Practical No. 9 (f)

Aim: Text Classification for Spam Detection using Text Mining

Theory:

Spam detection is a critical task in email filtering, messaging platforms, and various online communication systems. Text classification using text mining techniques provides an effective solution for automatically identifying and filtering out spam messages from legitimate ones. In this context, text mining involves extracting relevant features from text data and using machine learning algorithms to classify messages as spam or non-spam.

1. Data Preprocessing:

- The first step involves preprocessing the text data, which includes tasks such as tokenization, lowercasing, removing punctuation, stop word removal, and stemming or lemmatization to normalize the text.
- Additionally, special characters, URLs, and email addresses are often removed or replaced with placeholders.

2. Feature Extraction:

- After preprocessing, features are extracted from the text data. Common techniques for feature extraction in text mining include:
    - Bag-of-Words (BoW): Representing text documents as vectors of word frequencies.
    - TF-IDF (Term Frequency-Inverse Document Frequency): Weighing the importance of words in a document relative to their frequency in the entire corpus.
    - Word Embeddings: Dense vector representations of words learned from large text corpora using techniques like Word2Vec, GloVe, or FastText.

3. Training a Classifier:

- Once the features are extracted, a classification model is trained using a labeled dataset. Popular machine learning algorithms for text classification include:
    - Naive Bayes: A probabilistic classifier based on Bayes' theorem, known for its simplicity and efficiency.
    - Support Vector Machines (SVM): A supervised learning algorithm that finds the hyperplane that best separates classes in a high-dimensional space.
    - Random Forest: An ensemble learning method that builds multiple decision trees and combines their predictions.
    - Deep Learning Models: Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), or Transformer-based architectures like BERT can also be used for text classification.

4. Model Evaluation:

- The trained model is evaluated on a separate test dataset using performance metrics such as accuracy, precision, recall, F1-score, and area under the ROC curve (AUC-ROC).
- Cross-validation techniques may be employed to assess the generalization performance of the model.

5. Deployment:

- Once the model achieves satisfactory performance, it can be deployed into production systems for real-time spam detection.
- Integration with email servers, messaging platforms, or web applications enables automatic filtering of spam messages.

Program Code:

```
# Importing necessary libraries
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report

# Load the spam dataset
url =
"https://raw.githubusercontent.com/rajeevratan84/datascienceforbusiness/master/spam.csv"
data = pd.read_csv(url, encoding='latin-1')

# Keeping only the 'v1' and 'v2' columns and renaming them to 'label' and 'text'
data = data[['v1', 'v2']]
data.columns = ['label', 'text']

# Preprocessing the text data
data['text'] = data['text'].str.lower()  # Convert text to lowercase

# Splitting the dataset into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(data['text'], data['label'], test_size=0.2, random_state=42)

# Creating a Bag-of-Words representation of the text data
vectorizer = CountVectorizer(stop_words='english')
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)

# Training a Naive Bayes classifier
classifier = MultinomialNB()
classifier.fit(X_train_vectorized, y_train)

# Predicting on the test data
y_pred = classifier.predict(X_test_vectorized)

# Evaluating the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

Output:

Result/Conclusion:

By following this lab content, students should gain practical experience in text classification for spam detection, understanding the key steps involved, and exploring ways to improve model performance.