JPMC Applied AI/ML Associate Role – Take-Home Project

Balaganesh Manikandan

Introduction

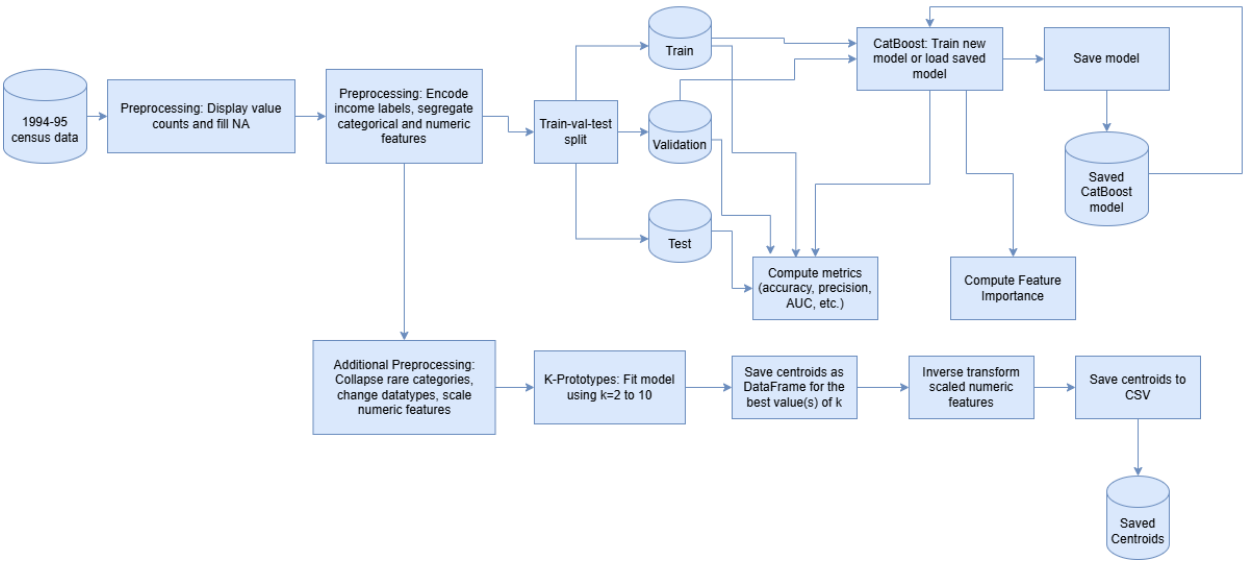
Income prediction and customer segmentation are two important business problems for a financial organization – these can help the company market different products to different groups, either based on income or other factors. For instance, a customer's income can influence the wealth management and investment strategies that he/she is most likely to look for. Using a weighted census dataset, this project has two main objectives:

1. Build and validate a model that can predict whether a particular group has an income of less than or greater than $50,000.
2. Build a customer segmentation model on the same data, and discuss how a client can use these models for marketing.

Data

This project uses a dataset containing weighted census data from 1994-95, containing 40 demographic and employment related variables, and a label for each observation indicating whether that population component had an income that is greater than or less than $50,000. Since each row of the data represents a section of the population rather than an individual, there is also a "weight" column representing the size of that section of the population. Overall, the dataset has about 200,000 rows, and as many as 32 of the 40 variables are categorical rather than numeric.

Architecture Diagram and Model Choice



The above figure shows the architecture diagram for both the models. For the classification task, I used **CatBoost** – a gradient boosted decision tree with support for categorical variables (which was very important since there were many categorical variables in the dataset). This differs from the scikit-learn tree-based models in that it supports datasets with *non-numeric categorical features without the need of any preprocessing*. The training and validation data were used to train the model, and both of these along with the test data were used to compute various metrics. I also computed the model feature importances which give an interpretation of which features influence the income predictions the most (interpretability is important for business decisions, which is why I chose a tree-based model).

For the segmentation task, K-Means clustering would usually be the obvious model choice, except that this dataset has many categorical features (rendering K-Means unsuitable for this task). One alternative that I considered was K-Medoids with Gower distance (one possible "distance" or "closeness" metric for mixed data), but this was not scalable since it required precomputing distances between all pairs of ~200,000 rows in the census dataset. The model I eventually used was **K-Prototypes**, which is also like K-Means, but to measure "distance" or "similarity" between data points, it uses the usual Euclidean distance for numeric features and counts the number of matching categories for categorical features. This method does not require precomputation, so it is scalable for the size of our dataset. The segmentation task required some additional preprocessing after the initial preprocessing code – this will be explained in a later section.

Initial Preprocessing and EDA

My first step in preprocessing was to display the value counts of all the columns and count the NA values in each column. Here were some of my observations about the data:

1. The columns "detailed industry recode", "detailed occupation recode", "own business or self employed", and "veterans benefits" were stored as integers, but from the column names, it can be inferred that these variables are intended to be categorical.
2. Some columns had the value "Not in universe" (such as "region of previous residence") indicating that some of the features were not applicable to some of the groups. (In this case, it could indicate that the region of previous residence was somewhere outside the US.)
3. None of the columns except "hispanic origin" had NA values (although some columns contained "?" values perhaps indicating that the value was unknown).

4. Some of the categorical columns had more than 20 different categories (such as "country of birth self"), which could cause issues in some of the training algorithms.

I then filled all the NA values in the "hispanic origin" column with "Do not know," which was another value assigned to some of the rows in the dataset. Next, I encoded the values in the "label" column – groups with income less than $50,000 were labeled 0 and the rest were labeled 1 (high income is the "positive" class). I also separated the features, the weights, and the target labels, and saved the indices of the categorical features (including the pseudo-numeric features I mentioned in point 1 above).

Classification – CatBoost

Before training the model, I performed a train-validation-test split of 60%-20%-20% on the data. For the classification task, as mentioned earlier, I used the CatBoost model with default parameters, and accuracy as a validation metric. I made sure I passed in the sample weights and the indices of categorical columns that I described earlier. Training on this dataset took 1000 iterations in about 7 minutes.

Using this model, I computed many different metrics on the train, validation, and test datasets, such as accuracy, ROC-AUC, precision, recall, and the confusion matrix. Here are the normalized confusion matrices for the model:

Train confusion matrix:

	Predicted low income	Predicted high income
Actual low income	0.92836341	0.00752964
Actual high income	0.02939348	0.03471347

Validation confusion matrix:

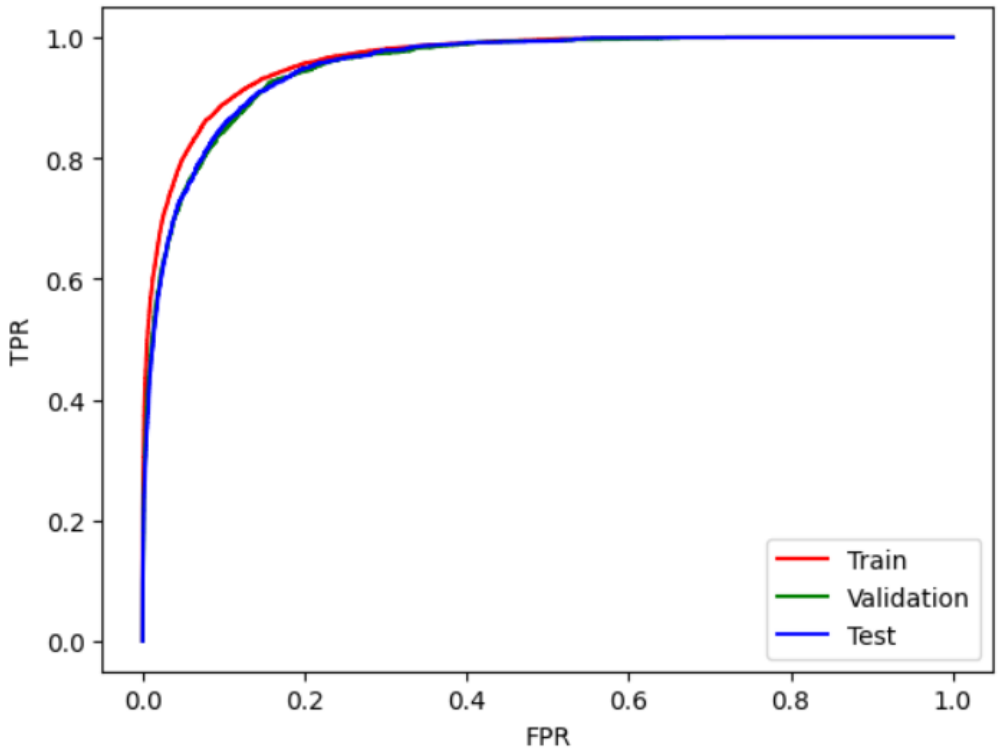
	Predicted low income	Predicted high income
Actual low income	0.92517945	0.01014331
Actual high income	0.03296674	0.0317105

Test confusion matrix:

	Predicted low income	Predicted high income
Actual low income	0.92498517	0.01176484
Actual high income	0.03238755	0.03086244

These confusion matrices tell us that the classes are imbalanced – over 90% of the data is in the low-income group. Thus, accuracy is not a good enough metric (in this case it was found to be about 95.6% on the test dataset). Here are some of the other test data metrics and their corresponding insights:

1. The test **ROC-AUC** was about **95.3%** - this means that on average, our classifier gave a "higher" prediction for a high-income data point than a low-income data point 95.3% of the time.
2. Here is the plot of the ROC curves for the three datasets:



The FPR is the percentage of times the classifier incorrectly predicts a low-income observation as high income among all the low-income observations in the dataset. The TPR (or recall) is the percentage of times the classifier correctly predicts a high-

income observation among all the high-income observations in the dataset. Thus, the above plot tells us that the classifier can be made to predict with a TPR of about 0.9 while keeping the FPR as low as 0.1, which tells us that this is a good model.

3. At a "decision threshold" probability of **0.5** (above which all observations are predicted as high-income), it is observed that the (test) precision and recall of the model are **72%** and **49%** respectively. (Recall is the same as TPR, while precision is percentage of times the classifier correctly predicts a high-income observation among all the *predicted-to-be* high-income observations.) This means that the model is still prone to incorrectly predicting a high-income observation as a low-income observation. However, when this threshold is changed to 0.3, then these figures change to **60%** and **65%** respectively.

Feature Importance

An advantage of the CatBoost model is that the relative importance of all the features in the model can be calculated after the model is trained. Feature importance roughly means the extent to which a small change in the value of that feature affects the model output.

	feature	importance
0	age	12.298
1	weeks worked in year	9.575
2	hispanic origin	9.077
3	family members under 18	9.054
4	veterans benefits	9.012
5	education	6.062
6	sex	4.859
7	detailed occupation recode	4.640
8	marital stat	4.387
9	tax filer stat	4.148
10	capital gains	2.907
11	detailed household summary in household	2.513
12	dividends from stocks	2.462
13	major occupation code	2.415
14	enroll in edu inst last wk	1.626
15	num persons worked for employer	1.561
16	class of worker	1.504
17	detailed household and family stat	1.487

Above is the table of the first few feature importance values from the trained model. According to this table, the age of a person appears to influence their income the most, which points at **marketing products towards specific age groups** in the hope that they may have higher income. The number of family members under 18 is also a good predictor of income, which is again a good marketing insight. However, there are some protected attributes that also show up in this table, such as "Hispanic origin" – even though it appears to have high predictive value, use of this attribute can be avoided in marketing decisions to ensure fairness (this can be done by excluding this attribute when training the model).

Segmentation – K-Prototypes

For the segmentation task, I assumed that the income label is generally unknown at the time of testing/model deployment, so I excluded the label from the data for this task. As usual, I separated the weights from the data for use in training, but I had to perform some extra preprocessing steps to speed up initialization and convergence of the algorithm:

1. Keep only the top 20 most frequently occurring values in each category, and collapse the remaining values into an "Other" category. (I also printed the new value counts to ensure I did not accidentally duplicate an existing "Other" or equivalent category.)
2. Ensure that all categorical columns were represented as strings, and numeric columns as floats.
3. Scale the numeric features and save the scaler object (this will be useful later).

I then ran the K-Prototypes algorithm for different values of K (number of clusters or segments), and using the "elbow method" to select the best number of clusters gave me K = 9. I then retrieved the "cluster centroids" for this model (can be thought of as "representative values" for each of the segments and converted them to a DataFrame. Since the numeric values were initially scaled before clustering, I used the saved scaler object to "inverse transform" those features so that they can be interpreted. The centroids for this run are too wide to display (since there are 40 features), but can be viewed here.

For the sake of simplicity and easy interpretability, I repeated this procedure for K = 4 as well, and the results can be found here. The main thing that sets these 4 clusters apart is the age – cluster 0 appears to represent children, teenagers, and young adults; clusters 1 and 3 appear to represent the age group of 30-50, and cluster 2 represents elderly people. The other columns tell us some information that the majority of these groups have in common – for instance, cluster 2 also has very high dividends from stocks, and the employment status sets cluster 1 (primarily full-time schedules) and cluster 3 (primarily

Armed Forces) apart. In this way, correlations between different variables become apparent in the different clusters.

Conclusion and Future Work

Here are the key conclusions from this project:

1. Classification:
    a. The CatBoost model performs reasonably well on the data, and it identifies features like “age”, “family members under 18”, and “education” as strong predictors of the income group (although the direction of correlation is not known, i.e., whether a high age is an indicator of lower or higher income).
    b. Protected attributes like “Hispanic origin” also show up as important features, but such attributes could be omitted from the model to ensure fairness.
2. Segmentation: A smaller number of clusters is ideal for interpretability and marketing purposes, and once again age was a major differentiating factor among the clusters. Correlations between different variables can also be identified via the cluster centroids (which answers the question “what identifies this segment of the data”), but care must be taken with the categorical variables, since these represent the majority value for that attribute in the cluster, and may still not accurately represent the cluster.

Future work could involve removing protected attributes or changing the training hyperparameters for classification, or trying more random initializations or efficient versions of Gower distance calculation (since this is a different metric for categorical data) for segmentation. Evaluation of the clustering quality using 2D projection methods like UMAP is another step to consider. Overall, these models can still serve as a baseline to derive more comprehensive marketing insights for financial customers in the future.

References

1. https://catboost.ai/docs/en/concepts/tutorials
2. https://github.com/catboost/tutorials
3. https://github.com/nicodv/kmodes/blob/master/kmodes/kprototypes.py
4. https://antonsruberts.github.io/kproto-audience/

5. https://medium.com/analytics-vidhya/the-ultimate-guide-for-clustering-mixed-data-1eefa0b4743b
6. https://draw.io/