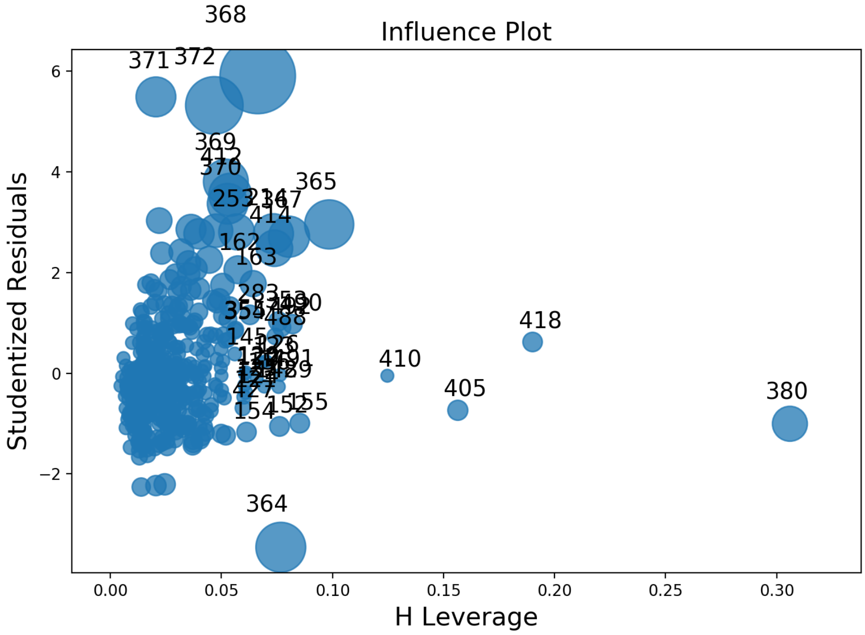
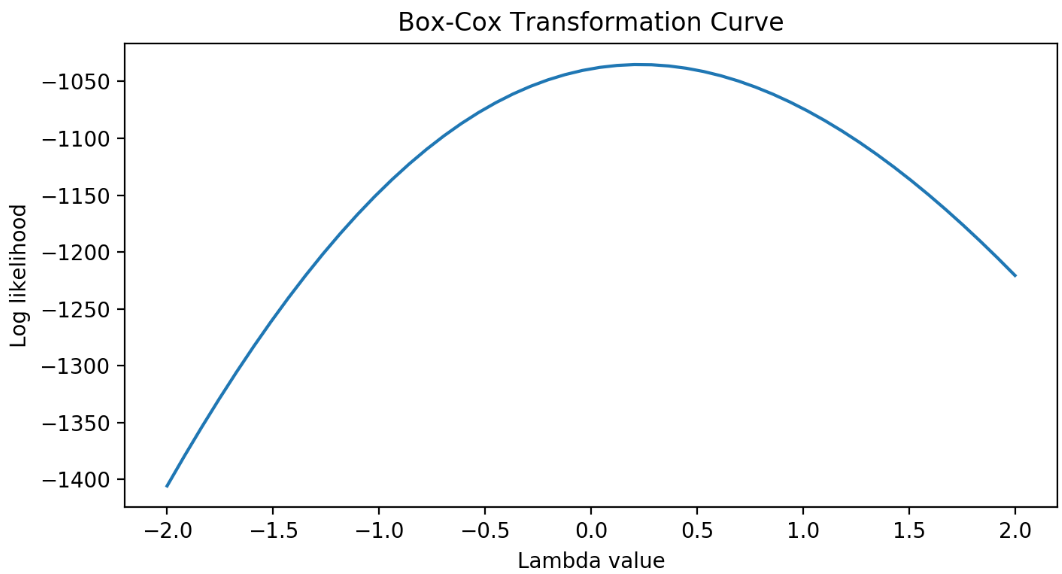
CS 498 HW 6 Yu Che Wang/ yuchecw2

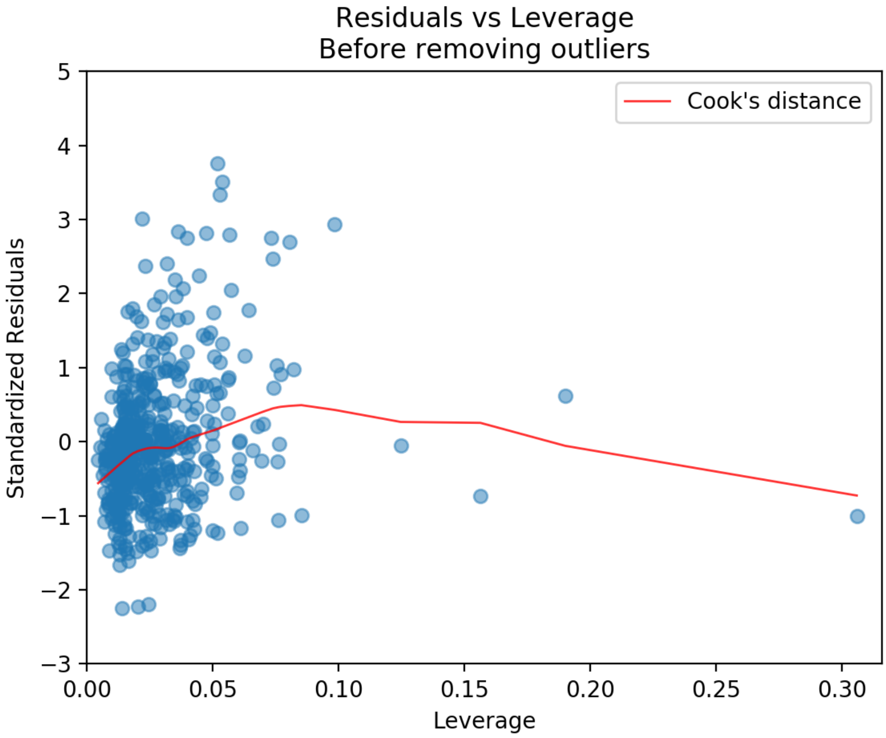
1. All the points (row number) I removed (indexed on the original dataset) as outlier points by looking at the influence plot:
   364, 365, 368, 370, 371, 372, 380, 405, 410, 418.
   I remove those points with high leverage, standardized residuals, and cook distance. The radius of the data point represents its cook distance.



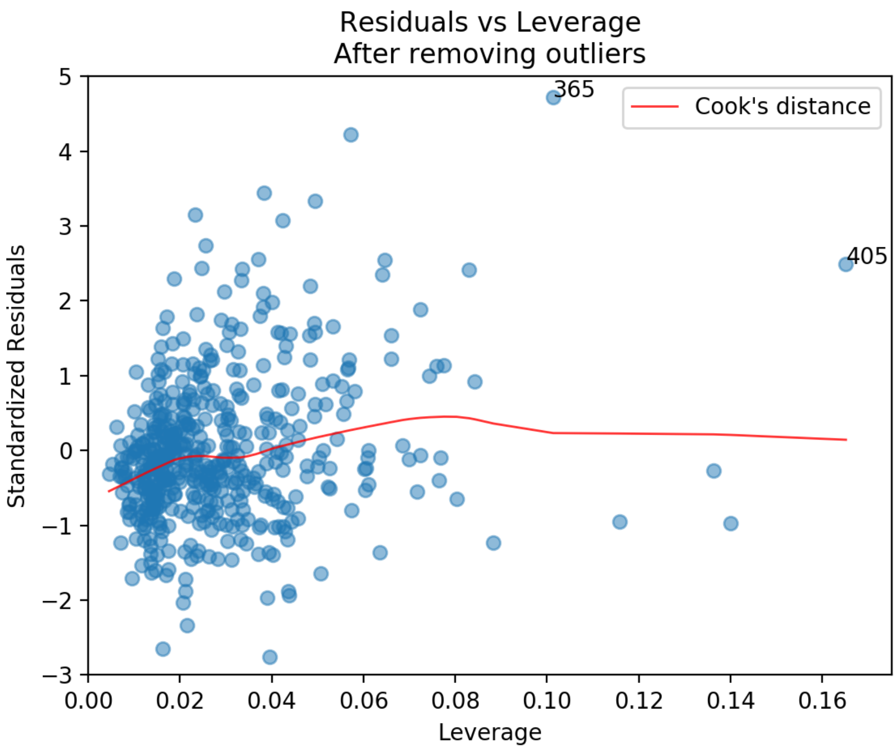
2. By using scipy.stats.boxcox, we obtain the best lambda value = 0.219.



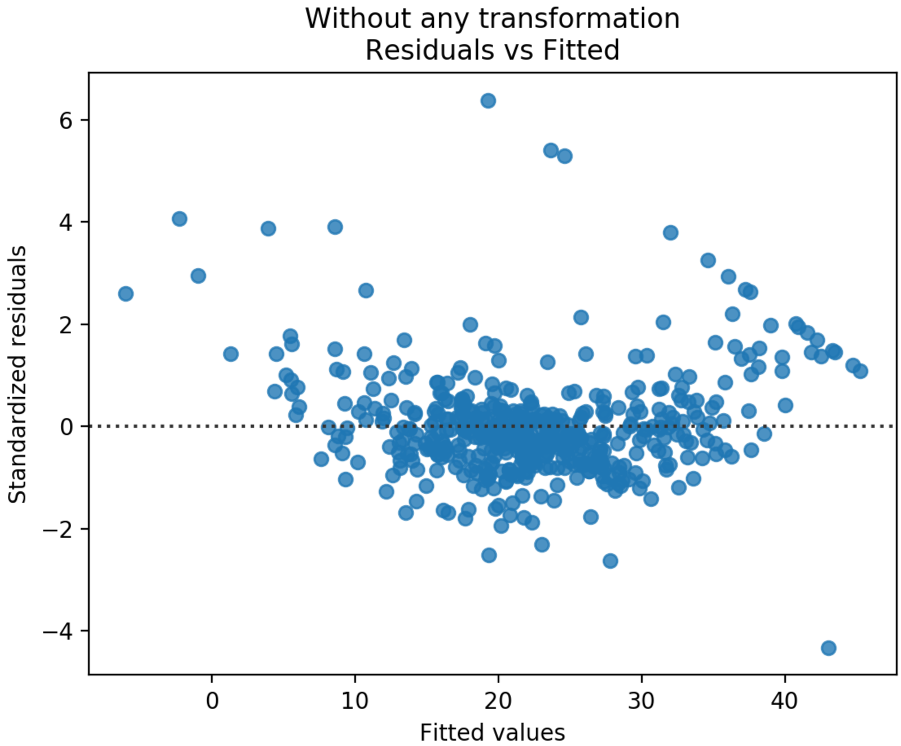
3. Diagnostic plot before removing outliers



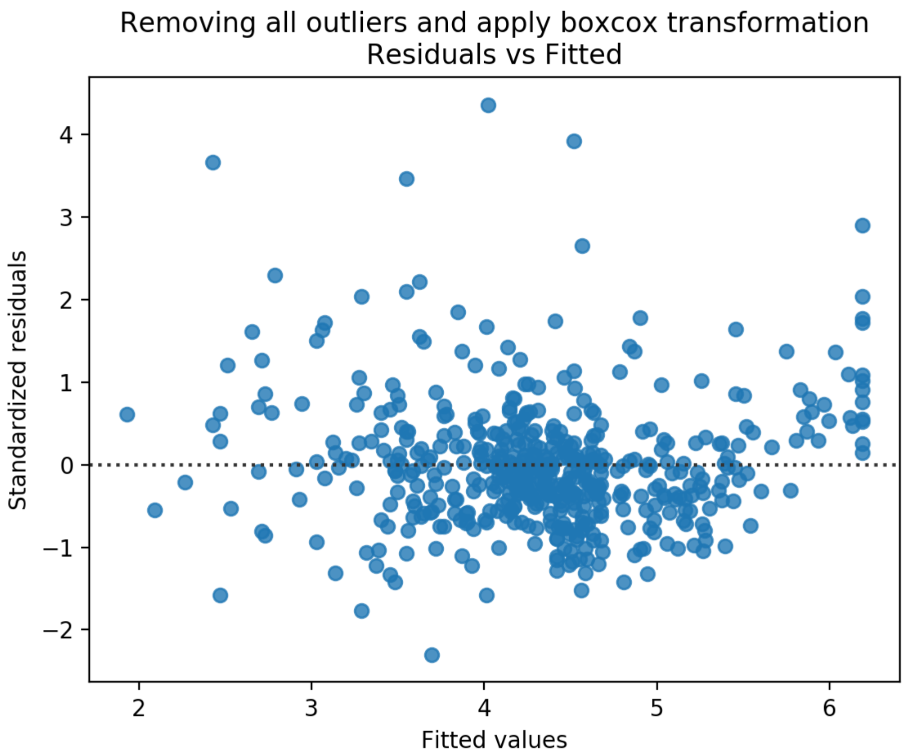
Diagnostic plot after removing outliers



4. Raw data

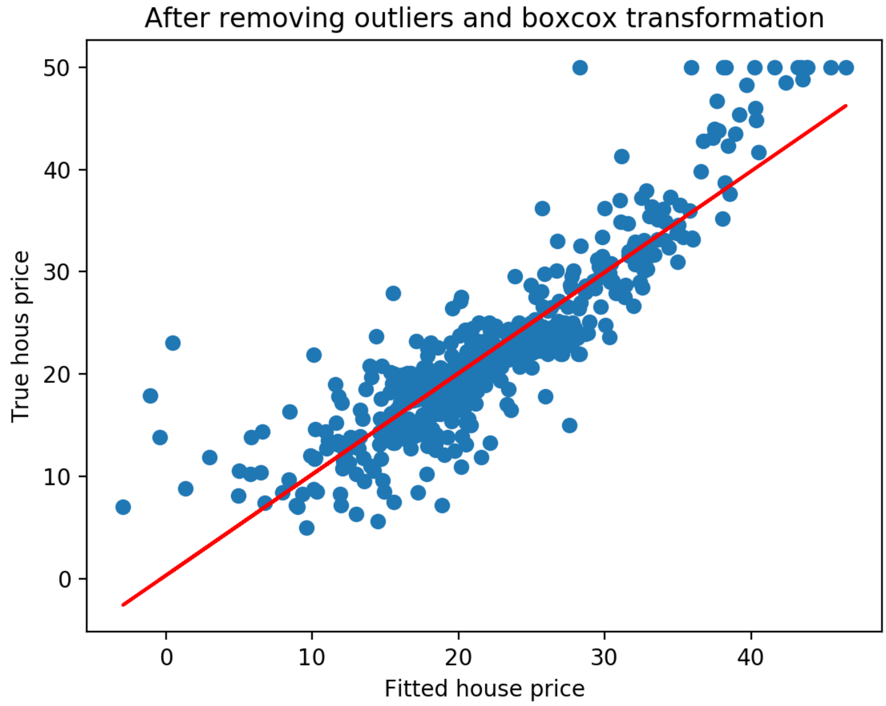


5. Removing all outliers and transforming the dependent variable



6. In the first plot, there are some outliers, data that are 4-6 standardized residuals away from the mean. Most of the data points in the second plot are less than two standardized residuals away from the mean. In addition, the data points are evenly distributed across the dotted line.

7. Fitted house price vs True house price



The data points lie on a line, which means that after removing outliers and applying box-cox transformation, the residuals are nearly zero.

8. Code fragments

Linear regression

```
# Linear regression
X = data[:,:13]
y = data[:,13].reshape(-1,1)
# Method 1
lm = sm.OLS(y, sm.add_constant(X)).fit()
# Method 2
reg = LinearRegression().fit(X, y)
r_squared = reg.score(X,y)
```

Box-cox transformation

```
# Boxcox transformation
y_boxcox, maxlog = stats.boxcox(y_remove)
```

Box-cox transformation with different values

```
n_interval = 50
lmbda_list = np.linspace(-2, 2, n_interval)
llf_list = np.zeros(n_interval)
for i in range(lmbda_list.shape[0]):
    llf_list[i] = stats.boxcox_llf(lmbda_list[i], y_remove)
```

To apply box-cox transformation with a specific value lmbda, one can use

y_boxcox = stats.boxcox(y, lmbda=lmbda)

9. Entire code

```python
import numpy as np
import numpy.linalg as la
import pandas as pd
import seaborn as sns
import scipy.stats as stats
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from statsmodels.stats.outliers_influence import OLSInfluence
import statsmodels.api as sm
import statsmodels.formula.api as smf
# Load the data
data = np.loadtxt('data.txt')
df = pd.DataFrame(data)

# Linear regression
X = data[:,:13]
y = data[:,13].reshape(-1,1)
# Method 1
lm = sm.OLS(y, sm.add_constant(X)).fit()
# Method 2
reg = LinearRegression().fit(X, y)
r_squared = reg.score(X,y)

# Generate influence plot
fig, ax = plt.subplots(figsize=(9,6))
fig = sm.graphics.influence_plot(lm, alpha=0.001, ax=ax, criterion='cooks')
fig.show()

####################
# Calculate leverage
####################
beta_hat = np.dot(la.inv(np.dot(X.T, X)), np.dot(X.T, y))
hat_matrix = X@la.inv(np.dot(X.T, X))@(X.T)
# plt.bar(list(range(hat_matrix.shape[0])),np.diag(hat_matrix))
# plt.boxplot(np.diag(hat_matrix))
leverage = np.diag(hat_matrix)
# Remove leverage outliers and record their indices
outliers_num_leverage= []
for i in range(leverage.shape[0]):
    if (leverage[i] > np.percentile(leverage, 75)+1.5*stats.iqr(leverage)):
        outliers_num_leverage.append(i)
leverage_remove_outliers = leverage[leverage <= np.percentile(leverage, 75)+1.5*stats.iqr(leverage)]
```

```python
# Calculate residuals and mean square error
y_predicted = X@beta_hat # Fitted value
e = y - X@beta_hat
N = y.shape[0]
mean_squared_error = np.dot(e.T, e)/N
# r_squred = np.var(X@beta_hat)/np.var(y)

###########################################
# Calculate the cook distance for each data
###########################################
cook_distance = np.zeros((y.shape[0],1))
for i in range(y.shape[0]):
    X_remove = np.delete(X, i, 0)
    y_remove = np.delete(y, i, 0)
    beta_i = np.dot(la.inv(np.dot(X_remove.T, X_remove)), np.dot(X_remove.T, y_remove))
    y_p = X_remove@la.inv(np.dot(X_remove.T, X_remove))@(X_remove.T)@y_remove
    cook_distance[i] = np.dot((y_p-X_remove@beta_i).T, y_p-X_remove@beta_i)/(N*mean_squared_error)
# plt.bar(list(range(y.shape[0])), cook_distance)
# plt.boxplot(cook_distance)
outliers_num_cook_distance = []
for i in range(cook_distance.shape[0]):
    if (cook_distance[i] > np.percentile(cook_distance, 75)+1.5*stats.iqr(cook_distance)):
        outliers_num_cook_distance.append(i)
cook_distance_remove_outliers = cook_distance[cook_distance <= np.percentile(cook_distance, 75)+1.5*stats.iqr(cook_distance)]

###################################
# Calculate standardized residuals
###################################
s = np.zeros((y.shape[0], 1))
for i in range(y.shape[0]):
    s[i] = e[i]/np.sqrt(mean_squared_error*(1-leverage[i]))
# plt.bar(list(range(y.shape[0])), np.abs(s))
# plt.boxplot(s)
outliers_num_s = []
for i in range(s.shape[0]):
    if (s[i] > np.percentile(s, 75)+1.5*stats.iqr(s)):
        outliers_num_s.append(i)
```

```python
# Remove outliers
by_eye = True
if by_eye:
    outliers_num = [364, 365, 368, 370, 371, 372, 380, 405, 410, 418]
else:
    outliers_num = np.union1d(outliers_num_leverage, outliers_num_cook_distance)
    outliers_num = np.union1d(outliers_num, outliers_num_s)

lm_remove = sm.OLS(y_remove, sm.add_constant(X_remove)).fit()
X_remove = np.delete(X, outliers_num, 0)
y_remove = np.delete(y, outliers_num, 0)
s_remove = np.delete(s, outliers_num, 0)

# Page1
n_interval = 50
lmbda_list = np.linspace(-2, 2, n_interval)
llf_list = np.zeros(n_interval)
for i in range(lmbda_list.shape[0]):
    llf_list[i] = stats.boxcox_llf(lmbda_list[i], y_remove)
fig, ax = plt.subplots(figsize=(8,4))
plt.plot(lmbda_list, llf_list)
plt.xlabel('Lambda value')
plt.ylabel('Log likelihood')
plt.title('Box-Cox Transformation Curve')
plt.show()

# Page 2
# Generate diagnostic plots before removing outliers
before = True # Change this variable to False to generate diagnostic plot after removing outliers
if before:
    model_residuals = lm.resid
    model_norm_residuals = lm.get_influence().resid_studentized_internal
    model_leverage = lm.get_influence().hat_matrix_diag
    model_cooks = lm.get_influence().cooks_distance[0]


    diagnostic_plot = plt.figure()
    plt.scatter(model_leverage, model_norm_residuals, alpha=0.5)
    sns.regplot(model_leverage, model_norm_residuals, scatter=False, ci=False, lowess=True,
                        line_kws={'color':'red', 'lw':1, 'alpha':0.8}, label='Cook\'s distance')
    diagnostic_plot.axes[0].set_xlim(0, max(model_leverage)+0.01)
    diagnostic_plot.axes[0].set_ylim(-3,5)
    plt.title('Residuals vs Leverage\nBefore removing outliers')
    plt.xlabel('Leverage')
    plt.ylabel('Standardized Residuals')
    plt.legend()

    leverage_top_3 = np.flip(np.argsort(model_cooks), 0)[:3]
    for i in leverage_top_3:
        diagnostic_plot.axes[0].annotate(i, xy=(model_leverage[i], model_norm_residuals[i]))
    plt.show()

# Boxcox transformation
y_boxcox, maxlog = stats.boxcox(y_remove)

# Influence plots after removing outliers and boxcox transformation
lm_remove = sm.OLS(y_boxcox, sm.add_constant(X_remove)).fit()
fig, ax = plt.subplots(figsize=(9,6))
fig = sm.graphics.influence_plot(lm_remove, alpha=0.001, ax=ax, criterion='cooks')
fig.show()

# Page 3
# Standardized residuals vs Fitted values without any transformation
sns.residplot(y_predicted, s)
plt.xlabel('Fitted values')
plt.ylabel('Standardized residuals')
plt.title('Without any transformation\nResiduals vs Fitted')
plt.show()
# Standardized residuals vs Fitted values removing all outliers and boxcox transformation
beta_hat_prime = np.dot(la.inv(np.dot(X_remove.T, X_remove)), np.dot(X_remove.T, y_remove))
y_predicted_prime = X_remove@beta_hat_prime # Fitted house price
sns.residplot(y_boxcox, np.delete(s, outliers_num))
plt.xlabel('Fitted values')
plt.ylabel('Standardized residuals')
plt.title('Removing all outliers and apply boxcox transformation\nResiduals vs Fitted')
plt.show()

# Page 4
# Fitted house price vs True house price
plt.scatter(y_predicted_prime, y_remove)
plt.plot(y_predicted_prime, LinearRegression().fit(y_predicted_prime, y_remove).predict(y_predicted_prime), 'r')
plt.xlabel('Fitted house price')
plt.ylabel('True hous price')
plt.title('After removing outliers and boxcox transformation')
plt.show()
```