Diabetes Prediction: Using Three Different Data Mining Tools and Categorical Data

Jam Ayub
Department of Computer Science
Forman Christian College (A Chartered University)
21-11482@formanite.fccollege.edu.pk

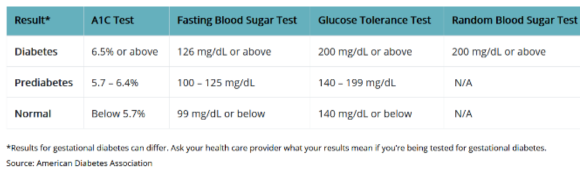
ABSTRACT

Diabetes is a disorder which is related to metabolic system. It is a multifactorial disease. In today's world of modern time this is one of the disease that causes the most deaths in the world. For these reasons use data mining tools will help in predicting results to useful analysis. Therefore, the present study aimed to compare RapidMiner, WEKA and python scripting and their distinction in terms of predicting diabetes. The dataset used in this study is from National Institute of Diabetes and Digestive and Kidney Diseases. This Dataset is used for the prediction of onset of diabetes within five years. After attributes selections on a decision tree, model selection, applying model, and testing a model we are able to predict Diabetes with high accuracy. From all the tree analytical tools, RapidMiner gives more detailed and fast analysis of the dataset. From Decision Tree, k-NN, Random Forest, Naïve Bayes and Rule Induction algorithms, Random Forest outperform the most. This is confirmed from RapidMiner analysis, WEKA and Python coding script. Overall, Random Forest provides the better classification for the prediction of PRIMA Diabetes dataset. Area under the curve of ROCs is also very large among the other algorithms. WEKA is good for beginners for data analysis from a UI tool but for a professional expert RapidMiner is best because provides very detailed and fast analysis. Python scripting is very helpful for those how are very new to data mining.

Keywords: RapidMiner, WEKA, k-NN, Decision Tree, Random Forest, Naive Bayes, Rule Induction

1. INTRODUCTION: Both developed and developing countries diabetes is the common diseases in and sometimes considered as the most common cause of death in the whole world. It is one of the regular sicknesses that focus on the old population around the world. As per the International Diabetes Federation, 451 million individuals largeness the world were diabetic in 2017. The assumptions are that this number will increment to stunning 693 million individuals in the coming 26 years [6]. Diabetes is considered as a constant sickness related to an unwont condition of the human spirit where the degree of pure breed glucose is conflicting because of some pancreas brokenness that prompts the creation of next to zero insulin by any stretch of the imagination, making diabetes of type 1 or cells wither impervious to insulin, causing diabetes of type 2 [4][5].The principle legitimization of diabetes stays obscure, yet researchers accept that both hereditary variables and the natural way of life assume a significant part in diabetes. Despite the fact that it's serious, it very well may be overseen by treatment and prescription. People with diabetes vagrant a danger of fostering some auxiliary medical problems like heart illnesses and nerve harm. Hence, early recognition and treatment of diabetes can forestall confusion and assistant in diminishing the danger of extreme medical conditions. Numerous analysts in the bioinformatics field have endeavored to compose this sickness and attempted to make frameworks and instruments that will help in diabetes forecast. They either constructed expectation models utilizing variegated sorts of AI calculations like classification or connotation calculations. Choice Trees, Support Vector Machine (SVM), and Linear Regression were the most overall calculations [3] [7].

HbA1c is the most cut point to diagnose diabetes. It's a glucose test. If its level goes less than 6.5does not exclude diabetes diagnosed. There are many methods to diagnose diabetes HbA1c test is one of them. FPG (Fasting Plasma Glucose) test is also used to diagnose it. There are some AI methods used in it to diagnose it. The samples of the blood glucose are used for it. Below are some tests and their normal values to predict diabetes.



Result*	A1C Test	Fasting Blood Sugar Test	Glucose Tolerance Test	Random Blood Sugar Test
Diabetes	6.5% or above	126 mg/dL or above	200 mg/dL or above	200 mg/dL or above
Prediabetes	5.7 – 6.4%	100 – 125 mg/dL	140 – 199 mg/dL	N/A
Normal	Below 5.7%	99 mg/dL or below	140 mg/dL or below	N/A

*Results for gestational diabetes can differ. Ask your health care provider what your results mean if you're being tested for gestational diabetes.
Source: American Diabetes Association

Figure 1: Results of some medical diagnosis test

In past times diabetes is diagnosed by a urine test. There were people known as water tasters. They taste urine, if its taste is sweet then a person has diabetes. Now there are technologies and AI methods to diagnose it. Deep learning and machine learning techniques are often appeared to be used for diagnose of diabetes. This requires a lot of human effort and time to extract the result of this feature. The deep learning approach is used to overcome the limitations of the Machine learning approach.

AI is a blossoming new innovation with a wide scope of uses. It can possibly turn into one of the vital parts of canny data frameworks, empowering minimized speculations, construed from huge data sets of recorded data, to be applied as information in different commonsense manners, for example, is inserted in programmed measures like master frameworks, or utilized straightforwardly for speaking with human specialists and for instructive purposes [8].

Bayesian neural networks were applied to this dataset utilizing similar erasures and preparing tests. Standard neural networks organization had an exactness of 75.35%, the Bayesian methodology 79.45% [1]. ARTMAP-IC neural networks organization added disseminated forecast and class example checking to the fundamental fluffy ARTMAP framework. It accomplished and the exactness of 81%, with ARTMAP at 66%, k-NN 77%, and calculated relapse 77% [2]. Khan utilized without multiplier feed forward networks (MFN) and effectively noticed that 49% of the patients had zero qualities for factors that can't be zero. By the by, he chose the entirety of the diabetic patients and an equivalent number of non-diabetics to achieve a decent set however with missing factors. It isn't evident whether the non-diabetics were haphazardly chosen or chosen to limit missing factors. Half of these is took by him as a reasonable subgroup (n=268) as a preparation set and normalized the 8 factors to zero mean and unit fluctuation. He figured exactness for the MFN and furthermore, discrete weight organizations (DWN) and consistent weight organizations (CWN) utilizing the n=268 assessment set and results were 78.0%, 76.9%, and 78.4% separately [8].

Liu coordinated order and affiliation rule mining in class affiliation rules (CAR) that were applied to the informational collection. The 4 CAR models, from the best 4 shows exactness of 73.10% contrasted with 75.45% for C4.5 rules. Lord et al. utilized 14 calculations on the PIDD. They disposed of the insulin variable with the most missing cases, leaving n=532. The exactness of the information mining apparatuses utilized was CART 76.0%, See5 73.0%, Scenario 30.0%, SPlus 79.0%, KnowledgeMiner 78.0%, DataMind 69.0%, WizWhy 74.10%, DMSK 67.21%, PcOLPARS 81.11%, NeuroShell2- -Neural 77.12%, PRW 80.02%, NeuroShell2—

PolyNet 78.09%, MQ Expert 77.04%, and Gnosis 81.32%. Characterization by totaling arising designs (CAEP) applied to the PIDD was at first 72.00% precise, yet could just recognize 30.00% of the diabetic patients. After adjustments, it was 75.0% precise [8].

Exactness for anticipating diabetic status goes from 66% to 81%. While a portion of these are method for a bigger gathering of randomizations, most are essentially one randomization into a preparation set and test set showing up at exactness. This risk an especially positive or negative exactness being an eccentricity of that specific randomization as opposed to the technique utilized. Taking a gander at the 392 complete cases, speculating all are non-diabetic gives an exactness of 65.1%. Since 1988, numerous many distributions utilizing different calculations have brought about exactness paces of 66% to 81%. Harsh sets as an information mining prescient instrument has been utilized in clinical regions since the last part of the 1980s, however not applied to the PIDD to our information [8].

All The aforementioned techniques have a limitation that is their complexity. Due to its intricacy we are using ML approach by datasets through which diabetes can be predicted as +ve or –ve by the variables provided to the algorithm. Certain limitations provided to given variable from the dataset, if the input values are excluded by them then it predicts that a person is diabetic or not.

1.1 Methods: For the scope of this project, we are analyzing two well-known data mining tools which are RapidMinner Studio and WEKA. The use of these tools will help us to understand more about the data that we have. The major benefit of using this kind of tool is that we do not have to spend much of our time in coding the program instead we are spending more time in analyses and predicting more efficient business decisions. This saves much of our time and effort. This tool will give you the same experience as coding with very little effort which will help data scientists to make better decisions rapidly and efficiently with many visualizing techniques. We will see down the road which data mining tool will be easier to use, give more presentable results, and in-depth analysis and coding grip over the algorithms we are going to use.

1.2 Description: The dataset is taken from kaggel and this is from the National Institute of Diabetes and Digestive and Kidney Diseases. This dataset is used for analysis of patient that has diabetes or not.

1.3 State the problem: This dataset has the onset prediction of diabetes within 5 years with given medical details in Pima Indians.

1.4 What are the input variables and what are the output variables?

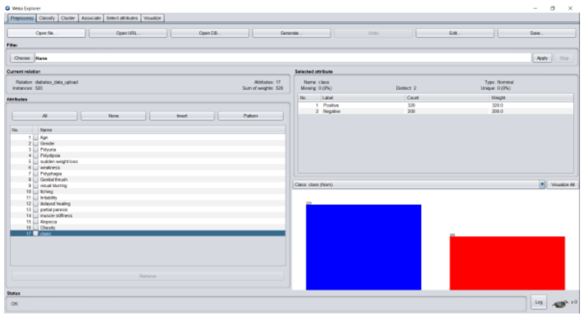
It is a binary (2-class) classification problem. There are 520 observations with 17 input variables and 1 output variable. The variable names are as follows:

Age	Numeric (years)
Gender	Nominal (yes/no)
Polyuria	Nominal (yes/no)
Polydipsia	Nominal (yes/no)
Sudden weight loss	Nominal (yes/no)
Weakness	Nominal (yes/no)
Polyphagia	Nominal (yes/no)
Genital thrush	Nominal (yes/no)
Visual blurring	Nominal (yes/no)
Itching	Nominal (yes/no)
Irritability	Nominal (yes/no)
Delayed healing	Nominal (yes/no)
Partial paresis	Nominal (yes/no)
Muscle stiffness	Nominal (yes/no)
Alopecia	Nominal (yes/no)
Obesity	Nominal (yes/no)
Class	Output variable (Nominal (positive/negative))

if patient has diabetes then the output will be positive and negative if not.

2. ANALYSES USING WEKA

Some of the popular algorithms you've heard about like logistic regression, support vector machine, decision tree, and neural network have to do with data mining. You just simply want to get your feet wet with WEKA to see how easy it is. You can get to the initial screen in WEKA by opening your data file just like the fig_1. Now you'd find it helpful to see examples of data pre-processing building models and interpreting results in WEKA. Well at the risk of redundancy this research paper is only for predicting the PRIMA dataset diabetes using WEKA by barely even going to scratch the surface through doing the coding. You won't be able to apply even the intermediate data mining techniques. When opened my dataset in WEKA Explorer, a well helpful screen showing some things about the data set. The classified tab is where it most commonly does its thing and will contain the algorithms.



Current relation
Relation: diabetes_data_upload
Instances: 520
Attributes: 17
Sum of weights: 520

Attributes

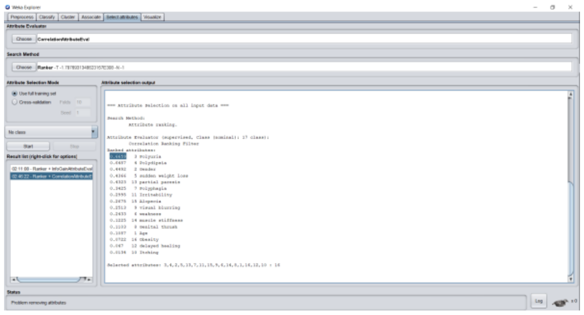
No.	Name
1	Age
2	Gender
3	Polyuria
4	Polydipsia
5	sudden weight loss
6	weakness
7	Polyphagia
8	Genital thrush
9	visual blurring
10	Itching
11	Irritability
12	delayed healing
13	partial paresis
14	muscle stiffness
15	Alopecia
16	Obesity
17	class

Selected attribute
Name: class
Missing: 0 (0%)
Distinct: 2
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	Positive	320	320.0
2	Negative	200	200.0

Figure 2: Dataset representation in WEKA

This shows very clearly that there are 520 instances and 17 attributes, out of there are 320 diabetes positive and 200 negative instances. To check attribute is more important than the other ones we use the selection tab in WEKA which will predict the ranks of the attributes using correlation under ranker algorithms. Following fig_2 show the rank of the attributes that have in this dataset. This shows that the Polyuria is the most effective attribute with a 0.6659 value that influenced the prediction of diabetes and the Itching factor in the lost one with the 0.0134 value. Another correlation of the attributes is given in the figure below.



Attribute Evaluator: CorrelationAttributeEval
Search Method: Ranker -T -1.7976931348623157E308 -N -1
Attribute Selection Mode: (●) Use full training set ( ) Cross-validation Folds 10 Seed 1
Result list (right-click for options)
02:11:08 - Ranker + InfoGainAttributeEval
02:46:22 - Ranker + CorrelationAttributeEval

Attribute selection output

```
=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 17 class):
        Correlation Ranking Filter
Ranked attributes:
 0.6659   3 Polyuria
 0.6487   4 Polydipsia
 0.4492   2 Gender
 0.4366   5 sudden weight loss
 0.4323  13 partial paresis
 0.3425   7 Polyphagia
 0.2995  11 Irritability
 0.2675  15 Alopecia
 0.2513   9 visual blurring
 0.2433   6 weakness
 0.1225  14 muscle stiffness
 0.1103   8 Genital thrush
 0.1087   1 Age
 0.0722  16 Obesity
 0.047   12 delayed healing
 0.0134  10 Itching

Selected attributes: 3,4,2,5,13,7,11,15,9,6,14,8,1,16,12,10 : 16
```

Figure 3: Correlation of attributes

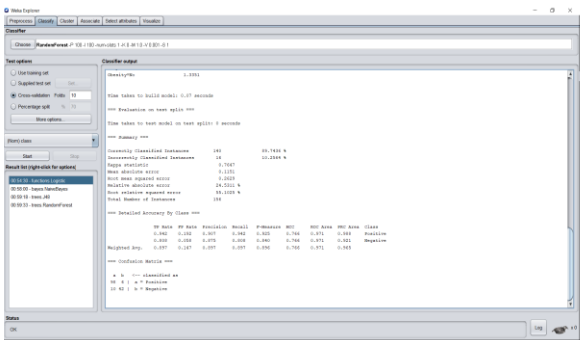
2.1 Classifiers:

For this, we are comparing the 4 different classifiers to evaluate which of them will be more suited for this data set for prediction. We are comparing the Logistic regression, Decision tree, Naive Bayes, and Random Forest tree.

For Logistic Regression we use 10 folds of cross-validation with 70% this shows the Correctly Classified Instances are 89.7436 % and Incorrectly Classified Instances are 10.2564 %. The detailed information on using Logistic Regression is in the figure_4 as posted below. Moreover the probability of the actual predicted class and the highest probability predicted for the other classes can be shown in the margin curve option as well in figure_5.

Probabilistic classifier such as Naïve Bayes is also is a very powerful algorithm for the prediction for the dataset. It is based on probability models that incorporate strong independence assumptions. For this we also use cross validation of 10 folds with 70% split of data. This give Correctly Classified Instances 85.2564 % and Incorrectly

Classified Instances 14.7436 % which is little less than the Logistic regression. Figure_6 shows its detail version of information.



```
RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1

Obesity=No               1.3351


Time taken to build model: 0.07 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Correctly Classified Instances         140               89.7436 %
Incorrectly Classified Instances        16               10.2564 %
Kappa statistic                          0.7647
Mean absolute error                      0.1151
Root mean squared error                  0.2629
Relative absolute error                 24.5311 %
Root relative squared error             55.1025 %
Total Number of Instances              156

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.942    0.192    0.907      0.942    0.925      0.766    0.971     0.988     Positive
                 0.808    0.058    0.875      0.808    0.840      0.766    0.971     0.921     Negative
Weighted Avg.    0.897    0.147    0.897      0.897    0.896      0.766    0.971     0.965

=== Confusion Matrix ===

  a  b   <-- classified as
 98  6 |  a = Positive
 10 42 |  b = Negative
```

Figure 4: Logistic regression modeling in WEKA

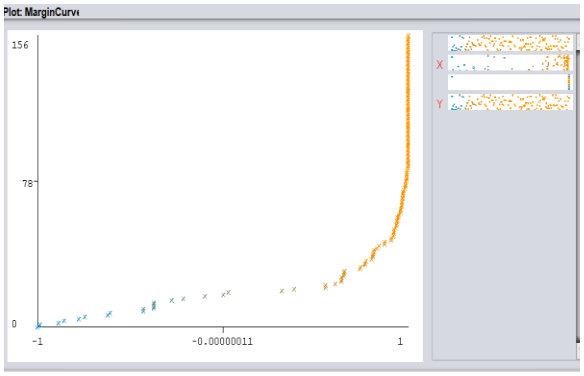
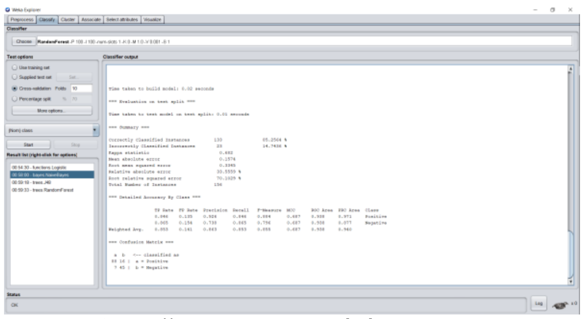


Figure 5: Logistic regression graph in WEKA



```
RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1

Time taken to build model: 0.02 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.01 seconds

=== Summary ===

Correctly Classified Instances         133               85.2564 %
Incorrectly Classified Instances        23               14.7436 %
Kappa statistic                          0.682
Mean absolute error                      0.1574
Root mean squared error                  0.3365
Relative absolute error                 33.5559 %
Root relative squared error             70.1029 %
Total Number of Instances              156

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.846    0.135    0.926      0.846    0.884      0.687    0.938     0.971     Positive
                 0.865    0.154    0.738      0.865    0.796      0.687    0.938     0.877     Negative
Weighted Avg.    0.853    0.161    0.863      0.853    0.855      0.687    0.938     0.940

=== Confusion Matrix ===

  a  b   <-- classified as
 88 16 |  a = Positive
  7 45 |  b = Negative
```

Figure 6: Naïve Bayes modeling in WEKA

After Logistic Regression which give 89% accuracy we try to find it using Decision tree of J48. J48 It is also known as a statistical classifier. With the extension of ID3 J48 is used to generate Decision Tree by C4. 5. Complete decision tree is shown in fig_7. This shows that the Polyuria is most important attribute in detecting the diabetic patient. This also shows less accuracy then Logistic Regression of Correctly Classified Instances in 87.8205 % and Incorrectly Classified Instances in 12.1795 % on cross validation of 10 folds. But this accuracy can be increased by changing the dataset of 8 folds over the cross validation and increased by Correctly Classified Instances in 94.8077 % and Incorrectly Classified Instances in 5.1923 %.

Let's move to the last comparison of this study using WEKA which is RandomForest Classifirer. This gives as the highest accuracy on cross validation stage with 10 folds Correctly Classified Instances in 94.8718 % and Incorrectly Classified Instances in 5.1282 %. This increased to Correctly Classified Instances in 97.3077 % and Incorrectly Classified Instances in2.6923 % on cross validation with 8 folds. This makes it outstanding in all of the above mentioned classifiers that are used in the comparison. See figure_8 for more detailed view of this.

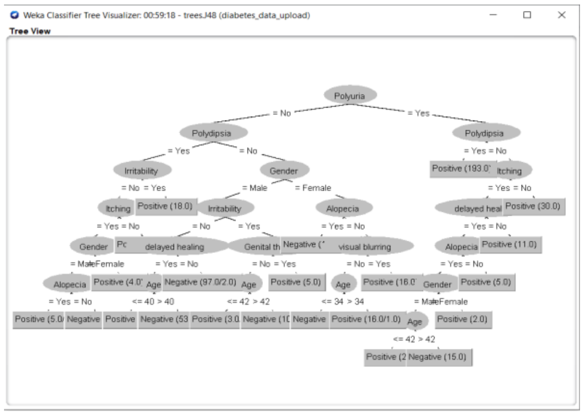
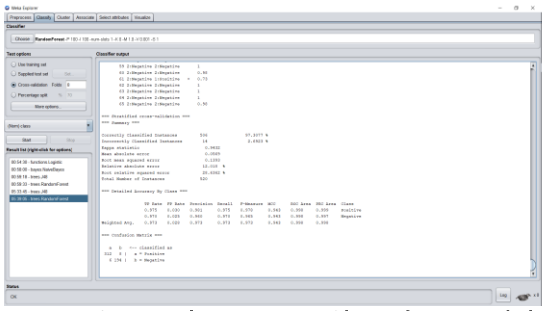


Figure 7: Decision Tree



```
RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1

     59 2:Negative 2:Negative       1
     60 2:Negative 2:Negative       0.98
     61 2:Negative 1:Positive   +   0.73
     62 2:Negative 2:Negative       1
     63 2:Negative 2:Negative       1
     64 2:Negative 2:Negative       1
     65 2:Negative 2:Negative       0.98

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         506               97.3077 %
Incorrectly Classified Instances        14                2.6923 %
Kappa statistic                          0.9432
Mean absolute error                      0.0609
Root mean squared error                  0.1393
Relative absolute error                 12.018  %
Root relative squared error             28.6362 %
Total Number of Instances              520

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.975    0.030    0.981      0.975    0.978      0.943    0.998     0.999     Positive
                 0.970    0.025    0.960      0.970    0.965      0.943    0.998     0.997     Negative
Weighted Avg.    0.973    0.029    0.973      0.973    0.973      0.943    0.998     0.998

=== Confusion Matrix ===

   a   b   <-- classified as
 312   8 |   a = Positive
   6 194 |   b = Negative
```

Figure 8: RandomForest Classifier modeling in WEKA

3. ANALYSES USING RAPIDMINER STUDIO

According to RapidMiner Studio's officials following are points of introduction for it. It increase productivity across the entire data science team, from analysts to experts.

- This increase the productivity with large number of libraries and more then 1,500+ algorithms.
- We can use our defined connections anytime from the processes through the drag and drop very easily.
- With RapidMiner Marketplace we can connect to any extensions with new recourses.
- Without writing the complex SQL queries you just have to drag the filter.
- It automatically understand the trends and patterns of the data distribution to plot different graphs.

These above mentioned points try's to put the impression that this will outperform the in the data science industry. We use many different steps to predict the diabetes and to check how much easier to use RpidMiner Studio and what are the benefits with which they come up to us to make prediction process easier then code we spend more time in analysis rather than coding. This evaluation involves the building of decision tree on dataset, Apply the model according to the suggestion of RapidMiner Studio, Testing the model, Model Selection. Finally in model selection we performed where we use Decision Tree, Random Forest, Naïve Bayes, Rule Induction, and k-NN Classifier for optimization.

3.1 Decision Tree

To visualize Decision tree on this data set is very informative and very detailed. To print attached the screenshot of it here is not feasible because it is so detailed and large that it can't not be fitted here. This is very handy tool to use that you just have to attach the dataset from the repository section {1}. This is all about the drag and drop the operators from the operations sections {2}. Here's we can use Read dataset operator to the process panel, then use filterer example operator from the operations section

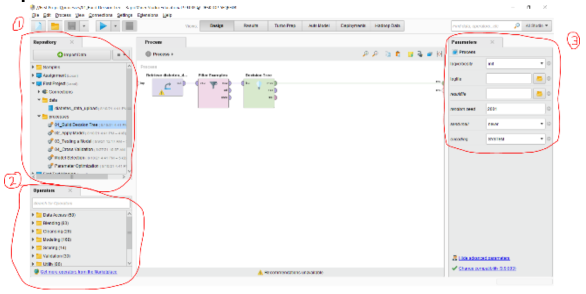


Figure 9: Section in RapidMiner

for filtering the dataset if there is any missing values from the dataset. This will separate the missing values from the dataset which is not in this case. Then we drag and drop the decision tree operator from the operations section and just attached the output of the filter example operator to the input of the decision tree and then attached the output of the decision tree to the result node of the processor section. Then you have to click the run button and here is your decision tree. Here one thing that need focus that are dataset is in categorical form to print decision tree on a categorical data is not simple. We have to convert it to the sample dummy values that convert the categorical values to the series of code. This we will see in more detail in the analysis of the Python code section. Here RapidMiner did all this for us and convert them to numeric series of code that will help in building the decision tree very easy for us.

3.2 Apply Model:

We are using our previous file of process where we build the decision tree. Then why we call this as the applying the model. This is because the following, the each node with the basis on o their diabetic trait factors is a very tedious way of doing it. Especially in this case where the decision tree is that much large that is can't be able to fit in the one process window. Now we want it to do automatically using RapidMiner studio and this action is commonly referred as the "apply the model". You see there we use

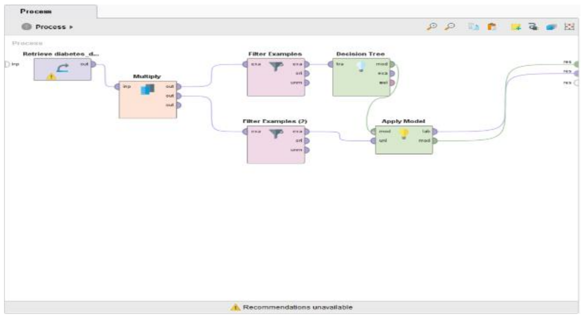
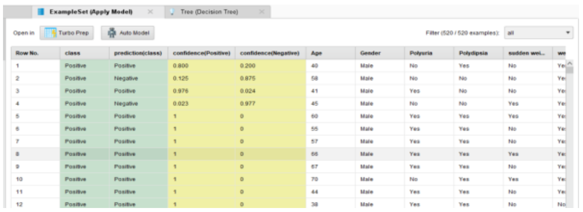


Figure 10: Modeling the data with decision tree

multiply operator for splitting the on dataset to the two different example filter. One for missing values and the other one is for the predicted data values. This will keep the same dataset and keep one filter for making the decision tree and other one is for applying the model testing. The first lab node of the apply model operation filter will give us the predicted values of the dataset and the second node of the apply model will output the model again too us. Results are in the following figure.



ExampleSet (Apply Model) | Tree (Decision Tree)

Open in: Turbo Prep | Auto Model — Filter (520 / 520 examples): all

Row No.	class	prediction(class)	confidence(Positive)	confidence(Negative)	Age	Gender	Polyuria	Polydipsia	sudden wei...	we
1	Positive	Positive	0.800	0.200	40	Male	No	Yes	No	Ye
2	Positive	Negative	0.125	0.875	58	Male	No	No	No	Ye
3	Positive	Positive	0.976	0.024	41	Male	Yes	No	No	Ye
4	Positive	Negative	0.023	0.977	45	Male	No	No	Yes	Ye
5	Positive	Positive	1	0	60	Male	Yes	Yes	Yes	Ye
6	Positive	Positive	1	0	55	Male	Yes	Yes	No	Ye
7	Positive	Positive	1	0	57	Male	Yes	Yes	No	Ye
8	Positive	Positive	1	0	66	Male	Yes	Yes	Yes	Ye
9	Positive	Positive	1	0	67	Male	Yes	Yes	No	Ye
10	Positive	Positive	1	0	70	Male	No	Yes	Yes	Ye
11	Positive	Positive	1	0	44	Male	Yes	Yes	No	Ye
12	Positive	Positive	1	0	38	Male	Yes	Yes	No	No

Figure 11: Visual representation of data in RapinMiner

3.3 Testing Model

In this section we do training and testing the model, so we can see how our predictions will likely be. Here again use one of our previous defined process which is decision tree process. We already apply the model in the previous section that predict for us which patient has the diabetes and which have not. Along with that we can see from this that which of the prediction is 100% predicted accurately predicted correct and which is not. So in this section we test and train the model and see the performance of it.

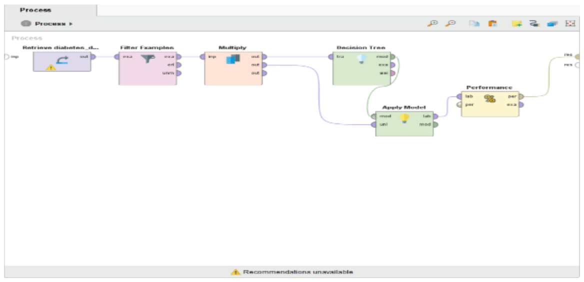
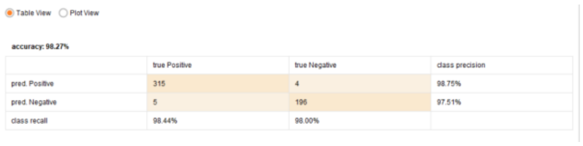


Figure 12: Adding the performance filter with model

Connect the labeled output to the results decision tree. We loaded in the data, we filtered out the unlabeled examples, that don't have 'negative' or 'positive' and thereby we created a training dataset. We fed the training-data into the 'decision tree' and built the model, but we kept a copy of it to drive it down to this branch there to 'apply model' operation, where we took that sample example-set and 'applied the model' to it to create predicted labels so we can finally compare them both. To test our model we just have to drag the performance model filter from the operations section and drop it, front of the 'apply model' filter. Following are the results that we get from it and it shows that we are getting 98% accuracy.

This also shows that AUC (pessimistic) is 95% and AUC (optimistic) is 99%.


[● Table View] ○ Plot View

accuracy: 98.27%

	true Positive	true Negative	class precision
pred. Positive	315	4	98.75%
pred. Negative	5	196	97.51%
class recall	98.44%	98.00%	

Figure 13: Results

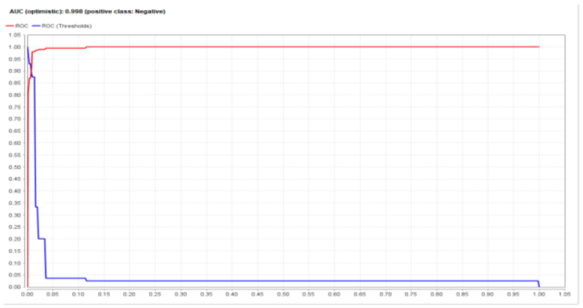


Figure 14: AUC (optimistic)

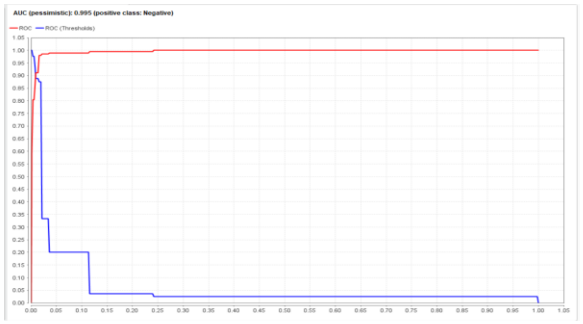


Figure 15: AUC (pessimistic)

3.4 Model Selection

It has been proven by the two gentleman Wolpert and Macready in their "No free lunch" theorem that there is no specific algorithm for any specific dataset that is designated to them. There are many algorithms that can be used for one problem and their accuracy differ from each other. RapidMiner did this for us in very easy way of comparison.

FINDING THE RIGHT MODEL:

- CLASSIFICATIONS:"IS THIS A OR B?
  WILL THIS BE A OR B?"
- REGRESSION:"HOW MUCH OR HOW MANY?
  HOW MANY WILL HAPPEN?"
- CLUSTERING:"HOW IS THIS ORGANIZES?
  WHAT BELONGS TO EACH OTHER?"
- ASSOCIATIONS & CORRELATIONS: "WHAT HAPPENS TOGETHER?
  WHAT CHANGES TOGETHER?"
- ANOMALY DETECTION:"IS THIS WEIRD?"

Here we are working with the most common problem with machine learning problem which is machine learning. Some time we don't know which algorithm we should use for the dataset because sometimes algorithm are not fit for the categorical datasets such as in our case this is a categorical data. To see detailed view of data is displayed in the following figure. Here RapidMiner again help us with the website name MOD RapidMiner. Here we can select the dataset types and it will suggest that which algorithm can be performed with this dataset. This suggest us that Decision tree, k-NN, Naïve Bayes, Rule Induction and Random Forest is best. We will see in this section that which one of these will perform the outmost. Here we will use cross validation and ROCs compare filter will help us in comparing the algorithm.

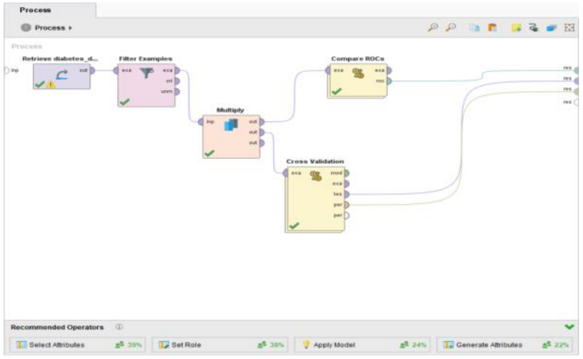


Figure 16: ROC filter in RapidMiner

In cross validation filter operator we are using decision tree and then applying model to it and then check its performance with their performance operator just like in the above section of testing model. Following figure will give you a visual experience of it and an inside view of it.

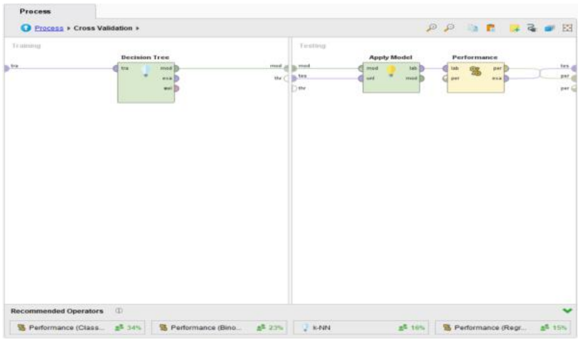


Figure 17: Internal view of Cross Validation filter

Now here in ROCs comparable filter we are using the 5 different algorithm to compare that which one perform the best. Here in ROCs we are using k-NN, Rule Induction, Naïve Bayes, Decision Tree and Random Forest for comparison. Following figure will display the process that how it works. We just have to drag and drop the algorithms from the operations section to inside of the ROCs filter.

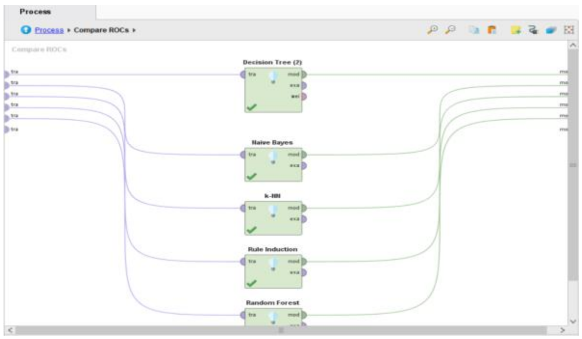
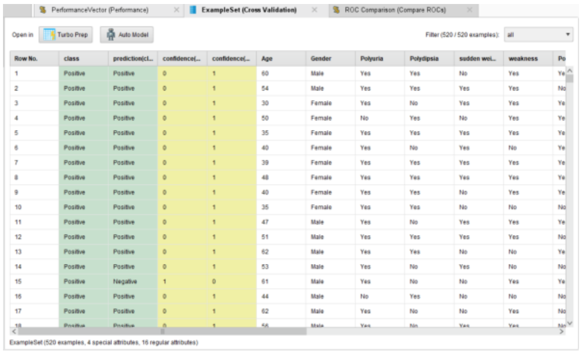


Figure 18: Internal view of ROCs filter with classifiers

After running the process we get the PerformanceVector, Cross Validation set results and the ROCs graph.

PerformanceVector

```
PerformanceVector:
accuracy: 96.35% +/- 2.47%
(micro average: 96.35%)
ConfusionMatrix:
True:  Positive       Negative
Positive:      308     7
Negative:      12      193
precision: 94.24% +/- 3.48%
(micro average: 94.15%)
(positive class: Negative)
ConfusionMatrix:
True:  Positive       Negative
Positive:      308     7
Negative:      12      193
recall: 96.50% +/- 4.74% (micro
average: 96.50%) (positive
class: Negative)
ConfusionMatrix:
True:  Positive       Negative
Positive:      308     7
Negative:      12      193
AUC (optimistic): 0.990 +/-
0.009 (micro average: 0.990)
(positive class: Negative)
AUC: 0.955 +/- 0.038 (micro
average: 0.955) (positive class:
Negative)
AUC (pessimistic): 0.943 +/-
0.052 (micro average: 0.943)
(positive class: Negative)
```


PerformanceVector (Performance)
ExampleSet (Cross Validation)
ROC Comparison (Compare ROCs)
Open in
Turbo Prep
Auto Model
Filter (520 / 520 examples):
all

Row No.	class	prediction(cl...	confidence(...	confidence(...	Age	Gender	Polyuria	Polydipsia	sudden wei...	weakness	Po
1	Positive	Positive	0	1	60	Male	Yes	Yes	No	Yes	Ye
2	Positive	Positive	0	1	54	Male	Yes	Yes	Yes	Yes	No
3	Positive	Positive	0	1	30	Female	Yes	No	Yes	Yes	Ye
4	Positive	Positive	0	1	50	Female	No	Yes	No	Yes	Ye
5	Positive	Positive	0	1	35	Female	Yes	Yes	Yes	Yes	Ye
6	Positive	Positive	0	1	40	Female	Yes	No	Yes	No	Ye
7	Positive	Positive	0	1	39	Female	Yes	Yes	Yes	Yes	Ye
8	Positive	Positive	0	1	48	Female	Yes	Yes	Yes	Yes	Ye
9	Positive	Positive	0	1	40	Female	Yes	Yes	No	Yes	Ye
10	Positive	Positive	0	1	35	Female	Yes	Yes	No	No	No
11	Positive	Positive	0	1	47	Male	Yes	No	Yes	Yes	Ye
12	Positive	Positive	0	1	51	Male	Yes	Yes	Yes	Yes	No
13	Positive	Positive	0	1	62	Male	Yes	Yes	No	No	Ye
14	Positive	Positive	0	1	53	Male	Yes	No	Yes	No	No
15	Positive	Negative	1	0	61	Male	Yes	No	No	Yes	Ye
16	Positive	Positive	0	1	44	Male	No	Yes	No	No	No
17	Positive	Positive	0	1	62	Male	Yes	No	No	Yes	No
18	Positive	Positive	0	1	56	Male	Yes	No	Yes	Yes	No

ExampleSet (520 examples, 4 special attributes, 16 regular attributes)

Figure 19: Cross Validation results

ROC = Receiver Operator Characteristics

ROC stands for 'Receiver operator characteristics' and was established during World War II to describe the performance of the people supervising and interpreting the signals on the Radar displays. Now let's look at a blank ROC graph with a hit rate on the y-axis and a false alarm rate on the x-axis.

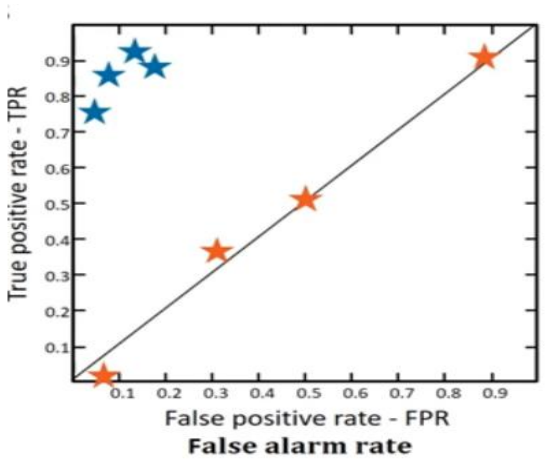


Figure 20: Sample ROC graph

Operating stuff which had an as good "hit rate" as "false alarm rate" is as useful as a coin when decided if measures were to be taken when a signal appeared on the radar screen. In the upper right or lower left, we would see extreme examples of an operator who never triggered an alarm and therefore had also a small false alarm rate or one how would always trigger an alarm. Therefore would never miss an actual true case but produced a lot of false alarms, neither of those is useful. The best 'receiver operators' had a very high 'hit rate' and a very small 'false alarm rate'.

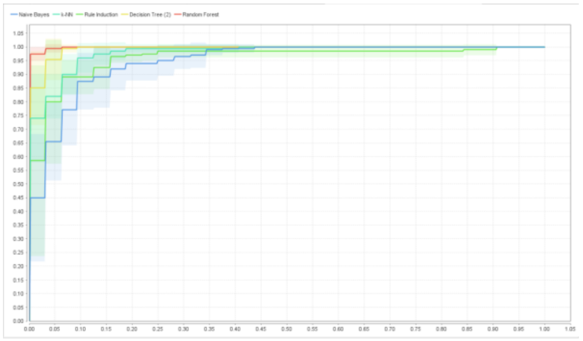


Figure 21: ROC graph of different algorithms

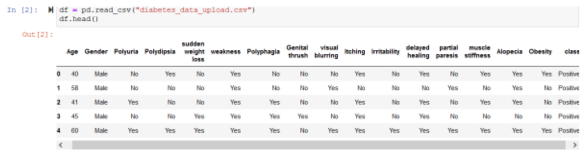
The same is true for our classification algorithms. One thing which is still to be explained is: how do we get from a ROC point characteristic? Which we can calculate from our confusion matrix to a curve. For that, we have to extract the confidence for each prediction made by our classification algorithms. The confidence is similar to the probability that the predicted class will actually be correct and therefore tells us how sure the model is with its prediction. In RapidMiner, we can retrieve those confidence values from the performance operators. Make sure to connect these to a results port so we get an output. Now as we run the process again you can see the confidence displayed. If we rank them you can see that the prediction changes for those examples. Here you can clearly shows that the Random Forest is outperform the most and stand alone among those other algorithm.

4. ANALYSES USING PYTHON CODING

As we know Python is a very powerful tool when it comes to data science. Both above mention tools are GUI tools but we need knowledge for modeling and selection of process of data analysis. In this section we see the prediction of diabetes using Python scripting. Code of this analysis is uploaded to a GitHub Repository.

4.1 Data

Below is the dataset that we used for this scripting analysis.



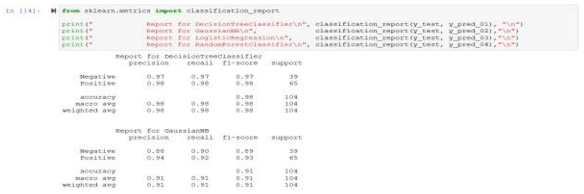
```
In [2]:  df = pd.read_csv("diabetes_data_upload.csv")
         df.head()
Out[2]:
```

	Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	visual blurring	Itching	Irritability	delayed healing	partial paresis	muscle stiffness	Alopecia	Obesity	class
0	40	Male	No	Yes	No	Yes	No	No	No	Yes	No	Yes	No	Yes	Yes	Yes	Positive
1	58	Male	No	No	No	Yes	No	No	Yes	No	No	No	Yes	No	Yes	No	Positive
2	41	Male	Yes	No	No	Yes	Yes	No	No	Yes	No	Yes	No	Yes	Yes	No	Positive
3	45	Male	No	No	Yes	Yes	Yes	Yes	No	Yes	No	Yes	No	No	No	No	Positive
4	60	Male	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Positive

Figure 22: Data representation with python

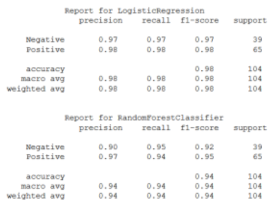
4.2 Discussion

Using the python machine learning techniques, we trained our Algorithm to make predictions for every individuals based on the onset given values. To achieve the goal, we trained our data using 80% of the data for training the model and 20% data for testing purposes. In order to increase the accuracy of the Algorithm we equally distributed the dataset based on Positive and Negative result. Hence as a result we are getting the accuracy above 90% in predicting the outcome on average for the 4 different classifiers. We use Logistic Regression, Decision Tree, Naïve Bayes and Random Forest for the evaluation that which person the best from them. Following are the detailed view of the Classification Report. This shows every Classifier predict more than 90% TruePositive instances. From this we interpret that the DecisionTreeClassifier() predicts 98% TruePositive, GaussianNB() predicts 94% TruePositive, LogisticRegression() predicts 98% TruePositive and RandomForestClassifier() predicts 98% TruePositive.



```
In [14]:  from sklearn.metrics import classification_report

          print("          Report for DecisionTreeClassifier\n", classification_report(y_test, y_pred_01), "\n")
          print("          Report for GaussianNB\n",             classification_report(y_test, y_pred_02),"\n")
          print("          Report for LogisticRegression\n",     classification_report(y_test, y_pred_03),"\n")
          print("          Report for RandomForestClassifier\n", classification_report(y_test, y_pred_04),"\n")

          Report for DecisionTreeClassifier
               precision    recall  f1-score   support

    Negative       0.97      0.97      0.97        39
    Positive       0.98      0.98      0.98        65

    accuracy                           0.98       104
   macro avg       0.98      0.98      0.98       104
weighted avg       0.98      0.98      0.98       104


          Report for GaussianNB
               precision    recall  f1-score   support

    Negative       0.88      0.90      0.89        39
    Positive       0.94      0.92      0.93        65

    accuracy                           0.91       104
   macro avg       0.91      0.91      0.91       104
weighted avg       0.91      0.91      0.91       104
```

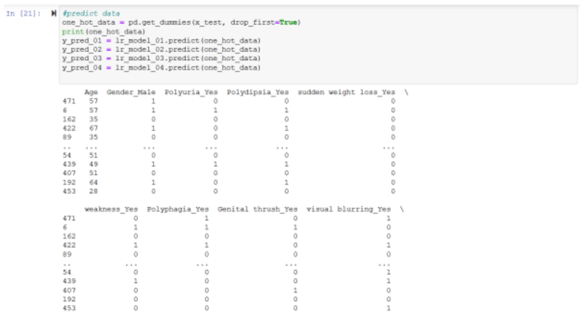
Figure 23: Naïve Bayes and Decision tree results



```
          Report for LogisticRegression
               precision    recall  f1-score   support

    Negative       0.97      0.97      0.97        39
    Positive       0.98      0.98      0.98        65

    accuracy                           0.98       104
   macro avg       0.98      0.98      0.98       104
weighted avg       0.98      0.98      0.98       104


          Report for RandomForestClassifier
               precision    recall  f1-score   support

    Negative       0.90      0.95      0.92        39
    Positive       0.97      0.94      0.95        65

    accuracy                           0.94       104
   macro avg       0.94      0.94      0.94       104
weighted avg       0.94      0.94      0.94       104
```

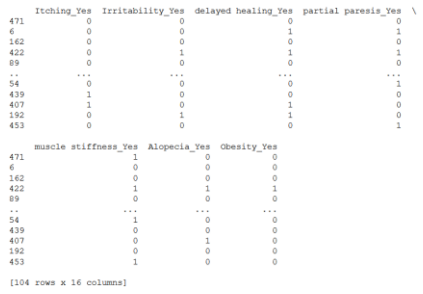
Figure 24: Logistic Regression and Random Forest results

As our data is a categorical variable so we use pandas.get_dummies. It Convert categorical variable into dummy variables. It actually Convert Series to dummy codes. This is because Logistic Regression and others needs binary output and our data is categorical variable so it convert strings to these codes.



```
In [21]:  #predict data
          one_hot_data = pd.get_dummies(x_test, drop_first=True)
          print(one_hot_data)
          y_pred_01 = lr_model_01.predict(one_hot_data)
          y_pred_02 = lr_model_02.predict(one_hot_data)
          y_pred_03 = lr_model_03.predict(one_hot_data)
          y_pred_04 = lr_model_04.predict(one_hot_data)

     Age  Gender_Male  Polyuria_Yes  Polydipsia_Yes  sudden weight loss_Yes  \
471   57            1             0               0                       0
6     57            1             1               1                       0
162   35            0             0               0                       0
422   67            1             0               1                       0
89    35            0             0               0                       0
..   ...          ...           ...             ...                     ...
54    51            0             0               0                       0
439   49            1             1               1                       0
407   51            0             0               0                       0
192   64            1             0               1                       0
453   28            0             0               0                       0

     weakness_Yes  Polyphagia_Yes  Genital thrush_Yes  visual blurring_Yes  \
471             0               1                   0                    1
6               1               1                   1                    0
162             0               0                   0                    0
422             1               1                   0                    1
89              0               0                   0                    0
..            ...             ...                 ...                  ...
54              0               0                   0                    1
439             1               0                   0                    1
407             0               0                   1                    0
192             0               0                   0                    0
453             0               0                   0                    1
```

Figure 25: Dataset in dummy series of values



```
     Itching_Yes  Irritability_Yes  delayed healing_Yes  partial paresis_Yes  \
471            0                 0                    0                    0
6              0                 0                    1                    1
162            0                 0                    0                    0
422            0                 1                    1                    1
89             0                 0                    0                    0
..           ...               ...                  ...                  ...
54             0                 0                    0                    1
439            1                 0                    0                    0
407            1                 0                    1                    0
192            0                 1                    1                    0
453            0                 0                    0                    1

     muscle stiffness_Yes  Alopecia_Yes  Obesity_Yes
471                     1             0            0
6                       0             0            0
162                     0             0            0
422                     1             1            1
89                      0             0            0
..                    ...           ...          ...
54                      1             0            0
439                     0             0            0
407                     0             1            0
192                     0             0            0
453                     1             0            0

[104 rows x 16 columns]
```

Figure 26: Dummy values

5. **Results:**

WEKA uses individual analysis of all the algorithms. We use Random Forest, Naïve Bayes, Logistic Regression and J48 Decision Tree. Random Forest gives 97. 31, Naïve Bayes gives 85.26% accuracy, Logistic Regression gives 89.74% accuracy, and J48 Decision Tree gives 94.81 on analysis. This also shows that the Polyuria is the most influenced factor on predicting the diabetes and Itching is le least one on comparing the correlation and information gain. With Python scripting code we also get the Random Forest with the most accuracy. This shows us that Python scripting take much of you time in writing the code. For this problem RapidMiner come into the picture and give the most detailed and fast analysis of the prediction. We compare k-NN, Naïve Bayes, Random Forest, Decision Tree and Rule Induction algorithms, Random Forest and Random Forest perform the best. We also analysis is using ROCs algorithm as well.

6. Conclusions:

Decision making in prediction of a disease is very critical method. Physicians need to know the accuracy factors at which rate the system can predict the disease. This can be formulated with data mining algorithms and machine learning approach. The current research shows that the Random forest algorithm is most suitable for predicting the disease. Along with that RapidMiner overcome the limitations of other data mining analysis such as WEKA and Python scripting. RapidMiner can be used to the mass level.

Author details

Jam Ayub is final year (7th Semester) student at Forman Christian College (A Chartered University) under the roll number 21-11482

References

1. Bioch, J. C. (1996). Classification using Bayesian neural nets. 1488-1493.
2. Carpenter, G. A. (1998). ARTMAP-IC and medical diagnosis: instance counting and inconsistent cases‖, Neural Networks. 323-336.
3. J. Pradeep Kandhasamy, S. (2015). Performance Analysis of Classifier Models to Predict Diabetes Mellitus. 45–51.
4. José Antonio Sanz, M. G. (2013). Medical diagnosis of cardiovascular diseases using an interval-valued fuzzy rule-based classification system. 1758–1765.
5. José Antonio Sanz, M. G. (2013). Medical diagnosis of cardiovascular diseases using an interval-valued fuzzy rule-based classification system. 103-111.
6. N H Cho, J. E. (2018). IDF Diabetes Atlas: Global estimates of diabetes prevalence for 2017 and projections for 2045.
7. Narges Razavian, S. B.-M. (2015). Population-Level Prediction of Type 2 Diabetes from Claims Data and Analysis of Risk Factors. 277–287.
8. Zolfaghari, R. (2012). Diagnosis of Diabetes in Female Population of Pima Indian Heritage with Ensemble of BP Neural Network and SVM. 1-7.