Business Intelligence Case Challenge

Dominik Dawiec

Uniwersytet Ekonomiczny w Krakowie

Czerwiec 2023

Spis treści

Wstęp

W ramach II Edycji konkursu „Business Intelligence Case Challenge”, zorganizowanego przez Koło Naukowe Analizy Danych na Uniwersytecie Ekonomicznym w Krakowie, autor został postawiony przed zadaniem prognozowania wysokości rocznego wynagrodzenia zasadniczego członków zarządów wybranych spółek notowanych na Giełdzie Papierów Wartościowych w 2021 roku. Głównym celem autora było wykorzystanie swoich umiejętności w analizie danych w celu dostarczenia wiarygodnych prognoz dotyczących wynagrodzeń. Całość projektu została zrealizowana z wykorzystaniem języka Python.

W projekcie wykorzystano metodologię **CRISP-DM** (*ang. The Cross-industry standard process for data mining*). Jest to powszechnie akceptowana i skuteczna metodyka analizy danych, która zapewnia efektywne prowadzenie projektów. Metodologia ta obejmuje szereg kluczowych etapów, w tym:

1. **zrozumienie uwarunkowań biznesowych**: zapoznanie się z celami i wymaganiami biznesowymi, aby zidentyfikować problemy, na które analiza danych ma odpowiedzieć,
2. **zrozumienie danych**: zbieranie informacji o dostępnych danych, ich strukturze, jakości i związanych z nimi problemach,
3. **przygotowanie danych**: czyszczenie, transformacja i integracja danych w celu stworzenia spójnego i gotowego do analizy zbioru danych,
4. **modelowanie**: tworzenie modeli analitycznych, które pomogą w odpowiedzi na pytania biznesowe i rozwiązaniu problemów,
5. **ewaluacja**: ocena skuteczności modeli i ich zgodności z oczekiwaniami biznesowymi,
6. **wdrożenie**: wprowadzenie wyników analizy danych do rzeczywistej działalności biznesowej i monitorowanie ich efektywności.

Ta kompleksowa metodyka pozwala na systematyczne podejście do analizy danych, zapewniając klarowną ścieżkę od zrozumienia potrzeb biznesowych do wdrożenia praktycznych rozwiązań tym samym każdy z jej etapów stanowi osobny rozdział niniejszej pracy.

1. Zrozumienie uwarunkowań biznesowych

Pierwszy etap obejmuje zrozumienie uwarunkowań biznesowych, tym samym w niniejszym etapie warto pochylić się nad poszczególnymi komponentami konkursu, które obejmują:

- **zadanie konkursowe**: przeprowadzenie predykcji wysokości rocznego wynagrodzenia zasadniczego członków zarządów wybranych spółek notowanych na Giełdzie Papierów Wartościowych w 2021 roku,
- **dane**: przesłane dane dotyczące wysokości rocznego wynagrodzenia zasadniczego członków zarządów dzielące się na zbiór uczący zawierający 461 rekordów oraz zbiór testowy zawierający 120 rekordów,
- **zmienna prognozowana**: wynagrodzenie podstawowe za rok 2021,
- **predyktory**: kolumny Funkcja, Obcokrajowiec, Płeć, Makrosektor, WIG, Skarb Państwa, Spółka rodzinna, Zatrudnienie, Komitet ds. wynagrodzeń w radzie nadzorczej, ROE, ROA, Zysk netto, Wartość rynkowa, EPS.

Dodatkowo, organizatorzy konkursu poinformowali uczestników, że nie jest konieczne wykorzystanie wszystkich dostępnych zmiennych, a także zezwolili na dodatkowe czyszczenie danych. Ponadto, autor niniejszej pracy wysłał zapytanie mailowe dotyczące możliwości wykorzystania inżynierii cech w projekcie na co otrzymał zgodę.

2. Zrozumienie danych

W pierwszej kolejności dokonano importu niezbędnych bibliotek, a następnie przeprowadzono proces wczytania danych z otrzymanego pliku Excel do dwóch ramek danych, nazwanych kolejno training_data oraz validation_data (Listing 1). Następnie sprawdzono jak wygląda przykładowe pięć pierwszych wierszy ramki danych training_data (Rysunek 1).

Listing 1 Import bibliotek oraz ramek danych w Pythonie

```python
# zaimportowanie podstawowych bibliotek
import pandas as pd
import numpy as np
import pandas_profiling

# zaimportowanie bibliotek do wizualizacji danych
import matplotlib.pyplot as plt
import plotly.express as px

# zaimportowanie bibliotek do inżynierii cech
from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import StandardScaler

# zaimportowanie bibliotek do budowy modeli uczenia maszynowego
from sklearn.ensemble import ExtraTreesRegressor, RandomForestRegressor, GradientBoostingRegressor
from sklearn.feature_selection import SelectFromModel
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import xgboost as xgb
from sklearn.model_selection import cross_val_score
from pycaret.regression import *

# umieszczenie danych z pliku excel w ramkach danych
training_data = pd.read_excel('Arkusz_z_danymi_BICC.xlsx', sheet_name='Zbiór uczący')
validation_data = pd.read_excel('Arkusz_z_danymi_BICC.xlsx', sheet_name='Zbiór testowy')

# sprawdzenie jak wygląda ramka danych
training_data.head()
```

Źródło: opracowanie własne

Rysunek 1 Pierwsze pięć wierszy ramki danych training_data

	WYNAGRODZENIE PODSTAWOWE	Funkcja	OBCOKRAJOWIEC	PŁEĆ	MAKROSEKTOR	WIG	Skarb Państwa	Spółka rodzinna	ZATRUDNIENIE\n2021 (Etaty)	KOMITET DS. WYNAGRODZEŃ W RADZIE NADZORCZEJ	ROE	ROA	Zysk netto (tys.PLN)	wartość rynkowa (mln PLN)	EPS
0	60000.00	członek zarządu	NIE	M	Handel i Usługi	Inny	NIE	TAK	1335	NIE	0.1257	0.0458	25737	202.09	3.400348
1	3387978.98	prezes zarządu	NIE	M	Finanse	Inny	NIE	TAK	44	NIE	0.0079	0.0042	3489	86.27	0.093832
2	292368.28	prezes zarządu	NIE	M	Dobra Konsumpcyjne	Inny	NIE	TAK	81	NIE	0.0488	0.0302	3330	30.65	1.233333
3	100533.00	członek zarządu	NIE	K	Handel i Usługi	mWIG 40	NIE	NIE	235	TAK	0.4024	0.2715	141265	2548.32	19.346645
4	733188.00	członek zarządu	NIE	K	Dobra Konsumpcyjne	Inny	NIE	NIE	1791	TAK	1.9754	0.1575	406580	5197.30	39.583941

Źródło: opracowanie własne

Po przeglądzie ramki danych, zauważono, że nazewnictwo kolumn nie jest jednolite. W celu zapewnienia spójności danych oraz ułatwienia przyszłych analiz, dokonano korekty nazw wybranych kolumn (Listing 2).

Listing 2 Ujednolicenie nazewnictwa kolumn dla obu ramek danych

```python
# Zdefiniowanie nowcych nazw dla wybranych kolumn
new_columns = {
    'WYNAGRODZENIE PODSTAWOWE': 'wynagrodzenie_podstawowe',
    'WYNAGRODZENIE PODSTAWOWE (PROGNOZY)': 'wynagrodzenie_podstawowe',
    'Funkcja': 'funkcja', 'OBCOKRAJOWIEC': 'obcokrajowiec',
    'PŁEĆ': 'plec', 'MAKROSEKTOR': 'makrosektor', 'WIG ': 'WIG',
    'Skarb Państwa': 'skarb_panstwa', 'Spółka rodzinna': 'spolka_rodzinna',
    'ZATRUDNIENIE\n2021 (Etaty)': 'zatrudnienie_etaty_2021',
    'KOMITET DS. WYNAGRODZEŃ W RADZIE NADZORCZEJ':   'komitet_ds_wynagrozen_w_radzie_nadzorczej',
    'ROE': 'ROE', 'ROA': 'ROA', 'Zysk netto (tys.PLN)': 'zysk_netto_tys_PLN',
    'wartość rynkowa (mln PLN)': 'wartosc_rynkowa_mln_PLN', 'EPS': 'EPS'}

# Zmiana nazw kolumn w ramce danych "training_data"
training_data = training_data.rename(columns=new_columns)

# Zmiana nazw kolumn w ramce danych "validation_data"
validation_data = validation_data.rename(columns=new_columns)
```

Źródło: opracowanie własne

Następnie, obie ramki danych zostały poddane analizie za pomocą biblioteki pandas profiling. Biblioteka pandas profiling umożliwia generowanie raportów dotyczących danych (Rysunek 2) w celu uzyskania szczegółowego zrozumienia ich struktury i charakterystyk takich jak rozkład wartości w poszczególnych kolumnach, statystyki opisowe, brakujące wartości, wartości unikalne oraz inne istotne cechy danych co pozwala na skrócenie czasu analizy danych oraz dostarczenie przydatnych informacji (Listing 3).

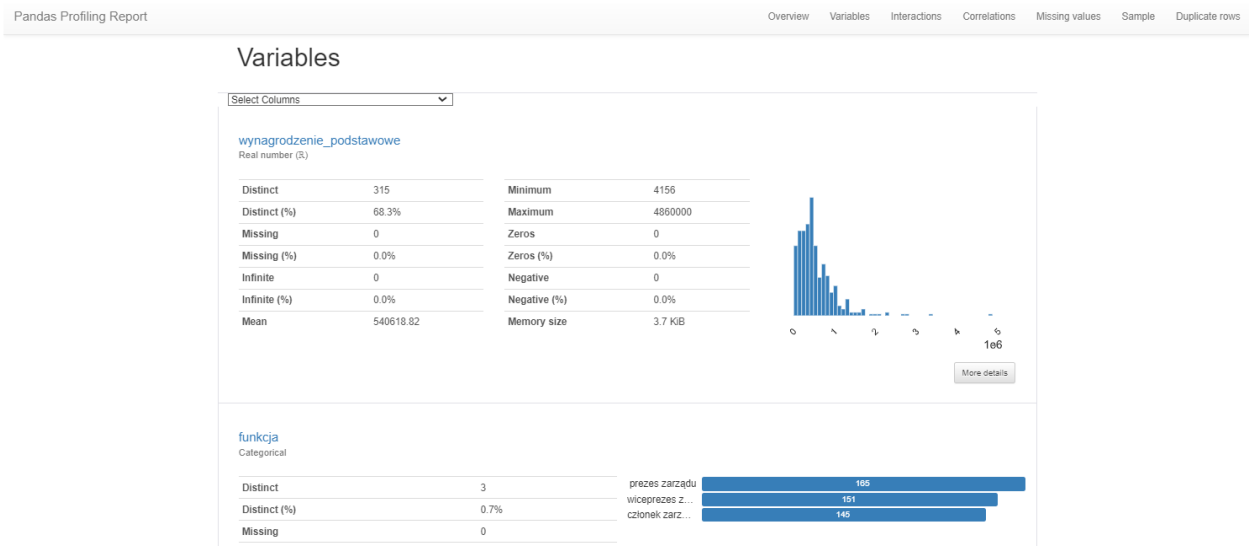
Listing 3 Stworzenie raportu z wykorzystaniem Pandas Profiling

```python
# Tworzenie raportu pandas profiling dla ramki danych training_data
profile = training_data.profile_report(title="Pandas Profiling Report")

# Wyświetlanie raportu w notebooku
profile.to_notebook_iframe()
```

Źródło: opracowanie własne

Rysunek 2 Raport wygenerowany w Pandas Profiling



Źródło: opracowanie własne

Analizując uzyskane raporty, autor dostrzegł kilka istotnych kwestii, na które warto zwrócić uwagę. Obejmują one między innymi:

- **wartości skrajne**: dane charakteryzują się wieloma wartościami skrajnymi, które mogą mieć istotny wpływ na wyniki analizy oraz modelu uczenia maszynowego tym samym konieczne jest uwzględnienie ich w kolejnych krokach analizy, aby dokładnie zrozumieć ich wpływ i ewentualne konsekwencje,
- **brakujące dane**: w kolumnach „zatrudnienie_etaty_2021” oraz „komitet ds wynagrozen w radzie nadzorczej” występuje część rekordów oznaczonych jako „BD” (Brak Danych),
- **uspójnienie kategorycznych wartości**: kolumna „komitet ds wynagrozen w radzie nadzorczej” zawiera wartości „NIE” oraz „NIE ”, które wymagają uspójnienia,
- **nietypowo niskie wartości**: między innymi w kolumnie „zatrudnienie_etaty_2021” zauważono, że niektóre wartości są bardzo niskie zawierając przykładowo wartości mniejsze od 10, co jest nietypowe w analizie spółek notowanych na giełdzie tym samym konieczne jest zbadanie przyczyn tych wartości i ich potencjalnego wpływu na wyniki,
- **zduplikowane wartości**: w ramach zbioru danych zidentyfikowano kilkadziesiąt zduplikowanych wartości w ramach wybranych kolumnach co może sugerować istnienie kilkudziesięciu przedsiębiorstw w ramach całego zbioru danych.

3. Przygotowanie danych

Na początku niniejszego etapu autor wprowadził dodatkową kolumnę o nazwie 'training_validation_column'. W ramce danych training_data, kolumna została wypełniona wartościami 1, natomiast w ramce danych validation_data przyjęła wartości 0. To działanie miało na celu umożliwienie połączenia obu ramek danych co pozwala na wykonanie pełnego procesu inżynierii cech i modelowania na jednym, połączonym zbiorze danych a następnie podzielenie go na podstawie wartości w kolumnie 'training_validation_column' po zakończeniu procesu przekształcania danych (Listing 4).

Listing 4 Utworzenie dodatkowej kolumny w obu ramkach danych

```python
# Dodanie kolumny 'training_validation_column' do training_data
training_data.insert(0, 'training_validation_column', 1)

# Dodanie kolumny 'training_validation_column' do validation_data
validation_data.insert(0, 'training_validation_column', 0)

# Połączenie training_data i validation_data w jeden DataFrame
training_data = pd.concat([training_data, validation_data], ignore_index=True)
```

Źródło: opracowanie własne

W niektórych sytuacjach informacja o występowaniu braków danych w określonych wierszach może mieć istotne znaczenie i może być wykorzystana jako ważna cecha w procesie inżynierii cech. Zamiast traktować braki danych jako przypadkowe lub nieistotne, można je potraktować jako cenne wskaźniki lub cechy, które wpływają na proces modelowania i analizy danych. W ramach projektu autor zdecydował się utworzyć dwie nowe kolumny „jedna_wartosc_BD” oraz „dwie_wartosci_BD”. Mają one na celu zidentyfikowanie wierszy, w których występują braki danych (BD) w określonych kolumnach pozwalając odkryć potencjalne wzorce, które mają istotne znaczenie dla analizowanego problemu (Listing 5).

Wykorzystanie informacji o brakach danych jako cechy pozwala wzbogacić zbiór danych i rozszerzyć możliwości modelowania. Dzięki temu można osiągnąć lepsze wyniki predykcyjne oraz bardziej precyzyjne analizy. Ta dodatkowa informacja może dostarczyć cennych wskazówek dla modelu uczenia maszynowego, pomagając mu w podejmowaniu bardziej trafnych decyzji na podstawie występowania braków danych w poszczególnych wierszach.

Listing 5 Stworzenie kolumn informujących czy dany wiersz zawiera braki danych

```python
# Tworzenie kolumny 'jedna_wartosc_BD'
training_data['jedna_wartosc_BD'] = (training_data['zatrudnienie_etaty_2021'] == 'BD') |
(training_data['komitet_ds_wynagrozen_w_radzie_nadzorczej'] == 'BD')
training_data['jedna_wartosc_BD'] = training_data['jedna_wartosc_BD'].astype(int)

# Tworzenie kolumny 'dwie_wartosci_BD'
training_data['dwie_wartosci_BD'] = (training_data['zatrudnienie_etaty_2021'] == 'BD') &
(training_data['komitet_ds_wynagrozen_w_radzie_nadzorczej'] == 'BD')
training_data['dwie_wartosci_BD'] = training_data['dwie_wartosci_BD'].astype(int)
```

Źródło: opracowanie własne

W kolejnym kroku ujednolicono kolumnę „komitet_ds_wynagrozen_w_radzie_nadzorczej" poprzez zamienienie wartości „NIE " na wartość „NIE" (Listing 6).

Listing 6 Ujednolicenie kolumny komitet_ds_wynagrozen_w_radzie_nadzorczej

```python
# Zamiana wartości "NIE " na "NIE" w kolumnie "komitet_ds_wynagrozen_w_radzie_nadzorczej"
training_data['komitet_ds_wynagrozen_w_radzie_nadzorczej'] =
training_data['komitet_ds_wynagrozen_w_radzie_nadzorczej'].replace('NIE ', 'NIE')
```

Źródło: opracowanie własne

Następnie dokonano wypełnienia wartości „BD" (brak danych) za pomocą imputacji (Listing 7). W przypadku kolumny „zatrudnienie_etaty_2021" zastosowano imputację medianą, natomiast dla kolumny „komitet_ds_wynagrozen_w_radzie_nadzorczej" wykorzystano najczęściej występującą wartość. Autor podjął również próbę imputacji brakujących wartości poprzez próbę ich zaprognozowania z wykorzystaniem między innymi algorytmu K najbliższych sąsiadów (KNN), jednak w wyniku przeprowadzonych testów metody te nie spełniły oczekiwań.

Listing 7 Imputacja brakujących danych

```python
training_data = training_data.replace('BD', np.nan) # zamiana wartości BD na nan

# Uzupełnienie wartości NaN w kolumnie liczbowej średnią
mean_value = training_data['zatrudnienie_etaty_2021'].mean()
training_data['zatrudnienie_etaty_2021'].fillna(mean_value, inplace=True)
# Uzupełnienie wartości NaN w kolumnie kategorycznej najczęściej występującą wartością
most_frequent_value = training_data['komitet_ds_wynagrozen_w_radzie_nadzorczej'].mode()[0]
training_data['komitet_ds_wynagrozen_w_radzie_nadzorczej'].fillna(most_frequent_value, inplace=True)
```

Źródło: opracowanie własne

Inżynieria cech to proces transformacji i tworzenia nowych cech na podstawie istniejących danych w celu ulepszenia analizy i modelowania. W kontekście projektu, autor przeprowadził inżynierię cech poprzez stworzenie kilkunastu nowych kolumn, które mają na celu dostarczenie dodatkowych informacji i wskaźników dla analizowanego problemu (Listing 8). Stworzone przez autora kolumny obejmują:

- „**kapital_wlasny**”: która została wyliczona na podstawie wzoru na zwrot z kapitału własnego (ROE), korzystając z kolumn „zysk_netto_tys_PLN” i „ROE”, informuje ona o wartości kapitału własnego spółki,
- „**aktywa**”: obliczana na podstawie wzoru na zwrot z aktywów (ROA), wykorzystując kolumny „zysk_netto_tys_PLN” oraz „ROA”, ta cecha dostarcza informacji o wartości aktywów spółki i może być przydatna w analizie,
- „**ilosc_akcji**”: wyliczana na podstawie wzoru na zarobek na akcję (EPS) wykorzystując kolumny „zysk_netto_tys_PLN” oraz „EPS”, ta cecha informuje o ilości akcji spółki,
- „**zysk_strata_netto**”: przyjmuje wartość 1, jeśli spółka osiągnęła zysk netto, a 0 w przypadku straty netto,
- „**wskaznik_oplacalnosci**”: obliczany na podstawie stosunku ROE do ROA, informuje o efektywności wykorzystania aktywów w generowaniu zysku,
- „**ROI**”: (Return on Investment) obliczany na podstawie zysku netto i kapitału własnego,
- „**zysk_netto_na_pracownika**”: informuje o średnim zysku netto przypadającym na jednego pracownika,
- „**liczba_osob_w_zarzadznie**”, „**liczba_wiceprezesow_w_zarzadzie**” i „**liczba_czlonkow_zarzadu_w_zarzadzie**”: informują o liczbie wszystkich osób w zarządzie spółki oraz o liczbie w podziale na wiceprezesów i członków zarządu,
- „**stosunek_zarzadu_do_pracowników**”: informuje o stosunku liczby osób w zarządzie do liczby pracowników,
- „**stosunek_zysku_netto_do_zarzadu**”: stosunek zysku netto przypadającego na jedną osobę w zarządzie,
- „**stosunek_aktywow_do_zarzadu**”: stosunek wartości aktywów do liczby członków zarządu,
- „**stosunek_kapitalu_wlasnego_do_zarzadu**”: stosunek kapitału własnego do liczby członków zarządu.

Listing 8 Proces inżynierii cech poprzez stworzenie nowych kolumn

Python

```
# Odtwarzanie kolumny kapital_wlasny na podstawie wzoru na ROE
training_data['kapital_wlasny'] = (training_data['zysk_netto_tys_PLN'] / (training_data['ROE'] /
100)).round()

# Odtwarzanie kolumny aktywa na podstawie wzoru na ROA
training_data['aktywa'] = (training_data['zysk_netto_tys_PLN'] / (training_data['ROA'] / 100)).round()

# Obliczanie ilości akcji na podstawie EPS i zysku netto
training_data['ilosc_akcji'] = (training_data['zysk_netto_tys_PLN'] / training_data['EPS']).round()

# kolumna informująca o zysku lub stracie netto
training_data['zysk_strata_netto'] = 0
training_data.loc[training_data['zysk_netto_tys_PLN'] > 0, 'zysk_strata_netto'] = 1

# kolumna pokazująca stosunek zysku netto do wartości rynkowej
training_data['zysk_netto_do_wartosci_rynkowej'] = training_data['zysk_netto_tys_PLN'] /
training_data['wartosc_rynkowa_mln_PLN']

# wskaźnik opłacalności na podstawie stosunku ROE do ROA
training_data['wskaznik_oplacalnosci'] = training_data['ROE'] / training_data['ROA']

# wskaźnik zwrotu z inwestycji (ROI) na podstawie zysk_netto_tys_PLN oraz kapital_wlasny
training_data['ROI'] = (training_data['zysk_netto_tys_PLN'] / training_data['kapital_wlasny'])*100

# kolumna pokazująca zysk netto na pracownika
training_data['zysk_netto_na_pracownika'] = training_data['zysk_netto_tys_PLN'] /
training_data['zatrudnienie_etaty_2021']
training_data.loc[training_data['zatrudnienie_etaty_2021'] == 0, 'zysk_netto_na_pracownika'] = 0

#kolumna infomrująca ile osób jest w zarządzie danego przedsiębiorstwa w tym ile osób jest w roli
wiceprezesa oraz członka zarządu
training_data['liczba_osob_w_zarzadznie'] = training_data.groupby(['zatrudnienie_etaty_2021',
'zysk_netto_tys_PLN',
'wartosc_rynkowa_mln_PLN'])['zatrudnienie_etaty_2021'].transform('count').fillna(1)
training_data['liczba_wiceprezesow_w_zarzadzie'] = training_data.groupby(['zatrudnienie_etaty_2021',
'zysk_netto_tys_PLN', 'wartosc_rynkowa_mln_PLN'])['funkcja'].transform(lambda x: x[x == 'wiceprezes
zarządu'].count())
training_data['liczba_czlonkow_zarzadu_w_zarzadzie'] =
training_data.groupby(['zatrudnienie_etaty_2021', 'zysk_netto_tys_PLN',
'wartosc_rynkowa_mln_PLN'])['funkcja'].transform(lambda x: x[x == 'członek zarządu'].count())
training_data['liczba_osob_w_zarzadznie'] = training_data['liczba_osob_w_zarzadznie'].fillna(0)
training_data['liczba_wiceprezesow_w_zarzadzie'] =
training_data['liczba_wiceprezesow_w_zarzadzie'].fillna(0)
training_data['liczba_czlonkow_zarzadu_w_zarzadzie'] =
training_data['liczba_czlonkow_zarzadu_w_zarzadzie'].fillna(0)

# kolumna inforująca o stosunku liczby członków zarządu do liczby pracowników
training_data['stosunek_zarzadu_do_pracowników'] = training_data['zatrudnienie_etaty_2021'] /
training_data['liczba_osob_w_zarzadznie']

# kolumna inforująca o stosunku zysku netto do liczby członków zarządu
training_data['stosunek_zysku_netto_do_zarzadu'] = training_data['zysk_netto_tys_PLN'] /
training_data['liczba_osob_w_zarzadznie']

# kolumna inforująca o stosunku aktywów do liczby członków zarządu
training_data['stosunek_aktywow_do_zarzadu'] = training_data['aktywa'] /
training_data['liczba_osob_w_zarzadznie']

# kolumna inforująca o stosunku kapitału własnego do liczby członków zarządu
training_data['stosunek_kapitalu_wlasnego_do_zarzadu'] = training_data['kapital_wlasny'] /
training_data['liczba_osob_w_zarzadznie']
```

Źródło: opracowanie własne

Następnie autor skorzystał z biblioteki scikit-learn w celu przeprowadzenia transformacji cech za pomocą PolynomialFeatures (Listing 9). PolynomialFeatures jest narzędziem wykorzystywanym do generowania nowych cech, które są kombinacjami wielomianowymi wybranych cech. Przykładowo, jeśli mamy cechy x_1 i x_2, transformacja do drugiego stopnia za pomocą PolynomialFeatures spowoduje wygenerowanie nowych cech, takich jak $(x_1)^2$, $x_1 * x_2$ i $(x_2)^2$. Dzięki temu możemy uwzględnić nieliniowe zależności między cechami, które mogą mieć istotny wpływ na wyniki analizy.

Stopień, do jakiego można przemnożyć cechy za pomocą PolynomialFeatures, jest elastyczny i zależy od potrzeb i złożoności problemu. Można wybrać stopień równy 2, jak zdecydował się autor w tym przypadku, ale również można eksperymentować z wyższymi stopniami, takimi jak 3, 4 itp. Warto jednak pamiętać, że wyższe stopnie przemnożenia cech mogą prowadzić do wzrostu wymiarowości danych i zwiększenia złożoności obliczeniowej. Dlatego ważne jest umiejętne dostosowanie stopnia przemnożenia do problemu i dostępnych zasobów obliczeniowych.

Listing 9 Wykorzystanie PolynomialFeatures do inżynierii cech

Python

```
# Wybieranie kolumn do transformacji
columns_to_transform = ['zatrudnienie_etaty_2021', 'ROE', 'ROA', 'zysk_netto_tys_PLN',
'wartosc_rynkowa_mln_PLN', 'EPS', 'kapital_wlasny', 'aktywa', 'ilosc_akcji',
'zysk_netto_do_wartosci_rynkowej', 'wskaznik_oplacalnosci', 'ROI', 'zysk_netto_na_pracownika',
'liczba_osob_w_zarzadznie', 'liczba_wiceprezesow_w_zarzadzie', 'liczba_czlonkow_zarzadu_w_zarzadzie',
'stosunek_zarzadu_do_pracowników', 'stosunek_zysku_netto_do_zarzadu']

# Tworzenie obiektu PolynomialFeatures
poly_features = PolynomialFeatures(degree=2, include_bias=False)

# Wykonanie transformacji dla wybranych kolumn
transformed_features = poly_features.fit_transform(training_data[columns_to_transform])

# Tworzenie nazw dla nowych cech
new_feature_names = poly_features.get_feature_names_out(columns_to_transform)

# Tworzenie ramki danych dla nowych cech
transformed_df = pd.DataFrame(transformed_features, columns=new_feature_names)

training_data = training_data.drop(columns=columns_to_transform).reset_index(drop=True)

# Dodawanie nowych cech do oryginalnej ramki danych
training_data = pd.concat([training_data, transformed_df], axis=1)
```

Źródło: opracowanie własne

4. Modelowanie

Przed budową efektywnych modeli uczenia maszynowego istotne jest zrozumienie różnic w skalach i rozkładach poszczególnych cech. Cechy o zróżnicowanych skalach mogą wpływać na działanie algorytmów uczących, dlatego konieczne jest zastosowanie procesu ich skalowania. Skalowanie cech polega na przekształceniu wartości cech w taki sposób, aby miały one zbliżony zakres lub rozkład. Jedną z popularnych technik skalowania jest standaryzacja, która ma na celu przeskalowanie cech tak, aby miały średnią wartość równą 0 i odchylenie standardowe równe 1. Standaryzacja jest odpowiednią metodą skalowania cech w przypadku, gdy dane mają różne rozkłady oraz skrajne wartości, dlatego autor postanowił wybrać tę metodę (Listing 10).

Listing 10 Przeskalowanie wybranych zmiennych z wykorzystaniem standaryzacji

```python
# Wybór kolumn liczbowych do standaryzacji (z wyjątkiem 'wynagrodzenie_podstawowe')
numeric_columns = ['zatrudnienie_etaty_2021', 'ROE', 'ROA', 'zysk_netto_tys_PLN',
'wartosc_rynkowa_mln_PLN', 'EPS', 'kapital_wlasny', 'aktywa', 'ilosc_akcji',
'zysk_netto_do_wartosci_rynkowej', 'wskaznik_oplacalnosci', 'ROI', 'zysk_netto_na_pracownika',
'stosunek_zarzadu_do_pracowników', 'stosunek_zysku_netto_do_zarzadu', 'liczba_osob_w_zarzadznie',
'liczba_wiceprezesow_w_zarzadzie']

# Standaryzacja danych
scaler = StandardScaler()
training_data[numeric_columns] = scaler.fit_transform(training_data[numeric_columns])
```

Źródło: opracowanie własne

Metoda kodowania danych, znana jako „one-hot encoding” lub kodowanie jednoznaczne, jest szeroko stosowana w analizie danych i uczeniu maszynowym. W kontekście niniejszego artykułu naukowego, kolumny funkcja, makrosektor, WIG oraz płeć zostały poddane kodowaniu przy użyciu tej techniki (Listing 11). One-hot encoding polega na przekształceniu kategorialnych danych, takich jak kolumny funkcja, makrosektor, WIG i płeć, na postać numeryczną, która jest bardziej odpowiednia dla algorytmów uczenia maszynowego. Proces ten polega na utworzeniu nowych binarnych zmiennych dla każdej kategorii w danej kolumnie. Każda zmienna przyjmuje wartość 0 lub 1, w zależności od tego, czy dana obserwacja należy do danej kategorii.

Listing 11 Kodowanie wybranych zmiennych z wykorzystaniem one-hot encoding

```python
# kodowanie one-hot encoding dla wybranych kolumn
encoded_data = pd.get_dummies(training_data, columns=['funkcja', 'plec', 'makrosektor', 'WIG'])
training_data = encoded_data
```

Źródło: opracowanie własne

W przypadku pozostałych kolumn, które zawierają wartości binarne TAK/NIE, zastosowano technikę kodowania danych przy użyciu mapowania (Listing 12). Ta metoda kodowania umożliwia przypisanie wartości numerycznych do kategorii TAK i NIE, aby reprezentować je w sposób odpowiedni dla analizy danych i uczenia maszynowego.

Listing 12 Kodowanie wybranych zmiennych z wykorzystaniem mapowania

```python
# Tworzenie mapy zamiany wartości
mapping = {'TAK': 1, 'NIE': 0}

# Zastosowanie mapowania dla odpowiednich kolumn
training_data['obcokrajowiec'] = training_data['obcokrajowiec'].map(mapping)
training_data['skarb_panstwa'] = training_data['skarb_panstwa'].map(mapping)
training_data['spolka_rodzinna'] = training_data['spolka_rodzinna'].map(mapping)
training_data['komitet_ds_wynagrozen_w_radzie_nadzorczej'] =
training_data['komitet_ds_wynagrozen_w_radzie_nadzorczej'].map(mapping)
```

Źródło: opracowanie własne

Na zakończenie procesu, dane zostały podzielone z powrotem na dwie odrębne ramki danych, training_data oraz validation_data (Listing 13).

Listing 13 Podział ramki danych na training_data i validation_data

```python
# Podział DataFrame merged_data na training_data i validation_data

validation_data = training_data.loc[training_data['training_validation_column'] == 0].copy()
training_data = training_data.loc[training_data['training_validation_column'] == 1].copy()

# Usunięcie kolumny 'training_validation_column' z training_data i validation_data
training_data.drop('training_validation_column', axis=1, inplace=True)
validation_data.drop('training_validation_column', axis=1, inplace=True)
```

Źródło: opracowanie własne

Następnie autor postanowił skorzystać z biblioteki PyCaret, która zapewnia możliwość Auto ML (Listing 14). AutoML to podejście, które umożliwia automatyzację procesu tworzenia

modeli uczenia maszynowego. Autor wykorzystał tabelę z wynikami modelowania (Rysunek 3) do wyboru najlepszych modeli, które poradziły sobie z prognozowaniem wynagrodzenia podstawowego na podstawie posiadanych danych, które następnie zostały napisane ręcznie i poddane procesowi ewaluacji w kolejnym etapie.

Listing 14 AutoML z wykorzystaniem biblioteki PyCaret

```python
exp_reg = setup(data=training_data, target='wynagrodzenie_podstawowe')

best_model = compare_models(round=2)  # Ustawienie zaokrąglenia wyników
```

Źródło: opracowanie własne

Rysunek 3 Część tabeli pokazującej wyniki AutoML w PyCaret

	Model	MAE	MSE	RMSE	R2
rf	Random Forest Regressor	221056.48	165736722824.56	357176.27	0.44
gbr	Gradient Boosting Regressor	217615.01	168046660006.69	366202.73	0.40
et	Extra Trees Regressor	225385.53	174551966179.38	370608.87	0.39
xgboost	Extreme Gradient Boosting	217677.24	187110047230.24	376755.31	0.39
ada	AdaBoost Regressor	270848.31	203988800334.61	407660.96	0.27
knn	K Neighbors Regressor	286025.01	220502063619.50	443795.26	0.07

Źródło: opracowanie własne

Najlepsze wyniki w prognozowaniu wynagrodzenia na podstawie dostępnych danych osiągnęły modele Gradient Boosting Regressor, Extreme Gradient Boosting (XGBoost), Random Forest Regressor i Extra Trees Regressor. W związku z tym, wszystkie te modele zostaną uwzględnione w kolejnym etapie projektu. Wszystkie wymienione modele należą do rodziny metod zespołowych, które mają na celu połączenie wielu słabszych modeli w celu uzyskania bardziej efektywnego i dokładnego modelu.

Gradient Boosting Regressor i XGBoost są oparte na technice gradient boosting, która polega na sekwencyjnym dodawaniu słabszych modeli, zazwyczaj drzew decyzyjnych, do zespołu. Każdy nowy model jest uczony w celu skorygowania błędów poprzednich modeli, co prowadzi do sukcesywnego poprawiania jakości predykcji.

Random Forest Regressor i Extra Trees Regressor są oparte na metodzie lasów losowych. Ta technika polega na tworzeniu wielu drzew decyzyjnych, które są trenowane na różnych podzbiorach danych i z wykorzystaniem różnych losowych cech. Ostateczny wynik jest obliczany na podstawie średniej lub głosowania modeli w przypadku regresji.

5. Ewaluacja

Jak wspomniano w poprzednim rozdziale w tym etapie zastosowano modele takie jak Gradient Boosting Regressor, Extreme Gradient Boosting, Random Forest Regressor i Extra Trees Regressor, w celu przewidywania wynagrodzenia podstawowego na podstawie dostępnych cech. Każdy z tych modeli znacznie rozbudowano poprzez między innymi zastosowanie metody SelectFromModel, która umożliwiła wybór najważniejszych cech na podstawie ich współczynnika ważności oraz zdefiniowano siatkę parametrów obejmującą różne kombinacje w celu znalezienia optymalnych wartości.

Następnie dokonano oceny modeli za pomocą różnych metryk, takich jak błąd średni absolutny (MAE), błąd średniokwadratowy (MSE), pierwiastek błędu średniokwadratowego (RMSE), współczynnik determinacji (R2) oraz procentowy błąd średni absolutny (MAPE). Te metryki stanowiły podstawę do wyboru najlepszego modelu spośród badanych (Tabela 1).

Tabela 1 Wyniki modeli uczenia maszynowego

Model/Metryka	MAE	MSE	RMSE	R2	MAPE
Extra Trees Regressor	14465.37	1618514928.89	40230.77	0.99	4.07
Gradient Boosting Regressor	75811.73	12090214599.57	109955.51	0.95	32.23
Extreme Gradient Boosting	81267.42	14324116097.73	119683.40	0.94	35.43
Random Forest Regressor	73995.17	18385763606.37	135594.11	0.92	43.24

Źródło: opracowanie własne

Poszczególne metryki można przeanalizować w następujący sposób:

- MAE (mean absolute error) jest to średni bezwzględny błąd predykcji modelu. Im niższa wartość, tym lepsza jakość predykcji,
- MSE (mean squared error) jest to średni kwadrat błędu predykcji modelu. Im niższa wartość, tym lepsza jakość predykcji,
- RMSE (root mean squared error) jest to pierwiastek z MSE, co daje nam miarę błędu w tych samych jednostkach co zmienna objaśniana,
- R2 (R-squared) jest to współczynnik determinacji, który wskazuje, jak bardzo zmienność zmiennej objaśnianej jest wyjaśniana przez model. Im bliżej wartości 1, tym lepiej model przewiduje zależność.

- MAPE (mean absolute percentage error) jest to średni bezwzględny błąd procentowy predykcji modelu. Im niższa wartość, tym lepsza jakość predykcji.

Analizując tabelę najlepiej poradził sobie model Extra Trees Regressor, który opiera się na drzewach decyzyjnych i stosuje zasadę losowego poddrzewa a sposób jego budowy przez autora przedstawia Listing 15.

Listing 15 Model Extra Trees Regressor

Python

```python
# Wczytanie danych treningowych
df = training_data

# Podział na cechy (X) i etykiety (y)
X = df.drop('wynagrodzenie_podstawowe', axis=1)
y = df['wynagrodzenie_podstawowe']

# Utworzenie modelu ExtraTreesRegressor
etr = ExtraTreesRegressor()

# Wybór zmiennych na podstawie współczynnika ważności cech
sfm = SelectFromModel(etr)
X_selected = sfm.fit_transform(X, y)

# Pobranie indeksów wybranych zmiennych
selected_feature_indexes = sfm.get_support(indices=True)

# Pobranie nazw wybranych zmiennych
selected_features = X.columns[selected_feature_indexes]

# Definicja siatki parametrów do przeszukania
param_grid = {
    'n_estimators': [100, 200, 300, 400],
    'max_depth': [None, 5, 10, 15],
    'min_samples_split': [2, 5, 10, 15]}

# Utworzenie obiektu GridSearchCV z uwzględnieniem wybranych zmiennych
grid_search = GridSearchCV(estimator=etr, param_grid=param_grid, cv=5)
grid_search.fit(X_selected, y)

# Wybór optymalnych parametrów
best_params = grid_search.best_params_

# Wyświetlanie najlepszych parametrów
print("\nBest parameters:", best_params)

# Utworzenie modelu ExtraTreesRegressor z optymalnymi parametrami
etr_best = ExtraTreesRegressor(**best_params)

# Trenowanie modelu na wybranych zmiennych
etr_best.fit(X_selected, y)

# Przewidywanie na danych treningowych
y_pred = etr_best.predict(X_selected)
```

```
# Obliczanie metryk
mae = mean_absolute_error(y, y_pred)
mse = mean_squared_error(y, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y, y_pred)
mape = np.mean(np.abs((y - y_pred) / y)) * 100

# Wyświetlanie wyników
print("\nMetrics:", best_params)
print("MAE:", mae)
print("MSE:", mse)
print("RMSE:", rmse)
print("R2:", r2)
print("MAPE:", mape)
```

Źródło: opracowanie własne

Opisując powyższy kod w Pythonie realizuje następujące kroki:

- wczytanie danych treningowych oraz podział ich na cechy (X) oraz etykiety (y),
- utworzenie wstępnego modelu ExtraTreesRegressor bez żadnych parametrów,
- wybór zmiennych o największym wpływie poprzez SelectFromModel z wykorzystaniem modelu ExtraTreesRegressor,
- zdefiniowanie siatki parametrów do przeszukania, która zawiera różne kombinacje liczby drzew (n_estimators), maksymalnej głębokości drzew (max_depth) i minimalnej liczby próbek do podziału węzłów (min_samples_split),
- utworzenie obiektu GridSearchCV, który przeszukuje siatkę parametrów dla modelu ExtraTreesRegressor z użyciem walidacji krzyżowej (cv=5),
- dopasowanie obiektu GridSearchCV do wybranych zmiennych i etykiet oraz wybór optymalnych parametrów dla modelu ExtraTreesRegressor na podstawie wyników przeszukiwania,
- utworzenie nowego modelu ExtraTreesRegressor z optymalnymi parametrami,
- trenowanie nowego modelu na wybranych zmiennych i etykietach oraz przewidywanie wyników na danych treningowych,
- obliczanie metryk oceny modelu, takich jak średni błąd bezwzględny (MAE), średni błąd kwadratowy (MSE), pierwiastek średniego błędu kwadratowego (RMSE), współczynnik determinacji (R2) i średni procentowy błąd bezwzględny (MAPE),
- wyświetlanie wyników, w tym optymalnych parametrów i obliczonych metryk.

6. Wdrożenie

Model Extra Trees Regressor został wdrożony poprzez prognozowanie wartości w ramce danych validation_data co przedstawia Listing 16.

Listing 16 Wdrożenie modelu poprzez prognozowanie wartości w validation_data

```python
# Przewidywanie na danych walidacyjnych
X_validation_selected = sfm.transform(validation_data.drop('wynagrodzenie_podstawowe', axis=1))
y_pred_validation = etr_best.predict(X_validation_selected)

# Zapisywanie przewidywanych wartości w kolumnie 'wynagrodzenie_podstawowe' dla danych walidacyjnych
validation_data['wynagrodzenie_podstawowe'] = y_pred_validation

# Wyświetlanie wyników dla danych walidacyjnych
print("Validation Data - Predicted 'wynagrodzenie_podstawowe':")
validation_data
```

Źródło: opracowanie własne

Wyniki prognoz zostały następnie zapisane w formie pliku csv., z którego prognozy zostały przekopiowane do pliku excel otrzymanego przez Koło Naukowe Analizy Danych w celu ich oceny oraz weryfikacji.

Zakończenie

W ramach tego artykułu omówiono proces analizy danych w kontekście konkursu „Business Intelligence Case Challenge”. Autor postawił sobie za cel prognozowanie wysokości rocznego wynagrodzenia zasadniczego członków zarządów wybranych spółek notowanych na Giełdzie Papierów Wartościowych w 2021 roku. Wykorzystano metodologię CRISP-DM, która zapewniła systematyczne podejście do analizy danych.

Przeprowadzenie analizy danych w ramach konkursu było cennym doświadczeniem dla autora. Pozwoliło mu wykorzystać swoje umiejętności w analizie danych. Uzyskane prognozy dotyczące wynagrodzeń stanowią istotną informację dla biznesu i mogą przyczynić się do podejmowania lepszych decyzji w zarządzaniu.

Przyszłość analizy danych i biznesowego wykorzystania informacji jest obiecująca, a konkursy takie jak „Business Intelligence Case Challenge” stanowią doskonałą okazję do rozwijania umiejętności i dzielenia się wiedzą z innymi profesjonalistami.

Spis listingów

Spis rysunków

Spis tabel