

Model Persistence: Save, Load, and Deploy

Why Model Persistence?

WHY SAVE MODELS?

1. REPRODUCIBILITY

- Same predictions every time
- Audit trail for compliance
- Version control models

2. DEPLOYMENT

- Train once, predict many times
- No need to re-fit on server
- Faster predictions

3. SHARING

- Share models with team
- Model registry (MLflow)
- API serving

COMMON FORMATS:

pickle / joblib

- + Fast, easy
- + Works with sklearn
- Python version sensitive
- Security risk (arbitrary code)

ONNX

- + Framework agnostic
- + Production-ready
- More complex setup

JSON (coefficients only)

- + Human readable

pickle & joblib Code

```
+ Framework agnostic
Saving and Loading sklearn Models

import pickle
import joblib
from sklearn.linear_model import Ridge

# =====
# METHOD 1: pickle
# =====
# Save
with open('model.pkl', 'wb') as f:
    pickle.dump(model, f)

# Load
with open('model.pkl', 'rb') as f:
    loaded_model = pickle.load(f)

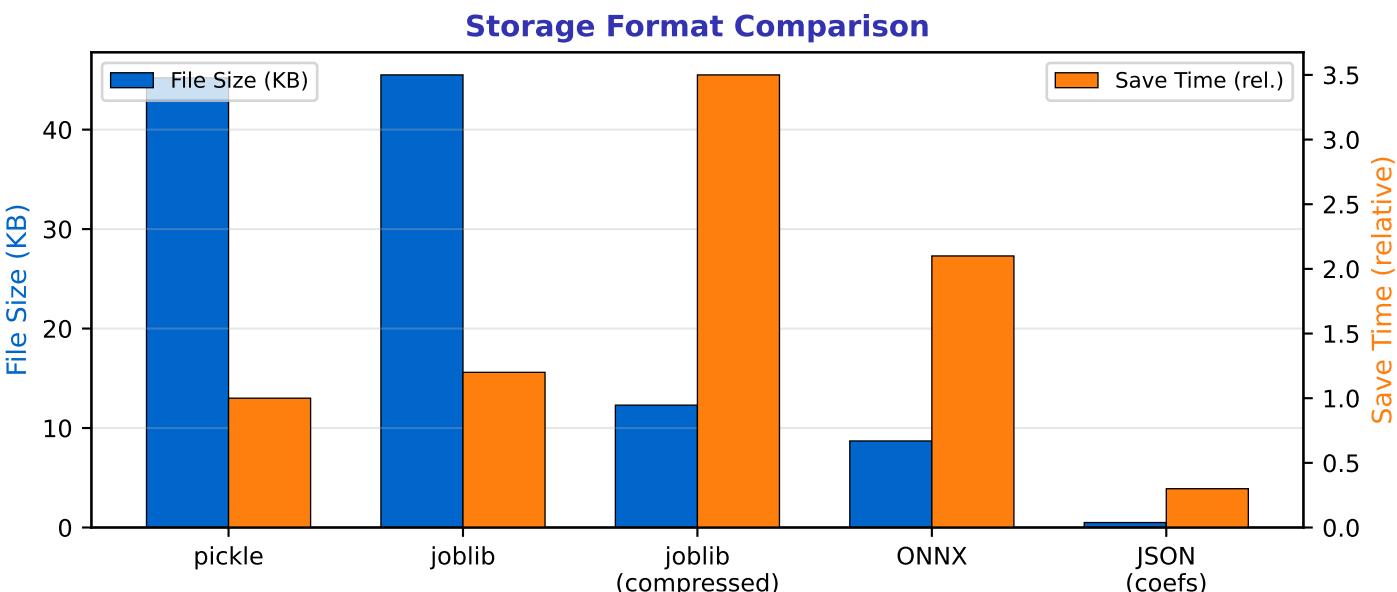
# =====
# METHOD 2: joblib (recommended for sklearn)
# =====
# Save (uncompressed)
joblib.dump(model, 'model.joblib')

# Save (compressed - smaller file)
joblib.dump(model, 'model.joblib.gz', compress=3)

# Load
loaded_model = joblib.load('model.joblib')

# =====
# Use loaded model
# =====
predictions = loaded_model.predict(X_new)

# Verify it's the same
assert np.allclose(model.coef_, loaded_model.coef_)
```



Best Practices

MODEL PERSISTENCE BEST PRACTICES

1. VERSION EVERYTHING
model_v1.0.0_2024-01-15.joblib
 - Include version number
 - Include training date
 - Track with git/DVC
2. SAVE METADATA
 - {
 - "model_type": "Ridge",
 - "features": ["MKT", "SMB", "HML"],
 - "train_r2": 0.73,
 - "train_date": "2024-01-15",
 - "sklearn_version": "1.3.0",
 - "python_version": "3.10.12"}
3. SAVE PREPROCESSING TOO
 - StandardScaler parameters
 - Feature names and order
 - Or use Pipeline (saves everything)
4. SECURITY
 - Never load untrusted pickles!
 - Pickle can execute arbitrary code
 - Use joblib with trusted sources only
5. TESTING
 - # After loading:
assert model.feature_names_in_ == expected_features
 - assert model.predict(X_test[:1]).shape == (1,)