جامعة حلب في المناطق المحررة
الدراسات العليا
كلية الهندسة المعلوماتية

تمهيدي ماجستير

تنقيب البيانات المتقدمة

Classification of Arabic articles

إعداد:

م. أماني عبه جي

م. محمّد ضبّيط

م. مهنّد حمّود

إشراف:

د. سائد القرالة

العام الدّراسي: 2022/2021

الملخص

يستخدم تصنيف النصوص على مجال واسع في مواقع البحث وفي مواقع التواصل الاجتماعي.

في هذ الدراسة نقوم باستخدام مجموعة مقالات بعدد 68312 مقالة، تم جمعهم من عدة صحف عربية وهذه المقالات مقسمة على خمس أقسام (رياضة، سياسة، ثقافة، اقتصاد، جنائية).

حيث نقوم بعملية التصنيف عبر خمس خوارزميات مختلفة (شجرة القرار Decision Tree، الغابة العشوائية Random Forest، الانحدار اللوجستي Logistic Regression، مصنف المتجهات الداعمة SVC (Support Vector Classifier)، شبكة عصبونية تلافيفية (ترشيحية) CNN).

وباستخدام أربع طرق لاستخراج الميزات (TFIDF، Count Vectorizer، WordEmbedding، XLNet).

وجدنا أن خوارزمية SVC مع ميزة TFIDF هي الأفضل بدقة 96% واستغرقت للتنفيذ حوالي 4 ساعات ولكن الشبكات العصبونية والانحدار اللوجستي مع ميزة Count Vector تتفوق بوقت التنفيذ مع حوالي 5 دقائق ودقة قريبة 95%.

وجدنا أيضاً أن استخدام تنظيف البيانات على النصوص العربية ليس له أي فوارق ملحوظة على دقة النتائج.

أولاً: المقدمة

مع النمو السريع للإنترنت، زادت وتوافرت النصوص عبر الإنترنت بشكل كبير، وهي تعد كنز من الكنوز التي تتصارع عليها الدول العظمى من أجل تحليل وفهم الشعوب. نتيجة لذلك، أصبح التنقيب عن النصوص هو الأسلوب الأساسي لمعالجة البيانات النصية وتنظيمها واستخراج المعلومات ذات الصلة من كمية هائلة من البيانات النصية. يمكن تعريف التنقيب عن النصوص بأنه عملية تحليل النص لاستخراج المعلومات منه لأغراض معينة.

تتضمن مهام التنقيب عن النصوص النموذجية تصنيف النصوص وتجميعها واستخراج الكيانات وتحليل المشاعر وتلخيص النصوص.

يستخدم تصنيف النصوص في العديد من التطبيقات مثل تصفية النصوص وتنظيم المستندات وتصنيف القصص الإخبارية والبحث عن معلومات مثيرة للاهتمام على الويب وتصفية البريد الإلكتروني العشوائي وما إلى ذلك.

توفِّر التصنيفات طريقة مفيدة لتجميع المقالات ذات الصلة معاً.

في حين أن التصنيف قد استخدم كطريقة لتنظيم المقالات والفقرات، فقد أصبح التصنيف وغيره من الطرق التقليدية للتنظيم، ولاستطلاع الموضوعات بشكل منهجي من أجل تطوير أفكار للمقال.

يمكن استكشاف العديد من الموضوعات من خلال التصنيف، أي تحديد الأنواع المختلفة والأصناف والطرق وتوضيحها .

لاتزال الأبحاث المتعلقة بتصنيف النصوص العربية قليلة مقارنة ببقية اللغات العالمية.

في هذا البحث تم اقتراح نظام تصنيف آلي للمقالات العربية. يتم تنفيذ النظام المقترح باستخدام نهج التعلم الخاضع للإشراف والذي يعرف بأنه إيجاد أصناف الفئات المحددة مسبقاً بناءً على الاحتمالية التي يتم اقتراحها بعد التدرب على مجموعة التدريب من المستندات المصنفة.

بالنسبة لمجموعة بيانات التدريب، تم استخدام مجموعة بيانات arabic_dataset_classifiction.csv.

وباستخدام أربع طرق لاستخراج الميزات (TFIDF، Count Vectorizer، WordEmbedding، XLNet).

التصنيف عبر خمس خوارزميات مختلفة (أشجار القرار، الغابات الكثيفة، الانحدار اللوجستي، SVC، الشبكات العصبونية).

الميزات والعيوب

يتميز هذا المشروع بكونه يدرس قاعدة بيانات باللغة العربية حيث أن الدراسات التي تعنى باللغة العربية قليلة مقارنة مع اللغات العالمية الأخرى، على الرغم من أنها اللغة الخامسة عالمياً من حيث عدد المتكلمين الأصليين.

قد يواجه هذا المشروع بعض المشكلات بسبب صعوبة معالجة اللغة العربية لاحتوائها على التشكيل الذي يمكن أن يغير معنى الكلام بدون أن تتغير الأحرف، والبنية النحوية التي تسبب تغير معنى الجمل رغم تشابه الكلمات، فهو مشروع أولي يحتاج للتطوير المستمر.

ثانياً: الأعمال المشابهة

في الآونة الأخيرة نشطت الدراسات التي تهتم بتصنيف النصوص وتخصيصها ضمن مسمى معين، كالنصوص السياسية أو الرياضية.

يوجد بعض الدراسات التي اعتمدت على اللغة العربية للدراسة منها:

1- **دراسة سمير ومحمد وفاتحة ولبنة وعبد المجيد (2018)** [1] عن تصنيف النص العربي باستخدام تقنيات التعلم العميق، التي تم فيها استخدام نفس قاعدة البيانات المستخدمة في هذا البحث، وقام الباحثون فيها بالتصنيف عبر ثلاث خوارزميات (LR، SVM، CNNs) مع ميزة TFIDF ووصلوا لأفضل نتيجة تصنيف عبر **الشبكات العصبونية العميقة بدقة 92.94**.
2- **دراسة أحمد وعلي (2021)** [2] عن كشف الأخبار المزيفة في التغريدات العربية خلال جائحة كورونا، حيث تم استخدام مجموعة بيانات بلغت 7 ملايين تغريدة تم تجمعها عن طريق مكتبة Tweepy Python باستخدام Twitter's API، واستخدام التعليم الآلي تحت الإشراف لإجراء تصنيف الأخبار المزيفة عبر 6 خوارزميات: (NB وLR وSVM وMLP وRF وXGB) وعبر 4 أنواع من الميزات:

(Count vector, word-level TF-IDF, n-gram-level TF-IDF, character-level TF-IDF)

وتم تحقيق أفضل النتائج باستخدام الانحدار اللوجستي LR، حيث حقق تصنيف مجموعة البيانات المشروحة يدوياً درجة F1 وأفضل نتيجة تمت باستخدام **LR** كمصنف و**count vector** كميزة بدقة **93.3%** ويليها LR كمصنف وTF-IDF كميزة بدقة 87,8%.

3- **وفي دراسة لمى وبول (2020)** [3] قام الباحثين بدمج ودراسة تحليل التغريدات العربية بهدف دعم منظمات الصحة العامة القادرة على ذلك والتعلم من بيانات وسائل التواصل الاجتماعي عن طريق استخدام التالي:

- تحليل الموضوعات التي تمت مناقشتها بين الناس خلال ذروة انتشار فيروس كورونا.
- تحديد وكشف الشائعات ذات الصلة ب COVID-19.
- التنبؤ بنوع مصادر التغريدات COVID-19.

كما قاموا بجمع البيانات عن طريق جمع التغريدات باللغة العربية حول رقم الأمراض المعدية من سبتمبر 2019 وقاموا بتحليل التغريدات المتعلقة فقط ب COVID-19 من ديسمبر 2019 إلى أبريل 2020 وكانت حوالي ما يقرب من ستة ملايين تغريدة باللغة العربية خلال هذه الفترة.

بعد ذلك، قاموا بمعالجة التغريدات من خلال بعض الإجراءات على التوالي:

(Cluster Analysis→ Rumour Detection → Source Type Prediction)

تم استخدام خوارزميات (LR, SVC, NB) حيث حُققت أفضل نتيجة باستخدام **LR** كمصنف **و count vector كميزة بدقة 84.03%** وأيضاً SVC كمصنف وTF-IDF كميزة بدقة مماثلة 84.03%.

4- **وفي دراسة نادر عصام وآخرون (2021)** [4] قدم لنا الباحثين طريقة محسنة لتحديد لهجة اللغة العربية عن طريق عدة تقنيات (LR, LSTM and BiLSTM) وBERT Model متمثلة بـ (ArabicBERT, AraBERT, MARBERT).

وحققوا أفضل أداء باستخدام **أسلوب MARBERT الذي قدم دقة 90.45٪** على مجموعة بيانات التعليقات العربية عبر الإنترنت (AOC).

وأيضاً طبقوا نموذج التعرف على اللهجة المستند إلى **BERT** لتحديد مصدر منطقة التغريدات المتعلقة بـ COVID-19، وتمكنوا من التعرف على منطقة المصدر لـ 1.76 مليون تغريدة عربية بعد أخذ عينات عشوائية من 2500 تغريدة من كل منطقة مجموعة والتحقق منها يدوياً، وحصلوا على متوسط دقة بنسبة **97.36٪.**

5- **وفي دراسة نورا وحمزة (2021)** [5] قام الباحثين بتسليط الضوء على نقص البحث في تحليل المشاعر للغة العربية والتحقيق إلى أي مدى من الدقة يمكن أن تساعد نماذج التعلم العميق في فهم سلوك المجتمع أثناء COVID-19.

فاقترحوا تصنيفاً للعاطفة متعدد العلامات من خلال استخدام محولين أساسهما BERT، وهما AraBERT وMARBERT، مع استبدال الرموز التعبيرية.

كما اقترحوا أيضاً تقنية DWLF لإعطاء وزن أكبر في وظيفة الخسارة لعينات الأقلية للطبقات، بالإضافة إلى ذلك إنشاء مجموعة بيانات جديدة SenAIT، من خلال دمج المشترك للعواطف باستخدام العديد من خطوات المعالجة المسبقة مع زخرفة الكلمات المختلفة.

حققت النماذج المقترحة نتائج متطورة على مجموعة بيانات معيارية وتم تحقيق أفضل أداء من خلال ضبط طراز **MARBERT** بدقة مع استبدال الرموز التعبيرية وDWLF.

وأظهرت النتائج التي تم الحصول عليها أهمية وصف الرموز التعبيرية للمشاعر وكيف يمكن لـ DWLF تعزيز أداء هذه الأنظمة.

ثالثاً: آلية عمل البرنامج

مخطط النظام

الشكل (1) – مخطط النظام (مراحل عمل النظام)

1- تجهيز البيانات

مجموعة البيانات "arabic_dataset_classifiction.csv" [6]

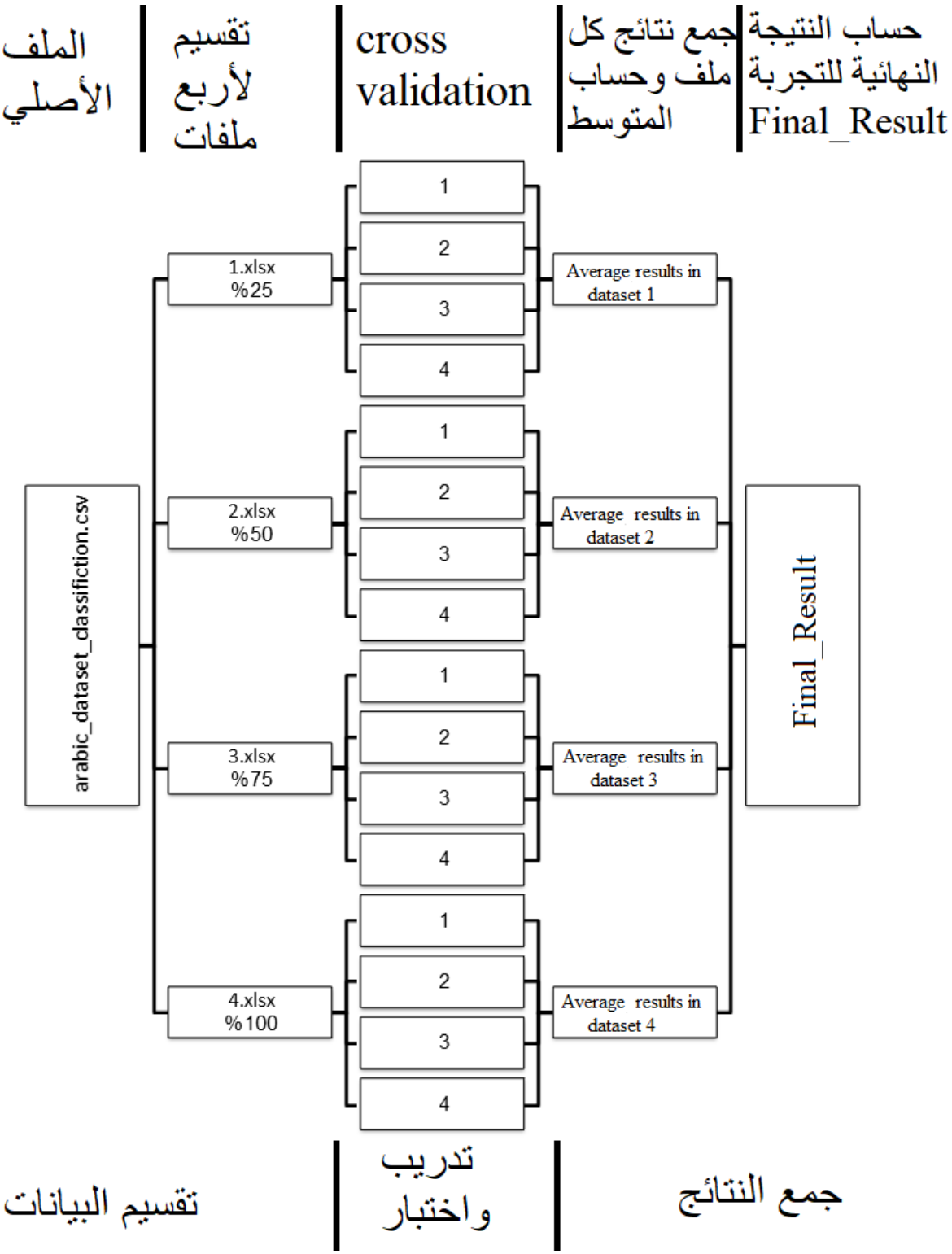
قمنا بتحويل ملف البيانات إلى صيغة xlsx وذلك بسبب بعض المشاكل التي واجهتنا في توافق تشفير النصوص (UTF8) مع اللغة العربية، ولسهولة تقسيم البيانات.

تم جعل عدد المقالات لكل الأصناف متساوية حيث كانت البيانات الأصلية لديها عدد كبير من المقالات لأحد الأصناف بما يساوي 40% من قاعدة البيانات وهذا الأمر يؤثر على عملية اختبار النتائج، فقمنا بحذف المقالات الزائدة لجعل جميع الأصناف بعدد أقل صنف من أجل مساواة الأصناف في قاعدة البيانات.

مجموعة البيانات تحتوي مقالات من مختلف المصادر حيث أن عدد البيانات 68312 عينة، تم جمعهم من 3 صحف عربية (الصباح، هس برس، أخبارنا)، ومصنفة إلى 5 أصناف (رياضة، سياسة، ثقافة، اقتصاد، جنائية).

مجموعة البيانات تحتوي على عمودين:

- عمود text: يحوي النص.
- عمود targe: يحوي رقم الصنف:
0 = ثقافة (فن)، 1 = جنائية (أخبار جنائية)، 2 = اقتصاد، 3 = سياسة، 4 = رياضة.



الشكل (2) – مراحل تقسيم البيانات حتى تجميع النتائج

التأكد من صحة النتائج

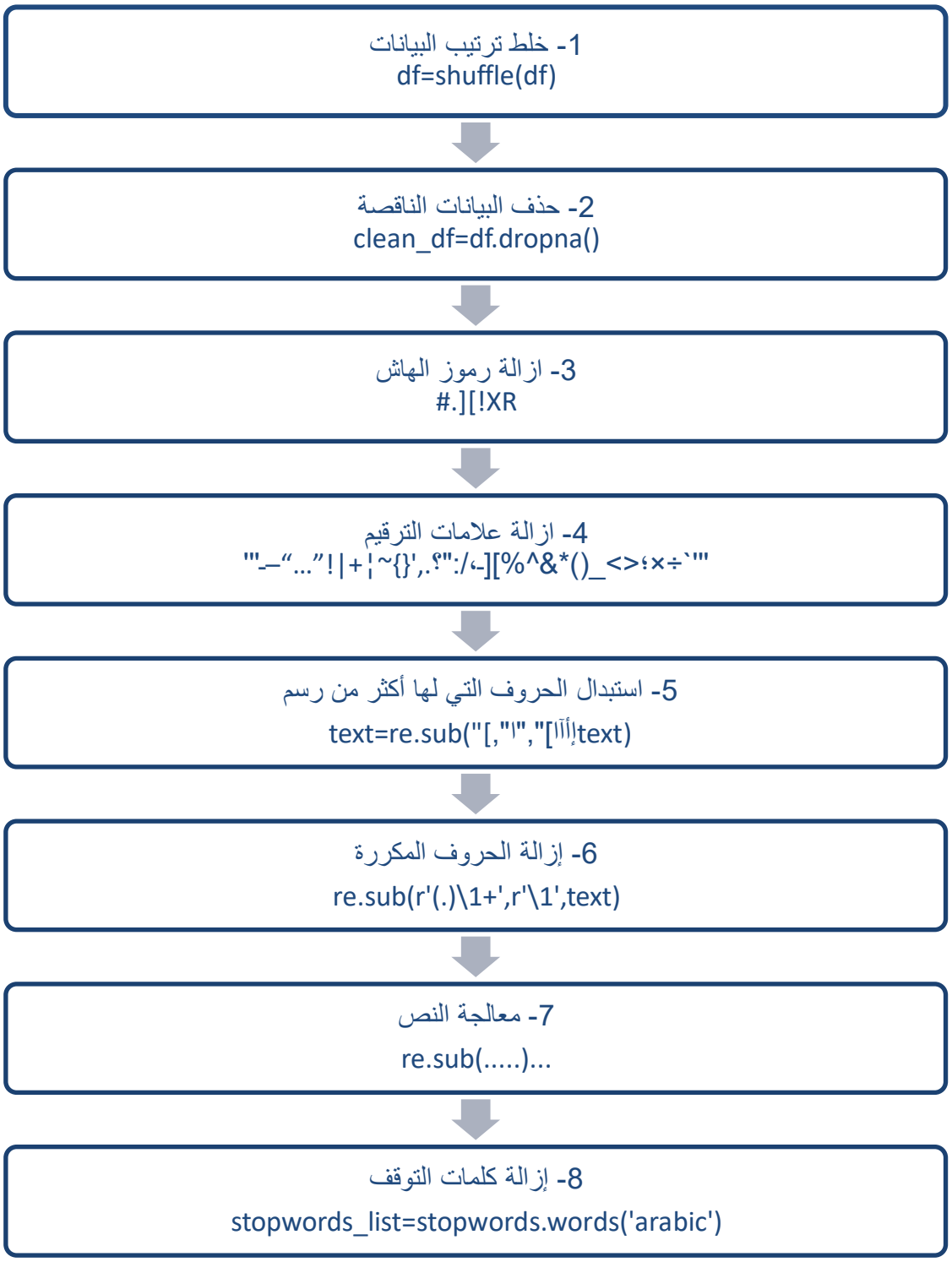
من أجل ضبط النتائج والتأكد منها اتبعنا الخطوات الآتية:

1. قمنا بتقسيم قاعدة البيانات الكلية إلى أربع ملفات:
   - dataset1: الأول يحوي 25٪ من البيانات بعدد 3500 لكل صنف ومجموع 17500.
   - dataset2: الثاني يحوي 50٪ من البيانات بعدد 7000 لكل صنف ومجموع 35000.
   - dataset3: الثالث يحوي 75٪ من البيانات بعدد 10500 لكل صنف ومجموع 52500.
   - dataset4: الرابع يحوي 100٪ من البيانات بعدد 13738 لكل صنف ومجموع 68690.
2. تم وضع أعداد عينات متساوية لكل الأصناف الخمسة في الملفات السابقة.
3. ثم قمنا ببعثرة ترتيب البيانات عبر تابع الـ shuffle.
4. قمنا باستخدام خوارزمية التحقق المتقاطع cross validation لتكرار التدريب مع تغيير بيانات الاختبار بغرض التأكد من صحة النتائج التي حصلنا عليها وأنه لم تتم عملية فرط التخصيص overfitting.

وبعدها قمنا بتدريب الموديلات المختلفة التي استخدمناها للتعليم على ملفات مجموعات البيانات مع استخدام لخوارزمية الـ cross validation وبعدد 4 مرات.

2- تنظيف البيانات

1) خلط ترتيب البيانات (تكون البيانات مرتبة حسب الأصناف في البداية).
2) حذف البيانات الناقصة (حذف سطر البيانات في حال عدم وجودة نص أو صنف) df.isnull()
3) إزالة رموز الهاشتاغ #.][!XR
4) إزالة علامات الترقيم ×÷'''`؛][%^&*()_<>-،:/"?.,{}~|+|!”…“–'''
5) استبدال الحروف التي تحوي أكثر من رسم ك گ ہ ة ي ى إأآا
6) إزالة المحارف المكررة.
7) معالجة المستند: وتحوي على عدة توابع مضمنة:
   - استبدال الكلمات المسبوقة برمز @ بالفراغ.
   - استبدال رمز _ بالفراغ.
   - استبدال رمز السطر \n بالفراغ.
   - حذف الأحرف الإنكليزية [a-z,A-Z].
   - استبدال رموز المواقع والروابط ((www\.[^\s]+)|(https?://[^\s]+)) بالفراغ.
8) حذف كلمات التوقف (الكلمات الشائعة) العربية.



الشكل (3) – مراحل معالجة البيانات (تنظيف البيانات)

3- استخراج الميزات

عملية تحويل النصوص إلى أرقام وتتم عبر العديد من الخوارزميات

سنقوم باستعمال الطرق التالية:

- **TFIDF:** تم ضبط البارامترات التالية:
  - max_features=1500
  - stop_words=stopwords.words('arabic')
- **Count Vectorizer:** تم ضبطه بالبارامترات التالية:
  - max_features=1500
- **word Embedding:**
  - VOCAB_SIZE بعدد الكلمات الموجودة :
  - EMBED_SIZE=300
  - input_length1000
- **XLNet**
  - 'num_train_epochs':1
  - 'train_batch_size':32
  - 'max_seq_length':128

4- التصنيف:

هو الخوارزمية التي تقوم بتصنيف المقالة إلى أحد الأصناف المحددة وذلك عبر تقديم بيانات مع أصنافها لتتدرب الخوارزمية عليها، ثم الاختبار على بيانات لم يتم التدرب عليها، وحساب دقة النظام.

تم استخدام الخوارزميات التالية:

- **شجرة القرار DecisionTree:** شجرة القرار هي تقنية تعلم خاضعة للإشراف يمكن استخدامها لكل من مشاكل التصنيف والانحدار، ولكنها في الغالب مفضلة لحل مشاكل التصنيف. تنظم على شكل شجرة، ويتم تقسيم البيانات باستمرار وفقاً لمعامل معين. حيث تمثل العقد الداخلية ميزات مجموعة البيانات، وتمثل الفروع قواعد القرار وتمثل كل عقدة ورقية النتيجة، من السهل فهمها لأنها تتبع نفس العملية التي يتبعها الإنسان أثناء اتخاذ أي قرار في الحياة الواقعية، متطلباتها لتنظيف البيانات أقل مقارنة بالخوارزميات الأخرى [7].
- **الشبكة العصبونية التلافيفية CNN:**
  - **CNN:** قمنا باستخدام شبكة عصبونية عميقة موديل Sequential بالطبقات التالية:

- Embedding
- Conv1D(filters=128, kernel_size=4, padding='same', activation='relu'))
- MaxPooling1D(pool_size=2))
- Conv1D(filters=64, kernel_size=4, padding='same', activation='relu'))
- MaxPooling1D(pool_size=2))
- Conv1D(filters=32, kernel_size=4, padding='same', activation='relu'))

- MaxPooling1D(pool_size=2))
- Flatten())
- Dense(256, activation='relu'))
- Dense(5, activation='sigmoid'))

  - **NN:** قمنا ببناء شبكة عصبية بسيطة بالموديل Sequential بالطبقات التالية:
- model.add(layers.Dense(10, input_dim=input_dim, activation='relu'))
- model.add(layers.Dense(5, activation='sigmoid'))

❖ **الغابة العشوائية Random Forest:** خوارزمية التعلم الآلي الشائعة تنتمي إلى تقنية التعلم بالإشراف، يمكن استخدامها لكل من مسائل التصنيف والانحدار. تعتمد على مفهوم التعلم الجماعي، وهي عملية الجمع بين عدة مصنفات لحل مشكلة معقدة وتحسين أداء النموذج، حيث تحتوي على عدد من أشجار القرار على مجموعات فرعية مختلفة من مجموعة البيانات المحددة ويأخذ المتوسط لتحسين الدقة التنبؤية لمجموعة البيانات هذه، وبدلاً من الاعتماد على شجرة قرار واحدة تأخذ الغابة العشوائية التنبؤ من كل شجرة وتعتمد على أغلبية الأصوات للتنبؤات، وتتنبأ الناتج النهائي. يؤدي العدد الأكبر من الأشجار في الغابة إلى دقة أعلى ويتجنب ظاهرة overfiting [8].

  - تم ضبطه بالبارماترات التالية:

- **n_estimators=1000,**
- **random_state=0**

❖ **الانحدار الخطي Logistic regression:** هي إحدى خوارزميات التصنيف تستخدم لتصنيف البيانات إلى فئات منفصلة مثل yes/no، 1/0...، تقوم هذه الخوارزمية بتحويل المخرج output باستخدام دالة Sigmoid إلى قيم احتمالية والتي ستصنف إلى إحدى الفئات ضمن هذه الخوارزمية [9].

❖ **مصنف المتجهات الداعمة SVC:** يتمثل المبدأ الأساسي للخوارزمية في العثور على مستوي مفرط يقوم بتقسيم مجموعة البيانات إلى صنفين بأفضل طريقة، تسمى جميع النقاط الموجودة على المستوي الفائق على كلا الجانبين متجهات الدعم، يمكن استخدام خوارزمية SVC لمهام التصنيف أو الانحدار ولكن في الأغلب تستعمل في التصنيف [10].

رابعاً: التجارب والنتائج

تعتمد مقارنة النتائج عادة على دقة النظام عند الاختبار على البيانات الجديدة، ولكن الاقتصار على الدقة فقط ليس عملياً، في نتائج الخوارزميات التي تم استخدامها سنقوم بمقارنة:

- **وقت التنفيذ:** سيتم اعتماد الوقت المستهلك لتدريب الموديل فقط.
- **الذاكرة المحجوزة:** وذلك بشكل تقريبي لعدم توفر آلية معروفة يمكن قياس الذاكرة المستهلك للبرنامج عبرها، سيتم اعتماد قياس حجم الرام المستهلك قبل بدء البرنامج وعند الانتهاء.
- **دقة النظام:** سيتم التقييم بناء على المعايير الأربعة (Accuracy,precision,recall,f1).

ملاحظة: تم عمل التجارب على منصة الـ Google Colab تم التنفيذ على المعالج الأساسي None وليس عن طريق كرت الشاشة GPU وتم قياس الوقت بناء على هذا الأمر، تم قياس وقت تدريب الموديل فقط وذلك على قاعدة البيانات الرابعة والتي تحوي جميع البيانات.

ذاكرة الكولاب 12G قبل بدأ البرنامج يكون النظام قد حجز حوالي 1G، لذلك سيتم قياس الذاكرة المحجوزة بعد الـ 1G الخاصة بالنظام.

نتائج التجارب

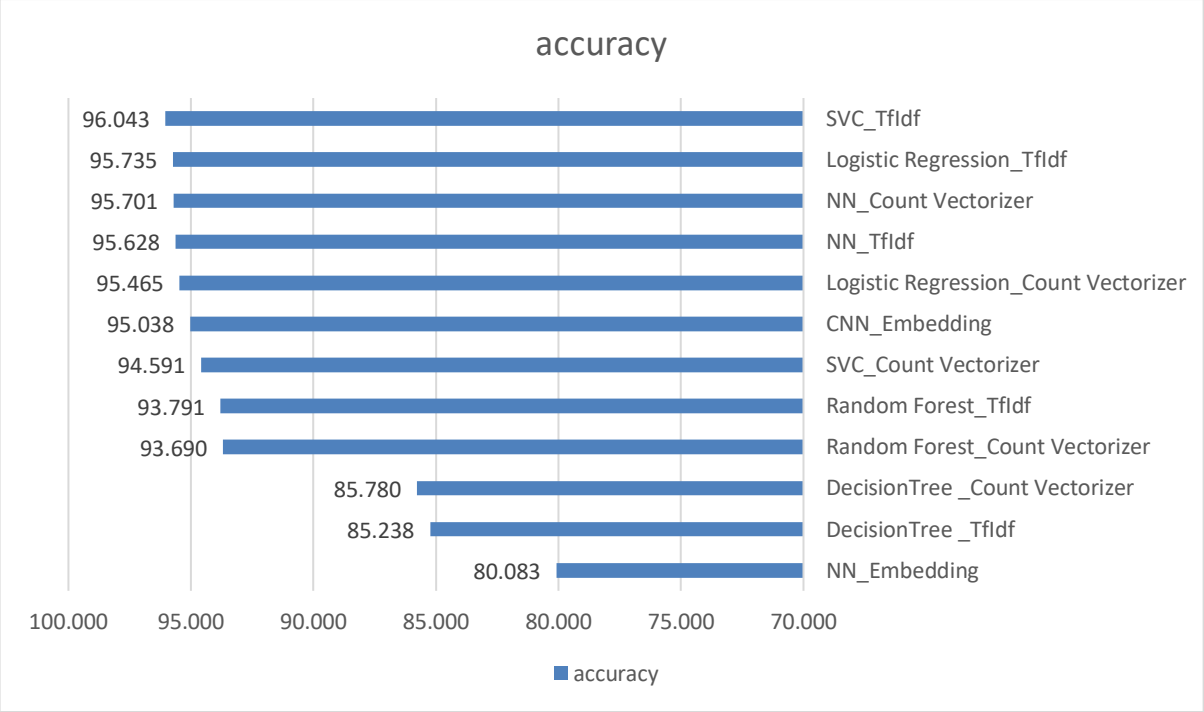
num	Classification	Feature Extraction	accuracy	Time	RAM ABOVE 1GB
1	SVC	Tfldf	96.043	3h 57min 16s	2.13
2	Logistic Regression	Tfldf	95.735	5min 36s	0.64
3	NN	Count Vectorizer	95.701	4min 50s	0.86GB
4	NN	Tfldf	95.628	4min 13s	1.05GB
5	Logistic Regression	Count Vectorizer	95.465	7min 36s	0.79
6	CNN	Embedding	95.038	5h 56min 21s	1.6
7	SVC	Count Vectorizer	94.591	2h 29min 4s	2.3
8	Random Forest	Tfldf	93.791	1h 8min 7s	3.12
9	Random Forest	Count Vectorizer	93.690	2h 55min 4s	2.1
10	Decision Tree	Count Vectorizer	85.780	7min 8s	1.6 GB
11	Decision Tree	Tfldf	85.238	7min 51s	2.2
12	NN	Embedding	80.083	>6h	1.7
13	XLNet	XLNet	20.225	>6h	

الجدول (1) – الدقة الناتجة والزمن المستغرق والذاكرة المستخدمة لكل تجربة

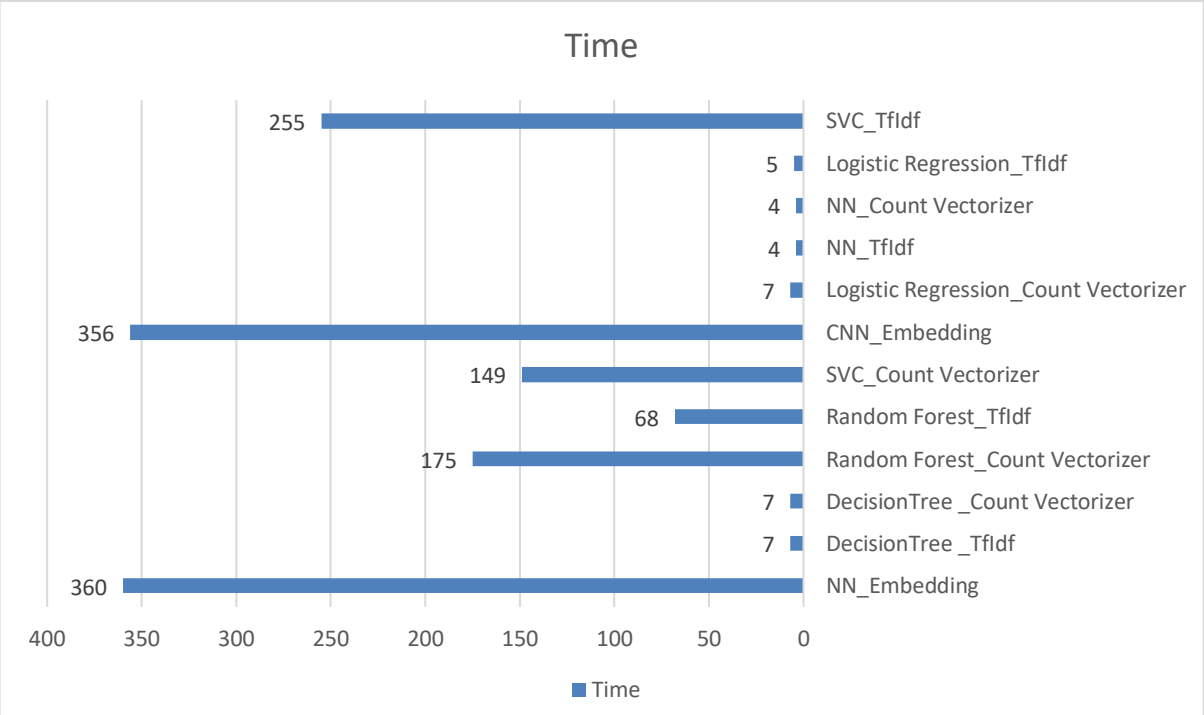
نلاحظ بعد ترتيب التجارب حسب الدقة أن:

خوارزمية SVC مع ميزة TFIDF حصلت على أعلى دقة ولكنها استغرقت وقت 3 ساعات و57 دقيقة، بينما NN مع ميزة Count Vector وLogistic Regression مع ميزة Count Vector حصلوا على دقة قريبة أيضاً **95%** ولكن الوقت بحدود 5 دقائق.

لذلك نوصي باستخدام أحد الطريقتين (NN_CV , LR_CV) في تصنيف المقالات العربية.



الشكل (4) – نتائج التجارب مرتبة حسب الدقة



الشكل (5) – الزمن المستغرق لكل تجربة بالدقائق بنفس الترتيب في الشكل السابق

تفاصيل التجارب

قمنا باستخدام مصفوفة التحقق (الارتباك) Confusion Matrix في حساب الدقة:

		Predicted Values	
		Predicted Positiv	**Predicted Negatives**
Actual Values	**Positives**	True Positives (TP)	False Negatives (FN)
	Negatives	False Positives (FP)	True Negatives (TN)

الشكل (6) – مصفوفة التحقق (الارتباك)

القوانين:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$F1 - \text{score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

1. مصنف SVC مع ميزة TFIDF بالبارامترات التالية:

- TfidfVectorizer(sublinear_tf=True,strip_accents='unicode', analyzer='word', ngram_range=(1, 1), max_features =10000)
- clf = SVC()

Classification	Feature Extraction	dataset	accuracy	f1	precision	recall
SVC	TFIDF	1	96.895	0.969	0.969	0.969
		2	96.707	0.967	0.967	0.967
		3	95.454	0.955	0.955	0.955
		4	95.117	0.951	0.951	0.951
Average			**96.043**	**0.960**	**0.960**	**0.960**

الجدول (2) – نتائج تجربة SVC مع TFIDF

2. مصنف SVC مع ميزة Count Vector بالبارامترات التالية:

- CountVectorizer(mafx_features =10000)
- clf = SVC()

Classification	Feature Extraction	dataset	accuracy	f1	precision	recall
SVC	Count Vectorizer	1	95.231	0.952	0.953	0.952
		2	95.321	0.953	0.954	0.953
		3	94.171	0.942	0.942	0.942
		4	93.641	0.937	0.937	0.936
Average			**94.591**	**0.946**	**0.946**	**0.946**

الجدول (3) – نتائج تجربة SVC مع Count Vectorizer

3. مصنف Random Forest مع ميزة TFIDF بالبارامترات التالية:

- tfidf = TfidfVectorizer(max_features=1500, min_df=5, max_df=0.7,token_pattern=r"(?u)\b\w\w+\b",stop_words=stopwords_list)
- classifier = RandomForestClassifier(n_estimators=1000, random_state=0)

Classification	Feature Extraction	dataset	accuracy	f1	precision	recall
Random Forest	TFIDF	1	94.799	0.948	0.948	0.948
		2	94.360	0.944	0.944	0.944
		3	93.202	0.932	0.932	0.932
		4	92.805	0.928	0.928	0.928
Average			**93.791**	**0.938**	**0.938**	**0.938**

الجدول (4) – نتائج تجربة Random Forest مع TFIDF

4. مصنف Random Forest مع ميزة Count Vector بالبارامترات التالية:

- vectorizer = CountVectorizer(max_features=1500)
- classifier = RandomForestClassifier(n_estimators=1000, random_state=0)

Classification	Feature Extraction	dataset	accuracy	f1	precision	recall
Random Forest	Count Vectorizer	1	94.735	0.947	0.948	0.947
		2	94.247	0.942	0.943	0.942
		3	93.085	0.931	0.931	0.931
		4	92.694	0.927	0.927	0.927
Average			**93.690**	**0.937**	**0.937**	**0.937**

الجدول (5) – نتائج تجربة Random Forest مع Count Vectorizer

5. مصنف Logistic Regression مع ميزة TFIDF بالبارامترات التالية:

- tfidf = TfidfVectorizer()
- classifier = LogisticRegression()

Classification	Feature Extraction	dataset	accuracy	f1	precision	recall
Logistic Regression	TFIDF	1	96.360	0.964	0.964	0.964
		2	96.398	0.964	0.964	0.964
		3	95.279	0.953	0.953	0.953
		4	94.903	0.949	0.949	0.949
Average			**95.735**	**0.957**	**0.957**	**0.957**

الجدول (6) – نتائج تجربة Logistic Regression مع TFIDF

6. مصنف Logistic Regression مع ميزة Count Vector بالبارامترات التالية:

- vectorizer = CountVectorizer()
- classifier = LogisticRegression()

Classification	Feature Extraction	dataset	accuracy	f1	precision	recall
Logistic Regression	Count Vector	1	96.360	0.964	0.964	0.964
		2	96.305	0.963	0.963	0.963
		3	94.836	0.948	0.948	0.948
		4	94.358	0.944	0.944	0.944
Average			**95.465**	**0.955**	**0.955**	**0.955**

الجدول (7) – نتائج تجربة Logistic Regression مع Count Vectorizer

7. مصنف Decision Tree مع ميزة TFIDF بالبارامترات التالية:

- TfidfVectorizer(max_features=1500, min_df=5, max_df=0.7,token_pattern=r"(?u)\b\w\w+\b",stop_words=stopwords_list)
- clf = tree.DecisionTreeClassifier(random_state=5)

Classification	Feature Extraction	dataset	accuracy	f1	precision	recall
Decision Tree	TFIDF	1	84.752	0.848	0.848	0.848
		2	86.494	0.865	0.865	0.865
		3	84.940	0.850	0.850	0.849
		4	84.767	0.848	0.848	0.848
Average			**85.238**	**0.853**	**0.853**	**0.852**

الجدول (8) – نتائج تجربة Decision Tree مع TFIDF

8. مصنف Decision Tree مع ميزة Count Vector بالبارامترات التالية:

- vectorizer = CountVectorizer(max_features=1500)
- clf = tree.DecisionTreeClassifier(random_state=5)

Classification	Feature Extraction	dataset	accuracy	f1	precision	recall
Decision Tree	Count Vectorizer	1	87.040	0.870	0.871	0.870
		2	86.332	0.863	0.864	0.863
		3	85.140	0.852	0.852	0.851
		4	84.607	0.846	0.846	0.846
Average			**85.780**	**0.858**	**0.858**	**0.858**

الجدول (9) – نتائج تجربة Decision Tree مع Count Vectorizer

9. مصنف CNN مع ميزة Embedding بالبارامترات التالية:

- model = Sequential()
- model.add(Embedding(VOCAB_SIZE, 300, input_length= 1000))
- model.add(Conv1D(filters=128, kernel_size=4, padding='same', activation='relu'))
- model.add(MaxPooling1D(pool_size=2))
- model.add(Conv1D(filters=64, kernel_size=4, padding='same', activation='relu'))
- model.add(MaxPooling1D(pool_size=2))
- model.add(Conv1D(filters=32, kernel_size=4, padding='same', activation='relu'))
- model.add(MaxPooling1D(pool_size=2))
- model.add(Flatten())
- model.add(Dense(256, activation='relu'))
- model.add(Dense(5, activation='sigmoid'))

Classification	Feature Extraction	dataset	accuracy	f1	precision	recall
CNN	Embedding	1	95.196	0.952	0.953	0.952
		2	95.578	0.956	0.957	0.956
		3	94.595	0.946	0.947	0.946
		4	94.783	0.948	0.948	0.948
Average			**95.038**	**0.950**	**0.951**	**0.950**

الجدول (10) – نتائج تجربة CNN مع Embedding

10. مصنف NN مع ميزة Embedding بالبارامترات التالية:

- model = Sequential()
- model.add(Embedding(VOCAB_SIZE, 300, input_length= 1000))
- model.add(Flatten())
- model.add(layers.Dense(10, activation='relu'))
- model.add(layers.Dense(5, activation='sigmoid'))

Classification	Feature Extraction	dataset	accuracy	f1	precision	recall
NN	Embedding	1	95.444	0.954	0.955	0.954
		2	81.304	0.783	0.849	0.813
		3	74.745	0.714	0.709	0.747
		4	68.837	0.644	0.635	0.688
Average			**80.083**	**0.774**	**0.787**	**0.801**

الجدول (11) – نتائج تجربة NN مع Embedding

11.مصنف NN مع ميزة Count Vector بالبارامترات التالية:

- vectorizer = CountVectorizer(max_features=10000)
- model = Sequential()

model.add(layers.Dense(10, activation='relu'))

model.add(layers.Dense(5, activation='sigmoid'))

Classification	Feature Extraction	dataset	accuracy	f1	precision	recall
NN	Count Vectorizer	1	96.953	0.964	0.970	0.970
		2	96.418	0.964	0.964	0.964
		3	94.953	0.950	0.950	0.950
		4	94.481	0.945	0.945	0.945
Average			**95.701**	**0.956**	**0.957**	**0.957**

الجدول (12) – نتائج تجربة NN مع Count Vectorizer

12. مصنف NN مع ميزة TFIDF بالبارامترات التالية:

- tfidf = TfidfVectorizer(max_features=5000)
- model = Sequential()
  model.add(layers.Dense(10, activation='relu'))
  model.add(layers.Dense(5, activation='sigmoid'))

Classification	Feature Extraction	dataset	accuracy	f1	precision	recall
NN	Tfldf	1	96.417	0.964	0.964	0.964
		2	96.314	0.963	0.963	0.963
		3	95.059	0.951	0.951	0.951
		4	94.721	0.947	0.947	0.947
Average			95.628	0.956	0.956	0.956

الجدول (13) – نتائج تجربة NN مع TFIDF

13. مصنف XLNet بالبارامترات التالية:

- model = ClassificationModel('xlnet', 'xlnet-base-cased',num_labels=5, args={'num_train_epochs':1, 'train_batch_size':32, 'max_seq_length':128},use_cuda=False)

Classification	Feature Extraction	dataset	accuracy
XLNet	XLNet	1	20.225

الجدول (14) – نتائج تجربة XLNet

ملاحظة: تجربة XLNet تحتاج إلى المزيد من العمل والضبط للوصول لدقة مقبولة، لذلك لم يتم إدراجها في مخططات المقارنة.

تجارب تنظيف النصوص

قمنا بإجراء مجموعة تجارب بدون معالجة النصوص، وبدون خلط البيانات، ثم مقارنتها بالنتائج السابقة.

وجدنا أن تنظيف النصوص لا يؤثر بدرجة كبيرة على الدقة.

أما خلط البيانات فله تأثير ولكن بنسبة صغيرة كما هو موضح في قيم الجدول الآتي.

state	Classification	Feature Extraction	accuracy	Time	RAM ABOVE 1GB
with clean	NN	Count Vectorizer	94.559	4min 50s	0.86GB
with out clean	NN	Count Vectorizer	94.524	4min 32s	0.72GB
without shuffle	NN	Count Vectorizer	93.780	4min 15s	0.88 GB
with clean	Logistic Regression	TFIDF	94.853	7min 23s	0.68GB
with out clean	Logistic Regression	TFIDF	94.926	7min 12s	0.59 GB
without shuffle	Logistic Regression	TFIDF	94.361	7min	0.68 GB
with clean	DecisionTree	TFIDF	84.542	8min 12s	1.97 GB
with out clean	DecisionTree	TFIDF	84.515	4min 45s	1.05 GB
without shuffle	DecisionTree	TFIDF	83.530	5min 32s	1.44 GB
with clean	Random Forest	TFIDF	92.8050	1h 8min 7s	3.12 GB
with out clean	Random Forest	TFIDF	92.751	1h 29min 3s	1.7 GB
without shuffle	Random Forest	TFIDF	92.283	1h 23min 56s	2.64 GB

الجدول (15) – نتائج تجارب تنظيف النصوص وعدمه، وخلط البيانات أو ترتيبها على عبر عدة خوارزميات

الأكواد

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

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

خامساً: المراجع

[1] Boukil, Samir, et al. "Arabic text classification using deep learning technics." International Journal of Grid and Distributed Computing 11.9 (2018): 103-114.

[2] MAHLOUS, Ahmed Redha; AL-LAITH, Ali. Fake news detection in Arabic tweets during the COVID-19 pandemic. Int J Adv Comput Sci Appl, 2021.

[3] ALSUDIAS, Lama; RAYSON, Paul. COVID-19 and Arabic Twitter: How can arab world governments and public health organizations learn from social media?. 2020.

[4] ESSAM, Nader, et al. Location Analysis for Arabic COVID-19 Twitter Data Using Enhanced Dialect Identification Models. *Applied Sciences*, 2021, 11.23: 11328.

[5] ALTURAYEIF, Nora; LUQMAN, Hamzah. Fine-Grained Sentiment Analysis of Arabic COVID-19 Tweets Using BERT-Based Transformers and Dynamically Weighted Loss Function. *Applied Sciences*, 2021, 11.22: 10694.

[6] Mohamed BINIZ. DataSet for Arabic Classification. Version 2. 2018.

[7] Nagesh Singh Chauhan. Decision Tree Algorithm, Explained. 2022.

[8] Hassan Khanfari. Random Forest Algorithm. 2021.

[9] Rasool Hasan. Logistic Regression Algorithm. 2020.

[10] Noel Bambrick. Support Vector Machines: A Simple Explanation. 2016.

المحتويات

أولاً: المقدمة 2

المميزات والعيوب 3

ثانياً: الأعمال المشابهة 3

ثالثاً: آلية عمل البرنامج 5

مخطط النظام 5

1- تجهيز البيانات 5

التأكد من صحة النتائج 7

2- تنظيف البيانات 7

3- استخراج الميزات 9

4- التصنيف: 9

رابعاً: التجارب والنتائج 10

نتائج التجارب 11

تفاصيل التجارب 13

تجارب تنظيف النصوص 19

الأكواد 19

خامساً: المراجع 20