Georgia Institute Of Technology
05: Team Palantir Final Report

Edgar Yu (eyu46)
Fabian Michael Smits (fsmits3)
Jong Hyun Park (jpark928)
Kumar Gaurav (kgaurav7)
Pamela Siew Hua Tiong (ptiong3)
Prasad Patwardhan (ppatwardhan3)

1. Introduction

Having an accurate prediction of household income yields benefits to various organizations, from e-commerce companies to government institutions. Kibekbaev & Duman (2015) used logistic regression models to predict and detect credit card fraud. Sundsøy et al (2016) accurately predicted socioeconomic status from mobile phone datasets using deep learning methods. And Chiu (2002) utilized genetic algorithm-based approaches to predict customers likely to purchase insurance. Also, according to Lescaroux (2010), predicting whether a customer has an intention to purchase a car can be made based on their income level. However, previous studies had a number of limitations that hindered the optimization of models' predictive power. In our study, we aim to develop a machine learning model, which is an ensemble of different individual learners, for income prediction in context of a binary classification problem. Finally, various feature reduction techniques (two PCA techniques, SVD and Factor Analysis) were applied to the dataset to see if it can significantly increase the performance of the ensemble model.

2. Survey

There have been many attempts in developing accurate models that can predict income levels. Naga and Burgess (1997) defined permanent income depending on characteristics such as household composition, educational and occupational status and tried to predict it using a theoretical model. Kibekbaev and Duman (2015) developed a profit-based model using logistic regression by modifying the cost function of a maximum likelihood estimator and achieved a high level of the true positive rate when tested on credit card transactions in turkish banks. In a different study, Kibekbaev and Duman (2016) found that a combination of optimized linear and nonlinear techniques can significantly outperform a traditional linear regression technique. Kristina et al (2019) conducted an income prediction of retail pharmacy using gradient boosting and random forest. Zhu (2016) applied various techniques including logistic regression, neural networks, SVM, random forest and CART on census income dataset. Lazar (2004) applied PCA and SVM for income prediction. Zu et al (2019) improved traditional the SVM technique by including latent variables to predict consumption choices of low-income groups. However, the study tried to solve a multi-label classification problem with a small number of training instances, direct application of the technique to our case should be considered carefully. A novel approach in data preprocessing was done by Kamiran and Calders (2010) where they applied preferential sampling techniques in preprocessing to remove discriminatory attributes from a census income dataset. A popular classification algorithm, random forest classifier, was applied to the same dataset by Menji and Bekena (2017). Eldering (1998) proposed in his patent the way to recommend a targeted advertisement using correlations between consumer and advertisement characterization vectors, a method similar to collaborative filtering used in current recommendation system engines. Nirupam et al (2016) performed income prediction on a large US census income dataset by classifying economic classes using K-means and Naive Bayes classifiers to predict their label, producing

an accuracy of around 50%. García-Alonso, Carlos R., et al (2010) showed that an artificial neural networks model using product-unit as activation function was also effective in income prediction. Zadrozny (2019) showed that sample the selection bias problem that is common in the field of machine learning can affect some learners for survey data, while there are some learners that are resilient to this bias and a correction method can mitigate this problem.

3. Proposed Method

3.1 Description of Algorithms

Dataset. The 1994 U.S. Census dataset contains 48,842 rows that capture demographic information, such as age and income of census participants. In the original data preprocessing, numerical attributes will be standardized to have mean of zero and standard deviation of one. Categorical attributes of multiple classes will be treated through the following algorithm after combined into fewer categories based on feature characteristics: 1) If class frequency is concentrated to one value, it will be given a binary value, 2) else, if there seems to be an ordinal characteristic in the attribute, it will be given ordinal integer values and 3) else, it will be one-hot-encoded to multiple columns. For example, there were more than 40 countries in the native-country attribute and the majority of the values were concentrated in the US. Thus, according to the algorithm described above, the native-country attribute contained binary values (0 and 1) indicating whether the person is from the US or not.

Dimension Reduction. A quick analysis on the matrix shows that it is a rank deficient matrix, which implies that the columns in the matrix are linearly dependent on each other and hence, might affect the dimension reduction result. Here, we have applied several, mainly linear dimension reduction methods such as Principal Component Analysis (PCA), Truncated Singular Value Decomposition (SVD) and Factor Analysis (FA).

PCA employs SVD for dimension reduction and our team has employed 2 methods of finding the optimum number of features using gridsearch- the first being a fixed number of features (PCA manual) and the second method is using explained variance (PCA variance); extracting features that have hit a certain threshold. Overall, the PCA manual has slightly better results but PCA variance has better F1 score overall.

Truncated SVD. Similar to PCA, except that the mean of the matrix is not centralized before performing SVD. Unlike PCA, truncated SVD is an estimation, making it suitable for low rank or sparse matrix. For our dataset, Truncated SVD performed better than PCA.

Factor Analysis. It helps to detect underlying, hidden variables between the features. PCA is then used to factor those variables together by their common variance. Factor analysis assumes that there is a linear relationship as well as the absence of multicollinearity in the dataset. It is our worst performing model for this dataset.

Models. We will be running the following algorithms: Adaboost, decision tree, neural networks, KNN, polynomial and linear logistic regression, random forest and SVM. For each algorithm, optimal sets of hyperparameters were found using gridsearch. Our final ensemble model takes all previous models as an input and outputs the final prediction.

Neural Networks are implemented to capture non-linear relationships between input attributes and the target attribute. A neural network model with 3 hidden layers was built using a tower method where the number of nodes is halved after each layer was used.

K Nearest Neighbors. In the KNN model, the model gives output based on the k number of neighbors. Euclidean distance was used as a

measurement to cluster the points to the nearest group before predicting.

Random Forest, a method based on bootstrap aggregated classification trees, is well suited to capture nonlinear dependencies as well as interactions between feature variables. The method is employed as we expect such effects to be highly influential in our dataset.

Logistic Regression is used in combination with different regularization methods. Results from cross validation suggest that in our case L2 regularization provides the largest enhancement to the predictive accuracy.

Support Vector Machine is an often used benchmark model for binary classification problems. It is particularly known for its predictive accuracy in high dimensional data sets. Consequently, our study provides a textbook case for its employment.

Ensemble Models are used to combine outcomes from multiple models to improve the overall performance of our prediction models in predicting income. According to Juhi (2019) and Smolyakov (2017), utilizing ensemble models helps minimize bias and variance. This paper aims to utilize emerging machine learning algorithms in order to improve upon the results of previous studies that relied on individual models to predict income. For the ensemble model, four different cases can be implemented: uniform or non-uniform weight for individual models with soft or hard voting. For soft voting, the model sums up probability for each label in all individual models and returns the argmax. For hard voting, the model gives the majority vote from each individual model. Non-uniform weight was calculated by dividing the accuracy score of each model by the sum. In this study, soft voting with uniform weight was performed.

Impact. By utilizing ensemble models to predict income levels and providing an interactive user interface, this paper aims to allow users to evaluate the predictive power of ensemble models versus individual models. If successful, the paper will allow us to conclude that there is a statistically significant difference in these models and highlight the greater or lesser predictive power of ensemble models. In order to determine the success of ensemble models versus individual models, we will compare metrics such as precision, recall, and F1 scores of the models.

3.2 Innovations

In this study, two major innovations were introduced: 1) An ensemble model of individual machine learning models will be built to see if it achieves greater predictive power than that of the best individual learner. 2) An interactive visualization delineates distributions of values in each attribute and allows users to select a combination of individual learners to see changes in prediction.

3.3 Method logistics

Risks and payoffs. Since our data is survey data, class imbalance problems mentioned by Kamiran and Calders (2010) can be a risk in our study as the sampled data does not fully represent the population. In order to mitigate this risk, visualizations will be created to understand the distribution of the dataset and based on the visualization, data preprocessing will be conducted to minimize the risk caused by class imbalance. From these experiments, we gain a deeper insight into how the use of ensemble methods can affect a model's predictive power. Our interactive user interface will also allow users to see how each individual model contributes to the final ensemble model.

Costs. There is no cost in data gathering since it is public. However, it is found during model training and hyperparameter tuning that using a local computing power takes a lot of time. Thus, it is recommended to use cloud computing source such as AWS Sagemaker. It costs $0.549 per hour

to use ml.c5.2xlarge machine in Singapore area and it is expected to finish all processes within 1 to 2 hours, costing less than $2 for computational source. Github is used for version control and to store the raw and processed data at no cost.

4. Experiments and Evaluation

4.1 Observations

Table 1 summarizes the result of the final ensemble model with four different preprocessed dataset, which are original dataset, PCA variance, PCA manual and factor analyzed dataset. The values are represented in four significant figures below:

Dataset	**Accuracy**	**Precision**	**Recall**	**F1**
Original	0.8566	0.7695	0.5946	0.6708
PCA Variance	0.8313	0.6940	0.5608	0.6203
PCA Manual	0.8302	0.6899	0.5611	0.6189
Factor Analysis	0.7981	0.6485	0.3889	0.4862
Truncated SVD	0.8309	0.6925	0.5605	0.6196

Table 1: The result of the ensemble model with five different preprocessed datasets.

4.2 Discussion

Please refer to the appendix for full results.

For the original preprocessed dataset, Adaboost was the model that showed the highest accuracy, recall and F1 score among all the eight individual models. In precision, random forest displayed the best performance of 0.7727. It is notable that two algorithms using decision trees (random forest and adaboost) showed exceptional performances in all four evaluation metrics. The ensemble model, however, showed 0.8566 for accuracy, 0.7695 for precision, 0.5946 for recall and 0.6708 for f1 score. Unfortunately, the ensemble model was worse than both adaboost and random forest in all four evaluation metrics although it was initially expected that the sum of probabilities on each binary class label could lead to capturing missed predictions for the best individual model (adaboost). Thus, from this result, it could be concluded that for the original preprocessed dataset, the majority of the individual models produce similar predictions and when adaboost model wrongly predicts, few models give right answers and there are even more cases when the right prediction from adaboost is nullified by other models. This can be further supported by the poor results generated from factor analysis which also detects underlying relationships. We also initially thought that since the model was experiencing a high dimensionality problem, accuracy could be improved if we reduced its dimensions. Unfortunately, the results were no better than the original dataset, and some of them performed worse than the original dataset. From the above, we can conclude that either the experiments were incorrect or insufficient, or that all the features are equally important, and removing them was wrong.

4.3 Future experiment design

In this study, many experimental methods were simplified due to a number of limitations in time and computational resources. Since hyperparameter searching for certain algorithms such as SVM takes a long time, a sufficiently wide range of hyperparameters could not be explored. Seeing that tree-based algorithms like random forest and adaboost performed better , tree pruning could also be done to optimise results. Also, searching methods better than gridsearch, like bayesian search, could lead to discovering more optimal individual models. More diverse classifying algorithms such as Naive Bayes classifier or Xgboost could be added in the ensemble model. For the ensemble model, this study used a simple voting classifier that

takes in individual models and gives output that has the majority votes. For future studies, more advanced meta-modeling techniques such as stacking done by Michailidis (2017) could be tried to see if it makes a significant improvement from our simple voting ensemble model. Finally, if time and computational resources allow, non-linear dimension reduction methods such as kernal PCA, tensor decomposition and isomap could be experimented to see if it yields better model performances.

4.4 Design of interactive visualization.

In order to visualize our results, we will be using a python library called Streamlit. Streamlit is a library that specializes in interactive web based visualization. Our implementation of the library allows the end-user to tweak model parameters and see changes in the visualization and output of the model.

The first part of the interactive visualization shows descriptive statistics of the dataset we are using, where the distributions of the features can be seen. The sliders in the sidebar can be adjusted to filter the data according to age, gender, or race.

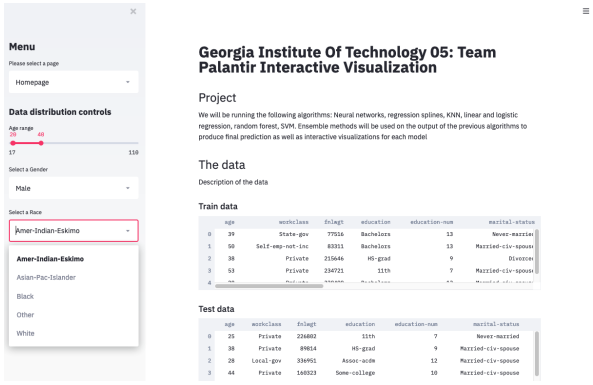

Figure 1: Descriptive statistics of our dataset

The second part of the interactive visualization shows the models that we examined in this project. Multiple models can be selected in the dropdown box and compared side by side using metrics like accuracy and precision.

To examine a specific model, select a model in the sidebar's dropdown, where you can then adjust the parameters for the model and execute the model with custom parameters to evaluate its performance.

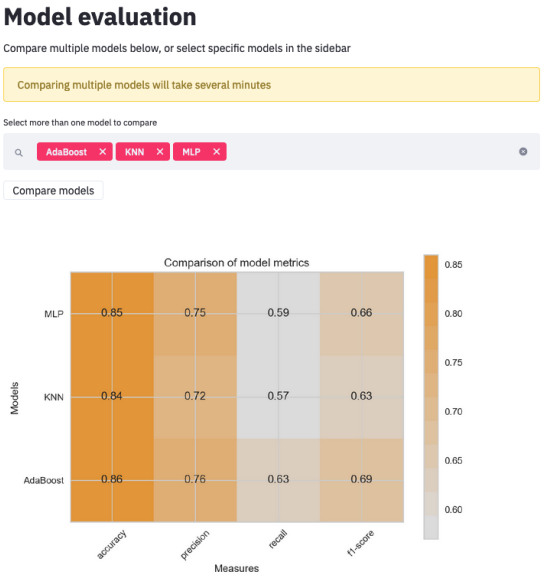

Figure 2: Comparison of multiple models and their relative performance, measured in accuracy, precision, recall, F1

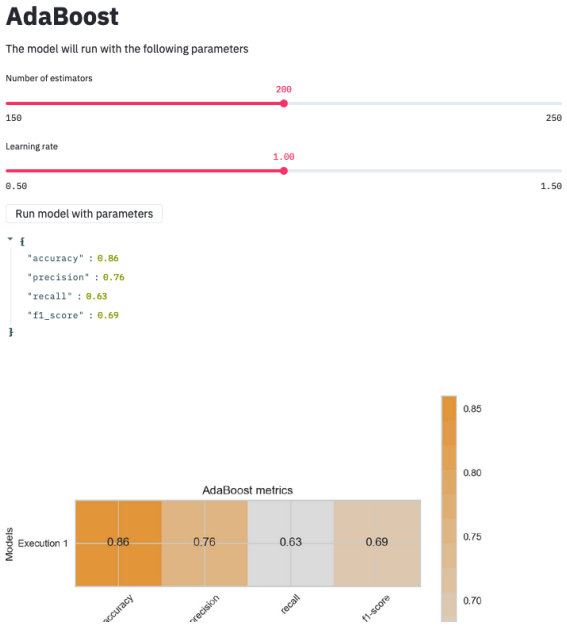

Figure 3: The parameters for a model can be adjusted by the user to measure model performance.

5. Team logistics

All team members have contributed similar amount of effort that made full use of their strengths.

Appendix (Page 1 of 3)

Model	Hyperparams	Accuracy	Precision	Recall	F1
AdaBoost	{'n_estimators':[200], 'learning_rate':[1]}	0.8606	0.7602	0.6322	0.6903
Decision Tree	{'criterion': ['entropy'], 'max_depth': [10], 'min_samples_split': [2], 'splitter': ['best']}	0.8525	0.7680	0.5727	0.6561
KNN	{'n_neighbors':[20] }	0.8388	0.7172	0.5676	0.6337
Logistic Regression	{"C":[0.8], "max_iter":[10000]}	0.8440	0.7252	0.5878	0.6494
Poly Logistic Regression	{"C":[0.6], "max_iter":[10000]}	0.8487	0.7211	0.6268	0.6706
Neural Network (MLP)	{'hidden_layer_sizes':[4], 'learning_rate_init':[0.001], 'activation':['tanh'], 'alpha':[0.001], 'batch_size':[256]}	0.8520	0.7542	0.5897	0.6619
Random Forest	{'max_depth': [15], 'criterion': ["entropy"], 'min_samples_split': [2], 'max_features': [10]}	0.8584	0.7735	0.5989	0.6751
SVM	{"coef0": [0.0], "kernel":['rbf']}	0.8509	0.7457	0.5968	0.6630
Ensemble (soft_uniform)		0.8566	0.7695	0.5946	0.6708

Table 2: Original Data results

Model	Hyperparams	Accuracy	Precision	Recall	F1
AdaBoost	{'learning_rate': 1, 'n_estimators': 200}	0.8252	0.6875	0.5286	0.5977
Decision Tree	{'criterion': 'entropy', 'max_depth': 10, 'min_samples_split': 4, 'splitter': 'best'}	0.7989	0.5996	0.5465	0.5718
KNN	{'n_neighbors': 20}	0.8228	0.6641	0.5638	0.6099
Logistic Regression	{'C': 0.6, 'max_iter': 10000}	0.8210	0.6786	0.5159	0.5862
Poly Logistic Regression	{'C': 0.6, 'max_iter': 10000}	0.7977	0.6015	0.5238	0.5600
Neural Network (MLP)	{'activation': 'relu', 'alpha': 0.001, 'batch_size': 256, 'hidden_layer_sizes': (32, 16, 8, 4), 'learning_rate_init': 0.001}	0.8272	0.6665	0.5935	0.6279
Random Forest	{'criterion': 'entropy', 'max_depth': 10, 'max_features': 10, 'min_samples_split': 2}	0.8335	0.6940	0.5768	0.6300
SVM	{'coef0': 0.0, 'kernel': 'rbf'}	0.8184	0.6741	0.5049	0.5773
Ensemble (soft_uniform)		0.8313	0.6940	0.5608	0.6203

Table 3: Dimension reduced by PCA(variance)

Appendix (Page 2 of 3)

Model	Hyperparams	Accuracy	Precision	Recall	F1
AdaBoost	{'learning_rate': 1, 'n_estimators': 200}	0.8240	0.7038	0.4895	0.5774
Decision Tree	{'criterion': 'entropy', 'max_depth': 10, 'min_samples_split': 4, 'splitter': 'best'}	0.7999	0.5977	0.5681	0.5825
KNN	{'n_neighbors': 20}	0.8209	0.6624	0.5530	0.6027
Logistic Regression	{'C': 0.8, 'max_iter': 10000}	0.8212	0.6800	0.5146	0.5858
Poly Logistic Regression	{'C': 1, 'max_iter': 10000}	0.7855	0.5693	0.5219	0.5446
Neural Network (MLP)	{'activation': 'relu', 'alpha': 0.001, 'batch_size': 512, 'hidden_layer_sizes': (64, 32, 16, 8), 'learning_rate_init': 0.01}	0.8170	0.6295	0.6200	0.6247
Random Forest	{'criterion': 'entropy', 'max_depth': 10, 'max_features': 10, 'min_samples_split': 2}	0.8337	0.6934	0.5795	0.6313
SVM	{'coef0': 0.0, 'kernel': 'rbf'}	0.8180	0.6757	0.4984	0.5737
Ensemble (soft_uniform)		0.8302	0.6899	0.5611	0.6189

Table 4: Dimension reduced by PCA(manual)

Model	Hyperparams	Accuracy	Precision	Recall	F1
AdaBoost	{'learning_rate': 1, 'n_estimators': 200}	0.8292	0.7023	0.5286	0.6032
Decision Tree	{'criterion': 'entropy', 'max_depth': 10, 'min_samples_split': 2, 'splitter': 'best'}	0.8019	0.6073	0.5476	0.5759
KNN	{'n_neighbors': 20}	0.8243	0.6703	0.5605	0.6105
Logistic Regression	{'C': 0.6, 'max_iter': 10000}	0.8223	0.6854	0.5116	0.5859
Poly Logistic Regression	{'C': 1, 'max_iter': 10000}	0.7986	0.5968	0.5559	0.5756
Neural Network (MLP)	{'activation': 'tanh', 'alpha': 0.001, 'batch_size': 512, 'hidden_layer_sizes': (32, 16, 8, 4), 'learning_rate_init': 0.01}	0.8215	0.6342	0.6462	0.6402
Random Forest	{'criterion': 'entropy', 'max_depth': 10, 'max_features': 10, 'min_samples_split': 2}	0.8335	0.6930	0.5789	0.6308
SVM	{'coef0': 0.0, 'kernel': 'rbf'}	0.8141	0.6592	0.5041	0.5713
Ensemble (soft_uniform)		0.8309	0.6925	0.5605	0.6196

Table 5: Dimension reduced by SVD

Appendix (Page 3 of 3)

Model	Hyperparams	Accuracy	Precision	Recall	F1
AdaBoost	{'learning_rate': 1, 'n_estimators': 200}	0.8292	0.7023	0.5286	0.6032
Decision Tree	{'criterion': 'entropy', 'max_depth': 10, 'min_samples_split' : 2, 'splitter': 'best'}	0.8019	0.6073	0.5476	0.5759
KNN	{'n_neighbors': 20}	0.8243	0.6703	0.5605	0.6105
Logistic Regression	{'C': 0.6, 'max_iter': 10000}	0.8223	0.6854	0.5116	0.5859
Poly Logistic Regression	{'C': 1, 'max_iter': 10000}	0.7986	0.5968	0.5559	0.5756
Neural Network (MLP)	{'activation': 'tanh', 'alpha': 0.001, 'batch_size': 512, 'hidden_layer_sizes ': (32, 16, 8, 4), 'learning_rate_init': 0.01}	0.8215	0.6342	0.6462	0.6402
Random Forest	{'criterion': 'entropy', 'max_depth': 10, 'max_features': 10, 'min_samples_split' : 2}	0.8335	0.6930	0.5789	0.6308
SVM	{'coef0': 0.0, 'kernel': 'rbf'}	0.8141	0.6592	0.5041	0.5713
Ensemble (soft_unifo rm)		0.8309	0.6925	0.5605	0.6196

Table 6: Dimension reduced by Factor Analysis

References

[1] Kibekbaev, Azamat, and Ekrem Duman. "Profit-based Logistic Regression: A case study in Credit Card Fraud Detection." The Fourth International Conference on Data Analytics. 2015.

[2] Sundsøy, Pål, et al. "Deep learning applied to mobile phone data for individual income classification." 2016 International Conference on Artificial Intelligence: Technologies and Applications. Atlantis Press, 2016.

[3] Chiu, Chaochang. "A case-based customer classification approach for direct marketing." Expert Systems with Applications 22.2 (2002): 163-168.

[4] "Juhi". "Simple Guide for Ensemble Learning Methods." Medium, Towards Data Science, towardsdatascience.com/simple-guide-for-ensemble-learning-methods-d87cc68705a2.

[5] Smolyakov, Vadim. "Ensemble Learning to Improve Machine Learning Results." Medium, Stats and Bots, 22 Aug. 2017, blog.statsbot.co/ensemble-learning-d1dcd548e936.

[6] Naga, Ramses H, and Robin Burgess. "Prediction and Determination of Household Permanent Income." 1997.

[7] Kibekbaev, Azamat, and Ekrem Duman. "Benchmarking Regression Algorithms for Income Prediction Modeling." Information Systems, vol. 61, 2016, pp. 40–52., doi:10.1016/j.is.2016.05.001.

[8] Pakhomova, Kristina, et al. "The Income Prediction Module of the Retail Store's Network." Applied Methods of Statistical Analysis, 2019.

[9] Zu, Xiaoqian, et al. "Prediction of Consumption Choices of Low-Income Groups in a Mixed-Income Community Using a Support Vector Machine Method." Sustainability, vol. 11, no. 14, 2019, p. 3981., doi:10.3390/su11143981.

[10] Kamiran, Faisal and Calders, Toon. "Classification with no Discrimination by Preferential Sampling." Proceedings of
the 19th machine learning conf. Belgium and the Netherlands, Leuven, Belgium, 2010.

[11] Lazar, A. "Income Prediction via Support Vector Machine." 2004 International Conference on Machine Learning and Applications, 2004. Proceedings., doi:10.1109/icmla.2004.1383506.

[12] Menji, Sisay and Bekena. "Using Decision Tree Classifier to Predict Income Levels", Munich Personal RePEc Archive, 2017.

[13] A. Eldering., Charles. Advertisement selection system supporting discretionary target market characteristics. US 6216129 B1, United States Patent and Trademark Office, Dec 3, 1998.

[14] Zhu, Haojun. Predicting Earning Potential Using the Adult Dataset, rstudio-pubs-static.s3.amazonaws.com/235617_51e06fa6c43b47d1b6daca2523b2f9e4.html.

[15] Nirupam, K.n., et al. "Data Analytics on Census Data to Predict the Income and Economic Hierarchy." International Journal of Data Analysis Techniques and Strategies, vol. 10, no. 3, 2018, p. 223., doi:10.1504/ijdats.2018.10015184..

[16] Lescaroux, François. “Car Ownership in Relation to Income Distribution and Consumers' Spending Decisions.” Journal of Transport Economics and Policy (JTEP). 44. 207-230. 10.2307/40600023, 2010.
[17] García-Alonso, Carlos R., et al. “Income Prediction in the Agrarian Sector Using Product Unit Neural Networks.” European Journal of Operational Research, vol. 204, no. 2, 2010, pp. 355–365., doi:10.1016/j.ejor.2009.09.033.
[18] Zadrozny, Bianca. “Learning and Evaluating Classifiers under Sample Selection Bias.” Twenty-First International Conference on Machine Learning - ICML 04, 2004, doi:10.1145/1015330.1015425.
[19] Michailidis, Marios. “StackNet Meta Modelling Framework.” StackNet, 2017. https://github.com/kaz-Anova/StackNet