# Skrypt do prezentacji

Predykcja Opóźnień Lotów - Uczenie Maszynowe

K. Arkit, D. Chomiak, Ł. Guziczak, D. Sobótka

14 czerwca 2025

*Ten dokument zawiera pełny skrypt do wygłoszenia prezentacji. Każda sekcja odpowiada jednemu slajdowi. Czas prezentacji: 10-15 minut.*

# Slajd 1: Slajd tytułowy

### Czas: 30 sekund

Dzień dobry Państwu. Nazywamy się Kamil Arkit, Dawid Chomiak, Łukasz Guziczak i Dawid Sobótka.

Dzisiaj przedstawimy Państwu nasz projekt z uczenia maszynowego dotyczący predykcji opóźnień lotów. Jest to problem, który dotyka miliony pasażerów rocznie i generuje ogromne koszty dla linii lotniczych.

Przejdźmy do definicji problemu.

# Slajd 2: Definicja problemu

### Czas: 2 minuty

Naszym celem było stworzenie modelu uczenia maszynowego, który przewidzi, czy lot będzie opóźniony o więcej niż 15 minut. Jest to klasyczny problem klasyfikacji binarnej.

Wykorzystaliśmy dataset US Flight Delays z 2015 roku, który zawiera informacje o 5,8 miliona lotów. Ze względu na ograniczenia obliczeniowe, pracowaliśmy na próbce 500 tysięcy rekordów, zachowując reprezentatywność danych.

Dataset zawiera 31 atrybutów opisujących każdy lot - od podstawowych informacji jak linia lotnicza i lotniska, przez szczegóły czasowe, aż po dane o dystansie.

Po prawej stronie widzą Państwo rozkład opóźnień w naszych danych. Kluczowa obserwacja: tylko 19,4

Dlaczego wybraliśmy ten problem? Po pierwsze, jest to duży, rzeczywisty dataset, który pozwala na praktyczne zastosowanie technik uczenia maszynowego. Po drugie, niezbalansowane klasy wymagają przemyślanego podejścia do modelowania. Po trzecie, możemy przetestować różnorodne techniki - od prostych modeli po zaawansowane ensemble.

# Slajd 3: Ewolucja modelu - 4 etapy

### Czas: 2,5 minuty

Na tym slajdzie przedstawiamy ewolucję naszego modelu przez cztery kluczowe etapy. Wykres pokazuje, jak zmieniały się główne metryki w kolejnych iteracjach.

**Etap 1 - Baseline:** Zaczęliśmy od podstawowego modelu z tylko 12 cechami - były to surowe atrybuty z datasetu jak miesiąc, dzień tygodnia, linia lotnicza czy dystans. Bez

żadnego feature engineeringu. Jak widać, model osiągnął niski recall - zaledwie 14,5

**Etap 2 - Data Leakage:** W drugim etapie celowo wprowadziliśmy błąd data leakage, dodając cechę DELAY_LOG, która zawierała informacje dostępne dopiero po fakcie. Metryki wystrzeliły w górę - recall 100

**Etap 3 - Fast Optimized:** Po usunięciu wycieku danych, skupiliśmy się na prawdziwej optymalizacji. Zastosowaliśmy ensemble trzech modeli - Random Forest, XGBoost i LightGBM. Dodatkowo usunęliśmy ekstremalne wartości opóźnień powyżej 300 minut. Model osiągnął recall 55,1

**Etap 4 - Final Model:** W ostatnim etapie naprawiliśmy błąd z usuwaniem outlierów - zachowaliśmy WSZYSTKIE dane, również ekstremalne opóźnienia. Rozszerzyliśmy liczbę cech do 28 poprzez feature engineering. Co to znaczy? Stworzyliśmy nowe, bardziej użyteczne zmienne: cykliczne kodowanie czasu (godzina jako sinus i cosinus, żeby model wiedział że 23:00 jest blisko 1:00), binarne flagi jak IS_RUSH_HOUR czy IS_WEEKEND, wskaźniki opóźnień dla linii i lotnisk, a nawet interakcje między cechami - na przykład łącząc godziny szczytu z ryzykiem konkretnej linii lotniczej. Finalny model osiągnął F1-score 0,491 i ROC-AUC 0,769.

Warto zauważyć spadek recall z 55,1

## Slajd 4: Porównanie modeli - krzywe ROC

### Czas: 2 minuty

Ten wykres przedstawia krzywe ROC dla wszystkich czterech etapów naszego projektu. Krzywa ROC pokazuje zależność między true positive rate a false positive rate dla różnych progów decyzyjnych.

Spójrzmy na ewolucję: - Niebieski Baseline z AUC 0,643 - ledwo lepszy niż losowy klasyfikator - Pomarańczowa krzywa Data Leakage - praktycznie idealna z AUC 0,999, co było ostrzeżeniem o problemie - Zielona Fast Optimized z AUC 0,751 - znacząca poprawa po usunięciu wycieku - Czerwony Final Model z AUC 0,769 - nasz najlepszy uczciwy wynik

Kluczowa obserwacja: mimo że w Final Model mieliśmy niższy recall niż w Fast Optimized, to ROC-AUC jest wyższy. To pokazuje, że model lepiej radzi sobie z rankingiem prawdopodobieństw, co daje większą elastyczność w doborze progu decyzyjnego w zależności od potrzeb biznesowych.

Etap z Data Leakage był cenną lekcją - pokazał nam, jak ważne jest dokładne sprawdzenie, czy nasze cechy są dostępne w momencie predykcji. W prawdziwym świecie taki błąd mógłby kosztować miliony.

## Slajd 5: Analiza najważniejszych cech

### Czas: 2 minuty

Przejdźmy do analizy, które cechy okazały się najważniejsze dla naszego modelu.

Wykres po lewej pokazuje top 15 cech według ważności w modelu XGBoost. Dominują cechy związane z czasem:

**IS_RUSH_HOUR** - najważniejsza cecha z wagą 16,5

**HOUR_SIN i HOUR_COS** - to cykliczne kodowanie godziny odlotu. Używamy funkcji sinus i cosinus, aby model zrozumiał, że godzina 23:00 jest blisko 1:00. Razem te cechy mają ponad 20

**IS_WEEKEND i IS_FRIDAY** - weekendy mają inny wzorzec ruchu niż dni robocze. Ciekawe, że piątek został wyodrębniony jako osobna cecha.

Wśród innych ważnych cech mamy: - Linię lotniczą (8,8- Lotnisko wylotu (7,9- Dystans lotu (7,6

Główny wniosek: czas odlotu ma największy wpływ na przewidywanie opóźnień. To sugeruje, że harmonogram lotów i obciążenie lotnisk w różnych porach dnia są kluczowe dla punktualności.

## Slajd 6: Wydajność finalna

### Czas: 2 minuty

Na tym slajdzie przedstawiamy szczegółową analizę wydajności naszego finalnego modelu.

Po lewej widzimy macierz pomyłek. Z prawie 100 tysięcy lotów w zbiorze testowym: - Poprawnie przewidzieliśmy 66,600 lotów na czas (True Negatives) - Wykryliśmy 12,400 opóźnionych lotów (True Positives) - Przegapiliśmy 10,400 opóźnień (False Negatives) - to nasz główny problem - Fałszywie alarmowaliśmy w 15,100 przypadkach (False Positives)

Wykres po prawej pokazuje kluczowe odkrycie: recall dramatycznie spada wraz z wielkością opóźnienia. Dla małych opóźnień (15-30 minut) wykrywamy 61

Nasze finalne metryki to: - F1-score: 0,491 - Recall: 54,4- Precision: 45,0- ROC-AUC: 0,769 - dobra zdolność do rankingu

Te wyniki pokazują klasyczny trade-off w uczeniu maszynowym między recall a precision. W kontekście biznesowym należałoby zdecydować, co jest gorsze: przegapione opóźnienie czy fałszywy alarm.

## Slajd 7: Podsumowanie

### Czas: 2 minuty

Podsumowując nasz projekt:

**Co udało się osiągnąć:** Przetestowaliśmy cztery różne podejścia, od prostego baseline po zaawansowany ensemble. Zidentyfikowaliśmy i naprawiliśmy problem data leakage, co było cenną lekcją. Poprzez feature engineering zwiększyliśmy liczbę cech z 12 do 28, włączając w to cykliczne kodowanie czasu i interakcje między zmiennymi. Najlepszy model - XGBoost - osiągnął ROC-AUC 0,769, co jest solidnym wynikiem dla tego problemu.

**Napotkane wyzwania:** Głównym wyzwaniem były niezbalansowane klasy - tylko 19,4

**Propozycje ulepszeń:** Widzimy kilka kierunków rozwoju: - Model dwuetapowy: najpierw klasyfikacja normalne/ekstremalne opóźnienie, potem dedykowane modele - Dane pogodowe znacząco poprawiłyby predykcje - Stacking ensemble z meta-learnerem mógłby lepiej łączyć różne modele - Kalibracja prawdopodobieństw dałaby bardziej wiarygodne estymaty ryzyka

Ten projekt pokazał nam złożoność rzeczywistych problemów ML i znaczenie iteracyjnego podejścia do modelowania.

## Slajd 8: Dziękujemy za uwagę!

### Czas: 30 sekund

Dziękujemy Państwu za uwagę i zapraszamy do zadawania pytań.

Cały kod źródłowy naszego projektu jest dostępny na GitHubie w repozytorium ML-FlightDelaysAndCancellations.

Jesteśmy gotowi odpowiedzieć na wszelkie pytania dotyczące metodologii, wyników czy potencjalnych zastosowań naszego modelu.

---

**Dodatkowe informacje na wypadek pytań:**

- **Dlaczego 15 minut?** - To standard branżowy dla definicji opóźnienia w USA
- **Dlaczego XGBoost?** - Najlepsza równowaga między wydajnością a czasem treningu
- **Koszt biznesowy?** - Średni koszt opóźnienia to $75/minuta dla linii lotniczej

- **Czas treningu?** - Final model: około 15 sekund na 500k rekordów
- **Możliwość wdrożenia?** - Tak, model jest gotowy do deploymentu, ROC-AUC 0,769 jest akceptowalny