hgb_features_modeling

September 21, 2025

1 Arrest Prediction — v1 (HistGradientBoosting + Engineered Features)

Goal: Beat RF v0 (PR-AUC 0.623) by adding time features, rare bucketing, and frequency encodings, then training a HistGradientBoosting baseline.

Dataset: data/processed/arrest_features.csv
Target: arrest (0/1)
Artifacts: saved to notebooks/artifacts/

[20]:
```
# Core imports
import os, time, json, numpy as np, pandas as pd
from pathlib import Path

# Modeling + metrics
from sklearn.model_selection import train_test_split, StratifiedKFold,␣
  ↪RandomizedSearchCV
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.metrics import (
    average_precision_score, roc_auc_score, classification_report,
    confusion_matrix, precision_recall_curve, roc_curve
)
from sklearn.experimental import enable_hist_gradient_boosting
from sklearn.ensemble import HistGradientBoostingClassifier

import matplotlib.pyplot as plt
from scipy.stats import loguniform, randint
import tempfile
from sklearn.metrics import precision_recall_fscore_support

# Paths
REPO = Path.cwd()
while REPO.name != "chicago-crime-pipeline" and REPO.parent != REPO:
    REPO = REPO.parent
DATA = REPO / "data" / "processed"
ART = REPO / "notebooks" / "artifacts"
```

```
ART.mkdir(parents=True, exist_ok=True)

# Load
df = pd.read_csv(DATA / "arrest_features.csv")
assert "arrest" in df.columns
print(df.shape, df["arrest"].value_counts(dropna=False).to_dict())

# Split (same seed/stratify as v0)
TARGET = "arrest"
y = df[TARGET].astype(int).values
X = df.drop(columns=[TARGET]).copy()
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
X_train.shape, X_test.shape
```

```
(10482, 10) {0: 8972, 1: 1510}
```

```
[20]: ((8385, 9), (2097, 9))
```

```
[21]: def slice_metrics(X_df, y_true, proba, threshold, slice_col, min_support=40):
    """
    Compute precision/recall/F1 per value of a categorical slice column.
    Saves nothing; just returns a DataFrame. You can write it to CSV after.
    """
    if slice_col not in X_df.columns:
        print(f"[skip] slice column not found: {slice_col}")
        return None

    df = pd.DataFrame({
        slice_col: X_df[slice_col],
        "y": y_true,
        "pred": (proba >= threshold).astype(int)
    })

    rows = []
    for val, g in df.groupby(slice_col):
        n = len(g)
        if n < min_support:
            continue
        p, r, f1, _ = precision_recall_fscore_support(
            g["y"], g["pred"], average="binary", zero_division=0
        )
        rows.append({
            slice_col: val, "support": int(n),
            "precision": float(p), "recall": float(r), "f1": float(f1)
        })
    if not rows:
        print(f"[note] no slices with support  {min_support} for {slice_col}")
        return None

    return pd.DataFrame(rows).sort_values("f1", ascending=False).
↪reset_index(drop=True)
```

[ ]:

```

```

```
Saved slice metrics for weekday → /Volumes/easystore/Projects/chicago-crime-
pipeline/notebooks/artifacts/slice_metrics_weekday_hgb_v1_2025090921-232438.csv
Saved slice metrics for hour_bin → /Volumes/easystore/Projects/chicago-crime-
pipeline/notebooks/artifacts/slice_metrics_hour_bin_hgb_v1_2025090921-232438.csv
Saved slice metrics for primary_type → /Volumes/easystore/Projects/chicago-
crime-pipeline/notebooks/artifacts/slice_metrics_primary_type_hgb_v1_2025090921-
232438.csv

=== weekday: top 5 by F1 ===

/var/folders/6z/l9wv3crd4n5bzgcrdd7vxq8m0000gn/T/ipykernel_79209/847869267.py:17
: FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  for val, g in df.groupby(slice_col):

    weekday  support  precision    recall        f1
0  Saturday      259   0.807692  0.700000  0.750000
1    Friday      311   0.682927  0.651163  0.666667
2  Thursday      379   0.701754  0.634921  0.666667
3    Sunday      298   0.727273  0.558140  0.631579
4    Monday      288   0.625000  0.571429  0.597015

=== weekday: bottom 5 by F1 (support   40) ===

     weekday  support  precision    recall        f1
2   Thursday      379   0.701754  0.634921  0.666667
3     Sunday      298   0.727273  0.558140  0.631579
4     Monday      288   0.625000  0.571429  0.597015
5  Wednesday      286   0.615385  0.533333  0.571429
6    Tuesday      276   0.490196  0.581395  0.531915


=== hour_bin: top 5 by F1 ===

  hour_bin  support  precision    recall        f1
0    18-23      622   0.666667  0.626263  0.645833
1    06-11      426   0.666667  0.603774  0.633663
2    12-17      638   0.626374  0.612903  0.619565
3    00-05      411   0.659574  0.543860  0.596154

=== hour_bin: bottom 5 by F1 (support   40) ===
  hour_bin  support  precision    recall        f1
0    18-23      622   0.666667  0.626263  0.645833
1    06-11      426   0.666667  0.603774  0.633663
2    12-17      638   0.626374  0.612903  0.619565
3    00-05      411   0.659574  0.543860  0.596154


=== primary_type: top 5 by F1 ===

        primary_type  support  precision    recall        f1
0          NARCOTICS       58   0.913793  1.000000  0.954955
1  WEAPONS VIOLATION       53   0.730769  1.000000  0.844444
2            ROBBERY       56   0.800000  0.571429  0.666667
3      OTHER OFFENSE      134   0.515152  0.680000  0.586207
4  CRIMINAL TRESPASS       43   0.392857  0.916667  0.550000

=== primary_type: bottom 5 by F1 (support   40) ===

            primary_type  support  precision    recall        f1
7                BATTERY      379   0.607143  0.283333  0.386364
8                  THEFT      473   0.419355  0.325000  0.366197
9        CRIMINAL DAMAGE      232   1.000000  0.111111  0.200000
10               ASSAULT      187   0.400000  0.090909  0.148148
11   MOTOR VEHICLE THEFT      160   0.000000  0.000000  0.000000
```

[31]:
```python
# Rebuild engineered features from X_train/X_test
X_train_fe = X_train.copy()
X_test_fe  = X_test.copy()

# Weekday
for Xdf in (X_train_fe, X_test_fe):
    Xdf["weekday"] = pd.to_datetime(Xdf["date"]).dt.day_name()

# Hour bins (ensure object dtype)
bins   = [0,6,12,18,24]
labels = ["00-05","06-11","12-17","18-23"]
for Xdf in (X_train_fe, X_test_fe):
    Xdf["hour_bin"] = pd.cut(Xdf["hour"].astype(int), bins=bins, right=False,␣
 ↪labels=labels).astype(object)

# Rare bucket helper
def rare_bucket(train_col, test_col, min_count=40):
    vc = train_col.value_counts()
    keep = set(vc[vc >= min_count].index)
    return (train_col.where(train_col.isin(keep), "__RARE__"),
            test_col.where(test_col.isin(keep), "__RARE__"))

# Rare bucket base categoricals if present
for col in ["location_description", "primary_type"]:
    if col in X_train_fe.columns:
        X_train_fe[col], X_test_fe[col] = rare_bucket(X_train_fe[col],␣
↪X_test_fe[col], 40)

# Frequency encodes
def add_freq_encode(col):
    freq = X_train_fe[col].astype(object).value_counts(normalize=True)
    X_train_fe[f"{col}_freq"] = X_train_fe[col].map(freq).astype("float64").
↪fillna(0.0).to_numpy()
    X_test_fe[f"{col}_freq"]  = X_test_fe[col].map(freq).astype("float64").
↪fillna(0.0).to_numpy()

for col in ["primary_type","location_description","weekday","hour_bin"]:
    if col in X_train_fe.columns: add_freq_encode(col)

# Target mean for primary_type
if "primary_type" in X_train_fe.columns:
    arrest_rate = pd.Series(y_train).groupby(X_train_fe["primary_type"]).mean()
    X_train_fe["ptype_arrest_rate"] = X_train_fe["primary_type"].
↪map(arrest_rate)
    X_test_fe["ptype_arrest_rate"]  = X_test_fe["primary_type"].
↪map(arrest_rate).fillna(float(arrest_rate.mean()))
else:
    X_train_fe["ptype_arrest_rate"] = 0.0
    X_test_fe["ptype_arrest_rate"]  = 0.0

# Interaction primary_type × hour_bin (rare-bucket)
if set(["primary_type","hour_bin"]).issubset(X_train_fe.columns):
    X_train_fe["ptype_x_hourbin"] = X_train_fe["primary_type"].astype(str) +␣
↪"_" + X_train_fe["hour_bin"].astype(str)
    X_test_fe["ptype_x_hourbin"]  = X_test_fe["primary_type"].astype(str)  +␣
↪"_" + X_test_fe["hour_bin"].astype(str)
    X_train_fe["ptype_x_hourbin"], X_test_fe["ptype_x_hourbin"] = rare_bucket(
        X_train_fe["ptype_x_hourbin"], X_test_fe["ptype_x_hourbin"], 30
    )
else:
    X_train_fe["ptype_x_hourbin"] = "__MISSING__"
    X_test_fe["ptype_x_hourbin"]  = "__MISSING__"
```

[32]:
```
# Your lists
cat_cols_fe =␣
↪["date","primary_type","location_description","location_grouped","weekday","hour_bin","ptyp
num_cols_fe =␣
↪["id","year","month","dow","hour","primary_type_freq","location_description_freq","weekday_

# Remove columns that don't exist (e.g., location_grouped may be absent)
```

```
present = set(X_train_fe.columns)
cat_cols_used = [c for c in cat_cols_fe if c in present]
num_cols_used = [c for c in num_cols_fe if c in present]

print("Using categorical:", cat_cols_used)
print("Using numeric    :", num_cols_used)

pre_fe = ColumnTransformer(
    transformers=[
        ("cat", OneHotEncoder(handle_unknown="ignore", sparse_output=False),␣
 ↪cat_cols_used),
        ("num", "passthrough", num_cols_used),
    ],
    remainder="drop",
    verbose_feature_names_out=False,
)
```

```
Using categorical: ['date', 'primary_type', 'location_description',
'location_grouped', 'weekday', 'hour_bin', 'ptype_x_hourbin']
Using numeric    : ['id', 'year', 'month', 'dow', 'hour', 'primary_type_freq',
'location_description_freq', 'weekday_freq', 'hour_bin_freq',
'ptype_arrest_rate']
```

[30]:
```
from sklearn.model_selection import train_test_split

# Subsample ~5k rows for faster search
SUB_N = 5000
if len(y_train) > SUB_N:
    X_sub, _, y_sub, _ = train_test_split(
        X_train_fe, y_train, train_size=SUB_N,
        stratify=y_train, random_state=42
    )
else:
    X_sub, y_sub = X_train_fe, y_train

hgb_search = RandomizedSearchCV(
    hgb_pipe, param_distributions=param_dist,
    n_iter=6,               # 6 candidates
    scoring="average_precision",
    refit=True, cv=2,       # 2 folds
    n_jobs=-1, random_state=42, verbose=2
)

hgb_search.fit(X_sub, y_sub, clf__sample_weight=sw_train[:len(y_sub)])
print("Best HGB params:", hgb_search.best_params_)
print("Best CV PR-AUC:", round(hgb_search.best_score_, 4))
```

```
Fitting 2 folds for each of 6 candidates, totalling 12 fits
```

```
[CV] END clf__l2_regularization=0.0003487351559952693,
clf__learning_rate=0.06207090305742937, clf__max_depth=5, clf__max_iter=90,
clf__max_leaf_nodes=39, clf__min_samples_leaf=176; total time=   0.0s
[CV] END clf__l2_regularization=0.00010062545641808922,
clf__learning_rate=0.1978522015446167, clf__max_depth=3, clf__max_iter=91,
clf__max_leaf_nodes=41, clf__min_samples_leaf=81; total time=   0.0s
[CV] END clf__l2_regularization=0.00010062545641808922,
clf__learning_rate=0.1978522015446167, clf__max_depth=3, clf__max_iter=91,
clf__max_leaf_nodes=41, clf__min_samples_leaf=81; total time=   0.0s
[CV] END clf__l2_regularization=0.0020059560245279666,
clf__learning_rate=0.18679147494991152, clf__max_depth=5, clf__max_iter=87,
clf__max_leaf_nodes=44, clf__min_samples_leaf=80; total time=   0.0s
[CV] END clf__l2_regularization=0.012306931514988042,
clf__learning_rate=0.1334357418214006, clf__max_depth=7, clf__max_iter=81,
clf__max_leaf_nodes=39, clf__min_samples_leaf=97; total time=   0.0s
[CV] END clf__l2_regularization=0.0003487351559952693,
clf__learning_rate=0.06207090305742937, clf__max_depth=5, clf__max_iter=90,
clf__max_leaf_nodes=39, clf__min_samples_leaf=176; total time=   0.0s
[CV] END clf__l2_regularization=0.012306931514988042,
clf__learning_rate=0.1334357418214006, clf__max_depth=7, clf__max_iter=81,
clf__max_leaf_nodes=39, clf__min_samples_leaf=97; total time=   0.0s
[CV] END clf__l2_regularization=0.0020059560245279666,
clf__learning_rate=0.18679147494991152, clf__max_depth=5, clf__max_iter=87,
clf__max_leaf_nodes=44, clf__min_samples_leaf=80; total time=   0.0s
[CV] END clf__l2_regularization=0.00010582064396389428,
clf__learning_rate=0.051624394129231366, clf__max_depth=5, clf__max_iter=138,
clf__max_leaf_nodes=46, clf__min_samples_leaf=74; total time=   0.0s
[CV] END clf__l2_regularization=0.00010582064396389428,
clf__learning_rate=0.051624394129231366, clf__max_depth=5, clf__max_iter=138,
clf__max_leaf_nodes=46, clf__min_samples_leaf=74; total time=   0.0s
[CV] END clf__l2_regularization=0.003853103152262984,
clf__learning_rate=0.1484885740998741, clf__max_depth=5, clf__max_iter=123,
clf__max_leaf_nodes=38, clf__min_samples_leaf=123; total time=   0.0s
[CV] END clf__l2_regularization=0.003853103152262984,
clf__learning_rate=0.1484885740998741, clf__max_depth=5, clf__max_iter=123,
clf__max_leaf_nodes=38, clf__min_samples_leaf=123; total time=   0.0s
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[30], line 21
     11     X_sub, y_sub = X_train_fe, y_train
     13 hgb_search = RandomizedSearchCV(
     14     hgb_pipe, param_distributions=param_dist,
     15     n_iter=6,               # 6 candidates
   (…)     18     n_jobs=-1, random_state=42, verbose=2
     19 )
---> 21 hgb_search.fit(X_sub, y_sub, clf__sample_weight=sw_train[:len(y_sub)])
```

```
     22 print("Best HGB params:", hgb_search.best_params_)
     23 print("Best CV PR-AUC:", round(hgb_search.best_score_, 4))

File /Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12/
↪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 /Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12/
↪site-packages/sklearn/model_selection/_search.py:1051, in BaseSearchCV.
↪fit(self, X, y, **params)
   1045     results = self._format_results(
   1046         all_candidate_params, n_splits, all_out, all_more_results
   1047     )
   1049     return results
-> 1051 self._run_search(evaluate_candidates)
   1053 # multimetric is determined here because in the case of a callable
   1054 # self.scoring the return type is only known after calling
   1055 first_test_score = all_out[0]["test_scores"]

File /Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12/
↪site-packages/sklearn/model_selection/_search.py:1992, in RandomizedSearchCV.
↪_run_search(self, evaluate_candidates)
   1990 def _run_search(self, evaluate_candidates):
   1991     """Search n_iter candidates from param_distributions"""
-> 1992     evaluate_candidates(
   1993         ParameterSampler(
   1994␣
↪            self.param_distributions, self.n_iter, random_state=self.random_state
   1995         )
   1996     )

File /Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12/
↪site-packages/sklearn/model_selection/_search.py:1028, in BaseSearchCV.fit.
↪<locals>.evaluate_candidates(candidate_params, cv, more_results)
   1021 elif len(out) != n_candidates * n_splits:
   1022     raise ValueError(
   1023         "cv.split and cv.get_n_splits returned "
   1024         "inconsistent results. Expected {} "
   1025         "splits, got {}".format(n_splits, len(out) // n_candidates)
   1026     )
-> 1028 _warn_or_raise_about_fit_failures(out, self.error_score)
   1030 # For callable self.scoring, the return type is only know after
   1031 # calling. If the return type is a dictionary, the error scores
   1032 # can now be inserted with the correct key. The type checking
   1033 # of out will be done in `_insert_error_scores`.
   1034 if callable(self.scoring):

File /Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12/
 ↪site-packages/sklearn/model_selection/_validation.py:505, in␣
 ↪_warn_or_raise_about_fit_failures(results, error_score)
    498 if num_failed_fits == num_fits:
    499     all_fits_failed_message = (
    500         f"\nAll the {num_fits} fits failed.\n"
    501         "It is very likely that your model is misconfigured.\n"
    502         "You can try to debug the error by setting error_score='raise'.
 ↪\n\n"
    503         f"Below are more details about the failures:
 ↪\n{fit_errors_summary}"
    504     )
--> 505     raise ValueError(all_fits_failed_message)
    507 else:
    508     some_fits_failed_message = (
    509         f"\n{num_failed_fits} fits failed out of a total of {num_fits}.
 ↪\n"
    510         "The score on these train-test partitions for these parameters"
   (…)    514         f"Below are more details about the failures:
 ↪\n{fit_errors_summary}"
    515     )

ValueError:
All the 12 fits failed.
It is very likely that your model is misconfigured.
You can try to debug the error by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
12 fits failed with the following error:
Traceback (most recent call last):
  File "/Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12
 ↪site-packages/pandas/core/indexes/base.py", line 3805, in get_loc
    return self._engine.get_loc(casted_key)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  File "index.pyx", line 167, in pandas._libs.index.IndexEngine.get_loc
  File "index.pyx", line 196, in pandas._libs.index.IndexEngine.get_loc
  File "pandas/_libs/hashtable_class_helper.pxi", line 7081, in pandas._libs.
 ↪hashtable.PyObjectHashTable.get_item
  File "pandas/_libs/hashtable_class_helper.pxi", line 7089, in pandas._libs.
 ↪hashtable.PyObjectHashTable.get_item
KeyError: 'ptype_arrest_rate'
```

```
The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12
 ↪site-packages/sklearn/utils/_indexing.py", line 443, in _get_column_indices
    col_idx = all_columns.get_loc(col)
              ^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12
 ↪site-packages/pandas/core/indexes/base.py", line 3812, in get_loc
    raise KeyError(key) from err
KeyError: 'ptype_arrest_rate'

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12
 ↪site-packages/sklearn/model_selection/_validation.py", line 859, in␣
 ↪_fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12
 ↪site-packages/sklearn/base.py", line 1365, in wrapper
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12
 ↪site-packages/sklearn/pipeline.py", line 655, in fit
    Xt = self._fit(X, y, routed_params, raw_params=params)
         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12
 ↪site-packages/sklearn/pipeline.py", line 589, in _fit
    X, fitted_transformer = fit_transform_one_cached(
                            ^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12
 ↪site-packages/joblib/memory.py", line 607, in __call__
    return self._cached_call(args, kwargs, shelving=False)[0]
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12
 ↪site-packages/joblib/memory.py", line 562, in _cached_call
    return self._call(call_id, args, kwargs, shelving)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12
 ↪site-packages/joblib/memory.py", line 832, in _call
    output = self.func(*args, **kwargs)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12
 ↪site-packages/sklearn/pipeline.py", line 1540, in _fit_transform_one
    res = transformer.fit_transform(X, y, **params.get("fit_transform", {}))
          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12
 ↪site-packages/sklearn/utils/_set_output.py", line 316, in wrapped
    data_to_wrap = f(self, X, *args, **kwargs)
                   ^^^^^^^^^^^^^^^^^^^^^^^^^^^

  File "/Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12
 ↪site-packages/sklearn/base.py", line 1365, in wrapper
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  File "/Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12
 ↪site-packages/sklearn/compose/_column_transformer.py", line 988, in␣
 ↪fit_transform
    self._validate_column_callables(X)
  File "/Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12
 ↪site-packages/sklearn/compose/_column_transformer.py", line 541, in␣
 ↪_validate_column_callables
    transformer_to_input_indices[name] = _get_column_indices(X, columns)
                                         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  File "/Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12
 ↪site-packages/sklearn/utils/_indexing.py", line 451, in _get_column_indices
    raise ValueError("A given column is not a column of the dataframe") from e
ValueError: A given column is not a column of the dataframe
```

```
[13]: hgb_final = hgb_search.best_estimator_
      hgb_final.fit(X_train_fe, y_train, clf__sample_weight=sw_train)
```

```
[13]: Pipeline(memory='/var/folders/6z/l9wv3crd4n5bzgcrdd7vxq8m0000gn/T/tmptrqagad5',
               steps=[('pre',
                       ColumnTransformer(transformers=[('cat',
      OneHotEncoder(handle_unknown='ignore',
      sparse_output=False),
                                                        ['date', 'primary_type',
                                                         'location_description',
                                                         'location_grouped',
                                                         'weekday', 'hour_bin']),
                                                       ('num', 'passthrough',
                                                        ['id', 'year', 'month', 'dow',
                                                         'hour', 'primary_type_freq',
                                                         'location_description_freq',
                                                         'weekday_freq',
                                                         'hour_bin_freq'])],
                                         verbose_feature_names_out=False)),
                      ('clf',
                       HistGradientBoostingClassifier(l2_regularization=np.float64(0.0
      020059560245279666),
      learning_rate=np.float64(0.18679147494991152),
                                                      max_depth=5, max_iter=87,
                                                      max_leaf_nodes=44,
                                                      min_samples_leaf=80,
                                                      random_state=42))])
```

[14]:
```
proba_hgb = hgb_final.predict_proba(X_test_fe)[:,1]

print("HGB TEST PR-AUC:", round(average_precision_score(y_test, proba_hgb), 4))
print("HGB TEST ROC-AUC:", round(roc_auc_score(y_test, proba_hgb), 4))

# Threshold tuning
prec, rec, thr = precision_recall_curve(y_test, proba_hgb)
f1s = 2*prec*rec/(prec+rec+1e-12)
best_idx = np.nanargmax(f1s)
thr_hgb = thr[best_idx] if best_idx < len(thr) else 0.5
pred_hgb = (proba_hgb >= thr_hgb).astype(int)

print("Best threshold:", float(thr_hgb), "Best F1:", float(f1s[best_idx]))
print(classification_report(y_test, pred_hgb, digits=3))
print("Confusion:\n", confusion_matrix(y_test, pred_hgb))
```

```
HGB TEST PR-AUC: 0.6569
HGB TEST ROC-AUC: 0.8878
Best threshold: 0.6953415078096913 Best F1: 0.6265060240958864
              precision    recall  f1-score   support

           0      0.934     0.946     0.940      1795
           1      0.652     0.603     0.627       302

    accuracy                          0.897      2097
   macro avg      0.793     0.774     0.783      2097
weighted avg      0.893     0.897     0.895      2097

Confusion:
 [[1698   97]
 [ 120  182]]
```

[ ]:
```
if 'ART' not in globals():
    ART = Path("notebooks/artifacts"); ART.mkdir(parents=True, exist_ok=True)
if 'stamp' not in globals():
    import time; stamp = time.strftime("%Y%m%d-%H%M%S")

cols_to_check = ["weekday", "hour_bin", "primary_type"]  # adjust if needed
slice_tables = {}

for col in cols_to_check:
    tbl = slice_metrics(X_test_fe, y_test, proba_hgb, thr_hgb, col,␣
  ↪min_support=40)
```

```
    if tbl is not None:
        slice_tables[col] = tbl
        out_path = ART / f"slice_metrics_{col}_hgb_v1_{stamp}.csv"
        tbl.to_csv(out_path, index=False)
        print(f"Saved slice metrics for {col} → {out_path}")

# (optional) quick peek in the notebook
for col, tbl in slice_tables.items():
    print(f"\n=== {col}: top 5 by F1 ===")
    display(tbl.head(5))
    print(f"=== {col}: bottom 5 by F1 (support   40) ===")
    display(tbl.tail(5))
```

```
[15]: stamp = time.strftime("%Y%m%d-%H%M%S")

metrics = {
    "timestamp": stamp,
    "model": "HGB + FE v1",
    "test_pr_auc": float(average_precision_score(y_test, proba_hgb)),
    "test_roc_auc": float(roc_auc_score(y_test, proba_hgb)),
    "threshold_tuned": float(thr_hgb),
    "confusion_tuned": confusion_matrix(y_test, pred_hgb).tolist(),
    "class_report_tuned": classification_report(y_test, pred_hgb,␣
  ↪output_dict=True),
    "best_params": {k: (float(v) if hasattr(v, "item") else v) for k,v in␣
  ↪hgb_search.best_params_.items()}
}

with open(ART / f"metrics_hgb_v1_{stamp}.json", "w") as f:
    json.dump(metrics, f, indent=2)

with open(ART / "decision_threshold_hgb_v1.txt", "w") as f:
    f.write(str(metrics["threshold_tuned"]))

# PR/ROC plots
prec, rec, _ = precision_recall_curve(y_test, proba_hgb)
fpr, tpr, _ = roc_curve(y_test, proba_hgb)

plt.figure(); plt.plot(rec, prec); plt.xlabel("Recall"); plt.ylabel("Precision")
plt.title(f"HGB PR curve (AP={metrics['test_pr_auc']:.3f})"); plt.grid(True,␣
  ↪alpha=0.3)
plt.savefig(ART / f"pr_curve_hgb_v1_{stamp}.png", bbox_inches="tight"); plt.
  ↪close()

plt.figure(); plt.plot(fpr, tpr); plt.plot([0,1],[0,1],'--')
plt.xlabel("FPR"); plt.ylabel("TPR"); plt.title(f"HGB ROC curve␣
  ↪(AUC={metrics['test_roc_auc']:.3f})")
```

```
plt.grid(True, alpha=0.3)
plt.savefig(ART / f"roc_curve_hgb_v1_{stamp}.png", bbox_inches="tight"); plt.
    ↪close()

print("Saved HGB v1 artifacts:", ART)
```

```
Saved HGB v1 artifacts: /Volumes/easystore/Projects/chicago-crime-
pipeline/notebooks/artifacts
```

[23]:
```
for col in ["weekday","hour_bin","primary_type"]:
    out = slice_metrics(X_test_fe, y_test, proba_hgb, thr_hgb, col)
    if out is not None:
        out.to_csv(ART / f"slice_metrics_{col}_hgb_v1_{stamp}.csv", index=False)
        print(f"Saved slice metrics for {col}")
```

```
Saved slice metrics for weekday
Saved slice metrics for hour_bin
Saved slice metrics for primary_type
```

```
/var/folders/6z/l9wv3crd4n5bzgcrdd7vxq8m0000gn/T/ipykernel_79209/847869267.py:17
: FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  for val, g in df.groupby(slice_col):
```

[24]:
```
def threshold_for_recall(y_true, proba, target=0.70):
    prec, rec, thr = precision_recall_curve(y_true, proba)
    idx = np.argmax(rec >= target)
    th = thr[max(idx-1, 0)] if idx < len(thr) else 0.5
    return float(th), float(prec[max(idx-1,0)]), float(rec[max(idx-1,0)])

thr_r70, p_at_r70, r_at_r70 = threshold_for_recall(y_test, proba_hgb, target=0.
    ↪70)
print("Threshold for recall 0.70:", thr_r70, "| precision ", p_at_r70, "|␣
    ↪recall ", r_at_r70)

pred_r70 = (proba_hgb >= thr_r70).astype(int)
print(classification_report(y_test, pred_r70, digits=3))
print("Confusion:\n", confusion_matrix(y_test, pred_r70))
```

```
Threshold for recall 0.70: 0.0003309188924939405 | precision
0.14401525989508823 | recall  1.0
              precision    recall  f1-score   support

           0      0.000     0.000     0.000      1795
           1      0.144     1.000     0.252       302

    accuracy                          0.144      2097
   macro avg      0.072     0.500     0.126      2097
```

```
weighted avg      0.021     0.144     0.036      2097

Confusion:
 [[   0 1795]
 [   0  302]]
```

```
/Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12/site-
packages/sklearn/metrics/_classification.py:1731: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
/Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12/site-
packages/sklearn/metrics/_classification.py:1731: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
/Volumes/easystore/Projects/chicago-crime-pipeline/.venv/lib/python3.12/site-
packages/sklearn/metrics/_classification.py:1731: UndefinedMetricWarning:
Precision is ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
```

[25]:
```
slice_tables_r70 = {}
for col in slice_tables.keys():
    tbl = slice_metrics(X_test_fe, y_test, proba_hgb, thr_r70, col,␣
 ↪min_support=40)
    if tbl is not None:
        slice_tables_r70[col] = tbl
        display(tbl.head(5)); display(tbl.tail(5))
```

```
     weekday  support  precision  recall        f1
0   Thursday      379   0.166227     1.0  0.285068
1  Wednesday      286   0.157343     1.0  0.271903
2    Tuesday      276   0.155797     1.0  0.269592
3     Sunday      298   0.144295     1.0  0.252199
4     Friday      311   0.138264     1.0  0.242938
```

```
    weekday  support  precision  recall        f1
2   Tuesday      276   0.155797     1.0  0.269592
3    Sunday      298   0.144295     1.0  0.252199
4    Friday      311   0.138264     1.0  0.242938
5    Monday      288   0.121528     1.0  0.216718
6  Saturday      259   0.115830     1.0  0.207612
```

```
/var/folders/6z/l9wv3crd4n5bzgcrdd7vxq8m0000gn/T/ipykernel_79209/847869267.py:17
: FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  for val, g in df.groupby(slice_col):
```

```
  hour_bin  support  precision  recall        f1
0    18-23      622   0.159164     1.0  0.274619
1    12-17      638   0.145768     1.0  0.254446
2    00-05      411   0.138686     1.0  0.243590
3    06-11      426   0.124413     1.0  0.221294

  hour_bin  support  precision  recall        f1
0    18-23      622   0.159164     1.0  0.274619
1    12-17      638   0.145768     1.0  0.254446
2    00-05      411   0.138686     1.0  0.243590
3    06-11      426   0.124413     1.0  0.221294

         primary_type  support  precision  recall        f1
0           NARCOTICS       58   0.913793     1.0  0.954955
1  WEAPONS VIOLATION       53   0.716981     1.0  0.835165
2  CRIMINAL TRESPASS       43   0.279070     1.0  0.436364
3       OTHER OFFENSE      134   0.186567     1.0  0.314465
4             BATTERY      379   0.158311     1.0  0.273349

            primary_type  support  precision  recall        f1
7                  THEFT      473   0.084567     1.0  0.155945
8        CRIMINAL DAMAGE      232   0.038793     1.0  0.074689
9               BURGLARY       96   0.031250     1.0  0.060606
10    DECEPTIVE PRACTICE      136   0.029412     1.0  0.057143
11   MOTOR VEHICLE THEFT      160   0.018750     1.0  0.036810
```

[ ]:
```
# target mean encoding for primary_type
arrest_rate = pd.Series(y_train).groupby(X_train_fe["primary_type"]).mean()
X_train_fe["ptype_arrest_rate"] = X_train_fe["primary_type"].map(arrest_rate)
X_test_fe["ptype_arrest_rate"]  = X_test_fe["primary_type"].map(arrest_rate).
 ↪fillna(arrest_rate.mean())

num_cols_fe.append("ptype_arrest_rate")

# Add interaction features to reduce time-of-day false positives\
# Combine hour_bin x primary_type
X_train_fe["ptype_x_hourbin"] = X_train_fe["primary_type"].astype(str) + "_" +␣
 ↪X_train_fe["hour_bin"].astype(str)
X_test_fe["ptype_x_hourbin"] = X_test_fe["primary_type"].astype(str) + "_" +␣
 ↪X_test_fe["hour_bin"].astype(str)
```

[ ]:
```

```