

ASSIGNMENT DATE	12-SEPTEMBER-2022
STUDENT NAME	MADALA MANU VARDHAN
STUDENT ROLL NUMBER	110719104029
MAXIMUM MARKS	2 MARKS

ASSIGNMENT:1

IPYNB.JSON

import numpy as np

import pandas as pd

import seaborn as sns

from matplotlib import pyplot as plt

from matplotlib import pyplot as plt

import warnings

warnings.filterwarnings("ignore")

from sklearn.model_selection import train_test_split

from sklearn.model_selection import GridSearchCV

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import roc_curve, auc, roc_auc_score, confusion_matrix

from sklearn.metrics import accuracy_score

#Importing data set from local drive of system

Liver_data = pd.read_csv('E:\liver_disease_1.csv')

#Checking dimensions of Data Set

Liver_data.shape

#5 rows of data set

Liver_data.head()

def partition(x):

```
if x == 'No':
```

```
    return 0
```

```
return 1
```

```
Liver_data['Dataset'] = Liver_data['Dataset'].map(partition)
```

```
# Plot histogram grid
```

```
Liver_data.hist(figsize=(15,15), xrot=-45, bins=10) ## Display the labels  
rotated by 45 degrees
```

```
# Clear the text "residue"
```

```
plt.show()
```

```
#Calculating Statics of dataset
```

```
Liver_data.describe()
```

```
#calculating stats for object datatype
```

```
#Liver_data.describe(include=['object'])
```

```
#checking correlation among the variables
```

```
Liver_data.corr()
```

```
#Drawing heatmap
```

```
plt.figure(figsize=(10,10))
```

```
sns.heatmap(Liver_data.corr())
```

```
#Data Cleaning
```

```
#Removing duplicates
```

```
Liver_data = Liver_data.drop_duplicates()
```

```
print( Liver_data.shape )
```

```
#Removing Outliers
```

```
sns.boxplot(Liver_data.Aspartate_Aminotransferase)
```

```
Liver_data.Aspartate_Aminotransferase.sort_values(ascending=False).head()
```

```
Liver_data = Liver_data[Liver_data.Aspartate_Aminotransferase <=2500 ]
```

```
Liver_data.isnull().values.any()
```

```
Liver_data=Liver_data.dropna(how='any')
```

```
Liver_data.shape
```

```
Liver_data.head()
```

```
#Data Standardization
```

```
Y = Liver_data.Dataset
```

```
X = Liver_data.drop('Dataset', axis=1)
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X,  
Y,test_size=0.2,random_state=1234,stratify=Liver_data.Dataset)
```

```
train_mean = X_train.mean()
```

```
train_std = X_train.std()
```

```
X_train = (X_train - train_mean) / train_std
```

```
X_train.describe()
```

```
X_test = (X_test - train_mean) / train_std
```

```
X_test.describe()
```

```
#Logistic Regression
```

```
tuned_params = {'C': [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000],  
'penalty': ['l1', 'l2']}
```

```
model = GridSearchCV(LogisticRegression(), tuned_params, scoring =  
'roc_auc', n_jobs=-1)
```

```
model.fit(X_train, Y_train)
```

```
#Predicting model Train and Test results
```

```
Y_train_pred = model.predict(X_train)
```

```
Y_pred = model.predict(X_test)
```

```
Y_pred = model.predict(X_test)
```

```
#Calculating probabilities
```

```
Y_pred_proba = model.predict_proba(X_test)[:,-1]
```

```
Y_pred_proba[:10]
```

```
#Calculating Confusion Matrix
```

```
confusion_matrix(Y_test, Y_pred).T
```

```
# Calculate ROC curve from y_test and pred
```

```
fpr, tpr, thresholds = roc_curve(Y_test, Y_pred_proba)
```

```
# Calculate AUC for Train set
```

```
print('Logistic Regression Training Score: \n',roc_auc_score(Y_train,  
Y_train_pred))
```

```

#Calculating accuracy
print('Logistic Regression Accuracy: \n', accuracy_score(Y_test,Y_pred))

#Gaussian Naives Bayes Implementation

from sklearn.naive_bayes import GaussianNB
gaussian = GaussianNB()
gaussian.fit(X_train, Y_train)
gauss_predicted = gaussian.predict(X_test)

print('Guassian Naives Bayes Traning Score:
\n',gaussian.score(X_train,Y_train))

print('Guassian Naives Bayes Test Score: \n',gaussian.score(X_test,Y_test))

print('Guassian Naives Bayes Accuracy: \n',
accuracy_score(Y_test,gauss_predicted))

#Bernoulli Naives Bayes Implemetation

from sklearn.naive_bayes import BernoulliNB

Bernoulli = BernoulliNB(binarize=0.0)
Bernoulli.fit(X_train,Y_train)
Bernoulli_predicted = Bernoulli.predict(X_test)

print('Bernoulli Naives Bayes Traning Score:
\n',Bernoulli.score(X_train,Y_train))

print('Bernoulli Naive Bayes Test Score: \n',Bernoulli.score(X_test,Y_test))

print('Bernoulli Naive Bayes Accuracy: \n',
accuracy_score(Y_test,Bernoulli_predicted))

#Model Comparision

```

```
model_performance = [['Logistic Regression  
Accuracy',accuracy_score(Y_test,Y_pred)*100],  
                     ['Guassian Naives Bayes  
Accuracy',accuracy_score(Y_test,gauss_predicted)*100],  
                     ['Bernoulli Naives  
Bayes',accuracy_score(Y_test,Bernoulli_predicted)*100]]
```

```
model_predict_df = pd.DataFrame(model_performance,columns =  
['Model','%Accuracy'])
```

```
model_predict_df
```