ABSTRACT

Preeclampsia is a complex pregnancy complication characterized by high blood pressure and signs of damage to another organ system, most often the liver and kidneys. It typically occurs after 20 weeks of pregnancy and can lead to serious, even fatal, complications for both mother and baby if left unmanaged. Early prediction and intervention are crucial to managing the risks associated with preeclampsia. This Project explores the development and application of machine learning (ML) models to predict the likelihood of preeclampsia in pregnant women. Utilizing a dataset comprised of medical records, including demographic information, medical histories, and laboratory test results, we trained and evaluated several ML algorithms to identify those at high risk for developing preeclampsia. The project compares the performance of various models, including logistic regression, support vector machines, random forests, in terms of accuracy, sensitivity, and specificity. The best-performing model offers a promising tool for healthcare providers to enhance antenatal care by identifying high-risk patients early in their pregnancy, thereby enabling timely and targeted interventions. This research not only contributes to the field of medical informatics by advancing the predictive capabilities of ML in antenatal care but also demonstrates the potential for ML to improve outcomes in preeclampsia and other pregnancy-related complications.

CHAPTER ONE
INTRODUCTION

1.1 Background of the Study

Preeclampsia is a complicated condition related to high blood pressure that impacts around 5-10% of all pregnancies. It is identified by elevated blood pressure and frequently a considerable presence of protein in the urine, typically emerging after the 20th week of pregnancy. It's vital to detect and manage it early to avoid serious risks, including death of the mother and baby. Nonetheless, forecasting preeclampsia, especially in its initial phases, is notably difficult because of its complex causes and the diverse ways it can manifest.

1.2 Problem Statement

Despite advancements in obstetric care, the early prediction of preeclampsia continues to pose a challenge. Current models and tests lack the sensitivity and specificity required to accurately identify women at risk during the early stages of pregnancy. This limitation leads to missed diagnoses, delayed interventions, and increased risk of adverse outcomes. There is a critical need for a predictive model that can effectively utilize clinical, biochemical, and biophysical markers to identify pregnant women at high risk for early preeclampsia.

1.3 Aims and Objectives

Develop a Predictive Model: To create a model that leverages advanced statistical and machine learning techniques to predict early preeclampsia with high accuracy.
Identify Key Predictors: To analyze a wide range of potential predictors, including demographic, clinical, genetic, and environmental factors, to understand their contribution to the risk of developing early preeclampsia.
Assess Effectiveness in Clinical Settings: To evaluate the predictive model's performance through prospective studies in diverse populations to ensure its effectiveness and applicability in real-world clinical settings.

1.4 Significance of the Study

The development of an accurate predictive model for early preeclampsia has the potential to significantly impact maternal and neonatal health outcomes. By enabling early identification and intervention, this study aims to reduce the incidence of severe preeclampsia and its associated complications. Additionally, it contributes valuable insights to the field of obstetrics, supporting the development of guidelines and policies to improve pregnancy care. Ultimately, this research could lead to better resource allocation, more personalized care, and improved health outcomes for mothers and their babies.

1.5 Scope of the Study

Population and Demographics: The study focuses on pregnant women across various demographics, including age, BMI, and ethnicity, to ensure diverse representation. It will particularly concentrate on those in the first and second trimesters, as the aim is to predict early preeclampsia.

Predictive Factors Analyzed: A wide range of factors will be considered, including genetic markers, blood pressure measurements, blood and urine biochemistry, and ultrasonography data. The study aims to integrate these factors into a comprehensive predictive model.

Technological Framework: Advanced statistical models and machine learning algorithms, such as logistic regression, random forests, and neural networks, will be employed to analyze the data and develop the prediction model.

Geographical Coverage: The study will be conducted in several healthcare facilities, including both urban and rural settings, to ensure the model's applicability across different healthcare systems and settings.

1.6 Limitations of the Study

Data Availability and Quality: The accuracy of the predictive model is highly dependent on the availability and quality of the data collected. Limitations in data collection, such as number of data collected, values or inaccuracies in self-reported information, may impact the study's outcomes.

Generalizability: While the study aims to include a diverse population, the findings may not be fully generalizable to all pregnant women worldwide due to differences in genetic, environmental, and healthcare factors.

Predictive Model Constraints: The complexity of preeclampsia as a disease means that even with advanced modeling techniques, the prediction model may not achieve perfect accuracy. The interplay of numerous factors contributing to preeclampsia can limit the model's predictive capability.

Ethical and Privacy Considerations: Ensuring the privacy and ethical treatment of participants' data is paramount. The study's scope is constrained by the need to comply with ethical standards and data protection regulations, which may limit the extent and depth of data analysis.

Resource and Time Constraints: The study's scope is subject to the availability of financial, technological, and human resources. Additionally, the time frame for the study may not allow for the longitudinal analysis of predictors over an extended period.

CHAPTER TWO
LITERATURE REVIEW

Preeclampsia, also referred to as toxemia is a condition unique to pregnancy. It impacts 3-5% of expectant mothers and is identified by the presence of swelling, hypertension, and protein in the urine. Preeclampsia increases a woman's risk of developing cardiovascular diseases later in life, it can also lead to the dysfunction of multiple organs like the kidneys and liver and can restrict fetal growth. Without appropriate management, it can have fatal consequences and is a significant cause of maternal and neonatal mortality in underprivileged areas. This health issue is a leading contributor to the high rates of death among mothers and newborns in economically disadvantaged regions. Currently, in severe cases, the only management strategy involves stabilizing the condition of the mother and the unborn child, followed by strategically timing the delivery to benefit both (Filipek A, Jurewicz, 2018). Annually, over half a million women succumb to pregnancy-related complications, with 10% experiencing hypertension and 2% to 8% facing preeclampsia, which can lead to serious health issues, including organ damage and clotting disorders, and adversely affect the baby's growth and timing of birth. (Duley L., 2009). Preeclampsia notably increases the likelihood of complications throughout pregnancy. If left untreated or in severe form, maternal pulmonary edema, eclampsia, brain injury, and death can occur. It is estimated that preeclampsia accounts for 10% of all global maternal deaths related to pregnancy.(Shamsi U et al..2013).Predicting preeclampsia is complex, Nonetheless, statistical learning techniques are capable of sifting through various data forms, including patient medical records and laboratory findings, to pinpoint the most pertinent details for forecasting.(Marić I, Tsur et al.. 2020)

2.1 Related Works:

Machine Learning (ML) is essential for enabling a model to learn from and adapt to input data autonomously, without needing direct programming. This technology endows machines with the ability to comprehend data, identify patterns, and subsequently make predictions or decisions(Kumar N, Aggarwal D. 2021). A predictive model has been crafted, based on Bayes' theorem, which assesses the likelihood of developing preeclampsia necessitating delivery within certain timeframes after 30–33 weeks of pregnancy. This model incorporates maternal traits and medical history, as well as biophysical and biochemical indicators. Early findings corroborate that the inherent risk for preeclampsia is influenced by maternal characteristics and escalates with older maternal age, higher body weight, and among women of Afro-Caribbean and South Asian descent. The likelihood of developing preeclampsia is greater in individuals who have a personal or family history of the condition. It's also higher in women who already have health issues like ongoing high blood pressure, diabetes, or autoimmune diseases such as systemic lupus erythematosus or antiphospholipid syndrome.(Tayyar A et al.. 2014).

CHAPTER THREE
METHODOLOGY

3.1 Design and Implementation

This project evaluates classification supervised Machine Learning model by visualizing the model's results in a single interface website built using Flask. The website's inputs are the model and the attributes for both testing and training sets. The output is the UI which contains the prediction on whether the patient has preeclampsia or not. More so, the figure below depicts the methodology adopted for this study. Importing datasets is the most important and foremost step before starting the analysis and model development. Machine Learning classifiers are proposed in this study for predictive analysis. We used a sample dataset from the electronic health records of Murtala Muhammad Specialist hospital in Northern Nigeria. The dataset is used to test and train on Random Forest and Gradient Boosting classifier algorithms, where RF was used as the final. Also, data preprocessing is used for cleansing and reduction of data, which also deals with missing values to improve accuracy. Before displaying the dataset, the streamed data undergoes preprocessing for direct visualization and analysis.

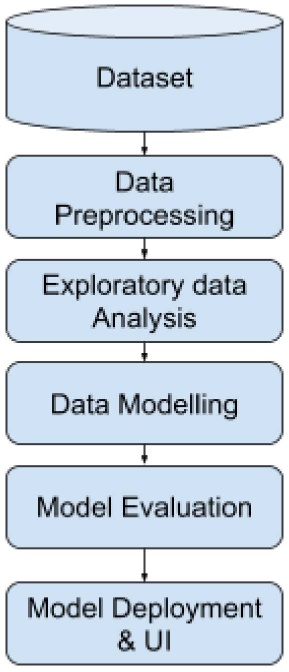
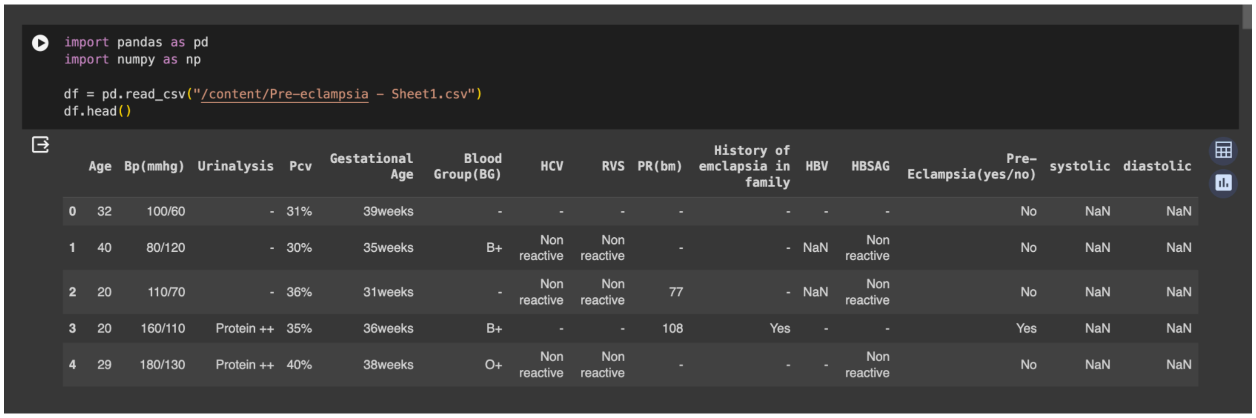


Fig 3.1 Project Lifecycle

3.2 Data Preprocessing

Data Collection:

Data collection is a fundamental aspect of machine learning, providing the essential raw materials required for training models. These models leverage the collected data to discern patterns, relationships, and ultimately, to make informed predictions. A thorough initial review of data sources is crucial to determine the feasibility and direction of further predictive modeling efforts. his dataset is derived from the Murtala Muhammad Specialist Hospital, a prominent healthcare institution in Northern Nigeria. It encompasses data from approximately 235 patients, distributed across 12 distinctive features, offering a comprehensive insight into the hospital's operational and patient care dynamics.To facilitate the prediction of pre-eclampsia, the initial step involves a meticulous examination of the dataset provided. This process begins with loading the data, followed by a detailed inspection of its structure and the types of variables it encompasses. Such an examination is pivotal for identifying the requisite preprocessing and analytical methodologies to be employed as shown below:



```
import pandas as pd
import numpy as np

df = pd.read_csv("/content/Pre-eclampsia - Sheet1.csv")
df.head()
```

	Age	Bp(mmhg)	Urinalysis	Pcv	Gestational Age	Blood Group(BG)	HCV	RVS	PR(bm)	History of emclapsia in family	HBV	HBSAG	Pre-Eclampsia(yes/no)	systolic	diastolic
0	32	100/60	-	31%	39weeks	-	-	-	-	-	-	-	No	NaN	NaN
1	40	80/120	-	30%	35weeks	B+	Non reactive	Non reactive	-	-	NaN	Non reactive	No	NaN	NaN
2	20	110/70	-	36%	31weeks	-	Non reactive	Non reactive	77	-	NaN	Non reactive	No	NaN	NaN
3	20	160/110	Protein ++	35%	36weeks	B+	-	-	108	Yes	-	-	Yes	NaN	NaN
4	29	180/130	Protein ++	40%	38weeks	O+	Non reactive	Non reactive	-	-	-	Non reactive	No	NaN	NaN

Fig 3.2.1 Uploading and reading datasets

Initial Data Inspection and Features Overview:

The dataset's rich composition of clinical and demographic features sets the stage for an in-depth analysis aimed at predicting pre-eclampsia. However, the identified preprocessing challenges, such as data cleanliness and feature clarification, underscore the need for meticulous data management to ensure the integrity of the subsequent analytical processes.

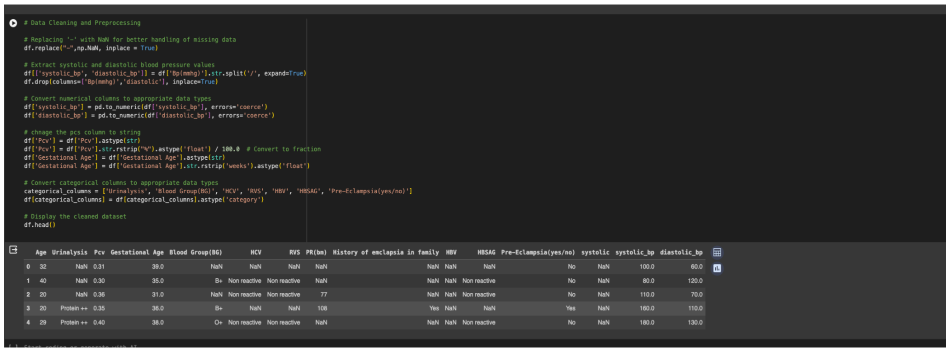
Features Overview:

- **Age:** Represents the patient's age, a fundamental demographic variable.
- **Urinalysis:** Documents the outcomes of urinalysis tests, crucial for detecting various conditions.
- **Pcv (Packed Cell Volume):** Measures the proportion of blood volume that is occupied by red blood cells, indicating hematocrit levels.
- **Gestational Age:** Specifies the age of the pregnancy, measured in weeks, vital for understanding pregnancy progression.
- **Blood Group (BG)**: Identifies the patient's blood type, important for transfusions and pregnancy-related compatibility issues.
- **HCV (Hepatitis C Virus):** Reflects the results of the Hepatitis C virus test.
- **RVS:** Denotes results from a medical test, requiring further clarification on the acronym's meaning.
- **PR(bm):** Potentially indicates pulse rate; however, further clarification is needed to confirm its precise definition.
- **History of Eclampsia in Family:** Signifies whether there is a familial history of eclampsia, a critical risk factor.
- **HBV (Hepatitis B Virus):** Shows the results of the Hepatitis B virus test.
- **HBSAG (Hepatitis B Surface Antigen)**: Indicates the presence of the Hepatitis B surface antigen, a key marker for infection.
- **Pre-Eclampsia (Yes/No):** The primary target variable, denoting the presence or absence of pre-eclampsia in the patient.
- **Systolic & Systolic_bp:** Highlighted as a potentially duplicate field, suggesting a typographical error and necessitating clarification.
- **Diastolic_bp:** Records the diastolic blood pressure, essential for monitoring cardiovascular health.

Cleaning the Data:

Handling Missing Data: Missing values were notably present across various columns like 'Urinalysis', 'Blood Group(BG)', 'HCV', 'RVS', 'PR(bm)', 'History of eclampsia in family', 'HBV', 'HBSAG', and 'systolic Decisions were made to either impute or drop these columns based on the missing data's extent.

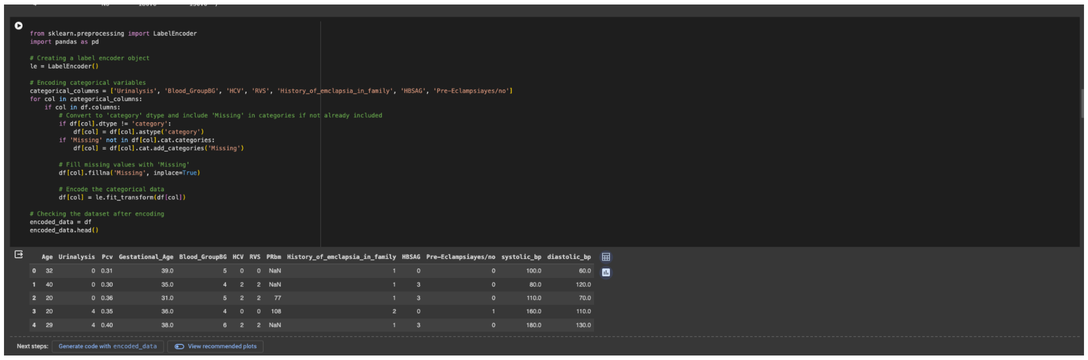
- **Data Types Correction:** Data types were adjusted to ensure numerical and categorical data were correctly classified, facilitating subsequent analysis.
- **Duplicate Columns**: A duplicate column analysis led to the removal of redundancies, streamlining the dataset for modeling.
- **Categorical Data Encoding:** Categorical variables underwent encoding to transform them into a format suitable for machine learning algorithms.



```python
# Data Cleaning and Preprocessing

# Replacing '-' with NaN for better handling of missing data
df.replace("-",np.NaN, inplace = True)

# Extract systolic and diastolic blood pressure values
df[['systolic_bp', 'diastolic_bp']] = df['Bp(mmhg)'].str.split('/', expand=True)
df.drop(columns=['Bp(mmhg)','diastolic'], inplace=True)

# Convert numerical columns to appropriate data types
df['systolic_bp'] = pd.to_numeric(df['systolic_bp'], errors='coerce')
df['diastolic_bp'] = pd.to_numeric(df['diastolic_bp'], errors='coerce')

# chnage the pcs column to string
df['Pcv'] = df['Pcv'].astype(str)
df['Pcv'] = df['Pcv'].str.rstrip("%").astype('float') / 100.0  # Convert to fraction
df['Gestational Age'] = df['Gestational Age'].astype(str)
df['Gestational Age'] = df['Gestational Age'].str.rstrip('weeks').astype('float')

# Convert categorical columns to appropriate data types
categorical_columns = ['Urinalysis', 'Blood Group(BG)', 'HCV', 'RVS', 'HBV', 'HBSAG', 'Pre-Eclampsia(yes/no)']
df[categorical_columns] = df[categorical_columns].astype('category')

# Display the cleaned dataset
df.head()
```

```
   Age  Urinalysis  Pcv   Gestational Age  Blood Group(BG)  HCV           RVS           PR(bm)  History of emclapsia in family  HBV  HBSAG         Pre-Eclampsia(yes/no)  systolic  systolic_bp  diastolic_bp
0  32          NaN  0.31             39.0              NaN           NaN           NaN     NaN                             NaN  NaN           NaN                     No       NaN        100.0          60.0
1  40          NaN  0.30             35.0               B+  Non reactive  Non reactive     NaN                             NaN  NaN  Non reactive                     No       NaN         80.0         120.0
2  20          NaN  0.36             31.0              NaN  Non reactive  Non reactive      77                             NaN  NaN  Non reactive                     No       NaN        110.0          70.0
3  20   Protein ++  0.35             36.0               B+           NaN           NaN     108                             Yes  NaN           NaN                    Yes       NaN        160.0         110.0
4  29   Protein ++  0.40             38.0               O+  Non reactive  Non reactive     NaN                             NaN  NaN  Non reactive                     No       NaN        180.0         130.0
```

Fig 3.2.3 Dtaa type classification and column analysis



```python
from sklearn.preprocessing import LabelEncoder
import pandas as pd

# Creating a label encoder object
le = LabelEncoder()

# Encoding categorical variables
categorical_columns = ['Urinalysis', 'Blood_GroupBG', 'HCV', 'RVS', 'History_of_emclapsia_in_family', 'HBSAG', 'Pre-Eclampsiayes/no']
for col in categorical_columns:
    if col in df.columns:
        # Convert to 'category' dtype and include 'Missing' in categories if not already included
        if df[col].dtype != 'category':
            df[col] = df[col].astype('category')
        if 'Missing' not in df[col].cat.categories:
            df[col] = df[col].cat.add_categories('Missing')

        # Fill missing values with 'Missing'
        df[col].fillna('Missing', inplace=True)

        # Encode the categorical data
        df[col] = le.fit_transform(df[col])

# Checking the dataset after encoding
encoded_data = df
encoded_data.head()
```

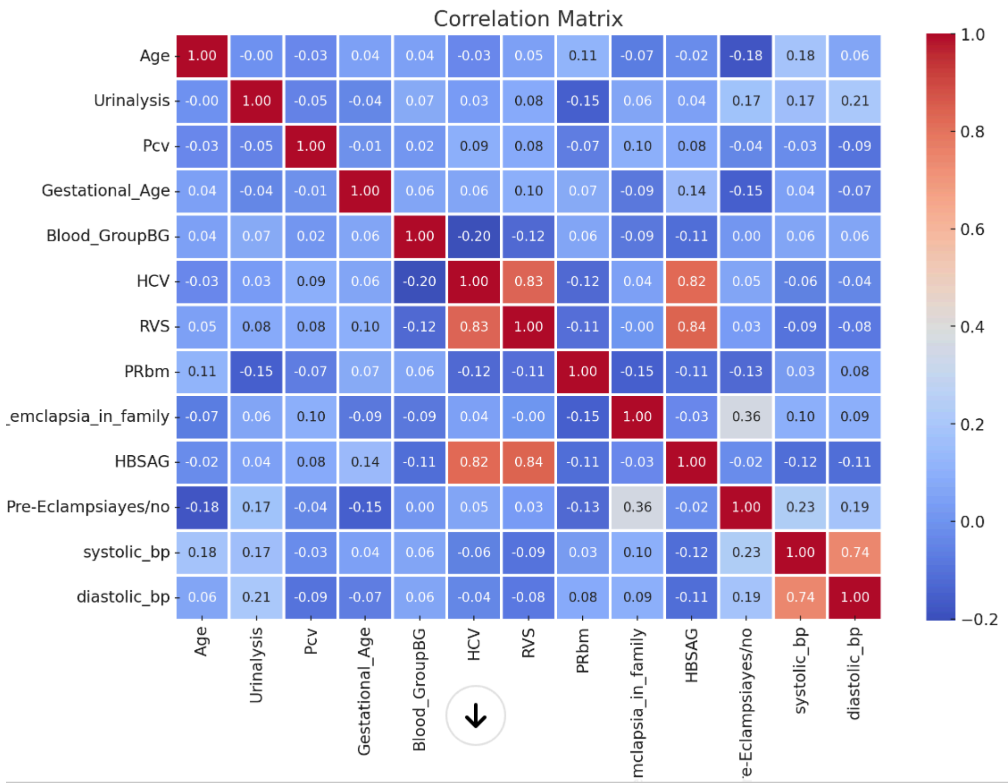
```
   Age  Urinalysis  Pcv   Gestational_Age  Blood_GroupBG  HCV  RVS  PRbm  History_of_emclapsia_in_family  HBSAG  Pre-Eclampsiayes/no  systolic_bp  diastolic_bp
0  32            0  0.31             39.0              5    0    0   NaN                               1      0                    0        100.0          60.0
1  40            0  0.30             35.0              4    2    2   NaN                               1      3                    0         80.0         120.0
2  20            0  0.36             31.0              5    2    2    77                               1      3                    0        110.0          70.0
3  20            4  0.35             36.0              4    0    0   108                               2      0                    1        160.0         110.0
4  29            4  0.40             38.0              6    2    2   NaN                               1      3                    0        180.0         130.0
```

Fig 3.2.4 label Encoding

3.3 Exploratory Data Analysis (EDA)

Embarking on an exploratory data analysis (EDA) is pivotal for uncovering the intricate relationships and patterns embedded within our dataset, with a particular focus on identifying variables that significantly influence the prediction of pre-eclampsia. This endeavor will encompass a thorough examination of the distribution of individual variables, alongside an exploration of their correlations and potential interactions with the target variable, 'Pre-Eclampsia (Yes/No)'.

Our analysis will unfold in stages, beginning with basic descriptive statistics to establish a foundational understanding of the dataset's characteristics. Subsequently, Delved into examination of distributions and correlations, before culminating in the visualization of key relationships that emerge from our investigation.



	Age	Urinalysis	Pcv	Gestational_Age	Blood_GroupBG	HCV	RVS	PRbm	mclapsia_in_family	HBSAG	e-Eclampsiayes/no	systolic_bp	diastolic_bp
Age	1.00	-0.00	-0.03	0.04	0.04	-0.03	0.05	0.11	-0.07	-0.02	-0.18	0.18	0.06
Urinalysis	-0.00	1.00	-0.05	-0.04	0.07	0.03	0.08	-0.15	0.06	0.04	0.17	0.17	0.21
Pcv	-0.03	-0.05	1.00	-0.01	0.02	0.09	0.08	-0.07	0.10	0.08	-0.04	-0.03	-0.09
Gestational_Age	0.04	-0.04	-0.01	1.00	0.06	0.06	0.10	0.07	-0.09	0.14	-0.15	0.04	-0.07
Blood_GroupBG	0.04	0.07	0.02	0.06	1.00	-0.20	-0.12	0.06	-0.09	-0.11	0.00	0.06	0.06
HCV	-0.03	0.03	0.09	0.06	-0.20	1.00	0.83	-0.12	0.04	0.82	0.05	-0.06	-0.04
RVS	0.05	0.08	0.08	0.10	-0.12	0.83	1.00	-0.11	-0.00	0.84	0.03	-0.09	-0.08
PRbm	0.11	-0.15	-0.07	0.07	0.06	-0.12	-0.11	1.00	-0.15	-0.11	-0.13	0.03	0.08
_emclapsia_in_family	-0.07	0.06	0.10	-0.09	-0.09	0.04	-0.00	-0.15	1.00	-0.03	0.36	0.10	0.09
HBSAG	-0.02	0.04	0.08	0.14	-0.11	0.82	0.84	-0.11	-0.03	1.00	-0.02	-0.12	-0.11
Pre-Eclampsiayes/no	-0.18	0.17	-0.04	-0.15	0.00	0.05	0.03	-0.13	0.36	-0.02	1.00	0.23	0.19
systolic_bp	0.18	0.17	-0.03	0.04	0.06	-0.06	-0.09	0.03	0.10	-0.12	0.23	1.00	0.74
diastolic_bp	0.06	0.21	-0.09	-0.07	0.06	-0.04	-0.08	0.08	0.09	-0.11	0.19	0.74	1.00

Fig 3.2.5 Variables Correlation matrix

Insights from the Exploratory Data Analysis

Descriptive Statistics

Our analysis spans a diverse range of patient ages and gestational ages, encapsulating a broad spectrum of pregnancy stages.The average systolic and diastolic blood pressures within the dataset stand at 139.85 mmHg and 90.77 mmHg, respectively, indicating a varied range of blood pressure readings among the participants.

Correlation Analysis: A pronounced positive correlation is observed with History_of_eclampsia_in_family, systolic_bp, and diastolic_bp in relation to the incidence of pre-eclampsia, underscoring their potential as significant predictors.Urinalysis results also exhibit a noteworthy positive correlation with pre-eclampsia occurrences, further highlighting its predictive value. In contrast, Age and Gestational_Age are inversely correlated with pre-eclampsia, suggesting a decrease in likelihood with advancing age and gestation period. PRbm's negative correlation implies that an increase in PRbm is associated with a reduced risk of pre-eclampsia.

Visual Analysis: The analysis is complemented by a heatmap visualization, which elucidates the strength and direction of the relationships between variables, providing a visual representation of their interconnectedness.

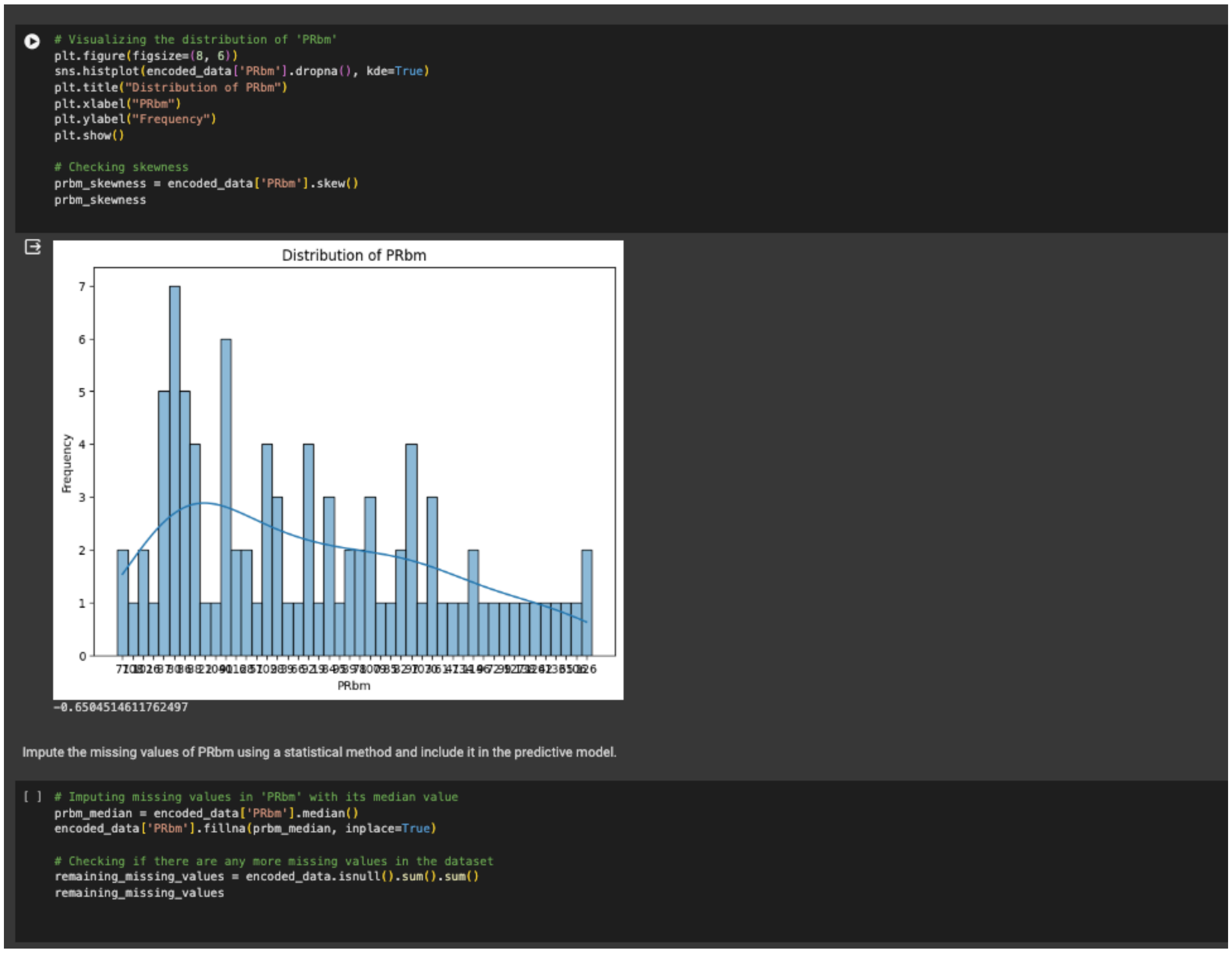
Key Observations

Notably, a family history of eclampsia, blood pressure readings, and urinalysis outcomes emerge as critical factors for pre-eclampsia prediction. Conversely, variables such as age and gestational age may influence the risk of pre-eclampsia in the opposite direction, meriting further investigation. The 'PRbm' column, characterized by its negative correlation with the target variable and the presence of missing data, presents a unique challenge in our predictive modeling efforts. To address this, a thoughtful approach towards imputing the missing values is warranted, ensuring that we retain this potentially valuable predictor in our model. In light of the numeric nature of the 'PRbm' data, we propose employing a common yet effective imputation technique tailored to quantitative data. The essence of this technique involves substituting missing values with a measure of central tendency—specifically, the median or the mean of the column. The decision to use the median or mean hinges on the data's distribution:

For skewed data: The median is preferred as it provides a more robust measure of central tendency, less influenced by outliers.

For symmetric data distributions: The mean serves as an appropriate choice, accurately reflecting the central point of the data.

The imputation strategy is grounded in the principle of maintaining the statistical integrity of the 'PRbm' column, while minimizing the impact of missing data on our predictive model. The implementation of this imputation will involve a preliminary analysis of the 'PRbm' distribution to ascertain its skewness, followed by the application of the selected measure of central tendency. This process not only enhances the completeness of our dataset but also bolsters the reliability of our subsequent analyses.



```
# Visualizing the distribution of 'PRbm'
plt.figure(figsize=(8, 6))
sns.histplot(encoded_data['PRbm'].dropna(), kde=True)
plt.title("Distribution of PRbm")
plt.xlabel("PRbm")
plt.ylabel("Frequency")
plt.show()

# Checking skewness
prbm_skewness = encoded_data['PRbm'].skew()
prbm_skewness
```

```
-0.6504514611762497
```

Impute the missing values of PRbm using a statistical method and include it in the predictive model.

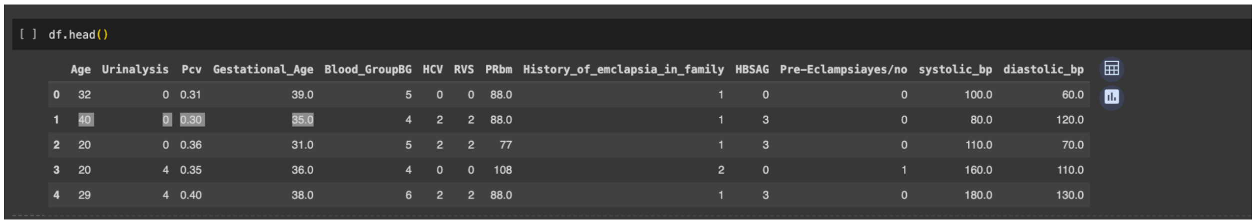
```
[ ] # Imputing missing values in 'PRbm' with its median value
    prbm_median = encoded_data['PRbm'].median()
    encoded_data['PRbm'].fillna(prbm_median, inplace=True)

    # Checking if there are any more missing values in the dataset
    remaining_missing_values = encoded_data.isnull().sum().sum()
    remaining_missing_values
```

Fig 3.2.6 Distribution of PRbm variable.

The distribution of 'PRbm' demonstrates a modest left-skew, as evidenced by a skewness coefficient of approximately -0.65. In scenarios characterized by such skewness, employing the median for imputation purposes is deemed more methodologically sound. This is attributed to the median's resilience to distortions caused by skewness and outliers.

We proceeded to remediate the missing values within the 'PRbm' dataset by utilizing its median value for imputation. This intervention will render the dataset complete, thereby facilitating the foundational work necessary for the development of a predictive model.

Upon successful imputation of all missing values, we have achieved a fully constituted dataset. This meticulously cleaned and preprocessed dataset primes us for the next phase of constructing a predictive model specifically designed for the anticipation of pre-eclampsia outcomes.



```
[ ] df.head()
```

	Age	Urinalysis	Pcv	Gestational_Age	Blood_GroupBG	HCV	RVS	PRbm	History_of_emclapsia_in_family	HBSAG	Pre-Eclampsiayes/no	systolic_bp	diastolic_bp
0	32	0	0.31	39.0	5	0	0	88.0	1	0	0	100.0	60.0
1	40	0	0.30	35.0	4	2	2	88.0	1	3	0	80.0	120.0
2	20	0	0.36	31.0	5	2	2	77	1	3	0	110.0	70.0
3	20	4	0.35	36.0	4	0	0	108	2	0	1	160.0	110.0
4	29	4	0.40	38.0	6	2	2	88.0	1	3	0	180.0	130.0

Fig 3.2.6 Cleaned dataset

3.4 Model Building

The dataset has been divided into training and testing subsets, with feature scaling uniformly applied to ensure consistency across the data. The training set comprises 162 observations, while the testing set contains 70 observations, each featuring 12 distinct variables.

Machine Learning Algorithms:

Here's a brief overview of the machine Learning models evaluated and the performance metrics of each:

Logistic Regression:A statistical method that models the probability of a binary outcome based on one or more predictor variables. It's widely used for binary classification problems (e.g., spam or not spam).

Random Forest:An ensemble learning method that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) of the individual trees. It's known for its high accuracy, robustness, and ability to handle large data sets with higher dimensionality.

Support Vector Machine (SVM):A powerful and versatile supervised learning algorithm used for both classification and regression. It works by finding the hyperplane that best separates different classes in the feature space, maximizing the margin between the classes' closest points (support vectors).

K-Nearest Neighbors (KNN):A simple, instance-based learning algorithm where the class of a sample is determined by the majority class among its k nearest neighbors.

Gradient Boosting Classifier:An ensemble technique that builds models in a stage-wise fashion and generalizes them by allowing optimization of an arbitrary differentiable loss function. It builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions and is typically used for its predictive accuracy.

	Logistic Regression	Random forest	SVM	KNN	Gradient Boosting classifier
Accuracy	58.57%	71.4%	65.71%	62.86%	68.57%
Precision	40.00%	59.09%	50.00%	45.83%	53.85%
Recall	41.67%	54.17%	29.17%	45.83%	58.33%
ROC AUC	54.53%	67.30%	56.97%	58.79%	66.12%

Fig 3.4.1 Results of different Different

Upon analysis, the data reveals that the Random Forest model surpasses its counterparts in terms of accuracy, precision, and ROC AUC scores, highlighting its superior capacity to predict pre-eclampsia within the dataset under review.

The Gradient Boosting Classifier has the highest Recall scores among all evaluated models. This measures the proportion of actual positives that were correctly identified by the model.marking it as a strong performer in the predictive modeling arena.

In contrast, the Support Vector Machine (SVM) and K-Nearest Neighbors (KNN) models exhibit lower levels of accuracy and ROC AUC scores. This comparative underperformance suggests that, relative to the Gradient Boosting and Random Forest models, they are less effective in accurately predicting pre-eclampsia within the dataset.

Considering the evidence, the Gradient Boosting Classifier and Random Forest models are identified as the most effective for predicting pre-eclampsia. To further enhance their performance and reliability, it is recommended to fine-tune these models and apply cross-validation techniques.

Fine Tuning and evaluation

The process of finding the best hyperparameters for a machine learning model is called Hyperparameter tuning. There are different methods for finding the best hyperparameters for your models. We adopted a grid search methodology to meticulously fine-tune the hyperparameters of both the Gradient Boosting and Random Forest models. This systematic approach was aimed at discovering the optimal combination of parameters that would enhance the performance of the models. Following the optimization, we observed significant improvements in the performance metrics of both models. Specifically, the Gradient Boosting Classifier achieved a well-balanced performance across its indicators, while the Random Forest Model showcased exceptional accuracy and precision, underscoring its predictive robustness.

For a comprehensive evaluation, we rigorously assessed the performance of these optimized models on a designated test set. This assessment was conducted using uniform metrics to facilitate a fair and accurate comparison between the models. The comparative analysis revealed that the Random Forest model excelled in terms of accuracy and precision, demonstrating its efficacy in accurately predicting outcomes and minimizing false positives. On the other hand, the Gradient Boosting Classifier showed superior performance in recall and ROC AUC metrics, indicating its strong ability to identify positive instances and its overall discriminative power.

This thorough evaluation highlighted the distinct strengths of each model, providing valuable insights that will inform our strategic decision-making in selecting the most suitable model. Our decision will be based on the specific requirements of our predictive tasks, ensuring that we leverage the unique advantages of each model to achieve the best possible outcomes.

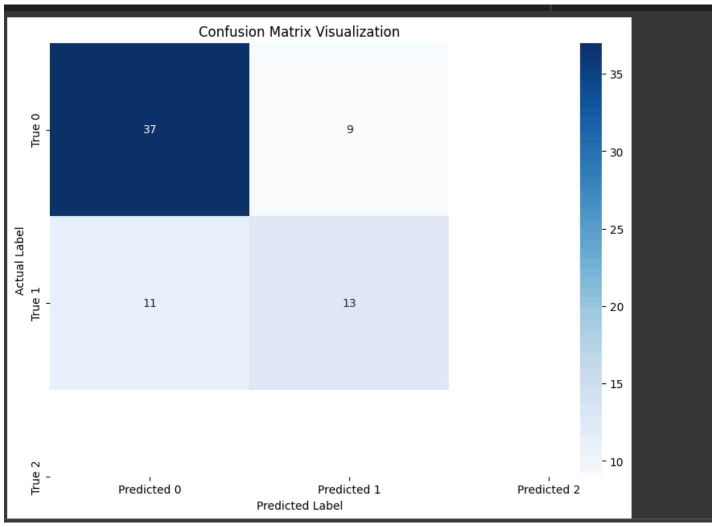


Fig 3.4.2 Results of different Different

	Random Forest	Gradient Boosting Classifier
Learning rate		0.1
Max Depth	5	4
Number of Estimators	200	200
Accuracy	72.92%	74.77%

Fig 3.4.3 Tuning the models using The simplified grid search

	Gradient Boosting Classifier	Random Forest
Accuracy	70.00%	71.43%
Precision	56.00%	61.11%
Recall	58.33%	45.83%
ROC AUC	67.21%	65.31%

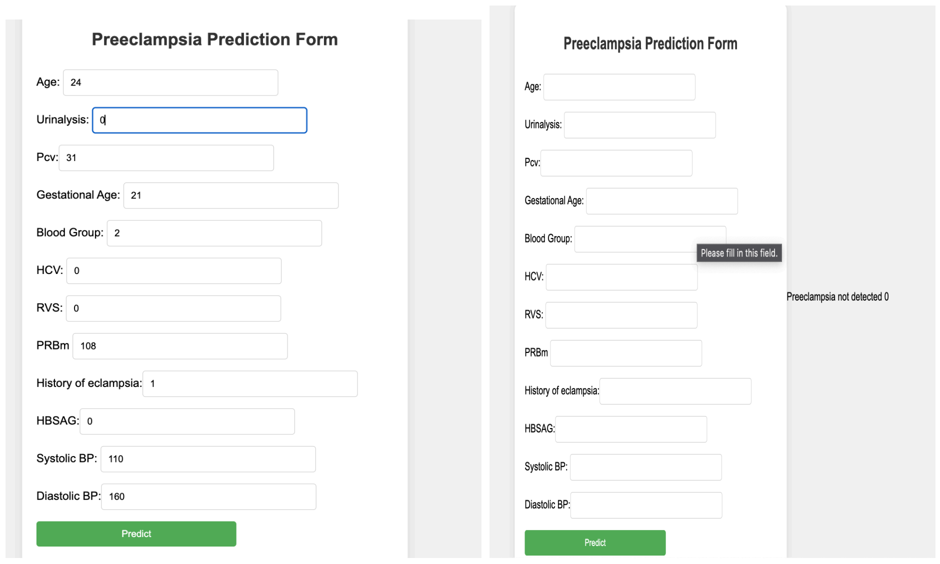
Fig 3.4.4 performance metrics for the tuned Gradient Boosting Classifier and Random Forest on test Sets

CHAPTER FIVE

RESULT ANALYSIS

Various models were trained and evaluated, including Logistic Regression, Random Forest, Support Vector Machine, K-Nearest Neighbors, and Gradient Boosting Classifier. The Random Forest model demonstrated superior performance with an accuracy of 71.4%, precision of 59.09%, and an ROC AUC score of 67.30%. After tuning, the models showed improved metrics, with the Random Forest model achieving an accuracy of 71.43% and precision of 61.11%. The findings indicate that the Random Forest model is particularly effective in predicting pre-eclampsia, suggesting its potential utility in healthcare settings for early identification and management of the condition. However, it is suggesting that there is a possibility of missing some true preeclampsia cases. It is evident that further investigative efforts and refinements in these models could significantly enhance their predictive accuracy and reliability to be clinically relevant.

As illustrated in the figure below, the web App user interface allows variables inputs to predict pre-eclampsia. For now the users will input 0 for "No" and 1 for "Yes" This data exploration enables patients to insert the data such as Age, Gestational Age, systolic and Diastolic bp etc. Additionally,The result of the prediction will show by the right side of the form.



Preeclampsia Prediction Form

Age: 24
Urinalysis: 0
Pcv: 31
Gestational Age: 21
Blood Group: 2
HCV: 0
RVS: 0
PRBm 108
History of eclampsia: 1
HBSAG: 0
Systolic BP: 110
Diastolic BP: 160

Predict

Preeclampsia Prediction Form

Age:
Urinalysis:
Pcv:
Gestational Age:
Blood Group:
Please fill in this field.
HCV:
RVS:
PRBm
History of eclampsia:
HBSAG:
Systolic BP:
Diastolic BP:

Predict

Preeclampsia not detected 0

Fig4.1 User Interface

CHAPTER FIVE

SUMMARY CONCULSION AND RECOMMENDATION

Conclusion

Our comprehensive analysis underscores the potential of machine learning algorithms, specifically the Random Forest in accurately predicting pre-eclampsia. It exhibits higher precision, implying that the positive predictions are more likely to accurately identify cases of pre-eclampsia. This research underscores the value of machine learning in enhancing diagnostic processes and highlights the potential for further exploration and integration of such models in medical practice to improve patient outcomes.

Recommendation

Enhanced Data Collection: Initiating more comprehensive data collection efforts is crucial for mitigating issues related to missing data, thereby improving the robustness and training efficacy of our models.

Ongoing Model Optimization: We advocate for the continuous refinement and validation of these models against larger and more varied datasets to refine their predictive capabilities.

Clinical Application Assessment: It is imperative to evaluate the practicality and effectiveness of integrating these predictive models within clinical workflows, aiming to augment decision-making processes in healthcare settings.

Acknowledged Limitations:

Data Integrity Concerns: The presence of a significant rate of missing data could potentially compromise the accuracy of our predictive models, highlighting the need for improved data collection and processing methodologies.

Applicability Across Populations: There is a necessity to validate these models across a diverse range of demographic groups to confirm their generalizability and ensure their broad applicability in different clinical environments.

This report encapsulates the findings of our analysis, offering a roadmap towards leveraging advanced machine learning techniques for the prediction of pre-eclampsia. By addressing the outlined recommendations and limitations, we can further our commitment to advancing healthcare outcomes through the integration of technology in medical diagnostics.

REFERENCE

1. Filipek A, Jurewicz E. PMID: 30656917 DOI: 10.18388/pb.2018_146. 2018. PubMed.

2. Duley L. The global impact of pre-eclampsia and eclampsia. Seminars in Perinatology. 2009;33(3):130–137.https://www.sciencedirect.com/science/article/abs/pii/S0146000509000214?via%3Dihub

3. Shamsi U, Saleem S, Nishter N. Epidemiology and risk factors of preeclampsia: an overview of observational studies. Al Ameen J Med Sci. 2013;6:292-300.

4. Marić I, Tsur A, Aghaeepour N, Montanari A, Stevenson DK, Shaw GM, Winn VD. Early prediction of preeclampsia via machine learning. Am J Obstet Gynecol MFM. 2020;2(2):100100. Epub 2020 Mar 14. PMID: 33345966. https://pubmed.ncbi.nlm.nih.gov/33345966/

5. Kumar N, Aggarwal D. Learning-Based Focused Web Crawler. IETE J. Res. 2021;67:1–9. https://scholar.google.com/scholar_lookup?title=LEARNING-Based+Focused+WEB+Crawler&author=Kumar,+N.&author=Aggarwal,+D.&publication_year=2021&journal=IETE+J.+Res.&volume=67&pages=1%E2%80%939&doi=10.1080/03772063.2021.1885312

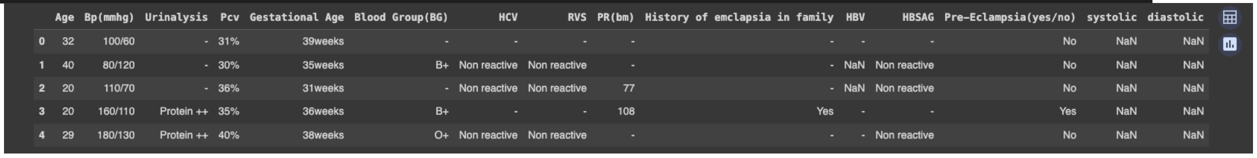
6. Tayyar A, Garcia-Tizon Larroca S, Poon LC, Wright D, Nicolaides KH. Competing risks model in screening for preeclampsia by mean arterial pressure and uterine artery pulsatility index at 30–33 weeks' gestation. Fetal Diagnosis and Therapy. 2014;36(1):18–27. https://pubmed.ncbi.nlm.nih.gov/24970282/

APPENDIX

The full code can be accessed in this Github

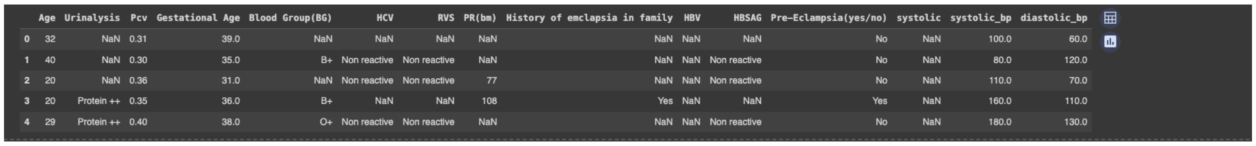
1. Pre-eclampsia.ipnyb (file)

```
Import pandas as pd
import numpy as np

df = pd.read_csv("/content/Pre-eclampsia - Sheet1.csv")
df.head()
```



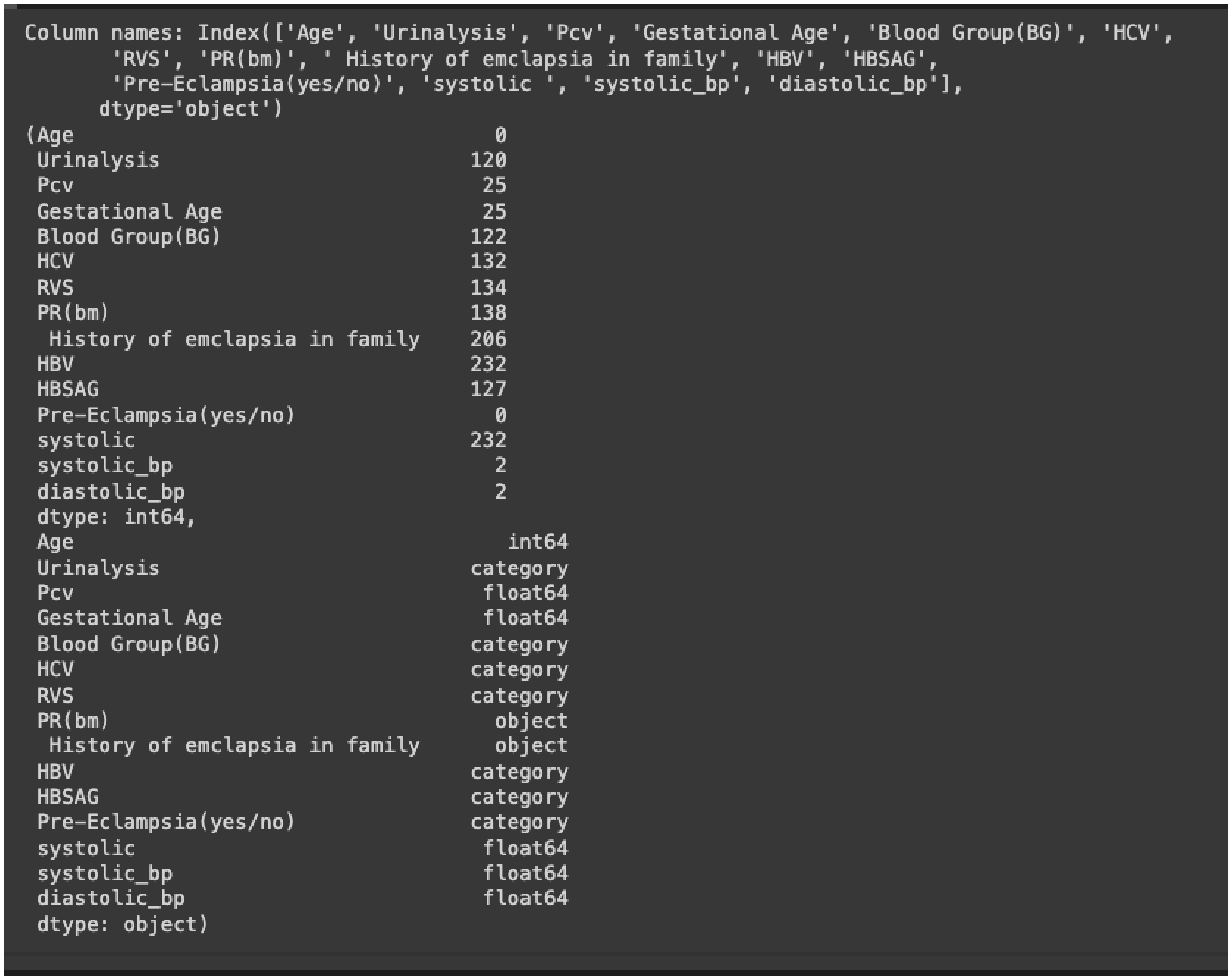
```
   Age Bp(mmhg) Urinalysis  Pcv Gestational Age Blood Group(BG)          HCV          RVS PR(bm) History of emclapsia in family  HBV        HBSAG Pre-Eclampsia(yes/no) systolic diastolic
0   32   100/60          -  31%         39weeks               -            -            -      -                              -    -            -                    No      NaN       NaN
1   40   80/120          -  30%         35weeks              B+ Non reactive Non reactive      -                              -  NaN Non reactive                    No      NaN       NaN
2   20   110/70          -  36%         31weeks               - Non reactive Non reactive     77                              -  NaN Non reactive                    No      NaN       NaN
3   20  160/110 Protein ++  35%         36weeks              B+            -            -    108                            Yes    -            -                   Yes      NaN       NaN
4   29  180/130 Protein ++  40%         38weeks              O+ Non reactive Non reactive      -                              -    - Non reactive                    No      NaN       NaN
```

```
# Data Cleaning and Preprocessing

# Replacing '-' with NaN for better handling of missing data
df.replace("-",np.NaN, inplace = True)

# Extract systolic and diastolic blood pressure values
df[['systolic_bp', 'diastolic_bp']] = df['Bp(mmhg)'].str.split('/', expand=True)
df.drop(columns=['Bp(mmhg)','diastolic'], inplace=True)

# Convert numerical columns to appropriate data types
df['systolic_bp'] = pd.to_numeric(df['systolic_bp'], errors='coerce')
df['diastolic_bp'] = pd.to_numeric(df['diastolic_bp'], errors='coerce')

# chnage the pcs column to string
df['Pcv'] = df['Pcv'].astype(str)
df['Pcv'] = df['Pcv'].str.rstrip("%").astype('float') / 100.0  # Convert to fraction
df['Gestational Age'] = df['Gestational Age'].astype(str)
df['Gestational Age'] = df['Gestational Age'].str.rstrip('weeks').astype('float')

# Convert categorical columns to appropriate data types
categorical_columns = ['Urinalysis', 'Blood Group(BG)', 'HCV', 'RVS', 'HBV', 'HBSAG',
'Pre-Eclampsia(yes/no)']
df[categorical_columns] = df[categorical_columns].astype('category')

# Display the cleaned dataset
df.head()
```



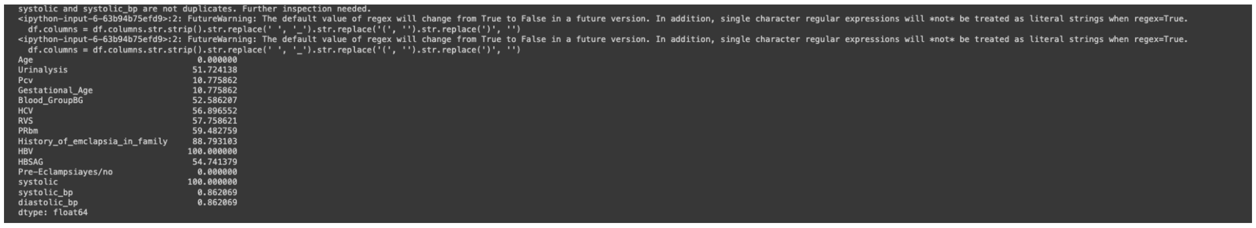
	Age	Urinalysis	Pcv	Gestational Age	Blood Group(BG)	HCV	RVS	PR(bm)	History of emclapsia in family	HBV	HBSAG	Pre-Eclampsia(yes/no)	systolic	systolic_bp	diastolic_bp
0	32	NaN	0.31	39.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	No	NaN	100.0	60.0
1	40	NaN	0.30	35.0	B+	Non reactive	Non reactive	NaN	NaN	NaN	Non reactive	No	NaN	80.0	120.0
2	20	NaN	0.36	31.0	NaN	Non reactive	Non reactive	77	NaN	NaN	Non reactive	No	NaN	110.0	70.0
3	20	Protein ++	0.35	36.0	B+	NaN	NaN	108	Yes	NaN	NaN	Yes	NaN	160.0	110.0
4	29	Protein ++	0.40	38.0	O+	Non reactive	Non reactive	NaN	NaN	NaN	Non reactive	No	NaN	180.0	130.0

```
Data Cleaning Process

# Checking for duplicate columns and any irrelevant columns
print("Column names:", df.columns)

# Checking for the number of missing values in each column
missing_values = df.isnull().sum()

# Checking data types of each column
data_types = df.dtypes

missing_values, data_types
```



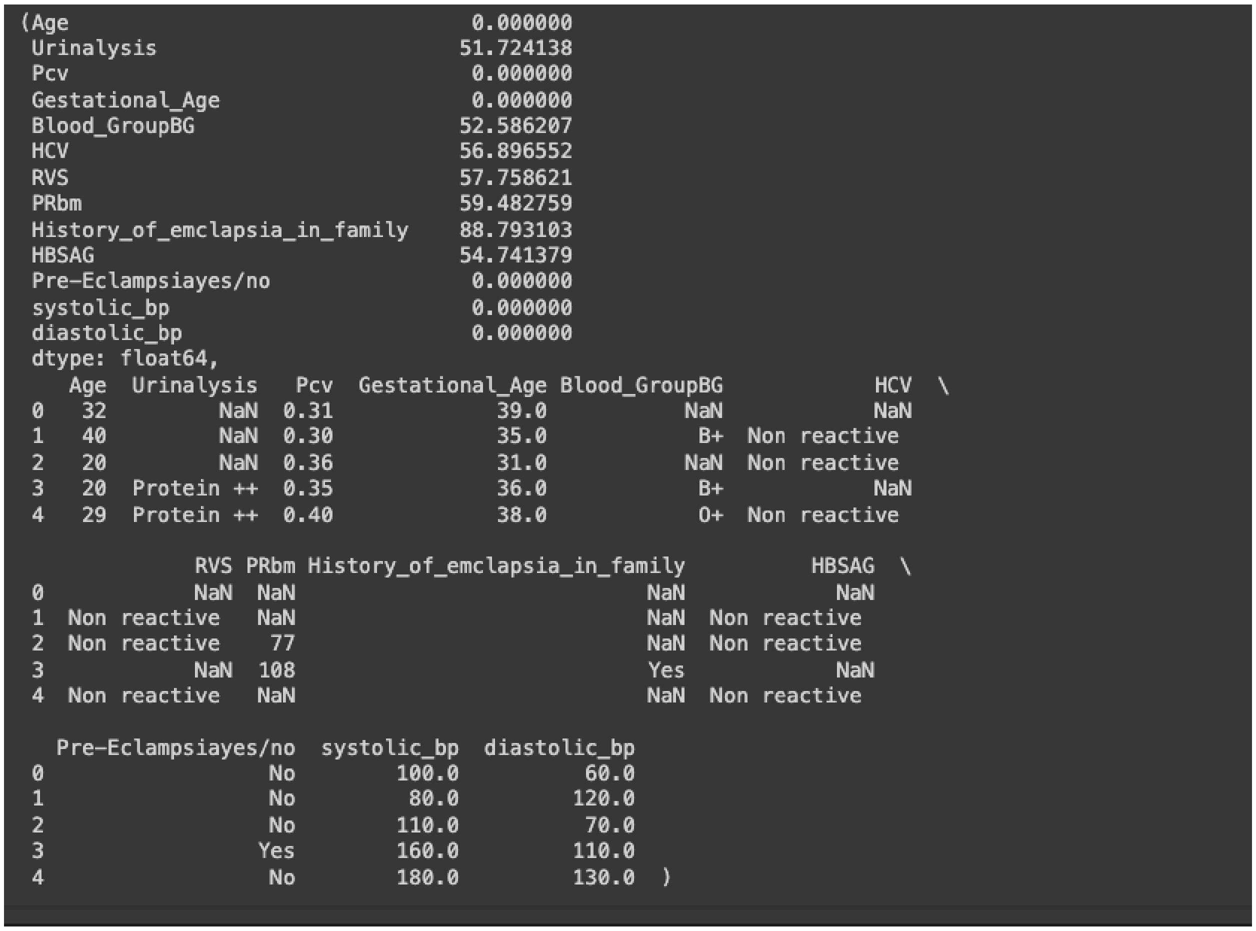
```
Column names: Index(['Age', 'Urinalysis', 'Pcv', 'Gestational Age', 'Blood Group(BG)', 'HCV',
       'RVS', 'PR(bm)', ' History of emclapsia in family', 'HBV', 'HBSAG',
       'Pre-Eclampsia(yes/no)', 'systolic ', 'systolic_bp', 'diastolic_bp'],
      dtype='object')
(Age                                  0
 Urinalysis                         120
 Pcv                                 25
 Gestational Age                     25
 Blood Group(BG)                    122
 HCV                                132
 RVS                                134
 PR(bm)                             138
  History of emclapsia in family    206
 HBV                                232
 HBSAG                              127
 Pre-Eclampsia(yes/no)                0
 systolic                           232
 systolic_bp                          2
 diastolic_bp                         2
 dtype: int64,
 Age                                   int64
 Urinalysis                         category
 Pcv                                 float64
 Gestational Age                     float64
 Blood Group(BG)                    category
 HCV                                category
 RVS                                category
 PR(bm)                               object
  History of emclapsia in family      object
 HBV                                category
 HBSAG                              category
 Pre-Eclampsia(yes/no)              category
 systolic                            float64
 systolic_bp                         float64
 diastolic_bp                        float64
 dtype: object)
```

```
# Simplifying column names
df.columns = df.columns.str.strip().str.replace(' ', '_').str.replace('(', '').str.replace(')', '')

# Dropping the 'systolic' column if it's a duplicate of 'systolic_bp'
if 'systolic' in df.columns and 'systolic_bp' in df.columns:
  if df['systolic'].equals(df['systolic_bp']):
     df.drop('systolic', axis=1, inplace=True)
  else:
     # If they are not duplicates, we need to decide what to do with these columns
     print("systolic and systolic_bp are not duplicates. Further inspection needed.")

# Dealing with missing values
# We'll first check the percentage of missing values in each column to decide on the approach
missing_percentage = df.isnull().sum() / len(df) * 100

missing_percentage
```



```
systolic and systolic_bp are not duplicates. Further inspection needed.
<ipython-input-6-63b94b75efd9>:2: FutureWarning: The default value of regex will change from True to False in a future version. In addition, single character regular expressions will *not* be treated as literal strings when regex=True.
  df.columns = df.columns.str.strip().str.replace(' ', '_').str.replace('(', '').str.replace(')', '')
<ipython-input-6-63b94b75efd9>:2: FutureWarning: The default value of regex will change from True to False in a future version. In addition, single character regular expressions will *not* be treated as literal strings when regex=True.
  df.columns = df.columns.str.strip().str.replace(' ', '_').str.replace('(', '').str.replace(')', '')
Age                                 0.000000
Urinalysis                         51.724138
Pcv                                10.775862
Gestational_Age                    10.775862
Blood_GroupBG                      52.586207
HCV                                56.896552
RVS                                57.758621
PRbm                               59.482759
History_of_emclapsia_in_family     88.793103
HBV                               100.000000
HBSAG                              54.741379
Pre-Eclampsiayes/no                 0.000000
systolic                          100.000000
systolic_bp                         0.862069
diastolic_bp                        0.862069
dtype: float64
```

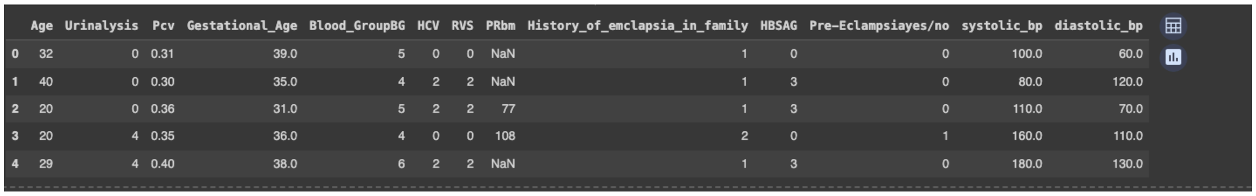
```python
# Dropping columns with a high percentage of missing values or irrelevance
columns_to_drop = ['HBV', 'systolic']
df.drop(columns=columns_to_drop, inplace=True)

# Deciding on a strategy for other columns with missing data
# For columns with a moderate amount of missing data, we'll use imputation
# For columns with a high percentage of missing data, we'll evaluate their importance

# Imputing missing values for 'Pcv' and 'Gestational_Age' with their median values
df['Pcv'].fillna(df['Pcv'].median(), inplace=True)
df['Gestational_Age'].fillna(df['Gestational_Age'].median(), inplace=True)

# Imputing missing values for blood pressure with median values
df['systolic_bp'].fillna(df['systolic_bp'].median(), inplace=True)
df['diastolic_bp'].fillna(df['diastolic_bp'].median(), inplace=True)

# Checking the updated dataset
updated_missing_percentage = df.isnull().sum() / len(df) * 100
updated_missing_percentage, df.head()
```



```
(Age                                 0.000000
 Urinalysis                         51.724138
 Pcv                                 0.000000
 Gestational_Age                     0.000000
 Blood_GroupBG                      52.586207
 HCV                                56.896552
 RVS                                57.758621
 PRbm                               59.482759
 History_of_emclapsia_in_family     88.793103
 HBSAG                              54.741379
 Pre-Eclampsiayes/no                 0.000000
 systolic_bp                         0.000000
 diastolic_bp                        0.000000
 dtype: float64,
    Age  Urinalysis   Pcv  Gestational_Age Blood_GroupBG            HCV  \
 0   32         NaN  0.31             39.0           NaN            NaN
 1   40         NaN  0.30             35.0            B+  Non reactive
 2   20         NaN  0.36             31.0           NaN  Non reactive
 3   20  Protein ++  0.35             36.0            B+            NaN
 4   29  Protein ++  0.40             38.0            O+  Non reactive

              RVS PRbm History_of_emclapsia_in_family          HBSAG  \
 0            NaN  NaN                            NaN            NaN
 1  Non reactive   NaN                            NaN  Non reactive
 2  Non reactive    77                            NaN  Non reactive
 3            NaN  108                            Yes            NaN
 4  Non reactive   NaN                            NaN  Non reactive

   Pre-Eclampsiayes/no  systolic_bp  diastolic_bp
 0                  No        100.0          60.0
 1                  No         80.0         120.0
 2                  No        110.0          70.0
 3                 Yes        160.0         110.0
 4                  No        180.0         130.0  )
```

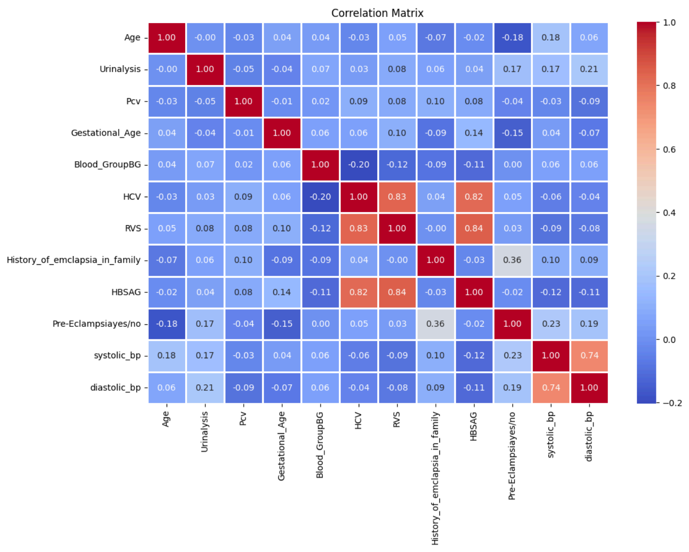
```
from sklearn.preprocessing import LabelEncoder
import pandas as pd

# Creating a label encoder object
le = LabelEncoder()

# Encoding categorical variables
categorical_columns = ['Urinalysis', 'Blood_GroupBG', 'HCV', 'RVS',
'History_of_emclapsia_in_family', 'HBSAG', 'Pre-Eclampsiayes/no']
for col in categorical_columns:
  if col in df.columns:
    # Convert to 'category' dtype and include 'Missing' in categories if not already included
    if df[col].dtype != 'category':
      df[col] = df[col].astype('category')
    if 'Missing' not in df[col].cat.categories:
```

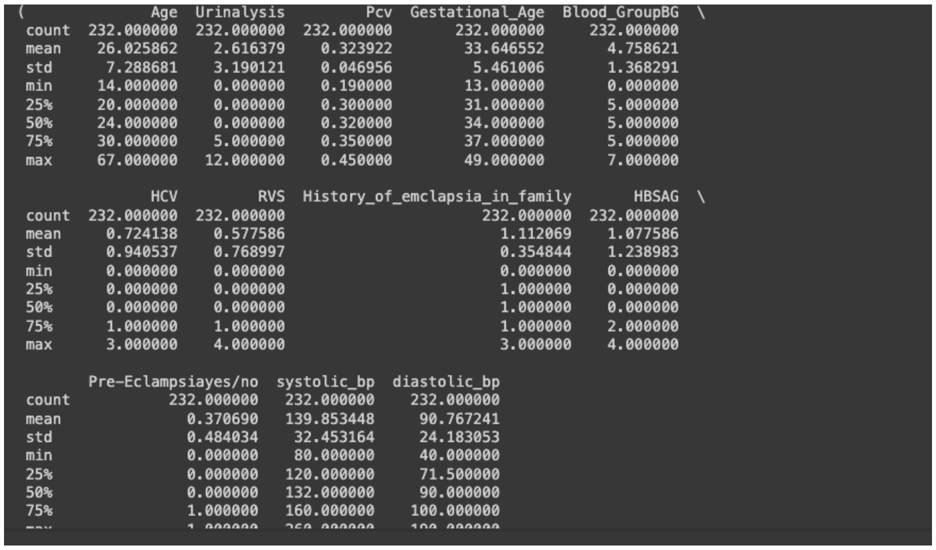
```python
        df[col] = df[col].cat.add_categories('Missing')

    # Fill missing values with 'Missing'
    df[col].fillna('Missing', inplace=True)

    # Encode the categorical data
    df[col] = le.fit_transform(df[col])

# Checking the dataset after encoding
encoded_data = df
encoded_data.head()
```



```
   Age  Urinalysis   Pcv  Gestational_Age  Blood_GroupBG  HCV  RVS  PRbm  History_of_emclapsia_in_family  HBSAG  Pre-Eclampsiayes/no  systolic_bp  diastolic_bp
0   32           0  0.31             39.0              5    0    0   NaN                               1      0                    0        100.0          60.0
1   40           0  0.30             35.0              4    2    2   NaN                               1      3                    0         80.0         120.0
2   20           0  0.36             31.0              5    2    2    77                               1      3                    0        110.0          70.0
3   20           4  0.35             36.0              4    0    0   108                               2      0                    1        160.0         110.0
4   29           4  0.40             38.0              6    2    2   NaN                               1      3                    0        180.0         130.0
```

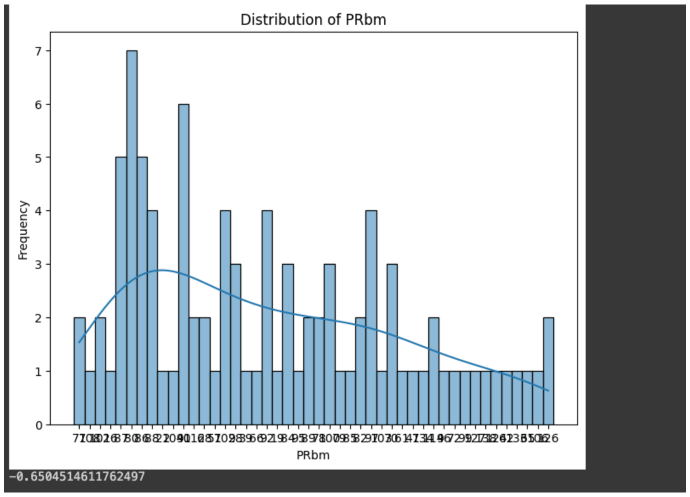
```python
import matplotlib.pyplot as plt
import seaborn as sns

# Basic Descriptive Statistics
descriptive_stats = encoded_data.describe()

# Correlation Matrix
correlation_matrix = encoded_data.corr()

# Plotting the Correlation Heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=2)
plt.title("Correlation Matrix")
plt.show()

descriptive_stats, correlation_matrix['Pre-Eclampsiayes/no'].sort_values(ascending=False)
```





```
(              Age  Urinalysis         Pcv  Gestational_Age  Blood_GroupBG  \
 count  232.000000  232.000000  232.000000       232.000000     232.000000
 mean    26.025862    2.616379    0.323922        33.646552       4.758621
 std      7.288681    3.190121    0.046956         5.461006       1.368291
 min     14.000000    0.000000    0.190000        13.000000       0.000000
 25%     20.000000    0.000000    0.300000        31.000000       5.000000
 50%     24.000000    0.000000    0.320000        34.000000       5.000000
 75%     30.000000    5.000000    0.350000        37.000000       5.000000
 max     67.000000   12.000000    0.450000        49.000000       7.000000

               HCV         RVS  History_of_emclapsia_in_family       HBSAG  \
 count  232.000000  232.000000                      232.000000  232.000000
 mean     0.724138    0.577586                        1.112069    1.077586
 std      0.940537    0.768997                        0.354844    1.238983
 min      0.000000    0.000000                        0.000000    0.000000
 25%      0.000000    0.000000                        1.000000    0.000000
 50%      0.000000    0.000000                        1.000000    0.000000
 75%      1.000000    1.000000                        1.000000    2.000000
 max      3.000000    4.000000                        3.000000    4.000000

        Pre-Eclampsiayes/no  systolic_bp  diastolic_bp
 count           232.000000   232.000000    232.000000
 mean              0.370690   139.853448     90.767241
 std               0.484034    32.453164     24.183053
 min               0.000000    80.000000     40.000000
 25%               0.000000   120.000000     71.500000
 50%               0.000000   132.000000     90.000000
 75%               1.000000   160.000000    100.000000
```

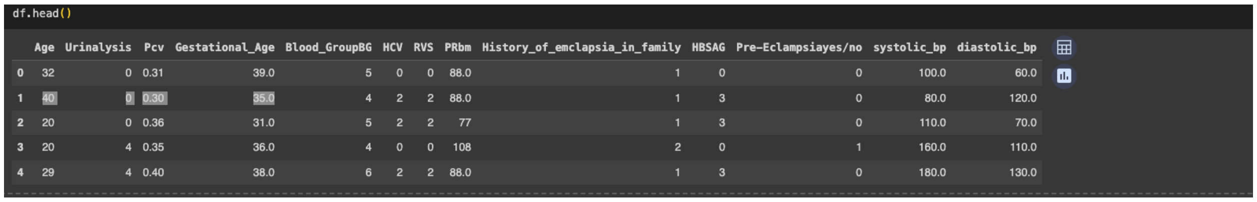
```
# Visualizing the distribution of 'PRbm'
plt.figure(figsize=(8, 6))
sns.histplot(encoded_data['PRbm'].dropna(), kde=True)
plt.title("Distribution of PRbm")
plt.xlabel("PRbm")
plt.ylabel("Frequency")
plt.show()

# Checking skewness
prbm_skewness = encoded_data['PRbm'].skew()
prbm_skewness
```



```
-0.6504514611762497
```

```
# Imputing missing values in 'PRbm' with its median value
prbm_median = encoded_data['PRbm'].median()
encoded_data['PRbm'].fillna(prbm_median, inplace=True)

# Checking if there are any more missing values in the dataset
remaining_missing_values = encoded_data.isnull().sum().sum()
remaining_missing_values
```

```
0
```



```
df.head()
```

	Age	Urinalysis	Pcv	Gestational_Age	Blood_GroupBG	HCV	RVS	PRbm	History_of_emclapsia_in_family	HBSAG	Pre-Eclampsiayes/no	systolic_bp	diastolic_bp
0	32	0	0.31	39.0	5	0	0	88.0	1	0	0	100.0	60.0
1	40	0	0.30	35.0	4	2	2	88.0	1	3	0	80.0	120.0
2	20	0	0.36	31.0	5	2	2	77	1	3	0	110.0	70.0
3	20	4	0.35	36.0	4	0	0	108	2	0	1	160.0	110.0
4	29	4	0.40	38.0	6	2	2	88.0	1	3	0	180.0	130.0

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Defining the features and target variable
X = encoded_data.drop('Pre-Eclampsiayes/no', axis=1)
y = encoded_data['Pre-Eclampsiayes/no']

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Feature Scaling
scaler = StandardScaler()
```

```
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

X_train_scaled.shape, X_test_scaled.shape, y_train.shape, y_test.shape
```

```
((162, 12), (70, 12), (162,), (70,))
```

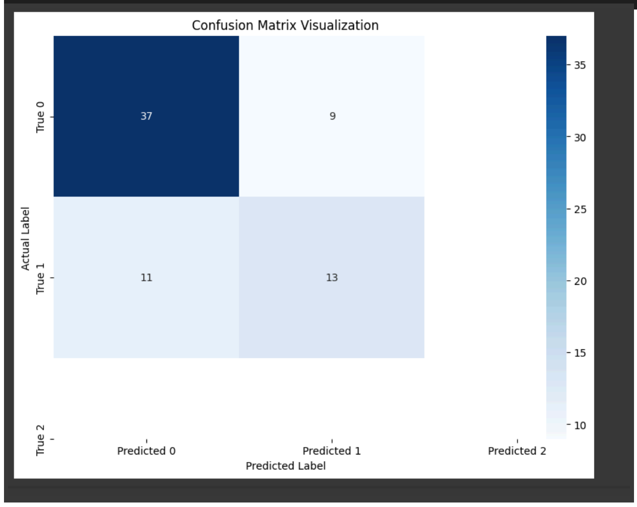
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, roc_auc_score,
confusion_matrix

# Building the Logistic Regression model
log_reg = LogisticRegression()
log_reg.fit(X_train_scaled, y_train)

# Making predictions on the test set
y_pred_log_reg = log_reg.predict(X_test_scaled)

# Evaluating the Logistic Regression model
accuracy_log_reg = accuracy_score(y_test, y_pred_log_reg)
precision_log_reg = precision_score(y_test, y_pred_log_reg)
recall_log_reg = recall_score(y_test, y_pred_log_reg)
roc_auc_log_reg = roc_auc_score(y_test, y_pred_log_reg)
conf_matrix_log_reg = confusion_matrix(y_test, y_pred_log_reg)

# Logistic Regression performance metrics
log_reg_metrics = {
  "Accuracy": accuracy_log_reg,
  "Precision": precision_log_reg,
  "Recall": recall_log_reg,
  "ROC AUC": roc_auc_log_reg
}

log_reg_metrics, conf_matrix_log_reg
```

```
({'Accuracy': 0.5857142857142857,
 'Precision': 0.4,
 'Recall': 0.4166666666666667,
 'ROC AUC': 0.5452898550724637},
array([[31, 15],
       [14, 10]]))
```

```
from sklearn.ensemble import RandomForestClassifier

# Building the Random Forest model
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train_scaled, y_train)

# Making predictions on the test set
y_pred_rf = rf.predict(X_test_scaled)

# Evaluating the Random Forest model
accuracy_rf = accuracy_score(y_test, y_pred_rf)
precision_rf = precision_score(y_test, y_pred_rf)
recall_rf = recall_score(y_test, y_pred_rf)
roc_auc_rf = roc_auc_score(y_test, y_pred_rf)
conf_matrix_rf = confusion_matrix(y_test, y_pred_rf)

# Random Forest performance metrics
rf_metrics = {
    "Accuracy": accuracy_rf,
    "Precision": precision_rf,
    "Recall": recall_rf,
    "ROC AUC": roc_auc_rf
}

rf_metrics, conf_matrix_rf
```

```
({'Accuracy': 0.7142857142857143,
 'Precision': 0.5909090909090909,
 'Recall': 0.5416666666666666,
 'ROC AUC': 0.6730072463768115},
array([[37,  9],
       [11, 13]]))
```

```
plt.figure(figsize=(10,7))
sns.heatmap(conf_matrix_rf, annot=True, fmt="d", cmap="Blues", xticklabels=['Predicted 0',
'Predicted 1', 'Predicted 2'], yticklabels=['True 0', 'True 1', 'True 2'])
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
plt.title('Confusion Matrix Visualization')
plt.show()
```



```
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier

# Create a Gradient Boosting classifier instance
clf = GradientBoostingClassifier()

# Now, you can use `clf` to fit data and make predictions
# clf.fit(X_train, y_train)
# predictions = clf.predict(X_test)

# Support Vector Machine Model
svm_model = SVC()
svm_model.fit(X_train_scaled, y_train)
y_pred_svm = svm_model.predict(X_test_scaled)

# Gradient Boosting Classifier Model
```

```
gb_model = GradientBoostingClassifier(random_state=42)
gb_model.fit(X_train_scaled, y_train)
y_pred_gb = gb_model.predict(X_test_scaled)

# K-Nearest Neighbors Model
knn_model = KNeighborsClassifier()
knn_model.fit(X_train_scaled, y_train)
y_pred_knn = knn_model.predict(X_test_scaled)

# Evaluating the models
models = [svm_model, gb_model, knn_model]
predictions = [y_pred_svm, y_pred_gb, y_pred_knn]
model_names = ['SVM', 'Gradient Boosting', 'K-Nearest Neighbors']
model_performance = {}

for i, model in enumerate(models):
    accuracy = accuracy_score(y_test, predictions[i])
    precision = precision_score(y_test, predictions[i])
    recall = recall_score(y_test, predictions[i])
    roc_auc = roc_auc_score(y_test, predictions[i])
    model_performance[model_names[i]] = {"Accuracy": accuracy, "Precision": precision, "Recall":
recall, "ROC AUC": roc_auc}

model_performance
```

```
{'SVM': {'Accuracy': 0.6571428571428571,
  'Precision': 0.5,
  'Recall': 0.2916666666666667,
  'ROC AUC': 0.5697463768115942},
 'Gradient Boosting': {'Accuracy': 0.6857142857142857,
  'Precision': 0.5384615384615384,
  'Recall': 0.5833333333333334,
  'ROC AUC': 0.6612318840579711},
 'K-Nearest Neighbors': {'Accuracy': 0.6285714285714286,
  'Precision': 0.4583333333333333,
  'Recall': 0.4583333333333333,
 'ROC AUC': 0.5878623188405797}}
```

```
from sklearn.model_selection import GridSearchCV

# Simplified parameter grid for Gradient Boosting Classifier
param_grid_gb_simplified = {
  'n_estimators': [100, 200],
  'learning_rate': [0.05, 0.1],
  'max_depth': [3, 4]
}

# Creating the simplified Grid Search for Gradient Boosting
gb_grid_search_simplified = GridSearchCV(GradientBoostingClassifier(random_state=42),
param_grid_gb_simplified, cv=5, scoring='accuracy')
gb_grid_search_simplified.fit(X_train_scaled, y_train)

# Best parameters and best score for Gradient Boosting
best_params_gb_simplified = gb_grid_search_simplified.best_params_
best_score_gb_simplified = gb_grid_search_simplified.best_score_

best_params_gb_simplified, best_score_gb_simplified
```

```
({'learning_rate': 0.1, 'max_depth': 4, 'n_estimators': 200},
 0.7477272727272728)
```

```
# Simplified parameter grid for Random Forest
param_grid_rf_simplified = {
  'n_estimators': [100, 200],
  'max_depth': [3, 5],
  'min_samples_split': [2, 4]
}

# Creating the simplified Grid Search for Random Forest
rf_grid_search_simplified = GridSearchCV(RandomForestClassifier(random_state=42),
param_grid_rf_simplified, cv=5, scoring='accuracy')
rf_grid_search_simplified.fit(X_train_scaled, y_train)

# Best parameters and best score for Random Forest
```

```
best_params_rf_simplified = rf_grid_search_simplified.best_params_
best_score_rf_simplified = rf_grid_search_simplified.best_score_

best_params_rf_simplified, best_score_rf_simplified
```

```
({'max_depth': 5, 'min_samples_split': 2, 'n_estimators': 200},
 0.7291666666666667)
```

```
# Creating and evaluating the tuned Gradient Boosting Classifier
tuned_gb_model = GradientBoostingClassifier(
  n_estimators=best_params_gb_simplified['n_estimators'],
  learning_rate=best_params_gb_simplified['learning_rate'],
  max_depth=best_params_gb_simplified['max_depth'],
  random_state=42
)
tuned_gb_model.fit(X_train_scaled, y_train)
y_pred_tuned_gb = tuned_gb_model.predict(X_test_scaled)

# Performance metrics for the tuned Gradient Boosting Classifier
accuracy_tuned_gb = accuracy_score(y_test, y_pred_tuned_gb)
precision_tuned_gb = precision_score(y_test, y_pred_tuned_gb)
recall_tuned_gb = recall_score(y_test, y_pred_tuned_gb)
roc_auc_tuned_gb = roc_auc_score(y_test, y_pred_tuned_gb)

tuned_gb_metrics = {
  "Accuracy": accuracy_tuned_gb,
  "Precision": precision_tuned_gb,
  "Recall": recall_tuned_gb,
  "ROC AUC": roc_auc_tuned_gb
}

tuned_gb_metrics
```

```
{'Accuracy': 0.7,
 'Precision': 0.56,
 'Recall': 0.5833333333333334,
 'ROC AUC': 0.6721014492753624}
```

```
# Creating and evaluating the tuned Random Forest model
tuned_rf_model = RandomForestClassifier(
    n_estimators=best_params_rf_simplified['n_estimators'],
    max_depth=best_params_rf_simplified['max_depth'],
    min_samples_split=best_params_rf_simplified['min_samples_split'],
    random_state=42
)
tuned_rf_model.fit(X_train_scaled, y_train)
y_pred_tuned_rf = tuned_rf_model.predict(X_test_scaled)

# Performance metrics for the tuned Random Forest
accuracy_tuned_rf = accuracy_score(y_test, y_pred_tuned_rf)
precision_tuned_rf = precision_score(y_test, y_pred_tuned_rf)
recall_tuned_rf = recall_score(y_test, y_pred_tuned_rf)
roc_auc_tuned_rf = roc_auc_score(y_test, y_pred_tuned_rf)

tuned_rf_metrics = {
    "Accuracy": accuracy_tuned_rf,
    "Precision": precision_tuned_rf,
    "Recall": recall_tuned_rf,
    "ROC AUC": roc_auc_tuned_rf
}

tuned_rf_metrics
```

```
{'Accuracy': 0.7142857142857143,
 'Precision': 0.6111111111111112,
 'Recall': 0.4583333333333333,
 'ROC AUC': 0.6530797101449275}
```

```
import pickle
pickle.dump(tuned_rf_model, open('model.pkl', 'wb'))
```

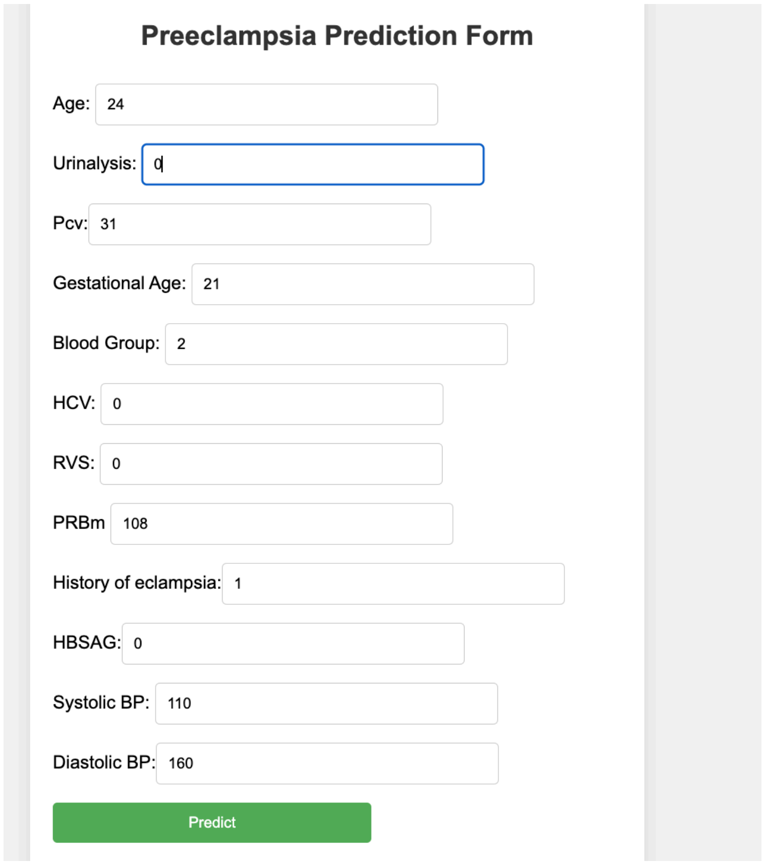
2. app.py

```python
from flask import Flask, request, render_template
import pickle

app = Flask(__name__)

# Load the model
model = pickle.load(open("model.pkl","rb"))



@app.route('/', methods=['GET', 'POST'])
def predict():
  prediction = ''
  if request.method == 'POST':
    # Extract features from the form
    feature1 = request.form.get('Age', type=float)
    feature2 = request.form.get('Urinalysis', type=float)
    feature3 = request.form.get('Pcv', type=float)
    feature4 = request.form.get('Gestational_Age', type=float)
    feature5 = request.form.get('Blood_GroupBG', type=float)
    feature6 = request.form.get('HCV', type=float)
    feature7 = request.form.get('RVS', type=float)
    feature8 = request.form.get('PRbm', type=float)
    feature9 = request.form.get('History_of_emclapsia_in_family', type=float)
    feature10 = request.form.get('HBSAG', type=float)
    feature11 = request.form.get('systolic_bp', type=float)
    feature12 = request.form.get('diastolic_bp', type=float)




    # Make prediction
    prediction = model.predict([[feature1, feature2, feature3, feature4, feature5, feature6, feature7,
feature8, feature9, feature10, feature11, feature12]])[0]
    if prediction == 0:
      prediction = f'Preeclampsia not detected {prediction}'
    else: prediction = f'Preeclampsia not detected {prediction}'
  return render_template('index.html', prediction=prediction)



if __name__ == '__main__':
  app.run(debug=True)
```

3. Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Preeclampsia Prediction Form</title>
<style>
  body {
     font-family: Arial, sans-serif;
     background-color: #f0f0f0;
     display: flex;
     justify-content: center;
     align-items: center;
     /*height: 100vh;*/
     margin: 0;
  }
  .form-container {
     background-color: #ffffff;
     padding: 20px;
     border-radius: 8px;
     box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
     width: 100%;
     max-width: 500px; /* Adjust this value as needed */
  }
  h2 {
     color: #333;
     text-align: center;
  }
  input[type="text"], input[type="submit"] {
     width: calc(60% - 20px); /* Adjust width to account for padding */
     padding: 10px;
     margin: 8px 0;
     border: 1px solid #ccc;
     border-radius: 4px;
  }
  input[type="submit"] {
     background-color: #4CAF50;
     color: white;
     cursor: pointer;
     border: none;
  }
  input[type="submit"]:hover {
     background-color: #45a049;
  }
  label {
     font-weight: bold;
     display: block;
     margin-top: 10px;
  }
</style>
</head>
<body>

  <div class="form-container">
     <h2>Preeclampsia Prediction Form</h2>
     <form method="post" action="/">
<!--           <label for="Age">Age:</label>-->
          Age: <input type="text" id="Age" name="Age" required><br>
           Urinalysis: <input type="text" id="Urinalysis" name="Urinalysis" required><br>
          Pcv:<input type="text" id="Pcv" name="Pcv" required><br>
           Gestational Age: <input type="text" id="Gestational_Age" name="Gestational_Age" required><br>
           Blood Group: <input type="text" id="Blood_GroupBG" name="Blood_GroupBG" required><br>
           HCV: <input type="text" id="HCV" name="HCV" required><br>
          RVS: <input type="text" id="RVS" name="RVS" required><br>
          PRBm <input type="text" id="PRbm" name="PRbm" required><br>
```

```
        History of eclampsia:<input type="text" id="History_of_emclapsia_in_family"
name="History_of_emclapsia_in_family" required><br>
        HBSAG:<input type="text" id="HBSAG" name="HBSAG" required><br>
        Systolic BP: <input type="text" id="systolic_bp" name="systolic_bp" required><br>
        Diastolic BP:<input type="text" id="diastolic_bp" name="diastolic_bp" required>
         <input type="submit" value="Predict">
      </form>
   </div>
<br>
   {{ prediction }}
</body>
</html>
```



Preeclampsia Prediction Form

Age:	24
Urinalysis:	0
Pcv:	31
Gestational Age:	21
Blood Group:	2
HCV:	0
RVS:	0
PRBm	108
History of eclampsia:	1
HBSAG:	0
Systolic BP:	110
Diastolic BP:	160