ОБРОБКА ПРИРОДНОЇ МОВИ (NLP) У PYTHON (КЛАСИФІКАЦІЯ ТЕКСТІВ) ЛЕКЦІЯ 2

КЛАСИФІКАЦІЯ ТЕКСТІВ

- Класифікація текстів це процес класифікації документів в одну або кілька певних категорій. Слід відрізняти класифікацію текстів від кластеризації. В останньому випадку тексти також об’єднуються за деякими критеріями, але заздалегідь задані категорії відсутні. Класифікація може здійснюватися власноруч або автоматично, за допомогою створеного набору правил чи із застосуванням методів машинного навчання.

- Присвоєння категорій текстів, які можуть бути веб-сторінкою, книгою, статтею для ЗМІ та іншим, має безліч застосувань, наприклад: аналіз настроїв, позначення тем, класифікація новин, визначення мови, виявлення намірів, виявлення спаму, маршрутизація клієнтів, класифікація резюме та інше.

- Текстові значення не можна використовувати безпосередньо в моделі машинного навчання. Щоб використовувати їх, нам потрібно перетворити їх в числові значення або функції. Виділення ознак і розробка ознак в тексті відносяться до вилучення інформації з даних в числовому форматі. Використаємо простий і інтуїтивно зрозумілий метод перетворення тексту в корисні функції TF-IDF.

```
#перетворіть усі дані у нижній регістр, а потім збережіть їх у
форматі списку
corpus = df['title'].apply(lambda x : str(x).lower()).tolist()
#встановити цільові змінні
y = df['category']
```

- Ми обрали метод для перетворення тексту в функції. Зараз залишається лише підібрати модель. Важливо випробувати різні моделі, перш ніж приймати рішення про остаточну. Крім того, моделювання найкращого алгоритму класифікації є дуже простим завданням за допомогою scikit-learn — можемо бачити результати різних моделей за допомогою функцій fit() та predict().

Реалізовані моделі:

- Multinomial Naive Bayes
- Support Vector Machines (SVM, LinearSVM)
- Neural Network with Softmax Layer
- Decision Trees
- Random Forests

Метрики, що використовуються для оцінки продуктивності моделей:

- Точність
- Влучність
- Повнота
- F1 міра
- Cohens Kappa міра
- Матриця невідповідностей

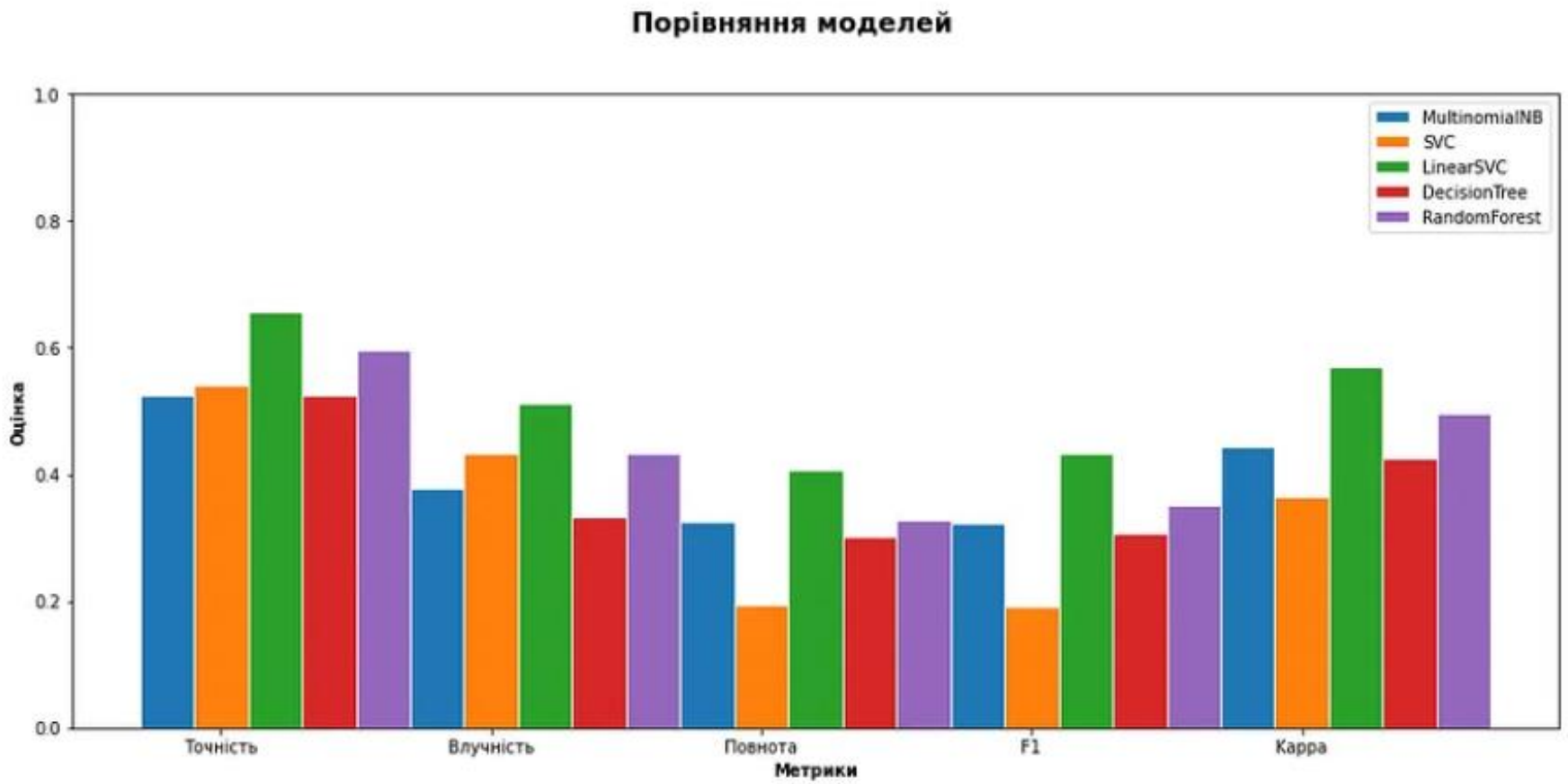
- Ми оцінюємо здатність кожного класифікатора вибрати відповідну категорію з урахуванням назви статті. Матриця невідповідностей створюється для вивчення результатів і розрахунку показників.

```
import pickle
from sklearn.feature_extraction.text import CountVectorizer

count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(corpus)

#зберегти вектори слів
pickle.dump(count_vect.vocabulary_, open("feature.pkl","wb"))

from sklearn.feature_extraction.text import TfidfTransformer

#перетворення векторів слів до TF IDF
tfidf_transformer = TfidfTransformer()
X = tfidf_transformer.fit_transform(X_train_counts)

#зберегти TF-IDF
pickle.dump(tfidf_transformer, open("tfidf.pkl","wb"))
#розділити дані на зразки для навчання та тестування
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)
```

- Щоб оцінити, яка з наших моделей працює краще, нам потрібно розрахувати та порівняти точність між класифікаторами. Оскільки це повторюваний процес, ми можемо полегшити нам, визначивши функцію для оцінки класифікаторів.

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

from sklearn import metrics

#проста функція оцінки
def evaluate(clf, X_train = X_train, X_test = X_test, y_train =
y_train, y_test = y_test):

    import seaborn as sns

    print('Класифікатор : {}'.format(clf))
    clf.fit(X_train, y_train)
    # Predict Test Data
    y_pred = clf.predict(X_test)

# Calculate accuracy, precision, recall, f1-score, and kappa score
    acc = metrics.accuracy_score(y_test, y_pred)
    prc = metrics.precision_score(y_test, y_pred, average='macro')
    rec = metrics.recall_score(y_test, y_pred, average='macro')
    f1 = metrics.f1_score(y_test, y_pred, average='macro')
    kappa = metrics.cohen_kappa_score(y_test, y_pred)
```

```
# Display confussion matrix
    cm = metrics.confusion_matrix(y_test, y_pred)

    print('Метрики : \n')
   # Print result
    print('Точність:', acc)
    print('Влучність:', prc)
    print('Повнота:', rec)
    print('F1 міра:', f1)
    print('Cohens Kappa міра:', kappa)
    print('Матриця невідповідностей:\n', cm)

    print('*'*100)
    print('\n\n')

    return {'acc': acc, 'prc': prc, 'rec': rec, 'f1': f1, 'kappa':
kappa, 'cm': cm}
```


Порівняння моделей
1.0
0.8
0.6
0.4
0.2
0.0
Оцінка
MultinomialNB
SVC
LinearSVC
DecisionTree
RandomForest
Точність
Влучність
Повнота
F1
Карра
Метрики

```
from sklearn.model_selection import GridSearchCV

#спробуйте налаштувати параметри. Для LinearSVC C є регульованим
параметром
params = {'C': [0.1, 1, 10, 100, 1000]}

#використовувати клас GridSearchCV. Основними аргументами є
(класифікатор, параметри), для яких ми хочемо виконати найкращий
пошук параметрів.
grid = GridSearchCV(LinearSVC(), params, refit=True, verbose=3)

#пристосувати дані навчання до нашої моделі grid_search та перевірити
прогнози на нашому тестовому наборі
grid.fit(X_train, y_train)
y_pred = grid.predict(X_test)

#Перевірте точність і влучність нашої моделі
print('Точність : {:.2f}%'.format(100*accuracy_score(y_pred,
y_test)))
print('Влучність : \n{}'.format(100*precision_score(y_pred, y_test,
average = None)))
```

- Оскільки ми знайшли нашу кращу модель і налаштували її, ми хочемо зберегти її, щоб виконувати прогнози на майбутнє безпосередньо, без необхідності заново навчати класифікатор. Нагадаємо, що ми використовували векторизатор, який також відповідав нашим даними. Таким чином, вкрай важливо зберігати один і той же векторизатор для наших тестових і прогнозних даних.

```
import pickle

#зберегти модель категорії

pickle.dump(grid, open('LinearSVM_model.pkl', 'wb'))

# завнтажити модель категорії та перевірити
tuned_model = pickle.load(open('LinearSVM_model.pkl', 'rb'))
result = tuned_model.score(X_test, y_test)*100

print('Точність : {:.2f}%'.format(result))

loaded_vec =
CountVectorizer(vocabulary=pickle.load(open("feature.pkl", "rb")))
loaded_tfidf = pickle.load(open("tfidf.pkl","rb"))
loaded_model = pickle.load(open("LinearSVM_model.pkl","rb"))

docs_new = "Макрон планує знову обговорити з Путіним ситуацію в
Україні"
X_new_counts = loaded_vec.transform([docs_new])
X_new_tfidf = loaded_tfidf.transform(X_new_counts)
predicted = loaded_model.predict(X_new_tfidf)
print('Категорія: ', predicted[0])
```

Точність : 64.68%
Категорія: Політика