Machine Learning Engineer Nanodegree

Capstone Project: Medical Appointment No Show Prediction

Chun Zhu

March 4, 2018

I. Definition

Project Overview

Patients who do not show up for their scheduled appointments without cancelling nor advance notice is a severe issue that significantly affects clinic efficiency, patient care and healthcare costs. When no-show happens, the medical resource is underutilized. The doctor's valuable time is lost, and the other patients miss their timely treatment because the schedule is taken by the no-show patients. According to some paper, an average of 42% of appointments become no-shows (http://www.annfammed.org/content/2/6/541.full) and the waste healthcare costs can reach hundreds of thousands of dollars yearly in US. Hence, accurate prediction of no-show appointment is very important for both the medical resource and the patients. It is also a cornerstone for future no-show reduction strategy.

Some research papers related to this topic:

http://www.ijimai.org/journal/node/1623
dmkd.cs.vt.edu/papers/THSE15.pdf

Problem Statement

The problem is to predict whether a patient will show up in a medical appointment in public hospitals in Brazil based on some attributes of the patient and the appointment itself. This is a binary classification problem.

The dataset contains 110527 records/appointments. Patients showed up in 88208 appointments - 79.8% of total appointments, while no show in 22319 appointments - 20.2% of total appointments. The classes are not balanced.

Random forest will be applied on the dataset as the classification algorithm. Cross-validation will be applied to evaluate best parameters. The training dataset includes both the features (input) and the output variable. From it, the supervised learning algorithms seek to learn the mapping function from the input to the output. The goal is to build a good model that can make predictions for the new dataset (test dataset).

Metrics

As the classes are imbalanced, F1-score will be the evaluation metric to quantify both the benchmark model and the solution model.

$TP = true\ positive$

$TN = true\ negative$

$FP = false\ positive$

$FN = false\ negative$

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F_1 = \frac{2 \times precision \times recall}{precision + recall}$$

F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

II. Analysis

Data Exploration & Visualization

The dataset is from Kaggle
https://www.kaggle.com/joniarroba/noshowappointments/data

The fields of data are below:

- `PatientId`: Identification of a patient

- `AppointmentID`: Identification of each appointment

- `Gender`: Male or Female. In common sense, women takes more care of their health in comparison to man.

- `ScheduledDay`: The day someone called or registered the appointment, this is before appointment.

- `AppointmentDay`: The day of the actual appointment, when they have to visit the doctor. The appointment day may affect the no-show result. Monday or Friday may not the same for the patient.

- `Age`: How old the patient is.

- `Neighborhood`: the neighborhood of the hospital in which the appointment is carried out. It is possible for patients to come from outside of the neighborhood, or even the city.

- `Scholarship`: whether the patient is covered by Bolsa Família, a social welfare program in Brazil. Ture of False.

- `Hipertension`: True(1) or False(0)

- `Diabetes`: True(1) or False(0)

- `Alcoholism`: True(1) or False(0)

- `Handicap`: an integer ranging from 0 to 4, indicating the level of the handicap the patient is suffering from.

- `SMS_received`: whether an SMS messages was sent to the patient to remind him/her of the appointment.

- `No-show`: the target variable, Yes or No. will be transformed to 1 or 0

Here are the first 5 rows of the data

PatientId	AppointmentID	Gender	ScheduledDay	AppointmentDay	Age	Neighbourhood	Scholarshi	Hipertensi	Diabetes	Alcoholism	Handcap	SMS_received	No-show
29872499824296	5642903	F	2016-04-29T18:38:08Z	2016-04-29T00:00:00Z	62	JARDIM DA PENHA	0	1	0	0	0	0	No
558997776694438	5642503	M	2016-04-29T16:08:27Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA	0	0	0	0	0	0	No
4262962299951	5642549	F	2016-04-29T16:19:04Z	2016-04-29T00:00:00Z	62	MATA DA PRAIA	0	0	0	0	0	0	No
867951213174	5642828	F	2016-04-29T17:29:31Z	2016-04-29T00:00:00Z	8	PONTAL DE CAMBURI	0	0	0	0	0	0	No
8841186448183	5642494	F	2016-04-29T16:07:23Z	2016-04-29T00:00:00Z	56	JARDIM DA PENHA	0	1	1	0	0	0	No

First of all, I found there are some typos in the column headers. So I made the following changes:

```
'PatientId':'PatientID'
'Neighbourhood':'Neighborhood'
'Hipertension':'Hypertension'
'Handcap':'Handicap'
'No-show':'NoShow'
```

Next, let's explore the number of patientsID and AppointmentID.

Number of records	110527
Number of PatientID	62299
Number of AppointmentID	110527

It is obvious that there were some patients who made several appointments during the time period. Although the history of the patients' appointments is important to the no-show prediction, I won't include these 2 features in the analysis in order to avoid the data leakage.

The original dataset includes ScheduledDay and AppointmentDay. We can notice that ScheduledDay feature contains information of hour, minute and second, but AppointmentDay only contains day information. The waiting time – the time between ScheduledDay and AppointmentDay - should be an interesting feature as the no-show rate should be much lower for the same day appointments. So the waiting time in day is added in the dataset as *WaitDay*.

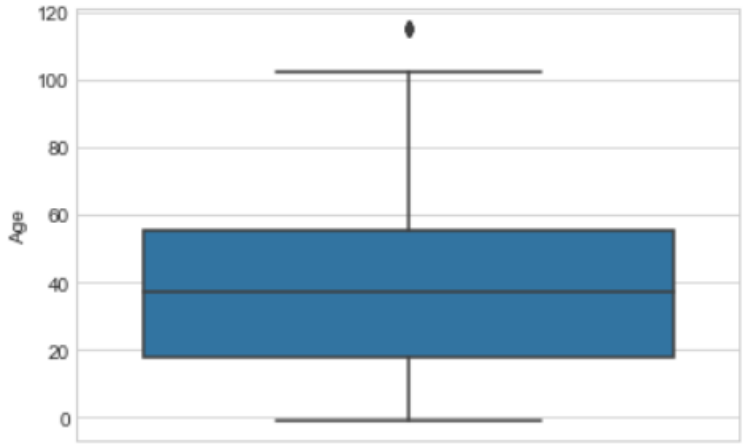
Another added feature is the day of the week for the AppointmentDay – *AppointmentDay_DOW*. In common sense, no-show rate should not be the same between weekday appointments and weekend appointments.

With waiting time and AppointmentDay_DOW, we can drop these 2 features: ScheduledDay and AppointmentDay.

Now let's explore all the remaining features. The unique values of these features are as follows:

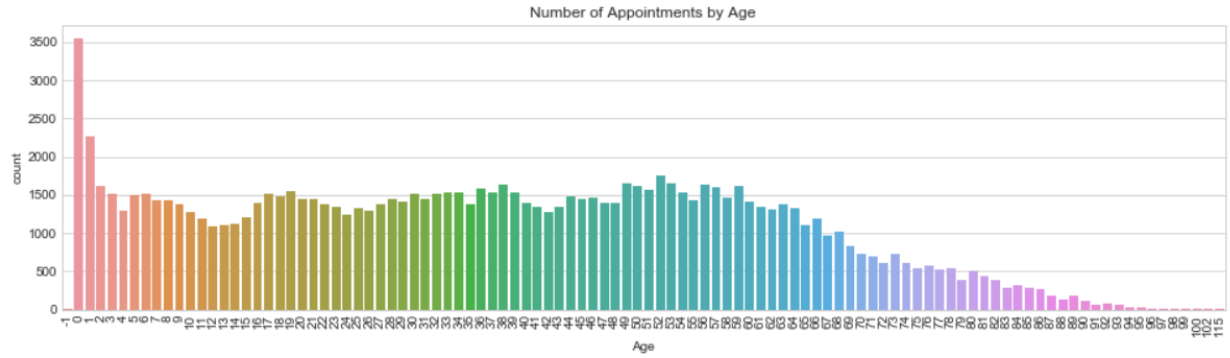
```
Gender: ['F' 'M']


Age:    [-1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 4
2, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,
 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 8
5, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 102, 115]


Number of neighborhoods: 81
Neighborhood:    ['AEROPORTO', 'ANDORINHAS', 'ANT\xc3\x94NIO HON\xc3\x93RIO', 'ARIOVAL
DO FAVALESSA', 'BARRO VERMELHO', 'BELA VISTA', 'BENTO FERREIRA', 'BOA VISTA', 'BONFIM
', 'CARATO\xc3\x8dRA', 'CENTRO', 'COMDUSA', 'CONQUISTA', 'CONSOLA\xc3\x87\xc3\x83O', '
CRUZAMENTO', 'DA PENHA', 'DE LOURDES', 'DO CABRAL', 'DO MOSCOSO', 'DO QUADRO', 'ENSEAD
A DO SU\xc3\x81', 'ESTRELINHA', 'FONTE GRANDE', 'FORTE S\xc3\x83O JO\xc3\x83O', 'FRADI
NHOS', 'GOIABEIRAS', 'GRANDE VIT\xc3\x93RIA', 'GURIGICA', 'HORTO', 'ILHA DAS CAIEIRAS
', 'ILHA DE SANTA MARIA', 'ILHA DO BOI', 'ILHA DO FRADE', 'ILHA DO PR\xc3\x8dNCIPE', '
ILHAS OCE\xc3\x82NICAS DE TRINDADE', 'INHANGUET\xc3\x81', 'ITARAR\xc3\x89', 'JABOUR',
'JARDIM CAMBURI', 'JARDIM DA PENHA', 'JESUS DE NAZARETH', 'JOANA D\xc2\xb4ARC', 'JUCUT
UQUARA', 'MARIA ORTIZ', 'MARU\xc3\x8dPE', 'MATA DA PRAIA', 'MONTE BELO', 'MORADA DE CA
MBURI', 'M\xc3\x81RIO CYPRESTE', 'NAZARETH', 'NOVA PALESTINA', 'PARQUE INDUSTRIAL', 'P
ARQUE MOSCOSO', 'PIEDADE', 'PONTAL DE CAMBURI', 'PRAIA DO CANTO', 'PRAIA DO SU\xc3\x81
', 'REDEN\xc3\x87\xc3\x83O', 'REP\xc3\x9aBLICA', 'RESIST\xc3\x8aNCIA', 'ROM\xc3\x83O',
 'SANTA CEC\xc3\x8dLIA', 'SANTA CLARA', 'SANTA HELENA', 'SANTA LU\xc3\x8dZA', 'SANTA L
\xc3\x9aCIA', 'SANTA MARTHA', 'SANTA TEREZA', 'SANTO ANDR\xc3\x89', 'SANTO ANT\xc3\x94
NIO', 'SANTOS DUMONT', 'SANTOS REIS', 'SEGURAN\xc3\x87A DO LAR', 'SOLON BORGES', 'S\xc
3\x83O BENEDITO', 'S\xc3\x83O CRIST\xc3\x93V\xc3\x83O', 'S\xc3\x83O JOS\xc3\x89', 'S\x
c3\x83O PEDRO', 'TABUAZEIRO', 'UNIVERSIT\xc3\x81RIO', 'VILA RUBIM']


WaitDay: [-7.0, -2.0, -1.0, 0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.
0, 12.0, 13.0, 14.0, 15.0, 16.0, 17.0, 18.0, 19.0, 20.0, 21.0, 22.0, 23.0, 24.0, 25.0,
 26.0, 27.0, 28.0, 29.0, 30.0, 31.0, 32.0, 33.0, 34.0, 35.0, 36.0, 37.0, 38.0, 39.0, 4
0.0, 41.0, 42.0, 43.0, 44.0, 45.0, 46.0, 47.0, 48.0, 49.0, 50.0, 51.0, 52.0, 53.0, 54.
0, 55.0, 56.0, 57.0, 58.0, 59.0, 60.0, 61.0, 62.0, 63.0, 64.0, 65.0, 66.0, 67.0, 68.0,
 69.0, 70.0, 71.0, 72.0, 73.0, 74.0, 75.0, 76.0, 77.0, 78.0, 79.0, 80.0, 81.0, 82.0, 8
3.0, 84.0, 85.0, 86.0, 87.0, 88.0, 89.0, 90.0, 91.0, 92.0, 93.0, 94.0, 95.0, 96.0, 97.
0, 100.0, 101.0, 102.0, 103.0, 104.0, 106.0, 107.0, 108.0, 109.0, 110.0, 111.0, 114.0,
 116.0, 118.0, 121.0, 122.0, 124.0, 125.0, 126.0, 131.0, 132.0, 138.0, 141.0, 145.0, 1
50.0, 154.0, 161.0, 168.0, 175.0, 178.0]


AppointmentDay_DOW:      ['Friday', 'Monday', 'Saturday', 'Thursday', 'Tuesday', 'Wedn
esday']


Scholarship:     [0 1]
Hypertension:    [1 0]
Diabetes:        [0 1]
Alchoholism:     [0 1]
Handicap:        [0 1 2 3 4]
SMS_received:    [0 1]
NoShow: ['No' 'Yes']
```

Except for `Age` and `WaitDay`, all the other variables will be treated as categorical variables. Now let's check the distribution of Age.



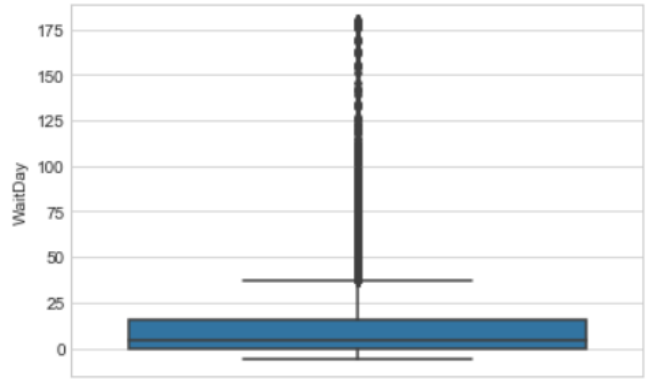
From the above boxplot we can see that the median age is around 35 and the IQR is between 18 and 55. The boxplot shows few datapoints as outliers which are ages above 100. But these data points are meaningful, I will not treat them as outliers.

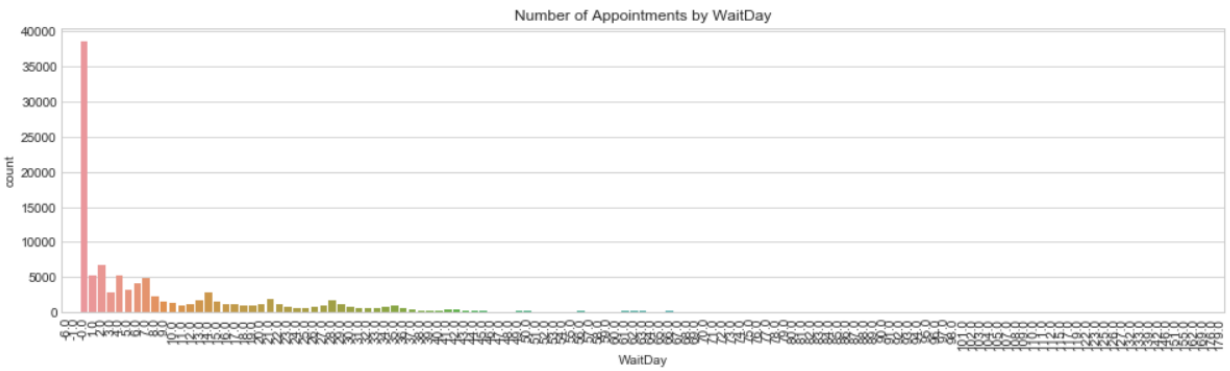
For more details, we can plot the number of appointments by age.



It is also noted that there are some patients with age = -1. These data points should also be removed.

The distribution of `WaitDay`:





The 0 `WaitDay` means the same day appointment. The `WaitDay` should not be negative. I assume these are typos which should be removed.

Now let's check the data imbalance. After removing the abnormalities, the overall no-show rate is 20.19%.

Algorithms and Techniques

The classifier I use is random forest, which is a popular ensemble learning method for classification, regression and other tasks. Random forest is an advanced type of decision tree, but it outperforms decision tree by avoiding the overfitting. Random forest averages multiple deep decision trees and applies feature bagging algorithm. In this way, random forest greatly improve the performance of the model.

The following parameters can be tuned to optimize the classifier:

n_estimators: The number of trees in the forest.

criterion: The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

max_depth: The maximum depth of the tree.

min_samples_split: The minimum number of samples required to split an internal node

min_samples_leaf: The minimum number of samples required to be at a leaf node.

class_weight: Weights associated with classes. If not given, all classes are supposed to have weight one.

Benchmark

The benchmark model is a simple out-of-the-box version of random forests trained on the same training data as the final solution. The parameters in the classifier are all default. The details are in the IMPLEMENTATION section.

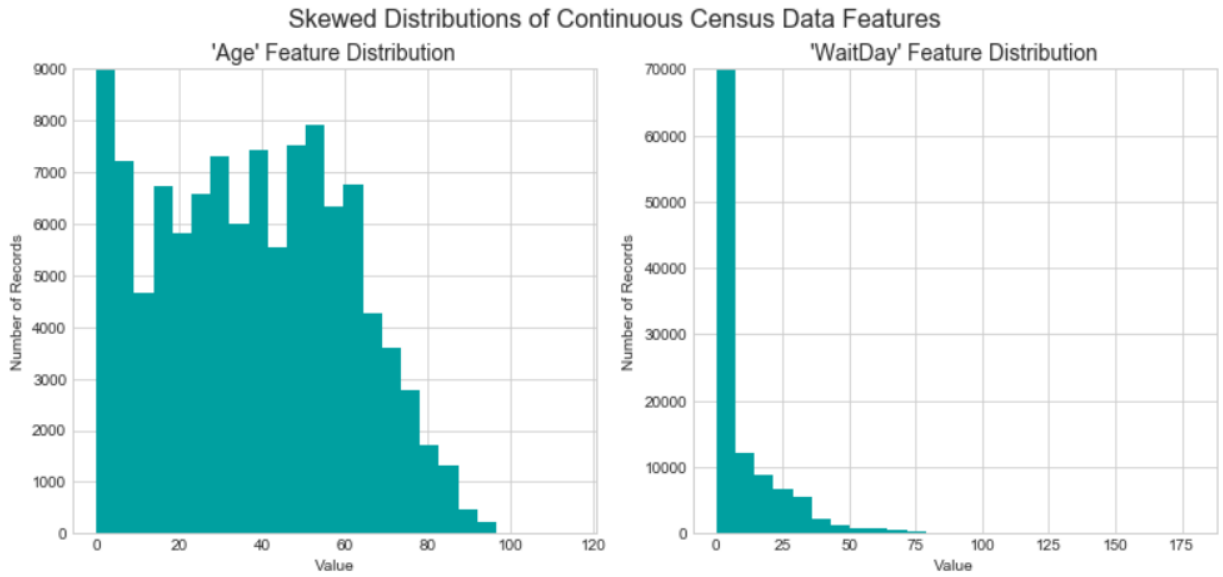
The F1 score of the benchmark model is 0.2877.

III. Methodology

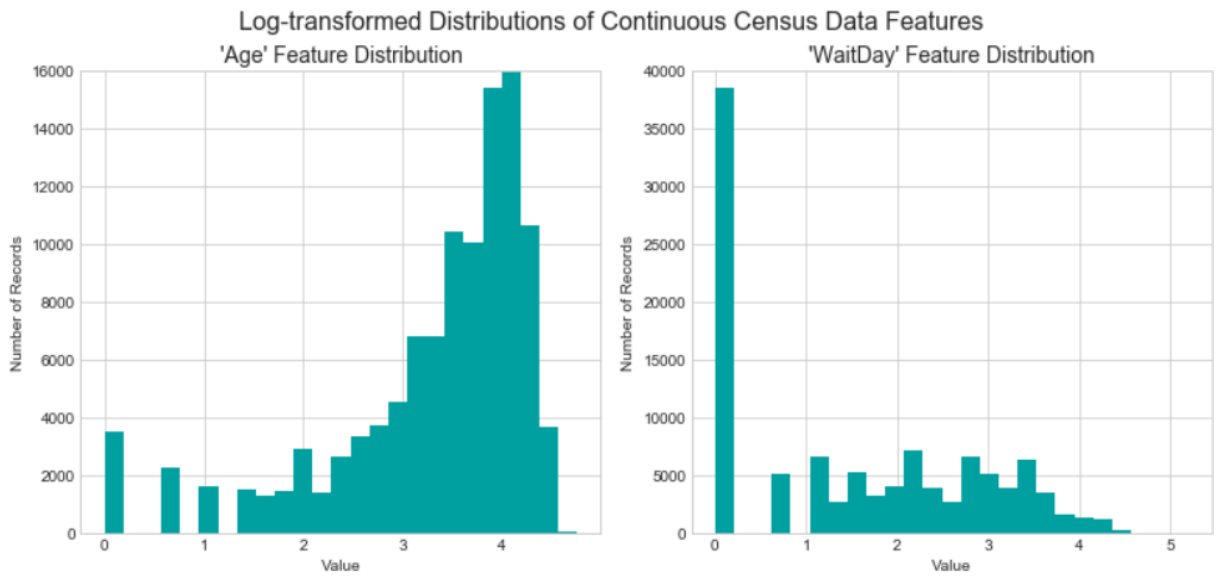
Data Preprocessing

First of all, we need to split the data into features and target labels. The target variable is '`NoShow`', and the rest are features variables.

Transforming Skewed Continuous Features



The 2 features – '`Age`' and '`WaitDay`' are skewed, especially '`WaitDay`'. It is common practice to apply a logarithmic transformation on the data so that the very large and very small values do not negatively affect the performance of a learning algorithm. Using a logarithmic transformation significantly reduces the range of values caused by outliers. However, the logarithm of 0 is undefined, so we must translate the values by a small amount above 0 to apply the logarithm successfully.



Now the distributions look more like normal distributions.

Normalizing Numerical Features

By applying `sklearn.preprocessing.MinMaxScaler` , we can normalize features '`Age`' and '`WaitDay`'.

This is one sample after transformation and normalization.

	Gender	Age	Neighborhood	Scholarship	Hypertension	Diabetes	Alcoholism	Handicap	SMS_received	WaitDay	AppointmentDay_DOW
5	F	0.76	REPÚBLICA	0	1	0	0	0	0	0.005618	Friday

Categorical variables

From data exploration section, we can see that features like '`Gender`', '`Neighborhood`' and '`AppointmentDay_DOW`' are non-numeric. We need to convert these categorical variables to numeric input using the one-hot encoding scheme. As there are only two possible categories for '`Gender`' ("`M`" and "`F`"), we can avoid using one-hot encoding and simply encode these two categories as 1 and 0, respectively.

Additionally, as with the non-numeric features, we need to convert the non-numeric target label, `'NoShow'` to numerical values for the learning algorithm to work. Since there are only two possible categories for this label ("`Yes`" and "`No`"), we can avoid using one-hot encoding and simply encode these two categories as `1` and `0`, respectively.

After one-hot encoding, there are 100 total features.

Implementation

Shuffle and split data

Now all categorical variables have been converted into numerical features, and all numerical features have been normalized. We will now split the data (both features and their labels) into training and test sets. 80% of the data will be used for training and 20% for testing.

Benchmark model performance

The benchmark model is a simple out-of-the-box version of random forest trained on the same training data as the final solution. The parameters in the classifier are all default.

The code is below:

```
rf_clf = RandomForestClassifier(random_state=0)
rf_clf.fit(X_train, y_train)
predictions_test = rf_clf.predict(X_test)
```

After the benchmark model is trained, the F1 score on the test set is 0.28.

Refinement

In this part, I will use *grid search technique* and *k-fold cross-validation training technique* to optimize the benchmark model.

GridSearchCV is a way to evaluate a model for each combination of parameters specified in the grid by using cross validation. Each combination of parameters will result in a score. The highest score implies the optimal parameters for the model.

The following parameters will be tuned to optimize the classifier:

n_estimators : The number of trees in the forest. Default = 10.

criterion : The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain. Default = “gini”

max_depth : The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples. Default = None

min_samples_split : The minimum number of samples required to split an internal node. default=2.

min_samples_leaf : The minimum number of samples required to be at a leaf node. default=1.

class_weight : Weights associated with classes. If not given, all classes are supposed to have weight one. default=None.

k-fold cross-validation is to randomly partition the data set into k samples equally. One sample is used for validation and the other k-1 folds are used to train the model. This procedure is repeated k times. Each time, a different sample is used as a validation test and a score is obtained. The final estimate is computed by averaging the k scores. The benefit of k-fold cross validation is that all the data points are used for both training and validation. Also, it can reduce the chance of overfitting a model.

My grid search parameters are as follows:

```
n_params = { 'n_estimators':[3,5,10,50],
             'criterion':['gini','entropy'],
             'max_depth': [3,4,5],
              'min_samples_split':[2,3,4,5],
              'min_samples_leaf':[1,2],
              'class_weight':['balanced',None]}
```

k-fold cross-validation is applied in the code as:

```
gsrf = GridSearchCV(rf_clf, n_params, cv= KFold(n_splits=5,shuffle=True),
scoring=scorer)
```

The best F1 score on test dataset we got is 0.4355, and the best classifier is:

```
RandomForestClassifier(bootstrap=True, class_weight='balanced',
            criterion='gini', max_depth=4, max_features='auto',
            max_leaf_nodes=None, min_impurity_decrease=0.0,
            min_impurity_split=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=10, n_jobs=1, oob_score=False, random_state=0,
            verbose=0, warm_start=False)
```

IV. Results

Model Evaluation and Validation

The final model is derived by grid-search technique and k-fold cross training techniques on the benchmark model. Each combination of parameters will result in a score. The highest score implies the optimal parameters for the model. The best F1 score is 0.4355, which is improved by 57% compared to the benchmark model.

Sensitivity

An optimal model is not necessarily a robust model. Sometimes, a model is either too complex or too simple to sufficiently generalize to new data. Sometimes, a model could use a learning algorithm that is not appropriate for the structure of the data given. Other times, the data itself could be too noisy or contain too few samples to allow a model to adequately capture the target variable — i.e., the model is underfitted. To test the robustness of the model, I applied the same training techniques 5 times with different training and testing sets to see how the F1 score on the testing sets changes with the data it's trained on.

```
F1 score on test data for different trials:

Trial 1: 0.4353
Trial 2: 0.4504
Trial 3: 0.4381
Trial 4: 0.4375
Trial 5: 0.4405

Range in F1 score: 0.0150
```

From the above result, we can conclude that the model is robust enough as the F1 scores are almost the same for different training samples.

Justification

The final solution is an optimized random forest classifier. I applied *grid search technique* and *k-fold cross-validation training technique* to optimize the benchmark model.

GridSearchCV is a way to evaluate a model for each combination of parameters specified in the grid by using cross validation. Each combination of parameters will result in a score. The highest score implies the optimal parameters for the model. In this way, the best parameters were selected to optimize the model. And for this random forest classifier, the parameters *n_estimators*, *criterion*, *max_depth*, *min_samples_split*, *min_samples_leaf* and *class_weight* were tuned to optimize the classifier.

k-fold cross-validation is to randomly partition the data set into k samples equally. One sample is used for validation and the other k-1 folds are used to train the model. This procedure is repeated k times. Each time, a different sample is used as a validation test and a score is obtained. The final estimate is computed by averaging the k scores. The benefit of k-fold cross validation is that all the data points are used for both training and validation. Also, it can reduce the chance of overfitting a model.

The Benchmark model has a F1 score of 0.2877, while the final model has the F1 score of 0.4355, which is improved by 57% compared to the benchmark model. So clearly the final model is stronger than the benchmark model.

The F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0. So F1 score of 0.4355 is not perfect, though it already improves a lot compared to the benchmark model. I think the information contained in the current training data is not enough. More features are needed to get a better result.

V. Conclusion

Free-Form Visualization

As this problem is not intended for visualization like image classification, I am not sure which visualization I can provide. For me, the most important quality of the problem is the F1 score and I need to improve it as much as I can.

Reflection

The process used for this project can be summarized using the following steps:

1. Download the data from Kaggle.
2. Explore the dataset, remove any unreasonable records, like age < 0, `AppointmentDay` is prior to `ScheduledDay`.
3. Add derived relevant features: waiting time between `ScheduledDay` and `AppointmentDay`, Day of the week for `AppointmentDay`.
4. Transform the skewed continuous features like `Age` and `WaitDay` and then do the feature scaling.
5. Convert these categorical variables to numeric input using the one-hot encoding scheme.
6. A simple out-of-the-box version of random forest is used as the benchmark model.
7. Apply *grid search technique* and *k-fold cross-validation training technique* to optimize the benchmark model.
8. Predict on the unseen test dataset by the best classifier.
9. Calculate f1 score of the test results and compared with the benchmark model.

The idea of adding new features is challenging for me. At the beginning I didn't think of adding new features, but I found it difficult to use `ScheduledDay` and `AppointmentDay` in the model training. The idea of Day of the week for AppointmentDay came to me when I made an appointment myself. When I chose the appointment day, I just realized that weekday and weekend should not be the same. Another new feature I wanted to add is appointment no-show/show history. Actually I have added it from the start. But thinking over and over, I realized that it will result in some problems. After searching online, I learned that it is a data leakage problem (*https://www.kaggle.com/wiki/Leakage*).

The interesting aspect of this project is that it is the first time I analyze healthcare dataset. I learned a lot related to healthcare when I explored the dataset. Also I learned about "data leakage".

As stated in the JUSTIFICATION part, the F1 score of 0.4355 for the final model is not perfect, though it already improves a lot compared to the benchmark model. I think the information contained in the current training data is not enough. More features are needed to get a better result.

Improvement

I will consider following improvements in the future:

1. Add another feature – appointment no-show/show history. It is obvious that there were some patients who made several appointments during the time period. In this project, I didn't include this feature because of data leakage risk. As I shuffle and randomly split the data into training and testing sets, including this feature will definitely result in data leakage. But later I can try not to split the data randomly. Instead, I can split it by appointment time. I think adding this feature will definitely improve the model a lot.

2. I am interested in using XGBoost. XGBoost is an optimized distributed gradient boosting system designed to be highly efficient, flexible and portable. XGBoost models dominate many Kaggle

competitions and they require more knowledge and *model tuning* than techniques like Random Forest. I think XGBoost will be extremely helpful in the future machine learning projects.