DATA-DRIVEN OPTIMIZATION OF PRICING STRATEGIES ON MICROSERVICE PLATFORMS: INSIGHTS FROM KHAMSAT

Ismael Mousa
Computer Science
An-Najah National University
s12115713@stu.najah.edu

ABSTRACT

This paper presents the Khamsat Predictor, a machine learning model developed to tackle pricing challenges in the freelance marketplace. Leveraging data from Khamsat [1], the largest Arabic freelance platform, the model aims to provide price predictions based on service characteristics. The primary objective is to enhance transparency and trust, alleviating price-related uncertainty for both sellers and clients. The model faces challenges such as small dataset size and class imbalance, which impact its performance. Experimental results, derived from both imbalanced and balanced datasets, highlight the influence of these challenges on model accuracy. Despite these obstacles, the approach demonstrates the potential for delivering useful price estimates. The proposed methodology integrates classical machine learning techniques, from data collection to model deployment, offering a foundation for further improvements and real-world application scalability.

1 Introduction

Khamsat is a leading Arabic platform for microservices, where individuals can buy and sell services starting at $5. It acts as a secure intermediary between freelancers and clients, offering a wide range of services such as content writing, graphic design, digital marketing, and web development. Khamsat is highly accessible, particularly for startups and individuals with limited budgets, while also providing freelancers with an opportunity to build portfolios and gain experience. The platform ensures secure transactions by holding funds until services are delivered and accepted, and allows buyers to rate services, helping guide future users. However, freelancers face intense competition due to the low starting price, which can affect the earning potential of highly skilled individuals. Overall, Khamsat is an excellent choice for cost-effective professional services and serves as a stepping stone for freelancers in the Arabic-speaking world.

One of the main obstacles faced by Khamsat users, whether freelancers providing services or customers seeking them, is setting the right price. Freelancers aim to maximize their profits while remaining competitive, whereas customers strive to minimize costs while ensuring they receive high-quality services. This dynamic often creates a pricing dilemma, as both parties must balance cost and value. In this research, we analyzed an acceptable number of the available freelancer offerings on the platform and applied machine learning techniques to address this challenge. Our goal is to assist freelancers in determining optimal pricing strategies based

on market trends, service demand, and quality metrics. Similarly, we aim to help customers estimate fair costs for services, enabling them to make more informed decisions.

2 Dataset

The dataset used in this study consists of 7,818 entries from 11 service categories on the Khamsat platform see Figure 1, with 26 features encompassing both numerical and categorical attributes.

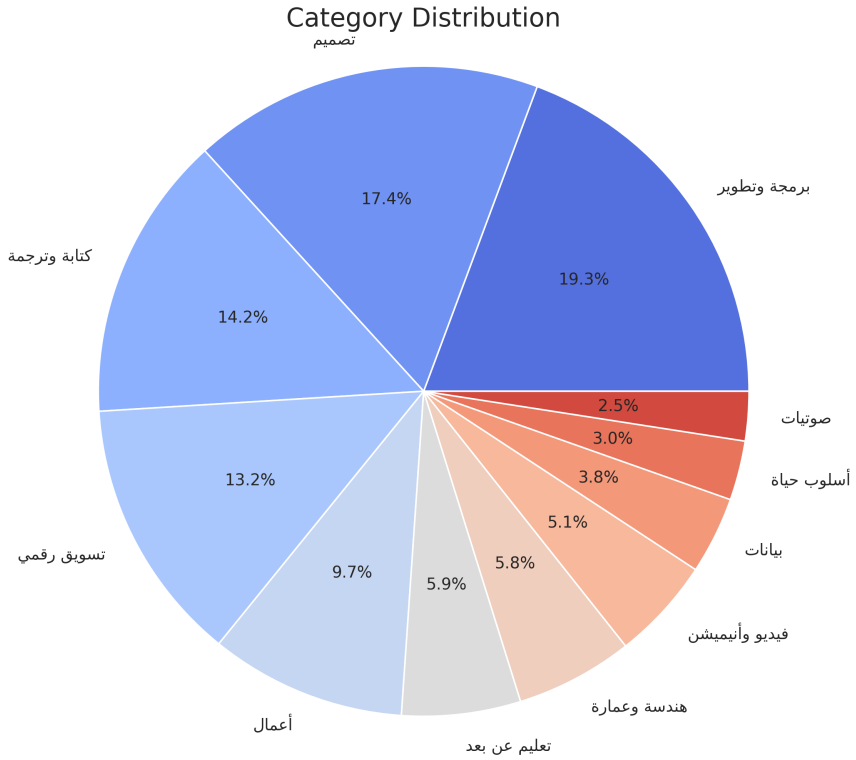

Figure 1: Category Values Distribution

2.1 Overview

The dataset is exclusively in Arabic and is free from missing or duplicated entries in the conventional sense. However, approximately 15.41% of the data contains placeholder values that were addressed during preprocessing. Key features include service details (e.g., Service Name, Additions Price) and seller characteristics (e.g., Owner Level, Owner Services).

2.2 Characteristics

The dataset is primarily composed of Arabic-language data, encompassing 7,818 rows and 26 columns. It includes 11 numerical, 14 categorical, and 1 boolean data type. While there are no explicitly missing values, placeholder values such as "لم يحسب" are present, indicating potential gaps. The dataset contains no duplicated entries. However, it exhibits bias, particularly in the overrepresentation of low-price services and specific seller levels. An initial exploratory analysis highlighted various challenges, including biases in feature distributions and inconsistencies in data formatting. For instance, some numerical fields, like prices and response times, were stored as text, requiring conversion before analysis. Figure 2 illustrates the distribution of numeric features through a histogram.

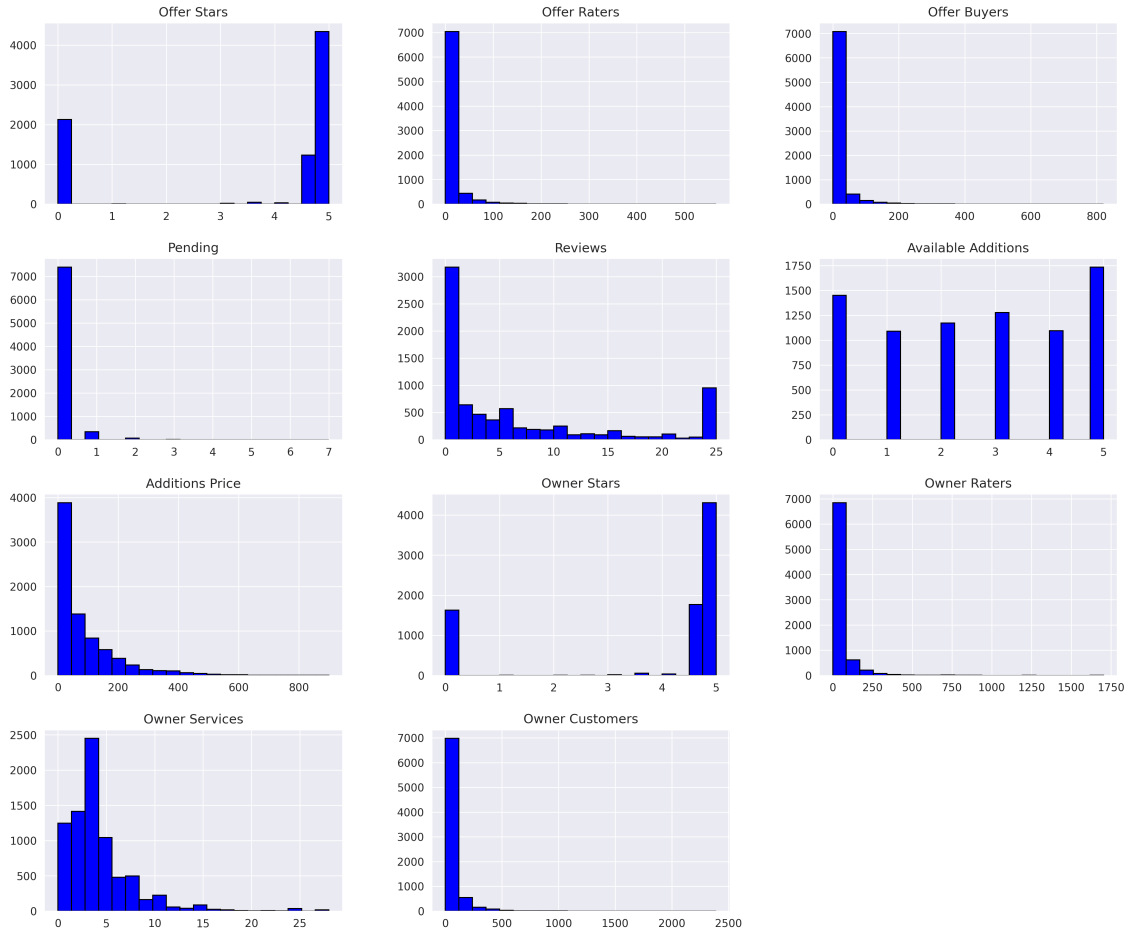

Figure 2: Data Histogram Numeric Features Only

2.3 Features

The dataset features can be grouped into two main categories: offer details and owner details. Offer details include fields such as Category Name*, Service Name*, Offer Stars**, Offer Raters**, Offer Response Time**, Offer Buyers**, Pending**, Price**, Duration**, Reviews**, Available Additions**, and Additions Price**.

On the other hand, owner details encompass fields like Owner Level*, Owner Verified***, Owner Stars**, Owner Raters**, Owner Completion Rate**, Owner Response Time**, Owner Services**, and Owner Customers**.

Additionally, certain features were excluded from the dataset, including Category URL*, Service URL*, Offer Name*, Offer URL*, Owner Name*, and Owner URL*.

Categorical* Numerical** Boolean***

2.4 Collection

The dataset was collected using a web scraping approach with Python's Selenium library [14]. The script extracted offer and owner details from the Khamsat platform by mapping categories to URLs, navigating service pages with Selenium's Chrome WebDriver, and gathering data while avoiding duplication.

3 Methodology

The implementation follows a structured workflow designed to maximize the effectiveness of the machine learning model by focusing on an in-depth understanding of the dataset.

3.1 Exploratory Data Analysis

The dataset underwent several weeks of exploration to understand its structure and the relationships between features, see Figure 3. This process led to the identification of the most impactful features for model training, and decisions were made regarding how to handle various data types. Feature selection and encoding techniques were designed accordingly.

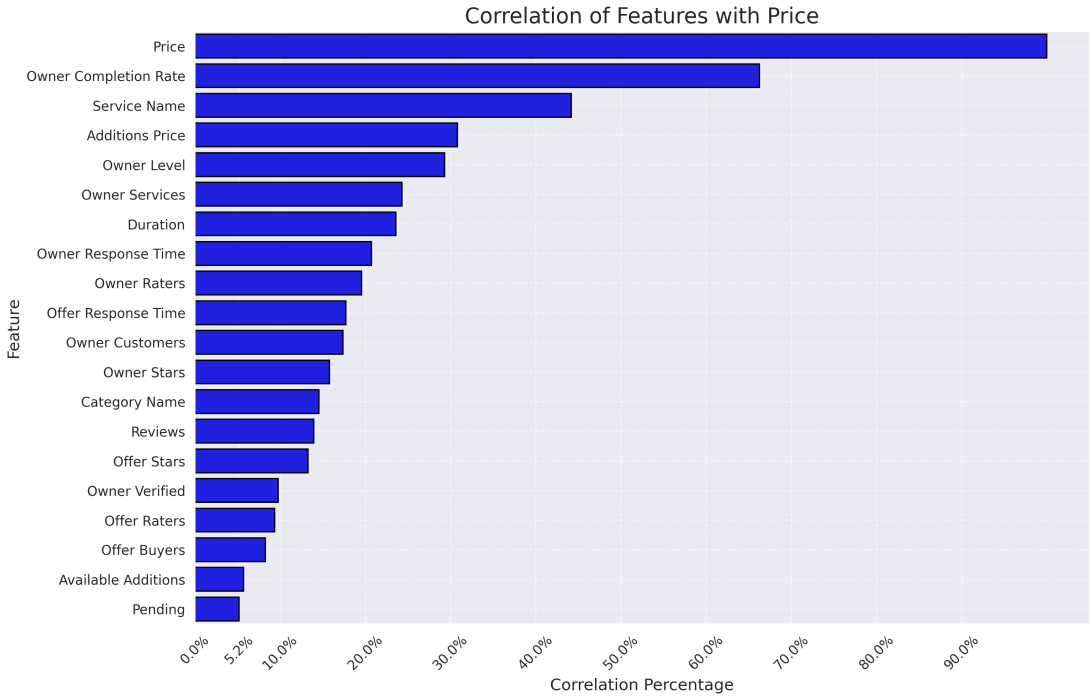

Figure 3: The Linear Correlation Between The Features and Price

3.2 Preprocessing

The preprocessing stage involved several key steps to prepare the dataset. For encoding, categorical values in the Duration feature were mapped to integers representing the corresponding number of days, with mappings stored in JSON files for modularity. Response times, such as Offer Response Time and Owner Response Time, were standardized from text-based values (e.g., "minutes" to "days") into hours using an automated utility, rounding values to two decimal points and storing mappings in JSON files. The Owner Level feature, recognized for its significant influence on pricing, as shown in Figure 4, was manually encoded with values ranging from 5 for beginners to 50 for advanced levels, outperforming traditional encoding methods.

Additionally, a customized one-hot encoding method was applied to features like Category Name and Service Name to ensure deployment compatibility. Placeholder values were handled using the K-Nearest Neighbors (KNN) [2] algorithm for imputation, as alternatives like median imputation or row removal negatively impacted model performance or dataset balance. Random Oversampling was applied to address class imbalance in the Price feature, resampling categories to match the highest frequency count ($5 price), as illustrated in Figure 5.

Figure 4: Price Distribution According to Owner Level

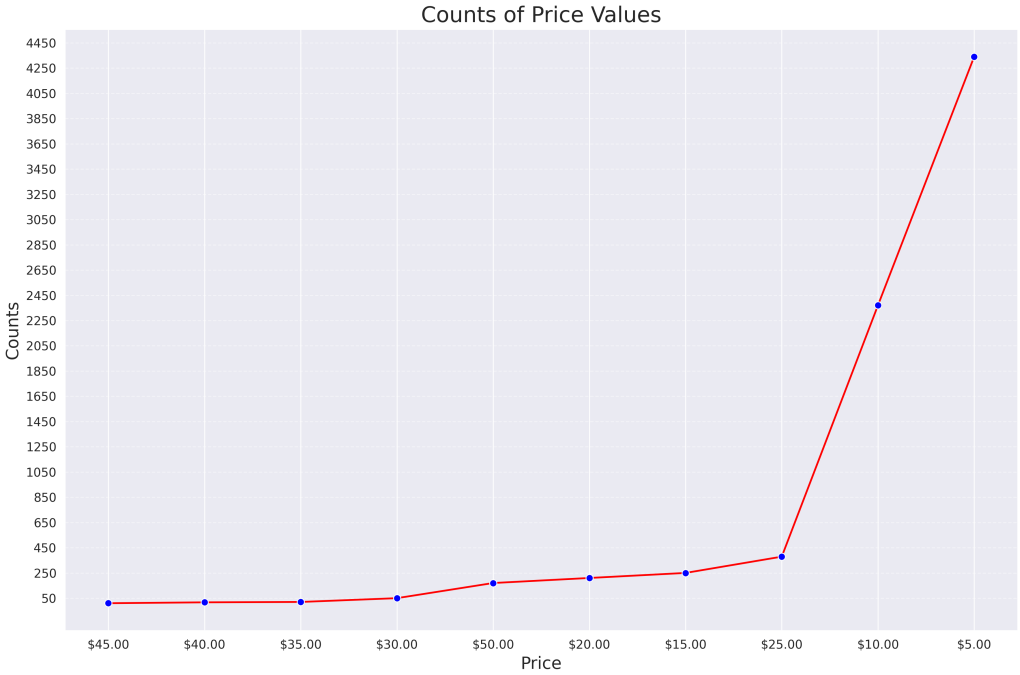

Figure 5: Price Values Distribution

Lastly, the dataset was split into training, validation, and test sets, with stratification ensuring balanced class distribution. 10% of the data was reserved for testing, and 10% of the training data was further split for validation, supporting hyperparameter tuning.

3.3 Modeling

In this study, a multimodal approach was adopted to evaluate and optimize several machine learning algorithms through hyperparameter tuning. Experiments were conducted on both the original imbalanced dataset and a balanced version created via random oversampling, with all workflows tracked using MLflow [11] for reproducibility. Key algorithms included SoftMax Regression, Support Vector Machines (SVC), and Random Forests [3, 4, 5], implemented using Scikit-learn [6]. Hyperparameter optimization was performed with Optuna [7], running up to 50 trials per model to maximize the F1-score [8]. Tables 1 and 2 summarize the tuned hyperparameters for each model on the imbalanced and balanced datasets, respectively.

Table 1: Hyperparameters tuned on the imbalanced dataset.

Hyperparameter	**SoftMax Regression**	**SVC**	**Random Forest Classifier**
Class Weight	balanced	balanced	balanced
Multi Class	multinomial	-	-
Solver	lbfgs	-	-
Kernel	-	linear	-
Gamma	-	0.48	-
C	0.53	0.79	-
Max Iterations	6972	8170	-
Decision Function	-	ovo	-
Estimators	-	-	395
Depth	-	-	14
Criterion	-	-	log loss

Table 2: Hyperparameters tuned on the balanced dataset.

Hyperparameter	**SoftMax Regression**	**SVC**	**Random Forest Classifier**
Class Weight	balanced	balanced	balanced
Multi Class	multinomial	-	-
Solver	lbfgs	-	-
Kernel	-	rbf	-
Gamma	-	0.33	-
C	0.88	0.57	-
Max Iterations	7356	9189	-
Decision Function	-	ovo	-
Estimators	-	-	133
Depth	-	-	22
Criterion	-	-	log loss

Model performance was evaluated using standard metrics, including Log Loss, Accuracy, Precision, Recall, and F1-score [9, 10, 8], providing a comprehensive assessment of generalization capabilities. Results are shown in Tables 3 and 4, highlighting the trade-offs between metrics on imbalanced and balanced datasets. SoftMax Regression achieved moderate performance, while Random Forests showed better accuracy and F1-scores. SVC, particularly on the balanced dataset, achieved the highest performance with 98% across most metrics.

The computation was executed on CPU-based systems due to constraints and the reliance on classical machine learning techniques. Python's data science stack, including Pandas, NumPy, and Scikit-learn [12, 13, 6], was used without GPU acceleration. Training times ranged from 2.4 minutes to 6.44 hours per model, with a total time of 12.32 hours.

Table 3: Performance metrics on the imbalanced dataset.

Model	Loss	Accuracy	Precision	Recall	F1
SoftMax Regression	1.62	40%	17%	23%	17%
SVC	1.18	42%	16%	18%	16%
Random Forest Classifier	1.28	**56%**	**22%**	**25%**	**22%**

Table 4: Performance metrics on the balanced dataset.

Model	Loss	Accuracy	Precision	Recall	F1
SoftMax Regression	0.95	68%	66%	68%	67%
SVC	0.05	**98%**	**98%**	**98%**	**98%**
Random Forest Classifier	0.23	97%	97%	97%	97%

4 Results

The results demonstrate the potential for providing accurate price predictions while also highlighting significant challenges related to data limitations. The small size and imbalance of the dataset, rooted in constraints during the data collection process, have introduced biases in price and category distributions, affecting the model's overall generalizability.

Despite these challenges, the findings indicate that these limitations are not insurmountable. Addressing data scarcity through collaboration with the Khamsat platform to access more comprehensive datasets or employing data augmentation techniques to generate synthetic samples offers viable pathways to mitigate these issues. While the current implementation of the Khamsat Predictor demonstrates promising results, overcoming these data-related challenges will be crucial to enhancing its performance.

5 Conclusion

This study demonstrates the feasibility of leveraging machine learning to tackle pricing challenges in microservice platforms, as exemplified by the Khamsat Predictor. By analyzing historical data, the predictor provides valuable insights into pricing trends, contributing to greater transparency and trust among users. Future efforts should focus on augmenting the dataset and addressing class imbalance to ensure robust and unbiased predictions. Beyond technical improvements, collaboration with platform operators presents an opportunity to enhance data accessibility and foster practical applications of the predictor.

The code supporting this study: https://github.com/IsmaelMousa/khamsat-predictor.

Acknowledgments

I am grateful to Dr.Adnan Salman for his fruitful comments, corrections, and inspiration.

References

[1] Hsoub, United Kingdom (2011), Khamsat.

[2] Guo, Gongde & Wang, Hui & Bell, David & Bi, Yaxin. (2004). KNN Model-Based Approach in Classification.

[3] McFadden, D. (1972). Conditional Logit Analysis of Qualitative Choice Behavior. UC Berkeley: Institute of Urban and Regional Development.

[4] Cortes, C., Vapnik, V. Support-vector networks. Mach Learn 20, 273–297 (1995).

[5] Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001).

[6] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[7] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In KDD.

[8] Goutte, Cyril & Gaussier, Eric. (2005). A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation. Lecture Notes in Computer Science. 3408. 345-359. 10.1007/978-3-540-31865-$1_2 5$.

[9] Mao, A., Mohri, M., & Zhong, Y. (2023). Cross-Entropy Loss Functions: Theoretical Analysis and Applications. arXiv preprint arXiv:2304.07288.

[10] ISO 5725-1:2023: Accuracy (trueness and precision) of measurement methods and results - Part 1: General principles and definitions., p.2 (2023).

[11] MLflow: A Tool for Managing the Machine Learning Lifecycle.

[12] Data structures for statistical computing in python, McKinney, Proceedings of the 9th Python in Science Conference, Volume 445, 2010.

[13] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020).

[14] Baiju Muthukadan. Selenium with Python, 2011-2024.