Project Proposal

skww86
Computer Science
Durham University
Durham, United Kingdom
skww86@durham.ac.uk

I. Project Proposal

A. Introduction

In this project, I will be working with the 'German Credit' dataset. This is a dataset containing attributes of people who had applied for a credit loan, and whether or not their applications were accepted or rejected, which was calculated based on each person's attributes. Some of these attributes relate to the person's demographics (e.g. age, gender), whereas others relate to the nature of the credit application (e.g. credit amount, duration). I am working closely with the research paper 'Computational Fairness: Preventing Machine-Learned Discrimination' by Michael Feldman [1], from which I will base a lot of the work I do and take inspiration from.

B. What I plan to do and the expected outcome

On a practical level, this data set might be used to train a machine learning model in order to classify future credit applications. This would be considered a classification problem, as each applicant is given a binary outcome of either accepted or rejected. In this project, I hope to emulate this by splitting the dataset into 2 subsets - 'train' and 'test', and implement a machine learning model to use and analyse the training set to gain intelligence to make predictions on what the classifications of the data in the 'test' set would be. I am hoping to analyse the predictive data that this machine learning model produces, and investigate the potential occurrence of any bias that it has made on the outcome between credit applications from a younger demographic of applicants, and credit applications from an older demographic of applicants. After having identified and analysed any such bias, I will then aim to implement a mitigation algorithm that will work in conjunction with the machine learning model (potentially a pre-processing algorithm to be run on the data set). I would then re-run the machine learning model and see if there is a mitigation in the bias produced in the final results. The aim would be to retain general accuracy in the machine learning model's predictions of what the classifications should be, whilst minimising any discrimination to a certain demographic of applicants should it exist.

C. Why this suits the module and my motiviation for doing it

I believe this project suits the module of 'Bias in Artificial Intelligence' because if I were to identify that the machine learning model, in its conventional form (i.e. without the use of a bias mitigation algorithm), was producing a set of classifications which were biased towards a certain demographic of people, this could be considered as discrimination. It is reasonable to think that a younger group of applicants may be more likely in general to have attributes that are undesirable towards leading to a successful credit application than older applicants. However, if a machine learning model then interprets this pattern from its training data, it may then establish that the applicant's age itself is an important factor in determining the outcome of the credit application, and use this logic in its predictions. This would be an inherent flaw in the artificial intelligence as the model would be intrinsically biased. I want to investigate how we can ensure that a machine learning model can train itself on the same data, in order to accurately predict an application's outcome, but by learning patterns between important attributes and not be driven by any patterns between demographical attributes, which in an non-discriminatory situation would not be relevant to whether or not the application should be successful. This is not as simple as just removing the demographic columns from the original data set, because other columns can still be predictive of these values, so a clever algorithm must be designed.

D. Technologies I plan to use

I plan to use the programming language 'Python' (version 3.7.4) along with various packages in order to read in and parse the dataset, and to implement algorithms that will use the dataset thereafter. Specifically, I plan to use the 'pandas' package to manage the dataset in memory and perform any actions on it. I will import the 'sklearn' package which will provide me with various tools I can use to carry out the machine learning procedure, along with useful analytical functions. I intend to utilise the 'LogisticRegression' module to carry out a Logistical Regression algorithm for my machine learning implementation. I will also use 'train_test_split', 'StratifiedShuffleSplit','accuracy_score' and 'balanced_accuracy_score' modules, again as part of the machine learning implementation – I will discuss these in further detail later on. I am using 'matplotlib' to help me plot graphs to visualise the results.

II. Project Progress

A. Data Analysis

I am using the 'German Credit Data' data set: https://online.stat.psu.edu/onlinecourses/sites/stat508/files/german_credit.csv. It contains 1000 rows; each row corresponds to a person's credit application, and 21 columns, each one providing information about the person and their credit application. The Creditability column signifies the outcome of the application, value either 1 or 0 (accepted or rejected). All columns take numerical values. Some of these columns' values are orderable, whereas in others, they represent discrete categories. In terms of any cleaning or transformation, the only thing I did was delete the following columns: Purpose, Guarantors, Type of Apartment, Telephone, and Foreign Worker i.e. columns that do not have orderable numerical values, and thus won't be compatible with the fairness algorithm I intend to implement. I split the dataset into those of age under 25, and those of age 25 or over, which I will refer to as the young and old group respectively. It is between these two demographic groups that I intend to compare the bias that an ML model applies. When skww86.py is run, the initial_analysis() function initially runs and prints to the terminal, the size of the group and, for each group, the mean and variances of every column (every remaining column is numerical; there are no categorical columns). The average value for Creditability is ~ 0.587 for the young group, and 0.72 for the old group. This is clear bias against the young group. However, we can also identify some other figures that could explain why this is the case. For certain categories, the average value for the young group is 'worse' than the value for the old group (worse in the context of trying to maximise chances of a successful application). E.g. a larger account balance is desirable but the average value for the young group is 2.18 compared to the old group's 2.64. For Value Savings/Stocks, young is 1.89 whilst old is 2.14 - the larger your savings the more likely your credit application will be accepted. Length of current employment - young is 2.89, old is 3.47. This could explain the Creditability discrepancy. However, there are one or two columns for which you can make the opposite argument; the old group on average requests a larger Credit Amount; the larger the amount the less likely a bank will grant it. So potentially the bias we see here has been further emphasised by a naive ML model that has identified age from training data as a factor in whether or not to grant credit.

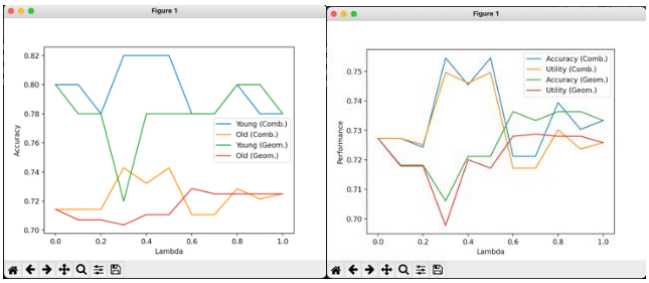
B. Conventional Implementation

This is a classification problem; the basic machine learning algorithm I chose to use is Logistic Regression, because it is a fairly standard and conventional classification algorithm, that is often used in many practical contexts. It may well be the algorithm that the German bank used/could use on its data if it were to try to predict credit risks for new applicants. The fair ML implementation that I will discuss later on is simply a data pre-processing algorithm that is, according to Feldman in his research paper, compatible with any classification algorithm, so Logistic Regression made sense to me and I was satisfied with its performance too. I then implemented a naive ML model. Aside from the initial cleaning described before, I did not carry out any further pre-processing on the data set. I randomly split the data set into 80% train, and 20% test. I ensured there was an equal proportion of successful and unsuccessful applicants in both the training set and the testing set, to improve the robustness, reliability and accuracy of the model. In terms of analysing the results, I recorded the accuracy (the proportion of the predictions that were correct), as well as the fairness, measured via 'Disparate Impact' (DI), and 'Zemel Fairness' (ZF). DI is the rate of unprivileged applicants receiving good outcomes divided by the rate of privileged applicants receiving good outcomes (here the privileged applicants are those in the old group). It follows that the larger the DI value, the greater the degree of fairness. ZF is the rate of privileged applicants receiving good outcomes minus the rate of unprivileged applicants receiving good outcomes. Whenever I refer to ZF though, I will refer to 1 minus the value calculated above, so that the final value can be considered in the same depth as DI with regards to the larger the value, the greater the degree of fairness. The results (which are outputted to the terminal) are accuracy: 0.73, DI: ~0.643, ZF: ~0.787. Before we analyse and discuss these results, we are going to run the ML model again. Since 85% of the original dataset contains applicants from the old group, when randomly sampling the testing and training sets like we did before, the testing set is inevitably going to contain a larger proportion of members from the old group than the new group. So to remove this flaw, we will ensure that half the testing set consists of randomly selected applicants from the old group, and half consists of randomly selected applicants from the young group. The size of the testing and training sets stay the same as they were before, and no other changes are made. The ML model runs on this new train/test split and the results are outputted to the terminal: accuracy: 0.685, DI: ~0.774, ZF: 0.86. There is a large increase in both the fairness measurements, showing that ensuring diversity in the testing set has a huge impact in reducing bias in the results. However, I believe one reason why this might have happened is because ensuring diversity in the testing set meant the training set was very undiverse (consisted mainly of applicants from the old group). Since the old applicants are more likely to receive good outcomes, the model was predominantly training on data with good outcomes and was therefore more susceptible to giving out good outcomes in its predictions. It is also worth noting that the accuracy reduced slightly, but not to such a large degree that the results can no longer be trusted. A reduction in accuracy is expected anyway if fairness must improve because as described previously, the original data set contains bias.

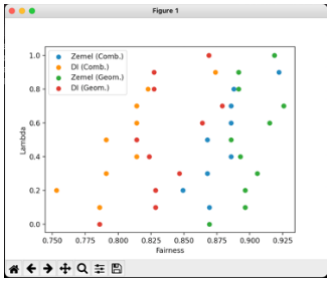
C. Fair Machine Learning Implementation

We now implement a fair machine learning method that runs on a randomly sampled testing set (like we did originally) whereby bias is mitigated in the results to a good degree, whilst still maintaining accuracy as best we can. I chose to use Michael Feldman's proposed 'Repair Function', a pre-processing tool that modifies the data set before it is passed onto an ML model. The bias occurs around the age of each applicant, but merely removing the age column wouldn't be effective because other columns can still be predictive of age. The goal of this function is to modify these other columns' values such that an ML model can no longer use them to be predictive of age. This is done by ensuring that, for each unprotected column, its values that correspond to the young group have the same distribution as its values that correspond to the old group. We do this via a 'quantile' approach, where each group's column value at any quantile will be the same.

For each column, we identify a target value at every quantile. This is done by first sorting the column values and identifying the values for each group at each quantile (selecting the median when multiple values exist, so in the case of the old group). The median of all these values is then calculated and this is identified as the target value at that quantile (for that column). This target value is then used to compute a final repair value, which is set as the new value in the relevant column for all entries at the relevant quantile. Computing this repair value is done via 2 different methods; geometrical and combinatorial; we use both in our results. The geometric method modifies the column value by a factor of the target value. The combinatorial method calculates the distance between the entry and the target values in a sorted list and selects a new value in the sorted list based on this distance. They both use a lambda parameter, which takes values from 0 to 1. At 0, the data set is unchanged, whilst as we progress towards 1, the repair process is carried out to a higher degree. This is extremely useful, as later we will analyse graphs with results from multiple different lambda values; in a practical sense someone might look at this graph and select a lambda value that optimises the degree of fairness and accuracy they desire. This is one of the key reasons I selected this fairness implementation - it is easy to analyse and compare varying 'strengths' of repair, and it also has a better practical outlook. Also, it is compatible with any ML model giving me flexibility in choosing any ML implementation I wished. We run the repair function 11 times for geometric repair at all values of lambda between 0 and 1 inclusive at increments of 0.1, and a further 11 times for combinatorial repair at the same lambda values. We measure fairness, via DI and ZF (defined previously), and accuracy, via basic 'accuracy' as the proportion of correct predictions (which we will additionally measure separately for young and old applicants) and 'utility', a measure that places equal weight between predictions that are actually 'yes' and 'no'. We created 3 plots to display the results, which are also generated by running skww86.py:



The leftmost plot above compares the accuracy in the predictions of the young applicants against the accuracy in the predictions of the old applicants, for each lambda value. The accuracy for the young group is considerably higher than that of the old group. To me this looks good. The original data set is biased against the young group, so if the accuracy in the older group was to remain high this bias would have to propagate through to the ML's predictions. The ML model has been a bit harsher on the old group this time round to help mitigate bias. Both combinatorial and geometric generally exhibit similar behaviour but there is a spike for the geometric repair's accuracy at lambda 0.3 on the young group. I'm not worried by this; in Feldman's results he too notices a spike (different to this) in one of his plots and he mentions how it is probably due to a quirk in the repair process, but the ability to measure results across all lambda values outweighs this issue as we can simply select a different lambda value if applying this implementation in a practical context. The rightmost plot above considers the holistic accuracy of the ML model for each lambda value, as well as utility. For each of geometric and combinatorial, the accuracy and utility remain tightly knitted. We know that for this dataset, 70% of the original applications were classified as successful so the high levels of utility show that the model isn't just aimlessly predicting successful applications to maximise accuracy, but rather is making intelligent choices. Aside from lambda = 0.3, 0.4, and 0.5, the combinatorial and geometric respective accuracies seem very similar. But at the aforementioned lambda values they differ quite significantly; the combinatorial repair performs well whilst the geometric repair falters. This is an example of why this analysis could be so useful for practical application - someone may look at this graph and decide they wish to use combinatorial repair at lambda = 0.3 or 0.5 because the accuracy is optimal there. But apart from the aforementioned exceptions, the accuracy does not decrease significantly, meaning our model can be trusted to retain accuracy whilst it mitigates bias.



This plot considers DI and ZF and their distribution across the lambda values. We would hope that as the dots move upwards (i.e. higher lambda values which correspond to a greater degree of 'repair') they generally move towards the right i.e. a larger degree of fairness. This is indeed the trend, so we can conclude our fairness implementation has mitigated bias. The geometric repair appears to outperform the combinatorial repair, which is useful information to anyone looking to apply this algorithm on a practical level. The rate of change of the DI is larger than the rate of change of the ZF, which is likely because the value of the ZF is initially larger than that of DI. In Feldman's paper, he makes similar plots to the first and second ones. In his graph that records accuracy and utility, he finds that accuracy and utility reduce significantly for lambda values greater than 0.7, whereas in our graph, accuracy is retained in this region. In his fairness graph, he gets the same trend that I do in that fairness increases as lambda increases, but the degree to which his fairness measurements increase is larger, especially for lambda values greater than 0.7. This all makes sense - the original data set is biased against the young group, so for an ML model's predictions to be fair to an increasingly large extent, naturally the less accurate it must be. I would therefore conclude my results are a less 'exaggerated' version of Feldman's. My accuracy is generally retained for all lambda values, which limits to a certain extent the fairness that the ML model can instigate. My results are more modest, yet still effective, and would be especially applicable to someone looking to make a small mitigation in bias whilst retaining as much accuracy as possible. The difference between my and Feldman's results do not surprise me; there's a lot of room for preferential decisions in this implementation. E.g. Feldman uses an SVM classifier; I used Logistic Regression, so our ML models are completely different, potentially explaining the larger degree of aggressiveness in Feldman's bias mitigation.

References

[1] M. Feldman, “Computational Fairness: Preventing Machine-Learned Discrimination” May 2015 in press.