Pattern Recognition And Machine Learning (CS5691) Jan-May 2024

Assignment - 3

Spam Email Detector

Dataset used for training: Spam email Dataset (kaggle.com)

Language used: Python

Libraries used: numpy, pandas, matplotlib, sklearn, chardet, re

Data Preprocessing

Step-1: The encoding of data in the csv file is different so chardet is used to detect the encoding and later the data frame is acquired using this encoding.

Step-2: Text is converted to lowercase so that we do not treat 'Free' and 'free' as two different words.

Step-3: Remove symbols and numbers using **re.sub** function in **re**. This will replace all numbers/symbols with empty strings. This will allow us to focus more on words containing only English alphabet.

```
df['text'] = df['text'].apply(lambda text: re.sub(r'[^a-zA-Z\s]', '', text))
df['text'] = df['text'].apply(lambda text: re.sub(r'\bsubject\b', '', text,
flags=re.IGNORECASE))
```

Step-4: **CountVectorizer** function is imported from **sklearn**. Calling this function on the 'email' column will allow it to traverse through the column and find all unique words and treat their frequency as a feature for each vector. Finally, it returns vector corresponding to each email containing the frequency of a particular word in the email. Also, it will remove any stop words (frequently used words in conversations) by setting the parameter stop_words to 'English'.

```
vectorizer = CountVectorizer(stop_words='english')
X = vectorizer.fit_transform(df['text'])
X = X.toarray()
# print(vectorizer.get_feature_names_out())
y = np.array(df['spam'])
y
```

```
emails = X
labels = y
```

Step-5: A labels array is made where a spam is denoted by ‘1’ and a ham with ‘0’.

Step-6: We can split the data for validation.

Designing a classifier:

Algorithms to be used:

1. Naive Bayes Algorithm
2. Logistic Regression
3. SVM
4. Gaussian Naive Bayes(Optional)

Naive Bayes Algorithm:

Convert the data into binary data for applying naive bayes algorithm.

Create a group of spam and ham emails using the labels in train data.

Calculate the required estimators:

$$\hat{p} = \frac{\sum_{i=1}^{N} y_i}{N}$$

$$\hat{p_{y,j}} = \frac{\sum_{i=1}^{N} \mathbb{1}(f_{ij}=1, y_i=y)}{\sum_{i=1}^{N} \mathbb{1}(y_i=y)}$$

For each email in the test dataset, calculate the values of

$$P(x_i|y_i = 1) = \hat{p} \times \prod_{j=1}^{D} (\hat{p_{1,j}}^{f_{ij}} (1 - \hat{p_{1,j}})^{1-f_{ij}})$$

$$P(x_i|y_i = 0) = (1 - \hat{p}) \times \prod_{j=1}^{D} (\hat{p_{0,j}}^{f_{ij}} (1 - \hat{p_{0,j}})^{1-f_{ij}})$$

4. Check the below condition which is true when probability to assigning 1 to label is more than assigning 0.(It can be obtained by above probabilities and assuming that probability of assigning 1 is more than 0 to reach the condition)

$$\sum_{i=1}^{d} (f_i \log \frac{\hat{p_{1,i}}}{\hat{p_{0,i}}} + (1 - f_i) \log \frac{1-\hat{p_{1,i}}}{1-\hat{p_{0,i}}}) + \log \frac{\hat{p}}{1-\hat{p}} >= 0$$

We use this logarithmic condition because when we multiply different probabilities to compare probability of being spam to non spam, we can get very small values which may not be computed efficiently due to computational limits.

Accuracy: On the test-split, accuracy is 96%

Gaussian Naive Bayes Algorithm:

Use the frequency data as it is(not converting it to binary).

Create a group of spam and ham emails using the labels in train data.

Calculate the required estimators:

$$\hat{p} = \frac{\sum_{i=1}^{N} y_i}{N}$$

$$\hat{\mu_y} = \frac{\sum_{i=1}^{N} x_i \times \mathbb{1}(y_i = y)}{\sum_{i=1}^{N} \mathbb{1}(y_i = y)}$$

$$\hat{\Sigma} = \frac{\sum_{i=1}^{N} (x_i - \hat{\mu_{y_i}})(x_i - \hat{\mu_{y_i}})^T}{N}$$

For each email in the test dataset, calculate:

$$f(x_t; \hat{\mu_1}, \hat{\Sigma}) \times \hat{p} > f(x_t; \hat{\mu_0}, \hat{\Sigma}) \times (1 - \hat{p})$$

For predicting label = 1.

Logistic Regression Algorithm:

Using the maximum likelihood concept, we find out the gradient. Then we perform the gradient ascent to find the optimal w.

Gradient:

$$\nabla LogL = \sum_{i=1}^{N} x_i(y_i - sigmoid(w^T x_{test}))$$

Gradient Ascent Update Step:

$$w_{t+1} = w_t + n_t \nabla LogL(w)$$

Then we calculate the predicted values using this equation:

$$y_{test} = sign(w^T x_{test})$$

Step-size (eta)	Accuracy (on test-split)
10^{-1}	98.78%
10^{-2}	99.13%
10^{-3}	96.6%
10^{-4}	90.22%

Therefore, best step-size = 10^{-2}

Time: Approximately 2 min for completion.

Max-Accuracy: On the test-split, accuracy is 99%

SVM Algorithm:

Used the library sklearn to call an instance of SVM classifier with linear kernel.

C-Value (Cost)	Accuracy (on test-split)
C = 1	98.43%
C = 0.1	98.25%
C = 5	98.43%

Code Snippet:

```
clf = svm.LinearSVC(C= 1, max_iter=10000)
clf.fit(emails, labels)
```

Time: Approximately 1 minute to fit the data, 1 min for prediction.

Accuracy: On the test-split, accuracy is 98.43% (Accuracy does not change much for C>=1)

Note: The time required to execute Gaussian Naive Bayes Algorithm is much higher, therefore it is implemented but not taken into account in determining the final classification

Final Classifier:

We know that some of the algorithms like SVM or Logistic Regression give reasonably correct predictions, but I feel the results will be better if we ensemble the available models.

Therefore, we can take the majority of the predictions by each model and return this as our final prediction. Formally,

$$final\_pred(x_{test}) = \mathbb{1}(logistic(x_{test}) + naive\_bayes(x_{test}) + SVM(x_{test}) \geq 2)$$

This classifier will generate better predictions as it will have properties of each of these algorithms.

Final Evaluation:

We will read the emails from test folder and predict the values using the models whose hyperparameters were trained using the training dataset previously. These predicted values from different models to make the final prediction using the final classifier.

The final prediction is an array in which each element returns 1 or 0 based on the corresponding email being spam or ham.

Final Predictions:

```
test_emails = vectorizer.transform(test_emails)
test_emails = test_emails.toarray()
pred_svm = clf.predict(test_emails)
pred_lr = [sgn(np.dot(email, w)) for email in test_emails]
pred_nba = np.zeros(len(test_emails))
for i, email in enumerate(test_emails):
    email = np.where(email != 0, 1, email)
    if np.dot(w_naive, email) + b_naive >= 0:
        pred_nba[i] = 1
```

```
final_pred = pred_svm + pred_lr + pred_nba
final_pred = np.where(final_pred >= 2, 1, 0)
```

```
with open('predictions.txt', 'w') as f:
    for i in range(len(final_pred)):
        f.write(f"Email {i+1} : {final_pred[i]}\n")
```

The predictions made by the final classifier is stored in the ‘predictions.txt’. This will contain Email No. along with the predicted value.