Sentiment Analysis of Steam Games Reviews

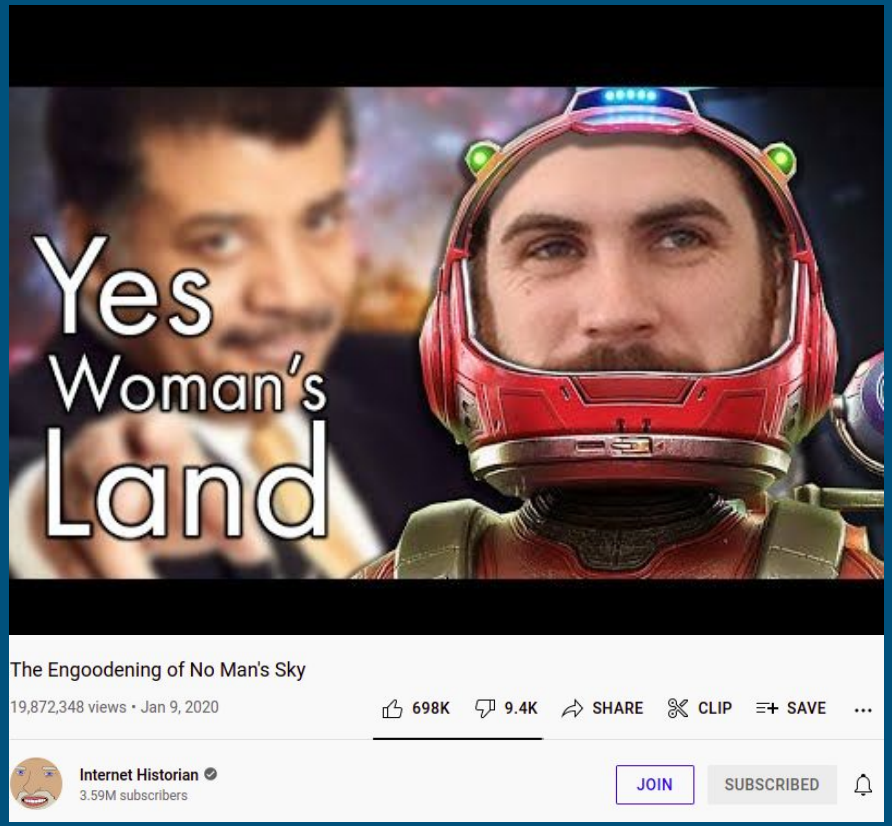
Optimization using Bayesian search

A project on NLP by Rafif

https://www.linkedin.com/in/mrafifrbbn/

https://github.com/mrafifrbbn

Background



"Then he starts breaking that (all the feedback of the game from various sources) down into datasets: people who haven't bought the game, people who have bought it and played it for a hundred hours, people who have returned it, etc.

Then he starts compiling those complaints into usable data, focusing on the people with the most sincere experience of the game.

Then he starts making a big list of all the things that need adding and prioritize them." - minute 39:00

- Applicable for many things: product reviews, tweets, etc.
- Very useful for passionate game developers.

Project Overview

Goal
To classify whether a game reviewer recommended the game or not and analyze what words are associated with them.

Dataset
Subset of the Steam Games Reviews dataset from Kaggle.

Focus
- End-to-end project on NLP.
- Text processing and visualization.
- Hyperparameter tuning using Bayesian search.

Dataset

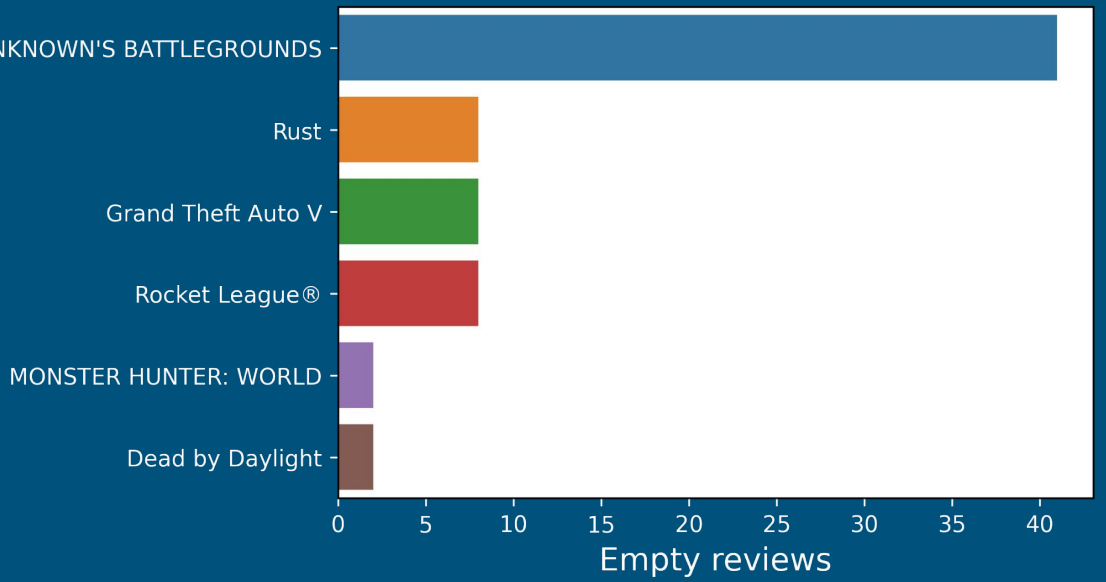
- Steam → video game distributor by Valve.
- Steam Reviews dataset: ~400000 reviews from March 2013 up to February 2019.
- This project: 20000 reviews (random sampling).

Columns	Type	Description
date_posted	obj	review date
funny	int	funny vote
helpful	int	helpful vote
hour_played	int	total hour played
is_early_access_review	bool	early access player
recommendation	obj	recommended or not
review	obj	review text
title	obj	game title

- Important columns:
  - `review` (feature)
  - `recommendation` (target)

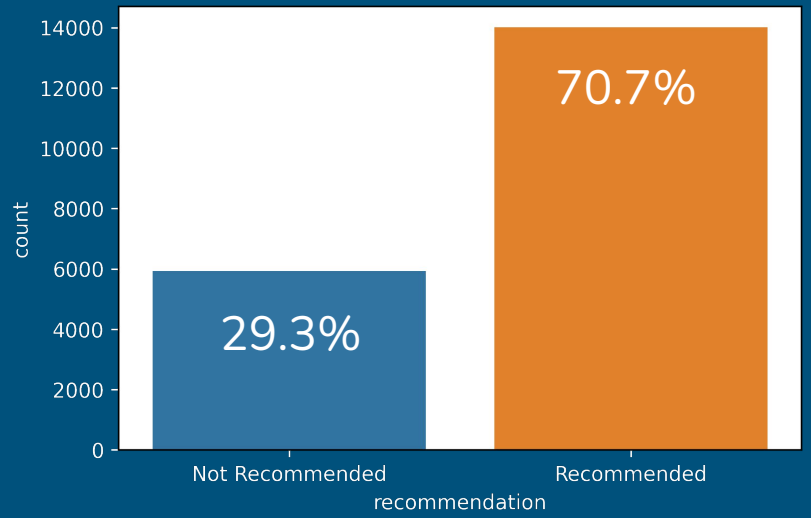
Data Pre-processing

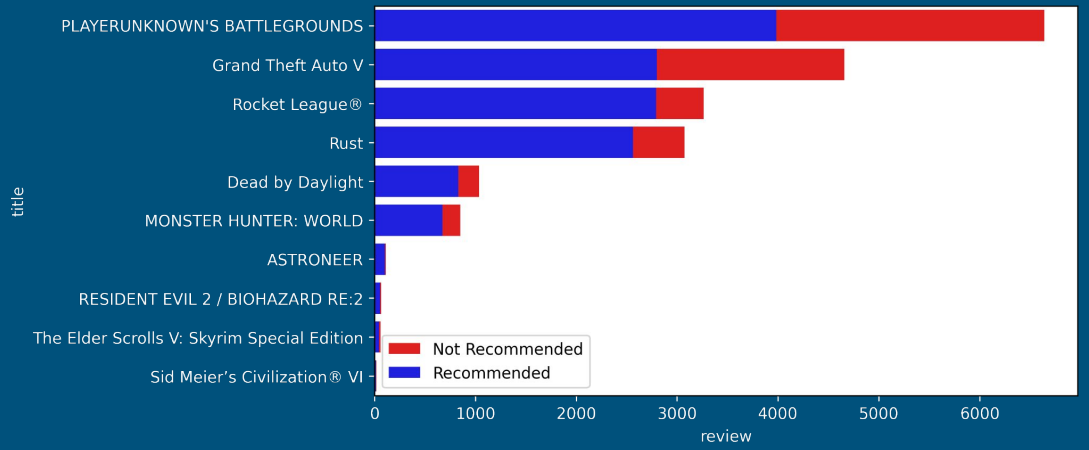
- 69 total empty reviews → removed.
- 1300 Duplicated reviews → short reviews, e.g. “awesome”, “very good”.

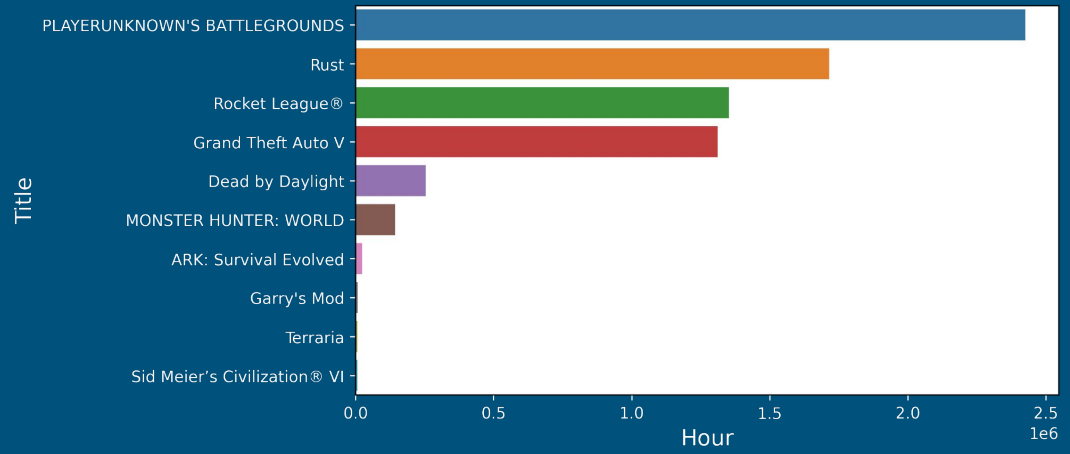


EDA

- Slight imbalance.
- Top games: PUBG, GTA V, Rocket League, Rust, DBD, Monster Hunter.

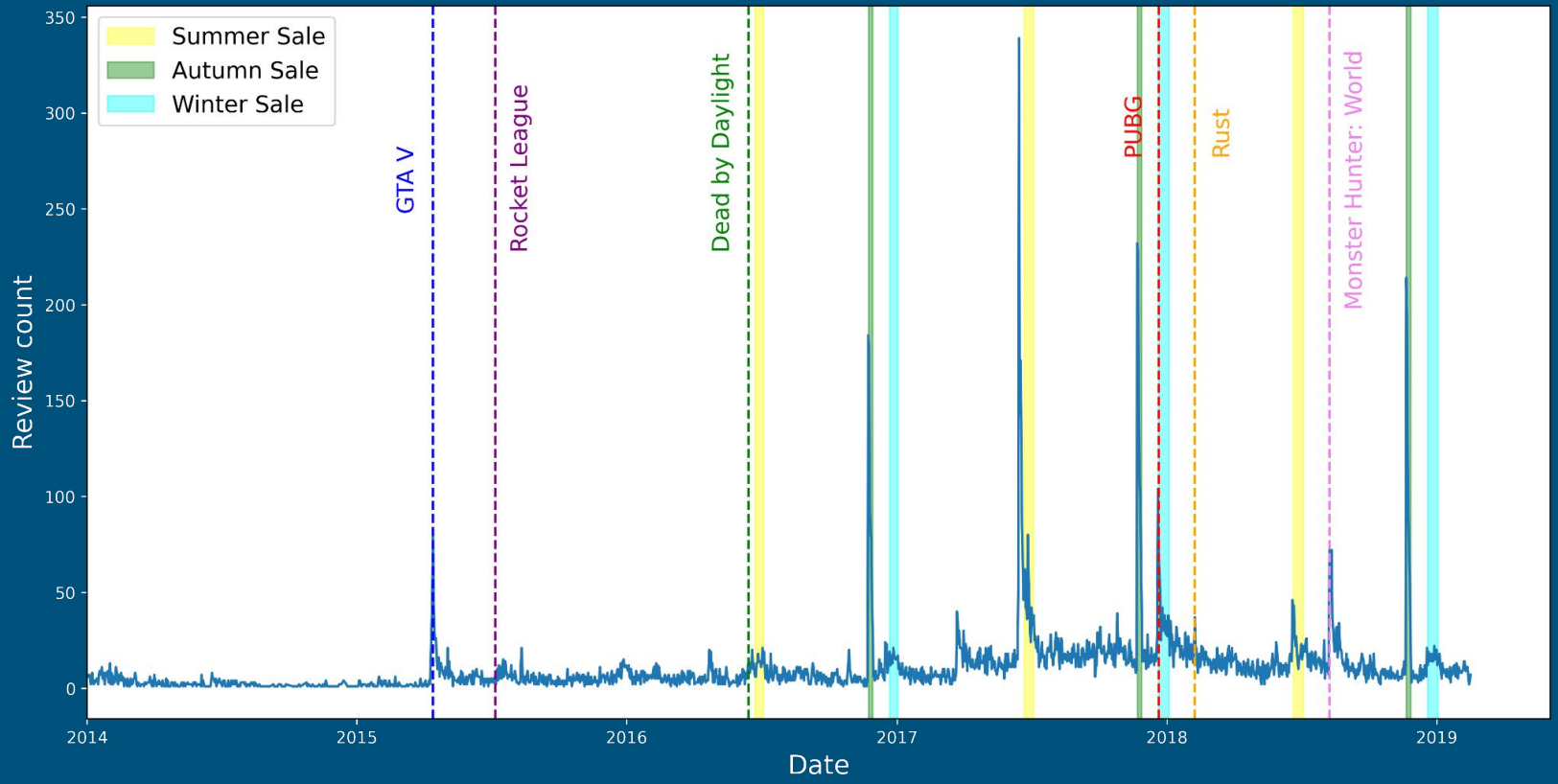






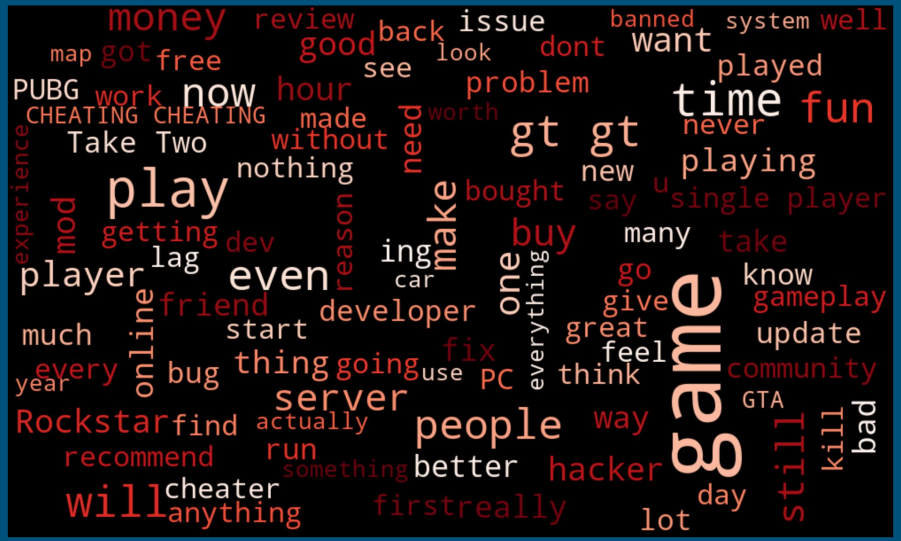
EDA: daily review count

Review spikes mostly correlate with steam sales.



EDA: word cloud

Positive reviews: ‘fun’, ‘worth’, ‘good’, ‘amazing’.



Negative reviews: ‘hacker’, ‘lag’, ‘bad’, ‘problem’.

Text Processing

1. Remove non-alphabetic characters using regular expression.
   e.g. emojis, chinese and japanese characters, digits.
2. Remove stop words using nltk library.
   e.g. ‘i’, ‘me’, ‘the’, ‘in’, ‘is’.
3. Stemming using Snowball Stemmer.
   e.g. ‘Consultant’, ‘consulting’, ‘consultantative’ → ‘consult’.
4. Remove irrelevant words manually by inspecting the word cloud.
   e.g. ‘game’, ‘play’, ‘get’.

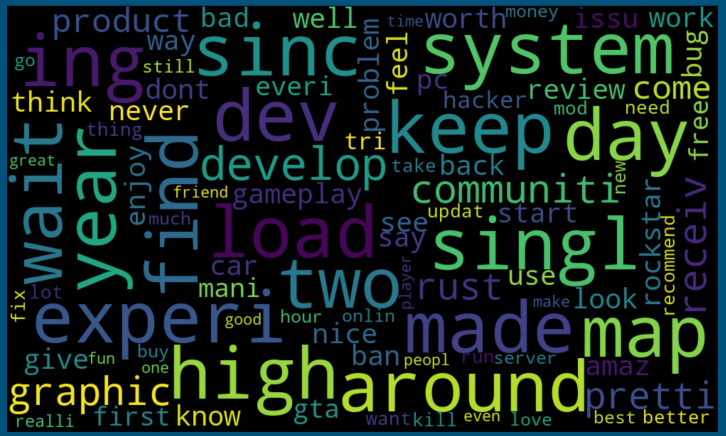
Text Processing

- Result:

```
Original review:
 Pretty much same controls and buggy mess as all the other battle royale games.

Cleaned review:
 pretti much control buggi mess battl royal
```

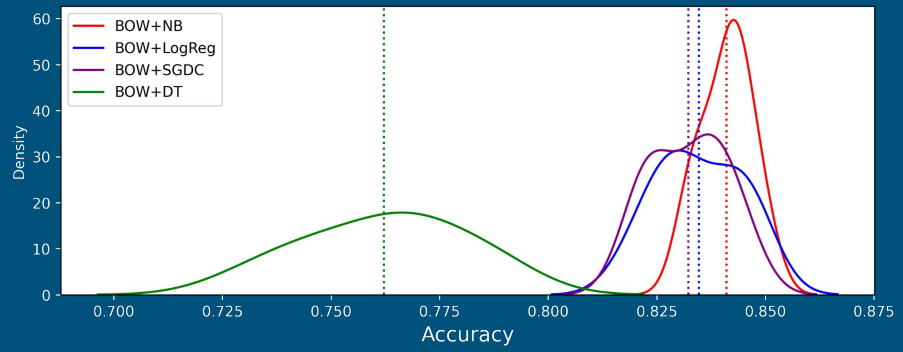
- Top words: ‘fun’, ‘good’, ‘like’.

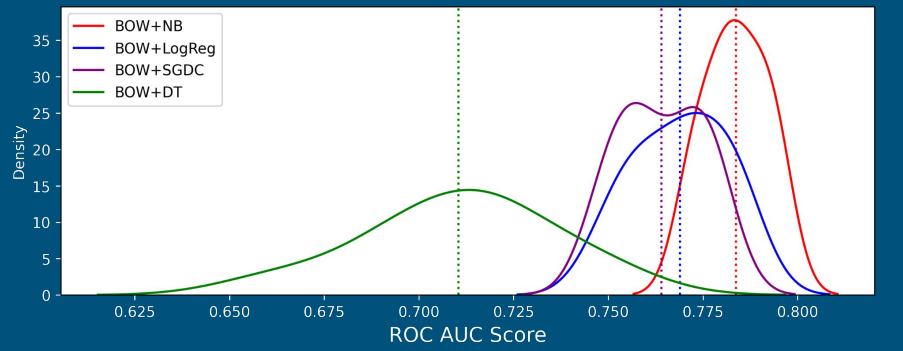


Model Building: Bag-of-Words

- Baseline model: bag-of-words vectorizer with naive bayes classifier → commonly used in sentiment analysis.
- Evaluation using cross-validation (10-folds).
- BOW+NB performs the best.

Classifier	Accuracy (%)	ROC AUC
Multinomial Naive Bayes	84.109	0.784
Logistic Regression	83.471	0.769
Stochastic Gradient Descent	83.227	0.764
Decision Tree	76.217	0.71

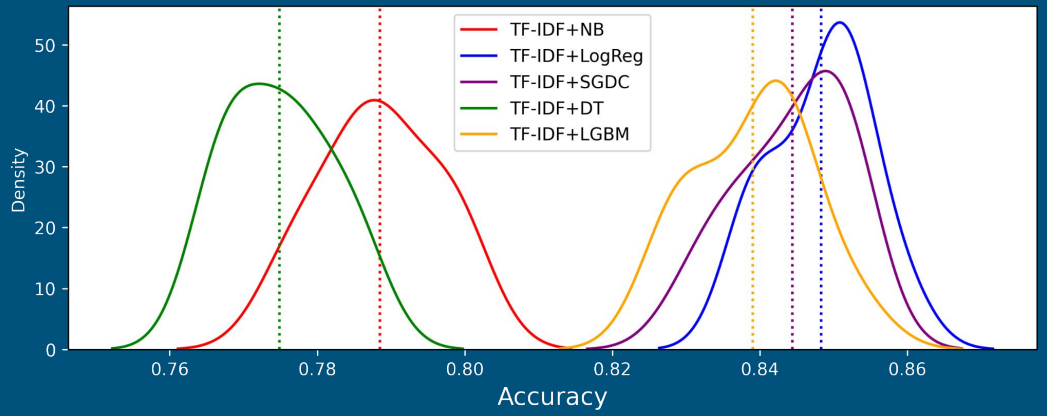


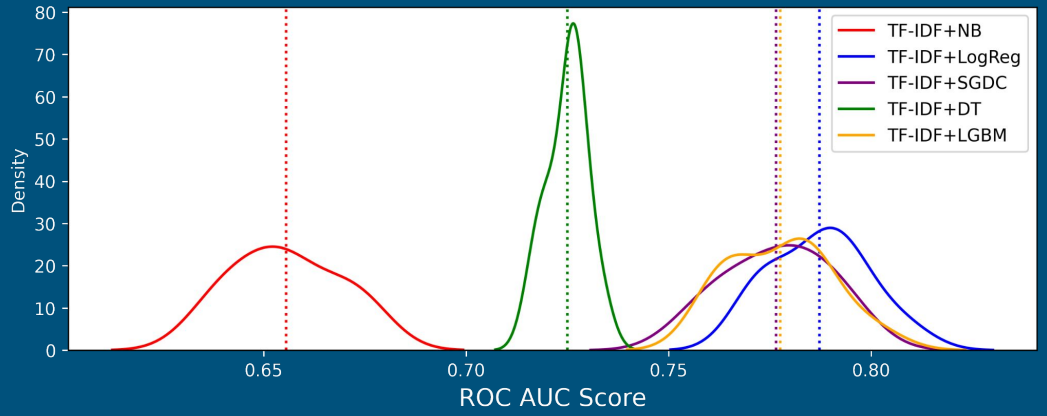


Model Building: TF-IDF

- Using TF-IDF as vectorizer.
- TF-IDF+LogReg performs the best.
- NB gets worse.

Classifier	Accuracy (%)	ROC AUC
Multinomial Naive Bayes	78.852	0.656
Logistic Regression	84.828	0.787
Stochastic Gradient Descent	84.441	0.777
Decision Tree	77.487	0.725
LightGBM	83.904	0.778





Model Building: Bayesian Search

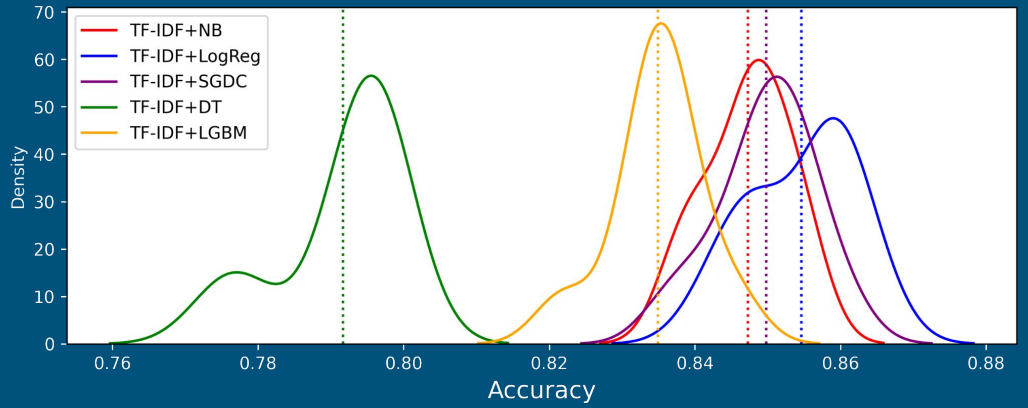
- Using `hyperopt` to tune the hyperparameters using Bayesian search.
- Only for TF-IDF models (generally better performance than BOW).
- Tuning vectorizer + classifier hyperparameters simultaneously by optimizing the accuracy score (minimizing misclassification rate).

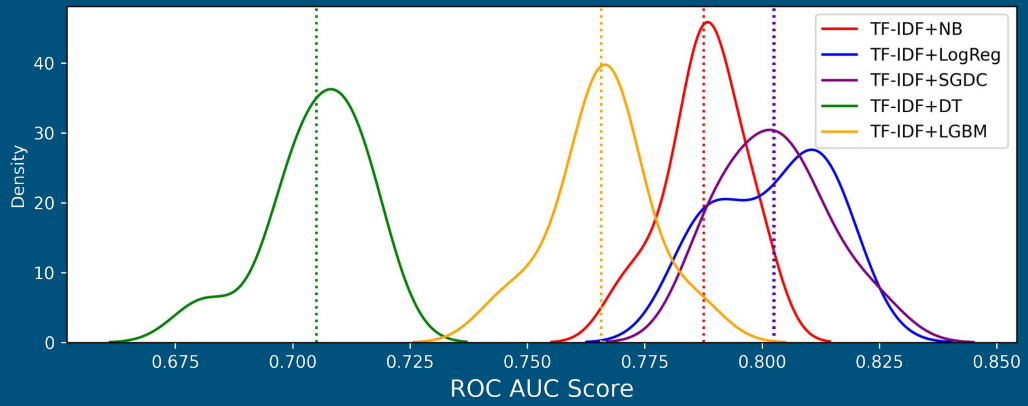
	Parameter name	Range
TF-IDF Vectorizer		
N-gram range	`ngram_range`	(1,1), (1,2), (1,3)
Maximum document frequency	`max_df`	0.5, 0.75, 1.0
Minimum document frequency	`min_df`	1, 5, 10
Maximum features	`max_features`	None, 500, 1000
Naive Bayes		
Smoothing parameter	`alpha`	logU[-3,1]
Logistic regression		
Regularization strength	`C`	logU[-3,1]
Norm of the penalty	`penalty`	l2, None
Optimization algorithm	`solver`	newton-cg, lbfgs
SGDC		
Smoothing parameter	`alpha`	logU[-5,0]
Decision Tree		
Splitting criterion at each node	`criterion`	gini, entropy
Maximum tree depth	`max_depth`	10, 20, 30, 40, 50
Minimum samples for a node	`min_samples_leaf`	5, 10, 20, 50, 100
LightGBM		
Maximum tree depth	`max_depth`	3, 4, 5, 6, 7, 8, 9, 10
Number of leaves	`num_leaves`	U[20,3000]
Learning rate	`learning_rate`	U[0.05,0.2]

Model Building: Bayesian Search

- TF-IDF+LogReg performs the best.
- NB is significantly improved.

Classifier	Accuracy (%)	ROC AUC
Multinomial Naive Bayes	84.727	0.788
Logistic Regression	85.465	0.802
Stochastic Gradient Descent	84.978	0.803
Decision Tree	79.168	0.705
LightGBM	83.493	0.766

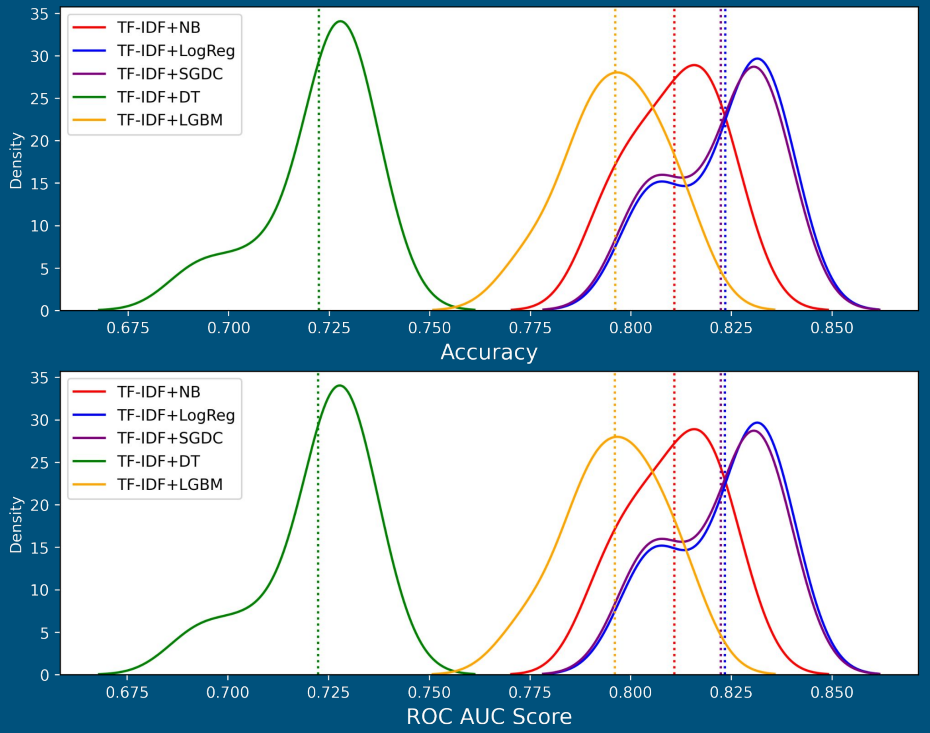




Model Building: training on balanced data

- Undersampling the positive reviews so the two classes are balanced (~8000 reviews total).
- No significant improvement → our original training set is okay.

Classifier	Accuracy (%)	ROC AUC
Multinomial Naive Bayes	81.084	0.811
Logistic Regression	82.348	0.823
Stochastic Gradient Descent	82.239	0.822
Decision Tree	72.247	0.722
LightGBM	79.613	0.796



Final Model

- Best model: tuned TF-IDF+LogReg.
- Best hyperparameters:

Hyperparameters	
ngram_range	(1,2)
max_df	0.5
min_df	1
max_features	None
C	2.158
penalty	l2
solver	newton-cg

- All common problems in online games.

Weight[?]	Feature
+8.084	great
+7.196	best
+6.801	amaz
+6.348	love
+6.069	awesom
+4.981	good
… 146716 more positive …	
… 106920 more negative …	
-4.138	hacker
-4.190	wors
-4.520	garbag
-4.643	take
-4.645	ing
-4.671	ban
-4.701	cheater
-4.701	crash
-4.728	unplay
-4.731	ruin
-5.088	trash
-5.282	money
-5.590	worst
-6.647	mod

Final Model: relevant feedback

- Top sentiments (filtered):

Positive		Negative	
Word	Weight	Word	Weight
great	+8.084	mod	-6.647
best	+7.196	crash	-4.701
amazing	+6.801	cheater	-4.701
love	+6.348	hacker	-4.138
awesome	+6.069	lag	-3.962
good	+4.981	server	-3.95
nice	+4.025	microtransaction	-2.926
addicting	+3.755	unbalance	-2.385
fun	+3.692	desync	-1.964
friend	+3.624	clunky	-1.791

- Looking at the reviews containing the negative words:
  - Players enjoy modding and want it back (GTA V).
  - A lot of cheaters and hackers that seems to be mainly Chinese players (PUBG).
  - Too much microtransactions (PUBG).
  - Some characters can use glitches/exploits, making the game unbalanced (DBD).
- Take notes, game devs and companies!

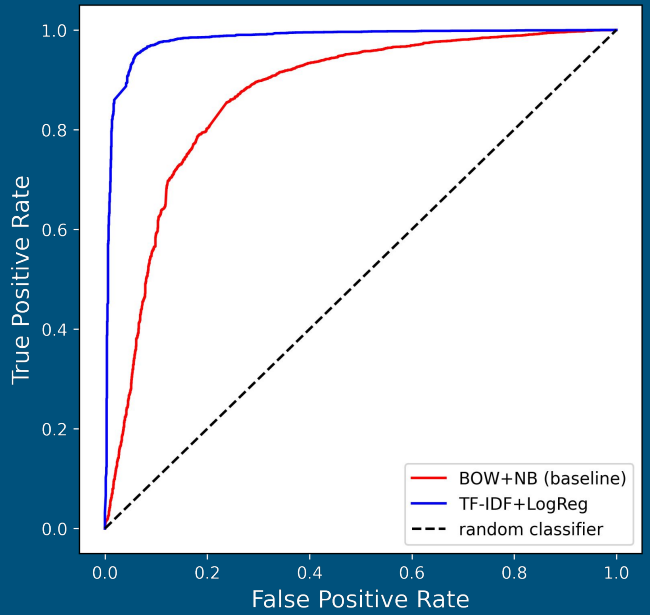
Model Evaluation: test data

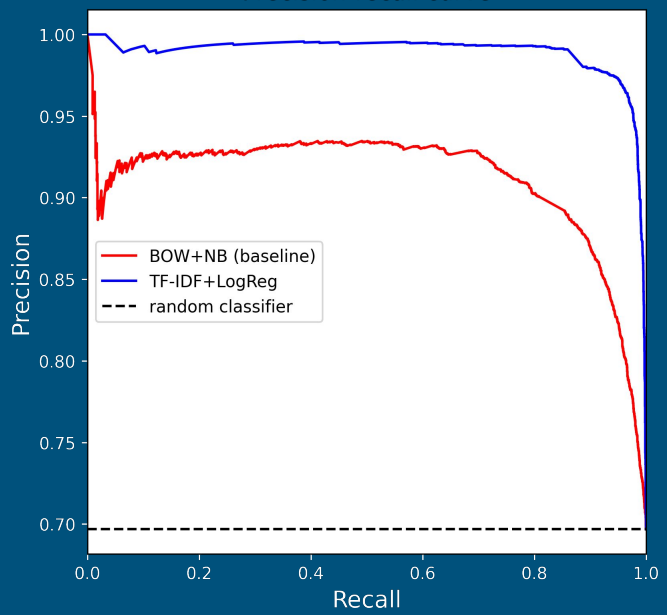
		Predicted	
		Not Recommended	Recommended
Actual	Not Recommended	1537	276
	Recommended	69	4098

- Evaluation summary:
  - True Positive = 4098
  - True Negative = 1537
  - False Positive = 276
  - False Negative = 69
  - Accuracy = 94.2%
  - ROC AUC = 0.916
- Baseline model (BOW+NB without tuning), accuracy is 83.6%, ROC AUC score is 0.784.
- High chance this model can distinguish the two sentiments on larger data.

Model Evaluation: ROC and PR curve

- Comparison with the baseline model.





- Performance greatly improved.

Conclusions

- In this dataset, most of the reviews are positive, with frequent words such as 'awesome', 'fun', and 'love'. While the negative reviews are filled with words such as 'cheater', 'bug', and 'hacker'.
- Best model: TF-IDF+LogReg with hyperparameters from Bayesian search.
- Accuracy increases from 83.6% to 94.2%, and ROC AUC score increases from 0.784 to 0.916 →may perform well on larger data.
- Deploy this model → gather reviews from Twitter, YouTube comments, internet forums, etc. to obtain more feedback.
- Problems with the games: modding, cheater/hacker, game crash, unbalance →important aspects to be fixed or added.

References

- Bais, R., Odek, P., and Ou, S. (2017). Sentiment Classification on Steam Reviews. CS229: Machine Learning course, Stanford University.
- Bergstra, J., Yamins, D., and Cox, D. (2013). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning* (pp. 115-123). PMLR.
- Marpaung, D. B. (2022). Steam Games Reviews Analysis (Sentiment Analysis). kaggle.com.
- Soni, H. (2021). Automate Hyperparameter Tuning for Multiple Models with Hyperopts. Published in *Towards Data Science*.
- Zhuo, Z. (2018). Sentiment analysis of steam review datasets using naive bayes and decision tree classifier. Illinois University Library.

Fin

Also check out the notebook on my GitHub:

https://github.com/mrafifrbbn/steam_reviews

Contact me on LinkedIn:

https://www.linkedin.com/in/mrafifrbbn/