Machine Learning Engineer Nanodegree

Capstone Project

Vitor R. Machado

September 16st, 2017

Definition

Project Overview

This project is a machine learning model capable of predicting the mortality of cancer patients in ICESP (a brazilian network of hospitals for cancer treatment). Cancer is a group of diseases involving abnormal cell growth, being one of the most complex group of diseases in the world. In Brazil, Sao Paulo state has a group of hospitals and treatment centers called ICESP known as the best option for cancer treatment in the country (read more in this link - in portuguese). The ICESP has several records of their patients. Their dataset is very detailed with a lot of information and can be downloaded through this link. Their dataset has 735237 records of patients from 2000 to 2017.

Problem Statement

The ICESP is a network of hospitals of a third world country, therefore it does not posses the most advanced equipments for cancer treatment. Moreover, the most advanced treatments for cancer are not available in this country yet. Furthermore, the ICESP does not have a method to predict if a given patient will survive or not given a combination of treatments. The goal of this project is to provide a machine learning model to predict the mortality of cancer patients. The strategy is to build, train and test a Random Forest Classifier which will use the ICESP dataset as input. Then the model will be able to predict if a future patient will survive or not the treatment. The dataset will be preprocessed using machine learning techniques (such as normalization, feature scaling and feature selection), then it will be splitted in 80% and 20% for training and testing respectively.

Metrics

The accuracy will be used as evaluation metric for the model. Accuracy is the number of correct predictions made divided by the total number of predictions made, multiplied by 100 to turn it into a percentage. Accuracy measures the overall performance of the model in predicting new values. This metric was chosen because it fits perfectly for the given problem; since that the problem is to predict whether a person survived or not, it makes sense to evaluate the accuracy of the model through the number of correct predictions made in the testing set.

Another metric will be used since accuracy alone is a weak evaluation metric (read more about the accuracy paradox). The metric F-beta score will be used along with accuracy. The F-beta score is the weighted harmonic mean of precision and recall. Combined, both metrics will make the model more reliable.

Analysis

Data Exploration

This section aims to understand more about the dataset for this project.

ESCOLARI	IDADE	SEXO	UFNASC	UFRESID	IBGE	CIDADE	CATEATEND	CLINICA	DIAGPREV	BASEDIAG	TOPO	TOPOGRUP	DESCTOPO	MORFO	OUTRACLA
9	54	1	BA	SP	3522109	ITANHAEM	9	15	1	3	C504	C50	MAMA, QUADRANTE SUPERIOR EXTERNO DA	85003	ECI
9	77	2	SP	SP	3513504	CUBATAO	2	15	1	3	C509	C50	MAMA, SOE (EXCLUI PELE DA MAMA C44.5)	85003	III B
9	59	2	SP	SP	3541000	NaN	9	15	2	3	C508	C50	MAMA, LESAO SOBREPOSTA DA	85003	NaN
4	31	2	SP	GO	5200050	ABADIA DE GOIAS	2	15	1	3	C539	C53	COLO DO UTERO	80703	NaN
9	67	2	GO	GO	5200050	ABADIA DE GOIAS	9	31	2	2	C693	C69	COROIDE	87203	NaN

To begin, the table below show some of the features in the dataset and the possible values for these features. To see all the features in the dataset, please check this link. In this sample it is possible to see that the dataset has both numerical and categorical features. Also, in some features there are **NaN** values, which may impact the model. Moreover, some features (such as **CIDADE**) have descriptive information of other features.

Numerical features such as IDADE, CONSDIAG and TRATCONS have a huge range between values. Taking IDADE (age) as an example, it is possible to see that the lowest age is 0 whereas the highest is 107.

```
Minimum  0
Maximum  107
Mean  59.6421029581
Median  62.0
Standard Deviation  16.9755032909
```

Numerical features require normalization because most of classifiers are sensitive to high range of values (for instance, some of them calculate the distance between two points by the Euclidean distance). Therefore, the range of all features should be normalized so that each feature contributes more proportionately.

Some categorical features have missing values.

```
Features with missing values:

PT            53866
PN            54382
PM            56385
OUTRACLA      97762
META04       104782
TRATHOSP          9
CICI            906
```

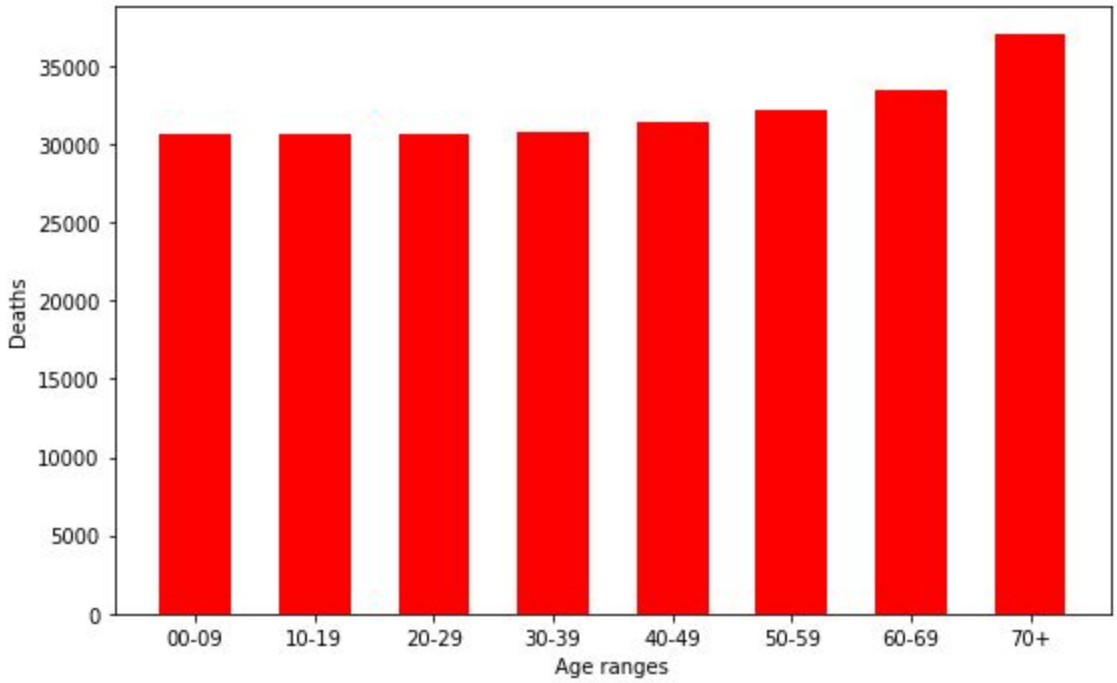
These missing values must be treated as well before sending the data to the models.

Moreover, features such as SEXO, DIAGPREV, TOPO are categorical features but they have numerical values. For instance, SEXO (sex of the patient) has the values 1 for man and 2 for woman. Since that these values are only to distinguish one from another, it is important to create dummy variables from this feature. The value 2 in this feature could have the double of the weight of the feature 1.

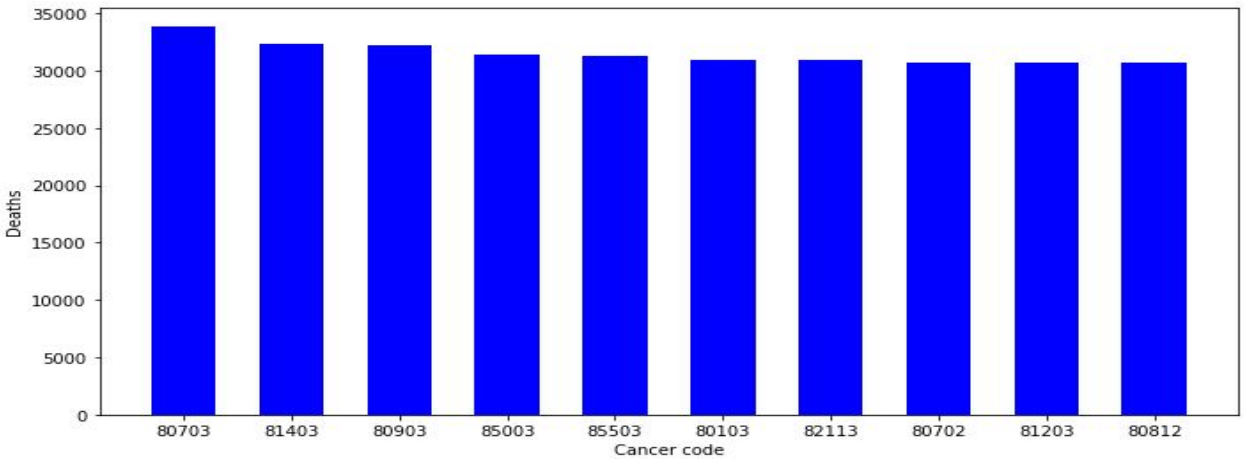
The output labels would be generated from the column ULTINFO. ULTINFO defines if a person survived (1 if survived without cancer, 2 otherwise) or not (3 if died from cancer and 4 if not) the treatment. This feature must be processed as well.

Exploratory Visualization

Exploratory visualization of the dataset was performed in order to understand more about the scenario, and to check which features could contribute more for the predictions. The first feature visualized was age. It was possible to see that the age is an important factor for the survival of cancer. The dataset showed that the number of deaths increases along with the age of the patients.



The same exploration were performed to investigate the most lethal types of cancer. 4 columns define the type (and the location) of the cancer: **MORFO**, **DESCMORFO**, **TOPO** and **DESCTOPO**. MORFO and DESCMORFO define the type of the cancer (code and description). For instance, the malignant melanoma has the code 87203. TOPO and DESCTOPO define the location of the cancer (code and description). For example, if the cancer is in the prostate the TOPO will be C619. The exploration showed the 10 more lethal types of cancer in ICESP.



```
Description of each cancer code:

80703 CARCINOMA ESCAMOCELULAR, SOE
81403 ADENOCARCINOMA, SOE
80903 CARCINOMA BASOCELULAR, SOE
85003 CARCINOMA DUCTAL INFILTRANTE, SOE
85503 CARCINOMA DE CELULAS ACINOSAS
80103 CARCINOMA, SOE
82113 ADENOCARCINOMA TUBULAR
80702 CARCINOMA ESCAMOCELULAR IN SITU, SOE
81203 CARCINOMA DE CELULAS TRANSICIONAIS, SOE
80812 DOENCA DE BOWEN
```

Algorithms and Techniques

The machine learning algorithm chosen was the Random Forest Classifier. Random Forest is an ensemble method in which N classification trees are constructed at training time. Then these trees outputs individuals labels, and the final output is the mode of the labels (classification) or the mean of the predictions (regression) of the individual trees. This algorithm was chosen because of its versatility and performance. Random Forest performs well in data with many features, and are considered a fast algorithm in training, testing and predicting. Also because this algorithm were chosen in a cancer prevention model and had a good performance in this task. The techniques used in the data were normalization (feature scaling - reduce the range of numerical values), feature selection (removal of the unnecessary features) and feature encoding (some of features where one hot encoded into dummy variables). The missing values were replaced by the mode value of the feature, and the output labels were also one hot encoded. The data were splitted into 80% and 20% for training and testing, and a cross-validation set were created using the training data to be used to search for the optimal values for the random forest. The parameters n_estimators, min_samples_split and max_depth were tuned through GridSearchCV.

Benchmark

The benchmarking process consisted of comparing the predicted classes of my random forest model in the testing set with the true labels of this test set in order to check the accuracy of the model and if it is able to make future predictions in the real world. To evaluate if the chosen model was the best choice for the project, another pre-defined model were built, using K Nearest Neighbors. The time consumed for training and testing, the accuracy and f-score of both models were compared.

Methodology

Data Preprocessing

The first step of the preprocessing task was to remove all the unnecessary features. According to the columns definition file for the dataset, some features were only descriptive features of other features. They are only pure text. Other features were about governmental information or irrelevant information about the patient (the place in which he was born, for

instance. These features were removed. The unnecessary features removed were *ESCOLARI, UFRESID, IBGE, CIDADE, CATEATEND, CLINICA, BASEDIAG, TOPOGRUP, DESCTOPO, DESCMORFO, ECGRUP, NAOTRAT, ANODIAG, CICIGRUP, CICISUBGRU, FAIXAETAR, INSTORIG, DRS, RRAS, DTULTINFO*, and *IBGEATEN*.

Some features contained several zeros or NaN values. These columns would not help the model to perform it's task. Below each column is showed followed by the most frequent value (the mode) and the percentage of the value for the feature in the dataset:

```
SEXO: value 1 - 50.12%
DIAGPREV: value 1 - 65.10%
N: value 0 - 50.83%
M: value 0 - 70.24%
PT: value nan - 51.34%
PN: value nan - 51.79%
PM: value nan - 53.64%
S: value 8 - 100.00%
G: value 8 - 94.12%
LOCALTNM: value 8 - 99.76%
IDMITOTIC: value 8 - 99.98%
PSA: value 8 - 98.35%
GLEASON: value 8 - 98.35%
OUTRACLA: value nan - 93.12%
META01: value nan - 87.13%
META02: value nan - 96.44%
META03: value nan - 98.98%
META04: value nan - 99.75%
TRATFANTES: value J - 100.00%
TRATFAPOS: value J - 96.58%
NENHUM: value 0 - 91.68%
CIRURGIA: value 1 - 62.31%
RADIO: value 0 - 73.12%
QUIMIO: value 0 - 64.68%
HORMONIO: value 0 - 87.89%
```

```
TMO: value 0 - 99.58%
IMUNO: value 0 - 99.43%
OUTROS: value 0 - 92.92%
NENHUMANT: value 1 - 99.92%
CIRURANT: value 0 - 100.00%
RADIOANT: value 0 - 100.00%
QUIMIOANT: value 0 - 100.00%
HORMOANT: value 0 - 100.00%
TMOANT: value 0 - 100.00%
IMUNOANT: value 0 - 100.00%
OUTROANT: value 0 - 100.00%
NENHUMAPOS: value 1 - 94.89%
CIRURAPOS: value 0 - 99.68%
RADIOAPOS: value 0 - 97.80%
QUIMIOAPOS: value 0 - 99.46%
HORMOAPOS: value 0 - 99.79%
TMOAPOS: value 0 - 99.97%
IMUNOAPOS: value 0 - 99.99%
OUTROAPOS: value 0 - 99.42%
CICI: value XIF - 59.84%
LATERALI: value 8 - 84.80%
PERDASEG: value 0 - 81.03%
ERRO: value 0 - 100.00%
```

As this list shows, the features *S, TRATFANTES, CIRURANT, RADIOANT, QUIMIOANT, HORMOANT, TMOANT, IMUNOANT, OUTROANT, ERRO*, have only static values. Except for the feature S, all of the others have only 0's in all dataset. These features were also removed. The features *META02, META03* and *META04* were also removed because more than 96% of the dataset had missing values for these three features.

The numerical features were normalized, The features IDADE, CONSDIAG and TRATCONST were normalized. Below I show the features before and after normalization:

IDADE

```
Minimum age  0
Maximum age  107
Mean of age  59.6355998711
Median of age  62.0
Standard Deviation of age  16.93
```

```
Minimum age  0.0
Maximum age  1.0
Mean of age  0.55734205487
Median of age  0.579439252336
Standard Deviation of age  0.1582
```

CONSDIAG

```
Minimum CONSDIAG  0
Maximum CONSDIAG  26524
Mean of CONSDIAG  48.0564134091
Median of CONSDIAG  20.0
Standard Deviation of CONSDIAG  149.96
```

```
Minimum CONSDIAG  0.0
Maximum CONSDIAG  1.0
Mean of CONSDIAG  0.00181180867927
Median of CONSDIAG  0.000754034082341
Standard Deviation of CONSDIAG  0.005
```

TRATCONS

```
Minimum TRATCONS  0.0
Maximum TRATCONS  1.0
Mean of TRATCONS  0.00226433917251
Median of TRATCONS  0.00120645453174
Standard Deviation of TRATCONS  0.005
```

```
Minimum TRATCONS  0.0
Maximum TRATCONS  1.0
Mean of TRATCONS  0.00226433917251
Median of TRATCONS  0.00120645453174
Standard Deviation of TRATCONS  0.005
```

The first step for processing the categorical features was to replace the missing values with the **mode** value for each feature. The mode for all features were showed in the list above. Below I list the features which contains missing values, and the number of missing values in the dataset:

```
Features with missing values:

PT           53920
PN           54397
PM           56336
OUTRACLA     97803
META04      104767
TRATHOSP         6
CICI           935
```

Then, the categorical features were one hot encoded into dummie variables. Below, the example of the feature SEXO after one hot encoding:

SEXO
2
2
1

SEXO_1	SEXO_2
0	1
0	1
1	0

The features *SEXO, DIAGPREV, TOPO, MORFO, EC, T, N, M, PT, PN, PM, G, LOCALTNM, IDMITOTIC, PSA, GLEASON, OUTRACLA, META04, TRATAMENTO, TRATHOSP, TRATFAPOS, TRATFANTES, DIAGTRAT, CICI*, and *LATERALI* were all one hot encoded.

To finish the preprocessing, the feature ULTINFO, which became the output class was also one hot encoded. The difference is that the values 1 and 2, for survival, were turned into 1, and 3 and 4, for death, became 0. At the end of the preprocessing, the dataset was transformed to the table below:

OUTROAPOS	ULTINFO	CONSDIAG	TRATCONS	PERDASEG	SEXO_1	SEXO_2	DIAGPREV_1	DIAGPREV_2	TOPO_C000	TOPO_C001
0	1	0.002188	0.016451	0	0	1	1	0	0	0
0	0	0.001446	0.001016	0	0	1	0	1	0	0
0	0	0.000000	0.000000	0	1	0	1	0	0	0

This is just a sample of all the features. After the preprocessing, the dataset was containing around **3600** features.

Implementation

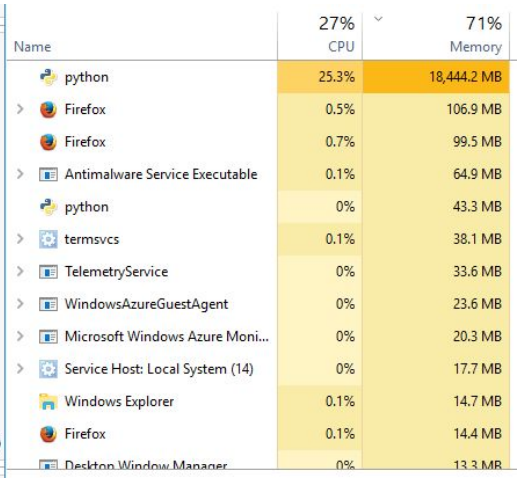
This part consisted of building, training, testing and evaluating the model. The dataset was splitted in 80% for training and 20% for testing.

```
Training set has 84026 samples.
Testing set has 21007 samples.
```

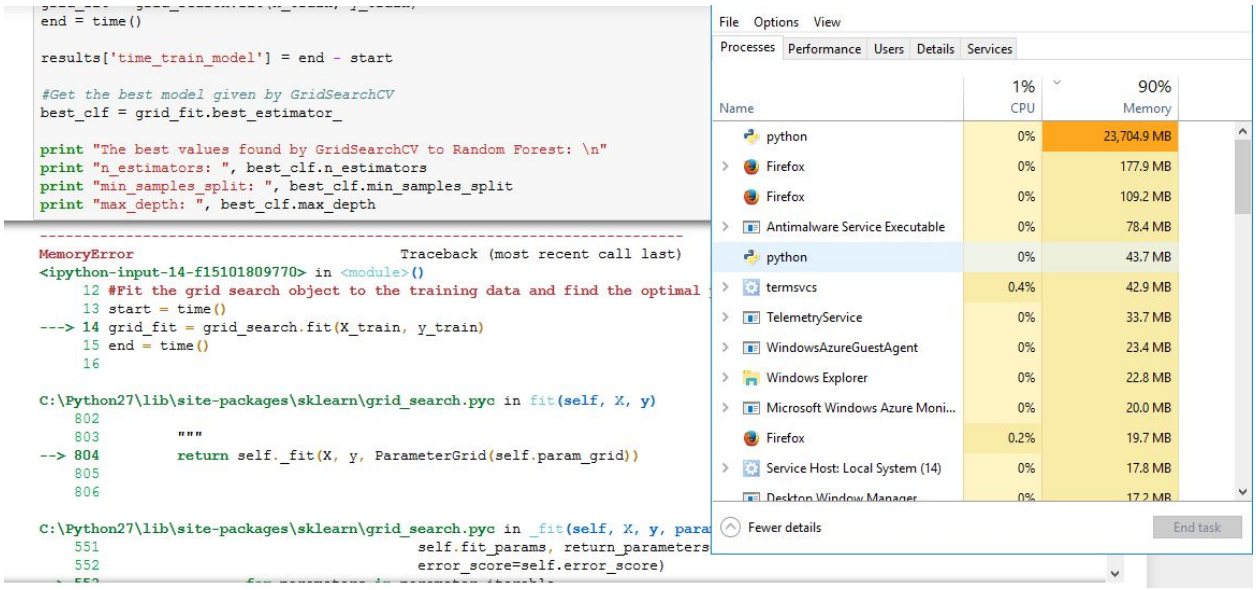
Then, these data sets were sent to the Random Forest and the KNN classifiers. To optimize the Random Forest, GridSearchCV was used along with cross validation. The KNN classifier was not optimized. After the training, both models were tested and the accuracy and f-score of both of them were measured.

The whole process was quite easy and straightforward. However, several issues occurred due to hardware limitations. The full dataset was too big and to heavy to be processed. Several memory errors occurred, in my personal laptop and even in a virtual machine created in Microsoft Azure. The machine used have 4 CPUs and 28GB of RAM. As shown below, in the fitting process of the KNeighborsClassifier the memory consumed by the process was around 18GB.

```
from sklearn.neighbors import KNeighborsClassifier

#Create the model for benchmarking
benchm_model = KNeighborsClassifier()

#Train the benchmarking model - calculate the time spent
start = time()
benchm_model = benchm_model.fit(X_train, y_train)
end = time()

results['time_train_b_model'] = end - start

#Make predictions with the benchmarking model - calculate the time spent
start = time()
b_model_predictions = benchm_model.predict(X_test)
end = time()
results['time_test_b_model'] = end - start

#Get the accuracy and fbeta for the benchmarking model
results['accuracy_b_model'] = accuracy_score(y_test, b_model_predictions)
results['fbeta_b_model'] = fbeta_score(y_test, b_model_predictions , beta=0.5)
```



The GridSearchCV was not able to run in the full dataset. The memory consumption jumped to 23GB, and the process gave a MemoryError like shown in the image below.



Unfortunately I was not able to solve this issue. I tried to change the parameter **n_jobs** for the GridSearchCV in order to alleviate the memory usage and force the processing in the CPUs but this approach did not solve the problem. **The only solution that worked was to slice the full dataset and use 1/7 of it.** This was the maximum number of data points that I could use without problems. Before slicing the data, the full dataset was totally shuffled.

```
Total number of records: 105033
Number of individuals who lived: 62296 - 59.31%
Number of individuals who died: 42737 - 40.69%
```

Refinement

GridSearchCV was the only tool used for refining. The goal was to refine the parameters max_depth, n_estimators and min_samples_split of the Random Forest classifier in order to get the best model. The best values for these parameters were the ones shown below.

```
The best values found by GridSearchCV to Random Forest:

n_estimators:  10
min_samples_split:  5
max_depth:  10
```

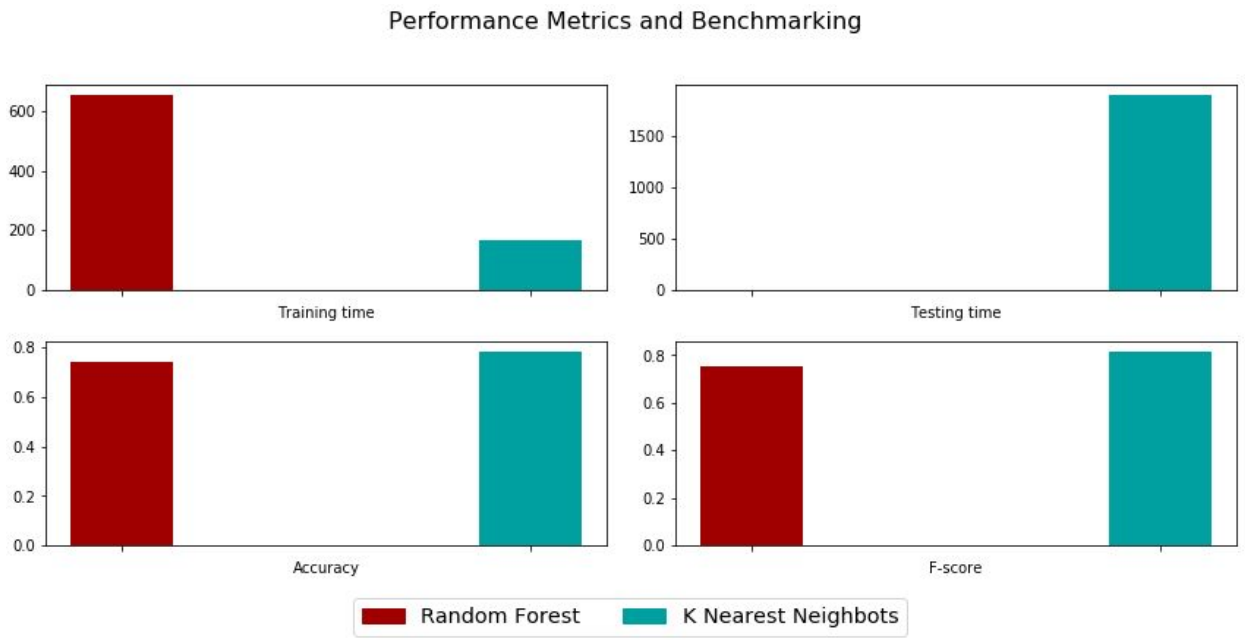
Results

Model Evaluation and Validation

Overall, the model had a decent performance in this dataset. The accuracy and f-score were inside an acceptable range of values, but the model spent a slightly big amount of time in training.

Justification

To compare the model to the benchmarking model, some graphs and a table are going to be presented below:



Metric	Optimized Random Forest Model	Benchmark KNN Model
Training time	655.22	168.20
Testing time	0.59	1896.06
Accuracy Score	0.74	0.78
F-score	0.75	0.81

The results shows that the random forest model is around 4x slower in training than the KNN model. This might be an advantage for the benchmarking model when training in large datasets. However, the KNN model is much slower than the optimized in testing. Around 3200x

slower. This is a poor result, and it is too expensive in terms of time consuming. The accuracy score were almost the same for both models, whereas the benchmarking model had a considerably higher F-beta score.

Conclusion

In conclusion, both models proved that it is possible to make predictions about the mortality of patients for ICESP. Both models had an acceptable accuracy, what suggests that both can predict future results, however, In my opinion, the optimized random forest model had a better overall result in this dataset in terms of time consumption and precision. If time to predict is an important factor the model to be chosen definitely is the random forest classifier. However the KNN proved to be an accurate model for this dataset, even if it wasn't optimized. Unfortunately, it seems that KNN will perform poorly in huge datasets in terms of computational cost. It is a *heavy* classifier. In this aspect, Random Forest definitely shines. In order to improve the Random Forest classifier, I can see no other solution (without making any search) than adding more data. I believe that more training and testing data would improve the model and increase the accuracy. Obviously in certain point the trade of between more data and accuracy would not worth it because much more data would not help the accuracy, but I believe that it is possible to achieve at least 85% of accuracy for this problem using this algorithm.