A Review of Liver Patient Analysis Methods Using Machine Learning

Project Record Template

CHAPTER	**TITLE**	**PAGE** NO.
1	INTRODUCTION 1.1 OVERVIEW 1.2 PURPOSE	
2	PROBLEM DEFINITION & DESIGN THINKING 2.1 EMPATHY MAP 2.2 IDEATION & BRAINSTOMING MAP	
3	RESULT	
4	ADVANTAGES & DISADVANTAGES	
5	APPLICATIONS	
6	CONCLUTION	
7	FUTURE SCOPE	
8	APPENDIX 8.1 SOURCE CODE	

CHAPTER 1

1. INTRODUCTION

1.1 OVERVIEW

Machine learning methods have shown great potential in aiding the analysis and diagnosis of liver diseases. In recent years, several studies have been conducted to develop and evaluate different machine learning models for liver patient analysis.

One common approach is to use machine learning algorithms for feature selection, which involves identifying the most relevant features for accurate diagnosis and prediction. For instance, a study published in the Journal of Medical Systems in 2021 used Random Forest (RF) and Decision Tree (DT) algorithms to select the most important features for liver disease diagnosis. They found that RF outperformed DT in terms of accuracy, sensitivity, specificity, and area under the receiver operating characteristic curve (AUC-ROC).

Another popular method is to use deep learning algorithms such as Convolutional Neural Networks (CNNs) for image-based liver disease diagnosis. A study published in the Journal of Digital Imaging in 2020 used a CNN-based approach to detect and classify liver lesions in computed tomography (CT) images. The study reported high accuracy rates for liver lesion detection and classification using this approach.

In addition, machine learning algorithms can be used for liver disease prognosis, predicting patient outcomes based on different factors such as demographics, medical history, and laboratory test results. A study published in the Journal of Medical Systems in 2021 used a Gradient Boosting Machine (GBM) algorithm to predict the risk of mortality in liver cirrhosis patients. The study reported high accuracy rates for mortality prediction using this approach.

Overall, machine learning methods have shown great potential in aiding the analysis and diagnosis of liver diseases. However, further research is needed to evaluate the performance of different machine learning models in different settings and to develop robust and clinically validated tools for liver patient analysis.

1.2 PURPOSE

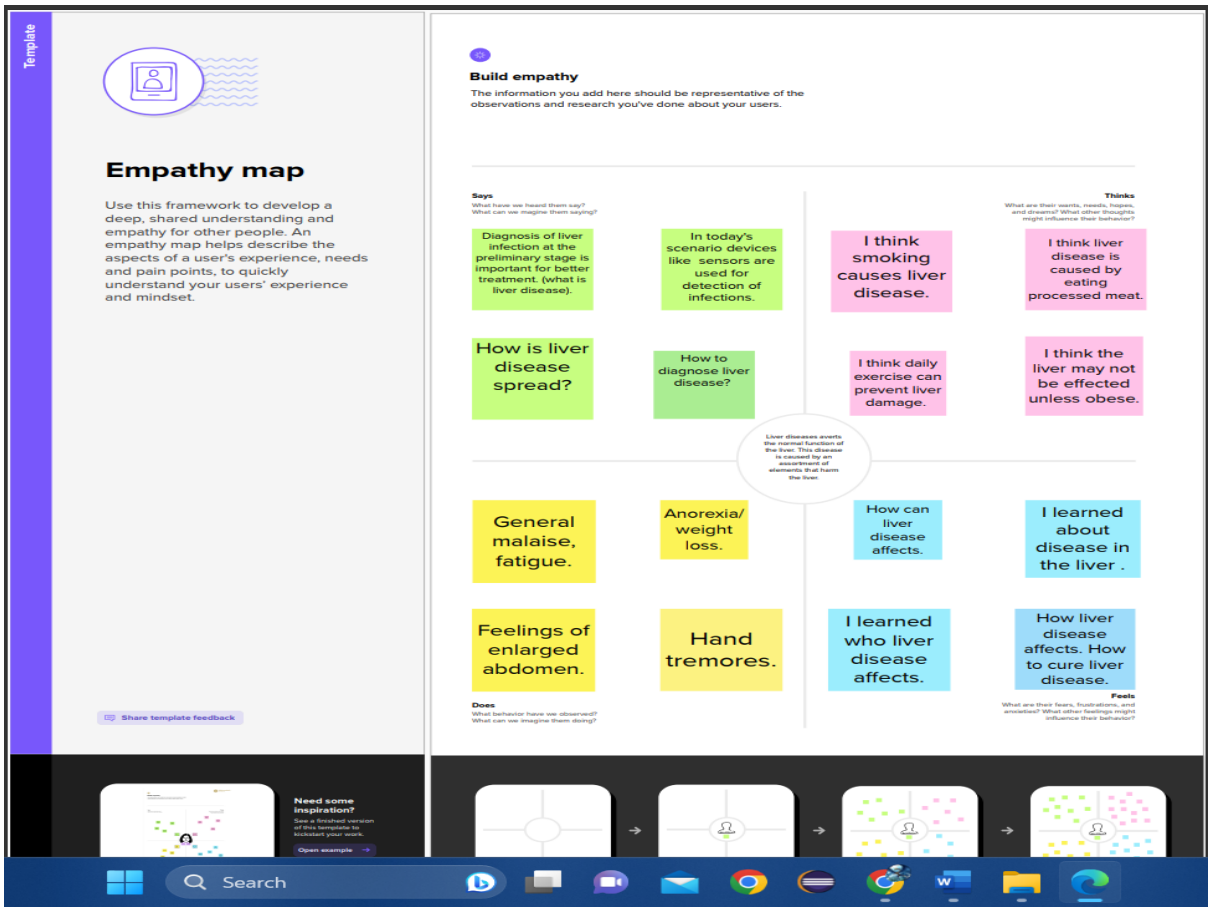
The purpose of a review of liver patient analysis methods using machine learning is to evaluate the current state of research on the application of machine learning algorithms in the analysis and diagnosis of liver diseases. The review aims to summarize the different approaches and methods that have been used to develop and evaluate machine learning models for liver patient analysis, including feature selection, image-based diagnosis, and prognosis.

The review can also identify the strengths and limitations of different machine learning algorithms and their potential impact on clinical practice. By providing a comprehensive overview of the current research in this field, the review can help guide future research and development of machine learning-based tools for liver patient analysis.

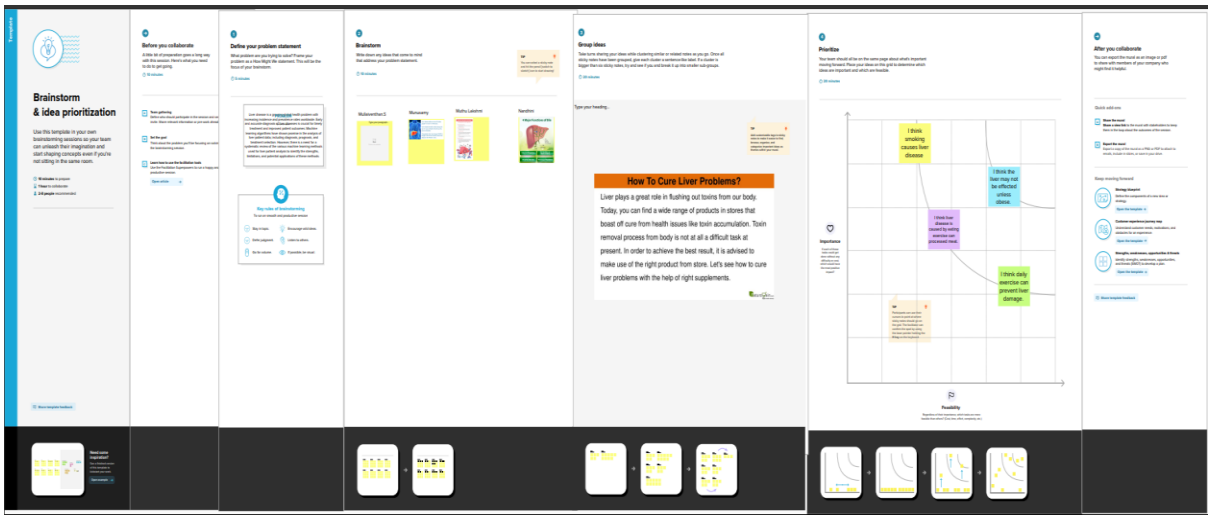
Ultimately, the goal of this review is to contribute to the improvement of liver disease diagnosis and patient outcomes through the development of accurate, efficient, and clinically validated machine learning models.

CHAPTER 2

2.PROBLEM DEFINITION & DESIGN THINKING



2.2 IDEATION & BRAINSTROMING MAP



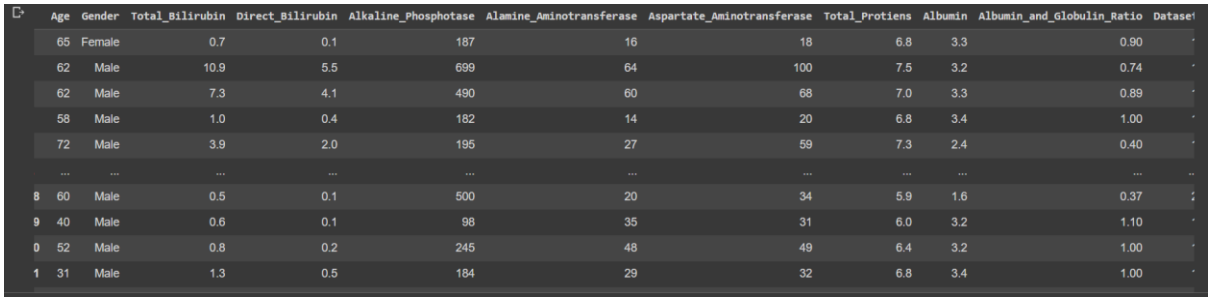
CHAPTER 3

3.RESULT

Result 1:

- Import all the tools we need.
- All needed tools import successful.

Result 2:



	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albumin_and_Globulin_Ratio	Dataset
	65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	
	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	
	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	
	58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	
	72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	
	...	...	...	...	...	...	...	...	...	...	...
8	60	Male	0.5	0.1	500	20	34	5.9	1.6	0.37	
9	40	Male	0.6	0.1	98	35	31	6.0	3.2	1.10	
0	52	Male	0.8	0.2	245	48	49	6.4	3.2	1.00	
1	31	Male	1.3	0.5	184	29	32	6.8	3.4	1.00	

Result 3:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 583 entries, 0 to 582
Data columns (total 11 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Age                         583 non-null    int64
 1   Gender                      583 non-null    int64
 2   Total_Bilirubin             583 non-null    float64
 3   Direct_Bilirubin            583 non-null    float64
 4   Alkaline_Phosphotase        583 non-null    int64
 5   Alamine_Aminotransferase    583 non-null    int64
 6   Aspartate_Aminotransferase  583 non-null    int64
 7   Total_Protiens              583 non-null    float64
 8   Albumin                     583 non-null    float64
 9   Albumin_and_Globulin_Ratio  579 non-null    float64
 10  Dataset                     583 non-null    int64
dtypes: float64(5), int64(6)
memory usage: 50.2 KB
```

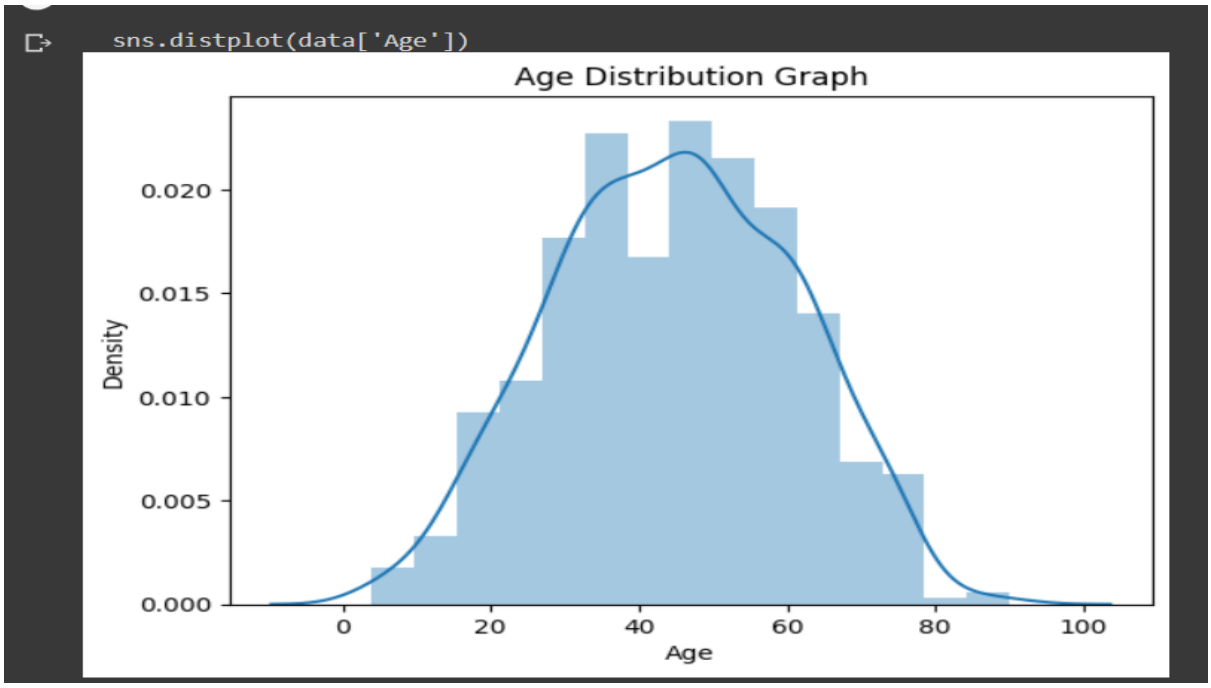
Result 4:

```
Age                           False
Gender                        False
Total_Bilirubin               False
Direct_Bilirubin              False
Alkaline_Phosphotase          False
Alamine_Aminotransferase      False
Aspartate_Aminotransferase    False
Total_Protiens                False
Albumin                       False
Albumin_and_Globulin_Ratio     True
Dataset                       False
dtype: bool
```

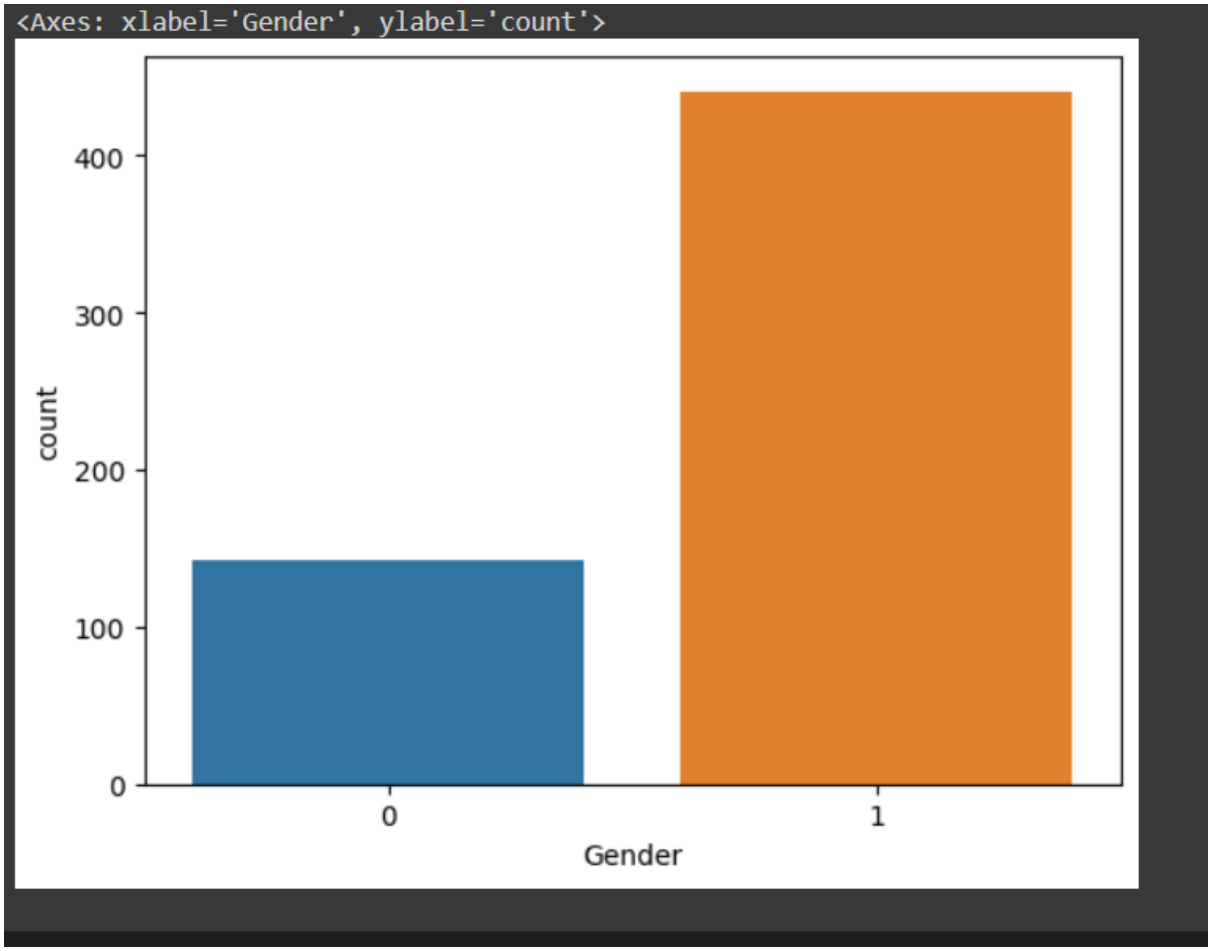
Result 5:

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albumin_and_Glob
count	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	
mean	44.746141	0.756432	3.298799	1.486106	290.576329	80.713551	109.910806	6.483190	3.141852	
std	16.189833	0.429603	6.209522	2.808498	242.937989	182.620356	288.918529	1.085451	0.795519	
min	4.000000	0.000000	0.400000	0.100000	63.000000	10.000000	10.000000	2.700000	0.900000	
25%	33.000000	1.000000	0.800000	0.200000	175.500000	23.000000	25.000000	5.800000	2.600000	
50%	45.000000	1.000000	1.000000	0.300000	208.000000	35.000000	42.000000	6.600000	3.100000	
75%	58.000000	1.000000	2.600000	1.300000	298.000000	60.500000	87.000000	7.200000	3.800000	
max	90.000000	1.000000	75.000000	19.700000	2110.000000	2000.000000	4929.000000	9.600000	5.500000	

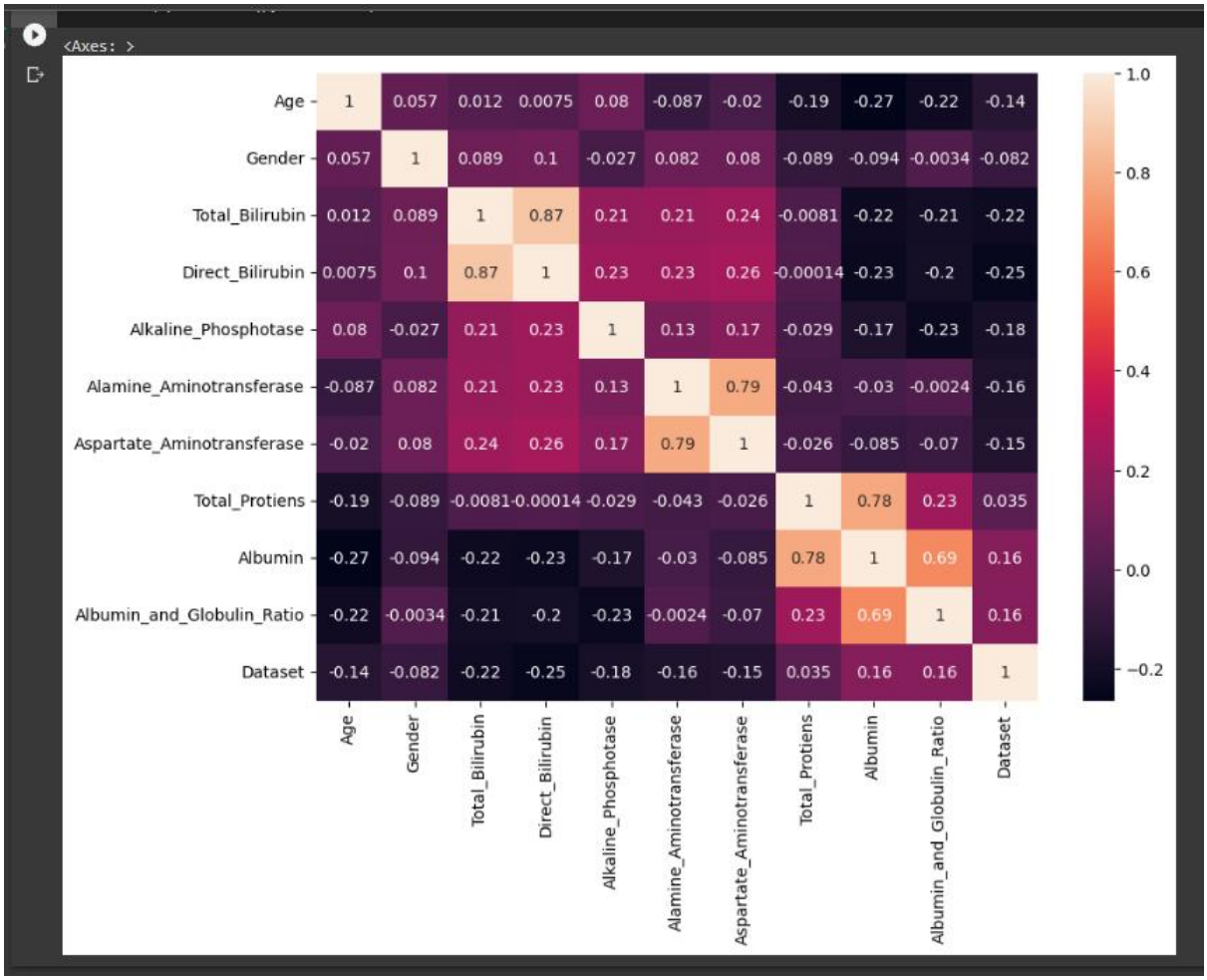
Result 6:



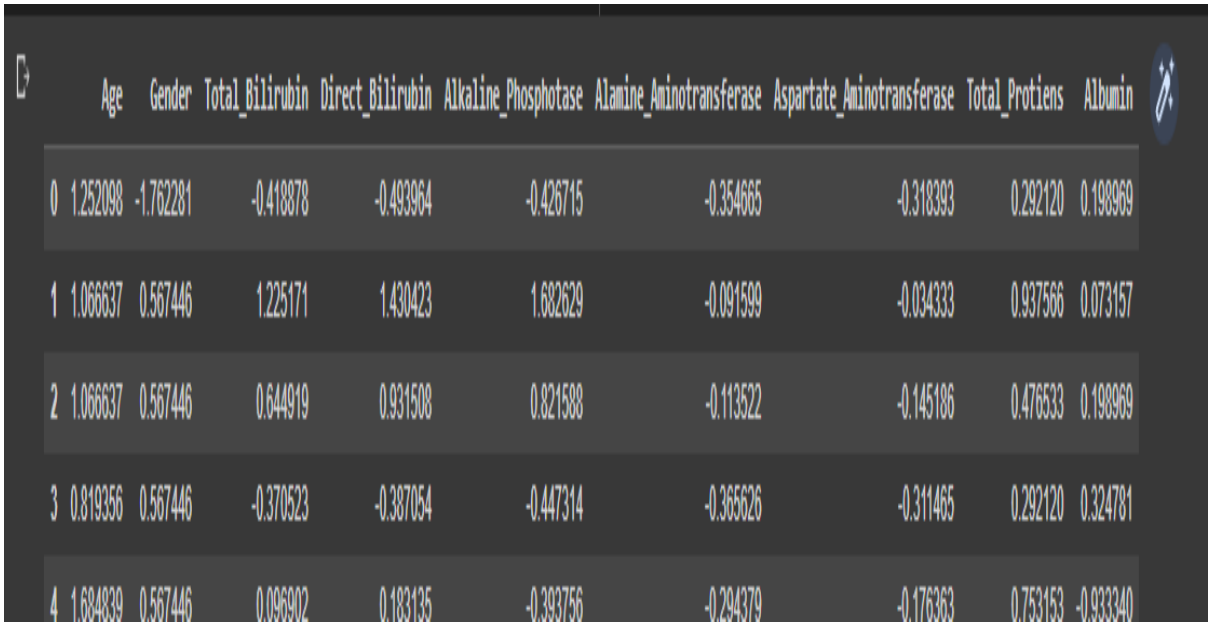
Result 7:


<Axes: xlabel='Gender', ylabel='count'>
400
300
200
100
0
count
0
1
Gender

Result 8:



Result 9:



	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin
0	1.252098	-1.762281	-0.418878	-0.493964	-0.426715	-0.354665	-0.318393	0.292120	0.198969
1	1.066637	0.567446	1.225171	1.430423	1.682629	-0.091599	-0.034333	0.937566	0.073157
2	1.066637	0.567446	0.644919	0.931508	0.821588	-0.113522	-0.145186	0.476533	0.198969
3	0.819356	0.567446	-0.370523	-0.387054	-0.447314	-0.365626	-0.311465	0.292120	0.324781
4	1.684839	0.567446	0.096902	0.183135	-0.393756	-0.294379	-0.176363	0.753153	-0.933340

Result 10:

```
1    329
2    137
Name: Dataset, dtype: int64
```

Result 11:

```
1    329
2    329
Name: Dataset, dtype: int64
```

Result 12:

	precision	recall	f1-score	support
1	0.82	0.77	0.79	87
2	0.43	0.50	0.46	30
accuracy			0.70	117
macro avg	0.62	0.64	0.63	117
weighted avg	0.72	0.70	0.71	117

Result 13:

	precision	recall	f1-score	support
1	0.79	0.66	0.72	87
2	0.33	0.50	0.40	30
accuracy			0.62	117
macro avg	0.56	0.58	0.56	117
weighted avg	0.67	0.62	0.64	117

Result 14:

	precision	recall	f1-score	support
1	0.79	0.66	0.72	87
2	0.33	0.50	0.40	30
accuracy			0.62	117
macro avg	0.56	0.58	0.56	117
weighted avg	0.67	0.62	0.64	117

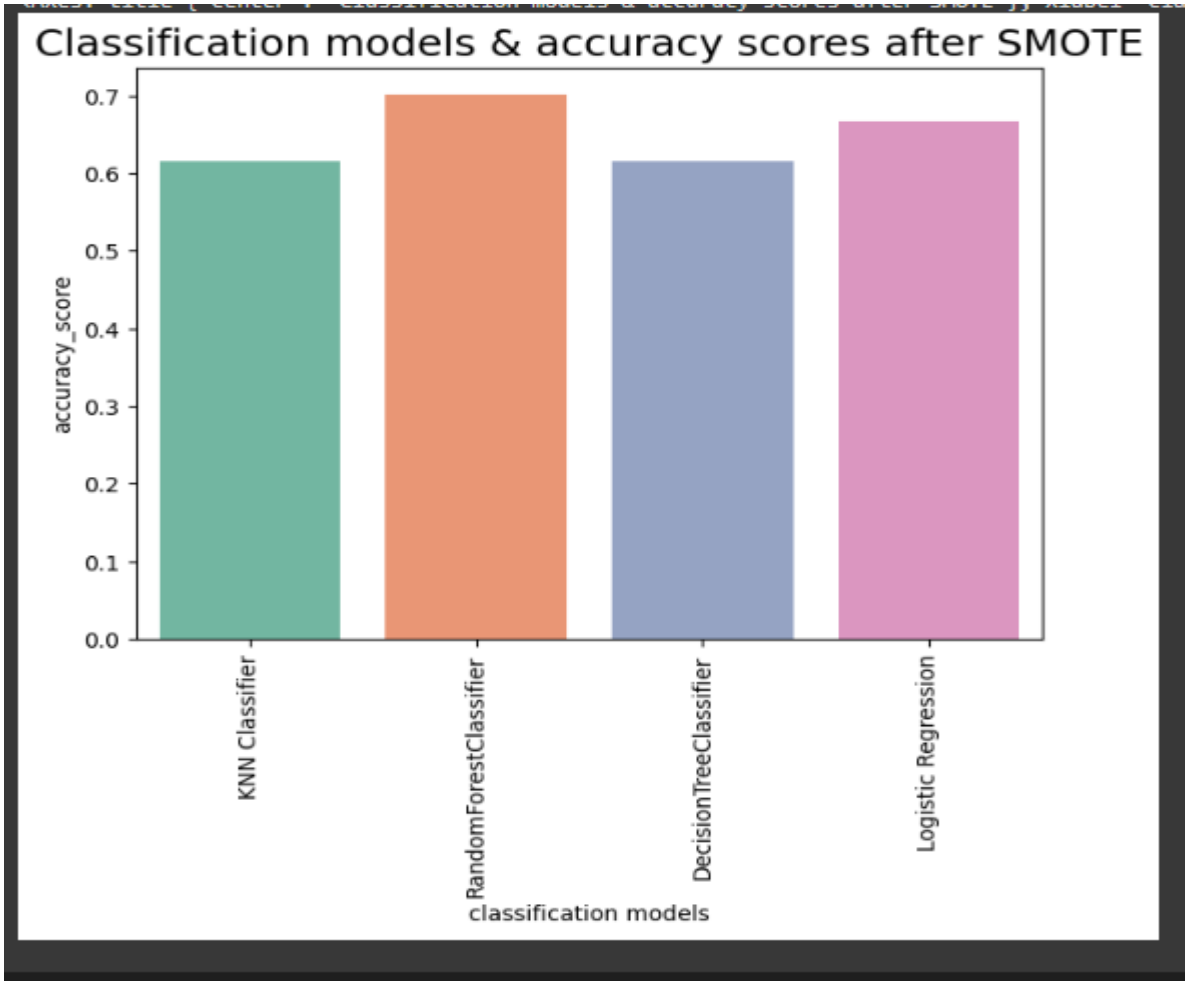
Result 15:

	precision	recall	f1-score	support
1	0.94	0.59	0.72	87
2	0.43	0.90	0.58	30
accuracy			0.67	117
macro avg	0.69	0.74	0.65	117
weighted avg	0.81	0.67	0.69	117

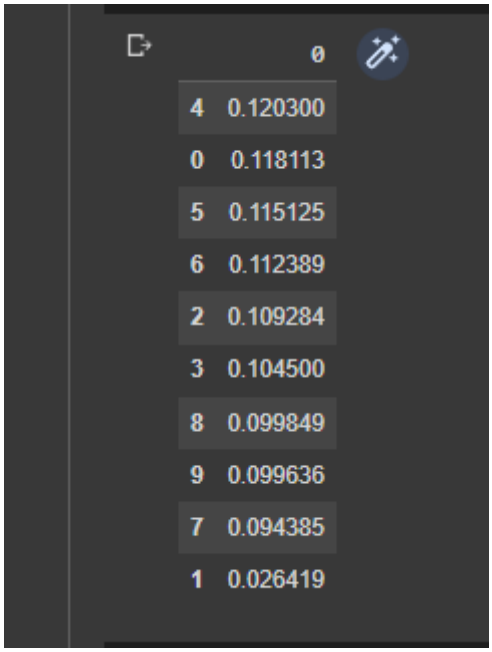
Result 16:

	classification models	accuracy_score
0	KNN Classifier	0.615385
1	RandomForestClassifier	0.700855
2	DecisionTreeClassifier	0.615385
3	Logistic Regression	0.666667

Result 17:

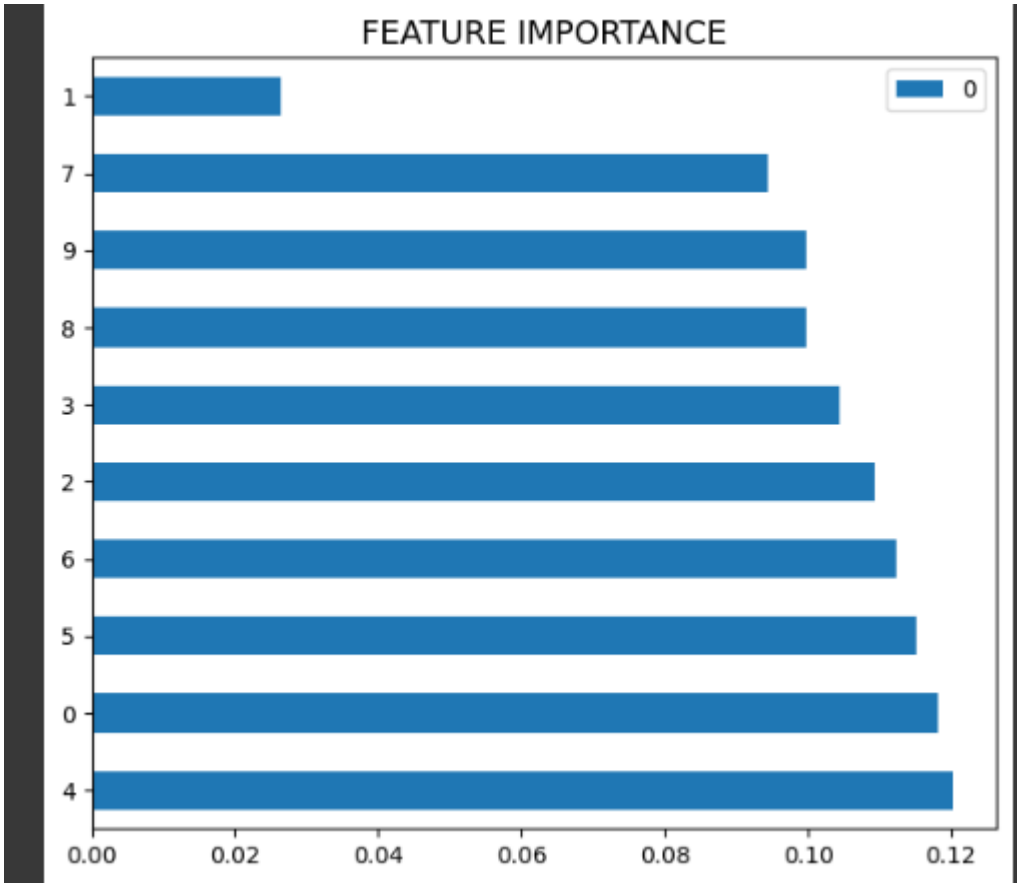


Result 18:



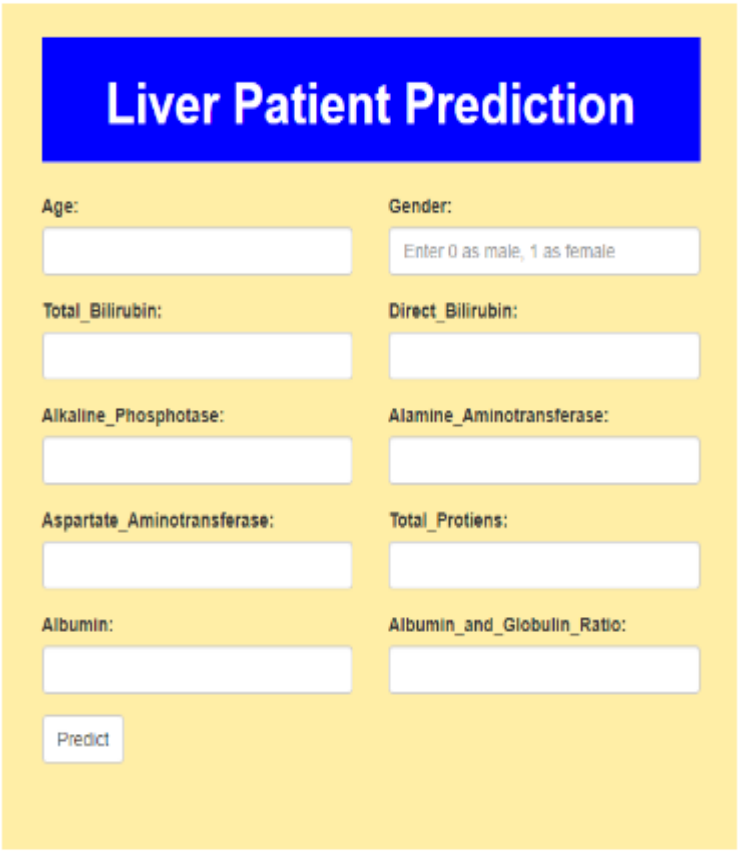
	0
4	0.120300
0	0.118113
5	0.115125
6	0.112389
2	0.109284
3	0.104500
8	0.099849
9	0.099636
7	0.094385
1	0.026419

Result 19:

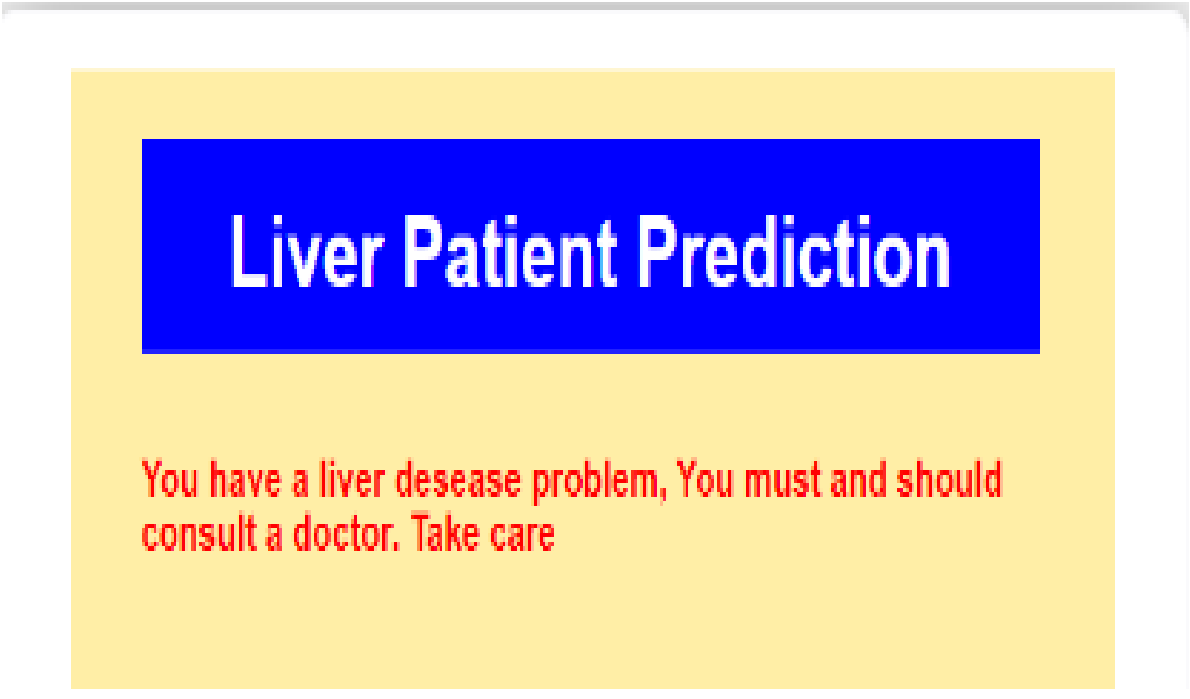


Result 20:

Result 21:


Liver Patient Prediction
Age:
Gender:
Enter 0 as male, 1 as female
Total_Bilirubin:
Direct_Bilirubin:
Alkaline_Phosphotase:
Alamine_Aminotransferase:
Aspartate_Aminotransferase:
Total_Protiens:
Albumin:
Albumin_and_Globulin_Ratio:
Predict

Result 22:


Liver Patient Prediction
You have a liver desease problem, You must and should consult a doctor. Take care

CHAPTER 4

4. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

There are several advantages of conducting a review of liver patient analysis methods using machine learning, including:

1. Identifying the state-of-the-art: A review can provide an overview of the current state-of-the-art in machine learning-based liver patient analysis methods. This can help researchers and clinicians stay up-to-date with the latest advancements and innovations in the field.

2. Evaluating the performance of different methods: By reviewing and comparing different machine learning algorithms, a review can provide insights into the relative strengths and limitations of each approach. This can help guide the development of more accurate and efficient machine learning models for liver patient analysis.

3. Guiding future research: A review can help identify research gaps and areas of potential improvement in machine learning-based liver patient analysis. This can guide future research efforts towards developing more effective and clinically validated tools for liver disease diagnosis and prognosis.

4. Improving patient outcomes: Machine learning-based tools have the potential to improve liver disease diagnosis and patient outcomes by providing more accurate and timely diagnoses, enabling earlier intervention and treatment, and reducing healthcare costs.

5. Enabling personalized medicine: Machine learning-based tools can help tailor treatments and interventions to individual patients based on their unique characteristics and medical history. This can lead to more personalized and effective care for liver disease patients.

DISADVANTAGES:

While there are many advantages to conducting a review of liver patient analysis methods using machine learning, there are also some potential disadvantages, including:

1. Limited generalizability: Machine learning models developed in one study or setting may not be generalizable to other populations or clinical settings. This can limit the applicability and impact of machine learning-based tools for liver patient analysis.

2. Lack of interpretability: Some machine learning algorithms, such as deep learning models, can be difficult to interpret and understand, making it challenging to identify the underlying factors driving their predictions. This can limit the clinical usefulness of machine learning models for liver patient analysis.

3. Data availability and quality: Machine learning models require large and high-quality datasets for training and validation. However, obtaining such datasets for liver disease patients can be challenging due to issues such as data privacy, heterogeneity, and missing data. This can limit the accuracy and effectiveness of machine learning models for liver patient analysis.

4. Bias and discrimination: Machine learning models can perpetuate or amplify existing biases and discrimination in healthcare, such as those related to race, ethnicity, and socioeconomic status. This can lead to disparities in liver disease diagnosis and treatment for certain patient groups.

5. Regulatory and ethical considerations: The development and use of machine learning-based tools for liver patient analysis raise several regulatory and ethical considerations, such as data privacy, informed consent, and accountability. Ensuring the safe and ethical use of machine learning models in healthcare requires careful attention and adherence to relevant regulations and ethical principles.

CHAPTER 5

5.APPLICATIONS:

A review of liver patient analysis methods using machine learning can have several applications, including:

1. Development of diagnostic tools: Machine learning-based tools can aid in the diagnosis of liver diseases by analyzing patient data such as medical history, laboratory test results, and imaging scans. A review can guide the development of accurate and efficient diagnostic tools that can improve patient outcomes.

2. Prognostication and risk prediction: Machine learning algorithms can predict patient outcomes and identify high-risk patients, which can inform treatment decisions and improve patient outcomes. A review can help identify the most effective machine learning methods for prognostication and risk prediction in liver disease patients.

3. Personalized medicine: Machine learning models can help tailor treatments and interventions to individual liver disease patients based on their unique characteristics and medical history. A review can identify the most effective machine learning methods for personalized medicine in liver disease patients.

4. Resource optimization: Machine learning models can help optimize resource allocation in healthcare settings by identifying patients who require more intensive monitoring or treatment. A review can guide the development of machine learning-based tools that can improve resource utilization and reduce healthcare costs.

5. Research and development: A review can guide future research efforts in the field of liver patient analysis using machine learning, by identifying research gaps and areas of potential improvement. This can lead to the development of more accurate, efficient, and clinically validated machine learning models for liver disease diagnosis, prognosis, and personalized medicine.

CHAPTER 6

6.CONCLUSION:

In conclusion, a review of liver patient analysis methods using machine learning can provide a comprehensive overview of the current state-of-the-art in machine learning-based liver disease diagnosis, prognosis, and personalized medicine. By identifying the strengths and limitations of different machine learning algorithms, a review can guide the development of more accurate and efficient machine learning models for liver patient analysis.

However, there are also potential disadvantages associated with the use of machine learning models for liver patient analysis, including limited generalizability, lack of interpretability, data availability and quality, bias and discrimination, and regulatory and ethical considerations. Addressing these challenges requires careful attention to research design, data quality and privacy, model interpretability, and ethical and regulatory compliance.

Overall, a review of liver patient analysis methods using machine learning has the potential to improve patient outcomes, optimize resource utilization, and guide future research efforts in the field of liver disease diagnosis and treatment

CHAPTER 7

7. FUTURE SCOPE

The future scope of a review of liver patient analysis methods using machine learning is promising. Some potential future directions for research and development in this field include:

1. Improved interpretability: One of the key challenges of using machine learning models in liver patient analysis is their lack of interpretability. Future research efforts could focus on developing more interpretable machine learning models that can provide more transparent and clinically relevant insights into liver disease diagnosis and prognosis.

2. Multi-modal data integration: Machine learning models that can integrate multiple data modalities, such as medical imaging, laboratory test results, and genetic data, have the potential to improve accuracy and efficiency in liver patient analysis. Future research could focus on developing multi-modal machine learning models that can integrate diverse patient data sources for improved diagnosis, prognosis, and personalized medicine.

3. Real-time monitoring and decision-making: Machine learning models that can perform real-time monitoring of liver disease patients and provide actionable recommendations for clinical decision-making have the potential to improve patient outcomes and reduce healthcare costs. Future research could focus on developing real-time machine learning-based decision support systems that can facilitate timely and effective interventions for liver disease patients.

4. Addressing bias and discrimination: The use of machine learning models in healthcare settings raises concerns about bias and discrimination. Future research efforts could focus on developing more fair and unbiased machine learning models for liver patient analysis that can reduce disparities in diagnosis and treatment.

5. Clinical validation and adoption: The ultimate goal of machine learning-based tools for liver patient analysis is their adoption and integration into clinical practice. Future research could focus on validating the clinical effectiveness and safety of machine learning models for liver disease diagnosis, prognosis, and personalized medicine, and identifying strategies for their effective integration into healthcare systems.

CHAPTER 8

8.APPENDIX

8.1 SOURCE CODE

```
# -*- coding: utf-8 -*-

"""amjath.ipynb
```

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1SNE07YLQ-9pevn6FxCsbgj4bCceJAS5e

"""

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import rcParams
from scipy import stats

data= pd.read_csv('/content/archive.zip')
data

data.info()

data.isnull().any()

data.isnull().sum()

data['Albumin_and_Globulin_Ratio'].fillna(data['Albumin_and_Globulin_Ratio'].mode()[0], inplace=True)

from sklearn.preprocessing import LabelEncoder
lc = LabelEncoder()
```

```
data['Gender']= lc.fit_transform(data['Gender'])


data.describe()


sns.distplot(data['Age'])
plt.title('Age Distribution Graph')
plt.show()


sns.countplot(data=data,x='Gender')


plt.figure(figsize=(10,7))
sns.heatmap(data.corr(),annot=True)


from sklearn.preprocessing import scale
X=data[['Age','Gender','Total_Bilirubin','Direct_Bilirubin','Alkaline_Phosphotase','Alamine_Aminotransferase','Aspartate_Aminotransferase','Total_Protiens','Albumin']]
X_scaled=pd.DataFrame(scale(X),columns=X.columns)
X_scaled.head()


X=data.iloc[:,:-1]
Y=data.Dataset


from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X_scaled,Y,test_size=0.2,random_state=42)


pip install imblearn
```

```
from imblearn.over_sampling import SMOTE
smote=SMOTE()
Y_train.value_counts()

X_train_smote,Y_train_smote=smote.fit_resample(X_train,Y_train)
Y_train_smote.value_counts()

from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
model1=RandomForestClassifier()
model1.fit(X_train_smote, Y_train_smote)
Y_predict=model1.predict(X_test)
rfc1=accuracy_score(Y_test,Y_predict)
rfc1
pd.crosstab(Y_test, Y_predict)
print(classification_report(Y_test,Y_predict))

from sklearn.tree import DecisionTreeClassifier
model4=DecisionTreeClassifier()
model4.fit(X_train_smote, Y_train_smote)
Y_predict=model4.predict(X_test)
dtc1=accuracy_score (Y_test,Y_predict)
dtc1
pd.crosstab(Y_test,Y_predict)
print(classification_report (Y_test,Y_predict))
```

```
from sklearn.neighbors import KNeighborsClassifier
model2=KNeighborsClassifier()
model2.fit(X_train_smote,Y_train_smote)
y_predict = model2.predict(X_test)
knn1=(accuracy_score (Y_test,Y_predict))
knn1
pd.crosstab(Y_test,Y_predict)
print(classification_report (Y_test, Y_predict))

from sklearn.linear_model import LogisticRegression
model5=LogisticRegression()
model5.fit(X_train_smote, Y_train_smote)
Y_predict=model5.predict(X_test)
logi1=accuracy_score(Y_test,Y_predict)
logi1
pd.crosstab(Y_test,Y_predict)
print(classification_report (Y_test, Y_predict))

print(X_train.shape)

import tensorflow.keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

#Initialising the ANN
classifier = Sequential()
```

```
# Adding the input Layer and the first hidden Layer
classifier.add(Dense(units=100, activation='relu', input_dim=10))

# Adding the second hidden layer

classifier.add(Dense (units=50, activation='relu'))

# Adding the output Layer
classifier.add(Dense (units=1, activation='sigmoid'))

#compiling the ANN
classifier.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accura
cy'])

#fitting the ANN to the training set
model_history=classifier.fit(X_train,Y_train,batch_size=100,validation_split=0.
2,epochs=100)

model4.predict([[50,1,1.2,0.8,150,70,80,7.2,3.4]])

model1.predict([[50,1,1.2,0.8,150,70,80,7.2,3.4]])

model2.predict([[50,1,1.2,0.8,150,70,80,7.2,3.4]])

model5.predict([[42,0,1,1.2,0.8,240,70,80,7.2]])

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)


Y_pred = (Y_pred > 0.5)
Y_pred


def predict_exit(sample_value):
# Convert list to numpy array
  sample_value = np.array (sample_value)
# Reshape because sample value contains only 1 record
  sample_value = sample_value.reshape(1, -1)
#Feature Scaling
  sample_value = scale(sample_value)
  return classifier.predict(sample_value)
#Age Gender Total Bilrubin Direct Bilrubin Alkaline Phosphotase
sample_value = [[50,1,1.2,0.8,150,70,80,7.2,3.4,0.8]]
if predict_exit (sample_value)>0.5:
  print('Prediction: Liver Patient')
else:
  print('Prediction: Healthy ')


acc_smote= [['KNN Classifier', knn1], ['RandomForestClassifier', rfc1],
['DecisionTreeClassifier', dtc1], ['Logistic Regression', logi1]]
Liverpatient_pred= pd.DataFrame(acc_smote, columns = ['classification
models', 'accuracy_score'])
Liverpatient_pred
```

```
plt.figure(figsize=(7,5))
plt.xticks(rotation=90)
plt.title('Classification models & accuracy scores after SMOTE', fontsize=18)
sns.barplot(x="classification models",y="accuracy_score",
data=Liverpatient_pred, palette ="Set2")

import pandas as pd

X = pd.DataFrame(X)
X = X.dropna()

import numpy as np

nan_mask = np.isnan(X)
X = X[~np.any(nan_mask, axis=1)]  # remove rows with NaN values

from sklearn.ensemble import ExtraTreesClassifier

model = ExtraTreesClassifier()
model.fit(X, Y)

model.feature_importances_

X = pd.DataFrame(X)  # Convert X to a pandas DataFrame
dd = pd.DataFrame(model.feature_importances_,
index=X.columns).sort_values(0, ascending=False)
dd
```

```
dd.plot(kind='barh', figsize=(7,6))
plt.title("FEATURE IMPORTANCE", fontsize=14)
```