Kod zastosowany w zadaniu 3.3 – całość:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk import pos_tag
from nltk.corpus import wordnet
import re
from deep_translator import GoogleTranslator
from time import sleep

nltk.download('stopwords')  # lista słów tj. a, the, itd.
nltk.download('wordnet')  # słownik do lematyzacji
nltk.download('omw-1.4')
nltk.download('averaged_perceptron_tagger')  # rozpoznawanie części mowy
nltk.download('averaged_perceptron_tagger_eng')

stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()

# Wczytanie danych
df = pd.read_excel("MED-lab-3-Zad 3-Mandrill-Dane.xlsx")
df.columns = ['Post', 'label']


# części mowy
def get_wordnet_pos(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN


# Funkcja preprocessingu
def preprocess_text(text):
    if not isinstance(text, str):
        return ""

    # tłumaczenie
    try:
        translated = GoogleTranslator(source='auto',
target='en').translate(text)
    except Exception:
        translated = text

    # zamiana na małe litery
    translated = translated.lower()

    # Usunięcie URL-i
    translated = re.sub(r'https?://\S+|www\.\S+', ' ', translated)
```

```
    # Usunięcie znaków specjalnych
    translated = re.sub(r'[^a-z\s]', ' ', translated)
    translated = re.sub(r'\s+', ' ', translated).strip()

    if not translated:
        return ""

    # podział postów na słowa
    tokens = translated.split()

    # usunięcie stopwords
    tokens = [w for w in tokens if w not in stop_words]

    if not tokens:
        return ""

    # lematyzacja
    pos_tags = pos_tag(tokens)
    tokens = [lemmatizer.lemmatize(w, get_wordnet_pos(tag)) for w, tag in
pos_tags]

    return " ".join(tokens).strip()


# czyszczenie postów
clean_texts = []
for post in df['Post']:
    clean_texts.append(preprocess_text(post))
    sleep(0.1)

# wpisanie do pliku xslx
df['clean_text'] = clean_texts

empty_count = (df['clean_text'].fillna('').str.strip() == '').sum()
print("Liczba pustych clean_text:", empty_count, "z", len(df))

df = df[df['clean_text'].fillna('').str.strip().ne('')].copy()
print("Po usunięciu pustych wierszy:", df.shape)

if df.empty:
    raise ValueError("Po preprocessingu wszystkie dokumenty są puste. "
                     "Najczęściej tłumaczenie zwracało błędy albo tekst po
filtrach był pusty.")

# wektoryzacja
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(df['clean_text'])
y = df['label']

# Podział na zbiory
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Naiwny klasyfikator Bayesa
model = MultinomialNB()
model.fit(X_train, y_train)

# Predykcja i ocena
y_pred = model.predict(X_test)
```

```
print("Dokładność:", accuracy_score(y_test, y_pred))
print("Macierz pomyłek:\n", confusion_matrix(y_test, y_pred))

# Predykcje dla całego zbioru + zapis
df['Predykcja'] = model.predict(vectorizer.transform(df['clean_text']))
df['Poprawnie'] = df['Predykcja'] == df['label']

df.to_excel("Polaczone_Posty_z_predykcjami.xlsx", index=False)
print("Plik zapisany jako 'Polaczone_Posty_z_predykcjami.xlsx'")
print(df[['Post', 'label', 'clean_text', 'Predykcja',
'Poprawnie']].head(10))
```