Heapatits

by iaprep2018 1

Submission date: 06-Dec-2022 06:16AM (UTC-0800)
Submission ID: 1973193082
File name: Hepatitis_B_Detection.docx (98.77K)
Word count: 6434
Character count: 40586

Hepatitis B classification using Machine learning techniques

First Author[#], Second Author[*], Third Author[#]

[#]*First-Third Department, First-Third University*
Address
[1]first.author@first-third.edu
[3]third.author@first-third.edu

[*]*Second Company*
Address Including Country Name
[2]second.author@second.com

Abstract— **Chronic HBV infection is a key risk factor for the development of advanced liver disease such as fibrosis, cirrhosis, and hepatocellular carcinoma (HCC). The proportional contribution of virological variables to illness development remains unknown, and methods to help in the deconvolution of complicated patient virus profiles are an unmet therapeutic need. Individual patients develop variable viral mutation signatures because of the viral polymerase's low-fidelity replication, resulting in 'quasispecies' populations. We offer the first comprehensive analysis of HBV quasispecies diversity based on ultra-deep sequencing of the whole HBV genome in two separate European and Asian patient groups. Seroconversion to the HBV e antigen (HBeAg) is an important clinical marker in infected patients. A model was constructed using a machine learning method to find the viral variations that properly characterise HBeAg status. Through enhanced patient categorization, serial surveys of patient quasispecies populations and sophisticated analytics will allow clinical decision support for persistent HBV infection and drive therapy options.**

***Keywords*— Machine learning, HBeAg, Hepatitis B, Logistic regression, Random Forest classifier, Decision tree classifier.**

I. Introduction

HBV is a DNA virus with a loosely circular, partly double-stranded genome and a distinct replication process. Because of the low fidelity reverse transcriptase and the fast replication rate, a single offspring virus in an infected cell may not be identical to the parent genome2. This results in the formation of a viral ecosystem made up of mutant swarms or 'quasispecies,' which are populations of genetically separate but closely related viral variations. As a result, HBV quasispecies have a range of viral variants with varying fitness, allowing for fast adaptability to selection pressures such as host immunological factors and antiviral medicines. As a result, HBV variations have an influence on illness development, clinical progression, and responsiveness to treatment interventions (if positively selected to sufficient abundance). Estimates for infection prevalence (4%), absolute number of infected persons (257 million), and yearly mortality (887000) show the substantial worldwide illness burden owing to HBV12,13. Viral hepatitis, along with hepatitis C virus (HCV), is the major cause of hepatocellular cancer. HBV infection is still a major public health concern across the world. HBV has infected around 257 million people, while CHB affects more than 350 million people. It is widely established that HBsAg seroclearance is an essential prognostic marker during CHB therapy. In persistently HBV-infected individuals, the yearly rate of spontaneous HBsAg seroclearance ranged from 0.45% to 2.38%, demonstrating that HBsAg seroclearance is an uncommon occurrence. Previous research has revealed a link between spontaneous or therapy induced HBsAg seroclearance and a better prognosis, improved liver histology, a lower incidence of hepatocellular carcinoma (HCC), and longer longevity. As a result, HBsAg seroclearance

is an essential objective for improving antiviral medication outcomes.

Previous research has found evidence of significant viral variables and host features of HBsAg seroclearance. Researchers looked at whether low blood HBsAg levels alone or in combination with a low serum HBV DNA load were relevant predictors of HBsAg seroclearance. In terms of host characteristics, age is one of the most important in HBsAg seroconversion, followed by gender, fatty liver, cirrhosis at baseline or acquired during follow-up, and baseline alanine aminotransferase (ALT) levels. Previous research building prediction models, on the other hand, were mostly focused on long-term tracking of restricted components and traditional statistical approaches, which may have distorted the estimations due to the possible collinearity issue for high-dimensional medical data. To fill a knowledge gap, we applied machine learning techniques rather than conventional models in this work to assess the relationship between available clinical factors and HBsAg seroclearance. In recent years, machine learning algorithms have received a lot of interest in the health area. It has been used effectively as a strong classification approach to extract useful information from high-dimensional, correlated, nonlinear, and unbalanced clinical datasets and make accurate diagnostic and prediction judgments. However, no current models have been found as having the highest performance for predicting HBsAg seroclearance. In this study, we built many acceptable machine learning models, including Random Forest classifier, Decision tree classifier, and LR, based on the dataset's features (highly dimensional and unbalanced), with the goal of identifying the best one. The primary goal of this project is to find the best machine learning model for predicting HBsAg seroclearance in a retrospective cohort of CHB patients.

II. Problem and Data set(s)

The Our problem statement is the binary classification problem statement. The objective of this task is to identify hepatitis a patient based on his history, and we are going to tell the life prediction of patient so here we are having a target variable called class and it is having a status either die or live.

III. Methods

Here for our analysis, we are considering three machine learning algorithms those are logistic regression Random Forest and decision tree the in-detail explanation of each algorithm is given below:

Decision tree:

It is a tree-structured classifier where all internal nodes or columns of the data set and branches explain the decision rules that are taken by the decision tree of outcome. Essentially, in a decision tree, we will have two notes that are the output of the decision note. This leaf notes will not be having any further branches but decision note instead. Decision tree is also a supervisor learning technique that will perform both regression and classification task for giving data. Additionally, one of the most elegant algorithms, the decision tree, has a graphic depiction of every step of how it functions, and the decision tree can be visualised once the model is ready for use, and it expands for the branches from the root note to construct a structure like this. We will use the cart algorithm, which combines classification and regression, to build the decision tree. The decision tree can include both numerical and category data. The rationale for adopting decision trees is that they typically function similarly to human decision-making processes and can be easily understood and seen due to their tree-like form. Due to the simplicity of decision tree operation, it is much simpler to comprehend how decision tree's function. To make new decisions, we must first choose the given complete data set. Next, using an attribute selection measure, we must identify the best attribute within the set. Finally, we must split the data set into subsets that contain the best attribute's potential values. Finally, we must create the decision tree no, which contains the best attribute. We may utilise the Gini index or information gain to choose the root node, and we can prune to reduce the chance of overfitting.

Logistic regression:

One of the most used machine learning techniques, logistic regression uses input label data to classify the output label. In essence, logistic regression is

employed to address classification issues, and it will use several independent variables together with one dependent variable as the output for training and performing classification. Regression essentially forecasts the results of categorical dependent variables; therefore, the results will be expressed as either yes or no, 0 or 1, or binary output. Therefore, it will return a numerical value and, using the Sigmoid function, transform that value to either 0 or 1, rather than only returning 0 or 1. Except for the final layer, which consists of this probability layer that turns ordinal values into binary values, logistic regression is nearly identical to the linear regression employed in regression procedures. Instead of fitting a regression line, we will fit a sigmoid curve for logistic regression; this sigmoid curve will help us generate probabilities from regression values and explains the likelihood of our problem statement, such as whether the image is a cat or not or the patient is pregnant or not. Logistic regression assumes that the dependent variable must be categorical and that the independent variables must not be multicollinear. The only difference between linear regression and logistic regression is logistic regression uses the concept of predictive modelling like linear regression but it is used to classify samples so it will fall under classification algorithm. The main distinction between logistic regression and linear regression is that the latter employs the idea of predictive modelling, much like the former, while the former uses it to classify data to fall under a classification procedure.

Random Forest classifier:

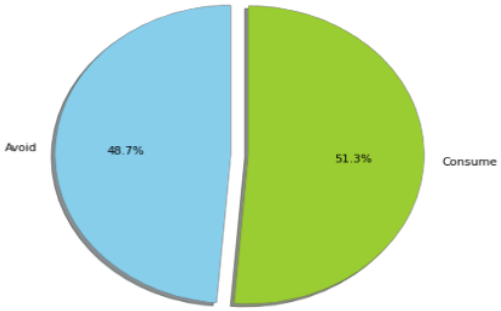
As we will create multiple decision trees under each algorithm, ran over is essentially the process of combining multiple classifiers to solve a complex problem and improve the performance of the model. Random forest is a popular machine learning algorithm that belongs to the branch of supervised machine learning and can be used for both classification and regression problems in machine learning. The more decision trees we build, the more accurate the classifier will be since random forest is described as a classifier that consists of varying numbers of decision trees that belong to various subsets of the provided data and averages out the predictive accuracy of the data set. We'll receive There are a few presumptions associated with this random forest; they are that each decision tree's predictions will have the fewest core relationships and that there will be some actual values in the data set's future variables so that the random forest can produce accurate results rather than random ones. The output produced by random forest is having high accuracy even for larger data sets, and it maintains stability in the output, which are the reasons for choosing random for s for this problem. The working of random forest algorithm is as follows: first will select random K point for training set and random K point for output.

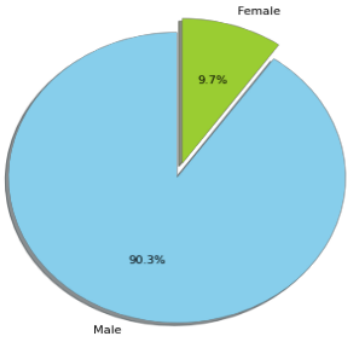
We'll receive There are a few presumptions associated with this random forest; they are that each decision tree's predictions will have the fewest core relationships and that there will be some actual values in the data set's future variables so that the random forest can produce accurate results rather than random ones. The output produced by random forest is having high accuracy even for larger data sets, and it maintains stability in the output, which are the reasons for choosing random for s for this problem. The working of random forest algorithm is as follows: first will select random K point for training set and random K point for output. We will construct decision trees for the subsets of data that are provided, then select the number of decision trees that should be constructed and repeat the process several times. Finally, we will export the model, and for new data points, we can find the predictions of new data points to the category that it belongs to base on the majority vote that it received. The few advantages of the random forest algorithm are that it can perform both classification and regression tasks, and it can also be used to perform regression tasks.

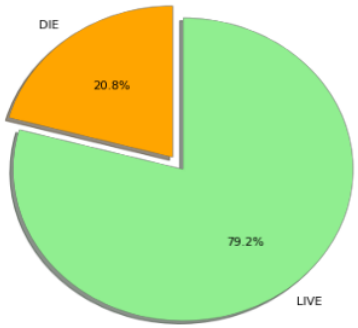
IV. Experimental setup

As part of exploratory data analysis, firstly we are checking how many columns and Rows we are having and we are in our data set we are having 20 columns and 154 records for each column and real not having any missing values in our data set and now we have to convert all values into numeric and after converting into memory clear having few missing values in some columns and we treated that missing you with mode and median later we checked skewness for different columns that are available

with the data set and all things are seems to be normal it distributed and left, right distributed their showing degree of skinners and for that we are going to apply long transfer to make it normally distributed and from histograms we can see that skewness is present in our data and mainly fixed and also we can observe that some of our variables patient to differentiate according to whether they belongs to class zero OR one distinction is not completely clear there is no near relationship between the variables plotted though some of them we can observed in a train to an interaction we can analyse the relationship between our candy circle variables and numerical variables for this part will take out advantage of pay grade of c burn that will allow us to plant little more freedom to choose X and y variables and here we are choosing swan plot a particular case of scatter plant which do not overlap the point and It is possible to observe that there is no difference in the variables plotted regarding the ANOREXIA status. This can be evidenced by the fact that not only patients from both levels of Class are distributed homogeneously but also there is not difference in the expression of the variables analysed regarding the levels of ANOREXIA. On the other hand, we can see a trend that patients with Class 0 tend to have ascites. However, there is no differences in how the variables are expressed regarding ASCITES status. The last thing that we will explore is if there is any strong correlation between the parameters. For this task, we will use the Pearson correlation coefficient because it is a good parameter to know the strength of the linear relationship between two variables. The importance of performing correlation analysis is our dataset lies on the fact that highly correlated variables can hurt some models or in other cases, could provide little extra information and considering them can be computational expensive without any real benefit. Also, knowing if our variables display a linear relationship can help us choose which machine learning algorithm is more suitable for our data. To perform the correlation analysis with all our variables, we first need to apply the function factorize to the columns containing non continuous variables to obtain a numeric representation of the categorical values contained in the dataset. Here are few visualizations that we applied for available dataset







V. Results

So no of the main decisions to make when performing machine learning is choosing the appropriate algorithm that fits the current problem we are dealing with. Supervised learning refers to the task of inferring a function from a labelled training dataset. We fit the model to the labelled training set with the main goal of finding the optimal parameters that will predict unknown labels of new examples included in the test dataset. There are two main types

of supervised learning: regression, in which we want to predict a label that is a real number, and classification, in which we want to predict a categorical label. In our case, we have a labelled dataset, and we want to use a classification algorithm to find the label in the categorical values: 0 and 1.

We can find many classifications supervised learning algorithms, some simple but efficient, such as linear classifier or logistic regression, and other ones more complex but powerful such as decision trees and k-means.

In this case, we will choose Logistic regression, Random Forest algorithm. Random forest is one of the most used machine learning algorithms since it is very simple, flexible and easy to use but produces reliable results. So, we will load the packages from scikit-learn that we need to perform Random Forest and to evaluate afterwards the model. As logistic regression performs best at the first instance but no performance improvement after hyper parameter tuning, the model accuracy score is 0.80 i.e.., 80%. So, we elected next good performing algorithm that is Random Forest. As we can observe above, our basic model has an accuracy of 77.19% which tell us that it must be further improved. There are several ways to improve random forest model: gather more data, tune the hyperparameters of the model or choose other models. We will choose the second one, we will now tune the hyperparameters of our random forest classifier.

Model parameters are normally learned during training; however, hyperparameters must be set manually before training. In the case of random forest, hyperparameters include:

n_estimators: number of trees in the forest
max_features: maximum number of features in each tree
max_depth: maximum splits for all trees
bootstrap: whether to implement bootstrap or not to build trees
criterion: assess stopping criteria for decision trees

Of course, when we implement basic random forest, Scikit-learn implements a set of default hyperparameters, but we are not sure if those parameters are the optimal for our problem.

In this point is when we need to consider two concepts: underfitting and overfitting. Underfitting occurs when the model is too simple, and it doesn't fit the data well: it has low variance but high bias. On the other hand, overfitting occurs when the model adjusts too well to the training set and performs poorly in new examples. If we tune the hyperparameters in the training dataset, we would then be prone to overfit our random forest classifier. So instead, we will go back to what was mentioned before: the cross validation.

We will use K-fold cross validation method to tune the hyperparameters: we will perform many iterations on the K-subset cross validation but using different model settings each time. Afterwards, we compare all models and select the best one; then, we will train the best model in the full training set and evaluate it on the testing set. We will take advantage of GridSearchCV package in Scikit-learn to perform this task.

So, after applying random forest to our dataset, we can conclude that our best model was able to predict survival from patients with hepatitis with an accuracy of 77% and a precision and recall of around 80%. This is not the best situation since we want our model to perform better, especially in this case that involves survival of patients. However, the moderate good results could be due to the small database and the large number of missing values.

The classification report is as below:

	precision	recall	f1-score	support
0	1.00	0.30	0.46	10
1	0.75	1.00	0.86	21
accuracy			0.77	31
macro avg	0.88	0.65	0.66	31
weighted avg	0.83	0.77	0.73	31

and this model is having good balance between the recall and precision. The confusion matrix can be observed from the code submitted.

VI. Conclusions

This work reveals unique viral variation patterns related with current HBeAg status, however it draws no conclusions about how this status was achieved

(there is no previous clinical data) or what the model signifies in terms of patient outcomes (data was not part of a longitudinal study). The same applies to patient profiles that are not represented in the research population, such as HBeAg negative inactive carriers with low HBV DNA burdens, on which we are unable to comment. The goal of this work was not to establish an alternative diagnostic test; instead, the ML model was built based on the known HBeAg status as identified by conventional diagnostic methods. Even though we offer a classification model with excellent discriminative accuracy, this does not necessarily translate to modifications in clinical practise or decision-making. We need prospective trials with serial sampling to catch patients during the seroconversion process and track treatment groups to make such a model applicable. Furthermore, such a model must be calibrated to the target population. However, the general model's success in differentiating HBeAg status in the n = 37 patients undergoing treatment from the Dataset B cohort, which acted as an independent test group, gave us hope. It was able to develop a generic model for HBeAg classification with broad applicability to the clinical population by adding a diversified sample population for feature-selection. In addition, we describe the prevalence of mutants associated with resistance in naive patients. Previous research has demonstrated the existence of these mutants in naive patients30,53, and our work further supports the requirement for baseline sequencing of infected individuals in the future to customize therapeutic regimens. Although the value of ML approaches to clinical decision making in infectious diseases is currently underappreciated, the use of deep sequencing and ML analysis to identify data patterns could facilitate the targeting of specific therapeutic interventions to high risk groups, aid patient stratification for more effective clinical trial design, link models to clinical decision support tools, and, through the incorporation of patient demographic data, facilitate the design of clinical trials that are more effective and efficient. Our findings show that plasma HBV quasispecies represent viral populations within hepatocytes, and that when these profiles are interrogated using machine learning approaches, they can recapitulate patient classification by clinical marker status while also revealing novel biology.

VII. References

[1] E. Orito *et al*., "Host-independent evolution and a genetic classification of the hepadnavirus family based on nucleotide sequences.," *Proceedings of the National Academy of Sciences*, vol. 86, no. 18, pp. 7059–7062, Sep. 1989, doi: 10.1073/pnas.86.18.7059.

[2] "Classification Model for Hepatitis B Disease Using Supervised Machine Learning Technique," *Computer Engineering and Intelligent Systems*, May 2022, doi: 10.7176/ceis/13-3-01.

[3] A. YUSUF and O. AKANDE, "HEPATITIS DISEASES PREDICTION USING MACHINE-LEARNING TECHNIQUES," *FUDMA JOURNAL OF SCIENCES*, vol. 5, no. 3, pp. 1–8, Nov. 2021, doi: 10.33003/fjs-2021-0503-515.

[4] A. Orooji and F. Kermani, "Machine Learning Based Methods for Handling Imbalanced Data in Hepatitis Diagnosis," *Frontiers in Health Informatics*, vol. 10, no. 1, p. 57, Jan. 2021, doi: 10.30699/fhi.v10i1.259.

[5] H. Chown, "A Comparison of Machine Learning Algorithms for the Prediction of Hepatitis C NS3 Protease Cleavage Sites," *Journal of Proteomics & Bioinformatics*, vol. 12, no. 5, 2019, doi: 10.35248/0974-276x.19.12.501.


```
import pandas as pd
import numpy as np
from pandas import Series, DataFrame
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')


df =  pd.read_csv('hep.csv')
```

```
# In[3]:


df.head()


df.columns = ['class', 'age', 'sex', 'steroid',
'antivirals', 'fatigue', 'malaise',
       'anorexia', 'liver_big', 'liver_firm',
'spleen_palable', 'spiders',
       'ascites', 'varices', 'bilirubin', 'alk_phosphate',
'sgot', 'albumin',
       'protime', 'histology']


# column names was provide with Data in separate
file.

# ##### Display Data

# In[5]:


df.head()


# Now, we can see the column names.

# ##### Check Describe:

# In[6]:


df.describe()


# As we can, we unable to describe all the column
of our df set. lets check info of Data.

# ##### Check Information :

# In[7]:


df.info()
```

```
# Now, by comparing the info() and describe() of
df, only the integer df type are visible to us. As we
can see above, the dataset has 155 rows
corresponding to the number of patients included in
this study, and 20 columns, corresponding to the
features or characteristics collected for each patient.
#
#

# ##### Change dfType

# In[8]:


df['steroid'] =
pd.to_numeric(df['steroid'],errors='coerce')
df['fatigue'] =
pd.to_numeric(df['fatigue'],errors='coerce')
df['malaise'] =
pd.to_numeric(df['malaise'],errors='coerce')
df['anorexia'] =
pd.to_numeric(df['anorexia'],errors='coerce')
df['liver_big'] =
pd.to_numeric(df['liver_big'],errors='coerce')
df['liver_firm'] =
pd.to_numeric(df['liver_firm'],errors='coerce')
df['spleen_palable'] =
pd.to_numeric(df['spleen_palable'],errors='coerce')
df['spiders'] =
pd.to_numeric(df['spiders'],errors='coerce')
df['ascites'] =
pd.to_numeric(df['ascites'],errors='coerce')
df['varices'] =
pd.to_numeric(df['varices'],errors='coerce')
df['bilirubin'] =
pd.to_numeric(df['bilirubin'],errors='coerce')
df['alk_phosphate'] =
pd.to_numeric(df['alk_phosphate'],errors='coerce')
df['sgot'] = pd.to_numeric(df['sgot'],errors='coerce')
df['albumin'] =
pd.to_numeric(df['albumin'],errors='coerce')
df['protime'] =
pd.to_numeric(df['protime'],errors='coerce')


# ##### Check Info()
```

```
# In[9]:


df.info()


# ##### Replace (1,2) values with (0,1)

# In[10]:


df["class"].replace((1,2),(0,1),inplace=True)
df["sex"].replace((1,2),(0,1),inplace=True)
df["age"].replace((1,2),(0,1),inplace=True)
df["steroid"].replace((1,2),(0,1),inplace=True)
df["antivirals"].replace((1,2),(0,1),inplace=True)
df["fatigue"].replace((1,2),(0,1),inplace=True)
df["malaise"].replace((1,2),(0,1),inplace=True)
df["anorexia"].replace((1,2),(0,1),inplace=True)
df["liver_big"].replace((1,2),(0,1),inplace=True)
df["liver_firm"].replace((1,2),(0,1),inplace=True)
df["spleen_palable"].replace((1,2),(0,1),inplace=True)
df["spiders"].replace((1,2),(0,1),inplace=True)
df["ascites"].replace((1,2),(0,1),inplace=True)
df["varices"].replace((1,2),(0,1),inplace=True)
df["histology"].replace((1,2),(0,1),inplace=True)


# In[11]:


df.head()


# ##### Check Null Values

# In[12]:


df.isna().sum()


# Here we can see many null values.

# ##### Fill Null Values
df['steroid'].mode()
df['steroid'].replace(to_replace=np.nan,value=1,inplace=True)

df['fatigue'].mode()
df['fatigue'].replace(to_replace=np.nan,value=0,inplace=True)

df['malaise'].mode()
df['malaise'].replace(to_replace=np.nan,value=1,inplace=True)

df['anorexia'].mode()
df['anorexia'].replace(to_replace=np.nan,value=1,inplace=True)

df['liver_big'].mode()
df['liver_big'].replace(to_replace=np.nan,value=1,inplace=True)

df['liver_firm'].mode()
df['liver_firm'].replace(to_replace=np.nan,value=1,inplace=True)

df['spleen_palable'].mode()
df['spleen_palable'].replace(to_replace=np.nan,value=1,inplace=True)

df['spiders'].mode()
df['spiders'].replace(to_replace=np.nan,value=1,inplace=True)

df['ascites'].mode()
df['ascites'].replace(to_replace=np.nan,value=1,inplace=True)

df['varices'].mode()
df['varices'].replace(to_replace=np.nan,value=1,inplace=True)

df['bilirubin'].skew(axis=0,skipna = True)
df['bilirubin'].median()
df['bilirubin'].replace(to_replace=np.nan,value=1,inplace=True)

df['alk_phosphate'].skew(axis=0,skipna = True)
df['alk_phosphate'].median()
```

```
df['alk_phosphate'].replace(to_replace=np.nan,value=85,inplace=True)


df['sgot'].skew(axis=0,skipna = True)
df['sgot'].median()
df['sgot'].replace(to_replace=np.nan,value=58,inplace=True)


df['albumin'].skew(axis=0,skipna = True)
df['albumin'].median()
df['albumin'].mean()
df['albumin'].replace(to_replace=np.nan,value=4,inplace=True)


df['protime'].skew(axis=0,skipna = True)
print("Here skewness is near to symmetri, so we can check both mean and median")

df['protime'].median()
df['protime'].mean()
df['protime'].replace(to_replace=np.nan,value=61,inplace=True)
print("Now we filled all the null values.")


# In[14]:


plt.figure(figsize=(6,3.5))
plt.subplot(1, 2, 1)
sns.distplot(df['sgot'],

kde_kws={"color":"blue","lw":1.5,"alpha":0.8},
         hist_kws={"color":"green","alpha":0.3})
plt.subplot(1, 2, 2)
sns.distplot(df['alk_phosphate'],

kde_kws={"color":"red","lw":1.5,"alpha":0.8},
         hist_kws={"color":"pink","alpha":0.6})
sns.despine();


# In[15]:


plt.figure(figsize=(7,3.5))
plt.subplot(1, 2, 1)
sns.distplot(df['bilirubin'],

kde_kws={"color":"green","lw":1.5,"alpha":0.8},

hist_kws={"color":"lightblue","alpha":0.8})
sns.despine()
plt.subplot(1, 2, 2)
sns.distplot(df['albumin'],

kde_kws={"color":"red","lw":1.5,"alpha":0.8},
         hist_kws={"color":"orange","alpha":0.3})
sns.despine();


# We can observed in the histograms that in fact several of our variables, including ALK_PHOSPHATE and SGOT that we had detected in the summary statistics, show a degree of skeweness. In order to fix that, we will log-transform our variables.

# In[16]:


g = sns.pairplot(df, x_vars = ['bilirubin', 'protime', 'alk_phosphate', 'sgot', 'albumin'],
                 y_vars = ['bilirubin', 'protime', 'alk_phosphate', 'sgot', 'albumin'],
                 hue = 'class',
                 kind= 'scatter',
                 palette = 'husl',
                 size = 2,
                 plot_kws={"s": 35, "alpha": 0.8})
g.fig.get_children()[-1].set_bbox_to_anchor((0.05, 0.9, 0.18, 0.1));


# From the plots, we can highlight several things:
#
# From the histograms we learn that the skewness present in our data was mainly fixed
# We can observed that in some our variables patients tend to differentiate according to whether they belong to Class 0 or Class 1; However this distinction is not completely clear.
```

There is not a linear relationship between the variables plotted, though in some of them we can observed a trend to an interaction (SGOT and ALK_PHOSPHATE, SGOT and BILIRUBIN,PROTIME and ALBUMIN, BILIRUBIN and ALK_PHOSPHATE)
We can then analyze the relationship between our categorical variables and our numerical variables. For this part, we will take advantage of PairGrid of seaborn that allows us to plot with a little more freedom to choose the x and y variables. In this case, we will use the graph swamplot, a particular case of scatterplot which do not overlap the points

In[17]:

```
graph = sns.PairGrid(df,
                     x_vars=["anorexia", "ascites"],
                     y_vars=['bilirubin', 'protime',
'alk_phosphate', 'sgot', 'albumin'],
                     hue = 'class')
graph.map(sns.swarmplot, s = 6)
graph.add_legend(frameon=True,
bbox_to_anchor=(0.33, 0.96));
```

It is possible to observe that there is no difference in the variables plotted regarding the ANOREXIA status. This can be evidenced by the fact that not only patients from both levels of Class are distributed homogeneously but also there is not difference in the expression of the variables analyzed regarding the levels of ANOREXIA. On the other hand, we can see a trend that patients with Class 0 tend to have ascites. However, there is no differences in how the variables are expressed regarding ASCITES status.
#
The last thing that we will explore is if there is any strong correlation between the parameters. For this task, we will use the Pearson correlation coefficient because it is a good parameter to know the strength of the linear relationship between two variables. The importance of performing correlation analysis is our dataset lies on the fact that highly correlated variables can hurt some models or in other cases, could provide little extra information and considering them can be computational expensive without any real benefit. Also, knowing if our variables display a linear relationship can help us choose which machine learning algorithm is more suitable for our data.
#
In order to perform the correlation analysis with all our variables, we first need to apply the function factorize to the columns containing non continuos variables in order to obtain a numeric representation of the categorical values contained in the dataset.

Check Describe

In[18]:

```
df.describe()
```

Here we can see various parameters such as Mean,Standard Deviation,Minimum value,Maximum value,Median and quartiles of the Data.

In[19]:

```
df.head(10)
```

Visualization

Plot Pie Chart of Class Column.

In[20]:

```
die =len(df[df['class'] == 0])
live = len(df[df['class']== 1])

plt.figure(figsize=(8,6))

# df to plot
labels = 'DIE','LIVE'
sizes = [die,live]
colors = ['orange', 'lightgreen']
```

```
explode = (0.1, 0)  # explode 1st slice

# Plot
plt.pie(sizes, explode=explode, labels=labels,
colors=colors,
autopct='%1.1f%%', shadow=True, startangle=90)

plt.axis('equal')
plt.show()
```

Here we can see the Ratio of alive and died.

Plot Pie Chart of Sex Column

In[21]:

```
male =len(df[df['sex'] == 0])
female = len(df[df['sex']==1])

plt.figure(figsize=(8,6))

# df to plot
labels = 'Male','Female'
sizes = [male,female]
colors = ['skyblue', 'yellowgreen']
explode = (0.1, 0)  # explode 1st slice

# Plot
plt.pie(sizes, explode=explode, labels=labels,
colors=colors,
autopct='%1.1f%%', shadow=True, startangle=90)

plt.axis('equal')
plt.show()
```

Here we can see the ratio of male and female

Plot Pie Chart of Steroid Column

In[22]:

```
no =len(df[df['steroid'] == 0])
yes = len(df[df['steroid']== 1])

plt.figure(figsize=(8,6))

# df to plot
labels = 'Avoid','Consume'
sizes = [no,yes]
colors = ['skyblue', 'yellowgreen']
explode = (0.1, 0)  # explode 1st slice

# Plot
plt.pie(sizes, explode=explode, labels=labels,
colors=colors,
autopct='%1.1f%%', shadow=True, startangle=90)

plt.axis('equal')
plt.show()
```

Draw count plot to visualize consumption of steriod in relation with Age

In[23]:

```
plt.figure(figsize=(15,6))
sns.countplot(x='age',data =  df, hue = 'steroid')
plt.show()
```

Plot Pie Chart of antivirals Column

In[24]:

```
no =len(df[df['antivirals'] == 0])
yes = len(df[df['antivirals']== 1])

plt.figure(figsize=(8,6))

# df to plot
labels = 'Avoid','Consume'
sizes = [no,yes]
colors = ['skyblue', 'yellowgreen']
explode = (0.1, 0)  # explode 1st slice

# Plot
plt.pie(sizes, explode=explode, labels=labels,
colors=colors,
autopct='%1.1f%%', shadow=True, startangle=90)
```

```
plt.axis('equal')
plt.show()
```

Draw count plot to visualize consumption of antivirals in relation with Age

In[25]:

```
plt.figure(figsize=(15,6))
sns.countplot(x='age',data =  df, hue = 'antivirals',palette='GnBu')
plt.show()
```

Plot Pie Chart of fatigue Column

In[26]:

```
no =len(df[df['fatigue'] == 0])
yes = len(df[df['fatigue']== 1])

plt.figure(figsize=(8,6))

# df to plot
labels = 'Never Exausted','Was Excausted'
sizes = [no,yes]
colors = ['skyblue', 'yellowgreen']
explode = (0.1, 0)  # explode 1st slice

# Plot
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=90)

plt.axis('equal')
plt.show()
```

Draw Count plot to visualize count of people sufer from fatigue in relation with Age.

In[27]:

```
plt.figure(figsize=(15,6))
sns.countplot(x='age',data =  df, hue = 'fatigue',palette='BrBG')
plt.show()
```

Plot Pie Chart of malaise Column

In[28]:

```
no =len(df[df['malaise'] == 0])
yes = len(df[df['malaise']== 1])

plt.figure(figsize=(8,6))

# df to plot
labels = 'Never in Discomfort','Was in Discomfort'
sizes = [no,yes]
colors = ['skyblue', 'yellowgreen']
explode = (0.1, 0)  # explode 1st slice

# Plot
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=90)

plt.axis('equal')
plt.show()
```

Draw Count plot to visualize count of people sufer from malaise in relation with Age.

In[29]:

```
plt.figure(figsize=(15,6))
sns.countplot(x='age',data =  df, hue = 'malaise',palette='RdPu')
plt.show()
```

Plot Pie Chart of anorexia Column

In[30]:

```
no =len(df[df['anorexia'] == 0])
yes = len(df[df['anorexia']== 1])

plt.figure(figsize=(8,6))

# df to plot
labels = 'NO','YES'
sizes = [no,yes]
colors = ['skyblue', 'yellowgreen']
explode = (0.1, 0)  # explode 1st slice

# Plot
plt.pie(sizes, explode=explode, labels=labels,
colors=colors,
autopct='%1.1f%%', shadow=True, startangle=90)

plt.axis('equal')
plt.show()
```

Draw Count plot to visualize count of people sufer from anorexia in relation with Age.

In[31]:

```
plt.figure(figsize=(15,6))
sns.countplot(x='age',data =  df, hue =
'anorexia',palette='RdPu')
plt.show()
```

Plot Pie Chart of liver_big Column

In[32]:

```
no =len(df[df['liver_big'] == 0])
yes = len(df[df['liver_big']== 1])

plt.figure(figsize=(8,6))

# df to plot
labels = 'NO','YES'
sizes = [no,yes]
colors = ['skyblue', 'yellowgreen']
explode = (0.1, 0)  # explode 1st slice

# Plot
plt.pie(sizes, explode=explode, labels=labels,
colors=colors,
autopct='%1.1f%%', shadow=True, startangle=90)

plt.axis('equal')
plt.show()
```

Draw Count plot to visualize count of people sufer from liver_big in relation with Age.

In[33]:

```
plt.figure(figsize=(15,6))
sns.countplot(x='age',data =  df, hue =
'liver_big',palette='RdPu')
plt.show()
```

Plot Pie Chart of liver_firm Column

In[34]:

```
no =len(df[df['liver_firm'] == 0])
yes = len(df[df['liver_firm']== 1])

plt.figure(figsize=(8,6))

# df to plot
labels = 'NO','YES'
sizes = [no,yes]
colors = ['skyblue', 'yellowgreen']
explode = (0.1, 0)  # explode 1st slice

# Plot
plt.pie(sizes, explode=explode, labels=labels,
colors=colors,
autopct='%1.1f%%', shadow=True, startangle=90)

plt.axis('equal')
plt.show()
```

Draw Count plot to visualize count of people sufer from liver_firm in relation with Age.

In[35]:

```
plt.figure(figsize=(15,6))
sns.countplot(x='age',data =  df, hue =
'liver_firm',palette='RdPu')
plt.show()
```

Plot Pie Chart of spleen_palable Column

In[36]:

```
no =len(df[df['spleen_palable'] == 0])
yes = len(df[df['spleen_palable']== 1])

plt.figure(figsize=(8,6))

# df to plot
labels = 'NO','YES'
sizes = [no,yes]
colors = ['skyblue', 'yellowgreen']
explode = (0.1, 0)  # explode 1st slice

# Plot
plt.pie(sizes, explode=explode, labels=labels,
colors=colors,
autopct='%1.1f%%', shadow=True, startangle=90)

plt.axis('equal')
plt.show()
```

Draw Count plot to visualize count of people sufer from spleen_palable in relation with Age.

In[37]:

```
plt.figure(figsize=(15,6))
sns.countplot(x='age',data =  df, hue =
'spleen_palable',palette='RdPu')
plt.show()
```

Plot Pie Chart of spiders Column

In[38]:

```
no =len(df[df['spiders'] == 0])
yes = len(df[df['spiders']== 1])

plt.figure(figsize=(8,6))

# df to plot
labels = 'NO','YES'
sizes = [no,yes]
colors = ['skyblue', 'yellowgreen']
explode = (0.1, 0)  # explode 1st slice

# Plot
plt.pie(sizes, explode=explode, labels=labels,
colors=colors,
autopct='%1.1f%%', shadow=True, startangle=90)

plt.axis('equal')
plt.show()
```

Draw Count plot to visualize count of people sufer from spiders in relation with Age.

In[39]:

```
plt.figure(figsize=(15,6))
sns.countplot(x='age',data =  df, hue =
'spiders',palette='RdPu')
plt.show()
```

Plot Pie Chart of ascites Column

In[40]:

```
no =len(df[df['ascites'] == 0])
yes = len(df[df['ascites']== 1])

plt.figure(figsize=(8,6))

# df to plot
```

```
labels = 'NO','YES'
sizes = [no,yes]
colors = ['skyblue', 'yellowgreen']
explode = (0.1, 0)  # explode 1st slice

# Plot
plt.pie(sizes, explode=explode, labels=labels,
colors=colors,
autopct='%1.1f%%', shadow=True, startangle=90)

plt.axis('equal')
plt.show()
```

Draw Count plot to visualize count of people sufer from ascites in relation with Age.

In[41]:

```
plt.figure(figsize=(15,6))
sns.countplot(x='age',data =  df, hue =
'ascites',palette='RdPu')
plt.show()
```

Plot Pie Chart of varices Column

In[42]:

```
no =len(df[df['varices'] == 0])
yes = len(df[df['varices']== 1])

plt.figure(figsize=(8,6))

# df to plot
labels = 'NO','YES'
sizes = [no,yes]
colors = ['skyblue', 'yellowgreen']
explode = (0.1, 0)  # explode 1st slice

# Plot
plt.pie(sizes, explode=explode, labels=labels,
colors=colors,
autopct='%1.1f%%', shadow=True, startangle=90)

plt.axis('equal')
plt.show()
```

Draw Count plot to visualize count of people sufer from varices in relation with Age.

In[43]:

```
plt.figure(figsize=(15,6))
sns.countplot(x='age',data =  df, hue =
'varices',palette='RdPu')
plt.show()
```

Draw a Scatter Plot to visualize bilirubin test values with Age and Hue Class

In[44]:

```
plt.figure(figsize=(10,8))
sns.scatterplot(x='age',y='bilirubin',data =  df,hue =
'class')
plt.title('Bilirubin test values according to AGE')
plt.show()
```

Draw a Scatter Plot to visualize alk_phosphate test values with Age and Hue Class

In[45]:

```
plt.figure(figsize=(10,8))
sns.scatterplot(x='age',y='alk_phosphate',data =
df,hue = 'class')
plt.title('alk_phosphate test values according to
AGE')
plt.show()
```

Draw a Scatter Plot to visualize sgot test values with Age and Hue Class

In[46]:

```
plt.figure(figsize=(10,8))
sns.scatterplot(x='age',y='sgot',data =  df,hue = 'class')
plt.title('sgot test values according to AGE')
plt.show()
```

Draw a Scatter Plot to visualize albumin test values with Age and Hue Class

In[47]:

```
plt.figure(figsize=(10,8))
sns.scatterplot(x='age',y='albumin',data =  df,hue = 'class')
plt.title('albumin test values according to AGE')
plt.show()
```

Draw a Scatter Plot to visualize protime test values with Age and Hue Class

In[48]:

```
plt.figure(figsize=(10,8))
sns.scatterplot(x='age',y='protime',data =  df,hue = 'class')
plt.title('protime test values according to AGE')
plt.show()
```

Plot Pie Chart of histology Column

In[49]:

```
no =len(df[df['histology'] == 0])
yes = len(df[df['histology']== 1])

plt.figure(figsize=(8,6))

# df to plot
labels = 'NO','YES'
sizes = [no,yes]
colors = ['skyblue', 'yellowgreen']
explode = (0.1, 0)  # explode 1st slice

# Plot
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=90)

plt.axis('equal')
plt.show()
```

Draw a countplot to show count of people having positive history with Age.

In[50]:

```
plt.figure(figsize=(15,6))
sns.countplot(x='age',data =  df, hue = 'histology',palette='GnBu')
plt.show()
```

Draw a Heat Map of df

In[51]:

```
plt.figure(figsize=(15,12))
sns.heatmap(df.corr(),
cmap='coolwarm',linewidths=.1,annot = True)
plt.show()
```

We can observe in the heatmap that some of the variables show a coefficient of ~0.6 or -0.4, but most of them display a very low correlation coefficient. So we can conclude that there is no strong linear correlation between our variables.

Prediction algorithm

One of the main decisions to make when performing machine learning is choosing the appropriate algorithm that fits the current problem we are dealing with. Supervised learning refers to the task of inferring a function from a labeled training dataset. We fit the model to the labeled training set with the main goal of finding the

optimal parameters that will predict unknown labels of new examples included in the test dataset. There are two main types of supervised learning: regression, in which we want to predict a label that is a real number, and classification, in which we want to predict a categorical label. In our case, we have a labeled dataset and we want to use a classification algorithm to find the label in the categorical values: 0 and 1.
#
We can find many classification supervised learning algorithms, some simple but efficient, such as linear classifier or logistic regression, and another ones more complex but powerful such as decision trees and k-means.
In this case, we will choose Random Forest algorithm. Random forest is one of the most used machine learning algorithm due to the fact that it is very simple, flexible and easy to use but produces reliable results. So we will load the packages from scikit-learn that we need to perform Random Forest and also to evaluate afterwards the model:

In[52]:

```
# Models from Scikit-Learn
from sklearn.linear_model import
LogisticRegression
from sklearn.neighbors import
KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import
RandomForestClassifier
from sklearn.model_selection import
train_test_split, GridSearchCV

from sklearn.metrics import roc_curve, auc,
accuracy_score
from sklearn.metrics import precision_recall_curve,
confusion_matrix

#from sklearn.preprocessing import Imputer
```

In[53]:

```
x = df.iloc[:, df.columns != 'class']
y = df.iloc[:, df.columns == 'class']
```

In[54]:

```
X_train, X_test, Y_train, Y_test = train_test_split(x,
y, test_size = 0.2,
                                    random_state = 42)
```

```
# Put models in a dictionary
models = {"Logistic Regression":
LogisticRegression(),
          "KNN": KNeighborsClassifier(),
          "Random Forest": RandomForestClassifier(),
          "Decision Tree": DecisionTreeClassifier()}

# Create a function to fit and score models
def fit_and_score(models, X_train, X_test, y_train,
y_test):
    """
    Fits and evaluates given machine learning
models.
    models : a dict of differetn Scikit-Learn machine
learning models
    X_train : training data (no labels)
    X_test : testing data (no labels)
    y_train : training labels
    y_test : test labels
    """
    # Set random seed
    np.random.seed(42)
    # Make a dictionary to keep model scores
    model_scores = {}
    # Loop through models
    for name, model in models.items():
        # Fit the model to the data
        model.fit(X_train, y_train)
        # Evaluate the model and append its score to
model_scores
        model_scores[name] = model.score(X_test,
y_test)
    return model_scores
```

In[56]:

```
model_scores = fit_and_score(models=models,
                             X_train=X_train,
                             X_test=X_test,
                             y_train=Y_train,
                             y_test=Y_test)
model_scores
```

```
from warnings import simplefilter
from sklearn.exceptions import
ConvergenceWarning
simplefilter("ignore",
category=ConvergenceWarning)
```

In[58]:

```
param_grid = [
    {'penalty' : ['l1', 'l2', 'elasticnet'],
    'C' : np.logspace(-4, 4, 20),
    'solver' : ['lbfgs','newton-cg','liblinear'],
    'max_iter' : [500, 1000, 1500]
    }
]
```

In[59]:

```
parameters_optimize = {
                'max_features': ['auto', 'sqrt', 'log2',
None],
                'max_depth': [2,3, 4],
                'criterion': ['gini', 'entropy'],
                'bootstrap': [True, False],
                'n_estimators': [2, 5, 10, 15, 20]
                }
```

In[ ]:

In[60]:

```
lr = LogisticRegression(random_state=10)
model = lr.fit(X_train, Y_train)
```

In[61]:

```
gsc = GridSearchCV(lr, param_grid = param_grid,
cv = 10,verbose=True, n_jobs=-1)
tuned_model = gsc.fit(X_train,Y_train)
```

In[62]:

```
tuned_model.best_estimator_
```

In[63]:

```
print (f'Accuracy - :
{tuned_model.score(X_test,Y_test):.3f}')
```

In[64]:

```
y_pred = tuned_model.predict(X_test)
print(accuracy_score(Y_test,y_pred))
```

There is no improvement in Logistic regression and we are choosing random forest to improve model performace

In[65]:

```
random_forest_hyp = RandomForestClassifier()
```

```
random_forest_search =
GridSearchCV(random_forest_hyp,
                           cv = 20,
                           param_grid =
parameters_optimize,
                           n_jobs = 3)
random_forest_search.fit(X_train, Y_train)
print('The best parameteres after GridSearchCV',
random_forest_search.best_params_)


# In[66]:


random_forest_hyp.set_params(bootstrap = True,
                    criterion = 'entropy',
                    max_depth = 4,
                    max_features = 'sqrt',
                    n_estimators = 15);


# In[67]:


random_forest_hyp.fit(X_train, Y_train);
y_predicted_grid =
random_forest_hyp.predict(X_test)
accuracy_grid = accuracy_score(Y_test,
y_predicted_grid)*100
print(round(accuracy_grid, 2), '%')


# In[68]:



plt.figure()
random_confusion = confusion_matrix(Y_test,
y_predicted_grid)
sns.heatmap(random_confusion, annot = True);


# In[69]:


fpr, tpr, _= roc_curve(Y_test, y_predicted_grid)
auc_random_grid = auc(fpr, tpr)
print(auc_random_grid)
```

```
# In[70]:


plt.figure()
plt.plot(fpr, tpr, color ='Green', linewidth = 1)
plt.title('ROC curve for Random Forest')
plt.plot([0,1], [0,1], 'k--', lw = 1)
plt.plot([0,0], [1,0], 'k--', lw = 1, color = 'black')
plt.plot([1,0], [1,1], 'k--', lw = 1, color = 'black')
plt.xlabel('False Positive rate')
plt.ylabel('True Positive rate');
```

Heapatits

ORIGINALITY REPORT

46%	33%	18%	29%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Coventry University Student Paper	11%
2	Submitted to Nottingham Trent University Student Paper	6%
3	openaccess.altinbas.edu.tr Internet Source	6%
4	repub.eur.nl Internet Source	3%
5	Submitted to University of Wales Institute, Cardiff Student Paper	2%
6	Submitted to University of Edinburgh Student Paper	1%
7	www.ncbi.nlm.nih.gov Internet Source	1%
8	www.hindawi.com Internet Source	1%
9	Submitted to Rutgers University, New Brunswick	1%

Student Paper

10	Neha Sharma, Santanu Ghosh, Monodeep Saha. "Open Data for Sustainable Community", Springer Science and Business Media LLC, 2021 Publication	1 %
11	Submitted to University College London Student Paper	1 %
12	www.analyticsvidhya.com Internet Source	1 %
13	sofiadutta.github.io Internet Source	1 %
14	Submitted to Higher Education Commission Pakistan Student Paper	1 %
15	towardsdatascience.com Internet Source	<1 %
16	Submitted to Universidad del País Vasco Student Paper	<1 %
17	jovian.ai Internet Source	<1 %
18	Submitted to University of Florida Student Paper	<1 %
19	www.inertia7.com Internet Source	<1 %

20	oro.open.ac.uk Internet Source	<1 %
21	Submitted to Indiana University Student Paper	<1 %
22	hcheruiy.github.io Internet Source	<1 %
23	www.fjs.fudutsinma.edu.ng Internet Source	<1 %
24	www.medrxiv.org Internet Source	<1 %
25	github.com Internet Source	<1 %
26	Submitted to RMIT University Student Paper	<1 %
27	scipy-lectures.org Internet Source	<1 %
28	blog.ursb.me Internet Source	<1 %
29	Reza Safdari, Amir Deghatipour, Marsa Gholamzadeh, Keivan Maghooli. "Applying data mining techniques to classify patients with suspected hepatitis C virus infection", Intelligent Medicine, 2022 Publication	<1 %

30	František Murgaš, Michal Klobučník. "Quality of life in the city, quality of urban life or well-being in the city: Conceptualization and case study", Ekológia (Bratislava), 2018 Publication	<1 %
31	www.programcreek.com Internet Source	<1 %
32	Nikhil Ketkar, Jojo Moolayil. "Deep Learning with Python", Springer Science and Business Media LLC, 2021 Publication	<1 %
33	Submitted to The University of Manchester Student Paper	<1 %
34	jes.utm.md Internet Source	<1 %
35	iiste.org Internet Source	<1 %
36	Submitted to University of Southern California Student Paper	<1 %
37	Submitted to Tilburg University Student Paper	<1 %
38	datascience.uconn.edu Internet Source	<1 %
39	docplayer.net Internet Source	<1 %

40	export.arxiv.org Internet Source	<1 %
41	www.termedia.pl Internet Source	<1 %
42	Submitted to Monash University Student Paper	<1 %
43	attachments.waset.org Internet Source	<1 %
44	brainlit.neurodata.io Internet Source	<1 %
45	dokumen.pub Internet Source	<1 %
46	"Artificial Intelligence and Machine Learning Methods in COVID-19 and Related Health Diseases", Springer Science and Business Media LLC, 2022 Publication	<1 %
47	"Intelligent Systems", Springer Science and Business Media LLC, 2021 Publication	<1 %
48	Poornachandra Sarang. "Artificial Neural Networks with TensorFlow 2", Springer Science and Business Media LLC, 2021 Publication	<1 %

49	Usman Malik, Mukesh Barange, Julien Saunier, Alexandre Pauchet. "A novel focus encoding scheme for addressee detection in multiparty interaction using machine learning algorithms", Journal on Multimodal User Interfaces, 2021 Publication	<1 %
50	gitlab2.eeecs.qub.ac.uk Internet Source	<1 %
51	machinelearningmastery.com Internet Source	<1 %
52	pythonexamples.org Internet Source	<1 %
53	Submitted to University of Lancaster Student Paper	<1 %
54	Submitted to York St John University Student Paper	<1 %
55	cse.anits.edu.in Internet Source	<1 %
56	ebin.pub Internet Source	<1 %
57	fjs.fudutsinma.edu.ng Internet Source	<1 %
58	hrushi11.github.io Internet Source	<1 %

59	Xiaolu Tian, Yutian Chong, Yutao Huang, Pi Guo, Mengjie Li, Wangjian Zhang, Zhicheng Du, Xiangyong Li, Yuantao Hao. "Using Machine Learning Algorithms to Predict Hepatitis B Surface Antigen Seroclearance", Computational and Mathematical Methods in Medicine, 2019 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		

Heapatits