CT475 Assignment 1 :Description
Surya Balakrishnan Ramakrishnan
18231072
MSc Computer Science (Data Analytics)

Task 1: Selecting an open source Machine Learning Package.

For the task in hand I think the Random Forest Package would be most suited. The Random Forest is a supervised learning algorithm which is based on ensemble learning. In ensemble learning the model uses several algorithms combined together several times for constructing a powerful prediction model. The random forest algorithm combines several decision trees to form a forest of trees for the purpose of classification, hence justifying its name Random Forest Classifier.

Why Random Forest?

The random forest package is capable of performing classification and regression. The random forest algorithm as already stated is a supervised learning algorithm which works well with categorical variables and numeric variables in determining the target variable. In the given dataset **Autoimmune Dataset** there are a total of 8 numeric variables namely Blood_Pressure, BMI, Plasma_level, Adverse_events, Drug_in_serum, Liver_function, Activity_test, Secondary_test and the categorical variables Age, Autoimmune_Disease. Here the target variable is Autoimmune_Diesease. We are given a set of 376 records of patients with for whom the tests was done to check if they had autoimmune disease. The random forest classification would be able to classify any patient given the numeric variables whether if the patient has autoimmune disease or not.

Advantages of Random Forest

1) The random forest algorithm is not biased as the result comes from several trees
2) Even if new data point is introduced the random forest algorithm remains stable
3) The random forest algorithm works with both categorical and numeric variable.
4) The random forest algorithm works well even if some data is missing

Task 2: Performing Data Transformation

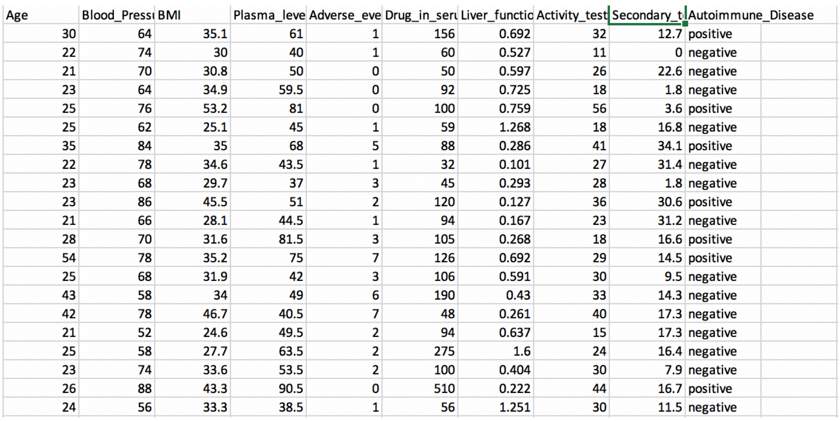
The given dataset **Autoimmune.txt** is a tab separated dataset. The first task would be to convert the txt file into a csv file. This step involves converting tab separated values into comma separated values. The following python code accomplishes the task.

```
import csv
txt_file = r"Autoimmune.txt"
csv_file = r"Autoimmune.csv"
with open(txt_file, "r") as in_text:
        in_reader = csv.reader(in_text, delimiter = '\t')
with open(csv_file, "w") as out_csv:
        out_writer = csv.writer(out_csv)
        for row in in_reader:
                out_writer.writerow(row)
```

once this is accomplished we transform the dataset by making it suitable for feeding it to the classification algorithm by doing some transformation of data. Also the target variable is set as the last column for the classifier to detect it. Here the most suitable method to feed the data set to the algorithm would be to feed each of the patient record row wise for which the transpose of the dataset has to be taken which is accomplished by the following python code

```
import csv
from itertools import izip
a = izip(*csv.reader(open("input.csv", "rb")))
csv.writer(open("output.csv", "wb")).writerows(a)
```
Then we add the headers for each of the column headers using the following python code.
```
import csv
fields=['Age', 'Blood_Pressure', 'BMI', 'Plasma_level', 'Adverse_events', 'Drug_in_serum', 'Liver_function', 'Activity_test', 'Secondary_test','Autoimmune_Disease']
with open(r'Autoimmune.csv', 'a') as f:
    writer = csv.writer(f)
    writer.writerow(fields)
```

The final csv would be of the following structure.



Age	Blood_Pressı	BMI	Plasma_leve	Adverse_eve	Drug_in_serı	Liver_functio	Activity_test	Secondary_t	Autoimmune_Disease
30	64	35.1	61	1	156	0.692	32	12.7	positive
22	74	30	40	1	60	0.527	11	0	negative
21	70	30.8	50	0	50	0.597	26	22.6	negative
23	64	34.9	59.5	0	92	0.725	18	1.8	negative
25	76	53.2	81	0	100	0.759	56	3.6	positive
25	62	25.1	45	1	59	1.268	18	16.8	negative
35	84	35	68	5	88	0.286	41	34.1	positive
22	78	34.6	43.5	1	32	0.101	27	31.4	negative
23	68	29.7	37	3	45	0.293	28	1.8	negative
23	86	45.5	51	2	120	0.127	36	30.6	positive
21	66	28.1	44.5	1	94	0.167	23	31.2	negative
28	70	31.6	81.5	3	105	0.268	18	16.6	positive
54	78	35.2	75	7	126	0.692	29	14.5	positive
25	68	31.9	42	3	106	0.591	30	9.5	negative
43	58	34	49	6	190	0.43	33	14.3	negative
42	78	46.7	40.5	7	48	0.261	40	17.3	negative
21	52	24.6	49.5	2	94	0.637	15	17.3	negative
25	58	27.7	63.5	2	275	1.6	24	16.4	negative
23	74	33.6	53.5	2	100	0.404	30	7.9	negative
26	88	43.3	90.5	0	510	0.222	44	16.7	positive
24	56	33.3	38.5	1	56	1.251	30	11.5	negative

Task 3: Running the Classification algorithm

(1) Naive Bayes Classifier :

Naïve Bays Classifier is a Bayesian classifier which suits well if there is a large number of records to classify. Even though the Naive Bayes classifier is one of the simplest classification algorithm it can outperform several complex algorithms. In this case the algorithm has to classify a whether the patient has Autoimmune Disease by classifying each of the new records into Positive or Negative Autoimmune_Disease (Target Variable) given the dataset **Autoimmune.csv**. The algorithm works based on a probabilistic model. The naïve bays classifier computes the class probability of each of the numeric variables of being positive and negative. In other words the classifier calculates the probability of the required hypothesis given the event ie P(H|E). Once the class probabilities are calculated the classifier classifies the record as Positive if the probability of positive is greater than negative and vice versa. The algorithm is run on the training dataset and to check if the results are consistent the algorithm is run on the test dataset. The split ratio of training and test datasets was fixed at 0.61. In this case for a total of 376 records the accuracy of 71.3%. In other words around the classifier got around 268 out of 376 records correctly classified.

(2) K Nearest Neighbours:

The K Nearest Neighbours algorithm is a supervised machine learning algorithm. The K Nearest Neighbours or in other words KNN is a type of lazy learning algorithm. The KNN algorithm doesn't use a separate training and test dataset for classification. It takes in all the given data for training and classifies a new data point. The KNN is a type of non-parametric learning algorithm which means no assumptions are made for the given data. The principle behind the KNN is very simple, it calculates the distance between the new data point to all other data points in the dataset. The distance calculation can either be Euclidian or Manhattan distance. In this case the KNN algorithm has to classify similar instances of data into either positive or negative. The initial step involves the KNN algorithm to plot all the 376 patient records in the given dataset. Now when a new record is given as input for the KNN algorithm the algorithm first analyses the different numerical and categorical variables of the new record and plots them. Then the KNN algorithm calculates the distance between the new record and other records. In this case number of nearest neighbours N was taken as 10 which is around 2.5% of the records. The more the N better the classifier performs. Now the algorithm calculates if number of positive neighbours is greater than the negatives neighbour or the other way around. Then the algorithm classifies the new data point into one of the two.

Task 4: Performing 10 Fold Cross Validation

We use the following python code to perform the 10 fold cross validation.

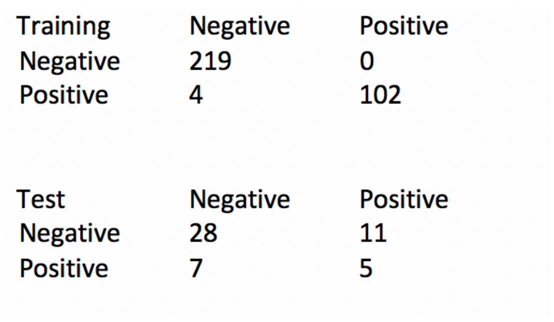
```
 from sklearn.model_selection import KFold
import pandas as pd

# data sample
data = pd.read_csv("Autoimmune.csv")

kfold = KFold(10, True, 1)
# enumerate splits
for train, test in kfold.split(data):
    print('train: %s, test: %s' % (train, test))

# prepare cross validation
kfold = KFold(3, True, 1)
# enumerate splits
for train, test in kfold.split(data):
    print('train: %s, test: %s' % (data[train], data[test]))
```

the obtained confusion matrix:



Training	Negative	Positive
Negative	219	0
Positive	4	102

Test	Negative	Positive
Negative	28	11
Positive	7	5

Task 5:

The two models give the similar results due to the reason that both the models return the same accuracy rate of 71.3%. One of the probable reasons for similar results could be the reason that since naïve bays classifier is best suited for a very large scale dataset. Also the accuracy of the classifier would increase if the number of patient records were more. Then in that case the classifier would have more records in the test and training datasets which would yield the classifier to be more accurate.

References:

1) https://www.kdnuggets.com/2015/06/top-20-r-machine-learning-packages.html
2) https://stackoverflow.com/questions/42776743/how-to-convert-a-tab-delimited-text-file-to-a-csv-file-in-python
3) https://stackoverflow.com/questions/2363731/append-new-row-to-old-csv-file-python
4) https://stackabuse.com/random-forest-algorithm-with-python-and-scikit-learn/
5) https://stackabuse.com/the-naive-bayes-algorithm-in-python-with-scikit-learn/
6) https://machinelearningmastery.com/k-fold-cross-validation/
7) https://stackoverflow.com/questions/4869189/how-to-transpose-a-dataset-in-a-csv-file