CS3120

Machine Learning and Neural Computing

Random Forest Assignment

Name	**:** D. R. R. Wijewardene
Index No.	**:** S14245
Faculty	**:** Science
Degree	**:** Bioinformatics

Contents

Theoretical Background

Artificial Intelligence involves machines which can simulate human intelligence processes. Machine Learning is one of its applications which can provide computer systems an ability to automatically learn and improve from experience instead of being programmed explicitly. Computer programs can be developed to access data and learn for themselves, so that they can implement that knowledge to decide the result of unforeseen data.

There are three types of machine learning. They are,

- **Supervised Learning** –

  In this type, the algorithm is trained on labeled data. They contain input values and target values. The algorithm can figure out a pattern between input values and the relevant output. Then it can use that pattern to predict outputs of unforeseen input values.
- Unsupervised Learning –

  In this type, the algorithm is trained on unlabeled data. In other words, the training data does not include the output values. The algorithm has to figure out the desired output after training for many iterations.
- Reinforcement Learning –

  In this type, the system is taught to make specific decisions using trial and error. The algorithm will be rewarded for every right decision so that this feedback can be used to gain knowledge about stronger strategies.

Supervised Learning can be separated to two types of problems named Classification and Regression.

- **Classification** type – The algorithm assigns data into specific categories.

  Eg: Support Vector Machines, Naïve Bayes, Decision Tree, **Random Forest**
- Regression type – The algorithm understands the relationship between dependent and independent variables.

  Eg: Linear regression, Logistic regression

Decision tree is mainly a Classification-type example of Supervised learning. It is a tree-shaped, flowchart-like diagram which is used to determine a course of action. Basically, the training data consists of inputs and their corresponding outputs, and in the tree the data is continuously split according to different parameters.

Each branch-end of the tree, or each leaf, represents a possible decision or outcome. The data splits at decision nodes. Various conditions are considered at each split point. The order of the features which cause splits is obtained by considering the cost of the split.

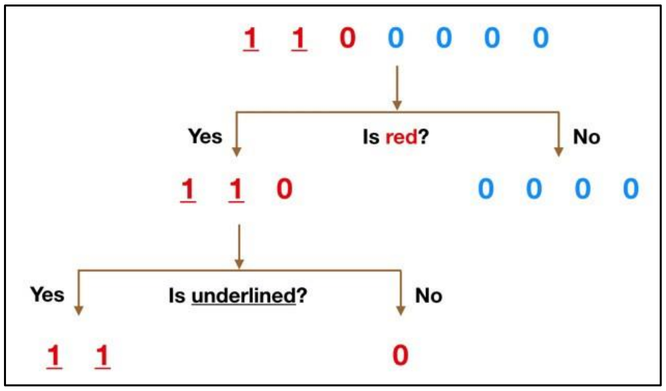


Figure 1: Simple Decision Tree Example
Resource: https://towardsdatascience.com/understanding-random-forest-58381e0602d2

Random Forest

Random Forest is also a Classification-type Supervised Learning method that is constructed by multiple decision trees which operate as an ensemble. Each individual decision tree produces a prediction. The final decision of the random forest is the decision of the majority of the decision trees.

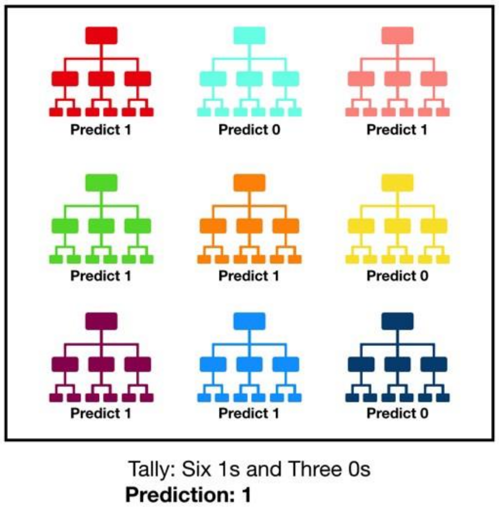


Figure 2: Visualization of a Random Forest Model Making a Prediction
Resource: https://towardsdatascience.com/understanding-random-forest-58381e0602d2

It basically uses feature randomness when building each individual tree. This creates an uncorrelated forest of trees. The prediction made considering a large number of relatively uncorrelated trees is much more accurate than that of an individual constituent tree. This is because the trees protect each other from their individual errors. Some trees maybe wrong and most others maybe right. Therefore, as a group they are able to navigate towards the correct direction.

Random Forest Algorithm

What basically happens is;

If,

Number of observations	= N
Number of features/variables	= M

- The N observations will be sampled at random with replacement.
- A number m, where m<M, will be selected at random at each node from the total number of variables.
- The best split on these m variables is used to split the node and this value remains constant as the forest grows.
- Each decision tree in the forest is grown to its largest extent.
- The forest will output a prediction based on the aggregated predictions of the trees in the forest. (Either majority vote or average)

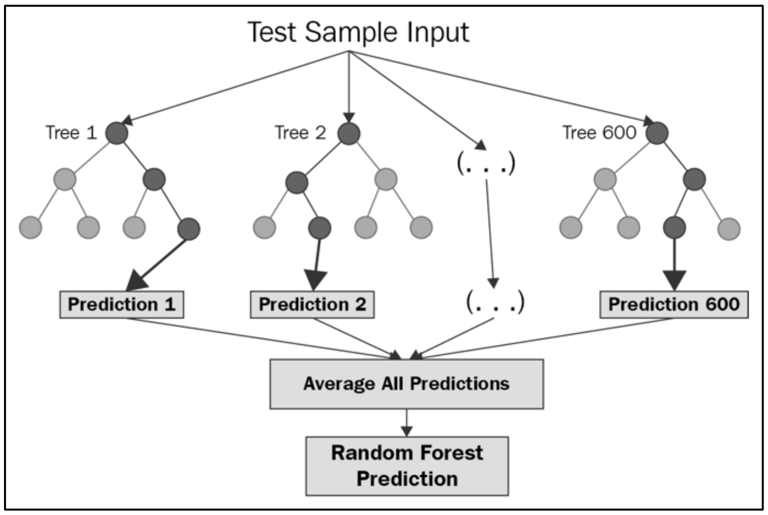


Figure 3: Random Forest Technique

Resource: https://corporatefinanceinstitute.com/resources/knowledge/other/random-forest/

Applications

Random forest has been applied in many different fields.

- The bank industry uses it for credit card fraud detection, customer segmentation, predicting loan defaults etc.
- Healthcare and medicine industry uses it for cardiovascular disease prediction, diabetes prediction, breast cancer prediction etc.
- The stock market field uses it for stock market prediction, sentiment analysis, bitcoin price detection etc.
- E-commerce field uses it for product recommendation, price optimization, search ranking etc.

Advantages

Random forest algorithm is advantageous in many aspects.

- It is versatile – Even though it is predominantly suitable for classification tasks, it can also be used for regression tasks.
- It gives higher accuracy through cross validation.
- It can handle missing values, and maintain the accuracy of a large proportion of data.
- It doesn't allow overfitting of trees.

    One of the biggest problems in machine learning is overfitting, but most of the time this won't happen in the random forest classifier. If there are enough trees in the forest, the classifier won't overfit the model.

- It can work with large datasets with higher dimensionality.

Limitations

- A large number of trees can make this algorithm too slow and ineffective for real-time predictions.

    Generally, there algorithms are fast to train, but quite slow to create predictions once they are trained. More trees are required for a more accurate prediction, which results in a slower model. In most real-world applications, the random forest algorithm is fast enough but there can be situations where run-time performance is important and other approaches would be preferred.

- This is a predictive modeling tool and not a descriptive tool, which means that if you're looking for a description of the relationships of your data, other approaches would be better.

An application of Random Forest Machine Learning technique

Source code

https://colab.research.google.com/drive/169yNRspP8YpCt9g9c_F9KImia7Zr9Gvd?usp=sharing

Dataset

I obtained a dataset regarding Acute Inflammation of the Urinary Bladder from the UCI machine learning repository to apply the Random Forest machine learning technique. It has 6 attributes, 1 final decision and 120 instances.

According to the information provided with the dataset, this data was created by a medical expert, and the main idea is to prepare the algorithm which will perform the presumptive diagnosis of a disease of the urinary system, "Inflammation of urinary bladder".

Symptoms

1.	Temperature of patient	{ 35C-42C }
2.	Occurrence of nausea	{ yes, no }
3.	Lumbar pain	{ yes, no }
4.	Urine pushing (continuous need for urination)	{ yes, no }
5.	Micturition pains	{ yes, no }
6.	Burning of urethra, itch, swelling of urethra outlet	{ yes, no }

Diagnosis:

Inflammation of urinary bladder	{ yes, no }

Acute inflammation of urinary bladder is characterized by sudden occurrence of pains in the abdomen region and the urination in form of constant urine pushing, micturition pains and sometimes lack of urine keeping. Temperature of the body is rising, however most often not above 38C. The excreted urine is turbid and sometimes bloody. At proper treatment, symptoms decay usually within several days. However, there is inclination to returns. At persons with acute inflammation of urinary bladder, we should expect that the illness will turn into protracted form.

Problem

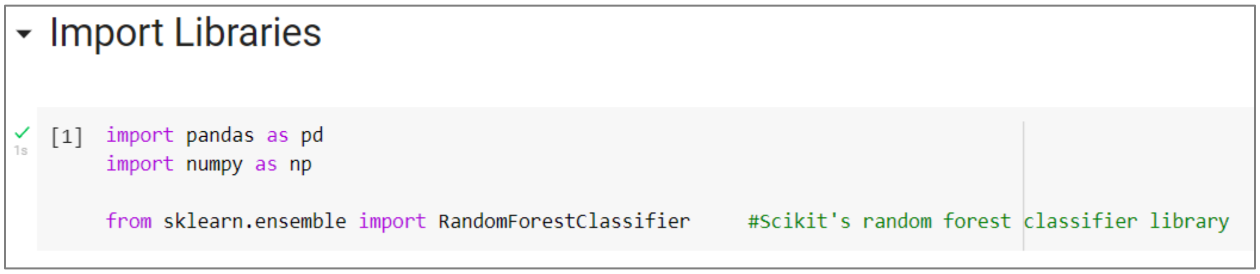
Developing an algorithm which could predict the diagnosis for "Inflammation of urinary bladder" (yes or no) when unforeseen symptom data of a patient is provided.

My approach

I applied the Random Forest machine learning technique to this dataset in order to develop an algorithm to solve the above problem.

Steps

1. **First, the necessary libraries were imported.**



Import Libraries

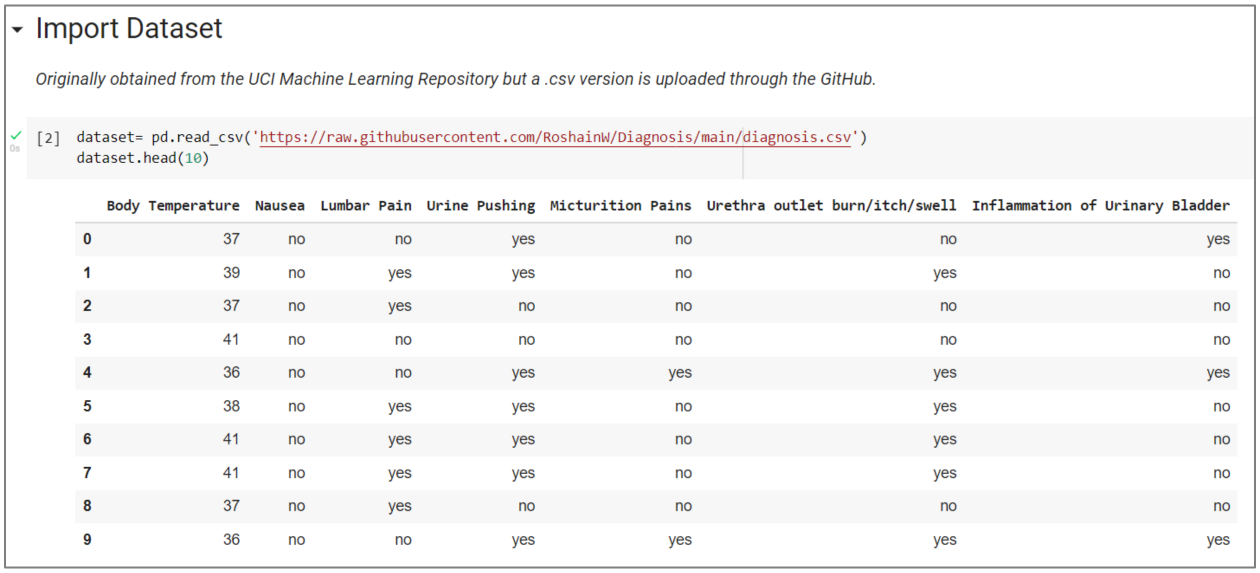
```
[1] import pandas as pd
    import numpy as np

    from sklearn.ensemble import RandomForestClassifier     #Scikit's random forest classifier library
```

Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data.

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.

The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction. The **sklearn.ensemble module** includes two averaging algorithms based on randomized decision trees: the **RandomForest algorithm** and the Extra-Trees method. Therefore, the RandomForestClassifier is imported from this module for our implementation.

2. **Then the dataset was imported.**



Import Dataset

Originally obtained from the UCI Machine Learning Repository but a .csv version is uploaded through the GitHub.

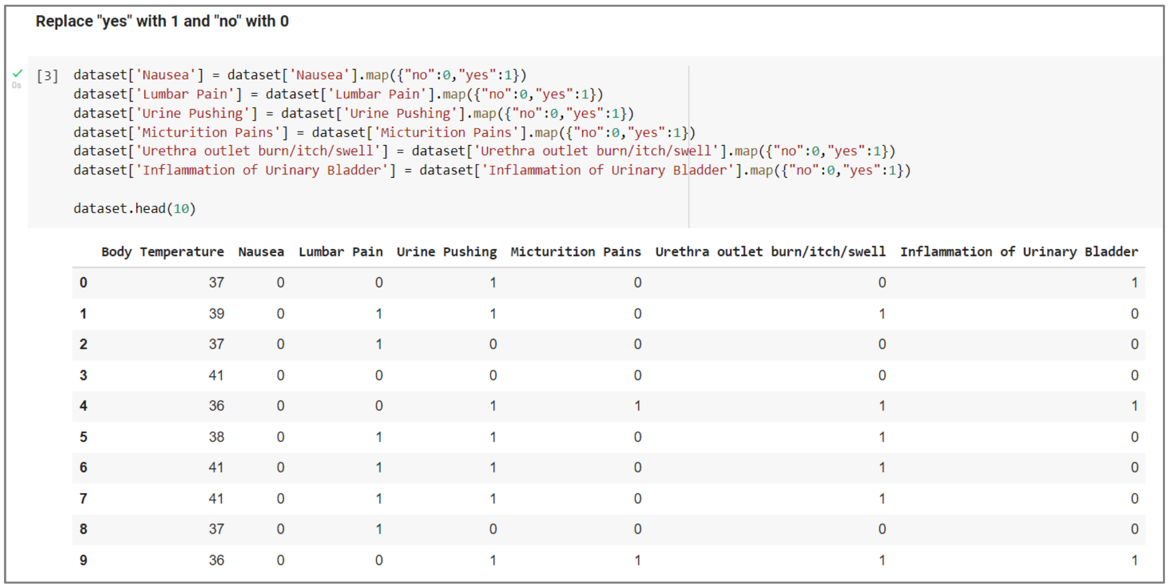
```
[2] dataset= pd.read_csv('https://raw.githubusercontent.com/RoshainW/Diagnosis/main/diagnosis.csv')
    dataset.head(10)
```

	Body Temperature	Nausea	Lumbar Pain	Urine Pushing	Micturition Pains	Urethra outlet burn/itch/swell	Inflammation of Urinary Bladder
0	37	no	no	yes	no	no	yes
1	39	no	yes	yes	no	yes	no
2	37	no	yes	no	no	no	no
3	41	no	no	no	no	no	no
4	36	no	no	yes	yes	yes	yes
5	38	no	yes	yes	no	yes	no
6	41	no	yes	yes	no	yes	no
7	41	no	yes	yes	no	yes	no
8	37	no	yes	no	no	no	no
9	36	no	no	yes	yes	yes	yes

3. The dataset was rearranged to fit the algorithm.

a) First the attributes which contained “yes” and “no” were changed so that they contained “0” instead of “no” and “1” instead of “yes”.

Even though random forest algorithms usually accept categorical inputs, since the sklearn library accepts only numerical values, these categories were converted accordingly.



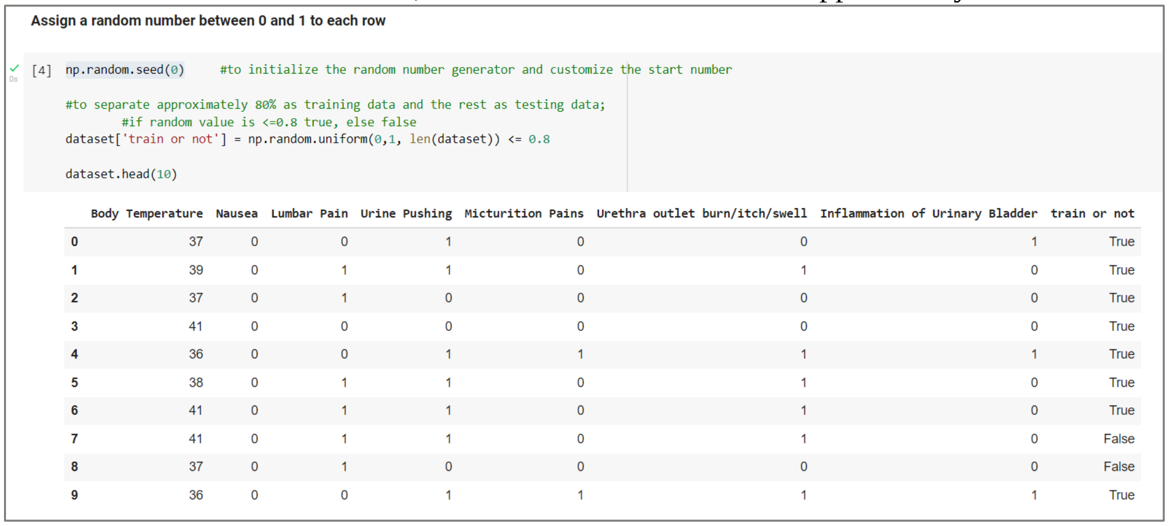
Replace "yes" with 1 and "no" with 0

```
dataset['Nausea'] = dataset['Nausea'].map({"no":0,"yes":1})
dataset['Lumbar Pain'] = dataset['Lumbar Pain'].map({"no":0,"yes":1})
dataset['Urine Pushing'] = dataset['Urine Pushing'].map({"no":0,"yes":1})
dataset['Micturition Pains'] = dataset['Micturition Pains'].map({"no":0,"yes":1})
dataset['Urethra outlet burn/itch/swell'] = dataset['Urethra outlet burn/itch/swell'].map({"no":0,"yes":1})
dataset['Inflammation of Urinary Bladder'] = dataset['Inflammation of Urinary Bladder'].map({"no":0,"yes":1})

dataset.head(10)
```

	Body Temperature	Nausea	Lumbar Pain	Urine Pushing	Micturition Pains	Urethra outlet burn/itch/swell	Inflammation of Urinary Bladder
0	37	0	0	1	0	0	1
1	39	0	1	1	0	1	0
2	37	0	1	0	0	0	0
3	41	0	0	0	0	0	0
4	36	0	0	1	1	1	1
5	38	0	1	1	0	1	0
6	41	0	1	1	0	1	0
7	41	0	1	1	0	1	0
8	37	0	1	0	0	0	0
9	36	0	0	1	1	1	1

b) The next step was to select a part of the dataset as the training set and the rest as the testing set. I decided to use 80% of my dataset to train the random forest.

This separation was done by first assigning a random number between 0 and 1 to each instance, and then the rows with numbers equal or less than 0.8 were assigned to be “True” and were selected as the training set, while the rest were assigned to be “False” and selected as the testing set. Here I included `np.random.seed(0)` to make the random numbers predictable. In other words, this was used so that with the seed reset, the same set of numbers will appear every time.

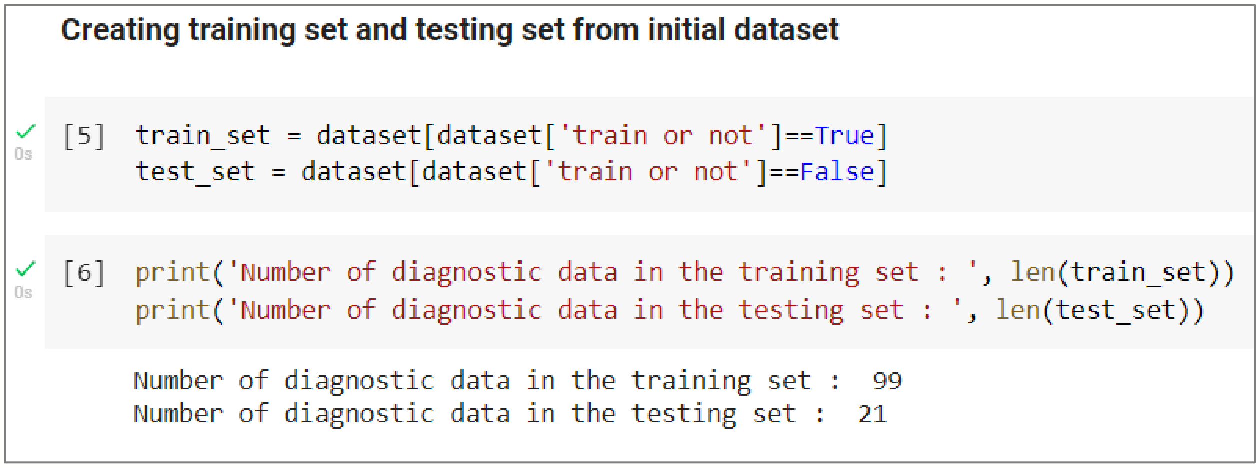


Assign a random number between 0 and 1 to each row

```
np.random.seed(0)     #to initialize the random number generator and customize the start number

#to separate approximately 80% as training data and the rest as testing data;
        #if random value is <=0.8 true, else false
dataset['train or not'] = np.random.uniform(0,1, len(dataset)) <= 0.8

dataset.head(10)
```

	Body Temperature	Nausea	Lumbar Pain	Urine Pushing	Micturition Pains	Urethra outlet burn/itch/swell	Inflammation of Urinary Bladder	train or not
0	37	0	0	1	0	0	1	True
1	39	0	1	1	0	1	0	True
2	37	0	1	0	0	0	0	True
3	41	0	0	0	0	0	0	True
4	36	0	0	1	1	1	1	True
5	38	0	1	1	0	1	0	True
6	41	0	1	1	0	1	0	True
7	41	0	1	1	0	1	0	False
8	37	0	1	0	0	0	0	False
9	36	0	0	1	1	1	1	True

Separating instances with “True” as training data and instances with “False” as testing data:



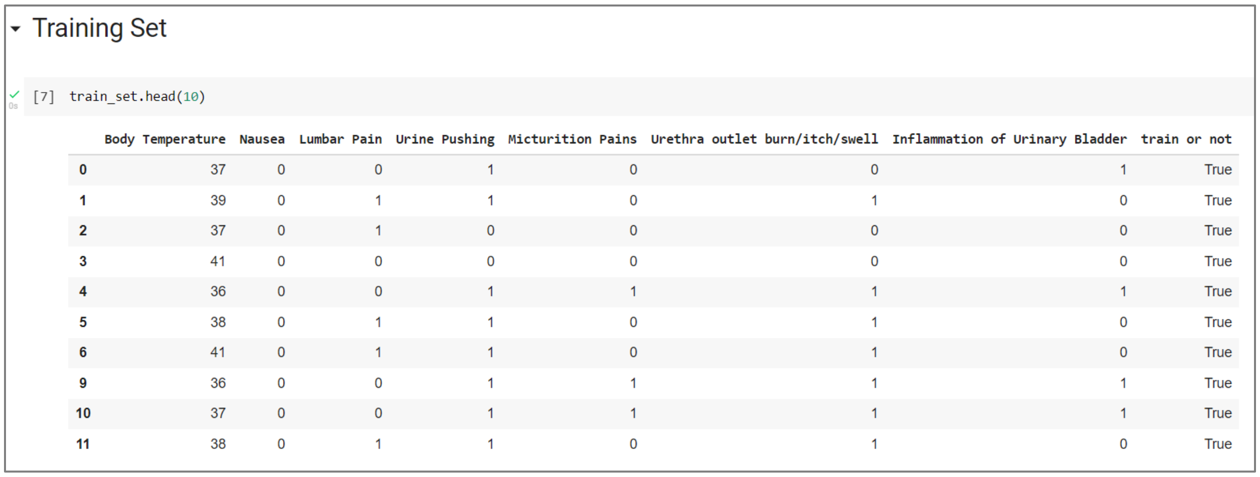
Creating training set and testing set from initial dataset

```
[5] train_set = dataset[dataset['train or not']==True]
    test_set = dataset[dataset['train or not']==False]
```

```
[6] print('Number of diagnostic data in the training set : ', len(train_set))
    print('Number of diagnostic data in the testing set : ', len(test_set))
```

```
Number of diagnostic data in the training set :  99
Number of diagnostic data in the testing set :  21
```

Training dataset:

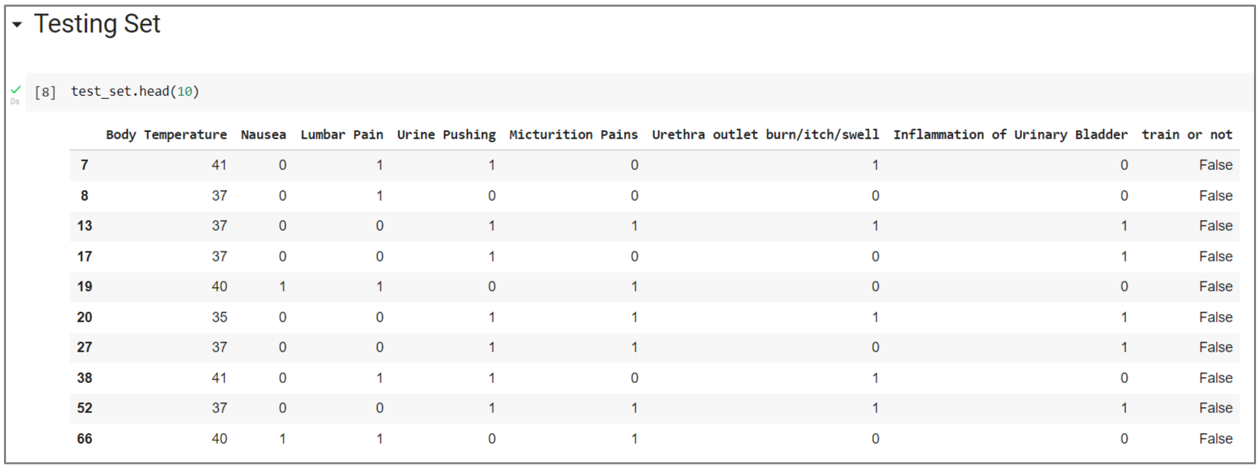


Training Set

```
[7] train_set.head(10)
```

	Body Temperature	Nausea	Lumbar Pain	Urine Pushing	Micturition Pains	Urethra outlet burn/itch/swell	Inflammation of Urinary Bladder	train or not
0	37	0	0	1	0	0	1	True
1	39	0	1	1	0	1	0	True
2	37	0	1	0	0	0	0	True
3	41	0	0	0	0	0	0	True
4	36	0	0	1	1	1	1	True
5	38	0	1	1	0	1	0	True
6	41	0	1	1	0	1	0	True
9	36	0	0	1	1	1	1	True
10	37	0	0	1	1	1	1	True
11	38	0	1	1	0	1	0	True

Testing dataset:

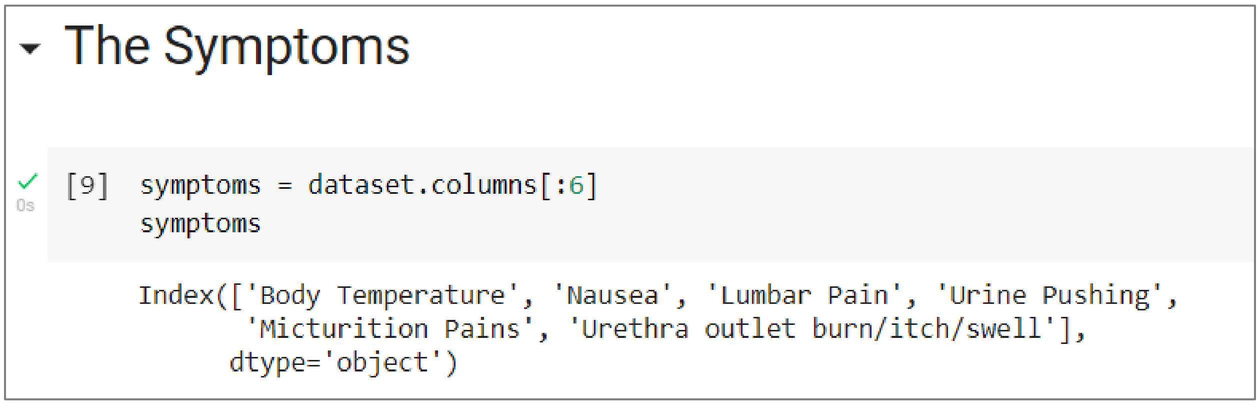


Testing Set

```
[8] test_set.head(10)
```

	Body Temperature	Nausea	Lumbar Pain	Urine Pushing	Micturition Pains	Urethra outlet burn/itch/swell	Inflammation of Urinary Bladder	train or not
7	41	0	1	1	0	1	0	False
8	37	0	1	0	0	0	0	False
13	37	0	0	1	1	1	1	False
17	37	0	0	1	0	0	1	False
19	40	1	1	0	1	0	0	False
20	35	0	0	1	1	1	1	False
27	37	0	0	1	1	0	1	False
38	41	0	1	1	0	1	0	False
52	37	0	0	1	1	1	1	False
66	40	1	1	0	1	0	0	False

4. **Next, the columns with the symptoms were assigned to a separate variable.**

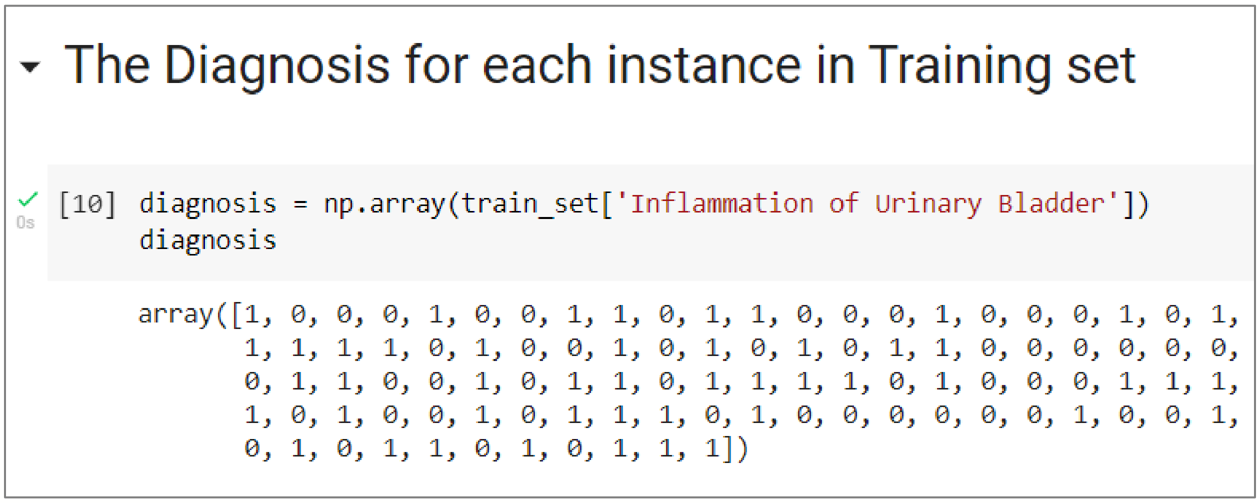


The Symptoms

```
[9] symptoms = dataset.columns[:6]
    symptoms
```

```
Index(['Body Temperature', 'Nausea', 'Lumbar Pain', 'Urine Pushing',
       'Micturition Pains', 'Urethra outlet burn/itch/swell'],
      dtype='object')
```

5. **Then the diagnosis results of all the instances of the training set were taken into a separate array.**

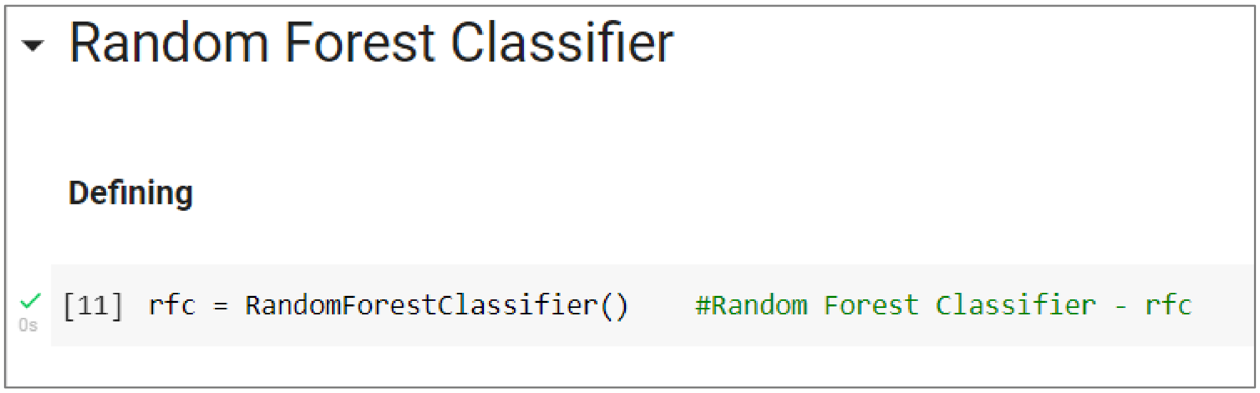


The Diagnosis for each instance in Training set

```
[10] diagnosis = np.array(train_set['Inflammation of Urinary Bladder'])
     diagnosis
```

```
array([1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1,
       1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1])
```

6. **The Random Forest Classifier was defined.**

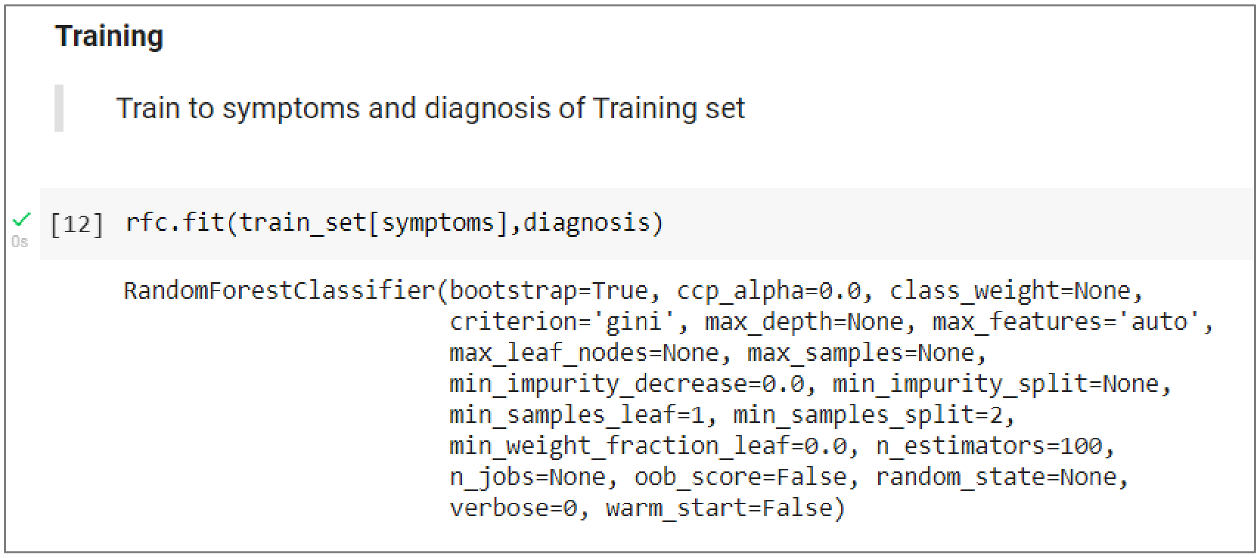


Random Forest Classifier

Defining

```
[11] rfc = RandomForestClassifier()    #Random Forest Classifier - rfc
```

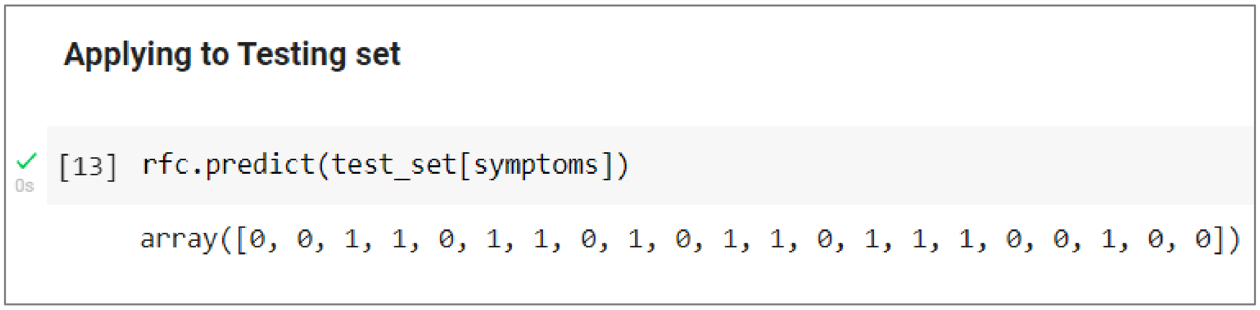
7. **The symptoms and diagnosis data of the training set was trained accordingly.**


Training

Train to symptoms and diagnosis of Training set

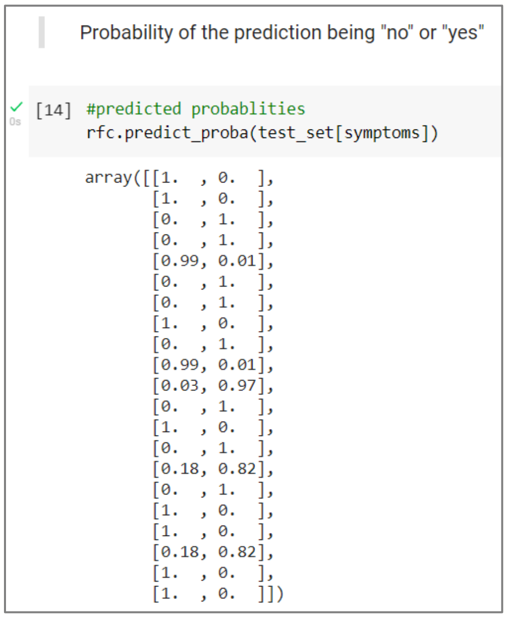
```
[12] rfc.fit(train_set[symptoms],diagnosis)

     RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                            criterion='gini', max_depth=None, max_features='auto',
                            max_leaf_nodes=None, max_samples=None,
                            min_impurity_decrease=0.0, min_impurity_split=None,
                            min_samples_leaf=1, min_samples_split=2,
                            min_weight_fraction_leaf=0.0, n_estimators=100,
                            n_jobs=None, oob_score=False, random_state=None,
                            verbose=0, warm_start=False)
```

8. **Then the learnt knowledge was applied to the testing set**


Applying to Testing set

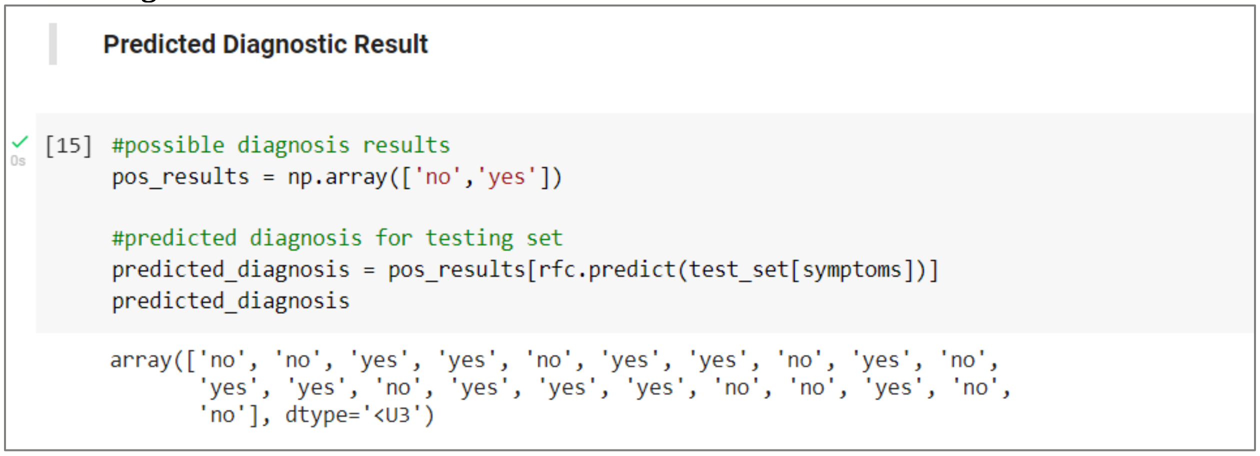
```
[13] rfc.predict(test_set[symptoms])

     array([0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0])
```

These predictions were the result of selecting the decision with highest probability. Each decision has a certain probability of being either "no" or "yes". The probabilities can also be checked.


Probability of the prediction being "no" or "yes"

```
[14] #predicted probablities
     rfc.predict_proba(test_set[symptoms])

     array([[1.  , 0.  ],
            [1.  , 0.  ],
            [0.  , 1.  ],
            [0.  , 1.  ],
            [0.99, 0.01],
            [0.  , 1.  ],
            [0.  , 1.  ],
            [1.  , 0.  ],
            [0.  , 1.  ],
            [0.99, 0.01],
            [0.03, 0.97],
            [0.  , 1.  ],
            [1.  , 0.  ],
            [0.  , 1.  ],
            [0.18, 0.82],
            [0.  , 1.  ],
            [1.  , 0.  ],
            [1.  , 0.  ],
            [0.18, 0.82],
            [1.  , 0.  ],
            [1.  , 0.  ]])
```

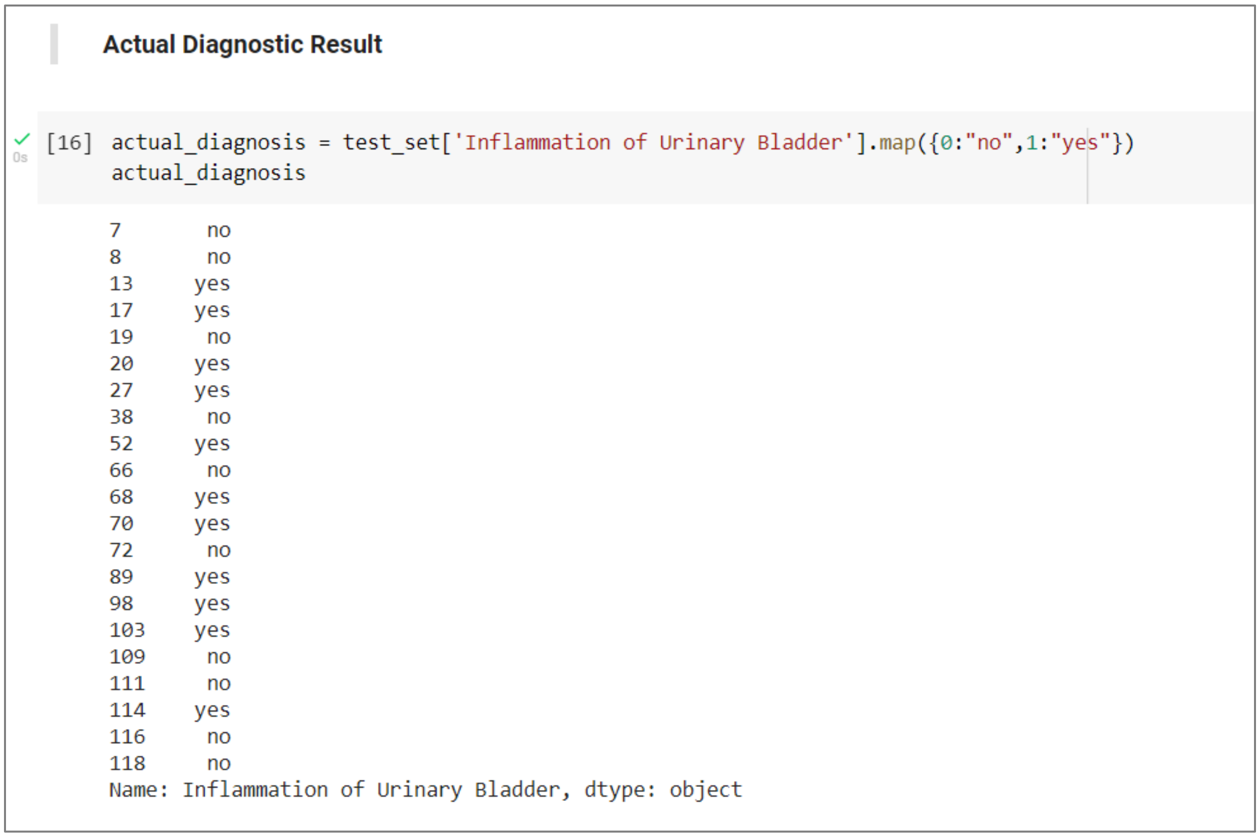
9. **The predicted diagnosis results of the testing dataset can be compared with the actual diagnostic results of those instances.**



Predicted Diagnostic Result

```
[15] #possible diagnosis results
     pos_results = np.array(['no','yes'])

     #predicted diagnosis for testing set
     predicted_diagnosis = pos_results[rfc.predict(test_set[symptoms])]
     predicted_diagnosis
```

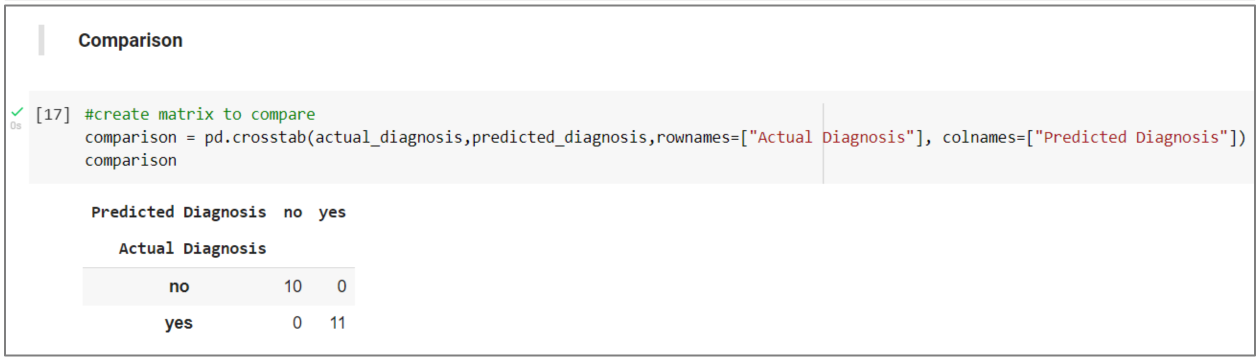
```
     array(['no', 'no', 'yes', 'yes', 'no', 'yes', 'yes', 'no', 'yes', 'no',
            'yes', 'yes', 'no', 'yes', 'yes', 'yes', 'no', 'no', 'yes', 'no',
            'no'], dtype='<U3')
```



Actual Diagnostic Result

```
[16] actual_diagnosis = test_set['Inflammation of Urinary Bladder'].map({0:"no",1:"yes"})
     actual_diagnosis
```

```
     7       no
     8       no
     13     yes
     17     yes
     19      no
     20     yes
     27     yes
     38      no
     52     yes
     66      no
     68     yes
     70     yes
     72      no
     89     yes
     98     yes
     103    yes
     109     no
     111     no
     114    yes
     116     no
     118     no
     Name: Inflammation of Urinary Bladder, dtype: object
```

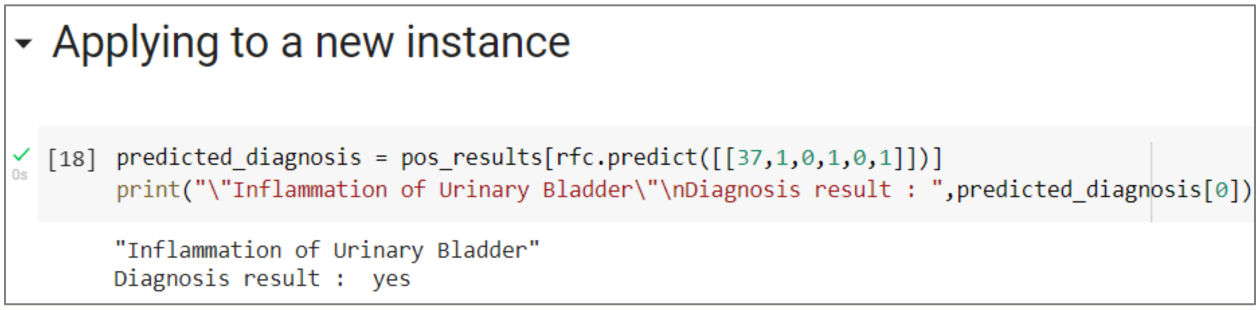


Comparison

```
[17] #create matrix to compare
     comparison = pd.crosstab(actual_diagnosis,predicted_diagnosis,rownames=["Actual Diagnosis"], colnames=["Predicted Diagnosis"])
     comparison
```

```
     Predicted Diagnosis  no  yes
        Actual Diagnosis
                      no  10    0
                     yes   0   11
```

10. **Finally, this trained random forest could be applied to any new set of symptoms for a new unforeseen instance to get the resulting diagnosis.**



▾ Applying to a new instance

```
[18] predicted_diagnosis = pos_results[rfc.predict([[37,1,0,1,0,1]])]
     print("\"Inflammation of Urinary Bladder\"\nDiagnosis result : ",predicted_diagnosis[0])

     "Inflammation of Urinary Bladder"
     Diagnosis result :  yes
```

Inferences and conclusions

Random Forests are often considered as one of the best to make accurate predictions. They can readily accommodate missing values, nonlinear relationships, interactions, and a large number of covariates.

In this case, the application of the Random Forest Machine Learning technique has provided 100% accurate results for the testing set after training for the training set. The 100% accuracy maybe due to the fact that this is a very limited dataset with only 120 instances. With much larger dataset this may vary. However, the accuracy might still be very high and close to 100%.

To try out the algorithm upon a new instance, I applied the symptoms as follows:

1. Temperature of patient	37
2. Occurrence of nausea	yes
3. Lumbar pain	no
4. Urine pushing (continuous need for urination)	yes
5. Micturition pains	no
6. Burning of urethra, itch, swelling of urethra outlet	yes

For these inputs, the algorithm decided that the diagnosis result should be “yes”.

Likewise, we can develop a system using this algorithm to produce diagnosis predictions for “Inflammation of Urinary Bladder” according to symptom inputs.

References

- *Pandas Introduction*. (n.d.). W3Schools. Retrieved October 12, 2021, from https://www.w3schools.com/python/pandas/pandas_intro.asp
- *Introduction to NumPy*. (n.d.). W3Schools. Retrieved October 12, 2021, from https://www.w3schools.com/python/numpy/numpy_intro.asp
- *1.11. Ensemble methods*. (n.d.). Scikit-Learn. Retrieved October 12, 2021, from https://scikit-learn.org/stable/modules/ensemble.html
- *sklearn.ensemble.RandomForestClassifier*. (n.d.). Scikit-Learn. Retrieved October 12, 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- *What does numpy.random.seed(0) do?* (2014, February 1). Stack Overflow. Retrieved October 12, 2021, from https://stackoverflow.com/questions/21494489/what-does-numpy-random-seed0-do
- Analytics, P. (2019, December 19). *What Is Machine Learning: Definition, Types, Applications and Examples*. Potentia Analytics. Retrieved October 10, 2021, from https://www.potentiaco.com/what-is-machine-learning-definition-types-applications-and-examples/
- Kumar, Arun & Salau, Ayodeji & Gupta, Swati & Arora, Sandeep. (2018). A Survey of Machine Learning Methods for IoT and their Future Applications.
- Education, I. C. (2021, June 30). *Supervised Learning*. IBM. Retrieved October 10, 2021, from https://www.ibm.com/cloud/learn/supervised-learning
- Goyal, K. (2021, April 3). *6 Types of Supervised Learning You Must Know About in 2021*. UpGrad Blog. Retrieved October 10, 2021, from https://www.upgrad.com/blog/types-of-supervised-learning/

- Yiu, T. (2021, September 29). *Understanding Random Forest - Towards Data Science*. Medium. Retrieved October 10, 2021, from https://towardsdatascience.com/understanding-random-forest-58381e0602d2
- Meena, M. (2020, December 8). *Applications of Random Forest*. OpenGenus IQ: Computing Expertise & Legacy. Retrieved October 10, 2021, from https://iq.opengenus.org/applications-of-random-forest/
- Molina, E. (2021, March 19). *A Practical Guide to Implementing a Random Forest Classifier in Python*. Medium. Retrieved October 10, 2021, from https://towardsdatascience.com/a-practical-guide-to-implementing-a-random-forest-classifier-in-python-979988d8a263
- Gupta, P. (2018, June 20). *Decision Trees in Machine Learning - Towards Data Science*. Medium. Retrieved October 10, 2021, from https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052

- THE END -