**Data Intensive Computing**
**Project Phase - 3**

| Team Members Names | UBIT Name | UB Person Number |
|---|---|---|
| Harshavardhan Reddy Nadedi | hnadedi | 50559985 |
| Likhith Kongara | lkongara | 50560571 |
| Sai Sohan Kosaraju | saisohan | 50560534 |

**Table of contents:**

## Code Documentation:

a. **Phase 1**

1. We start by getting our data ready and looking at it closely.

2. First, we load the data and clean it up. We have a step-by-step plan with a 10-step process outlined in the Jupyter Notebook.

3. Steps include converting columns to numeric types, handling null values, removing duplicates, ensuring consistent date formats, eliminating unwanted characters and white spaces, performing data scaling, feature selection, and standardizing selected columns.

4. Exploration follows John Tukey's EDA principles.

5. Various visualization techniques are employed:

- Pair plots, histograms, and box plots illustrate data distribution across features.
- Temporal trends in multiple features are plotted over time.
- Inter-feature correlations are analyzed to understand relationships between see how things are spread out across different parts of the data.

6. We also plotted the visualizations of how things change over time, looking for any interesting patterns.

7. We also check out how different parts of the data are connected to each other.

9. Finally, we use correlation matrices to examine feature relationships and their impact on the dataset's structure and analysis.

**b. Phase 2**

In Phase 2, We expand upon the groundwork established in Phase 1. by incorporating our data-cleaning code right from the start. This ensures a seamless transition into more advanced recommendation models. Here's a breakdown of what we do:

i. We integrate the cleaned data in phase 1 as the input for phase 2 by inducing some models.

ii. We first split the cleaned_data into 2 sets one for testing and one for training using sklearn library.

iii. After splitting we imported the logistic regression from sklearn and trained the model using the train_data and and tried to predict the outcomes for the test_data.

iv. We also evaluated the performance of the model using different metrics like accuracy, precision, f1 score, etc.

v. Later on we tried doing the same for different models training the model, testing it with the test_data, and calculating the metrics. For more ease in understanding, we constructed the confusion_matrix.

vi. On moving further we also applied some models like Naive Bayes, KNN, and neural networks on the train_data to ensure better performance.

vii. We got better performance with good accuracy and precision with the neural model. The neural model which we used is MLPClassifier.

viii. To understand more deeply and gain insights on which model is doing well we generated graphs for different models against their performances too which resulted in giving the neural model a higher end.

c. **Phase 3**

We've utilized Streamlit for deploying our code, making it accessible via a user-friendly web application.

ii. Within each function, Every component of the online program has Streamlit code integrated into it, ensuring a seamless user experience. The backend connection for each model has been established correctly, guaranteeing smooth functionality.

iii. Following the provided instructions, running the code allows users to interact with the application through a user interface, enabling them to easily navigate and utilize its features.

**User Interface working instruction:**

Use the below commands to run our diabetes disease prediction application in streamlit web user interface.

1. Install the streamlit python module with the command → pip install Streamlit

2. Then run our application code as a Python module with the command → streamlit run app.py or python -m streamlit run src/phase3/app.py

3. The user is provided with slide bar options to enter the input for all the required fields which displays the output dynamically. Users can input features such as age, BMI, glucose level, HbA1c level, and gender, and the model predicts the probability of diabetes and the chance of occurrence of diabetes for specific input values given by the user.

4. For filtering data according to some features like age, gender, glucose level, etc., and to gain more EDA outcomes we also created a Python application to filter data and visualize the data. and to run that application → python -m streamlit run eda.py or streamlit run eda.py.

5.After running the above py files to view them use below urls

Local URL: http://localhost:8502
Network URL: http://192.168.1.151:8502 for eda.py
Local URL: http://localhost:8501
Network URL: http://192.168.1.151:8501 for app.py

6. You can also look at README and requirements file for more detail

## Models usage and future recommendations

### Models' Usage:

The logistic regression model is utilized for predicting the likelihood of diabetes based on input health metrics.
It's deployed in the Streamlit application to provide users with predictions and feedback on their health metrics. Users can input features such as age, BMI, glucose level, HbA1c level, and gender, and the model predicts the probability of diabetes.

Our Application allows for the exploration and analysis of the diabetes dataset. It provides functionality to load the dataset, apply filters, and visualize the data using various plots.
Users can gain insights into the distribution of features, relationships between variables, and overall patterns within the dataset.

**Future Recommendations:**

Continuously evaluate the performance of the logistic regression model and explore opportunities for improvement.

Gather user feedback on the Streamlit application to identify areas for enhancement or additional features. Iterate the application based on user feedback to improve usability, functionality, and overall user experience.
Regularly update the dataset used for model training to incorporate new data and ensure model relevance. Monitor changes in health metrics or demographic trends that may impact the model's performance and update it accordingly.

The diabetes prediction application, powered by a logistic regression model and supported by exploratory data analysis (EDA), has potential real-world applications and can benefit various stakeholders:

Encourage users to learn more about diabetes, its risk factors, and preventive measures. It helps how lifestyle choices impact their risk of developing diabetes.

Individuals who are concerned about their risk of developing diabetes can use the application to assess their likelihood based on personal health metrics.
They can input their age, BMI, glucose level, HbA1c level, and gender to receive a prediction and feedback on their health status. This empowers individuals to take proactive steps towards managing their health and making informed decisions about lifestyle choices.

Healthcare professionals, including doctors, nurses, and nutritionists, can utilize the application to support patient education and engagement. By integrating the diabetes prediction tool into clinical practice, healthcare professionals can enhance patient-centered care and promote preventive healthcare initiatives.
They can also use the EDA tool to gain insights from the diabetes dataset, inform treatment strategies, and identify trends or patterns in diabetes prevalence and risk factors.

Public health authorities and policymakers can leverage the insights generated from the diabetes prediction application to inform public health interventions and policies. By understanding population-level trends in diabetes risk factors and prevalence, they can develop targeted prevention programs and allocate resources more effectively.
The EDA tool can aid in epidemiological research, surveillance, and monitoring of diabetes trends at the regional, national, or global level.

Researchers in the field of diabetes epidemiology, public health, and data science can utilize the application and dataset for further analysis and research. The availability of a pre-trained logistic regression model and exploratory data analysis tools accelerates research efforts and facilitates collaboration in diabetes research.
Community health organizations, non-profit groups, and patient advocacy organizations can promote the use of the diabetes prediction application to raise awareness and support diabetes prevention efforts within communities. They can integrate the application into health promotion campaigns, educational initiatives, and outreach activities to empower individuals to take control of their health and reduce their risk of developing diabetes.

## Diabetes Prediction Application Implementation

### Overview

This project aims to evaluate the probability of diabetes onset using user-provided health metrics. Utilizing a Logistic Regression model, renowned for its effectiveness in binary classification tasks, the application predicts diabetes likelihood. Developed with Streamlit, the platform offers a user-friendly web interface, facilitating effortless interaction and real-time predictions.

**Its features include:**
**User-Friendly Interface:** The application employs Streamlit to offer a straightforward and intuitive user interface, enabling users to input their health metrics effortlessly.
Immediate Predictions: Users receive instant predictions regarding their likelihood of developing diabetes and probabilities of Type 2 diabetes based on the entered health metrics.
Usage
**Enter Health Information:** Users input their health metrics using interactive sliders for Age, BMI, Glucose Level, and HbA1c Level, facilitating easy data entry.
Receive Predictions: The application calculates and presents users with their diabetes probability and compares their health metrics to normal ranges, offering insights into potential risks.

**Technologies Used**
**Python:** Serves as the primary programming language for development.
Streamlit: Utilized as the app framework to create the web interface, ensuring seamless interaction.
**Pandas & Numpy:** Employed for data manipulation and numerical computations, facilitating efficient data handling.
**Scikit-Learn:** Utilized for the implementation of the Logistic Regression model, providing robust tools for machine learning task
Models
During the training of the Logistic Regression model, the following considerations were made:
**Max Iterations:** Set to 1000 to ensure convergence of the optimization process.
**Features Used:** The model incorporates age, BMI, glucose levels, HbA1c levels, gender, and smoking history as input features, allowing for a comprehensive assessment of diabetes likelihood.
**Coefficient Analysis:** Conducted to discern the impact of each feature on the model's predictions, providing valuable insights into the factors contributing to diabetes risk.
Installation
**Python 3.8 or later.**
Installation of Python and necessary libraries.
Navigate to the project folder. Run pip install -r requirements.txt
Running the Application
To run the application, execute the following command in your terminal: streamlit run app.py

**Implementation**

**Imports:**

- **streamlit as st**: This line imports the Streamlit library and assigns it the alias st. Streamlit is a Python framework for building data apps.
- **pandas as pd**: This line imports the Pandas library and assigns it the alias pd. Pandas is used for data manipulation and analysis.
- **joblib**: This line imports the joblib library, used for loading pre-trained models and scalers.
- **from PIL import Image**: This line imports the Image class from the Pillow (PIL Fork) library used for image processing.

**Image Loading and Layout:**

- img = Image.open("glucometer.png"): This line opens an image named "glucometer.png" and stores it in the variable img.
- c1, c2, c3 = st.columns(3): This line creates a three-column layout using Streamlit's columns function. The variables c1, c2, and c3 represent the three columns, respectively.
- The following code block uses these columns for layout:
  - with c1:: This line defines content to be placed in the first column (c1). However, it simply adds an empty space with st.write(' ').
  - with c2:: This line defines content to be placed in the second column (c2). It uses st.image(img, width=200) to display the loaded image with a width of 200 pixels.
  - with c3:: This line defines content to be placed in the third column (c3). Similar to the first column, it adds an empty space with st.write(' ').

**Model Loading:**

- def load_models():: This line defines a function named load_models that loads pre-trained models used for diabetes prediction.
  - model = joblib.load('model.pkl'): This line loads a model saved in a file named "model.pkl" using joblib and stores it in the variable model.
  - preprocessor = joblib.load('preprocessor.pkl'): This line loads a preprocessor saved in a file named "preprocessor.pkl" using joblib and stores it in the variable preprocessor. Preprocessors are often used to scale or transform data before feeding it to a machine learning model.
  - return model, preprocessor: This line returns both the loaded model and preprocessor from the function.

**Main App Function:**

- def diabetes_app():: This line defines the main function of the Streamlit app named diabetes_app.

**App Styling:**

- The following code block defines custom CSS styles using Streamlit's markdown function with the unsafe_allow_html argument set to True. This allows embedding HTML code for styling purposes (**Caution:** Using this with untrusted content poses security risks).
  - The first block sets the background color and font styles for the entire app.
  - The second block styles the Streamlit app container itself, changing its background color, text color, and adding padding and rounded corners.

**App Title and User Input:**

- st.title('Diabetes Prediction Tool'): This line displays the app title as "Diabetes Prediction Tool".
- st.sidebar.title('User Input'): This line creates a sidebar section with the title "User Input" for user interaction.
- The following lines in the sidebar define slider elements using st.sidebar.slider for users to input their age, BMI, glucose level, HbA1c level, and select their gender from a radio button (st.sidebar.radio). Each slider has its own minimum and maximum values, along with a step size for adjustments.

**Data Preparation:**

- user_input_df = pd.DataFrame({...}): This line creates a Pandas DataFrame named user_input_df to store user-provided data. It includes columns for age, BMI, HbA1c level, blood glucose level, gender, location (set to a default value), and smoking history (set to a default value).

**Prediction and Feedback:**

- diabetes_probability = predict_diabetes(user_input_df): This line calls the predict_diabetes function (defined later) with the user_input_df as input. The function likely uses the model to predict the probability of diabetes for the provided data and stores the result in diabetes_probability.
- type_2_diabetes_probability = diabetes_probability * 0.7: This line calculates the probability of type 2 diabetes, assuming it's

**Backend/Model Building**

**Chosen Model for Prediction:**
**Logistic regression:**
Logistic Regression was chosen as the algorithm for this problem due to its suitability for binary classification tasks, which aligns with our target variable having two possible outcomes.

The decision was based on the observed linearity between features and the target variable within our dataset. Additionally, Logistic Regression offers several advantages for our analysis:

**Interpretability:** The results of Logistic Regression are easily interpretable, due to the coefficients clearly showing the kind and degree of the link between the independent and dependent variables. This facilitates understanding the factors influencing diabetes prevalence.

**Efficiency:** Large datasets can be handled with comparatively little computer effort thanks to the computational efficiency of logistic regression., which is advantageous considering the volume of data involved in our study.

**Regularization:** The algorithm supports regularization techniques like L1 (Lasso) and L2 (Ridge), enabling us to mitigate overfitting by penalizing large coefficient values. To optimize the model's performance, extensive tuning and training were conducted. This involved adjusting regularization parameters, feature selection, and handling imbalanced data. The effectiveness of the algorithm was assessed using relevant metrics:

**Accuracy:** The model achieved an accuracy of 95.03%, indicating that it correctly classified the majority of test samples, reflecting strong overall performance.

**Precision:** With a precision of 87.55%, the model provides a high level of correctness when predicting the positive class, crucial for identifying individuals at risk of diabetes.
**F1 Score:** The F1 score of 72.99% signifies a good balance between precision and recall, suggesting the model's effectiveness in capturing both true positives and minimizing false positives.

## Explanation of the Backend Code for Diabetes Prediction Report

This code performs the following steps to train a logistic regression model for predicting diabetes:

### 1. Data Loading and Preprocessing:

- **Libraries:** Imports necessary libraries like streamlit, pandas, joblib, seaborn, matplotlib, and numpy.
- **Data Loading:** Reads the diabetes dataset from a CSV file named "diabetes_dataset.csv" using pd.read_csv.

- **Data Exploration (Optional):** Prints the first few rows of the data using data.head().
- **Data Cleaning:**
  - Converts categorical columns (gender, smoking_history, and location) to categorical data types using astype('category').
  - Converts columns year and age to appropriate data types (int and float).
  - Fills missing values (if any) using forward fill (method='ffill').

**2. Feature Engineering:**

- **Train-Test Split:** Splits the data into training and testing sets using train_test_split from sklearn.model_selection. This allows for evaluating the model's performance on unseen data.
  - X represents the feature matrix (all features except the target variable).
  - y represents the target variable (diabetes).

**3. Data Preprocessing Pipeline:**

- Defines two categories of features:
  - **Numerical Features:** year, age, bmi, hbA1c_level, and blood_glucose_level.
  - **Categorical Features:** gender, location, and smoking_history.
- Creates a preprocessing pipeline using ColumnTransformer from sklearn.compose.
- Defines transformers for each category:
  - **Numerical Transformer:** Uses StandardScaler to standardize numerical features (scaling them to have a mean of 0 and a standard deviation of 1).
  - **Categorical Transformer:** Uses OneHotEncoder to encode categorical features into one-hot encoded vectors. This helps the model understand categorical data effectively.
- Fits the preprocessor on the training data (X_train).

**4. Model Training:**

- Defines a logistic regression model using LogisticRegression from sklearn.linear_model.
- Sets the maximum number of iterations (max_iter) to 1000.
- Trains the model on the preprocessed training data (X_train_prepared, y_train).

**5. Model Evaluation:**

- Predicts diabetes labels for the test data using the trained model (model.predict).
- Calculates the model's accuracy using accuracy_score from sklearn.metrics. Accuracy measures the proportion of correctly predicted labels.
- Calculates the model's Area Under the ROC Curve (ROC AUC) using roc_auc_score. ROC AUC is a performance metric for binary classification models.
- Prints the accuracy and ROC AUC scores.

**6. Model and Preprocessor Persistence:**

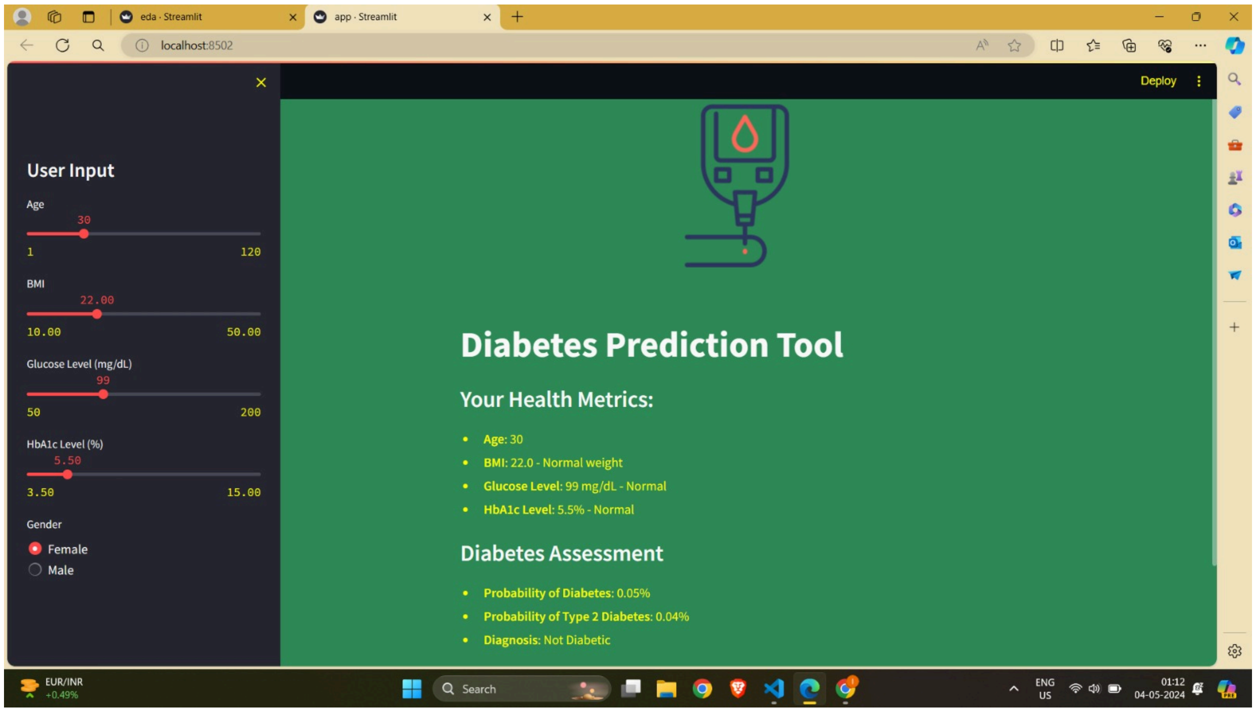
- Saves the trained model (model) and the preprocessor (preprocessor) using joblib.dump for future use.

**7. Feature Importance Analysis (Optional):**

- Retrieves feature names from the preprocessor's categorical transformer.
- Gets the model's coefficients, representing the weight of each feature in the model's decision function.
- Creates a DataFrame to store feature names, coefficients, and absolute coefficients.
- Sorts the DataFrame by absolute coefficient values in descending order.
- Prints the top 10 most important features based on their absolute coefficients.

**Note:** This code assumes the existence of a CSV file named "diabetes_dataset.csv" containing the diabetes data. You'll need to replace this filename with the actual path to your data file.

This explanation provides a detailed breakdown of the backend code for training a logistic regression model to predict diabetes. It highlights the data loading, preprocessing, model training, evaluation, and feature importance analysis steps. This information can be used to create a comprehensive report on the development of the diabetes prediction model.



User Input

Age: 30 (1 – 120)

BMI: 22.00 (10.00 – 50.00)

Glucose Level (mg/dL): 99 (50 – 200)

HbA1c Level (%): 5.50 (3.50 – 15.00)

Gender: [✓ Female] Male

**Diabetes Prediction Tool**

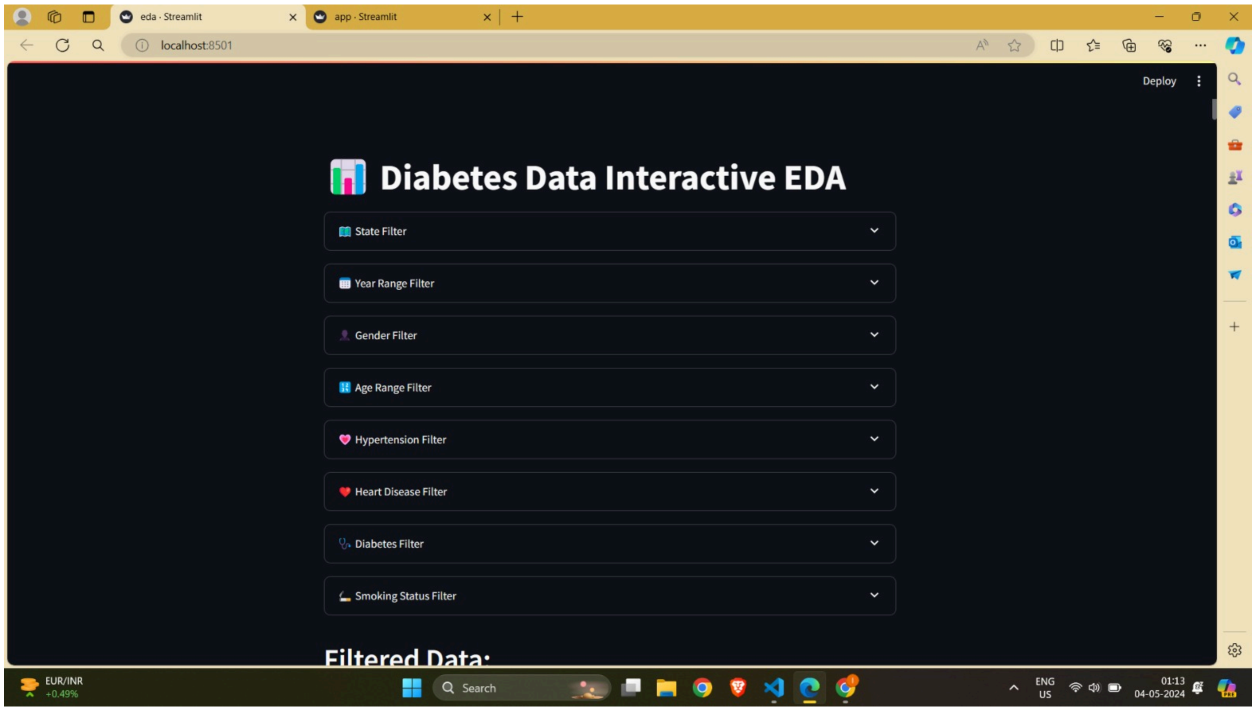
**Your Health Metrics:**

- **Age:** 30
- **BMI:** 22.0 - Normal weight
- **Glucose Level:** 99 mg/dL - Normal
- **HbA1c Level:** 5.5% - Normal

**Diabetes Assessment**

- **Probability of Diabetes:** 0.05%
- **Probability of Type 2 Diabetes:** 0.04%
- **Diagnosis:** Not Diabetic

Fig: FrontEnd page for taking user input and providing predicted output dynamically

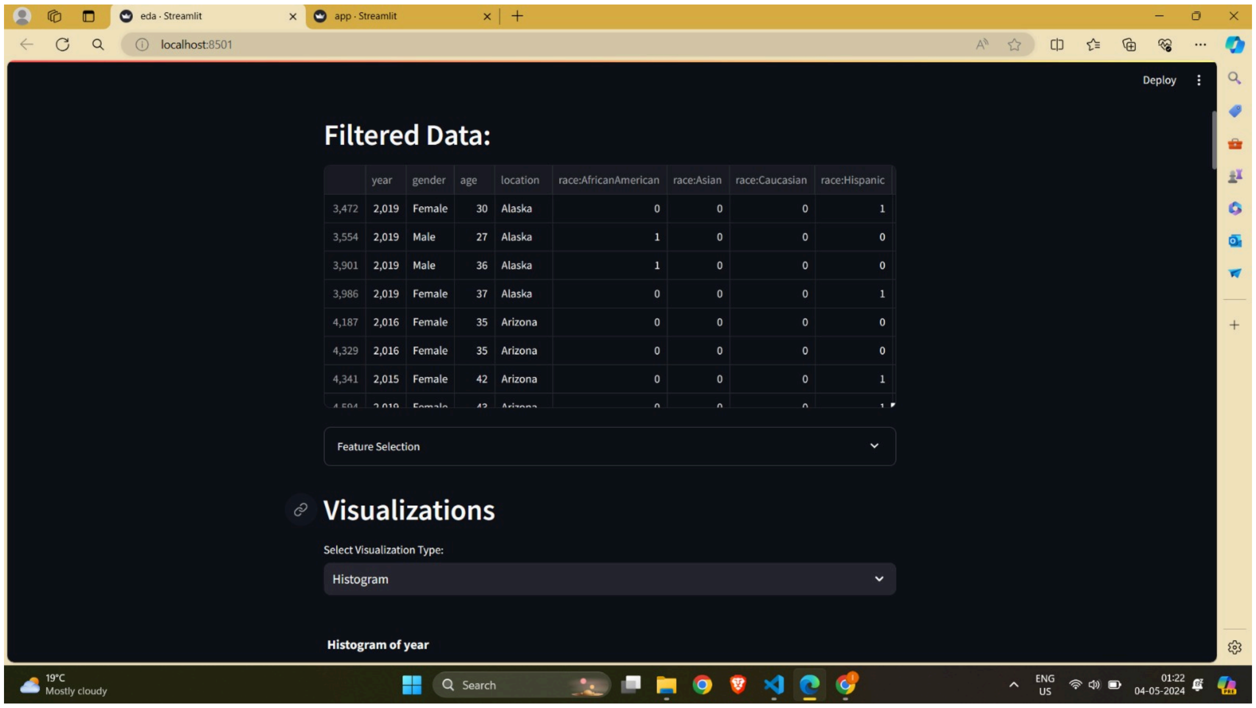


# Diabetes Data Interactive EDA

- State Filter
- Year Range Filter
- Gender Filter
- Age Range Filter
- Hypertension Filter
- Heart Disease Filter
- Diabetes Filter
- Smoking Status Filter

Filtered Data:

Fig: different options to filter the dataset



## Filtered Data:

| | year | gender | age | location | race:AfricanAmerican | race:Asian | race:Caucasian | race:Hispanic |
|---|---|---|---|---|---|---|---|---|
| 3,472 | 2,019 | Female | 30 | Alaska | 0 | 0 | 0 | 1 |
| 3,554 | 2,019 | Male | 27 | Alaska | 1 | 0 | 0 | 0 |
| 3,901 | 2,019 | Male | 36 | Alaska | 1 | 0 | 0 | 0 |
| 3,986 | 2,019 | Female | 37 | Alaska | 0 | 0 | 0 | 1 |
| 4,187 | 2,016 | Female | 35 | Arizona | 0 | 0 | 0 | 0 |
| 4,329 | 2,016 | Female | 35 | Arizona | 0 | 0 | 0 | 0 |
| 4,341 | 2,015 | Female | 42 | Arizona | 0 | 0 | 0 | 1 |
| 4,594 | 2,019 | Female | 43 | Arizona | 0 | 0 | 0 | 1 |

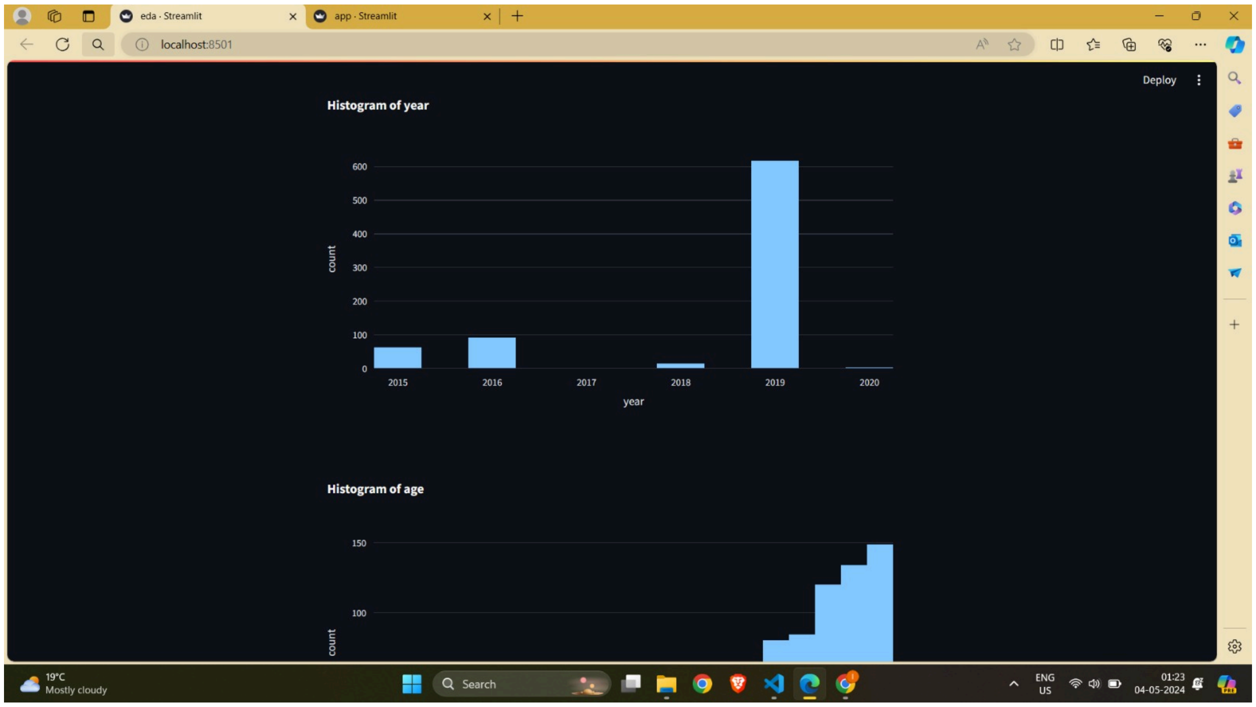
Feature Selection

## Visualizations

Select Visualization Type: Histogram

Histogram of year

Fig: gives us the data and its visualizations for the different filter options we choose.



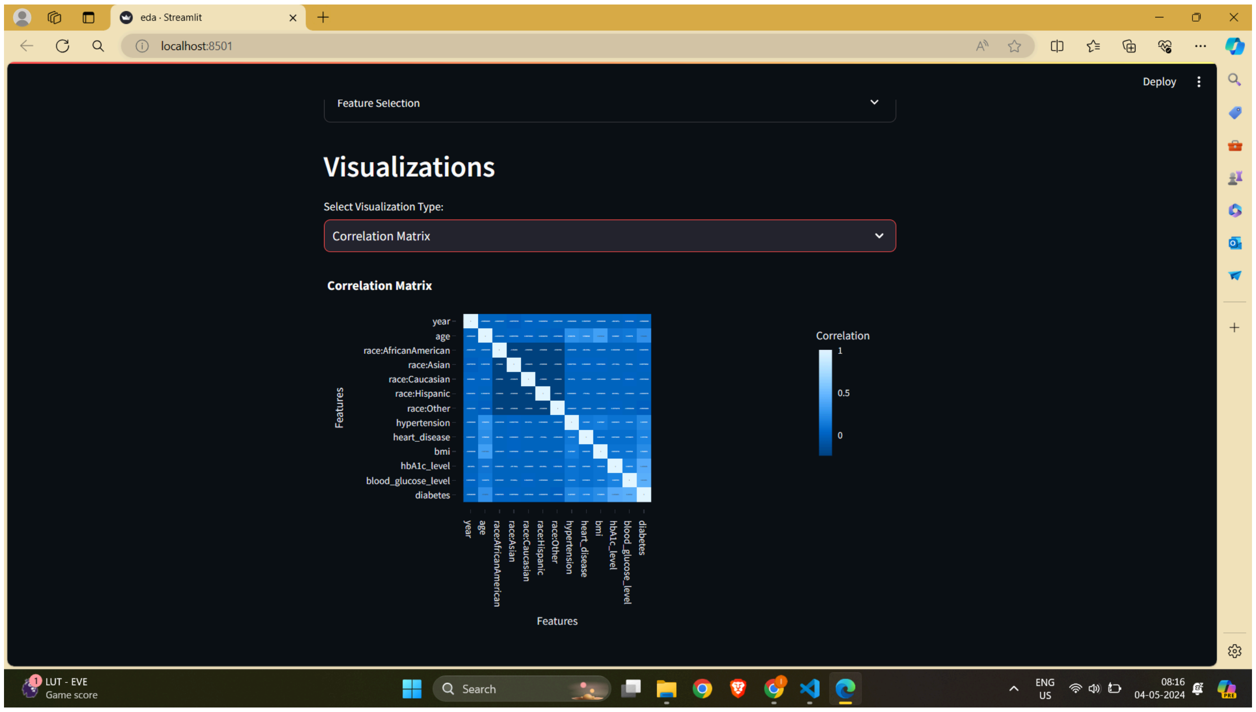


Fig : We can choose different visualization graphs from the dropdown and hit enter for it to render according to the data.

## 4. References

1. Professor’s Lecture slides and demo resources
2. https://docs.streamlit.io/
3.https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
4.https://towardsdatascience.com/how-to-build-your-first-machine-learning-model-in-python-e70fd1907cdd
5. https://www.datacamp.com/tutorial/guide-to-data-cleaning-in-python
6.https://www.w3schools.com/python/numpy/numpy_intro.asp
7. https://www.analyticsvidhya.com/blog/2021/02/introduction-to-exploratory-data-analysis-eda/
8. https://numpy.org/learn/
9. https://www.w3schools.com/python/pandas/default.asp
10. https://matplotlib.org/3.4.3/contents.html
11.https://blog.streamlit.io/how-to-master-streamlit-for-data-science/#6-deploy-your-streamlit-app
12. https://joblib.readthedocs.io/en/stable/