Etap 2 - przygotowywanie modeli

Autorzy

Maksymilian Baj

Bartosz Psik

Tworzenie modeli

Zbiór danych

Przygotowanie danych

W celu rozwiązania problemu biznesowego — **określenia czynników wpływających na to, że lokale są wynajmowane na dłuższy okres czasu** — posłużyliśmy się kilkoma źródłami danych:

- `listings.csv` – zawiera szczegółowe informacje o ofertach wynajmu
- `sessions.csv` – umożliwił wyliczenie liczby rezerwacji krótkich, długich oraz łącznych, co pozwoliło nam zbudować etykiety (`target`)
- `reviews.csv` – po analizie sentymentu komentarzy (z ograniczeniem do języka angielskiego) wyodrębniliśmy liczby komentarzy pozytywnych i negatywnych

Uwzględniliśmy także:

- stan wyposażenia lokalu
- liczbę udogodnień, w tym tych z listy najpopularniejszych

Dodatkowo w wyborze cech posłużyliśmy się **macierzą korelacji**, aby wybrać zmienne silnie skorelowane z funkcją celu, oraz kilkoma cechami eksperckimi (na które — jako potencjalni klienci — sami byśmy zwracali uwagę).

Modyfikacje zbioru danych

Podczas testowania różnych modeli, zbiór danych był modyfikowany:

- usuwaliśmy kolumny zbyt wysokiej krotności przy kodowaniu One-Hot-Encodingiem (OHE)
- usuwaliśmy kolumny zawierające zbyt wiele brakujących danych

Ostatecznie do trenowania modeli wykorzystaliśmy **około 28 800 rekordów** z dostępnych pierwotnie 39 000 listingów.

Do ewaluacji modeli podzieliliśmy również dane na 3 zbiory :

1. treningowy - do trenowania modelu
2. walidacyjny - do oceny treningu
3. testowy - do sprawdzania jak stworzony model jest dobry

Finalna struktura danych

Cechy numeryczne (numeric):

`['id', 'host_acceptance_rate', 'latitude', 'longitude', 'accommodates', 'bathrooms', 'bedrooms', 'beds', 'price', 'review_scores_rating', 'review_scores_accuracy', 'review_scores_cleanliness', 'review_scores_checkin', 'review_scores_communication', 'review_scores_location', 'review_scores_value', 'reviews_per_month', 'availability_eoy', 'number_of_reviews_ly', 'total_bookings', 'total_reviews', 'total_english_reviews', 'count_negative_english', 'count_positive_english', 'num_of_amenities', 'num_of_other_amenities', 'has_wifi', 'has_air_conditioning', 'num_of_top_10_common_amenities']`

Cechy kategoryczne (categorical):

`['host_is_superhost', 'host_verifications', 'neighbourhood_cleansed', 'property_type', 'room_type', 'bathrooms_text', 'license', 'instant_bookable', 'target', 'amenities', 'standardized_amenities', 'mean_embedding']`

Klasyfikator

Model naiwny

Zgodnie z założeniem, model naiwny zawsze przewiduje klasę większościową. W naszym przypadku **klasa mniejszościowa stanowi około 22%** wszystkich przypadków, więc oczekujemy, że dokładność modelu na zbiorze testowym będzie wynosić około **78%** (zależnie od próby danych).

Do oceny skuteczności modeli posługujemy się także:

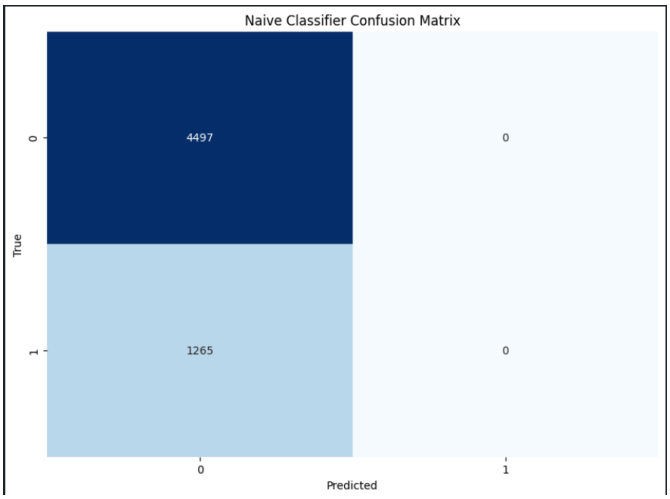
- **krzywą ROC**
- **wskaźnikiem AP (average precision)**

```
class NaiveClassifier(nn.Module):
    def __init__(self, input_size: int, output: int = 0):
        super(NaiveClassifier, self).__init__()
        self.input_size = input_size
        self.output = output


    def forward(self, x):
        batch_size = x.shape[0]
        logits = torch.zeros((batch_size, 2), dtype=torch.float32)
        logits[:, self.output] = 1.0
        return logits
```

  - Zakładamy, że model zwracał będzie klasę mniejszościową i zgodnie z naszym kodowaniem danych, tej klasie odpowiada liczba 0 → (output przekazywany do modelu)

Wyniki zwracane przez model :

```
Evaluating: 100%|██████████| 181/181 [00:00<00:00, 744.08it/s]
Accuracy: 0.7805
ROC AUC: 0.5000
Average Precision: 0.2195
```



Modele docelowe

Wersja 1 – tylko dane numeryczne

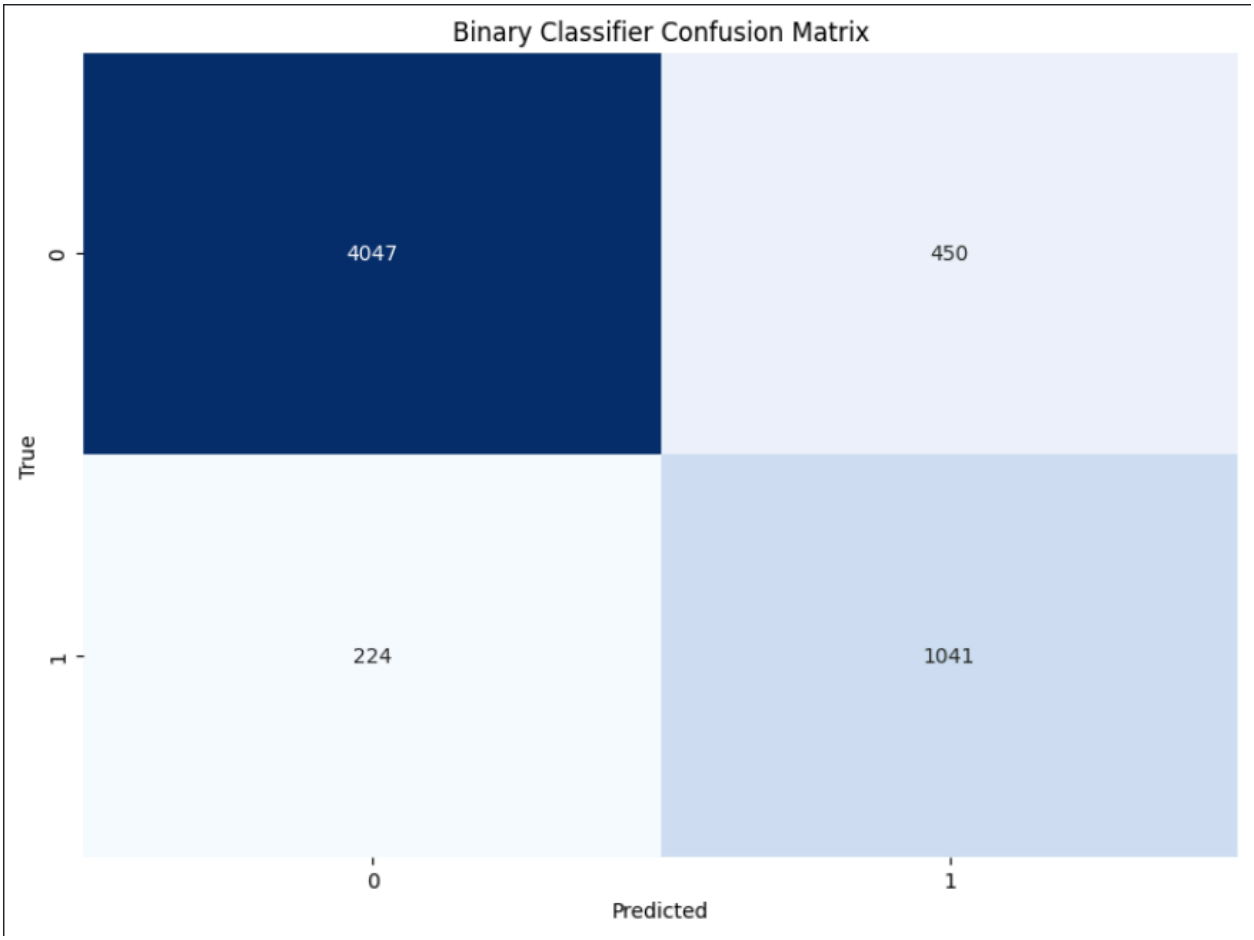
- W pierwszej kolejności sprawdziliśmy jak dane numeryczne wpływają na wyniki naszego klasyfikatora

```
class BinaryClassifier(nn.Module):
    def __init__(self, input_size: int, output_size: int):
        super(BinaryClassifier, self).__init__()
        self.input_size = input_size
        self.output_size = output_size

        self.fc1 = nn.Linear(input_size, 128)
        self.fc2 = nn.Linear(128, 64)
        self.fc3 = nn.Linear(64, output_size)

        self.actv = F.relu

    def forward(self, x):
        x = self.actv(self.fc1(x))
        x = self.actv(self.fc2(x))
        x = self.fc3(x)
```

```
    return x
```

Wyniki modelu:

```
Evaluating: 100%|██████████| 181/181 [00:00<00:00, 391.31it/s]
Accuracy: 0.8830
ROC AUC: 0.9487
Average Precision: 0.8231
```



- Accuracy: ok. 88%
- ROC AUC: ok. 0.94
- Average Precision: ok. 0.82

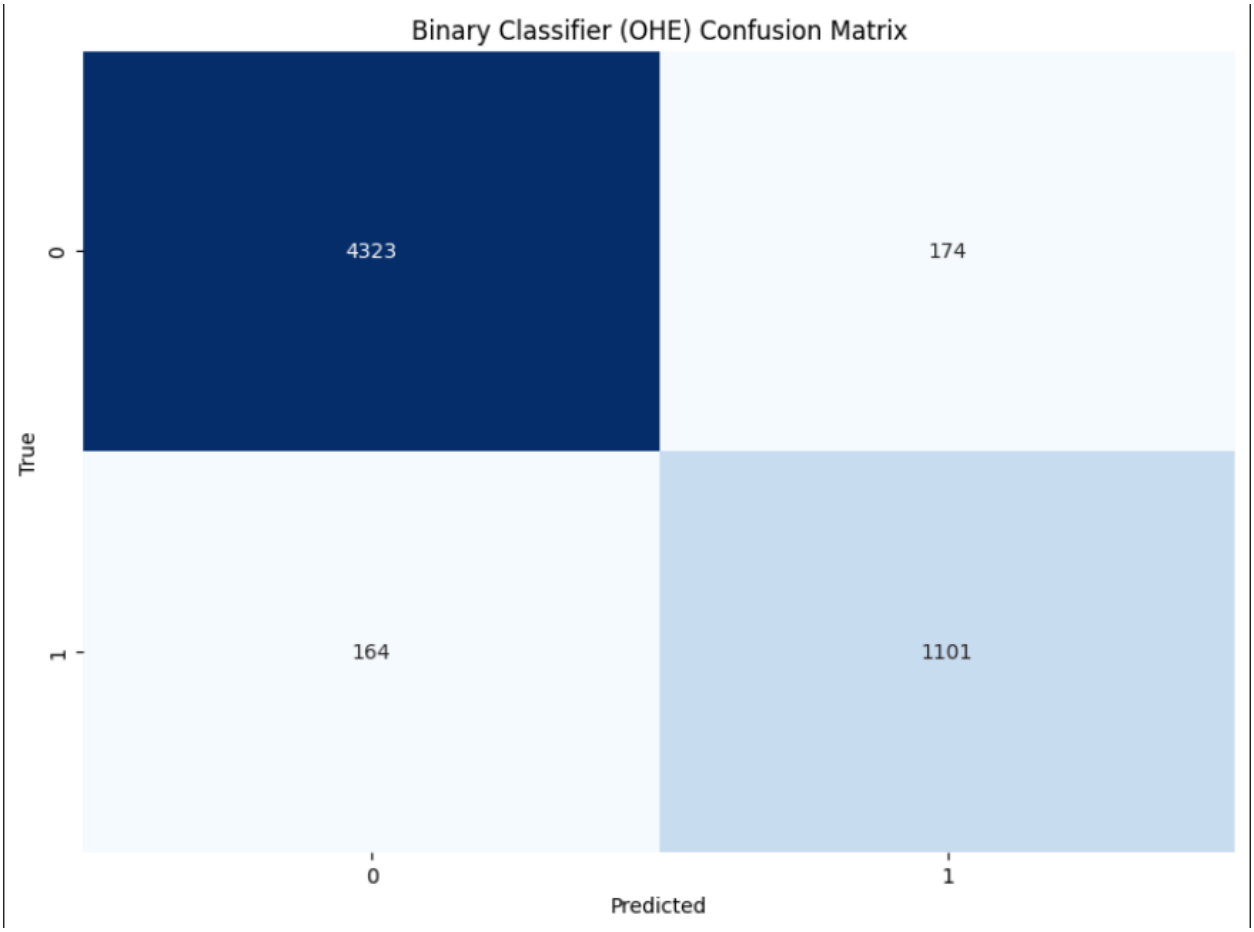
Jest to znaczna poprawa w stosunku do modelu naiwnego oraz modelu `LogisticRegression` z sklearn

Wersja 2 – One-Hot Encoding

- Drugi model dodatkowo wykorzystywał cechy kategoryczne zakodowane za pomocą One-Hot Encodingu, co zwiększyło liczbę cech do 140.

Wyniki modelu :

```
Evaluating: 100%|██████████| 181/181 [00:00<00:00, 365.83it/s]
Accuracy: 0.9413
ROC AUC: 0.9838
Average Precision: 0.9513
```



- Dzięki zastosowaniu danych kategorycznych, które z wcześniejszej analizy również wykazywały dużą korelację z danymi wyjściowymi, udało nam się

uzyskać znacznie lepsze wyniki niż niż przy użyciu danych tylko numerycznych

- Oznacza to, że niektóre wartości danych kategorycznych w znaczący sposób potrafią wpłynąć na ostateczny wynik klasyfikatora, z czego wynika, że mają duży wpływ podczas wyborów klientów
- Osiągnięte wartości są na poziomie 0.94+, czyli bardzo wysokim
  - Accuracy na poziomie 94%
  - ROC na poziomie 0.98
  - Average Precission na poziomie 0.95

Wersja 3 – Embeddingi

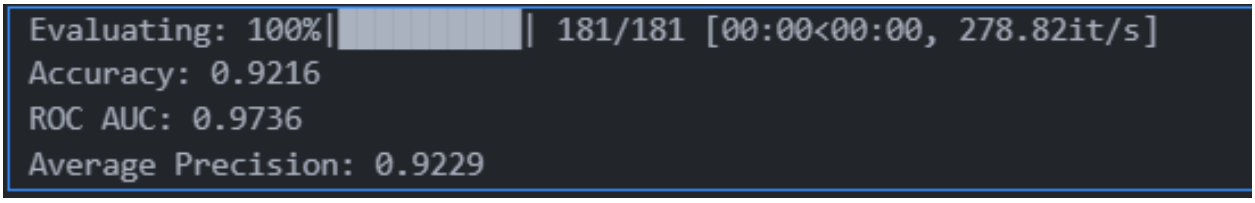
- W tej wersji dane kategoryczne reprezentujemy poprzez **embeddingi**, co pozwala lepiej uchwycić relacje pomiędzy kategoriami i ograniczyć wymiarowość
- W porównaniu do OHE wzięliśmy pod uwagę także wyposażenie `(mean_embeddings)` oraz `license` i `neighbourhood_cleansed` , które dla OHE posiadały zbyt wiele kategorii

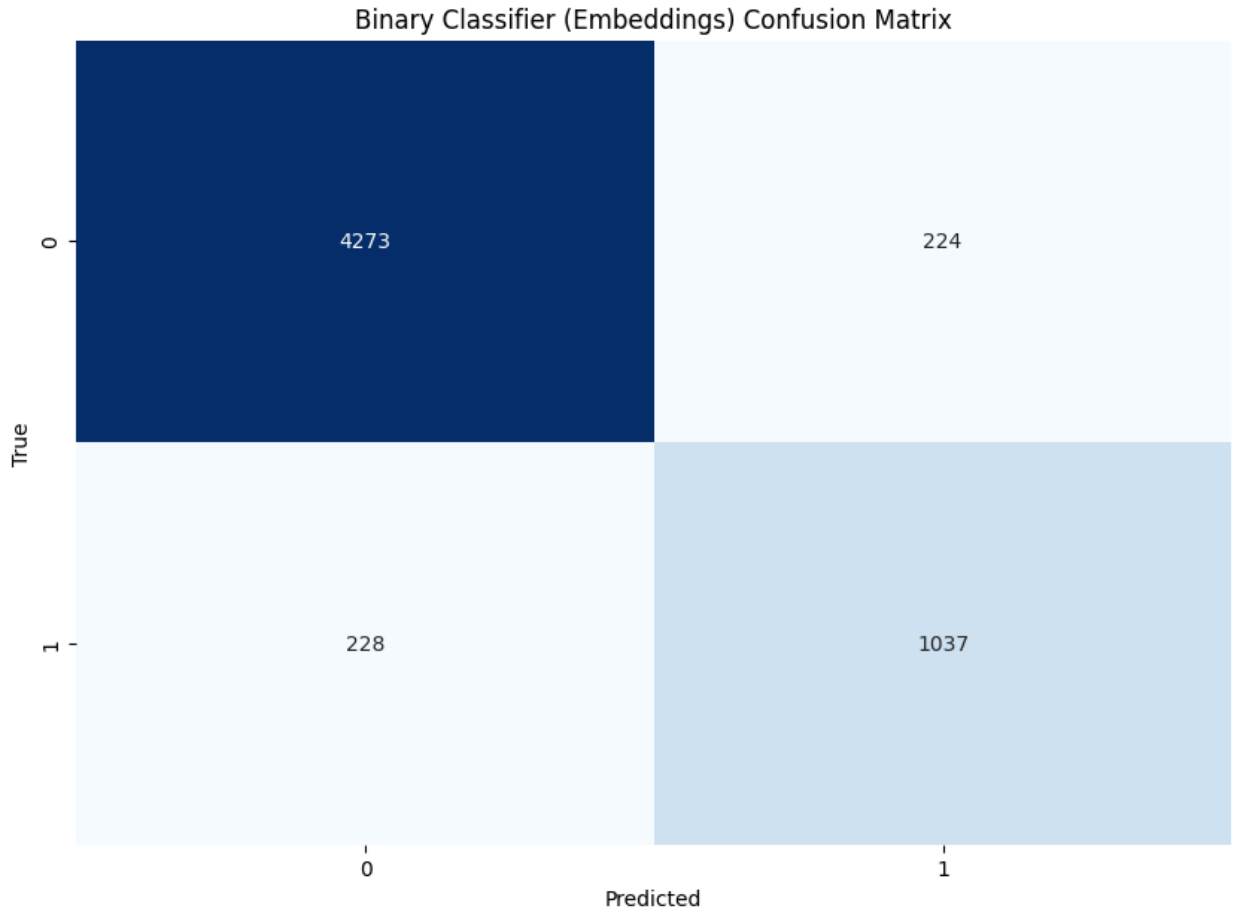
```
class BinaryClassifierEmbeddings(nn.Module):
    def __init__(self, num_numeric: int, embedding_sizes: list, output_size: int):
        super(BinaryClassifierEmbeddings, self).__init__()

        self.embeddings = nn.ModuleList([
            nn.Embedding(num_categories, emb_dim)
            for num_categories, emb_dim in embedding_sizes
        ])

        emb_dim_total = sum(emb_dim for _, emb_dim in embedding_sizes)
        self.input_size = num_numeric + emb_dim_total

        self.fc1 = nn.Linear(self.input_size, 128)
        self.fc2 = nn.Linear(128, 64)
        self.fc3 = nn.Linear(64, output_size)
```

```
        self.actv = F.relu

    def forward(self, x_cat, x_num):
        embedded = [emb(x_cat[:, i]) for i, emb in enumerate(self.embeddings)]

        if x_num.dim() == 1:
            x_num = x_num.unsqueeze(1)

        x = torch.cat(embedded + [x_num], dim=1)

        x = self.actv(self.fc1(x))
        x = self.actv(self.fc2(x))
        x = self.fc3(x)
        return x
```

- Zbudowany model został lekko przerobiony, żeby był w stanie oddzielić dane kategoryczne od danych numerycznych

Wyniki modelu:





- Accuracy: powyżej 92%
- ROC AUC: wysoki (powyżej 0.93)
- Average Precision: również wysoki

Model ten nieznacznie ustępuje wersji z OHE, ale dobrze radzi sobie z wysoką krotnością cech kategorycznych.

Feature Importance:

- Sprawdziliśmy także, brak jakich danych w znaczącym stopniu wpływa na wynik i oto rezultaty:
  - Dla par atrybutów (najważniejszych kilka pokazanych)

```
Permutation Feature Importances (by ROC AUC drop):
['host_is_superhost', 'review_scores_value']: 0.4473
['review_scores_value', 'num_of_top_10_common_amenities']: 0.4080
```

```
['bedrooms', 'review_scores_value']: 0.4034
['bathrooms', 'review_scores_value']: 0.3941
['review_scores_accuracy', 'review_scores_value']: 0.3939
['license', 'review_scores_value']: 0.3935
['room_type', 'review_scores_value']: 0.3934
['review_scores_value', 'count_positive_english']: 0.3932
['review_scores_value', 'total_english_reviews']: 0.3930
['review_scores_value']: 0.3923
['property_type', 'review_scores_value']: 0.3903
['review_scores_rating', 'review_scores_value']: 0.3884
['review_scores_value', 'total_reviews']: 0.3876
['review_scores_value', 'reviews_per_month']: 0.3872
['review_scores_value', 'total_bookings']: 0.3868
['latitude', 'review_scores_value']: 0.3864
['bathrooms_text', 'review_scores_value']: 0.3856
['mean_embedding', 'review_scores_value']: 0.3838
['beds', 'review_scores_value']: 0.3827
['longitude', 'review_scores_value']: 0.3826
```

- Dla pojedynczych atrybutów :

```
Permutation Feature Importances (by ROC AUC drop):
['review_scores_value']: 0.3853
['host_is_superhost']: 0.2181
['num_of_top_10_common_amenities']: 0.2085
['bedrooms']: 0.1475
['bathrooms']: 0.1225
['license']: 0.1114
['room_type']: 0.1036
['review_scores_cleanliness']: 0.1035
['review_scores_checkin']: 0.1029
['neighbourhood_cleansed']: 0.1024
['review_scores_accuracy']: 0.1014
['count_positive_english']: 0.1009
['num_of_amenities']: 0.1004
```

```
['latitude']: 0.1002
['accommodates']: 0.0995
['reviews_per_month']: 0.0995
['total_reviews']: 0.0995
['count_negative_english']: 0.0993
['total_english_reviews']: 0.0993
['total_bookings']: 0.0990
['property_type']: 0.0990
['price']: 0.0989
['review_scores_rating']: 0.0988
['review_scores_communication']: 0.0987
['longitude']: 0.0985
['review_scores_location']: 0.0985
['num_of_other_amenities']: 0.0983
['instant_bookable']: 0.0980
['host_verifications']: 0.0979
['beds']: 0.0979
['bathrooms_text']: 0.0978
['host_acceptance_rate']: 0.0969
['mean_embedding']: 0.0959
```

- Z tej analizy widać, że posiadanie niektórych atrybutów lub ich brak ma wpływ na wyniki klasyfikatorów co może nam podpowiadać, czym dokładnie sugerują się ludzie, podczas rezerwacji

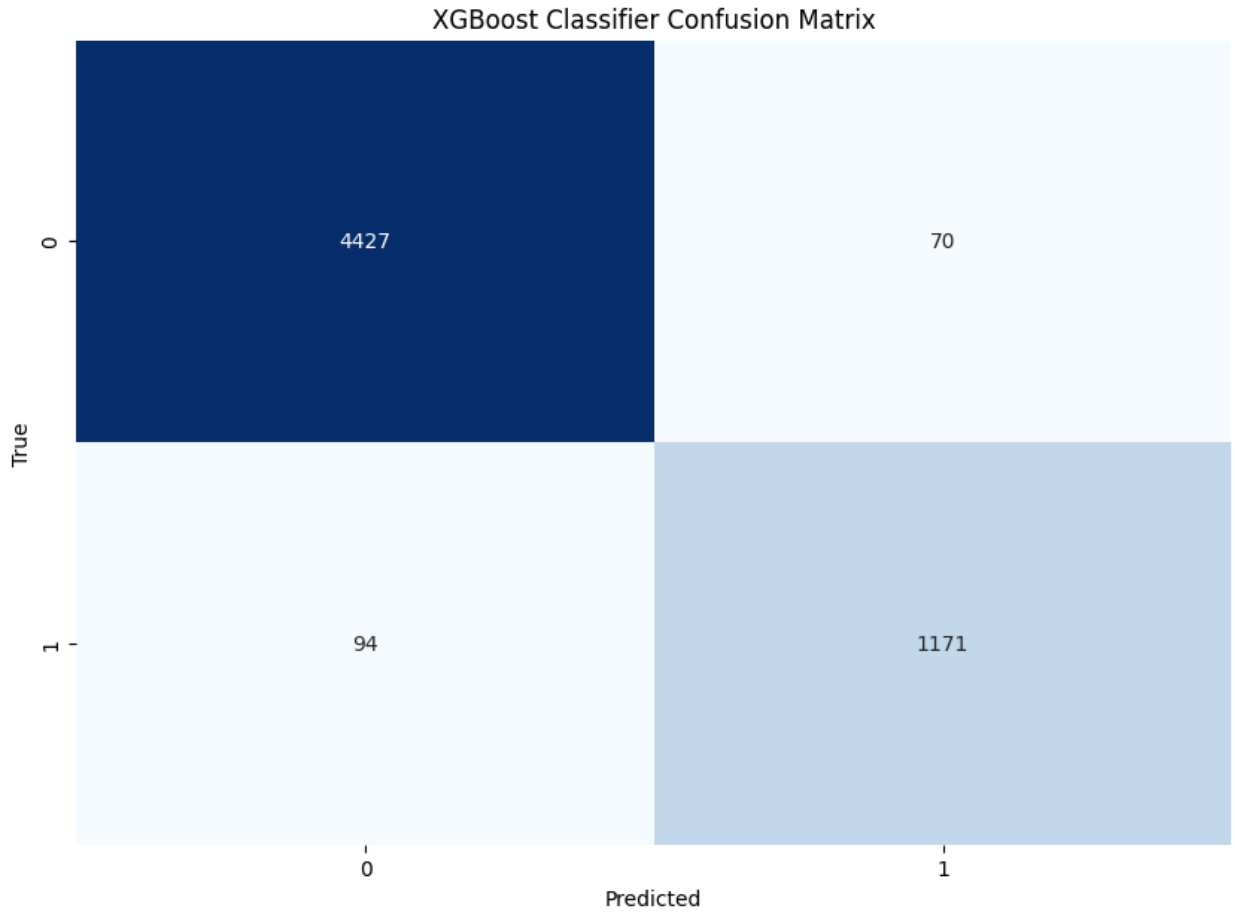
Wersja 4 - XGBoost

- Ostatni model zbudowano w oparciu o bibliotekę XGBoost, z danymi zakodowanymi przy użyciu OHE.
- Pozwoliło to także na analizę znaczenia cech wbudowanymi metodami biblioteki.

```
xgb_classifier = XGBClassifier()
```

Wyniki modelu :

```
Accuracy: 0.9715
ROC AUC: 0.9959
Average Precision: 0.9860

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98      4497
           1       0.94      0.93      0.93      1265

    accuracy                           0.97      5762
   macro avg       0.96      0.96      0.96      5762
weighted avg       0.97      0.97      0.97      5762
```



- Najlepsze rezultaty spośród wszystkich testowanych modeli

Feature Importance :

- W notantiku `classifiers_experiments.ipnyb` znajduję się także wykres wpływu parametrów na wynik predykcji. Widzimy tam, że na wynik głównie wpływa :
  - czy host jest superhostem
  - średnia ocena
  - kategorie / opis łazienek
  - liczba wynajęć całkowta
  - cena
  - liczba pomieszczeń (bedrooms, bathrooms)
  - liczba pozytywnych i negatywnych komentarzy

- sposób weryfikacji hosta
- rodzaj posiadłości (niektóre wartości wpływają), ale to także w poprzedniej analizie zauważyliśmy
- rodzaj pokoju

Możliwość rozwoju

Osiągnięte przez nas wyniki, w znaczącym stopniu przewyższył modele naiwne. Jest to już zadowalający efekt w celu rozpoznawania klasyfikacji lokali na lokale wynajmowane na krótko lub długo. Nasuwają się jednak jeszcze możliwości poprawy :

- Przetestowanie większej ilości atrybutów
- Poprawienie imputacji danych, szukając jakiś powiązań z innymi atrybutami

---

Plan wykorzystania modeli (finalny a nie obecny)

1. Klient poda dane dotyczące swojej oferty
2. Podane dane wykorzystamy w sposób następujący :
    a. połączymy je z istniejącą bazą danych, żeby dołączyć brakujące dane niepodane przez klienta, o ile takowe dane znajdują się w naszej bazie
    b. w przypadku braku wyliczeń dokonamy imputacji danych lub w najgorszym przypadku, jeśli imputacja będzie niemożliwa uzupełnimy 0
3. Danymi zasilimy model, który przekaże klientowi:
    a. czy jego lokal raczej będzie wynajmowany na długo czy na krótko
    b. co z podanych przez niego elementów (podanych nie dobranych z bazy) wpływa najbardziej na decyzję, dzięki czemu będzie mógł rozważyć zmiany odnośnie swojej oferty by może model zwrócił mu inną (bardziej oczekiwaną) predykcję
    c. wypiszemy, że w celu osiągnięcia większej pewności, należy podać wszystkie dane, z którymi wytrenowany model pracuje