GUJARAT TECHNOLOGICAL UNIVERSITY
Chandkheda, Ahmedabad

Affiliated

SAL EDUCATION

SAL EDUCATION CAMPUS

SAL Engineering and Technical Institute

AI Driven optical PCB Anomaly detection in Smartphone Assembly Line

A Report

Submitted By

Raj Paresh Bhai Patel **221260116038**

Under Subject of

SUMMER INTERNSHIP (3170001)

Of

BACHELOR OF ENGINEERING

In

Computer Engineering

Under the guidance of

Prof. Krupa Patel

Academic Year
2025-26

intel. digital readiness
+
Certificate of Completion
This certifies that
Raj Paresh Bhai Patel
has completed the AI for Manufacturing Certificate Course
under the collaboration of Gujarat Technological University (GTU) & Intel India
on July 15, 2025.
Shweta Khurana
Sr. Director – Asia Pacific & Japan,
Government Partnerships & Initiatives,
International Government Affairs, Intel
Dr. K. N. Kher
Registrar,
Gujarat Technological University
AI4MGTUS1507250421

SAL ENGINEERING AND TECHNICAL INSTITUTE

Information & Technology

2025

CERTIFICATE

Date:

This is to certify that the AI Driven optical PCB Anomaly detection in Smartphone Assembly Line has been carried out by **Raj Paresh Bhai Patel (221260116038)** under my guidance in completion of Summer Internship in Information Technology Branch 7th Semester of Gujarat Technological University, Ahmedabad during the academic Year 2025-26.

Prof. Krupa Patel Internal Guide Computer Engineering SETI	Prof. Sachi Bhavsar Head of Department IT & ICT SETI

ABSTRACT

The manufacturing industry is undergoing a profound transformation driven by Industry , with Artificial Intelligence (AI) at its core. This internship report, "AI for Manufacturing: From Concepts to Deployment," documents a comprehensive 14-week journey through the Intel® AI for Manufacturing certificate course. The program provided a structured deep dive into the practical application of AI technologies to solve real-world industrial problems.

The curriculum covered a wide spectrum of essential topics, beginning with the foundational principles of AI and Python programming. It progressively advanced through the entire AI project lifecycle, including data acquisition and preprocessing, model building and evaluation, and culminating in the deployment of a scalable web application. Key technical areas explored include Computer Vision for automated quality inspection and defect detection, Predictive Maintenance for minimizing machine downtime, Robotic Process Automation (RPA) for streamlining back-office tasks, and Edge Inferencing for enabling real-time, low-latency analysis directly on devices.

A significant outcome of this internship was the development and deployment of "Timelytics," an end-to-end machine learning project. This project involved building a predictive model to estimate order-to-delivery times using an XGBoost algorithm and deploying it as an interactive web application using the Streamlit framework. This hands-on experience provided invaluable insights into the complete pipeline of a data science project, from raw data to a functional tool that provides actionable business intelligence.

This report concludes that the integration of AI is not merely an upgrade but a fundamental shift towards intelligent, efficient, and sustainable manufacturing operations.

Contents

Chapter - 01

1.1 Introduction to Summer Internship

This report documents the activities and learnings from a focused 14-day summer internship undertaken as part of the Intel® AI for Manufacturing online certificate course. The internship was conducted remotely from February 22, 2025, to April 26, 2025, aligning with the academic curriculum of the 7th semester. The program was designed to bridge the gap between theoretical AI concepts and their practical application in the manufacturing sector, which is rapidly evolving under Industry 4.0. The internship provided a structured learning path, covering the entire AI project lifecycle, from problem definition and data handling to model deployment and ethical considerations, with each week of the course representing a day of intensive learning and application in this report.

1.2 Aim & Objectives

The primary aim of this internship was to acquire and demonstrate proficiency in applying AI solutions to optimize manufacturing processes. The key objectives were:

- To understand the fundamental concepts of Artificial Intelligence and Machine Learning and their significance in modern manufacturing.
- To gain hands-on experience with the Python programming language and essential libraries like Pandas, NumPy, and Scikit-learn for data analysis and modeling.
- To learn and implement the complete AI project lifecycle, including data preprocessing, feature engineering, model training, evaluation, and deployment.
- To explore specialized AI applications in manufacturing, including Computer Vision for quality control, Predictive Maintenance for asset management, and RPA for process automation.
- To develop a fully functional AI-based web application ("Timelytics") for predicting order-to-delivery times, thereby understanding the end-to-end process of deploying an AI model.
- To understand the ethical implications and responsible practices necessary for deploying AI systems in industrial settings.

1.3 Tools & Technologies Used

The following tools and technologies were utilized throughout the duration of the internship:

- **Programming Languages:** Python
- **Python Libraries:** NumPy, Pandas, Matplotlib, Scikit-learn, XGBoost, TensorFlow, OpenCV, Streamlit, Joblib
- **Development Environments:** Jupyter Notebook, Google Colab, Visual Studio Code, PyCharm
- **AI/ML Tools:** Teachable Machine (Google), Microsoft Responsible AI Toolbox (InterpretML, Fairlearn)
- **Web Frameworks:** Streamlit
- **Deployment Platforms:** Streamlit Community Cloud, Vercel
- **Version Control:** Git, GitHub
- **Other Tools:** Microsoft Word, PowerPoint

Chapter - 02

2.1 Day 1: Introduction to AI for Manufacturing

Activity Overview:

The inaugural day of the internship served as a strategic orientation, shifting the perspective of Artificial Intelligence from a futuristic concept to a present-day, practical tool for industrial optimization. The focus was on understanding the macro-level impact of AI on the global manufacturing landscape, with a specific emphasis on democratizing this technology for Small and Medium-sized Enterprises (SMEs). The session outlined how AI acts as a force multiplier, enabling smaller operations to compete with larger corporations by enhancing efficiency, reducing costs, and improving decision-making. The primary activity involved research and analysis to identify and articulate specific AI-enabled opportunities and their associated benefits tailored to the SME context.

Learning Outcomes:

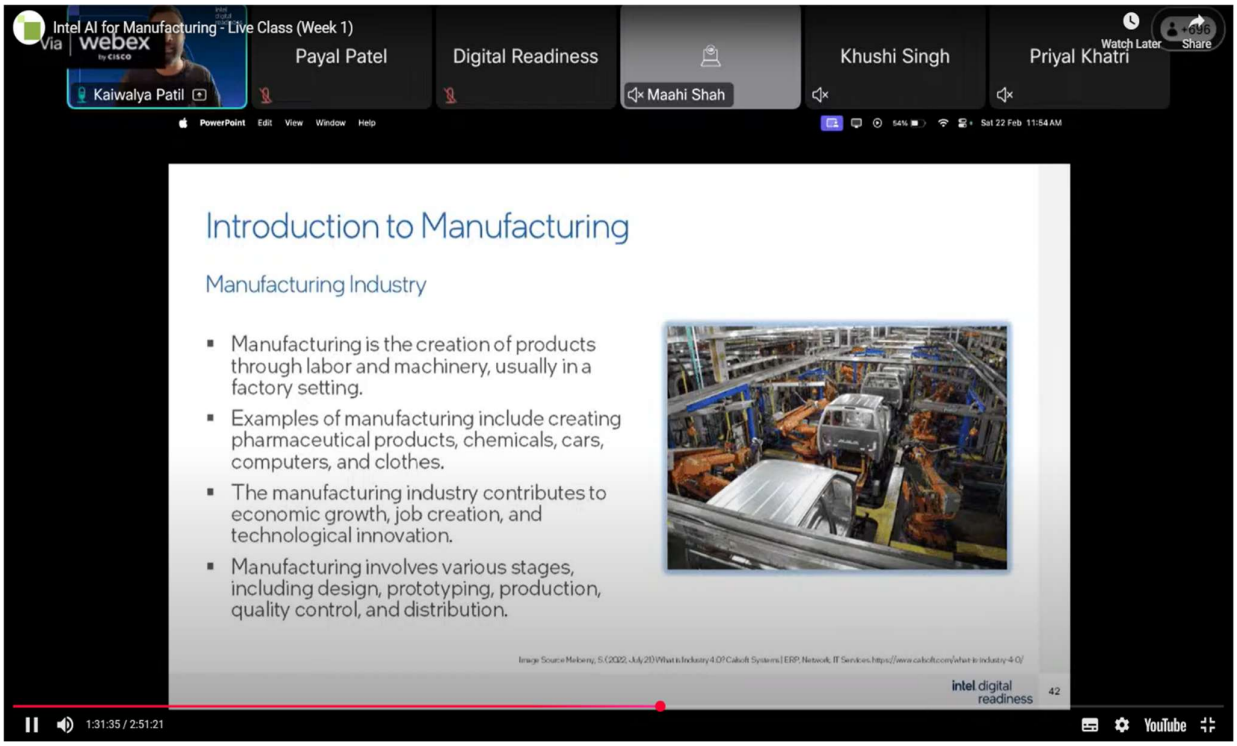
- **Industry Context:** Understood that AI is a foundational pillar of the Fourth Industrial Revolution (Industry), which is characterized by the fusion of digital, biological, and physical worlds. It enables the creation of "smart factories" where cyber-physical systems monitor physical processes, create a virtual copy of the physical world, and make decentralized decisions.

- **AI Opportunities for SMEs:** Researched and documented a comprehensive list of high-impact, accessible AI applications for SMEs, moving beyond theoretical concepts to practical implementations:
    1. **Predictive Maintenance:** Learned how SMEs can retrofit existing machinery with IoT sensors (vibration, temperature, acoustic) to collect real-time data. Machine Learning algorithms analyze this data to predict potential equipment failures before they occur, scheduling maintenance only when needed. This prevents costly unplanned downtime, extends asset life, and optimizes maintenance resource allocation.
    2. **Quality Control & Defect Inspection:** Explored the application of Computer Vision systems powered by AI. These systems use high-resolution cameras to automatically scan products on the production line. AI models are trained to detect defects (cracks, misalignments, color discrepancies, surface imperfections) with higher accuracy and speed than human inspectors, significantly reducing the rate of defective products reaching customers.
    3. **Supply Chain & Inventory Management:** Understood how AI tools can revolutionize logistics. By analyzing historical sales data, seasonality, market trends, and even external factors like weather, AI algorithms can forecast demand with high accuracy. This allows SMEs to optimize stock levels, avoiding both overstocking (which ties up capital) and stockouts (which lead to lost sales), and automate reordering processes.
    4. **AI-Powered Financial Management:** Discovered accessible AI-driven accounting software (e.g., Zoho Books, QuickBooks Online) that can automate repetitive tasks like bookkeeping, invoice processing, and expense categorization. These tools can also identify patterns indicative of fraud and provide predictive insights into cash

flow, enabling better financial planning.

- **Tangible Benefits:** Recognized and categorized the direct benefits AI brings to SMEs, which formed the core of a completed assignment:
  - **Cost Reduction:** Through automation of repetitive tasks (data entry, visual inspection) and optimized resource use (inventory, energy, maintenance).
  - **Improved Decision-Making:** By moving from intuition-based decisions to data-driven insights derived from analyzing large datasets.
  - **Increased Productivity:** Freeing up human workers from mundane tasks to focus on higher-value activities like innovation and customer relationship management.
  - **Competitive Advantage:** Allowing SMEs to offer higher quality products, faster delivery times, and more personalized services, thereby competing effectively with larger players.

Day - 01



2.2 Day 2: Introduction to AI Programming with Python

Activity Overview:

The second day was a deep and practical immersion into the primary tool of modern AI development: the Python programming language. The activities were designed to transition from the strategic overview of Day 1 to the technical execution of AI projects. The session involved a comprehensive review of Python's core syntax and concepts, followed by a critical exploration of the key libraries that form the ecosystem for data manipulation, scientific computing, and machine learning. This foundational day was crucial for building the technical competence required to implement the AI concepts discussed throughout the course.

Detailed Learning Outcomes:

- **Core Python Proficiency:** Revised and reinforced fundamental programming concepts that are the building blocks of all data science work:
    - **Variables and Data Types:** Practiced assigning and manipulating different data types (integers, floats, strings, booleans, lists, dictionaries).
    - **Control Flow:** Implemented if, elif, and else conditional statements to create decision-making logic in programs. Practiced using for and while loops to iterate over sequences and perform repetitive tasks.
    - **Functions:** Defined and called functions to create reusable blocks of code, improving code organization and reducing redundancy. Understood the use of parameters and return values.
    - **Basic I/O:** Used the input() function to capture user data and the print() function to display results, creating simple interactive scripts.
    - **Exercises:** Completed hands-on exercises, such as creating a program to determine if a student passed based on a test score, which solidified the understanding of conditional logic and user input handling.
- **Essential Libraries for Manufacturing AI:** Researched, documented, and understood the application of three cornerstone Python libraries in a manufacturing context. This was not just a theoretical exercise; it involved connecting each library's functionality to a real-world industrial problem:

    1. **NumPy (Numerical Python):**

        - **Purpose & Features:** Understood that NumPy provides support for large, multi-dimensional arrays and matrices, along of a collection of mathematical functions to operate on these arrays efficiently. Its core is the ndarray object, which is fast, flexible, and consumes less memory than Python lists for numerical data.
        - **Manufacturing Application:** Recognized that machines and sensors in a factory generate vast volumes of numerical data (e.g., temperature, pressure, vibration readings). NumPy is the fundamental library for processing and performing complex mathematical operations on this sensor data. It is indispensable for tasks like **Predictive Maintenance** (e.g., calculating the mean and standard deviation of vibration data to detect anomalies) and **Finite Element Analysis (FEA)** for product design simulations.
        - **Hands-on Example:** Wrote and executed code to process a simulated array

of temperature readings from a machine and calculate the average temperature, a common baseline analysis task.

Figure 2.2.1: Python Libraries for Manufacturing - NumPy Example

```
import numpy as np
# Simulating sensor data: temperature readings from a machine over time
temperature_readings = np.array([72.5, 73.1, 75.2, 76.8, 78.0, 74.9, 72.0])

# Calculating basic descriptive statistics for monitoring
average_temp = np.mean(temperature_readings)
max_temp = np.max(temperature_readings)
min_temp = np.min(temperature_readings)
temp_std_dev = np.std(temperature_readings) # Standard deviation indicates variability

print("Machine Temperature Analysis:")
print(f"Average Temperature: {average_temp:.2f} °F")
print(f"Maximum Temperature: {max_temp} °F")
print(f"Minimum Temperature: {min_temp} °F")
print(f"Standard Deviation: {temp_std_dev:.2f} °F") # A high value could indicate instabi
lity
```

```
Machine Temperature Analysis:
Average Temperature: 74.64 °F
Maximum Temperature: 78.0 °F
Minimum Temperature: 72.0 °F
Standard Deviation: 2.08 °F
```

Pandas (Python Data Analysis Library):

- **Purpose & Features:** Learned that Pandas provides high-performance, easy-to-use data structures, primarily the DataFrame. A DataFrame is a two-dimensional labeled data structure akin to a spreadsheet or SQL table, making it ideal for handling structured, tabular data. It offers powerful tools for data manipulation, cleaning, aggregation, filtering, and merging.
- **Manufacturing Application:** Identified Pandas as the workhorse for **Inventory Management** (tracking stock levels, supplier details), **Quality Control Reporting** (analyzing defect rates by production batch, shift, or machine), and **Supply Chain Optimization** (analyzing supplier performance data, logistics timelines). It is used for any task that involves working with CSV files, Excel spreadsheets, or database exports.
- **Hands-on Example:** Created a simple DataFrame to track product quality and used Pandas operations to filter and identify defective items.

Figure 2.2.2: Python Libraries for Manufacturing - Pandas Example

```
import pandas as pd
# Creating a DataFrame to simulate a quality control report
data = {
    'Product_ID': [101, 102, 103, 104, 105],
    'Product_Name': ['Gear A', 'Bolt B', 'Shaft C', 'Widget D', 'Nut E'],
    'Defective': [False, True, False, True, False]
}
df = pd.DataFrame(data)

# Using Pandas to filter and isolate defective products for review
defective_products = df[df['Defective'] == True]
print("Defective Products Requiring Rework:")
print(defective_products.to_string(index=False))
```

```
Defective Products Requiring Rework:
Product_ID Product_Name  Defective
     102        Bolt B       True
     104      Widget D       True
```

Matplotlib (Data Visualization Library):

- **Purpose & Features:** Explored Matplotlib as a comprehensive library for creating static, animated, and interactive visualizations in Python. Learned to create a wide variety of plots, including line charts, bar charts, histograms, scatter plots, and more. Visualization is critical for communicating insights from data effectively.
- **Manufacturing Application:** Understood its role in **Production Monitoring** (plotting units produced per hour/shift), **Defect Analysis** (creating Pareto charts to identify the most common defect types), **Predictive Maintenance** (visualizing sensor data trends over time to spot drifts), and **Supply Chain Analysis** (plotting inventory levels against time to identify reorder points).
- **Hands-on Example:** Plotted a simple line chart to visualize monthly production output, a key performance indicator (KPI) for any manufacturing unit.

```
import matplotlib.pyplot as plt
# Data: Monthly production output for the first half of the year
months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun']
production_output = [500, 550, 480, 510, 530, 550]

# Creating a line plot to visualize production trends
plt.figure(figsize=(10, 6))
plt.plot(months, production_output, marker='o', linestyle='-', color='b', linewidth=2)
plt.title('Monthly Production Output', fontsize=16, fontweight='bold')
plt.xlabel('Month', fontsize=12)
plt.ylabel('Units Produced', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

2.3 Day 3: AI Project Cycle for Manufacturing

Activity Overview:

The focus of the third day shifted from pure programming to the critical methodology of executing an AI project. The activity involved a deep study of a structured template for planning Machine Learning projects, specifically focusing on the "PREDICT" (how) and "EVALUATE" (how well) phases. This framework is designed to ensure that AI initiatives are not just technically sound but are also aligned with business objectives and are deployable in a real-world manufacturing environment. The day centered on analyzing a provided example of predicting machine part failures and understanding how to apply this planning framework to any industrial AI problem.

Detailed Learning Outcomes:

- Structured Project Scoping: Learned that a successful AI project begins long before any code is written. The PREDICT/EVALUATE template provides a holistic framework for scoping a project by forcing answers to critical questions across several dimensions:
    - ML Tasks:
        - Input & Output: Defined what data the model will consume (e.g., sensor data: temperature, vibration, pressure) and what it will produce (e.g., a prediction: "Will Fail" / "Won't Fail", or a probability score between 0 and 1).
        - ML Task Type: Classified the problem. The machine part failure example was a Binary Classification task (predicting one of two classes). Other common types include Regression (predicting a continuous value, like remaining useful life) and Multi-class Classification (identifying the type of defect).
    - Value Propositions (The "Why"):
        - Goal: Articulated the business goal clearly. For the example, it was to "Identify crucial machine parts which may not function properly."
        - Importance: Understood the business impact: "Improve the efficiency of the manufacturing unit" by reducing unplanned downtime, lowering repair costs, and optimizing maintenance schedules.
        - Beneficiaries: Identified the stakeholders who benefit from the model, such as maintenance managers, production planners, and plant operators.
    - Data Source & Collection:
        - Source: Identified where the raw data originates. In the example, this was "Sensor data" and "Daily logbook of the machine maintenance."
        - Collection: Defined the mechanism for acquiring training data, especially the "ground truth" labels. This involved "Check[ing] the machine maintenance logbook to see which machine parts malfunctioned/wore out in the last month" to create a labeled dataset linking sensor patterns to known outcomes.
    - Offline Evaluation:
        - Methods & Metrics: Determined how to validate the model's performance before it is ever deployed to the factory floor. For a classification task, this involves using metrics like Accuracy, Precision, Recall, F1-Score, and AUC-ROC, calculated on a held-out test dataset. The example simply stated "Evaluate the model on the predefined database."
    - Operational Considerations:

    - Building Models: Planned the model update cycle. The example specified "Retrain the model every month on the previous month's data," which is crucial for adapting to changing conditions and avoiding model drift.
    - Making Predictions: Defined the inference schedule: "Every day begin sorting the machine parts based on its probability of getting damaged." This specifies the frequency and purpose of the model's predictions.
    - Evaluation & Monitoring: Stipulated ongoing validation post-deployment: "Metric - Accuracy of the previous month's prediction." This creates a feedback loop to ensure the model remains accurate over time.

- Analysis of a Practical Example: Deconstructed the provided machine part failure prediction case study:
    - Input: Historical time-series data from sensors (vibration, thermal).
    - Output: A binary prediction (1: Failure Imminent, 0: Normal Operation).
    - Value: Move from reactive ("run-to-failure") or costly preventive maintenance to predictive maintenance.
    - Data Challenge: Understood the importance of the maintenance logbook as the source of truth for creating a supervised learning dataset.

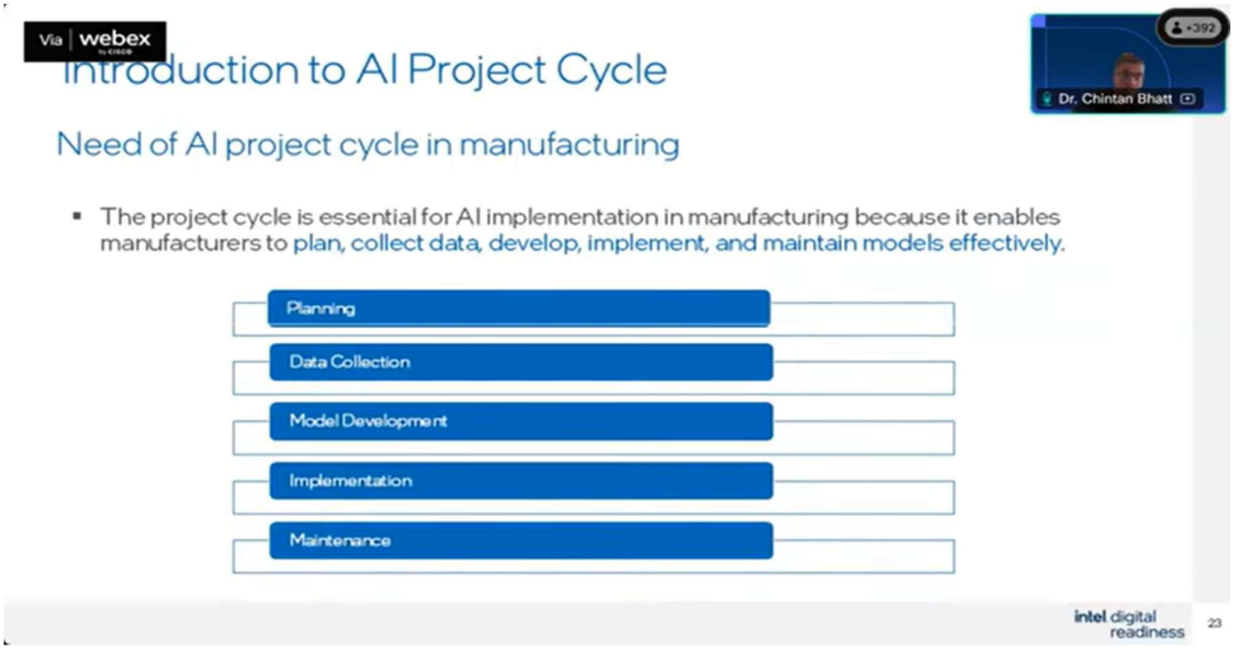
- Framework Application: Internalized that this template is a reusable tool for any AI project. It ensures that technical work is always tied to business value, data availability is confirmed, and deployment and monitoring are considered from the outset, drastically increasing the project's chances of success.

Figure 2.3.1: AI Project Cycle (PREDICT/EVALUATE) Template Analysis

Aspect	Question Answered	Example: Predictive Maintenance
ML Task	What is the input and output? What type of ML task is it?	**Input:** Sensor data (vibration, temperature) **Output:** Binary classification (Will Fail / Won't Fail) **Task:** Binary Classification
Value Proposition	What are you implementing and why? Who benefits?	**What:** Identify crucial parts for maintenance **Why:** Improve efficiency, reduce downtime **Who:** Maintenance team, production planners
Data Source	Which raw data sources can we use?	Sensor data feeds, machine maintenance logbooks
Data Collection	How do we get new labeled data?	Correlate sensor historical data with logbook entries of past failures
Offline Evaluation	How do we test the model before deployment?	Use historical data split into train/test sets. Calculate Accuracy, Precision, Recall
Features	What features are extracted from the raw data?	Mean vibration, temperature trends, max pressure, operating hours
Building Models	When and how often do we update the model?	Retrain the model every month on the latest data
Making Predictions	When and how often do we make predictions?	Every day to prioritize maintenance tasks
Evaluation & Monitoring	How do we check performance after deployment?	Metric – Accuracy of the previous month's prediction

The following table breaks down the machine part failure prediction

Day - 03



2.4 Day 4: Handling Data for AI Models

Activity Overview:

The focus of the third day shifted from pure programming to the critical methodology of executing an AI project. The activity involved a deep study of a structured template for planning Machine Learning projects, specifically focusing on the "PREDICT" (how) and "EVALUATE" (how well) phases. This framework is designed to ensure that AI initiatives are not just technically sound but are also aligned with business objectives and are deployable in a real-world manufacturing environment. The day centered on analyzing a provided example of predicting machine part failures and understanding how to apply this planning framework to any industrial AI problem.

Detailed Learning Outcomes:

Data Loading and Merging

- Worked with multiple datasets reflecting an e-commerce manufacturing pipeline, including orders, customers, sellers, products, and geolocation data.
- Learned to use Pandas merge() to perform database-style joins on keys such as order_id, customer_id, and seller_id.
- Practiced different join types (left, inner) to preserve data integrity and reconstruct a complete denormalized view of each order.

Data Cleaning and Imputation

- Systematically identified missing values in numerical, categorical, and datetime fields.
- Applied appropriate imputation strategies:
  - **Numerical data:** filled with the median to reduce sensitivity to outliers.
  - **Categorical data:** replaced missing values with placeholders like *"Unknown."*
  - **Datetime data:** used earliest available dates or business-logic defaults.
- Ensured consistency by converting columns to proper data types, such as transforming strings into datetime objects.

Exploratory Data Analysis (EDA)

- Summarized dataset structure and statistics using info() and describe().
- Visualized missing data patterns with a heatmap to spot quality issues.
- Conducted temporal analysis by plotting order distribution over time to uncover trends and seasonality.
- Explored correlations among numerical features (e.g., product dimensions vs. weight) to detect potential multicollinearity.

Feature Engineering

- Created **time-based features** such as wait time, purchase day of week, month, and year to capture temporal patterns.
- Derived **product features** like product volume (length × height × width) to better represent item characteristics.
- Developed a **geospatial feature** by estimating seller–customer distance with the Haversine formula as a proxy for delivery complexity.
- Defined the **target variable**, actual delivery time, from calculated wait times.

Encoding Categorical Variables

- Prepared categorical fields such as customer and seller states for modeling by converting them into numerical form using label encoding.

2.5 Day 5: Modeling & Evaluation of AI Systems

Activity Overview:

The fifth day transitioned from data preparation to the core of machine learning: building, evaluating, and interpreting models. The focus was on understanding how to measure the performance of AI systems accurately and objectively. The activities covered both regression (predicting continuous values like delivery time) and classification (categorizing data like defective/non-defective) tasks. This involved a deep dive into various evaluation metrics, their mathematical formulations, and, most importantly, their practical interpretation in a business context. A significant portion of the day was dedicated to the Confusion Matrix, a fundamental tool for diagnosing classification model performance.

Detailed Learning Outcomes:

- **Regression Metrics (For Predicting Continuous Values):**
  - **Mean Absolute Error (MAE):** Understood that MAE calculates the average absolute difference between the predicted values and the actual values. It provides a straightforward, interpretable measure of error in the same units as the target variable (e.g., days). Learned that it does not penalize large errors as heavily as other metrics.
    - **Formula:** MAE=1n∑i=1n|yi−y^i|$MAE=n1\sum i=1n|yi−y^i|$
    - **When to Use:** Ideal when all prediction errors are equally important. For example, an error of 2 days is twice as bad as an error of 1 day.
  - **Mean Squared Error (MSE) & Root Mean Squared Error (RMSE):** Understood that MSE calculates the average of the squares of the errors. This property means it disproportionately penalizes larger errors. RMSE is the square root of MSE, bringing the units back to the original scale of the target variable.
    - **Formula (MSE):** MSE=1n∑i=1n(yi−y^i)2$MSE=n1\sum i=1n(yi−y^i)2$
    - **Formula (RMSE):** RMSE=MSE$RMSE=MSE$
    - **When to Use:** Crucial when large errors are particularly undesirable and must be avoided at all costs. For instance, in predicting machine failure time, a very large error (predicting failure in 100 days when it happens in 10) could be catastrophic.
  - **R-squared (R^2) / Coefficient of Determination:** Learned that R^2 measures the proportion of the variance in the dependent variable (actual values) that is predictable from the independent variables (features). It provides a score of how well the model fits the data compared to a simple mean model.
    - **Interpretation:** An R^2 of 0.70 means that 70% of the variance in the target variable can be explained by the model's features.
    - **Limitation:** A high R^2 does not always mean a good model; it could be overfitting.

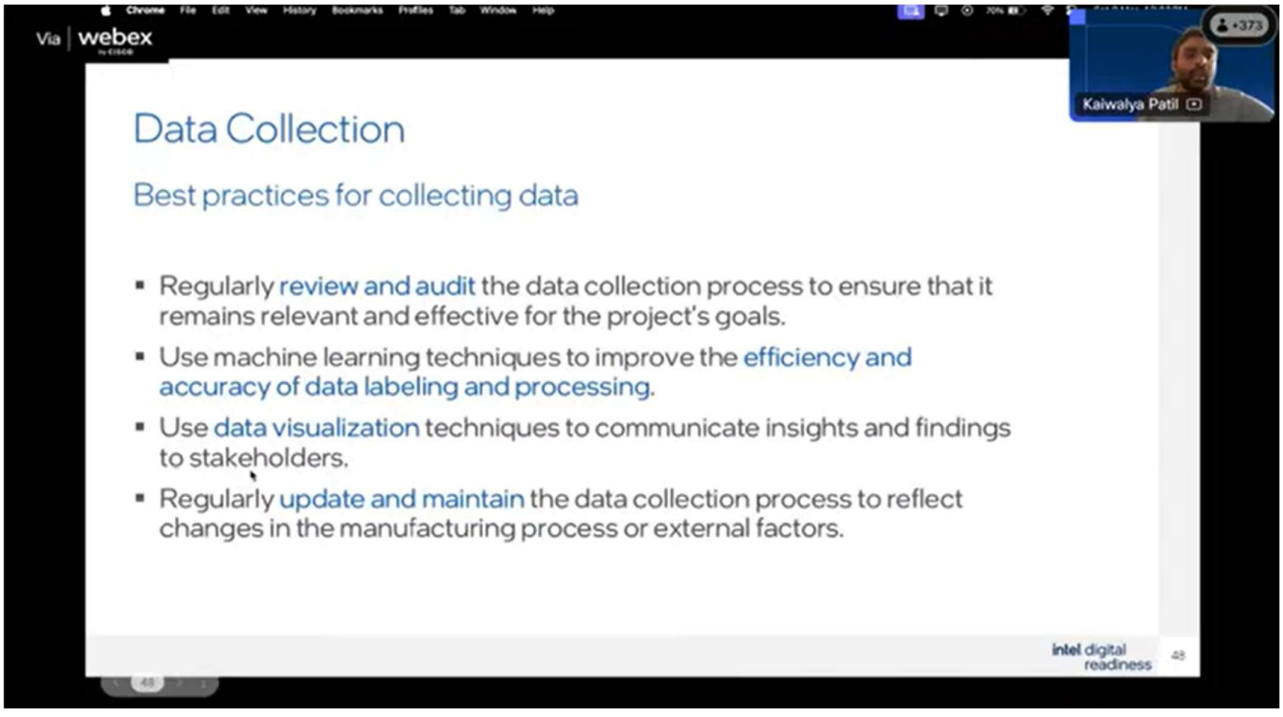
- **Classification Metrics & The Confusion Matrix:**
  - **The Confusion Matrix:** Mastered this essential 2x2 table that lays out the fundamental performance of a classification model by comparing actual vs. predicted classes. Defined its four components for a binary classifier:
    - **True Positives (TP):** The model correctly predicted the positive class (e.g., "Defective").
    - **True Negatives (TN):** The model correctly predicted the negative class (e.g.,

"Non-Defective").

- **False Positives (FP):** The model incorrectly predicted the positive class (Type I Error). (e.g., a good product is flagged as defective - costs money in false rework).
- **False Negatives (FN):** The model incorrectly predicted the negative class (Type II Error). (e.g., a defective product is missed - can damage brand reputation and customer safety).

- **Derived Metrics from Confusion Matrix:**
  - **Accuracy:** Overall, how often is the classifier correct? TP+TNTotal*TotalTP+TN*. Learned that accuracy can be misleading with imbalanced datasets (e.g., 95% non-defective, 5% defective).
  - **Precision:** When it predicts "positive," how often is it correct? TPTP+FP*TP+FPTP*. Measures the model's quality of positive predictions. **High precision is needed when the cost of FP is high** (e.g., initiating an unnecessary maintenance shutdown).
  - **Recall (Sensitivity):** How often does it correctly identify actual positives? TPTP+FN*TP+FNTP*. Measures the model's ability to find all positive samples. **High recall is needed when the cost of FN is high** (e.g., missing a critical defect or a machine failure).
  - **F1-Score:** The harmonic mean of Precision and Recall. 2×Precision×RecallPrecision+Recall2×*Precision*+*RecallPrecision*×*Recall*. Provides a single score that balances the trade-off between Precision and Recall. It is especially useful with uneven class distributions.

Day – 05



2.6 Day 6: AI Project Deployment with Streamlit

Activity Overview:

The sixth day marked the critical transition from theoretical modeling and local experimentation to practical, real-world application. The focus was on deployment—the process of making a trained machine learning model accessible to end-users through a user-friendly interface. The activity involved the end-to-end creation and deployment of "Timelytics," a web application that provides predictions for order-to-delivery times based on product and order characteristics. This process encapsulated the final step of the AI project lifecycle, turning a Python script into a live, interactive tool using the **Streamlit** framework and deploying it to the cloud via **Streamlit Community Cloud**.

Detailed Learning Outcomes:

- **Model Finalization and Serialization:**
  - **Model Training:** Finalized the **XGBoost Regressor** model trained on the meticulously processed dataset from Day 4. The model was configured to predict the target variable delivery_days.
  - **Serialization:** Learned the crucial step of model persistence using the joblib library. Saved the trained model object to a file (xgboost_otd_model.pkl) and the fitted LabelEncoder objects to another file (label_encoders.pkl). This allows the model to be loaded and used for predictions in a different environment (like a web server) without the need to retrain it, which is computationally expensive.
- **Web Application Development with Streamlit:**
  - **Framework Introduction:** Discovered **Streamlit**, an open-source Python framework that dramatically simplifies the process of building custom web apps for data science and machine learning. Its core advantage is the ability to create a rich UI with minimal code, using pre-built components for user input and display.
  - **Front-End Development:** Built the streamlit_app.py file, which defines the application's interface and logic:
    - **User Inputs:** Created various input widgets to capture all necessary features for the model:
      - st.sidebar.number_input() for numerical values (e.g., product_weight_g, product_size_cm3).
      - st.sidebar.selectbox() for categorical variables (e.g., customer_state, seller_state), dynamically populating the dropdown choices from the classes learned by the LabelEncoder.
      - st.sidebar.selectbox() for shipping method ('Express', 'Standard').
    - **Prediction Logic:** Wrote a function that:
      1. Captures all user inputs from the widgets.
      2. Loads the serialized model and label encoders.
      3. Preprocesses the inputs identically to the training data (e.g., encoding categorical variables using the pre-fitted encoders).
      4. Forms a feature vector and passes it to the model's .predict() method.
      5. Returns the prediction (e.g., 12.66 days).
    - **UI and Display:** Used st.title(), st.markdown(), and st.success() to create a clean and informative layout. Added a button (st.sidebar.button("Predict

Delivery Time")) to trigger the prediction, ensuring the app doesn't rerun on every input change.

- **Cloud Deployment and DevOps Basics:**
  - **Version Control:** Initialized a Git repository and committed all project files, including the Python script, serialized model files, and a requirements.txt file listing all dependencies.
  - **Dependency Management:** Created the requirements.txt file to ensure the deployment environment has all necessary libraries (e.g., streamlit, pandas, numpy, scikit-learn, xgboost, joblib). This is critical for reproducibility.
  - **Platform Deployment:** Learned the process of deploying the application on **Streamlit Community Cloud**:
    1. Pushed the code repository to **GitHub**.
    2. Signed into Streamlit Community Cloud and selected "New app".
    3. Connected my GitHub account, selected the repository, branch, and specified the main file path (streamlit_app.py).
    4. Clicked "Deploy". The platform automatically read the requirements.txt file, built the environment, and launched the application, providing a public URL (e.g., https://timelytics.streamlit.app/).
- **Testing and Validation:**
  - **Local Testing:** Ran the app locally using the command streamlit run streamlit_app.py to debug and ensure all functionality worked correctly before deployment.
  - **Post-Deployment Testing:** Accessed the live public URL and tested the application with various input combinations to verify that the cloud-deployed version was functioning identically to the local version and providing sensible predictions.

Day – 06

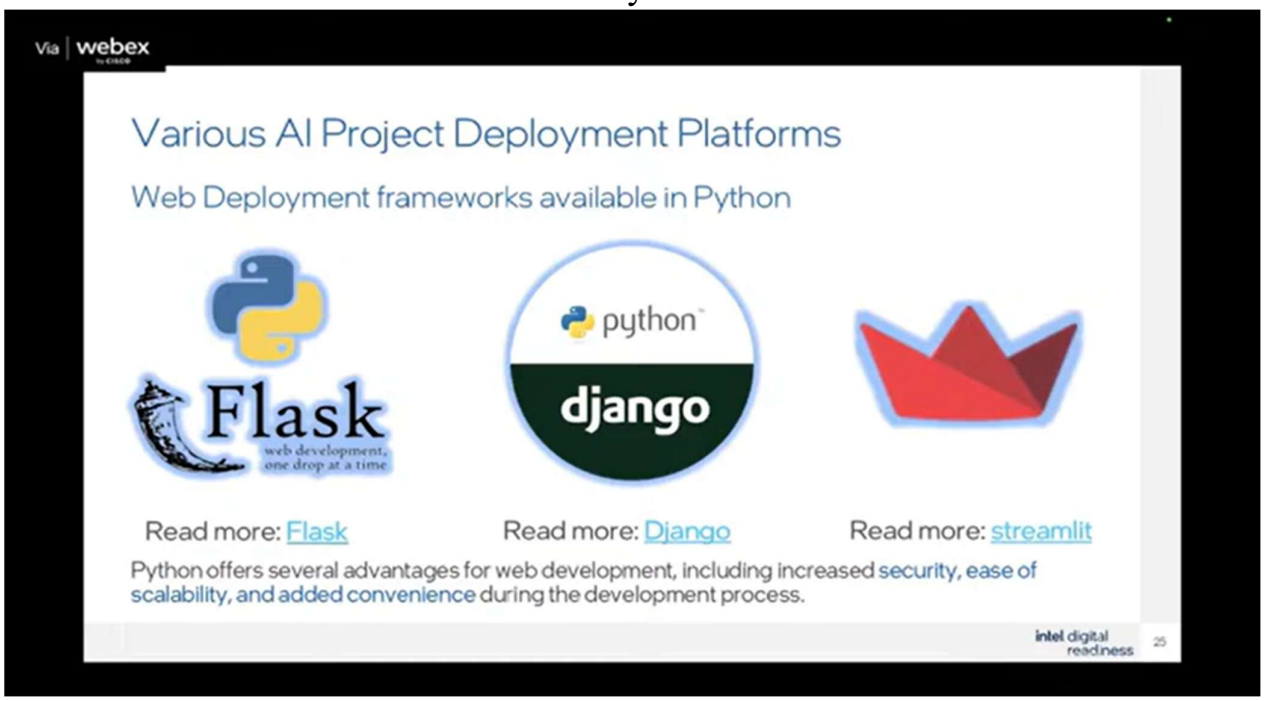
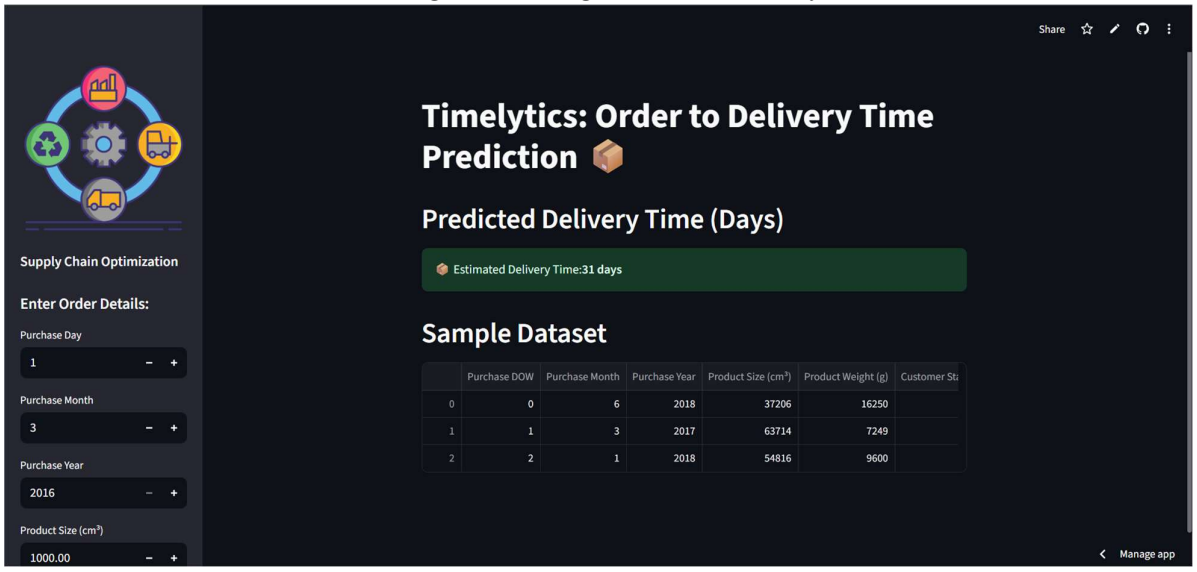


Fig 2.6.1 Assignment – 01 Timelytics



App.py code snippet:

```
import streamlit as st
import pandas as pd
import joblib
import numpy as np

model = joblib.load("models/xgboost_otd_model.pkl")
label_encoders = joblib.load("models/label_encoders.pkl")
st.title("Timelytics: Order to Delivery Time Prediction 📦")
st.sidebar.image("assets/supply_chain_optimisation.jpg",
                 use_container_width=True)  # Add your image here
st.sidebar.markdown("### Supply Chain Optimization")
st.sidebar.header("Enter Order Details:")
purchase_day = st.sidebar.number_input("Purchase Day",
                                       min_value=0,
                                       max_value=6,
                                       step=1)
purchase_month = st.sidebar.number_input("Purchase Month",
                                         min_value=1,
                                         max_value=12,
                                         step=1)
purchase_year = st.sidebar.number_input("Purchase Year",
                                        min_value=2016,
                                        max_value=2025,
                                        step=1)
product_size = st.sidebar.number_input("Product Size (cm³)",
                                       min_value=0.0,
                                       step=1.0)
product_weight = st.sidebar.number_input("Product Weight (g)",
                                         min_value=0.0,
                                         step=1.0)
customer_state = st.sidebar.selectbox(
    "Customer State", label_encoders["customer_state"].classes_)
seller_state = st.sidebar.selectbox("Seller State",
                                    label_encoders["seller_state"].classes_)
shipping_method = st.sidebar.selectbox(
```

```
    "Shipping Method", label_encoders["shipping_method"].classes_)
customer_state_encoded = label_encoders["customer_state"].transform(
    [customer_state])[0]
seller_state_encoded = label_encoders["seller_state"].transform([seller_state
                                                                 ])[0]
shipping_method_encoded = label_encoders["shipping_method"].transform(
    [shipping_method])[0]
input_data = pd.DataFrame(
    [[
        purchase_day,
        purchase_month,
        purchase_year,
        product_size,
        product_weight,
        customer_state_encoded,
        seller_state_encoded,
        shipping_method_encoded,
    ]],
    columns=[
        "purchase_day",
        "purchase_month",
        "purchase_year",
        "product_size",
        "product_weight",
        "customer_state",
        "seller_state",
        "shipping_method",
    ],
)
st.markdown("## Predicted Delivery Time (Days)")
if st.sidebar.button("Predict Delivery Time"):
    prediction = model.predict(input_data)
    st.success(
        f"📦 Estimated Delivery Time:**{int(np.round(prediction[0]))} days**")
st.markdown("## Sample Dataset")
sample_data = pd.DataFrame({
    "Purchase DOW": [0, 1, 2],
    "Purchase Month": [6, 3, 1],
    "Purchase Year": [2018, 2017, 2018],
    "Product Size (cm³)": [37206, 63714, 54816],
    "Product Weight (g)": [16250, 7249, 9600],
    "Customer State": [25, 25, 25],
    "Seller State": [20, 7, 20],
    "Distance (km)": [247.94, 250.35, 4.915],
})
st.dataframe(sample_data)
```

2.7 Day 7: Computer Vision in Manufacturing

Activity Overview:

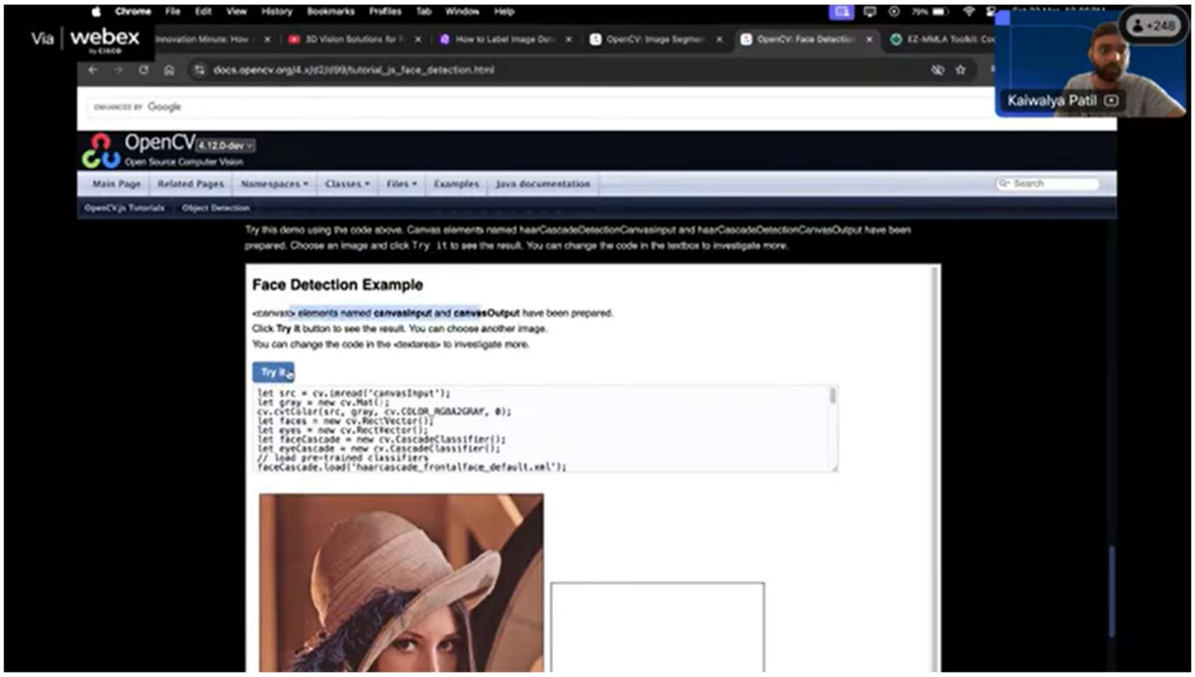
The seventh day explored the application of **Computer Vision (CV)**, a field of artificial intelligence that enables computers to interpret and understand visual information from the world. The focus was on its transformative role in automating visual inspection tasks within manufacturing. The activity provided a hands-on, practical introduction to building a CV model without writing complex code, using Google's **Teachable Machine**. Additionally, it involved strategic planning by researching and documenting real-world use cases for CV in an industrial setting, analyzing their impact, and considering the associated ethical implications.

Detailed Learning Outcomes:

- **Understanding Computer Vision in Industry:**
  - **The Automation of Sight:** Understood that CV allows machines to perform visual tasks that typically require human sight, but with greater consistency, speed, and accuracy. This is particularly valuable in manufacturing for quality control.
  - **Beyond Traditional Inspection:** Recognized that manual inspection is time-consuming, prone to human error and fatigue, and difficult to scale. CV systems provide a scalable, objective, and continuous alternative.
  - **The Role of Deep Learning:** Learned that modern CV is powered by deep learning models, particularly **Convolutional Neural Networks (CNNs)**, which can learn complex patterns and features directly from images, making them highly effective for defect detection.
- **Hands-On Model Building with Teachable Machine:**
  - **No-Code AI Tool:** Gained proficiency with Google's Teachable Machine, a web-based tool that democratizes AI by allowing users to create machine learning models through a simple, intuitive interface.
  - **Project Workflow:** Followed the entire process of creating an image classification project:
    1. **Project Creation:** Selected "Image Project" and chose a "Standard" model for high accuracy.
    2. **Class Definition:** Created classes representing the categories to be identified (e.g., "Defective", "Non-Defective"; or "Cardboard", "Glass", "Metal", "Plastic").
    3. **Data Collection:** For each class, uploaded a set of example images. This dataset is crucial for teaching the model the visual characteristics of each class.
    4. **Model Training:** Clicked "Train Model". The tool automatically handled the complex process of training a neural network in the browser.
    5. **Model Testing:** Used the built-in webcam or uploaded test images to instantly see the model's predictions and confidence levels, allowing for quick iterative testing.
    6. **Model Export:** Exported the trained model in various formats, including TensorFlow.js for web apps, TensorFlow Lite for mobile/edge devices, and a downloadable Keras model (keras_model.h5) for use in Python environments. This step is key for deployment and integration into larger systems.

- **Use Case Analysis and Project Scoping:**
    - Researched and documented three high-impact use cases for CV in manufacturing, structuring each with a problem statement, proposed CV solution, expected impact, and ethical considerations:
        1. **Defect Detection:**
            - **Problem:** Manual inspection is slow, inconsistent, and can miss subtle defects.
            - **Solution:** CV systems with high-resolution cameras automatically scan products on the production line. AI models analyze images in real-time to detect anomalies like cracks, scratches, misalignments, or color mismatches.
            - **Impact:** Increased inspection accuracy and speed, reduced cost of quality, and improved customer satisfaction.
            - **Ethical Considerations:** Data privacy of production line images; potential job displacement for human inspectors.
        2. **Predictive Maintenance:**
            - **Problem:** Unexpected machine failures cause costly unplanned downtime.
            - **Solution:** CV cameras monitor equipment for visual signs of wear and tear (e.g., corrosion, leaks, structural cracks) or misalignment. Combined with IoT sensor data, this provides a comprehensive health monitoring system.
            - **Impact:** Reduced downtime, extended machinery life, and improved worker safety by identifying potential hazards.
            - **Ethical Considerations:** Ensuring the safety and reliability of the system to prevent accidents it's meant to avoid.

Day – 07



2.8 Day 8: Predictive Maintenance

Activity Overview:

The eighth day focused on **Predictive Maintenance (PdM)**, a transformative application of AI in manufacturing that moves beyond traditional reactive (fixing equipment after it fails) or preventive (scheduled maintenance regardless of condition) approaches. The activity involved a detailed case study analysis of Tata Group's implementation of PdM across its manufacturing units (Tata Steel, Tata Motors). This provided a real-world, large-scale perspective on the implementation process, technologies involved, achieved results, and future trends. The day emphasized how IoT, AI, and data analytics converge to create intelligent maintenance systems.

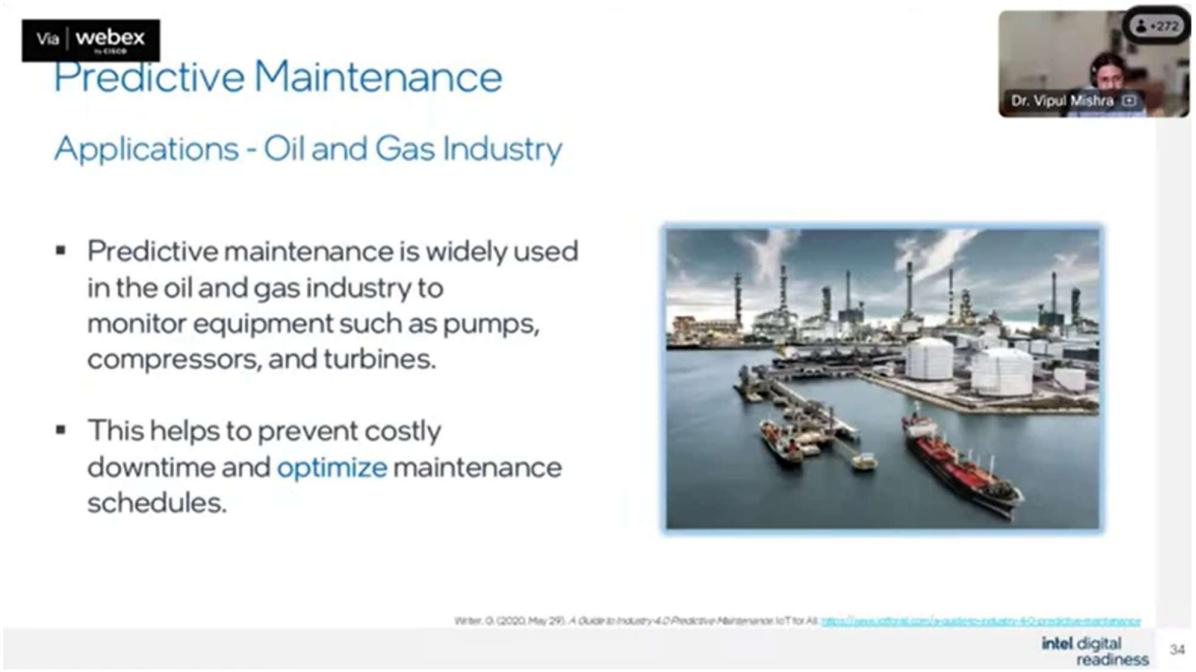
Detailed Learning Outcomes:

- **Understanding Predictive Maintenance:**
  - **Paradigm Shift:** Understood that PdM uses data-driven insights to predict equipment failures before they occur, allowing maintenance to be performed just in time. This optimizes resource use and maximizes asset uptime.
  - **Contrast with Traditional Methods:** Contrasted PdM with:
    - **Reactive Maintenance:** Runs to failure. High cost of unplanned downtime and secondary damage.
    - **Preventive Maintenance:** Time-based or usage-based schedules. Often leads to unnecessary maintenance and parts replacement, wasting resources.
  - **Core Principle:** PdM is based on the concept that machinery provides observable warning signs (via data) before a catastrophic failure occurs.
- **Implementation Process (The "How"):**
  - **IoT Sensor Integration:** Learned that the foundation of PdM is data acquisition. Companies like Tata integrate a network of **IoT-enabled sensors** onto critical machinery to continuously monitor key health parameters:
    - **Vibration Analysis:** Abnormal vibrations can indicate misalignment, imbalance, or bearing wear.
    - **Thermal Imaging:** Rising temperatures can signal friction, cooling system failure, or electrical issues.
    - **Acoustic Analysis:** Unusual sounds can detect leaks, cavitation, or component failure.
    - **Pressure and Flow Rates:** Deviations can point to blockages or pump failures.
  - **Data Acquisition and Cloud Processing:** Understood the data pipeline: sensors collect real-time data → data is transmitted to a cloud platform (e.g., AWS, Azure) → where it is stored and prepared for analysis.
  - **AI and Machine Learning Analysis:** Explored how AI algorithms are the core of prediction:
    - **Anomaly Detection:** Models learn "normal" operational baselines and flag significant deviations.
    - **Supervised Learning:** Using historical data where failures are labeled, models learn the patterns that precede specific types of failures.
    - **Prognostics:** Advanced models not only predict failure but also estimate the **Remaining Useful Life (RUL)** of a component.

- **Alerting and Automated Scheduling:** Learned that the system automatically generates alerts for maintenance teams when a potential issue is detected, often integrating with **Computerized Maintenance Management Systems (CMMS)** to schedule work orders proactively.

- **Case Study Analysis: Tata Group:**
  - **Results Achieved:** Analyzed the tangible business outcomes reported by Tata, which demonstrate PdM's ROI:
    - **Reduction in Downtime:** Significant decrease in unplanned production stoppages.
    - **Cost Savings:** Tata Steel reported an estimated **20% reduction in maintenance costs** by avoiding major repairs and optimizing spare part inventory.
    - **Extended Equipment Life:** Timely interventions based on actual condition prevent accelerated wear and tear.
    - **Improved Safety:** Preventing unexpected equipment failures reduces workplace accidents.
  - **Impact on Operations:** Understood the broader organizational impact:
    - **Increased Productivity:** Tata Motors improved vehicle output by meeting production targets more reliably.
    - **Sustainable Manufacturing:** Optimized maintenance reduces energy waste and resource consumption.
    - **Enhanced Customer Satisfaction:** Stable production levels ensure on-time delivery to customers.
- **Future Trends in PdM:**
  - **Digital Twins:** Creating a high-fidelity virtual model of a physical asset. The digital twin can simulate operational scenarios and predict failures without risking the actual equipment, allowing for incredibly sophisticated testing and prognostics.
  - **AI & Edge Computing:** Moving AI models directly onto sensors or local gateways (**edge devices**) to process data locally. This reduces latency, allowing for **real-time decision-making** (e.g., instantly shutting down a machine if an imminent failure is detected) and reduces bandwidth costs by only sending important insights to the cloud.
  - **5G-powered Smart Factories:** The high speed and low latency of 5G networks enable faster and more reliable communication between a massive number of IoT devices and central systems, creating a more responsive and interconnected factory environment.
  - **Blockchain for Maintenance Logs:** Using blockchain technology to create secure, tamper-proof, and transparent records of all maintenance activities. This ensures data integrity, simplifies audits, and builds trust in the maintenance history of critical assets.

Day – 08


Via | webex
Predictive Maintenance
Applications - Oil and Gas Industry
Predictive maintenance is widely used in the oil and gas industry to monitor equipment such as pumps, compressors, and turbines.
This helps to prevent costly downtime and optimize maintenance schedules.
Dr. Vipul Mishra
+272
intel digital readiness
34

2.9 Day 9: Robotic Process Automation (RPA)

Activity Overview:

The ninth day shifted focus from physical automation on the production line to automation in the back office and knowledge work through **Robotic Process Automation (RPA)**. The activity involved building a strategic business case for adopting RPA within a manufacturing organization. This required identifying repetitive, rule-based processes ideal for automation, forming a cross-functional team to assess feasibility, developing a clear implementation strategy, and creating a prioritization framework to ensure the highest ROI initiatives are tackled first. The culmination was the creation of a persuasive presentation aimed at stakeholders to secure investment in RPA technology.

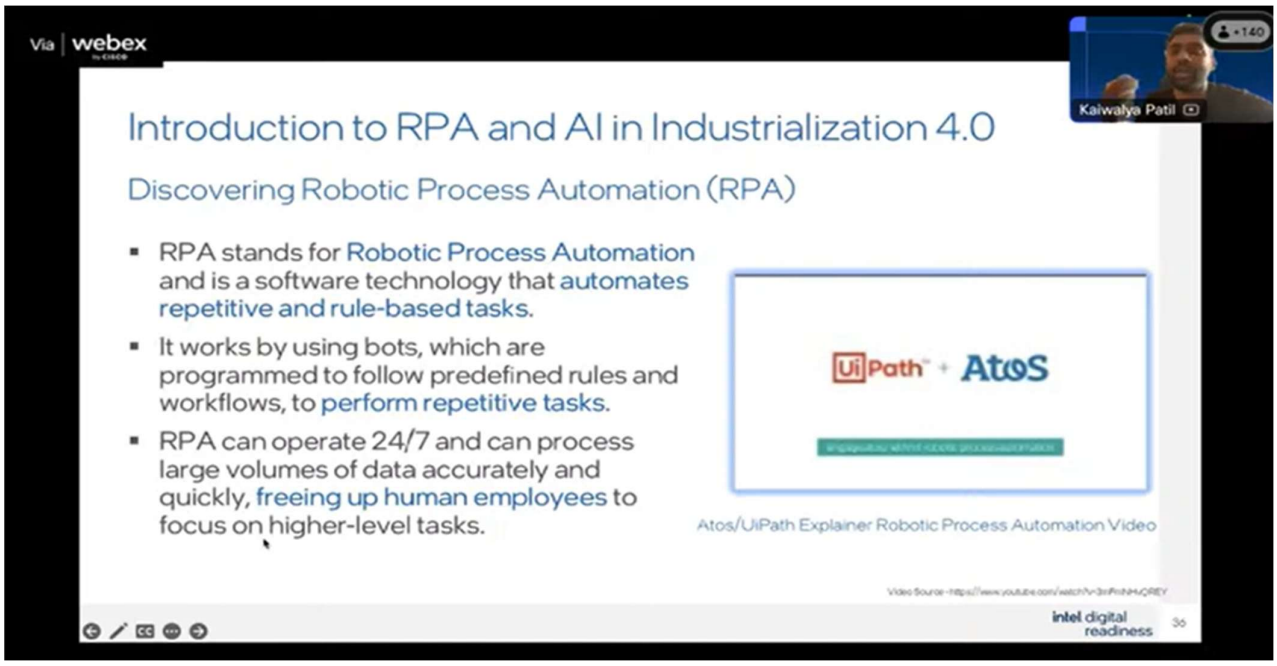
Detailed Learning Outcomes:

- **Understanding RPA and Its Value Proposition:**
  - **What is RPA?** Learned that RPA uses software "bots" to mimic human actions interacting with digital systems and software. These bots can execute rule-based tasks across applications just like a human would (e.g., logging into systems, copying data, filling forms, processing transactions).
  - **Key Differentiator:** Understood that RPA operates at the **user interface (UI) level**, unlike AI which often works with data and APIs. This allows it to automate tasks in legacy systems that lack modern integration capabilities.
  - **The Manufacturing Back-Office:** Identified that manufacturing involves numerous repetitive, high-volume administrative processes beyond the factory floor that are perfect for RPA, leading to significant efficiency gains and error reduction.
- **Identifying and Prioritizing Automation Candidates:**
  - **Process Selection Criteria:** Learned to evaluate processes based on three key factors to build a business case:
    1. **High Volume:** Tasks performed frequently (e.g., daily, with every order).
    2. **Rule-Based:** Tasks that follow a clear, logical set of rules without requiring subjective judgment.
    3. **Stable and Mature:** Processes that are well-defined and unlikely to change frequently.
  - **Manufacturing-Specific Use Cases:** Identified and documented prime candidates for RPA in a manufacturing context:
    - **Invoice and Purchase Order Processing:** Bots can extract data from incoming invoices/POs, validate it against system records, and enter it into an ERP system.
    - **Inventory Management Updates:** Automatically updating stock levels across systems, generating low-stock alerts, and even placing orders with suppliers.
    - **Quality Control and Compliance Reporting:** Automatically compiling inspection data from various sources into standardized reports for regulators.
    - **Bill of Materials (BoM) Generation:** Automating the creation of structured BOM documents from design or ERP systems.
    - **Machine Data Logging:** Bots can log into machine interfaces, extract operational data, and compile it into reports, acting as a bridge between old machinery and modern data platforms.

- **Prioritization Matrix:** Developed a framework to score and rank potential RPA projects based on **Complexity** (ease of implementation), **Volume** (how often the task is performed), and **Impact** (how much time/money it will save or errors it will reduce). This ensures resources are invested in the most valuable projects first (e.g., High Volume, Low Complexity, High Impact tasks are ideal starters).

- **Developing an RPA Implementation Strategy:**
  - **Cross-Functional Team Formation:** Understood that successful RPA requires collaboration between IT (for infrastructure, security, and integration), operations (for process knowledge), and process experts (to map and design the workflows).
  - **Phased Rollout (Pilot First):** Learned the importance of starting with a small, well-defined pilot project to demonstrate quick wins, build confidence, and work out implementation kinks before scaling across the organization.
  - **Tool Selection:** Explored factors for choosing an RPA platform (e.g., UiPath, Automation Anywhere, Blue Prism), including ease of use for "citizen developers," scalability, security, and compatibility with existing software.
  - **Change Management:** Emphasized the need to address employee concerns about automation, positioning RPA as a tool to eliminate mundane work and allow employees to focus on higher-value, strategic tasks.
- **Building the Business Case:**
  - **ROI Analysis:** Learned to calculate Return on Investment by estimating time savings (e.g., hours saved per task multiplied by employee cost), error reduction costs, and improved processing speed.
  - **Stakeholder Persuasion:** Structured a presentation to highlight not just cost savings, but also benefits like improved compliance (fewer errors in reports), enhanced scalability (handling volume increases without more staff), and better employee satisfaction (removing tedious work).

Day – 09



2.10 Day 10: Inferencing on Edge

Activity Overview:

The tenth day explored the cutting-edge paradigm of **Edge Inferencing**, which involves running machine learning models directly on devices ("edge devices") at the source of data generation, rather than sending data to a centralized cloud server for processing. The activity focused on analyzing how this approach specifically benefits wearable health monitors, but the principles directly apply to industrial IoT (IIoT) devices in manufacturing. The day involved understanding the architecture, trade-offs, and significant advantages of edge computing for AI applications requiring low latency, enhanced privacy, and operational reliability.

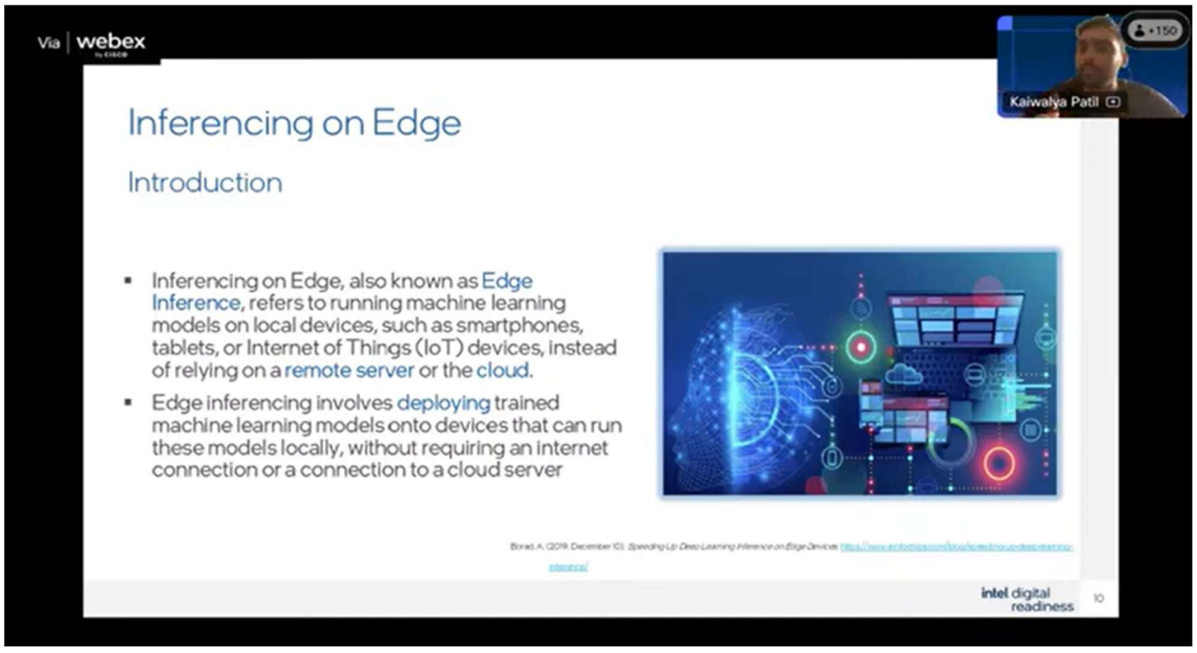
Detailed Learning Outcomes:

- **Understanding Edge vs. Cloud Architecture:**
  - **Cloud Inferencing:** The traditional model where data from sensors/devices is transmitted over the network to a powerful cloud server. The server runs the ML model and sends the prediction result back to the device. This can be slow and bandwidth-intensive.
  - **Edge Inferencing:** The ML model is deployed directly onto the device itself (e.g., a smartwatch, a camera, a vibration sensor). Data is processed locally, and only the results (or meaningful insights) are communicated to the cloud. This reduces latency and bandwidth use.
  - **Hybrid Approaches:** Understood that many systems use a combination, where simple, urgent decisions are made on the edge, while complex analysis and model retraining occur in the cloud.
- **Key Advantages of Edge Inferencing for Manufacturing:**
  - **Real-Time Response and Low Latency:** Learned that for time-sensitive applications, the delay (latency) of sending data to the cloud and back is unacceptable.
    - **Industrial Example:** A robotic arm on an assembly line using computer vision to identify defective parts must make a split-second decision to reject an item. Edge processing enables this instant response, whereas cloud latency could halt the production line.
  - **Enhanced Data Privacy and Security:** Recognized that sensitive data never leaves the local device.
    - **Industrial Example:** Video footage from a factory floor monitoring worker productivity or proprietary assembly processes can be analyzed on-site. Only anonymized productivity metrics or alerts are sent to the cloud, protecting privacy and intellectual property.
  - **Offline Operation and Reliability:** Understood that edge devices can function fully without a constant internet connection.
    - **Industrial Example:** A predictive maintenance system on an offshore oil rig or in a remote mine can continue to monitor equipment and provide alerts even if the satellite link goes down.
  - **Reduced Bandwidth and Cost:** Processed results (e.g., "Bearing X has a 95% failure probability") are tiny compared to the raw data stream (e.g., continuous HD video or high-frequency vibration data). This drastically reduces cloud storage and bandwidth costs.

- **Improved Power Efficiency:** For battery-powered IoT sensors, constantly transmitting raw data is power-intensive. Processing data locally and only sending occasional insights can extend battery life from days to years.

- **Technical Considerations and Challenges:**
  - **Hardware Constraints:** Edge devices have limited processing power, memory, and battery life compared to cloud servers. This necessitates the use of **model optimization** techniques:
    - **Quantization:** Reducing the numerical precision of the model's weights (e.g., from 32-bit floating point to 8-bit integers). This shrinks model size and speeds up inference with a minor trade-off in accuracy.
    - **Model Pruning:** Removing unnecessary neurons or connections from a neural network that don't significantly contribute to its predictions.
    - **Hardware Accelerators:** Using specialized chips like Google's Edge TPU, Intel's Movidius VPU, or NVIDIA's Jetson modules designed specifically for efficient ML inference at the edge.
  - **Model Deployment:** Learned about frameworks like **TensorFlow Lite** and **ONNX Runtime** that are designed to convert and run large models efficiently on resource-constrained edge devices.
- **Application to Wearable Health Monitors (Case Study):**
  - Analyzed how the principles above apply to health tech, which mirrors industrial sensor applications:
    - **Real-Time Alerts:** Instant detection of atrial fibrillation or a fall, triggering immediate haptic feedback to the user.
    - **Data Privacy:** Sensitive heart rate and activity data is processed on the watch, with only summary statistics sent to the phone/cloud.
    - **Offline Operation:** Health monitoring continues during a run or swim where there's no phone connection.
    - **Battery Life:** On-device processing is more efficient than streaming raw biometric data 24/7.

Day – 10



2.11 Day 11: Demand Forecasting

Activity Overview:

The eleventh day focused on **Demand Forecasting**, a critical application of AI that enables manufacturing organizations to predict future customer demand for their products. The activity involved exploring best practices for supply chain operations, performing a practical forecasting exercise using an online tool, and analyzing AI-powered forecasting tools used in the industry. The day emphasized how moving from traditional statistical methods to AI-driven forecasting can significantly enhance accuracy, reduce costs, and improve overall supply chain resilience.

Detailed Learning Outcomes:

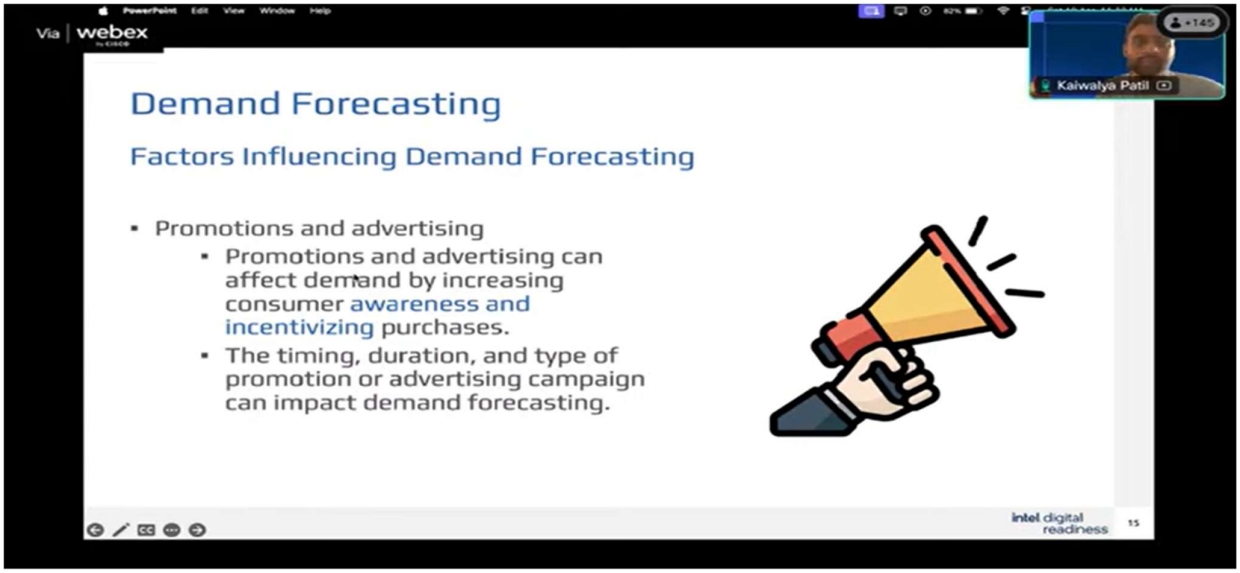
- **Foundations of Demand Forecasting:**
  - **Purpose and Impact:** Understood that accurate demand forecasting is the cornerstone of efficient supply chain management. It informs critical decisions in production planning, inventory management, procurement, and logistics, directly impacting costs, customer satisfaction, and revenue.
  - **Challenges with Traditional Methods:** Recognized the limitations of simple historical averaging or manual forecasting, which often fail to capture complex patterns, seasonality, and the impact of external factors (promotions, weather, competitors, economic shifts).
  - **AI Revolution:** Learned how AI and machine learning transform forecasting by automatically analyzing vast, diverse datasets to identify subtle patterns and relationships that humans cannot easily discern.
- **Best Practices for Supply Chain Forecasting:**
  - **Data Integration:** Emphasized the need to combine **historical sales data** with **external signals** such as:
    - **Calendar Events:** Holidays, weekends, festivals.
    - **Promotional Activities:** Marketing campaigns, discounts, sales events.
    - **Market Intelligence:** Competitor pricing, economic indicators, social media trends.
    - **Weather Data:** Especially relevant for seasonal products (e.g., beverages, apparel).
  - **Cross-Functional Collaboration:** Highlighted that effective forecasting requires input from sales, marketing, finance, and supply chain teams to incorporate domain knowledge and upcoming plans.
  - **Continuous Monitoring and Refinement:** Stressed the importance of regularly measuring forecast accuracy (using metrics like MAPE - Mean Absolute Percentage Error) and retraining models with new data to adapt to changing market conditions.
  - **Scenario Planning:** Advocated for using AI models to run "what-if" simulations (e.g., What if we launch a 20% discount? What if a key supplier is delayed?) to build more resilient supply chains.
- **Hands-On Forecasting Exercise:**
  - **Tool Proficiency:** Used the **Call Centre Helper Forecasting Tool**, an online time-series forecasting calculator, to gain practical experience with demand prediction.
  - **Process:**
    1. Prepared time-based historical data (monthly contact volume for a call

center, analogous to monthly product demand).
2. Input the data into the tool, which automatically applied a forecasting model (Triple Exponential Smoothing).
3. Analyzed the output: a forecast graph showing historical data, the trend line, and future predictions with confidence intervals.
4. Interpreted the results to understand expected future demand and its potential variability.

- **Key Takeaway:** Experienced how even simple automated forecasting tools can provide more accurate and objective predictions than manual estimates.

- **AI-Powered Forecasting Tools:**
  - **Platform Analysis:** Researched enterprise-level AI forecasting solutions used in manufacturing, such as **SAP Integrated Business Planning (IBP)**, **Oracle Demantra**, and **Blue Yonder**.
  - **Advanced Features:** Identified capabilities that set AI tools apart:
    - **Automated Pattern Recognition:** ML algorithms automatically detect seasonality, trends, and cyclical patterns without manual configuration.
    - **Integration with IoT:** Incorporating real-time data from production lines and warehouse sensors.
    - **Explainable AI:** Providing insights into *why* a certain forecast was generated (e.g., "The forecast increased due to a similar promotional pattern last year").
    - **Collaborative Workflows:** Allowing multiple stakeholders to adjust forecasts based on their intelligence and reach a consensus.
  - **Benefits Documented:** Listed the tangible advantages for manufacturers:
    - **Improved Forecast Accuracy (20-50%+ reduction in error):** Leading to fewer stockouts and less excess inventory.
    - **Faster Response to Market Changes:** Models quickly incorporate new sales data and external events.
    - **Optimized Inventory Levels:** Reducing carrying costs and freeing up working capital.
    - **Enhanced Customer Satisfaction:** Ensuring product availability when and where customers want it.

Day – 11



2.12 Day 12: AI Ethics and Responsible AI

Activity Overview:

The twelfth day addressed the critical and increasingly important domain of **AI Ethics and Responsible AI**. The activity moved beyond technical implementation to focus on the societal, legal, and moral implications of deploying AI systems. It involved analyzing the impact of emerging global AI regulations on the manufacturing sector and conducting a deep dive into the **Microsoft Responsible AI Toolbox**, a suite of open-source tools designed to help developers build fair, interpretable, reliable, and ethical AI systems. The day emphasized that responsible practices are not an obstacle but a essential component of sustainable and trustworthy AI innovation.

Detailed Learning Outcomes:

- **The Imperative for Responsible AI:**
  - **Understanding the Risks:** Recognized that AI systems, if not carefully designed and monitored, can perpetuate or even amplify existing biases, make unexplainable "black box" decisions, violate privacy, and cause significant harm if they fail in critical applications (e.g., autonomous vehicles, medical diagnostics, industrial control).
  - **Beyond Accuracy:** Learned that a model's high accuracy score is insufficient for deployment. It must also be **fair** (non-discriminatory), **interpretable** (understandable to humans), **robust** (resistant to manipulation or data drift), and **private** (protecting sensitive data).
  - **The Societal Contract:** Understood that as AI becomes more pervasive, manufacturers have an ethical and legal responsibility to ensure their technologies are safe and beneficial for employees, customers, and society at large.
- **Impact of Global AI Regulations:**
  - **Regulatory Landscape:** Analyzed how regulations like the **EU's AI Act** (which classifies AI systems by risk and imposes strict requirements on high-risk applications), the **U.S. Algorithmic Accountability Act**, and similar frameworks globally are shaping AI development.
  - **Manufacturing Compliance Requirements:** Identified specific obligations for manufacturing companies using AI:
    - **Transparency and Documentation:** Maintaining detailed records of how AI models are developed, trained, and deployed ("AI governance").
    - **Human Oversight:** Ensuring a human is in the loop for critical decisions, especially in high-risk areas like quality control or safety monitoring.
    - **Bias Audits and Fairness Assessments:** Proactively testing for and mitigating unwanted bias in AI systems, particularly those used in HR (hiring) or customer-facing applications.
    - **Robustness and Security:** Implementing measures to ensure AI systems are secure against cyber attacks and perform reliably under expected conditions.
  - **Cost of Non-Compliance:** Understood that failure to comply can result in massive financial penalties (up to 6% of global turnover under the EU AI Act), reputational damage, and loss of market access.

- **Microsoft Responsible AI Toolbox in Practice:**

- **Toolbox Overview:** Explored the suite of tools designed to operationalize responsible AI principles throughout the machine learning lifecycle:
  1. **InterpretML:**
     - **Purpose:** Provides techniques to explain **how** and **why** AI models make their predictions, moving away from "black box" models.
     - **Application:** In manufacturing, this could explain why a predictive maintenance model flagged a specific machine for failure (e.g., "This prediction was driven primarily by a 50% increase in vibration amplitude over the last 48 hours"). This builds trust with maintenance engineers.
  2. **Fairlearn:**
     - **Purpose:** Assesses and improves the **fairness** of AI systems. It evaluates model performance across different groups (e.g., different demographics, product lines, factory locations) to identify unfair disparities.
     - **Application:** Ensuring an AI-based hiring tool for factory positions does not discriminate based on gender or ethnicity. Or ensuring a quality control model performs equally well on products from all production lines.
  3. **Error Analysis:**
     - **Purpose:** Helps developers **debug** their models by identifying specific data cohorts or feature ranges where the model performs poorly (e.g., high error rates).
     - **Application:** Discovering that a visual inspection model has a very high false negative rate (misses defects) only on products with a specific shiny surface finish, indicating a need for more training data for that specific scenario.
  4. **DiCE (Diverse Counterfactual Explanations):**
     - **Purpose:** Generates "what-if" explanations. It shows users the minimal changes needed to input data to get a different model outcome.
     - **Application:** A product is flagged as high risk for failure. DiCE could show, "If the tolerance on component X were reduced by 0.1mm, this product would have been classified as low risk." This provides actionable guidance for design or process improvement.
  5. **EconML:**
     - **Purpose:** Goes beyond prediction to measure **causal effects**. It answers questions like "Did this specific change *cause* the observed improvement?"
     - **Application:** Measuring the true causal impact of a new maintenance protocol on machine uptime, controlling for other variables, to prove its effectiveness.

Day – 12


Via | webex
+141
Kaiwalya Patil
Creating Responsible AI in Manufacturing
Responsible AI and its importance
The use of artificial intelligence (AI) has the potential to significantly increase the global economy by trillions of dollars.
The advantages of implementing AI in business operations have been widely recognized.
However, according to research by Accenture, for AI to fully deliver its potential benefits, it must be implemented on a larger scale throughout the entire organization.
ALL
intel digital readiness

2.13 Day 13: Project Development

Activity Overview:

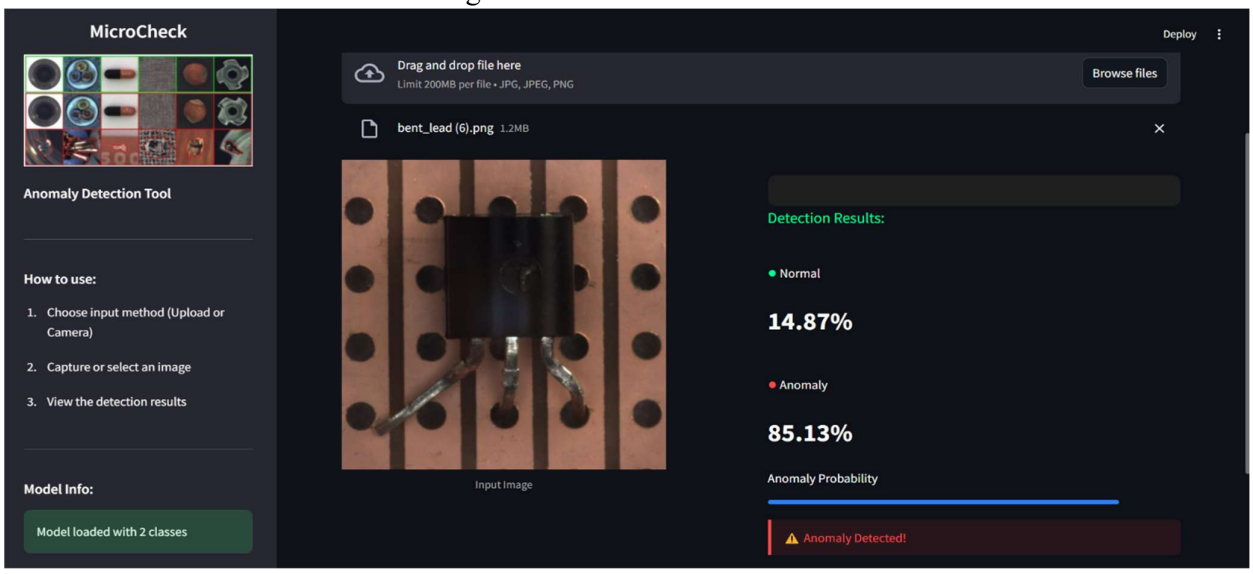
The thirteenth day focused on the end-to-end, practical development of a focused AI application for industrial component inspection. The activity centered on using Google's Teachable Machine to build a specialized anomaly detection model for a specific electronic component—a transistor—and then operationalizing this model by integrating it into a professional, cloud-deployed Streamlit web application. This project encapsulated the entire workflow from data curation and no-code model training to application development and deployment, moving beyond theory to create a functional prototype that demonstrates a direct application of AI for automated quality control in manufacturing.

Detailed Learning Outcomes:

End-to-End AI Solution Development:

- Applied the Full AI Project Lifecycle: Successfully navigated all stages of a practical AI project, from the initial problem scoping (anomaly detection for transistors) and data collection (sourcing and organizing images of "Normal" and "Anomaly" states) to model training, application integration, and final deployment to a live web environment using Streamlit Community Cloud.
- Mastered No-Code Model Prototyping: Effectively utilized Google's Teachable Machine as a rapid prototyping tool to train a production-ready TensorFlow model. Demonstrated the ability to optimize model performance for a binary classification task by strategically tuning critical hyperparameters such as the number of epochs, batch size, and learning rate to improve accuracy and prevent overfitting.
- Built and Deployed a Production Web Application: Architected and developed a modular and user-friendly Streamlit application ("MicroCheck") to serve the trained model. Gained practical experience in handling image I/O (Input/Output), implementing a real-time image processing pipeline, and structuring code into logical components (e.g., app.py, model_utils.py, image_utils.py) for maintainability and professional standards.

Assignment – 02 Micro-Check



App.py

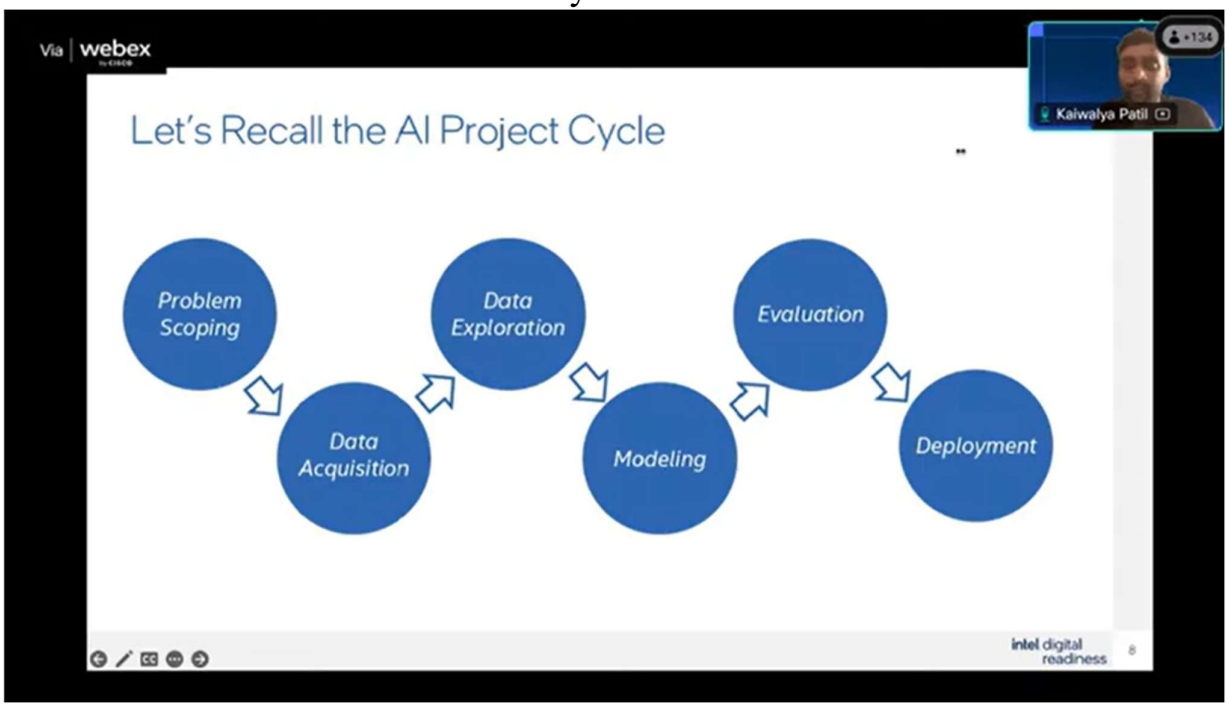
```
import streamlit as st
import tensorflow as tf
import numpy as np
import cv2
from PIL import Image
import io
import os
from model_utils import load_model, predict_image
from image_utils import preprocess_image

# Page configuration
st.set_page_config(
    page_title="MicroCheck - Anomaly Detection",
    page_icon="🔍",
    layout="wide"
)

# Custom CSS
st.markdown("""
<style>
    .main-header {
        font-size: 2.5rem;
        color: #4e8df5;
        text-align: center;
    }
    .output-container {
        background-color: #1E1E1E;
        border-radius: 10px;
        padding: 20px;
        margin-top: 20px;
    }
    .output-header {
        color: #00FF9C;
        font-size: 1.2rem;
        margin-bottom: 20px;
    }
    .result-label {
        color: #ffffff;
        font-size: 1rem;
        margin-bottom: 5px;
    }
    .probability-bar {
        height: 6px;
        background-color: #2D7FF9;
        border-radius: 3px;
        margin: 10px 0;
        transition: width 0.5s ease-in-out;
    }
    .normal-dot {
        color: #00FF9C;
        font-size: 1.2rem;
    }
    .anomaly-dot {
        color: #FF4B4B;
        font-size: 1.2rem;
    }
```

```
    .percentage {
        font-size: 2rem;
        font-weight: bold;
        margin: 5px 0;
    }
    .alert-box {
        background-color: rgba(255, 75, 75, 0.1);
        border-left: 4px solid #FF4B4B;
        padding: 10px 15px;
        margin-top: 10px;
        border-radius: 0 5px 5px 0;
    }
    .alert-text {
        color: #FF4B4B;
        display: flex;
        align-items: center;
        gap: 10px;
    }
    .footer {
        text-align: center;
        color: #888888;
        font-size: 0.8rem;
        margin-top: 3rem;
    }
</style>
""", unsafe_allow_html=True)

# Sidebar
with st.sidebar:
    st.markdown("<h1 style='text-align: center;'>MicroCheck</h1>",
unsafe_allow_html=True)
    st.image("assets/overview_dataset.jpg", width=300)
    st.markdown("### Anomaly Detection Tool")
    st.markdown("---")
    st.markdown("### How to use:")
    st.markdown("1. Choose input method (Upload or Camera)")
    st.markdown("2. Capture or select an image")
    st.markdown("3. View the detection results")
    st.markdown("---")
    st.markdown("### Model Info:")

    model_path = "model/keras_model.h5"
    labels_path = "model/labels.txt"

    if os.path.exists(model_path) and os.path.exists(labels_path):
        with open(labels_path, 'r') as f:
            class_names = [line.strip().split(' ', 1)[1] for line in
f.readlines()]
        st.success(f"Model loaded with {len(class_names)} classes")
    else:
        st.error("Model files not found. Please add keras_model.h5 and
labels.txt to the model folder.")
        class_names = ["Class not available"]

# Main content
st.markdown("<h1 class='main-header'>MicroCheck Anomaly Detection
System</h1>", unsafe_allow_html=True)
```

```
# Input method selection
input_method = st.radio("Choose input method:", ["File Upload", "Camera
Input"])

image_bytes = None
image = None

if input_method == "File Upload":
    uploaded_file = st.file_uploader("Choose an image...", type=["jpg",
"jpeg", "png"])
    if uploaded_file is not None:
        image_bytes = uploaded_file.getvalue()
        image = Image.open(io.BytesIO(image_bytes))
elif input_method == "Camera Input":
    camera_file = st.camera_input("Take a picture")
    if camera_file is not None:
        image_bytes = camera_file.getvalue()
        image = Image.open(io.BytesIO(image_bytes))

# Process image and show results
if image_bytes is not None:
    col1, col2 = st.columns([1, 1])  # Equal width columns

    with col1:
        # Display input image with fixed width
        st.image(image, caption="Input Image", width=400)

    with col2:
        if os.path.exists(model_path) and os.path.exists(labels_path):
            try:
                model = load_model(model_path)
                if model is None:
                    st.error("Failed to load model. Please check the model
file.")
                    st.stop()

                processed_image = preprocess_image(image_bytes)
                if processed_image is None:
                    st.error("Failed to process image. Please try a different
image.")
                    st.stop()

                prediction, confidence_scores = predict_image(model,
processed_image, class_names)

                # New UI Output
                st.markdown("<div class='output-container'>",
unsafe_allow_html=True)
                st.markdown("<div class='output-header'>Detection
Results:</div>", unsafe_allow_html=True)

                # Display results for both classes
                normal_score = float(confidence_scores[0])  # Ensure float
conversion
                anomaly_score = float(confidence_scores[1])  # Ensure float
conversion

                # Normal probability
```

```
                st.markdown(f"<div style='margin-bottom: 20px;'>",
unsafe_allow_html=True)
                st.markdown(f"<span class='normal-dot'>●</span> Normal",
unsafe_allow_html=True)
                st.markdown(f"<div
class='percentage'>{normal_score:.2f}%</div>", unsafe_allow_html=True)
                st.markdown("</div>", unsafe_allow_html=True)

                # Anomaly probability
                st.markdown(f"<div style='margin-bottom: 20px;'>",
unsafe_allow_html=True)
                st.markdown(f"<span class='anomaly-dot'>●</span> Anomaly",
unsafe_allow_html=True)
                st.markdown(f"<div
class='percentage'>{anomaly_score:.2f}%</div>", unsafe_allow_html=True)
                st.markdown("</div>", unsafe_allow_html=True)

                # Probability bar
                st.markdown("<div class='result-label'>Anomaly
Probability</div>", unsafe_allow_html=True)
                st.markdown(
                    f"""<div class='probability-bar' style='width: {min(100,
max(0, anomaly_score))}%;'></div>""",
                    unsafe_allow_html=True
                )

                # Alert box if anomaly detected
                if anomaly_score > 50:
                    st.markdown("""
                        <div class='alert-box'>
                            <div class='alert-text'>
                                ⚠ Anomaly Detected!
                            </div>
                        </div>
                    """, unsafe_allow_html=True)
                elif normal_score > anomaly_score:
                    st.success("Product appears normal")

                st.markdown("</div>", unsafe_allow_html=True)

            except Exception as e:
                st.error(f"An error occurred: {str(e)}")
                st.error("Please try again with a different image or check if
the model is properly loaded.")
        else:
            st.error("Model not found. Please upload the model files first.")

# Footer
st.markdown("<div class='footer'>MicroCheck | Powered by TensorFlow and
Streamlit | Created for educational purposes</div>", unsafe_allow_html=True)
```

Day – 13


Via webex
+134
Kaiwalya Patil
Let's Recall the AI Project Cycle
Problem Scoping
Data Acquisition
Data Exploration
Modeling
Evaluation
Deployment
intel digital readiness
8

2.14 Day 14: Industry Relevant AI Project – I

During our internship, we were introduced to real-world industrial challenges where undetected defects in manufacturing materials—such as metal and steel sheets before dispatch—could lead to significant financial losses, safety risks, and reduced product quality. These problems highlighted the growing need for automated, accurate, and efficient defect detection systems in modern industries.

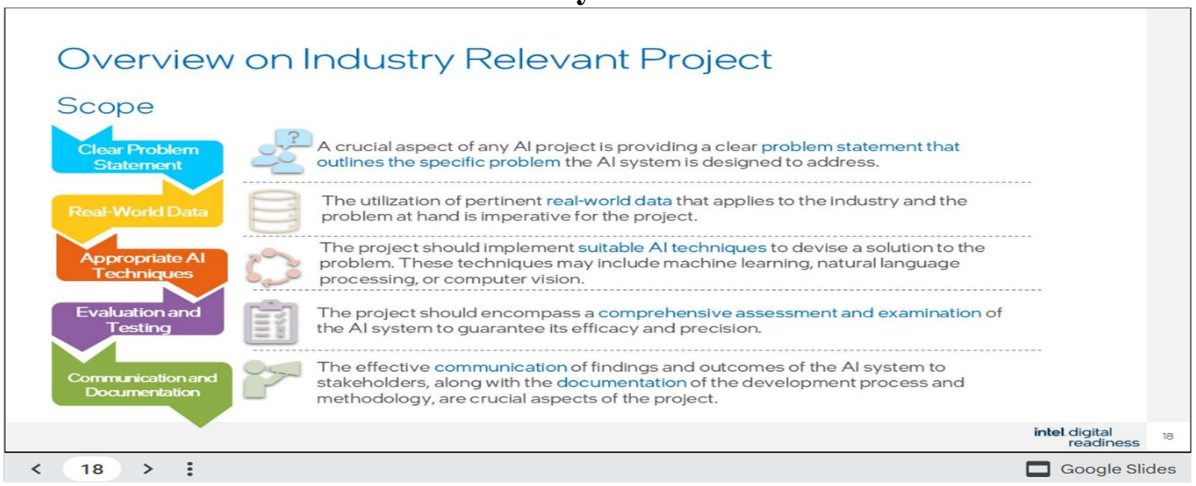
To address this, we selected **Optical PCB Defect Detection using Artificial Intelligence and Computer Vision** as our final project. The objective was to design a system that could automatically identify and classify PCB defects, reducing reliance on manual inspection and ensuring higher reliability in electronics manufacturing.

we worked on developing an **AI-driven defect detection system for Printed Circuit Boards (PCBs)** using **YOLOv8n**. We began by studying real industrial problems, such as defect detection in steel sheets before dispatch and anomaly detection in PCBs used in smartphone assembly lines, where even a small defect can cause major failures. From this, we defined our problem statement and started exploring publicly available datasets, reviewing multiple GitHub repositories and Kaggle datasets to gather images and annotations for different PCB defect categories.

Once the datasets were collected, we organized and pre-processed them, focusing initially on six common defects: **open circuit, short circuit, mouse bite, spur, spurious copper, and missing hole**. We trained a baseline YOLOv8n model and iteratively fine-tuned it, overcoming challenges such as class imbalance, small-defect misclassification, and overfitting. Later, we extended our dataset with the **SolDef_AI PCB dataset** to include four additional soldering-related defects: **excessive solder, poor solder, cold solder joint, and spike**. This broadened the capability of our model to cover a total of 10 defect types.

By the final weeks, we had built a complete pipeline for training, fine-tuning, and evaluation, along with a simple application interface where users could upload PCB images and instantly receive defect detection results. This project allowed us to strengthen our knowledge of **AI, computer vision, and deep learning**, while also demonstrating the potential of intelligent inspection systems to enhance **quality control, reduce inspection time, and minimize human error** in electronics manufacturing — aligning with the goals of **Industry 4.0**.

Day - 14



2.15 Day 15: Industry Relevant AI Project – I I

Capestone Project:

AI Driven Optical Pcb Anomaly Detection in Smartphone Assembly Line

App.py

```
#!/usr/bin/env python3
"""
PCB Defect Detection System
Main script for training, evaluation, and inference

Developed by: Raj Patel
LinkedIn: https://www.linkedin.com/in/raj-patel5
"""

import argparse
import sys
from pathlib import Path
import yaml

# Import Streamlit for web interface detection
try:
    import streamlit as st
    STREAMLIT_AVAILABLE = True
except ImportError:
    STREAMLIT_AVAILABLE = False

from src.data_preparation import PCBDataPreparator
from src.model import PCBDefectDetector

def main():
    """Main function with command-line interface. - Raj Patel"""
    parser = argparse.ArgumentParser(
        description="PCB Defect Detection System",
        formatter_class=argparse.RawDescriptionHelpFormatter,
        epilog="""
Examples:
  # Prepare data from all datasets
  python main.py prepare-data

  # Train model
  python main.py train --epochs 100

  # Predict on single image
  python main.py predict --image path/to/image.jpg

  # Live detection
  python main.py live

  # Evaluate model
  python main.py evaluate

  # Web interface
  python main.py web
```

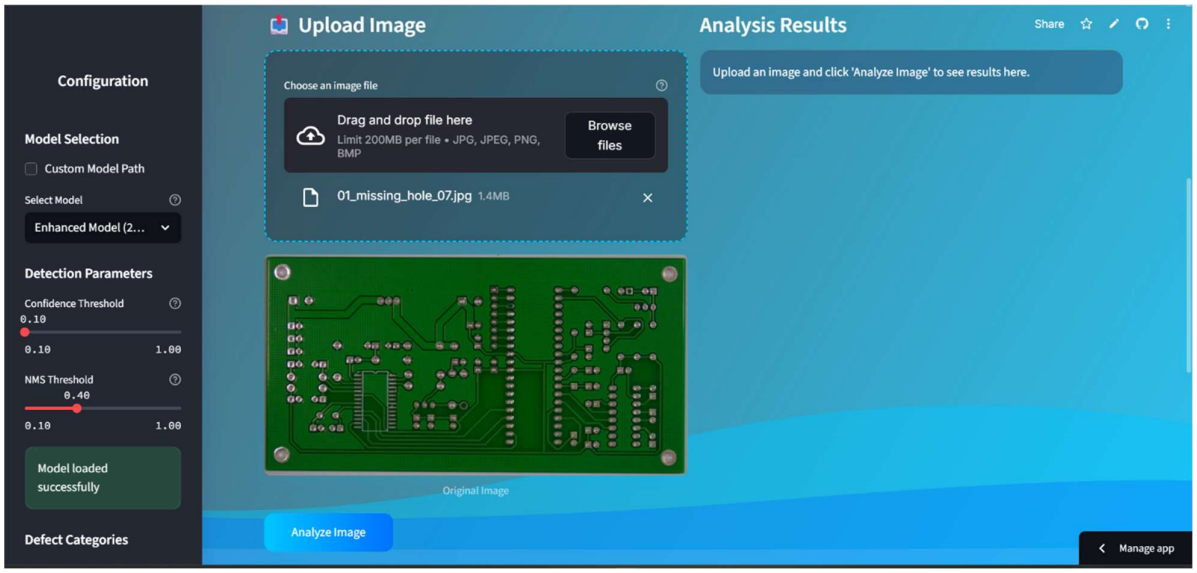
```
        """
    )

    subparsers = parser.add_subparsers(dest='command', help='Available
commands')

    # Prepare data command - Raj Patel
    prep_parser = subparsers.add_parser('prepare-data', help='Prepare and
combine all datasets')
    prep_parser.add_argument('--config', default='config.yaml',
help='Configuration file path')

    # Train command - Raj Patel
    train_parser = subparsers.add_parser('train', help='Train the model')
    train_parser.add_argument('--data-yaml',
default='output/yolo_format/data.yaml',
                              help='Path to data.yaml file')
    train_parser.add_argument('--epochs', type=int, default=None,
                              help='Number of training epochs')
    train_parser.add_argument('--model-path', default=None,
                              help='Path to pretrained model')
    train_parser.add_argument('--config', default='config.yaml',
                              help='Configuration file path')

    # Predict command - Raj Patel
    predict_parser = subparsers.add_parser('predict', help='Predict defects in
image')
    predict_parser.add_argument('--image', required=True, help='Path to input
image')
    predict_parser.add_argument('--model-path', default=None,
                                help='Path to trained model')
    predict_parser.add_argument('--config', default='config.yaml',
                                help='Configuration file path')
    predict_parser.add_argument('--no-save', action='store_true',
                                help='Do not save result image')

    # Live detection command - Raj Patel
    live_parser = subparsers.add_parser('live', help='Live defect detection')
    live_parser.add_argument('--camera', type=int, default=0,
                             help='Camera device ID')
    live_parser.add_argument('--model-path', default=None,
                             help='Path to trained model')
    live_parser.add_argument('--save-video', action='store_true',
                             help='Save detection video')
    live_parser.add_argument('--config', default='config.yaml',
                             help='Configuration file path')

    # Evaluate command - Raj Patel
    eval_parser = subparsers.add_parser('evaluate', help='Evaluate model
performance')
    eval_parser.add_argument('--data-yaml',
default='output/yolo_format/data.yaml',
                             help='Path to data.yaml file')
    eval_parser.add_argument('--model-path', default=None,
                             help='Path to trained model')
    eval_parser.add_argument('--config', default='config.yaml',
                             help='Configuration file path')
```

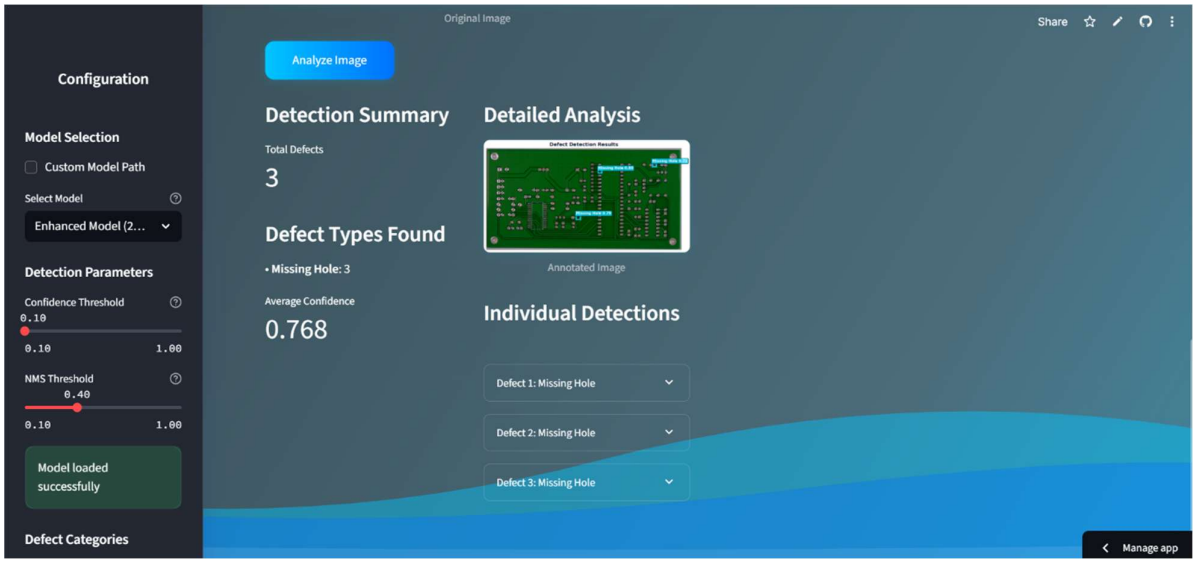
```
    # Web interface command - Raj Patel
    web_parser = subparsers.add_parser('web', help='Launch web interface')
    web_parser.add_argument('--port', type=int, default=8501,
                            help='Port for web interface')
    web_parser.add_argument('--host', default='localhost',
                            help='Host for web interface')

    # Export command - Raj Patel
    export_parser = subparsers.add_parser('export', help='Export model to
different format')
    export_parser.add_argument('--model-path', required=True,
                               help='Path to trained model')
    export_parser.add_argument('--format', default='onnx',
                               choices=['onnx', 'torchscript', 'tflite'],
                               help='Export format')
    export_parser.add_argument('--config', default='config.yaml',
                               help='Configuration file path')

    args = parser.parse_args()

    if not args.command:
        parser.print_help()
        return

    try:
        if args.command == 'prepare-data':
            prepare_data(args)
        elif args.command == 'train':
            train_model(args)
        elif args.command == 'predict':
            predict_image(args)
        elif args.command == 'live':
            live_detection(args)
        elif args.command == 'evaluate':
            evaluate_model(args)
        elif args.command == 'web':
            launch_web_interface(args)
        elif args.command == 'export':
            export_model(args)
        else:
            print(f"Unknown command: {args.command}")
            parser.print_help()

    except Exception as e:
        print(f"Error: {str(e)}")
        sys.exit(1)

def prepare_data(args):
    """Prepare and combine all datasets. - Raj Patel"""
    print("🔧 Preparing PCB defect datasets...")

    preparator = PCBDataPreparator(args.config)
    preparator.prepare_all_datasets()

    print("☑ Data preparation completed!")
    print(f"🗀 Output directory: {preparator.yolo_dir}")
    print(f"🗎 Data config: {preparator.yolo_dir / 'data.yaml'}")
```

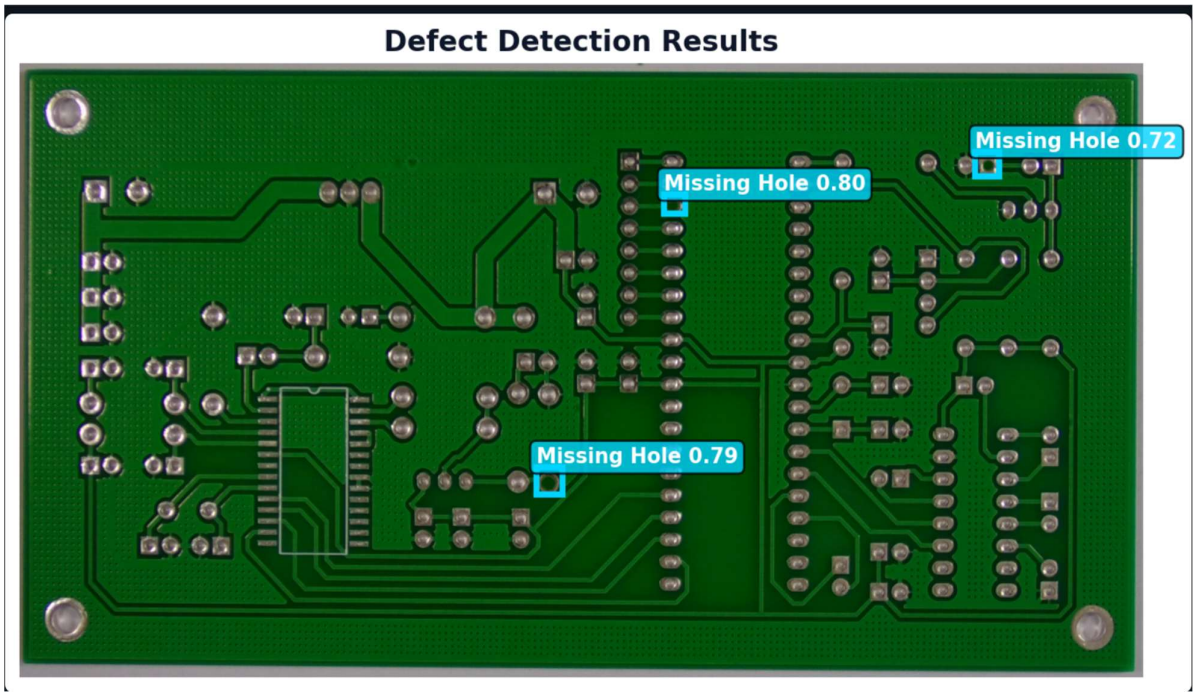
```
def train_model(args):
    """Train the PCB defect detection model. - Raj Patel"""
    print("🚀 Starting model training...")

    # Load configuration
    with open(args.config, 'r') as f:
        config = yaml.safe_load(f)

    # Initialize detector
    detector = PCBDefectDetector(args.config, args.model_path)

    # Check if data.yaml exists
    data_yaml_path = Path(args.data_yaml)
    if not data_yaml_path.exists():
        print(f"❌ Data file not found: {data_yaml_path}")
        print("Please run 'python main.py prepare-data' first.")
        return

    # Start training
    results = detector.train(args.data_yaml, args.epochs)

    print("☑ Training completed!")
    print(f"🗀 Model saved to:
{config['data']['output_path']}/pcb_defect_detection")

def predict_image(args):
    """Predict defects in a single image. - Raj Patel"""
    print(f"🔍 Predicting defects in: {args.image}")

    # Initialize detector
    detector = PCBDefectDetector(args.config, args.model_path)

    # Check if image exists
    if not Path(args.image).exists():
        print(f"❌ Image not found: {args.image}")
        return

    # Run prediction
    result = detector.predict_image(args.image, save_result=not args.no_save)

    # Display results
    print(f"📊 Detection Results:")
    print(f"   Image: {result['image_path']}")
    print(f"   Defects found: {result['num_defects']}")

    if result['num_defects'] > 0:
        print("   Defect details:")
        for i, detection in enumerate(result['detections']):
            print(f"     {i+1}. {detection['class_name']} (confidence:
{detection['confidence']:.3f})")
    else:
        print("   ☑ No defects detected!")

def live_detection(args):
    """Perform live defect detection. - Raj Patel"""
    print("🖳 Starting live detection...")
```

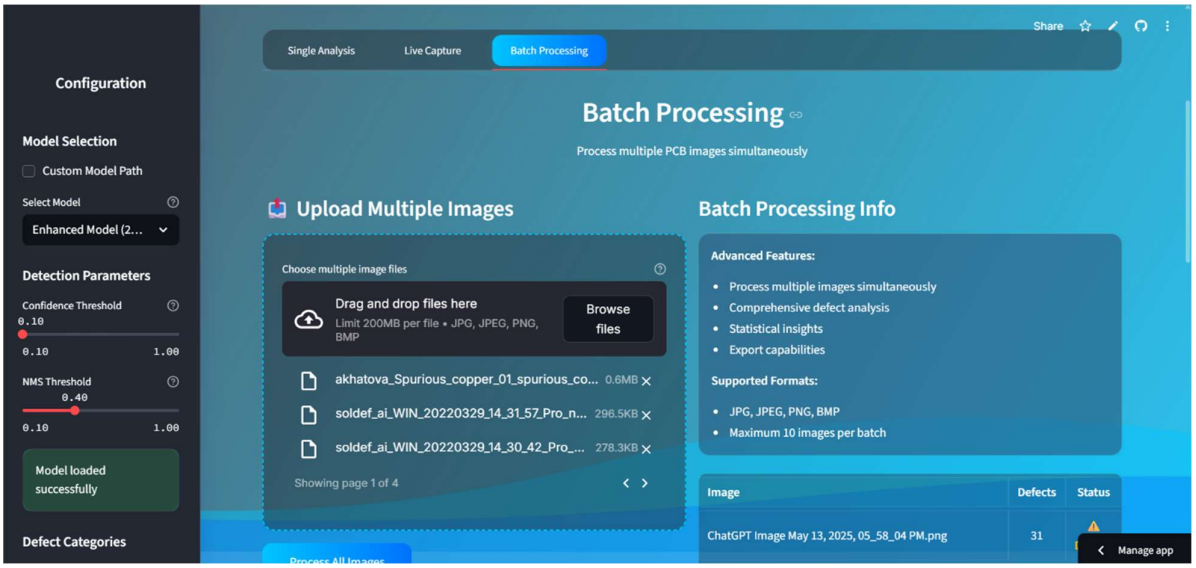
```
    # Initialize detector
    detector = PCBDefectDetector(args.config, args.model_path)

    # Start live detection
    detector.live_detection(args.camera, args.save_video)

def evaluate_model(args):
    """Evaluate model performance. - Raj Patel"""
    print("📈 Evaluating model performance...")

    # Initialize detector
    detector = PCBDefectDetector(args.config, args.model_path)

    # Check if data.yaml exists
    data_yaml_path = Path(args.data_yaml)
    if not data_yaml_path.exists():
        print(f"❌ Data file not found: {data_yaml_path}")
        print("Please run 'python main.py prepare-data' first.")
        return

    # Run evaluation
    metrics = detector.evaluate(args.data_yaml)

    print("☑️ Evaluation completed!")

def launch_web_interface(args):
    """Launch the web interface. - Raj Patel"""
    print(f"🌐 Launching web interface at http://{args.host}:{args.port}")

    try:
        import streamlit.web.cli as stcli
        import sys

        # Set up streamlit arguments
        sys.argv = [
            "streamlit", "run", "src/web_interface.py",
            "--server.port", str(args.port),
            "--server.address", args.host
        ]

        # Launch streamlit
        stcli.main()

    except ImportError:
        print("❌ Streamlit not installed. Please install with: pip install
streamlit")
    except Exception as e:
        print(f"❌ Error launching web interface: {str(e)}")

def export_model(args):
    """Export model to different format. - Raj Patel"""
    print(f"📦 Exporting model to {args.format} format...")

    # Initialize detector
    detector = PCBDefectDetector(args.config, args.model_path)

    # Export model
```

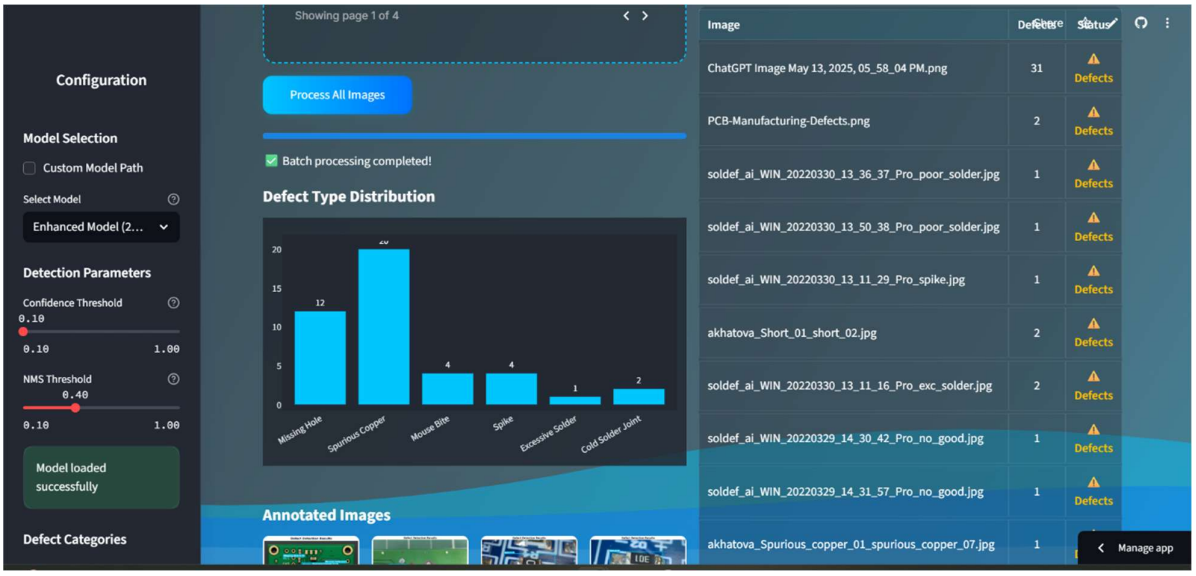
```
    output_path = detector.export_model(args.format)

    print(f"☑ Model exported to: {output_path}")

if __name__ == "__main__":
    # Check if running with streamlit
    import sys
    if STREAMLIT_AVAILABLE and ("streamlit" in sys.argv[0] or "streamlit" in "
".join(sys.argv).lower() or len(sys.argv) == 1):
        # Run Streamlit app directly
        print("🌐 Launching PCB Defect Detection Web Interface...")
        print("Author: Raj Patel")
        print("LinkedIn: https://www.linkedin.com/in/raj-patel5")

        # Import and run the web interface
        from src.web_interface import PCBDefectWebInterface

        app = PCBDefectWebInterface()
        app.main()
    else:
        # Run CLI interface
        main()
```




Original Image
Share
Configuration
Analyze Image
Model Selection
Custom Model Path
Select Model
Enhanced Model (2...
Detection Parameters
Confidence Threshold
0.10
0.10
1.00
NMS Threshold
0.40
0.10
1.00
Model loaded successfully
Defect Categories
Detection Summary
Total Defects
3
Defect Types Found
• Missing Hole: 3
Average Confidence
0.768
Detailed Analysis
Annotated Image
Individual Detections
Defect 1: Missing Hole
Defect 2: Missing Hole
Defect 3: Missing Hole
Manage app


Defect Detection Results
Missing Hole 0.72
Missing Hole 0.80
Missing Hole 0.79


Configuration
Model Selection
Custom Model Path
Select Model
Enhanced Model (2...
Detection Parameters
Confidence Threshold
0.10
0.10 1.00
NMS Threshold
0.40
0.10 1.00
Model loaded successfully
Defect Categories
Single Analysis
Live Capture
Batch Processing
Batch Processing
Process multiple PCB images simultaneously
Upload Multiple Images
Choose multiple image files
Drag and drop files here
Limit 200MB per file • JPG, JPEG, PNG, BMP
Browse files
akhatova_Spurious_copper_01_spurious_co... 0.6MB
soldef_ai_WIN_20220329_14_31_57_Pro_n... 296.5KB
soldef_ai_WIN_20220329_14_30_42_Pro_... 278.3KB
Showing page 1 of 4
Batch Processing Info
Advanced Features:
Process multiple images simultaneously
Comprehensive defect analysis
Statistical insights
Export capabilities
Supported Formats:
JPG, JPEG, PNG, BMP
Maximum 10 images per batch
Image
Defects
Status
ChatGPT Image May 13, 2025, 05_58_04 PM.png
31
Manage app


Configuration
Model Selection
Custom Model Path
Select Model
Enhanced Model (2...
Detection Parameters
Confidence Threshold
0.10
0.10 1.00
NMS Threshold
0.40
0.10 1.00
Model loaded successfully
Defect Categories
Showing page 1 of 4
Process All Images
Batch processing completed!
Defect Type Distribution
Missing Hole
Spurious Copper
Mouse Bite
Spike
Excessive Solder
Cold Solder Joint
Annotated Images
Image
Defects
Status
ChatGPT Image May 13, 2025, 05_58_04 PM.png
31
Defects
PCB-Manufacturing-Defects.png
2
Defects
soldef_ai_WIN_20220330_13_36_37_Pro_poor_solder.jpg
1
Defects
soldef_ai_WIN_20220330_13_50_38_Pro_poor_solder.jpg
1
Defects
soldef_ai_WIN_20220330_13_11_29_Pro_spike.jpg
1
Defects
akhatova_Short_01_short_02.jpg
2
Defects
soldef_ai_WIN_20220330_13_11_16_Pro_exc_solder.jpg
2
Defects
soldef_ai_WIN_20220329_14_30_42_Pro_no_good.jpg
1
Defects
soldef_ai_WIN_20220329_14_31_57_Pro_no_good.jpg
1
Defects
akhatova_Spurious_copper_01_spurious_copper_07.jpg
1
Manage app


Configuration
Model Selection
Custom Model Path
Select Model
Enhanced Model (2...
Detection Parameters
Confidence Threshold
0.10
0.10 1.00
NMS Threshold
0.40
0.10 1.00
Model loaded successfully
Defect Categories
Annotated Images
ChatGPT Image May 13, 2025, 05_58_04 PM.png
PCB-Manufacturing-Defects.png
soldef_ai_WIN_20220330_13_36_37_Pro_poor_solder.jpg
soldef_ai_WIN_20220330_13_50_38_Pro_poor_solder.jpg
soldef_ai_WIN_20220330_13_11_29_Pro_spike.jpg
akhatova_Short_01_short_02.jpg
soldef_ai_WIN_20220330_13_11_16_Pro_exc_solder.jpg
soldef_ai_WIN_20220329_14_30_42_Pro_no_good.jpg
soldef_ai_WIN_20220329_14_31_57_Pro_no_good.jpg
akhatova_Spurious_copper_01_spurious_copper_07.jpg
Defects
akhatova_Spurious_copper_01_spurious_copper_07.jpg
1
Defects
Download All Defected Images (ZIP)
Manage app

CONCLUSION

The 15-day summer internship, conducted through the Intel® AI for Manufacturing certificate course, has been a transformative journey from foundational concepts to the deployment of a sophisticated industry-ready project. This intensive program provided a comprehensive, end-to-end understanding of how artificial intelligence is revolutionizing modern manufacturing, moving beyond theoretical knowledge to practical, hands-on implementation.

The internship began by establishing a strong foundation, exploring the strategic value of AI for SMEs and mastering the essential tools of the trade, including Python, NumPy, Pandas, and Matplotlib. The critical importance of the AI project lifecycle was emphasized through the PREDICT/EVALUATE framework, instilling a discipline of meticulous planning and business alignment before any code is written. The subsequent deep dives into data handling, modeling, and evaluation revealed that the quality of an AI solution is fundamentally determined by the rigor applied in these stages—data preprocessing, feature engineering, and objective metric analysis form the bedrock of any successful deployment.

The journey progressed to specialized applications, each addressing a key manufacturing challenge. Computer vision demystified automated quality inspection, while predictive maintenance showcased the power of IoT and AI in preventing costly downtime. Robotic Process Automation highlighted opportunities for efficiency beyond the production floor, and edge inferencing unveiled the future of real-time, decentralized intelligence. The exploration of demand forecasting underscored AI's role in building resilient supply chains, and the critical module on AI ethics provided the essential framework for building trustworthy and responsible systems.

The culmination of this learning was the capstone project: **AI Driven Optical PCB Inspection for Smartphone Assembly Lines**. This project served as a synthesis of every acquired skill. It involved the application of computer vision, model training with TensorFlow, rigorous evaluation, containerization with Docker, cloud deployment on AWS, and the development of a user-friendly interface with Flask. This end-to-end process transformed an abstract concept into a functional, deployable tool that solves a real-world problem, demonstrating a direct impact on quality control efficiency and cost reduction.