
	Politechnika Bydgoska im. J. J. Śniadeckich Wydział Telekomunikacji, Informatyki i Elektrotechniki		
<b>Przedmiot</b>	Analiza Regresji i Szeregów Czasowych		
<b>Prowadzący</b>	dr inż. Damian Ledziński		
<b>Temat</b>	JForex cz. 2		
<b>Student</b>	Zuzanna Tarazewicz	<b>Nr Indeksu</b>	116954
<b>Ocena</b>		<b>Data Oddania</b>	

## Table of Contents

<b>Cel .....</b>	<b>3</b>
<b>Stworzenie modelu regresji liniowej.....</b>	<b>3</b>
Zaimportowanie odpowiednich bibliotek .....	3
Zaimportowanie danych i podzielenie zbioru .....	3
Wybór modeli .....	4
Pętla ucząca .....	4
Wykonanie .....	4
Wynik.....	5
Testowanie najlepszego modelu i wizualizacja .....	5
Testowanie modelu.....	5
Wizualizacja .....	5
<b>Wnioski .....</b>	<b>6</b>

## 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

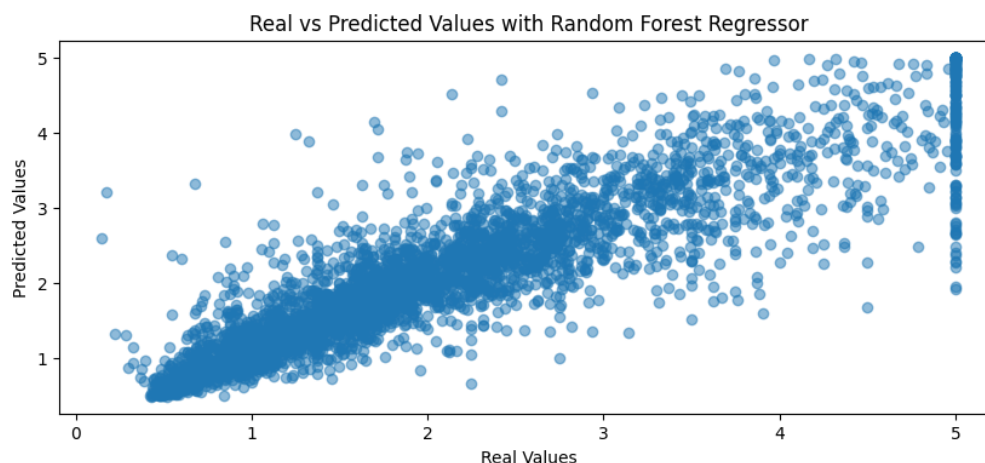
```
chosen_model = models['Random Forest Regressor']  
y_test_pred = chosen_model.predict(X_test)
```

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.



## Wnioski

- Las losowy najlepiej poradził sobie z podanym zadaniem.
- Las losowy jest lepszy pod tym względem, że ciężiej 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.