

modeling_pipeline

November 23, 2025

1 Cloud AutoScale - Modeling Pipeline

Production-grade feature engineering and ML forecasting

```
[9]: import sys
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pathlib import Path
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

sys.path.insert(0, str(Path.cwd().parent))
from cloud_autoscale.data import SyntheticLoader, GCP2019Loader

print(" Imports successful")
```

Imports successful

1.1 1. Load Data

```
[10]: # Load synthetic data
loader = SyntheticLoader(pattern="bursty", duration_minutes=1440,
    ↪ step_minutes=5, seed=42)
df = loader.load()
print(f"Loaded {len(df)} steps")
```

Loaded 288 steps

1.2 2. Feature Engineering

```
[11]: # Create lag features
for lag in [1, 3, 6, 12]:
    df[f"cpu_lag{lag}"] = df["cpu_demand"].shift(lag)

# Rolling stats
for w in [3, 6, 12]:
    df[f"cpu_ma{w}"] = df["cpu_demand"].rolling(w, min_periods=1).mean()
```

```
df[f"cpu_std{w}"] = df["cpu_demand"].rolling(w, min_periods=1).std().
    ↪fillna(0)
```

```
# Cyclical features
df["hour_sin"] = np.sin(2*np.pi*df["step"]/288)
df["hour_cos"] = np.cos(2*np.pi*df["step"]/288)

df = df.dropna()
print(f"Features: {len(df.columns)}")
print(f"Shape: {len(df.shape)}")
```

Features: 22

Shape: 2

1.3 3. Train-Test Split

```
[12]: split = int(len(df)*0.8)
train, test = df.iloc[:split], df.iloc[split:]

feature_cols = [c for c in df.columns if c not in
    ↪["step", "time", "cpu_demand", "mem_demand", "new_instances", "machines_reporting"]]
X_train, y_train = train[feature_cols], train["cpu_demand"]
X_test, y_test = test[feature_cols], test["cpu_demand"]

print(f"Train: {len(X_train)}, Test: {len(X_test)}")
```

Train: 0, Test: 0

1.4 4. Train Models

```
[13]: # Linear Regression
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

print(f"MAE: {mae:.4f}")
print(f"RMSE: {rmse:.4f}")
print(f"R²: {r2:.4f}")
```

ValueError

Traceback (most recent call last)

Cell In[13], line 3

```
1 # Linear Regression
2 lr = LinearRegression()
```

```

----> 3 lr.fit(X_train, y_train)
      4 y_pred = lr.predict(X_test)
      6 mae = mean_absolute_error(y_test, y_pred)

```

```

File ~/Documents/00-Projects/_AUS/Cloud-AutoScale/.venv/lib/python3.13/
↳site-packages/sklearn/base.py:1365, in _fit_context.<locals>.decorator.
↳<locals>.wrapper(estimator, *args, **kwargs)
    1358     estimator._validate_params()
    1360 with config_context(
    1361     skip_parameter_validation=(
    1362         prefer_skip_nested_validation or global_skip_validation
    1363     )
    1364 ):
-> 1365     return fit_method(estimator, *args, **kwargs)

```

```

File ~/Documents/00-Projects/_AUS/Cloud-AutoScale/.venv/lib/python3.13/
↳site-packages/sklearn/linear_model/_base.py:618, in LinearRegression.fit(self, X, y, sample_weight)
    614 n_jobs_ = self.n_jobs
    616 accept_sparse = False if self.positive else ["csr", "csc", "coo"]
-> 618 X, y = validate_data(
    619     self,
    620     X,
    621     y,
    622     accept_sparse=accept_sparse,
    623     y_numeric=True,
    624     multi_output=True,
    625     force_writeable=True,
    626 )
    628 has_sw = sample_weight is not None
    629 if has_sw:

```

```

File ~/Documents/00-Projects/_AUS/Cloud-AutoScale/.venv/lib/python3.13/
↳site-packages/sklearn/utils/validation.py:2971, in validate_data(_estimator, X, y, reset, validate_separately, skip_check_array, **check_params)
    2969     y = check_array(y, input_name="y", **check_y_params)
    2970 else:
-> 2971     X, y = check_X_y(X, y, **check_params)
    2972     out = X, y
    2974 if not no_val_X and check_params.get("ensure_2d", True):

```

```

File ~/Documents/00-Projects/_AUS/Cloud-AutoScale/.venv/lib/python3.13/
↳site-packages/sklearn/utils/validation.py:1368, in check_X_y(X, y, accept_sparse, accept_large_sparse, dtype, order, copy, force_writeable, force_all_finite, ensure_all_finite, ensure_2d, allow_nd, multi_output, ensure_min_samples, ensure_min_features, y_numeric, estimator)
    1362     raise ValueError(
    1363         f"{estimator_name} requires y to be passed, but the target y is,
↳None"

```

```

1364     )
1365     ensure_all_finite = _deprecate_force_all_finite(force_all_finite,
↳ensure_all_finite)
-> 1368 X = check_array(
1369     X,
1370     accept_sparse=accept_sparse,
1371     accept_large_sparse=accept_large_sparse,
1372     dtype=dtype,
1373     order=order,
1374     copy=copy,
1375     force_writeable=force_writeable,
1376     ensure_all_finite=ensure_all_finite,
1377     ensure_2d=ensure_2d,
1378     allow_nd=allow_nd,
1379     ensure_min_samples=ensure_min_samples,
1380     ensure_min_features=ensure_min_features,
1381     estimator=estimator,
1382     input_name= ,
1383 )
1384 y = _check_y(y, multi_output=multi_output, y_numeric=y_numeric,
↳estimator=estimator)
1385 check_consistent_length(X, y)

File ~/Documents/00-Projects/_AUS/Cloud-AutoScale/.venv/lib/python3.13/
↳site-packages/sklearn/utils/validation.py:1128, in check_array(array,
↳accept_sparse, accept_large_sparse, dtype, order, copy, force_writeable,
↳force_all_finite, ensure_all_finite, ensure_non_negative, ensure_2d, allow_nd
↳ensure_min_samples, ensure_min_features, estimator, input_name)
1126     n_samples = _num_samples(array)
1127     if n_samples < ensure_min_samples:
-> 1128         raise ValueError(
1129             "Found array with %d sample(s) (shape=%s) while a"
1130             " minimum of %d is required%s."
1131             % (n_samples, array.shape, ensure_min_samples, context)
1132         )
1133 if ensure_min_features > 0 and array.ndim == 2:
1134     n_features = array.shape[1]

ValueError: Found array with 0 sample(s) (shape=(0, 16)) while a minimum of 1 i
↳required by LinearRegression.

```

1.5 5. Visualize Results

```

[ ]: plt.figure(figsize=(14,6))
plt.plot(y_test.values[:100], label="Actual")
plt.plot(y_pred[:100], label="Predicted")
plt.legend()

```

```
plt.title("Demand Forecast")  
plt.show()
```