Revolutionizing Liver Care : Predicting Liver Cirrhosis using Advanced Machine Learning Techniques

Revolutionizing Liver Care : Predicting Liver Cirrhosis using Advanced Machine Learning Techniques

Liver cirrhosis is a chronic and progressive liver disease characterized by the irreversible scarring of liver tissue, which can lead to severe complications and liver failure if left untreated. This project aims to develop a predictive model for the early detection and prognosis of liver cirrhosis using machine learning techniques.The developed predictive model holds promise for early detection and prognosis of liver cirrhosis, enabling healthcare professionals to initiate timely interventions and personalized treatment strategies. By accurately identifying individuals at risk of cirrhosis, this research contributes to improved patient outcomes and the optimization of healthcare resources. Moreover, it highlights the potential of machine learning in the field of hepatology, showcasing its ability to leverage clinical data for predictive modeling and decision support.

Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analysed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

- Define Problem / Problem Understanding
    1. Specify the business problem

? Business requirements

? Literature Survey

? Social or Business Impact.

? Data Collection & Preparation

? Collect the dataset

? Data Preparation

- Exploratory Data Analysis
    1. Descriptive statistical

? Visual Analysis

- Model Building
    1. Training the model in multiple algorithms

? Testing the model

- Performance Testing & Hyperparameter Tuning
    1. Testing model with multiple evaluation metrics

? Comparing model accuracy before & after applying hyperparameter tuning

- Model Deployment
    1. Save the best model

? Integrate with Web Framework

- Project Demonstration & Documentation
    1. Record explanation Video for project end to end solution

? Project Documentation-Step by step project development procedure

Prior Knowledge:

You must have prior knowledge of following topics to complete this project.

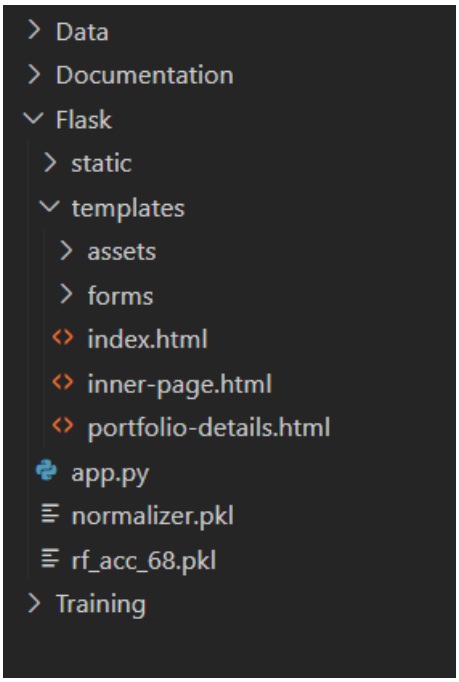
ML Concepts

- Supervised learning: https://www.javatpoint.com/supervised-machine-learning
- Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning
- Decision tree: https://www.javatpoint.com/machine-learning-decision-tree-classificationalgorithm
- Random forest: https://www.javatpoint.com/machine-learning-random-forest-algorithm

- KNN: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
- Xgboost: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-tounderstand-the-math-behind-xgboost/
- Evaluation metrics: https://www.analyticsvidhya.com/blog/2019/08/11-important-modelevaluation-error-metrics/
- Flask Basics : https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Structure:

Create the Project folder which contains files as shown below



- We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
- Static folder and assets folder contain the CSS and JavaScript files along with images.
- rf_acc_68.pkl and normalizer.pkl are our saved models. Further we will use these models for flask integration.
- Training folder contains a model training file that are jupyter notebooks.
-

- # Milestone 1: Define Problem / Problem Understanding
- saved
- # Activity 1: Specify the Diabetes Results
- Refer Project Description
- # Activity 2
- Activity 2:
- Here are some Diabetes Results for an Liver Cirrhosis predictor using machine learning:
-
- Introduction
- Considerations for deploying the predictive model into a real-world clinical setting.
- Integration of the model with existing healthcare systems for seamless data exchange and decision support.
- Limitations and Future Directions
- Discussion of Overview of liver cirrhosis and its impact on public health.
- The importance of early detection and prediction for effective treatment.
- Introduction to machine learning and its potential in healthcare.
- Dataset Acquisition and Preprocessing
- Selection of a suitable dataset containing liver-related features and cirrhosis labels.
- Data preprocessing steps, including cleaning, handling missing values, and feature selection.
- Splitting the dataset into training and testing sets.
- Exploratory Data Analysis
- Statistical analysis of the dataset to gain insights into the distribution and correlation of features.
- Visualization techniques to understand the patterns and trends in the data.
- Identification of potential risk factors and predictors for liver cirrhosis.
- Feature Engineering
- Transformation and creation of new features to enhance the predictive power of the model.
- Techniques such as scaling, normalization, and feature extraction.
- Handling imbalanced data issues if applicable.
- Machine Learning Models
- Introduction to various advanced machine learning algorithms suitable for liver cirrhosis prediction.
- Implementation of algorithms such as logistic regression, decision trees, random forests, support vector machines (SVM), and gradient boosting.
- Model training, hyperparameter tuning, and evaluation metrics selection.
- Model Evaluation
- Evaluation of the trained models using appropriate metrics (e.g., accuracy, precision, recall, F1-score, ROC curves).
- Cross-validation techniques to ensure model robustness and generalizability.

- Comparison of different models and selection of the best performing model.
- Interpretability and Explainability
- Techniques for interpreting machine learning models and understanding the underlying factors contributing to predictions.
- Feature importance analysis to identify the most influential features for liver cirrhosis prediction.
- Deployment and Integration
- limitations and challenges associated with liver cirrhosis prediction using machine learning.
- Potential solutions and future research directions to address these limitations.
- Conclusion
- Summary of the study's findings and the potential impact of using advanced machine learning techniques for liver cirrhosis prediction.
- Emphasis on the importance of early detection and personalized interventions for improved liver care.
-
- By following this documentation, healthcare professionals, researchers, and data scientists can gain insights into predicting liver cirrhosis using advanced machine learning techniques, potentially revolutionizing liver care and improving patient outcomes.

- # Activity 3: Literature Survey (Student Will Write)

- A literature survey for a liver cirrhosis Prediction project would involve researching and reviewing existing studies, articles, and other publications on the topic of liver cirrhosis Prediction. The survey would aim to gather information on current systems, their strengths and weaknesses, and any gaps in knowledge that the project could address. The literature survey would also look at the methods and techniques used in previous projects, and any relevant data or findings that could inform the design and implementation of the current project.

- # Milestone 2: Data Collection & Preparation

- ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

Activity 1: Collect the dataset.

Activity 1: Collect the dataset.

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link: https://www.kaggle.com/datasets/bhavanipriya222/liver-cirrhosis-prediction

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques.

Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Activity 1.1: Importing the libraries

Import the necessary libraries as shown in the image.

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import pickle as pkl
import numpy as np
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV, RidgeClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from xgboost import XGBClassifier
from sklearn.preprocessing import Normalizer
from sklearn.metrics import accuracy_score, f1_score, recall_score, precision_score, confusion_matrix
```

Activity 1.2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of the csv file.

```
In [2]: #Reading CSV file
        dataset = pd.read_excel('HealthCareData.xlsx')
        dataset.head()
```

Out[2]:

	S.NO	Age	Gender	Place(location where the patient lives)	Duration of alcohol consumption(years)	Quantity of alcohol consumption (quarters/day)	Type of alcohol consumed	Hepatitis B infection	Hepatitis C infection	Diabetes Result	...	Indirect (mg/dl)	Total Protein (g/dl)	Albumin (g/dl)	Globulin (g/dl)
0	1	55	male	rural	12	2	branded liquor	negative	negative	YES	...	3.0	6.0	3.0	4.0
1	2	55	male	rural	12	2	branded liquor	negative	negative	YES	...	3.0	6.0	3.0	4.0
2	3	55	male	rural	12	2	branded liquor	negative	negative	YES	...	3.0	6.0	3.0	4.0
3	4	55	male	rural	12	2	branded liquor	negative	negative	NO	...	3.0	6.0	3.0	4.0
4	5	55	female	rural	12	2	branded liquor	negative	negative	YES	...	3.0	6.0	3.0	4.0

Activity 2: Data Preparation

Activity 2: Data Preparation

As we have understood how the data is, let's pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling categorical data
- Handling Outliers

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Activity 2.1: Handling missing values

- Let's find the shape of our dataset first. To find the shape of our data, the df.shape method is used. To find the data type, df.info() function is used.

- For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.

```
df.shape
```

```
(950, 42)
```

```
df.isnull().any()
```

```
df.isnull().sum()
```

```
S.NO                                          0
Age                                           0
Gender                                        0
Place(location where the patient lives)     134
Duration of alcohol consumption(years)        0
Quantity of alcohol consumption (quarters/day)  0
Type of alcohol consumed                      0
Hepatitis B infection                         0
Hepatitis C infection                         0
Diabetes Result                               0
Blood pressure (mmhg)                         0
Obesity                                       0
Family history of cirrhosis/ hereditary       0
TCH                                         359
TG                                          359
LDL                                         359
HDL                                         368
Hemoglobin  (g/dl)                            0
PCV  (%)                                     30
```

```
df.isnull().sum()
```

```
S.NO                                          0
Age                                           0
Gender                                        0
Place(location where the patient lives)       0
Duration of alcohol consumption(years)        0
Quantity of alcohol consumption (quarters/day)  0
Type of alcohol consumed                      0
Hepatitis B infection                         0
Hepatitis C infection                         0
Diabetes Result                               0
Blood pressure (mmhg)                         0
Obesity                                       0
Family history of cirrhosis/ hereditary       0
TCH                                           0
TG                                            0
LDL                                           0
HDL                                           0
Hemoglobin  (g/dl)                            0
PCV  (%)                                      0
```

Activity 2.2: Handling Categorical Values

As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.

To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project we are using manual encoding with the help of list comprehension.

```
categorical_features = df.select_dtypes(include=[np.object])
categorical_features.columns
```

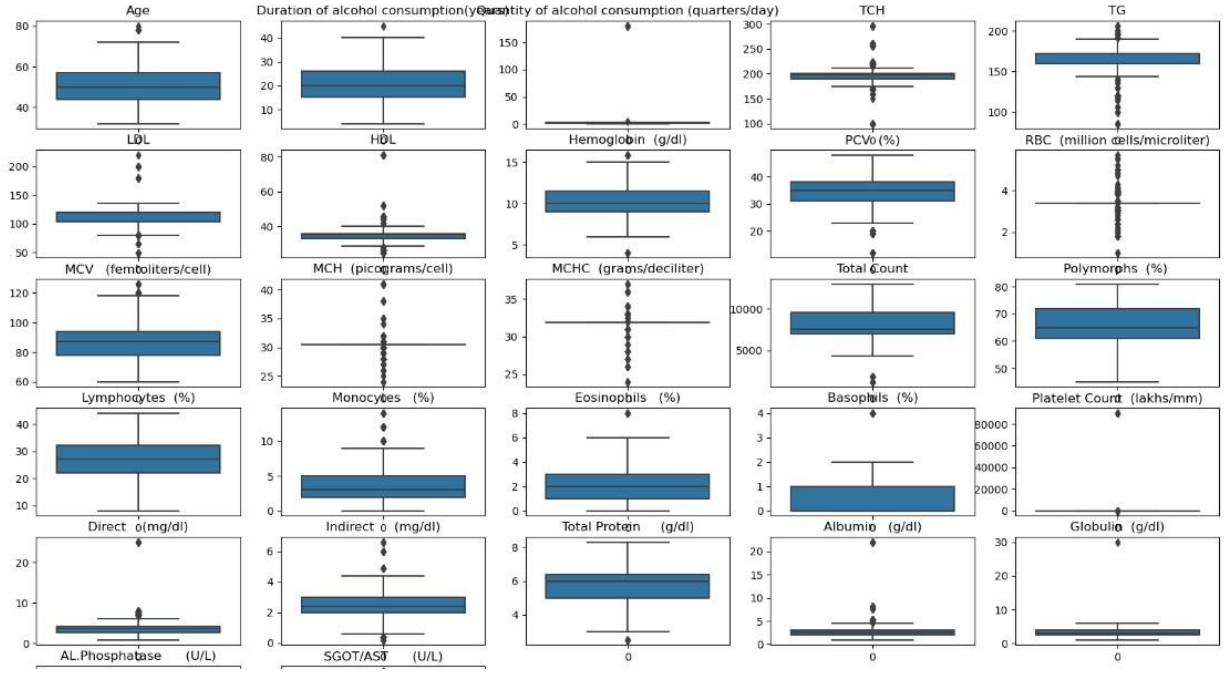
```
Index(['Gender', 'Place(location where the patient lives)',
       'Type of alcohol consumed', 'Hepatitis B infection',
       'Hepatitis C infection', 'Diabetes Result', 'Blood pressure (mmhg)',
       'Obesity', 'Family history of cirrhosis/ hereditary', 'TG', 'LDL',
       'Total Bilirubin    (mg/dl)', 'A/G Ratio',
       'USG Abdomen (diffuse liver or  not)', 'Outcome'],
      dtype='object')
```

Activity 2.3: Handling Outliers in Data

With the help of boxplot, outliers are visualized. And here we are going to find upper bound and lower bound of numerical features with some mathematical formula.

- From the below diagram, we could visualize that Discount_offered, Prior_purchases features have outliers. Boxplot from matplotlib library is used here.

```
c=0
plt.figure(figsize=(20,15))
for i in df.columns:
    if type(df[i][0])!=str:
        plt.subplot(7,5,c+1)
        sns.boxplot(df[i])
        plt.title(i)
        c+=1
plt.show()
```



- To find upper bound we have to multiply IQR (Interquartile range) with 1.5 and add it with 3rd quantile. To find lower bound instead of adding, subtract it with 1st quantile. Take image attached below as your reference.

```
q1 = df['Eosinophils   (%)'].quantile(0.25)
q3 = df['Eosinophils   (%)'].quantile(0.75)
iqr = q3 - q1
q1,q3,iqr
upper_limit = q3 + (1.5*iqr)
lower_limit = q1 - (1.5*iqr)
lower_limit , upper_limit
df['Eosinophils   (%)'] = np.where(df['Eosinophils   (%)'] > upper_limit , upper_limit ,
                        np.where(df['Eosinophils   (%)'] < lower_limit , lower_limit , df['Eosinophils   (%)']))
```

```
sns.boxplot(df['Eosinophils   (%)'])
```

```
sns.boxplot(df['Basophils  (%)'])
```

```
q1 = df['Basophils  (%)'].quantile(0.25)
q3 = df['Basophils  (%)'].quantile(0.75)
iqr = q3 - q1
q1,q3,iqr
upper_limit = q3 + (1.5*iqr)
lower_limit = q1 - (1.5*iqr)
lower_limit , upper_limit
df['Basophils  (%)'] = np.where(df['Basophils  (%)'] > upper_limit , upper_limit ,
                        np.where(df['Basophils  (%)'] < lower_limit , lower_limit , df['Basophils  (%)']))
```

```
sns.boxplot(df['Basophils  (%)'])
```

```
sns.boxplot(df['Platelet Count  (lakhs/mm)'])
```

- To handle the outliers transformation technique is used. Here L1 transformation is used.

- Data splitting

The data was split into train and test variables as shown below using the train_test_split() method of scikitlearn module with a split_size of 0.20 and a random_state = 42.

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

```
x_train
```

```
...
```

```
x_test
```

```
...
```

```
y_train
```

```
...
```

```
y_test
```

```
...
```

- Normalization

The data will be normalized using L1 regularisation that will be applied on x_train and x_test variables separately.

Milestone 3: Exploratory Data Analysis

Data analysis

Activity 1: Descriptive statistical

Activity 1: Descriptive statistical

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

```
df.describe()
```

	S.NO	Age	Duration of alcohol consumption(years)	Quantity of alcohol consumption (quarters/day)	TCH	HDL	Hemoglobin (g/dl)	PCV (%)	RBC (million cells/microliter)	MCV (femtoliters/cell)	...	Baso
count	950.000000	950.000000	950.000000	950.000000	591.000000	582.000000	950.000000	920.000000	398.000000	941.000000	...	901.0(
mean	475.500000	50.632632	20.606316	5.158947	197.544839	35.486254	10.263979	33.810000	3.390704	87.651435	...	0.4
std	274.385677	8.808272	7.980664	22.908785	26.694968	7.982057	1.942300	5.751592	0.937089	13.844181	...	0.7
min	1.000000	32.000000	4.000000	1.000000	100.000000	25.000000	4.000000	12.000000	1.000000	60.000000	...	0.0(
25%	238.250000	44.000000	15.000000	2.000000	180.000000	30.000000	9.000000	30.000000	2.825000	78.000000	...	0.0(
50%	475.500000	50.000000	20.000000	2.000000	194.000000	35.000000	10.000000	35.000000	3.500000	87.000000	...	0.0(
75%	712.750000	57.000000	26.000000	3.000000	210.000000	38.000000	11.500000	38.000000	4.000000	94.000000	...	1.0(
max	950.000000	80.000000	45.000000	180.000000	296.000000	81.000000	15.900000	48.000000	5.700000	126.000000	...	4.0(

8 rows × 27 columns

Activity 2: Visual analysis

Activity 2: Visual analysis

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

Activity 2.1: Univariate analysis

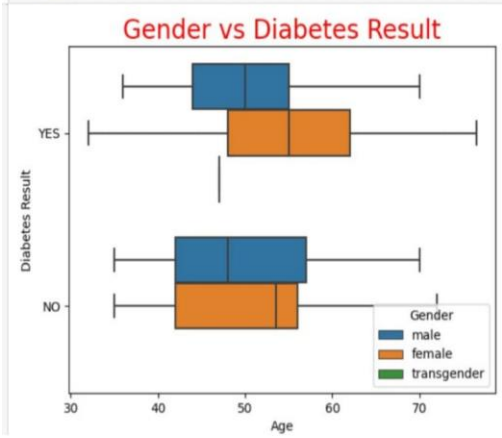
In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as histplot and countplot.

Seaborn package provides a wonderful function histplot. With the help of histplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.

From the plot we came to know

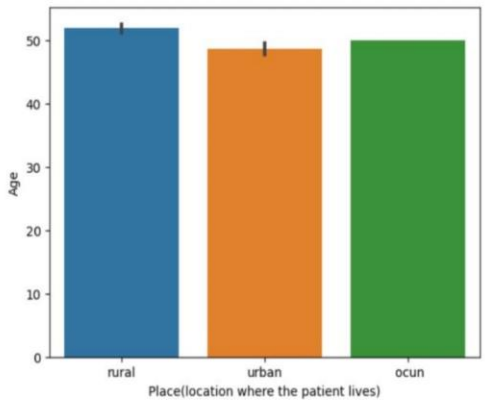
In our dataset we have some categorical features. With the countplot function, we are going to count the unique category in those features. We have created a dummy data frame with categorical features. With for loop and subplot we have plotted this below graph.

```
sns.boxplot(x='Age',y='Diabetes Result',data=df,hue='Gender')
plt.title('Gender vs Diabetes Result',color='red',size=20)
plt.show()
```



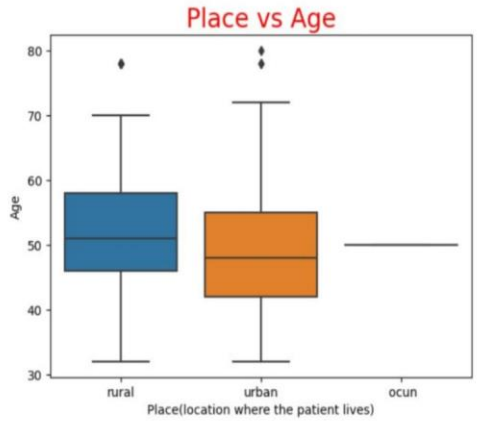

```
sns.barplot(x=df['Place(location where the patient lives)'],y=df['Age'])
```

```
<Axes: xlabel='Place(location where the patient lives)', ylabel='Age'>
```

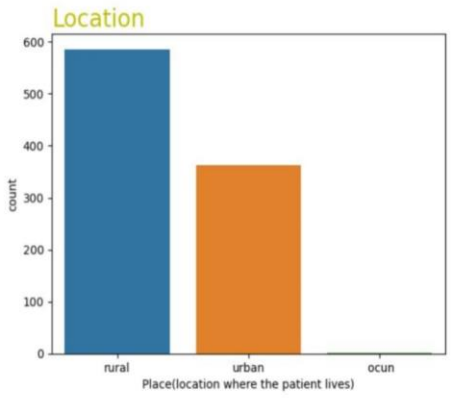



```
sns.boxplot(x='Place(location where the patient lives)',y='Age',data=df)
plt.title('Place vs Age',color='red',size=20)
```

```
Text(0.5, 1.0, 'Place vs Age')
```

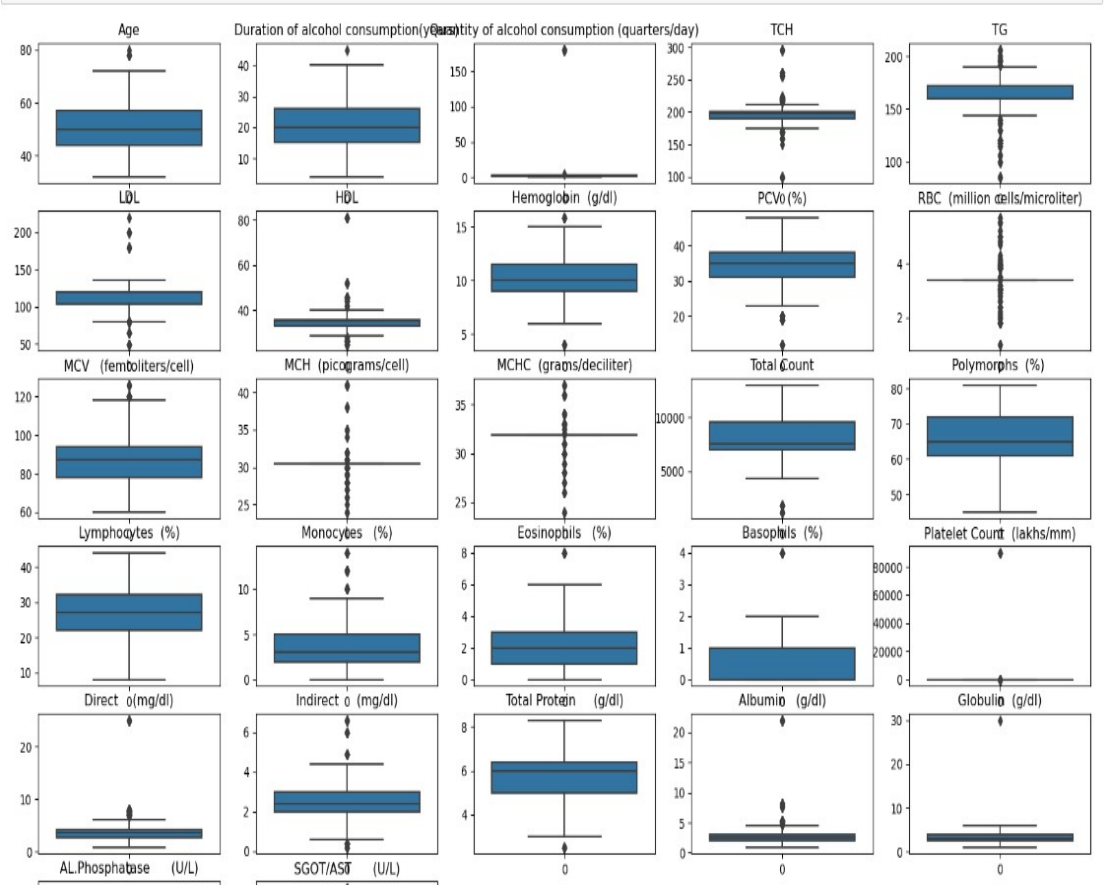



```
sns.countplot(data=df,x='Place(location where the patient lives)')
plt.title("Location",color='y',size=20,loc='left')
plt.show()
```



Activity 2.2: Bivariate analysis

To find the relation between two features we use bivariate analysis. Here we are visualizing of data

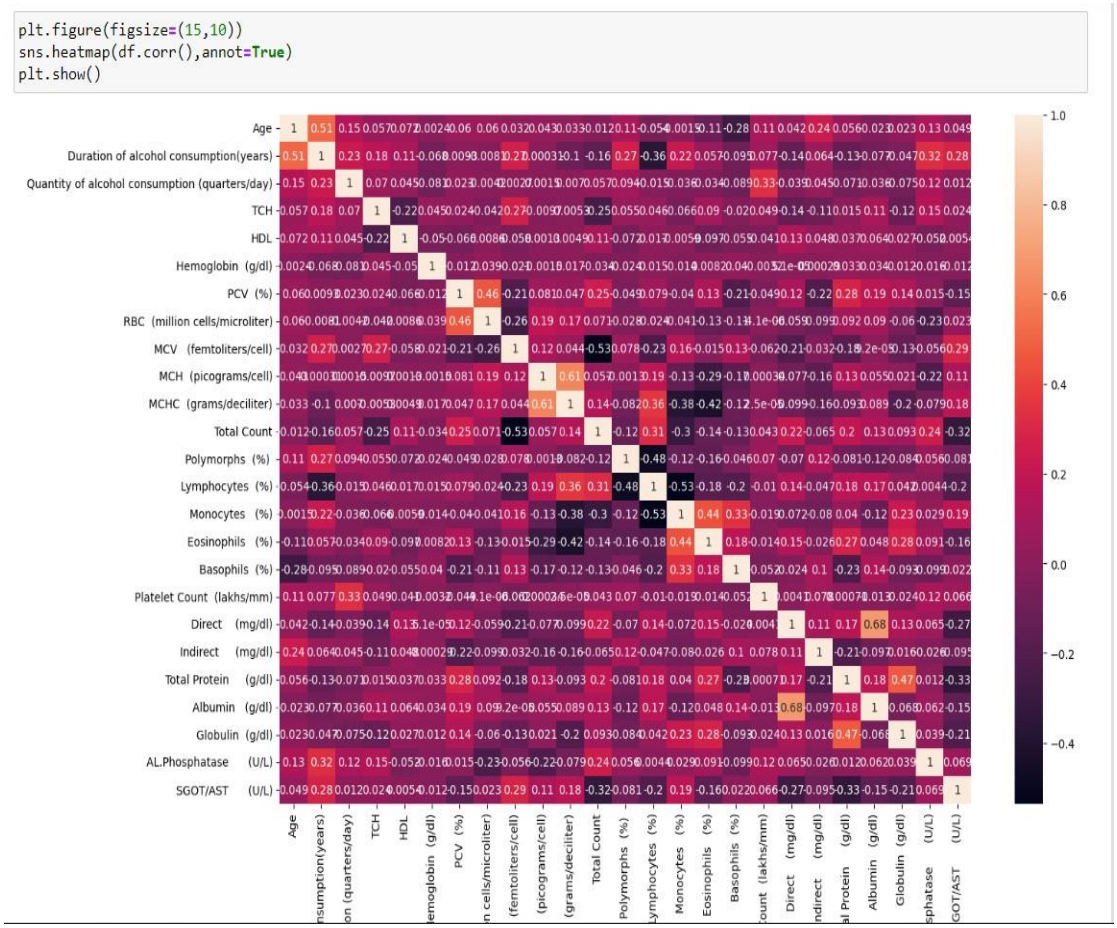


Outliers were found for 2 features as visualized above using box plots. To be specific using IQR (inter quartile range) it was observed that,

Activity 2.3: Multivariate analysis

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used heatmap from seaborn package.

• From the below image, we came to a conclusion that the product discount is the feature that most highly correlates to if a product is delivered on time and Number of calls and product cost are also highly correlated among other variables.



Splitting data into train and test

Now let's split the Dataset into train and test sets. First split the dataset into x and y and then split the data set.

Here x and y variables are created. On x variable, data is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_staten shuffle.

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

```
x_train
```

```
...
```

```
x_test
```

```
...
```

```
y_train
```

```
...
```

```
y_test
```

```
...
```

Milestone 4: Model Building

Model building

Activity 1: Training the model in multiple algorithms

Activity 1: Training the model in multiple algorithms

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying seven classification algorithms. The best model is saved based on its performance.

Activity 1.1: Writing function to train the models

A function named models_eval_mm is created and train, test data are passed as parameters. In the function, logistic regression, logistic regression cv, XGBclassifier, RidgeClassifier, KNN classifier, Random forest classifier and are initialised and training data is passed to the model with.fit() function. Test data is predicted with predict() function and saved in a new variable. For evaluating the model, train and test scores are used.

Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(x_train,y_train)
```

```
▾ GaussianNB
GaussianNB()
```

```
x_train
```

```
y_train
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(x_train,y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
x_train
```

```
y_train
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
log = LogisticRegression()
logistic = log.fit(x_train,y_train)
```

```
x_train
```

```
y_train
```

KNN

```
from sklearn.neighbors import KNeighborsClassifier
```

```
knn = KNeighborsClassifier()
knn.fit(x_train,y_train)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

```
print ("x_Train",x_train)
print("y_Train",y_train)
```

Activity 1.2: Calling the function

The function is called by passing the train, test variables. The models are returned and stored in variables as shown below. Clearly, we can see that the models are not performing well on the data. So, we'll optimise the hyperparameters of models using GridsearchCV.

Milestone 5: Performance Testing & Hyperparameter Tuning

Performance Testing