Project Milestone 1
2/21/2025

The dataset I chose was found on Kaggle: https://www.kaggle.com/datasets/kieranpoc/steam-reviews

This dataset contains 100,000,000+ steam reviews collected from the steam User Reviews that I will be using to predict if a review is likely to be positive/negative depending on the features below:

- Author
- Steamed
- number of games owned
- number of reviews
- playtime all time
- playtime over the last 2 weeks
- playtime at the time of the review
- when they last played the game
- language
- time created
- time updated
- if the review was positive or negative (Predictor)
- number of people who voted the review up
- number of people who voted the review funny
- a helpfulness score (steam generated)
- number of comments
- if the user purchased the game on Steam
- if the user checked a box saying they got the app for free
- if the user posted this review while the game was in Early Access
- Developer response (if any)
- when the developer responded (if applicable)

I plan to use logistic regression as the model to predict what category the review will most likely fall into, increasing the speed at which the company could react to predicted negative feedback. Depending on your feedback to my email I will attempt to deploy other models that may fit the data better about such as Gradient Boosted Trees.

Project Milestone 2

I started up the secure shell terminal after adjusting compute/storage on the VM and ran the curl with -L, and -o command to follow redirects and download my dataset to my virtual instance from Kaggle.

Next, I used ls -h to check if my steam file was in the VM's directory to see if I successfully downloaded it, and it popped up to confirm.

I tried to unzip my file, however I was getting an error for the unzip command stating that the command wasn't found, so I used the sudo get update and install unzip command. I then tried the unzip command again and was prompted to authenticate for access, I ran the gcloud auth login command and unzipped the file giving me: (all_reviews file and weighted_score_above_08.csv).

I then proceeded to use the google cloud bucket create command and checked if my bucket was created by running the gcloud storage ls command.

After that, I tried to copy the files in my virtual instance to my cloud, however When I tried to run the copy command it displayed "WARNING: Omitting file://all_reviews because it is a container, and recursion is not enabled", so I enabled the recursive command while copying since the CSV file is inside a folder.

After successfully copying the files to my bucket and creating a landing folder I made several other folders using the dev/null command in my bucket to create additional folders I will be using later.

Once I got the files into my landing folder, I moved the 40GB file from the extra folder it was in, to the landing folder and deleted the empty extra folder, so that both files would be directly in the landing folder.

Project Milestone 3

I was able to perform some EDA and get back some insights. Before cleaning there were 113,885,601 records. The steam china location variable had the most nulls in the dataset. The longest length review was 9801289 characters long. English games are the most popular followed by Chinese, Russian, and Spanish in that order. The top 3 games in steam marketplace are counter strike 2, PUBG, and GTA 5. Many more people voted up than not. I was not able to use matplotlib due to my kernal crashing whenever I tried to use a pandas dataframe, or even iterating through pyspark, so I collected samples from the data set(fraction=0.05) to run visualizations based on samples of column values, turned it into a pandas dataframe and ran some graphs based on games and votes_funny, as well as language and Voted_up.

As for the cleaning I dropped the columns steam_china_location, and timestamp_updated since I figured they would be of less use to me numerically for the model. I dropped the NA rows for recommendationid, author_steamid since they are key to our analysis. For the rest I filled in with subsets of either 0 or median/mode for important missing data such as author_playtime_forever, author_playtime_last_two_weeks, author_last_played, all the voted or votes variable nulls were filled with 0 as well as the weighted score, comment count, and hidden in steam china variable. I filtered the dataset to only English language to analyze games mostly in north Americas and parts of Europe. I also saw during my EDA that voted_up had some outliers that fell outside of the regular binary input for the column, so I removed record with those outliers. When I attempted to drop review nulls, I was surprised when I checked nulls for columns and saw that no matter how many times I dropped the review nulls, it wouldn't show up when I checked. I realized that due to the predefined schema it was reading some results as NaN instead of NULL or None so the functions for cleaning weren't properly targeting those values to drop, so I converted the Nan values to None and ran the cleaning functions again. I ultimately eliminated all the nulls and the data is ready to processes in ML.

As for the next step feature engineering, I believe the biggest challenge will be to assess multicollinearity between variables and figuring out what non-linear forms may represent the data. Overfitting may be a valid concern as well as we add interaction terms to boost model performance

Project Milestone 4

I'm currently doing this for the steam reviews dataset, where I am trying to predict the variable voted_up(which tells us if the review was negative or positive).

Model: logistic Regression

Predict if the review will be positive or negative(voted_up) | given variables:

game, author_num_games_owned, author_num_reviews, author_playtime_forever, author_playtime_last_two_weeks, author_playtime_at_review, author_last_played, timestamp_created, comment_count, steam_purchase, received_for_free, written_during_early_access, clean_review, clean_review_length, clean_review_features

Features

Column	**Data Type**	**Variable Type**	**FE treatment**
Game	String	Categorical	StringIndexer/OHE
author_num_games_owned	Integer	Continuous	Standard Scaler
author_num_reviews	Integer	Continuous	Standard Scaler
author_playtime_forever	Integer	Continuous	Standard Scaler
author_playtime_last_two_weeks	Integer	Continuous	Standard Scaler
author_playtime_at_review,	Integer	Continuous	Standard Scaler
author_last_played	Long	Continuous	Standard Scaler
timestamp_created	Long	Binary	Standard Scaler
comment_count	Integer	Continuous	Standard Scaler
steam_purchase	Integer	Binary	Standard Scaler
received_for_free	Integer	Binary	Standard Scaler
clean_review_length	integer	Continuous	Standard Scaler
Clean_review_features	vector	Vector	Passthrough

Label

voted_up	Integer	Binary

I started my feature engineering by importing all the necessary imports needed which include date/time encoders, vector assemblers, string indexers, scalers, etc, as well as evaluation imports and models for our predictions. I also made paths connecting to my google cloud storage buckets.

I read in the cleaned dataset from my last notebook and repartitioned the work to optimize parallelism. We then ran a quick summary on the variables to check their basic statistics and checked if there were any missed nulls before we start feature engineering.

I started by creating a review word_count column as a feature that may add predictive power. We have around 50 million reviews after filtering the reviews to English. I then filtered out short results that would possibly add noise to the sentiment analysis feature I will create.

I also feature engineered 14 predictors from 2 types of dates which is author_last_played and timestamp_created which were unix time stamps, so I was able to get as granular as minutes. After I

wanted the model to get more information out the feature engineered dates, so I cyclically encoded them to help the model interpret it better numerically.

After feature engineering the date predictors, I cast a double type on all variables that weren't strings for when I use VecAssembler and throw the inputs into the model.

I then set up and ran the tokenizer tf, hashingTF and IDF to split words in the review up and count each one using the hash tf function, then weigh the word and penalizes very common or non-value words. This operation gives us 2 more predictors(clean_review_tf, clean_review_features) that will be used for a sentiment analysis using textblob.

I then imported textBlob, created a function for sentiment analysis and ran it through a UDF to be able to partition and parallel process it using spark.

I dropped all non-numeric columns except games, which I got a list of the top 200 games, string indexed them, then OneHotEncoded them as preprocessing before throwing into the vector assembler.

For the last feature I engineered called age_of_review_days which gets the absolute difference between author last played and review created timestamps

I renamed the voted_up variable to label, so the model picks it up automatically. I also made a list of continuous columns I fed into the standard scaler and proceeded to add them to the vector assembler and combined it with the OHE in the final VecAssembler, completing my ML pipeline. We then specified the Logistic regression model as well as an areaUnderROC metric as an evaluator.

I then assembled the pipeline and then split the date into 70% training 30% test. I also created a grid to figure out which regularization hyperparameters were optimal for this dataset, as well as attempt to prevent some overfitting.

I created a cross validator to run my model against subset of my data and average out metrics to choose the best model and run cv.fit to initiate the pipeline. With 50 million records this took the longest out of everything to run/process(a few hours).

After I trained the model on the training data we take the best model and run it with unseen test data and evaluated areaUnderROC, which got a score of 0.8885.

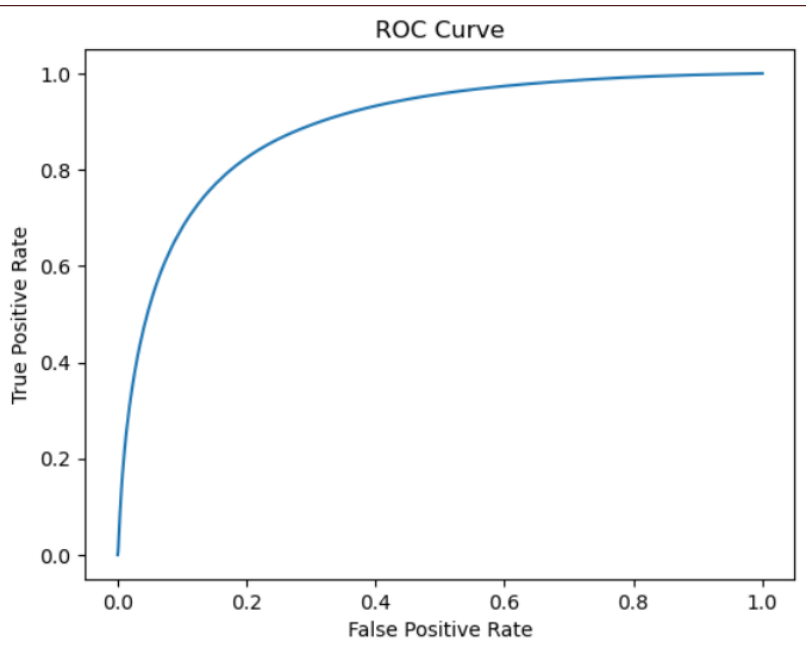
1 being perfect discernment of the predictions, 0.5 being no better than random, and <0.5 that being worse than random.

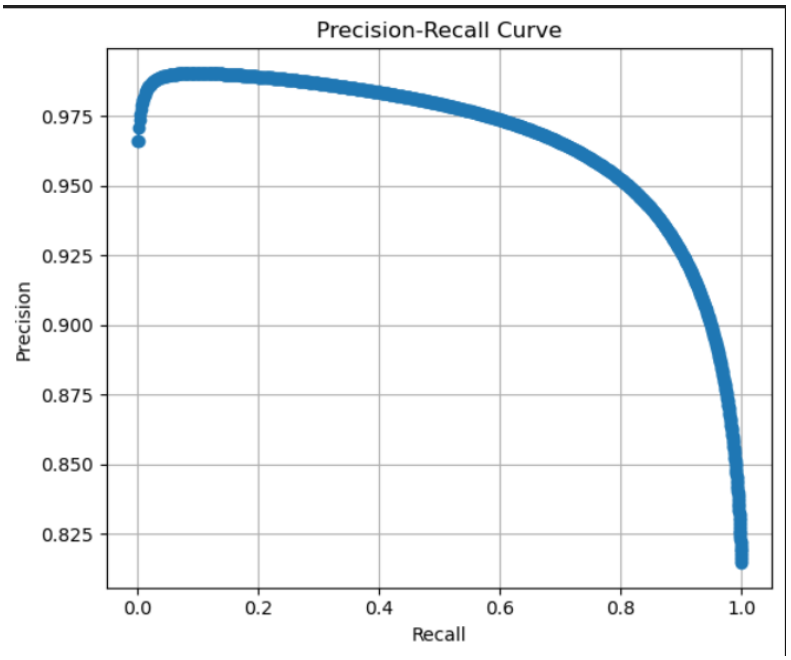
We create the confusion matrix and determine accuracy, precision, recall and f1-score which were take the values of the following numbers respectively: 0.8714, 0.8877, 0.9639, 0.9243

Lastly, we save the best model in our cloud storage bucket. The biggest challenge for me during feature engineering was trying to initialize the textblob throughout the clusters and attempting to debug the pipeline after calling cv.fit() since it takes hours, and some bugs don't cause an error until an hour into processing making it a very time intensive and irritating process!

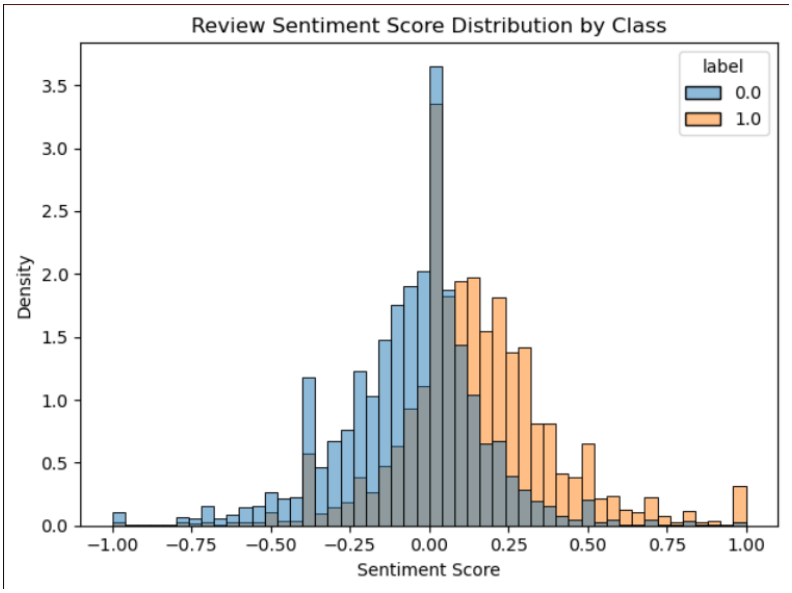
Milestone 5

I ended up creating 4 visualizations, 2 for better understanding evaluation and 2 for going deeper into data after feature engineering and modeling.





The ROC curve hugs the top left corner indicating good model performance, the precision-Recall curve shows a high start in precision which declines smooth and gradually showing the model is well calibrated.



The histogram shows that the clean_review_sentiment feature generally aligns with the review label — negative reviews are clustered around negative sentiment scores, and positive reviews tend toward the positive side. However, a noticeable overlap in the -0.1 to +0.2 range suggests that sentiment alone may not be sufficient for perfect classification, motivating the use of additional features or more sophisticated modeling approaches.

Top 5 Most Influential Features:

game_ohe_Overwatch® 2: -3.506990

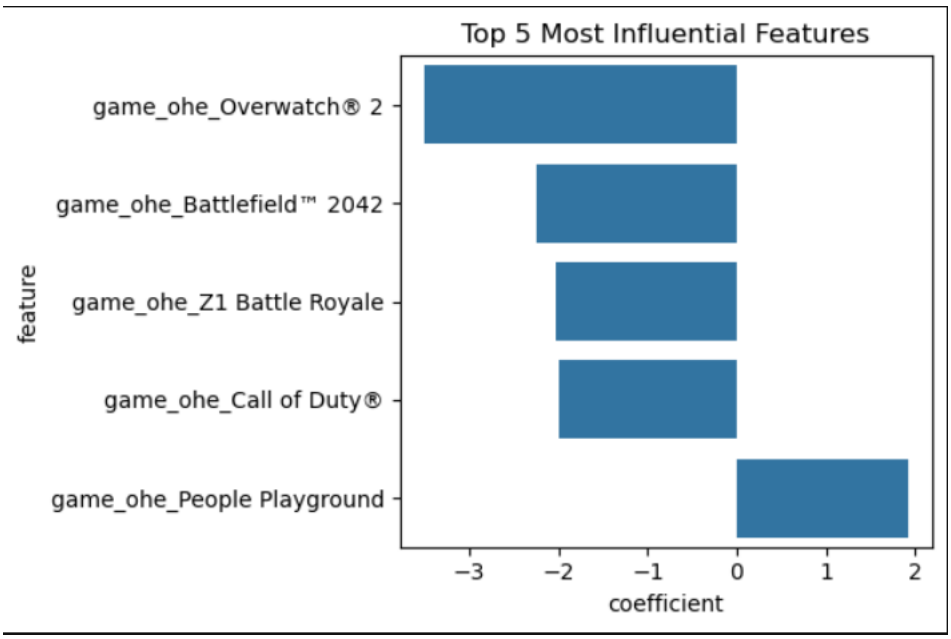
game_ohe_Battlefield™ 2042: -2.243893

game_ohe_Z1 Battle Royale: -2.035466

game_ohe_Call of Duty®: -2.000703

game_ohe_People Playground: 1.920801

This bar graph visually represents the top 5 most influential features for the model.



Summary and Conclusions

This project focused on building a predictive model using PySpark and Google Cloud Provider to predict whether a future review would be positive or negative based on other characteristics of the account.

I used a large dataset of over 100 million reviews from Kaggle(47GB). I stored, processed and analyzed using Dataproc and Spark's distributed computing and GCP's computing(clusters) as well as storage(Buckets).

The steps to building my pipeline were:

Data acquisition & Storage
- Boot up a virtual machine using GCP
- Download dataset to VM via cURL command and unzip
- Use VM to upload data into a structured folder for the project in GCP bucket

EDA
- We learned the top 5 games include Counter Strike 2, PUB G, GTA V
- Outliers were assessed and dealt with

Data Cleaning
- Removed or imputed nulls depending on the variable
- Filtered data to reviews only in english
- Converted some NaN values that weren't getting removed to null

Feature Engineering- Created new features:
- Clean_review_length
- Age_of_review_days
- Cyclical encoding for date features
- TF-IDF vector from clean_review
- Sentiment Scores from TextBlob
- Applied StringIndexer, OneHotEncoder, StandardScaler to prepare features
- Assembled features using VectorAssembler

Model Training & Evaluation
- Trained a Logistic Regression model with cross validation and hyperparameter tuning
- Stored the final model in GCP bucket for deployment or future use
- Evaluated models using
  - ROC AUC: 0.8885
  - Accuracy: 0.8714
  - Precision: 0.8877
  - Recall: 0.9639
  - F1-Score 0.9243

Visualizations
- Created a plot to visualize sentiment analysis density with label to see if the sentiment aligns with the user feedback.
- Precision-Recall curve to make sure our model is well calibrated

- Bar chart of top 5 influential features

Challenges:

Many initial crashes, getting a handle on resources needed for this huge dataset was very challenging. Between optimizing cluster settings to repartitioning, it took a chunk of the time of experimenting with this project to optimize settings for proper analysis.

Debugging pipelines, especially for huge datasets like this one was a very lengthy and time-consuming process since training took hours.

Setting up the initialization for textblob across the cluster was challenging the first time around.

Conclusion:

Using logistic Regression and feature engineering, we can accurately predict steam review sentiment. We learned that playtime behavior and games themselves are strong indicators of sentiment(Voted_up).

This end-to-end project demonstrated knowledge stemming from the beginning of a data pipeline, I.E. data engineering, to a machine learning model ready for testing and deployment. I successfully used big data tools, distributed systems, and ML pipelines to extract meaningful insights from a massive, uncleaned dataset. The model that has been saved as a result from all the work stated above can be used for helping game developers monitor and proactively respond to community feedback at scale.