NLP MINI PROJECT
on
TEXT CLASSIFICATION

Group Members :

1. **Saurabh Rajendra Jadhav (24)**
2. **Sanket Chandrashekhar Harvande(19)**
3. **Shreya Chandrakant Malavade(35)**

INTRODUCTION

- Text classification has been an important application and research subject since the origin of digital documents. Today, as more and more data are stored in the form of electronic documents, the text classification approach is even more vital. There exist various studies that apply machine learning methods such as Naive Bayes and Convolutional Neural Networks (CNN) to text classification and sentiment analysis.
- However, most of these studies do not focus on cross-domain classification i.e., machine learning models that have been trained on a dataset from one context are tested on another dataset from another context. This is useful when there is not enough training data for the specific domain where text data is to be classified.
- This thesis investigates how the machine learning methods Naive Bayes and CNN perform when they are trained in one context and then tested in another slightly different context. The study uses data from employee reviews in order to train the models, and the models are then tested on both the employee-review data but also on human resources-related data.

PROBLEM STATEMENT AND OBJECTIVE

- **Problem Statement:**

  To identify the spam and non-spam(Ham) words from the given dataset.

- **Objective:**

  We tried different ways to improve our text classification results. We have used:

- Regular Expressions
- Feature Engineering
- Multiple sci-kit-learn Classifiers
- Ensemble Methods

DATASET DETAILS

This corpus has been collected from free or free research sources on the Internet:

- A collection of 425 SMS spam messages was manually extracted from the Grumbletext Web site. This is a UK forum in which cell phone users make public claims about SMS spam messages, most of them without reporting the very spam message received. The identification of the text of spam messages in the claims is a very hard and time-consuming task, and it involved carefully scanning hundreds of web pages.
    - A list of 450 SMS ham messages was collected from Caroline Tag's Ph.D.
    - Finally, we have incorporated the SMS Spam Corpus v.0.1 Big. It has 1,002 SMS ham messages and 322 spam messages

ALGORITHMS AND LIBRARIES USED

- **Sys:**
  The python sys module provides functions and variables which are used to manipulate different parts of the Python Runtime Environment. It lets us access system-specific parameters and functions.
- **nltk:**
  NLTK is a toolkit build for working with NLP in Python. It provides us various text processing libraries with a lot of test datasets. A variety of tasks can be performed using NLTK such as tokenizing, parse tree visualization, etc.
- **sklearn**
  Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistent interface in Python.

ALGORITHMS AND LIBRARIES USED

- **Pandas**

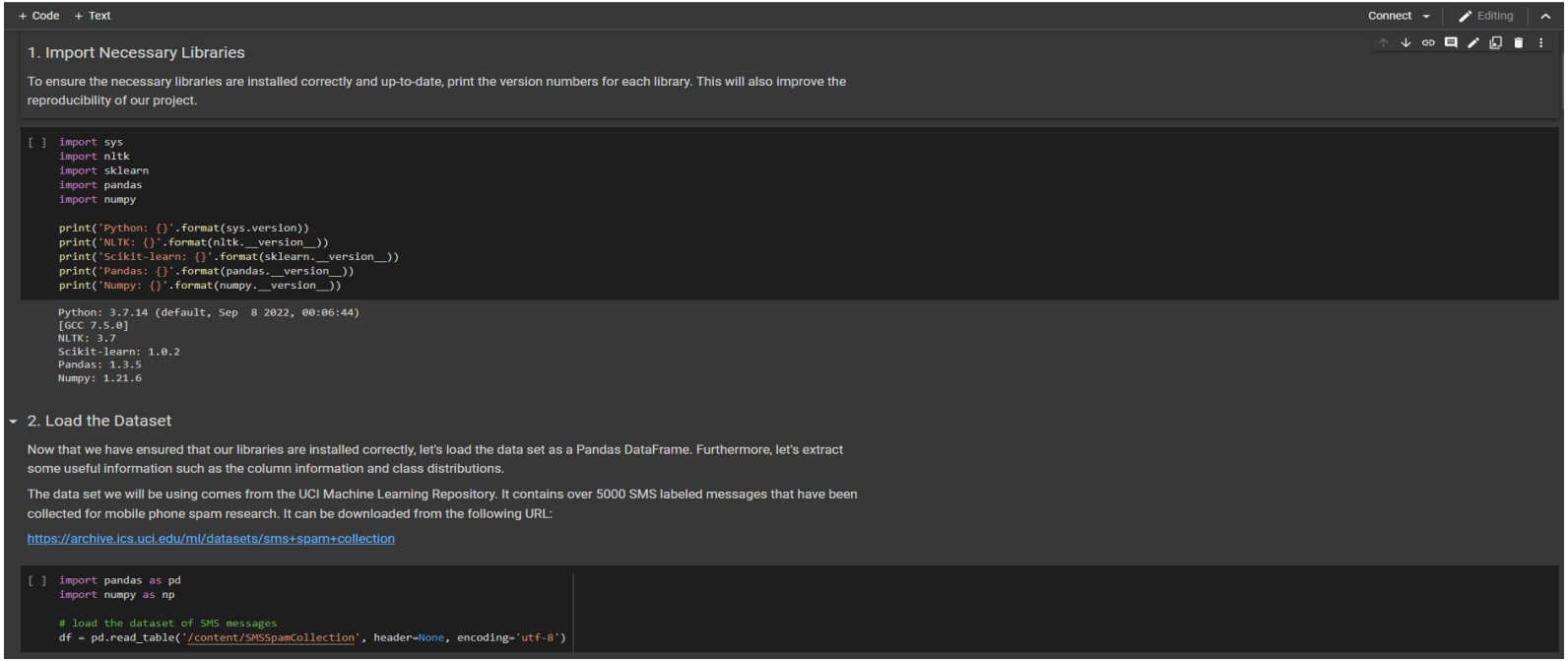
  Pandas is defined as an open-source library that provides high-performance data manipulation in Python. The name of Pandas is derived from the word Panel Data, which means an Econometrics from Multidimensional data. It is used for data analysis in Python.

  Data analysis requires lots of processing, such as restructuring, cleaning or merging, etc. There are different tools are available for fast data processing, such as Numpy, Scipy, Cython, and Panda. But we prefer Pandas because working with Pandas is fast, simple and more expressive than other tools.

- **numpy**

  NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

RESULTS & OUTPUTS



+ Code + Text

Connect Editing

1. Import Necessary Libraries

To ensure the necessary libraries are installed correctly and up-to-date, print the version numbers for each library. This will also improve the reproducibility of our project.

```
[ ] import sys
    import nltk
    import sklearn
    import pandas
    import numpy

    print('Python: {}'.format(sys.version))
    print('NLTK: {}'.format(nltk.__version__))
    print('Scikit-learn: {}'.format(sklearn.__version__))
    print('Pandas: {}'.format(pandas.__version__))
    print('Numpy: {}'.format(numpy.__version__))
```

```
Python: 3.7.14 (default, Sep  8 2022, 00:06:44)
[GCC 7.5.0]
NLTK: 3.7
Scikit-learn: 1.0.2
Pandas: 1.3.5
Numpy: 1.21.6
```

2. Load the Dataset

Now that we have ensured that our libraries are installed correctly, let's load the data set as a Pandas DataFrame. Furthermore, let's extract some useful information such as the column information and class distributions.

The data set we will be using comes from the UCI Machine Learning Repository. It contains over 5000 SMS labeled messages that have been collected for mobile phone spam research. It can be downloaded from the following URL:

https://archive.ics.uci.edu/ml/datasets/sms+spam+collection

```
[ ] import pandas as pd
    import numpy as np

    # load the dataset of SMS messages
    df = pd.read_table('/content/SMSSpamCollection', header=None, encoding='utf-8')
```

RESULTS & OUTPUTS

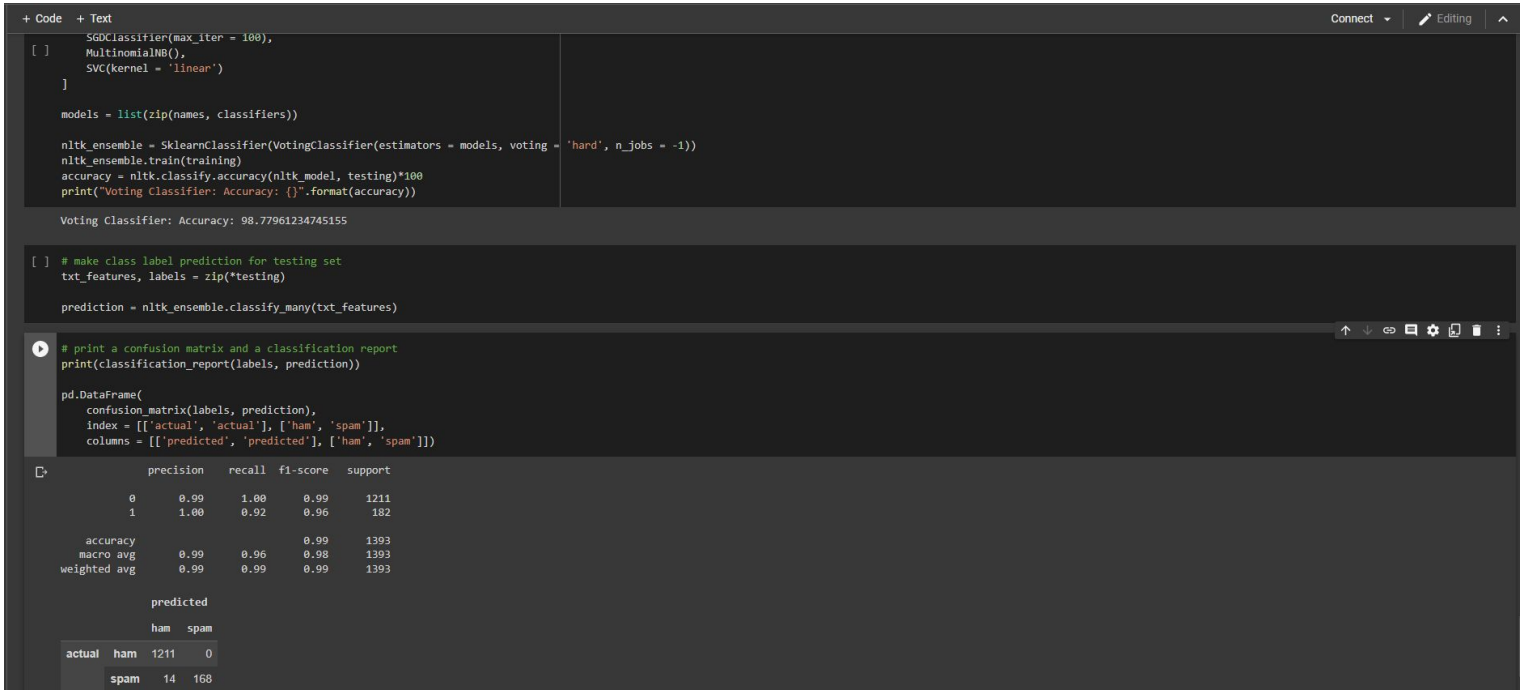
accuracy_score and classification_report.

```
[ ]  # We can use sklearn algorithms in NLTK
     from nltk.classify.scikitlearn import SklearnClassifier
     from sklearn.svm import SVC

     model = SklearnClassifier(SVC(kernel = 'linear'))

     # train the model on the training data
     model.train(training)

     # and test on the testing dataset!
     accuracy = nltk.classify.accuracy(model, testing)*100
     print("SVC Accuracy: {}".format(accuracy))
```

```
SVC Accuracy: 98.77961234745155
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression, SGDClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix

# Define models to train
names = ["K Nearest Neighbors", "Decision Tree", "Random Forest", "Logistic Regression", "SGD Classifier",
         "Naive Bayes", "SVM Linear"]

classifiers = [
    KNeighborsClassifier(),
    DecisionTreeClassifier(),
    RandomForestClassifier(),
    LogisticRegression(),
    SGDClassifier(max_iter = 100),
    MultinomialNB(),
    SVC(kernel = 'linear')
]

models = zip(names, classifiers)

for name, model in models:
```

RESULTS & OUTPUTS



+ Code + Text

Connect Editing

```
        SGDClassifier(max_iter = 100),
        MultinomialNB(),
        SVC(kernel = 'linear')
    ]

    models = list(zip(names, classifiers))

    nltk_ensemble = SklearnClassifier(VotingClassifier(estimators = models, voting = 'hard', n_jobs = -1))
    nltk_ensemble.train(training)
    accuracy = nltk.classify.accuracy(nltk_model, testing)*100
    print("Voting Classifier: Accuracy: {}".format(accuracy))
```

```
Voting Classifier: Accuracy: 98.77961234745155
```

```
# make class label prediction for testing set
txt_features, labels = zip(*testing)

prediction = nltk_ensemble.classify_many(txt_features)
```

```
# print a confusion matrix and a classification report
print(classification_report(labels, prediction))

pd.DataFrame(
    confusion_matrix(labels, prediction),
    index = [['actual', 'actual'], ['ham', 'spam']],
    columns = [['predicted', 'predicted'], ['ham', 'spam']])
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99      1211
           1       1.00      0.92      0.96       182

    accuracy                           0.99      1393
   macro avg       0.99      0.96      0.98      1393
weighted avg       0.99      0.99      0.99      1393
```

		predicted	
		ham	spam
actual	ham	1211	0
	spam	14	168

CONCLUSION

In the project, we learned the basics of tokenizing, part-of-speech tagging, stemming, chunking, and named entity recognition; furthermore, we dove into machine learning and text classification using a simple support vector classifier and a dataset which contains the large amount of words. We characterised the word as per their category as spam and not spam that is ham words

REFERENCES

[1] Y. Zheng, "An Exploration on Text Classification with Classical Machine Learning Algorithm," 2019 International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI), 2019, pp. 81-85, doi: 10.1109/MLBDBI48998.2019.00023.

[2] Venkatesh and K. V. Ranjitha, "Classification and Optimization Scheme for Text Data using Machine Learning Naïve Bayes Classifier," 2018 IEEE World Symposium on Communication Engineering (WSCE), 2018, pp. 33-36, doi: 10.1109/WSCE.2018.8690536.

[3] Ruchika, M. Sharma, and S. A. Hossain, "Text Classification on Twitter Data Using Machine Learning Algorithm," 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2021, pp. 1-3, DOI: 10.1109/ICRITO51393.2021.9596132.