ushur

Assessing ML model performance - A Primer

V 0.1

ntroduction

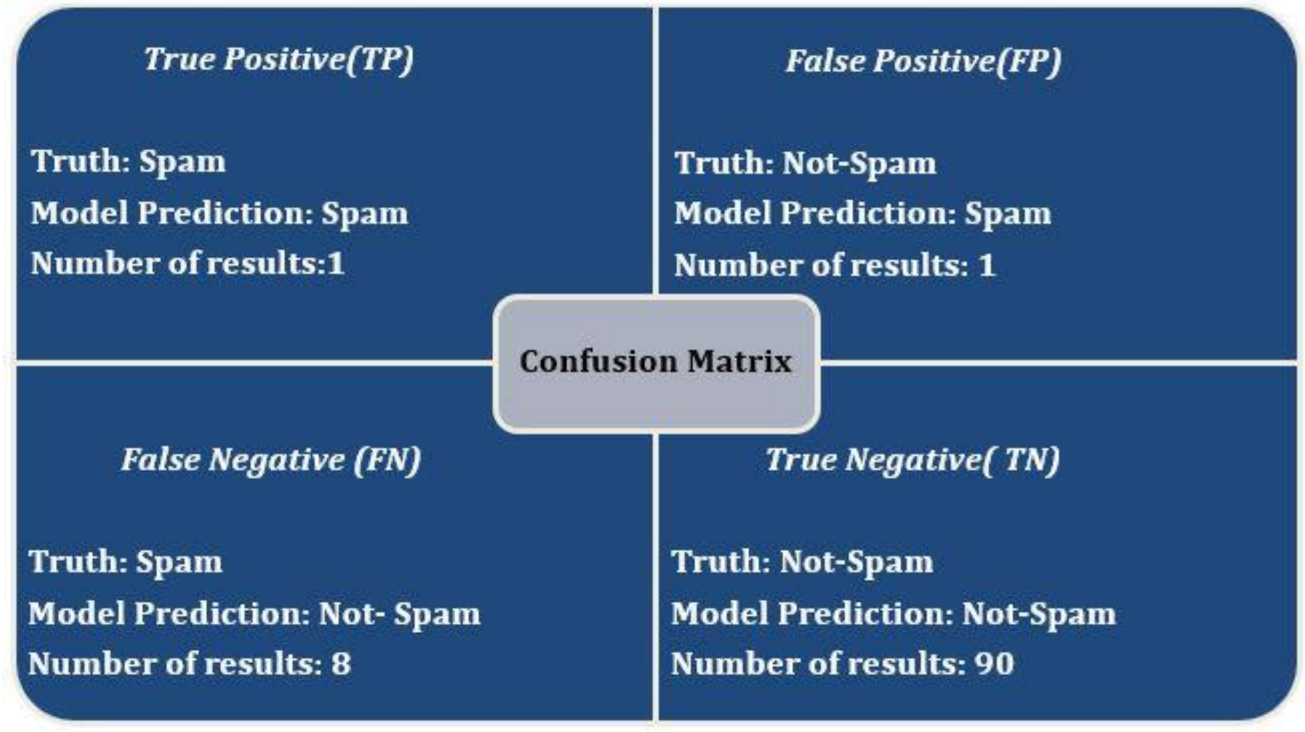
In the Ushur intelligent workflow automation platform, it is easy and intuitive to build various machine-learning models for Natural Language Processing (NLP) workflows. These range from simple sentence similarity detectors to complex classification and sentiment analysis models that employ state-of-art deep learning techniques. The following sections aim to illustrate the various methods that can be used to effectively assess the model performance & how tools within the Ushur ecosystem can enable the same.

Understanding Model Metrics

In evaluating the performance of Machine-Learning models for real-world business use cases, accuracy is often used as a defining metric. Accuracy can be intuitively thought of as a proportion of correct results the model has achieved. But using accuracy is a naïve approach and leads us to incorrectly gauge the model's predictive power. Therefore, in statistical model testing, additional metrics are used to estimate the overall performance of ML models. Let us take a quick tour of these metrics.

Consider a simple e-mail classification model that classifies 100 e-mail samples as either "spam" (positive class) or "not-spam" (negative class)

The spam prediction model can be summarized into a 2 x 2 ***"confusion matrix"***, which lists all the possible outcomes:



As seen above,

A *"True Positive"* is an outcome where the model correctly predicts the "positive" class. A *"True Negative"* is when it correctly predicts the "negative" class

A *"False Positive"* on the other hand, is when the model incorrectly predicts the "positive" class. A *"False Negative"* is when it incorrectly predicts the "negative" class.

A straightforward metric used for evaluating the performance of models is *"Accuracy"*. It is simply defined as the fraction of predictions the model got right.

Accuracy = Total number of correct predictions / Total number of Predictions

For binary classifications like above (spam vs not-spam), it can be rewritten in terms of positives and negatives as:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Where *TP* = True Positives, *TN* = True Negatives, *FP* = False Positives, and *FN* = False Negatives

Based on above formula, the accuracy for our model would be,

Accuracy = (1 + 90) / (1 + 90 + 1 + 8) = **0.91 or 91%**

91% accuracy looks like a good number and model seems to be doing a great job at identifying spam emails. But the devil is in the details!

In the test sample of 100, there are 91 actual non-spam emails. The model correctly identifies 90 of them as non-spam. ***But out of the 9 spam emails, the model only correctly identifies 1 as spam***. This implies that 8 out of 9 spam emails go undetected, which is a pretty lousy outcome! Consider a pathologically bad model (with zero-predictive ability) that predicts "not-spam" all the time. Even that model would get 91 (out of 100) samples correctly classified as "not-spam"! So, the overall accuracy of 91% didn't make us any better than a model with zero-predictive ability. To summarize, in most real-world use cases, there is an inherent imbalance in class data and accuracy alone will not suffice to assess the overall performance of ML models. We should instead look at two metrics called ***Precision*** and ***Recall***.

Precision defines how many predictions were actually correct amongst all positive predictions of the model.

Precision = TP / (TP + FP)

So, for our above model, Precision = 1 / (1 + 1) = 0.5 or 50%

This implies that when the model predicts an email as spam, it is correct 50% of the time. To improve the precision of a model, we need to minimize false-positives.

Recall defines what proportion of actual positives were correctly predicted (or how many did we miss).

Recall = TP / (TP + FN)

So, for our above model, Recall = 1 / (1 + 8) = 0.11 or 11%

This indicates that the model identifies 11% of spam emails correctly. To improve the recall of a model, we need to minimize the false-negatives.

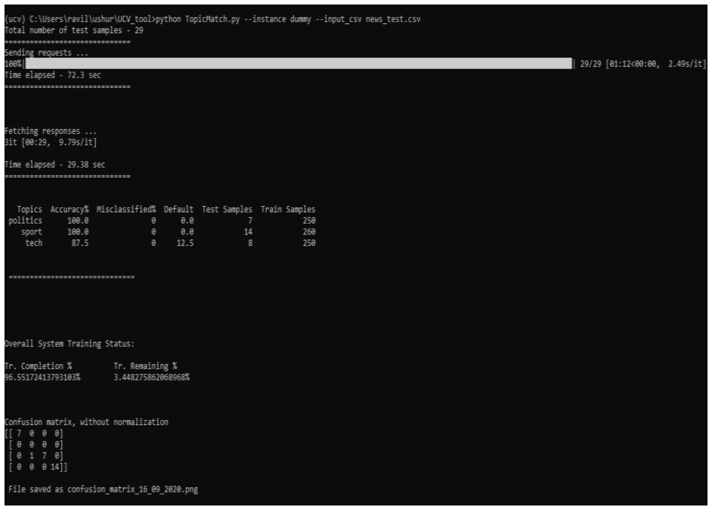
Based on the above, we have another metric called the **"F1 Score"**. It is a harmonic mean of precision and recall and is defined as:

F1 = 2 * (Precision * Recall) / (Precision + Recall)

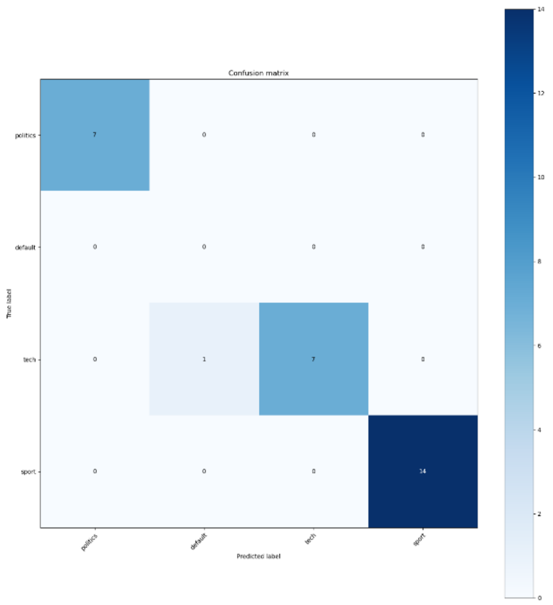
This metric can be thought of as an alternate to the overall accuracy and is very useful when we seek a balance between precision and recall

Testing Ushur ML models

Once a model is trained in the Ushur platform, its performance can be evaluated using the **UCV (Ushur Classification Verifier)** tool. It's a simple, python-based on-premise model testing tool. It utilizes the Ushur platform's REST API to submit inference requests and retrieve the classification results. All the relevant metrics (per-category precision, recall, f1-score, confusion matrix, etc) are output by this tool.



```
(ucv) C:\Users\ravil\ushur\UCV_tool>python TopicMatch.py --instance dummy --input_csv news_test.csv
Total number of test samples - 29
==============================
Sending requests ...
100%|██████████████████████████████████████████████████| 29/29 [01:12<00:00,  2.49s/it]
Time elapsed - 72.3 sec
==============================



Fetching responses ...
3it [00:29,  9.79s/it]

Time elapsed - 29.38 sec
==============================


   Topics  Accuracy%  Misclassified%  Default  Test Samples  Train Samples
 politics      100.0               0      0.0             7            250
    sport      100.0               0      0.0            14            260
     tech       87.5               0     12.5             8            250


 ==============================





Overall System Training Status:

Tr. Completion %          Tr. Remaining %
96.55172413793103%        3.448275862068968%



Confusion matrix, without normalization
[[ 7  0  0  0]
 [ 0  0  0  0]
 [ 0  1  7  0]
 [ 0  0  0 14]]

 File saved as confusion_matrix_16_09_2020.png
```



As shown above, the UCV tool outputs the precision/recall (misclassification/accuracy) of each category in the model. It also generates a confusion matrix in the form of a PNG file which visually depicts the spread of classifications across all categories. The categories which exhibit sub-optimal performance can be easily identified and earmarked for further improvement.

Make your work flow.

The general rules of thumb for improving the performance of sub-optimal categories are:

- If the number of training samples is very less, try adding more data.
- Check for labeling errors. Potential mislabels identified by the Ushur platform can also be made available for this exercise
- Double-check the data collection process for potential *"sampling bias"*. A biased sample is one that is not representative of the entire population. Data collection should be purely a random exercise, wherein each data point has an equal chance of being chosen (using an entire database/ backup/PST archive, collecting samples across a wide time range, etc).
- In some scenarios, two or more categories can have overlap. It might be hard to distinguish between categories based on the textual content of samples. In those cases, consider using the Ushur platform's intelligent data extraction capability to identify KBIs (Key Business Indicators) which can then be coupled with Metadata feature to override the model's predicted classifications.
- If the number of categories is large it might be helpful to start with fewer categories and progressively add more. The initial set of categories can be chosen based on volume/availability of data, business value, etc
- Experiment with other model types. The Ushur platform supports a wide variety of NLP models starting from simple TF-IDF, Doc2Vec, or word2vec based SVM models up to neural-network-based deep learning models like fasttext, BiLSTM, ULMfit, etc.

Assessing Model Performance

As illustrated in earlier sections, the overall accuracy should not be the be-all and end-all model metric. In most real-world business use cases, data is inherently imbalanced & as a result, the overall accuracy will be contributed significantly by a large number of true negatives/positives. But the false positives and false negatives are more interesting as they have associated business costs (both tangible and intangible).

The overall effectiveness of the model can be assessed by considering both precision and recall or F1 metrics. But in reality, improving recall hampers precision and vice-versa. We need to arrive at a suitable trade-off by assessing the business opportunity costs involved.

Consider the example of classifying emails in an insurance company into a “claim inquiry” vs a “general inquiry” category. If we “miss” detecting claim-related emails (low recall), the response to the user might be delayed inordinately. This might lead to a bad customer experience and also have an impact on business SLAs. On the other hand, if we incorrectly classify emails in the “general inquiry” category into the claim category, we might end up sending it to the wrong business unit/personnel thus generating additional internal work in the process. The “cost” in each of these scenarios need to be evaluated to arrive at the right set of acceptable values for the model.