Formation Machine Learning

# Projet 5

# Catégorisez automatiquement des questions

stackoverflow

Maud Charbonneau

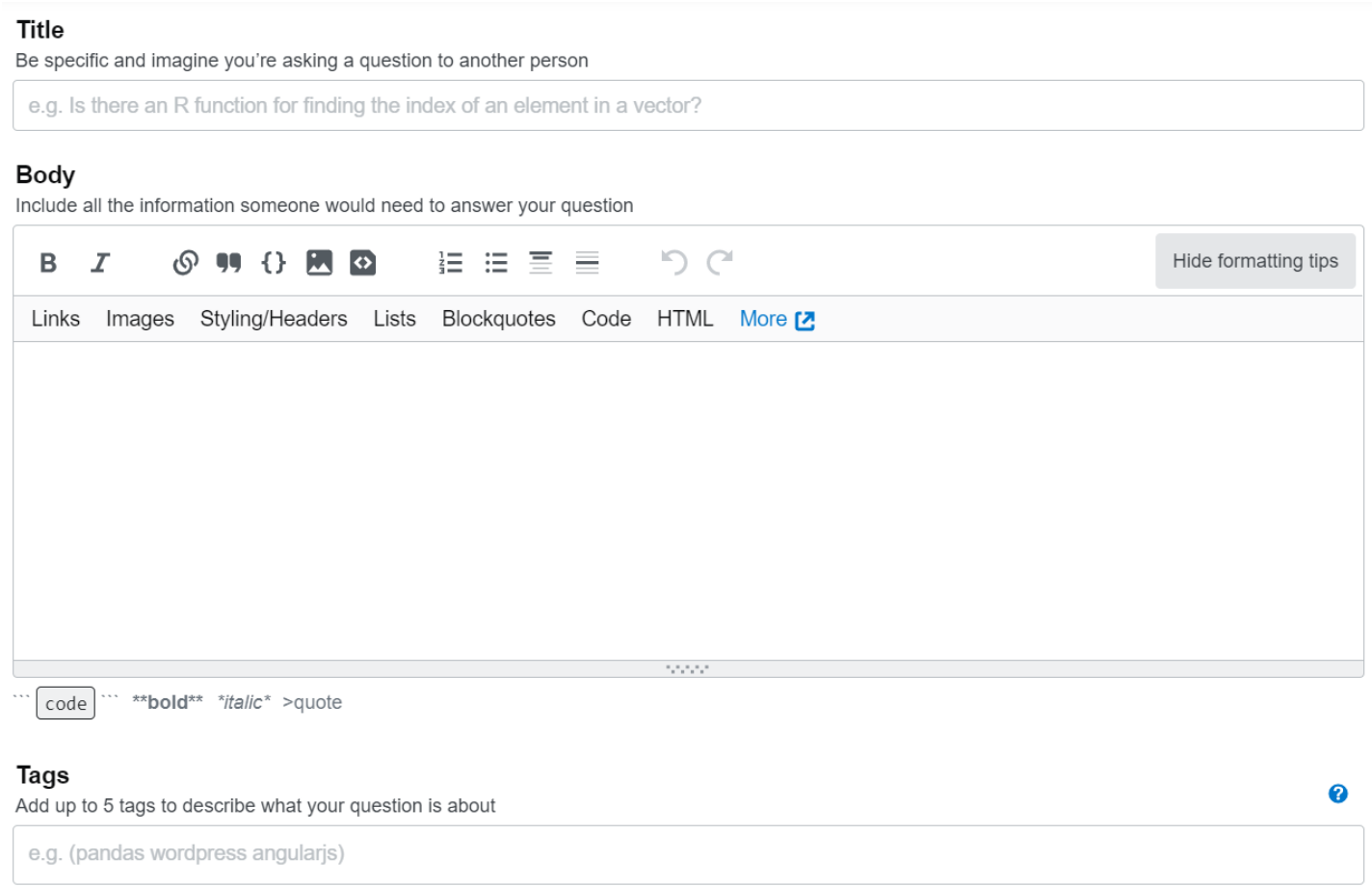
# Sommaire

1. Présentation du sujet et des données
2. Cleaning et exploration
3. Modélisation
4. Modèle final
5. API
6. Conclusion

# Présentation du sujet

- site de questions-réponses
- programmation informatique
- questions repertoriées par tags

- plus de 20 millions de questions
- difficultés à trouver des questions déjà posée



# Présentation des données

- Titre + corps + tags non nuls
- Plus de vus
- 255 000 données → Réduction à 60 000 pour faciliter les tests

| | Id | Score | ViewCount | Body | Title | Tags | AnswerCount | CommentCount | FavoriteCount |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 26477388 | 23 | 5693.0 | In SBT is the use of aggregate fol... | Is the use of 'aggregate' following by 'depend... | <build><sbt> | 1.0 | 0 | 9.0 |
| 1 | 20580028 | 22 | 37747.0 | I'm designed a <a href="http://en.wikipedia... | Flowchart "for each" loop loop without variabl... | <flowchart> | 5.0 | 1 | 2.0 |
| 2 | 15096219 | 22 | 21065.0 | I'm trying to register a new log \n\n<pr... | How to create a folder (if not present) with L... | <ruby-on-rails><ruby><logging> | 3.0 | 0 | 4.0 |
| 3 | 16853747 | 87 | 11309.0 | <pre><code>class Test{\n public static void... | Static block in Java not executed | <java><static><access-modifiers> | 5.0 | 2 | 26.0 |
| 4 | 2036744 | 27 | 21402.0 | I have to write some code in ML and it is m... | ML IDE and Compiler for Windows or Linux or Mac | <ide><compiler-construction><programming-langu... | 6.0 | 0 | 17.0 |

# Cleaning

## Cleaning de la question

Exemple:

*“Hello these are 3 examples to show the different steps of the cleaning process.”*

- Titre + corps
- Balises html
- Tokenisation
- Suppression des caractères spéciaux
- Convertion des majuscules en minuscules
- Suppression des stopwords
- Lemmatisation

[“hello”, “these”, “are”, “examples”, “to”, “show”, “the”, “different”, “steps”, “of”, “the”, “cleaning”, “process”]

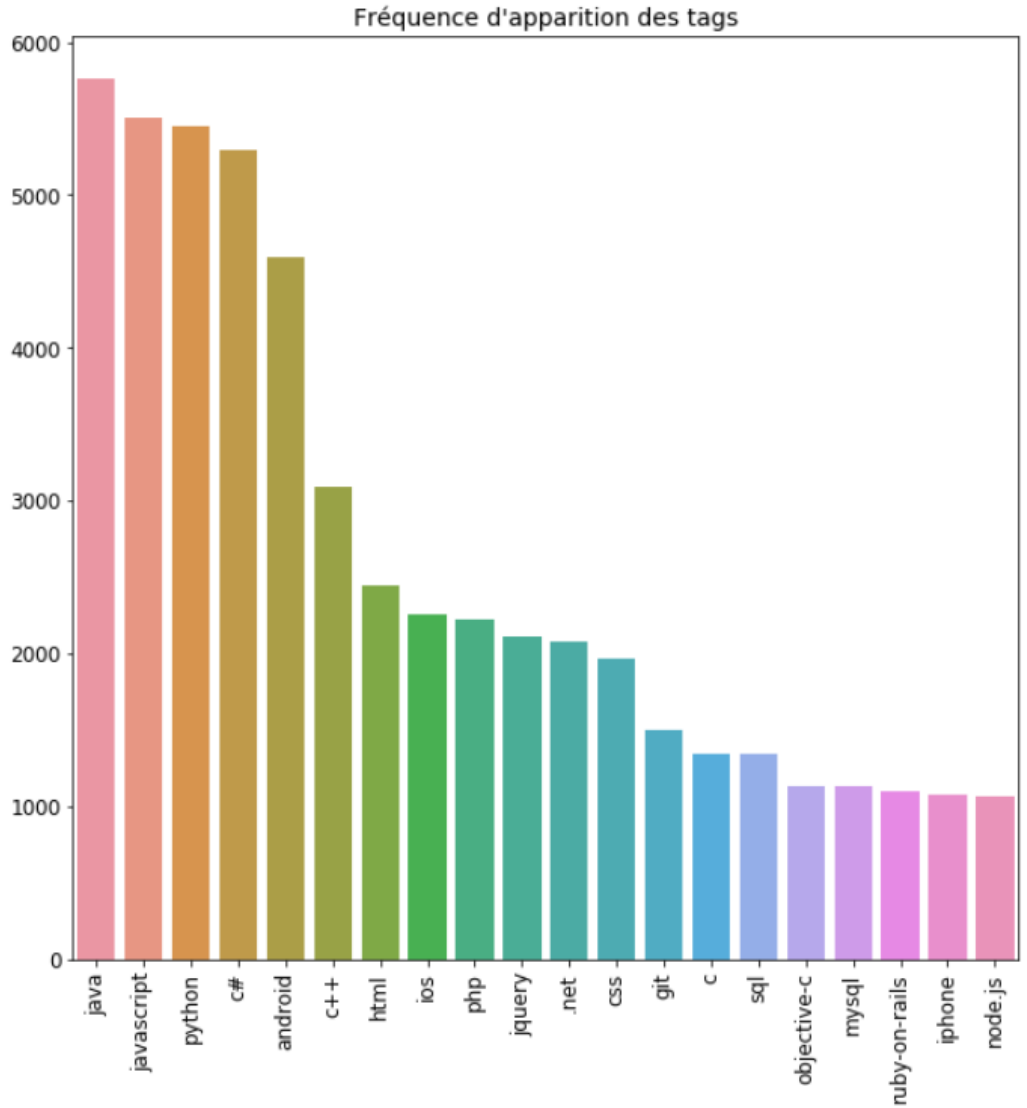
[“hello”, “examples”, “show”, “different”, “steps”, “cleaning”, “process”]

[“hello”, “example”, “show”, “different”, “step”, “cleaning”, “process”]

# Cleaning

## Cleaning des tags

- Suppression des caractères spéciaux
- Sélection des N tags les plus utilisés
  N = 20

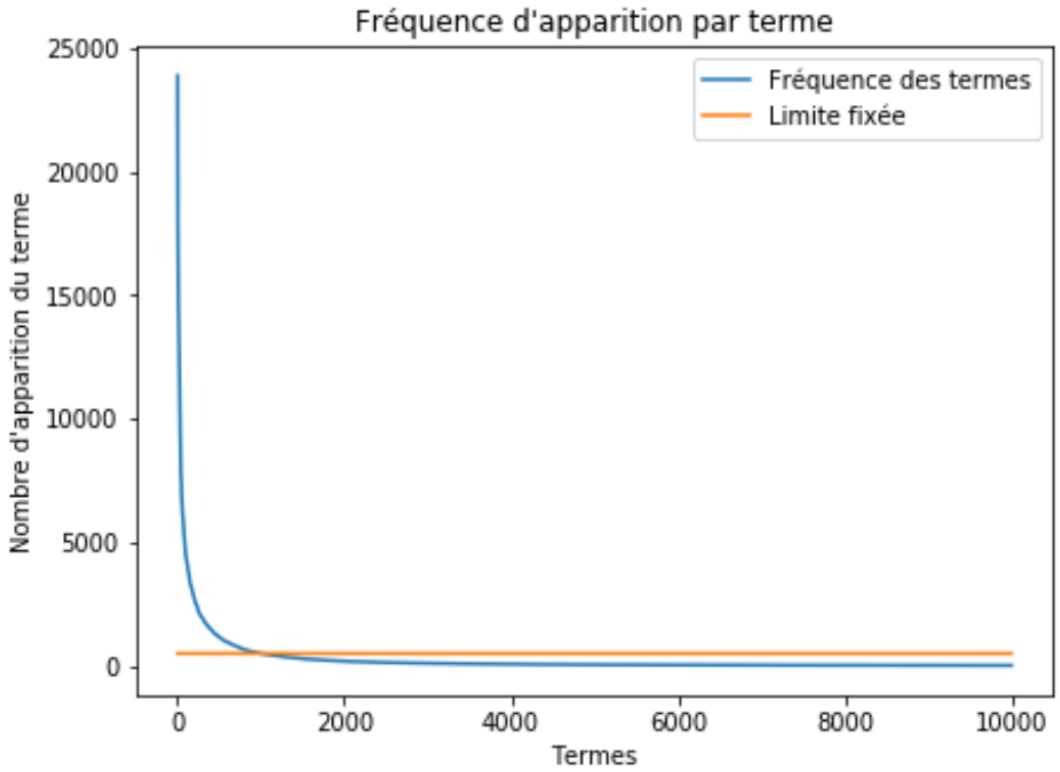


# Exploration

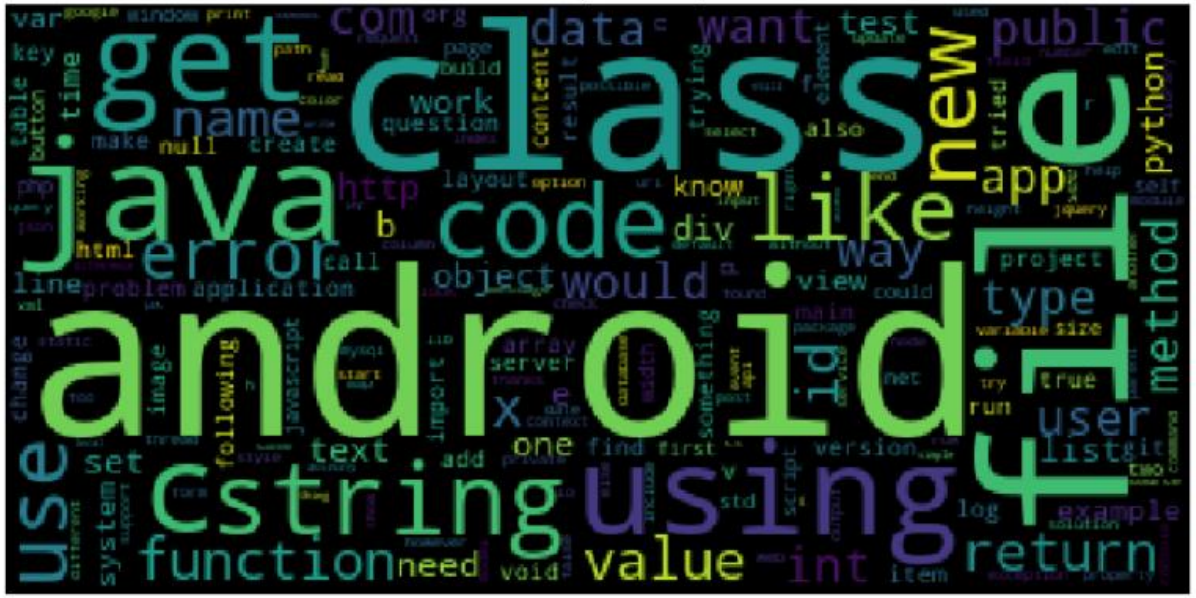
## Fréquence des termes

Limite = 500

Si fréquence(mot) < 500 --> Suppression mot




Termes les plus fréquents




Affichage des mots les plus fréquents

# Exploration

## Tf-Idf

Permet d'évaluer importance d'un terme dans une question, relativement à l'ensemble des questions

*poids=fréquence du terme × idf(terme)*

Transforme en matrice avec les mots de toutes les questions:

| | able | accept | access | according | account | achieve | across | action | active | activity | ... | xcode | xml | xmlns | year | yes | yet | z | zero |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.010169 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.013161 | 0.0 | 0.000000 |
| 1 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.015352 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.015575 | 0.0 | 0.000000 |
| 2 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.000000 |
| 3 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.012778 |
| 4 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.000000 |

Matrice train: 34802x997

# Exploration

## Analyse en Composantes Principales

But: réduction de dimension

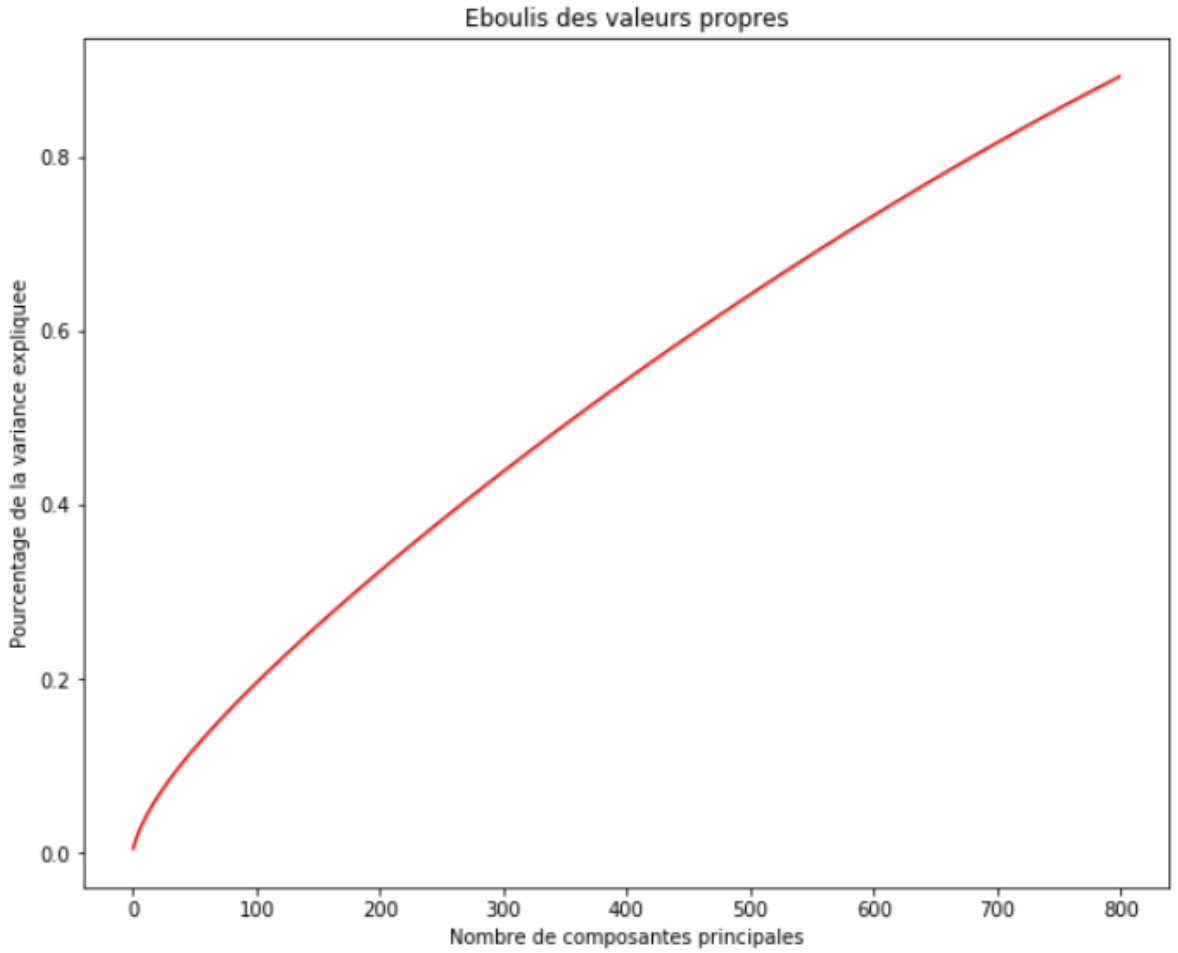
Nombre de variables:

- Avant ACP: 977
- Après ACP: 800

} Pourcentage de la variance expliquée

Affichage de l'éboulis des valeurs propres

On choisit 80% de la variance expliquée comme limite



# Modélisation

## Modélisation non supervisée

Latent Dirichlet Allocation ⟶ But: visualiser mots-clés

Chaque mot ∈ distribution par thème ⟹ chaque document ⟵ mots-clés pertinents

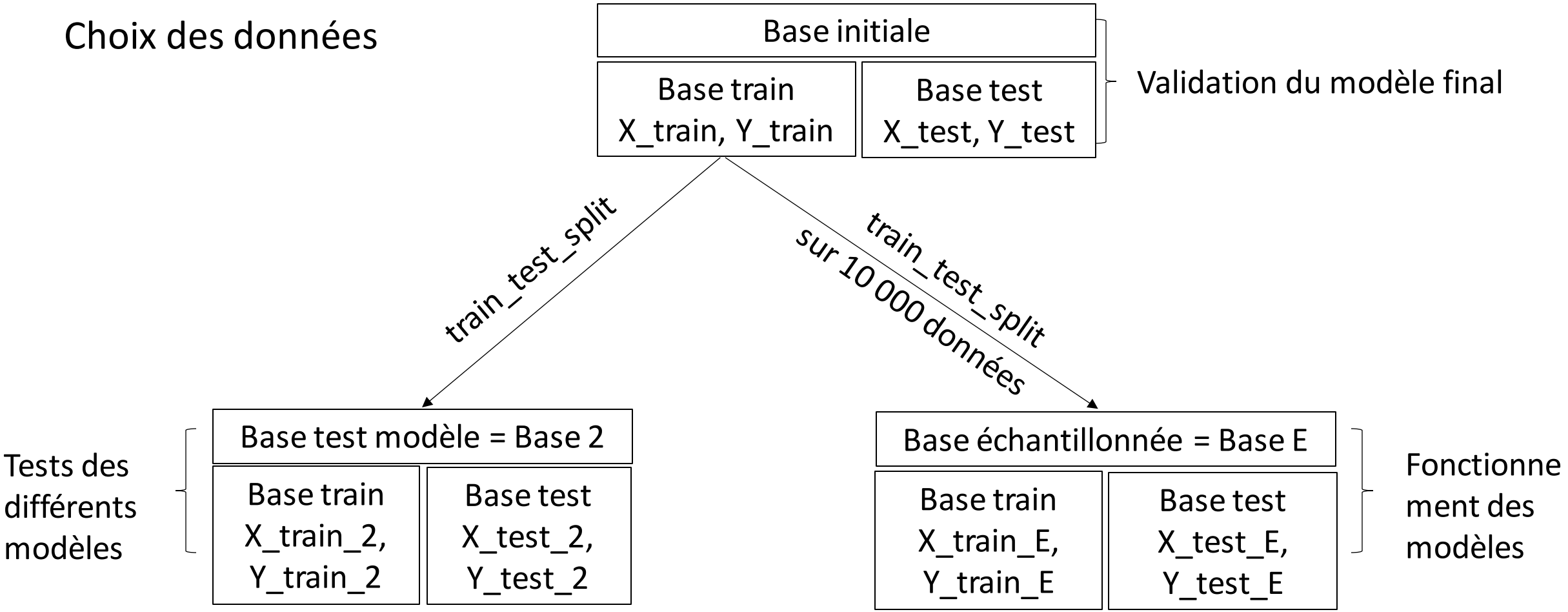
```
Topics in LDA model:
Topic #0:
use javascript using would page like code way file j know html browser c jquery function window application one google need exa
mple time want image work question good java find script web chrome event used user library get net looking
Topic #1:
public thread class method exception static void task java catch private new system difference async throw test interface null
println try return string block queue object code main call e final instance wait console boolean writeline args run override i
nt
Topic #2:
java android org jar eclipse com gradle maven spring annotation dependency hibernate lang junit xml bean compile build support
google plugin apache sun class project util internal springframework version activitythread servlet jdk test http main error so
urce property v groupid
Topic #3:
android view layout button image activity app self color intent item text id io screen height swift fragment want width parent
r background set change textview action bar programmatically new dialog animation notification show drawable size application k
eyboard like menu
```

Manque de précision du sujet

# Modélisation

## Modélisation supervisée

### Choix des données



# Modélisation

## Modélisation supervisée

### Modèles testés et choix du modèle

Modèles:

- Arbre de décision
- Régression logistique
- SVM

Moyen de mesure choisi:

**F-score:** mesure la capacité du système à donner toutes les solutions pertinentes et à refuser les autres.
Compromis entre précision et rappel.

| | **Arbre de décision** | **Régression logistique** | **SVM** |
|---|---|---|---|
| F-score | 0,39 | 0,66 | 0,68 |
| Temps d'exécution (en sec) | 101.86959886550903 | 24.22258186340332 | 1478.547325372696 |

Meilleur score = SVM MAIS temps d'exécution + 60x > temps d'exécution régression logistique.

Donc modèle choisi = régression logistique

# Modélisation

## Tuning des paramètres

Paramètres séléctionnés:

- Penalty : norme pénalisation
- C: inverse de la force de régularisation
- Max_iter: nombre d'itérations maximum pour que les solveurs convergent

Paramètres par défaut:

- Penalty = l2
- C = 1
- Max_iter = 100

  - Score = 0,66

GridSearchCV →

Meilleurs paramètres:

- Penalty = l2
- C = 0,01
- Max_iter = 100

  - Score = 0,70

# Modèle final

```
model_final = LogisticRegression(penalty='l2', C=0.01, max_iter=100)
```

Utilisation de la base initiale:

- .fit sur X_train et Y_train
- .predict sur X_test
- Compare prédiction et Y_test

F-score obtenu = 0,71

F-score initial = 0,66
Amélioration du score de 0,05

# API

## 1.3 Entrée de l'utilisateur

```
# User's title input
title = input("Title: ")
```

```
Title:
```

```
# User's body imput
body = input("Body: ")
```

```
Body:
```

Utilisateur entre le titre et le corps

```
# User's title input
title = input("Title: ")
```

```
Title:
```

## 1.4 Affichage du tag

```
# Cleaning and processing
features = pipe.transform(df_question['Question'])
mx_feature = pd.DataFrame(features.toarray(), columns = feature_names)
std_features = std_scale.transform(mx_feature)
acp = pca.transform(std_features)

# Prediction
predicted_tags = model_final.predict(acp)

# Print the tags
print(tags[predicted_tags])
```

```
['git']
```

# Conclusion

Modèles supervisés et non supervisés, meilleur résultat : **Régression logistique**

Avantages:

- Rapidité
- Précision

Désavantage:

Autres modèles plus précis (SVM)

Améliorations possibles:

Modifications des hyperparamètres

Tous les livrables de ce projet sont accessibles ici : https://github.com/maudch96/Projet5