Problem Statement

Liver Disease Prediction by using PyCaret – Machine Learning Project

Patients with Liver disease have been continuously increasing because of excessive consumption of alcohol, inhale of harmful gases, intake of contaminated food, pickles and drugs. This dataset was used to evaluate prediction algorithms in an effort to reduce burden on doctors.

Details

This data set contains 416 liver patient records and 167 non liver patient records collected from North East of Andhra Pradesh, India. The "Dataset" column is a class label used to divide groups into liver patient (liver disease) or not (no disease). This data set contains 441 male patient records and 142 female patient records.

Any patient whose age exceeded 89 is listed as being of age "90".

Columns:

- Age of the patient
- Gender of the patient
- Total Bilirubin
- Direct Bilirubin
- Alkaline Phosphotase
- Alamine Aminotransferase
- Aspartate Aminotransferase
- Total Protiens
- Albumin
- Albumin and Globulin Ratio
- Dataset: field used to split the data into two sets (patient with liver disease, or no disease)

Objective:

The primary goal is to visualize the data and establish trends or important characteristics, if any. The next important objective is to create a pycaret model that can predict the class of the patient by the virtue of its input.

What is PyCaret and Why Should you Use it?

PyCaret is an open-source, machine learning library in Python that helps you from data preparation to model deployment.

It is easy to use and you can do almost every data science project task with just one line of code.

Task to Do:

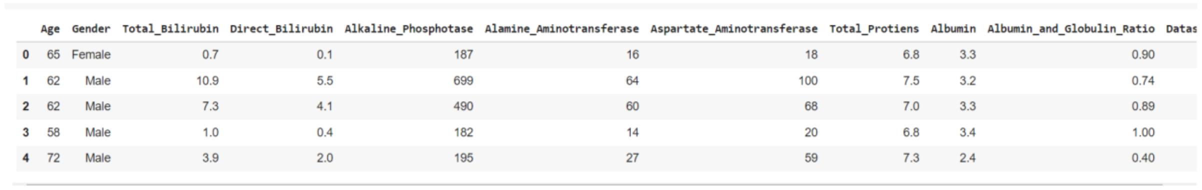
1. Installing PyCaret on your machine and import the dataset

```
#Make necessary imports
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
from pycaret.datasets import get_data

dataset = pd.read_csv("indian_liver_patient.csv")
dataset.head()
```

2. Let's start by reading the dataset using the Pandas library:

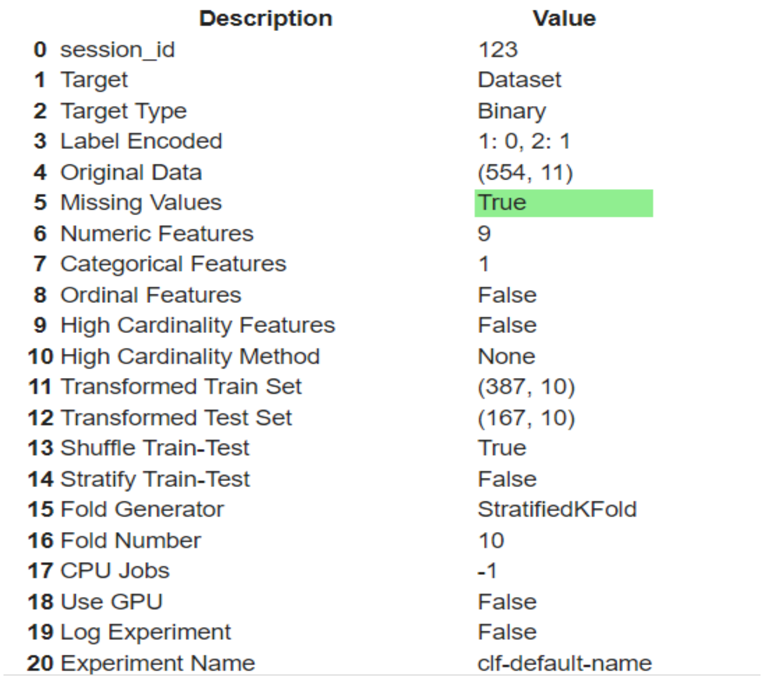


	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albumin_and_Globulin_Ratio	Datas
0	65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	0.89	
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	1.00	
4	72	Male	3.9	2.0	195	27	59	7.3	2.4	0.40	

3. The very first step before we start our machine learning project in PyCaret is to set up the environment. It's just a two-step process:

- **Importing a Module**: Depending upon the type of problem you are going to solve, you first need to import the module. In the first version of PyCaret, 6 different modules are available – regression, classification, clustering, natural language processing (NLP), anomaly detection, and associate mining rule
- **Initializing the Setup**: In this step, PyCaret performs some basic preprocessing tasks, like ignoring the IDs and Date Columns, imputing the missing values, encoding the categorical variables, and splitting the dataset into the train-test split for the rest of the modeling steps. When you run the setup function, it will first confirm the data types, and then if you press enter, it will create the environment for you to go ahead

4. Now setting environment in Pycaret by using classification module



	Description	Value
0	session_id	123
1	Target	Dataset
2	Target Type	Binary
3	Label Encoded	1: 0, 2: 1
4	Original Data	(554, 11)
5	Missing Values	True
6	Numeric Features	9
7	Categorical Features	1
8	Ordinal Features	False
9	High Cardinality Features	False
10	High Cardinality Method	None
11	Transformed Train Set	(387, 10)
12	Transformed Test Set	(167, 10)
13	Shuffle Train-Test	True
14	Stratify Train-Test	False
15	Fold Generator	StratifiedKFold
16	Fold Number	10
17	CPU Jobs	-1
18	Use GPU	False
19	Log Experiment	False
20	Experiment Name	clf-default-name

PyCaret by default imputes the missing value in the dataset by 'mean' for numeric features and 'constant' for categorical features.

To change the imputation method, numeric_imputation and categorical_imputation parameters can be used within setup.

Here's q quick reminder of the evaluation metrics used for supervised learning:

- Classification: Accuracy, AUC, Recall, Precision, F1, Kappa
- Regression: MAE, MSE, RMSE, R2, RMSLE, MAPE

5. **Training our machine learning model using PyCaret:**

Training a model in PyCaret is quite simple. You just need to use the *create_model* function that takes just the one parameter – the model abbreviation as a string. Here, we are going to first train a LogisticRegression model

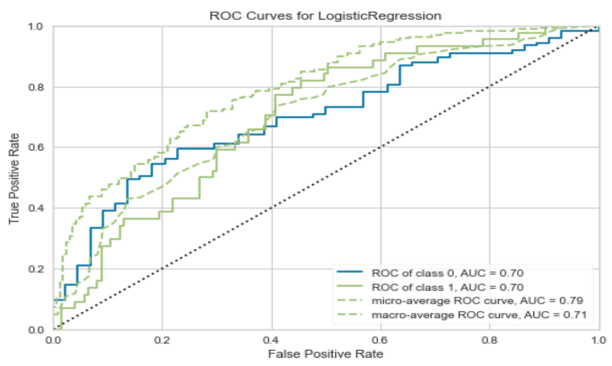
6. **Hyperparameter Tuning:**

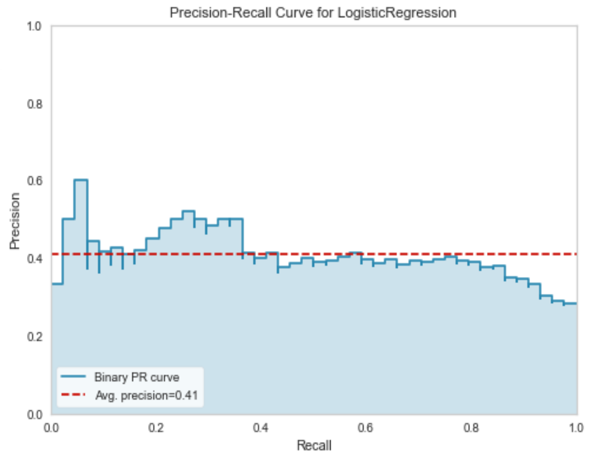
We can tune the hyperparameters of a machine learning model by just using the tune_model function which takes one parameter – the model abbreviation string (the same as we used in the *create_model* function).

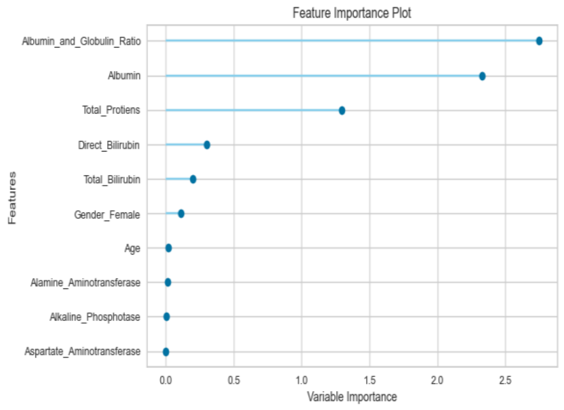
PyCaret provides us a lot of flexibility. For example, we can define the number of folds using the *fold* parameter within the tune_model function. Or we can change the number of iterations using the *n_iter* parameter. Increasing the *n_iter* parameter will obviously increase the training time but will give a much better performance.

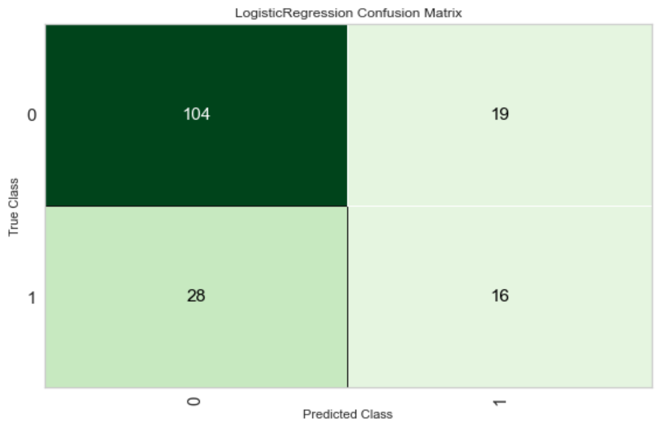
7. **Plotting Metrix:**

Classification: Accuracy, AUC, Recall, Precision, F1, Kappa



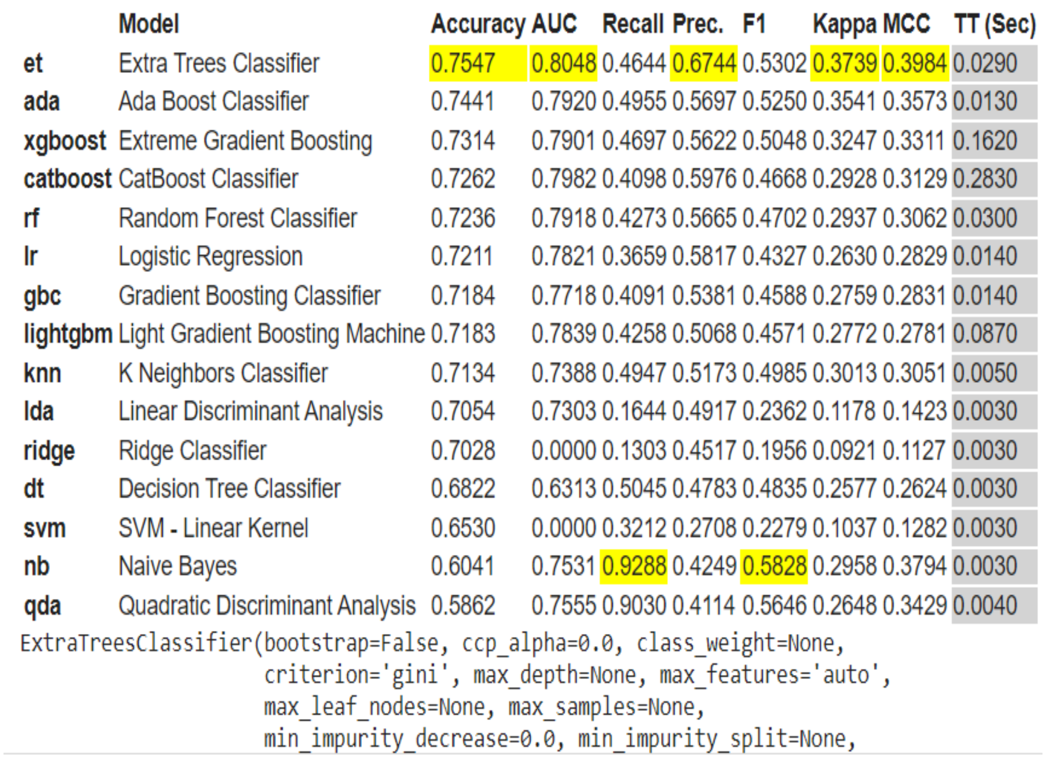






	0	1
0	104	19
1	28	16

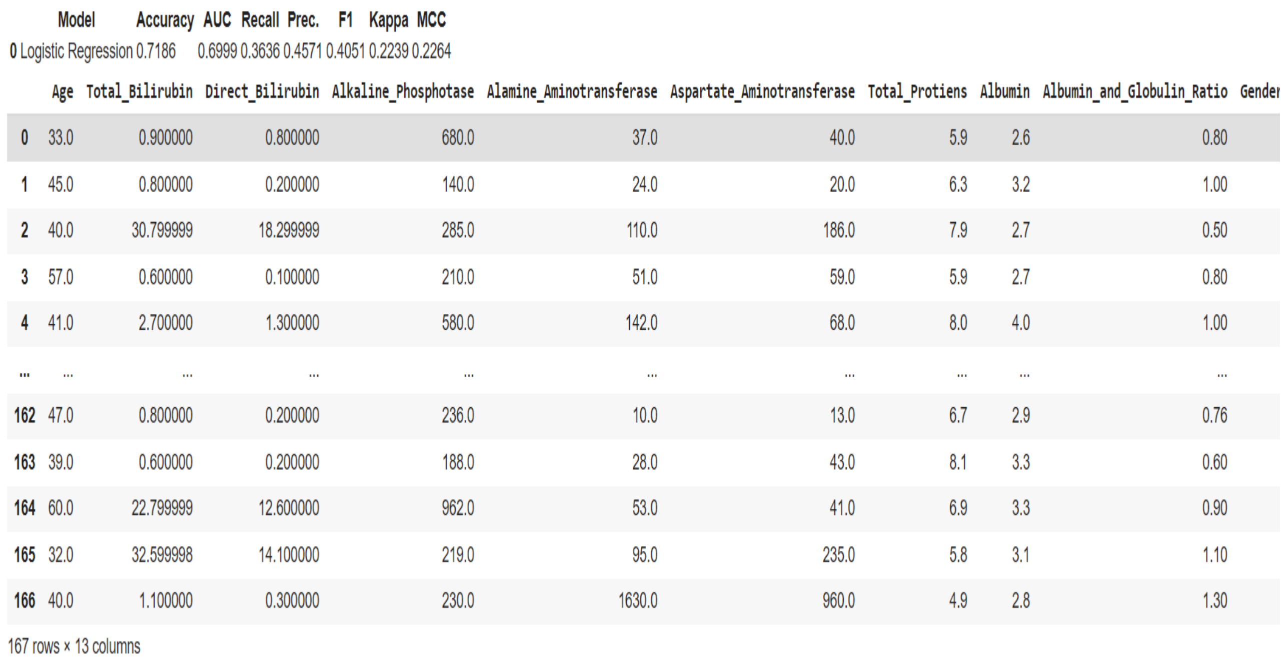
8. Compare model:



	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
et	Extra Trees Classifier	0.7547	0.8048	0.4644	0.6744	0.5302	0.3739	0.3984	0.0290
ada	Ada Boost Classifier	0.7441	0.7920	0.4955	0.5697	0.5250	0.3541	0.3573	0.0130
xgboost	Extreme Gradient Boosting	0.7314	0.7901	0.4697	0.5622	0.5048	0.3247	0.3311	0.1620
catboost	CatBoost Classifier	0.7262	0.7982	0.4098	0.5976	0.4668	0.2928	0.3129	0.2830
rf	Random Forest Classifier	0.7236	0.7918	0.4273	0.5665	0.4702	0.2937	0.3062	0.0300
lr	Logistic Regression	0.7211	0.7821	0.3659	0.5817	0.4327	0.2630	0.2829	0.0140
gbc	Gradient Boosting Classifier	0.7184	0.7718	0.4091	0.5381	0.4588	0.2759	0.2831	0.0140
lightgbm	Light Gradient Boosting Machine	0.7183	0.7839	0.4258	0.5068	0.4571	0.2772	0.2781	0.0870
knn	K Neighbors Classifier	0.7134	0.7388	0.4947	0.5173	0.4985	0.3013	0.3051	0.0050
lda	Linear Discriminant Analysis	0.7054	0.7303	0.1644	0.4917	0.2362	0.1178	0.1423	0.0030
ridge	Ridge Classifier	0.7028	0.0000	0.1303	0.4517	0.1956	0.0921	0.1127	0.0030
dt	Decision Tree Classifier	0.6822	0.6313	0.5045	0.4783	0.4835	0.2577	0.2624	0.0030
svm	SVM - Linear Kernel	0.6530	0.0000	0.3212	0.2708	0.2279	0.1037	0.1282	0.0030
nb	Naive Bayes	0.6041	0.7531	0.9288	0.4249	0.5828	0.2958	0.3794	0.0030
qda	Quadratic Discriminant Analysis	0.5862	0.7555	0.9030	0.4114	0.5646	0.2648	0.3429	0.0040

```
ExtraTreesClassifier(bootstrap=False, ccp_alpha=0.0, class_weight=None,
                     criterion='gini', max_depth=None, max_features='auto',
                     max_leaf_nodes=None, max_samples=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
```

9. Predict the model:



	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Logistic Regression	0.7186	0.6999	0.3636	0.4571	0.4051	0.2239	0.2264

	Age	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albumin_and_Globulin_Ratio	Gender
0	33.0	0.900000	0.800000	680.0	37.0	40.0	5.9	2.6	0.80	
1	45.0	0.800000	0.200000	140.0	24.0	20.0	6.3	3.2	1.00	
2	40.0	30.799999	18.299999	285.0	110.0	186.0	7.9	2.7	0.50	
3	57.0	0.600000	0.100000	210.0	51.0	59.0	5.9	2.7	0.80	
4	41.0	2.700000	1.300000	580.0	142.0	68.0	8.0	4.0	1.00	
...	...	...	...	...	...	...	...	...	...	
162	47.0	0.800000	0.200000	236.0	10.0	13.0	6.7	2.9	0.76	
163	39.0	0.600000	0.200000	188.0	28.0	43.0	8.1	3.3	0.60	
164	60.0	22.799999	12.600000	962.0	53.0	41.0	6.9	3.3	0.90	
165	32.0	32.599998	14.100000	219.0	95.0	235.0	5.8	3.1	1.10	
166	40.0	1.100000	0.300000	230.0	1630.0	960.0	4.9	2.8	1.30	

167 rows × 13 columns