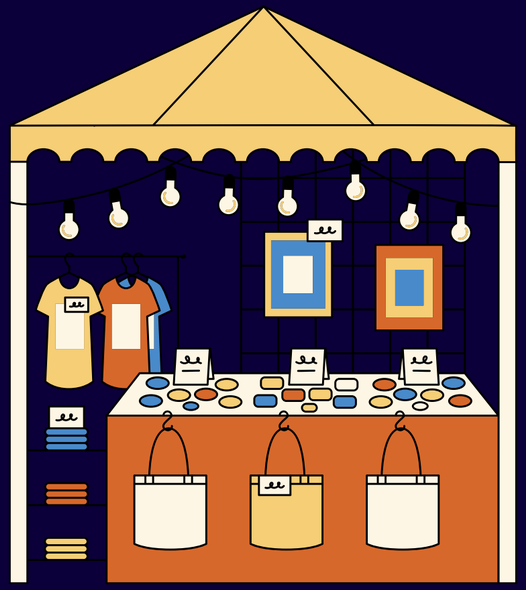
DATA ANALYSIS MACHINE LEARNING

PREDICTING OSCARS WINNER

PHAM THANH THAO - 17 MAR 2023

CONTEXT & OBJECTIVE

IMDB & OSCARS BUSINESS USE CASE



MERCHANDISE & BUDGET PLANNING

BRAND AMBASSADORS | SPONSORSHIP

FUTURE COLLABORATION

COULD WE PREDICT THE OSCARS WINNER FROM IMDB INFORMATION?

PLANNING

1 **DATA PROCESSING**

Python + SQL for descriptive analysis

2 **DATA ANALYSIS**

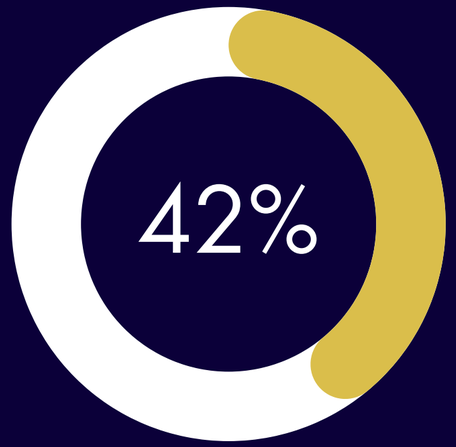
Python + Tableau for explorative data analysis

3 **MACHINE LEARNING**

Classification models for predictive analysis

DATA SOURCES

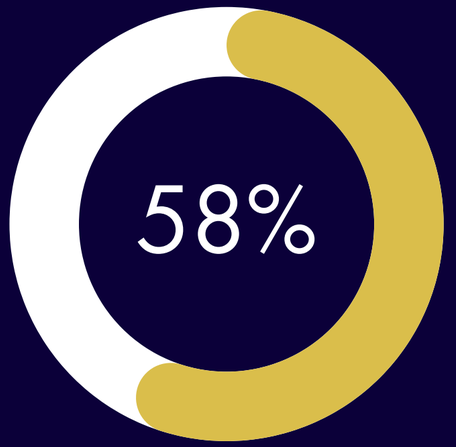
IMDB



Titles - id, primary names, average ratings, total votes count, genres, principle crew

Names - id, name, titles known for

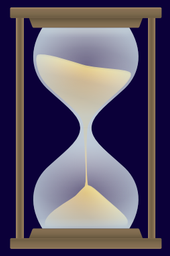
WEB SCRAPING



Nominated - movies & individuals

Winners - movies & individuals

Ratings - by genders & age



DATASETS

FROM IMDB

AVERAGE RATING

	tconst	averageRating	numVotes
0	tt0000001	5.70000	1956
1	tt0000002	5.80000	263

ACTOR NAME

	nconst	primaryName	birthYear	deathYear	primaryProfession	knownForTitles
0	nm0000001	Fred Astaire	1899	1987	soundtrack,actor,miscellaneous	tt0072308,tt0045537,tt0053137,tt0050419
1	nm0000002	Lauren Bacall	1924	2014	actress,soundtrack	tt0037382,tt0071877,tt0038355,tt0117057

TITLE BASICS

	tconst	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	genres
0	tt0000001	short	Carmencita	Carmencita	0	1894	\N	1	Documentary,Short
1	tt0000002	short	Le clown et ses chiens	Le clown et ses chiens	0	1892	\N	5	Animation,Short

TITLE PRINCIPLE CREW

	tconst	ordering	nconst	category	job	characters
0	tt0000001	1	nm1588970	self	\N	["Self"]
1	tt0000001	2	nm0005690	director	\N	\N
2	tt0000001	3	nm0374658	cinematographer	director of photography	\N

FROM WEBSCRAPING

NOMINEE/ WINNER

	tconst	Nominees
2976	tt0454876	TRUE

DATA CLEANING

CLEAN NULL & CHANGE DATA TYPES

```
def clean_N(df, col, replace, type):
    """
    Replace \N value in a column, replace it with a value of choice and reset the type of the column
    """
    df[col] = df[col].replace('\\N', replace)
    df[col] = df[col].astype(type)
    return df[col].value_counts().sort_index()
```

✓ 0.0s

SELECT RELEVANT DATA & NULL REPLACEMENT

```
movie_type = ['movie', 'tvMovie']
title_movies_clean = title_basics[((title_basics['endYear'] < 2023)
                                   | (title_basics['startYear'] > 1926))
                                  & (title_basics['titleType'].isin(movie_type))]
```

✓ 1.5s

```
title_basics['runtimeMinutes'].fillna(0, inplace=True)
title_basics[title_basics['runtimeMinutes'].str.contains('Animation').fillna(False)]
```

✓ 7.2s Python

	tconst	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	genres
2996552	tt13704268	tvEpisode	Bay of the Triffids/Doctor of Doom\tBay of the...	0	\N	1800	2024	Animation,Comedy,Family	NaN

NEW FEATURES

STAR METER

```
-- 3. AVERAGE RATING FOR THE MOVIES A PERSON HAS DONE, WEIGHTED WITH NUMBER OF VOTE COUNTS
CREATE TABLE nconst_ratings
AS
  (SELECT
    nconst,
    (sum(averageRating * numVotes)/ sum(numVotes)) as nconst_rating
  FROM(
    SELECT *, ROW_NUMBER() OVER (PARTITION BY nconst ORDER BY averageRating DESC) rn
    FROM nconst_knownfor_rating) x
  GROUP BY nconst);
-- nconst_rating = average rating of the top 4 titles they are known for, weighted by number of votes

ALTER TABLE nconst_ratings RENAME COLUMN nconst_rating TO rating_of_bestwork;
```

nconst	rating_of_bestwork
nm0000002	7.537273771697488
nm0000003	7.255458823529411
nm0000004	7.585644647710464

```
-- 4. SCORE OF EACH TITLE BASED ON THE RATING OF THE MAIN CREW (NCONST_RATING)
CREATE TABLE tconst_ratings_by_crew
AS (
SELECT tpc.tconst, avg(nconst_rating)
FROM title_principle_crew tpc
LEFT JOIN nconst_ratings nr
ON tpc.nconst = nr.nconst
GROUP BY tconst);

ALTER TABLE tconst_ratings_by_crew RENAME COLUMN `avg(nconst_rating)` TO rating_of_crew;
```

tconst	rating_of_crew
tt0017534	7.443602872263254

NUMBER OF NOMINATED CREW

tconst	ct_nom_crew
tt0003854	0
tt0011715	0
tt0011801	0
tt0013274	0
tt0014985	0
tt0015152	0
tt0015414	0
tt0015724	0
tt0015737	0
tt0016029	1

FOR MACHINE LEARNING DATA TRANSFORMATION

DEALING WITH MERGED CELL, EMPTY VALUE PRESENTED AS - & THOUSAND SEPARATOR AS ,

```
url = 'https://www.imdb.com/title/tt0017009/ratings?ref_=tturv_sa_3'
response = requests.get(url)
soup = bs4.BeautifulSoup(response.content, 'lxml')
```

```
table = soup.find('div', class_ = "allText")
table_user_rating = pd.read_html(str(table))
table_user_rating[1]
```

	Unnamed: 0	All Ages	<18	18-29	30-44	45+
0	All	7.3 12	-	-	6.8 5	8.5 6
1	Males	7.2 9	-	-	7.0 4	8.0 4
2	Females	7.5 3	-	-	6.0 1	9.0 2

```
import re
user_ratings = table_user_rating[1]
user_ratings = user_ratings.replace('–','0 0 0')
user_ratings = user_ratings.drop('Unnamed: 0', axis=1)
columns = ['all','under_18','f18_29','f30_44','f45']
user_ratings.columns = columns
user_ratings = user_ratings.applymap(lambda val:
                                     re.sub(r'(\d*),(\d*)',
                                     r'\1\2', val))
user_ratings
```

	all	under_18	f18_29	f30_44	f45
0	7.3 12	0 0 0	0 0 0	6.8 5	8.5 6
1	7.2 9	0 0 0	0 0 0	7.0 4	8.0 4
2	7.5 3	0 0 0	0 0 0	6.0 1	9.0 2

FOR MACHINE LEARNING DATA TRANSFORMATION

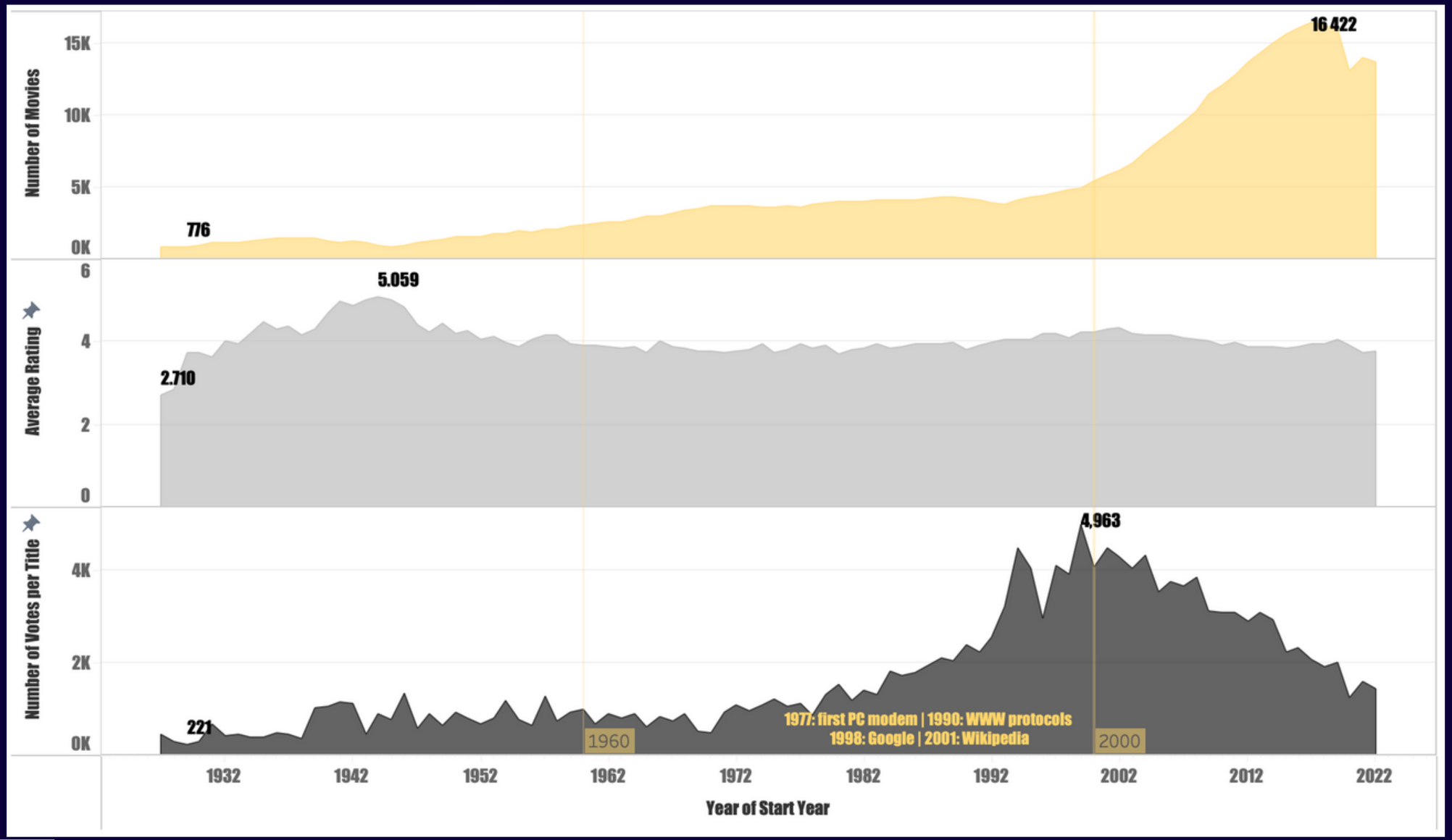
FLATTEN TABLE INTO ROW & COMPILE ROW INTO TABLE

```
def rating_row(df, col):
    """
    To flatten a rating table (6x5) into a row (1x30), and add the title of this table as the
    first column
    """

    df_name = df[col].str.split(' ', expand=True).rename(columns={0:'ratings', 1:'drop',
    2:'v_counts'}, index={0:'all', 1:'males', 2:'females'}).fillna(0)
    df_name.drop('drop',axis=1, inplace=True)
    df_name = df_name.stack()
    df_name.index = df_name.index.map('{0[0]}_{0[1]}'.format)
    df_name = df_name.to_frame().T
    df_name['age_group'] = col
    return df_name
```

```
ratings = pd.concat([all_ages_ratings, under_18_ratings, f18_29_ratings, f30_44_ratings,
f45_ratings]).set_index("age_group")
cols = ratings.columns
ratings[cols] = ratings[cols].apply(pd.to_numeric, errors='coerce')
ratings = ratings.stack()
ratings.index = ratings.index.map('{0[0]}_{0[1]}'.format)
ratings = ratings.to_frame().T
ratings['tconst'] = tconst
ratings = ratings[ ['tconst'] + [ col for col in ratings.columns if col != 'tconst' ]]
rating_list_fr = pd.concat([rating_list_fr, ratings], axis = 0, ignore_index=True)

sleep(random.uniform(0.2, 1.0))
```

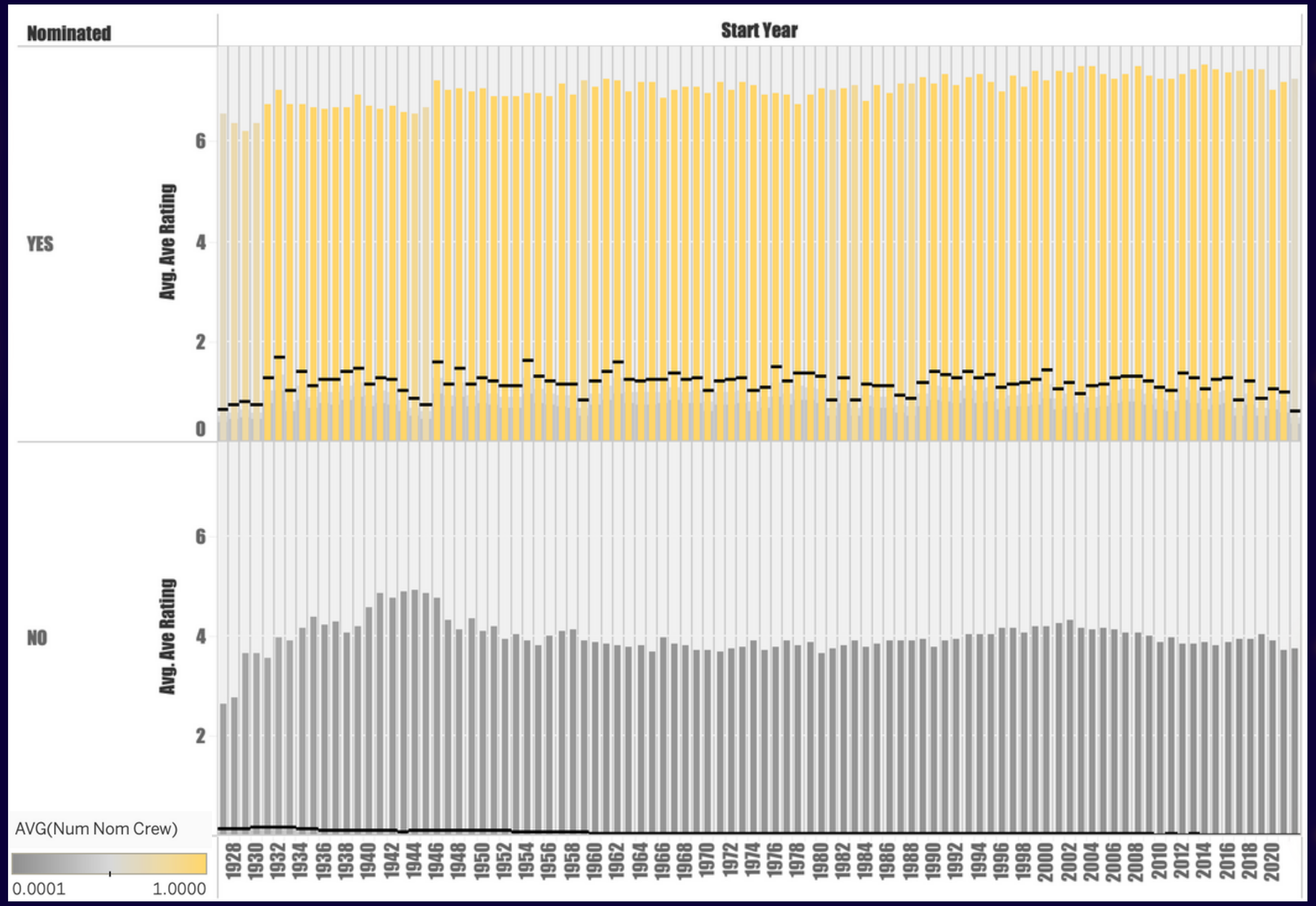
	tconst	all_males_ratings	all_males_v_counts
0	tt0000008	5.30000	1458.00000
1	tt0081609	6.60000	844.00000
2	tt0082034	6.30000	1407.00000
3	tt0082449	6.20000	5711.00000
4	tt0084287	7.30000	4890.00000

TREND


Number of Movies
15K
10K
5K
0K
16 422
776
Average Rating
6
4
2
0
5.059
2.710
Number of Votes per Title
4K
2K
0K
4,963
221
1977: first PC modem | 1990: WWW protocols
1998: Google | 2001: Wikipedia
1960
2000
1932
1942
1952
1962
1972
1982
1992
2002
2012
2022
Year of Start Year

NOMINATED CREW

NOMINATION



_ 7.1/ 10

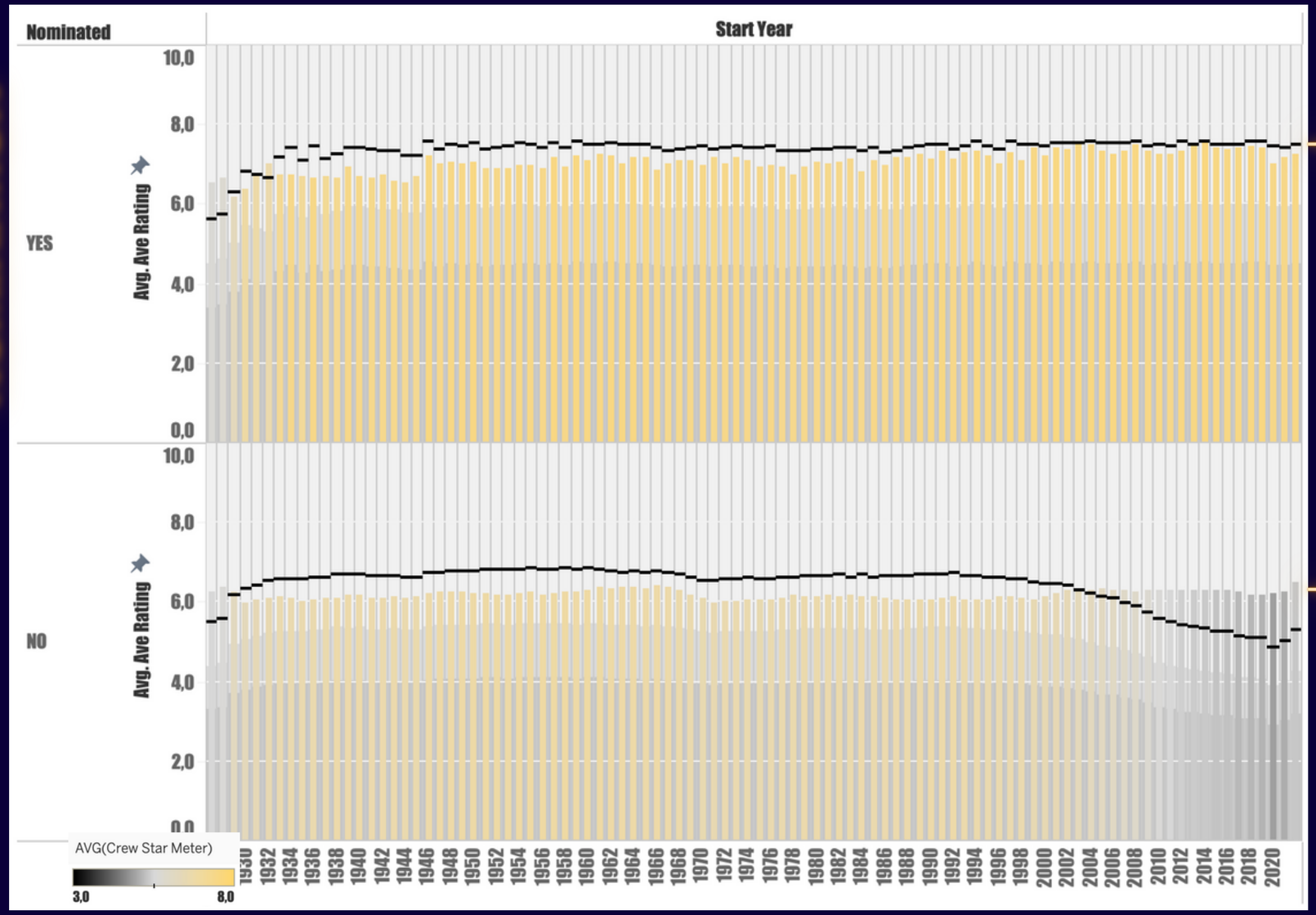
_ 1 CREW

_ 3.9/ 10

_ 0 CREW

STAR METER

NOMINATION

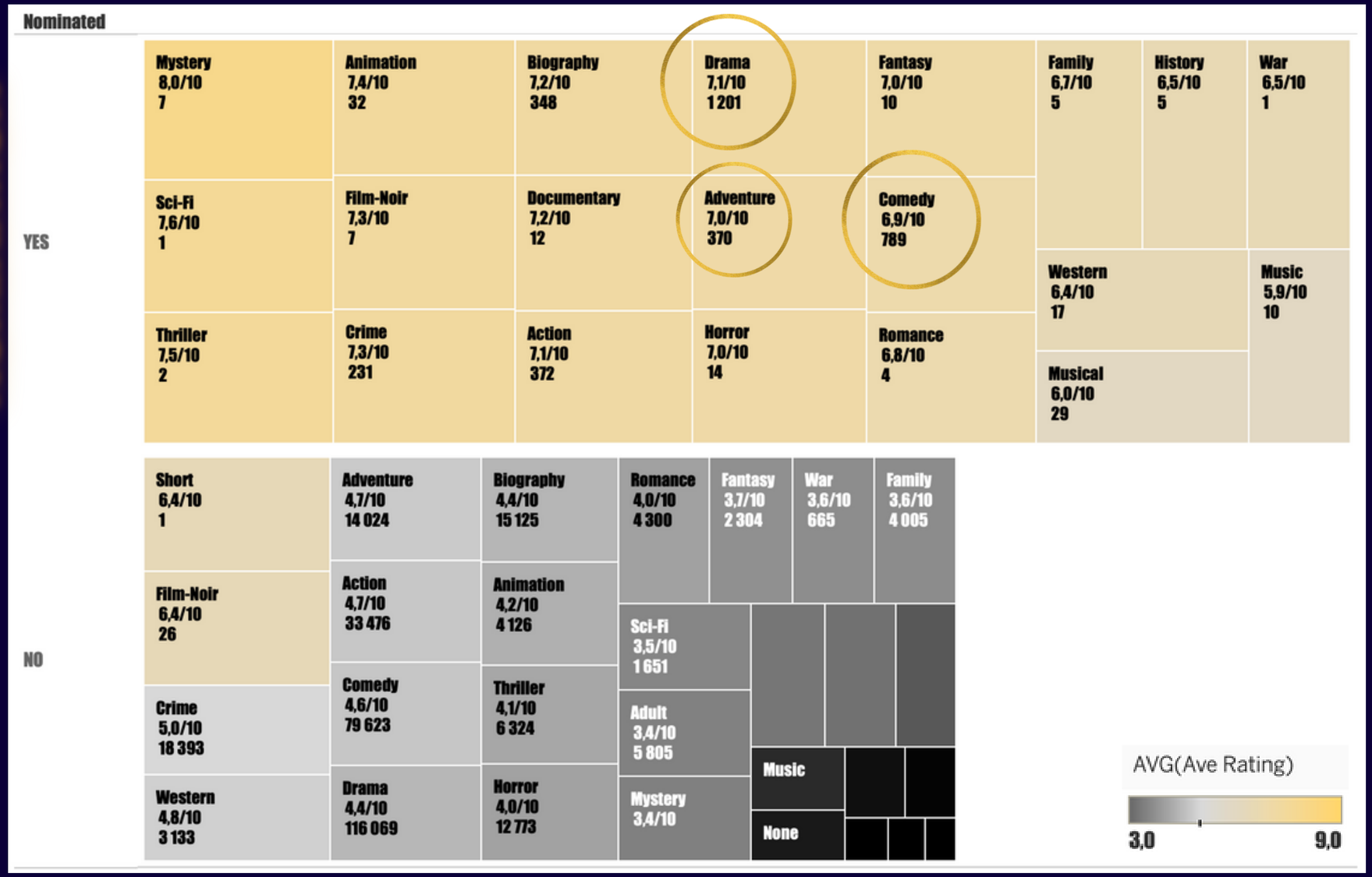


7.4/ 10

7/ 10

GENRES

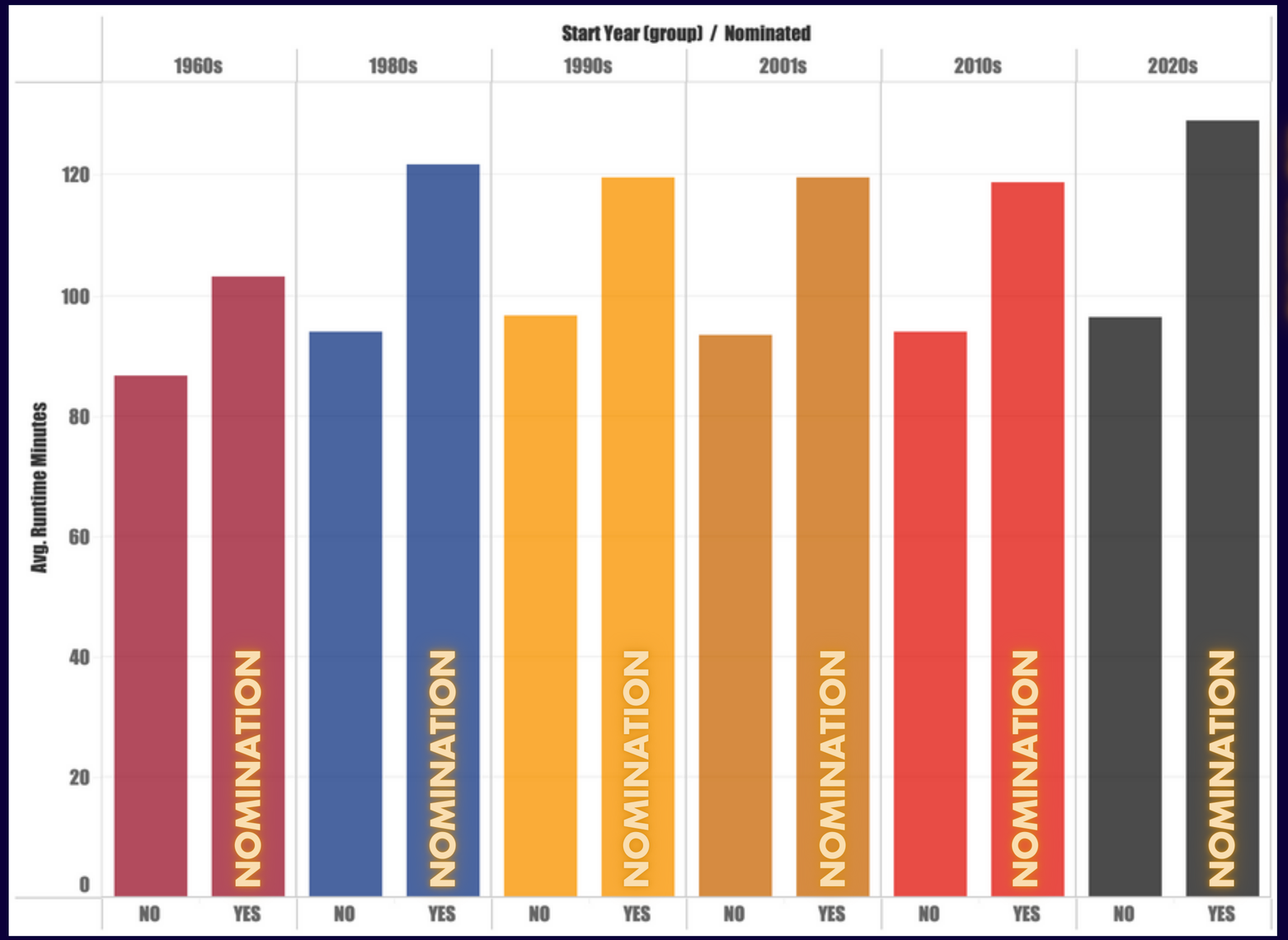
NOMINATION



WINNING CHANCE BY GENRES

	Won / Nominated		
	NO		YES
Genres1	YES	NO	YES
Drama	0,20%	24,99%	0,05%
Comedy	0,14%	17,15%	0,03%
Biography	0,05%	3,25%	0,02%
Adventure	0,06%	3,02%	0,02%
Action	0,07%	7,21%	0,01%
Crime	0,04%	3,96%	0,01%
Horror	0,00%	2,75%	0,00%
Film-Noir	0,00%	0,01%	0,00%
Documentary	0,00%	21,66%	0,00%
Music	0,00%	0,87%	0,00%
Fantasy	0,00%	0,50%	0,00%
Animation	0,01%	0,89%	0,00%

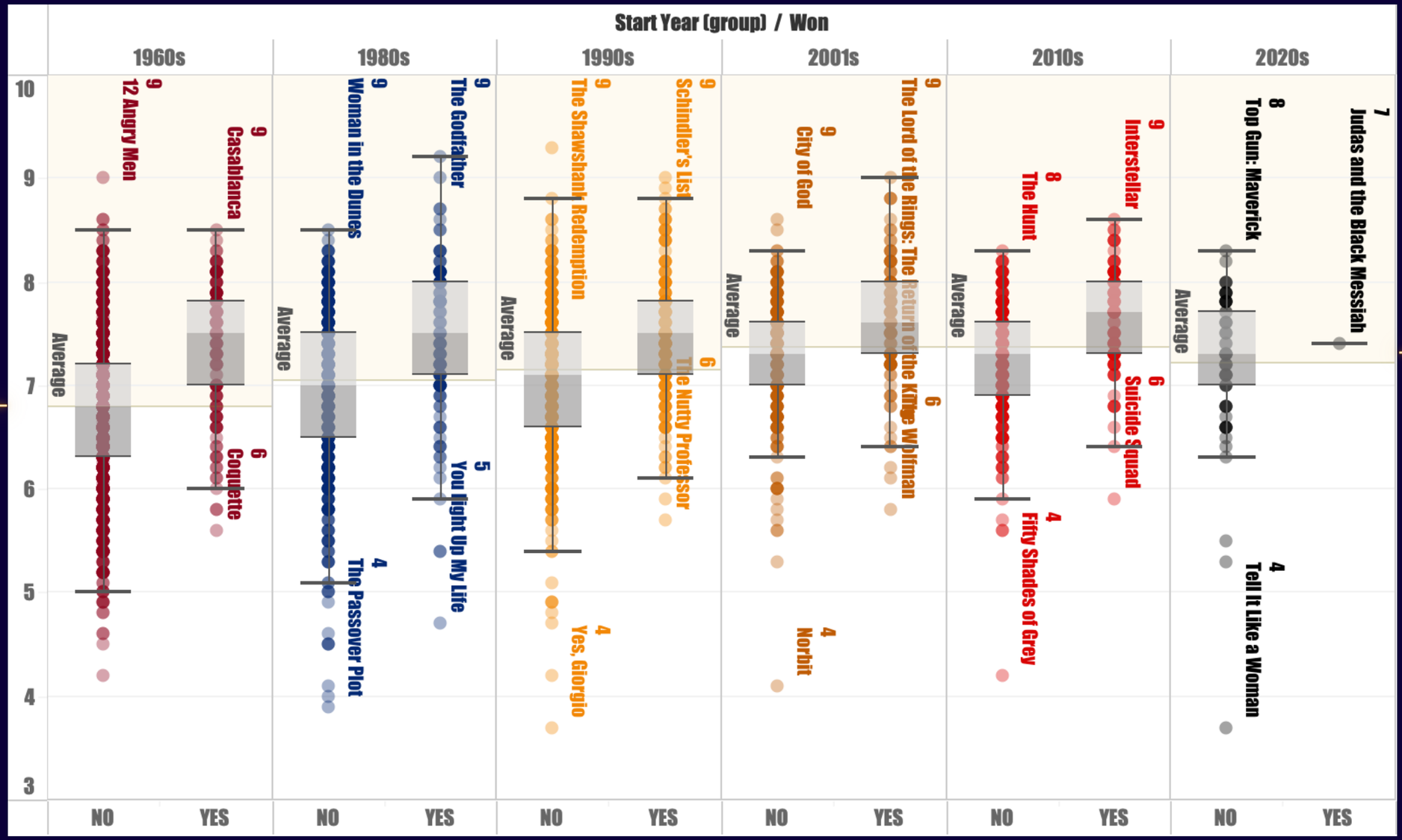
RUNNING TIME



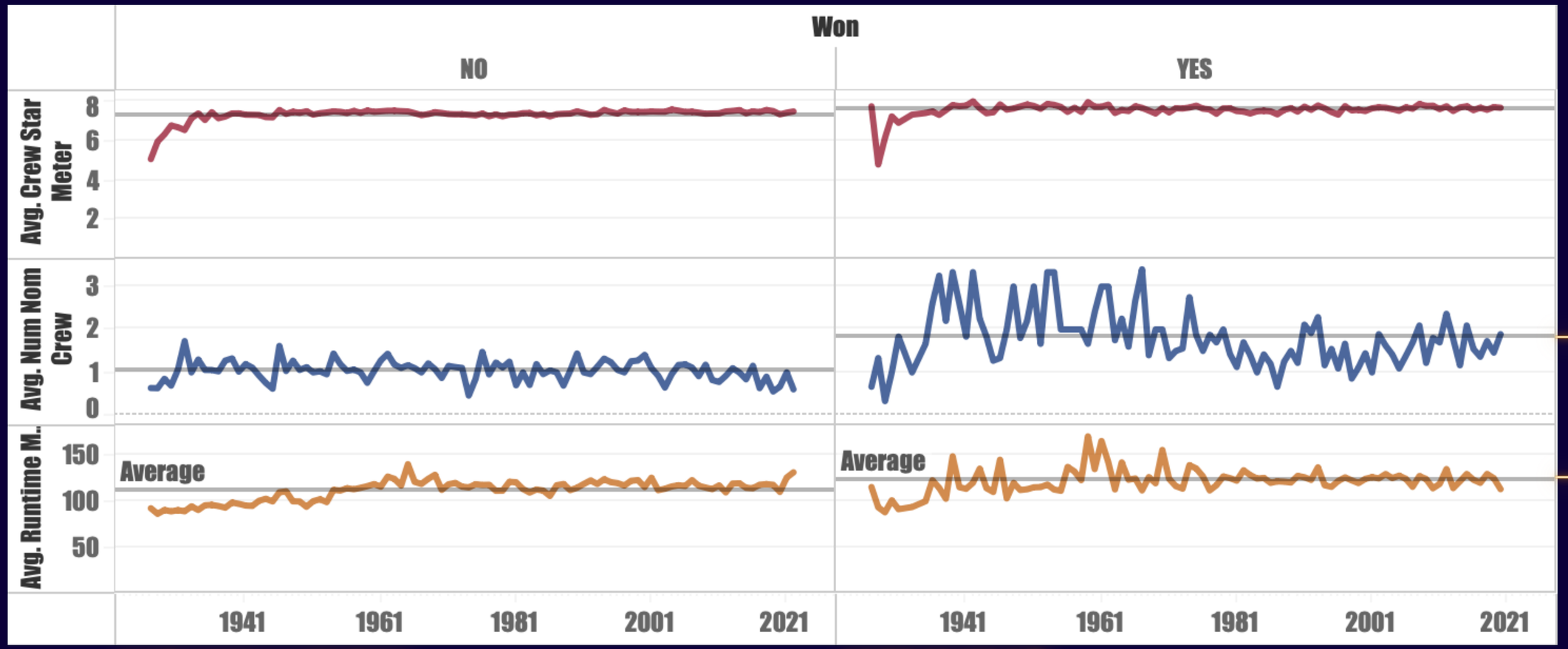
AVERAGE RATING

6.8/10

7.3/10



WINNING PUSH



+1 CREW

+10 MINS

NOMINATION

WINNING

QUERIES

SUBSTRING INDEX X UNION

```sql
-- 1. CREATE A TABLE THAT MAPS EACH INDIVIDUAL WITH THE MOVIES THEY ARE KNOWN FOR:
CREATE TABLE nconst_known_for
AS (
SELECT
  nconst, SUBSTRING_INDEX(SUBSTRING_INDEX(knownForTitles, ',' , 1), ', ', -1) knownForTitle FROM actor_name_clean
UNION ALL
SELECT
  nconst, SUBSTRING_INDEX(SUBSTRING_INDEX(knownForTitles, ',' , 2), ',', -1) knownForTitle FROM actor_name_clean
UNION ALL
SELECT
  nconst, SUBSTRING_INDEX(SUBSTRING_INDEX(knownForTitles, ',' , 3), ',', -1) knownForTitle FROM actor_name_clean
UNION ALL
SELECT
  nconst, SUBSTRING_INDEX(SUBSTRING_INDEX(knownForTitles, ',' , 4), ',', -1) knownForTitle FROM actor_name_clean
);
```

SQL X PYTHON

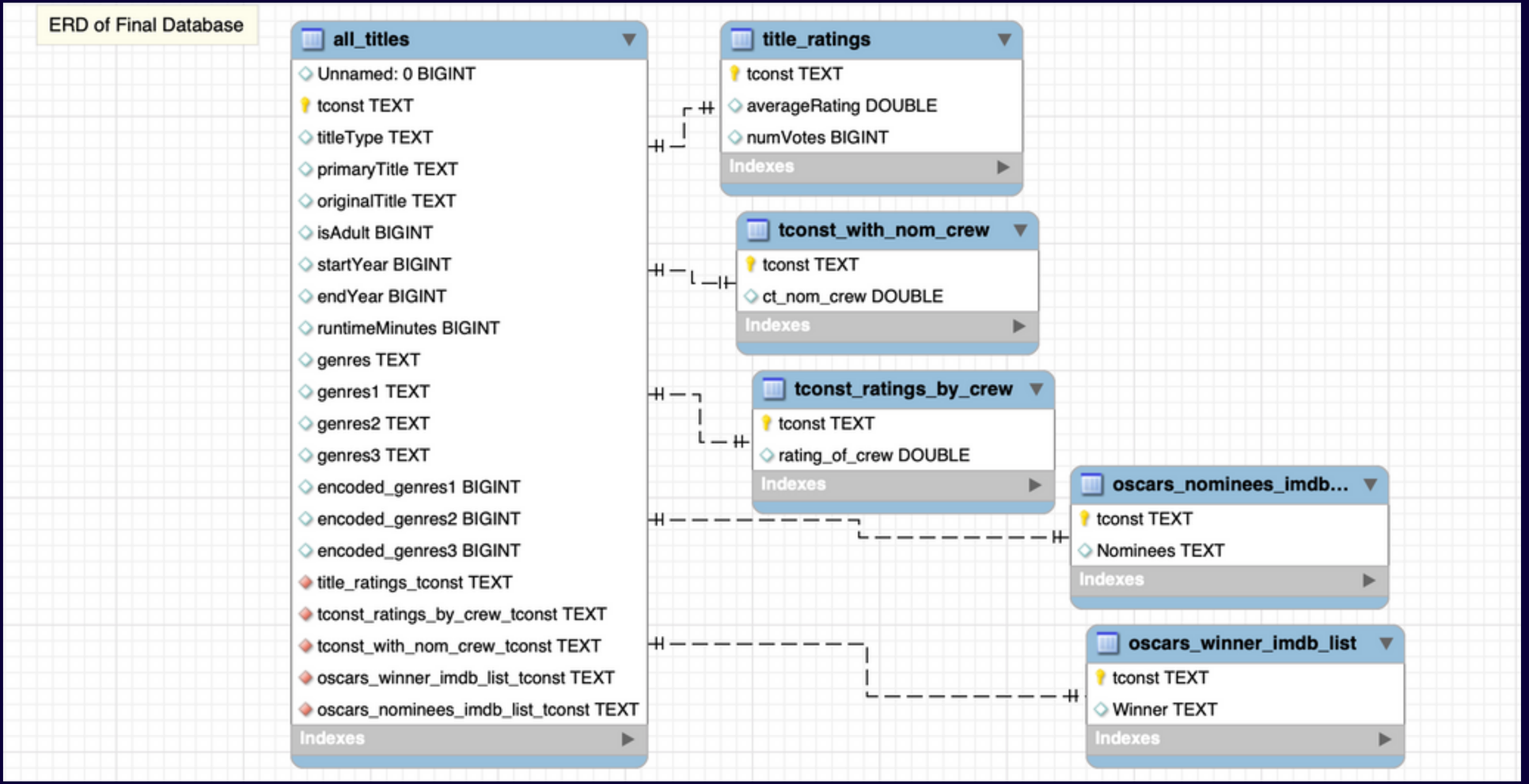
```python
def convert_pd_df_tosql(fname,table_name,schema="imdb"):
    connection_string = 'mysql+pymysql://root:' + pw + '@127.0.0.1:3306/'
    engine = create_engine(connection_string)
    df = pd.read_csv(fname)
    df.to_sql(table_name,engine,schema,index=False, chunksize=5000)
    return "Created table"
```

✓ 0.0s

LEFT JOIN

```sql
-- 7. CREATE FINAL DATABASE:
CREATE TABLE imdb_megadata_2
AS
  (SELECT  a.*,
           tr.averageRating as ave_rating,
           tr.numVotes as num_votes,
           cc.ct_nom_crew as count_nom_crew,
           tc.rating_of_crew as crew_star_meter,
           ond.Nominees as nominated,
           ow.Winner as won
  FROM all_titles_2 a
  LEFT JOIN title_ratings tr ON a.tconst = tr.tconst
  LEFT JOIN tconst_with_nom_crew cc ON a.tconst = cc.tconst
  LEFT JOIN tconst_ratings_by_crew tc ON a.tconst = tc.tconst
  LEFT JOIN oscars_winner_imdb_list ow ON a.tconst = ow.tconst
  LEFT JOIN oscars_nominees_imdb_list ond ON a.tconst = ond.tconst);

  SELECT * FROM imdb_megadata_2;
  SELECT * FROM all_titles;
```

ERD


ERD of Final Database
all_titles
Unnamed: 0 BIGINT
tconst TEXT
titleType TEXT
primaryTitle TEXT
originalTitle TEXT
isAdult BIGINT
startYear BIGINT
endYear BIGINT
runtimeMinutes BIGINT
genres TEXT
genres1 TEXT
genres2 TEXT
genres3 TEXT
encoded_genres1 BIGINT
encoded_genres2 BIGINT
encoded_genres3 BIGINT
title_ratings_tconst TEXT
tconst_ratings_by_crew_tconst TEXT
tconst_with_nom_crew_tconst TEXT
oscars_winner_imdb_list_tconst TEXT
oscars_nominees_imdb_list_tconst TEXT
Indexes
title_ratings
tconst TEXT
averageRating DOUBLE
numVotes BIGINT
Indexes
tconst_with_nom_crew
tconst TEXT
ct_nom_crew DOUBLE
Indexes
tconst_ratings_by_crew
tconst TEXT
rating_of_crew DOUBLE
Indexes
oscars_nominees_imdb...
tconst TEXT
Nominees TEXT
Indexes
oscars_winner_imdb_list
tconst TEXT
Winner TEXT
Indexes

MACHINE LEARNING

+ FEATURES

CORR/ F. IMPORTANCE

X_Y SPLIT

RESULTS

1. Decision Tree - d5s2l1

		Train	Test	Cross Val
0	Accuracy	0.98	0.98	
1	Precision	0.73	0.50	
2	Recall	0.41	0.38	
3	Ave of Scores			0.97
4	Std of Scores			0.00

2. Decision Tree - BP - d3s2l2

		Train	Test	Cross Val
0	Accuracy	0.98	0.98	
1	Precision	0.58	0.45	
2	Recall	0.57	0.48	
3	Ave of Scores			0.98
4	Std of Scores			0.00

3. Decision Tree - NR - d5s2l2

	Train	Test	Cross Val
Accuracy	0.95	0.94	
Precision	0.71	0.57	
Recall	0.52	0.40	
Ave of Scores			0.94
Std of Scores			0.00

4. Decision Tree - NR - BP - d5s4l2

	Train	Test	Cross Val
Accuracy	0.95	0.94	
Precision	0.71	0.58	
Recall	0.52	0.40	
Ave of Scores			0.94
Std of Scores			0.00

5. Decision Tree - U - BP - d3l2

		Train	Test	Cross Val
0	Accuracy	0.95	0.92	
1	Precision	0.92	0.16	
2	Recall	0.98	0.98	
3	Ave of Scores			0.95
4	Std of Scores			0.00

6. Decision Tree - NR - U - d5s2l1

7. Decision Tree - NR - U - BP - d5s2l1

		Train	Test	Cross Val
0	Accuracy	0.92	0.88	
1	Precision	0.90	0.31	
2	Recall	0.95	0.74	
3	Ave of Scores			0.92
4	Std of Scores			0.01

12. Decision Tree - R - d5s2l1

	Train	Test	Cross Val
Accuracy	0.98	0.98	
Precision	0.74	0.47	
Recall	0.33	0.29	
Ave of Scores			0.97
Std of Scores			0.00

13. Decision Tree - R - BP - d5s2l2

	Train	Test	Cross Val
Accuracy	0.98	0.98	
Precision	0.74	0.47	
Recall	0.32	0.29	
Ave of Scores			0.97
Std of Scores			0.00

N - ON NOMINATION AS TARGET
R - USING RATINGS ONLY

BP - BEST PARAMETERS
U - UPSAMPLING

RESULTS

8. Random Forest - BP - d10s3l1

		Train	Test	Cross Val
0	Accuracy	1.00	0.98	
1	Precision	1.00	0.50	
2	Recall	0.89	0.31	
3	Ave of Scores			0.98
4	Std of Scores			0.00

9. Random Forest - U - BP - d10s3e500

		Train	Test	Cross Val
0	Accuracy	0.99	0.97	
1	Precision	0.98	0.32	
2	Recall	1.00	0.81	
3	Ave of Scores			0.98
4	Std of Scores			0.00

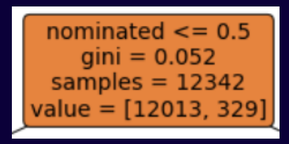
10. KNNC - BP - k7

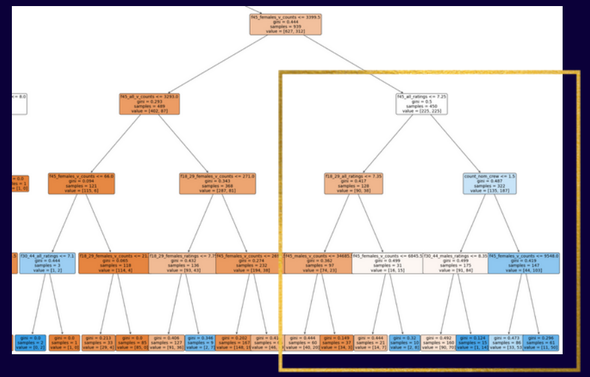
		Train	Test	Cross Val
0	Accuracy	1.00	0.98	
1	Precision	1.00	0.25	
2	Recall	1.00	0.10	
3	Ave of Scores			0.97
4	Std of Scores			0.00

11. KNNC - U - BP - k7

		Train	Test	Cross Val
0	Accuracy	1.00	0.90	
1	Precision	1.00	0.11	
2	Recall	1.00	0.79	
3	Ave of Scores			0.94
4	Std of Scores			0.00

RESULTS - MODEL 1





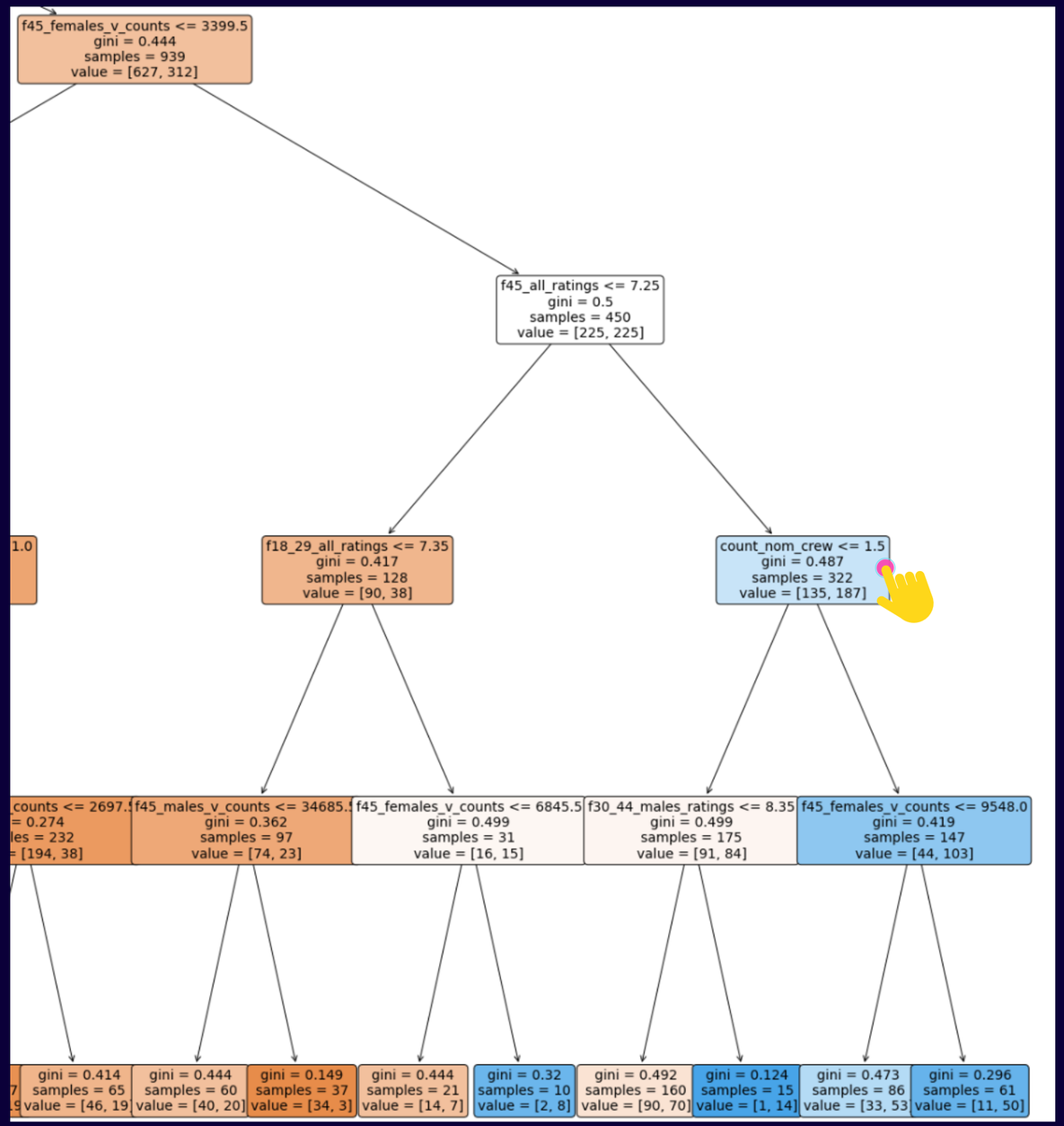
TV - WINNING

DEPTH = 5

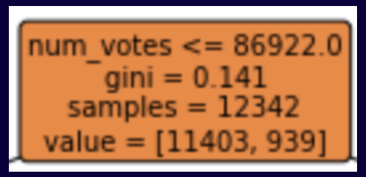
MIN SPLIT = 2

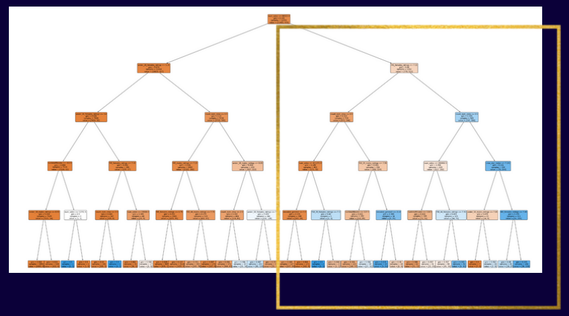
MIN LEAF = 1

PRECISION ON TEST = 0.50



RESULTS - MODEL 3





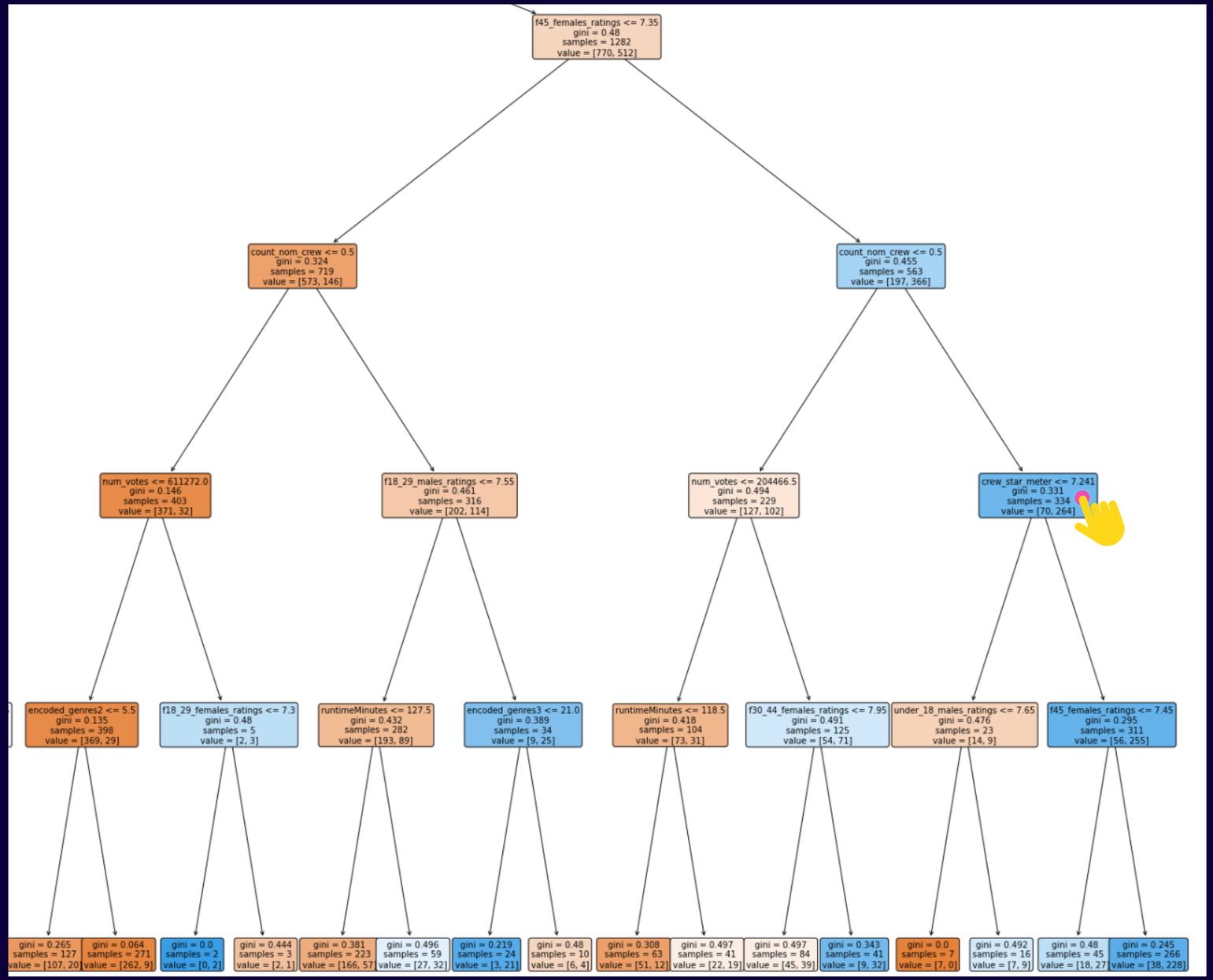
TV - NOMINATED

DEPTH = 5

MIN SPLIT = 2

MIN LEAF = 1

PRECISION ON TEST = 0.58



CONCLUSION

EDA

- Average rating: 7.5
- # nominated crew: 2
- Star meter: 7.6
- Run time: 123 minutes
- Genres: drama, comedy, biography or adventure. If the genres belongs to the likes of sci-fi or mystery, it would need to have a strong storyline or cast to drive

MACHINE LEARNING

- From 45 female vote count & ratings
- From 45 all ratings > 7.25
- # nominated crew: 2
- 50% precision test score

WHAT'S NEXT

MORE DATAS

BUDGET
BOX OFFICE
NOMINATION/ PERSON
FULL CREW CREDITS
FULL RATINGS BEFORE 1980

NLP

USER & CRITIC REVIEWS
SUBJECT

OTHER MODELS

NAIVE BAYES
SVM

CONTACTS

LINKEDIN
https://www.linkedin.com
/in/thanhthao/

GITHUB
https://github.com/tthaopham

EMAIL
thao.pham.ptt@gmail.com

THE DATAS

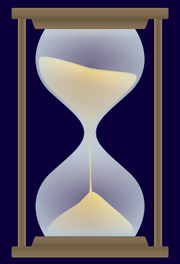
IMDB

12mil x 6
10mil x 9
54mil x 6
1.2mil x 3

464k x 21
EDA

WEB SCRAPING

600 mins
+
1140 mins



150k x 50
ML

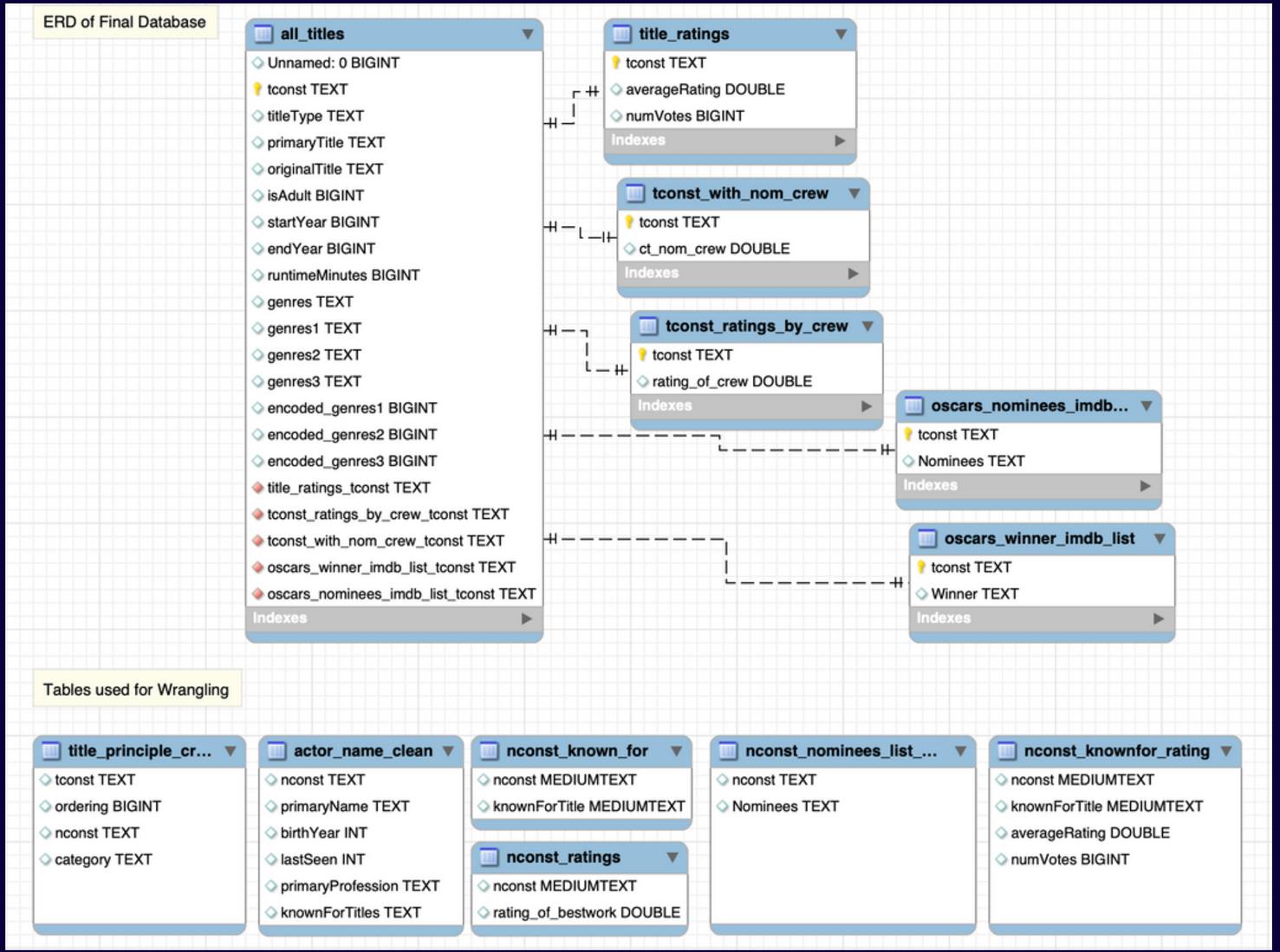
VALUE COUNTS

```
imdb_won['won'].value_counts()
✓ 0.7s

0    15051
1      377
Name: won, dtype: int64
```

```
imdb_nom['nominated'].value_counts()
✓ 0.1s

0    14293
1     1135
Name: nominated, dtype: int64
```

WINNER TITLES

ERD - FULL


ERD of Final Database
all_titles
Unnamed: 0 BIGINT
tconst TEXT
titleType TEXT
primaryTitle TEXT
originalTitle TEXT
isAdult BIGINT
startYear BIGINT
endYear BIGINT
runtimeMinutes BIGINT
genres TEXT
genres1 TEXT
genres2 TEXT
genres3 TEXT
encoded_genres1 BIGINT
encoded_genres2 BIGINT
encoded_genres3 BIGINT
title_ratings_tconst TEXT
tconst_ratings_by_crew_tconst TEXT
tconst_with_nom_crew_tconst TEXT
oscars_winner_imdb_list_tconst TEXT
oscars_nominees_imdb_list_tconst TEXT
Indexes
title_ratings
tconst TEXT
averageRating DOUBLE
numVotes BIGINT
Indexes
tconst_with_nom_crew
tconst TEXT
ct_nom_crew DOUBLE
Indexes
tconst_ratings_by_crew
tconst TEXT
rating_of_crew DOUBLE
Indexes
oscars_nominees_imdb...
tconst TEXT
Nominees TEXT
Indexes
oscars_winner_imdb_list
tconst TEXT
Winner TEXT
Indexes
Tables used for Wrangling
title_principle_cr...
tconst TEXT
ordering BIGINT
nconst TEXT
category TEXT
actor_name_clean
nconst TEXT
primaryName TEXT
birthYear INT
lastSeen INT
primaryProfession TEXT
knownForTitles TEXT
nconst_known_for
nconst MEDIUMTEXT
knownForTitle MEDIUMTEXT
nconst_ratings
nconst MEDIUMTEXT
rating_of_bestwork DOUBLE
nconst_nominees_list_...
nconst TEXT
Nominees TEXT
nconst_knownfor_rating
nconst MEDIUMTEXT
knownForTitle MEDIUMTEXT
averageRating DOUBLE
numVotes BIGINT

THANK YOU

NOMINATED CREW

25 MINS

25 MINS