Medical Appointment No Shows - Kaggle

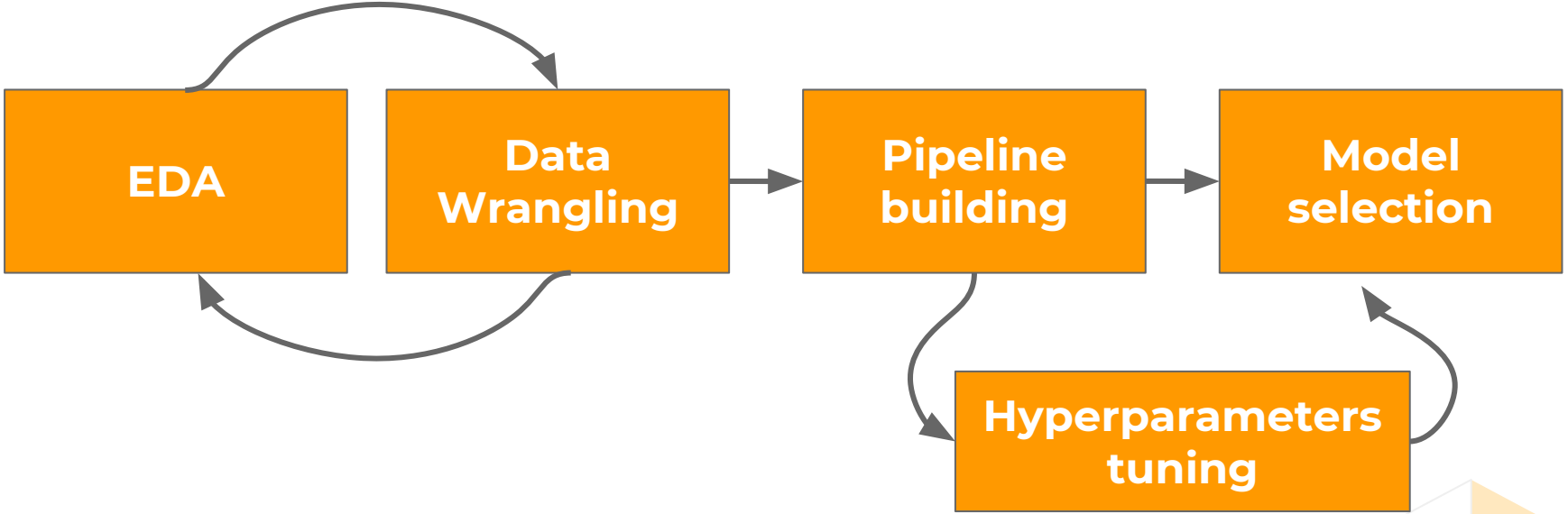
Leonardo Ignacio Córdoba

Problem statement

- There is a region in Brazil where patients don't show to about **20%** of the medical appointments they made.

- The tasks are:
  - Perform a **Exploratory Data Analysis**
  - **Predict** whether a patient will show or not to a given appointment.

Methodology

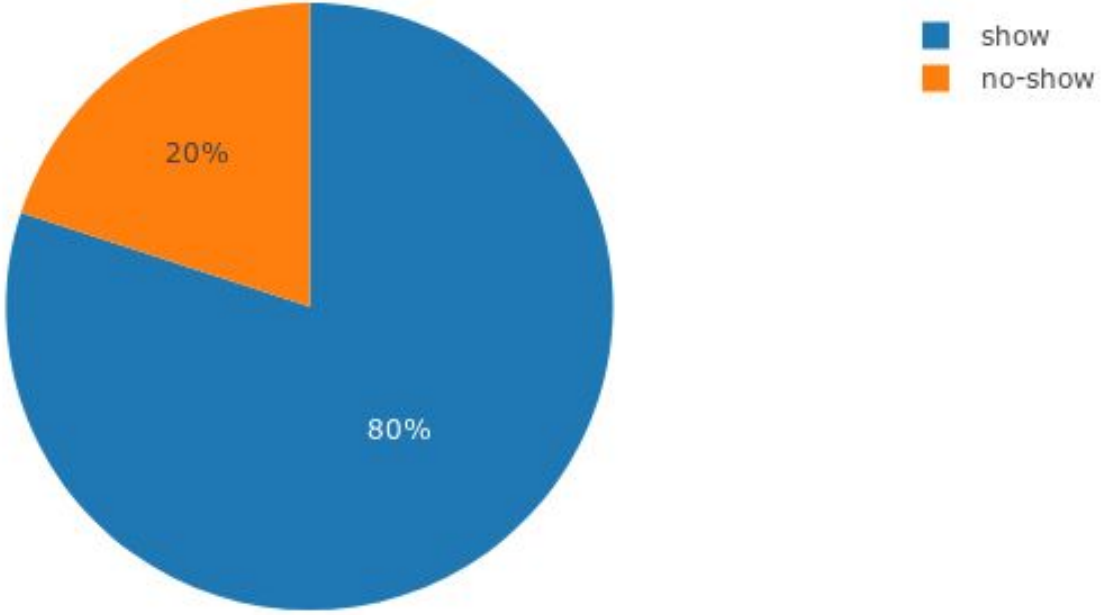


Exploratory Data Analysis

Exploratory Data Analysis

- 14 variables
- 110.527 rows
- No missing values
- Time series data: ScheduledDay and AppointmentDay
- Neighbourhood has high cardinality: 81 distinct values
- Gender and No-show needs encoding
- The positive class is 20,2%
- Age has a minimum value of -1 and a maximum of 115. The most frequent value is 0.

Class balance



- Balance of classes is not a great problem to deal with

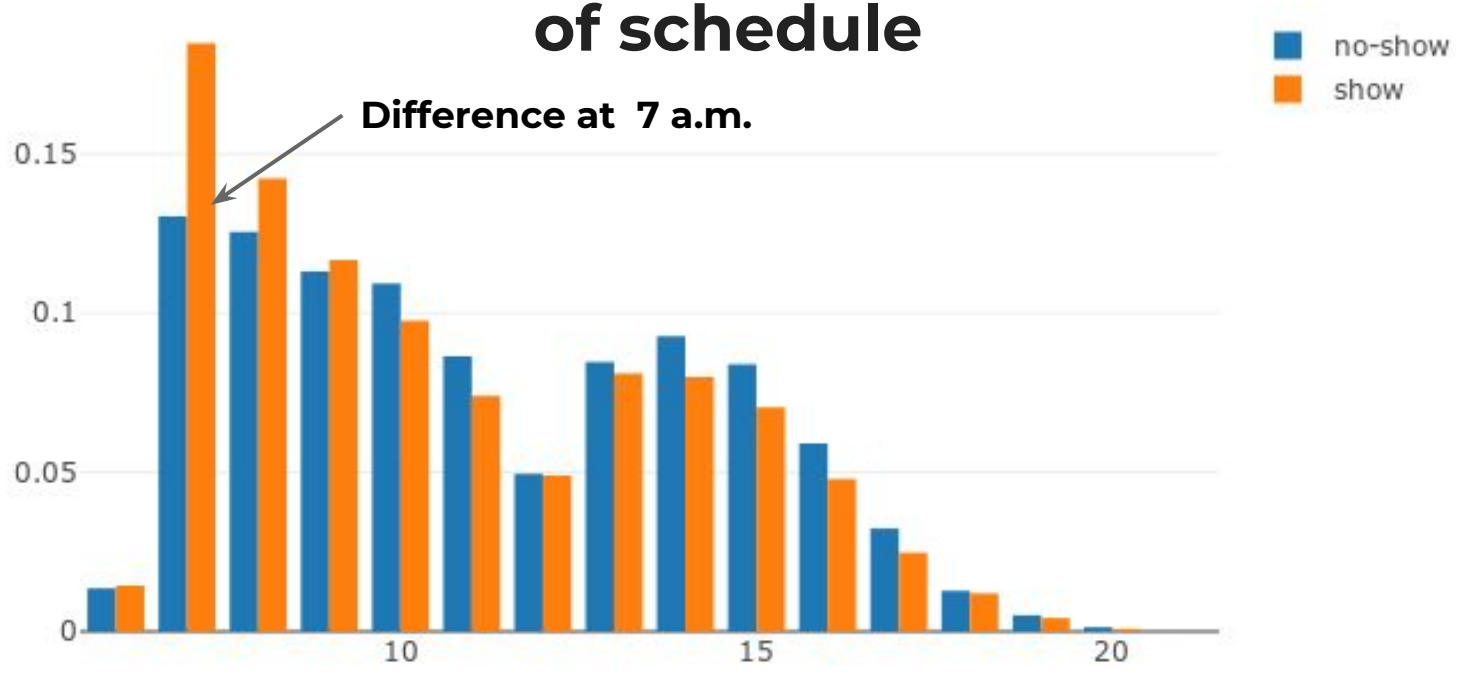
Manual feature creation

- **MinutesDiff**: time (in minutes) between the scheduling time of the appointment and the appointment
- **ScheduledSameDay**: flag that indicates if the patient scheduled a meeting to that same day
- **AppointmetDayofWeek**: day of the week of the appointment
- **ScheduledDayofWeek**: day of the week of the schedule
- **ScheduledHour**: hour in which the schedule was made
- **AppointmentMonth**: month of the appointment

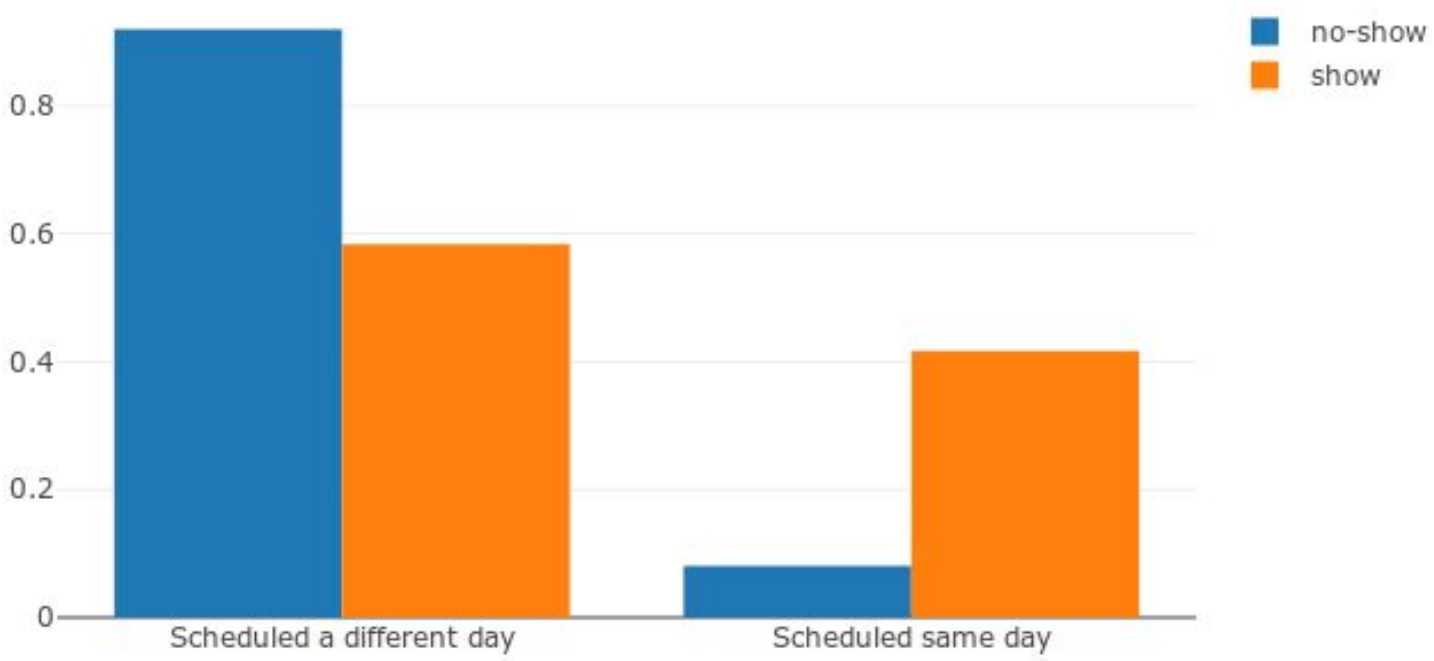
Manual feature creation

- **AppointmentsCount**: number of appointments of that patient before that day
- **NotShowedCount**: number of appointment of that patient that did not attend before that day
- **ProportionNotShowed**: proportion of previous appointments that the patient did not show
- **FirstAppoint**: flag that indicates if that is the first appointment or not

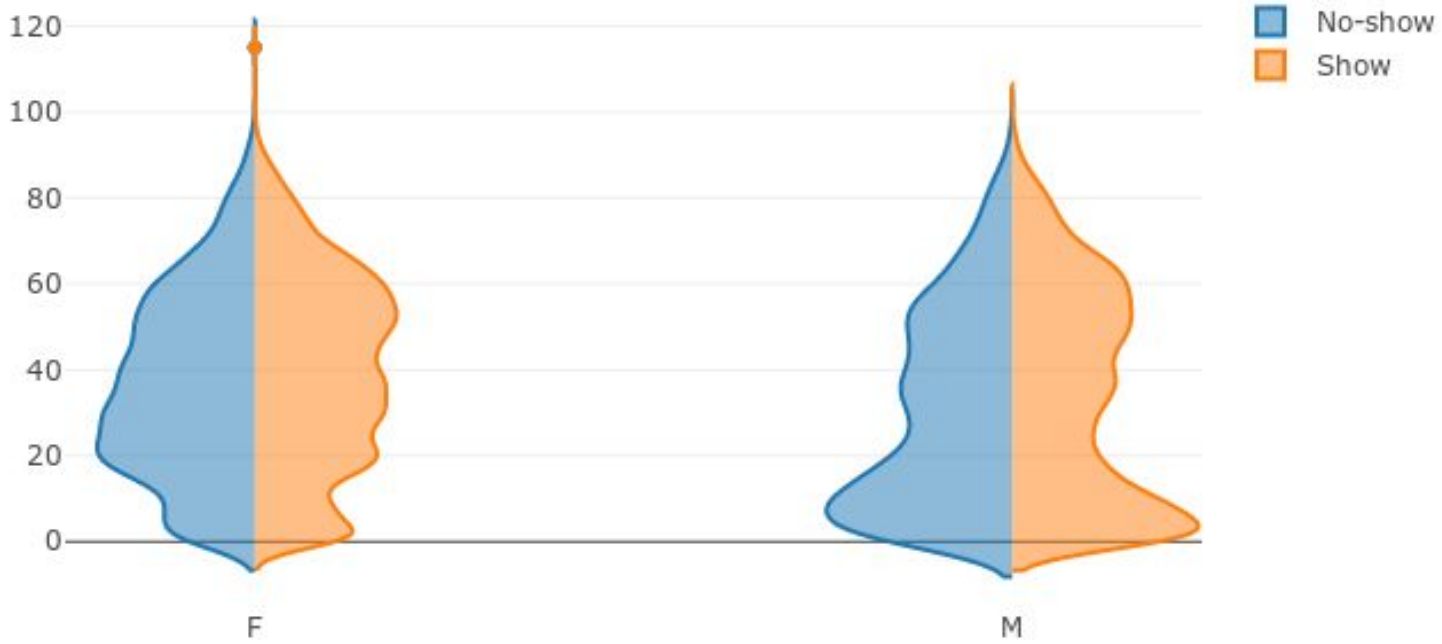
Probability of showing by hour of schedule



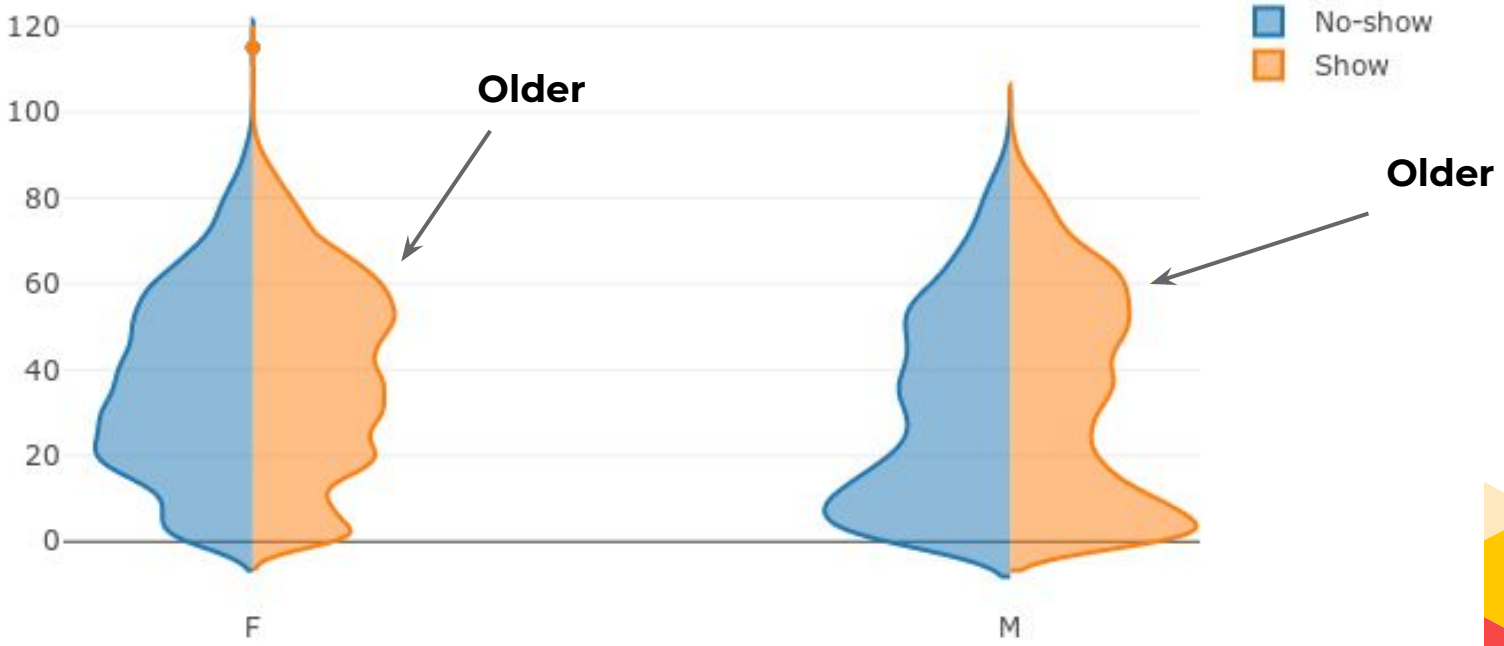
Probability of showing by day of schedule



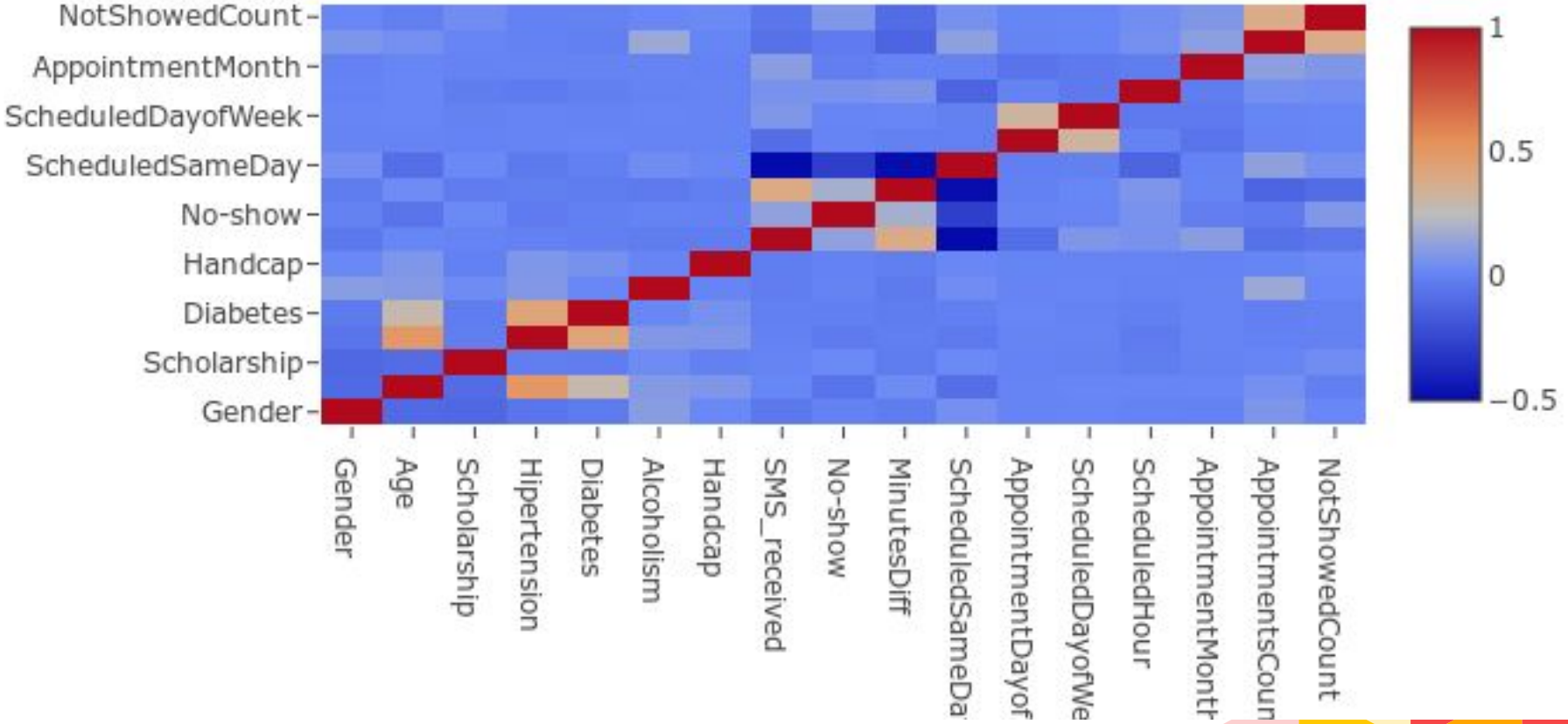
Probability of showing by age and gender



Probability of showing by age and gender

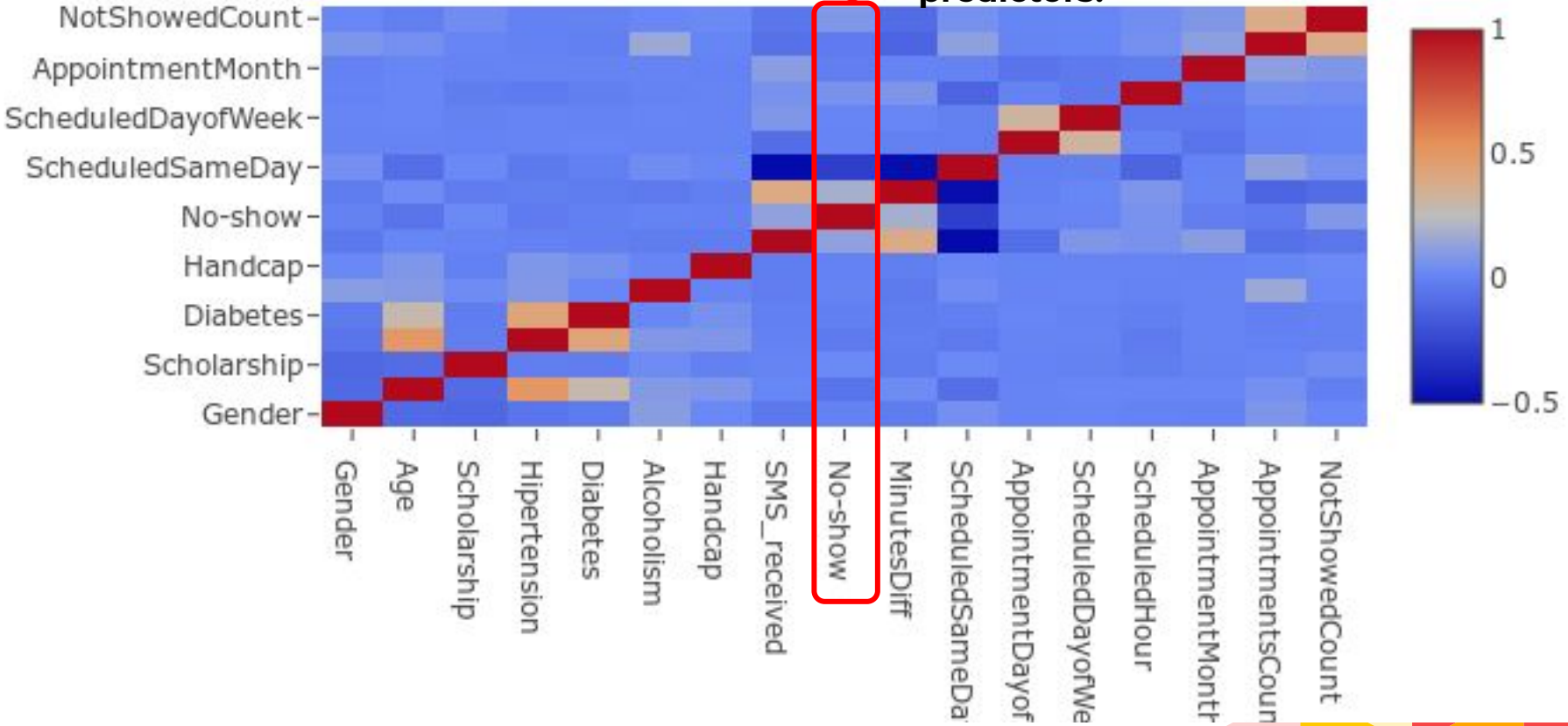


Correlation Plot



Correlation Plot

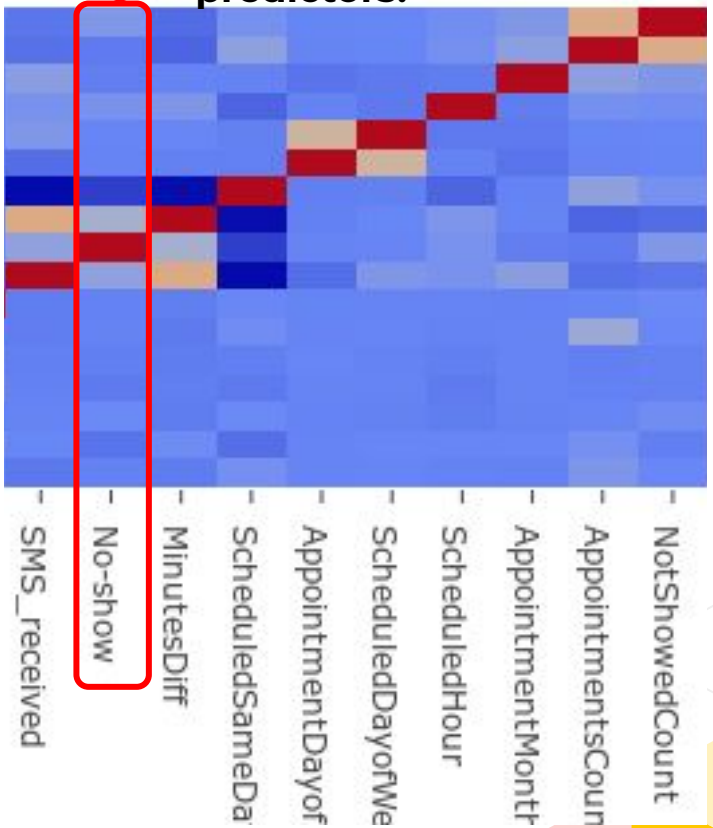
Problem: There are no good obvious predictors!



Correlation Plot

- **The most correlated variables are:**
- **ScheduledSameDay (-0,28)**
- **MinutesDiff (0,18)**
- SMSReceived (0,12)
- **NotShowedCount (0,08)**
- **ScheduledHour (0,06)**
- Gender (-0,06)

Problem: There are no good obvious predictors!



Automatic FE and model selection

Design of the pipeline

- A **pipeline** of transformations was made to get to a better model.
- **Step 1:**
  - Decide whether to use the top 10, top 20, top 30 (according to the odds of the target) or all the **neighborhoods**, and make dummies from them.
  - Make interactions of dummy variables ('Gender', 'Scholarship', 'Hipertension','Diabetes', 'Alcoholism', 'Handcap', 'SMS_received', 'ScheduledSameDay')
  - Make polynomial features from age and NotShowedCount
  - Merge all these variables with the remaining variables

Design of the pipeline

- **Step 2:**

Variables are **scaled** using the standard deviation and **centered** with the mean.

This improve performance of models that depend on distance measures and doesn't affect performance of tree based algorithms.

Design of the pipeline

- **Step 3:**

In this step one can choose whether to apply a **PCA** transformation.

This may help to find latent variable that summarize different information, improving the model.

- **Step 4:**

This is a **resampling** step. Using imblearn library one can choose to use a Random Subsampling strategy, an over sampling strategy (using SMOTE) or nothing.

Design of the pipeline

- **Step 5:**

In this step a **feature selection** procedure is used to keep the X percentile most influential variables, using a statistical test.

- **Step 6:**

This the final step, here you train the predictive **model**.

Models

The models trained are:

- KNN
- SVM
- Naive Bayes
- Logistic Regression
- Random Forest
- LightGBM
- CatBoost

For each model a random search over hyperparameters was made, both algorithm hyperparameters and pipeline hyperparameters

Results

Area under the ROC curve

- In binary classification problems this metric (also known as AUC) is the metric most commonly used to assess whether a model is better than another or not.
- **It can be interpreted as the probability of correctly ranking higher a positive case than a negative case.**
- It is a way of summarizing precision and recall

It is preferred over accuracy because:

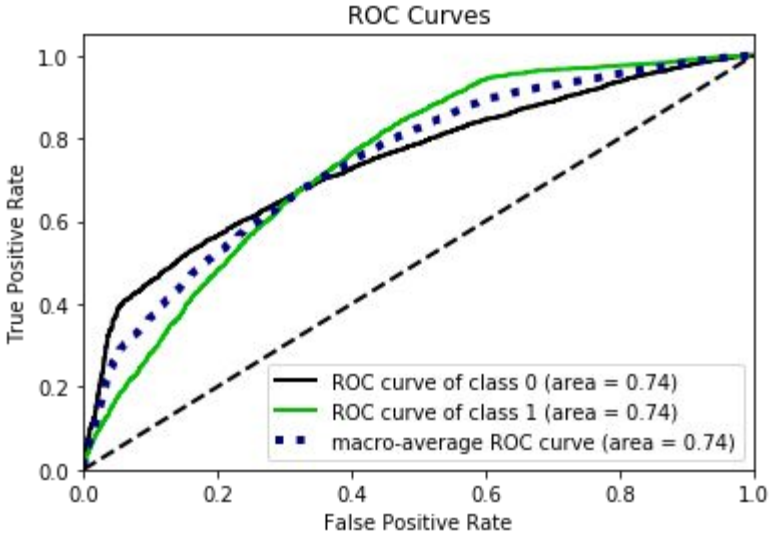
- Accuracy doesn't admit scores as prediction, it needs hard predictions which depends on what the optimal threshold is for each model.
- Accuracy doesn't take into account the proportion of the majority class
- Accuracy doesn't give you information on how are you performing in each class

Results

Model	Cross validation result	Test result
KNN	0,712	0,716
SVM	0,705	0,689
Naive Bayes	0,653	0,691
Logistic Regression	0,653	0,708
Random Forest	0.7487	0,7405
LightGBM	0,7512	0,7421
CatBoost	0,7503	**0,7433**

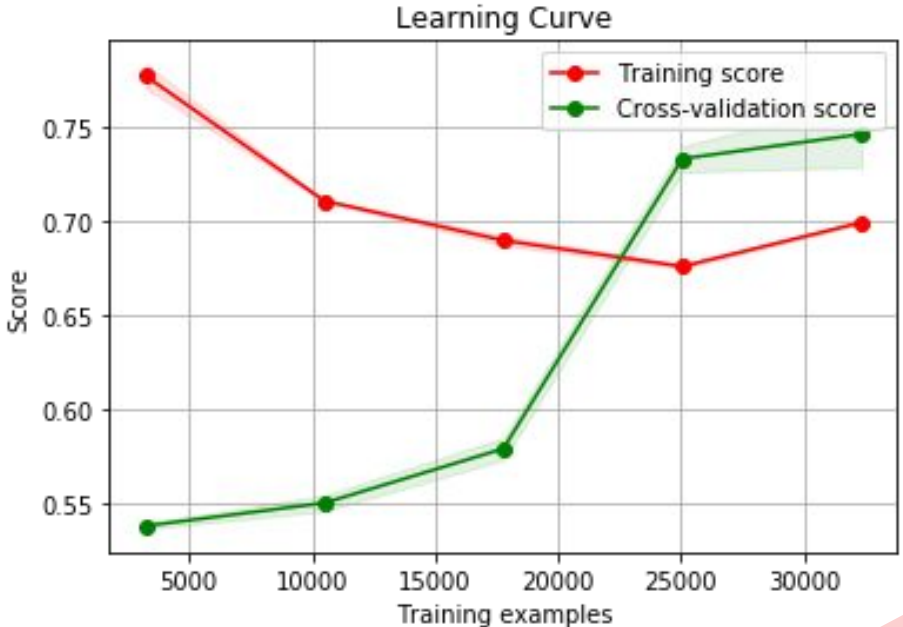
Best Model

- CatBoost was the best algorithm. Some of the best hyperparams were:
- No PCA or resampling method
- Use percentile 90 of best variables
- Use top 20 neighborhoods as dummies
- Max depth = 4 and 250 trees



Getting more data?

A typical way of improving models is getting more data to train.
When plotting the learning curve of CatBoost one realizes that effectively if we could get more data then the model's performance will probably improve



Improving the model

Some opportunities to improve the model's performance are:

- Getting more data
- More Features
- More hyperparameter searching, specially in ensemble models
- Stacking of models
- Getting a personalized objective function

Conclusions

- There are no obvious predictors... Complex feature engineering was needed to improve performance.

- The variables with more information were ScheduledSameDay MinutesDiff, SMSReceived.

- Ensemble models outperformed other models. In particular, CatBoost achieved the best performance.

- Getting more data and stacking might be ways of improving models performance

Thanks!

Questions?