Filling Gaps in Social Demographic Data using Machine Learning, Energy Consumption, and Alternative Data

Arjun Balaji[1], Deepta Jasthi[1], Diego Ponce[2], Emma Riley[2], Kevin Li[2], Laksh Bhambhani[1], Pulak Dugar[1]

[1]University of California, Berkeley
[2]East Bay Community Energy

Abstract-We evaluate the efficacy of machine learning methods in predicting social classes of its customers in the East Bay. Specifically, we study classification models built using AutoML focusing on the features directly, and through PCA and LDA components. We find that the multiclass feature model built using domain knowledge best forecasts social classes.

***Index Terms*—Exploratory Data Analysis, AutoML, Machine Learning, Classification, Principal Component Analysis, Linear Discriminant Analysis, Colinearity Analysis**

I. Introduction

East Bay Community Energy (EBCE) is the largest provider of clean energy in the East Bay of California. It's goal is to provide 100% clean, affordable, and just energy by 2030 to all of the East Bay. EBCE has recently received a dataset of social demographic data for many of our customers in the East Bay. However, this data has gaps and some customers are missing data. This research paper discusses the process to fill data gaps in the social demographic data so that all our customers have social demographic data.

The project will build a machine learning model that uses a labeled data set of social demographic characteristics, energy data, and alternative data (e.g., census data, building characteristics data) to predict the social demographics of customers for which we don't have data.

II. Dataset Construction & Methodology

A major part of our project begins with creating a dataset of customers from the raw data we had access to. This final dataset would be split into the following tables and subtables:

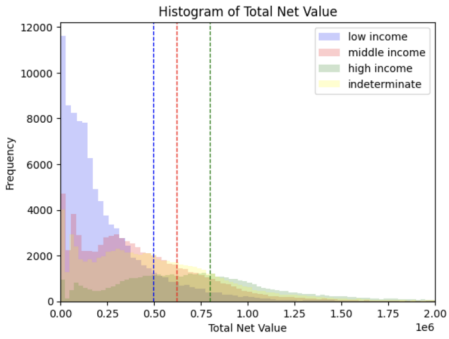
1) Social Demographics
    a) **4013 Reference Proxy:** The 4013 is a weekly extract of all the service agreement ids (sa_ids) that are in the service territories served by EBCE.
    b) **4013 Rates Proxy:** This table, organized with one row per sa_id contains codes to identify which rate a particular customer is billed at.
    c) **Aging Reports Processed Proxy:** This table provides information on arrearages, which are outstanding balances that are owed.
    d) **Smud Billing Details Proxy:** This table contains granular detail on the various components of a customer's bill.
    e) **Interaction Terms:** This table uses values from Aging Reports and Smud Billing Details to calculate average and max arrearage over bill periods
    f) **Vehicle Ownership Proxy:** This is a table that includes all vehicle characteristics available from DMV data and matches that data to EBCE customers.
    g) **Customer Parcel Matches Unique Proxy:** This table reflects tax assessor data at the sa_id level and contains useful information about the properties located in our service territory.
    h) **Cal Enviro CES V4:** Cal Enviro uses environmental, health, and socioeconomic information to produce scores for every census tract in the state.
    i) **California Investments Priority Populations:** This table identifies disadvantaged communities and low-income communities as defined for California Climate Investments, by census tract.
    j) **California Opportunity Maps:** California's opportunity map identifies census tracts/counties whose characteristics have been shown by research to support positive economic, educational, and health outcomes for low-income families.
2) Interval Data
    a) **Interval Data Proxy:** Running history of 1-hr interval usage of service points within our service territory, pulled from 01/01/2022 through present for customers served by EBCE.

III. Exploratory Data Analysis (EDA)

After constructing our final data set on a customer level, and getting rid of rows with 'NaNs' and Nulls, we visualized our data through histograms, box-plots, violin plots, and heat maps. Visualizing our data ultimately serves as a method of vetting our data and choosing the features that will result in a higher accuracy model.

A. Social Demographics Data

The final social demographics data table consists of 56 columns and 478205 rows that each represent a customer.



Low income mean: 493741.22425089235
Low income median: 157916.78048780488
Low income std. dev: 4744093.0185905155
Medium income mean: 622266.4421312972
Medium income median: 360911.5
Medium income std. dev: 4390149.720274347
High income mean: 798704.1961312569
High income median: 709096.5
High income std. dev: 1420711.4418304123

Fig. 1. Histogram of Total Net Value for the customers in the Social Demographics Table

Histograms like these (created for every column in the dataset) helped point out the differences and their magnitudes for each income group.

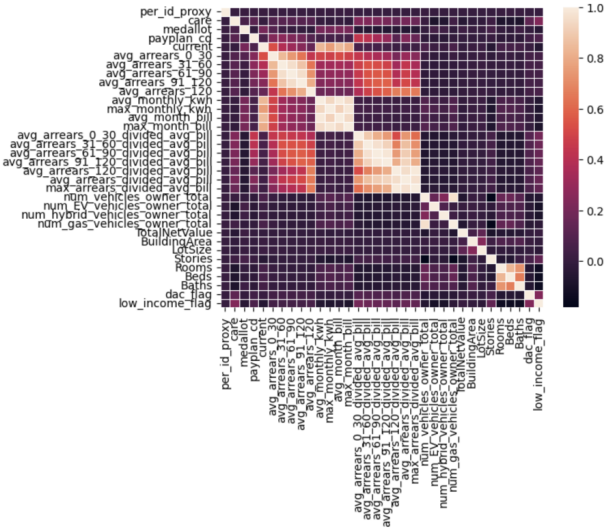


Fig. 2. Histogram of all Social Demographic Features

Additionally, the heatmap above (Fig 2) generated for features in the dataset served 2 functions:

1) It helped us get a sense of which features have a strong correlation with income groups especially low income (since ultimately we want to predict this for our customers).
2) It helps us get a sense of collinear features (for when we are training our model) so we could experiment with training without them.

Description	Name	Transformation
(1) Consumption figures		
$\overline{P}$ (daily, week)	c_total	sqrt(x)
$\overline{P}$ (daily, weekdays)	c_weekday	sqrt(x)
$\overline{P}$ (daily, weekend)	c_weekend	sqrt(x)
$\overline{P}$ for (6 a.m.–10 p.m.)	c_day	sqrt(x)
$\overline{P}$ for (6 p.m.–10 p.m.)	c_evening	sqrt(x)
$\overline{P}$ for (6 a.m.–10 a.m.)	c_morning	sqrt(x)
$\overline{P}$ for (1 a.m.–5 a.m.)	c_night	log(x)
$\overline{P}$ for (10 a.m.–2 p.m.)	c_noon	sqrt(x)
Maximum of $\overline{P}$, week	c_max	x
Minimum of $\overline{P}$, week	c_min	log(x)
(2) Ratios		
Mean $\overline{P}$ over maximum $\overline{P}$	r_mean/max	log(x)
Minimum $\overline{P}$ over mean $\overline{P}$	r_min/mean	sqrt(sqrt(x))
c_morning/c_noon	r_morning/noon	log(x)
c_evening/c_noon	r_evening/noon	log(x)
c_noon/c_total	r_noon/day	sqrt(x)
c_night/c_day	r_night/day	log(x)
c_weekday/c_weekend	r_weekday/weekend	log(x)
(3) Temporal properties		
Proportion of time with $\overline{P} > 0.5$ kW	t_above_0.5kw	x
Proportion of time with $\overline{P} > 1$ kW	t_above_1kw	x
Proportion of time with $\overline{P} > 2$ kW	t_above_2kw	x
Proportion of time with $\overline{P} >$ mean	t_above_mean	x
(4) Statistical properties		
Variance	s_variance	sqrt(sqrt(x))
$\sum(\lvert\overline{P}_t - \overline{P}_{t-1}\rvert)$ for all t	s_diff	sqrt(x)
Cross-correlation of subsequent days	s_x-corr	x
#$\overline{P}$ with $(\overline{P}_t - \overline{P}_{t\pm1} > 0.2$ kW)	s_num_peaks	x
(5) Principal components		
First 10 principal components	pca_i($i = 1\ldots10$)	x

Fig. 3. List of features created using Interval Data Proxy

B. Interval Data

The final interval data table consists of 25 columns and 440536 rows that each represent a customer. These 25 features (Fig 3) were calculated according to the features computed by Beckel et al. in a similar study. [1] We chose to omit the last feature that Beckel et al. used, the first 10 principal components, in our interval data set, as we would instead use them as independent features later, along with the axes produced by Linear Discriminant Analysis (LDA).

It is worth noting that the Interval data has less rows compared to Social Demographics data since outliers (Z-score greater than 3) were dropped. The last step we took was to replace the `income_range` variable with "low income" for all customers that had a "YES" for the `care` and `fera` variables. These variables indicate if a customer is part of a discount program as low-income household.

Ultimately, the following features were chosen to be used in the final dataset that the models can be trained on:

`income_range`, `p_city`, `nem`, `avg_arrears`, `avg_monthly_kwh`, `max_monthly_kwh`, `avg_month_bill`, `max_month_bill`, `num_vehicles_owner_total`, `total_net_value`, `ces_score`, `opportunity_flag`, `c_total`, `c_total_sqrt`, `c_weekday`, `c_weekday_sqrt`, `c_max`, `c_min`, `c_day`, `s_variance`, `s_diff`, `s_num_peaks`

IV. Models

We use 5 AutoML models to predict income groups:

1) Built using domain knowledge
2) Built using results from colinearity analysis
3) Built using the features that carry the most weight in PCA and LDA
4) Built using principal components as features
5) Built using LDA as features

In all models, we predict the income groups for the inputted customer based on data values in other columns.

A. Model 1: Built Using Domain Knowledge

The first model we built utilized the features we found to be useful for predicting income groups in our exploratory data analysis. To address the issue of class imbalances in the dataset, we created a new training dataset in Google Big Query with a balanced distribution of income classes. The goal was to ensure that each income class (low, middle, high) had a minimum number of samples for training our model effectively. We started by creating a random sample with replacement of customer rows, and used 80% of the minimum of low, middle, and high incomes for our training set. With the balanced training dataset in hand, we trained our model in Google's Auto ML. We then applied the model to the test dataset (the remaining 20% of the original data) to evaluate its performance. We performed a batch prediction, where the model predicted the income groups of low, middle, and high for all the rows in the test set.

We also made predictions following the same method for a binary dataset that predicted whether a customer was "low income" or "not low income".

B. Model 2: Built Using Colinearity Analysis

The next model utilized a subset of the features from Model 1 by factoring in Colinearity. We utilized the heatmap shown in Fig. 2 to determine the rows that had a high level of correlation. We chose features where the majority of columns were lighter (indicating low correlation with other features). We also kept features that were shown to have a high level of predictability in Model 1. The chosen features were the following:

`income_range`, `p_city`, `avg_arrears`, `avg_monthly_kwh`, `num_vehicles_owner_total`, `total_net_value`, `ces_score`, `opportunity_flag`, `c_total`, `nem`, `s_num_peaks`

Similarly to Model 1, we dealt with class imbalances and made predictions for a binary model and a multiclass model.

C. Model 3: Built using the features that carry the most weight in PCA and LDA

The third model utilized a subset of the features from Model 1 that were most heavily weighted in PCA and LDA. Performing PCA and LDA involved importing their respective modules from scikit-learn, setting the number of components to 30 for PCA and 2 for LDA (standard practice for LDA is the number of classes minus 1). From the subsequent computation of components, we ordered each feature in our original data set by their importance to each individual component of PCA/LDA, and took the first 5 features from first 5 most important components. We trained and tested one sub-model using a multi-class data set and a second sub-model using a binary data set as before.

D. Model 4: Built using principal components as features

The fourth model utilized the actual principal components of the training data (for both the multi-class and binary data sets) as features on which to train.

E. Model 5: Built using LDA axes as features

The fifth model utilized the axes computed by LDA as features on which to train. While one model was trained for the multi-class data set, it was impossible to train a second sub-model for the binary data set as before, since LDA would have only produced one axis as a sole feature on which to train (the number of classes minus 1 would equal 1, owing to the nature of binary data).

V. Results

We calculated four metrics for each model: accuracy, precision, recall, and F1 score. However, we will mainly focus on recall. Recall measures out of all positive real cases how many are predicted to be positive. Because it's more important to classify one as low income than to not, we will use this metric to evaluate the success of each model.

A. Model 1 Results: Built Using Domain Knowledge

The overall recall of the Multiclass Model 1 was 0.71. Below is a table with the metrics for this model. The first 4 numbers are metrics for the model as a whole while the table is a breakdown by class.

```
Accuracy: 0.705361
Precision: 0.817973
Recall: 0.705361
F1 score: 0.740717
| Class   |   Precision |   Recall |   F1 Score |
|---------+-------------+----------+------------|
| Low     |    0.948664 | 0.726979 |   0.823157 |
| Middle  |    0.306728 | 0.56269  |   0.39703  |
| High    |    0.399482 | 0.773763 |   0.526923 |
```

Fig. 4. Batch Prediction Metrics for a Multiclass Model 1

The overall recall of the Binary Model 1 was 0.72. Below is a table with the metrics for this model.

We see that the binary model performed better overall, with a slightly higher recall.

```
Accuracy: 0.717312
Precision: 0.798010
Recall: 0.717312
F1 score: 0.737245
| Class   |   Precision |   Recall |   F1 Score |
|---------+-------------+----------+------------|
| Low     |    0.751824 | 0.701553 |   0.725819 |
| Not Low |    0.720258 | 0.768418 |   0.743559 |
```

Fig. 5. Batch Prediction Metrics for a Binary Model 1

B. Model 2 Results: Built Using Colinearity Analysis

The overall recall for the Multiclass Model 2 was 0.70, which is slightly lower than our results from Model 1. The overall recall for the Binary Model 2 was 0.65, which is significantly lower than our recall value of 0.72 from Binary Model 1.

Model 1 performed better in terms of recall and other metrics, showing that colinearity did not negatively effect our model.

C. Model 3 Results: Built using the features that carry the most weight in PCA and LDA

The overall recall for the Multi-class Model 3 was 0.67. It is notable that the precision and F1 score for the 'Low Income' class were both very high, at 0.93 and 0.80 respectively; however, scores drop dramatically for the Middle and High Income classes.

```
Accuracy: 0.674339
Precision: 0.796881
Recall: 0.674339
F1 score: 0.712801
| Class   |   Precision |   Recall |   F1 Score |
|---------+-------------+----------+------------|
| Low     |    0.938742 | 0.702147 |   0.803388 |
| Middle  |    0.278145 | 0.491419 |   0.355229 |
| High    |    0.362402 | 0.760455 |   0.490873 |
```

Fig. 6. Batch Prediction Metrics for a Multi-class Model 3

The overall recall for the Binary Model 3 was 0.63, slightly worse than the score for the Multi-class Model 3. However, the Binary model produced similar scores of around 0.63 for Precision, Recall, and F1 Scores for both predicted classes ('Low Income' and 'Not Low Income').

```
Accuracy: 0.629496
Precision: 0.705230
Recall: 0.629496
F1 score: 0.648975
| Class   |   Precision |   Recall |   F1 Score |
|---------+-------------+----------+------------|
| Low     |    0.637269 | 0.623633 |   0.630378 |
| Not Low |    0.631518 | 0.645031 |   0.638203 |
```

Fig. 7. Batch Prediction Metrics for a Binary Model 3

D. Model 4 Results: Built using principal components as features

Out of the two sub-models for Model 4, the Binary model performed slightly better. However, the Recall and F1 scores of 0.51 and 0.53 are still quite low compared to the previous models discussed.

```
Accuracy: 0.506214
Precision: 0.630239
Recall: 0.506214
F1 score: 0.531225
| Class   |   Precision |   Recall |   F1 Score |
|---------+-------------+----------+------------|
| Low     |    0.538283 | 0.472962 |   0.503513 |
| Not Low |    0.529997 | 0.594314 |   0.560316 |
```

Fig. 8. Batch Prediction Metrics for a Binary Model 4

E. Model 5 Results: Built using LDA axes as features

Model 5 performed as poorly as Model 4, with overall Recall and F1 scores of 0.50 and 0.57.

```
Accuracy: 0.504343
Precision: 0.792396
Recall: 0.504343
F1 score: 0.566422
| Class   |   Precision |   Recall |   F1 Score |
|---------+-------------+----------+------------|
| Low     |    0.817279 | 0.480103 |   0.604876 |
| Middle  |    0.459365 | 0.480166 |   0.469536 |
| High    |    0.597149 | 0.816468 |   0.689795 |
```

Fig. 9. Batch Prediction Metrics for a Multi-class Model 5

VI. Discussion and Conclusion

Looking at the results of each of the 5 models and their performance on both a multi-class and binary data set, Model 1 trained on the multiclass data set seemed to perform the best, with Precision of 0.81, Recall of 0.71, and an F1 score (what we hope to optimize) of 0.74. The Binary Model 1 and Multi-class Model 3 lag behind slightly with F1 scores of 0.737 and 0.713, respectively.

It is interesting to note the poor performance of Models 4 and 5. While PCA and LDA are known to be effective at dimensionality reduction, they seemed to have performed very poorly on classifying test data. What is even more interesting is the Training F1 Score of around 0.83 for each of these models, leading us to suspect that the models were probably over-fit to the training data.

Ultimately, Model 1 (A multiclass model focusing on all features) and Model 3 (a subset of model 1 only trained on features that carry the most weight in PCA) performed the best. It is worth noting that Model 3 metrics are very similar to Model 1 which is good since our target column (income group) can be predicted with fewer features.

In the future, it is worth experimenting with model 3 and the different features (and number of features) that go into it as well as the model training characteristics in AutoML.

REFERENCES

[1] Christian Beckel, Leyna Sadamori, Thorsten Staake, and Silvia Santini. Revealing household characteristics from smart meter data. *Energy*, 78:397–410, 2014.