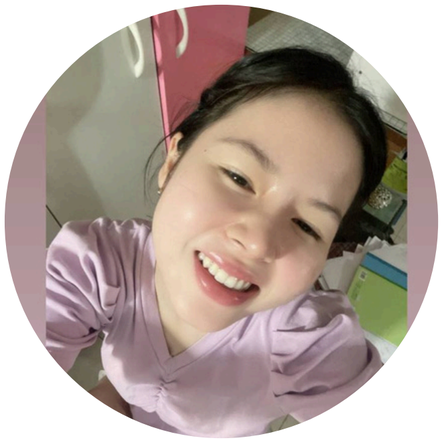
Diabetes predictor

TEAM 6

Dr. PHAUK Sokkhey

HOUN Sopanha
Marchine Learning Engineer



HANG Muykhorng
Data Analyst
Presentation specialist

HUN Sopheak
Project Manager

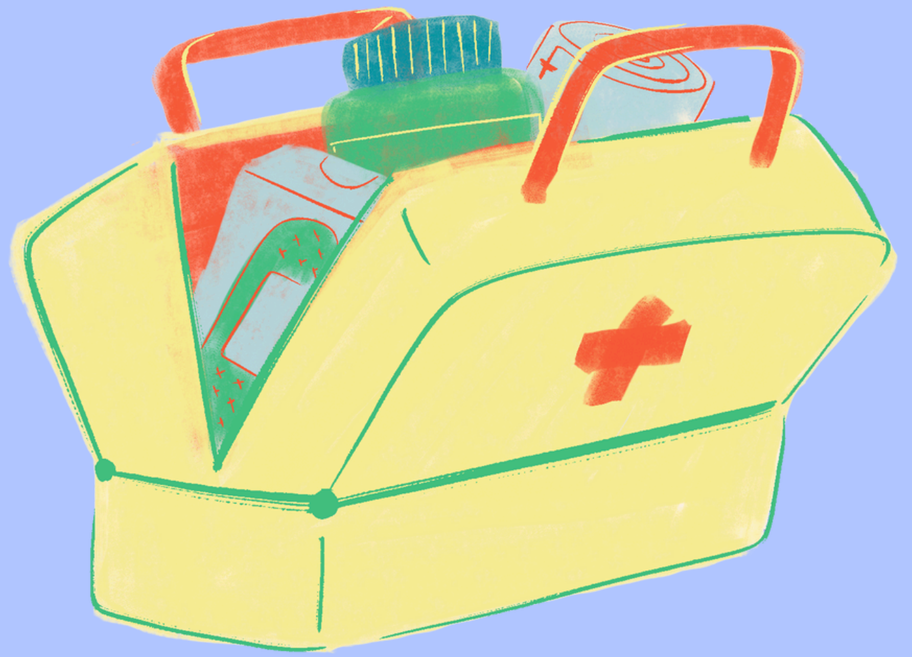
CHHOUK Phalthunin
Document Specialist

CHOEURN Brospov
Data Collector

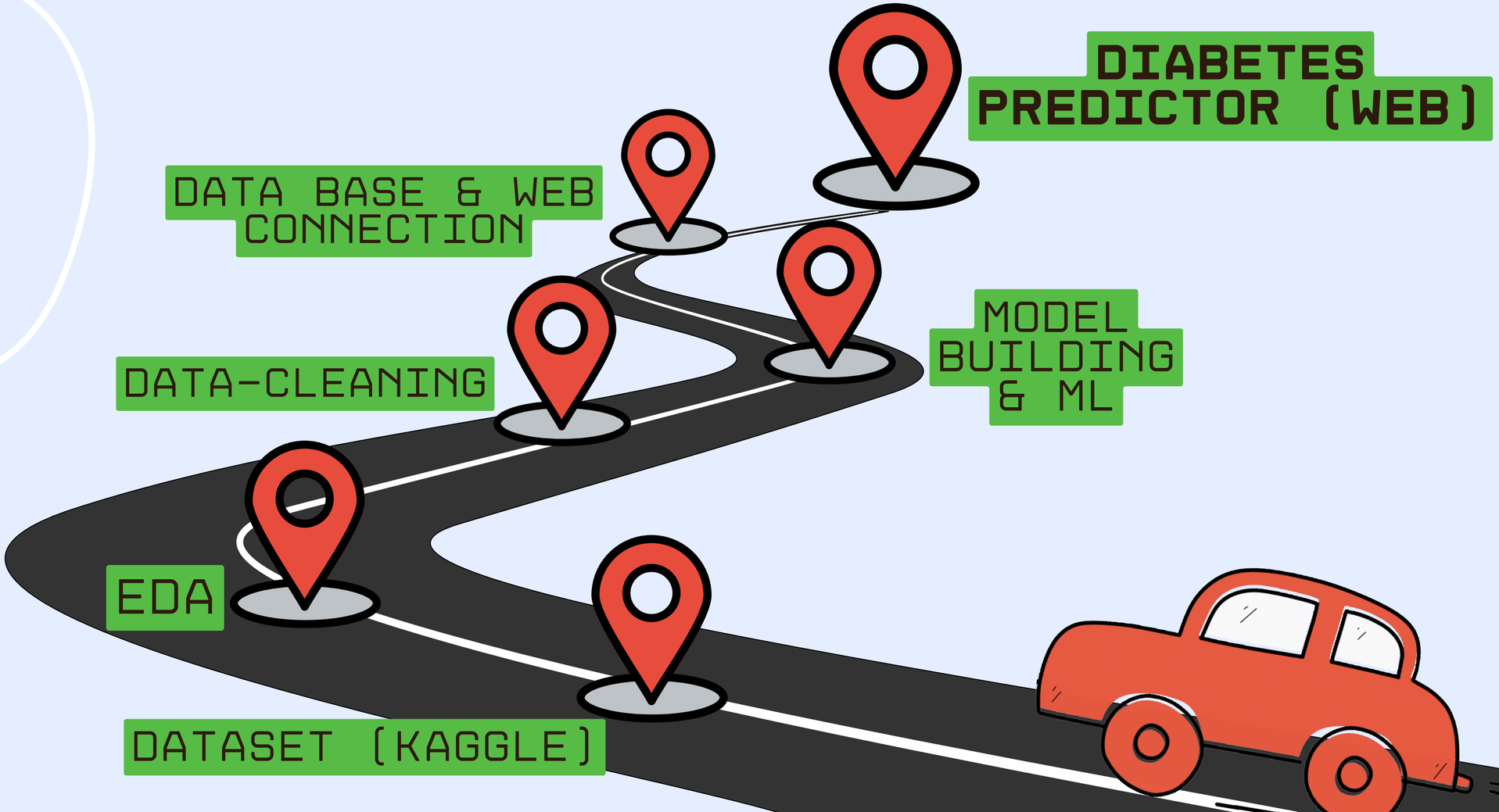
Contents

Objective

1. Promote Diabetes Awareness
2. Facilitate Early Detection
3. Leverage Technology
   -Intergrate ML, engineering into real life usages
4. provide accessible tool for hospitals in rural areas
5. Low-cost considered



Road Map

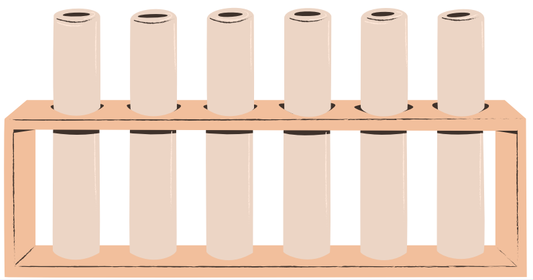

DIABETES PREDICTOR (WEB)
DATA BASE & WEB CONNECTION
MODEL BUILDING & ML
DATA-CLEANING
EDA
DATASET (KAGGLE)

EDA

A quick overvies of the dataset

```
<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
```

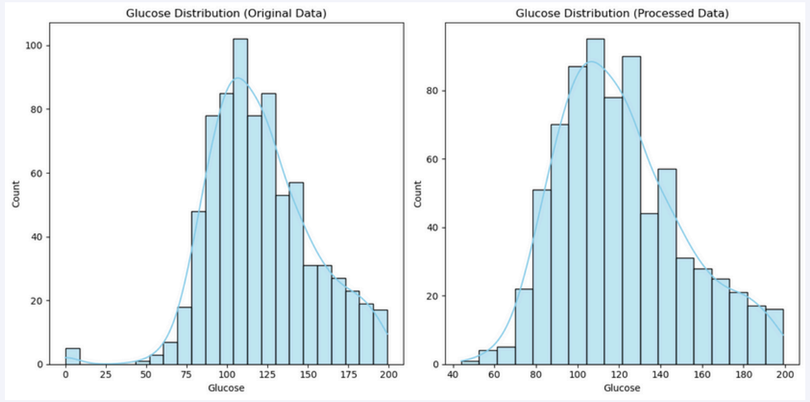
```
Summary Statistics:
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
min       0.000000    0.000000       0.000000       0.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000
75%       6.000000  140.250000      80.000000      32.000000  127.250000
max      17.000000  199.000000     122.000000      99.000000  846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000

Mode for Each Column:
 Pregnancies                 1.000
Glucose                     99.000
BloodPressure               70.000
SkinThickness                0.000
Insulin                      0.000
BMI                         32.000
DiabetesPedigreeFunction     0.254
Age                         22.000
Outcome                      0.000
Name: 0, dtype: float64
```



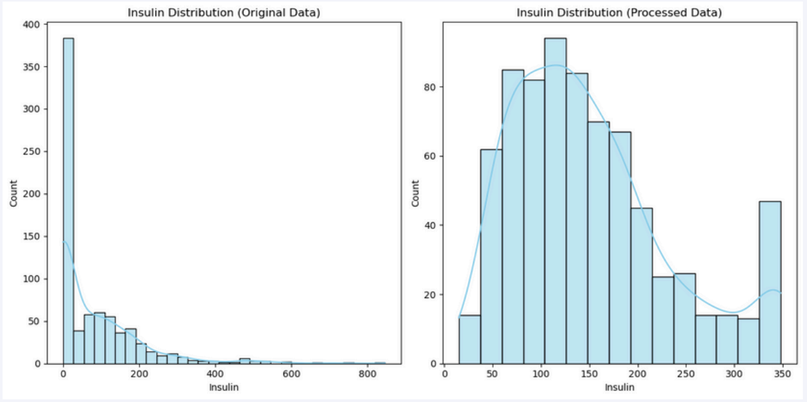
Data Cleaning

Handle the issues identified in the EDA:

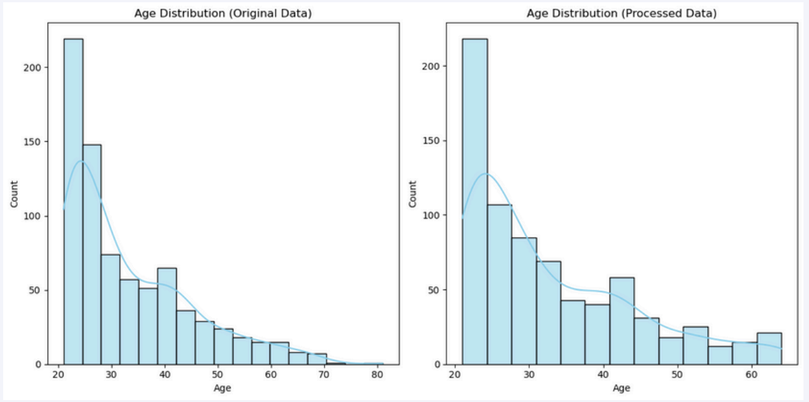
- **Remove or impute missing values**
- **Address outliers**


Glucose Distribution (Original Data)
Count
Glucose

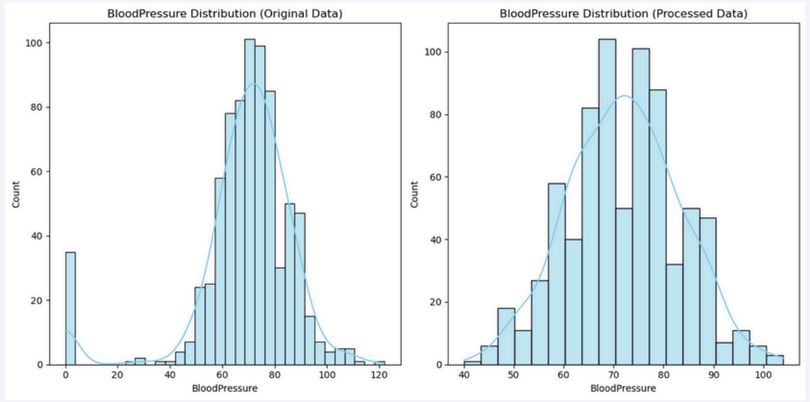
Glucose Distribution (Processed Data)
Count
Glucose


Insulin Distribution (Original Data)
Count
Insulin

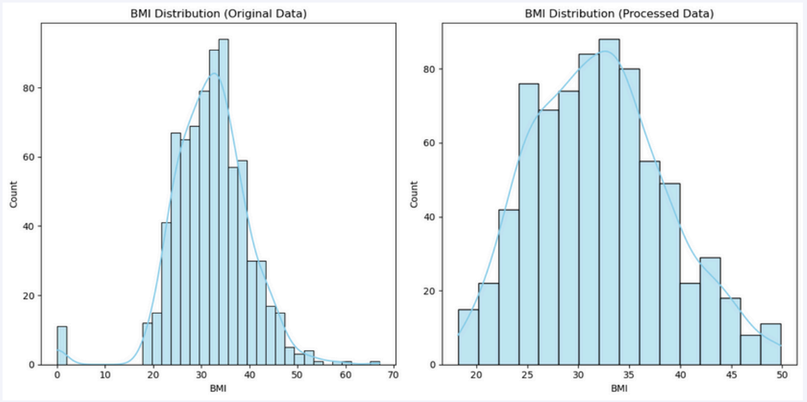
Insulin Distribution (Processed Data)
Count
Insulin


Age Distribution (Original Data)
Count
Age

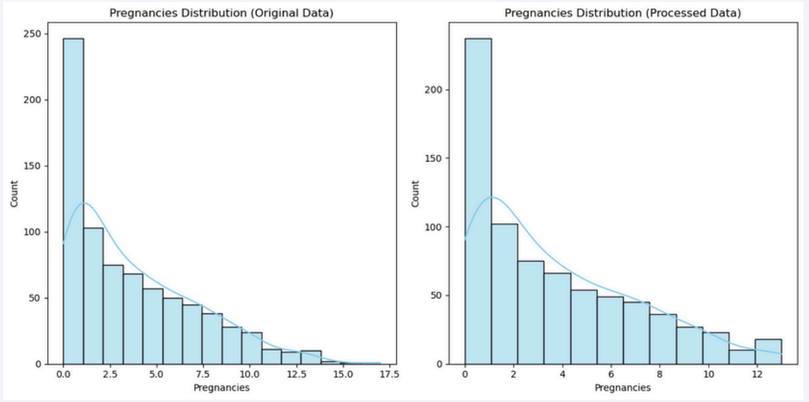
Age Distribution (Processed Data)
Count
Age


BloodPressure Distribution (Original Data)
Count
BloodPressure

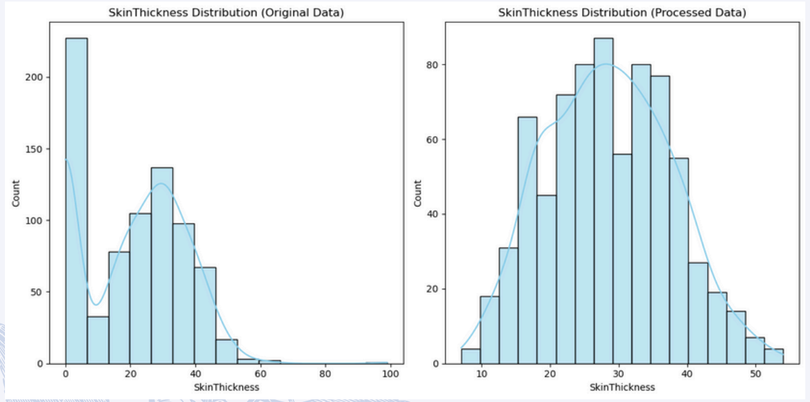
BloodPressure Distribution (Processed Data)
Count
BloodPressure


BMI Distribution (Original Data)
Count
BMI

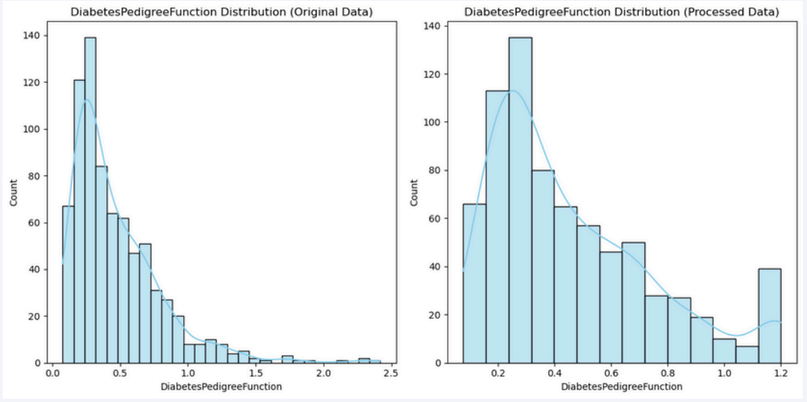
BMI Distribution (Processed Data)
Count
BMI


Pregnancies Distribution (Original Data)
Count
Pregnancies

Pregnancies Distribution (Processed Data)
Count
Pregnancies


SkinThickness Distribution (Original Data)
Count
SkinThickness

SkinThickness Distribution (Processed Data)
Count
SkinThickness

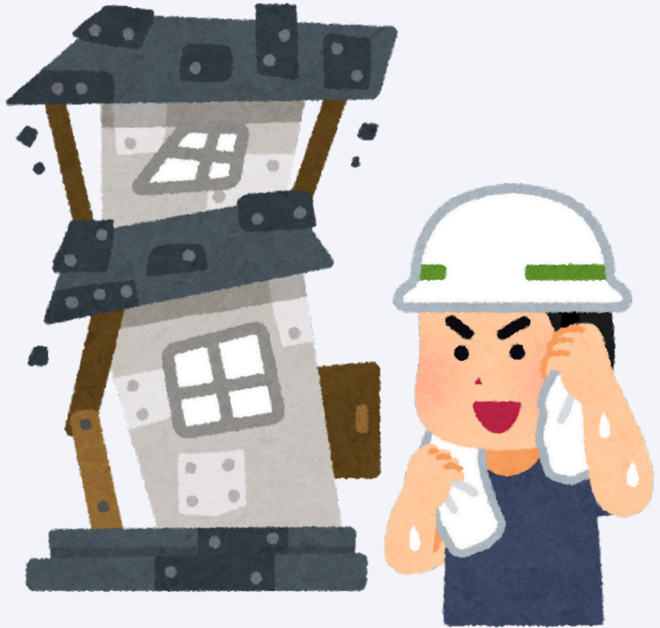

DiabetesPedigreeFunction Distribution (Original Data)
Count
DiabetesPedigreeFunction

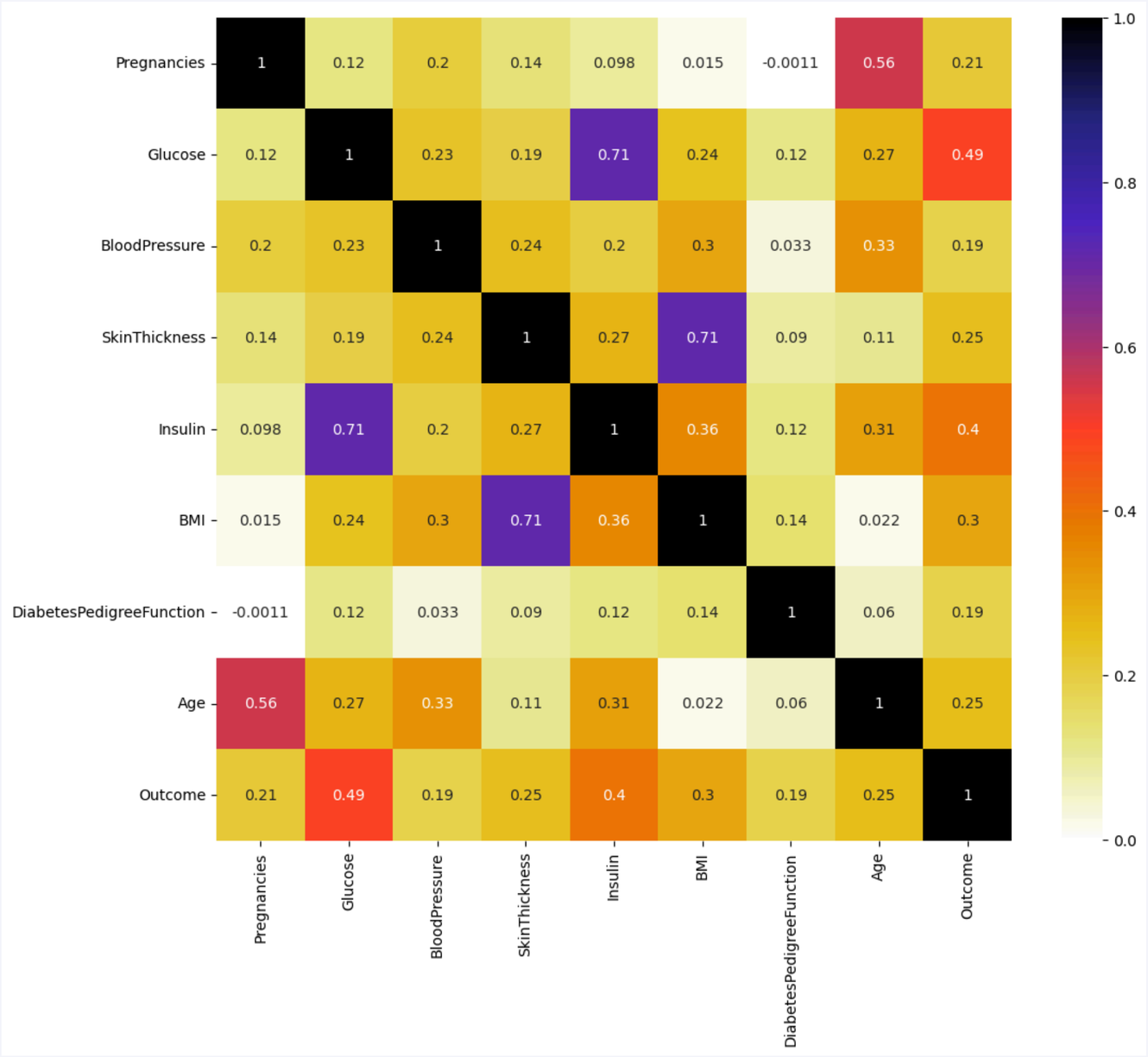
DiabetesPedigreeFunction Distribution (Processed Data)
Count
DiabetesPedigreeFunction


INSTITUT DE TECHNOLOGIE DU CAMBODGE

Modeling

- **Feature (Variables) Selection: the process of selecting a subset of the most relevant features from the dataset to improve model performance and reduce complexity.**





- **Splits the dataset into: 80% train, and 20% test**
- **Build model:**

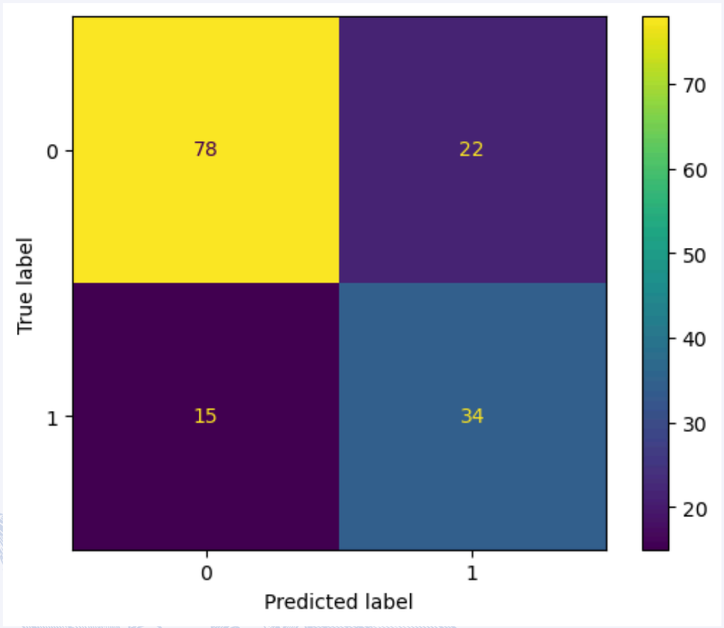
1. Logistic Regression

Logistic Regression

```
lg_model = LogisticRegression()
lg_model = lg_model.fit(x_train,y_train)
```

```
y_pre = lg_model.predict(x_test)
print(classification_report(y_test,y_pre))
```

```
              precision    recall  f1-score   support

           0       0.84      0.78      0.81       100
           1       0.61      0.69      0.65        49

    accuracy                           0.75       149
   macro avg       0.72      0.74      0.73       149
weighted avg       0.76      0.75      0.76       149
```



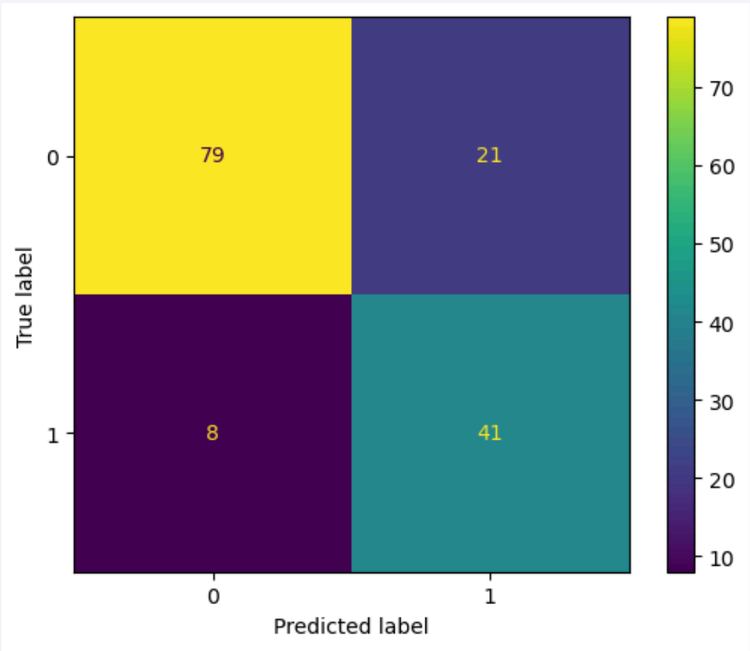
2. SVM

Supporting Vector Machine

```
svm_model = SVC()
svm_model = svm_model.fit(x_train,y_train)
```

```
y_pre = svm_model.predict(x_test)
print(classification_report(y_test,y_pre))
```

```
              precision    recall  f1-score   support

           0       0.91      0.79      0.84       100
           1       0.66      0.84      0.74        49

    accuracy                           0.81       149
   macro avg       0.78      0.81      0.79       149
weighted avg       0.83      0.81      0.81       149
```



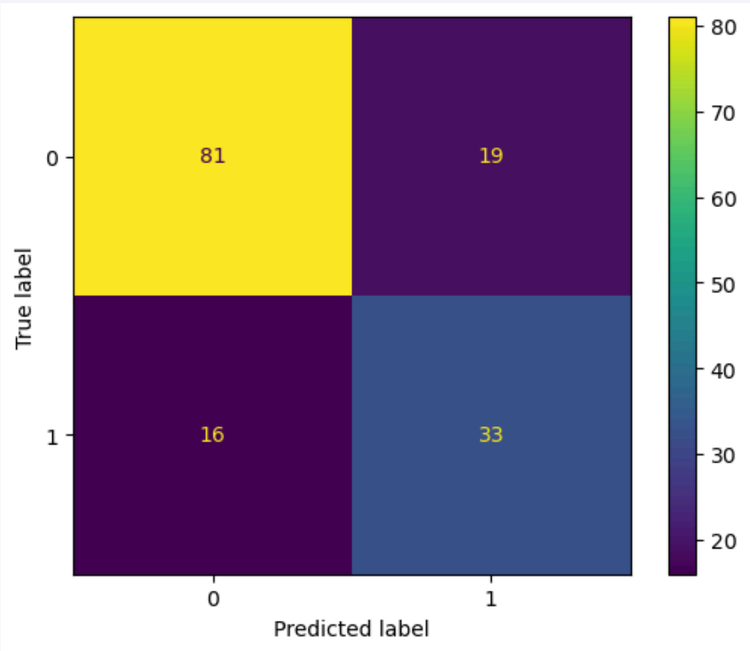
3. Gaussian Naive Bays

Gaussian Naive Bays

```
nb_model = GaussianNB()
nb_model = nb_model.fit(x_train,y_train)
```

```
y_pre = nb_model.predict(x_test)
print(classification_report(y_test,y_pre))
```

```
              precision    recall  f1-score   support

           0       0.84      0.81      0.82       100
           1       0.63      0.67      0.65        49

    accuracy                           0.77       149
   macro avg       0.73      0.74      0.74       149
weighted avg       0.77      0.77      0.77       149
```



4. Tuning for GaussianNB

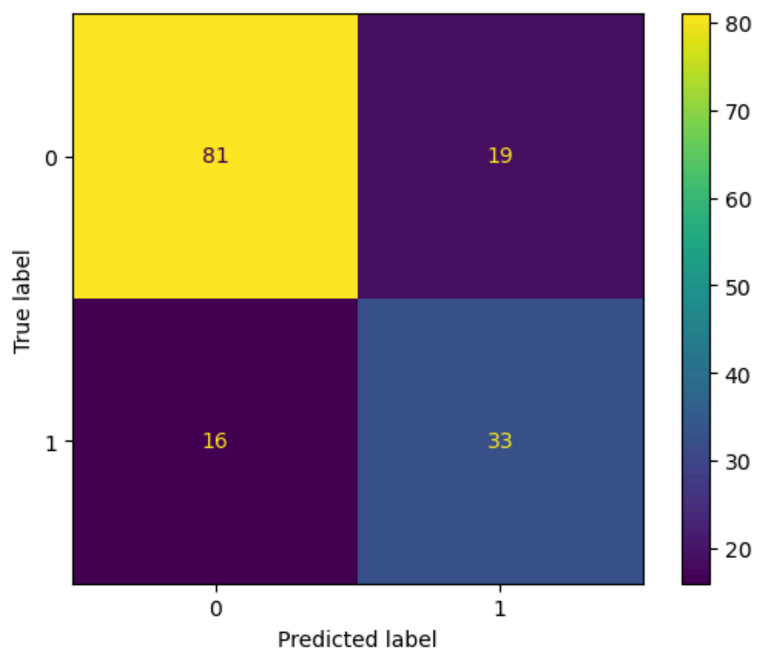
Tuning for GaussianNB

```
parameters = {
    'var_smoothing': [1e-9, 1e-8, 1e-7, 1e-6]  # The regularization term (for smoothing the variance)

}

nb_model = GaussianNB()
grid_search = GridSearchCV(estimator = nb_model,
                           param_grid = parameters,
                           scoring = 'accuracy',
                           cv = 5,
                           verbose=0)
# Fit the model
grid_search.fit(x_train, y_train)
grid_search.best_params_
```

```
{'var_smoothing': 1e-09}
```

```
y_pre = grid_search.predict(x_test)
print(classification_report(y_test,y_pre))
```

```
              precision    recall  f1-score   support

           0       0.84      0.81      0.82       100
           1       0.63      0.67      0.65        49

    accuracy                           0.77       149
   macro avg       0.73      0.74      0.74       149
weighted avg       0.77      0.77      0.77       149
```

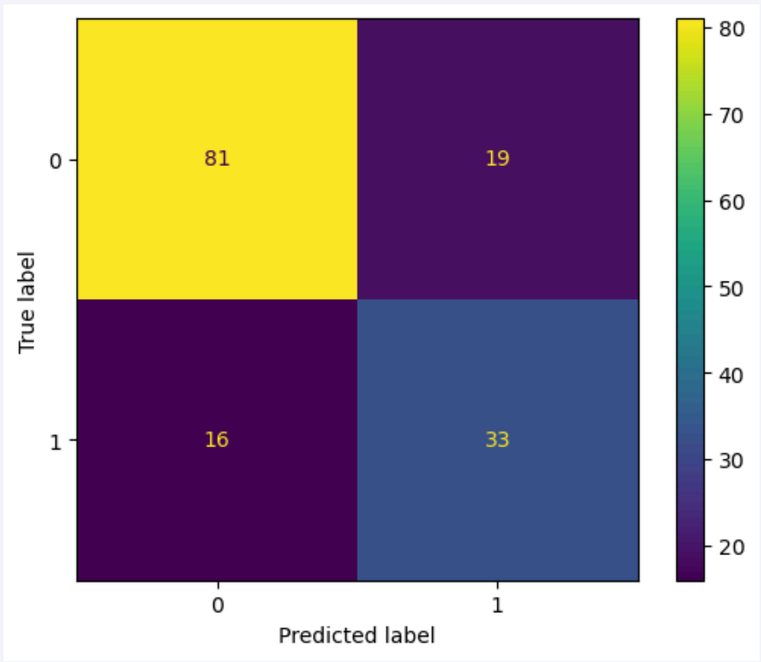


5. Random Forest

Random Forest

```
rf_model = RandomForestClassifier(criterion='gini',
                                  max_depth=8,
                                  min_samples_split=10)
rf_model.fit(x_train, y_train)
y_pre = grid_search.predict(x_test)
print(classification_report(y_test,y_pre))
```

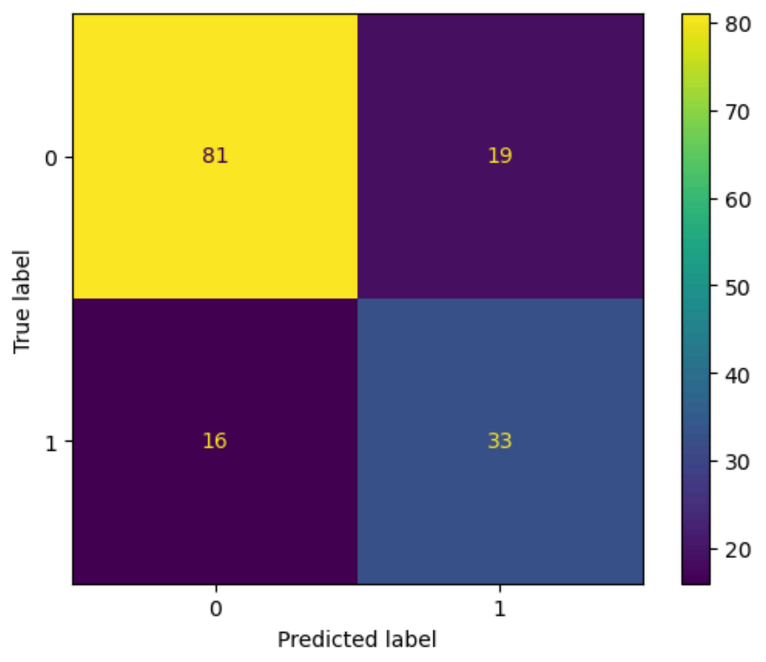
```
              precision    recall  f1-score   support

           0       0.84      0.81      0.82       100
           1       0.63      0.67      0.65        49

    accuracy                           0.77       149
   macro avg       0.73      0.74      0.74       149
weighted avg       0.77      0.77      0.77       149
```



6. Random Forest + Turning

```
y_pre = grid_search.predict(x_test)
print(classification_report(y_test,y_pre))
# print(classification_report(y_test, y_pre))
```

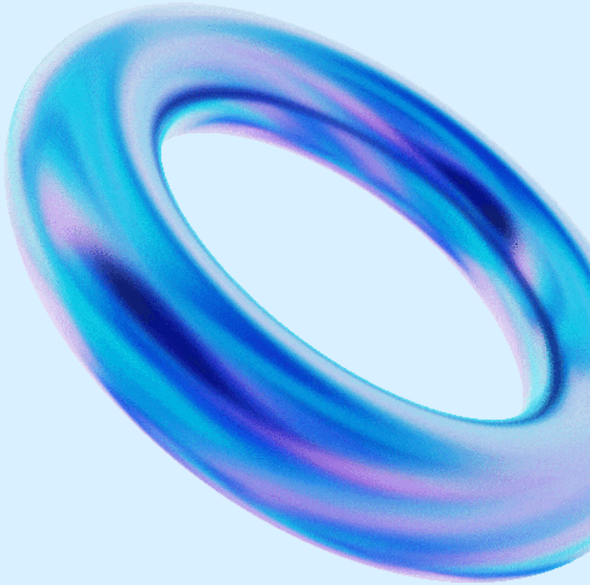
```
              precision    recall  f1-score   support

           0       0.84      0.81      0.82       100
           1       0.63      0.67      0.65        49

    accuracy                           0.77       149
   macro avg       0.73      0.74      0.74       149
weighted avg       0.77      0.77      0.77       149
```



-After comparing the models, we decided to choose **Support-Vector-Machine(SVM),** Since, it's

- Consistently performed better in terms of precision
- Has High accuracy
- Maintain Type I Error

-SVM balances the trade-off between undercutting and overfitting while handling high-dimensional data well, which make it the ideal choice for our task & objectives.

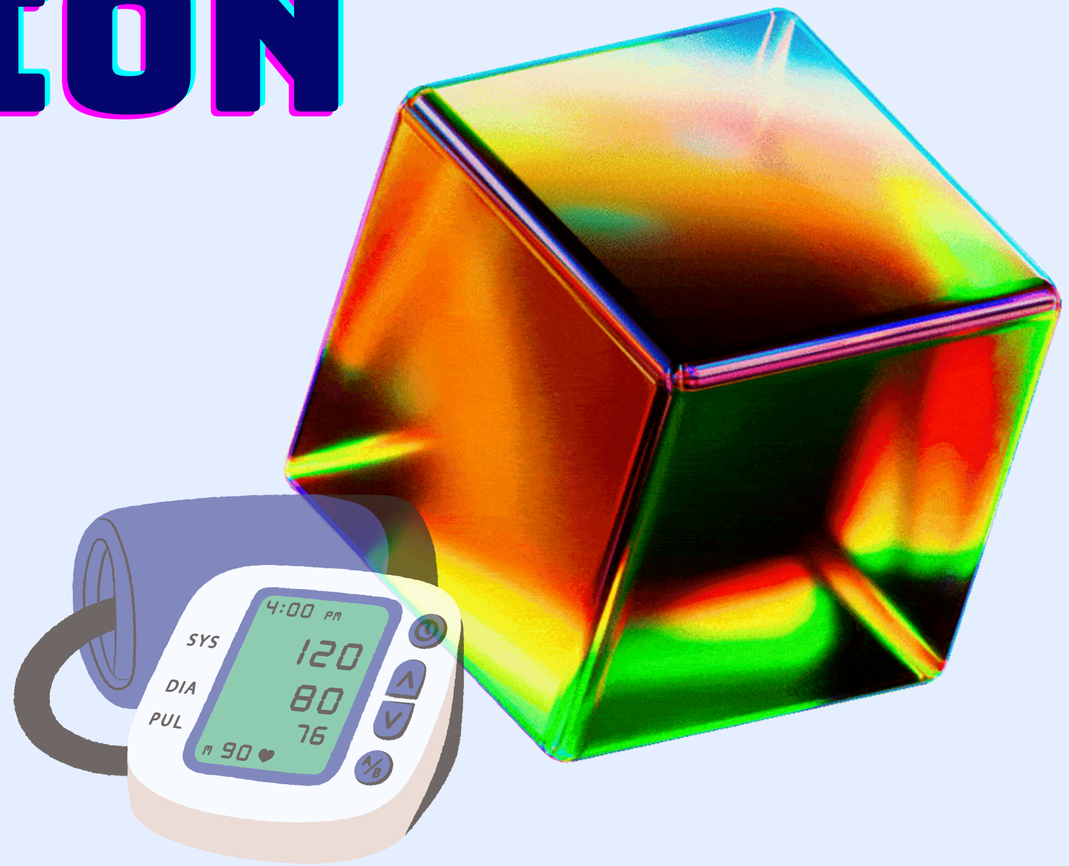


Model Deployment

We have decided to integrate our machine learning algorithm into a web-based platform.

This will allow people to access and use the model for free, making it widely available and importantly, making it an ideal tool for hospital in rural areas.

CONCLUSION



our Diabetes Predictor

has successfully demonstrated the power of machine learning in improving early diabetes diagnosis. By using the Support Vector Machine (SVM) model and deploying it via a web application, we've created a tool that helps healthcare professionals detect diabetes at an earlier stage, enabling timely treatment to prevent complications. The approach is particularly impactful in under-resourced areas, where it provides access to modern diagnostic tools, improving health outcomes in rural or underserved communities. This project highlights how machine learning can bridge the gap in healthcare access, empowering better data-driven decisions globally.

SPECIAL OFFER

Team 6

WEB DEMO!

Team 6