

Analiza ofert pracy w IT w Polsce w 2024 roku.

Łukasz Fabia (272724)
MSiD Lab 09:15 TP
Informatyka Stosowana

8 czerwca 2024

Spis treści

1	Wstęp	3
2	Dane	3
2.1	Model danej	3
2.2	Obsługa technologii, lokalizacji	3
2.3	Pożykiwanie danych	4
3	Wygląd do danych	5
4	Rozkłady i statystyki	6
4.1	Jak się pracuje w IT?	7
4.2	Kogo szukają pracodawcy?	8
4.3	Jak rozkładają się zarobki?	9
4.4	Jakie technologie są najbardziej poszukiwane?	10
4.5	Gdzie jest największy popyt na programistów?	11
4.6	Gdzie poszukiwani są juniorzy?	12
5	Powiązania między danymi	13
5.1	Powiązania między technologiami	13
5.2	Powiązania między innymi zmiennymi	15
5.3	Zarobek a technologie	16
6	Czy da się przewidzieć zarobki, w zależności od mojego tech-stacku?	16
6.1	Ogólnie o problemie	16
6.2	Dobór modeli	17
6.3	Trochę statystyki - metryki	17
6.3.1	Pierwiastek ze średniego błędu kwadratowego	17
6.3.2	Współczynnik determinacji	17
6.3.3	Średni błąd bezwzględny	17
6.4	Jak to zrobić?	18
6.4.1	Wyniki dla podziału danych 80:20	18
6.4.2	Podsumowanie wyników dla 80:20	21
6.4.3	Wyniki dla podziału danych 60:40	22
6.4.4	Podsumowanie wyników dla 60:40	24

7 Podsumowanie	25
8 Testowanie wyuczonego modelu	26
8.1 Wyniki testów	26
9 Bibliografia	27

1 Wstęp

Celem badań jest analiza danych dotyczących ofert pracy w IT. W swojej pracy postaram się odpowiedzieć na pytanie, które umiejętności w branży IT są najbardziej poszukiwane oraz jakie jest wynagrodzenie, w zależności od znajomości danych języków, frameworków czy narzędzi. W tym celu stworzę model, który będzie przewidywał stawki wynagrodzenia w zależności od danych wejściowych, o których później.

2 Dane

Dane pozyskam z serwisu justjoin.it, który zbiera oferty pracy z wielu różnych stron internetowych. Na wyżej wymienionej stronie mamy kategorie, które mogą być przydatne do filtrowania danych. Są one mało jednak mało przydatne, ponieważ przypisane do ofert, nawiązują w jakiś sposób do np. JS. Kategorie są następujące: JS, PHP, Ruby, Python, Java, Net, Mobile, C, DevOps, Security, Data, Go, Game, Scala. Moje dane będą pozyskiwane z tych podstron.

Przeanalizuję zarobki tylko na b2b oraz zarobki na podstawie umowy o pracę (uop). Są to najbardziej popularne formy zatrudnienia w IT. Umów takich jak zlecenie, umowa o staż praktycznie nie występują. Do analizy będę również brał pod uwagę inne parametry, które występują w ofercie.

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 zdefiniuję słownik - klucz, wartość, gdzie klucz to ustandaryzowana technologia, a wartość do synonimy tej technologii.

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

Celem tego zabiegu jest zmniejszenie liczby technologii, które będę brał pod uwagę. Kolejnym krokiem będzie obsługa lokalizacji. W tym przypadku, jeśli oferta dotyczy kilku miast, to po przetworzeniu pojawi się w zbiorze klika ofert z tymi samymi danymi, ale dla różnych miast. Podobnie obsłużę kontrakty, które są w ofercie pracy, ponieważ z punktu prawnego, jeśli oferta dotyczy obu kontraktów to tak naprawdę są to dwie różne oferty pracy.

2.3 Pozykiwanie danych

Skorzystam z 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 śmieciowych 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 kategoriycznych (np. miasta, typ pracy, kontrakty),
- wzięcie średniej zarobków, a następnie usunięcie widełek w zarobkach (np. 10-15k \implies 12.5k),
- rozbicie ofert pracy dla kontraktów oraz miast, 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, w których określona została stawka godzinowa, ponieważ nie można jej określić w łatwy sposób,
- usunięcie tytułów ofert.

Ofert ze stawką godzinową było kilka, więc nie wpływają 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.

Widélki dla Juniora:

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

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

Widélki dla Mida:

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

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

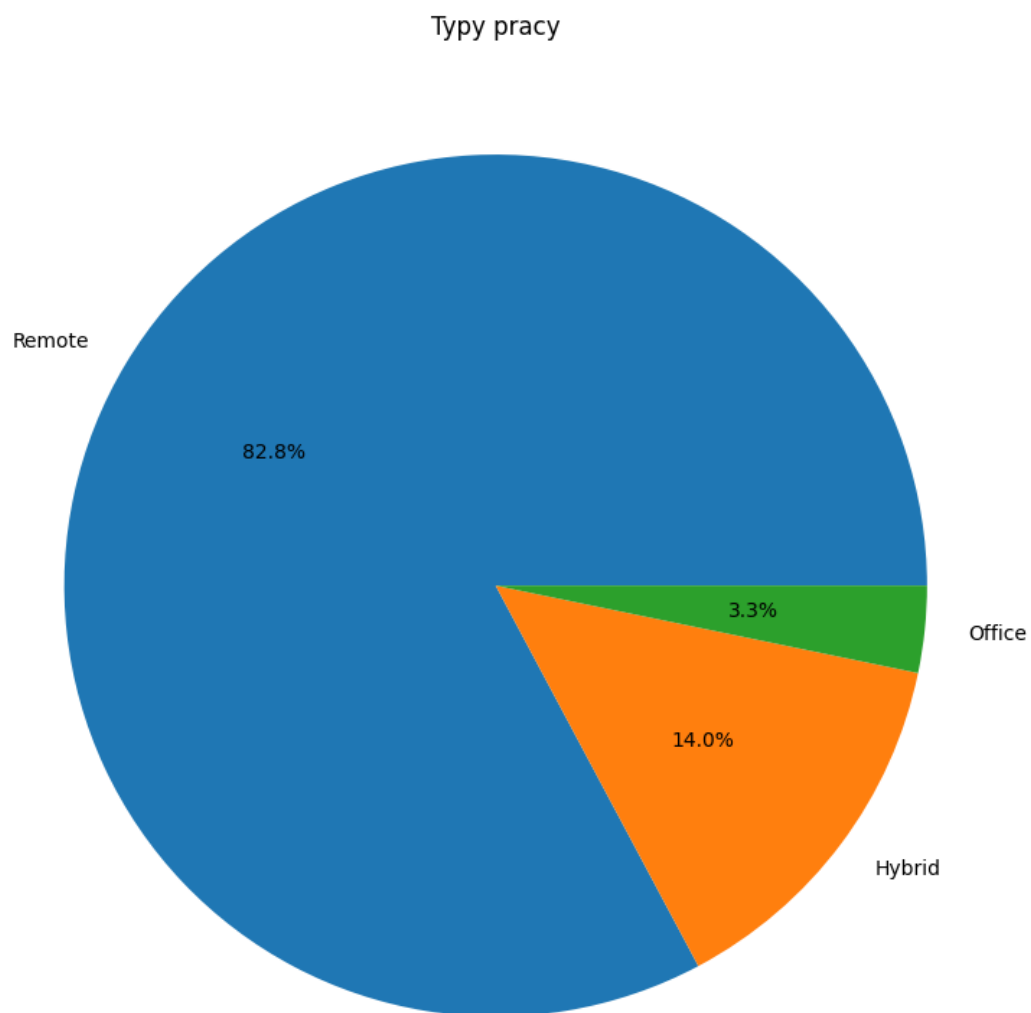
Widélki dla Seniora:

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

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

Wynagrodzenia te są na dobrym poziomie, ponieważ dla Mida i Seniora mamy większą pensję na b2b niż na uop, dla juniora jest na odwrót. Przyczyną może być to, że umowę o pracę dostają bardzo dobrzy programiści albo jest mało ofert pracy dla juniora na b2b.

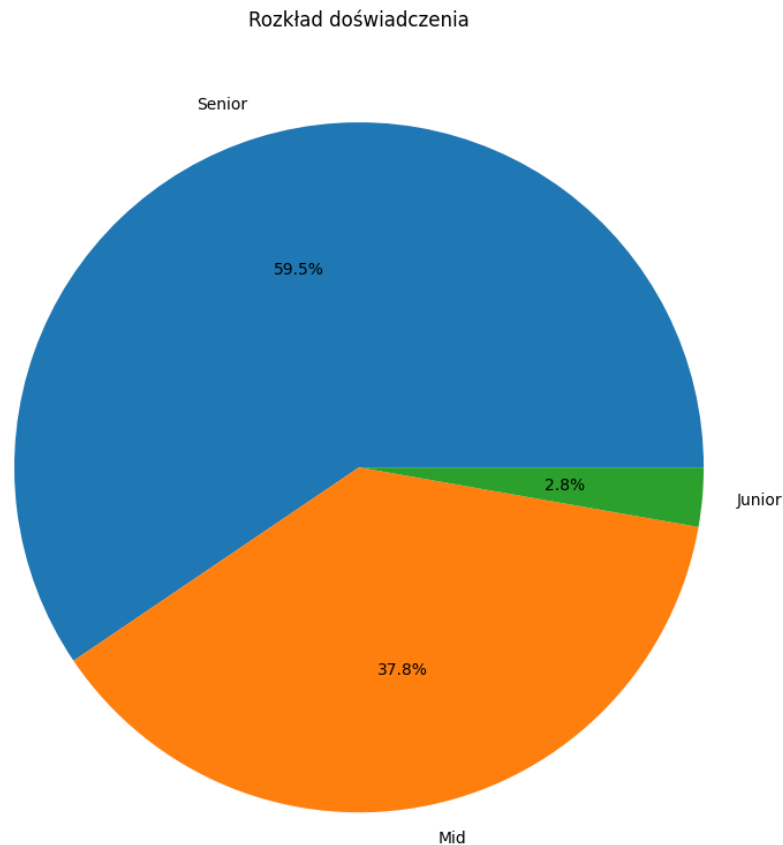
4.1 Jak się pracuje w IT?



Rysunek 1: Rozkład typów pracy

Najwięcej ofert dotyczy pracy zdalnej.

4.2 Kogo szukają pracodawcy?



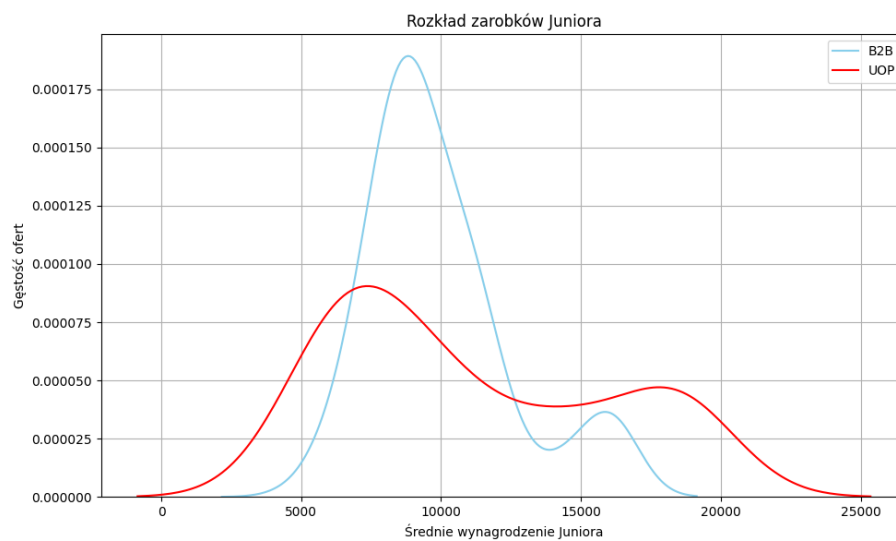
Rysunek 2: Rozkład typów pracy

Tak jak można było się spodziewać - najwięcej ofert pracy jest dla seniorów. Pracodawca ma większe zaufanie do Seniora/Mida. Gorzej jest z ofertami dla młodych programistów. Tutaj procent ofert wyniósł zaledwie 2.8%. W porównaniu do innych grup jest to niewiele.

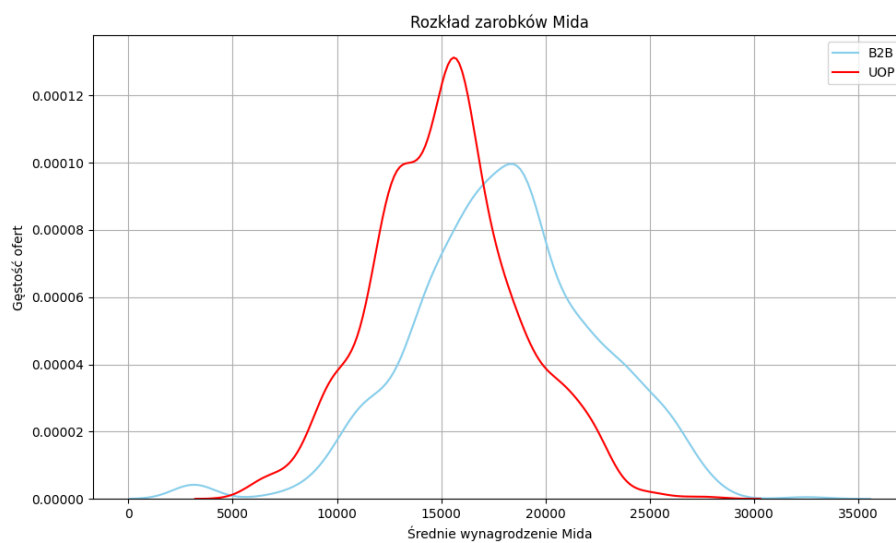
Czy to oznacza, że młodzi programiści mają trudniej, a słynne "eldorado" w IT jest tylko dla doświadczonych programistów?

Można stwierdzić, że juniorzy mają trudniejszy start w branży IT, ale jeśli im się uda, to zarobki są atrakcyjne.

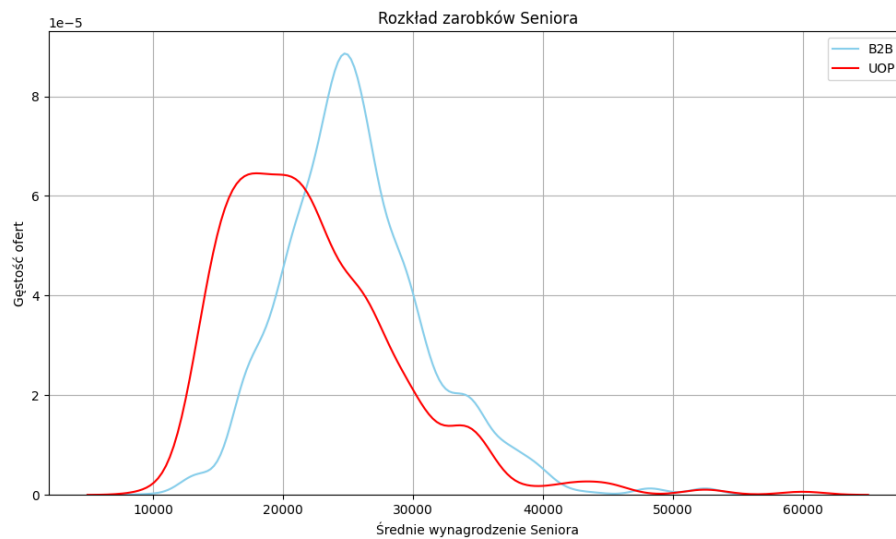
4.3 Jak rozkładają się zarobki?



Rysunek 3: Rozkłady zarobków dla poszczególnych umów dla juniorów

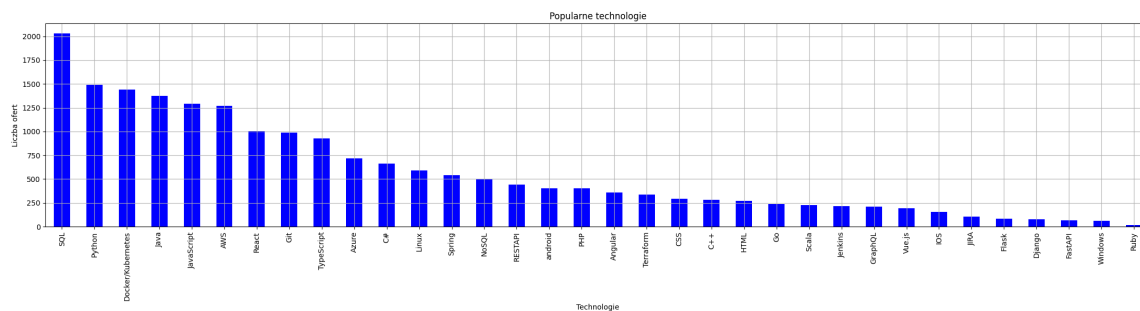


Rysunek 4: Rozkłady zarobków dla poszczególnych umów dla midów



Rysunek 5: Rozkłady zarobków dla poszczególnych umów dla seniorów

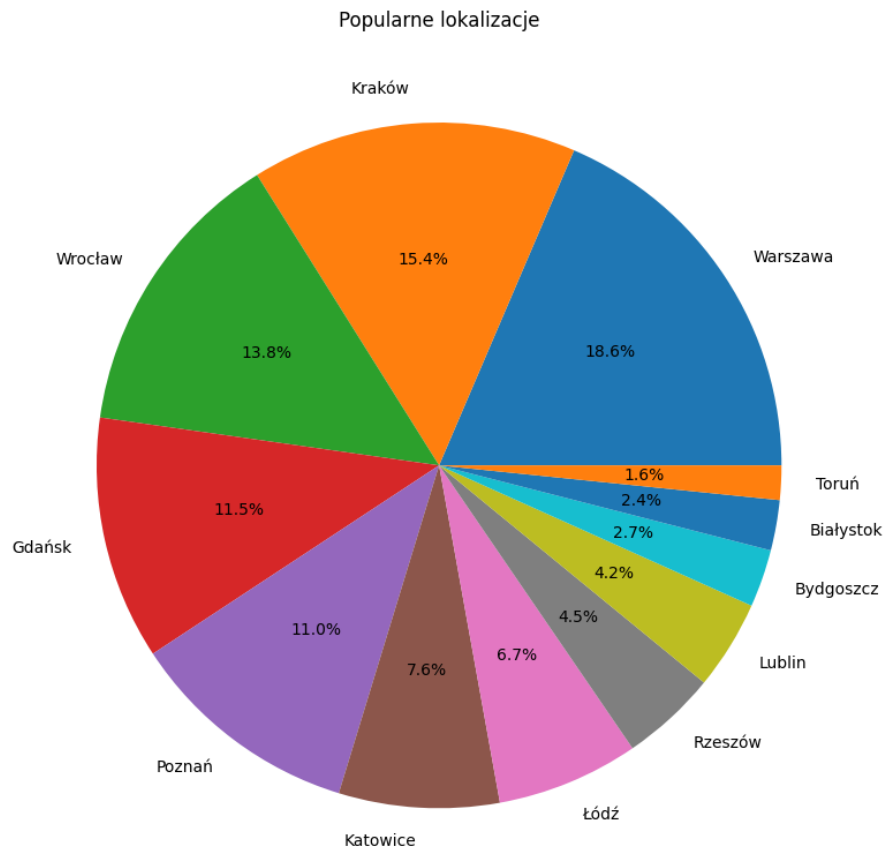
4.4 Jakie technologie są najbardziej poszukiwane?



Rysunek 6: Popularne technologie w ofertach pracy w Polsce

Najpopularniejszą technologią okazał się SQL. Bez jego znajomości trudno znaleźć pracę w IT. Oczywiście nie mogło zabraknąć Pythona oraz JavaScriptu, jeśli chodzi o języki skryptowe. Co warto, zaznaczyć narzędzia takie jak Docker czy Kubernetes również są bardzo popularne i warto je znać. Java wygrywa z C#.

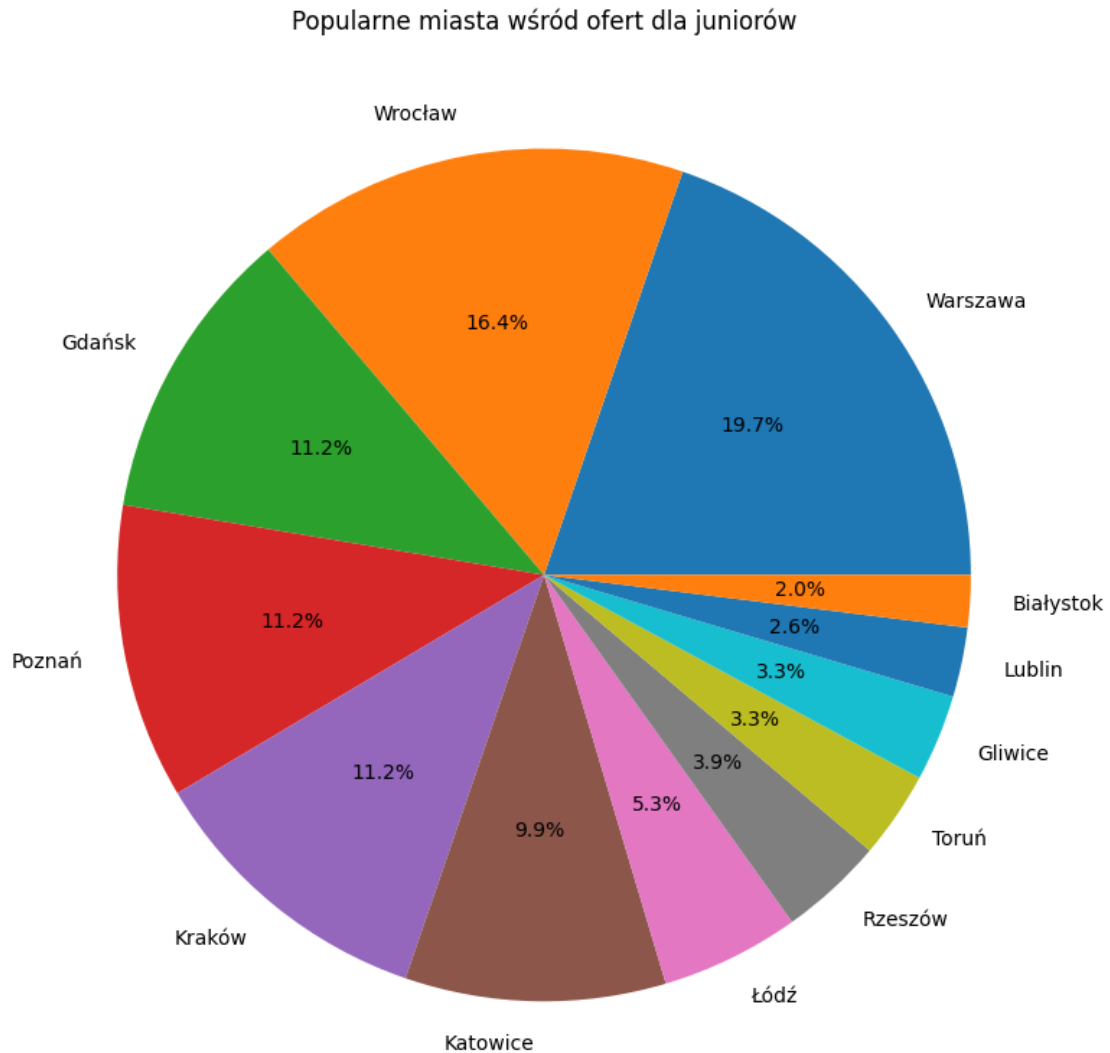
4.5 Gdzie jest największy popyt na programistów?



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 w **Gdańsku** również pojawiły się stosunkowo duże ilości ofert pracy.

4.6 Gdzie poszukiwani są juniorzy?

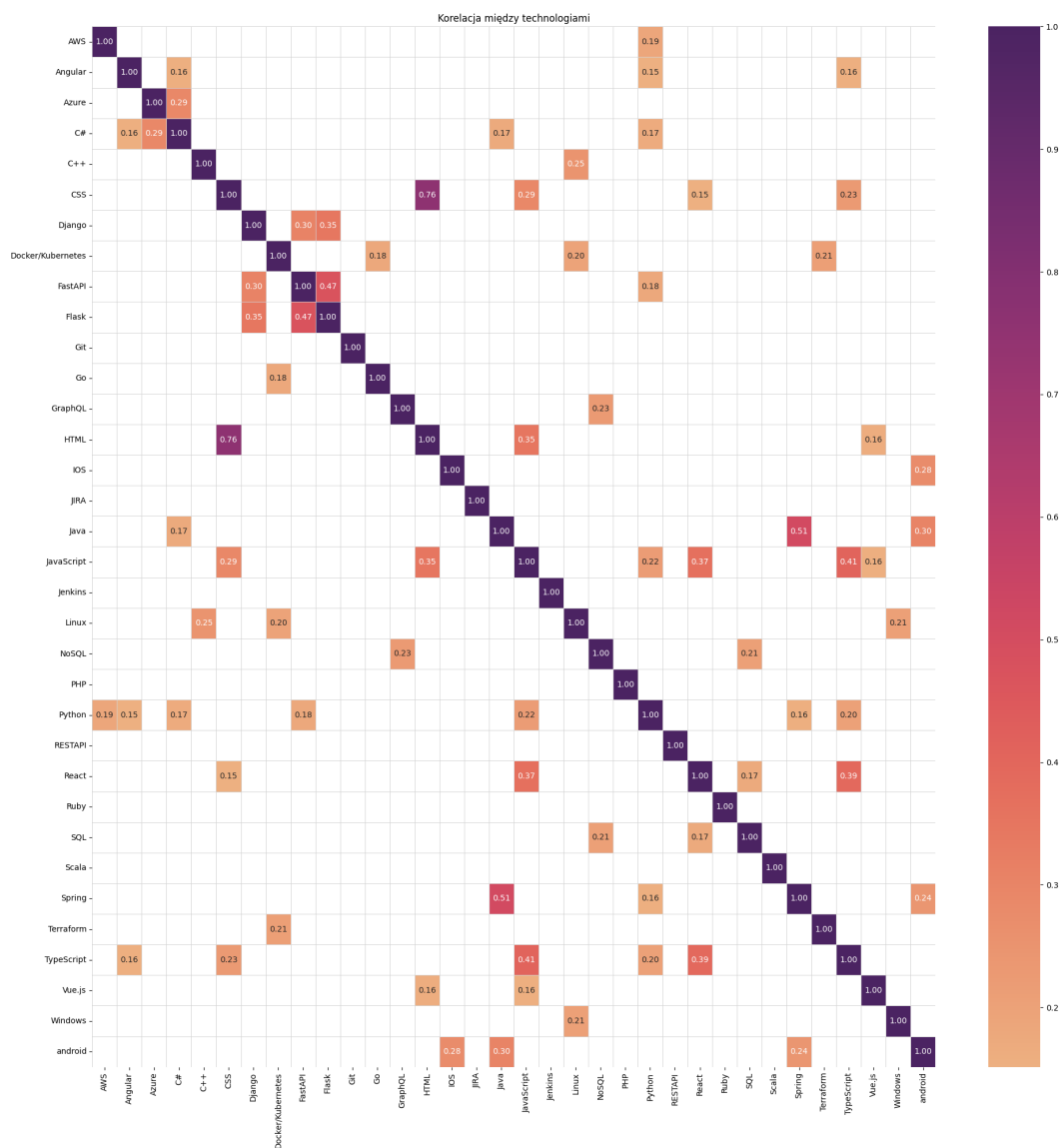


Rysunek 8: Popularne miasta w ofertach dla juniorów

Warszawa jest najbardziej przyjazna dla juniorów, ale warto zauważyć, że wykres nie różni się bardzo od poprzedniego. Jednak, to **Gdańsk** jest na 3 miejscu w zestawieniu dla juniorów.

5 Powiązania między danymi

5.1 Powiązania między technologiami



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

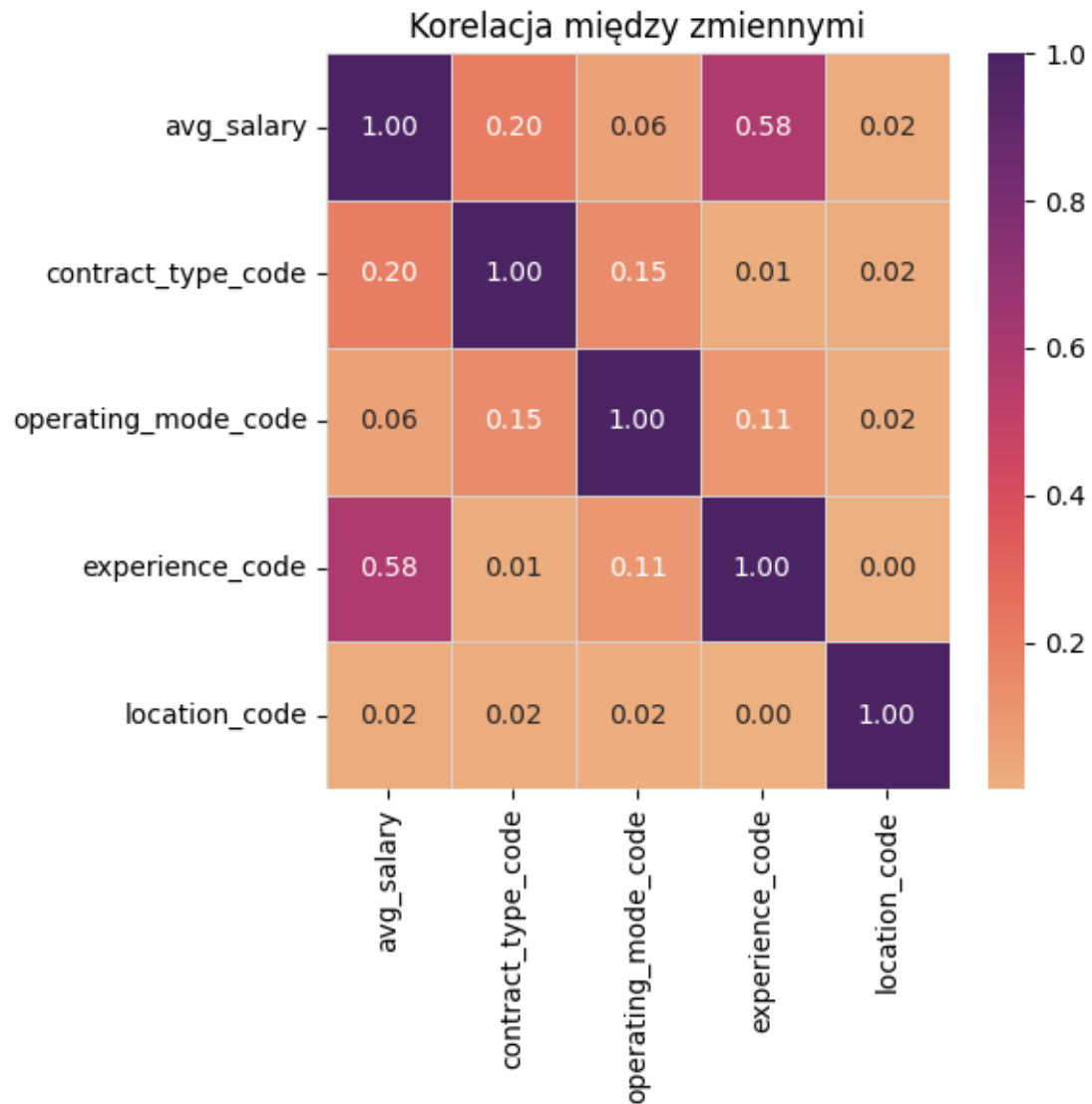
Co można zauważyć?

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

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

To tylko kilka przykładów wynikających z wykresu powyżej, ale warto zauważyć, że nie ma tu silnych powiązań między technologiami. Oferty pracy są zróżnicowane i zależą od firmy.

5.2 Powiązania między innymi zmiennymi

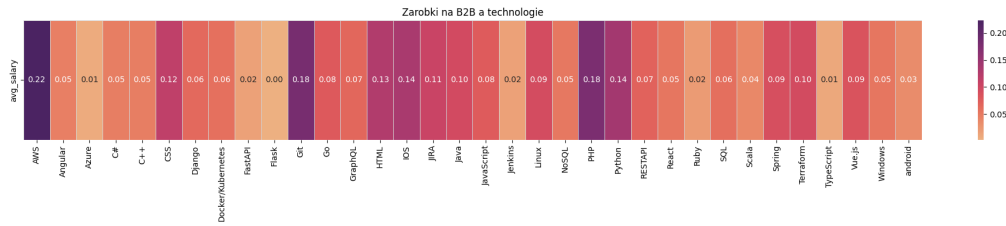


Rysunek 10: Powiązania między innymi zmiennymi

Co można zauważyć?

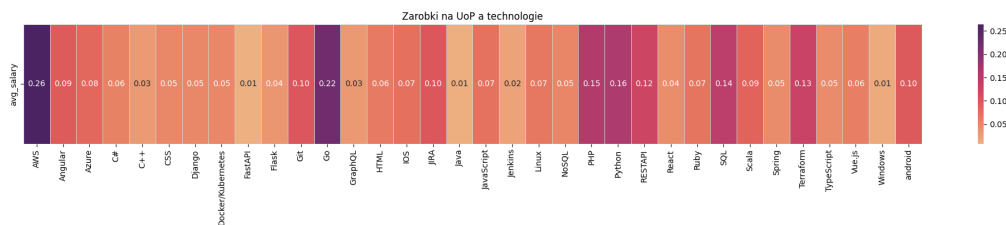
1. Średnia pensja jest mocno powiązana z doświadczeniem.
2. Typ kontraktu zależy od doświadczenia.

5.3 Zarobek a technologie



Rysunek 11: Powiązania między zarobkiem b2b a technologiami

AWS, Git, Python, PHP mają wpływ 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, da się przewidzieć zarobki od tech-stacku. 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. DecisionTreeRegressor
3. RandomForestRegressor

Wszystkie modele pochodzą z modułu *skelearn*.

6.3 Trochę statystyki - metryki

Do oceny modeli wykorzystam metryki takie jak:

- **Root Mean Squared Error** - pierwiastek z średniego błędu kwadratowego,
 - **R-squared** - współczynnik determinacji R^2 ,
 - **Mean Absolute Error** - średni błąd bezwzględny,
- ale wybrać będę po RMSE.

6.3.1 Pierwiastek ze 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].

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

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 się do danych.

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

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.

$$MAE = \frac{\sum_{i=1}^n |y_i - x_i|}{n}. \quad (3)$$

6.4 Jak to zrobić?

Moje rozwiązanie problemu 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 na podstawie RMSE. Następnie przewidzę zarobki dla kilku danych wejściowych, a na końcu przedstawię wyniki w postaci wizualizacji.

Streszczenie

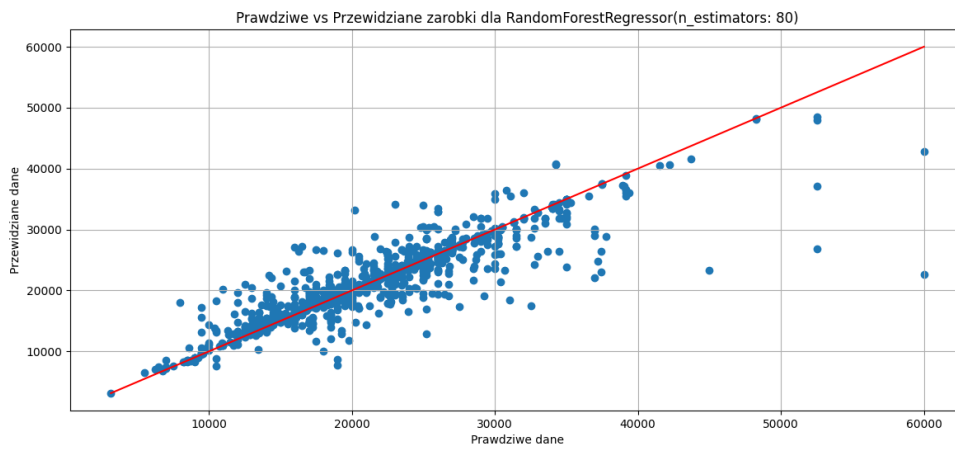
W następnych rozdziałach skupię się na wynikach modeli, a także na wizualizacji wyników, aby nie tworzyć zbyt długiego raportu, nie będę analizować słabych modeli tylko skupię się na dwóch najlepszych. Wszystkie wyniki z uczenia zostaną zapisane w folderze `../analysis/plots/wyniki/` ew. można też podejrzeć plik z rozwiązaniem problemu w `../analysis/analysis.ipynb`.

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

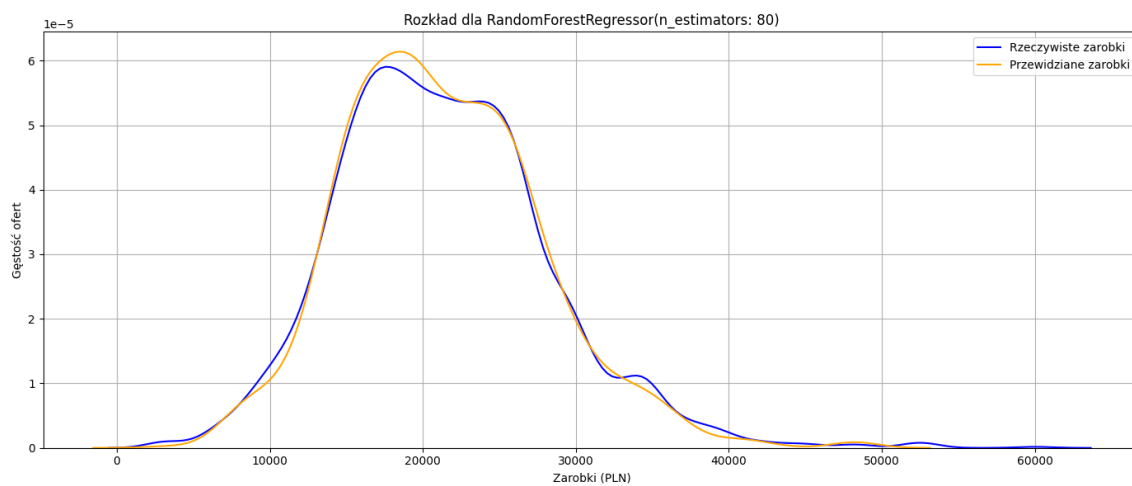
6.4.1 Wyniki dla podziału danych 80:20

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

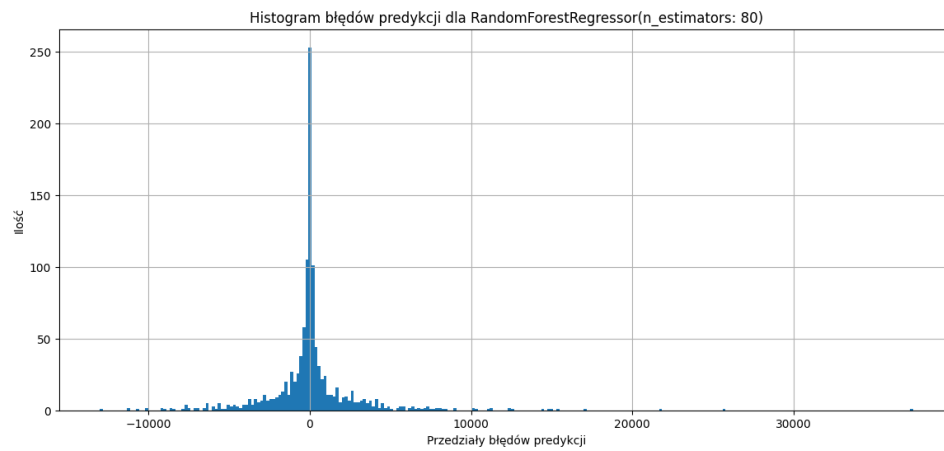
Łatwo zauważyć, że najlepszym modelem jest `RandomForestRegressor`, który ma najniższe wartości błędów oraz najwyższy współczynnik determinacji, kolejnym będzie `DecisionTreeRegressor`, choć nie jest on idealny powiedziałbym, słaby. Pozostałe modele, czyli modele regresji liniowej mają praktycznie te same wyniki.



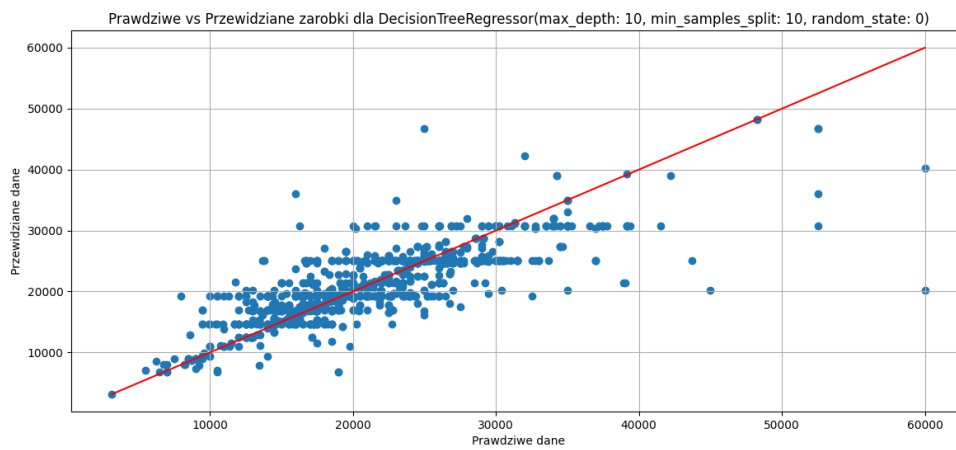
Rysunek 13: Dopasowanie danych przewidzianych do prawdziwych



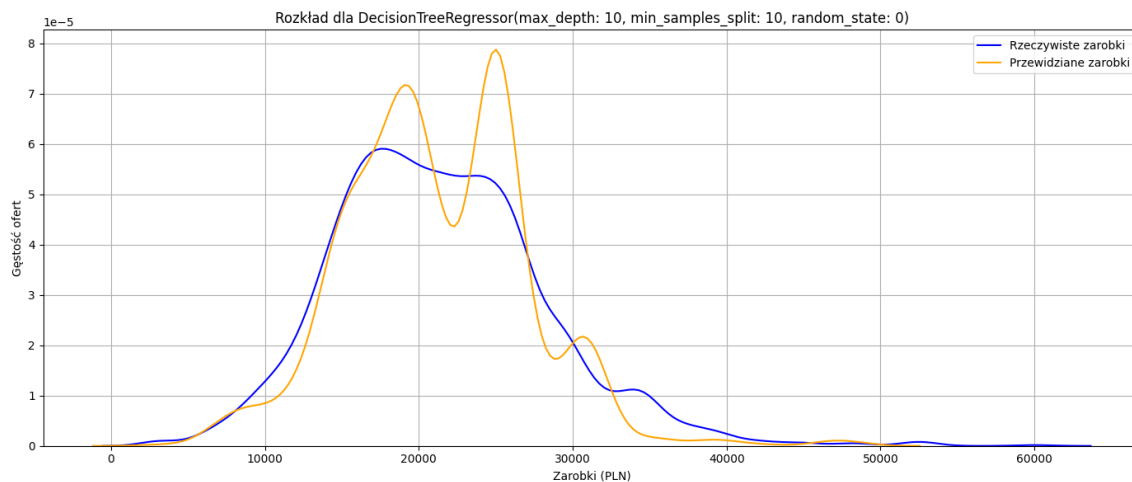
Rysunek 14: Rozkład dla przewidzianych i prawdziwych wartości



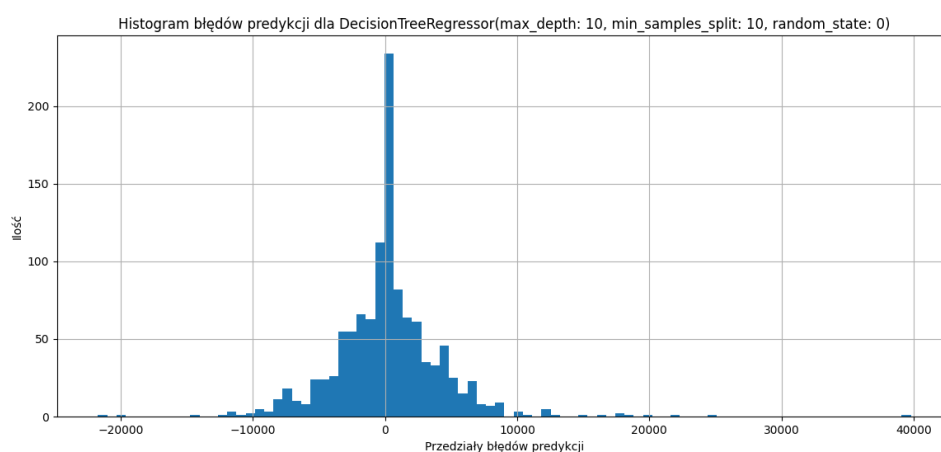
Rysunek 15: Rozkład dla przewidzianych i prawdziwych wartości



Rysunek 16: Dopasowanie danych przewidzianych do prawdziwych



Rysunek 17: Rozkład dla przewidzianych i prawdziwych wartości



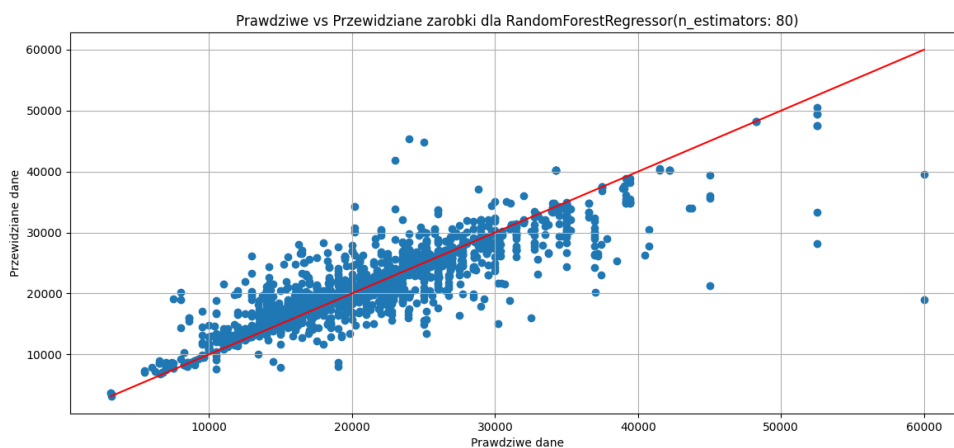
Rysunek 18: Rozkład dla przewidzianych i prawdziwych wartości

6.4.2 Podsumowanie wyników dla 80:20

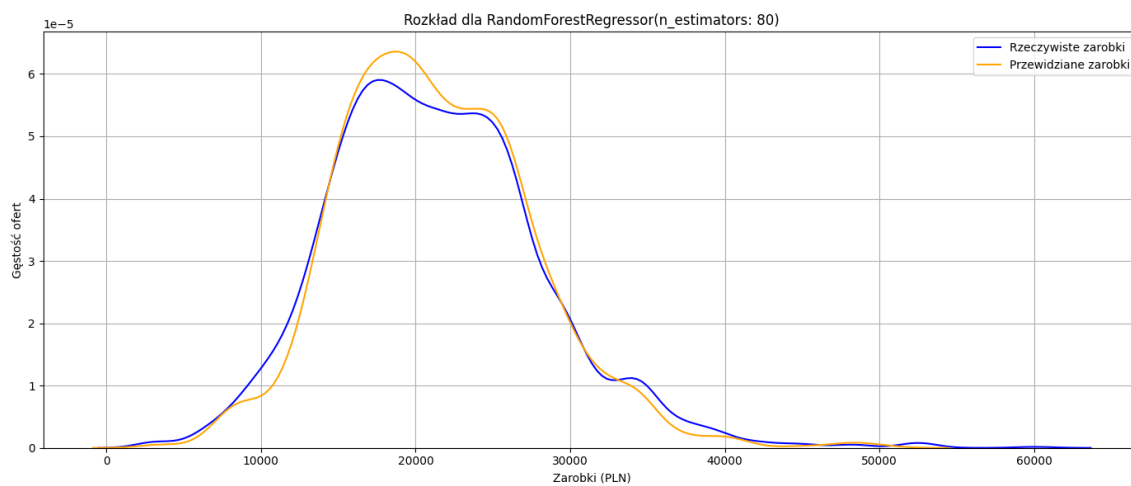
Wyniki pokazały nam, że najlepszym modelem do przewidywania zarobków jest `RandomForestRegressor` z parametrami `n_estimators=80`, chociaż błędy były dość wysokie, ale może to wynikać z rozpiętości wiedełek pensji lub mogą być spowodowane małą ilością ofert pracy dla juniorów. Warto zauważyć, że dla 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

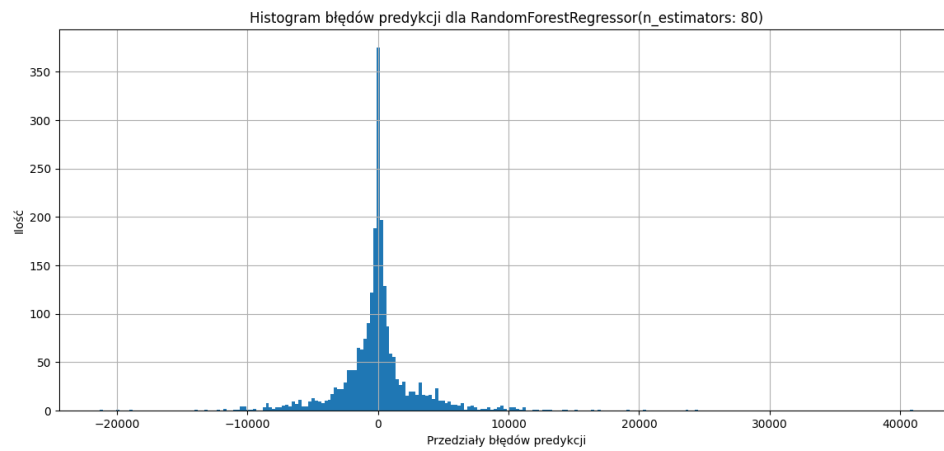
Model	Mean Absolute Error	Root Mean Squared Error	R ² Score
LinearRegression	3685.29	4954.55	0.50
DecisionTreeRegressor	2777.77	4115.00	0.66
RandomForestRegressor	1799.06	3265.79	0.79
Ridge	3683.98	4955.21	0.50
Lasso	3684.60	4957.81	0.50



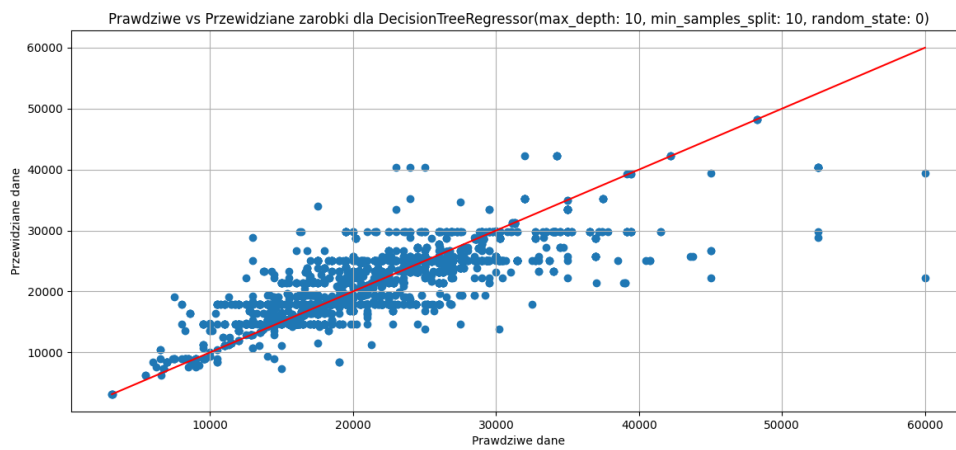
Rysunek 19: Dopasowanie danych przewidzianych do prawdziwych



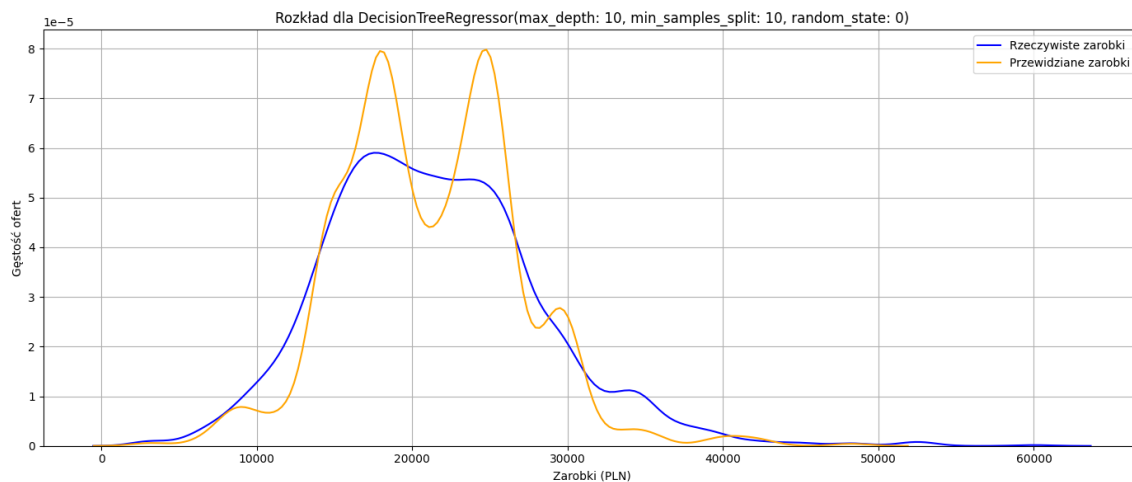
Rysunek 20: Rozkład dla przewidzianych i prawdziwych wartości



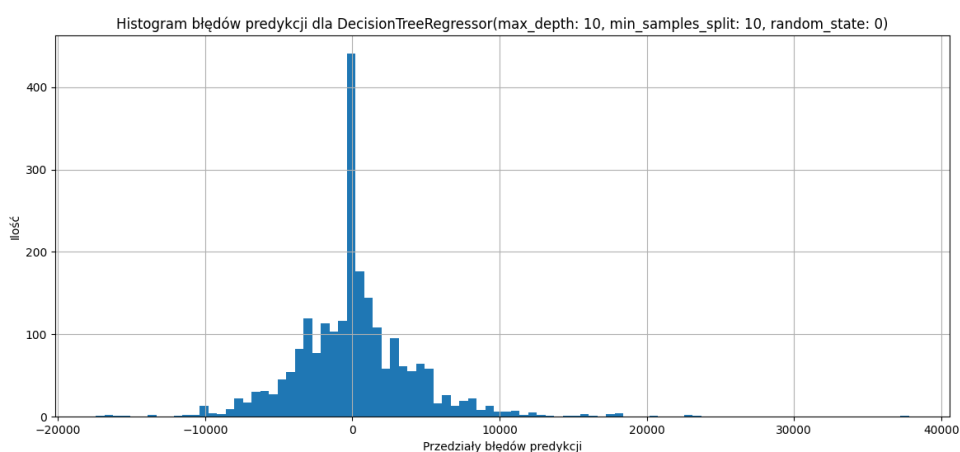
Rysunek 21: Rozkład dla przewidzianych i prawdziwych wartości



Rysunek 22: Dopasowanie danych przewidzianych do prawdziwych



Rysunek 23: Rozkład dla przewidzianych i prawdziwych wartości



Rysunek 24: Rozkład dla przewidzianych i prawdziwych wartości

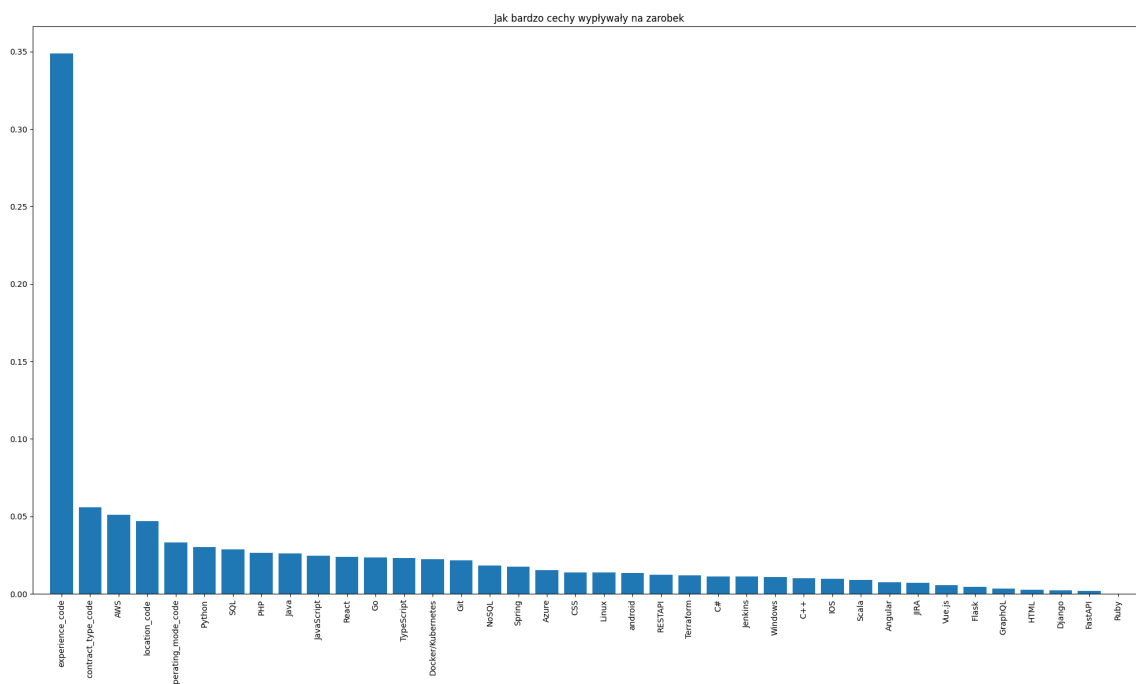
6.4.4 Podsumowanie wyników dla 60:40

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

Najlepszym modelem będzie `RandomForestRegressor(n_estimators=80)`, ponieważ dawał najmniejsze błędy chociaż i tak w skali zarobków nie były one małe. Do uczenia modelu lepiej wybrać podziałkę 80:20. Wydaje mi się również, że aby uzyskać lepsze wyniki, należałoby zaktualizować zbiór danych o nowe oferty (głównie oferty dla juniorów). Warto dodać, że gdy kompilujemy skrypt to metryki mogą się różnić dla `RandomForestRegressor`, ale są to różnice rzędu 20-50 +/-, metryki. Może to wynikać z faktu jak dzielone są dane na treningowe i testowe. Oczywiście, model może być jeszcze lepiej rozwinięty, jeśli dodane zostałyby nowe cechy, np. stopień naukowy lub nowe technologie lub większe wyspecyfikowanie technologii.

Podczas pracy również można było sporządzić wykres, który przedstawiał najważniejsze zmienne wpływające na wynagrodzenie.



Rysunek 25: Zmienne mające wpływ 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 pensję na b2b i na umowie o pracę w zależności od 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*:

- 9380.35 PLN Brutto na umowie o pracę,
- 10153.79 PLN Brutto na b2b

Wnioski po analizie zaprezentowanych wyników:

1. Istotna jest konfiguracja technologii w procesie tworzenia modelu a nie jej ilość.
2. Jeśli model dostał na treningu oferty z wysokimi zarobkami to wyniki będą trochę zawyżone.
3. Rodzaj zastosowanej technologii ma wpływ na większe lub mniejsze wynagrodzenie.

Myślę, że udało mi się stworzyć przykładowy model przewidujący zarobki w zależności od innych zmiennych w ofercie pracy. Wśród modeli ten okazał się mieć najmniej błędów, co jest satysfakcjonujące, a przewidywane wartości są, powiedzmy jasno, 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