GUNJAN CHANDRA

UNIVERSITY OF OULU, MAY 02, 2023

# *MACHINE LEARNING METHODS FOR HEALTHCARE DATA ANALYSIS*

# *WHAT IS MACHINE LEARNING?*

Machine learning is a type of artificial intelligence (AI) that enables computers to learn from data and experience, rather than being explicitly programmed.

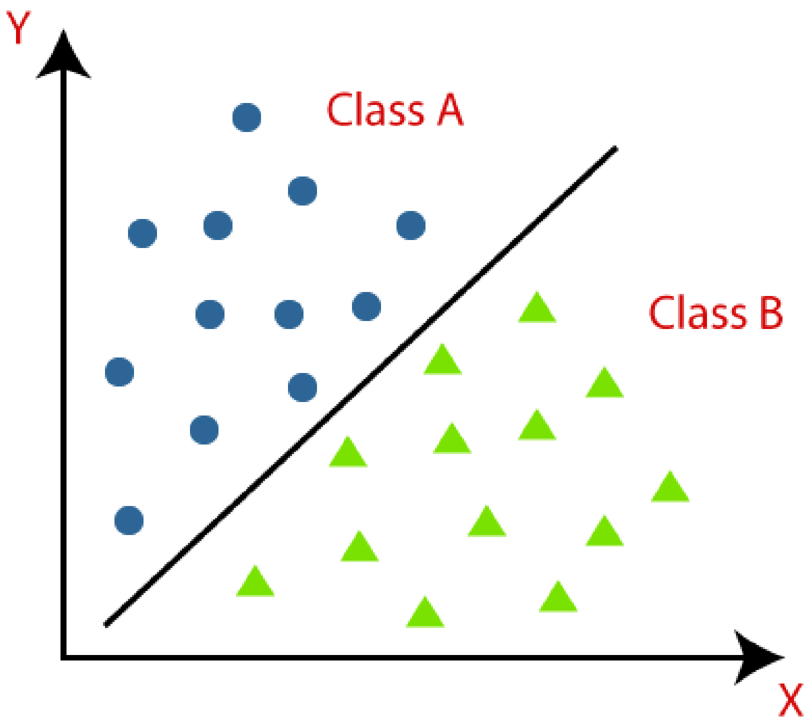
# TYPES OF MACHINE LEARNING

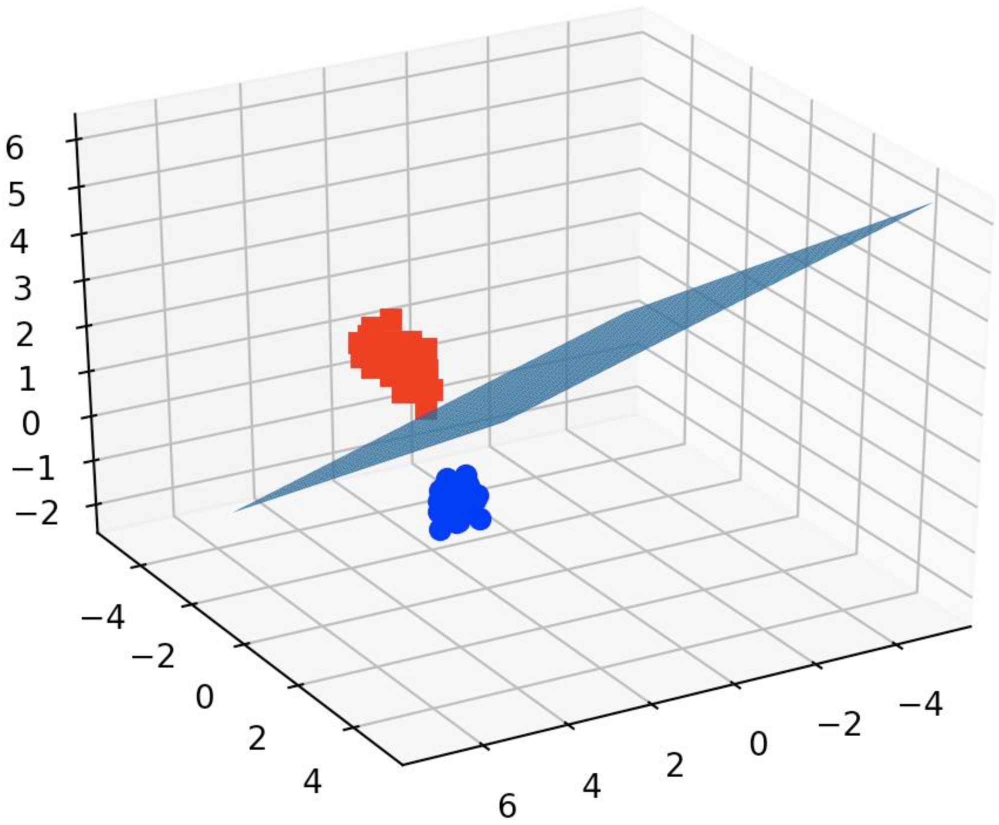
Supervised learning uses labelled data to predict an output variable.

Unsupervised learning looks for patterns or structure in unlabelled data.

Reinforcement learning learns from feedback to make decisions that maximize a reward.

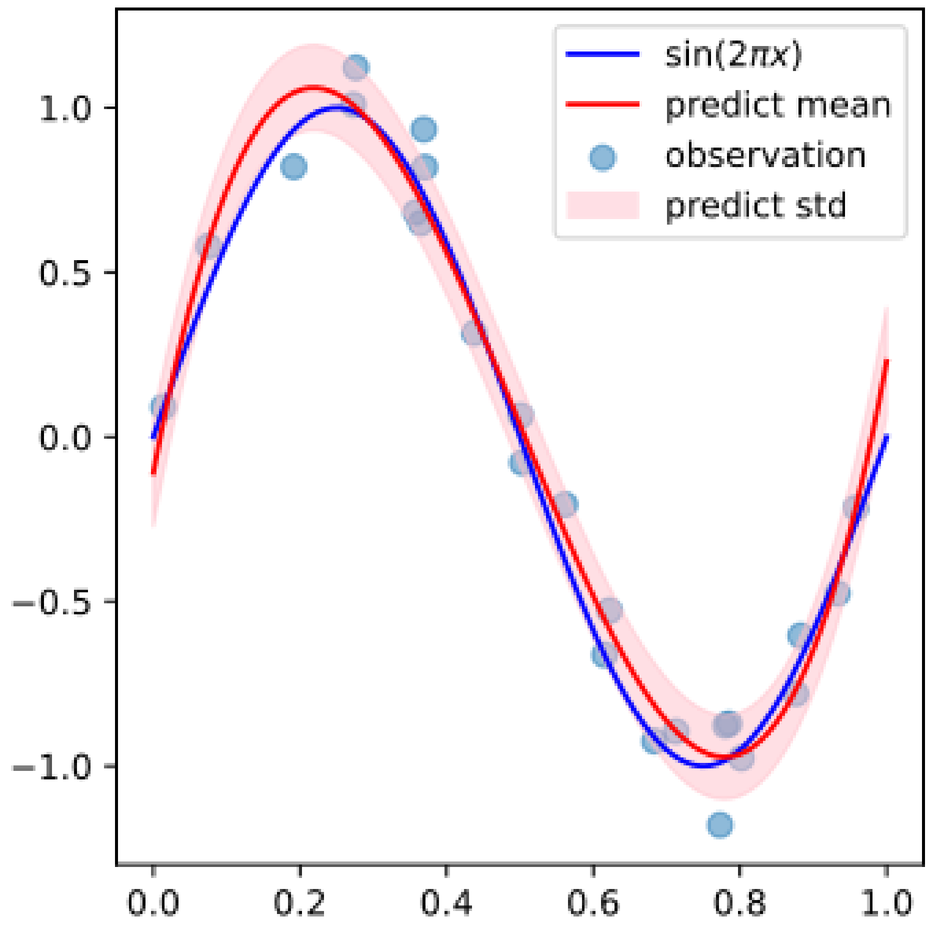
There are also subtypes and variations of machine learning algorithms, and specific techniques for specific tasks.

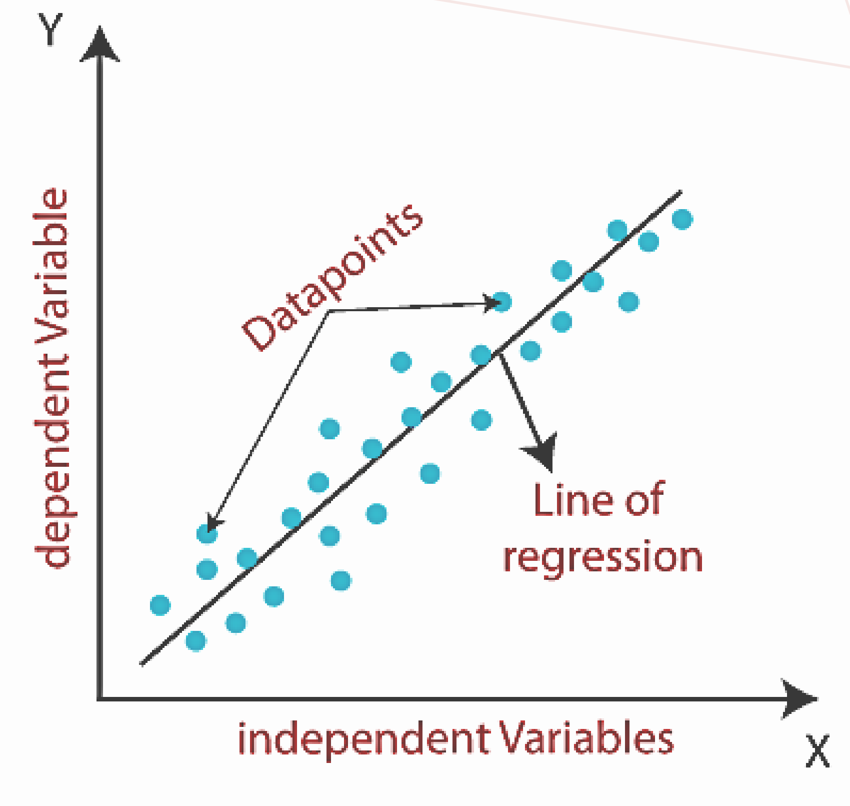



Classification predicts a categorical output variable based on input.

# *SUPERVISED LEARNING*

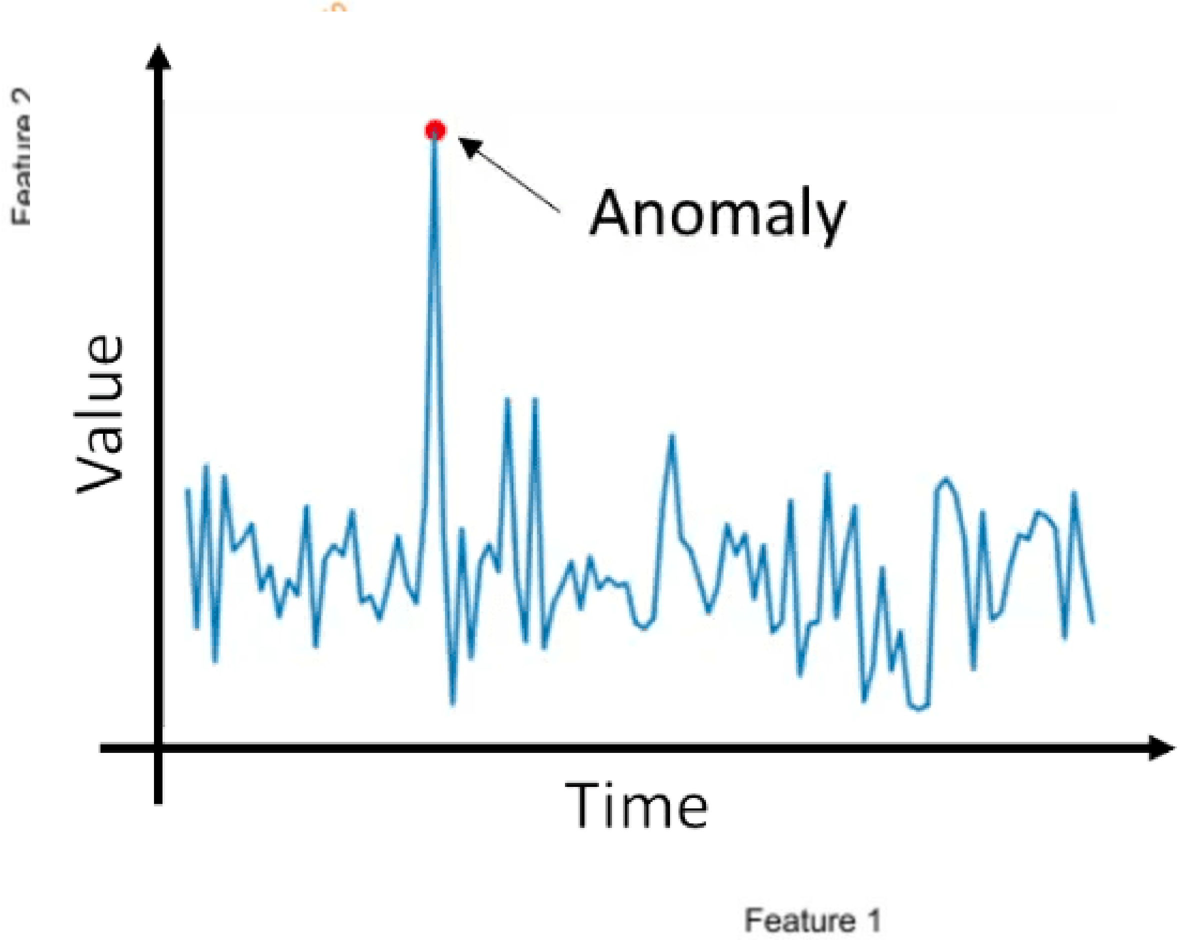



Regression predicts a continuous output variable based on input features

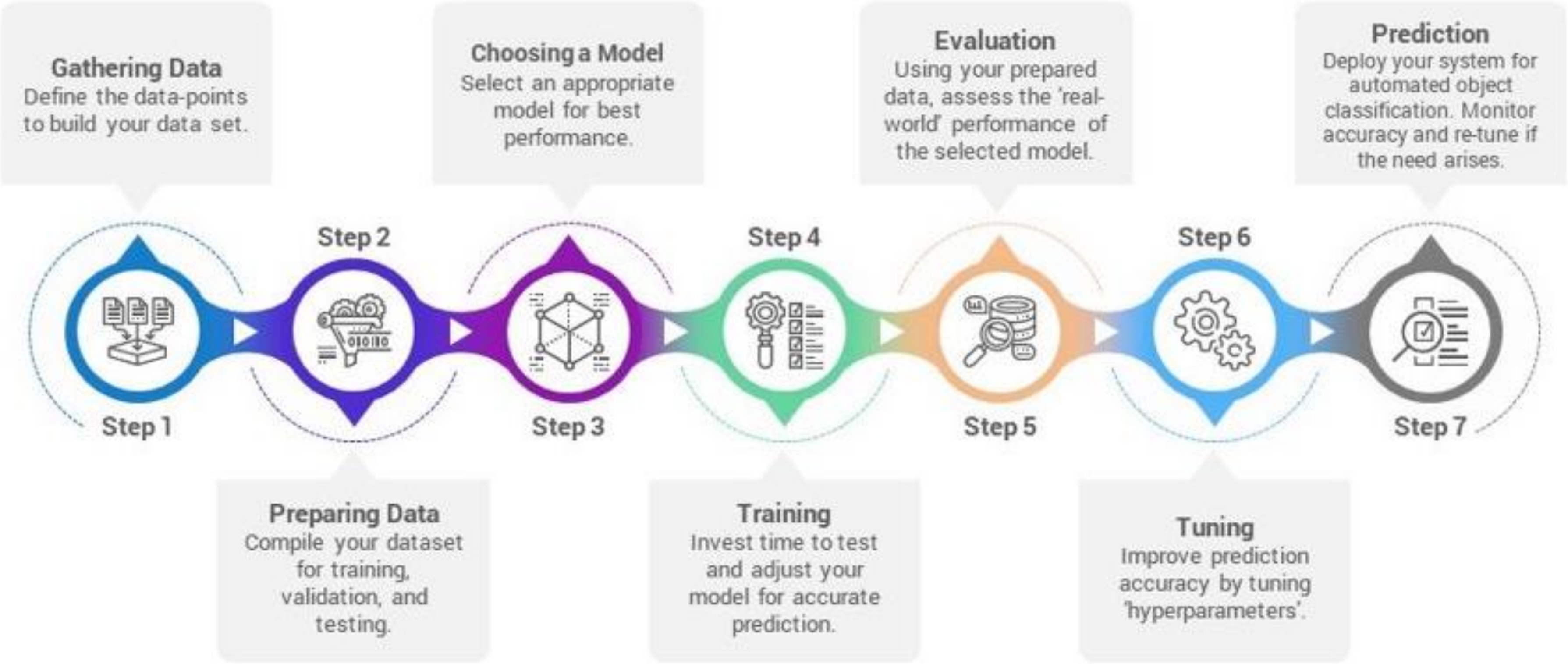
# *SUPERVISED LEARNING*

Principal Component Analysis (PCA)
K-Means Clustering
Anomaly detection

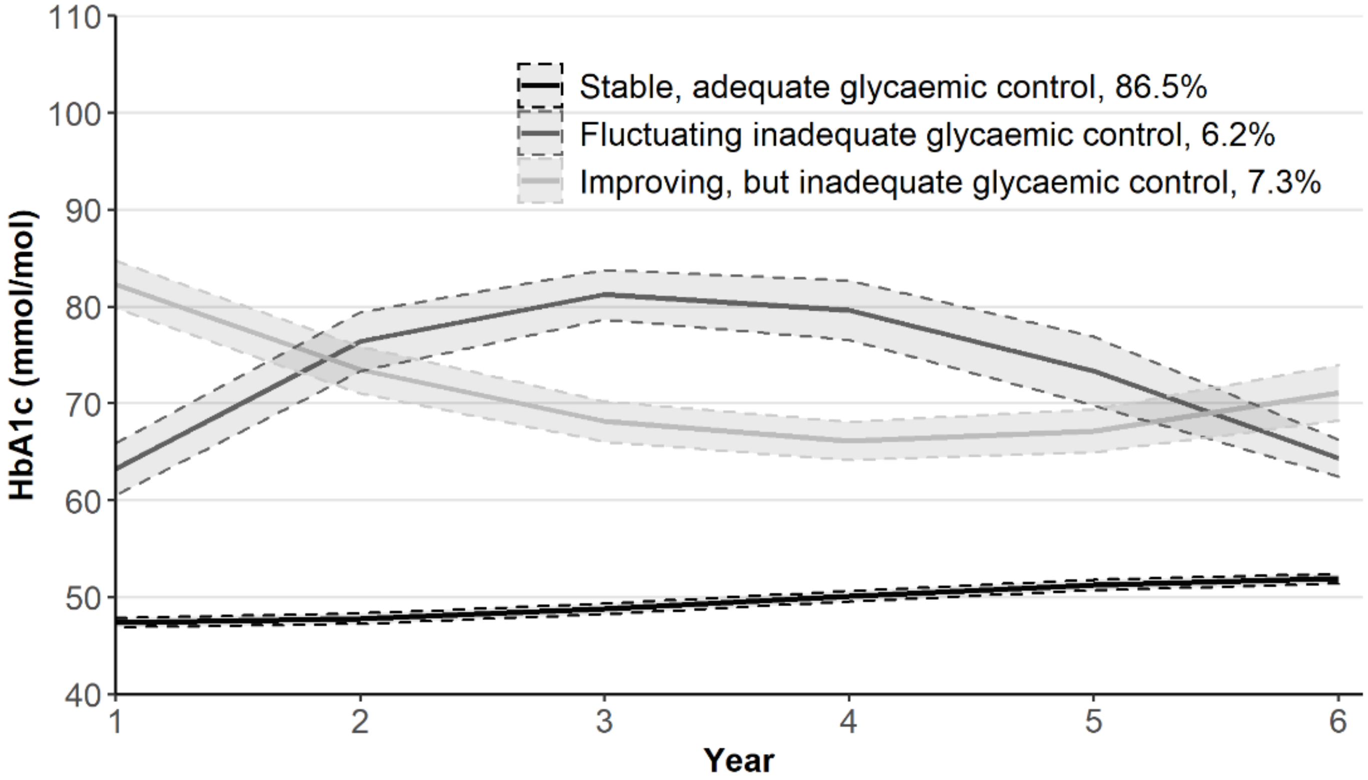



# UNSUPERVISED LEARNING

- Clustering
- Dimension reduction
- Anomaly detection




# *MACHINE LEARNING PIPELINE*



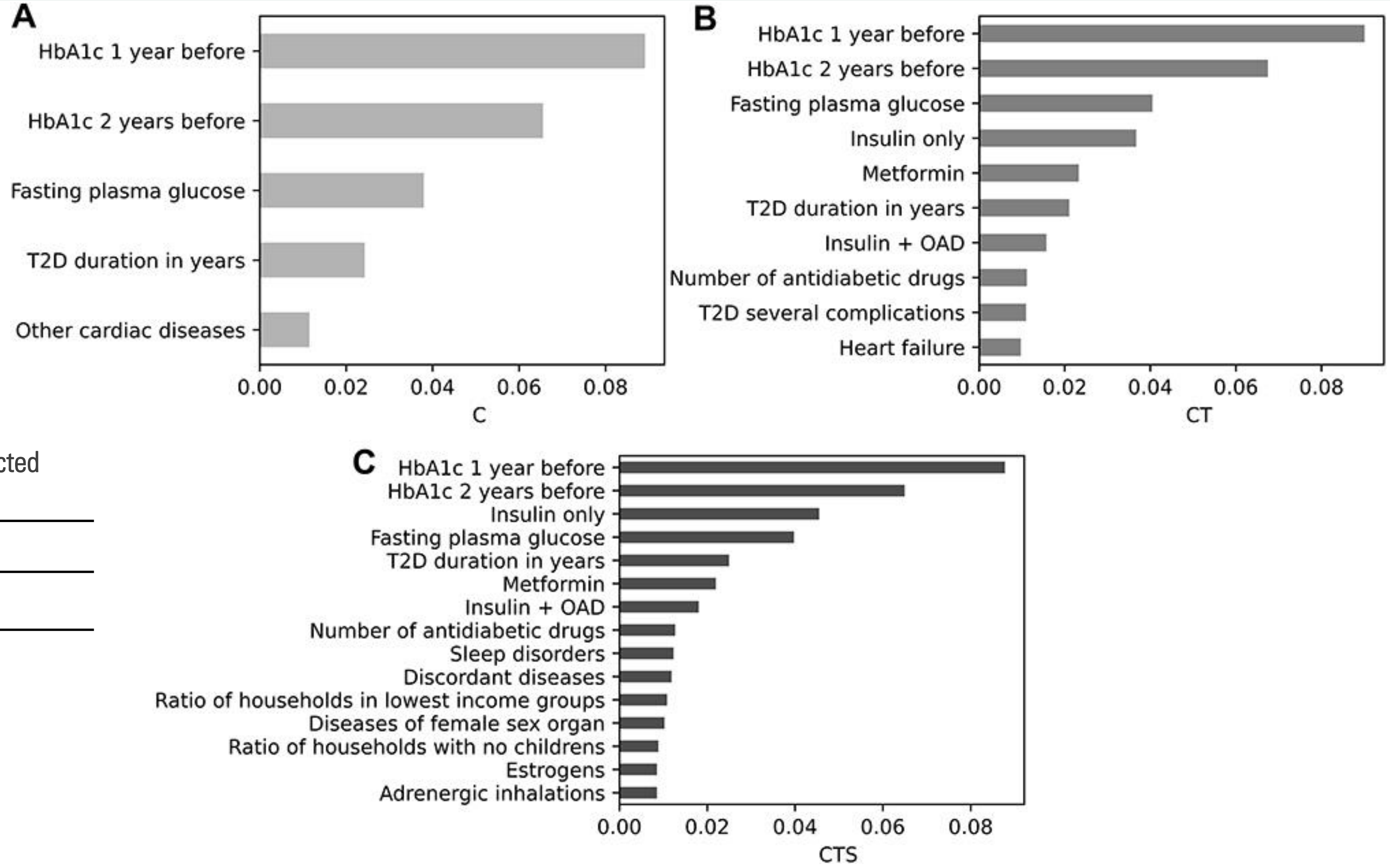

# *DATA-DRIVEN IDENTIFICATION OF LONG-TERM GLYCEMIA CLUSTERS AND THEIR INDIVIDUALIZED PREDICTORS IN FINNISH PATIENTS WITH TYPE 2 DIABETES*

# *DATA PRE-PROCESSING*

- Data cleaning: This involves detecting and correcting errors or inconsistencies in the data.
- Data transformation: This involves transforming the data into a suitable format for ML algorithms.
- Feature engineering: This involves selecting and creating relevant features from the raw data that can improve the performance of ML algorithms.
- Data integration: This involves combining data from multiple sources to create a more comprehensive dataset for analysis.

```python
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler
from sklearn.feature_selection import SelectKBest, chi2

# Load the dataset
data = pd.read_csv('data.csv')

# Drop the rows with missing values
data.dropna(inplace=True)

# Transform categorical variables to numerical
labelencoder = LabelEncoder()
data['Gender'] = labelencoder.fit_transform(data['Gender'])
onehotencoder = OneHotEncoder()
data = pd.get_dummies(data, columns=['Education', 'Marital Status'], prefix=['Education', 'Marital'])

# Create new features
data['Age Income Ratio'] = data['Age'] / data['Income']
data['Credit Score Income Ratio'] = data['Credit Score'] / data['Income']

# Merge data from multiple sources
data2 = pd.read_csv('data2.csv')
dataset = pd.merge(data, data2, on='Patient ID')

# split data into training and testing sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(dataset.drop('target', axis=1),dataset['target'], test_size=0.2, random_state=0)

# Preprocess data using StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Impute missing values
imputer = SimpleImputer(strategy='mean')
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)

# Feature selection using chi2
from sklearn.feature_selection import SelectKBest, chi2
selector = SelectKBest(chi2, k=5)
X_train = selector.fit_transform(X_train, y_train)
X_test = selector.transform(X_test)
```

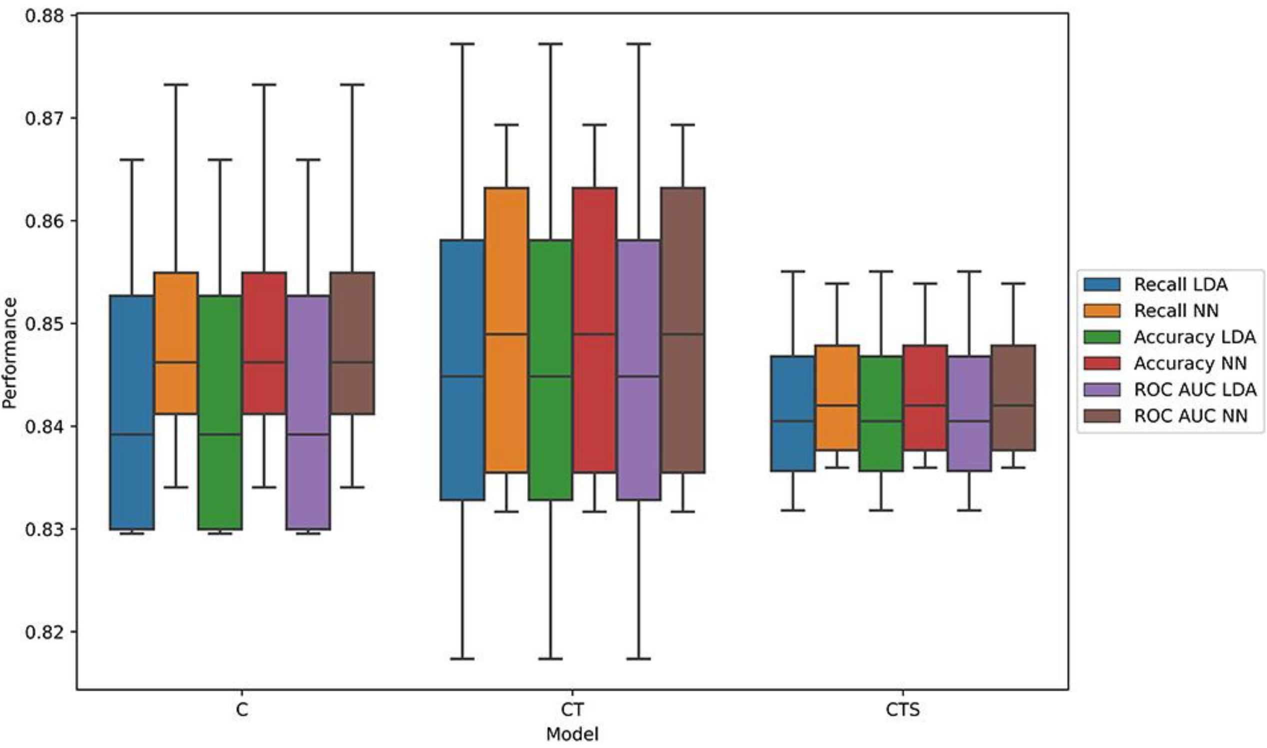
# FEATURE IMPORTANCE AND SELECTION

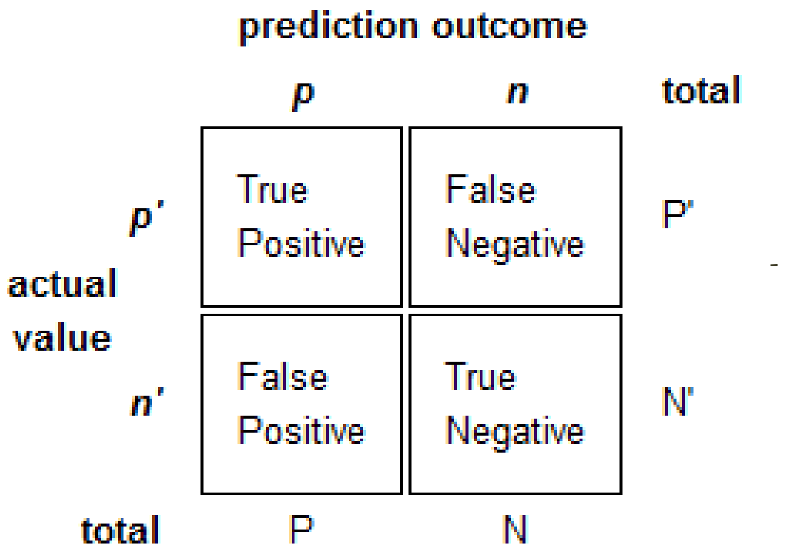



| Type of predictors | Total number of predictors | Number of selected predictors |
|---|---|---|
| Clinical (C) | 83 | 5 |
| Clinical + Treatment (CT) | 233 | 10 |
| Clinical + Treatment + SES (CTS) | 299 | 15 |

# MODEL SELECTION, TRAINING, AND EVALUATION

```python
from sklearn.model_selection import KFold
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score

# Define the models to evaluate
models = [('Linear Discriminant Analysis', LinearDiscriminantAnalysis()),
          ('MLP Classifier', MLPClassifier(hidden_layer_sizes=(100,)))]

# Define the number of folds for k-fold cross-validation
n_folds = 4

# Split data into k-folds and evaluate each model on each fold
for model_name, model in models:
    kfold = KFold(n_splits=n_folds, shuffle=True, random_state=42)
    scores = []
    for train_idx, test_idx in kfold.split(X, y):
        X_train, y_train = X[train_idx], y[train_idx]
        X_test, y_test = X[test_idx], y[test_idx]
        model.fit(X_train, y_train)
        score = model.score(X_test, y_test)
        scores.append(score)
    mean_score = sum(scores) / n_folds
    print(f'{model_name}: {mean_score:.3f}')
```

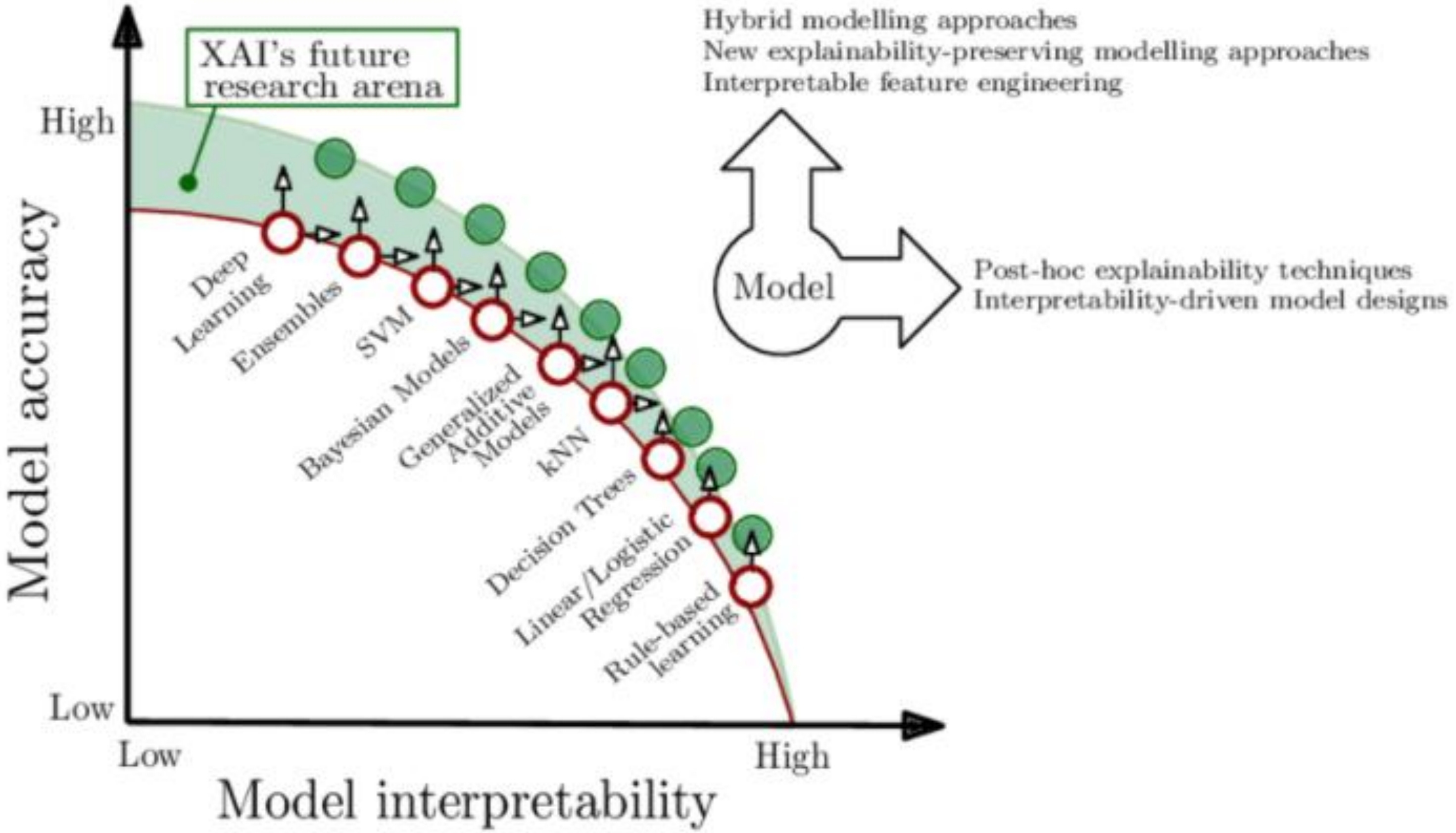



| Predictors | Model | Confusion Matrix | | | | F1 Score | Balanced Accuracy | ROC AUC |
|---|---|---|---|---|---|---|---|---|
| Clinical | LDA | True | Predicted | | Class | 0.69 | 0.84 | 0.92 |
| | | | 4394 | 726 | Adequate | | | |
| | | | 68 | 329 | Inadequate | | | |
| | NN | True | Predicted | | Class | 0.66 | 0.85 | 0.91 |
| | | | 4191 | 929 | Adequate | | | |
| | | | 47 | 350 | Inadequate | | | |
| Clinical + Treatment | LDA | True | Predicted | | Class | 0.69 | 0.85 | 0.92 |
| | | | 4405 | 715 | Adequate | | | |
| | | | 67 | 330 | Inadequate | | | |
| | NN | True | Predicted | | Class | 0.66 | 0.85 | 0.91 |
| | | | 4202 | 918 | Adequate | | | |
| | | | 48 | 349 | Inadequate | | | |
| Clinical + Treatment + SES | LDA | True | Predicted | | Class | 0.69 | 0.84 | 0.92 |
| | | | 4402 | 714 | Adequate | | | |
| | | | 70 | 326 | Inadequate | | | |
| | NN | True | Predicted | | Class | 0.66 | 0.84 | 0.91 |
| | | | 4249 | 867 | Adequate | | | |
| | | | 57 | 339 | Inadequate | | | |

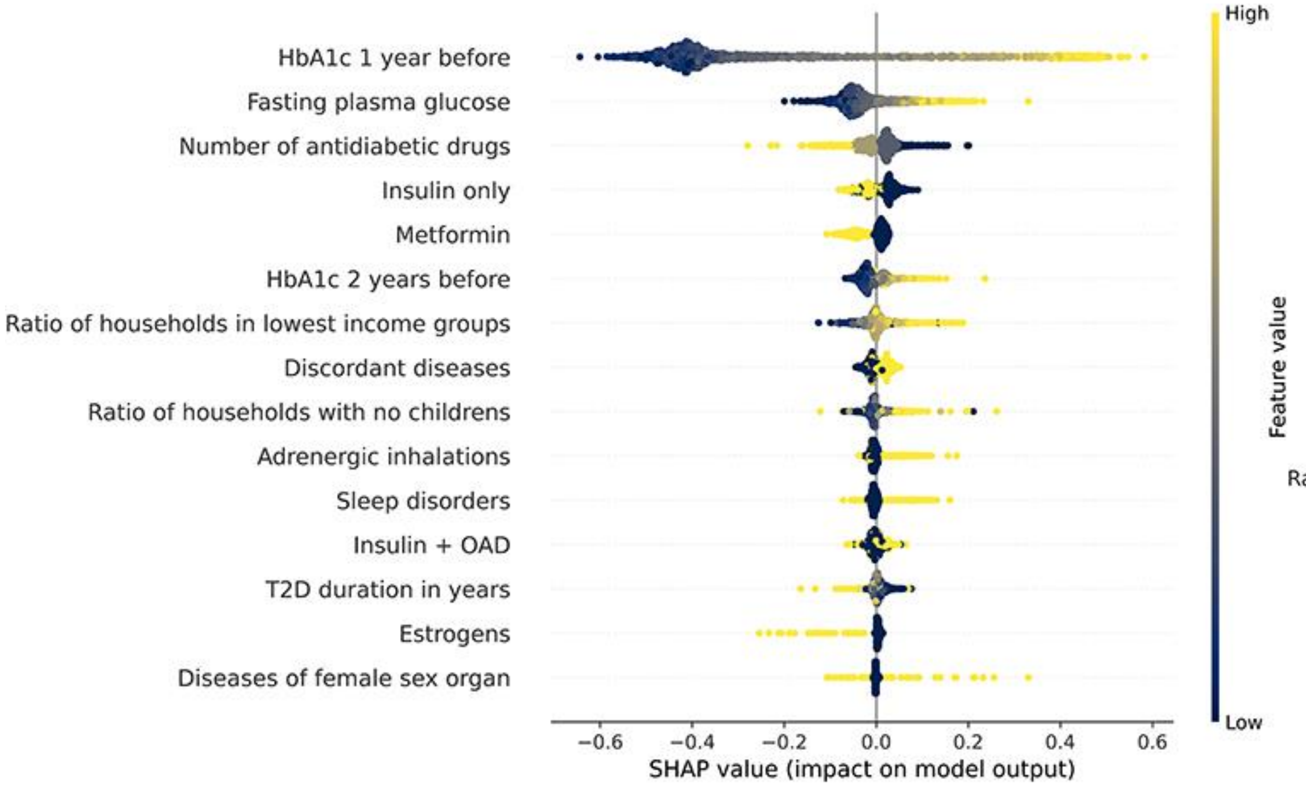
**Abbreviations**: LDA, linear discriminant analysis; NN, neural network; ROC AUC; receiver operating characteristic area under the curve; SES, socio-economic status.

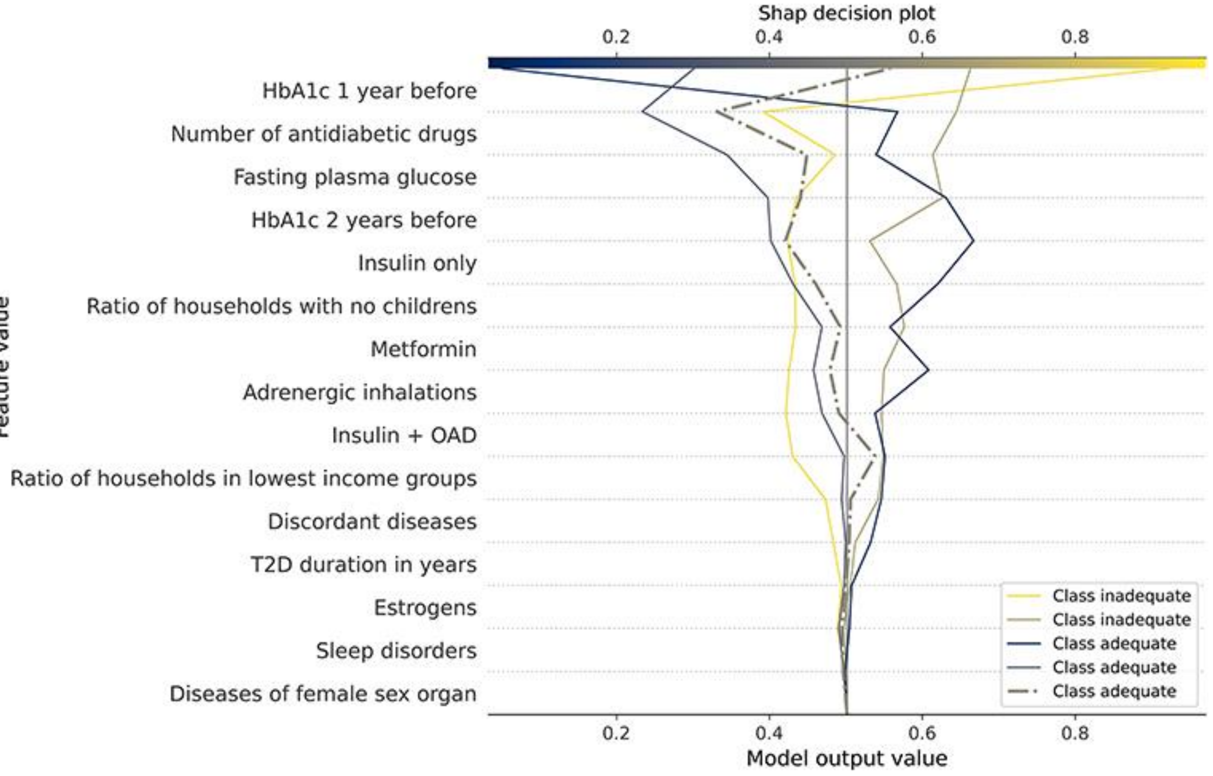
# Accuracy vs Interpretability Trade-off




Explain to Justify
Explain to Control
Explain to Discover
Explain to Improve

# *WHY EXPLAINABLE?*




```
# Create a SHAP explainer object for the trained model
explainer = shap.Explainer(model, X_train)

# Generate a SHAP Bee Swarm plot for the first 100 test samples
shap_values = explainer(X_test[:100])
shap.plots.beeswarm(shap_values)

# Generate a SHAP Decision plot for the first test sample
shap_values = explainer(X_test[0])
shap.decision_plot(explainer.expected_value, shap_values[0], X_test[0])
```

# *SHAP PLOT*

# *THANK YOU!*

QUESTION?