In [2]:

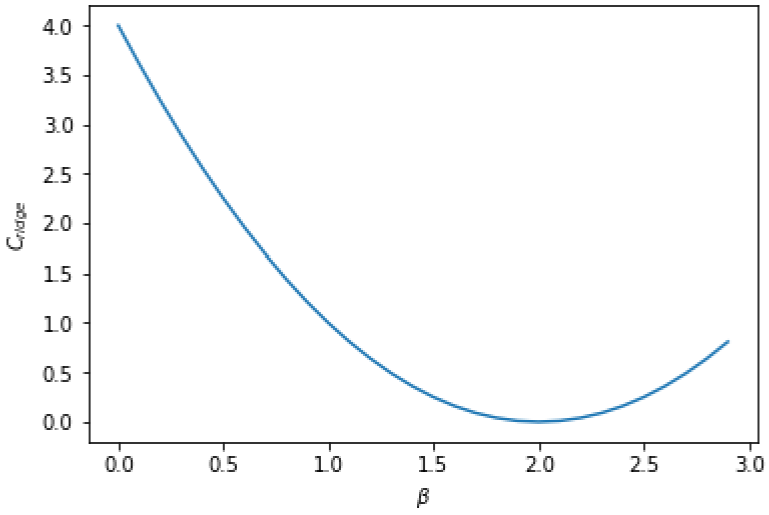
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import pearsonr
import statsmodels.formula.api as smf
from sklearn.model_selection import KFold
```

1. Ridge regression and LASSO for one variable.

(a)

In [3]:

```
def c_ridge(b):
    return (2-b)**2

b = np.arange(0, 3, 0.1)
fig, ax = plt.subplots()
ax.plot(b, c_ridge(b))
ax.set_xlabel(r'$\beta$')
ax.set_ylabel(r'$C_{ridge}$')
plt.show()
```

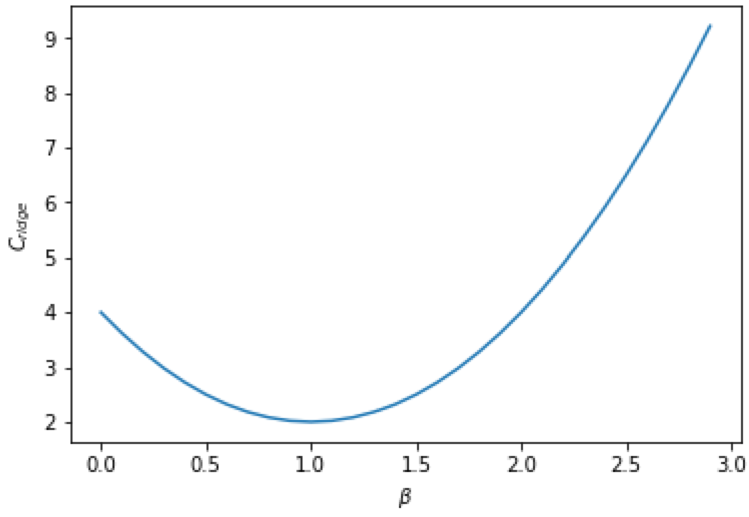


$$C_{\text{ridge}}(\ ) = (2 - \tilde{\ }\ )^2$$
$$C'_{\text{ridge}}(\ ) = -2(2 - \tilde{\ }\ ) = 0 \Rightarrow \boxed{\hat{\ } = 2}$$

(b)

In [4]:

```
def c_ridge(b):
    return (2-b)**2 + b**2

b = np.arange(0, 3, 0.1)
fig, ax = plt.subplots()
ax.plot(b, c_ridge(b))
ax.set_xlabel(r'$\beta$')
ax.set_ylabel(r'$C_{ridge}$')
plt.show()
```



$$C_{\text{ridge}}(\beta) = (y - \beta)^2 + \lambda\beta^2$$
$$C'_{\text{ridge}}(\beta) = -2(y - \beta) + 2\lambda\beta = 0 \Rightarrow \boxed{\hat{\beta} = \frac{y}{1+\lambda}}$$

When $\lambda = 1$,

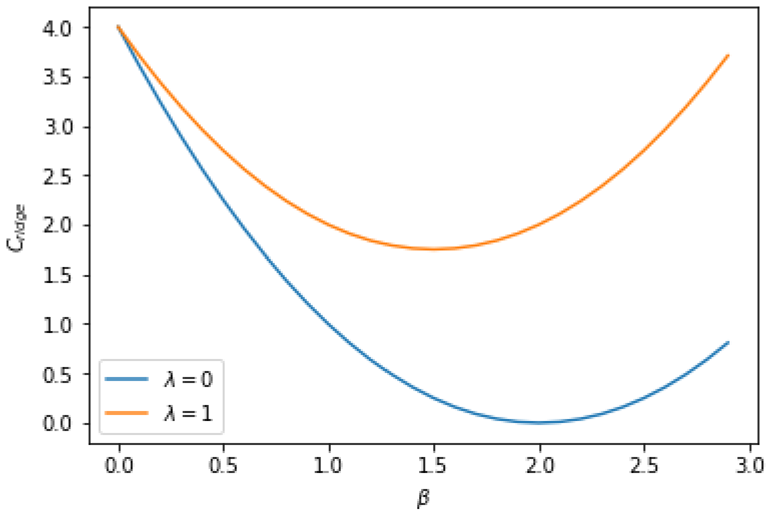
$$\boxed{\hat{\beta} = \frac{y}{2}}$$

As λ increases, $|\hat{\beta}|$ decreases and eventually saturates at 0.

(c)

In [5]:

```
def c_ridge(b, l):
    return (2-b)**2 + l*np.abs(b)

b = np.arange(0, 3, 0.1)
fig, ax = plt.subplots()
ax.plot(b, c_ridge(b, 0), label=r'$\lambda=0$')
ax.plot(b, c_ridge(b, 1), label=r'$\lambda=1$')
ax.set_xlabel(r'$\beta$')
ax.set_ylabel(r'$C_{ridge}$')
ax.legend()
plt.show()
```



From the above plot, it's obvious that $\hat{}_1 < \hat{}_0$, which shows LASSO shrinks the absolute value of $\hat{}$.

2. Forward stepwise model selection.

(a)

In [6]:

```
df = pd.read_csv('anesthesia.csv')
df.head()
```

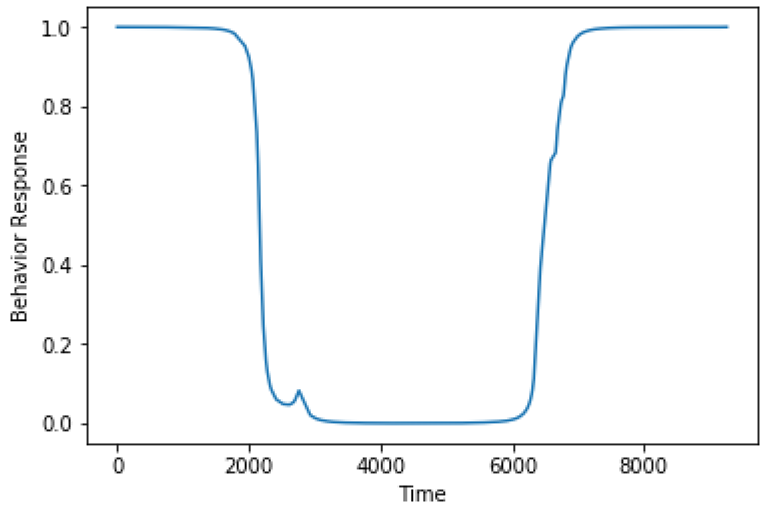
Out[6]:

	Time	F0Hz_1	F1Hz_2	F3Hz_3	F4Hz_4	F5Hz_5	F6Hz_6	F8Hz_7	F9H
0	5.004	3.115293	1.676500	1.097419	0.900837	0.537178	0.454494	0.512818	-0.13
1	15.004	2.864158	1.499845	0.879378	1.020294	0.281333	0.722017	0.086080	0.080
2	25.004	2.039253	1.057344	0.163134	0.351954	0.149567	0.325558	0.231917	0.284
3	35.004	2.417074	0.348083	0.582521	0.468952	0.176949	0.116783	0.200230	0.166
4	45.004	2.507836	1.036731	0.622822	0.436470	0.465713	0.703881	0.048926	-0.32

5 rows × 105 columns

In [7]:

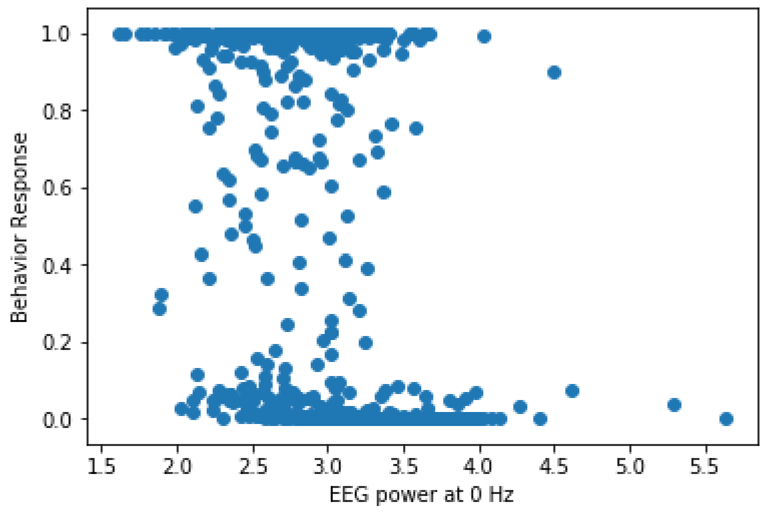
```
fig, ax = plt.subplots()
ax.plot(df.Time, df.BehaviorResponse)
ax.set_xlabel('Time')
ax.set_ylabel('Behavior Response')
plt.show()
```



(b)

In [8]:

```
fig, ax = plt.subplots()
ax.scatter(df.F0Hz_1, df.BehaviorResponse)
ax.set_xlabel('EEG power at 0 Hz')
ax.set_ylabel('Behavior Response')
plt.show()
```



As EEG power at 0 Hz increases, the probability of BehaviorResponse=1 decreases.

(c)

In [9]:

```
corr = pearsonr(df.F0Hz_1, df.BehaviorResponse)[0]
print('The correlation coefficient is %f.' % corr)
```

```
The correlation coefficient is -0.463212.
```

(d)

In [10]:

```
res_lm = smf.ols(formula='BehaviorResponse ~ 1 + F0Hz_1', data=df).fit(disp=0)
res_lm.summary()
```

Out[10]:

OLS Regression Results

Dep. Variable:	BehaviorResponse	**R-squared:**	0.215
Model:	OLS	**Adj. R-squared:**	0.214
Method:	Least Squares	**F-statistic:**	252.4
Date:	Mon, 06 Nov 2017	**Prob (F-statistic):**	1.96e-50
Time:	09:38:29	**Log-Likelihood:**	-510.49
No. Observations:	926	**AIC:**	1025.
Df Residuals:	924	**BIC:**	1035.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	1.8503	0.084	22.000	0.000	1.685	2.015
F0Hz_1	-0.4433	0.028	-15.888	0.000	-0.498	-0.389

Omnibus:	995.624	**Durbin-Watson:**	0.198
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	63.317
Skew:	-0.152	**Prob(JB):**	1.78e-14
Kurtosis:	1.755	**Cond. No.**	20.3

The slope coefficient is statistically significant. (P-value < 0.05)

(e)

In [11]:

```
# create formula
formula = 'BehaviorResponse ~ 1'
for col in df.columns:
    if col not in ['Time', 'BehaviorResponse']:
        formula = formula + ' + ' + col
print(formula)
```

```
BehaviorResponse ~ 1 + F0Hz_1 + F1Hz_2 + F3Hz_3 + F4Hz_4 + F5Hz_5 +
F6Hz_6 + F8Hz_7 + F9Hz_8 + F10Hz_9 + F11Hz_10 + F12Hz_11 + F14Hz_12
+ F15Hz_13 + F16Hz_14 + F17Hz_15 + F19Hz_16 + F20Hz_17 + F21Hz_18 +
F22Hz_19 + F23Hz_20 + F25Hz_21 + F26Hz_22 + F27Hz_23 + F28Hz_24 + F3
0Hz_25 + F31Hz_26 + F32Hz_27 + F33Hz_28 + F34Hz_29 + F36Hz_30 + F37H
z_31 + F38Hz_32 + F39Hz_33 + F41Hz_34 + F42Hz_35 + F43Hz_36 + F44Hz_
37 + F45Hz_38 + F47Hz_39 + F48Hz_40 + F49Hz_41 + F50Hz_42 + F52Hz_43
+ F53Hz_44 + F54Hz_45 + F55Hz_46 + F56Hz_47 + F58Hz_48 + F59Hz_49 +
F60Hz_50 + F61Hz_51 + F63Hz_52 + F64Hz_53 + F65Hz_54 + F66Hz_55 + F6
7Hz_56 + F69Hz_57 + F70Hz_58 + F71Hz_59 + F72Hz_60 + F73Hz_61 + F75H
z_62 + F76Hz_63 + F77Hz_64 + F78Hz_65 + F80Hz_66 + F81Hz_67 + F82Hz_
68 + F83Hz_69 + F84Hz_70 + F86Hz_71 + F87Hz_72 + F88Hz_73 + F89Hz_74
+ F91Hz_75 + F92Hz_76 + F93Hz_77 + F94Hz_78 + F95Hz_79 + F97Hz_80 +
F98Hz_81 + F99Hz_82 + F100Hz_83 + F102Hz_84 + F103Hz_85 + F104Hz_86
+ F105Hz_87 + F106Hz_88 + F108Hz_89 + F109Hz_90 + F110Hz_91 + F111Hz
_92 + F113Hz_93 + F114Hz_94 + F115Hz_95 + F116Hz_96 + F117Hz_97 + F1
19Hz_98 + F120Hz_99 + F121Hz_100 + F122Hz_101 + F124Hz_102 + F125Hz_
103
```

In [12]:

```
res_lm = smf.ols(formula=formula, data=df).fit(disp=0)
res_lm.summary()
```

```
Out[12]:
```

OLS Regression Results

Dep. Variable:	BehaviorResponse	**R-squared:**	0.884
Model:	OLS	**Adj. R-squared:**	0.869
Method:	Least Squares	**F-statistic:**	60.54
Date:	Mon, 06 Nov 2017	**Prob (F-statistic):**	3.02e-319
Time:	09:38:31	**Log-Likelihood:**	373.22
No. Observations:	926	**AIC:**	-538.4
Df Residuals:	822	**BIC:**	-36.03
Df Model:	103		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Intercept	0.6519	0.131	4.994	0.000	0.396	0.908
F0Hz_1	-0.0122	0.017	-0.698	0.485	-0.046	0.022
F1Hz_2	-0.0394	0.025	-1.560	0.119	-0.089	0.010
F3Hz_3	0.0006	0.028	0.020	0.984	-0.054	0.055
F4Hz_4	-0.0060	0.027	-0.226	0.822	-0.058	0.046
F5Hz_5	0.0008	0.025	0.032	0.974	-0.049	0.050
F6Hz_6	0.1510	0.022	6.858	0.000	0.108	0.194
F8Hz_7	-0.0003	0.024	-0.012	0.991	-0.047	0.047
F9Hz_8	0.0066	0.027	0.247	0.805	-0.046	0.059
F10Hz_9	-0.1492	0.024	-6.324	0.000	-0.195	-0.103
F11Hz_10	-0.1303	0.023	-5.771	0.000	-0.175	-0.086
F12Hz_11	-0.1368	0.022	-6.167	0.000	-0.180	-0.093
F14Hz_12	-0.0925	0.024	-3.886	0.000	-0.139	-0.046
F15Hz_13	0.0077	0.025	0.304	0.761	-0.042	0.057
F16Hz_14	-0.0317	0.025	-1.246	0.213	-0.082	0.018
F17Hz_15	-0.0009	0.026	-0.037	0.971	-0.051	0.049
F19Hz_16	-0.0191	0.025	-0.753	0.451	-0.069	0.031
F20Hz_17	-0.0358	0.026	-1.380	0.168	-0.087	0.015
F21Hz_18	-0.0736	0.027	-2.734	0.006	-0.126	-0.021
F22Hz_19	0.0318	0.028	1.155	0.248	-0.022	0.086
F23Hz_20	0.0120	0.026	0.452	0.651	-0.040	0.064
F25Hz_21	-0.0057	0.028	-0.206	0.837	-0.060	0.049
F26Hz_22	-0.0223	0.028	-0.781	0.435	-0.078	0.034

F27Hz_23	0.0373	0.028	1.328	0.185	-0.018	0.092
F28Hz_24	-0.0381	0.029	-1.320	0.187	-0.095	0.019
F30Hz_25	0.0307	0.029	1.075	0.283	-0.025	0.087
F31Hz_26	-0.0145	0.027	-0.539	0.590	-0.067	0.038
F32Hz_27	0.0254	0.029	0.888	0.375	-0.031	0.082
F33Hz_28	0.0167	0.028	0.595	0.552	-0.038	0.072
F34Hz_29	0.0242	0.030	0.815	0.415	-0.034	0.082
F36Hz_30	-0.0239	0.028	-0.862	0.389	-0.078	0.031
F37Hz_31	0.0113	0.028	0.401	0.688	-0.044	0.066
F38Hz_32	-0.0360	0.028	-1.270	0.205	-0.092	0.020
F39Hz_33	-0.0177	0.029	-0.615	0.539	-0.074	0.039
F41Hz_34	0.0098	0.030	0.330	0.742	-0.049	0.068
F42Hz_35	-0.0230	0.028	-0.819	0.413	-0.078	0.032
F43Hz_36	-0.0297	0.028	-1.051	0.293	-0.085	0.026
F44Hz_37	0.0051	0.029	0.174	0.862	-0.053	0.063
F45Hz_38	-0.0078	0.028	-0.277	0.782	-0.063	0.048
F47Hz_39	0.0032	0.030	0.107	0.915	-0.056	0.062
F48Hz_40	-0.0326	0.029	-1.131	0.258	-0.089	0.024
F49Hz_41	-0.0149	0.029	-0.511	0.610	-0.072	0.042
F50Hz_42	0.0376	0.030	1.265	0.206	-0.021	0.096
F52Hz_43	0.0064	0.030	0.215	0.830	-0.052	0.065
F53Hz_44	-0.0354	0.030	-1.187	0.236	-0.094	0.023
F54Hz_45	-0.0303	0.029	-1.056	0.291	-0.087	0.026
F55Hz_46	0.0121	0.030	0.409	0.683	-0.046	0.070
F56Hz_47	0.0217	0.030	0.724	0.469	-0.037	0.080
F58Hz_48	0.0217	0.030	0.723	0.470	-0.037	0.081
F59Hz_49	0.0019	0.030	0.063	0.949	-0.056	0.060
F60Hz_50	0.0465	0.031	1.491	0.136	-0.015	0.108
F61Hz_51	0.0116	0.029	0.404	0.686	-0.045	0.068
F63Hz_52	0.0208	0.029	0.705	0.481	-0.037	0.079
F64Hz_53	0.0139	0.030	0.463	0.643	-0.045	0.073
F65Hz_54	-0.0380	0.030	-1.276	0.202	-0.096	0.020
F66Hz_55	-0.0728	0.030	-2.458	0.014	-0.131	-0.015
F67Hz_56	0.0147	0.029	0.500	0.617	-0.043	0.072
F69Hz_57	-0.0244	0.029	-0.841	0.400	-0.081	0.033
F70Hz_58	0.0277	0.029	0.944	0.346	-0.030	0.085

F71Hz_59	0.0289	0.029	0.997	0.319	-0.028	0.086
F72Hz_60	-0.0192	0.030	-0.651	0.515	-0.077	0.039
F73Hz_61	-0.0600	0.029	-2.039	0.042	-0.118	-0.002
F75Hz_62	0.0268	0.028	0.948	0.343	-0.029	0.082
F76Hz_63	-0.0701	0.029	-2.400	0.017	-0.127	-0.013
F77Hz_64	0.0075	0.030	0.247	0.805	-0.052	0.067
F78Hz_65	-0.0225	0.030	-0.759	0.448	-0.081	0.036
F80Hz_66	-0.0009	0.030	-0.031	0.976	-0.060	0.058
F81Hz_67	-0.0231	0.028	-0.823	0.411	-0.078	0.032
F82Hz_68	-0.0181	0.030	-0.595	0.552	-0.078	0.041
F83Hz_69	-0.0066	0.029	-0.224	0.823	-0.064	0.051
F84Hz_70	-0.0037	0.030	-0.124	0.902	-0.062	0.055
F86Hz_71	-0.0292	0.029	-1.005	0.315	-0.086	0.028
F87Hz_72	0.0318	0.030	1.075	0.283	-0.026	0.090
F88Hz_73	0.0017	0.028	0.062	0.951	-0.053	0.057
F89Hz_74	-0.0250	0.029	-0.852	0.395	-0.082	0.033
F91Hz_75	-0.0363	0.030	-1.218	0.223	-0.095	0.022
F92Hz_76	-0.0449	0.029	-1.535	0.125	-0.102	0.013
F93Hz_77	0.0286	0.028	1.011	0.313	-0.027	0.084
F94Hz_78	-0.0041	0.029	-0.142	0.887	-0.060	0.052
F95Hz_79	-0.0259	0.029	-0.900	0.369	-0.083	0.031
F97Hz_80	0.0099	0.030	0.331	0.741	-0.049	0.068
F98Hz_81	0.0219	0.029	0.761	0.447	-0.035	0.078
F99Hz_82	0.0159	0.028	0.574	0.566	-0.039	0.070
F100Hz_83	-0.0434	0.029	-1.484	0.138	-0.101	0.014
F102Hz_84	-0.0477	0.030	-1.613	0.107	-0.106	0.010
F103Hz_85	-0.0314	0.028	-1.116	0.265	-0.087	0.024
F104Hz_86	0.0043	0.029	0.152	0.880	-0.052	0.060
F105Hz_87	-0.0228	0.030	-0.750	0.454	-0.083	0.037
F106Hz_88	0.0045	0.031	0.144	0.886	-0.057	0.066
F108Hz_89	0.0788	0.030	2.593	0.010	0.019	0.139
F109Hz_90	-0.0073	0.030	-0.245	0.807	-0.066	0.051
F110Hz_91	0.0339	0.032	1.056	0.291	-0.029	0.097
F111Hz_92	0.0094	0.033	0.288	0.773	-0.055	0.074
F113Hz_93	-0.0617	0.038	-1.604	0.109	-0.137	0.014
F114Hz_94	0.0643	0.044	1.475	0.141	-0.021	0.150

F115Hz_95	0.0248	0.054	0.463	0.643	-0.080	0.130
F116Hz_96	-0.1223	0.067	-1.818	0.069	-0.254	0.010
F117Hz_97	0.0081	0.081	0.100	0.920	-0.151	0.167
F119Hz_98	0.0794	0.096	0.824	0.410	-0.110	0.269
F120Hz_99	-0.0291	0.116	-0.252	0.801	-0.257	0.198
F121Hz_100	-0.0648	0.127	-0.512	0.609	-0.313	0.184
F122Hz_101	0.1803	0.165	1.091	0.276	-0.144	0.505
F124Hz_102	-0.1144	0.164	-0.697	0.486	-0.437	0.208
F125Hz_103	0.0389	0.171	0.228	0.820	-0.296	0.374

Omnibus:	110.208	**Durbin-Watson:**	0.676
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	212.561
Skew:	-0.732	**Prob(JB):**	6.97e-47
Kurtosis:	4.835	**Cond. No.**	568.

The slope coefficient of F0Hz_1 is NOT statistically significant. (P-value > 0.05)

(f)

In [13]:

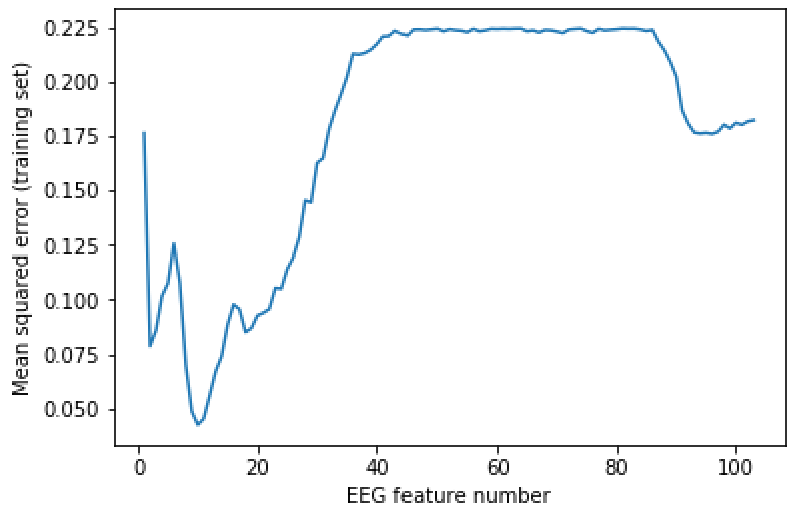
```
# get all EEG power columns
x_cols = [col for col in df.columns if col not in ['Time', 'BehaviorResponse']]
# loop over all of them
mse_info = []
for x_col in x_cols:
    formula = 'BehaviorResponse ~ 1 + %s' % x_col
    res_lm = smf.ols(formula=formula, data=df).fit(disp=0)

    # calculate mean squared error
    y_pred = res_lm.predict(df)
    y = df.BehaviorResponse
    diff = y_pred - y
    mse = (np.dot(diff, diff))/len(diff)

    mse_info.append({'EEG_feature': x_col,
                     'MSE': mse})

mse_info = pd.DataFrame(mse_info)
mse_info.head()
```

Out[13]:

	EEG_feature	MSE
0	F0Hz_1	0.176345
1	F1Hz_2	0.078632
2	F3Hz_3	0.085894
3	F4Hz_4	0.101702
4	F5Hz_5	0.107533

In [14]:

```
# make a plot
fig, ax = plt.subplots()
ax.plot(mse_info.index+1, mse_info.MSE)
ax.set_xlabel('EEG feature number')
ax.set_ylabel('Mean squared error (training set)')
plt.show()
```



In [15]:

```
# find lowest MSE
mse_info.loc[mse_info.MSE.argmin(axis=0), :]
```

Out[15]:

```
EEG_feature      F11Hz_10
MSE             0.0423969
Name: 9, dtype: object
```

As a result, F11Hz_10 gives the best prediction results.

(g)

In [16]:

```
# get all EEG power columns
x_cols = [col for col in df.columns if col not in ['Time', 'BehaviorResponse']]
# remove X1
x_cols.remove('F11Hz_10')
# loop over all of the remaining features
mse_info = []
for x_col in x_cols:

    formula = 'BehaviorResponse ~ 1 + F11Hz_10 + %s' % x_col
    res_lm = smf.ols(formula=formula, data=df).fit(disp=0)

    # calculate mean squared error
    y_pred = res_lm.predict(df)
    y = df.BehaviorResponse
    diff = y_pred - y
    mse = (np.dot(diff, diff))/len(diff)

    mse_info.append({'EEG_feature': x_col,
                     'MSE': mse})

mse_info = pd.DataFrame(mse_info)
mse_info.head()

# find lowest MSE
mse_info.loc[mse_info.MSE.argmin(), :]
```

Out[16]:

```
EEG_feature     F12Hz_11
MSE            0.0358823
Name: 9, dtype: object
```

As a result, F11Hz_10 and F12Hz_11 combined gives the best prediction results.

(h)

```
In [17]:
```

```
kf = KFold(n_splits=10, shuffle=False)
formula = 'BehaviorResponse ~ 1 + F11Hz_10 + F12Hz_11'
info = []
for train_index, test_index in kf.split(df):
    df_train, df_test = df.iloc[train_index, :], df.iloc[test_index, :]
    # train model
    res_lm = smf.ols(formula=formula, data=df_train).fit(disp=0)

    # calculate training mse
    y_pred = res_lm.predict(df_train)
    y = df_train.BehaviorResponse
    diff = y_pred - y
    train_mse = (np.dot(diff, diff))/len(diff)

    # calculate test mse
    y_pred = res_lm.predict(df_test)
    y = df_test.BehaviorResponse
    diff = y_pred - y
    test_mse = (np.dot(diff, diff))/len(diff)

    # save info
    info.append({'training mse': train_mse,
                 'test mse': test_mse})

info = pd.DataFrame(info)
info.mean(axis=0)
```

```
Out[17]:

test mse        0.041080
training mse    0.035628
dtype: float64
```

The training and test MSE are shown above.

(i)

In [18]:

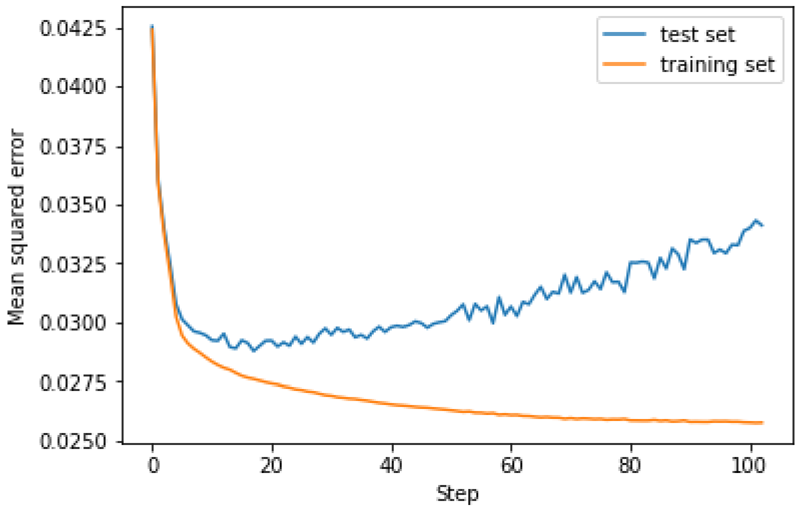
```
# This code takes several minutes to run...
FINAL_info = []

# 10-fold cross validation
kf = KFold(n_splits=10, shuffle=True)

# get all EEG power columns
x_cols = [col for col in df.columns if col not in ['Time', 'BehaviorResponse']]
x_cols = x_cols

# add one feature a time, do it len(x_cols) times
formula = 'BehaviorResponse ~ 1'
for i in range(len(x_cols)):
    # a process of adding one feature, and calculate training and testing mse

    # step 1: loop over all remaining features and select one with the lowest tr
aining mse.
    mse_info = []
    for x_col in x_cols:
        formula_try = (formula + ' + ' + x_col)
        res_lm = smf.ols(formula=formula_try, data=df).fit(disp=0)

        # calculate mean squared error
        y_pred = res_lm.predict(df)
        y = df.BehaviorResponse
        diff = y_pred - y
        mse = (np.dot(diff, diff))/len(diff)

        mse_info.append({'EEG_feature': x_col,
                         'MSE': mse})

    # find lowest training MSE
    mse_info = pd.DataFrame(mse_info)
    selected_col = mse_info.loc[mse_info.MSE.argmin(axis=0), 'EEG_feature']


    # step 2: update the current formula and remaining features
    formula = (formula + ' + ' + selected_col)
    x_cols.remove(selected_col)

    # step 3: do 10-fold cross validation on the selected model and store both M
SEs.

    train_info = []
    test_info = []
    for train_index, test_index in kf.split(df):
        df_train, df_test = df.iloc[train_index, :], df.iloc[test_index, :]
        # train model
        res_lm = smf.ols(formula=formula, data=df_train).fit(disp=0)

        # calculate training mse
        y_pred = res_lm.predict(df_train)
        y = df_train.BehaviorResponse
        diff = y_pred - y
        train_mse = (np.dot(diff, diff))/len(diff)

        # calculate test mse
        y_pred = res_lm.predict(df_test)
        y = df_test.BehaviorResponse
        diff = y_pred - y
        test_mse = (np.dot(diff, diff))/len(diff)

        # save info
        train_info.append(train_mse)
        test_info.append(test_mse)

    train_mse = np.mean(train_info)
    test_mse = np.mean(test_info)
    FINAL_info.append({'step': i,
                       'formula': formula,
                      'train_mse': train_mse,
                      'test_mse': test_mse})


FINAL_info = pd.DataFrame(FINAL_info)
FINAL_info.head()
```

Out[18]:

	formula	step	test_mse	train_mse
0	BehaviorResponse ~ 1 + F11Hz_10	0	0.042538	0.042390
1	BehaviorResponse ~ 1 + F11Hz_10 + F12Hz_11	1	0.036064	0.035872
2	BehaviorResponse ~ 1 + F11Hz_10 + F12Hz_11 + F...	2	0.034009	0.033737
3	BehaviorResponse ~ 1 + F11Hz_10 + F12Hz_11 + F...	3	0.032484	0.032064
4	BehaviorResponse ~ 1 + F11Hz_10 + F12Hz_11 + F...	4	0.030779	0.030240

In [19]:

```
fig, ax = plt.subplots()
ax.plot(FINAL_info.step, FINAL_info.test_mse, label='test set')
ax.plot(FINAL_info.step, FINAL_info.train_mse, label='training set')
ax.set_xlabel('Step')
ax.set_ylabel('Mean squared error')
ax.legend()
plt.show()
```



In [20]:

```
# find the lowest test error
FINAL_info.loc[FINAL_info.test_mse.argmin(),'formula']
```

Out[20]:

```
'BehaviorResponse ~ 1 + F11Hz_10 + F12Hz_11 + F21Hz_18 + F10Hz_9 + F
6Hz_6 + F14Hz_12 + F76Hz_63 + F92Hz_76 + F66Hz_55 + F119Hz_98 + F102
Hz_84 + F73Hz_61 + F16Hz_14 + F108Hz_89 + F113Hz_93 + F1Hz_2 + F65Hz
_54 + F100Hz_83'
```

We choose the model with the lowest test set MSE, which is the model shown above.