UFC Predictions

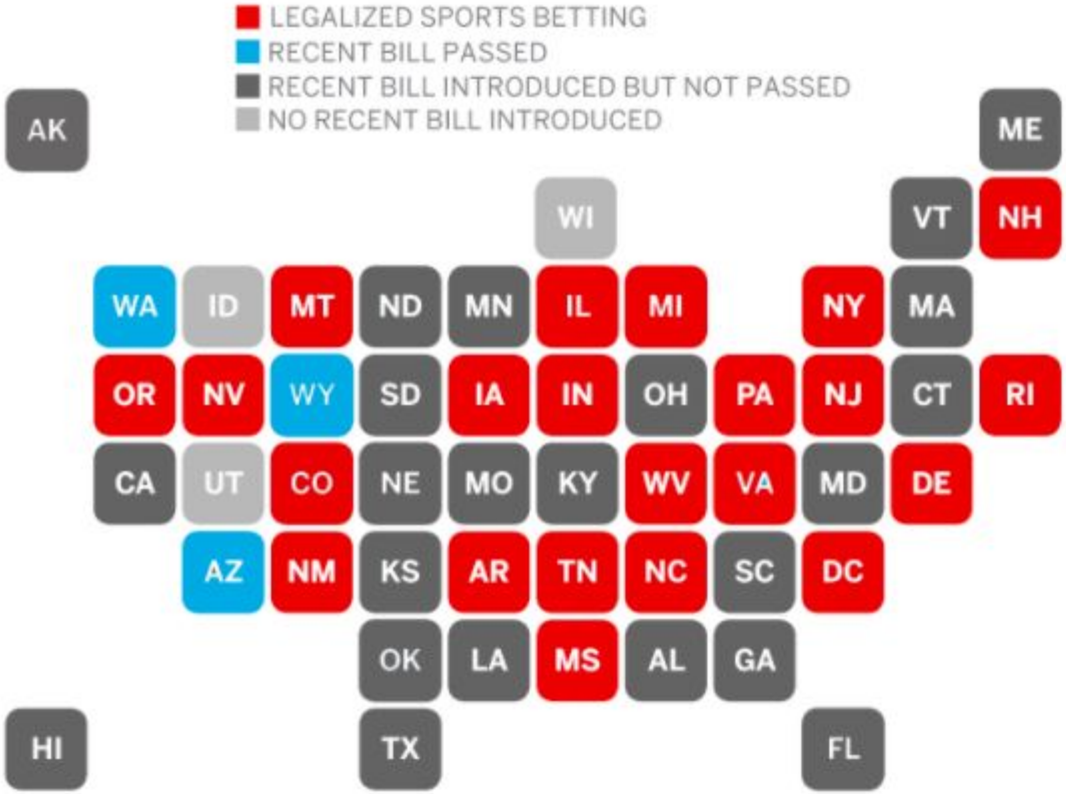
TJ Liggett, Madden Pikula, Ryan Wesp, Trevor Gibson

Purpose

- Sports Betting Popularity: $21 billion spent in 2020 alone

- UFC in 1993: Sports betting has increased since it’s beginning

- More US States are legalizing sports betting every year


🟥 LEGALIZED SPORTS BETTING
🟦 RECENT BILL PASSED
⬛ RECENT BILL INTRODUCED BUT NOT PASSED
⬜ NO RECENT BILL INTRODUCED
AK
ME
WI
VT
NH
WA
ID
MT
ND
MN
IL
MI
NY
MA
OR
NV
WY
SD
IA
IN
OH
PA
NJ
CT
RI
CA
UT
CO
NE
MO
KY
WV
VA
MD
DE
AZ
NM
KS
AR
TN
NC
SC
DC
OK
LA
MS
AL
GA
HI
TX
FL

Purpose: So what?

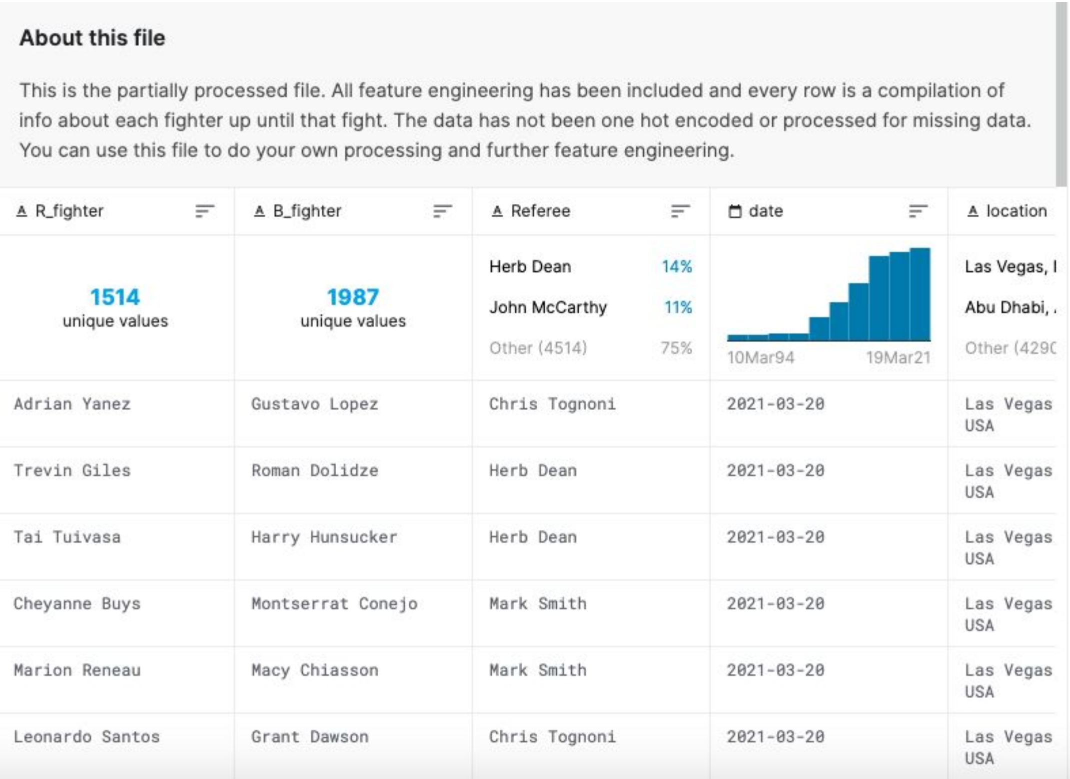
- Helps understand how odds are created

- Looking to understand the most important factors in UFC fights

- Shows prevalence of predictive analytics

Data Description

- Kaggle.com
- UFC-Fight historical data from 1993 to 2021
- Contains info on every UFC fight from 1993 to March 3, 2021
- 144 Total Columns of data
    - Ex: Age, Strikes landed, Stance, Referee, Strikes Attempted, etc.

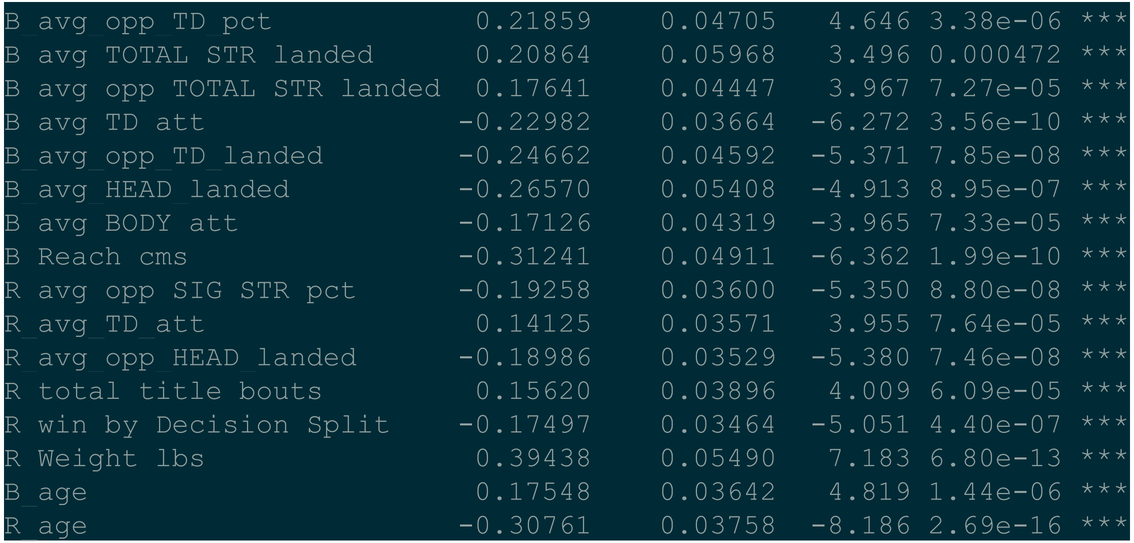


About this file

This is the partially processed file. All feature engineering has been included and every row is a compilation of info about each fighter up until that fight. The data has not been one hot encoded or processed for missing data. You can use this file to do your own processing and further feature engineering.

R_fighter	B_fighter	Referee	date	location
1514 unique values	1987 unique values	Herb Dean 14% John McCarthy 11% Other (4514) 75%	10Mar94 19Mar21	Las Vegas, I Abu Dhabi, . Other (429(
Adrian Yanez	Gustavo Lopez	Chris Tognoni	2021-03-20	Las Vegas USA
Trevin Giles	Roman Dolidze	Herb Dean	2021-03-20	Las Vegas USA
Tai Tuivasa	Harry Hunsucker	Herb Dean	2021-03-20	Las Vegas USA
Cheyanne Buys	Montserrat Conejo	Mark Smith	2021-03-20	Las Vegas USA
Marion Reneau	Macy Chiasson	Mark Smith	2021-03-20	Las Vegas USA
Leonardo Santos	Grant Dawson	Chris Tognoni	2021-03-20	Las Vegas USA

Significant Variables

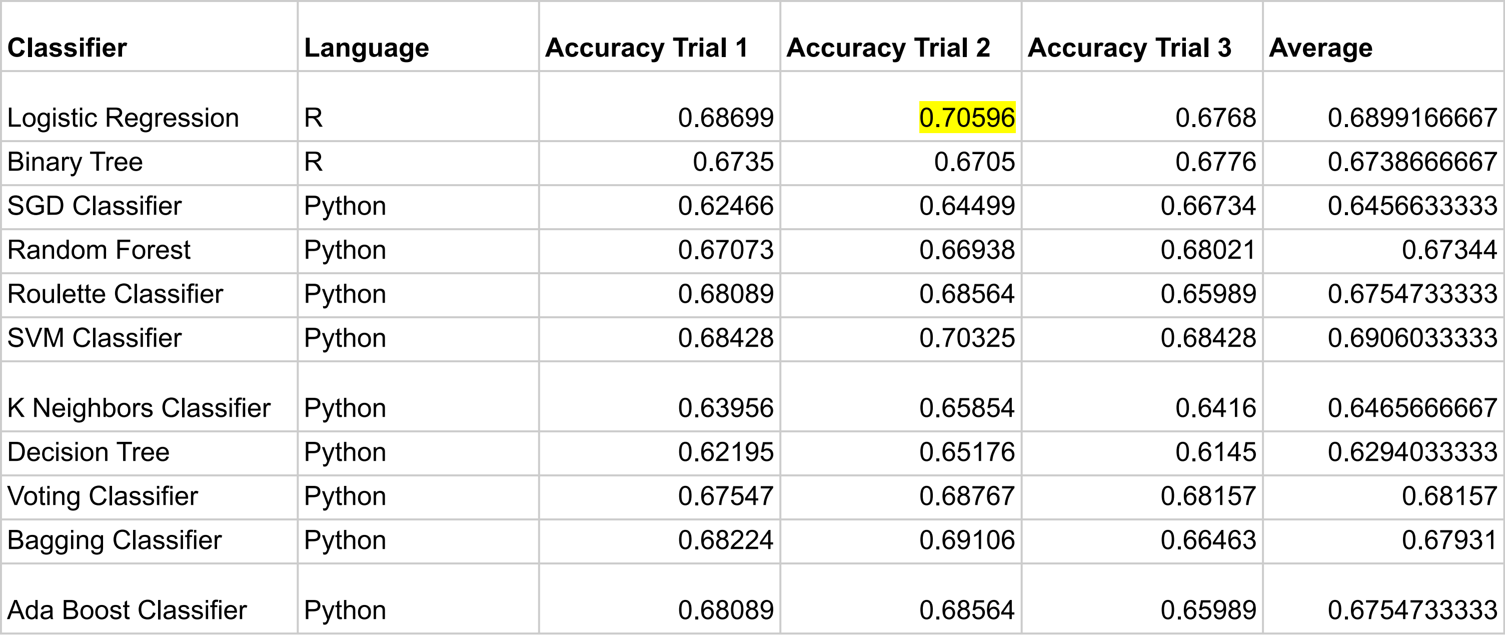


```
B_avg_opp_TD_pct            0.21859    0.04705   4.646 3.38e-06 ***
B avg TOTAL STR landed      0.20864    0.05968   3.496 0.000472 ***
B avg opp TOTAL STR landed  0.17641    0.04447   3.967 7.27e-05 ***
B avg TD att               -0.22982    0.03664  -6.272 3.56e-10 ***
B_avg_opp_TD_landed        -0.24662    0.04592  -5.371 7.85e-08 ***
B_avg_HEAD_landed          -0.26570    0.05408  -4.913 8.95e-07 ***
B avg BODY att             -0.17126    0.04319  -3.965 7.33e-05 ***
B Reach cms                -0.31241    0.04911  -6.362 1.99e-10 ***
R avg opp SIG STR pct      -0.19258    0.03600  -5.350 8.80e-08 ***
R_avg_TD_att                0.14125    0.03571   3.955 7.64e-05 ***
R_avg_opp_HEAD_landed      -0.18986    0.03529  -5.380 7.46e-08 ***
R total title bouts         0.15620    0.03896   4.009 6.09e-05 ***
R win by Decision Split    -0.17497    0.03464  -5.051 4.40e-07 ***
R Weight lbs                0.39438    0.05490   7.183 6.80e-13 ***
B_age                       0.17548    0.03642   4.819 1.44e-06 ***
R_age                      -0.30761    0.03758  -8.186 2.69e-16 ***
```

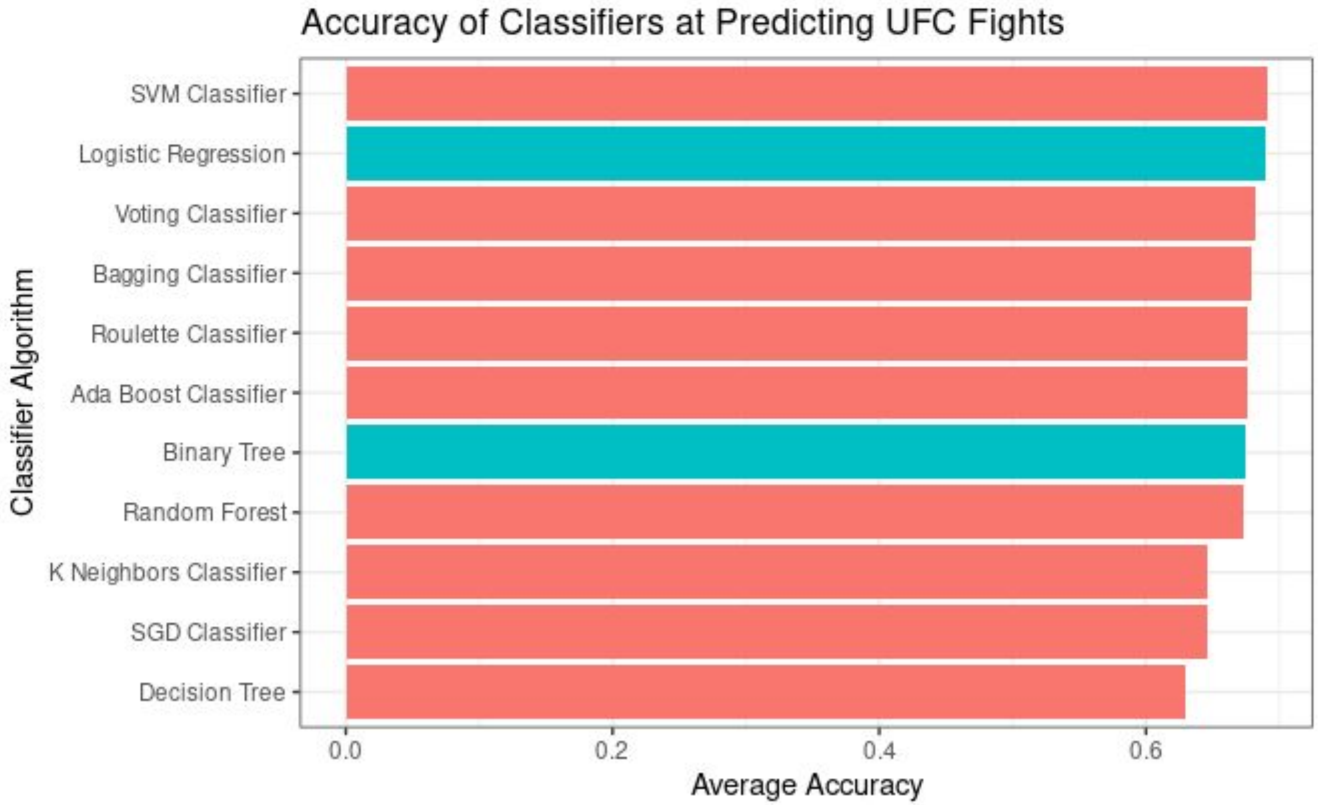
What type of Model/Methods

- R Studio
  - Logistic Regression
- Jupyter Notebook
  - Stochastic Gradient Descent Classifier
  - Random Forest Classifier
  - Support Vector Machine Classifier
  - KNeighborsClassifier
  - Decision Tree Classifier
  - Voting Classifier
  - Bagging Classifier
  - Ada Boost Classifier

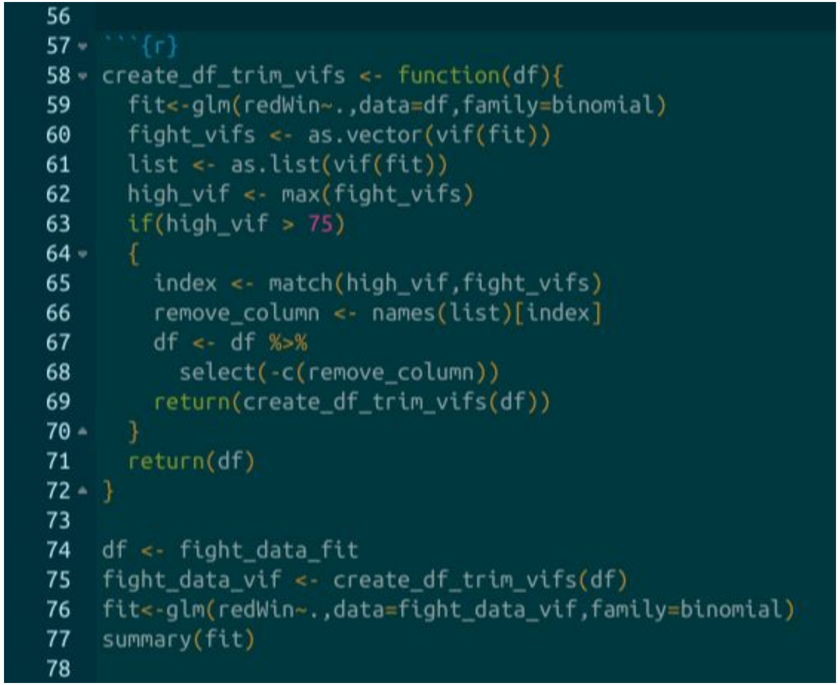
Results



Classifier	Language	Accuracy Trial 1	Accuracy Trial 2	Accuracy Trial 3	Average
Logistic Regression	R	0.68699	0.70596	0.6768	0.6899166667
Binary Tree	R	0.6735	0.6705	0.6776	0.6738666667
SGD Classifier	Python	0.62466	0.64499	0.66734	0.6456633333
Random Forest	Python	0.67073	0.66938	0.68021	0.67344
Roulette Classifier	Python	0.68089	0.68564	0.65989	0.6754733333
SVM Classifier	Python	0.68428	0.70325	0.68428	0.6906033333
K Neighbors Classifier	Python	0.63956	0.65854	0.6416	0.6465666667
Decision Tree	Python	0.62195	0.65176	0.6145	0.6294033333
Voting Classifier	Python	0.67547	0.68767	0.68157	0.68157
Bagging Classifier	Python	0.68224	0.69106	0.66463	0.67931
Ada Boost Classifier	Python	0.68089	0.68564	0.65989	0.6754733333


Accuracy of Classifiers at Predicting UFC Fights
SVM Classifier
Logistic Regression
Voting Classifier
Bagging Classifier
Roulette Classifier
Ada Boost Classifier
Binary Tree
Random Forest
K Neighbors Classifier
SGD Classifier
Decision Tree
Classifier Algorithm
0.0
0.2
0.4
0.6
Average Accuracy

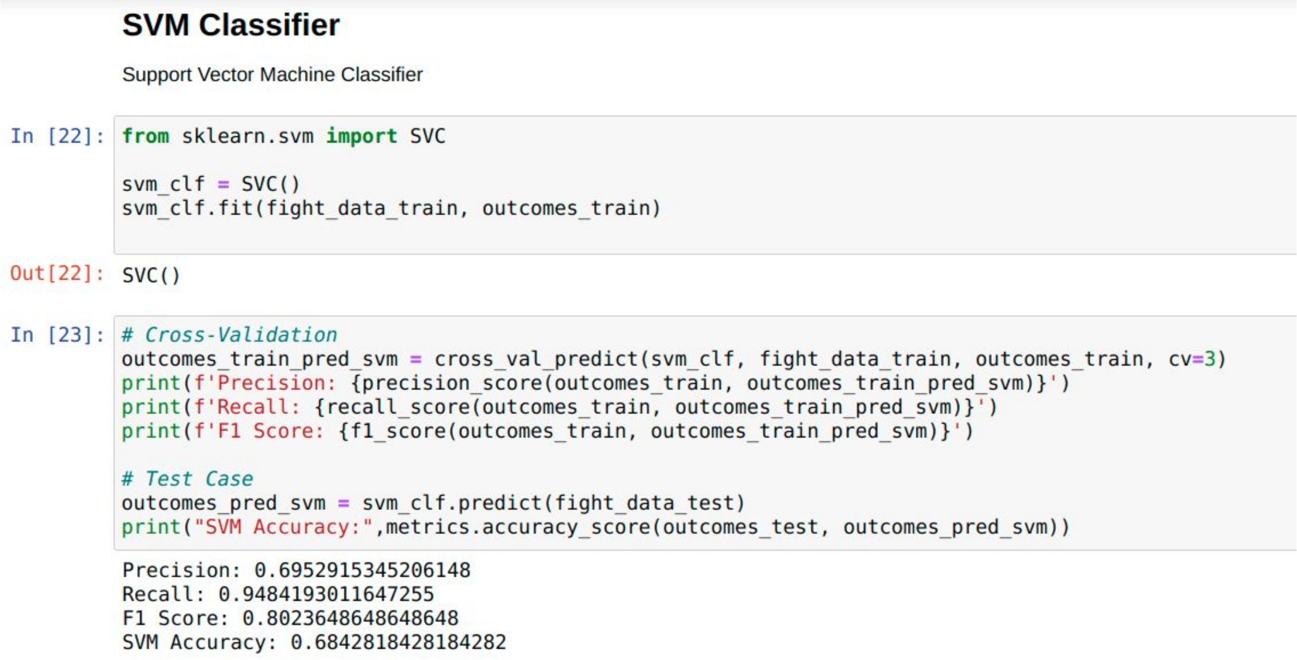
Code!



~~~r
```{r}
create_df_trim_vifs <- function(df){
  fit<-glm(redWin~.,data=df,family=binomial)
  fight_vifs <- as.vector(vif(fit))
  list <- as.list(vif(fit))
  high_vif <- max(fight_vifs)
  if(high_vif > 75)
  {
    index <- match(high_vif,fight_vifs)
    remove_column <- names(list)[index]
    df <- df %>%
      select(-c(remove_column))
    return(create_df_trim_vifs(df))
  }
  return(df)
}

df <- fight_data_fit
fight_data_vif <- create_df_trim_vifs(df)
fit<-glm(redWin~.,data=fight_data_vif,family=binomial)
summary(fit)
~~~

Code!



SVM Classifier

Support Vector Machine Classifier

```
In [22]: from sklearn.svm import SVC

         svm_clf = SVC()
         svm_clf.fit(fight_data_train, outcomes_train)
```

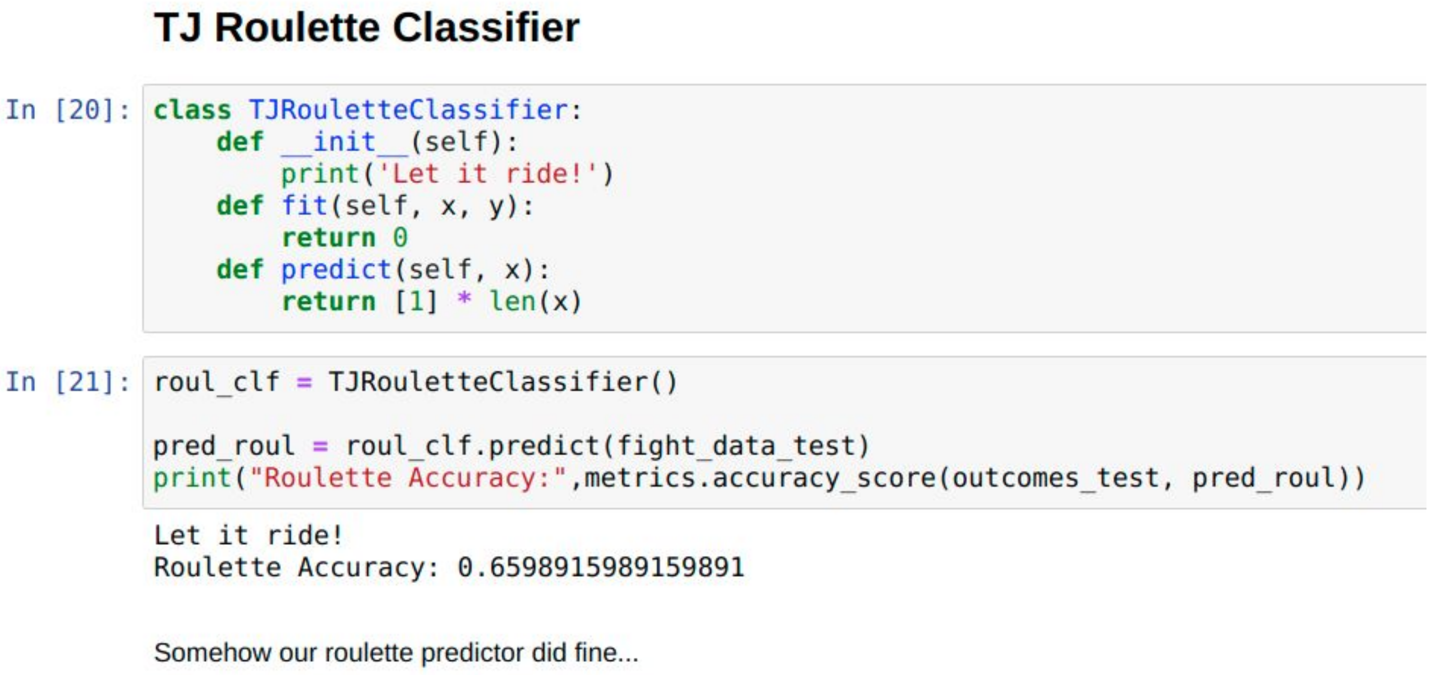
```
Out[22]: SVC()
```

```
In [23]: # Cross-Validation
         outcomes_train_pred_svm = cross_val_predict(svm_clf, fight_data_train, outcomes_train, cv=3)
         print(f'Precision: {precision_score(outcomes_train, outcomes_train_pred_svm)}')
         print(f'Recall: {recall_score(outcomes_train, outcomes_train_pred_svm)}')
         print(f'F1 Score: {f1_score(outcomes_train, outcomes_train_pred_svm)}')

         # Test Case
         outcomes_pred_svm = svm_clf.predict(fight_data_test)
         print("SVM Accuracy:",metrics.accuracy_score(outcomes_test, outcomes_pred_svm))
```

```
Precision: 0.6952915345206148
Recall: 0.9484193011647255
F1 Score: 0.8023648648648648
SVM Accuracy: 0.6842818428184282
```

Code!



TJ Roulette Classifier

```
In [20]: class TJRouletteClassifier:
             def __init__(self):
                 print('Let it ride!')
             def fit(self, x, y):
                 return 0
             def predict(self, x):
                 return [1] * len(x)
```

```
In [21]: roul_clf = TJRouletteClassifier()

         pred_roul = roul_clf.predict(fight_data_test)
         print("Roulette Accuracy:",metrics.accuracy_score(outcomes_test, pred_roul))
```

```
Let it ride!
Roulette Accuracy: 0.6598915989159891
```

Somehow our roulette predictor did fine...

Problems Faced

- Algorithm Runtimes
    - Rpart doesn’t scale well to more variables
    - AdaBoost takes forever

Conclusion

- Predicting UFC Fights is Hard!
- Algorithms in R and Python are comparable, nice to know both
- Always bet red!

Questions?