Machine Learning Engineer Nanodegree

Capstone Project

Shotaro Fujimoto

February 18th, 2019

I. Definition

Project Overview

Today, more and more businesses are moving to a subscription based model – from cell phone plans to Amazon Prime to Netflix. It's incredibly easy source of recurring revenue, as opposed to a la carte selling. The one big thing that works against subscription based business is customers canceling the service, or churning.

In addition a lot of these subscription businesses offer a free trial period. They get the payment information and automatically bill the customer after the free trial period is over.

I work at AT&T, and it's DirecTV NOW (DTVNOW) internet based entertainment product is no different. It's a subscription business and we pay very close attention to customer churn. DTVNOW also offers a 7 day free trial.

Here is a similar problem tackled by a Saas subscription company classifying forever free customers and paid subscribers:

https://datastories.com/gallery/predictive-user-scoring-example

Another use case, this time written by the analytics consulting company that worked with a startup to solve a similar questions:

https://www.strong.io/blog/predicting-customer-behavior-machine-learning-to-identify-paying-customers

Problem Statement

Since DTVNOW is a subscription product, we lose revenue when a customer discontinues service, or churns. By using supervised learning, we can build a model that can predict if a customer will churn or stay after the first several days of the trial period. Then, we can proactively reach out to customers that we think won't stick, and market to them so they will stay on the product.

Some of the features that the model will look at will be account level data – customer demographics, package, location, tenure, and October 2018 viewership. And of course, if the customer still had service at the end of October or not.

Some of the machine learning algorithms I plan to use are SVM, Random Forest, and Naïve Bayes.

Strategy

- Get data of October 2018 DTVNOW new free trial accounts, and all of the above features
- Use aforementioned supervised learning algorithms to correctly predict if the free trial accounts will convert into paying customers
- If we can indeed correctly predict the conversion, then we can give a list of the free trial customers to the marketing team so they can craft outreach strategies to convince them to be paid customers.
- This can be done by email marketing, or even giving a small credit to entice them to use the product for a few months.

Metrics

F1 score would be the primary evaluation metric of the project. I decided on the F1 score because I believe it is the best balance of precision and recall (Shung, 2018).

Precision is a good measure to determine, when the costs of False Positive is high, such as email spam detection. And recall is a good measure to determine when the cost associated with False Negative will be extremely high, such as diseases.

For this project, since false positive and false negatives are similarly bad for the company (reaching out to a customer that would have kept the subscription or not reaching out to a customer that won't keep the product), I believe the f1 score is a good indicator of the quality of the model.

The equation of the F1 score is as follows:

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

II. Analysis

The features that I used for the analysis are as follows:

- The type of package (Live a Little - $40, Just Right - $55, Go Big- -$65, Gotta Have It - $75,Hispanic - Todo Y Mas - $45)
- What sales channel they were acquired from
- Do they have a wireless product with AT&T or not
- Do they have a DirecTV cable box product with AT&T or not
- Do they have an UVerse cable box product with AT&T or not
- Do they have an internet broadband product with AT&T or not
- Do they have an landline product with AT&T or not

- How many seconds of viewership did they have on 0 days after sign up
- How many seconds of viewership did they have on 1 day after sign up
- How many seconds of viewership did they have on 2 days after sign up
- How many seconds of viewership did they have on 3 days after sign up
- Did they have video startup failures
- Did they have video startup exit before video start (user induced video errors; EBVS for short)
- Average start time of their successful plays
- Average rebuffering ratio
- Did they watch any sports channels
- Did they watch any news channels
- Did they watch any kids channels
- What percentage of their viewership was live content
- What percentage of their viewership was DVR content
- What percentage of their viewership was VOD content
- How many unique channels did they watch in the first 3 days.
- Did they open the email sent out by AT&T

Exploratory Visualization

By using the describe() function in python, you can quickly get basic stats of each of the above columns I listed above.

```
         day_0_dur     day_1_dur     day_2_dur     day_3_dur           vsf  \
count 83134.000000  83134.000000  83134.000000  83134.000000  83134.000000
mean      0.011022      0.006611      0.015353      0.012722      0.002519
std       0.036959      0.020277      0.044072      0.035506      0.012008
min       0.000000      0.000000      0.000000      0.000000      0.000000
25%       0.000000      0.000000      0.000000      0.000000      0.000000
50%       0.000000      0.000000      0.000000      0.000000      0.000000
75%       0.000000      0.000319      0.003496      0.004188      0.001589
max       1.000000      1.000000      1.000000      1.000000      1.000000

              ebvs  avg_startup_time  rebuffering_ratio   sports_flag  \
count 83134.000000      83134.000000       83134.000000  83134.000000
mean      0.001491       5135.304072           0.007400      0.243138
std       0.007821       4439.249709           0.041674      0.428980
min       0.000000         -3.000000           0.000000      0.000000
25%       0.000000       2051.000000           0.000000      0.000000
50%       0.000192       4878.930000           0.000000      0.000000
75%       0.000862       7404.380000           0.000000      0.000000
max       0.593086     187131.670000           1.000000      1.000000

         news_flag     kids_flag     live_perc      dvr_perc     vod_perc  \
count 83134.000000  83134.000000  83134.000000  83134.000000  83134.00000
mean      0.195383      0.224192      0.394306      0.013572      0.04462
std       0.396498      0.417052      0.466510      0.080100      0.16218
min       0.000000      0.000000      0.000000      0.000000      0.00000
25%       0.000000      0.000000      0.000000      0.000000      0.00000
50%       0.000000      0.000000      0.000000      0.000000      0.00000
75%       0.000000      0.000000      1.000000      0.000000      0.00000
max       1.000000      1.000000      1.000000      1.000000      1.00000

             ch_ct    email_flag         wirls           dtv          uvtv  \
count 83134.000000  83134.000000  83134.000000  83134.000000  83134.000000
mean      0.038732      0.233659      0.256237      0.138968      0.028376
std       0.074147      0.423160      0.436557      0.345916      0.166045
min       0.000000      0.000000      0.000000      0.000000      0.000000
25%       0.000000      0.000000      0.000000      0.000000      0.000000
50%       0.000000      0.000000      0.000000      0.000000      0.000000
75%       0.062500      0.000000      1.000000      0.000000      0.000000
max       0.875000      1.000000      1.000000      1.000000      1.000000
```

```
         day_0_dur     day_1_dur     day_2_dur     day_3_dur           vsf  \
count 83134.000000  83134.000000  83134.000000  83134.000000  83134.000000
mean      0.011022      0.006611      0.015353      0.012722      0.002519
std       0.036959      0.020277      0.044072      0.035506      0.012008
min       0.000000      0.000000      0.000000      0.000000      0.000000
25%       0.000000      0.000000      0.000000      0.000000      0.000000
50%       0.000000      0.000000      0.000000      0.000000      0.000000
75%       0.000000      0.000319      0.003496      0.004188      0.001589
max       1.000000      1.000000      1.000000      1.000000      1.000000

              ebvs  avg_startup_time  rebuffering_ratio   sports_flag  \
count 83134.000000      83134.000000       83134.000000  83134.000000
mean      0.001491       5135.304072           0.007400      0.243138
std       0.007821       4439.249709           0.041674      0.428980
min       0.000000         -3.000000           0.000000      0.000000
25%       0.000000       2051.000000           0.000000      0.000000
50%       0.000192       4878.930000           0.000000      0.000000
75%       0.000862       7404.380000           0.000000      0.000000
max       0.593086     187131.670000           1.000000      1.000000

         news_flag     kids_flag     live_perc      dvr_perc     vod_perc  \
count 83134.000000  83134.000000  83134.000000  83134.000000  83134.00000
mean      0.195383      0.224192      0.394306      0.013572      0.04462
std       0.396498      0.417052      0.466510      0.080100      0.16218
min       0.000000      0.000000      0.000000      0.000000      0.00000
25%       0.000000      0.000000      0.000000      0.000000      0.00000
50%       0.000000      0.000000      0.000000      0.000000      0.00000
75%       0.000000      0.000000      1.000000      0.000000      0.00000
max       1.000000      1.000000      1.000000      1.000000      1.00000

             ch_ct    email_flag         wirls           dtv          uvtv  \
count 83134.000000  83134.000000  83134.000000  83134.000000  83134.000000
mean      0.038732      0.233659      0.256237      0.138968      0.028376
std       0.074147      0.423160      0.436557      0.345916      0.166045
min       0.000000      0.000000      0.000000      0.000000      0.000000
25%       0.000000      0.000000      0.000000      0.000000      0.000000
50%       0.000000      0.000000      0.000000      0.000000      0.000000
75%       0.062500      0.000000      1.000000      0.000000      0.000000
max       0.875000      1.000000      1.000000      1.000000      1.000000
```

```
               uvint          uvvce     Brazilian        Go Big  Gotta Have It  \
count   83134.000000   83134.000000  83134.000000  83134.000000   83134.000000
mean        0.157036       0.032021      0.000217      0.164253       0.137970
std         0.363837       0.176056      0.014713      0.370507       0.344871
min         0.000000       0.000000      0.000000      0.000000       0.000000
25%         0.000000       0.000000      0.000000      0.000000       0.000000
50%         0.000000       0.000000      0.000000      0.000000       0.000000
75%         0.000000       0.000000      0.000000      0.000000       0.000000
max         1.000000       1.000000      1.000000      1.000000       1.000000

       Hispanic - Todo Y Mas    Just Right  Live a Little    Vietnamese  \
count           83134.000000  83134.000000   83134.000000  83134.000000
mean                0.030409      0.234705       0.432001      0.000445
std                 0.171710      0.423817       0.495358      0.021092
min                 0.000000      0.000000       0.000000      0.000000
25%                 0.000000      0.000000       0.000000      0.000000
50%                 0.000000      0.000000       0.000000      0.000000
75%                 0.000000      0.000000       1.000000      0.000000
max                 1.000000      1.000000       1.000000      1.000000

         AT&T RETAIL  AUTHORIZED RETAIL   CALL CENTER       CRICKET  \
count   83134.000000       83134.000000  83134.000000  83134.000000
mean        0.072389           0.023985      0.132052      0.012041
std         0.259133           0.153004      0.338549      0.109069
min         0.000000           0.000000      0.000000      0.000000
25%         0.000000           0.000000      0.000000      0.000000
50%         0.000000           0.000000      0.000000      0.000000
75%         0.000000           0.000000      0.000000      0.000000
max         1.000000           1.000000      1.000000      1.000000

             Cricket  NR/LSP/Indirect eComm/CP  ONLINE & Direct Mail
count   83134.000000              83134.000000          83134.000000
mean        0.000144                  0.014459              0.744509
std         0.012014                  0.119372              0.436140
min         0.000000                  0.000000              0.000000
25%         0.000000                  0.000000              0.000000
50%         0.000000                  0.000000              1.000000
75%         0.000000                  0.000000              1.000000
max         1.000000                  1.000000              1.000000
```

There are 126,425 new accounts that signed up for the DTVNOW 7 day trail. Of which 84,282 ended up converting into a paying customer after the trail period is up. 42,143 canceled the subscription before the free trial was over.

	convert
1	84,282
0	42,143
Grand Total	**126,425**

Here's an example of a customer's complete data:

chnl_prtnr_cust_cd	181031000000000
eop	1
base_pkg_prd_nm	Live a Little
chnl_actvt_chnl_grp	ONLINE & Direct Mail
wirls_hshld_ind	N
dtv_hshld_ind	Y
uvrs_tv_hshld_ind	N
uvrs_int_hshld_ind	N
uvrs_vce_hshld_ind	N
day_0_dur	5011
day_1_dur	45546
day_2_dur	46110
day_3_dur	39633
vsf	0
ebvs	8
avg_startup_time	8281.75
rebuffering_ratio	0
sports_flag	1
news_flag	0
kids_flag	1
live_perc	0.999611152
dvr_perc	0.000388848
vod_perc	0
ch_ct	8
email_flag	0
aud_flag	0
devices	3

If the feature is a classification type (such as the type of base package), I used the dummy function in pandas to one hot encode the data into 1 and 0 for each type. If the data type is numeric (such as the duration or the number of channels) I min-max scaled the data in Python.

Also, since there were significantly more customers that stayed on the product after 7 days compared to the customers that churned out, I random sampled from the customers that stayed to make the data set even.

Algorithms and Techniques

I first tried the SVM algorithm. SVM algorithm tries to maximize the margin between the converted paying customers and the canceled customers by drawing a line or a plane. However, the SVM model didn't complete after over an hour of computation. Because my data set has over 25 features (columns) it takes too long for the algorithm to compute the divider for that many dimensions. It's just not practical in my use case.

Next I tried the Naïve Bayes algorithm. Naïve bayes calculates probability independently by each feature. The benefit is that this algorithm is extremely fast. However, this model only returned about 0.61 accuracy.

Finally, I tried the Random Forest algorithm. Random forest is a tree based algorithms where it takes different combination of features to build hundreds of trees and builds the best tree at the end. Random Forest algorithm is very fast and is pretty accurate. It returned decent f1 score without taking a long time to calculate, so I decided to enhance the Random Forest algorithm.

Benchmark

Here is an example of an academic paper that did something very similar except it is in the fitness membership business:

https://umu.diva-portal.org/smash/get/diva2:1161821/FULLTEXT01.pdf

The above study has a AUC score of around 80%. Given that a totally random guess would yield 50% accuracy, I would like to achieve something very close to the fitness membership study.

Given the above study has a ROC score, to be able to compare apples to apples to my analysis, I will provide the ROC score in the conclusion section.

III. Methodology

Feature transformation

For columns that return a Y or N (for example, do they have a wireless product with AT&T or not), I wrote a function in Python to transform them into 1 and 0s.

For columns that return text (for example, sales acquisition channel such as AT&T retail store, call center, etc.) I used the dummy function in Python to create new columns with 1 and 0s.

For columns that return a number (for example duration), I min-max scaled so every value will be between 0 and 1.

Lastly, I used the dropna function to drop any rows with null values, question marks, etc.

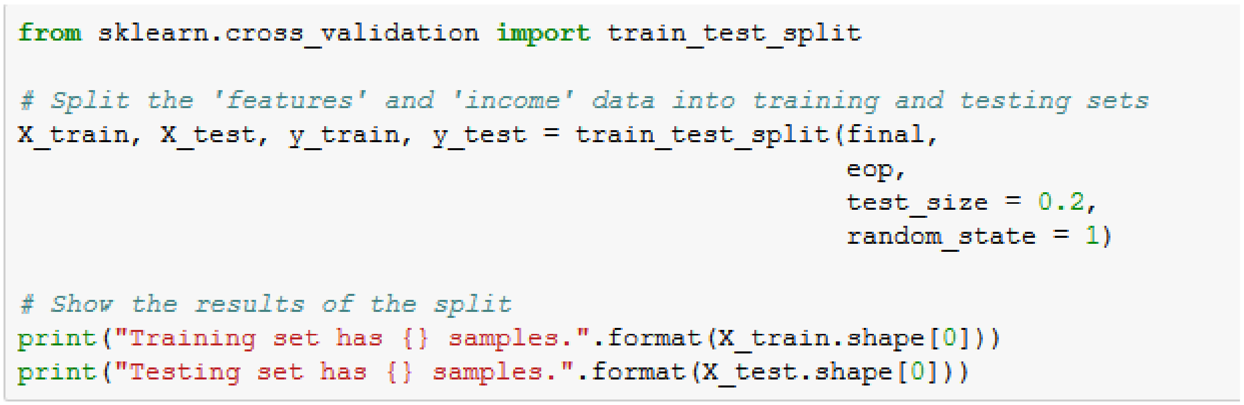
Before I fed the data into the predictive model, I balanced the data so that the final data set has equal number of accounts that turned into paying customers and customers that canceled. This is because about 2/3 of customers turn into paying customers, so without balancing, the model can just say everyone with turn into a paying customer and have 67% accuracy.

For this I understand there's both sides to the argument: keep unbalanced data set or balance data set. And data will always be inbalanaced, whether is spam/not spam email or healthy/sick patients.

https://datascience.stackexchange.com/questions/810/should-i-go-for-a-balanced-dataset-or-a-representative-dataset

Implementation

I used train_test_split to split the data.

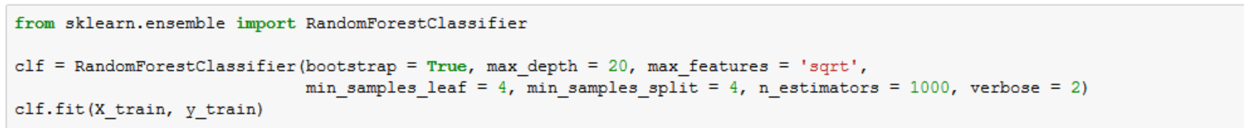


```
from sklearn.cross_validation import train_test_split

# Split the 'features' and 'income' data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(final,
                                                    eop,
                                                    test_size = 0.2,
                                                    random_state = 1)

# Show the results of the split
print("Training set has {} samples.".format(X_train.shape[0]))
print("Testing set has {} samples.".format(X_test.shape[0]))
```

And the fed into the random forest classifier.



```
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(bootstrap = True, max_depth = 20, max_features = 'sqrt',
                             min_samples_leaf = 4, min_samples_split = 4, n_estimators = 1000, verbose = 2)
clf.fit(X_train, y_train)
```

Refinement

I made a lot of refinements to the model after my initial model.

1. I changed the test train split from 25% to other values.

2. I looked for the best hyperparameter combination using RandomizedSearchCV.

The hyperparameters that I tuned are as follows:

'bootstrap': [True, False],

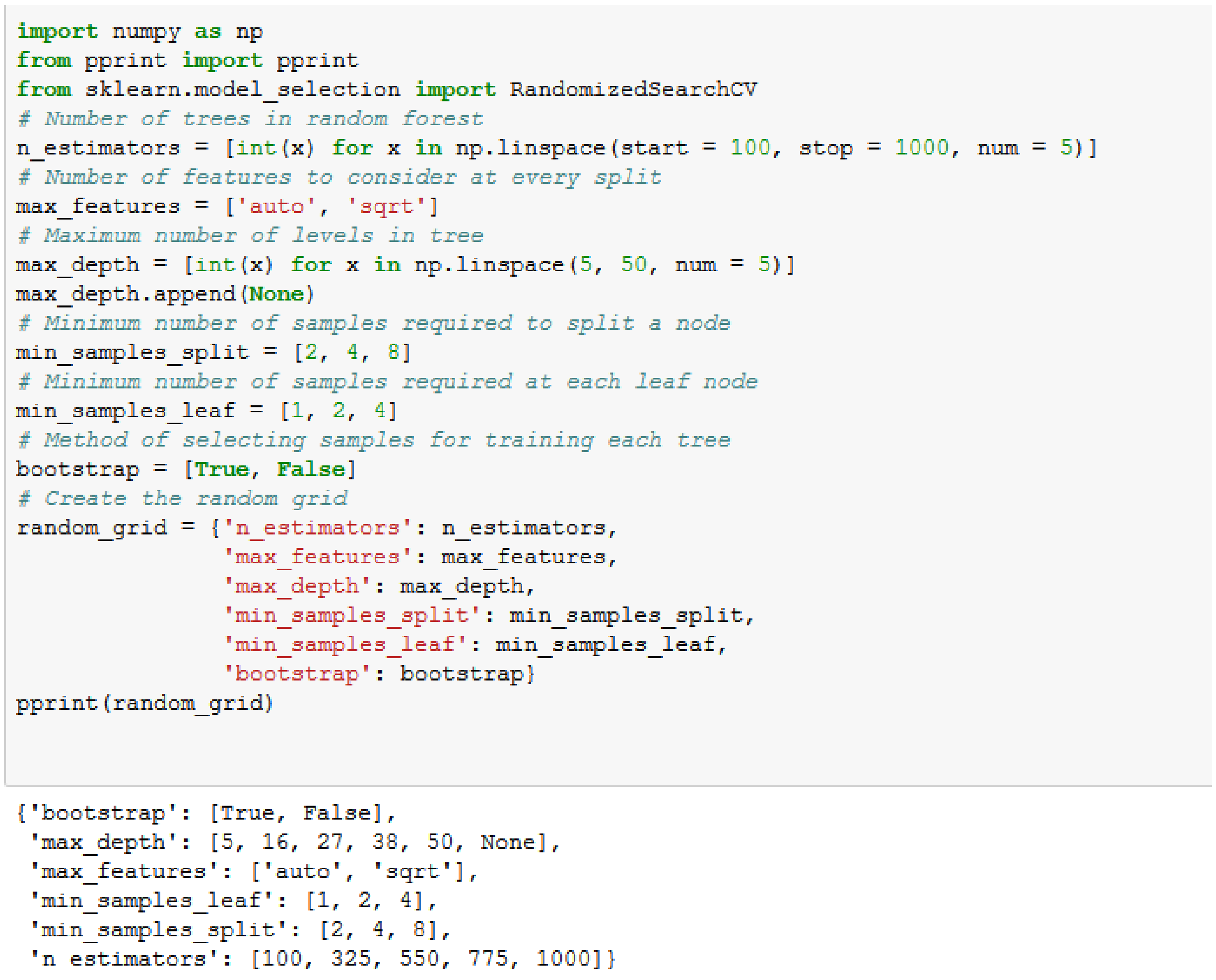
 'max_depth': [5, 16, 27, 38, 50, None],

 'max_features': ['auto', 'sqrt'],

'min_samples_leaf': [1, 2, 4],

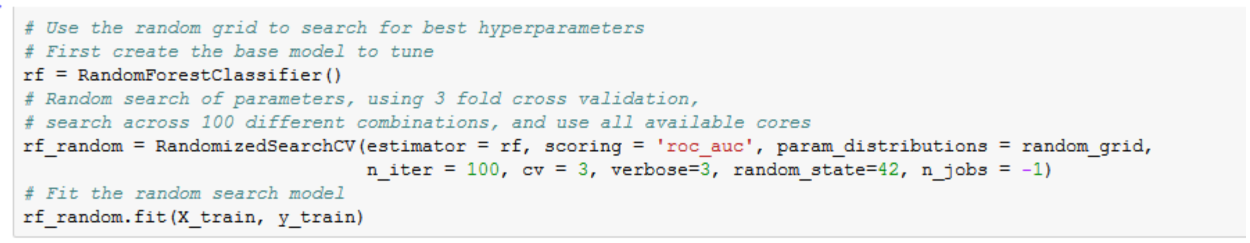
'min_samples_split': [2, 4, 8],

'n_estimators': [100, 325, 550, 775, 1000]

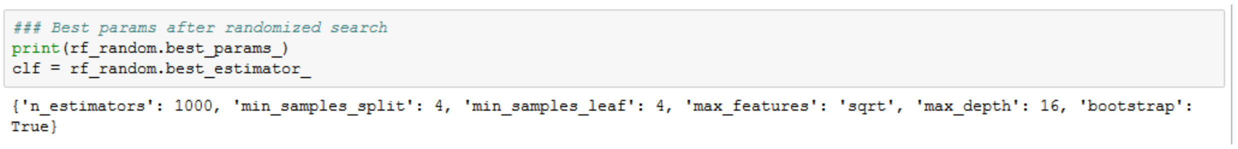


```
import numpy as np
from pprint import pprint
from sklearn.model_selection import RandomizedSearchCV
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1000, num = 5)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 50, num = 5)]
max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 4, 8]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
pprint(random_grid)
```

```
{'bootstrap': [True, False],
 'max_depth': [5, 16, 27, 38, 50, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 4, 8],
 'n_estimators': [100, 325, 550, 775, 1000]}
```



```
# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestClassifier()
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available cores
rf_random = RandomizedSearchCV(estimator = rf, scoring = 'roc_auc', param_distributions = random_grid,
                               n_iter = 100, cv = 3, verbose=3, random_state=42, n_jobs = -1)
# Fit the random search model
rf_random.fit(X_train, y_train)
```



```
### Best params after randomized search
print(rf_random.best_params_)
clf = rf_random.best_estimator_
```

```
{'n_estimators': 1000, 'min_samples_split': 4, 'min_samples_leaf': 4, 'max_features': 'sqrt', 'max_depth': 16, 'bootstrap':
True}
```

My conclusion is that if you're doing a shallow learning algorithm like a classification problem, hyperparameter tuning might help your scores maybe a few percentage points, but won't make it significantly better. Whereas CNNs in deep learning, hyperparameters are super sensitive and is worth spending time looking for the best combinations.

3. I added more features (columns) such as number of devices used during the first 3 days. I also tried consolidating columns to see if that will improve the f1 score. For example, instead of having a column for day 1 duration, day 2 duration, etc. I created a column that has the sum of duration for the first 3 days.

4. After talking to a colleague, I tried the GBM algorithm. His argument is that Random Forest is a majority voting of different trees, whereas GBM makes incremental improvements, thus, will be a slightly better algorithm.

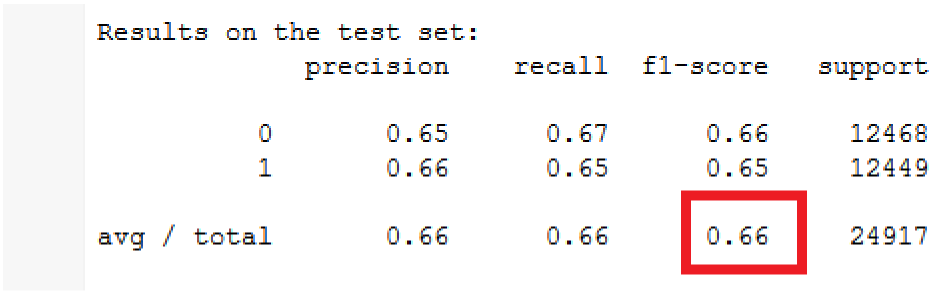
5. I even did hyperparameter tuning on my GBM model...

Honestly, none of the above refinements made my f1 score much better.

IV. Results

Model Evaluation and Validation

The f1 score of the best model was around 0.66, which wasn't as high as I would have liked.



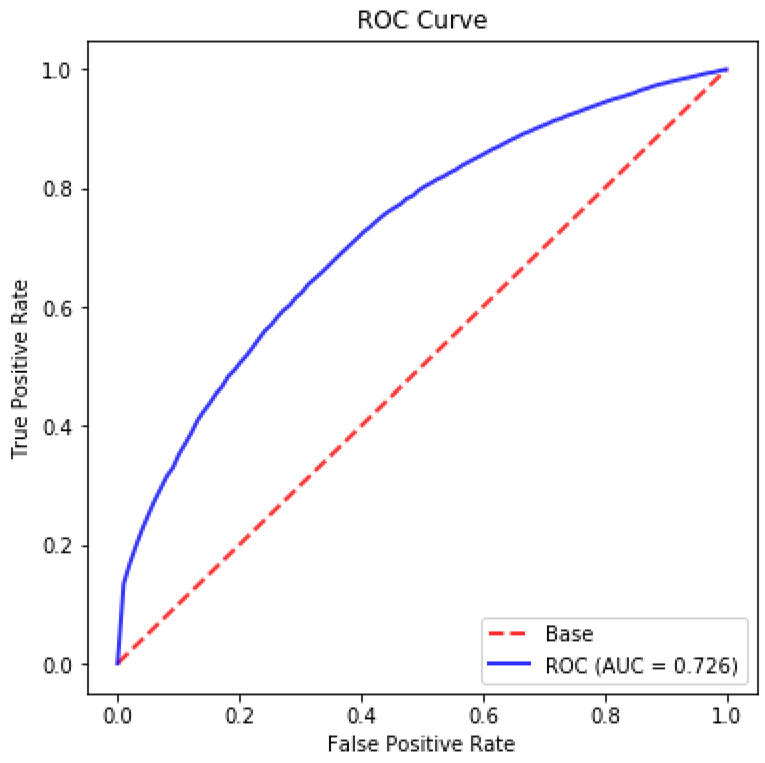
```
Results on the test set:
             precision    recall  f1-score   support

          0       0.65      0.67      0.66     12468
          1       0.66      0.65      0.65     12449

avg / total       0.66      0.66      0.66     24917
```

Like I mentioned in the previous chapter, I tried a lot of different things to try to improve the model. I even tried using a different month's data (September 2018 vs October 2018) to make sure there was no weird seasonality. I just didn't see the improvement that I would have liked.

As for scoring, neither the f1 score or the AUC were as high as my expectation, or in comparison to the fitness industry churn prediction that I cited above. My AUC value is 0.726 whereas the fitness study easily scored over 0.8 with boosted decision tree models.

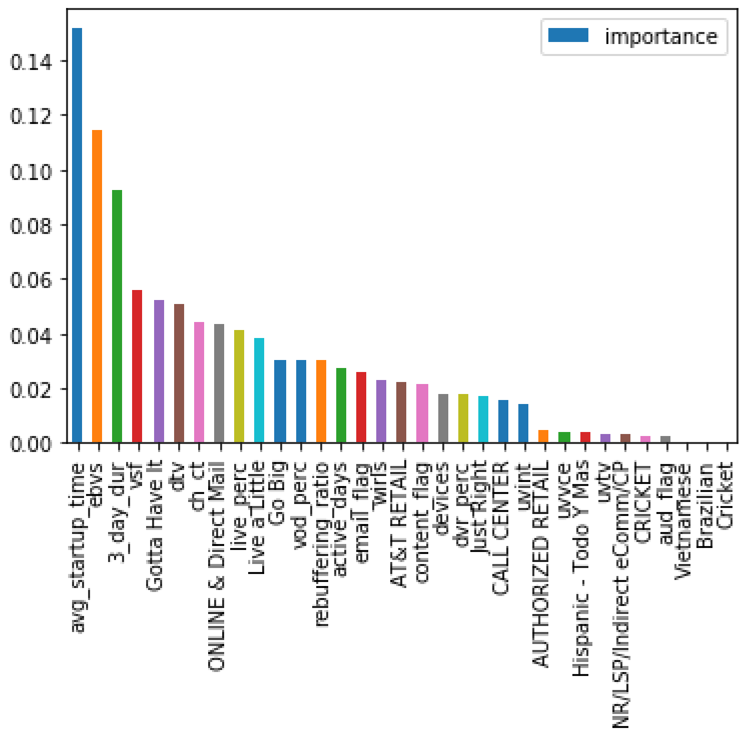


V. Conclusion

<u>Reflection</u>

To summarize my project, it was to predict if a customer would end up staying on a subscription after a free 7 day trial. I used multiple features including product performance, acquisition channel, and product type to solve the problem above.

I used four different algorithms including SVM, Naïve Bayes, Random Forest, and GBM. I liked the tree based models for classification problems, as it gives me a good balance performance and speed. Also, I like the fact that you can see which features are the most important.



Although I didn't get the high scores that I wanted, the feature importance graph tells us what features are important when a 7 day trial customer converts into a paying customer. Thing like average start up time, errors, the duration of the first few days are very important. Now that we know this, we can craft marketing efforts to customers that have low values for these features (meaning they are likely to NOT convert into a paying customer) and see if we can turn them into a paying customer. Although the f1 score was not as high as I liked, I gained very valuable insights from the project.

Also, the top features all make sense to me. If the average startup time of the videos are high, then I would be frustrated, and would not value the product enough to pay monthly. Same goes with the number of errors. If I'm not watching a lot of videos in the first three days, I wouldn't keep the product.

Using the information above, I can take this to the marketing team and have them create app pop ups to entice customers to watch more, or send emails to customers that don't have much viewership in the first three days of free trial. That way, they might see the value of the product and end up converting to a paying customer.

Looking back on the project, the most interesting part was trying the various algorithms. It is very interesting to see the speed of different algorithms. Being in industry, I don't necessarily need the most accurate model, especially if it comes at a cost of time and resources. Rather, I would value a model that is pretty accurate, and also runs pretty fast.

The most difficult part of the project was thinking through the features to feed the model and actually obtaining the data. For feature selection, you need domain knowledge. You need to understand the business problem. I brainstormed a lot with my colleagues and my manager to get their feedback on what other good feature sets that I could feed into the model. Then I had to ask around to see where the data resides. And often times they sit in different data ecosystems (ex. Hadoop vs Vertica vs Teradata). Some customer account information is encrypted for security reasons, so I had to decrypt, and join data sources. It was a lot of work.

The other difficult part was refining the model, only to see no improvement. As I documented in the Methodology section, I spent A LOT of time refining the model, trying different algorithms, adding features (each time spending more time on data prep!).

Improvement

I can't really think of ways to improve the model at this stage. My manager suggested in add customer chat data (when they have issues with the product or billing, they chat after logging into their account) but that data is not validated yet so I can't feed that data into the model at the moment. This can be an ongoing process. As we get new data to feed, I can periodically update the model to see the scores improve.

Also this project can be scaled to other product that AT&T has. Right now, I'm looking at the internet based video entertainment product, but I can look at wireless retention, broadband, landline, etc.

Bibliography

Shung, K. P. (2018). *Accuracy, Precision, Recall or F1?* Retrieved from https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9