**DocAssist (Building Intelligent Medical Decision Support System)**

***Problem Statement***

The objective of this project is to develop an intelligent medical decision support system that analyzes patient data to assist doctors in making informed decisions about the best treatment options for individual patients. By leveraging machine learning and data analysis, the system will provide personalized treatment recommendations based on the patient's medical history, symptoms, lab results, and other relevant factors.

***Data Collection and Preprocessing pipeline***

* Imported the necessary Python libraries, including **pandas**, **numpy**, **matplotlib**, and **seaborn**.
* Loaded the primary dataset provided by KnowledgeHut and assigned it to a variable named **df**.
* Imported an additional dataset containing patient information and named it **pat\_details**.
* Merged the two datasets based on a common key and created a new column named **Patient Name** to include patient details in the final dataset.
* Validated the dataset for any missing or null values to ensure data completeness.
* Checked for duplicate rows in the dataset to eliminate redundancy.
* Reviewed the data types of all columns and examined the statistical properties of the dataset.
* Renamed the **Output** column to **Treatment Recommendation** for improved readability and relevance

***EDA and Feature Engineering***

* Updated the values in the **Output** column to **‘Healthy’** and **‘Diseased’**, ensuring the labels are more intuitive and user-friendly for analysis.
* Conducted **Exploratory Data Analysis (EDA)** and implemented **Feature Engineering techniques** to enhance the quality and insights derived from various input columns.
* HAEMATOCRIT
* Reduced the dataset size from **314 rows to 45 rows** by rounding decimal values, optimizing the dataset for faster model execution.
* Conducted **Univariate** and **Bivariate analysis**, using various plotting techniques to compare input features with the **Output** column.
* HAEMOGLOBINS
* Reduced the dataset size from **128 rows to 16 rows** by rounding decimal values, optimizing the dataset for faster model execution.
* Conducted **Univariate** and **Bivariate analysis**, using various plotting techniques to compare input features with the **Output** column.
* ERYTHROCYTE
* Reduced the dataset size from **406 rows to 8 rows** by rounding decimal values, optimizing the dataset for faster model execution.
* Conducted **Univariate** and **Bivariate analysis**, using various plotting techniques to compare input features with the **Output** column.
* LEUCOCYTE
* Reduced the dataset size from **255 rows to 40 rows** by rounding decimal values, optimizing the dataset for faster model execution.
* Conducted **Univariate** and **Bivariate analysis**, using various plotting techniques to compare input features with the **Output** column.
* THROMBOCYTE
* Validated the dataset and confirmed that there were no decimal values in the data. As a result, the column data was kept intact for model training.
* Conducted **Univariate** and **Bivariate analysis**, utilizing various plotting techniques to compare input features with the **Output** column.
* MCH
* Reduced the dataset size from **177 rows to 25 rows** by rounding decimal values, optimizing the dataset for faster model execution.
* Conducted **Univariate** and **Bivariate analysis**, using various plotting techniques to compare input features with the **Output** column.
* MCHC
* Validated the dataset and confirmed that there were no decimal values in the data. As a result, the column data was kept intact for model training.
* Conducted **Univariate** and **Bivariate analysis**, utilizing various plotting techniques to compare input features with the **Output** column.
* MCV
* Reduced the dataset size from **390 rows to 52 rows** by rounding decimal values, optimizing the dataset for faster model execution.
* Conducted **Univariate** and **Bivariate analysis**, using various plotting techniques to compare input features with the **Output** column.
* AGE
* Validated the dataset and retained the **Age** column without modification, as it is essential to include a range of age values for effective model development.
* Conducted **Univariate** and **Bivariate analysis**, using various plotting techniques to examine the relationship between input features and the **Output** column.
* SEX
* Conducted **Univariate** and **Bivariate analysis**, using various plotting techniques to examine the relationship between input features and the **Output** column.
* **Pairplot** and **Hisplot** on all relevant columns of the dataset to analyse the details

***Treatment Recommendation Algorithm***

* Updated **X** by dropping the **Treatment Recommendation**, **Patient Name**, and **SEX** columns, and assigned the **Output** column to **y** for the target variable.
* Applied **Label Encoding** to convert the **Treatment Recommendation** column from categorical (non-numeric) data to numerical values, ensuring efficient model training.
* Imported **train test split** and split the data into training and testing sets, adjusting the test size to **0.12** and setting the **random state** to **7** for reproducibility.
* Imported **Standard Scaler** and transformed both the training and testing data to normalize the features, removing the mean and scaling to unit variance. This ensures that all features are on comparable scales, which is essential for the effective development of many machine learning algorithms.
* Validated the correlation between the features and the **Output** column, visualizing it using a **heatmap**.

***Description of design choices***

* Implemented **Random Forest**, **Logistic Regression** and **SVM** modelsandfound as the highest accuracy achieved is by **Random Forest model**
* To Maximize data usage and reduce overfitting **cross validation** has been performed on the dataset
* **Grid Search** has been performed as to validate the most accurate and sufficient Machine learning Model and hyperparameters for better model performance

***Performance evaluation of the model***

* Imported the **Random Forest Classifier** algorithm and set the following parameters:
* **creterion**: entropy
* **max\_depth**: 10
* **min\_samples\_leaf**: 1
* **min\_samples\_split**: 2
* **n\_estimator**: 100
* Fit the data into the model and made predictions on the dataset.
* Imported the **Accuracy Score**, **Confusion Matrix**, **f1\_score, recall\_score, precision\_score** and **Classification Report** to evaluate the model's performance.
* Achieved an **Accuracy Score** of **78%** and **F1 Score** of 83% after tuning the hyperparameters.

***Model Interpretability Report***

* Imported **Pickle** for the development of the **User Interface**.
* Designed a **Business Intelligence (BI) User Interface** that is user-friendly and efficient. The interface allows users to input health parameters, and the model predicts whether treatment is needed for the patient based on the factors trained in the machine learning model.

***Discussion of future work***

**Additional Feature Engineering:**

* More advanced feature engineering could be performed, including the creation of new features based on domain knowledge, or the transformation of existing ones to improve model interpretability and performance.

**User Interface Deployment for End-Users:**

* The model could be deployed on **end-user smartphones**, allowing patients to efficiently validate their health conditions after undergoing any pathological services. This would enable users to easily assess their health status without the need to visit a primary care physician or specialist, thus increasing accessibility and convenience.

**Evaluation with New Data:**

* As more data becomes available, continuous training and evaluation of the model will be essential to maintain its relevance and improve its predictive power over time.

**Deployment and Scalability:**

* The model could be deployed on cloud platforms for scalability, allowing it to handle a larger number of users and datasets. Exploring **API** development and **integration with healthcare platforms** would enable broader application in real-world healthcare environments.

**End User usage:**

* This will provide users with a convenient way to assess their health condition independently, without the need to consult a primary care physician or specialist.

***Success Metrics***

* After tuning the hyperparameters and evaluating various machine learning models, the highest accuracy was achieved using the **Random Forest Classifier**.
* The model can be effectively utilized by regional doctors to analyze patient conditions by updating relevant parameters, offering a time-saving and efficient solution, particularly beneficial for intern doctors in the absence of more experienced physicians.
* **Data security** remains a top priority, ensuring compliance with HIPAA regulations. All patient names used in the model are hypothetical to protect privacy and adhere to healthcare standards.
* **Patient feedback** has been overwhelmingly positive, with users highlighting the **user interface's** accuracy, simplicity, and user-friendliness, making it highly effective for real-time use cases.

***Instructions for Running the Machine Learning Model***

* **Launch Colab or Jupyter Notebook**:
  + Begin by opening a new notebook in **Google Colab** or **Jupyter Notebook**.
* **Establish Server Connection:**
  + Ensure that you are connected to the appropriate server in your environment. If using **Google Colab**, it will connect automatically.
* **Execute the Code:**
  + Save and run the provided code within your notebook environment to initiate the machine learning process.
* **Deploy the Model:**
  + Follow the deployment instructions and click on the URL provided to update the **External URL** for proper integration.
* **Interact with the User Interface:**
  + Select the relevant parameters from the **User Interface**. Once the parameters are chosen, click on the **Predict** **button**.
  + The system will generate customized predictions based on the input you have provided