CS5228: Final Project Report

Chan Wei Xin (A0086886E), Hanaé Camille Carrié (A0214205X), Kaggle Team name: Chan_Carrie

1 Introduction

The task of this In-Class Kaggle competition (`https://www.kaggle.com/c/cs5228`) is to predict whether a person's income in the United-States exceeds US$50,000 a year or not based on a modified version of the Adult dataset available on the University of California Irvine (UCI) machine learning repository [2]. This is a binary classification problem.

The code of the project is hosted on GitHub at the following address: `https://github.com/hanaecarrie/CS5228_kaggle_income50K_classification`. We used the Python language for coding, mainly the matplotlib, seaborn, pandas and scikit-learn librairies for visualisation and model training and Jupyter Notebook for their useful interactive format.

2 Exploratory data analysis

2.1 Dataset description

Attributes	Type	Range or Possible values	Comments
age	Continuous	from 17 to 90.	a few elderly above 80.
marital-status	Categorical	7 possibilities like Married-civ-spouse, Divorced or Never-married.	very few ”Married-AF-spouse”.
relationship	Categorical	6 categories like Own-Child, Husband, Not-in-family.	
education	Categorical	16 levels from Preschool to Doctorate	The 2 attributes are linked with a one-to-one relationship between each discrete number and category.
education-num	Ordinal	from 1 to 16 encoding the education level.	“education-num” ranks the “education” categories accordingly from basic to advanced.
workclass	Categorical	8 categories like Private or State-gov	70% adults work in the private sector.
occupation	Categorical	14 job kinds like Farming-fishing or Tech-support	Some consistent missing values encoded as ”?” in both ”workclass” and ”occupation”.
sex	Categorical	Female, Male.	Mutually exclusive.
capital-gain	Continuous	from 0 to 99,999.	92% adults have a 0 capital-gain.
capital-loss	Continuous	from 0 to 3,900.	95% adults have a 0 capital-loss.
hours-per-week	Continuous	from 1 to 99.	
native-country	Categorical	41 countries like United-States, Mexico, India, Canada.	90% adults are born in the US.
fnlwgt	Continuous	from 13,769 to 1,490,400	final weight attribute is the number of people the census estimates that the specific record represents.

Figure 1: Attributes of the dataset in both train and test sets

The Kaggle dataset consists of a separate training and test dataset, both consisting of 24,421 records each. Binary labels of whether the person has an annual income exceeding US$50,000 or not are only provided for the training dataset. Approximately half of the test dataset is used for the calculation of public leaderboard scores, while the other half is retained for the calculation of private leaderboard scores. The dataset consists of 13 attributes that are listed in the table of Figure 1, without the race attribute present in the original Adult dataset. Some comments about attributes range, outliers and correlations are provided in the same table.

The training dataset suffers from class imbalance, with 75.15% of the samples being from the negative class ($\leq \$50K$, label=0) and the remaining 24.85% being positive samples ($> \$50K$, label=1).

2.2 Visualisation

Let's first note that all 13 features and classes distributions are alike for both the training and the test set, suggesting that both data sets may be highly similar. Thus, for visualisation, we will present only the training set to take advantage of the label information.

2.2.1 Individual features

Figure 2 presents the histogram or categorical plot of each individual feature regarding the income category (0 or 1, $\leq$ or $>$ \$50K). The plots are not normalized in order to better appreciate the difference of samples in each sub-category. The percentage is indicated using red dots. From this figure, we can say that, among the people interviewed in this study:

age	The percentage of high income follows a bell shape. About 40% of workers between 45-55 earn more than \$50K. But the proportion of high salaries drops below 30% for workers below 35 or above 60.
sex	About 10% females and 30% males in this study have a high salary.
country	Among the huge majority of workers born in the US (not shown on the plot for visibility purposes), around 25% of them have a salary above \$50K. Among workers born elsewhere, less than 10% people from Mexico and Central and South America have a high salary, while about 30 to 45 % of people from India, North America, Europe and Asia have a high salary.
education	The chance earning more than \$50K given an education level below high school diploma is slim (below 10%) while this proportion increases dramatically with the higher education level. Their is a slight decrease for 'Asso-admin' and 'Doctorate'.
marital-status	Around 45% of married people have a high income, while less than 10% of never-married people (usually younger adults), separated, widowed or divorced people do.
relationship	Consistent with the marital status comments.
workclass	About 20% workers in the private sector earn a high salary. More than half 'self-emp-inc' earn more than \$50K.
occupation	Some occupations like Exec-Management or Prof-speciality come with a high salary in about 50% cases, while others like Handlers/Cleaners or Other services lead to high income in less than 10% cases.
hours/week	Most workers work about 35-40 hours per week. A higher percentage of people working longer have a higher salary but there are only few of them.
capital gain	Surprisingly, people with a higher capital (above 5,000) gain have lower incomes than people with a lower one.
capital loss	20% of people without capital loss have a high salary. For others, the proportion of high salary surprisingly increases with capital loss from about 5% to around 70%.
fnlwgt	The plots does not display a clear relationship between fnlwgt and income.

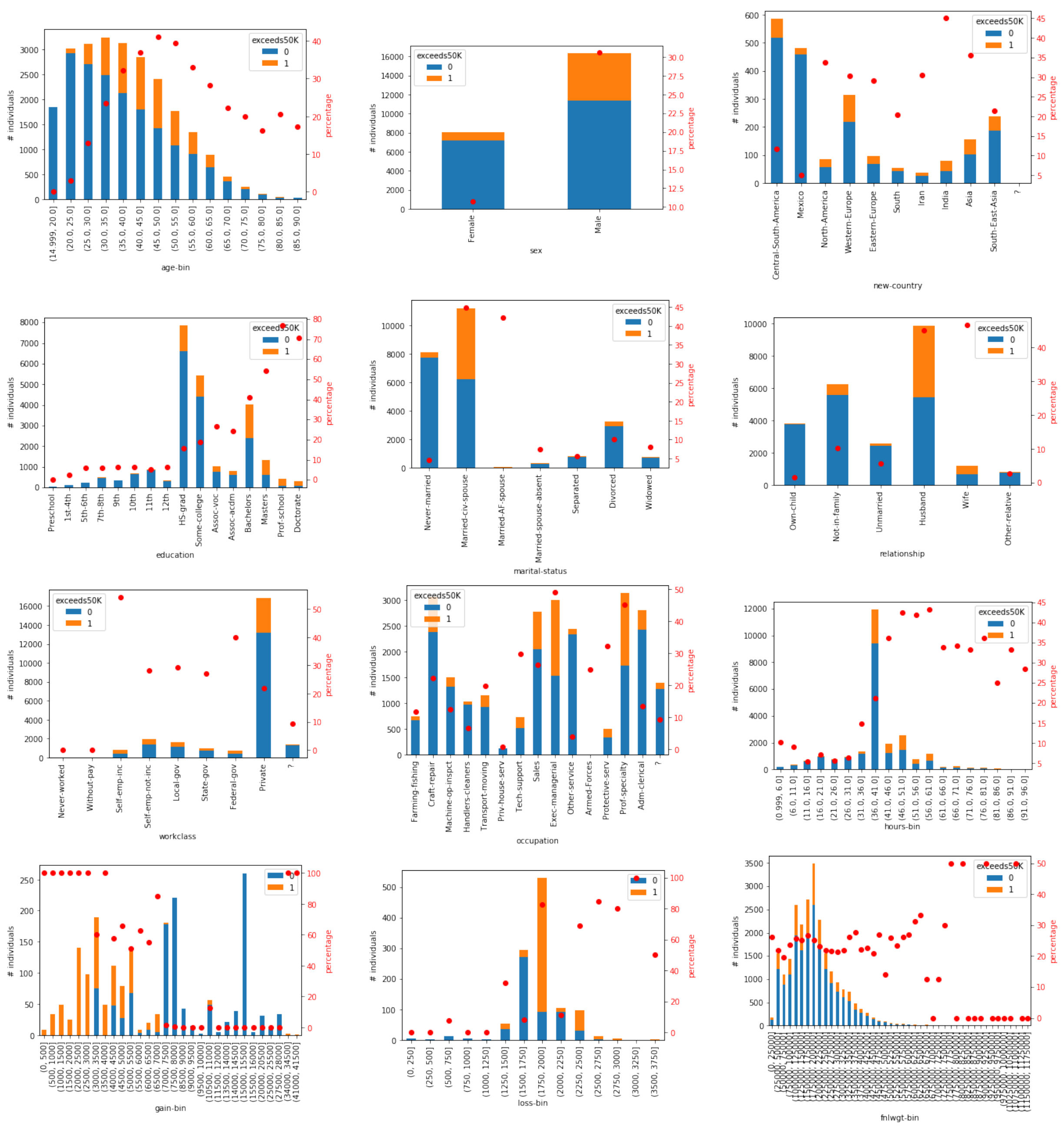

Figure 2: Features distribution VS income class

2.2.2 Groups of features

Let’s now visualise groups of features that seem to be somehow linked.

Figure 3 (a) shows that most never-married people are not in family or own-child.

Figure 3 (b) shows that some occupations like craft/repair, executive manager and sales count more self employed worker while there is a larger proportion of professors working for the government.

Figure 3 (c) shows that executive manager and professor are occupations where people have a higher level of education (more masters and doctorate)

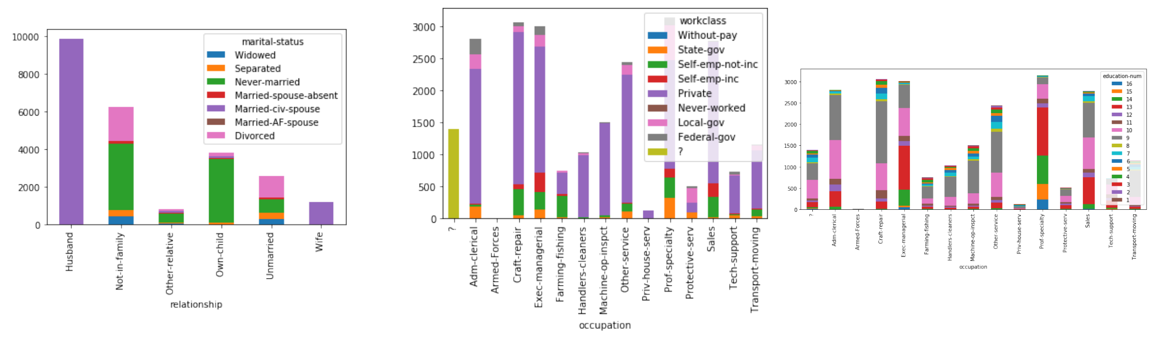

Figure 3: worker profile: relationship VS marital status (a) & job profiles: occupation VS workclass (b), occupation VS education level (c)

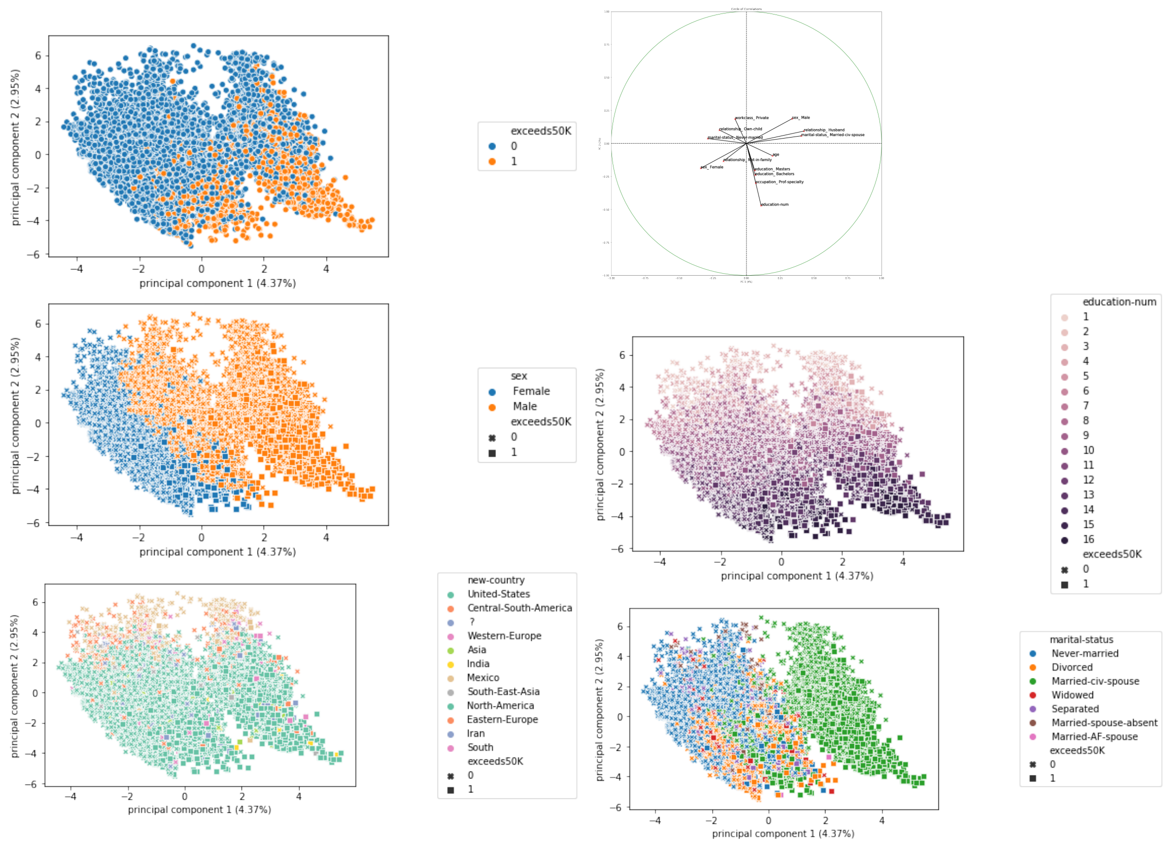

Figure 4: 2D PCA regarding income label (a) , sex (c), education level (d), native country (e) and marital status (f) and its corresponding circle of correlations (b)

2.2.3 Principal Component Analysis (PCA) & circle of correlations

To further summarize the dataset, we used PCA (after one-hot encoding and standard scaling i.e. z-score normalisation). Let's note that the 2 first principal components (PCs) explain only 7% of the data set variance. In addition, we explored higher PCs but did not find out highly valuable useful information for visualisation.

PCA is a unsupervised reduction dimension method. There is therefore no guarantee for the income class to be clustered with respect to salary. This is why this is interesting to see that most high salaries lay in the bottom right part of the graph (high PC1 and low PC2), as shown in Figure 4 (a).

If PCA is not enough to clearly distinguish between low and high incomes, it gives access to the circle of correlations in Figure 4 (b) describing how PCs are constructed as linear combinations of the original variables. Thus, PC1 mainly encodes the marital status and age. PC2 mostly encodes for higher education and workclass. The dataset projected on PC1-2 takes the shape of a butterfly in Figure 4 (a), which is quite informative. Indeed, the right wing of the butterfly are mostly married men, while the small left wing stand for not married men and the big left one for females, as displayed in Figure 4 (a) and (d). Figure 4 (b) shows an increasing education gradient from the top to the bottom of the PCA space. Most Mexican workers in this study have a low education level as shown in Figure 4 (b) and (c).

3 Data pre-processing

In this section, we detail the various steps involved in pre-processing the data. Multiple methods may be mentioned in each of the pre-processing steps, because different methods are used to pre-process the data to suit different classification algorithms.

3.1 Missing values

Missing values are encoded as "?" in both the training and test dataset. The missing values are present in three categorical attributes "workclass", "occupation" and "native-country", with Table 1 indicating the number of missing values in each attribute as well as the number of records with one or multiple missing values. Most of the records with missing values have missing values in both "workclass" and "occupation".

	# missing values per attribute			# records with n missing values		
dataset	workclass	occupation	native-country	n=1	n=2	n=3
train	1392 (5.7%)	1399 (5.7%)	410 (1.7%)	396 (1.6%)	1371 (5.6%)	21 (0.1%)
test	1407 (5.7%)	1410 (5.8%)	447 (1.8%)	425 (1.7%)	1382 (5.7%)	25 (0.1%)

Table 1: No. of missing values per attribute and no. of records with one or multiple missing values in the training and test dataset. Percentages are calculated over their respective training and test sample sizes.

A possible reason for the missing values might be because none of the categories in the attributes represent the correct choice. For example, a professional athlete would have an occupation that does not belong to any of the given categories. Hence, one of the ways that we handle missing values is to create a new category for the missing values to be classified under, for each of the attributes "workclass", "occupation" and "native-country".

Both the "native-country" and "workclass" attributes had a predominant category that accounted for a large fraction of the samples (see Figure 2 (c) and (g)) Hence, another method that we used to handle missing values was to impute the missing values of the "native-country" and "workclass" attribute with the most frequent category that occurred within the attribute, which was "United-States" and "Private" respectively. The "occupation" attribute was handled by creating a new category for the missing values, in the same way as the method mentioned above. We refer to this method as "Method 1" in Table 2.

3.2 Feature selection

"Education-num" and "Education" are redundant as explained in the table of Figure 1. Hence, we dropped the attribute "education" as "education-num" provides a ranked encoding of the same information. We chose not to include the final weight attribute "fnlwgt" for several versions of the dataset as it appears to be a noisy feature.

We combined the two attributes "capital-gain" and "capital-loss" by taking their difference in order to obtain the "net capital gain" feature, in order to reduce the sparsity in both attributes and to reduce the number of features in the data.

3.3 Feature encoding

Categorical attributes in the dataset were encoded using one-hot encoding and not ordinal encoding in order to avoid a ranked representation of the categories. However, one-hot encoding high cardinality categorical attributes such as "native-country" result in data with a high number of dimensions.

The CatBoost [4] classification algorithm does not require encoding of categorical attributes and is able to accept categorical attributes with the string data type. CatBoost implements its own strategy of transforming categorical attributes to numerical features. Other algorithms such as the categorical Naive Bayes classifier or LightGBM [3] classifier are able to handle categorical attributes but require them to be encoded numerically, using ordinal encoding.

3.4 Feature transformation

One-hot encoding of categorical attributes (especially the high-cardinality attribute "native-country") increases the dimensionality of the data and may not be suitable for algorithms with lower capacity. In order to reduce the dimensionality of the data, we perform PCA on the data after one-hot encoding and after standard scaling on the numerical attributes of the dataset, but not on the one-hot encoded features. However, centering of the data is still performed as part of the PCA.

For standard scaling, the mean and standard deviation parameters were estimated (i.e. fitted) from the training dataset and used to transform both the training and test datasets. Similarly, the principal components were determined only based on the training dataset during PCA. The test dataset was then projected onto the principal component space determined using only the training dataset. These procedures were performed in this manner in occur to avoid any form of data leakage. After both the training and test datasets are projected into the principal component space, we kept the top 10 principal components in order to give our reduced 10-dimensional training and test datasets.

In order for the data to be compatible with the categorical Naive Bayes classifier, the three continuous features "age", "hours-per-week" and "net_capital_gain" were discretised to form categorical features. The attribute "age" was divided into 8 equidistant bins between with the following boundaries: 15, 25, 35, 45, 55, 65, 75, 85 and 95. Secondly, the attribute "hours-per-week" was divided into 10 equidistant bins with the boundaries: 0, 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100. Lastly, the feature "net_capital_gain" was divided into 10 unequal bins with the boundaries: -10 000, -5 000, -1 000, 1 000, 2 000, 5 000, 10 000, 15 000, 20 000, 50 000 and 120 000.

3.5 Dataset versions

We chose different combinations of pre-processing methods at each pre-processing step to create different dataset versions that we thought would work well for each classification algorithm. We assessed the effect of using different combinations of data pre-processing methods on the binary classification performance of each algorithm through the accuracy score. Subsequently, the dataset version that gave the best accuracy score was selected for each classification algorithm. We present all the dataset versions that we use and their respective combination of pre-preprocessing methods at each pre-processing step in Table 2. The corresponding dataset version that is used for each classification algorithm is listed in Table 3.

Name	Missing values	Feature selection	Encoding	Feature transformation
A1	New category	None	One-hot	None
A2	New category	Dropped “education”; Replaced “capital-gain” and “capital-loss” with “net-capital-gain”	None	None
A3	Method 1*	Dropped “education”, “fnlwgt” and “native-country”; Replaced “capital-gain” and “capital-loss” with “net-capital-gain”	One-hot	None
A4	New category	″	Ordinal	Discretisation of numerical attributes
PCA1	Method 1*	″	One-hot	Standard scaling → PCA
PCA2	Method 1*	Dropped “education” and “fnlwgt”; Replaced “capital-gain” and “capital-loss” with “net-capital-gain”	One-hot	Standard scaling → PCA

Table 2: Different dataset versions and the combination of pre-processing methods used to create them. All pre-processing methods are mentioned in detail in their respective subsections. *Refer to section 3.1 for details on method 1.

4 Data mining

Following the allocation of a dataset version to each algorithm, hyperparameter optimisation was performed for each classification algorithm using a randomised 5-fold cross validation search. An exhaustive grid search is performed if the number of combinations of parameters is not prohibitively large. We used both accuracy and F1 (binary) scores as metrics in the randomised search. The set of hyperparameters that gave the highest accuracy score was used for the final model of each algorithm. These models are in turn used as the base classifiers in our stacked ensemble model.

There are several classification algorithms that are able to handle imbalanced datasets by allowing samples from the minority class to be upweighted. For these algorithms, a second set of hyper-parameters that optimised the F1 binary score was used to build an extra model. In general, these models have better recall but poorer precision. We use these models as additional base classifiers in a stacked ensemble model described below.

There is an option for early stopping on all of the gradient boosted algorithms mentioned in this section, where a separate validation set can be provided to evaluate the validation loss at each iteration of the gradient boosting algorithm. Early stopping is triggered if the validation loss does not improve over an arbitrary number of iterations. We did not utilise early stopping as we determined that optimising the number of iterations (i.e. number of trees) through randomised search gave better classification results.

4.1 CatBoost

CatBoost [4] is a gradient boosting decision tree algorithm that is able to handle categorical attributes with the string data type. CatBoost handles low cardinality categorical attributes by one-hot encoding, while high cardinality categorical attributes are transformed into numerical features via a permutation-based approach. By default only categorical attributes with a cardinality of two are one-hot encoded ("one_hot_max_size" = 2). We kept this default setting as the classification accuracy decreased when we increased the parameter to force CatBoost to one-hot encode all categorical attributes.

We used the two sets of hyperparameters that gave the best accuracy and F1 scores respectively in the random cross-validation search for our final two models. The model that gave the best accuracy had 700 trees, with default parameters of a maximum tree depth of 6 and "scale_pos_weight" of 1. The second model gave the best F1 score and had 1000 trees and "scale_pos_weight" of 2 that enabled it to achieve a higher recall. In general, the default parameters provided by CatBoost work well and require little optimisation.

4.2 XGBoost

XGBoost [1] is another gradient boosting decision tree algorithm that was evaluated. Unlike CatBoost or LightGBM, XGBoost does not implement any strategy to specifically deal with categorical attributes. These attributes have to be encoded as numerical values before passing into XGBoost. We used one-hot encoding to encode categorical attributes in the dataset.

XGBoost requires parameter tuning for it to achieve good performance. The first of the two XGBoost models we used had 1000 trees, with a "min_child_weight" of 2 (similar to the number of samples required in a child node), maxmimum number of leave nodes of 50, maximum tree depth of 6 and learning rate of 0.01 to prevent overfitting. The second XGBoost model has the same set of parameters but with "scale_pos_weight" set to 1.5 for it to upweight minority samples in order to boost recall.

4.3 LightGBM

LightGBM [3] is a gradient boosting decision tree algorithm that adopts a different strategy to handle high cardinality categorical features. Instead of one-hot encoding categorical features, LightGBM splits the feature by parititioning its categories into two separate subsets. Categorical features still have to be ordinal encoded and the columns they are in have to be explicitly stated to the algorithm. However, one-hot encoding categorical attributes still performed better than the above strategy. Hence, we opted to use one-hot encoded data in the end.

Two sets of parameters were determined by optimising the accuracy and F1 score respectively in a randomised search. The first model was built with the first set of parameters and consisted of 1000 trees with a maximum depth of 12. It had a learning rate of 0.1 and bagging fraction of 0.9. A minimum of 20 samples was required to be in a leaf node and the regularistion parameters alpha and lambda were both set to 1. These parameters helped to avoid overfitting of the model to the training dataset.

The second model consisted of 10,000 trees with a maximum depth of 12, and had a learning rate of 0.1 and bagging fraction of 0.8. The regularisation parameters alpha and lambda were both set to 0 and "min_data_in_leaf" was set to 5. The parameter "class_weight" was set to "balanced" that resulted in the upweighting of minority samples.

4.4 Random forest

Random forest is an ensemble classification algorithm that consists of multiple decision tree classifiers that are fitted independently on bootstrap samples of the training dataset. The scikit-learn implementation of the Random Forest classifier is unable to accept categorical attributes. We chose to one-hot encode all categorical attributes and that resulted in the dataset with the best classification performance.

The two sets of parameters that optimised the accuracy and F1 score respectively in a randomised search were used to construct the two models. The first model consists of 2500 trees with a maximum depth of 24. The parameters "min_samples_split" of 5, "min_samples_leaf" of 2 and "max_samples" of 0.8 help to prevent the individual trees from overfitting. The second model consists of 150 trees with a maximum depth of 56. It has the same parameter values for minimum number of samples for a split to occur and minimum number of samples in a leaf node as the first model. However, bootstrap sampling of the dataset is not performed ('max_samples': None) and the "class_weight" is set to "balanced_subsample" to upweight the minority positive samples.

4.5 Extra trees

Extra Trees is an ensemble method consisting of randomised decision trees, where the split value in each feature is randomly chosen instead of using the locally optimal split. Two models are created both consisting of 150 trees with a maximum depth of 56. Similarly the minimum number of samples that has to be present in a leaf node is set at 2 for both models. The only difference is that the 'class_weight" parameter is set to "balanced_subsample" for the second model.

4.6 Support vector machine

The support vector machine (SVM) classifier seeks to identify a hyperplane that separates positive and negative samples. Using the dimension reduced dataset "PCA1" (see Table 2) enabled the SVM algorithm to achieve better classification performance than the other dataset versions. We chose to enable the output of prediction probabilities, which were estimated using Platt scaling in the scikit-learn implementation. The radial basis function kernel performed the best in a grid search and was used for both models. The first model had a "C" regularisation parameter of 2 which optimised classification accuracy in the grid search. The second model had "class_weight" set to "balanced" and a "C" of 5 , parameters which had optimised the F1 score in the grid search.

4.7 k-nearest neighbours (k-NN)

The *k*-nearest neighbours algorithm classifier performed the best with dataset "A3" (see Table 2). Discarding categorical attributes with high cardinality helped to improve classification performance. However, neither dimension reduction through PCA nor standard scaling of the features helped to improve classification performance. A *k* of 21 and the Euclidean distance was used as parameters for the model as these parameters optimised the accuracy score during the grid search. Care was taken to avoid using even numbers for *k*, which might result in a tie situation between the number of positive and negative nearest neighbours.

4.8 Categorical Naive Bayes

In contrast with other classification algorithms, the categorical Naive Bayes classifier assumes that all features are categorical. The scikit-learn implementation requires categorical features to be ordinal encoded. However, all features will be treated as categorical with no rank ordering. There are limited options for tuning the parameters of naive Bayes classifiers, hence the default parameters were used for the model.

4.9 Multi-layer perceptron

The multi-layer perceptron (MLP) performed the best when fed with the PCA-transformed dataset "PCA2" (see Table 2). Using a reduced-dimension dataset allowed for use of a less complex model with fewer parameters. After a grid search, a single MLP model with 14 hidden layers of size 10, 12, 14, 16, 16, 16, 16, 16, 16, 14, 12, 10, 8, 4 was used as a base classifier, with a L2 regularisation parameter "alpha" of 0.01.

4.10 Stacked ensemble model

We employed a stacked ensemble model that involves training a meta-classifier that learns to combine predictions made from several base-level classifiers in an attempt to further improve classification accuracy. Two models were constructed for each algorithm with the ability to up-weight minority samples, namely the CatBoost, LightGBM, XGBoost, Random Forest, Extra Trees and SVM algorithms. A single model was constructed for the MLP, k-NN and categorical Naive Bayes classifiers. In total, 15 base classifiers were used; their details are mentioned in the subsections above.

To obtain the predicted values from the base classifiers, we followed the training procedure mentioned in [5]. In order to obtain the prediction probabilities of the base classifiers, 5-fold cross validation was performed. The prediction probabilities from the validation set for each fold is recorded and subsequently concatenated to obtain the prediction probabilities over the entire training set of size N. The prediction probabilities from the J base classifiers are concatenated to form a $N \times J$ feature matrix (Z) which together with the prediction labels y are known as the "level-one" data. Prediction probabilities of the test dataset were obtained by re-training the base classifiers on the entire training dataset before carrying out the predictions, instead of using the k-fold cross-validation trained models.

We concatenated feature matrix Z with dimension reduced dataset "PCA2" (see Table 2) to use as our "level-one" data. Classification performance was improved as compared to using feature matrix Z alone as "level-one" data for the meta-classifier.

5 Evaluation

In this Kaggle competition, the public leader-board scores are calculated using the mean F1 score, also known as the micro-averaged F1 score. The micro-averaged F1 score is equivalent to the accuracy score in a binary classification setting. Hence, we tune the hyperparameters of our models in order to optimise the accuracy score. The binary classification performance of the various models are also evaluated using the 5-fold cross-validation accuracy score. However, as there is a severe class imbalance in the training set, the accuracy score may be high even for a naive classifier that predicts the majority class (in this case the negative class). Hence, we provide the accuracy score achieved by this naive classifier as a performance baseline in Table 3.

In addition to the accuracy score, we also evaluated our models using the binary F1 score, which is calculated using the prediction and recall of only the positive class. The binary F1 score is the harmonic mean of the precision and recall of the positive class. A naive classifier that predicts all samples as negative would have a binary F1 score of 0 as it has 0% recall and precision.

We chose to use the stratified k-fold cross validation approach to evaluate binary classification performance, as opposed to the holdout cross validation approach where a separate validation set is split from the training dataset. In holdout cross validation, evaluation is performed on a fixed validation set. Thus, the evaluation performance is subject to which samples get partitioned into the validation set and a different random split may result in very different evaluation results. The stratified *k*-fold cross validation approach overcomes this problem by performing *k* rounds of cross validation, each using one of the *k* folds as the validation set. The results from each round are averaged to obtain the final score. Furthermore, the partitioning of the training dataset into *k* folds is done in a stratified manner to ensure that each fold has an approximately equal ratio of positive and negative samples.

Both the metaclassifier and base classifier use 5 fold CV.

6 Results

Dataset	Classification algorithm	Accuracy / %	F1 score (binary)	Leaderboard
87.48 +/- 0.47 A2	Stacked ensemble	87.48 ± 0.18	0.7144 ± 0.0029	87.649
A2	CatBoost	87.45 ± 0.18	0.7144 ± 0.0029	87.649
A1	LightGBM	87.21 ± 0.20	0.7121 ± 0.0070	87.706
A1	XGBoost	87.23 ± 0.31	0.7069 ± 0.0082	87.485
A1	Random Forest	86.37 ± 0.29	0.6815 ± 0.0078	86.543
PCA2	Multi-layer perceptron	85.30 ± 0.41	0.6744 ± 0.0203	-
A3	*k*-NN	85.24 ± 0.37	0.6608 ± 0.0093	-
A1	Extra Trees	85.00 ± 0.29	0.6479 ± 0.0095	84.226
PCA1	SVM (radial basis function)	85.17 ± 0.32	0.6399 ± 0.0079	-
A4	Naive Bayes (categorical)	81.85 ± 0.34	0.6753 ± 0.0050	-
-	Naive majority class classifier	76.00 ± 0.01	0.0000 ± 0.0000	-

Table 3: Comparison of 5-fold cross-validation classification results between different models. Hyperparameters of models have been optimised via a randomised search through the 5-fold cross-validation accuracy score. Each classification algorithm is fed with the dataset version that has been found to optimise its classification performance. Refer to Table 2 for details on the dataset

References

[1] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[2] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[3] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3146–3154. Curran Associates, Inc., 2017.

[4] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In *Advances in neural information processing systems*, pages 6638–6648, 2018.

[5] Mark J Van der Laan, Eric C Polley, and Alan E Hubbard. Super learner. *Statistical applications in genetics and molecular biology*, 6(1), 2007.