06_schema_drift_ai4i

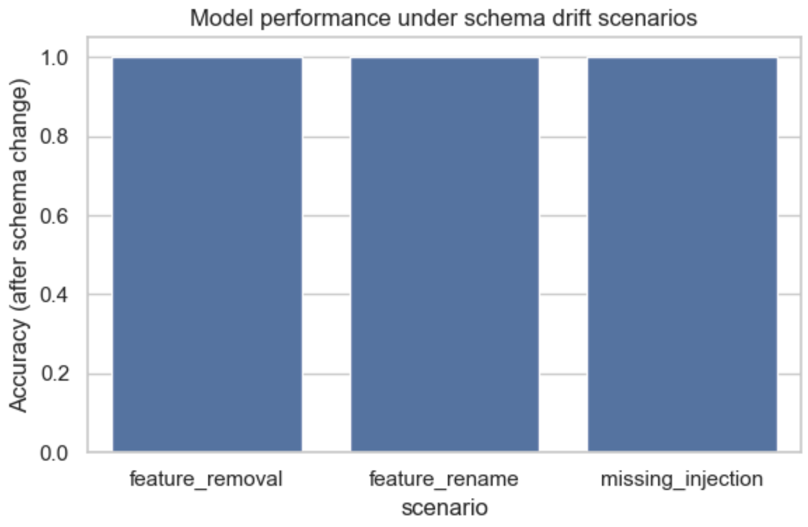
Objective: simulate schema drift for the AI4I dataset by dropping a feature, renaming a column, and introducing a missing feature. Evaluate whether the pipeline breaks, performance degrades, and whether metadata-aware checks detect the changes.

In [8]:
```
import pandas as pd
import numpy as np
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

# 06_schema_drift_ai4i

import os
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
sns.set(style='whitegrid')

# Load and preprocess using same steps as notebook 05
path = os.path.join('..','data','ai4i','ai4i2020.csv')
raw = pd.read_csv(path)
raw.columns = raw.columns.str.strip().str.lower().str.replace(r'[^0-9a-z]+','_', regex=True).str.strip('_')
# check target
if 'machine_failure' not in raw.columns:
    raise RuntimeError('machine_failure not found')
raw['machine_failure'] = raw['machine_failure'].astype(int)
# keep numeric features and udi for ordering
order_col = 'udi' if 'udi' in raw.columns else raw.columns[0]
data = raw.select_dtypes(include=[np.number]).copy()
data = data.dropna(subset=['machine_failure']).reset_index(drop=True)
# time-based split consistent with notebook 05
data = data.sort_values(order_col).reset_index(drop=True)
n = len(data); train_end = int(0.6 * n)
train = data.iloc[:train_end].copy(); test = data.iloc[train_end:].copy()
X_train = train.drop(columns=['machine_failure', order_col]); y_train = train['machine_failure']
X_test = test.drop(columns=['machine_failure', order_col]); y_test = test['machine_failure']
print('Prepared shapes:', X_train.shape, X_test.shape)
# Train baseline model on original schema
RANDOM_STATE = 42
clf = RandomForestClassifier(n_estimators=200, max_depth=10, random_state=RANDOM_STATE, n_jobs=-1)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
baseline_acc = accuracy_score(y_test, y_pred)
baseline_f1 = f1_score(y_test, y_pred, zero_division=0)
print('Baseline — Acc:{:.4f} F1:{:.4f}'.format(baseline_acc, baseline_f1))

## Schema Drift Scenarios

results = []
# Scenario A: Feature Removal (torque_nm)
feat_remove = 'torque_nm'
X_test_rm = X_test.copy()
if feat_remove in X_test_rm.columns:
    X_test_rm = X_test_rm.drop(columns=[feat_remove])
# metadata-aware check
expected = set(X_train.columns)
current = set(X_test_rm.columns)
missing = list(expected - current)
scenario = 'feature_removal'
info = {'scenario':scenario, 'missing_cols': missing}
# Try inference and capture outcome
try:
    y_pred_rm = clf.predict(X_test_rm)
    acc_rm = accuracy_score(y_test, y_pred_rm)
    info.update({'crashed':False, 'accuracy':acc_rm})
except Exception as e:
    # pipeline crashed due to schema mismatch; try a fallback by adding missing cols as zeros
```

```python
    X_test_fix = X_test_rm.copy()
    for c in missing:
        X_test_fix[c] = 0
    # ensure column order
    X_test_fix = X_test_fix[X_train.columns]
    y_pred_fix = clf.predict(X_test_fix)
    acc_fix = accuracy_score(y_test, y_pred_fix)
    info.update({'crashed':True, 'error':str(e), 'accuracy_fallback':acc_fix})
results.append(info)

# Scenario B: Feature Renaming (process_temperature_k -> proc_temp_k)
old = 'process_temperature_k'
new = 'proc_temp_k'
X_test_ren = X_test.copy()
if old in X_test_ren.columns:
    X_test_ren = X_test_ren.rename(columns={old:new})
scenario = 'feature_rename'
info = {'scenario':scenario}
try:
    y_pred_ren = clf.predict(X_test_ren)
    acc_ren = accuracy_score(y_test, y_pred_ren)
    info.update({'crashed':False, 'accuracy':acc_ren})
except Exception as e:
    # attempt naive fix: map new name back to expected by copying column if possible
    X_test_fix = X_test_ren.copy()
    if new in X_test_fix.columns:
        X_test_fix[old] = X_test_fix[new]
    # add any missing columns as zeros
    for c in set(X_train.columns) - set(X_test_fix.columns):
        X_test_fix[c] = 0
    X_test_fix = X_test_fix[X_train.columns]
    y_pred_fix = clf.predict(X_test_fix)
    acc_fix = accuracy_score(y_test, y_pred_fix)
    info.update({'crashed':True, 'error':str(e), 'accuracy_fallback':acc_fix})
results.append(info)

# Scenario C: Missing Feature Injection (30% NaNs in a key feature)
key = 'process_temperature_k' if 'process_temperature_k' in X_test.columns else X_test.columns[0]
X_test_nan = X_test.copy()
rng = np.random.RandomState(42)
idx = rng.choice(X_test_nan.index, size=int(0.3 * len(X_test_nan)), replace=False)
X_test_nan.loc[idx, key] = np.nan
scenario = 'missing_injection'
info = {'scenario':scenario, 'nan_fraction':0.3, 'key':key}
# RandomForest cannot handle NaN; attempt imputation then predict
try:
    # simple impute with column mean from train
    col_mean = X_train[key].mean()
    X_test_imp = X_test_nan.copy()
    X_test_imp[key] = X_test_imp[key].fillna(col_mean)
    y_pred_imp = clf.predict(X_test_imp)
    acc_imp = accuracy_score(y_test, y_pred_imp)
    info.update({'crashed':False, 'accuracy':acc_imp})
except Exception as e:
    info.update({'crashed':True, 'error':str(e)})
results.append(info)

# Summarize results into a comparison table and plot performance degradation
res_df = pd.DataFrame(results)
print(res_df)
os.makedirs(os.path.join('..','results','figures'), exist_ok=True)
os.makedirs(os.path.join('..','results','tables'), exist_ok=True)
# Prepare a simple accuracy column (use fallback if provided)
def pick_acc(row):
    if 'accuracy' in row and not pd.isna(row['accuracy']):
        return row['accuracy']
    if 'accuracy_fallback' in row and not pd.isna(row['accuracy_fallback']):
        return row['accuracy_fallback']
    return np.nan
res_df['accuracy_effective'] = res_df.apply(pick_acc, axis=1)
res_df.to_csv(os.path.join('..','results','tables','06_schema_drift_summary.csv'), index=False)
# Plot performance degradation
plt.figure(figsize=(6,4))
sns.barplot(data=res_df, x='scenario', y='accuracy_effective')
plt.ylabel('Accuracy (after schema change)')
plt.title('Model performance under schema drift scenarios')
plt.tight_layout()
plt.savefig(os.path.join('..','results','figures','06_schema_drift_performance.png'), dpi=300)
plt.show()
```

```
# ## Research notes
# - In data lakes, schema evolution (columns renamed/removed) is common; pipelines should verify expected schema vi
# - Our simple fallback (adding missing columns with zeros or mapping renamed columns) shows graceful degradation b
```

```
Prepared shapes: (6000, 10) (4000, 10)
Baseline — Acc:0.9992 F1:0.9818
            scenario missing_cols  crashed  \
0    feature_removal  [torque_nm]     True
1     feature_rename          NaN     True
2  missing_injection          NaN    False

                                              error  accuracy_fallback  \
0  The feature names should match those that were...            0.99925
1  The feature names should match those that were...            0.99925
2                                                NaN                NaN

   nan_fraction                    key  accuracy
0           NaN                    NaN       NaN
1           NaN                    NaN       NaN
2           0.3  process_temperature_k   0.99925
```


Model performance under schema drift scenarios
Accuracy (after schema change)
feature_removal
feature_rename
missing_injection
scenario