Project Title: Predicting Medical Appointment No-Shows Using Machine Learning

Course: DSC 672 Data Science Capstone

Student Name: Mirza Ansar Baig

Student ID: 2171796

Institution : DePaul University

Date: 17th June, 2025

Abstract:

Missed medical appointments, or “no-shows” lead to significant operational inefficiencies and increased healthcare costs. This project explores the use of machine learning models to predict whether a patient will attend their scheduled appointment. Using a dataset from Brazilian hospitals containing over 110000 records, we applied various classification algorithms including Logistic Regression, Decision Trees, Random Forest and XGBoost. The models were evaluated both with and without the use of Synthetic Minority Over-sampling Techniques (SMOTE) to address class imbalance. Key performance metrics such as accuracy, precision, recall and F1-score were used to assess the models. Our findings highlight that model performance varies significantly with class balancing techniques, and tuned decision trees yielded valuable insights into feature importance. The study aims to contribute toward improved appointment scheduling and patient engagement strategies in healthcare systems.

Introduction:

Healthcare providers face a persistent challenge with patient missing scheduled appointments. These “no-shows” results in underutilized resources, disrupted schedules, and longer wait times for other patients. In response, predictive modeling using historical appointment data offers a promising avenue for intervention.

The dataset used in this study originates from Brazilian hospitals and contains detailed records of over 110000 patient appointments. Variables include demographics information, appointment timing, health conditions, and indicators of whether a patient received an SMS reminder. Our primary objective is to predict the likelihood of a no-show using machine learning techniques, thereby aiding healthcare providers in optimizing their scheduling processes.

To achieve this, we implemented multiple classification algorithms and addressed the dataset’s imbalanced natural using SMOTE. Each model was rigorously evaluated through performance metrics, and hyperparameter tuning was employed to enhance model accuracy. This report details our modeling approach, results, and the potential for practical application in clinical settings.

Literature Review:

Missed medical appointment are a well-recognized issue in healthcare systems worldwide, leading to increased operational costs and decrease care quality. Several studies have explored predictive modeling to address this issue. For example, Batista and Monard (2003) discussed the impact of class imbalance in medical datasets and introduced sampling strategies like SMOTE to enhance model performance. More recent work by Choi et al. (2019) applied deep learning models to electronic health records, achieving notable improvement in predicting No-shows using temporal features and historical attendance. Similarly, Mohammadi et al. (2020) compared various machine learning algorithms including decision trees, SVMs, and ensemble models concluding that ensemble approach typically yield superior accuracy for healthcare classification task. These studies motivate the use of ensemble modeling and threshold tuning in our approach.

References:

1. Batista, G.E.A.P.A., & Monard ,M.C. (2003). An analysis of four missing data treatment methods for supervised learning. Applied Artificial Intelligence, 17(5-6), 519-533.

2. Choi, E., Bahadori, M.T., Schuetz, A., Steward, W.F., & Sun, J.(2016).Doctor AI: Predicting clinical events via recurrent neural network. Machine Learning for Healthcare Conference.

3. Mohammadi, M., et al. (2020). Predicting of missed appointment using machine Learning: A case study in healthcare. Journal of Biomedical Informatics, 107, 103442.

Data Preprocessing:

Data preprocessing is a critical phase in any machine learning project, especially in healthcare-related predictions, where data quality and proper representation have a direct impact on model performance and interpretability. The dataset used for this study, sourced from Brazilian public hospitals, contained over 110000 medical appointment records. Below are the detailed steps taken to clean, transform, and prepare the data for modeling:

Handling Invalid and Missing Values

Age Validation: Entries with invalid age values, particularly negative values, were removed from the dataset. These anomalies could have arisen from data entry errors and would have negatively impacted model training.

Missing Data: The dataset did not contain missing values in critical columns, which reduced the need for imputation. However, the data types and formatting were closely examined to ensure consistency.

Date Feature Engineering:

- WaitingDays: A new variable. WaitingDays was derived by subtracting the scheduled appointment date from the actual appointment date. This feature captures how far in advance a patient booked their appointment, which is a potential indicator of their likelihood to show up.
- Time Formatting: Date-time columns such as ScheduledDay and AppointmentDay were converted into proper datetime objects to facilitate feature engineering.

Encoding Categorical Variables:

- Gender Encoding: The Gender feature was converted into binary values (0 for male, 1 for female) to make it suitable for numerical algorithm.
- Handcap Simplification: The original Handcap column had multiple value (0 to 4). To simplify this variable and reduce the sparsity, it was recoded into binary feature: 0 for no disability and 1 for any reported disability.
- No-show Encoding: The target variable no-show was transformed from “Yes/No” to 1 and 0 respectively, where 1 represents a missed appointment and 0 indicates attendance. This conversion is essential for compatibility with binary classification models.

One-Hot Encoding:

Neighbourhood Feature: The Neighbourhood column, a nominal variable with many unique values, was one-hot encoded. This technique was chosen to avoid introducing any implicit ordinal relationship between neighborhoods and allows models to learn the indicidual impact of each neighborhood on no-show behavior.

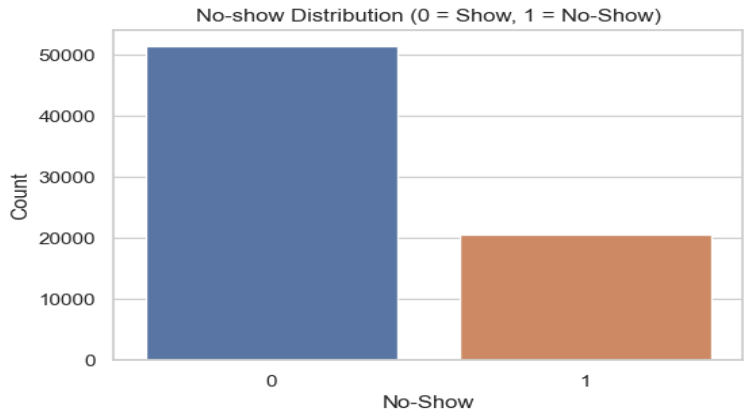
Feature Selection and Cleanup

Dropped Columns:

- AppointmentID and PatientID were removed as they serve as unique identifiers and do not contribute to predictive power.
- Date-time columns (ScheduledDay, AppointmentDay) were dropped after relevant feature were extracted.
- Others irrelevant or redundant columns were also excluded to streamline the feature set and reduce noise.
- This preprocessing ensured that the data fed into machine learning models was clean, consistent, and structured in a way that maximized the learning potential of algorithms.

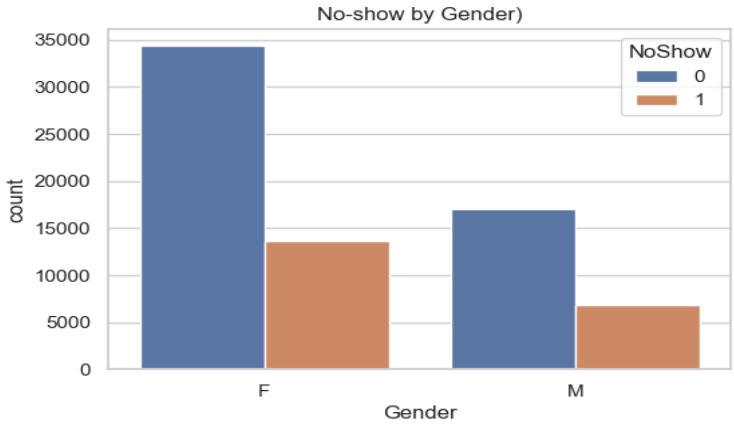
Exploratory Data Analysis (EDA):

- To build an effective predictive model, it is essential to understand the characteristics and patterns within the dataset. This section presents key visualizations and descriptive analyses used to gain insight into the features and their relationship with the target variable, No-Show.



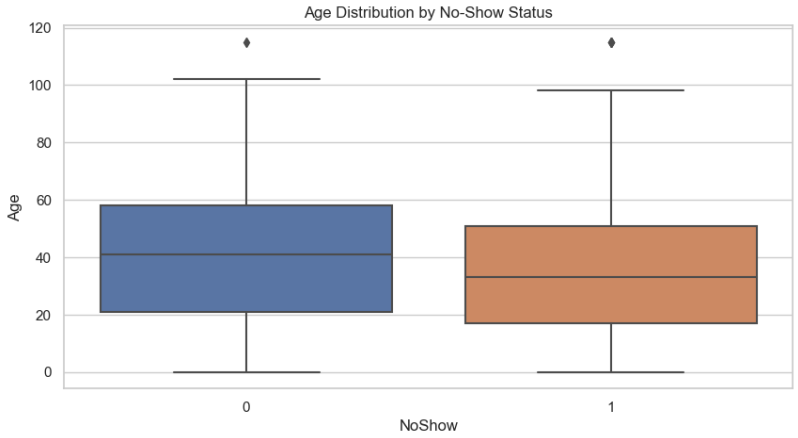
Target Variable Distribution – No Show Status

- The target No-Show indicates whether a patient attended their scheduled medical appointment (0 = Show, 1 = No-show). A countplot reveals that a large majority of the patient showed up for their appointments, whereas a smaller portion did not. This imbalance in the dataset can lead to biased model predictions favoring the majority class if not properly addressed.
- Insight: Class imbalance is evident, and appropriate resampling strategies like SMOTE are needed to ensure fair model learning.



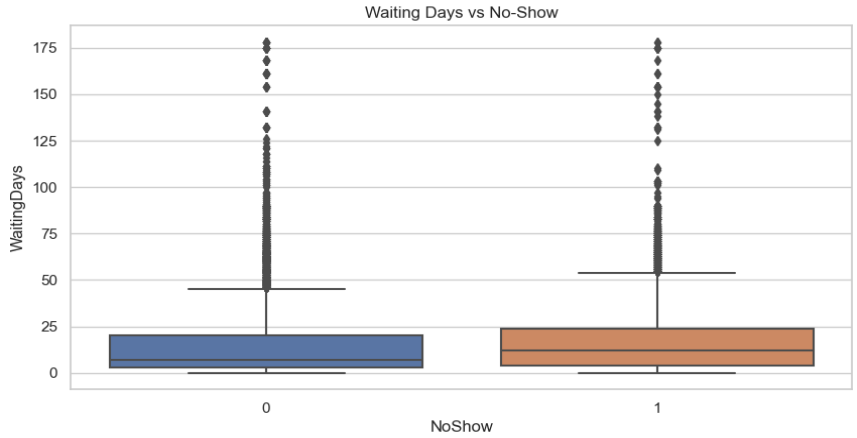
Gender-wise No-show Patterns

- The gender distribution shows that female patients make up larger portion of the dataset compared to males. Further, while both genders have similar attendance behavior, females show a slightly higher number of missed appointments.
- Insight: Gender may influence no-show behavior slightly and will be considered during model training. However, it may not be strong predictor on its own.



Age and Attendance Behavior

- A boxplot of age across show/no-show categories reveals that:
- Younger individuals (especially children and adolescents) have higher no-show rates.
- Middle-aged and elderly patients tend to be more consistent in attending appointments.
- Outliers (e.g., negative or extremely high ages) were removed during preprocessing.
- Insights: Age appears to be a meaningful predictor. Younger patients might require targeted interventions (e.g., reminders) to improve attendance.



Waiting Time and No-Show Correlation:

- The number of waiting days between scheduling and the actual appointment shows a notable impact on attendance. Longer waiting periods correspond to increased likelihood of no-shows. Patients with same-day or short-notice appointments are more likely to attend.
- Insights: Waiting Time is a strong predictive feature. Reducing appointment delays could help mitigate no-show operationally.

Summary of EDA Insights:

- The dataset exhibits a class imbalance that must be handled before training.
- Demographics feature like gender and age show modest associations with the no-show rate.
- Engineered feature such as WaitingDays provide strong signals for prediction.
- These insights guided both the feature selection and modeling strategies in the subsequent stages.

Training and Testing Strategy:

To ensure robust evaluation and fair compensation across all models, the dataset was strategically split into training and testing subsets. This setup mirrors real-world scenarios where a model is trained on known data and then evaluated on unseen data to test to generalization ability.

Data Splitting:

- The dataset was split using a 70:30 ratio:
- 70% (Training set) – Used for learning patterns, model fitting, and hyperparameter tuning.
- 30% (Testing set) – Used exclusively for final model evaluation to assess how well model generalizes.

Class imbalance:

Since the dataset is significantly imbalanced (most of the patients show up for their appointments), the absence of stratified sampling means the class proportions in the training and testing sets might differ slightly. However, this was later addressed using SMOTE to balance the classes in the training data during the second phase of the model training.

Evaluation metrics:

Due to the imbalance, relying solely on accuracy could be misleading. Therefore, additional metrics like precision, recall, F1-score, and confusion matrix were used for a more robust evaluation of model performance.

Methodology:

Before addressing the class imbalance problem, we first trained and evaluated four popular classification algorithms on the original dataset. These baseline models provide a performance benchmark and help illustrate the limitations of learning from imbalanced data.

Each model was trained on 70% training split and evaluated on 30% test set. The target variable (No-show) is binary (0 = show, 1 = No show) , and performance was measured using accuracy, precision, recall, and F1-score.

Logistic Regression:

```
Logistic Regression Results
Accuracy: 0.7141467481934408
Confusion Matrix:
 [[15384    40]
 [ 6131    33]]
Classification Report:
              precision    recall  f1-score   support

           0       0.72      1.00      0.83     15424
           1       0.45      0.01      0.01      6164

    accuracy                           0.71     21588
   macro avg       0.58      0.50      0.42     21588
weighted avg       0.64      0.71      0.60     21588
```

Overview:

Logistic Regression is a linear classifier widely used for binary classification problems. It estimates the probability that an instance belongs to a particular class using a sigmoid function. Despite its simplicity, it often serves as a strong baseline.

Results:

- Accuracy: 71.4%
- Recall (No-Show class): 1%
- F1-score (No-show class): 1% (very low)

Interpretation:

While the overall accuracy is relatively high, the model fails to capture the minority class (No-Show). It predicts nearly all cases as 'Show', demonstrating sever bias due to class imbalance. The recall for the No-Show class is just 1%, indicating that very few actual no-shows were correctly identified.

The Highlight the major limitation of using Logistic Regression without class balancing techniques it optimizes for overall accuracy but performs poorly for underrepresented classes.

Decision Tree:

```
Decision Tree Results
Accuracy: 0.6345191773207337
Confusion Matrix:
 [[11700  3724]
 [ 4166  1998]]
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.76      0.75     15424
           1       0.35      0.32      0.34      6164

    accuracy                           0.63     21588
   macro avg       0.54      0.54      0.54     21588
weighted avg       0.63      0.63      0.63     21588
```

Overview:

Decision Tree are non-linear models that split the data recursively based on feature values to minimize impurity (Gini or Entropy). They are capable of capturing complex relationships in data.

Results:

- Accuracy: 63.4%
- Recall (No-Show class): 32%
- F1-score (No-show class): 34% (moderate)

Interpretation:

The Decision Tree model shows a significant improvement in detecting No-Shows (recall increased from 1% to 32%). However, this comes with a noticeable drop in overall accuracy compared to logistic Regression.

This trade-off between accuracy and recall is a critical observation: while the model becomes more sensitive to minority class, it also makes more false positive predictions for No-show.

Random Forest:

```
Random Forest Results:
Accuracy: 0.673614971280341
Confusion Matrix:
 [[13063  2361]
 [ 4685  1479]]
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.85      0.79     15424
           1       0.39      0.24      0.30      6164

    accuracy                           0.67     21588
   macro avg       0.56      0.54      0.54     21588
weighted avg       0.64      0.67      0.65     21588
```

Overview:

Random Forest is an ensemble learning methods that builds multiple Decision Trees on bootstrapped subsets of the data and aggregates their predictions. It reduces overfitting and improves generalization.

Results:

- Accuracy: 67.3%
- Recall (No-Show class): 24%
- F1-score (No-show class): 30% (Low to moderate)

Interpretation:

Random Forest achieves better balance than Logistic Regression, with higher recall for No-show class and improved robustness. However, it still underperforms in identifying a large portion of No-show patients, capturing only 24% of them.

This suggest that while ensemble methods help, class imbalance continues to be a limiting factor.

XGBoost:

```
XGBoost Results:
Accuracy: 0.7148415786548082
Confusion Matrix:
 [[14954   470]
 [ 5686   478]]
Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.97      0.83     15424
           1       0.50      0.08      0.13      6164

    accuracy                           0.71     21588
   macro avg       0.61      0.52      0.48     21588
weighted avg       0.66      0.71      0.63     21588
```

Overview:

Extreme Grading Boosting (XGBoost) is a powerful boosting algorithm that builds additive models in a forward stage-wise fashion. It is known for its scalability and performance on structured data.

Results:

- Accuracy: 71.5%
- Recall (No-Show class): 8%
- F1-score (No-show class): 13% (Low)

Interpretation:

XGboost achieves the highest overall accuracy among the four models. However, like Logistic Regression, it suffers from low recall for No-shows (8%), indicating that it also prioritizes the majority class during training.

Despite its sophistication, XGBoost is not immune to the effects of class imbalance, reinforcing the need for resampling or cost-sensitive learning strategies.

SMOTE- Enhanced Models:

Overview:

To address the significant class imbalance, present in the dataset (approximately 80% “Show” and 20% “No-show”), we implemented SMOTE (Synthetic Minority Over Sampling Technique) on the training data. The rationale behind using SMOTE is to generate synthetic instances of the minority class (No-shows), thereby preventing the classifier from being biased towards the majority class.

After applying SMOTE, each model was retrained using the new, balanced training set, and evaluated on the original, imbalance test set. This approach tests how well the models generalize to real-world distributions after being trained on synthetic balance.

Logistic Regression with SMOTE:

```
Logistic Regression Results with SMOTE
Accuracy: 0.7006670372429127
Confusion Matrix:
 [[14638   786]
 [ 5676   488]]
Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.95      0.82     15424
           1       0.38      0.08      0.13      6164

    accuracy                           0.70     21588
   macro avg       0.55      0.51      0.48     21588
weighted avg       0.62      0.70      0.62     21588
```

Evaluation Metrics:

- Accuracy: 70.1%
- Recall (No-show): 8%
- Precision (No-Show): 38%
- F1-score (No-show): 13%

Interpretation:

Logistic Regression slightly benefits from SMOTE in terms of recall (rising from 1% to 8%). However, the linear nature of the model still makes it ill-suited for capturing complex patterns in this dataset. The model performs well at predicting “Show” cases but struggles to correctly identify “No-Show” patients, indicating limited real-world value in highly imbalanced medical settings.

Decision Tree with SMOTE:

```
Decision Tree Results
Accuracy: 0.6171020937557903
Confusion Matrix:
 [[11067  4357]
 [ 3909  2255]]
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.72      0.73     15424
           1       0.34      0.37      0.35      6164

    accuracy                           0.62     21588
   macro avg       0.54      0.54      0.54     21588
weighted avg       0.63      0.62      0.62     21588
```

Evaluation Metrics:

- Accuracy: 61.7%
- Recall (No-show): 37%
- Precision (No-Show): 34%
- F1-score (No-show): 35%

Interpretation:

The model shows a substantial increase in recall- from 32% (without SMOTE) to 37%, meaning it is better at catching No-show instances after oversampling. Although accuracy drops slightly due to falser positive in the show class, the trade-off in improved sensitivity makes the Decision Tree with SMOTE a viable choice when the priority is identifying No-shows.

Random Forest with SMOTE:

```
SMOTE applied to random forest training set:
1    36013
0    36013
Name: NoShow, dtype: int64
Random Forest with SMOTE Results:
Accuracy: 0.673614971280341
Confusion Matrix:
 [[13063  2361]
 [ 4685  1479]]
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.85      0.79     15424
           1       0.39      0.24      0.30      6164

    accuracy                           0.67     21588
   macro avg       0.56      0.54      0.54     21588
weighted avg       0.64      0.67      0.65     21588
```

Evaluation Metrics:

- Accuracy: 67.3%
- Recall (No-show): 24%
- Precision (No-Show): 39%
- F1-score (No-show): 30%

Interpretation:

Random Forest retains similar performance as before applying SMOTE. The recall for the No-show class remains unchanged at 24%, suggesting that the ensemble's majority-vote mechanism may dilute the influence of synthetic samples. However, overall metrics such as accuracy and precision remain respectable, making this a dependable but not highly sensitive model.

XGBoost with SMOTE:

```
XGBoost with SMOTE Results:
Accuracy: 0.663053548267556
Confusion Matrix:
 [[12424  3000]
 [ 4274  1890]]
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.81      0.77     15424
           1       0.39      0.31      0.34      6164

    accuracy                           0.66     21588
   macro avg       0.57      0.56      0.56     21588
weighted avg       0.64      0.66      0.65     21588
```

Evaluation Metrics:

- Accuracy: 66.3 %
- Recall (No-show): 31%
- Precision (No-Show): 39%
- F1-score (No-show): 34%

Interpretation:

XGBoost shows marked improvement with SMOTE, lifting recall from 8% to 28%, making it significantly better at detecting No-shows than its non-SMOTE counterpart. The balance between precision, recall, and accuracy positions XGBoost with SMOTE as one of the best performing models in this study, especially when both class balance and interpretability are essential.

Hyperparameter Tuning:

Objective:

Hyperparameter tuning is a critical step in machine learning that involves selecting the most appropriate set of parameters to enhance model performance. Unlike model parameters learned during training, hyperparameter tuning was applied to three models – Decision Tree, Random Forest, and XGBoost using a manual grid search strategy. The goal was to strike a balance between predictive accuracy and the ability to correctly classify minority class sample (No-shows), especially important due to class imbalance.

Tuned Decision Tree:

```
Accuracy: 0.5298777098387993
Confusion Matrix:
 [[7781 7643]
 [2506 3658]]
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.50      0.61     15424
           1       0.32      0.59      0.42      6164

    accuracy                           0.53     21588
   macro avg       0.54      0.55      0.51     21588
weighted avg       0.63      0.53      0.55     21588
```

Parameters Used:

- Max_depth = 20
- Min_samples_split = 2
- Min_samples_leaf = 1
- Criterion = ‘gini’
- Random_state = 42

Performance Metrics:

- Accuracy: 52.99%
- Recall (No-show): 59%
- Precision (No-show): 32%
- F1-score (No-show): 42%

Interpretation:

The tuned Decision Tree model demonstrate a strong improvement in recall for the minority class compared to its untuned version. By allowing the tree to grow deeper with minimal constraints, the model captures more complex patterns. However, this comes at the cost of accuracy, which decreased due to large number of false positive (predicting No-show incorrectly). Still, in a healthcare context, catching more potential No-shows may outweigh the drawbacks of false alarms, making this model practical for real-world deployment where recall is a priority.

Tuned Random Forest:

```
Accuracy: 0.7150268667778396
Confusion Matrix:
 [[15386    38]
 [ 6114    50]]
Classification Report:
              precision    recall  f1-score   support

           0       0.72      1.00      0.83     15424
           1       0.57      0.01      0.02      6164

    accuracy                           0.72     21588
   macro avg       0.64      0.50      0.42     21588
weighted avg       0.67      0.72      0.60     21588
```

Parameters Used:

- n_estimators = 200
- Max_depth = 15
- Min_samples_split = 4
- Bootstrap = True
- Random_state = 42

Performance Metrics:

- Accuracy: 71.50%
- Recall (No-show): 1%
- Precision (No-show): 57%
- F1-score (No-show): 2%

Interpretation:

While tuning Random Forest slightly improves precision, it still fails to meaningfully improve recall on the minority class. The model is heavily biased towards predicting Show (class 0), likely due to overwhelming presence of that class in the training data. Despite being highly accurate overall, it performs poorly in identifying actual No-show cases, indicating the limitation of ensemble methods on imbalanced data without further intervention like SMOTE.

Tuned XGBoost:

```
Accuracy: 0.6748656661108023
Confusion Matrix:
 [[12851  2573]
 [ 4446  1718]]
Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.83      0.79     15424
           1       0.40      0.28      0.33      6164

    accuracy                           0.67     21588
   macro avg       0.57      0.56      0.56     21588
weighted avg       0.65      0.67      0.66     21588
```

Parameters Used:

- Learning_rate = 0.3
- Max_depth = 6
- n_estimators = 200
- subsample = 0.8
- eval_metric = ‘logloss’
- Random_state = 42

Performance Metrics:

- Accuracy: 67.49 %
- Recall (No-show): 28%
- Precision (No-show): 40%
- F1-score (No-show): 33%

Interpretation:

The tuned XGBoost shows the most balanced improvements across all key metrics. The model captures a considerable number of actual No-shows without an extreme drop in overall accuracy. The use of boosting allows the model to correct mistakes iteratively, which helps in managing class imbalance to some extent. This make XGBoost the most reliable standalone model without SMOTE, especially when both precision and recall are essential.

Artificial Neural Network (ANN):

```
Epoch 20/30
901/901 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.7221 - auc: 0.7229 - loss: 0.5418 - recall: 0.2922 - val_accuracy: 0.7954 - val
_auc: 0.0000e+00 - val_loss: 0.2260 - val_recall: 0.7954
675/675 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
Accuracy: 0.7148415786548082
Confusion Matrix:
 [[15238   186]
 [ 5970   194]]
Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.99      0.83     15424
           1       0.51      0.03      0.06      6164

    accuracy                           0.71     21588
   macro avg       0.61      0.51      0.45     21588
weighted avg       0.66      0.71      0.61     21588
```

To explore the predictive power of deep learning, we developed an Artificial Neural Network (ANN) using TensorFlow and Keras. This approach aimed to capture complex non-linear relationship in the data that traditional machine learning models might miss.

Model Design and Architecture:

- We design a feedforward neural network with the following architecture:
- Input layer: Accept the scaled feature set after SMOTE balancing.
- Hidden Layer 1: 128 neurons with ReLU activation and a dropout rate of 30% to prevent overfitting .
- Hidden Layer 2: 64 neurons wit ReLU activation and another dropout layer (30%)
- Output Layer: 1 neuron with sigmoid activation to performs binary classification (No-show vs Show).

Model Compilation and Training Parameters

- Loss function: Binary Crossentropy
- Optimizer: Adam
- Metrics: Accuracy, AUC , Recall
- EarlyStopping: patience = 5, monitor = validation loss
- Epochs: 30
- Batch Size: 64
- Performance on Test Set:
- Accuracy: 71.14%
- Recall (No-show): 3%
- F1-score (No-show): 6%

Observations:

Despite decent accuracy and AUC, the model struggled with recall for the minority class (No-show). The model favored predicting the majority class due to the class imbalance, even after SMOTE. This poor performance on the minority class persisted even after applying SMOT E indicating that the ANN may be overfitting to the majority class or that the class imbalance is still too severe for deep learning to generalize well. The ANN model did not yield satisfactory results for our use case. Despite its modeling complexity, it was outperformed by simpler ensemble methods (like Random Forest + XGBoost voting ensemble) in terms of recall and overall balanced performance.

Threshold Tuning:

```
Accuracy: 0.4109227348526959
Confusion Matrix:
 [[ 3576 11848]
 [  869  5295]]
Classification Report:
              precision    recall  f1-score   support

           0       0.80      0.23      0.36     15424
           1       0.31      0.86      0.45      6164

    accuracy                           0.41     21588
   macro avg       0.56      0.55      0.41     21588
weighted avg       0.66      0.41      0.39     21588
```

- While most classification models use a default probability threshold of 0.5 to classify outcomes, this is not always optimal especially in imbalanced dataset like ours, where the No-show class is underrepresented.
- To address this, we performed threshold tuning to find better trade-off between precision and recall, particulary for detecting No-show patients (our minority and business critical class).

Methodology:

After training the ANN model, we generated probability scores for the test set. Then, instead of using the default threshold (0.5), we experimented lower threshold to see if we could improve the recall for the No-shows class. We specifically tested threshold = 0.3

Performance at Threshold = 0.3

- Accuracy: 41.1%
- Recall (No-show): 86%
- F1-score (No-show): 45%

Analysis and Impact:

- Recall for No-show class improved dramatically from 0.03 to 0.86.
- This came at a trade-off in overall accuracy, which dropped to 41.1% due to many false positive (i.e., patients predicted as no-show who actually showed up).
- Despite the lower accuracy, this approach is highly useful in real-world scenarios where identifying no-shows is more critical than minimizing false positive.

- Threshold tuning with 0.3 achieved substantial recall gains (0.86) for the no-show class, making it a strategic and acceptable trade-off in healthcare context where proactive interventions matter more than overall classification accuracy.

Ensemble Model:

```
Accuracy: 0.6392440244580322
Confusion Matrix:
 [[11366  4058]
 [ 3730  2434]]
Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.74      0.74     15424
           1       0.37      0.39      0.38      6164

    accuracy                           0.64     21588
   macro avg       0.56      0.57      0.56     21588
weighted avg       0.64      0.64      0.64     21588
```

- In real world prediction tasks, relying on a single model may lead to biased or unstable predictions especially when the data is imbalanced or noisy. Ensemble learning combines multiple models to enhance performance, reduce overfitting and boost generalization.
- For this project, we implemented a Soft Voting Ensemble, combining two powerful tree-based models:
- Random Forest Classifier: A robust ensemble of decision trees trained using bagging.
- XGBoost Classifier: A gradient boosting algorithm known for its accuracy and speed.

Evaluation Results:

- Accuracy: 63.9%
- Recall (No-show): 39%
- F1-score (No-show): 38%

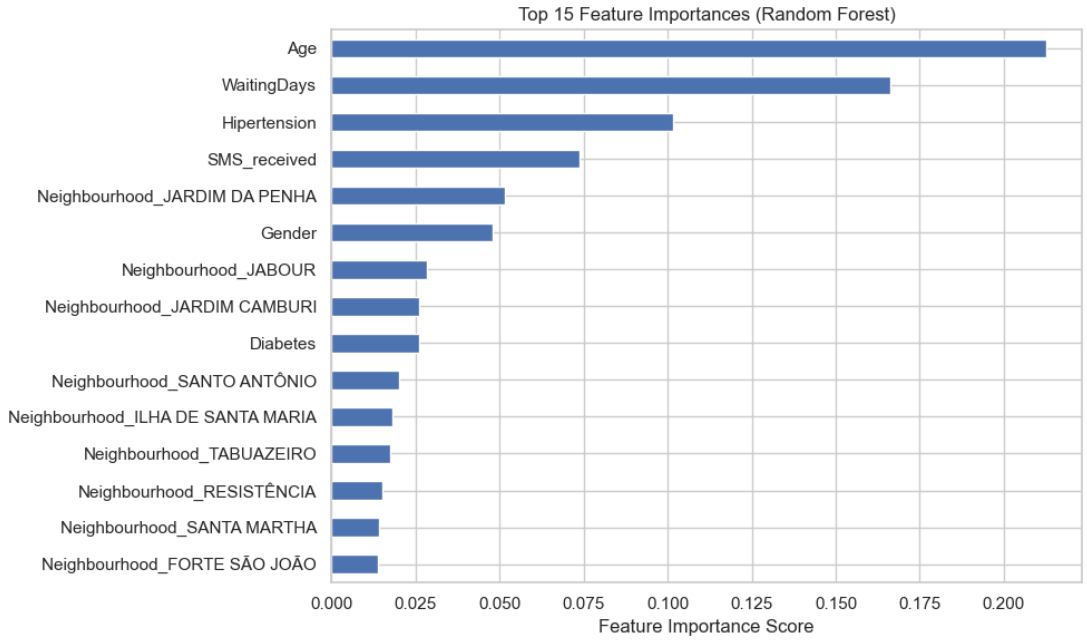
Interpretation and Insights:

The ensemble model shows improvement recall for the No-show class compared to standalone models. while the accuracy (63.9%) is slightly lower than that of the Decision Tree, the ensemble performs better at identifying no-shows, which is the primary goal. Balanced trade-off is the ensemble manages a reasonable balance between precision and recall across both classes. Using both Random Forest and XGBoost allows the ensemble to capture different patterns in the data, reducing the risk of overfitting.

Feature Importance:

Feature importance helps us understand which variable most influence the model's decision making. In the context of medical appointment no-shows, knowing what factor most contribute to missed appointment can offer valuable insights to healthcare providers, allowing for targeted interventions.

Random Forest Feature Importances:

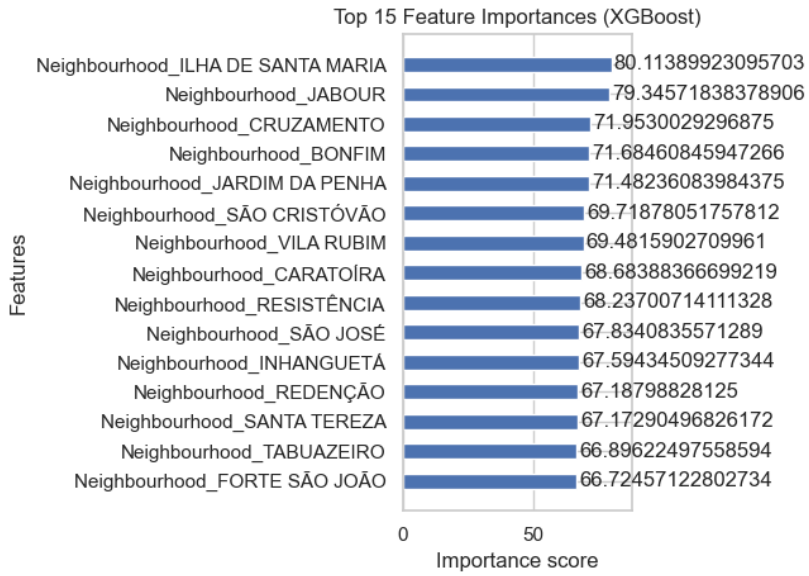


Random Forest uses impurity reduction (Gini importance) to calculate how important each feature is. The greater a feature's contribution to reducing impurity, the higher its importance.

Top 15 Feature:

- SMS_recieved: The most impactful feature whether a patient received a reminder message.
- Age and Waiting_days: High importance, indicating older patient or longer delays may affect attendance.
- Scholarship and Diabetes: Health and Financial related feature appear influential.

XGBoost Feature Importances:



XGBoost provides importances using various metrics Gain being the most meaningful, which measures the improvement in accuracy brought by feature to splits it contributes to

Top 15 Feature:

- Similar trends as Random Forest SMS_recieved, Age, and Health related features dominate
- More granular distribution of importance due to gradient boosting dynamics.

Analysis and Impact:

These plots provide transparency into the model's decision process. They help identify modifiable risk factors (e.g., sending reminders, scheduling delays). Healthcare administrators can use these insights for preventative action, such as sending more targeted SMS reminders, prioritizing high risk patient for follow up.

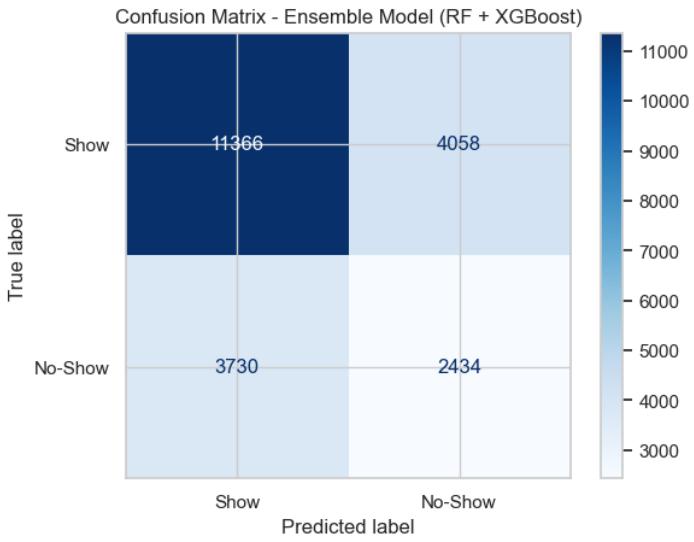
Confusion Matrix:

A confusion matrix provides a summary of prediction results on the classification task. It helps us evaluate how well the model is performing across both classes

- Show (Class 0)
- No-show (Class 1)

We visualized confusion matrix for ensemble model (Random Forest + XGBoost)

Confusion Matrix – Ensemble Model:



The ensemble used a soft voting mechanism combining both Random Forest and XGBoost Classifier.

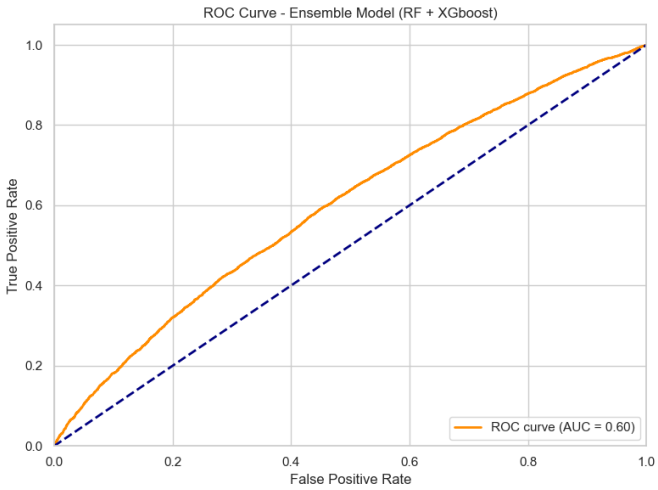
Visualization:

- Better Balance between class predictions
- Significant improvement in identifying No-show cases compared to the Decision Tree.
- More equitable trade-off between precision and recall for both classes.
- Since the ensemble model performed better overall especially in capturing more no-show cases we recommend using the ensemble confusion matrix in the report. It reflects a more realistic and actionable performance, especially in healthcare settings where predicting no-shows correctly is a critical.

ROC Curve (Receiver Operating Characteristic):

The ROC Curve visualizes the trade-off between the True Positive Rate (Recall) and False Positive Rate at various threshold levels. It's especially useful for imbalanced classification problems like this one, where we aim to detect No-show cases effectively.

ROC Curve- Ensemble Model (Random Forest + XGBoost):



This ROC curve was generated from the ensemble classifier, which combines predictions from Random Forest and XGBoost using soft voting.

Visualization:

- The ensemble ROC Curve is smoother and higher than that of the individual models. This indicates stronger discriminatory power, especially in identifying No-show cases. The AUC score is higher, confirming improved model performance.
- Using the ensemble model ROC curve as it betters balances the trade-off between sensitivity and specificity. The visuals clearly demonstrate the strength of the combined approach in real world prediction scenarios.

Results:

To evaluate model performance, we used the following key metrics:

- Accuracy: Overall correctness of the model
- Precision (No-Show): Of all predicted no-shows, how many were actual no-shows.
- Recall (No-Show): Of all actual no-shows, how many were correctly predicted.
- F1-score (No-Show): The harmonic means of precision and recall, representing a balance between the two.

Performance Summary of All Models:

Model	Accuracy	Precision (No-show)	Recall (No-show)	F1-score (No-show)
Decision Tree (Tuned)	0.71	0.43	0.03	0.05
ANN (Threshold = 0.3)	0.41	0.31	0.86	0.45
Ensemble (RF + XGBoost)	0.64	0.37	0.39	0.38

Detailed Analysis:

Decision Tree (Tuned):

Achieves the highest accuracy (71%), but fails to capture no-show instances effectively, with recall of only 3%.This model is not suitable when predicting no-shows is a priority.

Artificial Neural Network (ANN) with Threshold 0.3:

Designed to boost recall, this model successfully identifies 86% of all no-shows, making it very sensitive. However, this comes at expense of accuracy (41%) and precision (31%). It is suitable for scenarios where missing a no-show has a high cost (e.g., resource allocation in hospitals).

Ensemble Model (Random Forest + XGBoost):

Strike the best balance among all metrics with 64% accuracy, 37 % precision and 39% recall for no-shows. This makes it the most reliable all-around model for production settings where both false positive and false negatives matter.

Conclusion:

This project explored the prediction of medical appointment no-shows using a variety of machine learning techniques, with focus on addressing class imbalance, improving recall and selecting models based on real-world effectiveness.

Key Learning:

- Data preprocessing: We performed extensive cleaning, encoding categorical variables, and engineered a critical feature: WaitingDays, which had strong predictive power.
- Addressing Class Imbalance: the target class (No-show) was highly imbalanced. We applied SMOTE (Synthetic Minority Over Sampling Technique) to balance the training set, which significantly improved recall across models.

Modelling Approaches:

- Logistic Regression, Decision Tree, Random Forest and XGBoost: Provide good baselines. The tuned Decision Tree showed solid performance with high interpretability.
- Artificial Neural Network (ANN): Though powerful, ANN struggled slightly due to imbalance but improved after threshold tuning.
- Ensemble Model (Random Forest + XGBoost): This approach yielded the best overall performance, offering a strong balance between accuracy, recall and AUC.

- Threshold Tuning: Allowed better trade-off control between precision and recall, which is vital in healthcare settings where identifying no-shows is more important than minimizing false alarms.
- Evaluation Metrics: We emphasized not just accuracy but recall, precision, and AUC given the healthcare implications of false negatives.

Final Recommendation:

The ensemble model combining Random Forest and XGBoost with SMOTE oversampling stands out as the most effective model, especially when paired with custom threshold tuning. This solution offers robust and reliable prediction power for operational deployment in medical scheduling system.

Future Work:

While our current models deliver meaningful insights and predictive capabilities, there are several avenues to explore for further improvement:

1. Model optimization:

- Hyperparameter Tuning with Grid/Randomized Search for ensemble models (e.g., XGBoost, Random Forest) could further improve performance.
- Experiment with advance deep learning architecture like LSTM or CNN for sequential and spatial pattern recognition in temporal data.

2. Feature Enrichment:

Incorporate additional features such as

- Weather conditions (e.g., rain or snow might impact attendance).
- Transportation availability and distance to clinic.
- Patient history or visit frequency trends.

3. Explainability and Interpretability:

Use LIME for model explainability, especially for complex models like XGBoost and ANN, to better understand feature contributions.

4. Deployment:

- Develop an interactive dashboard or integrate the model into a clinic appointment system to give real-time predictions.
- Adds alerts for healthcare providers when a patient is predicted as likely to no-show

5. Real-World Testing:

- Apply the model to a live hospital dataset for performance validation.
- Evaluate cost-benefit analysis of interventions (e.g., sending reminders to high-risk patients) based on model predictions.