# Module Workshop – Machine Learning

J-Payroll – 14 April 2025

# About Speakers

**Dr. Adhiguna Mahendra, B.Cs., M.Cs.,**

Director of Data and AI at Nusantara Capital Authority.

21+ years of experience in tech industry. Leading the transformation of Nusantara Capital Authority into a data-driven organization.

**Ekki Rinaldi B.Cs, M.Cs.,**

Former Head of Data and AI Nusantara Capital Authority.

8+ years of experience in building AI and blockchain product for government and enterprise use cases with more than 50+ successful projects.

jakarta smart city
NUSANTARA
BANK INDONESIA
G20 INDONESIA 2022
KEPOLISIAN NEGARA REPUBLIK INDONESIA
IMIGRASI
Telkom Indonesia the world in your hand
Kredivo Buy now, Pay later
BANK BRI
skmigas
Direktorat Jenderal Bea dan Cukai
Kementerian Keuangan Republik Indonesia Direktorat Jenderal Pajak

20+

Smart Cities & Government Projects

30+

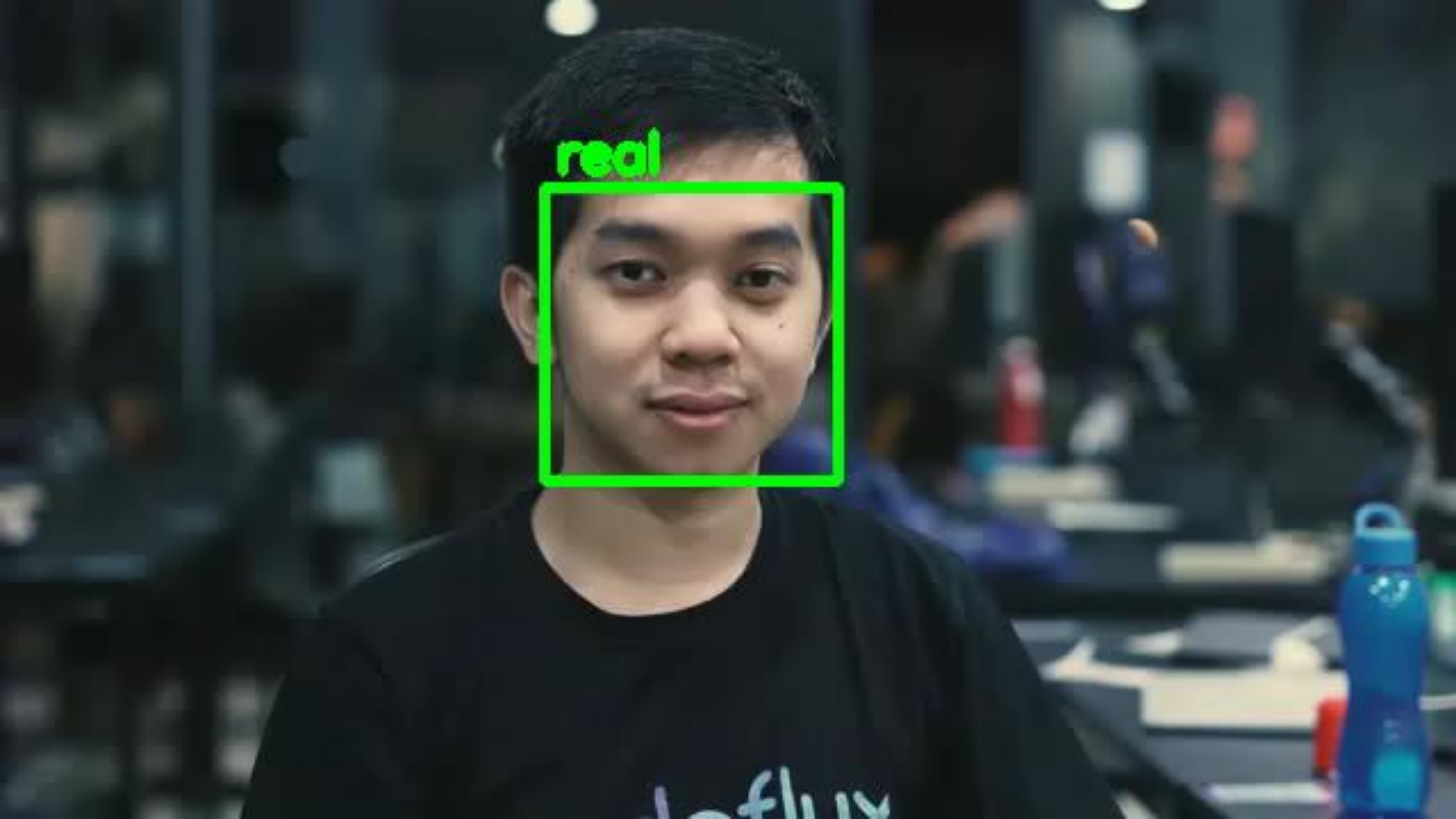
Enterprise

# Workshop Objectives

1. Introduction to Machine Learning in General
2. Use Case: Employee Churn
3. Use Case: Payroll Analysis

nodeflux

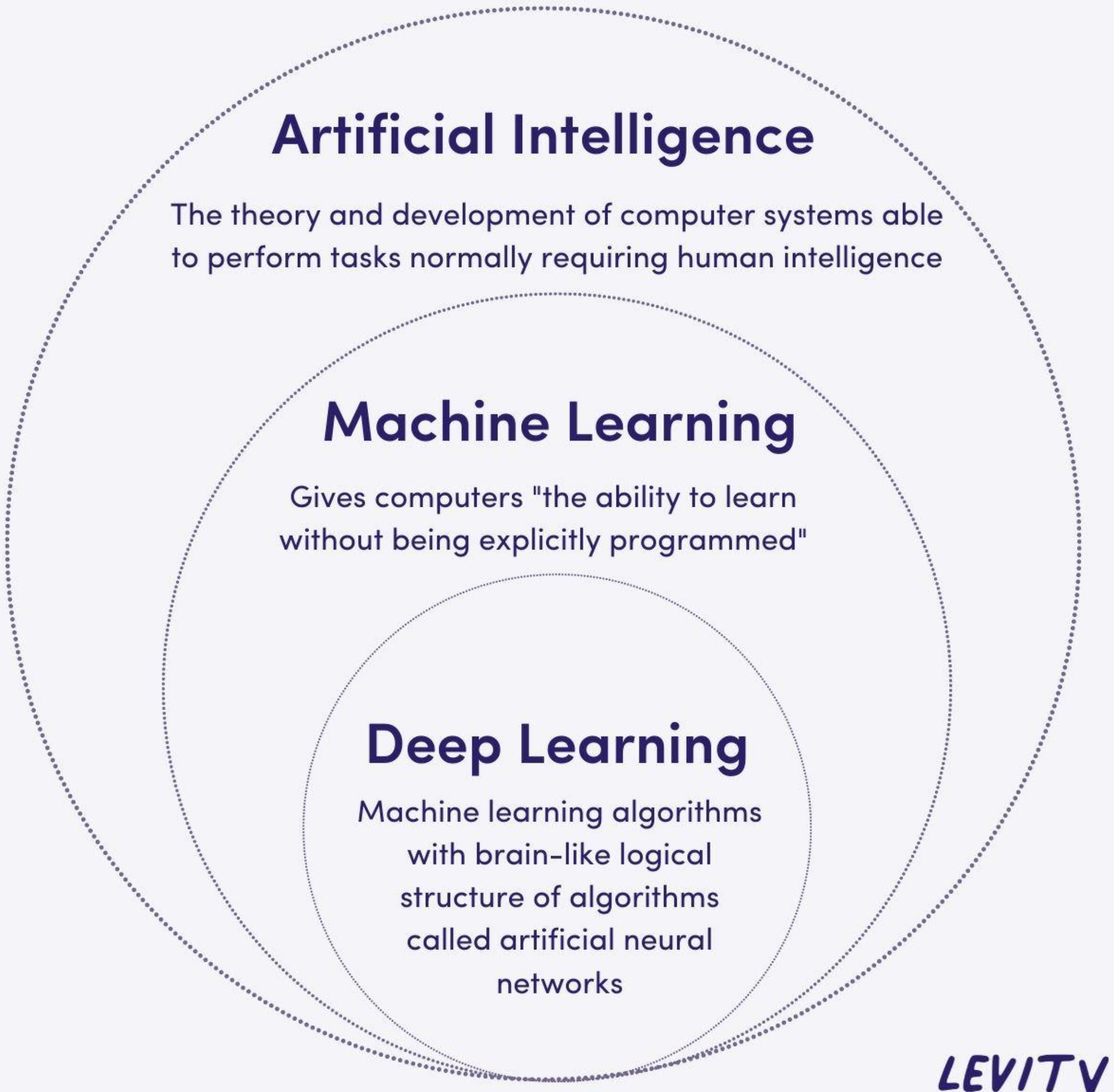
Extending Vision Beyond Imagination


real

# Foundations of Machine Learning

# Definition and Core Principles

Machine learning (ML) represents a subset of artificial intelligence focused on developing algorithms that automatically improve through experience. Unlike traditional programming where humans define explicit rules, ML systems identify patterns in data to make decisions or predictions. This paradigm shift enables handling complex problems where manual rule creation proves impractical, such as image recognition or natural language processing


Artificial Intelligence
The theory and development of computer systems able to perform tasks normally requiring human intelligence
Machine Learning
Gives computers "the ability to learn without being explicitly programmed"
Deep Learning
Machine learning algorithms with brain-like logical structure of algorithms called artificial neural networks

LEVITY

# Definition and Core Principles

The mathematical foundation of machine learning rests on statistical learning theory, optimization methods, and computational algorithms. Key components include:

- **Feature representation**: Turning raw data into meaningful numerical representations
- **Model architecture**: Designing mathematical structures that map inputs to outputs
- **Loss functions**: Quantifying prediction errors to guide learning
- **Optimization techniques**: Adjusting model parameters to minimize errors

# Types of Machine Learning Systems

**Supervised Learning**

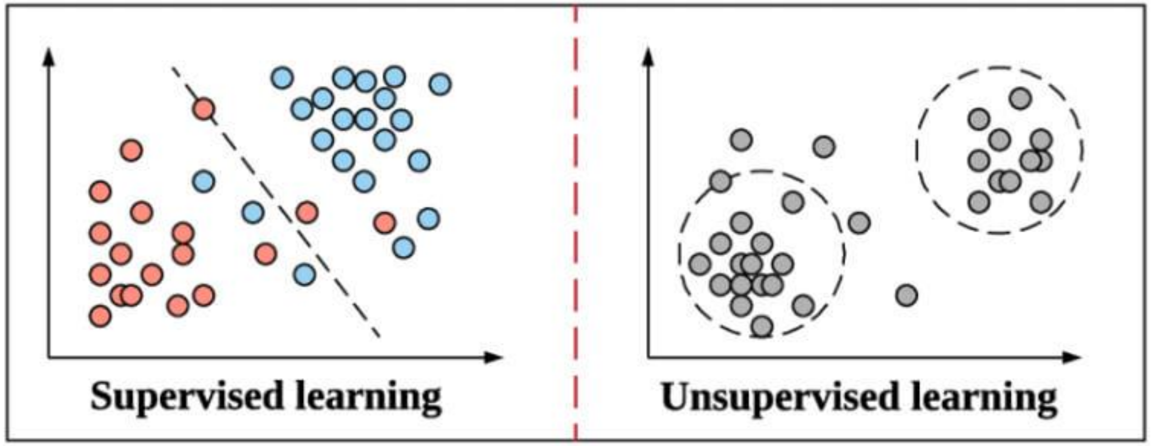
- **Regression**: Predicting continuous values (e.g., house prices) using algorithms like linear regression
- **Classification**: Categorizing inputs into discrete classes (e.g., spam detection) using methods such as logistic regression and support vector machines

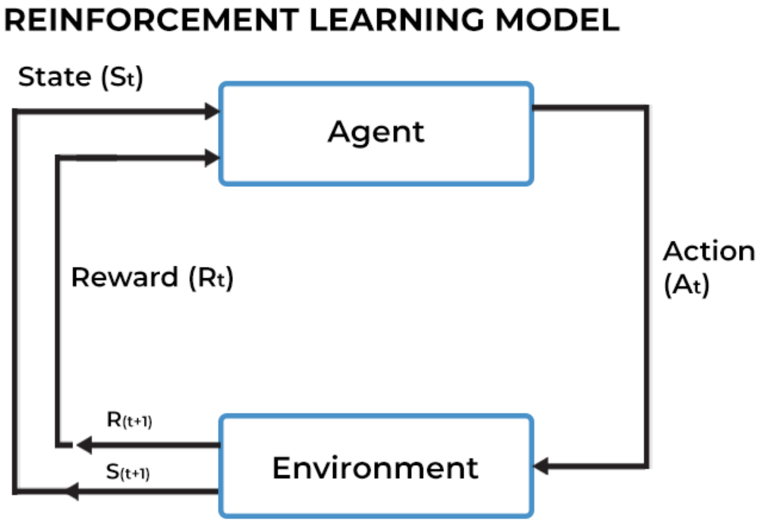
**Unsupervised Learning**

- **Clustering**: Grouping similar data points using algorithms like k-means
- **Dimensionality reduction**: Simplifying data while preserving essential information via techniques like PCA

**Reinforcement Learning**

- This approach trains agents through environmental interactions and reward signals, enabling complex decision-making in dynamic environments. Applications range from game AI to robotic control systems.


Supervised learning
Unsupervised learning


REINFORCEMENT LEARNING MODEL
State (St)
Agent
Reward (Rt)
Action
(At)
R(t+1)
S(t+1)
Environment

# Key Algorithms and Methodologies

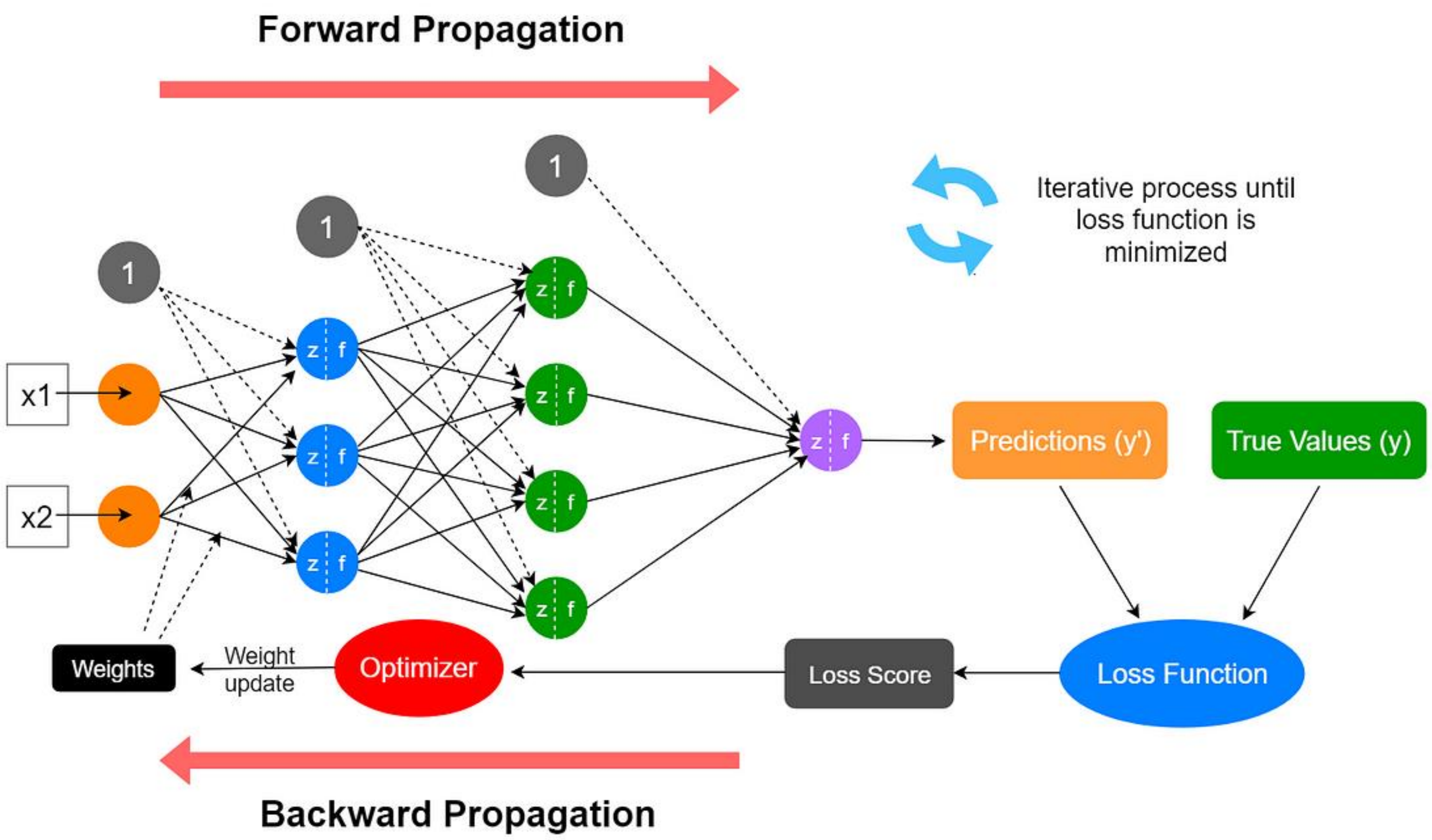
**Linear Models**

- Linear regression: Models linear relationships between variables using least squares estimation
- Logistic regression: Estimates class probabilities through logistic function transformation

**Tree-Based Methods**
Decision trees partition feature space through hierarchical splits, while ensemble methods like random forests combine multiple trees to improve accuracy and reduce overfitting

**Support Vector Machines (SVM)**
SVMs construct optimal hyperplanes for classification by maximizing margin between classes, effective in high-dimensional spaces.

**Neural Networks**
Deep learning architectures employ layered transformations to model complex patterns, driving breakthroughs in computer vision and natural language processing


Forward Propagation
Iterative process until loss function is minimized
1
1
1
x1
x2
z f
Predictions (y')
True Values (y)
Weights
Weight update
Optimizer
Loss Score
Loss Function
Backward Propagation

# Practical Applications Across Industries

- **Healthcare Revolution**
  ML enables early disease detection through medical imaging analysis and predictive modeling of patient outcomes. Algorithms can analyze radiology scans with accuracy rivaling human experts while predicting hospital readmission risks.
- **Financial Services Transformation**
  Applications span fraud detection through anomaly identification, algorithmic trading strategies, and personalized financial advising. ML models analyze transaction patterns to flag suspicious activity in real-time.
- **Manufacturing Optimization**
  Predictive maintenance systems use sensor data and ML models to anticipate equipment failures, reducing downtime. Quality control systems employ computer vision to detect product defects.
- **Retail Personalization**
  Recommendation engines leverage collaborative filtering and deep learning to suggest products, while dynamic pricing algorithms optimize margins based on market conditions.
- **Cybersecurity Advancements**
  ML enhances threat detection through network traffic analysis and malware classification. Behavioral biometrics authenticate users based on interaction patterns.

# Challenges and Considerations

**Data Quality Issues**

Model performance heavily depends on training data quality. Common challenges include:

- Insufficient data: Limited samples lead to poor generalization
- Label noise: Incorrect annotations degrade model accuracy
- Sampling bias: Non-representative data creates skewed predictions

**Algorithmic Bias and Fairness**

Models can perpetuate societal biases present in training data, leading to discriminatory outcomes. Mitigation strategies include:

- Bias auditing frameworks
- Adversarial debiasing techniques
- Diverse dataset curation

**Model Interpretability**

The "black box" nature of complex models like deep neural networks raises accountability concerns. Emerging solutions include:

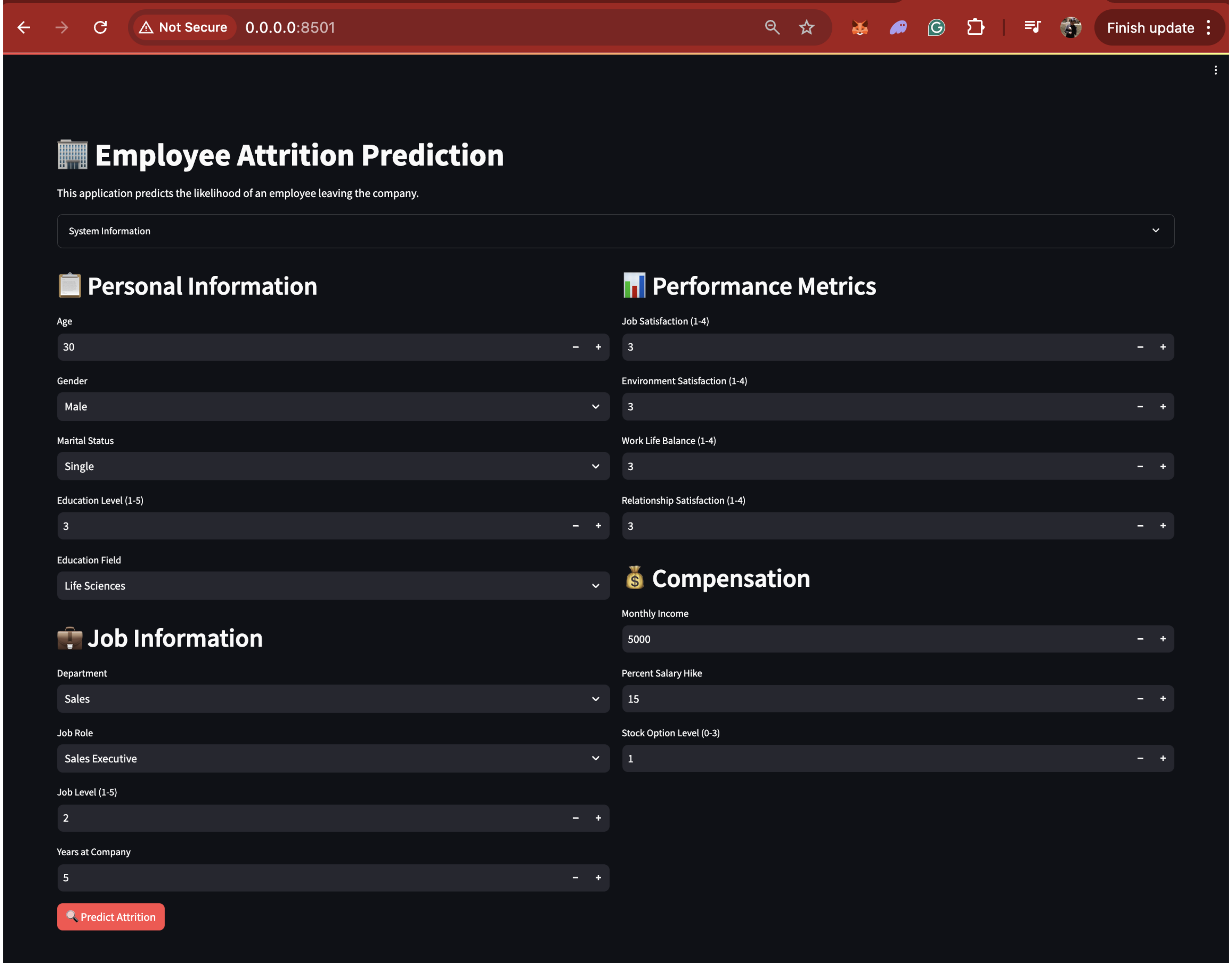
- Local interpretable model-agnostic explanations (LIME)
- Shapley value analysis
- Attention visualization in transformers

# Trends and Futures

1. **Edge Computing Integration**
   Deploying ML models on edge devices enables real-time processing with reduced latency, crucial for applications like autonomous vehicles and IoT systems.
2. **Automated Machine Learning (AutoML)**
   AutoML platforms automate feature engineering, model selection, and hyperparameter tuning, democratizing access to ML capabilities
3. **Federated Learning**
   This decentralized approach trains models across distributed devices while preserving data privacy, particularly valuable in healthcare and finance
4. **Quantum Machine Learning**
   Quantum computing promises to accelerate optimization problems and enable new ML paradigms through quantum-enhanced algorithms

# Machine Learning to Production

## Employee Attrition Prediction System


Not Secure
0.0.0.0:8501
Finish update
Employee Attrition Prediction
This application predicts the likelihood of an employee leaving the company.
System Information
Personal Information
Age
30
Gender
Male
Marital Status
Single
Education Level (1-5)
3
Education Field
Life Sciences
Job Information
Department
Sales
Job Role
Sales Executive
Job Level (1-5)
2
Years at Company
5
Predict Attrition
Performance Metrics
Job Satisfaction (1-4)
3
Environment Satisfaction (1-4)
3
Work Life Balance (1-4)
3
Relationship Satisfaction (1-4)
3
Compensation
Monthly Income
5000
Percent Salary Hike
15
Stock Option Level (0-3)
1

# Employee Attrition Prediction System

This project implements a machine learning system for predicting employee attrition using XGBoost. The system consists of three main components:

1. Model Training (`model_train.py`)
2. Inference API (`inference.py`)
3. Web Interface (`main.py`)

# Project Structure

````
```

employee-attrition-and-performance/
├── data-employee-attrition.csv    # Dataset file
├── model_train.py                 # Model training script
├── inference.py                   # FastAPI inference service
├── main.py                        # Streamlit web interface
├── requirements.txt               # Python dependencies
├── Dockerfile.inference           # Dockerfile for inference service
├── Dockerfile.web                # Dockerfile for web interface
└── docker-compose.yml            # Docker Compose configuration
```
````

# Features

## Model Training

- XGBoost classifier for attrition prediction
- Automated data preprocessing
- Model evaluation with accuracy metrics
- Model persistence for later use

## Inference API

- RESTful API using FastAPI
- Real-time predictions
- Input validation
- Detailed error handling
- API documentation (Swagger UI)

## Web Interface

- User-friendly interface using Streamlit
- Intuitive form inputs
- Real-time predictions
- Visual presentation of results
- Error handling and user feedback

# Requirements

- Python 3.11+
- Docker (optional, for containerized deployment)

# Installation & Setup: Local Development

1. Create a virtual environment:

```
python -m venv venv
source venv/bin/activate # On Mac/ubuntu
venv\Scripts\activate # On Windows
```

2. Install dependencies:

```
pip install -r requirements.txt
```

3. Train the model:

```
python model_train.py
```

4. Start the inference service:

```
python inference.py
```

5. In a new terminal, start the web interface:

```
streamlit run main.py
```

# Installation & Setup: Docker Deployment

Build and run using Docker Compose:

```
docker compose up --build
```

This will:

- Build both containers
- Train the model
- Start the inference service
- Launch the web interface

# Accessing the Services

### API Endpoint

- Access in `http://localhost:8000`

### API Documentation

- Access in `http://localhost:8000/docs`
- Interactive API documentation
- Test API endpoints directly
- View request/response schemas

### Web Interface

- Access in `http://localhost:8501`
- Fill in the employee information form
- Click "Predict Attrition" to get predictions
- View probability and confidence scores

# API Endpoints

```
1. **Root Endpoint** (`GET /`):
   - Check API status
   - View available endpoints
   - Verify model availability

2. **Prediction Endpoint** (`POST /predict`):
   - Submit employee data
   - Receive attrition predictions
   - Get probability scores
```

# Running Individual Components

### 1. Model Training

```
python model_train.py
```

This will:

- Load and preprocess the dataset
- Train the XGBoost model
- Save the model as 'xgboost_model.pkl'
- Display model performance metrics

### 2. Inference API

```
python inference.py
```

This will:

- Load the trained model
- Start the FastAPI server on port 8000
- Enable real-time predictions

### 3. Web Interface

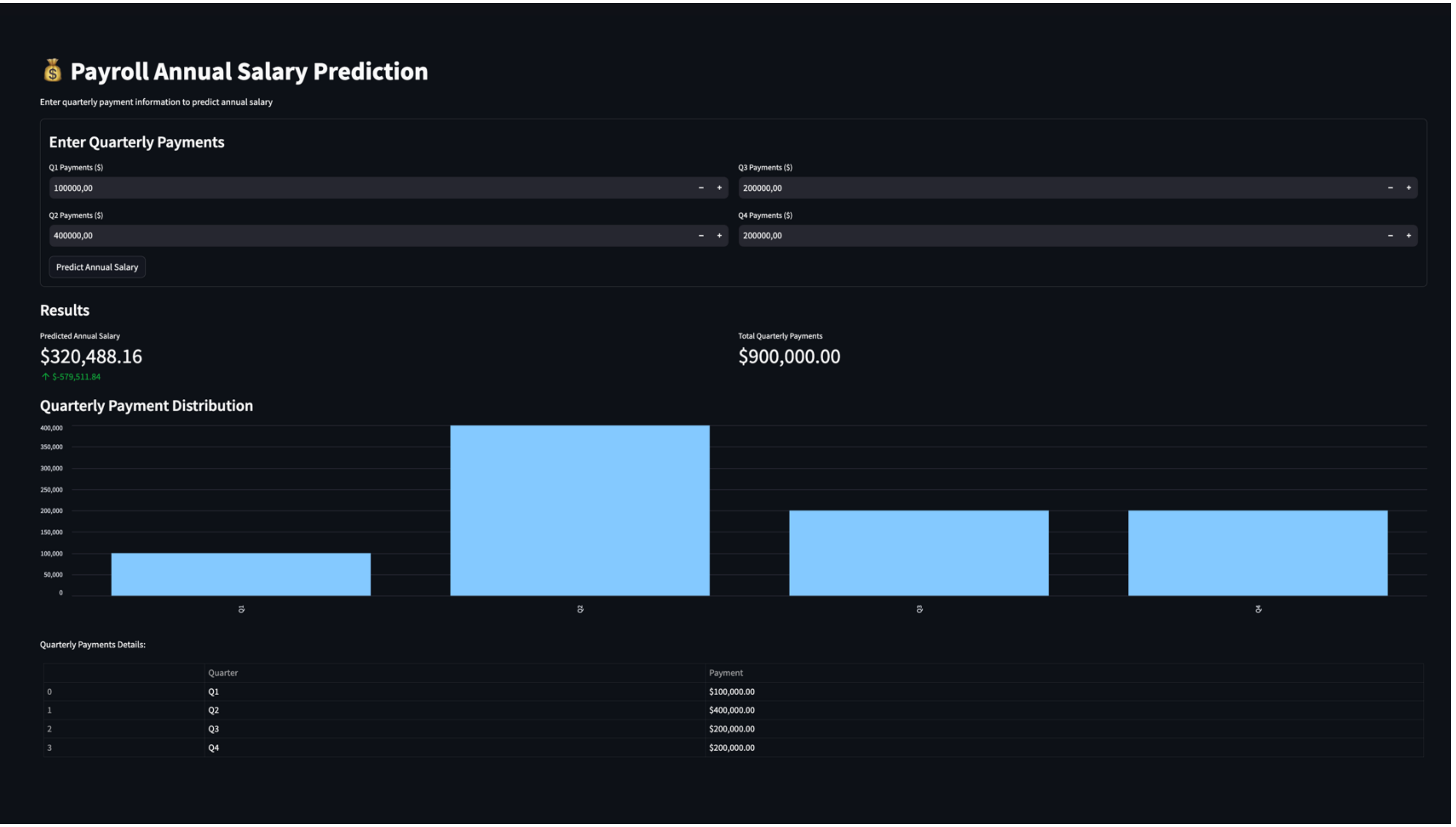
```
streamlit run main.py
```

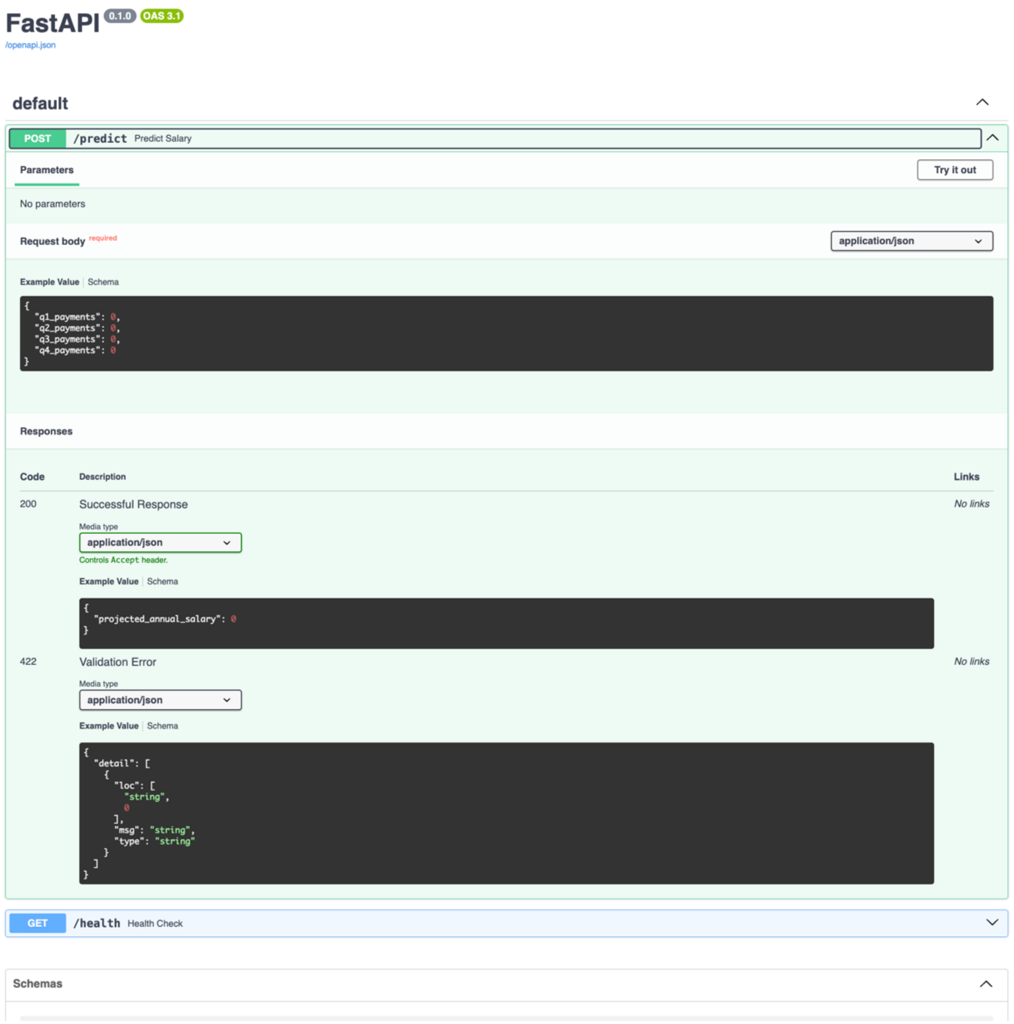
This will:

- Start the Streamlit interface on port 8501
- Connect to the inference API
- Provide the user interface

# Machine Learning to Production

**Payroll Modelling**


Payroll Annual Salary Prediction
Enter quarterly payment information to predict annual salary
Enter Quarterly Payments
Q1 Payments ($)
100000,00
Q3 Payments ($)
200000,00
Q2 Payments ($)
400000,00
Q4 Payments ($)
200000,00
Predict Annual Salary
Results
Predicted Annual Salary
$320,488.16
$-579,511.84
Total Quarterly Payments
$900,000.00
Quarterly Payment Distribution
400,000
350,000
300,000
250,000
200,000
150,000
100,000
50,000
0
Quarterly Payments Details:
Quarter
Payment
0 Q1 $100,000.00
1 Q2 $400,000.00
2 Q3 $200,000.00
3 Q4 $200,000.00


FastAPI 0.1.0 OAS 3.1
/openapi.json
default
POST /predict Predict Salary
Parameters
Try it out
No parameters
Request body required
application/json
Example Value | Schema
{
"q1_payments": 0,
"q2_payments": 0,
"q3_payments": 0,
"q4_payments": 0
}
Responses
Code Description Links
200 Successful Response No links
Media type
application/json
Controls Accept header.
Example Value | Schema
{
"projected_annual_salary": 0
}
422 Validation Error No links
Media type
application/json
Example Value | Schema
{
"detail": [
{
"loc": [
"string",
0
],
"msg": "string",
"type": "string"
}
]
}
GET /health Health Check
Schemas

# Features

- Machine learning model for salary prediction
- RESTful API for model inference
- User-friendly web interface
- Real-time predictions
- Data visualization
- Cross-platform compatibility (Windows, macOS, Linux)

# System Requirements

- Python 3.11 or higher
- Docker (optional, but recommended)
- 4GB RAM minimum
- 2GB free disk space

# Installation Using Docker (Recommended)

1. Install Docker and Docker Compose:

- Windows: Install Docker Desktop for Windows
- macOS: Install Docker Desktop for Mac
- Linux: Install Docker Engine and Docker Compose

2. Clone or download this repository:

```
git clone https://github.com/ekkirinaldi/workshop-machine-learning
cd payroll-modelling
```

3. Build and run the services:

```
docker-compose up --build
```

# Manual Installation

1. Create and activate a virtual environment:

```
# Windows
python -m venv venv
.\venv\Scripts\activate

# macOS/Linux
python3 -m venv venv
source venv/bin/activate
```

2. Install dependencies:

```
pip install -r requirements.txt
```

3. Train the model:

```
python payroll_train.py
```

4. Start the API server:

```
# Windows
uvicorn payroll_inference:app --host
0.0.0.0 --port 8000

# macOS/Linux
python -m uvicorn
payroll_inference:app --host 0.0.0.0
--port 8000
```

5. Start the UI (in a new terminal):

```
streamlit run payroll_ui.py
```

# Usage

1. Access the web interface:
- Open your browser and go to http://localhost:8501

2. Enter quarterly payment data:
- Input payment amounts for Q1, Q2, Q3, and Q4
- Click "Predict Annual Salary"
- View the prediction and visualization

3. API endpoints:
- Prediction API: http://localhost:8000/predict
- API documentation: http://localhost:8000/docs