CSCI-GA.3033-061 Predictive Analytics August 20, 2022

Solutions to Problem 1 of Final Exam Summer 2022 (125 Points)

Name: Anav Prasad (ap7152; N18439284) *Due: 11:59pm on Saturday, August 20*

Collaborators:

Unsupervised Learning Algorithms

Consider the following supervised learning algorithms (most of which has been adopted in your project and some were discussed in class)

- Support Vector Machines (25pts)
- Decision Trees (25pts)
- Random Forest (25pts)
- Neural Nets (25pts)
- KNN (25pts)

For each algorithm answer the following points, feel free to cite your sources if you used any.

1. Briefly describe the algorithm
2. Advantages of the algorithm
3. Disadvantage of the algorithm

Solution:

- **Support Vector Machines:**
  - **Description:**
    Support Vector Machines (SVMs) are supervised machine learning algorithms used for classification and regression problems. For classification, given a labelled training dataset, they work by training and perfecting a hyperplane that tries to perfectly classify the training dataset into two classes. The trained hyperplane can then be used to classify the test dataset by just plotting the points and seeing on which side of the hyperplane they fall. For regression, SVMs work by trying to find a hyperplane that best fits (by minimizing the distance between the training data points and the hyperplane) the given data and using the same for regression thereon.
  - **Advantages (for Classification):**
    * More effective in high dimensional spaces (that is when number of features are high).
    * Subsequently, also more effective when the number of dimensions (or features) are greater than the number of test samples.

    - Works well when the classes' data points, when plotted, land far from each other (making the drawing of a hyperplane easier).
    - Memory efficient as it uses a subset of training points (called support vectors) in the decision function to classify test points.
    - Different kernel functions (to increase to the number of dimensions so as to make the SVM more effective) can be specified for the decision functions and, its possible to specify custom kernels.
    - Also, SVMs are largely quite stable since small tweaks and minor changes to the data points does not greatly affect the hyperplane and, thus, its accuracy.
  - **Advantages (for Regression)**:
    - Easy implementation
    - Easy to generalize.
    - Since it tries to fit the flattest possible hyperplane through the data points within a given margin (given by ϵ), it handles outliers well.
  - **Disadvantages (for Classification and Regression):**
    - Not suitable for large datasets.
    - Since SVMs work by drawing hyperplanes through the data to classify it, it does not work well in cases when the data has more noise and, the different classes' data points are intermixed so thoroughly that even the use of custom kernels cannot properly separate them.
    - It might be apparent from the last point but the choosing and finding of an appropriate kernel is also quite difficult.
    - SVMs tend to under-perform when the number of features end up actually exceeding the number of training data points.
    - Results can be difficult to interpret. Also, somewhat relevantly, since the classification is done by interpreting where the data points with respect to the hyperplane, there is no probabilistic explanation for the classification.

- **Decision Trees:**
  - **Description:**
    A decision tree is a directed acyclic graph with no cross edges where internal node represents a *decision* and each leaf node represents an outcome. Decision trees are used for classification and regression purposes, with the internal nodes representing a key features whose value decision decides the final label/value of the test data.
  - **Advantages:**
    - Requires less data pre-processing as compared to other algorithms. Does not require scaling / normalization of the data.
    - Gives a very good idea about the relative importance of different features.
    - Can capture and utilize non-linear relationships.
    - Can handle missing values in the data.
    - Comparatively fast to run and efficient.

    - Can handle various input data types of features.
    - Very easy to intuitively understand.
  - **Disadvantages:**
    - Small changes in the data can cause huge changes to the decision tree. It is unstable that way.
    - Decision Trees are prone to overfitting.
    - The calculations for the creation of the decision tree can become very complex sometimes if the depth of the tree is high.
    - Training / Creation of the decision tree can often take significant amount of time depending on the training data given.
    - The complexity of the decision tree can sometimes spiral uncontrollably if either the number of feature is high or the number of distinct values for a feature is high (thus creating huge number of branches).

- **Random Forest:**
  - **Description:** Random Forest is a supervised machine learning algorithm that makes use of an ensemble of numerous decision trees to make its prediction. Thus, like Decision Trees, it can be used to solve classification and regression problems. It works by creating multiple random samples of the given training dataset and creating decision trees for each of the samples. While testing, the test example is fed to each of the decision trees and, a voting system is used to tally and get the final label/value from all of the labels/values obtained from the numerous decision trees.
  - **Advantages:**
    - Reduces the overfitting problem of decision trees by the existence of multiple decision trees voting on the final result. Thus improves accuracy over decision trees.
    - Can handle missing values.
    - Is robust against outliers.
    - Unlike Decision Trees, Random Forest is relatively stable as it doesn't gets affected by small changes in the data.
    - Does not require scaling or normalization of the data.
    - Can capture and utilize non-linear relationships.
    - Can handle various input data types of features.
  - **Disadvantages:**
    - It can become very complex very quickly due to the existence of high number of trees.
    - Consequently, it takes significantly longer time to train than decision trees.
    - It performs poorly in unbalanced datasets due to its use of the voting system.

- **Neural Networks:**
  - **Description:**
    Neural Networks, also known as Artificial Neutral Networks (ANNs), refer to machine

learning algorithms that try to mimic the structure of the human brain (by the use of several artificial neurons) to make predictions of any sort (be it classification or regression). ANNs comprise of several layers of 'neurons' that each process information and pass it along to the next layer for further processing. This allows ANNs to capture several deep relationships in data that would otherwise be very difficult to capture. The neuron of an ANN is simply a data structure that takes some input, performs some basic calculations on it (weight multiplication; bias addition), and forwards the output.

- **Advantages:**
  - Neural Networks have wide applications. From object detection to handwriting recognition to stock market prediction to health tracking and illness projection, neural networks have the capability to capture the subtleties in all kinds of data and provide valuable and useful outputs.
  - Neural Networks can be modified so that they continually keep on learning the more they are used.
  - Can handle missing information and work around the same.
  - Relatively stable. Neural Networks are not affected by minor changes in the data.
  - Efficient and fast to generate predictions / outputs.
  - With modifications to its structure, can process unorganized data.
- **Disadvantages:**
  - Can be very computationally expensive to train and require powerful hardware or a significant amount of time to train.
  - Require significant amount of training data to start giving decent results.
  - The users have no control over what the individual neurons are learning. Thus, faulty ANNs can be very difficult to debug and understand.
  - Requires lot of experimentation to perfect. As the last point mentioned, the programmer does not generally have much idea about what the neurons in the network are learning. As such, perfecting the structure of the neural network (number of neurons, number of layers etc) can be a chore.
  - Limited by numerical inputs. Neural networks requires its input to be in numerical format and does not understand anything else. While it can be easy to transform various kinds of inputs to numbers, it reduces the intuitive understand of the programmer of the entire system.
  - It can be difficult to understand when the network should be stopped training to stop it from overfitting.

- **kNNs:**
  - **Description:**
    k Nearest Neighbors (kNNs) is a simple machine learning algorithm that can be used for both classification and regression problems. Traditionally, kNN does not have a training period, instead it directly stores all the training points it has along with their labels. In the testing phase, it just finds the k nearest points to the test point, finds their label, and uses a voting system (might be 1-0, weighted by distance, etc) to get the label for the test point.

- **Advantages:**
  * Only one hyperparmeter (k).
  * Can be used for classification and regression.
  * Can capture non linear boundaries.
  * Can use variety of distance functions as per need (such as Euclidean, Manhattan, etc)
  * Very easy to introduce new data to this.
  * Very intuitive and simple to understand.
  * Very easy to implement
- **Disadvantages:**
  * Computationally expensive in cases of huge training dataset because, during the testing phase, it would need to compute the distance with every point in the training set to make its prediction.
  * Also, computationally expensive when the number of dimensions (or features) are high.
  * Requires scaling or normalization of features.
  * Choosing the optimal number of neighbors can be a chore.
  * Cannot handle imbalanced data.
  * Sensitive to outliers.
  * Cannot handle missing values in the data.

□

CSCI-GA.3033-061 Predictive Analytics August 20, 2022

Solutions to Problem 2 of Final Exam Summer 2022 (45 Points)

Name: Anav Prasad (ap7152; N18439284) *Due: 11:59pm on Saturday, August 20*

Collaborators:

Apply a decision Tree algorithm to derive the decision tree learned from the following dataset (25pts)

Weekend	Weather	Parents	Money	Decision
W1	Sunny	Yes	Rich	Cinema
W2	Sunny	No	Rich	Tennis
W4	Rainy	Yes	Rich	Cinema
W5	Rainy	No	Rich	Stay In
W6	Rainy	Yes	Poor	Cinema

Show all steps.

Explain how you would you apply Random Forests to the same dataset (briefly describe the algorithm being applied (20pts)), you do not need to apply the algorithm.

Solution:

PART 1: APPLYING ID3 DECISION TREE ALGORITHM

First of all, let's compute the entropy of *Decision*.

$$\begin{aligned} Entropy(Decision) &= -\frac{3}{5} \cdot \log_2\left(\frac{3}{5}\right) - \frac{1}{5} \cdot \log_2\left(\frac{1}{5}\right) - \frac{1}{5} \cdot \log_2\left(\frac{1}{5}\right) \\ &\approx 1.371 \end{aligned}$$

- **Step 1:**
  Now, let's compute the entropy and information gains of the various columns:

– Weather:

$$\begin{aligned}
Entropy(Sunny) &= -\frac{1}{2} \cdot \log\left(\frac{1}{2}\right) - \frac{1}{2} \cdot \log\left(\frac{1}{2}\right) \\
&= 1 \\
Entropy(Rainy) &= -\frac{2}{3} \cdot \log\left(\frac{2}{3}\right) - \frac{1}{3} \cdot \log\left(\frac{1}{3}\right) \\
&\approx 0.918 \\
Entropy(Weather) &= \frac{2}{5} \cdot 1 + \frac{3}{5} \cdot 0.918 \\
&= 0.9508 \\
IG(Weather, Decision) &= Entropy(Decision) - Entropy(Weather) \\
&= 1.371 - 0.9508 \\
&= 0.4202
\end{aligned}$$

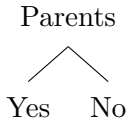
– Parents:

$$\begin{aligned}
Entropy(Yes) &= -\frac{3}{3} \cdot \log\left(\frac{3}{3}\right) \\
&= 0 \\
Entropy(No) &= -\frac{1}{2} \cdot \log\left(\frac{1}{2}\right) - \frac{1}{2} \cdot \log\left(\frac{1}{2}\right) \\
&= 1 \\
Entropy(Parents) &= \frac{3}{5} \cdot 0 + \frac{2}{5} \cdot 1 \\
&= 0.4 \\
IG(Parents, Decision) &= Entropy(Decision) - Entropy(Parents) \\
&= 1.371 - 0.4 \\
&= 0.971
\end{aligned}$$

– Money:

$$\begin{aligned}
Entropy(Rich) &= -\frac{2}{4} \cdot \log\left(\frac{2}{4}\right) - \frac{1}{4} \cdot \log\left(\frac{1}{4}\right) - \frac{1}{4} \cdot \log\left(\frac{1}{4}\right) \\
&= 1.5 \\
Entropy(Poor) &= -\frac{1}{1} \cdot \log\left(\frac{1}{1}\right) \\
&= 0 \\
Entropy(Money) &= \frac{4}{5} \cdot 1.5 + \frac{1}{5} \cdot 0 \\
&= 1.2 \\
IG(Money, Decision) &= Entropy(Decision) - Entropy(Money) \\
&= 1.371 - 1.2 \\
&= 0.171
\end{aligned}$$

Since the Information Gain of *Parents* is the highest, we can add it as the first node in the decision tree.



- **Step 2:**
  First, let's compute entropy of $Decision|Yes$ and $Decision|No$

$$\begin{aligned}
Entropy(Decision|Yes) &= -\frac{3}{3} \cdot \log\left(\frac{3}{3}\right) \\
&= 0 \\
Entropy(Decision|No) &= -\frac{1}{2} \cdot \log\left(\frac{1}{2}\right) - \frac{1}{2} \cdot \log\left(\frac{1}{2}\right) \\
&= 1
\end{aligned}$$

Now, since the entropy of $Decision|Yes$ is 0, and also by looking at our dataset, we can conclude the following:

$$(Decision|Yes) \rightarrow Cinema$$

The above always happens irrelevant of $Money$ and $Weather$ values.

Therefore, we can directly put $Cinema$ under Yes in the decision tree.

As for No, let's compute the entropy and information gains of the various columns:

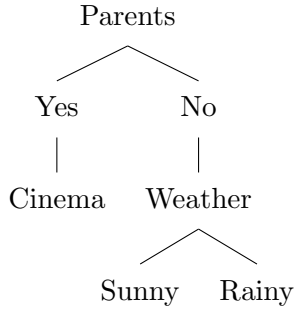
– $Weather|No$:

$$\begin{aligned}
Entropy(Sunny|No) &= -\frac{1}{1} \cdot \log\left(\frac{1}{1}\right) \\
&= 0 \\
Entropy(Rainy|Yes) &= -\frac{1}{1} \cdot \log\left(\frac{1}{1}\right) \\
&= 0 \\
Entropy(Weather|No) &= \frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 0 \\
&= 0 \\
IG(Weather, Decision|No) &= Entropy(Decision|No) - Entropy(Weather|No) \\
&= 1 - 0 \\
&= 1
\end{aligned}$$

– $Money|No$:

$$\begin{aligned}
Entropy(Rich|No) &= -\frac{1}{2} \cdot \log\left(\frac{1}{2}\right) \\
&= 1 \\
Entropy(Money|No) &= \frac{2}{2} \cdot 1 \\
&= 1 \\
IG(Money, Decision|No) &= Entropy(Decision|No) - Entropy(Money|No) \\
&= 1 - 1 \\
&= 0
\end{aligned}$$

We must pick *Weather* to go under *No* because of its higher information gain.

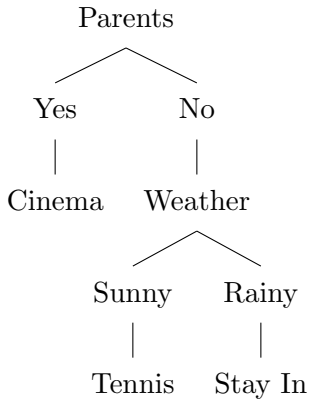


- **Step 3:**

At this point, we face one node choice at each branch; so, we can easily complete the tree.

$$(Decision|No, Sunny, Money = Rich) \rightarrow Tennis$$
$$(Decision|No, Sunny, Money = Poor) \rightarrow NA$$
$$(Decision|No, Rainy, Money = Rich) \rightarrow StayIn$$
$$(Decision|No, Sunny, Money = Poor) \rightarrow NA$$

So, the final decision tree would be like as follows:



PART 2: RANDOM FOREST ALGORITHM

Random Forest Classifier works by creating numerous random samples of the given training dataset and creating a decision tree out of each of the samples. The testing is done by running the testing data through each of the decision trees and using a voting system to then decide the final label/class from the results obtained from each decision tree.

I would apply the Random Forest Classifier to this dataset by using *RandomForestClassifier* model in the *scikit-learn* python library. I would do the data preprocessing using the *pandas* python library before feeding it into the RandomForestClassifier. The pseudo-code for the same is as follows:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import pandas as pd

data = pd.read_csv(filename, index_col=0)
train_features = ['Weather', 'Parents', 'Money']
X = data[train_features]
y = data['Decision']
test_size = 0.2 # if you want to do 80-20 train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = test_size)
n_estimators = 500 # The number of decision trees that you want RF Classifier to have
model = RandomForestClassifier(n_estimators = n_estimators)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
## Then y_pred can be used to check various evaluation metrics and plot various graphs
```

□

CSCI-GA.3033-061 Predictive Analytics August 20, 2022

Solutions to Problem 3 of Final Exam Summer 2022 (40 Points)

Name: Anav Prasad (ap7152; N18439284) *Due: 11:59pm on Saturday, August 20*

Collaborators:

You are hired as a senior data scientist at a large financial institution. The firm's executive committee is NOT happy with the in-house built search engine that provide information retrieval services across the institution for large number of users. The search engine was built using the open-source Apache Solr project on a Hadoop Cluster. The search engine provides over 100 Terabytes of text documents that include financial reports, lessons learned, presentations among others.
The executive committee received complains that the search results are irrelevant to the user's query.
After a preliminary investigation, your team found out that there is absence of contextual search in your search engine.

*"**Contextual search** is a form of optimizing web-based **search** results based on context provided by the user and the computer being used to enter the query."*

You will be sending an email of in exactly a maximum of one page that include your high-level strategy to improve the precision and recall of the search engine by adding context to the search experience (Contextual Search). The users of the system are professionals of finance background.

Write a concise email that has the following format:

1. Include in brief bullet points the actions that you will take to lead the project following a data science lifecycle.
2. Sketch a high level architecture that will include the names of the frameworks, algorithms, ETL tools, data science problems you will be solving (data classification, clustering, mining association rules...), data sources, possible infrastructures or tools and vendors and the skills needed in your team members that you will hire.
3. Bullet points of benefits on your strategy, framework and tools adopted.

Solution:
Dear [Mr. X],

To add the ability to recognize context in the search engine, I plan on following the upcoming roadmap:

- First, I will begin by studying the various common search strings that the search engine will encounter so that the contextual search is tailored for [company]'s purposes.
- Next, I will study the text corpus of the search engine so as to derive insights to best modify the search engine model.

- Next, I will pre-process the corpus so that it is ready for the contextual search model that I will be introducing and training.
- Next, I will build my contextual search model and train it on the corpus.
- Following that, I will be evaluating my contextual search model on various metrics to get a proper evaluation of the same.
- Finally, I will deploy my model to fix the issue of the search engine.

To achieve my aforementioned goals, I will be making use of ETL tools such as RapidMiner and Hadoop. Aside from that, I'll be making use of open-sourced Bidirectional Encoder From Transformers (BERT) model to achieve my contextual search task. Finally, I'll be making use of open-sourced Apache Solr 9.0 search engine to optimize search further. To make proper use of these tools, I will be needing and recruiting team members well versed in RapidMiner or Hadoop, the BERT Model, and Apache Solr.

I will be using the above mentioned tools and models for the following reasons.

- ETL tools such RapidMiner and Hadoop allow for testing of simple models to test their efficacy in a quick and easy manner. This will allow me to find out which methods work the most well and, thus, focus my attention on the same.
- The BERT Model, released in November 2018, is an open-sourced the state of the art Natural Language Processing (NLP) model. Usage of the same would grant me the best possible contextual analysis of the search strings and, therefore, lead to the best contextual search engine that I could provide.
- Apache Solr 9.0, released in May 2022, is a package from Apache Lucene library and is one of the most popular as well as very accurate search engines available today. If successfully integrated, it would optimize the search engine very significantly.

Best Regards,
Anav Prasad

□

CSCI-GA.3033-061 Predictive Analytics August 20, 2022

Solutions to Problem 4 of Final Exam Summer 2022 (50 Points)

Name: Anav Prasad (ap7152; N18439284) *Due: 11:59pm on Saturday, August 20*

Collaborators:

Customer Churn – Conceptual Questions (50pts). Please answer the following questions in your own words, using complete sentences.

1. What is Customer Churn? (5 points)

   Solution: Customer Churn refers to the fact that a set of the people, who were customers at the beginning of a time period, would stop being so (either by opting out of the service due to a variety of reasons or, by not renewing their subscription to the service (if the service is of subscription format)) at the end of the said time period. □

2. What is Customer Churn Rate? (5 points)

   Solution: Customer Churn Rate refers to the percentage loss of customers (or, intuitively, the *rate* at which the customers are *churning*) at the end of a time period. □

3. Why is predicting customer churn important? (5 points)

   Solution: Predicting customer churn is important because it allows a company to know how their customer base size is going to increase/decrease and to adjust their strategies accordingly. Moreover, knowing customer churn, they can try and adjust their strategies to reduce the same by various methods such as increasing incentives / benefits for the the customers likely to leave. □

4. What are the two main types of customer churn? (5 points)

   Solution: The main two types of customer churn are voluntary and involuntary whose meanings are apparent. Voluntary churn refers to customers whose circumstances have not changed but have voluntarily decided to leave the service. Involuntary churn refers to customers who have been forced to leave due to some circumstances beyond their control. □

5. When preparing your data for customer churn, what are the 3 main variables that you track? (5 points)

   Solution: The three main variables to track are as follows:

   - Recency
   - Frequency
   - Monetary Value

□

6. What is customer information type diversity? What are some examples of it? (5 points)

   Solution: Customer Information Type Diversity refers to the diversity among the customers that are churning. For example, of people, when they reach a certain age, might decide to opt out. Knowing and expecting that might be valuable to a company in predicting customer churn. □

7. What are four common classification models used for predicting Customer Churn? (5 points)

   Solution: The four most common classification models used for predicting customer churn are as follows:

   - Logistic Regression
   - Random Forest
   - Decision Trees
   - Binary Classification

   □

8. Let's say you are running a subscription-based company. What are some methods you can implement in your product/ company to help determine why Customers are churning? (5 points)

   Solution:

   - The simplest, if not very reliable way, would be to ask the churning customers the reason why they are opting out of the service and get their feedback.
   - A market survey can be done to determine the cause behind customer churn by comparing what their competitors are doing.
   - The customers can be actively monitored to analyze their behavior and data mining can be done to try and determine the true reason behind churning.

   □

9. What is the inherent flaw in customer churn analytics? (5 points)

   Solution: One of the flaws in customer churn analytics is the human randomness. There are innumerable reasons behind why people opt for or out of different services and it may be impossible to get enough data on people to accurately predict their decisions 100% of times. □

10. What is Customer Segmentation? How does it relate to customer churn? (5 points)

Solution: Customer Segmentation refers to the process by which the customers can be grouped into different segments based on common traits (such as demographics or behavior). It is very useful in customer churn analytics since, if a particular segment has high customer churn rate, that segment's traits can be used to counter the churn rate. For example, customers who generally opt for cheap services might be tempted by targeted offers of other cheap and useful services. □

CSCI-GA.3033-061 Predictive Analytics August 20, 2022

Solutions to Problem 5 of Final Exam Summer 2022 (50 Points)

Name: Anav Prasad (ap7152; N18439284) *Due: 11:59pm on Saturday, August 20*

Collaborators:

For this mini exercise provide snapshots and answers as well as RapidMiner processes used in your answers. **Make sure to try at least 3 different modeling techniques in order to obtain the best Churn prediction possible, and tabulate their accuracy results below.**

- Explain processes such as Feature selection or generation, sampling, etc., in the initial phase of CRISP – DM which has helped you to achieve best results. (10 points)
- Try and categorize most of the features provided in this dataset to the concept of RFM (Recency, Frequency, and Monetary) that was described in the demo. E.g., The feature Monthly charges can fall under Monetary category of Churn evaluation. (10 points)
- Explain why handling imbalanced datasets is necessary and what are some ways of doing so? Provide screenshots from your RapidMiner exercise supporting this. (10 points)
- Describe in short the results of all the models that you tried (with screenshots) and any hyperparameter optimization involved in modeling. (10 points)
- What evaluation metrics seemed more apt for the Churn evaluation according to your opinion and why? Explain in detail the insights one can gain from evaluating the Churn dataset. (10 points)

Solution:

- There were several initial steps to finish that was crucial to make the data ready for modeling.
  - **Replace Missing Values:**
    The given dataset had a few missing values that needed to be filled so that PCA and, further on, modeling could be done. I decided to replace missing values with averaged values as they seemed to give the best results.
  - **Nominal to Numeric:**
    Then, since I wanted to do PCA, I needed to convert the data of features such as 'gender', 'Partner', 'Dependents', etc to numerical data. For that I used the 'Nominal to Numeric' operator of RapidMiner to achieve my goal. The said operator worked by mapping columns that binary but non-numerical data to 0s and 1s. The other non-numerical columns were converted into multiple binary columns that contained the same information as before but in numeric format. For example, the field 'InternetService' that had three discrete non-numeric values {'DSL', 'Fiber optic', 'No'} was converted into three numeric binary columns 'InternetService = DSL', 'InternetService = Fiber optic' and, 'InternetService = No'.

– **PCA:**
I used PCA, or Principal Component Analysis, to reduce the number of dimensions of the given dataset. After experimenting with various variance thresholds, I came to realize that all of the non target columns (obviously excepting 'customerID') were highly correlated and thus, PCA, reduced them to one column even at a high variance threshold of 0.99. But, when all the columns were combined into one column, the accuracy always seemed to drop.

Figure 1: Logistic Regression without PCA

accuracy: 80.23% +/- 1.47% (micro average: 80.23%)

	true No	true Yes	class precision
pred. No	2300	419	84.59%
pred. Yes	277	525	65.46%
class recall	89.25%	55.61%	

Figure 2: Logistic Regression with PCA

accuracy: 73.19% +/- 0.14% (micro average: 73.19%)

	true No	true Yes	class precision
pred. No	2577	944	73.19%
pred. Yes	0	0	0.00%
class recall	100.00%	0.00%	

Thus, I decided to try keeping a fixed number of components in PCA and treated the fixed number of components as a hyperparameter. The results of that are mentioned below. But, overall, I did not receive more than 0.87% improvement in accuracy from doing PCA.

Figure 3: Optimize Parameter called on Number of components of PCA

Optimize Parameters (Grid) (19 rows, 6 columns)

iter... ↑	PCA (2).number_of_components	accuracy	precision	recall	f_measure
1	1	0.735	?	0	?
2	2	0.791	0.648	0.479	0.547
3	3	0.780	0.634	0.424	0.506
4	4	0.780	0.620	0.448	0.519
5	5	0.795	0.645	0.506	0.566
6	6	0.798	0.648	0.520	0.576
7	7	0.797	0.648	0.517	0.575
8	8	0.800	0.655	0.524	0.580
9	9	0.802	0.661	0.527	0.585
10	10	0.803	0.659	0.537	0.591
11	11	0.800	0.652	0.532	0.585
12	12	0.805	0.667	0.542	0.597
13	13	0.811	0.677	0.545	0.604
14	14	0.805	0.665	0.535	0.592
15	15	0.804	0.656	0.550	0.597
16	16	0.807	0.668	0.542	0.597
17	17	0.801	0.652	0.542	0.591
18	18	0.807	0.667	0.551	0.602
19	19	0.801	0.655	0.530	0.584

- **Sampling:**
  The need for this is described in third subpart (that is the next subpart) of this question.

Out of all of this, PCA helped me the get most increase in accuracy by 0.87

- **RFM:**
  - **Monetary Features:**
    MonthlyCharges, TotalCharges, Dependents, Partner, tenure, MultipleLines, InternetService, OnlineSecurity, OnlineBackup, DeviceProtection, StreamingTV, StreamingMovies
  - **Frequency Features:**
    Contract, Payment Method, Paperless Billing
  - **Recency Features:**
    PhoneService, TechSupport

- Improperly balanced datasets add an automatic bias to the set by the very reason that they are unbalanced. Consider rare diseases, such as Thalassemia. Thalassemia afflicted people have a rate of about 1 in 100,000 people in the United States. So, a model, made to predict if people have Thalassemia or not, did its prediction by flat out saying that a person does *not* have Thalassemia in all cases, it would still have an accuracy rate of about $\sim 99.999\%$. This would make it seem that the model does extremely well when we know that it does not work at all.
  Proper subsampling of the dataset could eliminate this issue.
  In the given dataset, the results with and without sampling (at 50:50 ratio) are as follows:

Figure 4: Logistic Regression without Sampling

accuracy: 80.39% +/- 1.18% (micro average: 80.39%)

	true No	true Yes	class precision
pred. No	4631	838	84.68%
pred. Yes	543	1031	65.50%
class recall	89.51%	55.16%	

Figure 5: Logistic Regression with 50:50 Sampling

accuracy: 80.43% +/- 1.75% (micro average: 80.43%)

	true No	true Yes	class precision
pred. No	2320	422	84.61%
pred. Yes	267	512	65.73%
class recall	89.68%	54.82%	

While doing sampling here only gives us a marginal improvement over the result obtained without doing any sampling, the following example shows (in which 30:70 sampling is done)

Figure 6: Logistic Regression with 30:70 Sampling

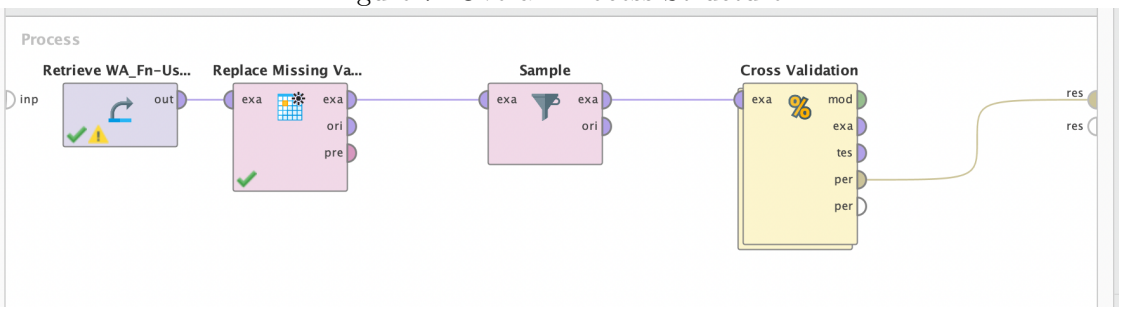
accuracy: 75.87% +/- 2.35% (micro average: 75.87%)

	true No	true Yes	class precision
pred. No	1166	304	79.32%
pred. Yes	386	1004	72.23%
class recall	75.13%	76.76%	

the stark difference between balanced dataset accuracy and improperly balanced dataset accuracy.

- **Modeling Results:**

Figure 7: Overall Process Structure



The results with different models are as follows:

– **Logistic Regression:**

Figure 8: Logistic Regression Structure in Cross Validation

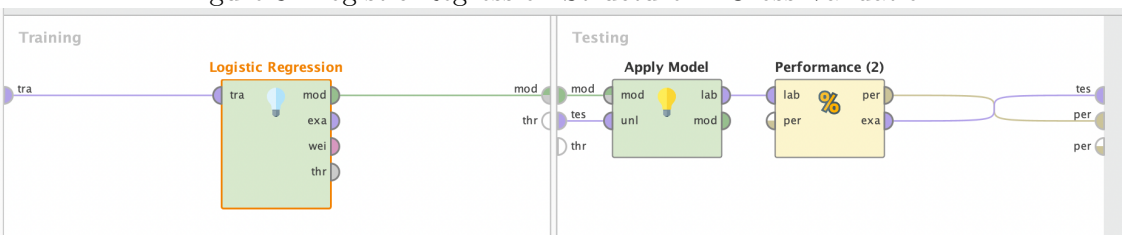


Figure 9: Logistic Regression Result

accuracy: 80.43% +/- 1.75% (micro average: 80.43%)

	true No	true Yes	class precision
pred. No	2320	422	84.61%
pred. Yes	267	512	65.73%
class recall	89.68%	54.82%	

– **Gradient Boosted Trees:**

Figure 10: Gradient Boosted Trees Structure in Cross Validation

accuracy: 77.39% +/- 2.48% (micro average: 77.39%)

	true No	true Yes	class precision
pred. No	2061	270	88.42%
pred. Yes	526	664	55.80%
class recall	79.67%	71.09%	

Figure 11: Gradient Boosted Trees Result

accuracy: 77.39% +/- 2.48% (micro average: 77.39%)

	true No	true Yes	class precision
pred. No	2061	270	88.42%
pred. Yes	526	664	55.80%
class recall	79.67%	71.09%	

– **Decision Tree:**

Figure 12: Decision Trees Structure in Cross Validation

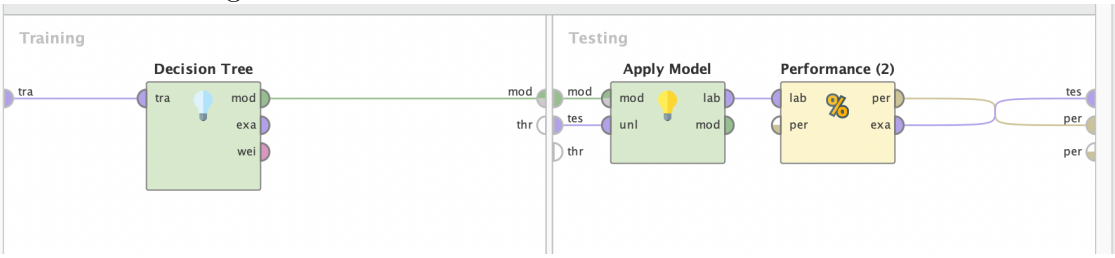


Figure 13: Decision Tree Result

accuracy: 76.12% +/- 1.55% (micro average: 76.11%)

	true No	true Yes	class precision
pred. No	2149	403	84.21%
pred. Yes	438	531	54.80%
class recall	83.07%	56.85%	

– **Random Forest:**

Figure 14: Random Forest Structure in Cross Validation

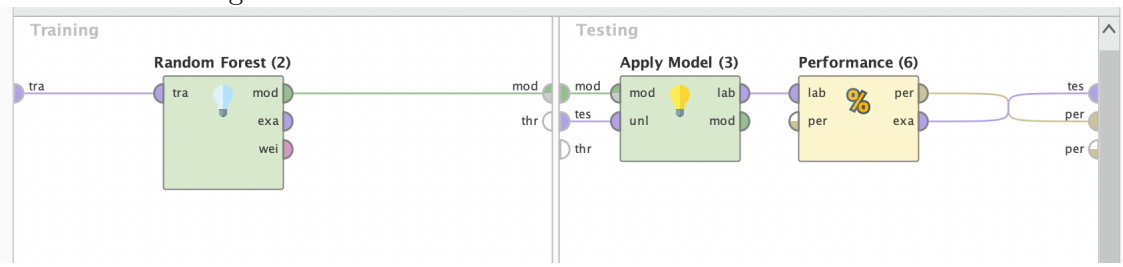


Figure 15: Random Forest Result

accuracy: 80.40% +/- 1.45% (micro average: 80.40%)

	true No	true Yes	class precision
pred. No	2366	469	83.46%
pred. Yes	221	465	67.78%
class recall	91.46%	49.79%	

– Tried to optimize number of decision trees in random forest:

Figure 16: Overall Process Structure with Optimize Parameters Operator

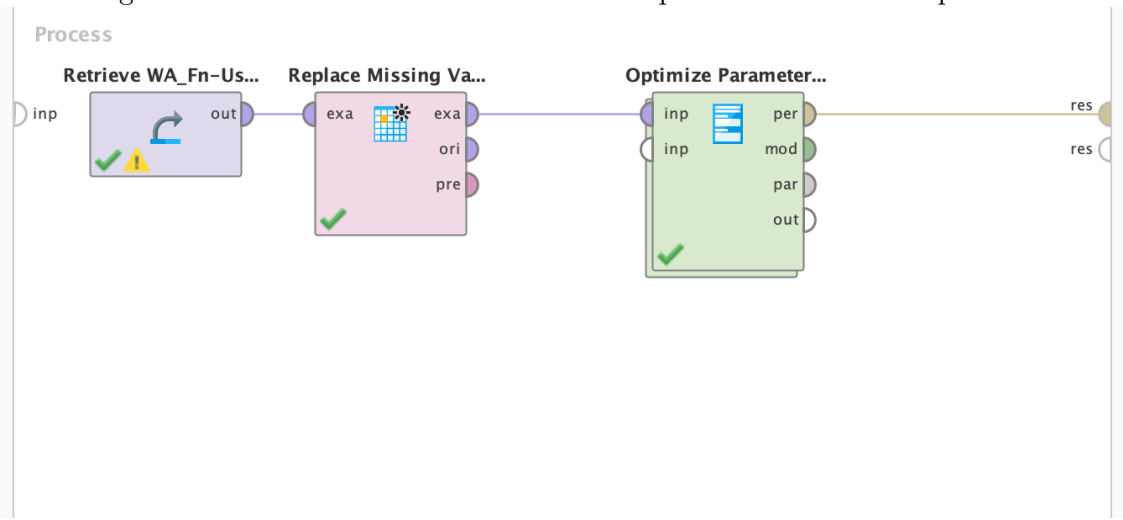


Figure 17: Optimize Parameter Structure

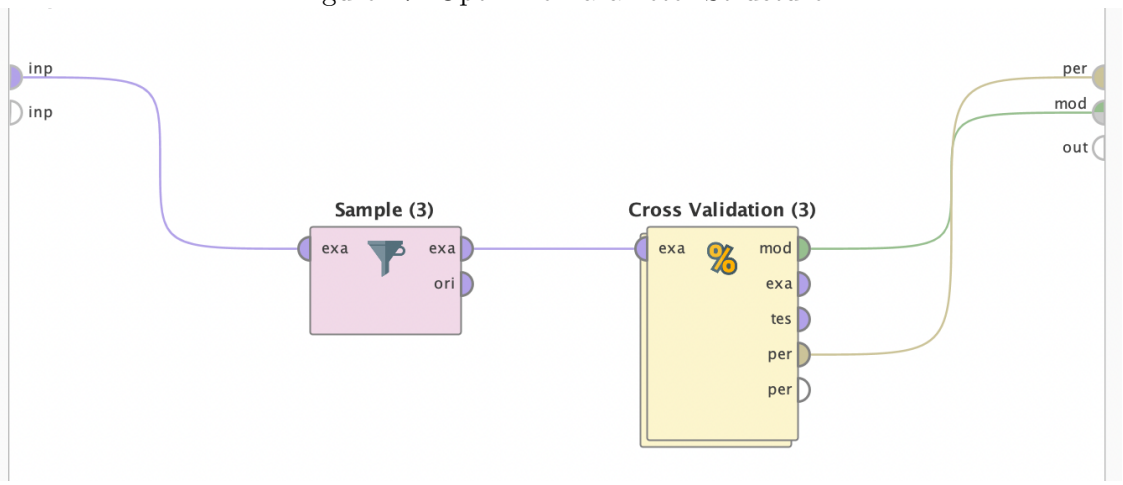


Figure 18: Random Forest Structure in Cross Validation

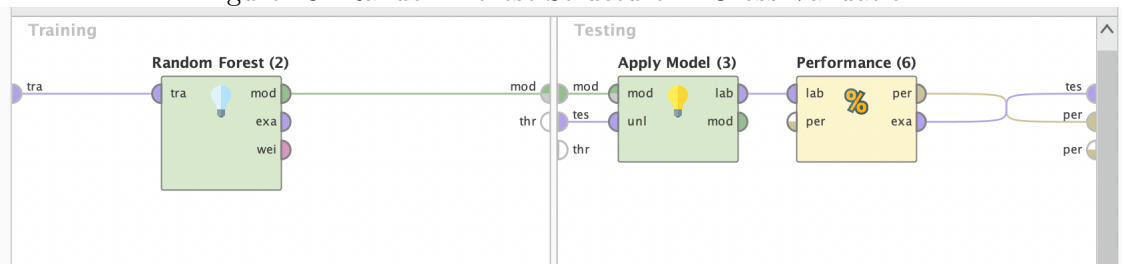


Figure 19: Results

Optimize Parameters (Grid) (2) (11 rows, 6 columns)

iteration	Random Forest (2).number_of_trees	accuracy	precision	recall	f_measure
1	100	0.786	0.653	0.414	0.506
2	190	0.800	0.682	0.469	0.553
3	280	0.798	0.670	0.472	0.552
4	370	0.787	0.636	0.458	0.532
9	820	0.797	0.655	0.507	0.571
6	550	0.801	0.673	0.490	0.567
5	460	0.800	0.665	0.502	0.571
7	640	0.793	0.659	0.458	0.540
8	730	0.798	0.667	0.479	0.556
10	910	0.802	0.673	0.490	0.567
11	1000	0.796	0.668	0.457	0.542

- **Evaluation Metrics:**
  The best evaluation metrics for Customer Churn Analytics are as follows:
  - **Accuracy:**
    Accuracy is used for trivial reasons. It allows us to see the percentage of correct churn classifications of our model.
  - **F1-Score:**
    The F1 Score is the harmonic mean of Precision and Recall. It helps give us a balanced idea of how the model is performing on the Churn class. There is usually a trade off between Precision and Recall.
  - **AUC:**
    The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve. The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes which, in this case, is Yes and No labels of Churn our churn column (customer churned or not churned).

  The results using F1-Score and AUC as metrics are as follows:
  - Logistic Regression:

Figure 20: Logistic Regression AUC

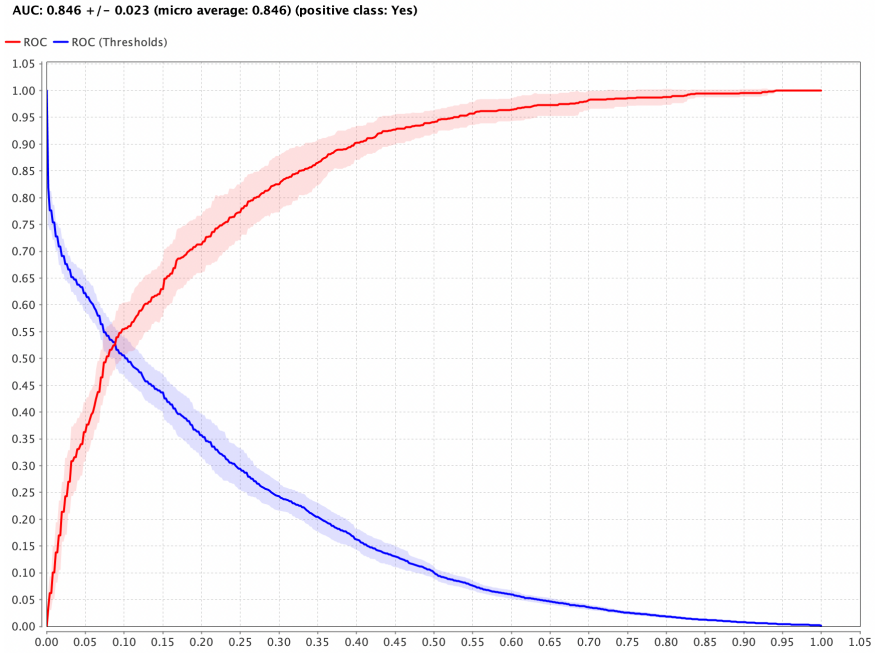


Figure 21: Logistic Regression F1 Score

f_measure: 59.76% +/- 3.36% (micro average: 59.78%) (positive class: Yes)

	true No	true Yes	class precision
pred. No	2320	422	84.61%
pred. Yes	267	512	65.73%
class recall	89.68%	54.82%	

– Gradient Boosted Trees:

Figure 22: Gradient Boosted Trees AUC

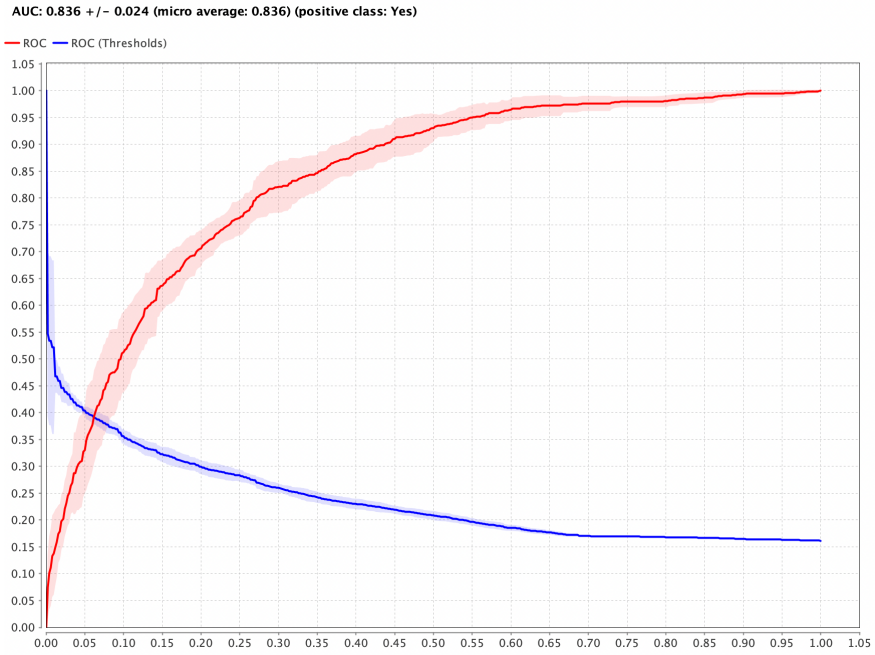


Figure 23: Gradient Boosted Trees F1 Score

f_measure: 62.58% +/- 2.98% (micro average: 62.52%) (positive class: Yes)

	true No	true Yes	class precision
pred. No	2061	270	88.42%
pred. Yes	526	664	55.80%
class recall	79.67%	71.09%	

– Decision Tree:

Figure 24: Decision Tree AUC

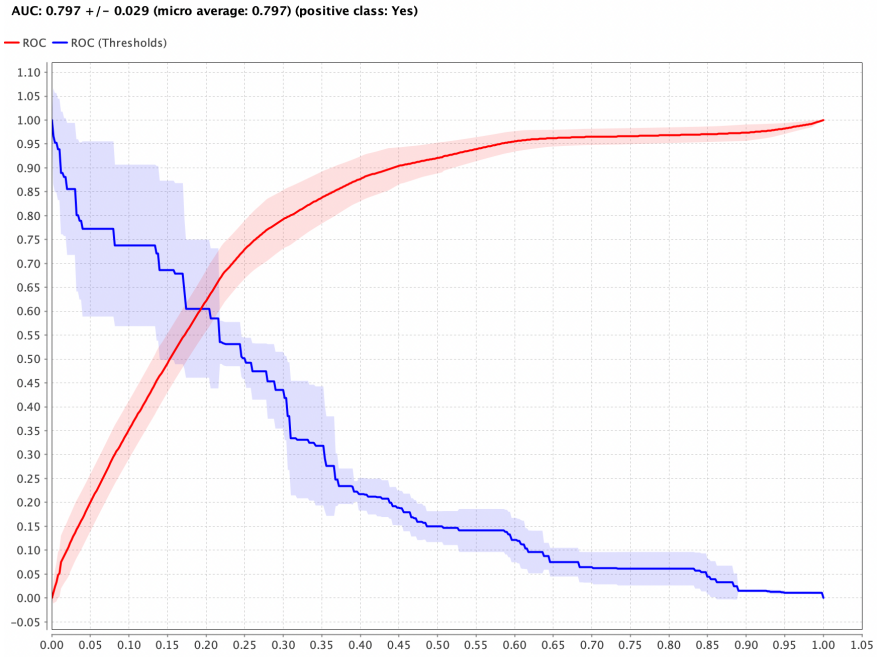


Figure 25: Decision Tree F1 Score

f_measure: 53.49% +/- 13.81% (micro average: 55.81%) (positive class: Yes)

	true No	true Yes	class precision
pred. No	2149	403	84.21%
pred. Yes	438	531	54.80%
class recall	83.07%	56.85%	

– Random Forest:

Figure 26: Random Forest AUC

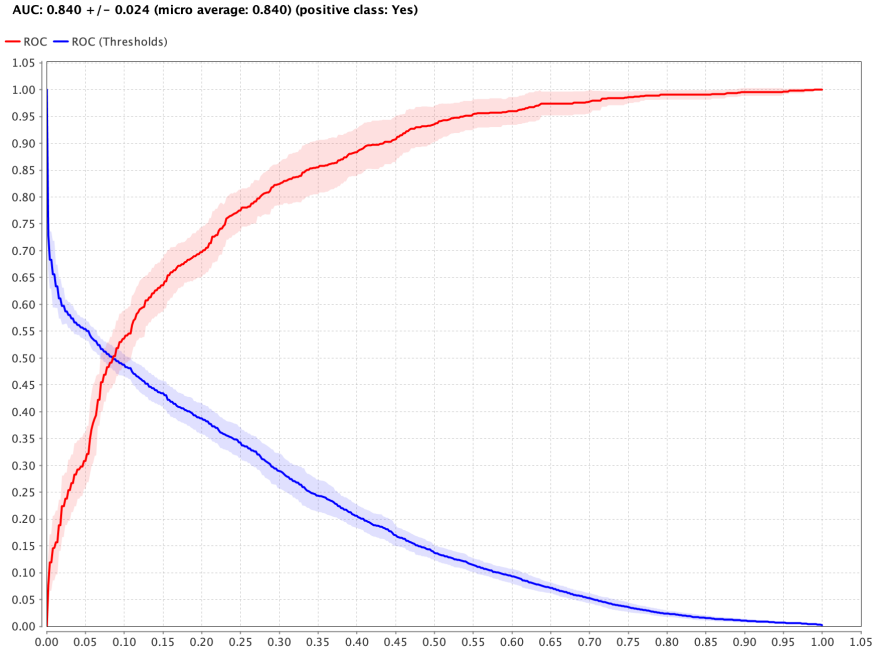


Figure 27: Random Forest F1 Score

f_measure: 57.31% +/- 3.80% (micro average: 57.41%) (positive class: Yes)

	true No	true Yes	class precision
pred. No	2366	469	83.46%
pred. Yes	221	465	67.78%
class recall	91.46%	49.79%	