PROJECT TECHNICAL REPORT

Guangbo Yu

Introduction

Goal

Our project is to utilize Machine Learning models to predict the pre_ordered for surgeon before surgery and decrease the blood waste.
In this project, we will build a predictive model for predicting the value of PRBC_ordered to give surgeon an estimation of how much blood should they order before surgery and decrease the blood loss. The blood waste, which will serve as an evaluation method, is calculated by the difference of red_blood_cells used in surgeries and the PRBC_ordered.

Tools

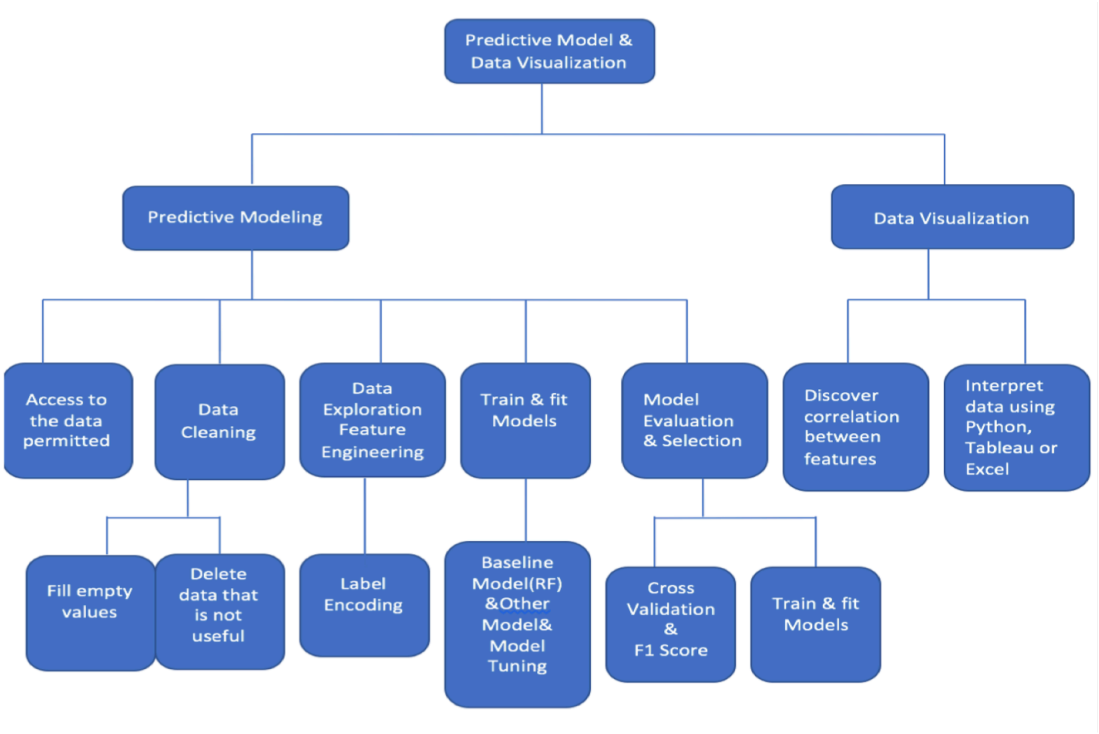
In this project, we will use python for the coding. The packages we used and will be used are pandas, numpy, seaborn, matplotlib, Sklearn, Scipy, Xgboost.

Problem Definition

In this project, to complete our goal, we will build two models. First, we used feature importance method to check the influence of each predictors and we found out that the most influential features are allergenic_blood_transfusion, surgery_speciality and EBL. Since the value of EBL, which is a post-surgery result, stands for estimated blood loss of one patient, we must predict the value of EBL first to predict the PRBC_Ordered. In real life, when a surgeon would like to estimate the blood which is needed for a surgery, he cannot know the estimated blood loss before surgery. But EBL does act as one important predictor, so our first model will output a new dataset which including the value of EBL and the output of the first model will act as an input for our second model. Then the second model will use this new dataset which contains EBL value to predict the pre-ordered red blood cells. So, the ultimate goal of this project is to build a pipeline of two models in order to give an accurate prediction on the pre-ordered blood if there is enough time.

The Learning System

Workflow



Detailed Steps

First, we started with data preprocessing, which includes filling missing data, outlier detection and data cleaning. Then we conducted feature engineering including label encoding for categorical data and data transformation for numerical data. And we built a predictive model (Random Forest) and evaluated it. Each step is highly related to the other and technical process may vary a little bit if there is a potentially better solution.

The Learning Algorithm

In this project, we will use Random Forest Regressor as our baseline model. First. What is a random forest? Random Forest is an ensemble tree method, which means it is comprised of many decision trees. As for each decision tree, the output may vary best on the problem itself. If it is a classification problem, then the output is a class, if it is a regression problem, them the output is a specific value. As for the random forest, each tree will apply the bagging method, which means only a part of the

features and the dataset will be chosen for one tree, and then the results of each tree will be calculated into one final value. If this is a classification problem, then the final value is calculated by the voting method, which means the class having the most votes will be our final result. If it's a regression problem, then the value is calculated by the weighted average of each tree.

The advantages of Random Forest are as below: first, we're not sure if there is any non-linear relation between predictors and Y. If the relation between x and y is not monotonic, tree model produces better results than linear models. Second, train model is fast to train. Last, it's easy to do the feature selection by checking feature importance.

But also, there are some disadvantages of this model: First, Random Forest is incomprehensible. It is a black box method which could lead to a lack of insights for the result. Second, it doesn't work well if categorical data has too many values and it doesn't work at all if we use one-hot encoding.

Split-data:

```
from sklearn.cross_validation import train_test_split

Train_X = data.drop(['prbc_ordered'],axis = 1)
Y = data.prbc_ordered

Train_Y = np.log1p(Y.values.reshape(-1,1))
X_train,X_test,Y_train,Y_test = train_test_split(Train_X,Train_Y,test_size = 0.2)
```

Training Model:

```
from sklearn.ensemble import RandomForestRegressor
clf = RandomForestRegressor()
clf.fit(X_train,Y_train)
```

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_split=1e-07, 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=None,
           verbose=0, warm_start=False)
```

Experimental Evaluations/ Tests/ Results

The Data Sets

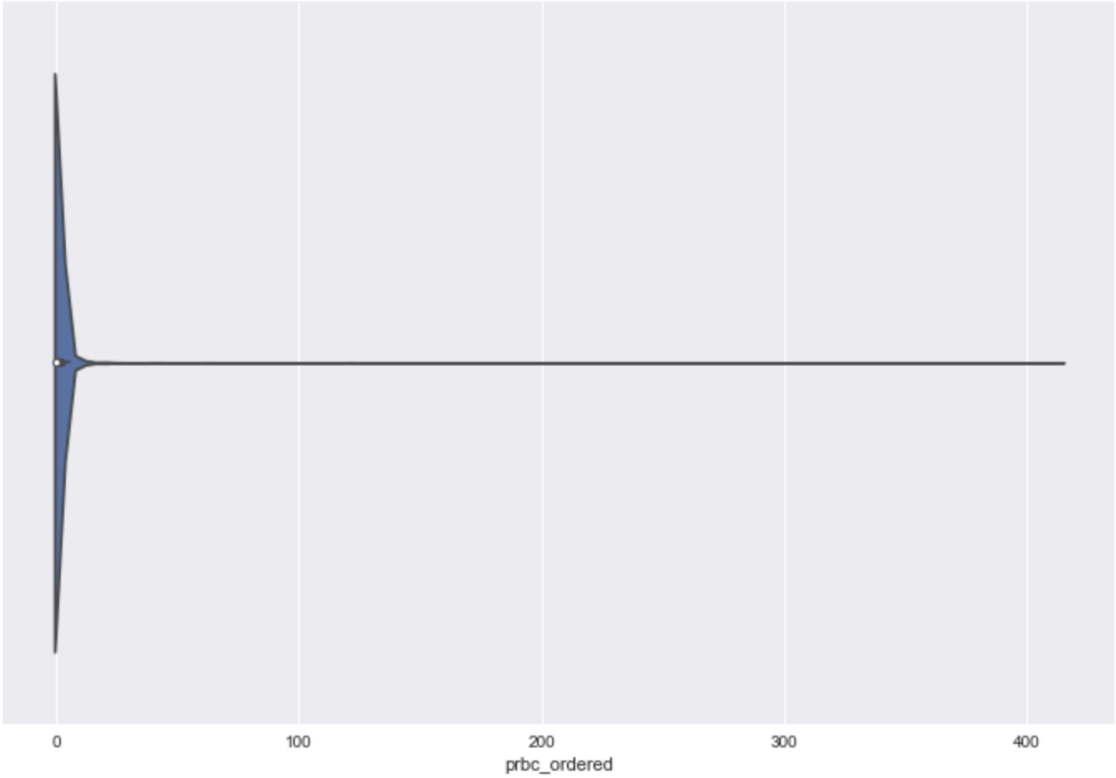
Filling Missing Data

Imputation Method

Mean value for missing data except for prbc_ordered, in which I used 0 to fill the missing value.

Outlier Detection

In this project, since prbc_ordered is our final prediction, we would try to figure out if there are major outliers first. Below is the distribution of this feature. According to this distribution, we would delete several data points that are very distant from the major range. So, we set our threshold as prbc_ordered <= 100 since only a few data points are beyond 100.



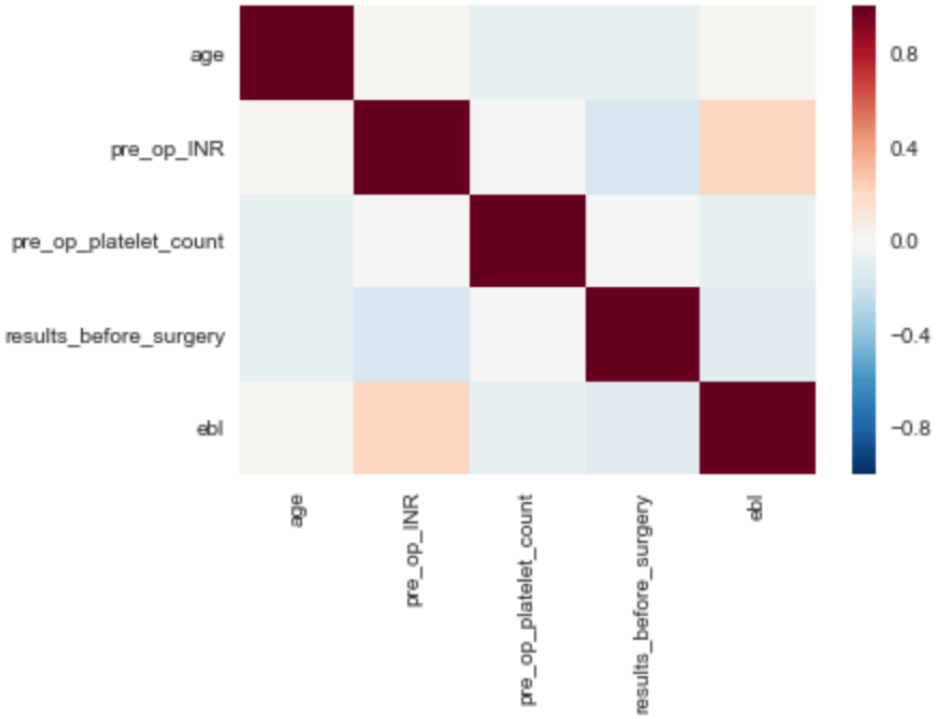
Data Cleaning

Also, there are other outliers which could have bad influence on the regression model. So, we plotted out the distribution graph and set our thresholds for data cleaning. All the data points which are higher than the threshold would be omitted and only within 100 data points are deleted. Apparently, it is safe to clean our data in this method.

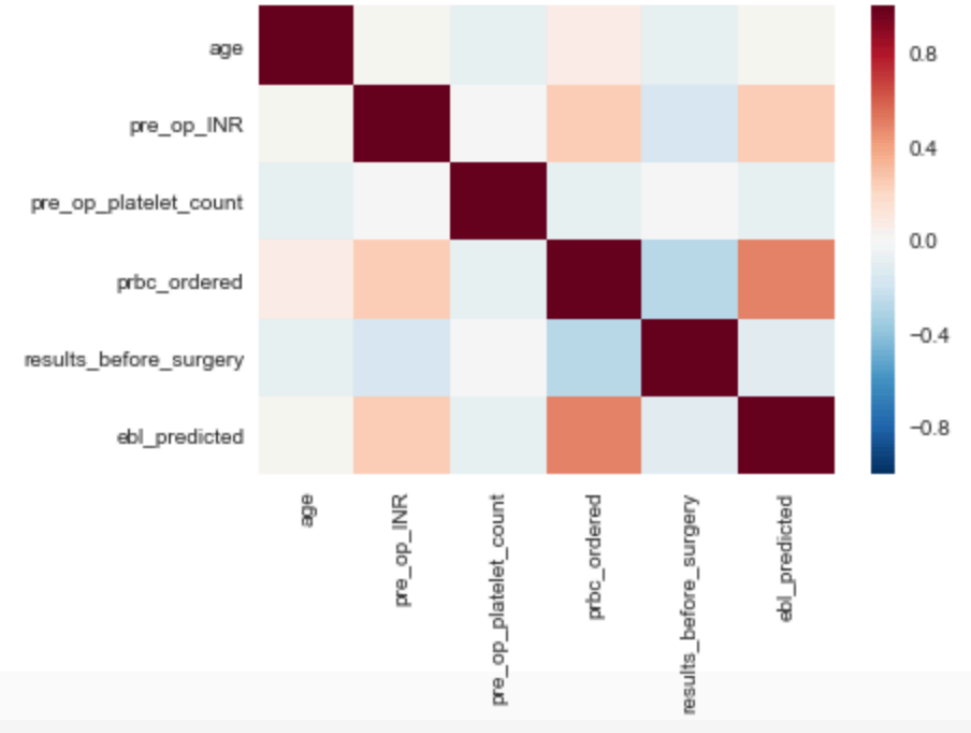
```
data = data[data.pre_op_INR <= 10]
data = data[data.pre_op_platelet_count <= 10000]
data = data[data.ebl <= 10000]
data = data[data.prbc_ordered <= 100]
data = data[data.red_blood_cells <= 30]
data = data[data.fresh_frozen_plasma <= 50]
data = data[data.platelets <= 40]
data = data[data.cryoprecipitate <= 80]
data = data[data.pre_op_INR <= 10]
data = data[data.pre_op_platelet_count <= 10000]
```

Correlation Matrix

As for numerical data, we plotted the correlation matrix in a heat map to avoid collinearity by checking if there are highly correlated features. If there are highly linearly correlated pairs, which means one could be not significantly important, we need to omit one of the features. And luckily, in both models, there were no highly correlated pairs. Model1 for predicting ebl:



Model2 for predicting prbc_ordered:



Feature Engineering

Label Encoding

There are many encoding methods for dealing with this dataset. Since our baseline model is Random Forest Regressor which is an ensemble tree method, then label encoding will outperform other encoding methods, especially one-hot encoding, which convert categorical values into feature sets.

Label Encoding

```
from sklearn import preprocessing

for col in cat_col:
    print ("label encoding %s:" % col)
    LBL_Model = preprocessing.LabelEncoder()
    LBL_Model.fit(data[col])
    labels = dict(zip(data[col].unique(), LBL_Model.transform(data[col].unique())))
    print (labels)
    #print (LBL_Model.classes_)
    data[col] = LBL_Model.transform(data[col])
```

Data transformation

The goal of the data transformation is to decrease the skewness of numerical data, thus improving the performance of the predictive model. After we conducted label encoding for categorical data, we also did feature engineering on numerical data. First, we checked the skewness for each of our numerical features. Then we set our threshold, which is 0.5, for high skewness and we conducted log transformation to make the data less skewed as it was.

```
skewed_cols = data[con_col].apply(lambda x: skew(x.dropna()))
print (skewed_cols.sort_values())
```

```
pre_op_platelet_count    -4.902737
results_before_surgery   -1.335195
pre_op_INR               -1.179715
age                      -0.460732
ebl_predicted            -0.319031
red_blood_cells           3.302621
prbc_ordered              3.548171
dtype: float64
```

```
Threshold = 0.5
col_skewness = [i for i in con_col if (abs(skewed_cols[i]) >= Threshold) & (i != 'prbc_ordered')
for each in col_skewness:
        data[each] = np.log1p(data[each])
```

Predictive Modeling

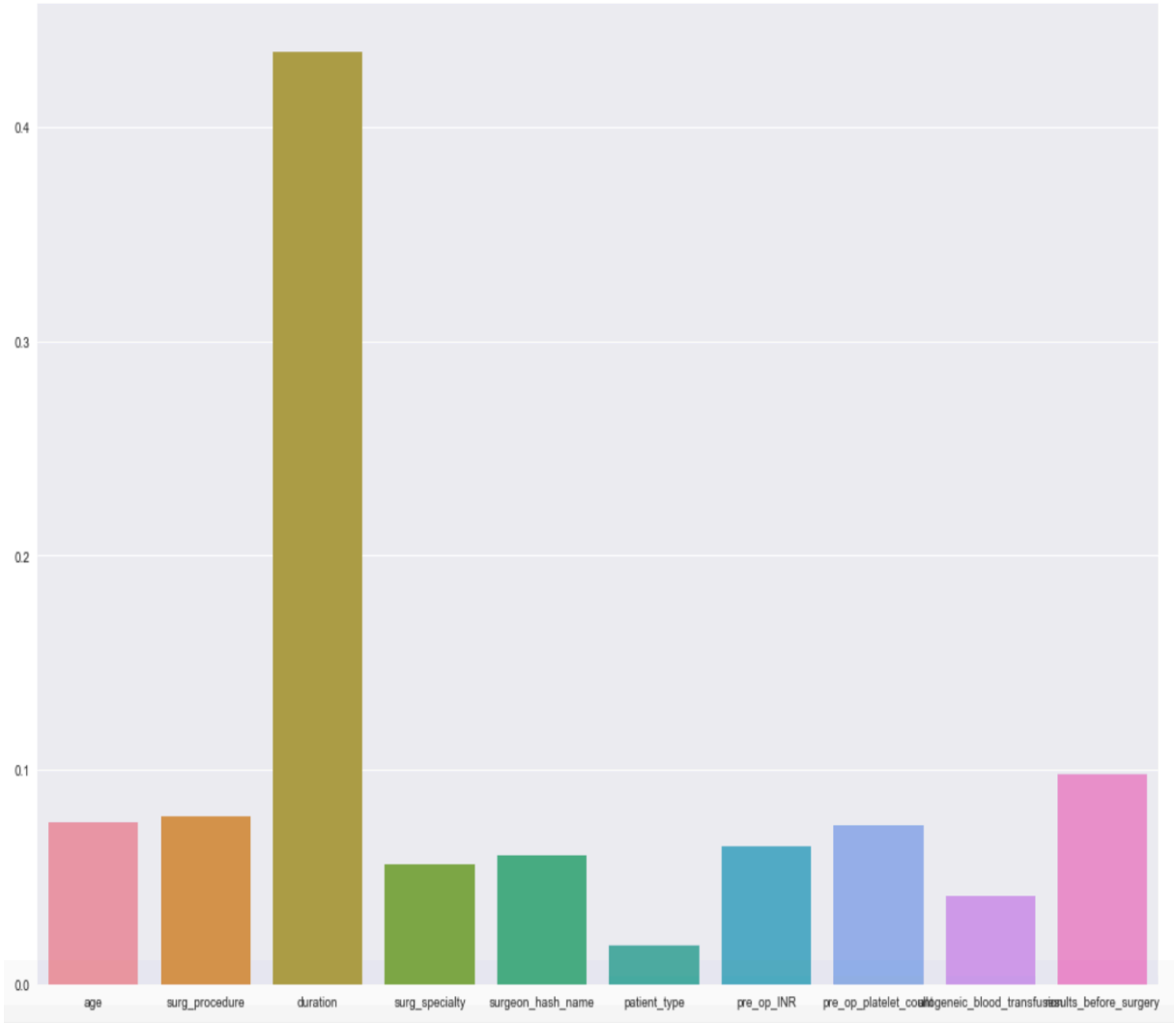
Baseline Model

We will use Random Forest as out baseline Model, the code is as follows:

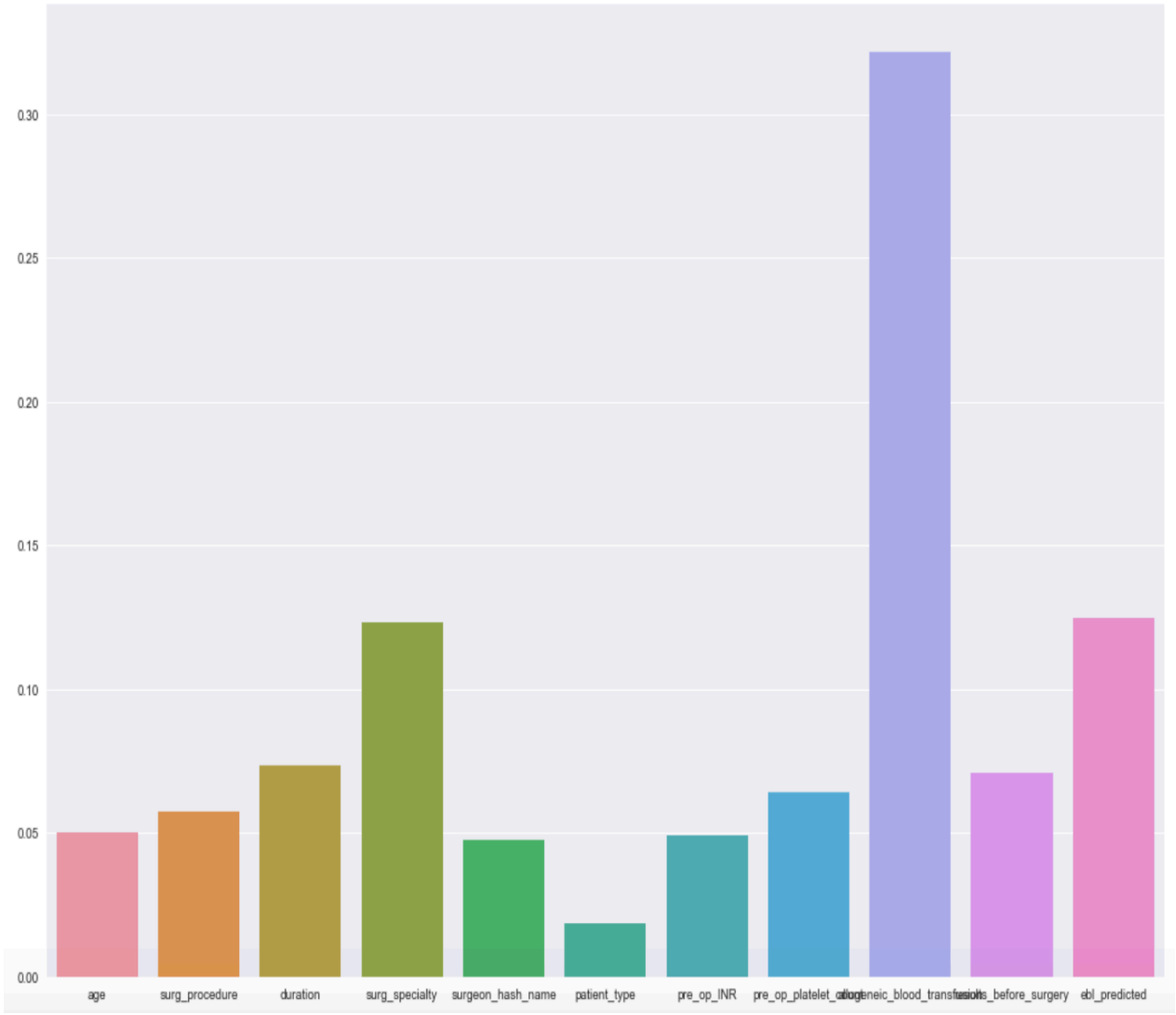
```
from sklearn.ensemble import RandomForestRegressor
clf = RandomForestRegressor()
clf.fit(X_train,Y_train)
```

Feature Importance:

After we built our baseline model, we checked the feature importance. Which may lead us to some insights. As for predicting EBL, the most influential predictor is the surgery duration. Which means the surgery duration is informative in predicting the estimated blood loss. But the relation is unknown yet.



As for predicting PRBC_Ordered, influential predictors are allergenic blood transfusion, EBL, surgeon name and surgery category. As for allergenic blood transfusion, we speculate that this is the most important feature since it only has two values. Besides, different surgery categories may require different volume of blood. Estimated blood loss will also somehow infer the value of pre-ordered blood.



Another Learning Algorithm

Linear Regression Model

In this project, we will also use linear regression model as a comparison to the random forest regressor. Linear regression performs well if the data has a linear shape but performs bad in capturing the non-linearity. The code is as below.

```
from sklearn import linear_model,metrics
model = linear_model.LinearRegression()
model.fit(X_train,Y_train)
```

Evaluation Method & Comparing Learning Systems

There is no fixed evaluation method at all. Evaluation varies based on your requirement and how well is your domain knowledge. Therefore in our case, we want to predict the value of prbc_ordered as well as decrease the blood waste. So our evaluation method is measuring the performance of the model by measuring the loss of red_blood_cell.

Red_blood_cell_loss is the value of prediction substract the value of red_blood_cells used in the surgery (Predicted_Prbc_ordered – red_blood_cells).

Also, we need to consider the situation where the predicted prbc_ordered is smaller than the used blood cell, which could lead to a very serious situation where the blood is not enough in the surgery, thus leading to some consequences. So, we also need to consider the number of negative values in wasted_blood. In our case, Random Forest Regressor outperforms Linear Regression.

Evaluation Method

```
##To evaluate our models, we would use the predicted values - red_blood_cells used
##to represent the blood waste. So the smaller the value the better
## RF
data['estimated_order'] = clf.predict(Train_X)
data['waste'] = np.subtract(data.estimated_order,data.red_blood_cells)
penalty_element = sum(n < 0 for n in data.waste)
print 'number of elements whose value is lower than the used blood cell:', penalty_element
print 'mean value of the waste blood:', data.waste.mean()
```

```
number of elements whose value is lower than the used blood cell: 1159
mean value of the waste blood: 0.323104240293
```

```
## LR
data['estimated_order'] = model.predict(Train_X)
data['waste'] = np.subtract(data.estimated_order,data.red_blood_cells)
penalty_element = sum(n < 0 for n in data.waste)
print 'number of elements whose value is lower than the used blood cell:', penalty_element
print 'mean value of the waste blood:', data.waste.mean()
```

```
number of elements whose value is lower than the used blood cell: 3451
mean value of the waste blood: 0.318083114382
```

Concluding Remarks & Future Work

According to the feature importance, we found those influential predictors. If we have enough time, we can visualize the correlation between those predictors and the Y to find some insights. Also, as for data cleaning part, we found a more appropriate way for dealing with part of the missing data.

As for future work, we may build another model which is called xgboost and compare it with Random Forest based on the evaluation method. Then we will tune the models to make it perform better.