1 Objectif

Description du service Azure Machine Learning de Microsoft. Travailler sous Azure Machine Learning Studio.

Microsoft Azure est la plateforme cloud de Microsoft. Elle offre un certain nombre de services pour le stockage, le calcul, le traitement et la transmission des données, la gestion des bases de données, le développement d'applications, etc[1].

Azure Machine Learning (Azure ML) est un service dédié à l'analyse prédictive[2]. Il propose les fonctionnalités nécessaires à la construction de modèles prédictifs, à leur évaluation, et à leur déploiement. Des algorithmes maisons, adossés à des méthodes reconnues, sont implémentées (régression logistique, forêts aléatoires, etc.). Nous pouvons démultiplier les analyses puisque Azure ML intègre le logiciel R et la grande majorité des packages associés. De fait, réaliser des traitements en ligne avec du code R est possible. Nous étudierons avec beaucoup de curiosité cette opportunité.

Azure Machine Learning Studio (ML Studio) est un front end accessible via un navigateur web. Il permet de piloter des analyses via l'élaboration de diagrammes de traitements, à l'instar des outils bien connus de data mining. On parle souvent de programmation visuelle (cf. SPAD, SAS EM, IBM SPSS Modeler, etc.).

Le data mining rentre de plain pied dans l'ère du « cloud computing » avec Azure ML. Les données sont stockées on ne sait où (les fameux « data centers ») et les calculs sont effectués à distance sur des serveurs externalisés. Un simple navigateur suffit pour développer les analyses et réaliser les traitements. Ainsi, un client léger avec des capacités limitées ne nous pénalise en rien, tant en matière de volumétrie qu'en matière de temps de calcul.

Microsoft propose une tarification sophistiquée du service. Mais nous pouvons nous exercer gratuitement sur l'outil ML Studio via le site https://studio.azureml.net/, en nous connectant avec un compte e-mail Microsoft. J'ai utilisé mon compte « @live.fr » en ce qui me concerne. Dans ce tutoriel, je montre les principales fonctionnalités de l'outil en réalisant quelques traitements types d'analyse prédictive.[3]

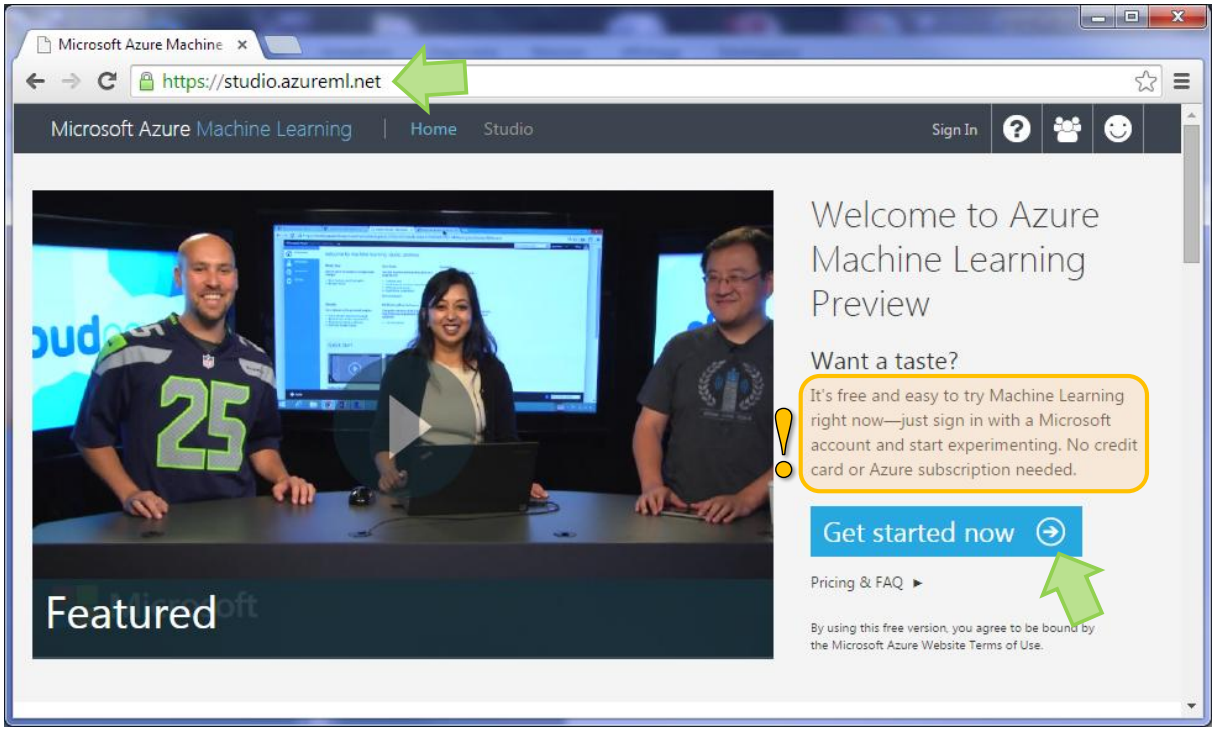
[1] http://azure.microsoft.com/fr-fr/overview/what-is-azure/

[2] http://azure.microsoft.com/en-us/services/machine-learning/

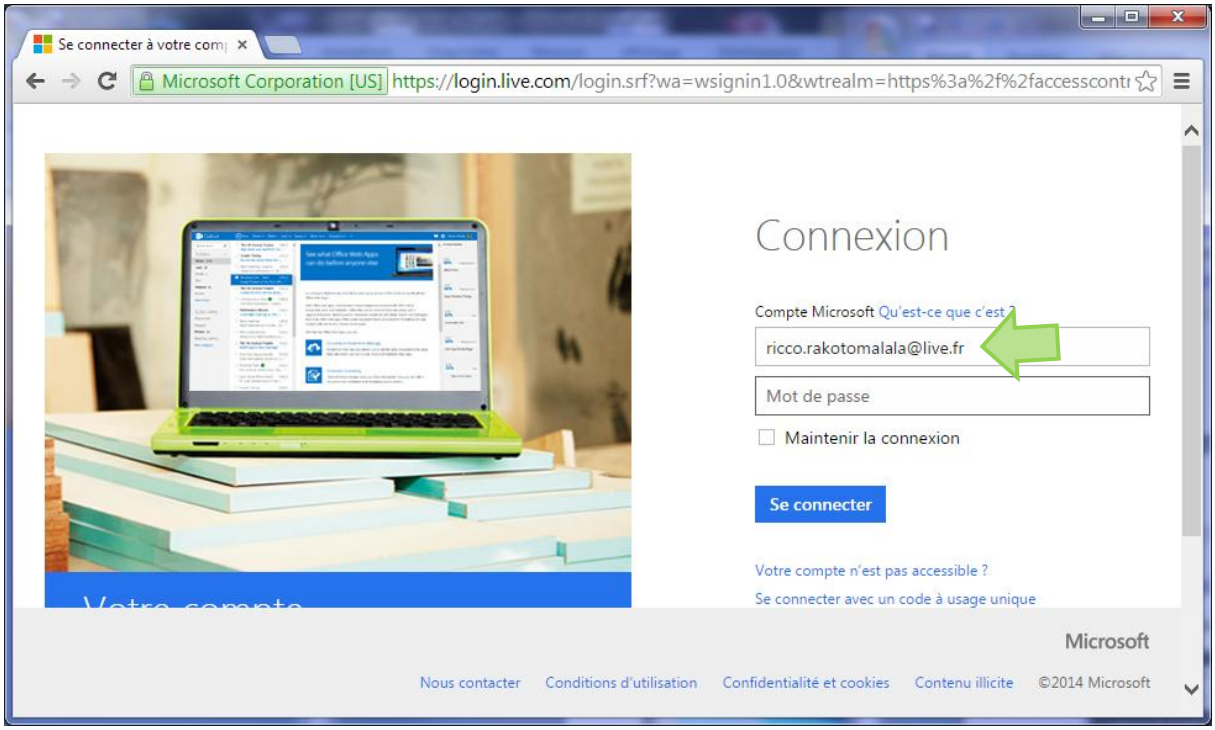
[3] Deux tutoriels m'ont beaucoup aidé : http://social.technet.microsoft.com/wiki/contents/articles/26689.predictive-analytics-with-microsoft-azure-machine-learning.aspx et http://msdn.microsoft.com/en-us/magazine/dn781358.aspx

2 Connexion à ML Studio

La page de garde du ML Studio se présente comme suit :

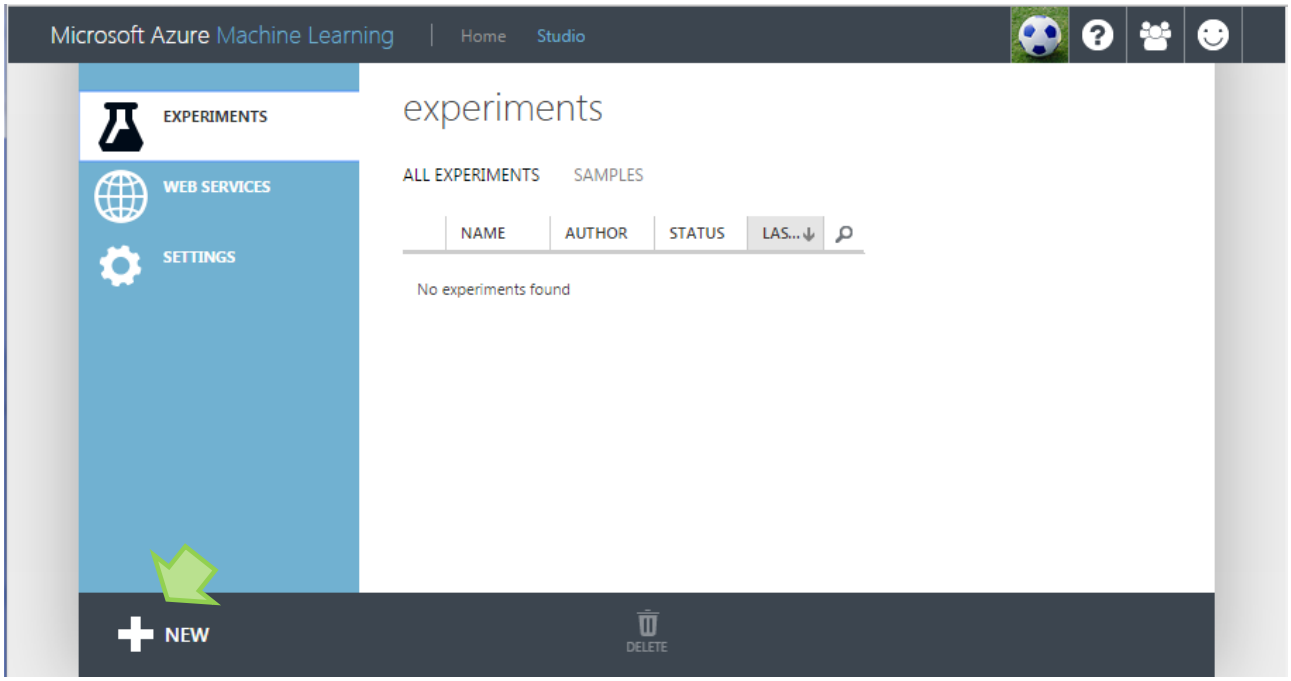


Il s'agit d'un site où nous pouvons évaluer gratuitement l'outil sans inscription spécifique. Disposer d'un compte Microsoft suffit. Nous cliquons sur « **Get started now** ».



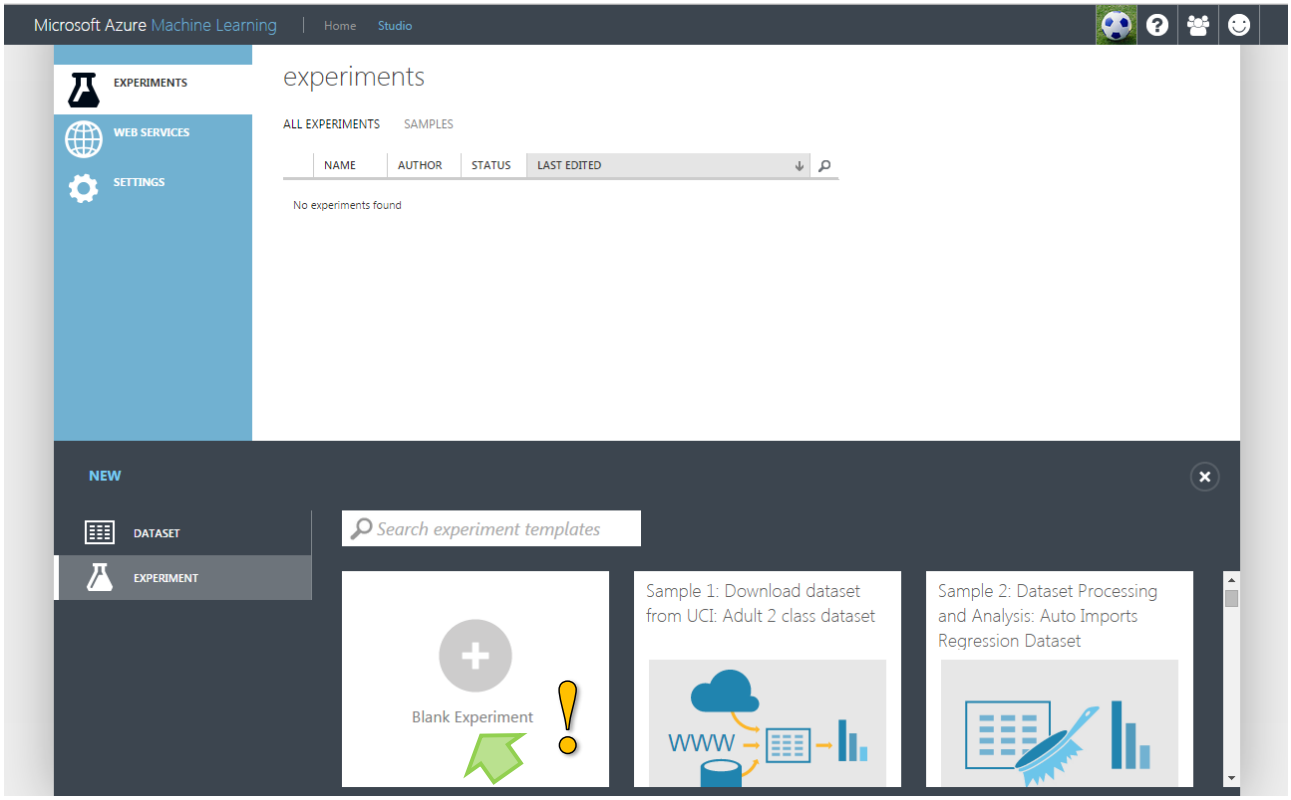
J'ai utilisé mon compte « @live.fr » pour ma part.

L'accueil est décomposé en plusieurs sections. La partie « **Experiments** » est dévolue à la création et à la gestion des projets.



3 Modélisation prédictive : évaluation et comparaison

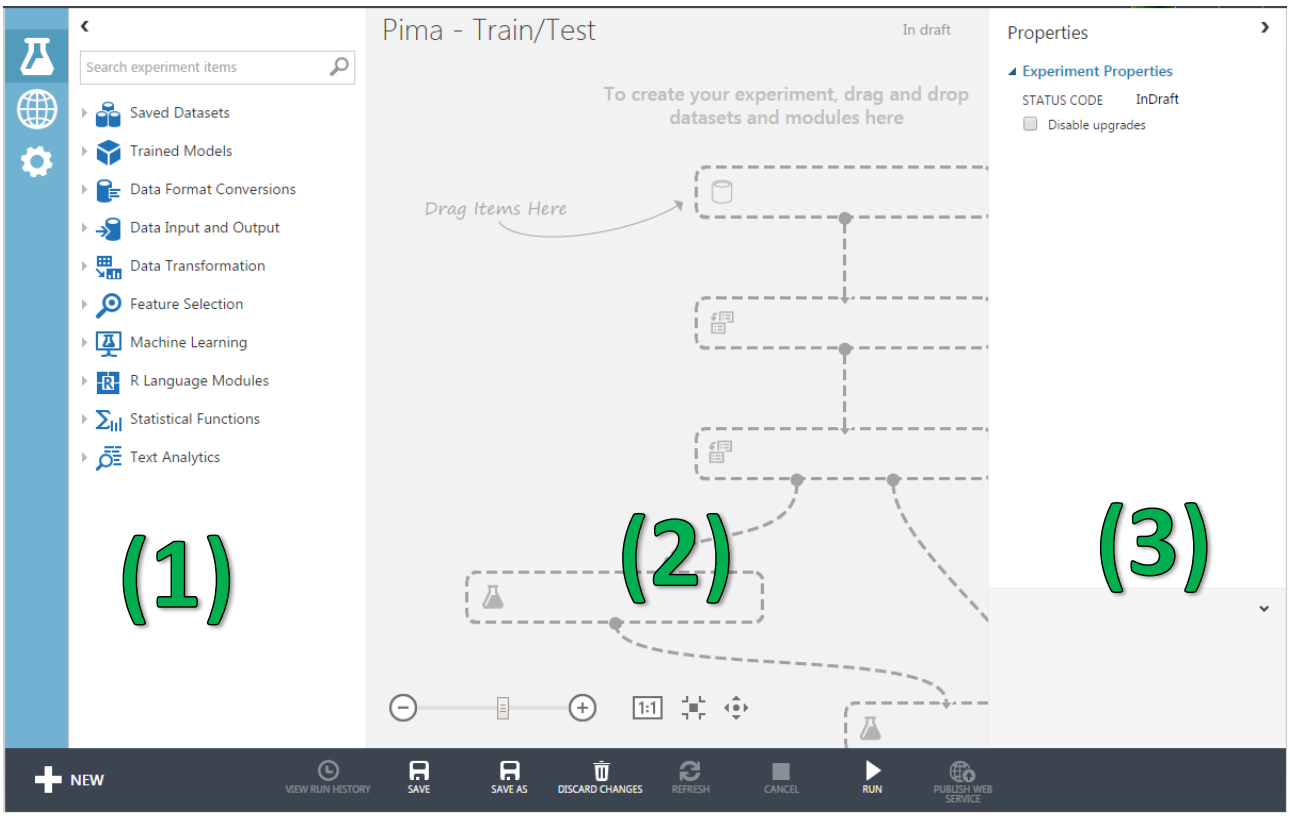
Pour définir un nouveau diagramme, nous cliquons sur **NEW**. ML Studio nous donne la possibilité de partir de zéro ou de nous appuyer sur des exemples types préprogrammées.



Nous choisissons de partir d'une page blanche[4].

3.1 Schéma apprentissage-test

Nous nommons le nouveau diagramme « Pima – Train / Test ».



La fenêtre principale est subdivisée en 3 parties :

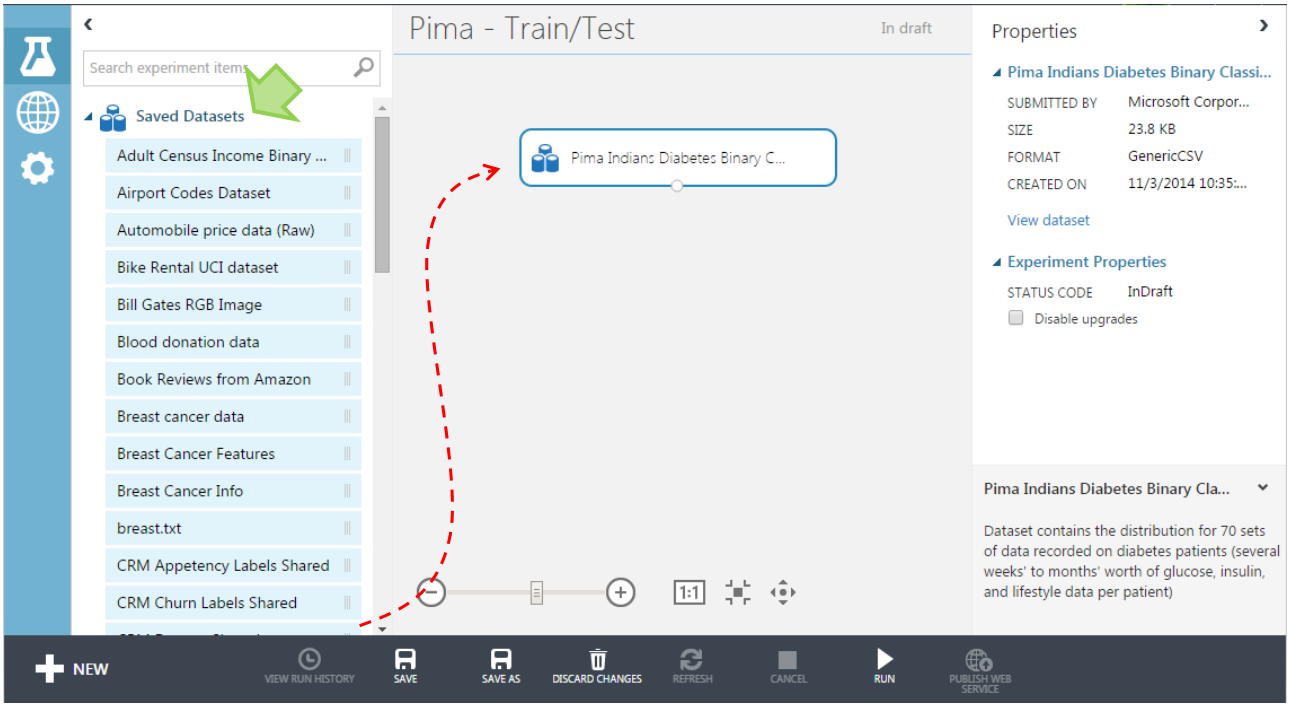
1. La palette des outils que nous pourrons utiliser pour les traitements.
2. L'espace où nous pourrons définir les traitements. Nous procéderons par drag-and-drop, en piochant les outils dans (1), et les plaçant dans l'espace de travail (2). Les icônes seront reliées entres elles pour spécifier les enchaînements.
3. Les propriétés du diagramme ou de l'outil sélectionné s'affichent dans la 3ème partie. Nous pourrons également préciser les paramètres des algorithmes lorsque cela est nécessaire.

3.1.1 Sélection des données

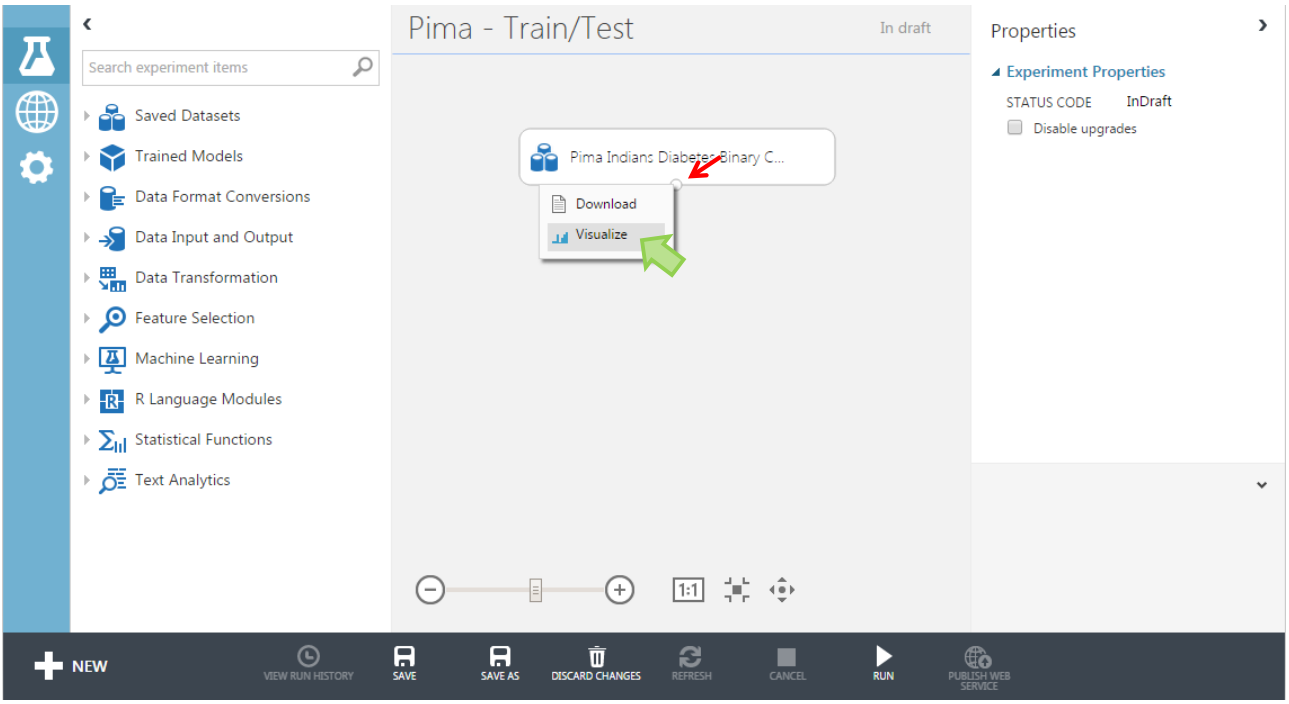
Effectuer des traitements sur ses propres données est possible. Nous y reviendrons plus bas (section 5). Pour l'heure, nous utilisons les données exemples fournies par Studio ML. Une grande partie provient du serveur UCI Machine Learning Repository.

[4] Vous imaginez bien que je me suis un peu exercé avant d'écrire ce tutoriel. Pour une réelle première prise en main, partir d'un exemple prédéfini est peut être plus approprié, ne serait-ce que pour identifier les éléments clés de l'outil.

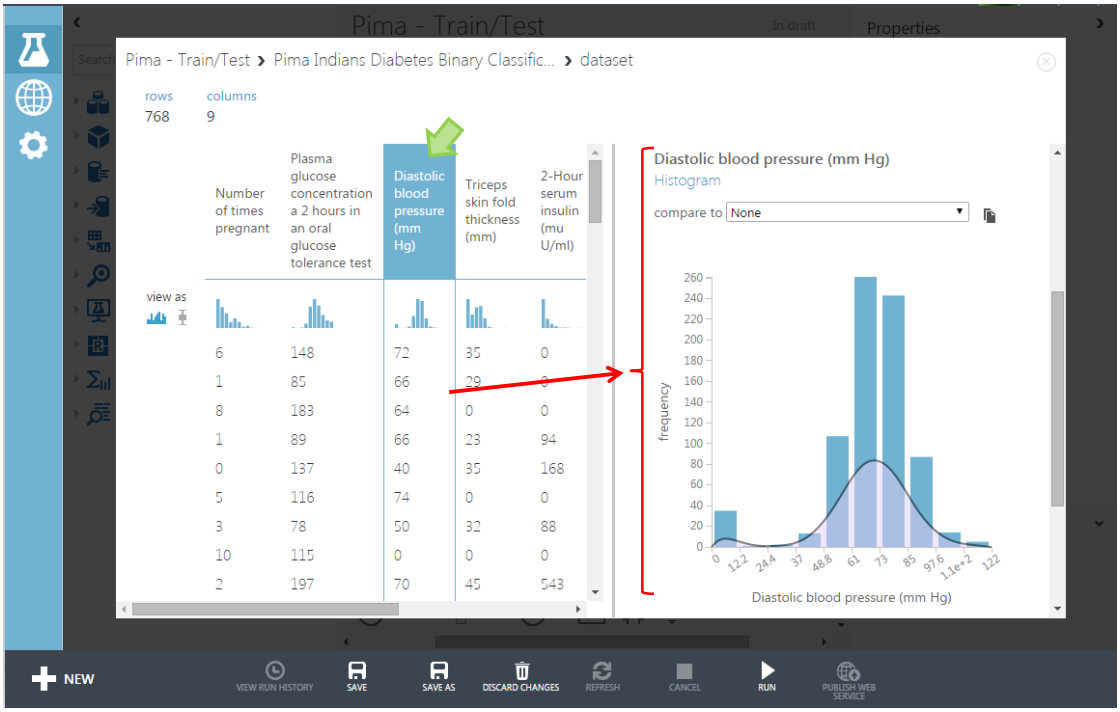
Nous développons le dossier « Saved Datasets ». Nous sélectionnons la base « Pima Indians Diabetes Binary Classifier » que nous glissons dans l'espace de travail.



Les caractéristiques de la base s'affichent à droite. Nous visualisons les données en effectuant un clic-droit sur le bouton de sortie et en sélectionnant l'item « **Visualize** » du menu contextuel.

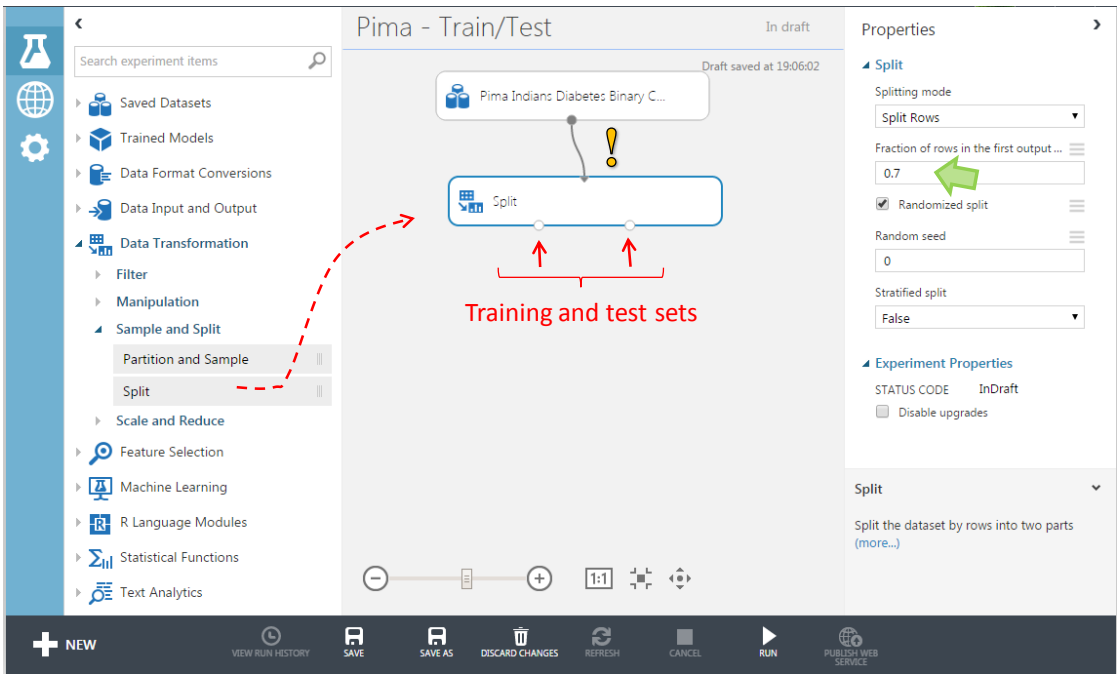


Les valeurs s'affichent dans une nouvelle fenêtre.



3.1.2 Subdivision des données en apprentissage et test

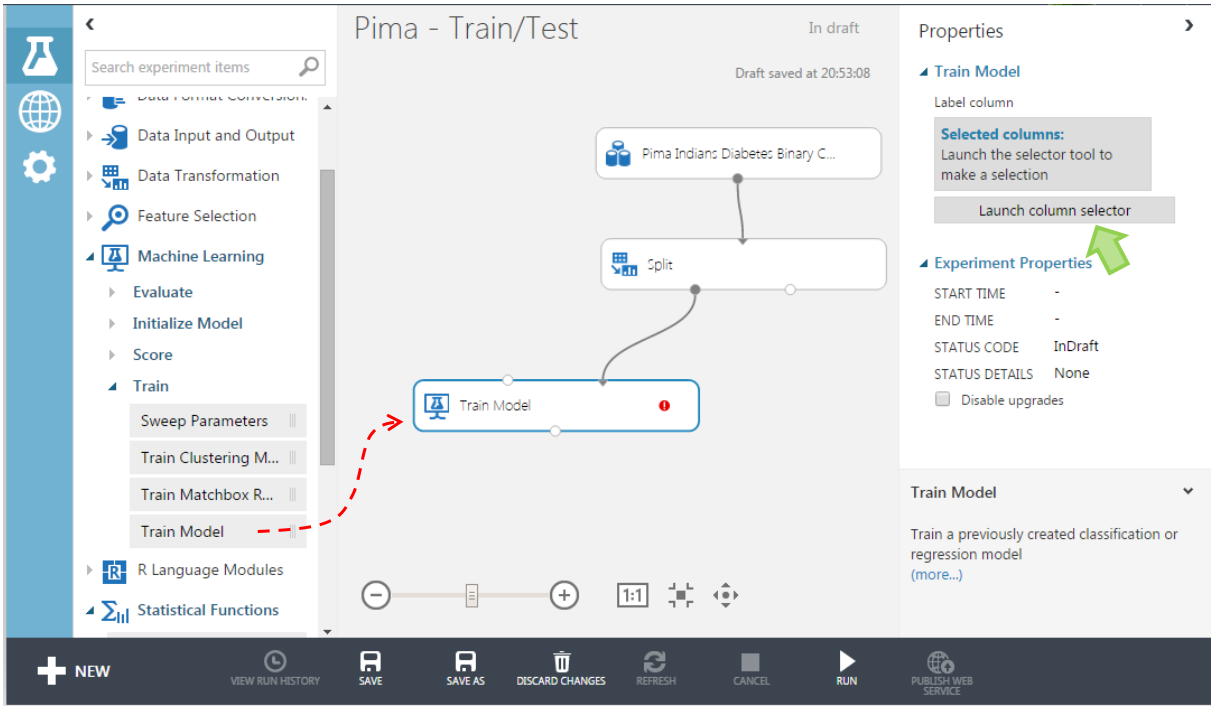
Pour obtenir une image honnête du comportement des modèles, nous devons subdiviser les données en échantillon d'apprentissage, consacré à leur construction, et en échantillon test, pour l'évaluation de leurs performances prédictives.



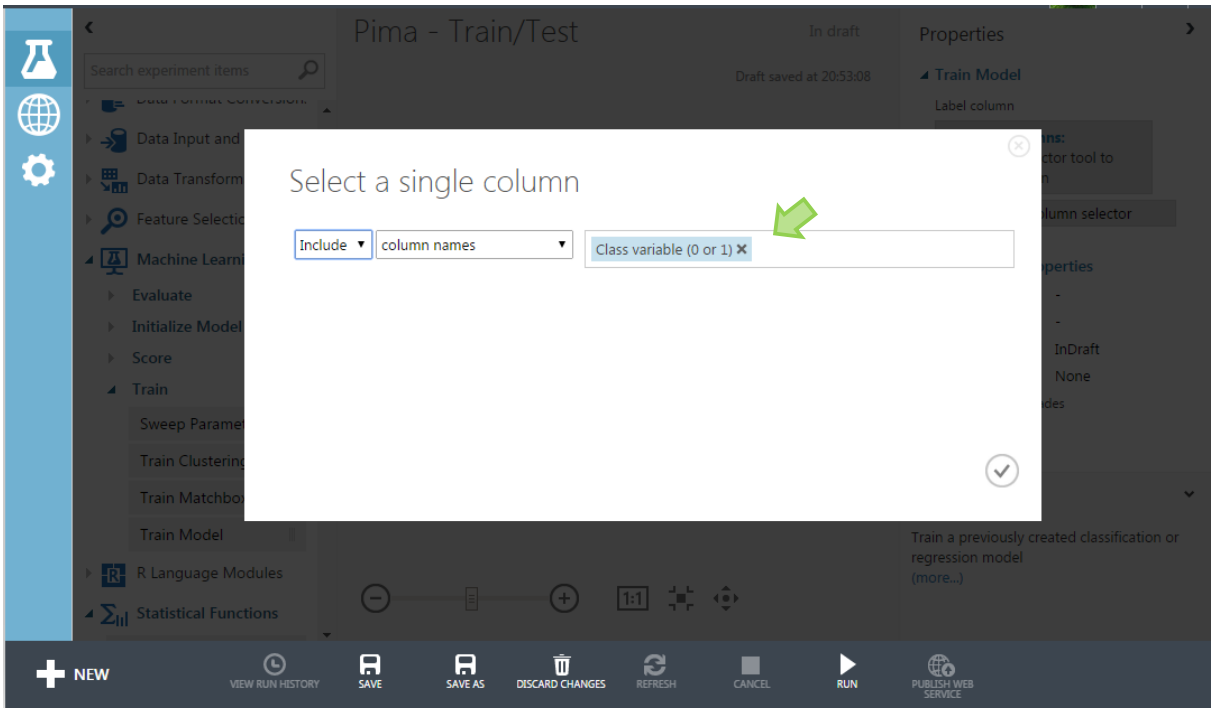
Le composant **Split** (Data Transformation / Sample and Split / Split) permet de partitionner les données en deux sous-échantillons. Le paramètre « **Fraction of rows in the first output** » précise la proportion des individus du premier échantillon, celui qui servira à l'apprentissage.

3.1.3 Choix de la méthode et apprentissage

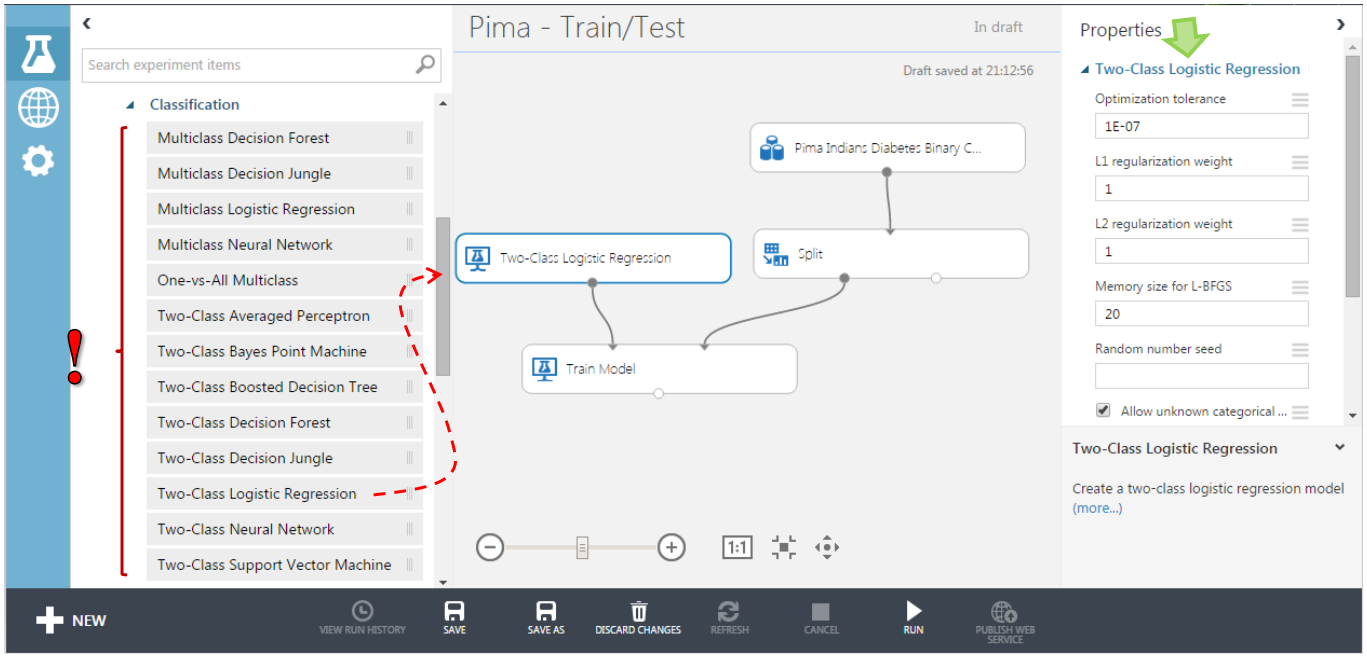
Le composant « **Train Model** » réalise le processus de modélisation. Il prend en entrée les données d'apprentissage…



En cliquant sur **Launch Column Selector**, nous le paramétrons pour spécifier la variable cible.



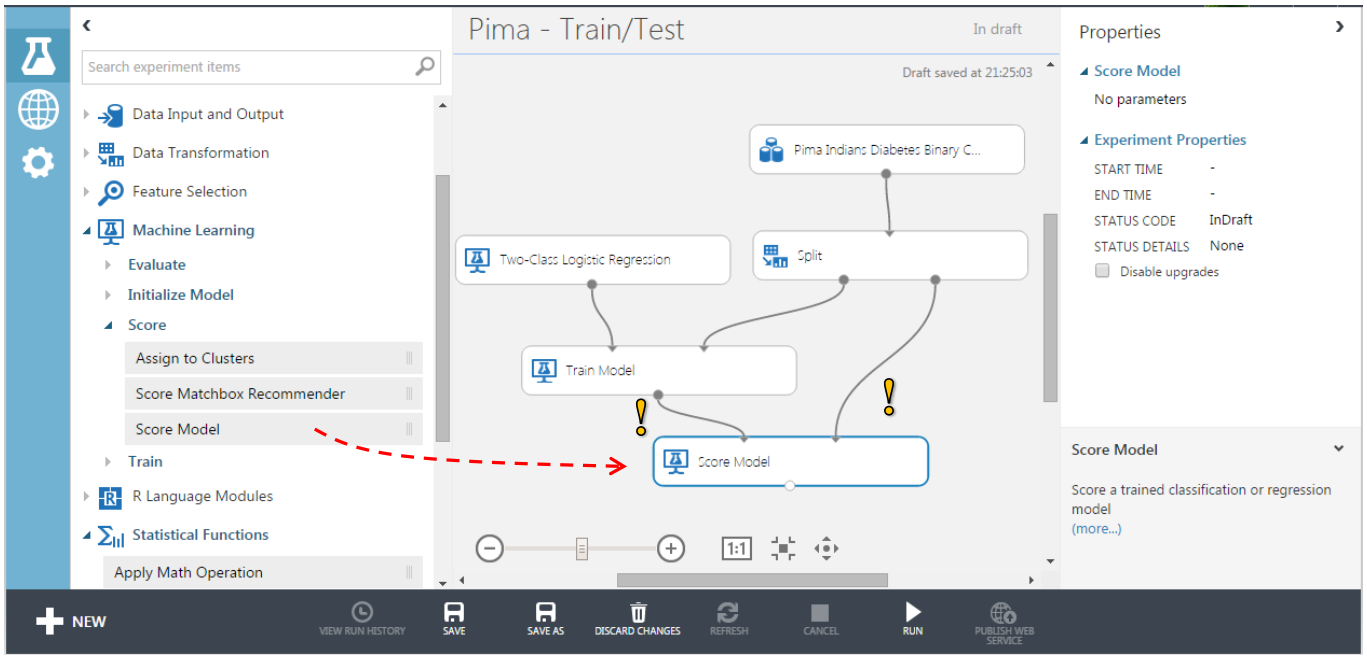
… et la méthode d'apprentissage « Two-Class Logistic Regression » que nous trouvons dans la branche Machine Learning / Initialize Model / Classification.



Les paramètres de la méthode s'affichent sur la partie droite de la fenêtre. On notera entre autres les paramètres de régularisation L1 et L2 pour le traitement des données à forte dimensionnalité (quand le ratio nombre de variables prédictives sur nombre d'observations est dangereusement élevé).

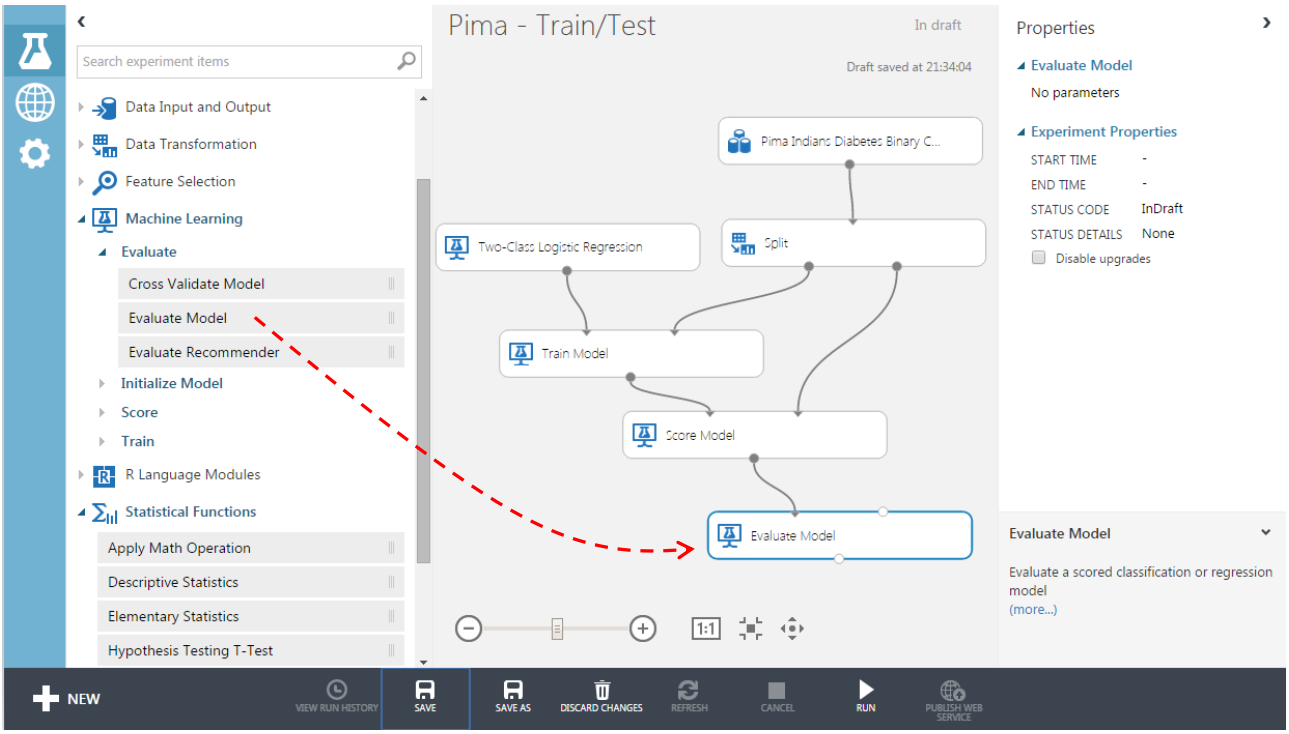
3.1.4 Prédiction et évaluation sur l'échantillon test

Nous devons « scorer » les individus de l'échantillon test pour pouvoir évaluer le modèle. Le composant Score Model (branche Machine Learning / Score) s'en charge.



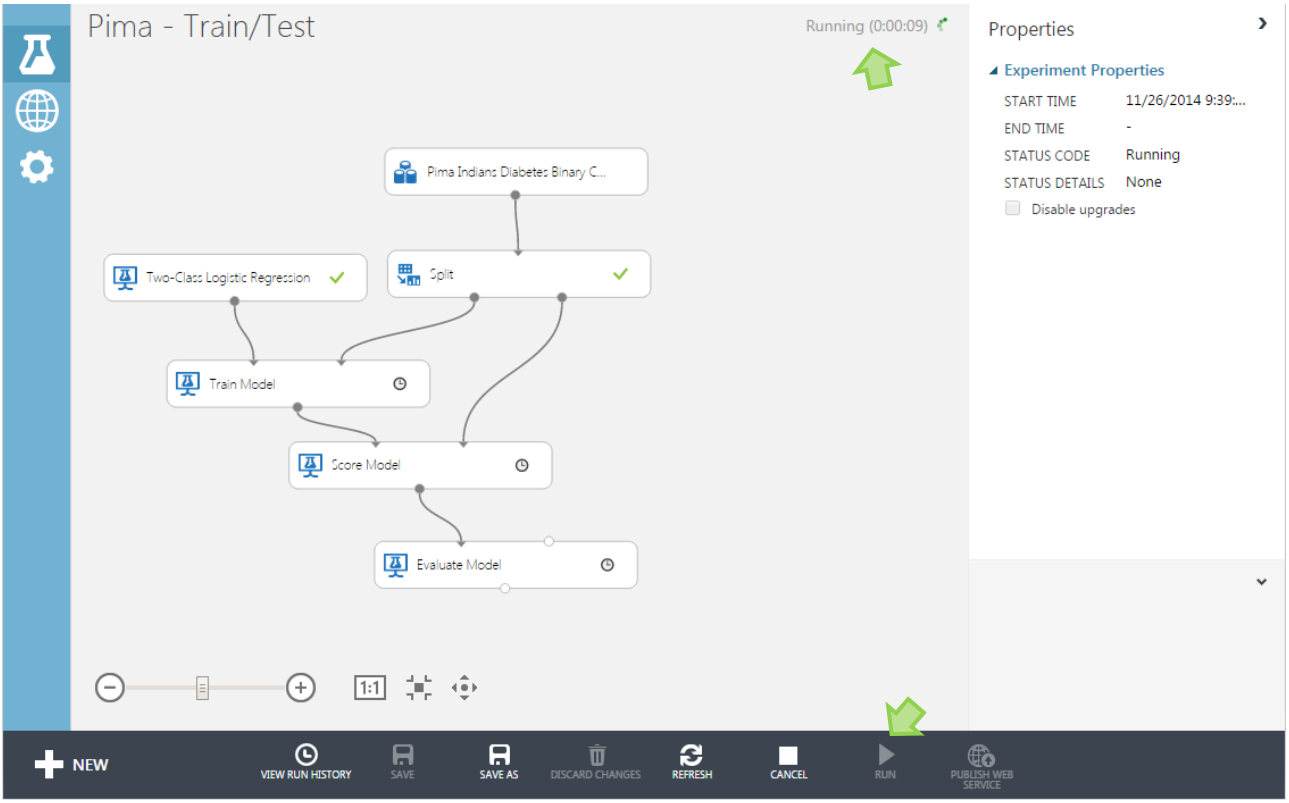
Il prend en entrée le modèle construit et la seconde sortie de **Split** c.-à-d. l'échantillon test.

Nous plaçons enfin le composant **Evaluate Model** (branche Machine Learning / Evaluate). Il peut prendre en entrée deux scores pour la comparaison de modèles (section 0).

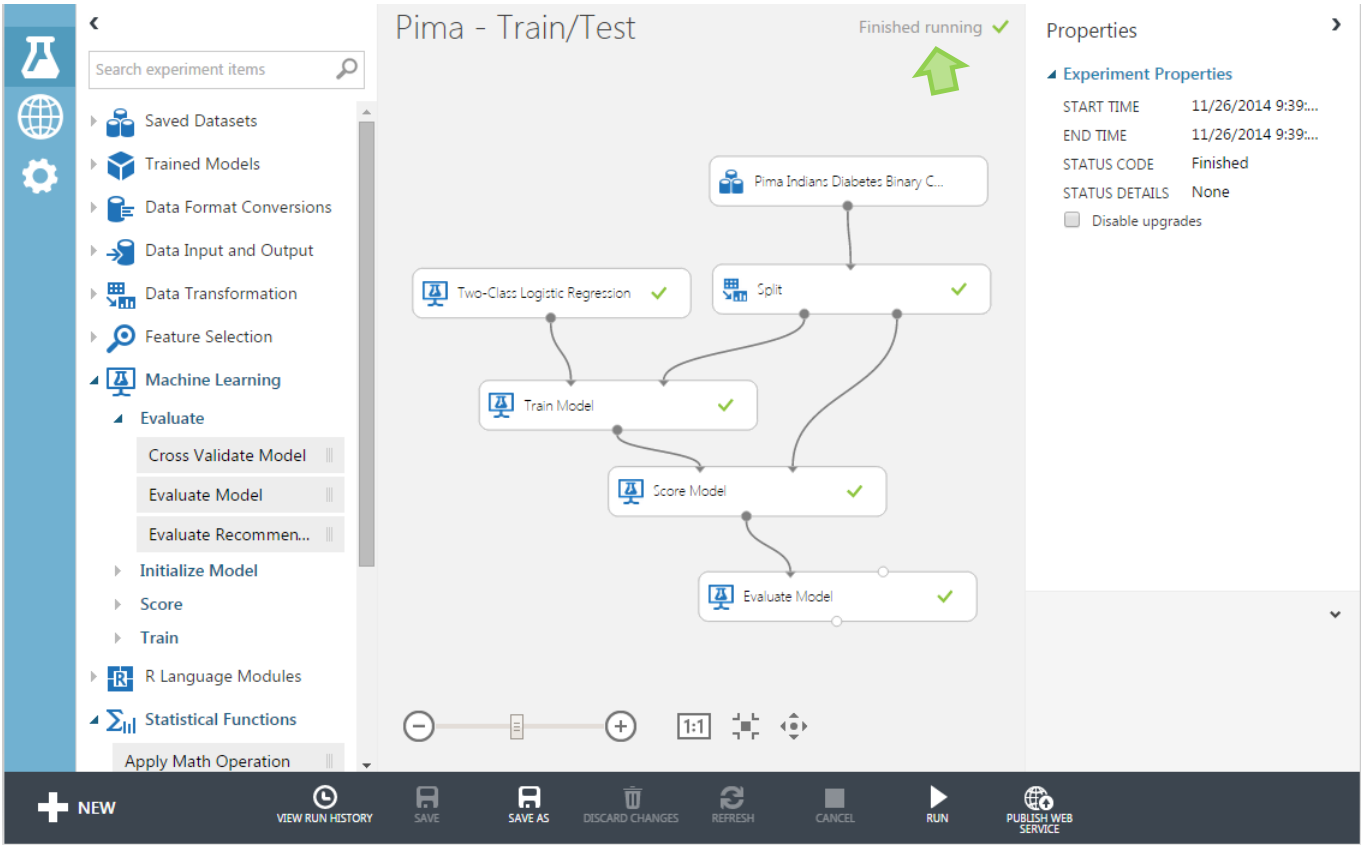


3.1.5 Lancement des traitements

Ca y est, tout est prêt. Nous lançons les traitements en cliquant sur le bouton RUN.



Après avoir placé le diagramme dans une file d'attente (message « Queued »), Azure ML lance les traitements. Nous pouvons suivre son avancée (message « Running »). Aucun calcul n'est effectué localement. J'ai surveillé attentivement le gestionnaire de tâches de Windows.

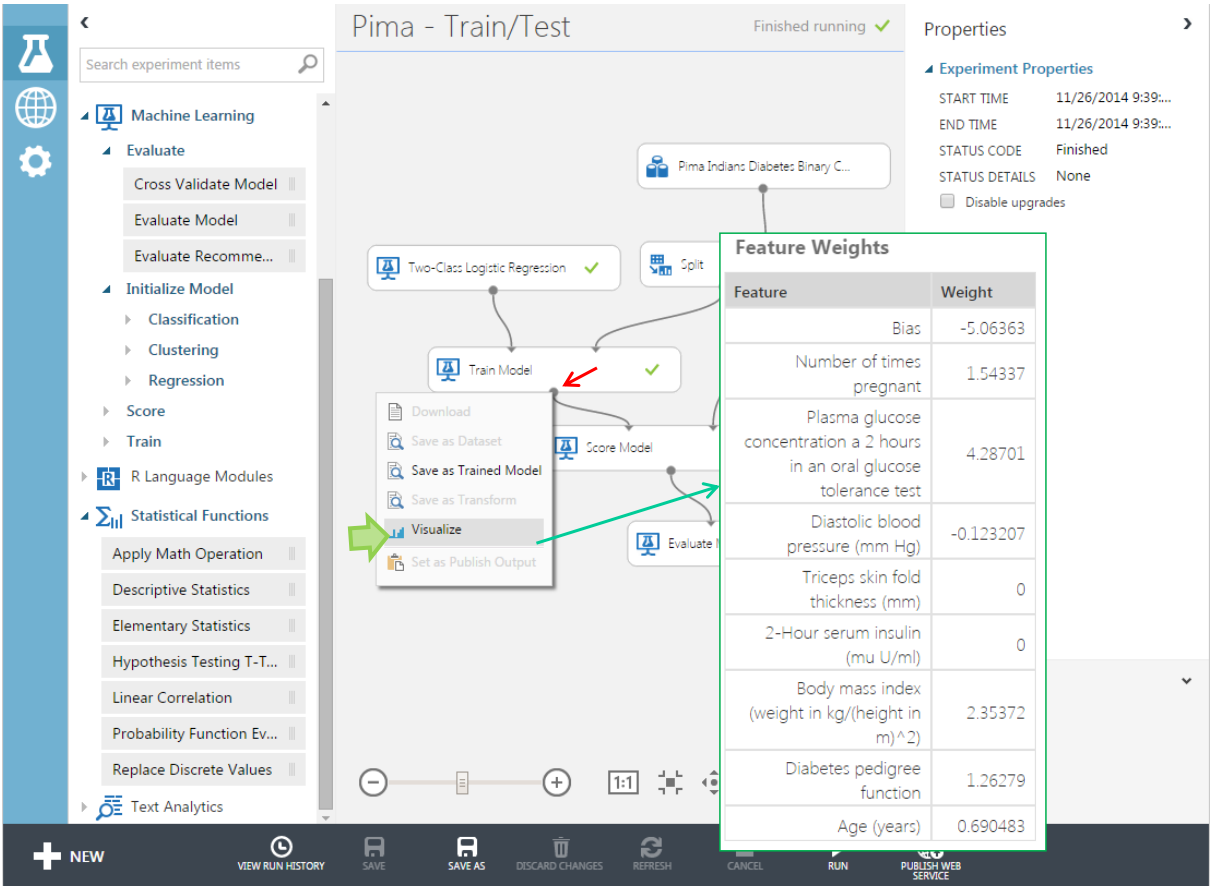


Chaque composant est validé par une encoche verte à l'issue de l'analyse. Le message « Finished running » est affiché. Un symbole rouge apparaît sur le composant incriminé en cas d'erreur.

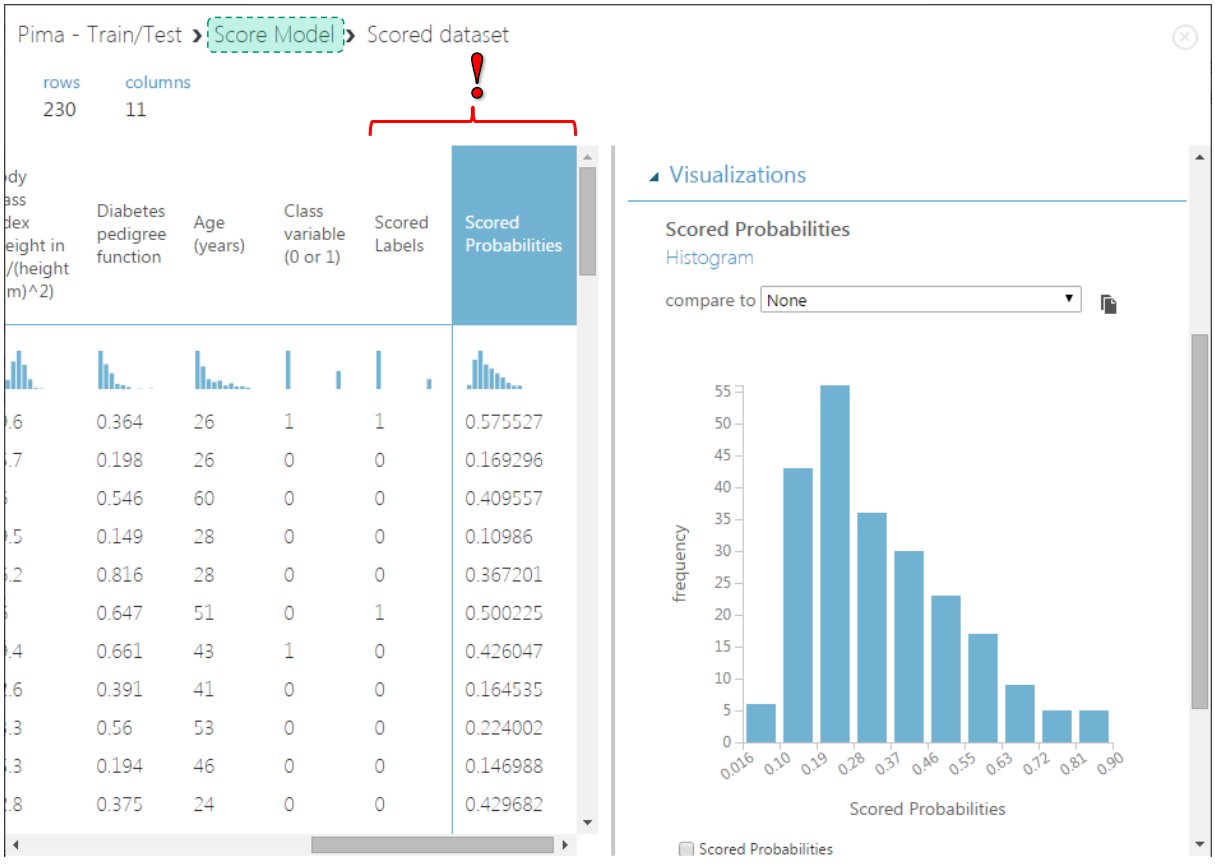
3.1.6 Visualisation des résultats

Nous pouvons consulter les résultats sur chaque composant. Il faut effectuer un clic-droit sur le composant visé pour ce faire, et cliquer sur l'item **Visualize** du menu contextuel.

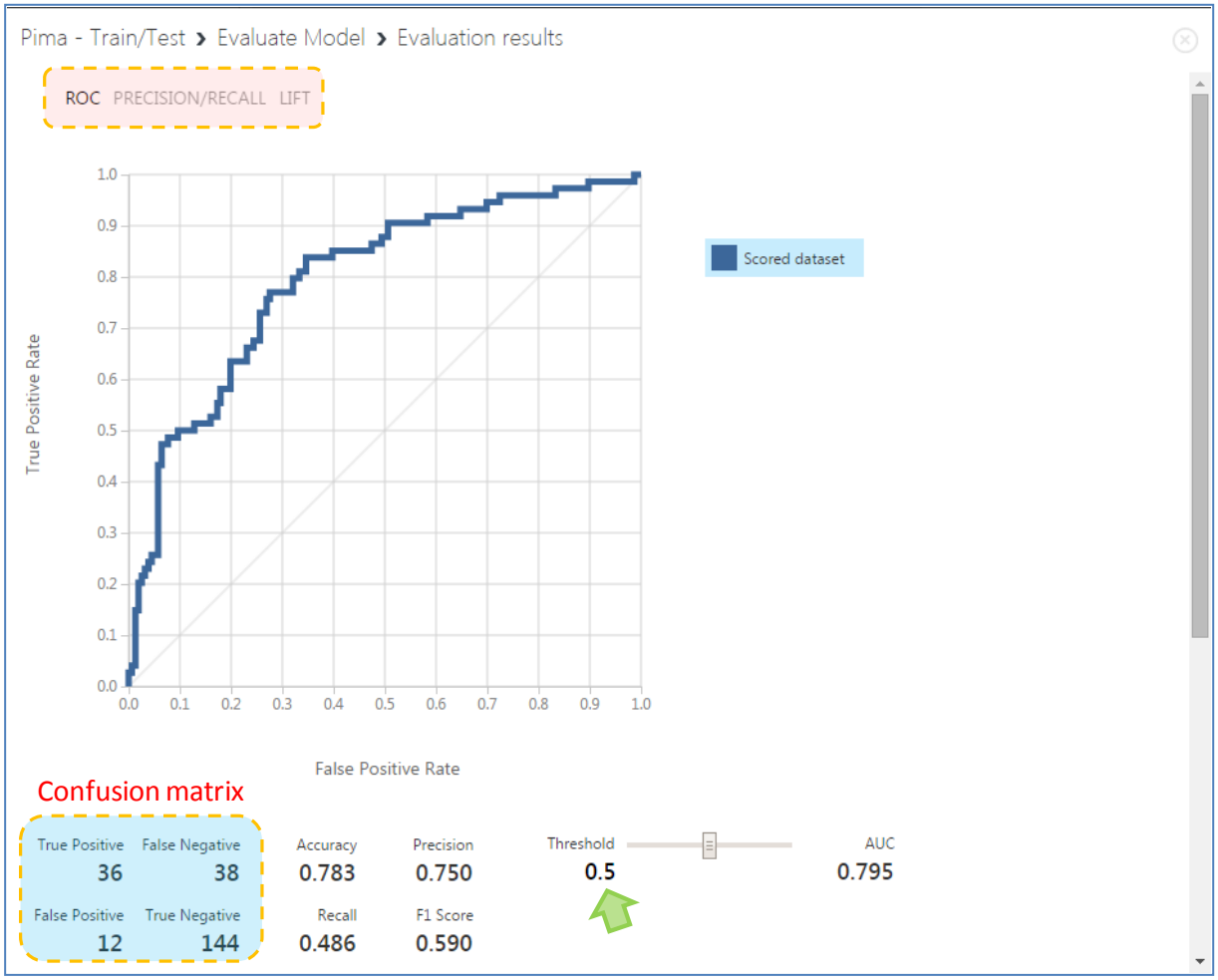
Pour **Train Model**, nous obtenons les coefficients de la régression logistique.



Nous avons les scores et les prédictions sur l'échantillon test sur le composant **Score Model**.



Evaluate Model propose une lecture assez complète de l'évaluation des modèles.

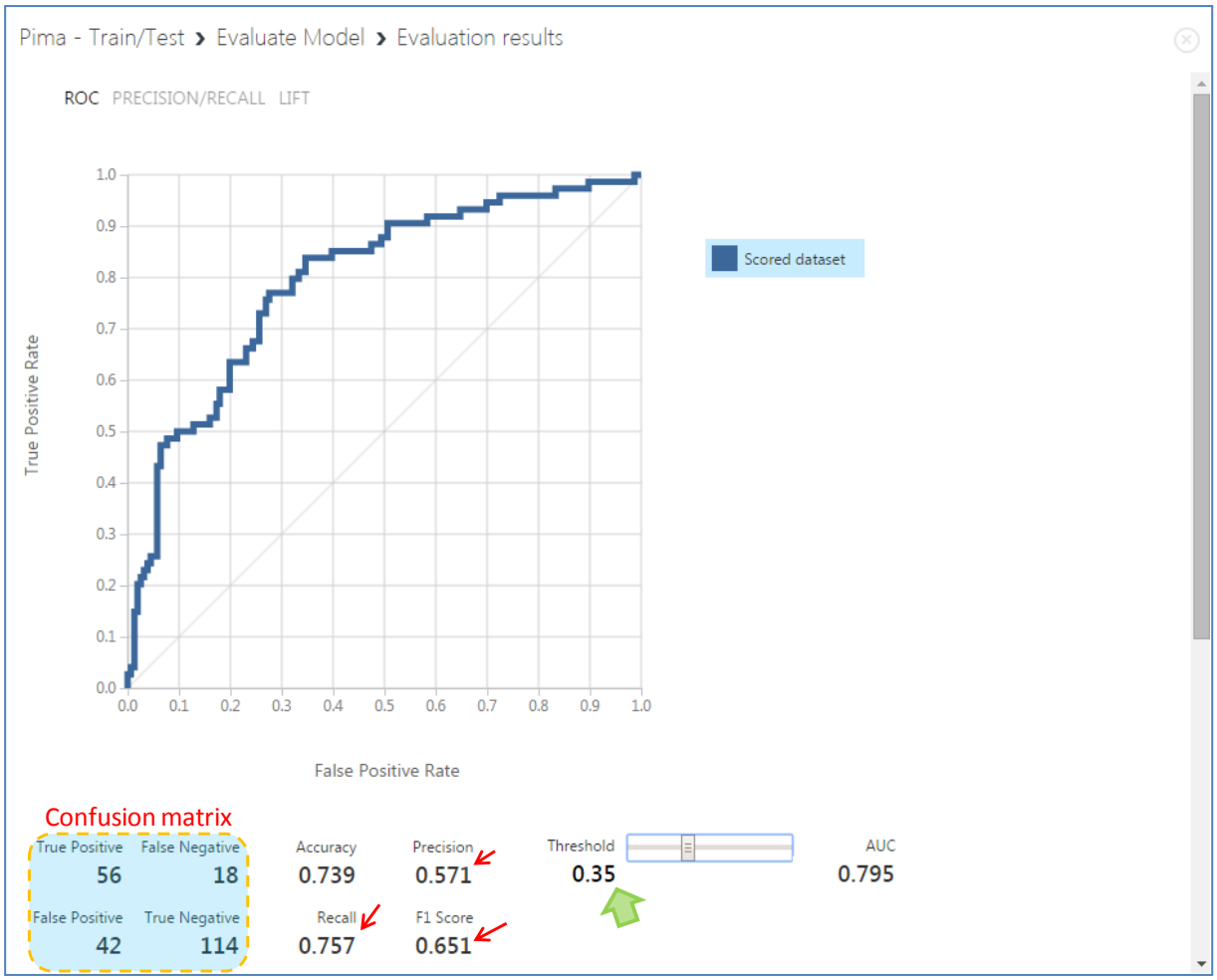


La matrice de confusion[5] et les indicateurs de performance sont calculés avec le seuil d'affectation usuel de **0.5**. Nous disposons du taux de succès (accuracy = 0.783), de la sensibilité (rappel = recall = 0.486), de la précision (precision = 0.750) et du F1-score (0.590).

Est directement proposée également la courbe ROC et calcule l'aire sous la courbe (AUC = 0.795). D'autres courbes sont disponibles (Precision / Recall, Lift).

La grande originalité de l'outil est qu'il est possible d'ajuster interactivement le seuil d'affectation. La matrice et les indicateurs sont automatiquement recalculés. Cela est particulièrement utile lorsque l'on souhaite effectuer des ajustements liés à des redressements (échantillon non représentatifs) ou à la prise en compte de coûts de mauvais classement.

[5] http://en.wikipedia.org/wiki/Confusion_matrix

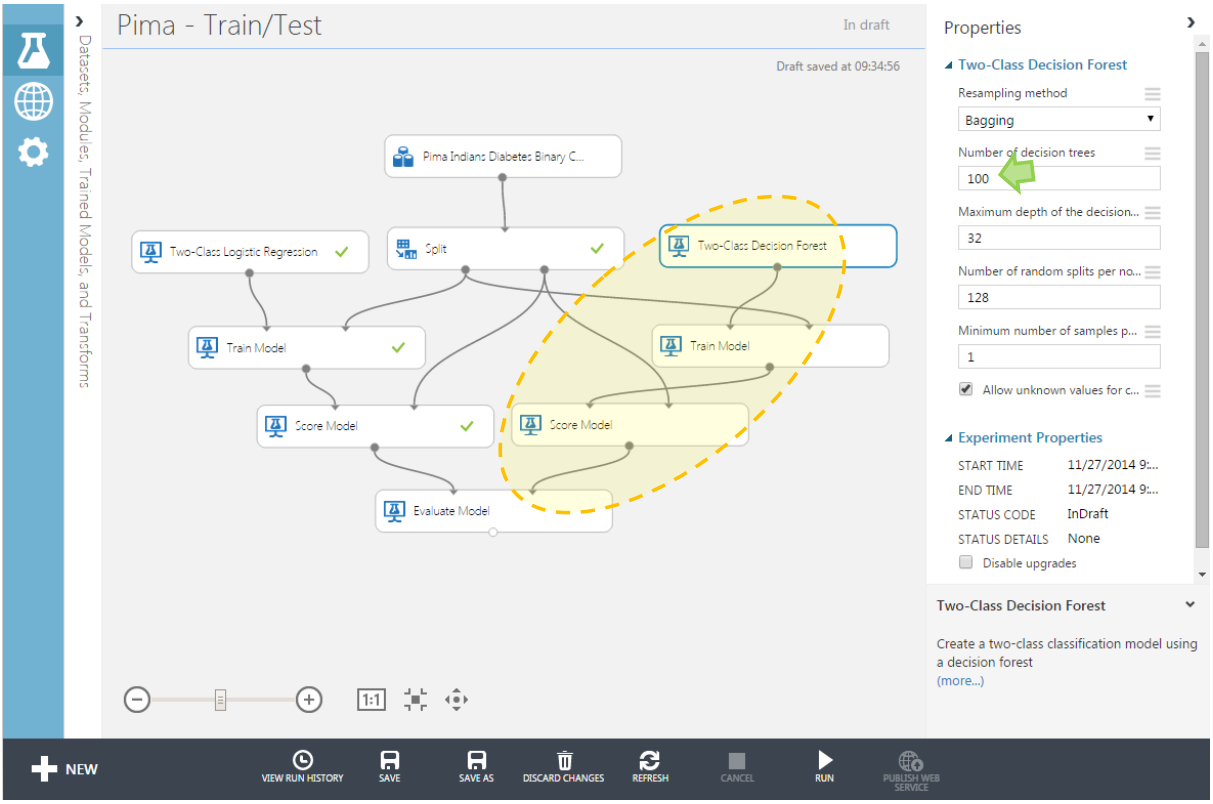


Le rappel s'améliore (0.75) lorsque l'on abaisse le seuil d'affectation à **0.35**, au détriment de la précision (0.571). Globalement, nous constatons que le F1-Score (0.651) s'améliore. La courbe ROC et l'aire sous la courbe ne sont absolument pas affectés par la modification du seuil bien évidemment.

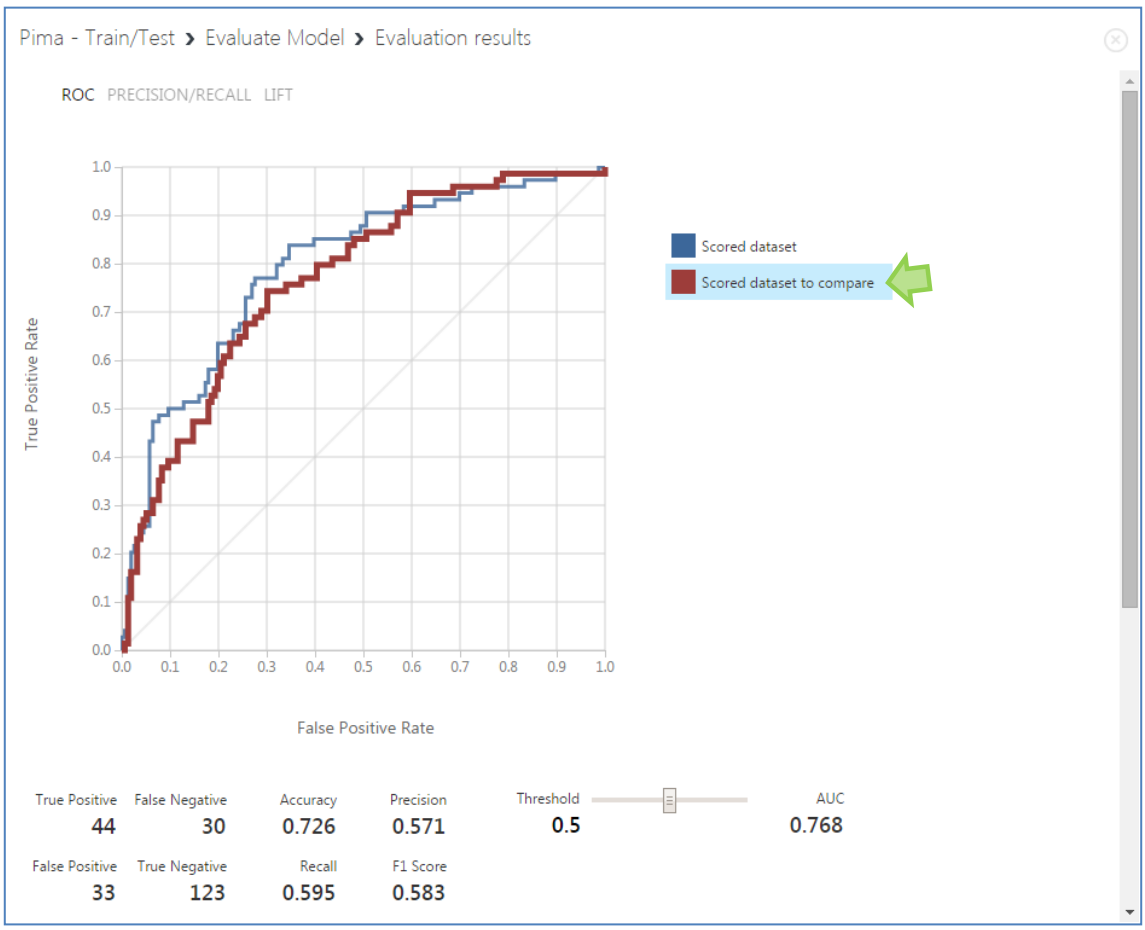
3.2 Comparaison de modèles

Construire le modèle le plus performant est l'une des principales tâches du data miner. Cela nous conduit à comparer différentes méthodes ou à essayer différentes valeurs des paramètres d'un algorithme donné[6]. Nous souhaitons évaluer le comportement d'un « Two-Class Decision Forest » (Number of Decision Trees = **100 arbres**) sur le fichier PIMA.

[6] L'outil **Sweep Parameters** (branche Machine Learning / Train) de Studio ML se charge de détecter automatiquement les bonnes valeurs des paramètres. Il procède par tâtonnement aléatoire ou en utilisant une grille pour systématiser la recherche. Nous pouvons déterminer l'indicateur à optimiser (taux de succès, F1-score, etc.). Le principe n'est pas sans rappeler la démarche des algorithmes wrapper pour la sélection de variables. Voir http://tutoriels-data-mining.blogspot.fr/2009/05/strategie-wrapper-pour-la-selection-de.html et http://tutoriels-data-mining.blogspot.fr/2010/01/wrapper-pour-la-selection-de-variables.html

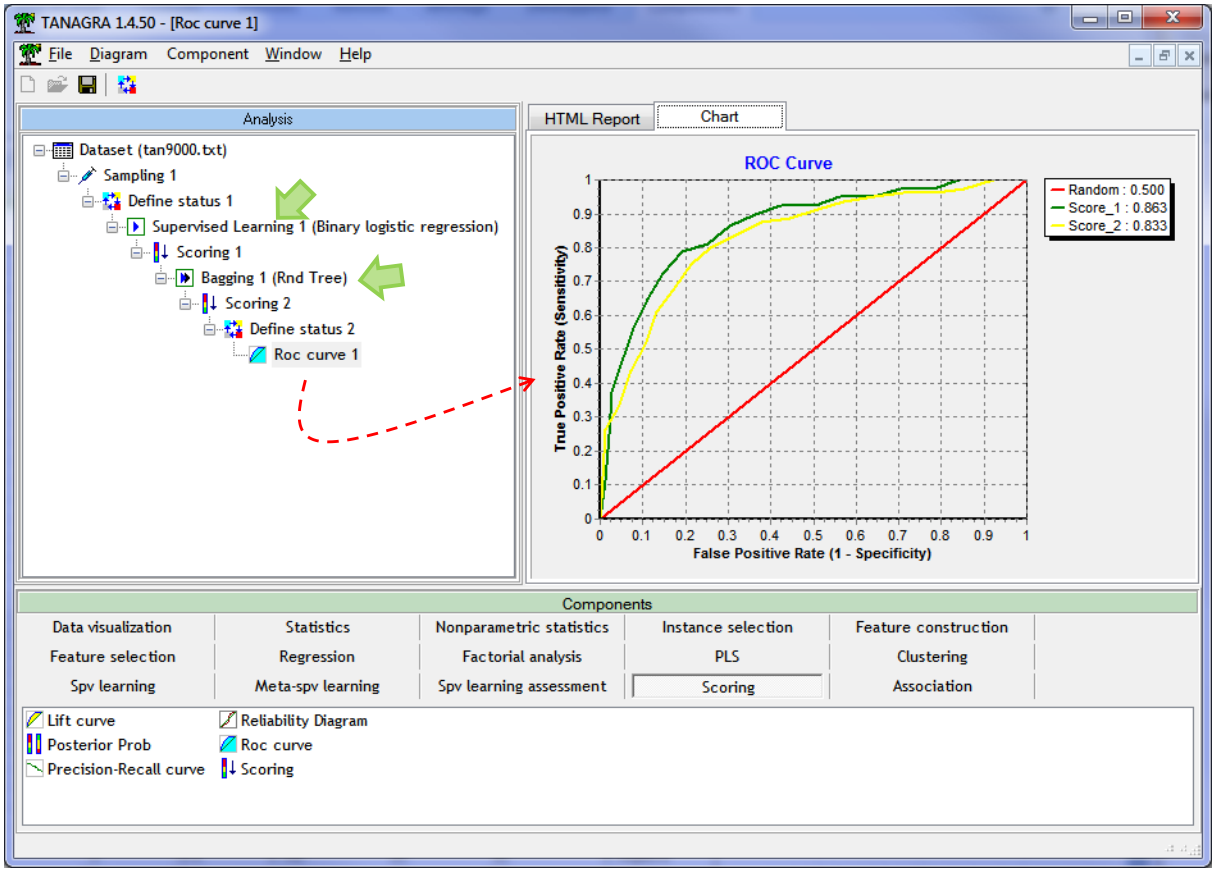


Après avoir lancé les calculs (clic sur **RUN**), nous visualisons la sortie de « **Evaluate Model** ».



Sur notre fichier PIMA, le Random Forest ne fait pas mieux finalement avec une AUC = 0.768.

A titre de comparaison, voici le diagramme correspondant aux traitements ci-dessus sous Tanagra. Nous distinguons la séquence : partition des données (Sampling), apprentissage + scoring pour les deux méthodes (Binary Logistic Regression, Bagging + Rnd Tree) et construction de la courbe ROC (Roc Curve).

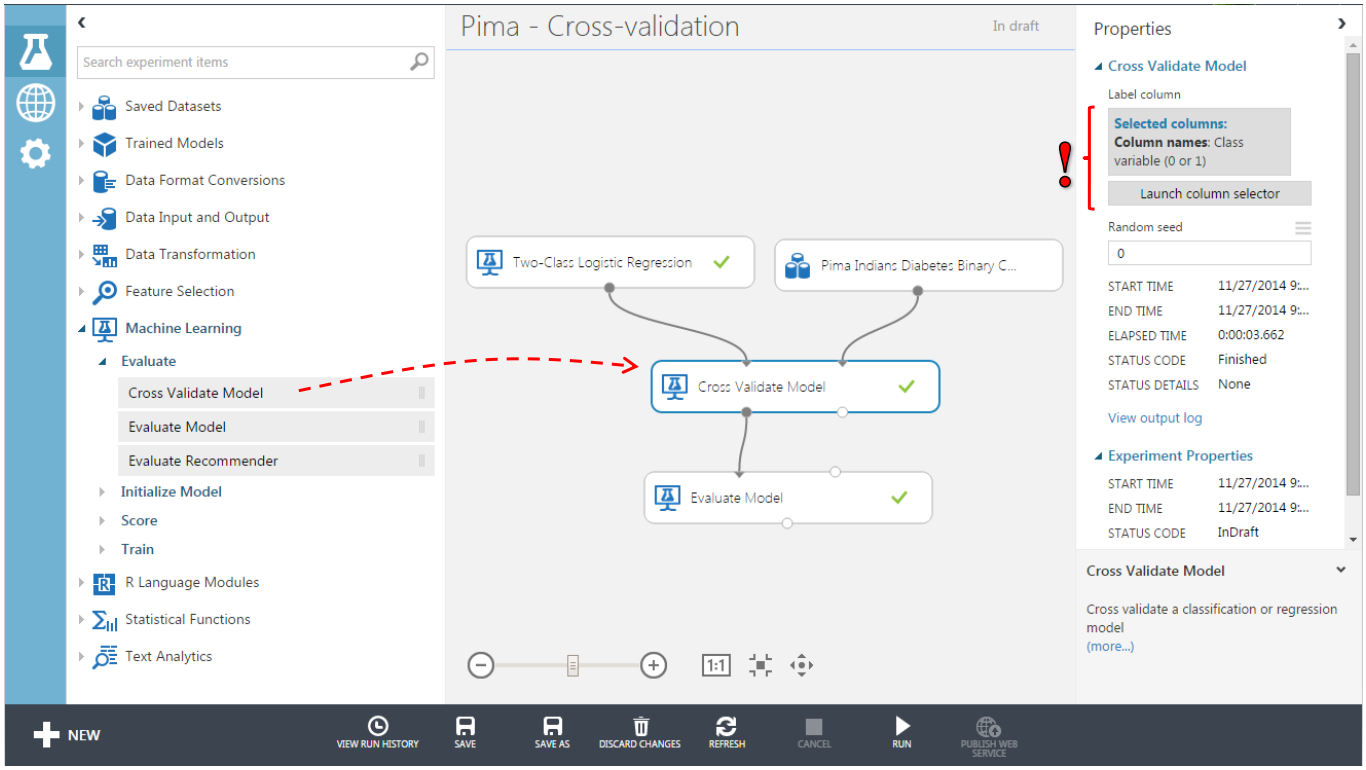


Par rapport à Azure ML, les résultats ne sont pas identiques parce que d'une part, même si nous avons adopté des proportions semblables, les échantillons d'apprentissage et de test sont différents ; d'autre part, les « Random Forest » intègrent une part d'aléatoire durant la modélisation.

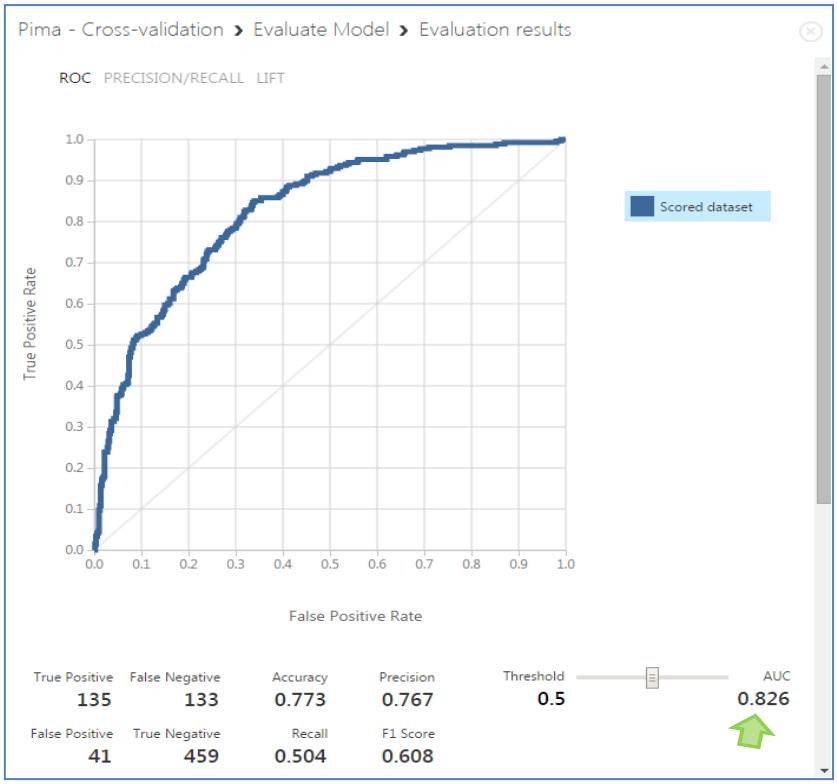
3.3 Evaluation par validation croisée des modèles

Lorsque le fichier est de taille réduite, il est préférable de passer par des techniques de ré-échantillonnage pour l'évaluation. Le schéma apprentissage-test risque de pénaliser la qualité du modèle en réduisant le nombre d'observations présentées à l'algorithme d'apprentissage. Nous créons un nouveau diagramme en suivant la même procédure que précédemment (section 3), nous traitons toujours le fichier PIMA.

Nous devons paramétrer **Cross Validate Model** (clic sur **Launch column selector**) pour préciser la variable cible (Selected Columns).



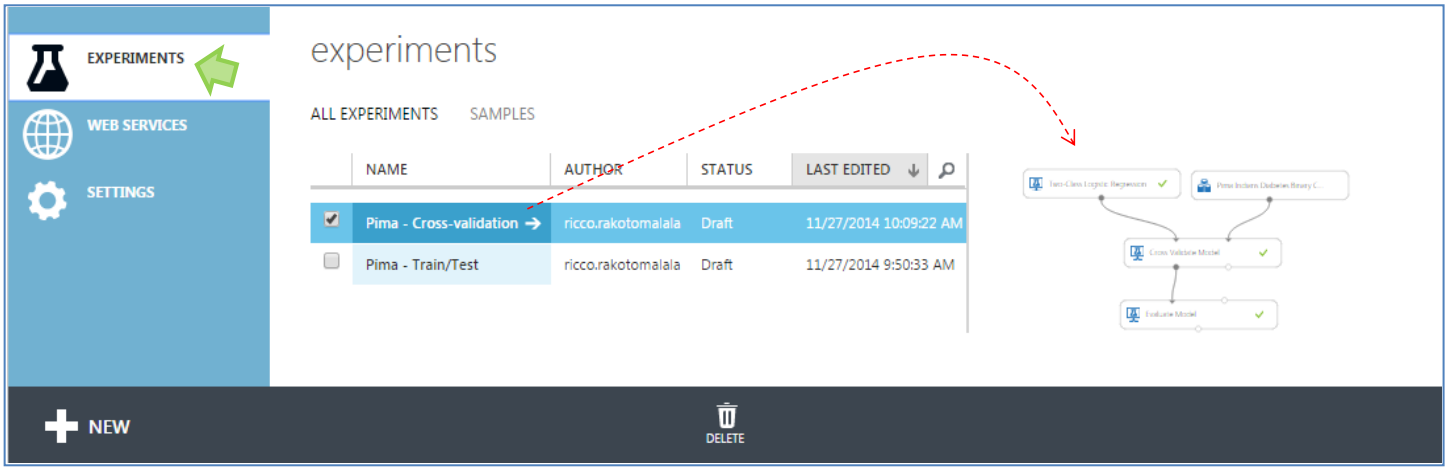
En sortie de « **Evaluate Model** », nous obtenons de nouveau une matrice de confusion et une courbe ROC, ainsi que les indicateurs associés, calculés en validation croisée.



L'AUC = 0.826 est un peu plus élevée qu'en apprentissage-test (0.975). Peut-être parce que l'algorithme utilise plus d'observations pour construire le modèle, on peut imaginer que cela induit un effet bénéfique. Mais, à mon sens, il faut surtout y voir une conséquence du travail sur échantillons de données : les résultats sont forcément empreints d'une certaine variabilité. Nous ne tirerons certainement pas de conclusions définitives à partir d'un écart de 3 centièmes.

3.4 Gestion des projets

En revenant dans l'espace « **Experiments** », nous distinguons les deux analyses que nous avons menées, avec une prévisualisation des diagrammes.



4 Typologie et utilisation de R

Dans cette section, nous évaluons les fonctionnalités d'Azure ML en matière de classification automatique (clustering en anglais). Nous mettons en œuvre la méthode des K-Means[7], adaptée au traitement des bases comportant un grand nombre d'observations.

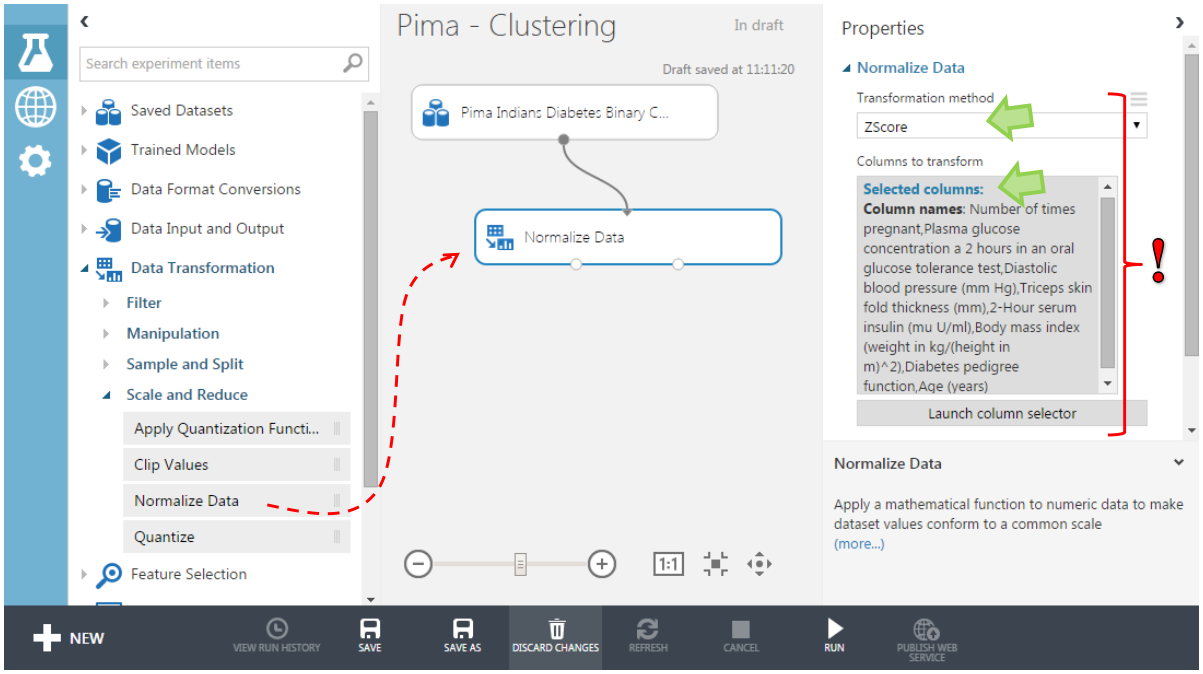
4.1 Construction des groupes

Nous travaillons de nouveau sur le fichier PIMA, en mettant de côté la variable « Class variable » que nous utiliserons en variable illustrative pour situer la nature de la partition proposée. Nous créons un diagramme vide que nous nommons « Pima Clustering » en cliquant sur le bouton NEW. Nous plaçons la base « Pima Indians Diabetes Binary Classification Dataset » (branche Saved Datasets) dans l'espace de travail.

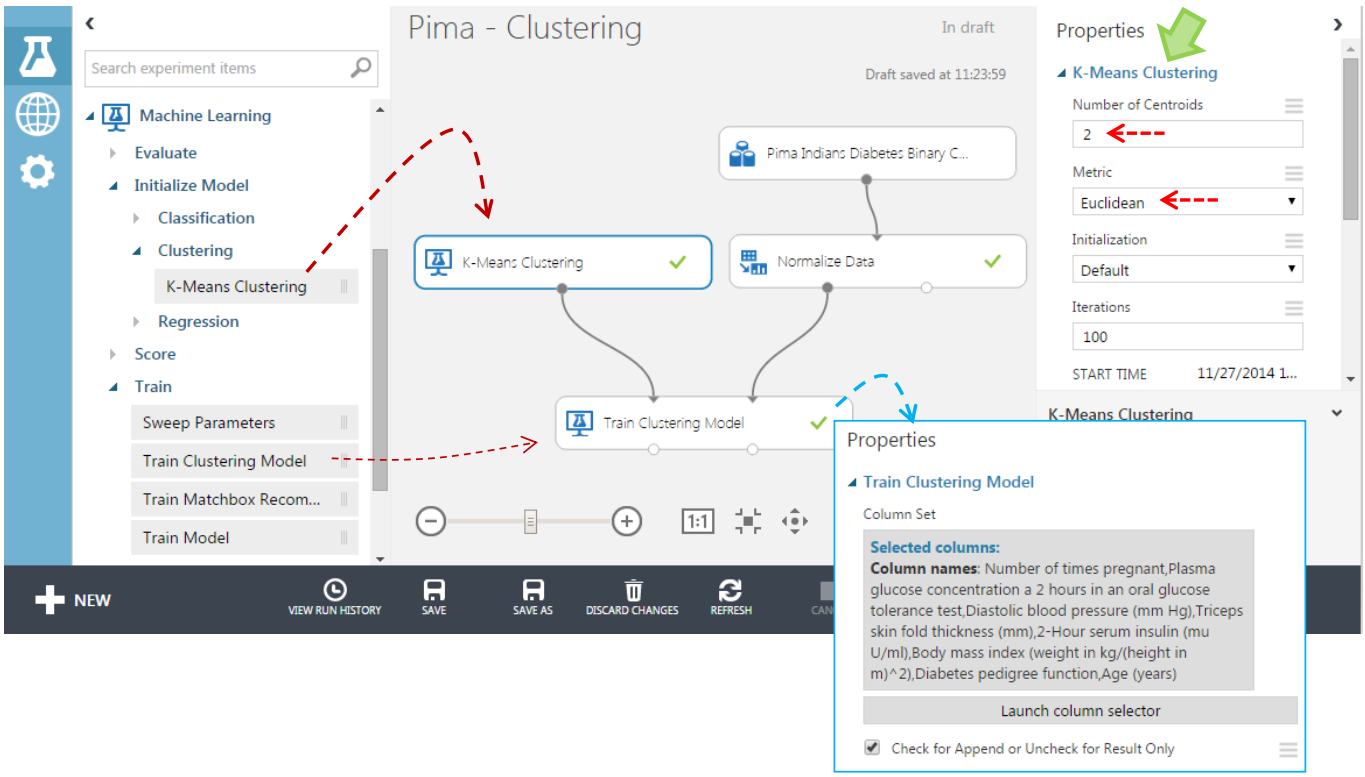
Les variables étant exprimées dans des unités différentes, il paraît approprié de les normaliser. Nous introduisons le composant « **Normalize Data** » (branche Data Transformation / Scale

[7] http://en.wikipedia.org/wiki/K-means_clustering

and Reduce). Dans les paramètres, nous choisissons la standardisation ZScore (Transformation method), et nous spécifions les colonnes à traiter (Launch column selector).



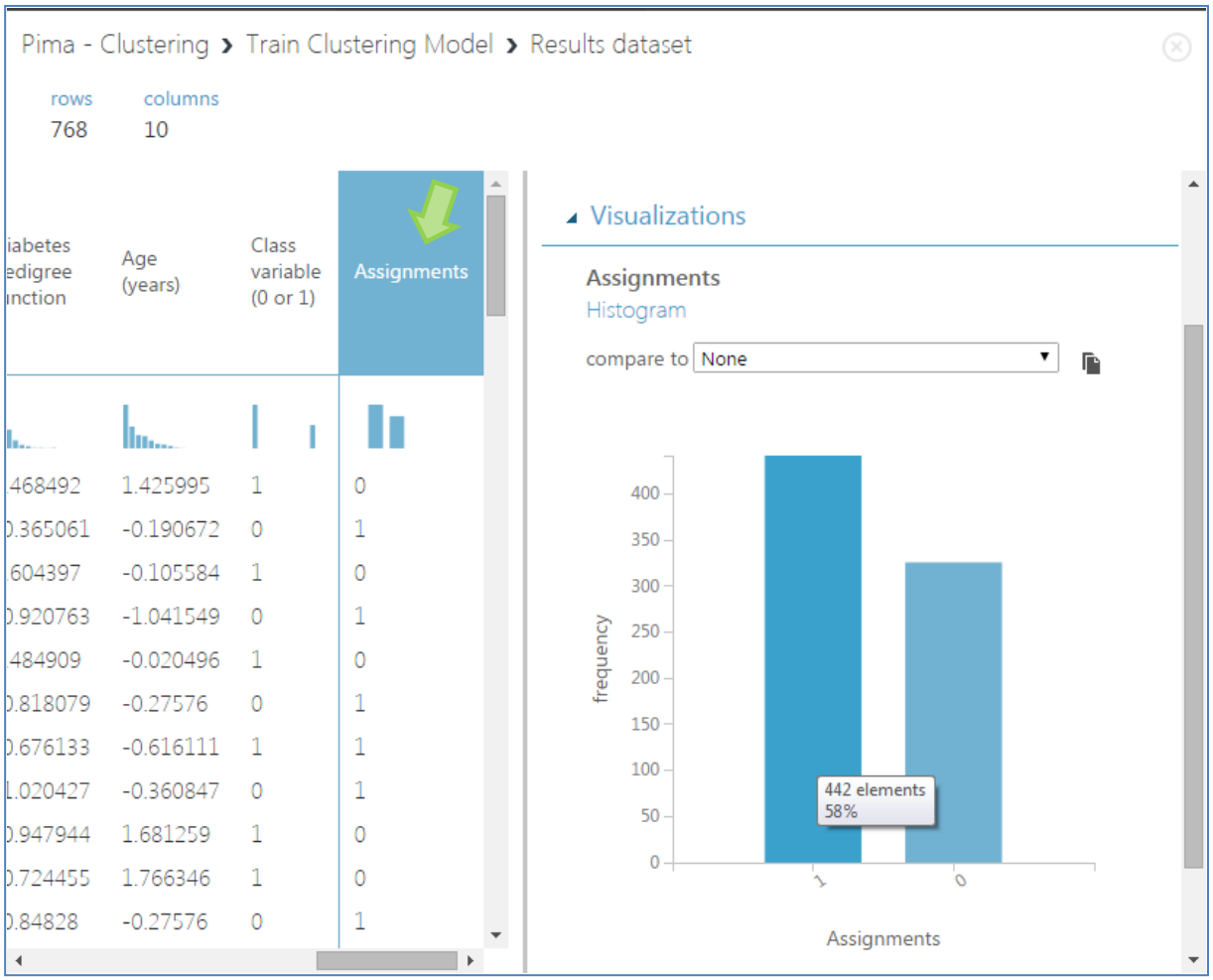
Nous appliquons les k-means (« **K-Means Clustering** ») aux données standardisées à travers le composant « **Train Clustering Model** ».



Il faut être attentif aux paramètres des composants :

- Pour les **K-Means**, nous demandons 2 groupes (Number of Centroids) et nous utilisons la distance euclidienne (Metric), d'où l'intérêt d'avoir standardisé les variables au préalable.
- Pour « **Train Clustering Model** », nous utilisons les variables précédemment transformées. Une nouvelle colonne est ajoutée à l'ensemble de données pour indiquer le groupe d'appartenance attribué par l'algorithme de classification (**Check for Append**).

Après traitement, l'ensemble de données avec la colonne additionnelle sont visibles dans la seconde sortie (**Results dataset**) du composant « Train Clustering Model ».



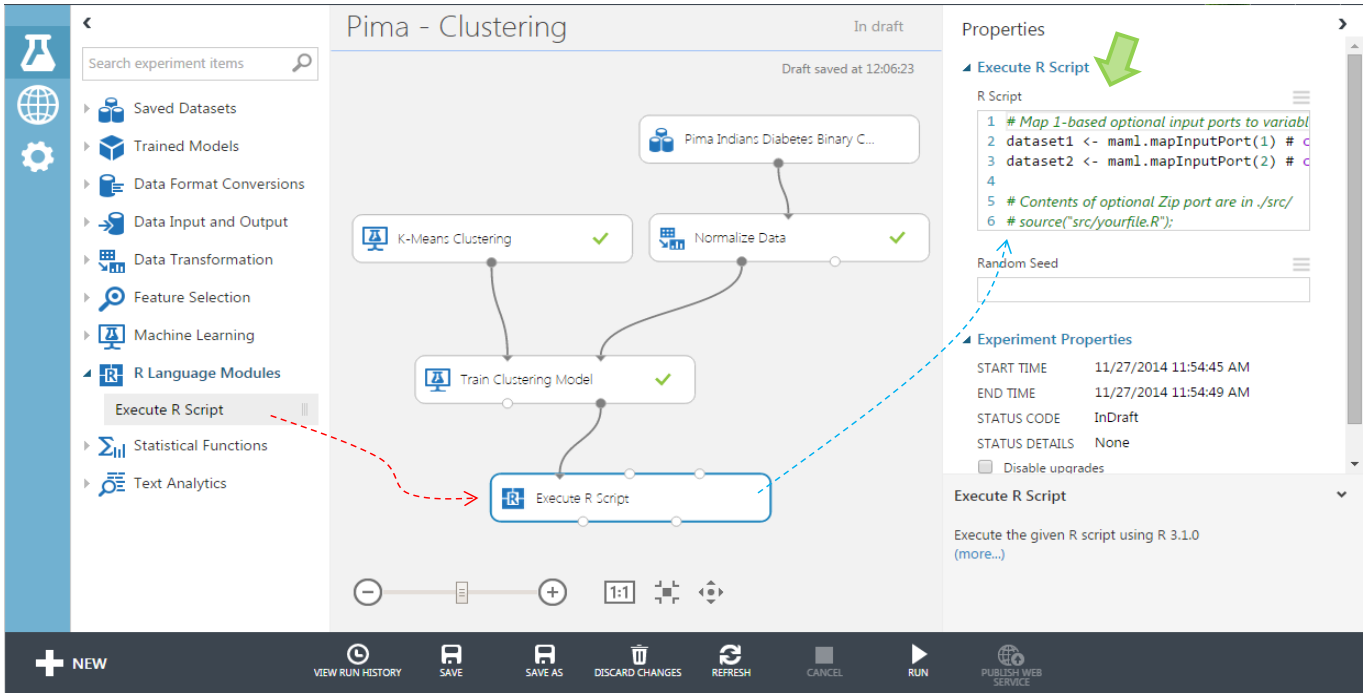
Nous distinguons les variables transformées, la colonne « class variable » et, surtout, la nouvelle colonne « **Assignments** » : 442 observations sont attribuées au groupe 1, le reste (768 – 442 = 326) au groupe 0.

4.2 Croisement avec la variable illustrative – Utilisation de R

A ce stade, une idée intéressante serait de confronter les groupes attribuées avec les classes d'appartenance initiales (diabète = positif ou négatif, cf. la colonne « class variable ») à l'aide d'un tableau croisé. On doit pouvoir le faire sous Azure ML (j'ai un peu cherché, je n'ai pas

trouvé), mais je me suis dit surtout que c'était là une opportunité intéressante d'introduire le composant « **Execute R Script** » qui permet d'exécuter du code R en ligne[8], sur l'ensemble de données que nous sommes en train d'analyser.

Nous l'insérons dans le diagramme et nous lui connectons les données en sortie de « Train Clustering Model » dont le contenu est visualisé ci-dessus. Un éditeur avec du code déjà pré-écrit apparaît. Il indique les identifiants à utiliser pour accéder aux données.

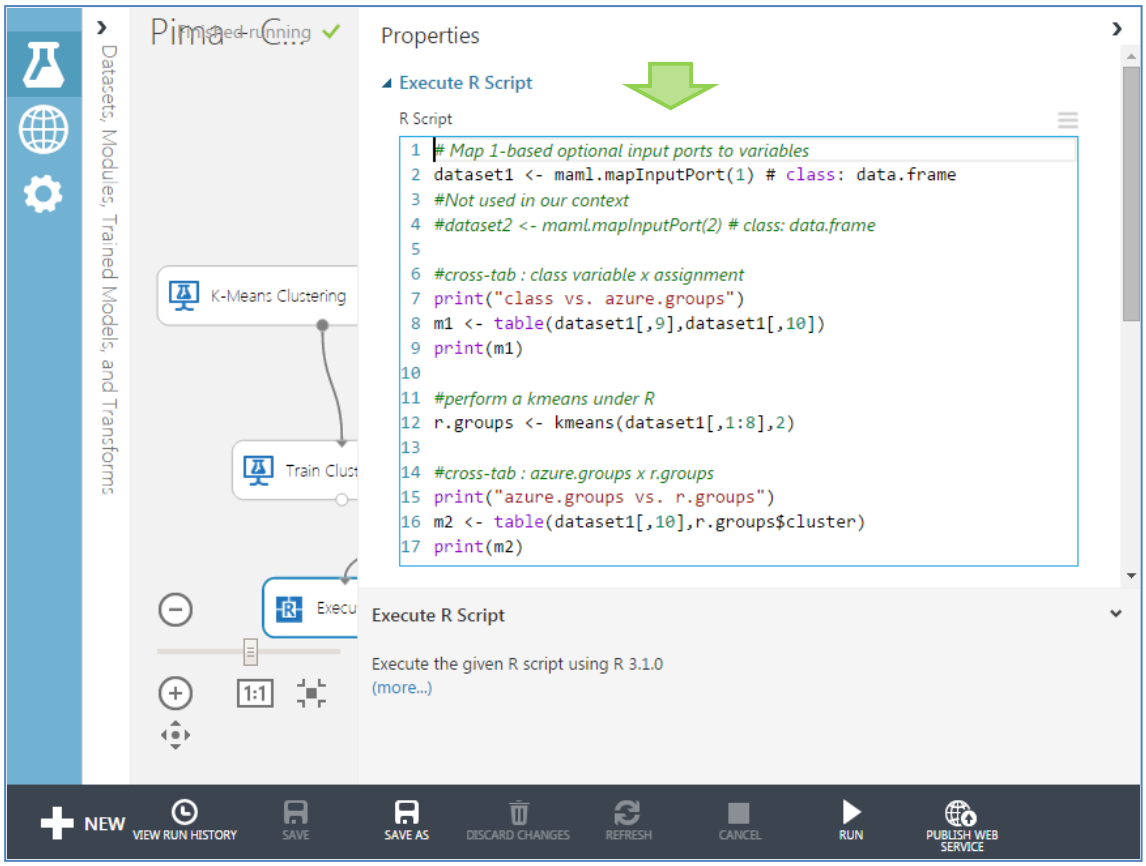


Il ne nous reste plus qu'à le compléter. Ci-dessous le code rédigé :

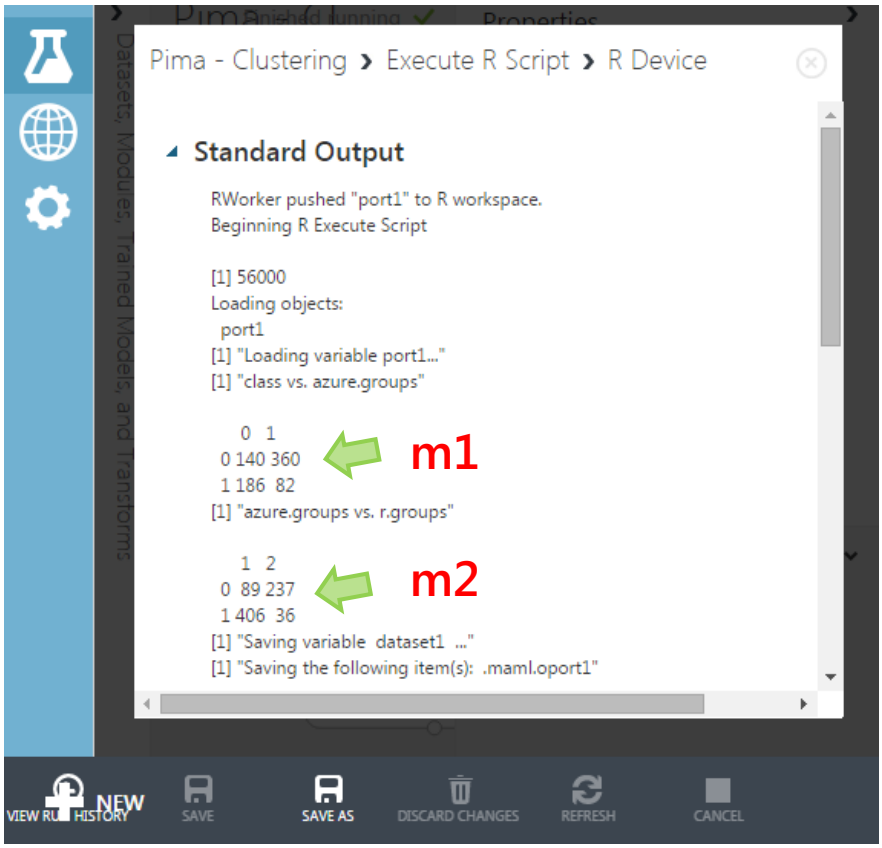
- « dataset1 » représente les données en entrée du composant.
- Il est possible de brancher une seconde source, nous n'utilisons pas cette fonctionnalité dans notre exemple, nous avons mis en commentaire le code correspondant.
- Nous croisons la « variable class » avec « assignment ».
- Nous réalisons un kmeans avec la procédure dédiée de R (package « stats »).
- Nous croisons les groupes attribués par Azure ML et par R.

On perçoit aisément les immenses possibilités de cette fonctionnalité. Nous pourrions calculer les PHI-2 à partir des tableaux de contingence, réaliser des représentations graphiques, etc. L'affaire est d'autant plus intéressante que nous pouvons spécifier les données en sortie du composant avec la commande **maml.mapOutputPort(.)**. Elles peuvent être exploitées dans la suite du diagramme sous ML Studio.

[8] J'ai désinstallé R de ma machine (argh...) pour être sûr qu'aucune ressource locale n'est exploitée.



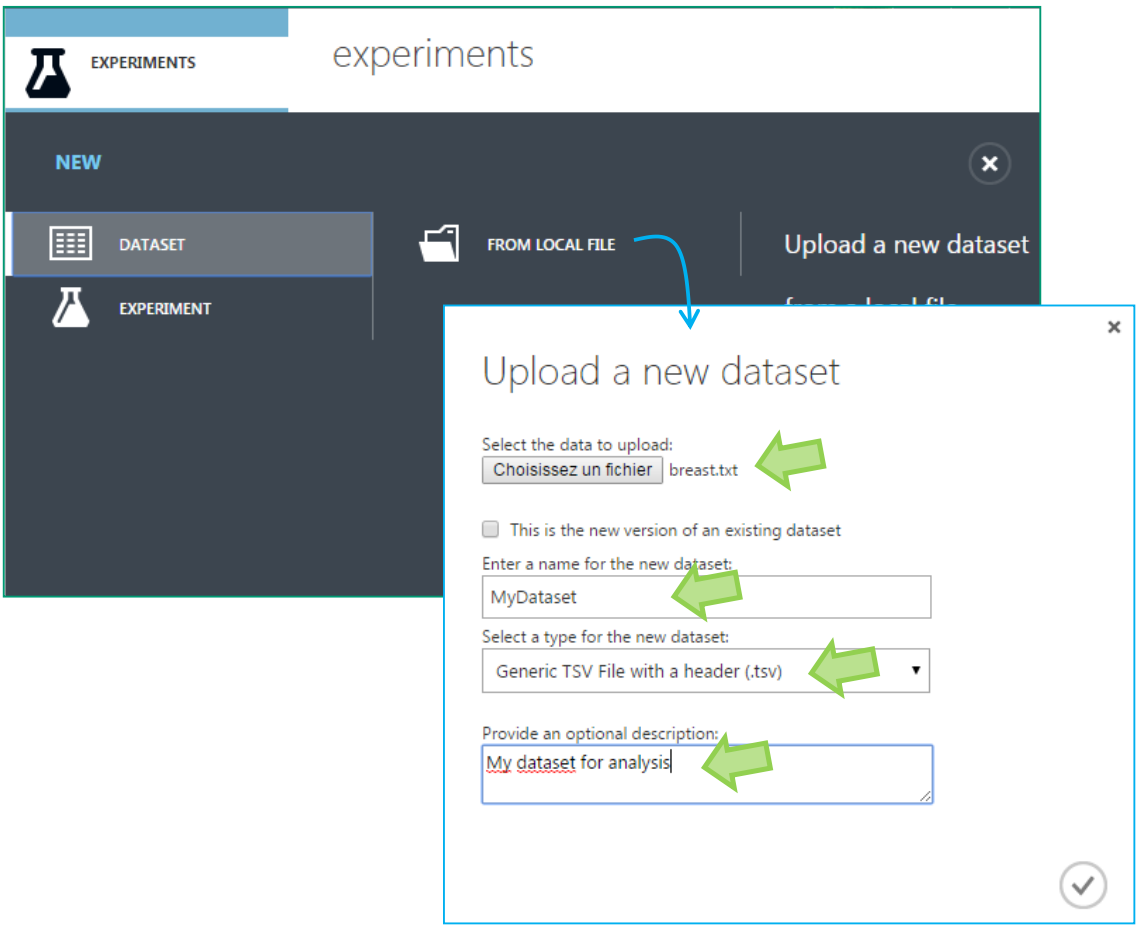
En cliquant sur la sortie « **R Device** », nous accédons à la fenêtre de résultats.



5 Travailler sur nos jeux de données

Travailler avec nos propres données rend la démonstration plus convaincante. Dans cette section, nous montrons comment importer les données. Nous bénéficions d'un certain volume de stockage malgré que nous nous soyons connectés avec un compte gratuit.

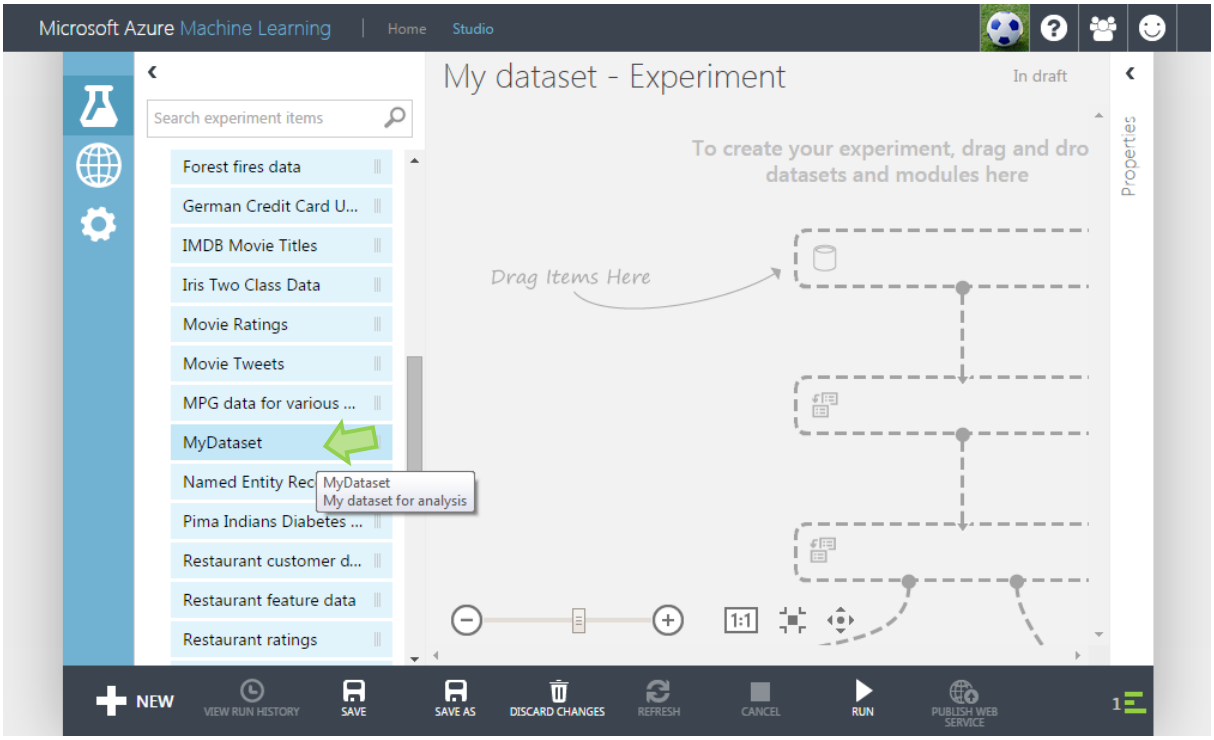
A partir de la fenêtre d'accueil, nous activons NEW. Nous choisissons l'option **DATASET** cette fois-ci. Nous cliquons sur l'icône « **From Local File** ».



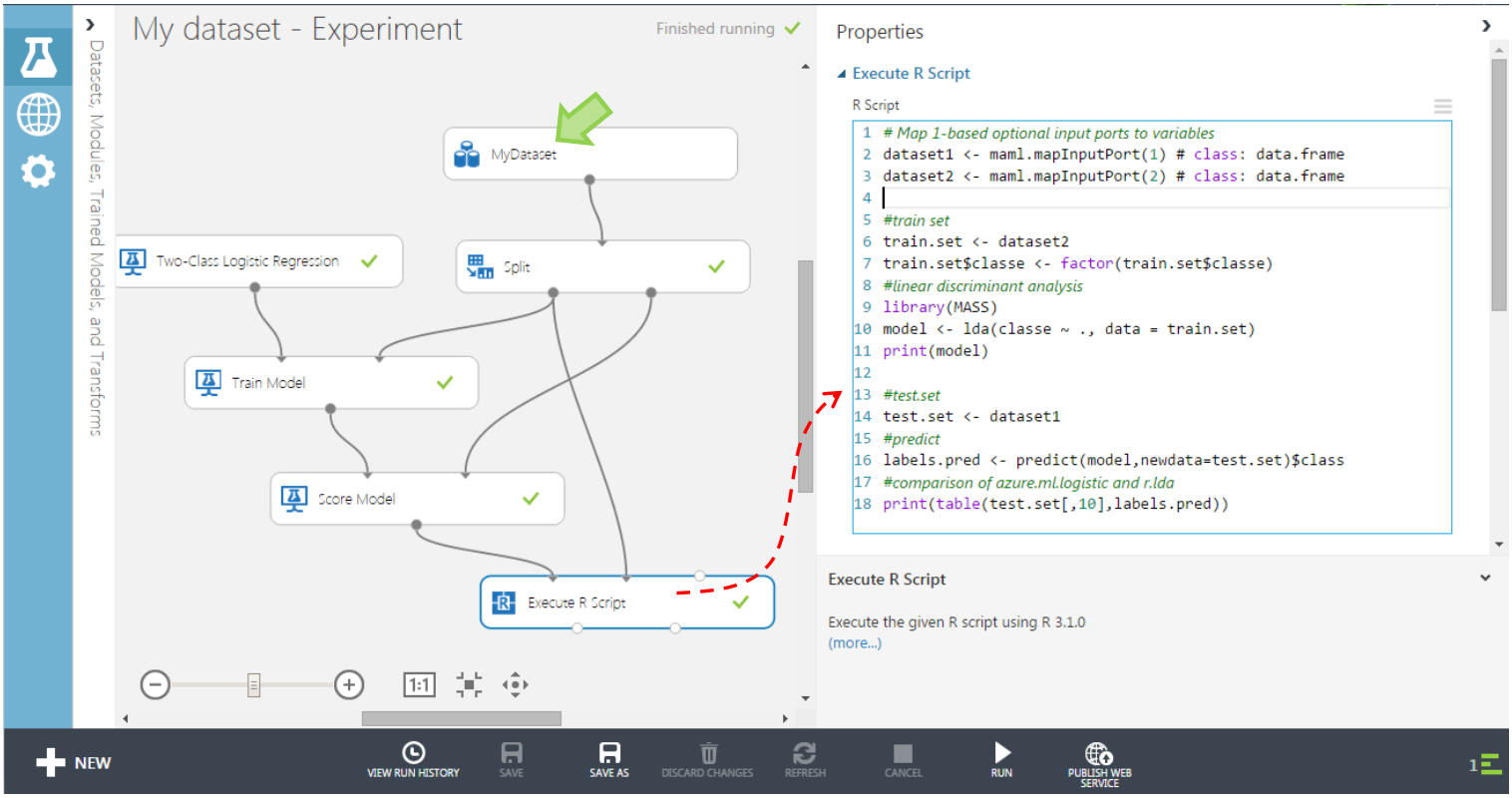
Avec la version gratuite, il n'est pas possible de supprimer un ensemble de données intégré dans la base interne. Nous avons donc intérêt à utiliser des noms génériques (ex. **MyDataset** dans mon exemple), et à écraser au fur et à mesure le dataset pour pouvoir prolonger notre expérimentation à d'autres problématiques (voir l'option de chargement « **This is the new version of an existing dataset** »).

Le fichier « breast.txt »[9] est au format texte avec séparateur tabulation, la première ligne correspond aux noms des variables, d'où le choix du type « Generic TSV File with a header ».

[9] http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29 ; la dernière colonne est la cible.



Une fois la base importée, elle apparaît parmi les « Saved dataset ». Nous pouvons dès lors travailler comme nous l'entendons. Dans le diagramme ci-dessous, nous comparons les prédictions de la régression logistique d'Azure ML et de l'analyse discriminante de R (lda).



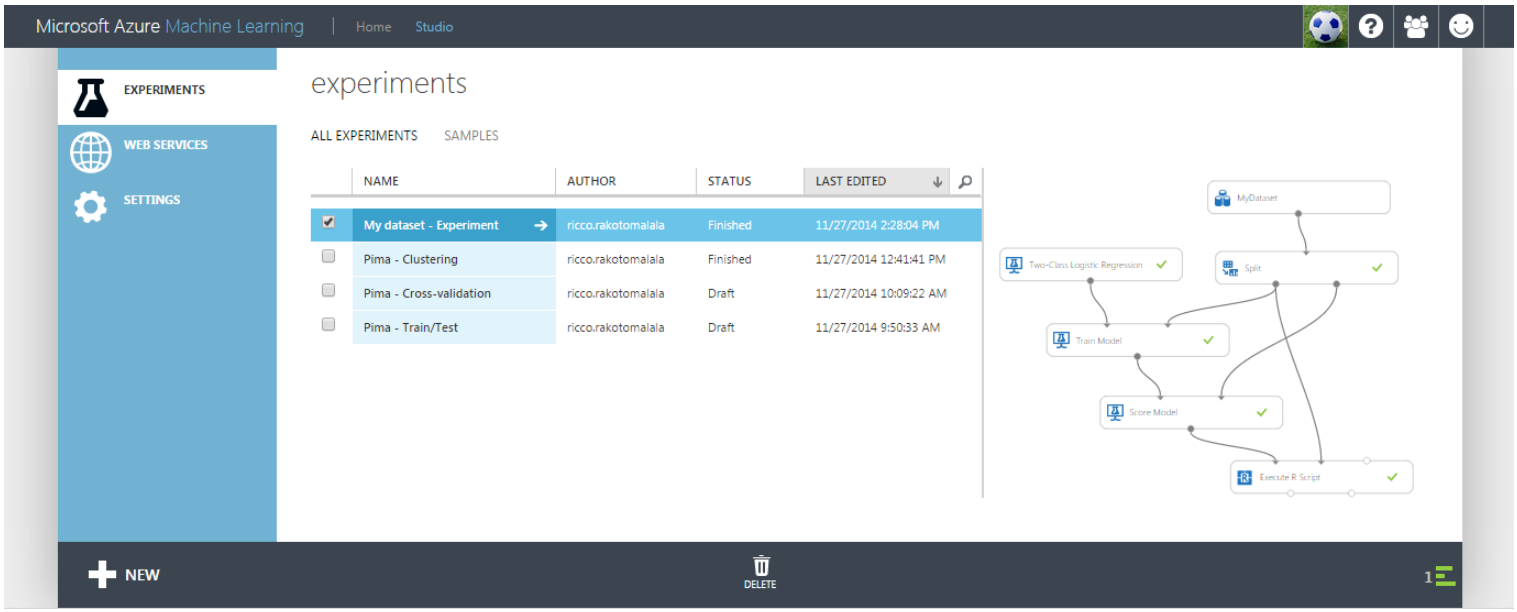
Ces deux classifieurs linéaires sont très fortement cohérents sur le fichier « breast.txt ».

```
              Chargement des données via les ports d'entrée
[ModuleOutput] [1] "Loading variable port1..."
[ModuleOutput] [1] "Loading variable port2..."
[ModuleOutput]
               Résultats de l'analyse discriminante (lda)
[ModuleOutput] Call:
[ModuleOutput]
[ModuleOutput] Prior probabilities of groups:
[ModuleOutput]    begnin malignant
[ModuleOutput] 0.6523517 0.3476483
[ModuleOutput]
[ModuleOutput] Group means:
[ModuleOutput]             clump ucellsize ucellshape mgadhesion   sepics  bnuclei bchromatin
[ModuleOutput] begnin    3.009404  1.382445   1.504702   1.416928 2.141066 1.438871   2.109718
[ModuleOutput] malignant 6.900000  6.417647   6.505882   5.676471 5.176471 7.611765   5.817647
[ModuleOutput]
[ModuleOutput]           normnucl  mitoses
[ModuleOutput] begnin    1.319749 1.078370
[ModuleOutput] malignant 5.947059 2.488235
[ModuleOutput]
[ModuleOutput] Coefficients of linear discriminants:
[ModuleOutput]
[ModuleOutput]                     LD1
[ModuleOutput] clump      0.17812989
[ModuleOutput] ucellsize  0.11636812
[ModuleOutput] ucellshape 0.06856035
[ModuleOutput] mgadhesion 0.01588687
[ModuleOutput] sepics     0.03025901
[ModuleOutput] bnuclei    0.27951837
[ModuleOutput] bchromatin 0.11701760
[ModuleOutput] normnucl   0.12118861
[ModuleOutput] mitoses    0.06734434
[ModuleOutput]
                      Comparaison des prédictions
[ModuleOutput]            labels.pred
[ModuleOutput]             begnin malignant
[ModuleOutput]   begnin       137         2
[ModuleOutput]   malignant      4        67
[ModuleOutput]
```

Les deux modèles ne diffèrent que sur 6 individus : 4 prédits « malignant » par la régression logistique sont prédits « begnin » par l'analyse discriminante ; inversement pour 2 autres.

6 Bilan des projets

Au final, voici les 4 projets que nous aurons traités dans ce tutoriel :



7 Conclusion

J'en ai rêvé, Microsoft l'a fait. C'est un outil très intéressant qu'on nous propose là. Au-delà de l'externalisation du stockage de données et des ressources de calcul, on imagine aisément les portes qu'ouvrent les technologies cloud mises en œuvre en back office, en matière de volumétrie et de capacité de calcul. Et on pilote tout ça via un simple navigateur…

Nous noterons que la bibliothèque de méthodes est somme toute peu fournie par rapport aux logiciels usuels de Data Mining. Cela montre surtout que Microsoft entend se concentrer sur les outils de productivité. Une des principales finalités du dispositif est le déploiement des modèles sous forme de service web[10]. De fait, les outils exploratoires graphiques sont peu développés.

Placé dans un tout autre contexte, Azure ML constituerait un formidable dispositif pour l'enseignement et la recherche. Le travail collaboratif peut être grandement facilité par le partage des espaces de travail[11]. La coordination dans les projets de recherche, les projets étudiants, ou même les travaux dirigés, n'en sera que meilleure. Voilà une piste qui me paraît pleine de promesses.

[10] Voir http://social.technet.microsoft.com/wiki/contents/articles/26689.predictive-analytics-with-microsoft-azure-machine-learning.aspx, sections 5 et 6.

[11] http://help.azureml.net/Content/html/20ff2e97-6a44-4312-88b6-64457f0583d8.htm