CSE422 Project Report

Project Title : Predicting The Impact of COVID-19 on Patients with Primary Liver Cancer

Group No. : 8002

Prepared By:

Name	Student ID
Noshin Tabassum	20101347
Kaji Sajjad Hossain	20101321
MD Zubairul Islam	20101322

Submitted To:

Mostafa Kamal Sagor (MKS)
Mehran Hossain (MRH)

Table Of Contents

Introduction

Our project is based on the analysis of the prediction of the full impact of the COVID-19 on primary liver cancer patients. Among the diverse population of patients susceptible to the virus's effects, individuals with pre-existing health conditions, particularly those afflicted by primary liver cancer, have emerged as a group of heightened concern. The intersection of COVID-19 and primary liver cancer presents a complex medical scenario that demands comprehensive analysis and proactive strategies.

The primary aim of our project is to develop a predictive framework that assesses the potential impact of COVID-19 on patients already diagnosed with primary liver cancer. By leveraging a combination of clinical data, medical history, demographic information, and emerging research on COVID-19's effects, our project seeks to create a model that can offer insights into how COVID-19 might interact with the specific vulnerabilities and challenges faced by individuals with primary liver cancer.

Our project is aiming to solve problems in identifying primary liver cancer patients who are at a higher risk of severe complications if they contract COVID-19. By predicting the potential impact of COVID-19 on primary liver cancer patients, clinicians can make more informed decisions regarding treatment plans. This includes considering the risks and benefits of various interventions, therapies, and medications, while also taking into account the potential interactions between the virus and ongoing cancer treatments. The project's predictions can be used to have informed discussions with patients and their families. Clear communication about potential risks and outcomes can empower patients to make decisions aligned with their preferences and values, considering both their liver cancer and COVID-19 prognosis. The predictive model developed could contribute to scientific research by shedding light on the intricate interactions between primary liver cancer and COVID-19. It could help researchers better understand the factors that contribute to worse outcomes and potentially guide the development of targeted interventions.

The motivation behind this project stems from the urgent need to bridge the knowledge gap surrounding the potential impact of COVID-19 on patients battling primary liver cancer. With limited empirical evidence available, healthcare professionals and patients are left grappling with uncertainty. Our project seeks to provide a proactive solution that empowers clinicians with insights into the potential disease course for these patients if they contract COVID-19. By doing so we aim to enhance patient care.

Dataset Description

Source of our Dataset

Link:
https://www.kaggle.com/datasets/fedesoriano/covid19-effect-on-liver-cancer-prediction-dataset

Reference: Kaggle - A data science competition platform and online community of data scientists and machine learning practitioners.

Description of our Dataset

Our dataset has a total of 27 columns and 2008 rows. The 27 columns represent the features of the project.

The features are Cancer, Year, Bleed, Mode_Presentation, Age, Gender, Etiology, Cirrhosis, Size, HCC_TNM_Stage, HCC_BCLC_Stage, ICC_TNM_Stage, Treatment_grps, Survival_fromMDM, Alive_Dead, Type_of_incidental_finding, Surveillance_programme, Surveillance_effectiveness, Mode_of_surveillance_detection, Time_diagnosis_1st_Tx, Date_incident_surveillance_scan, PS, Time_MDM_1st_treatment, Time_decisiontotreat_1st_treatment, Prev_known_cirrhosis, Months_from_last_surveillance.

Our analysis of the dataset is of Classification Type problem. Our targeted feature has data of categorical type. So we will apply the models of classification type after preprocessing our dataset in order to derive that targeted feature as our result.

There are a total of 2007 data points.

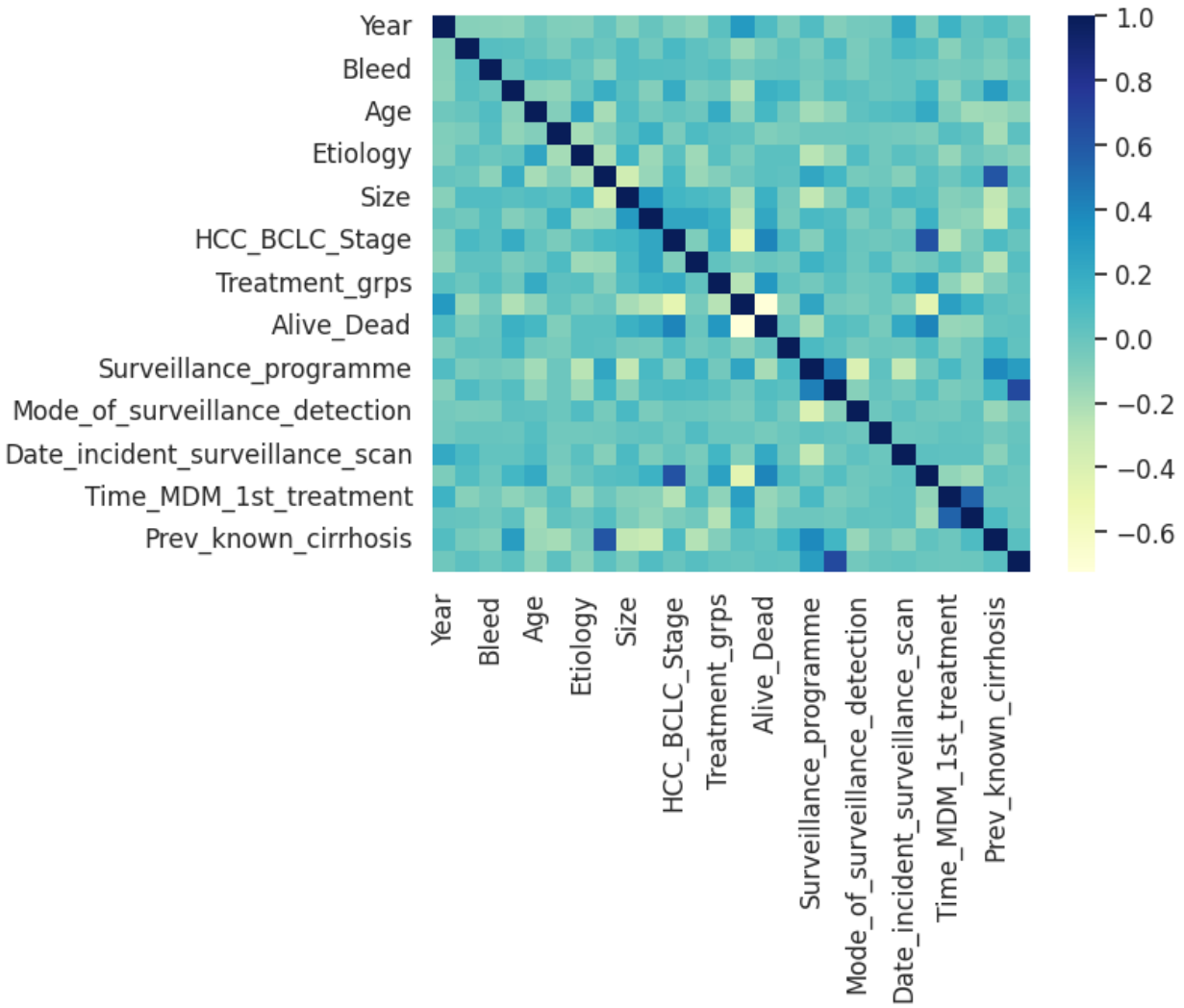
Our dataset contains both categorical and continuous numerical features.

Most of the features of our dataset are categorical. The categorical features are Cancer, Year, Bleed, Mode_Presentation, Gender, Etiology, Cirrhosis, HCC_TNM_Stage, HCC_BCLC_Stage, ICC_TNM_Stage, Treatment_grps, Alive_Dead, Type_of_incidental_finding, Surveillance_programme, Surveillance_effectiveness, Mode_of_surveillance_detection, Date_incident_surveillance_scan, PS, Prev_known_cirrhosis. The remaining features namely Age, Size, Survival_fromMDM, Time_diagnosis_1st_Tx, Time_MDM_1st_treatment and Months_from_last_surveillance are the continuously numerical features.

Correlation of all the features:

In our project, we have shown the correlation of all the features of our dataset by applying heatmap using the seaborn library. We have shown the correlation in two ways: based on color density amd based on the correlation values ranging from -1 to +1.

Based on color density the correlation of all the features looks like:



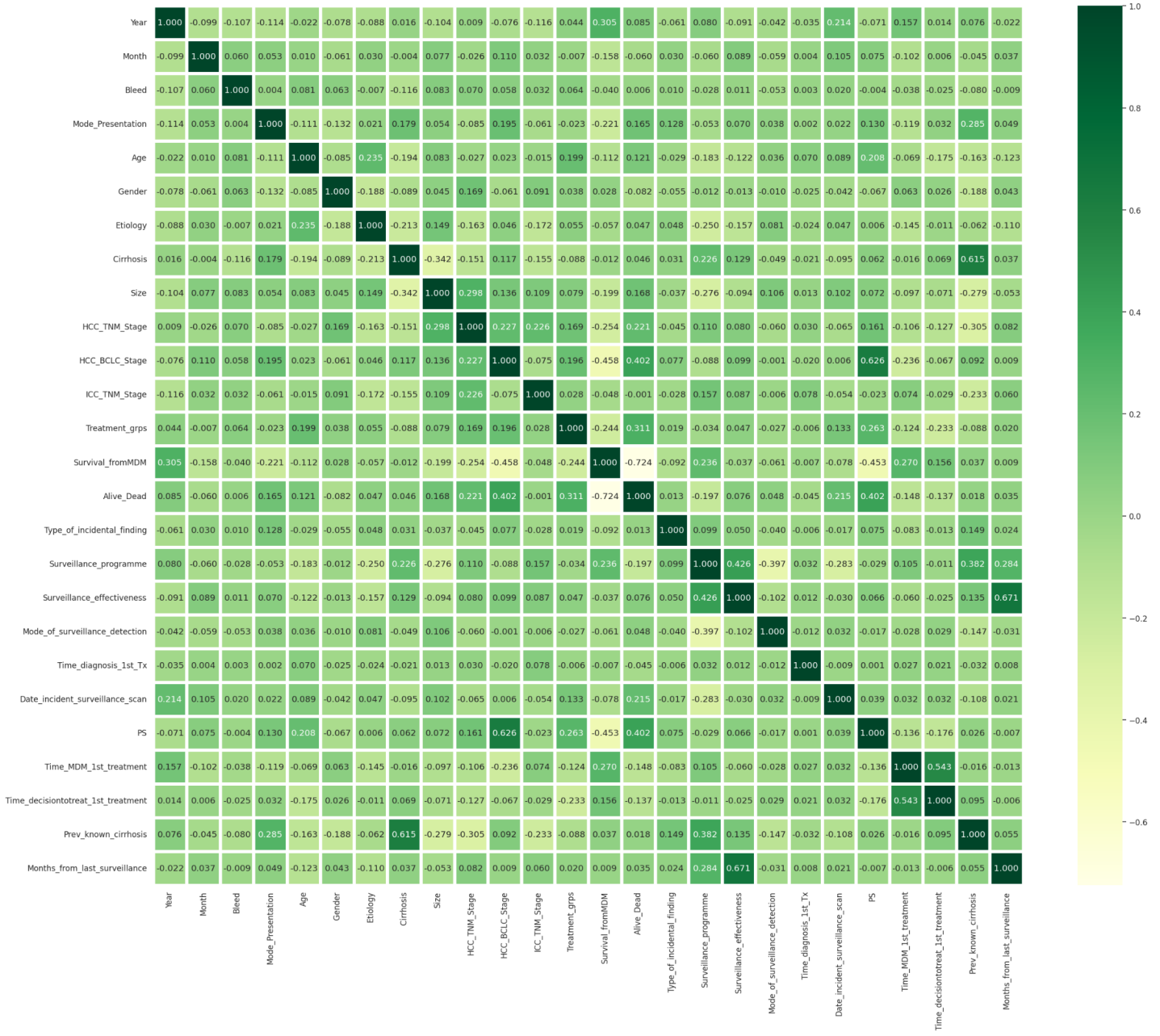
The above heatmap shows the correlation of all the features based on the color density. The more the color density (the deeper the color is) along the rows and columns , the more stronger the correlation is. It is seen that the correlation value increases close to +1 when the color gets much deeper (when the density of the color increases).

Based on the correlation values ranging from -1 to +1 the correlation of all the features looks like:

The Source Code:

```
import seaborn as sns
sns.set(font_scale=1.1)
corr_train=dataset.corr()
plt.figure(figsize=(30,25))
sns.heatmap(corr_train,annot=True,fmt='.3f',cmap='YlGn',linewidth=5,cbar=T
rue)
plt.show()
```

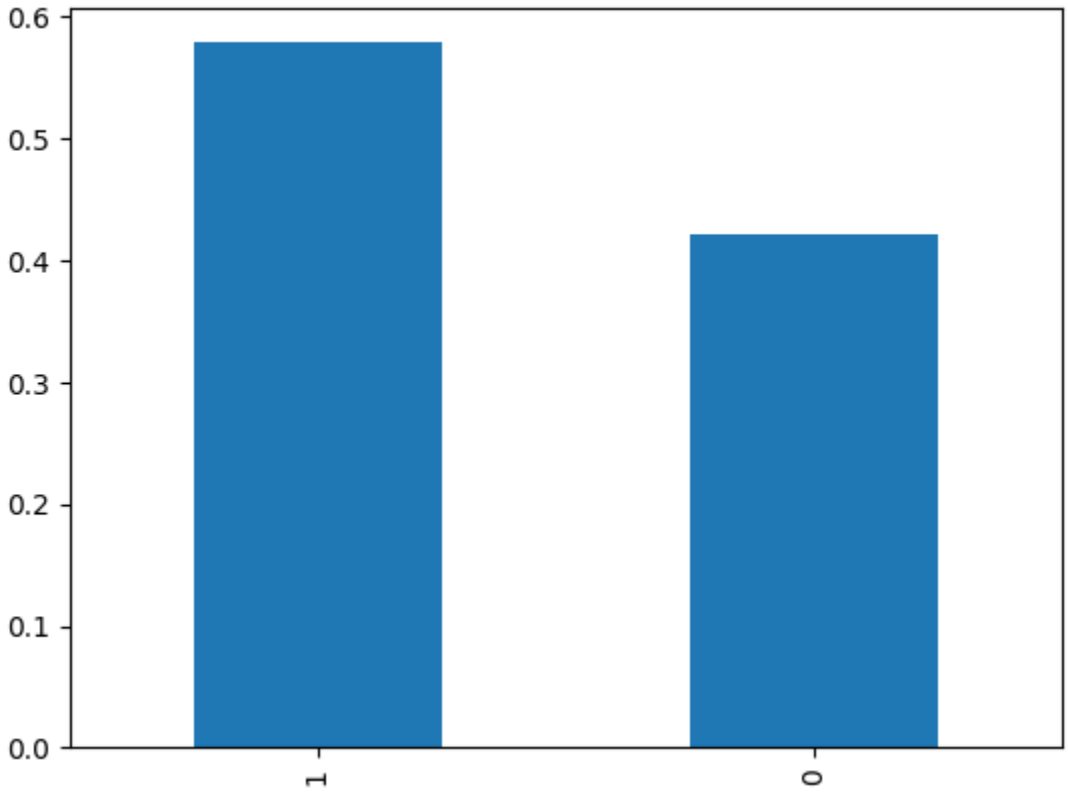
The Output:



The above correlation is derived by applying heatmap from the seaborn library. The above correlation is determined based on the values ranging from -1 to 1. We consider the absolute values of the negative values to determine the correlation of the features. The more the correlation value is close to positive 1, the stronger the correlation is among the features.

Balanceness/Biasness of our Dataset

Our dataset is balanced, i.e. our dataset is biased. Because for the output feature (which is ‘Year’) all the unique classes have an equal number of instances.



From the above bar chart , we can conclude that our dataset is balanced because the dataset is (58-42)% split in binary classification problems. Every class of our dataset ideally makes around or more than 50% of the dataset.

Dataset Pre-Processing

Dataset preprocessing, also known as data preprocessing, is a crucial step in the data preparation pipeline for machine learning and data analysis tasks. It involves a series of operations and transformations applied to raw data to make it suitable for analysis or model training. The primary goal of dataset preprocessing is to improve the quality, reliability, and effectiveness of the data used for further analysis or modeling.
In our Project , we have used some techniques of data preprocessing. They are Deleting duplicate and null values. Imputation for missing values, Handling Categorical Features/Feature Encoding and Feature Scaling.

- **Faults**
    1. Null Values: One of the notable faults in our dataset is our dataset contains many null values. It is seen that a total of 16866 null values are present in our dataset.

```
dataset.isna().sum()
```

```
Cancer                                0
Year                                  0
Month                                 0
Bleed                               576
Mode_Presentation                     0
Age                                   0
Gender                                0
Etiology                            568
Cirrhosis                           568
Size                                218
HCC_TNM_Stage                       568
HCC_BCLC_Stage                      568
ICC_TNM_Stage                      1434
Treatment_grps                        8
Survival_fromMDM                      0
Alive_Dead                            0
Type_of_incidental_finding         1448
Surveillance_programme              568
Surveillance_effectiveness         1436
Mode_of_surveillance_detection     1511
Time_diagnosis_1st_Tx              1289
Date_incident_surveillance_scan    1848
PS                                    9
Time_MDM_1st_treatment             1258
Time_decisiontotreat_1st_treatment 1521
Prev_known_cirrhosis                 18
Months_from_last_surveillance      1460
dtype: int64
```

2. Categorical Values: Another fault in our dataset is our dataset contains many categorical values. In our dataset there are about 18 categorical features which contain many categorical values such as ‘Y’, ‘N’, ‘A’, ‘Consistent’, ‘Alive’, ‘Dead’ etc.

- **Solutions**

    1. The solution for Null Values is Imputation for Null values:

We used the following section of code to fill up the missing values/null values:

```
from sklearn.impute import SimpleImputer
import numpy as np

# The imputation strategy:  mean, median, most_frequent,
constant
```

```
impute = SimpleImputer(missing_values=np.nan,
strategy='most_frequent')

list_of_features = ["Bleed", "Etiology",
"Cirrhosis","Size","HCC_TNM_Stage","HCC_BCLC_Stage",
"ICC_TNM_Stage","Treatment_grps","Type_of_incidental_find
ing","Surveillance_programme","Surveillance_effectiveness
","Mode_of_surveillance_detection","Time_diagnosis_1st_Tx
","Date_incident_surveillance_scan","Time_MDM_1st_treatme
nt","Time_decisiontotreat_1st_treatment","Prev_known_cirr
hosis","Months_from_last_surveillance","Type_of_incidenta
l_finding"]

dataset[list_of_features] =
impute.fit_transform(dataset[list_of_features])
```

The above code fills up the missing values by filling the most frequently used data in our dataset.

After imputation for null values there is no null values in the respective features:

```
dataset.isnull().sum()
```

```
Cancer                                0
Year                                  0
Month                                 0
Bleed                                 0
Mode_Presentation                     0
Age                                   0
Gender                                0
Etiology                              0
Cirrhosis                             0
Size                                  0
HCC_TNM_Stage                         0
HCC_BCLC_Stage                        0
ICC_TNM_Stage                         0
Treatment_grps                        0
Survival_fromMDM                      0
Alive_Dead                            0
Type_of_incidental_finding            0
Surveillance_programme                0
Surveillance_effectiveness            0
Mode_of_surveillance_detection        0
Time_diagnosis_1st_Tx                 0
Date_incident_surveillance_scan       0
PS                                    9
Time_MDM_1st_treatment                0
Time_decisiontotreat_1st_treatment    0
Prev_known_cirrhosis                  0
Months_from_last_surveillance         0
dtype: int64
```

2. The Solution for Categorical Values is Handling Categorical Features or Feature Encoding: We used Label Encoding Technique to encode the categorical values of categorical features. Label Encoding assigns a unique integer to each unique category or label within a categorical variable. The labels are ordered in a sequential manner, often starting from 0 or 1.
   We used the following section of code to encode the categorical features:

```
from sklearn.preprocessing import LabelEncoder

# Set up the LabelEncoder object
enc = LabelEncoder()

# Apply the encoding to the "Accessible" column
dataset['Year'] =
enc.fit_transform(dataset['Year'])

# Compare the two columns
print(dataset[['Year']].head(200))
```

This section of code shows that the categorical values which are ‘Prepandemic’ and ‘Pandemic’ are encoded to ‘1’ and ‘0’ respectively. The output of this code is shown below:

```
     Year
0       1
1       1
2       1
3       1
4       1
..    ...
195     0
196     0
197     0
198     0
199     0

[200 rows x 1 columns]
```

Like this , other categorical features like ‘Bleed’ , ‘Gender’, ‘Alive_Dead’ etc are encoded through Label Encoder.
Again, there are some other categorical features which contain more than 2 categorical values. We used `enc.fit_transform()` to assign those categorical values to 0,1,2,3, 4 etc. For example:

```
from sklearn.preprocessing import LabelEncoder

# Set up the LabelEncoder object
enc = LabelEncoder()

# Apply the encoding to the "Accessible" column
dataset['Etiology'] =
enc.fit_transform(dataset['Etiology'])

# Compare the two columns
print(dataset[['Etiology']].head(100))
```

Output:

```
    Etiology
0          4
1          0
2          0
3          0
4          0
..       ...
95         7
96         5
97         3
98         7
99         5

[100 rows x 1 columns]
```

Feature Scaling

Feature scaling is a data preprocessing technique used in machine learning to standardize or normalize the range of independent variables or features in a dataset. It's essential because many machine learning algorithms perform better when the input features are on a similar scale. Feature scaling ensures that no single feature dominates the learning process or introduces bias due to its larger numerical values.

In our project, we used MinMax Scaling Technique in our Feature Scaling. MinMaxScaler is a feature scaling technique used in data preprocessing for machine learning. It scales and transforms the features in a dataset so that they are mapped into a specific range, usually between 0 and 1. Our dataset has some features that has continuous numerical values. We applied MinMax scaling on those features so that the numerical values of continuous numerical features are normalized to standard values.

Feature Selection

Feature selection is a process in machine learning algorithm that involves choosing a subset of relevant features (variables, attributes, or columns) from the original set of features in a dataset. The goal of feature selection is to improve the performance of a machine learning model by reducing the complexity of the data while retaining the most informative and relevant features. We used heatmap and logical reasoning to select the best feature/relevant features for x_data.

The code for heatmap section:

```
import seaborn as sns
sns.set(font_scale=1.1)
corr_train=dataset.corr()
plt.figure(figsize=(30,25))
sns.heatmap(corr_train,annot=True,fmt='.3f',cmap='YlGn',linewidth=5,cbar=T
rue)
plt.show()
```

The code for logical reasoning:

```
corr_val = 0.08
```

```
relevant_features = covid_liver_corr['Year'][abs(covid_liver_corr['Year'])
> corr_val].index.tolist()
print("Relevant features:", relevant_features)
```

Here the threshold value of correlation of all features with respect to target label is 0.08 or 8%. By using the following code, we dropped the irrelevant features and created new preprocessed dataset for model application and training:

```
to_drop=['Cancer',
'Month','Age','Gender','Etiology','Cirrhosis','HCC_TNM_Stage','Treatment_g
rps','Type_of_incidental_finding','Surveillance_programme','Surveillance_e
ffectiveness','Mode_of_surveillance_detection','Time_diagnosis_1st_Tx','PS
','Prev_known_cirrhosis','Months_from_last_surveillance','Time_decisiontot
reat_1st_treatment']
new_dataset=dataset.drop(to_drop, axis=1)
new_dataset.head()
```

The new dataset with relevant/Selected features:

	Year	Bleed	Mode_Presentation	Size	HCC_BCLC_Stage	ICC_TNM_Stage	Survival_fromMDM	Alive_Dead	Date_incident_surveillance_scan	Time_MDM_1st_treatment
0	1	0	0.5	0.057143	0.25	1.0	0.998780	0	1	0.098929
1	1	0	0.5	0.142857	1.00	1.0	0.093293	1	1	0.143037
2	1	0	0.5	0.200000	0.50	1.0	0.457317	1	1	0.143037
3	1	0	0.0	0.333333	0.75	1.0	0.043598	1	1	0.143037
4	1	0	0.0	0.238095	0.00	1.0	0.991768	0	1	0.143037

Dataset Splitting

Dataset splitting, also known as data splitting or data partitioning, is a crucial step in the process of developing and evaluating machine learning algorithms. It involves dividing the available dataset into different subsets for the purposes of training, validation, and testing. The primary motivation behind dataset splitting is to assess the performance of a machine learning model on unseen data and to prevent overfitting, which occurs when a model performs well on the training data but poorly on new, unseen data.
We splitted our datasets into two subsets: train and test. We splitted 80% of our dataset for training model and 20% of our dataset for testing.

```
X = new_dataset.drop("Year", axis=1)

y = new_dataset["Year"]
```

In this code we created two datasets. X dataset contains the selected features without the targeted label. y dataset contains the targeted feature.

```
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)
```

By the above code we created training and testing datasets for the respective datasets X and y.

Model Training & Testing

The main idea behind this prediction based project is to derive a prediction whether a primary liver cancer patient is of Pre-Pandemic or Post-Pandemic based on the given categorical and numerical values or data of that patient. For this prediction we used classification based problems. We applied three models namely KNN, Decision Tree and Logistics Regression.

KNN Model

KNN (k-Nearest Neighbors) stands as a versatile supervised machine learning algorithm adept at addressing both classification and regression challenges. This method hinges on leveraging the collective insight of the k nearest data points within the training set to predict the outcome for a new, unseen data point. The concept of "closeness" is determined using a specified distance metric, such as Euclidean distance.

In the context of classification, KNN endeavors to assign a label to the new data point by examining the k closest neighbors from the training set. These neighbors collectively contribute to classifying the new point, based on the majority class representation among them. The selection of the hyperparameter k remains pivotal and offers an opportunity for fine-tuning to enhance the model's performance and predictive accuracy.

The used code in our project for implementation of KNN model:

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# Predict on the test set
y_pred = knn.predict(X_test)

# Evaluate the model
accuracy1 = accuracy_score(y_test, y_pred)
confusion_mat = confusion_matrix(y_test, y_pred)
```

```
#Find F1 Accuracy
f1 = f1_score(y_test,y_pred)
print("F1 accuracy:{:.2f}%".format(f1 * 100))
# Print accuracy and confusion matrix
print("Accuracy: {:.2f}%".format(accuracy1 * 100))
#print("Confusion Matrix:\n", confusion_mat)
```

Result:

```
F1 accuracy:91.52%
Accuracy: 90.55%
```

Decision Tree Model

A decision tree is a popular machine learning algorithm used for both classification and regression tasks. It models decisions and their possible consequences in a tree-like structure. Each internal node of the tree represents a decision based on a particular feature, and each leaf node represents an outcome or prediction. threshold.Once the decision tree has been constructed, it can be used to make predictions on new, unseen data by traversing the tree from the root node to a leaf node, based on the values of the input features. The class or value associated with the leaf node represents the predicted output variable.

The used code in our project for implementation of Decision Tree model:

```
clf = DecisionTreeClassifier()

# Train the classifier on the training data
clf.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = clf.predict(X_test)



y_true = y_test # true class labels of test set
y_pred = clf.predict(X_test) # predicted class labels of test set
```

```
cm = confusion_matrix(y_true, y_pred)

#Find F1 Accuracy
f1 = f1_score(y_test,y_pred)
print("F1 accuracy:{:.2f}%".format(f1 * 100))
# Calculate accuracy of the model
accuracy2 = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy2 * 100))
```

Result:

```
F1 accuracy:99.12%
Accuracy: 99.00%
```

Logistics Regression Model

Logistic Regression is a fundamental machine learning algorithm used for binary classification tasks. Despite its name, it's actually a method for classification, not regression. It predicts the probability that a given input belongs to one of two possible classes (typically 0 or 1), and then makes a discrete classification decision based on that probability.The estimated parameters of the logistic function are obtained using a process called maximum likelihood estimation, where the model is trained on a labeled dataset with known class labels. Once the logistic regression model is trained, it can be used to make predictions on new, unseen data points by calculating the probability that the input data point belongs to the positive class based on its feature values. A threshold can be chosen to convert these probabilities into binary class labels.

The used code in our project for implementation of Logistics Regression model:

```
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)

# Evaluate the model
```

```
accuracy4 = accuracy_score(y_test, y_pred)
confusion_mat = confusion_matrix(y_test, y_pred)

#Find F1 Accuracy
f1 = f1_score(y_test,y_pred)
print("F1 accuracy:{:.2f}%".format(f1 * 100))
# Print accuracy and confusion matrix
print("Accuracy: {:.2f}%".format(accuracy3 * 100))
#print("Confusion Matrix:\n", confusion_mat)
```

Result:

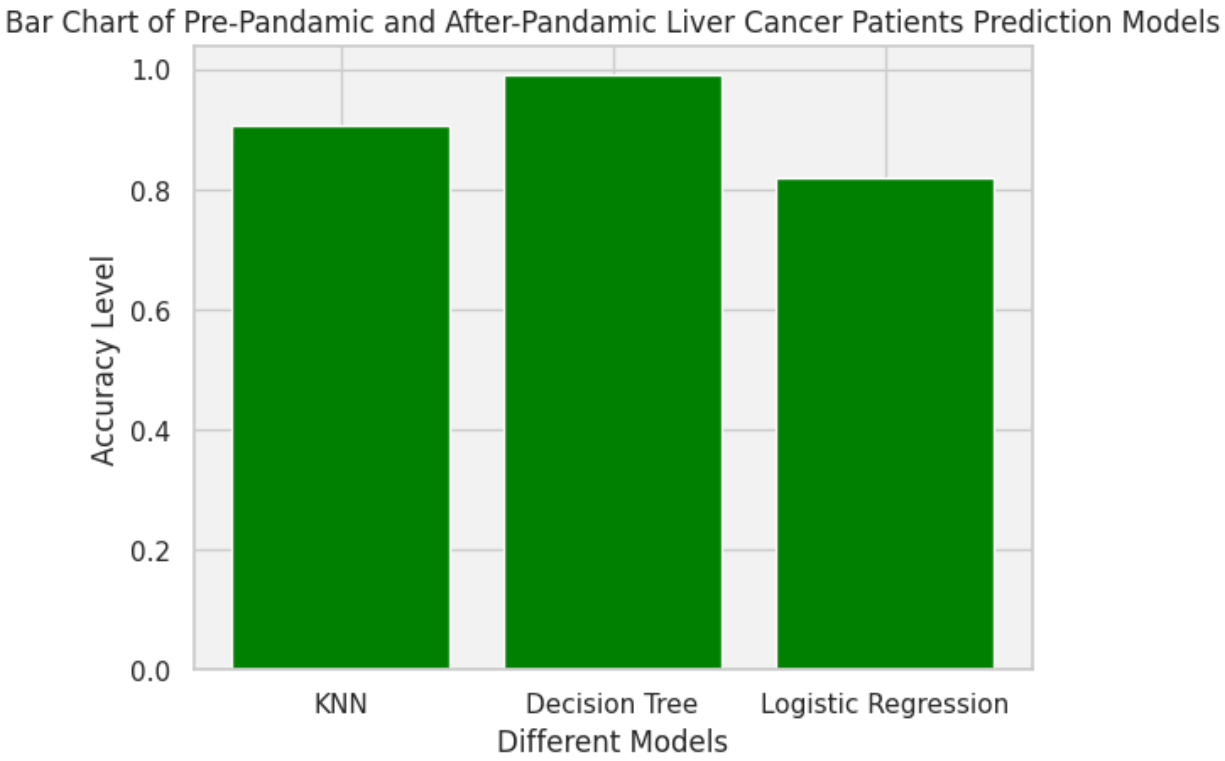
```
F1 accuracy:81.62%
Accuracy: 81.84%
```

Model Selection/Comparison Analysis

A table to demonstrate the comparison among the three applied models based on the accuracy:

Name of the Model	Accuracy (%)	Error (%)
KNN	90.55%	9.45%
Decision Tree	99.0%	1.0%
Logistics Regression	81.84%	18.16%

Barchart Showcasing prediction accuracy of all models:



Result analysis:

From the above barchart we see that all the three models have hit above 80% accuracy; which means when we train our preprocessed dataset through KNN, Decision Tree and Logistics Regression model, the input datasets give more than 80% correct and accurate results. Among the three models, we think the Decision Tree Model is most suitable for our project because it gives the highest accuracy (99.0%). On the other hand, the KNN model has also shown an optimized accuracy (90.55%), i.e. the KNN model is also suitable for our project but not as Decision Tree Model. The Logistic Regression Model is less suitable to train our dataset compared to the KNN and Decision Tree because it gives the lowest accuracy (81.84%).

Comparison among the models based on Precision

Precision For KNN Model:

	precision	recall	f1-score	support
0	0.89	0.89	0.89	191
1	0.90	0.90	0.90	211
accuracy			0.89	402
macro avg	0.89	0.89	0.89	402
weighted avg	0.89	0.89	0.89	402

Precision for Decision Tree Model:

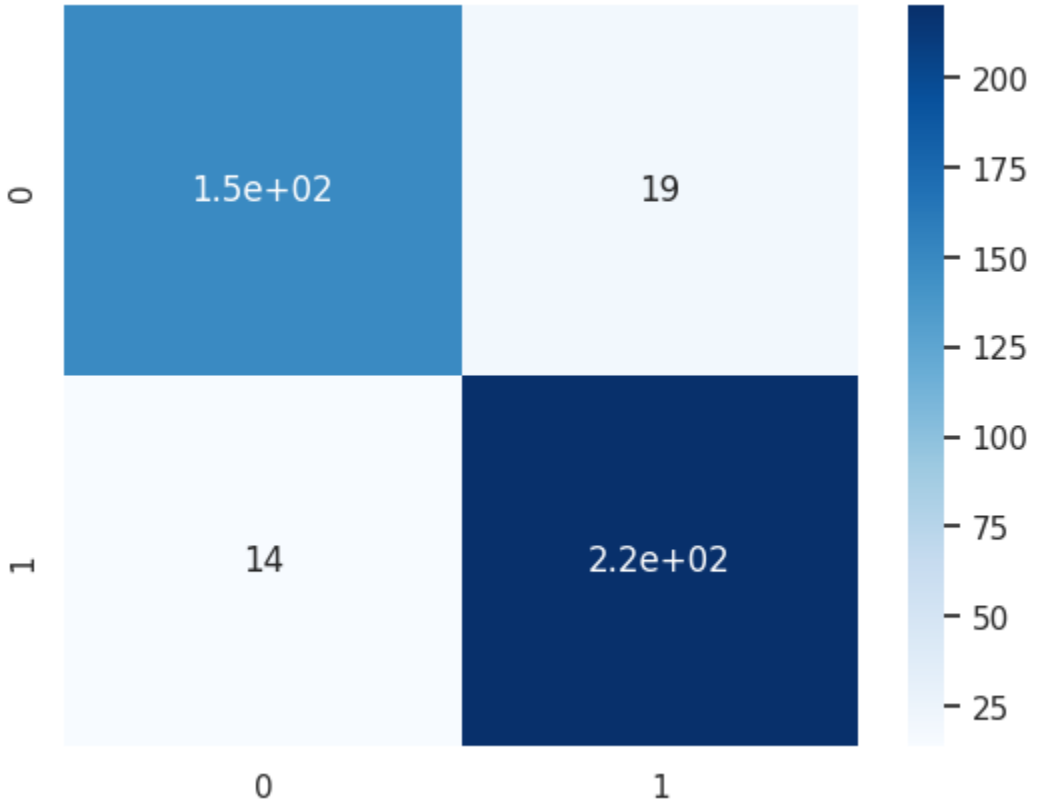
	precision	recall	f1-score	support
0	0.98	1.00	0.99	168
1	1.00	0.98	0.99	234
accuracy			0.99	402
macro avg	0.99	0.99	0.99	402
weighted avg	0.99	0.99	0.99	402

Precision for Logistics Regression Model:

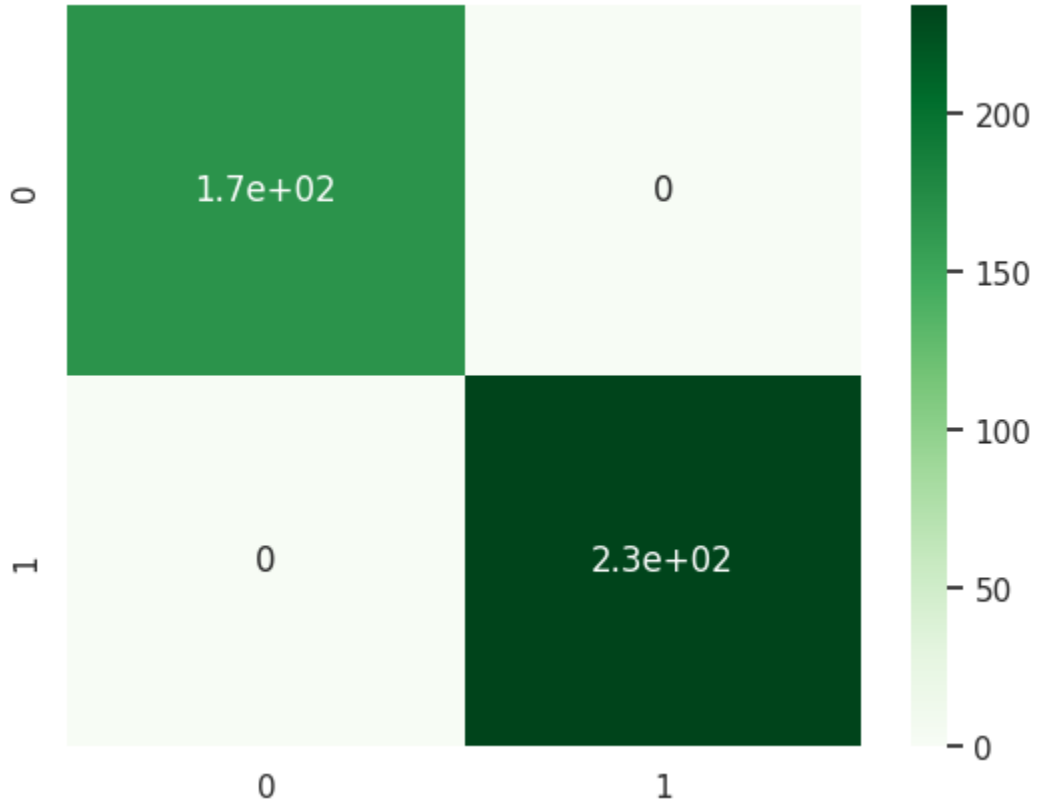
	precision	recall	f1-score	support
0	0.74	0.68	0.71	168
1	0.78	0.82	0.80	234
accuracy			0.77	402
macro avg	0.76	0.75	0.76	402
weighted avg	0.76	0.77	0.76	402

Comparison among the models based on Confusion Matrix

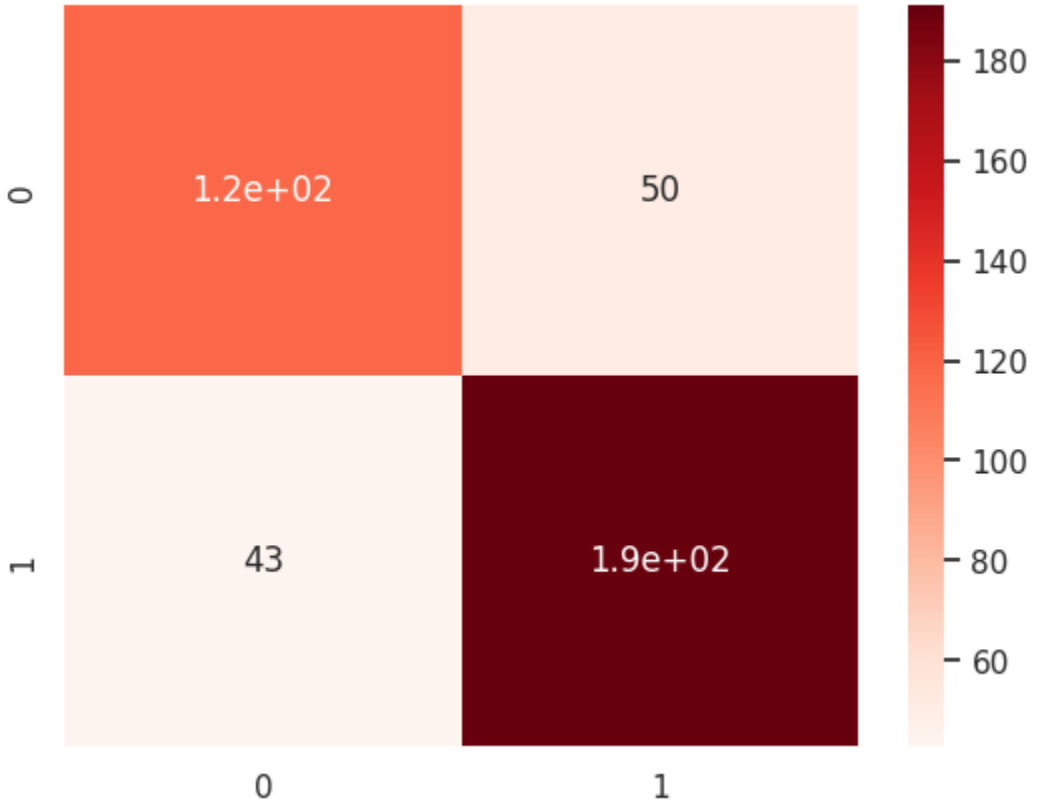
Confusion for KNN Model:



Confusion for Decision Tree Model:



Confusion for Logistics Regression Model:



Conclusion

In this project, our focus revolved around constructing and evaluating machine learning models geared towards predicting the impact of COVID-19 on primary liver cancer patients. The foundation for these predictions lay in a comprehensive compilation of clinical and demographic attributes. Through the course of this study, we unearthed the true potential of machine learning techniques in rendering precise conditions of a patient during COVID-19, thereby offering crucial early insights to trigger timely interventions.

The efficacy of the predictive models stood validated by performance metrics including accuracy, precision, recall, and F1 score. These metrics collectively testified to the models' effectiveness in identifying patients whether he/she is of before pandemic or after pandemic. These metrics also deliver us whether liver cancer patients are able to survive or not even after the arrival of COVID-19.

The utilization of KNN, with its proximity-based decision-making, has enabled the project to draw correlations between patient attributes and potential COVID-19 outcomes. By identifying similar cases, KNN offers insights into possible scenarios, empowering healthcare professionals with informed strategies to safeguard primary liver cancer patients. The Decision Tree Model, with its branching logic, provides a tangible visualization of key factors driving potential impacts. By revealing critical thresholds and variables, this model furnishes a roadmap for understanding the intricate relationships at play and facilitates early intervention strategies. Logistic Regression, adept at probability estimation, has bestowed the project with precise insights into the likelihood of COVID-19 impact. By quantifying risks, this model contributes to a nuanced understanding of the potential challenges primary liver cancer patients might face.