Academy Awards Prediction

Anton Martin

The Question

Framing the Question

Which movie would will win an Academy Award?

Which nominee would will win an Academy Award?

Which nominee would will win Academy Award for Best Picture?

Data Extraction

How winner is chosen

- Voting System
- Active members of the academy vote
  - Current or retired industry professionals
  - Past nominees or recommended by their peers

- Decide on features which may influence how they vote

- Runtime
- Genre
- IMDB rating
- MPAA Rating
- Release Month
- Pre nomination Box Office Gross
- Studio
- Total nominations
- Director
- Win / Lose

How data was collected - 1990 to 2018

- Existing Datasets Online
- IMDb API
    - IMDbPY
- Web scraping
    - Box Office Mojo

```
for name in oscars.YN:
    try:
        x = ia.search_movie(name)[0]
        ia.update(x)
        print(x['runtime'] , '/', x['rating'] , '/', x['title'])
    except:
        print('1')
```

```
for name in oscars.YN:
    try:
        x = ia.search_movie(name)[0]
        ia.update(x)
        print(x.movieID)
    except:
        print('1')
```

```
for x in oscars.ID:
    try:
        y = ia.get_movie(x)
        ia.update(y)
        print(y['title'], '/' , y['year'], '/' , y['runtime'] , '/', y['rating'] , '/' , y['genre'], '/' , y['certification'])
    except:
        print('1')
```

Box Office Mojo

IMDbPro

Search Site

Search...

Features
News
Release Sched.
Showtimes

Box Office
Daily
Weekend
Weekly
Monthly
Quarterly
Seasonal
Yearly
All Time
International

Indices
Studios
People
Genres
Franchises
Showdowns
Theater Counts

Academy Awards > 2018

2018 – 2017 – 2016 – 2015 – 2014 – 2013 – 2012 – 2011 – 2010 – 2009 – 2008 – 2007 – 2006 – 2005 – 2004 – 2003 – 2002 – 2001 – Full Index

By Category | By Movie | By Studio | Picture Detail | Actor Detail | Actress Detail

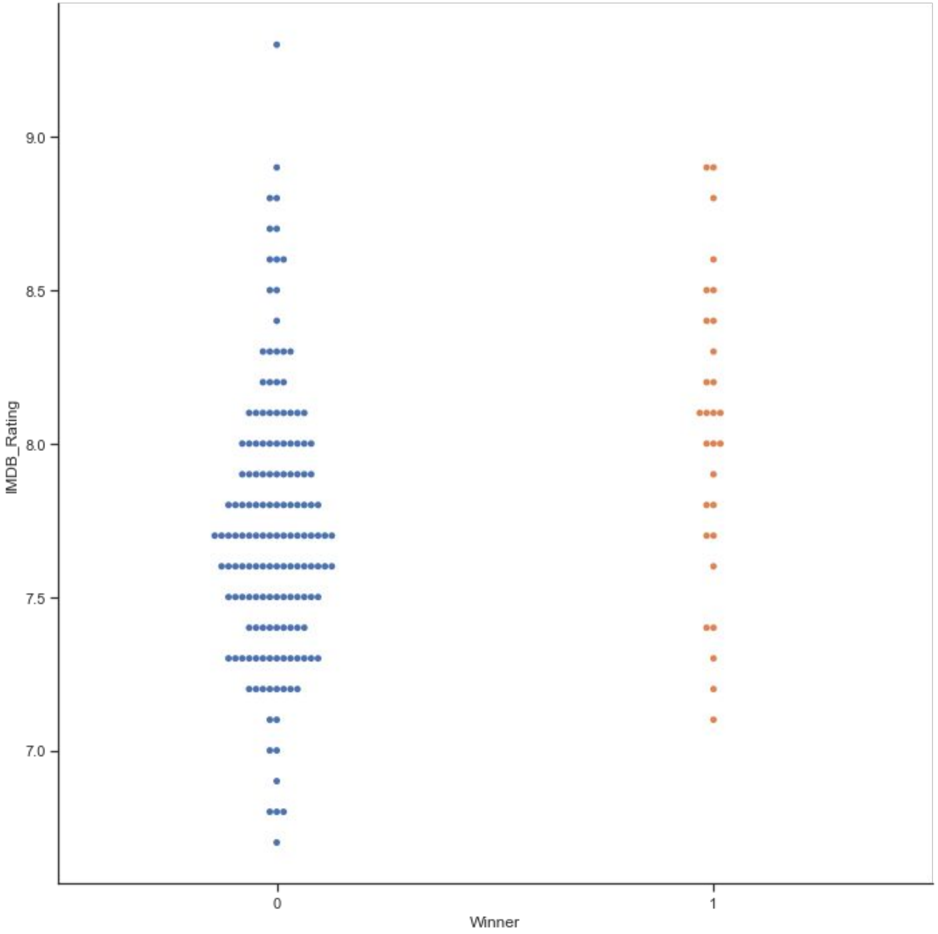
BEST PICTURE

YR Rank	Nominated Movie	Studio	Pre-Nom. (up to 1/22) Gross / Theaters*		Post-Nom. (1/22-2/24) Gross / Theaters*		Post-Awards (2/24+) Gross / Theaters*		Total Gross	Release Date
1	Black Panther	BV	$700,059,566	4,084	-	-	-	-	$700,059,566	2/16
10	Bohemian Rhapsody	Fox	$202,518,098	4,000	$10,599,927	1,423	$3,000,073	839	$216,118,098	11/02
11	A Star is Born (2018)	WB	$204,801,596	3,904	$6,041,750	1,192	$4,442,273	1,235	$215,285,619	10/05
36	Green Book	Uni.	$42,470,556	1,215	$27,175,145	2,648	$14,921,455	2,641	$84,567,156	11/16
58	BlacKkKlansman	Focus	$48,468,750	1,914	$806,590	171	-	-	$49,275,340	8/10
60	Vice	Annapurna	$39,528,524	2,534	$7,587,882	1,557	$716,908	702	$47,833,314	12/25
81	The Favourite	FoxS	$22,990,563	810	$9,134,242	1,554	$2,086,639	742	$34,211,444	11/23
		Total Gross:	$1,260,837,653		$61,345,536		$25,167,348		$1,347,350,537	
		Average Gross:	$180,119,665		$8,763,648		$3,595,335		$192,478,648	

Feature Selection / Engineering

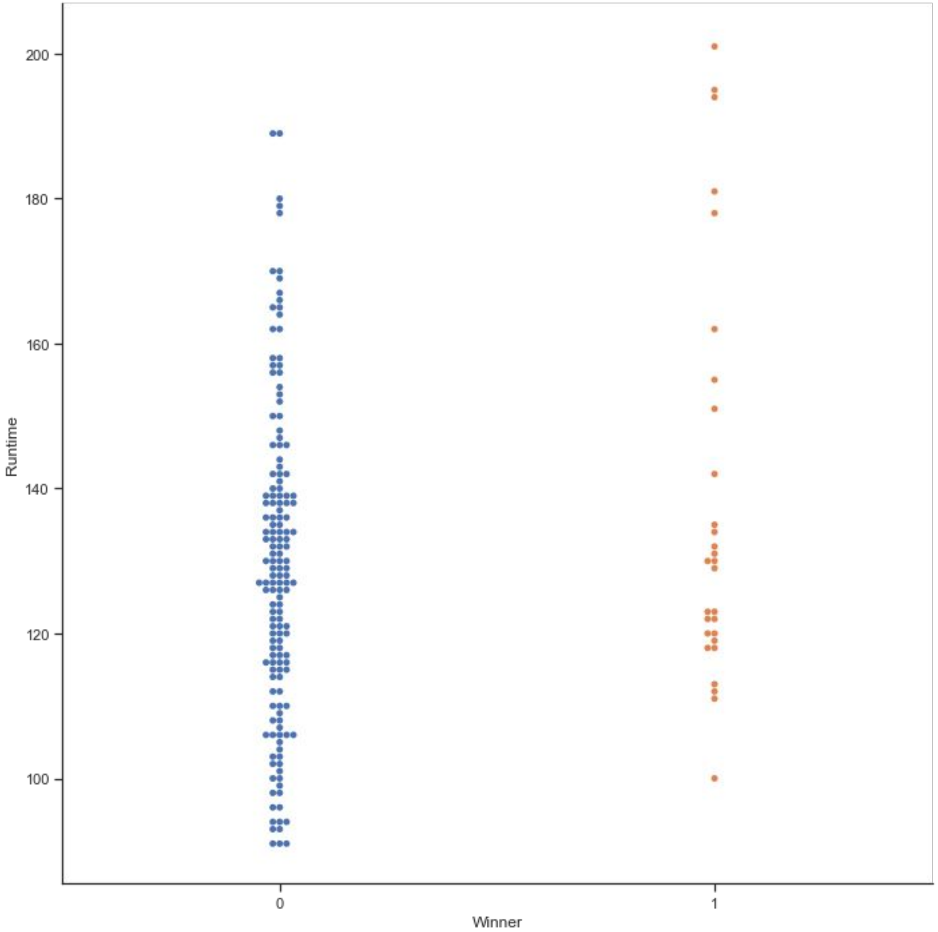
IMDb Rating

- No visible distinction
- All the movies nominated tend to be highly rated



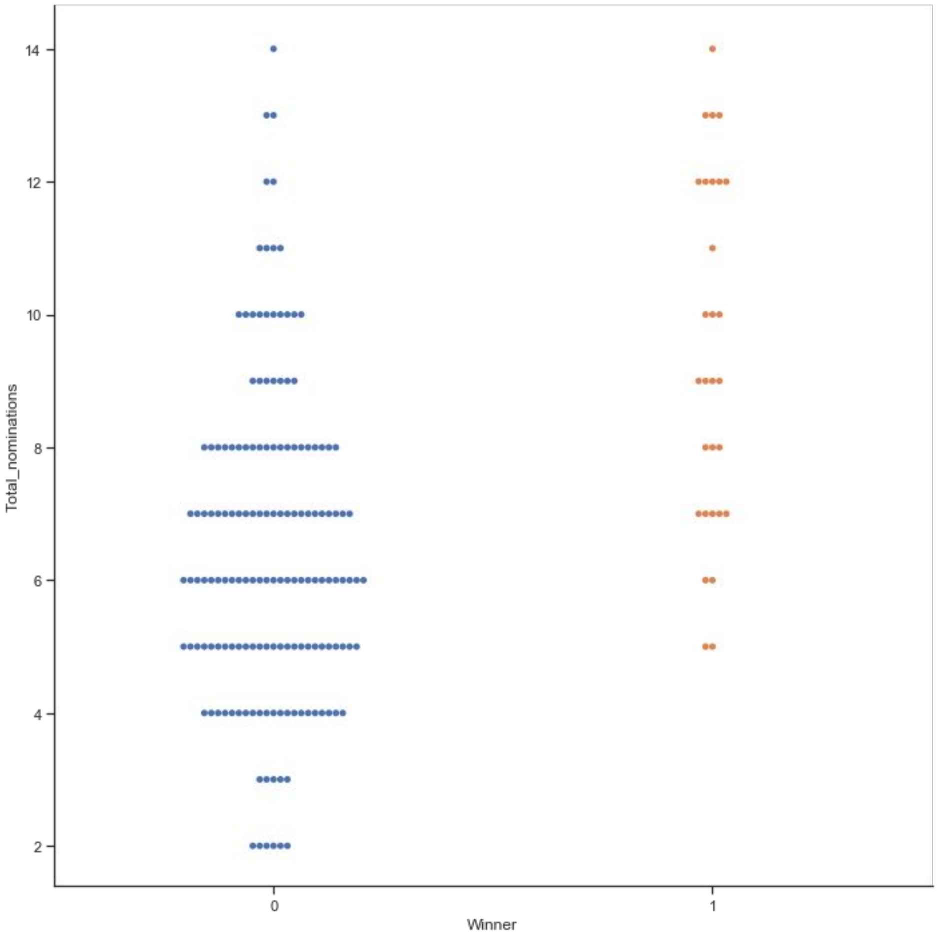
Runtime

- Wide spread
- Winners runtime longer than 100 min



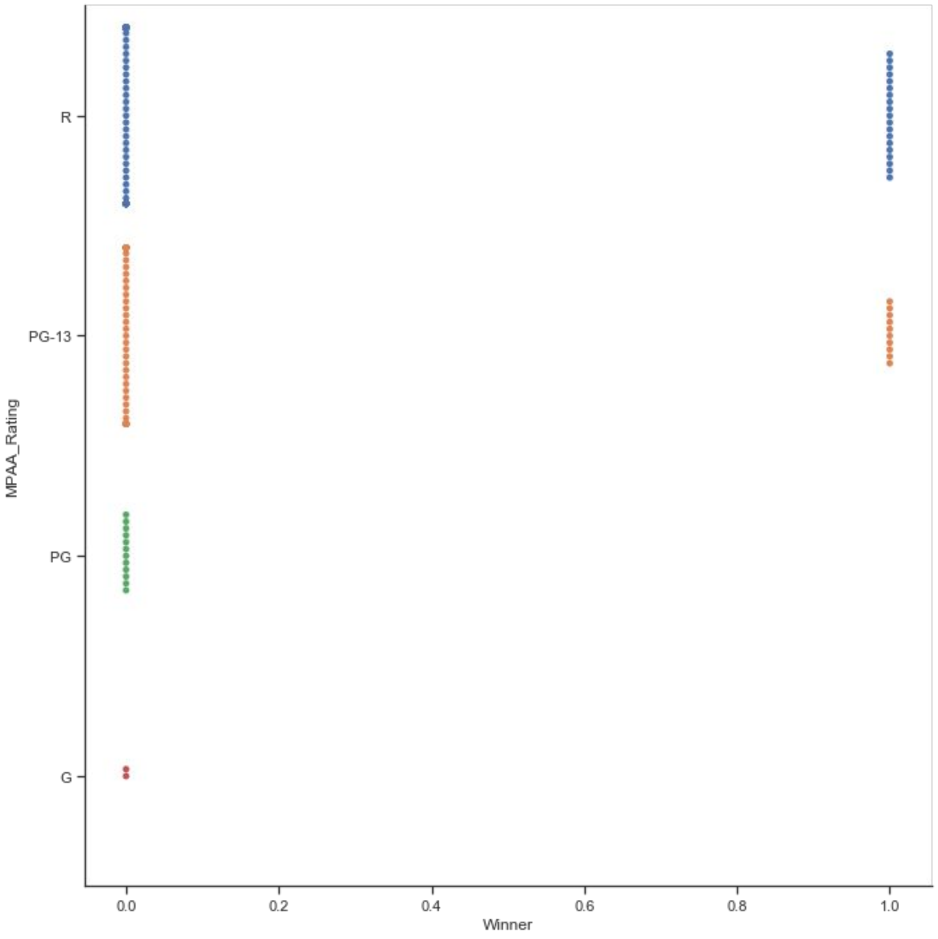
Total nominations

- Winners appear to have >4 nominations



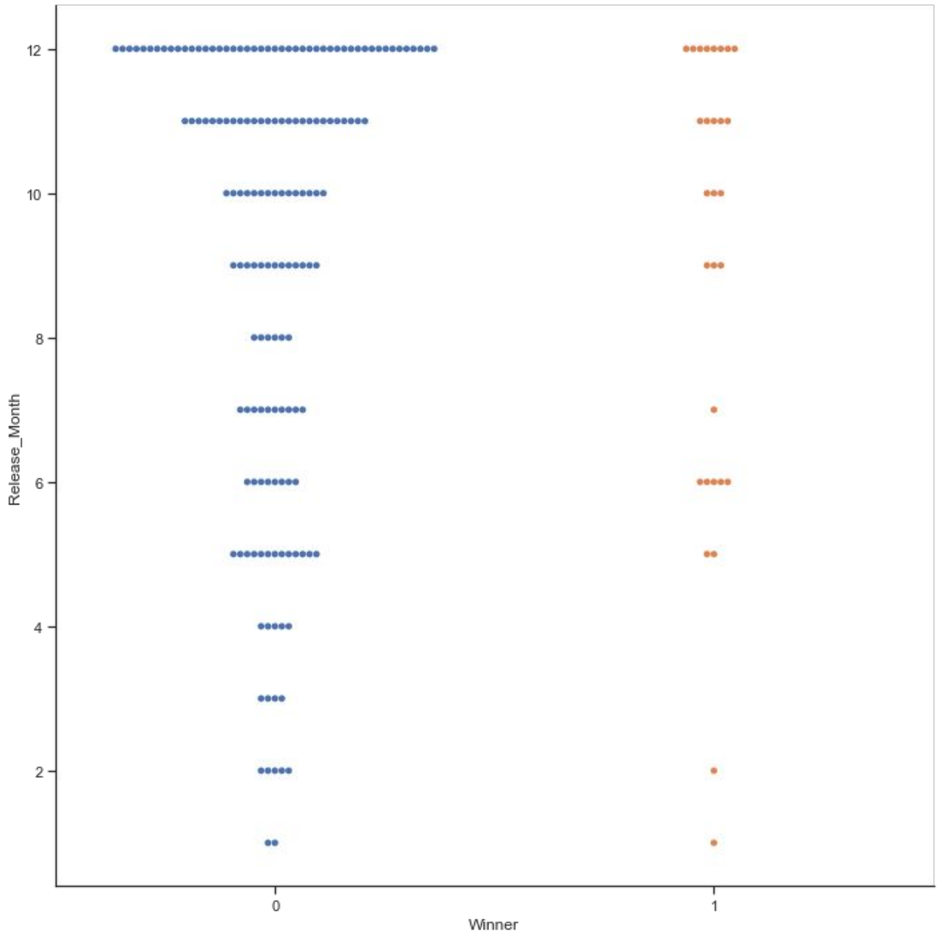
MPAA Rating

- PG and G movies don’t appear to win best picture



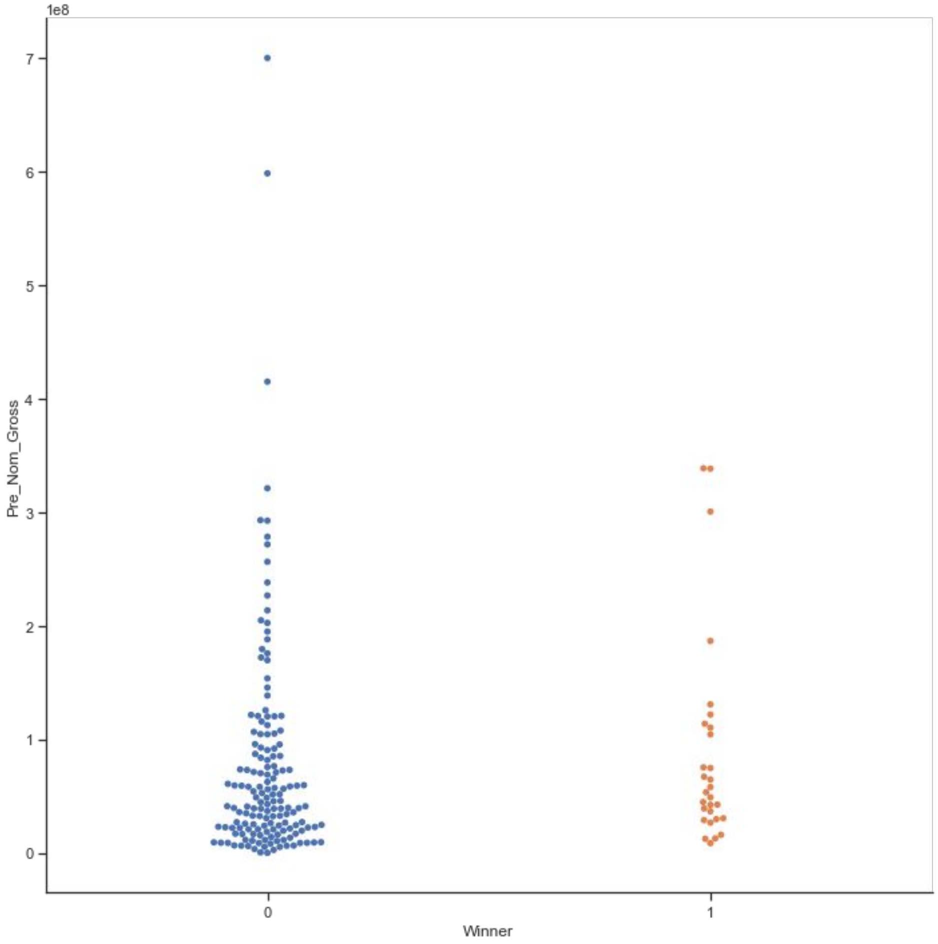
Release Month

- Later in the year have more nominations and wins

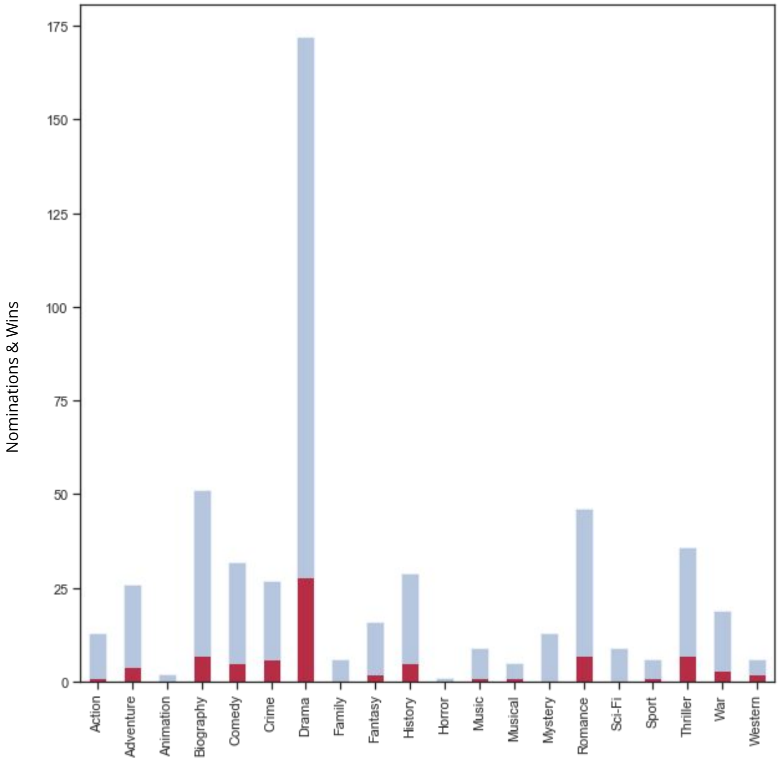


Pre Nomination Gross

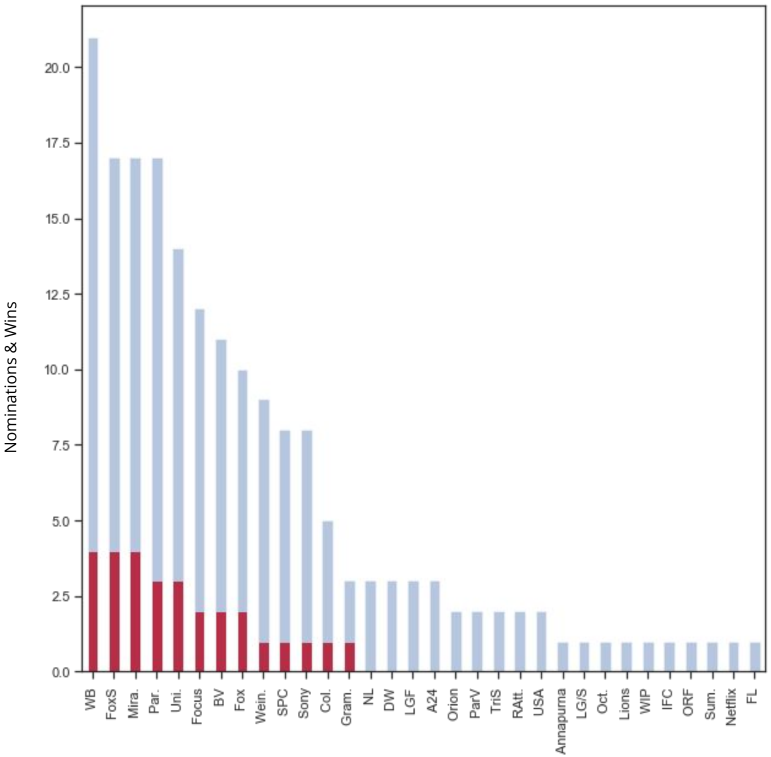
- No visible trend



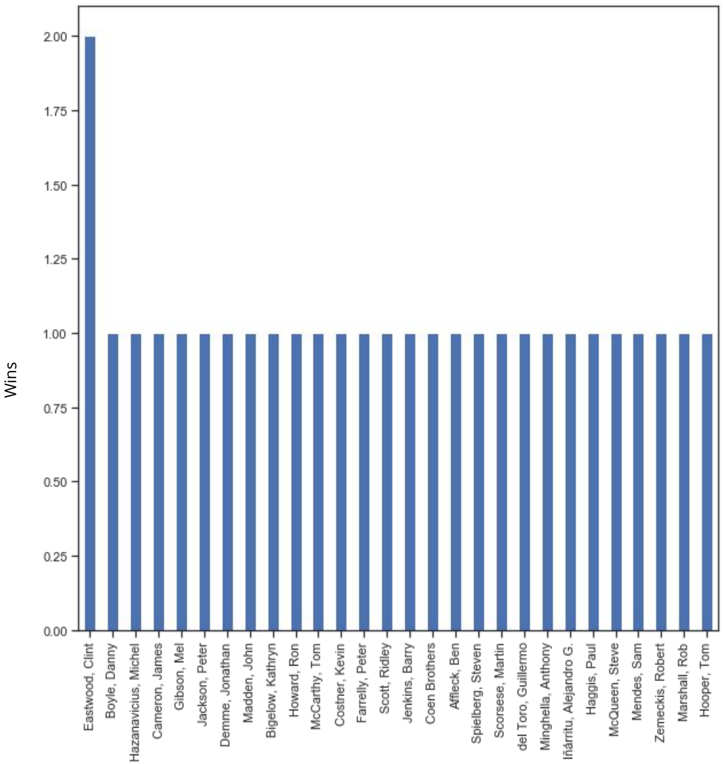
Genre



Studio



Director




```
Spielberg, Steven                    7
Scorsese, Martin                     6
Eastwood, Clint                      5
Lee, Ang                             4
Coen Brothers                        4
Daldry, Stephen                      3
Iñárritu, Alejandro G.               3
Russell, David O.                    3
Jackson, Peter                       3
Howard, Ron                          3
Payne, Alexander                     3
Tarantino, Quentin                   3
Van Sant, Gus                        2
Bigelow, Kathryn                     2
Frears, Stephen                      2
Wright, Joe                          2
Darabont, Frank                      2
Hooper, Tom                          2
Reitman, Jason                       2
Cuarón, Alfonso                      2
Malick, Terrence                     2
Anderson, Paul Thomas                2
Cameron, James                       2
Boyle, Danny                         2
Ivory, James                         2
Hallström, Lasse                     2
McKay, Adam                          2
Scott, Ridley                        2
Chazelle, Damien                     2
Gibson, Mel                          2
                                    ..
Hackford, Taylor                     1
Melfi, Theodore                      1
Altman, Robert                       1
Radford, Michael & Troisi, Massimo   1
Zemeckis, Robert                     1
Zeitlin, Benh                        1
Levinson, Barry                      1
Marshall, Penny                      1
Redford, Robert                      1
Haggis, Paul                         1
Villeneuve, Denis                    1
Cattaneo, Peter                      1
Zucker, Jerry                        1
Daniels, Lee                         1
Leigh, Mike                          1
McDonagh, Martin                     1
Affleck, Ben                         1
Mendes, Sam                          1
Brooks, James L.                     1
Benigni, Roberto                     1
Forster, Marc                        1
Cholodenko, Lisa                     1
Polanski, Roman                      1
Guadagnino, Luca                     1
Campion, Jane                        1
Brest, Martin                        1
Docter, Pete                         1
Cooper, Bradley                      1
Anderson, Wes                        1
Singer, Bryan                        1
Name: Director, Length: 127, dtype: int64
```

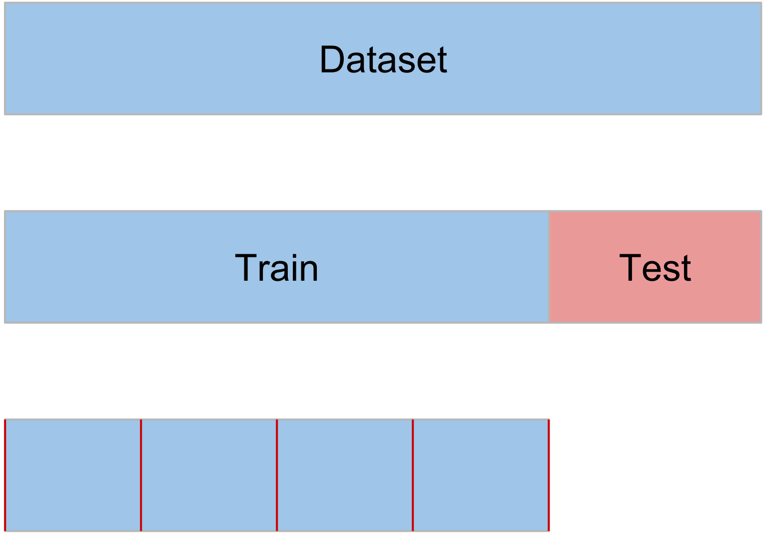
Feature Engineering

- Runtime
- Genre (20)
- MPAA Rating
- Release Month
- Studio (32)
- Total nominations
- Win / Lose

- Dummy variables / one-hot encoding
    - Genre
    - Studio
- Release month → Time to nomination

Building the Model

Three Way Data Split



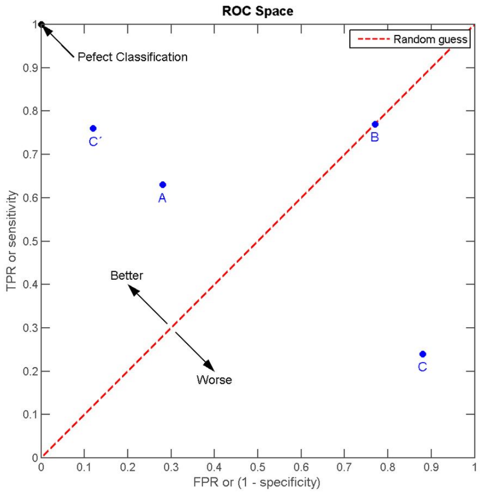
- Split Dataset into Train / Test
- Cross Validation on Train
- Keep Test as holdout set
    -

Metrics

- Accuracy
- Confusion Matrix
- ROC AUC
- Precision
- Recall
- F1 Score

	Predicted No	Predicted Yes
Actual No	True Negative	False Positive
Actual Yes	False Negative	True Positive

Accuracy = TP+TN/TP+FP+FN+TN
Precision = TP/TP+FP
Recall = TP/TP+FN
F1 Score = 2*(Recall * Precision) / (Recall + Precision)



Logistic Regression - All Features / Default Hyperparameters

```python
n = 0
accuracy = []
roc = []
matrix = []

print("~~~~ CROSS VALIDATION each fold ~~~~")
for train_index, test_index in kf.split(X_train, y_train):
    lr = LogisticRegression(solver = 'lbfgs').fit(X_train.iloc[train_index], y_train.iloc[train_index])

    prediction = lr.predict(X_train.iloc[test_index])
    actual = y_train.iloc[test_index]
    accuracy.append(np.mean(y_train.iloc[test_index] == lr.predict(X_train.iloc[test_index])))
    lr_pred_proba = lr.predict_proba(X_train.iloc[test_index])[:,1]
    matrix.append(metrics.confusion_matrix(y_true=y_train.iloc[test_index], y_pred=lr_pred_proba > .5))
    roc.append(metrics.roc_auc_score(y_true=y_train.iloc[test_index], y_score=lr_pred_proba> .5))

    n += 1

    print('Model {}'.format(n))

    print('Accuracy: {}' .format(accuracy[n-1]))
    print(prediction)
    print(matrix[n-1])
    print('ROC AUC: {}'.format(roc[n-1]))
    print('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')



print("~~~~ SUMMARY OF CROSS VALIDATION ~~~~")
print('Mean of accuracy for all folds : {} '.format(np.mean(accuracy)))
print('Mean of ROC AUC: {}'.format(np.mean(roc)))
d =(np.sum(matrix,0))/4
print(d)
```

```
~~~~ CROSS VALIDATION each fold ~~~~
Model 1
Accuracy: 0.8378378378378378
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[[31  0]
 [ 6  0]]
ROC AUC: 0.5
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Model 2
Accuracy: 0.8378378378378378
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[[31  0]
 [ 6  0]]
ROC AUC: 0.5
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Model 3
Accuracy: 0.8378378378378378
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[[31  0]
 [ 6  0]]
ROC AUC: 0.5
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Model 4
Accuracy: 0.8611111111111112
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[[31  0]
 [ 5  0]]
ROC AUC: 0.5
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~ SUMMARY OF CROSS VALIDATION ~~~~
Mean of accuracy for all folds : 0.8436561561561562
Mean of ROC AUC: 0.5
[[31.    0.  ]
 [ 5.75  0.  ]]
```

Logistic Regression - Selected Features / Tuned Hyperparameters

```
n = 0
accuracy = []
a = 0.5
roc = []
matrix = []

print("~~~~ CROSS VALIDATION each fold ~~~~")
for train_index, test_index in kf.split(X_train, y_train):

    lr = LogisticRegression(solver = 'lbfgs' , max_iter = 10000, class_weight='balanced' , C =0.45).fit(

    prediction = lr.predict(X_train.iloc[test_index])
    actual = y_train.iloc[test_index]
    accuracy.append(np.mean(y_train.iloc[test_index] == lr.predict(X_train.iloc[test_index])))
    lr_pred_proba = lr.predict_proba(X_train.iloc[test_index])[:,1]
    matrix.append(metrics.confusion_matrix(y_true=y_train.iloc[test_index], y_pred=lr_pred_proba > a))
    roc.append(metrics.roc_auc_score(y_true=y_train.iloc[test_index], y_score=lr_pred_proba> a))

    n += 1

    print('Model {}'.format(n))
    print('Accuracy: {}' .format(accuracy[n-1]))
    print(prediction)
    print((matrix[n-1]))
    #print(lr_pred_proba)
    print('ROC AUC: {}'.format(roc[n-1]))
    print('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')



print("~~~~ SUMMARY OF CROSS VALIDATION ~~~~")
print('Mean of accuracy for all folds : {} '.format(np.mean(accuracy)))
print('Mean of ROC AUC: {}'.format(np.mean(roc)))
d =(np.sum(matrix,0))/4
print(d)
ps = (d[1,1]/(d[1,1]+d[0,1]))
rs = (d[1,1]/(d[1,1]+d[1,0]))
f = 2*(ps*rs) / (ps+rs)
print("precision:" , ps)
print('recall:' , rs)
print('f1:',f)
```

```
~~~~ CROSS VALIDATION each fold ~~~~
Model 1
Accuracy: 0.8378378378378378
[1 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0]
[[27  4]
 [ 2  4]]
ROC AUC: 0.7688172043010751
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Model 2
Accuracy: 0.8648648648648649
[1 0 0 0 0 0 1 0 1 0 1 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]
[[27  4]
 [ 1  5]]
ROC AUC: 0.8521505376344086
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Model 3
Accuracy: 0.7567567567567568
[1 0 0 0 1 0 0 0 1 0 0 0 1 1 0 0 1 1 0 0 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
[[24  7]
 [ 2  4]]
ROC AUC: 0.7204301075268815
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Model 4
Accuracy: 0.75
[0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
[[25  6]
 [ 3  2]]
ROC AUC: 0.6032258064516128
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~ SUMMARY OF CROSS VALIDATION ~~~~
Mean of accuracy for all folds : 0.8023648648648649
Mean of ROC AUC: 0.7361559139784944
[[25.75  5.25]
 [ 2.    3.75]]
precision: 0.4166666666666667
recall: 0.6521739130434783
f1: 0.5084745762711865
```

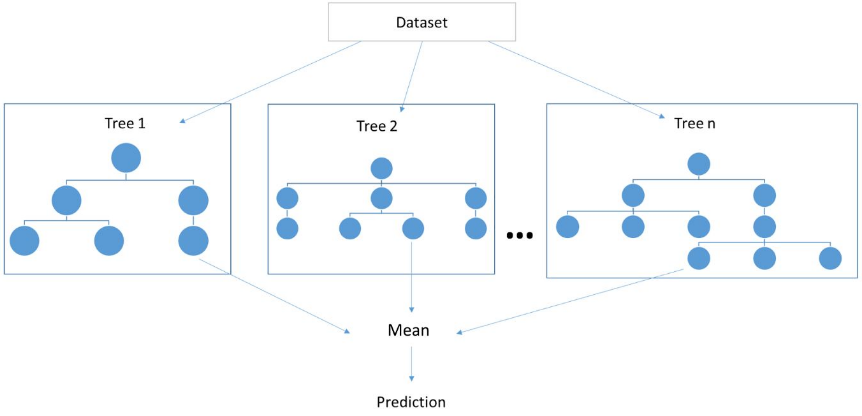
Random Forest

```python
n = 0
accuracy = []
a = 0.50
roc = []
matrix = []


print("~~~~ CROSS VALIDATION each fold ~~~~")
for train_index, test_index in kf.split(X_train, y_train):
    rfc = RandomForestClassifier(n_estimators = 180 ,random_state=1, class_weight='balanced',max_depth = 5,oob_score=True).fit(X

    prediction = rfc.predict(X_train.iloc[test_index])
    actual = y_train.iloc[test_index]
    accuracy.append(np.mean(y_train.iloc[test_index] == rfc.predict(X_train.iloc[test_index])))
    rfc_pred_proba = rfc.predict_proba(X_train.iloc[test_index])[:,1]
    matrix.append(metrics.confusion_matrix(y_true=y_train.iloc[test_index], y_pred=rfc_pred_proba > a))
    roc.append(metrics.roc_auc_score(y_true=y_train.iloc[test_index], y_score=rfc_pred_proba> a))

    n += 1


    print('Model {}'.format(n))
    print('Accuracy: {}' .format(accuracy[n-1]))
    print(prediction)
    print((matrix[n-1]))
    #print(lr_pred_proba)
    print('ROC AUC: {}'.format(roc[n-1]))
    print('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')

print("~~~~ SUMMARY OF CROSS VALIDATION ~~~~")
print('Mean of accuracy for all folds : {} '.format(np.mean(accuracy)))
print('Mean of ROC AUC: {}'.format(np.mean(roc)))
d =(np.sum(matrix,0))/4
print(d)
ps = (d[1,1]/(d[1,1]+d[0,1]))
rs = (d[1,1]/(d[1,1]+d[1,0]))
f = 2*(ps*rs) / (ps+rs)
print(ps)
print(rs)
print(f)
```

```
~~~~ CROSS VALIDATION each fold ~~~~
Model 1
Accuracy: 0.8108108108108109
[0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0]
[[29  2]
 [ 5  1]]
ROC AUC: 0.5510752688172044
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Model 2
Accuracy: 0.8918918918918919
[0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[[31  0]
 [ 4  2]]
ROC AUC: 0.6666666666666666
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Model 3
Accuracy: 0.8378378378378378
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0]
[[29  2]
 [ 4  2]]
ROC AUC: 0.6344086021505376
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Model 4
Accuracy: 0.8055555555555556
[0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
[[29  2]
 [ 5  0]]
ROC AUC: 0.467741935483871
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~ SUMMARY OF CROSS VALIDATION ~~~~
Mean of accuracy for all folds : 0.8365240240240239
Mean of ROC AUC: 0.5799731182795699
[[29.5   1.5 ]
 [ 4.5   1.25]]
precision: 0.45454545454545453
recall: 0.21739130434782608
f1: 0.29411764705882354
```

Logistic Regression vs Random forest

- Too few observations for random forest
- Random forest performs better with larger datasets with many observations and features



Logistic Regression - Test

```
a = 0.5
prediction = lr.predict(X_test)
actual = y_test
accuracy = (np.mean(y_test == lr.predict(X_test)))
lr_pred_proba = lr.predict_proba(X_test)[:,1]
matrix = metrics.confusion_matrix(y_true=y_test, y_pred=lr_pred_proba > a)
roc = (metrics.roc_auc_score(y_true=y_test, y_score=lr_pred_proba> a))

d = matrix
ps = (d[1,1]/(d[1,1]+d[0,1]))
rs = (d[1,1]/(d[1,1]+d[1,0]))
f = 2*(ps*rs) / (ps+rs)

print("Accuracy:" , accuracy)
print(matrix)
print("ROC AUC:" , roc)
print("precision:" , ps)
print('recall:' , rs)
print('f1:',f)
```

```
Accuracy: 0.7297297297297297
[0 1 0 0 0 1 0 1 1 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 1 1 0]
[[23  8]
 [ 2  4]]
ROC AUC: 0.7043010752688172
precision: 0.3333333333333333
recall: 0.6666666666666666
f1: 0.4444444444444444
```

Recap

Summary

1. To build a model that would predict Best Picture winner from nominees
2. Created dataset using existing datasets and IMDb API
3. Selected features to be used by exploring the data

   Created dummy variables for non numerical features
4. Built logistic regression & random forest model

   Selected best model
5. Tested model

Lessons Learnt

Problems Encountered	**Solutions / Potential Improvements**
Uneven distribution of winners in training data & each fold	Use stratified parameter to ensure winners present in each fold
Large number of nominees vs winners	class_weight = balanced
Trade off between false positives & false negatives	Explore thresholds
Tuning of hyperparameters	Use gridsearch for best parameters
Poor performance of Random Forest model	Increase number of observations and features

Other Potential improvements

- Automatically populate features and run model by entering name and year of movie

- Automatically run model on a yearly basis once Academy Award nominees are announced

Acknowledgements

- People
  - Hariharasudhan Balasubramanian / Daniel Tan
  - GA Team
- Resources
  - https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/
  - https://datahub.io/rufuspollock/oscars-nominees-and-winners
  - https://bigml.com/user/academy_awards/gallery/dataset/5c6886e1eba31d73070017f5
  - https://imdbpy.sourceforge.io/
  - https://www.boxofficemojo.com/oscar/