Data Mining Assignment 2
Classification

Maksim Karnaukh
1st year Masters at Department of Computer Science
University of Antwerp
Antwerp, Belgium
Email: maksim.karnaukh@student.uantwerpen.be

1. Introduction

This report is a continuation to the first report about Frequent Pattern Mining. This time we want to train a classification model on the dataset, to predict whether individuals have a high or a low income. We need to make sure the classification model behaves fairly, regarding the sex of the individuals. Hence, it will be our job to not only build an accurate, but also a just classifier.

2. Feature Engineering

First of all, as mentioned in the introduction, we use the same dataset as before. We only impute the missing values for the two columns "ability to speak english" and "gave birth this year" in the same way as before.

2.1. Feature Encoding

Since the classification models we will use can't work with categorical features, we need a way to encode them. All of the nominal features have a rather small amount of distinct values. Because of this, we will use one-hot encoding for them. The ordinal features are already label encoded. This is the standard way we will use to encode our features. It is of course possible that in a later section of this paper we will apply some custom encoding, e.g. use label encoding instead of one-hot encoding for all features for a Naive Bayes model.

2.2. Feature Selection

We will have to train our classifiers on a certain combination of features. The way we select the best combination of features for a certain model will be the result of two main approaches.

The first approach is to make use of the frequent pattern analysis results from the previous assignment to choose features. The general way to do this is to look at the features/columns and examine which of them are strongly correlated with a high or low income.

The second approach is using the three general classes of feature selection algorithms: filter methods, wrapper methods and embedded methods [8] [9] [10]. Filter methods we will use are the Chi-square statistic, Mutual Info and the (Pearson's) Correlation Coefficient. Concerning wrapper methods we will utilize Forward Selection and Backward Elimination.

When we will talk about selected feature combinations, we will generally just mention the features we excluded from the total set of features of the original dataset to make things easier. Furthermore, we want our models to behave fairly, regarding the sex of the individuals. A simple way to enforce this would be to exclude the 'sex' feature from the training dataset. This will be discussed in the section on fairness.

2.2.1. Frequent Pattern Analysis Results. We look at the results from our frequent pattern analysis, more specifically at the rules that have 'high' or 'low' as a consequent. Although we won't use the support as a final decider, the rules with higher support are prioritised for our reasoning process.

We might use the confidence as final decider, since in our case it will measure the conditional probability of the target label given the presence of a certain feature. Ultimately, in this case, we would like our rules for a feature to have an imbalanced confidence, meaning that the confidence values of the distinct values of the feature (the antecedents in this case) for 'high' and 'low' are as far as possible from each other (meaning far from 50/50), for as many distinct values as possible.

One such clear feature to be added to the list of excluded features is 'age'. The 'ability to speak english' feature, because of its rule supports for its distinct values (very low supports for values 1 to 4) and partly because of the confidence for the value '0' is also part of the list. The 'workclass' feature can also be excluded for the same reason as 'age'. To summarize, the list of excluded features here would be ['age', 'ability to speak english', 'workclass'].

However, the logic of the above method is correct only if our target (income) is evenly distributed, i.e. 'low' and 'high' income value counts are both close to 50% of the total. One can see that if we for example have a dataset with 70% 'high' income, that a rule's confidence of 70% with 'high' as consequent doesn't mean as much as we would have thought with our confidence-based method. In this case, it's generally harder to interpret the rules this way.

We know from the previous paper that 'income' is indeed not evenly distributed. Roughly 66% of our samples have 'low' income and 34% has 'high' income. The simple solution here is to look at the lift values as a final decider. Rules with a high lift value indicate higher dependence, as we more thoroughly explained in the previous paper. We roughly come to the same conclusion as before if we examine the lift values.

To use the association results for fairness, we can look at the Male and Female rules section of the previous paper. We saw, among others, that there were some straightforward rules. These straightforward rules, more precisely including features 'marital status' and 'gave birth this year', had a very high dependency with the 'sex' feature (high relative support, high confidence and high lift). The reason that we discuss these here is that if we would want to remove all explicit 'sex'-related features, we would thus remove 'sex', 'marital status' and 'gave birth this year'. This will be touched upon in the section on fairness.

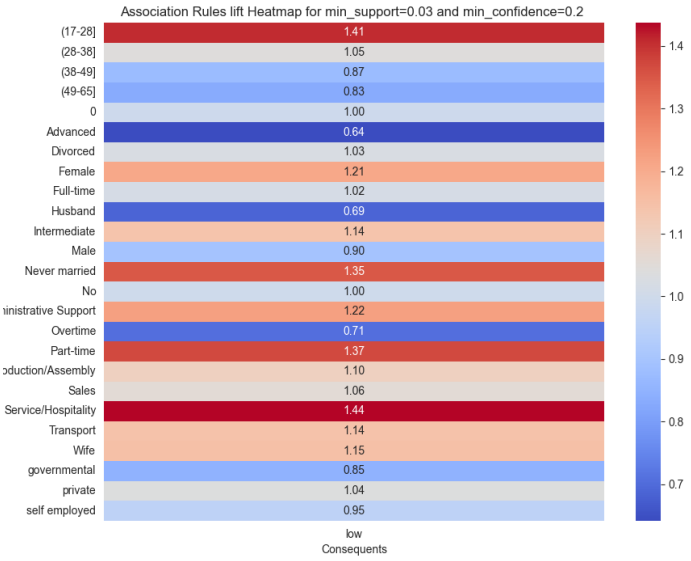

Figure 1: Lift heatmap for the association rules generated by the apriori algorithm with antecedents on the y-axis and 'low' income as consequent on the bottom.

2.2.2. Feature Selection Algorithms. The filter methods can be seen as a preprocessing techniques to give us more insight into the relationships between individual features (and the target variable). For the Chi-squared statistic, looking at categorical features where $p < 0.05$, we can see that 'marital status' generally has a high dependence with the target ('income' in this case). This also applies to features like 'education', 'occupation' and 'sex'. The Mutual Info calculation provides the same results.

The Correlation Coefficient quantifies the association between all categorical features. We see that 'education', 'sex' and 'marital status' are the highest correlated features with the target. Something to note is that 'workclass' and 'education' seem to be correlated. The same applies to ('marital status', 'sex'), ('education', 'occupation') and ('gave birth this year', 'sex').

The wrapper methods Forward Selection and Backward Elimination work by using a classifier and selecting or removing features respectively, until there is no more increase in the classifiers performance. Since accuracy is a common metric used in papers regarding the use of classification models, we will use this for our Forward Selection and Backward Elimination.

For the Backward Elimination, I used sklearn's SequentialFeatureSelector. This was carried out for every model that doesn't have a built-in feature importance function. I implemented my own Forward Selection, because I combined it together with my hyper-parameter tuning function. The way it works is, every time we choose whether to select a feature, we perform hyper-parameter tuning on the model with that current set of selected features. In the end, we have a model with the best performing hyper-parameters and the best selected subset of features.

Concerning the hyper-parameter tuning function we use for our models, we use a grid search k-fold cross validation with k=5, since our dataset is not very large. Important to note is that we first of all didn't have a dedicated test set with ground truths beforehand in addition to our dataset. I split the original dataset into a training and test set. The k-fold cross validation tuning was done on the training set only. Instead of just selecting the best resulting parameter combination from this, we take the top N results with N=10 by default, and score the model using these N different parameter combinations on the test set (technically this is the validation set) to see which one is actually the best for us to use. The reason for this is mainly to not choose a parameter combination that gave good results merely by chance.

3. Fairness

A model that is quite accurate, but highly biased and unfair is an unusable model [3]. Ensuring fairness within our model involves two main needed components: fairness metrics in order to measure how fair our model is and some kind of applied technique to make our model more fair.

3.1. Fairness Metrics

There are different ways to measure fairness w.r.t. one sensitive attribute. We will be using four main fairness metrics described in [6]. For simplicity, we'll limit ourselves to a binary sensitive variable encoded with a domain of 0, 1. Correspondingly, we will refer to the 0-class ('low' income) and the 1-class ('high' income) for the class.

Let X be the data matrix of non-sensitive features, Z the data matrix containing the sensitive feature(s) and Y the target variable. The metrics are defined as follows:

- **Disparate Impact (DI)**:
  DI = $\frac{P(\hat{Y}=+|Z=0)}{P(\hat{Y}=+|Z=1)}$
- **Discrimination Score (DS)**:
  DS = $P(\hat{Y}=+|Z=0) - P(\hat{Y}=+|Z=1)$
- **(Difference of) Equal Opportunity (DEO)**:
  DEO = $P(\hat{Y}=+|Z=1, Y=+) - P(\hat{Y}=+|Z=0, Y=+)$
- **Equalized Odds (EOdds)**:
  EOdds = $P(\hat{Y}=+|Z=1, Y=-) - P(\hat{Y}=+|Z=0, Y=-)$

We want the DI to be as close to 1 as possible, while we want the others to be as close to 0 as possible. We will also make use of the False Positive Rate (FPR) and True Positive Rate (TPR) metrics. Should we train a model and take the predictions for a split-on-sex test set, then we can calculate the male and female FPR and TPR. The difference between male and female FPR's is namely the Equalized Odds we saw above. The difference between the TPR's is equal to the Difference of Equal Opportunity. We will mainly use the four listed fairness metrics for evaluation of a model's fairness after its predictions, while we will incorporate the FPR and TPR (for males and females) into one of our model algorithms. The next subsection will discuss this in more detail.

3.2. Strategies for Improving Model Fairness

A simple way to enforce fairness would be to exclude the 'sex' feature, which is our sensitive attribute, from the training dataset. However, it is not recommended to do this since there might still be (correlated) bias from other features and besides that, the removal of the feature without a justified reason could worsen the accuracy of our models [13]. There are three general ways in which bias mitigation/fairness can be approached [5]: preprocessing, inprocessing and postprocessing techniques.

3.2.1. Fairness Through Preprocessing. Preprocessing techniques focus on the transformation of the given data to fix inherent biases. Resampling is a possibility. Our dataset is already small, so undersampling the male rows doesn't seem like a good idea. Oversampling the female rows might help since they are underrepresented. I tried oversampling using ADASYN [4], which at least evened out the male/female distribution, but it didn't quite help as much in terms of removing the bias. Other methods include reweighing (adjusting the weights of instances in our dataset to balance the impact of different groups), massaging (modifying class labels to achieve fairness) [13] and suppression (removing or suppressing the sensitive attribute from the dataset). The first two methods don't seem very convincing as they work rather counter intuitively, but we tried the third one.

We mentioned suppression, i.e. removing the 'sex' feature, a few times already and noted that this is generally not a very good idea. It however proved to be a more successful technique once we looked at the results and the small loss in accuracy. The idea here was to remove all the features that are explicitly related to 'sex'. There might still be some bias left from the other features, but we try to mitigate the bias as much as possible in total. Using the association rules results and our correlation plot, we can see that the combination 'sex', 'marital status' and 'gave birth this year' is the best one to exclude.

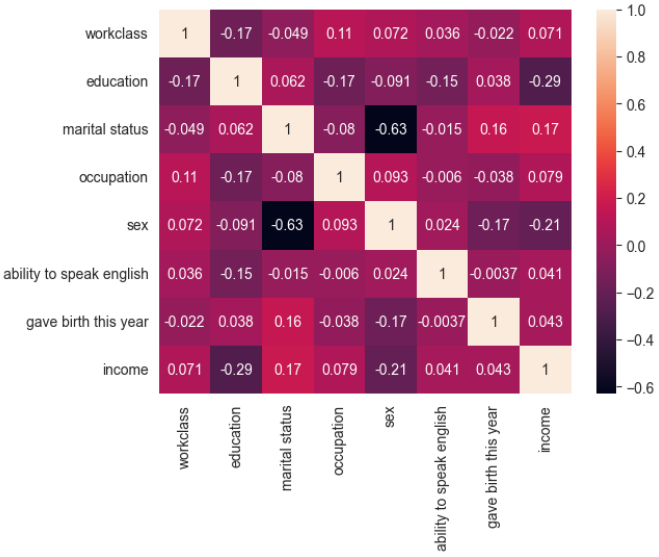

Figure 2: (Pearson's) Correlation Coefficient matrix for all categorical features.

3.2.2. Fairness Through 'Inprocessing'. Inprocessing techniques involve modifying the (learning) algorithm itself to enforce fairness constraints during model training. One such approach we adopted is the use of a composite metric within our tuning function. This composite metric combines both accuracy and fairness considerations in a weighted manner to guide the selection of hyper-parameters (during forward feature selection). By incorporating fairness into the model optimization process, we aim to ensure that the resulting model not only performs decently well in terms of predictive accuracy but also mitigates bias in its predictions.

In concrete terms, this means that we change our hyper-parameter tuning function to be able to select the best model not using the accuracy score alone, but also incorporating

the fairness metric. The fairness metric FM that we will use is the following:

$$FM = \frac{0.6 \times |FPR_{male} - FPR_{female}| + 0.4 \times |TPR_{male} - TPR_{female}|}{2}$$

Further, we can define our composite metric (CM) as follows, where the weights are both 0.5 by default,

$$CM = (w_{accuracy} \times accuracy) + (w_{fairness} \times (1 - FM))$$

We then use this composite metric instead of purely using accuracy for our selection of the best model. It has proven to give a positive effect towards bias mitigation. We could as well use any other formula; this one is rather straightforward and uses both FPR and TPR since we try to minimize both. We use the difference here since the absolute values in practice in our training are mostly both low (usually around 0.1 or 0.2, but sometimes lower or higher than those bounds). An alternative would be for example to use the ratio's.

4. Building the classification models

We have five kinds of base classification models we will use and compare in this paper: Decision Tree (DT), K-Nearest Neighbors (KNN), Logistic Regression (LR), Naive Bayes (NB) and one ensemble model which is Random Forest (RF).

Aside from using different feature combinations per model, I would quickly like to talk about the Naive Bayes model. Our dataset has mixed type features, where both 'age' and 'workinghours' are not originally categorical. Using a GaussianNB on all features without any modifications would not give proper and correct results. Furthermore, since Naive Bayes uses the independence assumption, one-hot encoding categorical features is not a smart idea. We should use label encoding for all categorical features.

One solution to the mixed type features is to use or implement a Combined Naive Bayes (CombinedNB) as I call it. It works by fitting a categorical model (CategoricalNB) to the categorical features and a numerical model (GaussianNB) to the numerical features and eventually combining the scores making use of the base formula:

$$P(Class|C_1, \ldots, C_n, N_1, \ldots, N_m) \propto \prod_{i=1}^{n}(P(C_i|Class)) \cdot \prod_{j=1}^{m}(P(N_j|Class)) \cdot P(Class)$$

where C_i stands for the i-th categorical feature and N_j stands for the j-th numerical feature. When multiplying the two scores to get the final score, we must not forget to divide by the prior.

I have implemented such a model myself. There is also a library implementation [11] that I found later on, which I used to compare my training results to. They use the name 'Mixed Naive Bayes' (MixedNB).

This is not the only Naive Bayes model we can use. Another thing we can do is categorize the numerical features 'age' and 'workinghours' in the same way as we did in the previous paper, and use a CategoricalNB on all features. In total we then have six classifiers that we can compare.

Table 1 shows the performance and fairness metrics for the models and their variants. We will primarily use accuracy to measure performance. The academic literature on the topic of classification models often uses accuracy as the default performance metric, and since the ('income') distribution of our dataset isn't abnormal or heavily skewed, this shouldn't cause the disadvantages of using accuracy to occur, e.g. the 9990-10 problem [2]. The f1-score, a combination of recall and precision, is also used as a way to double-check our performance. Should the accuracy disadvantages come in to play, we can look to mitigate them. The fairness metrics we use to measure fairness performance have been discussed already.

I trained and tested more model variants than there are in the table, but to prevent the table from getting even larger, I tried to include the more interesting ones. All of the models are hyper-parameter tuned by default, except the Naive Bayes models. As we can see by the general framing order of the models, we show the base model (hyper-parameter tuning only) and then a possible feature subset: 1 means we exclude the feature subset from the association results section, i.e. 'age', 'ability to speak english' and 'workclass', 2 means we exclude the sex-related features, i.e. 'sex', 'marital status' and 'gave birth this year'. FFS means we also used Forward Feature Selection (with our hyper-parameter tuning), which we discussed in one of the previous sections. Fairness metric means we used the composite metric (CM) as defined before instead of accuracy. For the first three models we also tried using AdaBoost for potential improvements.

4.1. Bad, Mediocre and Good Models

Here, we compare the different models regarding their performance and their fairness. We discuss at least one "bad", one "mediocre" and one "good" result, and choose one "good" model that we will use for the second part, which relates to the test set.

Although Table 1 pretty much only contains models where the fairness metric was used, we can conclude that the accuracy does drop a little bit (at most a few percent) if we try to increase fairness (use fairness metric or exclude sex-related columns). We see that removing the feature subset 1 generally makes the performance worse for every model and is usually less fair. The reason that the Random Forest model has no Forward Feature Selection variants is because this takes too long with the amount of hyper-parameter combinations we have.

TABLE 1: Performance and fairness of different trained models and their variants.
FFS = Forward Feature Selection
Feature subset 1 = {all features} - {age, ability to speak english, workclass}
Feature subset 2 = {all features} - {sex, marital status, gave birth this year}

Model	Variant	Accuracy	f1-score	DI	DS	EO	EOdds
Decision Tree	Base	0.774	0.77	0.421	-0.223	0.255	0.088
	Fairness metric	0.772	0.77	0.456	-0.208	0.193	0.090
	Feature subset 1 + Fairness metric	0.769	0.77	0.437	-0.215	0.225	0.092
	FFS + Fairness metric	0.767	0.75	0.488	-0.198	0.143	0.106
	Feature subset 2 + FFS + Fairness metric	0.762	0.76	0.701	-0.096	0.041	0.003
	AdaBoost + Fairness metric	0.791	0.79	0.403	-0.250	0.259	0.110
	AdaBoost + Feature subset 2 + Fairness metric	0.766	0.76	0.655	-0.117	0.081	0.009
Random Forest	Base	0.788	0.79	0.362	-0.264	0.329	0.109
	Fairness metric	0.786	0.78	0.375	-0.245	0.265	0.110
	Feature subset 1 + Fairness metric	0.779	0.78	0.336	-0.257	0.277	0.127
	AdaBoost + Fairness metric	0.791	0.79	0.403	-0.250	0.259	0.110
	AdaBoost + Feature subset 2 + Fairness metric	0.766	0.76	0.655	-0.117	0.081	0.009
Logistic Regression	Base	0.792	0.79	0.258	-0.302	0.438	0.124
	Fairness metric	0.791	0.79	0.266	-0.298	0.428	0.123
	Feature subset 1 + Fairness metric	0.772	0.77	0.253	-0.289	0.433	0.123
	FFS + Fairness metric	0.792	0.79	0.266	-0.298	0.426	0.125
	Feature subset 2 + FFS + Fairness metric	0.751	0.74	0.720	-0.080	0.075	0.025
	AdaBoost + Fairness metric	0.766	0.76	0.277	-0.260	0.391	0.105
	AdaBoost + Feature subset 2 + Fairness metric	0.751	0.74	0.765	-0.067	0.045	0.033
KNN	Base	0.761	0.76	0.265	-0.309	0.392	0.168
	Fairness metric	0.754	0.75	0.379	-0.241	0.272	0.121
	Feature subset 1 + Fairness metric	0.773	0.77	0.387	-0.235	0.243	0.112
	FFS + Fairness metric	0.762	0.75	0.421	-0.211	0.218	0.098
	Feature subset 2 + FFS + Fairness metric	0.752	0.74	0.717	-0.091	0.057	0.006
Combined NB	Base	0.774	0.78	0.258	-0.368	0.439	0.216
	Feature subset 1	0.772	0.77	0.178	-0.389	0.520	0.224
	Feature subset 2	0.754	0.75	0.854	-0.048	0.004	0.056
Categorical NB	Base	0.781	0.78	0.251	-0.382	0.469	0.219
	Feature subset 1	0.779	0.78	0.186	-0.383	0.512	0.214
	Feature subset 2	0.762	0.76	0.823	-0.062	0.018	0.050

Bad results here include first of all the base (hyper-parameter tuning only) variants and the fairness metric-only variants of the models, since their accuracies are not necessarily high while they perform quite badly w.r.t. fairness. Other such models are for example both Naive Bayes models using feature subset 1.

Mediocre results include the KNN model using forward feature selection and the fairness metric (still a decent accuracy and mediocre fairness results). This model used the 'sex', 'education', 'workinghours' and 'age' features. Other models are for example the AdaBoosted Decision Tree and Logistic Regression models with feature subset 2 and fairness metric (decent accuracy and rather high fairness results).

Good results include both Naive Bayes models using feature subset 2. The accuracy might be slightly lower there, but we see very good values for DI especially, and the rest. Furthermore, we have the models using forward feature selection, feature subset 2 and the fairness metric. These have generally good fairness results and a comparable accuracy.

The best model that I will use for the second part, will be the Decision Tree with feature subset 2, fairness metric and all non sex-related features (since we use feature subset 2) as a results of the Forward Feature Selection. The exact hyper-parameters used here are: {'criterion': 'gini', 'max_depth': 10, 'max_features': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'random_state': 42, 'splitter': 'random'}. This has the best performance regarding fairness and accuracy/f1-score.

5. Testing our best model

Next to the "income.xlsx" data we have received a file "test.xlsx". This is some very recent data that a bank has

collected about their loan applicants, and they want to use our best classification model to decide which people should get a loan. We will try to give a performance estimate on how our best model will behave on this data.

The naive answer to this would be to say that we should get (roughly) the same performance as we did on our validation set during training. The reason this would be naive, is because we make the assumption that the test set is very similar to the original dataset (the one we used to train our model on). In concrete terms, we make the assumption that the value distribution (for all features) is the same for both datasets.

There is actually a way to mathematically measure this. Using the Kullback-Leibler Divergence (KL Divergence) [12] [1] we can measure how close a probability distribution p is to a model distribution q. The way to interpret this value is that "D_{KL} is non-negative (≥ 0), not symmetric in p and q, zero if the distributions match exactly and can potentially equal infinity." [12] (p. 1).

TABLE 2: KL Divergence between our original dataset and the test set for each feature.

Feature	KL Divergence
workinghours	0.290940
education	0.068560
age	0.047542
occupation	0.028383
sex	0.017073
workclass	0.008632
ability to speak english	0.002802
marital status	0.001831
gave birth this year	0.000019

As we can see, there are some noticeable discrepancies in the distributions. Especially the 'workinghours' feature has a quite large difference. We ought to keep this in mind since our best model used all non sex-related features during training. The mean KL Divergence over all features is 0.051753. It's not very large, but it's there.

From this we can conclude that we cannot say for sure that our best model will perform the same on the test set as it did on our validation set. It's hard to give an exact estimate. I would say it would perform a little worse considering the overall difference in distributions. [7]

We can of course look at how many loans our model handed out, by looking at its predictions for the test set.

TABLE 3: Amount of high and low income values (loans handed out) per sex for the test set by our best model.

		Income	
		High	Low
Sex	Male	666	487
	Female	385	462
	Total	1051	949

So, now we know that the datasets have a difference, but we would still like to give a quantifiable performance estimate. In my search for papers on this topic, I wasn't lucky enough to find the exact thing I was looking for. My professor however gave me the idea of using a weighted accuracy system to give a rough performance estimate.

The idea of how it works is the following: in this case the 'workinghours' feature has the largest discrepancy, and while we could do this for multiple features, we will do it for this one only since the KL Divergences of the other ones are much lower and features like 'age' and 'education' also have a lot of distinct values which could introduce more noise into the estimate. We will express the accuracy of our best model (on the validation set) as a weighted sum of accuracies of the feature values. This should roughly equal our actual total accuracy we got on the validation set.

$$\text{Weighted Accuracy} = \sum_{\text{value} \in \text{feature\_values}} P(\text{value}) \times \text{accuracy}(\text{value})$$

The weights here are the probabilities of the feature values. The accuracy for a value is defined as the accuracy of the model of the predictions on only the samples that contain that value. We could do this for any metric and this can be extended to work for multiple features where we will then have to look at all feature value combinations and their probabilities.

Note that our best model accuracy on the validation set was 0.762 (cfr. Table 1). The weighted accuracy appears to be the same. The next step is then to calculate the feature value distribution for the test set so we can get those probabilities. We can plug in these probabilities (corresponding to their value of course) in the above formula using the same accuracies as before, since we assume the accuracies stay roughly the same for our model and only the value distribution changes. Ultimately, we get the weighted accuracy for our test set. In this case, we get 0.705. This is the rough performance estimate we will give for the test set.

6. Notes

The code and plots/pictures for this project can be found at https://github.com/MaksimKarnaukh/DataMining. The .xlsx file with the predictions can also be found there.

References

[1] Kullback-Leibler Divergence. https://hanj.cs.illinois.edu/cs412/bk3/KL-divergence.pdf.

[2] Compute accuracy of model evaluation, 2016. https://stats.stackexchange.com/questions/194240/compute-accuracy-of-model-evaluation.

[3] Amazon ditched AI recruiting tool that favored men for technical jobs , 2018. https://www.theguardian.com/technology/2018/oct/10/amazon-hiring-ai-gender-bias-recruiting-engine.

[4] ADASYN, 2024. https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.ADASYN.html.

[5] Ariba Khan Arashdeep Singh, Jashandeep Singh and Amar Gupta. Developing a Novel Fair-Loan Classifier through a Multi-Sensitive Debiasing Pipeline: DualFair, 2022. https://cap.csail.mit.edu/sites/default/files/research-pdfs/make-04-00011-v2.pdf.

[6] Constantin de Schaetzen. Increasing fairness in supervised classification : a study of simple decorrelation methods applied to the logistic regression, 2020. https://dial.uclouvain.be/downloader/downloader.php?pid=thesis%3A30729&datastream=PDF_01&cover=cover-mem.

[7] Scott Fortmann-Roe. Accurately Measuring Model Prediction Error , 2012. https://scott.fortmann-roe.com/docs/MeasuringError.html.

[8] Gunn S. Nikravesh M. Guyon, I. and L. Zadeh. Feature Extraction: Foundations and Applications, 2008. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=Feature+Extraction:+Foundations+and+Applications&btnG=.

[9] I. Guyon and A. Elisseeff. An Introduction to Variable and Feature Selection, 2003. https://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf?ref=driverlayer.com/web.

[10] Fadason T. Kempa-Liehr A. Pudjihartono, N. and J. O'Sullivan. A Review of Feature Selection Methods for Machine Learning-Based Disease Risk Prediction, 2022. https://www.frontiersin.org/articles/10.3389/fbinf.2022.927312/full.

[11] raibosome. Mixed Naive Bayes: Project Description, 2022. https://pypi.org/project/mixed-naive-bayes/#api-documentation.

[12] Jonathon Shlens. Notes on Kullback-Leibler Divergence and Likelihood Theor y, 2014. https://arxiv.org/pdf/1404.2000.

[13] Mykola Pechenizkiy Toon Calders, Faisal Kamiran. Building Classifiers with Independency Constraints, 2009. https://www.win.tue.nl/~mpechen/publications/pubs/CaldersICDM09.pdf.