Analyzing Important Factors in the Loan Approval Process

Neil Chen*, Callie Hartzog†
Department of electrical and Computer Engineering
University of California, San Diego
La Jolla, United States
*pychen@ucsd.edu, †chartzog@ucsd.edu

***Abstract*—This project looks into the factors that hold the most weight when an applicant's loan is approved or denied within the state of California. The motivation for this is to provide more transparency regarding the loan approval process, especially in a time where more and more Californians are looking towards taking out loans to afford buying a home. The data set used in this project is taken from the Consumer Financial Protection Bureau. Factors such as the applicant's race, income, locale, and sex are analyzed. Both decision tree and k-means clustering machine learning models are used to predict whether given loan applications are approved or denied.**

I. Introduction

California has been suffering from a housing crisis for several years now. Matters have only been worsened by the pandemic, causing California to now have the second lowest rate of home ownership [1]. This is due to the fact that housing availability is at an all time low, in turn causing housing prices to sky-rocket. Mortgage rates have also raised, making it harder for people to finance buying a home [2]. In order for people to be able to purchase a home, often they have to take out loans. Loan approval can be influenced by a variety of factors, including: income, race, locale, among others. The purpose of this project is to analyze what factors play the biggest role when a lender decides whether or not to approve an applicant's loan.

II. data Set

The data set used in this project comes from the Consumer Financial Protection Bureau. This data is collected under the Home Mortgage Disclosure Act (HMDA) and contains information regarding individual loan applications [3]. In particular, the data set for this project only contains information on mortgages for single family homes filed in 2017 within the state of California.

The data set has been further filtered to contain features more individualized towards the loan applicant. For example, some of the features kept in the data set include those detailing the applicant's race, sex, yearly income, and location in California (by county) amongst others. Some of the features that were removed were those describing the company giving the loan, the race and sex of any co-applicants, as well as the date that the loan application was filed.

III. Data Analysis

We initially began looking at the characteristics of the loan applicants themselves. The first feature we analyzed was applicant race. We plotted the approval versus denial ratio for each given race in Figure 1 below. For each race, the rates of approved loans versus denied loans are relatively equivalent in value. This leads to the conclusion that there is no racial bias when determining loan approval or denial.

Figure 1. Approval versus denial rates for each race in the data set

We then analyzed approval versus denial rates based on the applicant's sex. Of those who have applied for mortgage loans within California in 2017, 70.28% of them were Male and 29.72% of them were Female. Figure 2 depicts the rates

of approved and denied loan applications for both males and females. As with race, the approval and denial rates for both males and females are relatively equivalent. This again shows that when looking at which loans to approve, the sex of the applicant is not a determining factor.

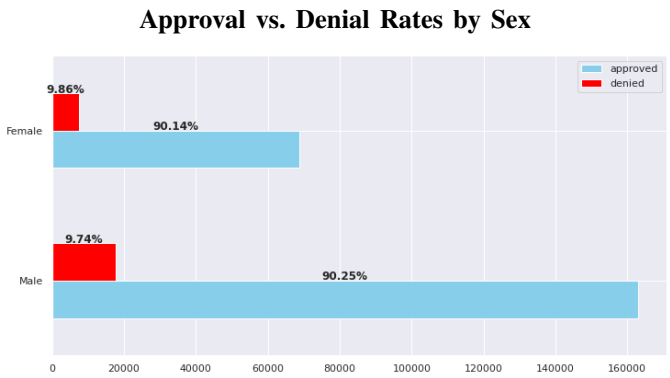


Figure 2. Approval versus denial rates for male and female applicants

We then looked at characteristics of the applications in regards to the counties within California they were filed in. First we analyzed where the various applications originated. Figure 3 shows the number of applications by county. We observed that most applications come from populated counties, such as Los Angeles County, Orange County, and San Diego County. Figure 4 shows that Los Angeles County, Orange County, Santa Clara County, and San Diego county combined account for almost half of the loan applications by loan amount. We infer that the reason is due to the high cost of real estate within these counties. Potential home buyers need to apply for loans in order to purchase houses in these counties.

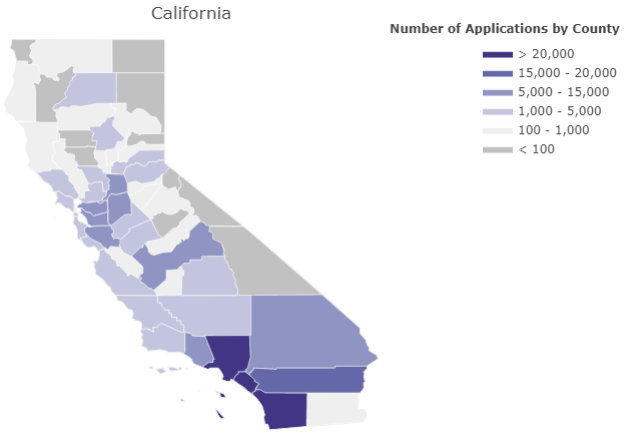


Figure 3. Number of Applicants by County

However, when it comes to the mean loan amount by county, as seen in Figure 5, the counties in the Bay Area stand out. San Mateo County, San Franscisco County, and Marin County have the highest mean loan amounts. This is not surprising due to the high cost of property there. In fact, those counties are

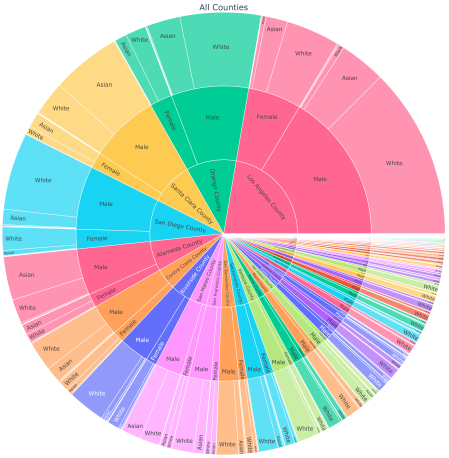


Figure 4. Application Breakdown by Loan Amount

in the top five for highest property prices according to the California Association of Realtors [4].

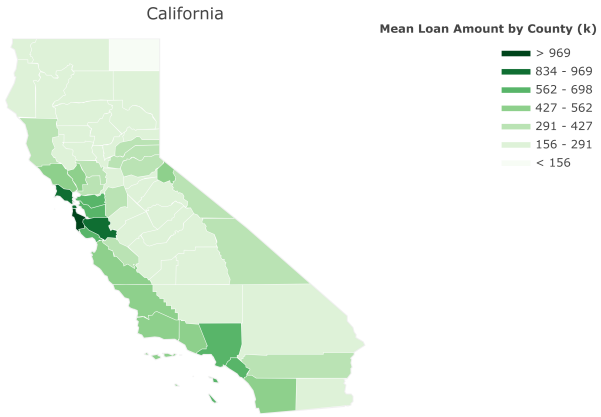


Figure 5. Mean Loan Amount by County

Secondly, we analyzed the mean income of the applicant by county. We can see the mean applicant income in the Bay Area is higher than that of other coastal counties, and that the mean applicant income in coastal counties is higher than that of most inland counties in Figure 6. This phenomenon is accordance with the data collected by the U.S. Census Bureau [5].

Thirdly, we inspected the approval rate by county, shown in Figure 7. Typically, the counties with high approval rates are also counties with high mean applicant incomes. At first glance this may not be apparent as Figures 6 and 7 look quite different. This is due to the fact that counties in the Bay Area have a much higher income than any other counties, making those surrounding it appear to have a lower mean income value. However, if you look at the legend of Figure 6, you can see that many counties still retain a mean applicant income value above $134,000.

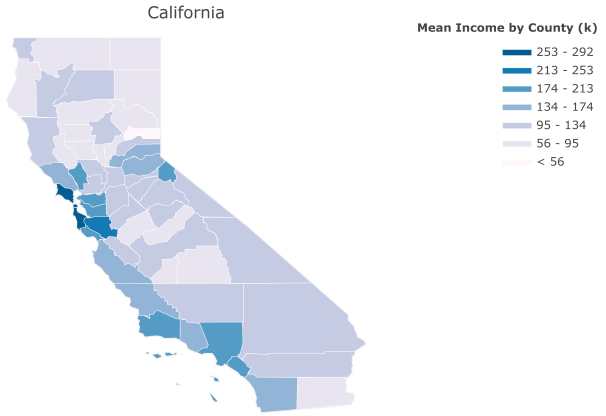


Figure 6. Mean Applicant Income by County

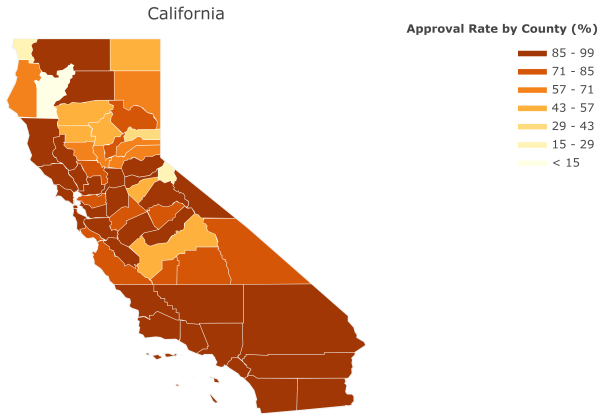


Figure 7. Approval Rate by County

IV. Model Prediction

A. Decision Tree

Decision tree machine learning models are non-parametric supervised models used for regression and classification problems. Our goal is to create a model that can predict whether a loan is approved or denied by learning simple decision rules inferred from the training data features. This model is able to handle both numeric variables (the income of the applicant and the loan amount) and categorical variables (the sex, the race, and the county of the applicant). Furthermore, it uses a white box model, hence we can visualize it and it can be explained easier than a black box model such as the ones used in artificial neural networks.

There are two numeric features in our data set, i.e., the income of the applicant and the loan amount, and there are five categorical features, i.e., the sex, the race, and the county of the applicant. We use built-in LabelEncoder function to transform non-numerical labels to numerical labels.

The model takes training vectors $x_i \in R^n$ where, $i = 1, ..., l$ and a label vector $y \in R^l$. The algorithm recursively partitions the feature space such that the samples of the labels or similar target values are grouped together. We denote the data at node m with Q_m with n_m samples. We denote a splitting policy θ that uses feature j with threshold t_m to split our data into two subsets:

$$Q_m^{left}(\theta) = \{(x, y)|x_j \leq t_m\} \tag{1}$$

$$Q_n^{right}(\theta) = Q_m \setminus Q_m^{left}\} \tag{2}$$

Equation 1 denotes the left subtree, whereas equation 2 denotes the right subtree.

The impurity of the split is defined by loss function H:

$$H(Q_m) = \Sigma_k p_{mk}(1 - p_{mk}) \tag{3}$$

where

$$p_{mk} = \frac{1}{n_m}\Sigma_{y \in Q_m} I(y = k) \tag{4}$$

We pick the optimal splitting policy by minimizing the impurity of the split, i.e.,

$$\theta^* = arg \min_{\theta} G(Q_m, \theta) \tag{5}$$

We keep performing the splitting operation until the maximum allowable depth is reached [6].

The Decision Tree model yields an accuracy of 99.2% when run on the complete data set. One of the benefits of the Decision Tree model is that the decision process can be visualized and then be analyzed. We restricted the number of leaf nodes to 30, otherwise it would be too messy to analyze it.

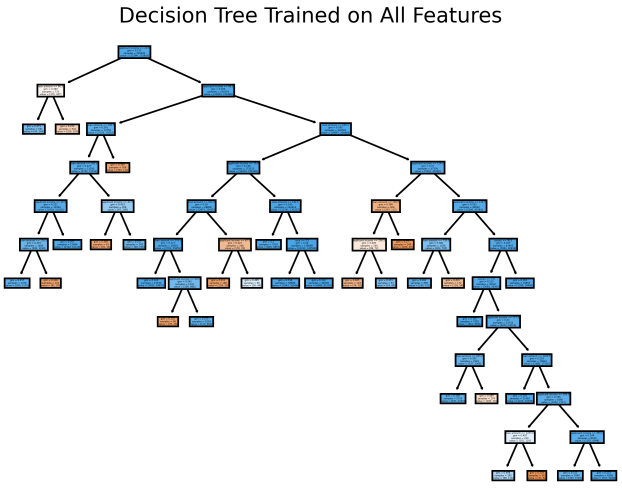


Figure 8. Decision Tree Trained on All Features

The decision tree trained on all features is shown in Figure 8. Looking at the first node (the top node), we can see that the model first splits when the applicant income is less than 20.5k. If the income of the applicant is less than that value, then the model looks whether if the applicant asks for more than 96.5k for his loan. If the applicant asks for more than that threshold, then the algorithm denies the applicant ion.

This makes sense for the bank since the applicant income is less than the median income (41k) and he asks a loan for more than 96.5k, which is more than five-fold of his yearly income. The algorithm basically looks only for the applicant income and the loan amount, that is, if the applicant only applies for a loan amount that is reasonable compared with his income, his loan will be approved.

We further examine whether other factors, such as the sex or the race of the applicant may affect the results. We plot the decision surface of the decision tree models on pair features. *X axis* represents *feature A* and *y axis* represents *feature B*. A blue dot represents the application is approved whereas a red dot means the application is denied. A blue region represents that if an entry falls within it, the model is likely to approve the application based on those two features; whereas a red region represents that the application is likely be denied.

Firstly, we plot the decision surface of the race of the applicant versus the loan amount. The result is shown in Figure 9. *X axis* represents the loan amount and *y axis* represents the race of the applicant. We discover there are several red vertical regions present in the graph. Most vertical red regions stretch from the bottom all the way to the top of the graph, indicating that the dominant factor of being denied is the loan amount in lieu of the race.

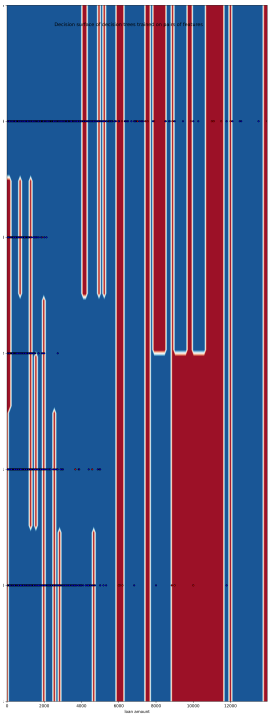


Figure 9. Decision Surface Race vs. Loan Amount

Secondly, we plot the decision surface of the sex of the applicant versus the loan amount. The result is shown in Figure 10. *X axis* represents the loan amount and *y axis* represents the sex of the applicant. Again, several vertical red regions are observed and they expand from the bottom all the way to the top of the graph, suggesting the dominant factor of being denied is the loan amount rather the sex.

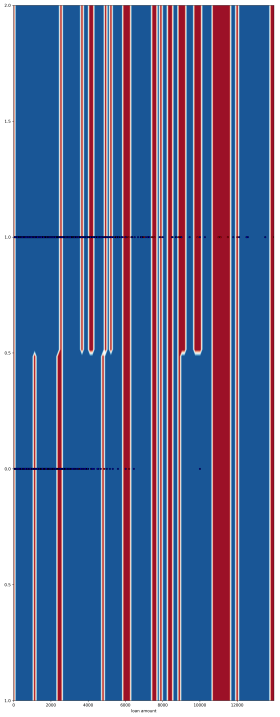


Figure 10. Decision Surface Sex vs. Loan Amount

B. K-means

K-means machine learning models are unsupervised models, meaning that the training data that goes in to the model is not labeled. Essentially, the model does not know in our case which data values constitute approved loans, and which are denied loan applications. The model must figure out on its own how to distinguish approved loans from denied loans. Upon initializing the k-means algorithm, we determine a set number of centroids to cluster our data around, In our case, we use two centroids for approved and denied loan applications.

These centroids are randomly chosen, and then for each data point we assign them to a centroid using the Euclidean Distance formula, $d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$. We continue to re-pick centroids, using the mean of a cluster, and classify data to centroids until the data no longer changes centroids between iterations [7].

The k-means algorithm was run on a one-hot encoded version of the data set. This means that each categorical variable was changed to a numerical column with a value of 1 if that feature was present, and 0 if not. The algorithm yielded an error rate of 17.89% when run on the complete data set. The k-means results, and the assignment of data points to their clusters can be seen in Figure 11 below. The model tended to classify lower loan amounts as approved, while higher amounts were classified as denied. Figure 12 shows which data points were incorrectly classified by the k-means model in yellow. The majority of the incorrectly classified data points in yellow

are found in applicants with lower incomes, regardless of loan amount. One possible reason for misclassification is that k-means looks for a decision boundary or plane to separate the two clusters. However, if you look at the actual distribution of data in Figure 13, you can see that this data set lacks an obvious decision boundary, making it hard for k-means to produce an accurate solution.

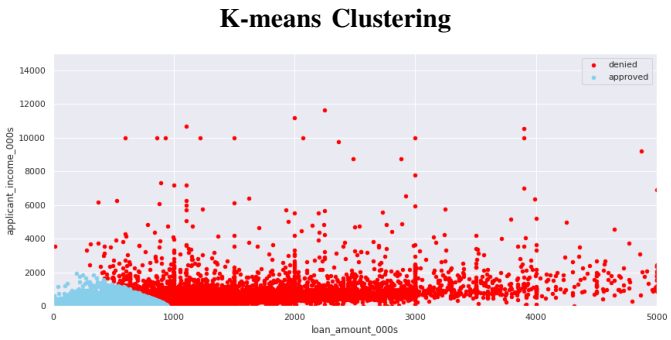


Figure 11. Results of k-means algorithm with approved and denied clusters

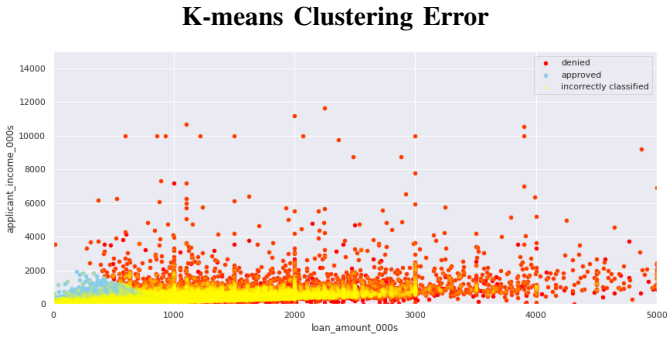


Figure 12. Incorrectly classified values from k-means algorithm

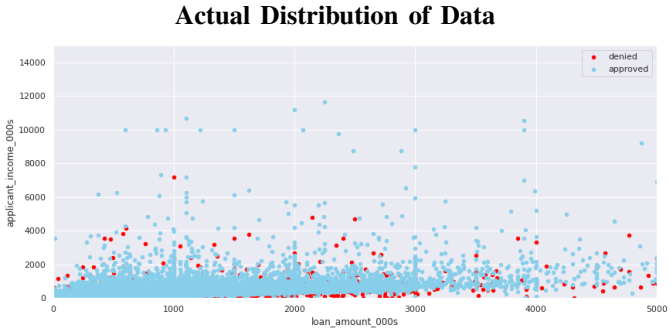


Figure 13. Distribution of approved and denied loans using labels from original data set

V. Conclusion

Both the decision tree and k-means models yielded usable, but not ideal, results. Although the decision tree model achieved 99.2% accuracy, many of the samples were classified as false positives. This means that loan applications that should be denied were being classified as false. This is likely due to the fact that the data set we used has a major class in-balance, and is dominated by instances of accepted loans. The model did not have enough denied data points to accurately make a prediction on unknown data. In the future, this model would likely benefit from down-sampling, where the number of accepted loans is reduced so that the class in-balance is remedied. On the other hand, the k-means model only had an accuracy rate of 82.11% due to the lack of a concrete decision boundary between denied and accepted loan applications. Ultimately, k-means was not a good model for this data set. Moving forward, another unsupervised model could be substituted in place of k-means to achieve a higher accuracy.

References

[1] Dan Walters, "California housing crisis both wide and deep," https://calmatters.org/commentary/2021/12/california-housing-crisis-both-wide-and-deep/, accessed 12-09-2022.

[2] Soumya Karlamangla, "What housing trends emerged in california in 2022," https://www.nytimes.com/2022/12/08/us/housing-trends-california.html, accessed 12-09-2022.

[3] Consumer Financial Protection Bureau, "Hmda data," https://www.consumerfinance.gov/data-research/hmda/historic-data/, accessed 11-17-2022.

[4] Shashank Shekhar, "California median home price by county," https://www.mortgageblog.com/california-median-home-price-county/, accessed 12-06-2022.

[5] index mundi, "California median household income (in 2018 dollars), 2014-2018 by county," https://www.indexmundi.com/facts/united-states/quick-facts/california/median-household-income#table, accessed 12-05-2022.

[6] Scikit Learn, "Decision trees," https://scikit-learn.org/stable/modules/tree.html, accessed 12-05-2022.

[7] Natasha Sharma, "K-means clustering explained," https://neptune.ai/blog/k-means-clustering, accessed 12-10-2022.

import the libraries

```
import os #paths to file
import numpy as np # linear algebra
import pandas as pd # data processing
import warnings# warning filter

!pip install plotly-geo==1.0.0
!pip install geopandas==0.3.0
!pip install pyshp==1.2.10
!pip install shapely==1.6.3

#ploting libraries
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.figure_factory as ff

# #relevant ML libraries
# from sklearn.preprocessing import LabelEncoder
# from sklearn.model_selection import train_test_split
# from sklearn.metrics import confusion_matrix
# from sklearn.metrics import classification_report
# from sklearn.metrics import accuracy_score

# #ML models
# from xgboost import XGBClassifier
# from sklearn.tree import DecisionTreeClassifier
# from sklearn.ensemble import RandomForestClassifier
# from sklearn.linear_model import LogisticRegression

#default theme
sns.set(context='notebook', style='darkgrid', palette='deep', font='sans-serif', font_scale=1, color_codes=False, rc=None)

warnings.filterwarnings("ignore")
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: plotly-geo==1.0.0 in /usr/local/lib/python3.8/dist-packages (1.0.0)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: geopandas==0.3.0 in /usr/local/lib/python3.8/dist-packages (0.3.0)
Requirement already satisfied: shapely in /usr/local/lib/python3.8/dist-packages (from geopandas==0.3.0) (1.6.3)
Requirement already satisfied: fiona in /usr/local/lib/python3.8/dist-packages (from geopandas==0.3.0) (1.8.22)
Requirement already satisfied: pandas in /usr/local/lib/python3.8/dist-packages (from geopandas==0.3.0) (1.3.5)
Requirement already satisfied: pyproj in /usr/local/lib/python3.8/dist-packages (from geopandas==0.3.0) (3.4.0)
Requirement already satisfied: descartes in /usr/local/lib/python3.8/dist-packages (from geopandas==0.3.0) (1.1.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.8/dist-packages (from descartes->geopandas==0.3.0) (3.2.2)
Requirement already satisfied: six>=1.7 in /usr/local/lib/python3.8/dist-packages (from fiona->geopandas==0.3.0) (1.15.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.8/dist-packages (from fiona->geopandas==0.3.0) (57.4.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.8/dist-packages (from fiona->geopandas==0.3.0) (2022.9.24)
Requirement already satisfied: munch in /usr/local/lib/python3.8/dist-packages (from fiona->geopandas==0.3.0) (2.5.0)
Requirement already satisfied: click-plugins>=1.0 in /usr/local/lib/python3.8/dist-packages (from fiona->geopandas==0.3.0) (1.1.1)
Requirement already satisfied: attrs>=17 in /usr/local/lib/python3.8/dist-packages (from fiona->geopandas==0.3.0) (22.1.0)
Requirement already satisfied: cligj>=0.5 in /usr/local/lib/python3.8/dist-packages (from fiona->geopandas==0.3.0) (0.7.2)
Requirement already satisfied: click>=4.0 in /usr/local/lib/python3.8/dist-packages (from fiona->geopandas==0.3.0) (7.1.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-packages (from matplotlib->descartes->geopandas==0.3.0) (0.11.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib->descartes->geop
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.8/dist-packages (from matplotlib->descartes->geopandas==0.3.0) (1.21.6)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib->descartes->geopandas==0.3.0) (1.4.4)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib->descartes->geopandas==0.3.0) (2.8.2
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.8/dist-packages (from pandas->geopandas==0.3.0) (2022.6)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pyshp==1.2.10 in /usr/local/lib/python3.8/dist-packages (1.2.10)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: shapely==1.6.3 in /usr/local/lib/python3.8/dist-packages (1.6.3)
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
!pip install plotly
!pip install plotly-geo==1.0.0
!pip install geopandas==0.3.0
!pip install pyshp==1.2.10
!pip install shapely==1.6.3

import plotly.figure_factory as ff
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: plotly in /usr/local/lib/python3.8/dist-packages (5.5.0)
Requirement already satisfied: six in /usr/local/lib/python3.8/dist-packages (from plotly) (1.15.0)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.8/dist-packages (from plotly) (8.1.0)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: plotly-geo==1.0.0 in /usr/local/lib/python3.8/dist-packages (1.0.0)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: geopandas==0.3.0 in /usr/local/lib/python3.8/dist-packages (0.3.0)
Requirement already satisfied: pandas in /usr/local/lib/python3.8/dist-packages (from geopandas==0.3.0) (1.3.5)
Requirement already satisfied: descartes in /usr/local/lib/python3.8/dist-packages (from geopandas==0.3.0) (1.1.0)
Requirement already satisfied: pyproj in /usr/local/lib/python3.8/dist-packages (from geopandas==0.3.0) (3.4.0)
Requirement already satisfied: fiona in /usr/local/lib/python3.8/dist-packages (from geopandas==0.3.0) (1.8.22)
Requirement already satisfied: shapely in /usr/local/lib/python3.8/dist-packages (from geopandas==0.3.0) (1.6.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.8/dist-packages (from descartes->geopandas==0.3.0) (3.2.2)
Requirement already satisfied: six>=1.7 in /usr/local/lib/python3.8/dist-packages (from fiona->geopandas==0.3.0) (1.15.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.8/dist-packages (from fiona->geopandas==0.3.0) (57.4.0)
Requirement already satisfied: click-plugins>=1.0 in /usr/local/lib/python3.8/dist-packages (from fiona->geopandas==0.3.0) (1.1.1)
Requirement already satisfied: munch in /usr/local/lib/python3.8/dist-packages (from fiona->geopandas==0.3.0) (2.5.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.8/dist-packages (from fiona->geopandas==0.3.0) (2022.9.24)
Requirement already satisfied: cligj>=0.5 in /usr/local/lib/python3.8/dist-packages (from fiona->geopandas==0.3.0) (0.7.2)
Requirement already satisfied: attrs>=17 in /usr/local/lib/python3.8/dist-packages (from fiona->geopandas==0.3.0) (22.1.0)
Requirement already satisfied: click>=4.0 in /usr/local/lib/python3.8/dist-packages (from fiona->geopandas==0.3.0) (7.1.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-packages (from matplotlib->descartes->geopandas==0.3.0) (0.11.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib->descartes->geopandas==0.3.0) (2.
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib->descartes->g
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.8/dist-packages (from matplotlib->descartes->geopandas==0.3.0) (1.21.6)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib->descartes->geopandas==0.3.0) (1.4.4
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.8/dist-packages (from pandas->geopandas==0.3.0) (2022.6)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pyshp==1.2.10 in /usr/local/lib/python3.8/dist-packages (1.2.10)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: shapely==1.6.3 in /usr/local/lib/python3.8/dist-packages (1.6.3)
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
<ipython-input-8-ffa76f62f6cb> in <module>
     13 values = range(len(fips))
     14
---> 15 fig = ff.create_choropleth(fips=fips, values=values)
     16 fig.layout.template = None
     17 fig.show(renderer="colab")

/usr/local/lib/python3.8/dist-packages/plotly/figure_factory/_county_choropleth.py in create_choropleth(fips, values, scope, binning_endpoints,
colorscale, order, simplify_county, simplify_state, asp, show_hover, show_state_data, state_outline, county_outline, centroid_marker,
round_legend_values, exponent_format, legend_title, **layout_options)
    622
    623     if not gp or not shapefile or not shapely:
--> 624         raise ImportError(
    625             "geopandas, pyshp and shapely must be installed for this figure "
    626             "factory.\n\nRun the following commands to install the correct "

ImportError: geopandas, pyshp and shapely must be installed for this figure factory.

Run the following commands to install the correct versions of the following modules:

```
$ pip install geopandas==0.3.0
$ pip install pyshp==1.2.10
$ pip install shapely==1.6.3
```
If you are using Windows, follow this post to properly install geopandas and dependencies:http://geoffboeing.com/2014/09/using-geopandas-
windows/

If you are using Anaconda, do not use PIP to install the packages above. Instead use conda to install them:

```
$ conda install plotly
$ conda install geopandas
```

NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below
```

```
df = pd.read_csv(r'/content/drive/MyDrive/ECE 225A Project/hmda_2017_ca_all-records_labels.csv')
df.head(3)
```

	as_of_year	respondent_id	agency_name	agency_abbr	agency_code	loan_type_name	loan_type	property_type_name	property_type	loan_purpose_r
0	2017	0000451965	Consumer Financial Protection Bureau	CFPB	9	Conventional	1	One-to-four family dwelling (other than manufa...	1	Home improver
			Consumer							

skim the dataset

```
# remove loan applicants that are not of conventional type
df.drop(df[(df.loan_type != 1)].index, inplace=True)

# remove property type that are not of One-to-four family dwelling
df.drop(df[(df.property_type != 1)].index, inplace=True)

# only focus on application for purchasing houses
df.drop(df[(df.loan_purpose != 1)].index, inplace=True)

# remove applicants who failed to complete the process
df.drop(df[(df.action_taken >= 4)].index, inplace=True)

# we only want owner-occupied as a principal dwelling
df.drop(df[(df.owner_occupancy != 1)].index, inplace=True)


race_replacement = {'Black or African American': 'Black',
                    'Information not provided by applicant in mail, Internet, or telephone application': 'NaN',
                    'American Indian or Alaska Native': 'Native',
                    'Native Hawaiian or Other Pacific Islander': 'Islander',
                    'Not applicable': 'NaN'}
df['applicant_race_name_1'].replace(race_replacement, inplace=True)


action_replacement = {'Loan originated': 'Approved',
                      'Application approved but not accepted': 'Approved',
                      'Application denied by financial institution': 'Denied'}
df['action_taken_name'].replace(action_replacement, inplace=True)


sex_replacement = {'Information not provided by applicant in mail, Internet, or telephone application': 'NaN',
                   'Not applicable': 'NaN'}
df['applicant_sex_name'].replace(sex_replacement, inplace=True)


# drop rows that is of NaN
df.replace(r'^\s*$', np.nan, regex=True, inplace=True)
df.replace('', np.nan, regex=True, inplace=True)
df.replace('NaN', np.nan, inplace=True)
df.replace('nan', np.nan, inplace=True)
df.dropna(how='all', inplace=True)


# drop rows that do not cantain the income info
subset = ['agency_abbr',
'loan_type_name',
'property_type_name',
'loan_purpose_name',
'owner_occupancy_name',
'loan_amount_000s',
'action_taken_name',
'county_name',
'applicant_race_name_1',
'applicant_sex_name',
'applicant_income_000s']
df.dropna(subset=subset, inplace=True)


# denote the indices of columns that we would like to drop
idx_want = [3, 5, 7, 9, 11, 13, 16, 23, 30, 50, 54, 57]
idx_remove = list(range(0, 78))
for x in idx_want:
    idx_remove.remove(x)

df.drop(df.columns[idx_remove], axis = 1, inplace=True)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 235042 entries, 11 to 1706698
Data columns (total 13 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
```

```
 0   agency_abbr            235042 non-null  object
 1   loan_type_name         235042 non-null  object
 2   property_type_name     235042 non-null  object
 3   loan_purpose_name      235042 non-null  object
 4   owner_occupancy_name   235042 non-null  object
 5   loan_amount_000s       235042 non-null  float64
 6   action_taken_name      235042 non-null  object
 7   county_name            235042 non-null  object
 8   applicant_race_name_1  235042 non-null  object
 9   applicant_sex_name     235042 non-null  object
 10  applicant_income_000s  235042 non-null  float64
 11  denial_reason_name_1   16562 non-null   object
 12  FIP                    235042 non-null  object
dtypes: float64(2), object(11)
memory usage: 25.1+ MB
```

verify that we skimmed the dataset properly

```
pd.unique(df['agency_abbr'])
```

```
array(['HUD', 'NCUA', 'CFPB', 'OCC', 'FDIC', 'FRS'], dtype=object)
```

```
pd.unique(df['loan_type_name'])
```

```
array(['Conventional'], dtype=object)
```

```
pd.unique(df['property_type_name'])
```

```
array(['One-to-four family dwelling (other than manufactured housing)'],
      dtype=object)
```

```
pd.unique(df['loan_purpose_name'])
```

```
array(['Home purchase'], dtype=object)
```

```
pd.unique(df['action_taken_name'])
```

```
array(['Approved', 'Denied'], dtype=object)
```

```
pd.unique(df['owner_occupancy_name'])
```

```
array(['Owner-occupied as a principal dwelling'], dtype=object)
```

```
pd.unique(df['county_name'])
```

```
array(['San Bernardino County', 'Fresno County', 'Sacramento County',
       'Santa Cruz County', 'Alameda County', 'Riverside County',
       'Contra Costa County', 'Butte County', 'Orange County',
       'Santa Clara County', 'Los Angeles County', 'Colusa County',
       'Placer County', 'Solano County', 'San Luis Obispo County',
       'El Dorado County', 'San Benito County', 'Calaveras County',
       'Amador County', 'San Diego County', 'Monterey County',
       'Kern County', 'Humboldt County', 'Yolo County', 'Ventura County',
       'Merced County', 'San Mateo County', 'Glenn County',
       'Stanislaus County', 'San Francisco County', 'Kings County',
       'Marin County', 'Napa County', 'Alpine County', 'Del Norte County',
       'Sonoma County', 'Sutter County', 'Shasta County',
       'Santa Barbara County', 'Madera County', 'Tulare County',
       'San Joaquin County', 'Siskiyou County', 'Nevada County',
       'Yuba County', 'Tehama County', 'Mariposa County',
       'Tuolumne County', 'Mendocino County', 'Trinity County',
       'Plumas County', 'Mono County', 'Modoc County', 'Lake County',
       'Sierra County', 'Lassen County', 'Inyo County', 'Imperial County'],
      dtype=object)
```

```
pd.unique(df['applicant_race_name_1'])
```

```
array(['White', 'Native', 'Black', 'Asian', 'Islander'], dtype=object)
```

```
df.describe()
```

	loan_amount_000s	applicant_income_000s
count	235042.000000	235042.000000
mean	502.709865	164.432744
std	417.462947	214.178501
min	1.000000	1.000000

```
df.reset_index(drop=True)
```

	agency_abbr	loan_type_name	property_type_name	loan_purpose_name	owner_occupancy_nam
0	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as principal dwellir
1	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as principal dwellir
2	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as principal dwellir
3	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as principal dwellir
4	NCUA	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as principal dwellir
...	...	...	...	...	
235037	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as principal dwellir
235038	CFPB	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as principal dwellir
235039	NCUA	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as principal dwellir
235040	CFPB	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as principal dwellir
235041	CFPB	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as principal dwellir

235042 rows × 13 columns

```
df['action_taken_name'].value_counts()
```

```
Approved    212843
Denied       22199
Name: action_taken_name, dtype: int64
```

```
overall_approved_rate = df['action_taken_name'].value_counts()['Approved'] / (df['action_taken_name'].value_counts()['Approved'] + df['action_taken_name
overall_approved_rate
```

```
0.9055530500931749
```

```
df.nunique()
```

```
agency_abbr                6
loan_type_name             1
property_type_name         1
loan_purpose_name          1
owner_occupancy_name       1
loan_amount_000s        2721
action_taken_name          2
county_name               58
applicant_race_name_1      5
applicant_sex_name         2
applicant_income_000s   1871
denial_reason_name_1       9
FIP                       58
dtype: int64
```

```
labels = {'Approved': 1, 'Denied' : 0}
df['approved'] = df['action_taken_name']
df['approved'].replace(labels, inplace=True)
```

```
06071    11707
06019     5408
06067    10935
06087     1328
06001    10986
06065    15494
06013     8893
06007     1448
06059    19025
06085    11944
06011       76
06061     4059
06095     3113
06079     1687
06017     1946
06069      485
06005      260
06073    18792
06009      326
06037    39710
06053     1519
06029     4245
06023      488
06113     1223
06111     4986
06047     1204
06081     4309
06021       89
06099     3288
06075     3748
06031      384
06041     1818
06055      680
06015       64
06097     2728
06101      536
06089     1025
06083     1832
06039      624
06107     1774
06077     5142
06093      184
06057      737
06103      230
06115      426
06043       68
06109      331
06045      283
06063      102
06003        4
06051       85
06049       15
06105       33
06033      306
06091       17
06035       86
06027       55
06025      553
```

```
approved_by_county.sum()
```

```
212843
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 235042 entries, 11 to 1706698
Data columns (total 14 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   agency_abbr            235042 non-null  object
 1   loan_type_name         235042 non-null  object
 2   property_type_name     235042 non-null  object
 3   loan_purpose_name      235042 non-null  object
 4   owner_occupancy_name   235042 non-null  object
 5   loan_amount_000s       235042 non-null  float64
 6   action_taken_name      235042 non-null  object
 7   county_name            235042 non-null  object
 8   applicant_race_name_1  235042 non-null  object
 9   applicant_sex_name     235042 non-null  object
 10  applicant_income_000s  235042 non-null  float64
 11  denial_reason_name_1   16562 non-null   object
 12  FIP                    235042 non-null  object
 13  approved               235042 non-null  int64
```

```
dtypes: float64(2), int64(1), object(11)
memory usage: 26.9+ MB
```

```
denied_by_county = df[df['approved'] == 0]['county_name'].value_counts(sort=False)
denied_by_county.sum()
```

```
22199
```

```
df.groupby(['county_name'])['loan_amount_000s'].mean()
```

```
Alpine County             449.666667
Amador County             277.241259
Butte County              260.208041
Calaveras County          257.697548
Colusa County             241.011628
Contra Costa County       572.258025
Del Norte County          253.148649
El Dorado County          402.312356
Fresno County             254.475351
Glenn County              197.275510
Humboldt County           278.833021
Imperial County           170.991763
Inyo County               304.242424
Kern County               227.024280
Kings County              226.085308
Lake County               225.314706
Lassen County             187.989583
Los Angeles County        595.828336
Madera County             235.535613
Marin County              920.416581
Mariposa County           246.653846
Mendocino County          338.766026
Merced County             234.808511
Modoc County              156.000000
Mono County               325.367347
Monterey County           492.502969
Napa County               555.151394
Nevada County             351.388551
Orange County             609.741518
Placer County             399.205719
Plumas County             215.064220
Riverside County          301.680373
Sacramento County         288.563656
San Benito County         417.743243
San Bernardino County     281.160074
San Diego County          517.746625
San Francisco County      946.297699
San Joaquin County        299.302951
San Luis Obispo County    449.362846
San Mateo County          969.650279
Santa Barbara County      556.442403
Santa Clara County        846.312171
Santa Cruz County         587.570937
Shasta County             251.685973
Sierra County             166.227273
Siskiyou County           189.756219
Solano County             331.781696
Sonoma County             497.428765
Stanislaus County         264.286833
Sutter County             253.801336
Tehama County             213.665399
Trinity County            221.657895
Tulare County             206.218832
Tuolumne County           239.461957
Ventura County            512.120325
Yolo County               358.040510
Yuba County               219.799578
Name: loan_amount_000s, dtype: float64
```

```
df = df.reset_index(drop=True)
df
```

	agency_abbr	loan_type_name	property_type_name	loan_purpose_name	owner_occupancy_name	loan_amount_000s	action_taken_name	county_na
0	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	238.0	Approved	S Bernardi Cour
1	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	123.0	Approved	Fres Cour
2	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	372.0	Approved	Sacramer Cour
3	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	636.0	Approved	Santa Cr Cour
4	NCUA	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	324.0	Approved	Alame Cour
...	...	...	...	...	...	...	...	
235037	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	300.0	Denied	San Die Cour
235038	CFPB	Conventional	One-to-four family dwelling (other than	Home purchase	Owner-occupied as a principal dwelling	400.0	Denied	San Die Cour

```
county_df = df.groupby(['county_name'])
county_df
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7f1ead3bcc10>
```

```
county_df
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7f1ead3bcc10>
```

235041	CFPB	Conventional	One-to-four family dwelling (other than	Home purchase	Owner-occupied as a principal dwelling	461.0	Denied	San Die Cour

Models

Preprocessing

```
encoded_data = pd.get_dummies(df)
```

```
encoded_data
```

	loan_amount_000s	applicant_income_000s	approved	agency_abbr_CFPB	agency_abbr_FDIC
0	238.0	90.0	1	0	0
1	123.0	100.0	1	0	0
2	372.0	84.0	1	0	0
3	636.0	189.0	1	0	0
4	324.0	82.0	1	0	0
...	...	...	...	...	...
235037	300.0	124.0	0	0	0
235038	400.0	130.0	0	1	0
235039	716.0	139.0	1	0	0
235040	200.0	48.0	1	1	0
235041	461.0	137.0	0	1	0

235042 rows × 147 columns

```
from sklearn import preprocessing
```

```
label_encoded_data = df.apply(preprocessing.LabelEncoder().fit_transform)
```

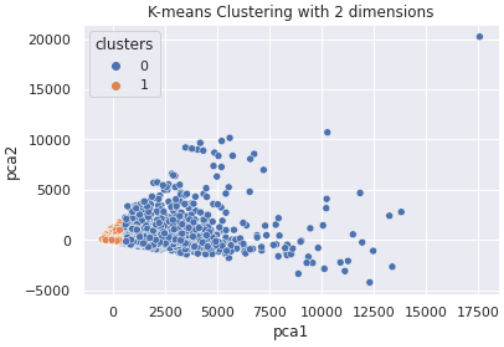
```
label_encoded_data
```

	agency_abbr	loan_type_name	property_type_name	loan_purpose_name	owner_occupancy_nam
0	3	0	0	0	
1	3	0	0	0	
2	3	0	0	0	
3	3	0	0	0	
4	4	0	0	0	
...	...	...	...	...	
235037	3	0	0	0	
235038	0	0	0	0	
235039	4	0	0	0	
235040	0	0	0	0	
235041	0	0	0	0	

K-means

```
from sklearn.cluster import KMeans

encoded_data_labels = encoded_data['approved']
encoded_data = encoded_data.drop(columns=['approved'])

km = KMeans(
    n_clusters=2, init='random',
    n_init=10, max_iter=300,
    tol=1e-04, random_state=42
)
encoded_data['clusters'] = km.fit_predict(encoded_data)
```

```
from sklearn.decomposition import PCA

### Run PCA on the data and reduce the dimensions in pca_num_components dimensions

pca_num_components = 2

reduced_data = PCA(n_components=pca_num_components).fit_transform(encoded_data)
results = pd.DataFrame(reduced_data,columns=['pca1','pca2'])

sns.scatterplot(x="pca1", y="pca2", hue=encoded_data['clusters'], data=results)
plt.title('K-means Clustering with 2 dimensions')
plt.show()
```




```
encoded_data[encoded_data['clusters'] == 1]
```

	loan_amount_000s	applicant_income_000s	agency_abbr_CFPB	agency_abbr_FDIC	agency_abb
0	238.0	90.0	0	0	
1	123.0	100.0	0	0	
2	372.0	84.0	0	0	
3	636.0	189.0	0	0	
4	324.0	82.0	0	0	

```
encoded_data
```

	loan_amount_000s	applicant_income_000s	agency_abbr_CFPB	agency_abbr_FDIC	agency_abb
0	238.0	90.0	0	0	
1	123.0	100.0	0	0	
2	372.0	84.0	0	0	
3	636.0	189.0	0	0	
4	324.0	82.0	0	0	
...	...	...	...	...	
235037	300.0	124.0	0	0	
235038	400.0	130.0	1	0	
235039	716.0	139.0	0	0	
235040	200.0	48.0	1	0	
235041	461.0	137.0	1	0	

235042 rows × 147 columns

```
encoded_data_labels
```

```
0         1
1         1
2         1
3         1
4         1
         ..
235037    0
235038    0
235039    1
235040    1
235041    0
Name: approved, Length: 235042, dtype: int64
```

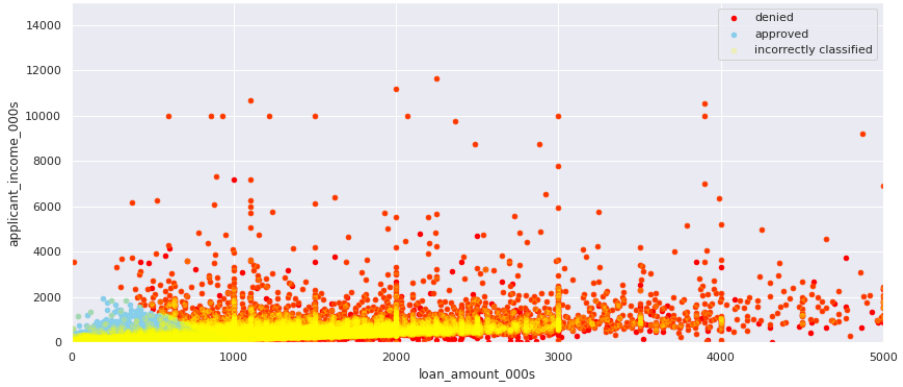
```
error_percentage = 0

for i in range(len(encoded_data['clusters'])):
  if encoded_data['clusters'][i] != encoded_data_labels[i]:
    error_percentage += 1

error_percentage/len(encoded_data['clusters'])
```

```
0.1789254686396474
```

```
wrong_values = encoded_data[encoded_data['clusters'] != encoded_data_labels].reset_index(drop=True)

wrong_values
```

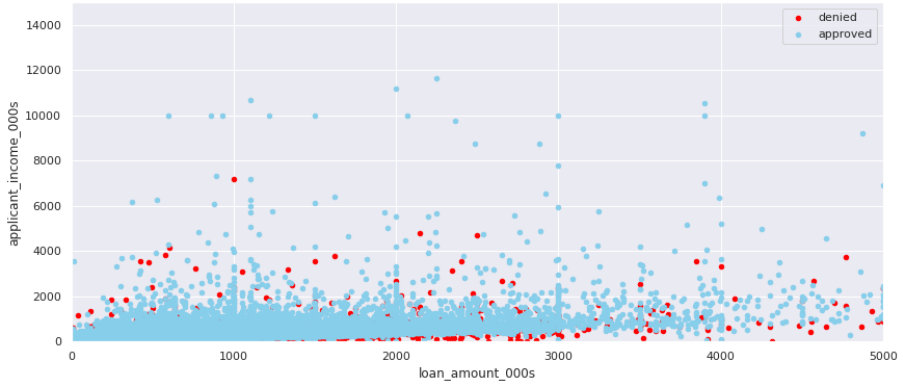
	loan_amount_000s	applicant_income_000s	agency_abbr_CFPB	agency_abbr_FDIC	agency_abbr_FRS	agency_abbr_HUD	agency_abbr_NCUA	agency_a
0	270.0	65.0	1	0	0	0	0	
1	256.0	134.0	0	0	0	1	0	
2	182.0	78.0	1	0	0	0	0	
3	424.0	228.0	0	0	0	1	0	
4	321.0	173.0	0	0	0	1	0	
...	...	...	...	...	...	...	...	

```
ax1 = encoded_data[encoded_data['clusters'] == 0].plot(kind='scatter', x='loan_amount_000s', y='applicant_income_000s', color='red', figsize=(14,6), lab
ax2 = encoded_data[encoded_data['clusters'] == 1].plot(kind='scatter', x='loan_amount_000s', y='applicant_income_000s', color='skyblue', ax=ax1, label='
plt.xlim([0, 5000])
plt.ylim([0,15000])
ax3 = encoded_data[encoded_data['clusters'] != encoded_data_labels].plot(kind='scatter', x='loan_amount_000s', y='applicant_income_000s', color='yellow'
```



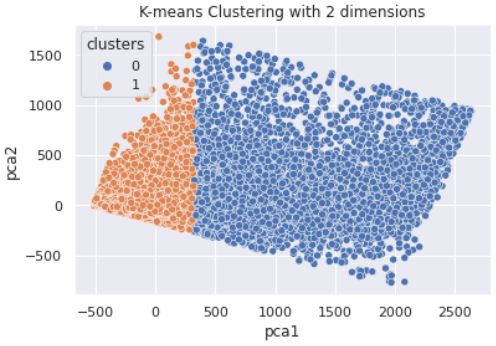

```
ax1 = encoded_data[encoded_data['approved'] == 0].plot(kind='scatter', x='loan_amount_000s', y='applicant_income_000s', color='red', figsize=(14,6), lab
ax2 = encoded_data[encoded_data['approved'] == 1].plot(kind='scatter', x='loan_amount_000s', y='applicant_income_000s', color='skyblue', ax=ax1, label='
plt.xlim([0, 5000])
plt.ylim([0,15000])
```

```
(0.0, 15000.0)
```




```
label_encoded_data_labels = label_encoded_data['approved']
label_encoded_data = label_encoded_data.drop(columns=['approved'])

km = KMeans(
    n_clusters=2, init='random',
    n_init=10, max_iter=300,
    tol=1e-04, random_state=42
)
label_encoded_data['clusters'] = km.fit_predict(label_encoded_data)
```

```
### Run PCA on the data and reduce the dimensions in pca_num_components dimensions

pca_num_components = 2

reduced_data = PCA(n_components=pca_num_components).fit_transform(label_encoded_data)
results = pd.DataFrame(reduced_data,columns=['pca1','pca2'])

sns.scatterplot(x="pca1", y="pca2", hue=label_encoded_data['clusters'], data=results)
plt.title('K-means Clustering with 2 dimensions')
plt.show()
```




```
error_percentage = 0

for i in range(len(label_encoded_data['clusters'])):
  if label_encoded_data['clusters'][i] != label_encoded_data_labels[i]:
    error_percentage += 1

error_percentage/len(label_encoded_data['clusters'])
```

```
0.19529275618825573
```

import the libraries

```
import os #paths to file
import numpy as np # linear algebra
import pandas as pd # data processing
import warnings# warning filter


#ploting libraries
import matplotlib.pyplot as plt
import seaborn as sns

# #relevant ML libraries
# from sklearn.preprocessing import LabelEncoder
# from sklearn.model_selection import train_test_split
# from sklearn.metrics import confusion_matrix
# from sklearn.metrics import classification_report
# from sklearn.metrics import accuracy_score

# #ML models
# from xgboost import XGBClassifier
# from sklearn.tree import DecisionTreeClassifier
# from sklearn.ensemble import RandomForestClassifier
# from sklearn.linear_model import LogisticRegression

#default theme
sns.set(context='notebook', style='darkgrid', palette='deep', font='sans-serif', font_scale=1, color_codes=False, rc=None)

#warning hadle
warnings.filterwarnings("ignore")


df = pd.read_csv(r'/content/drive/MyDrive/ECE 225A Project/hmda_2017_ca_all-records_labels.csv')
df.head(3)
```

	as_of_year	respondent_id	agency_name	agency_abbr	agency_code	loan_type_name	loan_type
0	2017	0000451965	Consumer Financial Protection Bureau	CFPB	9	Conventional	1
1	2017	0000146672	Consumer Financial Protection Bureau	CFPB	9	Conventional	1
2	2017	0000451965	Consumer Financial Protection Bureau	CFPB	9	Conventional	1

3 rows × 78 columns

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# remove loan applicants that are not of conventional type
df.drop(df[(df.loan_type != 1)].index, inplace=True)

# remove property type that are not of One-to-four family dwelling
df.drop(df[(df.property_type != 1)].index, inplace=True)

# only focus on application for purchasing houses
df.drop(df[(df.loan_purpose != 1)].index, inplace=True)

# remove applicants who failed to complete the process
df.drop(df[(df.action_taken >= 4)].index, inplace=True)

# we only want owner-occupied as a principal dwelling
df.drop(df[(df.owner_occupancy != 1)].index, inplace=True)


# denote the indices of columns that we would like to drop
idx_want = [3, 5, 7, 9, 11, 13, 16, 18, 23, 30, 50, 54, 57]
idx_remove = list(range(0, 78))
for x in idx_want:
```

```
    idx_remove.remove(x)

df.drop(df.columns[idx_remove], axis = 1, inplace=True)

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 274567 entries, 2 to 1706698
Data columns (total 13 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   agency_abbr           274567 non-null  object 
 1   loan_type_name        274567 non-null  object 
 2   property_type_name    274567 non-null  object 
 3   loan_purpose_name     274567 non-null  object 
 4   owner_occupancy_name  274567 non-null  object 
 5   loan_amount_000s      274509 non-null  float64
 6   action_taken_name     274567 non-null  object 
 7   msamd_name            269185 non-null  object 
 8   county_name           274358 non-null  object 
 9   applicant_race_name_1 274567 non-null  object 
 10  applicant_sex_name    274567 non-null  object 
 11  applicant_income_000s 271167 non-null  float64
 12  denial_reason_name_1  20596 non-null   object 
dtypes: float64(2), object(11)
memory usage: 29.3+ MB
```

verify that we skimmed the dataset properly

```
pd.unique(df['agency_abbr'])
```

```
array(['CFPB', 'HUD', 'NCUA', 'FDIC', 'OCC', 'FRS'], dtype=object)
```

```
pd.unique(df['loan_type_name'])
```

```
array(['Conventional'], dtype=object)
```

```
pd.unique(df['property_type_name'])
```

```
array(['One-to-four family dwelling (other than manufactured housing)'],
      dtype=object)
```

```
pd.unique(df['loan_purpose_name'])
```

```
array(['Home purchase'], dtype=object)
```

```
pd.unique(df['owner_occupancy_name'])
```

```
array(['Owner-occupied as a principal dwelling'], dtype=object)
```

```
df.describe()
```

	loan_amount_000s	applicant_income_000s
count	274509.000000	271167.000000
mean	521.031485	173.482426
std	1887.919612	1119.388664
min	1.000000	1.000000
25%	282.000000	81.000000
50%	418.000000	123.000000
75%	618.000000	192.000000
max	475000.000000	542821.000000

```
df = df.reset_index(drop=True)

df
```

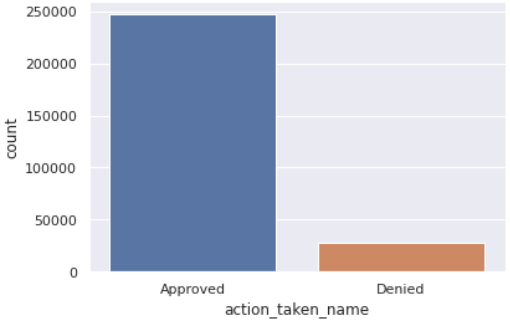
	agency_abbr	loan_type_name	property_type_name	loan_purpose_name	owner_occupancy_name	loan_amount_000s	action_taken_name	msamd_nam
0	CFPB	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	1079.0	Loan originated	Oaklan Haywar Berkeley C
1	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	238.0	Loan originated	Riversid Sa Bernardin Ontario - C
2	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	123.0	Loan originated	Fresno - C
3	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	372.0	Loan originated	Sacrament Rosevill Arde Arcade - C
4	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	636.0	Loan originated	Santa Cru Watsonvill - C
...	...	...	...	...	...	...	...	
274562	NCUA	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	716.0	Application approved but not accepted	San Dieg Carlsbad C
274563	CFPB	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	837.0	Loan originated	San Dieg Carlsbad C
274564	CFPB	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	200.0	Loan originated	San Dieg Carlsbad C
274565	FDIC	Conventional	One-to-four family dwelling (other than	Home purchase	Owner-occupied as a principal dwelling	760.0	Loan originated	San Dieg Carlsbad

```
action_replacement = {'Loan originated': 'Approved',
                      'Application approved but not accepted': 'Approved',
                      'Application denied by financial institution': 'Denied'}
df['action_taken_name'].replace(action_replacement, inplace=True)
```

274567 rows × 13 columns

```
sns.countplot(x=df["action_taken_name"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff380959f70>
```




```
sex_replacement = {'Information not provided by applicant in mail, Internet, or telephone application': 'NaN',
                   'Not applicable': 'NaN'}
df['applicant_sex_name'].replace(sex_replacement, inplace=True)
```

```
sns.countplot(x=df["applicant_sex_name"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff3808d37f0>
```

```
pd.unique(df['applicant_race_name_1'])
```

```
array(['Information not provided by applicant in mail, Internet, or telephone application',
       'White', 'American Indian or Alaska Native',
       'Black or African American', 'Asian',
       'Native Hawaiian or Other Pacific Islander', 'Not applicable'],
      dtype=object)
```

```
race_replacement = {'Black or African American': 'Black',
                    'Information not provided by applicant in mail, Internet, or telephone application': 'NaN',
                    'American Indian or Alaska Native': 'Native',
                    'Native Hawaiian or Other Pacific Islander': 'Islander',
                    'Not applicable': 'NaN'}
df['applicant_race_name_1'].replace(race_replacement, inplace=True)
pd.unique(df['applicant_race_name_1'])
```

```
array(['NaN', 'White', 'Native', 'Black', 'Asian', 'Islander'],
      dtype=object)
```

```
df['applicant_race_name_1']
```

```
0            NaN
1          White
2          White
3         Native
4          White
           ...
274562     White
274563       NaN
274564    Native
274565       NaN
274566     White
Name: applicant_race_name_1, Length: 274567, dtype: object
```

```
labels = {'Approved': 1, 'Denied' : 0}

df['approved'] = df['action_taken_name']

df['approved'].replace(labels, inplace=True)

df = df.reset_index(drop=True)

df
```

	agency_abbr	loan_type_name	property_type_name	loan_purpose_name	owner_occupancy_name	loan_amount_000s	action_taken_name	msamd_nam
0	CFPB	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	1079.0	Approved	Oaklanc Haywarc Berkeley C

Riversid

```
df
```

	agency_abbr	loan_type_name	property_type_name	loan_purpose_name	owner_occupancy_name	loan_amount_000s	action_taken_name	msamd_nam
0	CFPB	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	1079.0	Approved	Oaklanc Haywarc Berkeley C
1	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	238.0	Approved	Riverside Sa Bernardinc Ontario - C
2	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	123.0	Approved	Fresno - C
3	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	372.0	Approved	Sacramento Roseville Arder Arcade - C
4	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	636.0	Approved	Santa Cruz Watsonvill - C
...	...	...	...	...	...	...	...	.
274562	NCUA	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	716.0	Approved	San Diego Carlsbad C
274563	CFPB	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	837.0	Approved	San Diego Carlsbad C
274564	CFPB	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	200.0	Approved	San Diego Carlsbad C
274565	FDIC	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	760.0	Approved	San Diego Carlsbad C
274566	CFPB	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	461.0	Denied	San Diego Carlsbad C

274567 rows × 14 columns

```
approved = df[df['approved'] == 1]

approved = approved[approved['applicant_sex_name'] != 'NaN']

approved
```

	agency_abbr	loan_type_name	property_type_name	loan_purpose_name	owner_occupancy_name	loan_amount_000s	action_taken_name	msamd_nam
1	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	238.0	Approved	Riverside Sa Bernardin Ontario - C
2	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	123.0	Approved	Fresno - C
3	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	372.0	Approved	Sacramento Roseville Arden Arcade - C
4	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	636.0	Approved	Santa Cru Watsonvill - C
5	NCUA	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	324.0	Approved	Oaklan Hayward Berkeley C

```
denied = df[df['approved'] == 0]

denied = denied[denied['applicant_sex_name'] != 'NaN']

denied
```

	agency_abbr	loan_type_name	property_type_name	loan_purpose_name	owner_occupancy_name	loan_amount_000s	action_taken_name	msamd_nam
14	FDIC	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	226.0	Denied	Oaklan Hayward Berkeley C
25	CFPB	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	270.0	Denied	Sacramento Roseville Arden Arcade - C
27	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	256.0	Denied	Sacramento Roseville Arden Arcade - C
52	CFPB	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	182.0	Denied	Sacramento Roseville Arden Arcade - C
59	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	1104.0	Denied	Lo Angeles Long Beach Glendale C
...	...	...	...	...	...	...	...	.
274552	CFPB	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	769.0	Denied	San Diego Carlsbad C
274555	OCC	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	79.0	Denied	San Diego Carlsbad C
274560	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	300.0	Denied	San Diego Carlsbad C
274561	CFPB	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	400.0	Denied	San Diego Carlsbad C
274566	CFPB	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	461.0	Denied	San Diego Carlsbad C

25117 rows × 14 columns

```
approved_race_values = approved['applicant_race_name_1'].dropna().value_counts()

approved_race_values
```

```
White       153646
Asian        52561
```

```
Black         6187
Islander      1796
Native        1550
Name: applicant_race_name_1, dtype: int64
```

```
denied_race_values = denied['applicant_race_name_1'].value_counts()

denied_race_values
```

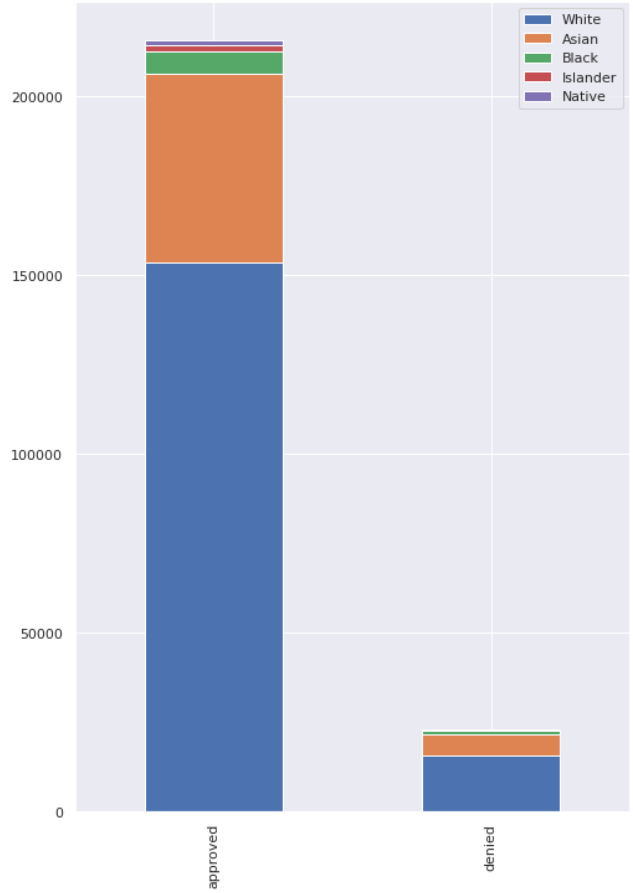
```
White       15480
Asian        5914
Black        1058
Islander      230
Native        223
Name: applicant_race_name_1, dtype: int64
```

```
race_values = pd.DataFrame(columns=['approved','denied'])

race_values['approved'] = approved_race_values
race_values['denied'] = denied_race_values


race_values
```

	approved	denied
White	153646	15480
Asian	52561	5914
Black	6187	1058
Islander	1796	230
Native	1550	223

```
(race_values.transpose()).plot.bar(stacked=True, figsize=(8,12))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff37ca24460>
```




```
import matplotlib.pyplot as plt

explode = (0.05, 0.05)
titles = ["White","Asian","Black","Islander","Native"]
```

```
for i in range(len(race_values['approved'])):

  plt.pie(race_values.iloc[i], colors=['skyblue', 'red'], labels=["Approved", "Denied"],
          autopct='%1.1f%%',pctdistance=0.4,
          explode=explode)

  # draw circle
  centre_circle = plt.Circle((-0.05, 0), 0.70, fc='white')
  fig = plt.gcf()

  # Adding Circle in Pie chart
  fig.gca().add_artist(centre_circle)

  # Adding Title of chart
  plt.title(titles[i])

  # Displaying Chart
  plt.show()
```

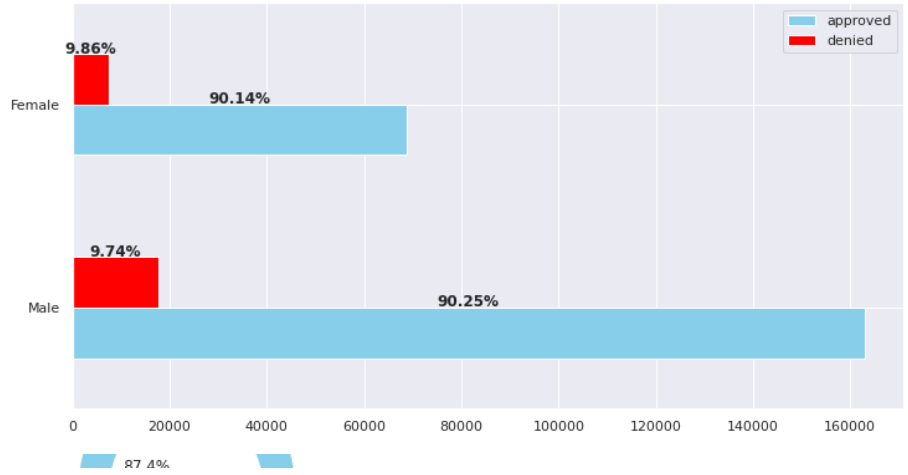
White

```
approved_sex_values = approved['applicant_sex_name'].value_counts()

denied_sex_values = denied['applicant_sex_name'].value_counts()
```

```
sex_values = pd.DataFrame(columns=['approved','denied'])

sex_values['approved'] = approved_sex_values
sex_values['denied'] = denied_sex_values

sex_values
```

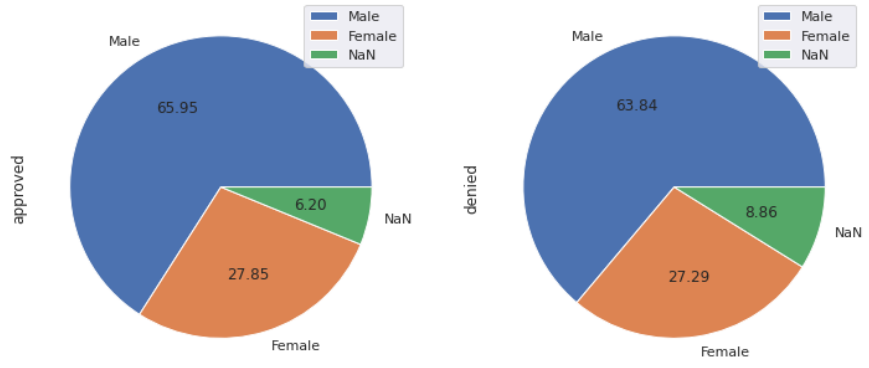
	approved	denied
Male	162893	17595
Female	68799	7522

```
percentages = ['90.25%', "90.14%", "9.74%", "9.86%"]
#graph = (sex_values).plot.barh(figsize=(20,12), color=['skyblue', 'red'])
graph = sex_values.plot(kind='barh', color=['skyblue', 'red'], figsize=(12,6))

i = 0
for p in graph.patches:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_xy()
    plt.text(x+width/2,
             y+height*1.01,
             percentages[i],
             ha='center',
             weight='bold')
    i+=1
plt.show()
```




```
sex_values.plot.pie(subplots=True, figsize=(12,16),autopct="%.2f")
```

```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7ff38023cdf0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7ff3803bbb20>],
      dtype=object)
```



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 274567 entries, 0 to 274566
Data columns (total 14 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   agency_abbr            274567 non-null  object 
 1   loan_type_name         274567 non-null  object 
 2   property_type_name     274567 non-null  object 
 3   loan_purpose_name      274567 non-null  object 
 4   owner_occupancy_name   274567 non-null  object 
 5   loan_amount_000s       274509 non-null  float64
 6   action_taken_name      274567 non-null  object 
 7   msamd_name             269185 non-null  object 
 8   county_name            274358 non-null  object 
 9   applicant_race_name_1  274567 non-null  object 
 10  applicant_sex_name     274567 non-null  object 
 11  applicant_income_000s  271167 non-null  float64
 12  denial_reason_name_1   20596 non-null   object 
 13  approved               274567 non-null  int64  
dtypes: float64(2), int64(1), object(11)
memory usage: 29.3+ MB
```

```
numerical_data = df.select_dtypes(exclude=["object"])
```

```
numerical_data
```

	loan_amount_000s	applicant_income_000s	approved
0	1079.0	278.0	1
1	238.0	90.0	1
2	123.0	100.0	1
3	372.0	84.0	1
4	636.0	189.0	1
...	...	...	...
274562	716.0	139.0	1
274563	837.0	405.0	1
274564	200.0	48.0	1
274565	760.0	229.0	1
274566	461.0	137.0	0

274567 rows × 3 columns

```
sns.heatmap(numerical_data.corr(), annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff3801bca00>
```


```
df[df['approved'] == 1].groupby(['county_name'])['loan_amount_000s'].mean()
```

```
county_name
Alameda County          648.156888
Alpine County           344.285714
Amador County           277.294498
Butte County            257.665195
Calaveras County        254.989446
Colusa County           237.310345
```

```
Contra Costa County        571.976946
Del Norte County           255.492958
El Dorado County           399.117144
Fresno County              253.771289
Glenn County               200.437500
Humboldt County            275.291096
Imperial County            174.647469
Inyo County                296.200000
Kern County                231.010164
Kings County               230.736715
Lake County                228.739726
Lassen County              195.210526
Los Angeles County         604.123890
Madera County              235.412104
Marin County               933.677230
Mariposa County            246.582278
Mendocino County           333.242424
Merced County              235.935780
Modoc County               131.117647
Mono County                345.663366
Monterey County            482.800827
Napa County                563.443456
Nevada County              348.174208
Orange County              605.232541
Placer County              395.459491
Plumas County              218.009091
Riverside County           301.973818
Sacramento County          290.707708
San Benito County          420.369403
San Bernardino County      283.030636
San Diego County           516.702619
San Francisco County       971.103996
San Joaquin County         304.150441
San Luis Obispo County     445.723393
San Mateo County          1001.368453
Santa Barbara County       560.154871
Santa Clara County         850.009489
Santa Cruz County          589.177062
Shasta County              252.431635
Sierra County              171.600000
Siskiyou County            187.048544
Solano County              333.181241
Sonoma County              493.026617
Stanislaus County          264.666574
Sutter County              249.134583
Tehama County              215.927481
Trinity County             228.390244
Tulare County              208.064599
Tuolumne County            241.024129
Ventura County             517.756382
Yolo County                262.690700
```

```
df[df['approved'] == 0].groupby(['county_name'])['loan_amount_000s'].mean()
```

```
county_name
Alameda County             669.244755
Alpine County              636.000000
Amador County              277.526316
Butte County               273.590278
Calaveras County           277.760000
Colusa County              265.454545
Contra Costa County        615.443470
Del Norte County           221.642857
El Dorado County           460.173611
Fresno County              271.423077
Glenn County               180.200000
Humboldt County            333.076923
Imperial County            154.671875
Inyo County                315.538462
Kern County                208.504000
Kings County               261.571429
Lake County                215.977273
Lassen County              145.300000
Los Angeles County         714.447425
Madera County              243.078431
Marin County              1094.238411
Mariposa County            231.583333
Mendocino County           404.888889
Merced County              263.829787
Modoc County               169.400000
Mono County                309.857143
Monterey County            628.353234
Napa County                527.178571
Nevada County              385.935185
Orange County              725.155750
Placer County              451.164384
Plumas County              233.181818
Riverside County           335.104139
Sacramento County          292.623671
San Benito County          418.952381
San Bernardino County      309.202643
```

```
San Diego County           597.334418
San Francisco County       904.042781
San Joaquin County         300.852744
San Luis Obispo County     518.489011
San Mateo County           988.952381
Santa Barbara County       675.338776
Santa Clara County         856.590344
Santa Cruz County          633.190476
Shasta County              222.631579
Sierra County              304.500000
Siskiyou County            201.304348
Solano County             5534.723039
Sonoma County              606.858657
Stanislaus County          283.198347
Sutter County              256.987013
Tehama County              208.205128
Trinity County             333.555556
Tulare County              190.774194
Tuolumne County            263.843137
Ventura County             577.357654
Yolo County                346.035714
```

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.ensemble import AdaBoostClassifier

import warnings
warnings.filterwarnings("ignore")


df = pd.read_csv("skimmed_dataframe.csv")


# transform non-numeric labels to numeric labels
# https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
le_agency = preprocessing.LabelEncoder()
le_county = preprocessing.LabelEncoder()
le_race = preprocessing.LabelEncoder()
le_sex = preprocessing.LabelEncoder()

df['agency_num']  = le_agency.fit_transform(df['agency_abbr'])
df['county_num']  = le_county.fit_transform(df['county_name'])
df['race_num']  = le_race.fit_transform(df['applicant_race_name_1'])
df['sex_num']  = le_sex.fit_transform(df['applicant_sex_name'])

df.head()
```

	agency_abbr	loan_type_name	property_type_name	loan_purpose_name	owner_occupancy_name	lo
0	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	
1	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	
2	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	
3	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	
4	NCUA	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling	

```
# create the features (drop the output and whatnots)
X = df.drop(['approved', 'agency_num', 'action_taken_name', 'loan_type_name', 'property_type_name', 'loan_purpose_name', 'owner_occupancy_name',
                  'denial_reason_name_1', 'FIPS', 'agency_abbr', 'county_name', 'applicant_race_name_1', 'applicant_sex_name'], axis=1)
X
```

	loan_amount_000s	applicant_income_000s	county_num	race_num	sex_num
0	238.0	90.0	35	4	1
1	123.0	100.0	9	4	1
2	372.0	84.0	33	3	0
3	636.0	189.0	43	4	0
4	324.0	82.0	0	1	0
...	...	...	...	...	...
235037	300.0	124.0	36	4	1
235038	400.0	130.0	36	4	0
235039	716.0	139.0	36	4	1
235040	200.0	48.0	36	3	0
235041	461.0	137.0	36	4	1

235042 rows × 5 columns

```
y = df['approved']
y
```

```
0         1
1         1
2         1
3         1
4         1
          ..
235037    0
235038    0
235039    1
235040    1
235041    0
Name: approved, Length: 235042, dtype: int64
```

```
# 80-20 split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=True)


# https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html
clf = AdaBoostClassifier(n_estimators=100, random_state=0)

# learn the digits on the train subset
clf.fit(X_train, y_train)

# predict the value of the digit on the test subset
predicted = clf.predict(X_test)


clf.score(X_train, y_train)
```

```
0.9055378577164647
```

```
from sklearn.metrics import classification_report

y_true = y_test.to_list()
y_pred = predicted.tolist()

target_names = ['class 0', 'class 1']
print(classification_report(y_true, y_pred, target_names=target_names))
```

```
                  precision    recall  f1-score   support

         class 0       0.59      0.01      0.02      4389
         class 1       0.91      1.00      0.95     42620

        accuracy                           0.91     47009
       macro avg       0.75      0.50      0.48     47009
    weighted avg       0.88      0.91      0.86     47009
```

```
import os #paths to file
import pandas as pd # data processing
import sklearn
import warnings # warning filter

from sklearn import preprocessing
import matplotlib.pyplot as plt


warnings.filterwarnings("ignore")


# load the skimmed dataframe
df = pd.read_csv('skimmed_dataframe.csv')

df = sklearn.utils.shuffle(df)


# transform non-numeric labels to numeric labels
le_agency = preprocessing.LabelEncoder()
le_county = preprocessing.LabelEncoder()
le_race = preprocessing.LabelEncoder()
le_sex = preprocessing.LabelEncoder()

df['agency_num']  = le_agency.fit_transform(df['agency_abbr'])
df['county_num']  = le_county.fit_transform(df['county_name'])
df['race_num']  = le_race.fit_transform(df['applicant_race_name_1'])
df['sex_num']  = le_sex.fit_transform(df['applicant_sex_name'])

df.head()
```

	agency_abbr	loan_type_name	property_type_name	loan_purpose_name	owner_occupancy_name
38655	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling
826	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling
46739	NCUA	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling
85097	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling
70872	HUD	Conventional	One-to-four family dwelling (other than manufa...	Home purchase	Owner-occupied as a principal dwelling

```
# create the features (drop the output and whatnots)
X = df.drop(['approved', 'agency_num', 'action_taken_name', 'loan_type_name', 'property_type_name', 'loan_purpose_name', 'owner_occupancy_name',
                  'denial_reason_name_1', 'FIPS', 'agency_abbr', 'county_name', 'applicant_race_name_1', 'applicant_sex_name'], axis=1)
X
```

	loan_amount_000s	applicant_income_000s	county_num	race_num	sex_num
38655	452.0	121.0	0	0	1
826	317.0	59.0	33	0	1
46739	540.0	157.0	29	4	1
85097	305.0	95.0	32	4	1
70872	300.0	70.0	48	4	1
...	...	...	...	...	...
119266	150.0	44.0	14	4	1
201404	1076.0	454.0	36	0	1
42375	224.0	66.0	32	0	0
157326	518.0	233.0	0	0	1
137036	471.0	160.0	18	1	1

235042 rows × 5 columns

```
Y = df['approved']
Y
```

```
38655     1
826       1
46739     1
85097     1
70872     1
         ..
119266    1
201404    1
42375     1
157326    0
137036    1
Name: approved, Length: 235042, dtype: int64
```

```
from sklearn import tree
clf = tree.DecisionTreeClassifier(max_leaf_nodes=30, random_state=0)

clf.fit(X, Y)
```

```
DecisionTreeClassifier(max_leaf_nodes=30, random_state=0)
```

```
clf.score(X, Y)
```

```
0.9097905906178471
```

```
# graph the decision tree
from sklearn.tree import plot_tree

plot_tree(clf, feature_names=['loan amount', 'applicant income', 'county', 'race', 'sex'], filled=True)
plt.title("Decision Tree Trained on All Features")
plt.savefig("loan-applicantion-decision-tree-truncated.png", dpi=500)
plt.show()
```