Pipelining Assisted XGBoost Income Prediction

CS5228 Course Project

May 2020

Group Name on Kaggle: DataMiner

Li Wencheng	Luan Min	Jiang Mingliang
A0212192R	A0105743L	A0061100J

Abstract

As a Kaggle in-class competition, we examined the Census Income dataset available at https://www.kaggle.com/c/cs5228. Our goal is to predict whether an individual's income per year will be greater than $50,000 based on attributes from the census data. We practiced Exploratory Data Analysis and Data Preprocessing, explored industrial grade New Models beyond the scope of the class and conducted Pipeline to achieve more efficient tuning. The final result is ranked number 2 on the Kaggle class leaderboard, as of the submission of this report.

Contents

Introduction

By implementing various preprocessing techniques and fine-tuning the XGBoost model, our team is able to achieve the highest Kaggle score of 0.87837 on CPU: Core i7-9700K @ 3.60GHz and 0.87821 on CPU: Core i5-8265U @ 1.60GHz.

Dataset

The training dataset has 24,421 entries (rows), each entry is an individual who provided input to the census research. Each of these entries has 14 attributes (columns) representing different categories of information collected about the individual.
The set of attributes are explained here:

- Age: the age of the individual [continuous]
- Workclass: the type of work of the individual [categorical]
- Fnlwgt: an undescriptive attribute [continuous]
- Education: the individual's level of education [categorical]
- Education-num: a duplicate level of education in the numerical form [continuous]
- Marital-status: the individual's marital status [categorical]
- Occupation: the individual's occupation [categorical]
- Relationship: the role the individual assumes in the family [categorical]
- Sex: the individual's biological gender [categorical]
- Capital-gain: the individual's capital gain in a fixed period [continuous]
- Capital-loss: the individual's capital loss in a fixed period [continuous]
- Hours-per-week: the number of hours the individual works in a week [continuous]
- Native-country: the individual's native country [categorical]
- Exceeds50k: whether the individual's annual income is greater than $50k [categorical]

Implementation Methodology

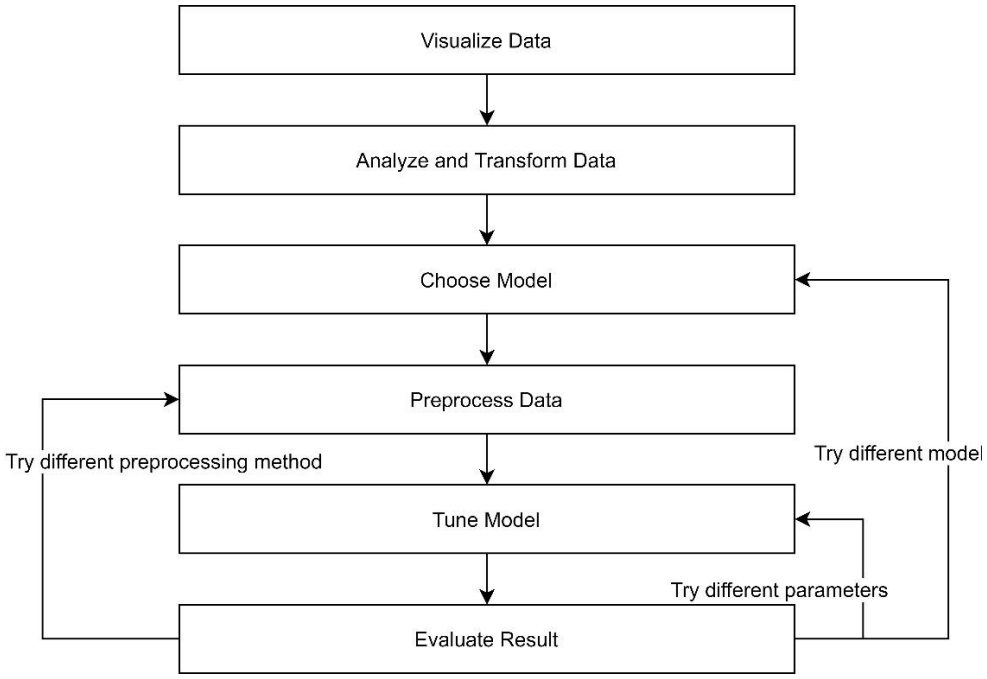

Figure 1: Project Workflow

We adopted this iterative workflow for our project implementation. This process has a number of advantages:

- Our model selection is not rigid but more dynamically decided by the rounds of execution results.
- Our data preprocessing is also better calibrated through this iterative process.
- With our pipelined approach, our tuning is also more robust than a simplistic grid search.

Exploration Analysis and Data Preprocessing

Data Correlation Analysis

Through plotting the pairwise relationships and calculating the correlation coefficients between “exceeds50k” and other continuous input attributes, we know that all the continuous input attributes except “fnlwgt” have positive linear relationship with “exceeds50k”. For “fnlwgt”, its correlation with “exceeds50k” is weak. Therefore, less effort is put into tuning and using it for our prediction.

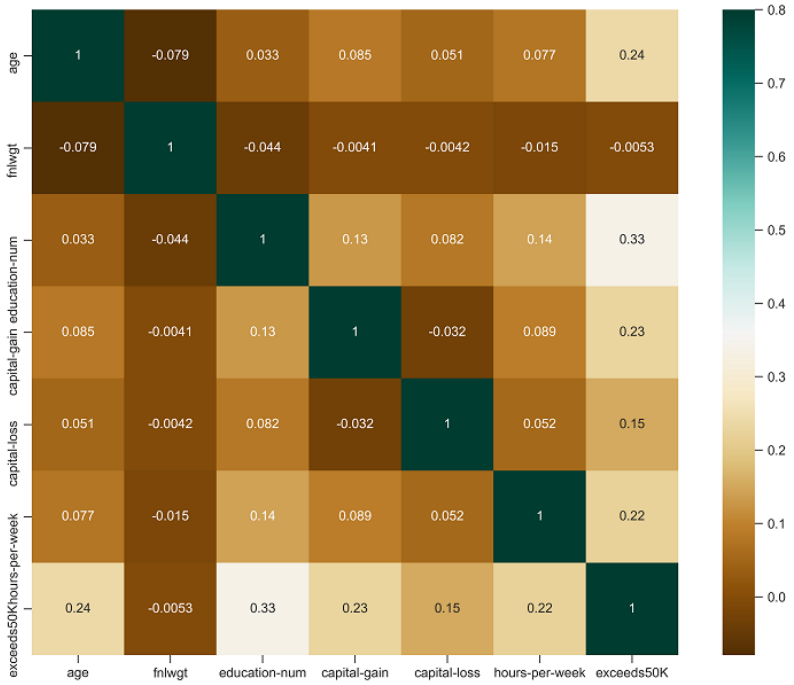

Figure 2:Heat Map between “exceeds50K” and attributes with continuous value

Merging categories for Marital-status

We merge categories of Marital-status into two representative categories, namely “Complete_Family” and “Incomplete_Family” for model training, based on the proportion of “exceed50K” and common sense.

We can observe that before combined into one category, the child categories have a comparable correlation to “exceeds50K” as shown in Figure 3 below.

- “Married-civ-spouse” and “Married-AF-spouse” are identified as “Complete_Family”.
- “Never-married”, “Separated”, “Widowed”, “Divorced” and “Married-spouse-absent” are identified as “Incomplete_Family”.

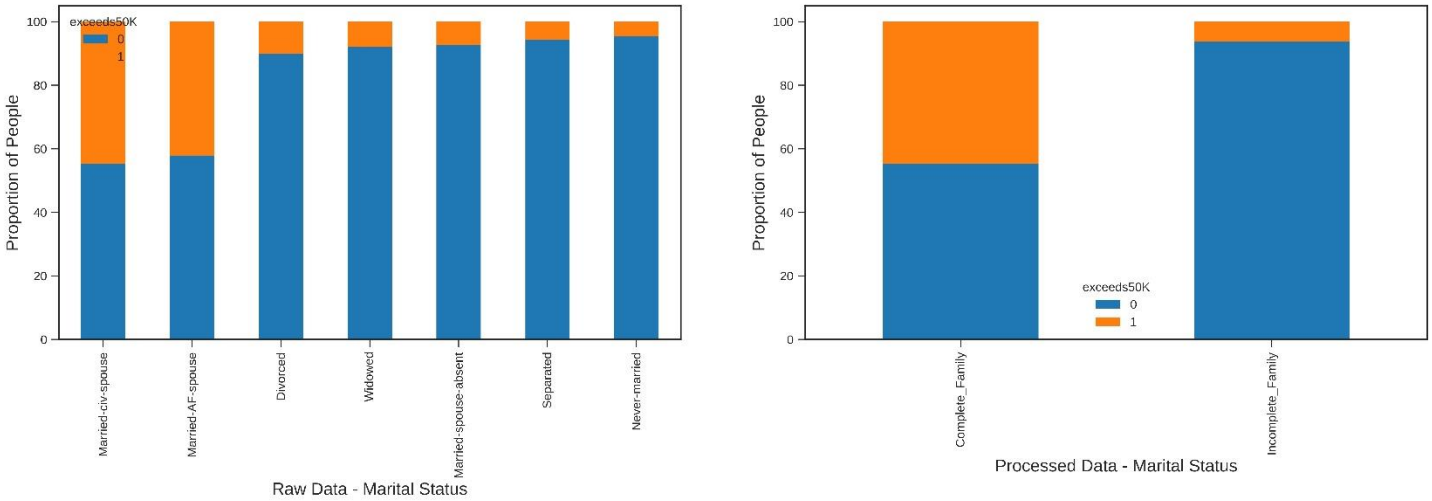

Figure 3： Proportion of people “exceed50K” before and after merging categories of Marital-status

Merging categories for Native-country

As our goal is to predict income, the economic situation and income receipt information of different countries will reveal information related to the citizens' income. We leverage on 2 published research ([1] Income Receipt Per Capita in 1994, [2] GDP Per Capita in 1994) to gauge a country's level of development, and rank the countries according to the two indexes below as shown in Figure 4. The categorization of the countries mainly refers to the Income Receipt Per Capita in Figure 4. For some of the countries without information of income receipt, we will refer to the GDP Per Capita.

Country	Rank of Income Receipt	Category
England	6	Better Than US
Holand-Netherlands	7	Better Than US
Japan	8	Better Than US
Ireland	10	Better Than US
Germany	12	Better Than US
France	14	Better Than US
United-States	15	United-States
Canada	17	Lower Than US
Italy	18	Lower Than US
Portugal	24	Lower Than US
Greece	40	Lower Than US
Hungary	48	Lower Than US
South	51	Lower Than US
Philippines	53	Lower Than US
Trinadad&Tobago	57	Lower Than US
Thailand	58	Lower Than US
Dominican-Republic	59	Lower Than US
Jamaica	60	Lower Than US
Mexico	64	Lower Than US
Columbia	76	Lower Than US
Peru	81	Lower Than US

Country	Rank of Income Receipt	Category
Poland	83	Lower Than US
Guatemala	95	Lower Than US
El-Salvador	96	Lower Than US
Ecuador	99	Lower Than US
China	103	Lower Than US
Honduras	105	Lower Than US
Iran	112	Lower Than US
Laos	118	Lower Than US
Nicaragua	120	Lower Than US
India	131	Lower Than US
Cambodia	141	Lower Than US
Cuba	No ranking info	Lower Than US
Puerto-Rico	No ranking info	Lower Than US
Taiwan	No ranking info	Lower Than US
Yugoslavia	No ranking info	Lower Than US
Vietnam	No ranking info	Lower Than US
Haiti	No ranking info	Lower Than US
Scotland	No ranking info	Lower Than US
Hong	No ranking info	Lower Than US
Outlying-US(Guam-USVI-etc)	No ranking info	Lower Than US
?	NA	Unknown

Figure 4: Rank of Income receipt of different countries and corresponding categorization

- "?" is replaced with the name "UNKNOWNcountry" and recategorized as "UNKNOWNcountry" category.
- Based on the 1994 income receipt ranking research, the non-US countries are recategorized into "Better_Than_USA" and "Lower_Than_USA".
- Countries with higher Income Receipt Per Capita than the US are categorized into "Better_Than_USA", including "Germany", "Japan", "England", "Ireland", "France", "Holand-Netherlands".
- Countries with lower Income Receipt Per Capita than the US are categorized into "Lower_Than_USA", including other non-US countries excluding "UNKNOWNcountry".
- Those with the native-country as "United-States" are unchanged and put in the category "United-States", because we observe that more than 89% of the records are under this category. With an overwhelming proportion compared to other countries, it is necessary to separate "United-States" from other countries.
- We choose not to re-categorize based on Figure 5, a direct plot of countries against the countries' proportion of "exceeds50k" for these reasons: 1) Some countries with very high such proportions have only few records. We deem this as non-representative. 2) People from countries like "China" and "Iran" tend to have biased attributes. For example, people from these countries have a relatively higher average education level and get a much higher probability to take "Prof-specialty" as the occupation.

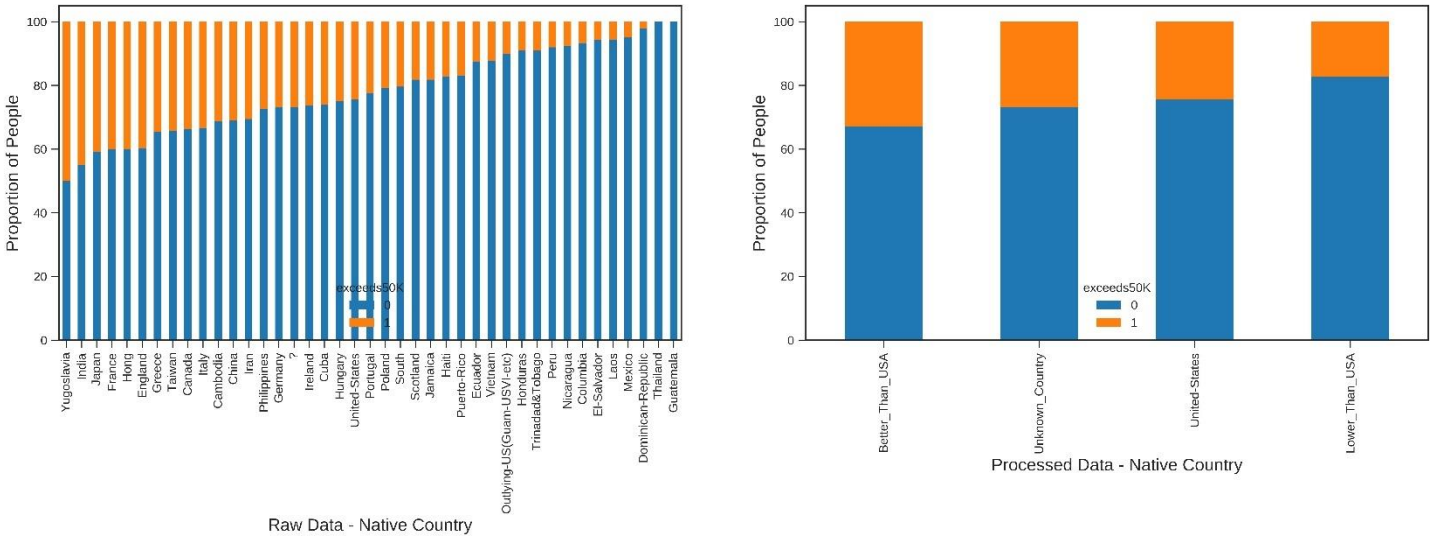

Figure 5: Proportion of people “exceed50K” before and after merging categories of Native-country

Merging categories for Relationship

- “Husband” and “Wife” are identified as “In_Family”. And other categories remain the same.

The reason why we categorize “Husband” and “Wife” into the same category “In_Family”, is that we already have gender information in the Sex attribute and the combination of “In_Family” and “Sex” is enough to provide us the information of “Husband” and “Wife”. Therefore, to avoid duplication of information and overfit training of the prediction model, it is necessary for us to combine the two categories.

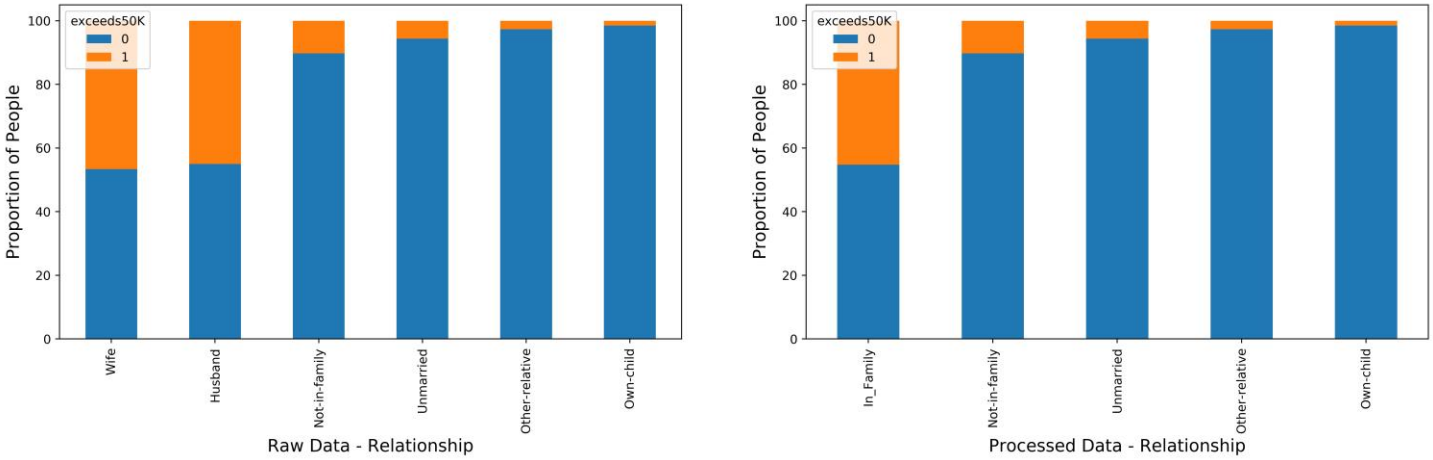

Figure 6: Proportion of people “exceed50K” before and after merging categories of Relationship

Merging categories for Workclass

For the Workclass attribute, we merge some of the data: 1) Workclass “Without-pay” and “Never-worked” are combined into one category, because both imply no income of a person. 2) “State-gov” and “Local-gov” are combined into one category, because they are of similar workclass in the United States.

The similar percentage of “exceeds50k” for these categories is also indicative of our approach.

- “?” is re-categorized as “UNKNOWNWorkclass”.
- “State-gov” and “Local-gov” are merged to “State & Local gov”.
- “Without-pay” and “Never-worked” are merged to “Without-pay & Never-worked”.

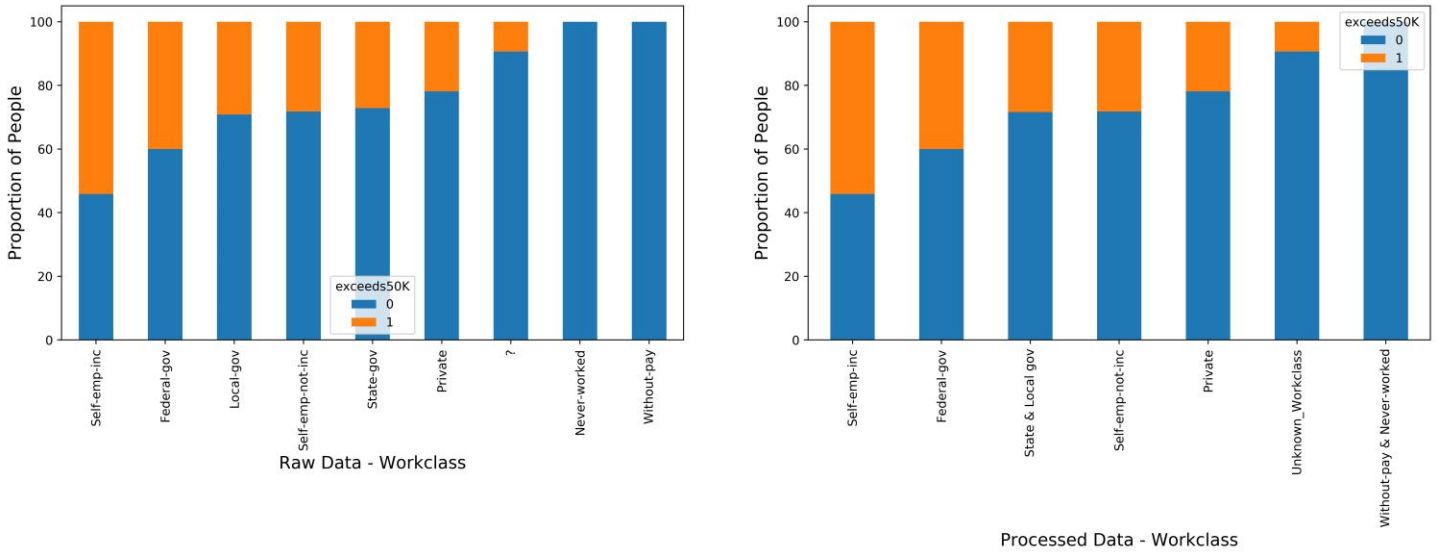

Figure 7: Proportion of people "exceed50K" before and after merging categories of Workclass

Deleting Education-num (Data analysis on Education and Education-num)

Through observations on the data value of attributes "Education" and "Education-num", we know that there is a one-to-one correspondence between the two attributes. The "Education-num" value is the rank of "Education" level from lowest to highest, but not the "years of education" taken by the person. We conclude that "Education-num" is just a duplication of "Education", which may cause overfit training on the prediction model. The numerical value also has the risk of misleading the model to build a numerical relationship with the prediction result. Therefore, we drop the attribute "Education-num" for our prediction model training.

Handling Continuous Values

MinMaxScaler is selected to scale the continuous features because:

- The distribution is not Gaussian, the standard deviation is small, MinMaxScaler performs better than StandardScaler.
- There are not so many outliers, therefore, it is unnecessary to use RobustScaler.
- Normalizer works on the rows, not the columns, which is not very suitable in such a scenario.

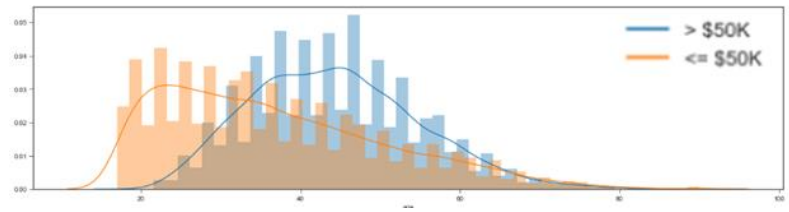

Figure 8: Age

- The younger the age, the lower the income.

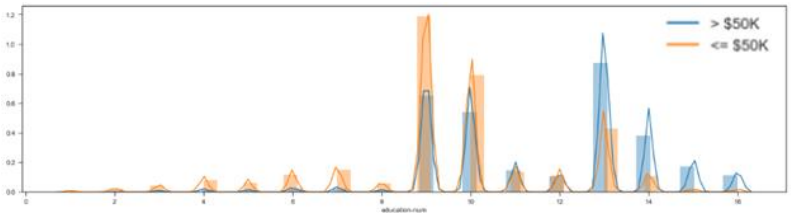

Figure 10: Education-num

- The relationship between education-num and income is not monotonic or linear. This column is dropped.

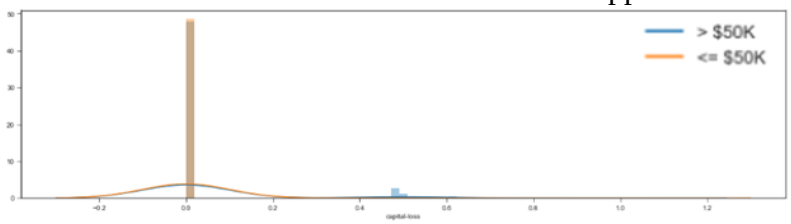

Figure 12: Capital-loss

- The majority values of capital-gain and capital-loss are zero, with a very high maximum value.

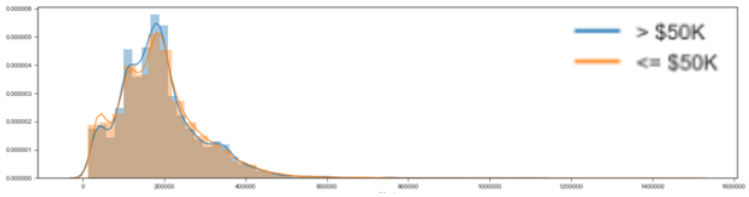

Figure 9: fnlwgt

- The difference in proportion of “exceed50K” among different fnlwgt is not straight-forward and obvious.

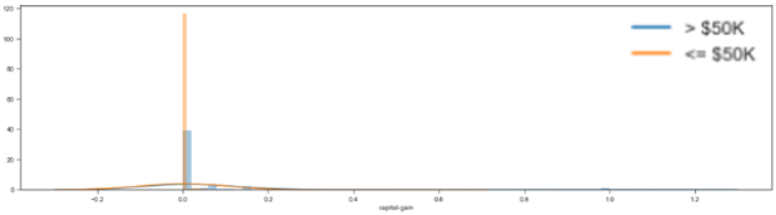

Figure 11: Capital-gain

- The graphs of capital-gain and capital-loss displayed have been scaled by MinMaxScaler.

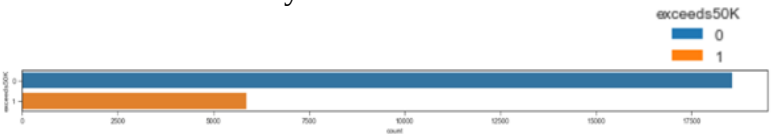

Figure 13: Proportion of different values in “Exceeds50K”

- The outcome is imbalanced, with the vast majority below 50K. Therefore, it is necessary to also use f1 score for model evaluation.

Handling missing values

It is found that replacing “?” with a new category performs better than 1) deletion, 2) replacement with the most frequent category, and 3) replacement with the predicted result of some algorithms.

Method Attempted	**Details**
1) Deletion	Delete the rows with “?”
2) Replacement	Replace with the most frequent category
3.1) Logistic Regression Prediction	Attempted LR model: LogisticRegression(penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1)
3.2) SVD Prediction	Singular Value Decomposition
3.3) KNN Prediction	Attempted KNN model: n_neighbors=5
3.4) SVM Prediction	Attempted SVM model: SVM(penalty='l2', loss='squared_hinge', dual=True, tol=0.0001, C=1.0, multi_class='ovr', fit_intercept=True, intercept_scaling=1, class_weight=None, verbose=0, random_state=None, max_iter=1000)
3.5) RF Prediction	Attempted RF model: RandomForestClassifier(n_estimators=100, criterion='gini')
New category	Replacing “?” with a new category named “Unknown”

Other unchanged categorical columns

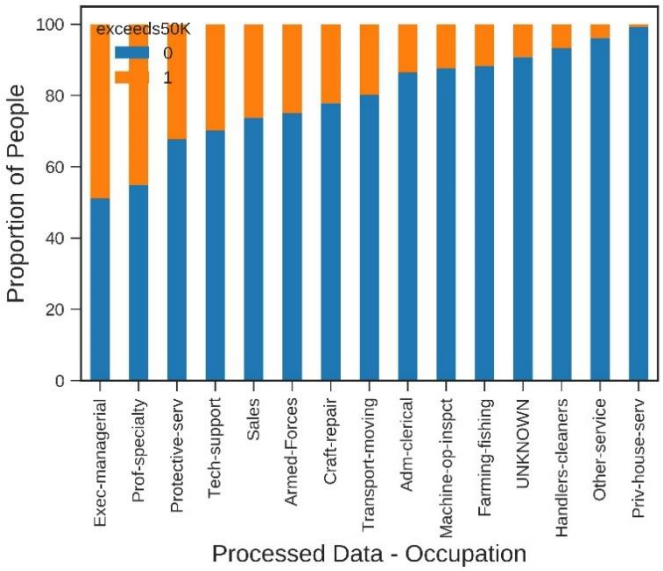

Figure 14: Occupation

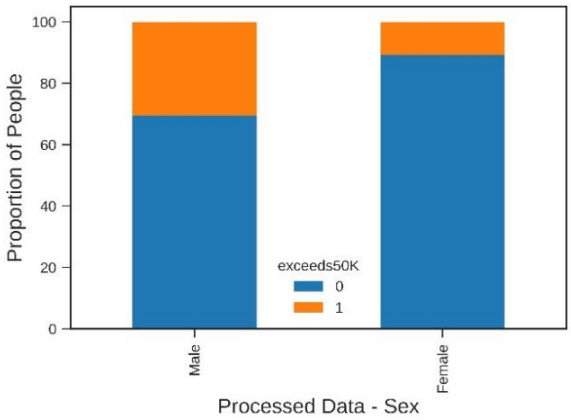

Figure 15: Sex

- The males have a higher proportion of "exceed50K" than the females.

Data Mining

Model selection

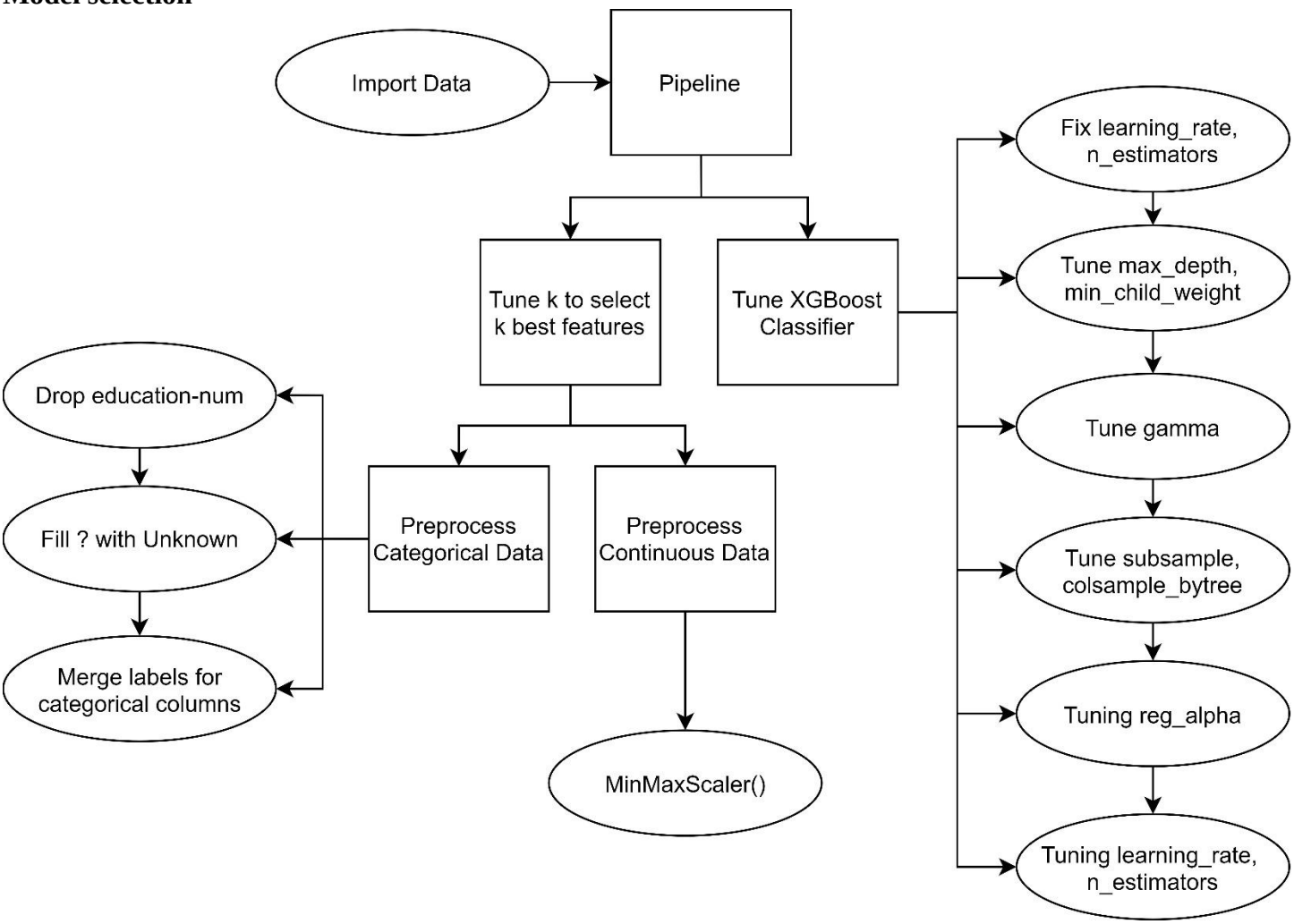

Figure 16: Code Structure

For tuning hyperparameters in different models, we apply Pipeline to explore the best combination of a given range of parameter sets. The Pipeline includes data transformation for categorical columns, data scaling for numerical columns, selecting k best features, model selection and parameter tuning for the model.

Taking the tuning process for XGBoost model as an example: As the hyperparameter “learning rate” and “n_estimators” are important basic parameters related to each other, we use GridSearchCV method to get the best combination of these two hyperparameters as the start of tuning: “learning rate” ranged from 0 to 0.3 with a step of 0.05 and “n_estimators” ranged from 0 to 1000 with a step of 100. For other parameters, we set them to the default value.

After getting the combination with the highest score under wide ranges and large steps, we perform another iteration for narrower ranges and smaller steps. After several iterations of evaluation, we get the best combination of “leaning_rate” and “n_estimators”. Then we continue to evaluate other important hyperparameters: “max_depth” & “min_child_weight”, “gamma”, “subsample” & “colsample_bytree” and “reg_alpha” in sequence as shown in Figure 16 with similar methods when training “leaning_rate” and “n_estimators”. After tuning all the above hyperparameters, we come back to the two most important parameter “leaning_rate” and “n_estimators and tune them again, as it is better to lower the learning rate and add more trees.

Reproducibility: CPU=Core i7-9700K @ 3.60GHz, random seed=42

Pipeline	Data Preprocessing, including data transformation and data normalization
	Select k Best, select the best k features
	Define a classifier for tuning
GridSearchCV (10-fold)	estimator =Pipeline
	Tuning hyperparameters for XGBClassifier and k for SelectKBest
	scoring = {'AUC': 'roc_auc', 'Accuracy': make_scorer(accuracy_score)}

Model	**Hyperparameters**	**Kaggle Score**
Random Forest	RandomForestClassifier(bootstrap=True, criterion='gini', min_samples_leaf=1, min_samples_split=15, n_estimators=2000)	0.86248
GradientBoost	SelectKBest = 12 GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None, learning_rate=0.1, loss='deviance', max_depth=3, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=100, n_iter_no_change=None, presort='deprecated', random_state=None, subsample=1.0, tol=0.0001, validation_fraction=0.1, verbose=0, warm_start=False	0.87567
CatBoost	SelectKBest = 12 {'clf': <CatBoostClassifier>, 'clf__border_count': 47, 'clf__depth': 6, 'clf__iterations': 300, 'clf__l2_leaf_reg': 22, 'clf__learning_rate': 0.1, 'clf__random_seed': 42}	0.87608
XGBoost	SelectKBest = 11 (Executed all the preprocessing methods) {'classifer': XGBClassifier(base_score=None, booster=None, colsample_bylevel=None, colsample_bynode=None, colsample_bytree=0.5, gamma=0.1, gpu_id=None, importance_type='gain', interaction_constraints=None, learning_rate=0.05, max_delta_step=None, max_depth=6, min_child_weight=None, missing=nan, monotone_constraints=None, n_estimators=300, n_jobs=None, num_parallel_tree=None, objective='binary:logistic', random_state=42, reg_alpha=0,	0.87837

	reg_lambda=None, scale_pos_weight=None, subsample=1, tree_method=None, validate_parameters=False, verbosity=None), 'kbest__k': 11, 'kbest__score_func': <function chi2>}	
Majority Vote	Merged the above CatBoost, Gradient Boost, XGBoost	0.87649

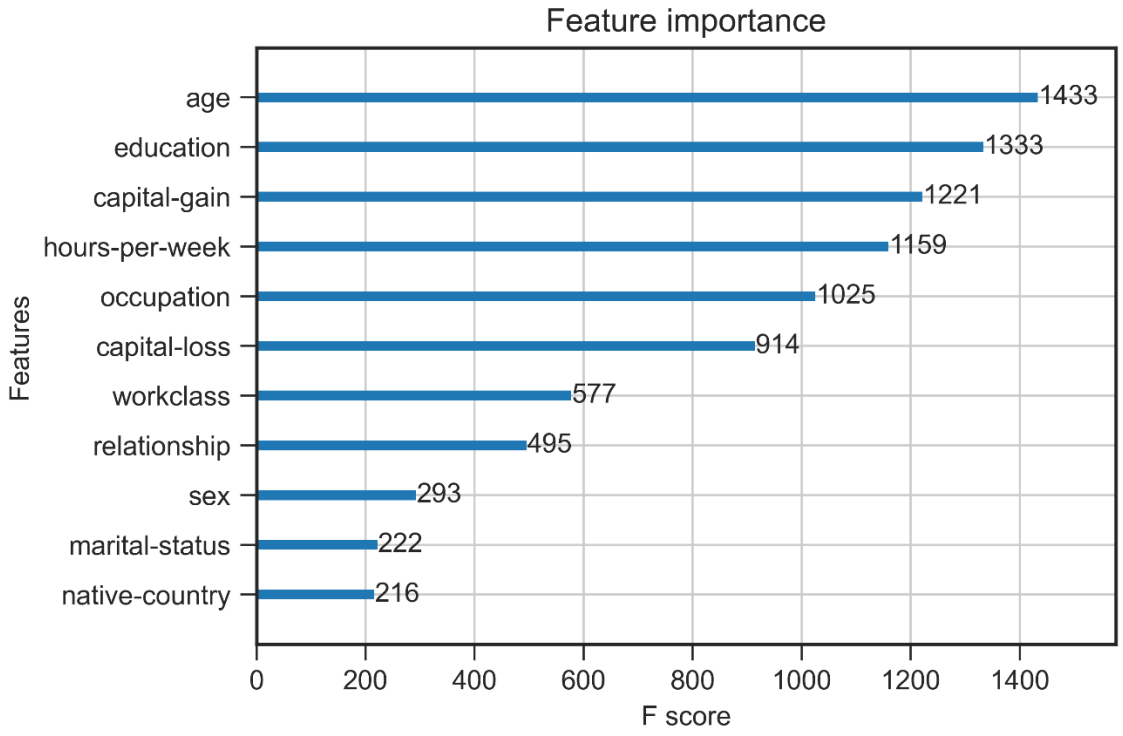

Figure 17: XGBoost importance plot

It is found that all three boosting algorithms can result in good result. The majority vote of the three boosting models does not improve the final result. It is also discovered that the best result occurs when selecting the best 11 features, when “Education-num” is dropped and “fnlwgt” is ignored, which accords with our data correlation analysis. The most relevant column is “Age”, while the least relevant column is “Native-country”.

Evaluation and Interpretation

For the evaluation of different models on predicting the result, we use Pipeline and 10-fold Cross-Validation. The dataset is split based upon the predefined random seed for reproducibility. We train the model using the training dataset and compare the performance of different models based on the accuracy and F1 score of the predicted results on the cross-validation dataset.

As the percentage of the records “exceeding the 50k threshold” and “not exceeding” are 24% and 76% respectively, we realize that the result is imbalanced. Therefore, we take both accuracy and F1 score into consideration for our model evaluations.

XGBoost

After intensive tuning for different methods, we find that the XGBoost method has the best prediction result. This is because XGBoost is an ensemble of decision trees that uses the Gradient Boost framework, which minimizes errors when constructing multiple sequential trees. It has consistently shown good results on small-to-medium tabular data [3]. In addition, XGBoost's regularization technique further reduces the overfitting and generalization errors, and compared to other scikit-learn families of algorithms, XGBoost has a good built-in missing data handling feature.

We find that XGBoost performs well even when we leave continuous input unprocessed. For example, when we transform the continuous feature “capital-gain” and “capital-loss” into categorical features and use the one-hot encoding scheme to convert the generated categorical features into binary features, XGBoost performs worse

than when we adopt the continuous features for prediction directly. Therefore, for XGBoost data pre-processing, the continuous attributes are more effective when left unprocessed. It is better to focus on the data transformation and re-categorization of categorical attributes when training the model.

SelectKBest: k=11

	k=10	**k=11**	**k=12**
mean_train_AUC	0.94120243	0.94301379	0.94897891
mean_test_AUC	0.9252771	0.9249419	0.92511548
mean_train_Accuracy	0.88177298	0.88387954	0.89046768
mean_test_Accuracy	0.86749214	0.86695977	0.86659102

We find that k=11 and k=12 return the same result. When k=11, the cross-validation returns the highest mean test accuracy, while k=12 returns highest mean test AUC. The 11 selected attributes are shown in Figure 17.

Comparison between XGBoost and other models
XGBoost vs Gradient Boost

- XGBoost uses sparse matrices with sparsity aware algorithms.
- XGBoost uses data structures for better processor cache utilization, which makes it faster.
- XGBoost supports multicore processing better, which reduces overall training time.
- XGBoost supports DART, the dropout regularization for regression trees.
- During testing, it is found that XGBoost performs better than Gradient Boost.

XGBoost vs CatBoost

- CatBoost uses a combination of one-hot encoding as well as an advanced mean encoding. For features with a low number of categories, it uses one-hot encoding. For the remaining categorical columns, CatBoost uses an encoding method similar to mean encoding with a mechanism to reduce overfitting. XGBoost doesn't have an inbuilt method to handle categorical features [4]. In our case, the number of categories for each column is not too big, therefore, this feature is not very useful.
- CatBoost has two modes to handle missing values, "Min" and "Max". In "Min" mode, missing values are treated as the minimum value for a feature. In such a way, it is assured that a split that separates missing values from all other values is considered when choosing the splits. "Max" works exactly the same as "Min", but only with maximum values. In XGBoost, the missing values will be allocated to the side that decreases the loss in each split.
- CatBoost grows a balanced tree. In each level, the feature-split pair that leads to the lowest loss is selected. XGboost splits up to the max_depth and then starts pruning the tree backwards. It removes the splits beyond which there is no positive gain.
- CatBoost applies Minimal Variance Sampling (MVS). MVS is a weighted sampling version, comparing with Stochastic Gradient Boosting. XGBoost does not use weighted sampling techniques, which leads to a slower splitting process compared to MVS.
- During testing, it is found that XGBoost and CatBoost perform similarly well.

Other attempted methods of merging categorical data
The below category merging methods do not help in improving our prediction accuracy and F1 score:

- Merging some of the occupations. E.g. Merge "Craft-rep", "Handlers-cleaners", "Farming-Fishing" as "Manual Occupation", etc.
- Creating a new column by multiplying age and hours-per-week, which are closely correlated.

Reference

[1] Income Receipt Per Capita in 1994 : https://www.nationmaster.com/country-info/stats/Economy/Income-receipts/BoP/Current-US$-per-capita#1994

[2] GDP Per Capita in 1994 : https://www.nationmaster.com/country-info/stats/Economy/GDP-per-capita#1994)

[3] V.Morde, V.A.Setty (2019, April 8). XGBoost Algorithm. Retrieved from https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

[4] Aviv Nahon (Dec 30, 2019). XGBoost, LightGBM or CatBoost — which boosting algorithm should I use? https://medium.com/riskified-technology/xgboost-lightgbm-or-catboost-which-boosting-algorithm-should-i-use-e7fda7bb36bc