STAT 303-2 Assignment 3 Complete

January 31, 2022

1 STAT303-2: Assignment 3

1.1 Instructions:

a. You may talk to a friend, discuss the questions and potential directions for solving them. However, you need to write your own solutions and code separately, and not as a group activity.

b. Do not write your name on the assignment. (1 point)

c. Export your Jupyter notebook as a PDF file. If you get an error, make sure you have downloaded the MikTex software (for windows) or MacTex (for mac). Note that after installing MikTex/MacTex, you will need to check for updates, install the updates if needed, and re-start your system. Submit the PDF file. (1 point)

d. Please include each question (or question number) followed by code and your answer (if applicable). Write your code in the 'Code' cells and your answer in the 'Markdown' cells of the Jupyter notebook. Ensure that the solution is well-organized, clear, and concise (3 points)

1. It's easy enough to identify different sections of the homework assignment (e.g., if there are different sections of an assignment, they're clearly distinguishable by section headers or the like)

2. It's clear which code/markdown blocks correspond to which questions.
3. There aren't excessively long outputs of extraneous information (e.g., no printouts of entire data frames without good reason)

This assignment is **due at 11:59pm on Wednesday, January 9th**. Good luck!

Submissions will be graded with a maximum of **52 points** – 47 points for code & answers, 5 points for anonymity and proper formatting.

1.2 Part 1

The datasets *house_feature_train.csv*, *house_price_train.csv*, *house_feature_test.csv*, and *house_price_test.csv* provide data on housing features and prices.

This part is worth 23 points overall.

(1a) Using house_feature_train.csv and house_price_train.csv, fit a multiple linear regression model without transformation to predict *house_price* based on *distance_MRT*, *latitude*, and *longitude*, *house_age*, and *number_convenience_stores*.

Print the model summary. What is the R^2 value?

(2 points for code, 1 point for answer)

[1]:
```
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
```

[2]:
```
trainf = pd.read_csv('house_feature_train.csv')
trainp = pd.read_csv('house_price_train.csv')
testf = pd.read_csv('house_feature_test.csv')
testp = pd.read_csv('house_price_test.csv')
train = pd.merge(trainf,trainp)
test = pd.merge(testf,testp)
train.head()
```

[2]:
```
   house_id  house_age  distance_MRT  number_convenience_stores  latitude  \
0       210        5.2      390.5684                          5  24.97937
1       190       35.3      616.5735                          8  24.97945
2       328       15.9     1497.7130                          3  24.97003
3         5        7.1     2175.0300                          3  24.96305
4       412        8.1      104.8101                          5  24.96674

   longitude  house_price
0  121.54245      2724.84
1  121.53642      1789.29
2  121.51696       556.96
3  121.51254      1030.41
4  121.54067      2756.25
```

[3]:
```
ols_object = smf.ols(formula
 ↪='house_price~house_age+distance_MRT+number_convenience_stores+latitude+longitude',
 ↪data = train)
model = ols_object.fit()
model.summary()
```

[3]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:            house_price   R-squared:                       0.567
Model:                            OLS   Adj. R-squared:                  0.558
```

```
Method:                 Least Squares   F-statistic:                     70.32
Date:                Mon, 31 Jan 2022   Prob (F-statistic):           7.57e-47
Time:                        02:01:27   Log-Likelihood:                -2187.9
No. Observations:                 275   AIC:                             4388.
Df Residuals:                     269   BIC:                             4410.
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                                 coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
Intercept                  -4.833e+05   5.79e+05     -0.835      0.404
-1.62e+06    6.56e+05
house_age                    -23.1472      3.786     -6.113      0.000
-30.602     -15.692
distance_MRT                  -0.2659      0.068     -3.886      0.000
-0.401      -0.131
number_convenience_stores    111.8909     17.743      6.306      0.000
76.959     146.823
latitude                   1.926e+04   4362.392      4.414      0.000
1.07e+04    2.78e+04
longitude                    35.8774   4570.578      0.008      0.994
-8962.778    9034.532
==============================================================================
Omnibus:                       62.033   Durbin-Watson:                   2.253
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              113.606
Skew:                           1.195   Prob(JB):                     2.14e-25
Kurtosis:                       5.049   Cond. No.                     2.28e+07
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 2.28e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

The R^2 value is 0.567.

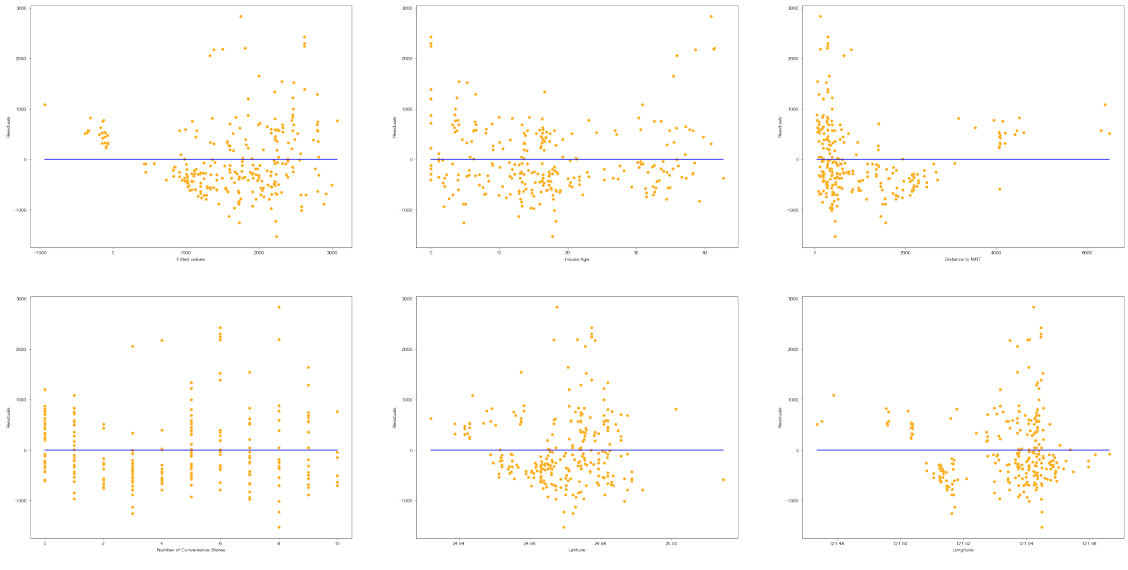
(1b) Obtain the residuals and plot them separately against fitted values and each of the five feature variables. Make one plot including the 6 subplots.

(3 points for visualization)

[4]:
```
fig, axs = plt.subplots(2, 3, figsize=(40, 20))
sns.set(font_scale = 2.5)
```

```
sns.scatterplot(ax = axs[0,0],x = model.fittedvalues, y=model.resid,color =␣
↪'orange')
sns.lineplot(ax = axs[0,0],x = [model.fittedvalues.min(),model.fittedvalues.
↪max()],y = [0,0],color = 'blue')
axs[0,0].set_xlabel('Fitted values')
axs[0,0].set_ylabel('Residuals')

sns.scatterplot(ax = axs[0,1],x = train.house_age, y=model.resid,color =␣
↪'orange')
sns.lineplot(ax = axs[0,1],x = [train.house_age.min(),train.house_age.max()],y␣
↪= [0,0],color = 'blue')
axs[0,1].set_xlabel('House Age')
axs[0,1].set_ylabel('Residuals')

sns.scatterplot(ax = axs[0,2],x = train.distance_MRT, y=model.resid,color =␣
↪'orange')
sns.lineplot(ax = axs[0,2],x = [train.distance_MRT.min(),train.distance_MRT.
↪max()],y = [0,0],color = 'blue')
axs[0,2].set_xlabel('Distance to MRT')
axs[0,2].set_ylabel('Residuals')

sns.scatterplot(ax = axs[1,0],x = train.number_convenience_stores, y=model.
↪resid,color = 'orange')
sns.lineplot(ax = axs[1,0],x = [train.number_convenience_stores.min(),train.
↪number_convenience_stores.max()],y = [0,0],color = 'blue')
axs[1,0].set_xlabel('Number of Convenience Stores')
axs[1,0].set_ylabel('Residuals')

sns.scatterplot(ax = axs[1,1],x = train.latitude, y=model.resid,color =␣
↪'orange')
sns.lineplot(ax = axs[1,1],x = [train.latitude.min(),train.latitude.max()],y =␣
↪[0,0],color = 'blue')
axs[1,1].set_xlabel('Latitude')
axs[1,1].set_ylabel('Residuals')

sns.scatterplot(ax = axs[1,2],x = train.longitude, y=model.resid,color =␣
↪'orange')
sns.lineplot(ax = axs[1,2],x = [train.longitude.min(),train.longitude.max()],y␣
↪= [0,0],color = 'blue')
axs[1,2].set_xlabel('Longitude')
axs[1,2].set_ylabel('Residuals')
```

[4]: Text(0, 0.5, 'Residuals')



(1c) Comment on the plot of residuals against fitted values. Does the model violate the assumption of linearity? Does the model violate constant variance assumption?

(3 points for answer)

The residuals are not scattered around 0 evenly. They are mostly positive for fitted values below 500 and negative for fitted values between 500 to 1500. The residuals seem to vary in a systematic patern between positive and negative. It indicates that the linearity assumption is violated. The variance of error term tends to be smaller for smaller values, which indicates that the constant variance assumption is violated.

(1d) Comment on the plot of residuals against the predictor variables. On the basis of these plots, should any further modifications of the regression model be attempted?

(2 points for answer)

From the plot of residuals against house age, the residuals now seem to be scattered around 0 evenly.

From the plot of residuals against Distance MRT, latitude, and longitude, the residuals seems to have a non-linear pattern and transformation of the variables can be attemped.

From the plot of residuals against number of convenience stores, the residuals seem to be scattered around 0 evenly other than for value 3, 4, and 6. The shape shows some sign of being curvilinear, and adding a higher order term might be helpful.

(1e) Calculate the RMSE using the test datasets for the model constructed in (a).

(2 points for code)

[5]:
```
#Computing RMSE on test data
pred_house_price = model.predict(testf)
np.sqrt(((testp.house_price - pred_house_price)**2).mean())
```

[5]:
```
510.0202466825708
```

(1f) Using appropriate transformation(s) and/or variable interaction(s), update the model in (a) to obtain a model that has an R-squared of at least 80%, and a RMSE (Root mean squared error) of at max $350k on test data.

Print the model summary and report the R-squared, and RMSE on test data.

(5 points for code)

Note:

(1) House prices are provided in thousands of dollars. A value of 556 in the *house_price* column indicates a house price of $556k.

(2) The test datasets are *house_feature_test.csv* and *house_price_test.csv*.

(3) R-squared is computed on training data, and RMSE is computed on test data.

[6]:
```
ols_object2 = smf.ols(formula ='np.
 ↪log(house_price)~house_age+I(house_age**2)+np.
 ↪log(distance_MRT)+number_convenience_stores+latitude+I(latitude**2)+longitude+I(longitude**
 ↪log(distance_MRT)*longitude', data = train)
model2 = ols_object2.fit()
model2.summary()
```

[6]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:     np.log(house_price)   R-squared:                       0.830
Model:                             OLS   Adj. R-squared:                  0.824
Method:                  Least Squares   F-statistic:                     129.2
Date:                 Mon, 31 Jan 2022   Prob (F-statistic):           1.31e-95
Time:                         02:01:28   Log-Likelihood:                -77.596
No. Observations:                  275   AIC:                             177.2
Df Residuals:                      264   BIC:                             217.0
Df Model:                           10
Covariance Type:             nonrobust
==============================================================================
======================
                                    coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
----------------------
```

```
Intercept                          -5.748e+06   1.75e+06     -3.282      0.001
-9.2e+06    -2.3e+06
house_age                            -0.0459      0.007     -6.676      0.000
-0.059      -0.032
I(house_age ** 2)                     0.0009      0.000      5.249      0.000
0.001       0.001
np.log(distance_MRT)                1150.6967    417.912      2.753      0.006
327.832    1973.562
number_convenience_stores             29.7825     17.157      1.736      0.084
-4.000      63.565
latitude                            6576.8171   5723.853      1.149      0.252
-4693.394    1.78e+04
I(latitude ** 2)                    -131.1884    114.626     -1.144      0.253
-356.885      94.509
longitude                           9.316e+04   2.83e+04      3.286      0.001
3.73e+04    1.49e+05
I(longitude ** 2)                   -382.9563    116.539     -3.286      0.001
-612.420    -153.492
number_convenience_stores:latitude    -1.1918      0.687     -1.735      0.084
-2.544       0.161
np.log(distance_MRT):longitude        -9.4699      3.438     -2.754      0.006
-16.240      -2.700
==============================================================================
Omnibus:                        3.665   Durbin-Watson:                   2.121
Prob(Omnibus):                  0.160   Jarque-Bera (JB):                3.456
Skew:                          -0.197   Prob(JB):                        0.178
Kurtosis:                       3.383   Cond. No.                     1.31e+12
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The smallest eigenvalue is 3.5e-14. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
"""
```

The R-squared of the transformed model is 83%

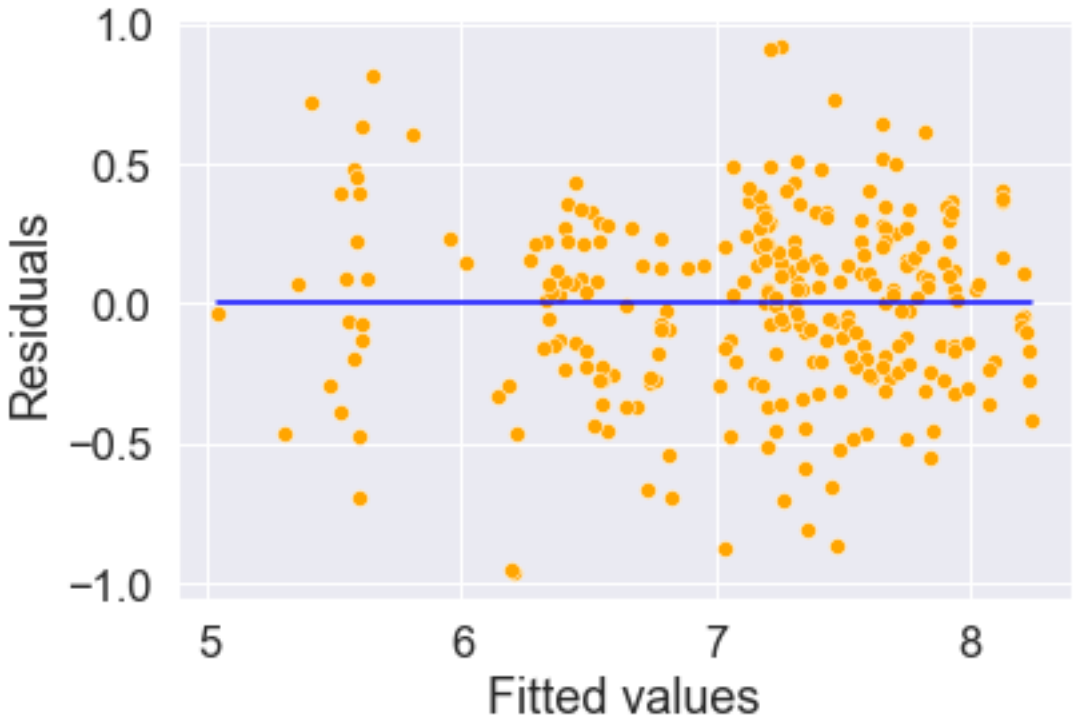
(1g) Are the assumptions of linearity and constant variance of errors satisfied in the model developed in the previous question? Make the appropriate plot and use it to answer the question.

(3 points for visualization, 2 points for answer)

[7]:
```
sns.set(font_scale = 1.5)
a=sns.scatterplot(x = model2.fittedvalues, y=model2.resid,color = 'orange')
sns.lineplot(x = [model2.fittedvalues.min(),model2.fittedvalues.max()],y =␣
↪[0,0],color = 'blue')
a.set_xlabel('Fitted values')
```

```
a.set_ylabel('Residuals')
```

[7]: `Text(0, 0.5, 'Residuals')`



The residuals now seem to scatter around 0 evenly, and the variance of error terms in more or less constant. The model assumptions of linearity and constant variance are met.

1.3 Part 2

The datasets `Austin_Affordable_Housing_Train.csv` and `Austin_Affordable_Housing_Test.csv` provide data on housing development projects that have received funding from the Affordable Housing Development Fund in Austin, Texas. The city provides property developers with tax credits and other forms of funding in exchange for agreements to set housing prices (e.g. rent) below market rate.

Each row represents a housing development in Austin. Variables include the amount (USD) provided by the city, the status of the housing project, the number of housing units, the period of affordability, and more. The data provided is a modification of an Affordable Housing Inventory found at https://data.austintexas.gov/Housing-and-Real-Estate/City-of-Austin-Affordable-Housing-Inventory/x5p7-qyuv.

Let's say that you're hired by the city as a consultant to work with subject matter experts in their Housing and Planning Department.

General Hint: For written sections, writing "it depends" (along with an explanation) often charac-

terizes a good answer.

This part is worth 24 points overall.

(2a) Suppose you run the line `status_vars = pd.get_dummies(housing_dataframe["Status"])`, append the columns of `status_vars` to your original data frame, and use the columns as predictors in a linear regression model. What potential problem would you likely be introducing into the model? How could it affect your results?

(2 points for answers)

We'd probably be introducing multicollinearity into our model, given that we're including all levels of a categorical variable—when we know all but one level of the variable, we can calculate the last value. Therefore OLS cannot distinguish between the effects of individual levels, and may result in a model with inflated variance. This could increase the standard error of coefficients, ultimately increasing p-values and decreasing our model's power.

(2b) Suppose that a subject matter expert recommends using the variables *Total_Units*, *Total_Affordable_Units*, *Total_Accessible_Units*, and *Market_Rate_Units* as predictors in your model. From a regression modeling standpoint, does this sound advisable? Produce metrics to quantify the potential impact of including the four predictors in a model. Interpret at least one of the metrics you provide, both statistically and in the context of the problem.

(2 points for code, 2 points for answer)

[8]:
```
tx_df0 = pd.read_csv("Austin_Affordable_Housing_Train.csv")
tx_df0.shape
```

[8]:
```
(200, 17)
```

[9]:
```
unit_predictors =␣
↪["Market_Rate_Units","Total_Affordable_Units","Total_Accessible_Units","Total_Units"]

for p in range(4):
    formula_p = unit_predictors[p] + " ~ " + (" + ").join([unit_predictors[x]␣
↪for x in range(4) if not x==p])
    print("- "+unit_predictors[p]+" VIF: "+str(1/(1 - smf.
↪ols(formula=formula_p,data = tx_df0).fit().rsquared)))
```

```
- Market_Rate_Units VIF: 40.57175457590176
- Total_Affordable_Units VIF: 159.751905045582
- Total_Accessible_Units VIF: 2.0339272685656833
- Total_Units VIF: 234.5282791807926
```

The variance inflation factor associated with *Total_Units* is 234.5282791807926, indicating that the ratio of the variance when fitting the model with all four predictors to the variance when fitting the model with *Total_Units* alone is 234.5282791807926. Clearly, there is a significant multicollinearity issue here and all four variables should not be included in a model.

If we know the number of market rate units, affordable units, and accessible units for a given housing property, we already have a good degree of information regarding the total number of

units. Using all four variables would be redundant.

(2c) Say that the subject matter expert agrees to use *Total_Affordable_Units*, *Affordability_Expiration_Year*, and *Units_Under_50_Percent_MFI* as predictors for *City_Amount*. Fit the appropriate model (without transformations). Then interpret the results associated with *Total_Affordable_Units*, as well as the overall model fit.

(1 point for code, 2 points for answer)

[10]:
```
#Not Required, just checking
unit_predictors =␣
↪["Total_Affordable_Units","Affordability_Expiration_Year","Units_Under_50_Percent_MFI"]

for p in range(3):
    formula_p = unit_predictors[p] + " ~ " + (" + ").join([unit_predictors[x]␣
↪for x in range(3) if not x==p])
    print("- "+unit_predictors[p]+" VIF: "+str(1/(1 - smf.
↪ols(formula=formula_p,data = tx_df0).fit().rsquared)))
```

```
- Total_Affordable_Units VIF: 2.042548048420181
- Affordability_Expiration_Year VIF: 1.1216420882990215
- Units_Under_50_Percent_MFI VIF: 2.2148085265398234
```

[11]:
```
tx_model = smf.ols(formula="City_Amount ~ Total_Affordable_Units + \
Affordability_Expiration_Year + Units_Under_50_Percent_MFI",data = tx_df0).fit()
tx_model.summary()
```

[11]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:            City_Amount   R-squared:                       0.491
Model:                            OLS   Adj. R-squared:                  0.483
Method:                 Least Squares   F-statistic:                     63.08
Date:                Mon, 31 Jan 2022   Prob (F-statistic):           1.37e-28
Time:                        02:01:28   Log-Likelihood:                -3056.1
No. Observations:                 200   AIC:                             6120.
Df Residuals:                     196   BIC:                             6133.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                                 coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
Intercept                  -8.409e+06   4.35e+06     -1.933      0.055
-1.7e+07    1.72e+05
```

```
Total_Affordable_Units         2.261e+04   2059.042     10.983      0.000
1.86e+04    2.67e+04
Affordability_Expiration_Year  4233.1306   2125.847      1.991      0.048
40.661    8425.600
Units_Under_50_Percent_MFI    -8946.2892   3784.569     -2.364      0.019
-1.64e+04   -1482.585
==============================================================================
Omnibus:                      137.743   Durbin-Watson:                   1.366
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3152.792
Skew:                           2.171   Prob(JB):                         0.00
Kurtosis:                      21.960   Cond. No.                     1.19e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.19e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

All else being equal, a one-unit increase in the total number of affordable units for a given property corresponds to a 22,610 USD increase in the amount of funding, on an average, provided by the city. This coefficient estimate has a test statistic of 10.983 and a corresponding p-value of less than 0.001, so at the $\alpha = 0.01$ level of significance we can reject the null hypothesis that there is no association between number of affordable units and the dollar amount provided by the city.

The R^2_{adj} value of 0.483 suggests that our set of predictors explains a rather modest amount of variance in *City_Amount*. So we may not have good predictive or explanatory power using this model (although I'd argue that the significance of individual coefficients should still be informative). An F-statistic of 63.08 and accompanying p-value of 1.37e-28 yields more than enough evidence to reject our null hypothesis that all coefficients are equal to zero (at the $\alpha = 0.01$ level of significance).

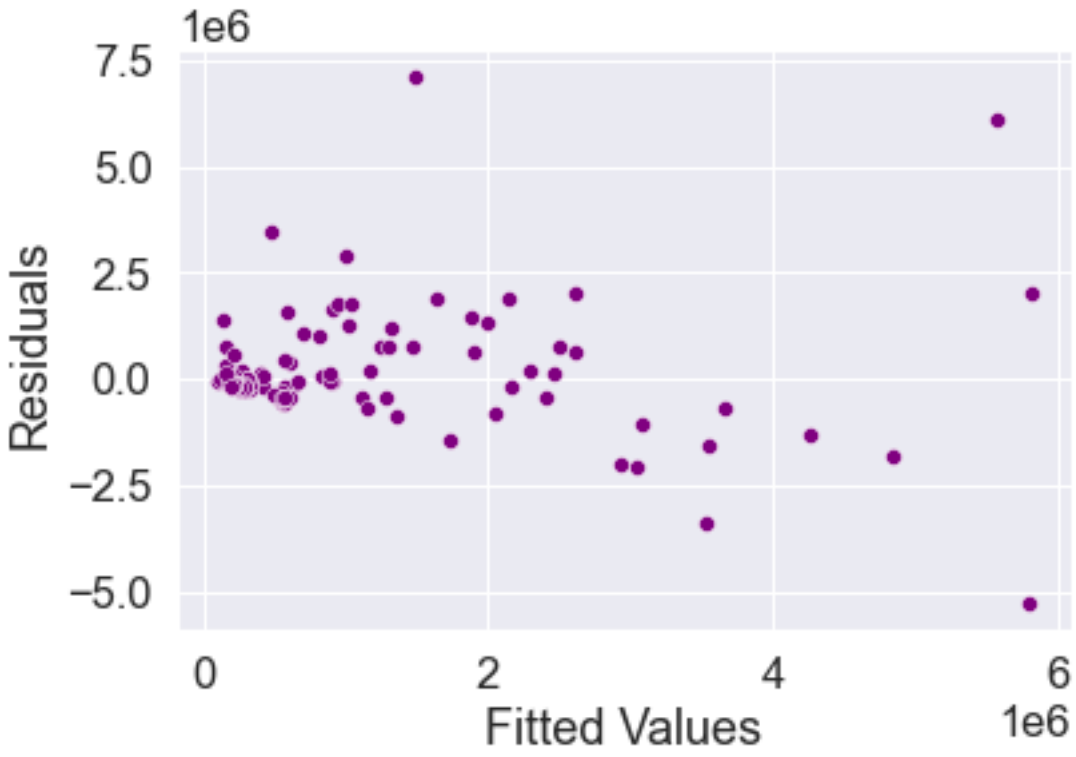
(2d) Using visualizations, investigate whether the model you fit in (2c) yields outlying observations. What count and proportion of observations would you classify as outliers?

Note: Show separate plots for both - residuals and studentized residuals. However, consider studentized residuals when identifying outliers.

(2 points for code, 1 point for answer)

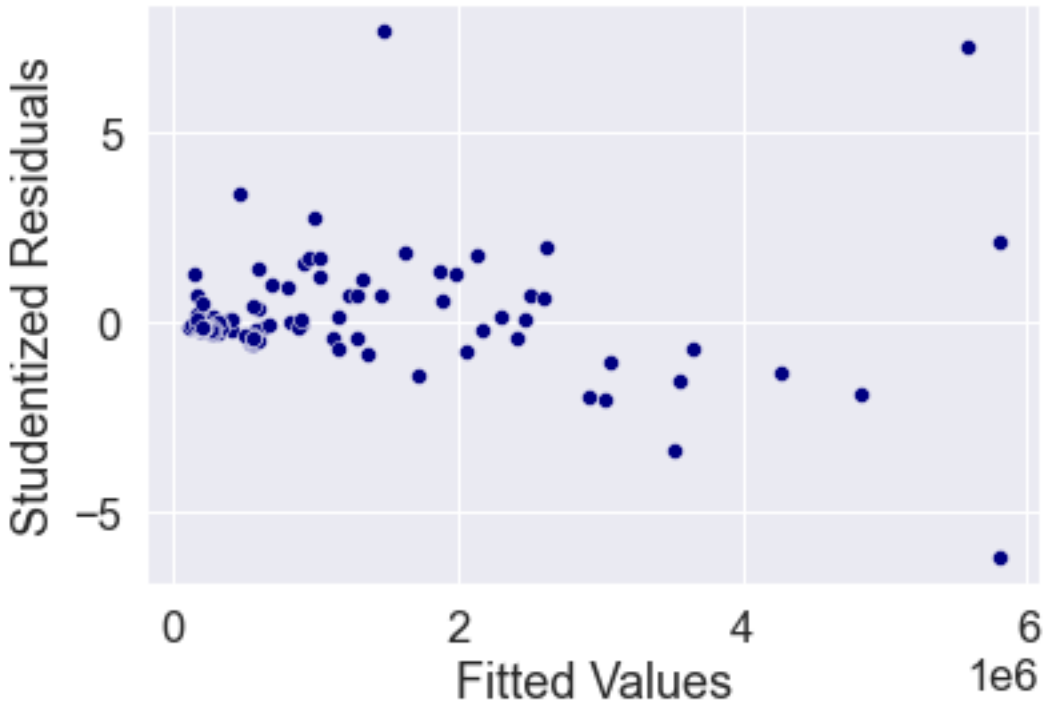
[12]:
```
sns.scatterplot(x = (tx_model.fittedvalues), y=(tx_model.resid),color ='purple')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
```

[12]:
```
Text(0, 0.5, 'Residuals')
```



[13]:
```
sns.scatterplot(x = (tx_model.fittedvalues), y=(tx_model.outlier_test().
↪student_resid),color ='navy')
plt.xlabel('Fitted Values')
plt.ylabel('Studentized Residuals')
```

[13]: Text(0, 0.5, 'Studentized Residuals')



[14]:
```
np.sum(np.abs(tx_model.outlier_test().student_resid)>3),np.mean(np.abs(tx_model.
↪outlier_test().student_resid)>3)
```

[14]: (5, 0.025)

Given the studentized residuals from our fitted model, there are five outliers which represent 2.5% of observations (based on the above definition of outlier).

(2e) Based on your results in (2d), would you choose to remove outlying observations? Briefly, why or why not?

(1 point for answer)

It depends on our goal for the model – most likely not, given that the p-values seem "good enough" in this case and outliers shouldn't affect the OLS line, but if we want a model with more explanatory power as measured by R^2_{adj} we may choose to remove outliers.

(2f) Consider a scenario in which the model will be used by property owners seeking to predict the amount of money they may receive from the city of Austin. How would this change, support, or complicate your answer in (2e), if at all?

(1 point for answer)

This probably wouldn't change my answer—outliers typically don't change the OLS line, and therefore their removal wouldn't affect prediction that much.

(2g) Say that the model will be used by a team of sociologists seeking statistical evidence at the $\alpha = 0.01$ significance level that a property's affordability expiration year has an effect on the amount of money issued by the city of Austin? How would this change, support, or complicate your answer in (2e), if at all?

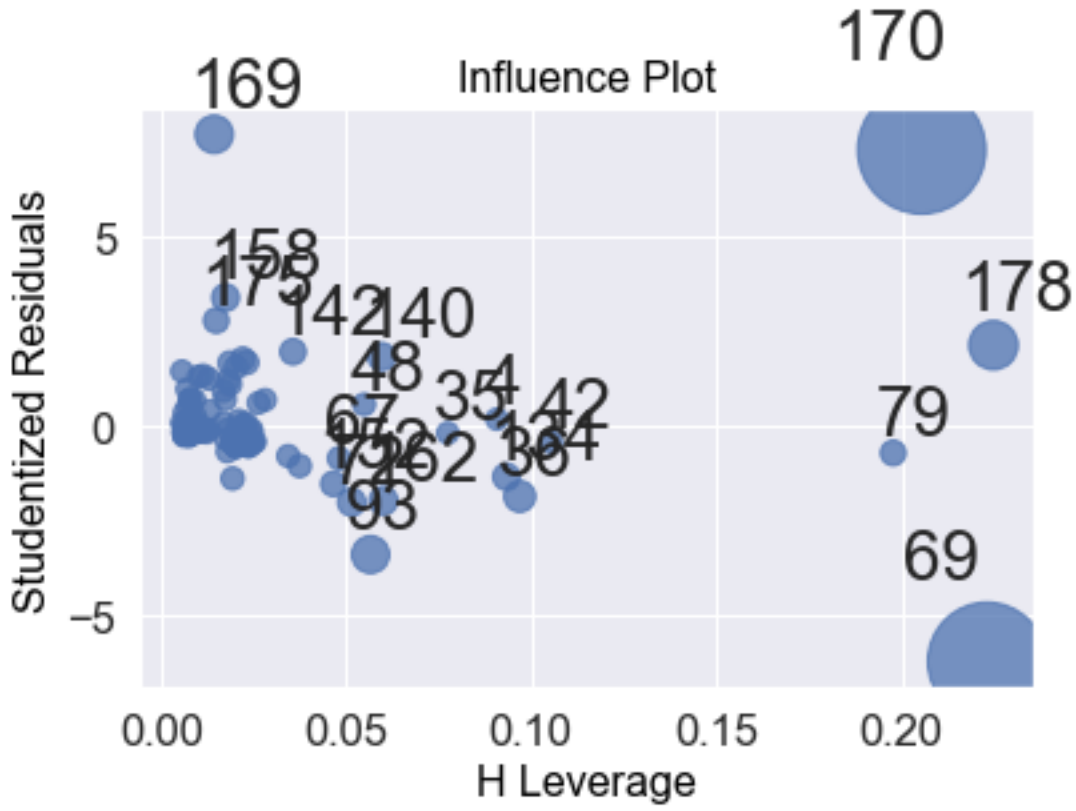
(1 point for answer)

This might change my answer, since the $\alpha = 0.01$ threshold isn't reached by the p-values associated with some coefficients, and we might want higher power. On the other hand, I'd be wary of removing outliers due to concerns that could be represented as "p-hacking."

(2h) Determine whether the model you fit in (2c) contains any high-leverage points. Produce a visualization, then report the count and proportions of observations that are high-leverage (defining an observation as "high-leverage" if its leverage statistic is greater than four times the average leverage statistic).

(2 points for code, 1 point for answer)

[15]:
```
influence = tx_model.get_influence()
leverage = influence.hat_matrix_diag

sm.graphics.influence_plot(tx_model);
```



[16]:
```
out = tx_model.outlier_test()
cutoff = 4*(tx_model.df_model+1)/tx_model.nobs
np.sum(leverage>cutoff), np.mean(leverage>cutoff)
```

[16]:
```
(8, 0.04)
```

Based on our visualization and calculation with a cutoff of four times the average leverage statistic, there are 8 observations that are high-leverage, or 4.00% of observations.

(2i) Based on your results in (2h), would you choose to remove high-leverage observations? Briefly explain, why or why not?

(1 point for answer)

Since high-levarage observations may inflate R^2_{adj} and artificially deflate standard errors (and thus p-values), I'd choose to remove high-leverage observations to ensure the integrity of statistical inference. Especially since some high-leverage observations may also be outliers (known as influential points), removing high-leverage observations may also correct the OLS line and therefore better the model's predictive abilities.

(2j) Identify and remove any influential points from the training data and refit the model. How does removing influential affect the model, if at all?

Think about using the model summary, and the test data provided.

(3 points for code, 2 points for answer)

[17]:
```
tx_test = pd.read_csv("Austin_Affordable_Housing_Test.csv")
tx_test.shape
```

[17]:
```
(43, 17)
```

[18]:
```
pred_amount = tx_model.predict(tx_test)
np.sqrt(((tx_test.City_Amount - pred_amount)**2).mean())
```

[18]:
```
787388.368551412
```

[19]:
```
tx_filtered_influential = tx_df0.drop(np.intersect1d(np.where(np.abs(tx_model.
↪outlier_test().student_resid)>3)[0],np.where(leverage>cutoff)[0]))
tx_filtered_influential.shape
```

[19]:
```
(198, 17)
```

[20]:
```
tx_model_filtered_influential = smf.ols(formula="City_Amount ~␣
↪Total_Affordable_Units + \
Affordability_Expiration_Year + Units_Under_50_Percent_MFI",data =␣
↪tx_filtered_influential).fit()
tx_model_filtered_influential.summary()
```

[20]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:            City_Amount   R-squared:                       0.504
Model:                            OLS   Adj. R-squared:                  0.497
Method:                 Least Squares   F-statistic:                     65.77
Date:                Mon, 31 Jan 2022   Prob (F-statistic):           2.20e-29
Time:                        02:01:29   Log-Likelihood:                -2991.3
No. Observations:                 198   AIC:                             5991.
Df Residuals:                     194   BIC:                             6004.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
=================
                                 coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
-----------------
Intercept                   -8.381e+06   3.66e+06     -2.289      0.023
-1.56e+07   -1.16e+06
Total_Affordable_Units        2.17e+04   2272.150      9.550      0.000
1.72e+04    2.62e+04
Affordability_Expiration_Year  4219.4469   1788.564      2.359      0.019
691.921    7746.973
Units_Under_50_Percent_MFI    -7698.1883   3746.489     -2.055      0.041
-1.51e+04    -309.109
==============================================================================
Omnibus:                      178.422   Durbin-Watson:                   1.709
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             4872.165
Skew:                           3.211   Prob(JB):                         0.00
Kurtosis:                      26.438   Cond. No.                     1.19e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.19e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
"""
```

[21]:
```
pred_amount = tx_model_filtered_influential.predict(tx_test)
np.sqrt(((tx_test.City_Amount - pred_amount)**2).mean())
```

[21]:
```
778947.897320495
```

In the new model based on the data without influential points, we observe a slight increase in R-squared value. This was expected as influential points do not conform with the general trend

of the rest of the data. However, the increase in R-squared is very small, which may be due to a small amount of aggregate influence of the two influential points. Note that the multiple influential points may also cancel or reduce each other's influence.

Removing influential points is likely help us better capture the general trend in the data, and develop a more accurate model. This is true in this case, as we observe the RMSE reducing slightly on test data.