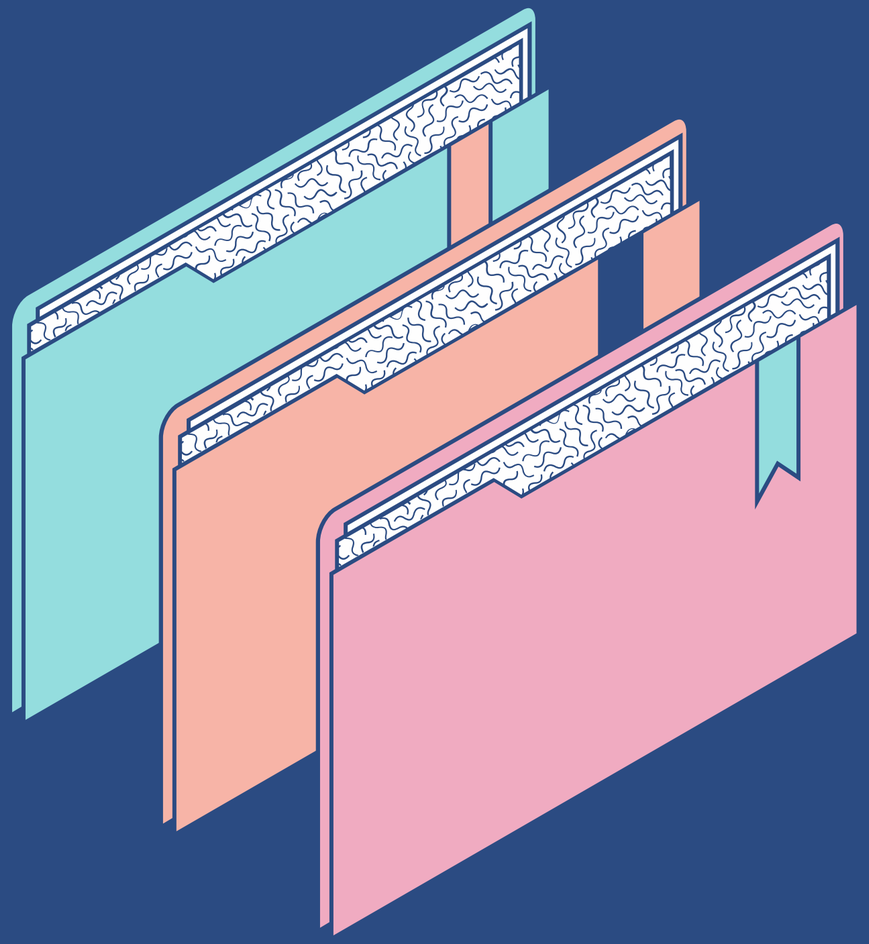
PRESENTING

Diabetes and Heart Disease Prediction Using Ensemble Machine Learning Techniques

Team members :

- 2602057690 - Vincentius Axelle Tanoto
- 2602067634 - Kent Christopher Hansel
- 2602170934 - Christopher Jovison



Theme

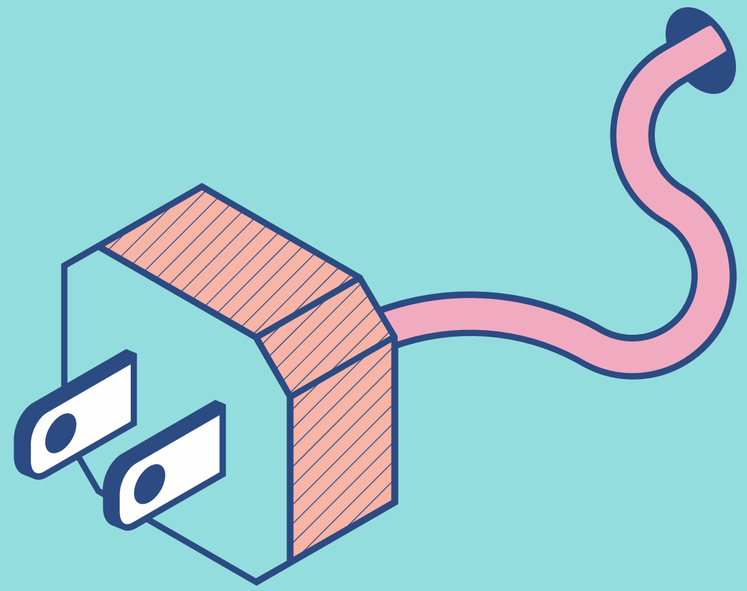
GOOD HEALTH AND WELL-BEING

Good health and wellbeing is a broad concept that encompasses both physical and mental well-being. It's about feeling good in your own body and mind, and having the energy to live a full life.

Problem definition ?

Diabetes is among the most prevalent chronic disease while heart attack causes the most deaths globally. Both diabetes and heart diseases are potentially dangerous to every individual yet difficult to diagnose at its early stages.

This program aims to tackle the problem of early detection of both diabetes and heart diseases, with an ensemble approach to machine learning.



Methodology

1 DATA REQUIERMENT

Multi-label dataset that encompasses both heart attack and diabetes, while having sufficient features

2 PREPROCESSING

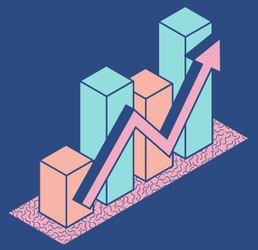
Cleaning, Correlation-based feature selection. splitting, addressing imbalance dataset using smote.

3 MODELLING

Bagging then ensemble through voting,

4 PERFORMANCE EVALUATION

Accuracy, Precision, Recall, f1-score, support



5 DEPLOYMENT

Through streamlit

1 Data Requirement

DIABETES AND HEART HEALTH INDICATORS DATASET

Dataset Description :
A multi-label dataset gathered from survey responses to the CDC's BRFSS2015 of diabetes health indicators.

ROWS	70,692
Feature Variables	20
Target Variables	2
Target Variable Type	Float (Binary)

Diabetes_binary	Hig	Hig	Ch	BMI	Sm	Str	HeartDiseaseorAttack	Ph	Frui	Veg	Hvy	An	NoD	Ge	Me	Phy	Diff	Se	Ag	Edu	Inc
0	1	0	1	26	0	0	0	1	0	1	0	1	0	3	5	30	0	1	4	6	8
0	1	1	1	26	1	1	0	0	1	0	0	1	0	3	0	0	0	1	12	6	8
0	0	0	1	26	0	0	0	1	1	1	0	1	0	1	0	10	0	1	13	6	8
0	1	1	1	28	1	0	0	1	1	1	0	1	0	3	0	3	0	1	11	6	8
0	0	0	1	29	1	0	0	1	1	1	0	1	0	2	0	0	0	0	8	5	8
0	0	0	1	18	0	0	0	1	1	1	0	0	0	2	7	0	0	0	1	4	7
0	0	1	1	26	1	0	0	1	1	1	1	1	0	1	0	0	0	1	13	5	6
0	0	0	1	31	1	0	0	0	1	1	0	1	0	4	0	0	0	1	6	4	3
0	0	0	1	32	0	0	0	1	1	1	0	1	0	3	0	0	0	0	3	6	8
0	0	0	1	27	1	0	0	0	1	1	0	1	0	3	0	6	0	1	6	4	4
0	1	1	1	24	1	0	1	1	1	1	0	1	0	3	0	4	0	0	12	4	6
0	0	0	1	21	0	0	0	1	1	1	0	1	0	1	0	0	0	1	4	6	8
0	1	1	1	27	0	0	0	1	1	1	0	1	0	2	0	0	0	1	7	6	8
0	1	0	1	58	0	0	0	0	1	1	0	1	0	3	3	3	0	1	10	4	6
0	0	1	1	29	1	0	0	1	1	0	0	1	0	1	0	0	1	0	10	5	1
0	0	0	1	18	1	0	0	1	1	0	0	1	0	3	0	0	0	0	10	4	6
0	0	0	1	30	0	0	0	1	0	1	0	1	0	2	0	0	0	0	9	5	7
0	0	0	1	30	1	0	0	1	1	1	0	1	0	1	0	0	0	1	10	6	7
0	0	0	1	20	0	0	0	1	1	1	0	1	0	2	0	0	0	0	8	6	8
0	1	0	1	26	0	0	0	0	1	1	0	0	1	3	0	15	0	1	7	5	5
0	0	0	1	22	0	0	0	1	1	1	0	1	0	1	0	0	0	0	6	6	8
0	1	0	1	29	1	0	0	1	1	1	0	1	0	5	0	30	0	1	10	5	8
0	0	0	1	22	0	0	0	1	0	1	1	1	0	2	0	0	0	0	8	6	8
0	0	0	1	30	0	0	0	1	1	1	0	1	0	1	0	0	0	0	9	4	6

Fig 1.0 Dataset overview

Dataset Features

DIABETES AND HEART HEALTH INDICATORS DATASET

FEATURES	DATA TYPES	DESCRIPTION
HighBP	Float	Have high blood pressure (binary)
HighChol	Float	Have high cholesterol (binary)
CholCheck	Float	Cholesterol check in 5 years (binary)
BMI	Float	Body Mass Index
GenHelth	Float	Health Scale (1-5)

FEATURES	DATA TYPES	DESCRIPTION
Smoker	Float	Is a smoker
Stroke	Float	Had a stroke (binary)
PhysActivity	Float	Physical active (binary)
Fruits	Float	Consume fruits per day (binary)
MentHlth	Float	Days of poor mental health (1.0-30.0)

Dataset Features

DIABETES AND HEART HEALTH INDICATORS DATASET

FEATURES	DATA TYPES	DESCRIPTION
Veggies	Float	Consume veggies per day (binary)
HvyAlcoholConsume	Float	Drinks per week (binary)
AnyHealthcare	Float	Have health care (binary)
NoDocbcCost	Float	Need a doctor, no money (binary)
DiffWalk	Float	Have difficulty walking (binary)

FEATURES	DATA TYPES	DESCRIPTION
Sex	Float	Female or male (binary)
Age	Float	13-level categories of age
Education	Float	Education level (1.0-6.0)
Income	Float	Income scale (1.0-8.0)

Targeted Output:

- Diabetes_binary (binary)
- HeartAttackorDisease (binary)

Preprocessing

HOW RAW DATA IS PROCESSED INTO A USABLE FORMAT

2

Google Colab
google.com

1. Data Cleaning

This process consists of checking for missing values and outliers. (None found in our dataset)

---

2. Correlation-based feature selection

Next a correlation matrix will be created and features will be simplified

---

3. Splitting datasets

Dataset is then split into two, one for training diabetes detection, one for training heart disease detection.

4. Addresing Imbalance

Individual dataset imbalance is addressed using smote.

② Preprocessing

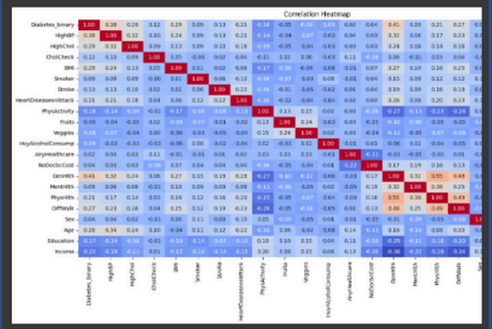
1 Data Cleaning

```
df.isna().sum()
```

```
Diabetes_binary         0
HighBP                  0
HighChol                0
CholCheck               0
BMI                     0
Smoker                  0
Stroke                  0
HeartDiseaseorAttack    0
PhysActivity            0
Fruits                  0
Veggies                 0
HvyAlcoholConsump       0
AnyHealthcare           0
NoDocbcCost             0
GenHlth                 0
MentHlth                0
PhysHlth                0
DiffWalk                0
Sex                     0
Age                     0
Education               0
Income                  0
dtype: int64
```

2 Correlation-based feature selection

```
plt.figure(figsize=(20, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()
```




```
df1 = df.drop(columns=['Income','Education'])
```

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70692 entries, 0 to 70691
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype
```

3 Splitting datasets

```
X = df1.drop(columns=['Diabetes_binary', 'HeartDiseaseorAttack'])
Y1 = df1['Diabetes_binary']
Y2 = df1['HeartDiseaseorAttack']
```

```
print(df1['HeartDiseaseorAttack'].value_counts())
print(df1['Diabetes_binary'].value_counts())
```

```
HeartDiseaseorAttack
0.0    60243
1.0    10449
Name: count, dtype: int64
Diabetes_binary
0.0    35346
1.0    35346
Name: count, dtype: int64
```

4 Addresing Imbalance

```
## SMOTE Diabetes
from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=42)
X_res, y_res = smote.fit_resample(X, Y1)

# X for diabetes
diabetes = pd.concat([pd.DataFrame(X_res), pd.DataFrame(y_res, columns=['Diabetes_binary'])], axis=1)

print(diabetes.info())
```

```
## SMOTE HeartDiseaseorAttack
from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=42)
X_res, y_res = smote.fit_resample(X, Y2)

# X for heart
heart = pd.concat([pd.DataFrame(X_res), pd.DataFrame(y_res, columns=['HeartDiseaseorAttack'])], axis=1)

print(heart['HeartDiseaseorAttack'].value_counts())
```

③ Modelling

Diabetes model

Models Used :

- Logistic Regression
- Random Forest Classifier
- K Neighbors Classifier
- Support Vector Machine
- XGBoost
- Gaussian Naive Bayes

Techniques used :

- Bagging
- Voting

Hierarchy

VOTING CLASSIFIER

```
ensemble_diabetes = VotingClassifier(estimators=[('bagging', bagging_diabetes),
                                                 ('lr', LR_diabetes),
                                                 ('rfc', RFC_diabetes),
                                                 ('knc', KNC_diabetes),
                                                 ('xgb', XGB_diabetes),
                                                 ('svc', SVC_diabetes),('nb', NB_diabetes)],
                                                  voting='hard')
ensemble_diabetes.fit(X_diabetes_train, Y_diabetes_train)
ensemble_diabetes_pred = ensemble_diabetes.predict(X_diabetes_test)
print(classification_report(Y_diabetes_test, ensemble_diabetes_pred))
```

BAGGING

Logistics regression x100

```
bagging_diabetes = BaggingClassifier(base_estimator=LR_diabetes, n_estimators=100, random_state=42)
bagging_diabetes.fit(X_diabetes_train, Y_diabetes_train)
bagging_diabetes_pred = bagging_diabetes.predict(X_diabetes_test)
print(classification_report(Y_diabetes_test, bagging_diabetes_pred))
```

BASE MODELS

```
LR_diabetes = LogisticRegression()
RFC_diabetes = RandomForestClassifier()
KNC_diabetes = KNeighborsClassifier()
SVC_diabetes = SVC()
XGB_diabetes = XGBClassifier()
NB_diabetes = GaussianNB()
```

③ Modelling

HeartDisease model

Hierarchy

Models Used :

- Logistic Regression
- Random Forest Classifier
- K Neighbors Classifier
- Support Vector Machine
- XGBoost
- Gaussian Naive Bayes

Techniques used :

- Bagging
- Voting

VOTING CLASSIFIER

```
ensemble_heart = VotingClassifier(estimators=[('bagging', bagging_heart),
                                              ('lr', LR_heart),
                                              ('rfc', RFC_heart),
                                              ('knc', KNC_heart),
                                              ('xgb', XGB_heart),
                                              ('svc', SVC_heart),
                                              ('nb', NB_heart)], voting='hard')
ensemble_heart.fit(X_heart_train, Y_heart_train)
ensemble_heart_pred = ensemble_heart.predict(X_heart_test)
print(classification_report(Y_heart_test, ensemble_heart_pred))
```

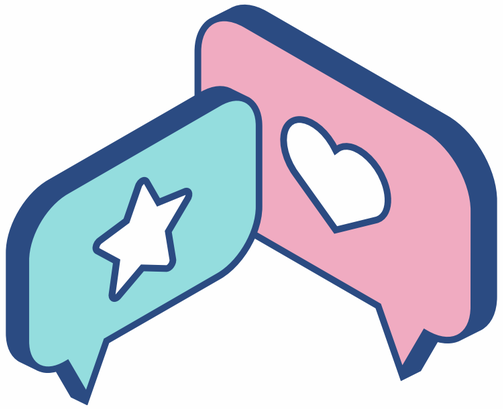
BAGGING

Logistics regression x100

```
bagging_heart = BaggingClassifier(base_estimator=LR_heart, n_estimators=100, random_state=42)
bagging_heart.fit(X_heart_train, Y_heart_train)
bagging_heart_pred = bagging_heart.predict(X_heart_test)
print(classification_report(Y_heart_test, bagging_heart_pred))
```

BASE MODELS

```
LR_heart = LogisticRegression()
RFC_heart = RandomForestClassifier()
KNC_heart = KNeighborsClassifier()
SVC_heart = SVC()
XGB_heart = XGBClassifier()
NB_heart = GaussianNB()
```



4 Performance Evaluation

Using classification_report

Bagging Diabetes

	precision	recall	f1-score	support
0.0	0.76	0.72	0.74	10601
1.0	0.73	0.77	0.75	10607
accuracy			0.75	21208
macro avg	0.75	0.75	0.75	21208
weighted avg	0.75	0.75	0.75	21208

Bagging Heart Disease

	precision	recall	f1-score	support
0.0	0.77	0.71	0.74	18180
1.0	0.73	0.79	0.76	17966
accuracy			0.75	36146
macro avg	0.75	0.75	0.75	36146
weighted avg	0.75	0.75	0.75	36146

Bagging + Voting Diabetes

	precision	recall	f1-score	support
0.0	0.76	0.73	0.74	10601
1.0	0.74	0.77	0.75	10607
accuracy			0.75	21208
macro avg	0.75	0.75	0.75	21208
weighted avg	0.75	0.75	0.75	21208

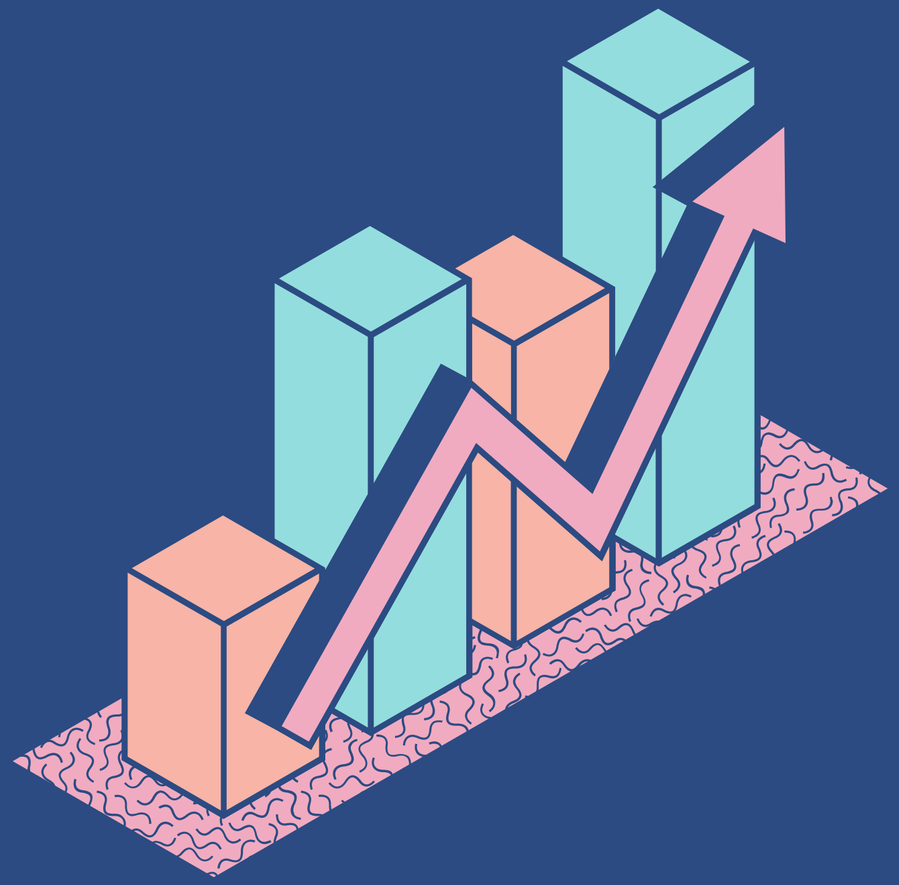
Bagging + Voting Heart Disease

	precision	recall	f1-score	support
0.0	0.85	0.73	0.79	18180
1.0	0.76	0.87	0.81	17966
accuracy			0.80	36146
macro avg	0.81	0.80	0.80	36146
weighted avg	0.81	0.80	0.80	36146

Deployment

Through Streamlit

5



1. Import library

```
import streamlit as st
import pickle
import numpy as np
import pandas as pd
from joblib import load
```

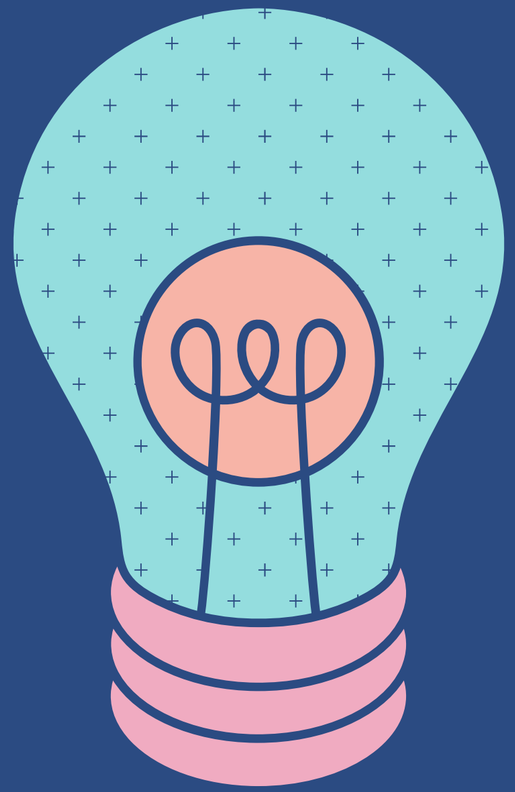
2. Loading model

```
diabetes_model = load('diabetes_model69.pkl')
heart_model = load('heart_model69.pkl')
```

3. Creating Prediction Function

```
def predict_diabetes(features):
    prediction = diabetes_model.predict(features)
    return prediction

def predict_heart_disease(features):
    prediction = heart_model.predict(features)
    return prediction
```



4. Display & Mapping

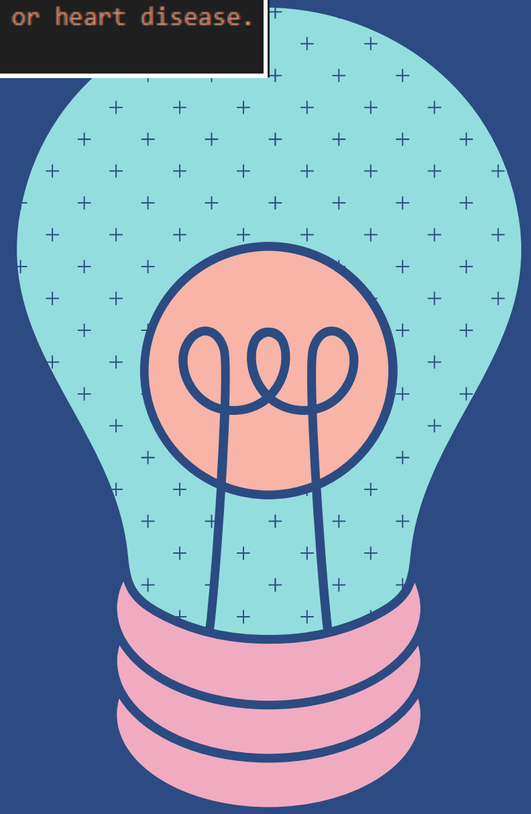
Display Title

```
def main():
    st.title('Diabetes and Heart Disease Prediction')
    st.write("""
## How to Use:
1. Select your age range, gender, and answer various health-related questions.
2. Adjust sliders and radio buttons based on your health information.
3. Once you've filled in all the details, click the buttons to predict diabetes or heart disease.
""")
```

Display & Mapping Slider

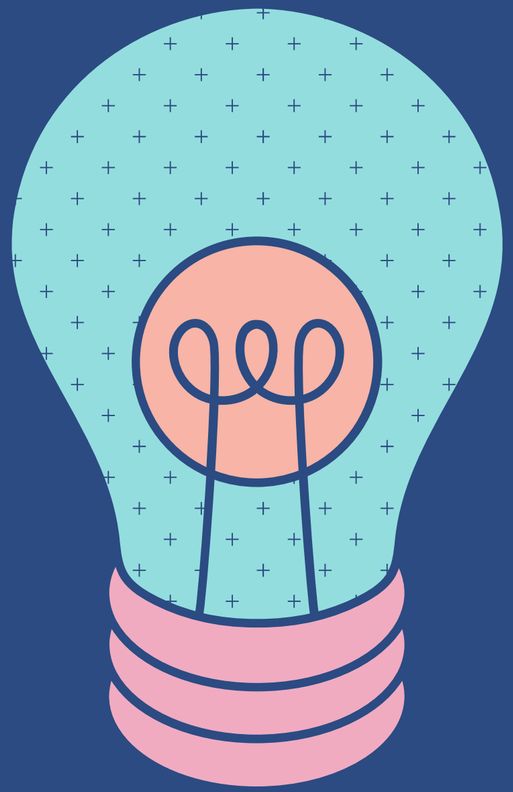
```
age_mapping = {
    '18-24': 1,
    '25-29': 2,
    '30-34': 3,
    '35-39': 4,
    '40-44': 5,
    '45-49': 6,
    '50-54': 7,
    '55-59': 8,
    '60-64': 9,
    '65-69': 10,
    '70-74': 11,
    '75-79': 12,
    '80 or older': 13
}

st.header('Age')
Age = st.radio('Select your age range:', list(age_mapping.keys()))
```



Display & Mapping Button

```
st.header("Difficulty Walking or Climbing Stairs")
DiffWalk = st.radio('Do you have serious difficulty walking or climbing stairs?', ['No', 'Yes'])
diff_walk_mapping = {'No': 0, 'Yes': 1}
```



5. Storing input into variable

```
    features = np.array([
        age_mapping[Age],
        sex_mapping[Sex],
        high_bp_mapping[HighBP],
        high_chol_mapping[HighChol],
        chol_check_mapping[CholCheck],
        BMI,
        smoker_mapping[Smoker],
        stroke_mapping[Stroke],
        phys_activity_mapping[PhysActivity],
        fruits_mapping[Fruits],
        veggies_mapping[Veggies],
        hvy_alcohol_mapping[HvyAlcoholConsump],
        healthcare_mapping[AnyHealthcare],
        no_doc_cost_mapping[NoDocbcCost],
        gen_hlth_mapping[GenHlth],
        MentHlth,
        PhysHlth,
        diff_walk_mapping[DiffWalk]
    ]).reshape(1, -1)
```

6. Button Creation

```python
if st.button('Check My Health Conditions'):
    diabetes_prediction = predict_diabetes(features)
    heart_prediction = predict_heart_disease(features)

    diabetes_result = 'Diabetes Detected' if diabetes_prediction[0] == 1 else 'No Diabetes'
    heart_result = 'Heart Disease Detected' if heart_prediction[0] == 1 else 'No Heart Disease'

    st.success(f'Diabetes Prediction: {diabetes_result}')
    st.success(f'Heart Disease Prediction: {heart_result}')

    if diabetes_prediction[0] == 1:
        st.write("It seems that diabetes is detected. Here are some suggestions:")
        st.write("- Consult with your healthcare provider for further evaluation and management.")
        st.write("- Follow a balanced diet and maintain a healthy weight.")
        st.write("- Engage in regular physical activity.")
        st.write("- Monitor your blood sugar levels regularly.")

    if heart_prediction[0] == 1:
        st.write("It seems that heart disease is detected. Here are some suggestions:")
        st.write("- Consult with your healthcare provider for further evaluation and management.")
        st.write("- Follow a heart-healthy diet low in saturated fats, cholesterol, and sodium.")
        st.write("- Engage in regular physical activity, as recommended by your healthcare provider.")
        st.write("- Monitor your blood pressure and cholesterol levels regularly.")
```

Predict Disease

The input stored in the variable “features” is used by the model to predict disease.

Create prediction text

Based on the model prediction, either disease detected text or disease not found text is stored in variables.

Output Prediction

Both diabetes and heart disease prediction previously stored in variables are outputted.

Additional Information

Based on prediction, extra information to possibly consult for help is outputted

Demonstration

Streamlit LINK