In [6]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from patsy import dmatrix
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error
from math import sqrt
```

In [628]:

```
df_train = pd.read_csv('/Users/lijiaxuan/Downloads/problem-set-4-master/data/gss
_train.csv')
df_test = pd.read_csv('/Users/lijiaxuan/Downloads/problem-set-4-master/data/gss_
test.csv')
```

In [636]:

```
X_train, Y_train = np.asarray(df_train['income06']).reshape(-1,1),np.asarray(df_
train['egalit_scale']).reshape(-1,1)
X_test, Y_test = np.asarray(df_test['income06']).reshape(-1,1),np.asarray(df_tes
t['egalit_scale']).reshape(-1,1)
print(X_train.shape,Y_train.shape,X_test.shape,Y_test.shape)
```

```
(1481, 1) (1481, 1) (493, 1) (493, 1)
```

# 1. Polynomial regression

In [644]:

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn import linear_model
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import KFold
mse = list(range(10))
clf = list(range(10))

for i in range(10):
    poly = PolynomialFeatures(degree=2+i)
    X_ = poly.fit_transform(X_train)
    X_test_ = poly.fit_transform(X_test)
    clf = linear_model.LinearRegression()
    #y_cv[i] = cross_val_predict(clf, X_, Y_train, cv=10)
    clf.fit(X_, Y_train)
    mse[i] = mean_squared_error(Y_train,clf.predict(X_))
print(mse)
print(min(mse))
```

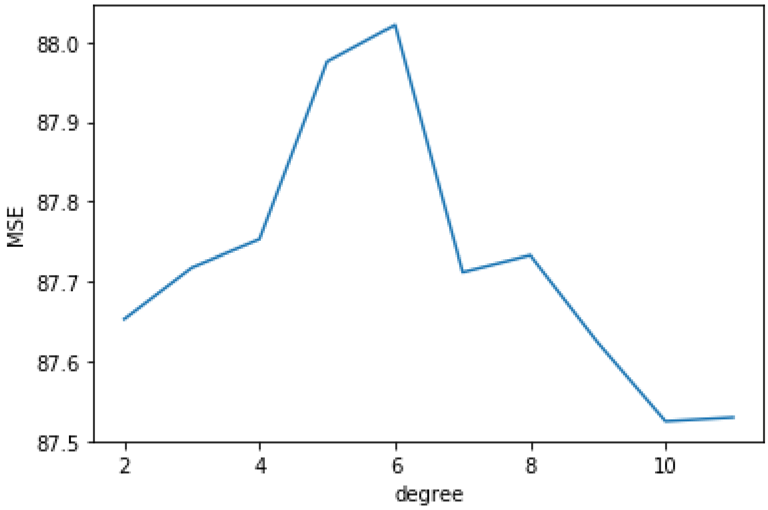
```
[86.89026213232579, 86.88767187023525, 86.88475428859854, 86.8388604
9277721, 86.81806780627095, 86.46890771724293, 86.37971600564143, 8
6.11396358707721, 86.09407503172477, 86.14647430834756]
86.09407503172477
```

In [594]:

```
plt.plot(list(range(2,12)),mse)
plt.xlabel('degree')
plt.ylabel('MSE')
```

Out[594]:

```
Text(0, 0.5, 'MSE')
```




From the graph above, we could tell that MSE reaches the lowest when degree is 9.

In [ ]:

```
plt.plot(X_test, clf[8].predict(PolynomialFeatures(degree=10).fit_transform(X_te
st)), color='blue')
plt.title('Polynomial Regression (degree 9)')
plt.xlabel('income')
plt.ylabel('egalit_scale')
plt.show()
```

In [651]:

```
def get_coef(x,y,deg):
    df = pd.DataFrame()
    df_ame = pd.DataFrame()
    ame_lst = []
    for degree in range(2,deg):
        df[degree] = x**degree
    model = LinearRegression().fit(df,y)
    coefs = model.coef_

    return coefs
```

In [655]:

```
coefs = get_coef(df_train['income06'],df_train['egalit_scale'],12)
print(len(coefs))
marginal_lst = []
for x_value in range(1,27):
    for degree in range(10):
        marg = coefs[i]*(i+3)*(x_value**(i+2))
        marginal_lst.append(marg)
```

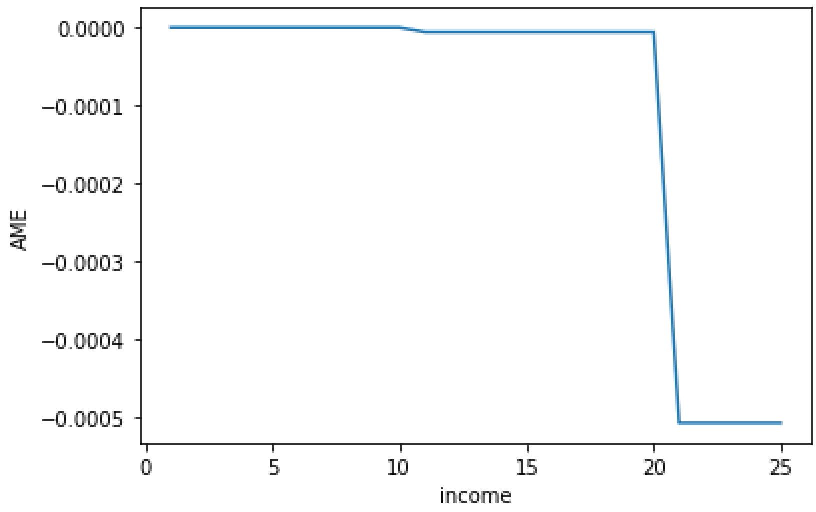
```
10
```

In [659]:

```
marginal_effect = pd.DataFrame(zip(list(range(1,26)),marginal_lst),columns = ['i
ncome','ame'])
plt.plot(list(range(1,26)),marginal_effect.ame)
plt.xlabel('income')
plt.ylabel('AME')
```

Out[659]:

```
Text(0, 0.5, 'AME')
```




# 2. Step function

In [602]:

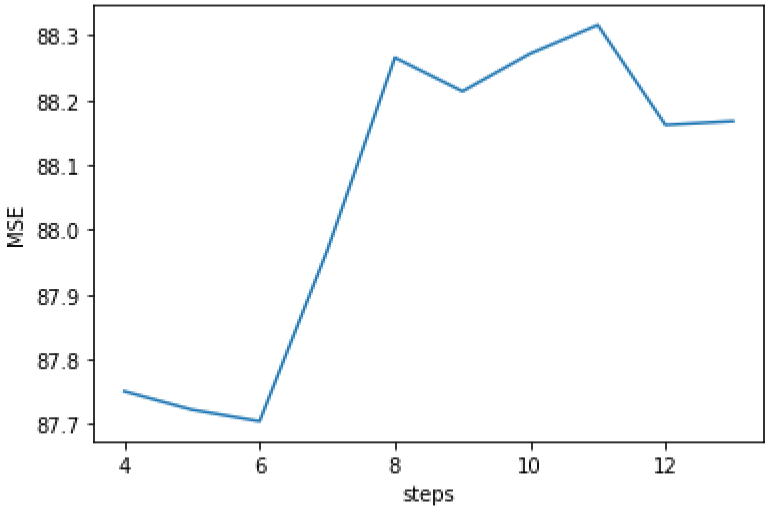
```
def step_function(i):
    df_cut, bins = pd.cut(df_train.income06, i, retbins=True, right=True)
    df_cut.value_counts(sort=False)
    df_steps = pd.concat([df_train.income06, df_cut, df_train.egalit_scale], key
s=['income','age_cuts','egalit'], axis=1)
# Create dummy variables for the age groups
    df_steps_dummies = pd.get_dummies(df_cut)
    df_steps_dummies.head()

    fit3 = sm.GLM(df_steps.egalit, df_steps_dummies).fit()
    fit3.summary().tables[1]

# Binning validation set into same 4 bins
    bin_mapping = np.digitize(df_test.income06, bins)

    X_val = pd.get_dummies(bin_mapping)

# Removing any outliers
    X_val = pd.get_dummies(bin_mapping).drop([1], axis=1)

# Prediction
    pred2 = fit3.predict(X_val)

# Calculating RMSE

    mse = (mean_squared_error(df_test.egalit_scale, pred2))
    return mse,pred2,fit3

mse = 1e03
mse_lst = []
for i in range(4,14):
    mse_lst.append(step_function(i)[0])
    if step_function(i)[0] < mse:
        pred2 = step_function(i)[1]
```

In [605]:

```
plt.plot(list(range(4,14)),mse_lst)
plt.xlabel('steps')
plt.ylabel('MSE')
```

Out[605]:

```
Text(0, 0.5, 'MSE')
```



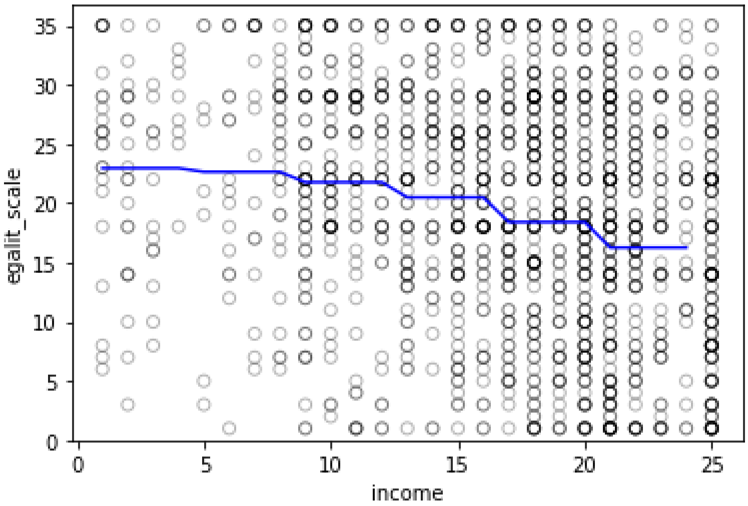

from the graph above, we could tell the step function reaches its best performance when the step equals 6, we could further plot how the prediction fit into the data with the graph below:

In [662]:

```
plt.scatter(df_train.income06, df_train.egalit_scale, facecolor='None', edgecolo
r='k', alpha=0.3)
plt.plot(income_grid, pred2, c='b')

plt.xlabel('income')
plt.ylabel('egalit_scale')
plt.ylim(ymin=0)
```

Out[662]:

```
(0, 36.715176600441495)
```




# 3. Natural regression spline

In [604]:

```
income_grid = np.arange(df_train.income06.min(), df_train.income06.max()).reshap
e(-1,1)
y_cv = list(range(10))
pred = list(range(10))
mse_lst = []
for i in range(10):
    transformer3 = dmatrix(f"cr(df_train.income06, df={i+3})", {"df_train.egalit
_scale": df_train.egalit_scale}, return_type='dataframe')
    model = linear_model.LinearRegression()
    #fit6 = sm.GLM(df_train.egalit_scale, transformed[i]).fit()
    y_cv[i] = cross_val_predict(model,transformer3,df_train.egalit_scale, cv=10)
    mse = mean_squared_error(y_cv[i],df_train.egalit_scale)
    # Specifying 4 degrees of freedom

    #pred = y_cv[i].predict(dmatrix("cr(income_grid, df=4)", {"income_grid": inc
ome_grid}, return_type='dataframe'))
    print(mse)
    mse_lst.append(mse)
```

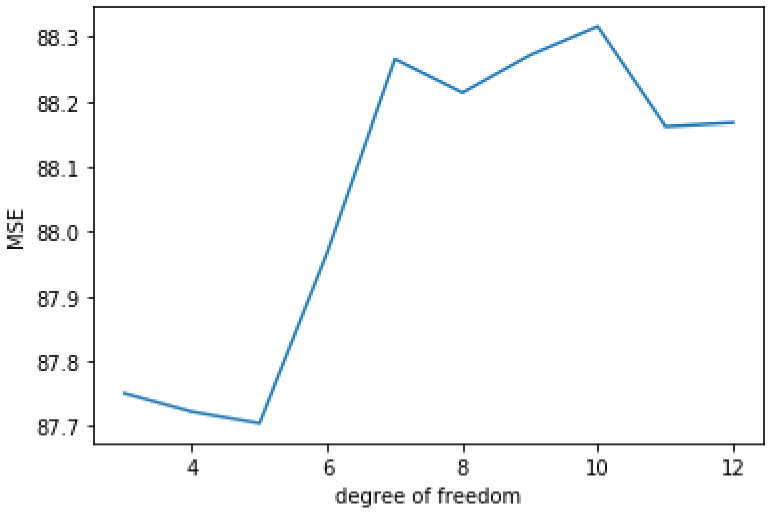
```
87.75082406218515
87.72256583381875
87.70478895827844
87.96801582583197
88.2653040994659
88.21349058168127
88.27155962674566
88.31566413248977
88.16164721614712
88.16748060440428
```

In [607]:

```
plt.plot(list(range(3,13)),mse_lst)
plt.xlabel('degree of freedom')
plt.ylabel('MSE')
```

Out[607]:

```
Text(0, 0.5, 'MSE')
```



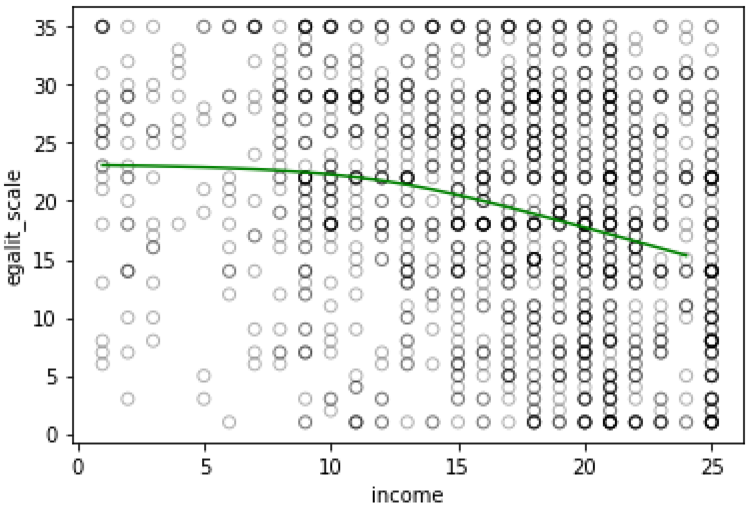
From the graph above we could tell the optimal degree of freedom is 5, and we could plot how this best model fit into the data as follows:

In [458]:

```
transformed_best = dmatrix("cr(df_train.income06, df=5)", {"df_train.egalit_scal
e": df_train.egalit_scale}, return_type='dataframe')
best = sm.GLM(df_train.egalit_scale, transformed_best).fit()
pred = best.predict(dmatrix("cr(income_grid, df=5)", {"income_grid": income_grid
}, return_type='dataframe'))
```

In [459]:

```
plt.scatter(df_train.income06, df_train.egalit_scale, facecolor='None', edgecolo
r='k', alpha=0.3)
#plt.plot(income_grid, pred4, color='b', label='Specifying three knots')
#plt.plot(income_grid, pred5, color='r', label='Specifying df=6')
plt.plot(income_grid, pred, color='g', label='Natural spline df=5')
plt.xlabel('income')
plt.ylabel('egalit_scale');
```




# 4. Egalit and Everything

In [665]:

```
#pre-processing
mapping = {'No': 0, 'Yes': 1,'YES':1,'NO':0,'NOT ALLOWED':0,'ALLOWED':1,'Never':
0,'<Once/yr':1,'Sev times/yr':2,'>Once/wk':4,'Every wk':3,'2-3 times /mo':5}
df_train_processed = df_train.replace({'black': mapping, 'born': mapping,'colat
h':mapping,'attend':mapping,'colmil':mapping,'colrac':mapping})
print(X_train.shape,Y_train.shape,X_test.shape,Y_test.shape)
df_train._get_numeric_data()
```

```
(1481, 1) (1481, 1) (493, 1) (493, 1)
```

Out[665]:

| | age | authoritarianism | childs | con_govt | egalit_scale | income06 | science_quiz | sibs | soc |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 21 | 4 | 0 | 4 | 22 | 25 | 7 | 2 | |
| **1** | 42 | 4 | 2 | 2 | 14 | 23 | 10 | 1 | |
| **2** | 70 | 1 | 3 | 4 | 20 | 19 | 4 | 0 | |
| **3** | 35 | 2 | 2 | 2 | 34 | 16 | 7 | 2 | |
| **4** | 24 | 6 | 3 | 3 | 35 | 5 | 5 | 2 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **1476** | 61 | 6 | 0 | 3 | 18 | 12 | 6 | 3 | |
| **1477** | 53 | 6 | 0 | 2 | 29 | 1 | 7 | 9 | |
| **1478** | 48 | 3 | 2 | 3 | 13 | 22 | 5 | 2 | |
| **1479** | 37 | 1 | 8 | 4 | 22 | 12 | 5 | 3 | |
| **1480** | 22 | 0 | 0 | 4 | 25 | 1 | 7 | 3 | |

1481 rows × 12 columns

In [666]:

```
df_train._get_numeric_data()
X_train, Y_train = df_train._get_numeric_data().loc[:, df_train._get_numeric_dat
a().columns != 'egalit_scale'],df_train['egalit_scale']
X_test, Y_test = df_test._get_numeric_data().loc[:, df_test._get_numeric_data().
columns != 'egalit_scale'],df_test['egalit_scale']
```

## 4.a. linear regression

In [620]:

```
#np.linalg.lstsq(X_train, Y_train)
from sklearn.model_selection import cross_val_predict
from sklearn.linear_model import LinearRegression
model = LinearRegression()
y_cv = cross_val_predict(model, X_train, Y_train, cv=10)
mse = mean_squared_error(Y_train,y_cv)
print(mse)
```

```
84.2047125318786
```

## 4.b.Elastic Net

In [621]:

```
from sklearn.linear_model import ElasticNetCV
elas = ElasticNetCV(cv= 10,alphas=np.linspace(0.1,1,10),l1_ratio = np.linspace(
0.1,1,10)).fit(X_train,Y_train)
elas_mse = mean_squared_error(Y_test,elas.predict(X_test))
non_zero_coef = [item for item in elas.coef_ if item != 0]
print(elas_mse)
print(len(non_zero_coef))
print(elas.l1_ratio_)
print(elas.alpha_)
```

```
84.6188735393559
9
0.7000000000000001
0.2
```

## 4.c. Principal component regression

In [694]:

```
from sklearn.decomposition import PCA
from scipy.signal import savgol_filter
mse_lst = []
for i in range(1,10):
# Define the PCA object
    pca = PCA()
    Xstd = StandardScaler().fit_transform(X_train)
    Xreg = pca.fit_transform(Xstd)[:,:i]
    regr = linear_model.LinearRegression()
# Cross-validation
    y_cv = cross_val_predict(regr, Xreg, Y_train, cv=10)
    mse_cv = mean_squared_error(Y_train, y_cv)
    mse_lst.append(mse_cv)
print(min(mse_lst))
print(mse_lst)
```

```
84.82197955019484
[91.98923486938209, 87.83775946759667, 87.16683979229317, 85.4804102
851862, 85.32679742189549, 85.20718409020952, 84.82197955019484, 84.
98099838260612, 85.18324583250048]
```

### 4.d. PLSRegression

In [697]:

```
from sklearn.cross_decomposition import PLSRegression
mse_lst = []
for i in range(1,10):
    pls2 = PLSRegression(n_components=i)
    y_cv = cross_val_predict(pls2, X_train, Y_train, cv=10)
    mse_cv = mean_squared_error(Y_train, y_cv)
    mse_lst.append(mse_cv)
print(min(mse_lst))
print(mse_lst)
```

```
84.20466526053394
[85.45331479113527, 84.33212802978161, 84.26560041144486, 84.2269319
2791266, 84.2064550819548, 84.20475328168419, 84.20466526053394, 84.
20469605824997, 84.2047099997479]
```

## 5. Feature Interaction plot

In [673]:

```
print(X_train.head(1))
```

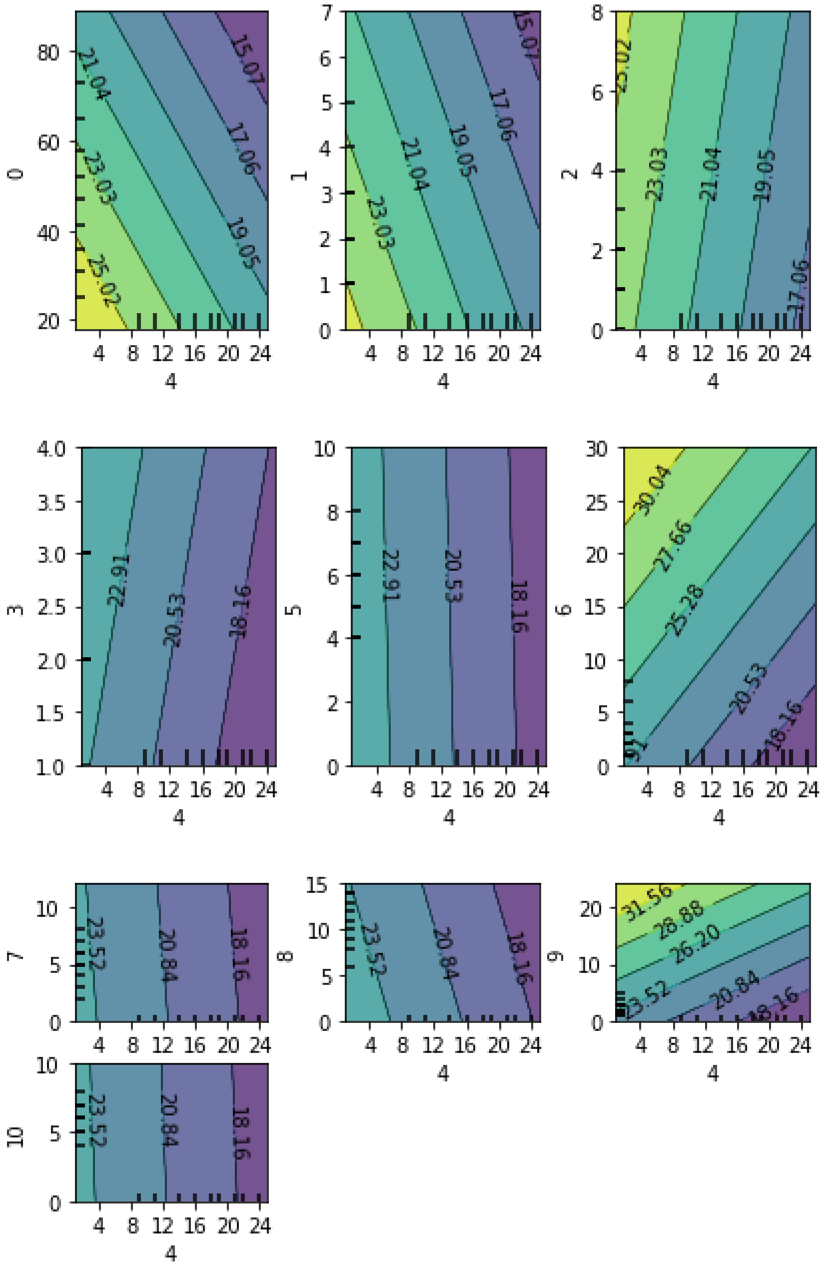
```
   age  authoritarianism  childs  con_govt  income06  science_quiz
sibs  \
0   21                 4       0         4        25             7
2

   social_connect  tolerance  tvhours  wordsum
0               5         10        3        5
```

In [671]:

```
features = []
for i in range(X_train.shape[1]):
    features.append((4,i))
```

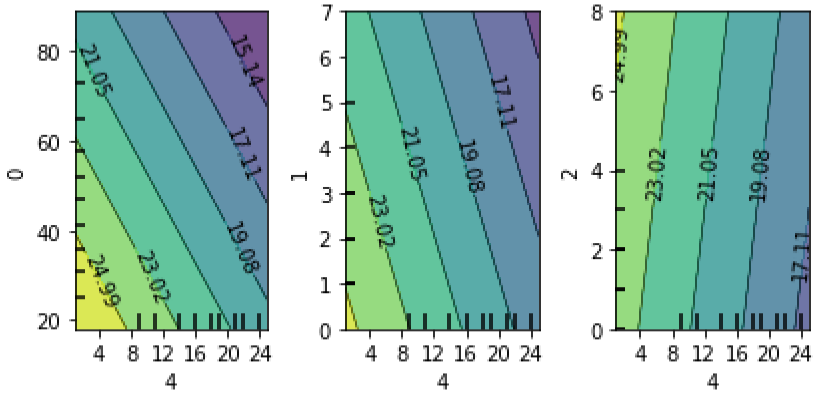
In [678]:

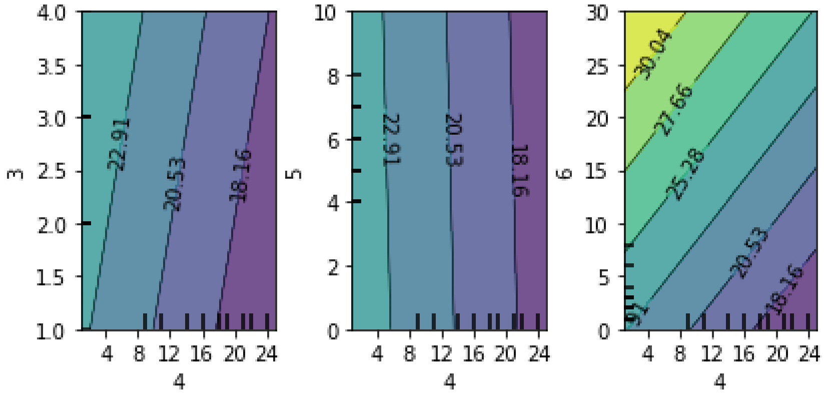
```
model = LinearRegression().fit( X_train, Y_train)
plot_partial_dependence(model,X_train,[(4,0),(4,1),(4,2)])
plot_partial_dependence(model,X_train,[(4,3),(4,5),(4,6)])
plot_partial_dependence(model,X_train,[(4,7),(4,8),(4,9),(4,10)])
```

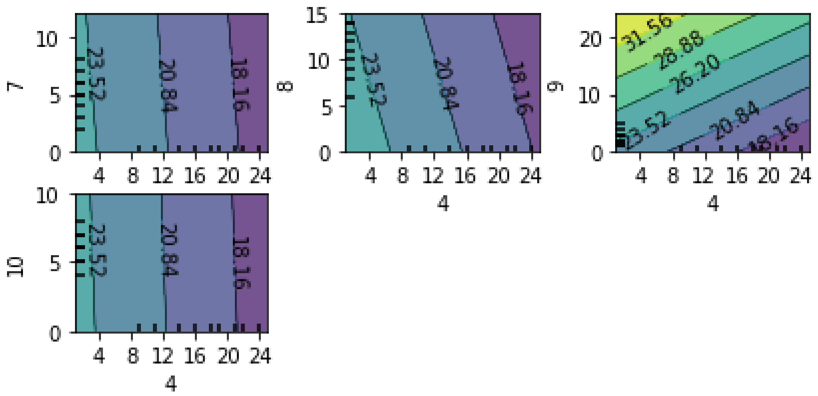


In [681]:

```
elas = ElasticNetCV(cv= 10,alphas=np.linspace(0.1,1,10),l1_ratio = np.linspace(
0.1,1,10)).fit(X_train,Y_train)
plot_partial_dependence(elas,X_train,[(4,0),(4,1),(4,2)])
plot_partial_dependence(model,X_train,[(4,3),(4,5),(4,6)])
plot_partial_dependence(model,X_train,[(4,7),(4,8),(4,9),(4,10)])
```

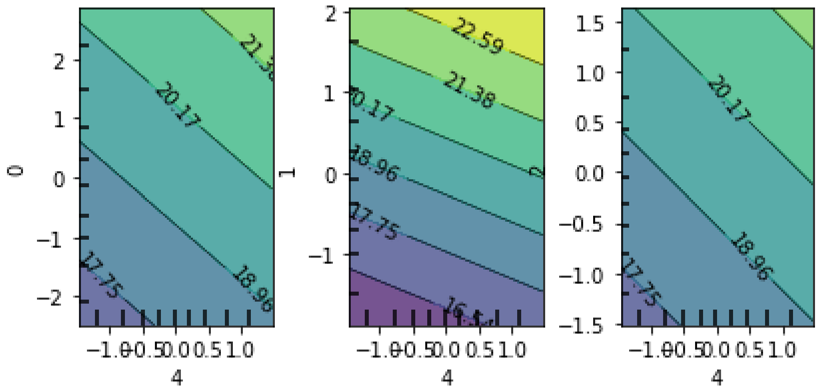


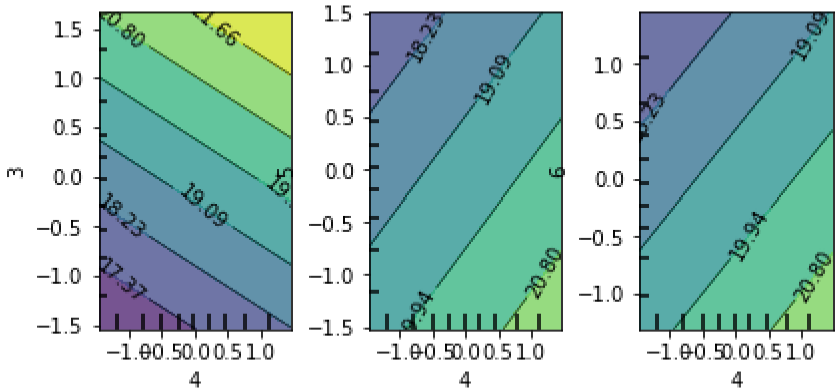



In [696]:

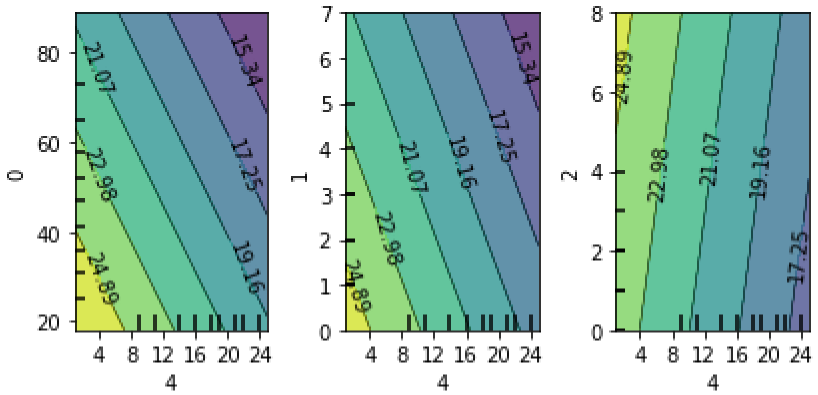
```
pca = PCA()
Xstd = StandardScaler().fit_transform(X_train)
Xreg = pca.fit_transform(Xstd)[:,:7]
regr = linear_model.LinearRegression()
regr = regr.fit(Xreg,Y_train)
plot_partial_dependence(regr,Xreg,[(4,0),(4,1),(4,2)])
plot_partial_dependence(regr,Xreg,[(4,3),(4,5),(4,6)])
```

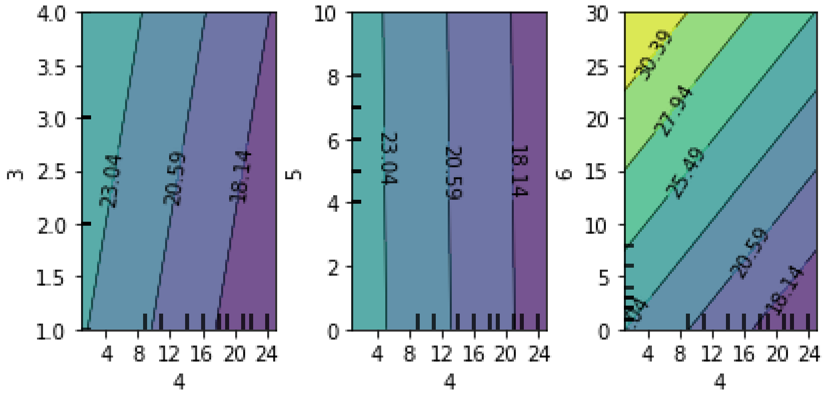


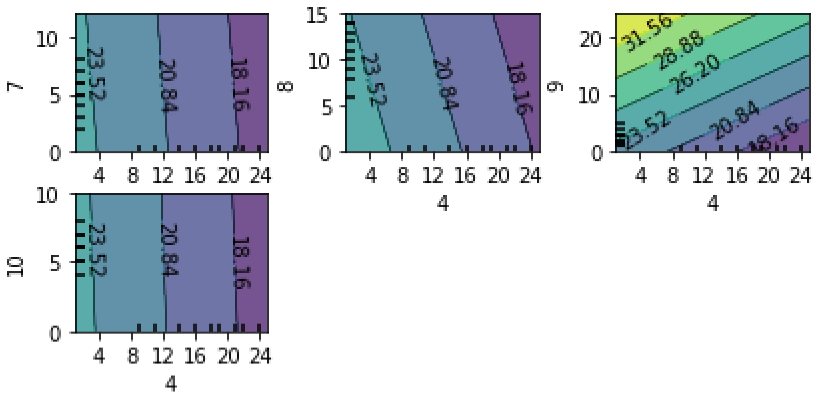


In [699]:

```
pls2 = PLSRegression(n_components=3)
pls2.fit(X_train, Y_train)
plot_partial_dependence(pls2,X_train,[(4,0),(4,1),(4,2)])
plot_partial_dependence(pls2,X_train,[(4,3),(4,5),(4,6)])
plot_partial_dependence(model,X_train,[(4,7),(4,8),(4,9),(4,10)])
```




Across these models, the age, authoritarianism, sibs, and tvhours are important features. For principal component regression, it is relatively hard to tell which features are important because these features have between compressed through PCA. For the other three models, age and authoritarianism are negatively correlated while tv hours and sibs are positively correlated. it is surprising that some features such as social connect and science quiz score do not contribute too much.