PREDICTING UFC FIGHTS

Michael Cassetti + Shayan Ruhany

PRESENTATION OVERVIEW

1. Problem Introduction
2. UFC Background
3. Our data + cleaning
4. Our testing approach
5. Running methods
6. Compiling methods
7. Analyzing data
8. Problems with our models
9. Ethical concerns
10. Conclusion + Questions

OUR PROBLEM

The UFC has **too many "oh sh*t" moments** and it's challenging to bet on fights and be profitable.

We want to develop **ML models** to accurately predict UFC fights to **bet profitably.**

WHAT IS THE UFC?

MMA PROMOTION

- 700 Fighters on roster
- 43 UFC events in 2021
- President Dana White
- Think Conor McGregor

- “Most oh sh*t moments” - Joe Rogan

WEIGHT CLASSES

8 MEN'S DIVISION

- 125 - Flyweight
- 135 - Bantamweight
- 145 - Featherweight
- 155 - Lightweight
- 170 - Welterweight
- 185 - Middleweight
- 205 - Light Heavyweight
- < 265 - Heavyweight

HOW TO WIN IN THE UFC

01 KNOCKOUT/TKO

Knockout your opponent or referee steps in right before

02 SUBMISSION

Opponent taps out from choke, hold, position, etc.

03 DECISION

Judges decide after the 3 or 5 round fight who won

04 DOCTOR'S STOPPAGE

Doctors deems one fighter unfit to continue

01

OUR DATA

GATHERING RAW DATA

OUR RAW DATA

119 x 4896

FIGHTER INFO	ODDS	HISTORICAL	COMPARATIVE
Name	R Odds	Wins	Better Rank
Reach	B Odds	Losses	Age Difference
Stance	KO odds	Takedown accuracy	Sig Strike Difference
Weight Class	Submission Odds	Sig Strike Accuracy	Win Streak Difference

02 CLEANING DATA

CLEANING RAW DATA

OUR DATA CLEANING METHOD

REMOVE JUNK

Get rid of useless columns

CREATE WEIGHT CLASS CSVs

Analyze each weight class individually

ONLY MALE FIGHTS

Not enough data for female fights

```
['R_fighter','B_fighter','date','location','country','gender','finish','finish_details','empty_arena','constant_1','finish
['B_match_weightclass_rank','R_match_weightclass_rank',"R_Women's Flyweight_rank","R_Women's Featherweight_rank","R_Women'
["B_Women's Flyweight_rank","B_Women's Featherweight_rank","B_Women's Strawweight_rank","B_Women's Bantamweight_rank",'B_H
```

03 TESTING APPROACH

HOW WE WILL MODEL

REGRESSION vs. CLASSIFICATION

CLASSIFICATION PROBLEM

TESTING METHODS

Decision Trees

Random Forest

AdaBoost

Test data through different ML models to gather F1 scores and feature importances

Compile Data

Bring all weight classes and models together to compare between everything

Make Sense of Data

Investigate feature importances and weight classes to determine betting strategy

04 RUNNING METHODS

DT, RF, & BOOSTING

RUNNING METHODS

STEP 1

- 8 Weight Classes x 3 Methods
- 24 Total files
  - 24 Feature importances
  - 24 F1 Scores
  - 24 Plots
  - 24 Pickle files

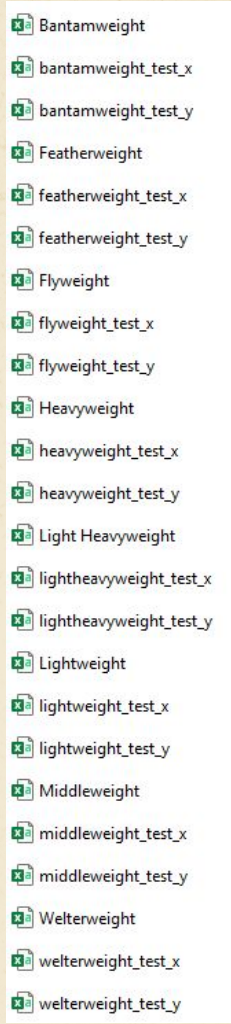
- Automate as much as possible

RUNNING METHODS

ALL_Weightclasses

- What if we just did every weight class together?
  - Simplified
  - Less work
  - Less analysis

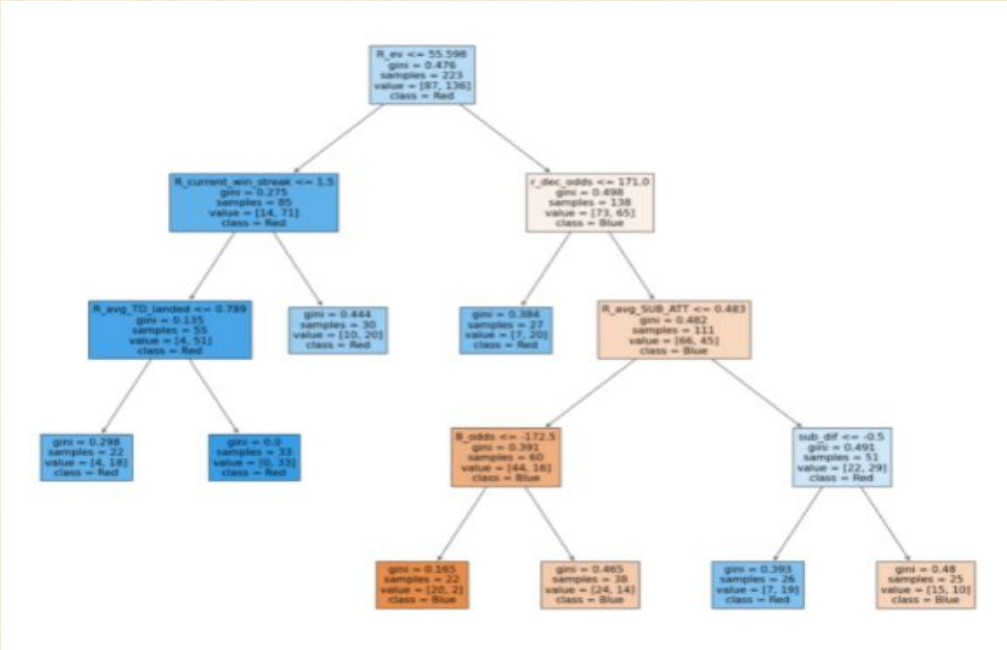
- F1 Scores showed no improvement
  - Decreased in some cases
- Shows that weight classes have different feature importances



DECISION TREES

RUNNING METHODS

- Tree for bantamweight weight class



DECISION TREES

- Tuning for bantamweight tree

HYPERPARAMETER TUNING

```
print('Initial score: ', model_cv.best_score_)
print('Initial parameters: ', model_cv.best_params_)
```

```
Initial score:  0.6285042133498131
Initial parameters:  {'max_depth': 10, 'min_samples_leaf': 30, 'min_samples
_split': 20}
```

```
# Adapt grid based on result from initial grid search
hyper_params_new = {
    'max_depth': list(range(2, 12)),
    'min_samples_split': list(range(15, 24)),
    'min_samples_leaf': list(range(22, 30))
}
```

```
# Call GridSearchCV()
model_cv = GridSearchCV(estimator = classifier,
                        param_grid = hyper_params_new,
                        scoring= 'f1_weighted',
                        cv = folds,
                        verbose = 1,
                        n_jobs = -1) # Will utilize all available CPUs
```

```
# Fit the model
model_cv.fit(train_X, train_y)
```

```
Fitting 5 folds for each of 720 candidates, totalling 3600 fits
```

```
3]: GridSearchCV(cv=KFold(n_splits=5, random_state=100, shuffle=True),
             estimator=DecisionTreeClassifier(random_state=42), n_jobs=-1,
             param_grid={'max_depth': [2, 3, 4, 5, 6, 7, 8, 9, 10, 11],
                         'min_samples_leaf': [22, 23, 24, 25, 26, 27, 28, 2
9],
                         'min_samples_split': [15, 16, 17, 18, 19, 20, 21,
22,
                                               23]},
             scoring='f1_weighted', verbose=1)
```

```
print('Improved score: ', model_cv.best_score_)
print('Improved parameters: ', model_cv.best_params_)
```

```
Improved score:  0.6385930755314816
Improved parameters:  {'max_depth': 4, 'min_samples_leaf': 22, 'min_samples
_split': 15}
```

RANDOM FOREST

RUNNING METHODS

- Feature importances for featherweight weight class

Feature Importance

```
# Storing importance values from the best fit model
importance = loaded_model.best_estimator_.feature_importances_
```

```
# Displaying feature importance as a dataframe
feature_imp = pd.DataFrame(list(zip(train_X.columns, importance)),
                columns = ['Feature', 'Importance'])

feature_imp = feature_imp.sort_values('Importance', ascending = False).r

feature_imp
```

2]:

	Feature	Importance
0	R_ev	0.055184
1	R_odds	0.053545
2	B_ev	0.050801
3	B_odds	0.047691
4	b_dec_odds	0.042396
...	...	...
71	R_win_by_TKO_Doctor_Stoppage	0.000000
72	B_draw	0.000000
73	B_win_by_Decision_Majority	0.000000
74	R_win_by_Decision_Majority	0.000000
75	R_draw	0.000000

76 rows × 2 columns

RANDOM FOREST

- Score slightly went down after tuning

HYPERPARAMETER TUNING

```
Initial score:  0.5867349938449047
Initial parameters:  {'n_estimators': 450, 'min_samples_split': 10, 'min_sa
mples_leaf': 5, 'max_depth': 6}
```

```
# Create the parameter grid based on the results of random search
param_grid = {
    'max_depth': [11, 12, 13],
    'min_samples_leaf': [2, 3, 4, 5],
    'min_samples_split': [2, 3, 4, 5],
    'n_estimators': [180, 200, 220]
}

pprint(param_grid)
```

```
{'max_depth': [11, 12, 13],
 'min_samples_leaf': [2, 3, 4, 5],
 'min_samples_split': [2, 3, 4, 5],
 'n_estimators': [180, 200, 220]}
```

```
# Call GridSearchCV()
model_cv = GridSearchCV(estimator = classifier,
                        param_grid = param_grid,
                        scoring= 'f1_macro',
                        cv = folds,
                        verbose = 1,
                        n_jobs = -1) # Will utilize all available CPUs
```

```
# Fit the model
start = time.time()            # Start Time
model_cv.fit(train_X, train_y)
stop = time.time()             # End Time
print(f"Training time: {stop - start}s")
```

```
Fitting 5 folds for each of 144 candidates, totalling 720 fits
Training time: 41.92339897155762s
```

```
print('Improved score: ', model_cv.best_score_)
print('Improved parameters: ', model_cv.best_params_)
```

```
Improved score:  0.5806372042728811
Improved parameters:  {'max_depth': 11, 'min_samples_leaf': 5, 'min_samples
_split': 2, 'n_estimators': 180}
```

ADABOOST

- Flyweight weight class feature importances

RUNNING METHODS

```
# Storing importance values from the best fit model
importance = grid_cv.best_estimator_.feature_importances_
```

```
# Displaying feature importance as a dataframe
feature_imp = pd.DataFrame(list(zip(train_X.columns, importance)),
                columns = ['Feature', 'Importance'])

feature_imp = feature_imp.sort_values('Importance', ascending = False).r

feature_imp
```

	Feature	Importance
0	b_sub_odds	0.083668
1	avg_sub_att_dif	0.057793
2	sig_str_dif	0.047240
3	loss_dif	0.045971
4	B_age	0.037425
...	...	...
71	R_current_win_streak	0.000000
72	B_Weight_lbs	0.000000
73	B_Reach_cms	0.000000
74	B_Stance	0.000000
75	B_total_title_bouts	0.000000

76 rows × 2 columns

ADABOOST

- Tuning for Flyweight weight class

HYPERPARAMETER TUNING

```
Initial score:  0.5867349938449047
Initial parameters:  {'n_estimators': 450, 'min_samples_split': 10, 'min_sa
mples_leaf': 5, 'max_depth': 6}
```

```python
# Create the parameter grid based on the results of random search
param_grid = {
    'max_depth': [11, 12, 13],
    'min_samples_leaf': [2, 3, 4, 5],
    'min_samples_split': [2, 3, 4, 5],
    'n_estimators': [180, 200, 220]
}

pprint(param_grid)
```

```
{'max_depth': [11, 12, 13],
 'min_samples_leaf': [2, 3, 4, 5],
 'min_samples_split': [2, 3, 4, 5],
 'n_estimators': [180, 200, 220]}
```

```python
# Call GridSearchCV()
model_cv = GridSearchCV(estimator = classifier,
                        param_grid = param_grid,
                        scoring= 'f1_macro',
                        cv = folds,
                        verbose = 1,
                        n_jobs = -1) # Will utilize all available CPUs
```

```python
# Fit the model
start = time.time()            # Start Time
model_cv.fit(train_X, train_y)
stop = time.time()             # End Time
print(f"Training time: {stop - start}s")
```

```
Fitting 5 folds for each of 144 candidates, totalling 720 fits
Training time: 41.92339897155762s
```

```python
print('Improved score: ', model_cv.best_score_)
print('Improved parameters: ', model_cv.best_params_)
```

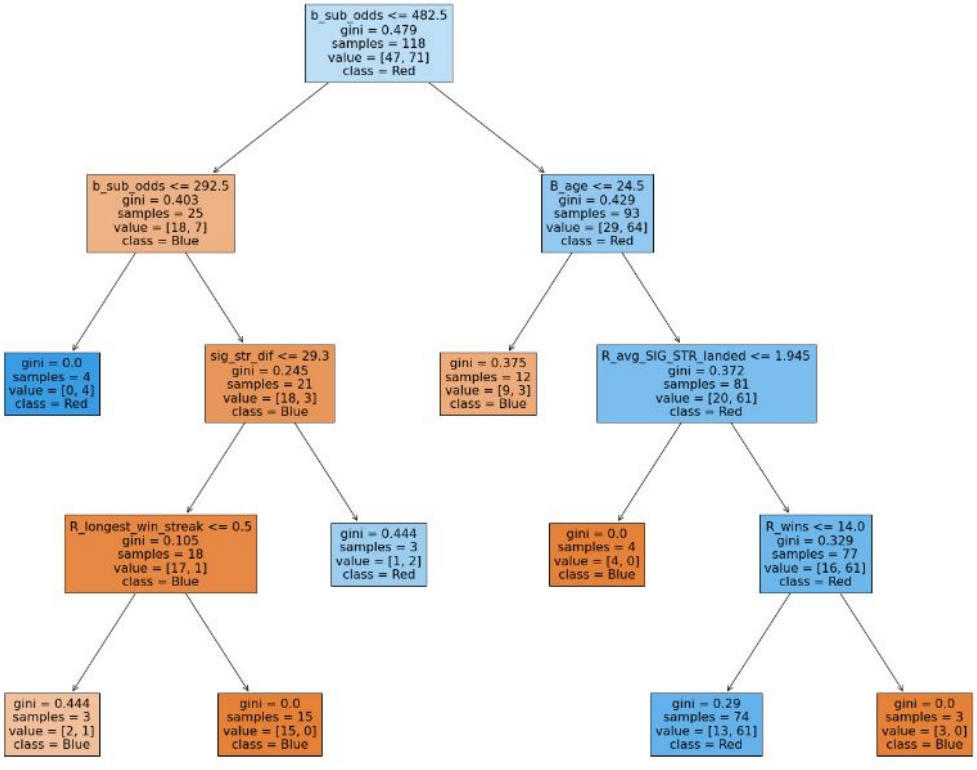
```
Improved score:  0.5806372042728811
Improved parameters:  {'max_depth': 11, 'min_samples_leaf': 5, 'min_samples
_split': 2, 'n_estimators': 180}
```

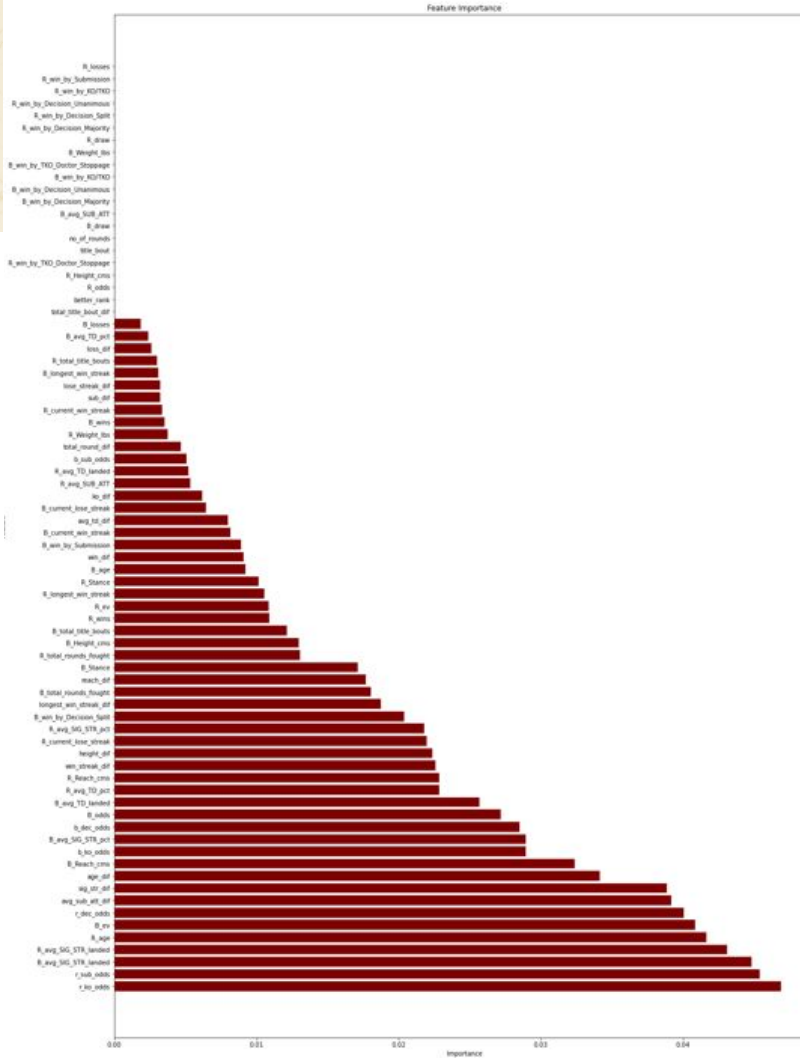
RUNNING METHODS

RULES

- Rule 1: Stay organized with all of the files
- Rule 2: Name everything uniformly
- Rule 3: Make sure Copy/Paste in right spots

NOT SO USEFUL....





05 COMPILING METHODS

PICKLE FILES & MORE

COMPILING METHODS

STEP 1

- Import all pickle files
  - Automate as much as possible
  - Keep names uniform

- Use **for loop** and list of models
  - Fast and efficient
  - Very modular

DECISION TREES

```
flyweight_decisiontrees_model = pickle.load(open('flyweight_tree_pickle.sav', 'rb'))
bantamweight_decisiontrees_model = pickle.load(open('bantamweight_tree_pickle.sav', 'rb'))
featherweight_decisiontrees_model = pickle.load(open('featherweight_tree_pickle.sav', 'rb'))
lightweight_decisiontrees_model = pickle.load(open('lightweight_tree_pickle.sav', 'rb'))
welterweight_decisiontrees_model = pickle.load(open('welterweight_tree_pickle.sav', 'rb'))
middleweight_decisiontrees_model = pickle.load(open('middleweight_tree_pickle.sav', 'rb'))
lightheavyweight_decisiontrees_model = pickle.load(open('light_heavyweight_tree_pickle.sav', 'rb'))
heavyweight_decisiontrees_model = pickle.load(open('heavyweight_tree_pickle.sav', 'rb'))
```

BOOSTING

```
flyweight_boosting_model = pickle.load(open('flyweight_boosting_pickle.sav', 'rb'))
bantamweight_boosting_model = pickle.load(open('bantamweight_boosting_pickle.sav', 'rb'))
featherweight_boosting_model = pickle.load(open('featherweight_boosting_pickle.sav', 'rb'))
lightweight_boosting_model = pickle.load(open('lightweight_boosting_pickle.sav', 'rb'))
welterweight_boosting_model = pickle.load(open('welterweight_boosting_pickle.sav', 'rb'))
middleweight_boosting_model = pickle.load(open('middleweight_boosting_pickle.sav', 'rb'))
lightheavyweight_boosting_model = pickle.load(open('lightheavyweight_boosting_pickle.sav', 'rb'))
heavyweight_boosting_model = pickle.load(open('heavyweight_boosting_pickle.sav', 'rb'))
```

Random Forest

```
flyweight_randomforest_model = pickle.load(open('flyweight_randomforest_pickle.sav', 'rb'))
bantamweight_randomforest_model = pickle.load(open('bantamweight_randomforest_pickle.sav', 'rb'))
featherweight_randomforest_model = pickle.load(open('featherweight_randomforest_pickle.sav', 'rb'))
lightweight_randomforest_model = pickle.load(open('lightweight_randomforest_pickle.sav', 'rb'))
welterweight_randomforest_model = pickle.load(open('welterweight_randomforest_pickle.sav', 'rb'))
middleweight_randomforest_model = pickle.load(open('middleweight_randomforest_pickle.sav', 'rb'))
lightheavyweight_randomforest_model = pickle.load(open('lightheavyweight_randomforest_pickle.sav', 'rb'))
heavyweight_randomforest_model = pickle.load(open('heavyweight_randomforest_pickle.sav', 'rb'))
```

COMPILING METHODS

STEP 2

- Add all feature importances together among each weight class
  - From 24 DF to 8 DF

- Sort by [‘Feature’] when adding together
  - Might be adding wrong feature importance values together

	Feature	Importance
0	b_sub_odds	0.181769
1	B_age	0.094701
2	b_dec_odds	0.084748
3	B_odds	0.071925
4	avg_sub_att_dif	0.044649
...	...	...
71	R_win_by_TKO_Doctor_Stoppage	0.000000
72	B_win_by_TKO_Doctor_Stoppage	0.000000
73	R_draw	0.000000
74	B_win_by_Decision_Majority	0.000000
75	R_win_by_Decision_Majority	0.000000

FLYWEIGHT

```
flyweight_df = flyweight_rf_imp + flyweight_dt_imp + flyweight_b_imp
flyweight_df['Feature'] = flyweight_rf_imp['Feature']

flyweight_df['Importance'] = flyweight_df['Importance'] / sum(flyweight_df['Importance'])
flyweight_df
```

COMPILING METHODS

STEP 3

- Average out the feature importances from the 8 weight classes
  - From 8 DF to 1 DF

- Now we have the overall most importance features
  - Very consistent throughout all weight classes

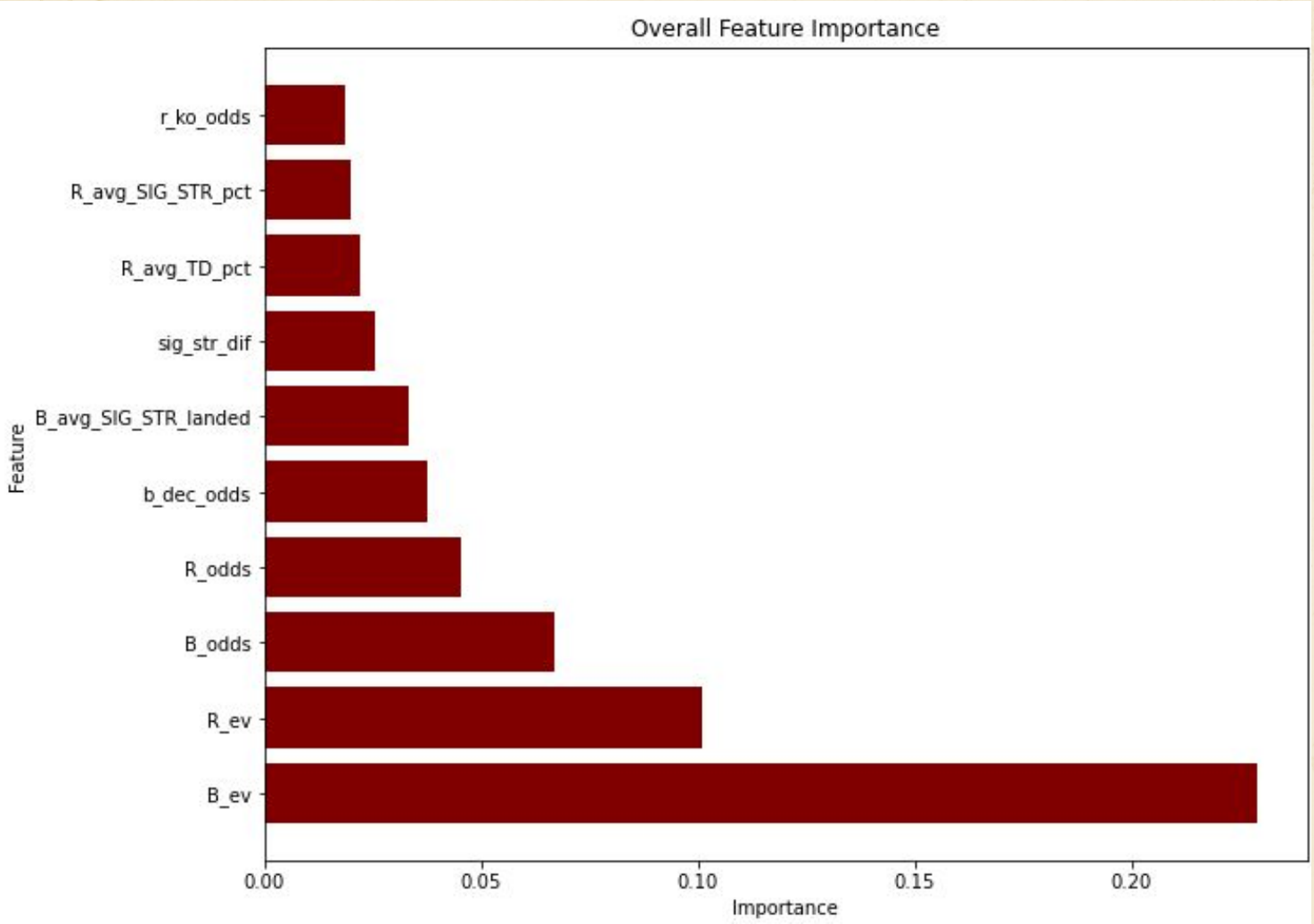
	Feature	Importance
0	B_ev	0.229117
1	R_ev	0.101199
2	B_odds	0.066907
3	R_odds	0.045285
4	b_dec_odds	0.037850
5	B_avg_SIG_STR_landed	0.033277
6	sig_str_dif	0.025358
7	R_avg_TD_pct	0.022012
8	R_avg_SIG_STR_pct	0.019914
9	r_ko_odds	0.018733
10	b_ko_odds	0.017864

```
total_df = flyweight_df + bantamweight_df + featherweight_df + lightweight_df + welterw
total_df['Feature'] = heavyweight_rf_imp['Feature']
total_df['Importance'] = total_df['Importance'] / sum(total_df['Importance'])
total_imp_df = total_df.copy()
total_df
```

06 ANALYZE DATA

F1 SCORES + BOX PLOT

IMPORTANCES



WHY INCLUDE ODDS?

ANALYZING DATA

BASED OFF MODELS

Bookie/Vegas sets the betting line based off of their prediction (model-based)

WHY ISN'T EVERYTHING ELSE 0?

If odds were only thing that mattered, feature importance would be 1

BEST IMPORTANCE

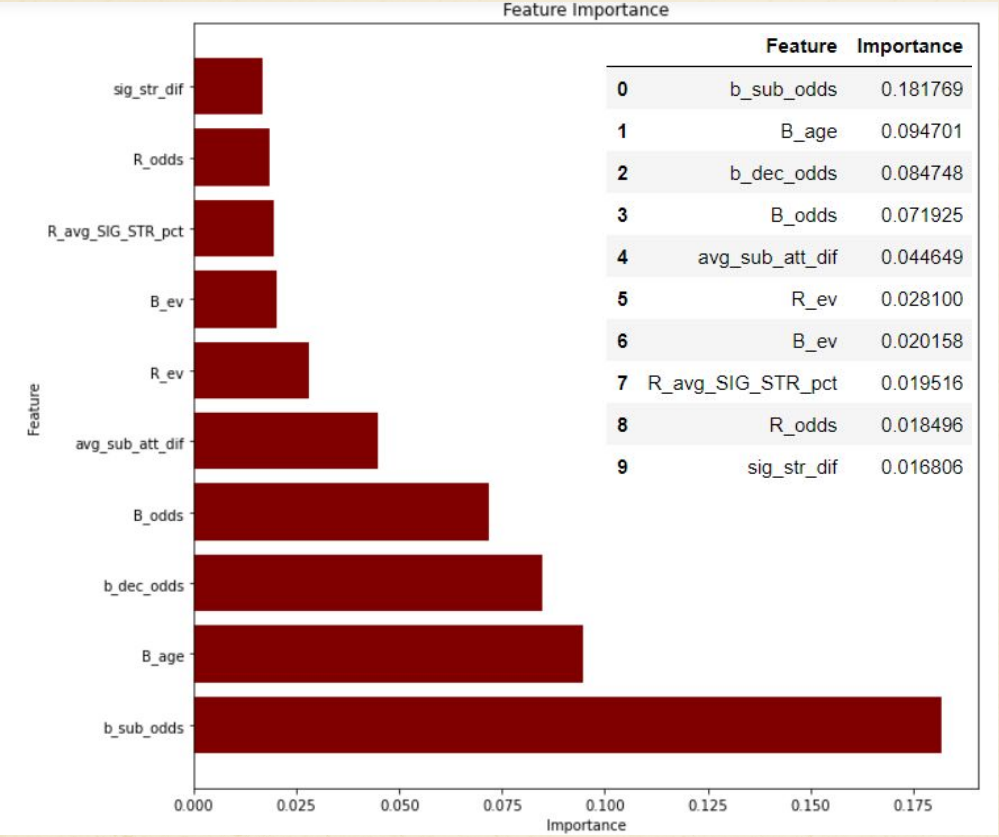
This is the most important feature in our model, helps it's accuracy

IMPROVED MODELS

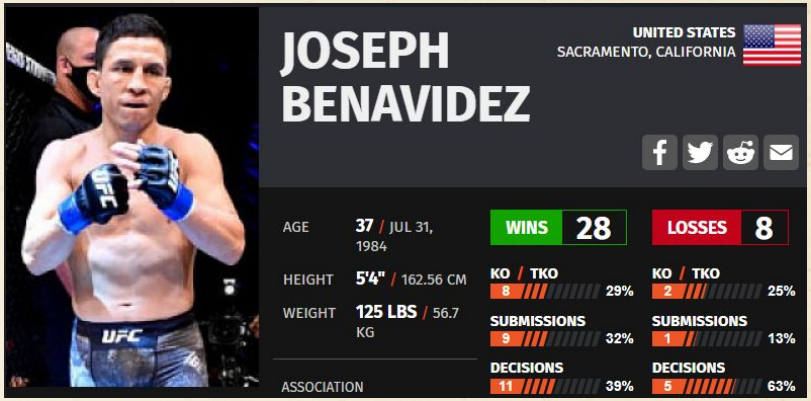
Use embedded models (Vegas set odds) + our data to better the models

FLYWEIGHT ANALYSIS

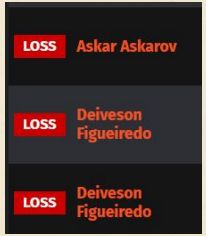
IMPORTANCES



	Feature	Importance
0	b_sub_odds	0.181769
1	B_age	0.094701
2	b_dec_odds	0.084748
3	B_odds	0.071925
4	avg_sub_att_dif	0.044649
5	R_ev	0.028100
6	B_ev	0.020158
7	R_avg_SIG_STR_pct	0.019516
8	R_odds	0.018496
9	sig_str_dif	0.016806

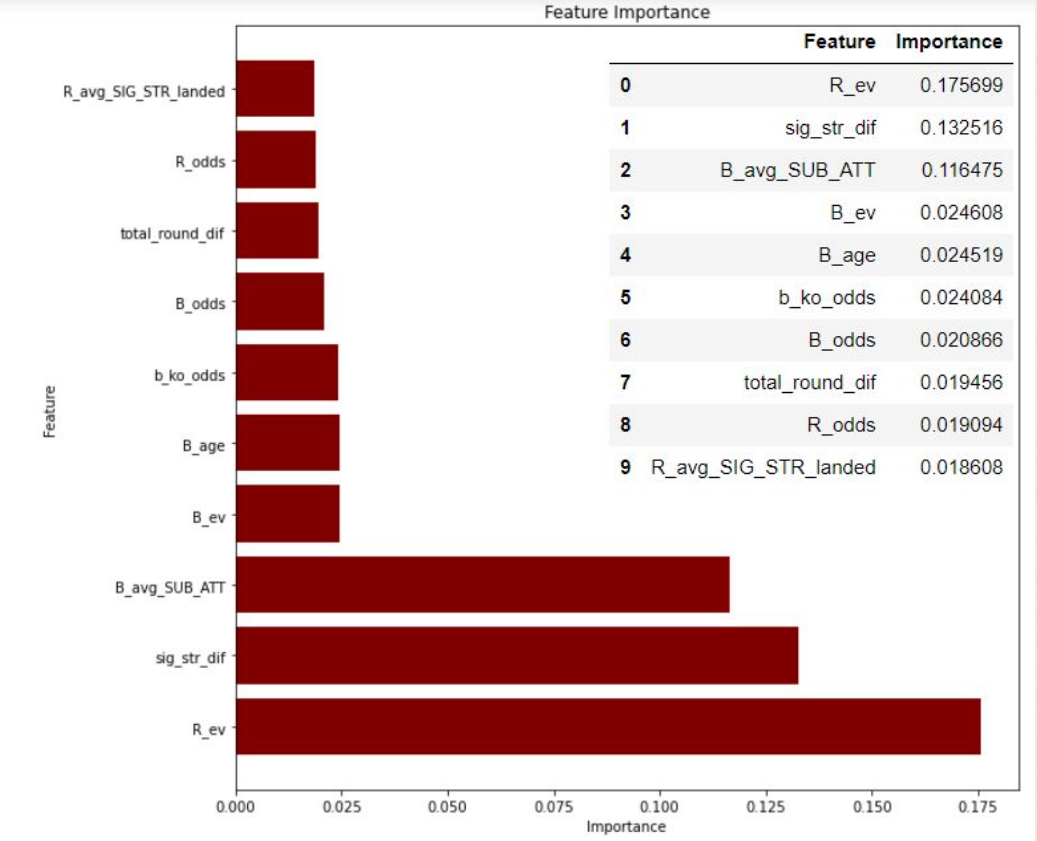


- Example: Joseph Benevidez
  - Older, been in weghtclass long time because it's not as competitive
  - Gives competition time to catch up and study
  - Last 3 Fights →

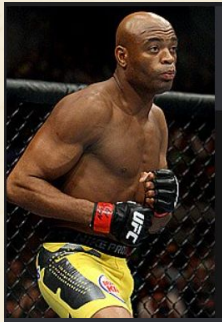


MIDDLEWEIGHT ANALYSIS

IMPORTANCES



	Feature	Importance
0	R_ev	0.175699
1	sig_str_dif	0.132516
2	B_avg_SUB_ATT	0.116475
3	B_ev	0.024608
4	B_age	0.024519
5	b_ko_odds	0.024084
6	B_odds	0.020866
7	total_round_dif	0.019456
8	R_odds	0.019094
9	R_avg_SIG_STR_landed	0.018608



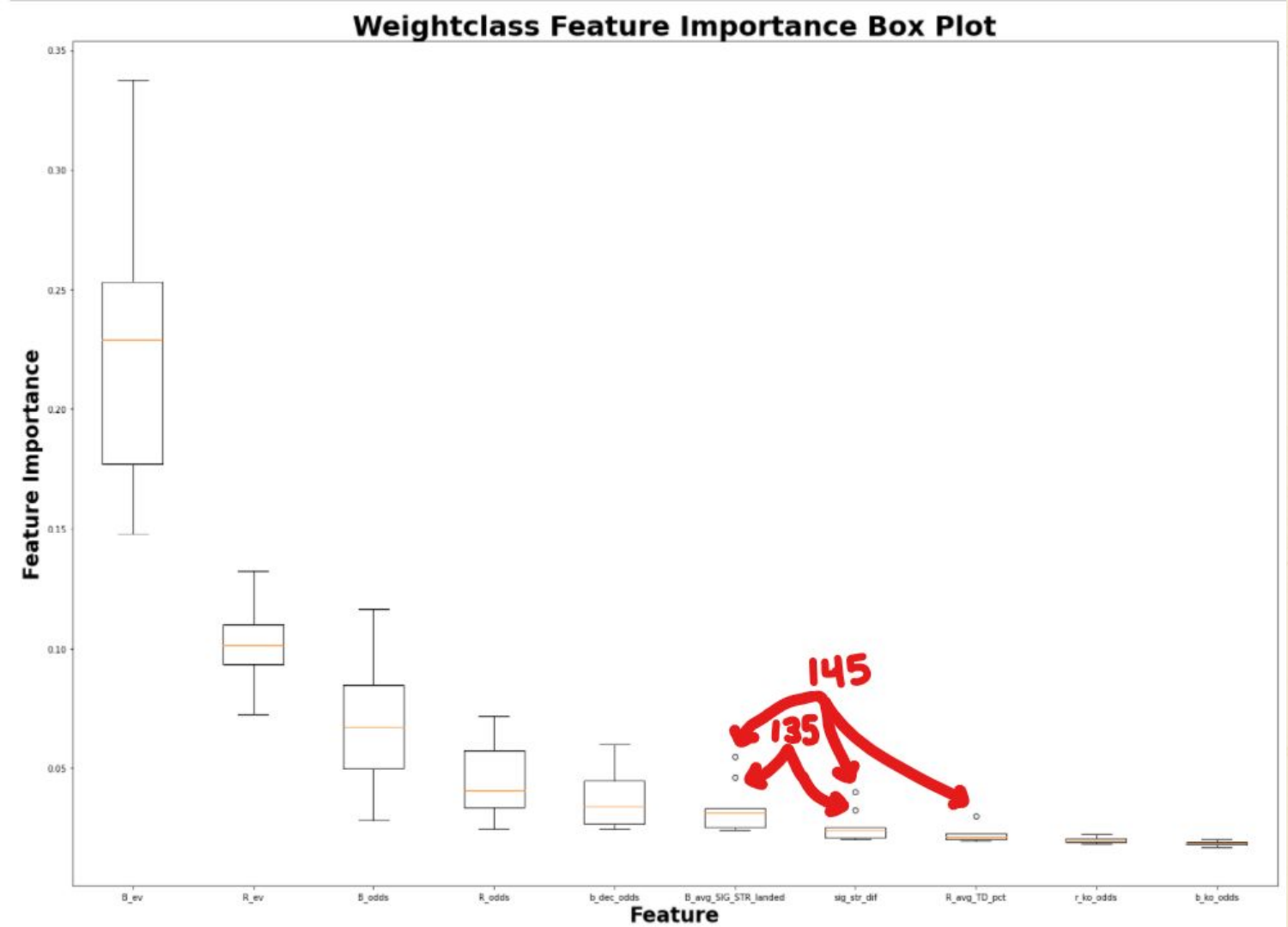
ANDERSON SILVA — BRAZIL, CURITIBA, PARANA

"The Spider"

AGE	46 / APR 14, 1975
HEIGHT	6'2" / 187.96 CM
WEIGHT	185 LBS / 83.91 KG
ASSOCIATION	TEAM NOGUEIRA / X-GYM /

WINS 34		LOSSES 11	
KO / TKO	23 — 68%	KO / TKO	4 — 36%
SUBMISSIONS	3 — 9%	SUBMISSIONS	2 — 18%
DECISIONS	8 — 24%	DECISIONS	4 — 36%
OTHERS	0 — 0%	OTHERS	1 — 9%

WIN	Stephan Bonnar	UFC 153 - Silva vs. Bonnar Oct / 13 / 2012	TKO (Knee to the Body and Punches) Marc Goddard	1 4:40
WIN	Chael Sonnen	UFC 148 - Silva vs. Sonnen 2 Jul / 07 / 2012	TKO (Knee to the Body and Punches) Yves Lavigne	2 1:55
WIN	Yushin Okami	UFC 134 - Silva vs. Okami Aug / 27 / 2011	TKO (Punches) Herb Dean	2 2:04
WIN	Vitor Belfort	UFC 126 - Silva vs. Belfort Feb / 05 / 2011	KO (Front Kick and Punches) Mario Yamasaki	1 3:25
WIN	Chael Sonnen	UFC 117 - Silva vs. Sonnen Aug / 07 / 2010	Submission (Triangle Armbar) Josh Rosenthal	5 3:10
WIN	Demian Maia	UFC 112 - Invincible Apr / 10 / 2010	Decision (Unanimous) Dan Miragliotta	5 5:00
WIN	Forrest Griffin	UFC 101 - Declaration Aug / 08 / 2009	KO (Punch) Kevin Mulhall	1 3:23
WIN	Thales Leites	UFC 97 - Redemption Apr / 18 / 2009	Decision (Unanimous) Yves Lavigne	5 5:00
WIN	Patrick Cote	UFC 90 - Silva vs. Cote Oct / 25 / 2008	TKO (Knee Injury) Herb Dean	3 0:39
WIN	James Irvin	UFC Fight Night 14 - Silva vs. Irvin	KO (Punches) Mario Yamasaki	1 1:01



BOX PLOT

- Featherweight: B_avg_sig_STR_landed, sig_str_dif, R_avg_TD_pct
- Bantamweight: B_avg_sig_STR_landed, sig_str_dif

ANALYZING DATA

F1 SCORES

- Evaluate the 24 F1 scores

- Lightweight best division
  - More fighters & fights (3/11 fights UFC 272)
- Middleweight worst division
  - Less fights & fights (0/11 fights UFC 272)

Method	Flyweight F1	Bantamweight F1	Featherweight F1	Lightweight F1	Welterweight F1	Middleweight F1	Light Heavyweight F1	Heavyweig
Random Forest	0.533252	0.593501	0.580637	0.691369	0.599861	0.551011	0.631731	0.5884
Decision Trees	0.64611	0.62062	0.62062	0.673717	0.622315	0.580291	0.629252	0.6292
Boosting	0.56757	0.613085	0.591998	0.613085	0.622076	0.534733	0.68983	0.5769

	Method	Flyweight F1	Bantamweight F1	Featherweight F1	Lightweight F1	Welterweight F1	Middleweight F1	Light Heavyweight F1	Heavyweight F1
0	Random Forest	0.533252	0.593501	0.580637	0.691369	0.599861	0.551011	0.631731	0.588449
1	Decision Trees	0.646110	0.620620	0.620620	0.673717	0.622315	0.580291	0.629252	0.629252
2	Boosting	0.567570	0.613085	0.591998	0.613085	0.622076	0.534733	0.689830	0.576989

07 OKAY, AND?

DATA TO REAL LIFE USE

PROBABILITY TO PROFIT

ANALYSIS BREAKDOWN

92.7

AVERAGE EXPECTED VALUE OF WINNER

Average bet of $100 yields $92.70 in profit

51.9%

IMPLIED PROBABILITY OF WINNER

According to our 3094 fights of data, you need to predict 51.9% of fights to break even on consistent $100 bets

BETTING PROFITABILITY

LIGHTWEIGHT + LHW

Best F1 scores (0.69) - better than 51.9%

LONG TERM

This strategy works in the long run and require diamond hands

Method	Flyweight F1	Bantamweight F1	Featherweight F1	Lightweight F1	Welterweight F1	Middleweight F1	Light Heavyweight F1	Heavyweight F1
Random Forest	0.533252	0.593501	0.580637	0.691369	0.599861	0.551011	0.631731	0.588449
Decision Trees	0.64611	0.62062	0.62062	0.673717	0.622315	0.580291	0.629252	0.629252
Boosting	0.56757	0.613085	0.591998	0.613085	0.622076	0.534733	0.68983	0.576989

NOT SO FAST

MODEL

PROBLEMS

PROBLEMS WITH OUR MODEL

NOT SO FAST

01 MMA TRAINING IS REVOLUTIONIZING

What's important in the UFC now might not be in 3-5 years

02 RELIES ON THE BETTING LINE

Model will struggle finding underdogs due to odd importance

03 ODDS WERE THE CLOSED ODDS

Odds change drastically, and these were just recorded at start of the fight

04 BETTING STRATEGY NOT OPTIMIZED

Betting $100 on every fight isn't optimized

OH SH*T

MOMENTS ALWAYS EXIST

BET AT YOUR OWN RISK!

ETHICAL CONCERNS

- Small sample size, and only in one promotion
- These models are made to profit, but it's not all about that
- Models should be used to increase performance and target specific training
- Can be used to train for **self defense** after seeing feature importance in professional MMA

CONCLUSION

- Lots of files → requires easy automation
- No models are perfect… (don't trust us too much)
- Taking additionally steps along the way to validate decisions

- **IME 372 is a cool class!**

ANY QUESTIONS?

Michael Cassetti + Shayan Ruhany