Module 2 Final Assignment by Jaeyuel Park and Bruce Benson

April 6, 2022

1 The Assignment

In this project, you will create models to understand two questions. Firstly, what variables influence whether a property owner appeals? Secondly, what influences the monetary reduction they gain from appeal? Your goal is to understand whether there is bias or unfairness in the tax system. Specific requirements of the study:

- Provide an executive summary describing your conclusions.
- Create models that estimate two things: the probability that a property owner will appeal; and the monetary reduction on their taxes if they win an appeal.
- Describe your analysis and inference throughout–how you chose the variables you used in your model, the hypothesis testing you conducted to prove signifciance, etc.
- Describe the effect that each indepndent varaible has on the outcome variable.
- Use visualizations as appropriate.
- Provide a conclusion that describes your method and the data you used, as well as potential next steps–for example, additional data you would want to collect to test your conclusions.

2 I. Executive Summary

(This is not complete, I usually write this last.)

Aside from routine data cleansing and prepration, the anaysis is in two parts. Part I will develop a model for predicting the probability of appeal and what variables drive the prediction. Note this is a binary outcome problem on the 'appeal' dependent variable, since a homeowner either appeals or doesn't.

The second part develops a model for predicting the amount of the adjustment when there is an appeal. It's probable that various biases will be revealed in this analysis, informing the question as to whether the tax appeals system is biased.

2.1 II. Exploring and Preparing the Data

[1]:
```
import pandas as pd
from scipy import stats
pd.options.mode.chained_assignment = None  # default='warn'
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from pylab import *
```

```
df = pd.read_csv(r"C:\Users\bbenson\OneDrive - FTI Consulting\Coursera␣
↪Courses\U Chicago Statistical Thinking and ML\Module 2 Final␣
↪Assignment\project2data.csv")
#df.to_csv(r'C:\Users\bbenson\OneDrive - FTI Consulting\Coursera Courses\U␣
↪Chicago Statistical Thinking and ML\copyofproject2data.csv', index = False)
```

2.2 II.A Initial Data Cleanup

A visual scan of the data shows there are many NAs in the appeal field because NA is legitimate when there was no appeal. (Of the entire data set of 19k+ records, there are only about 4,000 appeals.) There are other columns that have missing values. These are largely in the census-related fields or characteristics of the home itself. We'll drop these rows.

There are also rows which have zero or negative values in the (sale) value, medhinc, and taxes fields. These rows will be eliminated. While there may be cases where zero may be valid (taxes for example), they aren't staitically useful.

[2]:
```
len(df)
```

[2]: 19036

[3]:
```
df1 = df.
↪dropna(subset=['white','black','asian','hispanic','poverty','squarefoot','beds','college','
len(df1)
```

[3]: 13877

Droping rows with NAN reduces row count from 19,036 to 13,899.

[4]:
```
# Correct for negative av1
df1["av1_corr"] = np.abs(df1['av1'])
```

[5]:
```
# droping values < = 0
for x in df1.index:
    if df1.loc[x,"value"] <= 0:
        df1.drop(x,inplace = True)
    elif df1.loc[x,"taxes"] <= 0:
        df1.drop(x,inplace = True)
    elif df1.loc[x,"av1_corr"] <= 0:
        df1.drop(x,inplace = True)
    elif df1.loc[x,"medhinc"] <= 0:
        df1.drop(x,inplace = True)
len(df1)
```

[5]: 13827

Creating dummy variables

[6]:
```
##Categorical Variables to dummy variables
dummy_walk = pd.get_dummies(df['walkfac'])
df = pd.concat([df,dummy_walk],axis=1)
dummy_tri = pd.get_dummies(df['tri'])
df = pd.concat([df,dummy_tri],axis=1)
dummy_condo = pd.get_dummies(df['condo'])
df = pd.concat([df,dummy_condo],axis=1)

##Renaming Car-Dep:CD Somewhat Walkable:SW VeryWalkable:Vw Walker' Paradaise:WP␣
↪Northwest countY: NSCC Southwest county: SSCC
df.rename(columns={"Car-Dependent":"CD", "Somewhat Walkable": "SW", "Very␣
↪Walkable":"VW", "Walker's Paradise":"WP"},inplace=True)
df.rename(columns={"Northwest Suburban Cook County":"NSCC", "Southwest Suburban␣
↪Cook County": "SSCC"},inplace=True)
```

Removing Outliers

[7]:
```
##Removing outliers
df["log_value"] = np.log(df["value"])
sl=df['log_value'].quantile(.25)
sh=df['log_value'].quantile(.75)
iqr = sh-sl
sl_low = sl -1.5*iqr
sh_high = sh+1.5*iqr
df_iqr=df[(df['log_value']>sl_low)&(df['log_value']<sh_high)]
```

Dropping values less than or equal to zero in the sales value, taxes, medhinc, and av1 reduced the number of rows to 13,848.

[8]:
```
df=df.
↪dropna(subset=['white','black','asian','hispanic','poverty','squarefoot','beds','college','
df["av1"] = np.abs(df['av1'])
for x in df.index:
    if df.loc[x,"value"] <= 0:
        df.drop(x,inplace = True)
    elif df.loc[x,"taxes"] <= 0:
        df.drop(x,inplace = True)
    elif df.loc[x,"av1"] <= 0:
        df.drop(x,inplace = True)
    elif df.loc[x,"medhinc"] <= 0:
        df.drop(x,inplace = True)
```

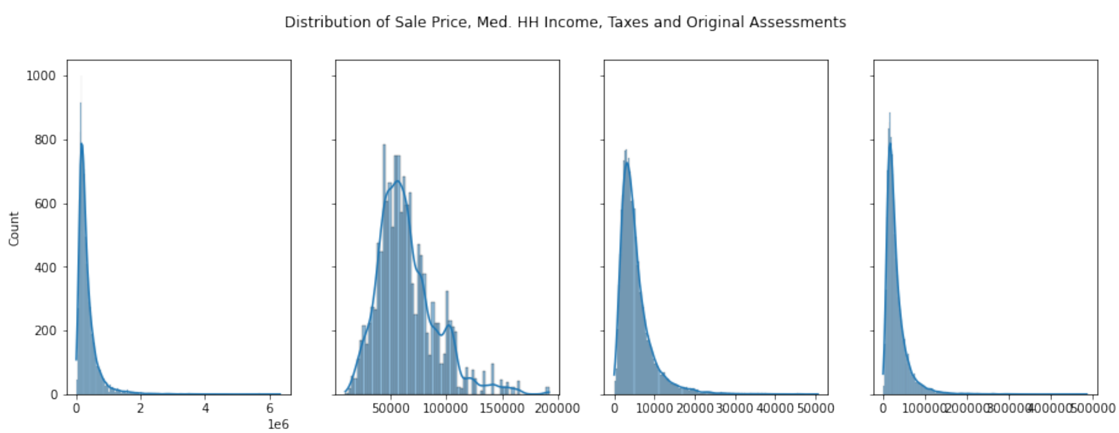
2.3 II.B Exploring Normal Distribution

We know that home sale values are skewed from the prior assignment. And even though sale value will probably be an independent variable in this analysis, we don't want it to have an out-sized effect on our models. Equally we need to check the distribuition of median household income and assessment values (which probably skew with sale values).

From the graph of UNLOGGED variables below, there is skew in all of them (although less with median household income).

[9]:
```python
pd.options.mode.chained_assignment = None  # default='warn'
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from pylab import *

def various_plots():
    fig, axes = plt.subplots(1, 4, figsize=(15,5), sharey=True)
    fig.suptitle('Distribution of Sale Price, Med. HH Income, Taxes and Original Assessments')
    Value = df1["value"]
    Medhinc = df1["medhinc"]
    Original_Assessment = df1["av1_corr"]
    Taxes = df1["taxes"]
    sns.histplot(ax=axes[0], x=Value.values,kde = True )
    sns.histplot(ax=axes[1], x=Medhinc.values, kde = True)
    sns.histplot(ax=axes[2], x=Taxes.values,kde = True )
    sns.histplot(ax=axes[3], x=Original_Assessment.values,kde = True )
various_plots()
```

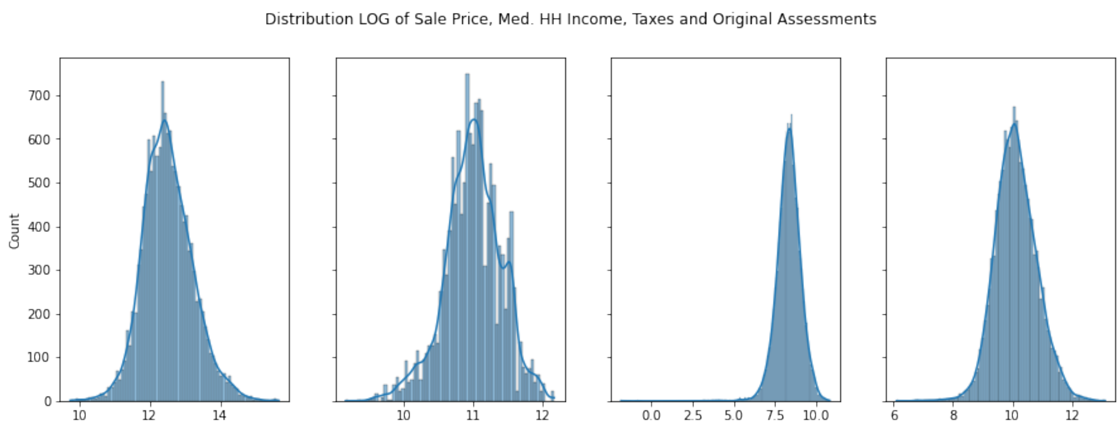


Plotting the log of these same values gives the graphs below...

[10]:
```python
pd.options.mode.chained_assignment = None  # default='warn'

df1["log_value"] = np.log(df1["value"])
df1["log_medhinc"] = np.log(df1["medhinc"])
df1["log_taxes"] = np.log(df1["taxes"])
df1["log_av1_corr"] = np.log(df1["av1_corr"])
```

[11]:
```
def various_log_plots():
    fig, axes = plt.subplots(1, 4, figsize=(15,5), sharey=True)
    fig.suptitle('Distribution LOG of Sale Price, Med. HH Income, Taxes and␣
 ↪Original Assessments')
    Value = df1["log_value"]
    Medhinc = df1["log_medhinc"]
    Original_Assessment = df1["log_av1_corr"]
    Taxes = df1["log_taxes"]
    sns.histplot(ax=axes[0], x=Value.values,kde = True )
    sns.histplot(ax=axes[1], x=Medhinc.values, kde = True)
    sns.histplot(ax=axes[2], x=Taxes.values,kde = True )
    sns.histplot(ax=axes[3], x=Original_Assessment.values,kde = True )
various_log_plots()
```



From these graphs of logged values, it's clear we should use the log of sale values and Original Assessment (last graph). There seems to be less benefit for MEDHINC and original assessments. The unlogged values skew left while the logged values skew right, but both are normally distributed. For consistency of interpretation we'll use the log in all cases.

PART I: Predicting the Probability of Appeals This part will focus on predicting the probability that someone will appeal their tax bill. We will create an "Appealed" variable equal to 1 if a given homeowner appealed, and a zero otherwise. We will test conventional OLS, vs. Probit and Logit models to see which yields the narrowest SER.

3 PART 1: Predicting the Probability of Appeal

3.1 A. Creating the Binary Dependent Variable

[12]:
```
df1['appealed'] = df1['appeal'].isnull()
```

[13]:
```
# Creating "appealed" variable
appealed_cnt = 0
```

```
for x in df1.index:
    if df1.loc[x,"appealed"] == False: #False means the value was not null,␣
 ↪which means there was an appeal, even if zero reduction was awarded
        df1.loc[x,'appealed'] = 1
    elif df1.loc[x,"appealed"] == True: #True means value is null, which means␣
 ↪they did not appeal
        df1.loc[x,'appealed'] = 0
        appealed_cnt = appealed_cnt + 1
df1['appealed_float'] = df1['appealed'].astype(float)
```

[14]:
```
print("Did not appleal count:",appealed_cnt)
```

```
Did not appleal count: 7419
```

Of the 13,848 rows in the cleaned data set, 7,440 did not appeal their tax assessment. (That is, roughly 54% did not appeal and 46 percent did). Note the code above created a variable called "applealed" in which a value of 1 means they did appeal, and 0 means they did not. This will be our binary Y value.

3.2 B. Initial Correlations

From the Pearson correlation table below, appeal values seem correlated with sale value (value), assessed value (av1), median household income, college, beds, and squarefootage. Note these last three may all be indicative of household income.

[15]:
```
df.corr(method="pearson")
```

[15]:

	pin14	av1	value	taxes	homeowner \
pin14	1.000000	-0.340826	-0.319606	-0.227181	-0.010361
av1	-0.340826	1.000000	0.872045	0.839806	0.040779
value	-0.319606	0.872045	1.000000	0.760297	0.024470
taxes	-0.227181	0.839806	0.760297	1.000000	0.024263
homeowner	-0.010361	0.040779	0.024470	0.024263	1.000000
white	-0.296579	0.389002	0.351030	0.364012	0.174969
black	0.442286	-0.323588	-0.289451	-0.274281	-0.143139
hispanic	-0.222243	-0.182425	-0.168206	-0.234290	-0.064163
asian	-0.399931	-0.076241	-0.074540	-0.139722	-0.037551
medhinc	-0.132826	0.463736	0.428648	0.489411	0.147278
poverty	0.051098	-0.200202	-0.169335	-0.269419	-0.181277
college	-0.320018	0.635910	0.602953	0.594503	0.096312
squarefoot	-0.112412	0.468267	0.452043	0.465315	-0.081917
beds	-0.116551	0.273514	0.268319	0.262360	-0.129768
walkscore	-0.549492	0.277960	0.287035	0.132448	-0.071486
elem_score	-0.449518	0.393775	0.366023	0.333724	0.116061
high_school_score	-0.304207	0.316681	0.282535	0.303663	0.119409
avg_school_score	-0.433014	0.410330	0.374397	0.371504	0.132449
appeal	-0.091582	0.441028	0.317343	0.193964	-0.117631
CD	0.459625	-0.124556	-0.135418	-0.019873	0.059444

```
SW                  0.034727 -0.192580 -0.178828 -0.160854   0.004387
VW                 -0.325360  0.026388  0.022209 -0.030477  -0.045542
WP                 -0.217682  0.432378  0.434212  0.319771  -0.022550
Chicago            -0.363631  0.224320  0.229348 -0.026039  -0.103956
NSCC               -0.417760  0.035786  0.025331  0.094327   0.041973
SSCC                0.616207 -0.246235 -0.244963 -0.030834   0.078783
Condominium              NaN       NaN       NaN       NaN        NaN
Non-condo                NaN       NaN       NaN       NaN        NaN
log_value          -0.470729  0.789946  0.837597  0.705299   0.058366

                       white     black  hispanic     asian   medhinc  …  \
pin14              -0.296579  0.442286 -0.222243 -0.399931 -0.132826  …
av1                 0.389002 -0.323588 -0.182425 -0.076241  0.463736  …
value               0.351030 -0.289451 -0.168206 -0.074540  0.428648  …
taxes               0.364012 -0.274281 -0.234290 -0.139722  0.489411  …
homeowner           0.174969 -0.143139 -0.064163 -0.037551  0.147278  …
white               1.000000 -0.906567 -0.024721 -0.007541  0.592712  …
black              -0.906567  1.000000 -0.326714 -0.414105 -0.414111  …
hispanic           -0.024721 -0.326714  1.000000  0.821688 -0.360684  …
asian              -0.007541 -0.414105  0.821688  1.000000 -0.294858  …
medhinc             0.592712 -0.414111 -0.360684 -0.294858  1.000000  …
poverty            -0.635388  0.501669  0.215275  0.181276 -0.663545  …
college             0.501367 -0.349043 -0.437490 -0.264472  0.727837  …
squarefoot          0.070897 -0.044519 -0.107369 -0.049077  0.124106  …
beds               -0.028060  0.022840 -0.032195  0.002258 -0.019465  …
walkscore          -0.002242 -0.147712  0.261441  0.339795 -0.081076  …
elem_score          0.615486 -0.572921 -0.159627  0.036178  0.502488  …
high_school_score   0.528989 -0.404472 -0.283988 -0.176984  0.436602  …
avg_school_score    0.641345 -0.540301 -0.270451 -0.096557  0.536871  …
appeal              0.085943 -0.065063 -0.065765 -0.036947  0.113230  …
CD                  0.069698  0.066639 -0.243916 -0.298759  0.134607  …
SW                 -0.073385  0.102226 -0.047686 -0.076724 -0.067258  …
VW                 -0.114261 -0.047880  0.305424  0.348939 -0.169917  …
WP                  0.187704 -0.177012 -0.049159  0.006976  0.170386  …
Chicago            -0.130794  0.014543  0.224530  0.241112 -0.156498  …
NSCC                0.118515 -0.165648 -0.094975  0.138352  0.088458  …
SSCC                0.059478  0.085377 -0.167553 -0.324934  0.103355  …
Condominium              NaN       NaN       NaN       NaN       NaN  …
Non-condo                NaN       NaN       NaN       NaN       NaN  …
log_value           0.497810 -0.452905 -0.153637 -0.005454  0.513272  …

                          CD        SW        VW        WP   Chicago      NSCC  \
pin14               0.459625  0.034727 -0.325360 -0.217682 -0.363631 -0.417760
av1                -0.124556 -0.192580  0.026388  0.432378  0.224320  0.035786
value              -0.135418 -0.178828  0.022209  0.434212  0.229348  0.025331
taxes              -0.019873 -0.160854 -0.030477  0.319771 -0.026039  0.094327
homeowner           0.059444  0.004387 -0.045542 -0.022550 -0.103956  0.041973
```

white	0.069698	-0.073385	-0.114261	0.187704	-0.130794	0.118515
black	0.066639	0.102226	-0.047880	-0.177012	0.014543	-0.165648
hispanic	-0.243916	-0.047686	0.305424	-0.049159	0.224530	-0.094975
asian	-0.298759	-0.076724	0.348939	0.006976	0.241112	0.138352
medhinc	0.134607	-0.067258	-0.169917	0.170386	-0.156498	0.088458
poverty	-0.228974	-0.013853	0.233235	-0.008652	0.350606	-0.158022
college	-0.093882	-0.198777	-0.040482	0.501707	0.134162	0.106344
squarefoot	-0.048637	-0.151158	0.052470	0.217064	0.139171	-0.025094
beds	-0.113270	-0.086357	0.079765	0.171622	0.156601	-0.011454
walkscore	-0.821380	-0.048665	0.525310	0.457092	0.571627	-0.003738
elem_score	-0.093559	-0.005928	-0.025765	0.187273	0.075863	0.234469
high_school_score	0.002939	0.014811	-0.061583	0.070190	-0.102346	0.256542
avg_school_score	-0.050458	0.005139	-0.050706	0.146322	-0.018856	0.292668
appeal	-0.071544	-0.101643	0.030262	0.169587	0.136276	-0.032128
CD	1.000000	-0.388358	-0.435016	-0.203181	-0.439539	-0.038011
SW	-0.388358	1.000000	-0.458836	-0.214307	-0.119851	0.106961
VW	-0.435016	-0.458836	1.000000	-0.240054	0.314690	-0.020960
WP	-0.203181	-0.214307	-0.240054	1.000000	0.333243	-0.072341
Chicago	-0.439539	-0.119851	0.314690	0.333243	1.000000	-0.303628
NSCC	-0.038011	0.106961	-0.020960	-0.072341	-0.303628	1.000000
SSCC	0.463108	0.055491	-0.302499	-0.290079	-0.818257	-0.299268
Condominium	NaN	NaN	NaN	NaN	NaN	NaN
Non-condo	NaN	NaN	NaN	NaN	NaN	NaN
log_value	-0.157043	-0.206051	0.077452	0.420231	0.249094	0.090613

	SSCC	Condominium	Non-condo	log_value
pin14	0.616207	NaN	NaN	-0.470729
av1	-0.246235	NaN	NaN	0.789946
value	-0.244963	NaN	NaN	0.837597
taxes	-0.030834	NaN	NaN	0.705299
homeowner	0.078783	NaN	NaN	0.058366
white	0.059478	NaN	NaN	0.497810
black	0.085377	NaN	NaN	-0.452905
hispanic	-0.167553	NaN	NaN	-0.153637
asian	-0.324934	NaN	NaN	-0.005454
medhinc	0.103355	NaN	NaN	0.513272
poverty	-0.255773	NaN	NaN	-0.277205
college	-0.198517	NaN	NaN	0.698288
squarefoot	-0.124232	NaN	NaN	0.485549
beds	-0.149916	NaN	NaN	0.303459
walkscore	-0.570199	NaN	NaN	0.309908
elem_score	-0.217436	NaN	NaN	0.496196
high_school_score	-0.052287	NaN	NaN	0.391637
avg_school_score	-0.157694	NaN	NaN	0.507973
appeal	-0.118427	NaN	NaN	0.234558
CD	0.463108	NaN	NaN	-0.157043
SW	0.055491	NaN	NaN	-0.206051

```
VW                 -0.302499          NaN         NaN   0.077452
WP                 -0.290079          NaN         NaN   0.420231
Chicago            -0.818257          NaN         NaN   0.249094
NSCC               -0.299268          NaN         NaN   0.090613
SSCC                1.000000          NaN         NaN  -0.304124
Condominium              NaN          NaN         NaN        NaN
Non-condo                NaN          NaN         NaN        NaN
log_value          -0.304124          NaN         NaN   1.000000

[29 rows x 29 columns]
```

3.3 C. Creating a binary model with 'appealed' as the binary outcome variable

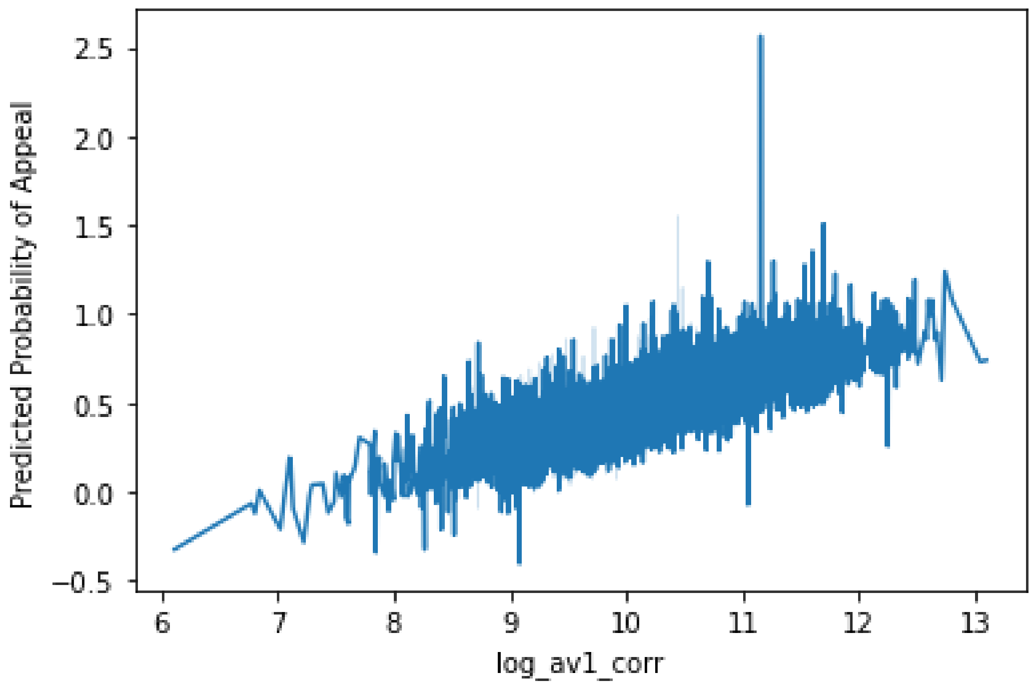
3.3.1 Using LPM Model

Having constructed the binary outcome variable "appealed", we want to see if LPM is a sufficient model for this assignment. That will be determined by whether it produces values outside of the range of zero to one. We'll include a large number of variables at first as an unrestricted model. Later we'll test some of the variables for their significance.

[16]:
```
y = df1['appealed_float']
```

[17]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import seaborn as sns
x = ␣
 ↪df1[['log_av1_corr','log_value','log_taxes','squarefoot','log_medhinc','homeowner','beds','
x = sm.add_constant(x)
LPMmodel = sm.OLS(y,x)
LPMresult = LPMmodel.fit()
predY = LPMresult.predict(x)
ax = sns.lineplot(x=df1["log_av1_corr"], y=predY)
ax.set(xlabel="log_av1_corr", ylabel="Predicted Probability of Appeal")
```

[17]:
```
[Text(0.5, 0, 'log_av1_corr'), Text(0, 0.5, 'Predicted Probability of Appeal')]
```



[18]:
```
LPMresult.summary()
```

[18]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:          appealed_float   R-squared:                       0.130
Model:                             OLS   Adj. R-squared:                  0.130
Method:                  Least Squares   F-statistic:                     148.0
Date:                 Tue, 20 Apr 2021   Prob (F-statistic):               0.00
Time:                         09:00:31   Log-Likelihood:                -9031.8
No. Observations:                13827   AIC:                         1.809e+04
Df Residuals:                    13812   BIC:                         1.821e+04
Df Model:                           14
Covariance Type:             nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            1.9561      0.481      4.069      0.000       1.014       2.899
log_av1_corr     0.2825      0.014     20.065      0.000       0.255       0.310
log_value       -0.2520      0.013    -19.956      0.000      -0.277      -0.227
log_taxes       -0.0069      0.009     -0.765      0.444      -0.024       0.011
squarefoot    8.256e-05   5.86e-06     14.087      0.000    7.11e-05    9.41e-05
log_medhinc     -0.0123      0.022     -0.559      0.576      -0.055       0.031
homeowner        0.0436      0.008      5.306      0.000       0.027       0.060
```

```
beds              -0.0152      0.004     -4.183      0.000      -0.022      -0.008
walkscore          0.0007      0.000      3.058      0.002       0.000       0.001
poverty            0.0615      0.069      0.895      0.371      -0.073       0.196
college            0.4158      0.053      7.824      0.000       0.312       0.520
white             -1.2722      0.433     -2.941      0.003      -2.120      -0.424
black             -1.3436      0.434     -3.097      0.002      -2.194      -0.493
hispanic          -0.2020      0.036     -5.588      0.000      -0.273      -0.131
asian             -1.1823      0.452     -2.616      0.009      -2.068      -0.296
==============================================================================
Omnibus:                   108369.040   Durbin-Watson:                   0.409
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1243.462
Skew:                           0.168   Prob(JB):                    9.68e-271
Kurtosis:                       1.570   Cond. No.                     4.91e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 4.91e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

As expected, the LPM model above produces prediction above 1 and below zero. We'll turn to Probit.

3.3.2 Using Probit Model

We'll use the Probit function below and see how well it fits the data.

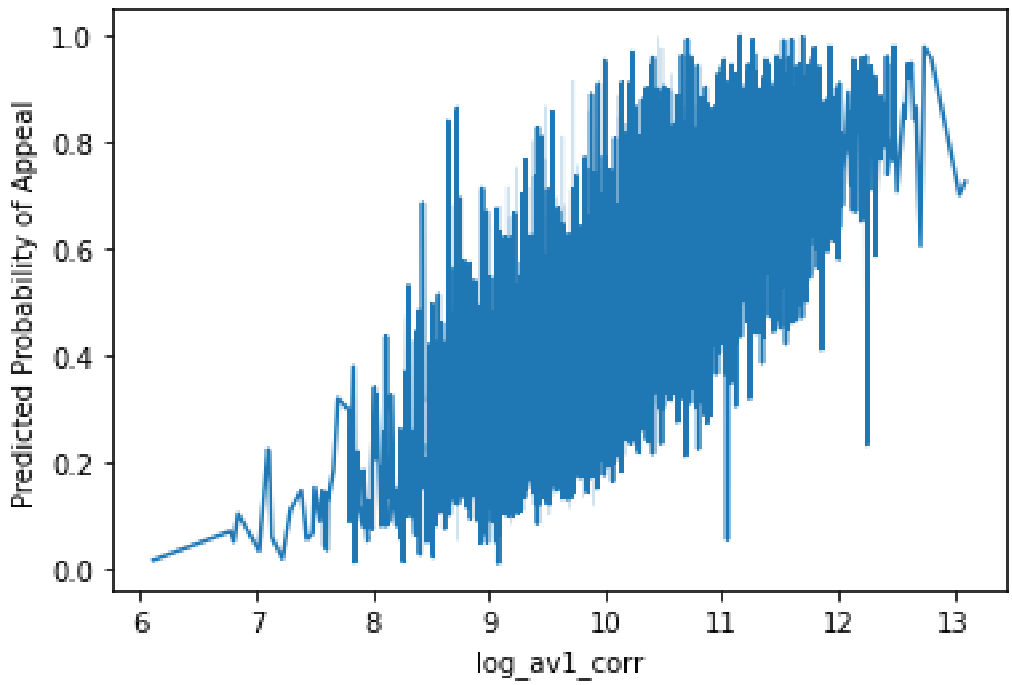
[19]:
```
PRx =␣
 ↪df1[['log_av1_corr','log_value','log_taxes','squarefoot','log_medhinc','homeowner','beds','
PRx = sm.add_constant(PRx)
PModel = sm.Probit(y,PRx)
PResult = PModel.fit()
PRpredY = PResult.predict(PRx)
```

```
Optimization terminated successfully.
         Current function value: 0.621033
         Iterations 5
```

Below is the same model as above using the Probit function. The graph uses log of av1 as x.

[20]:
```
#Probit plotted with Log AV1 as X
ax = sns.lineplot(x=df1["log_av1_corr"], y=PRpredY)
ax.set(xlabel="log_av1_corr", ylabel="Predicted Probability of Appeal")
```

[20]:
```
[Text(0.5, 0, 'log_av1_corr'), Text(0, 0.5, 'Predicted Probability of Appeal')]
```



[21]:
```
PResult.summary()
```

[21]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                          Probit Regression Results
==============================================================================
Dep. Variable:          appealed_float   No. Observations:                13827
Model:                          Probit   Df Residuals:                    13812
Method:                            MLE   Df Model:                           14
Date:                 Tue, 20 Apr 2021   Pseudo R-squ.:                  0.1006
Time:                         09:01:42   Log-Likelihood:                -8587.0
converged:                        True   LL-Null:                       -9547.2
Covariance Type:             nonrobust   LLR p-value:                     0.000
================================================================================
                   coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
const            4.6149      1.359      3.396      0.001       1.952       7.278
log_av1_corr     0.7549      0.039     19.212      0.000       0.678       0.832
log_value       -0.7024      0.036    -19.595      0.000      -0.773      -0.632
log_taxes       -0.0231      0.025     -0.913      0.361      -0.073       0.027
squarefoot       0.0003   1.85e-05     14.438      0.000       0.000       0.000
log_medhinc     -0.0422      0.062     -0.677      0.498      -0.164       0.080
homeowner        0.1198      0.023      5.195      0.000       0.075       0.165
beds            -0.0546      0.011     -5.052      0.000      -0.076      -0.033
walkscore        0.0023      0.001      3.413      0.001       0.001       0.004
```

```
poverty           0.0987      0.196      0.504      0.614     -0.285      0.482
college           1.1533      0.151      7.649      0.000      0.858      1.449
white            -3.6715      1.214     -3.024      0.002     -6.051     -1.292
black            -3.8798      1.218     -3.185      0.001     -6.267     -1.492
hispanic         -0.5652      0.102     -5.552      0.000     -0.765     -0.366
asian            -3.4539      1.268     -2.723      0.006     -5.940     -0.968
================================================================================
"""
```

3.3.3 Using Logit

Below, we'll run the same inclusive model using the Logit function. We also show two graphs, with log_av1 and Medhinc as x.

[29]:
```
#Logit Model and plot with log_av1 as x

LGx =␣
↪df1[['log_av1_corr','log_value','log_taxes','squarefoot','log_medhinc','homeowner','beds','
LGx = sm.add_constant(PRx)
LModel = sm.Logit(y,LGx)
LResult = LModel.fit()
LGpredY = LResult.predict(LGx)
LResult.summary()
```

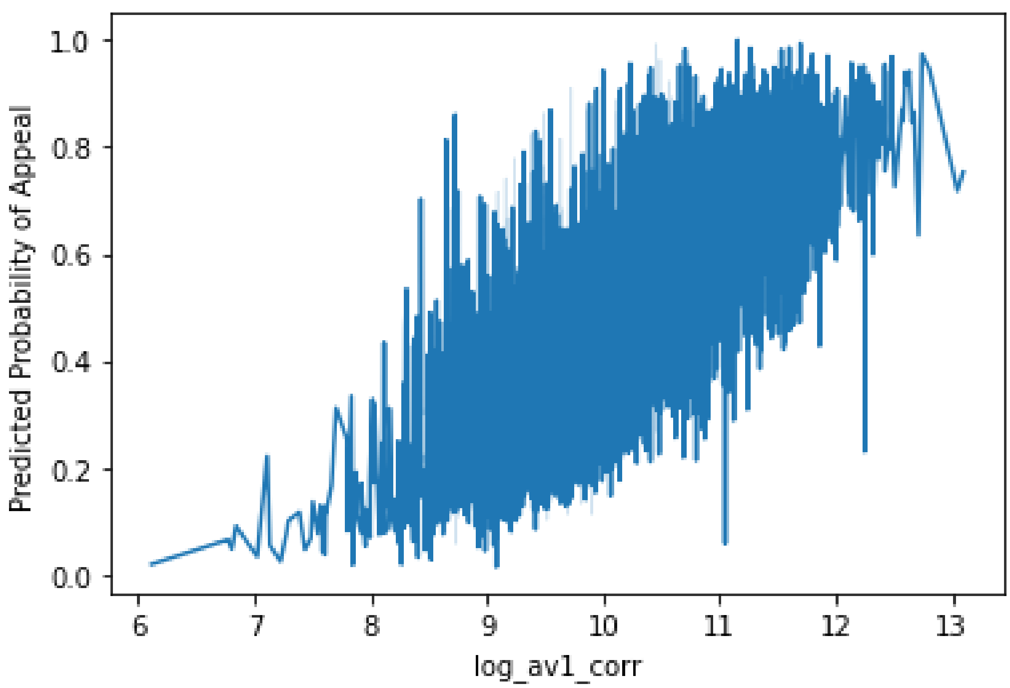
```
Optimization terminated successfully.
         Current function value: 0.619850
         Iterations 5
```

[29]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:          appealed_float   No. Observations:                13827
Model:                           Logit   Df Residuals:                    13812
Method:                            MLE   Df Model:                           14
Date:                 Tue, 20 Apr 2021   Pseudo R-squ.:                  0.1023
Time:                         09:03:20   Log-Likelihood:                -8570.7
converged:                        True   LL-Null:                       -9547.2
Covariance Type:             nonrobust   LLR p-value:                     0.000
================================================================================
                   coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
const            7.5611      2.243      3.371      0.001      3.165     11.957
log_av1_corr     1.3778      0.073     18.954      0.000      1.235      1.520
log_value       -1.2323      0.062    -19.735      0.000     -1.355     -1.110
log_taxes       -0.0640      0.043     -1.502      0.133     -0.148      0.019
squarefoot       0.0004   3.21e-05     13.779      0.000      0.000      0.001
```

```
log_medhinc    -0.0705     0.103    -0.687     0.492     -0.272      0.131
homeowner       0.1927     0.038     5.088     0.000      0.118      0.267
beds           -0.0908     0.018    -4.942     0.000     -0.127     -0.055
walkscore       0.0036     0.001     3.231     0.001      0.001      0.006
poverty         0.1463     0.323     0.453     0.650     -0.486      0.779
college         1.8098     0.249     7.257     0.000      1.321      2.299
white          -6.1648     1.996    -3.088     0.002    -10.078     -2.252
black          -6.4883     2.004    -3.238     0.001    -10.416     -2.561
hispanic       -0.9126     0.167    -5.452     0.000     -1.241     -0.585
asian          -5.8362     2.084    -2.801     0.005     -9.921     -1.752
================================================================================
"""
```

[30]:
```python
ax = sns.lineplot(x=df1["log_av1_corr"], y=LGpredY)
ax.set(xlabel="log_av1_corr", ylabel="Predicted Probability of Appeal")
```

[30]: [Text(0.5, 0, 'log_av1_corr'), Text(0, 0.5, 'Predicted Probability of Appeal')]



3.4 D. Calculating Each Model's Accuracy

[23]:
```python
LPM_predicted_probabilities = LPMresult.predict()
LPM_predicted_outcomes = []
for predicted_probability in LPM_predicted_probabilities:
    if predicted_probability<.5:
```

```
        LPM_predicted_outcomes.append(0)
    else:
        LPM_predicted_outcomes.append(1)

probit_predicted_probabilities = PResult.predict()
probit_predicted_outcomes = []
for predicted_probability in probit_predicted_probabilities:
    if predicted_probability<.5:
        probit_predicted_outcomes.append(0)
    else:
        probit_predicted_outcomes.append(1)

logit_predicted_probabilities = LResult.predict()
logit_predicted_outcomes = []
for predicted_probability in logit_predicted_probabilities:
    if predicted_probability<.5:
        logit_predicted_outcomes.append(0)
    else:
        logit_predicted_outcomes.append(1)

LPM_hits = 0
probit_hits = 0
logit_hits = 0
i = 0
for fraud in y:
    if LPM_predicted_outcomes[i]==fraud:
        LPM_hits = LPM_hits+1
    if probit_predicted_outcomes[i]==fraud:
        probit_hits = probit_hits + 1
    if logit_predicted_outcomes[i]==fraud:
        logit_hits = logit_hits + 1
    i = i+1

LPM_percent_correct = LPM_hits/len(y)*100
probit_percent_correct = probit_hits/len(y)*100
logit_percent_correct = logit_hits/len(y)*100

print("The LPM model got %f correct" % LPM_percent_correct)
print("The probit model got %f correct" % probit_percent_correct)
print("The logit model got %f correct" % logit_percent_correct)
```

```
The LPM model got 66.247198 correct
The probit model got 66.196572 correct
The logit model got 66.362913 correct
```

From the three lines above, we see that the logit model is most accurate, an Probit is the least accurate, but none by a signifcant amount. We will focus on the Probit model

and compute the Average Partial Effects (APE) for each of the factors to determine their contribution to the model.

3.5 E. Average Partial Effects of Each Variable

The following code calculates the APE for each variable in the Probit model.

[24]:
```
from scipy.stats import norm
#First, we calculate the dot product of the values for X and the ceofficients␣
↪in our model
linear_models = PRx.dot(PResult.params)
#Then, we calculate their pdfs
pdfs = norm.pdf(linear_models)
partial_effects = pdfs*PResult.params[1] #log_av1_corr
APE = partial_effects.mean()
print('AVERAGE PARTIAL EFFECT OF EACH VARIABLE:')
print("The APE for log of Av1 is %f" % APE)
partial_effects = pdfs*PResult.params[2] #Log of sale value
APE = partial_effects.mean()
print("The APE for log of sale value is %f" % APE)
partial_effects = pdfs*PResult.params[3]
APE = partial_effects.mean()
print("The APE for log of Taxes is %f" % APE)
partial_effects = pdfs*PResult.params[4]
APE = partial_effects.mean()
print("The APE for Squarefootage is %f" % APE)
partial_effects = pdfs*PResult.params[5]
APE = partial_effects.mean()
print("The APE for log of MEDHINC is %f" % APE)
partial_effects = pdfs*PResult.params[6] #Log of sale value
APE = partial_effects.mean()
print("The APE for Homeoner is %f" % APE)
partial_effects = pdfs*PResult.params[7]
APE = partial_effects.mean()
print("The APE for Beds is %f" % APE)
partial_effects = pdfs*PResult.params[8]
APE = partial_effects.mean()
print("The APE for Walkscore is %f" % APE)
partial_effects = pdfs*PResult.params[9]
APE = partial_effects.mean()
print("The APE for Poverty is %f" % APE)
partial_effects = pdfs*PResult.params[10]
APE = partial_effects.mean()
print("The APE for College is %f" % APE)
partial_effects = pdfs*PResult.params[11]
APE = partial_effects.mean()
print("The APE for White is %f" % APE)
partial_effects = pdfs*PResult.params[12]
```

```python
APE = partial_effects.mean()
print("The APE for Black is %f" % APE)
partial_effects = pdfs*PResult.params[13]
APE = partial_effects.mean()
print("The APE for Hispanic is %f" % APE)
partial_effects = pdfs*PResult.params[14]
APE = partial_effects.mean()
print("The APE for Asian is %f" % APE)
```

```
AVERAGE PARTIAL EFFECT OF EACH VARIABLE:
The APE for log of Av1 is 0.268395
The APE for log of sale value is -0.249747
The APE for log of Taxes is -0.008229
The APE for Squarefootage is 0.000095
The APE for log of MEDHINC is -0.014989
The APE for Homeoner is 0.042601
The APE for Beds is -0.019425
The APE for Walkscore is 0.000826
The APE for Poverty is 0.035085
The APE for College is 0.410045
The APE for White is -1.305404
The APE for Black is -1.379469
The APE for Hispanic is -0.200946
The APE for Asian is -1.228041
```

While the contribution of some of these log values seem small, their contribution is 100 times the value shown because of the log-linear nature of the model.

3.6 F. Conclusion for Part I

The Probit model we've constructed predicts the probability of a homeowner contesting their assesment. D above, we see it is accurate 66.2% of the time. The model indicates the following: - As discussed in the course, the simple LPM model is homoskadastic and produces smaller SERs for the regressors, but has the down-side of generating probabilities below 0 or greater than 1. - Logit and Probit avoid this but introduce heterskedasticity. We used Probit for ease of interpretation, complimented with APE analysis. - Overall accuracy of 66% for each of these models is quite good, given the inherent noice in this type of dataset. - Interestingly, college education (or simply education) seems to have a significant part to play in whether someone goes through the appeals process. It is a larger influencer than the assessment magnitude alone. This may be because those of higher education are more equiped to lodge these appeals. - Racial mix, in and of itself, does not seem to be a contributing factor.

3.7 PART 2 - Predictors of Appeal Reduction and Fairness

This part of the study focuses on what influences the monetary reduction they gain from appeal. Our goal is to understand whether there is bias or unfairness in the tax system.

Methodology: After cleaning the data, we first develop a model for predicting the likely amount of an appeal reduction. We'll select features that would support our hypotheses, and build multivariate ols models to prove them. We also include an analysis at the end of this section as to whether losing on appeal is has any bias.

Three Hypotheses

1. AV/MV: We naturally assumes that assesed price to market value ratio, would influence appeal. As this ratio increases, the influence of appeal would also increase. Since this ratio indicates the gap between assessed price and actual value. If it is high, then it might means that the value is over assessed.

2. Ethnicity, medhinc, college: Doerner and Ihlandfeldt (2012)[https://www.researchgate.net/publication/268294221_An_Empirical_Critique_of_the_Property_ examine the effect of appeals on assessment ratios in Miami-Dade County and find that they disproportionately benefit white, rich neighborhoods. We assume this still holds true for cook county as well, but will test it.

3. Region: There are three main regions in this dataset. We assume appeal in city would gain more, since there are many attorneys and retailers, while the population being more densed. We would explore these with different models.

[25]:
```python
df['appealed'] = 1
df.loc[df['appeal'].isna(),"appealed"] = 0
## Since this is to see how successfully appeal works, we only need data that appealed succesfully.
df_app = df[df['appealed']==1]
df_app = df[df['appeal']>0]
```

[26]:
```python
## Since this is to see how successfully appeal works, we only need data that appealed succesfully.
df_app = df[df['appealed']==1]
df_app = df[df['appeal']>0]
```

[27]:
```python
g=sns.PairGrid(df_app[['appeal','av1','value','white','medhinc','college','Chicago']])
g.map(sns.scatterplot)
```

[27]:
```
<seaborn.axisgrid.PairGrid at 0x199829b58e0>
```


```
[31]: import numpy as np
      import statsmodels.formula.api as sm
      model1 = sm.ols(formula="appeal ~ av1+ value+ av1/value", data=df_app)
      result1 = model1.fit()
      result1.summary()
```

```
[31]: <class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                 appeal   R-squared:                       0.274
Model:                            OLS   Adj. R-squared:                  0.273
Method:                 Least Squares   F-statistic:                     470.4
```

```
Date:                   Tue, 20 Apr 2021   Prob (F-statistic):          2.42e-259
Time:                           09:17:40   Log-Likelihood:                -41726.
No. Observations:                   3748   AIC:                         8.346e+04
Df Residuals:                       3744   BIC:                         8.348e+04
Df Model:                              3
Covariance Type:               nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   1113.9137    527.728      2.111      0.035      79.251    2148.576
av1            0.2978      0.016     18.556      0.000       0.266       0.329
value         -0.0146      0.001     -9.705      0.000      -0.017      -0.012
av1:value   5.017e-08   5.52e-09      9.091      0.000    3.93e-08     6.1e-08
==============================================================================
Omnibus:                    4337.925   Durbin-Watson:                   2.058
Prob(Omnibus):                 0.000   Jarque-Bera (JB):           702864.597
Skew:                          5.837   Prob(JB):                         0.00
Kurtosis:                     69.064   Cond. No.                     2.04e+11
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 2.04e+11. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

The first model is a rather simple multivariate OLS. It is working well as we expected, coefficient of primary assessed price and AV/MV is positive, and sale price being slightly negative. And the p-values for these coefficients are low enough to argue that it's significant.

[32]:
```
model2 = sm.ols(formula="appeal ~ av1+ value+ av1/value+white+medhinc",␣
 ↪data=df_app)
result2 = model2.fit()
result2.summary()
```

[32]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                 appeal   R-squared:                       0.279
Model:                            OLS   Adj. R-squared:                  0.278
Method:                 Least Squares   F-statistic:                     290.2
Date:                Tue, 20 Apr 2021   Prob (F-statistic):          4.72e-263
Time:                        09:17:54   Log-Likelihood:                -41711.
No. Observations:                3748   AIC:                         8.343e+04
Df Residuals:                    3742   BIC:                         8.347e+04
```

```
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   4559.7757    824.463      5.531      0.000    2943.334    6176.217
av1            0.3176      0.016     19.358      0.000       0.285       0.350
value         -0.0126      0.002     -8.176      0.000      -0.016      -0.010
av1:value   4.044e-08   5.78e-09      6.992      0.000    2.91e-08    5.18e-08
white      -3533.5210   1282.601     -2.755      0.006   -6048.186   -1018.856
medhinc       -0.0346      0.012     -2.872      0.004      -0.058      -0.011
==============================================================================
Omnibus:                     4294.758   Durbin-Watson:                   2.068
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           666031.843
Skew:                           5.748   Prob(JB):                         0.00
Kurtosis:                      67.286   Cond. No.                     5.24e+11
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 5.24e+11. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

The result does not seem match our expectation, our hypothesis 2. Actually, being white with higher income has negative impact on appeal.

[33]:
```
model3 = sm.ols(formula="appeal ~ av1+ value+ av1/value+white+medhinc␣
 ↪+college", data=df_app)
result3 = model3.fit()
result3.summary()
```

[33]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                 appeal   R-squared:                       0.282
Model:                            OLS   Adj. R-squared:                  0.281
Method:                 Least Squares   F-statistic:                     244.7
Date:                Tue, 20 Apr 2021   Prob (F-statistic):         1.43e-264
Time:                        09:18:01   Log-Likelihood:                -41705.
No. Observations:                3748   AIC:                         8.342e+04
Df Residuals:                    3741   BIC:                         8.347e+04
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
```

```
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   4887.0083    828.221      5.901      0.000    3263.200    6510.816
av1            0.3018      0.017     17.796      0.000       0.269       0.335
value         -0.0144      0.002     -8.894      0.000      -0.018      -0.011
av1:value   4.851e-08    6.2e-09      7.826      0.000    3.64e-08    6.07e-08
white      -3925.3186   1285.245     -3.054      0.002   -6445.167   -1405.470
medhinc       -0.0562      0.013     -4.176      0.000      -0.083      -0.030
college     8774.0065   2450.166      3.581      0.000    3970.215    1.36e+04
==============================================================================
Omnibus:                     4312.979   Durbin-Watson:                   2.070
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           685440.456
Skew:                           5.783   Prob(JB):                         0.00
Kurtosis:                      68.233   Cond. No.                     9.53e+11
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 9.53e+11. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

On the other hand, graduating college, is positively correlated with appeal. The p-value indicates that it is significant.

[34]:
```
model5 = sm.ols(formula="appeal ~ av1+ value+ av1/value+white+medhinc␣
 ↪+college+college*medhinc", data=df_app)
result5 = model5.fit()
result5.summary()
```

[34]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                 appeal   R-squared:                       0.282
Model:                            OLS   Adj. R-squared:                  0.281
Method:                 Least Squares   F-statistic:                     209.7
Date:                Tue, 20 Apr 2021   Prob (F-statistic):         2.30e-263
Time:                        09:18:31   Log-Likelihood:                -41705.
No. Observations:                3748   AIC:                         8.343e+04
Df Residuals:                    3740   BIC:                         8.347e+04
Df Model:                           7
Covariance Type:            nonrobust
================================================================================
===
                      coef    std err          t      P>|t|      [0.025
```

```
0.975]
--------------------------------------------------------------------------------
---
Intercept        4684.3658   1568.405      2.987      0.003    1609.353
7759.379
av1                 0.3019      0.017     17.784      0.000       0.269
0.335
value              -0.0143      0.002     -8.849      0.000      -0.018
-0.011
av1:value        4.845e-08   6.21e-09      7.804      0.000    3.63e-08
6.06e-08
white           -4004.9156   1387.788     -2.886      0.004   -6725.811
-1284.020
medhinc            -0.0522      0.030     -1.754      0.080      -0.111
0.006
college          9428.3163   4949.511      1.905      0.057    -275.688
1.91e+04
college:medhinc    -0.0097      0.064     -0.152      0.879      -0.134
0.115
==============================================================================
Omnibus:                     4312.566   Durbin-Watson:                   2.070
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           685018.971
Skew:                           5.782   Prob(JB):                         0.00
Kurtosis:                      68.213   Cond. No.                     1.98e+12
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.98e+12. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

Thought graduating college would lead to higher median income, but the p value is too high to argue significance of this variable

[35]:
```
model6 = sm.ols(formula="appeal ~ av1+ value+ av1/value+Chicago", data=df_app)
result6 = model6.fit()
result6.summary()
```

[35]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                 appeal   R-squared:                       0.279
Model:                            OLS   Adj. R-squared:                  0.279
Method:                 Least Squares   F-statistic:                     362.8
```

```
Date:                Tue, 20 Apr 2021   Prob (F-statistic):          2.38e-264
Time:                        09:18:43   Log-Likelihood:                -41711.
No. Observations:                3748   AIC:                         8.343e+04
Df Residuals:                    3743   BIC:                         8.346e+04
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    385.9966    542.522      0.711      0.477    -677.671    1449.664
av1            0.2896      0.016     18.040      0.000       0.258       0.321
value         -0.0158      0.002    -10.468      0.000      -0.019      -0.013
av1:value   5.463e-08   5.56e-09      9.828      0.000    4.37e-08    6.55e-08
Chicago     3105.6811    571.458      5.435      0.000    1985.282    4226.080
==============================================================================
Omnibus:                     4346.417   Durbin-Watson:                   2.072
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           719402.185
Skew:                           5.849   Prob(JB):                         0.00
Kurtosis:                      69.856   Cond. No.                     2.42e+11
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 2.42e+11. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

As assumed, living inside the city has a positive effect on appeal amount. The p-value is also small enough to argue it's significance.

[36]:
```
model7 = sm.ols(formula="appeal ~ av1+ value+ av1/
↪value+Chicago+white+college+medhinc", data=df_app)
result7 = model7.fit()
result7.summary()
```

[36]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                 appeal   R-squared:                       0.284
Model:                            OLS   Adj. R-squared:                  0.282
Method:                 Least Squares   F-statistic:                     211.5
Date:                Tue, 20 Apr 2021   Prob (F-statistic):          2.88e-265
Time:                        09:18:47   Log-Likelihood:                -41700.
No. Observations:                3748   AIC:                         8.342e+04
Df Residuals:                    3740   BIC:                         8.347e+04
```

```
Df Model:                           7
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept   3694.6744    919.522      4.018      0.000    1891.862    5497.487
av1            0.2969      0.017     17.445      0.000       0.264       0.330
value         -0.0151      0.002     -9.244      0.000      -0.018      -0.012
av1:value   5.108e-08   6.25e-09      8.170      0.000    3.88e-08    6.33e-08
Chicago     1886.1724    634.747      2.972      0.003     641.688    3130.657
white      -3468.6607   1293.066     -2.683      0.007   -6003.845    -933.477
college     6487.8594   2565.672      2.529      0.011    1457.606    1.15e+04
medhinc       -0.0406      0.014     -2.814      0.005      -0.069      -0.012
==============================================================================
Omnibus:                     4321.851   Durbin-Watson:                   2.073
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           697225.921
Skew:                           5.798   Prob(JB):                         0.00
Kurtosis:                      68.804   Cond. No.                     1.01e+12
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.01e+12. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

[37]:
```python
##VIF to check multi-colinearity
from statsmodels.stats.outliers_influence import variance_inflation_factor
ind_variables = df[["value","av1","Chicago","medhinc", "college","white"]] #Create a dataframe with our independent variables
ind_variables.insert(0, 'const', 1) #Insert a column of ones at the zeroth column of the dataframe
VIF_values = [] #Create a list to hold the VIFs that are calculated
for i in range(len(ind_variables.columns)): #Loop through the dataframe and calculate the VIF for each column
    VIF = variance_inflation_factor(ind_variables.values, i) #Calculate the VIF for the matrix on the ith column
    VIF_values.append(VIF) #Add the VIF to our list of values
VIFs = pd.DataFrame({"VIF":VIF_values}, index=ind_variables.columns) #Turn our list of VIFs into a dataframe
VIFs
```

[37]:

	VIF
const	10.730861
value	4.267175
av1	4.594948

```
Chicago    1.257184
medhinc    2.776097
college    3.032218
white      1.613285
```

Normally used cut-off for VIF is 5. Therfore, we don't have to be concerned about multicolinearity in this case.

3.7.1 T test

Although the result of model 7 suggests that all the variables are significant with p values less than 0.05, we want to make sure if it is. So to check this we are going to take null hypothesis h0: beta0 = beta1= beta2 = 0.

[38]:
```
import pandas as pd
import statsmodels.formula.api as sm
unrestricted_model = result7 #result7 was the unrestricted model, using all the␣
 ↪variables
model8 = sm.ols(formula="appeal ~ white+college+medhinc", data=df_app)
result8 = model8.fit()
restricted_model = result8 #res8 is the restricted model, using only White,␣
 ↪College, and medhinc

#for a fitted model, statsmodels helpfully stores the residuals in .resid. We␣
 ↪just need to square them and add them up
SSR_ur = sum(unrestricted_model.resid**2)
SSR_r = sum(restricted_model.resid**2)

k = 7  #the number of variables in the unrestricted model
q = 3 #there are two restrictions--count the equal signs in our null hypothesis
n = len(df_app) #the total number of samples

f_stat = ((SSR_r-SSR_ur)/q)/(SSR_ur/(n-k-1))
print("The f-statistic is %f" % f_stat)
```

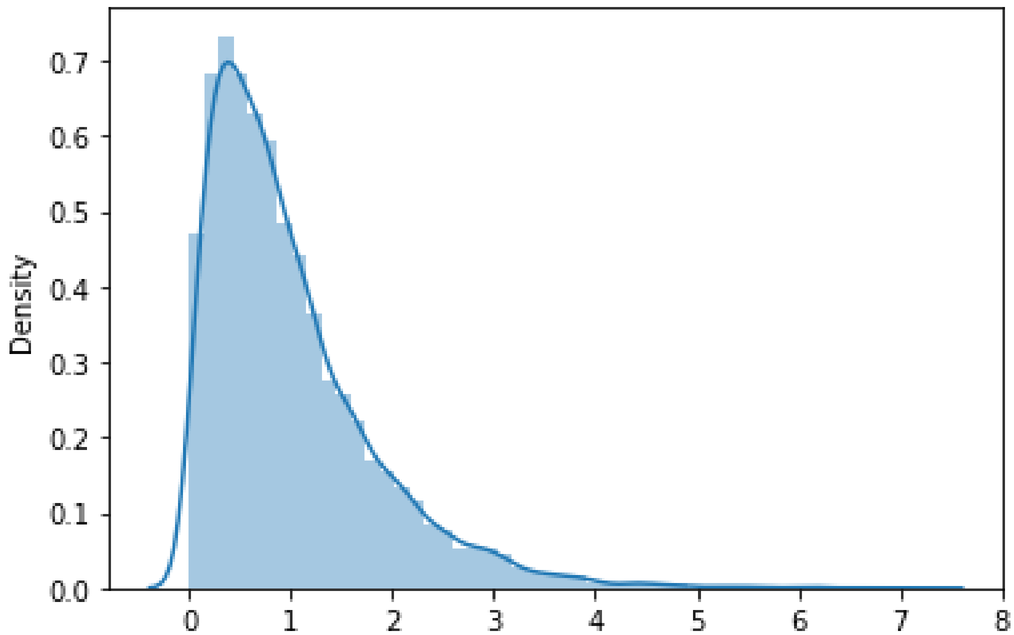
```
The f-statistic is 385.416570
```

[39]:
```
from scipy.stats import f
import seaborn as sns
r = f.rvs(q, n-k-1, size=10000) #plot some random values for this f-distribution
sns.distplot(r)
p_val = f.sf(f_stat,q,n-k-1)
print("The p-value of %f on this distribution is %f" % (f_stat, p_val))
```

```
C:\Users\bbenson\Anaconda3\lib\site-packages\seaborn\distributions.py:2551:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
```

```
histograms).
  warnings.warn(msg, FutureWarning)
```

```
The p-value of 385.416570 on this distribution is 0.000000
```



The result above shows that p-value is small enough, so we could not reject the joint hypothesis that the combination of white, college, medhinc affects appeal.

Adjusted R2

[40]:
```
print("The adjusted R Squared for the model that uses av1,value, and av1/value␣
 ↪is %f" % result1.rsquared_adj)
print("The adjusted R Squared for the model that uses av1,value,av1/value,␣
 ↪white, and medhinc is %f" % result2.rsquared_adj)
print("The adjusted R Squared for the model that uses av1,value,av1/
 ↪valueboardings, white, college, medhinc,college*medhinc is %f" % result3.
 ↪rsquared_adj)
print("The adjusted R Squared for the model that uses av1,value,av1/
 ↪valueboardings, the change in the DJ, and the change in the Nikkei is %f" %␣
 ↪result5.rsquared_adj)
print("The adjusted R Squared for the model that uses av1,value,av1/value, and␣
 ↪Chicago %f" % result6.rsquared_adj)
print("The adjusted R Squared for the model that uses av1,value,av1/
 ↪value,white, college, medhinc, chicago is %f" % result7.rsquared_adj)
```

```
The adjusted R Squared for the model that uses av1,value, and av1/value is
0.273148
```

```
The adjusted R Squared for the model that uses av1,value,av1/value, white, and
medhinc is 0.278462
The adjusted R Squared for the model that uses av1,value,av1/valueboardings,
white, college, medhinc,college*medhinc is 0.280734
The adjusted R Squared for the model that uses av1,value,av1/valueboardings, the
change in the DJ, and the change in the Nikkei is 0.280546
The adjusted R Squared for the model that uses av1,value,av1/value, and Chicago
0.278645
The adjusted R Squared for the model that uses av1,value,av1/value,white,
college, medhinc, chicago is 0.282237
```

The adjusted R squared generally increased as the number of variables increased. But the increasement is tiny. And the highest adjusted R squared is 0.282237 which is small. Meaning the dependent variables only explains less than 30% of dependent variables. There is no cut off with this numbers, but less than .30 would be considered low in general, therby meaning that our model has weak explanatory power.

3.7.2 Conclusion (Part 2)

Result and conclusion So we tried to explain monetary reward of appeal in this section. Our goal was to determine influencial factors of monetary reward of appealing. In this section, we simply took amount of appeal as a monetary reward. So we used OLS and different variables to explain the amount of appeal.

There were three assumptions, that we thought would influence monetary award(or appeal variable).

First, as AV/MV(assessed value to maket value ratio) increases, monetary award would increase. This idea seems natural, while being widely used in other studies. From model 1~7, we used, av1, value, and av1/value. It all acted as we expected, with assesed value having positive coefficients, market value negative, and AV/MV having positive coefficients. And through all models they were statistically significant with low p-values.

Second, among the research papers I read, An Empirical Critique of the Property Tax Appeals Proces, by Doner, argues that high income, majority white neighbors tend to get advantage on getting appeal. However, this assumption was wrong. In fact, with model 2,3,6,7 we were able to check that white and medhinc tend to have negative correlation with appeal, statistical significance. Meaning, they actually got disadvantages, when appealed.

There are various explanations on this surprising result on second hypothesis. First, there could've been a revision on assessment after the study of Doner, which was conducted in early 2010's. Second, maybe his study was wrong, most unlikely. At last, maybe our study was wrong. There are many deficits in our study with this part, which we'll explain later.

Third, region would matter. More specifically, whether you live inside the city or out of it matters. Thought it would matter in positive way, because there are various factors in the city that could influence on appeal amount. And it did matter and the coefficient was positive as expected, with low p value. We checked this fact through model 5~7.

Limit and deficits **Dependent Variable** Must admit frankly, that the model has many vulnerable points. First of all, the definition of monetary award of appealing, is rather too simple in this model. In fact, if you think just a little more deeper, you could see this does not make sense. Tax, value, tax rate, av1, and etc. There are so many things that need to be considered to design and define monetary reward in exact sense. It would be sophisticated, weights for each variables should be considered with other data, and during those process, trouble might occur with colinearity. Thus, we simply choose 'appeal' to represent monetary award, and this could've made the model less accurate.

Low R-squared Among the model I built, the highest adjusted R-Squared is 0.282. I was not expecting a high R squared(above 60%), but would have been happier if it was above 40%, or at least 30%. This model, the independent variables that I seleceted could only explan about 28% of the appeal price. Since it is social study, it could still be considered siginifcant, but still low.

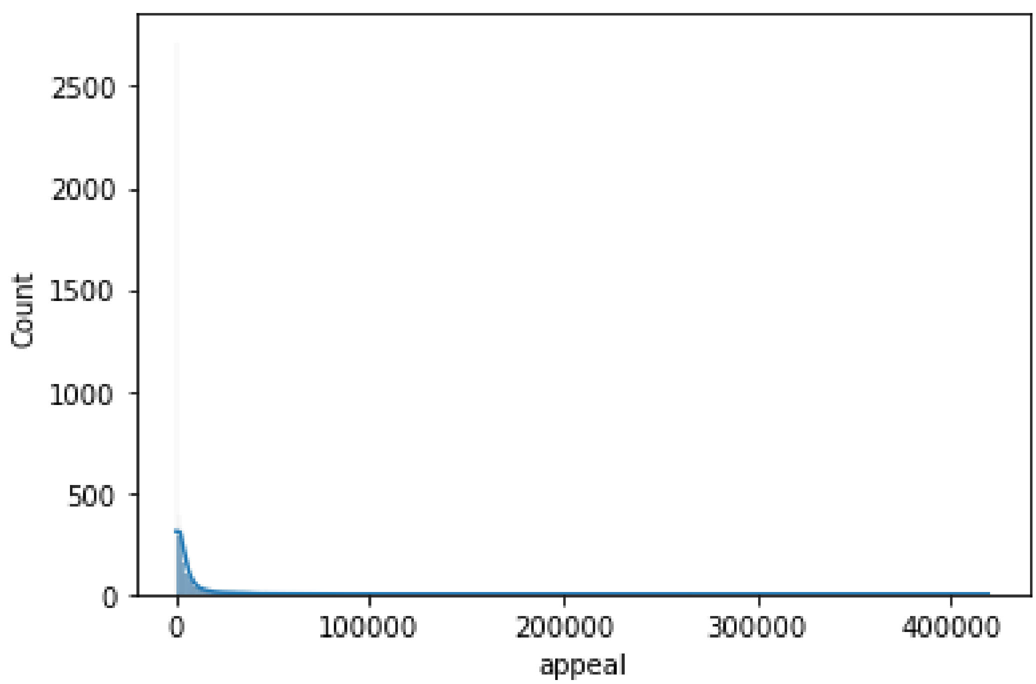
scaling,log, polynomials This is actually my 5th edition for the. There were numerous things that made me frustrated, defining dependent variables in various ways, selecting features, scaling, and etc. Scaling was the most annoying one though. It could be easily seen that the dependent variable is left-skewed. Wanted to normalize it. However, when i log scaled the appeal variable, the adjusted R squared tend to get lower with same independent variables.

3.8 Additional Analysis

This analysis is done by Bruce to help complement my analysis on monetary reduction effect on appeal.

[41]:
```
sns.histplot(x=df['appeal'],kde=True)
```

[41]: <AxesSubplot:xlabel='appeal', ylabel='Count'>



[42]:
```
#Logging some variables
pd.options.mode.chained_assignment = None  # default='warn'

df["log_value"] = np.log(df["value"])
df["log_medhinc"] = np.log(df["medhinc"])
df["log_taxes"] = np.log(df["taxes"])
df["log_av1"] = np.log(df["av1"])
```

[43]:
```
#Calculating % Reduction on Appeal
df['percent_reduction'] = df['appeal'] / df['av1']
# Deleting appeals that are greater than their original assessment (bad data)
for x in df.index:
    if df.loc[x,"percent_reduction"] >1:
        df.drop(x,inplace = True)
```

There is a 56.7% chance overall of winning on appeal. The counts are as follows:

[2]:
```
Win_Cnt = 0
for x in df.index:
    if df.loc[x,"percent_reduction"] == 0:
        df.loc[x,'won_lost'] = "Lost"
    elif df.loc[x,"percent_reduction"] != 0:
        df.loc[x,'won_lost'] = "Won"
        Win_Cnt = Win_Cnt + 1
#Create dummy variable from won_lost
dummy_won_lost = pd.get_dummies(df['won_lost'])
df = pd.concat([df,dummy_won_lost],axis=1)

print('Win Cnt =',Win_Cnt,'Lost Cnt =',len(df)-Win_Cnt,'Win Ratio␣
 ↪=',Win_Cnt*100/len(df))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-67ed93015de5> in <module>
      1 Win_Cnt = 0
----> 2 for x in df.index:
      3     if df.loc[x,"percent_reduction"] == 0:
      4         df.loc[x,'won_lost'] = "Lost"
      5     elif df.loc[x,"percent_reduction"] != 0:

NameError: name 'df' is not defined
```

3.9 Is there evidence of bias in the population of those who lost their appeals?

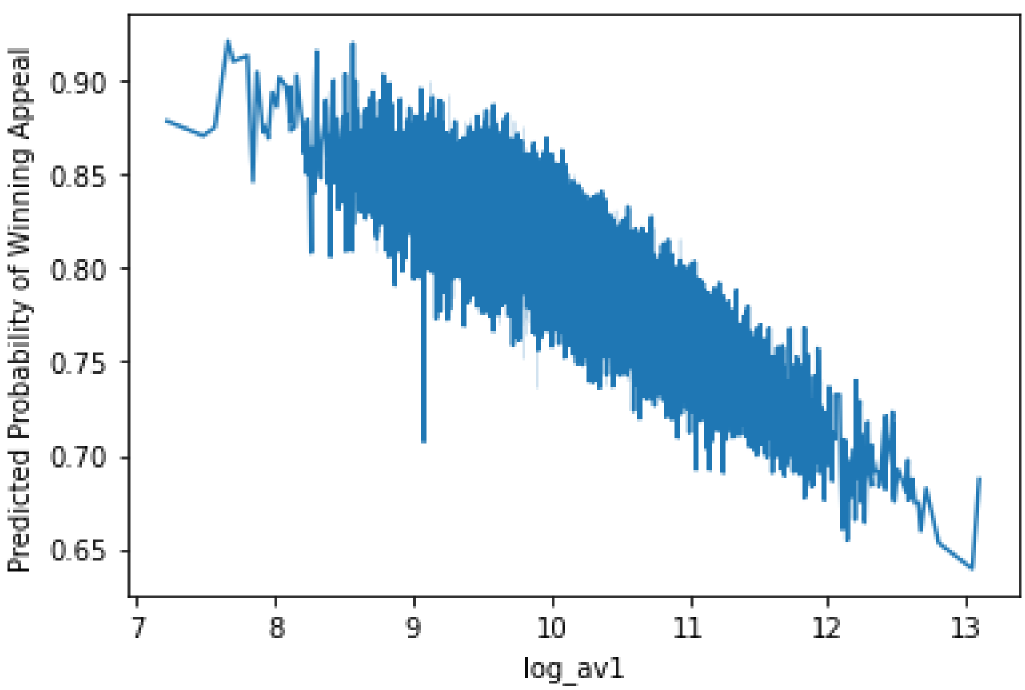
This question amounts to a binary outcome variable on win / lose around appeals. Our data already has a dumy variable that sets won to 1 and lost to zero.

[46]:
```
df['won_vs_lost'] = 0
for x in df.index:
    if df.loc[x,"won_lost"] == "Won":
        df["won_vs_lost"].loc[x] = 1
    elif df.loc[x,"won_lost"] == "Lost":
        df['won_vs_lost'].loc[x] = 0
```

[47]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import seaborn as sns
y = df['won_vs_lost']
X = df[['log_av1','log_value','log_medhinc','college','white','Chicago','NSCC']]
X = sm.add_constant(X)
LPMmodel = sm.OLS(y,X)
LPMresult = LPMmodel.fit()
predY = LPMresult.predict(X)
```

[48]:
```
ax = sns.lineplot(x=df["log_av1"], y=predY)
ax.set(xlabel="log_av1", ylabel="Predicted Probability of Winning Appeal")
```

[48]:
```
[Text(0.5, 0, 'log_av1'),
 Text(0, 0.5, 'Predicted Probability of Winning Appeal')]
```



[ ]:
```
LPMresult.summary()
```

The result shows that our second assumption was right. White neighborhood has a better chance of winning appeal. However, with this analysis it is hard to prove relationship between winning an appea and income. Because the p-value is too high, it is hard to assert statistical significance.

[ ]:
```

```

[ ]:
```

```