**Custom Explainability Solution**

A **custom explainability solution** in machine learning refers to techniques that make complex models more understandable and interpretable to users by providing tailored explanations for specific use cases, models, or stakeholders.

**Purpose**: The goal is to provide relevant, accurate, and interpretable explanations that help stakeholders understand the model’s decision-making process, build trust, and detect potential biases or errors.

**Scenario: Explainability for Diabetes Prediction Model**

Context: A healthcare provider uses a machine learning model to predict patients' diabetes risk based on factors like glucose level, BMI, and age. They aim to provide tailored explanations for doctors, patients, and regulators**.**

**Challenges:**

* Doctors need detailed insights into each prediction to guide patient care.
* Patients want clear, actionable feedback on their health risks.
* Regulators require proof that the model is fair and unbiased.

**Solution:**

1. Doctors: A dashboard shows each feature's impact on the prediction (e.g., high glucose increases risk), with SHAP values detailing the contribution of each factor.
2. Patients: Simplified, actionable feedback (e.g., "Your risk is high; consider a healthier diet") helps them understand their health without technical jargon.
3. Regulators: Reports with SHAP-based fairness metrics across demographics ensure compliance with anti-discrimination standards and model transparency.

**Techniques for custom explainability solution**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

import shap

import matplotlib.pyplot as plt

# Load the custom dataset

data = pd.DataFrame({

"Glucose": [148, 85, 183, 89, 137, 116, 78, 115, 197, 125, 110, 168, 139, 189, 166],

"BMI": [33.6, 26.6, 23.3, 28.1, 43.1, 25.6, 31.0, 35.3, 30.5, 0.0, 37.6, 38.0, 27.1, 30.1, 25.8],

"Age": [50, 31, 32, 21, 33, 30, 26, 29, 53, 54, 30, 34, 57, 59, 51],

"Outcome": [1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1]

})

# Separate features and target

X = data.drop("Outcome", axis=1)

y = data["Outcome"]

# Split into training and testing data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Train logistic regression model

model = LogisticRegression(max\_iter=200)

model.fit(X\_train, y\_train)

# Predict and calculate accuracy for Model Regulator's report

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

# Create a SHAP explainer

explainer = shap.Explainer(model, X\_train)

# Select a single sample from the test set (for example, the first instance)

new\_sample = X\_test.iloc[0]

# Calculate SHAP values for the selected sample

new\_sample\_shap\_values = explainer(new\_sample.values.reshape(1, -1))

# Predict the class for the selected sample

predicted\_class = model.predict(new\_sample.values.reshape(1, -1))[0]

# Function to generate reports and display plots based on user role

def generate\_report(instance, shap\_values\_instance, prediction, role):

report = f"### Role: {role}\n"

report += f"\*\*Decision:\*\* {'Diabetes Positive' if prediction == 1 else 'Diabetes Negative'}\n\n"

report += "### Patient Data:\n"

report += f"- \*\*Glucose:\*\* {instance[0]}\n"

report += f"- \*\*BMI:\*\* {instance[1]}\n"

report += f"- \*\*Age:\*\* {instance[2]}\n\n"

if role == "Patient":

report += ("### Explanation for Patients:\n"

"Your risk of diabetes is predicted based on your health indicators. "

"A higher glucose level, BMI, or age can increase the risk of diabetes.\n")

print(f"\nGenerated Report for Patient:\n{report}\n")

# Display SHAP Force Plot for Patient

print("SHAP Force Plot for Patient:")

shap.force\_plot(explainer.expected\_value, shap\_values\_instance, instance, feature\_names=X.columns, matplotlib=True)

plt.show()

elif role == "Doctor":

report += ("### Explanation for Doctors:\n"

"The prediction is based on glucose levels, BMI, and age as key indicators for diabetes risk.\n")

print(f"\nGenerated Report for Doctor:\n{report}\n")

# Display SHAP Waterfall Plot for Doctor

print("SHAP Waterfall Plot for Doctor:")

shap.waterfall\_plot(new\_sample\_shap\_values[0])

plt.show()

elif role == "Model Regulator":

report += ("### Explanation for Model Regulators:\n"

"This model's accuracy on the test set is a critical performance indicator.\n")

report += f"- \*\*Model Accuracy:\*\* {accuracy:.2f}\n"

print(f"\nGenerated Report for Model Regulator:\n{report}\n")

# Generate and display reports for each role

for role in ["Patient", "Doctor", "Model Regulator"]:

generate\_report(new\_sample.values, new\_sample\_shap\_values.values[0], predicted\_class, role)

Output:

Generated Report for Patient:

### Role: Patient

\*\*Decision:\*\* Diabetes Negative

### Patient Data:

- \*\*Glucose:\*\* 125.0

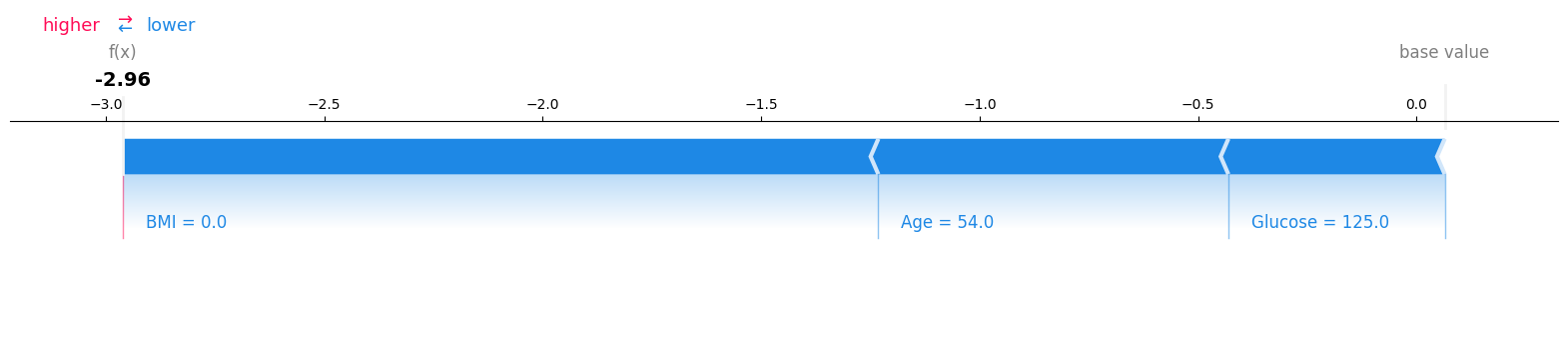
- \*\*BMI:\*\* 0.0

- \*\*Age:\*\* 54.0

### Explanation for Patients:

Your risk of diabetes is predicted based on your health indicators. A higher glucose level, BMI, or age can increase the risk of diabetes.

SHAP Force Plot for Patient:



Generated Report for Doctor:

### Role: Doctor

\*\*Decision:\*\* Diabetes Negative

### Patient Data:

- \*\*Glucose:\*\* 125.0

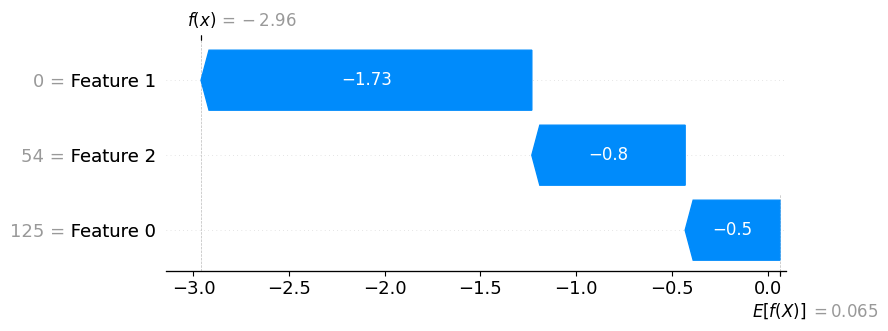
- \*\*BMI:\*\* 0.0

- \*\*Age:\*\* 54.0

### Explanation for Doctors:

The prediction is based on glucose levels, BMI, and age as key indicators for diabetes risk.

SHAP Waterfall Plot for Doctor:



Generated Report for Model Regulator:

### Role: Model Regulator

\*\*Decision:\*\* Diabetes Negative

### Patient Data:

- \*\*Glucose:\*\* 125.0

- \*\*BMI:\*\* 0.0

- \*\*Age:\*\* 54.0

### Explanation for Model Regulators:

This model's accuracy on the test set is a critical performance indicator.

- \*\*Model Accuracy:\*\* 0.67

Difference between custom explainability solution and explainability:

