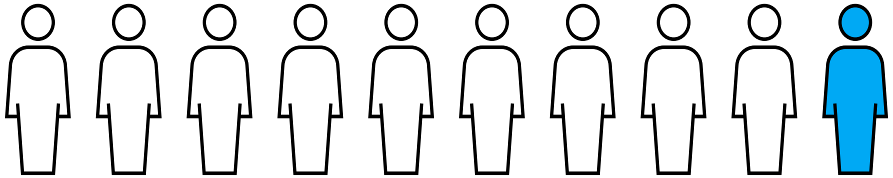
Diabetes Prediction Model

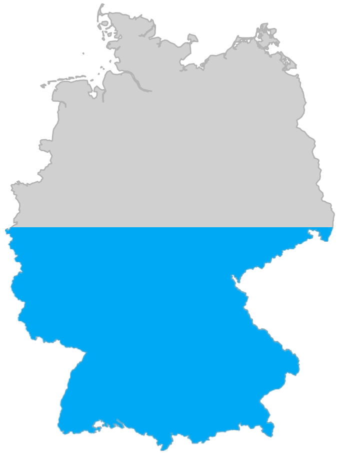
Machine Learning I

Group 4

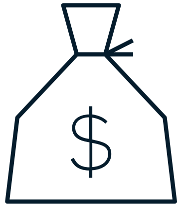
Gizela Thomas
Nitin Jangir
Santiago Botero
Santiago Ruiz
Leonardo v. Kietzell

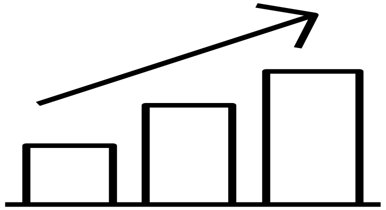


1 in 10 Americans lives with Diabetes



That is 40 million Americans (50% of Germany’s population)





>400 bn USD of total costs (more than Elon Musk’s net worth)

Source: Center for Disease Control and Prevention (CDC), American Diabetes Association (ADA)

Early detection of Diabetes does not only help decrease costs but can increase chances of better treatment

Current screening logic

American Diabetes Association | AACE American Association of Clinical Endocrinology

- **Adults ≥35 years**
- Any age with a **BMI ≥25** (≥23 for Asian Americans) + **one risk factor** (e.g., family history, hypertension, PCOS)
- Repeat **every 3 years** if normal

U.S. Preventive Services TASK FORCE

- **Adults 35-70 years** with **BMI ≥25** (≥23 for Asian Americans)
- Earlier screening for **high-risk racial/ethnic groups**
- Repeat **every 3 years** if normal

Could Machine Learning improve this?

- **Early Detection –** Identifies high-risk patients before symptoms appear.
- **Cost Reduction –** Lowers screening costs by automating risk predictions.
- **Better Prevention –** Helps doctors suggest lifestyle changes earlier.
- **Improved Accuracy –** Reduces false positives and false negatives.
- **Faster Analysis –** Processes large datasets instantly.
- **Scalability –** Can screen millions without extra resources.
- **Continuous Learning –** Improves over time with new patient data.
- **Personalized Screening –** Adapts risk assessment to individual health data.

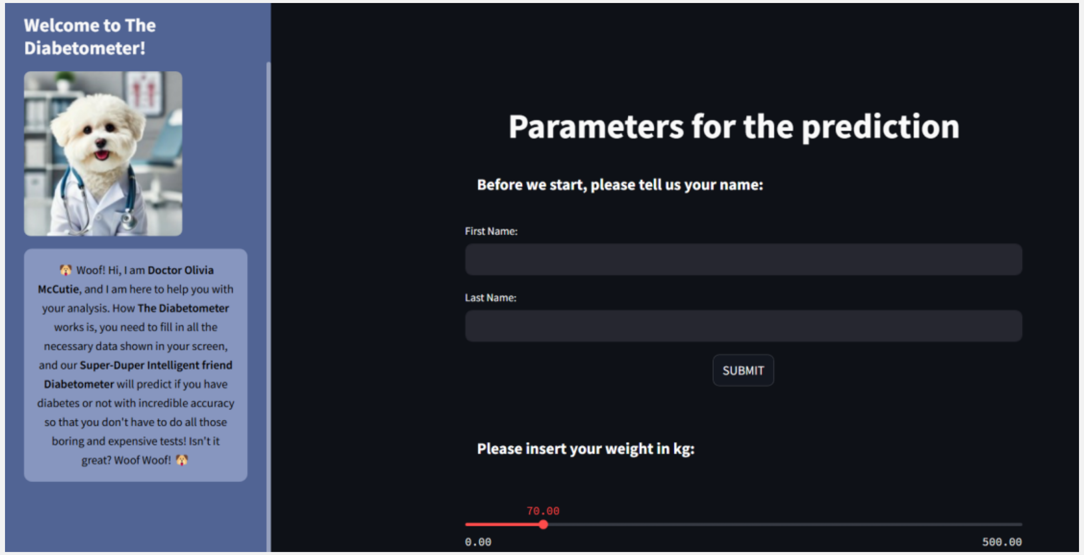
Source: Center for Disease Control and Prevention (CDC), American Diabetes Association (ADA), US Preventive Services Task Force (USPSTF)

We tried to tackle this and dared a first attempt to build an ML model that could give an early Diabetes indication

Our approach

- **Data Sources –** Simulated Diabetes Database and a research study on diabetes screening trends to ensure credibility.
- **Synthetic Data –** Generated additional data using ChatGPT to fill missing features not available in real-world datasets.
- **Key Risk Factors –** Ensured the dataset included important diabetes risk factors and comorbidities for a more complete analysis.
- **Research-Based Approach –** Based data generation on validated studies to align with real-world trends and medical guidelines.
- **Enhanced Dataset –** Created a more comprehensive dataset to improve model accuracy and screening effectiveness.

Our application: The Diabetometer



We simulated the data and features for our first iteration of the model to ensure that real-world applicability is given

We used simulated data for our application…

…to enable a real-world application for our model

1 **No medical and lifestyle features that doctors** or the corresponding agencies[1] **recommend…**

…which **limits** the applicability for a **meaningful diabetes risk factor assessment**

2 Features provided **would require prior testing at doctors by patients…**

…which **defeats the purpose of our pre-visit indicator tool** (“The Diabetometer”)

```
<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
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 13 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Age                           2000 non-null   int64
 1   Ethnicity                     2000 non-null   object
 2   Family History of Diabetes    2000 non-null   object
 3   Hypertension                  2000 non-null   object
 4   Hyperlipidemia                2000 non-null   object
 5   Cardiovascular Disease        2000 non-null   object
 6   Polycystic Ovary Syndrome     2000 non-null   object
 7   Pregnancies                   2000 non-null   int64
 8   Smoking Status                2000 non-null   object
 9   Sleep Duration (hours)        2000 non-null   float64
 10  Stress Levels (scale 1-10)    2000 non-null   int64
 11  BMI (kg/m²)                   2000 non-null   float64
 12  Outcome                       2000 non-null   object
dtypes: float64(2), int64(3), object(8)
memory usage: 203.3+ KB
```

1. Center for Disease Control and Prevention (CDC), American Diabetes Association (ADA), US Preventive Services Task Force (USPSTF)

Using simulated data will require us to *retrain* the model

Obtain real-life data with the exact features that we are currently using

Retrain the model with the actual data and discard any findings from simulated data training

Deploy retrained model in The Diabetometer application and ensure to only run a model trained with real-life data

Designing our Diabetometer

Created with Google Colab, Excel and Visual Studio Code

1

Collection & Simulation of Data

2

Exploratory Data Analysis & Data Preparation

3

Feature Scaling

4

Training, Testing & Deployment

1

Collection & Simulation of Data

Synthetic data simulated with Generative AI

Patients only from America

Women only

Areas for improvement of the input data

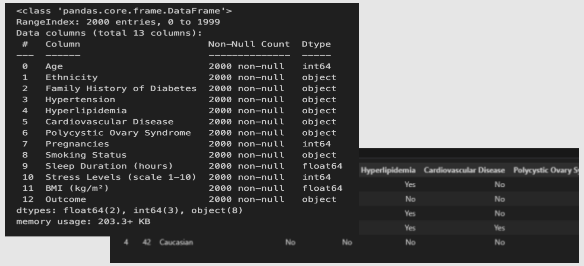
- Replace with actual data
- Increase the amount of data
- Expand geographies
- Expand to all genders

Exploratory Data Analysis & Data Preparation

First view of the existing data

Identification of features

Split of diabetes vs. not-diabetes

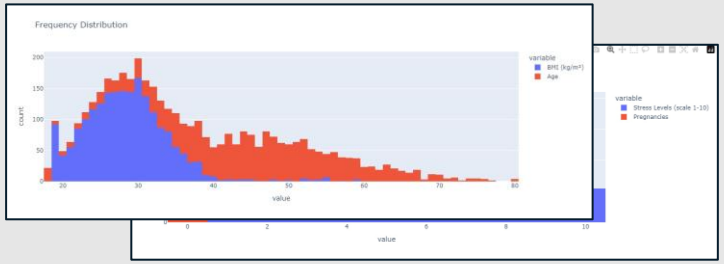


```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 13 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Age                         2000 non-null   int64
 1   Ethnicity                   2000 non-null   object
 2   Family History of Diabetes  2000 non-null   object
 3   Hypertension                2000 non-null   object
 4   Hyperlipidemia              2000 non-null   object
 5   Cardiovascular Disease      2000 non-null   object
 6   Polycystic Ovary Syndrome   2000 non-null   object
 7   Pregnancies                 2000 non-null   int64
 8   Smoking Status              2000 non-null   object
 9   Sleep Duration (hours)      2000 non-null   float64
 10  Stress Levels (scale 1-10)  2000 non-null   int64
 11  BMI (kg/m²)                 2000 non-null   float64
 12  Outcome                     2000 non-null   object
dtypes: float64(2), int64(3), object(8)
memory usage: 203.3+ KB
```

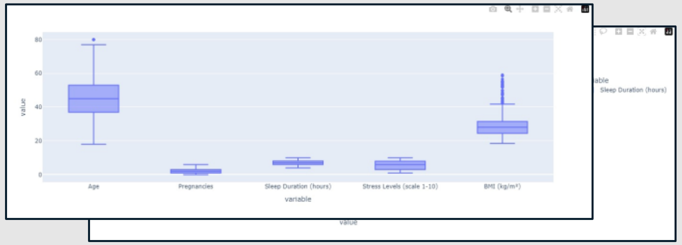
Correlation analysis

Frequency distribution

Outlier visualization



Dropping BMI outliers (>50 BMI which is a medical anomaly)



3

Feature Scaling

Z-score normalization (Standardization) for the numerical variables

Integer encoding for categorical variable smoking status

One-hot-encoding for categorical variable ethnicity

Featuring Scaling & One Hot Encoding

```
#performing z-score standardisation on numerical features
scaler = StandardScaler()
scaler.fit(X_train[['Age', 'Pregnancies', 'Sleep Duration (hours)', 'Stress Levels (scale 1-10)',   'BMI (kg/m²)']])

X_train[['Age', 'Pregnancies', 'Sleep Duration (hours)',    'Stress Levels (scale 1-10)',   'BMI (kg/m²)']] = scaler.transform(X_train[['Age', 'Pregnancies', 'Sleep Duration (hours)', 'Stress Levels (scale 1-10)',   'BMI (kg/m²)']])
X_test[['Age', 'Pregnancies', 'Sleep Duration (hours)', 'Stress Levels (scale 1-10)',   'BMI (kg/m²)']] = scaler.transform(X_test[['Age', 'Pregnancies', 'Sleep Duration (hours)',  'Stress Levels (scale 1-10)',   'BMI (kg/m²)']])



#performing one hot encoding for the Ethnicity column
encoder = OneHotEncoder(sparse_output=False)
encoder.fit(X_train[['Ethnicity']])

X_train_encoded = encoder.transform(X_train[['Ethnicity']])
X_test_encoded = encoder.transform(X_test[['Ethnicity']])


X_train_encoded_df = pd.DataFrame(X_train_encoded, columns=encoder.get_feature_names_out(['Ethnicity']))
X_test_encoded_df = pd.DataFrame(X_test_encoded, columns=encoder.get_feature_names_out(['Ethnicity']))


X_train = pd.concat([X_train.drop(columns=['Ethnicity']).reset_index(drop=True), X_train_encoded_df], axis=1)
X_test = pd.concat([X_test.drop(columns=['Ethnicity']).reset_index(drop=True), X_test_encoded_df], axis=1)
```

3 Deep dive: We selected key features to predict Diabetes

- **One-Hot Encoding –** Converted categorical variables into binary features so the model can interpret them correctly
- **Z-Score Normalization –** Scaled numerical features to have a mean of 0 and a standard deviation of 1, preventing bias from differing scales
- **Bias Prevention –** Ensured no single feature dominates due to larger values
- **Improved Accuracy –** Standardization and encoding help the model learn patterns more effectively
- **Fairer Predictions –** Reduces unintended weight differences across features, leading to more balanced outcomes

General		
Age	BMI	Ethnicity
Health		
Hypertension	Hyperlipidemia	Cardiovascular Disease
Lifestyle		
Smoking	Sleep	Stress
Other		
Family History of Diabetes	PCOS[1]	Pregnancy

Specific for women

1. Polycystic Ovary Syndrome

4

Training & Testing

Training and Hyperparameter tuning with Logistic Regression and Grid Search

Monitor false negatives

Concluded with a **cross-validation accuracy of ~0.599**

Model Training & Hyperparameter Tuning

```python
param_grid = {
    'penalty': ['elasticnet'],
    'C': [0.001, 0.01, 0.1, 1, 10],
    'l1_ratio': [0.1, 0.25, 0.5, 0.75, 0.9],
    'solver': ['saga']
}

log_reg = LogisticRegression(max_iter=1000)
grid_search = GridSearchCV(log_reg, param_grid, cv=10, scoring='recall')
#cv=5 means number of times the training set is parted while performing cross-validation
#scoring=recall tells the gridSearch algorithm to optimise the model to reduce false-negatives


grid_search.fit(X_train, y_train)

print("Best Hyperparameters:", grid_search.best_params_)
print("Best Cross-validation Accuracy:", grid_search.best_score_)


best_model = grid_search.best_estimator_
```

```
Best Hyperparameters: {'C': 10, 'l1_ratio': 0.1, 'penalty': 'elasticnet', 'solver': 'saga'}
Best Cross-validation Accuracy: 0.5991608391608392
```

Model testing and error metric evaluation with predict and predict_proba

Defined threshold of 0.3 to reduce false negatives

Concluded with **an accuracy of ~0.71-0.72**

Model Testing & Error Metric Evaluation

```python
y_test_pred = best_model.predict(X_test)

print(classification_report(y_test, y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.76      0.81      0.79       251
           1       0.63      0.57      0.60       145

    accuracy                           0.72       396
   macro avg       0.70      0.69      0.69       396
weighted avg       0.71      0.72      0.72       396
```

```python
#ALTERNATE
#using the predict_proba method to calculate the probabilities and defining a custom probability threshold of 35% to minimise false positives.

y_pred_proba = best_model.predict_proba(X_test)[:, 1]
y_pred_custom = (y_pred_proba > 0.35).astype(int)


print("Classification Report with custom threshold:")
print(classification_report(y_test, y_pred_custom))
```

```
Classification Report with custom threshold:
              precision    recall  f1-score   support

           0       0.83      0.69      0.75       251
           1       0.58      0.77      0.66       145

    accuracy                           0.71       396
   macro avg       0.71      0.73      0.71       396
weighted avg       0.74      0.71      0.72       396
```

We followed the classical steps to define the Machine Learning model from extraction to testing & defining thresholds

Data Extraction through CSV

EDA to analyze **feature relationships, correlation, frequency distribution, outliers** & **missing values**

Converting **binary Categorical Variables into integers**

Eliminating outliers

Integer Encoding for Smoking_Status

Train-Test spilt

Fitting **Training Data & Transforming** both datasets

One-Hot-Encoding for remaining categorical variables

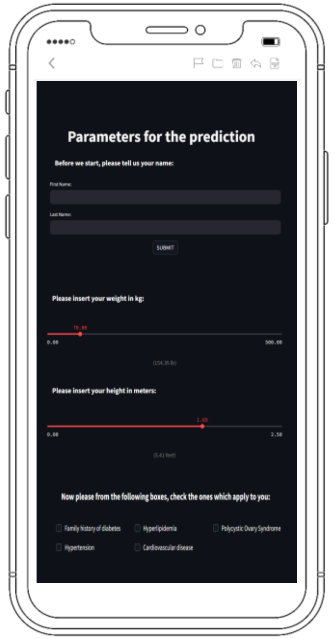
Using **LogRegression & GridSearchCV** to estimate the best model

Testing & defining threshold to reduce FN

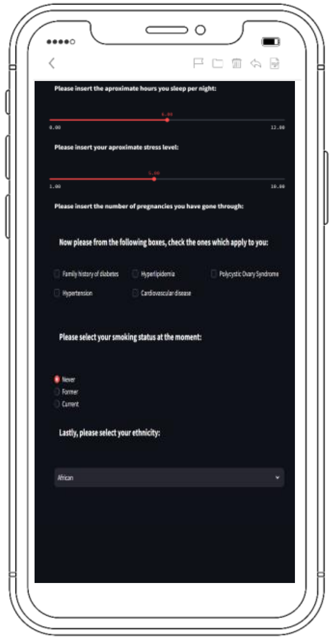
How does our Streamlit application work for you at home

1 → 2 → 3

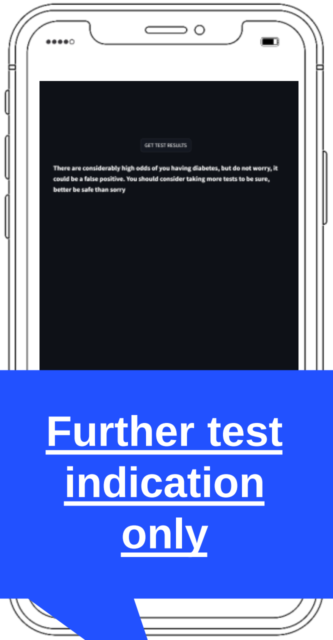
Input general data



Select risk factors



Receive test indication



Further test indication only

Diabetometer could significantly increase efficiency in tackling the Diabetes epidemic in the US

The Times

Diabetometer changes the way doctors work

Diabetometer, an ML-powered app, transforms diabetes management with an easy, do-at-home test. Its ML model helps users to get a first indication if further testing is needed. By integrating data-driven strategies, it enhances patient care, reducing risks and hospital visits.

The app's mascot Olivia

Higher Accuracy – Diabetometer reduces false diagnoses by analyzing more risk factors. Needs diverse training data.

Faster Screening – Automates risk assessment for instant results. Requires EHR integration.

Personalized Predictions – Adapts to individual health and lifestyle. Needs continuous model updates.

Fair & Unbiased – Ensures consistent results across demographics. Requires diverse datasets and monitoring

Self-Improving – Learns from new patient data over time. Needs regular validation.