Informatics Bootcamp:
Artificial Intelligence

Dr. Mostafa Rezapour

June 4, 2025

Wake Forest students attend their calculus class with professor Mostafa Rezapour outside on Manchester Plaza at the fire pits on Friday, March 12, 2021.

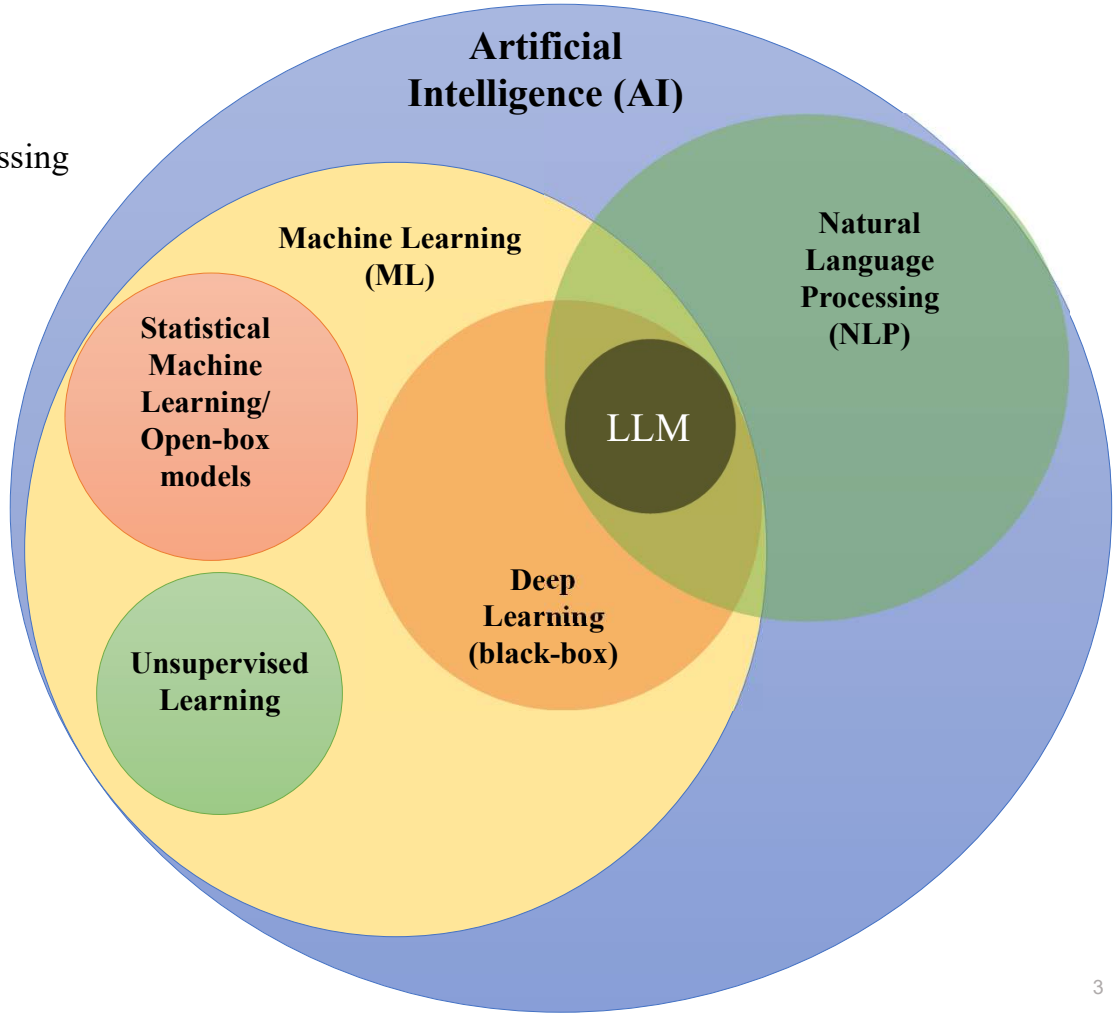
Wake Forest University
School of Medicine

CAIR
CENTER FOR ARTIFICIAL INTELLIGENCE RESEARCH

General overview of Artificial Intelligence (AI)

Artificial Intelligence (AI)

- Machine Learning (ML)
- Natural Language Processing
- Computer Vision
- Expert Systems



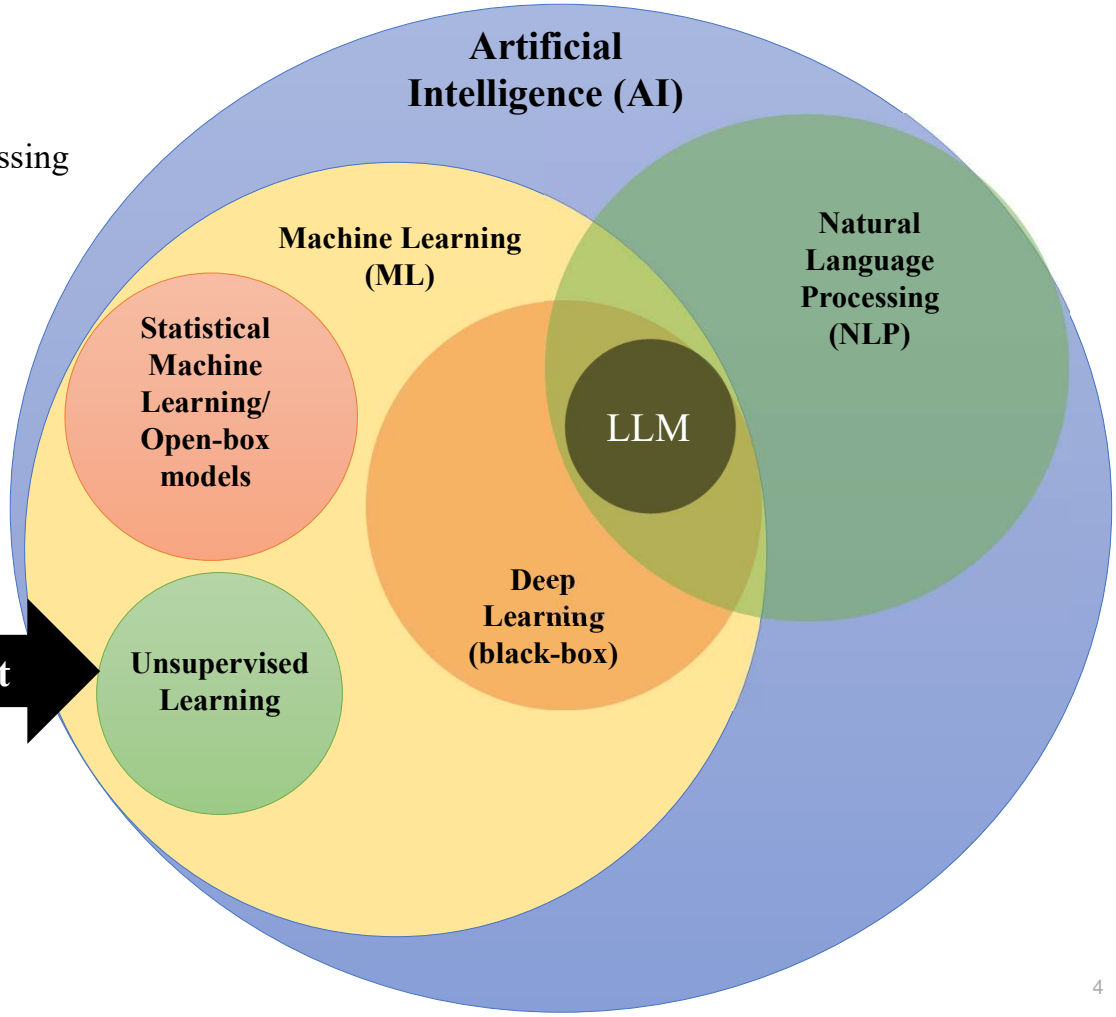
Machine Learning (ML)

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

General overview of AI/ML models and their relevance to regenerative medicine

Artificial Intelligence (AI)

- Machine Learning (ML)
- Natural Language Processing
- Computer Vision
- Expert Systems



Let's talk about

Machine Learning (ML)

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

What is Unsupervised Learning, and how can it be applied in our research?

What is Unsupervised Learning?

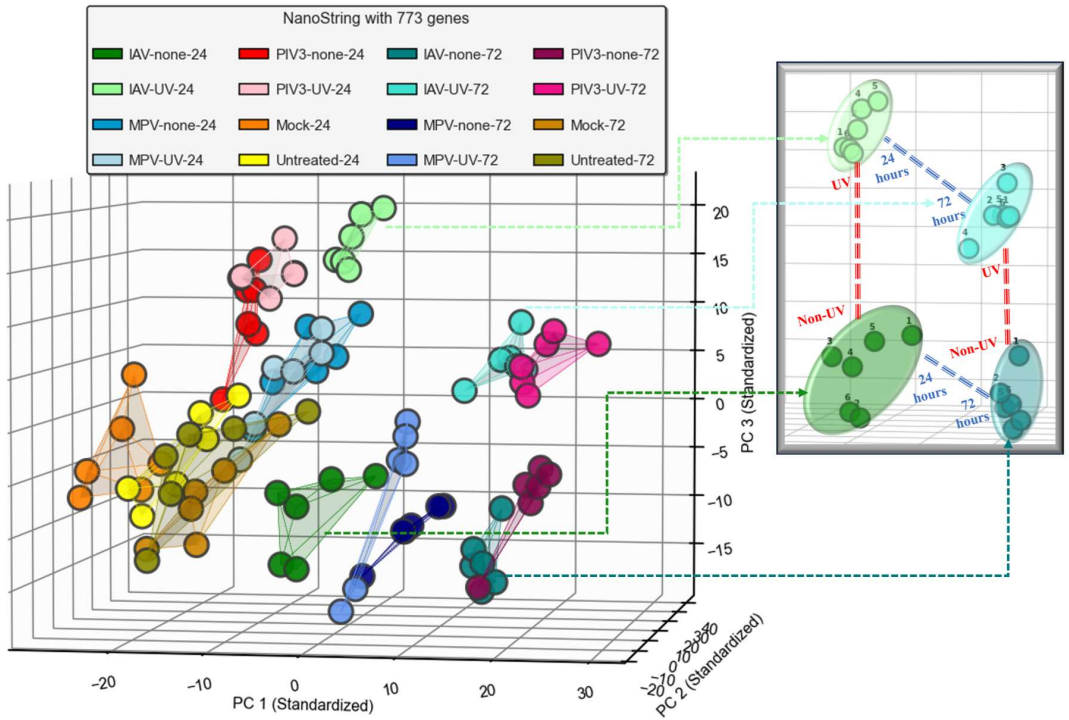
Definition: A type of machine learning where the model identifies patterns and structures in unlabeled data.

What is Unsupervised Learning, and how can it be applied in our research?

Why is it the First Step in Analyses?

Exploratory Data Analysis:

- Understand data structure and distribution.
- Identify hidden patterns or groupings.

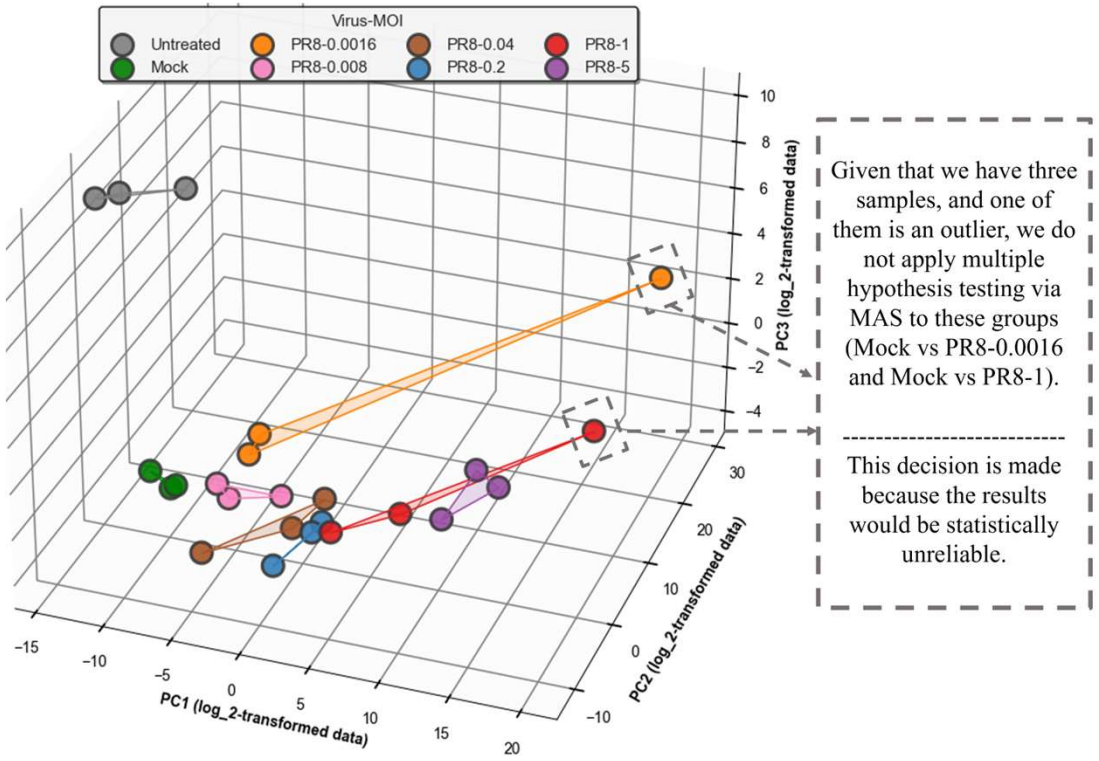


What is Unsupervised Learning, and how can it be applied in our research?

Why is it the First Step in Analyses?

Data Preprocessing:

- Detect outliers or anomalies.
- Reduce data dimensionality for efficiency.

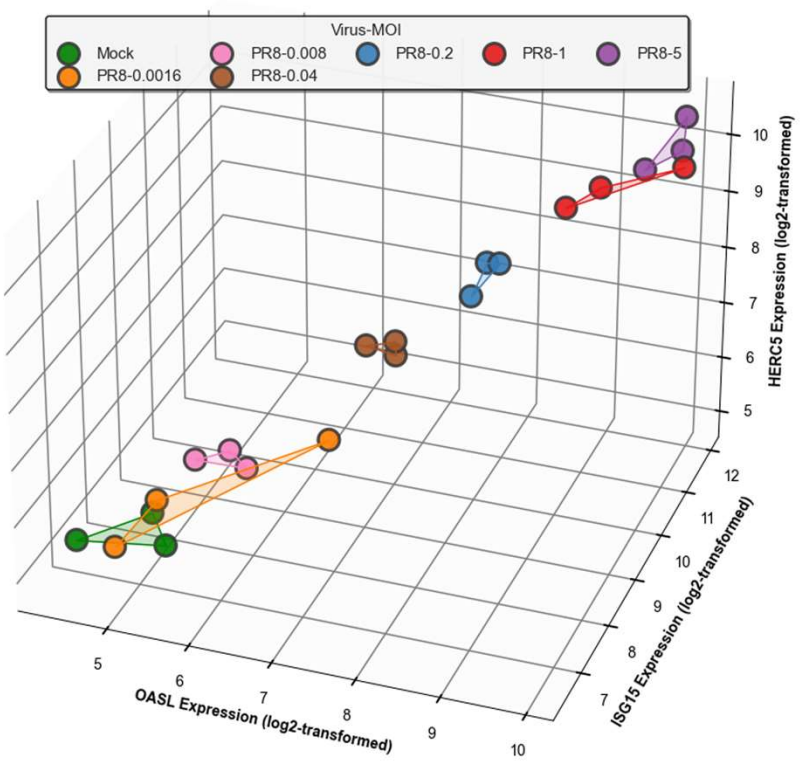


What is Unsupervised Learning, and how can it be applied in our research?

Why is it the First Step in Analyses?

Data Preprocessing:

- Detect outliers or anomalies.
- Reduce data dimensionality for efficiency.

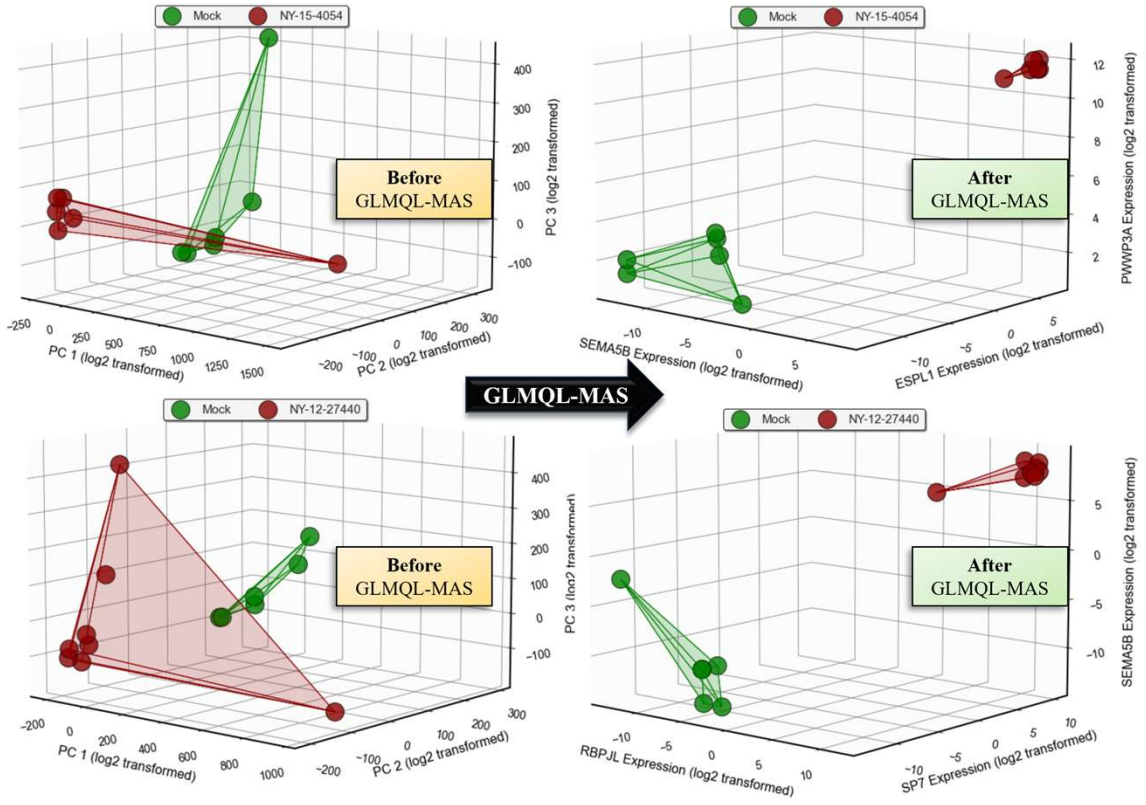


What is Unsupervised Learning, and how can it be applied in our research?

Why is it the First Step in Analyses?

Foundation for Supervised Models:

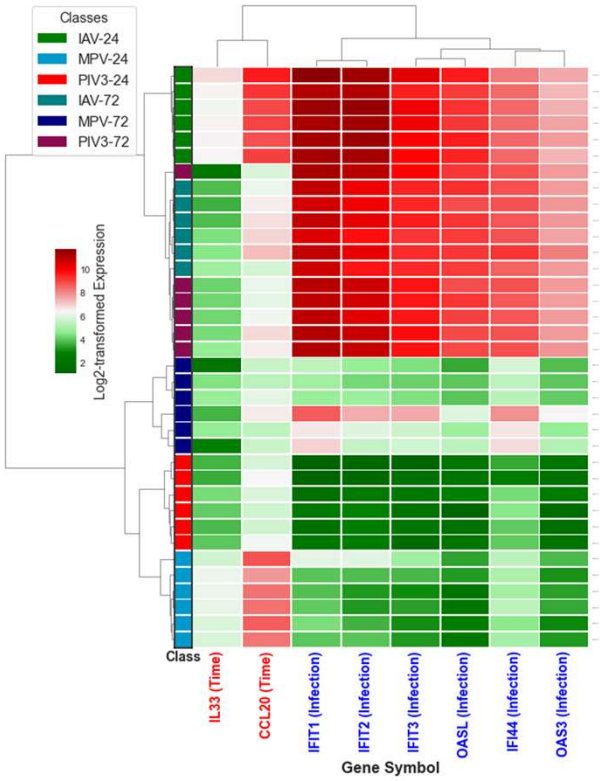
- Provides insights for feature engineering and labeled data preparation.



What is Unsupervised Learning, and how can it be applied in our research?

Why is it the First Step in Analyses?

- **Clustering Analysis:** Groups samples or features (e.g., patients, genes, metabolites) based on similarity, uncovering disease subtypes or co-regulated genes.

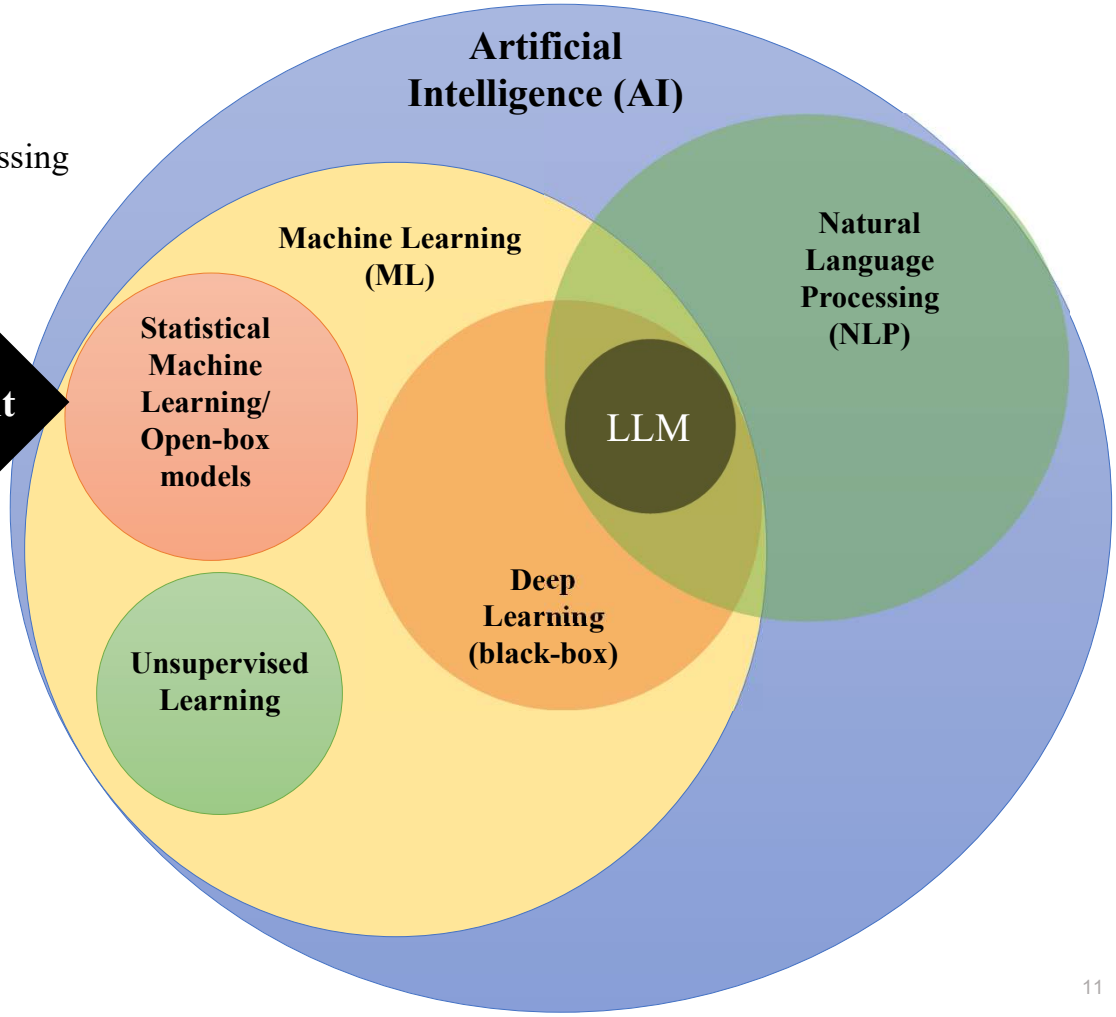


General overview of AI/ML models and their relevance to regenerative medicine

Artificial Intelligence (AI)

- Machine Learning (ML)
- Natural Language Processing
- Computer Vision
- Expert Systems

Now let's talk about



Machine Learning (ML)

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

What is Statistical Machine Learning, and how can it be applied in our research?

What is Statistical Machine Learning?

Statistical Machine Learning focuses on non-deep learning models that combine statistical principles with machine learning techniques to analyze data, identify patterns, and make predictions.

Characteristics:

- Relies on mathematical and statistical methods to make predictions.
- Models are interpretable and transparent (open-box).
- Examples include linear regression, decision trees, and support vector machines (SVM).

Pros:

- Requires fewer data points compared to deep learning models.
- Easier to interpret and validate, especially in biological contexts.
- Faster training and less computationally intensive.

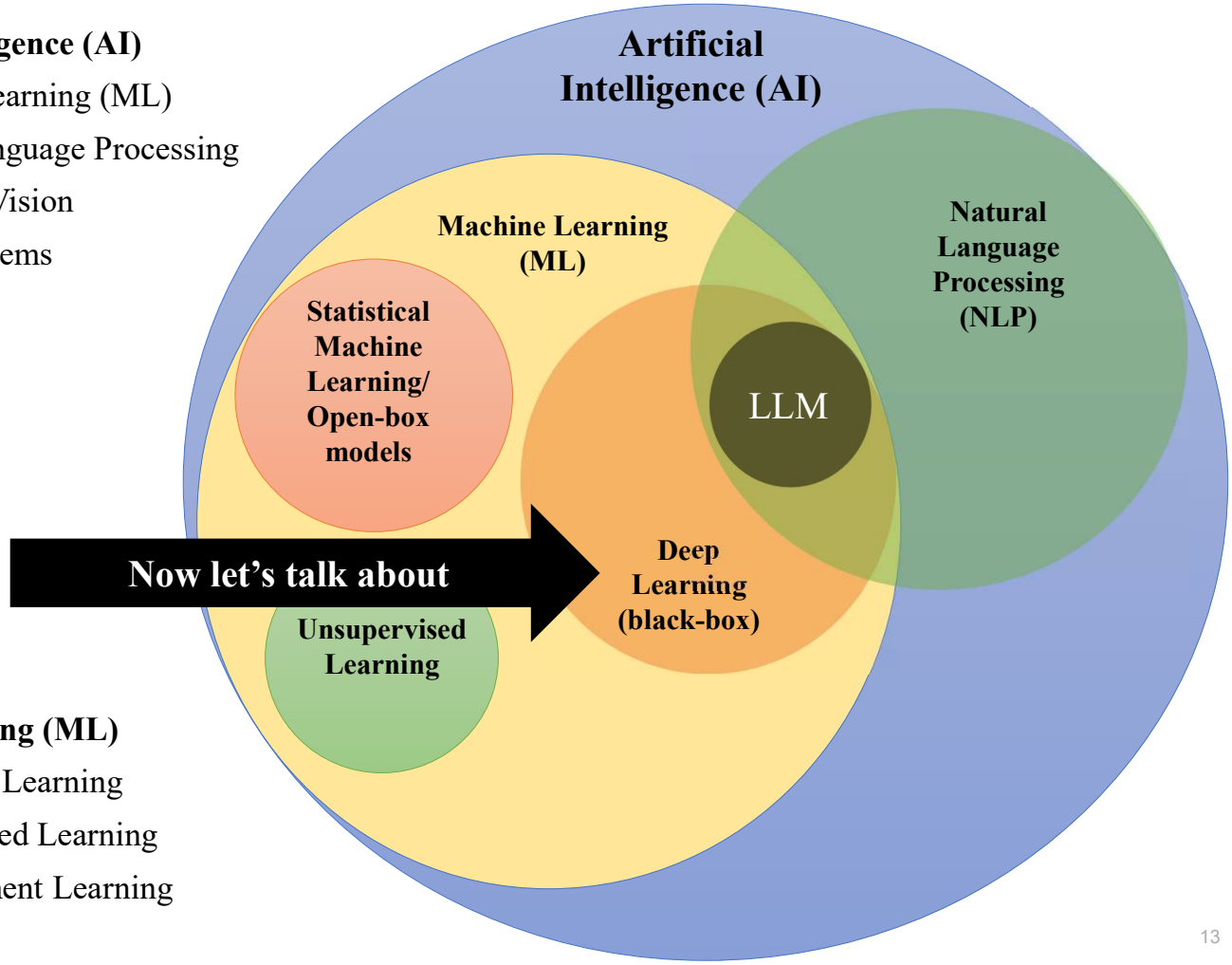
Cons:

- Limited performance with complex, unstructured data (e.g., images, large text datasets).
- May struggle with feature engineering and capturing non-linear relationship

General overview of AI/ML models and their relevance to regenerative medicine

Artificial Intelligence (AI)

- Machine Learning (ML)
- Natural Language Processing
- Computer Vision
- Expert Systems



Machine Learning (ML)

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

What is Deep Learning, and how can it be applied in our research?

Characteristics:

- Uses neural networks with multiple layers to model complex patterns and relationships in data.
- Often considered black-box models due to limited interpretability.
- Examples include convolutional neural networks (CNNs) and recurrent neural networks (RNNs).

Pros:

- Excels at processing large amounts of unstructured data (e.g., images, sequences).
- Can automatically identify features without manual feature engineering.
- State-of-the-art performance in fields like image analysis and natural language processing.

Cons:

- Requires substantial amounts of labeled data for training.
- Computationally expensive and time-intensive.
- Limited interpretability, which can be a challenge in fields requiring transparency.

What is Deep Learning, and how can it be applied in our research?

Comparison Table:

Feature	Statistical ML	Deep Learning
Data Requirement	Low to Moderate	High
Interpretability	High (Open-Box)	Low (Black-Box)
Performance on Complex Data	Moderate	High
Computational Demand	Low	High

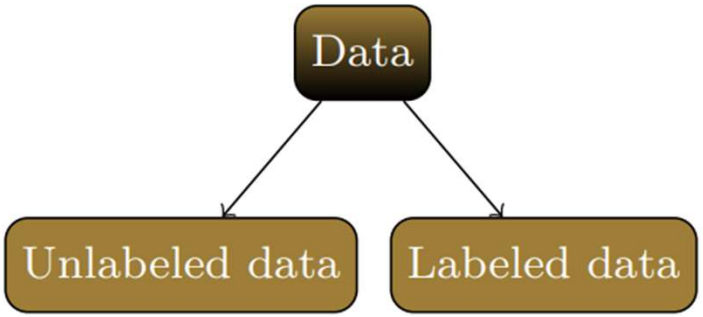
Minimum Sample-to-Feature Ratio:

Aim for at least 10 samples per feature after dimensionality reduction. For example, if you reduce your dataset to 500 genes via feature selection, aim for at least 5,000 samples.

General Benchmark:

A common starting point for deep learning is 500-1,000 samples per class, though this depends on the dataset's complexity.

Labeled vs Unlabeled Data



	Patient ID	Age	Blood Pressure	Cholesterol
0	6	45	130	Normal
1	7	33	122	Normal
2	8	52	138	High

	Patient ID	Age	Blood Pressure	Cholesterol	Diagnosis
0	1	35	120	Normal	Healthy
1	2	42	135	High	Diabetes
2	3	28	122	Normal	Healthy
3	4	57	140	High	Hypertension
4	5	60	128	Normal	Healthy

Labeled Data: Prediction

	Patient ID	Age	Blood Pressure	Cholesterol	BMI	Smoker	Family History	Exercise	Diagnosis
0	1	35	120	Normal	24.5	No	No	Moderate	Healthy
1	2	42	135	High	30.1	Yes	Yes	Low	Sick
2	3	28	122	Normal	21.7	No	No	High	Healthy
3	4	57	140	High	31.8	Yes	Yes	Low	Sick
4	5	60	128	Normal	26.4	No	Yes	Moderate	Healthy
5	6	45	130	High	29.8	Yes	No	Moderate	?

Labeled Data: Prediction

	Patient ID	Age	Blood Pressure	Cholesterol	BMI	Smoker	Family History	Exercise	Diagnosis
0	1	35	120	Normal	24.5	No	No	Moderate	Healthy
1	2	42	135	High	30.1	Yes	Yes	Low	Sick
2	3	28	122	Normal	21.7	No	No	High	Healthy
3	4	57	140	High	31.8	Yes	Yes	Low	Sick
4	5	60	128	Normal	26.4	No	Yes	Moderate	Healthy
5	6	45	130	High	29.8	Yes	No	Moderate	?

Let's learn from the rest of the labeled data with known labels and identify similarities

Labeled Data: Prediction

	Patient ID	Age	Blood Pressure	Cholesterol	BMI	Smoker	Family History	Exercise	Diagnosis
0	1	35	120	Normal	24.5	No	No	Moderate	Healthy
1	2	42	135	High	30.1	Yes	Yes	Low	Sick
2	3	28	122	Normal	21.7	No	No	High	Healthy
3	4	57	140	High	31.8	Yes	Yes	Low	Sick
4	5	60	128	Normal	26.4	No	Yes	Moderate	Healthy
5	6	45	130	High	29.8	Yes	No	Moderate	?

Let's learn from the rest of the labeled data with known labels and identify similarities

Labeled Data: Prediction

	Patient ID	Age	Blood Pressure	Cholesterol	BMI	Smoker	Family History	Exercise	Diagnosis
0	1	35	120	Normal	24.5	No	No	Moderate	Healthy
1	2	42	135	High	30.1	Yes	Yes	Low	Sick
2	3	28	122	Normal	21.7	No	No	High	Healthy
3	4	57	140	High	31.8	Yes	Yes	Low	Sick
4	5	60	128	Normal	26.4	No	Yes	Moderate	Healthy
5	6	45	130	High	29.8	Yes	No	Moderate	?

Let's learn from the rest of the labeled data with known labels and identify similarities

Labeled Data: Prediction

	Patient ID	Age	Blood Pressure	Cholesterol	BMI	Smoker	Family History	Exercise	Diagnosis
0	1	35	120	Normal	24.5	No	No	Moderate	Healthy
1	2	42	135	High	30.1	Yes	Yes	Low	Sick
2	3	28	122	Normal	21.7	No	No	High	Healthy
3	4	57	140	High	31.8	Yes	Yes	Low	Sick
4	5	60	128	Normal	26.4	No	Yes	Moderate	Healthy
5	6	45	130	High	29.8	Yes	No	Moderate	?

Let's learn from the rest of the labeled data with known labels and identify similarities

Labeled Data: Prediction

	Patient ID	Age	Blood Pressure	Cholesterol	BMI	Smoker	Family History	Exercise	Diagnosis
0	1	35	120	Normal	24.5	No	No	Moderate	Healthy
1	2	42	135	High	30.1	Yes	Yes	Low	Sick
2	3	28	122	Normal	21.7	No	No	High	Healthy
3	4	57	140	High	31.8	Yes	Yes	Low	Sick
4	5	60	128	Normal	26.4	No	Yes	Moderate	Healthy
5	6	45	130	High	29.8	Yes	No	Moderate	?

Wake Forest University
School of Medicine

CAIR
CENTER FOR ARTIFICIAL INTELLIGENCE RESEARCH

Labeled Data: Prediction

?

	Patient ID	Age	Blood Pressure	Cholesterol	BMI	Smoker	Family History	Exercise	Diagnosis
0	1	35	120	Normal	24.5	No	No	Moderate	Healthy
1	2	42	135	High	30.1	Yes	Yes	Low	Sick
2	3	28	122	Normal	21.7	No	No	High	Healthy
3	4	57	140	High	31.8	Yes	Yes	Low	Sick
4	5	60	128	Normal	26.4	No	Yes	Moderate	Healthy
5	6	45	130	High	29.8	Yes	No	Moderate	?

Training

	Patient ID	Age	Blood Pressure	Cholesterol	BMI	Smoker	Family History	Exercise	Diagnosis
0	1	35	120	Normal	24.5	No	No	Moderate	Healthy
1	2	42	135	High	30.1	Yes	Yes	Low	Sick
2	3	28	122	Normal	21.7	No	No	High	Healthy
3	4	57	140	High	31.8	Yes	Yes	Low	Sick
4	5	60	128	Normal	26.4	No	Yes	Moderate	Healthy
5	6	45	130	High	29.8	Yes	No	Moderate	?

```python
import pandas as pd

# Labeled Data Example
labeled_data = pd.DataFrame({
    'Patient ID': [1, 2, 3, 4, 5],
    'Age': [35, 42, 28, 57, 60],
    'Blood Pressure': [120, 135, 122, 140, 128],
    'Cholesterol': ['Normal', 'High', 'Normal', 'High', 'Normal'],
    'BMI': [24.5, 30.1, 21.7, 31.8, 26.4],
    'Smoker': ['No', 'Yes', 'No', 'Yes', 'No'],
    'Family History': ['No', 'Yes', 'No', 'Yes', 'Yes'],
    'Exercise': ['Moderate', 'Low', 'High', 'Low', 'Moderate'],
    'Diagnosis': ['Healthy', 'Sick', 'Healthy', 'Sick', 'Healthy']
})

# Unlabeled Data Example
unlabeled_data = pd.DataFrame({
    'Patient ID': [6],
    'Age': [45],
    'Blood Pressure': [130],
    'Cholesterol': ['High'],
    'BMI': [29.8],
    'Smoker': ['Yes'],
    'Family History': ['No'],
    'Exercise': ['Moderate'],
    'Diagnosis': ['?']
})

# Concatenate the labeled and unlabeled data
all_data = pd.concat([labeled_data, unlabeled_data], ignore_index=True)

print("All Data:")
all_data
```

Wake Forest University
School of Medicine

CAIR
CENTER FOR ARTIFICIAL INTELLIGENCE RESEARCH

In [51]:
```
X = all_data.drop('Diagnosis', axis=1)
y = all_data['Diagnosis']
```

In [52]:
```
X
```

Out[52]:

	Patient ID	Age	Blood Pressure	Cholesterol	BMI	Smoker	Family History	Exercise
0	1	35	120	Normal	24.5	No	No	Moderate
1	2	42	135	High	30.1	Yes	Yes	Low
2	3	28	122	Normal	21.7	No	No	High
3	4	57	140	High	31.8	Yes	Yes	Low
4	5	60	128	Normal	26.4	No	Yes	Moderate
5	6	45	130	High	29.8	Yes	No	Moderate

In [53]:
```
y
```

Out[53]:
```
0    Healthy
1       Sick
2    Healthy
3       Sick
4    Healthy
5          ?
Name: Diagnosis, dtype: object
```

```
In [37]: X
```

Out[37]:

	Patient ID	Age	Blood Pressure	Cholesterol	BMI	Smoker	Family History	Exercise
0	1	35	120	Normal	24.5	No	No	Moderate
1	2	42	135	High	30.1	Yes	Yes	Low
2	3	28	122	Normal	21.7	No	No	High
3	4	57	140	High	31.8	Yes	Yes	Low
4	5	60	128	Normal	26.4	No	Yes	Moderate
5	6	45	130	High	29.8	Yes	No	Moderate

```
X_encoded = pd.get_dummies(X)
X_encoded
```

Patient ID	Age	Blood Pressure	BMI	Cholesterol_High	Cholesterol_Normal	Smoker_No	Smoker_Yes	Family History_No	Family History_Yes	Exercise_High	Exercise_Low	Exercise_Moderate
1	35	120	24.5	0	1	1	0	1	0	0	0	1
2	42	135	30.1	1	0	0	1	0	1	0	1	0
3	28	122	21.7	0	1	1	0	1	0	1	0	0
4	57	140	31.8	1	0	0	1	0	1	0	1	0
5	60	128	26.4	0	1	1	0	0	1	0	0	1
6	45	130	29.8	1	0	0	1	1	0	0	0	1

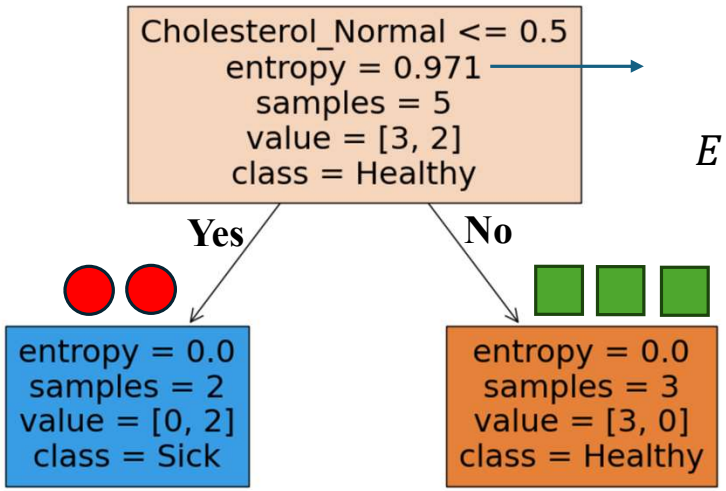
```
In [37]: X
```

Out[37]:

	Patient ID	Age	Blood Pressure	Cholesterol	BMI	Smoker	Family History	Exercise
0	1	35	120	Normal	24.5	No	No	Moderate
1	2	42	135	High	30.1	Yes	Yes	Low
2	3	28	122	Normal	21.7	No	No	High
3	4	57	140	High	31.8	Yes	Yes	Low
4	5	60	128	Normal	26.4	No	Yes	Moderate
5	6	45	130	High	29.8	Yes	No	Moderate

Patient ID	Age	Blood Pressure	BMI	Cholesterol_High	Cholesterol_Normal	Smoker_No	Smoker_Yes	Family History_No	Family History_Yes	Exercise_High	Exercise_Low	Exercise_Moderate
1	35	120	24.5	0	1	1	0	1	0	0	0	1
2	42	135	30.1	1	0	0	1	0	1	0	1	0
3	28	122	21.7	0	1	1	0	1	0	1	0	0
4	57	140	31.8	1	0	0	1	0	1	0	1	0
5	60	128	26.4	0	1	1	0	0	1	0	0	1
6	45	130	29.8	1	0	0	1	1	0	0	0	1

Patient ID	Age	Blood Pressure	BMI	Cholesterol_High	Cholesterol_Normal	Smoker_No	Smoker_Yes	Family History_No	Family History_Yes	Exercise_High	Exercise_Low	Exercise_Moderate	
1	35	120	24.5	0	1	1	0	1	0	0	0	1	
2	42	135	30.1	1	0	0	1	0	1	0	1	0	
3	28	122	21.7	0	1	1	0	1	0	1	0	0	
4	57	140	31.8	1	0	0	1	0	1	0	1	0	
5	60	128	26.4	0	1	1	0	0	1	0	0	1	



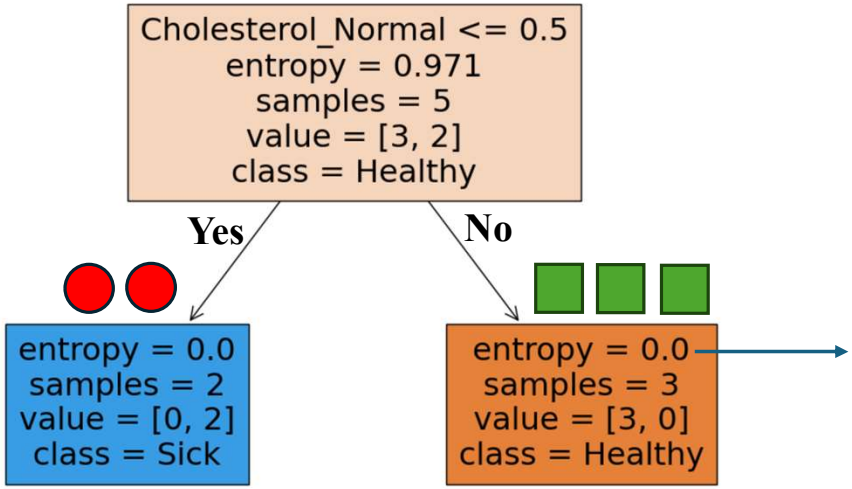
$$E(S) = -\sum_{i=1}^{n} p_i \log p_i,$$

$$E(S) = -(\frac{3}{5}\log\left(\frac{3}{5}\right) + \frac{2}{5}\log\left(\frac{2}{5}\right)) = 0.971$$

Wake Forest University
School of Medicine

CAIR
CENTER FOR ARTIFICIAL INTELLIGENCE RESEARCH

Patient ID	Age	Blood Pressure	BMI	Cholesterol_High	Cholesterol_Normal	Smoker_No	Smoker_Yes	Family History_No	Family History_Yes	Exercise_High	Exercise_Low	Exercise_Moderate
1	35	120	24.5	0	1	1	0	1	0	0	0	1
2	42	135	30.1	1	0	0	1	0	1	0	1	0
3	28	122	21.7	0	1	1	0	1	0	1	0	0
4	57	140	31.8	1	0	0	1	0	1	0	1	0
5	60	128	26.4	0	1	1	0	0	1	0	0	1



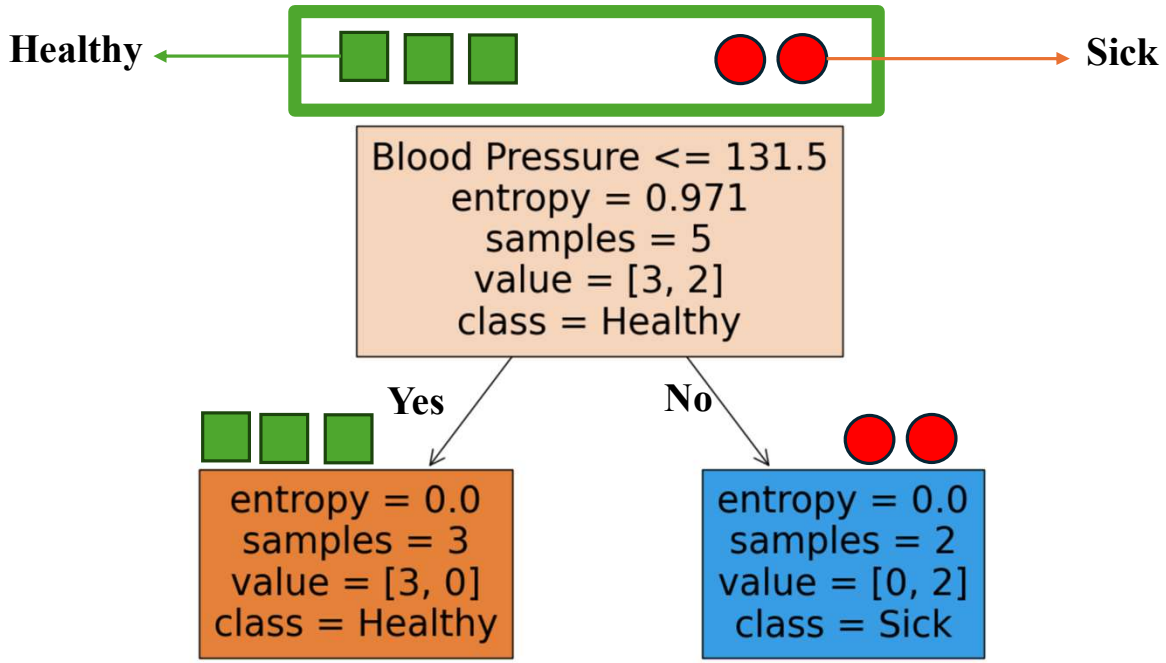
$$E(S) = -\sum_{i=1}^{n} p_i \log p_i,$$

$$E(S) = -(\frac{3}{3}\log\left(\frac{3}{3}\right)) = 0$$

Wake Forest University
School of Medicine

CAIR
CENTER FOR ARTIFICIAL INTELLIGENCE RESEARCH

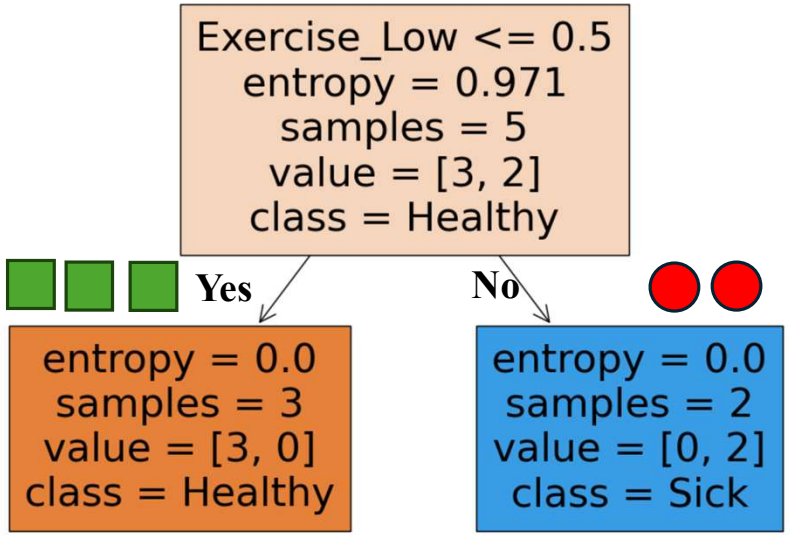
Patient ID	Age	Blood Pressure	BMI	Cholesterol_High	Cholesterol_Normal	Smoker_No	Smoker_Yes	Family History_No	Family History_Yes	Exercise_High	Exercise_Low	Exercise_Moderate
1	35	120	24.5	0	1	1	0	1	0	0	0	1
2	42	135	30.1	1	0	0	1	0	1	0	1	0
3	28	122	21.7	0	1	1	0	1	0	1	0	0
4	57	140	31.8	1	0	0	1	0	1	0	1	0
5	60	128	26.4	0	1	1	0	0	1	0	0	1



Wake Forest University
School of Medicine

CAIR
CENTER FOR ARTIFICIAL INTELLIGENCE RESEARCH

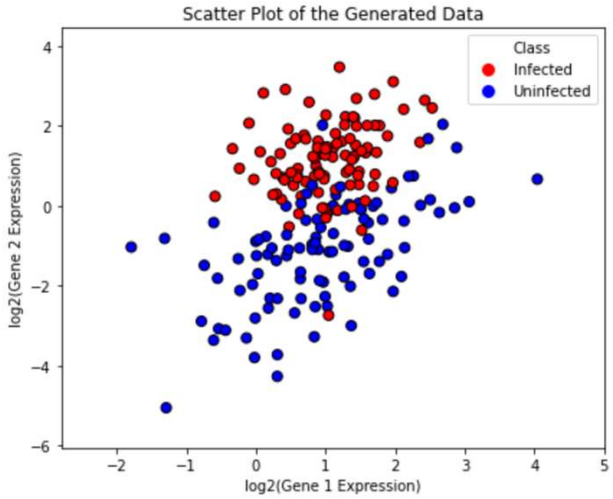
Patient ID	Age	Blood Pressure	BMI	Cholesterol_High	Cholesterol_Normal	Smoker_No	Smoker_Yes	Family History_No	Family History_Yes	Exercise_High	Exercise_Low	Exercise_Moderate
1	35	120	24.5	0	1	1	0	1	0	0	0	1
2	42	135	30.1	1	0	0	1	0	1	0	1	0
3	28	122	21.7	0	1	1	0	1	0	1	0	0
4	57	140	31.8	1	0	0	1	0	1	0	1	0
5	60	128	26.4	0	1	1	0	0	1	0	0	1



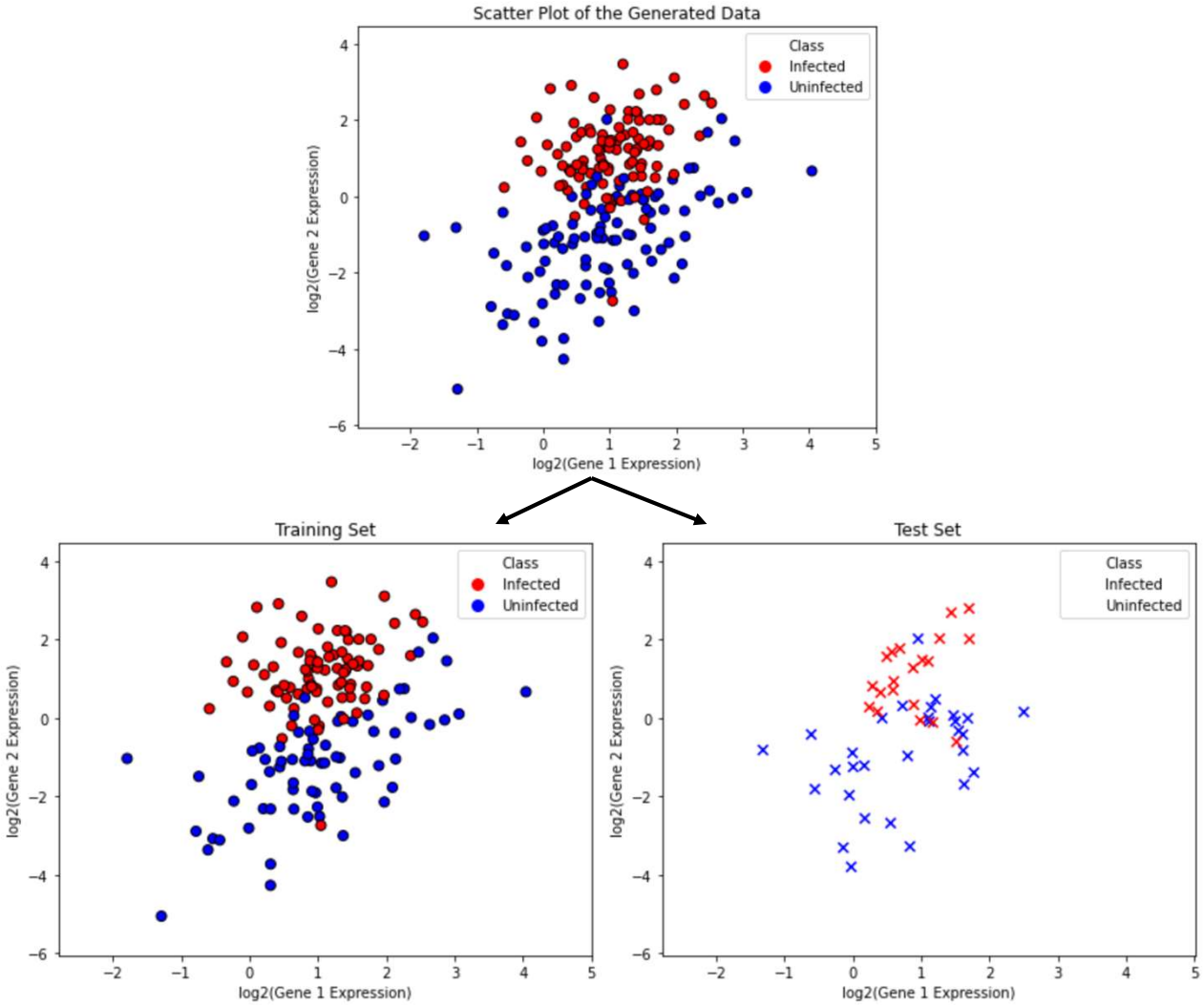
Wake Forest University
School of Medicine

CAIR
CENTER FOR ARTIFICIAL INTELLIGENCE RESEARCH

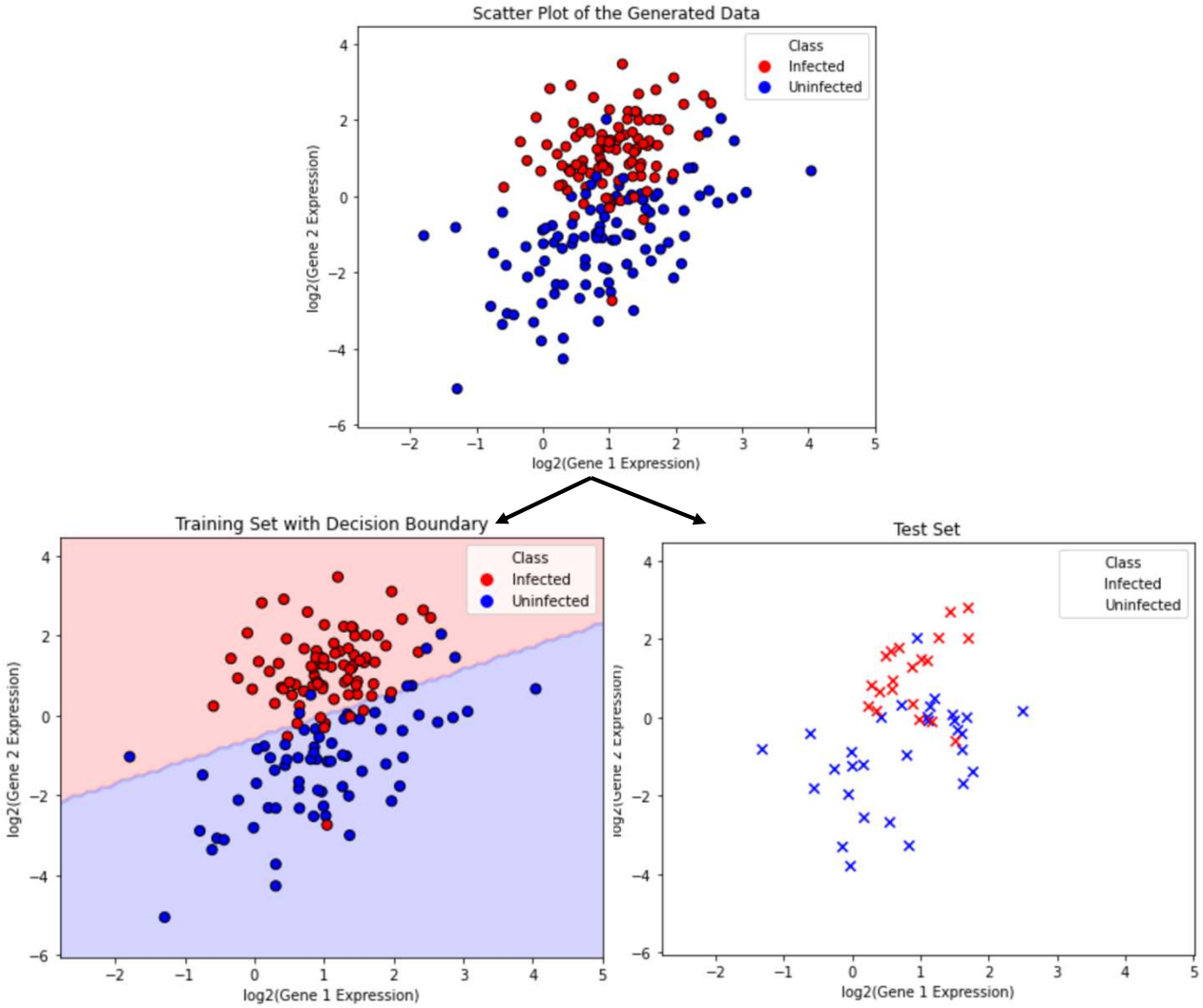
Training-test split:



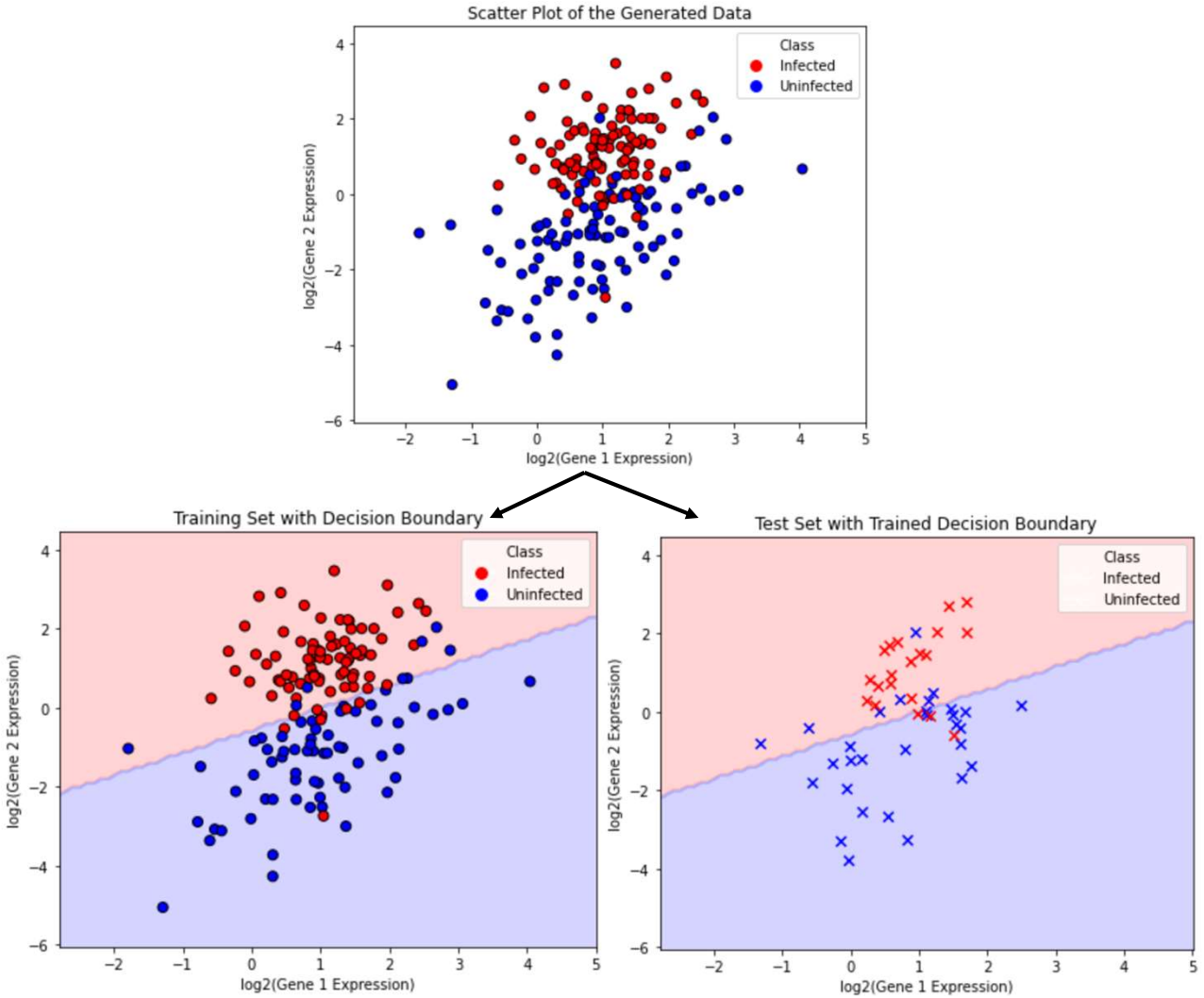
Training-test split:



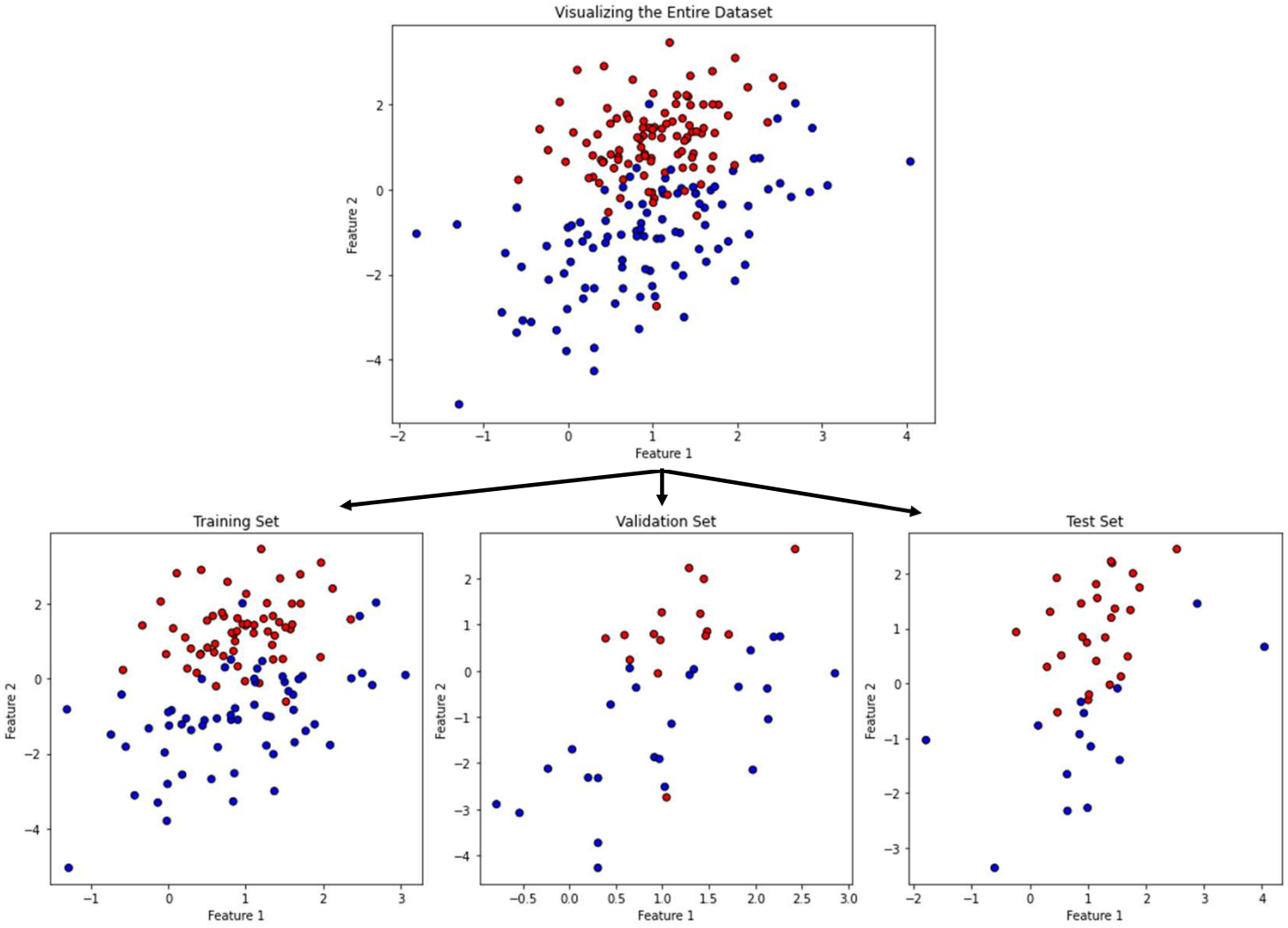
Training-test split:



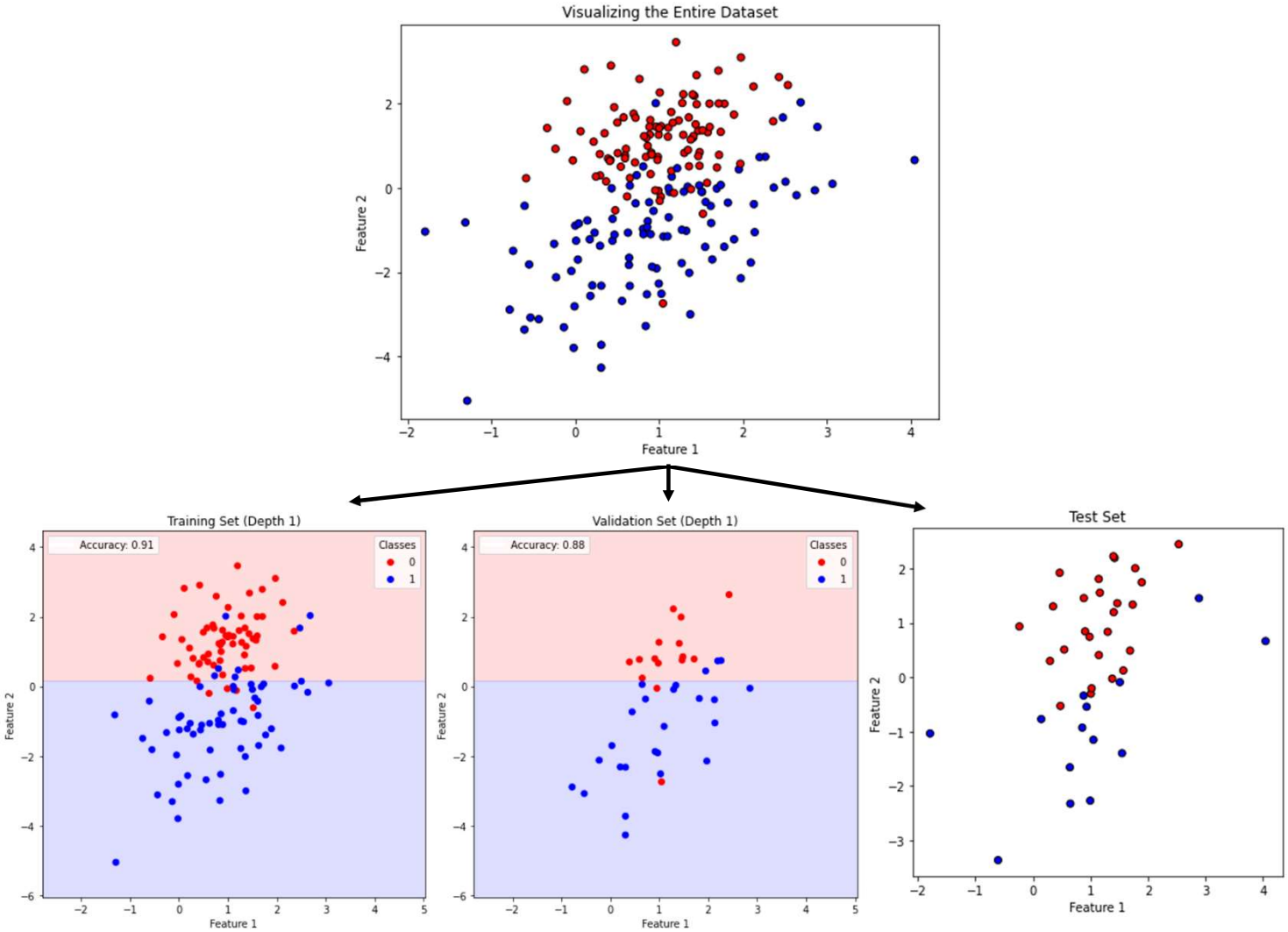
Training-test split:



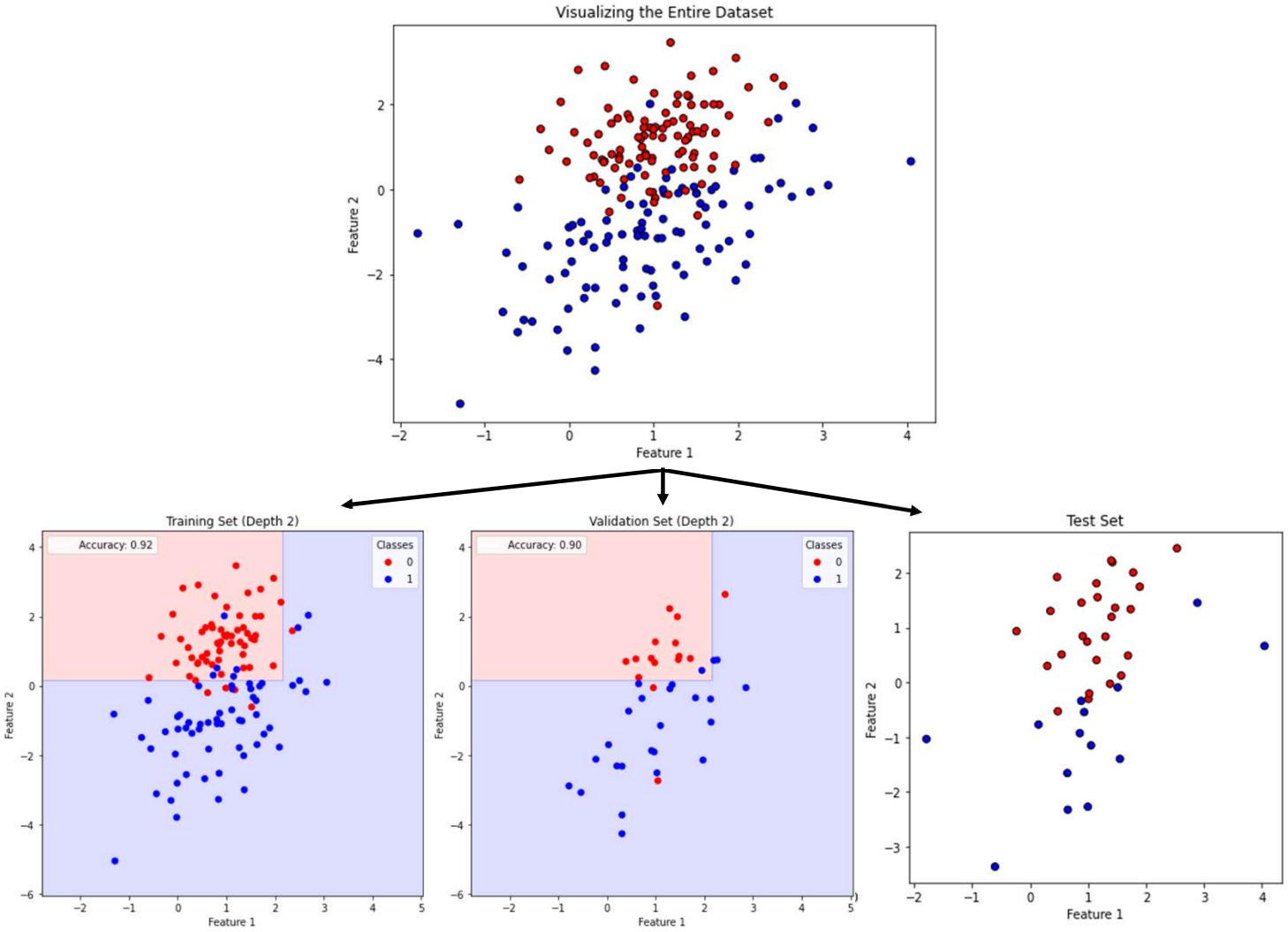
Training-validation-test split:



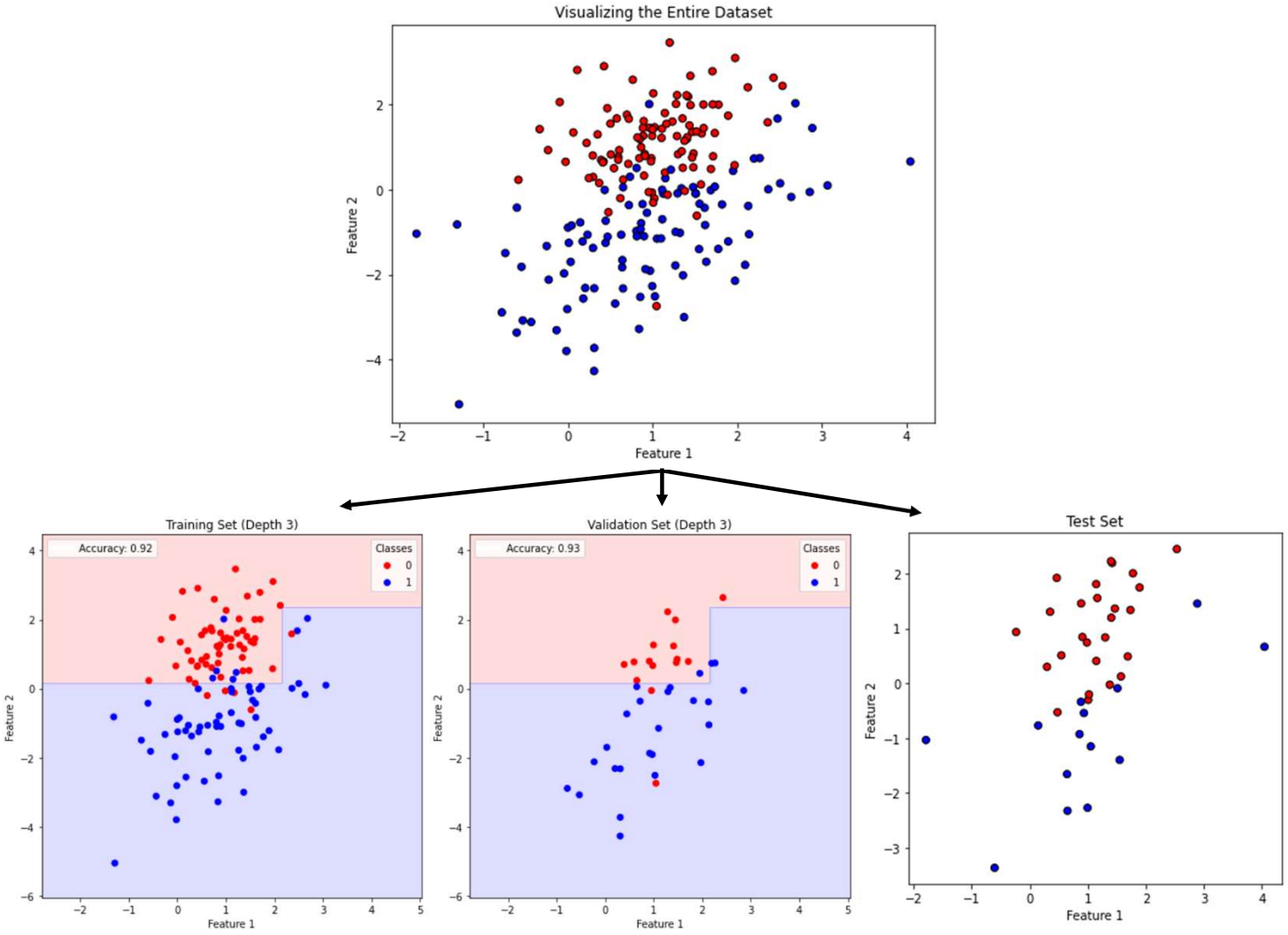
Training-validation-test split:



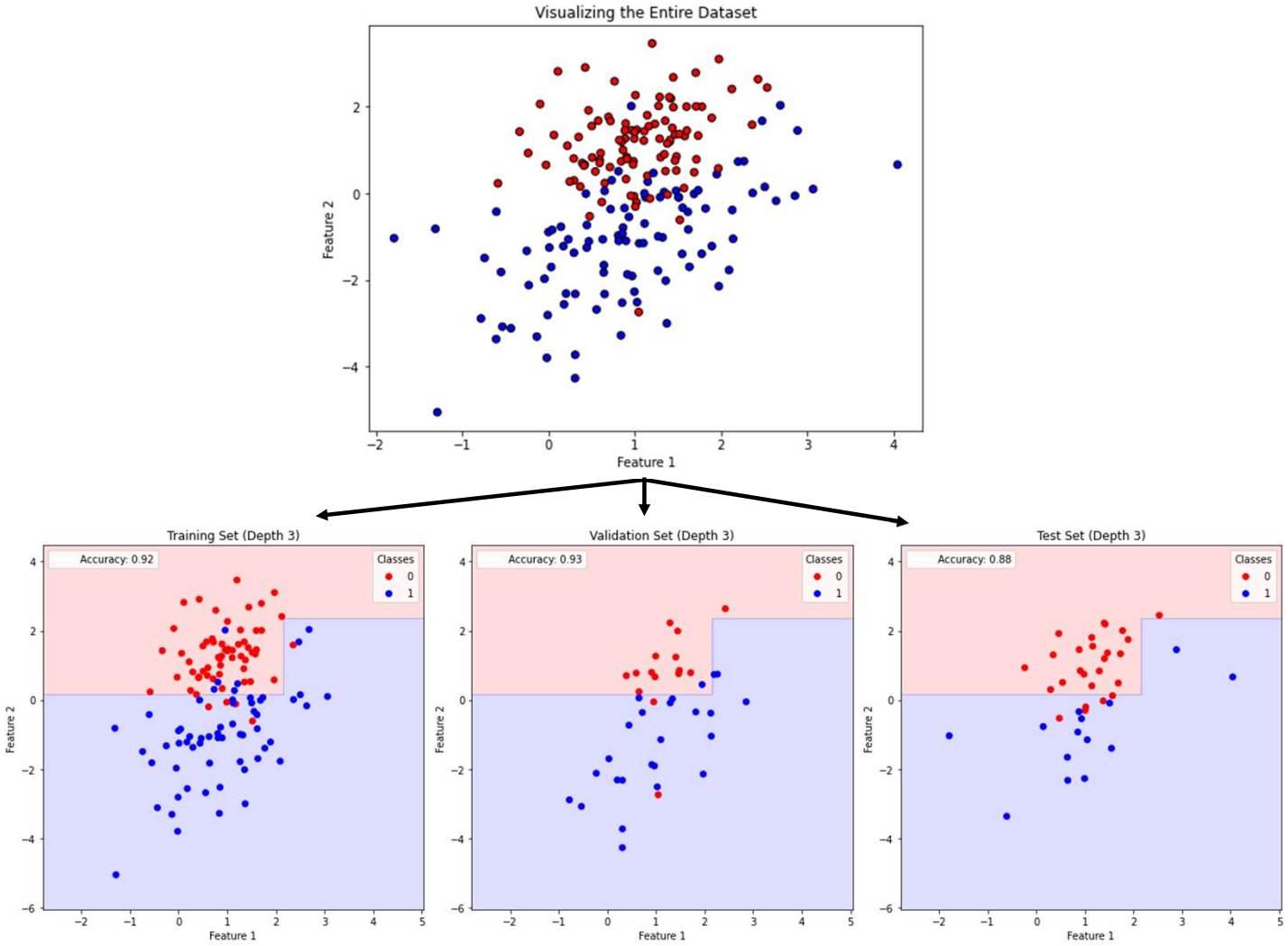
Training-validation-test split:

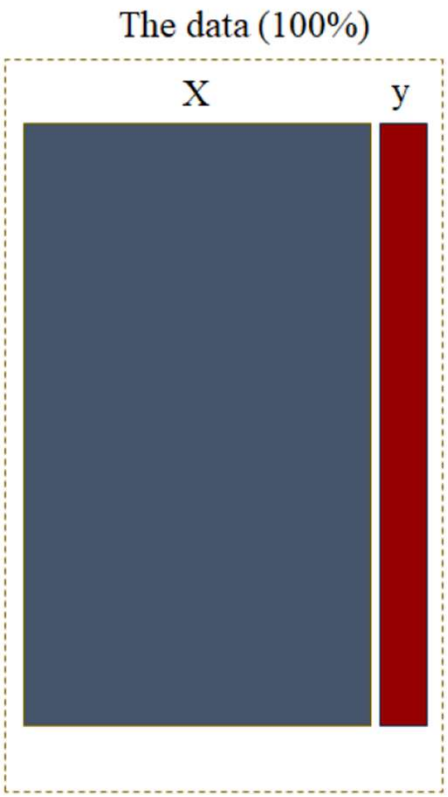


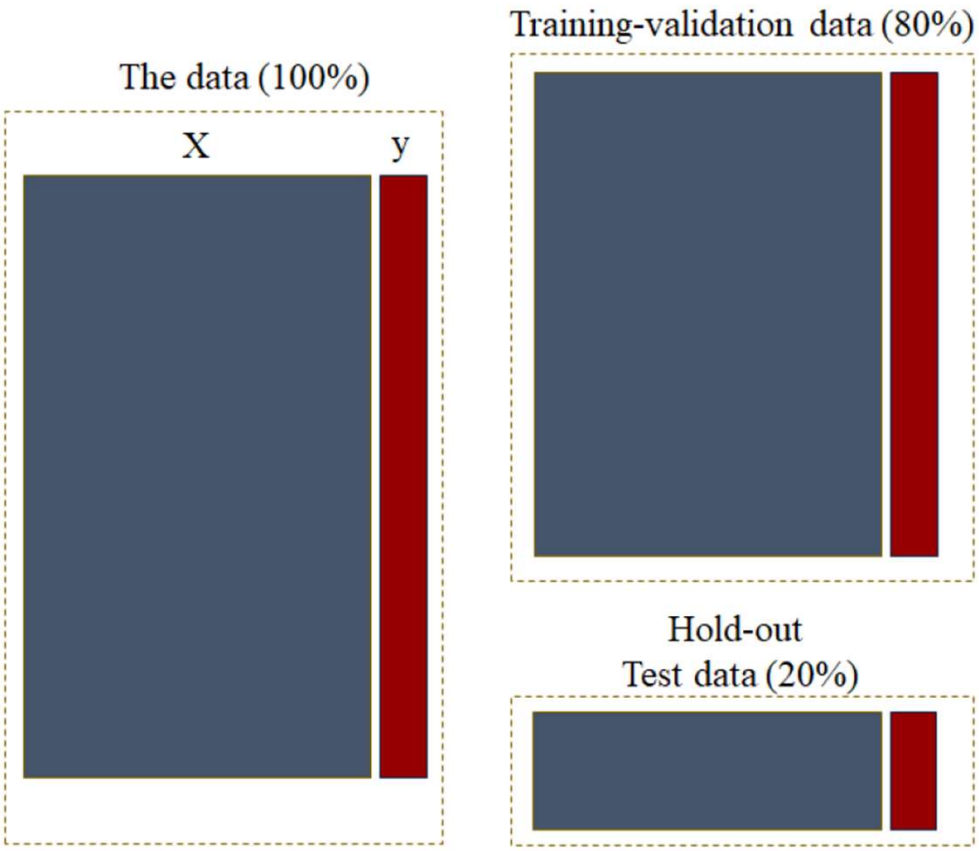
Training-validation-test split:

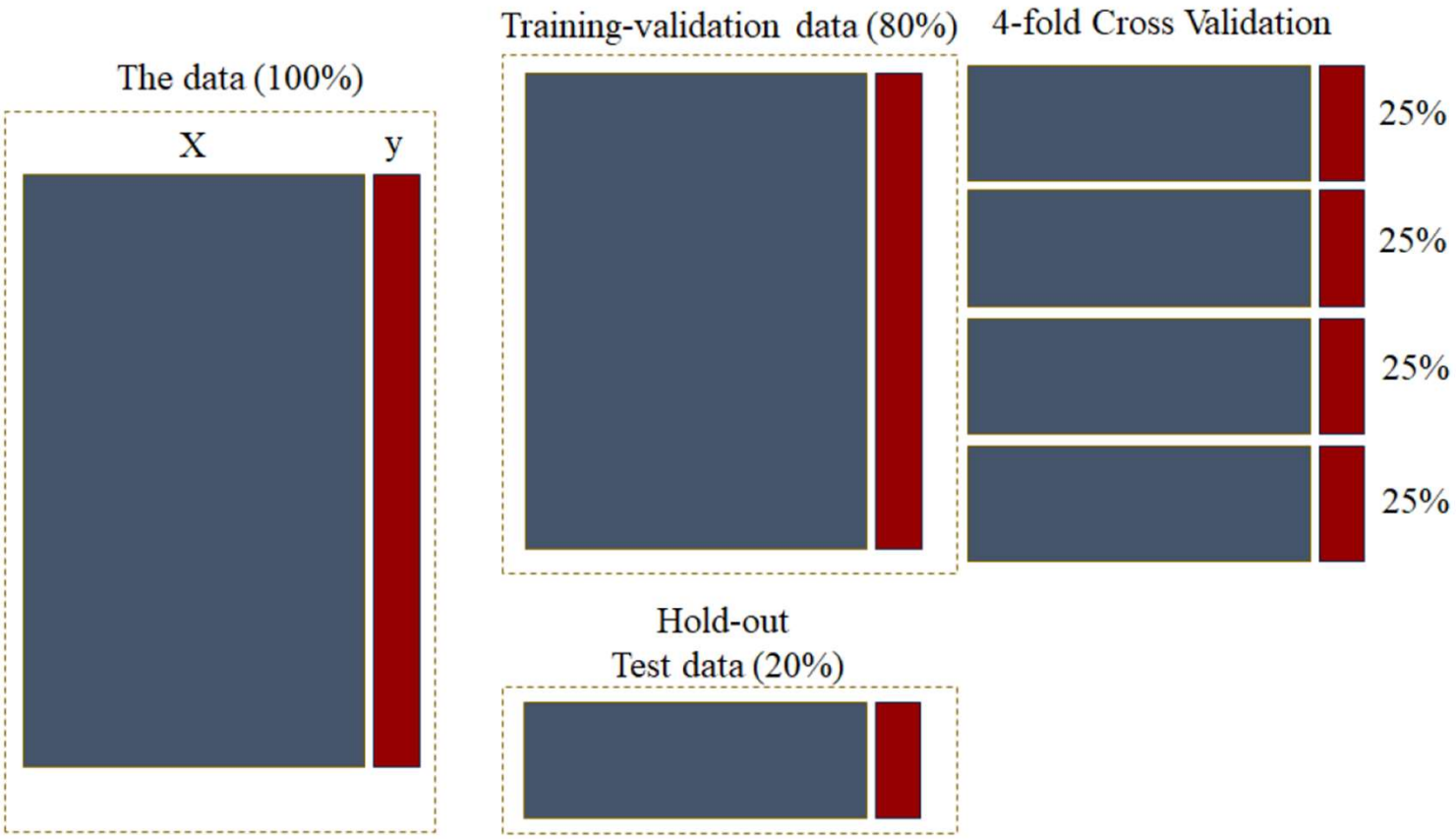


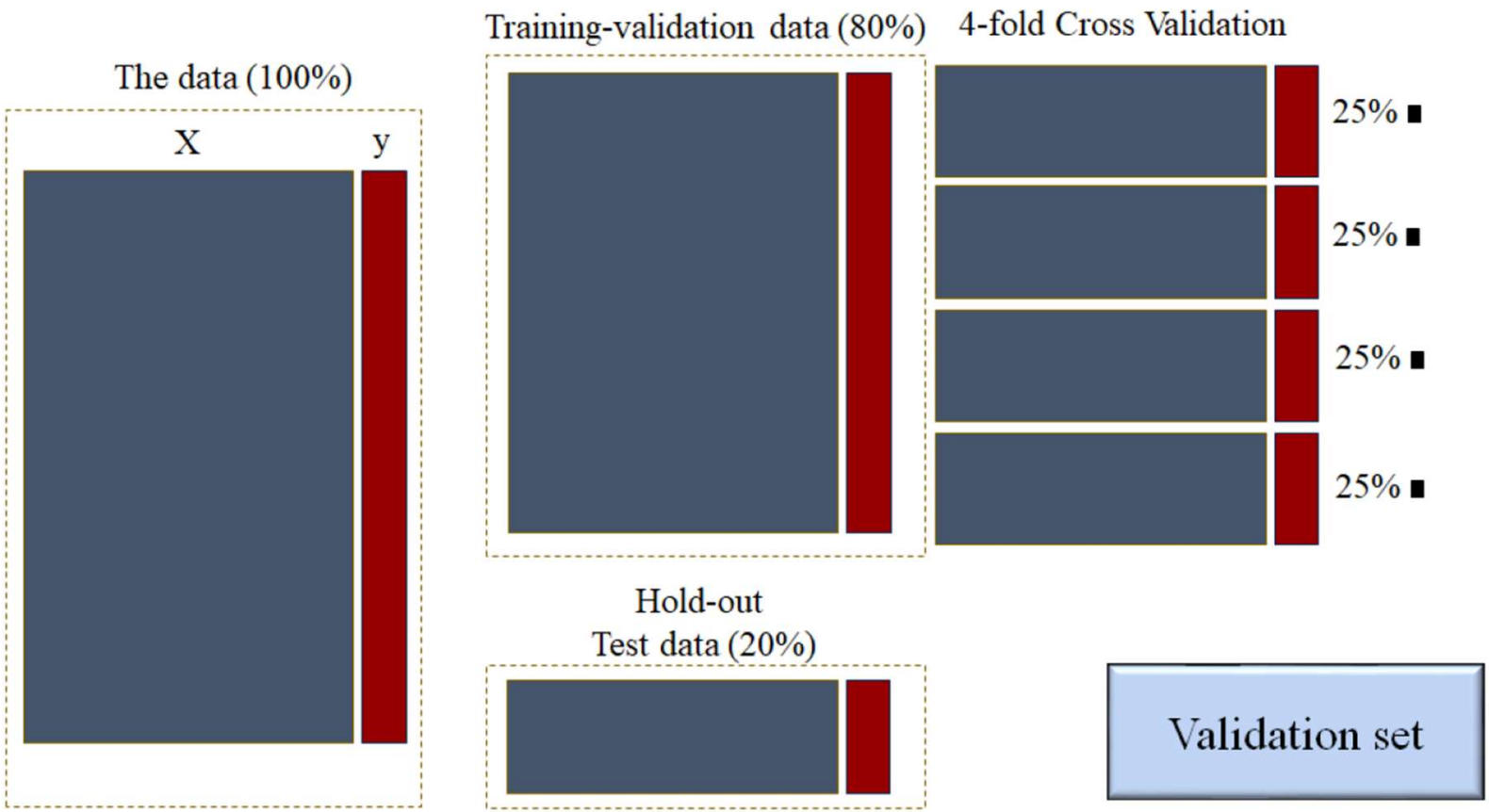
Training-validation-test split:




The data (100%)
X
y

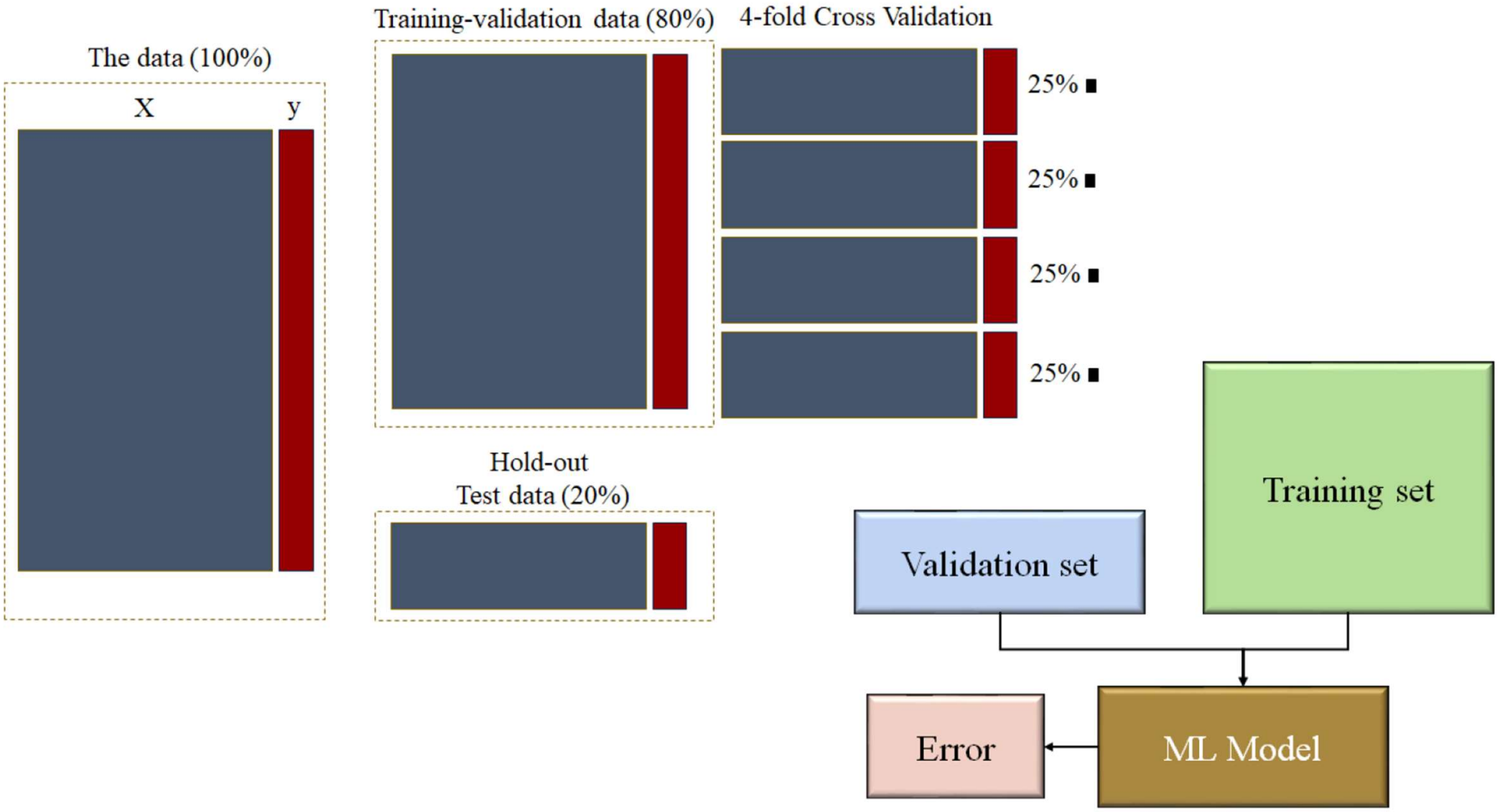

The data (100%)
X
y
Training-validation data (80%)
Hold-out
Test data (20%)

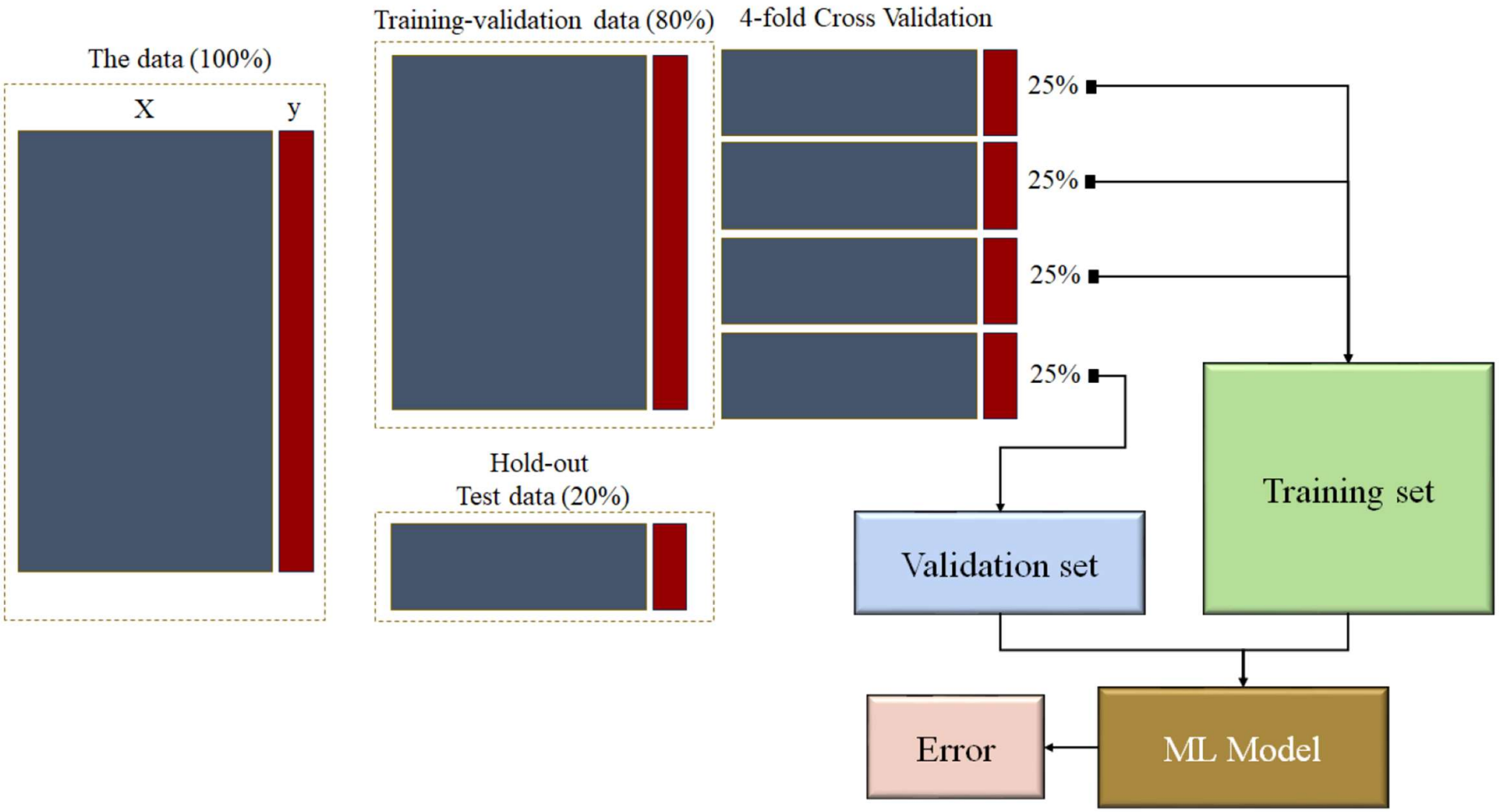

The data (100%)
X
y
Training-validation data (80%)
Hold-out
Test data (20%)
4-fold Cross Validation
25%
25%
25%
25%

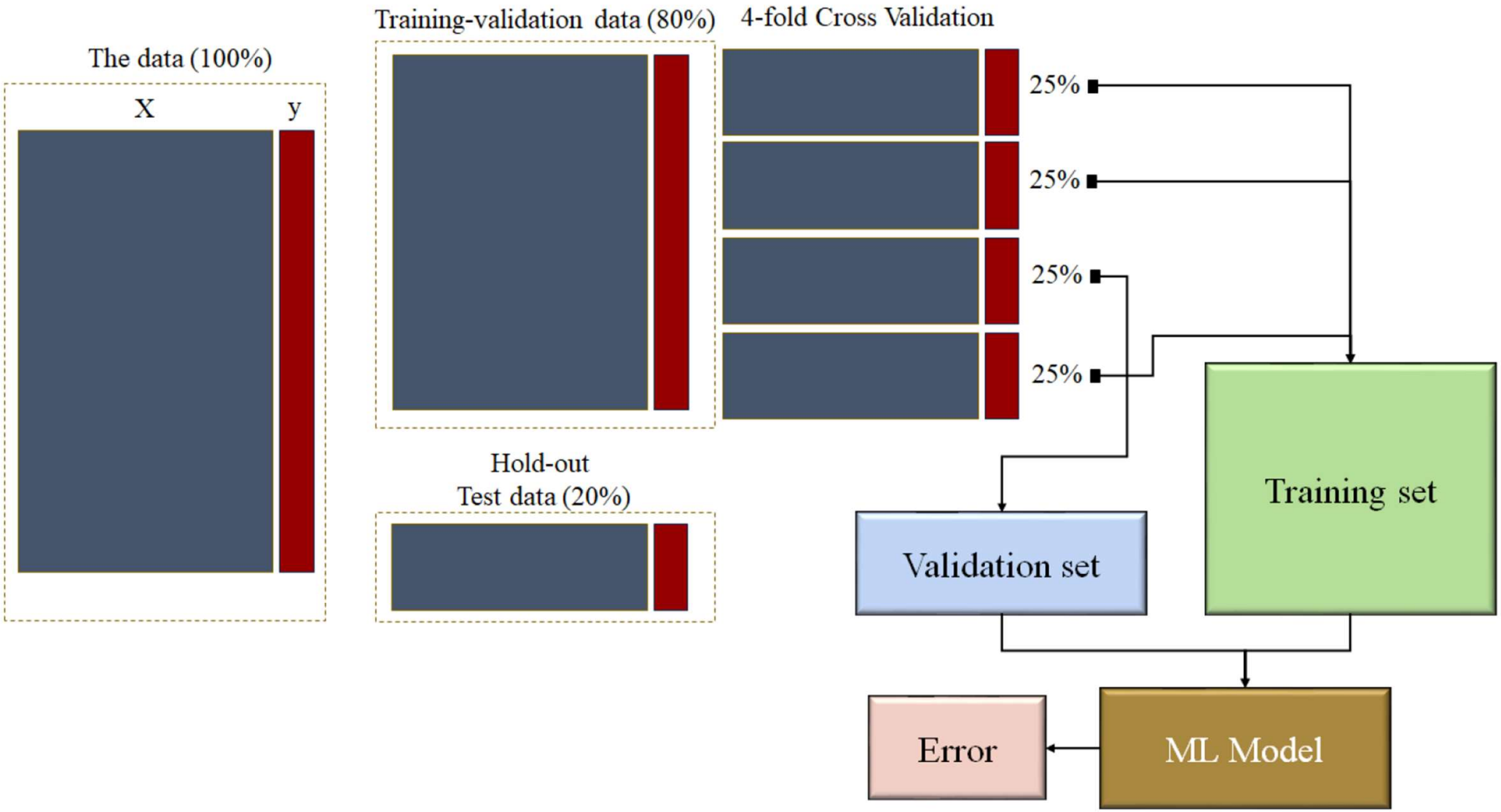


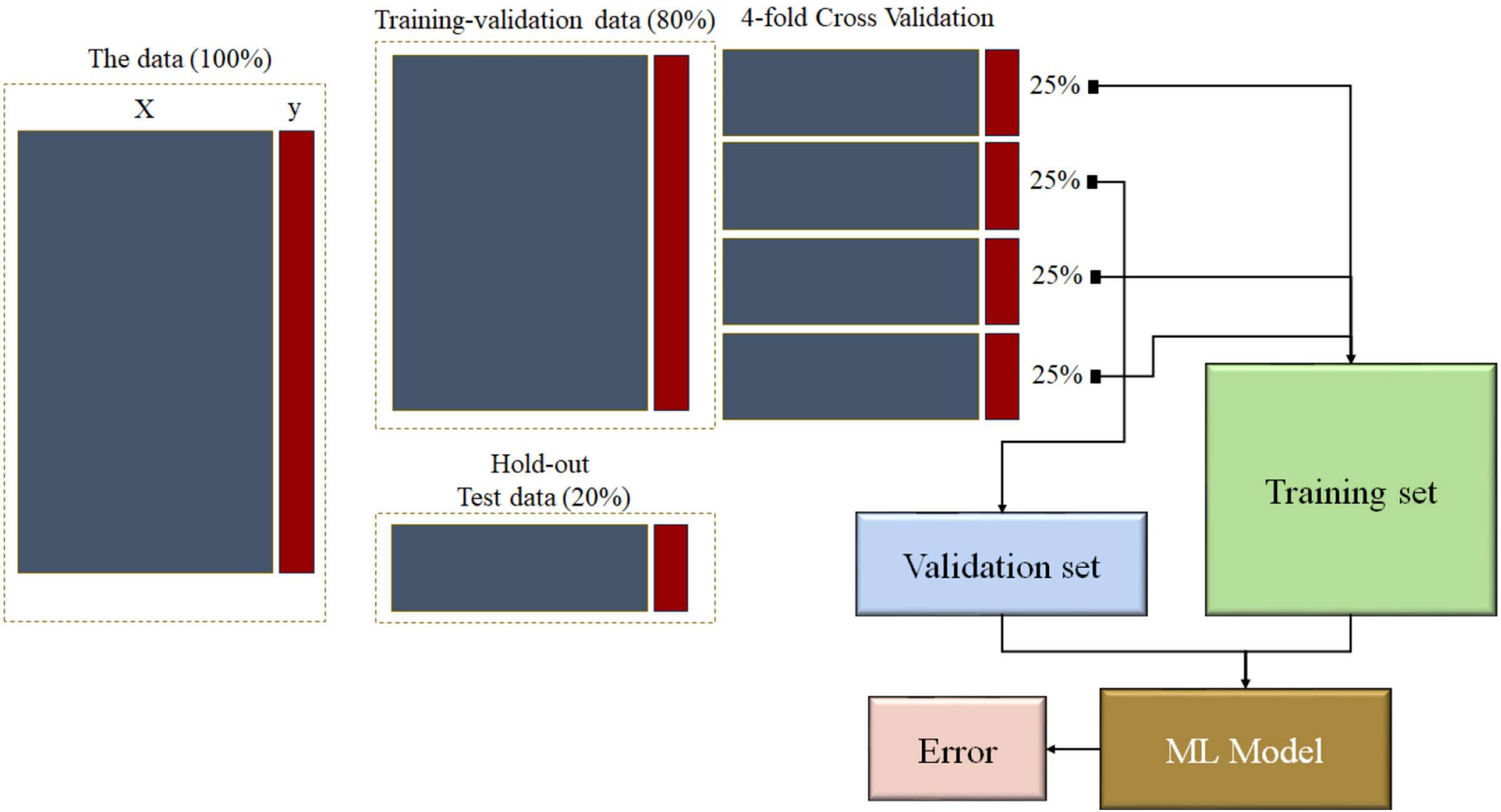
Wake Forest University
School of Medicine

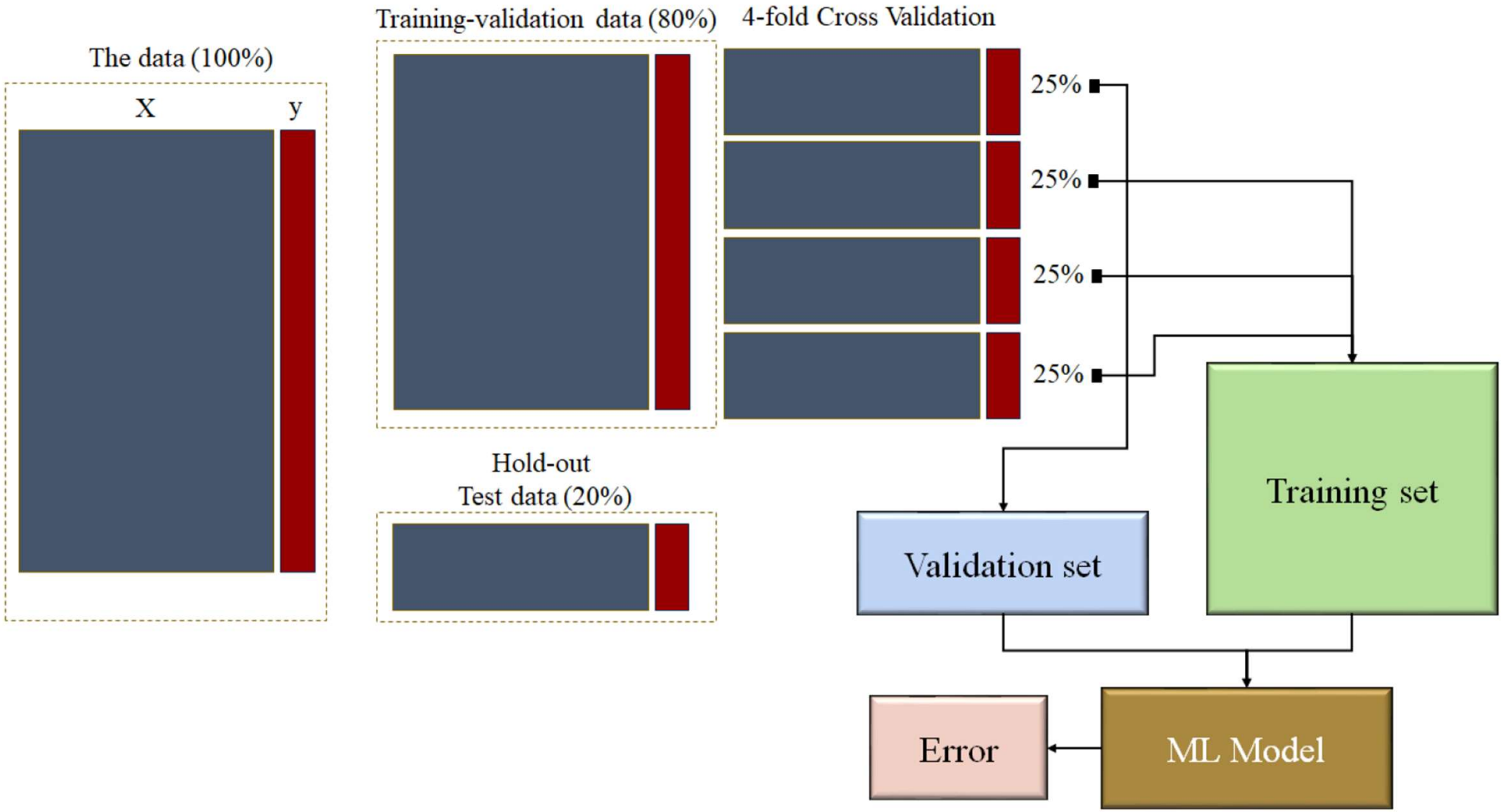
CAIR
CENTER FOR ARTIFICIAL INTELLIGENCE RESEARCH


The data (100%)
X
y
Training-validation data (80%)
4-fold Cross Validation
25%
25%
25%
25%
Hold-out
Test data (20%)
Training set
Validation set
Error
ML Model


The data (100%)
X
y
Training-validation data (80%)
Hold-out
Test data (20%)
4-fold Cross Validation
25%
25%
25%
25%
Training set
Validation set
Error
ML Model


The data (100%)
X
y
Training-validation data (80%)
4-fold Cross Validation
25%
25%
25%
25%
Hold-out
Test data (20%)
Validation set
Training set
Error
ML Model


The data (100%)
X
y
Training-validation data (80%)
4-fold Cross Validation
25%
25%
25%
25%
Hold-out
Test data (20%)
Training set
Validation set
Error
ML Model


The data (100%)
X
y
Training-validation data (80%)
4-fold Cross Validation
25%
25%
25%
25%
Hold-out
Test data (20%)
Validation set
Training set
Error
ML Model

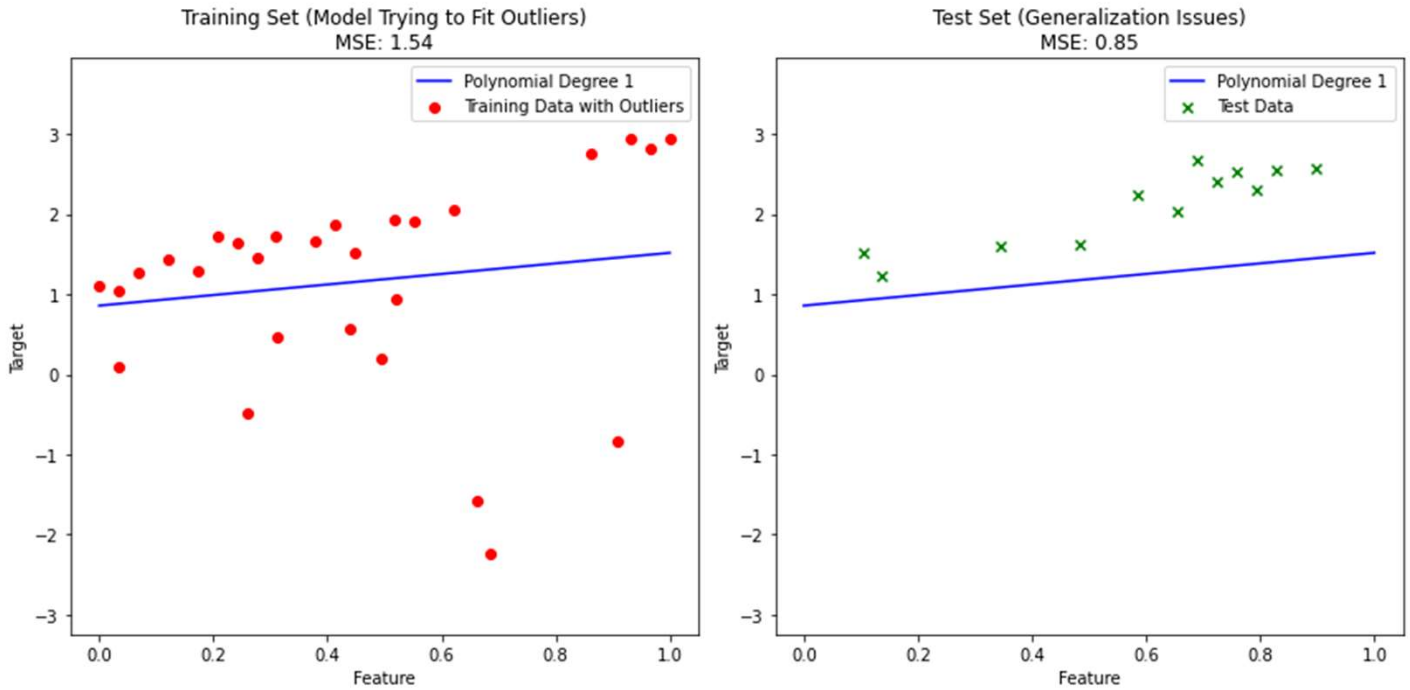

Hold-out
Test data (20%)
ML Model
Test Performance

Overfitting:

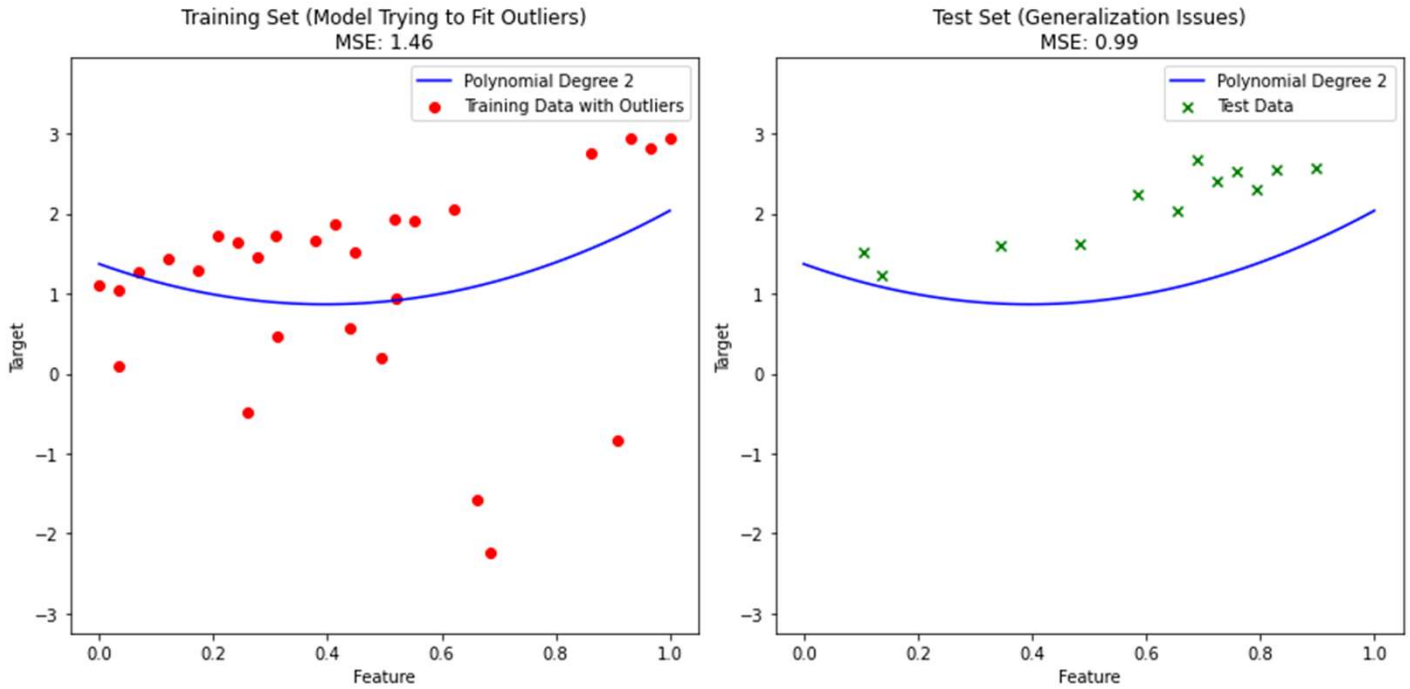
- Overfitting occurs when using overly complex models that exceed the necessary parameters for the given data.

- Overfitting can also happen when there are too many variables relative to the number of samples.

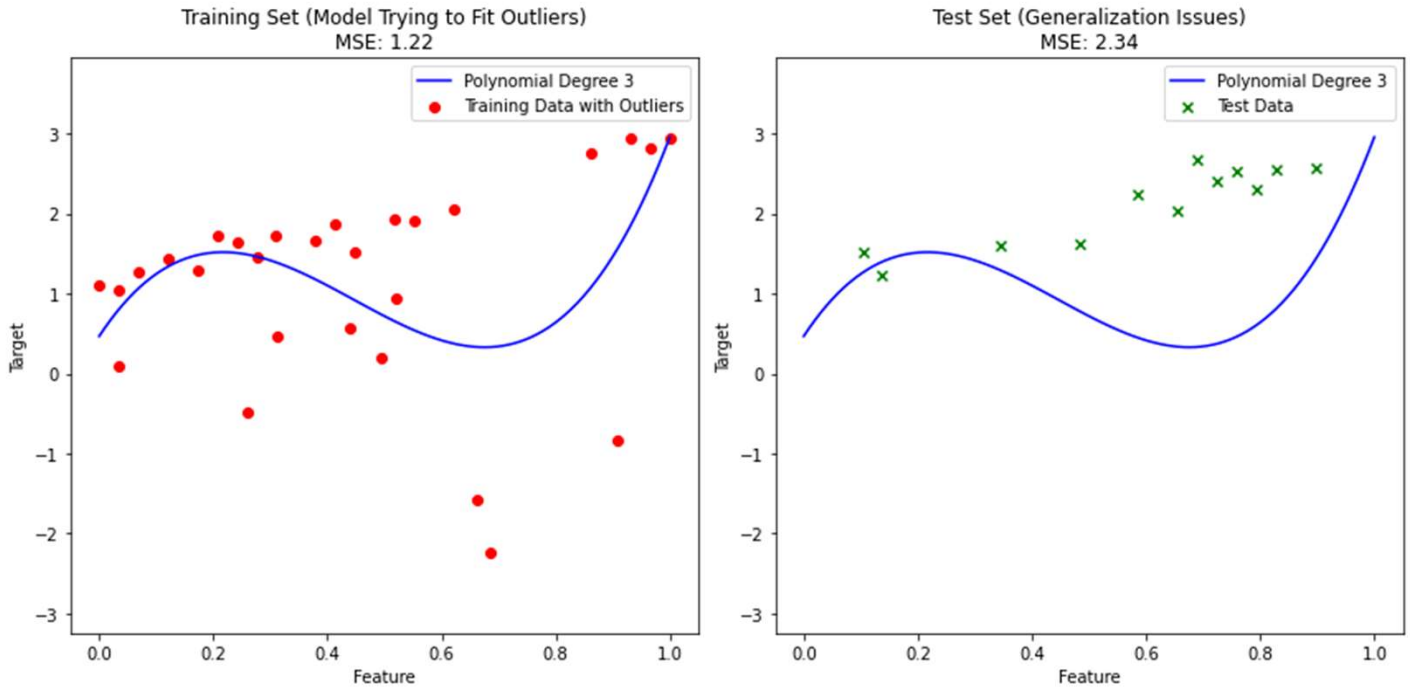
Overfitting: Complex models



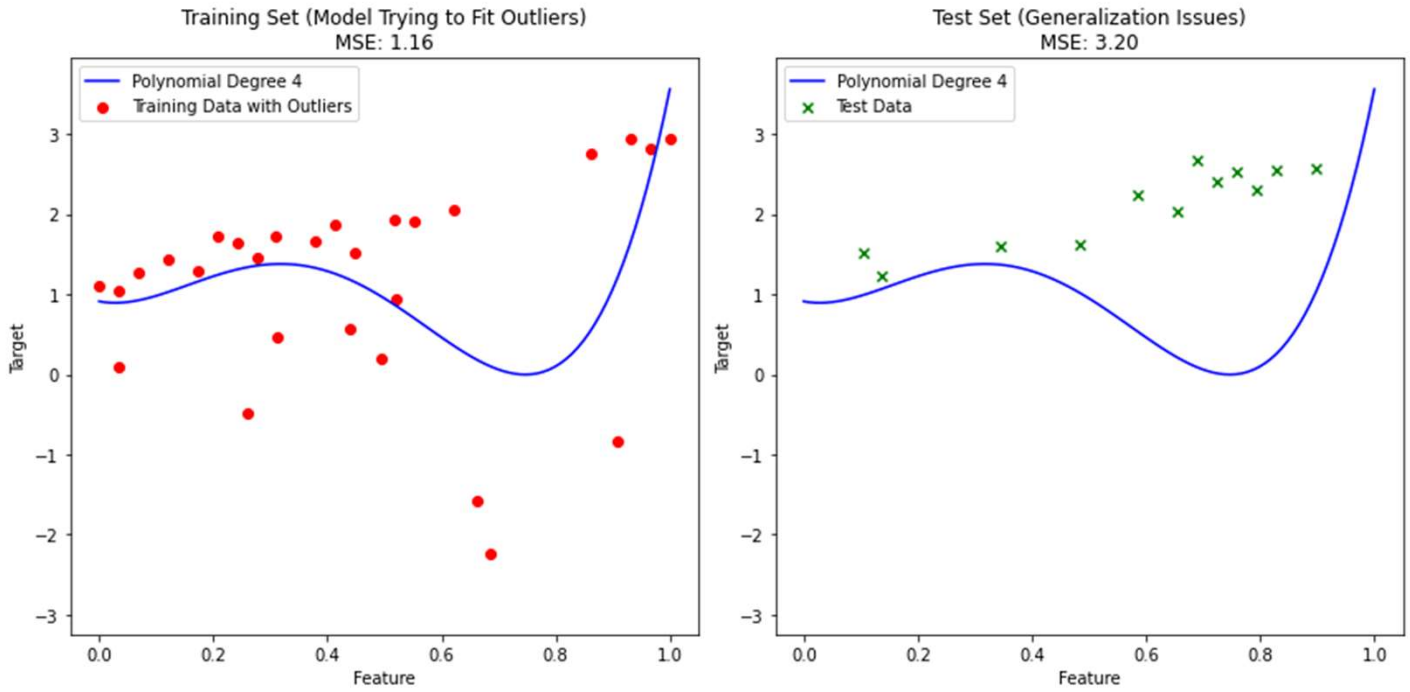
Overfitting: Complex models



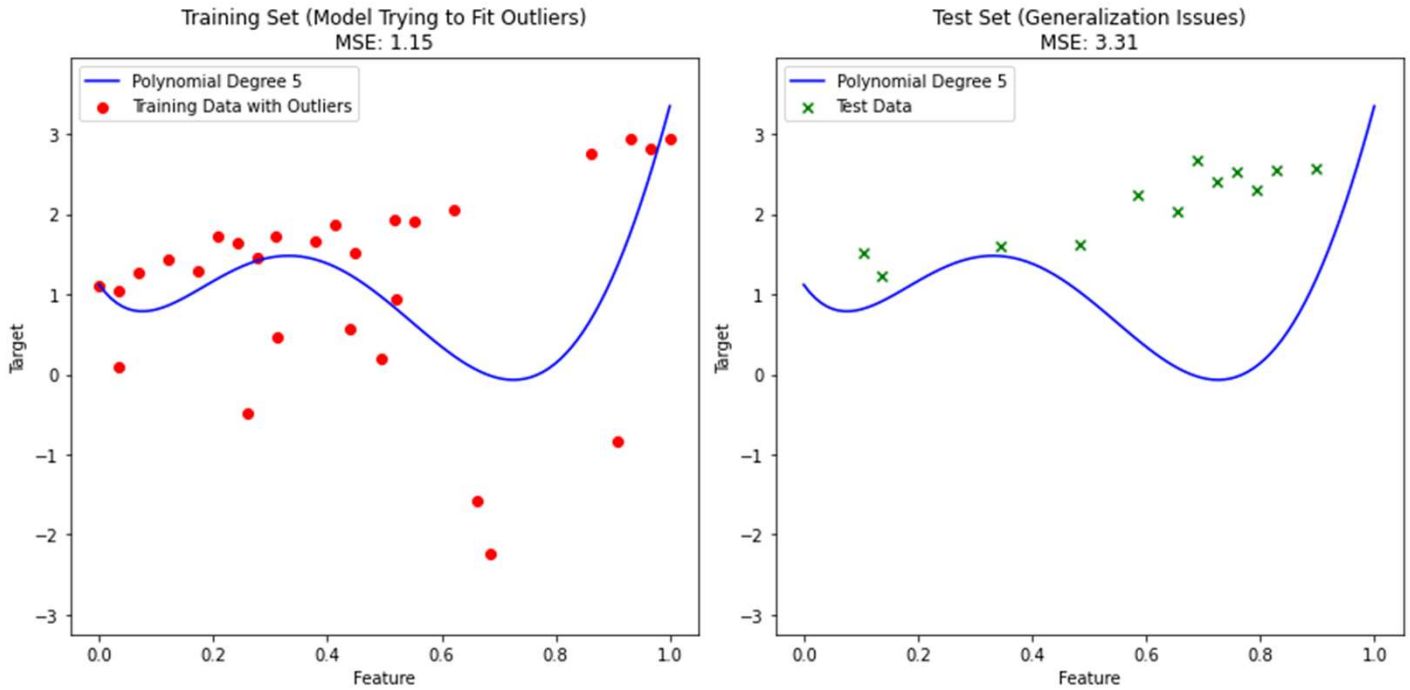
Overfitting: Complex models



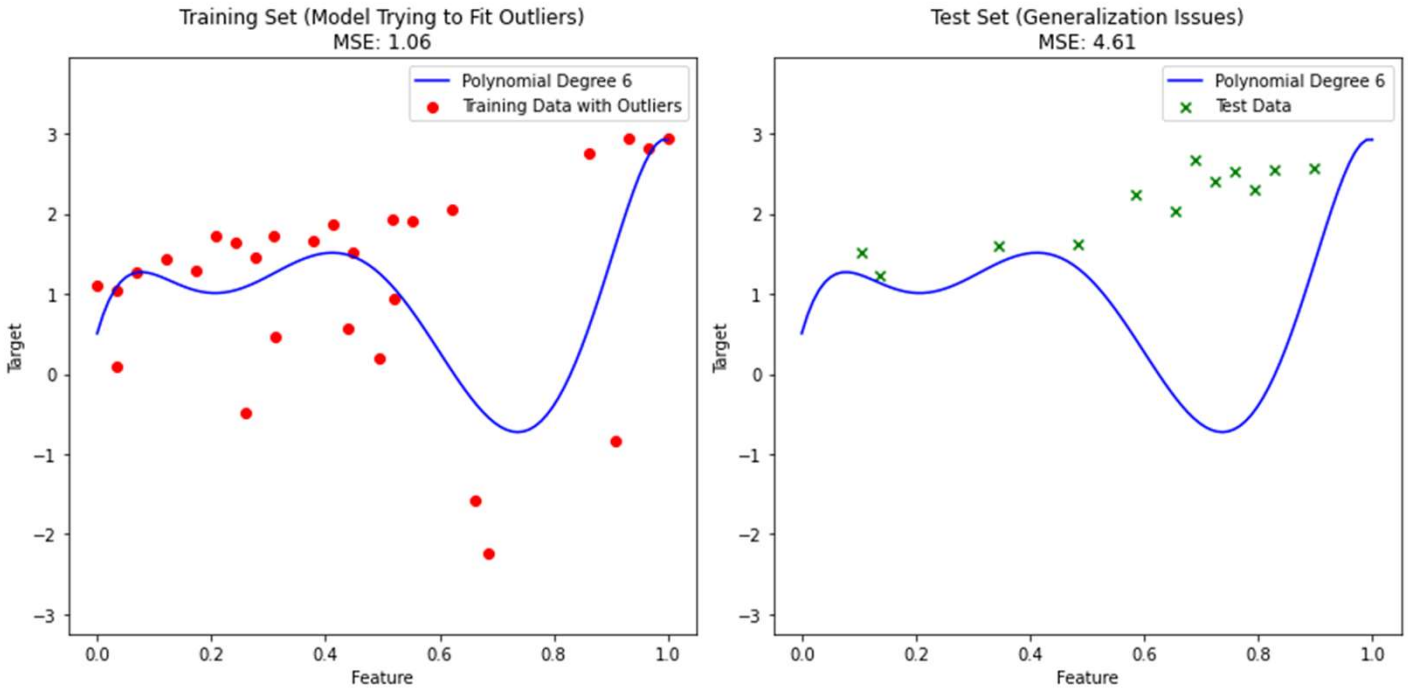
Overfitting: Complex models



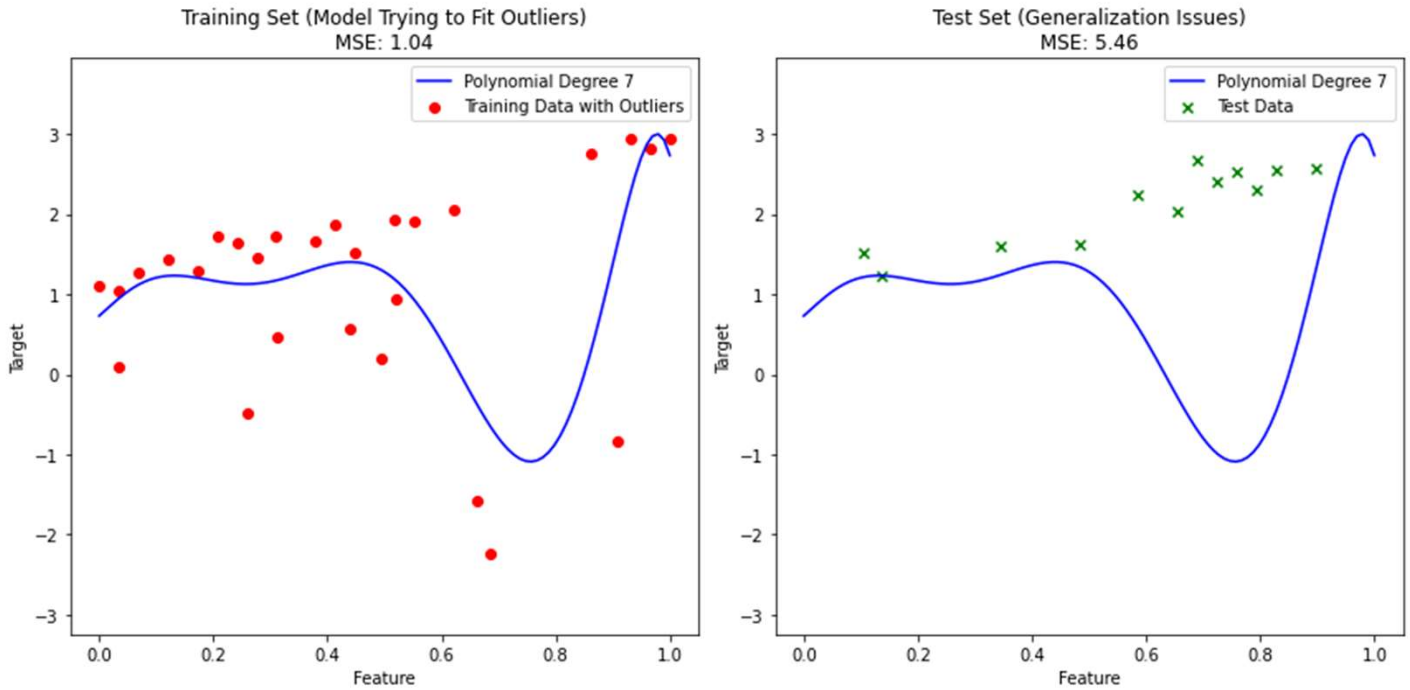
Overfitting: Complex models



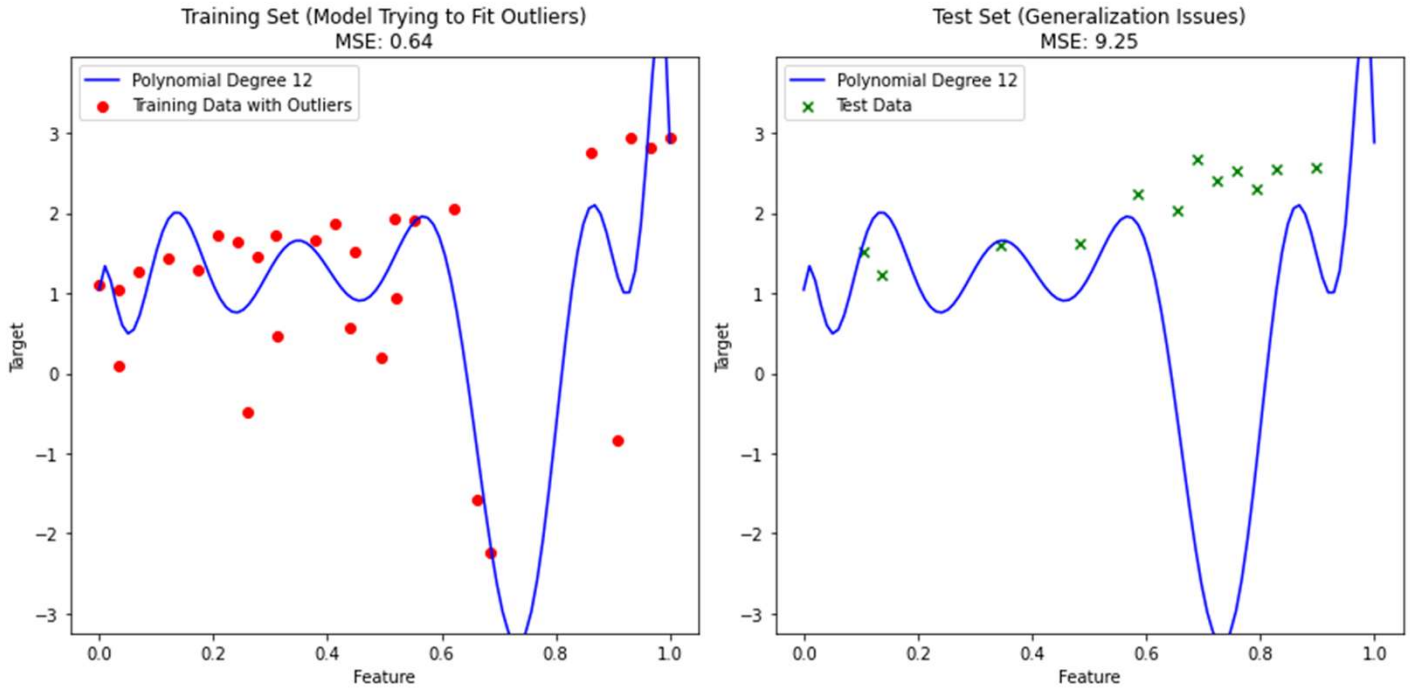
Overfitting: Complex models



Overfitting: Complex models

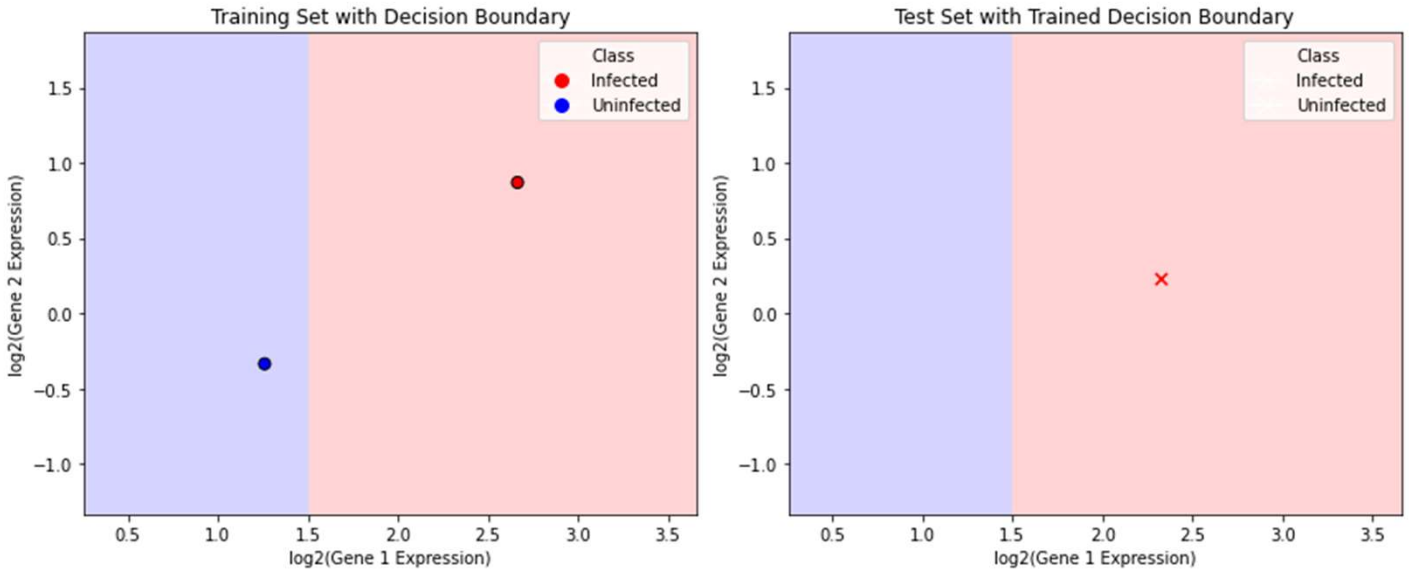


Overfitting: Complex models



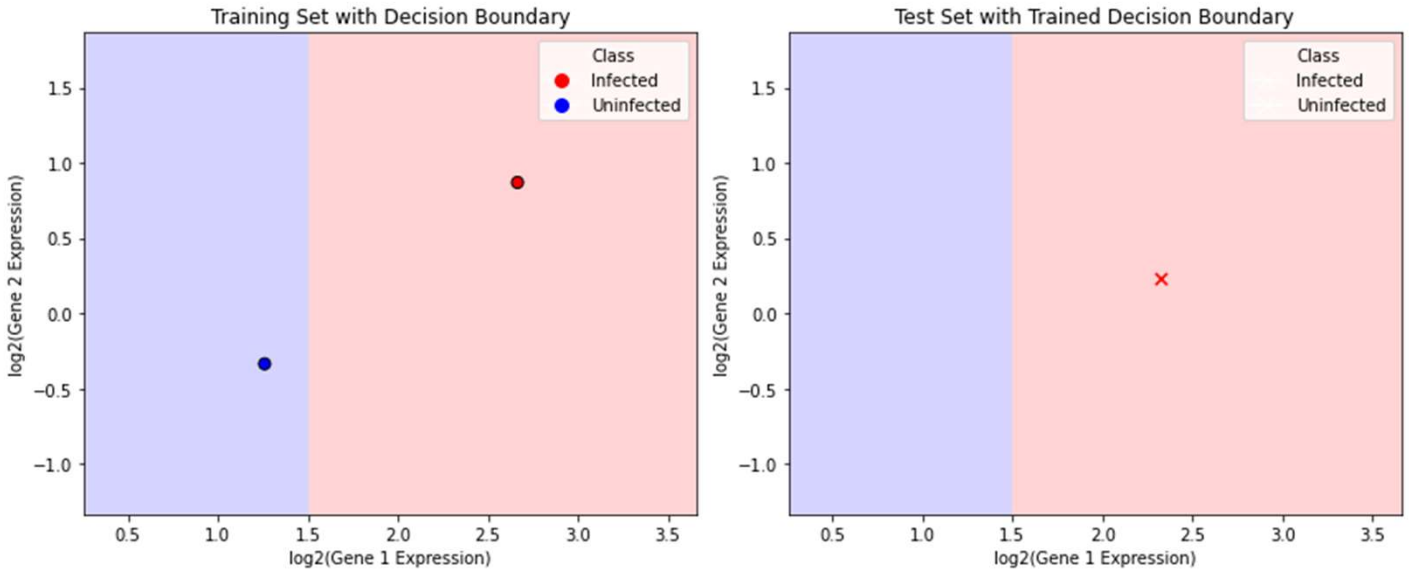
Overfitting: Too many variables relative to the number of samples

Case: m=3, n=2

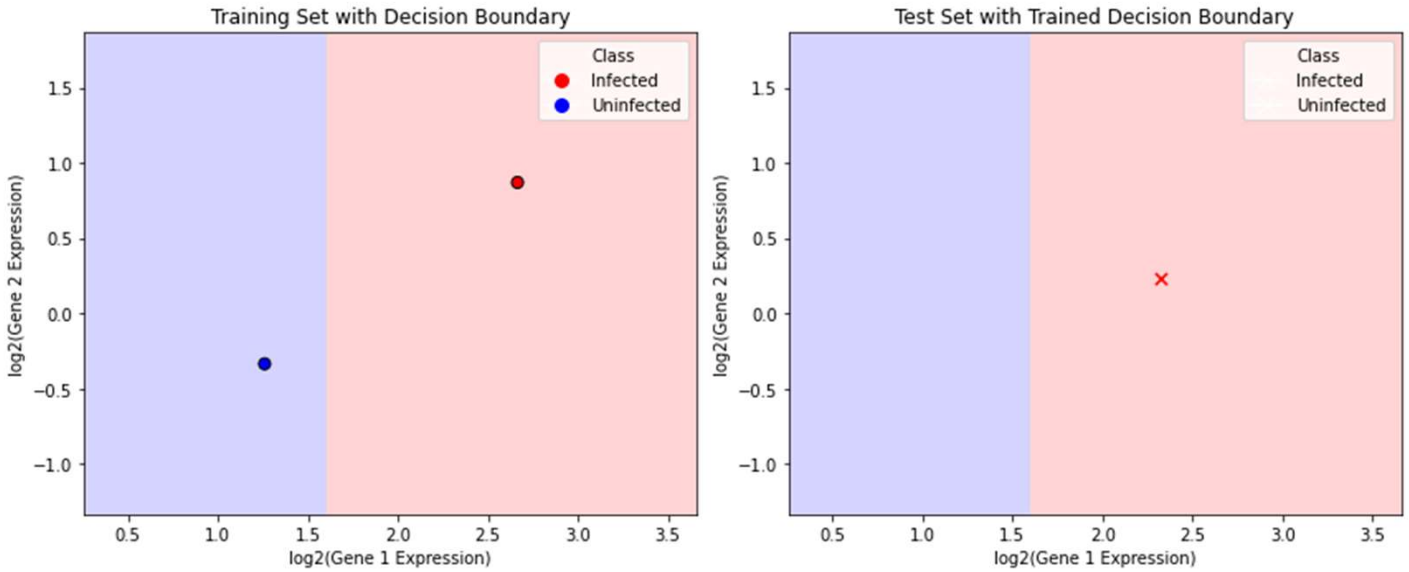


Sample ID	Sample 1	Sample 2	Sample 3
Class	Control	Infected	Infected
Gene 1	1.40167	3.4017	0.0001
Gene 2	0.0001	0.0001	0.0001

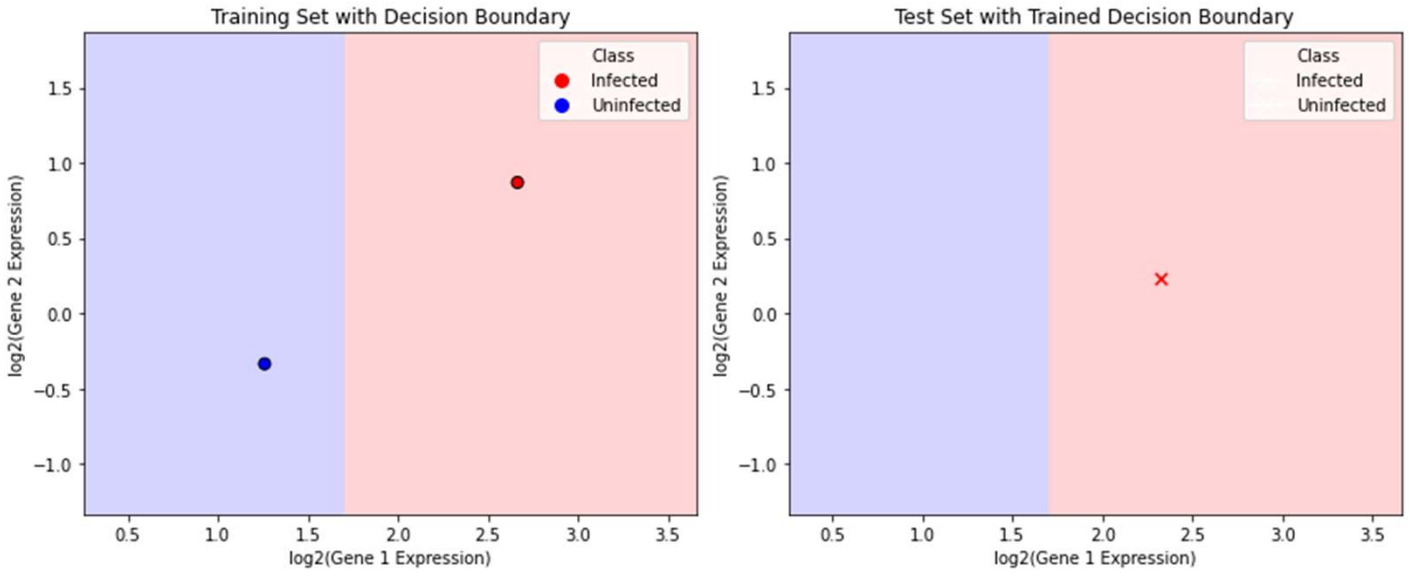
Overfitting: Too many variables relative to the number of samples



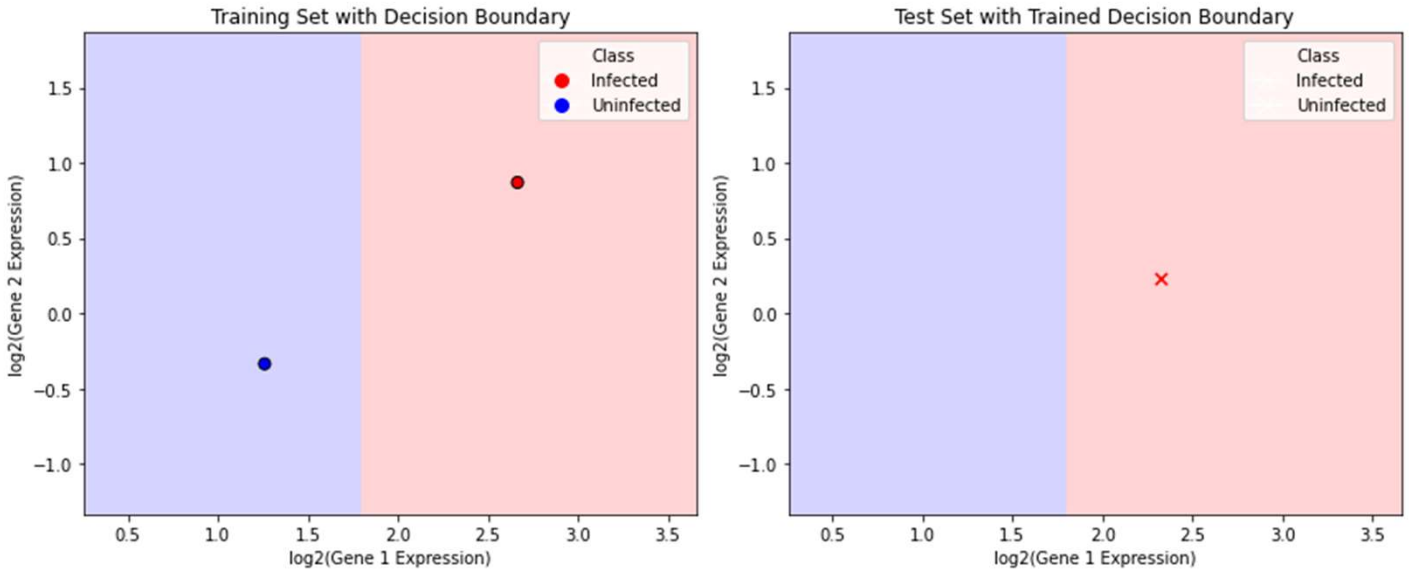
Overfitting: Too many variables relative to the number of samples



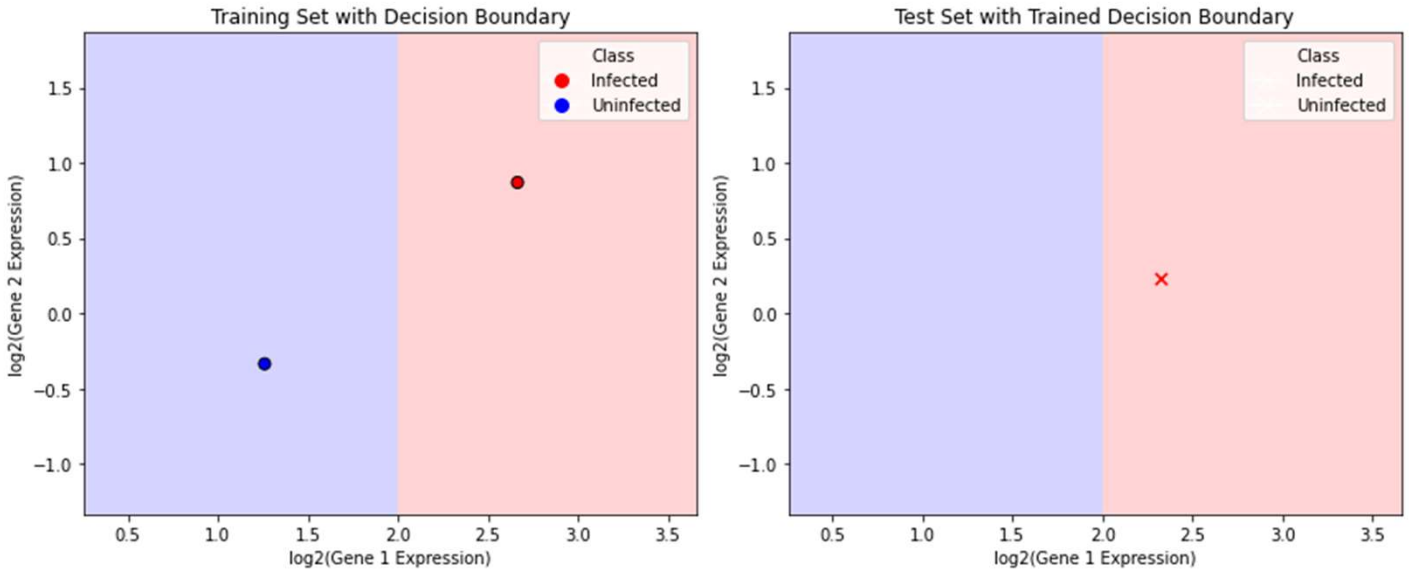
Overfitting: Too many variables relative to the number of samples



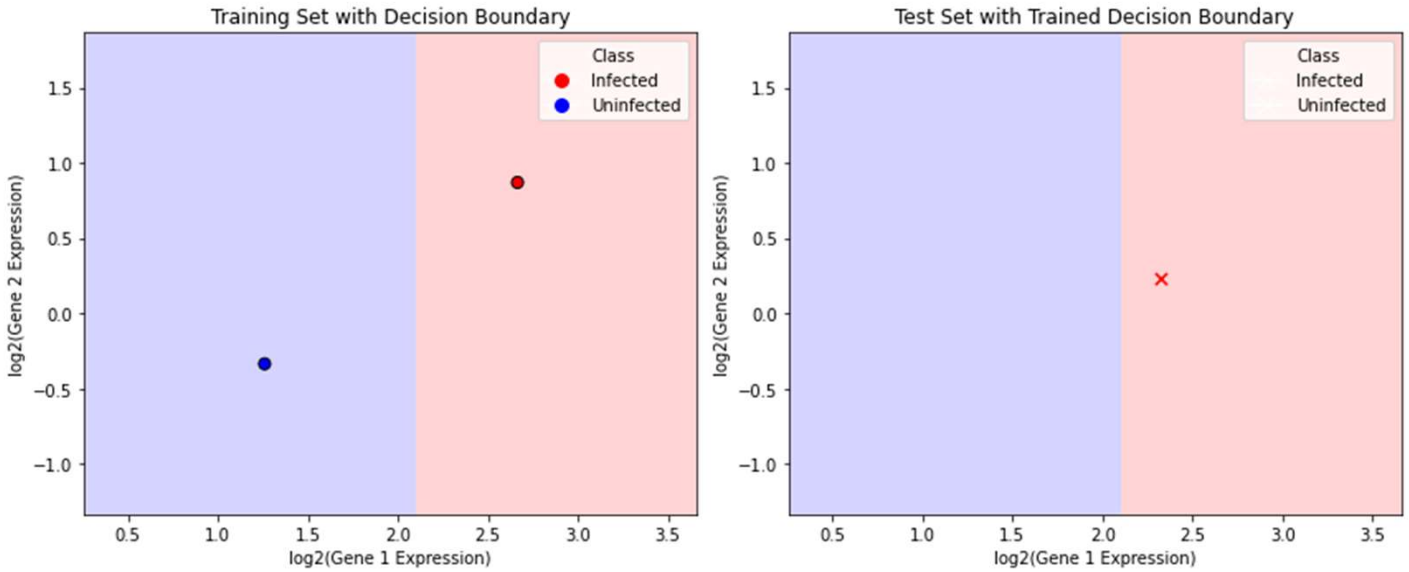
Overfitting: Too many variables relative to the number of samples



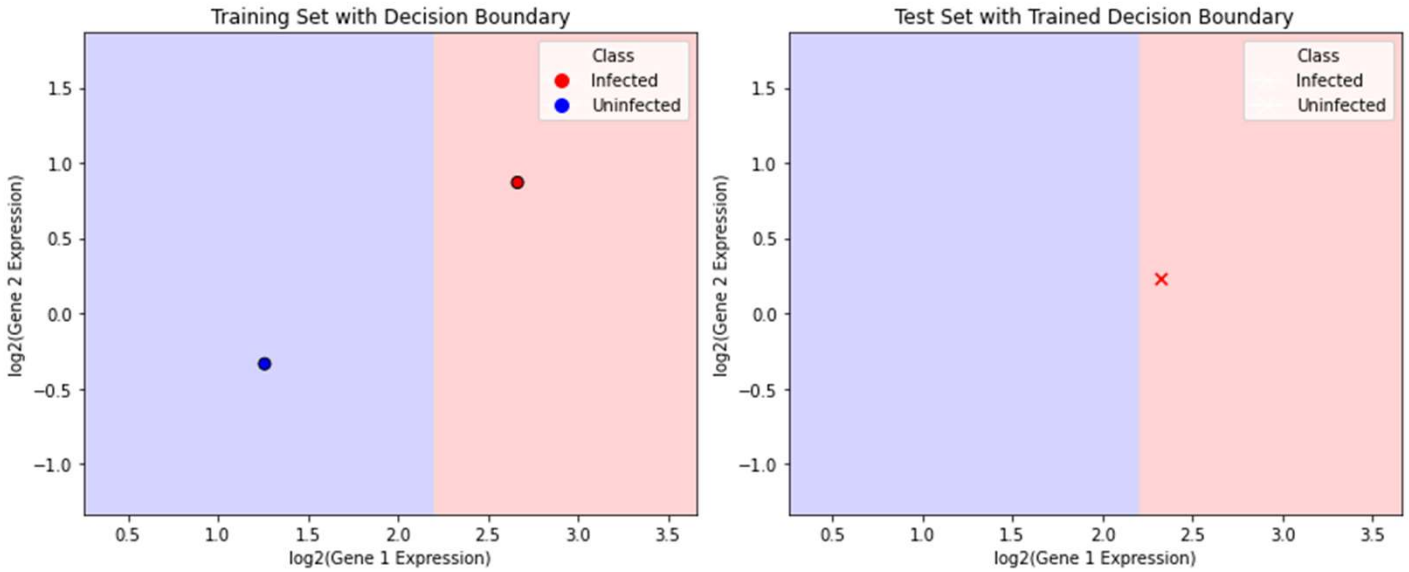
Overfitting: Too many variables relative to the number of samples



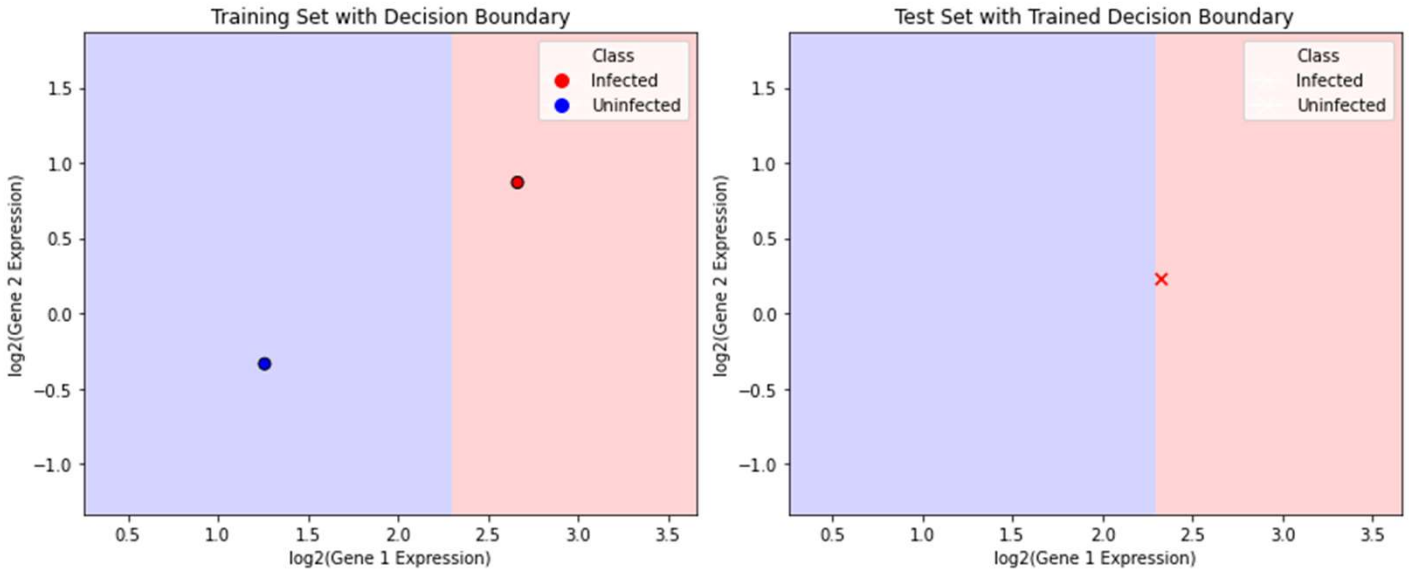
Overfitting: Too many variables relative to the number of samples



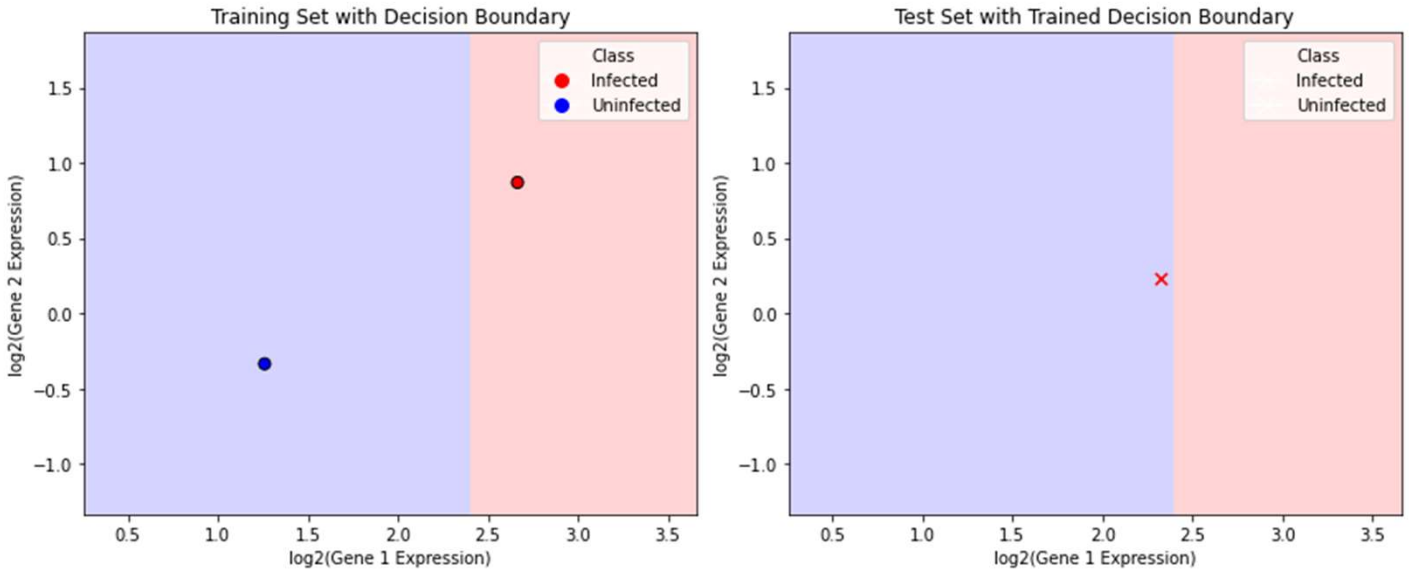
Overfitting: Too many variables relative to the number of samples



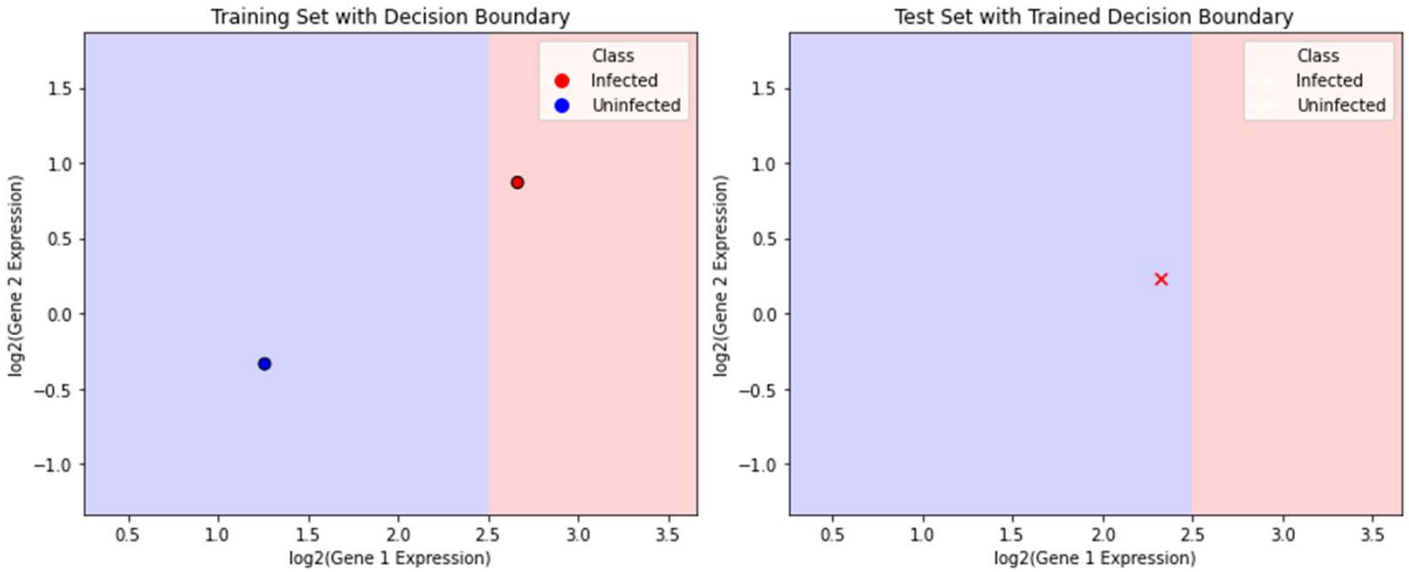
Overfitting: Too many variables relative to the number of samples



Overfitting: Too many variables relative to the number of samples



Overfitting: Too many variables relative to the number of samples



The Breast Cancer Wisconsin (Diagnostic) dataset

```
Feature Names:
['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension err
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimensi

Target Names:
['malignant' 'benign']

Number of Samples: 569
Number of Features: 30

Class Distribution:
malignant : 212
benign : 357
```

```python
from sklearn.datasets import load_breast_cancer

# Load the dataset
data = load_breast_cancer()

# Access the feature data
X = data.data
# Access the target variable
y = data.target

# Print the feature names
feature_names = data.feature_names
print("Feature Names:")
print(feature_names)

# Print the target names
target_names = data.target_names
print("\nTarget Names:")
print(target_names)

# Print the number of samples and features
n_samples, n_features = X.shape
print("\nNumber of Samples:", n_samples)
print("Number of Features:", n_features)

# Print the class distribution
class_distribution = {target_names[i]: list(y).count(i) for i in range(len(target_names))}
print("\nClass Distribution:")
for class_name, count in class_distribution.items():
    print(class_name, ":", count)
```

Wake Forest University
School of Medicine

CAIR
CENTER FOR ARTIFICIAL INTELLIGENCE RESEARCH

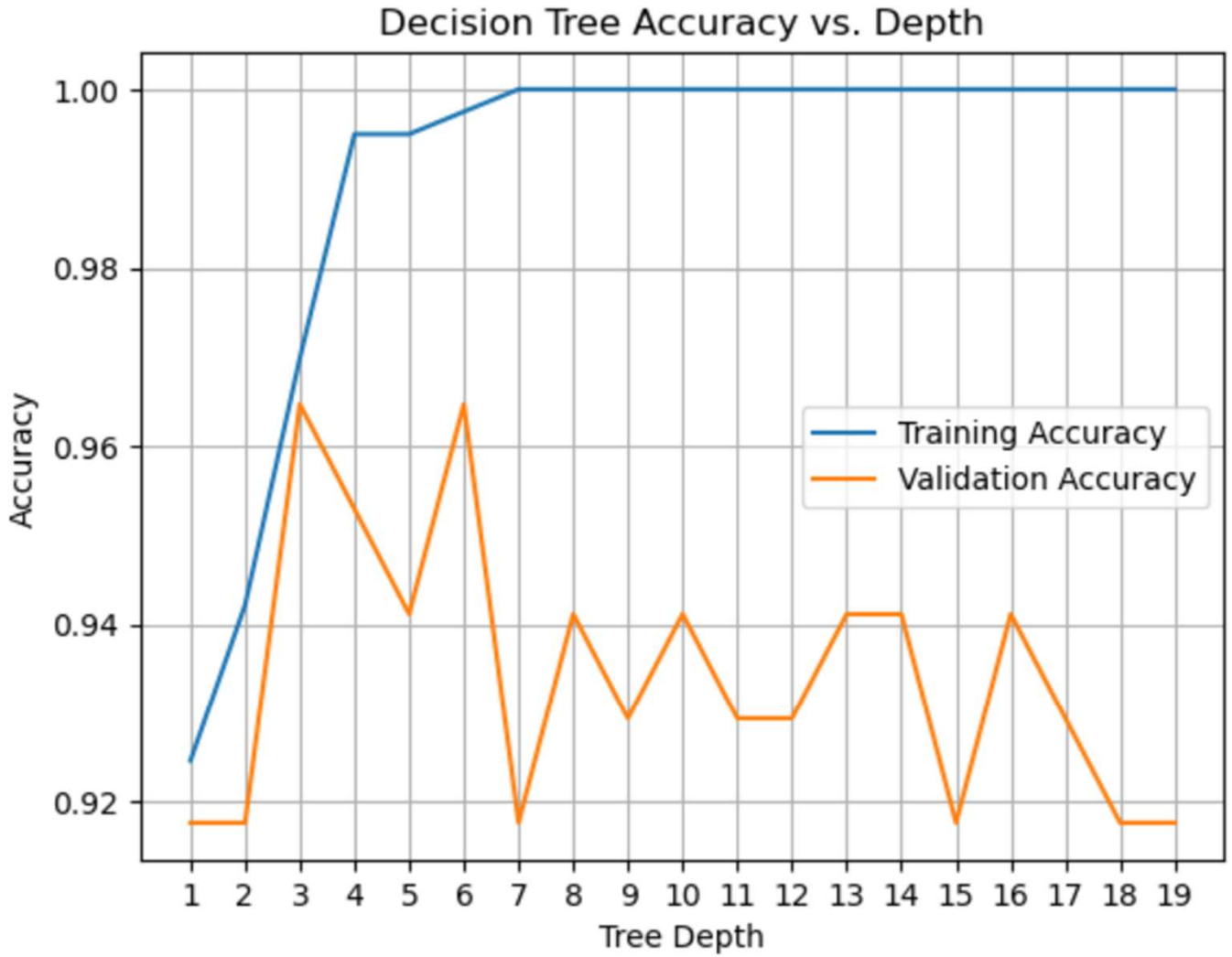
```python
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import numpy as np
# Load the dataset
data = load_breast_cancer()
X = data.data
y = data.target
# Split the dataset into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
# Train the decision tree with different depths and evaluate on validation set
depths = range(1, 20)
train_scores = []
val_scores = []
for depth in depths:
    # Create and train the decision tree classifier
    model = DecisionTreeClassifier(max_depth=depth)
    model.fit(X_train, y_train)
    # Predict on the training and validation sets
    y_train_pred = model.predict(X_train)
    y_val_pred = model.predict(X_val)
    # Calculate accuracy scores
    train_acc = accuracy_score(y_train, y_train_pred)
    val_acc = accuracy_score(y_val, y_val_pred)
    # Append scores to the lists
    train_scores.append(train_acc)
    val_scores.append(val_acc)
# Plot the training and validation accuracy curves
plt.plot(depths, train_scores, label='Training Accuracy')
plt.plot(depths, val_scores, label='Validation Accuracy')
plt.xlabel('Tree Depth')
plt.ylabel('Accuracy')
plt.xticks(np.arange(min(depths), max(depths)+1, 1))
plt.legend()
plt.title('Decision Tree Accuracy vs. Depth')
plt.grid(True)
plt.show()
```

```python
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import numpy as np
# Load the dataset
data = load_breast_cancer()
X = data.data
y = data.target
# Split the dataset into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
# Train the decision tree with different depths and evaluate on validation set
depths = range(1, 20)
train_scores = []
val_scores = []
for depth in depths:
    # Create and train the decision tree classifier
    model = DecisionTreeClassifier(max_depth=depth)
    model.fit(X_train, y_train)
    # Predict on the training and validation sets
    y_train_pred = model.predict(X_train)
    y_val_pred = model.predict(X_val)
    # Calculate accuracy scores
    train_acc = accuracy_score(y_train, y_train_pred)
    val_acc = accuracy_score(y_val, y_val_pred)
    # Append scores to the lists
    train_scores.append(train_acc)
    val_scores.append(val_acc)
# Plot the training and validation accuracy curves
plt.plot(depths, train_scores, label='Training Accuracy')
plt.plot(depths, val_scores, label='Validation Accuracy')
plt.xlabel('Tree Depth')
plt.ylabel('Accuracy')
plt.xticks(np.arange(min(depths), max(depths)+1, 1))
plt.legend()
plt.title('Decision Tree Accuracy vs. Depth')
plt.grid(True)
plt.show()
```

```python
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import numpy as np
# Load the dataset
data = load_breast_cancer()
X = data.data
y = data.target
# Split the dataset into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
# Train the decision tree with different depths and evaluate on validation set
depths = range(1, 20)
train_scores = []
val_scores = []
for depth in depths:
    # Create and train the decision tree classifier
    model = DecisionTreeClassifier(max_depth=depth)
    model.fit(X_train, y_train)
    # Predict on the training and validation sets
    y_train_pred = model.predict(X_train)
    y_val_pred = model.predict(X_val)
    # Calculate accuracy scores
    train_acc = accuracy_score(y_train, y_train_pred)
    val_acc = accuracy_score(y_val, y_val_pred)
    # Append scores to the lists
    train_scores.append(train_acc)
    val_scores.append(val_acc)
# Plot the training and validation accuracy curves
plt.plot(depths, train_scores, label='Training Accuracy')
plt.plot(depths, val_scores, label='Validation Accuracy')
plt.xlabel('Tree Depth')
plt.ylabel('Accuracy')
plt.xticks(np.arange(min(depths), max(depths)+1, 1))
plt.legend()
plt.title('Decision Tree Accuracy vs. Depth')
plt.grid(True)
plt.show()
```

```python
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import numpy as np
# Load the dataset
data = load_breast_cancer()
X = data.data
y = data.target
# Split the dataset into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
# Train the decision tree with different depths and evaluate on validation set
depths = range(1, 20)
train_scores = []
val_scores = []
for depth in depths:
    # Create and train the decision tree classifier
    model = DecisionTreeClassifier(max_depth=depth)
    model.fit(X_train, y_train)
    # Predict on the training and validation sets
    y_train_pred = model.predict(X_train)
    y_val_pred = model.predict(X_val)
    # Calculate accuracy scores
    train_acc = accuracy_score(y_train, y_train_pred)
    val_acc = accuracy_score(y_val, y_val_pred)
    # Append scores to the lists
    train_scores.append(train_acc)
    val_scores.append(val_acc)
# Plot the training and validation accuracy curves
plt.plot(depths, train_scores, label='Training Accuracy')
plt.plot(depths, val_scores, label='Validation Accuracy')
plt.xlabel('Tree Depth')
plt.ylabel('Accuracy')
plt.xticks(np.arange(min(depths), max(depths)+1, 1))
plt.legend()
plt.title('Decision Tree Accuracy vs. Depth')
plt.grid(True)
plt.show()
```

```python
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import numpy as np
# Load the dataset
data = load_breast_cancer()
X = data.data
y = data.target
# Split the dataset into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
# Train the decision tree with different depths and evaluate on validation set
depths = range(1, 20)
train_scores = []
val_scores = []
for depth in depths:
    # Create and train the decision tree classifier
    model = DecisionTreeClassifier(max_depth=depth)
    model.fit(X_train, y_train)
    # Predict on the training and validation sets
    y_train_pred = model.predict(X_train)
    y_val_pred = model.predict(X_val)
    # Calculate accuracy scores
    train_acc = accuracy_score(y_train, y_train_pred)
    val_acc = accuracy_score(y_val, y_val_pred)
    # Append scores to the lists
    train_scores.append(train_acc)
    val_scores.append(val_acc)
# Plot the training and validation accuracy curves
plt.plot(depths, train_scores, label='Training Accuracy')
plt.plot(depths, val_scores, label='Validation Accuracy')
plt.xlabel('Tree Depth')
plt.ylabel('Accuracy')
plt.xticks(np.arange(min(depths), max(depths)+1, 1))
plt.legend()
plt.title('Decision Tree Accuracy vs. Depth')
plt.grid(True)
plt.show()
```



Wake Forest University
School of Medicine

CAIR
CENTER FOR ARTIFICIAL INTELLIGENCE RESEARCH

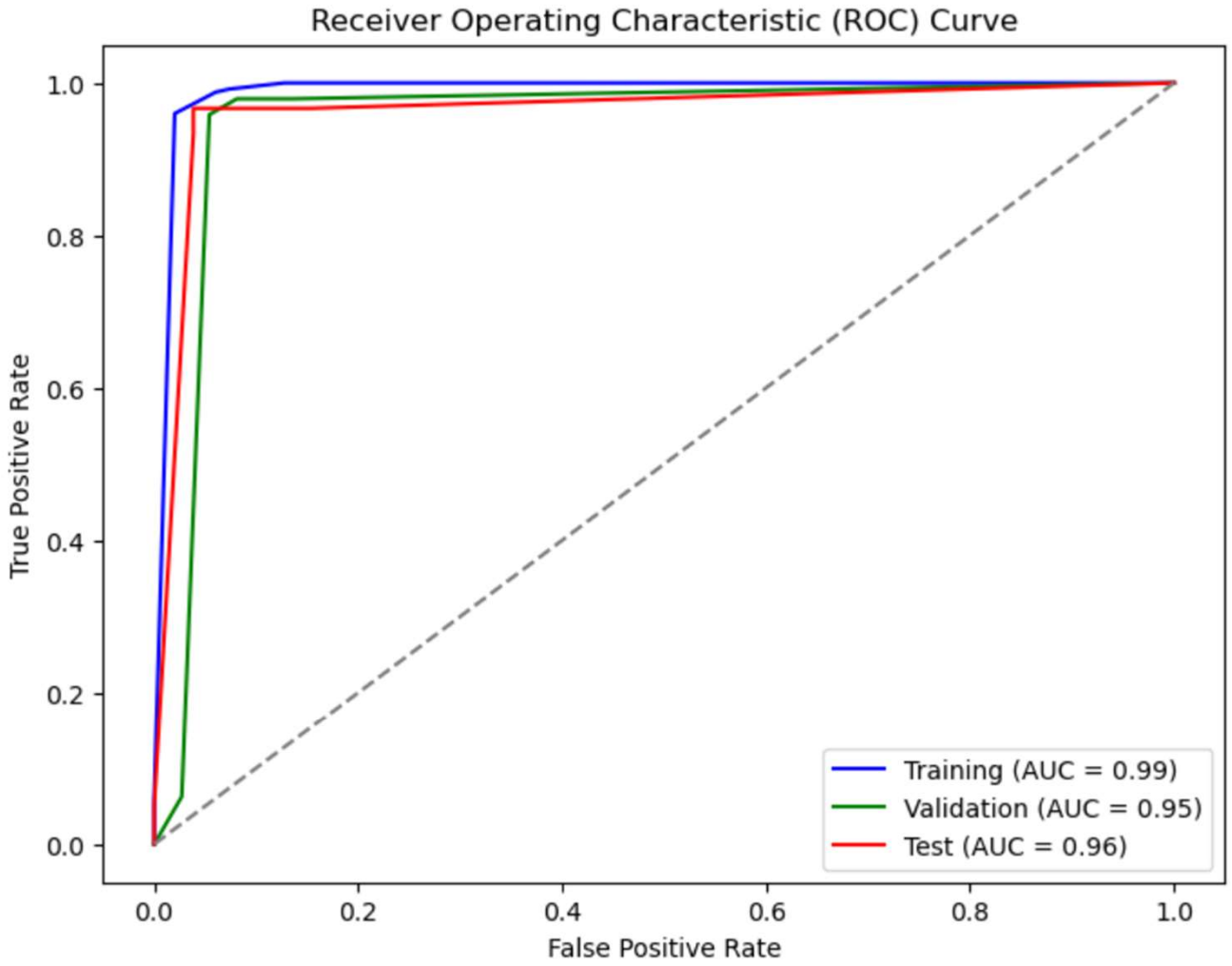
```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import roc_curve, roc_auc_score
# Load the dataset
data = load_breast_cancer()
X = data.data
y = data.target
# Split the dataset into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
# Train the decision tree with depth 3
depth = 3
model = DecisionTreeClassifier(max_depth=depth)
model.fit(X_train, y_train)
# Predict probabilities on the training, validation, and test sets
y_train_prob = model.predict_proba(X_train)[:, 1]
y_val_prob = model.predict_proba(X_val)[:, 1]
y_test_prob = model.predict_proba(X_test)[:, 1]
# Calculate the ROC curve and AUC scores
fpr_train, tpr_train, _ = roc_curve(y_train, y_train_prob)
fpr_val, tpr_val, _ = roc_curve(y_val, y_val_prob)
fpr_test, tpr_test, _ = roc_curve(y_test, y_test_prob)
auc_train = roc_auc_score(y_train, y_train_prob)
auc_val = roc_auc_score(y_val, y_val_prob)
auc_test = roc_auc_score(y_test, y_test_prob)
# Plot the ROC curves
plt.figure(figsize=(8, 6))
plt.plot(fpr_train, tpr_train, label=f"Training (AUC = {auc_train:.2f})", color='b')
plt.plot(fpr_val, tpr_val, label=f"Validation (AUC = {auc_val:.2f})", color='g')
plt.plot(fpr_test, tpr_test, label=f"Test (AUC = {auc_test:.2f})", color='r')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import roc_curve, roc_auc_score
# Load the dataset
data = load_breast_cancer()
X = data.data
y = data.target
# Split the dataset into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
# Train the decision tree with depth 3
depth = 3
model = DecisionTreeClassifier(max_depth=depth)
model.fit(X_train, y_train)
# Predict probabilities on the training, validation, and test sets
y_train_prob = model.predict_proba(X_train)[:, 1]
y_val_prob = model.predict_proba(X_val)[:, 1]
y_test_prob = model.predict_proba(X_test)[:, 1]
# Calculate the ROC curve and AUC scores
fpr_train, tpr_train, _ = roc_curve(y_train, y_train_prob)
fpr_val, tpr_val, _ = roc_curve(y_val, y_val_prob)
fpr_test, tpr_test, _ = roc_curve(y_test, y_test_prob)
auc_train = roc_auc_score(y_train, y_train_prob)
auc_val = roc_auc_score(y_val, y_val_prob)
auc_test = roc_auc_score(y_test, y_test_prob)
# Plot the ROC curves
plt.figure(figsize=(8, 6))
plt.plot(fpr_train, tpr_train, label=f"Training (AUC = {auc_train:.2f})", color='b')
plt.plot(fpr_val, tpr_val, label=f"Validation (AUC = {auc_val:.2f})", color='g')
plt.plot(fpr_test, tpr_test, label=f"Test (AUC = {auc_test:.2f})", color='r')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import roc_curve, roc_auc_score
# Load the dataset
data = load_breast_cancer()
X = data.data
y = data.target
# Split the dataset into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
# Train the decision tree with depth 3
depth = 3
model = DecisionTreeClassifier(max_depth=depth)
model.fit(X_train, y_train)
# Predict probabilities on the training, validation, and test sets
y_train_prob = model.predict_proba(X_train)[:, 1]
y_val_prob = model.predict_proba(X_val)[:, 1]
y_test_prob = model.predict_proba(X_test)[:, 1]
# Calculate the ROC curve and AUC scores
fpr_train, tpr_train, _ = roc_curve(y_train, y_train_prob)
fpr_val, tpr_val, _ = roc_curve(y_val, y_val_prob)
fpr_test, tpr_test, _ = roc_curve(y_test, y_test_prob)
auc_train = roc_auc_score(y_train, y_train_prob)
auc_val = roc_auc_score(y_val, y_val_prob)
auc_test = roc_auc_score(y_test, y_test_prob)
# Plot the ROC curves
plt.figure(figsize=(8, 6))
plt.plot(fpr_train, tpr_train, label=f"Training (AUC = {auc_train:.2f})", color='b')
plt.plot(fpr_val, tpr_val, label=f"Validation (AUC = {auc_val:.2f})", color='g')
plt.plot(fpr_test, tpr_test, label=f"Test (AUC = {auc_test:.2f})", color='r')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import roc_curve, roc_auc_score
# Load the dataset
data = load_breast_cancer()
X = data.data
y = data.target
# Split the dataset into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
# Train the decision tree with depth 3
depth = 3
model = DecisionTreeClassifier(max_depth=depth)
model.fit(X_train, y_train)
# Predict probabilities on the training, validation, and test sets
y_train_prob = model.predict_proba(X_train)[:, 1]
y_val_prob = model.predict_proba(X_val)[:, 1]
y_test_prob = model.predict_proba(X_test)[:, 1]
# Calculate the ROC curve and AUC scores
fpr_train, tpr_train, _ = roc_curve(y_train, y_train_prob)
fpr_val, tpr_val, _ = roc_curve(y_val, y_val_prob)
fpr_test, tpr_test, _ = roc_curve(y_test, y_test_prob)
auc_train = roc_auc_score(y_train, y_train_prob)
auc_val = roc_auc_score(y_val, y_val_prob)
auc_test = roc_auc_score(y_test, y_test_prob)
# Plot the ROC curves
plt.figure(figsize=(8, 6))
plt.plot(fpr_train, tpr_train, label=f"Training (AUC = {auc_train:.2f})", color='b')
plt.plot(fpr_val, tpr_val, label=f"Validation (AUC = {auc_val:.2f})", color='g')
plt.plot(fpr_test, tpr_test, label=f"Test (AUC = {auc_test:.2f})", color='r')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import roc_curve, roc_auc_score
# Load the dataset
data = load_breast_cancer()
X = data.data
y = data.target
# Split the dataset into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
# Train the decision tree with depth 3
depth = 3
model = DecisionTreeClassifier(max_depth=depth)
model.fit(X_train, y_train)
# Predict probabilities on the training, validation, and test sets
y_train_prob = model.predict_proba(X_train)[:, 1]
y_val_prob = model.predict_proba(X_val)[:, 1]
y_test_prob = model.predict_proba(X_test)[:, 1]
# Calculate the ROC curve and AUC scores
fpr_train, tpr_train, _ = roc_curve(y_train, y_train_prob)
fpr_val, tpr_val, _ = roc_curve(y_val, y_val_prob)
fpr_test, tpr_test, _ = roc_curve(y_test, y_test_prob)
auc_train = roc_auc_score(y_train, y_train_prob)
auc_val = roc_auc_score(y_val, y_val_prob)
auc_test = roc_auc_score(y_test, y_test_prob)
# Plot the ROC curves
plt.figure(figsize=(8, 6))
plt.plot(fpr_train, tpr_train, label=f"Training (AUC = {auc_train:.2f})", color='b')
plt.plot(fpr_val, tpr_val, label=f"Validation (AUC = {auc_val:.2f})", color='g')
plt.plot(fpr_test, tpr_test, label=f"Test (AUC = {auc_test:.2f})", color='r')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```



Wake Forest University
School of Medicine

CAIR
CENTER FOR ARTIFICIAL INTELLIGENCE RESEARCH

Real-World Applications of Decision Trees and Tree-Based Ensemble Models

Wake Forest University
School of Medicine

CAIR
CENTER FOR ARTIFICIAL INTELLIGENCE RESEARCH

Machine learning-based analytics of the impact of the Covid-19 pandemic on alcohol consumption habit changes among United States healthcare workers

Mostafa Rezapour, Muhammad Khalid Khan Niazi & Metin Nafi Gurcan

Wake Forest University
School of Medicine

CAIR
CENTER FOR ARTIFICIAL INTELLIGENCE RESEARCH

Data Collection

Data Source: Survey data from the University of Michigan's ICPSR.

Data Collection

Data Preprocessing

Initial Dataset (916 rows, 64 columns)

1. Are children home from school in the house?
2. Has the number of your work hours per week changed?
3. Have you varied your work schedule?
4. Have your sleep patterns changed?
5. Has the number of naps you are taking changed?
6. etc.

Data Collection

Data Preprocessing

Initial Dataset (916 rows, 64 columns)

Target Variable: Question 18a

Question 18a: "Please tell us how the amount of alcohol that you are consuming has changed?"

Response: **I am drinking less alcohol**	Response: **I am drinking more alcohol**
• Supervised classification Label: **0**	• Supervised classification Label: **1**
• Relative frequency: **13.9%**	• Relative frequency: **86.1%**
○ Gender:	○ Gender:
▪ **female: 9.52%,**	▪ **female: 70.33%,**
▪ **male: 4.40%,**	▪ **male: 15.38%,**
▪ **Not prefer to say: 0.0%**	▪ **Not prefer to say: 0.37%**
○ Age:	○ Age:
▪ **2.20 %** in 20s	▪ **12.82 %** in 20s
▪ **2.93 %** in 30s	▪ **29.67 %** in 30s
▪ **3.66 %** in 40s	▪ **20.14 %** in 40s
▪ **2.56 %** in 50s	▪ **13.55 %** in 50s
▪ **1.46 %** in 60s	▪ **8.79 %** in 60s
▪ **0.73 %** in 70s	▪ **1.10 %** in 70s
▪ **0.37 %** in 80s	▪ **0.00 %** in 80s

Data Collection

Data Preprocessing

Initial Dataset (916 rows, 64 columns)

Target Variable: Question 18a

Removed rows without Question 18a response

Dropped unrelated columns

1. Start Date
2. End Date
3. Response Type
4. IP Address
5. Recorded Date
6. Response ID
7. Recipient Last Name
8. Recipient First Name
9. Recipient Email
10. External Data Reference

Wake Forest University
School of Medicine

CAIR
CENTER FOR ARTIFICIAL INTELLIGENCE RESEARCH

Data Collection

Data Preprocessing

Initial Dataset (916 rows, 64 columns)

Target Variable: Question 18a

Removed rows without Question 18a response

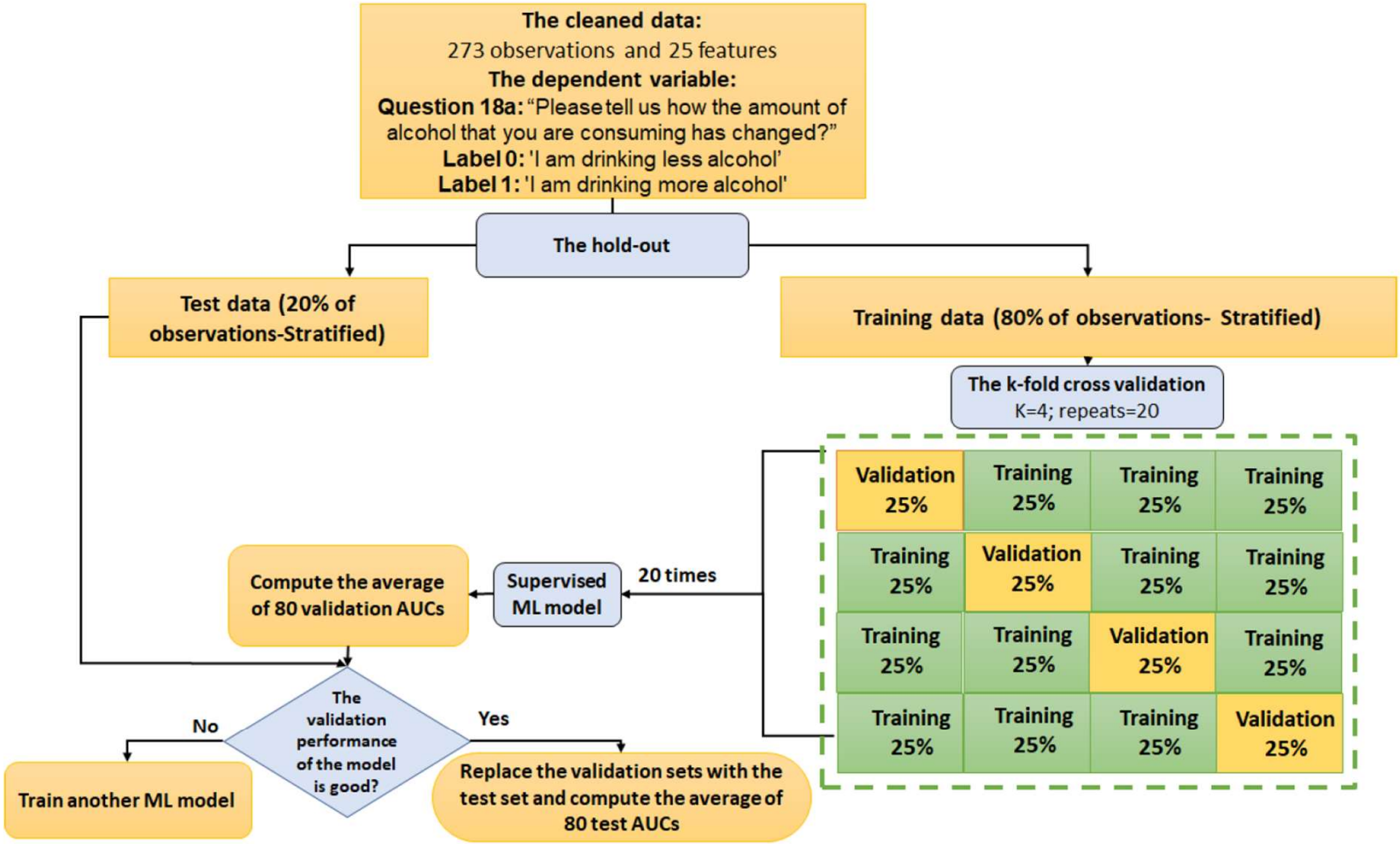
Dropped unrelated columns

Removed rows with too many missing values

Applied encoding techniques

Wake Forest University
School of Medicine

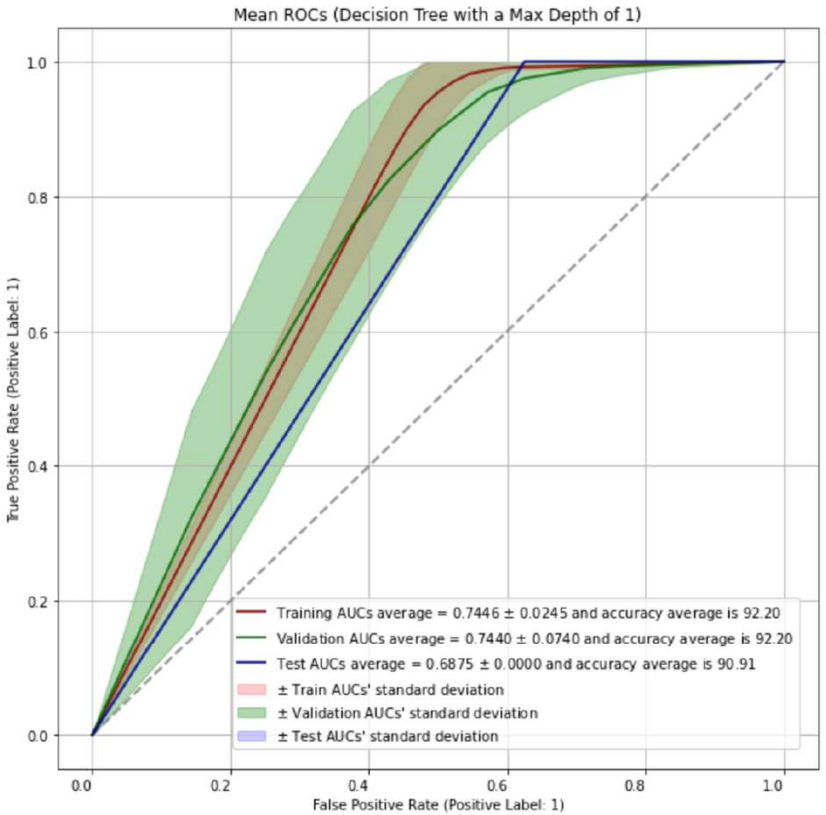
CAIR
CENTER FOR ARTIFICIAL INTELLIGENCE RESEARCH



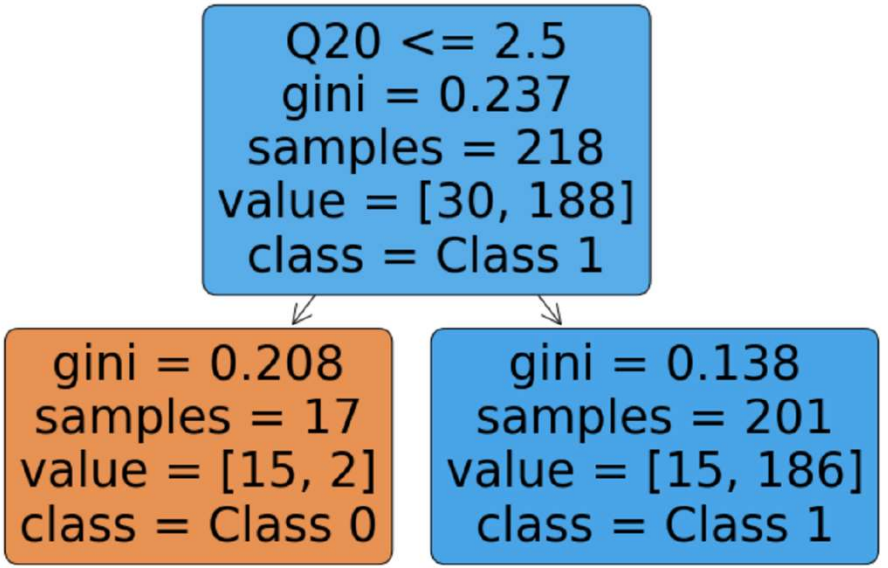
Wake Forest University
School of Medicine

CAIR
CENTER FOR ARTIFICIAL INTELLIGENCE RESEARCH

- Decision Tree with a Max Depth of 1

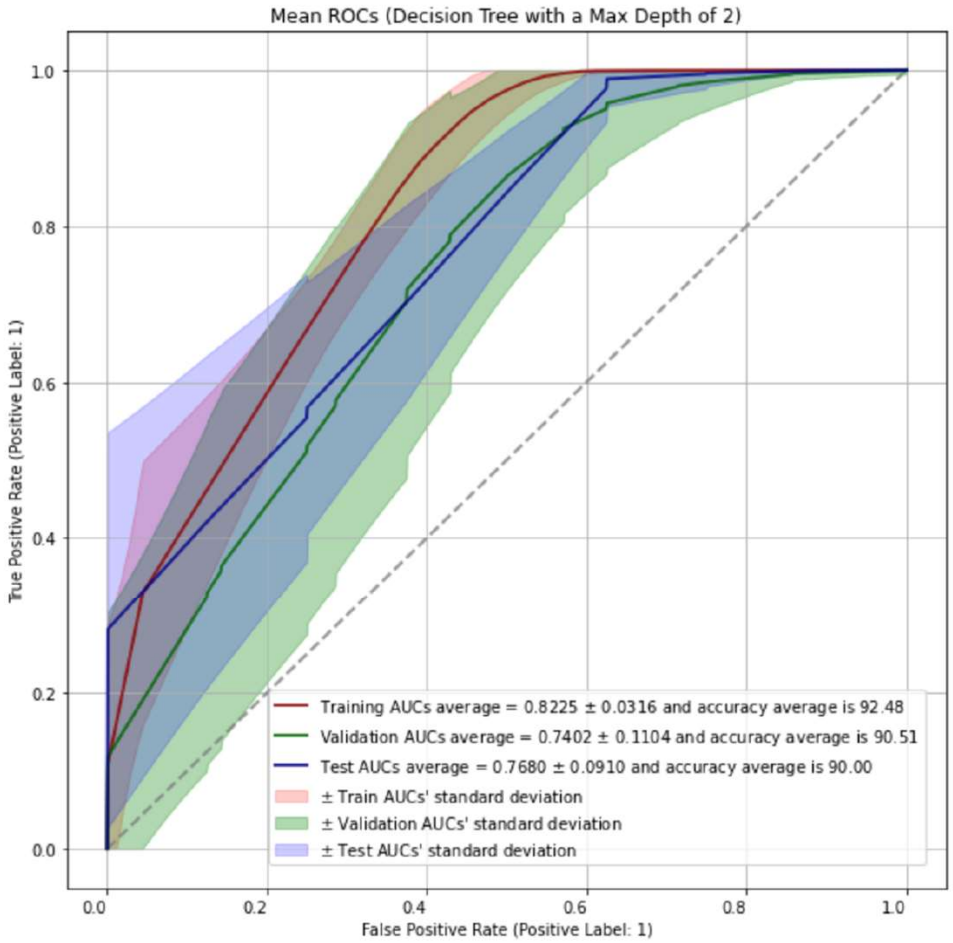


Q20: In the last month, approximately how often did you have a drink containing alcohol?



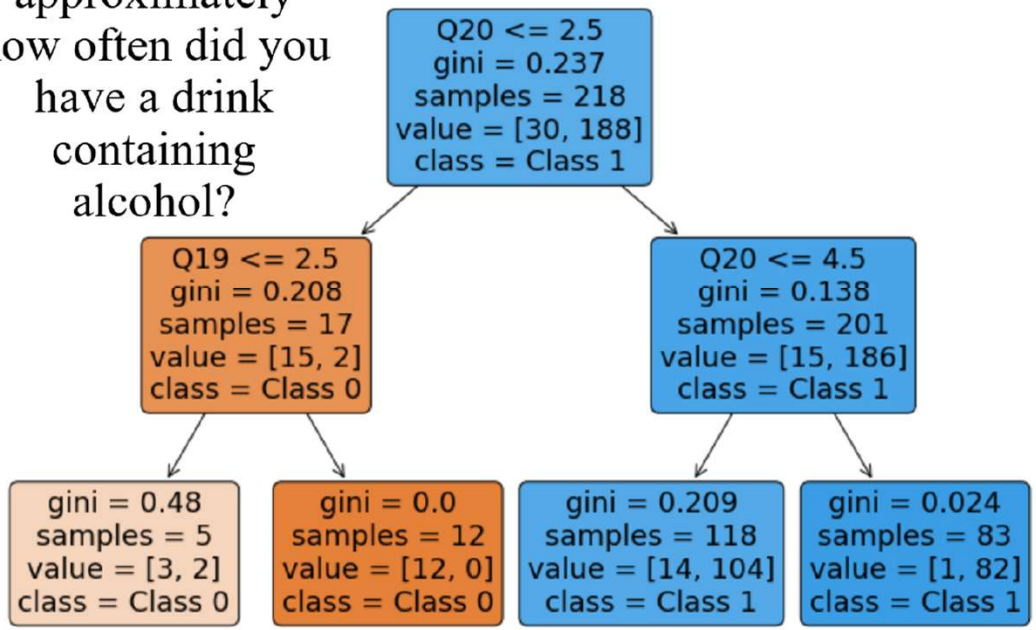
$$\text{Entropy (H)} = -\sum_{i=1}^{n} p_i \log(p_i), \quad \text{Gini}(D) = 1 - \sum_{i=1}^{n} p_i^2$$

- Decision Tree with a Max Depth of 2

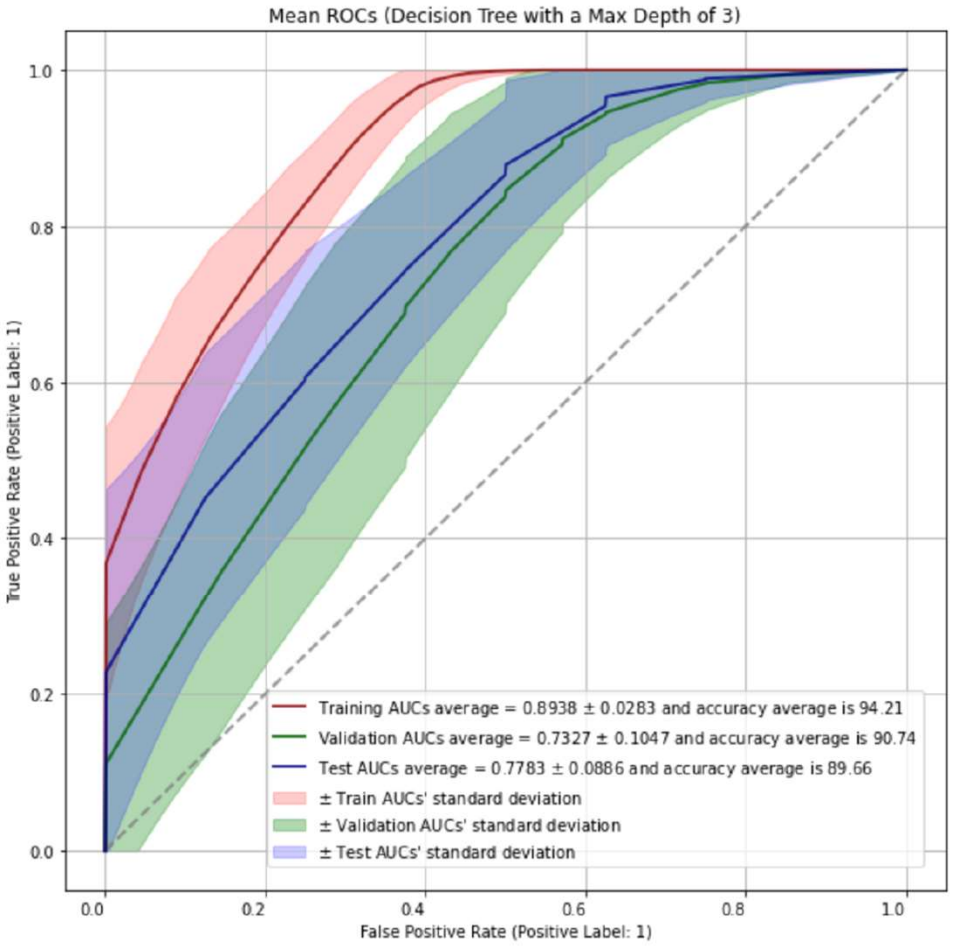


Q19: In January 2020, approximately how often did you have a drink containing alcohol?

Q20: In the last month, approximately how often did you have a drink containing alcohol?

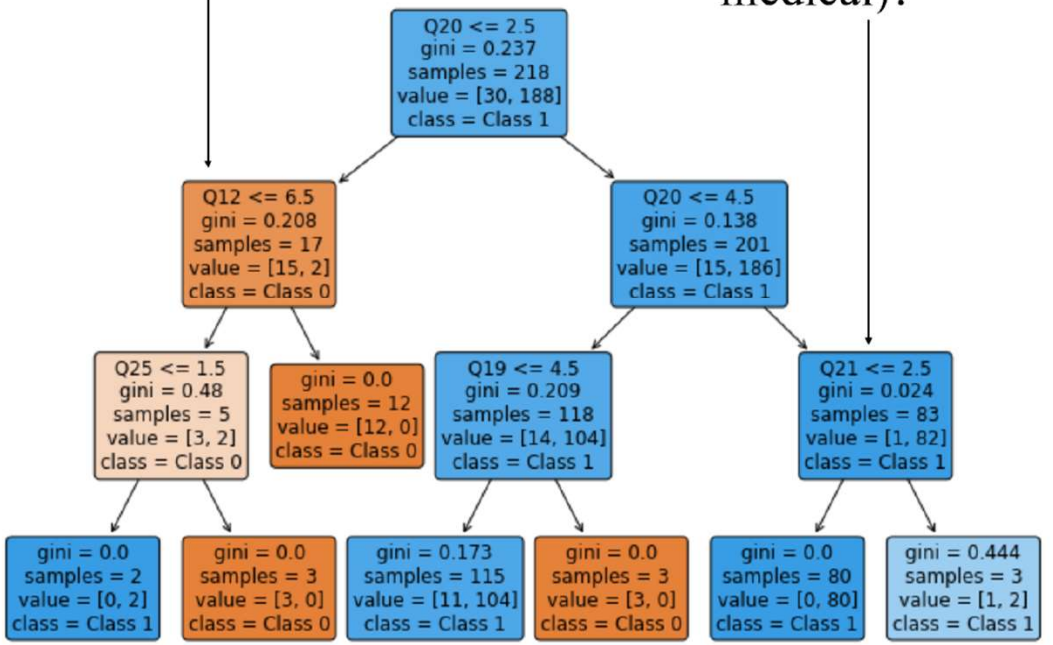


- Decision Tree with a Max Depth of 3

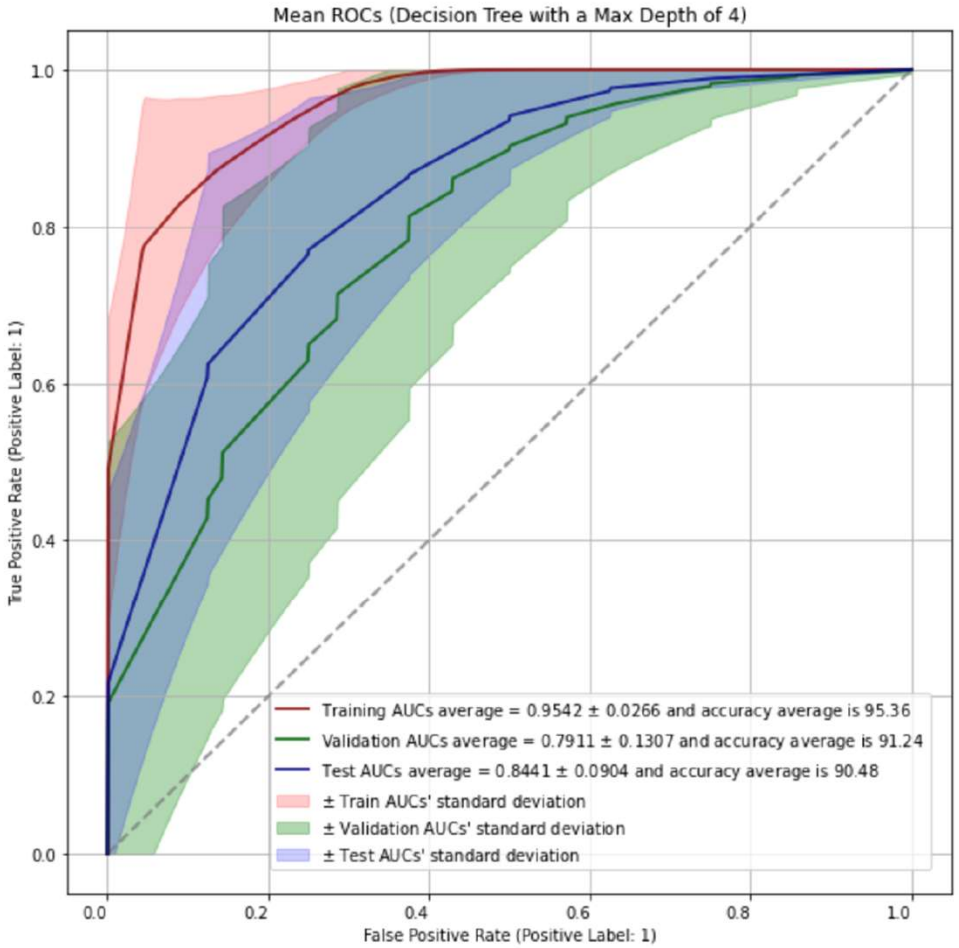


Q12: Approximately how many hours did you sleep on an average work night in January 2020?

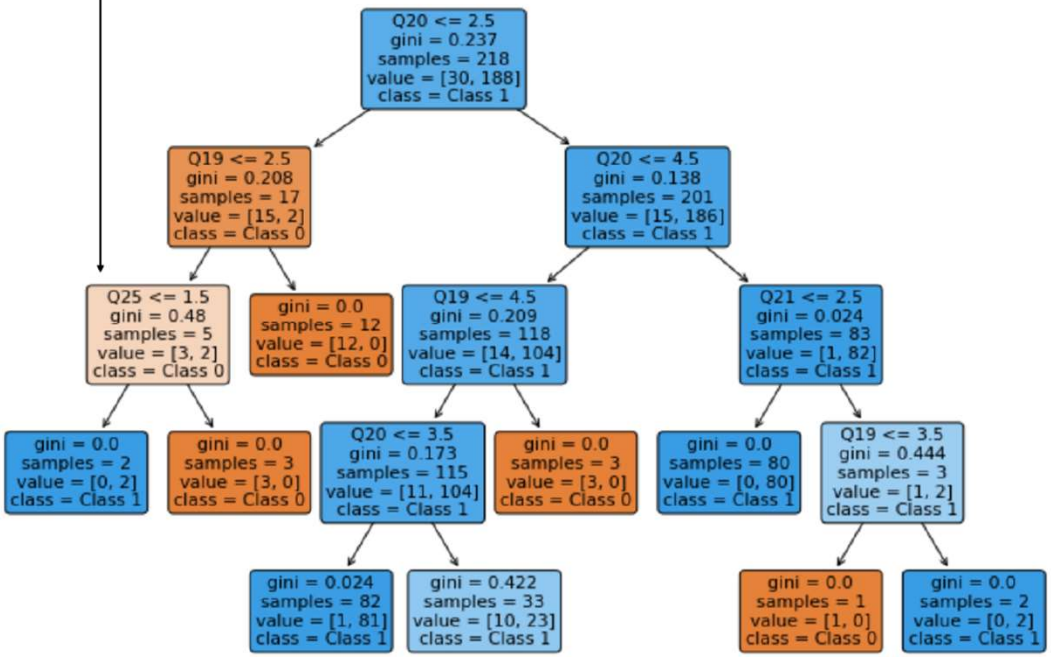
Q21: In January 2020, approximately how often did you use marijuana/cannabis (recreational or medical)?



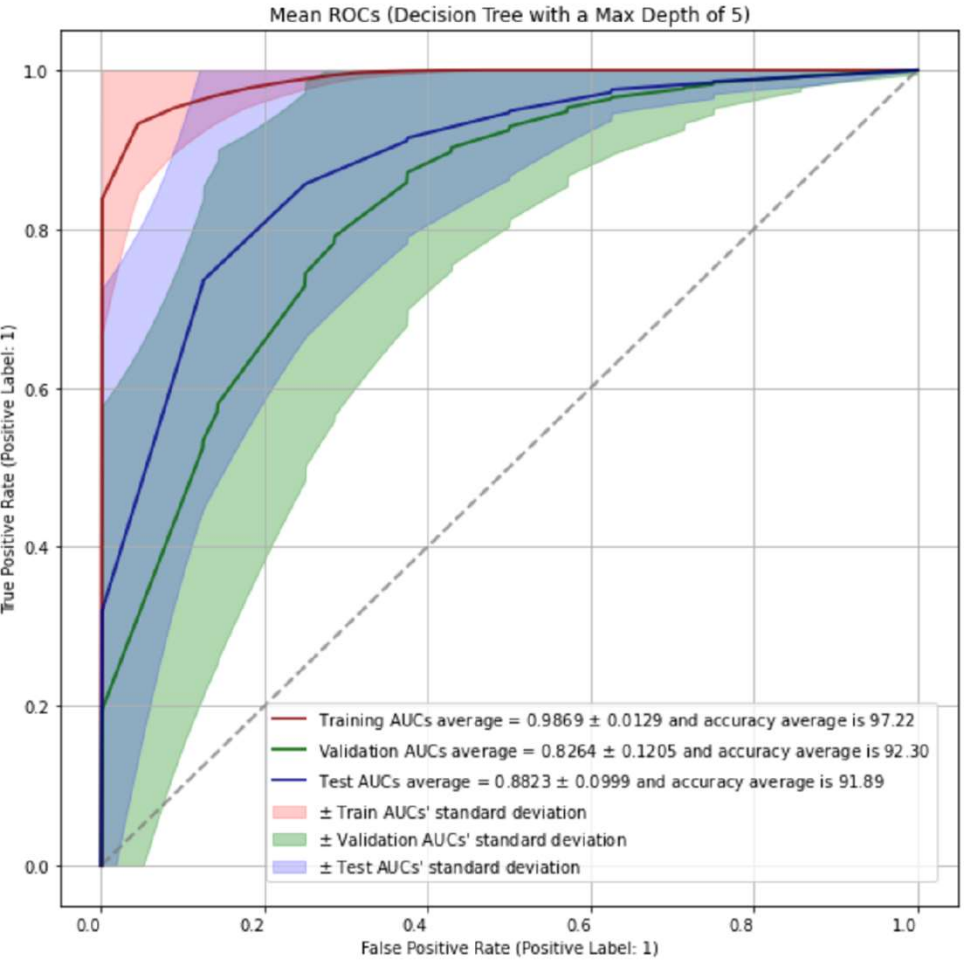
- Decision Tree with a Max Depth of 4

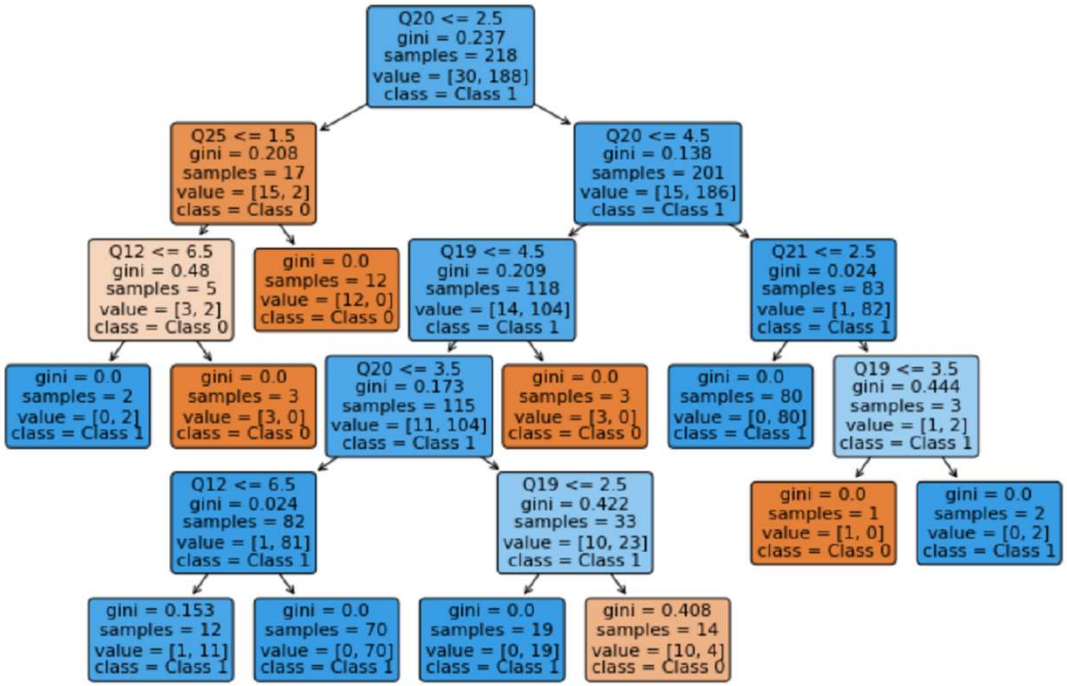


Q25: Have you had more "screen time" (e.g., use of smartphone, tablet, etc.) around bedtime?

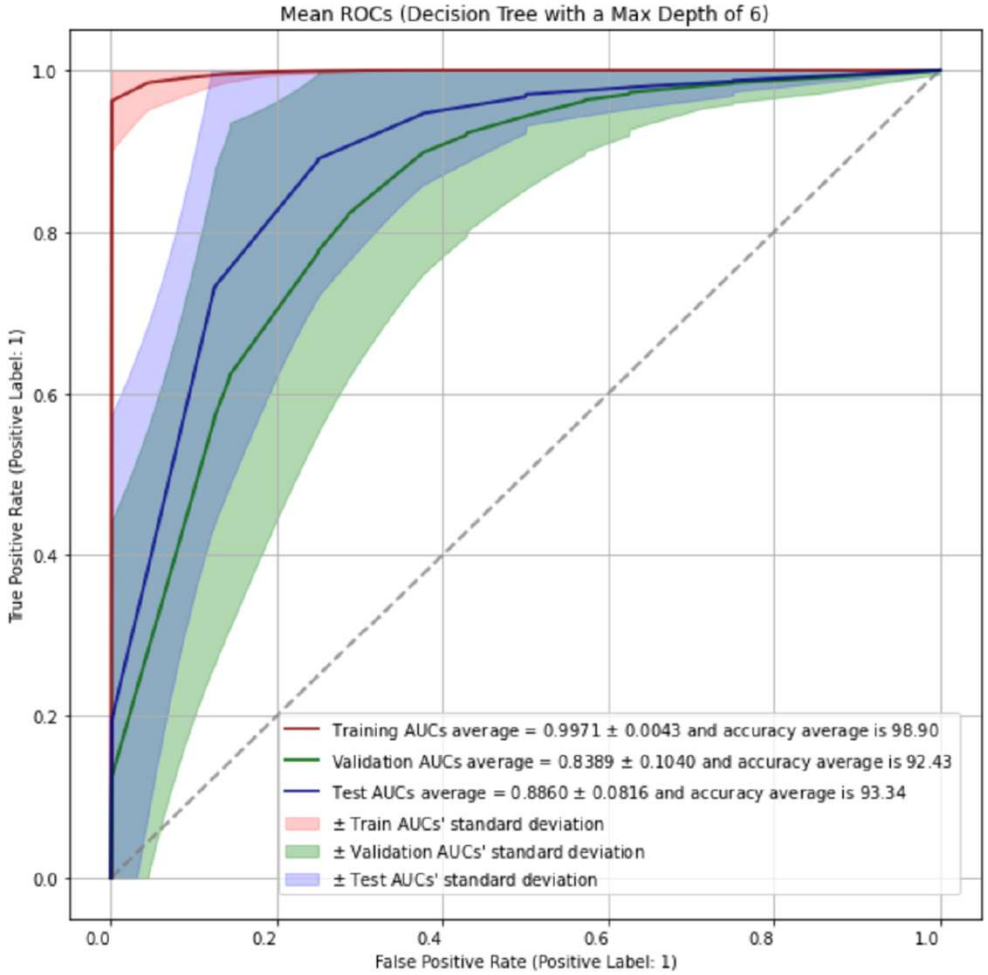


- Decision Tree with a Max Depth of 5

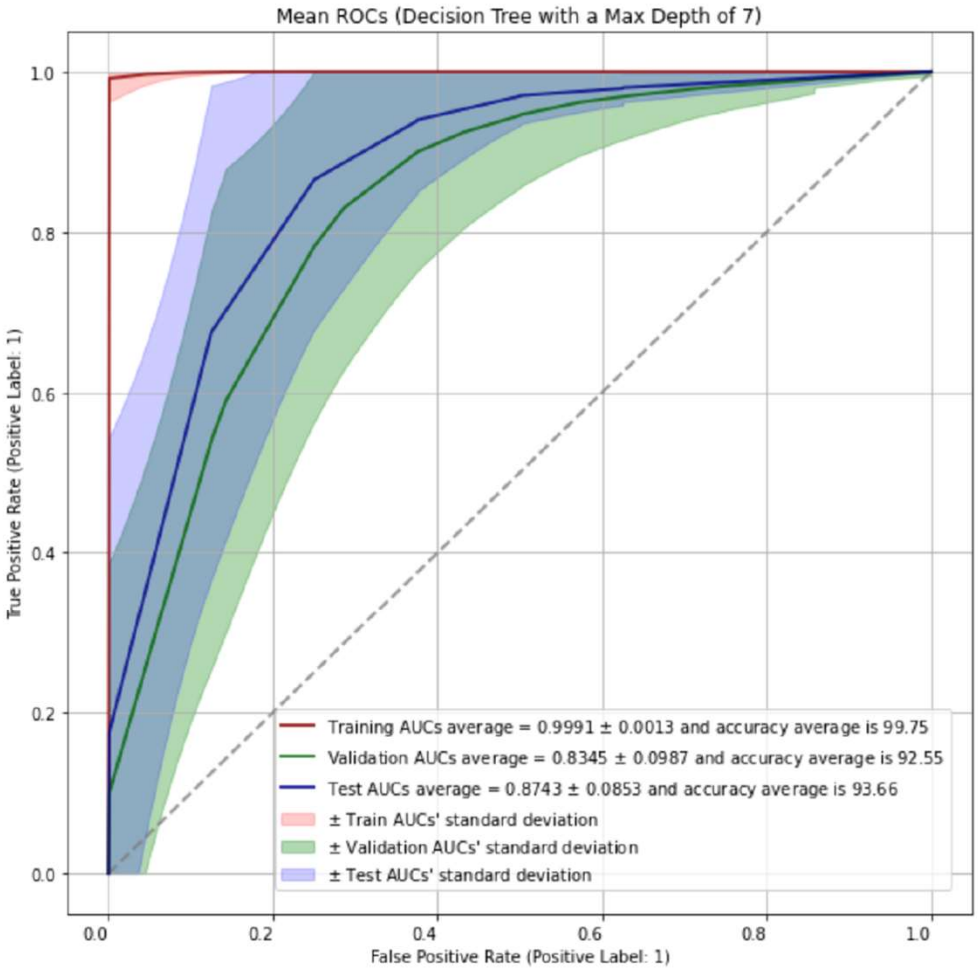


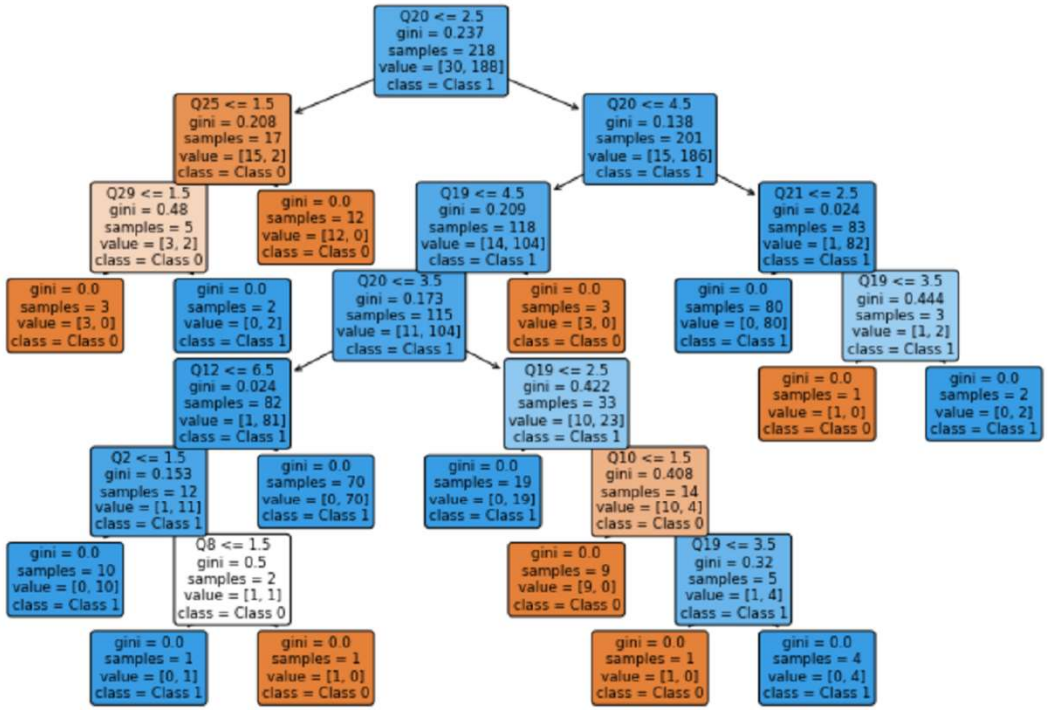


- Decision Tree with a Max Depth of 6

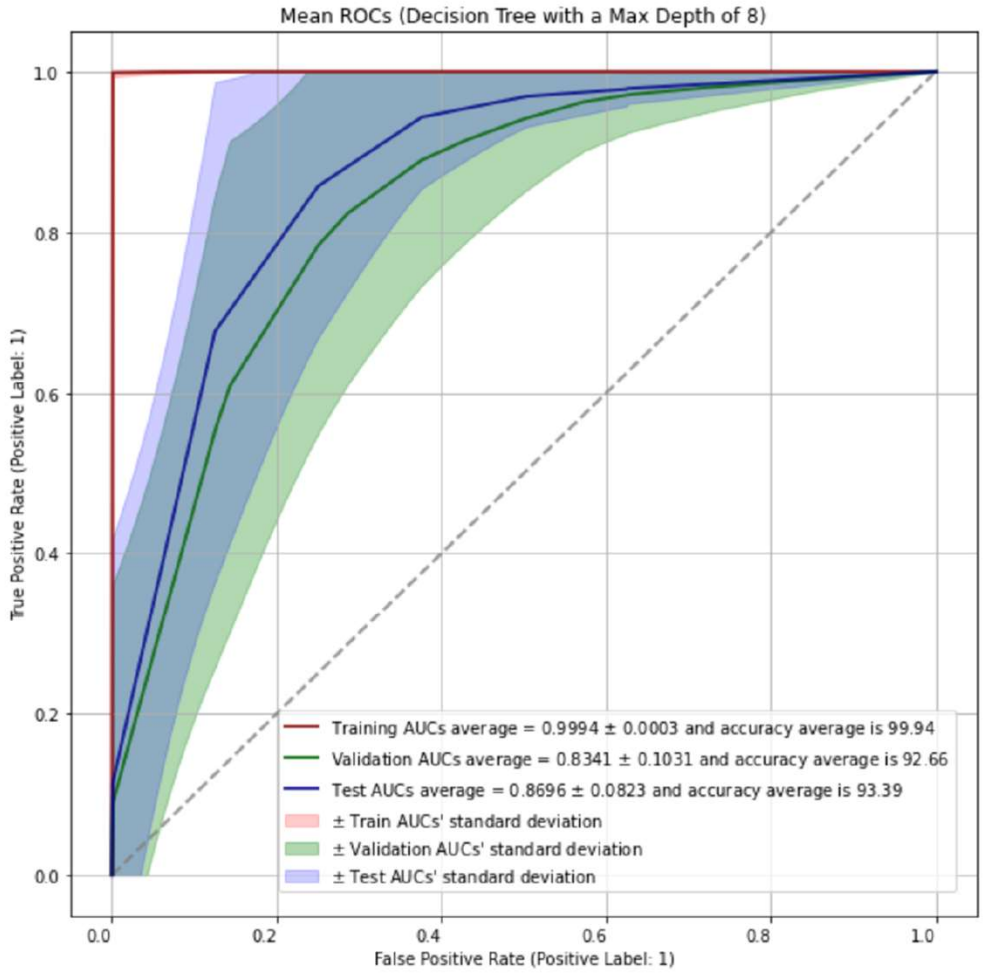


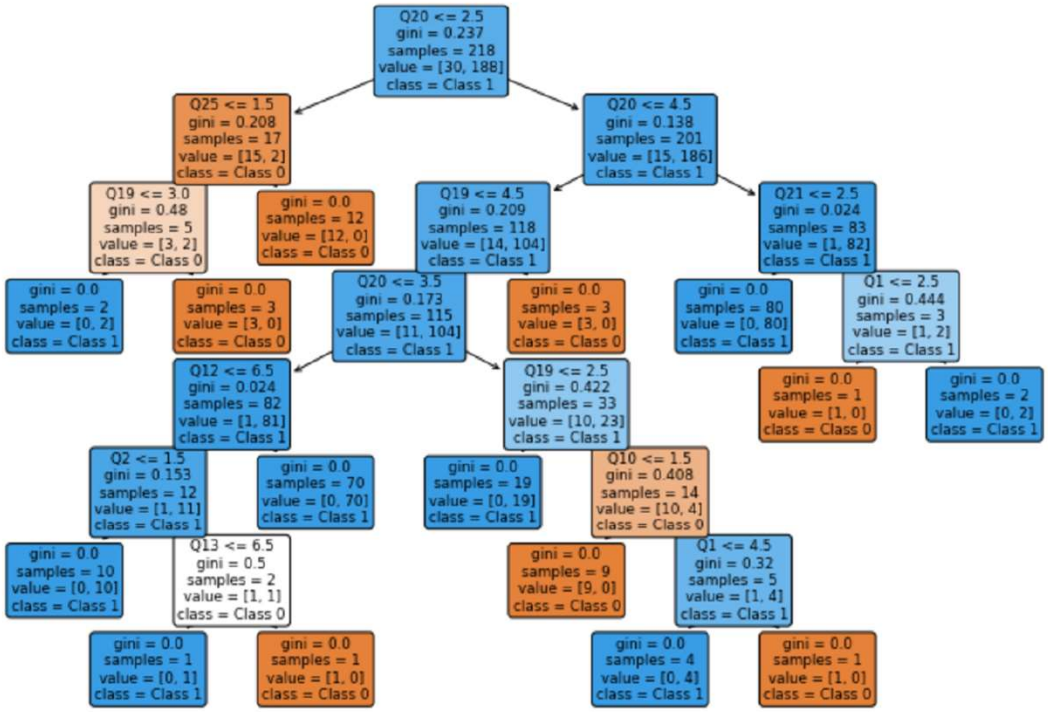
- Decision Tree with a Max Depth of 7



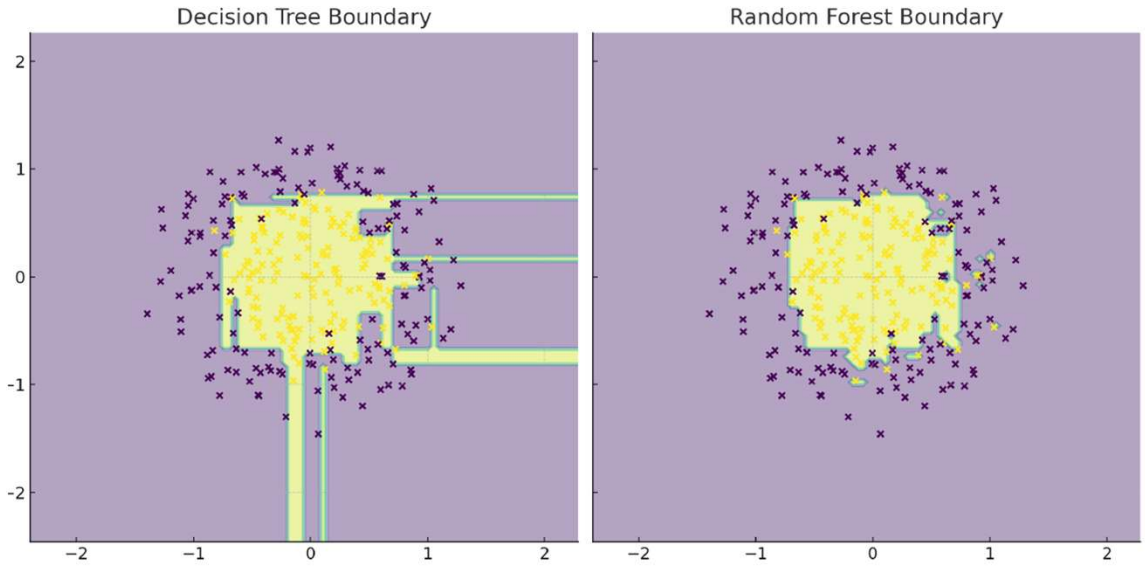


- Decision Tree with a Max Depth of 8

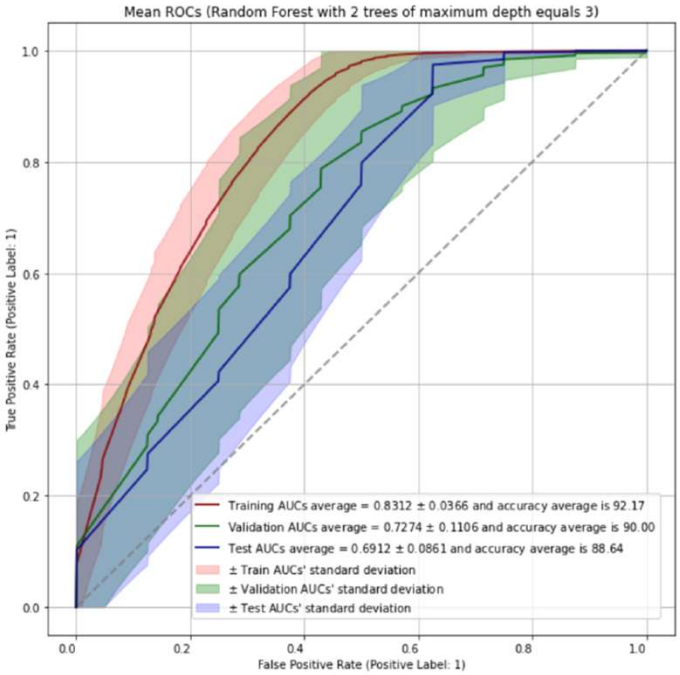


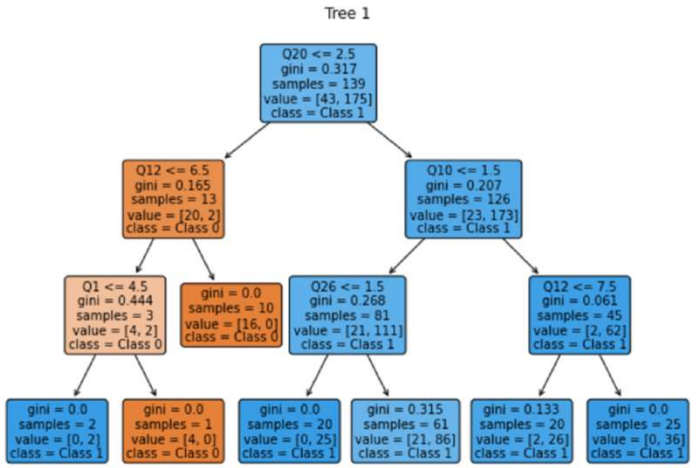


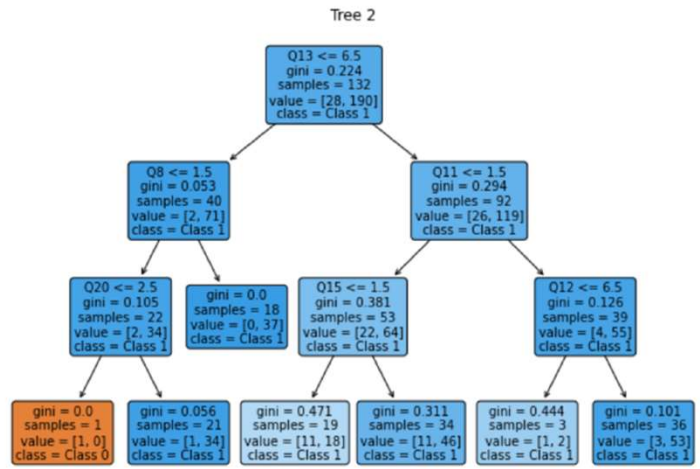
- Random Forest is an ensemble learning method that constructs multiple decision trees for classification or regression tasks.
- It combines the predictions from individual trees to improve accuracy and control overfitting.
- Handles Imbalance: More robust to class imbalance than single trees, leading to fairer predictions.



- Random Forest with 2 trees of maximum depth equals 3



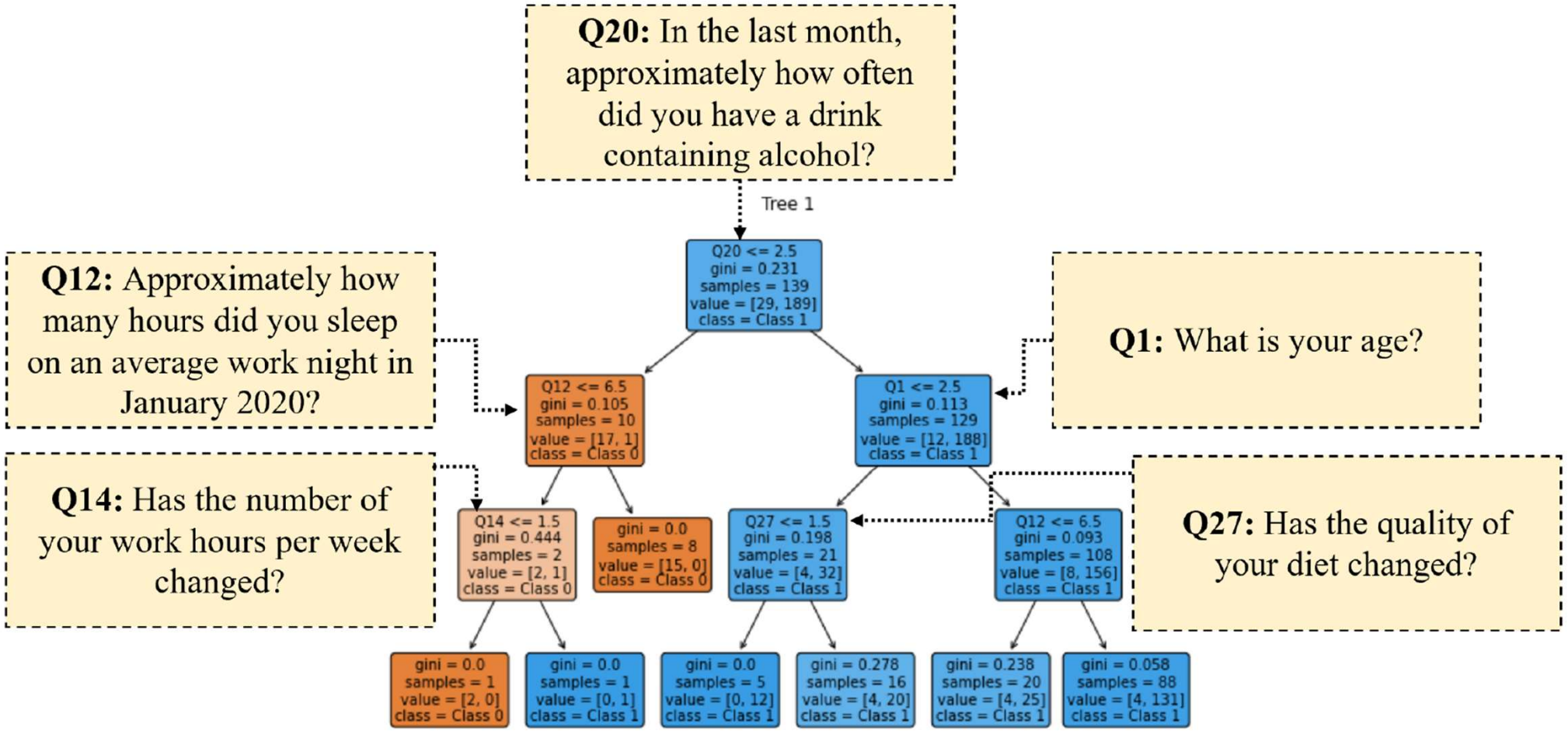




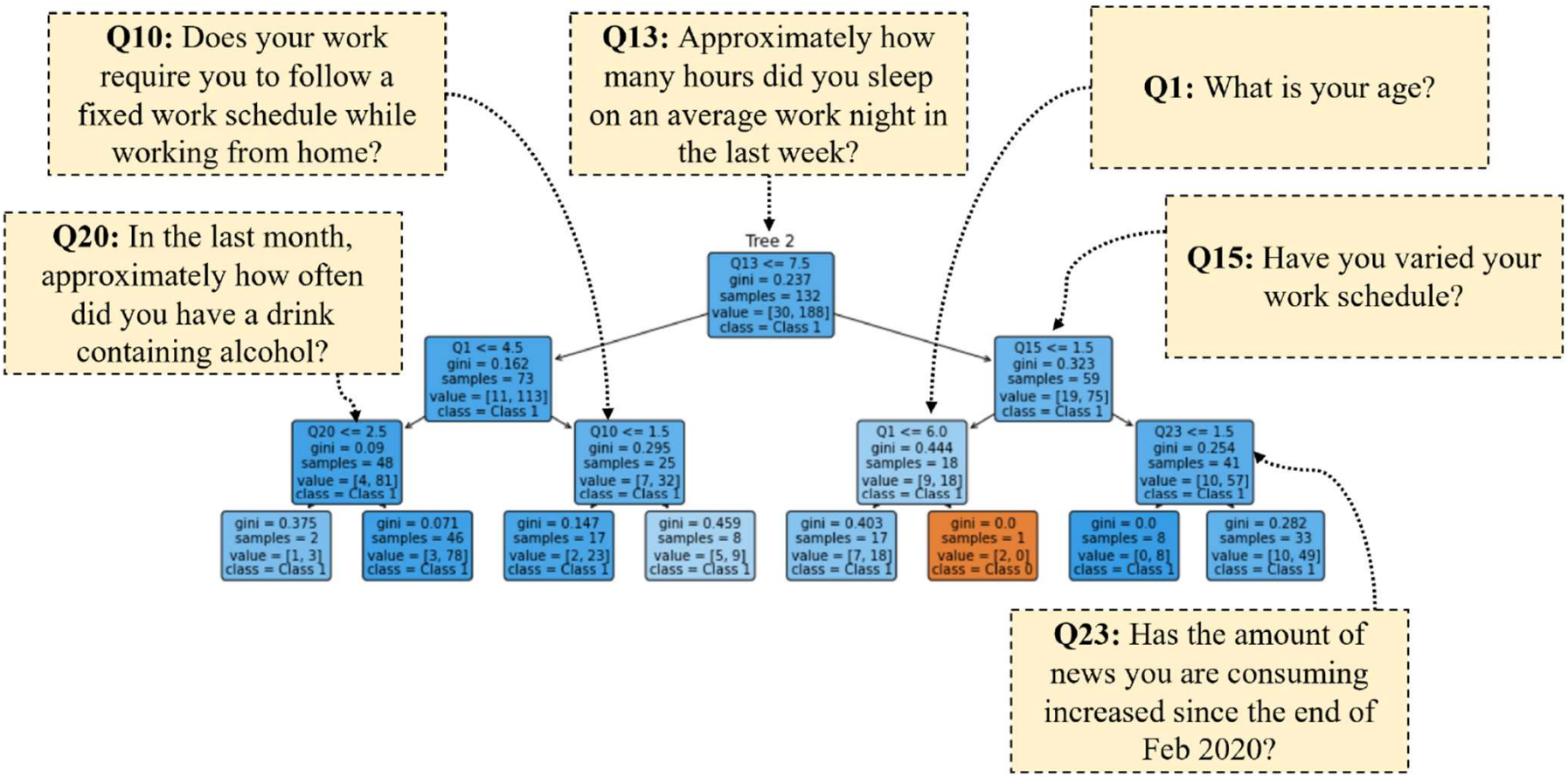
Wake Forest University
School of Medicine

CAIR
CENTER FOR ARTIFICIAL INTELLIGENCE RESEARCH

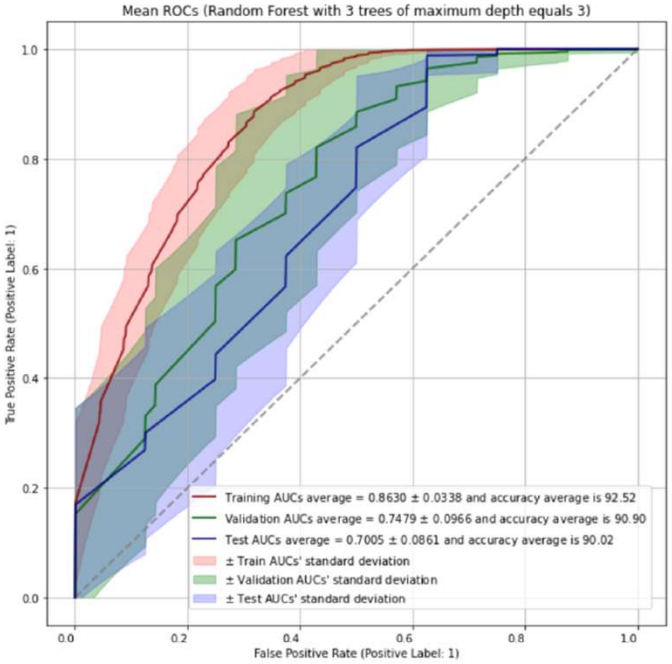
- Random Forest with 2 trees of maximum depth equals 3



- Random Forest with 2 trees of maximum depth equals 3



- Random Forest with 3 trees of maximum depth equals 3



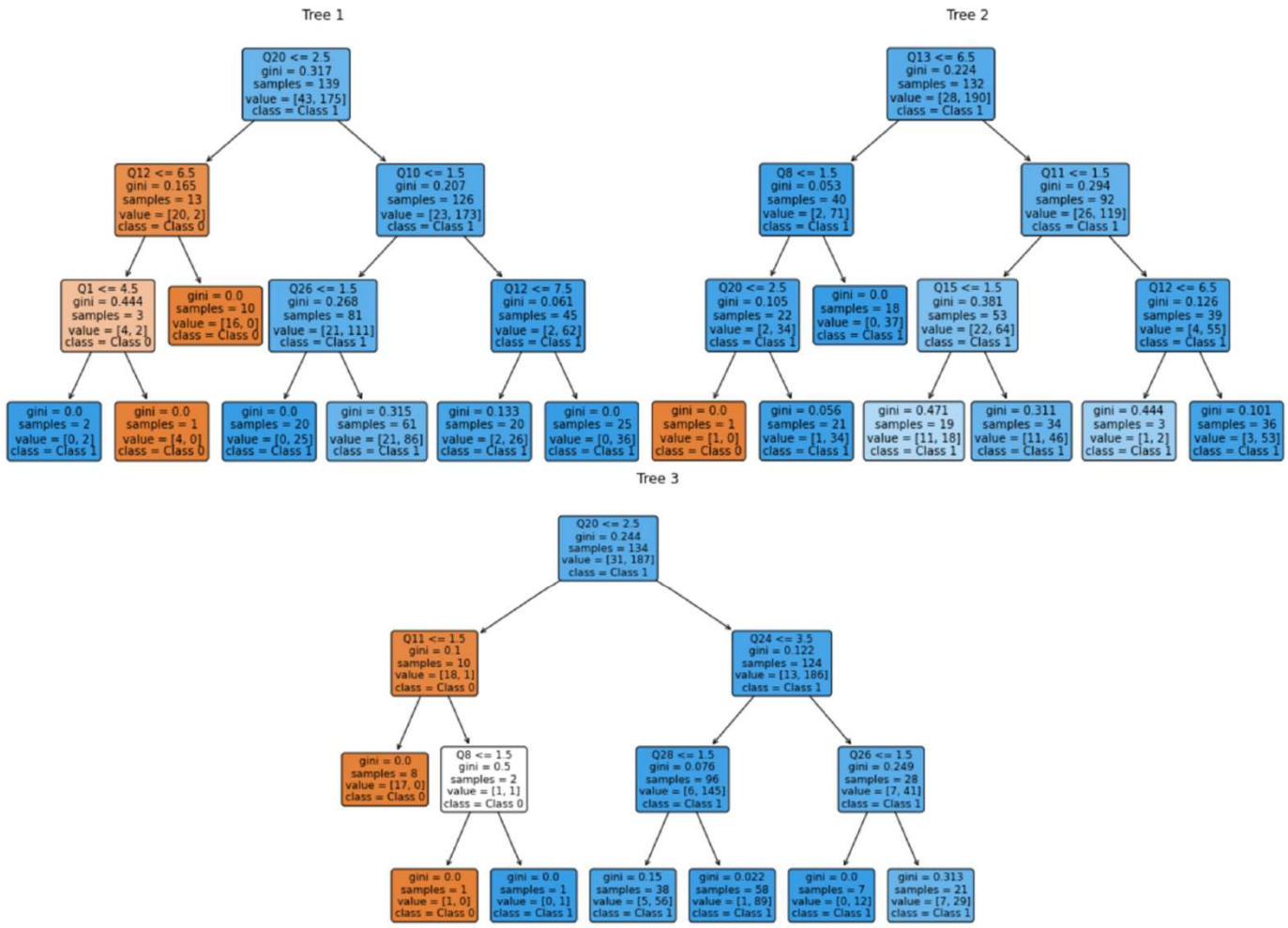


Table 1. The performance of multi-layer perceptron with two hidden layers containing 5 nodes (MLP), support vector machines (SVM), logistic regression (LR), decision tree with maximum depth of 3 (DT-3), decision tree with maximum depth of 4 (DT-4), and decision tree with maximum depth of 5 (DT-5), before and afer SMOTE is applied on training dataset (SMOTE was not applied on Validation or Test sets).

	Model						
Performance	**KNN (before SMOTE)**	**MLP (before SMOTE)**	**SVM (before SMOTE)**	**LR (before SMOTE)**	**DT-3 (before SMOTE)**	**DT-4 (before SMOTE)**	**DT-5 (before SMOTE)**
Train AUC	0.8851	0.9658	0.8184	0.8782	0.9019	0.9767	0.9950
Validation AUC	0.6212	0.6034	0.6912	0.7217	0.7877	0.8828	0.8995
Test AUC	0.5187	0.7127	0.7771	0.8044	0.7271	0.7563	0.7900
Train accuracy	87.33%	95.92%	63.01%	91.37%	94.16%	95.13%	97.92%
Validation accuracy	85.51%	80.31%	59.03%	87.34%	91.92%	91.61%	94.02%
Test accuracy	86.58%	85.65%	63.34%	90.39%	90.41%	90.59%	93.39%
	Model						
Performance	**KNN (after SMOTE)**	**MLP (after SMOTE)**	**SVM (after SMOTE)**	**LR (after MOTE)**	**DT-3 (after MOTE)**	**DT-4 (after MOTE)**	**DT-5 (after SMOTE)**
Train AUC	0.9961	0.9553	0.9141	0.9496	0.9224	0.9647	0.9841
Validation AUC	0.6606	0.6327	0.6826	0.6725	0.7737	0.7820	0.8006
Test AUC	0.5056	0.7128	0.7301	0.7375	0.7807	0.8077	0.8242
Train accuracy	91.50%	91.74%	79.36%	88.38%	85.37%	91.69%	94.80%
Validation accuracy	72.08%	73.41%	71.45%	77.12%	74.06%	83.21%	85.58%
Test accuracy	62.93%	79.30%	76.30%	83.18%	76.61%	85.82%	87.82%

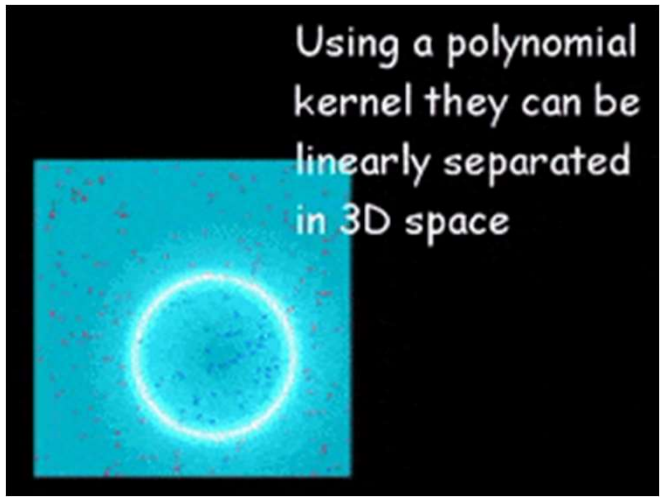
Table 2. The performance of Random Forest with 4 trees of maximum depth equals 3 (RF-4–3), Random Forest with 5 trees of maximum depth equals 3 (RF-5–3), Random Forest with 6 trees of maximum depth equals 3 (RF-6–3), XGBoostClassifer with 4 trees of maximum depth equals 3 (XGB-4–3), LightGBMClassifer with 4 trees of maximum depth equals 3 (LGBM-4–3), and CatBoostClassifer with 4 trees of maximum depth equals 3 (CAT-4–3), before and afer SMOTE is applied on training dataset (SMOTE was not applied on Validation or Test sets).

	Model					
Performance (averages)	**RF-4 (before SMOTE)**	**RF-5 (before SMOTE)**	**RF-6 (before SMOTE)**	**XGB-4 (before SMOTE)**	**LGBM-4 (before SMOTE)**	**CAT-4 (before SMOTE)**
Train AUC	0.8708	0.8763	0.8831	0.9567	0.8740	0.9240
Validation AUC	0.7180	0.7189	0.7246	**0.8895**	0.7767	0.8127
Test AUC	0.7827	0.7794	0.7710	**0.8458**	0.8011	0.8973
Train accuracy	92.27%	92.07%	91.86%	92.21%	86.24%	92.72%
Validation accuracy	91.04%	90.56%	90.23%	**91.66%**	86.25%	90.88%
Test accuracy	91.27%	90.52%	90.18%	**92.57%**	85.45%	90.68%
	Model					
Performance (averages)	**RF-4 (after SMOTE)**	**RF-5 (after SMOTE)**	**RF-6 (after MOTE)**	**XGB-4 (after MOTE)**	**LGBM-4 (after MOTE)**	**CAT-4 (after SMOTE)**
Train AUC	0.9448	0.9587	0.9624	0.9767	0.9586	0.9836
Validation AUC	0.7277	0.7513	0.7605	**0.8471**	0.8299	0.8428
Test AUC	0.7447	0.7844	0.7828	**0.9109**	0.8901	0.8902
Train accuracy	87.30%	88.47%	89.16%	92.06%	89.40%	93.54%
Validation accuracy	79.07%	83.57%	82.98%	85.06%	81.95%	85.69%
Test accuracy	79.98%	84.45%	82.91%	88.61%	85.77%	89.75%

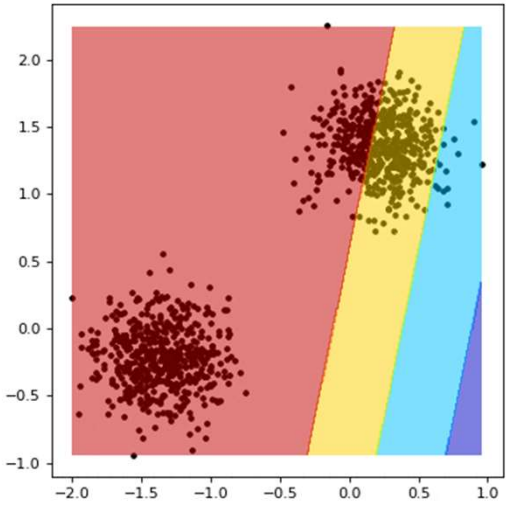
Brief Introduction to other ML models

Support Vector Machines (SVM):

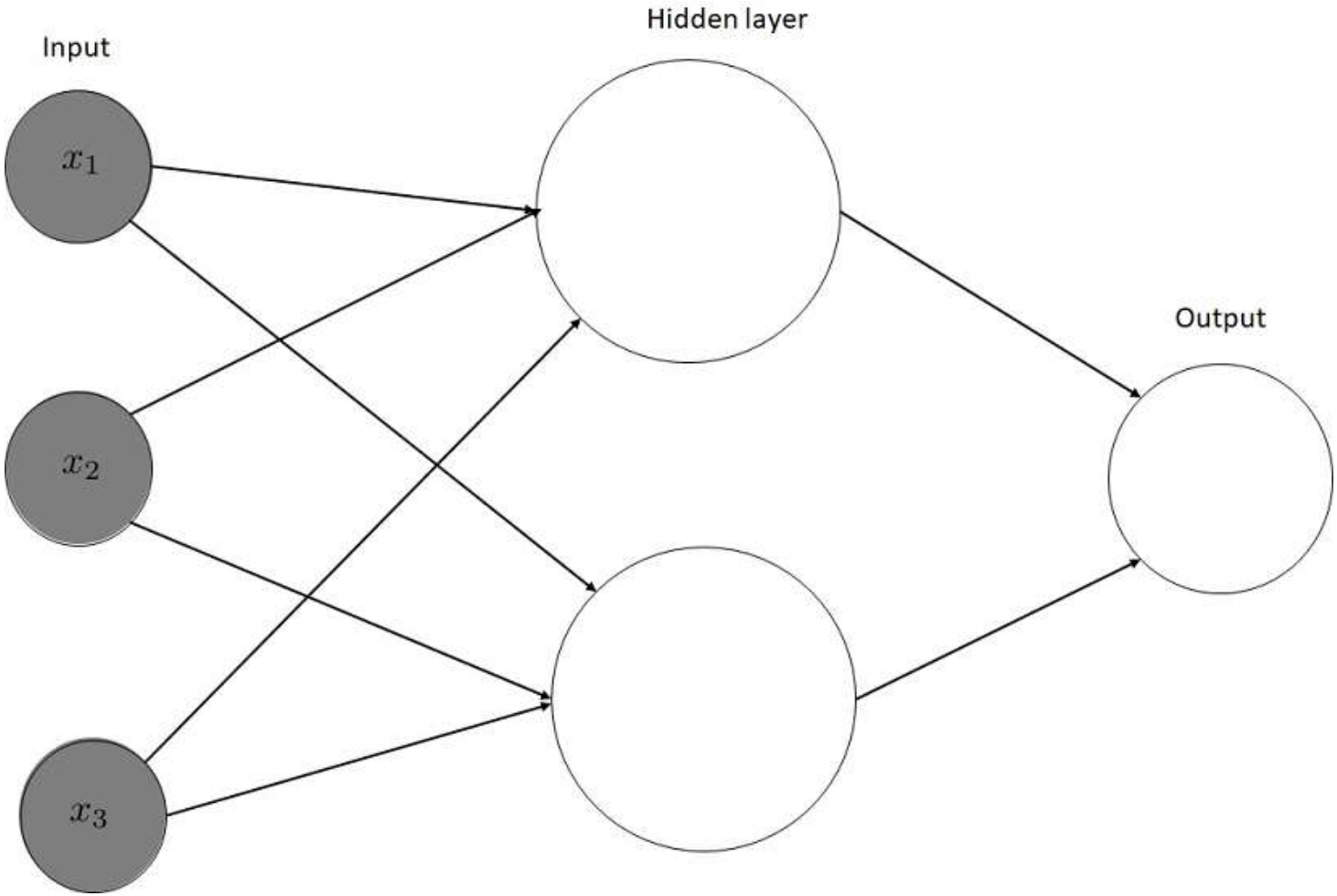
- SVM aims to find the best hyperplane that separates the data points of different classes with the largest possible margin.
- It is a binary classification algorithm but can be extended to handle multi-class classification problems.
- SVM works by mapping the input data into a higher-dimensional feature space and finding an optimal hyperplane that maximally separates the classes.
- The points closest to the hyperplane are called support vectors, which are crucial in defining the decision boundary.
- SVM can handle both linearly separable and non-linearly separable data through the use of different kernel functions.
- Common kernel functions include linear, polynomial, and radial basis function (RBF).

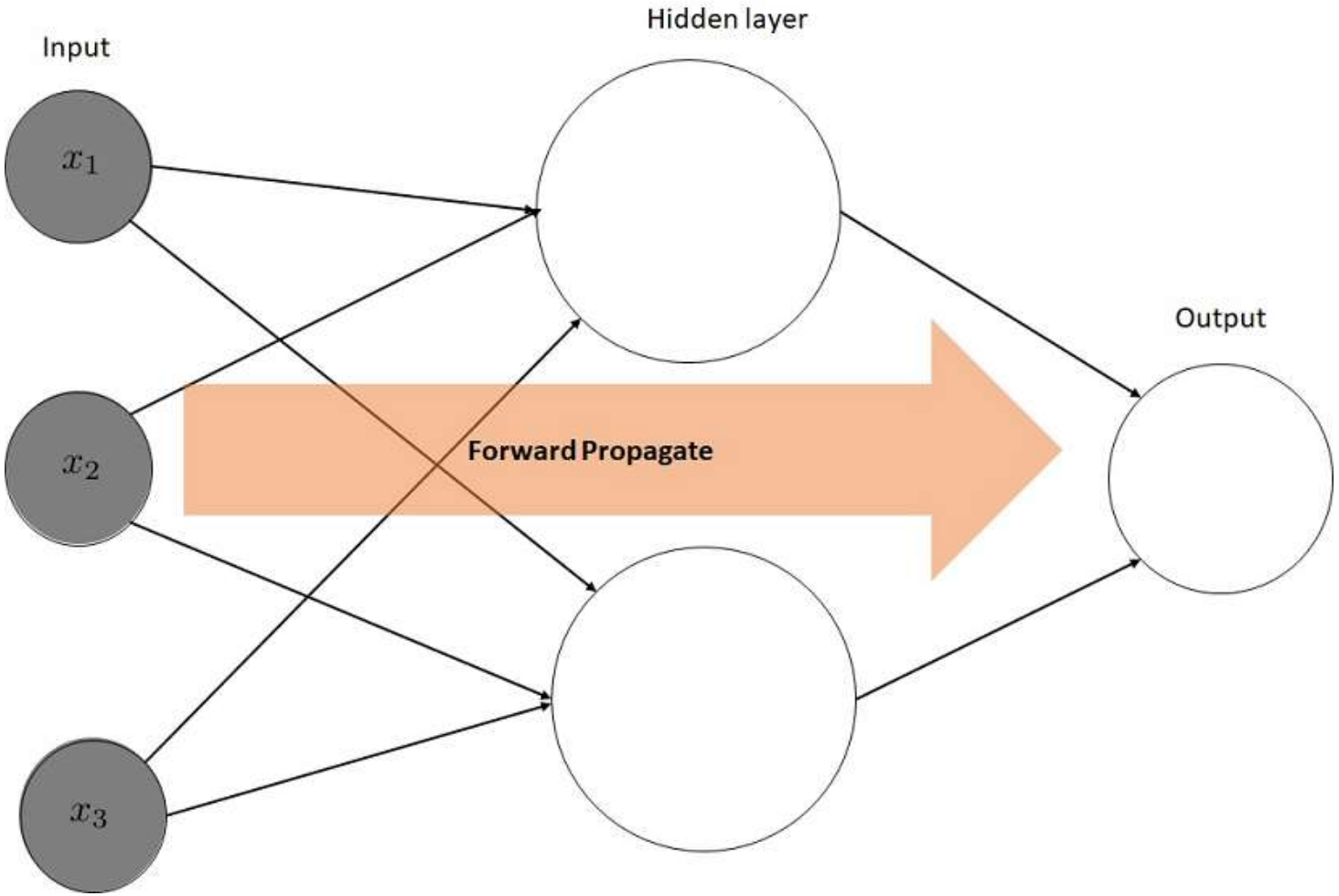


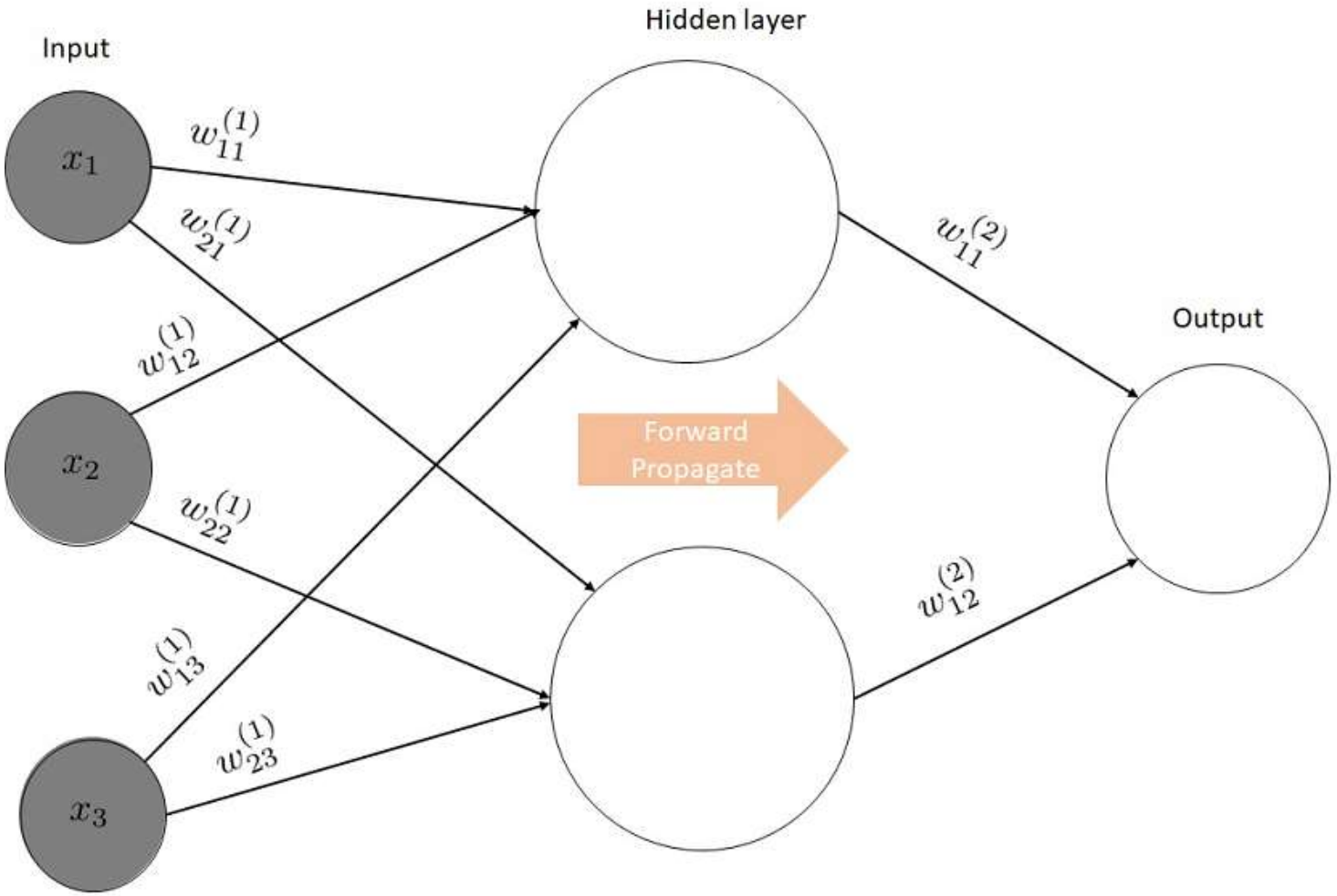
https://gfycat.com/gifs/search/svm

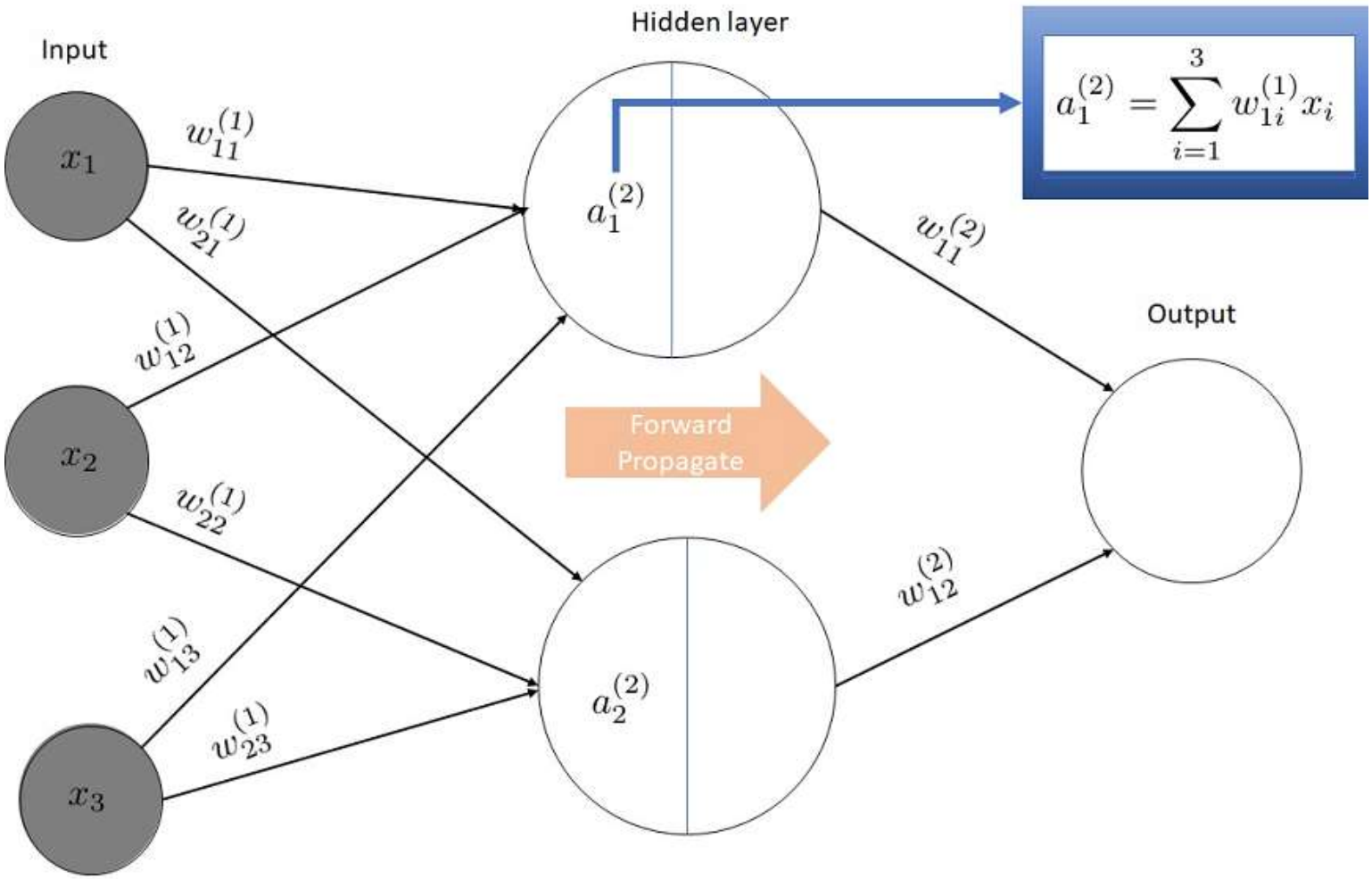


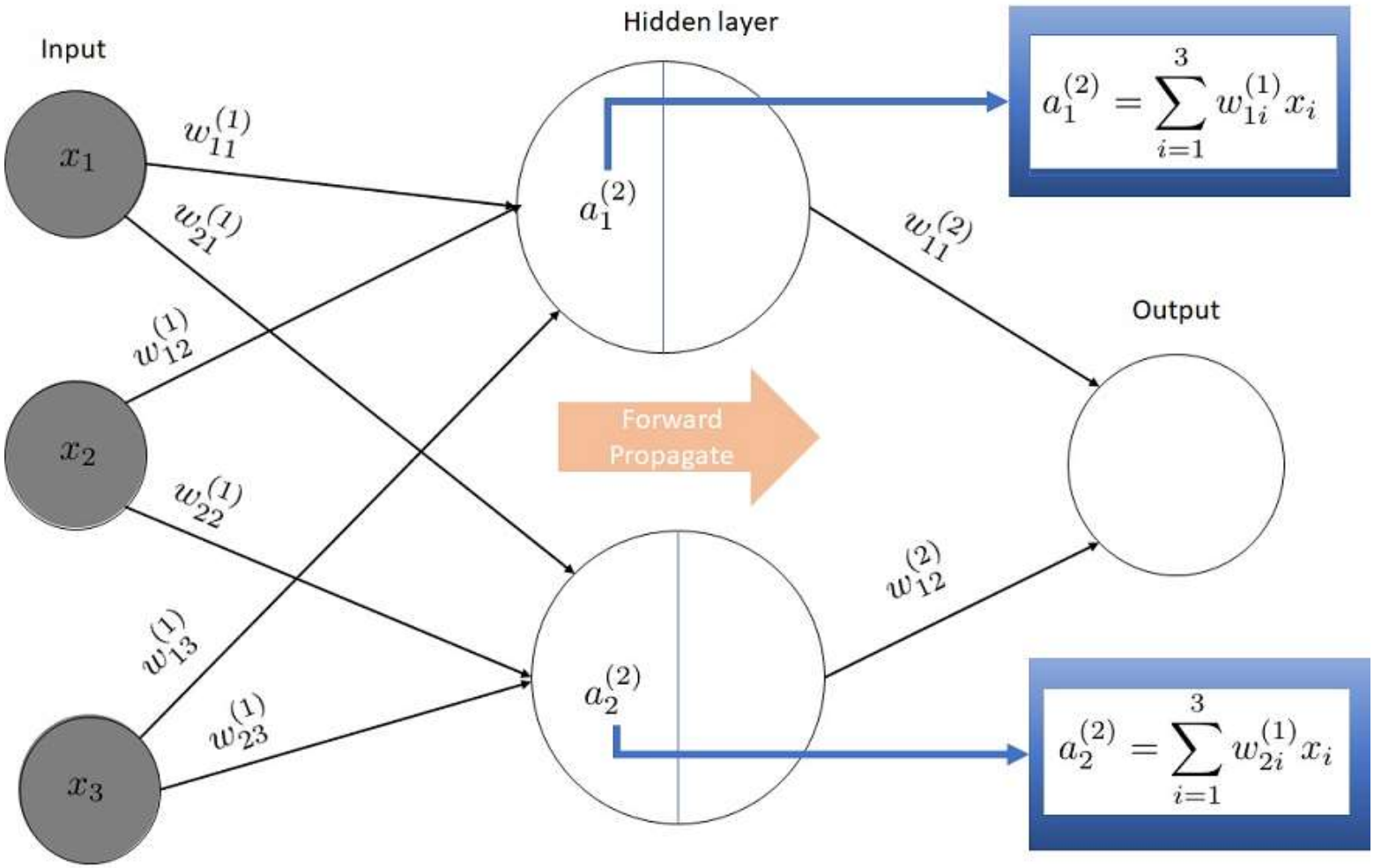
https://torchbearer.readthedocs.io/en/0.1.7/examples/svm_linear.html

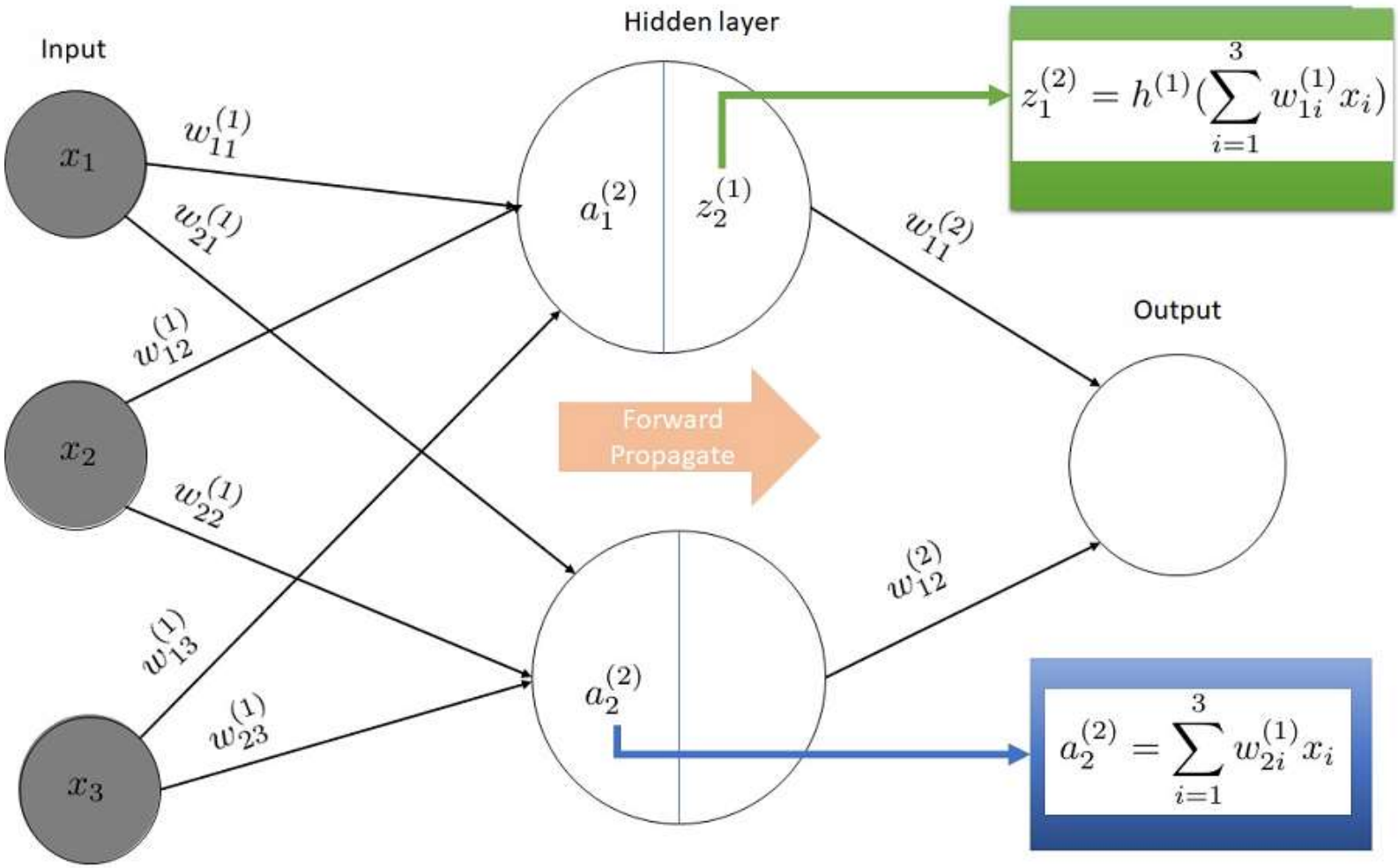

Input
Hidden layer
Output
x_1
x_2
x_3

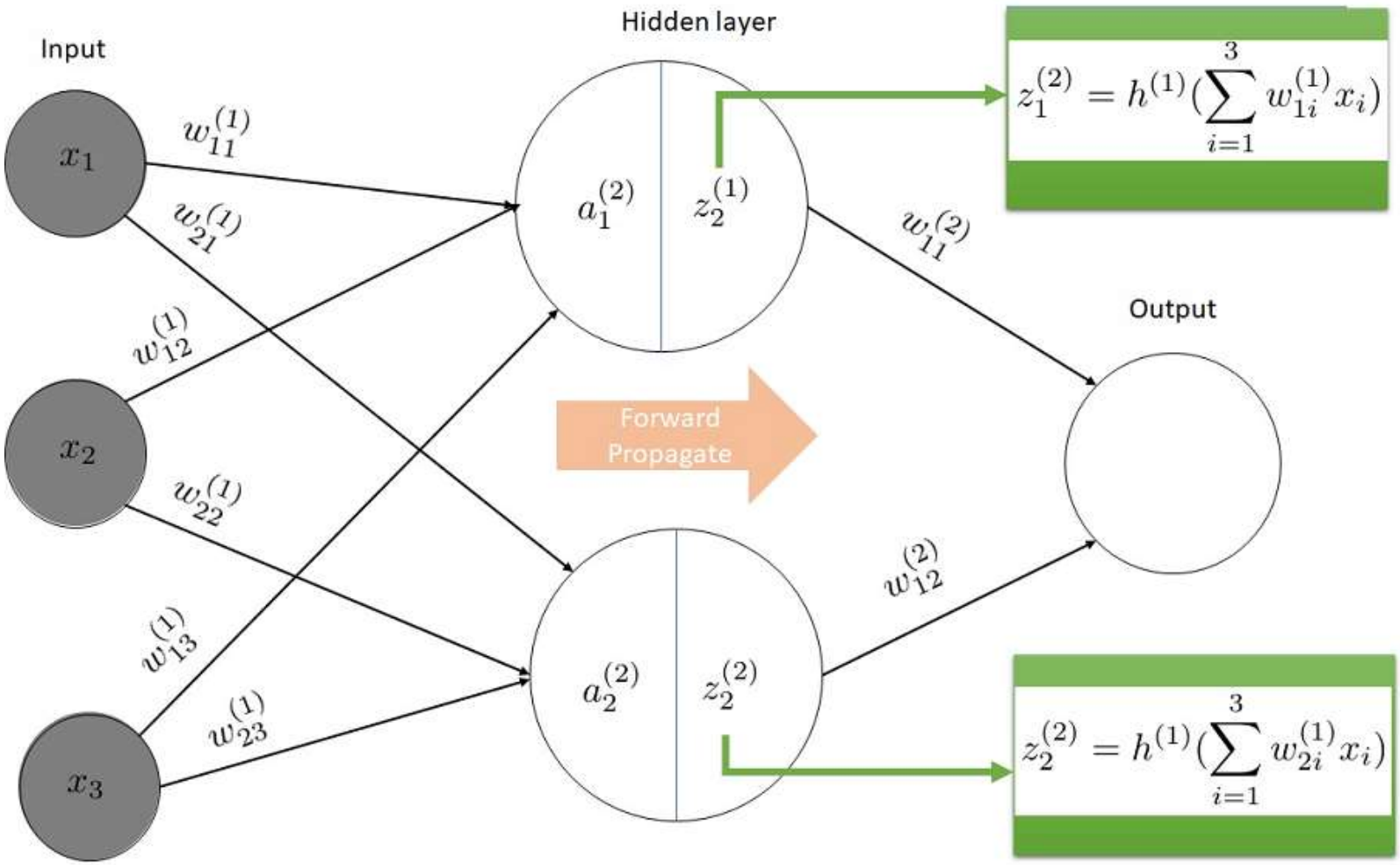

Input
Hidden layer
Output
x_1
x_2
x_3
Forward Propagate

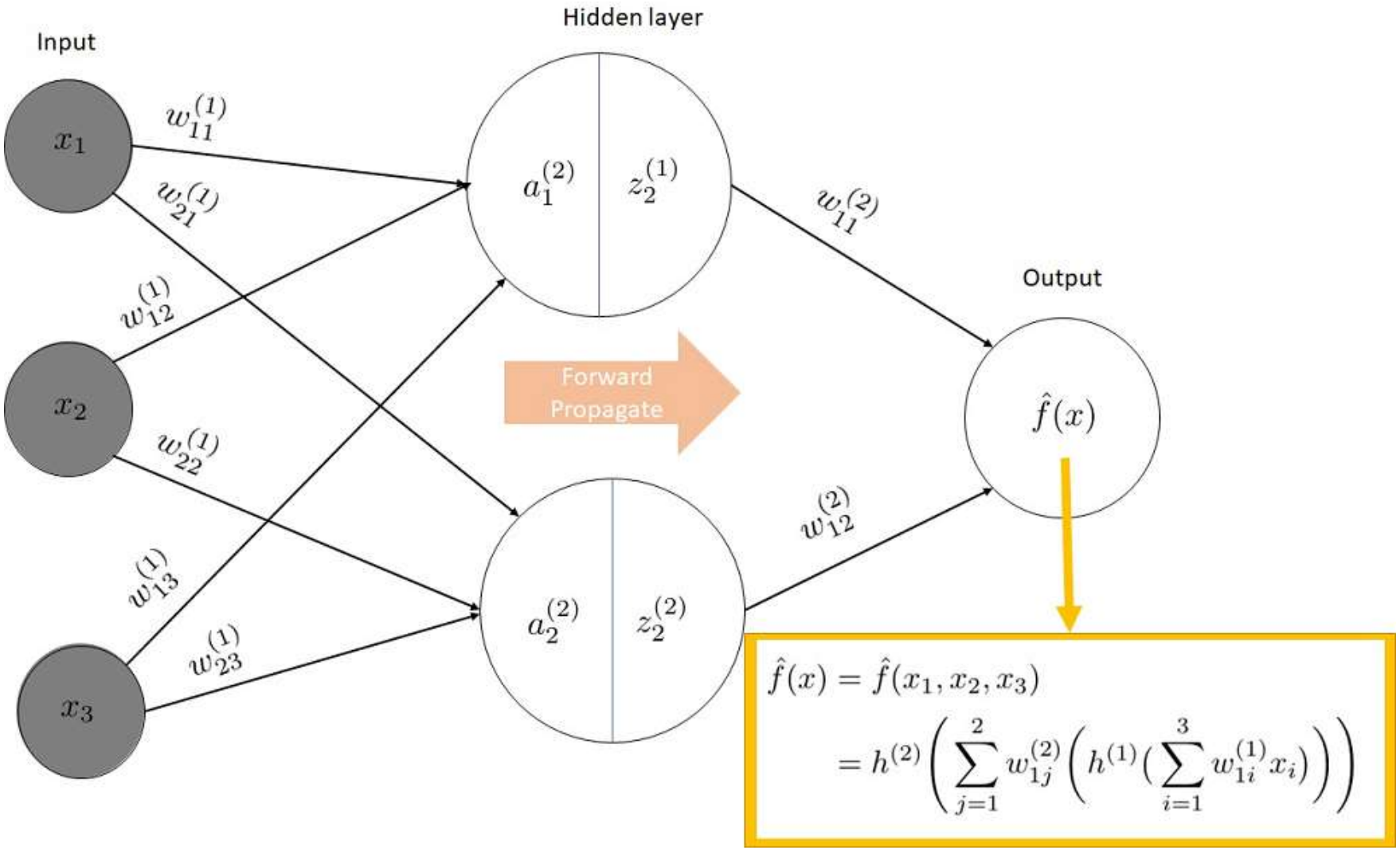

Input
Hidden layer
Output
x_1
x_2
x_3
$w_{11}^{(1)}$
$w_{21}^{(1)}$
$w_{12}^{(1)}$
$w_{22}^{(1)}$
$w_{13}^{(1)}$
$w_{23}^{(1)}$
$w_{11}^{(2)}$
$w_{12}^{(2)}$
Forward Propagate

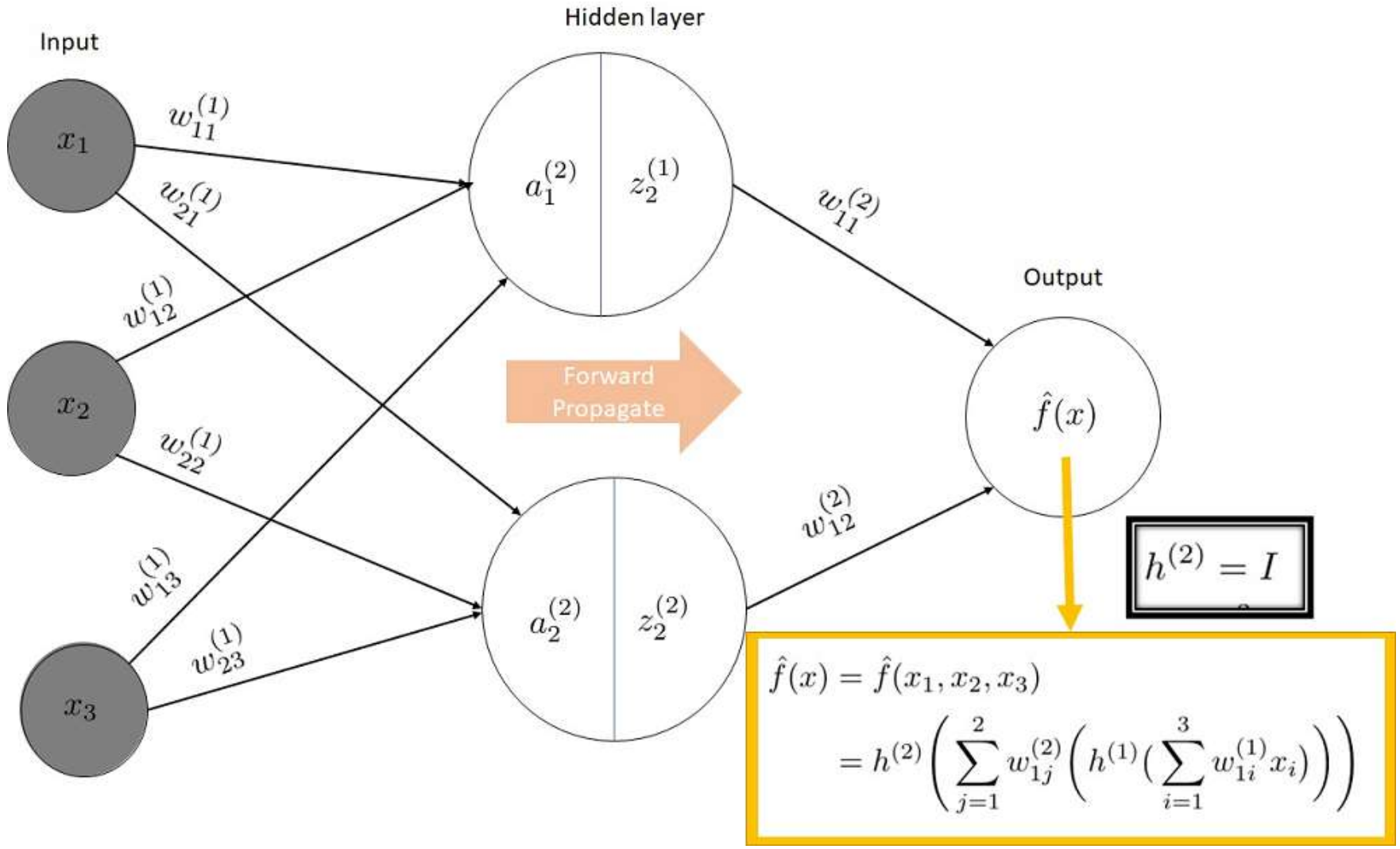

Input
Hidden layer
Output
x_1
x_2
x_3
$w_{11}^{(1)}$
$w_{21}^{(1)}$
$w_{12}^{(1)}$
$w_{22}^{(1)}$
$w_{13}^{(1)}$
$w_{23}^{(1)}$
$a_1^{(2)}$
$a_2^{(2)}$
$w_{11}^{(2)}$
$w_{12}^{(2)}$
Forward Propagate
$a_1^{(2)} = \sum_{i=1}^{3} w_{1i}^{(1)} x_i$

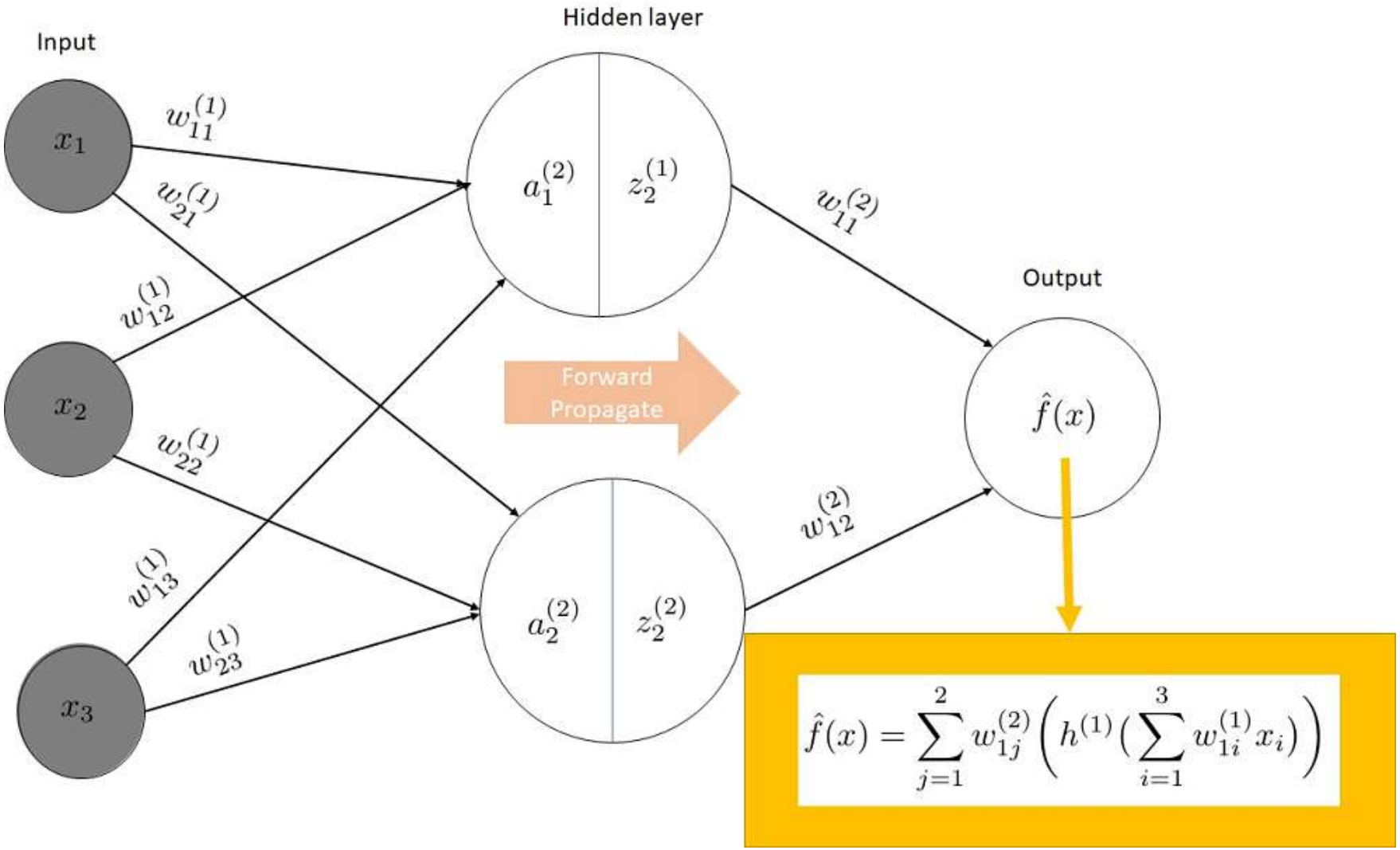

Input
Hidden layer
Output
x_1
x_2
x_3
$w_{11}^{(1)}$
$w_{21}^{(1)}$
$w_{12}^{(1)}$
$w_{22}^{(1)}$
$w_{13}^{(1)}$
$w_{23}^{(1)}$
$a_1^{(2)}$
$a_2^{(2)}$
$w_{11}^{(2)}$
$w_{12}^{(2)}$
Forward Propagate
$a_1^{(2)} = \sum_{i=1}^{3} w_{1i}^{(1)} x_i$
$a_2^{(2)} = \sum_{i=1}^{3} w_{2i}^{(1)} x_i$

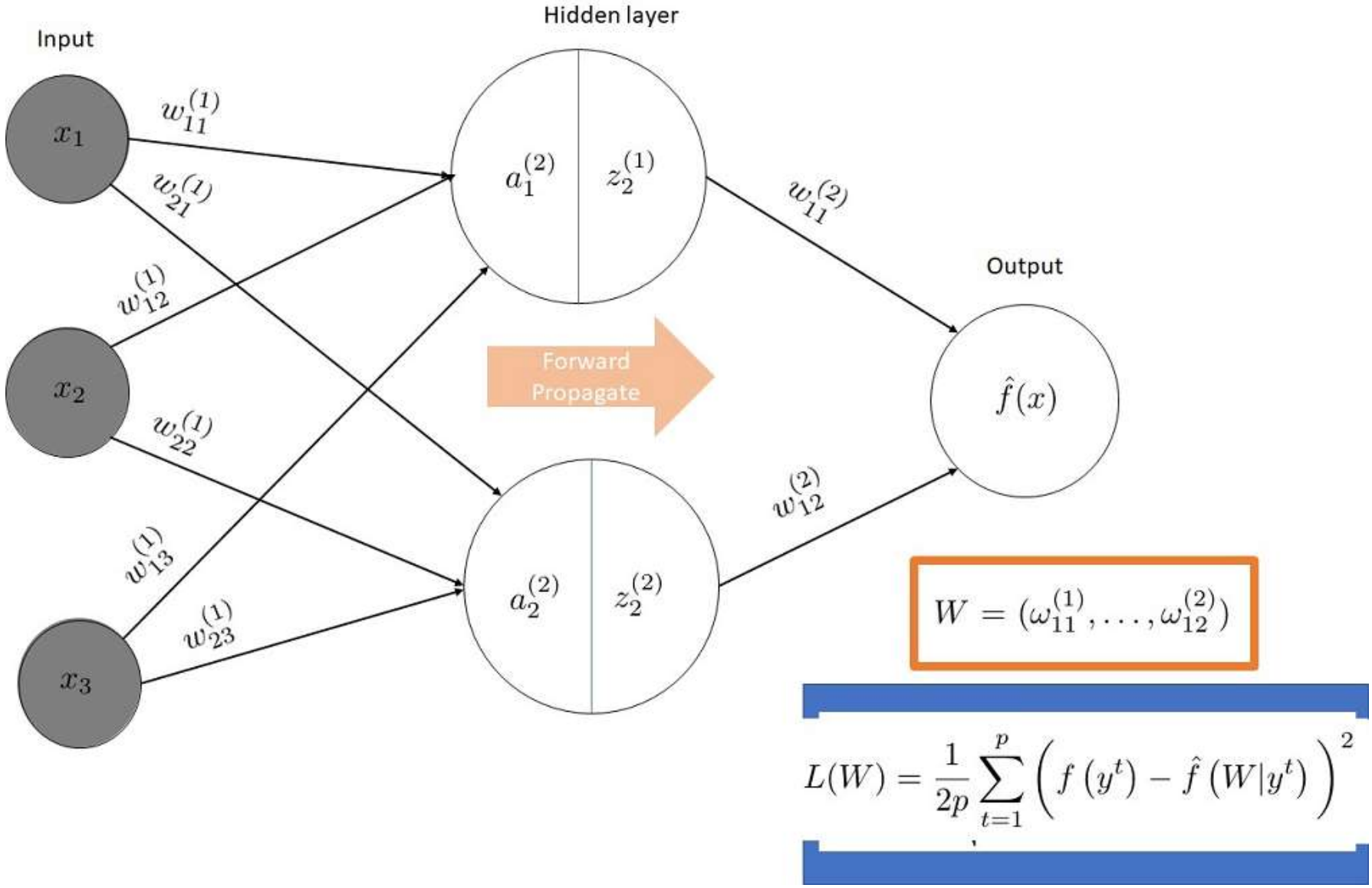

Input
Hidden layer
Output
x_1
x_2
x_3
$w_{11}^{(1)}$
$w_{21}^{(1)}$
$w_{12}^{(1)}$
$w_{22}^{(1)}$
$w_{13}^{(1)}$
$w_{23}^{(1)}$
$a_1^{(2)}$
$z_2^{(1)}$
$a_2^{(2)}$
$w_{11}^{(2)}$
$w_{12}^{(2)}$
Forward Propagate
$z_1^{(2)} = h^{(1)}(\sum_{i=1}^{3} w_{1i}^{(1)} x_i)$
$a_2^{(2)} = \sum_{i=1}^{3} w_{2i}^{(1)} x_i$

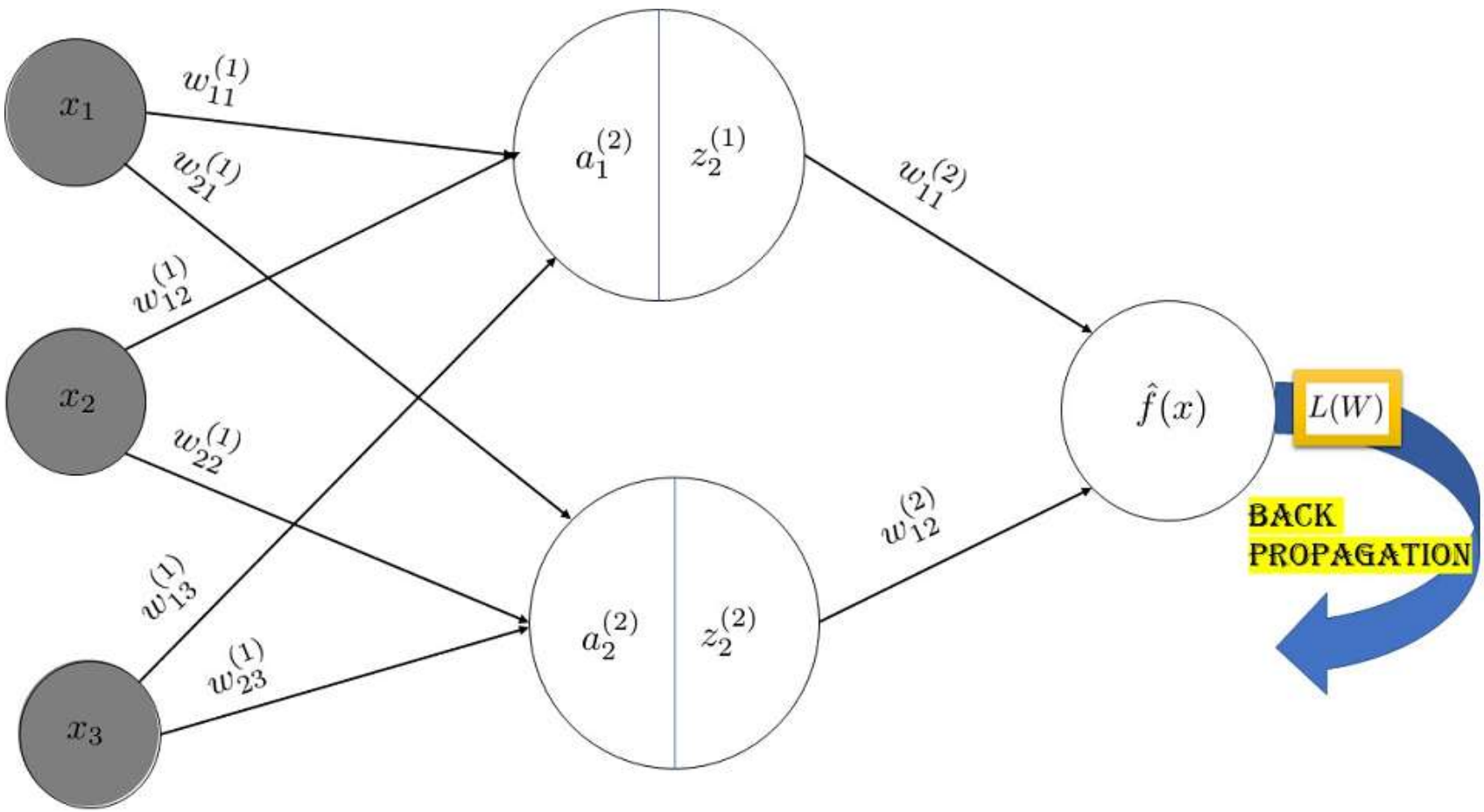

Input
Hidden layer
Output
x_1
x_2
x_3
$w_{11}^{(1)}$
$w_{21}^{(1)}$
$w_{12}^{(1)}$
$w_{22}^{(1)}$
$w_{13}^{(1)}$
$w_{23}^{(1)}$
$a_1^{(2)}$
$z_2^{(1)}$
$a_2^{(2)}$
$z_2^{(2)}$
$w_{11}^{(2)}$
$w_{12}^{(2)}$
Forward Propagate
$z_1^{(2)} = h^{(1)}(\sum_{i=1}^{3} w_{1i}^{(1)} x_i)$
$z_2^{(2)} = h^{(1)}(\sum_{i=1}^{3} w_{2i}^{(1)} x_i)$

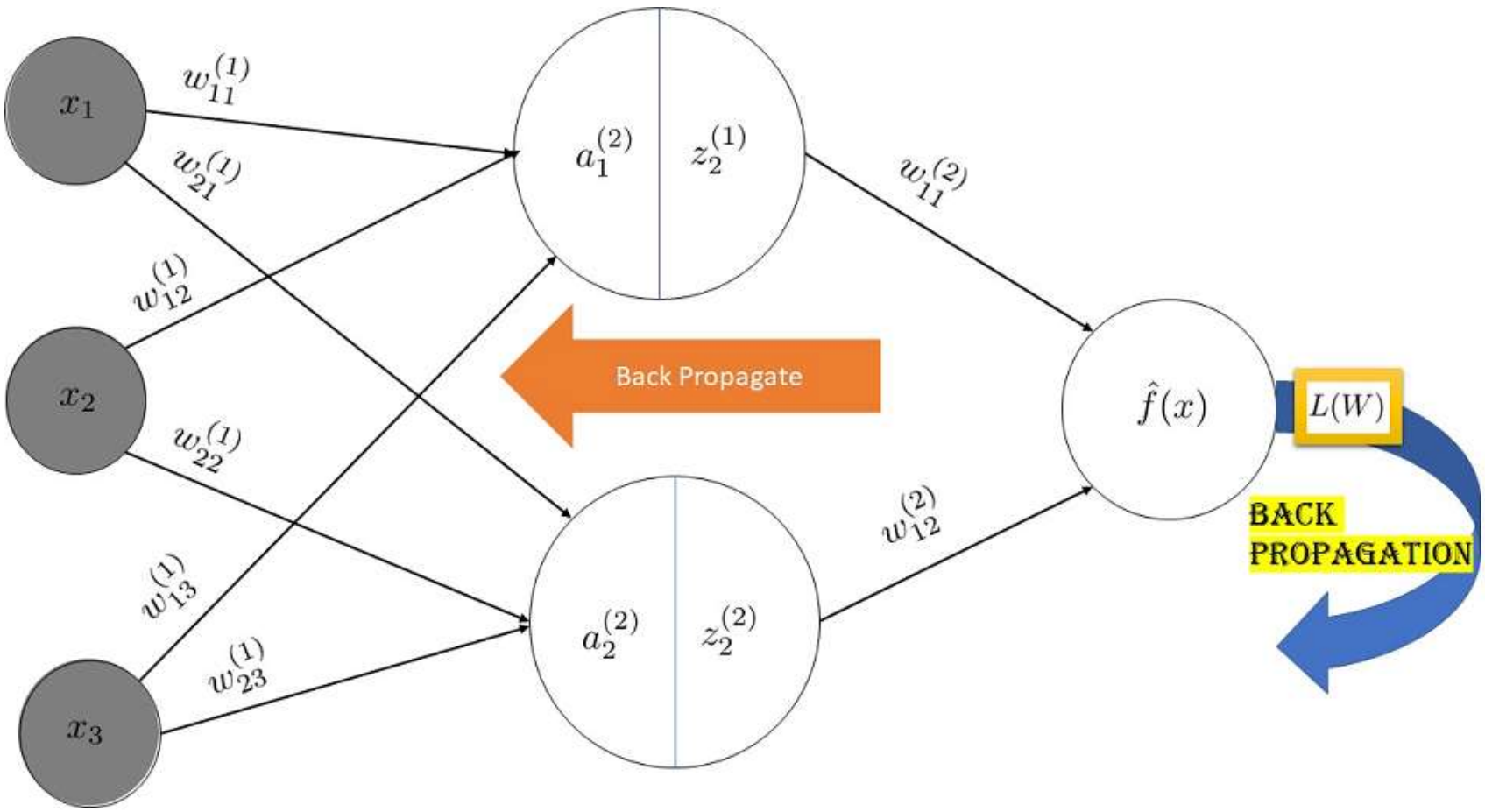

Input
Hidden layer
Output
x_1
x_2
x_3
$w_{11}^{(1)}$
$w_{21}^{(1)}$
$w_{12}^{(1)}$
$w_{22}^{(1)}$
$w_{13}^{(1)}$
$w_{23}^{(1)}$
$a_1^{(2)}$
$z_2^{(1)}$
$a_2^{(2)}$
$z_2^{(2)}$
$w_{11}^{(2)}$
$w_{12}^{(2)}$
Forward Propagate
$\hat{f}(x)$
$\hat{f}(x) = \hat{f}(x_1, x_2, x_3)$
$= h^{(2)}\left(\sum_{j=1}^{2} w_{1j}^{(2)}\left(h^{(1)}\left(\sum_{i=1}^{3} w_{1i}^{(1)} x_i\right)\right)\right)$

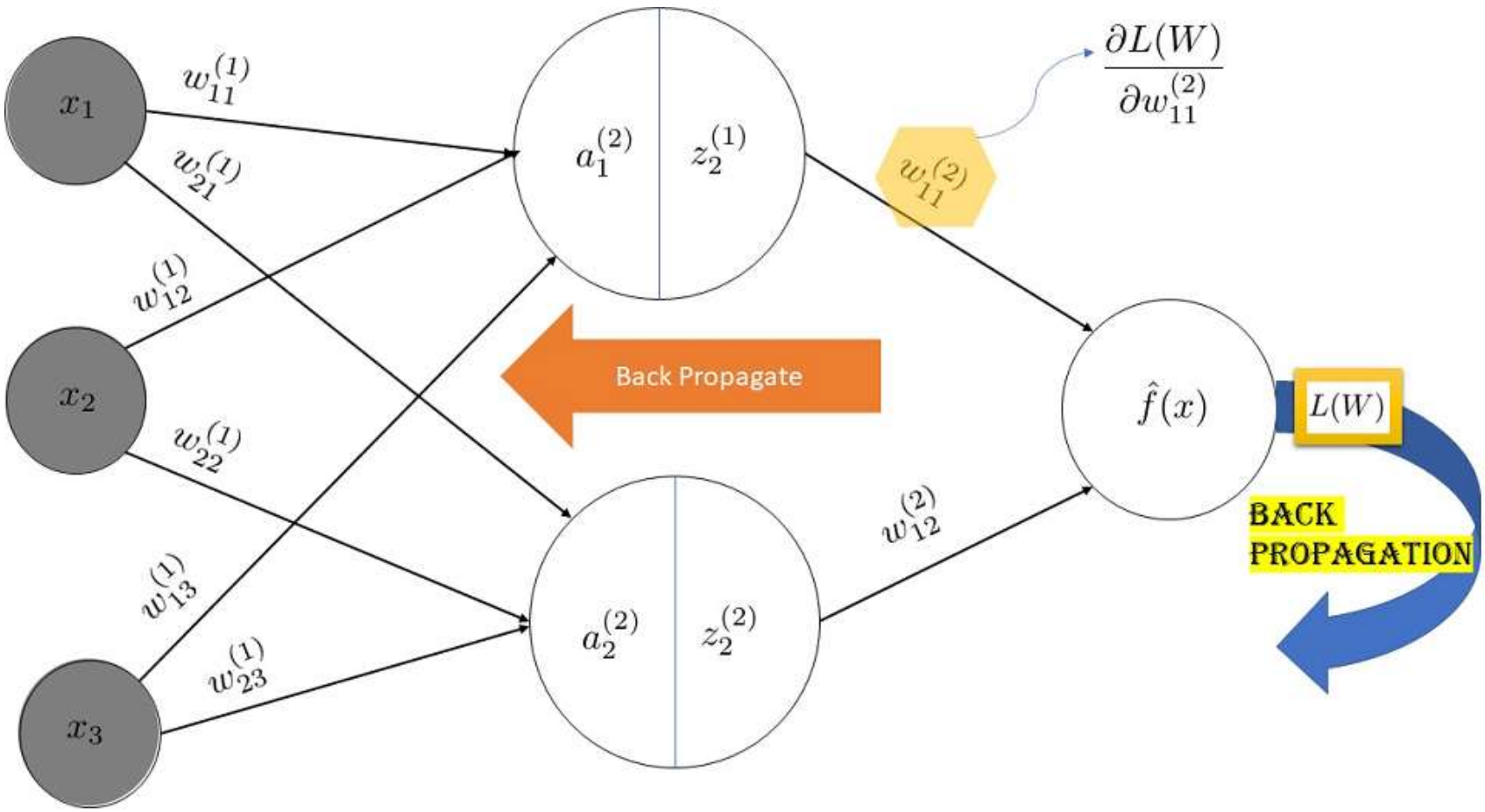

Input
Hidden layer
Output
x_1
x_2
x_3
$w_{11}^{(1)}$
$w_{21}^{(1)}$
$w_{12}^{(1)}$
$w_{22}^{(1)}$
$w_{13}^{(1)}$
$w_{23}^{(1)}$
$a_1^{(2)}$
$z_2^{(1)}$
$a_2^{(2)}$
$z_2^{(2)}$
$w_{11}^{(2)}$
$w_{12}^{(2)}$
Forward Propagate
$\hat{f}(x)$
$h^{(2)} = I$
$\hat{f}(x) = \hat{f}(x_1, x_2, x_3)$
$= h^{(2)}\left(\sum_{j=1}^{2} w_{1j}^{(2)}\left(h^{(1)}(\sum_{i=1}^{3} w_{1i}^{(1)} x_i)\right)\right)$

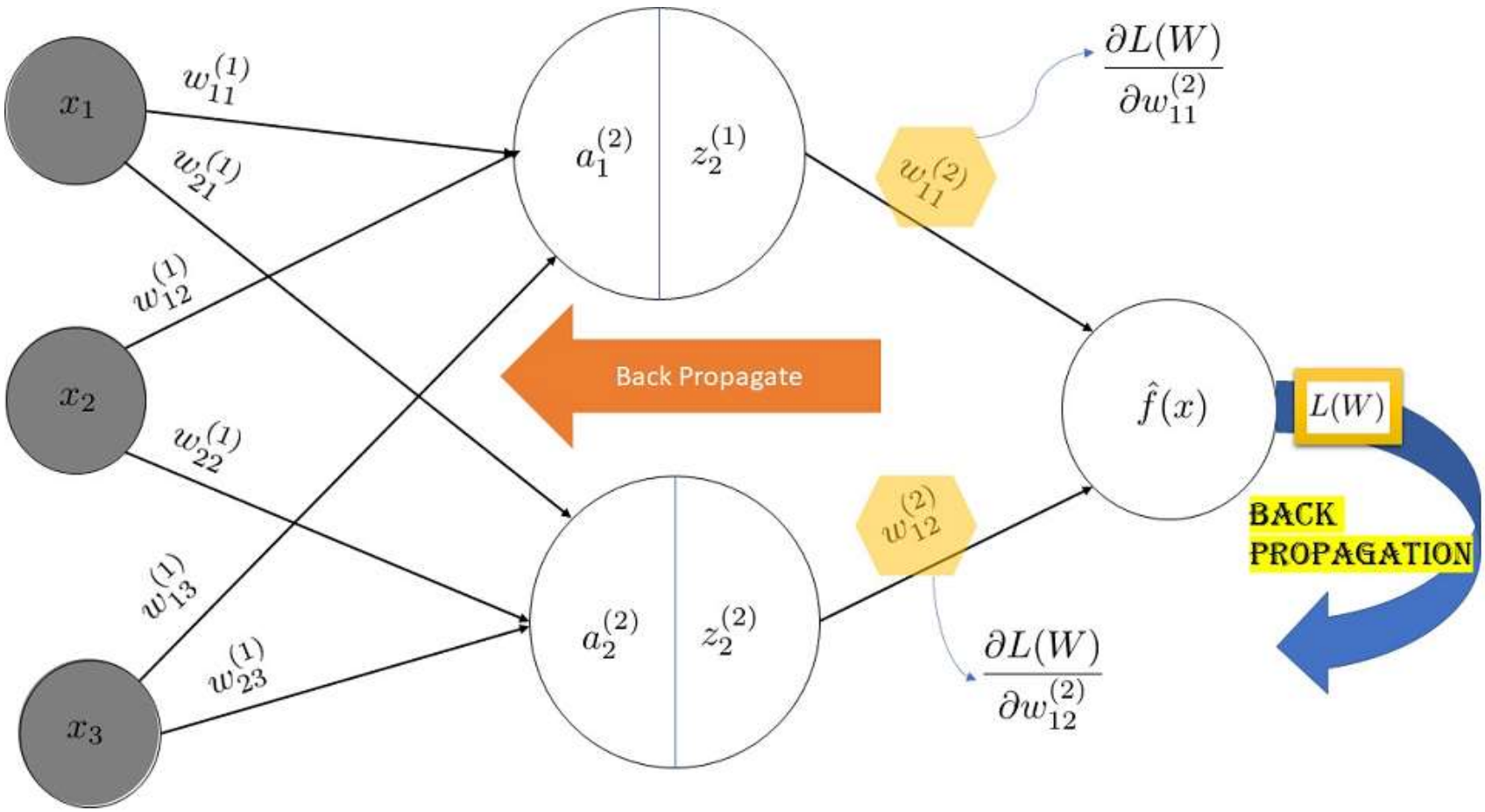

Input
Hidden layer
Output
x_1
x_2
x_3
$w_{11}^{(1)}$
$w_{21}^{(1)}$
$w_{12}^{(1)}$
$w_{22}^{(1)}$
$w_{13}^{(1)}$
$w_{23}^{(1)}$
$a_1^{(2)}$
$z_2^{(1)}$
$a_2^{(2)}$
$z_2^{(2)}$
$w_{11}^{(2)}$
$w_{12}^{(2)}$
Forward Propagate
$\hat{f}(x)$
$\hat{f}(x) = \sum_{j=1}^{2} w_{1j}^{(2)} \left(h^{(1)} \left(\sum_{i=1}^{3} w_{1i}^{(1)} x_i \right) \right)$

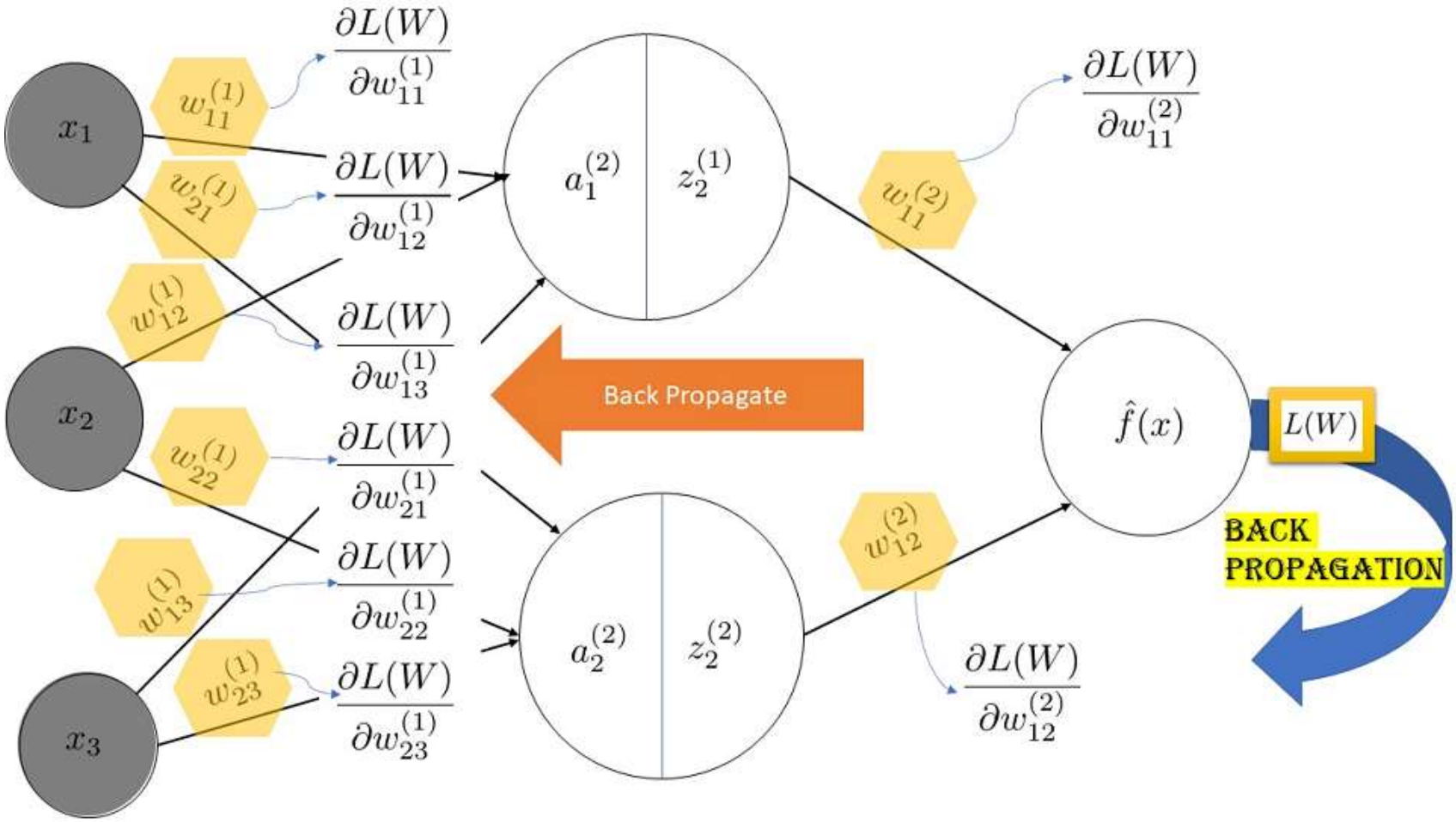

Input
Hidden layer
Output
x_1
x_2
x_3
$w_{11}^{(1)}$
$w_{21}^{(1)}$
$w_{12}^{(1)}$
$w_{22}^{(1)}$
$w_{13}^{(1)}$
$w_{23}^{(1)}$
$a_1^{(2)}$
$z_2^{(1)}$
$a_2^{(2)}$
$z_2^{(2)}$
$w_{11}^{(2)}$
$w_{12}^{(2)}$
Forward Propagate
$\hat{f}(x)$
$W = (\omega_{11}^{(1)}, \ldots, \omega_{12}^{(2)})$
$L(W) = \frac{1}{2p} \sum_{t=1}^{p} \left(f\left(y^t\right) - \hat{f}\left(W|y^t\right) \right)^2$


x_1
x_2
x_3
$w_{11}^{(1)}$
$w_{21}^{(1)}$
$w_{12}^{(1)}$
$w_{22}^{(1)}$
$w_{13}^{(1)}$
$w_{23}^{(1)}$
$a_1^{(2)}$
$z_2^{(1)}$
$a_2^{(2)}$
$z_2^{(2)}$
$w_{11}^{(2)}$
$w_{12}^{(2)}$
$\hat{f}(x)$
$L(W)$
BACK
PROPAGATION


x_1
x_2
x_3
$w_{11}^{(1)}$
$w_{21}^{(1)}$
$w_{12}^{(1)}$
$w_{22}^{(1)}$
$w_{13}^{(1)}$
$w_{23}^{(1)}$
$a_1^{(2)}$
$z_2^{(1)}$
$a_2^{(2)}$
$z_2^{(2)}$
$w_{11}^{(2)}$
$w_{12}^{(2)}$
Back Propagate
$\hat{f}(x)$
$L(W)$
BACK
PROPAGATION


x_1
x_2
x_3
$w_{11}^{(1)}$
$w_{21}^{(1)}$
$w_{12}^{(1)}$
$w_{22}^{(1)}$
$w_{13}^{(1)}$
$w_{23}^{(1)}$
$a_1^{(2)}$
$z_2^{(1)}$
$a_2^{(2)}$
$z_2^{(2)}$
$w_{11}^{(2)}$
$w_{12}^{(2)}$
$\frac{\partial L(W)}{\partial w_{11}^{(2)}}$
Back Propagate
$\hat{f}(x)$
$L(W)$
BACK PROPAGATION


x_1
x_2
x_3
$w_{11}^{(1)}$
$w_{21}^{(1)}$
$w_{12}^{(1)}$
$w_{22}^{(1)}$
$w_{13}^{(1)}$
$w_{23}^{(1)}$
$a_1^{(2)}$
$z_2^{(1)}$
$a_2^{(2)}$
$z_2^{(2)}$
Back Propagate
$w_{11}^{(2)}$
$w_{12}^{(2)}$
$\frac{\partial L(W)}{\partial w_{11}^{(2)}}$
$\frac{\partial L(W)}{\partial w_{12}^{(2)}}$
$\hat{f}(x)$
$L(W)$
BACK PROPAGATION


x_1
x_2
x_3
$w_{11}^{(1)}$
$w_{21}^{(1)}$
$w_{12}^{(1)}$
$w_{22}^{(1)}$
$w_{13}^{(1)}$
$w_{23}^{(1)}$
$\frac{\partial L(W)}{\partial w_{11}^{(1)}}$
$\frac{\partial L(W)}{\partial w_{12}^{(1)}}$
$\frac{\partial L(W)}{\partial w_{13}^{(1)}}$
$\frac{\partial L(W)}{\partial w_{21}^{(1)}}$
$\frac{\partial L(W)}{\partial w_{22}^{(1)}}$
$\frac{\partial L(W)}{\partial w_{23}^{(1)}}$
$a_1^{(2)}$
$z_2^{(1)}$
$a_2^{(2)}$
$z_2^{(2)}$
Back Propagate
$w_{11}^{(2)}$
$w_{12}^{(2)}$
$\frac{\partial L(W)}{\partial w_{11}^{(2)}}$
$\frac{\partial L(W)}{\partial w_{12}^{(2)}}$
$\hat{f}(x)$
$L(W)$
BACK PROPAGATION

Questions?

Dr. Mostafa Rezapour
Email: mrezapou@wakehealth.edu

Wake Forest University
School of Medicine

CAIR
CENTER FOR ARTIFICIAL INTELLIGENCE RESEARCH