Analiza/Predykcja cen samochodowych

Opis dokumentu

Ten dokument to dokumentacja analizy i predykcji cen samochodowych. Analiza dotyczy zbioru “sales\_ads\_train.csv” przygotowanego przez PJATK Data Science Club (DSC). W ramach projektu zmagamy się z problemem ściśle regresyjnym polegającym na określeniu najdokładniejszej ceny samochodu przy pomocy 23 unikalnych atrybutów wymienionych w sekcji [*Opis zbioru danych*].

Opis zbioru danych

Zbiór danych dzieli się na 4 pliki .csv:

* Sales\_ads\_train
* Sales\_ads\_test
* Synthetic\_training\_data\_mostlyai\_pl.csv
* synthetic\_training\_data\_sdv\_pl.csv

Każdy ze zbiorów zawiera 25 kolumn:

* **ID** – unikalny identyfikator ogłoszenia
* **Cena** – cena pojazdu (Atrybut decyzyjny, target value)
* **Waluta** – waluta ceny (głównie polski złoty, ale również euro)
* **Stan** – stan pojazdu (nowy lub używany)
* **Marka Pojazdu** – marka pojazdu
* **Model Pojazdu** – model pojazdu
* **Generacja Pojazdu** – generacja pojazdu
* **Wersja Pojazdu** – wersja pojazdu
* **Rok Produkcji** – rok produkcji samochodu
* **Przebieg Km** – przebieg w kilometrach
* **Moc KM** – moc silnika w koniach mechanicznych
* **Pojemność Cm3** – pojemność silnika w centymetrach sześciennych
* **Rodzaj Paliwa** – rodzaj paliwa
* **Emisja CO2** – emisja CO₂ w g/km
* **Napęd** – rodzaj napędu
* **Skrzynia Biegów** – typ skrzyni biegów
* **Typ Nadwozia** – typ nadwozia
* **Liczba Drzwi** – liczba drzwi
* **Kolor** – kolor nadwozia
* **Kraj Pochodzenia** – kraj pochodzenia pojazdu
* **Pierwszy Właściciel** – czy właściciel jest pierwszym właścicielem
* **Data Pierwszej Rejestracji** – wskazuje datę, kiedy samochód został po raz pierwszy zarejestrowany w urzędzie – czyli moment, w którym pojazd rozpoczął swoją eksploatację
* **Data Publikacji Oferty** – data publikacji ogłoszenia
* **Lokalizacja Oferty** – lokalizacja podana przez sprzedającego
* **Wyposażenie** – lista wyposażenia pojazdu (np. ABS, poduszki powietrzne, czujniki parkowania itp.)

Analiza wstępna

W przypadku problemu związanym z regresją, czyli próby estymacji wartości liczbowej z dziedziny liczb rzeczywistych, musimy zwrócić uwagę na pewne sytuacje/pułapki kryjące się w zbiorze danych. W zbiorze *sales\_ads\_train.csv* jest ogromna ilość danych (135k rekordów po 25 kolumn każdy ~ 500MB). Uwzględnianie tych wszystkich danych w uczeniu modelu lub też wcześniej, w samym preprocessingu i czyszczeniu danych mogłoby się okazać kosztowne. Pierwsza pytanie, warte zapamiętania, jest:

* **Czy nie lepiej wziąć pod uwagę pewną mniejszą próbkę tego zbioru danych?**

Następnie rozważmy problem pustych danych. W ich przypadku widzę kilka opcji:

1. **Usunąć wiersze zawierające wartości puste.**
2. **Usunąć kolumny zawierające wartości puste.**
3. **Uzupełnić wartości puste, uwzględniając resztę wartości w danej kolumnie.**
4. **Oznaczyć wartości puste jako “*nieznane*” (Flagging).**

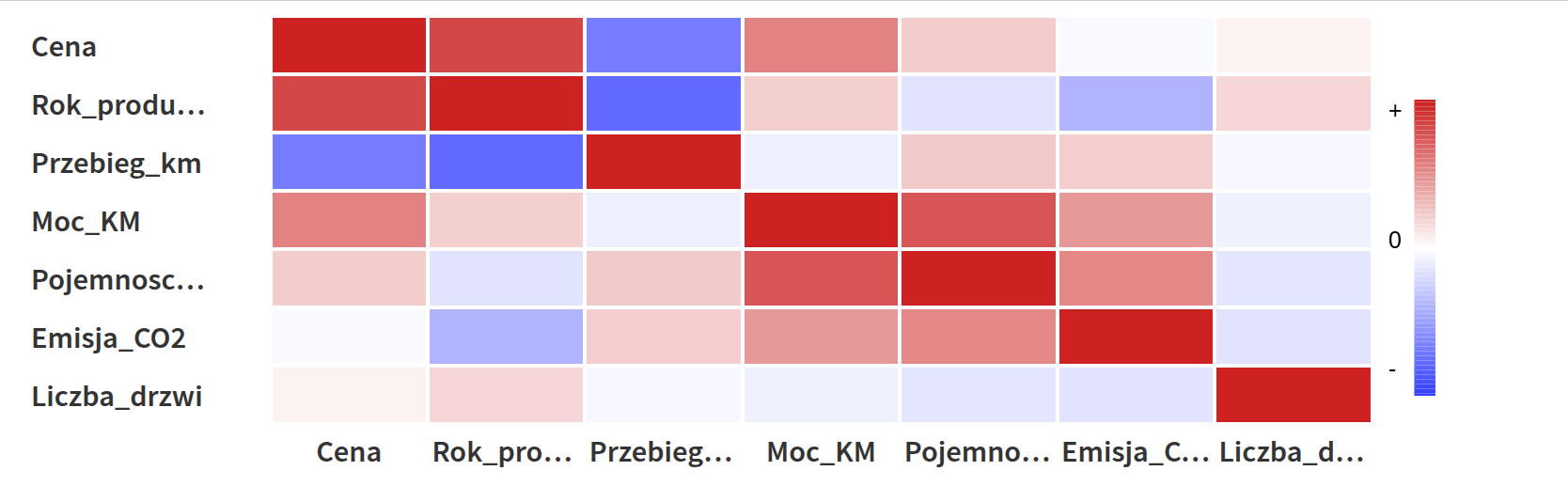
Pytanie jest która opcja z powyższych jest najlepsza? - Pomyślmy:

1. Usuwając wiersze faktycznie istnieje możliwość pozbycia się niechcianych wartości pustych i tym samym zmniejszenia zbioru danych do pewnej próbki tak jak zastanawialiśmy się w poprzednim pytaniu. Ten pomysł wydaje się być w porządku, jednak należy mieć na uwadzę, że istnieje możliwość pozbycia się zbyt dużej ilości rekordów niż byśmy chcieli.
2. Ten pomysł jest chyba najgorszy, ponieważ każda kolumna w analizowanym zbiorze zawiera wartości puste. W ten sposób pozbyliśmy się wszystkich (nie)zbędnych danych.
3. Okej to podejście jest fajne i dla nas częściowo wystarczające. Moglibyśmy uśrednić wartości ciągłe i uznać liczbę wyjściową za potencjalnego kandydata do uzupełnienia wartości brakujących. W naszym przypadku zadziała to na części kolumn, więc warto rozważyć tą opcję.
4. Wydaje się, że alternatywą z poprzedniego punktu dla danych kategorycznych możemy dodać jedną nową flagę która oznacza *“brak informacji”.* W ten sposób później uczony model będzie miał szansę to uwzględnić.
5. Istnieje też możliwość stworzenia algorytmu, który będzie w stanie uzupełnić modele samochodu i marki na podstawie odpowiadającej wartości. Można byłoby zrobić zbiór mapujący te unikalne wartości i uzupełnić brakujące dane na oryginalnym zbiorze.

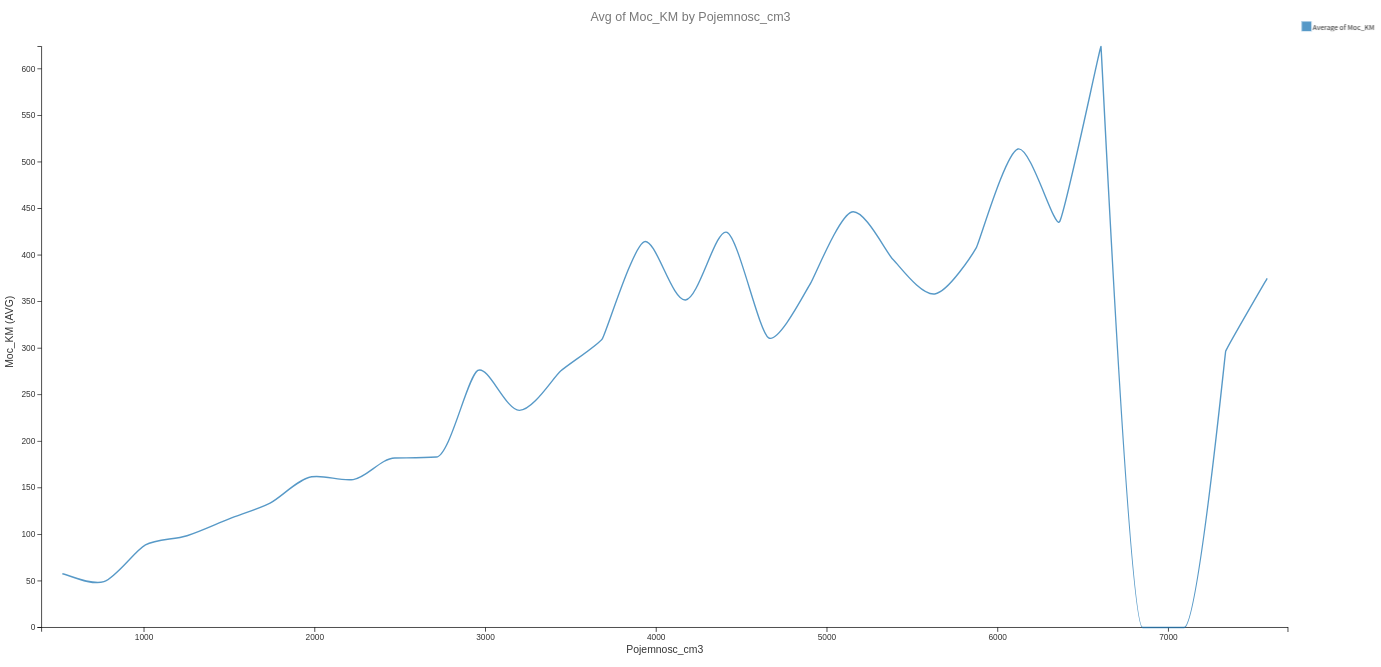
Analiza korelacji

W tej sekcji zajmiemy się analizą korelacji, czyli zależnością atrybutów między sobą. Ogólnie rzecz biorąc, w naszej kategorii problemowej, dobrze będzie szukać takich zmiennych, które będą blisko skorelowane z atrybutem decyzyjnym (Cena). Spójrzmy na to w przypadku, w którym jakiś atrybut rośnie tak samo jak atrybut decyzyjny (w najlepszym przypadku jest tą samą funkcją), wtedy w zasadzie wystarcza nam jeden atrybut, aby bardzo dokładnie wyznaczać wartość poszukiwaną, minimalizując miary (np.: RMSE). Jednak oczywiście nigdy się tak nie zdarza i tak samo jest w naszym przypadku. Celem znalezienia stopnia powiązania atrybutów można skorzystać z macierzy korelacji, która jednoznacznie zwraca współczynniki korelacji dla wszystkich atrybutów.

Należy jednak pamiętać, aby unikać atrybutów “nie decyzyjnych” blisko skorelowanych. Jest to spowodowane wystąpieniem problemu współliniowości. Może być wtedy ciężko określić indywidualny wpływ takich zmiennych na atrybut decyzyjny. Również można spojrzeć na to pod kątem, że po co nam dwa atrybuty zależne blisko skorelowane? W końcu oba niosą prawie tą samą informację i nie dodają nowej wartości.



Na powyższej macierzy znajdują się poziomy korelacji między atrybutami reprezentowanymi przez liczby rzeczywiste (im kolor bardziej czerwony, tym bardziej atrybuty skorelowane). Należy nie zwracać uwagę na diagonalną, ponieważ każdy atrybut jest idealnie skorelowany z samym sobą. Natomiast możemy zauważyć znaczną korelację między *Cena*, a *Rok Produkcji* co w naszym przypadku jest bardzo wartościową informacją, ponieważ to by oznaczało, że rok produkcji ma wysoki wpływ na cenę, której poszukujemy. Dodatkowo warto zwrócić uwagę na powiązanie atrybutów *Moc* i *Pojemność.* Ponieważ są to dwa atrybuty nie decyzyjne, które są dość wysoko skorelowane co oznacza, że niosą podobną informację. Można więc rozważyć usunięcie jednego z nich.



Wykres liniowy pokazujący zależność atrybutów *Moc* i *Pojemność.*

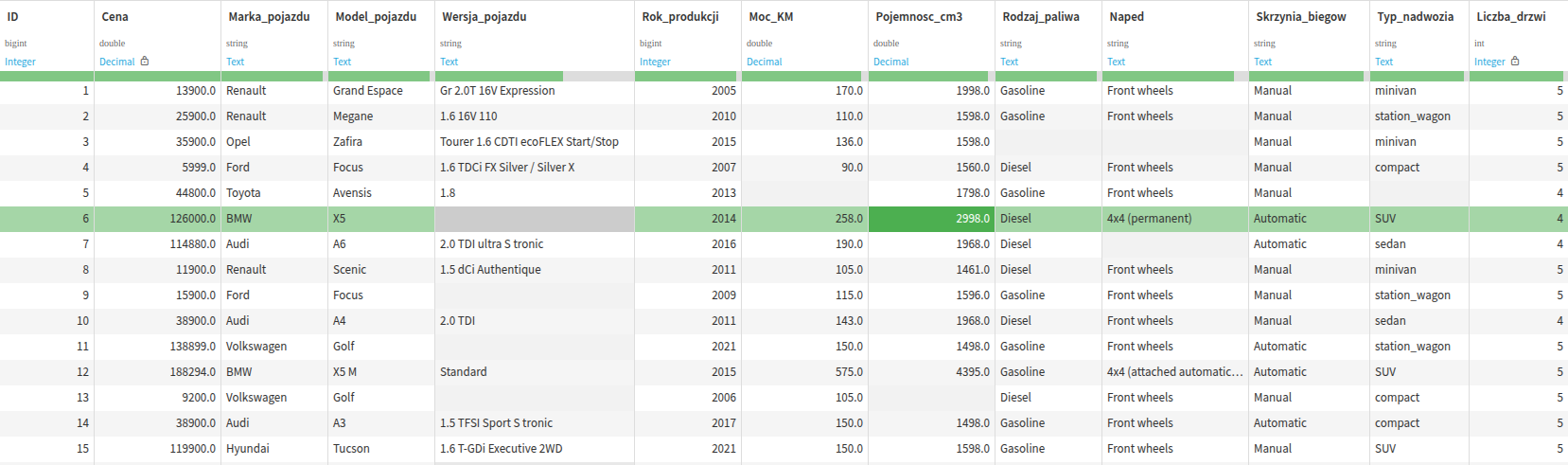
Jak pozbyć się brakujących wartości?

Jak mówiliśmy wcześniej, można to zrobić na kilka sposobów:

* Usunięcie rekordów z wartościami brakującymi,
* Usunięcie kolumn z dużą ilością wartości brakujących,
* Dodanie nowej flagi określającej wartość nieznaną (dane dyskretne),
* Uzupełnienie wartości brakujących średnią lub modą (dane ciągłe)

Powyższe sposoby rozwiązują problem, jednak pojawia się znaczna ilość kolejnych. Po usunięciu rekordów możemy otrzymać niewystarczającą liczbę danych do trenowania (w naszym przypadku ze 135\_000 rekordów zostało niecałe 4\_000). Usuwając kolumny potencjalnie pozbywamy się ważnej informacji. Dodawanie flagi i uzupełnianie wartości są bardziej wyrafinowanymi pomysłami, jednak w przypadku dużego zbioru najpewniej okażą się błędne i będą generować szum.

Na szczęście, jak wcześniej wspomnieliśmy, wpadliśmy na pomysł uzupełnienia tych danych algorytmicznie celem uzyskania wartości mniej lub bardziej bliskich prawdy. Do realizacji napisaliśmy algorytm automatyzujący uzupełnianie wartości brakujących w oparciu o Random Forest. Zdajemy sobie sprawę, że to też nie jest idealne rozwiązanie (w końcu to też działa w oparciu o statystykę), jednak tutaj wydaje się nam, że osiągniemy lepsze wyniki. Dlaczego? Dlatego, że model Random Forest będzie trenowany na zbiorze 60\_000 uzupełnionych rekordów i później będzie przewidywał wartości wybranej kolumny docelowej, które będą bardziej zbliżone do realnej wartości niż flaga “unknown” lub średnia. Jest to kosztowna operacja, aby uzupełnić cały zbiór o 135\_000 rekordów ale warta uwagi i przetestowania. Do predykcji tych wartości brakujących używamy też oryginalnego atrybutu decyzyjnego Cena, która też ma wpływ zwrotny na resztę tych atrybutów.



Powyżej zamieściliśmy zdjęcie zbioru, na którym chcieliśmy uzupełnić wartości (12 wybranych kolumn + kolumna ID do łączenia odpowiednich wartości).

Wyniki pozbycia się wartości pustych

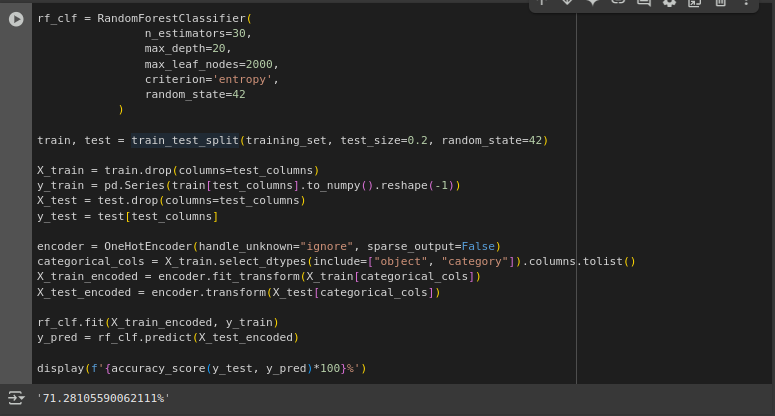
Jak wcześniej wspomnieliśmy, usuwanie rekordów na oryginalnym zbiorze danych przyniosło dość nieoczekiwane rezultaty. Ze 135\_000 rekordów pozostało nam niespełna 4\_000, co jednoznacznie mówi nam, że zgubiliśmy sporo danych.

Po zastosowaniu naszego podejścia do uzupełnienia wartości pustych przy użyciu Random Forest zauważyliśmy, że:

1. Nie zostały uzupełnione wszystkie wartości (ponieważ w pewnym momencie brakowało rekordów, które miałyby tylko wartości null dla atrybutów docelowych, przez co niemożliwe było generowanie nowych zbiorów predykcyjnych)
2. Usuwając pozostałe rekordy z wartościami pustymi zostaliśmy ze 115\_000 rekordów treningowych



Jak widać algorytm znacząco zmniejszył utratę informacji, pozostaje jedynie pytanie na ile te informacje są wiarygodne? Jako przykład podamy przewidywanie marki pojazdu, której dokładność wyniosła około 70%. Nie jest to, jak mówiliśmy, idealne rozwiązanie ale zdaje się lepsze niż podawanie tych samych wartości (średnia, flaga) dla wszystkich rekordów.



Trenowanie i testy modeli przewidujących ceny samochodów

Zdobyty zbiór treningowy został podzielony na dwa podzbiory - testowy i treningowy. Zamysł był taki, aby sprawdzać poprawność modelu na swoich danych nie wysyłając wyników na platformę kaggle. Jak poszło? Początkowo słabo. Po zaprojektowaniu pierwszego modelu bazowego błąd RMSE wyniósł aż 70\_000, co nie jest dobrym wynikiem. Jednak drugi, alternatywny model bazowy będący RandomForest do zastosowań regresyjnych osiągnął już lepszy wynik na poziomie 50\_000.

Inne rozwiązania?

Po przeprowadzeniu wcześniejszego flaggingu zauważyliśmy, że model RandomForest osiągnął znacznie lepsze wyniki niż wcześniej, co zaowocowało rezultatem na poziomie 26 000 opublikowanym na Kaggle. Jednak naszym celem pozostaje dalsze obniżenie błędu. Warto zastanowić się, jakie działania mogą przyczynić się do osiągnięcia jeszcze lepszych wyników.

### **Potencjalne Kierunki Dalszych Działań**

**1. Scalanie Zbiorów Danych:**Jednym z pomysłów jest połączenie oryginalnego zbioru danych z danymi uzupełnionymi algorytmicznie. Następnie moglibyśmy kontynuować uzupełnianie braków przy użyciu flag, licząc na dalszą poprawę wyników. Trzeba jednak wziąć pod uwagę, że im więcej rekordów zostanie uzupełnionych automatycznie, tym bardziej taki zbiór stanie się podatny na nieścisłości.

**2. Standardowy Pipeline dla Zadania Regresji:**Bardziej konwencjonalnym podejściem, często preferowanym nad automatyczne uzupełnianie danych, jest zastosowanie pełnego pipeline'u, obejmującego następujące etapy:

### **Etapy Przetwarzania Danych**

**1. Przetwarzanie Danych:**

* Konwersja wszystkich cen na jedną walutę oraz usunięcie zbędnej kolumny "Waluta".
* Ekstrakcja cech, takich jak **wiek pojazdu** (różnica między datą publikacji a datą pierwszej rejestracji), co pozwala na usunięcie surowych danych dat.
* Reprezentacja lokalizacji oferty w formie regionów lub nazw miast.
* Przekształcenie list wyposażenia w zmienną liczbową reprezentującą liczbę elementów.
* Kodowanie danych kategorycznych za pomocą **Label Encoding** lub **One-Hot Encoding**.
* Usunięcie kolumny **Emisja CO2**, jeśli nie wnosi wartości predykcyjnej.
* Uzupełnianie brakujących wartości flagami, co dotychczas przynosiło dobre rezultaty.

### **Testowanie Modeli**

Po wstępnym przetworzeniu danych, kolejnym krokiem jest przetestowanie kilku modeli regresyjnych, w tym:

* **Random Forest for Regression**
* **XGBoost**
* **LightGBM**
* **ExtraTree**

### **Ewaluacja Wyników**

Po optymalizacji hiperparametrów, z Optuna framework, kluczowe będzie przeprowadzenie ewaluacji modeli w celu wyboru najlepszego. Wskaźnikiem sukcesu pozostaje najniższa wartość **RMSE**, choć warto rozważyć również inne miary (np. **MAE** lub **R^2**) dla pełniejszej oceny.

Stosując powyższy pipeline, udało się dodatkowo obniżyć wartość RMSE o 1 000, co stanowi istotną poprawę w kontekście jakości predykcji.

[[Colab]](https://colab.research.google.com/drive/1YU9pxfRKHYefdM1E4ahxQM3F9idfvvpO?usp=sharing)