Analysing Poker Players

Predicting winnings and losing poker players

Marc Inizan & Joe Peirson

Who are we?

- We are two poker players looking to be able to quickly and precisely predict if a opposition player is either a winning or a losing player
- If successful in our predictions this will help improve our profits as playing against losing players is more profitable than playing against winning players

What data did we use?

- We had a dataset of hands played by over 8,800 players. Each player has played over 50 hands.
- This dataset included variables such as hands played, net won and then numerous ingame statistics on each player. These statistics included if a played had folded, called or raised on each round of betting.
- Our data was taken from a program called Hold'em Manager 2 which automatically tracks player statistics when we are sitting at a table.

	Player Name	Site	Hands	Net Won	VP$IP	PFR	3Bet	Postflop\nAgg%	W$WSF%	WTSD%
1	.PARTOUCHE.	22	178311	15940.79	0.251039	0.201261	0.077332	0.314294	0.422270	0.242501
2	julien8082	22	110413	20157.79	0.277494	0.239465	0.077709	0.374164	0.475568	0.241437
3	PapiersSVP	0	103957	17549.39	0.257664	0.221630	0.109587	0.345363	0.467605	0.300238
4	NextPlease	22	99879	7326.03	0.260485	0.212327	0.082384	0.328379	0.461007	0.263755
5	123RDC	22	89784	3664.84	0.270026	0.219449	0.080571	0.353109	0.449581	0.251829

Modelling - Assessing our data

We first made sure to check there was not a class imbalance in our results.

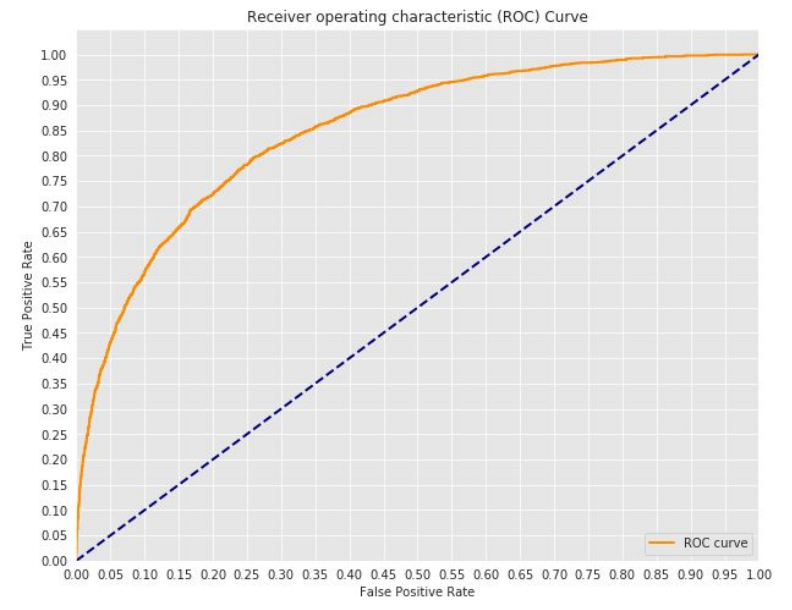
Our results showed that 35% of the players in our dataset were winning players, and 65% were losing players.

We deemed this is a even enough split not to balance the dataset out.

Modelling

- We started by splitting up our data into a training and test dataset.
- Our first baseline model gave a ROC AUC of 0.81. This represents the proportion between the space above the curve and below the curve.
- We added our top 10 interactions and our top 10 polynomials as new features to the dataset.
- After using modelling techniques such as k-nearest neighbours, random forest, decision trees and XGBoost, we chose to proceed with XGBoost as it gave us the best ROC AUC (0.85).

XGBoost Metrics




```
----------------------------------------------------------------------------------------------------

Grid search XGBoost Classifier metrics on TRAIN DATASET
ROC AUC Score: 0.849149676380529

------- Metrics for threshold 0.5 -------
- Precision Score: 0.7391937833899952
- Recall Score: 0.6117363344051447
- Accuracy Score: 0.7870199801615417
- F1 Score: 0.669452386188696

----------------------------------------------------------------------------------------------------

Grid search XGBoost Classifier metrics on TEST DATASET

ROC AUC Score: 0.8025960280856337

------- Metrics for threshold 0.5 -------
- Precision Score: 0.6450304259634888
- Recall Score: 0.5326633165829145
- Accuracy Score: 0.7427762039660056
- F1 Score: 0.58348623853211

----------------------------------------------------------------------------------------------------
Validation ROC AUC: 0.813
----------------------------------------------------------------------------------------------------

Best parameters for XGBoost:
 {'xgb__gamma': 5, 'xgb__max_depth': 3, 'xgb__min_child_weight': 10, 'xgb__subsample': 0.8}
```

Focusing on good players in small stakes games

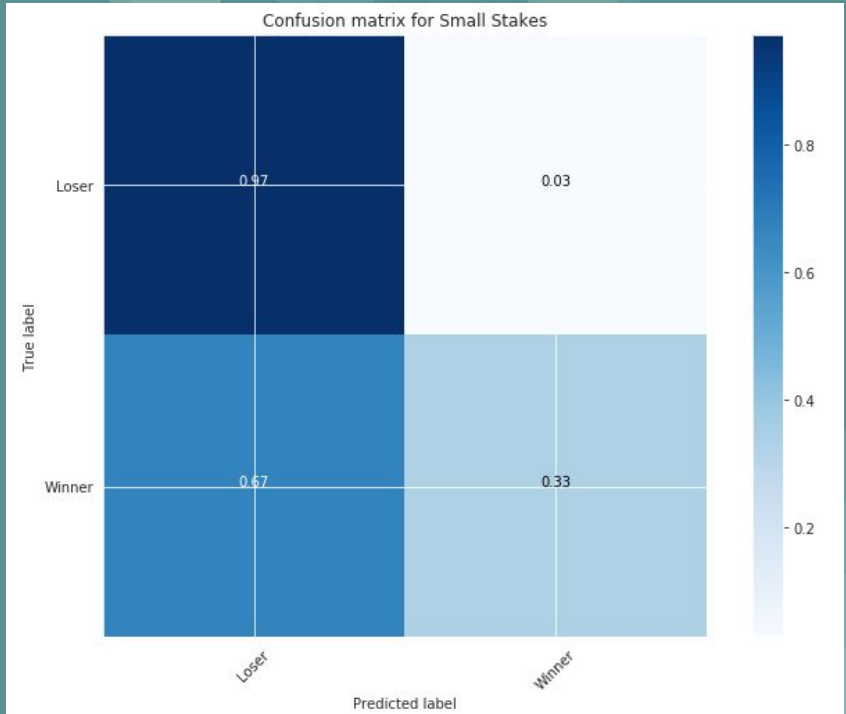
Calculating optimal threshold for small stakes

```
# Defining prevalence
prevalence = 0.25 # The prevalence for good players in small stakes games should be around 25 percent
CostFP_minus_CostTN = 12
CostFN_minus_CostTP = 6

# Calculating the optimal threshold for small stakes games
threshold_small_stakes = threshold_selection(prevalence, CostFP_minus_CostTN, CostFN_minus_CostTP, y=y
print("Threshold small stakes:", round(threshold_small_stakes, 3))
```

```
Threshold small stakes: 0.696
```

Our classifier will only tell us a third of the actual winners on the table but will make mistakes only 13% of the time when predicting winning players.



Focusing on bad players in high stakes games

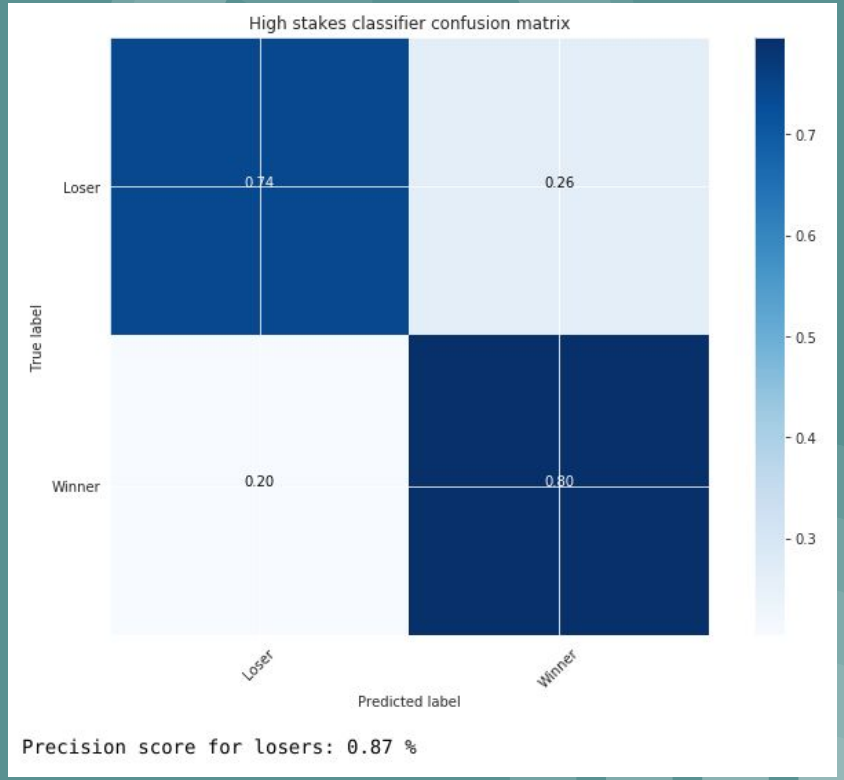
Calculating optimal threshold for high stakes

```
# Defining prevalence
prevalence = 0.75 # The prevalence for good players in high stakes games should be around 75 percent
CostFP_minus_CostTN = 12
CostFN_minus_CostTP = 4

# Calculating the optimal threshold for high stakes games
threshold_high_stakes = threshold_selection(prevalence, CostFP_minus_CostTN, CostFN_minus_CostTP, y=y
print("Threshold high stakes:", round(threshold_high_stakes, 3))
```

```
Threshold high stakes: 0.326
```

Our classifier will tell us 74 percent of the losers on the table and will make mistakes only 13% of the time when predicting losing players.



Predictions in small stakes

```
for i in range(8000, 8020):
    X = df_X.iloc[i].to_frame().T
    prediction(X=X, model=gs_xgb, stakes="small")
```

```
"Playing a small stakes game we'll consider McNamara a Losing Player"
"Playing a small stakes game we'll consider chacham60 a Losing Player"
"Playing a small stakes game we'll consider olav2305 a Losing Player"
"Playing a small stakes game we'll consider BONJOUEURSS a Losing Player"
"Playing a small stakes game we'll consider yesss69 a Losing Player"
"Playing a small stakes game we'll consider peterpan6905 a Losing Player"
"Playing a small stakes game we'll consider The kiffer a Losing Player"
"Playing a small stakes game we'll consider Veve5252 a Losing Player"
"Playing a small stakes game we'll consider IDSA91 a Losing Player"
"Playing a small stakes game we'll consider deuns43 a Losing Player"
"Playing a small stakes game we'll consider ZackEzz a Losing Player"
"Playing a small stakes game we'll consider Portugal_89 a Losing Player"
"Playing a small stakes game we'll consider nidrone a Losing Player"
"Playing a small stakes game we'll consider labellesarah a Losing Player"
"Playing a small stakes game we'll consider BigDaddy29 a Losing Player"
"Playing a small stakes game we'll consider PATCEL888 a Losing Player"
"Playing a small stakes game we'll consider elcooki a Losing Player"
"Playing a small stakes game we'll consider crapette0611 a Losing Player"
"Playing a small stakes game we'll consider booty 973 a Losing Player"
"Playing a small stakes game we'll consider tomb92 a Losing Player"
```

Predictions in high stakes

```
for i in range(8000, 8020):
    X = df_X.iloc[i].to_frame().T
    prediction(X=X, model=gs_xgb, stakes="high")
```

```
"Playing a high stakes game we'll consider McNamara a Losing Player"
"Playing a high stakes game we'll consider chacham60 a Losing Player"
"Playing a high stakes game we'll consider olav2305 a Losing Player"
"Playing a high stakes game we'll consider BONJOUEURSS a Losing Player"
"Playing a high stakes game we'll consider yesss69 a Winning Player"
"Playing a high stakes game we'll consider peterpan6905 a Losing Player"
"Playing a high stakes game we'll consider The kiffer a Losing Player"
"Playing a high stakes game we'll consider Veve5252 a Winning Player"
"Playing a high stakes game we'll consider IDSA91 a Winning Player"
"Playing a high stakes game we'll consider deuns43 a Losing Player"
"Playing a high stakes game we'll consider ZackEzz a Winning Player"
"Playing a high stakes game we'll consider Portugal_89 a Winning Player"
"Playing a high stakes game we'll consider nidrone a Winning Player"
"Playing a high stakes game we'll consider labellesarah a Winning Player"
"Playing a high stakes game we'll consider BigDaddy29 a Losing Player"
"Playing a high stakes game we'll consider PATCEL888 a Winning Player"
"Playing a high stakes game we'll consider elcooki a Winning Player"
"Playing a high stakes game we'll consider crapette0611 a Losing Player"
"Playing a high stakes game we'll consider booty 973 a Losing Player"
"Playing a high stakes game we'll consider tomb92 a Losing Player"
```

Conclusions

Using different threshold for different situations we were able to generate meaningful and precise predictions.

We can now deploy this model and use it when we play online poker.

- We are able to predict with high certainty if a player is a losing player in high stakes games.
- We were able to predict with high certainty if a player is a winning player in small stakes games.

Having those informations, we can now target ***fishes*** when playing high stakes poker and avoiding ***sharks*** when playing small stakes poker and translate those informations into profits

Questions & Answers