EDAandPreprocessing

September 30, 2018

1 HMDA Data Analysis

The Home Mortgage Disclosure Act (HMDA) requires mortgage lenders in the United States to disclose information about the mortgage lending decisions they have made. Specifically, it requires that each lender report on all of the official mortgage applications they have received and whether or not the application was accepted or denied. This dataset provides a rich problem for data analysis and data science and is publicly available through their API. We would like you to build a classifier that predicts from the disclosed attributes of the mortgage applications whether or not the application will be approved. We are less concerned about the final accuracy of the classifier and more about the process and framework you use to get there. Please prepare a brief presentation in which you walk us through your process, results, and discoveries along the way. We hope to see the code used to explore and analyze the data.

The HMDA dataset is large, feel free to select a specific cut of the data for model development, for example a specific geography or a specific type of mortgage. The dataset can be downloaded here: https://www.consumerfinance.gov/data-research/hmda/

```
In [133]: import pandas as pd
          import seaborn as sns
          import numpy as np
          import os
```

```
In [2]: %pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

1.1 Preprocessing

The HMDA website says they have something on the order of 10-20 million records each year. Assuming a mean of < 2 mortgage applications per accepted mortgage, that's a surprisingly high turnover for mortgages in this country. This is clearly going to be a multi-gigabyte dataset. I'd like to get a full business cycle in there, so we'll limit records to owner occupancy single-family homes in NY State, but get the full dataset since 2007.

```
In [3]: # We're going to have to download the file ahead of time. It's about 1.5GB so we don't u
        if not 'rawdata.pqt' in os.listdir():
```

```
        url = "https://api.consumerfinance.gov:443/data/hmda/slice/hmda_lar.csv?$where=state
        rawdata = pd.read_csv(url)
        rawdata.to_parquet("rawdata.pqt")
    else:
        rawdata = pd.read_parquet("rawdata.pqt")
    rawdata.shape
```

```
/home/cnaylor/env/tensorflow/lib/python3.6/site-packages/IPython/core/interactiveshell.py:2785:
  interactivity=interactivity, compiler=compiler, result=result)
```

```
Out[3]: (4281178, 78)
```

77 features is far too many to graph. Let's get an idea of how many are actually used.

```
In [4]: completeness = rawdata.notnull().sum(axis=0)
        len(completeness.where(completeness>rawdata.shape[0]/2).dropna())
```

```
Out[4]: 53
```

Ok, a bit more than half have data more than 50% of the time. Since our motivation is to predict the approval likelihood of new mortgages, we'll drop the rest as not worth modeling.

```
In [5]: rawdata = rawdata.loc[:,completeness.where(completeness>rawdata.shape[0]/2).dropna().ind
```

That's still too much to look at, though. What about redundancies?

```
In [6]: rawdata.iloc[0,:]
```

```
Out[6]: action_taken                                                      3
        action_taken_name              Application denied by financial institution
        agency_code                                                       7
        agency_abbr                                                     HUD
        agency_name                    Department of Housing and Urban Development
        applicant_ethnicity                                               2
        applicant_ethnicity_name                         Not Hispanic or Latino
        applicant_income_000s                                            31
        applicant_race_1                                                  5
        applicant_race_name_1                                         White
        applicant_sex                                                     2
        applicant_sex_name                                           Female
        application_date_indicator                                        0
        as_of_year                                                     2008
        census_tract_number                                             NaN
        co_applicant_ethnicity                                            5
        co_applicant_ethnicity_name                            No co-applicant
        co_applicant_race_1                                               8
        co_applicant_race_name_1                               No co-applicant
        co_applicant_sex                                                  5
```

```
co_applicant_sex_name                                             No co-applicant
county_code                                                                   NaN
county_name                                                                   NaN
hoepa_status                                                                    2
hoepa_status_name                                                Not a HOEPA loan
lien_status                                                                     1
lien_status_name                                           Secured by a first lien
loan_purpose                                                                    1
loan_purpose_name                                                   Home purchase
loan_type                                                                       1
loan_type_name                                                       Conventional
msamd                                                                         NaN
msamd_name                                                                    NaN
owner_occupancy                                                                 1
owner_occupancy_name                           Owner-occupied as a principal dwelling
preapproval                                                                     3
preapproval_name                                                   Not applicable
property_type                                                                   1
property_type_name                 One-to-four family dwelling (other than manufa...
purchaser_type                                                                  0
purchaser_type_name                Loan was not originated or was not sold in cal...
respondent_id                                                          7071400009
sequence_number                                                              4109
state_code                                                                     36
state_abbr                                                                     NY
state_name                                                               New York
hud_median_family_income                                                      NaN
loan_amount_000s                                                               26
number_of_1_to_4_family_units                                                 NaN
number_of_owner_occupied_units                                                NaN
minority_population                                                           NaN
population                                                                    NaN
tract_to_msamd_income                                                         NaN
Name: 0, dtype: object
```

Quite a few. Build out concordances for the dual index-string fields.

```
In [7]: concordances = {v:dict(rawdata[[v, v+'_name']].head(10000).drop_duplicates().values) for
                        v in [v for v in rawdata.columns if v+'_name' in rawdata.columns]}

        concordances
```

```
Out[7]: {'action_taken': {3: 'Application denied by financial institution',
          1: 'Loan originated',
          5: 'File closed for incompleteness',
          2: 'Application approved but not accepted',
          4: 'Application withdrawn by applicant',
          7: 'Preapproval request denied by financial institution',
```

```
 8: 'Preapproval request approved but not accepted'},
'applicant_ethnicity': {2: 'Not Hispanic or Latino',
 3: 'Information not provided by applicant in mail, Internet, or telephone application'
 1: 'Hispanic or Latino',
 4: 'Not applicable'},
'applicant_sex': {2: 'Female',
 1: 'Male',
 3: 'Information not provided by applicant in mail, Internet, or telephone application'
 4: 'Not applicable'},
'co_applicant_ethnicity': {5: 'No co-applicant',
 2: 'Not Hispanic or Latino',
 3: 'Information not provided by applicant in mail, Internet, or telephone application'
 1: 'Hispanic or Latino',
 4: 'Not applicable'},
'co_applicant_sex': {5: 'No co-applicant',
 2: 'Female',
 1: 'Male',
 3: 'Information not provided by applicant in mail, Internet, or telephone application'
 4: 'Not applicable'},
'hoepa_status': {2: 'Not a HOEPA loan'},
'lien_status': {1: 'Secured by a first lien'},
'loan_purpose': {1: 'Home purchase', 3: 'Refinancing', 2: 'Home improvement'},
'loan_type': {1: 'Conventional',
 2: 'FHA-insured',
 3: 'VA-guaranteed',
 4: 'FSA/RHS-guaranteed'},
'msamd': {nan: nan, 10580.0: 'Albany, Schenectady, Troy - NY'},
'owner_occupancy': {1: 'Owner-occupied as a principal dwelling'},
'preapproval': {3: 'Not applicable',
 1: 'Preapproval was requested',
 2: 'Preapproval was not requested'},
'property_type': {1: 'One-to-four family dwelling (other than manufactured housing)',
 2: 'Manufactured housing'},
'purchaser_type': {0: 'Loan was not originated or was not sold in calendar year covered
 6: 'Commercial bank, savings bank or savings association',
 9: 'Other type of purchaser',
 7: 'Life insurance company, credit union, mortgage bank, or finance company',
 8: 'Affiliate institution',
 2: 'Ginnie Mae (GNMA)',
 5: 'Private securitization',
 1: 'Fannie Mae (FNMA)',
 3: 'Freddie Mac (FHLMC)'}}
```

Now we can drop the names

In [8]:
```
rawdata.drop([k+'_name' for k in concordances], axis=1, inplace=True)
rawdata.shape
```

Out[8]:
```
(4281178, 39)
```

```
In [93]: non_conform = ['applicant_race', 'co_applicant_race']
         for v in non_conform:
             concordances[v+'_1'] = dict(rawdata[[v+'_1', v+'_name_1']].head(10000).drop_duplica
             rawdata.drop(v+'_name_1', axis=1, inplace=True)
         rawdata.shape
```

```
Out[93]: (3584858, 31)
```

More removals:

- We can drop state info, too, since it's all NY.
- I'm going to leave in agency abbreviation and remove the code and name. They didn't fit the pattern for the concordance.
- sequence_number is a unique id for each respondent_id (which is the reporting institution id, and may be relevant)

```
In [9]: rawdata.drop(['state_code', 'state_abbr', 'state_name', 'agency_name', 'agency_code', 's
        rawdata.shape
```

```
Out[9]: (4281178, 33)
```

Finally, we need to simplify the decision codes. We only care about loans that were approved or denied. We can filter out incomplete files, which will presumably be difficult to model anyway, 'Preapprovals' mean a decision was made on the customer's creditworthiness without a specific property attached. At a minimum, these should be modeled separately. For the purposes of this exercise, I'll get rid of those, too. Withdrawn applications should go, too. The remaining actions are 1,2,3, where 3 is a denial and 1 and 2 are approvals.

```
In [10]: rawdata = rawdata.loc[rawdata.action_taken <= 3,:]
         rawdata.shape
```

```
Out[10]: (3584858, 33)
```

```
In [11]: rawdata = rawdata.assign(approved=rawdata.action_taken < 3)
         rawdata.drop("action_taken", axis=1, inplace=True)
```

```
In [94]: rawdata.to_parquet("processed_data.pqt")
```

1.2 Exploratory Data Analysis

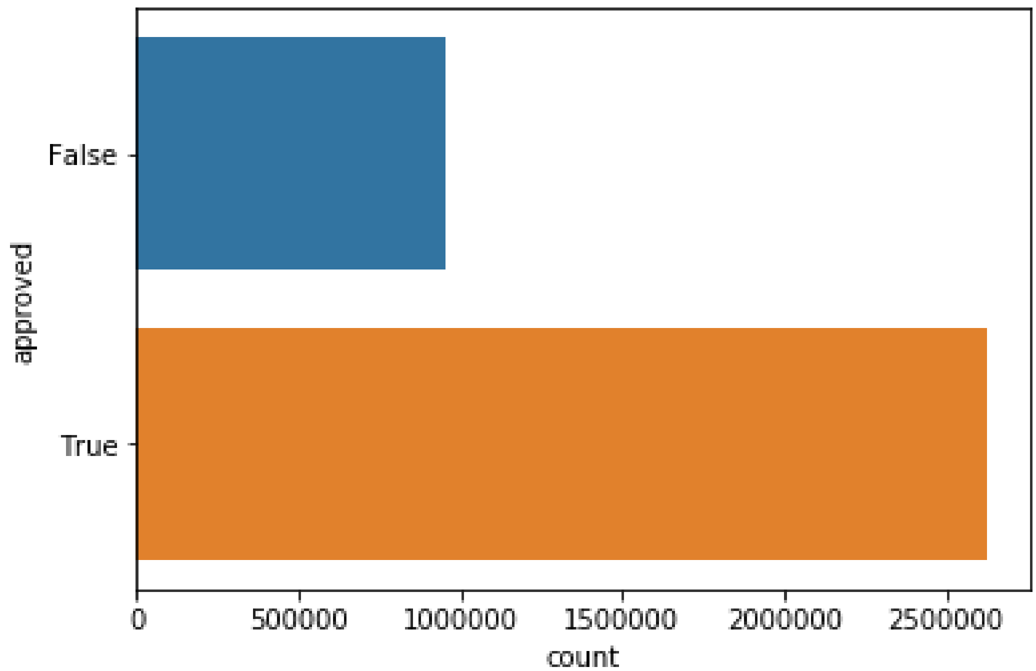
I'm going to start off modeling with an SVM to get a baseline before moving on to more directed regressions, so I'll need to look at distributions to see how to categorize continuous values.

We'll also want to see what our basic prior is on approvals.

```
In [14]: smpl = rawdata.sample(n=10000) #spare our processor if we wind up graphing points
```

```
In [17]: sns.countplot(y="approved", data=rawdata)
```

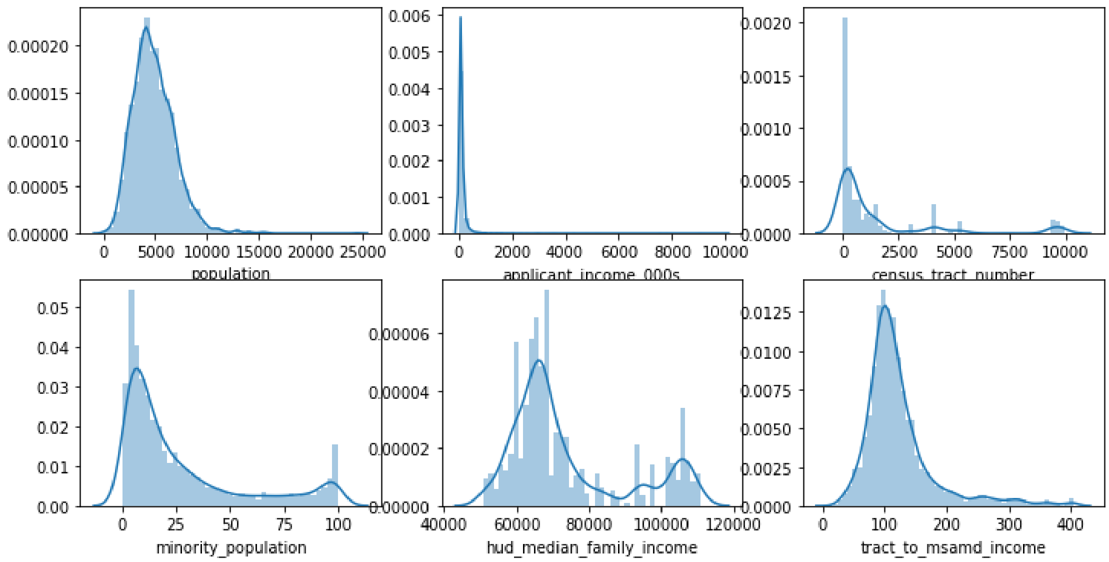
```
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff22df93518>
```



There actually aren't that many continuous/high category values.

```
In [18]: smpl.columns

Out[18]: Index(['agency_abbr', 'applicant_ethnicity', 'applicant_income_000s',
               'applicant_race_1', 'applicant_race_name_1', 'applicant_sex',
               'application_date_indicator', 'as_of_year', 'census_tract_number',
               'co_applicant_ethnicity', 'co_applicant_race_1',
               'co_applicant_race_name_1', 'co_applicant_sex', 'county_code',
               'county_name', 'hoepa_status', 'lien_status', 'loan_purpose',
               'loan_type', 'msamd', 'owner_occupancy', 'preapproval', 'property_type',
               'purchaser_type', 'respondent_id', 'hud_median_family_income',
               'loan_amount_000s', 'number_of_1_to_4_family_units',
               'number_of_owner_occupied_units', 'minority_population', 'population',
               'tract_to_msamd_income', 'approved'],
              dtype='object')

In [58]: lots_of_values = ['population', 'minority_population', 'applicant_income_000s', 'hud_me
                           'census_tract_number', 'tract_to_msamd_income']
         len(lots_of_values)

Out[58]: 6

In [108]: f, axes = plt.subplots(2,3, figsize=(12,6))
          for i, v in enumerate(lots_of_values):
              sns.distplot(smpl[v].dropna(), ax=axes[i % 2, i // 2])
```

```
/home/cnaylor/env/tensorflow/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarnin
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



Someone with an income of 10,000,000 applied for a mortgage? I'm kind of curious how much it was for.

```
In [50]: smpl.loc[smpl.applicant_income_000s>5000, ['applicant_income_000s', 'loan_amount_000s']
```

Out[50]:

	applicant_income_000s	loan_amount_000s
2214445	7545.0	77
2000875	9999.0	1463
1670314	5200.0	3000
3607446	7800.0	710
1846806	9021.0	6750

I'm guessing the first one is a data entry error. Who would go to the trouble?

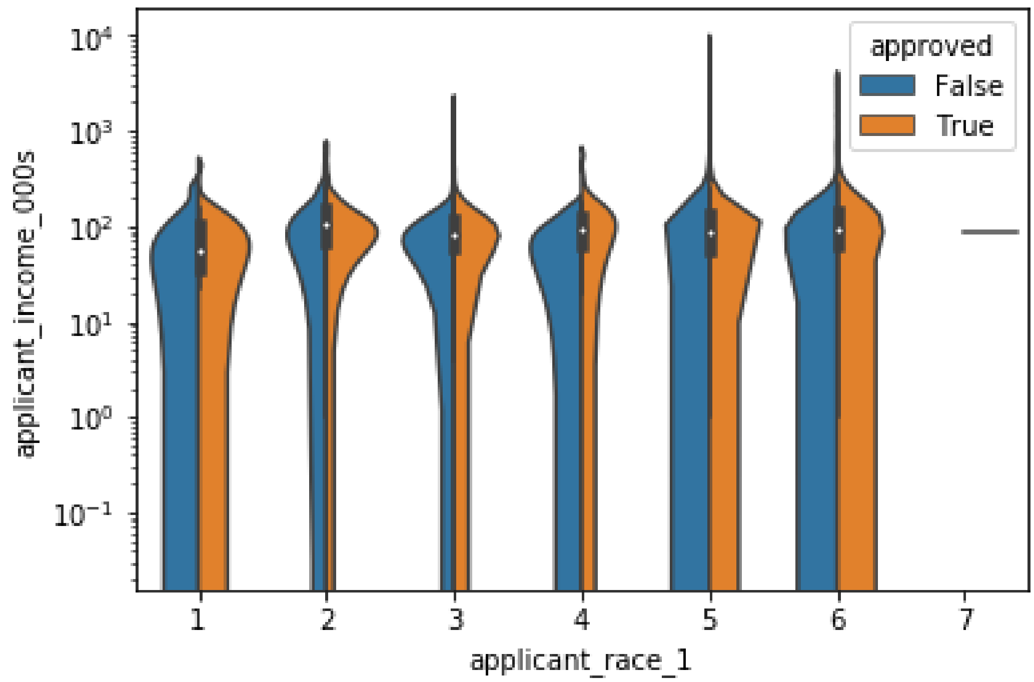
I suppose there must be a tax advantage for the others. Also note income of 9999 implies the data is censored, although I doubt there are enough people in the category to matter.

We can probably set our cutpoints just based on quantiles for these. `census_tract_number` should probably just be dropped for the SVM.

What's the relationship between race and income among mortgage applicants?

```
In [101]: fig, ax = plt.subplots()
          ax.set(yscale="log")
          g = sns.catplot(x='applicant_race_1', y='applicant_income_000s', hue='approved', kind=
                          split=True, data=smpl, ax=ax)
```

```
/home/cnaylor/env/tensorflow/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWarnin
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```




```
In [99]: concordances['applicant_race_1']

Out[99]: {5: 'White',
          6: 'Information not provided by applicant in mail, Internet, or telephone application'
          3: 'Black or African American',
          1: 'American Indian or Alaska Native',
          2: 'Asian',
          4: 'Native Hawaiian or Other Pacific Islander'}
```

The obvious patterns are the disproportionate approval for Asians, and disapproval for African Americans and Pacific Islanders.

1.3 Unsupervised

Let's look at unsupervised clustering to see if there are any natural distinctions in the data we should take into account. Given the size of the data set, PCA will probably be the fastest.

```
In [128]: from sklearn.decomposition import PCA
          from sklearn.preprocessing import StandardScaler
          from sklearn.pipeline import make_pipeline
          pipe = make_pipeline(StandardScaler(), PCA(n_components=5))#we'll only graph the first
          pipe.fit(smpl.select_dtypes(exclude=['object']).dropna())
          pca = pipe.named_steps['pca']
          print(pca.explained_variance_ratio_)
```

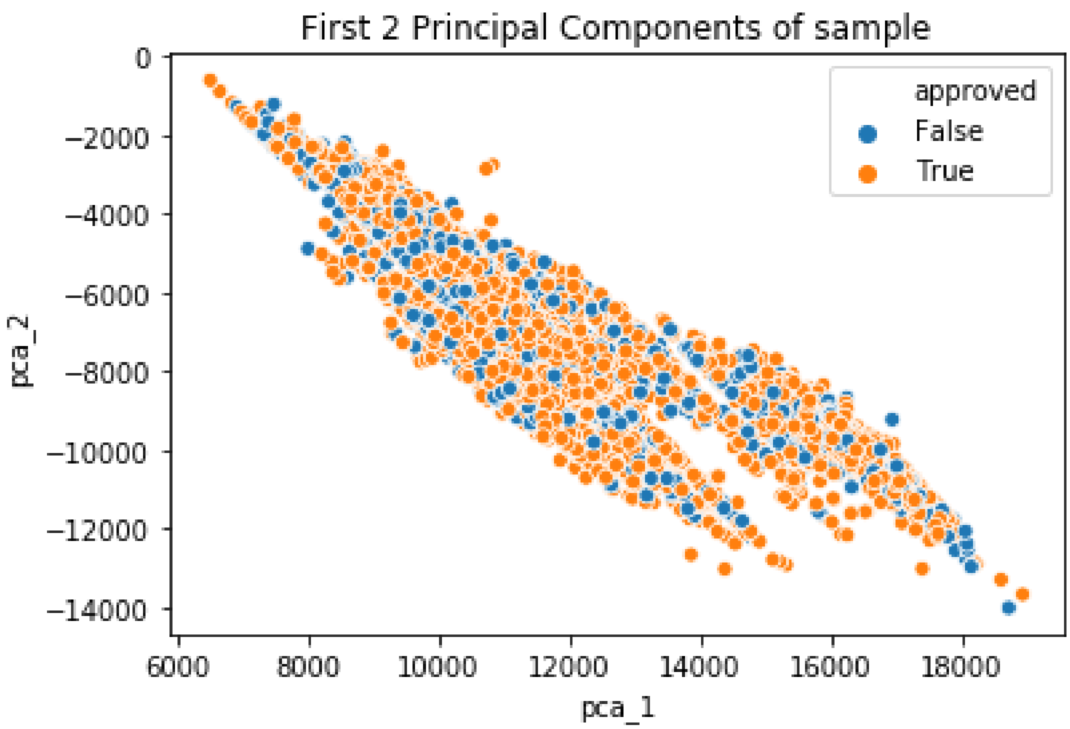
```
[0.12186255 0.10181401 0.08038989 0.07244039 0.06390785]
```

Not too quickly. Let's look at the first two components conditioned on mortgage approval

```
In [140]: pca_hat.shape
          smpl.dropna().approved.shape
```

```
Out[140]: (8151,)
```

```
In [154]: pca_hat = pd.DataFrame(pca.transform(smpl.select_dtypes(exclude=['object']).dropna())[
          pca_hat.loc[:, 'approved'] =  smpl.dropna().approved.values
          ax = sns.scatterplot(x='pca_1', y='pca_2', hue='approved', data=pca_hat)
          ax.set_title("First 2 Principal Components of sample")
          plt.show()
```



Looks like it won't be that simple.
Anyway, on to modeling, which we'll do on a new notebook.

```
In [102]: import pickle
```

```
In [104]: with open('concordance.pkl', 'wb') as f:
              pickle.dump(concordances, f)
```