|  |  |  |  |
| --- | --- | --- | --- |
| A red and white logo  Description automatically generated | Politechnika Bydgoska im. J. J. Śniadeckich  Wydział Telekomunikacji, Informatyki i Elektrotechniki | | A blue logo with a white background  Description automatically generated |
| **Przedmiot** | Analiza Regresji i Szeregów Czasowych | | |
| **Prowadzący** | dr inż. Damian Ledziński | | |
| **Temat** | JForex cz. 2 | | |
| **Student** | Zuzanna Tarazewicz | **Nr Indeksu** | 116954 |
| **Ocena** |  | **Data Oddania** |  |

Table of Contents

[Cel 3](#_Toc172485153)

[Stworzenie modelu regresji liniowej 3](#_Toc172485154)

[Zaimportowanie odpowiednich bibliotek 3](#_Toc172485155)

[Zaimportowanie danych i podzielenie zbioru 3](#_Toc172485156)

[Wybór modeli 4](#_Toc172485157)

[Pętla ucząca 4](#_Toc172485158)

[Wykonanie 4](#_Toc172485159)

[Wynik 5](#_Toc172485160)

[Testowanie najlepszego modelu i wizualizacja 5](#_Toc172485161)

[Testowanie modelu 5](#_Toc172485162)

[Wizualizacja 5](#_Toc172485163)

[Wnioski 6](#_Toc172485164)

# Cel

Celem danego laboratorium jest stworzenie modelu regresji liniowej na wybranym zbiorze danych. W tym przypadku jest to popularny zbiór danych *California Housing* dostępny w bibliotece *scikit learn*.

# Stworzenie modelu regresji liniowej

## Zaimportowanie odpowiednich bibliotek

from sklearn.datasets import fetch\_california\_housing

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import Ridge, LinearRegression

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

import pandas as pd

import matplotlib.pyplot as plt

import pickle

Aby utworzyć odpowiedni model potrzebne są następujące biblioteki:

* Scikit Learn – do pobrania zbioru danych, utworzenia modeli i użycia metryk sprawdzających jakość modelu
* Pandas – do użycia zbioru jako DataFrame i Seria
* Matplotlib – do wizualizacji jakości utworzonego modelu
* Pickle – do zapisania danego modelu.

## Zaimportowanie danych i podzielenie zbioru

cal = fetch\_california\_housing()

X = pd.DataFrame(cal.data, columns=cal.feature\_names)

y = pd.Series(cal.target)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=.2, random\_state=42)

Dane zostały zaimportowane pod zmienną cal. Następnie nastąpił podział na podzbiory X i y. Kolejnym krokiem w tej części zadania był podział zbioru danych na zbiór treningowy i zbiór testowy co zostało wykonane za pomocą metody train\_test\_split.

## Wybór modeli

models = {

'Random Forest Regressor': RandomForestRegressor(n\_estimators=100, random\_state=42),

'Linear Regression': LinearRegression(),

'Ridge Regression': Ridge()

}

W celu wytrenowania modelu, wybrane zostały trzy różne modele:

* Las Losowy;
* Regresja Liniowa;
* Regresja Grzbietowa.

Ich jakość zostanie w późniejszym etapie porównana.

## Pętla ucząca

### Wykonanie

for name, model in models.items():

model.fit(X\_train, y\_train)

y\_train\_pred = model.predict(X\_train)

y\_test\_pred = model.predict(X\_test)

train\_mae = mean\_absolute\_error(y\_train, y\_train\_pred)

test\_mae = mean\_absolute\_error(y\_test, y\_test\_pred)

train\_mse = mean\_squared\_error(y\_train, y\_train\_pred)

test\_mse = mean\_squared\_error(y\_test, y\_test\_pred)

print(f'==={name}===')

print(f'{name} - Train MAE: {train\_mae}')

print(f'{name} - Test MAE: {test\_mae}')

print(f'{name} - Train MSE: {train\_mse}')

print(f'{name} - Test MSE: {test\_mse}')

with open(f'{name.replace(" ", "\_").lower()}\_model.pkl', 'wb') as file:

pickle.dump(model, file)

Uczenie modeli następuje w funkcji iterując po, utworzonym w poprzedniej części, słowniku. Po wytrenowaniu i sprawdzeniu modelu liczone są średni błąd bezwzględny (*MAE – Mean Absolute Error*) i błąd średniokwadratowy (*MSE – Mean Squarred Error*), jako miara jakości modelu. Następnie są wyświetlane i model jest zapisywany w celu możliwego późniejszego użycia.

### Wynik

===Random Forest Regressor===

Random Forest Regressor - Train MAE: 0.12214563147407984

Random Forest Regressor - Test MAE: 0.3275993549176358

Random Forest Regressor - Train MSE: 0.03536180936744548

Random Forest Regressor - Test MSE: 0.2557259876588585

===Linear Regression===

Linear Regression - Train MAE: 0.5286283596581934

Linear Regression - Test MAE: 0.5332001304956565

Linear Regression - Train MSE: 0.5179331255246697

Linear Regression - Test MSE: 0.5558915986952437

===Ridge Regression===

Ridge Regression - Train MAE: 0.5286393234710571

Ridge Regression - Test MAE: 0.5332039182571147

Ridge Regression - Train MSE: 0.5179332149226819

Ridge Regression - Test MSE: 0.5558034669932211

Z tego wynika, że najlepiej z danym problemem spośród wybranych modeli radzi sobie las losowy.

## Testowanie najlepszego modelu i wizualizacja

### Testowanie modelu

chosen\_model = models['Random Forest Regressor']

y\_test\_pred = chosen\_model.predict(X\_test)

Jak zostało wyżej wspomniane – modelem z najlepszym performensem okazał się las losowy. W tym celu dodatkowo wykonywany jest test tego modelu, aby zwizualizować jego predykcje.

### Wizualizacja

plt.figure(figsize=(10, 4))

plt.scatter(y\_test, y\_test\_pred, alpha=0.5)

plt.xlabel('Real Values')

plt.ylabel('Predicted Values')

plt.title('Real vs Predicted Values with Random Forest Regressor')

plt.show()

Za pomocą modułu matplotlib została wykonana wizualizacja jakości modelu. W tym celu został wykonany scatter plot, który przedstawia różnicę w prawdziwych i przewidywanych wartościach.

A graph showing a blue line

Description automatically generated with medium confidence

# Wnioski

* Las losowy najlepiej poradził sobie z podanym zadaniem.
* Las losowy jest lepszy pod tym względem, że ciężej się przeucza ze względu na ilość będącym w nich drzew decyzyjnych.
* Nie do każdego zadania się nadają, jednak do tego okazały się dość dobre.
* Regresja liniowa nie jest najlepszym wyborem do każdego zadania wymagającego predykcji wartości.