```python
from pathlib import Path
from dataclasses import dataclass
from typing import Optional

import pandas as pd
from sklearn.model_selection import cross_validate
from sklearn.pipeline import Pipeline


@dataclass
class ExperimentConfig:
    name: str
    pipeline: Pipeline
    cv_folds: int = 5
    scoring: str = "neg_mean_absolute_percentage_error"
    extra: Optional[dict] = None


@dataclass
class ExperimentResult:
    name: str
    pipeline: Pipeline
    cv_score: float
    cv_std: float


class Experiment:
    def __init__(
        self,
        config: ExperimentConfig,
        output_dir: Path | str = "./artifacts/experiment-results",
    ):
        self.config = config
        self.output_dir = (
            Path(output_dir) if isinstance(output_dir, str) else output_dir
        )

    def run(
        self,
        X_train: pd.DataFrame,
        y_train: pd.Series,
        X_test: Optional[pd.DataFrame] = None,
        baseline_exp_result: Optional[ExperimentResult] = None,
        skip_full_training: bool = False,
    ) -> ExperimentResult:
        print(f"[Experiment: {self.config.name}]")
        print(f"Cross-validating ({self.config.cv_folds}-folds)...")

        scores = cross_validate(
            self.config.pipeline,
            X_train,
            y_train,
            cv=self.config.cv_folds,
            scoring=self.config.scoring,
            n_jobs=-1,
        )

        cv_score = -scores["test_score"].mean().item()
        cv_std = scores["test_score"].std().item()

        print(f"CV score: {cv_score:.4f} B± {cv_std:.4f}")
        if baseline_exp_result is not None:
            delta_cv_score = cv_score - baseline_exp_result.cv_score
            delta_cv_std = cv_std - baseline_exp_result.cv_std
            print(
                f"          {delta_cv_score:+.4f}  {delta_cv_std:+.4f} compared to {baseline_exp_result.name} (Negative is better)"
            )

        if not skip_full_training:
            print("Training on full training set...")
            self.config.pipeline.fit(X_train, y_train)

        if X_test is not None:
            print("Creating submission on test set...")
            submission_path = self.create_submission(X_test, self.output_dir)
            print(f"Submission created: {submission_path}")

        print("Experiment complete\n")

        return ExperimentResult(
            self.config.name, self.config.pipeline, cv_score, cv_std
        )

    def create_submission(self, X_test, dir: str | Path) -> Path:
        dir = Path(dir) if isinstance(dir, str) else dir
        if not dir.exists():
            dir.mkdir(parents=True)
        path = dir / f"{self.config.name}.csv"

        predictions = self.config.pipeline.predict(X_test)
        submission = pd.DataFrame({"ID": X_test.index, "Price": predictions})
        submission.to_csv(path, index=False)

        return path
```