Analiza ofert pracy w IT oraz tworzenie modelu przewidującego zarobki

Łukasz Fabia

20.05.2024

Spis treści

1 Wstęp

Celem badań jest analiza danych dotyczących ofert pracy w IT. W swojej pracy postaram się odpowiedzieć na pytanie, jakie są najbardziej poszukiwane umiejętności w branży IT oraz ile można zarobić znając dane języki, frameworki czy narzędzia. W tym celu stworze model, który będzie przewidywał zarobki w zależności od danych wejściowych, o których później.

2 Dane

Dane pozyskam z serwisu justjoin.it, który zbiera oferty pracy z wielu różnych serwisów, zatem ofert pracy będzie całkiem sporo. Na stronie mamy katergorie, które mogą być przydatne do filtrowania danych, ale są one mało przydatne, ponieważ one raczej są przypisane do ofert, które nawiązują w jakiś sposób do np. JS. katergorie są następujące: JS, PHP, Ruby, Python, Java, Net, Mobile, C, DevOps, Security, Data, Go, Game, Scala. Moje dane będą właśnie z tych podstron. Dodatkowo analizuję zarobki tylko na b2b oraz na umowie o prace (uop), ponieważ są to najbardziej popularne formy zatrudnienia w IT a inne formy takie jak umowa o zlecenie czy umowa o staż pratykcznie nie występują. Do analizy będę również brał pod uwagę lokalizację.

Technologia - język programowania, framework, narzędzie, które jest wymagane w ofercie pracy.

2.1 Model danej

Dane będą zawierały informacje o ofertach pracy, takie jak:

- tytuł oferty
- widełki dla B2B
- widełki dla UOP
- technologie dotyczące umowy
- lokalizacja
- doświadczenie junior, mid, senior
- typ pracy stacjonarnie, hybrydowo, zdalnie

2.2 Obsługa technologii, lokalizacji

Najpierw zdefiniuje sobie słownik klucz, wartość, gdzie klucz to ustandaryzowana technologia, a wartość do synonimy tej technologii.

np. "JavaScript": ["javascript", "js", "node.js", "nodejs", "express.js", "expressjs",],

Dzięki temu będę mógł przekonwertować technologie z oferty pracy na wektor binarny, gdzie 1 oznacza, że technologia jest wymagana, a 0, że nie jest wymagana. Kolejnym krokiem będzie obsługa lokalizacji. W tym przypadku jeśli oferta dot. kilku miast to znaczy, że pojawi się w zbiorze klika ofert z tymi samymi danymi, ale dla różnych miast.

2.3 Pozykiwanie danych

Dane będą pozyskiwane z ww. serwisu, za pomocą narzędzi do web scrappingu w moim przypadku będzie to `Selenium`, ponieważ strona ma dynamicznie ładowany content.

Kroki:

- napisanie skryptu pobierającego linki do ofert pracy z danej kategorii, ponieważ nie chcemy śmiecowych ofert typu Product manager
- napisanie skryptu przetwarzającego linki do ofert pracy, aby pobrać dane z oferty
- przekierowanie wyniku do pliku json.
- normalizacja oraz oczyszczanie danych, kodowanie technologii, do wektora przy pomocy MultiLabelBinarizer z `sklearn`
- kodowanie zmiennych kategorycznych (np. miasta, typ pracy, kontrakty)
- wzięcie średniej zarobków potem usunięcie widełek w zarobkach (np. 10-15k $\implies$ 12.5k)
- rozbicie ofert pracy dla kontraktów oraz miasta np. mamy ofertę dla miasta A i B obie mają podane zarobki na b2b i uop, więc powstaną 4 nowe oferty, czyli A_uop, A_b2b, B_uop, B_b2b.
- usunięcie ofert z wynagrodzniem godzinowym bo zalezą one od ilości przepracowanych godzin
- usunięcie tytułów ofert

Ofert ze stawką godzinową było kilka więc nie wypływają one na wyniki.

3 Wygląd do danych

uwaga przykładowe dane nie zawierają wszystkich kolumn bo jest ich za dużo, wszystkie dane można znaleźć w ../data/jobs.csv

Przykładowe dane:

avg_salary	con_code	loc_code	exp_code	mode_code	AWS	...	android
23000.0	1	50	2	0	1	...	0
21742.5	1	37	2	2	0	...	0
21742.5	1	17	2	2	0	...	0
21742.5	1	50	2	2	0	...	0
21742.5	1	54	2	2	0	...	0

Tabela 1: Klika pierwszych wierszy moich danych.

Uwaga

Skróciłem trochę nazwy, ponieważ musiałbym rozbijać tabele na kilka co byłoby mniej czytelne.

1. con_code - kod kontraktu
2. loc_code - kod lokalizacji
3. exp_code - kod doświadczenia
4. mode_code - kod typ pracy

4 Rozkłady i statystyki

Aktualnie w zbiorze *jobs.csv* znajduje się **5751** ofert pracy, które będą poddane analizie. Wszystkie dane są znormalizowane i gotowe do analizy. Analizę można zacząć od średniej zarobków dla kontraktu B2B oraz UOP.

Widełki dla Juniora:

PLN	B2B	UOP
średnie widełki	9902.18	11057.05
min widełki	5200.00	5500.00
max widełki	16100.00	19000.00

Tabela 2: Średnie zarobki w PLN dla `juniora` w Polsce

Widełki dla Mida:

PLN	B2B	UOP
średnie widełki	18025.50	15210.38
min widełki	3132.50	6000.00
max widełki	32500.00	27500.00

Tabela 3: Średnie zarobki w PLN dla `mida` w Polsce

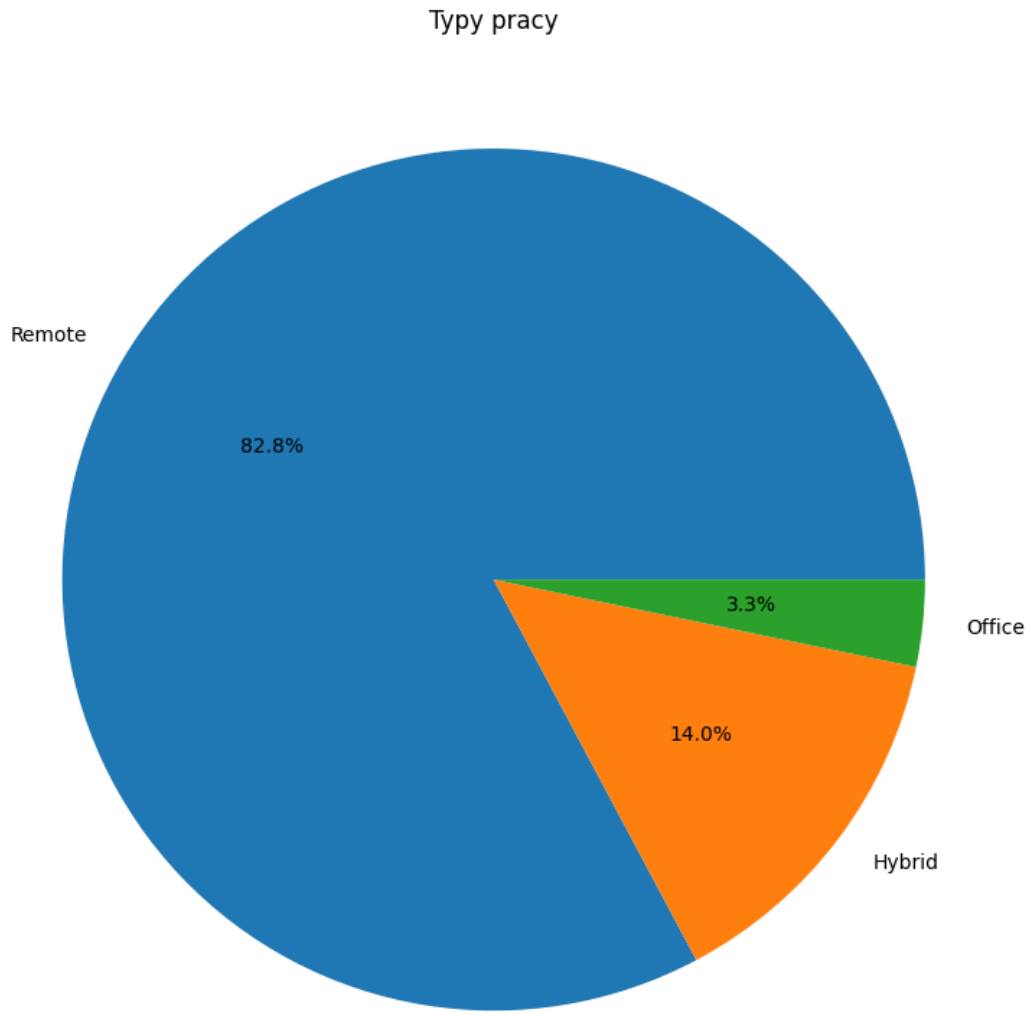
Widełki dla Seniora:

PLN	B2B	UOP
średnie widełki	25700.81	22318.03
min widełki	9791.00	10000.00
max widełki	52500.00	60000.00

Tabela 4: Średnie zarobki w PLN dla `seniora` w Polsce

Generalnie zarobki wyszły całkiem sensownie, ponieważ dla Mida i Seniora mamy większą pensję na B2B niż na UOP, dla juniora jest na odwrót, ale nie mam pojęcia dlaczego tak się dzieje, może na umowę o pracę dostają się bardzo dobrzy programiści albo jest poprostu mało ofert pracy dla juniora na b2b.

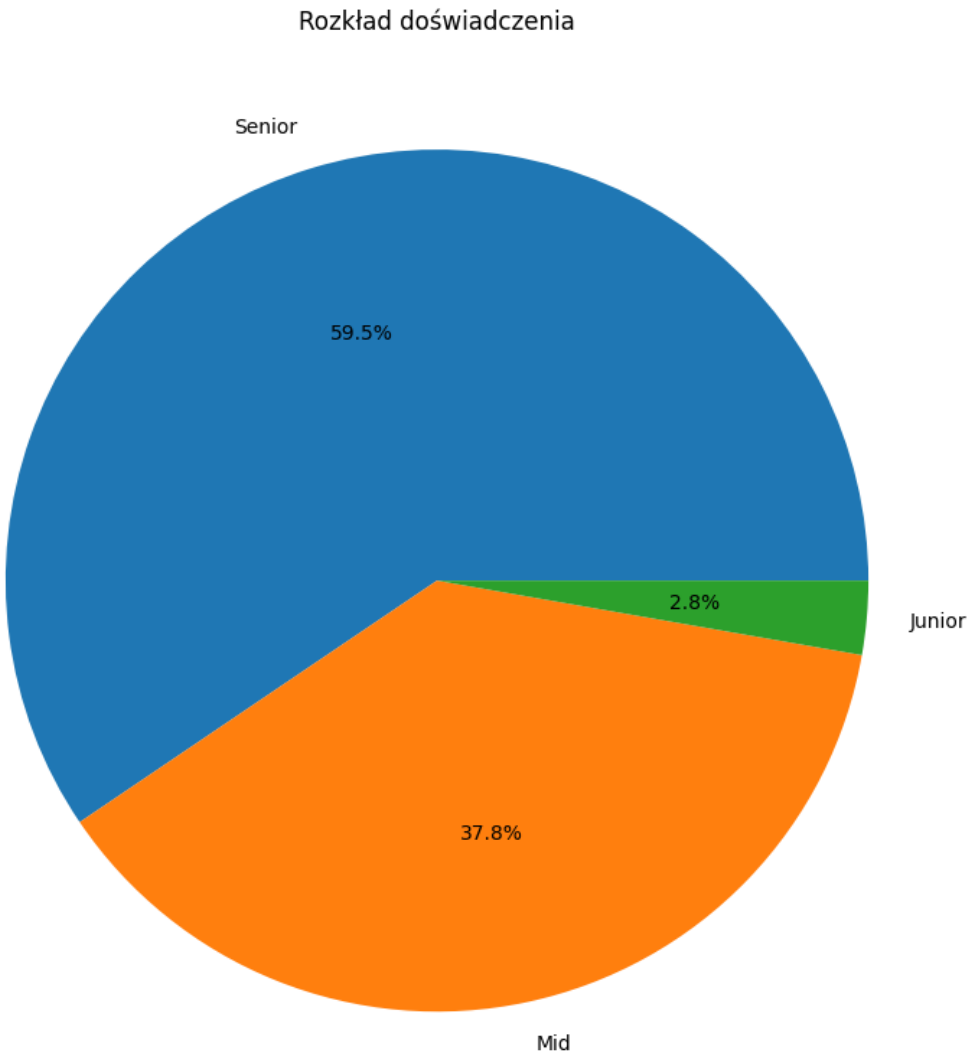
4.1 Jak się pracuje w IT?



Rysunek 1: Rozkład typów pracy

Jak widać najwięcej ofert pracy dotyczy pracy zdalnej.

4.2 Kogo szukają pracodawcy?



Rysunek 2: Rozkład typów pracy

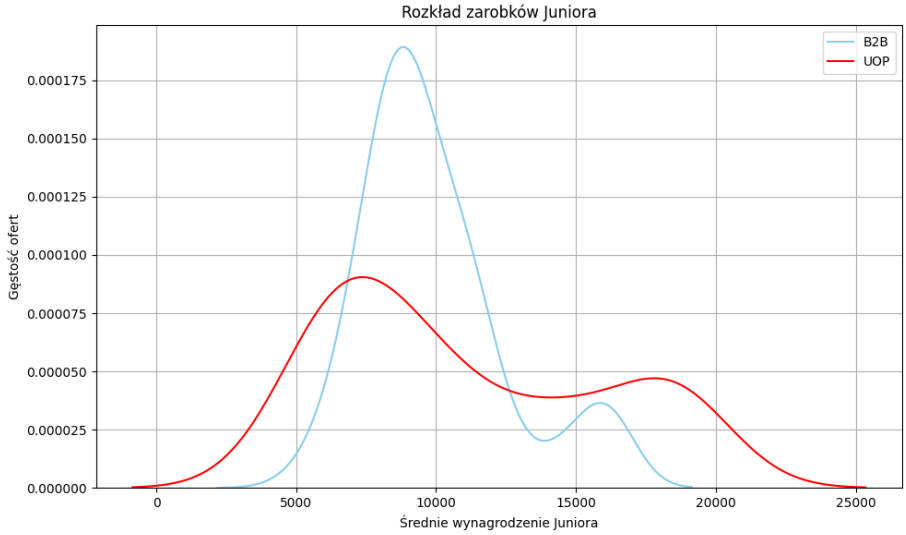
Tak jak można było się spodziewać - najwięcej ofert pracy jest dla seniorów, stąd też wynika dlaczego tak dużo kontraktów dotyczy pracy zdalnej, ponieważ pracodawca ma większe "zaufanie" do seniora niż do juniora. Chociaż warto powiedzieć sytuacja midów jest również dobra. Gorzej jest z ofertami dla młodych programistów. Tutaj liczba ofert wyniosła zaledwie 159, co jest bardzo małą liczbą w porównaniu do innych grup.

Czy to oznacza, że młodzi programiści mają trudniej, a słynne "eldorado" w IT jest tylko dla

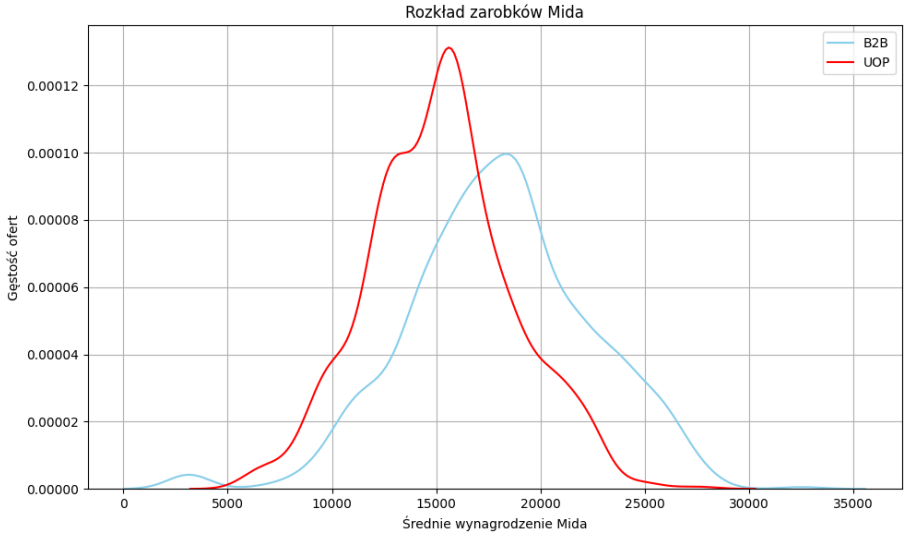
doświadczonych programistów?

Tutaj można powiedzieć, że juniorzy mają trudniej żeby **wejść** do branży, ale zarobki po wejściu są naprawdę atrakcyjne, no, ale tutaj problem może być z wejściem.

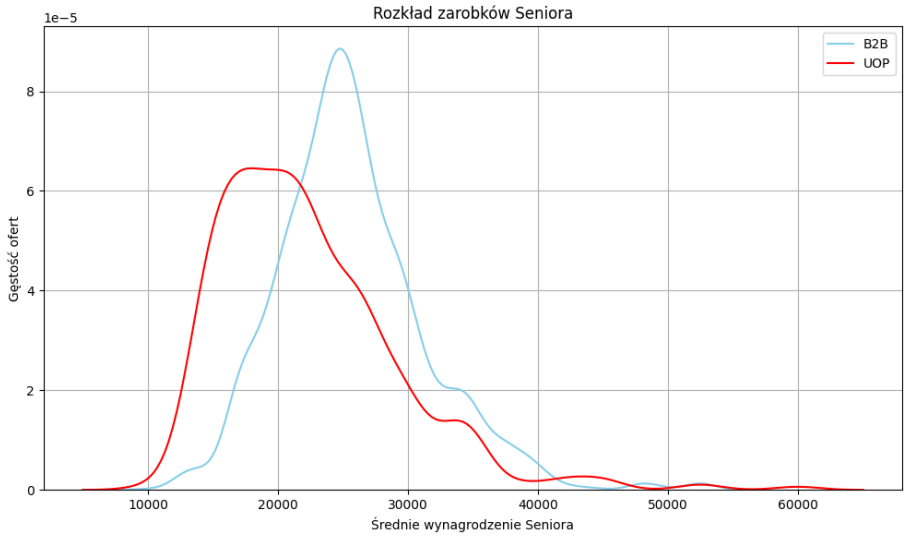
4.3 Jak rozkładają się zarobki?



Rysunek 3: Rozkłady zarobków dla poszczególnych umów dla juniorów

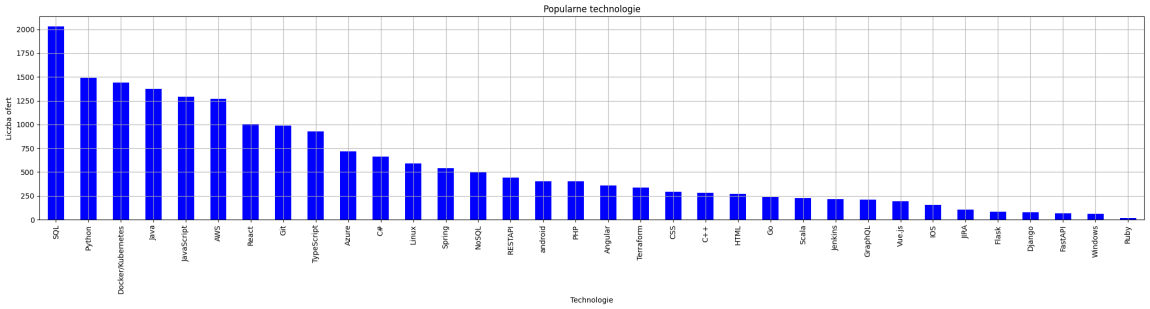


Rysunek 4: Rozkłady zarobków dla poszczególnych umów dla midów



Rysunek 5: Rozkłady zarobków dla poszczególnych umów dla seniorów

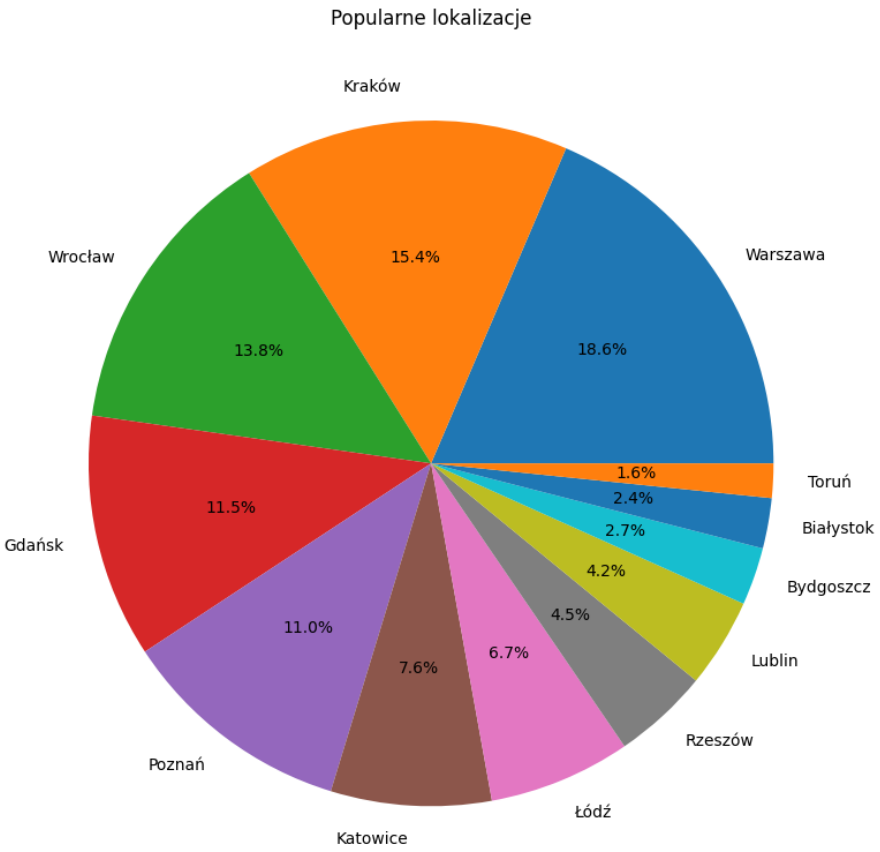
4.4 Jakie technologie są najbardziej poszukiwane?



Rysunek 6: Popularne technologie w ofertach pracy w Polsce

Tutaj moim zdaniem troche zaskoczenie ponieważ bez `SQL` ciężko znaleźć prace w IT, czyli bazy danych to jest podstawa przy rekrutowanu się do pracy. Oczywiście nie mogło zabraknąć `Pythona` oraz `JavaScriptu` jeśli chodzi o języki skryptowe. Co warto zazanczyć narzędzia takie jak `Docker` czy `Kubernetes` również są bardzo popularne i warto je znać. `Java` wygrywa z `C#` a `GNU/Linux` dobrze jest znać.

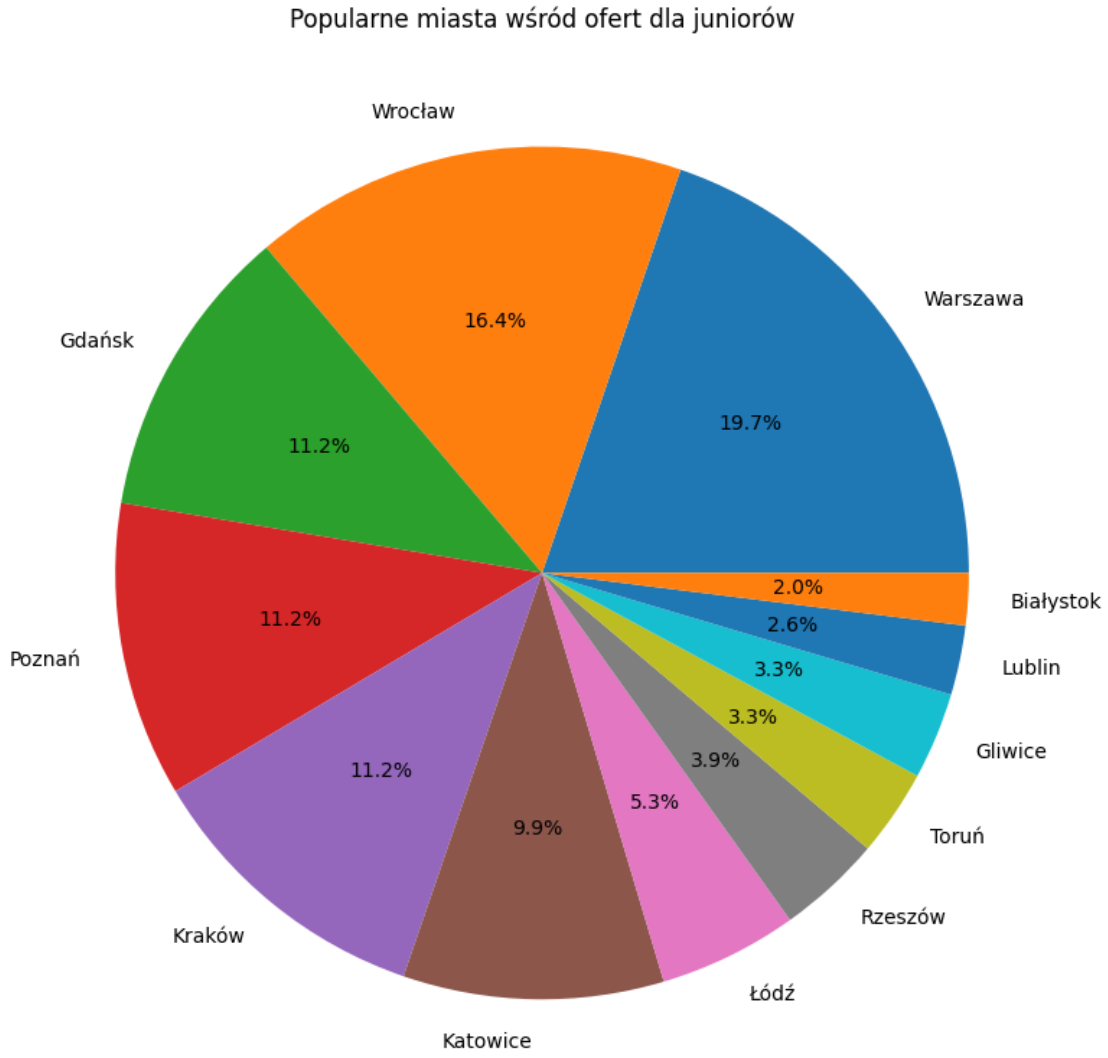
4.5 Gdzie jest największy popyt na programistów?



Rysunek 7: Popularne miasta w ofertach pracy w Polsce

Zestawienie miast jest zgodne z oczekiwaniami, najwięcej ofert pracy jest kolejno w **Warszawie**, **Krakowie** oraz **Wrocławiu**, chociaż **Gdańsk** również pojawiał się w dużej ilości ofert pracy.

4.6 Gdzie poszukiwani są juniorzy?

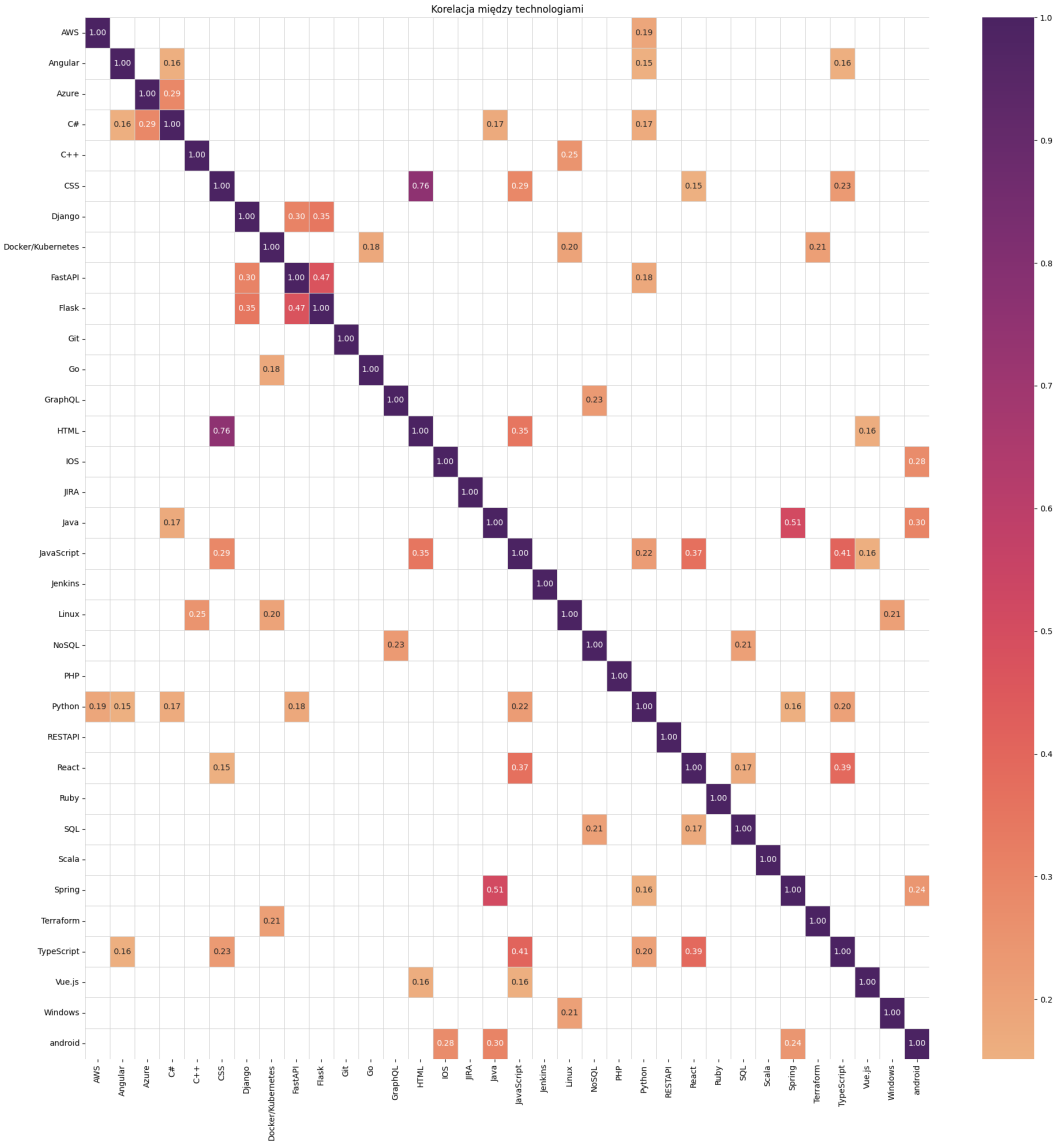


Rysunek 8: Popularne miasta w ofertach dla juniorów

Warszawa jest najbardziej przyjazna dla juniorów, ale warto zauważyć, że wykres nie różni się bardzo od poprzedniego z jednym, *ale* - **Gdańsk** jest na 3 miejscu w zestawieniu dla juniorów, co może być zaskoczeniem.

5 Powiązania między danymi

5.1 Powiązania między technologiami



Rysunek 9: Powiązania między technologiami, zawierająca tylko wartości korelacji większe niż 0.14

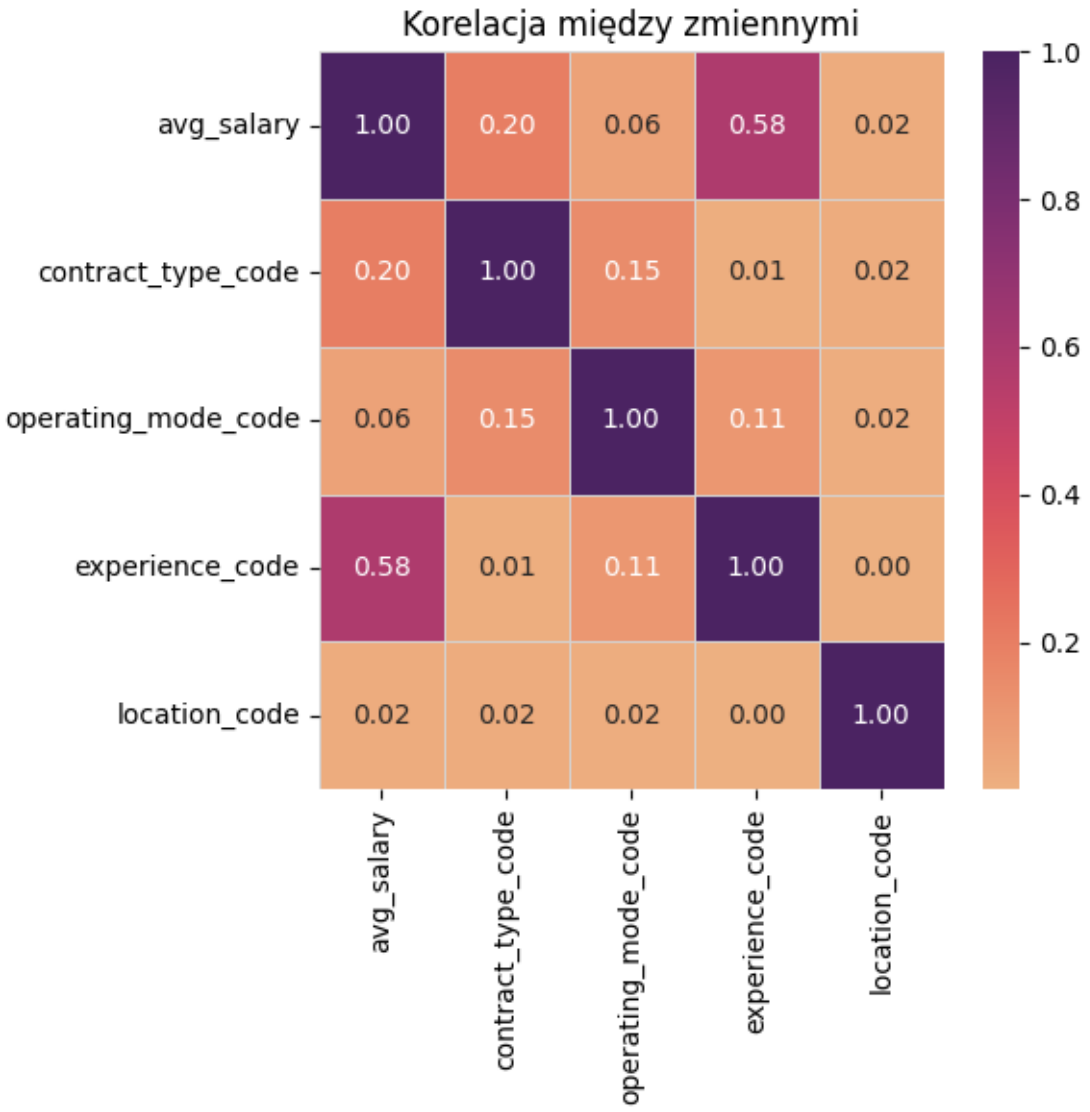
Co można zauważyć?

1. HTML i CSS idą ze prawie w parze - co jest zrozumiałe, bo to podstawy front-endu
2. Przy Javie warto znać Springa

3. React i JS i TS często pojawiają sie razem w ofertach pracy obok HTML i CSS
4. Jak sie uczy Django to warto znać inne frameworki backendowe takie jak Flask czy FastAPI
5. Jak sie idzie w Embedded to warto znać C/C++ oraz Linux
6. Technologie microsoftu takie jak C#, Azure występują razem

To tylko kilka przykładów wymienionych wynikający z obrazka powyżej, ale warto zauważyć, że nie ma tutaj dużo powiązań między technologiami, co może wynikać z tego, że są zbyt różne, aby były powiązane.

5.2 Powiązania między innymi zmiennymi

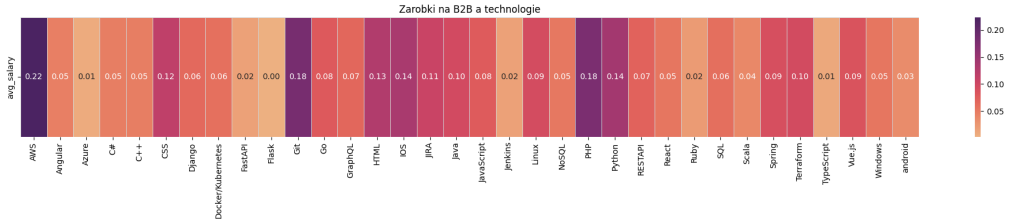


Rysunek 10: Powiązania między innymi zmiennymi

Co można zauważyć?

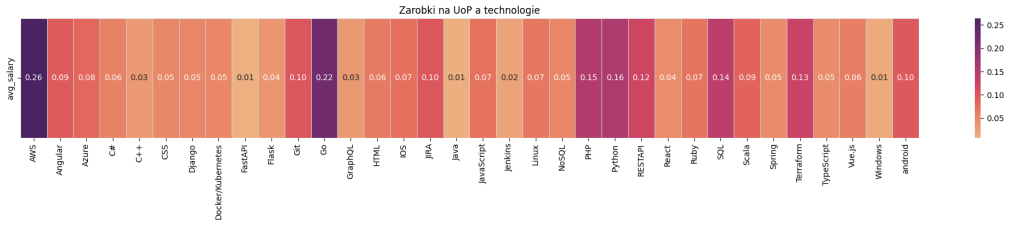
1. Średnia pensja jest mocno powiązana z doświadczeniem
2. Typ kontraktu też wiąże się z doświadczeniem

5.3 Zarobek a technologie



Rysunek 11: Powiązania między zarobkiem b2b a technologiami

AWS, Git, Python, PHP mają wypłw na pensję na b2b.



Rysunek 12: Powiązania między zarobkiem uop a technologiami

PHP, Python, AWS, SQL, Go, Terraform mają wpływ na pensję na uop.

6 Czy da się przewidzieć zarobki, w zależności od mojego tech-stacku?

6.1 Ogólnie o problemie

Oczywiście, że tak, kiedy mamy dane to możemy nauczyć model, który na wejściu dostanie zmienne i przewidzi dla nas zarobki. Dokładniej mówiąc model otrzyma na wejściu dane takie jak:

`Input:`

- Tech-stack
- Lokalizacja
- Typ pracy
- Doświadczenie
- Typ kontraktu (B2B, UoP)

`Output:`

- Zarobki w PLN

6.2 Dobór modeli

Modele które będą wykorzystane w analizie to:

1. Regresja liniowa
    - `LinearRegression`
    - `Ridge`
    - `Lasso`
2. `Decision tree`
3. `Random forest`

Wszyskie modele pochodzą z modułu **`sklearn`** *dostępniej pod tym linkiem*

6.3 Trochę statystyki - metryki

Do oceny modeli wykorzystam metryki takie jak, ale wybierać będę po RMSE, reszta metryk jest w ramach ciekawostki:

- **Root Mean Squared Error** - pierwiastek z średniego błędu kwadratowego
- **R-squared** - współczynnik determinacji R^2
- **Mean Absolute Error** - średni błąd bezwzględny

6.3.1 Pierwiastek z średniego błędu kwadratowego

Root Mean Squared Error (RMSE) - to pierwiastek z MSE, daje nam miarę błędu przewidywań w tych samych jednostkach co dane wejściowe. Można go interpretować jako średnią oczekiwaną różnicę +/- między wartością przewidywaną a rzeczywistą [2].

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{1}$$

6.3.2 Współczynnik determinacji

R-squared (R2) - to miara oceny dopasowania funkcji regresji do danych. Wartość bliska 1 oznacza, że funkcja regresji lepiej dopasowała sie do danych.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}, R^2 \in [0, 1] \tag{2}$$

6.3.3 Średni błąd bezwzględny

Mean Absolute Error (MAE) - to średni bezwzględny błąd między przewidywaniami a rzeczywistymi wartościami. Jest bardziej odporny na wartości odstające niż RMSE (outlinery), ponieważ nie podnosi błędów do kwadratu.

$$\text{MAE} = \frac{\sum_{i=1}^{n}|y_i - x_i|}{n}. \tag{3}$$

6.4 Jak to zrobię?

Moje podejście opiera się na wybraniu modeli regresji liniowej, drzewa decyzyjnego oraz lasu losowego, które będą tuningowane za pomocą `GridSearchCV` w celu znalezienia najlepszych hiperparametrów. Wyniki są dostępne w folderze `../analysis/models_tuning.csv`. Kolejnym krokiem jest przeprowadzenie uczenia modeli i wybranie najlepszego modelu na podstawie metryk. Następnie przewidzę zarobki dla kilku tech-stacków, a na końcu przedstawię wyniki w postaci wizualizacji.

Streszczenie

W następnych rodziałach skupię się na wynikach modeli, a także na wizualizacji wyników, aby nie tworzyć zbyt długiego raportu nie będe analizować słabych modeli tylko skupię się na dwóch najlepszych modelach.

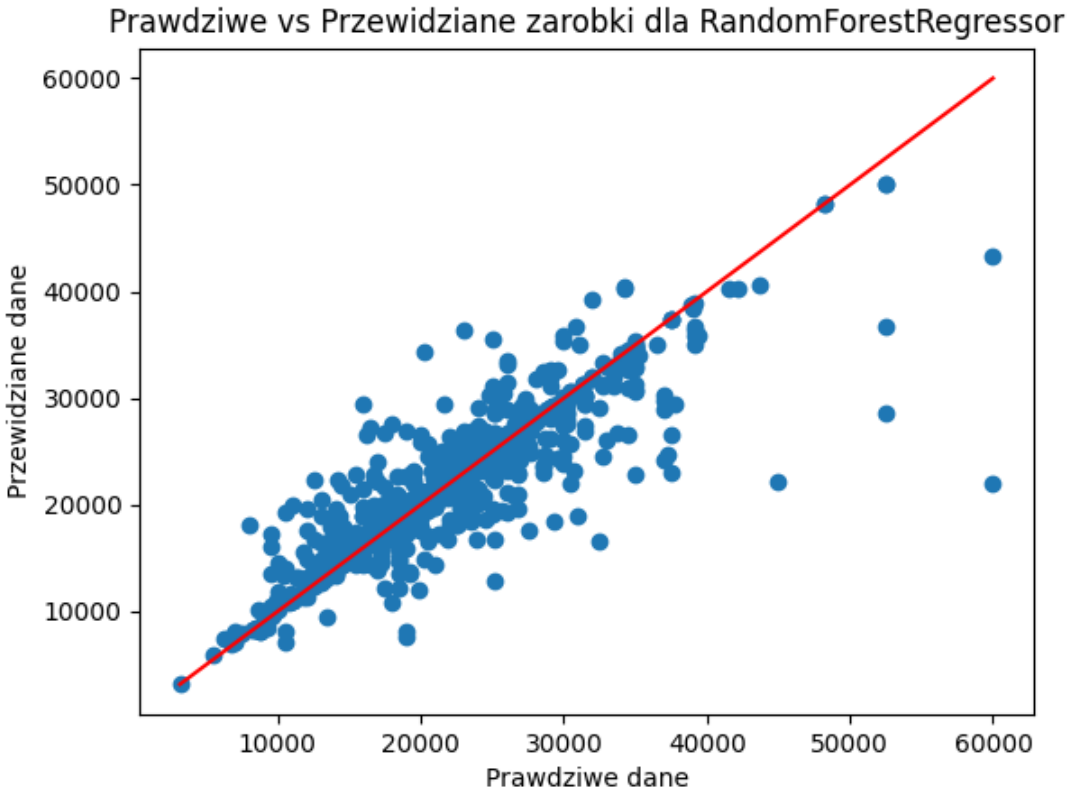
Reszta danych: Wszyskie wyniki z uczenia zostaną zapisane w folderze `../analysis/plots/wyniki/` ew. można też podejrzeć plik z rozwiązaniem problemu w `../analysis/analysis.ipynb`.

Stosowane podziałki to 80:20, czyli 80% danych do uczenia, a 20% do testowania modelu oraz 60:40.

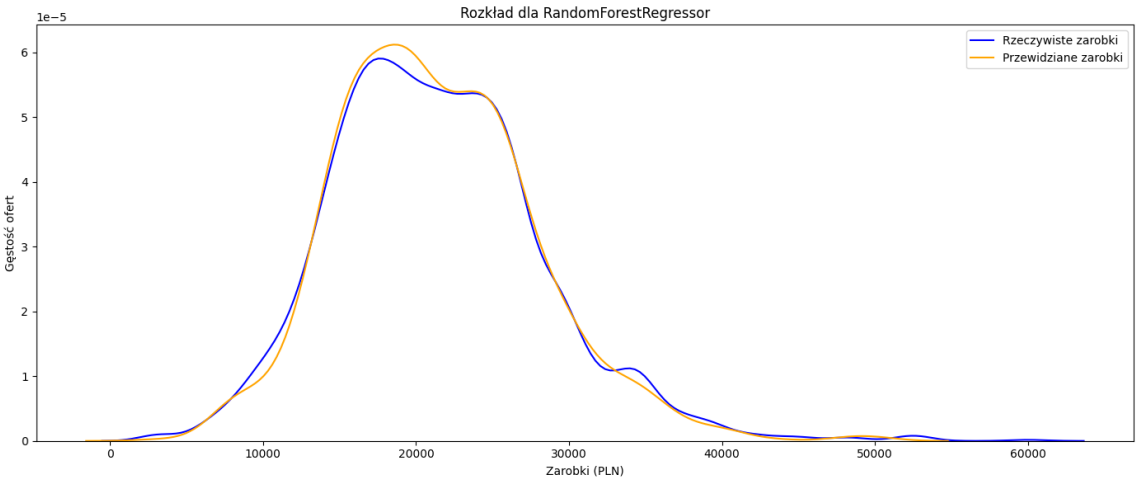
6.4.1 Wyniki dla podziału danych 80:20

Model	Mean Absolute Error	Root Mean Squared Error	R^2 Score
LinearRegression	3639.59	4995.80	0.50
DecisionTreeRegressor	2691.79	4140.79	0.66
RandomForestRegressor	1597.75	3188.99	0.80
Ridge	3637.09	4995.57	0.50
Lasso	3634.55	4995.54	0.50

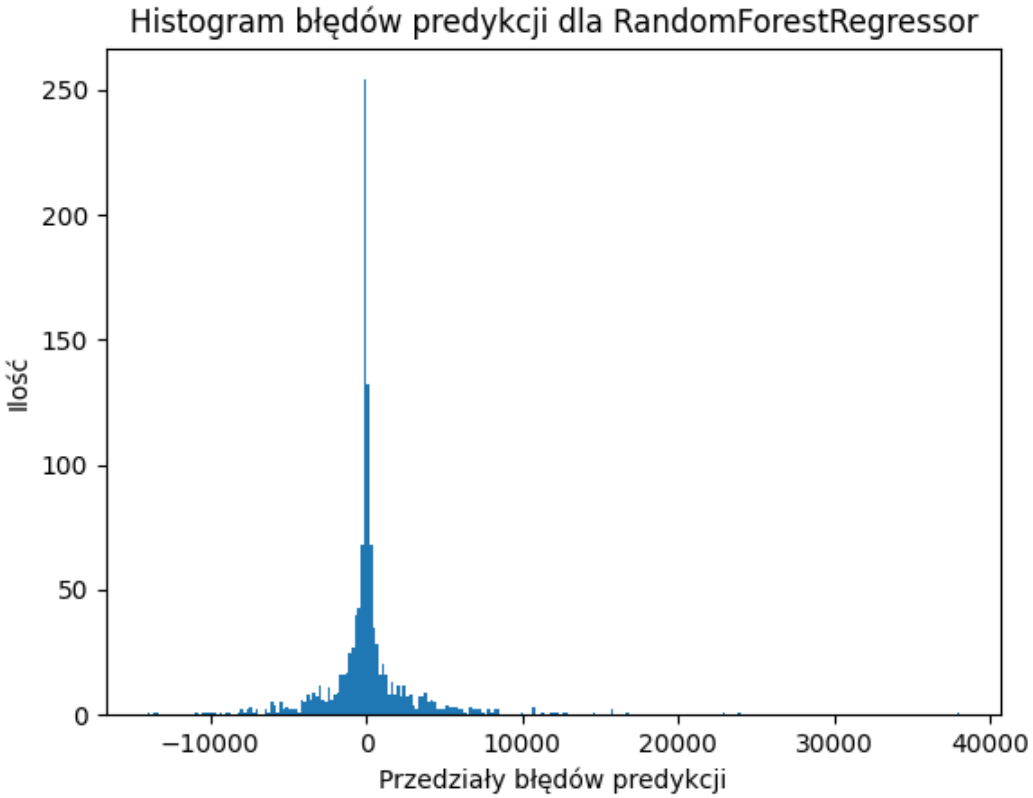
Łatwo widzieć, że najlepszym modelem jest `RandomForestRegressor`, który ma najniższe wartości błędów oraz najwyższy współczynnik determinacji, kolejnym będzie `DecisionTreeRegressor`, choć nie jest on idealny powiedziałbym, słaby. Co do reszty modele regresji liniowej mają pratykcznie te same wyniki.



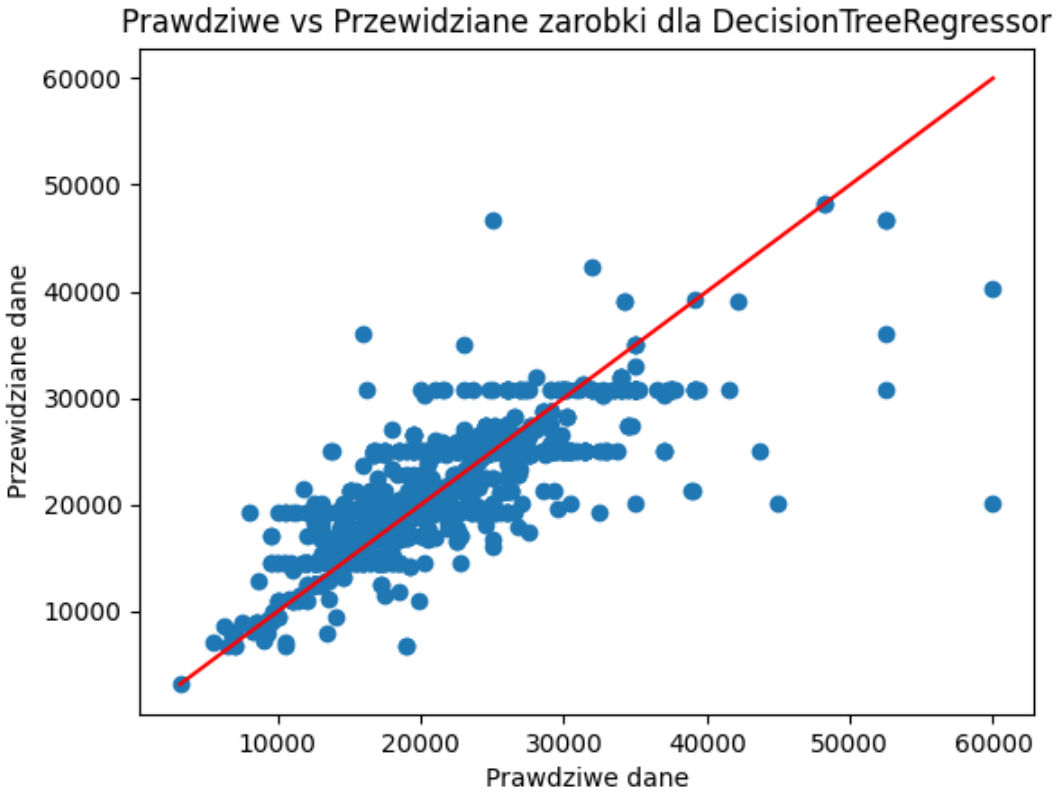
Rysunek 13: Dopasowanie danych przewidzianych do prawdziwych



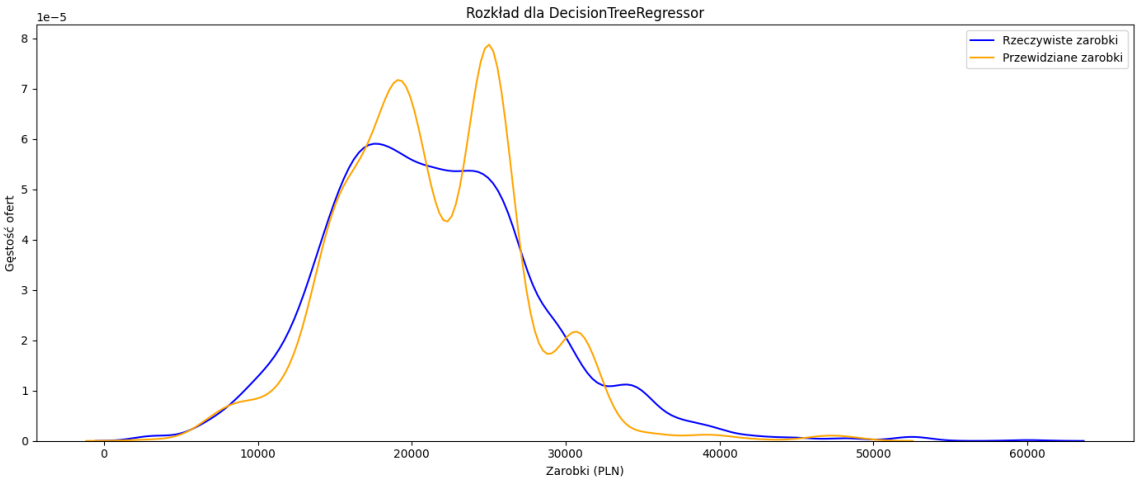
Rysunek 14: Rozkład dla przewidzianych i prawdziwych wartości



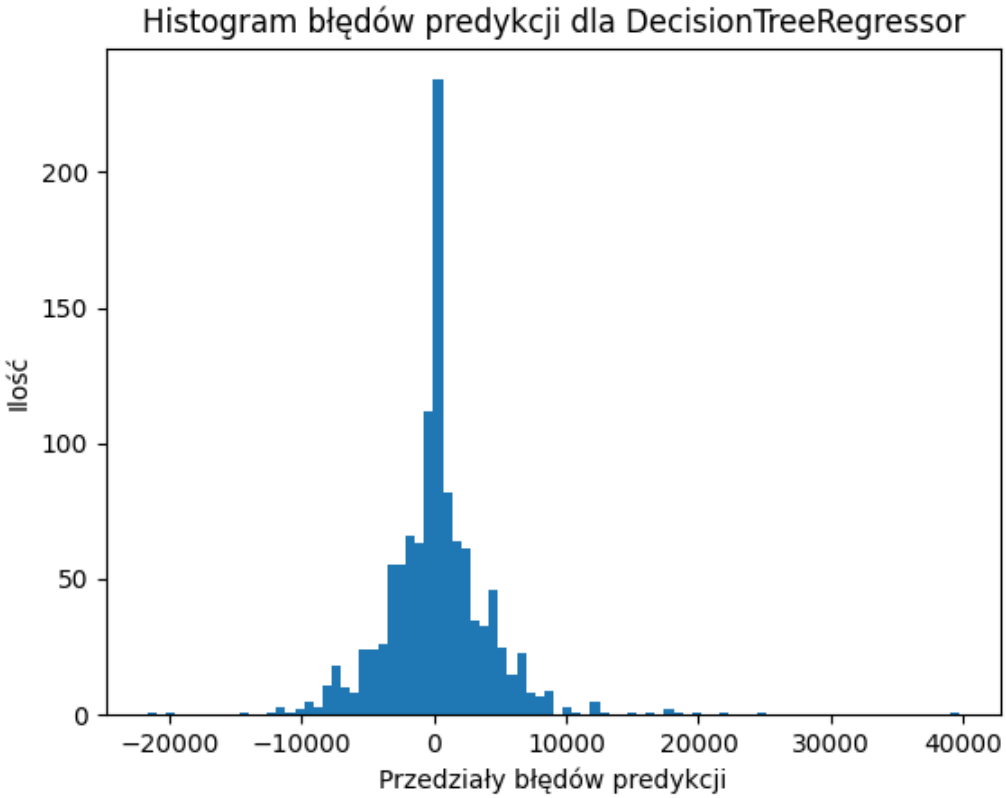
Rysunek 15: Rozkład dla przewidzianych i prawdziwych wartości



Rysunek 16: Dopasowanie danych przewidzianych do prawdziwych



Rysunek 17: Rozkład dla przewidzianych i prawdziwych wartości



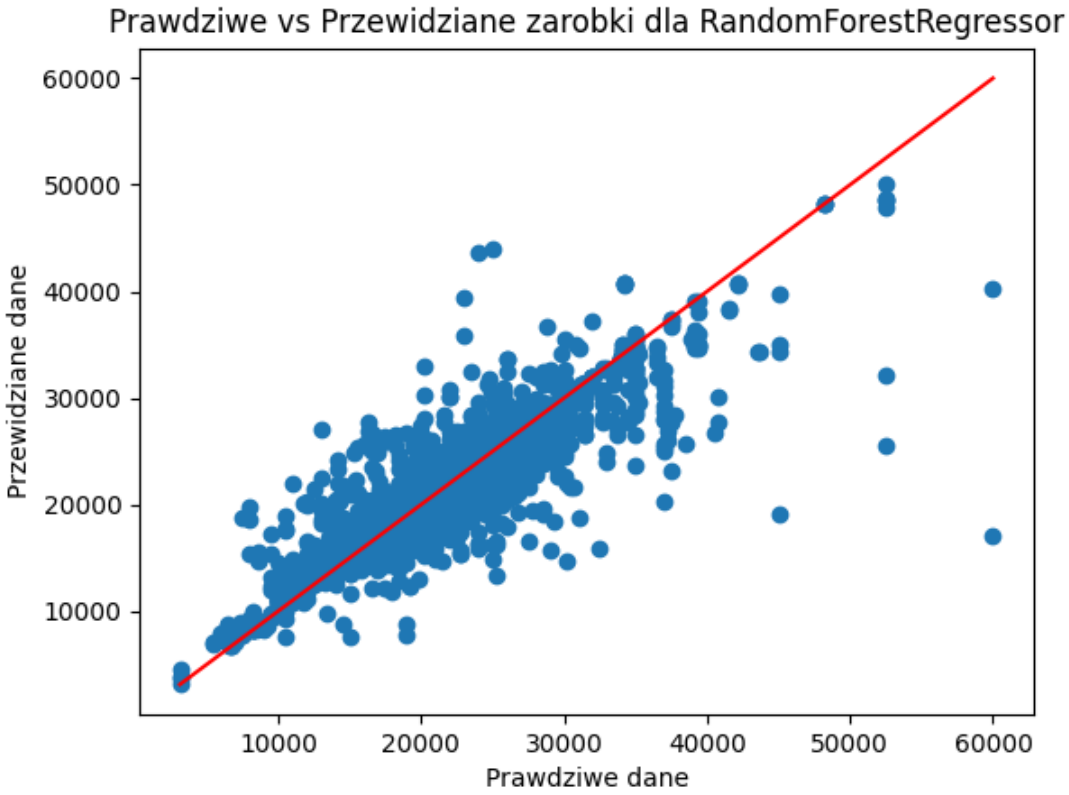
Rysunek 18: Rozkład dla przewidzianych i prawdziwych wartości

6.4.2 Podsumowanie wyników dla 80:20

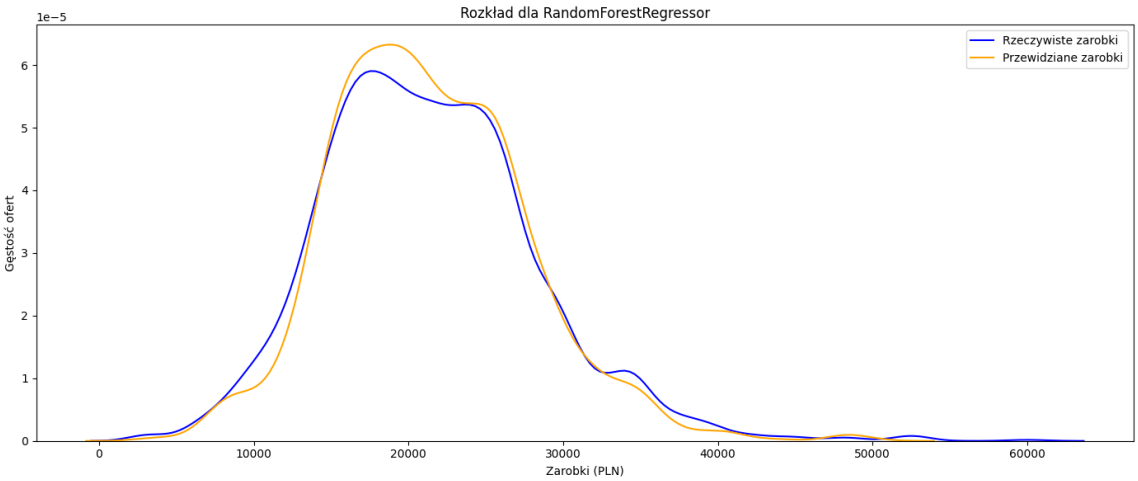
Wyniki pokazały nam, że najlepszym modelem do przewidywania zarobków od innych danych w ofercie jest `RandomForestRegressor` z parametrami `n_estimators=80`, chociaż błędy były dość wysokie, ale może to wynikać z dużego zakresu pensji oraz mogą być spowodowane małą ilością ofert pracy dla juniorów. Co warto zauważyć, w najlepszego modelu dane były w miarę skupione w prostej wyznaczającej idealny wynik. Dopasowanie rozkładu było też całkiem dobre, ponieważ wykresy w większej części nachodziły na siebie.

6.4.3 Wyniki dla podziału danych 60:40

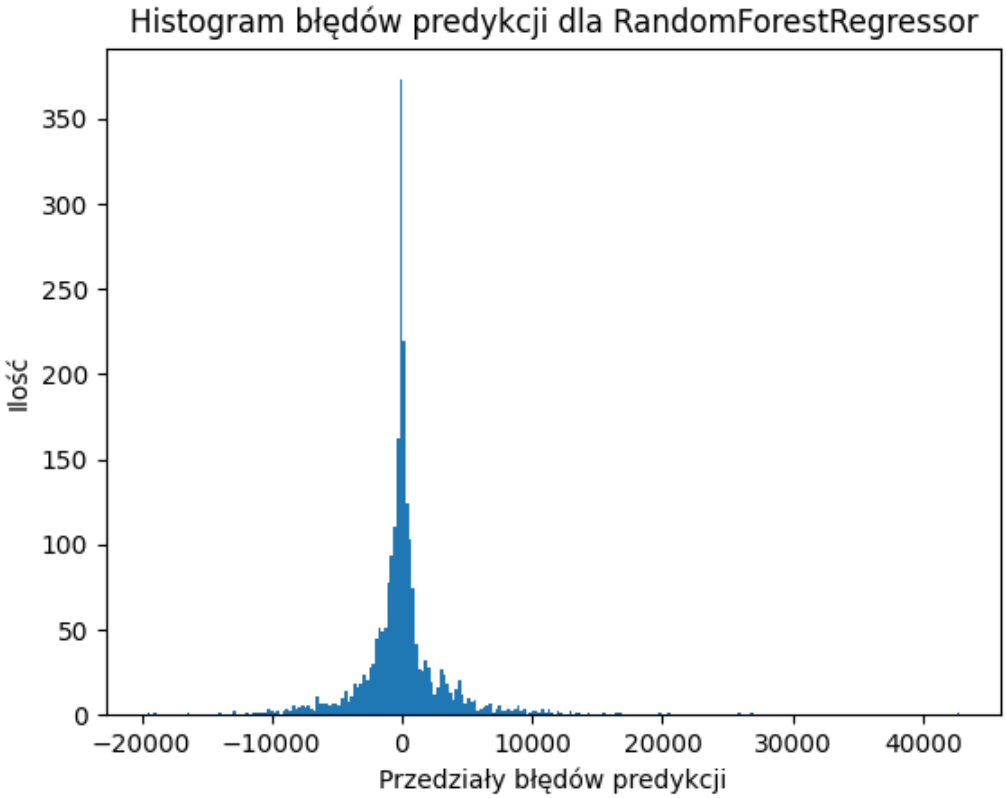
Model	Mean Absolute Error	Root Mean Squared Error	R^2 Score
LinearRegression	3685.29	4954.55	0.50
DecisionTreeRegressor	2777.77	4115.00	0.66
RandomForestRegressor	1799.06	3265.79	0.79
Ridge	3683.98	4955.21	0.50
Lasso	3684.60	4957.81	0.50



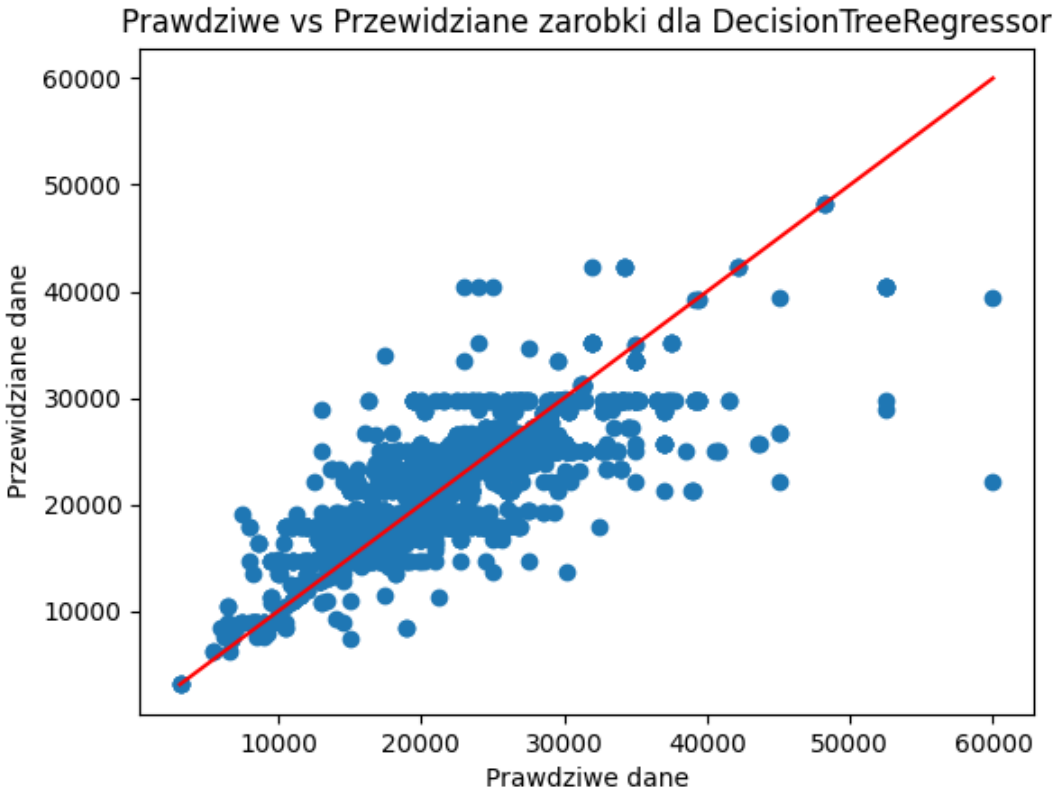
Rysunek 19: Dopasowanie danych przewidzianych do prawdziwych



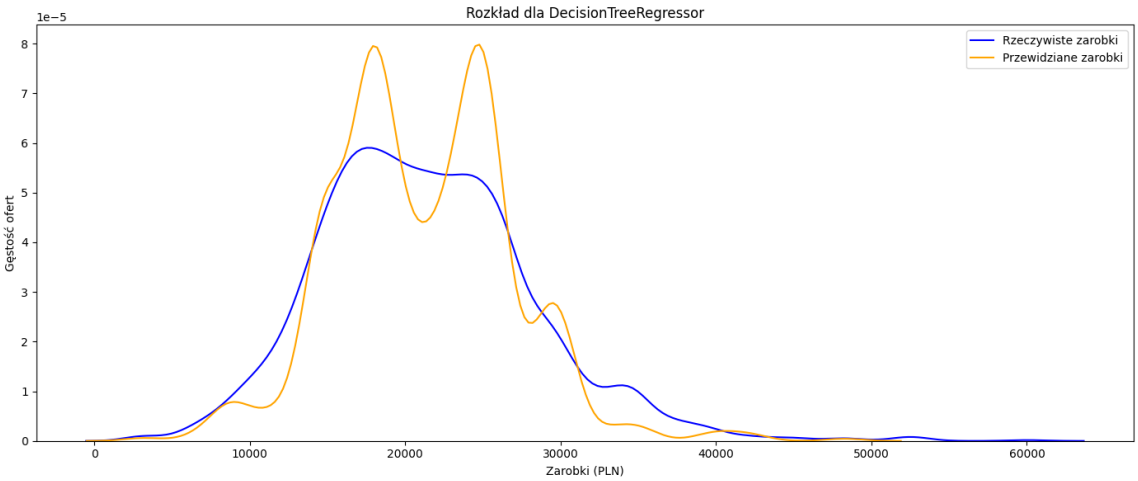
Rysunek 20: Rozkład dla przewidzianych i prawdziwych wartości



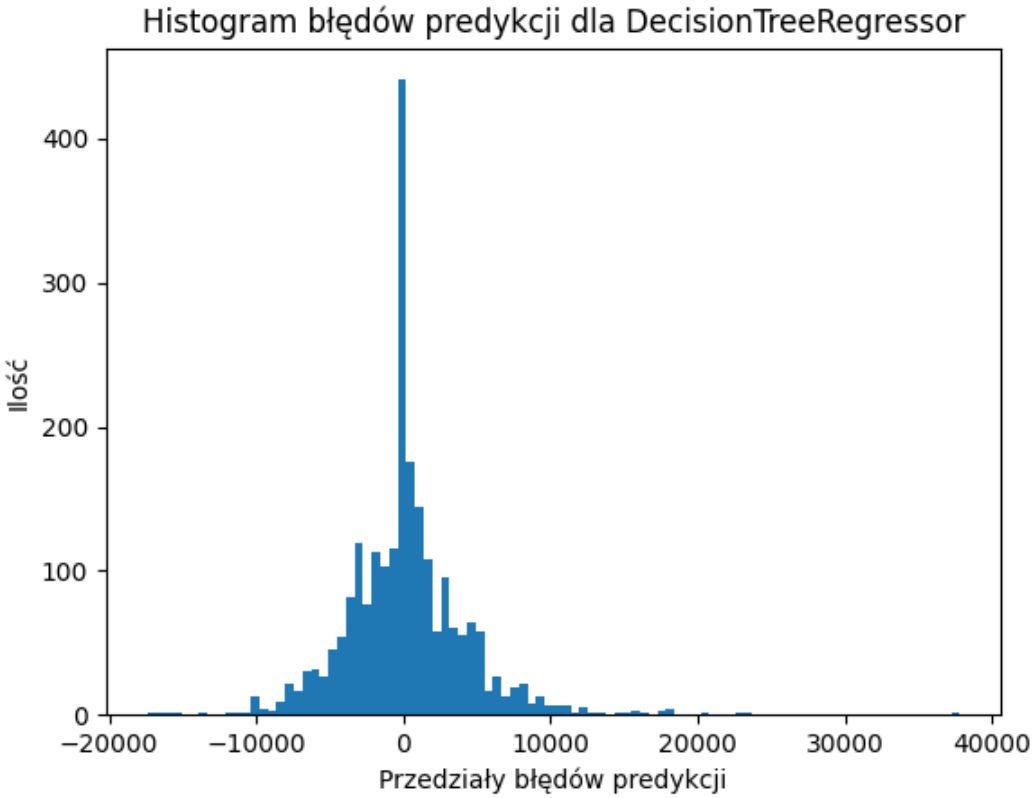
Rysunek 21: Rozkład dla przewidzianych i prawdziwych wartości



Rysunek 22: Dopasowanie danych przewidzianych do prawdziwych



Rysunek 23: Rozkład dla przewidzianych i prawdziwych wartości



Rysunek 24: Rozkład dla przewidzianych i prawdziwych wartości

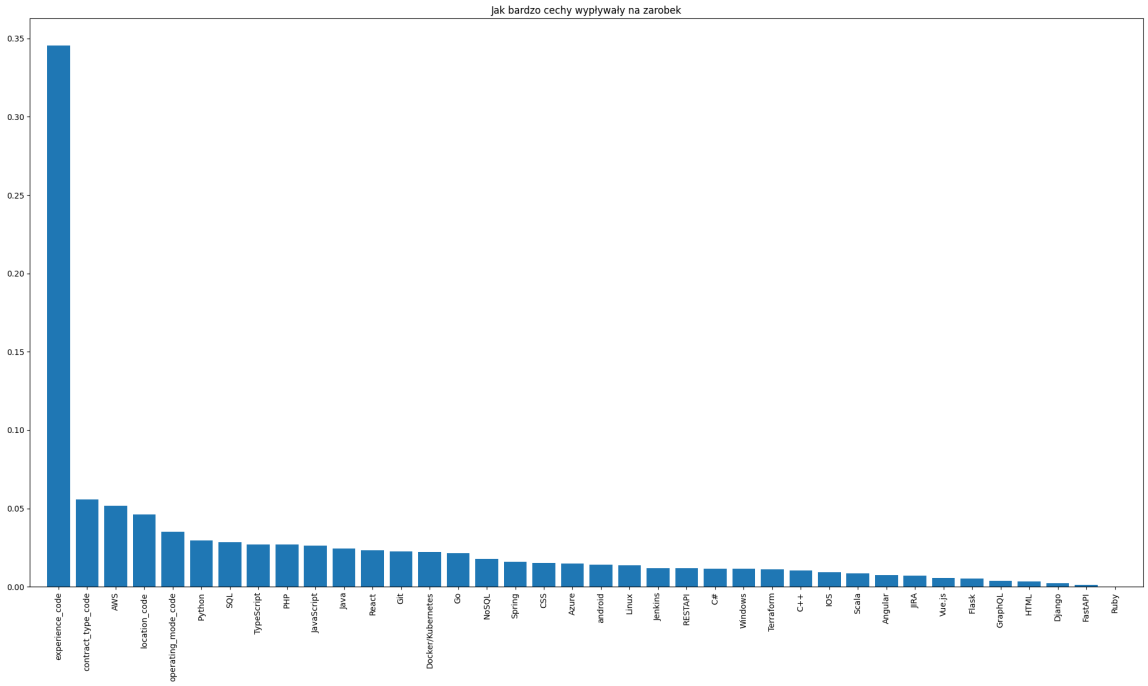
6.4.4 Podsumowanie wyników dla 60:40

Wyniki są mniej precyzyjne niż dla poprzedniego testu, ale to wynika z faktu, że model uczył się na mniejszej ilości ofert. Warto zauważyć, że najlepszy model to `RandomForestRegressor` z parametrami `n_estimators=80`, potem był `DecisionTreeRegressor`, który miał gorsze wyniki niż w poprzednim teście. Warto zauważyć, że modele regresji liniowej osiągnęły lepsze wyniki jeśli chodzi o błędy, ale współczynnik determinacji był gorszy niż dla poprzedniego podziału.

7 Podsumowanie

Podsumowując, jeśli chodzi o stworzenie modelu to najlepszym będzie `RandomForestRegressor(n_estimators=80)`, ponieważ dawał najmniejsze błędy chociaż i tak w skali zarobków nie były one małe. Do uczenia okazało się, że lepiej wybrać podziałkę 80:20, 80% dane treningowe a 20% dane testowe. Wydaje mi się również, że aby uzyskać lepsze wyniki, należałoby zaktualizować zbiór danych o nowe oferty (głównie oferty dla juniorów). Oczywiście model model może być jeszcze lepiej rozwinięty jeśli dodane zostałby nowe cechy np. stopień naukowy lub nowe technologie lub większe wyspecyfikowanie technologii.

Podczas pracy również można było sporządzić wykres, który przedstawiał najważniejsze zmienne w sensie wypływu na wynagrodzenie.



Rysunek 25: Zmienne mające wypłw na wynagrodzenie w ofercie pracy

Jak łatwo zauważyć, doświadczenie miało największy wpływ na przewidywaną wartość. Kolejnymi zmiennymi, które miały wpływ to **typ umowy**, **AWS**, **miasto**, **typ pracy**.

8 Testowanie wyuczonego modelu

Tak jak już wcześniej wspomniałem model, który uznałem za odpowiedni, czyli popełniający najmniejszy błąd spośród wszystkich to `RandoForestRegressor(n_estimators=80)`, dla podziałki 80:20. Przetestuję model, który przewidzi mi pensje na b2b i na umowie o pracę w zależności od moich technologii, które znam. Najważniejszą rzeczą jest tutaj to, żeby zarobki na b2b były większe niż na uopie, ponieważ pracodawcy nie muszą ponosić kosztów dodatkowych podatków, składek i świadczeń [1].

- **location**: Wrocław
- **exp**: Junior
- **operating_mode**: Hybrid
- **tech_stack**:
  - Docker/Kubernetes

- Python
- Linux
- React
- TypeScript
- JavaScript

8.1 Wyniki testów

Pensja w lokalizacji **Wrocław** dla **Junior** znającego *Docker/Kubernetes, Python, Linux, React, TypeScript, JavaScript*:

1. 9110.12 PLN Brutto na umowie o pracę
2. 9805.69 PLN Brutto na b2b

Jak można widzieć powyżej wyniki są troche zawyżone, z drugiej strony są uzasadnione błędem **RMSE**, ale dla tego testu wyszło, że na b2b zarabia się więcej niż na uopie, co ma sens. Ostatnią obserwacją może być, ilość technologii na wejściu, nie ma istotnego wpływu, chodzi mi o to, że bardziej liczy się konfiguracja technologii niż ich ilość, ponieważ model mógł nie dostać odpowiednich danych treninowych oraz warto dodać, że jeślie model dostał na treningu oferty z wysokimi zarobkami to wyniki będą zawyżone. Innym wyjaśnieniem może być to, że w ofertach dana technologia miała mniejszy bądź większy wpływ na wynagrodzenie/a.

Koniec końców udało się stworzyć jeden model przewidujący zarobki w zależności od innych zmiennych w ofercie pracy, który miał najmniejsze błędy spośród innych modeli, co jest zadowalające a przewidywane wartości są 'powiedzmy', że sensowne.

*Link to całego projektu znajduje się na moim **GitHubie***

9 Bibliografia

Literatura

[1] *Umowa o pracę a kontrakt B2B – jak zarobisz więcej?*, bizky.ai

[2] *Ocena modeli regresji*, qlik.com