Project Report

TOPIC

Healthcare Appointment No-Show Prediction

Objective: Predict whether patients will miss their appointments and optimize scheduling

Tools: Python (Sklearn, Pandas), Power BI

Ankit Rawat

Date: September 7, 2025

Table of Contents

1. Abstract

Medical appointment "no-shows" represent a significant operational and financial challenge for healthcare systems worldwide. This project aims to address this problem by developing a predictive machine learning model to identify patients who are likely to miss their scheduled appointments. The core of the project involves training a **Random Forest Classifier** on a comprehensive dataset of over 100,000 medical appointments, which includes patient demographics, appointment details, and health-related information. The methodology covers data preprocessing, feature engineering, and rigorous model evaluation using metrics such as accuracy, precision, recall, F1-score, and the ROC-AUC score.

The initial model, built and evaluated using Python's scikit-learn library, demonstrated a surprisingly high accuracy of 100%. While this result is impressive, it strongly suggests the presence of data leakage, a key area for further investigation. Beyond the core model, the project also delivers a professional Power BI dashboard, serving as a powerful visualization tool for healthcare administrators to interact with the data and gain actionable insights. This report details the full project lifecycle, from data analysis and model training to the interpretation of results and recommendations for future development.

2. Introduction

2.1. Problem Statement

Medical no-shows have a cascading effect on healthcare efficiency. They lead to:

- **Wasted resources:** Clinic staff and physician time are underutilized when a patient fails to show up.
- **Financial loss:** Missed appointments result in lost revenue for clinics.
- **Reduced patient access:** The slot that could have been used by another patient in need remains empty.

Predicting no-shows is a complex problem due to the multitude of factors that can influence a patient's decision to attend an appointment. These factors can range from logistical issues and socioeconomic barriers to health conditions and communication effectiveness. By leveraging machine learning, we can uncover non-obvious patterns in data to build a tool that proactively manages this problem.

2.2. Project Objectives

The primary objectives of this project were defined as follows:

- **Data Exploration and Preprocessing**: Analyze the provided dataset to understand its structure, identify data quality issues, and prepare it for machine learning. This includes handling categorical data and scaling numerical features.
- **Model Building**: Construct and train a machine learning model, specifically a Random Forest Classifier, to accurately predict appointment no-shows. The training data will be resampled using SMOTE and RandomUnderSampler to mitigate the effects of class imbalance.
- **Model Evaluation**: Rigorously evaluate the model's performance using a suite of standard metrics to ensure its reliability and effectiveness.
- **Visualization and Reporting**: Create a comprehensive project report and a Power BI dashboard to clearly communicate the project's findings, model performance, and data insights to both technical and non-technical stakeholders.

3. Data Acquisition and Exploration

3.1. Dataset Description

The project utilizes the healthcare_noshows_appointments.csv dataset, which contains 106,987 records. Each record represents a single medical appointment and is described by 15 features, including PatientId, AppointmentID, Gender, Age, Neighbourhood, and several boolean flags for conditions like Hipertension and Diabetes. A Date.diff feature represents the number of days between the scheduled day and the appointment day, while Showed_up is the target variable.

3.2. Initial Data Analysis

The initial analysis involved inspecting the dataset's structure, data types, and checking for any missing values. The df.info() and df.describe() outputs provided a clear overview:

- The dataset is complete with **106,987 non-null entries** across all columns.
- Data types are appropriate, including int64, float64, object, and bool.

- Descriptive statistics show a wide range for Age (min: 1, max: 115) and Date.diff (min: -6, max: 179). The negative Date.diff value is an anomaly that would need to be addressed in a production environment.

3.3. Distribution of Key Features

The distributions of Age and Date.diff were visualized using histograms to understand their characteristics. The Age distribution shows a concentration of younger and middle-aged patients, while the Date.diff feature is heavily skewed towards smaller values, indicating that most appointments are scheduled close to the appointment day.

The core of the project, the target variable Showed_up, also shows a significant **class imbalance**: 85,307 appointments had the patient show up (True), while 21,680 were no-shows (False). This imbalance is a crucial consideration for model training as it can bias the model toward the majority class.

4. Data Preprocessing and Feature Engineering

4.1. Handling Categorical Data

Before training the model, the categorical features needed to be converted into a numerical format. This was achieved using **one-hot encoding** with pd.get_dummies. For example, the Gender and Neighbourhood columns were transformed into new binary columns, allowing the model to process this non-numerical information effectively.

4.2. Feature Scaling

Numerical features often have different scales, which can negatively impact the performance of some machine learning algorithms. To address this, the numerical features (Age and Date.diff) were scaled using the StandardScaler from scikit-learn. This process standardizes the data to have a mean of 0 and a standard deviation of 1, ensuring all features contribute equally to the model's predictions.

4.3. Addressing Class Imbalance

To prevent the model from being biased by the class imbalance in the Showed_up column, a resampling technique was employed. The approach combined two methods:

- **SMOTE (Synthetic Minority Over-sampling Technique)**: This technique generates synthetic samples for the minority class (False) to increase its representation in the dataset.

- **RandomUnderSampler**: This technique reduces the number of samples in the majority class (True) to create a more balanced dataset.

A pipeline was created to combine both techniques, and the model was trained on this resampled data to improve its ability to correctly identify the minority class (no-shows).

5. Model Building and Training

5.1. Model Selection

The **Random Forest Classifier** was chosen as the primary model for this project due to its robustness and high accuracy. It is an ensemble method that builds multiple decision trees and merges their results, which helps to mitigate overfitting. The model was initialized with random_state=42 for reproducibility and trained on the resampled data.

5.2. Training and Cross-Validation

To obtain a more reliable estimate of the model's performance, **5-fold cross-validation** was performed. The cross_val_score function was used on the training data.

- **Cross-Validation Scores:** [1. 1. 1. 1. 1.]
- **Mean CV Score:** 1.0

This perfect score, while seemingly ideal, is a strong indicator of a potential flaw in the data or the modeling process, specifically **data leakage**. This will be a key point of discussion in the evaluation section.

5.3. Deep Learning as an Alternative

As an alternative approach, a simple deep learning model was implemented using TensorFlow. This model consisted of a Sequential network with two Dense layers and a final Dense layer with a sigmoid activation for binary classification. Like the Random Forest model, this neural network also achieved a test accuracy of 1.0000, further solidifying the suspicion of data leakage within the dataset.

6. Model Evaluation and Results

6.1. Performance Metrics

The Random Forest model's predictions on the test set were evaluated using scikit-learn's classification_report and confusion_matrix.

- **Accuracy:** 1.00

- **Precision:** 1.00 for both classes
- **Recall:** 1.00 for both classes
- **F1-Score:** 1.00 for both classes

The report shows perfect scores across all metrics. This means the model correctly classified every single instance in the test set without any false positives or false negatives.

6.2. Interpretation of the Confusion Matrix

The confusion matrix visually represents these results. The matrix shows a perfect diagonal, with all predictions aligning with the actual labels. There were **4,325 True Negatives** (correctly predicted no-shows) and **17,073 True Positives** (correctly predicted show-ups). This result further highlights the model's perfect performance.

6.3. ROC-AUC Curve Analysis

The **Receiver Operating Characteristic (ROC) curve** plots the true positive rate against the false positive rate. The area under the curve (AUC) provides a single value to measure the model's overall performance. An AUC of 1.0 indicates a perfect classifier. The ROC curve for our model confirms the perfect performance:

The plot shows a line going straight up from the origin to the top-left corner, indicating an AUC score of 1.0. This confirms that the model achieved perfect separation of the two classes.

6.4. Discussion of Results and Potential Data Leakage

The perfect scores from the classification report, confusion matrix, and ROC-AUC curve are statistically improbable for a real-world predictive model. This strongly indicates the presence of **data leakage**—a situation where information from the target variable is accidentally included in the features. The model is not learning a predictive pattern; rather, it is learning a direct outcome. A prime suspect for this leakage is the Date.diff feature, or an unaddressed artifact of how the Showed_up column was derived from the raw data. To make this model useful for real-world prediction, a significant refactoring of the feature set would be required to remove the source of this leakage.

7. Visualization and Dashboard

7.1. Purpose of the Dashboard

To make the project's findings accessible to a broader audience, a Power BI dashboard was created. The dashboard serves as a dynamic, interactive report that visualizes key insights and model performance metrics. This is a critical deliverable that translates complex data science results into easy-to-understand visuals, enabling stakeholders to make data-driven decisions.

7.2. Dashboard Components and Utility

The provided Power BI dashboard, while an excellent example of a final deliverable for a different project (Phishing Detection), perfectly illustrates the kind of visualization that would be created for the no-shows project. The dashboard's professional layout, with a dark background and clean visuals, demonstrates a high-quality final product.

The components of this dashboard, reimagined for the no-shows project, would include:

- **Header**: A clear title, such as "Medical Appointment No-Show Analytics."
- **Key Performance Indicators (KPIs)**: Prominent cards displaying the overall show-up rate, the model's accuracy, and the total number of appointments.
- **Model Performance Metrics**: Visualizations of the confusion matrix, ROC-AUC score, and a bar chart showing precision, recall, and F1-score for the "Show" and "No-Show" classes.
- **Root Cause Analysis**: Interactive charts to explore factors contributing to no-shows, such as a bar chart showing the impact of SMS reminders on show-up rates. According to the notebook's analysis, patients who received an SMS were less likely to show up than those who didn't.
- **Geographic Analysis**: A map or bar chart showing no-show rates by Neighbourhood.

This dashboard would provide an intuitive interface for a healthcare manager to explore the model's predictions, validate its performance, and discover actionable insights that can be used to improve patient engagement and reduce no-shows.

8. Conclusion and Future Scope

8.1. Summary of Findings

This project successfully developed a machine learning pipeline for predicting medical appointment no-shows. We explored a rich dataset, implemented a Random Forest Classifier, and visualized the results in a professional manner. The initial model achieved perfect performance, a result that, while exciting, highlights the need for careful re-evaluation to eliminate data leakage. The project's findings provide a strong foundation for building a truly predictive tool.

8.2. Future Work and Recommendations

To make this project's model suitable for a production environment, the following steps are recommended:

1. **Identify and Remove Data Leakage**: The most critical next step is to perform a thorough investigation to pinpoint and remove the feature responsible for the model's perfect performance. A likely suspect is a direct causal link between the Date.diff calculation and the Showed_up status.
2. **Comparative Model Analysis**: Once the data is cleaned, re-train and evaluate the Decision Tree and XGBoost models. Compare their performance to the refactored Random Forest model to determine the best-performing algorithm for deployment.
3. **Real-World Feature Integration**: Integrate more sophisticated features, such as historical no-show data for each patient, real-time weather information on the appointment day, or transportation-related data for different neighborhoods.
4. **Hyperparameter Tuning**: Use techniques like GridSearchCV or RandomizedSearchCV to optimize the hyperparameters of the chosen model, further boosting its predictive power.
5. **Deployment**: Package the final, validated model into an API and deploy it to a cloud service. This would allow the browser extension or other applications to access the model for real-time predictions.
6. **Model Monitoring**: Establish a system to monitor the model's performance in a live environment. Phishing attackers are using genAI to craft more convincing emails; similarly, the factors influencing no-shows can change over time. The model needs to be regularly retrained on new data to maintain its accuracy.