Imputation of financial variables from PUF to CPS

PUF contains the following variables which are important to understanding the effects of a tax reform, and should be included in the data powering the TaxCalculator:

- `E02000`: Total Sch E income or loss (income or loss from rental real estate, royalties, partnerships, S corporations, estates, trusts, and residual interests in REMICs)
- `E26270`: Income or loss from a combined partnership or S corporation
- `P22250`: Short-term capital gain or loss
- `P23250`: Long-term capital gain or loss
  (note that in CPS letters in variable names are lower-case)

We can leverage the variables contained in both PUF and CPS to make informed estimates of these variables' values for the CPS observations, which are currently missing these data. The following variables are contained in both PUF and CPS, and might serve as useful signals for our imputation variables:

Potential predictors

```
['EIC', 'DSI', 'MARS', 'XTOT', 'E00200', 'E00300', 'E00400',
'E00600', 'E00800', 'E00900', 'E01100', 'E01400', 'E01500',
'E01700', 'E02100', 'E02300', 'E02400', 'E03150', 'E03210',
'E03240', 'E03270', 'E03300', 'E17500', 'E18400', 'E18500',
```

```
'E19200', 'E19800_E20100','E20400', 'E32800', 'F2441', 'N24']
```

See descriptions here: Link

Note:

- While `E19800` and `E20100` are separate charity variables in PUF (reporting a tax unit's Schedule A cash/check and Schedule A non-cash/non-check gifts respectively), the original CPS data only report 1 charity variable equal to their sum for a given household. Thus, only their sum, `E19800_E20100` is used as a signal, rather than their separate values, in the models tested herein.
- `E00650` (qualified dividends) is also excluded despite being in both PUF and CPS as it's value is included in `E00600` (ordinary dividends)

Analysis of variables

Imputation variables

Our imputation variables represent income types that are generally uncommon, therefore, most tax units have no income of their types. Thus, for all of our imputation variables, most of the data is 0. Below is an extended analysis where the following additional characteristics are observable for the *non-zero* data for all 4 imputation variables:

- Long tails
- High skew
- Large (negative or positive) observations clustered by the lowest and highest AGI's, with large negative observations clustered by

AGI = 0. Naturally, wealthier units with higher AGI will have higher investment/business income, but those same units might also experience great losses with which they offset their AGI to 0.

E02000

65% of data are zero

Non-Zero Summary stats:

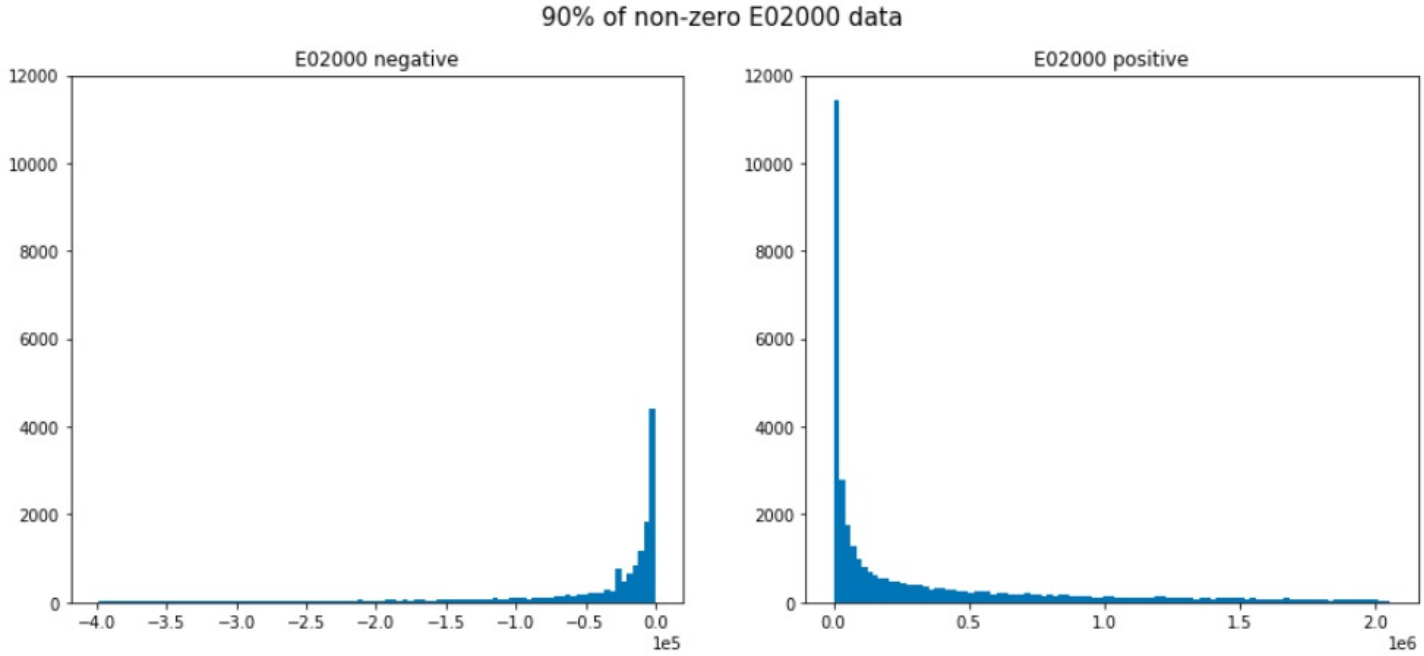
Mean = 345,312

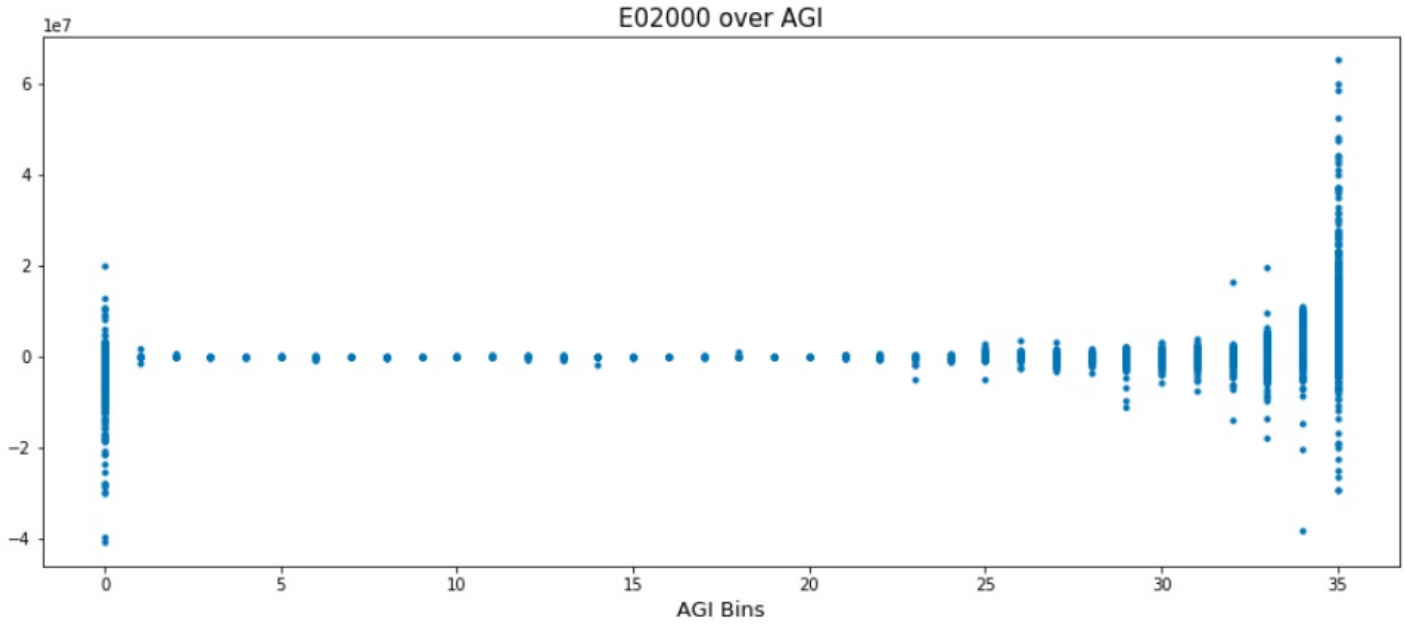
Std = 1,804,854

Max = 65,100,000

Min = -40,760,000

Skew = 6.95





E26270

72% of data are zero

Non-Zero Summary stats:

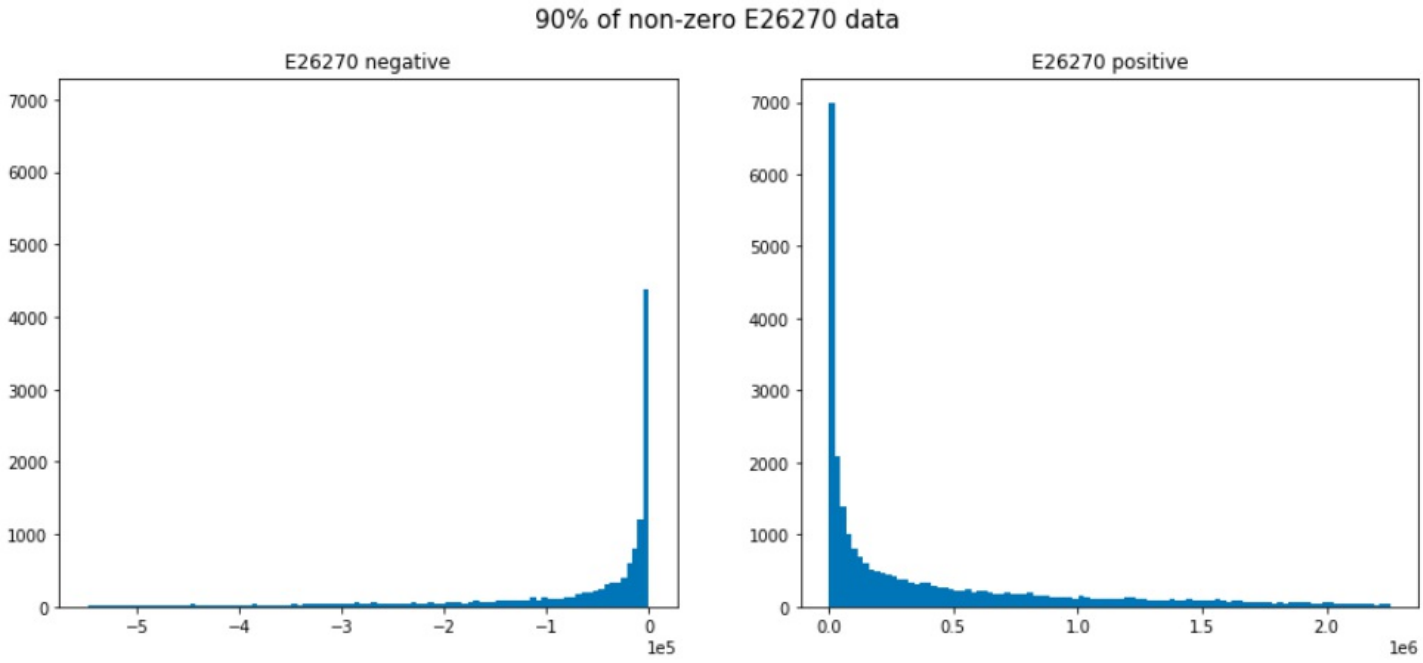
Mean = 382,947

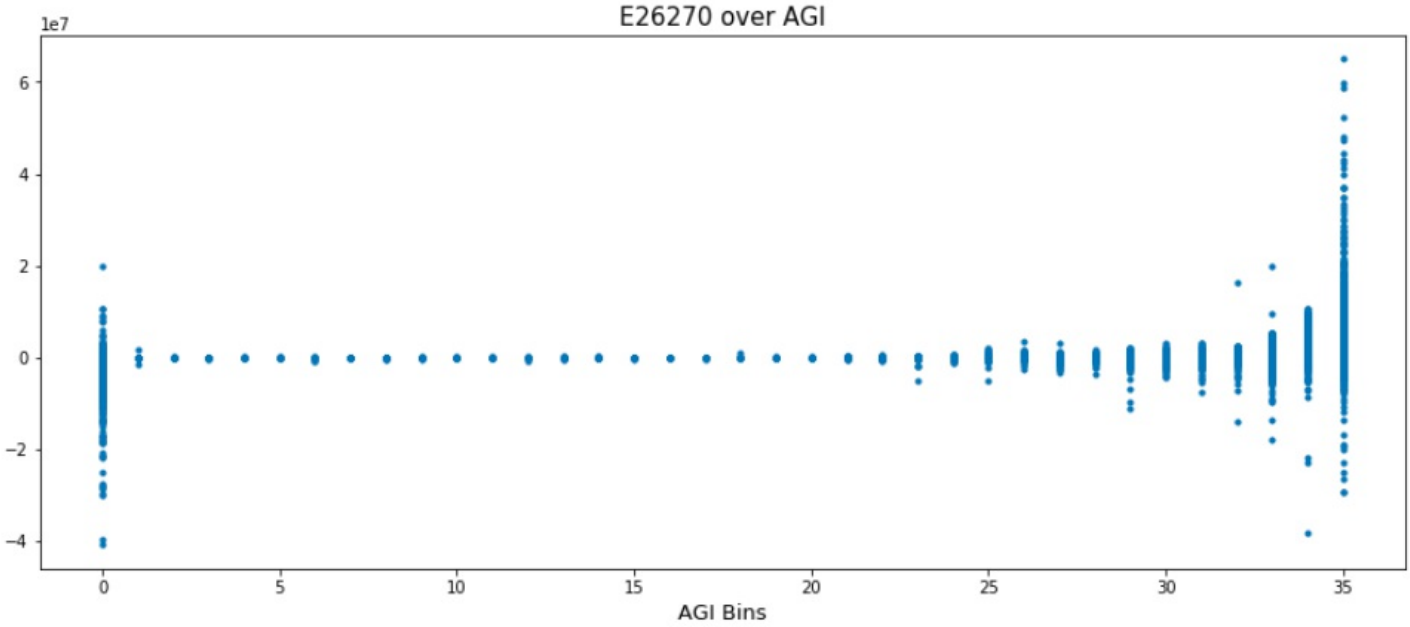
Std = 1,942,446

Max = 65,100,000

Min = -40,760,000

Skew = 6.16





P23250

66% of data are zero

Non-Zero Summary stats:

Mean = 417,571

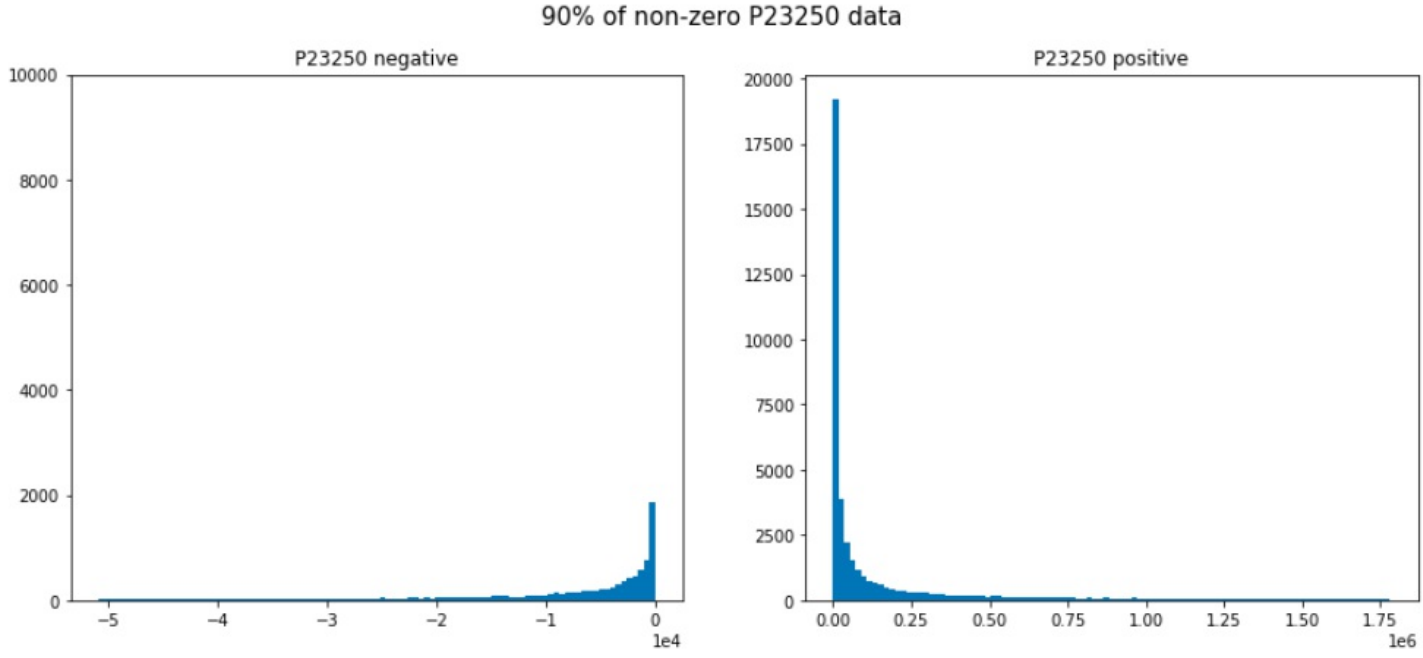
Std = 2,442,350

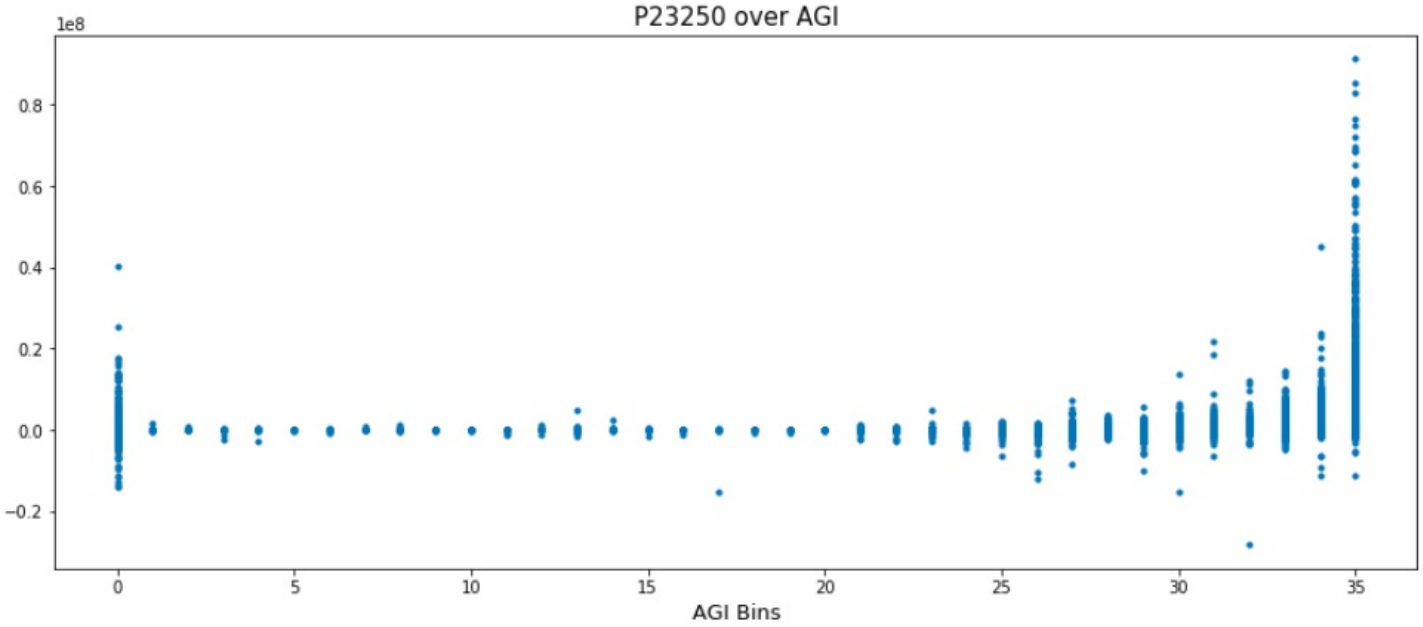
Max = 91,220,000

Min = -28,160,000

Skew = 15.16

Note: Negative plot's y-scale is half the positive plot's for readability





P22250

73% of data are zero

Summary stats:

Non-Zero Summary Statistics:

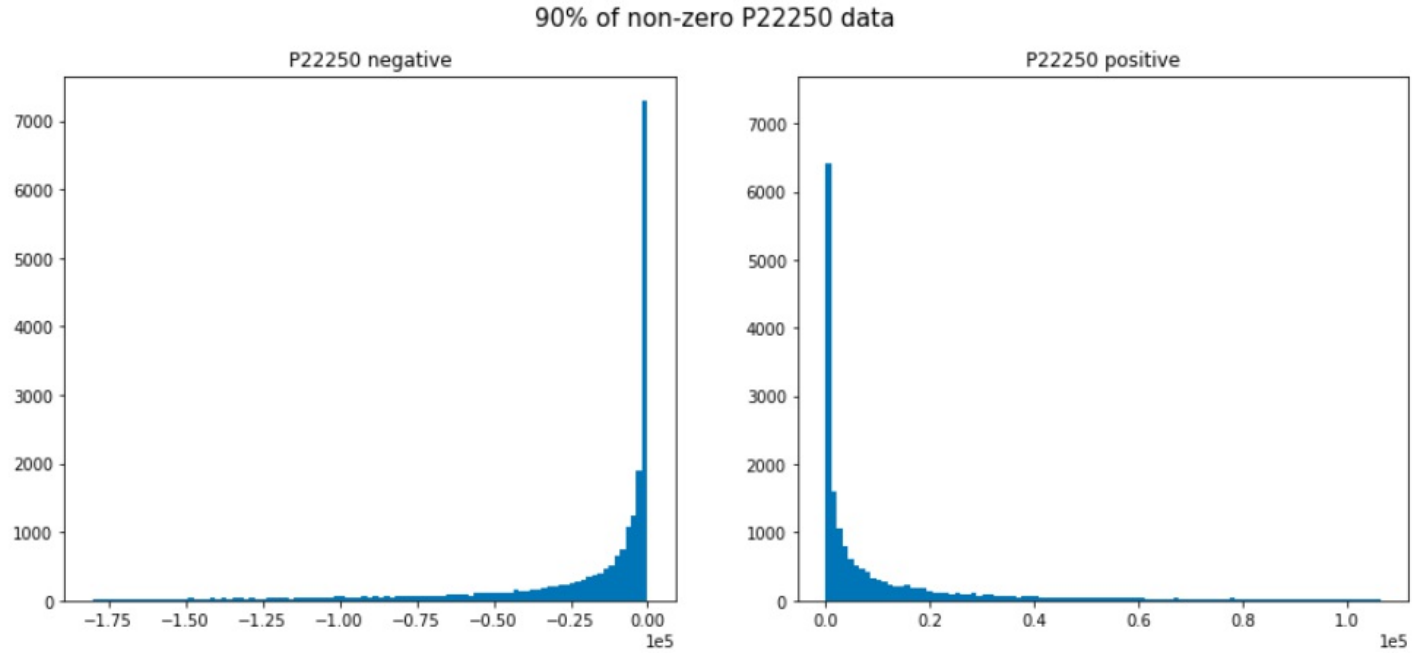
Mean = -14,018

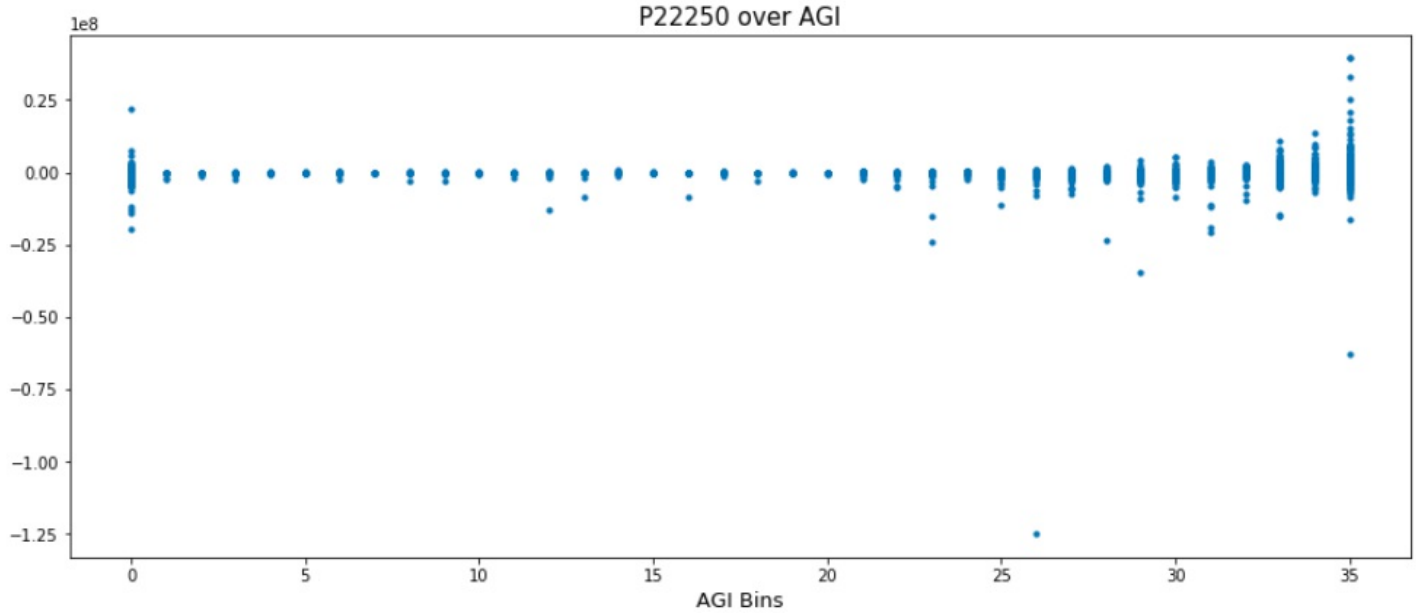
Std = 956,064

Max = 39,410,000

Min = -124,900,000

Skew = -55.81





Outliers: All imputation variables present long tails terminating in a small handful of tax units with particularly large negative or positive values, relative even to the other units on their end of the tail. That being said, `P22250` is somewhat unique in that it has two clear outliers visible in the bottom right quadrant of the above plot. These units have `RECID` of 102565 and 349145, with `P22250` values of about -$124,000,000 and -$60,000,000 respectively. All models tested produced better results in RMSE when these outliers were excluded.

Conclusions:

- Given `P22250`'s relatively high skew (among distributions with already-large skews) it should make for a very difficult prediction task, and thus, an effective test-case by which to compare models.
- It may be of benefit to regress on log(`P22250`) when using linear models, as the transformation would make the distribution of the non-zero data near-normal, and linear models function best with normally distributed variables.
- It may be of benefit to test two-stage imputations which first

predict `P22250`'s sign, then its continuous value. In a two-stage imputation, if a 0 is predicted, a 0 is imputed. Otherwise, continuous negative/positive values are imputed to those observations predicted to be negative/positive using separate models trained on the negative/positive training data. This would mitigate the tendency for models trained on all of the data to minimize their error by simply predicting near-zero values for most all observations, and allow them to capture the more extreme negative/positive values which are important to include so that a tax policy analysis will reflect the behavior of particularly wealthy tax units, or those with large business/investment incomes.

Independent variables (analysis continued)

Predictor correlations

Many of our predictors are correlated with one another to some degree. See the following table which reports the top 3 correlations for every predictor: Correlation table. See a snapshot of that table below, sorted on the column “Highest corr” which reports the most-correlated predictor and the associated correlation coefficient for each variable.

	Highest corr	Second highest	Third
F2441	E32800: 0.84	N24: 0.32	XTOT: 0.24
E32800	F2441: 0.84	N24: 0.27	XTOT: 0.2
N24	XTOT: 0.77	EIC: 0.37	F2441: 0.32
XTOT	N24: 0.77	EIC: 0.25	F2441: 0.24
E00200	E18400: 0.4	E18500: 0.32	E19800_E20100: 0.14
E00300	E20400: 0.4	E18400: 0.36	E19200: 0.35
E18400	E00200: 0.4	E18500: 0.39	E00300: 0.36

Most of these correlations are understandable. `F2441` reports the number of children/dependents in Form 2441, and `E32800` reports the expenses associated with those dependents. `N24` and `EIC` both report on the number of children. `E00200` reports wages, salaries and tips while `E18400` reports state/local income tax calculated largely on the basis of those wages, salaries and tips. `XTOT` reports the unit's total number of exemptions and probably correlates with the dependent/child variables given the existence of dependent/child exemptions. High levels of correlation between predictors can lead to multicollinearity, which can present issues in linear regressions. An extended discussion of multicollinearity can be found here.

Predictors are mostly zero

As with our imputation variables, the non-categorical predictors are largely zero. In fact, for all but two non-categorical predictors, greater than 50% of their data are 0. For 14 of 31, over 90% of their data are zero. See full list below:

(`Variable`: proportion zero)

1 - 7	8 - 14	15 - 21	22 - 28	29 - 31
`E00800`: 0.998	`E32800`: 0.957	`E03270`: 0.886	`N24`: 0.699	`E00300`: 0.425
`E03150`: 0.976	`E03210`: 0.954	`E01400`: 0.875	`E20400`: 0.666	`E00200`: 0.209
`E02100`: 0.958	`E03240`: 0.953	`E01700`: 0.808	`E19200`: 0.599	
`E00800`:	`F2441`:	`E00400`:	`E00600`: 0.586	

0.998	0.953	0.806		
`E01100`: 0.986	`E03300`: 0.95	`E02400`: 0.803	`E18400`: 0.56	
`E03150`: 0.976	`E17500`: 0.938	`E00900`: 0.773	`E19800_E20100`: 0.542	
`E02100`: 0.958	`E02300`: 0.935	`E01500`: 0.766	`E18500`: 0.535	

Log-transforming predictors

As with our dependent imputation variables, the independent (continuous) predictors have very long tails and linear models should benefit from using their log-transformed values. See this notebook which compares the distributions of the predictors before and after log-transformation: Log-predictors notebook. As expected, the linear models which used the non-log-transformed continuous predictors performed poorly, predicting values in the trillions for P22250. The log-transformation methodology used for both the imputation variable and the continuous predictors is:

```
if variable.min() >= 1:
    log_variable = np.log(variable)
    log_variable = np.log(variable - variable.min() + 1)
```

Thus, for columns where all data were greater than or equal to 1, the log transformation was directly applied. Else, the data were transformed so that all were greater than or equal to 1, and the log-transformation was applied. With regards to the predictions, since the models predict the value of `log_P22250` = log(P22250 - P22250.min() + 1), they are transformed

back to actual `P22250` units for both negative and positive predictions using code analogous to the following, which is first utilized in cell 15:

```
P22250_predictions = np.exp(log_P22250_predictions) + puf['P2
2250'].min() - 1
```

Prediction

Methodology

Train/test

Models were trained on an 80% subsample of PUF and tested on a 20% subsample so as to assess their performance on out-of-sample observations. The train/test split is done using a random seed so it is replicable (cell 2, bottom).

Stochastic imputation

It is useful to insert a degree of randomness (stochastic means random) into the process by which one imputes missing data using a regression model. This will give the predictions a more realistic spread. In the following models, randomness is imbued into the process at different stages of the imputation. For the random forests model, the randomness is built-in to how the model creates the decision trees. For the two-stage models, the randomness is applied in the first-stage categorical imputation, by using a random uniform number and the categorical model's predicted probabilties for each class to produce a predicted category, rather than simply (and non-stochastically) imputing the sign with the highest predicted probability. This stochastic sign-imputation is accomplished using the stochastic imputation function in cell 9.

Models

The following models were compared:

- **1-stage**: Random Forests
    - Random forests is one example of what's known as an "ensemble method" of regression. In this case, the "ensemble" is just a group of "decision trees" which are used together to predict the continuous value of `P22250`. Using many decision trees has advantages over using just one, as described in this wikipedia page:
        - "[Decision] trees that are grown very deep tend to learn highly irregular patterns: they overfit their training sets, i.e. have low bias, but very high variance. Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance in the final model.

      In this task, we are interested in high-performance models, rather than highly interpretable ones, so a Random Forest of decision trees is a good choice. To understand decision trees, check out this video: Decision Tree (CART)

- **2-stage**:
    - First stage: mnlogit to predict the sign of `P22250`
    - Second stage: Statsmodels' OLS
        - Regressing on log(`P22250`) using log-transformed predictors.
- **2-stage**:
    - First stage: mnlogit
    - Second stage: LassoCV
        - Regressing on log(`P22250`) using log-transformed predictors.
- **2-stage**:
    - First stage: mnlogit
    - Second stage: Random Forests
        - Not being a linear model, Random Forests never benefitted from log-transformed predictors, either as a lone model, or in conjunction with mnlogit.

Performance metrics

The following metrics were used to score the models' predictions:

- Log-loss: For scoring the sign prediction of the 1-stage Random Forests and the first stage of the 2-stage imputation, mnlogit.
- RMSE (root mean squared error): For scoring the final predictions.

Log-loss is a widely used metric for scoring classification models. The log-

loss function takes two inputs:

- Ground Truth (GT): The set of actual observations and their actual categories or classes. In our case, this is the testing subset of PUF, and its associated `P22250` signs.
- The model's predicted probabilities: A vector of probabilities (which add up to 1 for each observation) produced by the model which correspond to each potential category, and reflect its confidence that a given category is the true category of a given observation.

Log-loss outputs a number in [0, ∞). The higher the number, the *less* likely the model considers GT, and vis-versa. High numbers are bad. If the model assigned a probability of 1 to the correct category for every observation (and thus, a probability of 0 for every incorrect category), the log-loss would be 0, and the model would be perfect.

Note: The Random Forests model used (`sklearn.ensemble.randomforest.regressor`) does not actually produce probabilities for each category. Instead, it produces a batch of models (estimators) which predict the continuous value of the response variable. Thus, the Random Forests model's log-loss is produced by first deriving its "predicted probabilities" for each category from the batch of estimators' continuous predictions. This is done by calculating the proportion of estimators whose continuous predictions fall into each category (-/0/+) for each observation, and these are used as the vector of probabilities in the log-loss calculation.

Notebook: Link

The above notebook implements these models using the described methodology and metrics. These results are replicable assuming one has access to `puf2011.csv`. The notebook's structure looks like this:

- Data prep
- Models, performance metrics
- Comparing models to simply imputing 0 or the mean of P22250
- Comparing the plot of `P22250` over `AGIR1` (AGI variable with 35 levels) in the actual puf data, and from the predictions of each model.

Results

Log-loss
Random forests: 0.688 (cell 5)
mnlogit: 0.584 (cell 12)

Note: Changing the random seed which creates the train/test subsets has a large effect on Random Forests' log-loss. In a handful of trials, Random Forests' log-loss varied between 0.57 and 0.71! In my opinion, this indicates that the method used to calculate Random Forests' predicted probabilities is flawed, given how large the dataset is, you would expect these differences in train/test selection to be meaningless, and they *are* meaningless with respect to mnlogit's log-loss, which I found to vary in 0.580 - 0.589. This isn't to say Random Forests shouldn't be used, but that log-loss is not a particularly meaningful metric for comparing it to categorical models like mnlogit.

RMSE

Method	RMSE
Random forests	411,117 (cell 6)
mnlogit + OLS	389,051 (cell 16)
mnlogit + LassoCV	388,580 (cell 22)
mnlogit + Random forests	400,139 (cell 24)
Imputing zero	385,182 (cell 26)
Imputing mean	385,168 (cell 27)

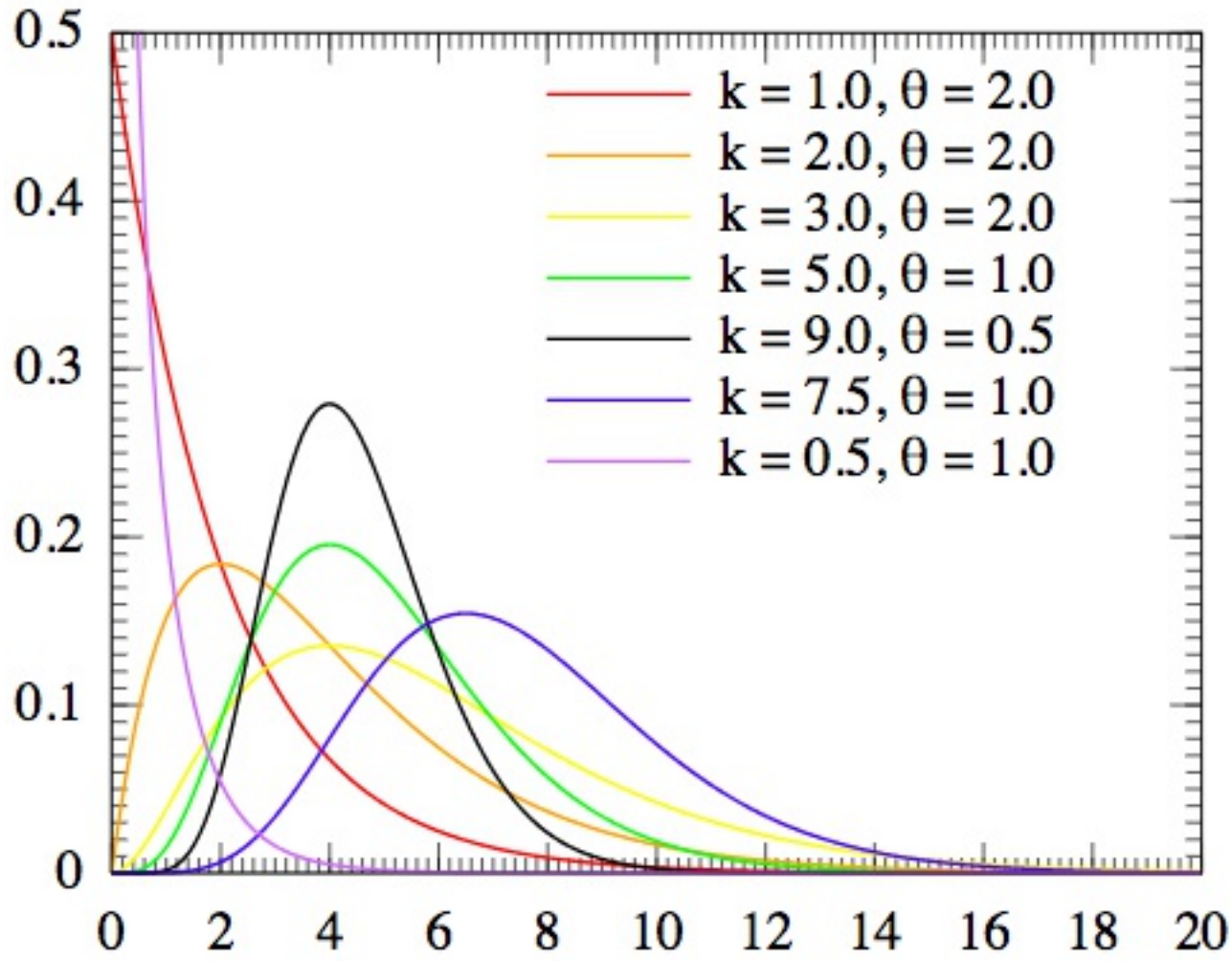
Conclusions

1. RMSE is a flawed metric.
    - Looking at the graphical analysis from cells 28-33, the reason that mnlogit + (OLS/LassoCV/Random Forests) outperform a one-stage Random Forests model is clear. The former models take less risks, predicting values clustered around zero, while the latter shows nuance in its predictions which include a handful of realistic extreme values. This results in a one-stage Random Forests with larger RMSE but much more realistic predictions, at least from a look-over of the grahps. In our tax model, we would prefer a dataset whose observations have a variety of business & capital gains income values over one which has low RMSE, but very few large or negative values.

2. A one-stage Random forests model creates what appear to be the best predictions so far.

Going forward

1. Investigate other success metrics. Max Ghenis, an OSPC collaborator has suggested quantile loss and provided the following resources in its regard:
    - Wiki
    - Stack Exchange
    - Sample implementation
2. Max has also suggested using “Deep quantile regression” as a good model for this task. He has provided the following resources in its regard:
    - Towards Data Science blog post
3. As the distributions of the imputation and predictor variables look quite similar to a gamma distribution (or two, reflected over the y-axis, see graphs below) I attempted to build a gamma-family model for this task using `sm.GLM()` as seen in the first example on this statsmodels page: Generalized Linear Models. Unfortunately the SVD fails to converge for almost all possible combinations of predictors when using this model, so I wouldn’t reccomend this function in particular, but a gamma-family model may still be of use.
   (see k = 0.5, $\Theta = 1$)



4. Implement a K-cross fold validation, which is a more robust technique of assessing model's out-of-sample performance than simply using an 80/20 train/test split. This technique ensures that all observations are used once in both training and testing subsets, and the average performance of models across all iterations is used to score them instead of just one.