Projet chef d’oeuvre

tagnow

Bastien Roques

Introduction

Suite à 4 mois de cours intensifs, j'ai effectué 15 mois d'alternance en entreprise.
Sanofi l'entreprise qui m'a accueillie est une entreprise transnationale française dont les activités incluent la pharmacie et les vaccins.

Pendant cette période en entreprise, j'ai été confronté à plusieurs problématiques métiers notamment concernant le service desk.
Le domaine de la pharma étant sujet à une politique de confidentialité des données, il m'est impossible de présenter un projet issu de mon travail auprès d'eux. C'est pourquoi j'ai choisi de créer un projet fictif mais ayant pris racine auprès de mon entreprise.

Problématique

Les développeurs soumettent régulièrement des tickets d'une manière qui a du sens pour eux mais lorsque d'autres personnes ne peuvent pas facilement identifier le contenu, cela devient un problème.
Comment faciliter le traitement des tickets sans que cela soit une contrainte de temps pour les utilisateurs ?

Je propose de réaliser une application (Service Desk) proposant automatiquement un tag au ticket soumis par l'utilisateur afin qu'il soit plus pertinent, et de ce fait rendre son analyse moins complexe.

L'application sera réalisé en Python sous forme de Web App comprenant :
- une base de données SQL pour stocker les différents tickets postés
- un espace user permettant de soumettre un ticket
- un espace support permettant de :
    - requêter les données enregistrées et de modifier les tags erronés
    - mettre à jour la base données

Analyse des besoins

On distingue deux types d'utilisateurs : les développeurs qui cherchent une réponse à un problème rencontré et le service support qui va vérifier la conformité des tickets soumis.

En tant que développeur, l'application doit me permettre de soumettre un ticket et de le baliser à ma place.
En tant que support, l'application doit me permettre d'accéder aux différents tickets balisés pour les analyser et de les modifier si nécessaire.

Etat de l'art

Le traitement du langage naturel ou NLP (*Natural Language Processing*), est généralement défini comme la manipulation automatique du langage naturel, comme la parole et le texte, par un logiciel.

L’analyse de texte moderne est désormais très accessible à l’aide de Python et d’outils open source, qui permettent de découvrir comment analyser les données textuelles.
Les données textuelles dont les formats sont pour la plupart non structurés, ne peuvent pas être représentées sous forme de tableau. Par conséquent, il est essentiel de les convertir en fonctionnalités numériques car la plupart des algorithmes d’apprentissage automatique sont capables de traiter uniquement les nombres.

Gestion de projet

Méthode

Pour la gestion de projet, nous avons choisi de nous inspirer de la méthode Kanban qui permet de visualiser facilement l’avancement du projet sous forme de tableau avec l’outil Trello. Chaque étiquette représentant un domaine de compétences.

Le tableau Kanban est divisé en quatre parties :

- **TODO** : Fonctionnalités en attente
- **WIP (Work In Progress)** : Fonctionnalité en cours de réalisation (2 max)
- **Testable** : Fonctionnalités testables par un lead développeur
- **DONE** : La fonctionnalité à été testé et validé par le lead développeur

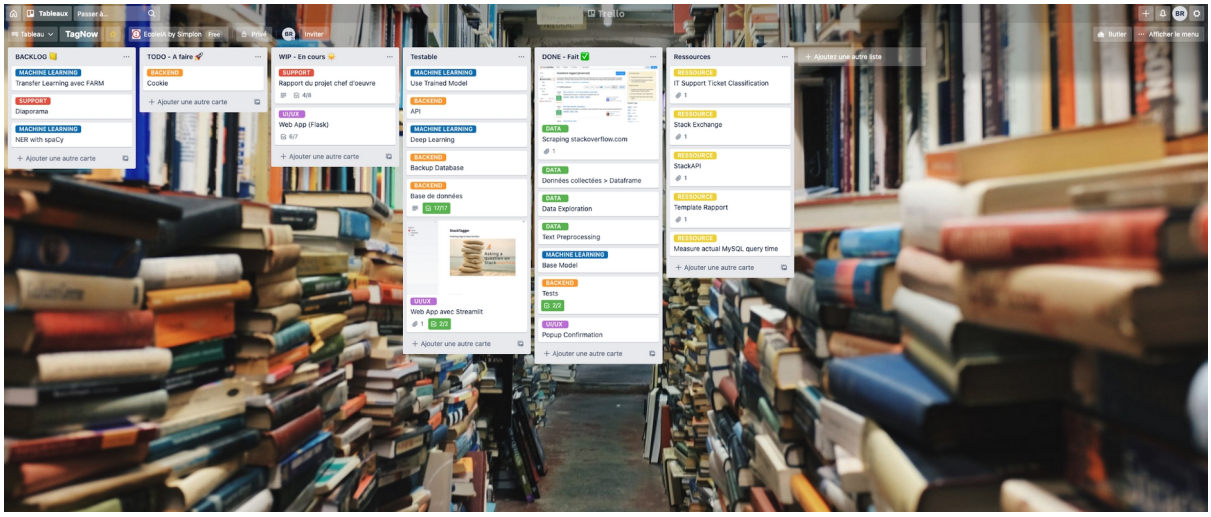
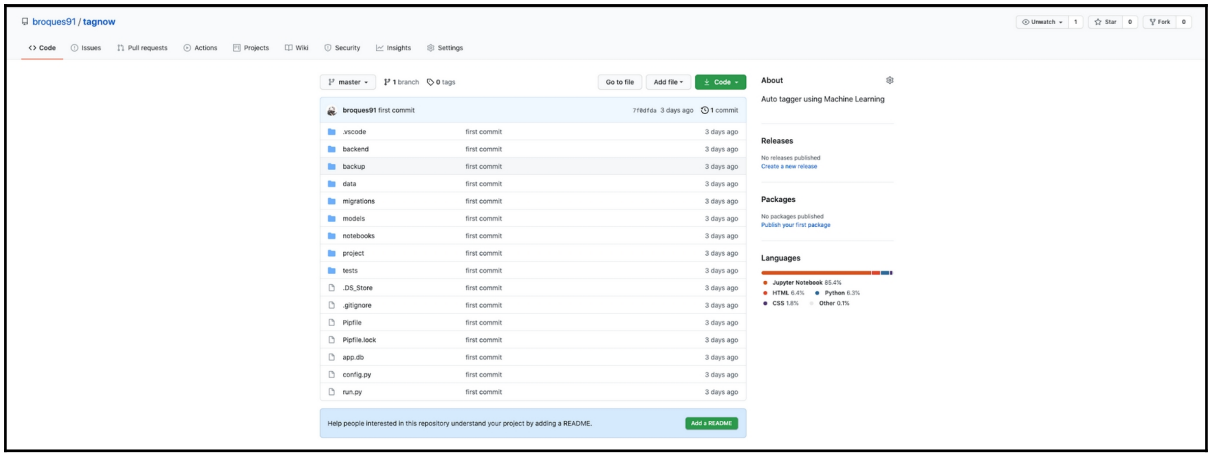


Tableau Kanban projet tagnow

Versionning

Pour le versionning du projet, le choix s’est orienté vers la plateforme GitHub pour sa facilité d’utilisation mais aussi car elle offre aussi un service de suivi de problèmes (issue tracking system) qui peut s’avérer utile lors de l’utilisation de plusieurs librairies dans un projet.



Intelligence artificielle

Dataset

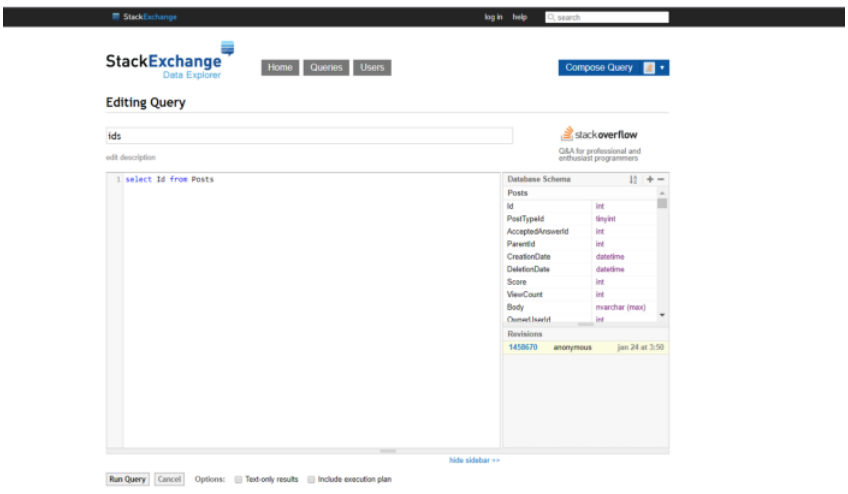
Nous avons choisi d'extraire des données du site Stack Overflow, un site qui permet à la communauté de développeurs du monde entier d'échanger sur des sujets liés au développement informatique.
Un utilisateur pose une question en renseignant un formulaire contenant 3 champs :

- Title : Titre de la question
- Body : la question
- Tags : liste de tags synthétisant le sujet de la question

Acquisition des données

Plusieurs méthodes sont possibles :

- L'idée première était de scraper le site stackoverflow.com mais nous nous sommes confrontés à une problématique de redondance des identifiants étant les mêmes pour des questions différentes.
- Requête SQL à cette page [Stack Overflow Exchange] et exporter le résultat au format csv.



StackExchange

- Requête API de Stack Exchange API avec un résultat au format json.

Nous avons choisi la requête pour faciliter l'intégration dans le code :

```
SITE = StackAPI('stackoverflow')
SITE.max_pages=10
questions = SITE.fetch('questions', min=20, tagged='java', sort='votes', filter='withbody')
```

Exemple : Sélection des questions labellisées en java

Nous avons choisi de sélectionner les cinq tags les plus populaires au sein de l'entreprise pour constituer notre jeu de données : c#, java, javascript, php et python.

Exploration

Le dataset brut est composé de 5000 données et de 21 colonnes

	tags	owner	is_answered	view_count	protected_date	accepted_answer_id	answer_count	score	last_activity_date	creation_date	...	question_id	content_license	link	title	body	locked_date	community_owned_date	closed_date	closed_reason	migrated_from
0	['javascript', 'arrays']	{'reputation': 102517, 'user_id': 364969, 'use...	True	7943032	1.418674e+09	5767357.0	105	9236	1616750777	1303597038	...	5767325	CC BY-SA 4.0	https://stackoverflow.com/questions/5767325/ho...	How can I remove a specific item from an array?	<p>I have an array of numbers and I'm using th...	NaN	NaN	NaN	NaN	NaN
1	['javascript', 'jquery', 'dom', 'visibility']	{'reputation': 120145, 'user_id': 21709, 'user...	True	2820858	1.578134e+09	178450.0	60	8043	1611315133	1223384598	...	178325	CC BY-SA 4.0	https://stackoverflow.com/questions/178325/how...	How do I check if an element is hidden in jQuery?	<p>Is it possible to toggle the visibility of ...	NaN	NaN	NaN	NaN	NaN
2	['javascript', 'syntax', 'jslint', 'use-strict']	{'reputation': 89473, 'user_id': 25847, 'user_...	True	1121081	1.370841e+09	1335881.0	28	7801	1615505646	1251303013	...	1335851	CC BY-SA 3.0	https://stackoverflow.com/questions/1335851/wh...	What does "use strict" do in JavaScr...	<p>Recently, I ran some of my JavaScript code ...	NaN	NaN	NaN	NaN	NaN
3	['javascript', 'jquery', 'redirect']	{'reputation': 101315, 'user_id': 44984, 'user...	True	6666709	1.447334e+09	506004.0	58	7714	1601182184	1233579256	...	503093	CC BY-SA 3.0	https://stackoverflow.com/questions/503093/how...	How do I redirect to another webpage?	<p>How can I redirect the user from one page t...	1.522911e+09	NaN	NaN	NaN	NaN
4	['javascript', 'function', 'variables', 'scope...	{'reputation': 509742, 'user_id': 9951, 'user_...	True	1498407	1.316450e+09	111111.0	86	7631	1611666531	1222006327	...	111102	CC BY-SA 3.0	https://stackoverflow.com/questions/111102/how...	How do JavaScript closures work?	<p>How would you explain JavaScript closures t...	1.497372e+09	1.370514e+09	NaN	NaN	NaN

5 rows × 21 columns

jeu de données brut

Nous n'avons conservé que les colonnes pertinentes pour la suite du projet.
A savoir :

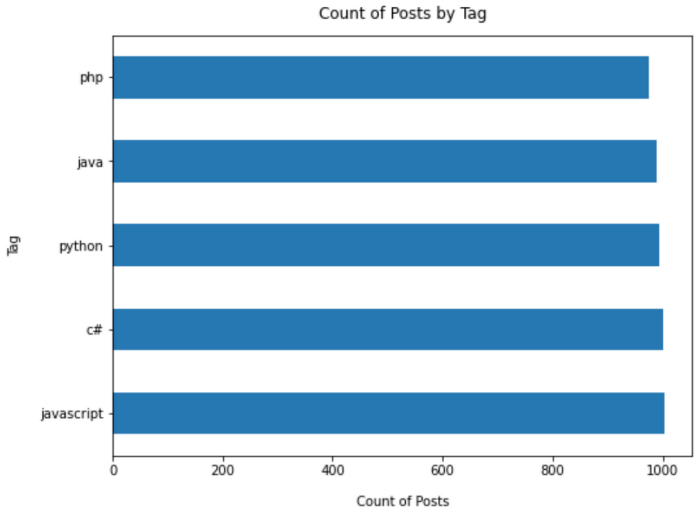
- question_id : identifiant unique de la question
- title : titre de la question
- body : question
- tags : tags de la question

	question_id	title	body	tags
0	5767325	How can I remove a specific item from an array?	<p>I have an array of numbers and I'm using th...	['javascript', 'arrays']
1	178325	How do I check if an element is hidden in jQuery?	<p>Is it possible to toggle the visibility of ...	['javascript', 'jquery', 'dom', 'visibility']
2	1335851	What does "use strict" do in JavaScr...	<p>Recently, I ran some of my JavaScript code ...	['javascript', 'syntax', 'jslint', 'use-strict']
3	503093	How do I redirect to another webpage?	<p>How can I redirect the user from one page t...	['javascript', 'jquery', 'redirect']
4	111102	How do JavaScript closures work?	<p>How would you explain JavaScript closures t...	['javascript', 'function', 'variables', 'scope...

jeu de données filtré

Target

La colonne tags est notre la colonne cible.
Sur 5000 observations, nous avons 4018 tags uniques.
Le premier tag de la liste est toujours le langage de programmation, les autres éléments étant des mots clés associés à ce langage.
Nous avons choisi de conserver uniquement le langage de programmation :



Suite à ce premier filtre nous obtenons une répartition des tags plus équilibrée avec environ 1000 observations par tag.

Préparation des données d'entraînement

Nettoyage des données

La tâche consiste à travailler le dataset pour qu'il puisse être exploité efficacement. Il faut harmoniser le texte au maximum pour faciliter la tâche de l'algorithme. Pour ce faire, nous allons effectuer les étapes suivantes :

- Remplacer toutes les majuscules par des minuscules
- Supprimer les caractères spéciaux (accents, ponctuation)
- Supprimer les balises html (exemple : …)
- Supprimer les "stop words", mots sans valeur ajoutée pour l'analyse globale du texte (exemple : "and", "I", "it", etc…)

Pour ce faire, nous avons utilisé la librairie NLTK qui fournit une liste par défaut des stopwords dans plusieurs langues et le module re pour les expressions régulières.

```python
REPLACE_BY_SPACE_RE = re.compile('[/(){}\[\]\|@,;]')
BAD_SYMBOLS_RE = re.compile('[^0-9a-z #+_]')
HTML_TAGS = re.compile(r'<.*?>')
STOPWORDS = set(stopwords.words('english'))

def clean_text(text):
    """
        text: a string

        return: modified initial string
    """
    text = text.lower()
    text = re.sub(REPLACE_BY_SPACE_RE," ",text)
    text = re.sub(HTML_TAGS,"",text)
    text = re.sub(BAD_SYMBOLS_RE,"",text)
    text = text.split();
    return ' '.join(_ for _ in text if _ not in STOPWORDS)
```

build_features.py

Ce pipeline nous permet d'avoir des posts à peu près propres. Cela permet au modèle d'analyser le texte plus efficacement.

Train-test split

Le découpage du jeu de données est une étape très importante afin d'éviter le risque de sur évaluer notre modèle (overfitting) ou tout simplement le contraire (under fitting). Nous découpons le jeu de données en deux :

- **Training set** : Il représente 70% des données. C'est sur ce jeu ci que le réseau va itérer durant la phase d'entraînement pour pouvoir s'approprier des paramètres, et les ajuster au mieux. C'est la phase d'apprentissage.
- **Test set** : Il va avoir pour rôle d'évaluer le modèle sous sa forme finale, et de voir comment il arrive à prédire comme si le réseau était intégré à notre application. Il est composé exclusivement de nouveaux échantillons, encore jamais utilisés pour éviter de biaiser les résultats. Celui-ci peut encore être estimé de l'ordre de 30% des données.

Nous utilisons la fonction train_test_split de SKlearn pour effectuer le fractionnement.
Le paramètre *test_size= 0,3* à l'intérieur de la fonction indique le pourcentage des données qui doivent être conservées pour les tests.

```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.3, random_state=42)
```

Une fois nos jeux de données d'entraînement et de test formatés, vient l'étape de transformation des données.

Feature engineering

Tokenization

C'est une technique de segmentation du texte, de cette façon chaque observation est transformée en un tableau de mots comme l'exemple ci-dessous :

```
This is what tokenization looks like!
```

```
from nltk.tokenize import sent_tokenize
text="Hello friends!. Good Morning! Today we will learn Natural Language Processing. It is very interesting"
tokenized_text=sent_tokenize(text)
print(tokenized_text)
```

```
['Hello friends!.', 'Good Morning!', 'Today we will learn Natural Language Processing.', 'It is very interesting']
```

Cependant, afin d'exécuter des algorithmes d'apprentissage automatique, il est nécessaire de convertir les fichiers texte en vecteurs de caractéristiques numériques.

Plusieurs méthodes existent :

Bag-of-words (bow)

Le modèle du "bag of words" ou “sac de mots” est couramment utilisé dans les méthodes de classification de documents.
Cela se résume à compter le nombre d’occurrences de chaque mot en attribuant un identifiant unique à chacun d’eux.
Chaque mot unique est alors utilisé comme une caractéristique (feature) pour la formation du classificateur.

Question		python	langage	programmation	java	ou
python langage programmation	→	1	1	1	0	0
java python ou python java		2	0	0	2	1

Nous utilisons la méthode CountVectorizer de la librarie Sklearn pour effectuer cette opération, elle permet la modification de beaucoup de paramètres si nécessaire.

```
from sklearn.feature_extraction.text import CountVectorizer
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train)
X_train_counts.shape
```

On procède à un fit sur les données d'entraînement et on applique ensuite la transformation sur les données de test.

TF-IDF (Term Frequency – Inverse Document Frequency)

Le but est ici aussi de convertir notre vecteur de chaînes de caractères en tableau de nombres.
Cependant, on fait l’hypothèse qu’un mot rare est plus discriminant.
C’est une mesure qui permet, à partir d’un ensemble de textes, de connaître l’importance relative de chaque mot en appliquant la formule suivante :

Tf-idf = fréquence du mot dans question(tf) * fréquence inverse du mot dans l’ensemble du corpus (idf)

Exemple :

Question		python	langage	programmation	java	ou
python langage programmation	→	1*1/3 = 1/3	1/1 = 1	1/1 = 1	0	0
java python ou python java		2*1/3 =2/3	0	0	2/2 = 1	1/1 = 1

Python et Java apparaissent le même nombre de fois.
Cependant, la valeur affectée à ‘java’ est supérieure à la valeur affectée à ‘python’ car au niveau du corpus java apparaît moins souvent. On a donc bien affecté un poids plus important à un mot plus rare.

Nous utilisons la méthode TfidfTransformer de la librarie Sklearn pour effectuer cette opération.

```
from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
X_train_tfidf.shape
```

Ici aussi, on procède d'abord à un fit sur les données d'entraînement puis on applique la transformation sur les données de test.

Mise en place des modèles

Approche supervisée : on a un problème de classification multi classes :
On a 5 classes possibles (tags). Chaque observation appartient à 1 classe. Chaque classe est affectée de 1 à K observations.

Nous avons utilisé des algorithmes classiques de classification : Support Vector Machine, Réseau de neurones, Régression logistique.

Baseline

C'est notre modèle de référence.
Il s'agit d'un modèle simple, qui est ensuite utilisé comme comparaison avec les modèles plus avancés que nous souhaitons tester.
Nous avons choisi l'algorithme de régression logistique :

```
from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression()
classifier.fit(X_train, y_train)
score = classifier.score(X_val, y_val)

print("Accuracy:", score)
```

```
Accuracy: 0.7782258064516129
```

La première ligne "*classifier = LogisticRegression()"* correspond à l'initialisation du modèle
La seconde correspond au fit (entrainement) du modèle sur le jeu de données d'entraînement.
La troisième correspond à l'évaluation du modèle sur les données de validation, on compare les tags prédits aux tag réels.
Ici notre baseline est donc d'environ 77% d'accuracy , l'objectif est d'améliorer ce résultat.

Linear Support Vector Machine avec TF-IDF

```
from sklearn.linear_model import SGDClassifier

sgd = Pipeline([('vect', CountVectorizer()),
                ('tfidf', TfidfTransformer()),
                ('clf', SGDClassifier(loss='hinge', penalty='l2',alpha=1e-3, random_state=42, max_iter=5, tol=None)),
               ])
sgd.fit(X_train, y_train)
```

Ici on utilise un pipeline afin d'intégrer les étapes de preprocessing avant l'entraînement du modèle. Cette méthode est aussi plus optimisée pour la sauvegarde du modèle, car toutes ces étapes sont enregistrées dans un seul fichier.

Réseau de neurones récurrent

La force des réseaux de neurones récurrents réside dans leur capacité de prendre en compte des informations contextuelles suite à la récurrence du traitement de la même information.

Le modèle que nous avons utilisé est un réseau récurrent formé d'une couche RNN (Recurrent Neural Network) et d'une couche MLP (Multilayer Perceptron). Un tel modèle s'implémente en quelques lignes avec le framework Keras de la librairie tensorflow.

On définit un modèle Sequential, sur lequel on ajoute une succession de couches qui correspondent à différentes étapes de la création du réseau. Les couches intermédiaires de Dropout sont une technique de régularisation, pour éviter au modèle de trop coller aux données vues en entraînement (overfitting) et améliorer sa généralisation.

La dernière couche MLP est de taille 5, de façon à ce que chaque neurone corresponde à une classe (un tag) à prédire. On normalise la prédiction via une fonction softmax pour obtenir une distribution de probabilité.

- Initialisation du modèle :

```
# Build the model
model = Sequential()
model.add(Dense(512, input_shape=(max_words,)))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes))
model.add(Activation('softmax'))
```

- max_words : le nombre maximum de mots sur lequel va s'entraîner le modèle
- num_classes : nombre de classes de tags (5)

- Compilation du modèle :

```
# Compile the model
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

- Entraînement du modèle :

```
# Fit the model
history = model.fit(X_train, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_split=0.1)
```

Résultats

Un problème de classification multi classes est assez simple à évaluer : ou on a juste, ou on a faux.
Une première métrique permettant de calculer cette performance est l'accuracy. Celle-ci permet de connaître la proportion de bonnes prédictions par rapport à toutes les prédictions. L'opération est simplement : Nombre de bonnes prédictions / Nombre total de prédictions.

Voici les résultats selon les configurations suivantes :

Feature Title

	Model	Accuracy
0	Naive Bayes	0.784946
1	Linear Support Vector Machine	0.802419
2	Logistic Regression	0.788306
3	Word2vec and Logistic Regression	0.210349
4	BOW Keras	0.766129

On constate que l'accuracy ne dépasse pas les 80% avec uniquement le titre de la question

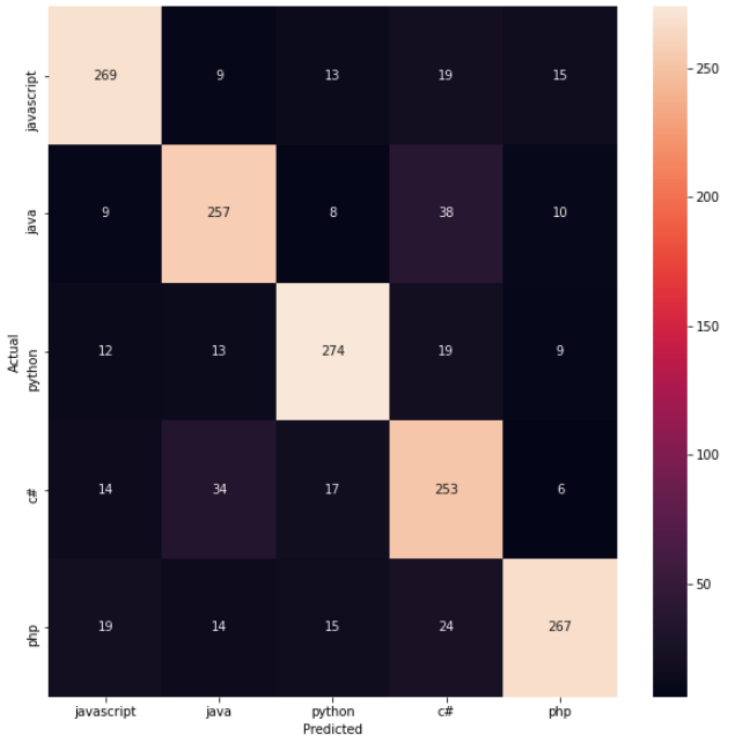
Feature Title + Body

	Model	Accuracy
0	Naive Bayes	0.838038
1	Linear Support Vector Machine	0.885081
2	Logistic Regression	0.860887
3	Word2vec and Logistic Regression	0.727151
4	BOW Keras	0.856855

En y ajoutant le body les résultats sont bien meilleurs, nous avons donc choisi cette option.

Un autre moyen simple de visualiser la performance du modèle est une matrice de confusion autrement appelée tableau de contingence.
Elle mettra en valeur non seulement les prédictions correctes et incorrectes, mais nous donnera surtout un indice sur le type d'erreurs commises.

Matrice de confusion



- **38** Observations **Java** ont été prédites comme étant du **C#**
- **34** Observations **C#** ont été prédites comme étant du **Java**
- **24** Observations **PHP** ont été prédites comme étant du **C#**

Nous constatons de suite que les langages C# et Java sont plus complexes à dissocier pour le modèle.

Classification Report

```
accuracy 0.8850806451612904
              precision    recall  f1-score   support

        java       0.86      0.79      0.82       305
          c#       0.89      0.86      0.87       301
         php       0.85      0.95      0.90       307
  javascript       0.92      0.89      0.91       284
      python       0.92      0.93      0.93       291

    accuracy                           0.89      1488
   macro avg       0.89      0.89      0.88      1488
weighted avg       0.89      0.89      0.88      1488
```

classification report : linear support vector machine

Une bonne accuracy n'étant pas toujours synonyme de bon modèle, le rapport précédent nous a permis d'évaluer d'autres métriques telles que la précision et le recall.
La précision nous indiquant quelle proportion pour chaque post taggé par notre modèle comme étant du python par exemple est vraiment du python.
Cela permet de mesurer le taux de faux positifs.

Le rappel *(recall)* correspond au nombre de tickets correctement attribués à la classe i par rapport au nombre total de tickets appartenant à la classe i.

Cela confirme notre première impression concernant le langage java, le modèle présente des résultats néanmoins encourageants.

Analyse

A l'échelle de phrases courtes comme le titre, la compréhension pour une machine reste assez complexe vu le contexte (certaines subtilités de langages restent difficiles à saisir). Néanmoins, avec un texte plus long, comme le body voire en concaténant les deux features, cela permet un gain considérable en termes d'accuracy.

Le modèle utilisé en production est le linear support vector machine car il présente les meilleurs résultats.
Cependant il sera toujours possible de modifier ce choix par la suite.

Application

Base de données

Le choix d'un SGBD (Système de Gestion de Bases de Données relationnelles) fut évident étant donnée la structure des données. Nous utilisons MySQL pour sa simplicité, sa robustesse et sa flexibilité.

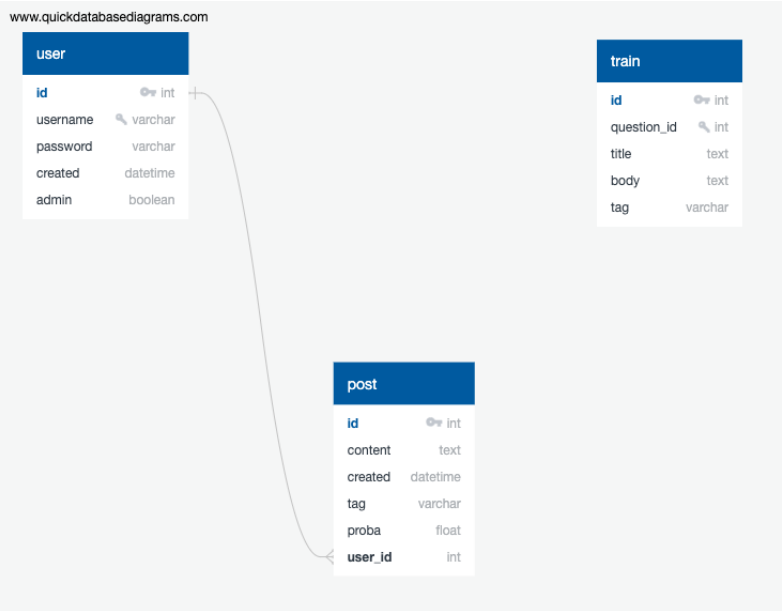
Par défaut Flask ne gère pas les bases de données, nous faisons appel à un l'ORM (Object Relational Mapping, ou Mapping objet-relationnel en français) SQLAlchemy, qui fera le pont entre la base de données et l'application.

Modèle

Notre base de données se compose de trois tables :

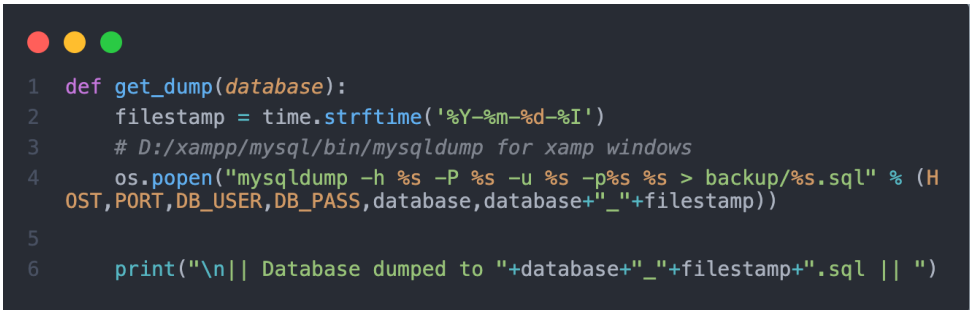
User	Post	Train
- id (int) - username (varchar) - password (varchar) - created_at (datetime) - admin (boolean)	- id (int) - question_id (varchar) - content (text) - created_at (datetime) - tag (tag) - proba (float)	- id (int) - question_id (varchar) - title (text) - body (text) - tag (varchar)

Le champ *question_id* étant l'identifiant unique de chaque question faisant référence à la base StackExchange contrairement au champ *id* qui constitue l'index dans notre base de données.



Backup

La mise en place d'un moyen de sauvegarde de la base de données était primordial afin de rassurer le client sur un éventuel crash de celle-ci.
Pour cela, il existe un script permettant une sauvegarde de la base de donnée sous forme d'export au format .sql, précisant à quelle date la sauvegarde a été effectuée.



```python
def get_dump(database):
    filestamp = time.strftime('%Y-%m-%d-%I')
    # D:/xampp/mysql/bin/mysqldump for xamp windows
    os.popen("mysqldump -h %s -P %s -u %s -p%s %s > backup/%s.sql" % (HOST,PORT,DB_USER,DB_PASS,database,database+"_"+filestamp))

    print("\n|| Database dumped to "+database+"_"+filestamp+".sql || ")
```

db_backup.py

Backend

Intégration de l'IA

Quand le client fait appel à l'endpoint "*/predict/*", le programme Python envoie une requête pour récupérer les données entrées par l'utilisateur dans le formulaire de soumission d'un post. On stocke alors ces informations dans des variables qui sont utilisées comme paramètres pour notre modèle.
On stocke alors la prédiction et ses probabilités dans de nouvelles variables reconduites dans la page html ou template afin qu'elles soient visibles par le client.

```python
model = joblib.load('models/model_svm.mod')
selected_model = "SGDClassifier"
prediction = model.predict([message])[0]
pred_proba = model.predict_proba([message])[0]

return render_template(
    'prediction.html',
    query=message,
    model=selected_model,
    prediction=prediction.capitalize(),
    predict_tag=prediction,
    proba=[round(proba*100, 2) for proba in pred_proba]
)
```

views.py

Le modèle ayant été sauvegardé sous forme de pipeline, il suffit de le charger et d'effectuer la prédiction du message de l'utilisateur.

Mise à jour de la base de données d'entraînement

```
# Collect Data
download_data.main()
# Clean Data
build_features.start_process()
# Read Data
df = pd.read_csv("data/processed/trainfull.csv", sep=";", index_col=0)
df = df[['question_id', 'title', 'body', 'tags']]
df['question_id'] = df['question_id'].astype(str)
# Get ID
train_ids = db.session.query(Train.question_id)
ids = train_ids.all()
indexes = [i[0] for i in ids]

for i in df.index:
    # Check Duplicate ID
    if i not in indexes:
        record = Train(
            question_id=df.at[i, 'question_id'],
            title=df.at[i, 'title'],
            body=df.at[i, 'body'],
            tag=df.at[i, 'tags'],
        )
        db.session.add(record)  # Add all the records
db.session.commit()  # Attempt to commit all the records
print("Time elapsed: " + str(time() - t) + " s.")
return redirect(url_for('database'))
```

Lors de l'importation des données collectées pour ensuite les insérer dans la base de données il y un processus d'identification des ID qui va permettre d'éviter les doublons.
Le programme va alors parcourir les ID de chaque post et insérer uniquement ceux qui ne sont pas déjà présents en base de données

Soumission d'un post utilisateur

La soumission d'un post est réalisée via un formulaire, les données sont alors collectées via une requête POST et insérées dans la base de données avec l'ORM SQLAlchemy.

```python
@app.route('/insert/', methods=['POST'])
@login_required
def insert():
    question_id = str(uuid.uuid4().hex[:6]).upper()
    message = request.form['message']
    model = request.form['model']
    tag = request.form['tag']
    pred_proba = request.form['proba']
    list_proba = pred_proba.strip(']][').split(', ')
    proba = [float(p) for p in list_proba]

    ticket = Post(question_id=str(question_id),
                  question=str(message),
                  pub_date=dt.now(),
                  model=str(model),
                  tag=str(tag),
                  proba=max(proba))
    db.session.add(ticket)
    db.session.commit()
    return redirect(url_for('index'))
```

Insertion d'un post dans la base de données

Frontend

cf annexes

Nous avons utilisé le framework CSS Bootstrap afin de rendre l'interface ainsi que l'expérience utilisateur plus agréable.
Le front de l'application est composé de plusieurs pages répertoriées dans les espaces suivants :

General

La page login va permettre à l'utilisateur de s'authentifier via un formulaire faisant lien avec la base de données.
Si l'utilisateur est loggé comme n'étant pas admin il sera redirigé vers la page d'accueil de l'application.
Si il est enregistré comme étant admin il sera redirigé vers l'espace support de l'application
Dans le cas où l'utilisateur ne possède pas de compte, il est possible d'en créer un nouveau via la page "signup"

User

La page d'accueil de l'application permet à l'utilisateur de soumettre un ticket via un formulaire. Lorsque l'utilisateur valide son post, une fenêtre popup de confirmation apparaît avec le post et le tag recommandé, cependant l'utilisateur peut via un menu déroulant modifier le tag sélectionné et voir la probabilité de chaque langage selon l'IA.

Support

L’espace support est accessible uniquement à un administrateur, celui-ci fournit un accès à trois pages :

- Posts : Exploration des tickets soumis par les utilisateurs
- Database : Exploration et mise à jour des données d'entraînement
- Model : Page permettant l’entraînement des modèles

Qualité et Monitoring

Pour s’assurer de la qualité de l’application , nous effectuons des tests.
Nous avons choisi d’opter pour des tests automatisés : leur but étant de confirmer que les différentes parties du programme correspondent aux spécifications techniques.

Test unitaire

Le test unitaire vérifie qu'une fonction réalise l'action souhaitée. Il ne s'agit pas ici de tester l'interaction entre les différentes fonctionnalités mais plutôt une partie d'une fonctionnalité.
Nous avons donc créé un test unitaire pour valider que la fonction create_user() a pour effet de créer un nouvel utilisateur dans la base de données.

Test fonctionnel

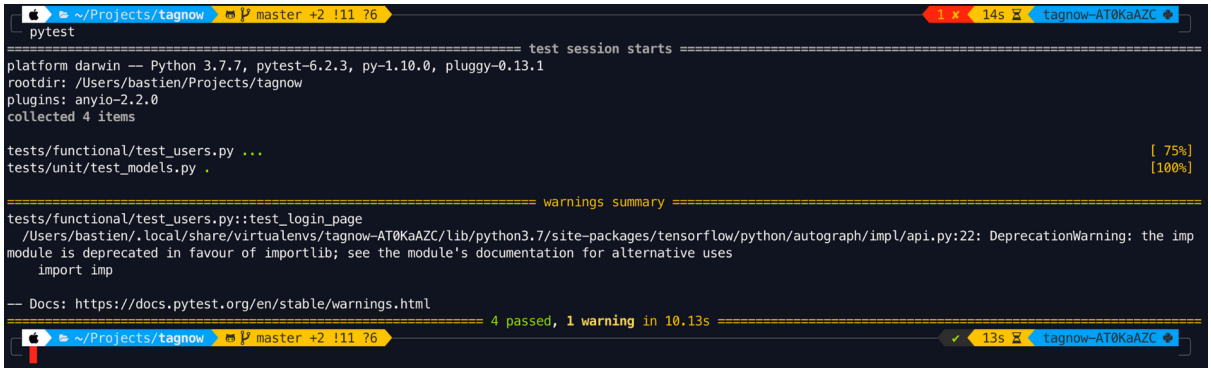
Les tests fonctionnels vont vérifier qu'une fonctionnalité, dans son ensemble, marche comme nous le souhaitons. Ils sont réalisés par l'ordinateur, car cela les rend plus rapides à exécuter, ou par un humain. Ils reprennent souvent un parcours utilisateur.

Nous avons créé un test fonctionnel pour la validation du processus d'ajout d'un utilisateur. Le test va alors vérifier qu'un utilisateur peut se connecter, accéder à la page d’accueil de l’application pour soumettre son ticket.

```python
def test_new_user(new_user):
    """
    GIVEN a User model
    WHEN a new User is created
    THEN check the username, password, created, and admin fields are defined correctly
    """
    assert new_user.username == 'broquestest'
    assert new_user.password == 'Simplon2021'
    assert new_user.created == datetime.now().date()
    assert new_user.admin == 0
```

test_models.py

Le module pytest permet alors d’effectuer ces tests de manière autonome.



pytest

Le monitoring consiste à suivre et analyser les performances du modèle déployé pour garantir une qualité acceptable telle que définie par le cas d'usage. Il fournit des avertissements sur les problèmes de performances, aide à diagnostiquer leur principale cause, à les déboguer et les résoudre.

Dans notre cas avec Flask le paramètre (debug=True) nous permet de suivre les différents logs de notre application.

```
 * Serving Flask app "project" (lazy loading)
 * Environment: development
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 287-064-795
127.0.0.1 - - [15/Apr/2021 20:15:55] "GET /admin/user/ HTTP/1.1" 200 -
127.0.0.1 - - [15/Apr/2021 20:15:59] "GET /admin/ HTTP/1.1" 200 -
127.0.0.1 - - [15/Apr/2021 20:15:59] "GET /static/sb-admin-2.css HTTP/1.1" 404 -
127.0.0.1 - - [15/Apr/2021 20:16:00] "GET /admin/post/ HTTP/1.1" 200 -
127.0.0.1 - - [15/Apr/2021 20:17:02] "GET /admin/user/ HTTP/1.1" 200 -
127.0.0.1 - - [15/Apr/2021 20:17:03] "GET /admin/train/ HTTP/1.1" 200 -
```

Bilan et axes d’amélioration du projet

Bilan

Le projet a été très stimulant car il m’a permis de rassembler les connaissances acquises tout au long de cette formation. Les résultats obtenus ont été au-delà de mes espérances, ce projet m’a aidé à me confronter aux différentes étapes de construction d’une application et je peux maintenant échanger plus aisément avec les différents acteurs : Data Analysts, Data Scientists et DevOps.

Points négatifs

La distinction entre certains langages reste complexe à mettre en place malgré le nombre de données collectées. On peut donc supposer un problème dû au traitement des données. Les algorithmes utilisés sont lourds et mettent donc du temps à tourner. Le fait de faire la partie machine learning en local (sur ma machine) a été fastidieux et souvent mon kernel a été interrompu.

Points positifs

La réalisation d’un projet machine end-to-end a été un vrai challenge. Les parties Data et Visualisation ont été les plus passionnantes pour moi et j'aurais aimé prendre plus de temps pour les améliorer.

La création de la WebApp a été la tâche la plus abordable que ce soit en compréhension ou en application. Le backend a été plus difficile à appréhender, mais à présent j’arrive à avoir plus de recul sur les problèmes que je peux rencontrer.
Je peux maintenant mieux les aborder en découpant ces problèmes en plusieurs tâches accessibles.

Organiser le processus de machine learning en plusieurs phases me conforte dans mon raisonnement initial d’application des méthodes agiles pour ce type de projet.

Axes d’améliorations

Partie Machine Learning :

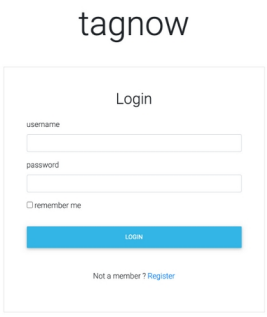
- Approche non supervisée
- Utilisation du transfert learning
- Utilisation des services cloud

Partie Web App :

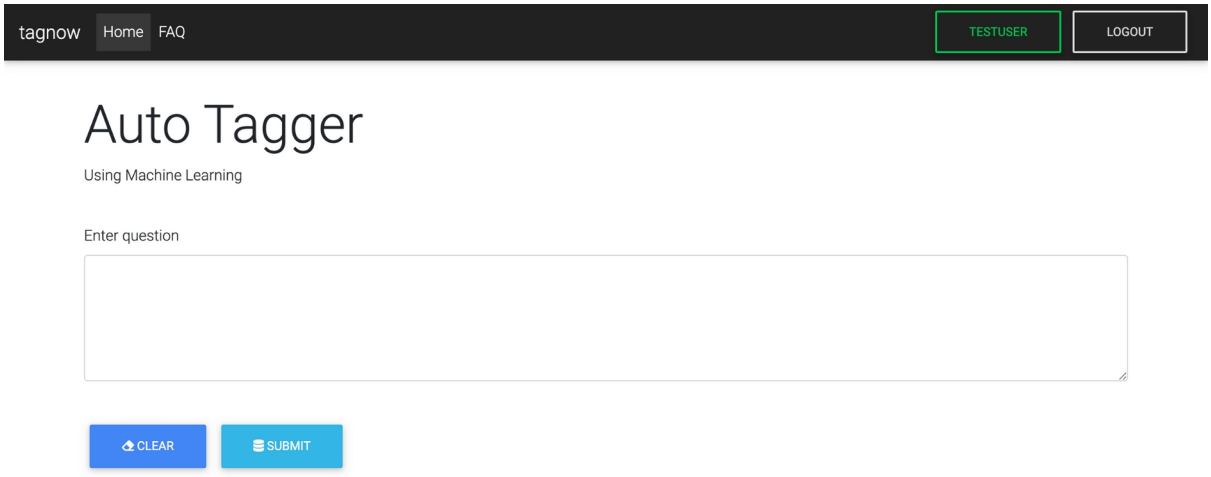
- Création d’une page report pour l’espace support
- Création d’une page posts pour l’espace user

Annexes

User

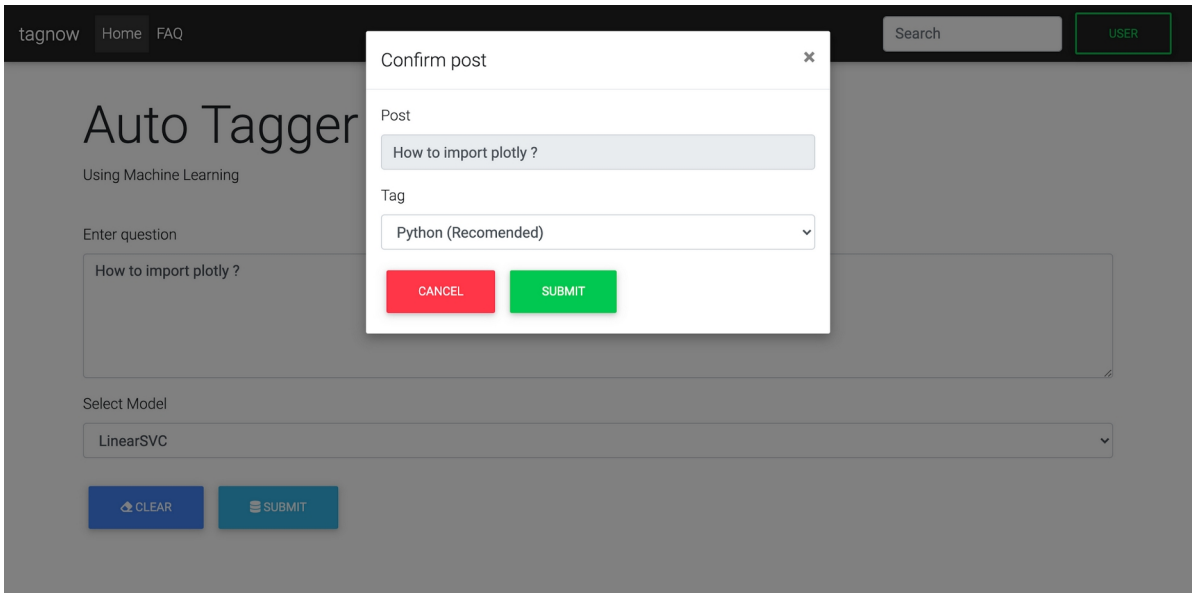


login.html

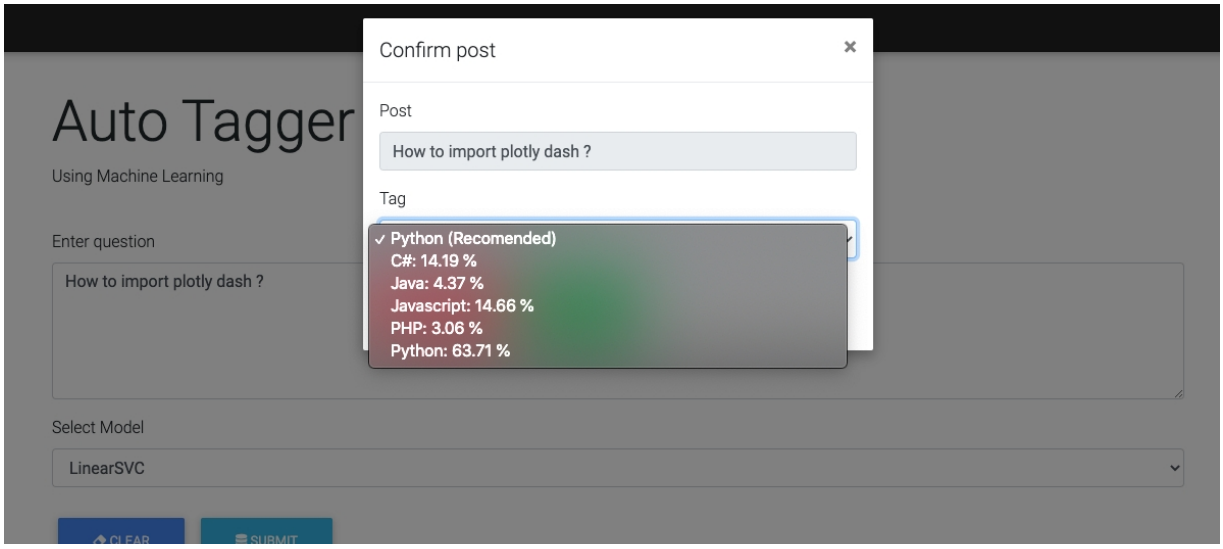


home.html

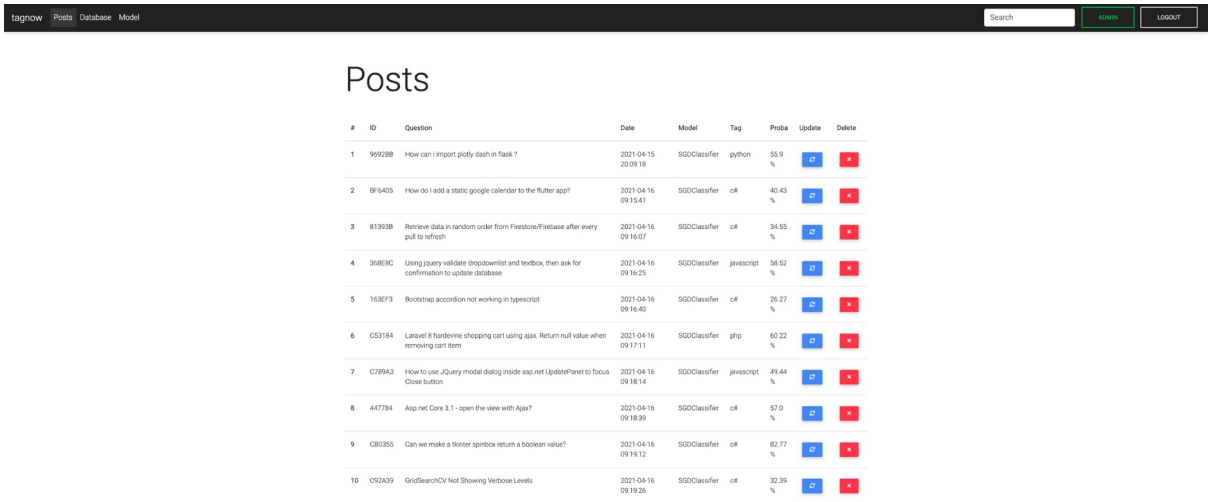
Prediction



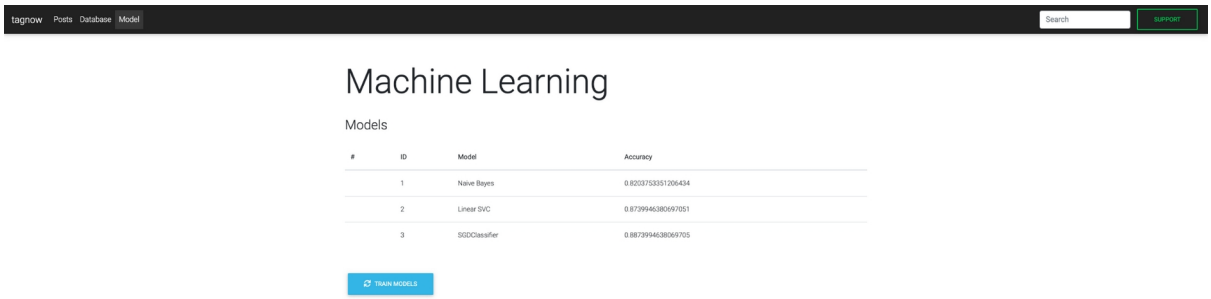
predict.html



predict.html



posts.html



model.html