In [28]:
```python
from polars import read_csv, col, Int64, min_horizontal, concat, Series, len as pl_len
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.metrics import log_loss, roc_auc_score, brier_score_loss
from xgboost import XGBClassifier
from numpy import corrcoef
from pathlib import Path
import pickle
```

Load data

In [2]:
```python
df = read_csv("ML_TAKES_ENCODED.csv")
```

Generate Features

Strike Zone

Turn Strikes and Balls from categorical to numerical feature

In [3]:
```python
df = df.with_columns((col("PITCHCALL") == "StrikeCalled").cast(Int64).alias("IS_STRIKE"))
```

Strike Zone Features

Determine if in zone

In [4]:
```python
df = df.with_columns(
    ((col("BOT_ZONE") <= col("PLATELOCHEIGHT"))
     &
     (col("PLATELOCHEIGHT") <= col("TOP_ZONE"))).cast(Int64).alias("IN_ZONE"))
```

Determine how close to the edge of the zone the ball was

In [5]:
```python
df = df.with_columns(
    min_horizontal(
        (col("PLATELOCHEIGHT") - col("BOT_ZONE")).abs(),
        (col("PLATELOCHEIGHT") - col("TOP_ZONE")).abs()
    ).alias("NEAR_VERT_EDGE")
)
```

In [6]:
```python
PLATE_CENTER_WIDTH_FT = 0.708
```

In [7]:
```python
df = df.with_columns((col("PLATELOCSIDE") - PLATE_CENTER_WIDTH_FT).abs().alias("NEAR_HORZ_EDGE"))
```

In [8]:
```python
df = df.with_columns(PLATE_LOC_SIDE=col("PLATELOCSIDE"))
df = df.with_columns(PLATE_LOC_HEIGHT=col("PLATELOCHEIGHT"))
```

Pitch Movement Features

Approach Angle

In [9]:
```python
df = df.with_columns(VERT_APPROACH=col("VERTAPPRANGLE"))
df = df.with_columns(HORZ_APPROACH=col("HORZAPPRANGLE"))
```

Spead and Break

TODO: Need to test if these features are relevant or just noise

In [10]:
```python
df = df.with_columns(INDUCED_VERT_BREAK=col("INDUCEDVERTBREAK"))
df = df.with_columns(HORZ_BREAK=col("HORZBREAK"))
df = df.with_columns(REL_SPEED=col("RELSPEED"))
```

Final Features

In [11]:
```python
features = [
    "PLATE_LOC_HEIGHT",
    "PLATE_LOC_SIDE",
    "TOP_ZONE",
    "BOT_ZONE",
    "IN_ZONE",
    "NEAR_VERT_EDGE",
    "NEAR_HORZ_EDGE",
    "BALLS",
    "STRIKES",
    "VERT_APPROACH",
    "HORZ_APPROACH",
    "INDUCED_VERT_BREAK",
    "HORZ_BREAK",
    "REL_SPEED"
]
```

Remove nulls from critical features

In [12]:
```python
df = df.drop_nulls(subset=features + ["IS_STRIKE"])
```

Train Model

In [13]:
```python
print(f"Rows: {len(df)}")
print(f"Strike rate: {df.select('IS_STRIKE').mean().item():.3f}")
print(f"Features: {len(features)}")
print(f"Years: {df.select('GAME_YEAR').unique().to_series().to_list()}")
print(f"Unique catchers: {df.select('CATCHER_ID').n_unique()}")
```

```
Rows: 1109138
Strike rate: 0.315
Features: 14
Years: [2022, 2023, 2021]
Unique catchers: 164
```

Generate X and Y

In [14]:
```python
def train_model(features_list, df):
    train_df = df.filter(col("GAME_YEAR").is_in([2021, 2022]))
    test_df = df.filter(col("GAME_YEAR") == 2023)
    X_train = train_df.select(features_list).to_numpy()
    y_train = train_df.select("IS_STRIKE").to_numpy().ravel()
    X_test = test_df.select(features_list).to_numpy()
    y_test = test_df.select("IS_STRIKE").to_numpy().ravel()

    model = XGBClassifier(
        objective='binary:logistic',
        max_depth=4,
        learning_rate=0.05,
        n_estimators=300,
        min_child_weight=100,
    )

    model.fit(X_train, y_train)
    train_pred = model.predict_proba(X_train)[:, 1]
    test_pred = model.predict_proba(X_test)[:, 1]

    print("\n=== Model Performance ===")
    print(f"Train Log Loss: {log_loss(y_train, train_pred):.4f}")
    print(f"Test Log Loss:  {log_loss(y_test, test_pred):.4f}")
    print(f"Test AUC:       {roc_auc_score(y_test, test_pred):.4f}")
    print(f"Test Brier:     {brier_score_loss(y_test, test_pred):.4f}")

    # Feature importance
    print("\n=== Feature Importance ===")
    for feat, imp in sorted(zip(features_list, model.feature_importances_), key=lambda x: -x[1]):
        print(f"{feat}: {imp:.4f}")

    return model, train_df, test_df
```

Test 1

In [15]:
```python
train_model(features, df)
```

```
=== Model Performance ===
Train Log Loss: 0.1522
Test Log Loss:  0.1507
Test AUC:       0.9843
Test Brier:     0.0465

=== Feature Importance ===
IN_ZONE: 0.5181
PLATE_LOC_SIDE: 0.1671
NEAR_HORZ_EDGE: 0.1373
NEAR_VERT_EDGE: 0.0830
PLATE_LOC_HEIGHT: 0.0350
STRIKES: 0.0304
BOT_ZONE: 0.0074
BALLS: 0.0049
VERT_APPROACH: 0.0042
HORZ_APPROACH: 0.0041
HORZ_BREAK: 0.0028
TOP_ZONE: 0.0024
REL_SPEED: 0.0020
INDUCED_VERT_BREAK: 0.0012
```

Out[15]:
```
(XGBClassifier(base_score=None, booster=None, callbacks=None,
               colsample_bylevel=None, colsample_bynode=None,
               colsample_bytree=None, device=None, early_stopping_rounds=None,
               enable_categorical=False, eval_metric=None, feature_types=None,
               feature_weights=None, gamma=None, grow_policy=None,
               importance_type=None, interaction_constraints=None,
               learning_rate=0.05, max_bin=None, max_cat_threshold=None,
               max_cat_to_onehot=None, max_delta_step=None, max_depth=4,
               max_leaves=None, min_child_weight=100, missing=nan,
               monotone_constraints=None, multi_strategy=None, n_estimators=300,
               n_jobs=None, num_parallel_tree=None, ...),
 shape: (735_687, 47)
```

AUTOPITCHTYPE --- str	PI_PITCH_TYPE --- str	BALLS --- i64	STRIKES --- i64	…	HORZ_APPROACH --- f64	INDUCED_VERT_BREAK --- f64	HORZ_BREAK --- f64	REL_SPEED --- f64
SL	SL	1	1	…	2.707	-1.387	6.674	86.401
FF	FA	0	0	…	-1.679	18.681	9.419	97.871
SI	FA	1	2	…	-0.139	6.236	9.715	93.631
CH	CH	3	2	…	-0.31	-0.226	-14.682	82.94
SI	SI	1	0	…	0.721	6.267	-16.091	90.162
…	…	…	…	…	…	…	…	…
SL	SL	1	2	…	-3.524	-0.71	-13.25	81.752
CH	CH	0	0	…	2.218	8.658	17.752	87.576
SL	SL	0	0	…	2.778	8.188	4.998	84.42
FF	FA	3	2	…	-1.311	16.889	8.339	97.963
FF	FA	1	0	…	-0.265	16.847	10.249	94.367

,

shape: (373_451, 47)

AUTOPITCHTYPE --- str	PI_PITCH_TYPE --- str	BALLS --- i64	STRIKES --- i64	…	HORZ_APPROACH --- f64	INDUCED_VERT_BREAK --- f64	HORZ_BREAK --- f64	REL_SPEED --- f64
CU	SL	2	2	…	-4.003	-5.905	-10.884	79.542
FF	FA	3	1	…	-1.948	15.191	1.2	92.951
CH	CH	0	1	…	-0.381	-1.009	-12.235	85.014
CH	CH	0	0	…	0.642	6.877	18.228	89.125
FF	FA	1	2	…	0.303	13.353	11.435	91.866
…	…	…	…	…	…	…	…	…
FF	FA	1	1	…	-0.886	12.867	12.432	93.113
FC	FC	0	1	…	-2.622	3.339	-3.747	86.349
FF	FA	1	0	…	1.147	17.214	-9.261	95.786
FF	FA	0	0	…	0.458	17.937	-8.264	93.526
SI	SI	2	1	…	-1.23	9.12	16.596	92.877

)

Test log loss slightly better than train (0.1507 vs 0.1525) — no overfitting AUC of 0.984 — model discriminates very well Brier score 0.047 — probabilities are well-calibrated

Test 2

Simplifying Feature

In [16]:
```python
features_2 = [
"PLATE_LOC_HEIGHT",
"PLATE_LOC_SIDE",
"TOP_ZONE",
"BOT_ZONE",
"NEAR_VERT_EDGE",
"IN_ZONE"
]
```

In [17]:
```python
train_model(features_2, df)
```

```
=== Model Performance ===
Train Log Loss: 0.1573
Test Log Loss:  0.1557
Test AUC:       0.9831
Test Brier:     0.0481

=== Feature Importance ===
IN_ZONE: 0.6264
PLATE_LOC_SIDE: 0.2258
NEAR_VERT_EDGE: 0.0823
PLATE_LOC_HEIGHT: 0.0577
BOT_ZONE: 0.0054
TOP_ZONE: 0.0023
```

Out[17]:
```
(XGBClassifier(base_score=None, booster=None, callbacks=None,
               colsample_bylevel=None, colsample_bynode=None,
               colsample_bytree=None, device=None, early_stopping_rounds=None,
               enable_categorical=False, eval_metric=None, feature_types=None,
               feature_weights=None, gamma=None, grow_policy=None,
               importance_type=None, interaction_constraints=None,
               learning_rate=0.05, max_bin=None, max_cat_threshold=None,
               max_cat_to_onehot=None, max_delta_step=None, max_depth=4,
               max_leaves=None, min_child_weight=100, missing=nan,
               monotone_constraints=None, multi_strategy=None, n_estimators=300,
               n_jobs=None, num_parallel_tree=None, ...),
 shape: (735_687, 47)
```

AUTOPITCHTYPE --- str	PI_PITCH_TYPE --- str	BALLS --- i64	STRIKES --- i64	…	HORZ_APPROACH --- f64	INDUCED_VERT_BREAK --- f64	HORZ_BREAK --- f64	REL_SPEED --- f64
SL	SL	1	1	…	2.707	-1.387	6.674	86.401
FF	FA	0	0	…	-1.679	18.681	9.419	97.871
SI	FA	1	2	…	-0.139	6.236	9.715	93.631
CH	CH	3	2	…	-0.31	-0.226	-14.682	82.94
SI	SI	1	0	…	0.721	6.267	-16.091	90.162
…	…	…	…	…	…	…	…	…
SL	SL	1	2	…	-3.524	-0.71	-13.25	81.752
CH	CH	0	0	…	2.218	8.658	17.752	87.576
SL	SL	0	0	…	2.778	8.188	4.998	84.42
FF	FA	3	2	…	-1.311	16.889	8.339	97.963
FF	FA	1	0	…	-0.265	16.847	10.249	94.367

,

shape: (373_451, 47)

AUTOPITCHTYPE --- str	PI_PITCH_TYPE --- str	BALLS --- i64	STRIKES --- i64	…	HORZ_APPROACH --- f64	INDUCED_VERT_BREAK --- f64	HORZ_BREAK --- f64	REL_SPEED --- f64
CU	SL	2	2	…	-4.003	-5.905	-10.884	79.542
FF	FA	3	1	…	-1.948	15.191	1.2	92.951
CH	CH	0	1	…	-0.381	-1.009	-12.235	85.014
CH	CH	0	0	…	0.642	6.877	18.228	89.125
FF	FA	1	2	…	0.303	13.353	11.435	91.866
…	…	…	…	…	…	…	…	…
FF	FA	1	1	…	-0.886	12.867	12.432	93.113
FC	FC	0	1	…	-2.622	3.339	-3.747	86.349
FF	FA	1	0	…	1.147	17.214	-9.261	95.786
FF	FA	0	0	…	0.458	17.937	-8.264	93.526
SI	SI	2	1	…	-1.23	9.12	16.596	92.877

)

Worse. Log loss went from 0.1507 to 0.1557.

Test 3

Re adding `STRIKES` and `NEAR_HORZ_EDGE`

In [18]:
```python
features_3 = [
    'IN_ZONE',
    'PLATE_LOC_SIDE',
    'NEAR_HORZ_EDGE',
    'NEAR_VERT_EDGE',
    'PLATE_LOC_HEIGHT',
    'STRIKES',
]
```

In [19]:
```python
train_model(features_3, df)
```

```
=== Model Performance ===
Train Log Loss: 0.1557
Test Log Loss:  0.1536
Test AUC:       0.9836
Test Brier:     0.0476

=== Feature Importance ===
IN_ZONE: 0.4885
PLATE_LOC_SIDE: 0.2009
NEAR_HORZ_EDGE: 0.1576
NEAR_VERT_EDGE: 0.0583
STRIKES: 0.0538
PLATE_LOC_HEIGHT: 0.0409
```

Out[19]:
```
(XGBClassifier(base_score=None, booster=None, callbacks=None,
               colsample_bylevel=None, colsample_bynode=None,
               colsample_bytree=None, device=None, early_stopping_rounds=None,
               enable_categorical=False, eval_metric=None, feature_types=None,
               feature_weights=None, gamma=None, grow_policy=None,
               importance_type=None, interaction_constraints=None,
               learning_rate=0.05, max_bin=None, max_cat_threshold=None,
               max_cat_to_onehot=None, max_delta_step=None, max_depth=4,
               max_leaves=None, min_child_weight=100, missing=nan,
               monotone_constraints=None, multi_strategy=None, n_estimators=300,
               n_jobs=None, num_parallel_tree=None, ...),
 shape: (735_687, 47)
```

AUTOPITCHTYPE --- str	PI_PITCH_TYPE --- str	BALLS --- i64	STRIKES --- i64	…	HORZ_APPROACH --- f64	INDUCED_VERT_BREAK --- f64	HORZ_BREAK --- f64	REL_SPEED --- f64
SL	SL	1	1	…	2.707	-1.387	6.674	86.401
FF	FA	0	0	…	-1.679	18.681	9.419	97.871
SI	FA	1	2	…	-0.139	6.236	9.715	93.631
CH	CH	3	2	…	-0.31	-0.226	-14.682	82.94
SI	SI	1	0	…	0.721	6.267	-16.091	90.162
…	…	…	…	…	…	…	…	…
SL	SL	1	2	…	-3.524	-0.71	-13.25	81.752
CH	CH	0	0	…	2.218	8.658	17.752	87.576
SL	SL	0	0	…	2.778	8.188	4.998	84.42
FF	FA	3	2	…	-1.311	16.889	8.339	97.963
FF	FA	1	0	…	-0.265	16.847	10.249	94.367

,

shape: (373_451, 47)

AUTOPITCHTYPE --- str	PI_PITCH_TYPE --- str	BALLS --- i64	STRIKES --- i64	…	HORZ_APPROACH --- f64	INDUCED_VERT_BREAK --- f64	HORZ_BREAK --- f64	REL_SPEED --- f64
CU	SL	2	2	…	-4.003	-5.905	-10.884	79.542
FF	FA	3	1	…	-1.948	15.191	1.2	92.951
CH	CH	0	1	…	-0.381	-1.009	-12.235	85.014
CH	CH	0	0	…	0.642	6.877	18.228	89.125
FF	FA	1	2	…	0.303	13.353	11.435	91.866
…	…	…	…	…	…	…	…	…
FF	FA	1	1	…	-0.886	12.867	12.432	93.113
FC	FC	0	1	…	-2.622	3.339	-3.747	86.349
FF	FA	1	0	…	1.147	17.214	-9.261	95.786
FF	FA	0	0	…	0.458	17.937	-8.264	93.526
SI	SI	2	1	…	-1.23	9.12	16.596	92.877

)

An improvement but still not better than the first test. Log loss went from 0.1507 to 0.1536.

Test 4

Re adding `TOP_ZONE` and `TOP_ZONE`

In [20]:
```python
features_4 = [
    'IN_ZONE',
    'PLATE_LOC_SIDE',
    'PLATE_LOC_HEIGHT',
    'TOP_ZONE',
    'BOT_ZONE',
    'NEAR_HORZ_EDGE',
    'NEAR_VERT_EDGE',
    'STRIKES',
]
```

In [21]:
```python
train_model(features_4, df)
```

```
=== Model Performance ===
Train Log Loss: 0.1550
Test Log Loss:  0.1533
Test AUC:       0.9836
Test Brier:     0.0474

=== Feature Importance ===
IN_ZONE: 0.4992
PLATE_LOC_SIDE: 0.1839
NEAR_HORZ_EDGE: 0.1505
NEAR_VERT_EDGE: 0.0702
STRIKES: 0.0526
PLATE_LOC_HEIGHT: 0.0368
BOT_ZONE: 0.0049
TOP_ZONE: 0.0019
```

Out[21]:
```
(XGBClassifier(base_score=None, booster=None, callbacks=None,
               colsample_bylevel=None, colsample_bynode=None,
               colsample_bytree=None, device=None, early_stopping_rounds=None,
               enable_categorical=False, eval_metric=None, feature_types=None,
               feature_weights=None, gamma=None, grow_policy=None,
               importance_type=None, interaction_constraints=None,
               learning_rate=0.05, max_bin=None, max_cat_threshold=None,
               max_cat_to_onehot=None, max_delta_step=None, max_depth=4,
               max_leaves=None, min_child_weight=100, missing=nan,
               monotone_constraints=None, multi_strategy=None, n_estimators=300,
               n_jobs=None, num_parallel_tree=None, ...),
 shape: (735_687, 47)
```

AUTOPITCHTYPE --- str	PI_PITCH_TYPE --- str	BALLS --- i64	STRIKES --- i64	…	HORZ_APPROACH --- f64	INDUCED_VERT_BREAK --- f64	HORZ_BREAK --- f64	REL_SPEED --- f64
SL	SL	1	1	…	2.707	-1.387	6.674	86.401
FF	FA	0	0	…	-1.679	18.681	9.419	97.871
SI	FA	1	2	…	-0.139	6.236	9.715	93.631
CH	CH	3	2	…	-0.31	-0.226	-14.682	82.94
SI	SI	1	0	…	0.721	6.267	-16.091	90.162
…	…	…	…	…	…	…	…	…
SL	SL	1	2	…	-3.524	-0.71	-13.25	81.752
CH	CH	0	0	…	2.218	8.658	17.752	87.576
SL	SL	0	0	…	2.778	8.188	4.998	84.42
FF	FA	3	2	…	-1.311	16.889	8.339	97.963
FF	FA	1	0	…	-0.265	16.847	10.249	94.367

,

shape: (373_451, 47)

AUTOPITCHTYPE --- str	PI_PITCH_TYPE --- str	BALLS --- i64	STRIKES --- i64	…	HORZ_APPROACH --- f64	INDUCED_VERT_BREAK --- f64	HORZ_BREAK --- f64	REL_SPEED --- f64
CU	SL	2	2	…	-4.003	-5.905	-10.884	79.542
FF	FA	3	1	…	-1.948	15.191	1.2	92.951
CH	CH	0	1	…	-0.381	-1.009	-12.235	85.014
CH	CH	0	0	…	0.642	6.877	18.228	89.125
FF	FA	1	2	…	0.303	13.353	11.435	91.866
…	…	…	…	…	…	…	…	…
FF	FA	1	1	…	-0.886	12.867	12.432	93.113
FC	FC	0	1	…	-2.622	3.339	-3.747	86.349
FF	FA	1	0	…	1.147	17.214	-9.261	95.786
FF	FA	0	0	…	0.458	17.937	-8.264	93.526
SI	SI	2	1	…	-1.23	9.12	16.596	92.877

)

Still not better. Log loss went from 0.1507 to 0.1533.

Test 5

add `VERT_APPROACH`, `HORZ_APPROACH`, `BALLS`

In [22]:
```python
features_5 = [
    'PLATE_LOC_HEIGHT',
    'PLATE_LOC_SIDE',
    'TOP_ZONE',
    'BOT_ZONE',
    'IN_ZONE',
    'NEAR_VERT_EDGE',
    'NEAR_HORZ_EDGE',
    'BALLS',
    'STRIKES',
    'VERT_APPROACH',
    'HORZ_APPROACH',
]
```

In [23]:
```python
model, train_df, test_df = train_model(features_5, df)
```

```
=== Model Performance ===
Train Log Loss: 0.1525
Test Log Loss:  0.1509
Test AUC:       0.9842
Test Brier:     0.0466

=== Feature Importance ===
IN_ZONE: 0.5165
PLATE_LOC_SIDE: 0.1760
NEAR_HORZ_EDGE: 0.1414
NEAR_VERT_EDGE: 0.0786
PLATE_LOC_HEIGHT: 0.0371
STRIKES: 0.0291
BOT_ZONE: 0.0069
BALLS: 0.0049
VERT_APPROACH: 0.0036
HORZ_APPROACH: 0.0036
TOP_ZONE: 0.0023
```

Improvement from the first test but almost similar. `INDUCED_VERT_BREAK`, `HORZ_BREAK`, and `REL_SPEED` are the features that don't matter

Verdict

Sticking with the 11 features(test 5). Even though the feature importance is low on some, the combonation of the features leads to a lower

Check Model Stability

In [24]:
```python
def validate_year_to_year_stability(model, train_df, test_df, features):
    df = concat([train_df, test_df])
    X = df.select(features).to_numpy()
    df = df.with_columns(Series("CS_PROB", model.predict_proba(X)[:, 1]))
    catcher_year = df.group_by(["CATCHER_ID", "GAME_YEAR"]).agg([
        pl_len().alias("opportunities"),
        col("IS_STRIKE").sum().alias("actual_cs"),
        col("CS_PROB").sum().alias("expected_cs"),
    ]).with_columns(
        ((col("actual_cs") - col("expected_cs")) / col("opportunities") * 100)
        .alias("cs_added_per_100")
    )

    catcher_year = catcher_year.filter(col("opportunities") >= 500)

    df_2022 = catcher_year.filter(col("GAME_YEAR") == 2022).select(["CATCHER_ID", "cs_added_per_100"])
    df_2023 = catcher_year.filter(col("GAME_YEAR") == 2023).select(["CATCHER_ID", "cs_added_per_100"])

    merged = df_2022.join(df_2023, on="CATCHER_ID", suffix="_2023")

    corr = corrcoef(
        merged.select("cs_added_per_100").to_numpy().ravel(),
        merged.select("cs_added_per_100_2023").to_numpy().ravel()
    )[0, 1]

    print(f"\n=== Year-to-Year Stability ===")
    print(f"2022 vs 2023: r = {corr:.3f} (n = {len(merged)} catchers)")
```

In [25]:
```python
validate_year_to_year_stability(model, train_df, test_df, features_5)
```

```
=== Year-to-Year Stability ===
2022 vs 2023: r = 0.586 (n = 65 catchers)
```

0.586 is in the range.

In [29]:
```python
model_path = Path("framing_model.pkl")
model_data = {
    'model': model,
    'feature_columns': features_5,
}
```

In [30]:
```python
with model_path.open('wb') as f:
    pickle.dump(model_data, f)
```

In [ ]: