Project Report

INNOPOLIS UNIVERSITY

Commodities category prediction

Students	Nesterov Grigoriy Aleksandr Vashchenko Georgy Andryushchenko
Course	MLOps Engineering
Semester	Summer
Year	2024

Contents

1. Introduction

In recent years, marketplaces have taken a significant place in the Russian market. COVID-19 has proven to be one of the most significant drivers of interest growth in such services. In 2023 alone, total expenditures on marketplaces increased by 50%. Niche marketplaces are also showing substantial growth[2].

However, marketplaces are built not only around customers, sellers are equally important. Therefore, it becomes crucial to help sellers choose the most appropriate category for the goods they are trying to sell. This way, customers are more likely to find the product in the category they expect it to be, and sellers do not need to spend a lot of time selecting the best category out of hundreds available.

To address this challenge, our project utilizes the «Online Retail Dataset» from Kaggle. This dataset contains information about purchases in an online store over a certain period, allowing us to conduct an in-depth analysis and identify key patterns that can help improve the category selection process for products.

2. Business and Data Understanding

In this project, the primary business problem is to increase seller loyalty by providing them with product category recommendations based on product features, including its description. It is essential to understand that proper product categorization allows customers to find the desired products more quickly, which increases the likelihood of purchase, and consequently, seller satisfaction and their willingness to continue working with the platform. The current situation shows that sellers often face difficulties in selecting the appropriate category for their products, leading to reduced sales and dissatisfaction. Therefore, we need to answer the question: "Is it possible to predict the product category with 90% accuracy based on a product description?". Using data from the "Online Retail Dataset" on the Kaggle platform will enable us to analyze and identify key parameters for building a product category prediction model [5].

2.1 Terminology

2.1.1 Business terminology

- Product - item offered for a sale on a marketplace;
- Seller - person or organization gaining profit by products sales;
- Product feature - textual, numerical or visual representation of product property;
- Product description - detailed description of the product qualities, design purposes, and features, written by the seller;
- Product card - organized collection of product features containing all sufficient product features which are essential for customers and are mandatory for product publication on a marketplace;
- Product category - product feature that is used by marketplace to organize the catalogue of products;
- Filtering system - set of manually designed and updated rules ensuring the validity of product features and descriptions, their alignment with the values of the marketplace rules, governmental laws, and social virtues.

2.1.2 ML terminology

- Machine learning model - a software program capable of making predictions within a specific domain, such as predicting product sales by category in online retail. Example: A machine learning model trained on historical customer purchase data can predict future sales volumes for different product categories based on factors like seasonality, customer demographics, and marketing promotions.
- Dashboard - an interactive website or application designed to display and visualize key properties and insights derived from relevant data. For example, accuracy of ML model by predict sales in stores. Example: An e-commerce analytics dashboard showing metrics such as daily sales trends, top-selling products, customer

demographics, and profitability by product category. Example: In an online retail platform, the confusion matrix helps to visualize how well a sales prediction model identifies successful product category sales (positive) and unsuccessful sales (negative) based on actual vs. predicted outcomes.

- Confusion matrix - the confusion matrix is a table used to describe the performance of a classification model. It contains four entries:
  - TP (True Positive): The number of correctly predicted positive samples.
  - TN (True Negative): The number of correctly predicted negative samples.
  - FP (False Positive): The number of incorrectly predicted positive samples.
  - FN (False Negative): The number of incorrectly predicted negative samples.

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

Table 1: Confusion Matrix

- Precision - the ratio of correctly predicted positive observations to the total predicted positives. It is calculated using the formula: Precision $= \frac{TP}{TP+FP}$. Example: In an e-commerce platform, precision measures how accurately the model predicts successful sales (positive predictions) for specific product categories among all predicted sales.
- Recall - the ratio of correctly predicted positive observations to all observations in the actual class (also known as Sensitivity or True Positive Rate). It is calculated using the formula: Recall $= \frac{TP}{TP+FN}$. Example: In an online retail recommendation system, recall measures how well the model captures all successful sales (true positives) for specific product categories among all actual sales.
- F1-Score - the harmonic mean of Precision and Recall. It is used to evaluate the performance of a classification model, especially when the balance between Precision and Recall is important. The formula for the F1-score is defined as follows: $\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$. Example: In an e-commerce platform, the F1-score helps assess how well the model balances the ability to accurately predict successful product category sales (precision) with capturing all actual successful sales (recall).
- Accuracy - a metric that measures the overall performance of a classification model. It calculates the ratio of correctly predicted observations to the total number of observations. The formula for accuracy is defined as follows: Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$. Example: In an online retail sales forecasting system, accuracy quantifies how well the model correctly predicts both successful (sales) and unsuccessful (non-sales) outcomes for product categories.
- Weighted accuracy - a metric that considers the contribution of each class (category) to the overall accuracy of the model, weighted by the size of each class. The

formula for calculating weighted accuracy is expressed as follows: $\text{Accuracy}_{\text{weighted}} = \frac{\sum_{i=1}^{N} w_i \cdot \text{Accuracy}_i}{\sum_{i=1}^{N} w_i}$ where:

- N - is the total number of classes (categories).
- w_i - is the weight assigned to class i, typically proportional to the class distribution in the dataset.
- Accuracy_i - is the accuracy for class i, calculated as $\frac{TP_i+TN_i}{TP_i+TN_i+FP_i+FN_i}$, where TP_i, TN_i, FP_i, and FN_i are the counts for true positives, true negatives, false positives, and false negatives, respectively, for class i.

Example: In an online retail platform with multiple product categories, weighted accuracy evaluates how accurately the sales prediction model forecasts sales for each category, giving more weight to categories with higher sales volumes or strategic importance.

2.2 Scope of the ML project

2.2.1 Background

Ozon is one of the most famous and largest marketplaces in Russia. Based on their official tutorial [6], the seller must fill in the following entries while registering a new product:

1. Category;
2. Product title;
3. Stock keeping unit;
4. Price, fees, and discounts;
5. Packaging parameters - size, weights, variants of packages (number of items in one package; length of the rope, cable, hose, etc);
6. Photos;
7. Expiration dates;
8. Brand;
9. Color;
10. And few other dozens of product features.

As we could see, there are many entries to be filled in and while some of them are fairly simple to fill in - like packaging parameters, others require in-depth understanding of marketplace tag systems - including categories.

The major problem with categories is caused by the weak borders between the categories and subcategories - the same pair of shoes might be categorized both as a sports shoe and casual shoes. Therefore, we as a marketplace platform may simplify the category selection by suggesting the most appropriate category based on other essential features of the product.

We believe that with the use of the simplified category selection approach sellers are less likely to put their product in an inappropriate category and thus get higher sells. Therefore, our platform fees will also increase.

2.2.2 Business problem

Sellers set up an incorrect product category to result in an overall loss of profit for both the platform and the sellers.

2.2.3 Business objectives

Taking into account the analysis above we may formulate the business objective as follows: increase the loyalty of the sellers by providing them with product category recommendation based on product features, including its description. Therefore, we need to answer the question: "Is it possible to predict the product category with 90% accuracy based on a product description?".

2.2.4 ML objectives

Create a classical ML model or model with neural network that uses NLP to predict the product category based on its textual description, product name, available variations, and price. Ideally, preserve model interpretability. Avoid use of large models to keep deployment, operation, and maintenance cheap.

2.3 Success criteria

2.3.1 Business Success Criteria

- Simplify the product card creation via suggestion of the most likely product category. Representative sellers group can evaluate the final service convenience;
- Reduce new sellers churn by at least 1%.

2.3.2 ML Success Criteria

- Achieve at least (1 / 20) in task accuracy in category prediction on unseen data, where 20 is count of predict classes;
- Use classical statistical ML models to ensure low computational demands;
- Use Neural Network ML models;
- Use only classical NLP techniques - such as TF-IDF and Word2Vec embeddings to keep models simple and intrepretable.

2.3.3 Economic Success Criteria

- Increase in Revenue: Achieve a 5% increase in total platform revenue attributed to enhanced product categorization accuracy and efficiency.
- Cost Savings: Reduce operational costs related to manual categorization efforts by 15% through automation and improved accuracy.

- Seller Retention Improvement: Improve seller retention rates by at least 3% due to better product visibility and placement within appropriate categories.

2.4 Data collection

2.4.1 Data collection report

Kaggle[5] dataset was selected as the data source. The data were provided in a csv file in tabular form. The data contain the following structure: 12 columns, 464433 rows. To encourage reproducibility, we also saved a copy of the data on our Yandex disk.

2.4.2 Data version control report

To ensure the reliability and reproducibility of our machine learning models, we use Data Version Control (DVC). This tool helps us manage large datasets, track data versions, and log changes. Each version of the data, such as the initial raw data, cleaned data, and feature-engineered data, is tagged and documented. Changes to the data are recorded with detailed descriptions.

We back up data using a combination of cloud storage and on-premises servers, ensuring redundancy and accessibility. DVC stores metadata in our Git repository, while actual data is stored remotely. Historical data is retained for audit and compliance purposes. Access to the data is restricted to authorized team members, with roles and permissions managed to ensure data integrity.

The importance of data version control includes:

- Retraining models based on new data or approaches.
- Addressing model degradation over time.
- Quickly rolling back to previous versions if needed.
- Ensuring compliance with corporate or government regulations.

2.5 Data quality verification

2.5.1 Data description

In total, provided sample contains 464433 records distributed along 12 columns:

1. itemid - unique item identifier. It appears to be actually unique;
2. shopid - shop identifier. In total we have information from 7856 shops;
3. item_name - name of the product that is used during its sale. We found 118346 unique item names but chances are that some of them are just different names for the same item;
4. item_description - textual description of item provided by a seller;
5. item_variation - array of possible item variations - different colors, package sizes, shoe size and so on;

6. price - price of the item in USD;
7. stock - how much items are still available;
8. category - to which of 21 unique categories the item belongs;
9. cb_option - binary feature indicating whether the item can be sold cross-border. Only 11% of items can not be sold to other countries;
10. is_preferred - binary feature indicating whether the item is sold by a shop focusing on this category and having an appropriate license or working with a manufacturer directly. Only 4% of the items are sold by preferred shops;
11. sold_count - how many items the shop has sold so far;
12. item_creation_date - the date when the seller has created this item card. It ranges from Jan 1, 2016 to Sept 9, 2017.

2.5.2 Data exploration

First of all, we decided to see how many repeated description do we have in total - and we found out that 75.5% of descriptions are used at least twice. It means that different sellers are likely to use the same descriptions. This claim is strongly supported by the fact 17.6% of the item name - item description pairs are unique. Still, the number of unique descriptions is pretty high and thus we need to apply some NLP pipelines to efficiently utilize that data.

We further discover that on average, the item title contains 60 characters and and is 10 times shorter than the item description (that contains on average 635 characters). Distribution of lengths are presented of Figures 1 and 2. We may notice that the distribution of name lengths rapidly drops after about 70 characters while the distribution of description lengths resembles a truncated normal distribution. From this we may conclude that descriptions are longer than titles but still remain quite shord.

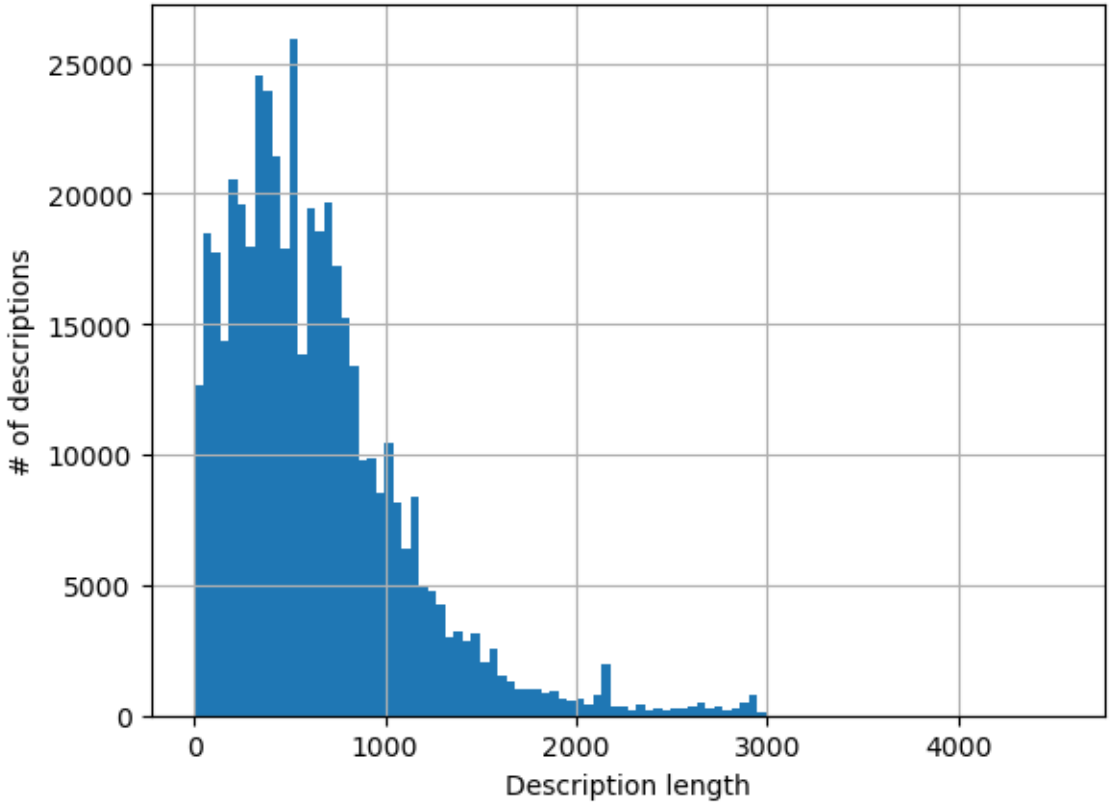

Figure 1: Distribution of description lengths

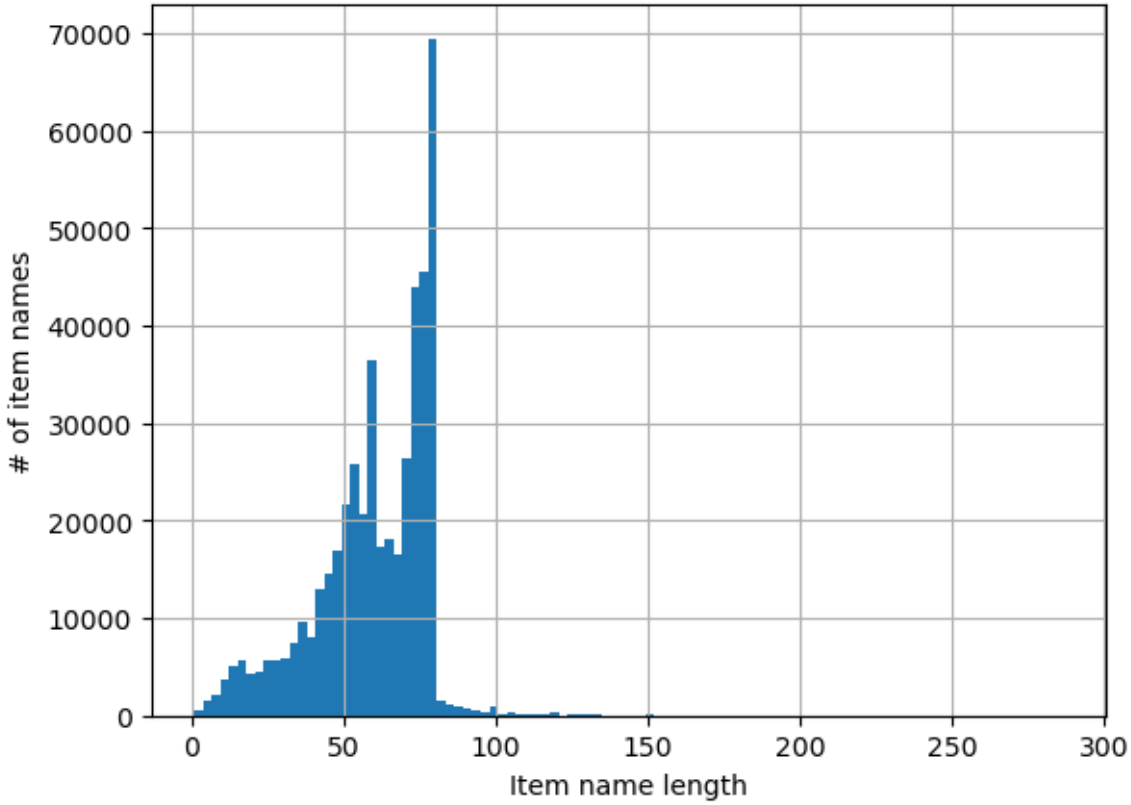

Figure 2: Distribution of description lengths

We also decided to analyze the number of unique descriptions per item. The results are presented on a figure 3. Analyzing the plot we may conclude that majority of items have at most two unique descriptions.

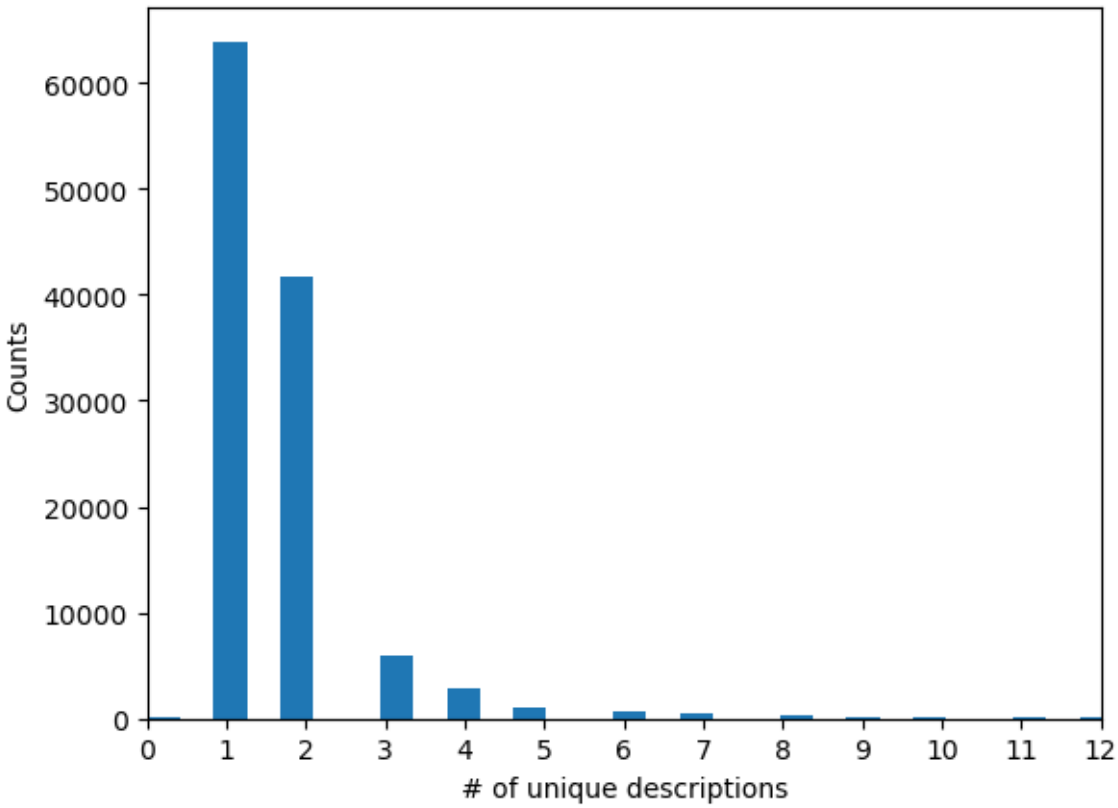

Figure 3: Distribution of the number of unique descriptions for items

2.5.3 Data requirements

The data should be defined in the following order:

1. itemid - unique numeric item identifier between 1 and more;
2. shopid - shop numeric item identifier between 1 and more;
3. item_name - string data type between 5 and more length;
4. item_description - string data type between 10 and more length;
5. item_variation - array of possible item variations between 0 and more length;
6. price - float number between 0 and 9000000;
7. stock - numerical item between 0 and 400000;
8. category - string data type with length between 5 and more;
9. cb_option - binary feature 0 or 1;
10. is_preferred - binary feature 0 or 1;
11. sold_count - integer numeric between 0 and 8007;
12. item_creation_date - date type of data between Jan 1, 2016 and Sept 9, 2017.

2.5.4 Data quality verification report

Analyzing the quality of the data we came up with the following conclusions:

1. Chances are we have only top level categories available on a market place in this sample - usually market place also have a hierarchy of categories. For example, shoes might be subdivided by gender, by expected usage scenario (sports, official events, daily, and so on), season (summer, winter, autumn or spring), age group and in many other ways;
2. There are 1111 (about 2% of the data) rows having missing values represented by NaNs. These missing values appear in one columns - item_name, item_description, and category.

2.6 Project feasibility

2.6.1 Inventory of resources

Our resources for this project:

- Grigoriy Nesterov - Data Engineer;
- Georgy Andryushchenko - ML Engineer;
- Aleksandr Vashchenko - Data Scientist;
- Ozon Sellers educational platform[6];
- Extract of the data from online retail platform[5].

2.6.2 Requirements, assumptions and constraints

The key requirements of the project include:

1. On-time deployment - before 19th of July;
2. High quality of suggestions.

To guarantee on time deployment the time available (8 weeks) must be efficiently distributed. Table 2 contains expected project completion plan.

Week(-s)	What should be accomplished by the end of the week?
1	Business and Data Understanding
2-3	Data engineering/Preparation
3-4	Model engineering
5	Model validation
6	Model deployment and Model monitoring and maintenance

Table 2: Expected project execution plan

Some of the stages have completion time significantly overestimated to give some freedom for creativity and exploration, if necessary. Otherwise this time can be used for quality improvement of our service.

To ensure the high quality of suggestions we have to rely on several assumptions to be true:

1. Provided data sample provides a high quality representation of typical product cards;
2. Available features contain all the information necessary to properly predict the item category;
3. Distribution of categories in the data properly represents that of the real data.

Provided data sample is in public domain and therefore can be freely used without any litigation. Data is anonymized and does not include any personal information so even if the sample leaks nobody is harmed. All of the sudden, the volume of the data might be insufficient to use state-of-the-art natural language processing approaches involving neural networks.

Further, our project team is significantly limited in choice of tools as administration of IU Hadoop cluster is a responsibility of other authority.

2.6.3 Risks and contingencies

We organize major risks and their mitigation strategies in the Table 3.

Data induced risks	
Risk	Mitigation strategy
Non-representative data sample	Intensive work with customer to ensure high-quality of a new data sample
Outdated sample	Obtain a fresh sample or design a thorough model assessment and controlled deployment
Imbalanced data	Use re-sampling techniques
Model induced risks	
Risk	Mitigation strategy
Low quality of classification	Collect more data or introduce new features, including product pictures
Invalid product features or descriptions	Rule-based filters ensuring and enforcing reasonable inputs to the model and publicly tolerated contents of the product card
Change in category system	Design a correspondence between old and new categories. Retrain the model with new categories when sufficient sample can be collected.
Inappropriate descriptions designed to mislead the model, advertise some product, offend other customers, etc.	Rule based product card filtering
Other risks	
Risk	Mitigation strategy
Competitors deploy similar service	Introduce more features into analysis - add product photos into classification pipeline

Table 3: Main project risks and their mitigation strategy

2.6.4 Costs and benefits

Major expense categories include: project development and maintenance; periodic model adjustment for changes in data and/or company or governmental policies; monthly expenses: electricity, internet traffic.

As it is more a proof of concept project rather than a production ready system we expect to spend about 300000 rubles for salaries of two developers. Ideally, the team also needs a security engineer and preferable a software architect and that doubles the expenses. Maintenance of the project can be performed on demand by the forces of on-site teams. If the team spends on average 1 week in a month to maintain the service, it will cost 75000 rubles monthly. Model adjustment is unlikely to happen often but if we assume it happen in each quarter, it will cost about 300000 rubles on development itself and 50000 extra to organize extraction and anonimization of the data. Additional traffic and computational demands are not that high unless we use a modern NLP approaches thus we ignore them in our calculations.

To sum up, the project estimated cost in nearest two months is 300000 rubles.

It's annual maintenance will cost about 2.3 million rubles. Notice that the current estimates do not include the expenses on the internet traffic and hardware platforms into calculation as the service that is to be developed is not computationally demanding and can be deployed on a currently available hardware.

In 2023 the growth rate of the number of sellers has significantly dropped because according to the reports about 53% of sellers quit marketplaces after the first year of their business. Among the main reasons 27% of sellers mention low sales due to inefficiency of their product cards and significant time needed to fill in the card. Therefore, we believe that simplification of the product card creation process might significantly reduce the rate of sellers churn.

Let us estimate the annual profit from this system: in 2023 number of sellers on Ozon platform increased by 21% and reached 450000. About 18000 sellers are likely to leave the platform by the end of the year. If the service will manage to keep just 10% of those sellers on a marketplace, it will generate 4 million rubles of sales given the average order price in 2023 of 2188 rubles. Platform takes 10% as a fees so it will produce 400 thousands rubles extra. In such a simple model we expect the project to be loss-making. However, we understand that there are many effects we could not directly take into account or estimate: if 0,1% of sellers will make not 1 but 2 sales in a year it will generate 9.9 millions of income extra thus making the project profitable; our current sellers are likely to get more committed to our marketplace as they can clearly see that we also take their needs into account and further they might invite their friends to start their business on our platform or switch from another marketplace; our customers are more likely to find product they are looking for in a right category thus returning more frequently to our marketplace.

Overall, we can see that the project has a moderate risk. In the worst case we will lose about 10 million rubles in 3 years which negligibly small with respect to 10 billions of income of Ozon in 2023. If, however, the project deploys successfully in the most sceptical scenario we expect the income to increase by 0.1% annually. If some of the aforementioned factors will start to work we might expect this ratio to increase at least 10 times.

2.7 Project plan

The first stage of the project takes one week and includes data quality assessment, data cleaning and imputation if needed.

The second stage takes two weeks and includes basic data understanding, its limitations, engineering of new features. The largest risk on this phase is non-representative data sample. All of the sudden in this case we can not take any actions to mitigate that risk, and the only action we may take if it becomes true is to help the customer to organize the aggregation of the appropriate data sample. This risk also includes an outdated data which is different in properties from a current data. If we can not obtain a fresh or high quality dataset then the designed model might fail in production system, so we will have either organize A/B testing and see how model performs in the real environment, or invite a group of sellers to test new system.

During the third stage that lasts for 3 weeks we will try different models and

assess their quality. The models also include different pre-processing steps like TF-IDF or Word2Vec embeddings. To ensure the interpretability we may use neural network and classical statistical models. We might need to introduce new features and thus roll back to one of previous stages.

Finally, in the last week of the project we aim to organize the project deliverables in a format ready for deployment and presentation to management of the business. It might happen that other marketplace delivers similar product and we will have to adjust model to ensure its higher quality and convenience for the end user.

3. Data Preparation

3.1 Select data

With the results of Business understanding and exploratory data analysis we decide:

- Keep itemid - this column contains only unique values and therefore does not provide any valuable information for a predictive model;
- Keep shopid - since shops usually tend to focus on a few closely related categories, the shopid might be useful for a prediction;
- item_name, item_description - after the embedding name and description might become the most important features - as people can understand the category of object based on its name and description, the model with a high quality embeddings will be likely able to choose the appropriate category;
- item_variation - variations might be helpful to guide the category - for example if available variations include S, M, L, XL - the item is likely an apparel, or if it is 35, 39, or 42 it is likely a shoe;
- price - some categories are naturally more expensive than the others, so based on price the models might learn to exclude certain categories;
- stock - might be useful as for some luxury products the stock is likely to be small while for a daily usage products it is likely to be high;
- category - the target variable that we will have to encode;
- cb_option - for example, some countries might limit import or export of certain categories of products;
- is_preferred - in combination with shopid this might significantly improve the prediction as for example if the model learns that shop specializes on selling foods and it is preffered for this item, then the item is likely to be a food;
- sold_count - the reasoning is similar to stock features;
- item_creation_date - some categories have a seasonal trend. Usually, at the end of the summer cloth shops will add new clothes for autumn season.

3.2 Clean data

Based on an Exploratory Data analysis, be believe that we can safely drop 1111 rows containing at least one missing value. even after that step we will still have 463 thousand transactions for other manipulations, including feature enginiring and predictive analysis. This number of records is still sufficient for the basic machine learning and NLP algorithms.

3.3 Construct data

We split the date into components: year, month, day.

We also apply tokenization and Word2Vec to the item name, item description, and existing variations of an item.

3.4 Standardize data

Standardization involves several key processes, which we outline below:

We normalize all features to ensure they are on a similar scale. We use Standard scaling to transform these features to 0 variance and 1 mean.

For text features such as item_name, item_description, and item_variation, we apply embeddings. This converts the text into numerical vectors that capture semantic relationships between words, making them usable for machine learning models.

The target variable, category, is encoded using label encoding, where each category is assigned a unique integer. This encoding is suitable for classification tasks and allows the model to learn the relationship between features and the target categories.

All data containers, after individual preprocessing steps, are merged into a single, unified dataset. This involves concatenating the normalized numerical features, encoded categorical variables, date features, and text embeddings into one comprehensive dataframe.

The final dataset is organized into a uniform schema. Each feature is clearly labeled, and data types are standardized (e.g., all numerical features are floats, categorical features are integers). This ensures that the dataset is ready for the modeling stage without further modifications.

By standardizing the data, we ensure that it is clean, consistent, and ready for model training, thereby improving the accuracy and efficiency of our machine learning models.

4. Model Engineering

4.1 Literature research on similar problems

Research 1: Applying Machine Learning to Retail Demand Forecasting [4]

- Description: A study by Nasseri et al. (2023) applies tree-based ensemble methods and LSTM to retail demand forecasting, demonstrating improved forecasting accuracy.
- Results: The work shows that ensemble-based methods and LSTM can significantly improve the accuracy of demand forecasting compared to traditional statistical methods.

Research 2: Applying machine learning in retail for personalization and logistics improvement [3]

- Description: An article on the iTransition website describes various applications of machine learning in retail, including market analysis, personalizing purchases, and optimizing logistics processes.
- Results: The application of machine learning enables retailers to improve operational efficiency, enhance customer interactions, and optimize inventory management.

Research 3: Text classification for predicting multiple categories in the retail sector [1]

- Description: Research on arXiv uses LSTM and BERT models for text classification and product category prediction in retail.
- Results: The application of advanced machine learning models achieves high classification accuracy, even when dealing with unbalanced classes.

These researches show that the use of advanced machine learning techniques such as ensemble decision trees, LSTM, and BERT can significantly improve the accuracy of demand forecasting, enhance personalization of purchases, and optimize logistics processes and product classification. By applying these techniques, operations can be made more efficient, customer interactions can be improved, and inventory management can be optimized, leading to increased customer satisfaction and business profitability.

4.2 Define quality measures of the model

It was decided to use the following metrics to determine the quality of the model: accuracy, f1-score.

4.3 Model Selection

It was decided to use logistic regression as a baseline and neural networks as models to achieve the objective, multiple linear layers were chosen as the structure.

A general list of all models in use:

1. LogisticRegression
2. DecisionTree
3. Custom NN model with Linear Layers with following architecture:
    - Linear layer (310 to hidden_dim).
    - Leaky relu.
    - Some number of linear layer (hidden_dim to hidden_dim) with leaky relu.
    - Linear layer (hidden_dim to 20).
4. Custom NN model with Transformer with following architecture:
    - Torch transformer encoder layer (dim=310, dropout=0.1).
    - Linear layer (310 to 20).

The input is a data vector with dimensionality 310, which corresponds to the dimensionality of the text features transformed with TF-IDF with added numeric features.

The output is a vector of data with dimensionality 20, which corresponds to the number of classes in the current classification task.

4.4 Incorporate domain knowledge

In examining the data and the domain, it became clear that it is worth analyzing the product description first to determine the product class. Thus, it was decided to convert the description into a numerical vector representation and use it for product class prediction.

4.5 Model training

To train and evaluate our models, we created two distinct test datasets, each constituting 20% of the total dataset. By creating two different test datasets, we aimed to validate the consistency and robustness of our models across varied data subsets. This strategy enhances the reliability of our model performance metrics and helps in identifying any potential overfitting or underfitting issues.

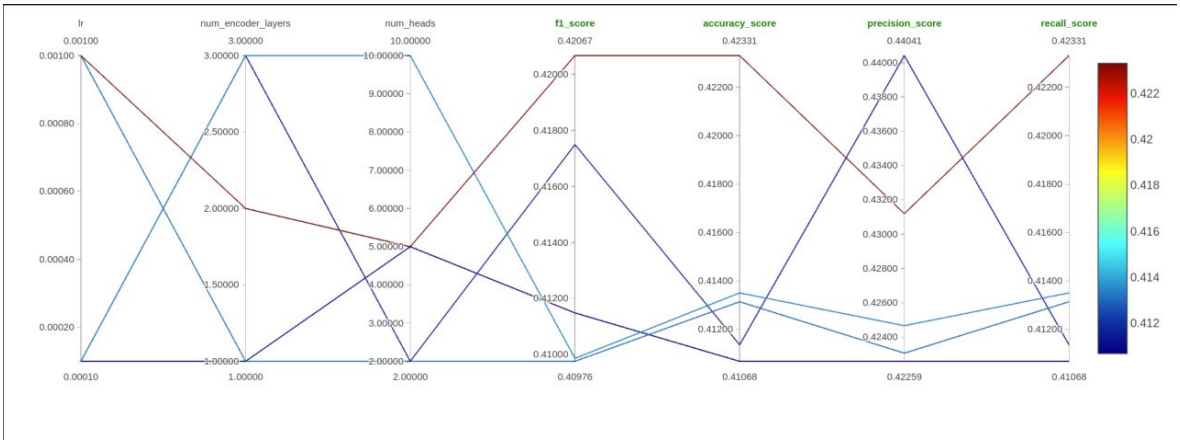

Figure 4: Transformer Model Results

The transformer model shows a variation in performance across different hyperparameters. The model's performance metrics, such as f1_score, accuracy_score, precision_score, and recall_score, vary significantly depending on the learning rate (lr), the number of encoder layers, and the number of heads. The best model configuration appears at a specific point with the highest f1_score of approximately 0.42067. Vulnerabilities: The performance of the model is sensitive to the learning rate, with small changes leading to significant variations in metrics. The model's complexity increases with the number of encoder layers and heads, potentially leading to overfitting if not properly regularized.

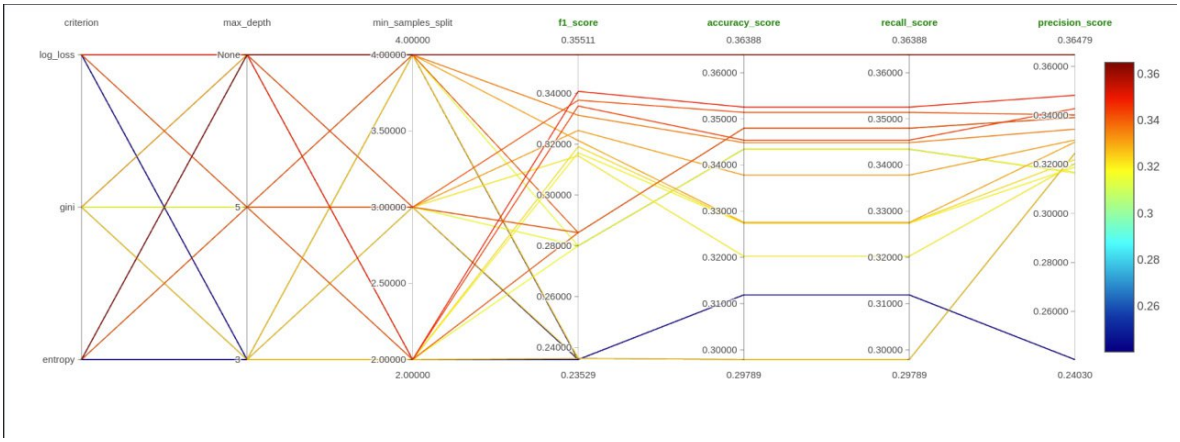

Figure 5: Decision Tree Model Results

The decision tree model's performance is impacted by the criterion, max_depth, and min_samples_split. The f1_score peaks at 0.35511, and accuracy_score is about 0.36388. The model is straightforward but can suffer from high variance. Vulnerabilities: Overfitting is a concern, especially with deeper trees. Sensitivity to changes in max_depth and min_samples_split, which can lead to instability in performance.

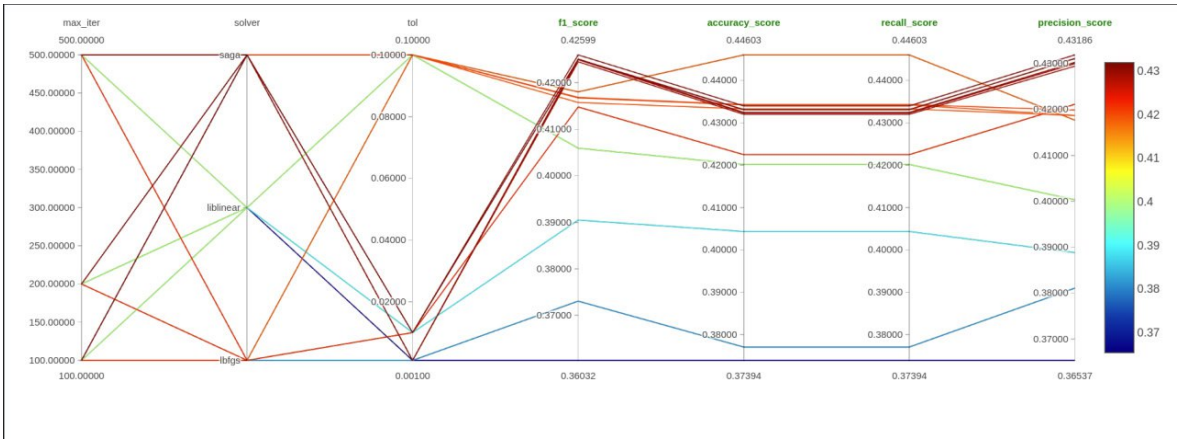

Figure 6: Logistic Regression Model Results

The logistic regression model's performance varies based on max_iter, solver, and tol. The best f1_score observed is 0.42599, with an accuracy_score of 0.44603. Vulnerabilities: Choice of solver can affect convergence and performance. Requires careful tuning of tolerance to balance between precision and computational efficiency.

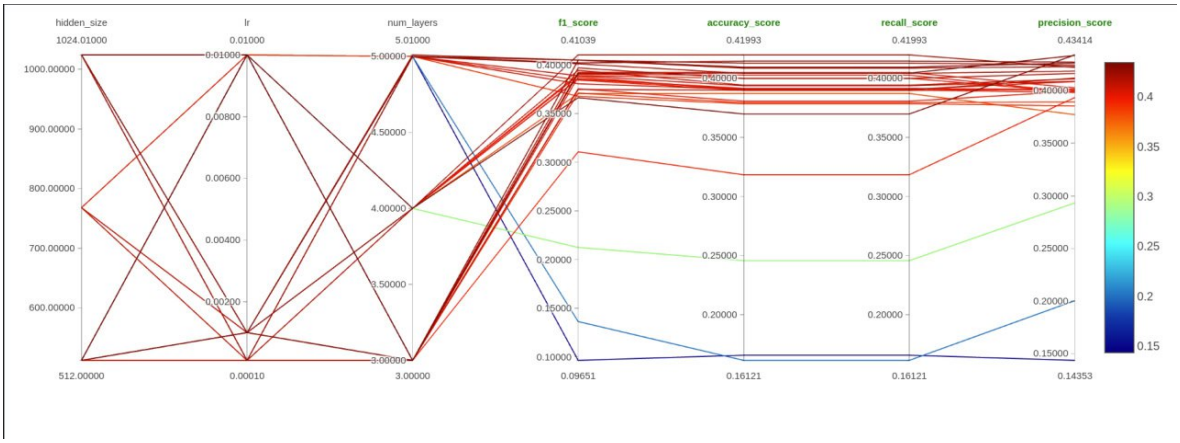

Figure 7: Linear Layers Model Results

The linear layers model shows that hidden_size, lr, and num_layers significantly affect the performance metrics. The highest f1_score is around 0.41039. Vulnerabilities: Overfitting with larger hidden_size and more layers. Sensitivity to learning rate adjustments.

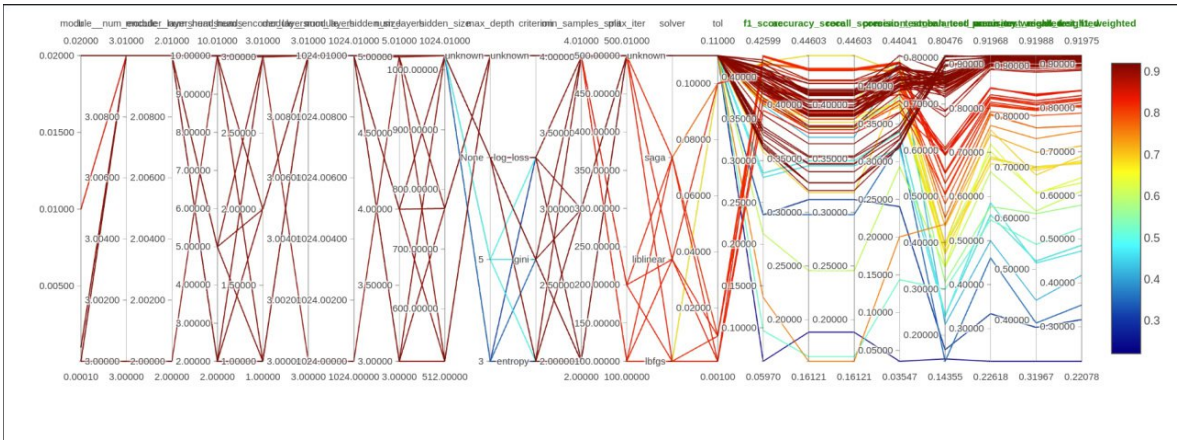

Figure 8: All Runs Comparison

The comparison of all runs shows the relative performance of each model type. The best performing models are highlighted with the highest scores for f1_score, accuracy_score, recall_score, and precision_score. The logistic regression and transformer models appear to perform better compared to others.

4.6 Assure reproducibility

To achieve our objectives, we employed cross-validation to identify the optimal model. The following models and their respective parameters were evaluated using a 3-fold cross-validation approach:

1. LogisticRegression:
   - Solvers: ["lbfgs", "liblinear", "saga"]
   - Tolerance: [0.1, 0.01, 0.001]
   - Maximum Iterations: [100, 200, 500]
2. DecisionTree:
   - Criteria: ["gini", "entropy", "log_loss"]
   - Maximum Depth: [None, 3, 5]
   - Minimum Samples Split: [2, 3, 4]
3. Custom NN model with Linear Layers with following architecture:
   - Hidden Layer Sizes: [512, 768, 1024]
   - Number of Layers: [3, 4, 5]
   - Learning Rates: [0.01, 0.001, 0.0001]
4. Custom NN model with Transformer with following architecture:
   - Learning Rates: [0.01, 0.001, 0.0001]
   - num_encoder_layers: [1, 2, 3]

- num_heads: [2, 5, 10]

For the implementation, the following services were utilized:

- ZenML: For managing and automating the machine learning workflows.
- Hydra: For managing configuration and parameter tuning.
- Airflow: For orchestrating and scheduling workflows.
- MLflow: For tracking experiments, managing models, and logging metrics.

Consider the tables: 4, 5, 6 and 7 to reproduce the result.

Pseudocode to implement an algorithm for training different models with different hyperparameters:

Data: cfg, estimator, X_train, y_train
Result: gs (trained GridSearchCV object)
Input : Configuration (cfg), estimator, training data (X_train, y_train)
Output: GridSearchCV object (gs)
Initialize StratifiedKFold with n_splits = cfg.model.folds, random_state = cfg.random_state, and shuffle = True;
Create param_grid as a dictionary from params;
Create scoring as a list from cfg.model.metrics.values();
Set evaluation_metric to cfg.model.evaluation_metric;

GridSearchCV

Initialize gs with:
- estimator = estimator
- param_grid = param_grid
- scoring = scoring
- n_jobs = cfg.cv_n_jobs
- refit = evaluation_metric
- cv = cv
- verbose = 1
- return_train_score = True

Fit gs with X_train and y_train;

Algorithm 1: Grid Search with Cross-Validation

5. Model evaluation

5.1 Model validation report

Comparing the hyperparameters, the following Challengers were obtained for the models and hyperparameters considered.

params / run	v148	v149	v157	v158	v166
max_iter	100	100	200	200	500
solver	saga	saga	saga	saga	saga
tol	0.01	0.001	0.01	0.001	0.01
Accuracy	0.434	0.432	0.433	0.432	0.433
F1_score	0.426	0.425	0.425	0.425	0.425

Table 4: LogisticRegression Challengers (first model)

params / run	v133	v134	v135	v142	v144
criterion	entropy	entropy	entropy	log_loss	log_loss
max_depth	None	None	None	None	None
max_samples_split	2	3	4	2	4
Accuracy	0.345	0.351	0.364	0.353	0.345
F1_score	0.335	0.337	0.355	0.341	0.331

Table 5: DecisionTree Challengers (second model)

params / run	v121	v133	v137	v138	v139
hidden_size_dim	1024	512	1024	1024	1024
num_layers	3	5	3	4	5
learning rate	0.001	0.001	0.001	0.001	0.001
Accuracy	0.409	0.41	0.413	0.42	0.415
F1_score	0.405	0.405	0.405	0.41	0.402

Table 6: Linear Layers Challengers (third model)

params / run	v11	v15	v21	v26	v28
learning rate	0.001	0.001	0.0001	0.0001	0.0001
num_encoder_layers	1	2	1	3	3
num_heads	2	5	5	2	10
Accuracy	0.413	0.423	0.411	0.411	0.414
F1_score	0.41	0.421	0.411	0.417	0.41

Table 7: Transformers Challengers (fourth model)

Validation with Giskard was performed using the champion model, which is one of the models at the heart of the transformer architecture.

Based on the Giskard report, the overall performance of the model is strong, yet certain data groups exhibited below-average results. This discrepancy attributed to an insufficient number of examples from these groups during training, incorrect labels in the training dataset, or distributional shifts between the training and testing datasets.

The report highlighted two significant ethical concerns related to model bias with switching gender. Addressing these biases is essential to ensure fair and unbiased predictions. This can be achieved by thoroughly analyzing the root causes of these biases and developing strategies to mitigate them. Paying attention to ethical considerations will not only improve the model's accuracy but also enhance its reliability and user trust.

To enhance the model's performance, it is recommended to increase the number of examples in the underperforming data groups and perform a thorough review of the labels in the training dataset. Adhering to the bias mitigation guidelines provided in the Giskard report will help in understanding the underlying causes and offering effective solutions to improve the model.

5.2 Discussion

Upon analyzing the Giskard report and comparing it with the ML modeling results, several key observations can be made. The Giskard report provides a detailed assessment of the model's performance, highlighting specific areas of concern such as biases and the performance of different data groups. In contrast, the ML modeling results focus on the overall performance metrics of the model across various runs and configurations.

5.2.1 Giskard Report Insights

The Giskard report points out that certain data groups exhibit below-average performance. This discrepancy could be due to a lack of sufficient examples from these groups during training, incorrect labels, or distributional shifts between the training and testing datasets.

The report identifies ethical concerns related to model bias. Addressing these biases is crucial for ensuring fair and unbiased predictions. The report suggests strategies for bias mitigation, such as increasing the number of examples in underperforming groups and thoroughly reviewing the training labels.

5.2.2 ML Modeling Results

The ML modeling results show that the model generally performs well across different configurations and runs. The charts provided in the ML modeling results illustrate the validation metrics for various models, such as Transformer, Decision Tree, Logistic Regression, and Linear Layers.

The overall metrics, such as accuracy, precision, recall, and F1-score, are consistent and meet the expected benchmarks for a deployable model. The comparison of all runs indicates that the chosen model configuration (MLP Champion) performs optimally.

The results from different runs and models demonstrate the robustness of the selected model. It indicates that the model has been fine-tuned and performs reliably across different scenarios.

5.3 Deployment decision

Based on the comparison of the Giskard report and the ML modeling results, the decision to deploy the model can be informed by the following considerations:

While the model shows strong overall performance, the biases and ethical concerns highlighted in the Giskard report need to be addressed. Implementing the recommended bias mitigation strategies will ensure that the model provides fair and unbiased predictions.

The consistent and robust performance metrics observed in the ML modeling results support the deployability of the model. However, it is crucial to ensure that the performance improvements and bias mitigation steps are applied before deployment.

6. Model Deployment

6.1 Define Inference Hardware

For the deployment of the ML model, a laptop equipped with an NVIDIA GeForce RTX 4070 GPU is utilized. This hardware provides substantial computational power and is well-suited for handling the inference workload of the model, ensuring fast and efficient predictions.

6.2 Deployment Resources

- Hardware (Laptop):
  - NVIDIA GeForce RTX 4070 GPU mobile (8 GB)
  - Intel Core i7 13700HX CPU
  - 32 GB RAM (DDR5)
- Software:
  - Docker: Containerization platform for creating, deploying, and running applications.
  - Flask: Lightweight WSGI web application framework for Python, used to create a simple API endpoint.

6.3 Model Evaluation under Production Conditions

Business and Economic Success Criteria

To ensure that the model meets business and economic success criteria, the following benchmarks have been established:

- Accuracy Threshold: The model should maintain high accuracy on real-world data.
- Latency Requirement: The model should return predictions quickly enough to be usable in a practical setting.
- Resource Utilization: The model should operate efficiently, without overloading the available hardware resources, ensuring scalability and stability.

Testing the Model on Different Data Samples

- Data Sampling: A diverse set of data samples, including edge cases and typical cases, was selected to evaluate the model's robustness.
- Performance Metrics: Key metrics such as accuracy, inference time, and resource utilization were monitored.
- Testing Results:
  - Accuracy: The model achieved satisfactory accuracy on the test dataset.

- Inference Time: The model's inference time met the practical requirements for speed.
- GPU Utilization: GPU utilization remained within acceptable limits, demonstrating efficient resource use.

6.4 Deployment Strategy

The deployment strategy included the following actions:

6.4.1 Docker Containerization

A Dockerfile was created to encapsulate the environment, ensuring consistency across different deployment stages. This file contains all necessary dependencies and configurations for the model to run seamlessly within a container.

6.4.2 Flask API

A Flask application was developed to serve the model predictions via an HTTP API endpoint. This setup allows external applications and services to interact with the model by sending HTTP requests and receiving predictions as responses.

6.4.3 Gradio UI

A Gradio interface was implemented to provide a user-friendly web-based UI for interacting with the model. Gradio simplifies the process of creating interactive interfaces, allowing users to upload data, visualize predictions, and adjust input parameters in real-time. This addition enhances accessibility and usability, making the model more approachable for non-technical stakeholders.

6.4.4 Docker Deployment

The Docker image containing the model and all necessary dependencies was built, and the container was run to deploy the model as a service. This approach ensures that the model runs consistently across different environments and simplifies the deployment process.

6.4.5 Testing

Tools such as 'curl' and Postman were used to send HTTP requests to the Flask API endpoint and verify the responses. This step confirmed the successful deployment of the model and ensured that it was providing accurate and timely predictions.

6.5 Conclusion

The deployment of the ML model using Docker and Flask was successful. The model met all predefined business and economic criteria, demonstrating satisfactory performance in terms of accuracy, inference time, and resource utilization. The deployment is now fully operational, providing a robust solution for practical application in the designated field.

References

[1] Anonymous Authors. Text classification to predict multiple categories in the retail sector using lstm and bert models. arXiv preprint arXiv:2403.01638, 2023.

[2] Tinkoff eCommerce и Data Insight. Рынок нишевых маркетплейсов в России, 2023. URL https://www.tadviser.ru/images/9/99/Tinkoff-ecommerce-and-data-insight-research-niche-marketplace-market-in-russia.pdf.

[3] iTransition Team. Machine learning in retail: 10 ways to upgrade your store, 2023. URL https://www.itransition.com/machine-learning/retail.

[4] H. Nasseri et al. Applying machine learning in retail demand prediction—a comparison of tree-based ensembles and long short-term memory-based deep learning. Applied Sciences, 13(19):11112, 2023.

[5] Nirmal NK. Online retail dataset, 2021. URL https://www.kaggle.com/datasets/nirmalnk/online-retail-dataset. Accessed: 2024/06/18.

[6] Ozon Sellers. Ozon sellers. create commodity card, 2023. URL https://seller-edu.ozon.ru/work-with-goods/zagruzka-tovarov/creating-goods/sozdanie-tovarov-v-lk.