Module 1: Data Ingestion and Preprocessing

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

def load_and_clean_data(file_path):
    """
    Loads engine sensor data and performs initial cleaning.
    """
    print(f"Loading data from {file_path}...")
    df = pd.read_csv(file_path)

    # Handling missing values if any
    if df.isnull().values.any():
        df = df.fillna(df.mean())

    # Feature scaling setup
    features = ['Engine_RPM', 'Coolant_Temp (°C)', 'Fuel_Pressure (kPa)',
                'Intake_Pressure (kPa)', 'Oil_Temp (°C)', 'Load_Percent']
    X = df[features]
    y = df['Label']

    return X, y

def prepare_datasets(X, y):
    # Splitting into train and test sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    return X_train_scaled, X_test_scaled, y_train, y_test, scaler
```

Module 2: Diagnostic Visualization

```
import matplotlib.pyplot as plt
import seaborn as sns

def plot_engine_diagnostics(df):
    """
    Generates multi-plot visualization for engine health.
    """
    plt.figure(figsize=(15, 10))

    # RPM vs Load
    plt.subplot(2, 2, 1)
    sns.scatterplot(data=df, x='Engine_RPM', y='Load_Percent', hue='Label')
    plt.title('Engine RPM vs Load')

    # Temp Correlation
    plt.subplot(2, 2, 2)
    sns.regplot(data=df, x='Coolant_Temp (°C)', y='Oil_Temp (°C)')
    plt.title('Coolant vs Oil Temperature Correlation')

    # Pressure Distribution
    plt.subplot(2, 2, 3)
    sns.boxplot(x='Label', y='Fuel_Pressure (kPa)', data=df)
    plt.title('Fuel Pressure by Label')

    # Save results
    plt.tight_layout()
    plt.savefig('engine_diagnostics.png')
    print("Diagnostics plot saved as engine_diagnostics.png")
```

Module 3: Machine Learning Model (Random Forest)

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix

def train_diagnostic_model(X_train, y_train):
    """
    Trains a Random Forest classifier to detect engine anomalies.
    """
    model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=42)
    model.fit(X_train, y_train)
    return model

def evaluate_model(model, X_test, y_test):
    predictions = model.predict(X_test)
    print("Classification Report:")
    print(classification_report(y_test, predictions))

    cm = confusion_matrix(y_test, predictions)
    sns.heatmap(cm, annot=True, fmt='d')
    plt.ylabel('Actual')
    plt.xlabel('Predicted')
    plt.show()
```

Module 4: Main Execution Pipeline

```
def main_pipeline(csv_file):
    # 1. Ingestion
    X, y = load_and_clean_data(csv_file)

    # 2. Preparation
    X_train, X_test, y_train, y_test, scaler = prepare_datasets(X, y)

    # 3. Visualization
    raw_df = pd.read_csv(csv_file)
    plot_engine_diagnostics(raw_df)

    # 4. Modeling
    model = train_diagnostic_model(X_train, y_train)

    # 5. Evaluation
    evaluate_model(model, X_test, y_test)

    print("Pipeline Execution Complete.")

if __name__ == "__main__":
    DATA_PATH = 'Riya Sharma .csv'
    main_pipeline(DATA_PATH)
```