CS273P Project 2: Predict Income (Classification)

TEAM MEMBERS AND WORK DISTRIBUTION

Team Name: EL PSY CONGROO (Final Kaggle Score: 0.91216)

Team Members	Student ID	Responsibilities
Po-Ren Pan	10202449	Random Forests, Stacked Random Forests
Chun-Kai Chan	41525034	K-Nearest Neighbors, Random K-Nearest Neighbors
Yi-Dar Liu	32845675	Exploratory Data Analysis, Logistic Regression, Simple Ensemble, Writeup Consolidation

First, we discussed the problems to solve, and then each was responsible for coding, data analysis, model selection, and writing of a specific technique. Afterwards we shared ideas and results to find ways to improve model performance.

INTRODUCTION

In this project we employ a 1994 Census bureau dataset of population survey to predict a person's income. The dataset contains 20,000 training data points and 10,000 test data points. The target value is a boolean variable Y, which is 1 if the income of a person exceeds $50K per year and 0 otherwise. The features consist of 13 demographic characteristics, such as age, education, and occupation. Scoring of predictions is done using AUC, the area under the ROC (receiver-operator characteristic) curve, by Kaggle. It is a number between 0.5 and 1, and higher AUC score indicates that the model has better prediction accuracy. We would like to tackle the following three problems:

1. Technique selection: Which technique works best for the dataset?
2. Data manipulation (including selection and transformation): Which data should be used to train our model? Will transformed data produce better results?
3. Model selection (including feature and hyperparameter selection): Which model has the largest predictive power?

Each problem is discussed in detail below.

Technique selection

In this project, we explore three kinds of machine learning (ML) techniques: Random Forests (RF), K-Nearest Neighbors (KNN), and Logistic Regression (LR). Since the techniques are based on different assumptions and rationales (summarized in the following table), their predictions should be uncorrelated. Therefore, they should be suitable for the ensemble method.

	Tool	Assumption and Rationale	Advantage and Disadvantage
RF	sklearn	Assumption: Target value is based on a series of branching on feature thresholds (decision tree) Rationale: Learn a collection of models and combine them to get a more accurate and stable prediction	Advantage: • Use bagging and randomization to reduce memorization effect and variance • Computationally fast Disadvantage: • Randomly sample data for training • It is hard to interpret because RF is an ensemble algorithm of decision trees
KNN	sklearn	Assumption: Close features lead to close target values Rationale: Use the average value of k nearest neighbors as the output	Advantage: • Predicted value will always be within the domain of target value, so it will always be between 0 and 1. Disadvantage: • Do not work well in high dimensions • Computing distances in large data sets is slow (e.g. 500 runs take 30 minutes)
LR	Minitab	Assumption: There is a linear relationship between features and the natural logarithm of the odds ratio. Rationale: Maximize the likelihood of the dataset	Disadvantage: • Have to deal with the quasi-complete separation issue, which may make the model fit less perfectly • Need to create dummy variables for categorical variables • Coefficients are sensitive to data values, especially outliers • If the relationship is nonlinear, the predictive power is low.

Data manipulation (including selection and transformation)

The following two ways can be used to select training data, and the optimal way may depend on the technique used.

	Advantage	Disadvantage
Use the whole data set	Use the full information	• Data may contain errors, which will affect model performance • Computation is slow
Eliminate special cases before training	Models have better in-sample performance	• Definition of "special case" is subjective • May fail to predict special cases

In this project, features include both numerical and categorical variables. Numerical variables can be transformed into categorical variables by partitioning or new features by creating their derivatives. Categorical variables can be modified by merging classes or assigning numerical values to them. We would like to know if transformed data generate better results.

Model selection (including feature and hyperparameter selection)

The following two options can be used to select features, and the optimal option may depend on the technique used.

	Advantage	Disadvantage
Include all features	Use the all information	• Features may not be independent • Computation is slow • Some features may have no influence on target value and thus hurt a model's out-of-sample performance (overfitting)
Include only a subset of features	Can focus on the most important features when optimizing parameters	• Judging feature importance is subjective • Other features may have subtle influence on target value

Hyperparameter selection will be discussed in the following individual technique sections.
Notation: [i] means ith reference in **APPENDIX I**.

RANDOM FOREST (RF)

RF classifier generates several decision trees and averages over them to get the final result. Because decision boundary in a tree must be a number, we have to transform categorical variables into numerical variables. Two approaches are tried. One approach is called "one-hot encoding." It generates one boolean column for each category, and only one of these columns could take on the value 1 for each data point. This approach transforms a categorical variable into k boolean variables, where k is the number of classes of the categorical variable. The other approach is assigning each category a nominal value like 1, 2, 3, ... Let's call it "value assignment." This approach does not change the number of features.

In addition, we also transform the following features in expectation of better model performance:

- age: **APPENDIX II Figure A5** shows that income increases with age first, peaks at age = 50, and then gradually deceases. Therefore, we reformulate age into a new feature, |age-50|.
- native.country: In order to simplify our model, the feature is recategorized into various regions (North America, South America, Western Europe, Eastern Europe, East Asia, and South Asia).
- sex and relationship: Because "Husband" and "Wife" in "relationship" are highly correlated with "Male" and "Female" in "sex," respectively, the two variables are merged into one feature with disjoint classes. Please refer to **APPENDIX III Table A9** for details.

Besides using the whole dataset, we also try eliminating data which apparently have specific target values before training our RF model. The prefiltering process is similar to that of LR, which is described in **APPENDIX IV Table A10**.

After preprocessing the data, we tune the following 4 hyperparameters to find the optimal model:

1. n_estimators: number of trees
2. max_depth: maximum depth of each tree
3. max_features: maximum number of features used in each tree
4. min_samples_leaf: minimum number of data per leaf

An issue in RF is that each tree predicts 0 or 1 for each data point, but AUC is calculated using "soft prediction," which is the confidence level of Y = 1. In order to get an accurate estimate of the value, we need to select a sufficiently large n_estimators. The other three hyperparameters are used to control model complexity.

We perform 5-fold cross-validation using different hyperparameter settings. Each iteration out of 5 generates an AUC score on the validation data ("validation AUC") and another on the training data ("training AUC"). We take average of the validation/training AUC scores and select the hyperparameters with the highest average validation AUC. We find that if the categorical variables are transformed using one-hot encoding, the optimal max_features is exactly $\sqrt{\text{number of features}}$. However, if the other approach is used (value assignment), because the total number of variables is only 13, setting max_features = $\sqrt{13}$ = 4 may make the model underfitted. Therefore, in this case, we put all the features into the model.

Table R1 shows the results of the optimized RF models. In terms of Kaggle score, Model 1 (one-hot encoding and no prefiltering) outperforms Model 2 (value assignment) and Model 3 (with prefiltering). It has the following two implications:

1. Although one-hot encoding expands the number of features, it is still more suitable for handling categorical variables in RF than value assignment.
2. There are exceptions in the real world (surely). For example, in the test data, we find that a Mexican with only preschool education can earn $41,310 from capital gain. If we subjectively assign Y = 0 to this data point based on his education, we may make an error. Therefore, we still have to consider various factors to make a well-rounded prediction.

Table R1: RF Models and Results

Model	Approach of Handling Categorical Variables	Prefiltering	Hyperparameters*	Average Training AUC	Average Validation AUC	Kaggle Score
1	One-hot encoding	No	(50, 50, 8, 4)	0.88532	0.85313	**0.88578**
2	Value assignment	No	(50, 8, 13, 3)	0.86397	0.85415	0.87869
3	One-hot encoding	Yes	(50, 50, 8, 4)	0.88561	0.85321	0.88305

*A tuple in this column corresponds to (n_estimators, max_depth, max_features, min_samples_leaf)

K-NEAREST NEIGHBORS (KNN)

KNN makes predictions based on the nearest k data points. In order to measure distances among data points, all categorical values have to be transformed into numerical values. The way of transformation can make a huge difference in prediction. Intuitively, close data points should have similar properties. Take the categorical variable "education" for an instance, data points of "preschool" should be closer to those of "1st-4th" than those of "Doctorate." If we assign numbers to categories in a wrong way, say "Preschool" is closer to "Doctorate," the nearest k data points of a person with preschool education may contain data of "Doctorate," resulting in an incorrect prediction. Therefore, categories that have similar characteristics should be assigned adjacent numbers. We measure categories' similarity by P(Y=1|X=c), i.e. the percentage of Y = 1 within a given class c of a categorical variable X. Once the percentages are calculated for a given categorical variable, we assign numbers 1, 2, 3, ... to the classes based on the increasing order of P(Y=1|X=c). For example, P(Y=1|sex=Female) = 0.112 and P(Y=1|sex=Male) = 0.315, so we assign 1 to "Female" and 2 to "Male." If a class has special characteristics, it is assigned extraordinarily large/small numbers so that this class's data are far away from other classes. Take "occupation" as an example, "Priv-house-serv" always results in Y = 0 while "Armed-Forces" always results in Y = 1. Hence we assign the former with "-1000" and the latter with "1000." We isolate these data points from others so that the KNN model makes predictions mostly based on data points in the same class. After transforming categorical data into numerical data, we can use KNN to do the prediction.

First, the standard KNN is constructed. We use 5-fold cross-validation to get error rates for different k and select k which produces the lowest error rate. **Figure K1** shows that the optimal k is 20 and the error rate is 0.1484. Then we perform predictions for test data and obtain Kaggle score = 0.90135.

Improvement: Random K-Nearest Neighbors (RKNN)

RKNN [2] introduces a randomized method for KNN to drop unnecessary features and expects to get better prediction results as well as reduce calculation time. The randomized method contains 2 phases: (1) calculate the feature support and (2) select top important features. In these 2 phases, we set k = 20 based on the previous result of 5-fold cross-validation.

Phase 1: Feature Support

Feature support means the classification capability of a feature. To determine the *support* of a feature, we randomly select a subset of features, X_m, which has m features, out of totally p features. Then we use the KNN classifier to train the model and get the accuracy (measured in terms of error rate) for each X_m. The steps above are repeated r times. After all the iterations, each feature will appear in some KNN classifiers. The mean accuracy of these classifiers, called *support*, is a measure of the feature's relevance with the outcome. The lower support a feature has, the more it is relevant to the outcome.

Our RKNN model's parameter setting is: p (number of features) = 13, r (number of rounds) = 500, and m (number of features selected for each round) = $\sqrt{13}$ = 4. **Figure K2** shows the supports of all features. We see that "capital.gain" (x-axis = 9) has the lowest support (error rate), which indicates that it is an important feature in predicting Y. On the other hand, "native.country" (x-axis = 12) has the highest support (error rate), so it is very likely to be dropped in the next phase. The order of all features in terms of increasing supports is as follows: capital.gain, occupation, education.num, education, marital.status, capital.loss, relationship, hours.per.week, workclass, age, race, sex, and native.country.

Phase 2: Feature Selection
After getting the *supports* of all features, we carry out feature selection using a conservative approach described in [2]. We decrement the number of features used for prediction and get different error rates for different numbers of features. The optimal number of features should produce the lowest error rate. As **Figure K3** shows, when we select top 13 – 6 = 7 important features, the model has the best prediction ability with error rate = 0.147.

Result
After the 2 phases, we keep 7 features: capital.gain, occupation, education.num, education, marital.status, capital.loss, and relationship. Still, 5-fold cross-validation is used to calculate the optimal k = 26, which is shown in **Figure K4**. The training error rate is 0.1484 and the Kaggle score is 0.90336. RKNN successfully eliminates nearly half of features and gains a slightly lower error rate compared with the original KNN. Thus, it not only makes better prediction but also reduces calculation time.

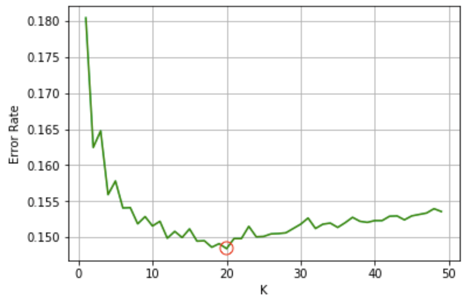
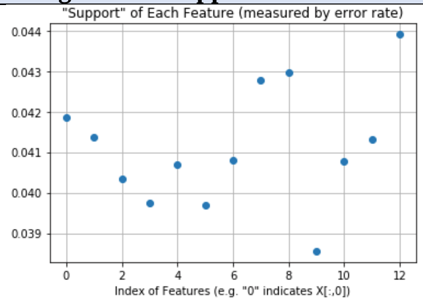
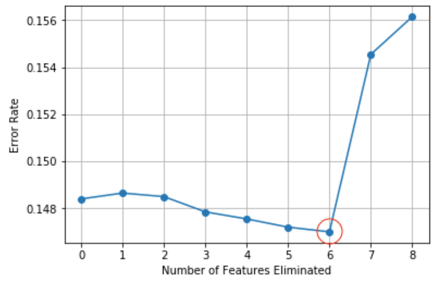
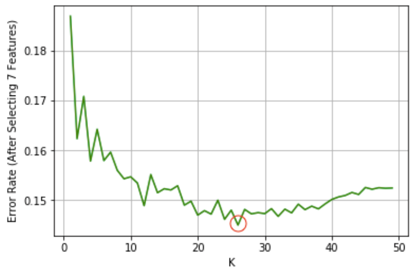
Figure K1: Cross-Validation Error as Function of k	**Figure K2: Supports of Features**
	

Figure K3: Cross-Validation Error as Function of Number of Features Eliminated	**Figure K4: Cross-Validation Error as Function of k after Feature Selection**
	

LOGISTIC REGRESSION (LR)

Issues in LR: quasi-complete separation and dummy variable

When conducting LR, we face a serious issue: *quasi-complete separation*. It means that if there exists a condition which can perfectly separate the sample into Y = 0 and Y = 1, then maximum likelihood estimates (MLE) of the regression coefficients do not exist. It implies that every combination of categorical values needs to have at least one data point. However, given the data set we have, it is very hard to circumvent the problem. One reason is that the data set primarily consists of categorical variables. Another reason is that some combinations are unlikely per se in the real world. For example, a foreigner is impossible to work in the US government. Nonetheless, considering LR's low correlation with other classification techniques, we still decide to include it in our model set.

The quasi-complete separation problem also has the following implications:

1. We have to include as many data points as possible in our training data. It makes difficult to perform cross-validation, which reduces sample size and may expose us to quasi-complete separation problems.

2. We have to merge categories of zero or small sample sizes in hope that every combination of categorical values has at least one observation. It is involved with many subjective judgments.
3. Some variables cannot be in the same model if no data point exists in some combinations of their values. For example, even after recategorization, we find that it is impossible for workclass, education, and marital.status to co-exist in the same LR model. Since statistical analysis shows that they have significant impact on Y, we still desire to include all of them in our model. Luckily, because LR is nonlinear, we can generate multiple models containing different variables and combine them using ensemble methods. In order to differentiate LR models containing subsets of features from ensemble models which combine these LR models, we call the former "submodels" and the latter "models" in this section.

Our model building process follows the principles stated above.

Another issue related to categorical variables is the use of *dummy variables*. In order to represent the relationship between Y and a categorical variable, for each class other than the first one, we must create a boolean variable whose value is 1 if the characteristic is present and 0 otherwise. The variable is called a *dummy variable*. A categorical variable of k classes produces k – 1 dummy variables. Therefore, the more classes a categorical variable has, the more dummy variables it creates, and the more complicated the resulting model will be. Merging categories not only helps us conquer the quasi-complete separation problem but also simplifies the LR model.

Data manipulation and collinearity elimination

Our exploratory data analysis includes:

- In order to find patterns between variables and Y, we calculate the percentage of Y = 1 within each class of categorical variables. We also partition numerical variables into fixed intervals and calculate the percentage of Y = 1 within each interval.
- In order to investigate quasi-complete separation, we cross-tabulate independent variables to check missing entries.

Selected results are displayed in **APPENDIX II**. Using the results, we manipulate data based on the following rules:

- If percentage of Y = 1 is not proportional to a numerical variable, then we transform or categorize it, e.g. age, hours.per.week.
- If a class is almost certain to have Y = 0 or 1 (i.e. complete separation), then we delete data of the class from training sample and directly assign the specific label to the class when performing predictions.
- If some categories of small sample sizes are alike in the sense that they have similar percentages of Y = 1 and at least one of them has a null entry in a cross-table, we merge them.

In addition, collinear variables are eliminated. We use *variance inflationary factor* (VIF) to evaluate the collinearity among the variables. If a variable can be explained by other variables, then it will have a large VIF. Statistical package Minitab is used to calculate VIF. For numerical variables, we refer to Snee [1] and drop variables with VIFs more than 5. In the case of categorical variables, if most of the corresponding dummy variables have VIFs more than 5, then we drop the categorical variable.

APPENDIX IV Table A10 details our data manipulation process for each variable. After dropping collinear variables (education.num, relationship, and race), our feature pool for LR is as follows:

Numerical Variable	age (A), capital.gain (CG), capital.loss (CL), hours.per.week (H)
Categorical Variable	workclass (W), education (E), marital.status (M), occupation (O), sex (S), capital.loss (CL), hours.per.week (H), nation.country (N)

Please note that capital.loss and hours.per.week can be either numerical or categorical. (Please refer to **APPENDIX IV Table A10** for transformation details.) Treating them as numerical or categorical depends on the modeling method we use.

Method 1: Statistical Inference Modelling

Minitab is used to perform logistic regression. As we mentioned, despite their statistical significance, it is impossible for workclass, education, and marital.status to co-exist in the same LR model. Therefore, we have to create at least three submodels, each containing one of the three variables. First, we treat hours.per.week as a numerical variable. We find that in this case, sex and capital.loss are not statistically significant (p-values > 0.2), so we drop the two variables from the feature pool. The initial model is shown in **Table L1**.

All of the three submodels include age, capital.gain, and hours.per.week because numerical variables are less likely to cause quasi-complete separation problems. Please note that the categorical variables in the three submodels are mutually exclusive. Let's call such a model an "exclusive" one. Each submodel predicts a value for each test data point. If submodels have similar training error rates, we take unweighted average of the three predictions to obtain the final result; otherwise, we take weighted average based on submodels' training accuracy rate, which is 1 minus training error rate. The model's Kaggle score is 0.87059. Another method is trying to add as many variables as possible to each submodel until quasi-complete separation happens. Let's call such a model a "greedy" one. Unfortunately, the greedy model does not improve Kaggle score, which shows the model is overfitted.

Next, we treat hours.per.week as a categorical variable. Strangely, sex and capital.loss become significant in this case. Likewise, exclusive and greedy methods are tried. In the exclusive case, we try to allocate uncorrelated variables in the same submodel and make submodels have almost equal counts of variables (one dummy variable counts as one variable). The resulting model is shown in **Table L2**.

Table L1: Exclusive Model when hours.per.week is Numerical

Submodel	Numerical Variable	Categorical Variable
1	A, CG, H	W, O
2	A, CG, H	E
3	A, CG, H	N, M

Table L2: Exclusive Model when hours.per.week is Categorical

Submodel	Numerical Variable	Categorical Variable
1	A, CG	W, S, CL
2	A, CG	E, H
3	A, CG	M, O, N

This time, we try a new ensemble method. If a submodel is confident that a person can earn more than $50K a year, then we believe in the model and directly assign the model's predicted value to the data point; otherwise, we still take unweighted average of the three predictions as the final result. The rationale behind the method is that if there is a key factor which can determine a person's income, then other factors may not be so important. For example, a PhD should earn a lot regardless of his or her workclass/occupation. Let's call the method "strong advocate." In the exclusive case, the unweighted average method's Kaggle score is 0.88487, and the strong advocate method's Kaggle score is 0.87866; in the greedy case, the former becomes 0.85694, and the latter 0.85680. Apparently, exclusive models are better than greedy models, and unweighted average method is better than strong advocate method (in terms of Kaggle score). In short, "simpler is better." The models' in-sample and out-of-sample performances are summarized in **Table L3** and **Figure L4** (Model 1-6).

Table L3: Summary of LR Models (Method 1 in light green, Method 2 in light red)

Model ID	Number of Submodels	Numerical Variable	Categorical Variable	Model Type*	Ensemble Method**	Training Error Rate	Kaggle Score
1	3	A, CG, H	W, O, E, N, M	Exclusive	Unweighted Average	0.18225	0.87059
2	3	A, CG, H	W, O, E, N, M	Greedy	Unweighted Average	0.17183	0.87023
3	3	A, CG	W, O, E, N, M +H, S, CL	Exclusive	Unweighted Average	0.17185	**0.88487**
4	3	A, CG	W, O, E, N, M +H, S, CL	Exclusive	Strong Advocate	0.17960	0.87866
5	3	A, CG	W, O, E, N, M +H, S, CL	Greedy	Unweighted Average	0.17180	0.85694
6	3	A, CG	W, O, E, N, M +H, S, CL	Greedy	Strong Advocate	0.17165	0.85680
7	2	A, CG, CL	W, E, O	Exclusive	Weighted Average	0.21305	0.83616
8	2	A, CG, CL	W, E, O	Exclusive	Strong Advocate	0.20215	0.83008
9	2	A, CG, CL	W, E, O	Greedy	Unweighted Average	0.18725	0.83913
10	2	A, CG, CL	W, E, O	Greedy	Strong Advocate	0.18805	0.83730

*Exclusive: categorical variables in submodels are mutually exclusive; Greedy: categorical variables in submodels are overlapping
**Unweighted Average: take unweighted average of submodels' predictions; Weighted Average: weight is based on training accuracy rate; Strong Advocate: if a submodel predicts a likelihood of more than 50%, use the one; otherwise, take unweighted average

Figure L4: Model Performance as Function of Complexity (x-axis represents Model ID in the first column of Table L3)

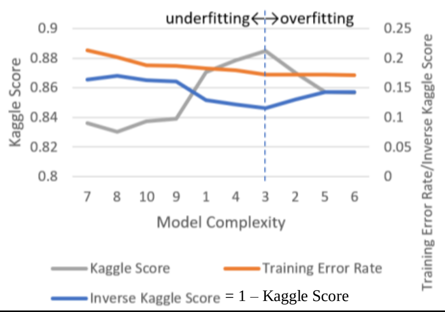


Table L5: Inferred Cause-Effect Relationships among Variables (cause=>effect)

Inherent Factor	Direct Factor	Hidden Factor	Result	Ramification
	age=>	work experience=>		
	workclass=>	organization=> hours.per.week		
race=> sex=> native.country=>	education=>	skill level=> occupation=>	Y=>	marital.status relationship
	occupation=>	job type=>		
	capital.gain=>	extra income=>		
	capital.loss=>	income killer=>		

Method 2: Logical Inference Modelling

Inspired by Method 1's results, we would like to simplify the model further. Our hypothesis is that given direct factors which influence income (e.g. workclass, occupation), inherent factors (e.g. sex, nation.country) should become irrelevant. For example, the previous statistical analysis shows that people from Latin America earn significantly less than people from other continents. We suspect that it is because most Latinos do not receive adequate education and (maybe thus) do unskilled jobs. Results in **APPENDIX II** corroborate our conjecture. **Table A1** shows that 46% of Latinos only receive primary education and 58% perform "Other-service." Besides, only 3% studied in the graduate school and only 4% and 5% work as managers and professionals, respectively. **Table A2** suggests that "Doctorate" (74%), "Masters" (56%), and "Prof-school" (77%) are more likely than "1th-12th" (6%) to earn more than $50K, and "Exec-managerial" (49%) and "Prof-specialty" (44%) are more likely than "Other-service" (13%) to obtain higher income.

Moreover, given workclass, we do not think hours.per.week should impact income. As **APPENDIX II Table A3** shows, more than half of employees in corporations and governments work around 40 hours per week, while work hours of self-employed people exhibit larger variance. Also, despite the strong correlation between marital.status and Y (please refer to **APPENDIX II Table A4**), we do not think marital.status should be an "explanatory" variable for Y. It does not make sense to infer that "because one is married, one should earn higher." Instead, financially stable people should be more likely to woo partners and get married. Therefore, we decide to drop hours.per.week and marital.status from the feature pool as well.

Table L5 in page 6 summarizes our inferences so far. Please note that education is a direct factor of both income and occupation. **APPENDIX II Table A2** shows that well-educated people (Professional School, Master, PhD) are more likely to assume professional positions as well as earn more than their peers working in similar occupations. To conclude, based on the logical inference, we downsize the feature pool to only 6 variables: age, workclass, education, occupation, capital.gain, and capital.loss. Please note that capital.loss is treated as a numerical variable in this method.

As with Method 1, both exclusive and greedy schemes are tried. Because marital.status is dropped from the model, we can use only two submodels in this modelling method. Their training error rates and Kaggle scores are summarized in **Table L3** and **Figure L4** (Model 7-10) in page 6. Unlike Method 1, greedy models demonstrate better in-sample and out-of-sample performances, which suggests that the simplified model is underfitted. Unfortunately, it seems that inherent factors may still affect a person's income.

Our LR results reveal the following insights:

- Both inherent factors and external factors influence Y, as shown by the fact that statistical inference models perform better than logical inference models.
- No single factor or small subset of factors can solely determine a person's income, as shown by the fact that unweighted/weighted average method outperforms strong advocate method (in terms of Kaggle score).
- When adequate features are selected for modelling, simple, less correlated submodels (exclusive) generate better ensemble performance than complicated, overlapping submodels (greedy).

Figure L4 in page 6 shows that among all 10 models, Model 3 (hours.per.week as categorical, exclusive, unweighted average) has the best test performance. Therefore, we choose Model 3 as the final LR model used to create (the second layer) ensembles with other techniques (i.e. RF and KNN).

ENSEMBLE GENERATION

Finally, we combine the three predictors and hope to get better results. The average method and the stacking method are chosen as our ensemble algorithms. The average method is straightforward. That is, we average the predicted Y values as our final result. The stacking method is a two-step training scheme. First, we train our learners ("base learner") and generate predictions $\hat{Y}_{tr}$ and $\hat{Y}_{te}$ for training data and test data, respectively. We use $\hat{Y}_{tr}$ as training data of another model ("stacked learner"), and then use $\hat{Y}_{te}$ as input in the model to predict the final result. Random Forest is used to stack our base learners. Their results are shown below.

Method 1: Unweighted Average of RF, RKNN, and LR

Interestingly, despite some variation among our predicted values, the unweighted average method (Kaggle score = 0.91216) outperforms any single model. It demonstrates the synergy of independent models.

Method 2: Weighted Average of RF, RKNN, and LR

We tried using Logistic Regression to stack our base learners, but the deviance statistic, which determines whether or not the current model provides a good fit to the data, shows that the resulting model is not a good-fitting one. Consequently, we turn to take weighted average of the predictions based on the base learners' training accuracy rates. The weights for RF, KNN, and LR are: 0.34407, 0.33414, and 0.32179. Unfortunately, this method does not improve ensemble performance (Kaggle score = 0.91206).

<u>Method 3: Stack RF, RKNN, and LR using Random Forest</u>

In the first approach of stacking, we only choose the best learned model from each technique. As with the base RF model, we use 5-fold cross-validation to obtain the optimal hyperparameters based on average validation AUC score. **Table E1** shows the resulting stacked model's hyperparameters and results, which are much better than those of individual models. However, its Kaggle score is worse. In our view, it may be because there are too few features (three base learners' predictions) in the stacked model, so we decide to stack more models in Method 4.

<u>Method 4: Stack 3 RFs, RKNN, and LR using Random Forest</u>

In Method 4, we stack 5 models, which are 3 RF models described in **Table R1** (Model 1-3) in page 3, RKNN, and LR. We still use 5-fold cross-validation to optimize hyperparameters. The resulting stacked model's hyperparameters and results are shown in **Table E1**. Despite better in-sample performance, Method 4 has a lower Kaggle score than Method 3. It suggests that the stacked model in Method 4 is overfitted.

We think a reason for stacked RF's poor performance may be that decision tree's objective is splitting data points to reduce entropy instead of estimating confidence level. Using the average of predicted classes as soft prediction may weaken RF's predictive power. Another possible explanation is that input variables are correlated real values within range [0, 1]. In this case, it may be difficult for RF to find proper thresholds to generate good trees.

Table E1: Stacked RF Models and Results

Method	Model	Hyperparameters*	Average Training AUC	Average Validation AUC	Kaggle Score
3	RF+RKNN+LR	(5, 100)	0.93476	0.89870	0.87230
4	3RFs+RKNN+LR	(5, 200)	0.95271	0.90675	0.85887

*A tuple in this column corresponds to (min_samples_leaf, n_estimators)

CONCLUSION

We conclude our analysis by answering the questions described in **INTRODUCTION**:

1. **<u>Which technique works best for the dataset?</u>**

   The following table summarizes performances of all models. As can be seen from the table, in terms of Kaggle score, RKNN is the best individual model, and Method 1 is the best ensemble method.

Individual	Kaggle Score	Ensemble	Kaggle Score
RF*	0.88578	Method 1 (Unweighted Average)	**0.91216**
KNN	0.90135	Method 2 (Weighted Average)	0.91206
RKNN	**0.90336**	Method 3 (Stack RF+RKNN+LR)	0.87230
LR*	0.88487	Method 4 (Stack 3RFs+RKNN+LR)	0.85887

   *Only list the best model's score

2. **<u>Which data should be used to train our model? Will transformed data produce better results?</u>**

   Two ways of data selection are tried: use the whole dataset and eliminate special cases. Due to the quasi-complete separation problem, LR can only choose the second option; on the contrary, RF and RKNN can choose either way. The results show that prefiltering leads to worse test performance. The reason is that there are always exceptions in the human society.

   Various data transformation schemes are utilized. RF and LR follow similar processes, such as splitting categorical variables into boolean/dummy variables, transforming age into |age-50|, merging classes, and combining/eliminating collinear variables. In order to calculate distances, KNN assigns numerical values to classes based on percentage of Y = 1. These efforts not only enable our learners to work (not doable otherwise) but also enhance their predictive power.

3. **<u>Which model has the largest predictive power?</u>**

   According to the results of KNN and LR, education, marital.status, occupation, capital.gain, and capital.loss appear to be the most important factors affecting a person's income.

   Each model's optimal hyperparameters are chosen as follows:

RF	one-hot encoding, no prefiltering, (n_estimators, max_depth, max_features, min_samples_leaf) = (50, 50, 8, 4)
KNN	7 features, k = 26, uniform weight on the nearest k data points
LR	3 submodels, hours.per.week as categorical, exclusive, unweighted average
Ensemble	unweighted average

APPENDIX I: REFERENCES

1. Snee, R. D., “Some Aspects of Nonorthogonal Data Analysis, Part I. Developing Prediction Equations,” Journal of Quality Technology, 5 (1973), 67–79.
2. Shengqiao L., E James H., Donald A A., “Random KNN Feature Selection – A Fast and Stable Alternative to Random Forests,” BMC Bioformatics 2011, 12:450.

APPENDIX II: SELECTED EXPLORATORY DATA ANALYSIS RESULTS

Table A1: Distribution of education and occupation given native.country

Count of >50K	Column Labels				
Row Labels	Europe	Latin America	Asia	North America	Grand Total
Bachelors	23%	7%	31%	17%	17%
Doctorate	2%	0%	6%	1%	1%
HS-grad	27%	26%	19%	33%	33%
Masters	6%	2%	11%	6%	6%
Prof-school	2%	1%	5%	2%	2%
Some-college	14%	15%	15%	23%	22%
1th-12th	14%	46%	8%	10%	12%
Assoc	12%	3%	5%	8%	8%
Grand Total	**100%**	**100%**	**100%**	**100%**	**100%**

Count of >50K	Column Labels				
Row Labels	Europe	Latin America	Asia	North America	Grand Total
Adm-clerical	11%	10%	15%	13%	12%
Craft-repair	15%	15%	9%	13%	13%
Exec-managerial	17%	4%	14%	14%	13%
Other-service	32%	58%	30%	34%	35%
Prof-specialty	19%	5%	22%	14%	14%
Sales	6%	8%	10%	12%	12%
Grand Total	**100%**	**100%**	**100%**	**100%**	**100%**

Table A2: Distribution of occupation given education & Percentage of Y = 1 within each pair of <occupation, education>

Count of >50K	Column Labels								
Row Labels	Bachelors	Doctorate	HS-grad	Masters	Prof-school	Some-college	1st-12th	Assoc	Grand Total
Adm-clerical	10%	1%	14%	5%	1%	19%	5%	15%	12%
Craft-repair	4%	1%	19%	1%	1%	13%	18%	16%	13%
Exec-managerial	27%	14%	8%	30%	9%	13%	3%	12%	13%
Other-service	14%	2%	45%	5%	3%	34%	64%	31%	35%
Prof-specialty	29%	80%	2%	51%	82%	6%	1%	13%	14%
Sales	16%	3%	11%	8%	3%	15%	9%	12%	12%
Grand Total	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**

Sum of >50K	Column Labels								
Row Labels	Bachelors	Doctorate	HS-grad	Masters	Prof-school	Some-college	1st-12th	Assoc	Grand Total
Adm-clerical	25%	50%	12%	36%	40%	12%	5%	12%	14%
Craft-repair	36%	50%	21%	40%	100%	28%	10%	30%	22%
Exec-managerial	57%	92%	33%	73%	82%	37%	27%	47%	49%
Other-service	30%	50%	12%	36%	50%	14%	4%	19%	13%
Prof-specialty	38%	73%	28%	49%	78%	25%	6%	30%	44%
Sales	45%	57%	19%	62%	70%	23%	6%	29%	28%
Grand Total	**42%**	**74%**	**16%**	**56%**	**77%**	**20%**	**6%**	**26%**	**25%**

Table A3: Distribution of hours.per.week (categorized) given workclass

Count of >50K	Column Labels											
Row Labels	0	1	2	3	4	5	6	7	8	9	10	Grand Total
Federal-gov	0%	1%	3%	2%	76%	12%	5%	1%	0%	0%	0%	100%
Local-gov	0%	2%	4%	5%	64%	15%	7%	2%	1%	0%	0%	100%
Private	0%	2%	7%	7%	58%	18%	7%	1%	1%	0%	0%	100%
Self-emp-inc	0%	0%	3%	3%	33%	30%	21%	6%	3%	0%	0%	100%
Self-emp-not-inc	1%	3%	7%	8%	33%	23%	15%	6%	2%	1%	1%	100%
State-gov	0%	4%	8%	3%	66%	12%	5%	2%	1%	0%	0%	100%
Grand Total	**0%**	**2%**	**7%**	**6%**	**56%**	**18%**	**8%**	**2%**	**1%**	**0%**	**0%**	**100%**

Table A4: Distribution of marital.status given Y = 1 & Percentage of Y = 1 within marital.status

marital.status:	
Row Labels	Sum of >50K
Divorced	6%
Married-AF-spouse	0%
Married-civ-spouse	85%
Married-spouse-absent	0%
Never-married	6%
Separated	1%
Widowed	1%
Grand Total	**100%**

marital.status:	
Count of >50K	
Row Labels	1
Divorced	11.1%
Married-AF-spouse	35.7%
Married-civ-spouse	45.7%
Married-spouse-absent	8.3%
Never-married	4.8%
Separated	7.0%
Widowed	10.3%
Grand Total	**25.1%**

Figure A5: Percentage of Y = 1 as Function of age

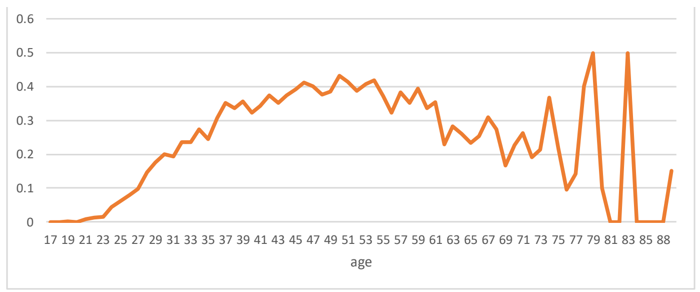


Figure A6: Percentage of Y = 1 as Function of capital.gain (K)

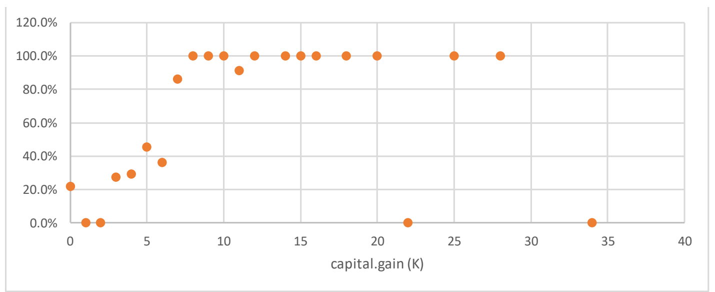


Figure A7: Scatterplot of capital.loss and Y

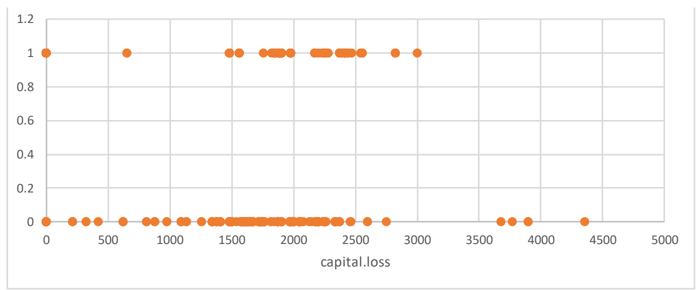
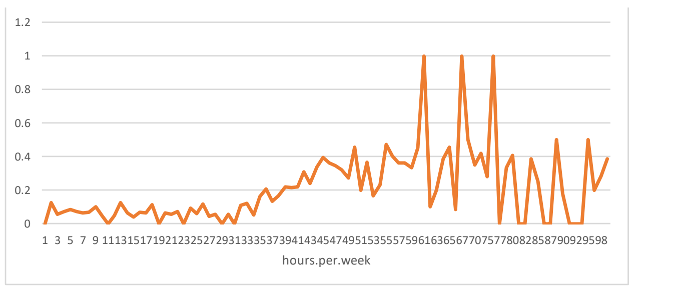


Figure A8: Percentage of Y = 1 as Function of hours.per.week



APPENDIX III: DATA MANIPULATION OF RANDOM FORESTS

Table A9: Combine sex and relationship into one Feature

sex	relationship	Combined Feature
Male	Not-in-family	Not-in-family-Male
Female	Not-in-family	Not-in-family-Female
Male	Own-child	Own-child-Male
Female	Own-child	Own-child-Female
Male	Unmarried	Unmarried-Male
Female	Unmarried	Unmarried-Female
Male	Other-relative	Other-relative-Male
Female	Other-relative	Other-relative-Female
	Husband	Husband
	Wife	Wife

APPENDIX IV: DATA MANIPULATION OF LOGISTIC REGRESSION

Table A10: Variable Manipulation in Logistic Regression (all mentioned figures are in APPENDIX II)			
Variable	Acronym	N/C*	Manipulation
age	A	N	**Figure A5** shows that income increases with age first, peaks at age = 50, and then gradually deceases. The fluctuation at late ages is due to small sample sizes. We find that Americans can work legally from age = 20 and retire at age = 65, so we merge age ≤ 20 into age = 20 and age ≥ 65 into age = 65. In order to make the variable proportional to Y, we transform age into |age-50|.
workclass	W	C	Because "Without-pay" is almost certain to have Y = 0, we remove such data from training data and directly assign Y = 0 to the class.
education	E	C	Because "Preschool" is almost certain to have Y = 0, we remove such data from training data and directly assign Y = 0 to the class. We think 1st-12th grades are similar, so we combine them into one category "1st-12th."
education.num	EN	C	Strictly speaking, the variable is a categorical variable. Although higher education number means better education level, the interval between numbers has no real meaning. For example, Bachelors = 13, Masters = 14, and Doctorate = 16, but in terms of difficulty, the difference between Masters and Doctorate is more than double the difference between Bachelors and Masters. Because the variable can be replaced by education, we do not use it in our LR model.
marital.status	M	C	We find that people with spouses are more likely to earn more, so we recategorize the classes into only two groups: "Single" and "Partnered."
occupation	O	C	Because "Priv-house-serv" is almost certain to have Y = 0, we remove such data from training data and directly assign Y = 0 to the class. On the contrary, "Armed-Forces" is almost certain to have Y = 1, so we remove such data from training data and directly assign Y = 1 to the class. Since unskilled jobs have smaller sample sizes and similar income characteristics, we merge "Handlers-cleaners", "Protective-serv", "Tech-support", "Transport-moving", and "Machine-op-inspct" into "Other-service." Indeed, we find that very few people with advanced degrees (Professional School, Masters, and Doctorate) perform these jobs.
relationship	RL	C	We find that the variable's dummy variables have particularly high VIFs. The reason is that the variable can be deduced by two other variables - marital.status and sex. A married male must be a husband, and a married female must be a wife. Thus, we drop the variable.
race	RC	C	We find the variable's dummy variables have high VIFs. In particular, it can be deduced by nation.country. For example, Taiwanese like us must be Asians. Because nation.country has relatively low VIFs, we drop race and keep nation.country.
sex	S	C	Unchanged
capital.gain	CG	N	**Figure A6** exhibits a positive relationship between percentage of Y = 1 and capital gain, so we should put the variable in our LR model.
capital.loss	CL	N/C	**Figure A7** reveals an interesting phenomenon: if capital.loss is between 1500 and 3000, Y is more likely to be 1. People who engage in investing tend to be the ones with spare money. However, if the loss is too much (> 3000), then their incomes could fall below $50K. Accordingly, we transform the variable into a new categorical variable, which is 1 if capital.loss is between 1500 and 3000 and 0 otherwise. Treating the variable as numerical or categorical depends on the modeling method we use.
hours.per.week	H	N/C	**Figure A8** shows that percentage of Y = 1 does not monotonically increase with the variable. When hours.per.week < 30, the percentage vacillates below 20%; after that, income rises with hours.per.week; when hours.per.week > 45, the relationship becomes volatile. Therefore, we try including the variable both as a numerical one and as a categorical one (partition = 10 hours) in our LR models.
native.country	N	C	Most non-US countries have small sample sizes. For example, only 25 Taiwanese are in the data set. We try various merging schemes: 1. Merge countries in the same region, e.g. East Asia, Western Europe, South America, etc. 2. Merge countries in the same continent, e.g. Asia, Europe, Latin America, etc. 3. Divide all data into only two groups: "Foreigner" and "US Citizen and Canadian" Finally, we choose the second scheme because it produces more meaningful results. Unfortunately, we find that people from Latin America have significantly lower income, while people from other continents do not differ very much.

*N: Numerical, C: Categorical