Personality prediction from user’s posts

Using Myers-Briggs Type Indicators

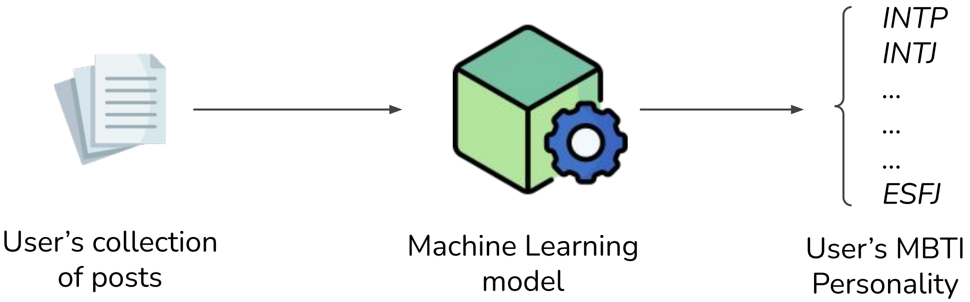
Eduardo Rinaldi - 1797800

What is MBTI?

- It discriminates between **16 possible personality types**, identified by 4 different characteristics that can each present themselves in two alternative ways:
    - Introversion (**I**) vs Extroversion (**E**): indicates how you are energized
    - Intuition (**N**) vs Sensing (**S**): indicates how you obtain information for your decisions
    - Thinking (**T**) vs Feeling (**F**): measures your preference to operate from your head or your heart
    - Perceiving (**P**) vs Judging (**J**): indicates how you like to order your life
- Usually, in order to know which of the 16 personalities is closest to ours one, we have to answer a **questionnaire**

Project objective

- This project aims to **automate** this task creating a model that, taken a user's collection of posts as input, it discriminates between **16 personalities** choosing the most suitable one
- This is a **classification task**

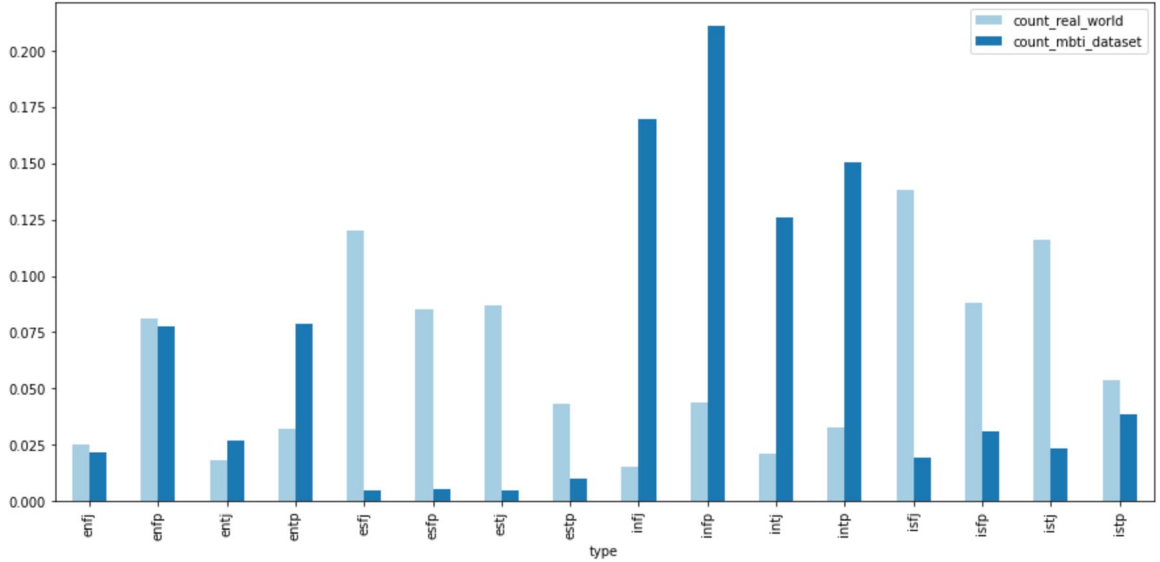


Dataset (1): Kaggle MBTI

- Was collected through the ***PersonalityCafe forum*** as it provides a large selection of people and their MBTI personality type (dataset is **labeled**)
- It's a "`.csv`" file containing over **8600 rows of data**; each row contains:
  - **"posts"**: last 50 things a user have posted (Each entry separated by `"|||"`)
  - **"type"**: MBTI type
- **Notice:** splitting each row by "|||" will produce a dataset with (~)430k rows

Credit and source here

Kaggle MBTI vs Real world distribution



Real world distribution provided from [here]

- Very unbalanced dataset
  - **Introvert** personalities are most frequents, **Sensing** (*xSxx*) personalities less frequents
  - Several classes with too few examples for applying some kind of training on them (e.g. “*esfj*”)
- Real world distribution is almost the opposite of our dataset distribution

Dataset (2): Reddit MBTI

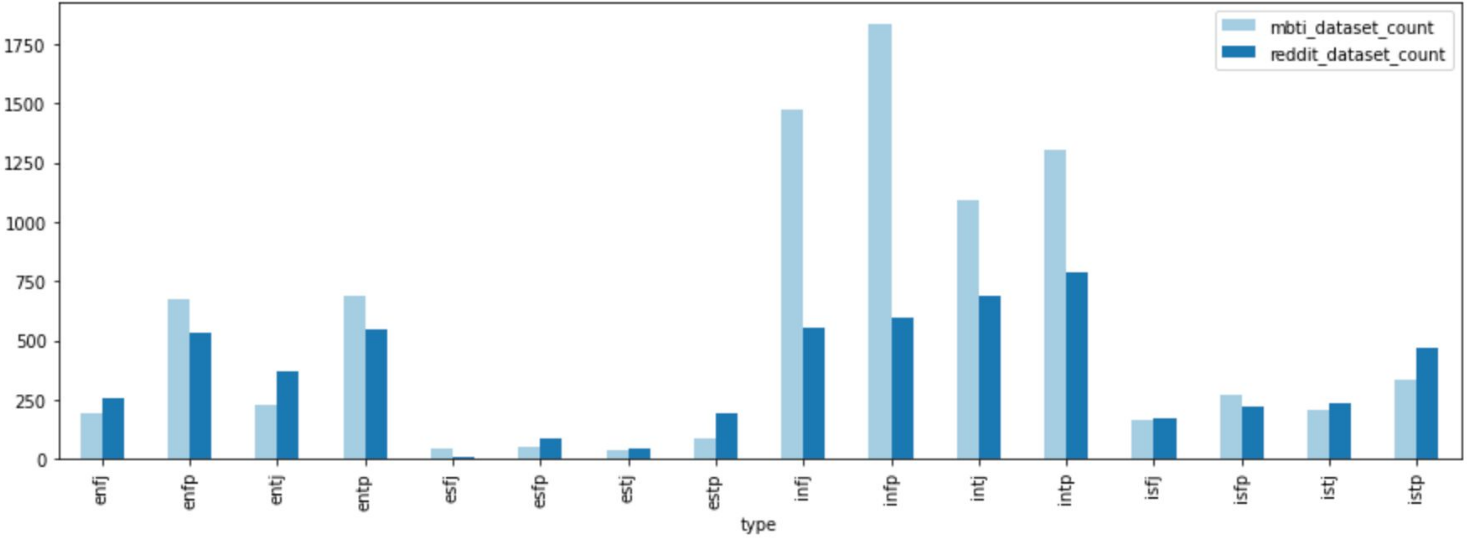
- Was collected through **Reddit** using a scraper created by me, based on *Reddit API*
- It's a "`.parquet`" dataset composed by **5754 rows**, each of which contains:
  a. **"redditor_id":** posts author id
  b. **"post":** last ***n*** things a user have posted and each entry separated by `"|||"`
  c. **"text_type":** identify if it's a *comment*, *title* or a *post*
  d. **"type":** MBTI type personality associated to the author
  e. **"num_post":** number of post in the row (***n***, ranging from 50 to 100)

- But, **how did the scraper collect data**?

Dataset (2): Reddit MBTI

1. I collected a list of users who posted something in a **MBTI related subreddit** ("*r/mbti*", "*r/infp*", ...)
    a. Personality information is given by a **badge** that is assigned to the user (i.e. `"author_flair_text"`)
2. Then I collected the most recent things each collected user posted on the **entire** Reddit platform (i.e. posts not MBTI related are also included)
3. At the end, for each collected post I assigned personality (i.e. type) based on author's badge.

Kaggle MBTI vs Reddit MBTI



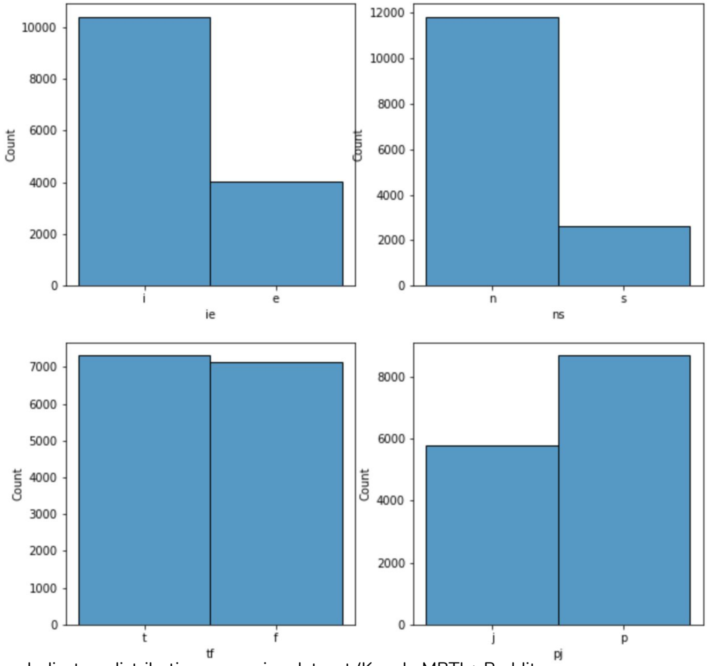
Similar distributions; this implies same problems:

- Unbalanced dataset
- Few examples for less frequent classes

Problems (1) - Few examples on less frequent classes

Solution:

- split “type” in 4 different indicators
- train **4 different binary classifiers**, each of which can have its own training algorithm and parameters to tune.



Indicators distribution over union dataset (Kaggle MBTI + Reddit dataset)

Problems (2) - Unbalanced dataset

Unbalanced dataset: *"Exxx"* and *"xSxx"* types are still very unbalanced

Possible solutions:

- **Undersampling majority classes:** few data for applying this strategy
- **Oversampling minority classes:** since we're dealing with a very unbalanced dataset, balancing it with a lot of duplicates from minority classes could lead to overfitting
- Prefer **other evaluation metrics** over accuracy, so f1-score, precision and recall.

Only last solution has been adopted for this project

Grouped posts vs single post

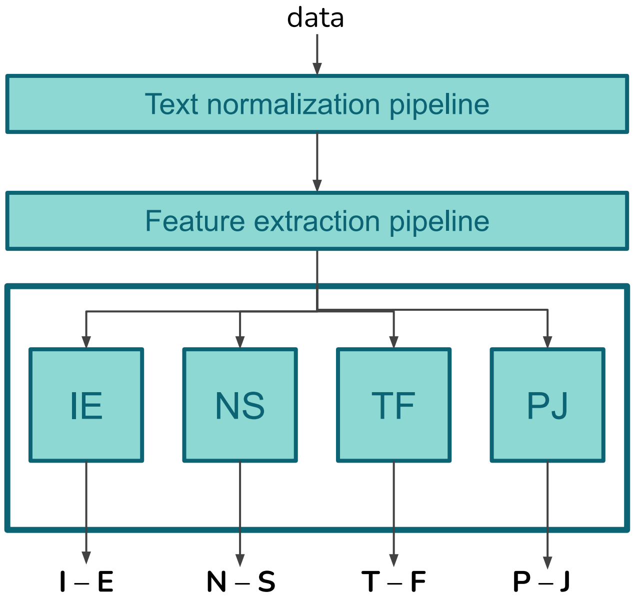
Only ~14.5k examples, but each one has very large text (a lot of information about a user).
We will test 2 approaches:

- **“Single post approach” (SPA):** consists in splitting each row by “|||”, so we will obtain a new dataset with about 1 million examples.
- **“Grouped posts approach” (GPA):** consists in using actual dataset with multiple posts on each row

Idea

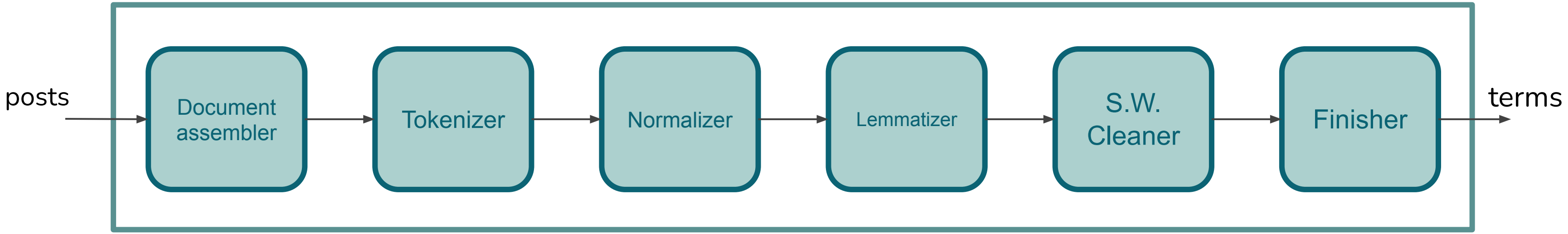
The idea is to create a pipeline composed by **3 main stages**:

- Text cleaning and normalization
- Feature extraction
- 4 Binary classifier



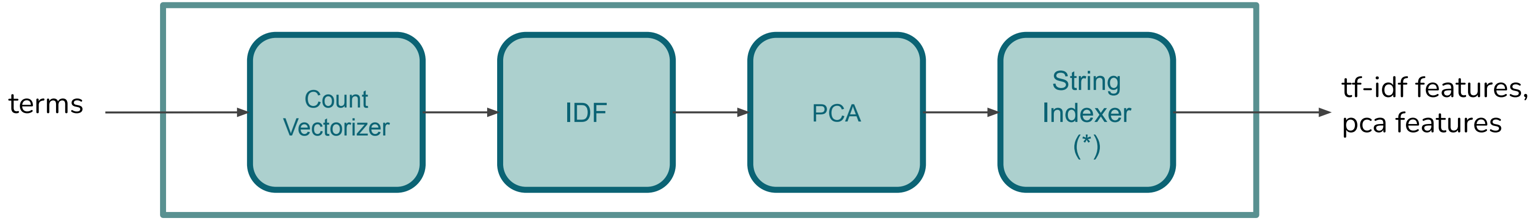
Text cleaning and normalization

Text normalization **pipeline**, using **SparkNLP** *annotators* and *transformers*



Feature extraction

Feature extraction **pipeline**



(*) One StringIndexer for each type indicator (total of 4)

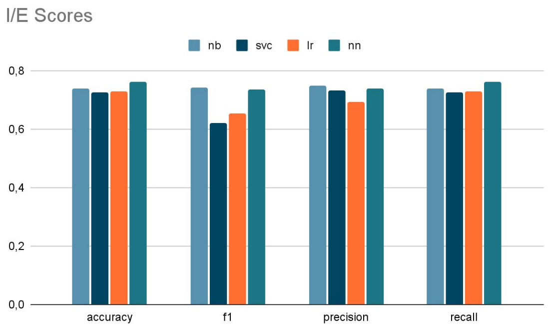
ML algorithms used

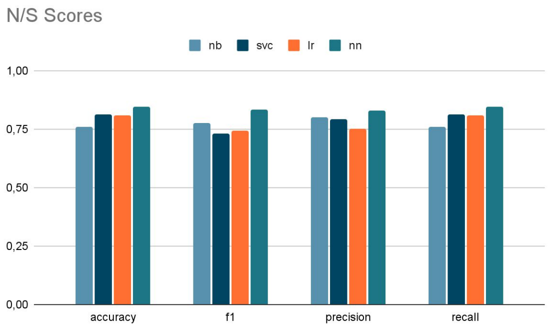
For each type indicator I trained the following models:

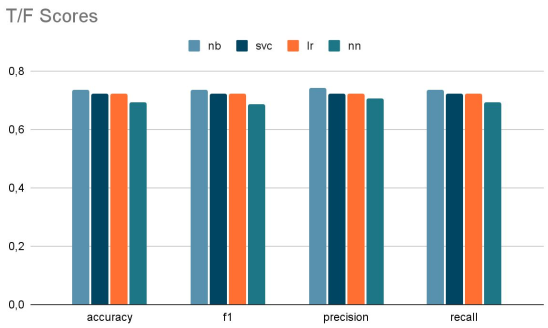
- **Naive Bayes (NB):** trained on TF-IDF features
- **Linear SVC (SVC):** trained on PCA features
- **Logistic Regression (LR):** trained on PCA features
- **Multilayer Perceptron (NN):** trained on PCA features

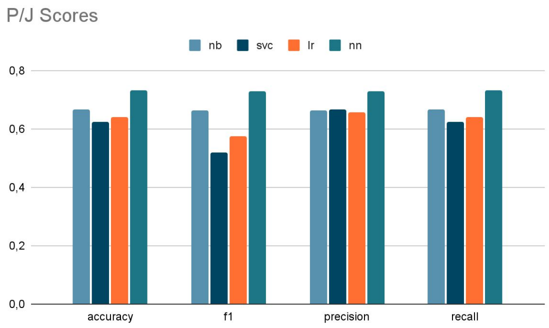
Through “`pyspark.ml.CrossValidator`”module, an hyperparameters tuning phase has been done on all the models used.

Results - Grouped posts approach





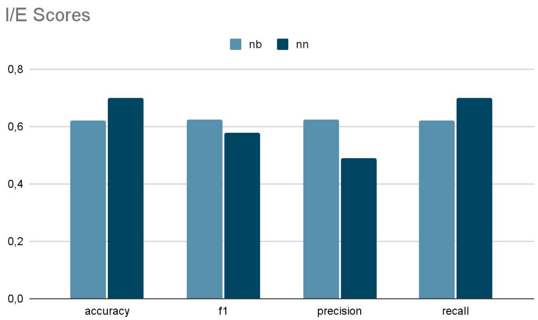


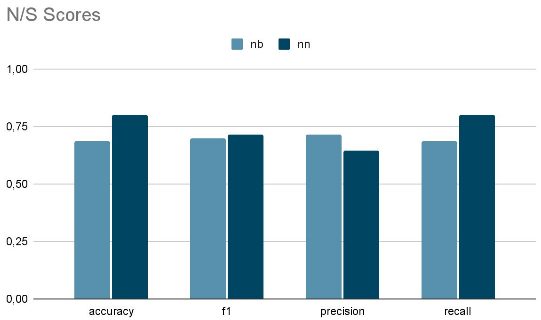


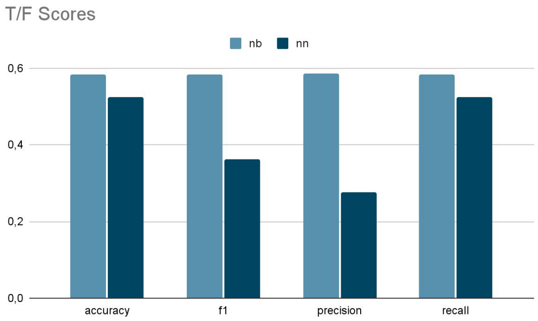
Grouped posts approach final model

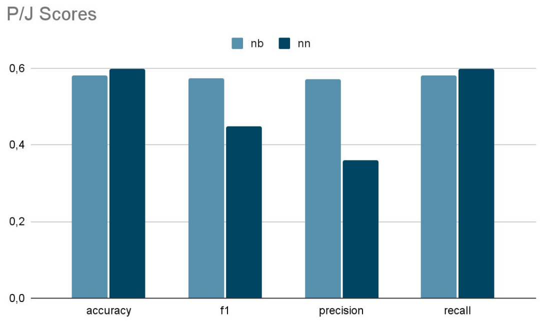
Model	F1-Score	Parameters	Indicator
Naive Bayes	0.74146	❏ `modelType="Multinomial"`	IE
Neural Network	0.83143	❏ `layers=[100, 200, 200, 100, 32, 16, 2]` ❏ `stepSize=0.001` ❏ `maxIter=100` ❏ `solver="l-bfgs"`	NS
Naive Bayes	0.73386	❏ `modelType="Multinomial"`	TF
Neural Network	0.73016	❏ `layers=[100, 200, 200, 100, 32, 16, 2]` ❏ `stepSize=0.01` ❏ `maxIter=100` ❏ `solver="l-bfgs"`	PJ

Results - Single post approach









In **S.P.A.** I trained **only NB and NN** for then, after seeing results, decide whether to continue with this approach or not (training takes a **LOT of time** for this approach)

Which is the best approach?

- **S.P.A.** gives us lower scores w.r.t. **G.P.A.**
- Testing G.P.A.'s final model on the same test set we used for testing SPA model, we can notice that it reaches very similar results to SPA model.
- It makes no sense to continue with S.P.A.
- Notice also that higher scores (on both approaches) are achieved by NS indicator classifiers

Predicting Twitter users personality

Using **Twitter API** (through **Tweepy**) we can obtain last things posted by a specific user. I tested G.P.A. final model on users with a well known personality (according to)

User	Real personality	Predicted personality
@BarackObama	ENFJ	ENFP
@MichelleObama	INTJ	ENFP
@BillGates	INTJ	ENTJ
@ConanOBrien	ENTP	ENTJ
@TheRock	ENTP	ENFP
@morgan_freeman	INFJ	INFJ
@TheElliotPage	INTP	ENTP

Final considerations and possible improvements

- Taking into account that dataset's labels are provided by what **Reddit**/**Personalitycafe** users think its their personality, we can assume that with high chances there's noise in our dataset.
- However, it is very unlikely that a user provides all 4 indicators wrong (i.e. provide its "opposite" personality)
- Possible improvements can be given by:
    - generalizing our model by **training** it also **on Twitter/Facebook/other social posts**
    - training different models, such as a more advanced Neural Network
    - training our dataset on users with a "verified" personality