Sprawozdanie z projektu indywidualnego: "Narzędzie do wyceny laptopów w oparciu o techniki analizy danych i uczenie maszynowe"

Jan Machowski, 325492

semestr letni, 2024

Spis treści

1 Informacje ogólne

1.1 Założenia projektu

Celem projektu jest stworzenie narzędzia do wyceny laptopów w oparciu o techniki analizy danych i uczenie maszynowe. Zbiór danych uczenia jest ustalony z góry, można wykorzystać gotowe zbiory lub stworzyć własny za pomocą technik web-scrapping'u. W oparciu o dane należy zbudować model uczenia maszynowego, który będzie przeprowadzał wycenę.

1.2 Kroki podjęte w celu realizacji projektu

Aby zrealizować projekt podjęto następujące kroki:

1. Pobranie danych `laptopPrice.csv`;
2. Analiza danych i pierwsze uczenie modeli;
3. Ocena i podjęcie decyzji o zmianie zbioru danych;
4. Pobranie danych `data.csv`;
5. Druga analiza danych i uczenie modeli;
6. Pozytywna ocena efektów uczenia i rozpoczęcie projektowania aplikacji;
7. Tworzenie funkcji filtrujących i przekształcających dane;
8. Tworzenie funkcji przekazujących dane między modułami;
9. Tworzenie interfejsu użytkownika;
10. Poprawa metody przygotowania parametrów do przewidywania;
11. Implementacja API w celu pobrania kursów walut;
12. Ostateczne poprawki estetyczne.

1.3 Wykorzystane technologie

W celu napisania narzędzia, zarówno frontend'u jak i backend'u, wykorzystano język Python. Wykorzystano elementy z modułów:

1. **pandas** - wczytywanie danych, ramki oraz przekształcanie;
2. **sklearn**:
    - linear_model - modele uczenia maszynowego;
    - ensemble - modele uczenia maszynowego;
    - metrics - ocena modelu.
3. **math** - wybrane funkcje matematyczne;
4. **re** - wyrażenia regularne, w celu modyfikacji danych;
5. **requests** - obsługa API, pobieranie walut;
6. **tkinter** - graficzny interfejs użytkownika;
7. **pyinstaller** - spakownaie aplikacji.

1.4 Możliwości programu

Program NWDL służy do wyceny ceny laptopów na podstawie wybranych przez użytkownika parametrów. Jego praktycznym zastosowaniem jest estymacja ceny rynkowej, jaką powinien mieć laptop o określonych parametrach - może to być wykorzytane zarówno przez indywidualne osoby, jak i firmy.

1.5 Interfejs użytkownika

Interfejs użytkownika składa się z prostego okna podzielonego na tematyczne sekcje. Każda sekcja zawiera listy wysuwane, odpowiadające cechom laptopów.

Sekcje i listy wysuwane zawarte w programie:

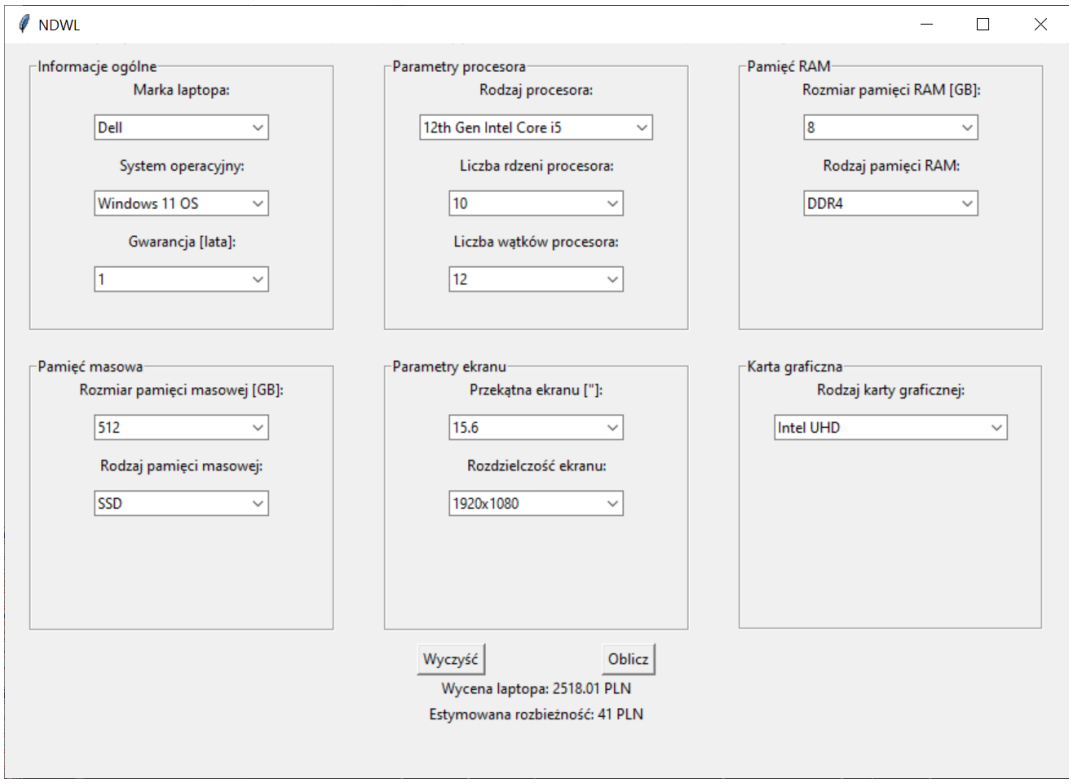
1. Informacje ogólne - marka laptopa, system operacyjny, gwarancja;
2. Parametry procesora - rodzaj procesora, liczba rdzeni, liczba wątków;
3. Pamięć RAM - rozmiar pamięci RAM, rodzaj pamięci RAM;
4. Pamięć masowa - rozmiar pamięci masowej, rodzaj pamięci masowej;
5. Parametry ekranu - przekątna ekranu, rozdzielczość ekranu;
6. Karta graficzna - rodzaj karty graficznej.

W celu wyświetlenia parametrów, są one najpierw pobrane z bazy danych, przekształcone (patrz 2.2.3) oraz przekazane do interfejsu użytkownika. Należy wspomnieć, iż parametry wyświetlane użytkownikowi mogą być filtrowane na podstawie wcześniejszych wyborów, aby zachować spójność logiczną, przykładowo aby użytkownik nie był w stanie wybrać procesora z 12 rdzeniami i 10 wątkami.

Pod sekcjami znajdują się dwa przyciski:

- Wyczyść - przycisk, który ustawia wartości wszystkich parametrów na "n/a";
- Oblicz - przycisk służący do wyznaczenia wyceny laptopa.

Gdy przycisk "Oblicz"zostanie wciśnięty, na spodzie okienka wyświetli się wycena laptopa oraz estymowana rozbieżność tej ceny, wyznaczana na podstawie błędu średniokwadratowego.[1]



Rysunek 1: Graficzny interfejs użytkownika z przykładową wyceną.

[1]W przypadku wyznaczania ceny przy braku pasujących danych, co jest opisane w sekcji 2.2.4, cena względem której wyznaczany jest błąd średniokwadratowy jest ceną w danych standardowych.

1.6 Instrukcja obsługi

Program można uruchomić za pomocą pliku `NDWL.exe`[2] Opcjonalnie, można uruchomić program z linii poleceń, mając zainstalowany kompilator języka python. Wówczas należy wywołać program `main.py`. Przykładowo dla środowiska PowerShell, komenda:

```
PS C:\Users\janma> python .\main.py
```

W celu wykorzystania programu, należy:

1. Wybrać parametry laptopa, które nas interesują, z list wysuwanych. Pozostawienie wartości "n/a"oznacza brak wyboru parametru;
2. Wcisnąć przycisk "Oblicz".

1.7 Opisy modułów i funkcji

Program składa się z dwóch modułów:

1. main;
2. datamod.

1.7.1 Moduł main

Moduł main jest głównym modułem programu. Moduł ten posiada graficzny interfejs użytkownika, funkcje wprost związane z użytkownikiem i akcjami wykonywanymi podczas korzystania z programu. Moduł zawiera funkcje:

1. **ev*()**
   Zbiór funkcji obsługujących zmiany w listach wysuwanych tak, aby zapytania wysyłane do DataMod były sensowne;
2. **caseApple()**
   Funkcja obsługująca filtrację danych związaną z parametrami natywnymi dla komputerów marki Apple;
3. **getCBoxValues()**
   Funkcja tworząca tablicę parametrów, które użytkownik wybrał spośród dostępnych;
4. **checkValues()**
   Funkcja sprawdzająca, czy użytkownik wybrał minimum jeden parametr laptopa;
5. **butter()**
   Funkcja wyświetlająca szacowaną cenę laptopa na podstawie wejściowych parametrów;
6. **butclr()**
   Funkcja ustawiająca wartość każdej listy wysuwanej na "n/a".

1.7.2 Moduł datamod

Moduł datamod jest modułem logicznym programu. Ten moduł zawiera klasę DataMod, która jest odpowiedzialna za pobranie i przekształcanie danych, uczenie maszynowe oraz predykcję i ocenę predykcji. Klasa DataMod jest wykorzystywana przez moduł main w celu wydobycia poszczególnych parametrów laptopów oraz wyznaczenia wyceny laptopa. Moduł zawiera funkcje:

1. **clearData()**
   Funckja przekształcająca dane wejściowe tak, aby ułatwić użytkowanie i uczenie;

[2]Niektóre programy antywirusowe mogą zareagować przy uruchomieniu pliku. Zalecane jest wyłączenie ich lub uruchomienie programu poprzez IDE.

2. **enhanceData()**
   Funkcja przekształcająca dane tak, aby ujednolicić rodzaje procesorów CPU oraz GPU, liczbę rdzeni i wątków oraz rozdzielczość ekranu;
3. **teachModel()**
   Funkcja ucząca modele na podstawie przekształconych danych;
4. **prepareParams()**
   Funkcja przekształcająca parametry otrzymane z modułu main tak, aby model mógł je poprawnie zinterpretować;
5. **getPrice()**
   Funkcja wyznaczająca wycenę;
6. **getRMS()**
   Funckja wyznaczająca błąd średniokwadratowy wyceny;
7. Inne funkcje o składni **get*()**
   Funckje udostępniające poszczególne wartości parametrów dostępnych w bazie danych.

Dokładniejsze opisy logiki aplikacji można znaleźć w sekcji 2.2.

2 Szczegółowy opis realizacji

2.1 Analiza danych

2.1.1 Wstępne założenia

Z uwagi na założenie projektu, tj. wyznaczenie pewnej wartości jednego predyktora (tj. ceny laptopa) na podstawie wartości innych predyktorów, postanowiono wykorzystać modele regresyjne lub klasyfikacyjne. Sprawdzono działanie następujących modeli:

- regresję liniową;
- model Lasso, opierający się na normie L1;
- model Ridge, opierający się na normie L2;
- model ElasticNet, łączący normy L1 i L2;
- model lasów losowych;
- model wzmacnienia gradientowego drzew (reg + klas);
- model histogramowego wzmacnienia gradientowego drzew;
- algorytm AdaBoost (reg + klas);
- model regresji wspartej wektorem SVR;
- klasyfikację najbliższych prototypów;
- klasyfikację K-najbliższych sąsiadów.

Na podstawie ich oceny, której dokonano za pomocą wyznacznika determinacji R^2, błędu średniokwadratowego RMS oraz czasu uczenia i predykcji wybrano najlepsze modele, co opisano w sekcji 2.1.3

2.1.2 Pozyskanie danych

Zbiór danych wykorzystany do projektu został pozyskany ze strony kaggle.com.

Początkowo wykorzystano dane zawarte w pliku `laptopPrice.csv`. Po dokonaniu analizy, uczenia wybranych modeli i oceny ich przydatności uznano, że owe dane nie będą dawały satysfakcjonujących rezultatów. Z tego powodu, wybrano drugi zestaw danych - `data.csv`.

Dokładniejszy opis procesu analizy i powodu zmiany został opisany w sekcji 2.1.3.

2.1.3 Pierwsza analiza danych

Przed rozpoczęciem pisania programu, dokonano doświadczeń i prób porównujących modele na pozyskanych danych. W tym celu stworzono notatnik Jupyter, którego nazwa później została zmieniona na `oldData.ipynb`. Notatnik bazuje na danych `laptopprice.csvi` występuje w nim kodowanie danych kategorycznych za pomocą **kodu 1 z n**, porównanie metod regresji oraz wyszukiwanie najlepszych parametrów określonych metod.

Determinacja parametrów modeli została dokonana dzięki klasie GridSearchCV z pakietu sklearn.model_selection. W pierwszym kroku stworzono odpowiednią tablicę parametrów, które klasa ma przetestować, a następnie przetestowano model za pomocą klasy. Parametry w obiektach param_grid_* zostały wybrane arbitralnie.

Przykład kodu:

```
param_grid_rf = {
    'n_estimators': [10, 50, 100, 500, 1000],
    'max_depth': [3, 5, 7, 9, 11, 13]
}
rfr = RandomForestRegressor()

rfr2 = GridSearchCV(rfr, param_grid_rf, cv=3, n_jobs=-1)
rfr2.fit(X_train, y_train)

print(rfr2.best_params_)
```

Podczas oceny skuteczności modeli brano pod uwagę następujące cechy:

- wartość współczynnika determinacji R^2, wyłącznie dla regresji;
- wartość walidacji krzyżowej CV;
- wartość błędu średniokwadratowego RMS;
- czas, potrzebny na wyuczenie oraz predykcję modelu.

Notatnik podzielono na części dedykowane poszczególnym modelom - regresyjnym i klasyfikacyjnym. Poniżej zamieszczono wyniki doświadczenia:

Tabela 1: Ocena skuteczności modeli regresji

Model regresji	R^2	CV	RMS	Czas uczenia [s]
Liniowa	0.7485	0.6708	657.36	1.6
Lasso	0.7487	0.6714	657.72	0.0
Ridge	0.7486	0.6709	657.56	0.0
Lasy losowe	0.7040	0.8281	544.13	1.9
Wzmacnianie gradientowe	0.8549	0.7591	495.40	3.3
Wzmacnianie gradientowe histogramowe	0.7752	0.6911	611.56	1.3
AdaBoost	0.6095	0.5427	1021.94	4.7
SVR	0.5703	0.5470	850.18	0.3
ElasticNet	0.7487	0.6714	657.62	0.0

Tabela 2: Ocena skuteczności modeli klasyfikacji

Model klasyfikacji	CV	RMS	Czas uczenia [s]
Najbliższe prototypy	0.0152	1022.80	0.0
KNN	0.02	1417.04	0.0
Lasy losowe	0.0805	691.85	0.3
Wzmacnianie gradientowe	0.0562	1541.43	30.3
AdaBoost	0.0684	922.54	2.3

Jak widać, modele klasyfikacyjne osiagają o wiele gorsze wyniki niż regresyjne. Tym samym zrezygnowano z wykorzystania modeli klasyfikacyjnych podczas projektu.

Na podstawie danych z tabeli 1 wyznaczono wartości średnie. Wartości te zostały jednak znacząco zaniżone przez wyniki modeli AdaBoost oraz SVR. Z uwagi na to, wyznaczono średnie arytmetyczne wartości tylko siedmiu modeli.

Średnie wartości przedstawiają się następująco:

- $\overline{R^2} = 0.7612$;
- $\overline{CV} = 0.6938$;
- $\overline{RMS} = 611$;
- średni czas uczenia = 1.2 s.

Dokonując analizy wyników zauważono, że żaden z modeli nie osiągnął odpowiedniego wyniku. Najwyższą dokładność posiadał model wzmacniania gradientowego lasów wyjaśniając 85% danych, kosztem relatywnie dużego czasu uczenia modelu. Nie jest to zły wynik, lecz nie jest również satysfakcjonujący.

Z uwagi na relatywnie niski współczynnik determinacji oraz wynik walidacji krzyżowej dokonano decyzji znalezienia nowego zbioru danych.

2.1.4 Druga analiza danych

Z uwagi na niezadawalające wyniki determinacji dla poprzednich danych, postanowiono wyszukać nowych. Ponownie na stronie kaggle.com, znaleziono zbiór danych `data.csv`. W związku z tym, stworzono nowy notatnik jupyter do analizy zbioru i oceny uczenia poszczególnych modeli: `new.ipynb`.

Podobnie jak w poprzednim notatniku, dokonano wstępnego przekształcenia danych kategorycznych oraz sprawdzono ich zdolność w predykcji ceny laptopów.

Poniżej przedstawiono wyniki analizy:

Tabela 3: Druga ocena skuteczności modeli regresji

Model regresji	R^2	CV	RMS [PLN]	Czas uczenia [s]
Liniowa	0.9477	0.7353	653.95	0.4
Lasso	0.9477	0.7349	653.95	0.2
Ridge	0.9357	0.7606	710.86	0.1
Lasy losowe	0.9589	0.6817	679.59	1.7
Wzmacnianie gradientowe	0.8549	0.7591	495.40	1.5
Wzmacnianie gradientowe histogramowe	0.9161	0.6940	828.40	6.1
AdaBoost	0.6629	0.3571	1661.21	1.2
SVR	0.3031	0.2524	2388.55	0.3
ElasticNet	0.9571	0.6833	592.31	0.7

Ponownie, wartości algorytmów AdaBoost oraz SVR znacząco różnią się od reszty. Można z pewnością założyć, że te dwa modele nie spełniają oczekiwań na potrzeby aplikacji.

Bez dwóch ww. algorytmów, średnie arytmetyczne wartości przedstawiają się następująco:

- $\overline{R^2} = 0.9311$;
- $\overline{CV} = 0.7212$;
- $\overline{RMS} = 659$;
- średni czas uczenia = 1.5 s.

Jak wynika z powyższej tabeli, dla danych `data.csv` modele osiągały niewiele lepsze wartości w przypadku oceny walidacji krzyżowej, błędu średniokwadratowego oraz czasu uczenia, jednak współczynniki determinacji modeli znacznie się poprawiły - wzrosły o średnio 17%.

W przypadkach modeli, które w poprzedniej analizie miały wyznaczane parametry również zastosowano funkcję GridSearchCV. Zabieg ten nieznacznie zwiększył dokładność modeli, ale sprawił że uczenie trwało o wiele dłużej.

Poniższe porównanie wykonano dla regresji lasów losowych. Wyniki dla pozostałych regresji były analogiczne:

Tabela 4: Porównanie oceny modelu RandomForestRegressor dla różnych parametrów

n_estimarots	max_depth	R^2	CV	RMS [PLN]	Czas uczenia [s]
1000	13	0.9602	0.6752	573.07	33.5
500	13	0.9590	0.6759	578.93	17.2
250	13	0.9612	0.6767	563.53	8.5
100	13	0.9570	0.6788	593.09	3.4
50	13	0.9556	0.6767	579.14	1.7
50	10	0.9462	0.6669	663.59	1.4
50	11	0.9529	0.6609	620.91	1.5
50	12	0.9549	0.6753	607.05	1.7
50	14	0.9603	0.6664	569.70	1.8
50	15	0.9602	0.6735	557.51	1.9

Należy zaznaczyć, że w zależności od przebiegu uczenia modelu lasów losowych, parametry będą się różnić. Różnice te są nieznaczne - podczas przeprowadzania doświadczenia nie stwierdzono różnicy wartości poszczególnych ocen większej niż 1%, zakładając te same parametry modelu.

Spośród powyższych modeli wybrano pięć, które wykorzystano w końcowej implementacji projetu. Są to:

- Regresja liniowa;
- Model Lasso;
- Model Ridge;
- Regresja lasów losowych;
- Regresja z gradientowym wzmacnianiem drzew.

Powyższe modele zostały wybrane z uwagi na najlepsze wyniki w doświadczeniu. Aby otrzymać jak najlepsze wyniki,

2.2 Tworzenie aplikacji

Posiadając zbiór danych oraz modele, które można wykorzystać do wyceny laptopu, można było rozpocząć pracę nad finalnym programem.

2.2.1 Podział na moduły

Wpierw, dokonano podziału na moduły main oraz datamod, dla klarownego podziału na część logiczną i część interfejsową.

W module main zawarto wszystkie funkcje związane z komunikacją z użytkownikiem. Drugi moduł zawiera logikę aplikacji, czyli uczenie maszynowe, przekształcanie danych wejściowych i przesyłanie wyników do modułu main.

2.2.2 Graficzny interfejs użytkownika

Stworzenie graficznego interfejsu użytkownika było prostym zadaniem. Zdecydowano się wykorzystać oddzielne ramki do tematycznego podziału cech laptopa. Wykorzystano listy wysuwane w celu przedstawienia poszczególnych wartości cech użytkownikowi, z uwagi na ich prostotę, intuicyjność oraz łatwość w zarządzaniu.

2.2.3 Przekształcanie i filtrowanie danych wejściowych

W celu zachowania jasności oraz spójności z rzeczywistością, należy przekształcić dane tak, aby nie stracić możliwości uczenia, usprawnić tę możliwość oraz zyskać dodatkową klarowność programu. Aby to uzyskać podjęto następujące kroki:

1. Przekonwertowano ceny z Indyjskich Rupii na Polskie Złote, za pomocą API Frankfurter

2. Ujednolicono rodzaje procesorów CPU oraz GPU;

3. Rozdzielono kolumnę *CPU* na kolumny *cores* i *treads*;

4. Usunięto jendostki z danych liczbowych, tj. rozmiarów pamięci RAM oraz pamięci masowej, oraz przekształcono je na typy liczbowe zamiast kategorycznych;

5. Ujednolicono wymiary rozdzielczości ekranów.

Ujednolicenie rodzajów procesorów CPU i GPU polegało na usunięciu części nazwy procesora odpowiedzialnej za doprecyzowanie modelu, pozostawiając tylko rodzinę procesora. Przykładowo, model procesora "5th Gen AMD Ryzen 5 5600HS"stał się "5th Gen AMD Ryzen 5". W tym celu, wykorzystano bibliotekę `re`, zawierającą wyrażenia regularne - wówczas, usunięcie ostatniego ciągu znaków sprowadziło się do prostej formuły zamiany ciągu na symbol pusty.

W ramach ujednolicenia wyszukano oraz poprawiono kodem nieścisłości w danych. Częstym zdarzeniem dla procesorów graficznych okazało się nazywanie tej samej rodziny procesorów na różne sposoby. Przykładowo, znalazły się takie nieścisłości jak, ale nie wyłącznie:

- podwójne spacje;
- *Intel Integrated Integrated*;
- *Intel Integrated Intel*;
- *Intel Intel*;
- *GeForce, Geforce i GEFORCE*;
- *GeForce RTX RTX*;
- *AMD Radeon AMD*;

Krok ten nie tylko ujednolica parametry dla łatwiejszego użytkowania, ale wspomaga model uczenia maszynowego poprzez znaczne ograniczenie liczby zmiennych kategorycznych, które należy zakodować.

W celu przekształcenia danych kategorycznych-symbolicznych (m.in. rodzaj procesora, marka laptopa czy typ pamięci RAM) na typ umożliwiający wykorzystanie ich w procesie uczenia maszynowego wykorzystano funkcję `get_dummies` z biblioteki pandas. Funkcja ta tworzy nową ramkę danych traktując dane kategoryczne wedle zasady kodu 1 z n: dodawane są kolumny dla każdej odmiennej wartości, następnie wypełniane wartością True albo False, w zależności od tego czy zgadza się z oryginalnym opisem.

Wartą komentarza jest również kolumna *CPU*. W oryginalnych danych, jej wartości najczęściej posiadają postać: "X Cores, Y Threads", z ewentualnymi dodatkami związanymi z rodzajem rdzeni. Odkryto, że m.in. dla marek Asus, Tecno, Lenovo czy HP, liczba rdzeni była podawana słownie, np. "Hexa Core, 12 Threads".

Podjęto kroki w celu unormowania tych danych - wartości słowne zostały przekształcone na liczbowe, następnie stworzono dwa oddzielne parametry - *cores* i *threads*, przechowujące odpowiednio liczbę rdzeni i wątków. Dane te nie zostały jednak przekształcone na kategoryczne-porządkowe, z uwagi na brak danych dot. procesorów firmy Apple - w wielu przypadkach, dane te nie posiadały liczby wątków procesora, wyłącznie rdzenie. Spowodowało to powstanie "dziur"w danych. Aby zapobiec wyborowi danych nieistniejących, dane te nie są pokazywane użytkownikowi podczas wyświetlania parametrów.

Nielogiczne dane znalazły się również w kolumnach *resolution_height* i *resolution_width* oryginalnych danych - często występował przypadek, gdy wysokość ekranu była większa niż szerokość. Ten problem łatwo rozwiązano poprzez stworzenie nowej kolumny *resolution* oraz sprawdzenie, czy szerokość na pewno jest większa od wysokości - w przeciwnym razie wartości zamieniono, otrzymując standardowe oznaczenie rozdzielczości ekranu monitora.

Dodatkowo, w celu zachowania realizmu, w module main zostały podjęte kroki dotyczące wyświetlania parametrów dla komputerów firmy Apple. Faktem jest, iż firma Apple nie korzysta z rozwiązań producentów z których korzystają inne firmy produkujące laptopy, i vice versa. Z uwagi na to, gdy wybierze się dowolny parametr, który jest natywny do laptopa firmy Apple, zostaną zastosowane specjalne filtry.

2.2.4 Problem z pustymi predyktorami

Oryginalnie, metody odpowiedzialne za predykcję rozpoczęły działanie od wydobywania z bazy danych rekordów, które zgadzały się z podanymi parametrami. Szybko zauważono, że nie jest to optymalne rozwiązanie - narzędzie powinno być w stanie dokonać wyceny produktów nie tylko na podstawie istniejących danych, a na podstawie podanych parametrów.

Wówczas zaistniał problem z wykorzystaniem regresji liniowej i jej pochodnych - jeżeli poda jej się parametry kategoryczne, które nie występowały w uczeniu, model może podać odbiegające od rzeczywistości ceny. Przykładowo, w jednej z faz testowych otrzymano cenę laptopa Apple rzędu $7,452 * 10^{20}$ PLN.

Z uwagi na to, postanowiono wprowadzić następujące zmiany do funkcji przygotowującej parametry:

1. Gdy w zbiorze danych istnieje minimalnie jeden rekord zawierający dane zgodne z podanymi przez użytkownika parametrami, te dane zostaną wykorzystane do predykcji ceny. W przypadku liczby rekordów większej od jednego, wyciągana jest średnia arytmetyczna cen;

2. Gdy zbiór danych nie zawiera ani jednego rekordu spełniającego kryteria, wówczas parametry są uzupełniane o tzw. dane standardowe oraz ogranicza się predykcję do modeli RandomForestRegressor oraz GradientBoostingRegressor.

RandomForestRegressor oraz GradientBoostingRegressor zostały wybrane z uwagi na fakt, iż nie potrzebują mieć pełnego zestawu predyktorów w celu wykonania predykcji - są w stanie wyznaczyć cenę laptopa na podstawie podanych, niepełnych parametrów. Stąd, w przypadkach nietypowych, ale praktycznie możliwych (np. nowoczesny komputer z pamięcią masową rozmiaru 64GB), wykorzytano te dwa modele.

Dane standardowe, w tym przypadku, oznaczają wartości parametrów, które w swojej kategorii występują co najmniej dwukrotnie częściej niż drugi najczęstrzy parametr. W przypadku ceny jest to średnia wszystkich cen. Wartości te zostały wyznaczone za pomocą histogramów oraz funkcji zliczających wystąpienia w pythonie. Są to:

- System operacyjny - Windows 11 OS;
- Gwarancja - 1 rok;
- Rozmiar pamięci masowej - 512 GB;
- Typ pamięci masowej - SSD;
- Typ pamięci RAM - DDR4;
- Rozdzielczość - 1920x1080;
- Cena - 3860 PLN.

Oczywiście, te dane są ściśle związane z posiadanym zbiorem danych - w przypadku zmiany zbioru i chęci kontynuowania tego rozwiązania, należy sprawdzić jakie parametry odpowiadają kryteriom danych standardowych.

3 Testy narzędzia

W celu zbadania dokładności narzędzia przeprowadzono testy polegające na wycenie laptopów według parametrów podanych na stronie x-kom.

Podzielono testy na dwie części - jedną dla laptopów, których dane znajdują się w zbiorze danych oraz drugą, dla których się nie znajdują.

Poniżej przedstawiono wyniki testów:

Tabela 5: Test narzędzia wyceniającego laptopy znajdujące się w zbiorze danych

L.P.	Laptop	Indeks	Cena ze zbioru [PLN]	Cena estymowana [PLN]	Błąd [PLN]
1	Dell Inspiron 5420 Laptop	97	2555.71	2531.71	60
2	Apple MacBook Air M2	466	11570.38	10749.09	605
3	Asus Vivobook 16X 2023	164	3761.46	3515.37	330
4	Pavilion 15-eg3079TU	160	3520.31	3477.42	100
5	MSI Modern 14	122	2169.87	2373.78	226

Jak widać, dla danych istniejących model potrafi dokładnie wycenić laptopy.

Tabela 6: Test narzędzia wyceniającego laptopy nieznajdujące się w zbiorze danych

L.P.	Laptop	Strona	Cena rynkowa [PLN]	Cena estymowana [PLN]	Błąd [PLN]
1	Lenovo Yoga Slim	Link	3499.00	3449.32	410
2	Apple MacBook Air M2	Link	4349.00	5443.52	1583
3	Dell Inspiron 3520	Link	2399.00	2880.21	979
4	HP 17	Link	4149.00	4550.10	690
5	Asus Vivobook 15	Link	2199.0	3037.47	822

Powyższe wyniki udowadniają, że jest miejsce na udoskonalenie rozwiązania. W przypadku parametrów, które co najmniej raz występują w bazie danych, błąd estymacji jest relatywnie niewielki w porównaniu do ceny (rzędu 5-8%). Natomiast w przypadku laptopów, których dane nie znajdują się w bazie, estymat relatywnie dobrze wskazuje ceny rynkowe, jednak estymowany błąd odstaje od założonego. Jest to spowodowane danymi standardowymi - użytkonik nie podaje parametru estymowanej ceny, więc gdy w bazie danych nie ma rekordów spełniających kryteria, do wyliczenia błędu RMS narzędzie bierze pod uwagę cenę standardową, tj. 3860 PLN. Łatwo to dowieść empirycznie, odejmując lub dodając błąd do ceny estymowanej. Niewątpliwie, program posiada miejsce do usprawnienia.

4 Proponowany kierunek rozwoju narzędzia

W celu rozwoju narzędzia proponuje się:

1. Stworzyć system web scrapping'u - dane, na których operuje program, są suchymi, zebranymi i modyfikowanymi danymi. W celu utrzymania aktualności wyceny oraz parametrów laptopów, należy stworzyć system aktualizacji danych oparty na web scrapping'u;

2. Udoskonalić obsługę pustych predyktorów - aktualne rozwiązanie, o ile pozornie wydaje się skuteczne, potrafi być zawodne. Celem rozwoju aplikacji należy zaimplementować doskonalsze rozwiązania obsługi predykcji z brakiem pewnych parametrów.

5 Podsumowanie

Program NDWL jest w stanie oszacować cenę laptopów o podanych przez użytkownika parametrach. Oszacowanie, oparte na modelach regresji, nie jest idealne - głównymi czynnikami powodującymi rozbieżność wyceny od cen rzeczywistych są obsługa przypadków pustych predyktorów oraz ilość dostępnych danych. Implementując te usprawnienia, których podczas tworzenia projektu nie zdążono wykonać, projekt może stać się jeszcze lepszym rozwiązaniem dla użytkowników.