Projet - Réalisation d'une interface d'analyse de données par apprentissage supervisé

Franck Doronzo, Afaf Ben Haj, Marie Vachet

Décembre 2021

Introduction

Dans le cadre de l'UE machine learning, nous avons travaillé sur la réalisation d'une interface web d'analyse de données par apprentissage supervisé, en python.

Nous verrons dans ce rapport quels sont les choix que nous avons fait afin de répondre au cahier des charges du projet. Nous proposerons également une démonstration de l'utilisation de notre package sur un jeu de données.

Ce rapport s'accompagne d'un lien GitHub sur lequel se trouvent le contenu du projet. Il est disponible sur le lien suivant : https://github.com/FD155/Supervised$_{L}earning_{I}nterface$

1 Choix des algorithmes

1.1 Objectifs

Comme explicité dans le cahier des charges le choix des algorithmes doit se faire entre "3 algorithmes de classification et 3 algorithmes de régression" nous avons donc décidé d'implémenter les algorithmes suivants:
Classification :

- l'analyse discriminante linéaire
- la méthode des k plus proches voisins
- la régression logistique

Régression:

- la régression linéaire
- arbre de décision
- la régression SVM
- les k plus proches voisins

1.2 Contraintes

Les contraintes principales sont présentées dans notre cahier des charges fourni dans le sujet. Ces dernières se résument en six points:

- les entrées seront uniquement des fichiers csv
- la sélection des variables cibles / explicatives doit être possible

- la possibilité de sélectionner plusieurs algorithmes à appliquer
- une validation croisée sera effectuée afin d'estimer les mesures d'évaluation du modèle et calculer le temps de calcul en sortie
- Utiliser Bokeh ou Dash pour la réalisation de l'application

1.3 Plan du projet

Notre projet s'est effectué selon plusieurs étapes qui sont les suivantes :

1. La création des fonctions des algorithmes d'apprentissage supervisé.
2. La réalisation de l'application Dash.
3. La fusion de l'application avec les fonctions.

2 Architecture de l'application

2.1 Composition

La composition de notre interface est divisée en plusieurs fichiers. Le premier fichier, "fonctions.py" répertorie l'ensemble des fonctions de machine learning. Le second fichier, nommé "callbacks", comporte toutes les fonctions callback utilisées pour lier les différents éléments de l'interface. Chaque callback est relié à des éléments html de la page "layout.py". Ensuite, nous avons réalisé un header avec une barre de navigation mais n'avons pas vraiment exploité son plein potentiel. Enfin les fichiers "index.py" et "app.py" servent à la construction de l'application et à son lancement sur le serveur.

2.1.1 Construction des algorithmes

Dans cette partie nous expliquerons le fonctionnement des deux premiers algorithmes d'apprentissage supervisé afin d'avoir une visualisation de la façon dont les algorithmes ont été construit.
Ce premier fichier fait appel à l'ensemble des algorithmes que nous avons décidé d'implémenter. Tout d'abord l'encodeur de variables qui permettra à l'utilisateur de pouvoir utiliser des algorithmes de classification peu importe le jeu de données :

```
def OneHotEncoder__(X):
    start = time()
    cat_ftrs = []
    for col in X.columns:
        if (is_string_dtype(X[col])):
            cat_ftrs.append(col)
    col_trans = make_column_transformer((OneHotEncoder(sparse=False,handle_unknown='ignore')
    ,cat_ftrs), remainder='passthrough')
    X_trans = col_trans.fit_transform(X)
    done = time()
    elapsed = done - start
    print(type(X_trans))
    print(elapsed)
    return(X_trans)
```

Listing 1: Fonction d'encodage variables qualitatives

Viennent ensuite les algorithmes de classification tels que l'analyse discriminante. Comme dans l'ensemble de nos algorithmes, les deux premiers paramètres représentent l'ensemble des variables explicatives (X) ainsi que la variable cible (y). On trouve ensuite les hyperparamètres correspondant à l'algorithme choisi,

ici le solver pour l'analyse discriminante linéaire. Quant au dernier hyperparamètre choisis" celui-ci fait référence à l'activation du GridsearchCV afin que l'utilisateur aie la possibilité de choisir les hyperparamètres manuellement ou qu'il puisse récupérer les meilleurs hyperparamètres automatiquement. En retour et pour l'ensemble de nos algorithmes nous retourneront les différentes métriques d'évaluation du modèle entrainé, comme spécifié dans le cahier des charges.

```
def adl(X,y,w_solver=None,nb_shrinkage=None,nb_components=None,w_covariance=None,choisis=
    False,params=None):
    start = time()
    X=OneHotEncoder__(X)

    if choisis==False:
        params = {"shrinkage":['auto',0.1,0.3,0.5,0.7,0.9]}
        grid = GridSearchCV(LinearDiscriminantAnalysis(solver='lsqr'),param_grid = params,cv
    =5)
        fit = grid.fit(X,y)
        best_param = fit.best_params_
        lda = LinearDiscriminantAnalysis(solver='lsqr')
        lda.set_params(**best_param)
    else:
        lda = LinearDiscriminantAnalysis(solver=w_solver,shrinkage=nb_shrinkage,n_components
    =nb_components,store_covariance=w_covariance)

    f1 = cross_val_score(lda,X,y,cv=5,scoring="f1_micro").mean()
    recall = cross_val_score(lda,X,y,cv=5,scoring="recall_micro").mean()
    precision = cross_val_score(lda,X,y,cv=5,scoring="precision_micro").mean()
    y_pred = cross_val_predict(lda, X, y, cv=5)

    done = time()
    elapsed = done - start

    result=dict()
    result["f1_score"]=f1
    result["precision"]= precision
    result["rappel"]= recall
    result["temps"]= elapsed
    result["prediction"]= y_pred
    return(result)
```

Listing 2: ADL

De la même façon l'algorithme KNN pour la classification prend lui aussi en paramètres les variables explicatives, la variable cible ainsi que les hyperparamètres disponibles sur la librairie sklearn pour celui-ci et enfin l'hyperparamètre "choisis" pour l'optimisation des hyperparamètre automatiquement.

```
def knnClass(X,y,cat=None,params=None,nb_neighbors=None,nweights=None,nmetric=None,
    nalgorithm=None,choisis = False):
    start = time()

    if choisis == False:
        params = {'n_neighbors': np.arange(1,10),'weights':['uniform', 'distance'] , 'metric
    ': ['euclidean','manhattan',"minkowski"], 'algorithm':["ball_tree", "kd_tree", "brute"]}
        grid = GridSearchCV(KNeighborsClassifier(),params,cv=5,)
        fit = grid.fit(X,y)
        best_param = fit.best_params_
        knn = KNeighborsClassifier()
        knn.set_params(**best_param)

    else:
        knn = KNeighborsClassifier(n_neighbors=nb_neighbors,weights=nweights,metric=nmetric,
    algorithm=nalgorithm)

```

```
15
16      f1 = cross_val_score(knn,X,y,cv=5,scoring="f1_micro").mean()
17      recall = cross_val_score(knn,X,y,cv=5,scoring="recall_micro").mean()
18      precision = cross_val_score(knn,X,y,cv=5,scoring="precision_micro").mean()
19      print(f1, recall, precision)
20      y_pred = cross_val_predict(knn, X, y, cv=5)
21
22      done = time()
23      elapsed = done - start
24
25      result=dict()
26      result["f1_score"]=f1
27      result["precision"]= precision
28      result["rappel"]= recall
29      result["temps"]= elapsed
30      result["prediction"]= y_pred
31      return(result)
```

Listing 3: KNN

Les autres algorithmes sont construits de la même manière et sont disponibles dans le code fourni sur le GitHub.

2.2 L'interface

Pour lancer notre interface, il vous faudra,à partir d'un invite de commande anaconda, installer les libraries suivantes :
-pip install dash
-conda install -c conda-forge dash-html-components
-pip install dash-bootstrap-components
Puis, en ligne de commande ouvrir le fichier "index.py". Enfin, vous pourrez ouvrir le lien indiqué dans votre navigateur.

Voici comment notre interface se présente :
C'est sur la page principale que l'utilisateur va pouvoir insérer son jeu de données au format CSV. Une fois inséré, des informations clés s'affichent à l'écran telles que les variables composant le jeu de données, le type de variable, le nombre de valeurs manquantes ainsi que le nombre de valeurs uniques. L'utilisateur pourra toujours, sur cette page, avoir un aperçu de la composition de son jeu de données ainsi qu'avoir accès aux statistiques descriptives des données.

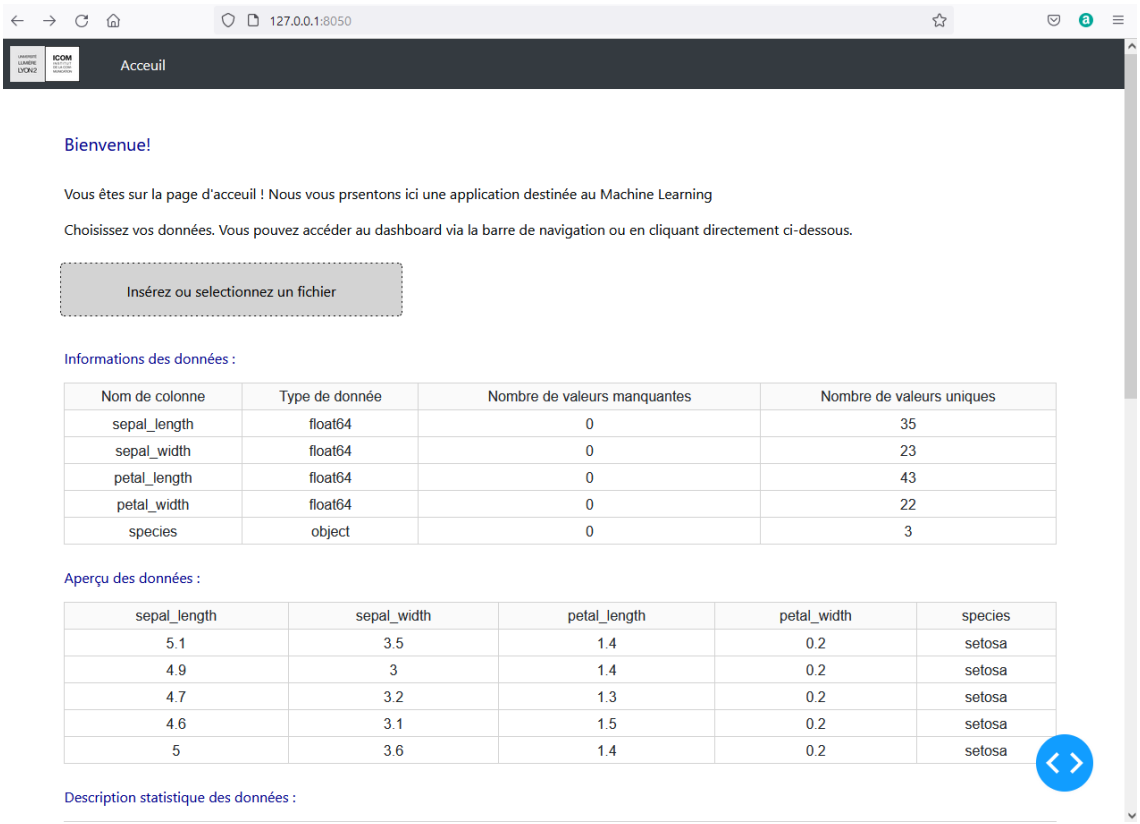

Figure 1: index de l'application

Par la suite un nouveau volet fait son apparition :

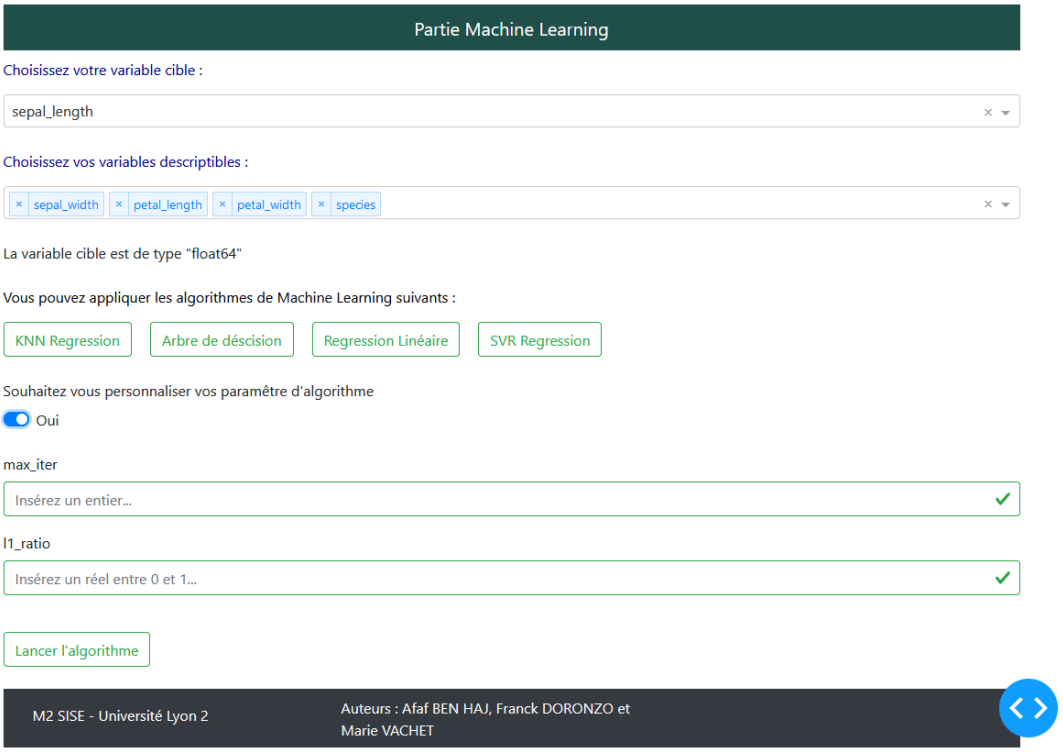

Figure 2: Paramétrage algo

C'est ici que l'utilisateur va pouvoir paramétrer l'algorithme qu'il souhaite utiliser, tout d'abord en sélectionnant la variable cible qu'il souhaite ainsi que les variables explicatives (toutes seront disponibles sauf la variable cible sélectionnée). Après avoir effectué son choix, l'utilisateur verra apparaitre le type de la variable cible, mais aussi les algorithmes compatibles avec le jeu de données. Si le curseur "Souhaitez vous personnaliser vos paramètres d'algorithme" est activé, l'utilisateur se retrouvera donc avec la possibilité de personnaliser les paramètres de l'algorithme choisi par le biais de listes déroulantes ou d'input box. Dans le cas contraire, ce sera donc l'optimisation des paramètres automatique qui se fera (par le biais du GridSearchCV et la variable "choisis" présentée dans les algorithmes précédemment).
Il ne reste plus à l'utilisateur qu'à lancer l'algorithme par le bouton créé à cet effet et la page des résultat s'affichera comme ceci :

On y retrouve les différentes métriques, le temps de calcul, ainsi qu'un graphique permettant d'analyser les prédictions tout comme inscrit dans le cahier des charges.

L'algorithme "SVR" n'est pas dispponible car le temps de calcul n'est pas assez optimisé.

3 Conclusion

Pour conclure, ce projet nous a permis d'approfondir nos compétences en python, notamment sur l'implémentation d'algorithmes pour l'apprentissage supervisé, mais aussi la compréhension de ceux-ci par le biais des différents hyperparèmetres qu'ils peuvent proposer. Ce projet nous aura également permis d'utiliser un nouvel outil de création d'application, Dash qui est un framework très complet pour la visualisation web de données. De plus, c'est un outil largement utilisé, construit sur Flask et Plotly.

Ce dernier nous aura aussi permis de travailler sur le volet "intégration" d'un projet : la coordination des tâches, des ressources, du groupe de travail, tout autre élément du projet, pour en l'occurrence ici pouvoir répondre à la question comment intégrer nos algorithmes avec notre interface? Ce projet nous a permis de développer de réelles compétences en matière de gestion de projet (répartition de tâches, etc.) et donc d'appréhender la réalisation de projet au sein d'une équipe. Compétences qui seront réellement utiles dans le monde professionnel.