



ILLINOIS TECH

Project Proposal

CSP 554: Big Data Technologies

By

Project 3-Team A

Name	A_ID	Email
Divyansh Soni	A20517331	dsoni2@hawk.iit.edu

Under the Guidance of Prof. Joseph Rosen

1 Introduction to the Project Use Case

Assuming we work at a small e-commerce company that sells a variety of clothes online, our customers provide feedback across all online channels. The feedback data which is product review is collected for sentiment analysis purposes. The product review on our e-commerce website can be viewed by users. The accessibility of feedback on our e-commerce website reduces the level of data privacy concerns. Since the reviews are openly available for viewing, the inherent transparency minimizes potential privacy issues associated with the review data. When we build a big data machine learning system for sentiment analysis, we may not need to be overly concerned about user privacy.

As a business, we aim to quickly capture this customer feedback to identify any changes in market trends or customer behavior and to be promptly alerted about potential product issues. Our task involves building an NLP model that takes these product reviews as input. We will use a model to classify the sentiment of the reviews into three classes: positive, neutral, and negative. The sentiment class is typically expressed as an integer value for model training, such as 1 for positive sentiment, 0 for neutral sentiment, and -1 for negative sentiment.

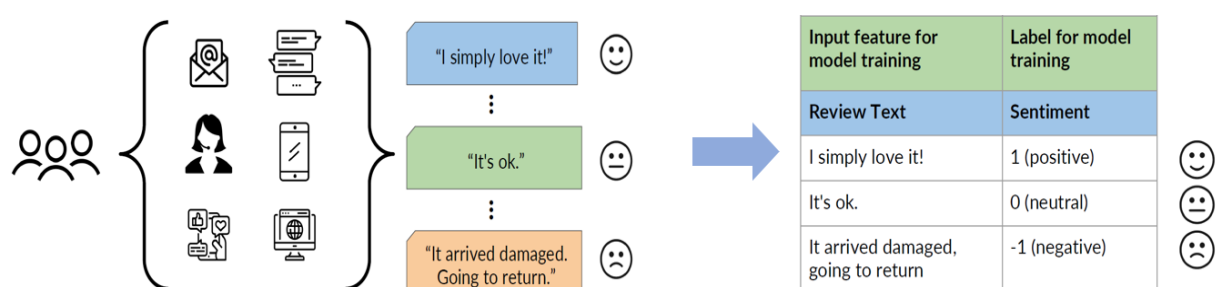


Figure 1.1 Use Case: sentiment analysis for product reviews

In this case, we are attempting to ascertain whether a particular product review is positive (1), neutral (0), or negative (-1). We have three distinct classes that we aim to predict. Therefore, in this scenario, we are dealing with a classification problem—specifically, the

classification of reviews into categories, which is a common text classification task. Text analysis, or natural language processing (NLP), has been around for a long time. In fact, early work dates back to the late 1940s and the early 1950s, and there have been significant advancements over the last decade (Feng, Z, 2023). However, the field has experienced tremendous progress, primarily driven by the evolution and improvement of machine learning algorithms for text analysis.

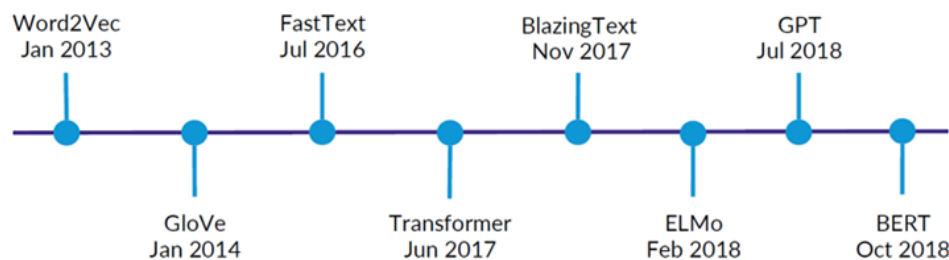


Figure 1.2 Recent Evolution of ML Algorithms for Text Analysis

We will build a big data machine learning system by developing a natural language processing model based on BERT

2 Big Data Machine Learning System

In this section, we will delve into the complexities and solutions of constructing a Big Data Machine Learning system using AWS SageMaker. This section provides a comprehensive understanding of the tools and techniques essential for constructing an efficient Big Data Machine Learning system for sentiment analysis.

2.1 Comparative Analysis of Popular End-to-End ML Platforms

There are several popular end-to-end machine learning platforms that are widely used for developing big data machine learning systems such as Amazon SageMaker, Google Cloud Machine Learning Engine (GCP ML Engine), and Microsoft Azure Machine Learning (Azure ML). End-to-end machine learning platforms provide a unified environment for managing the entire machine learning lifecycle, from data preparation and model development to deployment and monitoring (Brumbaugh, Eli, et al, 2019). This can help to streamline the development process and make it easier to build and deploy big data machine learning systems. Amazon SageMaker, Google Cloud Machine Learning Engine (GCP ML Engine), and Microsoft Azure Machine Learning (Azure ML) are all cloud-based machine learning (ML)

platforms that provide a set of tools and services to build, train, and deploy machine learning models. Here is a brief comparison of the 3 popular cloud-based platforms for developing and deploying machine learning models.

Feature	Amazon SageMaker	GCP ML Engine	Azure ML
Deployment options	pre-built models, SageMaker Studio, SageMaker Canvas	Cloud Functions, Vertex AI Pipelines, AI Platform Notebooks	Azure Functions, Azure Machine Learning Studio, Azure Machine Learning Notebooks
Training frameworks	TensorFlow, PyTorch, MXNet, Chainer, CNTK, XGBoost, SparkML	TensorFlow, PyTorch, Scikit-learn, XGBoost, SparkML	TensorFlow, PyTorch, Scikit-learn, CNTK, XGBoost
Data preparation and pre-processing	SageMaker Studio, Data Wrangler, SageMaker Autopilot	AI Platform Notebooks, AI Platform Dataflow	Azure Data Factory, Azure Machine Learning Studio, Azure Machine Learning Notebooks,
Model management and monitoring	SageMaker Studio, Model Monitor, SageMaker Experiments	AI Platform Notebooks, AI Platform Prediction, AI Platform Monitoring	Azure Monitor, Azure Machine Learning Studio, Azure Machine Learning Experimentation Service
Scalability	Auto-scaling of compute resources, support for large-scale deployments	Elastic scaling of compute resources, support for large-scale deployments	Auto-scaling of compute resources, support for large-scale deployments
Pricing	Pay-as-you-go pricing based on compute usage and storage	Pay-as-you-go pricing based on compute usage and storage	Pay-as-you-go pricing based on compute usage and storage

Figure 2.1.1 Comparison of the Popular Cloud-based ML Platforms

In the brief comparison, we found that there is no definitive best end-to-end machine learning platform, as all platforms offer comparable features. After carefully considering the various cloud machine learning platforms and taking into account our existing familiarity with the Amazon Web Services environment, we have decided to use Amazon SageMaker for developing the product review sentiment analysis big data machine learning system.

2.2 Comparative Analysis of Popular AI Frameworks

When it comes to AI frameworks for implementing a big data machine learning system, there are two popular frameworks available for various supervised and unsupervised machine learning tasks, including but not limited to image classification, computer vision, and speech processing. TensorFlow and PyTorch are both popular deep learning frameworks that are widely used for natural language processing (NLP). Here we briefly introduce and compare the popular AI frameworks based on the information on the official websites.

(1) TensorFlow:

TensorFlow is developed by Google. It is an open-source machine learning framework widely used for tasks such as deep learning, neural network implementations, and numerical computations (Wikipedia, 2023). TensorFlow also has a large and active community, which means that there are many resources available to help us get started.

(2) PyTorch:

PyTorch developed by Facebook's AI Research lab (FAIR), is known for its dynamic computational graph, making it popular for research in deep learning and natural language processing (Wikipedia, 2023). PyTorch also has a growing community, and there are many resources available to help us learn the framework.

Each framework has its own strengths and weaknesses. Below is a brief comparison of TensorFlow and PyTorch based on information from the official websites and reviews from various technical blogs.

Feature	TensorFlow	PyTorch
Maturity	Mature	Newer
Scalability	Highly scalable	Less scalable
Ease of use	More complex	Easier to use
Flexibility	Less flexible	More flexible

Figure 2.2.1 Comparison of the Popular ML Frameworks

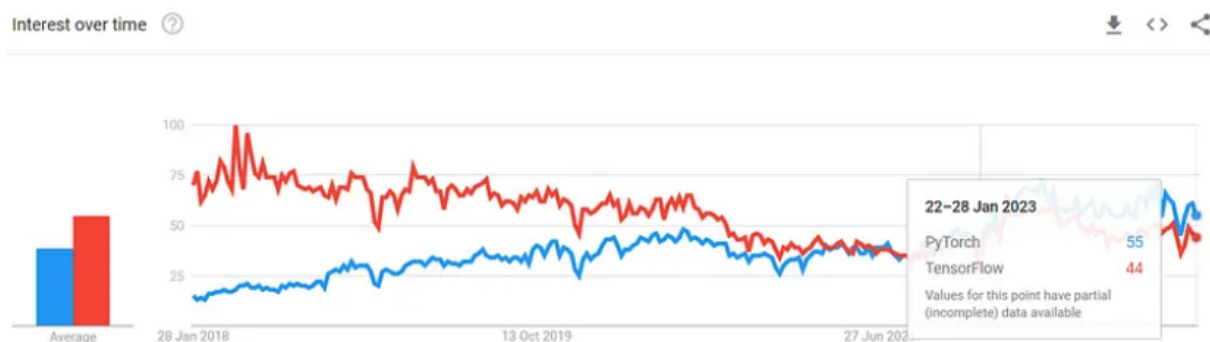


Figure 2.2.2 Comparison of the Popular ML Frameworks

PyTorch and TensorFlow have experienced significant increases in popularity, likely attributed to the emergence of transfer learning and the availability of pre-trained models offered by Hugging Face. Nevertheless, it appears that PyTorch boasts superior model availability (K, Valantis, 2023) for big data machine learning. Upon comparing two popular AI frameworks, PyTorch emerges as the most suitable choice for building a product review sentiment analysis big data machine learning system.

2.3 Comparative Analysis of Toolkits for Data Profiling

Except for these popular machine learning frameworks, there is a popular machine learning library called Spark MLlib. Spark MLlib is the machine learning library provided by Apache Spark, an open-source distributed computing system (Apache Spark, 2023). MLlib is designed to work seamlessly with Spark and enables scalable and distributed machine learning on large

datasets. It provides various algorithms and tools for machine learning tasks, including classification, regression, clustering, and collaborative filtering. In comparison to Amazon SageMaker within the Amazon Web Services ecosystem, and considering our existing familiarity with the Amazon Web Services environment and its ease of use feature, we have opted to utilize AWS Glue and AWS Athena in SageMaker platform for data profiling on the product review dataset. Below is a brief comparison.

Feature	AWS Glue	AWS Athena	SparkMLlib
Data sources	Supports a wide variety of data sources, including S3, RDS, DynamoDB, and Redshift	Supports S3, but can also be integrated with other data sources through custom connectors	Supports a wide variety of data sources, including S3, HDFS, and Hive
Profiling capabilities	Provides a comprehensive set of data profiling capabilities, including data type detection, cardinality analysis, and distribution analysis	Provides a limited set of data profiling capabilities, primarily focused on data type detection and basic statistics	Provides a comprehensive set of data profiling capabilities, including data type detection, cardinality analysis, distribution analysis, anomaly detection, and feature engineering
Scalability	Can scale to handle large datasets	Can scale to handle large datasets	Can scale to handle large datasets
Ease of use	Easy to use, with a visual interface and a drag-and-drop ETL editor	Easy to use, with a SQL-like query language	More complex to use than AWS Glue or AWS Athena, but requires more programming expertise

Figure 2.3.1 Comparison of the Toolkits for Data Profiling

There are two useful Amazon Cloud Services in the end-to-end ML platform (Amazon SageMaker) that we can use to detect statistical bias—one focus of data profiling—with AWS Glue and AWS Athena: Amazon SageMaker Data Wrangler and Amazon SageMaker Clarify.

(1) Amazon SageMaker Data Wrangler

Data Wrangler provides us with the capability to connect to various data sources, visualize data, and transform data by applying numerous transformations within the Data Wrangler environment (AWS, 2023). It enables us to detect statistical bias in datasets and generate reports regarding the detected bias. Additionally, it offers capabilities to perform feature importance calculations on the training dataset. Data Wrangler provides us with more of a UI-based visual experience. Data Wrangler is only using a subset of the data to detect bias in that dataset.

(2) Amazon SageMaker Clarify

Amazon SageMaker Clarify serves as a tool for conducting statistical bias detection on datasets. It has the capability to detect statistical bias and generate bias reports for training datasets (AWS, 2023). Moreover, it can identify bias in both trained and deployed models. In addition, SageMaker Clarify provides features for machine learning explainability and can detect drift in both data and models. It adopts more of an API-based approach and allows us to scale out the bias detection process. Specifically, Clarify employs a construct known as processing jobs, enabling us to configure a distributed cluster for executing the bias detection

job at scale.

In comparison, we have chosen to use SageMaker Clarify for bias detection.

3 Data Preparation and Profiling

In Section 3, we analyze pivotal aspects of data transformation, bias, and imbalance within the product review sentiment analysis big data machine learning project. With a processed dataset focusing on 'review_body,' 'product_category,' and 'sentiment,' we explore potential biases and imbalances that could impact model performance. Our focus lies on class imbalance and the Difference in Proportions of Labels (DPL) metric, addressing disparities in product category ratings. Additionally, we emphasize the significance of data profiling, advocating for the use of AWS Glue and AWS Athena over traditional tools. This section unveils a comprehensive analysis of sentiment data distribution, guiding us towards a nuanced understanding of the dataset's dynamics for more informed big data machine learning model development.

3.1 Data Transformation, Bias and Imbalance

In the product review dataset, comprising 10 features, we selected three features highly relevant to product reviews. Subsequently, we transformed the data into a comma-separated values (CSV) file, including only the 'review_body,' 'product_category,' and 'sentiment,' derived from the original dataset.

	sentiment	review_body	product_category
0	1	If this product was in petite i would get the...	Blouses
1	1	Love this dress! it's sooo pretty. i happene...	Dresses
2	0	I had such high hopes for this dress and reall...	Dresses
3	1	I love love love this jumpsuit. it's fun fl...	Pants
4	1	This shirt is very flattering to all due to th...	Blouses
...

Figure 3.1.1 Data Transformation

A dataset is deemed biased if it cannot fully and accurately represent the underlying problem space. The product review dataset could be statistically biased if it contains a disproportionately large number of reviews for one product category. Statistical bias refers to the tendency of a statistic to either overestimate or underestimate a parameter, reflecting

imbalances in these training datasets (Black, Anne C., et al, 2011). The imbalances in training datasets potentially lead to suboptimal model performance and skewed results. Imbalances can lead to biased model training, where the model may exhibit a preference for the majority class, as it is more heavily represented in the training data (Branco, Paula, et al, 2017). There are several metrics that can be used to measure different portions of bias across the datasets. In this big data machine learning project, we have chosen to focus on detecting imbalances in the training dataset, given that this machine learning system is designed for text classification. The first metric is a class imbalance, which refers to an unequal distribution of instances among different classes in a classification dataset (Luque, Amalia, et al, 2019). When we apply the class imbalance metric to the product review dataset, it addresses this specific question: Is there a significantly higher number of total reviews for a specific product category compared to any other category in the dataset? The second metric we adopt is Difference in Proportions of Labels (DPL). The Difference in Proportions of Labels (DPL) is a metric that can be used to quantify disparities in the distribution of class labels across different groups or subpopulations within a dataset (Das, Sanjiv, et al, 2021). It's particularly relevant in scenarios where we want to assess if there are significant differences in the prevalence of certain outcomes among different groups.

$$DPL = |P_1 - P_2|$$

where P_1 and P_2 are the proportions of positive outcomes in two different groups, we aim to employ DPL (Disproportionate Impact on Ratings) on the product review dataset to assess whether a specific product category exhibits ratings that are disproportionately higher than those in other categories.

3.2 Data Profiling

Data profiling is the process of examining and summarizing data to understand its structure, content, and quality (Wikipedia, 2022). It is a crucial step in data analysis and machine learning, as it helps to identify potential problems with the data before it is used for modeling or decision-making. There are three key benefits of data profiling (Wikipedia, 2022):

1. Improved data quality: Data profiling can help to identify and correct errors and inconsistencies in data, which can improve the accuracy of downstream analysis.
2. Increased understanding of data: Data profiling can provide insights into the structure and content of data, which can help to inform data modeling and decision-making.
3. Reduced risk of bias: Data profiling can help to identify potential biases in data, which can help to prevent biased outcomes in machine learning models.

Data profiling can be performed using a variety of tools and techniques, and the specific steps involved will vary depending on the type of data and the goals of the analysis. The toolkits for data profiling we've discussed in section 2.3. In the big data machine learning system for product review sentiment analysis, we choose to understand and prepare data for model training by focusing on comprehending sentiment data distribution (3.2.2 Sentiment Data Distribution), detecting bias (3.2.3 Bias Detection), and determining feature importance (3.2.4 Feature Importance).

3.2.1 Rethinking Data Profiling: The Limitations of Pandas and Numpy

When dealing with big data in a machine learning system, it's not recommended to utilize pandas, numpy to do data profiling. The libraries of pandas, numpy have limitations when dealing with big data (Singh, Aishwarya, 2018).

Pandas Limitations

1. **Memory Overhead:** Pandas dataframes can consume significant memory, especially when dealing with large datasets. This can pose challenges for profiling operations on memory-constrained systems.
2. **Limited Data Types:** Pandas natively supports a limited range of data types, such as integers, floats, and strings. This can restrict profiling capabilities for non-standard or complex data types.
3. **Data Type Inference Issues:** Pandas may misinterpret data types, leading to profiling inconsistencies. For instance, numerical data might be inferred as categorical, affecting profiling accuracy.
4. **Handling Missing Values:** Pandas' handling of missing values can complicate profiling tasks. Missing values may not be explicitly represented, making it difficult to identify and analyze their impact on data distribution.

NumPy Limitations

1. **Data Structure Limitations:** NumPy arrays are restricted to homogeneous data, making it challenging to profile datasets with mixed data types. This can hinder comprehensive data analysis.
2. **Metadata Absence:** NumPy arrays lack metadata, such as column names or row labels, making it difficult to interpret and profile the data contextually. This can lead to profiling ambiguities.
3. **Profiling Inefficiencies:** NumPy arrays are not optimized for profiling tasks, requiring more complex and time-consuming operations to extract meaningful insights from the

data.

AWS Glue and AWS Athena are Amazon Cloud Services which offer several advantages for handling large datasets and performing data profiling tasks:

1. **Scalability:** AWS Glue and AWS Athena are designed to handle massive datasets efficiently, making them suitable for processing and analyzing large volumes of product reviews. They can scale seamlessly to accommodate increasing data volumes without performance bottlenecks.
2. **Performance:** AWS Glue and AWS Athena are optimized for performance, enabling them to process large datasets quickly and efficiently. They utilize parallel processing and distributed computing to accelerate data profiling operations.
3. **Data Integration:** AWS Glue and AWS Athena integrate seamlessly with other AWS services, such as Amazon S3 for data storage and Amazon Redshift for data warehousing. This seamless integration facilitates data ingestion, transformation, and analysis workflows.
4. **Cost-effectiveness:** AWS Glue and AWS Athena offer a cost-effective solution for data profiling, as you only pay for the resources you consume. This eliminates the need to purchase and maintain expensive hardware or software licenses.
5. **Ease of Use:** AWS Glue and AWS Athena provide a user-friendly interface and support SQL queries, making them accessible to both data scientists and non-technical users.

In comparison, we chose to utilize AWS Glue and AWS Athena to do data profiling.

3.2.2 Sentiment Data Distribution and Analysis

Sentiment data distribution analysis is a crucial step in building a robust and effective sentiment analysis machine learning system. Understanding the distribution of sentiment labels in the dataset provides valuable insights that can significantly impact the performance and generalization ability of the model.

(1) Sentiment Distribution: Product Review Count by Sentiment

When conducting sentiment distribution analysis for product reviews, we typically want to examine the count of reviews for each sentiment category. This analysis provides a clear overview of how sentiments are distributed across the dataset. We want to know how many reviews per sentiment.



Figure 3.2.2.1 Sentiment Distribution: Review Count by Sentiment

We can see:

- There are 2,370 reviews labeled as "Negative" (sentiment score of -1).
- There are 2,823 reviews labeled as "Neutral" (sentiment score of 0).
- There are 17,433 reviews labeled as "Positive" (sentiment score of 1).

It's evident that positive reviews significantly outnumber negative and neutral ones, which appears favorable for the e-commerce company's business. We cannot consider it a significant imbalance among the classes. Hence, we will proceed with bias detection subsequently in section 3.2.3 to get more insights.

(2) Sentiment Distribution Across Product Categories: Identifying the Highest Average Ratings

It's important to consider the number of reviews in each category. A category with a high average sentiment score but a small number of reviews may not be as reliable as a category with a slightly lower score but a larger number of reviews. To understand which product categories are highest rated by average sentiment. We group the dataset by product category and calculate the average sentiment score for each category, compute the average sentiment score for each product category by taking the mean of the sentiment scores for all reviews within that category and then rank the product categories based on their average sentiment scores

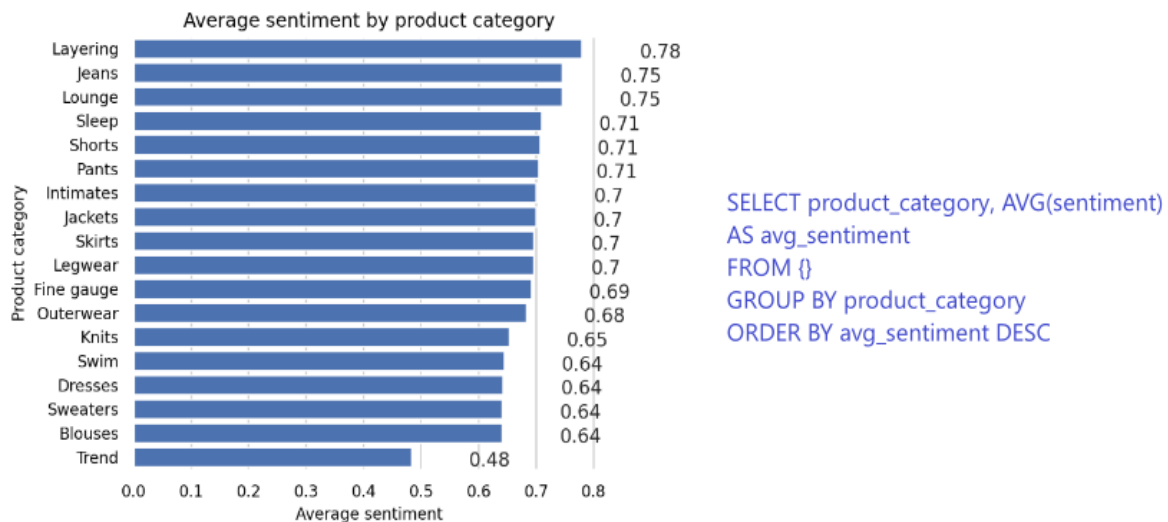


Figure 3.2.2.2 Sentiment Distribution Across Product Categories

(3) Product Review Distribution: Analyzing Categories with the Most Reviews

This information about which product categories have the most reviews can be valuable for understanding the popularity and engagement levels of different product categories within the dataset. It also provides context when analyzing sentiment distribution or identifying the highest-rated categories based on average sentiment. Below is the logic we find which product categories have the most reviews:

- 1) Aggregate Data by Product Category: Group the dataset by product category.
- 2) Count the Reviews: Calculate the total number of reviews for each product category.
- 3) Rank Product Categories by Review Count: Rank the product categories based on the total number of reviews. The categories with the highest review counts will be at the top of the ranking.

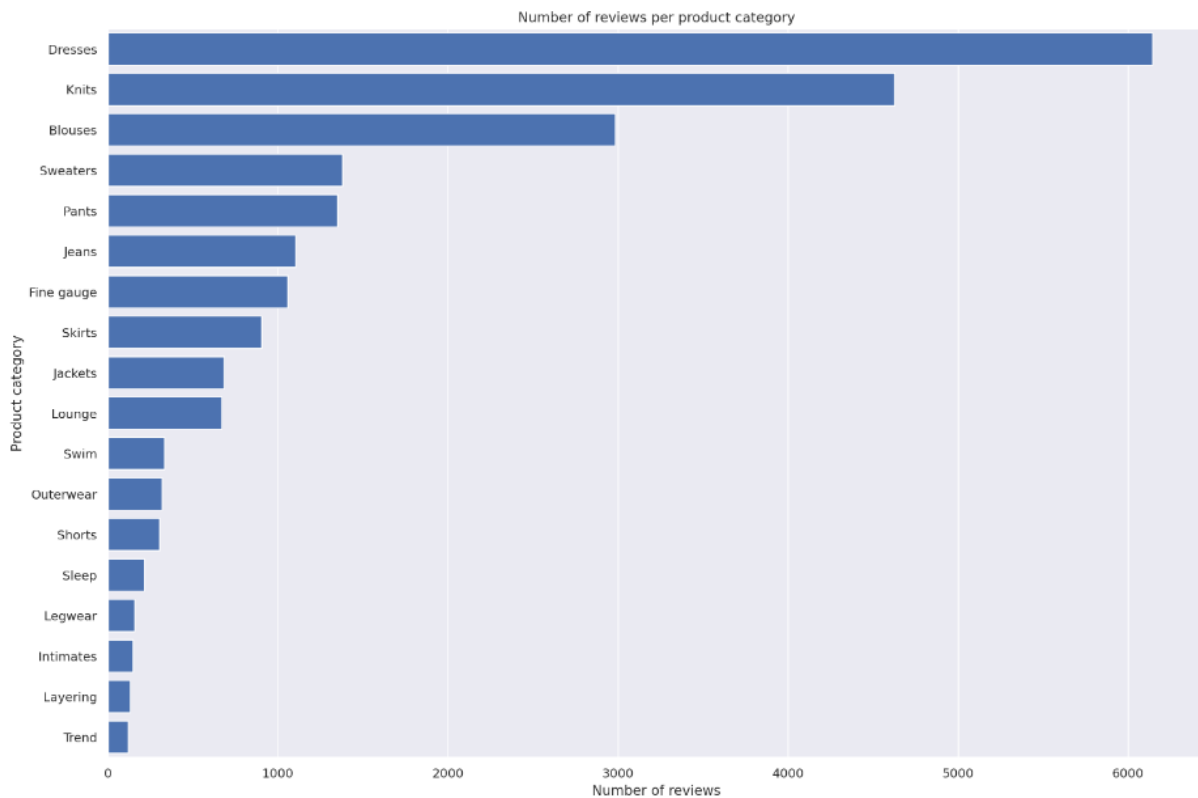


Figure 3.2.2.3 Sentiment Distribution (most reviews) Across Product Categories

(4) Analyze the Distribution of Reviews per Sentiment per Category

Visualizing the distribution of reviews per sentiment per category can enhance the understanding of how sentiments are distributed across various product categories, helping us identify any patterns or areas that may require further investigation or attention. Below is the logic we visualize the distribution of reviews per sentiment per category:

```
SELECT product_category,
       sentiment,
       COUNT(*) AS count_reviews
FROM {}
GROUP BY product_category, sentiment
ORDER BY product_category ASC, sentiment DESC, count_reviews
```

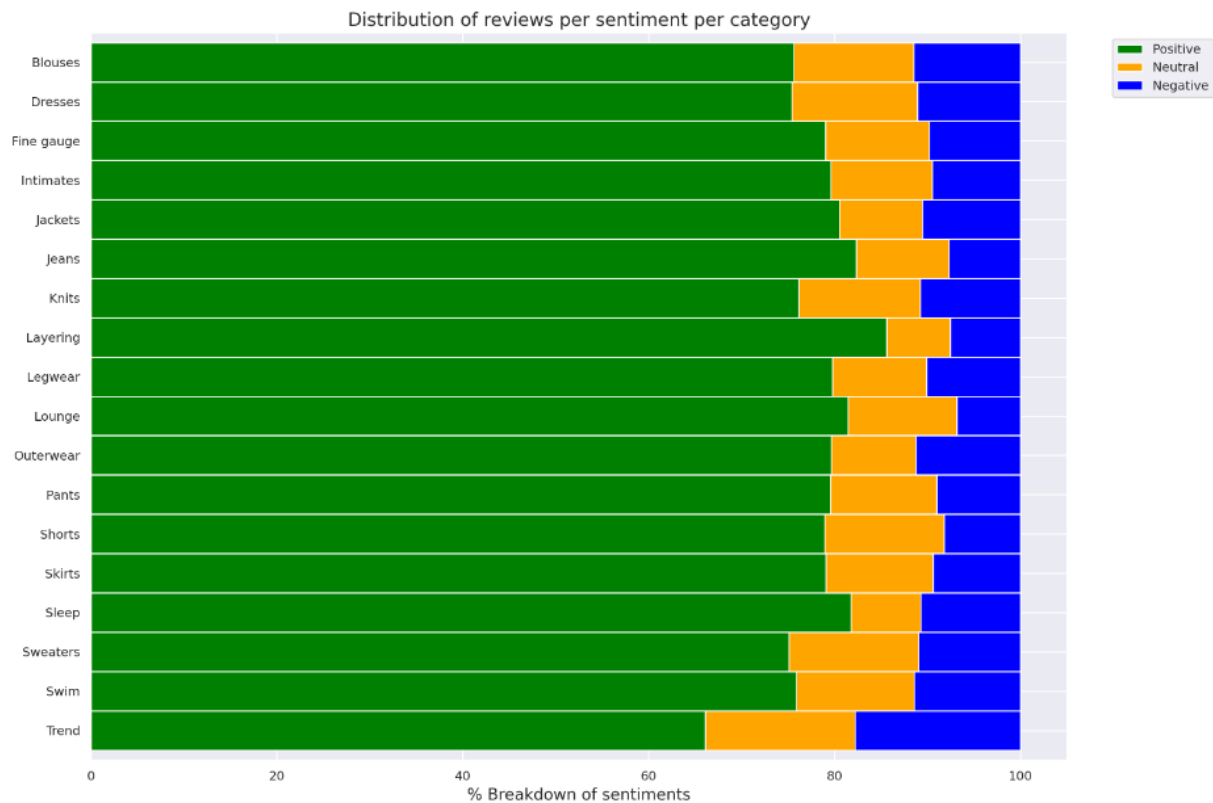


Figure 3.2.2.4 Distribution of Reviews per Sentiment per Category

3.2.3 Bias Detection and Analysis

The sentiment distribution: review count by sentiment in section 3.2.2 shows us that positive reviews significantly outnumber negative and neutral ones. For better visualization, we chose relatively small subset of data to visualize the occurring imbalances. We utilized the Seaborn library to create a count plot based on the 'sentiment' and 'product_category' columns.

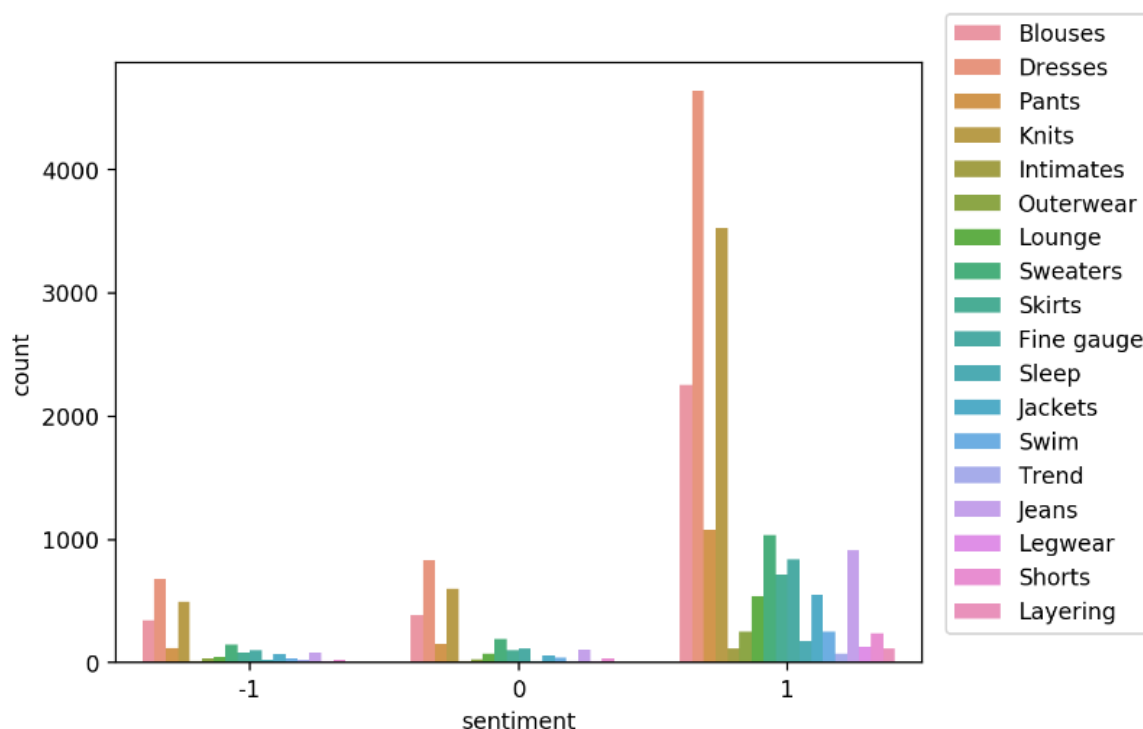


Figure 3.2.3 Distribution of Sentiments Across Product Categories

We can see that there are way more positive reviews than negative or neutral. Such a dataset is called class imbalance. In the context of sentiment analysis or any classification task, an unbalanced dataset can pose challenges for machine learning models. Models trained on imbalanced datasets may become biased towards the majority class, leading to poor performance on minority classes. Visualizing class imbalances in a smaller subset of data can provide a quick and intuitive understanding of the distribution of classes. However, when working with larger datasets, conducting a more formal bias analysis becomes crucial. Bias detection and analysis involves a deeper investigation into how different classes are represented in the dataset, especially in the context of the real-world application. Below is the fundamental steps we do bias detection for conducting this big data machine learning system on Amazon SageMaker platform:

Step 1: Configure a DataConfig

Provide information about the input data to the processor using the DataConfig of the Clarify container. This includes details about the dataset to be analyzed, such as the dataset file, its format, headers, and labels.

Step 2: Configure BiasConfig

Measure bias by calculating a metric and comparing it across groups. Specify the required information in the BiasConfig API. SageMaker Clarify needs information on sensitive columns (facet_name) and desirable outcomes (label_values_or_threshold). For example, the

sensitive facet may be `product_category`, and the desired outcome could be when `sentiment==1`.

Step 3: Configure Amazon SageMaker Clarify as a Processing Job

Construct a `SageMakerClarifyProcessor` object to scale the process of data bias detection. Set parameters such as `instance_count` (representing the number of nodes in the distributed cluster) and `instance_type` (specifying processing capability for each node, including compute and memory capacity).

Step 4: Run the Amazon SageMaker Clarify Processing Job

Execute the configured processing job to compute the requested bias metrics for the input data.

Step 5: Output the Bias Report and Analysis

Review and analyze the output, which typically includes a bias report covering various metrics related to bias. This report provides insights into the presence of bias in the dataset and helps in understanding and mitigating biases in the machine learning model. The bias report (unbalance) is uploaded to the submission with code scripts together.

In the following steps, we need to address bias. We balance the dataset based on `product_category` and `sentiment`. The bias detection and analysis procedure after balancing the dataset by `product_category` and `sentiment` is the same as the fundamental steps used for bias detection. The bias report (balance) is uploaded to the submission with code scripts together. We opt not to delve into bias analysis in this section as the generated reports are more detailed and comprehensive.

4 Train and Tune Models on Amazon SageMaker

In our pursuit of a robust big data machine learning system for product review sentiment analysis, we made strategic decisions to guide us on building a big data machine learning system. In Section 4.1, the spotlight shifted to hyperparameter tuning, comparing grid search and random search strategies. Opting for efficiency, scalability, and robustness, we selected Amazon SageMaker Automatic Model Tuning (AMT) with a random search strategy. In section 4.2, we conduct model train evaluation and monitoring.

4.1 Hyperparameter Tuning

In the process of training models in a big data machine learning system, hyperparameter tuning emerges as a crucial step aimed at optimizing model performance. This optimization may entail achieving the highest accuracy or minimizing errors to attain the best quality model. However, hyperparameter tuning is typically a time-consuming and compute-intensive process (Bischi, Bernd, et al, 2023). There are some popular algorithms that can be used for automated model tuning, such as grid search algorithm, and random search algorithm (Zahedi, Leila, et al, 2021). The grid search algorithm and random search algorithm both work reasonably well in smaller search bases, also both fail fairly quickly when the search space grows beyond a certain size (Bischi, Bernd, et al, 2023). The grid search algorithm tests every combination by training the model with each set of hyperparameters and selecting the best possible parameters. The advantage of grid search is that it enables us to explore all potential combinations. This approach works particularly well when dealing with a small number of hyperparameters and a limited range of values to explore for each hyperparameter. Nevertheless, as the number of hyperparameters or the range of values to explore for these hyperparameters increases, this process can become significantly time-consuming (Alibrahim, Hussain, and Simone A. Ludwig, 2021). In random search, it begins by defining sets of available hyperparameters, comprising the names and values we wish to explore. Instead of exhaustively searching every combination, the algorithm randomly selects hyperparameter values within the defined search space (Zahedi, Leila, et al, 2021). Additionally, stop criteria in the random search strategy, such as elapsed time or a maximum number of completed pieces of training, can be specified. When the stop criteria are met, we select the best-performing set of hyperparameters from the trained models available so far. There are some of the key advantages of random search over grid search for hyperparameter tuning in big data machine learning (Alibrahim, Hussain, and Simone A. Ludwig, 2021):

- Efficiency: Random search significantly reduces the computational cost of hyperparameter tuning, making it more suitable for large-scale problems.
- Scalability: Random search can effectively handle high-dimensional hyperparameter spaces, which often arise in big data machine learning tasks.
- Exploration: Random search avoids the potential biases introduced by the predefined grid in grid search, allowing for a more comprehensive exploration of the hyperparameter space.
- Flexibility: Random search can be easily adapted to different hyperparameter

distributions, making it more versatile for various machine learning models.

- **Robustness:** Random search is less prone to overfitting and local optima compared to grid search, leading to more generalizable models.

Random search is more efficient and less computationally expensive, especially when dealing with large datasets and high-dimensional hyperparameter spaces (Alibrahim, Hussain, and Simone A. Ludwig, 2021). And we don't have the need to test every combination by training the model with each set of hyperparameters. Compared to the grid search strategy, the random search strategy is a better approach for us to do hyperparameter tuning when constructing the big data machine learning system.

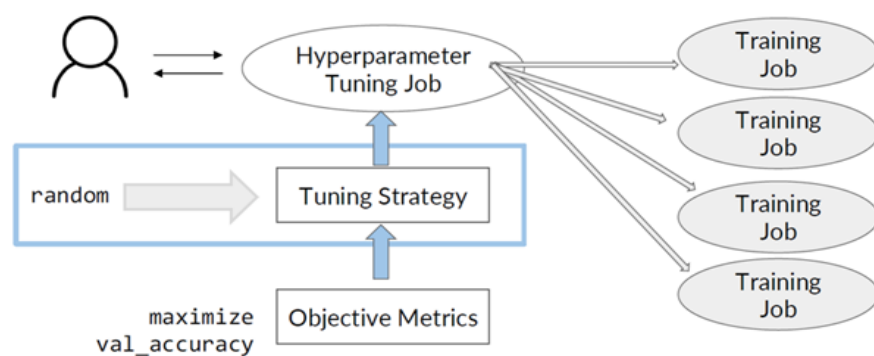


Figure 4.2.1 Hyperparameter Tuning Using Random Search

Amazon SageMaker Automatic Model Tuning (AMT) is a comprehensively managed platform designed for large-scale gradient-free function optimization (Perrone, Valerio, et al, 2021). We can use AMT to do hyperparameter tuning at scale and find the best version of the model by running multiple training jobs on the dataset using the hyperparameter range values that we specify. For tuning strategies, SageMaker natively supports random optimization strategies. By the comparison of grid search and random search, we opted to use the random search supported by Amazon SageMaker for hyperparameter tuning. Here are the steps we do hyperparameter tuning

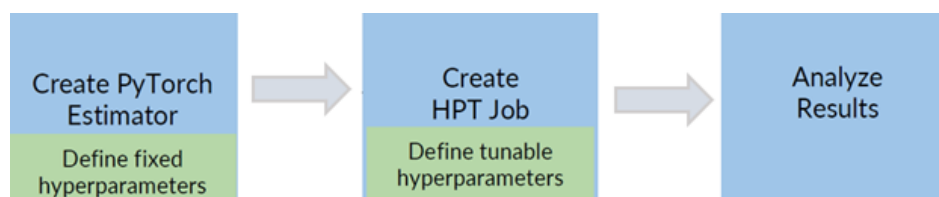


Figure 4.2.2 Basic Steps for Hyperparameter Tuning on SageMaker

TODO: code implementation, and then show the best candidate - the one with the highest accuracy result.

TODO: code implementation, and then evaluate the best candidate (pull the information about the best candidate from the data frame and then take the Training Job name)

TODO: code implementation, and then show accuracy results for the best candidate.

4.2 Model Training, Model Evaluation and Monitoring

Training evaluation and monitoring are critical aspects of developing machine learning models. These processes help ensure that the model is learning effectively, generalizing well to new data, and performing as expected. An important step in model development is evaluating it with a test dataset - one that the model has never encountered during its training process. The final metrics obtained from this evaluation can be used to compare different machine learning models. The higher the value of these metrics, the better the model can generalize.

TODO: code implementation

Reference

- [1] Alibrahim, Hussain, and Simone A. Ludwig. 'Hyperparameter Optimization: Comparing Genetic Algorithm against Grid Search and Bayesian Optimization'. *2021 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, 2021, pp. 1551–59. DOI.org (Crossref), <https://doi.org/10.1109/CEC45853.2021.9504761>.
- [2] Bischl, Bernd, et al. 'Hyperparameter Optimization: Foundations, Algorithms, Best Practices, and Open Challenges'. *WIREs Data Mining and Knowledge Discovery*, vol. 13, no. 2, Mar. 2023, p. e1484. DOI.org (Crossref), <https://doi.org/10.1002/widm.1484>.
- [3] Perrone, Valerio, et al. 'Amazon SageMaker Automatic Model Tuning: Scalable Gradient-Free Optimization'. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge*

Discovery & Data Mining, ACM, 2021, pp. 3463–71. DOI.org (Crossref), <https://doi.org/10.1145/3447548.3467098>.

[4] Zahedi, Leila, et al. *Search Algorithms for Automated Hyper-Parameter Tuning*. arXiv:2104.14677, arXiv, 29 Apr. 2021. arXiv.org, <http://arxiv.org/abs/2104.14677>.

[5] Brumbaugh, Eli, et al. 'Bighead: A Framework-Agnostic, End-to-End Machine Learning Platform'. *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, IEEE, 2019, pp. 551–60. DOI.org (Crossref), <https://doi.org/10.1109/DSAA.2019.00070>.

[6] Feng, Z. (2023). Past and Present of Natural Language Processing. In: Formal Analysis for Natural Language Processing: A Handbook. Springer, Singapore. https://doi.org/10.1007/978-981-16-5172-4_1

[7] Black, Anne C., et al. 'Missing Data Techniques for Multilevel Data: Implications of Model Misspecification'. *Journal of Applied Statistics*, vol. 38, no. 9, Sept. 2011, pp. 1845–65. DOI.org (Crossref), <https://doi.org/10.1080/02664763.2010.529882>.

[8] Branco, Paula, et al. 'A Survey of Predictive Modeling on Imbalanced Domains'. *ACM Computing Surveys*, vol. 49, no. 2, June 2017, pp. 1–50. DOI.org (Crossref), <https://doi.org/10.1145/2907070>.

[9] Das, Sanjiv, et al. 'Fairness Measures for Machine Learning in Finance'. *The Journal of Financial Data Science*, vol. 3, no. 4, Oct. 2021, pp. 33–64. DOI.org (Crossref), <https://doi.org/10.3905/jfds.2021.1.075>.

[10] Luque, Amalia, et al. 'The Impact of Class Imbalance in Classification Performance Metrics Based on the Binary Confusion Matrix'. *Pattern Recognition*, vol. 91, July 2019, pp. 216–31. DOI.org (Crossref), <https://doi.org/10.1016/j.patcog.2019.02.023>

[11] K, Valantis. 'Battle of The Giants: TensorFlow vs PyTorch 2023'. *Medium*, 28 Jan. 2023, <https://medium.com/@valkont/battle-of-the-giants-tensorflow-vs-pytorch-2023-fd8274210a38>

[12] 'PyTorch'. Wikipedia, 4 Nov. 2023. Wikipedia, <https://en.wikipedia.org/w/index.php?title=PyTorch&oldid=1183408410>.

[13] 'TensorFlow'. Wikipedia, 10 Sept. 2023. Wikipedia, <https://en.wikipedia.org/w/index.php?title=TensorFlow&oldid=1174808074>.

[14] 'Bias Detection and Model Explainability – Amazon Web Services'. Amazon Web Services,

Inc., <https://aws.amazon.com/sagemaker/clarify/>. Accessed 20 Nov. 2023.

[15] 'Data Profiling'. Wikipedia, 4 Aug. 2022. Wikipedia, https://en.wikipedia.org/w/index.php?title=Data_profiling&oldid=1102297638.

[16] MLlib | Apache Spark. <https://spark.apache.org/mllib/>. Accessed 20 Nov. 2023.

[17] 'Prepare Data For ML, Data Wrangling Tool - Amazon SageMaker Data Wrangler - AWS'. Amazon Web Services, Inc., <https://aws.amazon.com/sagemaker/data-wrangler/>. Accessed 20 Nov. 2023.

[18] Singh, Aishwarya. 'Ultimate Guide to Handle Big Datasets for Machine Learning Using Dask (in Python)'. Analytics Vidhya, 9 Aug. 2018, https://www.analyticsvidhya.com/blog/2018/08/dask-big-datasets-machine_learning-python/.