**MLOps Assignment - 2**

**Group 18:**

**Members –**

· Aditya Gaurav – 2022AC05101

· Sakthivel G – 2022AC05688

· Kalra Geetansh – 2022AC05174

· Bhuvanjeet Singh Gandhi – 2022AC05606

**Files**:

Repository to track the code - <https://github.com/Bhuvanjeet/MLOps-Assignment-2->

Dataset - liver\_disease\_1.csv

Part 1,2,3 - MLOps\_assignment2\_Group18.ipynb , MLOps\_assignment2\_Group18.html, MLOps\_assignment2\_Group18.mp4

EDA Report - Liver\_disease\_dataset\_eda\_report.html

Lime explanation - lime\_explanation\_0.html

Part 4 - aws\_lambda.py, model\_deployment\_using\_aws.mp4

**1. Data Collection and Preprocessing:**

**Liver Disease Dataset** is used - <https://www.kaggle.com/datasets/uciml/indian-liver-patient-records>

The 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)

Packages

* **pandas:** For data manipulation and analysis.
* **autosklearn:** Automated machine learning library used for building and training models.
* **dataprep.eda:** Used for generating exploratory data analysis (EDA) reports.
* **sklearn.preprocessing:** Provides tools like StandardScaler for feature scaling.
* **sklearn.model\_selection:** Used for splitting the dataset into training and test sets.
* **pickle:** For saving (dumping) and loading models or other objects.
* **lime and shap:** XAI tools used for model interpretability, helping us explain the model’s predictions.
* **NumPy:** For numerical operations

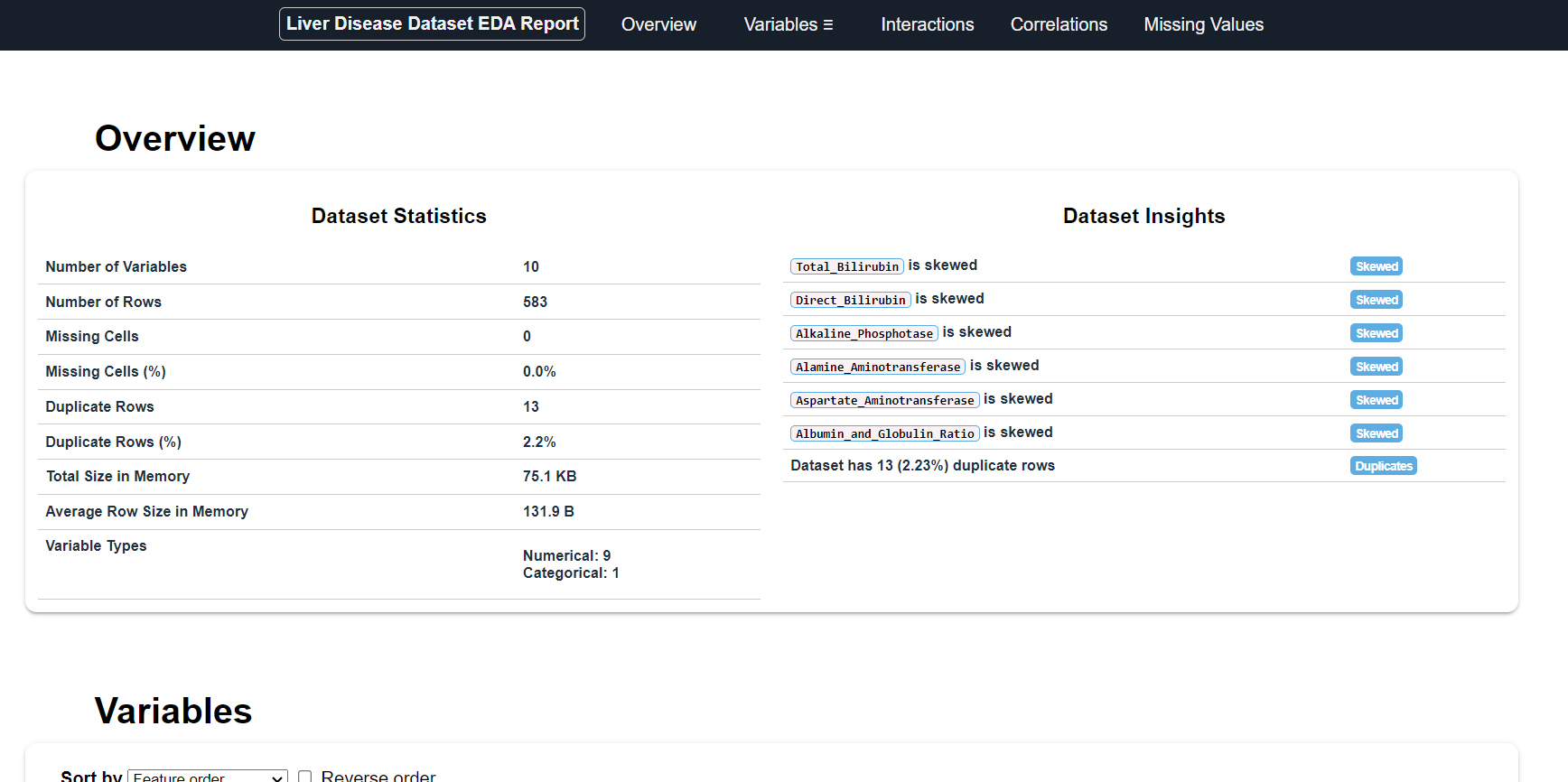
Tools specifically used for Data Preparation and EDA:

* **dataprep:** Used for data preparation and exploratory data analysis.
* **shap:** A tool for model interpretability, providing global and local explanations of model predictions.
* **lime:** Another model interpretability tool that offers local explanations for individual predictions.

We are using the **auto-sklearn Docker image** from the official site to run this notebook, which already includes most of the dependencies on **auto-sklearn**.

**Steps:**

1. **Load the Dataset:** The liver disease dataset is loaded into a pandas DataFrame (df) from a CSV file (liver\_disease\_1.csv).
2. **Exploratory Data Analysis (EDA):**
   * The create\_report() function from dataprep.eda is used to generate an automated exploratory data analysis report for the dataset.
   * This report provides detailed insights into the data, including statistics, distributions, and relationships between features.
3. **Saving the EDA Report:** • The report is saved as an HTML file (***Liver\_disease\_dataset\_eda\_report.html***) for further review and analysis.



**Code Snippet:**

from **dataprep.eda** import **create\_report**

report **=** **create\_report**(df, title**=**'Liver Disease Dataset EDA Report')

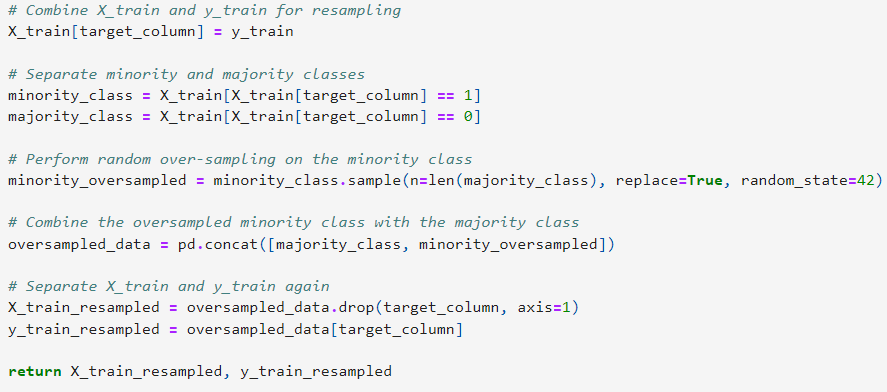
**report.save**('Liver\_disease\_dataset\_eda\_report.html')

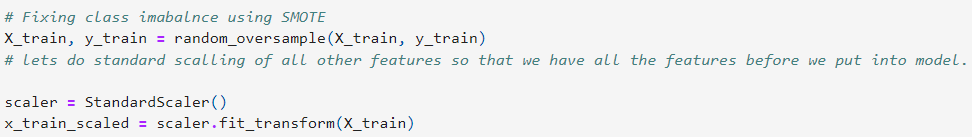
## **Data Preprocessing and Oversampling for Class Imbalance:**

1. **Random Oversampling Function:**
   * The random\_oversample() function performs random oversampling on the minority class in the training data to address class imbalance. It duplicates the minority class instances to match the size of the majority class, helping to create a balanced dataset.
2. **Handling Missing Values:**
   * Missing values in the Albumin\_and\_Globulin\_Ratio column are forward-filled (ffill), ensuring that no null values remain in the dataset.
3. **Data Splitting:**
   * The dataset is split into features (X) and target labels (y), where the target label indicates whether the patient has liver disease (1 for “Yes”, 0 for “No”).
   * The dataset is split into training and test sets using train\_test\_split(), with 20% of the data reserved for testing. Stratified sampling is used to maintain the proportion of liver disease cases in both training and test sets, preventing any potential data leakage.
4. **Class Imbalance Handling:**
   * Random oversampling is applied to the training data using the random\_oversample() function, which helps mitigate the imbalance between liver disease and non-liver disease cases.
5. **Feature Scaling:**
   * Standard scaling is applied to the training features using StandardScaler(), ensuring that all features are on a similar scale, which improves the model’s performance.
   * The scaled data is saved to a new DataFrame X\_train\_final.
6. **Saving the Scaler:**
   * The fitted scaler is saved as a pickle file (scaler.pkl) to ensure that the same scaling can be applied to the test data later.

This preprocessing step is crucial to ensure the model is trained on balanced and standardized data, minimizing bias and improving generalization.

**Code Snippet**:





**2. Model Selection, Training, and Hyperparameter Tuning**

The goal of this task was to train multiple models, tune hyperparameters, and select the best-performing model for predicting liver disease.

To automate the model selection and hyperparameter tuning process, we used AutoML via the Auto-sklearn library, which automates the entire workflow, including data preprocessing, model training, and hyperparameter optimization.

## **Experimentation Process:**

1. Auto-sklearn Setup:
   * Auto-sklearn was configured with a total time limit of 180 seconds for the entire task and a 40-second time limit per run. This allowed Auto-sklearn to test various models and configurations within a defined time frame.
   * Auto-sklearn automatically handled tasks like data preprocessing (including feature scaling), model selection, and hyperparameter tuning across various algorithms.
2. Generated Models:
   * Auto-sklearn trained several models, including Random Forest, AdaBoost, Extra Trees, Passive-Aggressive Classifier, MLPClassifier, and Linear Discriminant Analysis (LDA).
   * Each model was tested with different hyperparameters and preprocessing techniques to optimize performance.
   * The top-performing models were ranked based on their cost (lower is better) and ensemble weight (how important they are in the final ensemble model).
3. Top Models:
   * **RandomForestClassifier:** Ranked 1st, it achieved the lowest cost and was included in the final ensemble with an ensemble weight of 0.02.
   * **AdaBoostClassifier:** Featured multiple times in the top 5 models, with different hyperparameters and base estimators.
   * **ExtraTreesClassifier:** Appeared frequently in the top models list, with varying hyperparameter settings.
   * **PassiveAggressiveClassifier:** Ranked 3rd, contributing to the ensemble model.
4. Ensemble Model:
   * Auto-sklearn automatically created an ensemble of the best models based on their performance. The ensemble model combined predictions from multiple models (like RandomForest and AdaBoost) to create a more robust final prediction.

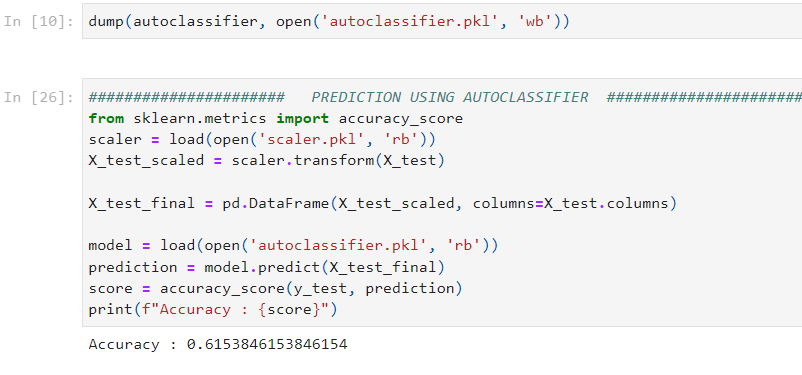
## **Justification for Model Choice:**

* **RandomForestClassifier** was chosen as the best-performing model based on its rank (1st), low cost (0.18), and inclusion in the final ensemble. Random Forest is well-known for its strong performance in classification tasks and its robustness to overfitting, making it a suitable choice for our liver disease dataset.
* **AdaBoostClassifier** also performed well and was included in the final ensemble with different configurations. This demonstrates that boosting methods can effectively enhance model performance by focusing on difficult-to-classify instances.
* **ExtraTreesClassifier** appeared frequently in the top models list, reinforcing its importance in the ensemble. Extra Trees is known for its ability to handle high-dimensional data, which may have contributed to its success.

By using **Auto-sklearn**, we were able to **automate the model training and hyperparameter tuning process, testing a wide range of algorithms and configurations within a limited time frame.** The final ensemble model, consisting of RandomForest, AdaBoost, Extra Trees, and other classifiers, provides a well-balanced approach to predicting liver disease. The ability to automatically select and combine models ensures robust performance and minimizes overfitting.

**Code Snippet**:



****

**3. Explainable AI (XAI) Implementation**

## **3.1 SHAP Summary Plot Explanation for Liver Disease Prediction:**

The **SHAP** (Shapley Additive Explanations) summary plot shows how the features in the liver disease dataset contribute to the model’s predictions for the entire test data. Each dot represents a SHAP value for a specific feature and patient, and the color of the dot (ranging from blue to red) represents the feature’s value (low to high). The SHAP values on the X-axis indicate whether a feature increases or decreases the likelihood of predicting liver disease.

### **3.1.1 Key Insights:**

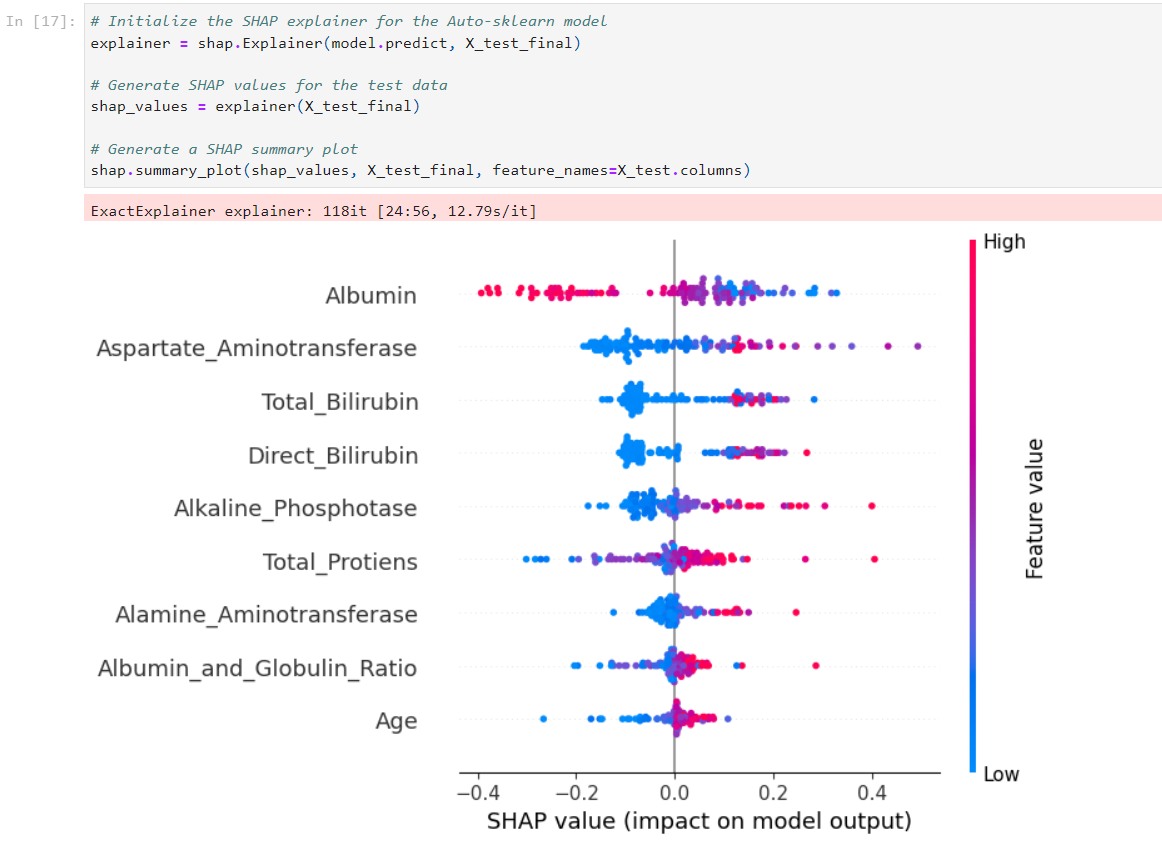
* **Albumin:** High Albumin values (red) contribute negatively to the prediction of liver disease, meaning that higher levels of albumin reduce the likelihood of liver disease (negative SHAP values). Conversely, lower values (blue) increase the probability of liver disease.
* **Aspartate\_Aminotransferase (AST):** Higher values (red) increase the likelihood of liver disease, as indicated by the positive SHAP values. AST is an enzyme linked to liver function, and elevated levels suggest liver damage, increasing the likelihood of the disease.
* **Total\_Bilirubin and Direct\_Bilirubin:** Both these bilirubin levels show that higher values (red) contribute positively to liver disease predictions, indicating that higher bilirubin levels strongly increase the probability of liver dysfunction.
* **Alkaline\_Phosphatase:** Higher values (red) of this enzyme also push the prediction towards liver disease, while lower values (blue) reduce the likelihood of the disease.
* **Total\_Proteins:** Lower levels (blue) of total proteins increase the likelihood of liver disease, while higher levels (red) reduce it, as higher protein levels generally indicate healthier liver function.
* **Age:** Older age (red) contributes positively to predicting liver disease, while younger age (blue) lowers the probability. This aligns with the increased risk of liver disease as age progresses.
* **Albumin\_and\_Globulin\_Ratio:** A higher ratio (red) tends to decrease the likelihood of liver disease, while lower ratios (blue) contribute positively to predicting liver disease.

### **3.1.2 Overall Interpretation:**

* High values of certain features like **Aspartate\_Aminotransferase**, **Bilirubin**, **Alkaline\_Phosphatase**, and **Age** are strong indicators that push the model towards predicting liver disease.
* Low values of features like **Albumin**, **Total\_Proteins**, and **Albumin\_and\_Globulin\_Ratio** push the model towards predicting liver disease as well.

This SHAP summary plot provides a comprehensive view of how each feature in the liver disease dataset impacts the model’s predictions for the test data. By analyzing the SHAP values and feature contributions, we can better understand the factors driving the model’s predictions.

**Code Snippet:**



## **3.2 LIME Explanation for Liver Disease Prediction:**

The **LIME** (Local Interpretable Model-agnostic Explanations) output provides an explanation for an individual prediction made by the model regarding the presence or absence of liver disease. This explanation helps us understand which features contributed most to the model’s decision for this particular patient.

## **3.2.1 Key Components:**

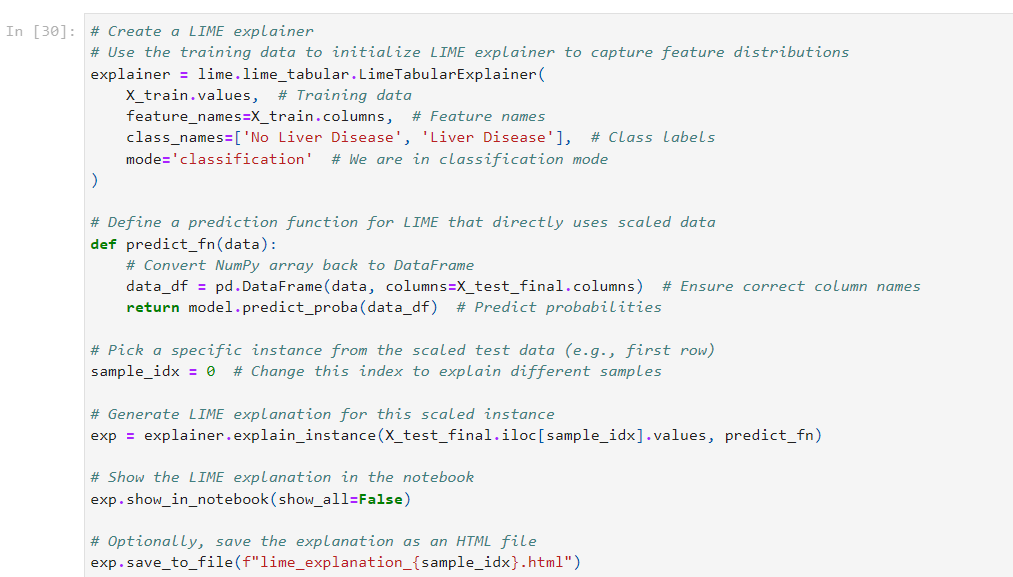
1. **Prediction Probabilities:**
   * The model predicts a 56% probability for “No Liver Disease” and a 44% probability for “Liver Disease”. This indicates that the model slightly favors the prediction of “No Liver Disease” for this instance.
2. **Features Contributing to “No Liver Disease”:**
   * **Total\_Proteins:** With a value of 5.80, this feature strongly contributes to the prediction of “No Liver Disease” (1.92 impact).
   * **Albumin:** A value of 2.60 positively impacts the prediction of “No Liver Disease” (1.42 impact).
   * \*\*Albumin\_and\_Globulin\_Ratio: A small positive contribution (0.11 impact), suggesting this feature slightly pushes the prediction towards “No Liver Disease”.
   * **Age:** The age of the patient (32.00) also contributes positively to predicting “No Liver Disease”, but with a minimal impact (0.04).
3. Features Contributing to “Liver Disease”:
   * **Aspartate\_Aminotransferase (AST):** This feature, with a value of -0.20, slightly increases the likelihood of liver disease.
   * **Alamine\_Aminotransferase:** A value of -0.18 also contributes towards predicting “Liver Disease”.
   * **Direct\_Bilirubin and Total\_Bilirubin:** These two bilirubin levels (-0.39 and -0.38 respectively) indicate an elevated likelihood of liver disease, suggesting some liver dysfunction.
   * **Alkaline\_Phosphotase:** A value of -0.17 contributes to the prediction of liver disease.

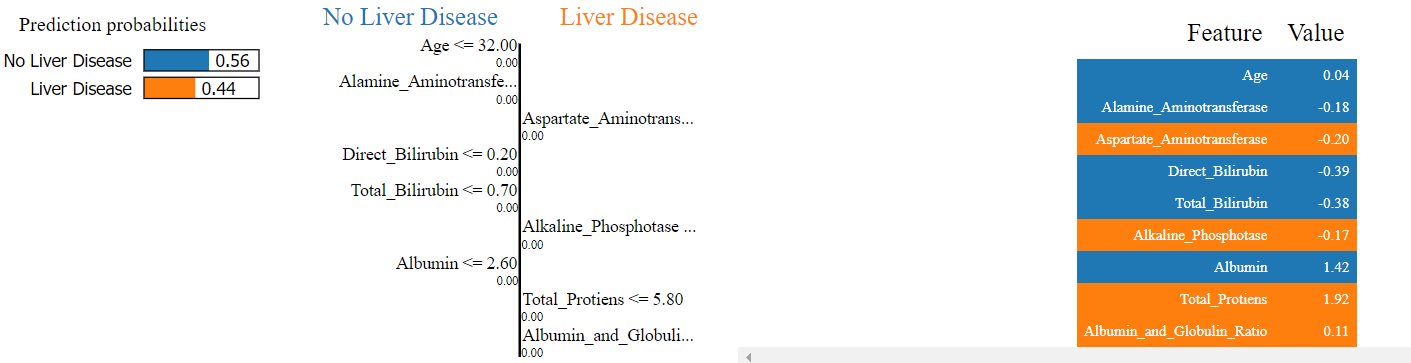
## **3.2.2 Overall Interpretation:**

* The model’s final decision slightly favors “No Liver Disease” based on a combination of feature contributions. Features like Total\_Proteins, Albumin, and Albumin\_and\_Globulin\_Ratio strongly push the prediction towards “No Liver Disease”, while liver enzyme levels (such as AST and Bilirubin) provide opposing contributions, increasing the likelihood of “Liver Disease”. However, the positive impact of protein-related features outweighs the negative impact of the liver enzyme levels in this case.

This LIME explanation offers clear insights into why the model predicted “No Liver Disease” for this patient by highlighting the relative importance of specific features.

**Code Snippet:**





## **3.3 Insights into the Model’s Decision-Making Process:**

Through this exercise using **SHAP** and **LIME**, we were able to gain clear insights into the decision-making process of the liver disease prediction model. Both **XAI** (Explainable AI) tools provided a deep understanding of how specific features contributed to the model’s output for individual predictions as well as the overall feature importance.

### **3.3.1 Feature Importance and Model Interpretability:**

1. SHAP (Shapley Additive Explanations):
   * **Global Insights:** SHAP provided a holistic view of how features like Aspartate\_Aminotransferase, Bilirubin, and Alkaline\_Phosphotase contribute to the model’s predictions across the entire test dataset. The summary plot showed that high values of these features pushed the model toward predicting liver disease, while features like Albumin and Total\_Proteins reduced the likelihood of the disease.
   * **Local Insights:** SHAP also provided explanations for individual patients, showing how each feature’s value either positively or negatively influenced the model’s prediction. This level of interpretability is important in understanding the role of different liver function indicators for each patient.
2. LIME (Local Interpretable Model-Agnostic Explanations):
   * **Instance-Level Interpretability:** LIME helped explain why the model made specific predictions for individual patients. By breaking down each prediction and showing how the combination of features like Albumin, Total\_Proteins, and Bilirubin influenced the model’s output, LIME made the decision-making process clear at a local (instance) level.
   * **Clear Decision Boundaries:** The LIME output also visually highlighted which features pushed the prediction towards “Liver Disease” and which ones leaned towards “No Liver Disease”, helping us understand the balance of contributing factors for each decision.

### **3.3.2 Importance of Interpretability:**

Interpretability is crucial in medical contexts like liver disease prediction because it ensures that predictions made by the model can be trusted and validated. Stakeholders such as **doctors** and **healthcare professionals** need to understand why a model predicted a certain outcome before making any decisions based on it. By using XAI tools like SHAP and LIME, we provide this necessary transparency, offering:

1. **Trust and Reliability:** Knowing which features the model considers important (e.g., liver enzyme levels or proteins) ensures that the model aligns with clinical expectations and that its predictions are reliable.
2. **Actionability:** Interpretability allows healthcare professionals to make informed decisions based on the model’s predictions. If a feature such as Bilirubin strongly indicates liver disease, doctors can prioritize further tests based on this.
3. **Error Analysis:** These tools also help identify potential biases or errors in the model’s decision-making process. If a particular feature is overly influencing predictions in an unexpected way, we can adjust the model accordingly.

### **3.3.3 How XAI Tools Helped Achieve Interpretability:**

1. SHAP provided a clear picture of how each feature affects the prediction globally and locally, offering a balanced perspective of both general trends and specific cases.
2. LIME allowed us to drill down into individual predictions, making the model’s decision transparent for each test case. This is especially useful for understanding outliers or edge cases in the dataset.

Both tools complement each other in helping us understand and trust the model, ensuring that the predictions align with real-world medical knowledge, and providing actionable insights to healthcare professionals.

**4. Model Deployment Using Cloud Services**

1. Made a zip file of all the pkl files being used and also created a function which is to predict using the model pkl from the input we will receive.
2. Headed towards the **AWS console** and selected **Lambda**.
3. Created a new function, selected python 3.8 as config and added a layer of **sklearn** from here: <https://github.com/model-zoo/scikit-learn-lambda/blob/master/layers.csv>
4. Once this was done, Added our zip file to the Lambda and tested it out giving an input:

{

"features": [

0.04367624,

-0.38178958,

-0.39272429,

-0.16585637,

-0.18299239,

-0.19847158,

1.91957338,

1.41811513,

0.11146485

]

}

1. It was working fine.
2. Headed towards **API gateway**, created a **RestAPI** and added the lambda function we created.
3. Selected **POST** method and successfully tested it out.
4. Then create the api and deployed it.
5. Used the **Invoke url** we received: <https://imxvvubd1g.execute-api.us-east-1.amazonaws.com/Dev>
6. **In postman, selected POST method, added this url there and from Body--> raw--> added the same input feature as step4.**
7. Successfully received a **response**.

The above steps are detailed in the attached video - **model\_deployment\_using\_aws.mp4**