CAPSTONE PROJECT: STARBUCKS

Andreas Yiannakou

MACHINE LEARNING ENGINEER NANODEGREE

I. Definition

Project Overview

This project is the Capstone project of the Udacity Machine Learning Engineer Nanodegree. The data set contain simulated data that mimics customer behaviour on the Starbucks rewards mobile app. The data has been simplified so as to only contain one product. Once every few days, Starbucks sends out an offer to users of the mobile app. An offer can be merely an advertisement for a drink or an actual offer such as a discount or BOGO (buy one get one free). Some users might not receive any offer during certain weeks. Not all users receive the same offer, and that is the challenge to solve with this data set.

Problem Statement

The aim of this project is to create a model looking at customers who had either completed and offer without viewing or before they had viewed the offer and customers who did not complete the offer.

Once a model to predict whether a user would naturally have completed an offer has been created the evaluation can be extended and used to predict whether or not the users who had viewed and completed the offer would have done so naturally anyway and calculate whether the offers generated additional revenue.

Metrics

Due to the level of imbalance in the classes, choosing accuracy as the scoring metric would have led to the model just predicting all labels as ‘0’, achieving a high accuracy but offering no predictive power. Given the baseline model was to predict that all labels are ‘0’, it ruled out the f1-score as a suitable metric since the f1-score for the baseline model would have been 0.

This is why I eventually settled on using the area under the curve of a receiver operating characteristic (ROC) curve as the metric, this gave the baseline model a score of 0.5, a suitable benchmark for the models to attempt to improve upon. Whilst area under the curve is the primary metric and the scoring metric when training the model and selecting the best estimator, I will also look at accuracy and the confusion matric in order to gain a better understanding on the models output.

II. Analysis

Data Exploration

The data is contained in three files:

- portfolio.json - containing offer ids and meta data about each offer (duration, type, etc.)
- profile.json - demographic data for each customer
- transcript.json - records for transactions, offers received, offers viewed, and offers completed

Portfolio (Promotional Data)

This file describes the characteristics of each offer, including its duration and the amount a customer needs to spend to complete it (difficulty).

- id (string) - offer id
- offer_type (string) - type of offer ie BOGO, discount, informational
  - BOGO (Buy One Get One Free)
  - Discount (Spend x amount get y off)
  - Informational (Advert for product)
- difficulty (int) - minimum required spend to complete an offer
- reward (int) - reward given for completing an offer
- duration (int) – how many days the offer runs for
- channels (list of strings) – how the offer is sent to the customer, this is a list of different mediums e.g. web, mobile

Informational offers do not have rewards and therefore will be removed from this project as the second phase of the problem once I have built a classifier is to look at if rewards were unnecessarily given away to customers who would have completed the offers regardless.

Profile (Customer Data)

This file contains customer demographic data including their age, gender, income, and when they created an account on the Starbucks rewards mobile application.

- age (int) - age of the customer
- became_member_on (int) - date when customer created an app account
- gender (str) - gender of the customer (note some entries contain 'O' for other rather than M or F)
- id (str) - customer id
- income (float) - customer's income
- Some users have missing information
  - They have no gender; no income and their age are set to 118
- 3 genders – {Male, Female, Other}
- Age follows a somewhat normal distribution, discussed further later on
- The income is fairly clean after removing the missing information and since it is continuous it will be scaled but there is no requirement to bin the data

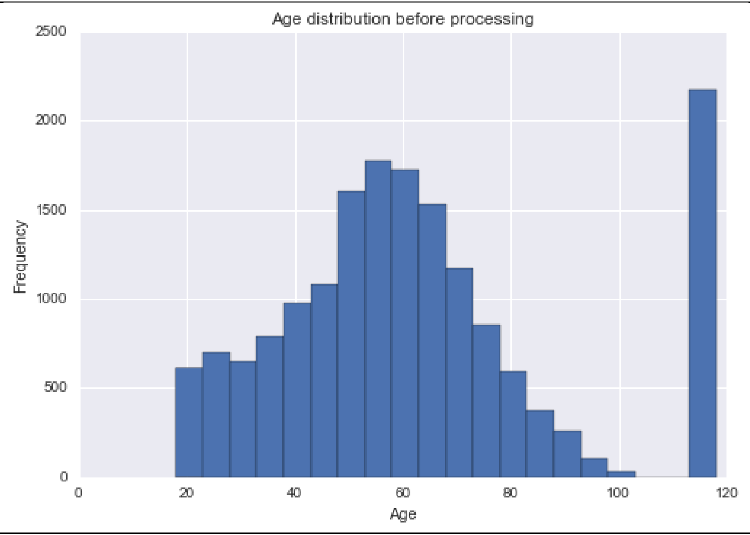
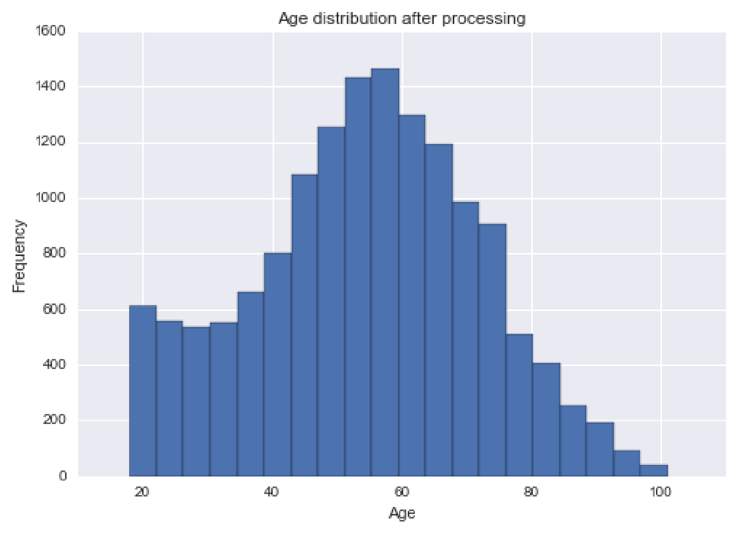
Transcript (Offers & Transaction Data)

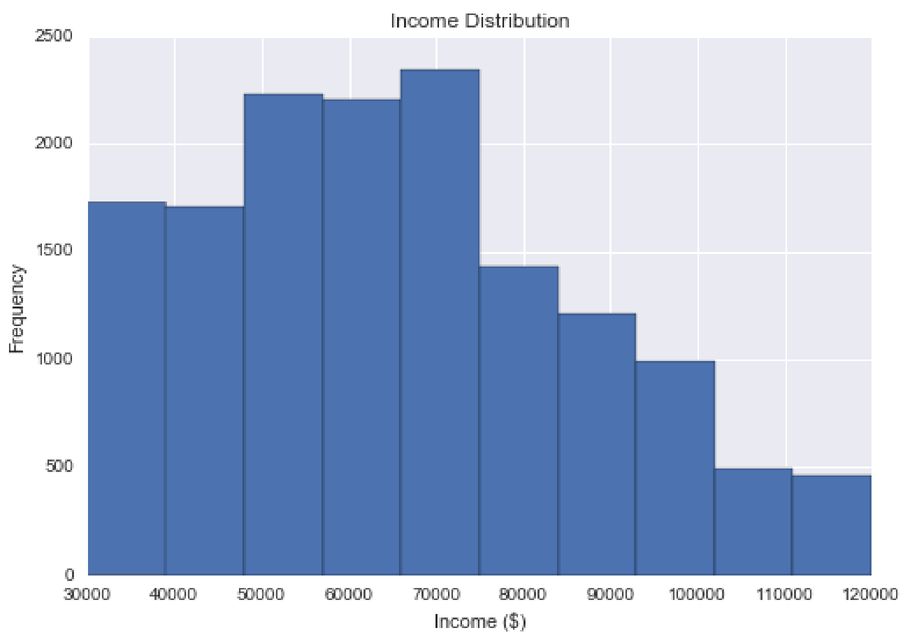
The third file describes customer purchases and when they received, viewed, and completed an offer. This is what is usually defined as tall data as there data is normalised therefore there are multiple records for the same customer and offer.

- event (str) – record description (ie transaction, offer received, offer viewed, etc.)
- person (str) – customer id
- time (int) – time in hours. The data begins at time t=0
- value – (dict of strings) – either an offer id or transaction amount depending on the record

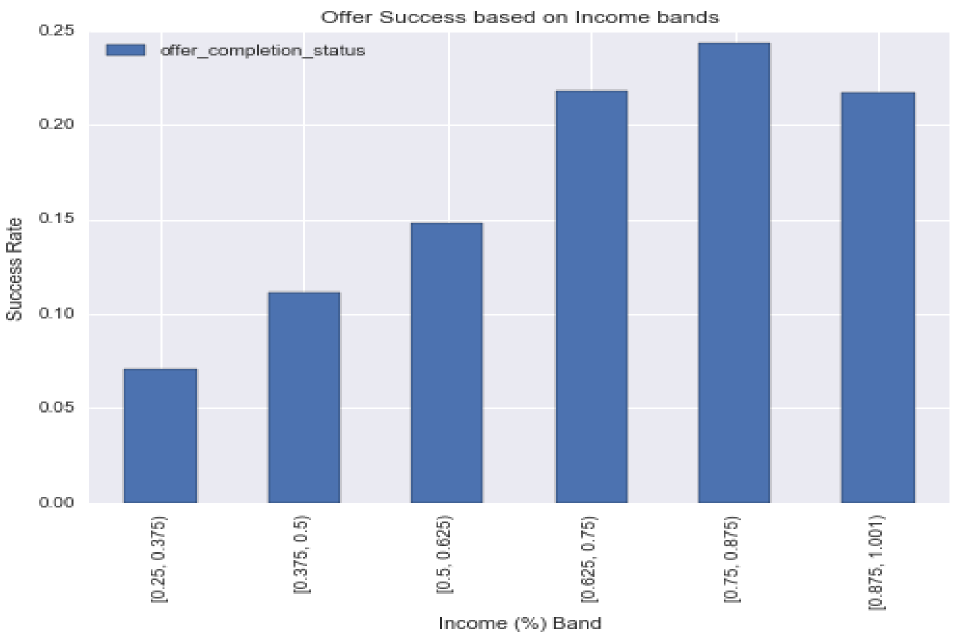
An offer is only successful when a customer both views an offer and meets or exceeds its difficulty within the offer's duration. There are cases when the customer meets the offer condition before they view the offer, this is what the model I am building will try to classify in order to extend the problem and look into whether people who did view the offer and then meet the criteria would have done so anyway.

Exploratory Visualization

Before		The age data in the profile data set has a normal distribution, which peaks at around age 60, with the exception of large number of 118-year-old customers. I imagine that the data is from 2018 and the customers date of birth has been put in the system as 1st January 1900 which a lot of systems default to. When processing the data, I removed customers who information such as income missing and all of the 118-year-olds were all removed when I did this showing that the data was indeed erroneous.
After		



Income follows a slight normal distribution although it is quite difficult to see. The number of customers peaks between $50,000 & $70,000 and then begins to decrease afterwards. Whilst the number of customers is fairly uniform for incomes up until %50,000 and tails off quite a lot over $100,000.



In this graph income is grouped in six different bands based where the income value is income/$120,000 (the max income in the data). The income bands are then compared against their success rates, whilst we can see the success rates in general are low there is clearly a trend towards customers with higher incomes being more likely to complete the offers. With those with an income of between $90,000 & $105,000 having a success rate of almost 25% almost 10% higher than those with income between $60,000 & $75,000.

Algorithms and Techniques

Given this is a fairly simple binary classification problem there are a lot of different algorithms which can be used to try and solve this problem, I have used 3 which generally perform well but also have less of a requirement for feature selection due to the way the models are trained and optimised. The algorithms also minimise the requirement for data transformation as none of them are reliant on all features needed to be between 0 and 1, despite that the majority of the features are between this range regardless.

Gradient Boosted Classifier

The Gradient Boosted classifier is an ensemble method which builds weak classifiers in this case small decision trees sequentially with the next decision tree trying to predict the error of the previous set of trees. Gradient Boosted model have relative success over the years and generally are strong classifiers, they do however require a lot of tuning in order to optimise their results. As the name suggests they use boosting, where the training data is a sample of the overall training data with replacement, meaning the models are not always fed the way data.

- n_estimators (number of trees built)
    - 100, 150, 200
- subsample (number of samples used for each model, 0.6=60%)
    - 0.6, 0.8, 1
- max_features (how many features are compared at each split)
    - None (looks at all features), auto (square root of number of features)
- Loss (loss function the model is trying to optimise)
    - deviance, exponential
        - ‘deviance’ refers to deviance (= logistic regression) for classification with probabilistic outputs. For loss ‘exponential’ gradient boosting recovers the AdaBoost algorithm.
- random_state set to 1 to ensure the results are reproducible

Random Forest Classifier

Random Forests much like Gradient Boosted models are an ensemble method which uses decision trees to create predictions, Random Forests also use boosting when training the model. Random Forests however build full size decision trees and can be trained simultaneously as they are all trying to predict the end target rather than the previous trees error. Random Forests also restrict the features which a node can split on, in order to create decorrelated trees to improve results.

- max_depth (used to limit the number of nodes in each trees)
    - 5, 8, 10
- n_estimators (number of trees built)
    - 100, 150, 200
- max_features (how many features are compared at each split)
    - sqrt (square root of number of features), log2 (log2 (number of features)),
- max_subsample set to 0.6 (60%)
- random_state set to 1 to ensure the results are reproducible

Logistic Regression

Logistic Regression is the simplest of the models, it is a linear model which calculate the log odds of an event happening in this scenario the odds on a customer successfully completing an offer. It gets is name from using the logistic (also known as sigmoid) function.

- penalty (method used to regularise the model)
    - l1, l2, elasticnet (this is a combination of l1 & l2)
- C (defines the strength of regularisation, the smaller the most the model is regularised)
    - 0.5, 1, 1.5
- solver (algorithm used for optimisation of the results)
    - liblinear, saga, lbfgs
- random_state set to 1 to ensure the results are reproducible

Summary

All the models work in slightly different ways and different models will both require different parameters in order to fit the data but will also require different amounts of tuning via hyperparameter, ensemble tree-based models whilst powerful often require a lot of tuning in order to optimise the results.

A grid search parameter will be used in order to explore all the different combinations of the parameters that are feasible (e.g. elasticnet only works for the saga solver) and return the best estimator, the grid search function will use cross validation to assess the models in order to decrease the risk of overfitting.

Benchmark

The benchmark model proved somewhat difficult in this approach given the level of imbalance in the classes, given the key metric I was using to validate the models, I ended up using a model which predicted '0' that no user would have completed the offer for each record.

There are a couple of reasons this is significant, the first is that the accuracy is very high as the number of people who do not complete the offer is high, whilst the area under the curve is always 0.5, although this is true even if the naïve model predicted that every customer completed their offers. Whilst it is not a fantastic benchmark numerically it does provide a reasonable value to work towards and surpass.

The second reason is that in the second phase of the project where we look at whether a customer who did view the offer and complete it would have predicted that the offers are 100% effective and that customers would not have completed the offer had they not viewed the offer. Which feels more aligned to what we would aim for with a naïve hypothesis, whilst the model aims to reject this and try to validate how many customers would have completed the offer anyway.

III. Methodology

Data Preprocessing

Key Assumptions

- Ignore informational promotions, since there is no reward, they would not affect the results for the second phase of analysis in which look at whether rewards negatively affected overall profit.
- A customer can only have one active promotion at a time, if a customer receives a second promotion within before the duration of the previous had been completed then the previous promotion is finished early and the new promotion becomes active.

Cleaning Data

Each of the json files was processed and the data was cleaned, in the case of the transaction data there was more of a transformation of the data in order to get all the data required for a customer and their offer in a single row.

Portfolio (Promotional Data)

- Create a column for each channel with a flag to say whether the channel is used in the promotion
- Create dummy variables for the offer types with a flag to show whether the promotion has that offer type
- Rename the ‘id’ and ‘duration’ columns to ‘offer_id’ and ‘duration_days’ to be more descriptive
- Drop the email channel flag as all promotions are distributed via email and therefore all the values in the column are the same

Profile (Customer Data)

- Remove NAs from the data
    - This also removed the profiles with an age of 118
- Create bins to group the age data
    - Into the following groups {‘Unknown’,'18 - 30','31-50', '51-70','70+'}
- Create dummy variables for the age groupings
    - Remove the ‘Unknown’ & ‘70+’ flags
- Create dummy variables for gender
    - Rename the variables from [‘F’, ‘M’, ‘O’] to [‘Female’, ‘Male’, ‘Other’]
    - Remove the ‘Other’ gender flag
- Scale the income values to be between 0 and 1

Transcript (Offers & Transaction Data)

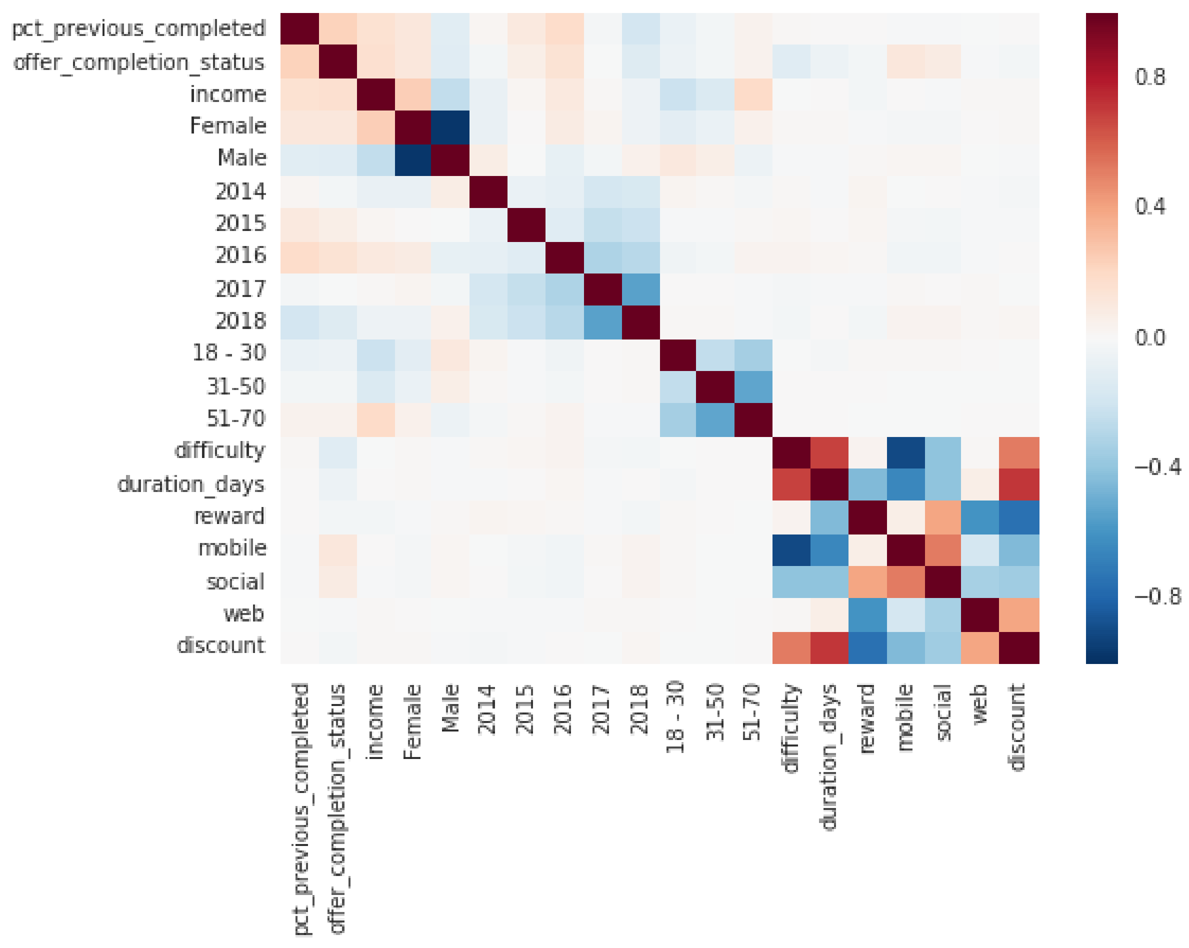
- Remove customers who are not in the profile data
- Remove offers which are not in the portfolio data
- Pivot the offers information to have a flag for whether the user viewed and completed the offer
- Set the end time of an offer to either the start date + duration or the start time of the next offer

Implementation & Refinement

Removing one of the dummy variables

When turning categorical data into dummy variables, often one of the dummy variables is removed since the information is held by the combination of all the other variables since the one dummy variable always equals 1 – sum(all other dummy variables created from the same original feature). Whilst it is not mandatory to remove then especially given the ensemble methods being used, it something I tend to do anyway.

Correlation



Correlation was explored to check whether there were any features heavily correlated with each other however with the exception of the portfolio (promotions) data the other metrics were not very correlated and nothing was particularly correlated with the target variable.

Balancing Classes

There was class imbalance between a customer completing and not completing an offer, I did initially balance the classes using hyperparameter optimisation as sklearn allows you set this as an option however the models became far slower to build, I then looking at over and under sampling. The function I created has a flag which allows you to select if you want to over or under sample however, I decided to oversample as this allowed me to have around 5 times more examples in the training set then under sampling. The samples were taken from the original data in the training set and sampled with replacement, a random state was also used to ensure the results were reproducible.

Hyperparameter Optimisation

This has been touched upon in the previous section, I used hyperparameter optimisation to help tune the parameters for each of the models. The best estimator for each model was then used to create predictions on the test set. During the building process, I did experiment with some parameters which are not in included in the final grid search e.g. no maximum depth or subset of features for the random forest, however this led to the model overfitting and therefore I removed the options.

The models are sklearn models which meant that integrated seamlessly with the GridSearchCV function allowing me to keep a consistent approach to training the models and also accessing the outputs of the models.

IV. Results

Model Evaluation and Validation

Best Estimator Parameters

Below are the parameters found by the hyperparameter optimisation to optimise the results of the cross validation for the training data.

Gradient Boosted Classifier

- n_estimators: 200
- subsample: 0.6 (60%)
- max_features: None (All features)
- Loss: deviance

Random Forest Classifier

- max_depth: 10
- n_estimators: 200
- max_features: sqrt (square root of number of features)

Logistic Regression

- penalty: l2 (Euclidean)
- C: 1
- solver: lbfgs

We can see from the results of the grid search that the parameters for Gradient Boosting and the Random Forest have some similarities, both models use 200 trees in order to achieve the best results and both use 60% of the training data size (not 60% of the training data, as data points are sampled with replacement).

Training Results

Below are the results on the training data, the baseline was tested against the whole training data at once whilst the models were evaluated using 5-fold cross validation.

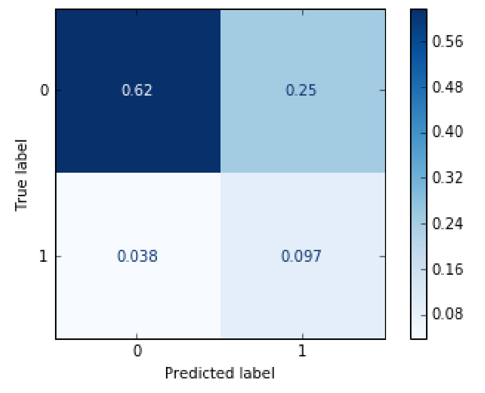
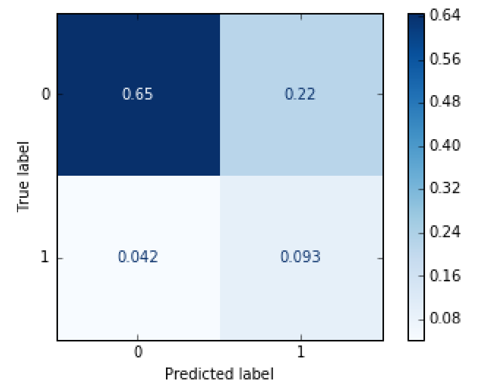
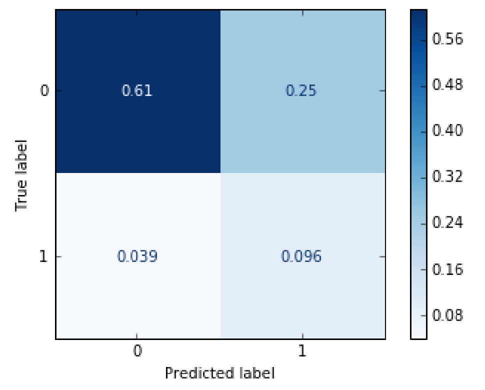
Model	Area Under Curve (AUC)	Accuracy Score
Baseline	0.500	50.0%
Gradient Boosted	0.733	73.3%
Random Forest	**0.800**	80.0%
Logistic Regression	0.707	70.7%

We can see that all the models perform above the baselines and interestingly all the models have the same area under curve and accuracy scores. The Random Forest performs the best out of the models by quite a distance with an area under the curve of 08 with the next best model being the other ensemble method the Gradient Boosted classifier which had a score of 0.733.

Testing Results

Model	Area Under Curve (AUC)	Accuracy Score
Baseline	0.500	**86.4%**
Gradient Boosted	**0.717**	71.6%
Random Forest	**0.717**	74.0%
Logistic Regression	0.711	71.1%

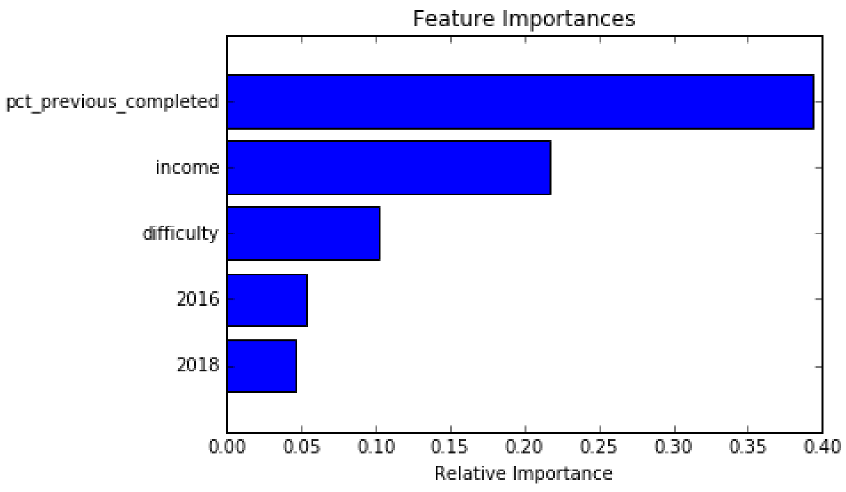
Whilst all the models were eclipsed in terms of accuracy by the baseline model, they were all well above the baseline with regards to the key scoring metric AUC with both the ensemble methods having a scorer of 0.717 and the Logistic Regression, the only model to increase its AUC, close behind with a score of 0.711. The Random Forest had the best accuracy of the 3 models being tested again with a testing accuracy of 74% although that is a whole 6% lower than the training accuracy suggesting the model is slightly overfitting on the training data. Given the Logistic Regression model performed better on the testing set, the model appears to have regularised well.

Gradient Boosted Classifier		Whilst the Gradient Boosted classifier had a similar AUC to the Random Forest, it had a lower accuracy this is because the model classifies label '1' better than the other models at, correctly classifying roughly 72% however due to this being the minority class it doesn't help to increase the overall accuracy. Since it has a far lower accuracy for classifying class '0' with an accuracy of just 64% (62/87).
Random Forest Classifier		The Random Forest has the best accuracy of the three models and we can see from the 3 confusion matrices that this is mainly due to it classifying 65/87 of the records labelled '0' correctly which is higher than both the other models. Whilst it classifies less records where the customer has successfully completed the offer than the other two models it still comes out with a greater accuracy.
Logistic Regression		Logistic Regression has the worst accuracy of the 3 models, it classifies the least amount of records labelled '0', however it performs better when classifying the '1' labels.

Feature Importance

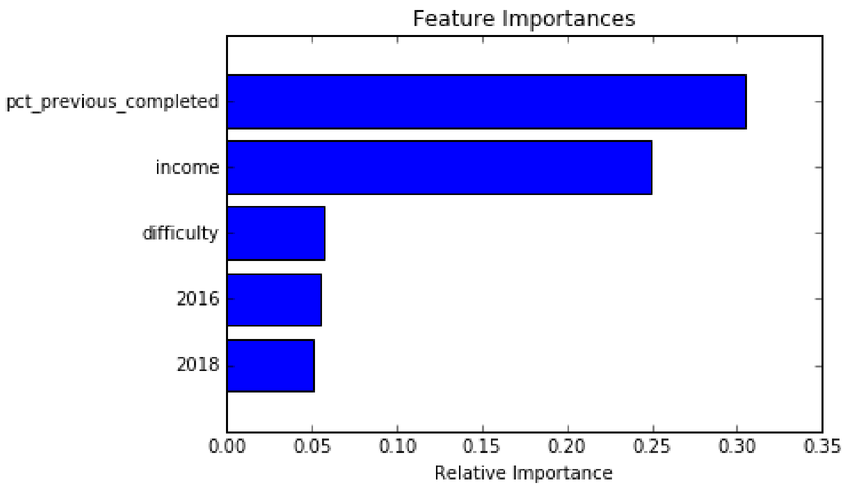
The graphs in this section show the top 5 features for the models and their relative importance.

Gradient Boosted Classifier



We can see from the graph that the % of times the customer had completed an offer previously stands out as the key metric in this model, the next two features are income and difficulty which almost half in importance each time.

Random Forest Classifier



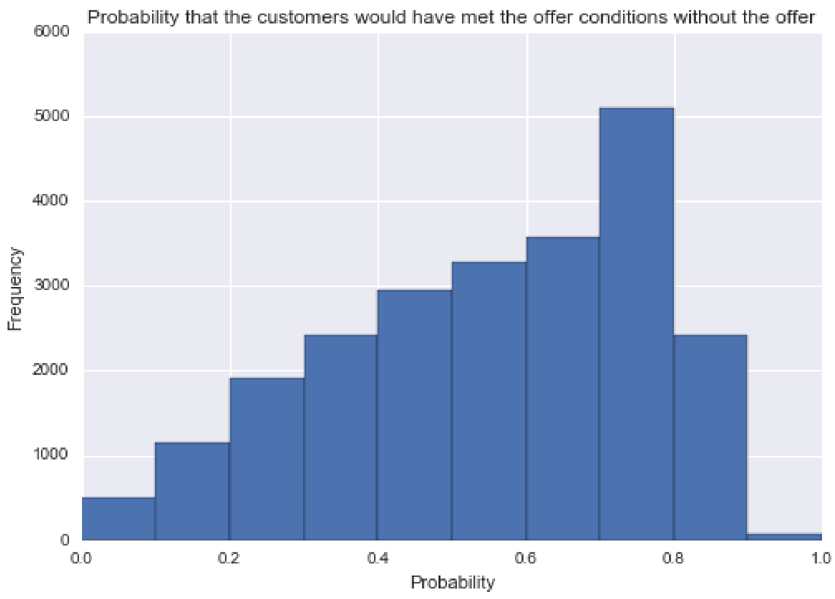
We can see here that the top five features are the same as the Gradient Boosted model however the relative importance distributions quite a bit. Once again, the feature created earlier which calculates the % of times the customer had completed an offer previously tops the relative importance however this time is more closely followed by income with difficulty be much closer to the remaining features.

The top features intuitively make sense, the customer's previous behaviour was always likely to be a good indicator of their future behaviour whilst their level on income will be a good measure of how much they are likely to spend and finally difficulty indicates how easy an offer is to complete. I think the interest part comes when you think about what is missing in the top 5, for instance the level reward and age group flags for the customers suggesting age is not a key metric for prediction.

Business Outcome (Phase Two)

In this section I will look use the best model from the previous section, in this case the Random Forest classifier to predict whether the customers who we know viewed the offer and then proceeded to complete the offer would have done so anyway. This allows us to then look at a couple of financial outcomes, first using these predictions we can look at the amount of "additional" revenue that the offers brought in, this can then be offset by the additional cost of rewarding members who would have spent the money anyway. It is worth highlighting that this analysis labours under the simplified assumption that the value of the transactions does not increase in the case of them seeing the offer and that the customer would not have spent any money if they did not complete the offer. One overestimates revenue gained and the other underestimates it therefore in a large enough sample that would seek to balance itself out.

Model Predictions



Here we can see the number of customers likely to meet the offer conditions anyway steadily increases until a probability of 0.8 where it begins to sharply decline with barely any customers having a likelihood of naturally meeting the offer conditions of 90+%.

The Metrics

$$\textit{Net Increase in Revenue} = \sum((1-\mathrm{c}) * (\mathrm{difficulty} - \mathrm{reward} * \mathrm{r}) - \sum(\mathrm{c} * \ \mathrm{reward} * \mathrm{r})$$

where c *is the class label predicted by the model,* $1 =$ *customer would have naturally completed the offer and r is the real cost of offering a customer a monetary value e.g. £5 voucher equates to £2.50 costs for Starbucks*

$$\textit{Net Expected Increase in Revenue} = \sum(1-p_1) * (\mathrm{difficulty} - \mathrm{reward} * (1-\mathrm{m})) - \sum(p_1 * \ \mathrm{reward} * (1-\mathrm{m}))$$

where p_1 *is the probability of the customer completing the offer naturally and r is the real cost of offering a customer a monetary value e.g. £5 voucher equates to £2.50 costs for Starbucks*

First of all, it is worth highlighting the model predicts that 61.8% of customers would have completed the offer naturally, whilst the average probability of completing the offer was lower at 55.3%.

Whilst the equations can be further simplified, I have left them the way they are as I plan to calculate the summations separately as one is the increase in revenue and the other is the increase in costs. As I didn’t know how much a reward costs Starbucks in reality since they are vouchers for customers to buy additional products, I have used different ratios to analyse the revenues in more detail.

Actual cost of a reward	Net Increase in Revenue	Net Expected Increase in Revenue
25%	\$62,196.75	\$71,804.88
50%	\$33,086.50	\$42,694.63
75%	\$3,976.25	\$13,584.38

We can see from the table that in general the offers performed well and created revenue for Starbucks on the assumption that the cost of a reward was at most 75% of its face value. One interesting thing that came out of these calculations is that although the mean probability of someone completing the offer is lower than the % of customers the predicted would complete the offer, the revenue is higher in the expected calculation suggesting that the model predicts they will take the offer when the difficulty is lower.

Actual cost of a reward	Net Non-Required Costs	Net Expected Non-Required Costs
25%	\$17,784.25	\$ 15,954.48
50%	\$35,568.50	\$ 31,908.96
75%	\$ 53,352.75	\$ 47,863.45

The table above is really what the project has been trying to achieve. This is the opportunity for saving money based on better targeted offers, the value here is the costs associated with offering customers rewards for offers they would have completed anyway and as you can see, even when the actual cost of a reward is toggled down to 25% this still equates to over \$15,000.

Justification

Whilst the accuracy in phase one was higher for the baseline model, we expected this due to the imbalance in the classes and therefore accuracy is not a true indicator of predictive power in this instance. Had I not balanced the classes and then used the imbalanced training data to create models trained on accuracy score then the models would have replicated the naïve models and predicting every record as the customer failing to complete the offer. This is why the area under the curve score was used in order to create a model which focussed more on the true and false positive rates of the model.

I believe that the results from the models are significant enough to have solved the problem set out in the problem statement and give a good indicative answer as to how many people would have completed the offer anyway and the costs associated with poor targeted campaigns.

V. Conclusion

Reflection

The data processing was a somewhat fiddly, the transcript data which showed the logs of customers receiving data as well as their transactions needed a lot of work in order to get it into a shape in which it was easy to determine whether the user had completed the offer before or after they had viewed the offer and also due to my assumption that a customer could not have two valid offers at the same time I had to work out whether some offers were stopped early.

From a personal perspective I would have liked the accuracy to be closer to the baseline model, however the closer to the baseline the more likely it would be to replicate the baseline and remove all predictive power, so it is somewhat of a double-edged sword in that respect.

Improvement

As always in a project such as this, there is always room for improvement and additional things to try whether that be different algorithms, creating more features, pairing with other data. One approach that could be taken is to go a random search rather than a grid search and expand which parameters are assessed, this may lead to improved scores but also may lead to overfitting.

Increasing the features could help improve the results, adding information on the average spend of a user prior to the promotion or something along those lines may help to improve the model also as we saw that their previous behaviour in terms of completing offers was a very strong indicator of completing the new offer.

Other models could have been used such as neural networks and SVMs, this may well have led to comparable or better results however these models, often referred to as black box methods, are likely to lose some of the interpretability the chosen models provide. Striking that balance is always difficult but in a business context like this it can be so vital in order for the model results to be used and trusted by members of the business.

Future Extensions

There are a number of extensions which could be applied to this project, the most natural follow on would be to assess the marketing strategy and look at ways to improve the people that are targeted with offers. Models could be built for various things such as clustering customers in specific groups and then sending tailored campaigns based on those segments. Other options would be to build a model to determine which offer customers would react to best. Alternatively, the data could be explored further to assess which campaigns work well and which don't to inform on the composition of future campaigns.

The aim would also be to get this model into production, that way the model can be rerun with no data on a regular basis to adjust the weights and also to give scores for new customers. This could be feed into either a dashboard/web app or a system in which marketing campaigns are sent so that customers who are likely to complete the offer anyway could be omitted from it.

References

- Udacity Starbucks Challenge (link)
- sklearn
    - GradientBoostedClassifer (link)
    - RandomForest Classifier (link)
    - LogisticRegression (link)
    - GridSearchCV (link)
- pandas
    - cut (link)
    - get_dummies (link)
- How to reverse an argsort (link)
- Confusion matrices and how to plot them (link)