# ملحق شرح تشغيل المشروع

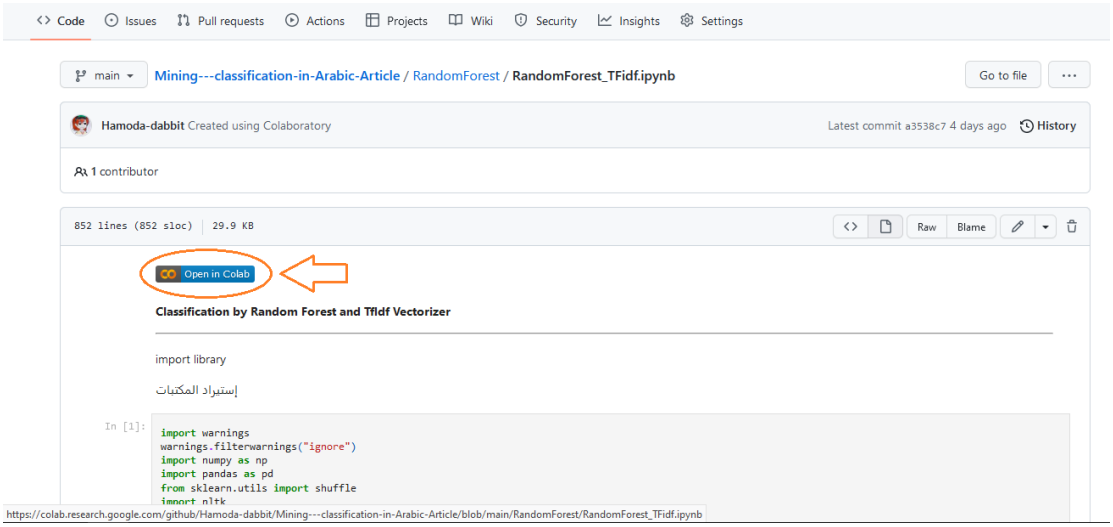
في هذا التقرير نقوم بشرح عملية استخدام الأكواد في تجارب **تصنيف المقالات العربية**

## أولا تفاصيل المشروع

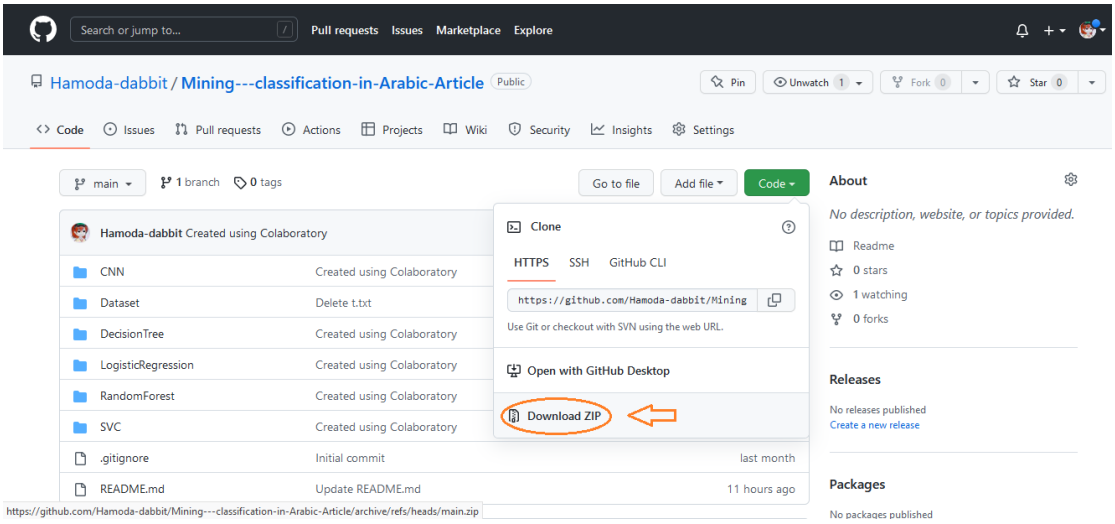
جميع أكواد ونتائج المشروع متاحة في مستودعنا على الـ GitHub على الرابط التالي:

https://github.com/Hamoda-dabbit/Mining---classification-in-Arabic-Article

يمكن تشغيل المشروع على الـ GitHub عبر الدخول إلى التجربة المرادة وتشغيلها عبر الكولاب




أو عبر تحميل المشروع كاملاً عبر الزر التالي




ثم تشغيل التجربة المرادة على بيئة Jupyter Notebook

محتويات المشروع:

كل ملف في المشروع هو تجربة مستقلة ولا يوجد أي ملف يقوم باستعمال توابع أو كلاسات من ملفات أخرى.

قمنا بتقسم المشروع حسب المصنفات، تجارب كل مصنف في مجلد وداخل المجلد يوجد تجارب المصنف مع الميزات المستخدمة، تم تسمية الملفات باختصار المصنف مع اختصار الميزة، توزع ملفات المشروع في المجلدات كالتالي:

- **CNN**
  - CNN_Embedding.ipynb
  - NN_CV.ipynb
  - NN_CV_withoutClean.ipynb
  - NN_Embedding.ipynb
  - NN_TFIDF.ipynb
  - XLNet.ipynb
- **DecisionTree**
  - DecisionTree_CV.ipynb
  - DecisionTree_TFidf.ipynb
- **LogisticRegression**
  - LR_CV.ipynb
  - LR_TFIDF.ipynb
  - LR_TFIDF_whithoutClear.ipynb
- **RandomForest**
  - RandomForest_CV.ipynb
  - RandomForest_TFidf.ipynb
- **SVC**
  - SVC_CV.ipynb
  - SVC_TFIDF.ipynb

بالإضافة لمجلد ملفات البيانات:

- **Dataset**
  - 1.xlsx
  - dataset.md

تم وضع أول ملف من البيانات فقط مع مشروع الـ GitHub وذلك بسبب حجم الملفات الذي يمكن رفعه على الموقع، يمكن تحميل بقية ملفات البيانات من رابط الدرايف الموجود في الملف النصي داخل المجلد.

**ثانيا تجهيز البيانات:**

يمكن تحميل البيانات من الرابط التالي:

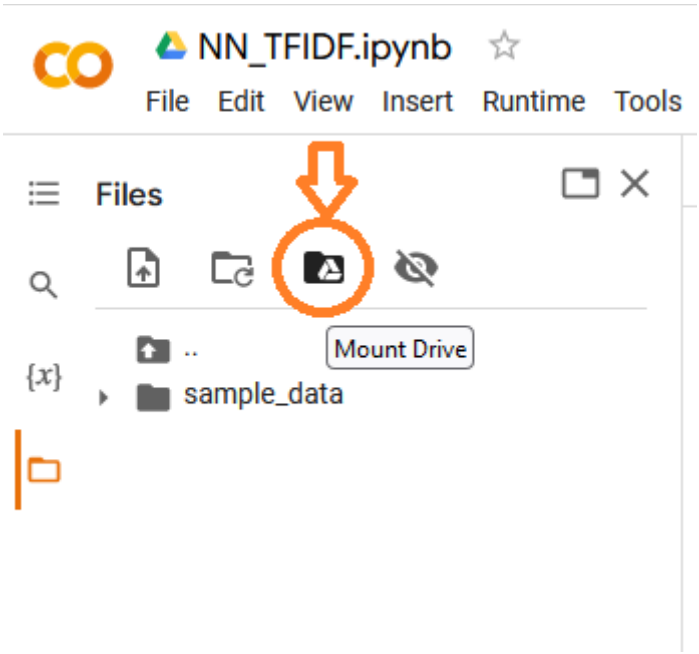
https://drive.google.com/drive/folders/1rV6vZ_lGaL0HUeABV7lj8v0aQp-lBNGY?usp=sharing

الذي يحوي على أربع ملفات بيانات:

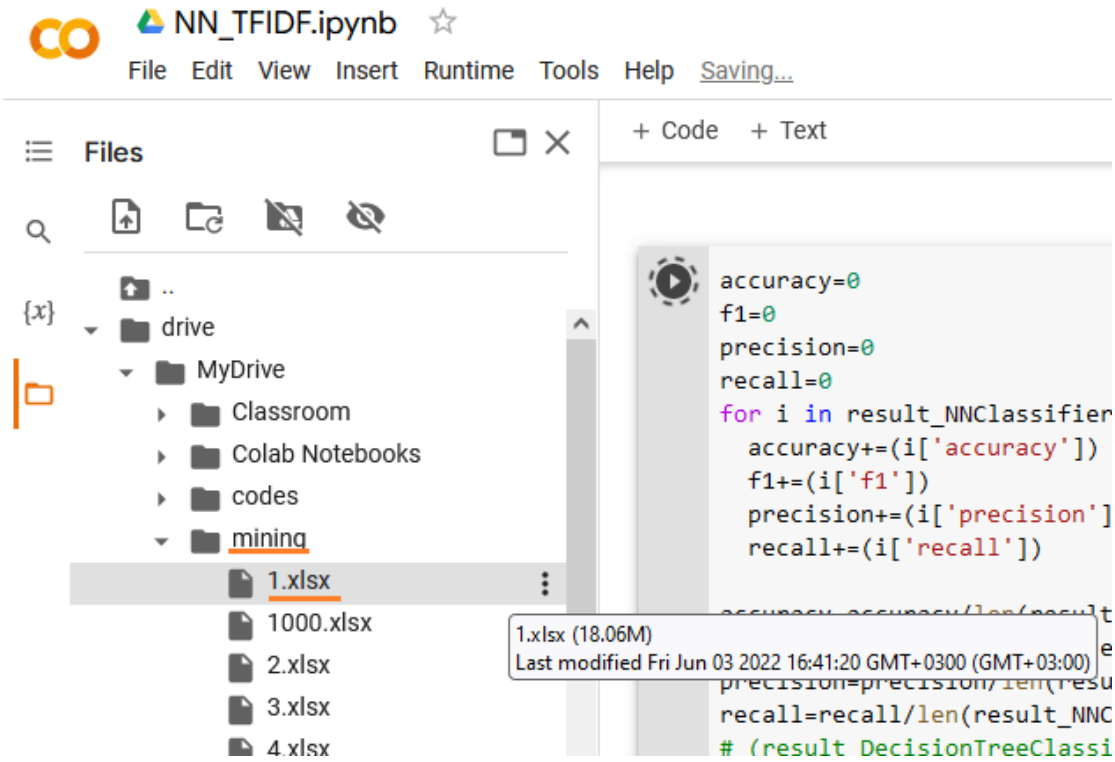
- 1.xlsx
- 2.xlsx
- 3.xlsx
- 4.xlsx

يجب تحميل الملفات الأربع ثم وضعها في مجلد باسم mining.

- ❖ في حال تشغيل الكود على الحاسب الشخصي يجب وضع مجلد mining في مجلد الكود الذي سيتم تشغيله.
- ❖ في حال تشـغيل الكود على الكولاب يجب رفع المجلد mining إلى حسـاب الدرايف ثم ربط حسـاب الدرايف مع مشـروع الكولاب، وذلك عبر الضـغط على زر الدرايف الموضح في الصورة



بعد عملية الربط يجب أن يظهر مجلد mining في المشروع كما هو موضح في الصورة




## ثالثاً تشغيل التجربة

بعد تشغيل أحد التجارب عبر منصة كولاب (الموضح في الخطوة السابقة)

ثم رفع ملفات البيانات في مجلد باسم mining إلى الدرايف وربطه مع المشروع (الموضح سابقاً)

سنجد التجربة مقسمة إلى مراحل التشغيل التالية يتم تشغيل مراحل كود بالترتيب:

- استيراد المكتبات
- استيراد ملفات البيانات **1- Import Data set**

ويحتوي أربع تعليمات لكل قاعدة بيانات، نقوم بتشغيل القاعدة المراد عمل التجربة عليها وتعليق تعليمات بقية القواعد.

**1- Import Data set**

```
df = pd.read_excel("drive/MyDrive/mining/1.xlsx")
# df = pd.read_excel("drive/MyDrive/mining/2.xlsx")
# df = pd.read_excel("drive/MyDrive/mining/3.xlsx")
# df = pd.read_excel("drive/MyDrive/mining/4.xlsx")
```

- بعثرة البيانات shuffle
- تنظيف البيانات **2- Clean Data**
- استخراج الميزات (تحويل النص إلى أرقام) **3- Feature Extraction:**
- عمل تابع لحساب الدقة accuracy calculation function
- عملية التصنيف **4- classification**

نقوم بإضـــافة تعليمة %%time التي تقوم بحســـاب الوقت المســـتغرق أثناء تنفيذ هذه المرحلة من الكود (التعليمات داخل صـندوق الكود الواحد) حيث تقوم بطباعة الوقت بعد مربع الكود عند الانتهاء من التنفيذ.

بعد هذه المرحلة ينتج لدينا أربع نتائج اختبار (بســـبب عملية corss validation) نقوم بتخزين النتائج في مصفوفة

```
%%time
from sklearn.model_selection import StratifiedKFold
from sklearn import tree

strtfdKFold = StratifiedKFold(n_splits=4)
kfold = strtfdKFold.split(X1, y1)
clf = tree.DecisionTreeClassifier(random_state=5)
result_DecisionTreeClassifier=[]
#
for k, (train_index, test_index) in enumerate(kfold):
    X_train, X_test = X1[train_index], X1[test_index]
    y_train, y_test = y1.iloc[train_index], y1.iloc[test_index]
    model = clf.fit(X_train, y_train)
    y_pred=(model.predict(X_test))
    result_DecisionTreeClassifier.append(calculate_results(y_test, y_pred))
```

```
CPU times: user 7min 51s, sys: 2.67 s, total: 7min 54s
Wall time: 7min 57s
```

- حساب متوسط النتائج الأربعة

```
accuracy=0
f1=0
precision=0
recall=0
for i in result_DecisionTreeClassifier:
  accuracy+=(i['accuracy'])
  f1+=(i['f1'])
  precision+=(i['precision'])
  recall+=(i['recall'])

accuracy=accuracy/len(result_DecisionTreeClassifier)
f1=f1/len(result_DecisionTreeClassifier)
precision=precision/len(result_DecisionTreeClassifier)
recall=recall/len(result_DecisionTreeClassifier)
# (result_DecisionTreeClassifier)
results_DecisionTreeClassifier=[{'accuracy':accuracy,'f1':f1,'precision':precision,'recall':recall}]
results_DecisionTreeClassifier
```

```
[{'accuracy': 84.60592943983504,
  'f1': 0.8463474809846698,
  'precision': 0.8467355770869606,
  'recall': 0.8460592943983505}]
```

يتم حساب متوسط النتائج الأربعة وتخزينها في متحول وطباعتها.

عند هذه المرحلة تنتهي التجربة على قاعدة البيانات الواحدة.

- النتائج **5- Show Results**

لتجميع نتائج القواعد الأربعة في نفس الملف نقوم بتشغيل كود الطباعة التالي عند استخدام قاعدة البيانات الأولى

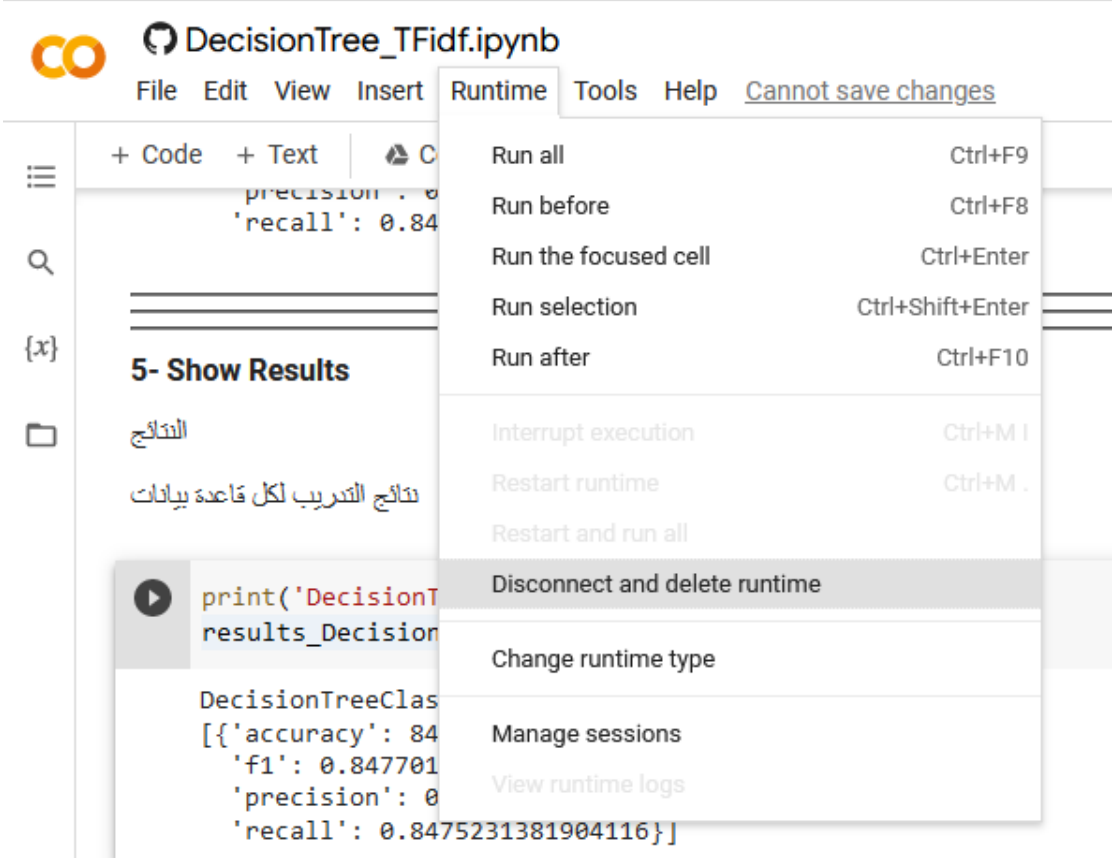
```
print('DecisionTreeClassifier results in dataset 1:')
results_DecisionTreeClassifier
```

```
DecisionTreeClassifier results in dataset 1:
[{'accuracy': 84.75231381904115,
  'f1': 0.8477014138491688,
  'precision': 0.8479472021812396,
  'recall': 0.8475231381904116}]
```

فيتم حفظ نتيجة التجربة في ناتج عملية الطباعة.

الخطوة التالية أن نقوم بتصـــفير الذواكر وإعادة تنفيذ التجربة ولكن على قاعدة البيانات الثانية ثم الثالثة ثم الرابعة.

نقوم بتصفير الذاكرة من قائمة Runtime خيار Disconnect and delete runtime




نعود لبداية المشروع

نقوم بربطه مع الدرايف عبر زر الربط

نقوم بتنفيذ مربعات الأكواد تسلسلياً حتى الوصول لمرحلة النتائج

لا نقوم بتنفيذ مربع طباعة نتيجة القاعدة الأولى حتى لانفقدها، نتجاوها للمربع التالي ونقوم بتنفيذ كود طباعة ناتج القاعدة الثانية

النتائج

نتائج التدريب لكل قاعدة بيانات

```
[ ] print('DecisionTreeClassifier results in dataset 1:')
    results_DecisionTreeClassifier
```

```
DecisionTreeClassifier results in dataset 1:
[{'accuracy': 84.75231381904115,
  'f1': 0.8477014138491688,
  'precision': 0.8479472021812396,
  'recall': 0.8475231381904116}]
```

```
print('DecisionTreeClassifier results in dataset 2:')
results_DecisionTreeClassifier
```

```
DecisionTreeClassifier results in dataset 2:
[{'accuracy': 86.4940590922711,
  'f1': 0.8650148783873235,
  'precision': 0.8652240244394017,
  'recall': 0.8649405909227109}]
```

نكرر الخطوات للقاعدة الثالثة والرابعة

أخيراً ينتج لدينا نتائج أربع قواعد بيانات، نستطيع حساب المتوسط لها بطريقة يدوية، أو عن طريق نسخ النتائج ولصقها (يدوياً) في مربع كود وتخزينها في مصفوفة وحساب المتوسط برمجيا وهو الذي قمنا بعمل في مربع الكود الأخير.

قمنا أيضاً بطباعة وقت تنفيذ التدريب الذي قمنا بنسخه يدوياً من نتيجة مرحلة التدريب، أما الذاكرة RAM فقمنا بكتابتها عن طريق فحص الذاكرة المستخدمة بعد الانتهاء من تنفيذ التجربة لقاعدة البيانات الرابعة.

```
Final_Result=[{'accuracy': 84.75231381904115,
  'f1': 0.8477014138491688,
  'precision': 0.8479472021812396,
  'recall': 0.8475231381904116},
  {'accuracy': 86.4940590922711,
  'f1': 0.8650148783873235,
  'precision': 0.8652240244394017,
  'recall': 0.8649405909227109},
  {'accuracy': 84.94045040256954,
  'f1': 0.8495581834605119,
  'precision': 0.8497768376618365,
  'recall': 0.8494045040256953},
  {'accuracy': 84.7669488926263,
  'f1': 0.8478120876570386,
  'precision': 0.8480099659559468,
  'recall': 0.8476694889262631}
  ]

accuracy=0
f1=0
precision=0
recall=0
for i in Final_Result:
  accuracy+=(i['accuracy'])
  f1+=(i['f1'])
  precision+=(i['precision'])
  recall+=(i['recall'])

accuracy=accuracy/len(Final_Result)
f1=f1/len(Final_Result)
precision=precision/len(Final_Result)
recall=recall/len(Final_Result)
# (result_DecisionTreeClassifier)
Final_Result=[{'accuracy':accuracy,'f1':f1,'precision':precision,'recall':recall}]
print("RAM: 2.2G  ,  CPU times: user 7min 51s")
print('Final_Result:')
(Final_Result)
```

```
RAM: 2.2G  ,  CPU times: user 7min 51s
Final_Result:
[{'accuracy': 85.23844305162703,
  'f1': 0.8525216408385107,
  'precision': 0.8527395075596063,
  'recall': 0.8523844305162702}]
```

يمكن معرفة حجم استخدام الرام عبر وضع المؤشر على مقاييس الأداء في القسم الأيمن العلوي