i want a one practicle where im going to train a supervised model for fire detection model and you can create own small dataset by our own and train the model i need these four proper steps Data → Model → Evaluation → Deployment write a proper python code which i can run my jupyter lab

dataset will be like 4 sensor data and label has to be 1 or 0 and it has to be a classification model with the help of scikit learn

Below is a fully **practical**, **end-to-end**, **Jupyter-ready** supervised ML pipeline for **fire detection using 4-sensor synthetic data**, following the exact pipeline:

✅ Data → Model → Evaluation → Deployment

The dataset is **synthetically generated** to simulate real sensor readings.
You can directly copy–paste this into your **Jupyter Notebook** and run it.

🔥 Fire Detection Model (Full Practical Example)

Using: Scikit-learn, NumPy, Pandas, Joblib

✅ STEP 1 — DATA (Create + Preprocess)

python Copy code

```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# ---------------------------
# 1) Create Synthetic Dataset
# ---------------------------

np.random.seed(42)
```

```
# 4 sensor readings
# Example interpretation:
#   sensor1: temperature
#   sensor2: smoke density
#   sensor3: CO level
#   sensor4: humidity

num_samples = 600

# Normal environment readings
normal_data = np.random.normal(loc=[30, 5, 2, 40], scale=[3, 1, 0.5, 5]
normal_labels = np.zeros(300)

# Fire environment readings (higher temp, smoke, CO, lower humidity)
fire_data = np.random.normal(loc=[70, 20, 15, 20], scale=[5, 3, 2, 4],
fire_labels = np.ones(300)

# Combine
X = np.vstack([normal_data, fire_data])
y = np.hstack([normal_labels, fire_labels])

df = pd.DataFrame(X, columns=["sensor1_temp", "sensor2_smoke", "sensor3
df["label"] = y

print("Sample Data:")
df.head()
```

✅ STEP 2 — MODEL (Training a Classifier)

We use **RandomForestClassifier** (works very well for tabular data).

python Copy code

```
from sklearn.ensemble import RandomForestClassifier

# ------------------------------------
# Train-test split
# ------------------------------------
X = df.drop("label", axis=1)
y = df["label"]

X_train, X_test, y_train, y_test = train_test_split(
```

```
    X, y, test_size=0.25, random_state=42, stratify=y
)

# -----------------------------------
# Scaling (optional but good practice)
# -----------------------------------
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# -----------------------------------
# Train Model
# -----------------------------------
model = RandomForestClassifier(n_estimators=200, random_state=42)
model.fit(X_train_scaled, y_train)

print("Model training completed.")
```

✅ STEP 3 — EVALUATION (Accuracy, Confusion Matrix, Report)

python Copy code

```
from sklearn.metrics import accuracy_score, confusion_matrix, classifi
import seaborn as sns
import matplotlib.pyplot as plt

# -------------------------------
# Predictions
# -------------------------------
y_pred = model.predict(X_test_scaled)

# -------------------------------
# Metrics
# -------------------------------
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pre

# -------------------------------
# Confusion Matrix
# -------------------------------
cm = confusion_matrix(y_test, y_pred)
```

```
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=["No Fire", "Fire"],
            yticklabels=["No Fire", "Fire"])
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

✅ STEP 4 — DEPLOYMENT (Save Model + Example Prediction API Code)

Save the model + scaler using joblib

python Copy code

```python
import joblib

# Save model & scaler
joblib.dump(model, "fire_detection_model.pkl")
joblib.dump(scaler, "fire_detection_scaler.pkl")

print("Model and scaler saved!")
```

Example: Loading Model (Deployment Simulation)

python Copy code

```python
# Deployment: loading the model
loaded_model = joblib.load("fire_detection_model.pkl")
loaded_scaler = joblib.load("fire_detection_scaler.pkl")

# Example new sensor reading (Simulate real device)
new_data = np.array([[65, 18, 12, 25]])  # temp, smoke, CO, humidity

# Scale the input
new_data_scaled = loaded_scaler.transform(new_data)
```

```
# Prediction
pred = loaded_model.predict(new_data_scaled)
print("Fire Detected!" if pred[0] == 1 else "No Fire Detected.")
```

🎉 This completes the full ML pipeline

✔ Data creation

✔ Model training

✔ Evaluation

✔ Deployment-ready saved model

If you want, I can also generate:

✅ A **Flask API**
✅ A **FastAPI deployment**
✅ A **real-time sensor simulation**
✅ A **plot of sensor distributions**
Just tell me!

i need to do the model evaluation on the unseen dataset

Below is the **extended version** where you perform **evaluation on an unseen dataset** (a dataset not used in training or testing).

I will:

1. **Generate a separate unseen dataset**
2. **Load the saved model & scaler**
3. **Evaluate on unseen data**
4. **Show accuracy, confusion matrix, classification report**

You can copy–paste directly into Jupyter.

🔥 STEP 5 — EVALUATION ON UNSEEN DATA

✅ A) Create a NEW Unseen Dataset (never used earlier)

This dataset has similar statistical distribution but is newly generated.

python Copy code

```python
import numpy as np
import pandas as pd

np.random.seed(999)   # different seed → new dataset

# Generate NEW unseen dataset (200 samples)
unseen_normal = np.random.normal(
    loc=[30, 5, 2, 40],
    scale=[3, 1, 0.5, 5],
    size=(100, 4)
)

unseen_fire = np.random.normal(
    loc=[70, 20, 15, 20],
    scale=[5, 3, 2, 4],
    size=(100, 4)
)

X_unseen = np.vstack([unseen_normal, unseen_fire])
y_unseen = np.hstack([np.zeros(100), np.ones(100)])

unseen_df = pd.DataFrame(
    X_unseen,
    columns=["sensor1_temp", "sensor2_smoke", "sensor3_CO", "sensor4_hu
)
unseen_df["label"] = y_unseen

print("Unseen dataset sample:")
unseen_df.head()
```

✅ B) Load the saved model & scaler

python Copy code

```python
import joblib

model = joblib.load("fire_detection_model.pkl")
scaler = joblib.load("fire_detection_scaler.pkl")
```

✅ C) Evaluate on Unseen Dataset

python Copy code

```python
from sklearn.metrics import accuracy_score, confusion_matrix, classifi
import seaborn as sns
import matplotlib.pyplot as plt

# Split features and labels
X_u = unseen_df.drop("label", axis=1)
y_u = unseen_df["label"]

# Scale unseen data using TRAINING SCALER
X_u_scaled = scaler.transform(X_u)

# Predict
y_u_pred = model.predict(X_u_scaled)

# Evaluation Metrics
print("Unseen Data Accuracy:", accuracy_score(y_u, y_u_pred))
print("\nUnseen Data Classification Report:\n", classification_report(y

# Confusion Matrix
cm = confusion_matrix(y_u, y_u_pred)

plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Greens",
            xticklabels=["No Fire", "Fire"],
            yticklabels=["No Fire", "Fire"])
plt.title("Unseen Dataset Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

✅ RESULT: TRUE GENERALIZATION CHECK

This unseen dataset evaluation answers:

- Does the model work on new data?
- Is it overfitting or generalizing?

You will see **accuracy, precision, recall, F1 score**, and a new **confusion matrix**.

If you want, I can also add:

- **k-Fold Cross Validation**
- **ROC–AUC curve**
- **Precision–Recall curve**
- **GridSearchCV for hyperparameter tuning**
- **MLflow logging**

Just tell me!