Response Engineering

Spec

The idea is to create a model that can interpolate for every row of data.

Given the hypothetical scenario that every pitch resulted in a swing, the model predicts the whiff%. We call this xWhiff%

Given the hypothetical scenario that the pitch resulted in contact, the model predicts the characteristics of the hit(ExitSpeed, LaunchAngle, HitSpinRate, Direction), we can use those to calculate an expected hit quality score, or xHQ

We'll train two models with the strike/hit data to see if any of these metrics are viable. If both metrics are viable, we can combine them to create a new hybrid pitch quality score, xhPQ

Packages:

Read the csv:

```r
# Path-to-data, /data/datasets.csv
ucla <- read_csv("./data/UCLA2023-2024.csv")
```

```
Rows: 31775 Columns: 198
-- Column specification --------------------------------------------------------
Delimiter: ","
chr   (40): Date, Pitcher, PitcherThrows, PitcherTeam, Batter, BatterSide, B...
dbl  (148): PitchNo, PAofInning, PitchofPA, PitcherId, BatterId, Inning, Out...
lgl    (4): MeasuredDuration, PitchLastMeasuredX, PitchLastMeasuredY, PitchL...
dttm   (2): LocalDateTime, UTCDateTime
time   (4): Time, Tilt, UTCTime, SpinAxis3dTilt
```

```
i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
penn <- read_csv("./data/PennState2024.csv")
```

```
Rows: 8539 Columns: 198
-- Column specification --------------------------------------------------------
Delimiter: ","
chr   (38): Pitcher, PitcherThrows, PitcherTeam, Batter, BatterSide, BatterT...
dbl  (127): PitchNo, PAofInning, PitchofPA, PitcherId, BatterId, Inning, Out...
lgl   (25): Notes, MeasuredDuration, PitchLastMeasuredX, PitchLastMeasuredY,...
dttm   (2): LocalDateTime, UTCDateTime
date   (2): Date, UTCDate
time   (4): Time, Tilt, UTCTime, SpinAxis3dTilt

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
purdue <- read_csv("./data/Purdue2024.csv")
```

```
Rows: 11288 Columns: 198
-- Column specification --------------------------------------------------------
Delimiter: ","
chr   (38): Pitcher, PitcherThrows, PitcherTeam, Batter, BatterSide, BatterT...
dbl  (117): PitchNo, PAofInning, PitchofPA, PitcherId, BatterId, Inning, Out...
lgl   (36): Notes, MeasuredDuration, PitchLastMeasuredX, PitchLastMeasuredY,...
dttm   (2): LocalDateTime, UTCDateTime
date   (2): Date, UTCDate
time   (3): Time, Tilt, UTCTime

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
michigan <- read_csv("./data/Michigan2024.csv")
```

```
Rows: 10202 Columns: 198
-- Column specification --------------------------------------------------------
Delimiter: ","
chr   (39): Pitcher, PitcherThrows, PitcherTeam, Batter, BatterSide, BatterT...
```

```
dbl  (117): PitchNo, PAofInning, PitchofPA, PitcherId, BatterId, Inning, Out...
lgl   (35): MeasuredDuration, PitchLastMeasuredX, PitchLastMeasuredY, PitchL...
dttm   (2): LocalDateTime, UTCDateTime
date   (2): Date, UTCDate
time   (3): Time, Tilt, UTCTime

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Binding Rows from the CSV files

```r
# Need to convert from character to date object
ucla <- ucla %>%
  mutate(Date = as.Date(Date)) %>%
  mutate(UTCDate = as.Date(UTCDate)) %>%
  mutate(AwayTeamForeignID = as.character(AwayTeamForeignID))

# The data set that we will be mutating
main <- bind_rows(ucla, penn, michigan, purdue)
```

Clean the Data

Drop NA's for the variables we will use

```r
filtered_vars <- c(
  "Pitcher",
  "PitcherId",
  "TaggedPitchType",
  "RelSpeed",           # Speed at release
  "ZoneSpeed",          # Speed at the plate
  "EffectiveVelo",      # Perceived pitch speed
  "VertBreak",          # Full vertical break
  "InducedVertBreak",   # Break excluding gravity
  "HorzBreak",          # Horizontal movement
  "SpinRate",           # Raw spin
  "SpinAxis",           # 0-360 spin axis
  "RelHeight",          # Release height
  "RelSide",            # Horizontal release side
  "Extension",          # Distance toward plate
  "VertApprAngle",      # Vertical approach angle
```

```r
  "HorzApprAngle",       # Horizontal approach angle
  "VertRelAngle",        # Vertical release angle
  "HorzRelAngle",        # Horizontal release angle
  "Tilt",                # Spin tilt
  "PlateLocHeight",      # Raw vertical location
  "PlateLocSide",        # Raw horizontal location
  "ZoneTime",            # Time to reach plate
  "SpeedDrop",            # Velo loss from release to plate
  "Balls",
  "Strikes",
  "Outs",
  "BatterSide"
)

main %>%
  select(all_of(filtered_vars)) %>%
  summarise(across(everything(), ~sum(is.na(.)))) %>%
  pivot_longer(everything(), names_to = "column", values_to = "na_count") %>%
  arrange(desc(na_count))
```

```
# A tibble: 27 x 2
   column           na_count
   <chr>               <int>
 1 EffectiveVelo         669
 2 SpeedDrop             669
 3 SpinRate              550
 4 VertBreak             545
 5 InducedVertBreak      545
 6 HorzBreak             545
 7 SpinAxis              545
 8 Tilt                  545
 9 ZoneTime              503
10 Extension             493
# i 17 more rows
```

```r
main <- main %>%
  drop_na(all_of(filtered_vars))

main <- main %>%
  arrange(UTCDateTime, PitchNo)
```

Retrieve Colnames

```r
colnames <- as.data.frame(cbind(indx = 1:length(colnames(main)), colnames = colnames(main)))
colnames
```

```
   indx                                colnames
1     1                                 PitchNo
2     2                                    Date
3     3                                    Time
4     4                             PAofInning
5     5                               PitchofPA
6     6                                 Pitcher
7     7                               PitcherId
8     8                           PitcherThrows
9     9                             PitcherTeam
10   10                                  Batter
11   11                                BatterId
12   12                              BatterSide
13   13                              BatterTeam
14   14                              PitcherSet
15   15                                  Inning
16   16                              Top_Bottom
17   17                                    Outs
18   18                                   Balls
19   19                                 Strikes
20   20                         TaggedPitchType
21   21                           AutoPitchType
22   22                               PitchCall
23   23                                   KorBB
24   24                           TaggedHitType
25   25                              PlayResult
26   26                              OutsOnPlay
27   27                              RunsScored
28   28                                   Notes
29   29                                RelSpeed
30   30                            VertRelAngle
31   31                            HorzRelAngle
32   32                                SpinRate
33   33                                SpinAxis
34   34                                    Tilt
35   35                               RelHeight
36   36                                 RelSide
37   37                               Extension
38   38                               VertBreak
```

39	39	InducedVertBreak
40	40	HorzBreak
41	41	PlateLocHeight
42	42	PlateLocSide
43	43	ZoneSpeed
44	44	VertApprAngle
45	45	HorzApprAngle
46	46	ZoneTime
47	47	ExitSpeed
48	48	Angle
49	49	Direction
50	50	HitSpinRate
51	51	PositionAt110X
52	52	PositionAt110Y
53	53	PositionAt110Z
54	54	Distance
55	55	LastTrackedDistance
56	56	Bearing
57	57	HangTime
58	58	pfxx
59	59	pfxz
60	60	x0
61	61	y0
62	62	z0
63	63	vx0
64	64	vy0
65	65	vz0
66	66	ax0
67	67	ay0
68	68	az0
69	69	HomeTeam
70	70	AwayTeam
71	71	Stadium
72	72	Level
73	73	League
74	74	GameID
75	75	PitchUID
76	76	EffectiveVelo
77	77	MaxHeight
78	78	MeasuredDuration
79	79	SpeedDrop
80	80	PitchLastMeasuredX
81	81	PitchLastMeasuredY

82	82	PitchLastMeasuredZ
83	83	ContactPositionX
84	84	ContactPositionY
85	85	ContactPositionZ
86	86	GameUID
87	87	UTCDate
88	88	UTCTime
89	89	LocalDateTime
90	90	UTCDateTime
91	91	AutoHitType
92	92	System
93	93	HomeTeamForeignID
94	94	AwayTeamForeignID
95	95	GameForeignID
96	96	Catcher
97	97	CatcherId
98	98	CatcherThrows
99	99	CatcherTeam
100	100	PlayID
101	101	PitchTrajectoryXc0
102	102	PitchTrajectoryXc1
103	103	PitchTrajectoryXc2
104	104	PitchTrajectoryYc0
105	105	PitchTrajectoryYc1
106	106	PitchTrajectoryYc2
107	107	PitchTrajectoryZc0
108	108	PitchTrajectoryZc1
109	109	PitchTrajectoryZc2
110	110	HitSpinAxis
111	111	HitTrajectoryXc0
112	112	HitTrajectoryXc1
113	113	HitTrajectoryXc2
114	114	HitTrajectoryXc3
115	115	HitTrajectoryXc4
116	116	HitTrajectoryXc5
117	117	HitTrajectoryXc6
118	118	HitTrajectoryXc7
119	119	HitTrajectoryXc8
120	120	HitTrajectoryYc0
121	121	HitTrajectoryYc1
122	122	HitTrajectoryYc2
123	123	HitTrajectoryYc3
124	124	HitTrajectoryYc4

125	125	HitTrajectoryYc5
126	126	HitTrajectoryYc6
127	127	HitTrajectoryYc7
128	128	HitTrajectoryYc8
129	129	HitTrajectoryZc0
130	130	HitTrajectoryZc1
131	131	HitTrajectoryZc2
132	132	HitTrajectoryZc3
133	133	HitTrajectoryZc4
134	134	HitTrajectoryZc5
135	135	HitTrajectoryZc6
136	136	HitTrajectoryZc7
137	137	HitTrajectoryZc8
138	138	ThrowSpeed
139	139	PopTime
140	140	ExchangeTime
141	141	TimeToBase
142	142	CatchPositionX
143	143	CatchPositionY
144	144	CatchPositionZ
145	145	ThrowPositionX
146	146	ThrowPositionY
147	147	ThrowPositionZ
148	148	BasePositionX
149	149	BasePositionY
150	150	BasePositionZ
151	151	ThrowTrajectoryXc0
152	152	ThrowTrajectoryXc1
153	153	ThrowTrajectoryXc2
154	154	ThrowTrajectoryYc0
155	155	ThrowTrajectoryYc1
156	156	ThrowTrajectoryYc2
157	157	ThrowTrajectoryZc0
158	158	ThrowTrajectoryZc1
159	159	ThrowTrajectoryZc2
160	160	PitchReleaseConfidence
161	161	PitchLocationConfidence
162	162	PitchMovementConfidence
163	163	HitLaunchConfidence
164	164	HitLandingConfidence
165	165	CatcherThrowCatchConfidence
166	166	CatcherThrowReleaseConfidence
167	167	CatcherThrowLocationConfidence

```
168  168                          SpinAxis3dTransverseAngle
169  169                        SpinAxis3dLongitudinalAngle
170  170                           SpinAxis3dActiveSpinRate
171  171                           SpinAxis3dSpinEfficiency
172  172                                     SpinAxis3dTilt
173  173                                  SpinAxis3dVectorX
174  174                                  SpinAxis3dVectorY
175  175                                  SpinAxis3dVectorZ
176  176                SpinAxis3dSeamOrientationRotationX
177  177                SpinAxis3dSeamOrientationRotationY
178  178                SpinAxis3dSeamOrientationRotationZ
179  179 SpinAxis3dSeamOrientationBallAngleHorizontalAmb1
180  180   SpinAxis3dSeamOrientationBallAngleVerticalAmb1
181  181                SpinAxis3dSeamOrientationBallXAmb1
182  182                SpinAxis3dSeamOrientationBallYAmb1
183  183                SpinAxis3dSeamOrientationBallZAmb1
184  184 SpinAxis3dSeamOrientationBallAngleHorizontalAmb2
185  185   SpinAxis3dSeamOrientationBallAngleVerticalAmb2
186  186                SpinAxis3dSeamOrientationBallXAmb2
187  187                SpinAxis3dSeamOrientationBallYAmb2
188  188                SpinAxis3dSeamOrientationBallZAmb2
189  189 SpinAxis3dSeamOrientationBallAngleHorizontalAmb3
190  190   SpinAxis3dSeamOrientationBallAngleVerticalAmb3
191  191                SpinAxis3dSeamOrientationBallXAmb3
192  192                SpinAxis3dSeamOrientationBallYAmb3
193  193                SpinAxis3dSeamOrientationBallZAmb3
194  194 SpinAxis3dSeamOrientationBallAngleHorizontalAmb4
195  195   SpinAxis3dSeamOrientationBallAngleVerticalAmb4
196  196                SpinAxis3dSeamOrientationBallXAmb4
197  197                SpinAxis3dSeamOrientationBallYAmb4
198  198                SpinAxis3dSeamOrientationBallZAmb4
199  199                                         Top.Bottom
```

Filter for only fastballs:

```
main %>%
  group_by(TaggedPitchType) %>%
  summarise(count = n())
```

```
# A tibble: 13 x 2
   TaggedPitchType  count
   <chr>            <int>
```

```
 1 ChangeUp          6843
 2 Curveball         4332
 3 Cutter            1648
 4 Fastball         26353
 5 FourSeamFastBall  1725
 6 Knuckleball          5
 7 OneSeamFastBall     19
 8 Other               66
 9 Sinker            5263
10 Slider           13175
11 Splitter           167
12 TwoSeamFastBall     72
13 Undefined         1359
```

```
main <- main %>%
  filter(TaggedPitchType == "Fastball" | TaggedPitchType == "FourSeamFastBall" | TaggedPitchT
  select(1:50)
cat("Number of rows in fastball:", nrow(main), "\n")
```

```
Number of rows in fastball: 28097
```

Filter out the swinging data

```
main %>%
  group_by(PitchCall) %>%
  summarise(count = n())
```

```
# A tibble: 10 x 2
   PitchCall            count
   <chr>                <int>
 1 BallCalled           10025
 2 BallIntentional          4
 3 BallinDirt              51
 4 FoulBall              2316
 5 FoulBallFieldable       21
 6 FoulBallNotFieldable  3050
 7 HitByPitch             231
 8 InPlay                5387
 9 StrikeCalled          4922
10 StrikeSwinging        2090
```

```
main_swing <- main %>%
  filter(PitchCall %in% c("StrikeCalled", "StrikeSwinging", "InPlay", "FoulBall", "FoulBallF

main_swing %>%
  group_by(PitchCall) %>%
  summarise(count = n())
```

```
# A tibble: 6 x 2
  PitchCall            count
  <chr>                <int>
1 FoulBall              2316
2 FoulBallFieldable       21
3 FoulBallNotFieldable  3050
4 InPlay                5387
5 StrikeCalled          4922
6 StrikeSwinging        2090
```

Normalize and Feature Engineering

Make the response variable: Whiff(1) Not Whiff(0)

```
main_swing <- main_swing %>%
  mutate(is_whiff = ifelse(PitchCall=="StrikeSwinging", 1, 0)) %>%
  relocate(is_whiff, .after=PitchCall)
```

Spin Axis Normalization

```
summary(main$SpinAxis)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  30.04  178.61  206.59  197.84  221.94  356.19
```

```
cat("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ \n")
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```r
# we need to normalize spin axis because 359 degrees and 1 degrees are very close

# Convert SpinAxis to radians
# Idea: convert the spin axis to a x and y unit circle direction
main_swing <- main_swing %>%
  mutate(
    SpinAxis_rad = SpinAxis * pi / 180,
    SpinAxis_sin = sin(SpinAxis_rad),
    SpinAxis_cos = cos(SpinAxis_rad),
  ) %>%
  relocate(SpinAxis_rad, SpinAxis_sin, SpinAxis_cos, .after=SpinAxis)

summary(main_swing %>% select(SpinAxis_rad, SpinAxis_sin, SpinAxis_cos))
```

```
  SpinAxis_rad     SpinAxis_sin       SpinAxis_cos
 Min.   :0.5243   Min.   :-1.00000   Min.   :-1.0000
 1st Qu.:3.1126   1st Qu.:-0.66801   1st Qu.:-0.9203
 Median :3.6096   Median :-0.45111   Median :-0.8233
 Mean   :3.4537   Mean   :-0.27441   Mean   :-0.7748
 3rd Qu.:3.8732   3rd Qu.: 0.02895   3rd Qu.:-0.6873
 Max.   :5.6815   Max.   : 0.99987   Max.   : 0.8657
```

Batter Side : 1 is Right, 0 is Left

```r
main_swing <- main_swing %>%
  mutate(
    BatterSide = ifelse(BatterSide == "Right", 1, 0)
  ) %>%
  mutate(BatterSide = factor(BatterSide))

summary(main_swing %>% select(BatterSide))
```

```
 BatterSide
 0: 7263
 1:10523
```

Count

```r
main_swing <- main_swing %>%
  mutate(
    Count = factor(paste0(Balls, "-", Strikes))
  ) %>%
  relocate(Count, .after=Strikes)
```

Outs

```r
main_swing <- main_swing %>%
  mutate(Outs = factor(Outs))
```

Subset the Dataset:

```r
useful_vars <- c(
  "is_whiff",
  "Pitcher",
  "PitcherId",
  "TaggedPitchType",
  "Count",
  "Outs",
  "BatterSide",
  "RelSpeed",           # Speed at release
  "ZoneSpeed",          # Speed at the plate
  "EffectiveVelo",      # Perceived pitch speed
  "VertBreak",          # Full vertical break
  "InducedVertBreak",   # Break excluding gravity
  "HorzBreak",          # Horizontal movement
  "SpinRate",           # Raw spin
  "SpinAxis",           # 0-360 spin axis
  "SpinAxis_sin",
  "SpinAxis_cos",
  "RelHeight",          # Release height
  "RelSide",            # Horizontal release side
  "Extension",          # Distance toward plate
  "VertApprAngle",      # Vertical approach angle
  "HorzApprAngle",      # Horizontal approach angle
  "VertRelAngle",       # Vertical release angle
  "HorzRelAngle",       # Horizontal release angle
  "PlateLocHeight",     # Raw vertical location
  "PlateLocSide",       # Raw horizontal location
  "ZoneTime",           # Time to reach plate
```

```r
  "SpeedDrop"            # Velo loss from release to plate
)

main_sw_filter <- main_swing %>%
  select(any_of(useful_vars))
```

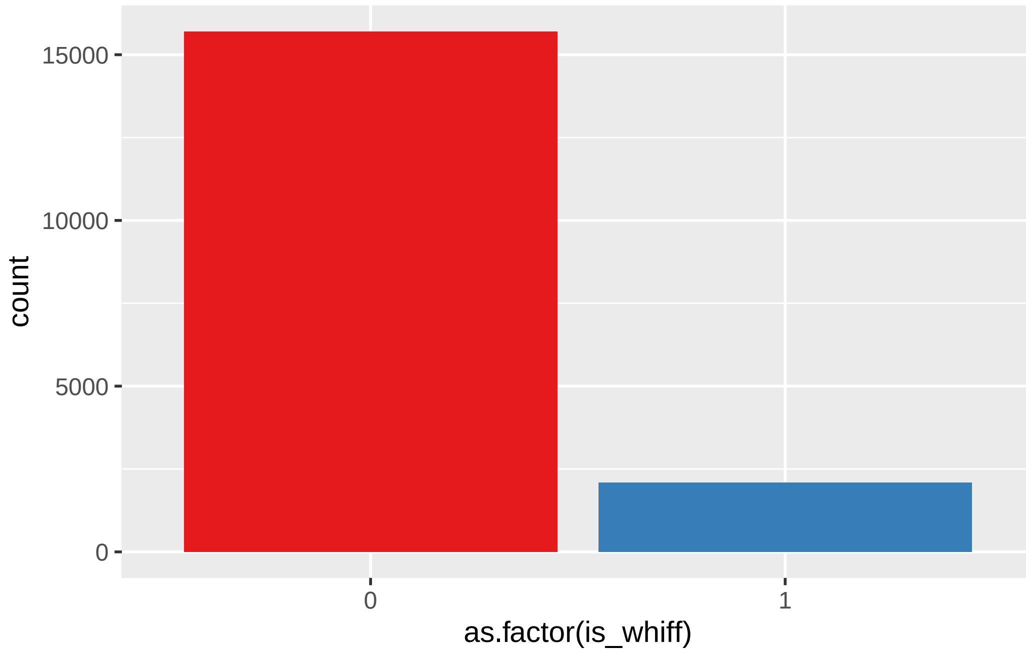
```r
head(main_sw_filter)
```

```
# A tibble: 6 x 26
  is_whiff Pitcher     PitcherId TaggedPitchType Count Outs  BatterSide RelSpeed
     <dbl> <chr>           <dbl> <chr>           <fct> <fct> <fct>         <dbl>
1        0 Riedel, Ca~    1.00e9 Fastball        1-2   0     1              89.3
2        0 Riedel, Ca~    1.00e9 Fastball        1-2   0     1              89.3
3        1 Riedel, Ca~    1.00e9 Fastball        3-2   0     1              88.6
4        1 Riedel, Ca~    1.00e9 Fastball        3-2   0     1              88.6
5        0 Riedel, Ca~    1.00e9 Fastball        0-0   1     0              89.6
6        0 Riedel, Ca~    1.00e9 Fastball        0-0   1     0              89.6
# i 18 more variables: ZoneSpeed <dbl>, VertBreak <dbl>,
#   InducedVertBreak <dbl>, HorzBreak <dbl>, SpinRate <dbl>, SpinAxis <dbl>,
#   SpinAxis_sin <dbl>, SpinAxis_cos <dbl>, RelHeight <dbl>, RelSide <dbl>,
#   Extension <dbl>, VertApprAngle <dbl>, HorzApprAngle <dbl>,
#   VertRelAngle <dbl>, HorzRelAngle <dbl>, PlateLocHeight <dbl>,
#   PlateLocSide <dbl>, ZoneTime <dbl>
```

EDA:

Looking at data set balance

```r
ggplot(main_swing, aes(x=as.factor(is_whiff), fill=as.factor(is_whiff) )) +
  geom_bar( ) +
  scale_fill_brewer(palette = "Set1") +
  theme(legend.position="none")
```




```
main_sw_filter %>%
  group_by(is_whiff) %>%
  summarise(count = n())
```

```
# A tibble: 2 x 2
  is_whiff count
     <dbl> <int>
1        0 15696
2        1  2090
```

So we see that approx ~1/8 of the swung at pitches were whiffs, which means the dataset might be imbalanced

Data Preprocessing Pipeline:

```
model_df_vars <- c(
  "is_whiff",
  "RelSpeed",           # Speed at release
  "VertBreak",          # Full vertical break
```

```r
  "HorzBreak",          # Horizontal movement
  "SpinRate",           # Raw spin RPM
  "SpinAxis_sin",
  "SpinAxis_cos",
  "RelHeight",          # Release height
  "RelSide",            # Horizontal release side
  "Extension",          # Distance toward plate
  "VertApprAngle",      # Vertical approach angle
  "HorzApprAngle",      # Horizontal approach angle
  "VertRelAngle",       # Vertical release angle
  "HorzRelAngle",       # Horizontal release angle
  "Count", # Count as a factor string
  "Outs", # 0-2
  "BatterSide" # 1 - Right, 0 - Left
)

model_df <- main_sw_filter %>%
  select(all_of(model_df_vars))

head(model_df)
```

```
# A tibble: 6 x 17
  is_whiff RelSpeed VertBreak HorzBreak SpinRate SpinAxis_sin SpinAxis_cos
     <dbl>    <dbl>     <dbl>     <dbl>    <dbl>        <dbl>        <dbl>
1        0     89.3     -15.1     -12.4    2311.        0.557       -0.830
2        0     89.3     -15.1     -12.4    2311.        0.557       -0.830
3        1     88.6     -13.7     -14.4    2302.        0.583       -0.813
4        1     88.6     -13.7     -14.4    2302.        0.583       -0.813
5        0     89.6     -13.0     -15.8    2292.        0.618       -0.786
6        0     89.6     -13.0     -15.8    2292.        0.618       -0.786
# i 10 more variables: RelHeight <dbl>, RelSide <dbl>, Extension <dbl>,
#   VertApprAngle <dbl>, HorzApprAngle <dbl>, VertRelAngle <dbl>,
#   HorzRelAngle <dbl>, Count <fct>, Outs <fct>, BatterSide <fct>
```

Scale all numeric variables and apply one-hot encoding to the categorical ones:

```r
rec <- recipe(is_whiff ~ ., data = model_df) %>%
  step_normalize(all_numeric_predictors()) %>%     # Normalize (center & scale)
  step_dummy(all_nominal_predictors())             # One-hot encode all factors

rec_prepped <- prep(rec)
```

```r
model_df_transformed <- bake(rec_prepped, new_data = NULL)
```

```r
head(model_df_transformed)
```

```
# A tibble: 6 x 28
  RelSpeed VertBreak HorzBreak SpinRate SpinAxis_sin SpinAxis_cos RelHeight
     <dbl>     <dbl>     <dbl>    <dbl>        <dbl>        <dbl>     <dbl>
1  -0.0721     0.614     -1.63    0.710         1.56      -0.277     -0.624
2  -0.0721     0.614     -1.63    0.710         1.56      -0.277     -0.624
3  -0.339      0.859     -1.82    0.660         1.61      -0.190     -0.514
4  -0.339      0.859     -1.82    0.660         1.61      -0.190     -0.514
5   0.0166     0.985     -1.94    0.602         1.67      -0.0556    -0.475
6   0.0166     0.985     -1.94    0.602         1.67      -0.0556    -0.475
# i 21 more variables: RelSide <dbl>, Extension <dbl>, VertApprAngle <dbl>,
#   HorzApprAngle <dbl>, VertRelAngle <dbl>, HorzRelAngle <dbl>,
#   is_whiff <dbl>, Count_X0.1 <dbl>, Count_X0.2 <dbl>, Count_X1.0 <dbl>,
#   Count_X1.1 <dbl>, Count_X1.2 <dbl>, Count_X2.0 <dbl>, Count_X2.1 <dbl>,
#   Count_X2.2 <dbl>, Count_X3.0 <dbl>, Count_X3.1 <dbl>, Count_X3.2 <dbl>,
#   Outs_X1 <dbl>, Outs_X2 <dbl>, BatterSide_X1 <dbl>
```

```r
# Sanitty check for NA values
model_df_transformed %>%
  summarise(across(everything(), ~sum(is.na(.)))) %>%
  pivot_longer(everything(), names_to = "column", values_to = "na_count") %>%
  arrange(desc(na_count))
```

```
# A tibble: 28 x 2
   column        na_count
   <chr>            <int>
 1 RelSpeed             0
 2 VertBreak            0
 3 HorzBreak            0
 4 SpinRate             0
 5 SpinAxis_sin         0
 6 SpinAxis_cos         0
 7 RelHeight            0
 8 RelSide              0
 9 Extension            0
10 VertApprAngle        0
# i 18 more rows
```

Splitting Training and Test Data:

```
set.seed(42)
split_index <- sample(seq_len(nrow(main_swing)), size = 0.8 * nrow(main_swing))

# Create training and testing datasets
train_data <- model_df_transformed[split_index, ]
test_data <- model_df_transformed[-split_index, ]

dim(train_data)
```

```
[1] 14228    28
```

```
dim(test_data)
```

```
[1] 3558   28
```

Train the Models

Normal Logistic Regression

```
m1 <- glm(is_whiff ~ ., data = train_data, family = binomial())
summary(m1)
```

```
Call:
glm(formula = is_whiff ~ ., family = binomial(), data = train_data)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)   -2.57969    0.08146 -31.667  < 2e-16 ***
RelSpeed      -0.09572    0.04684  -2.044 0.040992 *
VertBreak     -3.23213    1.23592  -2.615 0.008918 **
HorzBreak      1.17529    2.84767   0.413 0.679813
SpinRate       0.05897    0.03058   1.928 0.053839 .
SpinAxis_sin   0.24619    0.16449   1.497 0.134468
SpinAxis_cos   0.18015    0.05381   3.348 0.000814 ***
```

```
RelHeight       0.25784    0.04034   6.391 1.64e-10 ***
RelSide         0.19786    0.09577   2.066 0.038826 *
Extension       0.08521    0.04215   2.022 0.043227 *
VertApprAngle   3.51576    1.06730   3.294 0.000988 ***
HorzApprAngle -0.62173    1.69140  -0.368 0.713186
VertRelAngle  -3.46018    1.33351  -2.595 0.009465 **
HorzRelAngle   1.59256    3.80557   0.418 0.675595
Count_X0.1      0.36487    0.10610   3.439 0.000584 ***
Count_X0.2      0.42088    0.13363   3.150 0.001635 **
Count_X1.0      0.08776    0.10751   0.816 0.414331
Count_X1.1      0.42126    0.10648   3.956 7.61e-05 ***
Count_X1.2      0.61050    0.11077   5.511 3.56e-08 ***
Count_X2.0      0.27213    0.13107   2.076 0.037873 *
Count_X2.1      0.57573    0.11134   5.171 2.33e-07 ***
Count_X2.2      0.54949    0.10724   5.124 2.99e-07 ***
Count_X3.0     -1.84700    0.45559  -4.054 5.03e-05 ***
Count_X3.1      0.56127    0.13433   4.178 2.94e-05 ***
Count_X3.2      0.45639    0.11296   4.040 5.34e-05 ***
Outs_X1         0.09692    0.06556   1.478 0.139347
Outs_X2         0.12122    0.06664   1.819 0.068915 .
BatterSide_X1  0.01699    0.05669   0.300 0.764456
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 10307.5  on 14227  degrees of freedom
Residual deviance:  9476.3  on 14200  degrees of freedom
AIC: 9532.3

Number of Fisher Scoring iterations: 6
```

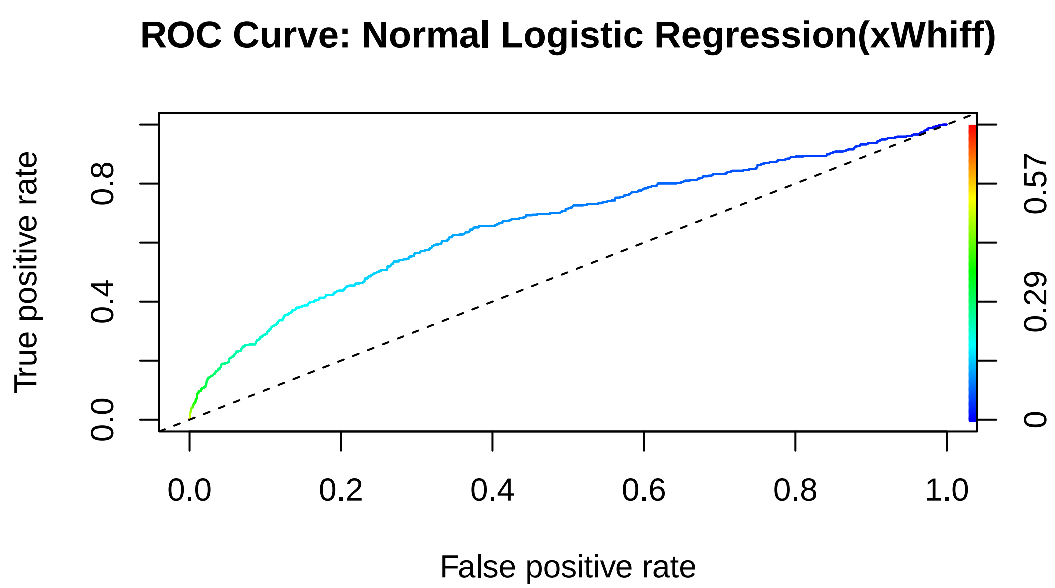
Plotting the ROC curve and calculating the AUC:

```r
test_X <- test_data %>% select(-is_whiff)

pred <- as.numeric(predict(m1, test_X))
pred_probs <- as.numeric(1 / (1 + exp(-pred)))
true_labels <- as.numeric(test_data$is_whiff)

# Calc ROC
pred <- prediction(pred_probs, true_labels)
```

```
perf_m <- performance(pred, "tpr", "fpr")

# Calc AUC
auc <- performance(pred, "auc")
auc_value <- auc@y.values[[1]]
cat("Logistic Regression AUC =", auc_value, "\n")
```

```
Logistic Regression AUC = 0.6657093
```

```
# Plot the Curve
plot(perf_m,
     colorize = TRUE,
     colorkey.label = "Cutoff",
     main = "ROC Curve: Normal Logistic Regression(xWhiff)")
abline(a = 0, b = 1, lty = 2, col = "black")
```



Confusion Matrix:

```
# Convert probabilities to binary class predictions
pred_classes <- factor(ifelse(pred_probs > 0.5, 1, 0))
true_classes <- factor(true_labels)
```

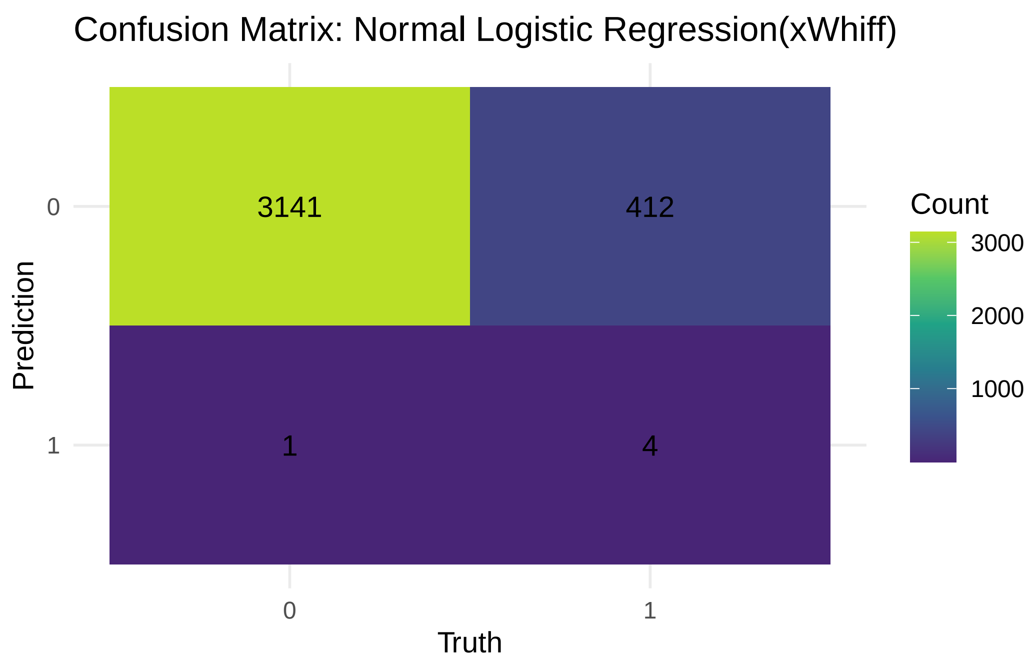
```r
# Caret Confusion Matrix
conf_matrix <- caret::confusionMatrix(data = pred_classes, reference = true_classes)
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 3141  412
         1    1    4

               Accuracy : 0.8839
                 95% CI : (0.8729, 0.8943)
    No Information Rate : 0.8831
    P-Value [Acc > NIR] : 0.4507

                  Kappa : 0.0163

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.999682
            Specificity : 0.009615
         Pos Pred Value : 0.884042
         Neg Pred Value : 0.800000
             Prevalence : 0.883080
         Detection Rate : 0.882799
   Detection Prevalence : 0.998595
      Balanced Accuracy : 0.504649

       'Positive' Class : 0
```

```r
# Compute confusion matrix and plot
results <- tibble(
  truth = true_classes,
  prediction = pred_classes
)

cm <- conf_mat(data = results, truth = truth, estimate = prediction)

autoplot(cm, type = "heatmap") +
```

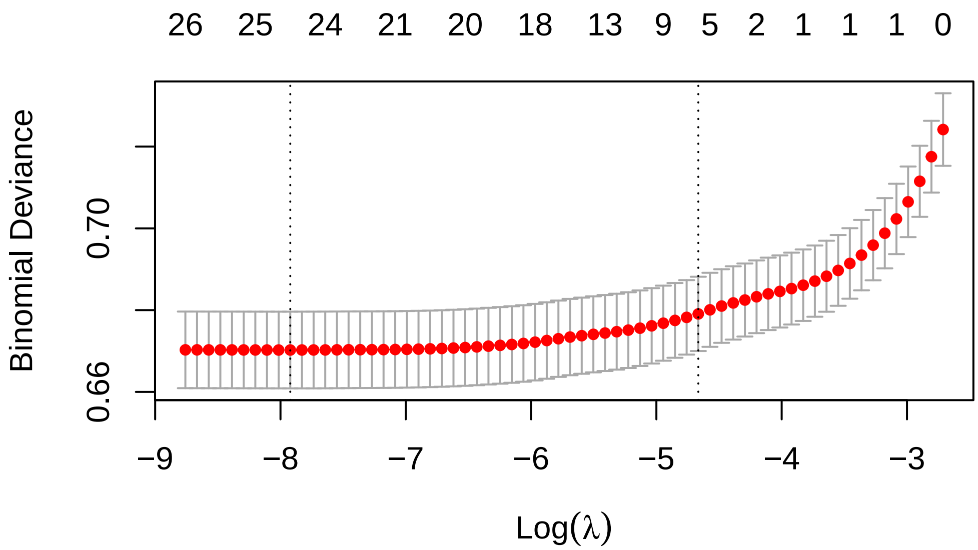
```r
  scale_fill_viridis_c(option = "D", begin = 0.1, end = 0.9) +
  theme_minimal() +
  theme(legend.position = "right") +
  labs(title = "Confusion Matrix: Normal Logistic Regression(xWhiff)", fill = "Count")
```

```
Scale for fill is already present.
Adding another scale for fill, which will replace the existing scale.
```



LASSO Logistic Regression

```r
# Prepare data
x <- model.matrix(is_whiff ~ ., train_data)[, -1]  # Remove intercept
y <- train_data$is_whiff

# Fit LASSO
cv_lasso <- cv.glmnet(x, y, alpha = 1, family = "binomial")  # LASSO
plot(cv_lasso)
```




```
# Get best lambda
best_lambda <- cv_lasso$lambda.min
cat("Best Lambda:", best_lambda, "\n")
```

```
Best Lambda: 0.0003627733
```

```
# Refit final model at best lambda
m2 <- glmnet(x, y, alpha = 1, lambda = best_lambda, family = "binomial")
coef(m2)
```

```
28 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept)   -2.503619556
RelSpeed      -0.025248012
VertBreak      .
HorzBreak      .
SpinRate       0.051527466
SpinAxis_sin   0.189875318
SpinAxis_cos   0.126842549
RelHeight      0.250452507
RelSide        0.091293547
Extension      0.002084440
```

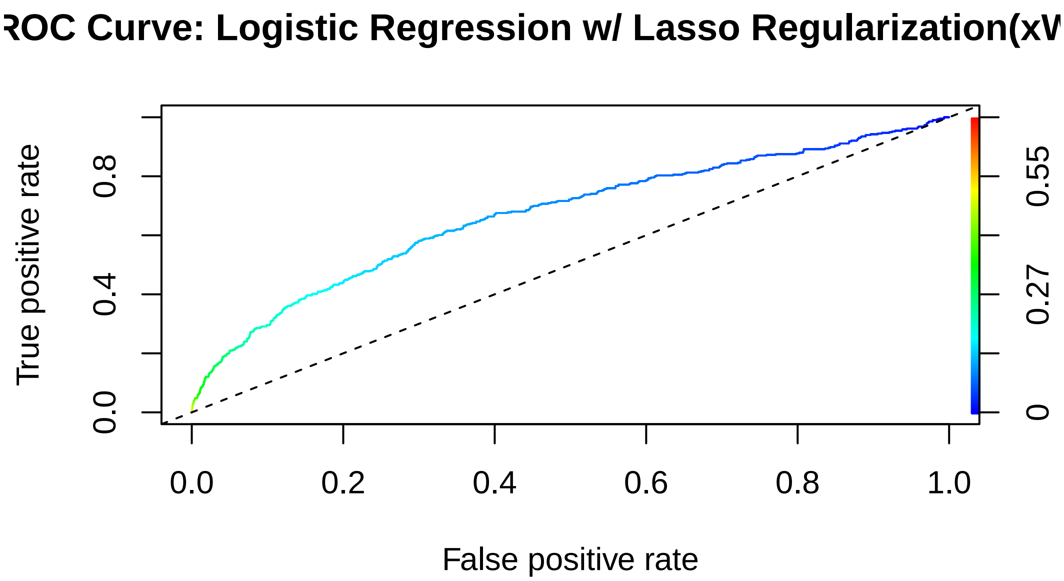
```
VertApprAngle  0.713475394
HorzApprAngle  0.041181245
VertRelAngle   0.029610139
HorzRelAngle   0.003824411
Count_X0.1     0.308183584
Count_X0.2     0.358936014
Count_X1.0     0.022529760
Count_X1.1     0.363889255
Count_X1.2     0.552315672
Count_X2.0     0.206202732
Count_X2.1     0.512415391
Count_X2.2     0.487348642
Count_X3.0    -1.749510489
Count_X3.1     0.491025622
Count_X3.2     0.396226371
Outs_X1        0.077964603
Outs_X2        0.103050214
BatterSide_X1  0.001601198
```

Plotting the ROC curve and calculating the AUC:

```r
x_test <- model.matrix(is_whiff ~ ., data = test_data)[, -1]
pred_probs <- predict(m2, newx = x_test, s = best_lambda, type = "response")
true_labels <- as.numeric(test_data$is_whiff)

# Calc ROC
pred <- prediction(pred_probs, true_labels)
perf_m <- performance(pred, "tpr", "fpr")

# Calc AUC
auc <- performance(pred, "auc")
auc_value <- auc@y.values[[1]]
cat("Logistic Regression w/ Lasso AUC =", auc_value, "\n")
```

```
Logistic Regression w/ Lasso AUC = 0.6695897
```

```r
# Plot the Curve
plot(perf_m,
     colorize = TRUE,
     colorkey.label = "Cutoff",
     main = "ROC Curve: Logistic Regression w/ Lasso Regularization(xWhiff)")
abline(a = 0, b = 1, lty = 2, col = "black")
```



Confusion Matrix:

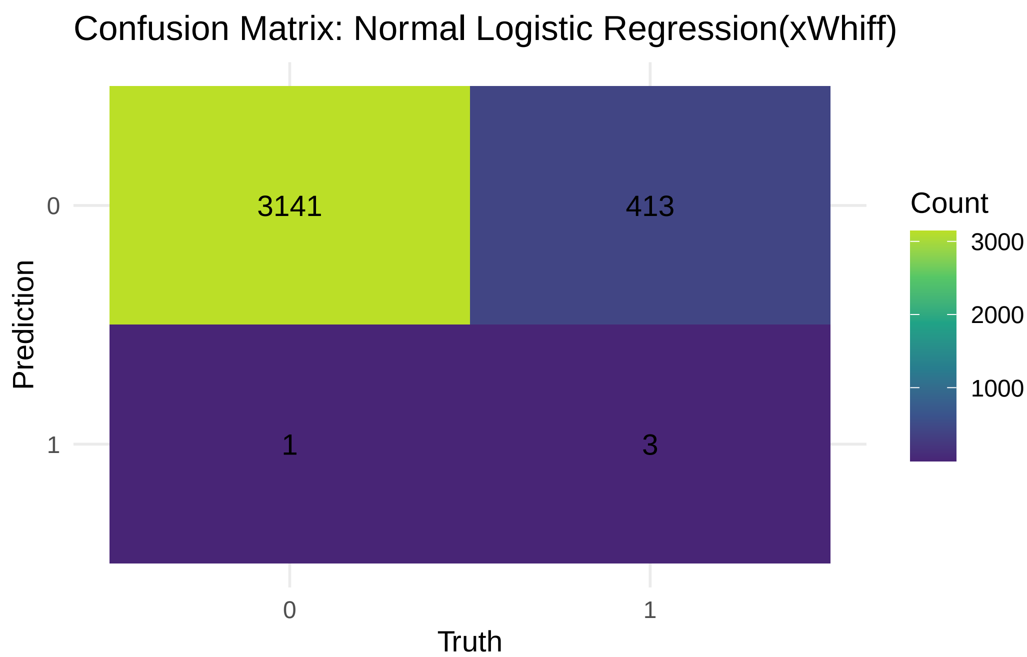
```r
# Convert probabilities to binary class predictions
pred_classes <- factor(ifelse(pred_probs > 0.5, 1, 0))
true_classes <- factor(true_labels)

# Caret Confusion Matrix
conf_matrix <- caret::confusionMatrix(data = pred_classes, reference = true_classes)
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 3141  413
         1    1    3

               Accuracy : 0.8836
                 95% CI : (0.8726, 0.894)
    No Information Rate : 0.8831
    P-Value [Acc > NIR] : 0.4714

                  Kappa : 0.0121
```

```
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.999682
            Specificity : 0.007212
         Pos Pred Value : 0.883793
         Neg Pred Value : 0.750000
             Prevalence : 0.883080
         Detection Rate : 0.882799
   Detection Prevalence : 0.998876
      Balanced Accuracy : 0.503447

       'Positive' Class : 0
```

```r
# Compute confusion matrix and plot
results <- tibble(
  truth = true_classes,
  prediction = pred_classes
)

cm <- conf_mat(data = results, truth = truth, estimate = prediction)

autoplot(cm, type = "heatmap") +
  scale_fill_viridis_c(option = "D", begin = 0.1, end = 0.9) +
  theme_minimal() +
  theme(legend.position = "right") +
  labs(title = "Confusion Matrix: Normal Logistic Regression(xWhiff)", fill = "Count")
```

```
Scale for fill is already present.
Adding another scale for fill, which will replace the existing scale.
```


Confusion Matrix: Normal Logistic Regression(xWhiff)
Prediction
0
1
3141
413
1
3
0
1
Truth
Count
3000
2000
1000