed\_word = stemmer.stem(word)  
 stemmed\_words.append(stemmed\_word)  
 stemmed\_text = " ".join(stemmed\_words)  
  
 *return* stemmed\_text  
  
  
data\_before = data.copy()  
data["Content"] = data["Content"].apply(clean\_text)  
max\_length = data["Content"].apply(len).max()  
display(data\_before.head(6))  
display(data.head(6))

Датасет до препроцесінга:



Датасет після препроцесінга:



Єдиний недолік цього методу в тому, що видаляються посилання.

Ділимо датесет на тренувальну та тестову вибірки. Проводимо токенізацію:

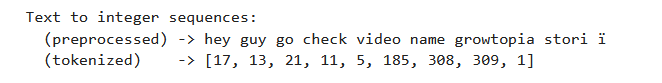
# Split dataset into training set and test set  
X = data["Content"]  
y = data["Category"]  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, random\_state = random\_state)  
  
# Tokenization  
tokenizer = Tokenizer()  
tokenizer.fit\_on\_texts(texts = X\_train)  
vocab\_size = len(tokenizer.word\_index) + 1  
print(f"Vocabulary size: {vocab\_size}")  
print(f"Max text length: {max\_length}\n")



Кодуємо коментарі в послідовності індексів:

# Convert texts to sequences of indexes. Then add padding  
X\_train = tokenizer.texts\_to\_sequences(X\_train)  
X\_test = tokenizer.texts\_to\_sequences(X\_test)  
preprocessed\_text = tokenizer.sequences\_to\_texts([X\_train[0]])[0]  
print("Text to integer sequences:\n"  
 f" (preprocessed) -> {preprocessed\_text}\n"  
 f" (tokenized) -> {X\_train[0]}")  
X\_train = pad\_sequences(X\_train, maxlen = max\_length)  
X\_test = pad\_sequences(X\_test, maxlen = max\_length)

Приклад індексації:



Cтворимо модель RNN із 6 шарів:

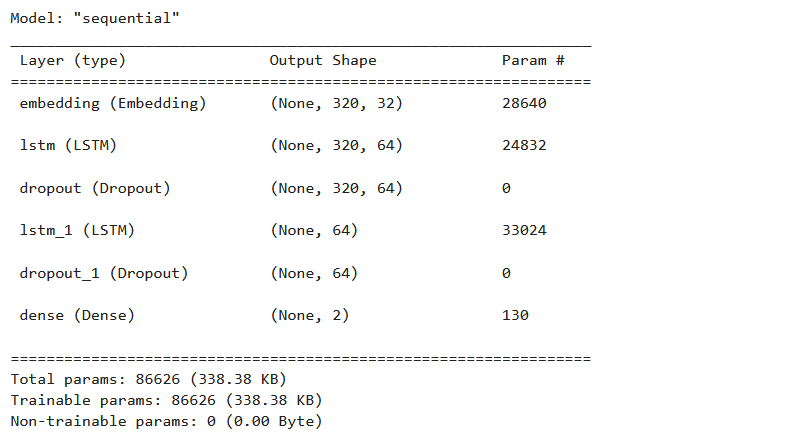
1 шар вбудовування (Embedding)

2 LSTM

1 повнозв'язний шар (Dense)

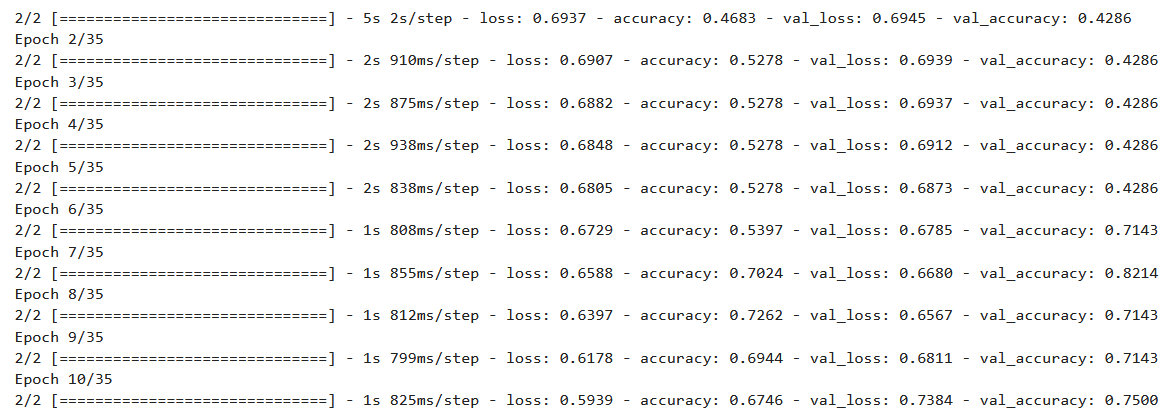
2 дропаути для того щоб уникнути перенавчання

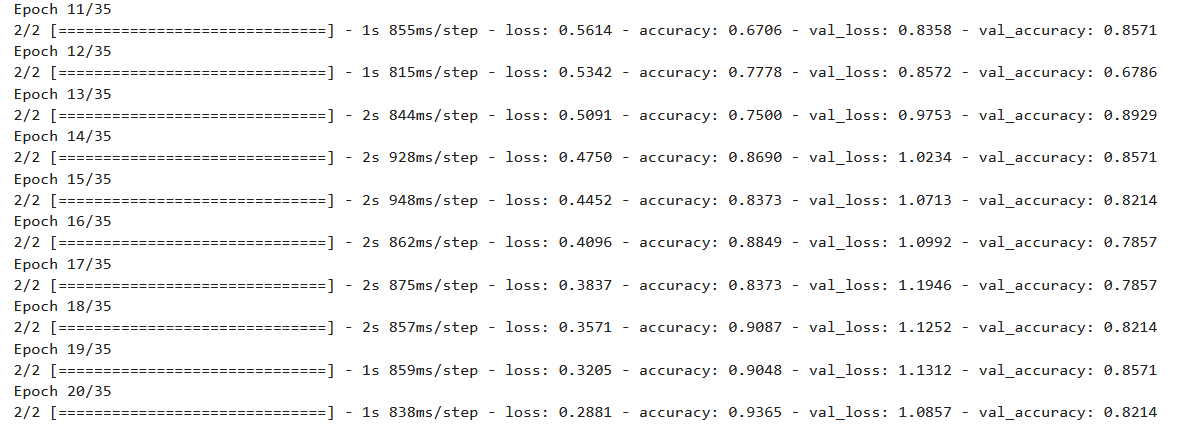
# Create RNN model  
model = Sequential(  
 [  
 Embedding(input\_dim = vocab\_size, input\_length = max\_length, output\_dim = 32),  
 LSTM(units = 64, return\_sequences = *True*),  
 Dropout(0.2),  
 LSTM(units = 64),  
 Dropout(0.3),  
 Dense(units = len(unique\_vals), activation = "softmax")  
 ]  
)  
model.summary()

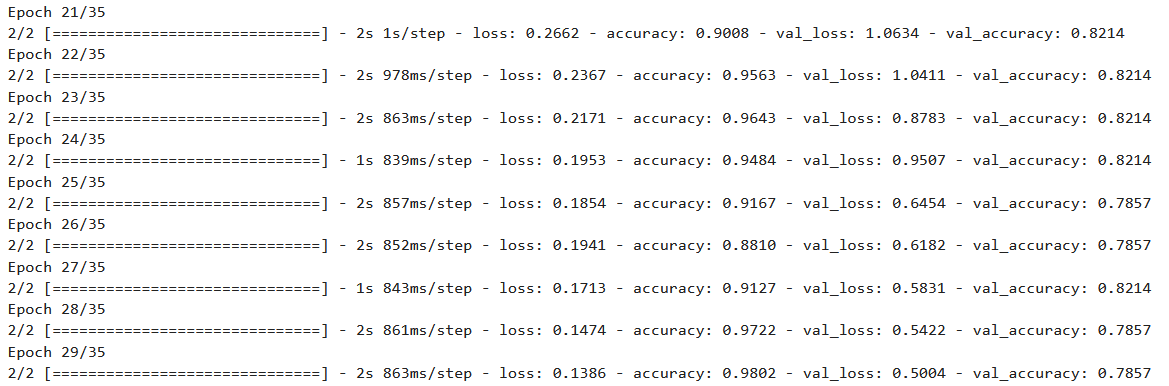


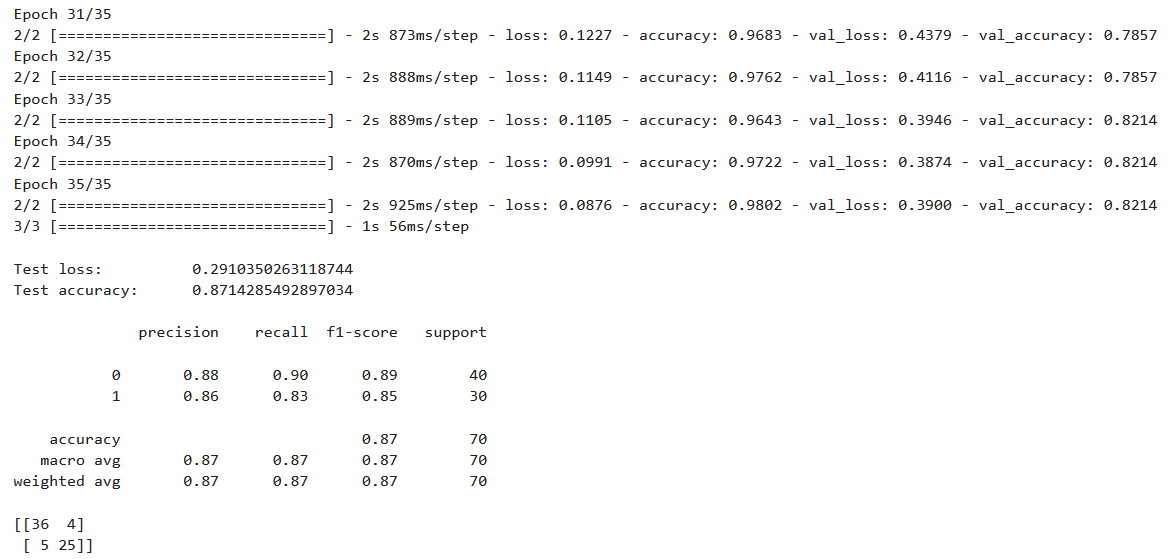
Тренуємо модель, а потім пробуємо класифікувати коментарі з тестової вибірки:

# Compile and fit RNN  
model.compile(  
 optimizer = "adam",  
 loss = "sparse\_categorical\_crossentropy",  
 metrics = ["accuracy"]  
)  
model.fit(X\_train, y\_train, batch\_size = 128, epochs = 35, validation\_split = 0.1)  
y\_pred = model.predict(X\_test)  
estimate\_accuracy(X\_test, y\_test, y\_pred)









Отже, вдалося досягти точності детектування 87%.

Всього коментарів: 70 Розмір тестової вибірки

Не спамових (0): 40

Спамових (1): 30

True Negative (TN): 36 Не спамові, які були віднесені до не спамових

False Negative (FN): 4 Спамові, які були віднесені до не спамових

False Positive (FP): 5 Не спамові, які були віднесені до спамових

True Positive (TP): 25 Спамові, які були віднесені до спамових