Pharmaceutical Industry - Quality Control in Manufacturing

Source Code (Python)

1. Install Dependencies

```
pip install pandas scikit-learn matplotlib seaborn
```

2. Code Implementation

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Simulated Data (for demonstration)
data = {
    'Batch_ID': [f'B{i:03d}' for i in range(1, 101)],
    'Temperature': np.random.uniform(20, 30, 100),
    'Humidity': np.random.uniform(40, 70, 100),
    'pH': np.random.uniform(6.5, 7.5, 100),
    'API_Concentration': np.random.uniform(95, 100, 100),
    'Moisture_Content': np.random.uniform(1.5, 3.5, 100),
    'Compression_Force': np.random.uniform(10, 20, 100),
}

# Label: Quality Pass (1) or Fail (0) based on thresholds
data['Quality_Pass'] = ((data['Temperature'] >= 22) &
                        (data['Temperature'] <= 28) &
                        (data['Humidity'] <= 60) &
                        (data['pH'] >= 6.8) &
                        (data['API_Concentration'] >= 97) &
                        (data['Moisture_Content'] <= 2.5) &
                        (data['Compression_Force'] >= 12)).astype(int)

df = pd.DataFrame(data)

# Features & Target
X = df.drop(['Batch_ID', 'Quality_Pass'], axis=1)
y = df['Quality_Pass']

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model Training
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)
```

```
# Evaluation
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Feature Importance
importances = model.feature_importances_
feature_importance_df = pd.DataFrame({'Feature': X.columns, 'Importance': importances})
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)

# Plot Feature Importance
plt.figure(figsize=(8, 5))
sns.barplot(data=feature_importance_df, x='Importance', y='Feature', palette='viridis')
plt.title('Feature Importance in Quality Control')
plt.show()
```