CS156 Assignment 3
Yoel Ferdman
Prof Sterne

Variables included in the model

- i. I included:
    1. Loan amount: the total amount requested by the borrower.
    2. Debt-to-income-ratio: A ratio calculated using the borrower's total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower's self-reported monthly income.
    3. Employment length: Years from 0 to 10 of length of time employed.
    4. Of course - How much money was funded. If they were funded fully from the loan dataset- it was just that variable of how much they were funding (occasionally slightly varied from the amount requested) and if it was in the reject data it was zero.
- ii. I wish I could have included a lot of the other available variables such as risk score, verification status, and loan status, and more. The problem with these are that they are not in both datasets – reject and loaned. The only way we can make meaningful predictions about whether a loan will be accepted or rejected based on certain variables is if we have data from that variable for both rejected clients and accepted. I was only left with 3 variables: the amount they requested, dti, and employment length. Unfortunately this led to a small amount of available variables for modeling.

2. any cleaning or transformations that you carried out on the data
    - i. In order to combine the data I had to make the names of the variables that I was going to use from both datasets the same format and spelling so that I can concatenate easily. I had to make sure that when combining the datasets, they had the same amount of rows so that when I run my regression it doesn't outweigh either model, so I proportionately skipped importing rows randomly in each dataset to make them equal.
    - ii. I had to multiply the dti by 100 to format it similarly in both datasets.
    - iii. I had to convert categorical data to dummy variables, notably the employment length. While it would be nice to have data that used exact numbers for us to know employment length, this dataset provided strings with estimations of "<1 year" or ">8 years" which is difficult to convert. I figured I would just use it as categorical rather than process the text and output numbers.
3. the type of model you used and any settings that the model required
    - i. I used a regular linear regression from scikit learn to model the data. It didn't require special settings, but I did have to make sure that the

datasets I was importing were formatted correctly. I also started off by cutting down my data points to a small proportion of what was available so that it would run more smoothly and faster.

4. the training method you used, and any techniques that you used to avoid overfitting the data
    i. I split the data into a test set and training set before training it so that I can test the validity of the model using train_test_split from scikit learn. I used a test size of 40% meaning I only used 60% of the data to train the model. (I actually used less than this because when I imported the data I only imported a proportion of it because we were dealing with such large datasets, it was too big for the model to use if I import it all.
5. an estimate of how well the model will perform on unseen data
    i. I tested the model on "unseen data" (ahem… *my test set*) and got so-so results. I got an R^2 score ("variance score") ranging from .32 to .38 where 1 is a perfect score (predicts output perfectly). This isn't that good. I would hope for at least .7 or .8 minimum if I'm going to use this for something. I think the problem is that I treated this as a linear dataset where it might be better modelled using some other model. For example regression discontinuity design (from CS112) might be very useful here. If we had a score that the banks/loaners had given each client, both reject and those who received loans as to how "loanable" they were, with a strict cutoff, then we can use RDD to compare those really close to the cutoff, above and below to see what difference it would make. This would tell us a lot about the cut off people. But what about everyone else? While my model gives a solid number to help predict how much one should ask for to be sure they will get that amount of money, it might be very skewed to those who were rejected. Let's say I'm a great candidate for a loan and I ask for $5000. My model might tell you to only ask for $4000 or $3500. (This is seen in the code at the bottom). The reason for this is because it takes into account all those in the reject data with similar profiles (in this case, similar dti and employment length). Just because those two variables might be similar to many rejects doesn't mean that they are a bad candidate. In a better model I would include a lot more variables to avoid this issue. However for the constraints of the assignment, I successfully created a calculator (based on the data) that helps one predict how much they can successfully get funded.

CODE: https://gist.github.com/anonymous/b3c04ab05ac46495cb6cb743e5f41bec

```
import numpy as np
import matplotlib.pyplot as plt
import random
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_predict
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import warnings
warnings.filterwarnings(action="ignore", module="scipy", message="^internal gelsd")



#cleaning and transforming the data:

loan_data = "LoanStats3a.csv"
reject_data = "RejectStats_updated.csv"



# length of data
num_lines = sum(1 for i in open(loan_data))
num_lines_r = 65500

#  size of our data that we want to use
size = int(num_lines / 9)
size_r = int(float(num_lines_r) / float(13.85568))

# The row indices to skip to make data smaller
skip_idx = random.sample(range(1, num_lines), num_lines - size)
skip_idx_r = random.sample(range(1, num_lines_r), num_lines_r - size_r)


# Read the data
loan_sample = pd.read_csv(loan_data, skiprows=skip_idx, usecols = ['loan_amnt',
'loan_status','total_pymnt','verification_status','dti', 'emp_length', 'funded_amnt'] )
reject_sample = pd.read_csv(reject_data, skiprows = skip_idx_r, usecols =['loan_amnt',
'Risk_Score', 'dti','emp_length', 'funded_amnt'])

#change categorical data to dummies so we can run regression
dum_loan_sample = pd.get_dummies(loan_sample,columns=['verification_status',
'loan_status','emp_length'])
dum_reject_sample = pd.get_dummies(reject_sample,columns=['emp_length'])


#independent and dependent variables
```

```
ind_var_one = dum_loan_sample[['loan_amnt','dti', 'emp_length_2 years', 'emp_length_3 years',
'emp_length_4 years', 'emp_length_5 years', 'emp_length_6 years', 'emp_length_7 years',
'emp_length_8 years', 'emp_length_9 years', 'emp_length_< 1 year']]
ind_var_two = dum_reject_sample[['loan_amnt', 'dti','emp_length_2 years', 'emp_length_3
years', 'emp_length_4 years', 'emp_length_5 years', 'emp_length_6 years', 'emp_length_7
years', 'emp_length_8 years', 'emp_length_9 years', 'emp_length_< 1 year']]

dep_var_one = dum_loan_sample[['funded_amnt']]
dep_var_two = dum_reject_sample[['funded_amnt']]

ind_var_list = [ind_var_one, ind_var_two]
dep_var_list = [dep_var_one, dep_var_two]

ind_var = pd.concat(ind_var_list)
dep_var = pd.concat(dep_var_list)

#splitting the data to train/test to prevent overfitting
X_train, X_test, y_train, y_test = train_test_split(ind_var, dep_var, test_size=0.4,
random_state=0)

#fitting the model//Create linear model
reg=LinearRegression()
reg.fit(X_train,y_train)


#evaluating its performance

y_pred = reg.predict(X_test)

y_pred_2 = reg.predict([[5000,50,0,0,0,0,0,0,0,1,0]])

print 'Good Candidate for $5k should ask for' , y_pred_2

print 'R^2 score is', reg.score(X_test,y_test)
#coefficients
print('Coefficients: \n', reg.coef_)
# The mean squared error
print("Mean squared error: %.2f"
    % mean_squared_error(y_test, y_pred))

# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % r2_score(y_test, y_pred))
```

```
Results:
Good Candidate for $5k should ask for [[ 3710.60271865]]
R^2 score is 0.348508894253
('Coefficients: \n', array([[  5.61494170e-01,  -2.65791155e-02,  -1.31799341e+03,
        -2.07920302e+02,  -1.77981592e+02,   1.20076106e+03,
         1.20719695e+02,  -5.11929220e+02,  -6.29417498e+02,
         5.32334455e+02,  -2.50142827e+03]]))
Mean squared error: 35301526.61
Variance score: 0.35
```