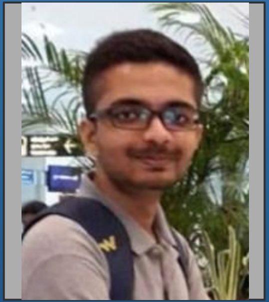
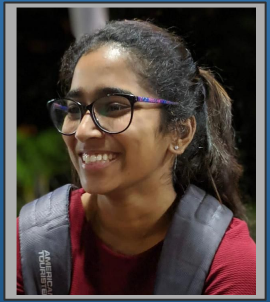
B1-01



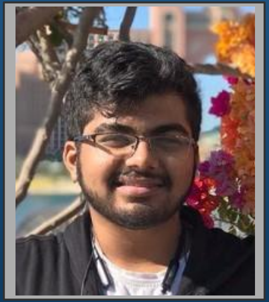
Paleti Krishnasai - CED18I039
Lead Developer

Supervises and Leads Development of the Medical Assistant



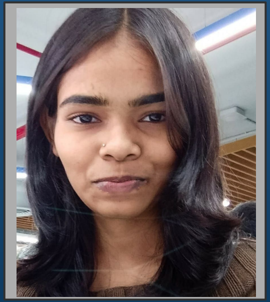
Uppalapati Pranita - CED18I062
Developer

Works on the development of the Backend of the Application



Hrishikesh R Menon - COE18B024
Lead Designer

Concept Designer and Main Developer of FrontEnd



Preety Banjare - COE17B038
Concept Designer

Concept and UI/UX Designer

EPIONE

RECOVER TO A HEALTHY LIFE

Virtual Medical Assistant

B1-01
2018

Week 1 Progress

EPIONE(Minimum Viable Product)

EPIONE is a Virtual Medical Assistant that is available online for clients to access anywhere.

This webapp takes in manually inputted data and

- generates a summarized report when asked by the user for a fixed time basis.
- outputs the risk factor of diabetes based on certain minimum viable parameters as inputs and gives suitable advice on moving forward.

Bill of Materials

Name of Material	Status	Cost
Kaggle	**Downloaded-Tested**	Free
Flask	**Downloaded and Ready**	Free
Jupyter Notebook	**Downloaded and Running**	Free
HTML,CSS,Bootstrap	**Downloaded and Running**	Free
Python Libraries	**Downloaded and Running**	Free
Machine Learning Algorithm	**Ready and Working**	Free

Week 2 Progress

Project Plan with Appropriate Work Breakdown

1. **Data Collection**
   a. Collecting Proper Datasets for our MVP,through open source websites.

2. **Data Visualization**
   a. Descriptive Statistics
   b. Statistical analysis
   c. Visualizing data distribution.

3. **Data Preprocessing**
   a. Data Cleaning
   b. Data Handling(missing/corrupted).

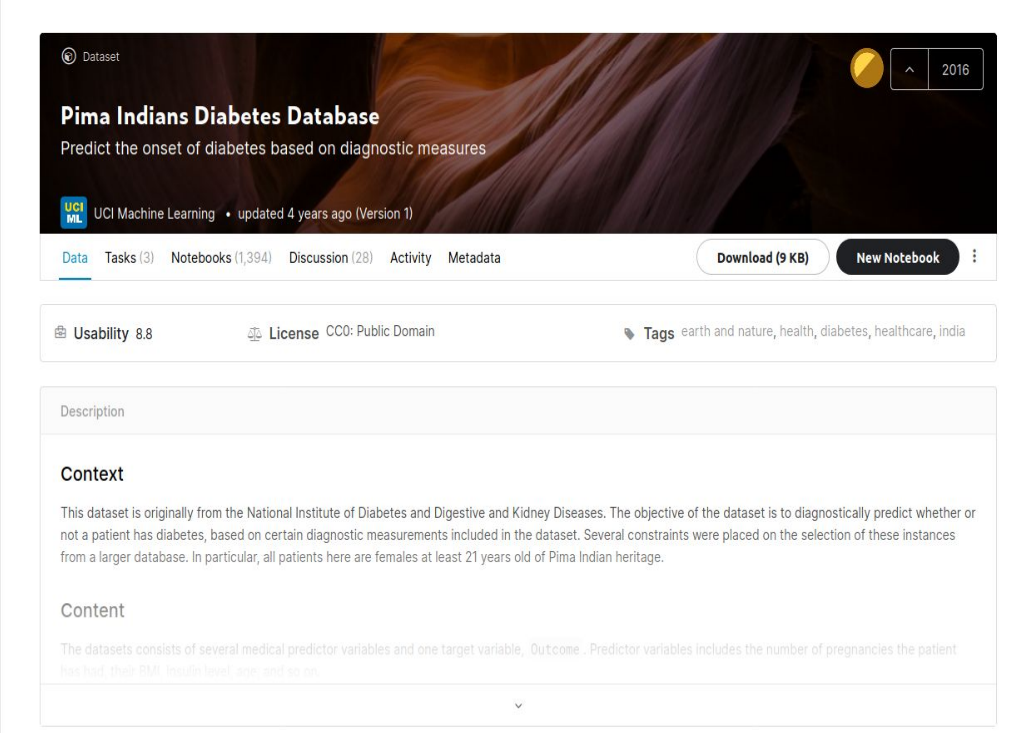
4. **Model Creation**
   a. Feature Selection
   b. Using Machine Learning Algorithms we can
      i. Risk index generation, predictive modelling.
      ii. As well as for health care ratings.
      iii. Performance testing.

5. **Web App Development**
   a. Using HTML,CSS,JavaScript to create a visually appealing web application to house our backend
   b. Using the Flask Web Framework we will build the MVP using our data models.

Week 3 Progress

Technical Schematic

Dataset Collection



Week 4 Progress

Now that the most suitable available dataset has been collected, week 4 was utilized to understand the dataset

Descriptive Statistics and Analysis

```
In [3]: dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

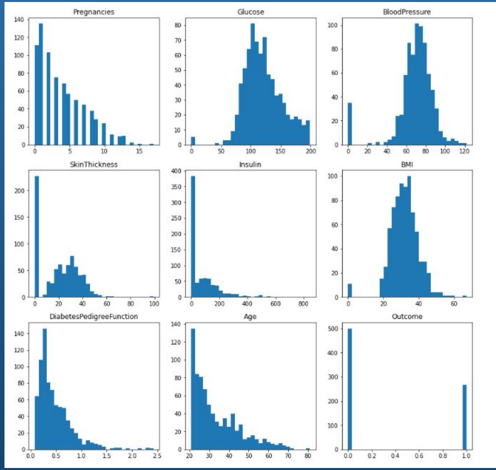
```
In [4]: dataset.isnull().sum()

Out[4]: Pregnancies                 0
        Glucose                     0
        BloodPressure               0
        SkinThickness               0
        Insulin                     0
        BMI                         0
        DiabetesPedigreeFunction    0
        Age                         0
        Outcome                     0
        dtype: int64
```

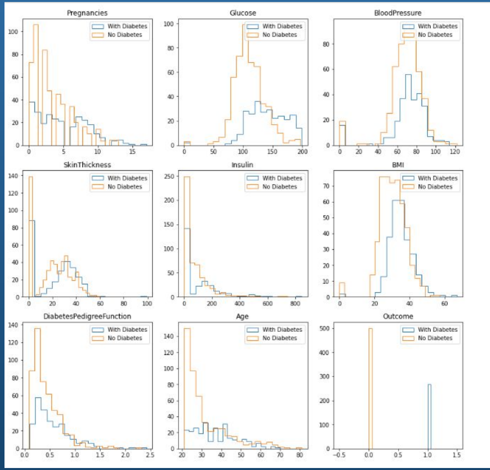
```
In [5]: dataset.describe()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Dataset Visualisation



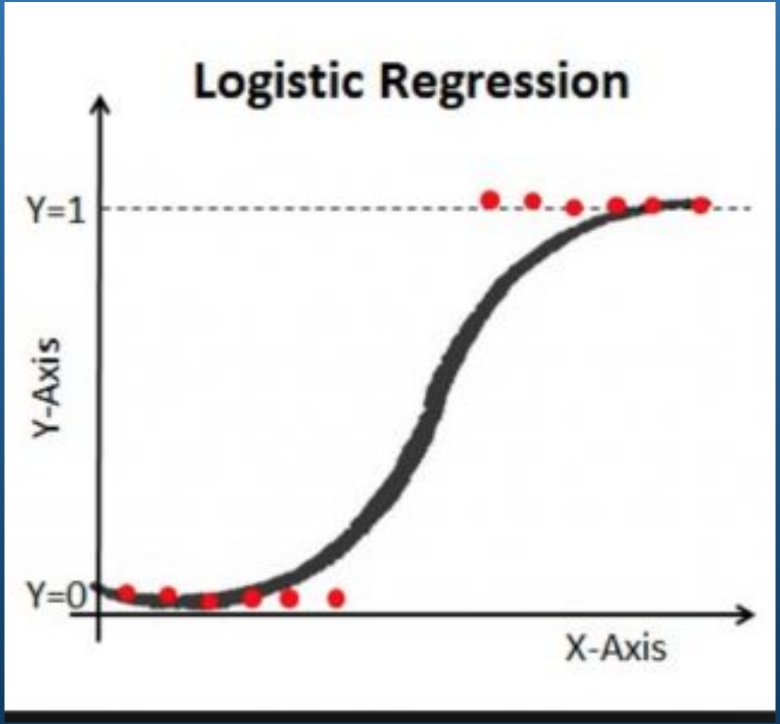


Research on Optimal Algorithm

Logistic Regression

Logistic regression is a linear classification method that learns the probability of a sample belonging to a certain class. Logistic regression tries to find the optimal decision boundary that best separates the classes.Logistic regression directly models the posterior probability of P(y|x) by learning the input to output mapping by minimising the error.Logistic regression splits feature space linearly, and typically works reasonably well even when some of the variables are correlated.

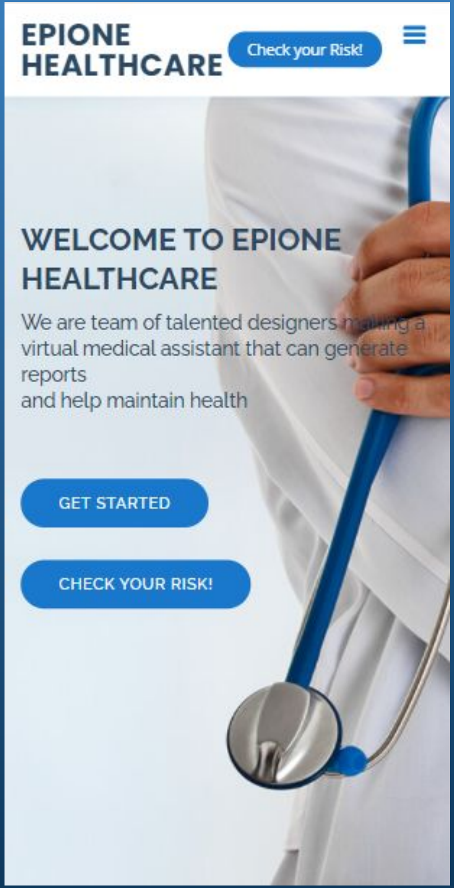
Given the correlated feature space which is following a normal distribution after cleaning up of noisy samples , and given that logistic regression is a generalized linear model which works well with correlated features, <u>LOGISTIC REGRESSION</u> was chosen as a more compatible algorithm for the MVP dataset.



Prototyping Phase

Conceptual Front End

Desktop Views



Mobile View

B1-01
2018

Data Cleaning and Handling

Dropping Duplicate Rows

In [11]:
```
dataset = dataset.drop_duplicates(keep='first')

dataset.shape
```
Out[11]: (768, 9)

Handling incomplete Data

In [12]:
```
dataset = dataset.drop(dataset[dataset['BMI']==0].index)
dataset = dataset.drop(dataset[dataset['BloodPressure']==0].index)
dataset = dataset.drop(dataset[dataset['Insulin']==0].index)
dataset = dataset.drop(dataset[dataset['Glucose']==0].index)
dataset = dataset.drop(dataset[dataset['SkinThickness']==0].index)

dataset.shape
```

Feature Selection

Feature Selection

The Pregnancies column contains information on number of pregnancies, but while there is a correlation between diabetes likelihood and number of pregnancies, our data-set does not actually distinguish between males and females. If, for instance, a male is more likely to have diabetes based on other indicators, we don't want the prediction for him having diabetes being marked down just because his number of Pregnancies is 0. To avoid the problem of the predictive model being confounded by this, we will get rid of the Pregnancies column.

There is also the fact that we have very limited understanding on how the DiabetesPedigreeFunction column's values are determined. This column measures an individual's hereditary predisposition to diabetes. Realistically, it is not going to be possible for an individual to input this measurement into our predictive model. We'll thus drop this column as well

```
features = cols.copy()
features.remove('Outcome')
features.remove('DiabetesPedigreeFunction')
features.remove('Pregnancies')


print(features)
```
```
['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'Age']
```

^ Final Features that were selected

Predictive Modelling - Logistic Regression

```
warnings.filterwarnings("ignore")

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.model_selection import KFold
```

```
# Instantiate the model
weight = {
    1:1.05,
    0:1
}
log = LogisticRegression(class_weight = weight)

kf = KFold(n_splits=6)
score = cross_val_score(log, dataset[features], dataset['Outcome'], cv=kf, scoring='accuracy')
print(score)
print("The mean accuracy is:",score.mean())
```

```
[0.81818182 0.71212121 0.72307692 0.83076923 0.87692308 0.78461538]
The mean accuracy is: 0.790947940947941
```

Risk Index Generation

```
def diabetes_risk_prediction(glucose, bp, skinthickness, insulin, bmi, age):
    indicator_list = [glucose, bp, skinthickness, insulin, bmi, age]
    predictions = log_model.predict_proba(np.array(indicator_list).reshape(1, -1))
    risk = predictions[0,1]
    print("-"*len("Health Indicator Analysis"))
    print("Health Indicator Analysis")
    print("-"*len("Health Indicator Analysis"))
    if risk < 0.3:
        print("You are probably in good health, keep it up.")
    elif risk > 0.9:
        print("Go to a hospital. Odds are high you have diabetes.")

    elif risk > 0.7:
        print("See a doctor as soon as you can . You might be on the way to developing diabetes if you don't change your lifestyle.")



    return print("Your Diabetes Risk Index is {:.2f}/50.".format(risk*0.5*100))

diabetes_risk_prediction(90,80,30,120,60,50)
```

```
-------------------------
Health Indicator Analysis
-------------------------
Your Diabetes Risk Index is 32.81/50.
```

We calculate the Posterior probability to evaluate the risk through a scaled metric.

ML Model - Performance testing and Accuracy

Re-instantiate the logistics regression and model and re-fit the data.

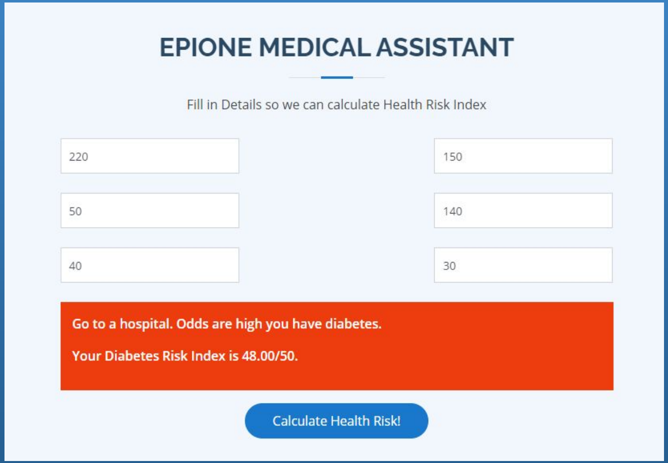
```
def diabetes_risk_prediction2(glucose, bp, skinthickness, insulin, bmi, age):
    weight = {
    1:1.05,
    0:1}

    log_model2 = LogisticRegression(class_weight = weight)
    log_model2.fit(dataset[features], dataset['Outcome'])

    indicator_list = [glucose, bp, skinthickness, insulin, bmi, age]
    predictions = log_model2.predict_proba(np.array(indicator_list).reshape(1, -1))
    risk = predictions[0,1]
    return risk

diabetes_risk_prediction2(100,80,30,120,30,45) #Test array.
```

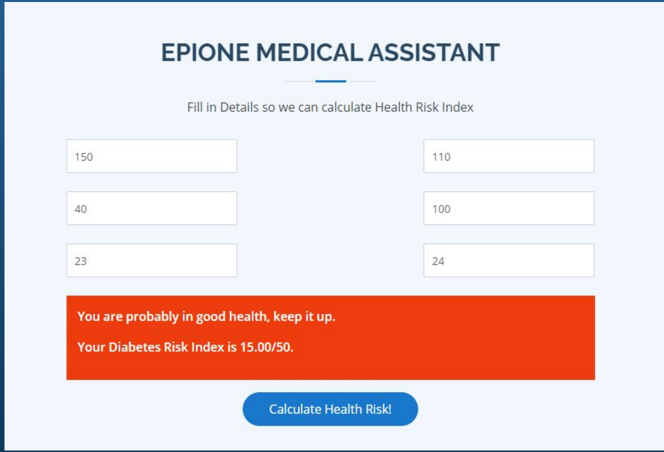
```
0.22443666622161737
```

```
kf = KFold(n_splits=6)
score = cross_val_score(log, dataset[features], dataset['Outcome'], cv=kf, scoring='accuracy')
print(score)
print("The mean accuracy is:",score.mean())
```

```
[0.81818182 0.71212121 0.72307692 0.83076923 0.87692308 0.78461538]
The mean accuracy is: 0.790947940947941
```

Using KFold cross validation with slightly more weight placed on a positive check for Diabetes, with 6 folds, we get an accuracy of around 79%.





DEMONSTRATION

Desktop Views

Mobile View

Design Pipeline

- Data Collection and Preprocessing
(Status: Completed) --20/01/2021
- Data Analysis and Visualization
(Status: Completed) -- 27/01/2021
- Data Trimming and Cleaning
(Status: Completed)--03/02/2021
- Algorithm Selection
(Status: Completed)--03/02/2021
- FrontEnd Work
(Status: Completed)--03/03/2021
- Data Model Training and Testing
(Status: Completed)--17/02/2021
- Performance Testing and Bug Identifying
(Status: Completed)--24/02/2021
- Backend Implementation and UI/UX Design
(Status: Completed)--10/03/2021
- Front to Back Integration
(Status: Completed)--17/03/2021
- Testing, Optimizing and Deployment
(Status: Completed)--24/03/2021

Click Here For
Detailed MVP Report