Xiong_Yinjiang_HW4

February 16, 2020

[27]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
import seaborn as sb
from patsy import dmatrix
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import ElasticNet
from sklearn.decomposition import PCA
from sklearn.cross_decomposition import PLSRegression
from sklearn.preprocessing import MinMaxScaler
from mlxtend.evaluate import feature_importance_permutation
from sklearn.inspection import plot_partial_dependence
```

1.Perform polynomial regression to predict egalit_scale as a function of income06. Use and plot 10-fold cross-validation to select the optimal degree d for the polynomial based on the MSE. Plot the resulting polynomial fit to the data, and also graph the average marginal effect (AME) of income06 across its potential values. Be sure to provide substantive interpretation of the results.

[10]:
```
gss_train = pd.read_csv('gss_train.csv')
```

[160]:
```
gss_train.head(5)
```

[160]:
```
   age       attend  authoritarianism black born  childs       colath  \
0   21        Never                 4    No  YES       0  NOT ALLOWED
1   42        Never                 4    No  YES       2      ALLOWED
2   70     <Once/yr                 1   Yes  YES       3      ALLOWED
3   35  Sev times/yr                2    No  YES       2      ALLOWED
4   24  Sev times/yr                6    No   NO       3  NOT ALLOWED

        colrac     colcom      colmil  ... social_connect social_cons3  \
0  NOT ALLOWED      FIRED  NOT ALLOWED  ...              5          Mod
1  NOT ALLOWED  NOT FIRED      ALLOWED  ...              5      Liberal
2  NOT ALLOWED  NOT FIRED      ALLOWED  ...              5      Liberal
3  NOT ALLOWED      FIRED  NOT ALLOWED  ...             10      Liberal
4  NOT ALLOWED      FIRED      ALLOWED  ...              4          Mod
```

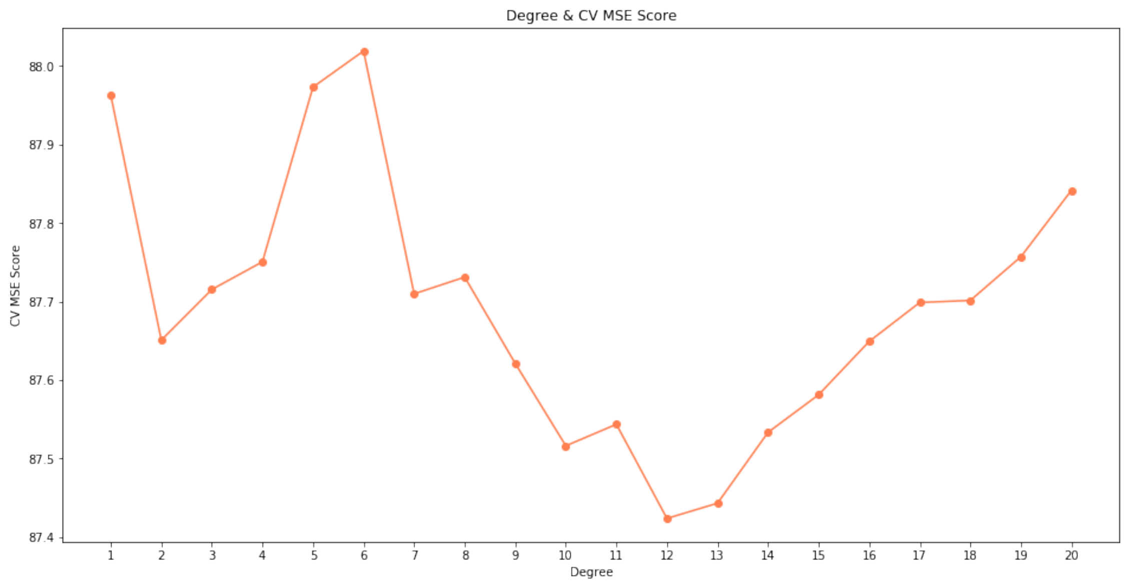
```
      south   spend3           teensex tolerance tvhours vetyears wordsum  \
0  Nonsouth  Conserv      ALWAYS WRONG        10       3     NONE       5
1  Nonsouth      Mod  NOT WRONG AT ALL        13       3     NONE       6
2  Nonsouth  Conserv      ALWAYS WRONG        10       3     NONE       6
3  Nonsouth  Liberal      ALWAYS WRONG        11       3     NONE       6
4  Nonsouth  Conserv  ALMST ALWAYS WRG         7       2     NONE       4

    zodiac
0    ARIES
1    ARIES
2   TAURUS
3  SCORPIO
4  SCORPIO

[5 rows x 45 columns]
```

[225]:

```
# a function to generate mse
def polyreg(n=1, var=gss_train['income06'], y=gss_train['egalit_scale']):
    x = pd.DataFrame()
    for degree in range(n+1):
        if degree == 0:
            continue
        else:
            x_add = pd.DataFrame(data={'x^{}'.format(degree):var**degree})
            x = pd.concat([x, x_add], axis=1)
    lm = LinearRegression()
    return np.mean(-cross_val_score(lm, x, y, cv=10,␣
 ↪scoring='neg_mean_squared_error'))
    #return x
```

[226]:

```
# plot mse for degree 1 to 10
plt.figure(figsize=(16,8))
d =␣
 ↪['1','2','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19','20']
mse = []
for n in range(1,21):
    mse.append(polyreg(n=n))
plt.plot(d, mse, marker='o', color='coral', label='MSE')
plt.xlabel('Degree')
plt.ylabel('CV MSE Score')
plt.title('Degree & CV MSE Score');
```



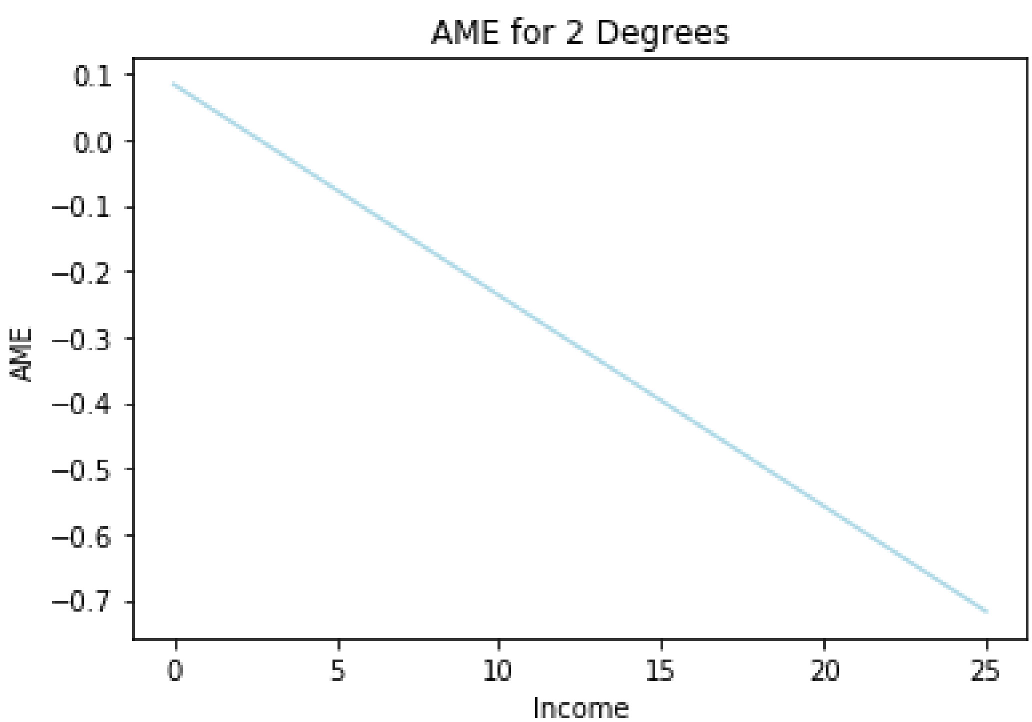
Degree 12 gives the lowest MSE in 10-fold cross validations. Degree 2 has a local minimum MSE when degree < 9.

[250]:
```
# AME: since python doesn't have AME for continuous variables, the following␣
↪function allows calculation for ame
# at different degrees
def polyreg_ame(n=1, var=gss_train['income06'], y=gss_train['egalit_scale']):
    x = pd.DataFrame()
    x_der = pd.DataFrame()
    ame_total = []
    for degree in range(n+1):
        if degree == 0:
            continue
        else:
            x_add = pd.DataFrame(data={'x^{}'.format(degree):var**degree})
            x = pd.concat([x, x_add], axis=1)
    lm = LinearRegression().fit(x, y)
    coef = lm.coef_
    for degree in range(n):
        x_der_add = pd.DataFrame(data={'x^{}'.format(degree): (var**degree) *␣
↪coef[degree] * (degree+1)})
        ame_var = np.mean(np.array(x_der_add))
        x_der = pd.concat([x_der, x_der_add], axis=1)
        ame_total.append(ame_var)
        ame = np.sum(ame_total)
    return ame, coef
```

[284]:
```
def plot_ame(coef):
    x = np.linspace(0, 25, 256, endpoint = True)
    y = 0
    for n in range(len(coef)):
        coef[n] = coef[n] * (n+1)
    for n in range(len(coef)):
        y += (x ** n) * coef[n]
    plt.plot(x, y, 'lightblue')
    plt.title('AME for {} Degrees'.format(len(coef)))
    plt.xlabel('Income')
    plt.ylabel('AME')
```

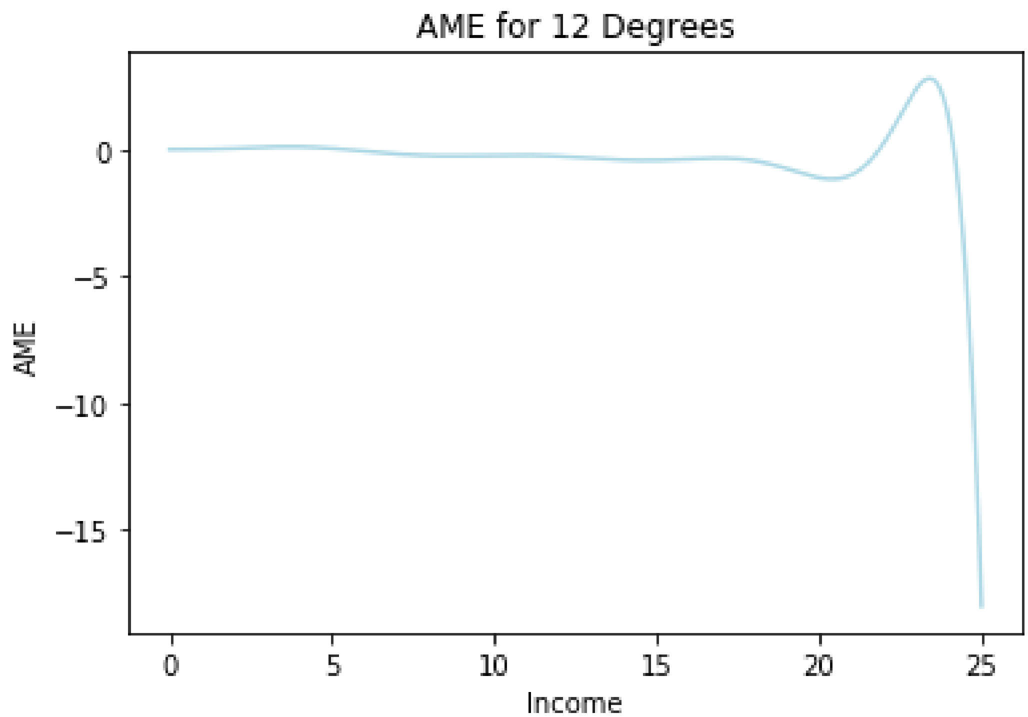
[287]:
```
# Here we examine the AMEs and its graphs at degree = 2 and 12
# 2 is an interpretable degree with the lowest CV MSE
# 12 is the global minimum CV MSE degree
print('The AME for 2 degree polynomial is', polyreg_ame(n=2)[0])
print('The graph for AME:')
plot_ame(polyreg_ame(n=2)[1])
```

```
The AME for 2 degree polynomial is -0.4507318899384385
The graph for AME:
```



```
[314]: print('The AME for 12 degree polynomial is', polyreg_ame(n=12)[0])
print('The graph for AME:')
plot_ame(polyreg_ame(n=12)[1])
```

```
The AME for 12 degree polynomial is -1.8378588879677409
The graph for AME:
```



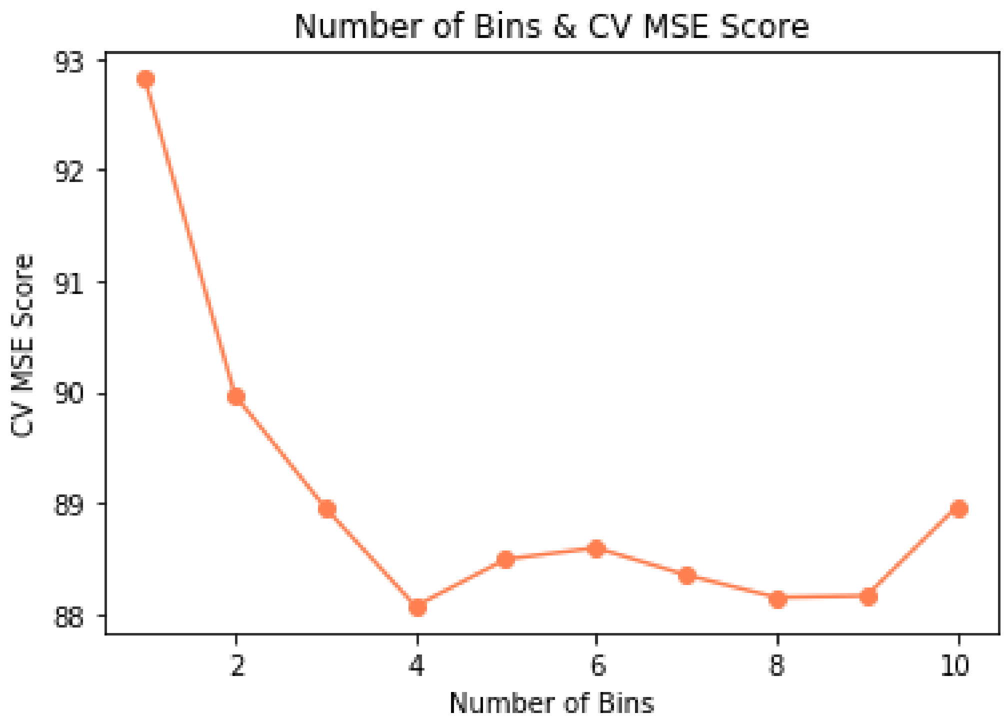
As we know the AME is the partial derivative average, we expect the AME graph for 2 degrees to be linear (2-1) and 12 degrees to be a 11-degree polynomial. The AME is the average of all the observed data.

2.Fit a step function to predict egalit_scale as a function of income06, and perform 10-fold cross-validation to choose the optimal number of cuts. Plot the fit and interpret the results.

```
[73]: # a function to generate mse
def step(bin=4, x=gss_train['income06'], y=gss_train['egalit_scale']):
    df_cut = pd.cut(x, bin)
    df_steps_dummies = pd.get_dummies(df_cut)
    lm = LinearRegression()
    return np.mean(-cross_val_score(lm, df_steps_dummies, y, cv=10,␣
 ↪scoring='neg_mean_squared_error'))
```

```
[74]: # plot mse for 1 to 10 bins
bin = []
```

```
mse = []
for n in range(1,11):
    bin.append(n)
    mse.append(step(bin=n))
plt.plot(bin, mse, marker='o', color='coral', label='MSE')
plt.xlabel('Number of Bins')
plt.ylabel('CV MSE Score')
plt.title('Number of Bins & CV MSE Score');
```

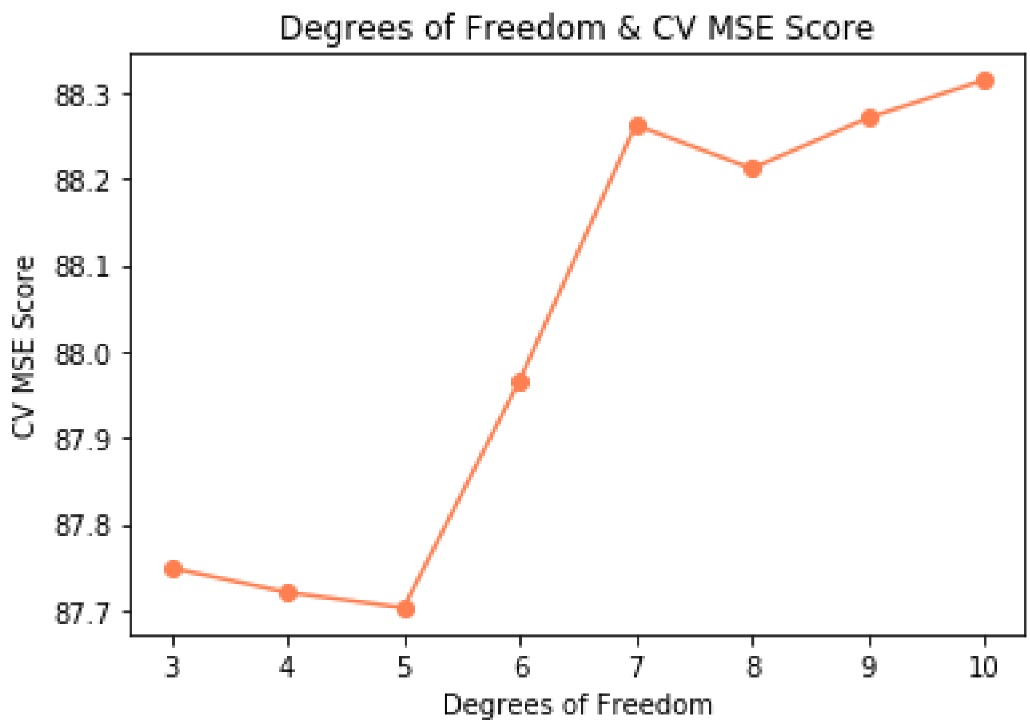


The graph indicates that 4 bins (3 cuts) yield the optimal MSE. More than 4 bins tend to overfit the model.

3.Fit a natural regression spline to predict egalit_scale as a function of income06. Use 10-fold cross-validation to select the optimal number of degrees of freedom, and present the results of the optimal model.

[105]:
```
# a fuction to generate mse for cubic natural spline
def natural_spline(df=4, x=gss_train['income06'], y=gss_train['egalit_scale']):
    natural_sp = dmatrix("cr(x,df = {})".format(df), {"x": x},␣
 ↪return_type='dataframe')
    lm = LinearRegression()
    return np.mean(-cross_val_score(lm, natural_sp, y, cv=10,␣
 ↪scoring='neg_mean_squared_error'))
```

```
[119]: # plot the degrees of freedom
# note that since the cubic polynomial has been predetermined, the degrees of␣
 ↪freedom only reflect the number of knots + 1
df = []
mse = []
for n in range(3,11):
    df.append(n)
    mse.append(natural_spline(df=n))
plt.plot(df, mse, marker='o', color='coral', label='MSE')
plt.xlabel('Degrees of Freedom')
plt.ylabel('CV MSE Score')
plt.title('Degrees of Freedom & CV MSE Score');
```



5 degrees of freedom (4 knots) yield the best MSE in 10-fold cross validation.

4.Estimate the following models using all the available predictors (be sure to perform appropriate data pre-processing (e.g., feature standardization) and hyperparameter tuning (e.g. lambda for PCR/PLS, lambda and alpha for elastic net). Also use 10-fold cross-validation for each model to estimate the model's performance using MSE):
a. Linear regression

b. Elastic net regression

c. Principal component regression

d. Partial least squares regression

```
[11]: # Data Preprocessing
scaler = MinMaxScaler(feature_range=(0,1))
def standardize (df):
    new_df = pd.DataFrame()
    for predictor in df:
        if df[predictor].dtypes == 'int64':
            column = df[predictor].values.reshape(-1,1)
            scaler.fit(column)
            new_df[predictor] = scaler.transform(column).reshape(1,-1)[0]
    return new_df
```

```
[12]: def dummies (df, new):
    for predictor in df:
        if df[predictor].dtypes == object:
            dum = pd.get_dummies(df[predictor])
            dum = dum.drop(dum.columns[0], axis=1)
            new = pd.concat([new, dum], axis=1)
    return new
```

```
[13]: gss_train_clean = dummies(gss_train, standardize(gss_train))
X = gss_train_clean.drop(['egalit_scale'], axis=1)
y = gss_train_clean['egalit_scale']
```

```
[14]: X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1481 entries, 0 to 1480
Columns: 102 entries, age to VIRGO
dtypes: float64(11), uint8(91)
memory usage: 259.0 KB
```

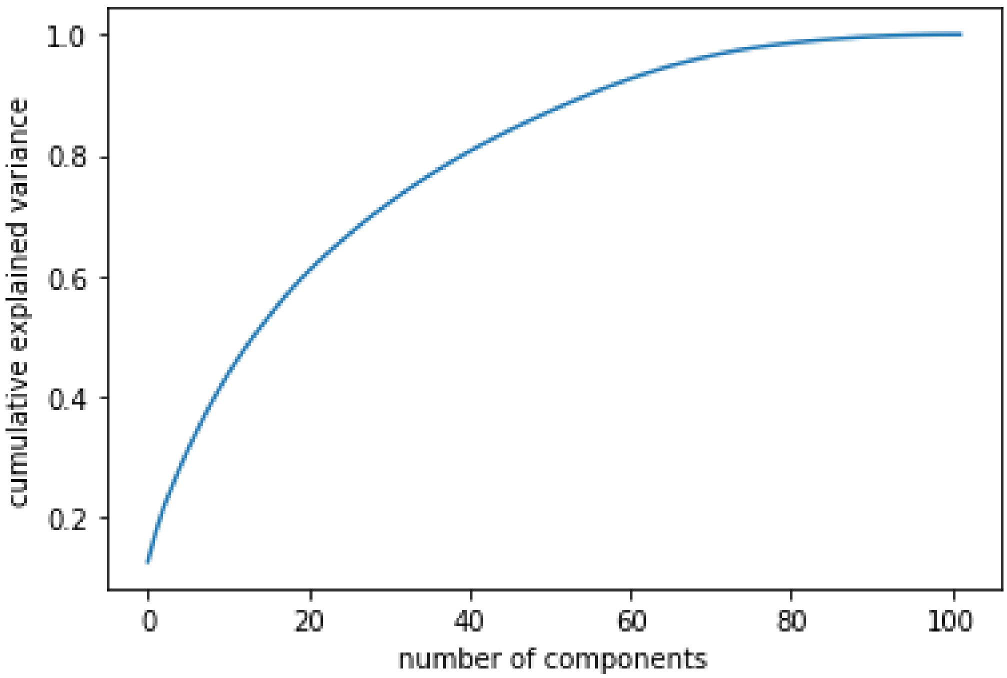
```
[15]: # Linear regression
lm = LinearRegression()
print('The 10-fold CV MSE in linear model is', np.mean(-cross_val_score(lm, X,␣
 ↪y, cv=10, scoring='neg_mean_squared_error')))
```

```
The 10-fold CV MSE in linear model is 0.05520732886554379
```

```
[16]: # Elastic net regression
en = ElasticNet(l1_ratio=0.1, alpha=0.01)
# hyperparameter defined as l1_ratio=0.1 and alpha=0.01
print('The 10-fold CV MSE in elastic net is', np.mean(-cross_val_score(en, X, y,␣
 ↪cv=10, scoring='neg_mean_squared_error')))
```

```
The 10-fold CV MSE in elastic net is 0.052820415279701785
```

[17]:
```
# to select the hyperparameter, we plot the cumulative explained variance
pca = PCA().fit(X)
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance');
# since we do not care about visualization, from the graph, we determine 40 as␣
↪the number of components,
# which explains about 80% of the variance in x
# thus, we reduce the dimensions from more than 102 to 40
```



[42]:
```
# PCA
pca = PCA(n_components=40)
principalComponents = pca.fit_transform(X)
print('The 10-fold CV MSE in PCA is', np.mean(-cross_val_score(lm,␣
↪principalComponents, y, cv=10, scoring='neg_mean_squared_error')))
```

```
The 10-fold CV MSE in PCA is 0.05555539622353064
```

[19]:
```
# PLS: we use the same number of components
pls = PLSRegression(n_components=40)
print('The 10-fold CV MSE in PCA is', np.mean(-cross_val_score(pls, X, y, cv=10,␣
↪scoring='neg_mean_squared_error')))
```

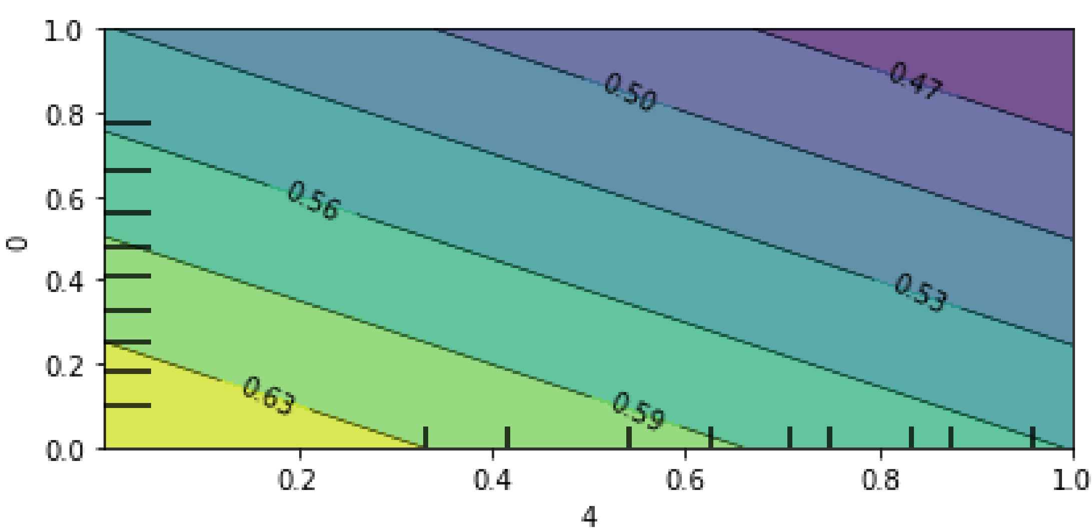
```
The 10-fold CV MSE in PCA is 0.05520741571936818
```

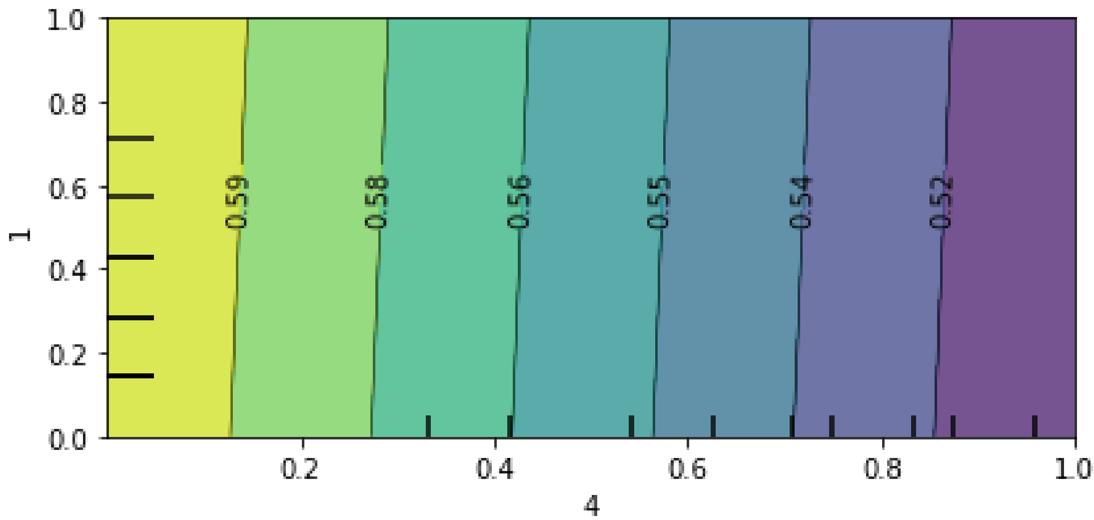
5.For each final tuned version of each model fit, evaluate feature importance by generating feature interaction plots. Upon visual presentation, be sure to discuss the substantive results for these models and in comparison to each other (e.g., talk about feature importance, conditional effects, how these are ranked differently across different models, etc.).

[36]:
```
# income06 is at index 4
X.head(2)
```

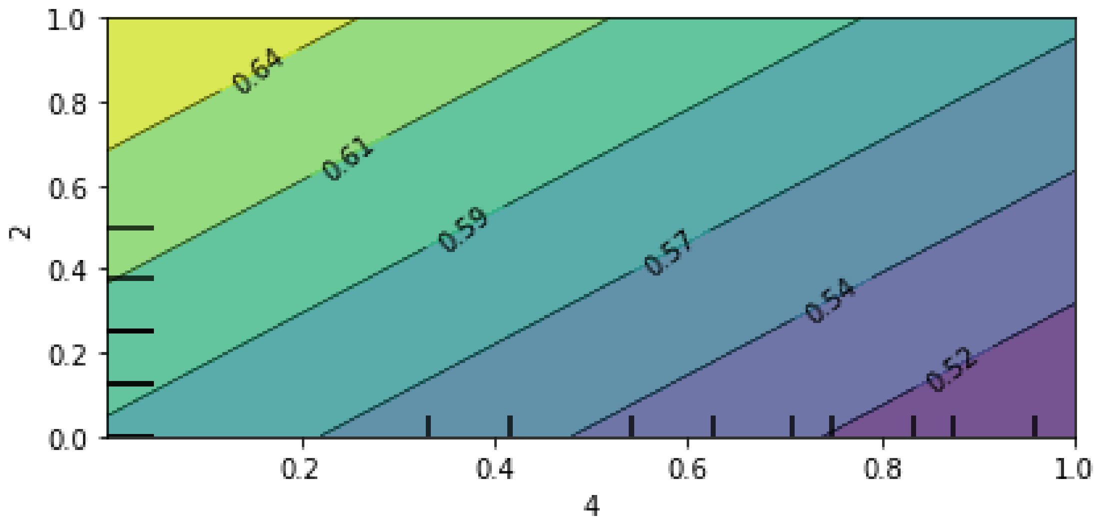
[36]:
```
        age  authoritarianism  childs  con_govt  income06  science_quiz  \
0  0.042254          0.571429    0.00  1.000000  1.000000           0.7
1  0.338028          0.571429    0.25  0.333333  0.916667           1.0

       sibs  social_connect  tolerance  tvhours  ...  CANCER  CAPRICORN  \
0  0.066667        0.416667   0.666667    0.125  ...       0          0
1  0.033333        0.416667   0.866667    0.125  ...       0          0

   GEMINI  LEO  LIBRA  PISCES  SAGITTARIUS  SCORPIO  TAURUS  VIRGO
0       0    0      0       0            0        0       0      0
1       0    0      0       0            0        0       0      0

[2 rows x 102 columns]
```

[37]:
```
# plot the partial dependence between income06 and age(0), authoritarianism(1),␣
↪childs(2), con_govt(3), science_quiz(5)
# sibs(6) and social_connect(7) in linear model
lm.fit(X,y)
plot_partial_dependence(lm, X, [(4,0)])
plot_partial_dependence(lm, X, [(4,1)])
plot_partial_dependence(lm, X, [(4,2)])
plot_partial_dependence(lm, X, [(4,3)])
plot_partial_dependence(lm, X, [(4,5)])
plot_partial_dependence(lm, X, [(4,6)])
plot_partial_dependence(lm, X, [(4,7)])
```

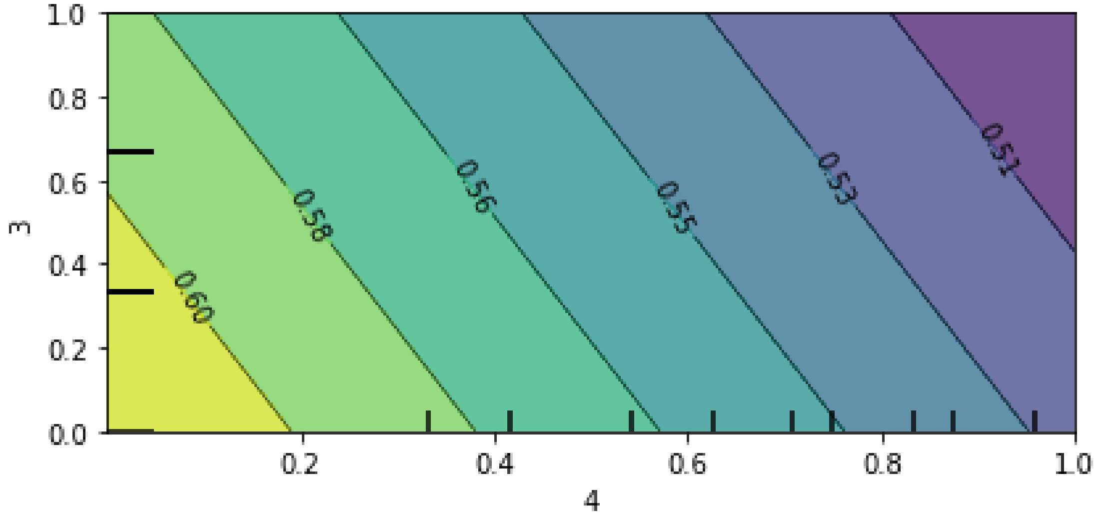


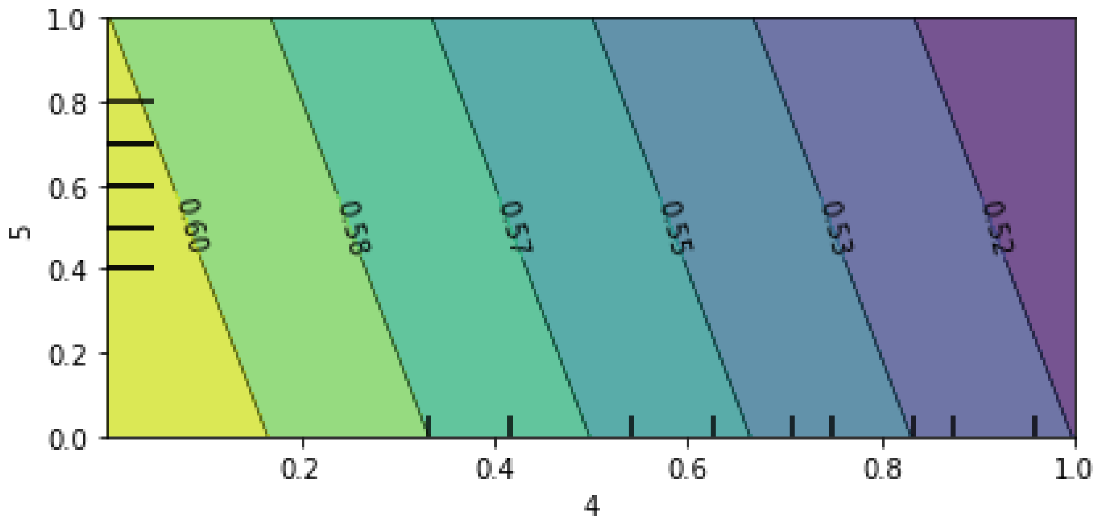


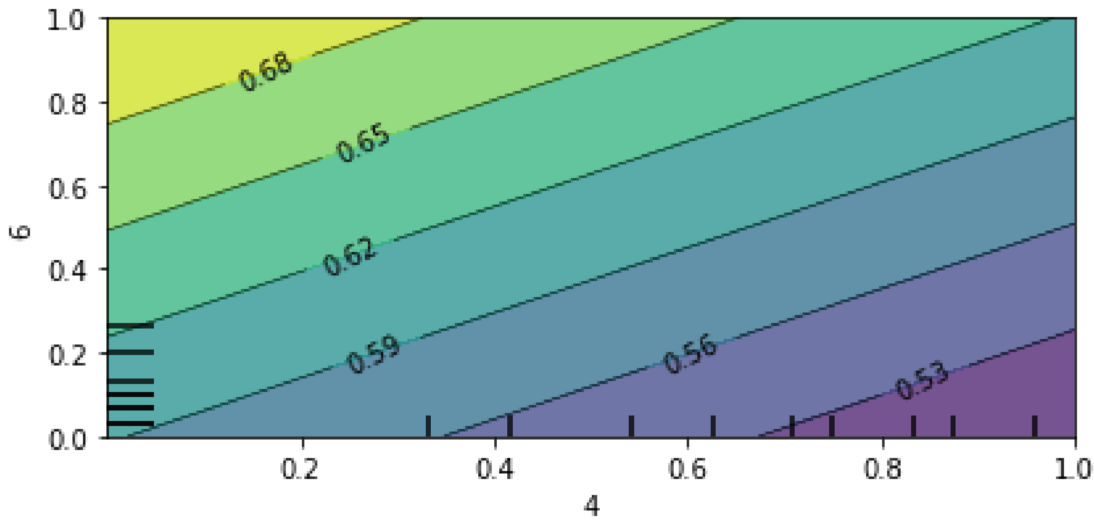
1.0
0.8
0.6
0.4
0.2
0.0
1
0.59
0.58
0.56
0.55
0.54
0.52
0.2
0.4
0.6
0.8
1.0
4

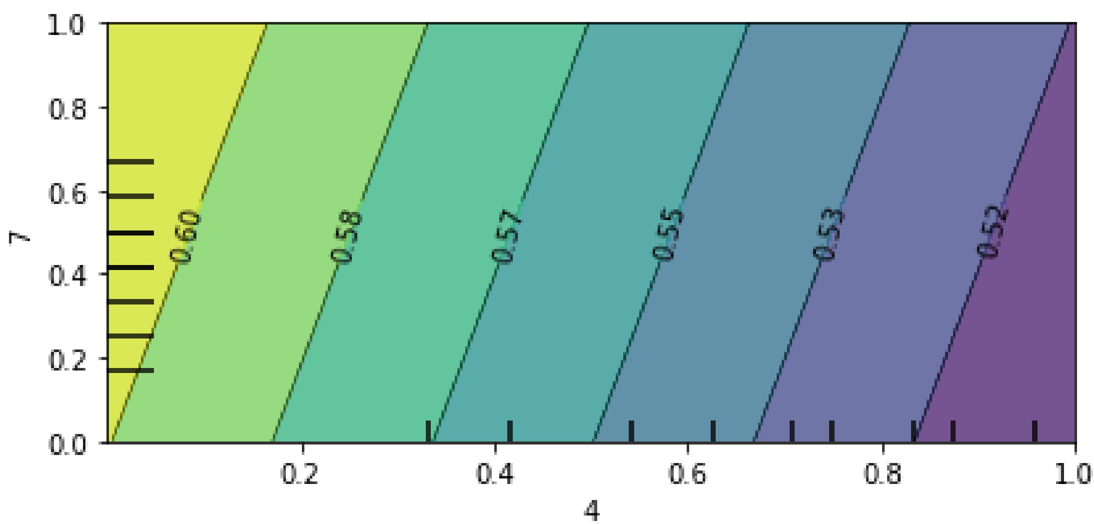


1.0
0.8
0.6
0.4
0.2
0.0
2
0.64
0.61
0.59
0.57
0.54
0.52
0.2
0.4
0.6
0.8
1.0
4

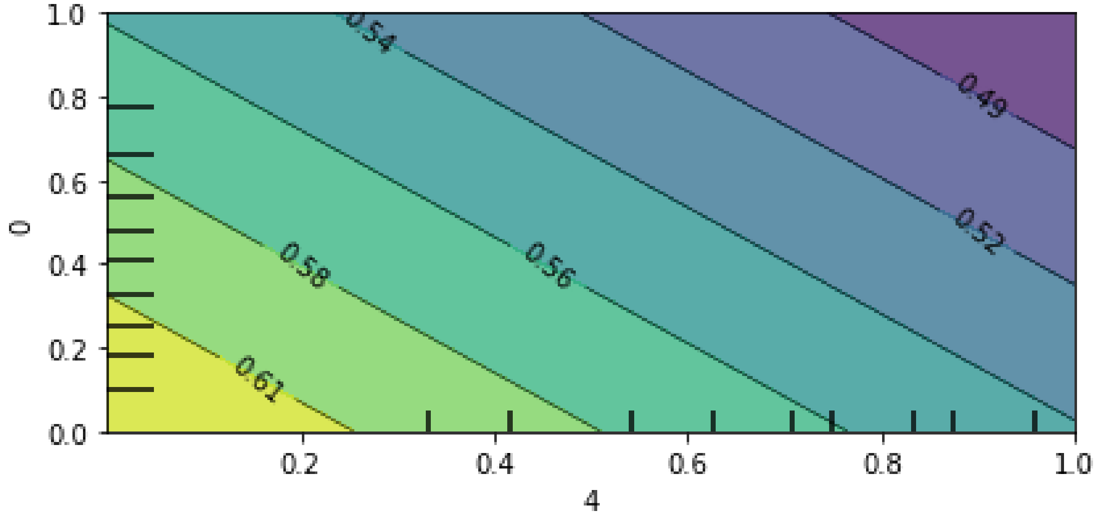

0.60
0.58
0.56
0.55
0.53
0.51
1.0
0.8
0.6
0.4
0.2
0.0
3
0.2
0.4
0.6
0.8
1.0
4

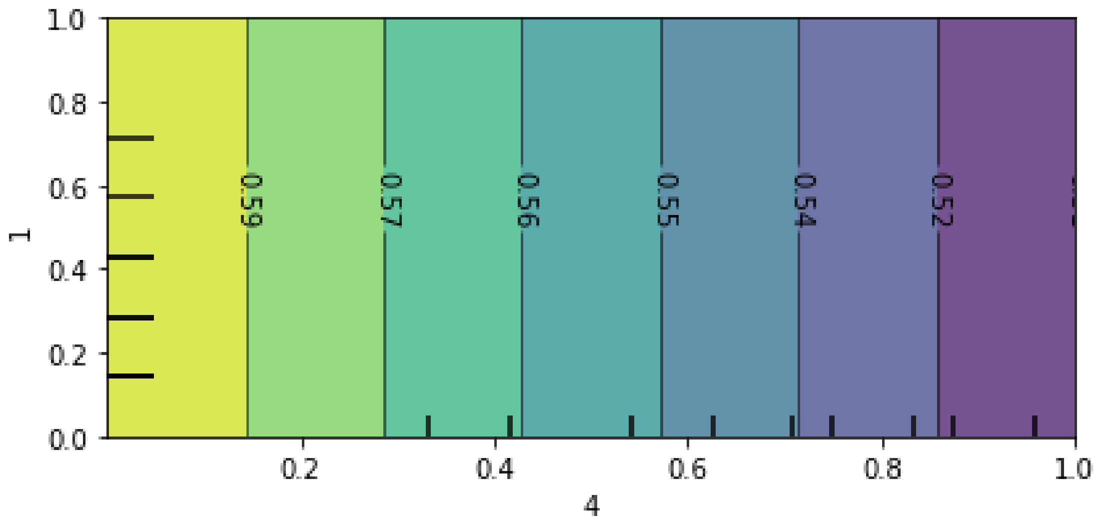

0.60
0.58
0.57
0.55
0.53
0.52
1.0
0.8
0.6
0.4
0.2
0.0
5
0.2
0.4
0.6
0.8
1.0
4

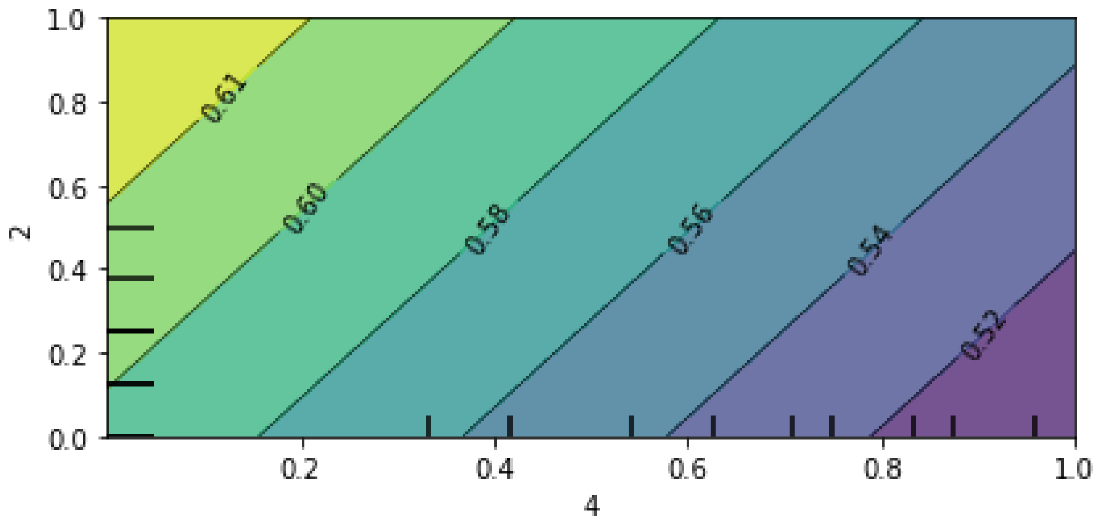


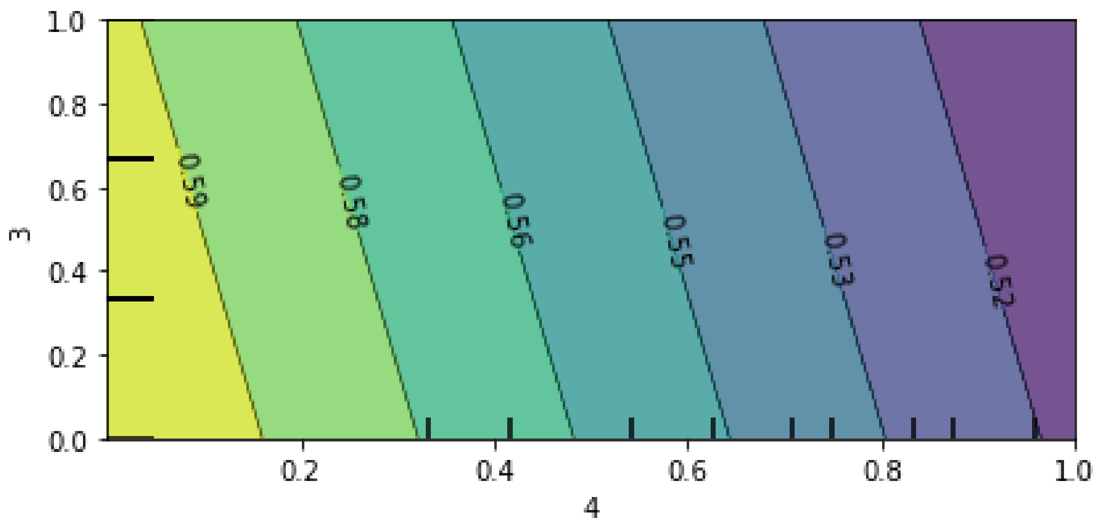


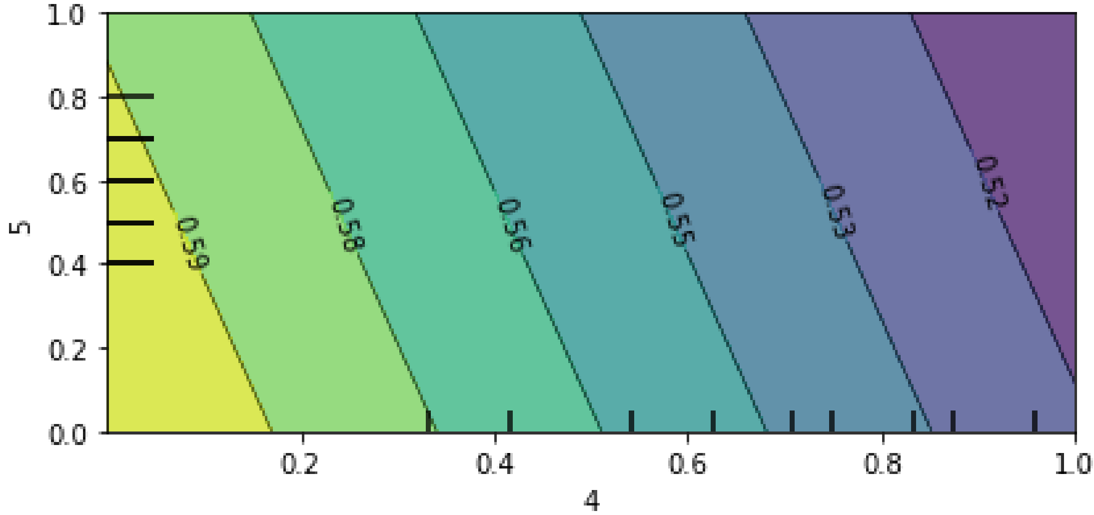
[45]:
```
# plot the partial dependence between income06 and age(0), authoritarianism(1),␣
↪childs(2), con_govt(3), science_quiz(5)
# sibs(6) and social_connect(7) in elastic net model
en.fit(X, y)
plot_partial_dependence(en, X, [(4,0)])
plot_partial_dependence(en, X, [(4,1)])
plot_partial_dependence(en, X, [(4,2)])
plot_partial_dependence(en, X, [(4,3)])
plot_partial_dependence(en, X, [(4,5)])
plot_partial_dependence(en, X, [(4,6)])
plot_partial_dependence(en, X, [(4,7)])
```

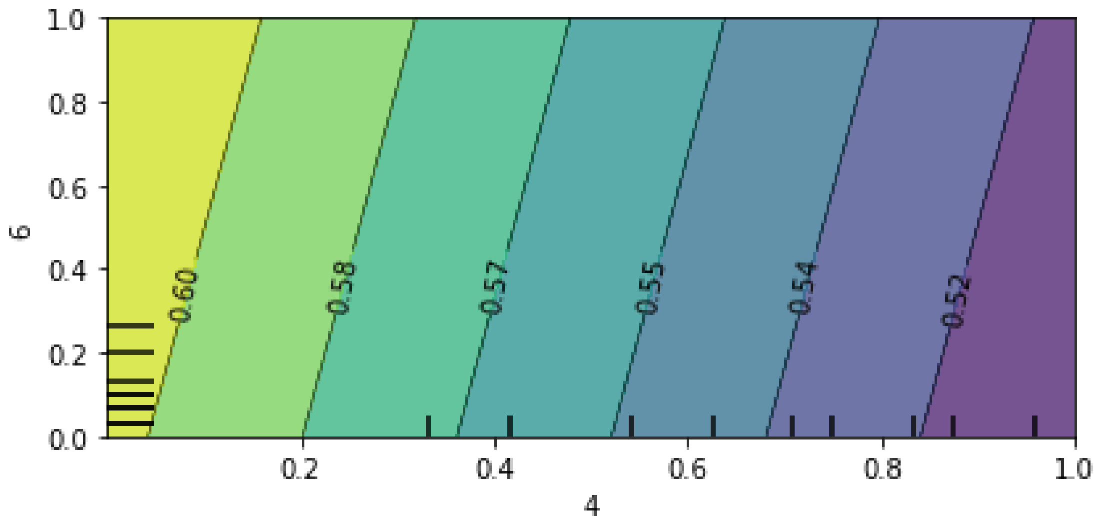

1.0
0.8
0.6
0.4
0.2
0.0
0
0.54
0.49
0.52
0.58
0.56
0.61
0.2
0.4
0.6
0.8
1.0
4

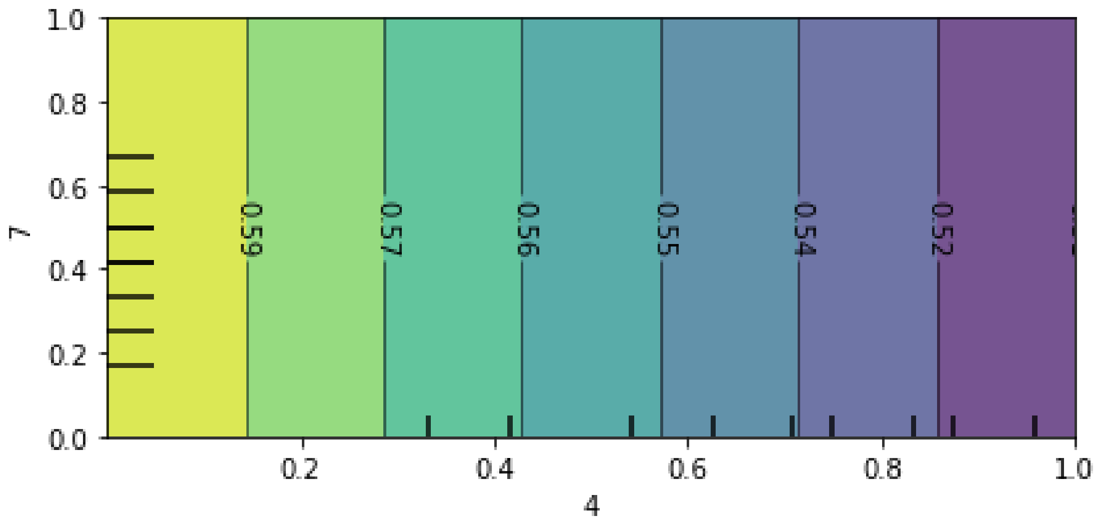

1.0
0.8
0.6
0.4
0.2
0.0
1
0.59
0.57
0.56
0.55
0.54
0.52
0.2
0.4
0.6
0.8
1.0
4


1.0
0.8
0.6
0.4
0.2
0.0
2
0.61
0.60
0.58
0.56
0.54
0.52
0.2
0.4
0.6
0.8
1.0
4

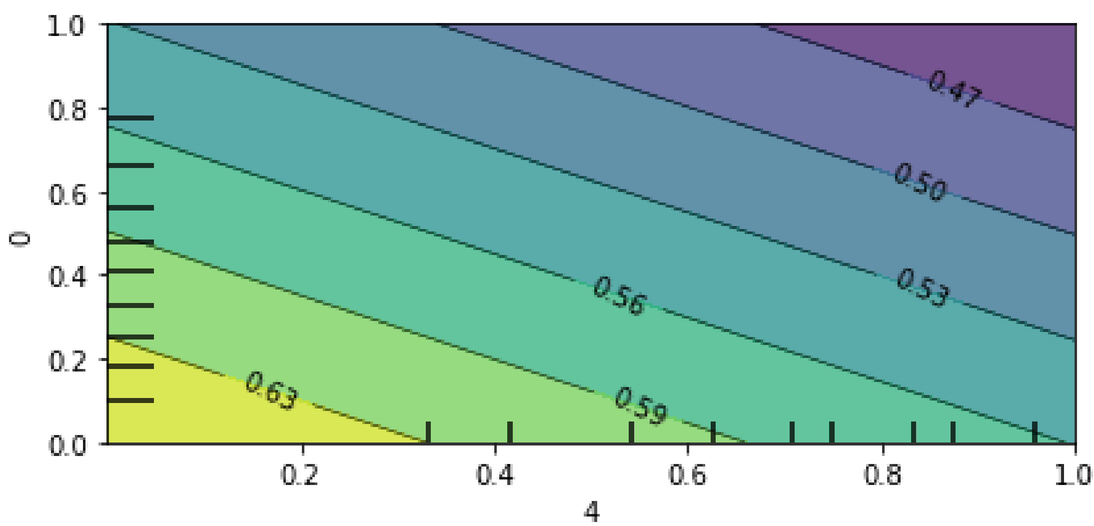

1.0
0.8
0.6
0.4
0.2
0.0
3
0.59
0.58
0.56
0.55
0.53
0.52
0.2
0.4
0.6
0.8
1.0
4

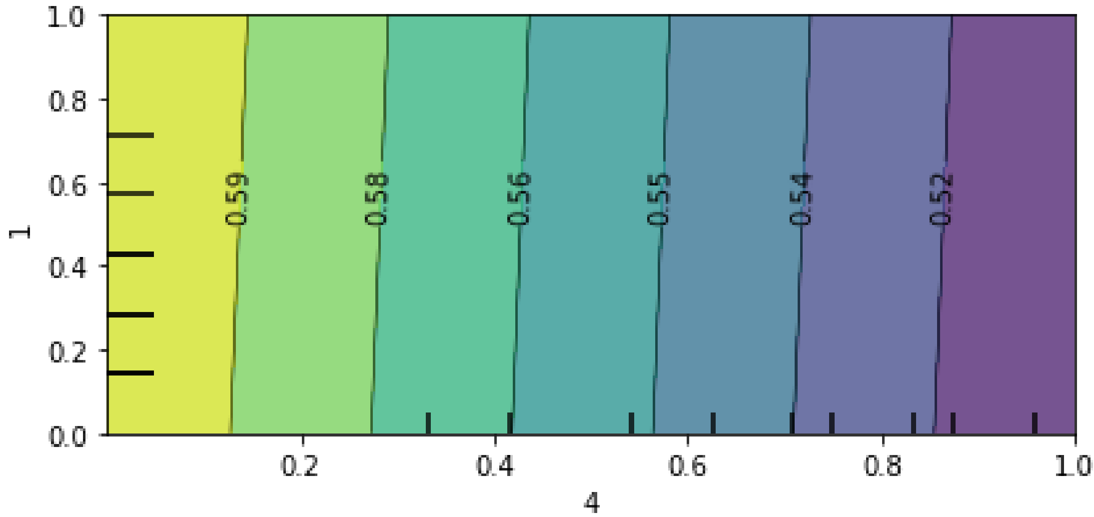

1.0
0.8
0.6
0.4
0.2
0.0
5
0.2
0.4
0.6
0.8
1.0
4
0.59
0.58
0.56
0.55
0.53
0.52

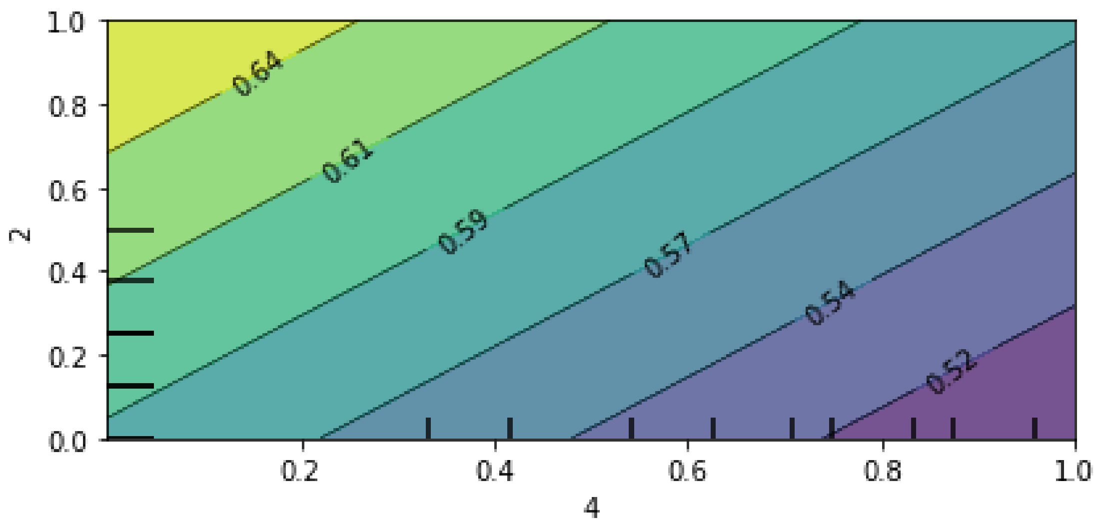

1.0
0.8
0.6
0.4
0.2
0.0
6
0.2
0.4
0.6
0.8
1.0
4
0.60
0.58
0.57
0.55
0.54
0.52

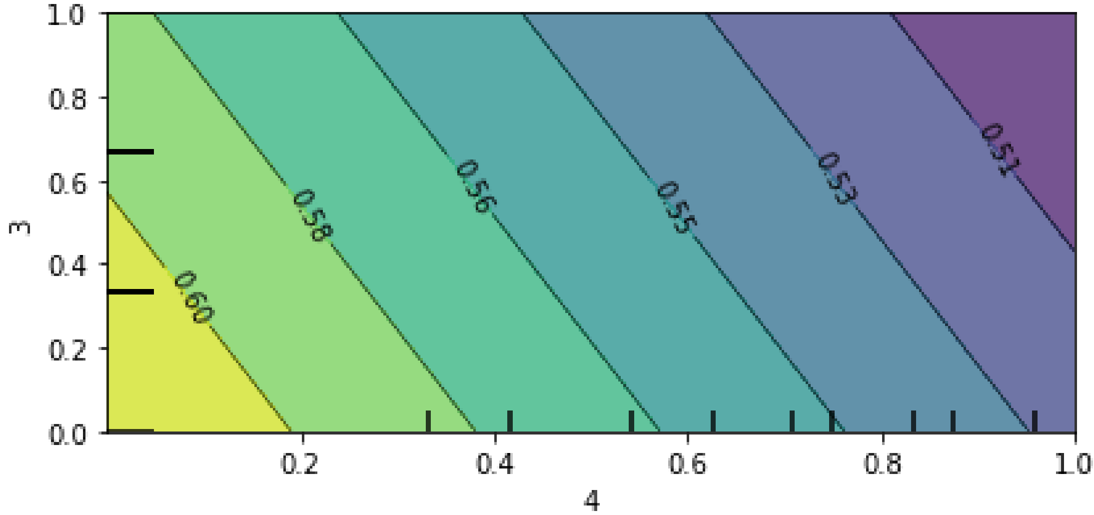


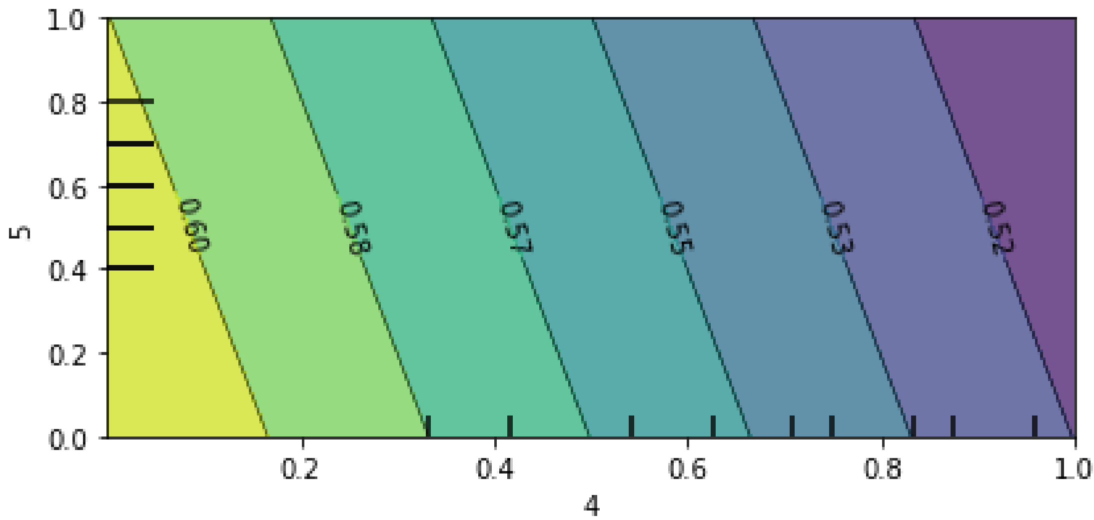
[46]:
```
# plot the partial dependence between income06 and age(0), authoritarianism(1),
↪childs(2), con_govt(3), science_quiz(5)
# sibs(6) and social_connect(7) in partial least square model
pls.fit(X, y)
plot_partial_dependence(pls, X, [(4,0)])
plot_partial_dependence(pls, X, [(4,1)])
plot_partial_dependence(pls, X, [(4,2)])
plot_partial_dependence(pls, X, [(4,3)])
plot_partial_dependence(pls, X, [(4,5)])
plot_partial_dependence(pls, X, [(4,6)])
plot_partial_dependence(pls, X, [(4,7)])
```

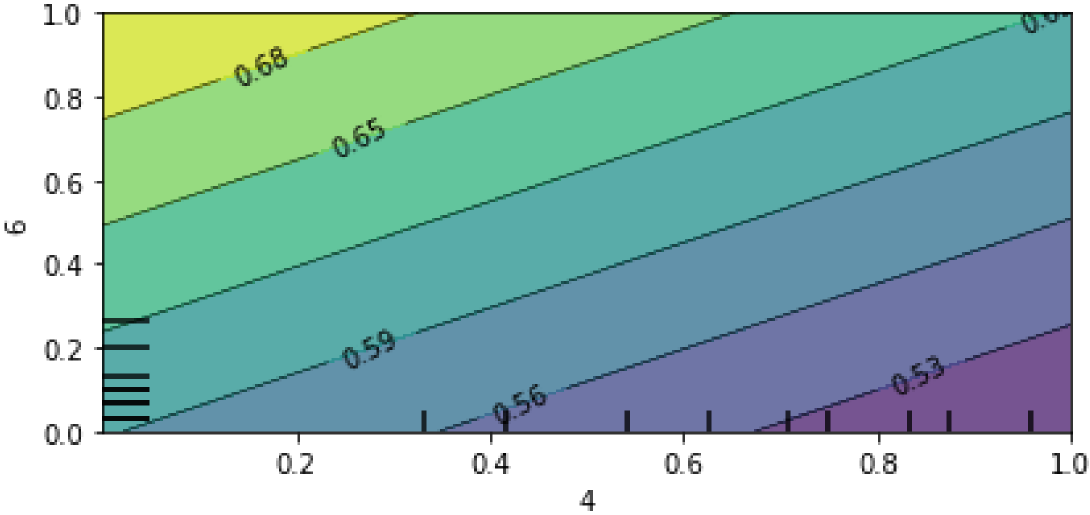


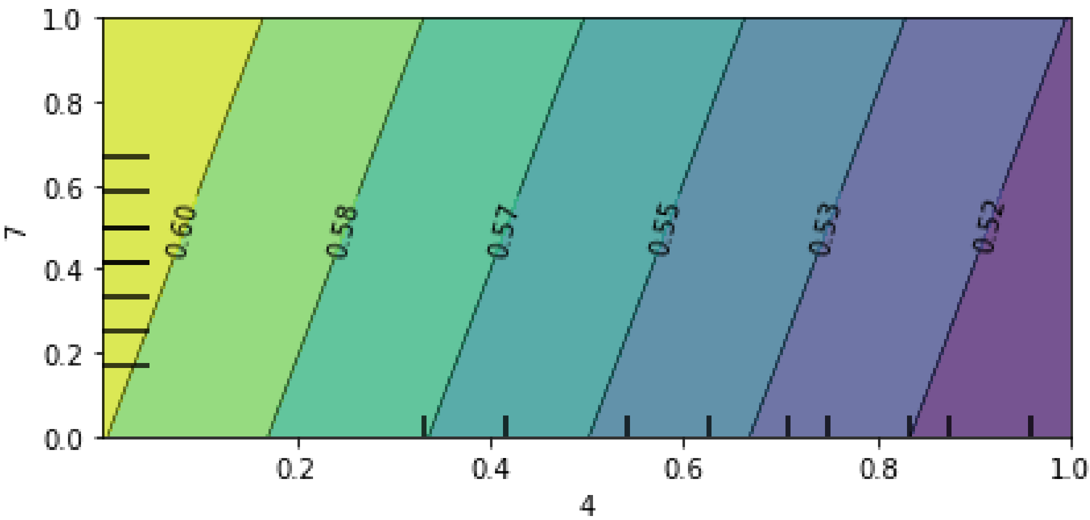

1.0
0.8
0.6
0.4
0.2
0.0
1
0.59
0.58
0.56
0.55
0.54
0.52
0.2
0.4
0.6
0.8
1.0
4


1.0
0.8
0.6
0.4
0.2
0.0
2
0.64
0.61
0.59
0.57
0.54
0.52
0.2
0.4
0.6
0.8
1.0
4


1.0
0.8
0.6
0.4
0.2
0.0
3
0.60
0.58
0.56
0.55
0.53
0.51
0.2
0.4
0.6
0.8
1.0
4


1.0
0.8
0.6
0.4
0.2
0.0
5
0.60
0.58
0.57
0.55
0.53
0.52
0.2
0.4
0.6
0.8
1.0
4





We pick income as the key variable and evaluate interactions with other variables.
In the linear model, all except for authoritarianism seem to have interaction with income.
In the elastic net model, authoritarianism and social connect seem to have the lowest interaction with income.
The partial least square is similar to the linear model in that only authoritarianism has no interaction.

[ ]: