Predykcja wolumenu sprzedaży produktów w czasie

Miłosz Hańczyk
milosz.hanczyk@pk.edu.pl

Krzysztof Kołodziejczyk
krzysztof.kolodziejczyk@pk.edu.pl

17 grudnia 2024

1 Abstract

Niniejsza praca bada zastosowanie technik uczenia maszynowego do predykcji wolumenu sprzedaży produktów w czasie. Wykorzystując szeregowy zbiór danych obejmujący dzienne dane sprzedażowe z rosyjskich sklepów, staramy się opracować model prognozowania przyszłej miesięcznej sprzedaży. W naszej analizie testujemy różne metody predykcyjne, w tym regresję liniową, metody oparte o drzewa i lasy losowe oraz głębokie sieci neuronowe po uprzedniej dogłębnej analizie danych. Wydajność modelu oceniamy za pomocą błędu średniokwadratowego (RMSE), a przewidywania są ograniczane do przedziału [0, 20], zgodnie z zasadami konkursu [3] ze strony Kaggle, z której pochodzą dane. Wyniki sugerują, że nasze podejście i sposób pracy skutecznie wychwytuje trendy sprzedażowe, co potwierdza wzrastająca skuteczność podczas przeprowadzania eksperymentów, a najlepszym modelem do zadanego problemu okazuje się XGBoost z odpowiednią inżynierią cech.

2 Wprowadzenie

Predykcja wolumenu sprzedaży produktów jest kluczowym elementem zarządzania łańcuchem dostaw i zapasami w branży detalicznej. Celem tej pracy jest zbadanie różnych technik predykcji wolumenu sprzedaży na przykładzie zbioru danych historycznych pochodzących z Rosji. Zbiór danych pochodzi ze strony Kaggle z konkursu pt. *Predict Future Sales* utworzonego w celach edukacyjnych. Może służyć on jako benchmark do testowania różnych podejść do problemu prognozowania przyszłych wartości sprzedaży w czasie. Autor konkursu zapewnia zbiór danych z jego objaśnieniem oraz przedstawia cel predykcji jako przewidzenie przyszłych miesięcznych wolumenów sprzedaży. Jest to problem z kategorii *Time Series Forecasting*. Po utworzeniu modelu będzie on ewaluowany za pomocą metryki *Root Mean Squared Error (RMSE)*, a same wyniki będą zawężone do przedziału [0, 20], wg. zaleceń autora konkursu. W celu zrozumienia danych i utworzenia potrzebnych cech wykonane zostaną odpowiednie wykresy oraz analizy statystyczne.

3 Zbiór danych

Dane wykorzystywane w niniejszej analizie pochodzą z konkursu „Predict Future Sales" dostępnego na platformie Kaggle [3]. Zbiór danych zawiera szczegółowe informacje dotyczące sprzedaży w rosyjskich sklepach, obejmując okres od stycznia 2013 do października 2015 roku. Składa się z sześciu plików CSV, z których każdy dostarcza różne aspekty informacji:

- **sales_train.csv** – zestaw treningowy zawierający dane historyczne dziennej sprzedaży produktów, uwzględniający m.in. kolumny z datą transakcji, liczbą sprzedanych sztuk (*item_cnt_day*), oraz ceną jednostkową produktu.
- **shops.csv**, **item_categories.csv**, **items.csv** – pliki zawierające odpowiednio szczegóły dotyczące sklepów, kategorii produktów oraz samych produktów.
- **test.csv** – zestaw testowy, który zawiera ID

par sklep-produkt dla okresu prognozy (listopad 2015), na którym będzie oceniana skuteczność modelu. Warto zaznaczyć, że jest to iloczyn kartezjański sklepów z wszystkimi przedmiotami, które zostały sprzedane w zadanym miesiącu.

- **sample_submission.csv** – plik wzorcowy pokazujący format przewidywań wymagany do oceny w konkursie.

Każdy rekord sprzedaży posiada liczne atrybuty, w tym *shop_id* oraz *item_id* (identyfikatory sklepu i produktu), *item_price* (cena produktu) oraz *item_cnt_day* (liczba sprzedanych jednostek). Zmienne kategoryczne i numeryczne posiadają odpowiednie typy danych, a brakujące wartości nie występują. Zbiór charakteryzuje się znaczną różnorodnością z 60 sklepami, 21,170 produktami oraz 84 kategoriami, a dokładne statystyki oraz schemat danych zostały podane w Tabeli 1

Dane sprzedażowe są prezentowane z uwzględnieniem sekwencyjnego numeru miesiąca (*date_block_num*), co umożliwia prowadzenie analiz czasowych. Zgodnie z zasadami konkursu, prognozy ograniczone są do wartości w przedziale [0, 20], jednak autor nie odnosi się do wartości ujemnych kolumny *item_cnt_day* - zostaną one potraktowane jak towary zwrócone. Zbiór testowy odnosi się do listopada 2015 (przypisany $date\_block\_num = 34$). W związku z małą ilością dodatkowych informacji o sklepach i produktach w zestawie testowym, model musi umiejętnie generalizować na podstawie danych historycznych oraz wartości, które da się wygenerować na bazie danych tekstowych.

4 Przegląd literatury

Prognozowanie sprzedaży produktów w czasie jest dobrze zbadanym zagadnieniem w literaturze. Metody klasyczne, takie jak regresja liniowa i modele ARIMA, często stanowią podstawę analiz czasowych. Wraz z rozwojem uczenia maszynowego, algorytmy takie jak lasy losowe, gradient boosting, a także głębokie sieci neuronowe, takie jak LSTM oraz GRU, zaczęły dominować w analizach predykcyjnych. W literaturze można znaleźć liczne przykłady zastosowania modeli opartych na danych historycznych w branży detalicznej, które pokazują, że zaawansowane techniki mogą znacznie poprawić dokładność przewidywań.

5 Motivation

Celem projektu jest zgłębienie wiedzy z zakresu przewidywania szeregów czasowych na niestandardowych problemach, gdzie dane składają się z wielu szeregów czasowych grupowanych przez osobne encje. Projekt ma na celu także przetestowanie różnych modeli głębokich sieci neuronowych i porównanie ich skuteczności z metodą uznawaną za *state of the art*, czyli XGBoost, w kontekście pracy z danymi tabelarycznymi. Dodatkowo, celem jest sprawdzenie się w konkursie na Kaggle, gdzie rozwiązanie wyzwań związanych z przewidywaniem sprzedaży może przynieść wartościowe doświadczenie praktyczne. Projekt jest szczególny ze względu na modelowanie *demandu*, gdzie wartości predykcji mają często skłonność do przyjmowania wartości 0 i wzbudzania się przy określonych warunkach, co stanowi dodatkowe wyzwanie w opracowywaniu odpowiednich metod predykcyjnych, ponieważ nie jest standardem przy zagadnieniu *Time Series Forecasting*.

6 Evaluation

W celu oceny jakości przewidywań wykorzystujemy wskaźnik błędu średniokwadratowego (RMSE). Ten wskaźnik jest powszechnie stosowany w analizach regresyjnych, gdyż penalizuje duże błędy. Nasze przewidywania są ograniczane do przedziału [0, 20], co odpowiada rzeczywistym wartościom w danych. Na dzień publikacji raportu najlepszy wynik na stronie Kaggle to ok. 0.75 na zbiorze testowym, jednak jest on rezultatem dogłębnej pracy przy dostosowywaniu wyników do benchmarku. Patrząc realistycznie wynik rzędu ok 1.0-1.2 RMSE będzie satysfakcjonujący z uwagi na czas poświęcony zadaniu. Przy wdrożeniu produkcyjnym potrzebne byłoby głębsze zrozumienie tematu i więcej wiedzy domenowej, która z pewnością pozwalałaby na jeszcze lepsze wyniki, co pokazują również rozwiązania uczestników, dlatego też nie

Tabela 1: Schemat danych dla zmergowanego zbioru danych

Cecha	Typ danych	Szczegóły
shop_id	kategoryczna	60 unikalnych
shop_name	tekstowa	
item_id	kategoryczna	21,170 unikalnych
item_name	tekstowa	
item_category_id	kategoryczna	84 unikalnych
item_category_name	object	
date	datetime64[ns]	dd/mm/yyyy
date_block_num	porządkowa	
item_price	numeryczna	(0.0, 2169.0]
item_cnt_day	numeryczna	[-22, 2169]

możemy stwierdzić jednoznacznie czy dany wynik pozwoli na wdrożenie systemu.

7 Zasoby

Do implementacji projektu użyto następujących narzędzi: Python w wersji 3.10, biblioteki takie jak pandas, scikit-learn [5], Light XGBoost (lgbm) [1] oraz PyTorch [6]. Modelowanie i analizy zostały przeprowadzone na jednostce z systemem MacOS z procesorem M3 Pro.

8 Zastosowane metody

W analizie wykorzystano kilka różnych modeli predykcyjnych, w tym regresję liniową, las losowy, oraz XGBoost. Dodatkowo, do przewidywania czasowego użyliśmy modelu głębokiej sieci neuronowej opartej o LSTM. W ramach edukacyjnych, jako benchmark przeprowadzono również testy na naiwnym modelowaniu za pomocą średniej globalnej sprzedaży oraz jej wariacji. Każdy z modeli został wytrenowany na danych z (N-2) miesięcy, zwalidowany na danych z (N-1)-go miesiąca oraz przetestowany na danych z N-go miesiąca, którego prawdziwe wartości znajdują się jedynie na platformie Kaggle i są również ukryte dla użytkowników.

9 Eksperyment

9.1 Preprocessing

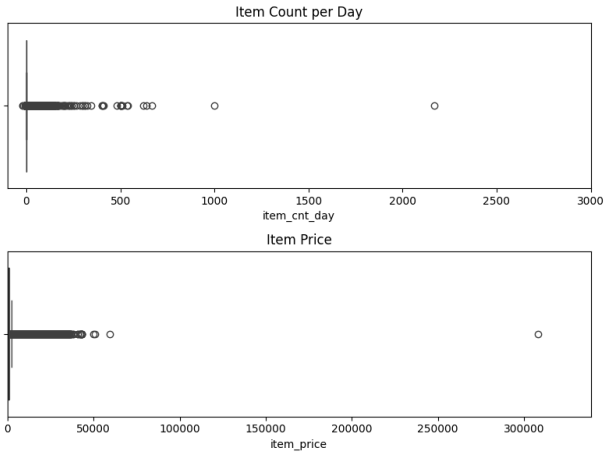
W ramach przygotowania danych do modelowania, przeprowadzono kilka kluczowych kroków przetwarzania wstępnego, aby poprawić jakość i użyteczność zbioru danych. Część z nich pokrywa się dla każdego z używanych modeli, a część jest specyficzna dla konkretnego modelu. W poniższej analizie zestawiono wszystkie wartości, a dokładne cechy użyte w konkretnych modelach przedstawiono w tabelce [TAB].

1. **Wczytanie i połączenie danych**: Zestawy danych `sales_train.csv`, `shops.csv`, `item_categories.csv`, `items.csv`, `test.csv` zostały wczytane i połączone przy użyciu funkcji z zakresu łączenia danych (np. join). Utworzone zostały dodatkowo cechy wyciągnięte z kolumn, które pomagają w zidentyfikowaniu kontekstu z danych tekstowych. Dla sklepów wyciągnięto cechę miasto, a dla kategorii przedmiotów wyciągnięto cechę generalnej kategorii przedmiotów, jak np. Muzyka, Płyty CD.

2. **Tworzenie pełnych zbiorów treningowego i testowego**: Utworzono pełny zbiór treningowy `df_train` oraz testowy `df_test`, a następnie połączono je w zbiór `df`, który zapisano jako `data/data.parquet`.

3. **Eksploracja i analiza wartości odstających**: Zidentyfikowano wartości odstające za po-

mocą wykresów pudełkowych, przedstawionych na Rysunku 1, oraz sprawdzono występowanie błędnych cen i wartości sprzedaży, takich jak $item\_price > 100,000$ oraz $item\_cnt\_day > 1,000$. Takie wartości zostały usunięte z zbioru danych, aby zminimalizować wpływ ekstremalnych przypadków.



Rysunek 1: Box plot

4. **Imputacja brakujących wartości**: Dla produktów o cenie $item\_price \leqslant 0$ zastosowano imputację na podstawie średniej ceny dla danej kombinacji sklepu, produktu i miesiąca.

5. **Zastąpienie duplikujących się identyfikatorów sklepów**: Wykryto duplikaty sklepów (np. shop_id $(0, 57)$, $(1, 58)$) reprezentujące te same fizyczne lokalizacje, które różnią się tylko identyfikatorem. Zmieniono ID duplikatów, aby zredukować redundancję i poprawić interpretowalność modelu. Powtarzające się sklepy pokazano na Rysunku 2

6. **Agregacja danych miesięcznych**: Sumowano dzienne dane sprzedaży do poziomu miesięcznego, uzyskując zmienną `item_cnt_month`, reprezentującą miesięczną liczbę sprzedanych produktów.

7. **Oversampling brakujących kombinacji**:

	shop_id	shop_name_en	city
0	0	Yakutsk Ordzhonikidze, 56 Franchise	Yakutsk
2704872	57	Yakutsk Ordzhonikidze, 56	Yakutsk

	shop_id	shop_name_en	city
9857	1	Yakutsk Central Mall Franchise	Yakutsk
2822300	58	Yakutsk Central Mall	Yakutsk

	shop_id	shop_name_en	city
291381	10	Zhukovsky Chkalov St 39m?	Zhukovsky
312778	11	Zhukovsky Chkalov St 39m²	Zhukovsky

Rysunek 2: Powtarzające się sklepy

Przygotowano pełny zestaw kombinacji sklep-produkt-miesiąc, aby zapewnić, że dla każdego sklepu i produktu będą występować dane dla wszystkich okresów, uzupełniając brakujące wartości zerami. Warto zaznaczyć, że produkt/sklep zostaje dodany do puli występujących kombinacji w miesiącu, gdy pierwszy raz pojawi się w danych. Od tego momentu jest on wliczany do ogólnego produktu kartezjańskiego tworzonego co miesiąc.

8. **Inżynieria cech**:

   - **Cechy sinus i cosinus dla miesiąca**: Dodano przekształcone cechy `month_sin` i `month_cos`, które kodują cykliczność miesięcy, co pomaga modelowi w uchwyceniu sezonowych wzorców sprzedaży.
   - **Cechy opóźnione (lagged features)**: Utworzono cechy opóźnione dla `item_cnt_month` na podstawie wartości sprzedaży z poprzednich miesięcy (do 12 miesięcy wstecz), umożliwiające modelowi analizę trendów.
   - **Cechy kroczące (rolling features)**: Dodano cechy średnich kroczących dla opóźnień miesięcznych, aby model mógł uwzględniać długoterminowe trendy w sprzedaży.

- **Cechy średnich historycznych**: Dodano średnie historyczne wartości sprzedaży (`item_cnt_month`) i ceny (`item_price` na różnych poziomach agregacji, takich jak *(shop_id x item_id)*, *(item_id)*, *(item_id x item_category_id)*, oraz *(item_category_id)*.
- **Obliczenie miesięcy od ostatniego zakupu**: Utworzono cechę `months_since_last_buy`, która przedstawia liczbę miesięcy od ostatniej sprzedaży danego produktu w danym sklepie, co pozwala lepiej uwzględniać okresy braku aktywności. Wartość ta jest równa 0, jeśli transakcja nastąpiła w poprzednim miesiącu i odpowiednio zwiększa się o 1 z każdym kolejnym miesiącem.
- **Optymalizacja typu danych**: Zoptymalizowano typy danych w celu redukcji zużycia pamięci, przypisując kolumnom odpowiednie typy, np. `uint8` dla identyfikatorów kategorii oraz `float16` dla wartości zmiennoprzecinkowych.

Rezultat tych operacji to kompleksowy i zoptymalizowany zbiór danych, zawierający zarówno surowe, jak i wygenerowane cechy, który stanowi solidną podstawę do przeprowadzenia modelowania predykcyjnego.

9.2 Trening modelu

W celu uzyskania optymalnej wydajności w predykcji miesięcznej liczby sprzedanych produktów przetestowano różne algorytmy regresji, w tym XGBoost, regresję liniową, wielowarstwowe perceptrony (MLP), lasy losowe (RandomForest) oraz sieci LSTM. Każdy model został przeszkolony na danych treningowych z okresu Styczeń 2013 – Wrzesień 2015, a jego wydajność oceniono za pomocą metryki RMSE na zbiorze walidacyjnym z Października 2015 roku oraz przetestowano na zbiorze testowym z Listopada 2015 roku. Nie użyto metody walidacji krzyżowej z uwagi na prawdopodobne wystąpienie zjawisko 'data leak', w którym wartości ze zbioru treningowego z przyszłych miesięcy wpływałyby na prognozy ze zbioru walidacyjnego. Wartość, którą prognozowano zawężono do przedziału $[0, 20]$, jeszcze przed procesem trenowania, ponieważ lepiej minimalizowało to wariancję modelu w porównaniu z treningiem na wartościach niezawężonych. Celem było minimalizowanie błędu przewidywań oraz sprawdzenie, który model najlepiej dopasowuje się do danych o dużej liczbie zerowych wartości sprzedaży oraz nieregularnych trendów.

9.2.1 Naive mean

Jako benchmark posłużono się modelem powstałym jako średnia ze wszystkich operacji na miesięcznym poziomie sprzedaży. Posłuży on jako referencja do pozostałych modeli, aby sprawdzić poprawność założeń pozostałych modeli.

9.2.2 Regresja Liniowa (LinearRegression)

Pierwszym modelem zastosowanym w eksperymencie była klasyczna regresja liniowa, która została przeszkolona na przetworzonych danych, w tym na zakodowanych (one-hot encoding) zmiennych kategorycznych oraz cechach numerycznych, takich jak przekształcone sinusoidalne wartości miesiąca oraz szeregi czasowe (cechy opóźnione i kroczące). Regresja liniowa okazała się mało skuteczna w modelowaniu nieliniowych wzorców w porównaniu do innych modeli, stanowiła podstawowy punkt odniesienia dla bardziej złożonych, ponieważ w przeciwieństwie do mdoelu *Naive mean* wprowadza zależność od zmieniających się cech.

9.2.3 Random Tree Regressor

Model drzewa losowego, znany z wysokiej wydajności na danych tabelarycznych, został włączony jako baseline-owy model w eksperymencie. Dzięki możliwości pracy z danymi kategorycznymi, drzewo losowe było dobrze dopasowane do przechwytywania skomplikowanych interakcji między cechami. W modelu zastosowano parametry dobrane ręcznie, aby zredukować nadmierne dopasowanie maksymalizując metryki. Został on dalej rozszerzony o swoje bardziej skomplikowane wersje, tj. las losowy oraz XGBoost.

9.2.4 Random Forest

Model lasu losowego zotał wykorzystany jako naturalne rozszerzenie drzewa losowego. Podobnie jak poprzedni, został ręcznie zoptymalizowany i przetestowany.

9.2.5 XGBoost

Model XGBoost cechował się wysoką skutecznością w przechwytywaniu niestabilnych wzorców sprzedaży w różnych okresach. Wskazywał najmniejsze wartości błędu pomimo braku skomplikowanych testów hiperparametrów i użycia najmniejszej liczby zmiennych. Z tego powodu został on również wybrany do dalszego dopasowywania w kroku optymalizacji hiperparametrów.

9.2.6 Wielowarstwowy Perceptron (MLP)

Do modelowania złożonych, nieliniowych zależności w danych wybrano standardowy model sieci neuronowej typu MLP. Po wstępnych testach testowano model o architekturze [1024, 512, 256, 32, 1], gdzie każda z tych wartości oznacza liczbę neuronów w warstwie. Po warstwach ukrytych dodano etapy Dropoutu, Normalizacji Batchowej oraz funkcję nieliniową (w tym przypadku ReLU [4]). Proces treningowy obejmował kodowanie zmiennych kategorycznych oraz normalizację cech numerycznych. Sieć neuronowa okazała się skuteczna w rozpoznawaniu nieliniowych zależności w danych, co pozwoliło na poprawę wyników w stosunku do modelu np. regresji liniowej.

9.2.7 Sieć LSTM (Long Short-Term Memory)

Jako zaawansowany model do analizy szeregów czasowych, LSTM został zastosowany do przewidywania wolumenu sprzedaży. LSTM jest specjalnie przystosowany do przechwytywania długoterminowych zależności w danych, co było szczególnie przydatne przy analizie nieregularnych wzorców sprzedaży na przestrzeni miesięcy. Model uczono na sekwencjach danych z ostatnich kilkunastu miesięcy, przy czym architektura sieci LSTM i liczba warstw została dobrana eksperymentalnie, aby zoptymalizować wydajność. W modelowaniu pozbyto się niektórych cech typu lagged oraz rolling, ponieważ założono, że część z tych informacji będzie płynęła z sekwencji danych czasowych.

9.2.8 Porównanie wyników

Każdy z powyższych modeli był oceniany pod kątem wydajności przy użyciu RMSE, a ich wyniki były ze sobą porównywane. Poniższa tabela 2 przedstawia RMSE dla każdego modelu na zbiorze walidacyjnym:

Model	RMSE Validation
Naive Mean	0.508
Lin. Reg.	0.458
Decision Tree	0.463
Random Forest	0.
XGBoost	0.379
MLP	XX.XX
LSTM	XX.XX

Tabela 2: Porównanie wyników modeli na zbiorze walidacyjnym.

9.3 Optymalizacja

W celu poprawy wydajności modelu XGBoost, przeprowadzono optymalizację hiperparametrów na dwa sposoby, za pomocą biblioteki Optuna [2] oraz z użyciem algorytmu genetycznego. Optuna umożliwia automatyczne przeszukiwanie przestrzeni hiperparametrów przy użyciu strategii optymalizacji Bayesowskiej, co pozwala na szybsze i bardziej efektywne znalezienie optymalnych parametrów. Podejście związane z użyciem algorytmu genetycznego opierało się na bibliotece PyGAD [7].

9.3.1 Proces Optymalizacji dla XGBoost

Jako najbardziej obiecujący model wybrano XGBoost, na którym również przeprowadzano przeszukiwanie przestrzeni hiperparametrów. Optymalizacja miała na celu minimalizację błędu RMSE na zbiorze walidacyjnym. Przygotowano funkcję celu (*objective*), która przyjmuje różne kombinacje hiperparametrów,

trenuje model `LGBMRegressor` i ocenia jego skuteczność na podstawie metryki RMSE.

Optymalizowane hiperparametry obejmowały:

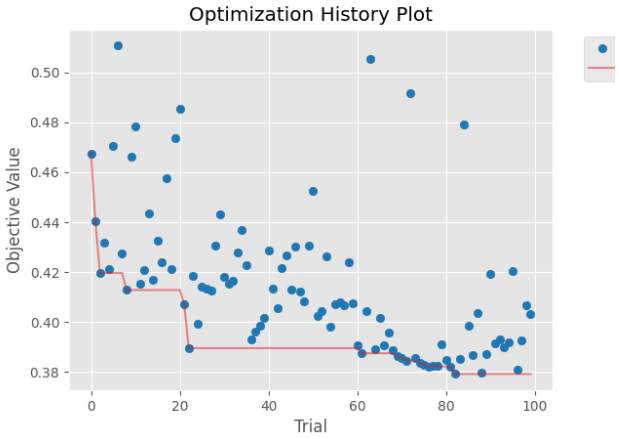
- `num_leaves`: liczba liści drzewa decyzyjnego, zakres od 20 do 150,
- `max_depth`: maksymalna głębokość drzewa, zakres od -1 (bez ograniczeń) do 50,
- `learning_rate`: współczynnik uczenia, od 1×10^{-5} do 1.0,
- `n_estimators`: liczba estymatorów, ustawiona na stałą wartość 50 (ze względu na ograniczenia zasobów),
- `reg_alpha` i `reg_lambda`: współczynniki regularyzacji L1 i L2, zakres od 1×10^{-3} do 10.0,
- `colsample_bytree`: współczynnik kolumn do losowego próbkowania, wartości dyskretne od 0.3 do 1.0,
- `subsample`: współczynnik próbkowania dla próbek, wartości dyskretne od 0.4 do 1.0,
- `min_child_samples`: minimalna liczba próbek w liściu, od 1 do 300,
- `cat_smooth` (znany również jako `min_data_per_groups`): gładkość kategorii, od 1 do 100.

Przeprowadzono 100 iteracji (`n_trials=100`), każda z inną kombinacją hiperparametrów, aby znaleźć minimalną wartość RMSE. Na Rysunku 3 przedstawiono historię przeszukiwań przestrzeni z wykresem aktualnie najlepszego znalezionego rozwiązania.

Na Rysunku 4 przedstawiono wizualizację różnych konfiguracji hiperparametrów. Bazując na tym wykresie można wnioskować, o

Rysunek 5 pokazuje relacje między różnymi wartościami wybranych hiperparametrów a wynikiem. Widać z nich trendy przeszukiwania Bayesowskiego, które znajdując coraz lepsze rozwiązanie, dalej przeszukiwało dany fragment przestrzeni. Ciemniejszy kolor oznacza lepszy wynik.

Spośród wszystkich hiperparametrów **learning_rate** okazał się być najbardziej znaczącym,



Rysunek 3: Historia optymalizacji hiperparametrów

który najbardziej wpływał na wyniki optymalizacji. Dokładne wyniki procentowe wag ważności hiperparametrów przedstawia Rysunek 6.

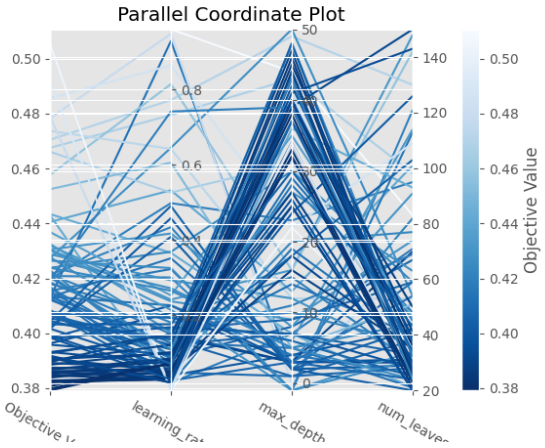
9.3.2 Wyniki Optymalizacji

Optymalizacja zakończyła się uzyskaniem najlepszego zestawu parametrów, który poprawił wydajność modelu XGBoost na zbiorze walidacyjnym, zapewniając niższy błąd RMSE. Najlepsze parametry oraz uzyskany wynik RMSE zostały zapisane i wykorzystane do ostatecznego treningu modelu oraz do generowania przewidywań na zbiorze testowym. Tabela 3 zawiera najlepsze hiperparametry oraz zoptymalizowany wynik.

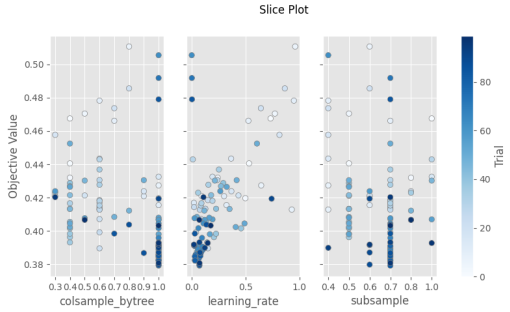
9.3.3 Optymalizacja z użyciem algorytmu genetycznego

Optymalizacja hiperparametrów została przeprowadzona również przy użyciu algorytmu genetycznego z biblioteki PyGAD [7]. Wykorzystano następujące parametry algorytmu:

- `Liczba Generacji`: 5,
- `Liczba Rozwiązań wybieranych jako Rodzic`: 4,



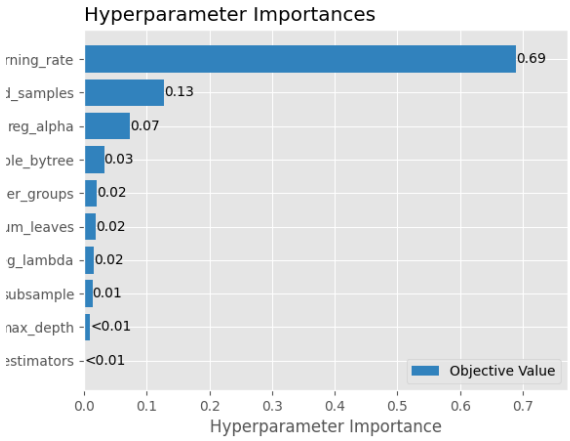
Rysunek 4: Wykres Równoległych Koordynatów



Rysunek 5: Wartosci końcowe dla wybranych hiperparametrow

- `Populacja`: 10,
- `Typ krzyżowania`: Single-point,
- `Prawdopodobieństwo mutacji`: 20%.

Najlepszy zestaw hiperparametrów został znaleziony po 5 generacjach, osiągając wynik **RMSE na walidacji równy 0.384**. Otrzymane wartości zostały wykorzystane do dalszego treningu modelu. W porównaniu do podejścia z użyciem biblioteki Optuna otrzymano nieco gorsze wyniki.



Rysunek 6: Ważność hiperparametrów w optymalizacji

Parameter	Value
num_leaves	28
max_depth	43
learning_rate	0.071
n_estimators	50
reg_alpha	1.862
reg_lambda	0.608
colsample_bytree	1.0
subsample	0.7
min_child_samples	236
min_data_per_groups	89
RMSE Validation	**0.379**

Tabela 3: Zoptymalizowane hiperparametry

9.3.4 Dobór cech z użyciem algorytmu genetycznego

W celu przeprowadzenia selekcji cech wykorzystano algorytm genetyczny z biblioteki PyGAD [7]. Optymalizacja została wykonana z użyciem hiperparametrów modelu uzyskanych wcześniej za pomocą biblioteki Optuna.

Funkcja fitness

$$\frac{1}{10^{-6} + RMSE}$$

Funkcja fitness powinna osiągać jak najwyższe wyniki dla coraz lepszych rozwiązań. W tym celu musimy odwrócić wartość RMSE, ponieważ maleje ono dla coraz lepszych rozwiązań.

Parametry algorytmu W trakcie selekcji cech algorytm genetyczny wykorzystano z następującymi ustawieniami:

- `Liczba Generacji`: 5,
- `Liczba Rozwiązań wybieranych jako Rodzic`: 4,
- `Populacja`: 10,
- `Typ krzyżowania`: Single-point,
- `Prawdopodobieństwo mutacji`: 20%.

Wyniki Najlepsze rozwiązanie uzyskane po 5 generacjach reprezentowało podzbiór cech, który został wybrany do dalszego treningu modelu. Finalny zestaw cech obejmował:

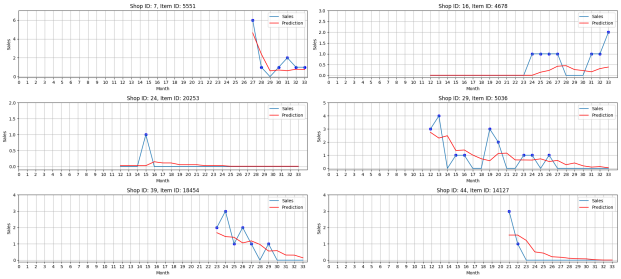
- month_cos
- lagged_1
- lagged_3
- lagged_6
- lagged_7
- lagged_8
- lagged_11
- rolling_6
- rolling_9
- avg_shop_item_item_cnt_day_lag_1
- avg_item_item_price_lag_1
- avg_shop_item_category_item_price_lag_1
- avg_shop_item_category_item_cnt_day_lag_1
- avg_item_category_item_price_lag_1
- months_since_last_buy

Model trenowany na wybranych cechach oraz zoptymalizowanych hiperparametrach osiągnął wynik **RMSE na walidacji równy 0.375**. Wynik ten okazał się lepszy w porównaniu do optymalizacji przeprowadzonej na pełnym zbiorze cech.

Wyniki potwierdzają, że zastosowanie algorytmu genetycznego w zadanym problemie do selekcji cech może prowadzić do poprawy jakości predykcji oraz zmniejszenia złożoności modelu.

10 Wizualizacja wyników

W analizie przedstawione zostaną wyniki dla wybranych kombinacji sklep-produkt w czasie. Na wykresach niebieską linią oznaczono dane historyczne sprzedaży w czasie, natomiast czerwoną linią przedstawiono predykcję uzyskaną za pomocą modelu LightGBM. Wyniki zerowej sprzedaży odzwierciedlają sytuacje, w których dany produkt nie był sprzedawany w określonym sklepie w danym miesiącu, mimo że pojawił się on po raz pierwszy w tym lub poprzednim miesiącu w jakimkolwiek innym sklepie. Taka sytuacja ilustruje ograniczenia wynikające z dynamiki wprowadzania produktów do sprzedaży w różnych lokalizacjach. Rysunek 7 obrazuje predykcje dla wybranych kombinacji.



Rysunek 7: Predykcje w czasie dla algorytmu LGBM

11 Podsumowanie

Celem eksperymentu było stworzenie modelu predykcyjnego, który precyzyjnie przewidywałby miesięczną

sprzedaż produktów na podstawie szeregu czasowego, zawierającego dane historyczne ze sklepów w Rosji. Proces rozpoczęto od wstępnego przetworzenia danych, łącząc różne źródła, usuwając odstające wartości oraz tworząc nowe cechy, które wzbogaciły analizę i modelowanie. Dzięki odpowiedniej inżynierii cech, w tym dodaniu zmiennych opóźnionych, średnich kroczących oraz wyrażeń sinusoidalnych, przygotowano dane do zasilenia modeli.

Przeprowadzono eksperymenty z różnymi algorytmami, w tym XGBoost, regresją liniową, wielowarstwowymi perceptronami, lasami losowymi oraz sieciami LSTM. Każdy model testowano pod kątem skuteczności w uchwyceniu złożonych trendów sprzedaży. Najlepiej dopasowanym modelem okazał się XGBoost, którego hiperparametry dodatkowo zoptymalizowano za pomocą Optuna. W wyniku optymalizacji uzyskano zestaw parametrów, który zapewnił minimalizację błędu RMSE na zbiorze walidacyjnym.

Całość eksperymentu wykazała, że podejście polegające na zastosowaniu zaawansowanej inżynierii cech, wzbogacone o proces optymalizacji hiperparametrów, znacząco poprawiło dokładność prognoz. Model XGBoost z optymalnymi hiperparametrami najlepiej odpowiadał na wyzwania związane z analizą niestabilnych i nieregularnych wzorców w danych sprzedażowych, dostarczając solidnych wyników predykcyjnych.

Literatura

[1] lgbm. `https://lightgbm.readthedocs.io/en/stable/`. Accessed: 2024-11-10.

[2] optuna. `https://github.com/optuna/optuna`. Accessed: 2024-11-10.

[3] Predict future sales. `https://www.kaggle.com/competitions/competitive-data-science-predict-future-sales`. Accessed: 2024-11-06.

[4] relu. `https://en.wikipedia.org/wiki/Rectifier_(neural_networks)`. Accessed: 2024-11-10.

[5] scikit-learn. `https://scikit-learn.org/stable/`. Accessed: 2024-11-10.

[6] torch. `https://pytorch.org`. Accessed: 2024-11-10.

[7] Ahmed Fawzy Gad. Pygad: An intuitive genetic algorithm python library. *Multimedia Tools and Applications*, pages 1–14, 2023.