Lesson 32: Complete ML Pipeline

Data Science with Python – BSc Course

45 Minutes

Learning Objectives

The Problem: We've learned many techniques separately. How do we combine preprocessing, feature engineering, and modeling into a reproducible, leak-free workflow?

After this lesson, you will be able to:

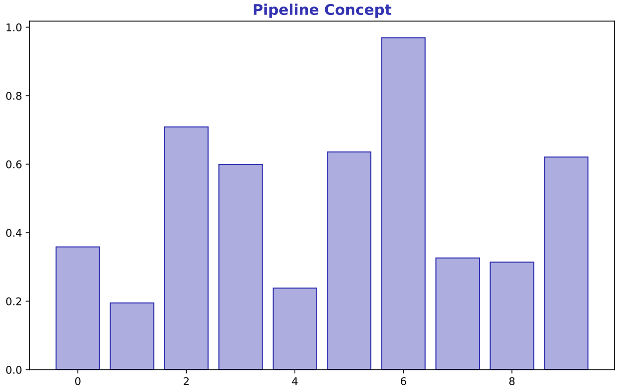
- Build sklearn pipelines for end-to-end workflows
- Apply cross-validation correctly (avoiding data leakage)
- Tune hyperparameters with GridSearchCV and RandomizedSearchCV
- Handle time series data with proper train/test splits

Finance Application: Production-ready ML systems for trading and risk

Pipeline Concept

Chaining Steps Together

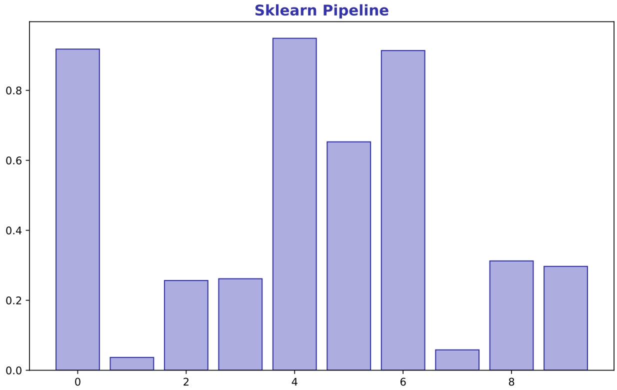
- Pipeline = sequence of transformers + final estimator
- Each step's output becomes next step's input



Pipelines ensure transformations are applied consistently to train and test data

Building Your First Pipeline

- `Pipeline([('scaler', StandardScaler()), ('model', Ridge())])`
- Call `.fit(X_train, y_train)` and `.predict(X_test)`

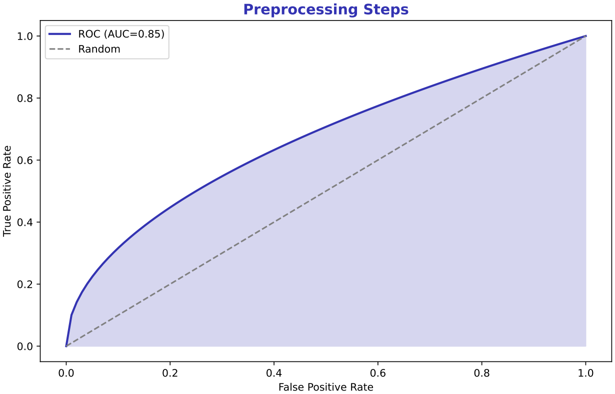


Naming convention: ('step_name', transformer_object)

Preprocessing Steps

Common Transformers

- StandardScaler, MinMaxScaler for numeric features
- OneHotEncoder for categorical features
- SimpleImputer for missing values

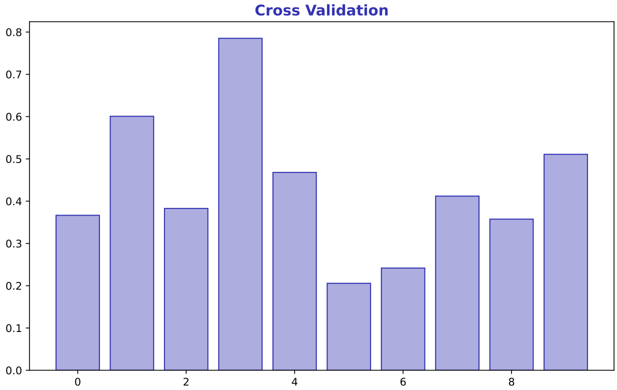


Use ColumnTransformer to apply different transforms to different columns

Cross-Validation

Robust Performance Estimation

- K-Fold: split data K ways, train on K-1, test on 1, rotate
- `cross_val_score(pipeline, X, y, cv=5)` returns K scores

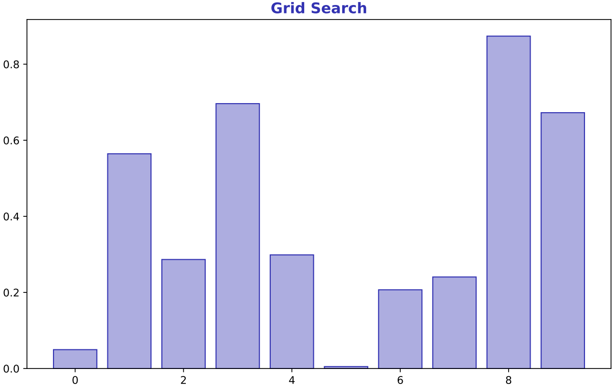


CV inside pipeline = transformers fit only on training fold (no leakage)

GridSearchCV

Exhaustive Hyperparameter Search

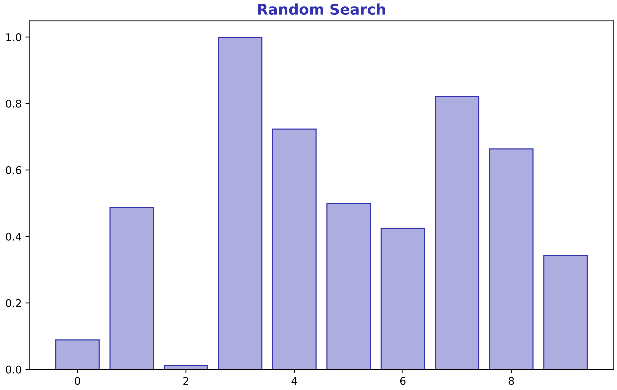
- Define parameter grid: {'model__alpha': [0.1, 1, 10]}
- GridSearchCV tries all combinations with CV



Access best params via `grid.best_params_` **and best score via** `grid.best_score_`

Efficient Search for Large Spaces

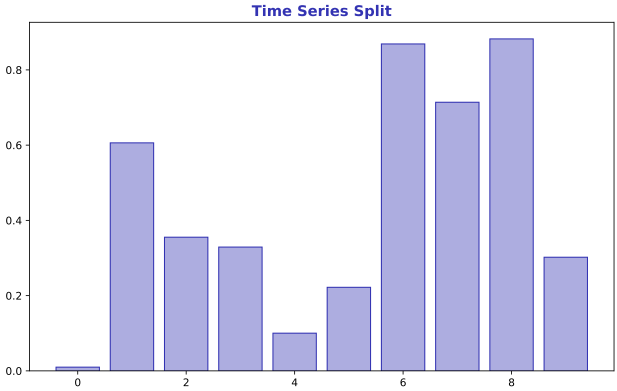
- Sample random combinations instead of exhaustive grid
- Specify distributions: `uniform(0.01, 10)` for continuous params



Rule: Use random search when grid has >100 combinations

Respecting Temporal Order

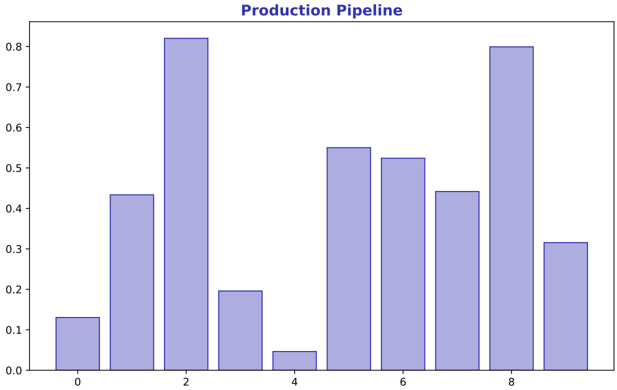
- Standard CV shuffles – invalid for time series
- TimeSeriesSplit: train on past, test on future (rolling)



Finance critical: Never let future data leak into training

From Development to Deployment

- Save entire pipeline with `joblib.dump(pipe, 'model.pkl')`
- Load and predict: `pipe = joblib.load('model.pkl')`



Pipeline saves all preprocessing steps – deploy once, predict anywhere

Hands-On Exercise (25 min)

Task: Build End-to-End Prediction Pipeline

1. Create pipeline: StandardScaler $\rightarrow$ PCA(5) $\rightarrow$ Ridge
2. Use GridSearchCV to tune `pca__n_components` and `ridge__alpha`
3. Evaluate with TimeSeriesSplit (5 splits)
4. Print best parameters and cross-validation score
5. Save best pipeline to disk with joblib

Deliverable: Best params + CV score + saved model file.

Extension: Add ColumnTransformer for mixed numeric/categorical features

Lesson Summary

Problem Solved: We can now build complete, reproducible ML workflows that prevent data leakage.

Key Takeaways:

- Pipelines chain preprocessing and modeling together
- Cross-validation estimates generalization performance
- GridSearchCV/RandomizedSearchCV for hyperparameter tuning
- TimeSeriesSplit for financial data – never leak future

Next Lesson: Perceptron (L33) – introduction to neural networks

Memory: Pipeline = chain. GridSearchCV = try all. TimeSeriesSplit = respect time.