

HIAST

الجمهوريّة العربيّة السّوريّة
المعهد العالي للعلوم التطبيقية والتكنولوجيا
قسم النظم المعلوماتية
العام الدراسي 2018/2019

تقرير مشروع سنة رابعة

تصنيف الطرود الشبكيّة بالاعتماد على خوارزميّات تعلم الآلة

Network Traffic Classification Using Machine
Learning Algorithms

إعداد

يزن التّسر

إشراف

م. نضال الشاطر

10/9/2019

شکر

أتوجه بجزيل الشكر إلى أسرة المعهد العالي لمؤسسة تعليمية راقية. وأنص بالشکر قادر قسم المعلوميات.

وأخص بالذكر المشرف المهندس نصال الشاطر الذي أعطاني جزءاً كبيراً من وقته وجهني في جميع خطواته هذا المشروع.

الخلاصة

يتم بشكل يومي اكتشاف المئات من البرمجيات الخبيثة، وبالتالي فإنَّ الأدوات التي تعتمد على معرفة بنية البيانات بشكل تام لكشف مثل هذه البرمجيات ستواجه صعوبة كبيرة لأداء مهمتها بشكل فعال عندما تكون هذه البرمجيات حديثة المنشأ ولم يتم تدريب الأداة لاكتشاف بنيتها بعد. لهذا السبب كان لا بدَّ من إنشاء أدوات لا تعتمد على البنية في تحليلها للبيانات المُتبادلة على الشبكة.

يقدم هذا العمل أداة قادرة على التمييز بين الطُّرُود الشبكيَّة الخبيثة والطُّبِيعيَّة، وذلك باستخدام العديد من خوارزميات تعلم الآلة، التي تقوم باكتشاف الأنماط والعلامات المميزة للطُّرُود الشبكيَّة الخبيثة من خلال تدربها على قواعد معطيات مصممة لهذا الغرض. كما تتيح هذه الأداة الكثير من الخيارات أمام المستخدم، والتي تسمح له إلى حدٍ ما بالتحكم بالطريقة المُتبعة في القيام بعملية الفرز، الأمر الذي يجعل هذه الأداة ملائمة لطيفٍ أوسع من التطبيقات العملية.

كما توفر هذه الأداة إمكانية إنشاء مخططات بيانيَّة لمقارنة أداء كل من الخوارزميات المُتاحة عند تغيير قيم الخيارات المستخدمة فيها. إضافةً إلى مقارنة أداء الخوارزميات بين بعضها البعض، مما يسهل على المستخدم عملية اختيار الخوارزمية والخيارات المناسبين للتطبيق الذي يريد. كما تسهل هذه المخططات عملية عرض النتائج بشكل مرئي واضح ومحبب.

نهايةً نستعرض في هذا العمل النتائج التي قدّمتها كل من الخوارزميات ونقارن فيما بينها لنتوصل في النهاية إلى الخيار الأمثل لمسألتنا هذه.

Abstract

Every day, hundreds of new malware samples are being discovered. Therefore, tools that depend solely on knowing the structure of data will face great difficulties when attempting to do discover a newly created malware program that weren't introduced to these tools in the training process. That's why we need tools that don't depend solely on the structure of data to discover these programs.

This work offers a tool that can discover malware programs using machine learning algorithms, which use features in its training process “using datasets specialized for this purpose” to discover a pattern that allows it to differentiate between normal and malware programs. This tool also gives the user a large set of options that enables him to control how the classification process is done, which makes this tool suitable for many fields of work.

This tool can also draw plots, which enables us to easily compare between the results we get when using different algorithms and options, allowing us to choose these options that best serve our needs.

Finally, we use this tool to view the results provided by the machine learning algorithms to finally decide which algorithm and options suit this application the best.

المحتويات

I.....	الخلاصة
II.....	Abstract
V.....	قائمة الأشكال
IX.....	قائمة الجداول
X.....	الاختصارات
1.....	مقدمة عامة
2.....	الهدف من المشروع
2.....	المخطط العام للعمل
2.....	متطلبات المشروع
4.....	الفصل الأول: الدراسة النظرية
4.....	1.1 أصناف خوارزميات تعلم الآلة
4.....	1.1.1 التعلم تحت الإشراف
5.....	2.1.1 التعلم دون إشراف
5.....	3.1.1 التعلم التعزيزي
6.....	2.1 خوارزميات تعلم الآلة المستخدمة
6.....	1.2.1 الانحدار اللوجستي
9.....	2.2.1 شجرة القرار
11.....	3.2.1 الغابة العشوائية
12.....	4.2.1 أقرب k جار
14.....	5.2.1 متجهات الدعم
15.....	6.2.1 بايز مع التوزع الغاوسي
16.....	7.2.1 العنقة بـ k قيمة متوسطة

الفصل الثاني: التنفيذ والاختبارات والنتائج	20
1.2 بيئة العمل	20
1.1.2 لغة Python	20
2.1.2 أداة Jupyter Notebook	20
2.2 المكتبات البرمجية المستخدمة	21
3.2 النتائج العملية	22
1.3.2 قواعد المعطيات المستخدمة	22
2.3.2 تهيئة قواعد المعطيات	24
3.3.2 تقسيم الحقول إلى مجموعات	25
4.3.2 نتائج تطبيق خوارزميات تعلم الآلة	27
1.4.3.2 الانحدار логисти	28
2.4.3.2 شجرة القرار	33
3.4.3.2 الغابة العشوائية	36
4.4.3.2 أقرب k جار	39
5.4.3.2 متوجهات الدعم	44
6.4.3.2 بايز مع التوزع الغاوسي	48
7.4.3.2 العنقدة ب k قيمة متوسطة	50
5.3.2 مقارنة أداء الخوارزميات	51
الفصل الثالث: واجهات التخاطب مع المستخدم	54
الخاتمة والأفاق المستقبلية	66
الملاحق	68
المراجع	70

قائمة الأشكال

6.....	الشكل (1) منهجية التعلم التعزيزي
7.....	الشكل (2) المخطط البياني للتابع (z) المستخدم في خوارزمية Logistic Regression
8.....	الشكل (3) مثال عن خوارزمية Logistic Regression
10.....	الشكل (4) مثال عن خوارزمية Decision Tree
11.....	الشكل (5) مثال عن خوارزمية Random Forest
13.....	الشكل (6) مثال عن خوارزمية K Nearest Neighbors
14.....	الشكل (7) مثال عن خوارزمية Support Vector
17.....	الشكل (8) مثال عن الـ Elbow Method
18.....	الشكل (9) مثال عن خوارزمية K Means Clustering
25.....	الشكل (10) مخطط لقيم أحد المعاملات من مجموعة المعطيات "Very Different Features"
26.....	الشكل (11) مخطط لقيم أحد المعاملات من مجموعة المعطيات "Different Features"
27.....	الشكل (12) مخطط لقيم أحد المعاملات من مجموعة المعطيات "Very Similar Features"
28.....	الشكل (13) مخطط يوضح أثر زيادة حجم معطيات التدريب على دقة النتائج التي تقدمها خوارزمية الانحدار логистي من أجل قيم المعطيات قبل عملية التسوية
29.....	الشكل (14) مخطط يوضح أثر زيادة حجم معطيات التدريب على دقة النتائج التي تقدمها خوارزمية الانحدار логистي من أجل قيم المعطيات بعد عملية التسوية
30.....	الشكل (15) مخطط يوضح أثر تسوية قيم المعطيات على دقة النتائج التي تقدمها خوارزمية الانحدار логистي "المنحدرات السوداء للقيم قبل التسوية، والحرماء للقيم بعد التسوية"
31.....	الشكل (16) مخطط يوضح أثر زيادة حجم معطيات التدريب على المدة الزمنية اللازمة لانهاء عملية التدريب من أجل خوارزمية الانحدار логистي
32.....	الشكل (17) مخطط يوضح أثر زيادة حجم معطيات التنبؤ على المدة الزمنية اللازمة لانهاء عملية التنبؤ من أجل خوارزمية الانحدار логистي
32.....	الشكل (18) مقارنة بين زمن التدريب و زمن التنبؤ عند استخدام خوارزمية الانحدار логистي
33.....	الشكل (19) مخطط يوضح أثر زيادة حجم معطيات التدريب على دقة النتائج التي تقدمها خوارزمية شجرة القرار

الشكل (20) مخطط يوضح أثر زيادة حجم معطيات التدريب على المدة الزمنية اللازمة لانهاء عملية التدريب من أجل خوارزمية شجرة القرار	34.....
الشكل (21) مخطط يوضح أثر زيادة حجم معطيات التنبؤ على المدة الزمنية اللازمة لانهاء عملية التنبؤ من أجل خوارزمية شجرة القرار	35.....
الشكل (22) مقارنة بين زمن التدريب و زمن التنبؤ عند استخدام خوارزمية شجرة القرار	35.....
الشكل (23) مخطط يوضح أثر زيادة حجم معطيات التدريب على دقة النتائج التي تقدمها خوارزمية الغابة العشوائية	36.....
الشكل (24) مخطط يوضح أثر زيادة حجم معطيات التدريب على المدة الزمنية اللازمة لانهاء عملية التدريب من أجل خوارزمية الغابة العشوائية	37.....
الشكل (25) مخطط يوضح أثر زيادة حجم معطيات التنبؤ على المدة الزمنية اللازمة لانهاء عملية التنبؤ من أجل خوارزمية شجرة القرار	38.....
الشكل (26) مقارنة بين زمن التدريب و زمن التنبؤ عند استخدام خوارزمية الغابة العشوائية	38.....
الشكل (27) مخطط يوضح أثر زيادة حجم معطيات التدريب على دقة النتائج التي تقدمها خوارزمية أقرب k جار من أجل قيم المعطيات قبل عملية التسوية	39.....
الشكل (28) مخطط يوضح أثر زيادة حجم معطيات التدريب على دقة النتائج التي تقدمها خوارزمية أقرب k جار من أجل قيم المعطيات بعد عملية التسوية	40.....
الشكل (29) مخطط يوضح أثر تسوية قيم المعطيات على دقة النتائج التي تقدمها خوارزمية أقرب k جار "المنحدرات السوداء للقيم قبل التسوية، والحرماء للقيم بعد التسوية	40.....
الشكل (30) مخطط يوضح أثر زيادة حجم معطيات التدريب على المدة الزمنية اللازمة لانهاء عملية التدريب من أجل خوارزمية أقرب k جار	42.....
الشكل (31) مخطط يوضح أثر زيادة حجم معطيات التنبؤ على المدة الزمنية اللازمة لانهاء عملية التنبؤ من أجل خوارزمية أقرب k جار	42.....
الشكل (32) مقارنة بين زمن التدريب و زمن التنبؤ عند استخدام خوارزمية أقرب k جار	43.....
الشكل (33) مخطط يوضح أثر زيادة حجم معطيات التدريب على دقة النتائج التي تقدمها خوارزمية متوجه الدعم من أجل قيم المعطيات قبل عملية التسوية	44.....
الشكل (34) مخطط يوضح أثر زيادة حجم معطيات التدريب على دقة النتائج التي تقدمها خوارزمية متوجه الدعم من أجل قيم المعطيات بعد عملية التسوية	44.....

الشكل (35) مخطط يوضح أثر تسوية قيم المعطيات على دقة النتائج التي تقدمها خوارزمية متوجه الدعم "المنحنىات السوداء للقيم قبل التسوية، والحراء للقيم بعد التسوية 45
الشكل (36) مخطط يوضح أثر زيادة حجم معطيات التدريب على المدة الزمنية اللازمة لانهاء عملية التدريب من أجل خوارزمية متوجه الدعم 46
الشكل (37) مخطط يوضح أثر زيادة حجم معطيات التنبؤ على المدة الزمنية اللازمة لانهاء عملية التنبؤ من أجل خوارزمية متوجه الدعم 46
الشكل (38) مقارنة بين زمن التدريب و زمن التنبؤ عند استخدام خوارزمية الانحدار الوجستي 47
الشكل (39) مخطط يوضح أثر زيادة حجم معطيات التدريب على دقة النتائج التي تقدمها خوارزمية بايز من أجل قيم المعطيات قبل عملية التسوية 47
الشكل (40) مخطط يوضح أثر زيادة حجم معطيات التدريب على دقة النتائج التي تقدمها خوارزمية بايز من أجل قيم المعطيات بعد عملية التسوية 49
الشكل (41) مخطط يوضح أثر تسوية قيم المعطيات على دقة النتائج التي تقدمها خوارزمية بايز "المنحنىات السوداء للقيم قبل التسوية، والحراء للقيم بعد التسوية 49
الشكل (42) تغيرات دقة خوارزمية (K Means Clustering) عند تغيير حجم معطيات التدريب 50
الشكل (43) دقة كل من الخوارزميات عندما يكون حجم معطيات التدريب ضمن المجال [0, 5000] 51
الشكل (44) مقارنة بين دقة خوارزمية شجرة القرارات وخوارزمية الغابة العشوائية 53
الشكل (45) واجهة "Setup" لاختيار ملفات قواعد المعطيات 54
الشكل (46) عملية الاختيار الديناميكية لملفات قواعد المعطيات 55
الشكل (47) واجهة "Run Mode" 55
الشكل (48) صيغة النتائج عند استخدام واجهة "Run Model" 56
الشكل (49) واجهة توليد النماذج المتعددة تحضيراً لعمليات المقارنة 57
الشكل (50) واجهة التعامل مع خوارزمية K Means Clustering 59
الشكل (51) صيغة النتائج عند تشغيل واجهة K Means Clustering 59
الشكل (52) واجهة رسم مخططات توضيح أثر تغيير قيم المعاملات على قيم المتحول الهدف 60
الشكل (53) واجهة رسم المخططات التفاعلية لتغيرات أداء الخوارزميات بتغيير الخيارات المطلوبة 61

الشكل (54) شكل المخطّطات الناتجة عند تشغيل واجهة رسم المخطّطات التّفاعلية	62
الشكل (55) واجهة رسم مخطّطات المقارنة	63
الشكل (56) الأشكال الناتجة عن تشغيل واجهة رسم مخطّطات المقارنة	64
الشكل (57) اختفاء تسميات المحاور عند عدم توافق التّسميات بين مخطّطات المقارنة	65

قائمة الجداول

19.....	الجدول 1- مقارنة نظرية بين خوارزميات تعلم الآلة
52.....	الجدول 2- مقارنة نتائج خوارزميات تعلم الآلة

الاختصارات

LR: Logistic Regression	الانحدار اللوجستي
DT: Decision Tree	شجرة القرار
RF: Random Forest	الغابة العشوائية
KNN: K Nearest Neighbors	أقرب k جار
SV: Support Vector	متجه الدّعم
GNB: Gaussian Naïve Bayes	بايز مع التوزّع الغاوسي
CSV: Comma Separated Values	ملفّات القيم المفصولة بفواصل
WCSS: Within Cluster Sum of Squares	مجموع مربعات الفروق ضمن العنقود
CWI: Centrum Wiskunde & Informatica	مركز العلوم والحساب الآلي

مقدمة عامة

أبلغت العديد من مؤسسات البحث الأمني عن اكتشاف المئات من نماذج جديدة للبرامج الخبيثة يومياً، واكتشاف الملايين منها شهرياً، والكثير منها نابع من نفس أسر البرمجيات الخبيثة.

ووفقاً لنقرير حالة البرامج الخبيثة لعام 2017، فإنَّ برنامج Malwarebytes قد اكتشف أكثر من مليار حادث متعلق بالبرامج الخبيثة بين يونيو ونوفمبر لعام 2016 [1]. وهذا يمثل مشكلة لمحلي البرامج الخبيثة، فالأدوات التقليدية لا تسمح لهم بمواكبة الأنواع الجديدة من بنى البيانات، حيث تعتمد الأدوات التقليدية على معرفة بنية البيانات بدلاً من أن تكون قادرة على معالجة البيانات بغض النظر عن بنيتها، مما يجعل عملية تحليل البرامج الخبيثة عملية بطيئة وغير فعالة في معظم الأحيان.

كما أنَّ مُنشئي البرامج الخبيثة يعملون بجدٍ أيضاً، فإنَّ تطور أدوات التصني للبرمجيات الخبيثة يقابله دوماً تطور في هذه البرمجيات، حيث أنَّ مطوري هذه البرمجيات باتوا يستخدمون تقنيات مكافحة التحليل مثل مكافحة التفكير، مكافحة تصحيح الأخطاء، والتشفير، واستخدام تقنية "virtual machine detection" لمعرفة فيما لو كان يتم تفعيل البرنامج على آلية افتراضية أم لا، مما يجعل طرق التحليل التقليدية عاجزة عن التصني لمثل هذه البرامج. وبالتالي، فنظرًا لتزايد عدد أنواع البرامج الضارة وأنواعها وذكائها بشكل دائم، كان لا بدًّ من إنشاء أدوات جديدة قادرة على القيام بعملية التحليل بشكل مبسط وألبي دون الاعتماد على بنية البيانات.

الهدف من المشروع

نَهَدَفُ فِي هَذَا الْمَشْرُوْع إِلَى إِنْشَاء أَدَة تَقْوِيم بِتَحْلِيلِ الْبَيَانَات غَيْرِ الْمُشَفَّرَة الَّتِي تَمَّ تِبَادِلُهَا عَبْر شبَّكَة مَا. حيث تحاول باستخدام خوارزميات تعلم الآلة إيجاد الأنماط والخصائص المميزة للطُّرُود الخبيثة ومن ثم تستخدم هذه المعرفة المتراكمة التي حصلت عليها في عملية التّدريب للتّنبؤ بطبيعة الطُّرُود الجديدة. وإن ما يميّز هذه التقنية هو قدرتها على التعامل مع الطُّرُود الشبكيّة بشكل مباشر، واستخدام الخصائص المستنبطَة من هذه الطُّرُود لكشف طبيعتها بعض النّظر عن بنية البيانات التي تحملها هذه الطُّرُود، الأمر الذي يجعل عملية اكتشاف البرمجيات الخبيثة الجديدة أكثر فعالية. كما يجدر بالذكر أنَّ خوارزميات تعلم الآلة يمكن تدريبيها على معطيات جديدة بشكل آلي سريع وفعال، مما يجعل عملية تطويرها لتواكب تطور البرمجيات الخبيثة أمراً سهلاً للغاية مقارنةً بتطوير الأدوات التقليدية.

المخطط العام للعمل

تُقْسَمُ هَذِه الوثيقَة إِلَى ثَلَاث أَجزاء رَئِيسِيَّة، يَتَضَمَّنُ الْجُزْءُ الْأَوَّل دراسة نظرية بغية تقديم أساس نظري للخوارزميات المستخدمة. ويعرض الجزء الثاني دراسة للنتائج التي قدّمتها كل من هذه الخوارزميات، مع مقارنة هذه النتائج مع بعضها البعض. في حين يعرض الجزء الثالث شرحاً عن واجهات المستخدم وأية عمليات ووظائف التي تقوم بها. يلي هذه الأجزاء خاتمة تلخيص محتويات الوثيقة وتطرح آفاقاً مستقبلية للعمل.

متطلبات المشروع

المتطلبات الوظيفية:

1. يجب أن يكون تعامل المستخدم مع النظام عبر واجهات بسيطة لسهولة الاستخدام لمن ليس لديهم خبرة برمجية كافية.
2. يجب أن يسمح النظام للمستخدم باختيار ملفات التدريب والاختبار بشكل ديناميكي أثناء عمل البرنامج.
3. يجب أن يسمح النظام للمستخدم باختيار خوارزمية تعلم الآلة التي يرغب بها مع السماح له بالتحكم بحجم معطيات التدريب والاختبار لتناسب حاجته.
4. يجب أن يسمح النظام للمستخدم بمقارنة أداء كل من الخوارزميات "من ناحية دقة النتائج وزمان التدريب و زمن التّنبؤ" وذلك عند تغيير حجم معطيات التدريب والاختبار عن طريق السماح للمستخدم برسم مخططات بيانية توضح تأثير هذه التغييرات على الأداء.

5. يجب أن يسمح النظام للمستخدم بمقارنة أداء الخوارزميات بين بعضها عن طريق السماح للمستخدم برسم مخططات بيانية توضح الفروق في أداء هذه الخوارزميات.

6. يجب أن يسمح النظام للمستخدم بحفظ المخططات التي رسمها ضمن البرنامج مع منحه إمكانية تعديل مسميات المخططات.

المتطلبات غير الوظيفية:

1. سهولة التعامل: يجب أن يكون البرنامج سهل الاستخدام لمن ليس لديهم خبرة برمجية.
2. قابلية التوسيع: يجب أن توجد إمكانية لإضافة خوارزميات جديدة وخيارات إضافية للنظام.

الفصل الأول

الدراسة النظرية

نقدم في هذه الفصل تحليلًا للدراسات النظرية والمرجعيات والتقنيات المعتمدة في هذا المشروع، حيث سنورد شرحاً عن أهم خوارزميات تعلم الآلة المتّبعة في تصنيف المعطيات، ومناقشة حسناً وسليّنات كل من هذه الخوارزميات مع المقارنة فيما بينها. اعتمدنا في دراستنا هذه على بعض الواقع الرسمية العاملة في مجال تعلم الآلة وعلم البيانات (Data Science) [5][6].

بشكل عام توجد خوارزميات تعلم الآلة تقوم بتصنيف البيانات إلى مجموعات معينة وتسمى بخوارزميات التصنيف (Classification Algorithms)، في حين توجد خوارزميات أخرى تهدف إلى إعطاء قيم عدديّة لبيانات الخرج عوضاً عن التصنيف، وبما أن هذا المشروع يهدف إلى تصنیف الطرود الشبكية، فلن نتطرق إلى هذا النوع من الخوارزميات.

1.1 أصناف خوارزميات تعلم الآلة:

تصنّف خوارزميات تعلم الآلة إلى ثلاثة أصناف، حيث أنَّ كل صنف من هذه الأصناف يعتمد طرائق مختلفة في استخدام البيانات في عملية التعلم.

1.1.1 التعلم تحت الإشراف (Supervised Learning):

التعلم تحت الإشراف هو المكان حيث توجد فيه بيانات المدخلات (X) وبيانات المخرجات (Y) من البداية وفي نفس الوقت، تستخدم الخوارزمية كل هذه البيانات لتتعلم كيفية إيجاد الرابط لتحويل أو تعين المدخلات هذه إلى المخرجات، في محاولة للاقتراب قدر الإمكان من دالة " $Y = f(X)$ " تمكّنها عندما يكون لديها بيانات إدخال جديدة (X) من التنبؤ ببيانات المخرجات (Y) لنّاك البيانات.

يطلق على هذا النوع من التعلم تعلم تحت الإشراف لأن المعطيات تكون حاوية لبيانات المخرجات وبالتالي يكون هنالك طرف مشرف قام باستنتاج قيم بيانات المخرجات، ويمكن اعتبار أن هذه الطرف يعرف الإجابات الصحيحة.

تتحمّل مشاكل التعلم تحت الإشراف بشكل أكبر حول الانحدار (Regression) والتصنيف (Classification). حيث يتعامل الانحدار بشكل عام مع المعاملات ذات القيم المستمرة، بينما يتعامل التصنيف مع المعاملات التي تأخذ قيمها من مجموعة محدودة من القيم.

ومن أهم خوارزميات هذا النوع توجد خوارزمية شجرة القرارات والانحدار اللوجستي وغيرها الكثير.

2.1.1 التعلم دون إشراف (Unsupervised Learning):

التعلم دون إشراف هو المكان الذي توجد فيه بيانات المدخلات (X) ولا توجد بيانات المخرجات في المقابل، حيث تهدف إلى العثور على الأنماط في البيانات.

يطلق عليه التعلم دون إشراف لأنه وعلى عكس التعليم تحت الإشراف لا توجد إجابات صحيحة وليس هناك معلم.

خوارزميات التعلم دون إشراف يمكنها تأدية مهام معالجة أكثر تعقيداً من أنظمة التعلم تحت الإشراف. أفضل استخدام للتعلم دون إشراف هو عندما لا تتوفر لديك بيانات حول النتائج المرجوة، مثل تحديد سوق مستهدف لمنتج جديد. ومع ذلك، إذا أردنا الحصول على فهم أفضل لبيانات المستهلكين الحالية، فإن التعلم تحت الإشراف هو الأسلوب الأمثل.

تمحور مشاكل التعلم دون الإشراف بشكل أكبر حول التّجميع (Clustering). حيث يتم تقسيم المعطيات إلى مجموعات منفصلة لها خصائص متشابهة.

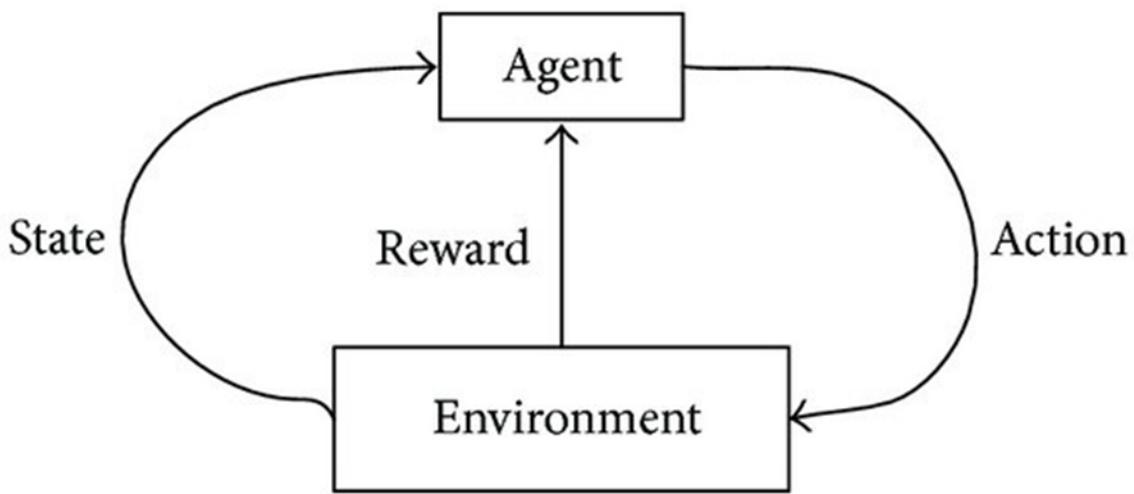
من أشهر خوارزميات هذا النوع توجد خوارزمية العنقة ب k قيمة متوسطة.

3.1.1 التعلم التعزيزي (Reinforced Learning):

التعلم تعزيزي يتمحور حول اتخاذ الإجراء المناسب لتعظيم المكافأة (Reward) في موقف معين، في محاولة للعثور على أفضل سلوك ممكن أو المسار الذي يجب أن يتخذه في موقف محدد.

يختلف التعلم التعزيزي عن التعلم تحت إشراف في طريقة التعامل مع البيانات التدريب، حيث إن في التعلم تحت إشراف يتم تدريب النموذج على الإجابة الصحيحة نفسها، ولكن في التعلم التعزيزي، لا توجد إجابة في الأساس، ولكن الأداة تقرر ما يجب عليها فعله لأداء المهمة المطلوبة، دون وجود أي بيانات للتدريب.

يوضح الشكل التالي كيفية عمل التعلم التعزيزي:



الشكل (1) منهجهة التعلم التعزيزي

تم عملية التدريب بالشكل التالي:

تقوم الخوارزمية أولاً بقراءة عينة من معطيات التدريب، والتي تعبر عن حالة ما. ثم تقوم هذا الخوارزمية باتخاذ قرار ما بخصوص هذه الحالة. تتم بعد ذلك عملية تقييم لهذا القرار ضمن بيئه العمل، و يتم منح "مكافأة" للخوارزمية بحسب القرار الذي اتخذه، يتم بعد ذلك تعديل النموذج بحسب الحالة، القرار والمكافأة.

في هذا المشروع، سنصبّ معظم تركيزنا على التعلم تحت الإشراف، حيث سنقوم باستخدام ستة خوارزميات تعتمد على التعلم تحت الإشراف، في حين لن نناقش سوى خوارزمية واحدة تعتمد على التعلم دون إشراف، وذلك لأن هذا النوع من الخوارزميات ليس مناسباً للمعطيات التي نمتلكها، حيث أن المعطيات لدينا تحتوي قيم بيانات الخرج، والتي لن يتم الاستفادة منها في حال استخدمنا خوارزمية تعتمد على التعلم دون إشراف. كما أنها لن نناقش أي من خوارزميات التعلم التعزيزي لأنها بعيدة عن هدف المشروع الأساسي، إلا وهو تصنيف المعطيات.

2.1 خوارزميات تعلم الآلة المستخدمة:

1.2.1 الانحدار اللوجستي (Logistic Regression "LR") :

مع أنَّ الاسم يوحي بأنَّ هذا النموذج هو نموذج انحدار "أي يقوم باعطاء قيم لبيانات الخرج"، إلا أنها في الحقيقة خوارزمية تصنيف.

تقوم هذه الخوارزمية بالتنبؤ باحتمالية وقوع الحدث من خلال ملائمة البيانات على وظيفة لوجستية، لذلك يُعرف أيضًا باسم الانحدار اللوجستي. وبما أنه يتتبّع بالاحتمالات، فالقيم الناتجة تقع في المجال [1, 0] (الارتباط الموجب له قيمة موجبة والعكس بالعكس).

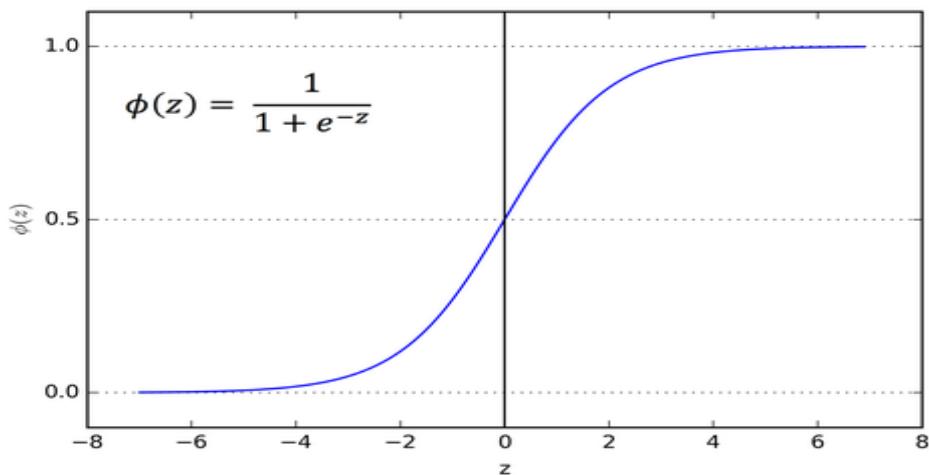
يتم استخدام المعادلة الرياضية التالية لحساب قيمة المتحول z :

$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 \dots \quad (1)$$

حيث \dots, x_1, x_2 هي قيم معاملات الدخل، $\theta_0, \theta_1, \dots, \theta_n$ هي معاملات النموذج الرياضي و z هي قيمة الخرج المبدئية للنموذج.

مبدئيًّا يتم اختيار قيم معاملات النموذج بشكل عشوائي، وبعد كل عملية تدريب على كل entry من معطيات التدريب، يتم تعديل قيم هذه المعاملات بحيث تصبح قيمة الخطأ أصغرية، وفي حال كانت العلاقة بين المتحول الهدف ومعاملات الدخل هي بالفعل علاقة خطية، أو لها علاقة ذات طبيعة شبه خطية "أي أن العلاقة التي تربطهم يمكن تقريرها بخطأ مقبول من علاقة خطية"، فإن قيم معاملات النموذج ستستقر بعد عدد معين من عينات التدريب للحصول على النموذج النهائي، أما في حال كانت العلاقة بعيدة عن الطبيعة الخطية، فإن قيم هذه المعاملات ستبقى في حالة تغيير دائمة ولن يستقر النموذج وبالتالي لن نحصل على النتائج المرجوة.

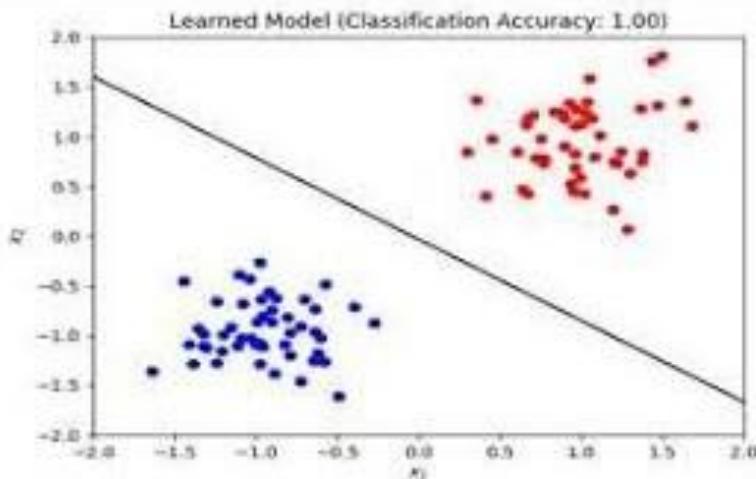
نلاحظ أنَّ قيمة z الآن هي قيمة مستمرة ليس بالضرورة أن تعبّر عن احتمال، وللحصول على قيمة بين 0 و1 يتم استخدام التابع $\frac{1}{1+e^{-z}} = g(z)$ والذي له الشكل التالي:



الشكل (2) المخطط البياني للتابع $(z)g$ المستخدم في خوارزمية Logistic Regression

نلاحظ أنَّ قيم التابع واقعة بين القيمتين 0 و 1 بشكل منتظم، وعند تطبيق هذا التابع على المتحول z سنحصل على قيمة h_θ والتي تعبّر عن احتمال انتفاء العينة هذه إلى الفئة

الأولى، ففي حال كان الاحتمال أكبر من النصف، تُعتبر العينة من الفئة الأولى، والعكس بالعكس.



الشكل (3) مثال عن خوارزمية *Logistic Regression*

الحسنات:

- 1- إنَّ هذه الخوارزمية سهلة الفهم، كما أنَّها سريعة وبسيطة التنفيذ على الحاسوب.
- 2- تقوم باعطاء قيم معينة ضمن المجال [1, 1] لكل من المعاملات في بيانات الدخل، وهذه القيم تدلّ بدورها على درجة ارتباط المتحوّلات الهدف (بيانات الخرج) مع كلّ من هذه المعاملات (كل ما زادت القيمة المطلقة، زاد الارتباط)، كما تدلّ على جهة الارتباط (بحسب الاشارة)، وبالتالي يمكن حساب احتمال خطأ النتائج التي يعطيها النموذج.
- 3- في حال كان الارتباط خطياً بين معاملات الدخل والمتحوّلات الهدف فإنَّ هذا النموذج يعطي أداءً ودقةً ممتازة.
- 4- لا تحتاج إلى حجم كبير من معطيات الدخل (التي يتم استخدامها في عملية التدريب) لكي تصل إلى النموذج المطلوب (أي حساب قيم الارتباط لكل من معاملات الدخل).

السيئات:

- 1- في حال كانت طبيعة الارتباط بعيدة عن الخطية ولو بمعدل بسيط، فإنَّ أداء هذه الخوارزمية ودقة نتائجها سينتهي بسرعة، وفي حال كان الارتباط بعيداً عن الطبيعة الخطية، فلن يعود بالأمكان الاستفادة من نتائج هذه الخوارزمية.

2- إنَّ وجود معاملات مرتبطة خطياً مع بعضها البعض سيكون له تأثير سلبي على دقة النتائج، ونفس الأمر ينطبق على وجود الكثير من القيم الشاذة.

3- تواجه هذه الخوارزمية صعوبة في التعامل مع المعاملات ذات الطبيعة التصنيفية (أي التي لها مجموعة قيم محددة) حيث أنها لا تستطيع الاستفادة بشكل جيد من المعلومات التي توفرها هذه المعاملات.

4- قد تحتاج إلى القيام بتسوية قيم المعاملات (Normalization) لكي نحصل على نتائج أكثر دقة (تسوية قيم مجموعة من المعاملات تعني جعل قيم كل من هذه المعاملات تتواجد ضمن نفي مجال القيم).

2.2.1 شجرة القرار (“DT”): (Decision Tree “DT”)

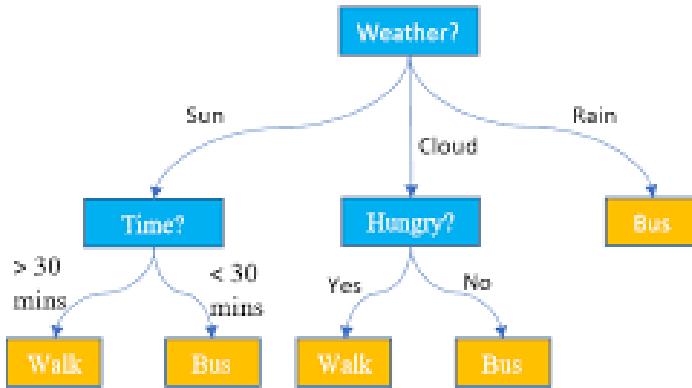
يتَّخذ نموذج شجرة القرار شكل الشجرة الثنائية، وتمثل كل عقدة في هذه الشجرة أحد معاملات الدخل (x) ونقطة انقسام معتمدة على هذا المتغير، حيث أن كل عقدة تحوي شرطاً ما على أحد المعاملات، وبحسب تحقق الشرط أو عدمه، يتم اختيار العقدة التالية في الشجرة، وتتكرر العملية هذه عدداً من المرات حتى الوصول إلى أحد أوراق الشجرة، التي بدورها تحتوي قرار التصنيف النهائي.

أثناء عملية بناء الشجرة، يتم اختيار جذر الشجرة بحيث يكون المعامل المرتبط بها ذا انترويبيَّة عظمى، وفي كل مستوى من المستويات اللاحقة لبناء الشجرة، يتم اختيار المعامل التالي بحيث تكون كمية المعلومات المتبادلة بين المستوى الحالي والمستوى السابق أعظمياً، يتم حساب هذه القيمة باستخدام العلاقات الرياضية التالية:

$$H(s) = -\sum P_c \cdot \log(P_c) \quad (2)$$

$$IG(s) = H(s) - \sum P_t \cdot H(t) \quad (3)$$

حيث أنَّ $H(s)$ هو انترويبيَّة المعامل s و $IG(s)$ هو كمية المعلومات المتبادلة.



الشكل (4) مثال عن خوارزمية *Decision Tree*

الحسنات:

1- يمكنها التعامل مع معظم أنواع الدخل دون مشاكل تذكر، وبالخصوص المعطيات التصنيفية، كما أنها تعامل مع المعطيات ذات العدد الكبير من المعاملات بشكل جيد، ويمكنها التعامل مع معظم أنواع الارتباطات وليس فقط الخطية منها.

2- لا حاجة للقيام بمعالجة مسبقة للبيانات قبل تطبيق الخوارزمية عليها.

3- يمكنها توليد تفسير مقبول للنتائج والذي يعتمد على الطريق الذي تم سلكه في الشجرة للوصول للعقدة الورقة التي احتوت القرار النهائي.

4- يمكنها معالجة مشكل الارتباط الخطّي بين المعاملات بشكل جيد.

5- تعد من الخوارزميات الأسرع تنفيذًا على الحاسب نظرًا لبساطتها، كما أنها سهلة الفهم.

6- لا تحتاج إلى حجم كبير جدًا من معطيات الدخل.

السيئات:

1- قد نصل إلى حالة تسمى (overfitting)، أي أن نموذج شجرة القرار أصبح مرتبطة إلى حد كبير بمعطيات الدخل مما سيؤدي إلى انخفاض دقة قراراته بالنسبة للمعطيات الجديدة.

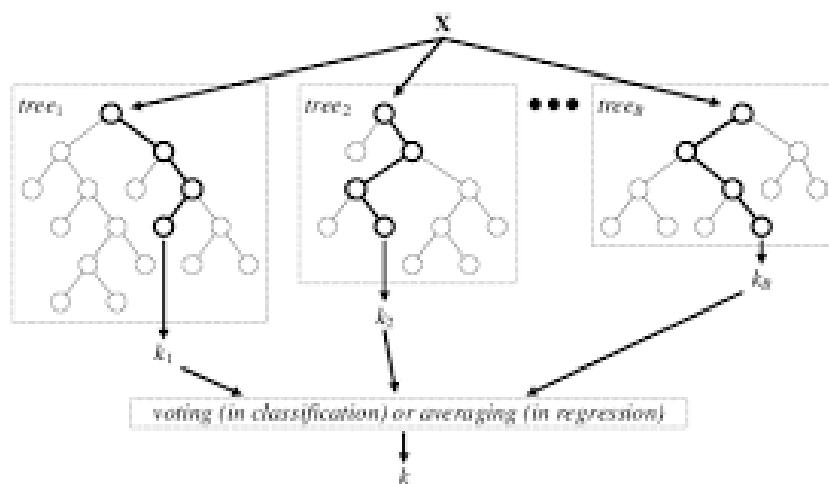
2- قد تؤدي القيم الشاذة (outliers) إلى التأثير بشكل كبير على دقة النموذج.

3- قد يتم خسارة كمية كبيرة من المعلومات التي تحملها المعاملات ذات القيمة المستمرة.

3.2.1 الغابة العشوائية (Random Forest "RF")

إنَّ الفكرة الأساسية في نموذج الغابة العشوائية هو محاولة استخدام أكثر من شجرة قرار واحدة. حيث يتم تشكيل مجموعة من أشجار القرار، لكن يتم تشكيل هذه الأشجار بشكل عشوائي، حيث يتم لكل شجرة اختيار مجموعة جزئية من المعاملات بشكل عشوائي لتدخل في بنائها، كما يتم اختيار ترتيب هذه المعاملات بشكل عشوائي أيضاً. وبهذا نحصل على مجموعة من الأشجار "غابة" المبنية بشكل عشوائي، ومن هنا أتت تسمية "الغابة العشوائية". بعد تشكيل النموذج تتم عملية التنبؤ عن طريق الحصول على تنبؤات كل من الأشجار، ويتم اعتماد النتيجة النهائية بحسب النتيجة الأكثر تكراراً من بين النتائج التي قدّمتها الأشجار.

إنَّ بناء الأشجار بشكل عشوائي عوضاً عن الطريقة المتتبعة في نموذج شجرة القرار يفيد في تلافي مشكلة (overfitting) والتي قد تعاني منها شجرة القرار.



الشكل (5) مثال عن خوارزمية Random Forest

الحسنات:

- 1- تتمتع الغابة العشوائية بكافة مزايا وخصائص شجرة القرار.
- 2- من أهم ميزات هذه الخوارزمية التي تميزها عن شجرة القرار هي أنها لا تعاني من مشكلة الـ (overfitting)، حيث أنه في حال عانت إحدى أشجار القرار من هذه المشكلة، فإنَّ بقية الأشجار ستقلل من أثر هذه المشكلة بما أن القرار يتم اتخاذه بشكل وسطي بين جميع القرارات.

السيئات:

- 1- تعد الخوارزمية معقدة نسبياً وصعبة الفهم.
- 2- تحتاج وقتاً أطول من شجرة القرار لإعطاء النتائج.

4.2.1 أقرب k جار (K Nearest Neighbors "KNN")

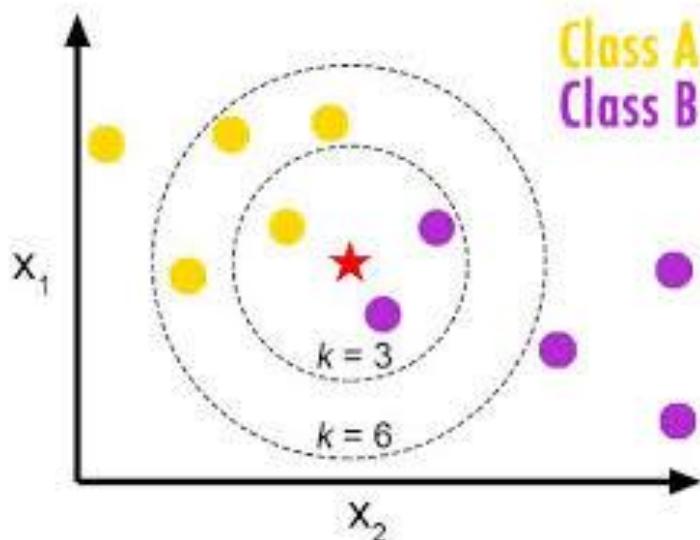
KNN هي خوارزمية بسيطة تخزن جميع الحالات المتاحة لديها من بيانات التدريب وتصنف الحالات الجديدة حسب أغلبية جيرانها، وذلك عن طريق وظيفة حسابية لقياس المسافة بينهم. هذه الوظائف يمكن أن تكون المسافة – Manhattan Euclidean، – مانهاتن، – Minkowski Hamming.

إذا كانت $1 = K$ ، فسيتم تعين الحالة ببساطة إلى أقرب فئة مجاورة لها.

يتم التنبؤ بالفئة الخاصة بنقطة بيانات جديدة من خلال البحث في مجموعة التدريب بأكملها عن الفئات الأكثر قرباً واختيار أقرب k فئة، ومن ثم يتم اعتماد الجواب النهائي بالأعتماد على "رأي" الأغلبية، ففي حال كان عدد النقاط المختارة والتي تنتهي إلى الفئة الأولى أكبر من نظيره في الفئة الثانية، عندها نعتبر النقطة الجديدة تنتهي للفئة الأولى، والعكس بالعكس.

الحيلة تكمن في كيفية تحديد عوامل التشابه بين البيانات. إن أبسط طريقة لمعرفة مدى التشابه -إذا كانت البيانات كلها بنفس المقياس (جميعها بالبوصة على سبيل المثال)- هي استخدام المسافة الإقليدية.

يستهلك تطبيق خوارزمية KNN الكثير من الذاكرة لتخزين جميع البيانات، ولكنه لا يؤدي أي عمليات حسابية إلا عند الحاجة للتنبؤ. يمكنك أيضاً تحديث حالات التدريب بمراور الوقت للحفاظ على دقة التنبؤات.



الشكل (6) مثال عن خوارزمية K Nearest Neighbors

الحسنات:

- 1- من أهم ميزات هذه الخوارزمية هي عدم الحاجة للقيام بعمليات حسابية أثناء عملية التدريب، حيث أن كافة الحسابات تحصل عند القيام بعملية التنبؤ، وبالتالي فإن فترة التدريب تكون قصيرة.
- 2- تعدّ من أسهل خوارزميات تعلم الآلة وأكثرها بساطة.
- 3- تعدّ جيدة نسبياً من أجل المعطيات التي تحتوي أخطاء.

السيئات:

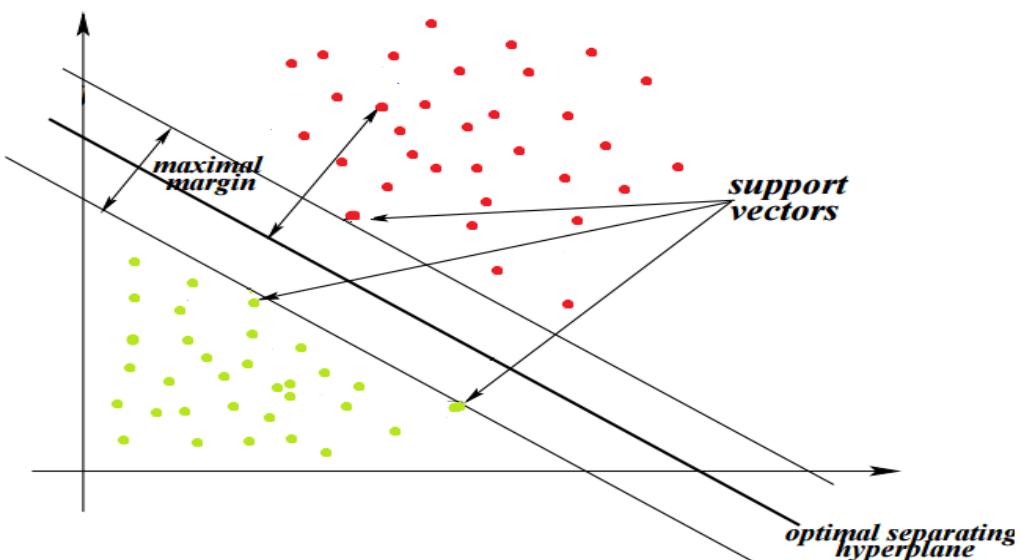
- 1- تعاني من مشكلة الانزياح نحو القيم المحلية (local approximation).
- 2- تحتاج إلى كمية كبيرة من الحسابات أثناء عملية التنبؤ مقارنةً مع بقية الخوارزميات، وبالتالي لا تصلح من أجل التطبيقات التي تحتاج إلى زمن تنبؤ صغير (معالجة).
- 3- اختيار قيم مناسبة للعدد k قد يكون صعباً وغير بديهياً، وهذا الاختيار يكون له تأثير جوهري على الحلول التي يطرحها النموذج في الكثير من الأحوال، ولذلك يجب اختيار هذه القيمة بعناية.
- 4- من الضروري في الكثير من الحالات القيام بتسوية قيم المعاملات للحصول على النتائج المرجوة.
- 5- يتراجع أداؤها نسبياً مع زيادة عدد معاملات الدخل.

5.2.1 متجه الدّعم (Support Vector "SV"):

تعد SVM واحدة من أكثر الخوارزميات شهرة على الإطلاق. تهدف خوارزمية SVM إلى إيجاد نطاق في مساحة N الأبعاد حيث N هو عدد الصفات عن طريق إنشاء نطاق فاصل "يتوسطه خط" يقسم بين البيانات حسب فئتها، كمثال، إما الفئة 0 أو الفئة 1.

تسمى المسافة بين النطاق وأقرب نقاط البيانات بالهامش. ودائماً حصول النطاق على هامش أكبر يجعل منه نطاقاً أفضل. تعرف النقاط التي يتم الاعتماد عليها في تعريف هذا النطاق الفاصل وبناء المصنف بالـ Support Vectors.

يوضح الشكل التالي، حالة بسيطة لمتجه دعم يقوم بالفصل بين مجموعتين من النقاط.



الشكل (7) مثال عن خوارزمية Support Vector

في حال كانت العلاقة غير خطية بين معطيات الدخل والمتحوّل الهدف، تقوم الخوارزمية باستخدام ما يُعرف

بـ (kernel function)، حيث يتم اختيار الحد الفاصل ليكون ذا N بعد "مثلاً" في حال كانت الارتباط من المرتبة الثانية سيكون متجه الدّعم هو تابع من المرتبة الثانية، لن نقوم بشرح طبيعة هذه العملية كونها خارج نطاق اهتمامنا.

الحسنات:

- 1- لا تعاني هذه الخوارزمية من مشكلة الانزياح نحو القيم المحلية.
- 2- بإمكانها معالجة الكثير من أنماط الارتباط دون مشاكل.
- 3- يمكنها التعامل مع القيم الشاذة بشكل جيد.
- 4- يمكنها إعطاء نتائج جيدة من أجل حجم صغير من المعطيات.
- 5- يمكنها التعامل بشكل جيد مع المعطيات ذات العدد الكبير من المعاملات.

السيئات:

- 1- عملية التدريب تحتاج الكثير من الوقت مقارنةً مع باقي الخوارزميات.
- 2- توجد العديد من المعاملات التي تتحكم بعمل الخوارزمية والتي يصعب معرفة القيم المناسبة لها لتعطي الأداء المثالي.
- 3- لا يمكنها اعطاء احتمال لصحة التوقع.
- 4- تعاملها مع المعطيات التصنيفية ليس جيداً.
- 5- التصنيف لعدة أصناف يحتاج إلى استخدام عدّة نماذج.

6.2.1 بايز مع التوزع الغاوسي (GNB):

إنها تقنية تصنيف تستند إلى نظرية Bayes Naive Bayes يفترض أن جميع المعاملات مستقلة احتمالياً مثنى، وأنها ذات توزع غاوسي. من السهل بناء نموذج Gaussian، ويعتبر مفيد بشكل خاص في التعامل مع التعلم مع مجموعات البيانات الكبيرة جداً. جنباً إلى جنب مع بساطته، فإنه يتفوق على أساليب التصنيف المتطرفة للغاية.

تعتمد هذه الخوارزمية بشكل أساس على قانون بايز في الاحتمالات:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

حيث A, B هي أحداث مستقلة احتمالياً

$P(B)$: احتمال حدوث الحدث B

$P(A)$: احتمال حدوث الحدث A

$P(B|A)$: احتمال حدوث الحدث B مع العلم أن الحدث A قد وقع

$P(A|B)$: احتمال حدوث الحدث A مع العلم أن الحدث B قد وقع

وبالتالي لحساب احتمال كون الخرج هو Y علمًا أن الدخل هو X تُستخدم العلاقة التالية:

$$P(C_i|x_1, x_2, \dots, x_n) = \frac{P(x_1, x_2, \dots, x_n|C_i) \cdot P(C_i)}{P(x_1, x_2, \dots, x_n)} \text{ for } 1 \leq i \leq k$$

وبما أننا نعتبر أن المعاملات مستقلة فيمكن تبسيط العلاقة إلى الشكل التالي:

$$P(C_i|x_1, x_2, \dots, x_n) = \left(\prod_{j=1}^{j=n} P(x_j|C_i) \right) \cdot \frac{P(C_i)}{P(x_1, x_2, \dots, x_n)} \text{ for } 1 \leq i \leq k$$

لحساب هذه الاحتمالات يتم اعتماد مبدأ يشابه ذلك في خوارزمية Logistic Regression، حيث أنه بعد كل عملية تدريب، يتم تعديل قيم الاحتمالات بحيث نقل الخطأ، وفي حال كانت المعاملات مستقلة عن بعضها، ستنتظر قيم المتغيرات بعد عدد معين من عمليات التدريب، وكلما زاد الارتباط بين المتغيرات، كلما قل استقرار النموذج.

الحسنات:

- 1- يقدم هذا النموذج نتائج جيدة حتى مع حجم صغير من معطيات الدخل.
- 2- في حال كانت المعاملات مستقلة احتمالياً، فإن هذه الخوارزمية تقارب إلى الحل أسرع من أي خوارزمية أخرى.
- 3- لا تواجه هذه الخوارزمية صعوبات في حال وجود معطيات ليست لها علاقة بالمتغير الهدف.
- 4- يعد هذا النموذج من النماذج البسيطة وسهلة الفهم، كما أن الزمان اللازم لإجراء عمليات التدريب والتنبؤ قصير نسبياً.
- 5- يمكن استخدامها لتصنيف المعطيات إلى فئتين أو أكثر.

السيئات:

تفرض هذه الخوارزمية استقلال معاملات الدخل، وهذا الفرض يكون غير صحيح في معظم التطبيقات الحياتية، مما قد يؤدي إلى الوصول إلى نتائج خاطئة.

7.2.1 العنقة بـ k قيمة متوسطة (K means Clustering):

K means هي الخوارزمية غير الخاضعة للإشراف لحل مشكلة التجميع. من خلال طريقة بسيطة يتم توزيع وتصنيف بيانات المدخلات على عدد معين (k) من مجموعات البيانات. تتميز نقاط البيانات داخل المجموعة الواحدة بالتجانس وبعضها البعض والاختلاف من مجموعات البيانات الأخرى.

تعمل هذه الخوارزمية وفق الخطوات التالية:

1. تختار الخوارزمية عشوائياً k نقطة، وهي النقاط التي ستصبح مركز كل مجموعة تعرف باسم centroid.
2. تتجه كل نقطة من بيانات إلى أقرب مجموعة لها، حسب كل نقطة مركزية للمجموعة بالطبع.

3. إيجاد النقطة المركزية لكل مجموعة يتم باعتبار بيانات المجموعة كلها. ومع تغير نقاط البيانات للمجموعة تتغير النقطة المركزية بالطبع.

4. نظراً لأن لدينا نقاط مركزية جديدة، سنكرر الخطوتين 2 و 3 حتى تستقر نقاط البيانات في مجموعاتها.

كيف نحدّد قيمة k :

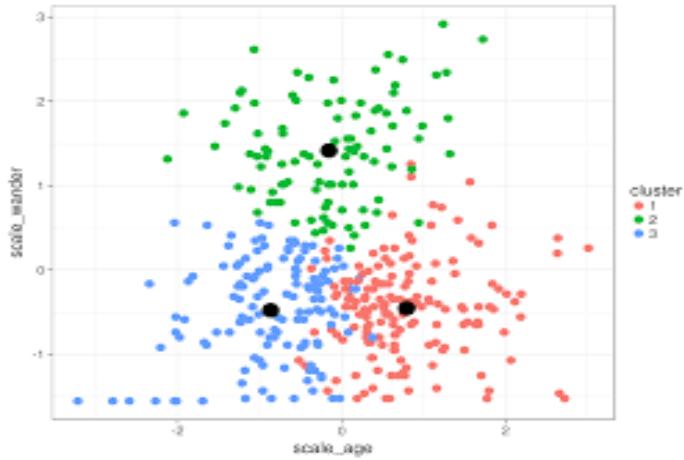
هناك طريقة شهيرة تُعرف باسم elbow method تُستخدم لتحديد القيمة المثلثي ل k ، الفكرة الأساسية وراء هذه الطريقة هي رسم القيم المختلفة للتكلفة مع تغيير k .

The Elbow Method



الشكل (8) مثال عن الـ Elbow Method

Wcss (Within Cluster Sum of Squares): هو مقدار يعبر عن كثافة العنقود، ويتم حسابه عن طريق حساب مجموع أبعاد كل من نقاط العنقود عن مركزه (يتم حساب الأبعاد باستخدام النظيم الثاني، أي بأخذ جذر مجموع مربعات الفروق بين قيم كل من المعاملات).



الشكل (9) مثال عن خوارزمية *K Means Clustering*

الحسنات:

- 1- هذا النموذج يعد سهلاً وبسيطاً كما أنه سريع التنفيذ وذا تعقيد منخفض.
- 2- يتمتع هذا النموذج بالمرنة فبإمكانه التلائم مع تغير المعطيات بسرعة.
- 3- مناسب من أجل قواعد المعطيات الكبيرة.

السيئات:

- 1- يتأثر النموذج هذا بترتيب المعطيات.
- 2- أي تغيير في طبيعة المعطيات "مثل عملية التسوية" قد يؤدي إلى الحصول على نتائج مختلفة كلّياً.
- 3- لا تستطيع التعامل مع المعطيات التصنيفية بشكل جيد.
- 4- من الصعب اختيار قيمة مناسبة للمعامل k بشكل مبدئي، بل قد تُضطر إلى تجرب العديد من القيم حتى نصل إلى القيمة المطلوبة.

وبعد التعرّف على أشهر خوارزميات التصنيف نوضح في الجدول (1) مقارنة بسيطة بينها.

الجدول 1- مقارنة نظرية بين خوارزميات تعلم الآلة

الحاجة لتسوية القيم	التعامل مع أنواع الارتباطات المتنوعة	التعامل مع المعطيات التصنيفية	دقة النتائج لأجل حجم معطيات تدريب صغير	السهولة	سرعة التثبي	سرعة التدريب	متوسط الدقة	الخوارزمية
ليس بالضرورة، قد تؤدي لتحسين النتائج	سيء	سيء	جيد	سهلة	سريع	سريع	جيد	LR
لا حاجة	ممتاز	ممتاز	وسط	سهلة	سريع	سريع	جيد جداً	DT
لا حاجة	ممتاز	ممتاز	وسط	صعبة	سريع	سريع	ممتاز	RF
نعم	جيد	سيء	وسط	سهلة	بطيء جداً	سريع جداً		KNN
ليس بالضرورة، قد تؤدي لتحسين النتائج	جيد جداً	سيء	جيد	صعبة	بطيء	بطيء	جيد	SV
لا حاجة	جيد	سيء	وسط	سهلة	سريع	سريع	جيد	Gaussian Naïve Bayes
ليس بالضرورة، قد تؤدي لتحسين النتائج	جيد	سيء	جيد	سهلة	سريع	بطيء	جيد	K Means Clustering

حصلنا على المعلومات الخاصة بالدراسة النظرية بما في ذلك جدول المقارنة هذا من موقع Medium .“medium.com”

الفصل الثاني

التنفيذ والاختبارات والنتائج

نستعرض في هذا الفصل شرحاً عن بيئة العمل المستخدمة في تنفيذ النظام مع توضيح الأدوات والمكتبات المستخدمة. كما نستعرض خطوات العمل التي اتبناها للوصول إلى النتائج النهائية، ثم سنستعرض هذه النتائج.

1.2 بيئة العمل:

استخدمنا في تنفيذ هذا النظام لغة Python 3.7.3 على أداة Jupyter Notebook

1.1.2 لغة Python

هي لغة برمجة، من اللغات عالية المستوى، تتميز ببساطة كتابتها وقراءتها، سهلة التعلم، تستخدم أسلوب البرمجة الكائنية Object Oriented Programming ، مفتوحة المصدر، وقابلة للتطوير. تعتبر لغة بايثون لغة تفسيرية أي أنها تحتاج أن تُفسَّر باستخدام برنامج يدعى المفسر لتنفيذ البرامج المكتوبة بها، وتستخدم بشكل واسع في العديد من المجالات، كبناء البرامج المستقلة باستخدام الواجهات الرسومية المعروفة وفي عمل برامج الويب، بالإضافة إلى استخدامها كلغة برمجة نصية للتحكم في أداء بعض من أشهر البرامج المعروفة أو في بناء برامج ملحقة لها. وبشكل عام يمكن استخدام بايثون لبرمجة البرامج البسيطة للمبتدئين، وإنجاز المشاريع الضخمة كأي لغة برمجية أخرى. غالباً ما يُنصح المبتدئون في ميدان البرمجة بتعلم هذه اللغة لأنها من بين أسرع اللغات البرمجية تعلمًا.

نشأت بايثون في مركز CWI (مركز العلوم والحاسب الآلي) بأمستردام على يد جايدو فان روسم في أواخر الثمانينيات من القرن المنصرم، وكان أول إعلان عنها في عام 1991 . تم كتابة نواة اللغة بلغة سي. أطلق فان روسم الاسم "بايثون" على لغته تعبيراً عن إعجابه بفرقة مسرحية هزلية شهيرة من بريطانيا، كانت تطلق على نفسها الاسم موتنى بايثون.

تتميز بايثون بمجتمعها النشط، كما أن لها الكثير من المكتبات البرمجية ذات الأغراض الخاصة والتي برمجها أشخاص من مجتمع هذه اللغة، مثلًا مكتبة sklearn التي يمكن باستخدامها العمل مع خوارزميات تعلم الآلة Machine Learning بشكل سهل وسلس ببضعة أسطر من الرمaz. ويمكن لبايثون التعامل مع العديد من أنواع قواعد البيانات مثل MySQL وغيره، كما يمكنها التعامل مع الكثير من أنواع الملفات مثل ملفات excel و .csv (comma separated values)

2.1.2 أداة Jupyter Notebook

Jupyter Notebooks، أو ما تسمى ب IPython Notebooks هي أداة ويب تقوم بإنشاء ملفات Jupyter Notebook Documents والتي هي ملفات JSON، لاحقة هذه الملفات

هي "ipynb" ، البنية العامة لملف هي مجموعة من الخلايا (cells) التي تحتوي رماز ، دخل وخرج، نصوص أو مخططات وصور وغيرها.

يمكن تحويل هذا النمط من الملفات إلى العديد من الصيغ الأخرى مثل ملفات Python، nbconvert HTML, LaTeX, PDF, Presentation Slides استخدامها للعديد من لغات البرمجة وليس فقط .python.

ما يميز هذه الأداة هي القدرة على بناء برامج تفاعلية بشكل سهل وسلس، وهي ممتازة بشكل خاص من أجل مشاريع ال Data Science & Machine Learning كونها توفر تفاعليّة عاليّة مما يسهل اختبار النتائج والتعديل بشكل أكثر سلاسة.

2.2 المكتبات البرمجية المستخدمة

قمنا بتنصيب مكتبات لغة Python التي استخدمناها في مشروعنا هذا بمساعدة من مقالات أحد العاملين في هذا المجال [2].

مكتبة Pandas: وهي مكتبة مجانية تُستخدم لبناء بنى المعطيات والتعديل عليها، فهي تسمح بالقيام بالعديد من الوظائف كالبحث، حذف وإضافة أسطر وأعمدة، وتعديل القيم داخل الخلايا. وهي المكتبة التي استخدمناها عند دمج بنى المعطيات الأربع في بنية واحدة ومن ثم فصلها لبنيتي تدريب واختبار بشكل عشوائي "تستخدم هذه المكتبة مكتبة مُساعدة تسمى numpy.

مكتبة Seaborn: وهي مكتبة مجانية تُستخدم لرسم الكثير من أنواع المخططات مع منح المستخدم عدداً كبيراً من الخيارات والمواصفات التي بإمكانه التحكم بها، وهي المكتبة التي استخدمناها لرسم المخططات التي توضح النتائج التي توصلنا إليها "تستخدم هذه المكتبة مكتبة مُساعدة تسمى "plotly".

مكتبة Sklearn: وهي مكتبة مجانية لخوارزميات تعليم الآلة للبرمجة بلغة الباليثون تقدم خوارزميات التصنيف

والعنقدة. تم استخدامها لتطبيق خوارزميات التصنيف Logistic Regression, Decision Tree, Random Forest, K Nearest Neighbors, Support Vector, Naïve Bayes, K Means Clustering لبناء النماذج التي ساعدت على فرز الطرود الشبكية ما بين فئة الطرود الطبيعية وفئة الطرود غير الطبيعية.

تم استخدام الإصدار 0.20.3 SKlearn في تنفيذ النظام.

مكتبة Cufflinks: وهي مكتبة مجانية تشبه في عملها مكتبة Seaborn لكن تتميز المخططات فيها بكونها تفاعلية.

مكتبة Widgets: وهي مكتبة مجانية تسمح بإنشاء واجهة تخطب مع المستخدم، مثل الأزرار وغيرها.

مكتبة tkinter: وهي مكتبة تسمح بانتقاء الملف المطلوب بشكل ديناميكي أثناء عمل البرنامج، وقد استخدمناها للسماح للمستخدم باختيار الملفات التي تمثل قواعد معطيات التدريب والاختبار.

مكتبة OS: وهي مكتبة تسمح بإنشاء مجلدات ضمن البرنامج، استخدمناها لإنشاء المجلدات التي سنحفظ ضمنها ملفات المخططات.

3.2 النتائج العملية

نهدف في هذا المشروع ليس فقط إلى استخدام إحدى خوارزميات تعلم الآلة للحصول على نتائج تصنيفية، بل نهدف أيضاً إلى مقارنة أداء كل من هذه الخوارزميات من ناحية الدقة والזמן اللازم و عوامل أخرى وذلك لمعرفة أكثر هذه الخوارزميات ملائمةً للمسألة المطروحة، إضافةً إلى للسماح للمستخدم بأن يختار ما يناسبه منها بحسن متطلباته ورغباته.

1.3.2 قواعد المعطيات المستخدمة:

حصلنا على قواعد المعطيات المستخدمة في مشروعنا هذا من المهندس المشرف إضافة إلى حصلنا على قواعد معطيات أخرى من موقع Kaggle التعليمي [4][3]. يوجد في قواعد المعطيات التي لدينا 42 حقلاً بما فيهم الحقل المعتبر عن المت Howell الهدف (Normal/Anomaly)، والذي نهدف إلى تصنیف الطّرود على أساسه، وفيما يلي شرح مختصر لدلاله كل من الحقول الموجودة:

- Duration: مدة الاتصال.
- Protocol Type: البروتوكول المستخدم في الاتصال.
- Service: الخدمة المستخدمة في الشبكة الهدف.
- Flag: حالة الاتصال (طبيعي، وجود خطأ).
- Src_bytes: عدد البيانات المنقولة من المصدر إلى الهدف خلال اتصال واحد.
- Dst_bytes: عدد البيانات المنقولة من الهدف إلى المصدر خلال اتصال واحد.
- Land: يأخذ هذا الحقل القيمة 1 في حال تطابق عنوان (IP) و رقم الـ (Port) لكل من المصدر والهدف، وإلا يأخذ القيمة 0.
- Wrong_fragment: عدد الأجزاء (Fragments) من الاتصال التي احتوت خطأ.
- Urgent: عدد الطّرود الطارئة (التي كانت خانة "urgent" فيها تحمل القيمة 1) في الاتصال.
- Hot: عدد المؤشرات "الساخنة" في الاتصال، والتي تعبر عن محاولات الوصول إلى محتويات الجهاز، إنشاء برامج أو تشغيل برامج.
- Num_failed_logins: عدد المحاولات الفاشلة لتسجيل الدخول.
- Logged_in: يأخذ هذا الحقل القيمة 1 في حال نجاح تسجيل الدخول، وإلا يأخذ القيمة 0.
- Num_compromised: عدد الشروط في حالة الخطر.
- Root_shell: يأخذ هذا الحقل القيمة 1 في حال امتلاك الاتصال صلاحيات المدير، وإلا يأخذ القيمة 0.
- Su_attempted: يأخذ هذا الحقل القيمة 1 في حال محاولة استخدام التّعلیمة "su" (تغيير المستخدم)، وإلا يأخذ القيمة 0.

- عدد العمليات التي نُفِّذت في نمط المدير (root). Num_root
- عدد الملفات التي تم إنشاؤها خلال الاتصال. Num_file_creations
- عدد "shell prompts" التي تم استخدامها في الاتصال. Num_shells
- عدد عمليات الوصول إلى ملفات التي تمّت خلال الاتصال. Num_access_files
- عدد التعليمات الصادرة خلال اتصال من نوع "ftp". Num_obtained_cmds
- يأخذ هذا الحقل القيمة 1 في حال تسجيل الدخول كمدير، وإلا يأخذ القيمة 0. Is_hot_login
- يأخذ هذا الحقل القيمة 1 في حال تسجيل الدخول كضيف ، وإلا يأخذ القيمة 0. Is_guest_login
- عدد الاتصالات إلى نفس هدف الاتصال الحالي والتي تمّت خلال آخر ثانيتين. Count
- عدد الاتصالات إلى نفس "port number" الاتصال الحالي والتي تمّت خلال آخر ثانيتين. Srv_count
- عدد الاتصالات التي كانت فيها قيمة الحقل (4) هي 1 من بين الاتصالات التي تمّ احصائتها في الحقل (23). Serror_rate
- عدد الاتصالات التي كانت فيها قيمة الحقل (4) هي 1 من بين الاتصالات التي تمّ احصائتها في الحقل (24). Srv_serror_rate
- عدد الاتصالات التي كانت فيها قيمة الحقل (4) هي 1 من بين الاتصالات التي تمّ احصائتها في الحقل (23). Rerror_rate
- عدد الاتصالات التي كانت فيها قيمة الحقل (4) هي 0 من بين الاتصالات التي تمّ احصائتها في الحقل (24). Srv_rerror_rate
- نسبة عدد الاتصالات التي كانت لنفس الخدمة (service) من بين الاتصالات التي تمّ احصائتها في الحقل (23). Same_srv_rate
- نسبة عدد الاتصالات التي كانت لخدمات مختلفة من بين الاتصالات التي تمّ احصائتها في الحقل (23). Diff_srv_rate
- نسبة عدد الاتصالات التي كانت لأجهزة هدف مختلفة من بين الاتصالات التي تمّ احصائتها في الحقل (24). Srv_diff_host_rate
- عدد الاتصالات ذات نفس عنوان (IP) للهدف. Dst_host_count
- عدد الاتصالات ذات نفس رقم البوابة (Port number). Dst_host_srv_count
- عدد الاتصالات التي كانت لنفس الخدمة من بين الاتصالات التي تمّ احصاؤها في الحقل (32). Dst_host_same_srv_rate
- عدد الاتصالات التي كانت لنفس الخدمة من بين الاتصالات التي تمّ احصاؤها في الحقل (32). Dst_host_diff_srv_rate
- عدد الاتصالات التي كانت لنفس رقم البوابة (Port number) من بين الاتصالات التي تمّ احصاؤها في الحقل (33). Dst_host_same_src_rate

- Dst_host_srv_diff_host_rate: نسبة عدد الاتصالات التي كانت لأجهزة هدف مختلفة من بين الاتصالات التي تم احصاؤها في الحقل (33).
- Dst_host_serror_rate: نسبة عدد الاتصالات التي كانت قيمة الحقل (4) فيها هي 1 من بين الاتصالات التي تم احصاؤها في الحقل (32).
- Dst_host_srv_serror_rate: نسبة عدد الاتصالات التي كانت قيمة الحقل (4) فيها هي 1 من بين الاتصالات التي تم احصاؤها في الحقل (33).
- Dst_host_rerror_rate: نسبة عدد الاتصالات التي كانت قيمة الحقل (4) فيها هي 0 من بين الاتصالات التي تم احصاؤها في الحقل (32).
- Dst_host_srv_rerror_rate: نسبة عدد الاتصالات التي كانت قيمة الحقل (4) فيها هي 0 من بين الاتصالات التي تم احصاؤها في الحقل (33).

حصلنا على أربعة من قواعد المعطيات هذه من موقع (Kaggle) الذي يوفر بنى معطيات متنوعة في العديد من المجالات من أجل تطبيقات تعلم الآلة. وقد اخترنا أربع قواعد معطيات لكي نضمن وجود تنوع في القيم حيث أن كل قاعدة معطيات تم أخذها خلال فترة زمنية معينة من مصدر معين.

2.3.2 تهيئة قواعد المعطيات:

لكي تستطيع خوارزميات تعلم الآلة التعامل مع المعطيات بشكل سلس وفعال، لا بد من القيام بتهيئة هذه المعطيات، ما يعني تحويلها لصيغة يمكن لهذه الخوارزميات التعامل معها.

قمنا بدايةً بدمج قواعد المعطيات الأربع في قاعدة معطيات واحدة ثم فصلناها إلى قسمين بشكل عشوائي لتشكيل قاعدة معطيات للتدريب (تحتوي 70% من عدد الأسطر) وقاعدة معطيات للاختبار (تحتوي 30% من عدد الأسطر) وذلك باستخدام لغة python. وفي النهاية نتج لدينا الملفين ”Train.csv“ و ”Test.csv“ اللذان اعتمدنا عليهما في دراستنا العملية.

الآن سنقوم بحذف جميع الحقول ذات القيم الثابتة، وذلك لأنها لن تقدم لنا أيّ معلومات إضافية عن الطرد، في حين أنها قد تشكل عبئاً على بعض الخوارزميات حيث ستؤدي إلى الحاجة للمزيد من العمليات الحسابية.

بعد الانتهاء من ذلك نلاحظ وجود حقول ذات قيم غير رقمية وهي حقول تصنيفية (مثل حقل ”protocol_type“)، وبما أن بعض الخوارزميات المستخدمة لا تستطيع التعامل مع القيم غير العددية، سنقوم بالعملية التالية على كل من هذه الحقول، سنقوم بمقابلة كل من القيم التصنيفية بقيمة عدديّة ما بدءاً من الواحد وانتهاءً بعدد القيم المختلفة لهذا الحقل، ومن ثم سنقوم بإنشاء حقل جديد يكافئ الحقل الأصلي لكن مع استبدال القيم الموجودة بمقابلاتها العددية، وفي النهاية نقوم بحذف الحقل الأصلي.

كما لاحظنا في الدراسة النظرية فإن عملية تسوية قيم المعطيات لها دور جوهري في أداء بعض الخوارزميات المستخدمة، ولذلك سنقوم أيضاً بإنشاء نسخة من قواعد المعطيات التي لدينا بعد اجراء عملية التسوية على القيم.

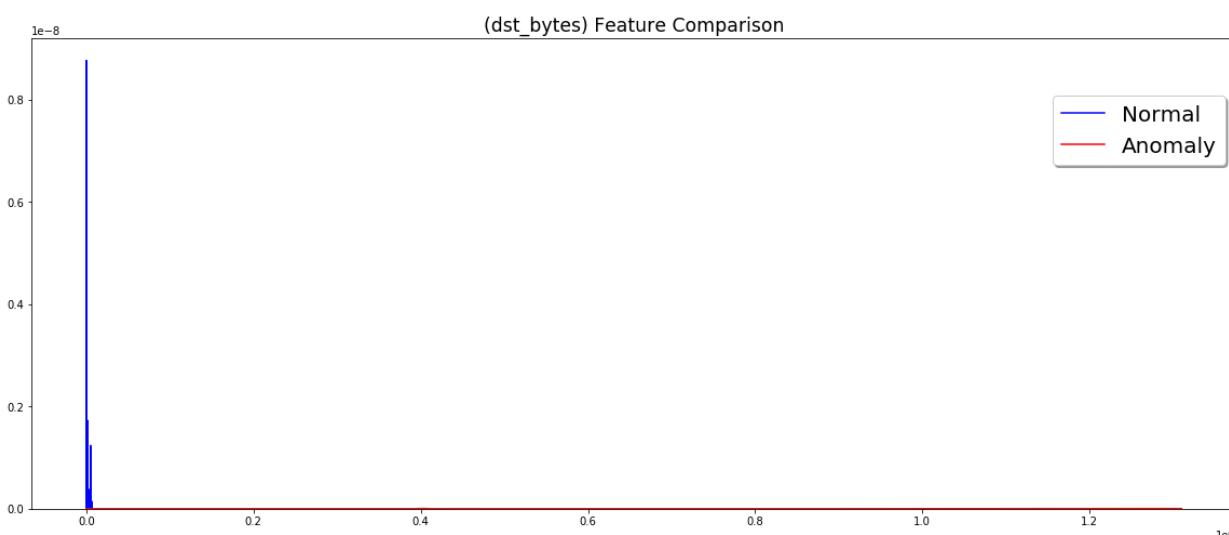
تم جميع العمليات السابقة ضمن البرنامج نفسه باستخدام مكتبات من لغة Python ولا يحصل أي تعديل على ملف بنية المعطيات نفسه.
بهذا تكون قد انتهت عملية التهيئة للمعطيات التي لدينا.

3.3.2 تقسيم الحقول إلى مجموعات:

لاحظنا خلال دراستنا النظرية أنَّ طبيعة المعطيات وال العلاقات التي تربطها بعضها لها تأثيرات مختلفة على كل من خوارزميات التصنيف المستخدمة، ولهذا السبب سنقوم بتقسيم الحقول الموجودة إلى ثمانية مجموعات "أربع مجموعات من أجل الحقول ذات القيم الأصلية وأربعة من أجل الحقول المقابلة لها والتي تحمل القيم بعد عملية التسوية"، وذلك اعتماداً على مخططات تقوم بعملية ربط بين قيم الحقل وقيم المتحول الهدف المقابلة لها، لكل من الحقول مخطط مقارنة خاص به، يعبر المنحني الأزرق فيها عن التوزع لقيم الحقل من أجل الطرود ذات التصنيف (Normal) بينما يعبر المنحني الأحمر عن التوزع لقيم الحقل من أجل الطرود ذات التصنيف (Anomaly)، نلاحظ أنه كلما زاد الاختلاف بين هذين المنحنين، كلما كانت قيم الحقل لها تأثير أكبر على قيم المتحول الهدف.

المجموعات:

Very Different Features: وهي مجموعة الحقول التي يختلف فيها توزُّع قيم الطرود ذات التصنيف (Normal) كثيراً عن مقابلاتها من الطرود ذات التصنيف (Anomaly)، مما يعني أنَّ قيم هذه الحقول لها تأثير كبير نسبياً على قيم المتحول “Very Different Features (Scaled) والتي تحوي القيم بعد عملية التسوية). وهذا مثال عن أحد المخططات لأحد الحقول التي تتنمي لهذه المجموعة:

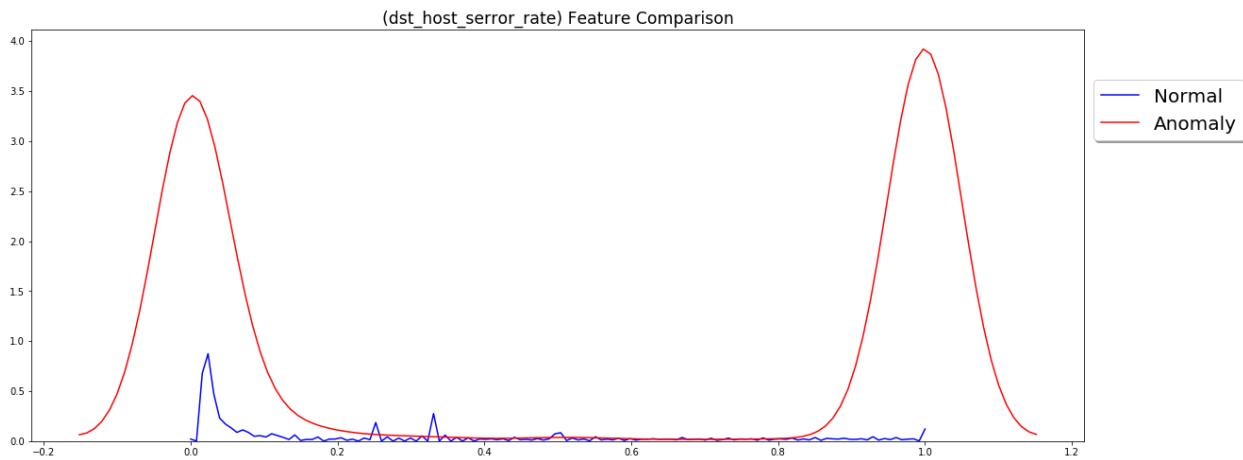


الشكل (10) مخطط لقيم أحد المعاملات من مجموعة المعطيات ”Very Different Features”

نلاحظ في هذا المخطط أنَّ المنحنى الأزرق مختلف كلياً عن المنحنى الأحمر.
تحتوي هذه المجموعة أربعة من الحقول، وهي:

- Dst_bytes •
- Num_compromised •
- Num_root •
- Src_bytes •

: وهي مجموعة الحقول التي يختلف فيها توزُّع قيم الطرود ذات التصنيف (Normal) بشكل واضح عن مقابلاتها من الطرود ذات التصنيف (Anomaly)، مما يعني أنَّ قيم هذه الحقول لها تأثير كبير نوعاً ماً على قيم المتحول الخرج (إضافةً للمجموعة المقابلة لها ”Different Features (Scaled)“ والتي تحوي القيم بعد عملية التسوية). وهذا مثال عن أحد المخططات لأحد الحقول التي تنتهي لهذه المجموعة:



الشكل (11) مخطط لقيم أحد المعاملات من مجموعة المعيديات ”Different Features“

نلاحظ في هذا المخطط أنَّ المنحنى الأزرق مختلف بشكل واضح عن المنحنى الأحمر.

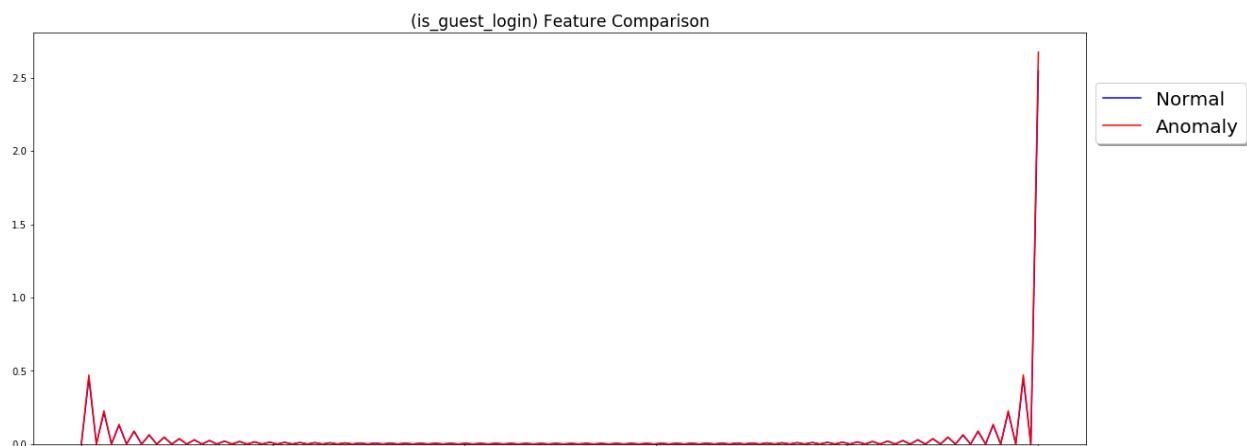
تحتوي هذه المجموعة تسعة من الحقول، ألا وهي:

- dst_host_serror_rate •
- Serror_rate, srv_serror_rate •
- dst_host_srv_count •
- dst_host_srv_serror_rate •

إضافةً إلى الحقول الأربع الموجودة في المجموعة السابقة.

: وهي مجموعة الحقول التي يُشابه فيها توزُّع قيم الطرود ذات التصنيف (Normal) بشكل كبير مقابلاتها من الطرود ذات التصنيف

(Anomaly)، مما يعني أنَّ قيم هذه الحقول لها تأثير صغير نسبياً على قيم المتحول الخرج (إضافةً للمجموعة المقابلة لها ”Very Similar Features (Scaled)“ والتي تحوي القيم بعد عملية التسوية). وهذا مثال عن أحد المخططات لأحد الحقول التي تنتمي لهذه المجموعة:



الشكل (12) مخطط لقيمة أحد المعاملات من مجموعة المعطيات ”Very Similar Features“

نلاحظ في هذا المخطط أنَّ المنحنى الأزرق يكاد يطابق المنحنى الأحمر. تحتوي هذه المجموعة خمسة من الحقول ألا وهي:

- is_guest_login
- is_host_login
- land
- root_shell
- wrong_fragment

يجب التنويه إلى أننا لن نستخدم في تطبيقنا هذه المجموعة من الحقول كونها لا تحوي كمية كبيرة من المعلومات عن المتحول الهدف، بل سنقوم باستخدام المجموعات المتممة لها والتي تحتوي 36 حقولاً، أي التي تحوي جميع الحقول ما عدا حقول هذه المجموعة.
وسنسمّي هذه المجموعة ”Not Very Similar Features“.

All Features: وهي المجموعة التي تحتوي جميع الحقول في قاعدة المعطيات، بما فيهم الحقول الموجودة في المجموعات الأخرى، وبالتالي هي التي تحمل أكبر كمية من المعلومات.

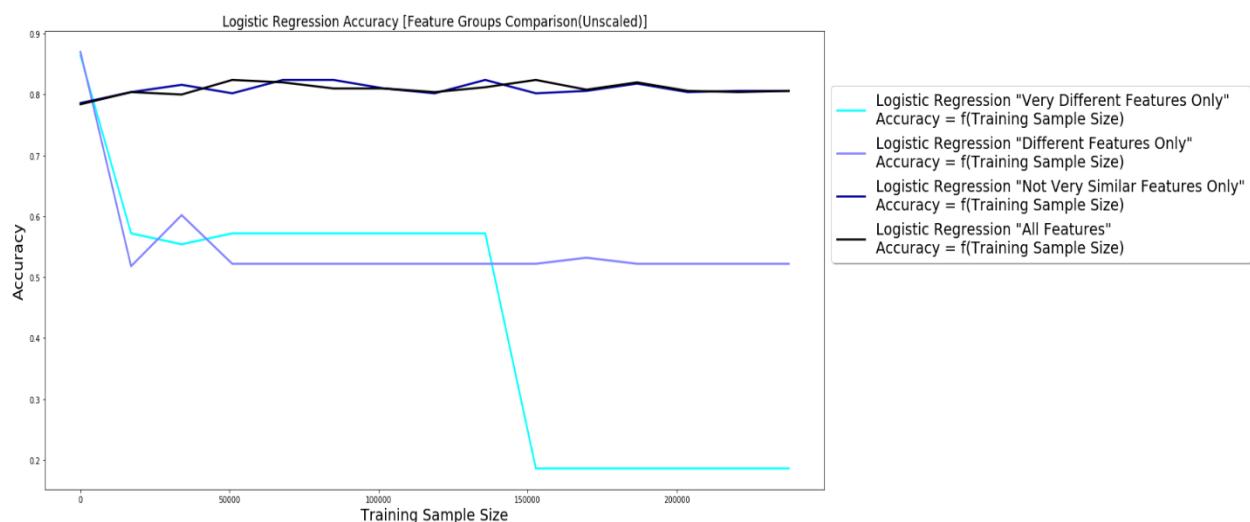
4.3.2 نتائج تطبيق خوارزميات تعلم الآلة:
الآن وبعد أن انتهينا من تهيئه المعطيات وتقسيم الحقول إلى مجموعات، أصبح بإمكاننا البدء بتطبيق خوارزميات تعلم الآلة.

سنقوم بدراسة تحليلية لكل من الخوارزميات المستخدمة ومقارنة نتائج تطبيقها على كل من مجموعات المعطيات التي سبق وذكرناها، حيث سنقوم باستخدام كل من هذه الخوارزميات لتشكيل 15 نموذج عند حجوم متزايدة خطياً لمعطيات التدريب "مع ثبيت حجم معطيات الاختبار" و 15 نموذج عند حجوم متزايدة خطياً لمعطيات التنبؤ "مع ثبيت حجم معطيات التدريب"، ومن ثم مقارنة هذه النتائج مع الدراسة النظرية واستقراء معلومات حول طبيعة المعطيات التي لدينا، وذلك من خلال ملاحظة أثر تغير حجم معطيات التدريب، حجم معطيات الاختبار، و مجموعة الحقول المستخدمة على دقة الخوارزمية وزمن التدريب والتنبؤ لها. كما نقوم بمحاولة معرفة الحد من حجم قيم التدريب الذي يصل عنده النموذج المدروس إلى حالته المستقرة، أي عندما يصبح تطبيق المزيد من عمليات التدريب ذا أثر طفيف على النموذج، ونقول عندها أن النموذج وصل لحالة الاستقرار ولا حاجة للقيام بالمزيد من عمليات التدريب.

1.4.3.2 الانحدار اللوجستي (Logistic Regression)

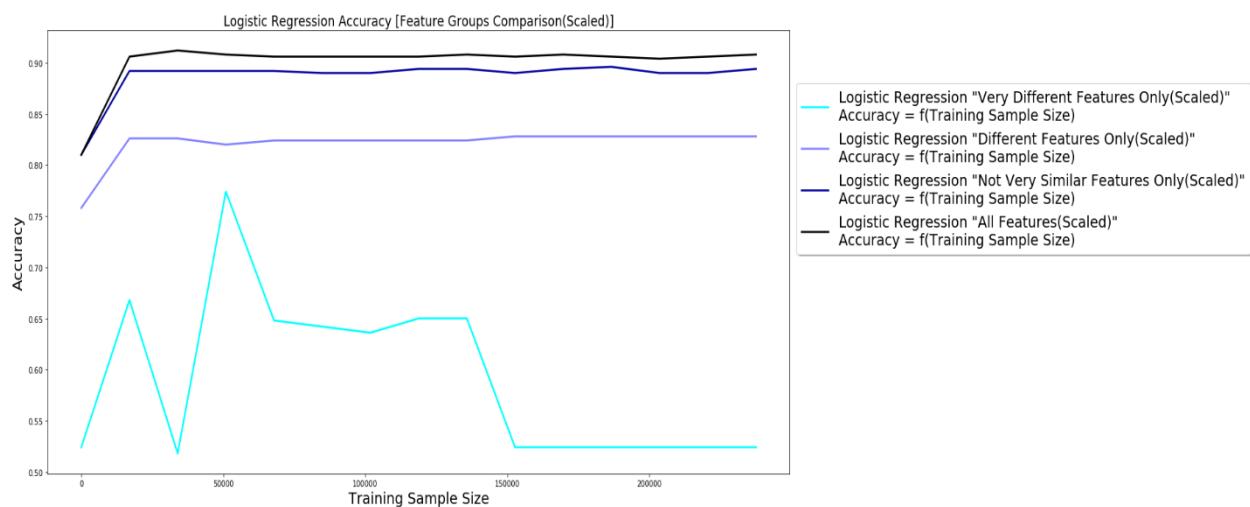
الدقة:

يوضح الشكل (13) مقارنة أداء خوارزمية الانحدار اللوجستي عند استخدام كل من مجموعات المعطيات قبل عملية تسوية القيم.



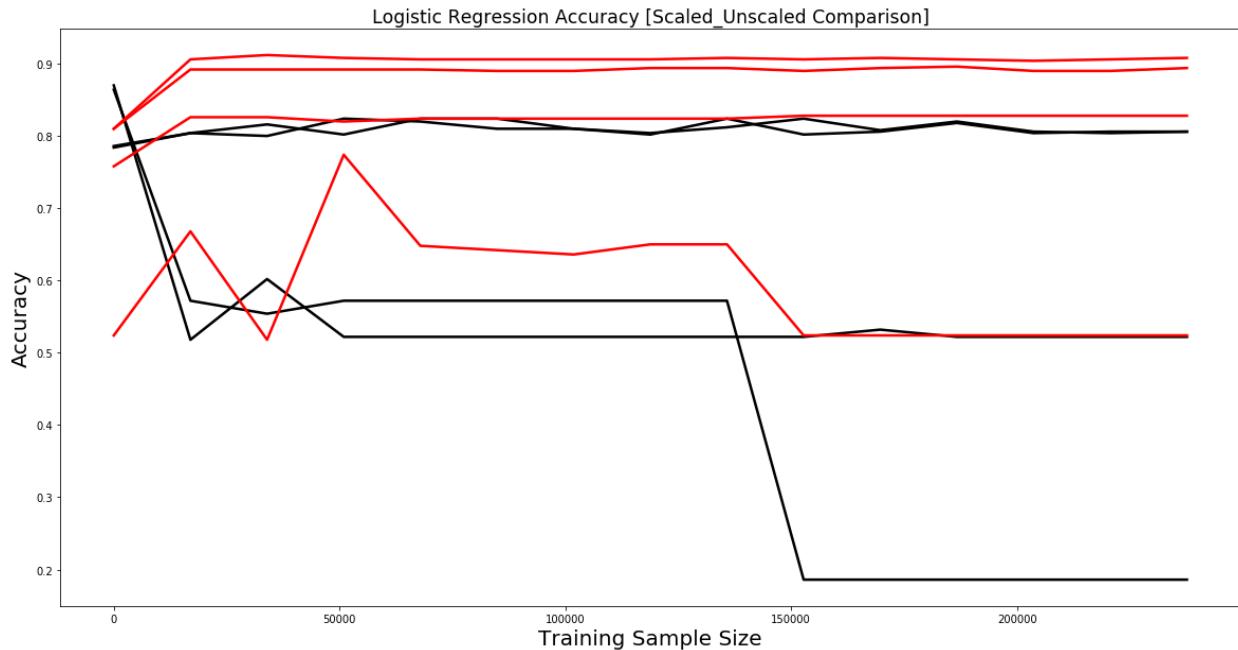
الشكل (13) مخطط يوضح أثر زيادة حجم معطيات التدريب على دقة النتائج التي تقدمها خوارزمية الانحدار اللوجستي من أجل قيم المعطيات قبل عملية التسوية

بينما يوضح الشّكل (14) مقارنة أداء خوارزميّة الانحدار اللوجستي عند استخدام كل من مجموعات المعطيات بعد عمليّة تسوية القيم.



الشّكل (14) مخطّط يوضّح أثر زيادة حجم معطيات التّدريب على دقة النّتائج التي تقدّمها خوارزميّة الانحدار اللوجستي من أجل قيم المعطيات بعد عمليّة التّسوية

ويوضّح الشّكل (15) مقارنة بين دقة النّتائج قبل وبعد عمليّة تسوية القيم، حيث تعبر المنحنيات السّوداء عن الدقة قبل التّسوية لكل من المجموعات الأربع، ونظائرها الحمراء تعبر عن الدقة بعد التّسوية.



الشكل (15) مخطط يوضح أثر تسوية قيم المعطيات على دقة النتائج التي تقدمها خوارزمية الانحدار اللوجستي "المنحنيات السوداء للقيم قبل التسوية، والحمراء للقيم بعد التسوية"

الوصول للنموذج المستقر:

نلاحظ أن الدقة التي حصلنا عليها عندأخذ عينات كبيرة الحجم من معطيات التدريب قريبة جداً من التي حصلنا عليها من أجل الحجوم الصغيرة، وهذا يطابق الدراسة النظرية حيث ذكرنا سابقاً أن هذه الخوارزمية لا تحتاج إلى حجوم كبيرة من معطيات التدريب حتى تصل إلى النموذج المستقر "في حال وجوده". فنلاحظ أن النموذج يصل إلى حالته المستقرة بعد أقل من 5000 عملية تدريب.

المقارنة بين أداء مجموعات المعطيات:

نلاحظ أن النموذج يعطي نتائج أفضل من أجل مجموعات المعطيات ذات العدد الأكبر من الحقول. فنلاحظ أنه بالنسبة للمنحي المعيّر عن المجموعة (Very Different Features) أن الدقة تتناقص بشكل كبير عند زيادة حجم الدخل لتصل في حالة المجموعة الأولى إلى دقة 20%， أي ما هو أسوء من 50% "حالة إعطاء قيم عشوائية تماماً للمتحول الهدف". ومنه نجد أن استخدام هذه المجموعة غير صالح لهذه الخوارزمية وهذا يدل على أن طبيعة العلاقة بين قيم حقول هذه المجموعة وقيمة المتحول الهدف هي ليست ذات طبيعة خطية.

أما في حالة استخدام جميع الحقول، فإن امتلاك الخوارزمية لـكامل المعلومات المتوفّرة سمح لها باعطاء نتائج جيّدة، بدقة تصل إلى 82%.

بالتأمّل بهذه النتائج نستطيع القول أن الارتباط بين قيم المتحول الهدف وقيم الحقول كان قريباً إلى حد ما من الطبيعة الخطية نظراً لإعطاء الخوارزمية دقة جيّدة.

مقارنة النتائج قبل وبعد عملية تسوية القيم:

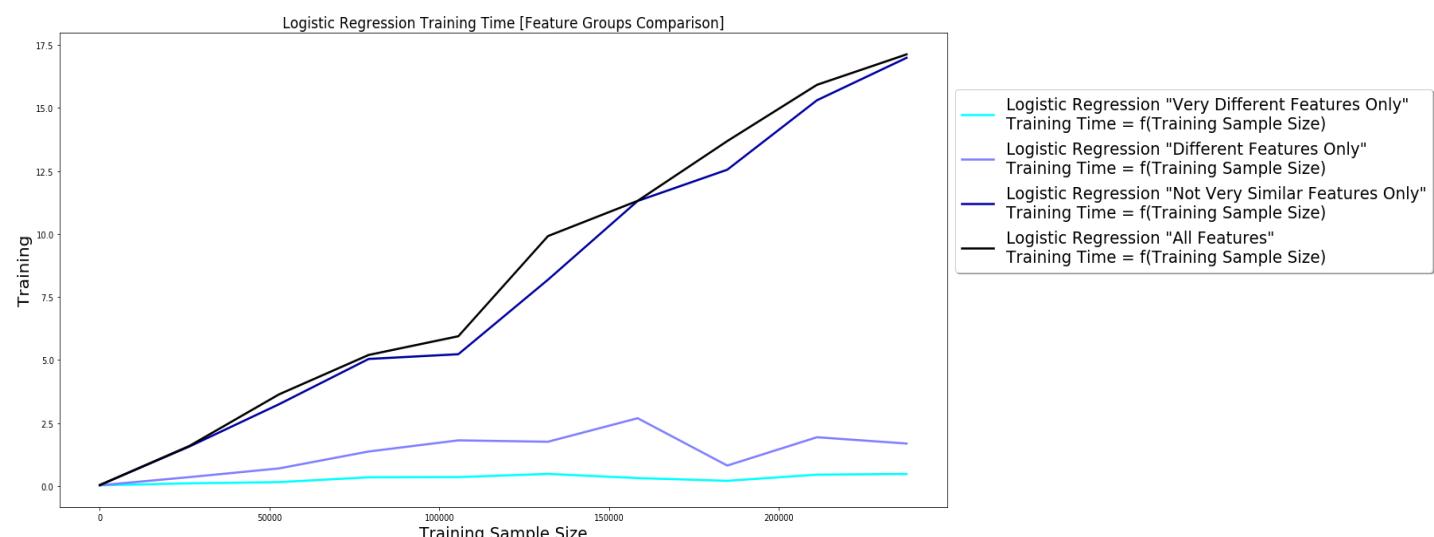
نلاحظ من الشّكل (3) أنَّ عملية التسوية للقيم حسنت كثيراً من دقة النتائج وهذا يطابق الدّراسة النظرية حيث ذكرنا فيها أنَّ عملية تسوية القيم سيكون لها أثر إيجابي على أداء الخوارزمية، فنلاحظ أنَّ استخدام المجموعة ”All Features (Scaled)“ قدّم أفضل النتائج بدقة تصل إلى 92%.

أفضل مجموعة معطيات:

مما سبق نجد أنَّنا نحصل على أدق النتائج من هذه الخوارزمية عند استخدام المجموعة ”All Features (Scaled)“ التي تحوي كافة الحقول بعد تسوية قيمها، بدقة وصلت إلى 92% وذلك بعد عملية تدريب على الأقل 4000.

زمن التّدريب:

يوضح الشّكل (16) زمن التّدريب عند استخدام كل من مجموعات المعطيات:

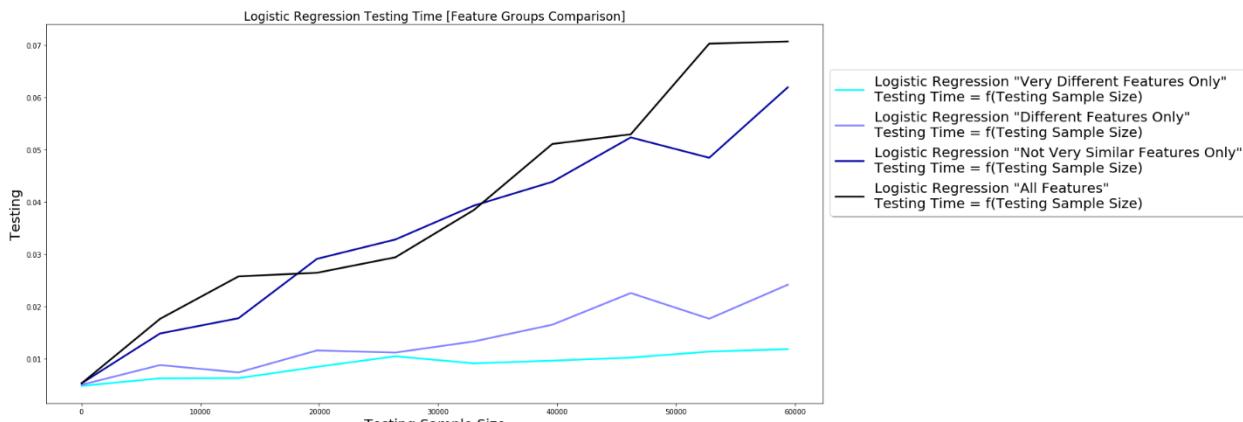


الشّكل (16) مخطط يوضح أثر زيادة حجم معطيات التّدريب على المدة الزّمنية اللازمة لانهاء عملية التّدريب من أجل خوارزمية الانحدار اللوجستي

نلاحظ من الشّكل (16) أنَّ زيادة حجم معطيات التّدريب أدّى إلى زيادة زمن التّدريب، ونفس الأمر ينطبق بالنسبة لزيادة عدد الحقول، وهذا الأمر طبيعيٌّ نظراً لأنَّ معالجة المزيد من المعطيات والحقول يتطلّب المزيد من العمليات الحسابية.

زمن التنبؤ:

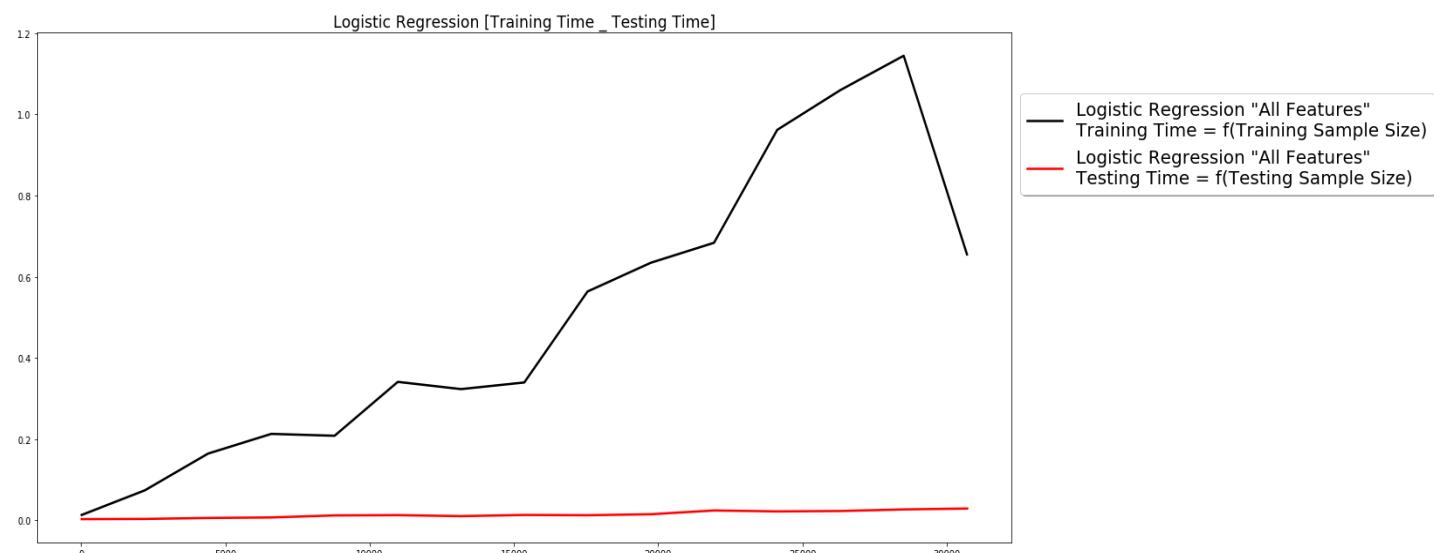
يوضح الشكل (17) زمن التنبؤ عند استخدام كل من مجموعات المعلميات:



الشكل (17) مخطط يوضح أثر زيادة حجم معطيات التنبؤ على المدة الزمنية اللازمة لانهاء عملية التنبؤ من أجل خوارزمية الانحدار логистي

نلاحظ من الشكل (17) أنَّ زيادة حجم معطيات التنبؤ أدى إلى زيادة زمن التنبؤ، ونفس الأمر ينطبق بالنسبة لزيادة عدد الحقول، وهذا الأمر طبيعي نظراً لأنَّ معالجة المزيد من المعطيات والحقول يتطلب المزيد من العمليات الحسابية.
مقارنة زمن التدريب مع زمن التنبؤ:

يوضح الشكل (18) الفرق بين زمن التدريب وزمن التنبؤ لخوارزمية الانحدار логистي:



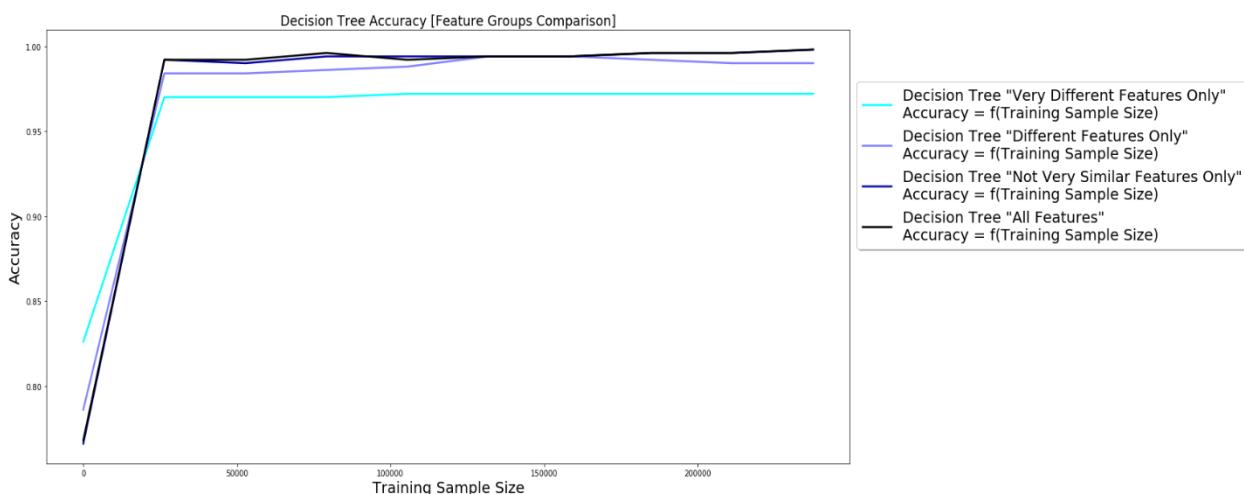
الشكل (18) مقارنة بين زمن التدريب وزمن التنبؤ عند استخدام خوارزمية الانحدار логистي

نلاحظ من المخطط أنَّ زمن التَّنْبُؤ يكاد يكون مهملًا أمام زمن التَّدريب، وهذا بديهيٌ كون عملية التَّنْبُؤ تتم من خلال تطبيق المعادلة الخطية مباشرةً على قيم الدَّخول للحصول على النَّتائج، والتي تُعد عملية سريعة جدًّا مقارنةً بعملية التَّدريب التي تقضي إجراء حسابات كثيرة لأجل تعديل قيم معاملات النَّموذج بغية الحصول على خطأً أصغر.

2.4.3.2 شجرة القرار (Decision Tree)

الدقة:

يوضح الشكل (19) مقارنة أداء خوارزمية شجرة القرار عند استخدام كل من مجموعات المعطيات:



الشكل (19) مخطط يوضح أثر زيادة حجم معطيات التَّدريب على دقة النَّتائج التي تقدمها خوارزمية شجرة القرار

الوصول للنموذج المستقر:

نلاحظ أنَّ الدقة تزايَدت بشكلٍ سريع حتَّى وصلت إلى حدٍ "حوالى 98%" عند حجم دخل 35000 سطر، وبعدها توقفت الزيادة، وذلك من أجل جميع مجموعات المعطيات، مما يدلُّ على وصول النَّموذج إلى حالة المستقرة عند حجم معطيات التَّدريب هذا، ولا حاجة إلى المزيد من عمليات التَّدريب. وإنَّ وصول النَّموذج إلى حالة المستقرة بعد 35000 عملية تدريب "والذي يُعدُّ رقمًا صغيرًا نوعًا ما" يطابق الدراسة النظرية حيث ذكرنا عدم الحاجة إلى حجم كبير لمعطيات التَّدريب للوصول إلى النَّتائج المطلوبة.

المقارنة بين أداء مجموعات المعطيات:

نلاحظ من المخطط أنَّ الدقة عند استخدام عدد أكبر من الحقول أعطت نتائج أفضل، فاستخدام كامل الحقول أعطى أفضل النَّتائج واستخدام 4 فقط أعطى أسوئها، على الرَّغم من ذلك فإنَّ الدقة التي وصلت لها النَّتائج حتَّى عند استخدام 4 حقول فقط تجاوزت الـ 97%， والتي تعد دقةً ممتازة. كما نلاحظ أنَّ الدقة كانت جيِّدة عند القيم الصغيرة نسبيًّا لحجم معطيات التَّدريب "فوق 80%". بينما

نلاحظ وصول التمودج إلى دقة شبه مثالية بعد استقراره "أكثر من 99%" عند استخدام كامل الحقول.

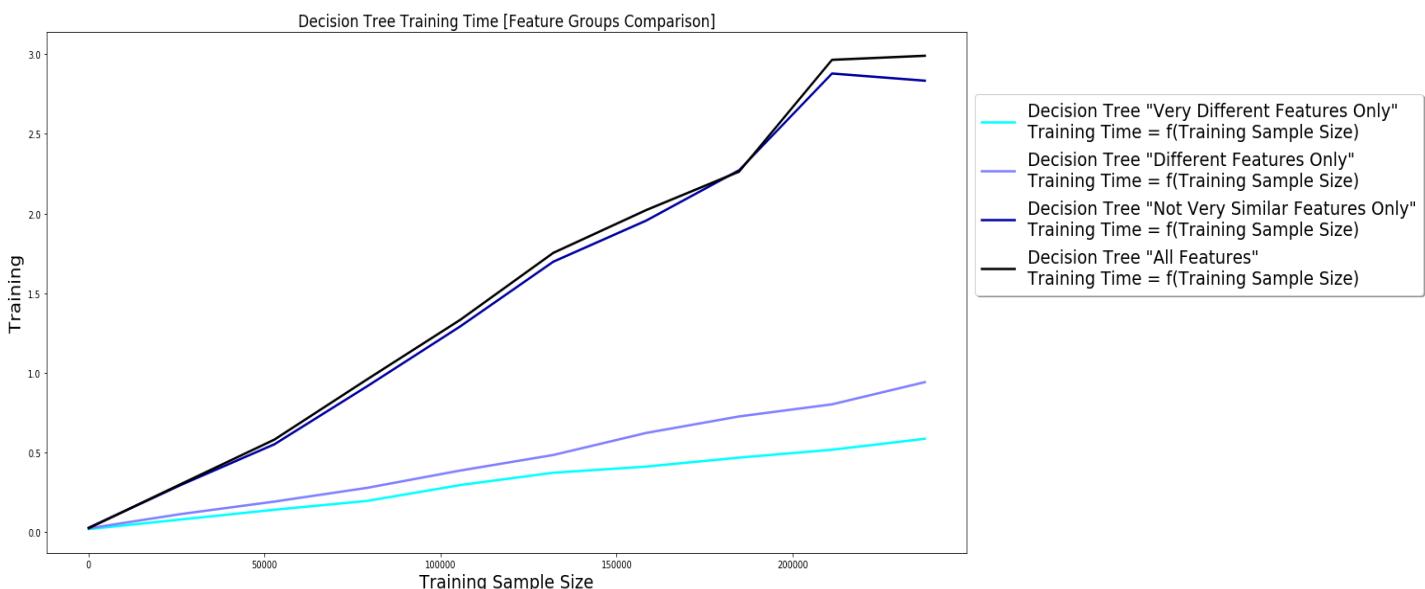
يجدر الذكر بأنه كان بالإمكان أن نصل لمثل هذه النتائج دون الحاجة إلى القيام بأي عمليات تعديل على المعطيات، حيث أن خوارزمية شجرة القرار يمكنها التعامل مع القيم غير العددية، وبالتالي فلم يكن هناك حاجة لأي عملية من عمليات المعالجة على البيانات حتى تستطيع الخوارزمية الوصول إلى هذه النتائج، وهذا أيضاً يطابق الدراسة النظرية حيث ذكرنا عدم الحاجة إلى تعديل البيانات قبل تطبيق الخوارزمية.

أفضل مجموعة بيانات:

مما سبق نجد أننا نحصل على أدق النتائج من هذه الخوارزمية عند استخدامنا للمجموعة "All Features" بـ 99% وذلك بعد 35000 عملية تدريب على الأقل.

زمن التدريب:

يوضح الشكل (20) زمن التدريب عند استخدام كل من مجموعات المعطيات:

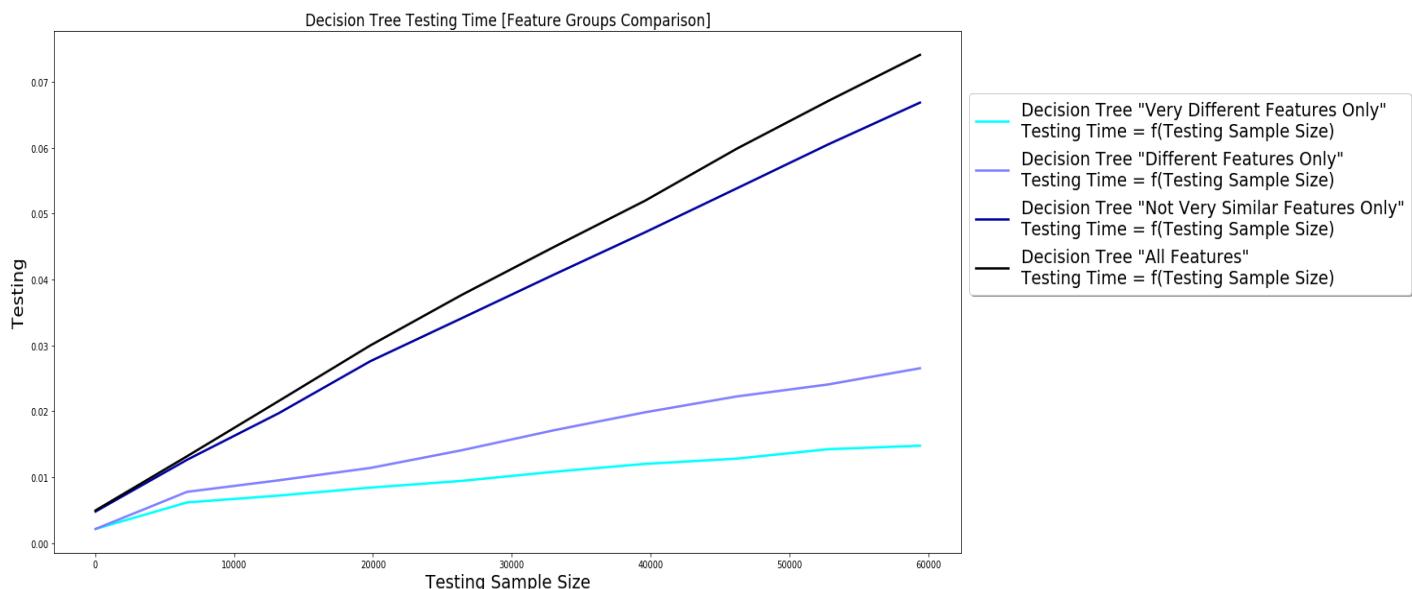


الشكل (20) مخطط يوضح أثر زيادة حجم معطيات التدريب على المدة الزمنية اللازمة لانهاء عملية التدريب من أجل خوارزمية شجرة القرار

ونلاحظ أنَّ الزَّمن يزداد عند زيادة عدد الحقول وزيادة حجم معطيات التدريب وهذا أمر طبيعي.

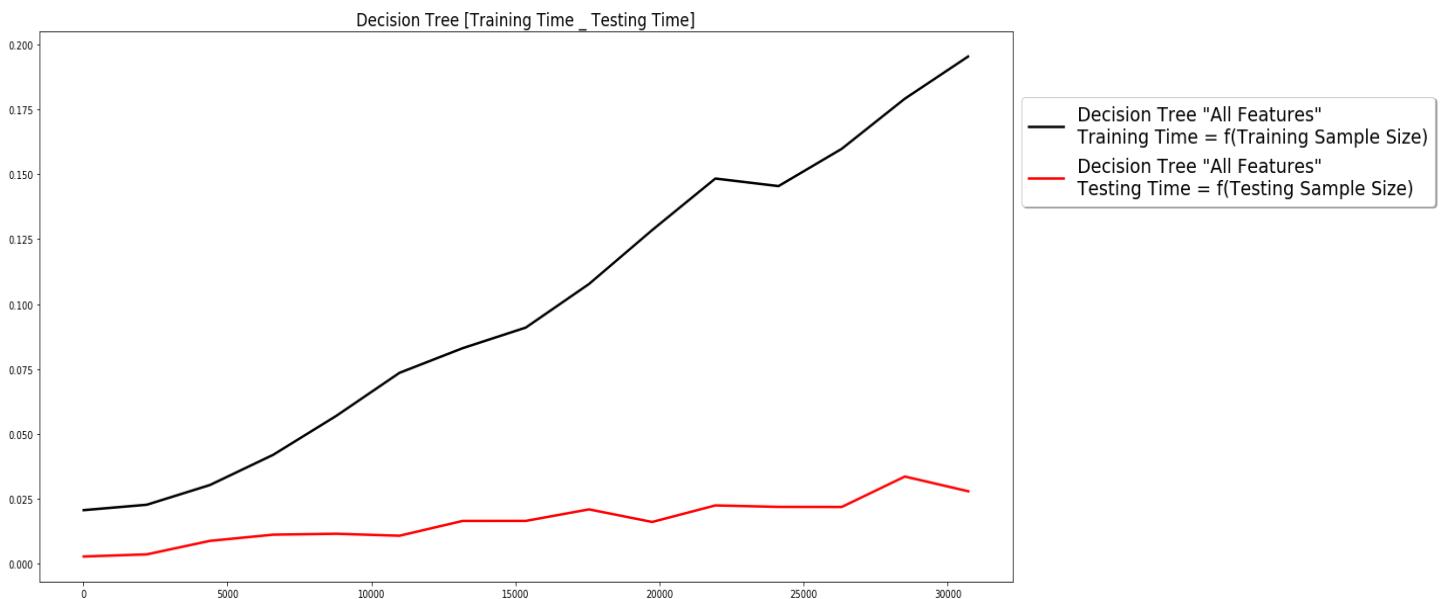
زمن التنبؤ:

يوضح الشكل (21) زمن التنبؤ عند استخدام كل من مجموعات المعلميات:



الشكل (21) مخطط يوضح أثر زيادة حجم معلميات التنبؤ على المدة الزمنية اللازمة لانهاء عملية التنبؤ من أجل خوارزمية شجرة القرار

ونلاحظ أنَّ الزَّمن يزداد عند زيادة عدد الحقول وزيادة حجم معلميات التدريب وهذا أمر طبيعي.
مقارنة زمن التدريب بزمن التنبؤ



الشكل (22) مقارنة بين زمن التدريب وزمن التنبؤ عند استخدام خوارزمية شجرة القرار

نلاحظ من المخطط أنَّ زمن التَّنبؤ يكاد يكون مهملًا أمام زمن التَّدريب، وهذا بديهيٌ كون عملية التَّنبؤ تتم من خلال مرور واحد على شجرة القراء مع اختبار شرط بسيط في كل مستوى من الشجرة، وَتُعد هذه العملية سريعة جدًا مقارنة بعملية التَّدريب التي تتطلب بناء الشجرة، توليد الشروط وتعديلها.

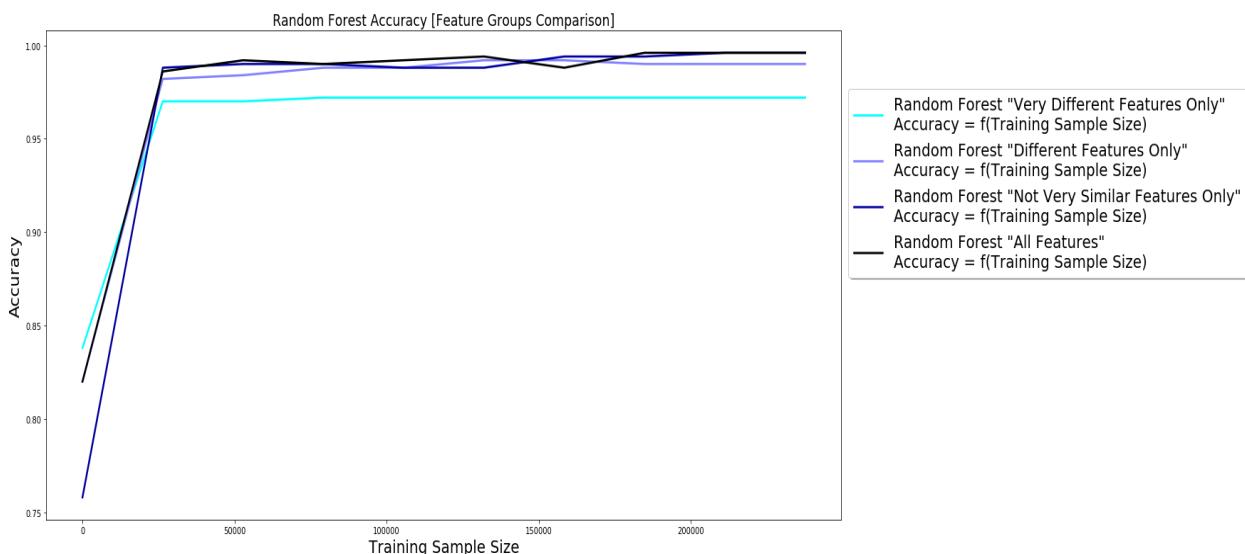
3.4.3.2 الغابة العشوائية (Random Forest):

كما ذكرنا في الدراسة النظرية، فإنَّ تحديد قيمة عدد الأشجار ليس بالأمر البديهي ولا توجد طريقة معينة لاختيار القيمة المثلثي.

ولذلك قمنا بتجربة العديد من القيم ضمن المجال [2, 10] ووجدنا أنَّ أفضل النَّتائج كانت عند القيمة 5، ومنه هذا هو عدد الأشجار الذي اعتمدناه.

الدقة:

يوضح الشكل (23) مقارنة أداء خوارزمية شجرة القراء عند استخدام كل من مجموعات المعطيات:



الشكل (23) مخطط يوضح أثر زيادة حجم معطيات التَّدريب على دقة النَّتائج التي تقدمها خوارزمية الغابة العشوائية

الوصول للنموذج المستقر:

نلاحظ أنَّ الدقة كانت جيدة عند القيم الصغيرة نسبيًّا لحجم معطيات التَّدريب "فوق 80%", كما نلاحظ أنَّ الدقة تزايِدت بشكل سريع حتى وصلت إلى حد معين عند حجم دخل 35000 سطر، وبعدها توقفت الزيادة، وذلك من أجل جميع مجموعات المعطيات، مما يدلُّ على وصول النموذج إلى حالته المستقرة عند حجم معطيات التَّدريب هذا، ولا حاجة إلى المزيد من عمليات التَّدريب، وإنَّ وصول النموذج إلى حالته المستقرة بعد 35000 عملية تدريب "والذي يعد رقماً صغيراً نوعاً ما" يطابق الدراسة النظرية حيث ذكرنا عدم الحاجة إلى حجم كبير لمعطيات التَّدريب للوصول إلى النَّتائج المطلوبة. نلاحظ هنا أنَّ نموذج شجرة القراء وصل إلى حالته المستقرة بعد 35000 عملية

تدريب أيضاً، وهذا أمر طبيعي كون نموذج الغابة العشوائية هو عبارة عن مجموعة من نماذج أشجار قرار.

المقارنة بين أداء مجموعات المعطيات:

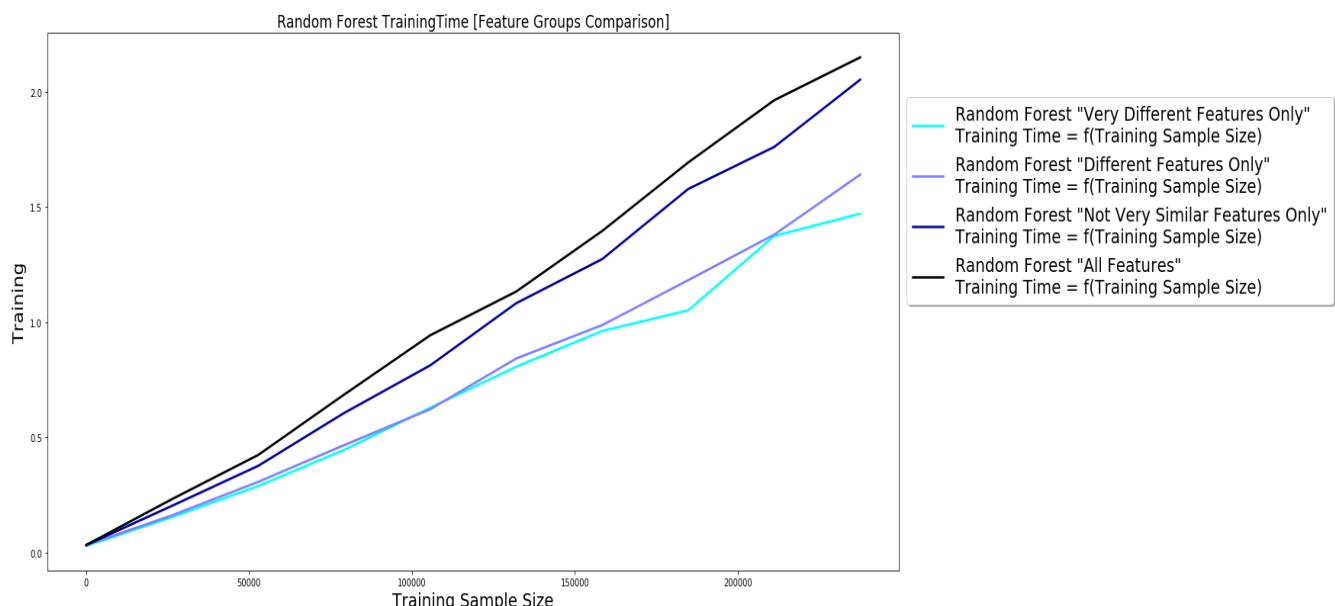
نلاحظ أن طبيعة المخططات التي حصلنا عليها مشابهة إلى حد بعيد تلك التي حصلنا عليها عند استخدام خوارزمية شجرة القرار، مع تحسن طفيف في الدقة، ومنه نتوصل إلى نفس النتائج. استخدام كامل الحقول هو الخيار الأمثل، بينما يبقى استخدام أي من المجموعات الثلاثة الأخرى خياراً مقبولاً بدقة قريبة من تلك التي سنحصل عليها عند استخدام كامل الحقول.

أفضل مجموعة معطيات:

مما سبق نجد أننا نحصل على أدق النتائج من هذه الخوارزمية عند استخدامنا للمجموعة "All Features" بدقّة تجاوزت 99% وذلك بعد 35000 عملية تدريب على الأقل.

زمن التدريب:

يوضح الشكل (24) زمن التنبؤ عند استخدام كل من مجموعات المعطيات:

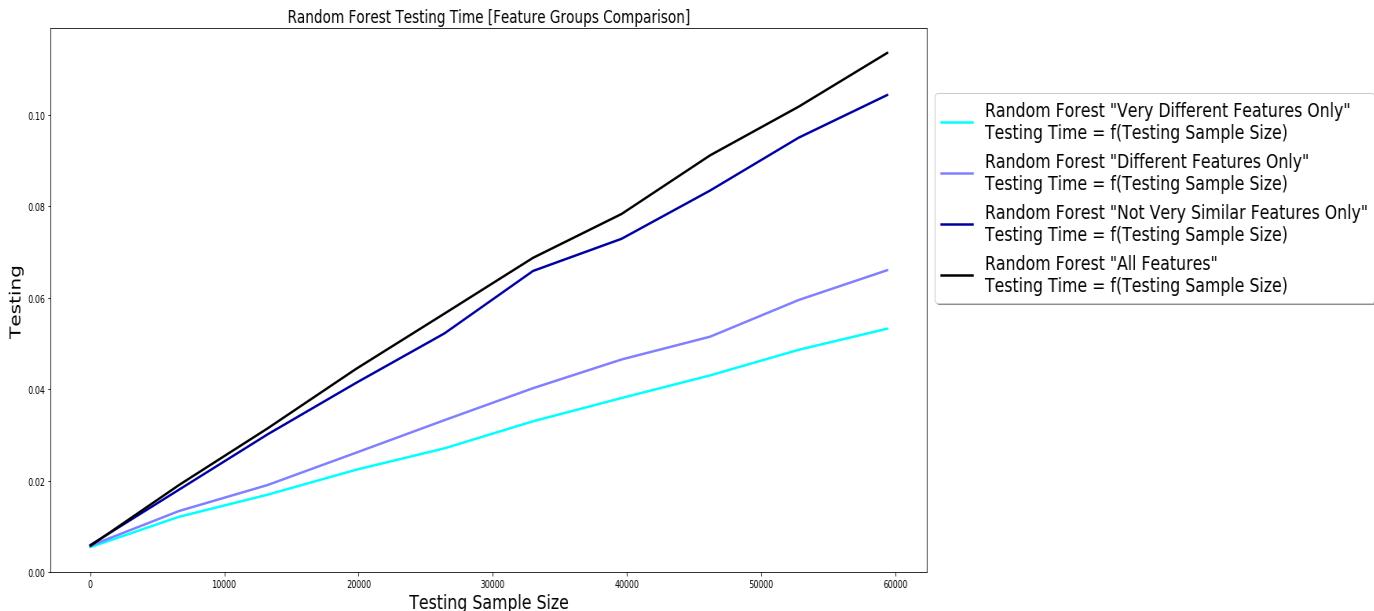


الشكل (24) مخطط يوضح أثر زيادة حجم معطيات التدريب على المدة الزمنية اللازمة لانهاء عملية التدريب من أجل خوارزمية الغابة العشوائية

ونلاحظ أن الزّمن يزداد عند زيادة عدد الحقول وزيادة حجم معطيات التدريب وهذا أمر طبيعي.

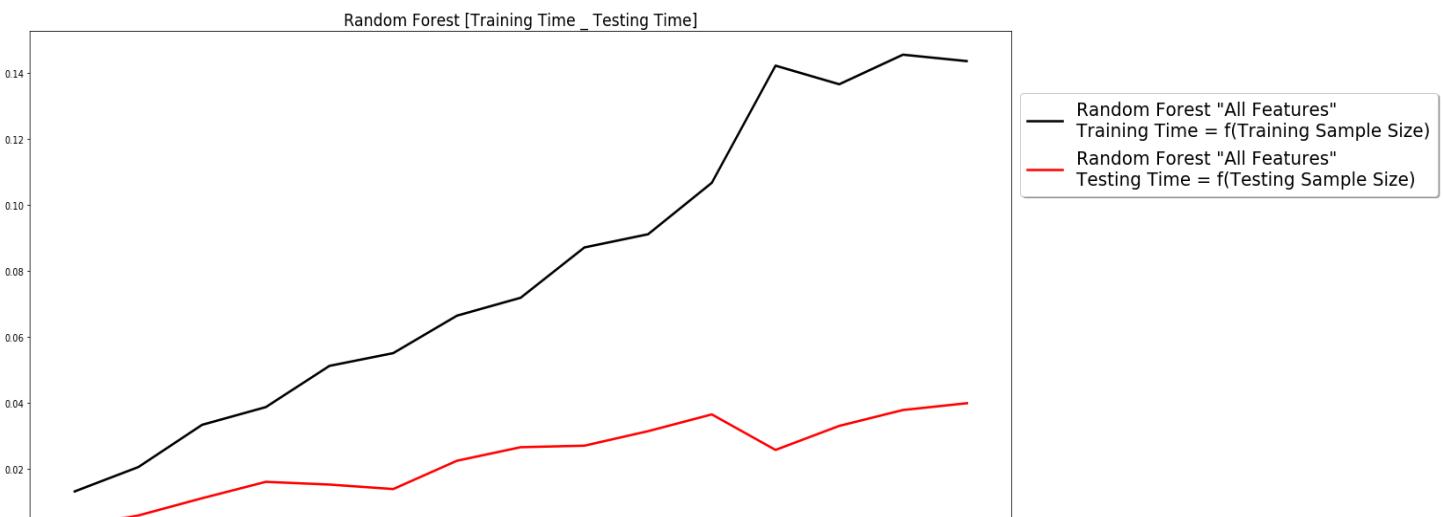
زمن التنبؤ:

يوضح الشكل (25) زمن التنبؤ عند استخدام كل من مجموعات المعلميات:



الشكل (25) مخطط يوضح أثر زيادة حجم معلميات التنبؤ على المدة الزمنية اللازمة لانهاء عملية التنبؤ من أجل خوارزمية شجرة القرار

ونلاحظ أنَّ الزَّمن يزداد عند زيادة عدد الحقول وزيادة حجم معلميات التدريب وهذا أمر طبيعي.
مقارنة زمن التدريب بزمن التنبؤ



الشكل (26) مقارنة بين زمن التدريب وزمن التنبؤ عند استخدام خوارزمية الغابة العشوائية

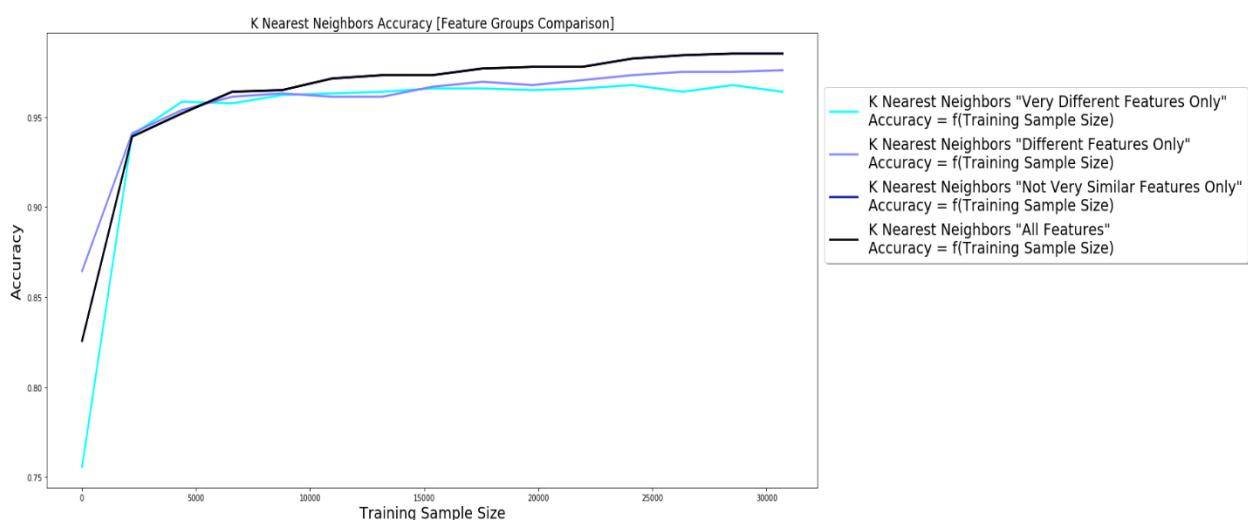
نلاحظ من المخطط أنَّ زمن التنبؤ صغير أمام زمن التدريب، وهذا بديهي كون عملية التنبؤ تتم من خلال مرور واحد على كل شجرة من أشجار القرار مع اختبار شرط بسيط في كل مستوى منها، وثُمَّ هذه العملية سريعة نسبياً، وهي تتعلق بعدد الأشجار المستخدمة "5 في حالتنا".

4.4.3.2 أقرب k جار (K Nearest Neighbors):

كما ذكرنا في الدراسة النظرية فإنَّ تحديد قيمة k ليست بديهية وبالتالي قمنا بتجرب العديد من القيم ضمن المجال [1, 20] وحصلنا على أفضل النتائج عند القيمة 5، وبالتالي سنعتمد هذه القيمة في تجربتنا.

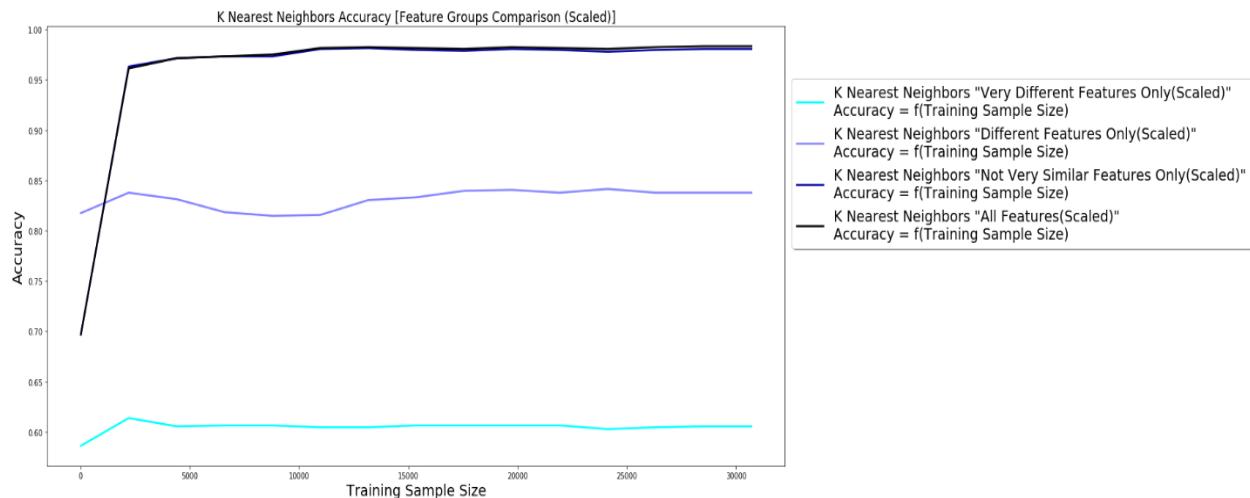
الدقة:

يوضح الشكل (27) مقارنة أداء خوارزمية أقرب k جار عند استخدام كل من مجموعات المعطيات قبل عملية تسوية القيم.



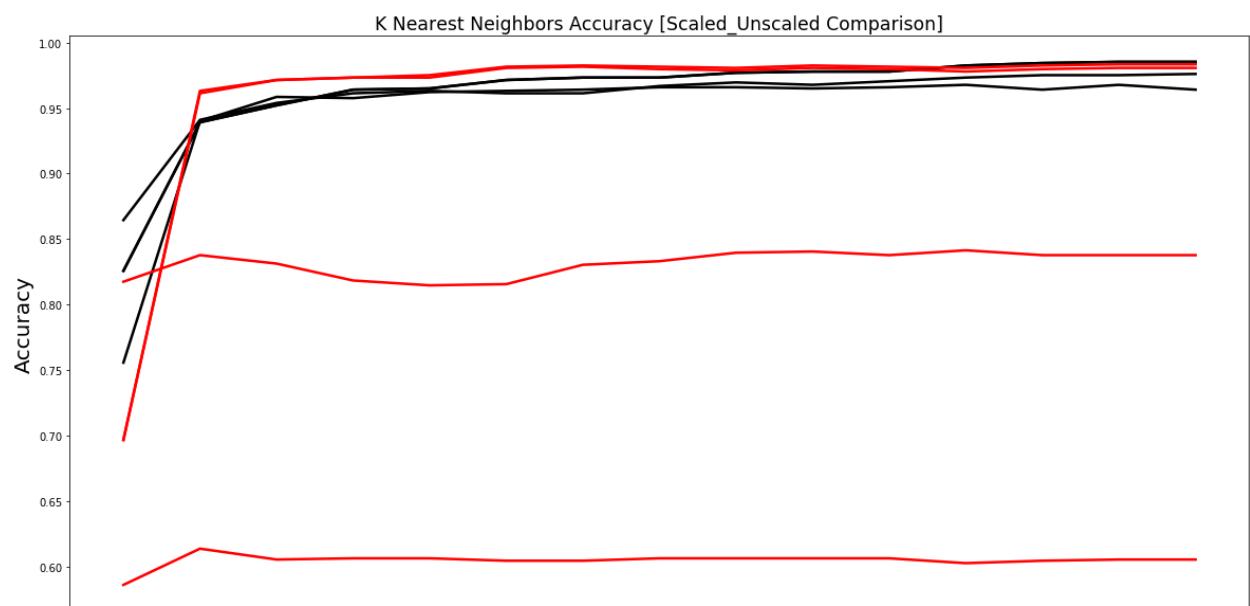
الشكل (27) مخطط يوضح أثر زيادة حجم معطيات التدريب على دقة النتائج التي تقدمها خوارزمية أقرب k جار من أجل قيم المعطيات قبل عملية التسوية

بينما يوضح الشكل (28) مقارنة أداء خوارزمية أقرب k جار عند استخدام كل من مجموعات المعطيات بعد عملية تسوية القيم.



الشكل (28) مخطط يوضح أثر زيادة حجم معطيات التدريب على دقة النتائج التي تقدمها خوارزمية أقرب k جار من أجل قيم المعطيات بعد عملية التسوية

ويوضح الشكل (29) مقارنة بين دقة النتائج قبل وبعد عملية تسوية القيم، حيث تعبر المنحنيات السوداء عن الدقة قبل التسوية لكل من المجموعات الأربع، وناظائرها الحمراء تعبر عن الدقة بعد التسوية.



الشكل (29) مخطط يوضح أثر تسوية قيم المعطيات على دقة النتائج التي تقدمها خوارزمية أقرب k جار "المنحنيات السوداء للقيم قبل التسوية، والحرماء للقيم بعد التسوية"

الوصول للنموذج المستقر:

بالنسبة للمنحنى المترافق لقيم المعطيات قبل عملية التسوية، نلاحظ وصول النموذج إلى حالة شبه مستقرة بعد حوالي 3000 عملية تدريب حيث أصبح التزايد في الدقة بسيطاً، بينما وصلت لحالة مستقرة بعد حوالي 10000 عملية تدريب، حيث باتت القيم شبه ثابتة. أما بالنسبة للمنحنى المترافق لقيم المعطيات بعد عملية التسوية، فنلاحظ وصلها لحالة الاستقرار بعد حوالي 2000 عملية تدريب فقط.

المقارنة بين أداء مجموعات المعطيات:

بالنسبة للنتائج قبل عملية التسوية، نلاحظ أن استخدام جميع مجموعات المعطيات أعطى نتائج متقاربة، مع أفضلية بسيطة لمجموعات ذات العدد الأكبر من الحقول، حيث وصلت الدقة عند استخدام المجموعة "Very Different Features" إلى حوالي 95%， بينما وصلت عند استخدام كامل الحقول إلى حوالي 98%.

أما بالنسبة للنتائج بعد عملية التسوية، فنلاحظ تدهور كبير في الدقة عند استخدام المجموعتين "Very Different Features (Scaled)", "Different Features (Scaled)"، حيث لم تتجاوز الدقة عند استخدام الأولى إلى 62%， بينما لم تتجاوز الـ 85% من أجل المجموعة الثانية. أما بالنسبة للمجموعتين "All Features (Scaled)", "Not Very Similar Features (Scaled)" فقد وصلتا إلى دقة ممتازة "حوالي 96%".

مقارنة النتائج قبل وبعد عملية تسوية القيم:

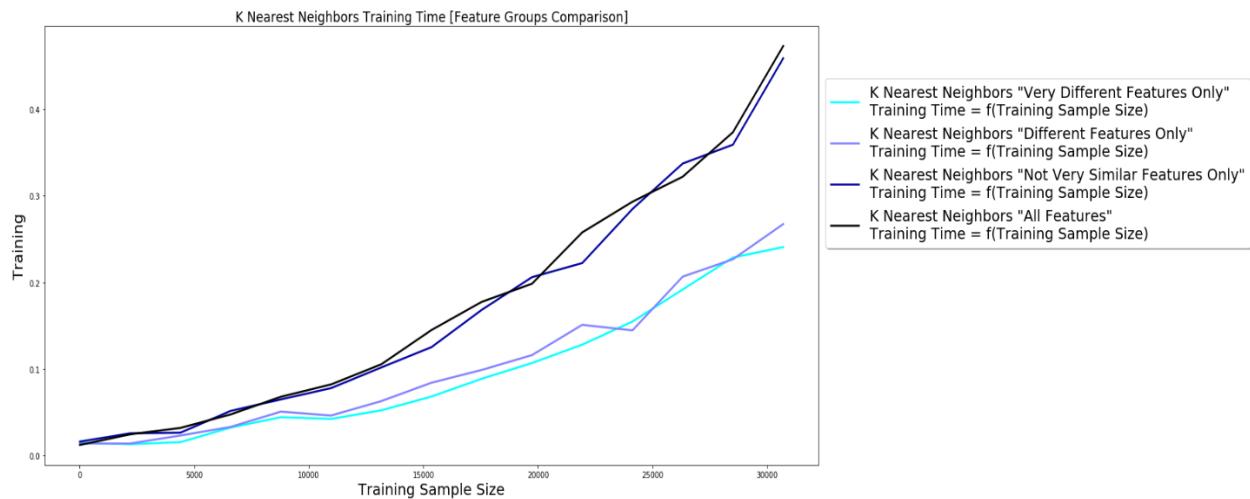
كما لاحظنا من المخططات الثلاثة السابقة، فإن عملية التسوية كان لها دور سلبي عند استخدام المجموعتين الأولى "Very Different Features" والثانية "Different Features"， بينما حسنت الدقة بشكل طفيف بالنسبة للمجموعتين الأخريين.

أفضل مجموعة بيانات:

مما سبق نجد أننا نحصل على أدق النتائج من هذه الخوارزمية عند استخدامنا للمجموعة "All Features (Scaled)"، بدقة تصل إلى حوالي 97% وذلك بعد 3000 عملية تدريب على الأقل.

زمن التدريب:

يوضح الشكل (30) زمن التدريب عند استخدام كل من مجموعات المعلميات:

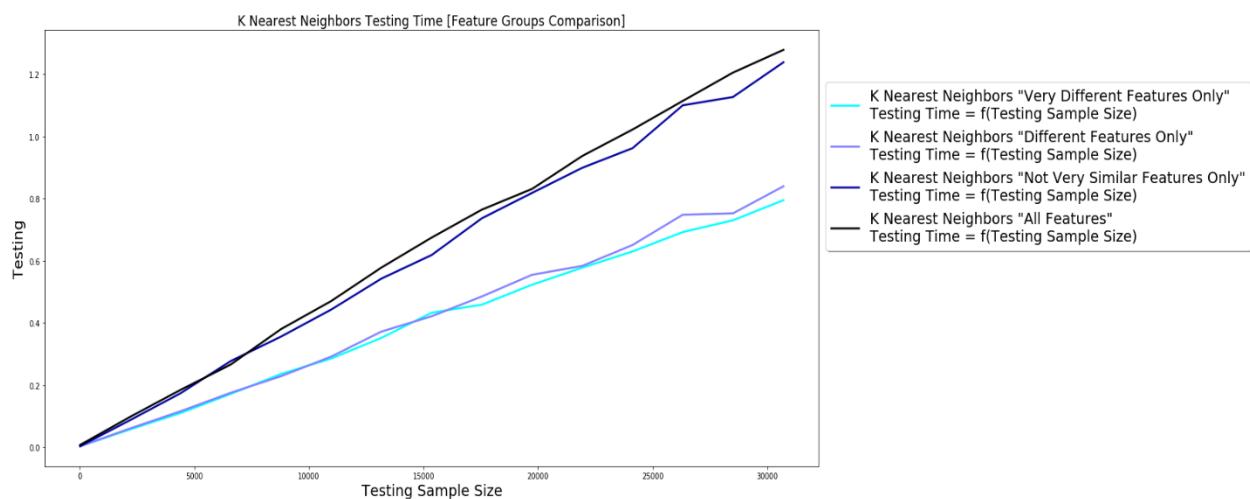


الشكل (30) يوضح أن زيادة حجم معلميات التدريب على المدة الزمنية اللازمة لانهاء عملية التدريب من أجل خوارزمية أقرب k جار

نلاحظ أن زيادة حجم معلميات التدريب أدى إلى زيادة زمن التدريب، ونفس الأمر ينطبق بالنسبة لزيادة عدد الحقول، وهذا الأمر طبيعي نظراً لأن معالجة المزيد من المعلميات والحقول يتطلب المزيد من العمليات الحسابية.

زمن التنبؤ:

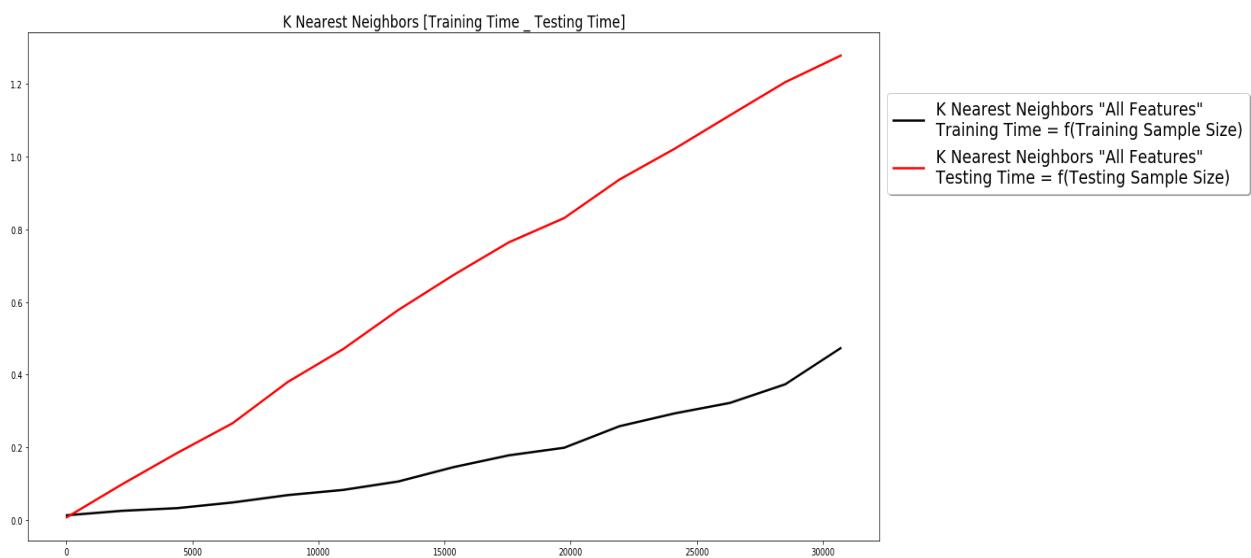
يوضح الشكل (31) زمن التنبؤ عند استخدام كل من مجموعات المعلميات:



الشكل (31) يوضح أن زيادة حجم معلميات التنبؤ على المدة الزمنية اللازمة لانهاء عملية التنبؤ من أجل خوارزمية أقرب k جار

نلاحظ أنَّ زيادة حجم معطيات التَّنبؤ أدى إلى زيادة زمن التَّنبؤ، ونفس الأمر ينطبق بالنسبة لزيادة عدد الحقول، وهذا الأمر طبيعي نظراً لأنَّ معالجة المزيد من المعطيات والحقول يتطلب المزيد من العمليات الحسابية.

مقارنة زمن التَّدريب مع زمن التَّنبؤ:
يوضح الشَّكل (32) الفرق بين زمن التَّدريب وزمن التَّنبؤ لخوارزمية أقرب k جار:



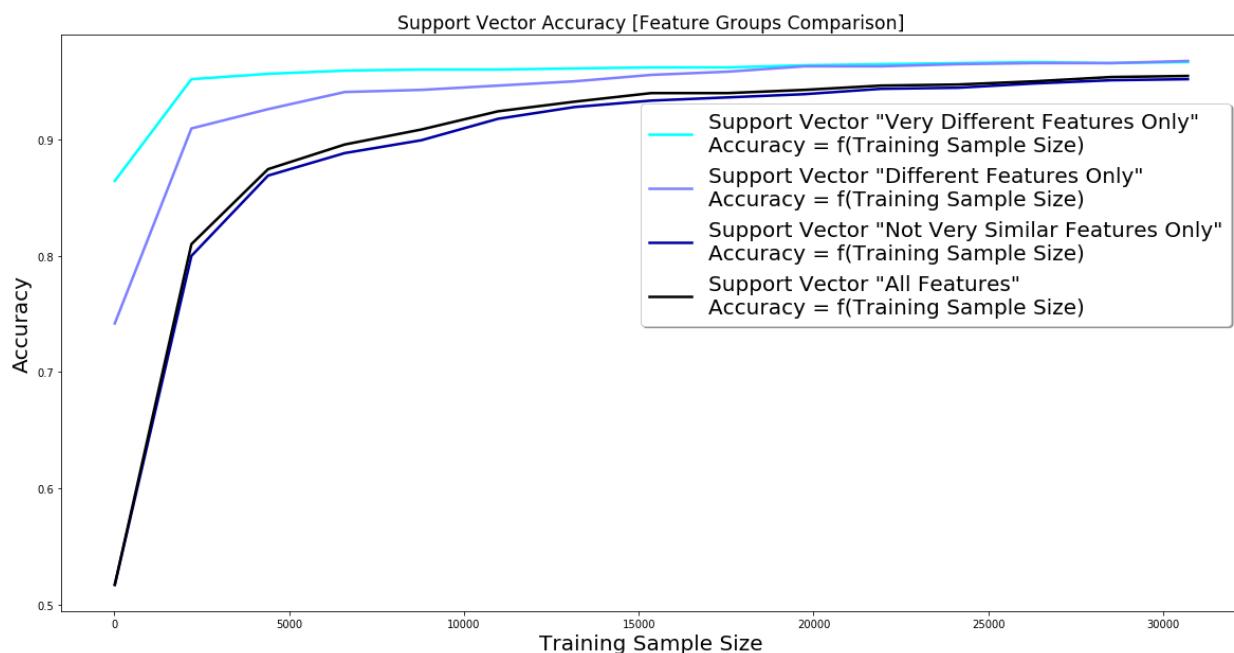
الشكل (32) مقارنة بين زمن التَّدريب وزمن التَّنبؤ عند استخدام خوارزمية أقرب k جار

ونلاحظ هنا أنَّ فترة التَّدريب صغيرة بالنسبة لفترة التَّنبؤ على عكس باقي الخوارزميات، وهذا يطابق الدراسة النَّظرية، حيث أنَّ عملية التَّدريب تقصر على تخزين النقاط بينما تتطلب عملية التَّنبؤ حساب الأبعاد عن كل النقاط المُخزَّنة ومن ثم ترتيب هذه النقاط بحسب البعد وأخذ أصغر k قيمة، والتي تتطلب بدورها وقتاً أطول.

5.4.3.2 متجه الدّعم (Support Vector):

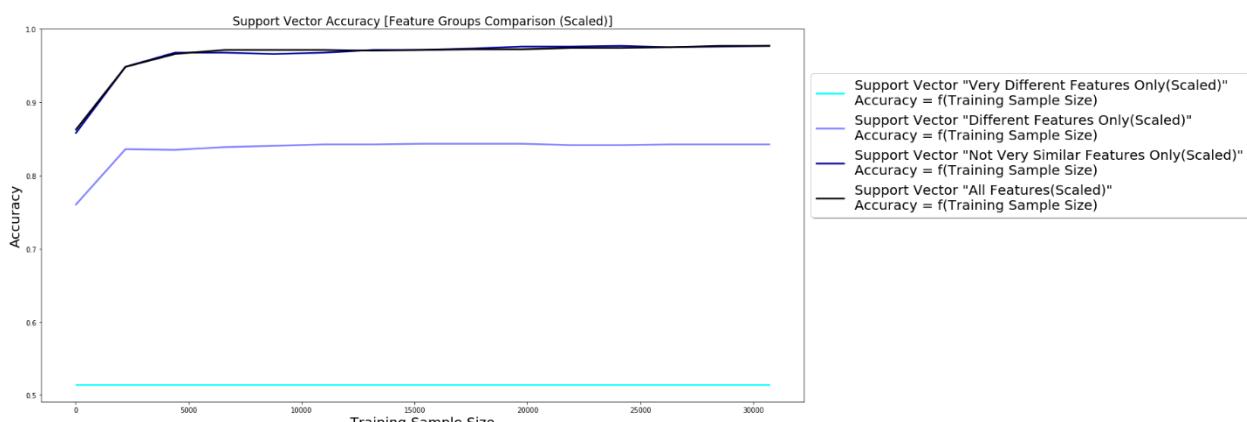
الدقة:

يوضح الشكل (33) مقارنة أداء خوارزمية متجه الدّعم عند استخدام كل من مجموعات المعطيات قبل عملية تسوية القيم.



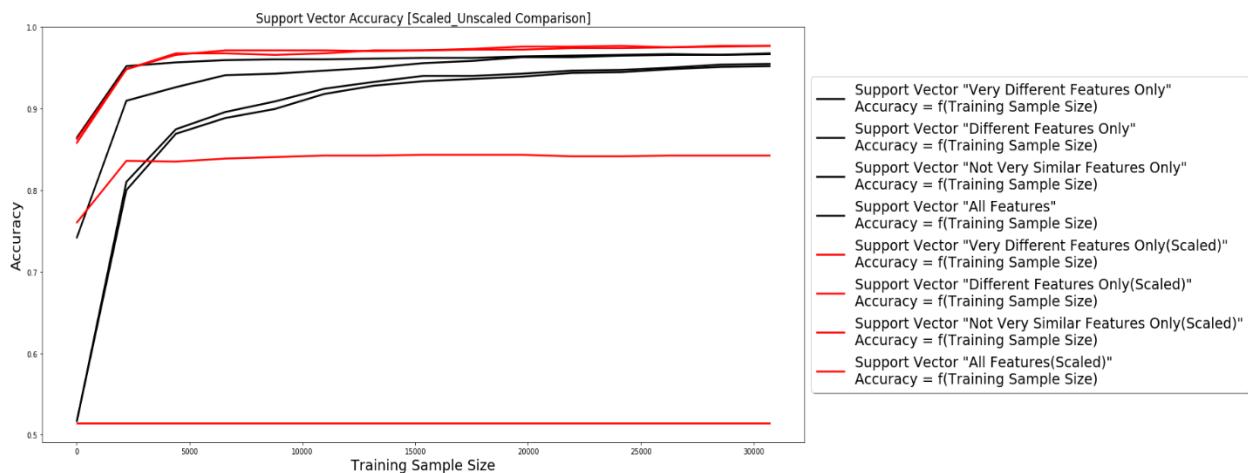
الشكل (33) منحّط يوضح أنّر زياده حجم معطيات التّدريب على دقة النّتائج التي تقدّمها خوارزمية متجه الدّعم من أجل قيم المعطيات قبل عملية التّسوية

بينما يوضح الشكل (34) مقارنة أداء خوارزمية متجه الدّعم عند استخدام كل من مجموعات المعطيات بعد عملية تسوية القيم.



الشكل (34) منحّط يوضح أنّر زياده حجم معطيات التّدريب على دقة النّتائج التي تقدّمها خوارزمية متجه الدّعم من أجل قيم المعطيات بعد عملية التّسوية

ويوضح الشكل (35) مقارنة بين دقة النتائج قبل وبعد عملية تسوية القيم، حيث تعبر المنحنيات السوداء عن الدقة قبل التسوية لكل من المجموعات الأربع، ونطائراً لها الحمراء تعبر عن الدقة بعد التسوية.



الشكل (35) مخطط يوضح أثر تسوية قيم المعطيات على دقة النتائج التي تقدمها خوارزمية متوجه الدعم "المنحنيات السوداء لقيم قبل التسوية، والحراء لقيم بعد التسوية"

الوصول للنموذج المستقر:

نلاحظ أنَّ النموذج وصل إلى الحالة المستقرة بعد حوالي ال 2000 عملية تدريب على الأكثر، وذلك عند استخدام أي من المجموعات الأربع، سواءً كان ذلك قبل عملية تسوية القيم أم بعدها.

المقارنة بين أداء مجموعات المعطيات:

قبل القيام بعملية التسوية، نلاحظ أنَّ استخدام المجموعات ذات عدد الحقول الأصغر أعطى نتائج أفضل، على عكس باقي الخوارزميات، حيث وصلت الدقة عند استخدام أول مجموعتين إلى حوالي 96%， بينما لم تتجاوز الـ 93% عند استخدام المجموعتين الثالثة والرابعة. أمَّا بعد القيام بعملية التسوية، فنلاحظ عودة السلوك العام، حيث أصبح استخدام المجموعات ذات عدد الحقول الأكبر يعطي نتائج أكثر دقة، حيث وصلت الدقة عند استخدام المجموعتين 3 و 4 إلى حوالي 97%， وعند استخدام الثانية إلى حوالي 83%， بينما أصبح استخدام المجموعة الأولى شبه لا فائدة منه بدقة لا تتجاوز 52%.

مقارنة النتائج قبل وبعد عملية تسوية القيم:

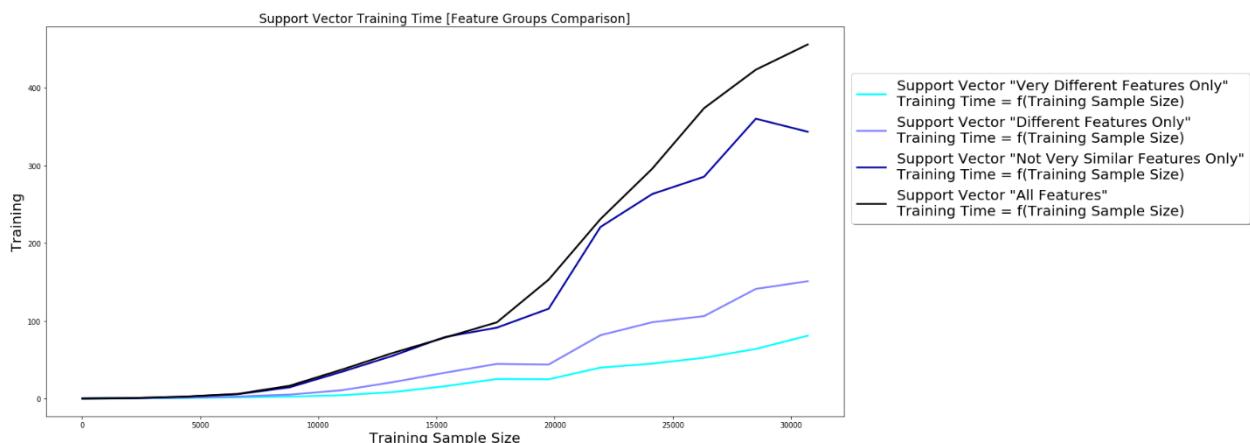
نلاحظ أنَّ عملية التسوية حسنت الأداء وذلك عند استخدام المجموعتين 3 و 4، بينما أدت إلى تدهوره عند استخدام المجموعتين 1 و 2.

أفضل مجموعة بيانات:

مما سبق نجد أننا نحصل على أدق النتائج من هذه الخوارزمية عند استخدامنا للمجموعة "All Features (Scaled)" بدقّة تجاوزت 97% وذلك بعد 5000 عملية تدريب على الأقل.

زمن التدريب:

يوضح الشكل (36) زمن التدريب عند استخدام كل من مجموعات المعطيات:

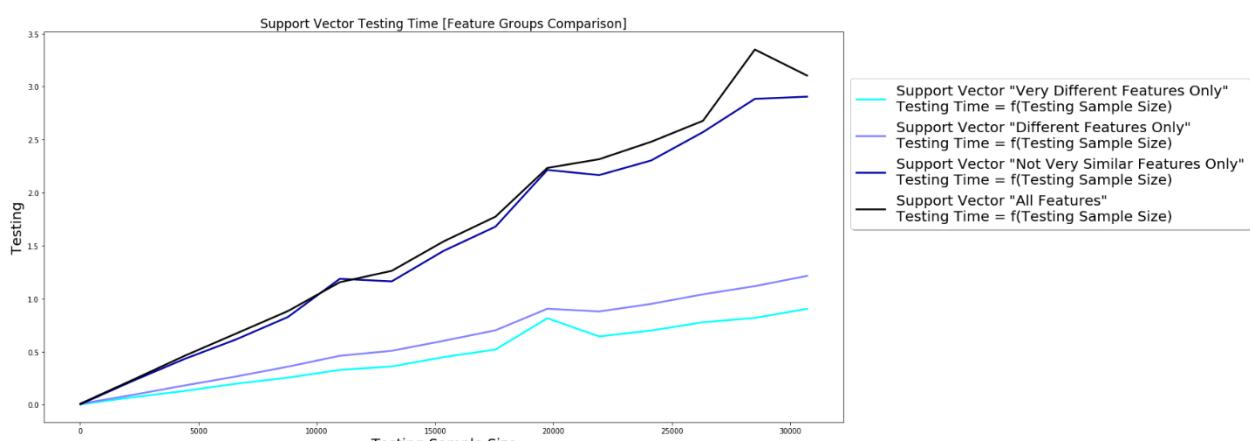


الشكل (36) مخطط يوضح أثر زيادة حجم معطيات التدريب على المدة الزمنية اللازمة لانهاء عملية التدريب من أجل خوارزمية متوجه الدعم

نلاحظ أنَّ زيادة حجم معطيات التدريب أدى إلى زيادة زمن التدريب، ونفس الأمر ينطبق بالنسبة لزيادة عدد الحقول، وهذا الأمر طبيعي نظراً لأنَّ معالجة المزيد من المعطيات والحقول يتطلب المزيد من العمليات الحسابية.

زمن التنبؤ:

يوضح الشكل (37) زمن التنبؤ عند استخدام كل من مجموعات المعطيات:

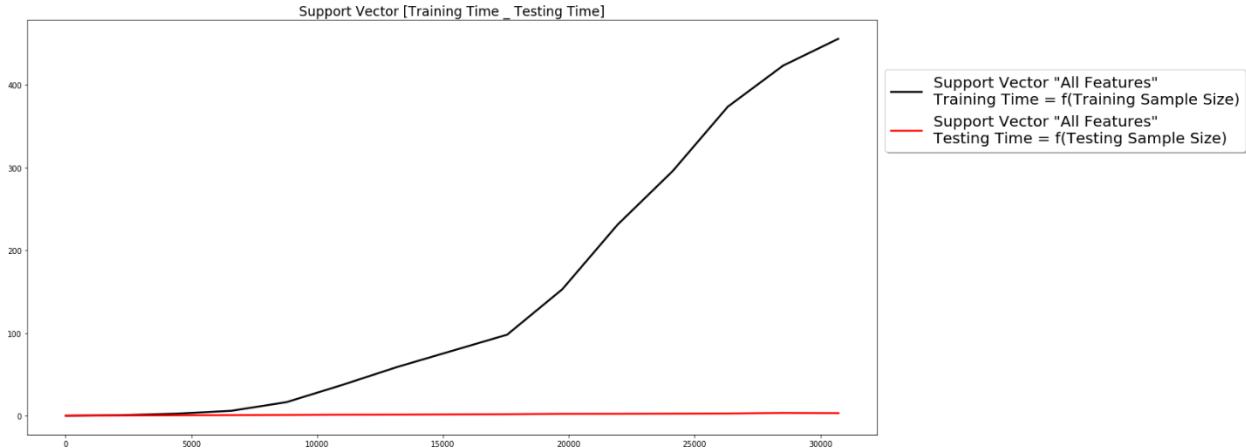


الشكل (37) مخطط يوضح أثر زيادة حجم معطيات التنبؤ على المدة الزمنية اللازمة لانهاء عملية التنبؤ من أجل خوارزمية متوجه الدعم

نلاحظ أنَّ زيادة حجم معطيات التَّنبُؤ أدى إلى زيادة زمن التَّنبُؤ، ونفس الأمر ينطبق بالنسبة لزيادة عدد الحقول، وهذا الأمر طبيعي نظراً لأنَّ معالجة المزيد من المعطيات والحقول يتطلَّب المزيد من العمليَّات الحسابيَّة.

مقارنة زمن التَّدريب مع زمن التَّنبُؤ:

يوضِّح الشَّكَل (38) الفرق بين زمن التَّدريب وزمن التَّنبُؤ لخوارزميَّة الانحدار اللوجستي:



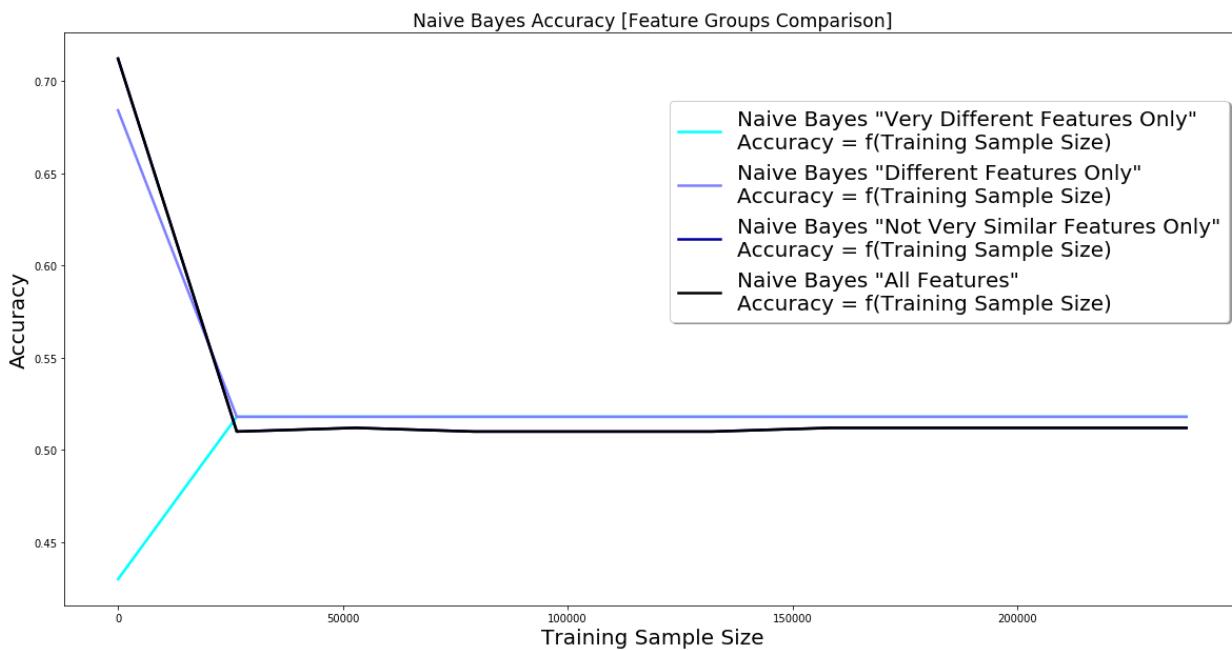
الشَّكَل (38) مقارنة بين زمن التَّدريب وزمن التَّنبُؤ عند استخدام خوارزميَّة الانحدار اللوجستي

نلاحظ من المخطط أنَّ زمن التَّنبُؤ يكاد يكون مهملًا أمام زمن التَّدريب، وهذا بديهيٌّ كون عملية التَّنبُؤ هي عبارة عن عمليَّات حسابيَّة بسيطة لمعرفة الطرف الذي تنتهي له النَّقطة، بينما تتطلَّب عملية التَّدريب عمليَّات حسابيَّة طويلة ومعقدة.

6.4.3.2 بايز مع التوزع الغاوسي (Gaussian Naïve Bayes):

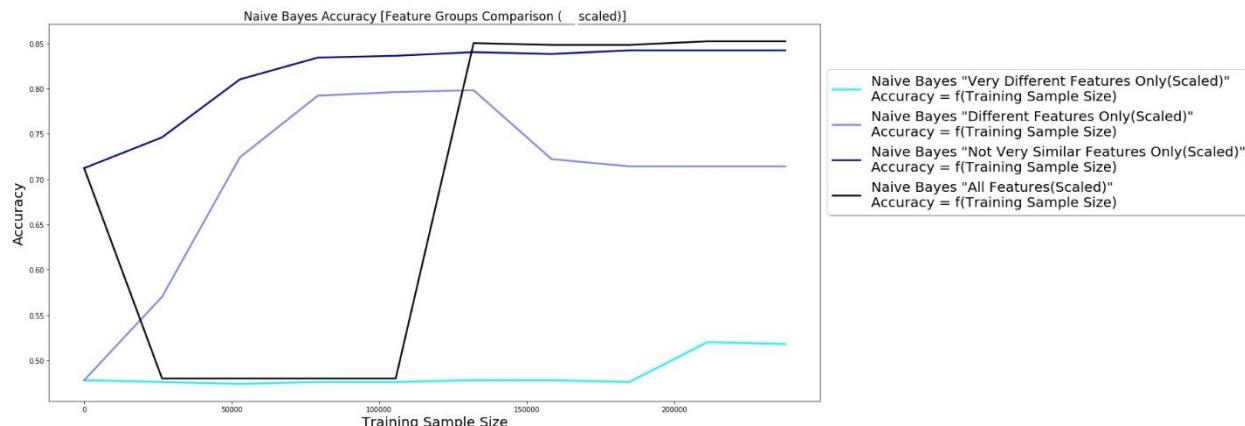
الدقة:

يوضح الشكل (39) مقارنة أداء خوارزمية بايز عند استخدام كل من مجموعات المعلميات قبل عملية تسوية القيم.



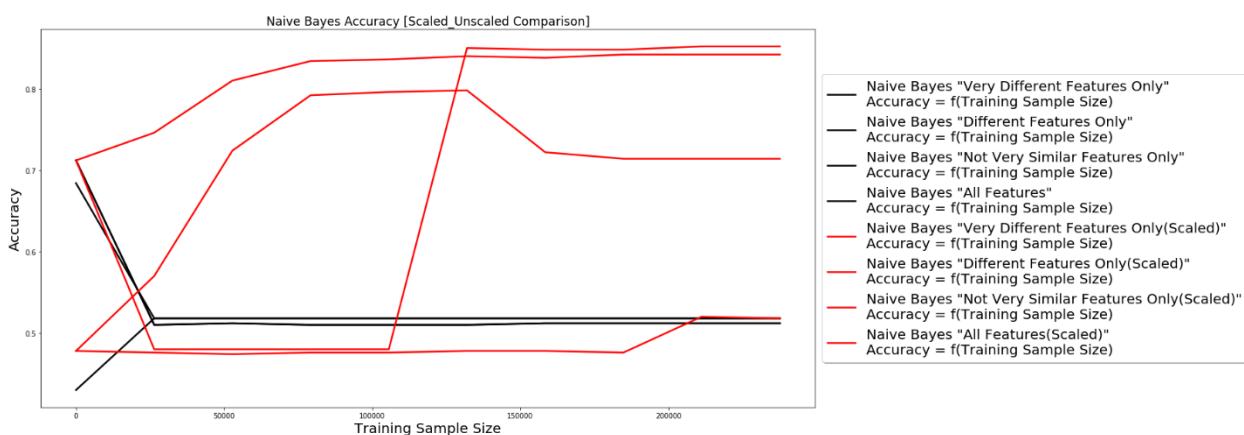
الشكل (39) يوضح أن زيادة حجم معلميات التدريب على دقة النتائج التي تقدمها خوارزمية بايز من أجل قيم المعلميات قبل عملية التسوية

بينما يوضح الشكل (40) مقارنة أداء خوارزمية بايز عند استخدام كل من مجموعات المعطيات بعد عملية تسوية القيم.



الشكل (40) مخطط يوضح أثر زيادة حجم معطيات التدريب على دقة النتائج التي تقدمها خوارزمية بايز من أجل قيم المعطيات بعد عملية التسوية

ويوضح الشكل (41) مقارنة بين دقة النتائج قبل وبعد عملية تسوية القيم، حيث تعبر المنحنيات السوداء عن الدقة قبل التسوية لكل من المجموعات الأربع، وناظائرها الحمراء تعبر عن الدقة بعد التسوية.



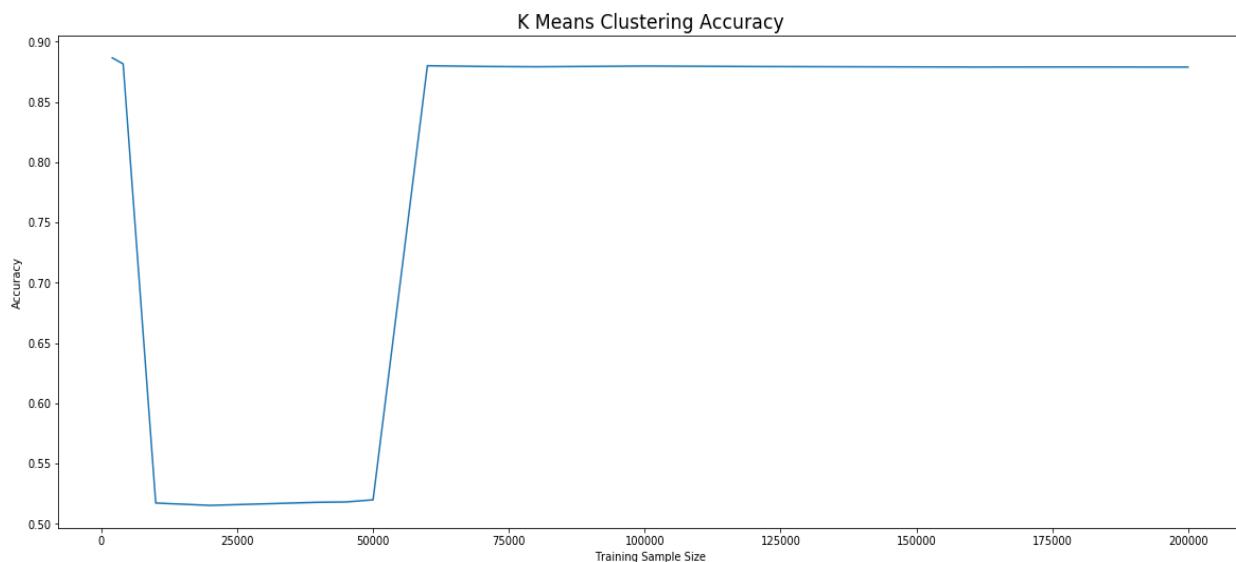
الشكل (41) مخطط يوضح أثر تسوية قيم المعطيات على دقة النتائج التي تقدمها خوارزمية بايز "المنحنيات السوداء للقيم قبل التسوية، والحرماء للقيم بعد التسوية"

على عكس باقي الخوارزميات، نلاحظ فشل خوارزمية بايز في الوصول إلى نموذج مستقر عند تسوية القيم، بينما نلاحظ وصولها إلى الاستقرار لكن بدقة سيئة للغاية لا تتجاوز الـ 53% وذلك قبل تسوية القيم، الأمر الذي يدل على عدم صلاحية هذه الخوارزمية لمسألتنا هذه. وحسب الدراسة النظرية يمكننا أن نستنتج أن سبب هذا الفشل هو كون المتحولات مرتبطة ببعضها ارتباطاًوثيقاً، وأن افتراض استقلال هذه المتحولات أدى إلى عجز الخوارزمية عن الوصول إلى نموذج جيد ومستقر لهذه المسألة. ولذلك سننجب استخدام هذه الخوارزمية في مسألتنا هذه.

7.4.3.2 العقدة بـ k قيمة متوسطة (K Means Clustering) :

بدايةً يجدر بالذكر أنّا نهدف من هذه الخوارزمية إلى فصل النقاط إلى مجموعتين، Normal وAnomaly، لذلك سنستخدم القيمة $2 = k$. على عكس الخوارزميات الأخرى، فإن هذه الخوارزمية غير خاضعة للإشراف، وبالتالي فإنّها لن تستخدم في تنبؤها قيم المتحول الهدف الموجودة في معطيات التدريب إلا لتحديد فئة كل من العندوين الناتجين، ولهذا فإن هذه الخوارزمية ستضيّع كمية كبيرة من المعلومات.

حسب الدراسة النظرية، فإن عمل هذه الخوارزمية مشابه لعمل خوارزمية K Nearest Neighbors، والتي تعتمد أيضاً على حساب أبعاد النقاط عن بعضها واعطاء أفضلية للنقاط الأقرب. وبما أن المجموعة ”All Features (Scaled)“ هي التي أعطت أفضل النتائج عند استخدام (K Nearest Neighbors)، فسنستخدم هذه المجموعة من أجل عملية التنبؤ هنا. يوضح الشكل (42) تغيرات دقة خوارزمية (K Means Clustering) عند تغيير حجم معطيات التدريب.



الشكل (42) تغيرات دقة خوارزمية (K Means Clustering) عند تغيير حجم معطيات التدريب

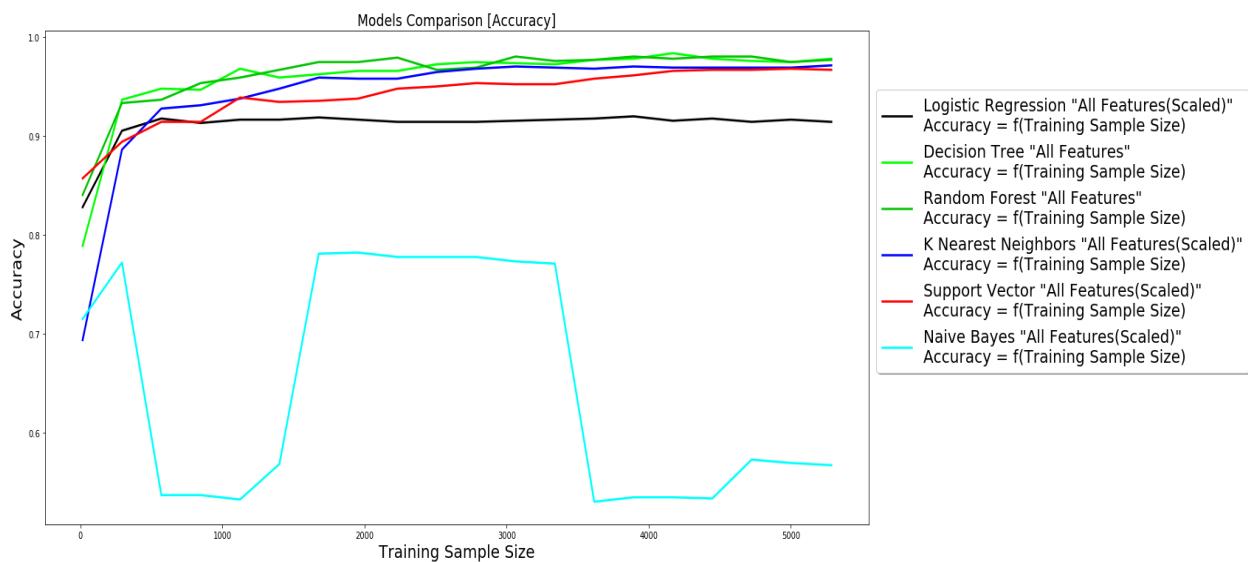
نلاحظ من الشّكل وصول المُودج إلى الحالة المستقرّة بعد حوالي 60000 عملية تدريب، ووصل إلى دقة تصل إلى 87%， ما يجعلها أقل الخوارزميّات دقةً "ذلك دون الأخذ بالحسبان نتائج خوارزميّة بايز التي وصلنا لنتيجة أنها غير صالحة لمسألتنا".

بملاحظة أنَّ هذه الخوارزميّة تتطلّب أكبر حجم من معطيات التدريب للوصول للنموذج المستقرّ، وأعطت أقل قيمة، فهذا يجعلها أسوأ الخيارات الممكّنة، وبالتالي لن نستخدم هذه الخوارزميّة إلا في حال لم يكن لدينا قيم المتحول الهدف في معطيات التدريب "وبالتالي يُفضّل ألا نستخدمها في مسأّلتنا هذه".

5.3.2 مقارنة أداء الخوارزميّات المُعتمدة:

قبل البدء بعمليّة المقارنة، نلاحظ أنَّ لدينا 3 حدود للاستقرار، فخوارزميّة شجرة القرار والغابة العشوائيّة وصلتا للاستقرار بعد 35000 عملية تدريب، خوارزميّة الانحدار اللوجستي وصلت للاستقرار بعد 5000 عملية تدريب، بينما وصلت خوارزميّة متّجه الدّعم وأقرب k جار للاستقرار بعد 2000 عملية.

يوضّح الشّكل (43) أداء كل من الخوارزميّات عندما يكون حجم معطيات التدريب ضمن المجال :[0, 5000]



الشّكل (43) دقة كل من الخوارزميّات عندما يكون حجم معطيات التدريب ضمن المجال [0, 5000]

نلاحظ من هذا الشّكل أنَّ دقة خوارزميّة شجرة القرار والغابة العشوائيّة تجاوزت دقة بقية الخوارزميّات بعد عدد قليل جدًا من عمليّات التدريب "أقل من 2000 عملية تدريب".
يوضّح الجدول (2) المقارنة بين أداء كل من هذه الخوارزميّات عندما يكون حجم معطيات التدريب المتوفر هو 2000، 5000، 35000.

الجدول 2- مقارنة نتائج خوارزميات تعلم الآلة

الخوارزمية	زمن التنبؤ عند التمودج المستقر	زمن التدريب للوصول للنموذج المستقر	الوصول للنموذج المستقر	أفضل مجموعة معطيات	الدقة 2000 عينة"	الدقة 5000 عينة"	الدقة 35000 عينة"	الدقة 60000 عينة"	الوصول للنموذج المستقر
LR	0.018s	0.180s	5000	All Features (Scaled)	91.2%	92%	92.00%	92.00%	98%
DT	0.021s	0.250s	35000	All Features	98%	98.5%	99.47%	99.47%	98.4%
RF	0.040s	0.195s	35000	All Features	98.4%	98.8%	99.39%	99.39%	97.27%
KNN	6.940s	0.046s	2000	All Features (Scaled)	97.27%	97.27%	97.27%	97.27%	96.35%
SV	0.816s	0.200s	2000	All Features (Scaled)	96.35%	96.35%	96.35%	96.35%	88.65%
K Means Clustering	0.060s	0.210s	60000	All Features (Scaled)	88.65%	88.02%	51.20%	88.00%	

نلاحظ من الجدول السابق تقارب أ زمنة التدريب بين الخوارزميات، ما عدا خوارزمية KNN التي كان لها زمن تدريب قصير سبياً، "أصغر بحوالي 5 مرات من المتوسط"، وذلك يطابق الدراسة النظرية، فعملية التدريب في هذه الخوارزمية هي الأبسط على الإطلاق، والتي تقصر على تخزين النقاط فقط. ويجد بالذكر أنَّ تدريب نموذجي شجرة القرار والغابة العشوائية أخذ أوقاتاً قريبة من نظائرها على الرغم من كون عدد العينات للوصول للاستقرار أكبر بحوالي 7 مرات من LR، وبحوالي 18 مرّة من KNN و SV، مما يدل على بطيء عملية تدريب هذه الخوارزميات مقارنة بهم.

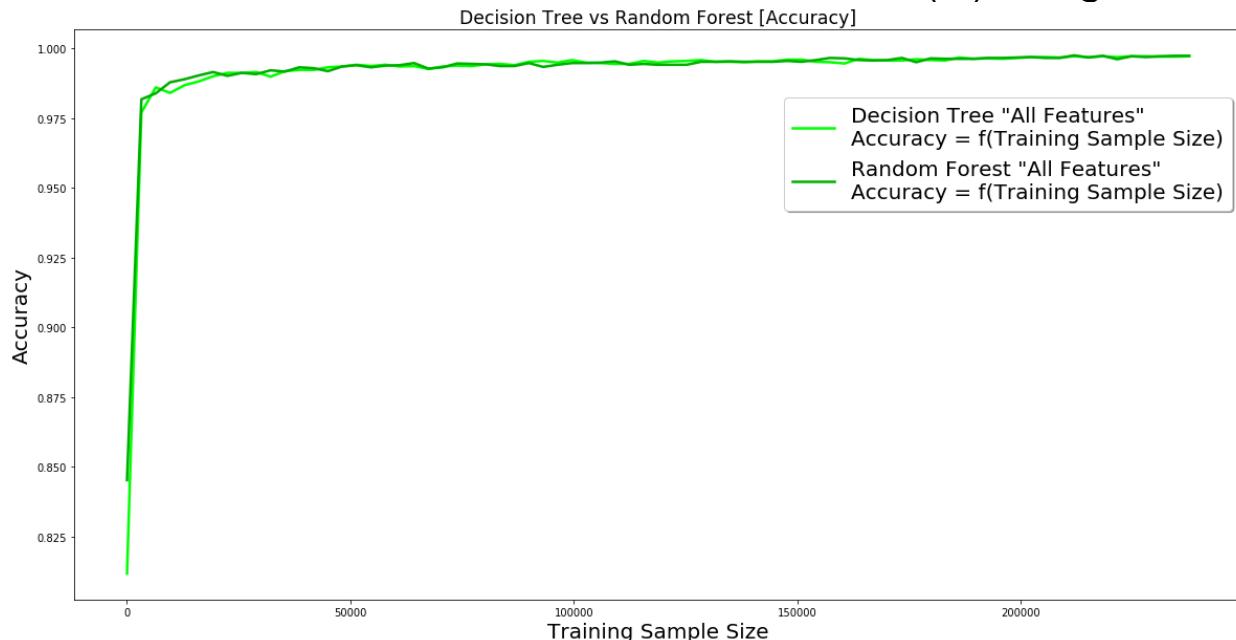
أما بالنسبة لأ زمنة التنبؤ، فنلاحظ تقارب جميع القيم ما عدا قيمة كل من الخوارزميتين SV و KNN، حيث كان زمن التنبؤ ل SV أكبر بحوالي 20 مرّة من متوسط قيمة الخوارزميات الأربع، وزمن التنبؤ ل KNN أكبر بحوالي 180 مرّة، مما يبين ببطء عملية التنبؤ لهاتين الخوارزميتين.

من الجدول السابق نلاحظ أنَّ خوارزميتي شجرة القرار والغابة العشوائية كان لهما أفضل دقة عند جميع قيم معطيات التدريب، ولم يحتاجا إلى تسوية قيم المعطيات أو لتهيئة المعطيات للوصول لهذه النتائج، كما أنَّ زمني التدريب والتنبؤ ليسا أكبر بكثير من مقابلاتها في بقية الخوارزميات، كما نلاحظ التقارب بين نتائج هاتين الخوارزميتين.

ممّا سبق نلاحظ أنَّ جميع المؤشرات، سواءً كانت الدقة، زمن التدريب، زمن التنبؤ، أو حجم معطيات التدريب المتوفر، فإنَّ خوارزميتي شجرة القرار والغابة العشوائية هما الخيار الأفضل لمسألتنا هذه.

في النهاية، وبسبب تقارب نتائج خوارزميتي شجرة القرار والغابة العشوائية، لنجري مقارنة بينهما لاتخاذ القرار النهائي.

يوضح الشّكل (44) المقارنة بين دقة كل من هاتين الخوارزميتين:



الشكل (44) مقارنة بين دقة خوارزمية شجرة القرارات وخوارزمية الغابة العشوائية

نلاحظ من الشّكل أَنَّه وبعد وصول النّموذجين إلى حالتهما المستقرّة أَنَّ النّتائج متقاربة جدًا ولا توجد أي أفضلية لأحد النّموذجين على الآخر.

وبالتالي يمكننا القول بالاعتماد على هذه النّتائج بأنَّ كلاً الخوارزميتين تعطّيات النّتائج الأمثلية. يجدر بالذكر أَنَّه وبحسب الدراسات النّظرية فإنَّ خوارزمية الغابة العشوائية أقلَّ تعرّضاً لمشكلة الـ “overfitting” التي سبق وتحذّثنا عنها، وبالتالي ففي حال استخدام معطيات أخرى مختلفة عن التي استخدمناها في عملتنا هذا، قد يكون من الأفضل استخدام نموذج الغابة العشوائية، لكن سنفترض حسب ما وجدنا من نتائج عملية بأنَّ كلَّ من هاتين الخوارزميتين لهما نفس الأداء.

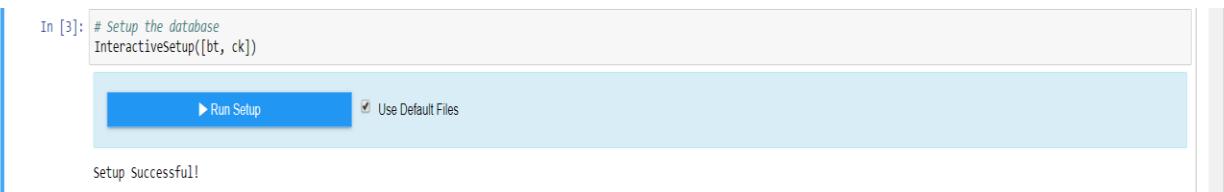
الفصل الثالث

واجهات التّخاطب مع المستخدم

اعتمدنا لبناء واجهات النّظام على أداة Jupyter Notebook و على مكتبة Widgets الموجودة في لغة Python.

يتم توليد كل من الواجهات بتشغيل الخلية الحاوية على الرّموز البرمجي الموافق لها، تتم عملية التشغيل هذه عن طريق تحديد الخلية ومن ثم نقر ”Shift + Enter“ على لوحة المفاتيح.

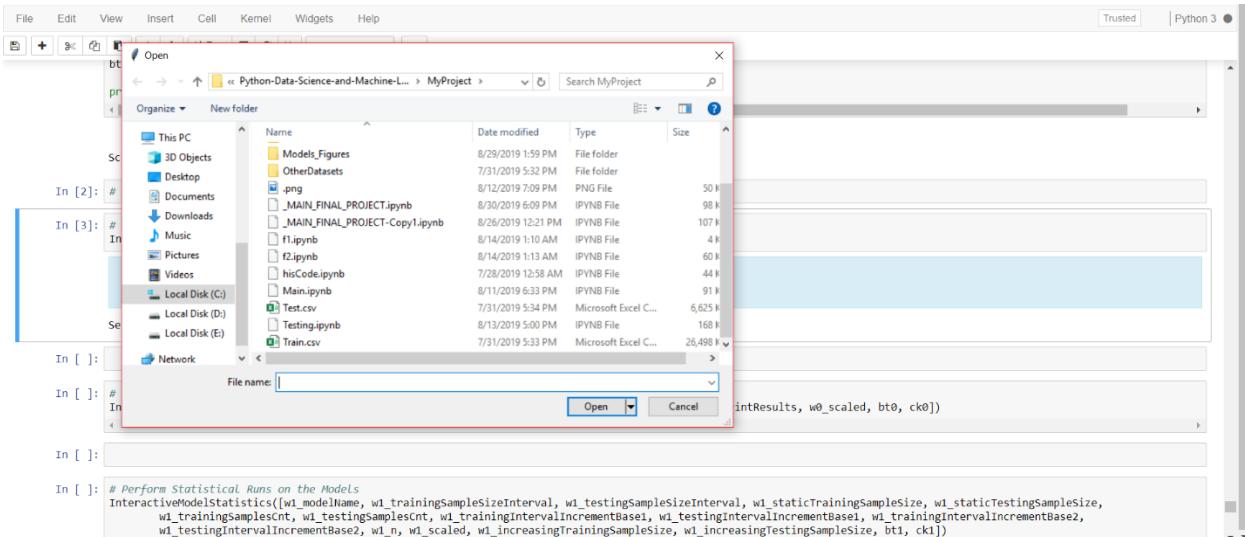
بدايةً يجب أن نقرأ ملفي التّدريب والاختبار، يتم ذلك باستخدام الواجهة التالية:



الشكل (45) واجهة ”Setup“ لاختيار ملفات قواعد المعطيات

بالضغط على زر ”Run Setup“ ستتم عملية قراءة الملفين إضافةً إلى إجراء جميع عمليات التّهيئه للمعطيات.

يسمح الخيار ”Use Default Files“ باستخدام الملفات الافتراضية والتي هي ”Test.csv“ و ”Train.csv“، وفي حال عدم اختيار هذا الخيار ستظهر واجهة لاختيار ملف التّدريب وثمّ أخرى لملف الاختبار، وتظهر الواجهة بالشكل التالي:

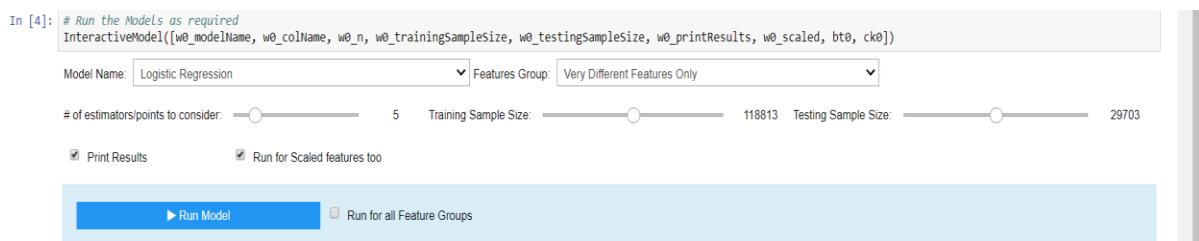


الشكل (46) عملية الاختيار التدريجية لملفات قواعد المطبيات

بعد اختيار الملفين "أو اختيار الخيار "Run Setup" (Use Default Files)" نضغط على زر "Run Setup" وبعد الانتهاء تظهر رسالة "Setup Successful" تدل على انتهاء العملية بنجاح.

بعد الانتهاء من عملية Setup أصبح الآن بإمكاننا تشغيل الواجهات الأخرى.

الآن ننتقل إلى واجهة "Run Model"، والتي تمكّنا من اختيار الخوارزمية التي نريد نطبيقها مع اختيار عدد من الخيارات ومن ثم استعراض النتائج حسب الرغبة. تظهر هذه الواجهة بالشكل التالي:



توجد العديد من الخيارات التي تؤمنها هذه الواجهة، وهي كالتالي:

: اسم الخوارزمية/الموديل المراد استخدامه. Model Name

: اسم مجموعة المطبيات المراد استخدامها. Features Group

of estimators to consider: هذا الخيار يحدّد عدد الأشجار في خوارزمية RF، ويحدّد قيمة المتحول k في خوارزمية KNN، وهو غير فعال في حال استخدام إحدى الخوارزميات الأخرى.

. حجم معطيات التدريب (عدد الأسطر). Training Sample Size

. حجم معطيات التنبؤ (عدد الأسطر). Testing Sample Size

: عند تفعيل هذا الخيار، يتم طباعة النتائج التي ولّدها هذا النموذج. Print Results

: في حال اختيار هذا الخيار، سيتم توليد نموذج لكل من مجموعات المعطيات الأربع الأولى. Run for all Feature Groups

: عند اختيار هذا الخيار مع الخيار السابق، سيتم توليد نماذج لمجموعات المعطيات جميعها، حتى تلك الموافقة لقيم بعد التسوية. Run for Scaled Features too

بعد اختيار الخيارات المطلوبة نضغط زر ”Run Model“ فيبدأ التنفيذ، وفي حال كان الخيار ”Print“ مفعّل، سيتم طباعى النتائج بالشكل التالي:

```
Model Name: Logistic Regression
Training Sample Size: 118813
TestingSampleSize: 101
-----
Very Different Features Only
Accuracy is: 0.6435643564356436
[[54 4]
 [32 11]]
      precision    recall   f1-score   support
          0       0.63      0.93      0.75      58
          1       0.73      0.26      0.38      43

   micro avg       0.64      0.64      0.64      101
   macro avg       0.68      0.59      0.56      101
weighted avg       0.67      0.64      0.59      101

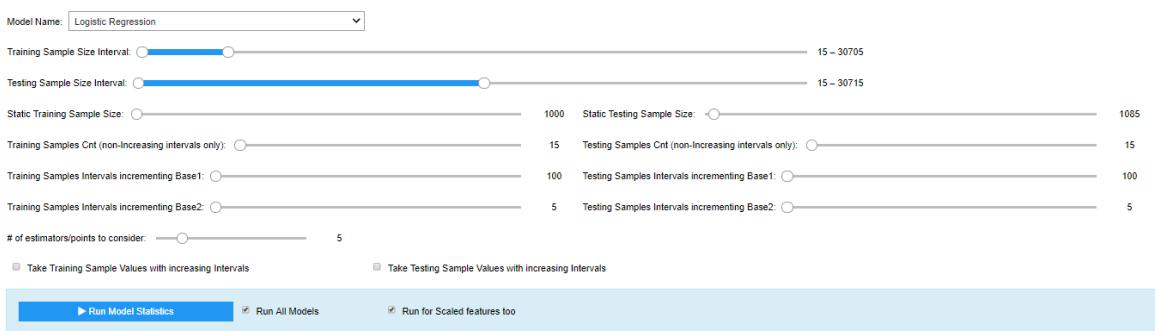
Training Time: 0.3422768910000116
Testing Time: 0.00378448499997062
Total Time: 0.34606137600000864

Normal/Anomaly Lables:
(0: 0), (1: 0), (2: 0), (3: 0), (4: 0), (5: 1), (6: 0), (7: 0), (8: 0), (9: 0)
(10: 0), (11: 0), (12: 0), (13: 0), (14: 1), (15: 0), (16: 0), (17: 0), (18: 1), (19: 0)
(20: 0), (21: 0), (22: 0), (23: 0), (24: 0), (25: 1), (26: 1), (27: 0), (28: 0), (29: 0)
(30: 0), (31: 0), (32: 0), (33: 0), (34: 1), (35: 0), (36: 0), (37: 0), (38: 0), (39: 0)
(40: 0), (41: 0), (42: 1), (43: 0), (44: 0), (45: 1), (46: 0), (47: 0), (48: 1), (49: 1)
(50: 0), (51: 0), (52: 0), (53: 1), (54: 0), (55: 0), (56: 0), (57: 0), (58: 0), (59: 0)
(60: 0), (61: 0), (62: 0), (63: 0), (64: 1), (65: 0), (66: 0), (67: 0), (68: 0), (69: 0)
(70: 0), (71: 0), (72: 0), (73: 0), (74: 0), (75: 0), (76: 0), (77: 0), (78: 0), (79: 0)
(80: 0), (81: 0), (82: 0), (83: 0), (84: 0), (85: 1), (86: 0), (87: 1), (88: 0), (89: 0)
(90: 0), (91: 1), (92: 0), (93: 0), (94: 0), (95: 0), (96: 0), (97: 0), (98: 0), (99: 0)
(100: 0)
.....
#####
#####
```

الشكل (48) صيغة النتائج عند استخدام واجهة ”Run Model“

كما هو موضح بالشكل، فإن النتائج المطبوعة تتضمن معلومات إحصائية ألا وهي precision، recall و f1_score.

الواجهة التالية تمكّننا من إجراء عدد من عمليات توليد النماذج واختبارها، وذلك لإجراء مقارنات على أداء الخوارزمية عند تغيير أحجام معطيات التدريب والتنبؤ:



الشكل (49) واجهة توليد النماذج المتعددة تحضيراً لعمليات المقارنة

توجد العديد من الخيارات التي تؤمنها هذه الواجهة، وهي كالتالي:

اسم الخوارزمية/النموذج المراد استخدامه، " يتم استخدام جميع المجموعات". Model Name

المجال الذي سيتم منهأخذ قيم حجوم معطيات التدريب. Training Sample Size Interval

المجال الذي سيتم منهأخذ قيم حجوم معطيات الاختبار/التنبؤ. Testing Sample Size Interval

قيمة حجم معطيات التدريب المثبتة عندما تتغير قيم معطيات الاختبار. Static Training Sample Size

قيمة حجم معطيات الاختبار المثبتة عندما تتغير قيم معطيات التدريب. Static Testing Sample Size

عدد العينات ليتمأخذها من مجال القيم المحدد في الخيار (2)، حيث يتمأخذ القيم بشكل منظم على المجال، وتعبر هذه القيم عن حجوم معطيات التدريب التي سيتم توليد النموذج على أساسها، يُفعّل هذا الخيار فقط عند عدم اختيار الخيار (13).

عدد العينات ليتمأخذها من مجال القيم المحدد في الخيار (3)، حيث يتمأخذ القيم بشكل منظم على المجال، وتعبر هذه القيم عن حجوم معطيات الاختبار التي سيتم اختيار النموذج عليها يُفعّل هذا الخيار فقط عند عدم اختيار الخيار (14).

يُفعّل هذا الخيار فقط عند اختيار الخيار (13). تحدّد قيمة هذا الحقل معدّل التزايد الخطّي في الفروق بين قيم العينات المتتالية المأخوذة لمعطيات التدريب، عندما تكون قيمة هذا الحقل غير معدومة تصبح العينات أشبه بعينات من تابع كثير حدودي من الدرجة الثانية.

قيمة هذا الحقل معدّل التّزايد الخطّي في الفروق بين قيم العيّنات المتّالية المأخوذة لمعطيات الاختبار، عندما تكون قيمة هذا الحقل غير معدومة تصبح العيّنات أشبه بعيّنات من تابع كثير حدودي من الدرجة الثانية.

قيمة هذا الحقل معدّل التّزايد من الدرجة الثانية في الفروق بين قيم العيّنات المتّالية المأخوذة لمعطيات التّدريب، عندما تكون قيمة هذا الحقل غير معدومة تصبح العيّنات أشبه بعيّنات من تابع كثير حدودي من الدرجة الثالثة.

قيمة هذا الحقل معدّل التّزايد من الدرجة الثانية في الفروق بين قيم العيّنات المتّالية المأخوذة لمعطيات الاختبار، عندما تكون قيمة هذا الحقل غير معدومة تصبح العيّنات أشبه بعيّنات من تابع كثير حدودي من الدرجة الثالثة.

#: هذا الخيار يحدّد عدد الأشجار في خوارزمية RF، ويحدّد قيمة المتحول k في خوارزمية KNN، وهو غير فعال في حال استخدام إحدى الخوارزميات الأخرى.

Take Training Sample Values with Increasing Intervals: في حال اختيار هذا الخيار، سيتم أخذ عيّنات حجوم معطيات التّدريب بشكل تابع من الدرجة الثانية أو الثالثة عوض عن أخذ القيم بشكل منتظم "تابع خطّي".

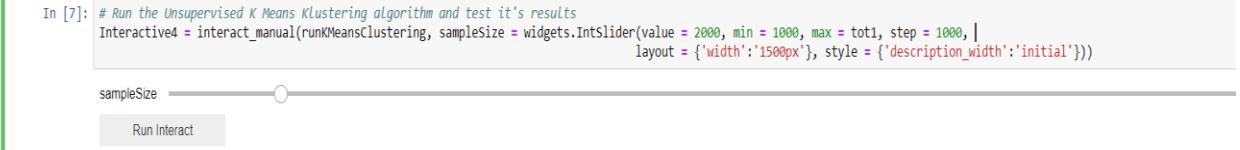
Take Testing Sample Values with Increasing Intervals: في حال اختيار هذا الخيار، سيتم أخذ عيّنات حجوم معطيات الاختبار بشكل تابع من الدرجة الثانية أو الثالثة عوض عن أخذ القيم بشكل منتظم "تابع خطّي".

Run All Models: في حال اختيار هذا الخيار، سيتم تشغيل جميع الخوارزميات على الخيارات المحدّدة عوض عن الخوارزمية التي تم اختيارها في الخيار (1) فقط.

Run for Scaled Features too: في حال اختيار هذا الخيار، سيتم استخدام المجموعات الموافقة للقيم بعد التّسوية أيضاً.

بعد اختيارات الخيارات المطلوبة نضغط زر "Run Statistics" فيتم القيام بالوظيفة المطلوبة.

توجد لدينا أيضاً واجهة بسيطة للتعامل مع خوارزمية K Means Clustering وهي بالشكل التالي:

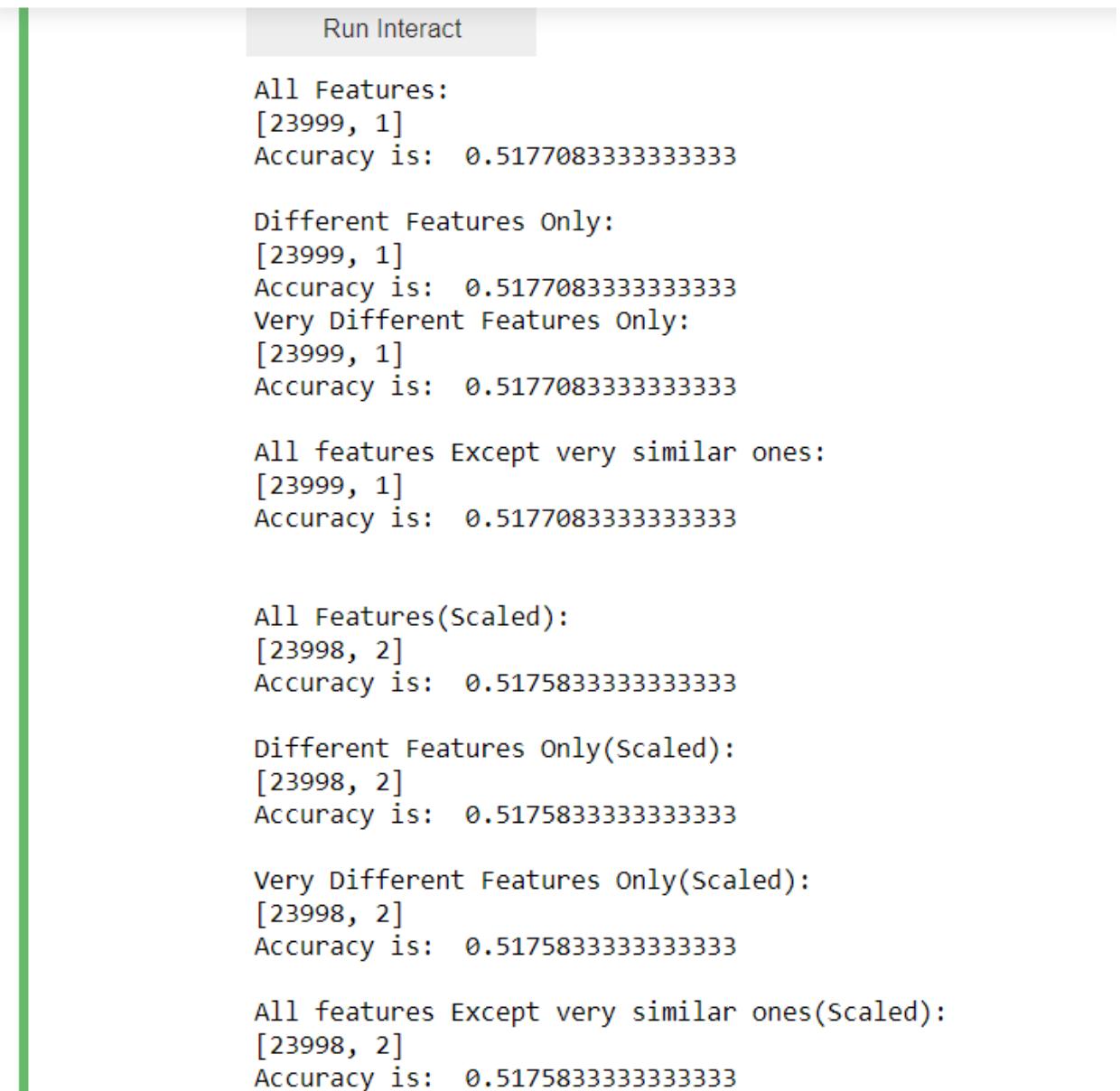


```
In [7]: # Run the Unsupervised K Means Klustering algorithm and test it's results
Interactive4 = interact_manual(runKMeansClustering, sampleSize = widgets.IntSlider(value = 2000, min = 1000, max = tot1, step = 1000, | layout = {'width': '1500px'}, style = {'description_width': 'initial'}))
```

sampleSize

الشكل (50) واجهة التعامل مع خوارزمية K Means Clustering

الخيار الوحيد المتاح هو حجم معلمات التدريب، وبعد الضغط على زر "Run Interact" يتم طباعة النتائج لكل من المجموعات الثمانية "ذات القيم قبل وبعد التسوية"، تظهر النتائج بالشكل التالي:



```
Run Interact
```

```
All Features:
[23999, 1]
Accuracy is: 0.5177083333333333

Different Features Only:
[23999, 1]
Accuracy is: 0.5177083333333333
Very Different Features Only:
[23999, 1]
Accuracy is: 0.5177083333333333

All features Except very similar ones:
[23999, 1]
Accuracy is: 0.5177083333333333

All Features(Scaled):
[23998, 2]
Accuracy is: 0.5175833333333333

Different Features Only(Scaled):
[23998, 2]
Accuracy is: 0.5175833333333333

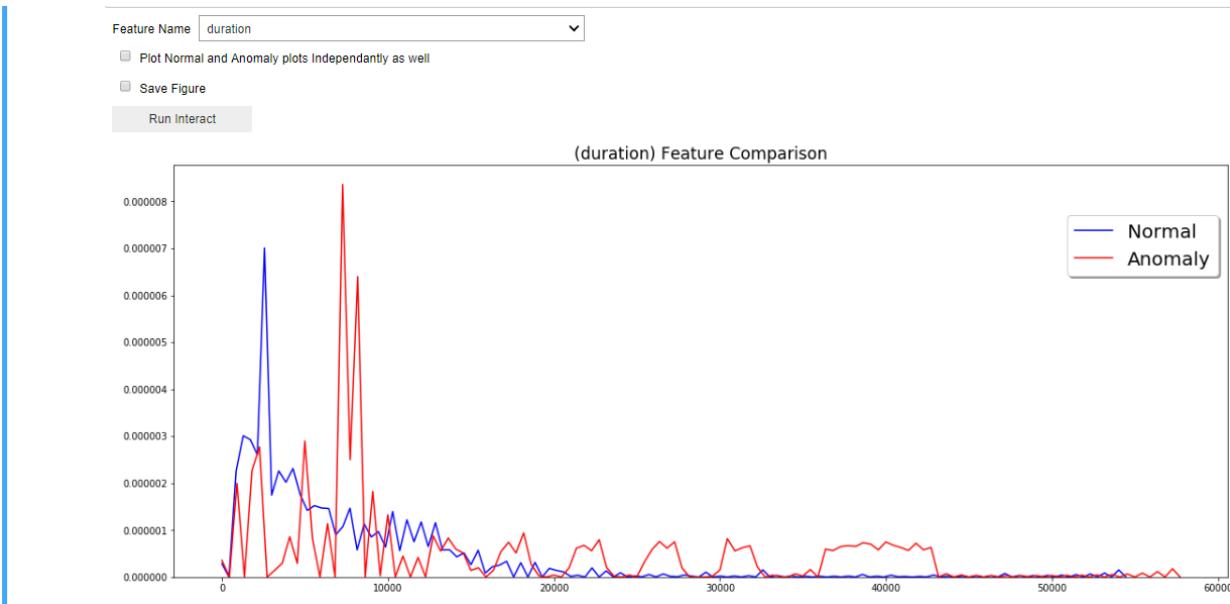
Very Different Features Only(Scaled):
[23998, 2]
Accuracy is: 0.5175833333333333

All features Except very similar ones(scaled):
[23998, 2]
Accuracy is: 0.5175833333333333
```

الشكل (51) صيغة النتائج عند تشغيل واجهة K Means Clustering

حيث تظهر لكل مجموعة حجم كل من العنوبيين المولدين ودقة النتائج.

تظهر بعد ذلك الواجهة التالية، والتي تقوم برسم المنحنيات التي توضح أثر كل من معاملات المعطيات على قيم المتحول الهدف:



الشكل (52) واجهة رسم مخططات توضيح أثر تغيير قيم المعاملات على قيم المتحول الهدف

توجد عدّة خيارات في هذه الواجهة، وهي كالتالي:

اسم المعامل المراد دراسته .Feature Name

عند اختيار هذا الخيار ، يتم رسم كل من المنحنيين على مخطط مستقل بعد رسم مخطط المقارنة .Plot Normal and Anomaly plots independently as well

عند اختيار هذا الخيار ، يتم حفظ المخطط بصيغة png . ضمن مجلد تم توليه مسبقاً ضمن البرنامج اسمه "Features_Figures" .Save Figure

لدينا بعد ذلك الواجهة التالية ، والتي تقوم بحسب الخيارات المختارة برسم المخطط المطلوب ، تتميز هذه الواجهة بتوسيع المخططات بشكل تفاعلي ، حيث أن كل عملية تغيير في أحد الخيارات ستقوم بشكل تلقائي برسم المخطط الجديد دون الحاجة للضغط على زر تشغيل . تظهر هذه الواجهة بالشكل التالي :

Model Name:	Logistic Regression
Function:	Accuracy = f(Training Sample Size)
Features Group:	Very Different Features Only
Plot Type:	Line Plot
Marker Type:	o
Marker Size:	<input type="range" value="150"/>
Plot's Color:	black <input checked="" type="checkbox"/>
Line Width:	<input type="range" value="2.50"/>
X Label:	"Default"
Y Label:	"Default"
Plot's Title:	"Default"
Label Font Size:	<input type="range" value="20"/>
Title Font Size:	<input type="range" value="20"/>
Figure Size (X axis):	<input type="range" value="30"/>
Figure Size (Y axis):	<input type="range" value="8"/>
<input type="checkbox"/> Use Iplots	

الشكل (53) واجهة رسم المخططات التفاعلية لتغيرات أداء الخوارزميات بتغيير الخيارات المطلوبة

توجد العديد من الخيارات التي تؤمنها هذه الواجهة، وهي كالتالي:

اسم النموذج المراد استعراض نتائجه. Model Name

التابع المراد رسم مخططه البياني، توجد العديد من الخيارات أهمها تغيرات الدقة مع حجم معطيات التدريب. Function

مجموعة المعطيات المراد استخدامها. Features Group

نوع المخطط المطلوب، يوجد نوعين متاحين هما المخطط النقاطي “scatter plot” والمخطط الخطى “line plot”.

شكل الرموز المستخدمة في مخطط “scatter plot”， توجد العديد من الخيارات وهي شكل المعين، النقطة، إشارة ‘x’، والمرربع.

حجم الرموز المستخدمة في مخطط “scatter plot” . Marker Size

لون المخطّط: Plot's Color

عرض الخطّ في مخطّط: Line Width.

X Label: تسمية محور السينات، في حال كانت قيمة الحقل هي "Default" مع إشارتي التنصيص، عندها يتم وضع الاسم الافتراضي والذي يتم أخذه من التابع الذي تم اختياره في الخيار (2).

Y Label: تسمية محور العينات، في حال كانت قيمة الحقل هي "Default" مع إشارتي التنصيص، عندها يتم وضع الاسم الافتراضي والذي يتم أخذه من التابع الذي تم اختياره في الخيار (2).

Plot's Title: عنوان الشكل، في حال كانت قيمة الحقل هي "Default" مع إشارتي التنصيص، عندها يتم وضع الاسم الافتراضي والذي يتم أخذه من اسم التموزج المختار واسم المجموعة المختارة من الخيارات (1) و(2) على الترتيب.

حجم الخط لأسماء المحاور: Label Font Size

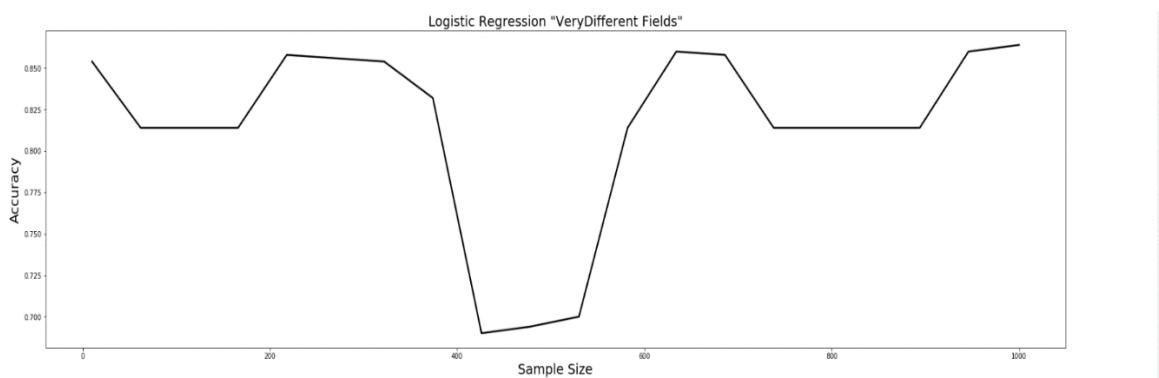
حجم خط عنوان الشكل: Title Font Size

حجم الشكل على طول محور السينات: Figure Size (X axis)

حجم الشكل على طول محور العينات: Figure Size (Y axis)

Use Iplots: في حال اختيار هذا الخيار، سيتم استخدام مخطوطات مكتبة Cufflinks عوض عن Seaborn والتي تتميز بكونها تفاعلية.

في كل لحظة سيوجد مخطّط موافق لقيم الخيارات ظاهراً بالشكل التالي دون الحاجة للضغط على أي زر تشغيل:



الشكل (54) شكل المخطوطات الناتجة عند تشغيل واجهة رسم المخطوطات التفاعلية

وفي النهاية تظهر لدينا الواجهة التي تسمح لنا برسم عدّة مخططات من اختيارنا فوق بعضها البعض لإجراء عملية المقارنة فيما بينها. تظهر هذه الواجهة بالشكل التالي:

Model Name: Logistic Regression

Function: Accuracy = $f(\text{Training Sample Size})$

Features Group: Very Different Features Only

Plot Type: Line Plot

Marker Type: o

Marker Size: 150

Plot's Color: black

Line Width: 2.50

Clear Previous Plots

Save Figure / Draw New Plot

Figure Title:

Run Interact Figure Title

الشكل (55) واجهة رسم مخططات المقارنة

توجد العديد من الخيارات التي تؤمنها هذه الواجهة، وهي كالتالي:

اسم النموذج المراد استعراض نتائجه. Model Name

التابع المراد رسم مخططه البياني، توجد العديد من الخيارات أهمها تغييرات الدقة مع حجم معطيات التدريب. Function

مجموعة المعطيات المراد استخدامها. Features Group

نوع المخطط المطلوب، يوجد نوعين متاحين هما المخطط النقاطي “scatter plot” والمخطط الخطّي “line plot”. Plot Type

شكل الرموز المستخدمة في مخطط “scatter plot”， توجد العديد من الخيارات وهي شكل المعين، النقطة، إشارة ‘x’، والمرربع. Marker Type

حجم الرموز المستخدمة في مخطط “scatter plot”. Marker Size

لون المخطط. Plot's Color

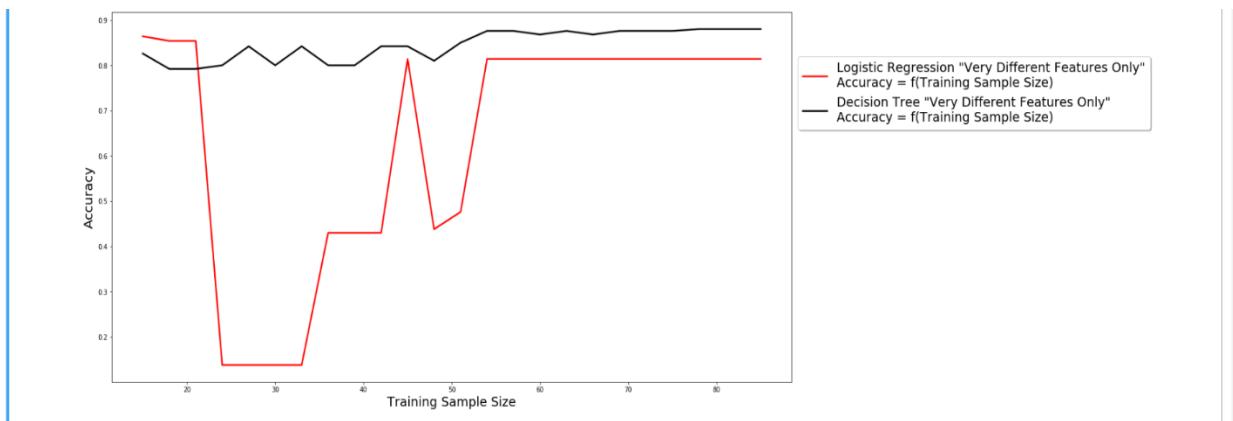
عرض الخط في مخطط .“line plot” Line Width

Clear Previous Plots: في حال تفعيل هذا الخيار، سيتم محو جميع المخططات المرسومة مسبقاً قبل رسم المخطط الجديد.

Save Figure/Draw New Plot: في حال عدم تفعيل هذا الخيار، سيتم رسم المخطط الحالي، أما في حال تفعيله، فلن يتم الرسم، بل سيتم حفظ المخطط الحالي المكون من عدّة منحنيات "أو منحنٍ واحد" بصيغة png. في مجلد تم إنشاؤه ضمن البرنامج اسمه "Models Figures".

عنوان المخطط: Figure Title

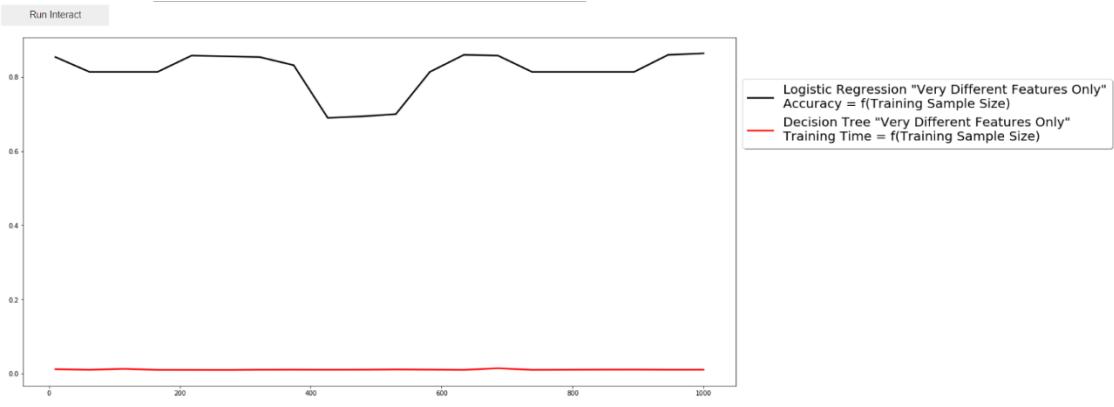
تظهر المخططات الناتجة بالشكل التالي:



الشكل (56) الأشكال الناتجة عن تشغيل واجهة رسم مخططات المقارنة

نلاحظ وجود عنوان تفسيري في المخطط يوضح شكل المنحني ولونه، اسم الخوارزمية المستخدمة ومجموعة المعطيات المستخدمة، إضافة إلى التابع المرسوم.

يجدر بالذكر أن أسماء محوري البيانات والعينات يتم استنتاجها من أسماء التوابع المرسومة، وفي حال كانت التسميات غير متطابقة، تترك المحاور دون تسمية، يوضح الشكل التالي هذه الحالة:



الشكل (57) اختفاء تسميات المحاور عند عدم توافق التسميات بين مخططات المقارنة

نلاحظ أنَّ أحد التابعين المرسومين هوتابع للدقة بدلاًلة حجم معطيات التدريب، بينما التابع الآخر فهوتابع لزمن التدريب بدلاًلة حجم معطيات التدريب. وبما أنَّ هاذين التابعين لا يعبران عن نفس المتحولات، تمَّ حذف تسميات المحاور.

الخاتمة

في وقتنا المعاصر، باتت مسألة تصنيف البيانات المتبادلة عبر الشبكات مسألة صعبة، وباتت التقنيات التقليدية للكشف عن البرامج الخبيثة والاختراق عاجزة لأداء هذه المهمة. في حين أظهر مجال التعلم الآلي نتائج جيدة جدًا تعجز التقنيات التقليدية عن بلوغها، وأصبح بالإمكان استخدام هذه التقنيات للدفاع عن الجريمة الإلكترونية، ولهذا السبب قررنا أن نقوم بإنشاء أداة بسيطة نسبياً لقيام بهذه المهمة.

قمنا في هذا العمل بإجراء دراسة نظرية عن خوارزميات تعلم الآلة المستخدمة في عملنا لتوضيح آلية عمل كل منها بشكل مختصر، وإبراز حسناتها وسعيّاتها، ما يعطي فكرة عامة عن كيفية وصول كل من هذه الخوارزميات للنتائج التي طرحتها، كما تفيد الدراسة النظرية بأخذ فكرة مبدائية عن الخيار المناسب للمسألة المطروحة.

بعد ذلك بدأنا بالجانب العملي، حيث قمنا أولاً بعملية إعداد وتهيئة وتسوية للبيانات التي لدينا وذلك بغية تحسين النتائج التي سنحصل عليها من خوارزميات تعلم الآلة. بعد ذلك قمنا بدراسة تحليلية لكل من الخوارزميات المتاحة لمعرفة الطريقة الأفضل لاستخدام كل منها، والحسنات والسيئات التي تقابل استخدامها. ثم قمنا بمقارنة نتائج كل من الخوارزميات وتوصلنا إلى أن استخدام خوارزميتي شجرة القرارات والغابة العشوائية كان الخيار الأمثل بالنسبة لجميع المعاملات المدروسة، والتي هي دقة النتائج، زمن التدريب، وزمن التنبؤ وحجم معطيات التدريب المتأخر. حيث تجاوزت دقة هاتين الخوارزميتين عتبة الـ 99% بعد 35000 عملية تدريب، بينما تجاوزت الدقة عتبة الـ 98% عند استخدام عدد من معطيات التدريب ضمن المجال [35000, 35000]، وهي دقة تجاوزت ما توصلت إليه بقية الخوارزميات في كل من هذه الحالات. أما من ناحية زمن التدريب، فإن هاتين الخوارزميتين كان لهما زمن تدريب قصير نسبياً، فلم تتطلب أي من الخوارزميات الأخرى زمن تدريب أقل بكثير من هاتين الخوارزميتين، والأمر ذاته ينطبق على زمن التنبؤ.

على الرغم من أن الأداة التي أنشأناها كانت تتيح للمستخدم اختيار العديد من الخيارات المتعلقة بآلية إجراء عملية الفرز، إلا أنه كان بالإمكان إضافة ديناميكية أكثر للأداة للسماح بالقيام بالمزيد من العمليات، وقد عجزنا عن فعل ذلك نظراً لضيق الوقت المتأخر مقارنة بالوقت اللازم لبناء نظام أكثر ديناميكية.

نورد فيما يلي اقتراحات لآفاق مستقبلية:

- 1- توسيع النظام ليشمل عدداً أكبر من خوارزميات تعلم الآلة.
- 2- السماح باستخدام عدد أكبر من صيغ الملفات مثل ملفات الإكسل "xls" وغيرها.
- 3- السماح باستخدام قواعد معطيات أكثر تنوعاً.
- 4- تأمين طريقة تسمح للمستخدم بالتحكم بعملية تهيئة المعطيات وتسوية قيمها إن رغب.

- 5- السماح للمستخدم بتقسيم الحقول إلى مجموعات بحسب الرغبة عوضاً عن القيام بذلك بشكل مباشر وثابت ضمن البرنامج فقط.
- 6- تأمين طريقة تسمح بعملية المقارنة بين نتائج الخوارزميات بشكل آلي وإعطاء اقتراحات للمستخدم عن الخيارات المفضلة التي ستعطي الأداء الأقرب لرغبته.
- 7- بناء تطبيق مستقل عن أداة Jupyter Notebook لعرض واجهات التخاطب بين المستخدم والبرنامج "تطبيق Android مثلًا".
- 8- بناء أداة قادرة على التقاط الطرود الشبكية ومعالجتها واستخراج السمات منها لتتم بعدها عملية التنبؤ بطبيعة الطرود بشكل مباشر أثناء العمل على الشبكة.

الملاحق

الملحق آ

قوانين حساب الدقة "Accuracy" ، "Precision" ، "Recall"

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

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

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

$$f1_{score} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

.TP: عدد التنبؤات الصحيحة ذات القيمة true.

.TN: عدد التنبؤات الصحيحة ذات القيمة false.

.FP: عدد التنبؤات الخاطئة ذات القيمة true.

.FN: عدد التنبؤات الخاطئة ذات القيمة false.

يعبر Accuracy عن دقة النتائج بشكل عام. بينما يعبر Recall عن اكتمال النتائج التي كانت قيمتها true، وهذا يعني في مسألتنا هذه نسبة عدد الطرود الخبيثة المكتشفة إلى عدد الطرود الخبيثة الكلية. ويعبر Precision عن دقة النتائج ذات القيمة true، حيث يعبر عن نسبة عدد الطرود الخبيثة المكتشفة إلى عدد الطرود التي تم "الحكم عليها" بأنها خبيثة.

الملحق بـ

الكود البرمجي

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
get_ipython().run_line_magic('matplotlib', 'inline')
import timeit
from plotly import __version__
from plotly.offline import download_plotlyjs, init_notebook_mode,
plot, iplot

import cufflinks as cf

# For Notebooks
init_notebook_mode(connected=True)

# For offline use
cf.go_offline()

import warnings
warnings.filterwarnings('ignore')

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import classification_report, confusion_matrix

# Machine Learning Algorithms we will be using:
from sklearn.linear_model import LogisticRegression
```

```

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.cluster import KMeans
from sklearn.naive_bayes import GaussianNB

from sklearn.cluster import KMeans

from ipywidgets import widgets
from ipywidgets.widgets import Label, FloatProgress, FloatSlider,
Button, Layout, HBox, VBox
from IPython.display import display
import bqplot as bq
import threading
from IPython.display import display
from IPython.html.widgets import *
from IPython.core.display import display, HTML

from tkinter import Tk
from tkinter.filedialog import askopenfilename

import os

# Instructions to make the width of the cells cover the width of the
screen

Tk().withdraw() # we don't want a full GUI, so keep the root window
from appearing

display(HTML("<style>.container { width:100% !important; }</style>"))

```

```

# Creating the files in which figures will be saved, only if they
don't already exist

if not os.path.exists('Models_Figures'):
    os.makedirs('Models_Figures')

if not os.path.exists('Features_Figures'):
    os.makedirs('Features_Figures')


# the titles of the feature groups "in a user friendly format"
titles = ['Very Different Features Only', 'Different Features Only',
'All Features Except Very Similar Ones',
    'All features', 'Very Different Features Only(Scaled)',
'Different Features Only(Scaled)',

    'All Features Except Very Similar Ones(Scaled)', 'All
Features(Scaled)']


# the names of the supervised Machine Learning(ML) algorithms used in
this project

modelsNames = ['Logistic Regression', 'Decision Tree', 'Random
Forest', 'K Nearest Neighbors',
    'Support Vector', 'Naive Bayes']


# Names of the dataframe columns, each column (except the first one)
correspond to a feature group

colNames = ['SampleSize', 'VeryDifferent', 'Different',
'NotVerySimilar', 'All',
    'VeryDifferent(Scaled)',
'Different(Scaled)', 'NotVerySimilar(Scaled)', 'All(Scaled)']


# The names of features in the very_different group

very_different = ['src_bytes', 'dst_bytes', 'num_compromised',
'num_root']


# The names of the features in the different group

```

```

different = ['serror_rate', 'srv_serror_rate', 'dst_host_srv_count',
'dst_host_serror_rate',
'dst_host_srv_serror_rate'] + very_different

# The names of the features in the very_similar group
very_similar = ['land', 'wrong_fragment', 'root_shell',
'is_host_login', 'is_guest_login']

# The names of the dataframes generated when running a Model, each
dataframe clarifies the relation between 2 parameters, the
training/testing sample size,
# and the accuracy/training time/testing time/total time of the model
dataFramesNames = ['Accuracy = f(Training Sample Size)', 'Training
Time = f(Training Sample Size)',

                    'Testing Time = f(Training Sample Size)', 'Total
Time = f(Training Sample Size)',

                    'Accuracy = f(Testing Sample Size)', 'Training Time
= f(Testing Sample Size)',

                    'Testing Time = f(Testing Sample Size)', 'Total
Time = f(Testing Sample Size)']

# The names of the dataframes "in a user friendly format"
colNames2 = ['Very Different Features Only', 'Different Features
Only', 'Not Very Similar Features Only', 'All Features',
'Very Different Features Only(Scaled)', 'Different
Features Only(Scaled)',

'Not Very Similar Features Only(Scaled)', 'All
Features(Scaled)']

# Declaring the global variables which will be used by one or more
function, defining global variables provides us with shared variables
amongst multiple functions,
# it also allows us to use the values of these variables wherever we
want in our code

```

```

bt=0; ck=0; w_modelName=0; w_dataframeName=0; w_colName=0;
w_plotType=0; w_markerType=0; w_color=0; w_xLabel=0; w_yLabel=0;
w_title=0; w_markerSize=0; w_lineWidth=0;

w_labelFontSize=0; w_titleFontSize=0; w_figSizeX=0; w_figSizeY=0;
w_iplot=0; tot1=0; tot2=0;

w2_modelName=0; w2_dataframeName=0; w2_colName=0; w2_plotType=0;
w2_markerType=0; w2_color=0; w2_markerSize=0; w2_lineWidth=0;
w2_clear=0;

bt0=0; ck0=0; w0_modelName=0; w0_colName=0; w0_n=0;
w0_trainingSampleSize=0; w0_testingSampleSize=0; w0_printResults=0;
w0_scaled=0; w0_testOnly=0;

width1=0; w1_modelName=0; w1_trainingSampleSizeInterval=0;
w1_testingSampleSizeInterval=0; w1_staticTrainingSampleSize=0;

w1_staticTestingSampleSize=0; w1_trainingSamplesCnt=0;
w1_testingSamplesCnt=0; w1_trainingIntervalIncrementBase1=0;
w1_testingIntervalIncrementBase1=0;

w1_trainingIntervalIncrementBase2=0;
w1_testingIntervalIncrementBase2=0; w1_n=0; w1_scaled=0;
w1_increasingTrainingSampleSize=0; w1_increasingTestingSampleSize=0;
bt1=0; ck1=0;

lastFileNum=-1;

```

```

def setup(my_train_dataset = 'Train.csv', my_test_dataset =
'Test.csv', interactive_pick = False):

    """Description:

        this function reads two csv files passed as parameters, and then
        reformats the data into usable formats

```

Parameters:

my_train_dataset: the dataset you wish to train your models by

my_test_dataset: the dataset you wish to test your models on

interactive_pick: if set to True, then an interactive GUI appears that allows you to visually select the files you want

"""

```

# declaring the globan varibales used in this function

    global train_dataset, test_dataset, col, service_types,
protocol_types, flag_types, train_features, train_labels

    global test_features, test_labels, scaled_train_features,
scaled_test_features, att_names, numeric_data, very_different

    global different, very_similar, train_features_very_different,
train_features_different, train_features_not_very_similar

    global test_features_very_different, test_features_different,
test_features_not_very_similar

    global scaled_train_features_very_different,
scaled_train_features_different,
scaled_train_features_not_very_similar

    global scaled_test_features_very_different,
scaled_test_features_different, scaled_test_features_not_very_similar

    global titles, train_features_categories,
test_features_categories, tot1, tot2, modelsNames, colNames,
models_num, model

    global testAccuracyDatafram, testTrainingTimeDataframe,
testTestingTimeDataframe, testTotalTimeDataframe

    global trainAccuracyDatafram, trainTrainingTimeDataframe,
trainTestingTimeDataframe, trainTotalTimeDataframe

    global dataFrames, dataFramesNames, colNames2


train_file = ''; test_file = ''

if interactive_pick == False:

    train_file = my_train_dataset

    test_file = my_test_dataset

else:

    train_file = askopenfilename()

    test_file = askopenfilename()

# reading the files and storing them in a dataframe in system's
memory

```



```
# generating a scaled version of our data, meaning that all fields  
values will be fitted into the same range of values, maintaining the  
relative  
  
# relation between values, for example, if we have 2 fields with  
values [1, 2, 4] and [5000, 15000, 20000]  
  
# then the scaled values will be (for example, if the common range  
is [2, 8]): [2, 4, 8] and [2, 6, 8]  
  
# we do this to see the effect of scaling data on the performance  
of our ML algorithms  
  
scaler = StandardScaler()  
scaler.fit(train_features)  
scaled_train_features = scaler.transform(train_features)  
  
  
scaled_train_features = pd.DataFrame(scaled_train_features,  
columns = train_features.columns)  
  
  
scaler = StandardScaler()  
scaler.fit(test_features)  
scaled_test_features = scaler.transform(test_features)  
  
  
scaled_test_features = pd.DataFrame(scaled_test_features, columns  
= test_features.columns)  
  
  
att_names = train_features.columns  
numeric_data = train_dataset.drop(axis = 1, columns = ['service',  
'protocol_type', 'flag', 'class'])  
  
#.....  
.....  
  
train_features_very_different = train_features[very_different]
```

```
test_features_very_different = test_features[very_different]

train_features_different = train_features[different]
test_features_different = test_features[different]

train_features_not_very_similar =
train_features.drop(very_similar, axis = 1)

test_features_not_very_similar = test_features.drop(very_similar,
axis = 1)

scaled_train_features_very_different =
scaled_train_features[very_different]

scaled_test_features_very_different =
scaled_test_features[very_different]

scaled_train_features_different = scaled_train_features[different]
scaled_test_features_different = scaled_test_features[different]

scaled_train_features_not_very_similar =
scaled_train_features.drop(very_similar, axis = 1)

scaled_test_features_not_very_similar =
scaled_test_features.drop(very_similar, axis = 1)

#.....................................
.....
# creating a global List of the train/test feature groups
train_features_categories = [train_features_very_different,
train_features_different, train_features_not_very_similar,
```

```

            train_features,
scaled_train_features_very_different, scaled_train_features_different,
                     scaled_train_features_not_very_similar,
scaled_train_features]

test_features_categories = [test_features_very_different,
test_features_different, test_features_not_very_similar,
                           test_features,
scaled_test_features_very_different, scaled_test_features_different,
                     scaled_test_features_not_very_similar,
scaled_test_features]

#.....  

.....  

tot1 = len(train_features)  

tot2 = len(test_features)

#.....  

.....  

# Number of Models to store in system memory, each one correspond
to one of our ML algorithm, also each model has 2 versions, the
scaled, and the unscaled version

models_num = len(modelsNames)
models_num *= 2
model = [0] * models_num

# creating global dataframes, each represents the relation between
sample size and the desired parameter as described later in the code

trainAccuracyDatafram = [0] * models_num
```

```

trainTrainingTimeDataframe = [0] * models_num
trainTestingTimeDataframe = [0] * models_num
trainTotalTimeDataframe = [0] * models_num

testAccuracyDatafram = [0] * models_num
testTrainingTimeDataframe = [0] * models_num
testTestingTimeDataframe = [0] * models_num
testTotalTimeDataframe = [0] * models_num

# creating a global List of these dataframes
dataFrames = [trainAccuracyDatafram, trainTrainingTimeDataframe,
trainTestingTimeDataframe,
trainTotalTimeDataframe, testAccuracyDatafram,
testTrainingTimeDataframe, testTestingTimeDataframe,
testTotalTimeDataframe]

# calling the 'declare_widgets' function, which will be described
later in the code, mainly, in declares all the widgets used later in
the code

declare_widgets()

# functions used to generate the numerical fields that correspond to
the categorical ones

def service_num(s):
    global service_types
    for i in range(0, len(service_types)):
        if service_types[i] == s:
            return i

def protocol_num(s):

```

```

global protocol_types
for i in range(0, len(protocol_types)):
    if protocol_types[i] == s:
        return i

def flag_num(s):
    global flag_types
    for i in range(0, len(flag_types)):
        if flag_types[i] == s:
            return i

def is_anomaly(s):
    if s == 'normal':
        return 0
    else:
        return 1

#.....
# this function draws a plot that clarifies the effect of the i'th
fields's value on the destination variable
# this is a helper function for the xPkde function
def pkde(i, seperate = False, saveF = False, saveDir = '', featureName
= ''):

    """Description:
    Prints a KDE that describes the relation between the target
    variable and the i'th feature in the input data

    Parameters:
    i: the index of the desired features
    """
    global numeric_data

```

```

plt.figure(figsize = (20, 8))

sns_save = sns.kdeplot(numeric_data[numeric_data['isAnomaly'] == 0][numeric_data.columns[i]], color = 'blue')

sns.kdeplot(numeric_data[numeric_data['isAnomaly'] == 1][numeric_data.columns[i]], color = 'red')

plt.legend(labels = ['Normal', 'Anomaly'], bbox_to_anchor=(1, 0.9), ncol=1, fancybox=True, shadow=True, fontsize = 20, markerscale = 2)

plt.title('(' + featureName + ')' + ' Feature Comparison',
fontdict = {'fontsize':'xx-large'})

if saveF == True:

    sns_save.figure.savefig(saveDir + ' Comparison',
bbox_inches="tight")

if seperate == False:

    return

plt.figure(figsize = (15, 8))

sns_save = sns.kdeplot(numeric_data[numeric_data['isAnomaly'] == 0][numeric_data.columns[i]], color = 'blue')

plt.legend(labels = ['Normal'], bbox_to_anchor=(1, 0.9), ncol=1, fancybox=True, shadow=True, fontsize = 20, markerscale = 2)

plt.title('(' + featureName + ')' + ' Feature "Normal"', fontdict = {'fontsize':'xx-large'})

if saveF == True:

    sns_save.figure.savefig(saveDir + ' Normal',
bbox_inches="tight")

plt.figure(figsize = (15, 8))

sns_save = sns.kdeplot(numeric_data[numeric_data['isAnomaly'] == 1][numeric_data.columns[i]], color = 'red')

plt.legend(labels = ['Anomaly'], bbox_to_anchor=(1, 0.9), ncol=1, fancybox=True, shadow=True, fontsize = 20, markerscale = 2)

plt.title('(' + featureName + ')' + ' Feature "Anomaly"', fontdict = {'fontsize':'xx-large'})

```

```
if saveF == True:  
    sns_save.figure.savefig(saveDir + 'Anomaly',  
bbox_inches="tight")  
  
# calls pkde function for all features  
  
def pkde_All(seperate = False):  
    global numeric_data  
    for i in range(0, len(numeric_data.columns)):  
        pkde(i, seperate = seperate)  
  
#.....  
.....  
# provided by the predictions of the model, returns the accuracy of  
the model, it also prints the confusion matrix and the classification  
report  
  
def pacc(pred, sampleSize = 0, printResults = True):  
    global test_labels, tot2  
  
    if sampleSize == 0: sampleSize = tot2  
    x = confusion_matrix(test_labels.sample(n = sampleSize,  
random_state = 101), pred)  
    acc = (x[0][0] + x[1][1]) / (x[0][0] + x[0][1] + x[1][0] +  
x[1][1])  
  
    if printResults == True:  
        print('Accuracy is: ', acc)  
        print(x)  
  
    if printResults == True:
```



```

tot_time.append(train_time[-1] + test_time[-1])

if printResults == True:
    print(titles[index])
    acc.append(pacc(pred, sampleSize = testingSampleSize, printResults
= printResults))

if printResults == True:
    print('\nTraining Time: ', train_time[-1])
    print('Testing Time: ', test_time[-1])
    print('Total Time: ', tot_time[-1])
    print('\nNormal/Anomaly Lables:')
    for i in range(0, len(pred)):
        print('({}: {})'.format(i, pred[i]), end = ' ')
        if i % 10 == 9 or i == len(pred) - 1:
            print('')
    else:
        print(', ', end = '')
    print('.....')

# Generates an entry in each of the 4 dataframes "accuracy, training
time, testing time, total time"

# each entry represents the results of running a specific algorithm
with certain parameters

#.....  

def testing_results(trainingSampleSize = 'All', testingSampleSize =
'All',
                     scaled = True, printResults = True,
                     variableParameter = 'training', test_only = False):

```

```

    global train_features, test_features, train_features_categories,
test_features_categories, tot1, tot2

    if trainingSampleSize == 'All': trainingSampleSize = tot1
    if testingSampleSize == 'All': testingSampleSize = tot2

        if trainingSampleSize > len(train_features) or trainingSampleSize
<= 0:

            print("Invalid Training Sample Size!!")
            return False

        if testingSampleSize > len(test_features) or testingSampleSize <=
0:

            print("Invalid Testing Sample Size!!")
            return False

    if printResults == False:

        var = trainingSampleSize

        if variableParameter == 'testing': var = testingSampleSize

        acc = [var]

        train_time = [var]
        test_time = [var]
        tot_time = [var]

    else:

        acc = []
        train_time = []
        test_time = []
        tot_time = []

lim = 4

if scaled == True: lim = 8

```


Parameters:

modelsIndex: the desired model's index

0: Logistic Regression

1: Decision Tree

2: Random Forest

3: K Nearest Neighbors

4: Support Vector

5: Naive Bayes

.....

maxTrainingSampleSize: the maximum training sample size to test on, if it's value is 'All',

then it is set to the size of the training dataset

.....

maxTestingSampleSize: the maximim testing sample size to test on, if it's value is 'All',

the it is set ti the size of the testing dataset

.....

scaledFlag: if set to true, runs the scaled and unscaled versions of training features,

otherwise runs only the unscaled ones

.....

n: in case of using Random Forest model, defines the number of estimators

in case of using K Nearest Neighbors, defines the number of points to use while estimating

.....

random_stat: used as a seed to choose subsets randomly from the dataset

if you desire your results to be compatible, it is necessary to use the same seed

in all your related function, so it's best to leave the default value

.....

staticTrainingSize: the training size used when running variable testing data sizes

.....

staticTestingSize: the testing size used when running variable training data sizes

.....

minTestingSampleSize: the minimum testing sample size to test on, if it's value is 'All',

the it is set ti the size of the training dataset

.....

minTrainingSampleSize: the minimum training sample size to test on, if it's value is 'All',

the it is set ti the size of the training dataset

.....

increasing_trainingSampleSize: if set to False, then the sample sizes of training subsets are taken in a linear format,

otherwise the intervals between consecutive values increase in each step

.....

increasing_testingSampleSize: if set to False, then the sample sizes of testing subsets are taken in a linear format,

otherwise the intervals between consecutive values increase in each step

.....

testingSamplesCnt: if the method for choosing subset sizes is linear, then this value defines the number of values to take in the defined interval

.....

trainingSamplesCnt: if the method for choosing subset sizes is linear, then this value defines the number of values to take in the defined interval

.....

training_increment_base1: if the method for choosing sizes is non-linear, then this value defines 1st degree increase in interval values

.....

testing_increment_base1: if the method for choosing sizes is non-linear, then this value defines 1st degree increase in interval values

.....

training_increment_base2: if the method for choosing sizes is non-linear, then this value defines 2nd degree

increase in interval values

.....

testing_increment_base2: if the method for choosing sizes is non-linear, then this value defines 2nd degree

increase in interval values

.....

```

    returns: A list of 8 dataframes, each has 9 columns (or 5 if
scaled values aren't calculated)

    first one is the sample size, either the training or the testing
one, next 4 columns are the

    unscaled column groups "Very Different, Different, Not Very
Similar, All"

    and the next 4 are the scaled versions of the previous 4

the 6 dataframes describe the following relations:

0) Accuracy = f(Training Sample Size)
1) Training Time = f(Training Sample Size)
2) Testing Time = f(Training Sample Size)
3) Total Time = f(Training Sample Size)

4) Accuracy = f(testing Sample Size)
5) Training Time = f(testing Sample Size)
6) Testing Time = f(testing Sample Size)
7) Total Time = f(testing Sample Size)

"""

global colNames, model, train_features_categories,
test_features_categories, tot1, tot2

global testAccuracyDatafram, testTrainingTimeDataframe,
testTestingTimeDataframe, testTotalTimeDataframe

global trainAccuracyDatafram, trainTrainingTimeDataframe,
trainTestingTimeDataframe, trainTotalTimeDataframe

if maxTrainingSampleSize == 'All': maxTrainingSampleSize = tot1
if maxTestingSampleSize == 'All': maxTestingSampleSize = tot2

```

```

    initializeModel(modelIndex = modelIndex, n = n, random_state =
random_state)

    accArray = []
    trainingTimeArray = []
    testingTimeArray = []
    totalTimeArray = []

    cols = colNames
    if scaledFlag == False:
        cols = colNames[0 : 5]
    cols[0] = 'Training Sample Size'

    for i in generateSampleSizes(increasing =
increasing_trainingSampleSize, cnt = trainingSamplesCnt,
                                base = minTrainingSampleSize, lim = maxTrainingSampleSize,
                                increment_base1 = training_increment_base1,
                                increment_base2 = training_increment_base2):
        acc, tr, ts, tot = testing_results(trainingSampleSize = i,
                                         testingSampleSize = staticTestingSize,
                                         printResults = False, scaled =
scaledFlag, variableParameter = 'training', test_only = False)
        accArray.append(acc)
        trainingTimeArray.append(tr)
        testingTimeArray.append(ts)
        totalTimeArray.append(tot)

    trainAccuracyDatafram[modelIndex] =
pd.DataFrame(np.array(accArray), columns = cols)
    trainTrainingTimeDataframe[modelIndex] =
pd.DataFrame(np.array(trainingTimeArray), columns = cols)
    trainTestingTimeDataframe[modelIndex] =
pd.DataFrame(np.array(testingTimeArray), columns = cols)

```

```

    trainTotalTimeDataframe[modelIndex] =
pd.DataFrame(np.array(totalTimeArray), columns = cols)

accArray = []
trainingTimeArray = []
testingTimeArray = []
totalTimeArray = []

cols[0] = 'Testing Sample Size'

for index in range(0, len(cols) - 1):

    model[index].fit(train_features_categories[index].sample(n
= staticTrainingSize, random_state = 101),
                      train_labels.sample(n = staticTrainingSize,
random_state = 101))

    for i in generateSampleSizes(increasing =
increasing_testingSampleSize, cnt = testingSamplesCnt,
                                base = minTestingSampleSize, lim = maxTestingSampleSize,
increment_base1 = testing_increment_base1,
                                increment_base2 = testing_increment_base2):
        acc, tr, ts, tot = testing_results(trainingSampleSize =
staticTrainingSize, testingSampleSize = i,
                                      printResults = False, scaled =
scaledFlag, variableParameter = 'testing', test_only = True)
        accArray.append(acc)
        trainingTimeArray.append(tr)
        testingTimeArray.append(ts)
        totalTimeArray.append(tot)

testAccuracyDatafram[modelIndex] =
pd.DataFrame(np.array(accArray), columns = cols)

```

```

    testTrainingTimeDataframe[modelIndex] =
pd.DataFrame(np.array(trainingTimeArray), columns = cols)

    testTestingTimeDataframe[modelIndex] =
pd.DataFrame(np.array(testingTimeArray), columns = cols)

    testTotalTimeDataframe[modelIndex] =
pd.DataFrame(np.array(totalTimeArray), columns = cols)

    ret = [trainAccuracyDatafram[modelIndex],
trainTrainingTimeDataframe[modelIndex],
trainTestingTimeDataframe[modelIndex],

        trainTotalTimeDataframe[modelIndex],
testAccuracyDatafram[modelIndex],
testTrainingTimeDataframe[modelIndex],

        testTestingTimeDataframe[modelIndex],
testTotalTimeDataframe[modelIndex]]

return ret

def runAllModelsStatistics(maxTrainingSampleSize = 5000,
maxTestingSampleSize = 1000,

                           scaledFlag = True, n = 5, random_state = 101,
staticTrainingSize = 1000, staticTestingSize = 1000,

                           minTestingSampleSize = 10, minTrainingSampleSize = 10,
increasing_trainingSampleSize = True,

                           increasing_testingSampleSize = True, testingSamplesCnt = 30,
trainingSamplesCnt = 30,

                           training_increment_base1 = 10, testing_increment_base1 = 10,
training_increment_base2 = 0,
                           testing_increment_base2 = 0):

    """Description:
calls 'runModelStatistics' function for all of the 6 models

```

Parameters:

maxTrainingSampleSize: the maximum training sample size to test on, if it's value is 'All',

then it is set to the size of the training dataset

.....

maxTestingSampleSize: the maximim testing sample size to test on, if it's value is 'All',

the it is set ti the size of the testing dataset

.....

scaledFlag: if set to true, runs the scaled and unscaled versions of training features,

otherwise runs only the unscaled ones

.....

n: in case of using Random Forest model, defines the number of estimators

in case of using K Nearest Neighbors, defines the number of points to use while estimating

.....

random_stat: used as a seed to choose subsets randomly from the dataset

if you desire your results to be compatible, it is necessary to use the same seed

in all your related function, so it's best to leave the default value

.....

staticTrainingSize: the training size used when running variable testing data sizes

.....

staticTestingSize: the testing size used when running variable training data sizes

.....

minTestingSampleSize: the minimum testing sample size to test on, if it's value is 'All',

the it is set ti the size of the training dataset

.....

minTrainingSampleSize: the minimum training sample size to test on, if it's value is 'All',

the it is set ti the size of the training dataset

.....

increasing_trainingSampleSize: if set to False, then the sample sizes of training subsets are taken in a linear format,

otherwise the intervals between consecutive values increase in each step

.....

increasing_testingSampleSize: if set to False, then the sample sizes of testing subsets are taken in a linear format,

otherwise the intervals between consecutive values increase in each step

.....

testingSamplesCnt: if the method for choosing subset sizes is linear, then this value defines the number of values to take in the defined interval

```

    trainingSamplesCnt: if the method for choosing subset sizes is
linear, then this value defines the number of

        values to take in the defined interval

    .....
    .....
    .....

    training_increment_base1: if the method for choosing sizes is
non-linear, then this value defines 1st degree

        increase in interval values

    .....
    .....
    .....

    testing_increment_base1: if the method for choosing sizes is
non-linear, then this value defines 1st degree

        increase in interval values

    .....
    .....
    .....

    training_increment_base2: if the method for choosing sizes is
non-linear, then this value defines 2nd degree

        increase in interval values

    .....
    .....
    .....

    testing_increment_base2: if the method for choosing sizes is
non-linear, then this value defines 2nd degree

        increase in interval values

    .....
    .....
    .....

    """
    for i in range(0, 6):

        runModelStatistics(modelIndex = i, maxTrainingSampleSize =
maxTrainingSampleSize,
                           maxTestingSampleSize = maxTestingSampleSize,
                           scaledFlag = scaledFlag, n = n, random_state = random_state,

```

```

                staticTrainingSize = staticTrainingSize,
staticTestingSize = staticTrainingSize,

                minTestingSampleSize = minTestingSampleSize,
minTrainingSampleSize = minTrainingSampleSize,

                increasing_trainingSampleSize =
increasing_trainingSampleSize,
                increasing_testingSampleSize =
increasing_testingSampleSize,
                testingSamplesCnt = testingSamplesCnt,
trainingSamplesCnt = trainingSamplesCnt,
                training_increment_base1 =
training_increment_base1, testing_increment_base1 =
testing_increment_base1,
                training_increment_base2 =
training_increment_base2, testing_increment_base2 =
testing_increment_base2)

#.....  

.....  

def runModelAllColumns(modelIndex, trainingSampleSize = 'All',
testingSampleSize = 'All',
                scaled = True, printResults = True, test_only =
False, random_state = 101, n = 5):
    """Description:
        runs the model on all the features groups "Very Different,
        Different, Not Very Similar, All" and their scaled
        versions if the scaled flag is set to True
    """

```

Parameters:

modelsIndex: the desired model's index

0: Logistic Regression

1: Decition Tree

```
2: Random Forest  
3: K Nearest Neighbors  
4: Support Vector  
5: Naive Bayes  
.....  
  
trainingSampleSize: the size of the training data, if it's value  
is 'All', then it is set to the  
size of the training dataset  
.....  
  
testingSampleSize the size of the testing data, if it's value is  
'All', then it is set to the  
size of the testing dataset  
.....  
  
scaled: if this flag is set to False, then scaled features aren't  
computed, and vice-versa  
.....  
  
printResults: if this flag is set to True, print the results, and  
vice-versa  
.....  
  
test_only: if this flag is set to True, skip training the model  
and jump straight to testing,  
note that the model should be atleast trained once before it can  
be tested or it will through an error  
.....  
  
random_stat: used as a seed to choose subsets randomly from the  
dataset
```

```
    if you desire your results to be compatible, it is necessary to  
use the same seed
```

```
    in all your related function, so it's best to leave the default  
value
```

```
.....
```

```
n: in case of using Random Forest model, defines the number of  
estimators
```

```
    in case of using K Nearest Neighbors, defines the number of  
points to use while estimating
```

```
.....
```

```
returns:
```

```
4 lists, each starting with the value of the sample size, and then  
8 values corresponding to the 8 features groups
```

```
these values describe the following values, depending on the list  
index:
```

- 0) accuracy
- 1) training time
- 2) testing time
- 3) total time

```
"""
```

```
global tot1, tot2
```

```
if trainingSampleSize == 'All': trainingSampleSize = tot1
```

```
if testingSampleSize == 'All': testingSampleSize = tot2
```

```
initializeModel(modelIndex = modelIndex, n = n, random_state =  
random_state)
```

```
    return testing_results(trainingSampleSize = trainingSampleSize,  
testingSampleSize = testingSampleSize,
```

```
        scaled = scaled, printResults = printResults,  
test_only = False)  
  
#.....  
.....
```

```
def runModel(modelIndex, columnIndex, n = 5, random_state = 101,  
trainingSampleSize = 'All',  
            testingSampleSize = 'All', test_only = False,  
printResults = True):
```

"""Description:

 runs the model on one of the features groups specified by their index

Parameters:

modelsIndex: the desired model's index

 0: Logistic Regression

 1: Decition Tree

 2: Random Forest

 3: K Nearest Neighbors

 4: Support Vector

 5: Naive Bayes

```
.....
```

columnIndex: the index of the desired features group:

0) Very Different

1) Different

2) Not Very Similar

3) All

4) Very Different(Scaled)

- 5) Different (Scaled)
 - 6) Not Very Similar (Scaled)
 - 7) All (Scaled)
-

trainingSampleSize: the size of the training data, if it's value is 'All', then it is set to the

size of the training dataset

.....

testingSampleSize the size of the testing data, if it's value is 'All', then it is set to the

size of the testing dataset

.....

printResults: if this flag is set to True, print the results, and vice-versa

.....

test_only: if this flag is set to True, skip training the model and jump straight to testing,

note that the model should be atleast trained once before it can be tested or it will through an error

.....

random_stat: used as a seed to choose subsets randomly from the dataset

if you desire your results to be compatible, it is necessary to use the same seed

in all your related function, so it's best to leave the default value

.....

```

    n: in case of using Random Forest model, defines the number of
estimators

    in case of using K Nearest Neighbors, defines the number of
points to use while estimating

    .....
    .....

    returns:

    4 values which are the accuracy, training time, testing time, and
total time of this model run

"""

global tot1, tot2

if trainingSampleSize == 'All': trainingSampleSize = tot1
if testingSampleSize == 'All': testingSampleSize = tot2

initializeModel(modelIndex = modelIndex, n = n, random_state =
random_state)

acc = []
train_time = []
test_time = []
tot_time = []

testing_results_for_model(index = columnIndex, titles = titles,
train_features_categories = train_features_categories,
test_features_categories = test_features_categories,
trainingSampleSize = trainingSampleSize,
testingSampleSize = testingSampleSize, acc = acc, train_time =
train_time, test_time = test_time,

```



```

    model[0].fit(train_features.sample(n = sampleSize, random_state =
101))

    model[1].fit(train_features_different.sample(n = sampleSize,
random_state = 101))

    model[2].fit(train_features_very_different.sample(n = sampleSize,
random_state = 101))

    model[3].fit(train_features_not_very_similar.sample(n =
sampleSize, random_state = 101))

    model[4].fit(scaled_train_features.sample(n = sampleSize,
random_state = 101))

    model[5].fit(scaled_train_features_different.sample(n =
sampleSize, random_state = 101))

    model[6].fit(scaled_train_features_very_different.sample(n =
sampleSize, random_state = 101))

    model[7].fit(scaled_train_features_not_very_similar.sample(n =
sampleSize, random_state = 101))

tst = list(train_labels.sample(n = sampleSize, random_state =
101))

pred = model[0].labels_; print('All Features:'); cnt = [0] *
cstrsNum; acc = 0; i = 0

for num in pred:
    cnt[num] += 1

    if num == tst[i]: acc += 1

    i += 1

print(cnt)

if 2 * acc < sampleSize: acc = sampleSize - acc
acc /= sampleSize

print('Accuracy is: ', acc)

```

```

pred = model[1].labels_; print('\nDifferent Features Only:');
cnt = [0] * clstrsNum; acc = 0; i = 0

for num in pred:
    cnt[num] += 1
    if num == tst[i]: acc += 1
    i += 1
print(cnt)
if 2 * acc < sampleSize: acc = sampleSize - acc
acc /= sampleSize
print('Accuracy is: ', acc)

pred = model[2].labels_; print('Very Different Features Only:');
cnt = [0] * clstrsNum; acc = 0; i = 0

for num in pred:
    cnt[num] += 1
    if num == tst[i]: acc += 1
    i += 1
print(cnt)
if 2 * acc < sampleSize: acc = sampleSize - acc
acc /= sampleSize
print('Accuracy is: ', acc)

pred = model[3].labels_; print('\nAll features Except very similar
ones:'); cnt = [0] * clstrsNum; acc = 0; i = 0

for num in pred:
    cnt[num] += 1
    if num == tst[i]: acc += 1
    i += 1
print(cnt)
if 2 * acc < sampleSize: acc = sampleSize - acc
acc /= sampleSize

```

```

print('Accuracy is: ', acc)

pred = model[4].labels_; print('\n\n All Features (Scaled) :'); cnt =
[0] * clstrsNum; acc = 0; i = 0

for num in pred:
    cnt[num] += 1
    if num == tst[i]: acc += 1
    i += 1
print(cnt)
if 2 * acc < sampleSize: acc = sampleSize - acc
acc /= sampleSize
print('Accuracy is: ', acc)

pred = model[5].labels_; print('\n Different Features
Only(Scaled) :'); cnt = [0] * clstrsNum; acc = 0; i = 0

for num in pred:
    cnt[num] += 1
    if num == tst[i]: acc += 1
    i += 1
print(cnt)
if 2 * acc < sampleSize: acc = sampleSize - acc
acc /= sampleSize
print('Accuracy is: ', acc)

pred = model[6].labels_; print('\n Very Different Features
Only(Scaled) :'); cnt = [0] * clstrsNum; acc = 0; i = 0

for num in pred:
    cnt[num] += 1
    if num == tst[i]: acc += 1

```

```

        i += 1
print(cnt)
if 2 * acc < sampleSize: acc = sampleSize - acc
acc /= sampleSize
print('Accuracy is: ', acc)

pred = model[7].labels_; print('\nAll features Except very similar
ones (Scaled):'); cnt = [0] * clstrsNum; acc = 0; i = 0

for num in pred:
    cnt[num] += 1
    if num == tst[i]: acc += 1
    i += 1
print(cnt)
if 2 * acc < sampleSize: acc = sampleSize - acc
acc /= sampleSize
print('Accuracy is: ', acc)
return acc

```

Plotting Functions:

```

# these two functions are helper functions used later by the xPlot
function, they draw a plot/interactive ploy given the desired
parameters,
# these functions are not designed to be used directly by the user
instead the xPlot function is used

def myPlot(data, col_index, color = 'blue', plot_type = sns.lineplot,
marker_type = 'o', marker_size = 100, linewidth = 3,
newFig = True, xlabel = 'Sample Size', ylabel =
'Accuracy', title = '',

```

```

        labelFontSize = 15, titleFontSize = 20, currModel = '',
figSize = (20, 10)):

    if newFig: plt.figure(figsize = figSize)

    if plot_type == sns.scatterplot:

        plot_type(x = data.columns[0], y = colNames[col_index], data =
data, color = color,
                   marker = marker_type, s = marker_size)

    if plot_type == sns.lineplot:

        plot_type(x = data.columns[0], y = colNames[col_index], data =
data, color = color, linewidth = linewidth)

    if newFig == True:

        plt.xlabel(xlabel, fontsize = labelFontSize)
        plt.ylabel(ylabel, fontsize = labelFontSize)

        if title == '': title = '{} "{} Fields"'.format(currModel,
colNames[col_index])

        plt.title(title, fontsize = titleFontSize)

def myiPlot(data, col_index, title = '', color = 'blue', mode =
'lines+markers', size = 10, width = 3,
           xTitle = 'Sample Size', yTitle = 'Accuracy', currModel =
'', marker_type = 'x'):

    if title == '': title = '{} "{} Fields"'.format(currModel,
colNames[col_index])

    if marker_type == 'o':
        marker_type = 'circle'
    if marker_type == 's':
        marker_type = 'square'
    if marker_type == 'd':
        marker_type = 'diamond'

```

```

        data.iplot(x = data.columns[0], y = colNames[col_index], mode =
mode, size = size, color = color, width = width,
                    title = title, xTitle = xTitle, yTitle = yTitle, symbol
= marker_type)

# this function returns the model index provided with the model name
def getModelIndexFromName(name):
    for i in range(0, len(modelsNames)):
        if modelsNames[i].lower() == name.lower():
            return i
        break
    return -1

# this function returns the dataframe index (which represents a
specific functional relation for a model) provided with the name of
the functional relation
def getDataframeByName(name):
    for i in range(0, len(dataFramesNames)):
        if dataFramesNames[i] == name:
            return dataFrames[i], i
    return -1

# this function returns the feature group index, provided with it's
name
def getColIndexFromName(name):
    for i in range(0, len(colNames2)):
        if colNames2[i].lower() == name.lower():
            return i + 1
        break
    return -1

```

```

def xPkde(name, seperate = False, saveF = False):
    """Description:
        prints 3 KDE plots to view the effect of the i'th parameter on the
        destination parameter

    Parameters:
        name: the name of the desired parameter
    """
    index = 0

    myStr = 'Features_Figures\\{}\ Feature '.format(name)

    for i in range(0, len(numeric_data.columns)):
        if numeric_data.columns[i] == name:
            index = i
            break

    pkde(index, seperate, saveF, myStr, featureName = name)

def xPlot(ModelName, DataframeName, ColName, PlotType, MarkerType,
          MarkerSize, Color, LineWidth, Xlabel,
          Ylabel, Title, LabelFontSize, TitleFontSize,
          figSizeX, figSizeY, iplotFlag, newFig = True):
    """Description:
        This function plots the required Model for the specified fratures
        group and the required parameters

```

Parameters:

ModelName: the name of the model

Options:

Logistic Regression

Decision Tree

Random Forest

K Nearest Neighbors

Support Vector

Naive Bayes

DataframeName: the names of the parameters to plot for 'e.g.
Accuracy = f(Training Sample Size)

Options:

Accuracy = f(Training Sample Size)

Training Time = f(Training Sample Size)

Testing Time = f(Training Sample Size)

Total Time = f(Training Sample Size)

Accuracy = f(Testing Sample Size)

Training Time = f(Testing Sample Size)

Testing Time = f(Testing Sample Size)

Total Time = f(Testing Sample Size)

colName: the name of the features group

Options:

Very Different Features Only

Different Features Only

Not Very Similar Features Only

All Features

Very Different Features Only(Scaled)

Different Features Only(Scaled)

Not Very Similar Features Only(Scaled)

All Features (Scaled)

PlotType: 'Scatter Plot' or 'Line Plot'

```

MarkerType: 'x', 'o', 's' = square, 'd' = diamond
MarkerSize: size of the markers
Xlabel, Ylabel: the labels of the x and y axis respectively, if
set to "Default" (WITH the quotes), then the values are set
according to the DataframeName parameter
iplotFlag: if set to true, plot and interactive plot
newFig: if set to false, the plotted figure will be plotted on top
of any previous one (only effective for normal "uninteractive" plots)

```

```
"""
```

```

modelIndex = getModelIndexFromName(ModelName)
df, I = getDataframeByName(name = DataframeName)
colIndex = getColIndexFromName(ColName)

```

```

yl = Ylabel
if Ylabel == '"Default"':
    if I % 4 == 0:
        yl = 'Accuracy'
    if I % 4 == 1:
        yl = 'Training Time'
    if I % 4 == 2:
        yl = 'Testing Time'
    if I % 4 == 3:
        yl = 'Total Time'

```

```

xl = 'Sample Size'
if Xlabel != '"Default"':
    xl = Xlabel

```

```
x_size = figSizeX
```

```

if figSizeX == '"Default"':
    x_size = 25

y_size = figSizeY
if figSizeY == '"Default"':
    y_size = 8

ttl = Title
if ttl == '"Default"':
    ttl = ''

plot_t = sns.lineplot
Mode = 'lines'
if PlotType == 'Scatter Plot':
    plot_t = sns.scatterplot
    Mode = 'markers'

if iplotFlag == False:
    return myPlot(
        data = df[modelIndex],
        col_index = colIndex,
        plot_type = plot_t,
        color = Color,
        marker_type = MarkerType,
        marker_size = MarkerSize,
        linewidth = LineWidth,
        xlabel = xl,

```

```

        ylabel = yl,
        title = ttl,
        currModel = modelsNames[modelIndex],

        labelFontSize = LabelFontSize,
        titleFontSize = TitleFontSize,
        figSize = (x_size, y_size),
        newFig = newFig
    )

else:
    return myiPlot(
        data = df[modelIndex],
        col_index = colIndex,
        title = ttl,
        color = Color,
        mode = Mode,
        size = MarkerSize / 15,
        width = LineWidth,
        xTitle = xl,
        yTitle = yl,
        currModel = modelsNames[modelIndex],
        marker_type = MarkerType,
    )
}

def xPlot_newfig(ModelName, DataframeName, ColName, PlotType,
MarkerType, MarkerSize, Color, LineWidth, Xlabel,
Ylabel, Title, LabelFontSize, TitleFontSize,
figSizeX, figSizeY, iplotFlag):

```

```

"""Description:

    this function is identical to 'xPlot', the only differenc is that
it always generates a new figure, see 'xPlot' description for more
info

"""

xPlot(ModelName = ModelName, DataframeName = DataframeName,
ColName = ColName, PlotType = PlotType, MarkerType = MarkerType,
MarkerSize = MarkerSize,

    Color = Color, LineWidth = LineWidth, Xlabel = Xlabel,
Ylabel = Ylabel, Title = Title, LabelFontSize = LabelFontSize,
    TitleFontSize = TitleFontSize, figSizeX = figSizeX, figSizeY
= figSizeY, iplotFlag = iplotFlag, newFig = True)

# this function is called by the addPlot function, it plots all the
plots corresponding to the parameters stored in these two lists on one
plot

plotsParameters = []
legendLabels = []

def plotAll(saveF = False, save_title = ''):

    global plotsParameters, legendLabels
    plt.figure(figsize = (20, 10))

    xl = plotsParameters[0][1].split(maxsplit =
2)[2].split('()')[1][0:-1]
    yl = plotsParameters[0][1].split()[0]

    for p in plotsParameters:
        xPlot( ModelName = p[0], DataframeName = p[1], ColName = p[2],
PlotType = p[3], MarkerType = p[4], MarkerSize = p[5], Color = p[6],
LineWidth = p[7],
            Xlabel = '-', Ylabel = '-', Title = '-', LabelFontSize =
10, TitleFontSize = 10, figSizeX = 20, figSizeY = 10, iplotFlag =
False, newFig = False)

```

```

x = p[1].split(maxsplit = 2)[2].split('()')[1][0:-1]
y = p[1].split(' ') [0]
if x != xl or y != yl:
    xl = ''
    yl = ''

plt.legend(labels = legendLabels, bbox_to_anchor=(1, 0.9),
           ncol=1, fancybox=True, shadow=True, fontsize = 20,
           markerscale = 2)
plt.title(label = save_title, fontdict = {'fontsize':'xx-large'})

plt.xlabel(xl, fontsize = 20)
plt.ylabel(yl, fontsize = 20)

if saveF == True:
    plt.savefig('Models_Figures\\{}\'.format(save_title + '.png'),
bbox_inches="tight")

# this function is used by the add plot function to clear all the
existing plot parameters in the two lists

def clearPlots():
    global plotsParameters, legendLabels
    plotsParameters = []
    legendLabels = []

def addPlot(ModelName, DataframeName, ColName, PlotType, MarkerType,
MarkerSize, Color, LineWidth, clear = False, saveF = False, save_title
= ''):

    """Description:
        This function plots the required Model for the specified fratures
        group and the required parameters, it plots this Plot on top of any
        previous one,

```

and the x, y labels are set according to the parameters only if they are identical for all the plots, otherwise the labels are set to be blank

this function also displays a legend describing each one of the plots

it is possible to clear the previous plots before plotting this one by setting the 'clear' flag to True

"Note that this function is global, it uses a global list to hold and plot it's plots"

Parameters:

ModelName: the name of the model

Options:

Logistic Regression

Decision Tree

Random Forest

K Nearest Neighbors

Support Vector

Naive Bayes

DataframeName: the names of the parameters to plot for 'e.g.
Accuracy = f(Training Sample Size)

Options:

Accuracy = f(Training Sample Size)

Training Time = f(Training Sample Size)

Testing Time = f(Training Sample Size)

Total Time = f(Training Sample Size)

Accuracy = f(Testing Sample Size)

Training Time = f(Testing Sample Size)

Testing Time = f(Testing Sample Size)

Total Time = f(Testing Sample Size)

colName: the name of the features group

Options:

```

Very Different Features Only
Different Features Only
Not Very Similar Features Only
All Features
Very Different Features Only(Scaled)
Different Features Only(Scaled)
Not Very Similar Features Only(Scaled)
All Features(Scaled)

PlotType: 'Scatter Plot' or 'Line Plot'
MarkerType: 'x', 'o', 's' = square, 'd' = diamond
MarkerSize: size of the markers
clear: if set to true, delete all the previous plots and plot this
plot as the first one

```

```
"""
```

```

global plotsParameters, legendLabels
if clear == True:
    clearPlots()
if saveF == False:
    plotsParameters.append([ModelName, DataframeName, ColName,
                           PlotType, MarkerType, MarkerSize, Color, LineWidth])
    legendLabels.append('{} "{}"\n{}'.format(ModelName, ColName,
                                             DataframeName))
plotAll(saveF = saveF, save_title = save_title)

```

```

# this function is called by a button to call the setup function
def xSetup(d):
    global bt, ck
    if ck.value == True:
        setup()
    else:
        setup(interactive_pick = True)
    print('Setup Successful!')

```



```

# this function is called by a button to call the runModel function
def xRunModel(d):
    global w0_modelName, w0_colName, w0_n, w0_trainingSampleSize,
w0_testingSampleSize, w0_printResults, w0_scaled, w0_testOnly, bt0,
ck0

    if w0_printResults.value == True:
        print('Model Name: {}\nTraining Sample Size:
{} \nTesting Sample Size: {}'.format(w0_modelName.value,
w0_trainingSampleSize.value,
w0_testingSampleSize.value))

    modelIndex = getModelIndexFromName(name = w0_modelName.value)
    columnIndex = getColIndexFromName(name = w0_colName.value) - 1

    if w0_printResults.value == True:
        if modelIndex == 2:
            print('Number of Estimators = ', w0_n.value)
        if modelIndex == 3:
            print('Number of points to consider = ', w0_n.value)

    if w0_printResults.value == True:

```

```

print('-----')

if ck0.value == False:
    runModel(
        modelIndex = modelIndex,
        columnIndex = columnIndex,
        n = w0_n.value,
        random_state=101,
        trainingSampleSize = w0_trainingSampleSize.value,
        testingSampleSize = w0_testingSampleSize.value,
        test_only = w0_testOnly.value,
        printResults = w0_printResults.value
    )
else:
    runModelAllColumns(
        modelIndex = modelIndex,
        trainingSampleSize = w0_trainingSampleSize.value,
        testingSampleSize = w0_testingSampleSize.value,
        scaled = w0_scaled.value,
        printResults = w0_printResults.value,
        test_only = w0_testOnly.value,
        random_state=101,
        n = w0_n.value
    )

if w0_printResults.value == True:
    print('\n#####\n#####\n#####\n#####\n#####\n#####\n')

```

```

# this function is called by a button to call the runModelStatistics
function, either for one, or for all the models, depending on the
user's choice

def xRunModelStatistics(d):
    global w1_modelName, w1_trainingSampleSizeInterval,
w1_testingSampleSizeInterval, w1_staticTrainingSampleSize

    global w1_staticTestingSampleSize, w1_trainingSamplesCnt,
w1_testingSamplesCnt, w1_trainingIntervalIncrementBase1,
w1_testingIntervalIncrementBase1

    global w1_trainingIntervalIncrementBase2,
w1_testingIntervalIncrementBase2, w1_n, w1_scaled,
w1_IncreasingTrainingSampleSize, w1_IncreasingTestingSampleSize, bt1,
ck1

    modelIndex = getModelIndexFromName(name = w1_modelName.value)

    i = 0; j = 6

    if ck1.value == False:
        i = modelIndex
        j = modelIndex + 1

    for ind in range(i, j):
        runModelStatistics(
            modelIndex = ind,
            maxTrainingSampleSize =
w1_trainingSampleSizeInterval.value[1],
            maxTestingSampleSize =
w1_testingSampleSizeInterval.value[1],
            scaledFlag = w1_scaled.value,
            n = w1_n.value,
            random_state = 101,
            staticTrainingSize = w1_staticTrainingSampleSize.value,
            staticTestingSize = w1_staticTestingSampleSize.value,

```

```

        minTestingSampleSize =
w1_testingSampleSizeInterval.value[0],

        minTrainingSampleSize =
w1_trainingSampleSizeInterval.value[0],

        increasing_trainingSampleSize =
w1_increasingTrainingSampleSize.value,

        increasing_testingSampleSize =
w1_increasingTestingSampleSize.value,

        testingSamplesCnt = w1_testingSamplesCnt.value,
        trainingSamplesCnt = w1_trainingSamplesCnt.value,
        training_increment_base1 =
w1_trainingIntervalIncrementBase1.value,
        testing_increment_base1 =
w1_testingIntervalIncrementBase1.value,
        training_increment_base2 =
w1_trainingIntervalIncrementBase2.value,
        testing_increment_base2 =
w1_testingIntervalIncrementBase2.value,
    )

```

def InteractiveSetup(w) :

"""Description:

runs a GUI to setup the database

Parameters:

w: a list of 2 widgets, the button and the checkbox required for the GUI, which are declared globally

"""

display(HBox([w[0], w[1]], box_style = 'info'))

def InteractiveModel(w) :

"""Description:

```
runs a GUI to run Models according to user's input
```

Parameters:

```
w: a list of widgets defined globally, which provide the user  
with the ability to pick the desired parameters
```

```
"""
```

```
display(HBox([w[0], w[1]]))  
display(HBox([w[2], w[3], w[4]]))  
display(HBox([w[5], w[6]]))  
display(HBox([w[7], w[8]]), box_style = 'info'))
```

```
def InteractiveModelStatistics(w):
```

"""Description:

```
runs a GUI to perform a statistical run on some or all of the  
available Models using user specified parameters
```

Parameters:

```
w: a list of widgets defined globally, which provide the user  
with the ability to pick the desired parameters
```

```
"""
```

```
display(HBox([w[0]]))  
display(HBox([w[1]]))  
display(HBox([w[2]]))  
display(HBox([w[3], w[4]]))  
display(HBox([w[5], w[6]]))  
display(HBox([w[7], w[8]]))  
display(HBox([w[9], w[10]]))  
display(HBox([w[11]]))  
display(HBox([w[13], w[14]]))  
display(HBox([w[15], w[16], w[12]]), box_style = 'info'))
```

```

# this function declares and initializes all the widgets used in the
code, it is called at the end of the setup function

def declare_widgets():

    global bt, ck, w_modelName, w_dataframeName, w_colName,
w_plotType, w_markerType, w_color, w_xLabel, w_yLabel, w_title,
w_markerSize, w_lineWidth

        global w_labelFontSize, w_titleFontSize, w_figSizeX, w_figSizeY,
w_iplot

        global w2_modelName, w2_dataframeName, w2_colName, w2_plotType,
w2_markerType, w2_color, w2_markerSize, w2_lineWidth, w2_clear

        global bt0, ck0, w0_modelName, w0_colName, w0_n,
w0_trainingSampleSize, w0_testingSampleSize, w0_printResults,
w0_scaled, w0_testOnly, modelsNames, colNames2

        global width1, w1_modelName, w1_trainingSampleSizeInterval,
w1_testingSampleSizeInterval, w1_staticTrainingSampleSize

        global w1_staticTestingSampleSize, w1_trainingSamplesCnt,
w1_testingSamplesCnt, w1_trainingIntervalIncrementBase1,
w1_testingIntervalIncrementBase1

        global w1_trainingIntervalIncrementBase2,
w1_testingIntervalIncrementBase2, w1_n, w1_scaled,
w1_increasingTrainingSampleSize, w1_increasingTestingSampleSize, bt1,
ck1

    desc_size = '200px' # 'initial'
    width = '800px' # 'initial'

    w_modelName = widgets.Dropdown(options = modelsNames, layout =
{'width':width},
                                    style = {'description_width' : desc_size},
                                    description = 'Model Name: ')

    w_dataframeName = widgets.Dropdown(options = dataFramesNames,
layout = {'width':width},
                                    style = {'description_width' : desc_size},
                                    description = 'Function: ')

```

```

w_colName = widgets.Dropdown(options = colNames2, layout =
{'width':width},
                               style = {'description_width' : desc_size},
                               description = 'Features Group: ')

w_plotType = widgets.Dropdown(options = ['Line Plot', 'Scatter
Plot'], layout = {'width':width},
                               style = {'description_width' : desc_size},
                               description = 'Plot Type: ')

w_markerType = widgets.Dropdown(options = ['o', 'x', 's', 'd'],
layout = {'width':width},
                               style = {'description_width' : desc_size},
                               description = 'Marker Type: ')

w_color = widgets.ColorPicker(layout = {'width':width}, style =
{'description_width' : desc_size},
                               description = "Plot's Color: ")

w_xLabel = widgets.Text(value = '"Default"', layout =
{'width':width},
                           style = {'description_width' : desc_size},
                           description = 'X Label: ')

w_yLabel = widgets.Text(value = '"Default"', layout =
{'width':width},
                           style = {'description_width' : desc_size},
                           description = 'Y Label: ')

w_title = widgets.Text(value = '"Default"', layout =
{'width':width},
                           style = {'description_width' : desc_size},
                           description = "Plot's Title: ")

w_markerSize = widgets.IntSlider(value = 150, min = 50, max = 500,
step = 50, layout = {'width':width},
                               style = {'description_width' : desc_size},
                               description = 'Marker Size: ')

w_lineWidth = widgets.FloatSlider(value = 2.5, min = 0.5, max =
10, step = 1, layout = {'width':width},
                               style = {'description_width' : desc_size},
                               description = 'Line Width: ')

w_labelFontSize = widgets.IntSlider(value = 20, min = 10, max =
35, step = 5, layout = {'width':width},
                               style = {'description_width' : desc_size},
                               description = 'Label Font Size: ')

```

```

        style = {'description_width' : desc_size},
description = 'Label Font Size: ')

w_titleFontSize = widgets.IntSlider(value = 20, min = 10, max =
35, step = 5, layout = {'width':width},

        style = {'description_width' : desc_size},
description = 'Title Font Size: ')

w_figSizeX = widgets.IntSlider(value = 30, min = 10, max = 35,
step = 5, layout = {'width':width},

        style = {'description_width' : desc_size},
description = 'Figure Size (X axis): ')

w_figSizeY = widgets.IntSlider(value = 8, min = 4, max = 20, step
= 2, layout = {'width':width},

        style = {'description_width' : desc_size},
description = 'Figure Size (Y axis): ')

w_iplot = widgets.Checkbox(value = False, layout =
{'width':width},

        style = {'description_width' : desc_size},
description = 'Use Iplots ')

w2_modelName = widgets.Dropdown(options = modelsNames, layout =
{'width':width},

        style = {'description_width' : desc_size},
description = 'Model Name: ')

w2_dataframeName = widgets.Dropdown(options = dataFramesNames,
layout = {'width':width},

        style = {'description_width' : desc_size},
description = 'Function: ')

w2_colName = widgets.Dropdown(options = colNames2, layout =
{'width':width},

        style = {'description_width' : desc_size},
description = 'Features Group: ')

w2_plotType = widgets.Dropdown(options = ['Line Plot', 'Scatter
Plot'], layout = {'width':width},

```

```

        style = {'description_width' : desc_size},
description = 'Plot Type: ')

w2_markerType = widgets.Dropdown(options = ['o', 'x', 's', 'd'],
layout = {'width':width},

        style = {'description_width' : desc_size},
description = 'Marker Type: ')

w2_color = widgets.ColorPicker(layout = {'width':width}, style =
{'description_width' : desc_size},

        description = "Plot's Color: ")

w2_markerSize = widgets.IntSlider(value = 150, min = 50, max =
500, step = 50, layout = {'width':width},

        style = {'description_width' : desc_size},
description = 'Marker Size: ')

w2_lineWidth = widgets.FloatSlider(value = 2.5, min = 0.5, max =
10, step = 1, layout = {'width':width},

        style = {'description_width' : desc_size},
description = 'Line Width: ')

w2_clear = widgets.Checkbox(value = False, layout =
{'width':width},

        style = {'description_width' : desc_size},
description = 'Clear Previous Plots ')

bt0 = widgets.Button(description = 'Run Model', button_style =
'primary', icon = 'play',

        layout = {'width':'300px'}, style =
{'description_width' : 'initial'}, )

ck0 = widgets.Checkbox(value = False, layout = {'width':'200px'},
style = {'description_width' : 'initial'}, description = 'Run for all
Feature Groups ')

bt0.on_click(xRunModel)

width0 = '500px'

```

```

w0_modelName = widgets.Dropdown(options = modelsNames, description =
= 'Model Name: ', layout = {'width':width0}, style =
{'description_width':'initial'})

w0_colName = widgets.Dropdown(options = colNames2, description =
'Features Group: ', layout = {'width':width0}, style =
{'description_width':'initial'})

w0_n = widgets.IntSlider(value = 5, min = 2, max = 20, step = 1,
description = '# of estimators/points to consider: ',
layout = {'width':width0}, style =
{'description_width':'initial'})

w0_trainingSampleSize = widgets.IntSlider(value = int(tot1 / 2),
min = 1, max = tot1, step = 50,
description = 'Training Sample Size: ',
layout = {'width':width0}, style = {'description_width':'initial'})

w0_testingSampleSize = widgets.IntSlider(value = int(tot2 / 2),
min = 1, max = tot2, step = 50,
description = 'Testing Sample Size: ',
layout = {'width':width0}, style = {'description_width':'initial'})

w0_printResults = widgets.Checkbox(value = True, description =
'Print Results ', layout = {'width':'200px'}, style =
{'description_width':'initial'})

w0_scaled = widgets.Checkbox(value = True, description = 'Run for
Scaled features too ', layout = {'width':'150px'}, style =
{'description_width':'initial'})

w0_testOnly = widgets.Checkbox(value = False, description = 'Test
Only(skip training) ', layout = {'width':'200px'}, style =
{'description_width':'initial'})

width1 = '800px'

w1_modelName = widgets.Dropdown(options = modelsNames, description =
= 'Model Name: ', layout = {'width':'500px'}, style =
{'description_width':'initial'})

```

```

w1_trainingSampleSizeInterval = widgets.IntRangeSlider(value =
(10, 1000), min = 5, max = tot1, step = 10,
description = 'Training Sample Size
Interval: ', layout = {'width':'1200px'}, style =
{'description_width':'initial'})

w1_testingSampleSizeInterval = widgets.IntRangeSlider(value = (10,
1000), min = 5, max = tot2, step = 10,
description = 'Testing Sample Size
Interval: ', layout = {'width':'1200px'}, style =
{'description_width':'initial'})

w1_staticTrainingSampleSize = widgets.IntSlider(value = 1000, min
= 5, max = tot1, step = 10,
description = 'Static Training Sample
Size: ', layout = {'width':width1}, style =
{'description_width':'initial'})

w1_staticTestingSampleSize = widgets.IntSlider(value = 500, min =
5, max = tot2, step = 10,
description = 'Static Testing Sample
Size: ', layout = {'width':width1}, style =
{'description_width':'initial'})

w1_trainingSamplesCnt = widgets.IntSlider(value = 20, min = 5, max
= 1000, step = 5,
description = 'Training Samples Cnt
(non-Increasing intervals only): ', layout = {'width':width1}, style =
{'description_width':'initial'})

w1_testingSamplesCnt = widgets.IntSlider(value = 20, min = 5, max
= 1000, step = 5,
description = 'Testing Samples Cnt
(non-Increasing intervals only): ', layout = {'width':width1}, style =
{'description_width':'initial'})

w1_trainingIntervalIncrementBase1 = widgets.IntSlider(value = 50,
min = 100, max = 50000, step = 50,
description = 'Training Samples Intervals
incrementing Base1: ', layout = {'width':width1}, style =
{'description_width':'initial'})

w1_testingIntervalIncrementBase1 = widgets.IntSlider(value = 50,
min = 100, max = 50000, step = 50,

```

```

        description = 'Testing Samples Intervals
incrementing Basel: ', layout = {'width':width1}, style =
{'description_width':'initial'})

w1_trainingIntervalIncrementBase2 = widgets.IntSlider(value = 5,
min = 0, max = 10000, step = 50,
description = 'Training Samples Intervals
incrementing Base2: ', layout = {'width':width1}, style =
{'description_width':'initial'})

w1_testingIntervalIncrementBase2 = widgets.IntSlider(value = 5,
min = 0, max = 10000, step = 50,
description = 'Testing Samples Intervals
incrementing Base2: ', layout = {'width':width1}, style =
{'description_width':'initial'})

w1_n = widgets.IntSlider(value = 5, min = 2, max = 20, step = 1,
description = '# of estimators/points to consider:',
layout = {'width':'500px'}, style =
{'description_width':'initial'})

w1_scaled = widgets.Checkbox(value = False, description = 'Run for
Scaled features too ', layout = {'width':'500px'}, style =
{'description_width':'initial'})

w1_increasingTrainingSampleSize = widgets.Checkbox(value = False,
description = 'Take Training Sample Values with increasing Intervals
',
layout = {'width':'500px'}, style = {'description_width':'initial'})

w1_increasingTestingSampleSize = widgets.Checkbox(value = False,
description = 'Take Testing Sample Values with increasing Intervals ',
layout = {'width':'500px'}, style = {'description_width':'initial'})

bt1 = widgets.Button(description = 'Run Model Statistics',
button_style = 'primary', icon = 'play',
layout = {'width':'300px'}, style =
{'description_width' : 'initial'}, )

ck1 = widgets.Checkbox(layout = {'width':'200px'}, style =
{'description_width' : 'initial'}, description = 'Run All Models ')

```

```

bt1.on_click(xRunModelStatistics)

# this button and checkbox are the ones used to call the setup
function, as the setup function is the one that actually decalers all
the widgets, these two widgets must be decalred first

# to call the setup function

bt = widgets.Button(description = 'Run Setup', button_style =
'primary', icon = 'play',
                     layout = {'width':'300px'}, style =
{'description_width' : 'initial'}, )

ck = widgets.Checkbox(value = True, layout = {'width':'200px'}, style
= {'description_width' : 'initial'}, description = 'Use Default Files
')

bt.on_click(xSetup)
print('Sccess')

# GUI ......

# Setup the database

InteractiveSetup([bt, ck])

# In[ ]:

# Run the Models as required

InteractiveModel([w0_modelName, w0_colName, w0_n,
w0_trainingSampleSize, w0_testingSampleSize, w0_printResults,
w0_scaled, bt0, ck0])

# In[ ]:

```

```

# Perform Statistical Runs on the Models

InteractiveModelStatistics([w1_modelName,
w1_trainingSampleSizeInterval, w1_testingSampleSizeInterval,
w1_staticTrainingSampleSize, w1_staticTestingSampleSize,
w1_trainingSamplesCnt, w1_testingSamplesCnt,
w1_trainingIntervalIncrementBase1, w1_testingIntervalIncrementBase1,
w1_trainingIntervalIncrementBase2,
w1_testingIntervalIncrementBase2, w1_n, w1_scaled,
w1_increasingTrainingSampleSize, w1_increasingTestingSampleSize, bt1,
ck1])

# In[ ]:

# Run the Unsupervised K Means Klustering algorithm and test it's
results

Interactive4 = interact_manual(runKMeansClustering, sampleSize =
widgets.IntSlider(value = 2000, min = 1000, max = tot1, step = 1000,
layout = {'width':'1500px'}, style = {'description_width':'initial'}))

# In[ ]:

# Plotting the figure Accuracy = f(Training Sample Size) for the K
Means Clustering Algorithm

p = []
a = [2000, 4000, 10000, 20000, 40000, 45000, 50000, 60000, 70000,
80000, 100000, 120000, 140000, 160000, 180000, 200000]
for i in range(0, len(a)):
    tmp = runKMeansClustering(a[i])
    p.append([a[i], tmp])
    i += 1
df = pd.DataFrame(p)

plt.figure(figsize = (20, 8))

```

```

sns_save = sns.lineplot(data = df, x = 0, y = 1)

plt.title('K Means Clustering Accuracy', fontdict = {'fontsize':'xx-large'})

plt.xlabel('Training Sample Size')

plt.ylabel('Accuracy')

sns_save.figure.savefig('KMeansClustering', bbox_inches="tight")

# PLOTTING

# In[ ]:

# Draw 3 plots to view the effect of the features on the destination parameter

# the blue plot represents the normal packets, and the red one represents the anomaly ones

Interact3 = interact_manual(xPkde, name = widgets.Dropdown(options =
numeric_data.columns[0 : -1], description = 'Feature Name',
style =
{'description_width':'initial'}, layout = {'width':'500px'}),

seperate = widgets.Checkbox(value = False,
description = 'Plot Normal and Anomaly plots Independantly as well',
style = {'description_width':'initial'},
layout = {'width':'1000px'}),

saveF = widgets.Checkbox(value = False,
description = 'Save Figure',
style = {'description_width':'initial'},
layout = {'width':'1000px'})

)

#display(Interact3)

# get Plots according to the entered parameters, this GUI is interactive, the plot will change as you modify the widgets stats

Interactive = interactive(xPlot_newfig, ModelName = w_modelName,
DataframeName = w_dataframeName, ColName = w_colName, PlotType =
w_plotType,
MarkerType = w_markerType, Title = w_title, Color = w_color,
Xlabel = w_xlabel, Ylabel = w_ylabel, LabelFontSize = w_labelFontSize,

```

```

        LineWidth = w_lineWidth, MarkerSize = w_markerSize,
TitleFontSize = w_titleFontSize, figSizeX = w_figSizeX, figSizeY =
w_figSizeY, iplotFlag = w_iplot)

display(Interactive)

# get Plots according to the entered parameters, you can draw more
# than one plot on top of eachother for comparision purposes,
# if you draw two plots that have incompatible axis, the x and y
# lables are hidden as they don't have a specific value

Interactive2 = interact_manual(addPlot, ModelName = w2_modelName,
DataframeName = w2_dataframeName,

    ColName = w2_colName, PlotType = w2_plotType, MarkerType =
w2_markerType, Color = w2_color, LineWidth = w2_lineWidth, MarkerSize
= w2_markerSize, clear = w2_clear,
    saveF = widgets.Checkbox(value = False, layout =
{'width':'800px'},
                           style = {'description_width' : '200px'},
                           description = 'Save Figure / Draw New Plot '),
    save_title = widgets.Text(value = '', layout =
{'width':'800px'},
                           style = {'description_width' : '200px'},
                           description = 'Figure Title: '))

```

المراجع

- [1] Malwarebytes. ‘State of Malware Report’, 2017. [online]. Available: <https://www.malwarebytes.com/pdf/white-papers/stateofmalware.pdf> [Accessed: 20/8/2019].
- [2] Github.com, ‘Jupyter notebook extensions’, 2018. [online]. Available: https://github.com/ipython-contrib/jupyter_contrib_nbextensions. [Accessed: 10/8/2019].
- [3] Kaggle.com, ‘Datasets’. [online]. Available: <https://www.kaggle.com/datasets>. [Accessed: 10/8/2019].
- [4] TejaBlaze, ‘Network Traffic Classification’, 2018. [online]. Available: https://github.com/TejaBlaze/Network_Traffic_Classification/tree/master/DC_Project_files?fbclid=IwAR3qaTr-0tjzLEUmSpHbPQmj6r2abli-FJgq68drhN2CwPM71xbXhiGpBiE. [Accessed: 10/8/2019].
- [5] Farhad Malik, ‘Machine Learning Algorithms Comparison’, 2018. [online]. Available: <https://medium.com/fintechexplained/machine-learning-algorithm-comparison-f14ce372b855>. [Accessed: 18/8/2019].
- [6] Danny Varghese, ‘Comparative Study on Classic Machine learning Algorithms’, 2018. [online]. Available: <https://towardsdatascience.com/comparative-study-on-classic-machine-learning-algorithms-24f9ff6ab222>. [Accessed: 18/8/2019].