

**2 - Machine Learning Techniques**

**Inlämning 2**

You have been provided with a script that **already handles data loading, cleaning, preprocessing, and vectorization** for a text classification task in Swedish. This is a continuation from **Inlämning 1**, where you learned how to perform text preprocessing (lowercasing, removing stopwords, etc.).

* **Important**: Since the preprocessing is already done here (and you practiced it in Exercise 1), **you do not need to replicate that** in this assignment.
* The code you wrote in **Inlämning 1** will be integrated into future **Group Project** files (coming soon).
* **You do not need to import any external scripts** (e.g., your Inlämning 1 script) for this assignment; simply **focus on training and testing your models** on the CSV file that’s already being preprocessed in the provided code.

**Your Task**

1. **Examine the Provided Code**
   * It loads, shuffles, and splits the data.
   * It cleans the text (lowercase, removes punctuation, digits, HTML tags, and Swedish stopwords) and optionally performs stemming.
   * Finally, it converts the text into TF-IDF vectors and produces x\_train, y\_train, x\_test, y\_test.
2. **Choose and Import a Classifier**
   * Common options: LogisticRegression, SVC, RandomForestClassifier, SGDClassifier, or any other suitable method from sklearn.
   * If the data involves **multi-label classification**, consider using OneVsRestClassifier or a similar approach.
3. **Train Your Model**
   * Call .fit(x\_train, y\_train) on your classifier.
   * If it’s multi-label data, you can either use .predict\_proba() with a chosen threshold (e.g., 0.3) or directly do .predict() if your model supports it.
4. **Predict and Evaluate**
   * Use .predict(x\_test) (or .predict\_proba(x\_test) + thresholding for multi-label) to get predictions.
   * Compute **accuracy** (via accuracy\_score) and **other relevant metrics** (e.g., precision, recall, F1-score) for a thorough evaluation.
5. **Experiment!**
   * Adjust hyperparameters (e.g., C in LogisticRegression or SVC, number of trees in RandomForestClassifier, etc.).
   * Compare **multiple** models to see which yields the **best accuracy** on x\_test.
6. **Submit Your Best Approach**
   * Turn in your final code.
   * Document **which models** you tried and **how** you tuned them.
   * The group with the **highest accuracy** on the test data wins bragging rights!

**Tips**

* Consult the **scikit-learn documentation** for example usage and parameter-tuning strategies (GridSearchCV, RandomizedSearchCV, etc.).
* Keep a simple log of your experiments (which model, which hyperparameters, what accuracy) to compare results.
* If your dataset has **multiple labels per instance**, be sure to handle multi-label classification properly (e.g., by using OneVsRestClassifier or threshold-based approaches if you use predict\_proba).