Medical Appointment No Shows

Lucas Quinlan
Angel Conde
Joel Ortega
Joseph Yiga

Why is studying medical appointment attendance important?

Missed Appointments, Missed opportunities: Tacklin The Patient No-Show Problem

… “no-shows” cost the U.S. health care system more than $150 billion a year.

For individual Physicians the ¨no-shows” cost $200 per unused time slot. After all, medical practices still have to pay their staffs and cover expenses like rent and the cost of equipment.

Sachin H. Jain. Forbes

No-show effect: Even one missed appointment risks retention.

For patients ages 31-45, 46-60, and 61-plus with one or more “no-shows”, compared to their pairs no “no-shows”, their attrition rate went up to 57, 63, and 73 percent respectively.

Patients suffering from chronic illnesses are at a high risk. The study shows it’s harder for them to return after the first missing appointment.

Chris Hayhurst. Athenahealth

Each appointment taken and missed, represents a person with a health issue that could not be addressed at that moment.

The project

Goal: For our project we want to use the dataset to explore factors that contribute to missed appointments, and try to develop a model to predict patients that would be at risk of missing appointments.

Dataset: Medical Appointment No Shows

14 variables associated with 110,527 medical appointments.

- Patient ID,
- GenderAge,
- Day of Appointment,
- Day Appointment was Scheduled,
- Neighborhood,
- Presence of Various Known Pre-existing conditions,
- Appointment Reminder Sent, and Outcome

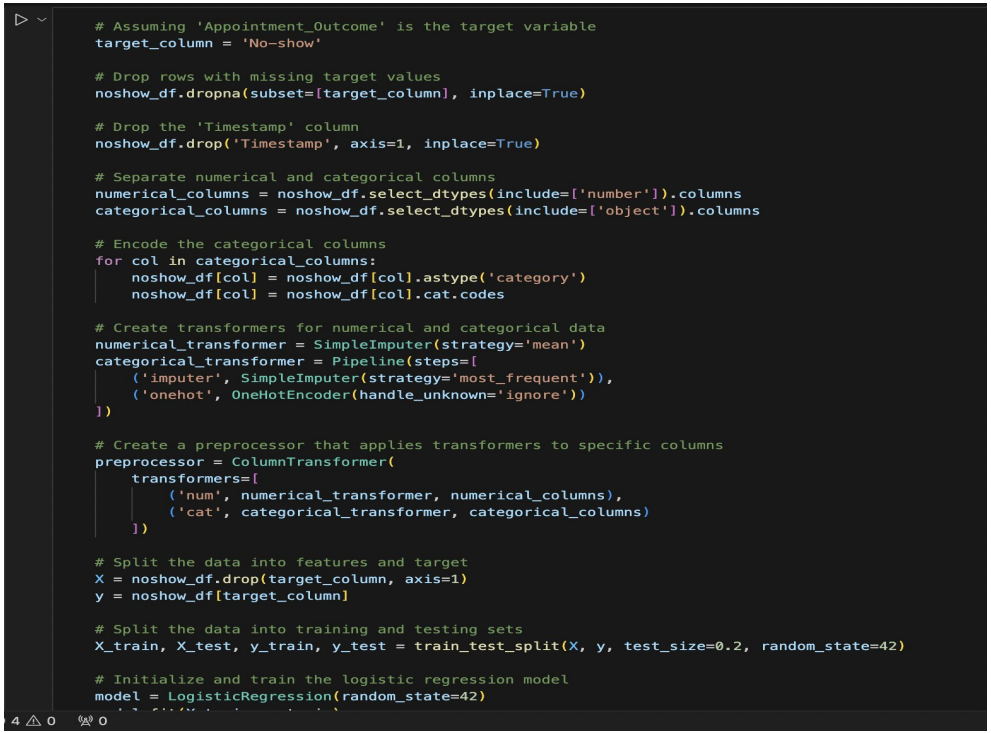
Data Cleaning and Pre Processing

A quick overview of the steps taken to ensure our data set was of sufficient quality and suitable for the prompt:

- Some heat maps and bar charts were utilised to identify through visual representation, the distribution of missing values
- Examples of dropped columns included the Appointment ID column and dates.
- Outlier detection was also incorporated using visualizations and they were dropped.

Handling of categorical variables

- We identified the categorical columns in the data set as the ones with objects
- We utilised one-hot encoder as it was the ideal encoder for this particular data set which included nominal data in no inherent order
- Utilising one-hot encoder for this particular data set would also drive interpretability



```python
# Assuming 'Appointment_Outcome' is the target variable
target_column = 'No-show'

# Drop rows with missing target values
noshow_df.dropna(subset=[target_column], inplace=True)

# Drop the 'Timestamp' column
noshow_df.drop('Timestamp', axis=1, inplace=True)

# Separate numerical and categorical columns
numerical_columns = noshow_df.select_dtypes(include=['number']).columns
categorical_columns = noshow_df.select_dtypes(include=['object']).columns

# Encode the categorical columns
for col in categorical_columns:
    noshow_df[col] = noshow_df[col].astype('category')
    noshow_df[col] = noshow_df[col].cat.codes

# Create transformers for numerical and categorical data
numerical_transformer = SimpleImputer(strategy='mean')
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

# Create a preprocessor that applies transformers to specific columns
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_columns),
        ('cat', categorical_transformer, categorical_columns)
    ])

# Split the data into features and target
X = noshow_df.drop(target_column, axis=1)
y = noshow_df[target_column]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the logistic regression model
model = LogisticRegression(random_state=42)
```

Exploratory Data Analysis - EDA

- EDA is conducted to understand the structure of data and assist in model/ feature selection for the machine learning algorithm
- Key component of EDA is the use of histograms to showcase distribution of relevant variables, showing skewness, and spread.
- The median position in the box plots for example, helped identify the skewness of the data set
- EDA helped us identify anomalies not obvious in the raw data
-

Model Implementation and Optimization

- Utilizing Logistic Regression based on the specifics of the data set, we were able to initialize, train and evaluate the predictive model
- Preprocessing techniques were utilised to ensure the data was ready for training.
- In the metrics selection process, one of the factors considered involved minimizing false positives, and this encouraged precision
- One of the limitations we came across was the sensitivity of the Linear Regression Model to outliers and this leads to bias, and also in the event that the variables we utilised did not have a linear relationship, this specific model would not be utilised.
- Iterative changes to the code were carried out to optimize the model.
- Examples include; feature engineering (inputting or dropping columns), using different Encoders
- One of our models involved neural networks, a step in the process optimization process involved adjusting the number of layers in each test

Preprocessing and Data Engineering

- Removed outliers
- Created new column for number of days between when the appointment was scheduled and when the appointment occurred
- Mapped target outcome to numeric values
- Removed patient/appointment ID columns

```
# Preprocessing
# Remove row(s) with age <0 and >100
noshow_df = noshow_df[noshow_df['Age'] >= 0]
noshow_df = noshow_df[noshow_df['Age'] < 100]

# Convert dates to datetime
noshow_df['ScheduledDay'] = pd.to_datetime(noshow_df['ScheduledDay']).dt.normalize()
noshow_df['AppointmentDay'] = pd.to_datetime(noshow_df['AppointmentDay']).dt.normalize()

# Calculate the difference in days
diff = (noshow_df['AppointmentDay'] - noshow_df['ScheduledDay']).dt.days

# Insert 'Days Diff' column
noshow_df.insert(4, 'Days Diff', diff)

# Rename No-show column to Outcome and map values
noshow_df['Outcome'] = noshow_df['No-show']
noshow_df = noshow_df.drop(columns=['No-show'])
noshow_df['Outcome'] = noshow_df['Outcome'].map({'No':'1', 'Yes':'0'})

# Drop Date and ID columns
noshow_df = noshow_df.drop(columns=['ScheduledDay','AppointmentDay', 'PatientId', 'AppointmentID'])
```

Neural Net - 1st Attempt

- Used all features from preprocessing
- One-hot encoded categorical features
- 80% used for training, 20% for testing
- 2 layers and 50 epochs
- Good accuracy, but poor precision and recall for No-Show predictions

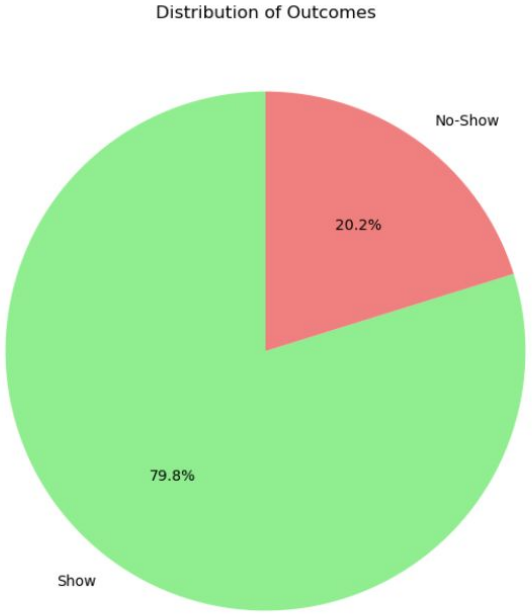
	Model 1
Training Data	80%
Test Data	20%
# Input Features	91
Layer 1 Nodes	10
Layer 2 Nodes	5
Epochs	50
Testing Loss	0.4719
Testing Accuracy	80%
Precision No-Show	0.5
Recall No-Show	0.02
f1-score No-Show	0.04
Precision Show	0.8
Recall Show	1
f1-score Show	0.89
Accuracy	0.8

Neural Net

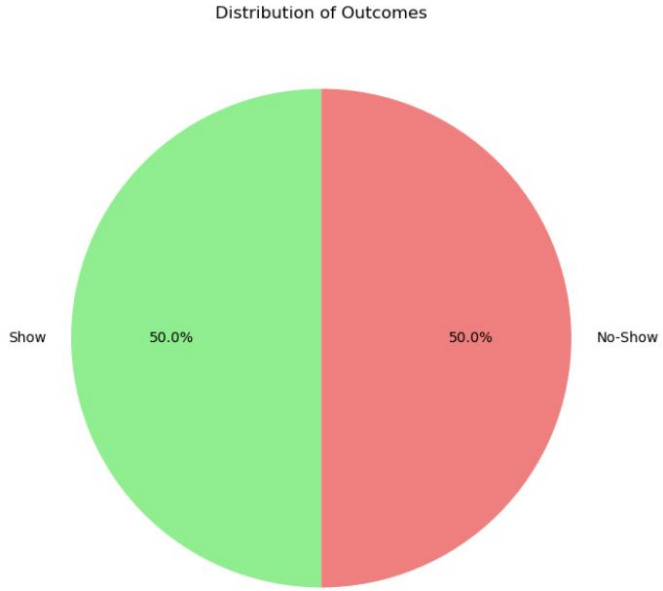
- Our target outcomes distribution was largely imbalanced
  - 80% Show
  - 20% No-Show
- To fix this we used an undersampling technique to balance the dataset
- This did however decrease our dataset from 110,515 rows to 44,632

```
# Undersample overrepresented class (Show) to evenly distribute the Outcome data
resample_df = pd.concat([X, y], axis=1)

show = resample_df[resample_df['Outcome'] == 1]
no_show = resample_df[resample_df['Outcome'] == 0]

undersampled_show_count = resample(show, replace=False, n_samples = len(no_show), random_state=50)

undersampled_df = pd.concat([undersampled_show_count, no_show])

X_undersampled = undersampled_df.drop('Outcome', axis=1)
y_undersampled = undersampled_df['Outcome']

X_train, X_test, y_train, y_test = train_test_split(X_undersampled, y_undersampled, test_size=0.2, random_state=50)
```

Neural Net



Before



After

Neural Net - 2nd Attempt

- Used all features from preprocessing
- One-hot encoded categorical features
- Balanced target outcome using undersampling technique
- 80% used for training, 20% for testing
- 2 layers and 50 epochs
- Much more even Precision and Recall scores between outcomes, but a decrease in Accuracy

	Model 1	Model 2
Training Data	80%	80%
Test Data	20%	20%
# Input Features	91	91
Layer 1 Nodes	10	10
Layer 2 Nodes	5	5
Epochs	50	50
Testing Loss	0.4719	0.6108
Testing Accuracy	80%	67%
Precision No-Show	0.5	0.63
Recall No-Show	0.02	0.8
f1-score No-Show	0.04	0.71
Precision Show	0.8	0.73
Recall Show	1	0.54
f1-score Show	0.89	0.62
Accuracy	0.8	0.67

Neural Net - 3rd Attempt

- Explored additional preprocessing steps:
  - Added columns for day of week for the appointment, and when it was scheduled
  - Mapped Gender and Neighborhood columns to numeric values
  - Added column to generate a score of comorbidities present
- Increased training data to 85%
- Added 3rd hidden layer
- Increased epochs to 100
- Saw a small increase in accuracy

	Neural Net		
	Model 1	Model 2	Model 3
Training Data	80%	80%	85%
Test Data	20%	20%	15%
# Input Features	91	91	13
Layer 1 Nodes	10	10	10
Layer 2 Nodes	5	5	5
Layer 3 Nodes			2
Epochs	50	50	100
nn.eval Loss	0.4719	0.6108	0.5945
nn.eval Accuracy	80%	67%	68%
Precision No-Show	0.5	0.63	0.64
Recall No-Show	0.02	0.8	0.83
f1-score No-Show	0.04	0.71	0.72
Precision Show	0.8	0.73	0.75
Recall Show	1	0.54	0.53
f1-score Show	0.89	0.62	0.62
Accuracy	0.8	0.67	0.68

Random Forest and Logistic Regression

- Used the same preprocessed data as final neural net model
- Similar accuracy, precision, and recall as neural net model

	Random Forest
	Model 1
Training Data	80%
Test Data	20%
Precision No-Show	0.64
Recall No-Show	0.72
f1-score No-Show	0.67
Precision Show	0.67
Recall Show	0.59
f1-score Show	0.63
Accuracy	0.65
	Confusion Matrix:
	[[3209 1265]
	[1835 2618]]

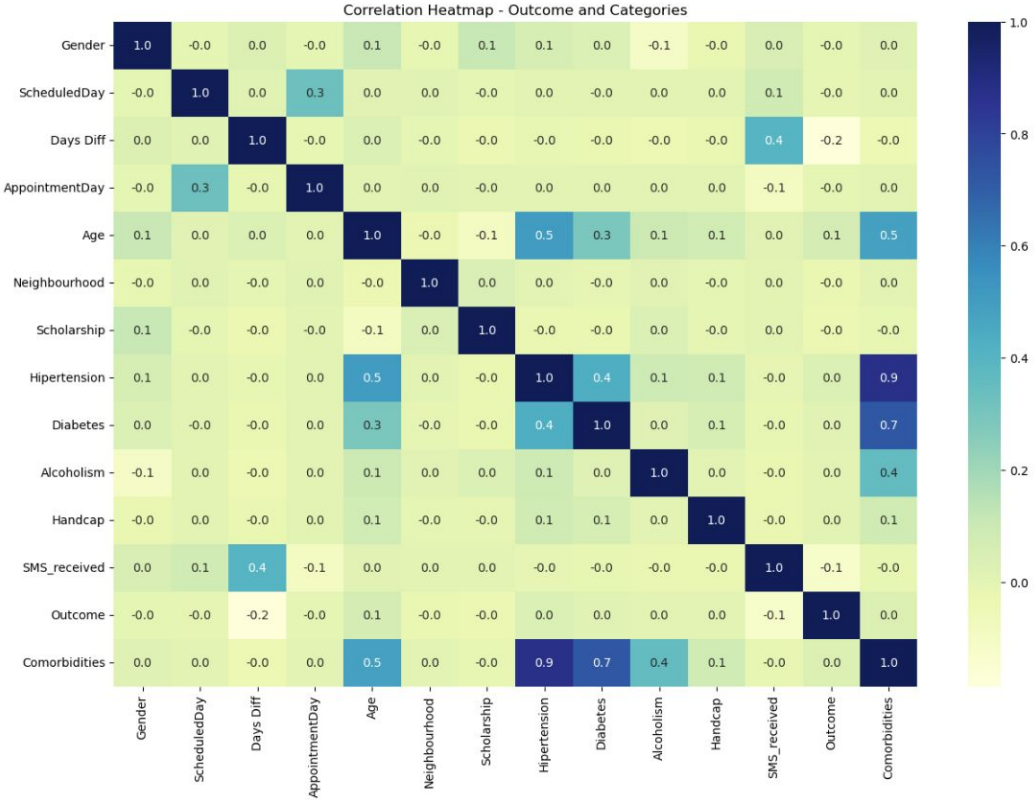
	Logistic Regression
	Model 1
Training Data	80%
Test Data	20%
Precision No-Show	0.65
Recall No-Show	0.56
f1-score No-Show	0.6
Precision Show	0.61
Recall Show	0.7
f1-score Show	0.65
Accuracy	0.63
	Confusion Matrix:
	[[2491 1983]
	[1354 3099]]

Model Comparison

- Model with best overall performance was the neural net with the additional engineered features

	Neural Net			Random Forest	Logistic Regression
	Model 1	Model 2	Model 3	Model 1	Model 1
Training Data	80%	80%	85%	80%	80%
Test Data	20%	20%	15%	20%	20%
# Input Features	91	91	13		
Layer 1 Nodes	10	10	10		
Layer 2 Nodes	5	5	5		
Layer 3 Nodes			2		
Epochs	50	50	100		
nn.eval Loss	0.4719	0.6108	0.5945		
nn.eval Accuracy	80%	67%	68%		
Precision No-Show	0.5	0.63	0.64	0.64	0.65
Recall No-Show	0.02	0.8	0.83	0.72	0.56
f1-score No-Show	0.04	0.71	0.72	0.67	0.6
Precision Show	0.8	0.73	0.75	0.67	0.61
Recall Show	1	0.54	0.53	0.59	0.7
f1-score Show	0.89	0.62	0.62	0.63	0.65
Accuracy	0.8	0.67	0.68	0.65	0.63
				Confusion Matrix:	Confusion Matrix:
				[[3209 1265]	[[2491 1983]
				[1835 2618]]	[1354 3099]]

Outcome & Different Categories



Reasons for No-Show

Patients commonly give several reasons for missing appointments that include forgetfulness, confusion, or miscommunication over appointment information, feeling better, transportation issues, and difficulty leaving work or school.

Conclusion

If our model could be trained, optimized, and improved, we would be able to predict, base on the patient's characteristics, who is more likely to miss their appointment and take appropriate measures to reduce the risk of no-shows. This could help to decrease monetary losses and improve people's health.