Raport – Narzędzie do wyceny diamentów

Jan Szopa, 07.06.2024

1. Wprowadzenie

1.1. Cel

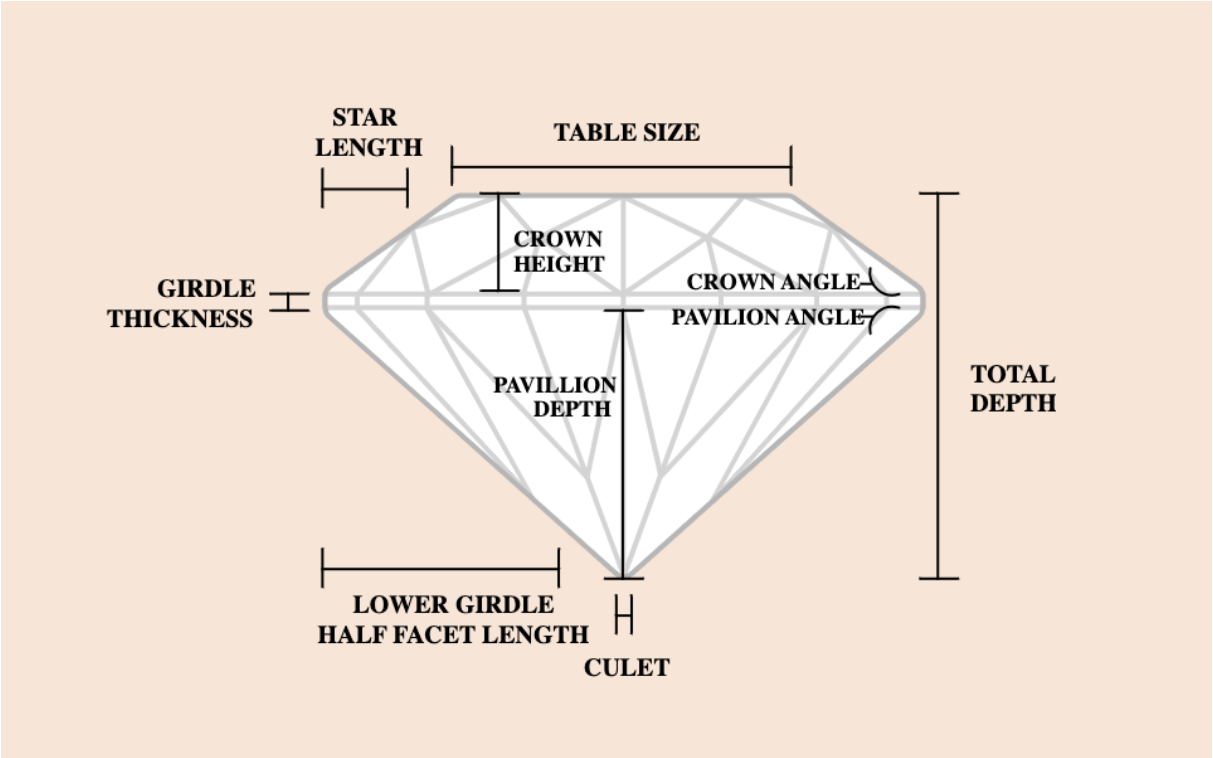
Celem projektu jest wytworzenie narzędzia/aplikacji prognozującej ceny diamentów na podstawie ich cech, wymiarów (opisanych w punkcie 1.2). Używam do tego modeli regresji dostępnych w bibliotekach Python'a, czyli uczenia maszynowego, konkretnie uczenia nadzorowanego.

1.2. Zbiór danych

Wykorzystany przeze mnie zbiór danych Diamonds.csv zawiera informację o 53,940 diamentach. Obiekty w tym zbiorzę są opisane przez 10 atrybutów:

- **price** - cena w dolarach amerykańskich (326-18 823 USD)
- **carat (0.2--5.01)** - Karat to fizyczna masa diamentu mierzona w karatach metrycznych. Jeden karat jest równy się 0.2 grama
- **cut (Fair, Good, Very Good, Premium, Ideal)** – miara szlifu diamentu, im bardziej precyzyjnie oszlifowany jest diament, tym bardziej jest wartościowy
- **color, od J (najgorszy) do D (najlepszy)** - kolor diamentów, diamenty występują w wielu odcieniach, w zakresie od bezbarwnego do jasnożółtego lub jasnobrązowego. Najrzadsze oraz najcenniejsze są diamenty bezbarwne
- **clarity (I1 (najgorszy), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (najlepszy))** – miara przejrzystości diamentu, mogą one mieć cechy wewnętrzne zwane inkluzjami lub cechy zewnętrzne zwane skazami. Diamenty bez inkluzji i skaz są rzadkie i cenne
- **wymiary**
    - **x** - długość w milimetrach (0--10.74)
    - **y** – szerokość w mm (0--58.9)
    - **z** - głębokość w mm (0--31.8)
- **depth (43--79)** - całkowity procent głębokości równy depth = z / średnia (x, y) = 2 * z / (x + y) Głębokość diamentu to jego wysokość (w milimetrach) mierzona od kuletu (dolnego wierzchołka) do parametru table (płaski, górna powierzchnia).
- **table (43--95)** - szerokość wierzchołka diamentu w stosunku do najszerszego punktu. Table diamentu odnosi się do płaskiej fasety diamentu widocznej, gdy kamień jest skierowany ku górze.

Zilustrowanie:



2. Budowa Modelu regresji

2.1. Eksploracja danych

2.1.1. Oczyszczenie danych

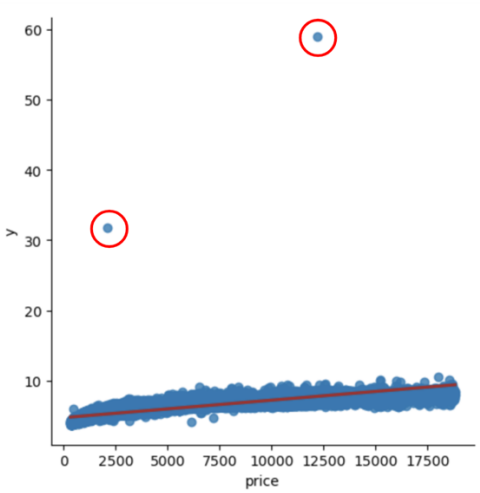
W pierwszej kolejności oczyściłem dane, w badanym zbiorze nie wystąpiły braki tzn. nieuzupełnione wartości. Znalazłem jednak błędne dane w postaci 20 obiektów, w których co najmniej jeden z wymiarów (x, y, z) był równy zero. Liczba tych błędnych danych była jednak znikoma w porównaniu do całego zbioru, więc je z niego usunąłem.

	carat	depth	table	price	x	y	z
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000
mean	0.797940	61.749405	57.457184	3932.799722	5.731157	5.734526	3.538734
std	0.474011	1.432621	2.234491	3989.439738	1.121761	1.142135	0.705699
min	0.200000	43.000000	43.000000	326.000000	0.000000	0.000000	0.000000
25%	0.400000	61.000000	56.000000	950.000000	4.710000	4.720000	2.910000
50%	0.700000	61.800000	57.000000	2401.000000	5.700000	5.710000	3.530000
75%	1.040000	62.500000	59.000000	5324.250000	6.540000	6.540000	4.040000
max	5.010000	79.000000	95.000000	18823.000000	10.740000	58.900000	31.800000

2.1.2. Wartości odstające

Obserwując wykresy punktowe zależności par atrybutów od siebie(pairplot) zauważyłem, że dla atrybutów x, y, z, depth oraz table ładnie kształtują nam się zależności liniowe z innymi atrybutami, jednak mają pojedyncze, ale bardzo widoczne wartości odstające od reszty. W celu dokładniejszego wytrenowania modelu, usunąłem je ze zbioru.

Przykład, wykres punktowy atrybutu y w zależności od ceny, z naniesioną linią regresji:



2.1.3. Ocena korelacji

Z ceną diamentu bardzo silnie skorelowane są atrybuty x, y, z oraz carat (około 0.9 czyli praktycznie zależne liniowo), więc obowiązkowo trzeba je wykorzystać przy uczeniu modelu. Wymienione atrybuty wykazują również silną korelację między sobą, więc można by rozważyć odstawienie niektórych do procesu uczenia. Atrybuty depth oraz table wykazały słabą korelację z ceną. Końcowo, do uczenia modelu wykorzystałem wszystkie atrybuty, ponieważ dla takiego wariantu otrzymałem najlepsze wyniki.

2.1.4. Zakodowanie atrybutów kategorycznych

Atrybuty cut, color oraz clarity są typu kategorycznego, więc do uczenia modelu regresji trzeba je przekształcić. Wykonałem to zgodnie z zasadą one-hot encoding.

Przykład dla atrybutu cut, tylko jedna z kolumn, dla jednego obiektu, może mieć wartość true/1:

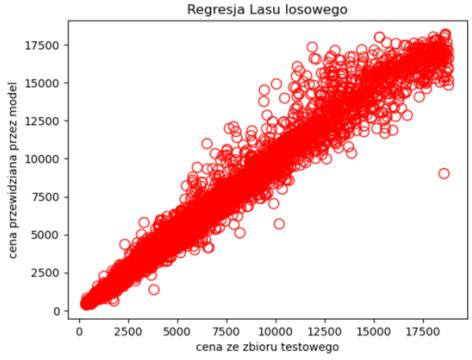
Kolumny powstałe z atrybutu 'cut':

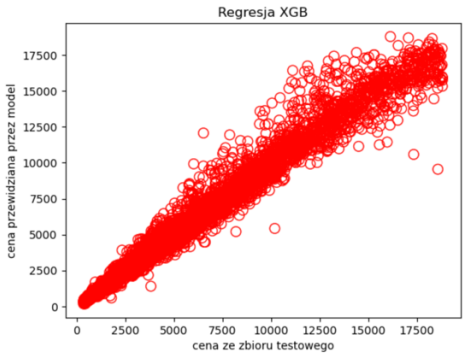
	cut_Fair	cut_Good	cut_Ideal	cut_Premium	cut_Very Good
0	False	False	True	False	False
1	False	False	False	True	False
2	False	True	False	False	False
3	False	False	False	True	False
4	False	True	False	False	False

2.2. Dobór modelu

Z uwagi na duże różnice w skali między wartościami różnych atrybutów, na co wrażliwy jest model regresji, przed uczeniem należało przeprowadzić standaryzację danych. Następnie wytrenowałem modele regresji: Liniowej, Lasso, Ridge, klasyfikatora Adaboost, Lasu losowego, K-sąsiadów, Drzewa decyzyjnego oraz XGB. Najlepsze wyniki, dla miary regresji R2(porównanie ceny ze zbioru testowego z ceną przewidzianą przez model) otrzymałem dla modeli regresji Lasu losowego i XGB, na poziomie 0.98 (prawie idealne).

Ostatecznie wybrałem model regresji XGB ze względu na minimalnie lepszy wynik oraz na minimalnie lepszy wykres ceny testowej od przewidzianej przez model. Oba modele się sprawdzą, różnice między nimi są bardzo małe.





3. Działanie aplikacji

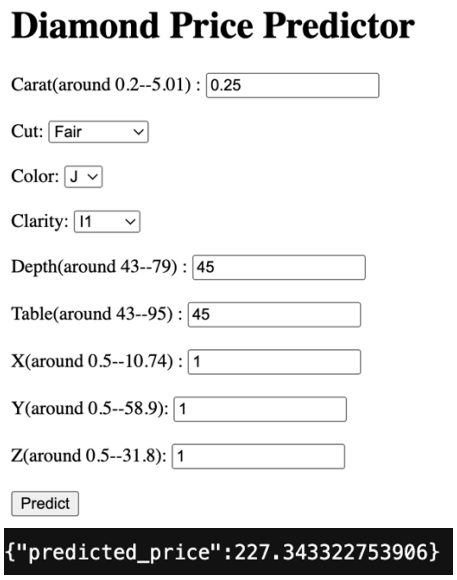
Aplikację prognozującą zbudowałem w oparciu o serwer API, wykorzystując bibliotekę Python'a Flask.

1. Po uruchomieniu aplikacji, na wyjściu otrzymujemy link do formularza html, gdzie wypełniamy informację o diamencie do wyceny. Wartości atrybutów carat, x, y, z, depth, table użytkownik wprowadza ręcznie, natomiast cut, color, clarity z rozwijanych list.
2. Następnie wciskamy przycisk predict, dane wprowadzone w formularzu są wysyłane do serwera za pomocą metody POST
3. Serwer odbiera dane i przetwarza je (analogicznie jak do procesu uczenia):
    1. przekształca na ramkę danych
    2. koduje za pomocą one-hot encoding dla atrybutów cut, color, i clarity
    3. dopełnia brakujące kolumny**
    4. standaryzuje dane
4. Załadowany model przewiduję cenę
5. Przewidywana cena diamentu jest zwracana do użytkownika w formie odpowiedzi JSON
6. Przeglądarka użytkownika wyświetla wynik na stronie internetowej

**gdy wykonamy one-hot encoding dla jednego obiektu, nie otrzymamy tych wszystkich kolumn co przy kodowaniu całego zbioru, więc musimy dodać brakujące, z wartością false/0.

Przykłady użycia:

1. Wartości bliskie dolnym granicom



2. Zwiększmy np. carat i dajmy lepszy color

Diamond Price Predictor

Carat(around 0.2--5.01) : 1

Cut: Fair

Color: H

Clarity: I1

Depth(around 43--79) : 45

Table(around 43--95) : 45

X(around 0.5--10.74) : 1

Y(around 0.5--58.9): 1

Z(around 0.5--31.8): 1

Predict

{"predicted_price":1371.78479003906}

3. Teraz zwiększmy wymiary oraz table i depth

Diamond Price Predictor

Carat(around 0.2--5.01) : 1

Cut: Fair

Color: H

Clarity: I1

Depth(around 43--79) : 61.5

Table(around 43--95) : 55.0

X(around 0.5--10.74) : 6

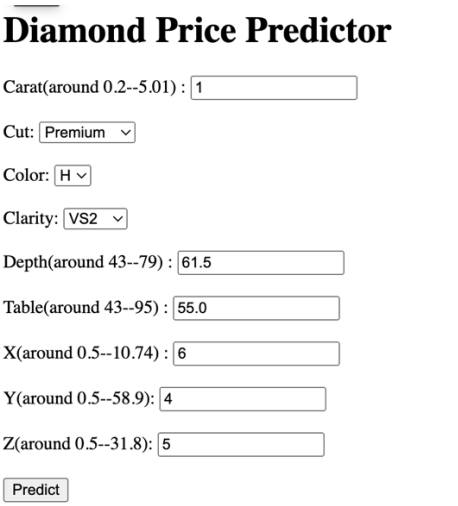
Y(around 0.5--58.9): 4

Z(around 0.5--31.8): 5

Predict

{"predicted_price":2013.3125}

4. Teraz lepszy cut i clarity



{"predicted_price":4884.216796875}

4. Bibliografia

- Strona przykładowego jubilera - https://www.diamondere.com/information/diamond-guide
- Strona z badanym zbiorem danych - https://www.kaggle.com/datasets/shivam2503/diamonds/data
- ChatGPT - https://chat.openai.com/