extended_case_6

May 29, 2020

1 How should I price auto insurance in the United States?

1.1 Introduction

Business Context. The ability to price an insurance quote properly has a significant impact on insurers' management decisions and financial statements. You are the chief data scientist at a large insurance company and you are tasked to build an accurate predictive model to understand what factors affect the claim amount. Your findings will be used as a basis to make better management decisions about investments, new products and sales strategy, build trust and stability through accurate financial statements. Your goal is to use the data to predict the severity of insurance claims.

Business Problem. Your task is to **build a model to predict the cost of insurance from data using various characteristics of a policyholder.**

Analytical Context. The data resides in a CSV file which has been pre-cleaned for you and can directly be read in. Throughout the case, you will be iterating on your initial model many times based on common pitfalls that arise which we discussed in previous cases.

[1]:
```
### Load relevant packages

import pandas                    as pd
import numpy                     as np
import matplotlib.pyplot         as plt
import seaborn                   as sns
import statsmodels.api           as sm
import statsmodels.formula.api as smf
#import plotly.plotly             as py
import os

# This statement allow to display plot without asking to
%matplotlib inline
# always make it pretty
plt.style.use('ggplot')
#from pandas.plotting import scatter_matrix
```

1.2 Diving into the data

[2]:
```
DATA = pd.read_csv('ALLSTATEcost-cleaned.csv',
    dtype = { # indicate categorical variables
        "A": "category",
        "B": "category",
        "C": "category",
        "D": "category",
        "E": "category",
        "F": "category",
        "G": "category",
        "car_value": "category",
        "day": "category",
        "state": "category",
    }
)
```

The following are the columns in the dataset:

1. **day**: Day of the week (0-6, 0=Monday)
2. **state**: State where shopping point occurred
3. **group_size**: How many people will be covered under the policy (1, 2, 3 or 4)
4. **homeowner**: Whether the customer owns a home (0=no, 1=yes)
5. **car_age**: Age of the customer's car (How old the car is)
6. **car_value**: Value of the car when it was new
7. **risk_factor**: An ordinal assessment of how risky the customer is (0,1, 2, 3, 4)
8. **age_oldest**: Age of the oldest person in customer's group
9. **age_youngest**: Age of the youngest person in customer's group
10. **married_couple**: Does the customer group contain a married couple (0=no, 1=yes)
11. **C_previous**: What the customer formerly had or currently has for product option C (0=nothing, 1, 2, 3,4)
12. **duration_previous**: How long (in years) the customer was covered by their previous issuer
13. **A,B,C,D,E,F,G**: The coverage options:
14. **A**: Collision (levels: 0, 1, 2);
15. **B**: Towing (levels: 0, 1);
16. **C**: Bodily Injury (BI, levels: 1, 2, 3, 4);
17. **D**: Property Damage (PD, levels 1, 2, 3);
18. **E**: Rental Reimbursement (RR, levels: 0, 1);
19. **F**: Comprehensive (Comp, levels: 0, 1, 2, 3);
20. **G**: Medical/Personal Injury Protection (Med/PIP, levels: 1, 2, 3, 4)
21. **cost**: cost of the quoted coverage options

[3]:
```
DATA.head(10)
```

[3]:
```
  day state  group_size  homeowner  car_age car_value  risk_factor  \
0   1    OK           1          0        9         f          0.0
1   1    OK           1          0        9         f          0.0
2   4    PA           1          1        7         f          0.0
```

```
3   4    PA           1          1        7         f          0.0
4   3    AR           1          0        4         d          4.0
5   3    AR           1          0        4         d          4.0
6   3    AR           1          0        4         d          4.0
7   1    OK           1          0       13         f          3.0
8   1    OK           1          0       13         f          3.0
9   1    OK           1          0       13         f          3.0

   age_oldest  age_youngest  married_couple  C_previous  duration_previous  A  \
0          24            24               0         3.0                9.0  0
1          24            24               0         3.0                9.0  2
2          74            74               0         2.0               15.0  2
3          74            74               0         2.0               15.0  2
4          26            26               0         3.0                1.0  1
5          26            26               0         3.0                1.0  1
6          26            26               0         3.0                1.0  1
7          22            22               0         0.0                0.0  0
8          22            22               0         0.0                0.0  2
9          22            22               0         0.0                0.0  2

   B  C  D  E  F  G  cost
0  0  1  1  0  0  4   543
1  1  1  3  1  3  2   611
2  0  2  3  1  2  2   691
3  0  2  3  1  2  2   695
4  0  1  1  0  2  2   628
5  0  2  1  0  2  2   625
6  0  2  1  0  2  2   628
7  0  1  1  0  0  2   596
8  0  1  1  0  3  2   711
9  0  1  1  0  3  2   722
```

1.2.1 Exercise 1:

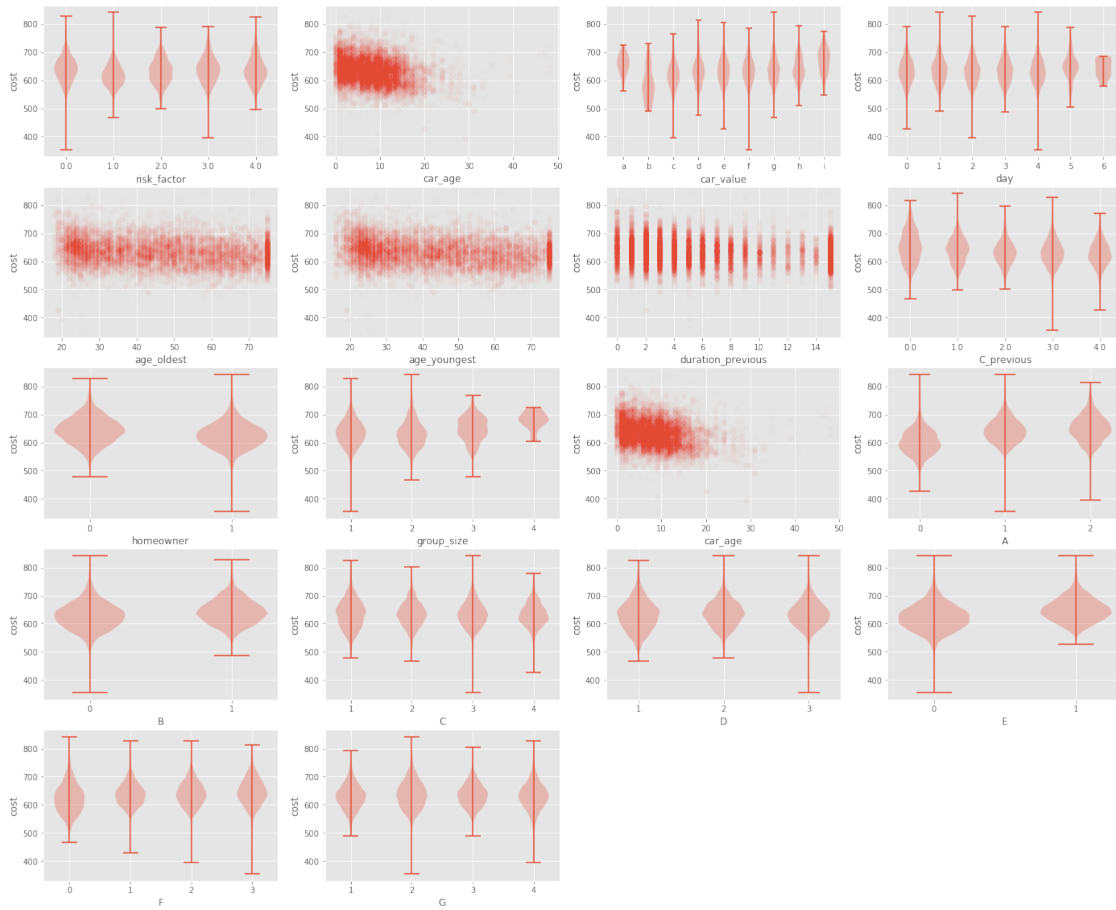
Write code to visualize the relationship between cost and the following variables. Choose your plots judiciously based on what you know about each variable. Different variable types (categorical vs. numerical) should have different types of plots (e.g. scatter, boxplot, violin plot, etc.) Group your plots together using the `plt.subplot()` function.

1. `car_age`
2. `age_oldest`
3. `age_youngest`
4. `duration_previous`
5. `C_previous`
6. `homeowner`
7. `group_size`
8. `car_age`

9. Categories A-G

Answer. One possible solution is shown below:

[4]:
```
plt.figure(figsize=(24,20))
varstolook = ["risk_factor","car_age", "car_value", "day", "age_oldest",␣
↪"age_youngest" ,
              "duration_previous","C_previous","homeowner","group_size",
              "car_age", "A", "B", "C", "D", "E", "F", "G"
              ]
for i,feature in enumerate(varstolook):
    plt.subplot(5,4,i+1)
    colvalues = DATA[feature]
    unique = sorted(set(colvalues.dropna().values))
    if len(unique) < 10:
        # categorical: let's make a violin plot
        plt.violinplot([DATA.cost.values[colvalues == level] for level in␣
↪unique],
                       positions=range(len(unique)))
        plt.xticks(range(len(unique)), labels=unique)
    else:
        plt.scatter(colvalues.values, DATA.cost.values, alpha=0.01,␣
↪edgecolor=None)
    plt.xlabel(feature)
    plt.ylabel('cost')
```



1.2.2 Exercise 2:

Convert all categorical data to be in the one-hot encoding format.

Answer. We do the following:

[5]:
```
DATA_onehot = pd.get_dummies(DATA, columns=['state','car_value',
                                   'A','B','C', 'D','E','F','G'],
                                   )
```

[6]:
```
DATA_onehot.head(10)
```

[6]:
```
   day  group_size  homeowner  car_age  risk_factor  age_oldest  age_youngest  \
0    1           1          0        9          0.0          24            24
1    1           1          0        9          0.0          24            24
2    4           1          1        7          0.0          74            74
3    4           1          1        7          0.0          74            74
4    3           1          0        4          4.0          26            26
```

```
5  3  1  0   4  4.0  26  26
6  3  1  0   4  4.0  26  26
7  1  1  0  13  3.0  22  22
8  1  1  0  13  3.0  22  22
9  1  1  0  13  3.0  22  22

   married_couple  C_previous  duration_previous  …  E_0  E_1  F_0  F_1  \
0               0         3.0                9.0  …    1    0    1    0
1               0         3.0                9.0  …    0    1    0    0
2               0         2.0               15.0  …    0    1    0    0
3               0         2.0               15.0  …    0    1    0    0
4               0         3.0                1.0  …    1    0    0    0
5               0         3.0                1.0  …    1    0    0    0
6               0         3.0                1.0  …    1    0    0    0
7               0         0.0                0.0  …    1    0    1    0
8               0         0.0                0.0  …    1    0    0    0
9               0         0.0                0.0  …    1    0    0    0

   F_2  F_3  G_1  G_2  G_3  G_4
0    0    0    0    0    0    1
1    0    1    0    1    0    0
2    1    0    0    1    0    0
3    1    0    0    1    0    0
4    1    0    0    1    0    0
5    1    0    0    1    0    0
6    1    0    0    1    0    0
7    0    0    0    1    0    0
8    0    1    0    1    0    0
9    0    1    0    1    0    0

[10 rows x 78 columns]
```

1.3 Fitting a multiple linear regression

1.3.1 Exercise 3:

Split your data into training and testing sets (an 80-20 split is a good starting point).

Note: Keep random seed as 2019 in the code cell

Answer. One possible solution is given below:

[7]:
```
np.random.seed(2019) # a seed makes the analysis reproducible
ndata = len(DATA_onehot)
idx_train = np.random.choice(range(ndata),int(0.8*ndata),replace=False)
idx_test  = np.asarray(list(set(range(ndata)) - set(idx_train)))
train     = DATA.loc[idx_train]
```

```
test      = DATA.loc[idx_test]
```

1.3.2 Exercise 4:

4.1 Fit a multiple linear regression model to the training data regressing cost against all the other variables. Call this `model1`. What is the AIC value?

Answer. One possible solution is given below:

[8]:
```
reg_formula = "cost ~ " + " + ".join(col for col in DATA.columns if col !=␣
↪'cost')
print(reg_formula)
```

```
cost ~ day + state + group_size + homeowner + car_age + car_value + risk_factor
+ age_oldest + age_youngest + married_couple + C_previous + duration_previous +
A + B + C + D + E + F + G
```

[9]:
```
model1 = smf.ols(formula = reg_formula, data = train).fit()
print(model1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   cost   R-squared:                       0.433
Model:                            OLS   Adj. R-squared:                  0.430
Method:                 Least Squares   F-statistic:                     128.5
Date:                Sat, 16 Nov 2019   Prob (F-statistic):               0.00
Time:                        11:38:49   Log-Likelihood:                -61631.
No. Observations:               12352   AIC:                         1.234e+05
Df Residuals:                   12278   BIC:                         1.240e+05
Df Model:                          73
Covariance Type:            nonrobust
================================================================================
=====
                        coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
Intercept           682.1414      9.139     74.644      0.000     664.228
700.055
day[T.1]              1.8716      0.966      1.938      0.053      -0.022
3.765
day[T.2]             -1.6922      1.003     -1.687      0.092      -3.658
0.274
day[T.3]              0.3649      1.022      0.357      0.721      -1.638
2.367
day[T.4]             -2.1277      1.045     -2.037      0.042      -4.175
-0.080
```

day[T.5]	3.5332	2.745	1.287	0.198	-1.847	8.913
day[T.6]	-2.9007	8.712	-0.333	0.739	-19.977	14.175
state[T.AR]	2.0422	3.126	0.653	0.514	-4.086	8.170
state[T.CO]	-7.6047	2.557	-2.974	0.003	-12.617	-2.592
state[T.CT]	30.9584	2.880	10.748	0.000	25.312	36.604
state[T.DC]	41.9676	4.959	8.464	0.000	32.248	51.687
state[T.DE]	38.1680	4.667	8.178	0.000	29.020	47.316
state[T.FL]	12.2307	2.174	5.626	0.000	7.969	16.492
state[T.GA]	9.8604	2.404	4.101	0.000	5.148	14.573
state[T.IA]	-47.1622	3.506	-13.453	0.000	-54.034	-40.291
state[T.ID]	-19.7335	4.150	-4.755	0.000	-27.867	-11.600
state[T.IN]	-9.8334	2.571	-3.824	0.000	-14.873	-4.793
state[T.KS]	-3.3598	4.451	-0.755	0.450	-12.084	5.365
state[T.KY]	20.5117	2.900	7.073	0.000	14.828	26.196
state[T.MD]	24.6093	2.487	9.893	0.000	19.733	29.485
state[T.ME]	-33.2810	4.044	-8.229	0.000	-41.209	-25.353
state[T.MO]	-20.8915	2.944	-7.096	0.000	-26.663	-15.120
state[T.MS]	0.8295	3.297	0.252	0.801	-5.632	7.291
state[T.MT]	-11.2580	5.782	-1.947	0.052	-22.592	0.076
state[T.ND]	10.3985	6.080	1.710	0.087	-1.520	22.317
state[T.NE]	-9.9600	5.359	-1.858	0.063	-20.465	0.545
state[T.NH]	-19.2138	3.930	-4.889	0.000	-26.918	-11.510
state[T.NM]	-1.0168	3.874	-0.262	0.793	-8.611	6.578
state[T.NV]	22.9524	2.823	8.129	0.000	17.418	28.487

state[T.NY]	40.0363	2.416	16.573	0.000	35.301	44.772
state[T.OH]	-7.8976	2.317	-3.409	0.001	-12.438	-3.357
state[T.OK]	-12.1464	2.770	-4.386	0.000	-17.575	-6.718
state[T.OR]	-7.7234	2.874	-2.687	0.007	-13.358	-2.089
state[T.PA]	10.0179	2.213	4.528	0.000	5.681	14.355
state[T.RI]	25.5978	4.322	5.922	0.000	17.125	34.070
state[T.SD]	-17.8559	12.770	-1.398	0.162	-42.887	7.175
state[T.TN]	-10.4169	2.621	-3.975	0.000	-15.554	-5.280
state[T.UT]	-16.0961	2.842	-5.664	0.000	-21.667	-10.525
state[T.WA]	5.5185	2.601	2.122	0.034	0.420	10.617
state[T.WI]	-31.0784	2.991	-10.389	0.000	-36.942	-25.215
state[T.WV]	25.5210	4.309	5.923	0.000	17.074	33.967
state[T.WY]	-3.3079	6.794	-0.487	0.626	-16.626	10.010
car_value[T.b]	-65.6866	10.467	-6.276	0.000	-86.203	-45.171
car_value[T.c]	-52.4144	8.765	-5.980	0.000	-69.594	-35.234
car_value[T.d]	-44.8674	8.682	-5.168	0.000	-61.884	-27.850
car_value[T.e]	-45.3580	8.663	-5.236	0.000	-62.338	-28.378
car_value[T.f]	-45.3271	8.674	-5.225	0.000	-62.330	-28.324
car_value[T.g]	-41.6584	8.700	-4.788	0.000	-58.712	-24.605
car_value[T.h]	-32.6790	8.798	-3.714	0.000	-49.924	-15.434
car_value[T.i]	-10.7050	9.688	-1.105	0.269	-29.695	8.285
A[T.1]	26.9559	1.543	17.474	0.000	23.932	29.980
A[T.2]	32.3753	1.845	17.549	0.000	28.759	35.991
B[T.1]	2.4547	0.792	3.101	0.002	0.903	4.006

```
C[T.2]                 1.1954      1.127      1.060      0.289      -1.014
3.405
C[T.3]                 1.2361      1.166      1.060      0.289      -1.050
3.523
C[T.4]                 3.9454      1.753      2.251      0.024       0.509
7.382
D[T.2]                -2.3908      1.208     -1.979      0.048      -4.759
-0.023
D[T.3]                -2.1810      1.256     -1.736      0.083      -4.644
0.282
E[T.1]                 7.9994      0.871      9.181      0.000       6.291
9.707
F[T.1]                18.0952      1.682     10.759      0.000      14.798
21.392
F[T.2]                16.2991      1.617     10.078      0.000      13.129
19.469
F[T.3]                11.9351      2.344      5.091      0.000       7.340
16.531
G[T.2]                 7.9951      0.944      8.473      0.000       6.145
9.845
G[T.3]                 1.1834      1.192      0.993      0.321      -1.153
3.519
G[T.4]                 4.2016      1.263      3.327      0.001       1.726
6.677
group_size             3.4578      1.476      2.343      0.019       0.565
6.351
homeowner            -14.1118      0.734    -19.222      0.000     -15.551
-12.673
car_age               -0.7381      0.068    -10.793      0.000      -0.872
-0.604
risk_factor           -0.6134      0.231     -2.659      0.008      -1.066
-0.161
age_oldest             0.5071      0.064      7.979      0.000       0.383
0.632
age_youngest          -0.9177      0.062    -14.767      0.000      -1.039
-0.796
married_couple       -10.2573      1.393     -7.361      0.000     -12.989
-7.526
C_previous            -6.0917      0.361    -16.858      0.000      -6.800
-5.383
duration_previous     -1.4467      0.073    -19.699      0.000      -1.591
-1.303
==============================================================================
Omnibus:                      567.517   Durbin-Watson:                   1.965
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1406.975
Skew:                           0.260   Prob(JB):                    3.01e-306
Kurtosis:                       4.569   Cond. No.                     5.53e+03
==============================================================================
```

```
Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 5.53e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[10]:
```python
print(model1.aic)
```

```
123410.41443632482
```

4.2 According to `model1`, which states are most and least expensive?

Answer. DC and New York are the expensive states, while Indiana (IA) seems to be the least expensive.

4.3 Interpret the coefficients of `group_size`, `homeowner`, `car_age`, `risk_factor`, `age_oldest`, `age_youngest` `married_couple` , `duration_previous`. Do the signs and values of these coefficients make sense to you in the context of this business problem?

Answer. Being an owner reduces your insurance cost by about $14 on average, assuming everything else is held the same. `car_age` has a negative coefficient; this makes sense since older cars should pay less for insurance as they are worth less. Being a married couple also reduces the insurance rate. This makes sense since couples could lead to a more financially stable household.

`duration_previous` has a negative coefficient. This makes sense as it says if a customer was insured by a previous company for a long time, it signals that the customer is not a risky driver. If the customer goes from one insurance company to other in a short amount of time, then it should be worrisome because they could be risky drivers that are constantly being punted off by various insurance companies. Finally, the coefficients of `age_oldest` and `age_youngest` are nearly zero; this makes sense since the probability and magnitude of car damage is unlikely to be monotonically related to the ages of the children.

1.3.3 Exercise 5:

Which variables from `model1` are statistically significant? (For categorical variables, consider them to be significant if at least one of their categories are statistically significant). Refit the model using only these variables; call this `model2`. How does this model compare to the previous model?

Answer. Because we are throwing so many variables at the model, let's apply the Bonferroni correction to tighten the threshold on which to consider a variable as significant. There are a total of 73 degrees of freedom, so let's set the threshold at 0.05/73:

[11]:
```python
model1.pvalues[model1.pvalues<0.05/73]
```

[11]:
```
Intercept           0.000000e+00
state[T.CT]         7.980229e-27
state[T.DC]         2.880883e-17
```

```
state[T.DE]           3.169722e-16
state[T.FL]           1.886398e-08
state[T.GA]           4.131790e-05
state[T.IA]           5.755818e-41
state[T.ID]           2.002456e-06
state[T.IN]           1.317464e-04
state[T.KY]           1.594148e-12
state[T.MD]           5.423620e-23
state[T.ME]           2.082377e-16
state[T.MO]           1.356505e-12
state[T.NH]           1.028681e-06
state[T.NV]           4.729177e-16
state[T.NY]           5.025715e-61
state[T.OH]           6.536454e-04
state[T.OK]           1.165811e-05
state[T.PA]           6.024151e-06
state[T.RI]           3.262302e-09
state[T.TN]           7.078938e-05
state[T.UT]           1.516045e-08
state[T.WI]           3.538870e-25
state[T.WV]           3.254093e-09
car_value[T.b]        3.593542e-10
car_value[T.c]        2.289835e-09
car_value[T.d]        2.401202e-07
car_value[T.e]        1.667639e-07
car_value[T.f]        1.765579e-07
car_value[T.g]        1.701257e-06
car_value[T.h]        2.045318e-04
A[T.1]                1.489241e-67
A[T.2]                4.095847e-68
E[T.1]                4.971007e-20
F[T.1]                7.105514e-27
F[T.2]                8.531378e-24
F[T.3]                3.618296e-07
G[T.2]                2.664304e-17
homeowner             3.778461e-81
car_age               4.926962e-27
age_oldest            1.611138e-15
age_youngest          6.318400e-49
married_couple        1.941193e-13
C_previous            4.677549e-63
duration_previous     4.467600e-85
dtype: float64
```

[12]:

```
form = "cost ~ state + car_value + A + E + F + G + homeowner + car_age +␣
↪age_oldest + age_youngest + married_couple + C_previous + duration_previous"
model2 = smf.ols(formula = form, data = train).fit()
```

```python
print(model2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   cost   R-squared:                       0.431
Model:                            OLS   Adj. R-squared:                  0.428
Method:                 Least Squares   F-statistic:                     157.6
Date:                Sat, 16 Nov 2019   Prob (F-statistic):               0.00
Time:                        11:38:49   Log-Likelihood:                -61657.
No. Observations:               12352   AIC:                         1.234e+05
Df Residuals:                   12292   BIC:                         1.239e+05
Df Model:                          59
Covariance Type:            nonrobust
================================================================================
=====
                        coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
Intercept           685.5323      8.997     76.198      0.000     667.897
703.167
state[T.AR]           2.2351      3.116      0.717      0.473      -3.873
8.343
state[T.CO]          -8.2373      2.541     -3.242      0.001     -13.217
-3.257
state[T.CT]          29.4072      2.833     10.380      0.000      23.854
34.960
state[T.DC]          41.3386      4.957      8.339      0.000      31.622
51.055
state[T.DE]          36.6629      4.641      7.899      0.000      27.565
45.761
state[T.FL]          10.8659      2.140      5.078      0.000       6.671
15.060
state[T.GA]           9.2314      2.373      3.890      0.000       4.580
13.883
state[T.IA]         -47.7198      3.477    -13.722      0.000     -54.536
-40.903
state[T.ID]         -20.1102      4.129     -4.871      0.000     -28.204
-12.017
state[T.IN]         -11.5457      2.517     -4.587      0.000     -16.480
-6.612
state[T.KS]          -4.3616      4.438     -0.983      0.326     -13.061
4.337
state[T.KY]          19.2030      2.883      6.660      0.000      13.551
24.855
state[T.MD]          23.0698      2.450      9.415      0.000      18.267
27.873
```

state[T.ME]	-34.6074	4.014	-8.621	0.000	-42.476	-26.739
state[T.MO]	-21.2660	2.933	-7.250	0.000	-27.016	-15.516
state[T.MS]	0.7301	3.280	0.223	0.824	-5.700	7.160
state[T.MT]	-11.5413	5.779	-1.997	0.046	-22.870	-0.213
state[T.ND]	8.8786	6.045	1.469	0.142	-2.971	20.728
state[T.NE]	-10.4010	5.347	-1.945	0.052	-20.882	0.080
state[T.NH]	-20.8137	3.906	-5.329	0.000	-28.469	-13.158
state[T.NM]	-1.4803	3.859	-0.384	0.701	-9.044	6.083
state[T.NV]	22.5853	2.799	8.070	0.000	17.099	28.071
state[T.NY]	38.7262	2.377	16.289	0.000	34.066	43.386
state[T.OH]	-8.6244	2.294	-3.759	0.000	-13.122	-4.127
state[T.OK]	-13.0155	2.752	-4.730	0.000	-18.409	-7.622
state[T.OR]	-8.3614	2.867	-2.917	0.004	-13.981	-2.742
state[T.PA]	8.9045	2.187	4.071	0.000	4.617	13.192
state[T.RI]	24.3839	4.311	5.656	0.000	15.934	32.834
state[T.SD]	-18.6269	12.773	-1.458	0.145	-43.665	6.411
state[T.TN]	-10.6129	2.621	-4.050	0.000	-15.750	-5.476
state[T.UT]	-16.5092	2.834	-5.825	0.000	-22.064	-10.954
state[T.WA]	4.7061	2.594	1.814	0.070	-0.379	9.792
state[T.WI]	-31.9879	2.966	-10.787	0.000	-37.801	-26.175
state[T.WV]	25.7597	4.302	5.987	0.000	17.326	34.193
state[T.WY]	-3.0965	6.787	-0.456	0.648	-16.399	10.206
car_value[T.b]	-66.5523	10.477	-6.352	0.000	-87.089	-46.015
car_value[T.c]	-53.2001	8.770	-6.066	0.000	-70.391	-36.009

```
car_value[T.d]      -45.7371      8.687     -5.265      0.000     -62.765
-28.709
car_value[T.e]      -46.1576      8.669     -5.324      0.000     -63.151
-29.164
car_value[T.f]      -46.0428      8.681     -5.304      0.000     -63.059
-29.027
car_value[T.g]      -42.5199      8.707     -4.884      0.000     -59.587
-25.453
car_value[T.h]      -33.4730      8.805     -3.801      0.000     -50.733
-16.213
car_value[T.i]      -11.1000      9.688     -1.146      0.252     -30.090
7.890
A[T.1]               26.8499      1.537     17.471      0.000      23.837
29.862
A[T.2]               32.0598      1.842     17.401      0.000      28.448
35.671
E[T.1]                9.1686      0.767     11.961      0.000       7.666
10.671
F[T.1]               18.2903      1.674     10.926      0.000      15.009
21.572
F[T.2]               16.3810      1.612     10.160      0.000      13.221
19.541
F[T.3]               12.3990      2.340      5.299      0.000       7.813
16.985
G[T.2]                8.0299      0.943      8.514      0.000       6.181
9.879
G[T.3]                1.1820      1.189      0.994      0.320      -1.148
3.512
G[T.4]                4.4388      1.258      3.528      0.000       1.973
6.905
homeowner           -14.2108      0.731    -19.449      0.000     -15.643
-12.779
car_age              -0.7368      0.068    -10.809      0.000      -0.870
-0.603
age_oldest            0.6064      0.050     12.118      0.000       0.508
0.704
age_youngest         -1.0133      0.048    -21.155      0.000      -1.107
-0.919
married_couple       -7.5494      0.851     -8.873      0.000      -9.217
-5.882
C_previous           -5.8768      0.308    -19.072      0.000      -6.481
-5.273
duration_previous    -1.4764      0.073    -20.191      0.000      -1.620
-1.333
==============================================================================
Omnibus:                      557.799   Durbin-Watson:                   1.964
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1369.483
Skew:                           0.258   Prob(JB):                    4.17e-298
```

```
Kurtosis:                       4.548   Cond. No.                     5.53e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 5.53e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[13]:
```
print(model2.aic)
```

```
123434.75323787131
```

We've drastically reduced the number of total variables while keeping the AIC about the same.

1.3.4 Exercise 6:

In addition to the variables in `model2`, add terms for:

1. square of `age_youngest`
2. square term for the age of the car
3. interaction term for `car_value` and `age_youngest`

Answer. One possible solution is below:

[14]:
```
model3 = smf.ols(formula = "cost ~ state + car_value + A + E + F + G +␣
 ↪homeowner + car_age + age_oldest + age_youngest + married_couple +␣
 ↪C_previous + duration_previous + I(age_youngest**2) + I(car_age**2) +␣
 ↪car_value*age_youngest", data = train).fit()
print(model3.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   cost   R-squared:                       0.445
Model:                            OLS   Adj. R-squared:                  0.442
Method:                 Least Squares   F-statistic:                     143.0
Date:                Sat, 16 Nov 2019   Prob (F-statistic):               0.00
Time:                        11:38:49   Log-Likelihood:                -61496.
No. Observations:               12352   AIC:                         1.231e+05
Df Residuals:                   12282   BIC:                         1.237e+05
Df Model:                          69
Covariance Type:            nonrobust
================================================================================
===============
                                  coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
---------------
Intercept                     678.3106     25.889     26.200      0.000
```

```
627.564     729.058
state[T.AR]                          2.2460      3.087      0.728      0.467
-3.805       8.297
state[T.CO]                         -6.5116      2.516     -2.588      0.010
-11.444      -1.579
state[T.CT]                         31.4596      2.808     11.205      0.000
25.956      36.963
state[T.DC]                         43.4524      4.904      8.861      0.000
33.840      53.064
state[T.DE]                         36.1320      4.601      7.853      0.000
27.113      45.151
state[T.FL]                          9.9821      2.121      4.705      0.000
5.824      14.140
state[T.GA]                          9.7198      2.352      4.133      0.000
5.110      14.329
state[T.IA]                        -46.7699      3.439    -13.598      0.000
-53.512     -40.028
state[T.ID]                        -20.3820      4.084     -4.991      0.000
-28.388     -12.377
state[T.IN]                        -11.5683      2.495     -4.637      0.000
-16.459      -6.678
state[T.KS]                         -4.9967      4.387     -1.139      0.255
-13.596       3.603
state[T.KY]                         20.8712      2.856      7.308      0.000
15.273      26.469
state[T.MD]                         24.2324      2.432      9.965      0.000
19.466      28.999
state[T.ME]                        -33.3123      3.977     -8.377      0.000
-41.107     -25.517
state[T.MO]                        -22.0774      2.906     -7.597      0.000
-27.774     -16.381
state[T.MS]                          1.5572      3.249      0.479      0.632
-4.811       7.926
state[T.MT]                        -14.1164      5.713     -2.471      0.013
-25.314      -2.919
state[T.ND]                          9.0805      5.985      1.517      0.129
-2.651      20.812
state[T.NE]                        -10.7034      5.287     -2.025      0.043
-21.066      -0.341
state[T.NH]                        -21.1769      3.868     -5.475      0.000
-28.758     -13.595
state[T.NM]                         -1.7545      3.820     -0.459      0.646
-9.243       5.734
state[T.NV]                         23.9633      2.777      8.629      0.000
18.520      29.407
state[T.NY]                         40.0842      2.358     16.998      0.000
35.462      44.707
state[T.OH]                         -8.6003      2.272     -3.786      0.000
```

```
-13.053      -4.148
state[T.OK]                        -13.7880      2.727     -5.056      0.000
-19.133      -8.443
state[T.OR]                         -7.9841      2.837     -2.815      0.005
-13.544      -2.424
state[T.PA]                          9.9366      2.169      4.581      0.000
5.685      14.188
state[T.RI]                         22.8542      4.266      5.357      0.000
14.492      31.217
state[T.SD]                        -12.0619     12.619     -0.956      0.339
-36.798      12.674
state[T.TN]                        -10.4689      2.596     -4.033      0.000
-15.557      -5.381
state[T.UT]                        -17.6919      2.809     -6.298      0.000
-23.198     -12.185
state[T.WA]                          5.2644      2.576      2.043      0.041
0.214      10.315
state[T.WI]                        -29.7522      2.936    -10.134      0.000
-35.507     -23.997
state[T.WV]                         27.8902      4.253      6.557      0.000
19.553      36.227
state[T.WY]                         -2.2236      6.708     -0.331      0.740
-15.372      10.925
car_value[T.b]                     -42.9709     30.095     -1.428      0.153
-101.961      16.019
car_value[T.c]                      -1.5730     25.876     -0.061      0.952
-52.294      49.148
car_value[T.d]                      -1.8465     25.804     -0.072      0.943
-52.426      48.733
car_value[T.e]                       3.0833     25.764      0.120      0.905
-47.417      53.584
car_value[T.f]                       3.8171     25.784      0.148      0.882
-46.724      54.358
car_value[T.g]                      11.2694     25.870      0.436      0.663
-39.440      61.979
car_value[T.h]                      23.9304     26.256      0.911      0.362
-27.536      75.397
car_value[T.i]                      75.3890     31.258      2.412      0.016
14.119     136.659
A[T.1]                              26.7680      1.519     17.627      0.000
23.791      29.745
A[T.2]                              31.6278      1.821     17.371      0.000
28.059      35.197
E[T.1]                               9.7030      0.759     12.790      0.000
8.216      11.190
F[T.1]                              18.0620      1.654     10.918      0.000
14.819      21.305
F[T.2]                              16.6697      1.593     10.463      0.000
```

```
13.547      19.793
F[T.3]                          13.4206      2.311      5.806      0.000
8.890      17.951
G[T.2]                           7.8860      0.933      8.452      0.000
6.057       9.715
G[T.3]                           1.3493      1.176      1.147      0.251
-0.956       3.655
G[T.4]                           4.7528      1.244      3.821      0.000
2.315       7.191
homeowner                      -13.3141      0.725    -18.356      0.000
-14.736     -11.892
car_age                         -1.2582      0.159     -7.892      0.000
-1.571      -0.946
age_oldest                       0.4862      0.050      9.729      0.000
0.388       0.584
age_youngest                    -2.4447      0.436     -5.609      0.000
-3.299      -1.590
car_value[T.b]:age_youngest    -0.0795      0.552     -0.144      0.886
-1.162       1.003
car_value[T.c]:age_youngest    -0.5539      0.430     -1.287      0.198
-1.398       0.290
car_value[T.d]:age_youngest    -0.3436      0.421     -0.815      0.415
-1.169       0.482
car_value[T.e]:age_youngest    -0.4586      0.420     -1.092      0.275
-1.282       0.365
car_value[T.f]:age_youngest    -0.4586      0.420     -1.091      0.275
-1.283       0.366
car_value[T.g]:age_youngest    -0.5369      0.422     -1.271      0.204
-1.365       0.291
car_value[T.h]:age_youngest    -0.6037      0.430     -1.402      0.161
-1.447       0.240
car_value[T.i]:age_youngest    -1.1272      0.545     -2.069      0.039
-2.195      -0.059
married_couple                  -7.1073      0.842     -8.445      0.000
-8.757      -5.458
C_previous                      -6.5723      0.308    -21.336      0.000
-7.176      -5.968
duration_previous               -1.4717      0.072    -20.343      0.000
-1.614      -1.330
I(age_youngest ** 2)             0.0216      0.001     17.079      0.000
0.019       0.024
I(car_age ** 2)                  0.0265      0.007      3.642      0.000
0.012       0.041
==============================================================================
Omnibus:                      613.777   Durbin-Watson:                   1.970
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1768.408
Skew:                           0.230   Prob(JB):                         0.00
Kurtosis:                       4.796   Cond. No.                     6.66e+05
```

```
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.66e+05. This might indicate that there are strong multicollinearity or other numerical problems.
```

1.4 Feature selection

To reduce the number of features, it can often be helpful to aggregate the categories; for example, we can create a new variable by assigning each state to a larger region:

[15]:
```
state_regions = pd.read_csv('https://raw.githubusercontent.com/cphalpert/
↪census-regions/master/us%20census%20bureau%20regions%20and%20divisions.csv')
# should download the above file
state_regions
```

[15]:

	State	State Code	Region	Division
0	Alaska	AK	West	Pacific
1	Alabama	AL	South	East South Central
2	Arkansas	AR	South	West South Central
3	Arizona	AZ	West	Mountain
4	California	CA	West	Pacific
5	Colorado	CO	West	Mountain
6	Connecticut	CT	Northeast	New England
7	District of Columbia	DC	South	South Atlantic
8	Delaware	DE	South	South Atlantic
9	Florida	FL	South	South Atlantic
10	Georgia	GA	South	South Atlantic
11	Hawaii	HI	West	Pacific
12	Iowa	IA	Midwest	West North Central
13	Idaho	ID	West	Mountain
14	Illinois	IL	Midwest	East North Central
15	Indiana	IN	Midwest	East North Central
16	Kansas	KS	Midwest	West North Central
17	Kentucky	KY	South	East South Central
18	Louisiana	LA	South	West South Central
19	Massachusetts	MA	Northeast	New England
20	Maryland	MD	South	South Atlantic
21	Maine	ME	Northeast	New England
22	Michigan	MI	Midwest	East North Central
23	Minnesota	MN	Midwest	West North Central
24	Missouri	MO	Midwest	West North Central
25	Mississippi	MS	South	East South Central
26	Montana	MT	West	Mountain
27	North Carolina	NC	South	South Atlantic

```
28          North Dakota          ND     Midwest  West North Central
29              Nebraska          NE     Midwest  West North Central
30         New Hampshire          NH   Northeast         New England
31            New Jersey          NJ   Northeast     Middle Atlantic
32            New Mexico          NM        West            Mountain
33                Nevada          NV        West            Mountain
34              New York          NY   Northeast     Middle Atlantic
35                  Ohio          OH     Midwest  East North Central
36              Oklahoma          OK       South  West South Central
37                Oregon          OR        West             Pacific
38          Pennsylvania          PA   Northeast     Middle Atlantic
39          Rhode Island          RI   Northeast         New England
40        South Carolina          SC       South      South Atlantic
41          South Dakota          SD     Midwest  West North Central
42             Tennessee          TN       South  East South Central
43                 Texas          TX       South  West South Central
44                  Utah          UT        West            Mountain
45              Virginia          VA       South      South Atlantic
46               Vermont          VT   Northeast         New England
47            Washington          WA        West             Pacific
48             Wisconsin          WI     Midwest  East North Central
49         West Virginia          WV       South      South Atlantic
50               Wyoming          WY        West            Mountain
```

1.4.1 Exercise 7:

7.1 Create a new column where a state is replaced with the region it is in according to the above table.

Answer. One possible solution is shown below:

[16]:
```
area = dict(zip(state_regions["State Code"], state_regions["Region"]))
DATA_region = DATA.copy()
DATA_region["region"] = DATA.state.map(area).astype("category")
DATA_region.drop(columns="state", inplace=True)
DATA_region.head()
```

[16]:
```
   day  group_size  homeowner  car_age car_value  risk_factor  age_oldest  \
0    1           1          0        9         f          0.0          24
1    1           1          0        9         f          0.0          24
2    4           1          1        7         f          0.0          74
3    4           1          1        7         f          0.0          74
4    3           1          0        4         d          4.0          26

   age_youngest  married_couple  C_previous  duration_previous  A  B  C  D  E  \
0            24               0         3.0                9.0  0  0  1  1  0
1            24               0         3.0                9.0  2  1  1  3  1
```

```
2              74                0          2.0                 15.0  2  0  2  3  1
3              74                0          2.0                 15.0  2  0  2  3  1
4              26                0          3.0                  1.0  1  0  1  1  0

   F  G  cost     region
0  0  4   543      South
1  3  2   611      South
2  2  2   691  Northeast
3  2  2   695  Northeast
4  2  2   628      South
```

7.2 Fit the model as in `model13` but this time use `region` instead of `state`. Call this `model4`.

Answer. One possible solution is shown below:

[17]:
```
# fit the model with `region` instead of `state`
train      = DATA_region.loc[idx_train]
test       = DATA_region.loc[idx_test]
model4 = smf.ols(formula = "cost ~ region + car_value + A + E + F + G +␣
 ↪homeowner + car_age + age_oldest + age_youngest + married_couple +␣
 ↪C_previous + duration_previous + I(age_youngest**2) + I(car_age**2) +␣
 ↪car_value*age_youngest", data = train).fit()
print(model4.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   cost   R-squared:                       0.373
Model:                            OLS   Adj. R-squared:                  0.371
Method:                 Least Squares   F-statistic:                     197.6
Date:                Sat, 16 Nov 2019   Prob (F-statistic):               0.00
Time:                        11:38:49   Log-Likelihood:                -62258.
No. Observations:               12352   AIC:                         1.246e+05
Df Residuals:                   12314   BIC:                         1.249e+05
Df Model:                          37
Covariance Type:            nonrobust
================================================================================
===============
                                  coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
---------------
Intercept                     685.3350     27.268     25.133      0.000
631.885     738.785
region[T.Northeast]            31.8027      1.222     26.030      0.000
29.408      34.198
region[T.South]                23.0226      1.014     22.707      0.000
21.035      25.010
```

```
region[T.West]                      13.9623      1.188     11.751      0.000
11.633      16.291
car_value[T.b]                     -67.1335     31.752     -2.114      0.035
-129.372      -4.895
car_value[T.c]                     -27.0335     27.370     -0.988      0.323
-80.683      26.616
car_value[T.d]                     -30.1911     27.290     -1.106      0.269
-83.684      23.301
car_value[T.e]                     -22.3340     27.253     -0.820      0.413
-75.753      31.085
car_value[T.f]                     -20.6645     27.275     -0.758      0.449
-74.127      32.798
car_value[T.g]                     -13.9859     27.363     -0.511      0.609
-67.622      39.651
car_value[T.h]                       1.9292     27.760      0.069      0.945
-52.484      56.342
car_value[T.i]                      49.7230     33.086      1.503      0.133
-15.131     114.577
A[T.1]                              39.3006      1.386     28.348      0.000
36.583      42.018
A[T.2]                              44.8901      1.723     26.052      0.000
41.513      48.268
E[T.1]                              12.4640      0.793     15.714      0.000
10.909      14.019
F[T.1]                              -3.2250      1.315     -2.452      0.014
-5.804      -0.646
F[T.2]                              -3.1728      1.281     -2.477      0.013
-5.684      -0.662
F[T.3]                              -7.0299      2.184     -3.219      0.001
-11.311      -2.749
G[T.2]                              10.3528      0.930     11.127      0.000
8.529      12.177
G[T.3]                               6.9446      1.055      6.584      0.000
4.877       9.012
G[T.4]                               7.8538      1.262      6.222      0.000
5.380      10.328
homeowner                          -13.8865      0.761    -18.257      0.000
-15.377     -12.396
car_age                             -1.3622      0.168     -8.103      0.000
-1.692      -1.033
age_oldest                           0.5019      0.053      9.512      0.000
0.398       0.605
age_youngest                        -2.6340      0.461     -5.711      0.000
-3.538      -1.730
car_value[T.b]:age_youngest          0.3397      0.583      0.583      0.560
-0.802       1.482
car_value[T.c]:age_youngest         -0.1018      0.455     -0.223      0.823
-0.994       0.791
```

```
car_value[T.d]:age_youngest      0.1606      0.446      0.360      0.719
-0.713       1.034
car_value[T.e]:age_youngest     -0.0181      0.444     -0.041      0.967
-0.889       0.853
car_value[T.f]:age_youngest     -0.0297      0.445     -0.067      0.947
-0.901       0.842
car_value[T.g]:age_youngest     -0.1016      0.447     -0.227      0.820
-0.977       0.774
car_value[T.h]:age_youngest     -0.2470      0.455     -0.543      0.587
-1.139       0.645
car_value[T.i]:age_youngest     -0.6569      0.577     -1.139      0.255
-1.788       0.474
married_couple                  -7.5115      0.889     -8.451      0.000
-9.254      -5.769
C_previous                      -6.6627      0.321    -20.749      0.000
-7.292      -6.033
duration_previous               -1.4101      0.076    -18.508      0.000
-1.559      -1.261
I(age_youngest ** 2)             0.0189      0.001     14.247      0.000
0.016       0.022
I(car_age ** 2)                  0.0321      0.008      4.175      0.000
0.017       0.047
==============================================================================
Omnibus:                      452.830   Durbin-Watson:                   1.969
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              902.338
Skew:                           0.266   Prob(JB):                   1.15e-196
Kurtosis:                       4.212   Cond. No.                     6.63e+05
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 6.63e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[18]:
```
print(model4.aic)
```

```
124592.02186850144
```

We can see that the AIC has not changed much from `model1`; however, the model has been significantly simplified via reducing dozens of state feature categories into just 3 regions.

1.4.2 Exercise 8:

8.1 What should we do next to minimize features?

Answer. Since we have already removed features with insignificant p - values, the next step would be to get rid of multicollinear features. These can destabilize our model coefficients and lead to overfitting in production.

8.2 Using a method of your choice, find the numerical feature(s) in `model4`, except for the three we added in Exercise 6, which exhibit multicollinearity. (Hint: consider looking at correlations.)

Answer. One possible solution is shown below:

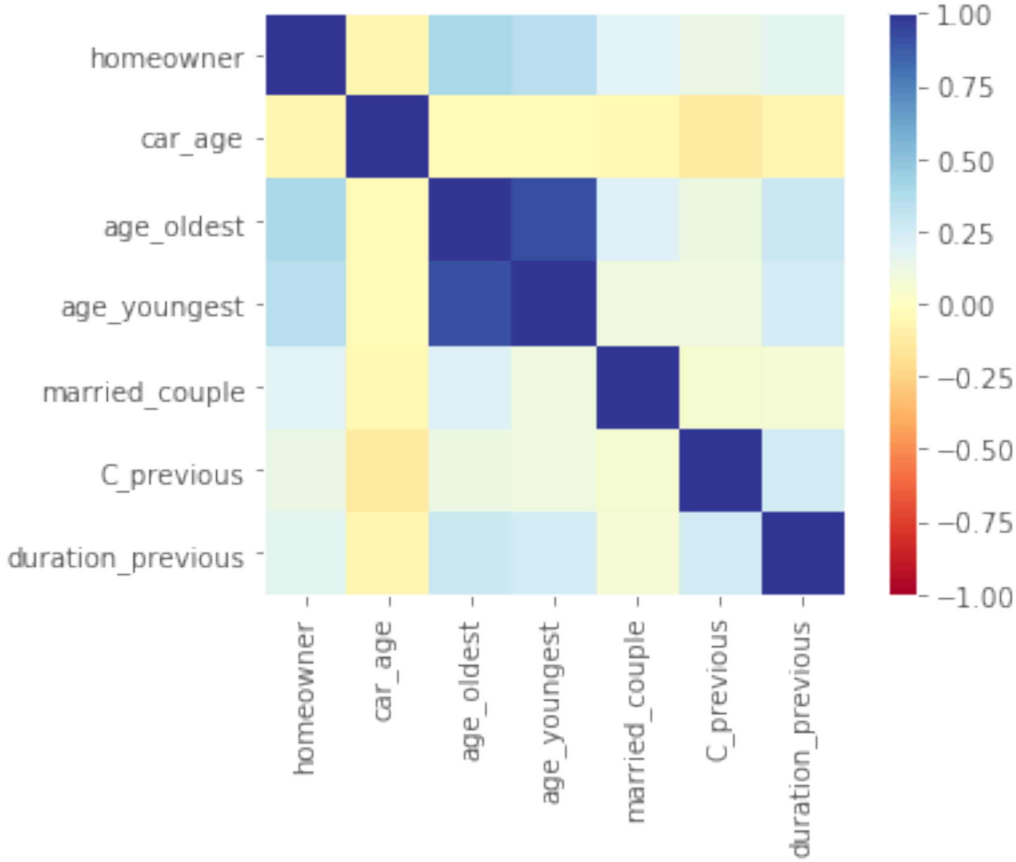
[19]:
```
predictors = ['homeowner', 'car_age', 'age_oldest', 'age_youngest',␣
 ↪'married_couple', 'C_previous', 'duration_previous']
DATA_region[predictors].corr()
```

[19]:
```
                   homeowner   car_age  age_oldest  age_youngest  \
homeowner           1.000000 -0.051016    0.403996      0.351993
car_age            -0.051016  1.000000   -0.026542     -0.028675
age_oldest          0.403996 -0.026542    1.000000      0.917221
age_youngest        0.351993 -0.028675    0.917221      1.000000
married_couple      0.190645 -0.040803    0.207119      0.103528
C_previous          0.134586 -0.125872    0.127766      0.116889
duration_previous   0.175860 -0.060889    0.275785      0.247515

                   married_couple  C_previous  duration_previous
homeowner                0.190645    0.134586           0.175860
car_age                 -0.040803   -0.125872          -0.060889
age_oldest               0.207119    0.127766           0.275785
age_youngest             0.103528    0.116889           0.247515
married_couple           1.000000    0.055971           0.071656
C_previous               0.055971    1.000000           0.257368
duration_previous        0.071656    0.257368           1.000000
```

[20]:
```
# the same information displayed visually
plt.imshow(
    DATA[predictors].corr(), # the correlation matrix
    vmin=-1,        # minimum value for the colorbar
    vmax=1,         # maximum value for the colorbar
    cmap="RdYlBu", # color scheme
)
plt.grid(False)
# label the axes:
plt.xticks(range(len(predictors)), labels=predictors, rotation=90)
plt.yticks(range(len(predictors)), labels=predictors)
plt.colorbar()
```

[20]:
```
<matplotlib.colorbar.Colorbar at 0x11c5d87f0>
```



The correlation between `age_youngest` and `age_oldest` is 0.917221, and due to this, we remove `age_oldest` - the two predictors are redundant and carry almost the same information.

8.3: Refit `model4` after dropping these redundant predictor(s); call this `model5`.

Answer. One possible solution is given below:

[21]:
```
DATA_drop_oldest = DATA_region.drop(columns=['age_oldest'])
train_drop_oldest = DATA_drop_oldest.loc[idx_train]
test_drop_oldest = DATA_drop_oldest.loc[idx_test]
model5 = smf.ols(formula = "cost ~ region + car_value + A + E + F + G +␣
 ↪homeowner + car_age + age_youngest + married_couple + C_previous +␣
 ↪duration_previous + I(age_youngest**2) + I(car_age**2) +␣
 ↪car_value*age_youngest", data = train_drop_oldest).fit()
print(model5.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   cost   R-squared:                       0.368
Model:                            OLS   Adj. R-squared:                  0.366
```

```
Method:                 Least Squares   F-statistic:                     199.2
Date:                Sat, 16 Nov 2019   Prob (F-statistic):               0.00
Time:                        11:38:50   Log-Likelihood:                -62303.
No. Observations:               12352   AIC:                         1.247e+05
Df Residuals:                   12315   BIC:                         1.250e+05
Df Model:                          36
Covariance Type:            nonrobust
================================================================================
===============
                            coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
---------------
Intercept               688.8682     27.365     25.174      0.000
635.229     742.507
region[T.Northeast]      31.9351      1.226     26.046      0.000
29.532      34.338
region[T.South]          23.2001      1.017     22.803      0.000
21.206      25.194
region[T.West]           14.1541      1.192     11.871      0.000
11.817      16.491
car_value[T.b]          -66.2424     31.867     -2.079      0.038
-128.707      -3.778
car_value[T.c]          -24.6634     27.468     -0.898      0.369
-78.506      29.179
car_value[T.d]          -27.5288     27.387     -1.005      0.315
-81.212      26.155
car_value[T.e]          -20.1220     27.350     -0.736      0.462
-73.733      33.489
car_value[T.f]          -18.7289     27.373     -0.684      0.494
-72.384      34.926
car_value[T.g]          -12.2807     27.462     -0.447      0.655
-66.111      41.549
car_value[T.h]            3.9317     27.859      0.141      0.888
-50.677      58.541
car_value[T.i]           50.5361     33.206      1.522      0.128
-14.553     115.625
A[T.1]                   39.3508      1.391     28.282      0.000
36.623      42.078
A[T.2]                   44.8711      1.729     25.947      0.000
41.481      48.261
E[T.1]                   12.6780      0.796     15.932      0.000
11.118      14.238
F[T.1]                   -3.4629      1.320     -2.623      0.009
-6.050      -0.875
F[T.2]                   -3.3604      1.285     -2.614      0.009
-5.880      -0.841
F[T.3]                   -6.8635      2.192     -3.131      0.002
```

```
-11.160      -2.567
G[T.2]                         10.4466      0.934     11.188      0.000
8.616      12.277
G[T.3]                          6.9850      1.059      6.598      0.000
4.910       9.060
G[T.4]                          7.9331      1.267      6.262      0.000
5.450      10.416
homeowner                     -12.5448      0.750    -16.723      0.000
-14.015     -11.074
car_age                        -1.3211      0.169     -7.833      0.000
-1.652      -0.990
age_youngest                   -2.3224      0.462     -5.030      0.000
-3.228      -1.417
car_value[T.b]:age_youngest     0.3049      0.585      0.522      0.602
-0.841       1.451
car_value[T.c]:age_youngest    -0.1526      0.457     -0.334      0.738
-1.048       0.743
car_value[T.d]:age_youngest     0.1023      0.447      0.229      0.819
-0.774       0.979
car_value[T.e]:age_youngest    -0.0682      0.446     -0.153      0.878
-0.942       0.806
car_value[T.f]:age_youngest    -0.0801      0.446     -0.179      0.858
-0.955       0.795
car_value[T.g]:age_youngest    -0.1500      0.448     -0.335      0.738
-1.029       0.729
car_value[T.h]:age_youngest    -0.3022      0.457     -0.662      0.508
-1.197       0.593
car_value[T.i]:age_youngest    -0.6869      0.579     -1.186      0.235
-1.822       0.448
married_couple                 -5.3483      0.862     -6.202      0.000
-7.039      -3.658
C_previous                     -6.6964      0.322    -20.780      0.000
-7.328      -6.065
duration_previous              -1.3388      0.076    -17.594      0.000
-1.488      -1.190
I(age_youngest ** 2)            0.0208      0.001     15.774      0.000
0.018       0.023
I(car_age ** 2)                 0.0307      0.008      3.980      0.000
0.016       0.046
==============================================================================
Omnibus:                      464.981   Durbin-Watson:                   1.971
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              928.469
Skew:                           0.273   Prob(JB):                    2.43e-202
Kurtosis:                       4.227   Cond. No.                     6.63e+05
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
```

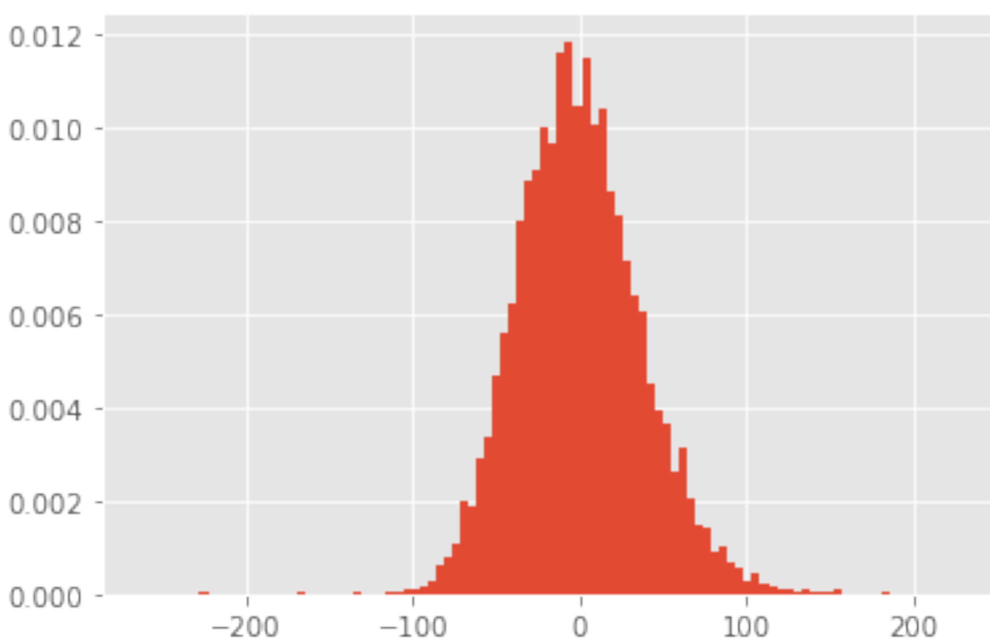
```
specified.
[2] The condition number is large, 6.63e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

We notice that the coefficient for `age_youngest` has changed: it has gone from `-2.634` in the previous model to `-2.322`. In extreme cases, the sign of the coefficient can even flip. This illustrates how much multicollinearity can destabilize a model.

8.4 What would you do to diagnose the `model5` fit? What does this diagnosis suggest to you? (Hint: try plotting the residuals in various ways.)

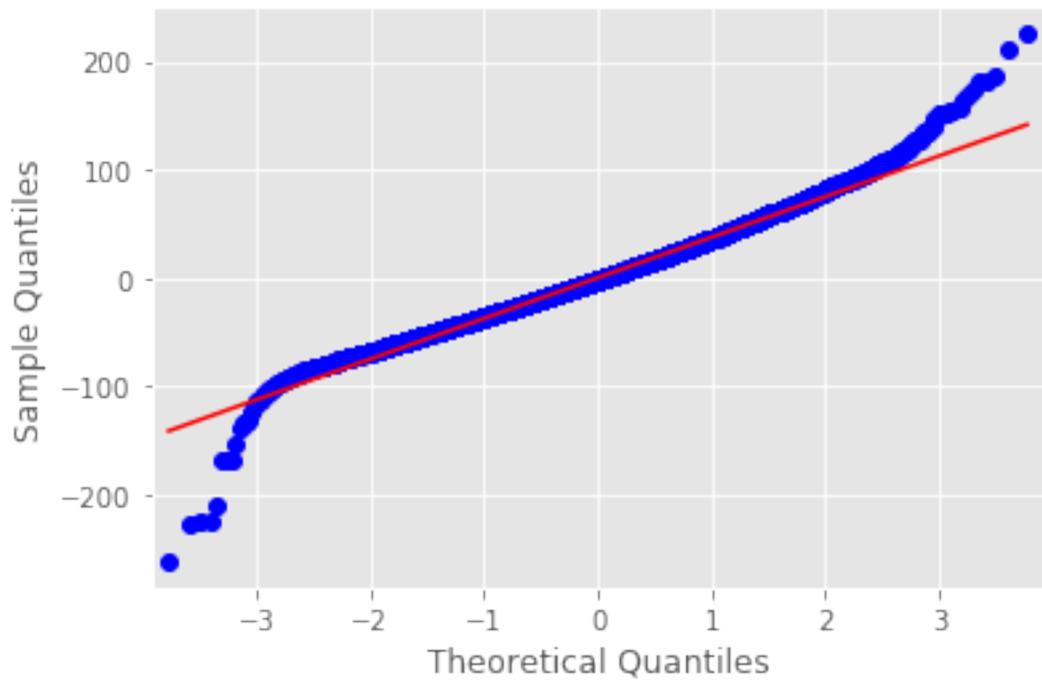
Answer. A natural step is to visualize the residuals, via plots like histograms and the QQ plot. Let's start with the histogram:

[22]:
```
plt.hist(model5.resid,
    density=True,      # the histogram integrates to 1
                       # (so it can be compared to the normal distribution)
    bins=100,          #  draw a histogram with 100 bins of equal width
    label="residuals" # label for legend
    );
```



This seems ok. Let's look at the QQ plot now:

[23]:
```
sm.qqplot(model5.resid, line="s");
```



Now we begin to see some issues. We notice that the far away quantiles do not conform to the line – this means that the tails are fatter than were apparent at first from the histogram. This means that we should consider some variable transformations to get this back in line.

1.4.3 Exercise 9:

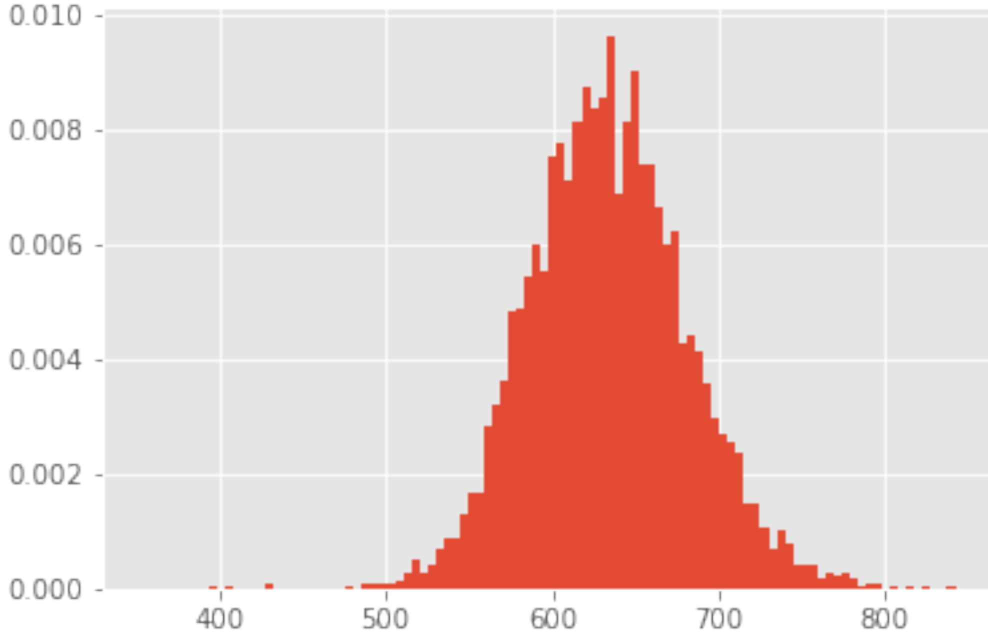
9.1 Find the best Box-Cox transformation of `cost` used to fit `model5`. What value do you get?

Answer. Shown below:

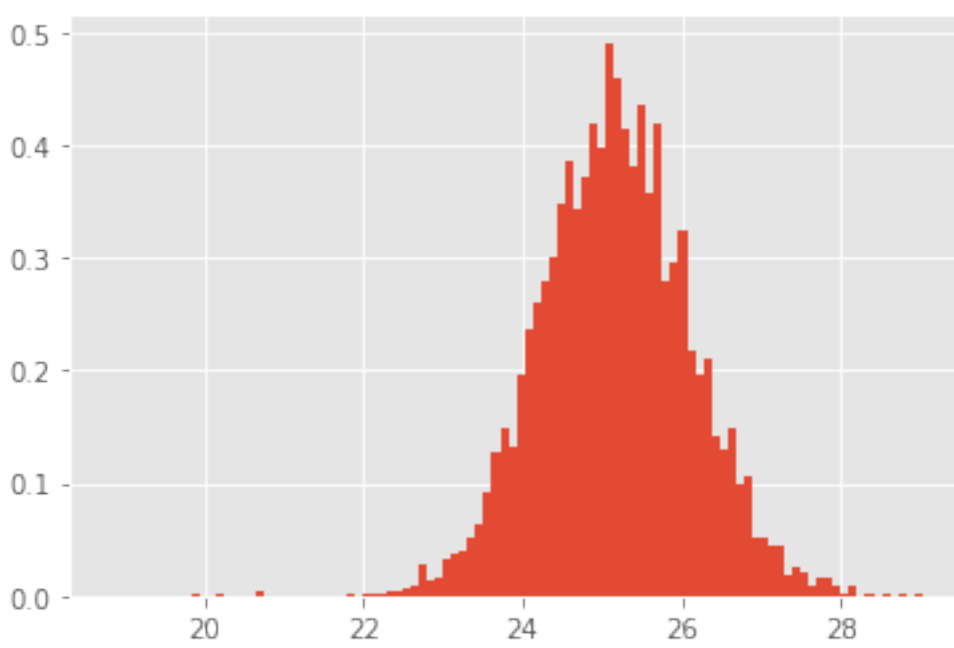
[24]:
```
from scipy import stats
price,fitted_lambda = stats.boxcox(DATA_drop_oldest['cost'])
round(fitted_lambda,2)
```

[24]: 0.53

[25]:
```
plt.hist(DATA_drop_oldest['cost'],density=True, bins = 100);
```



[26]:
```
plt.hist(np.sqrt(DATA_drop_oldest['cost']),density=True, bins = 100);
```



They both look okay, though the square root transformation looks slightly better visually.

9.2 Refit `model5`, but now with the transformation as suggested by the Box-Cox. Call it `model6`.

Answer. One possible solution is shown below:

[27]:
```
DATA_drop_oldest['cost_sqrt'] = np.sqrt(DATA_drop_oldest['cost'])
train_drop_oldest = DATA_drop_oldest.loc[idx_train]
test_drop_oldest = DATA_drop_oldest.loc[idx_test]
model6 = smf.ols(formula = 'cost_sqrt ~ day + group_size + homeowner + car_age␣
 ↪+ car_value + risk_factor + age_youngest + married_couple + C_previous +␣
 ↪duration_previous + A + B + C + D + E + F + G + region', data =␣
 ↪train_drop_oldest).fit()
print(model6.summary())
```

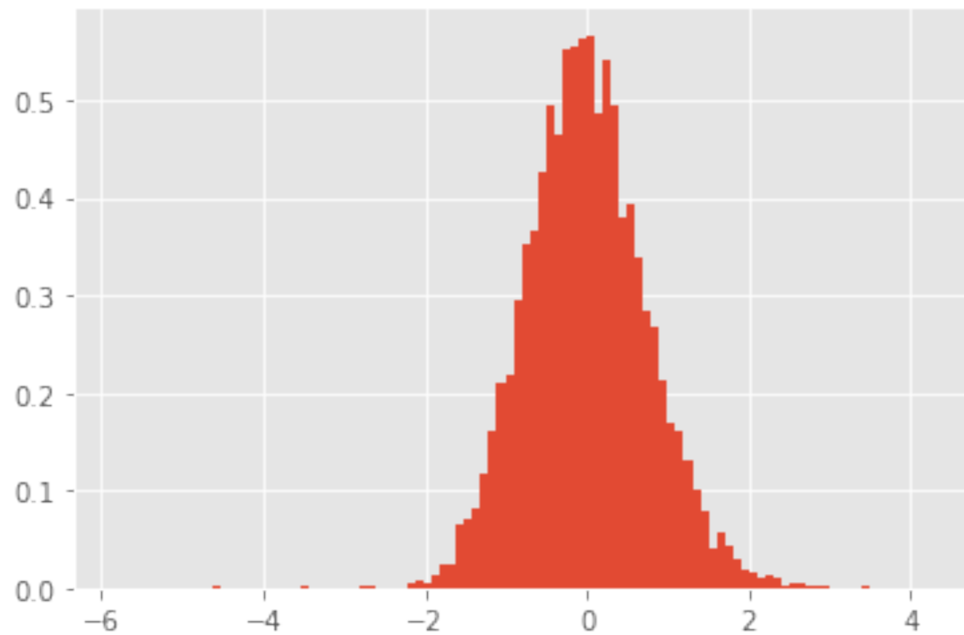
```
                            OLS Regression Results
==============================================================================
Dep. Variable:              cost_sqrt   R-squared:                       0.361
Model:                            OLS   Adj. R-squared:                  0.359
Method:                 Least Squares   F-statistic:                     174.0
Date:                Sat, 16 Nov 2019   Prob (F-statistic):               0.00
Time:                        11:38:51   Log-Likelihood:                -13960.
No. Observations:               12352   AIC:                         2.800e+04
Df Residuals:                   12311   BIC:                         2.831e+04
Df Model:                          40
Covariance Type:            nonrobust
================================================================================
=======
                          coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-------
Intercept              25.8493      0.190    135.824      0.000      25.476
26.222
day[T.1]                0.0482      0.020      2.382      0.017       0.009
0.088
day[T.2]               -0.0453      0.021     -2.156      0.031      -0.087
-0.004
day[T.3]                0.0197      0.021      0.921      0.357      -0.022
0.062
day[T.4]               -0.0397      0.022     -1.810      0.070      -0.083
0.003
day[T.5]                0.1045      0.058      1.817      0.069      -0.008
0.217
day[T.6]               -0.0557      0.183     -0.304      0.761      -0.415
0.303
car_value[T.b]         -1.4529      0.220     -6.618      0.000      -1.883
-1.023
car_value[T.c]         -1.1686      0.184     -6.366      0.000      -1.528
-0.809
```

car_value[T.d]	-1.0351	0.182	-5.698	0.000	-1.391	-0.679
car_value[T.e]	-1.0429	0.181	-5.752	0.000	-1.398	-0.687
car_value[T.f]	-1.0377	0.182	-5.715	0.000	-1.394	-0.682
car_value[T.g]	-0.9767	0.182	-5.365	0.000	-1.334	-0.620
car_value[T.h]	-0.8112	0.184	-4.401	0.000	-1.172	-0.450
car_value[T.i]	-0.3334	0.203	-1.643	0.100	-0.731	0.064
A[T.1]	0.7758	0.028	27.474	0.000	0.720	0.831
A[T.2]	0.8900	0.035	25.550	0.000	0.822	0.958
B[T.1]	0.0455	0.016	2.801	0.005	0.014	0.077
C[T.2]	0.0164	0.023	0.721	0.471	-0.028	0.061
C[T.3]	0.0267	0.024	1.128	0.259	-0.020	0.073
C[T.4]	0.0947	0.036	2.638	0.008	0.024	0.165
D[T.2]	-0.0108	0.025	-0.433	0.665	-0.060	0.038
D[T.3]	0.0030	0.026	0.117	0.907	-0.047	0.053
E[T.1]	0.2115	0.018	11.726	0.000	0.176	0.247
F[T.1]	-0.0527	0.026	-1.993	0.046	-0.104	-0.001
F[T.2]	-0.0568	0.026	-2.201	0.028	-0.107	-0.006
F[T.3]	-0.1432	0.044	-3.249	0.001	-0.230	-0.057
G[T.2]	0.2065	0.019	11.070	0.000	0.170	0.243
G[T.3]	0.1421	0.021	6.681	0.000	0.100	0.184
G[T.4]	0.1494	0.025	5.890	0.000	0.100	0.199
region[T.Northeast]	0.6361	0.025	25.866	0.000	0.588	0.684
region[T.South]	0.4705	0.021	22.675	0.000	0.430	0.511
region[T.West]	0.2810	0.024	11.695	0.000	0.234	0.328

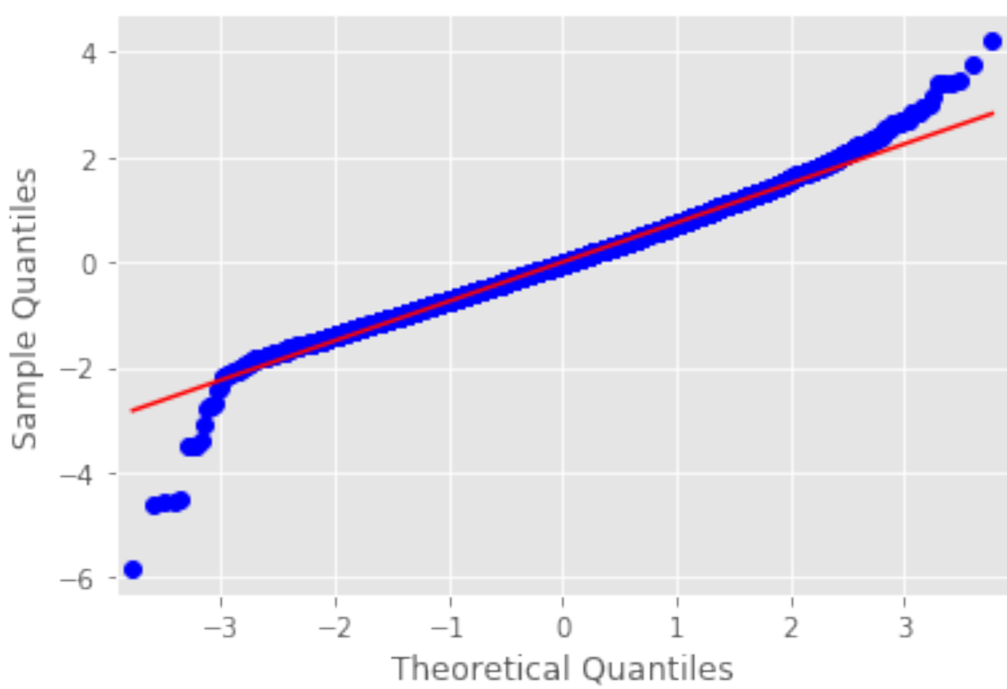
```
group_size              0.1895      0.024      7.796      0.000       0.142
0.237
homeowner              -0.2770      0.015    -18.322      0.000      -0.307
-0.247
car_age                -0.0146      0.001    -10.205      0.000      -0.017
-0.012
risk_factor             0.0096      0.005      2.105      0.035       0.001
0.018
age_youngest           -0.0085      0.000    -19.347      0.000      -0.009
-0.008
married_couple         -0.2707      0.027     -9.964      0.000      -0.324
-0.217
C_previous             -0.1349      0.008    -17.965      0.000      -0.150
-0.120
duration_previous      -0.0265      0.002    -17.321      0.000      -0.029
-0.023
==============================================================================
Omnibus:                      354.496   Durbin-Watson:                   1.966
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              795.698
Skew:                           0.147   Prob(JB):                    1.65e-173
Kurtosis:                       4.208   Cond. No.                     3.87e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 3.87e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[28]:
```python
plt.hist(model6.resid,
    density=True,      # the histogram integrates to 1
                       # (so it can be compared to the normal distribution)
    bins=100,          #  draw a histogram with 100 bins of equal width
    label="residuals" # label for legend
    );
```



[29]:
```
sm.qqplot(model6.resid, line="s");
```



We see that the AIC of `model6` is much better than that of `model5`. However, the QQ plot is still exhibiting strange tail behavior.