Advanced AIML and Data Science Project

Here's a code example for an **Advanced AIML and Data Science Project** using **Predictive Maintenance for Industrial Equipment**. In this project, we will use machine learning to predict equipment failures based on sensor data (such as temperature, pressure, and vibration). We'll use the **Random Forest** model for classification to predict if a machine will fail or not.

Let's break it down step by step.

Project: Predictive Maintenance for Industrial Equipment

Step 1: Import Libraries

python

Copy

```
# Import necessary libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification_report, confusion_matrix

from sklearn.preprocessing import StandardScaler
```

Step 2: Load Dataset

In this example, we'll assume that we have a dataset in CSV format (maintenance_data.csv), which contains sensor readings (temperature, pressure, vibration) and a target variable that indicates whether the equipment failed (failure column).

python

Copy

```
# Load the dataset (replace with your dataset path)
df = pd.read_csv('maintenance_data.csv')


# Display the first few rows of the dataset
print(df.head())
```

Step 3: Data Preprocessing

Let's check for missing values, handle categorical features (if any), and scale the numerical features for better model performance.

python

Copy

```
# Check for missing values
print(df.isnull().sum())


# Drop rows with missing values or handle them appropriately
df.dropna(inplace=True)


# Feature selection - assuming sensor data and failure status are the features
X = df.drop(columns=['failure'])  # Features
y = df['failure']  # Target variable


# Normalize the features (important for distance-based models)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)


# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

Step 4: Model Building

Now, let's train a Random Forest Classifier to predict equipment failure.

python

Copy

```
# Initialize the Random Forest Classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
rf_classifier.fit(X_train, y_train)

# Make predictions on the test data
y_pred = rf_classifier.predict(X_test)
```

Step 5: Model Evaluation

We will evaluate the model's performance using metrics like accuracy, precision, recall, F1-score, and confusion matrix.

python

Copy

```
# Classification report
print(classification_report(y_test, y_pred))

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot confusion matrix
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['No Failure', 'Failure'], yticklabels=['No Failure', 'Failure'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

Step 6: Model Tuning

If necessary, you can tune the Random Forest model to improve performance. You can adjust hyperparameters like n_estimators, max_depth, min_samples_split, etc. Using **GridSearchCV** or **RandomizedSearchCV** can help you find the best combination of parameters.

python

Copy

```
from sklearn.model_selection import GridSearchCV


# Define the parameter grid

param_grid = {

    'n_estimators': [50, 100, 200],

    'max_depth': [None, 10, 20, 30],

    'min_samples_split': [2, 5, 10]

}


# Initialize GridSearchCV

grid_search = GridSearchCV(estimator=RandomForestClassifier(random_state=42), param_grid=param_grid, cv=3, n_jobs=-1, verbose=2)


# Fit GridSearchCV

grid_search.fit(X_train, y_train)


# Print the best parameters and score

print("Best Parameters:", grid_search.best_params_)

print("Best Score:", grid_search.best_score_)
```

Step 7: Predictive Maintenance Deployment

Once you're happy with the model's performance, you can deploy it to monitor new equipment data and predict potential failures. This can be done using tools like Flask or FastAPI for web deployment or integrating the model into industrial IoT systems.

Conclusion:

This project helps you understand how to:

1. Preprocess sensor data for predictive maintenance.
2. Train and evaluate a classification model (Random Forest in this case).
3. Tune and optimize the model to improve its prediction accuracy.
4. Use metrics like confusion matrix and classification report to evaluate performance.

You can expand this project by:

- Using more advanced models (like XGBoost, Neural Networks).
- Adding time-series data (like historical equipment usage).
- Incorporating sensor data streams for real-time predictions.