In [254]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from patsy import dmatrix
from sklearn.linear_model import LinearRegression, ElasticNetCV
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import KFold, cross_val_score
from sklearn.preprocessing import PolynomialFeatures, scale, MinMaxScal
er
from sklearn.decomposition import PCA
from sklearn.cross_decomposition import PLSRegression as PLS
from mlxtend.evaluate import feature_importance_permutation
```

In [234]:
```
gss_test = pd.read_csv("gss_test.csv")
gss_train = pd.read_csv("gss_train.csv")
```

Egalitarianism and Income

1.

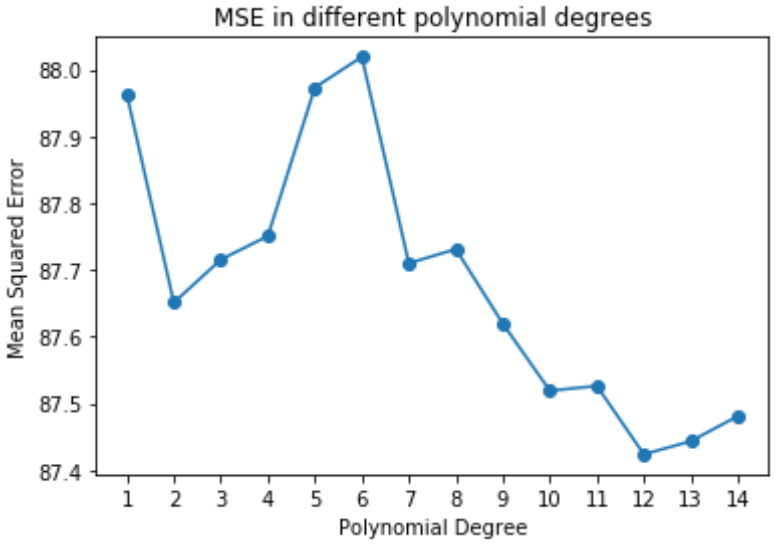
In [235]:
```
y_train = gss_train['egalit_scale']
y_test = gss_test['egalit_scale']
x_train = gss_train['income06'].values.reshape(-1,1)
x_test = gss_test['income06'].values.reshape(-1,1)

k_fold = KFold(n_splits=10, random_state=1, shuffle=False)
lr = LinearRegression()
ls_score = []
for i in range(1,15):
    poly = PolynomialFeatures(degree=i)
    model = lr.fit(poly.fit_transform(x_train), y_train)
    scores = cross_val_score(model, poly.fit_transform(x_train), y_trai
```

```
n, scoring="neg_mean_squared_error", cv=k_fold)
    ls_score.append(np.mean(np.abs(scores)))
ls_score
```

Out[235]:
```
[87.96300193115194,
 87.65085217286715,
 87.71542380949789,
 87.7506412125123,
 87.97322944694855,
 88.01923052261373,
 87.70981678916657,
 87.73112239645913,
 87.61927533519619,
 87.51883223096313,
 87.52597638080267,
 87.4235408929306,
 87.44320029140064,
 87.48095145550201]
```

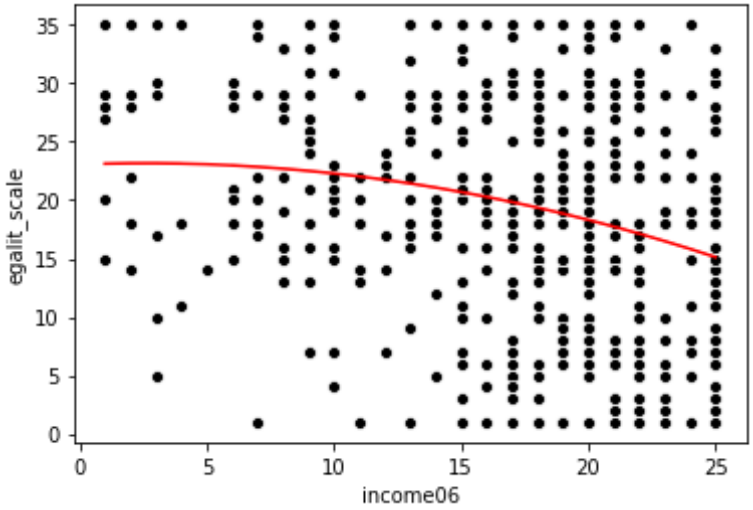
In [236]:
```
plt.plot(np.arange(1,15), ls_score, marker='o', label=ls_score)
plt.xticks(np.arange(1, 15, 1))
plt.title('MSE in different polynomial degrees')
plt.xlabel('Polynomial Degree')
plt.ylabel('Mean Squared Error')
plt.show()
```



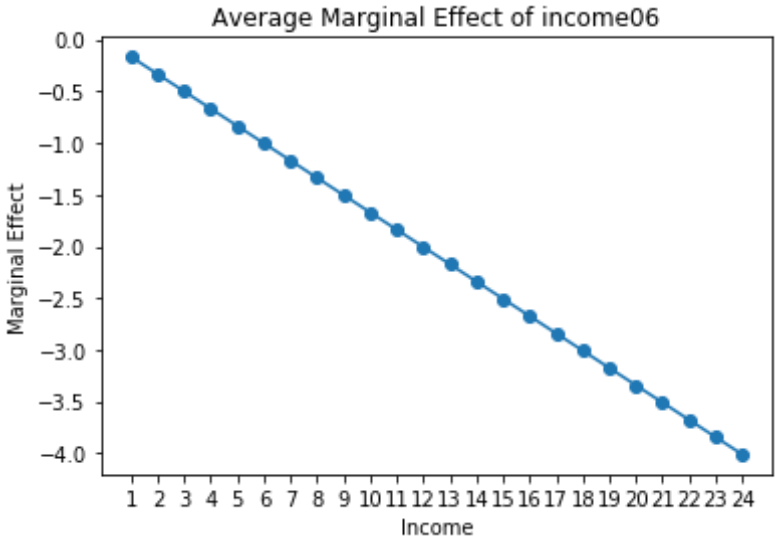
As is shown in the graph, the minimum MSE is when degree = 12 and a local minimum degree =2.

In [237]:
```
poly = PolynomialFeatures(degree=2)
model = lr.fit(poly.fit_transform(x_train), y_train)
sns.scatterplot(gss_test['income06'], y_test, color='black')
sns.lineplot(gss_test['income06'], model.predict(poly.fit_transform(x_t
est)), color='red')
```

Out[237]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1d0c6f65208>
```



In [238]:
```
coefs = lr.coef_
ls_margin = [-coefs[1] * 2 * j for j in range(1, 25)]
plt.plot(np.arange(1,25), ls_margin, marker='o', label=ls_margin)
plt.xticks(np.arange(1, 25, 1))
plt.title('Average Marginal Effect of income06')
plt.xlabel('Income')
plt.ylabel('Marginal Effect')
plt.show()
```



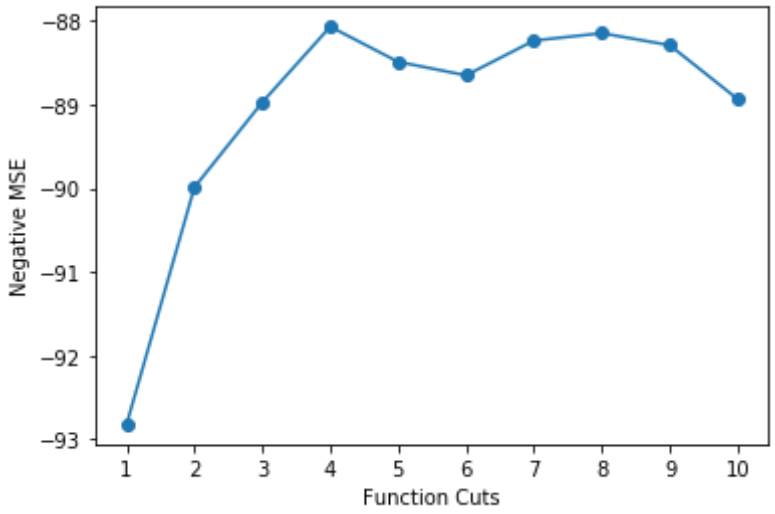
As shown in the picture, there is a negative linear relationship between income and marginal effect of income. It shows that as income increases, the marginal value of it decreases to a negative value

2.

In [239]:
```
x_train = gss_train['income06'].values
x_test = gss_test['income06'].values
dict_step = {}
for i in range(1, 11):
    x_cut, bins = pd.cut(x_train, bins=i, retbins=True)
    dummies = pd.get_dummies(x_cut)
    dummies_model = LinearRegression().fit(dummies, y_train)
    scores = cross_val_score(dummies_model, dummies, y_train, scoring=
"neg_mean_squared_error", cv=10)
    dict_step[i] = scores.mean()
max_step = max(dict_step, key=lambda k: dict_step[k])
max_step
```

Out[239]: 4

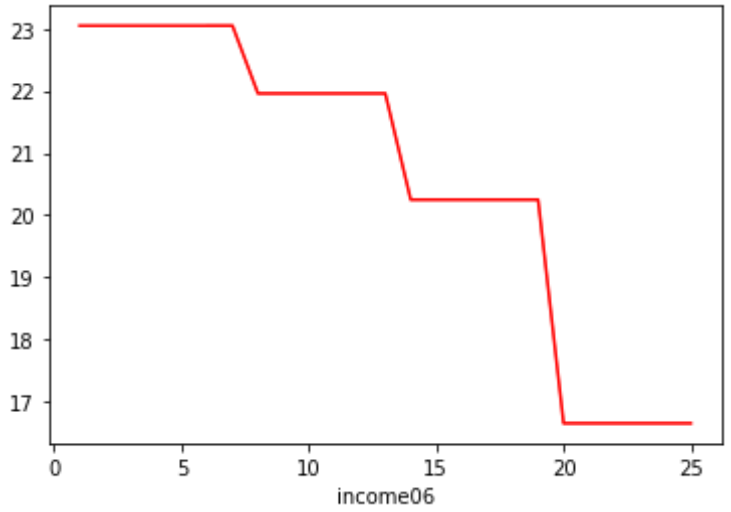
In [240]:
```
plt.plot(np.arange(1,11), list(dict_step.values()), marker='o')
plt.xticks(np.arange(1, 11))
plt.xlabel('Function Cuts')
plt.ylabel('Negative MSE')
plt.show()
```



Hence, the optimal number of cuts is 4

In [241]:
```
x_cut, bins = pd.cut(x_train, bins=4, retbins=True)
dummies = pd.get_dummies(x_cut)
dummies_model = LinearRegression().fit(dummies, y_train)
bin_mapping = np.digitize(x_test, bins, right=True)
sns.lineplot(gss_test['income06'], dummies_model.predict(pd.get_dummies
(bin_mapping)), color='red')
```

Out[241]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1d0c4264e88>
```



The above two plots show that step function with bins = 4 best fit the data. As their income increases, their degrees of egalitarian drops accordingly.

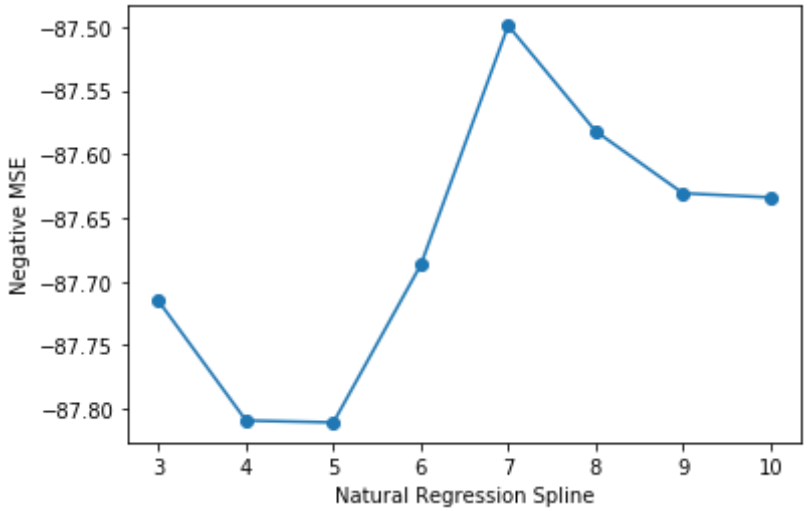
3.

In [242]:
```
dict_natural = {}
for i in range(3, 11):
    x_natural = dmatrix("bs(train, df=i, degree=3)", {"train": x_train,
"df": i}, return_type='dataframe')
    model = LinearRegression().fit(x_natural, y_train)
    scores = cross_val_score(model, x_natural, y_train, scoring="neg_me
an_squared_error", cv=10)
    dict_natural[i] = scores.mean()
max_natural = max(dict_natural, key=lambda k: dict_natural[k])
max_natural
```

Out[242]: 7

In [243]:
```
plt.plot(np.arange(3,11), list(dict_natural.values()), marker='o')
plt.xticks(np.arange(3,11))
plt.xlabel('Natural Regression Spline')
```

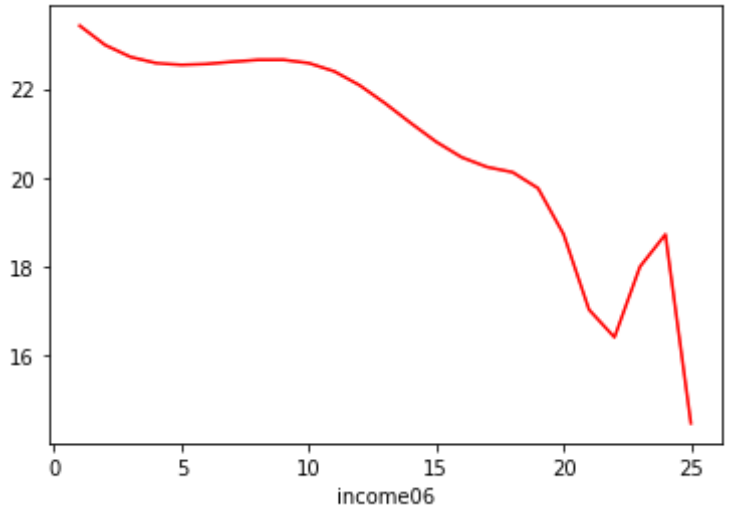
```
plt.ylabel('Negative MSE')
plt.show()
```



Hence, the optimal number of degree of freedom is 7

In [244]:
```
x_natural = dmatrix("bs(train, df=7, degree=3)", {"train": x_train,"df"
: i}, return_type='dataframe')
model = LinearRegression().fit(x_natural, y_train)
test_spline = dmatrix("bs(test, df=7, degree=3)", {"test": x_test},retu
rn_type='dataframe')
sns.lineplot(gss_test['income06'], model.predict(test_spline), color='r
ed')
```

Out[244]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1d0c8111d48>
```



Similarly, the above two plots show that step function with degree of freedom = 7 best fit the data. As their income increases, their degrees of egalitarian drops accordingly but increases later on.

4.

In [245]:
```
x_train = gss_train.drop('egalit_scale', axis=1)
x_test = gss_test.drop('egalit_scale', axis=1)
y_train = gss_train['egalit_scale']
y_test = gss_test['egalit_scale']
x_train.head()
```

Out[245]:

	age	attend	authoritarianism	black	born	childs	colath	colrac	colcom	colmi
0	21	Never	4	No	YES	0	NOT ALLOWED	NOT ALLOWED	FIRED	NOT ALLOWED

	age	attend	authoritarianism	black	born	childs	colath	colrac	colcom	colmi
1	42	Never	4	No	YES	2	ALLOWED	NOT ALLOWED	NOT FIRED	ALLOWED
2	70	<Once/yr	1	Yes	YES	3	ALLOWED	NOT ALLOWED	NOT FIRED	ALLOWED
3	35	Sev times/yr	2	No	YES	2	ALLOWED	NOT ALLOWED	FIRED	NOT ALLOWED
4	24	Sev times/yr	6	No	NO	3	NOT ALLOWED	NOT ALLOWED	FIRED	ALLOWED

5 rows × 44 columns

In [246]:
```
# Drop the object columns and turn int into (0, 1)
scaler = MinMaxScaler(feature_range=(0, 1))

def pre_processing(df):
    for column in df:
        if df[column].dtypes == object:
            df.drop([column], axis=1, inplace=True)
        elif df[column].dtypes == 'int64':
            transform_col = df[column].values.reshape(-1,1)
            scaler.fit(transform_col)
            df[column] = scaler.transform(transform_col)
    return df
```

In [247]:
```
x_train = pre_processing(x_train)
x_test = pre_processing(x_test)

x_train.shape
```

Out[247]: (1481, 11)

In [248]:
```
lr = LinearRegression().fit(x_train, y_train)
scores = - cross_val_score(lr, x_train, y_train, scoring="neg_mean_squa
```

```
red_error", cv=k_fold)
lr_mse = np.mean(scores)
print("Linear Regression: ", lm_mse)
```

```
Linear Regression:  84.20369084148084
```

In [249]:
```
import warnings
warnings.filterwarnings('ignore')
alpha = np.arange(0, 1.1, step=0.1)
elastic = ElasticNetCV(cv=10, alphas=alpha).fit(x_train, y_train)
elastic_mse = mean_squared_error(elastic.predict(x_test), y_test)
print('the test MSE of elastic: ' + str(elastic_mse))
print('l1 ratio: ' + str(elastic.l1_ratio_))
print('alpha: ' + str(elastic.alpha_))
```

```
the test MSE of elastic: 84.92574185114157
l1 ratio: 0.5
alpha: 0.0
```

In [250]:
```
dict_mse_pca = {}
for i in np.arange(1, 12, 1):
    pca = PCA(n_components = i)
    mse_pca = np.sum(-cross_val_score(LinearRegression().fit(x_train, y
_train), pca.fit_transform(x_train),
                                      y_train, cv=10, scoring="neg_mean
_squared_error" ))/10
    dict_mse_pca[i] = mse_pca
min_mse = min(dict_mse_pca, key=lambda k: dict_mse_pca[k])
print(min_mse, dict_mse_pca[min_mse])
```

```
11 84.20369084148085
```

We see that best model for PCA has compnents = 11 and MSE = 84.20369

In [251]:
```
dict_mse_pls = {}
for i in np.arange(1, 12, 1):
    pls = PLS(n_components = i)
    mse_pls = np.sum(-cross_val_score(pls, x_train, y_train, cv=10, sco
```

```
ring="neg_mean_squared_error" ))/10
    dict_mse_pls[i] = mse_pls
min_mse_pls = min(dict_mse_pls, key=lambda k: dict_mse_pls[k])
print(min_mse_pls, dict_mse_pls[min_mse_pls])
```

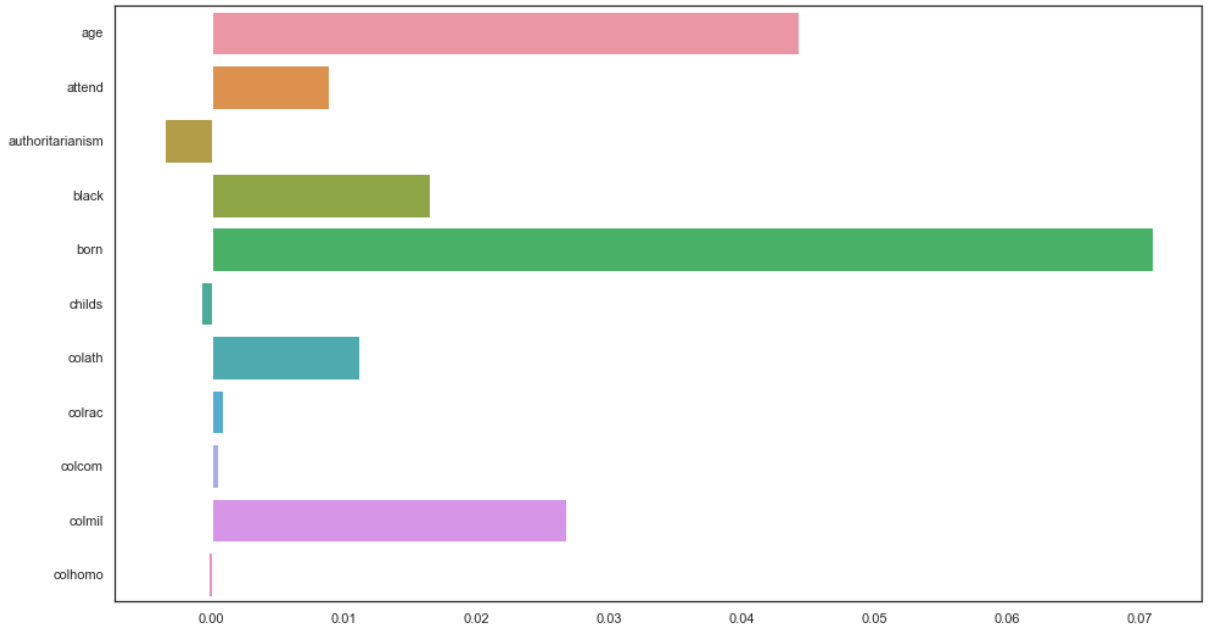
```
7 84.20364375409964
```

We see that best model for PLS has compnents = 7 and MSE = 84.20364

5.

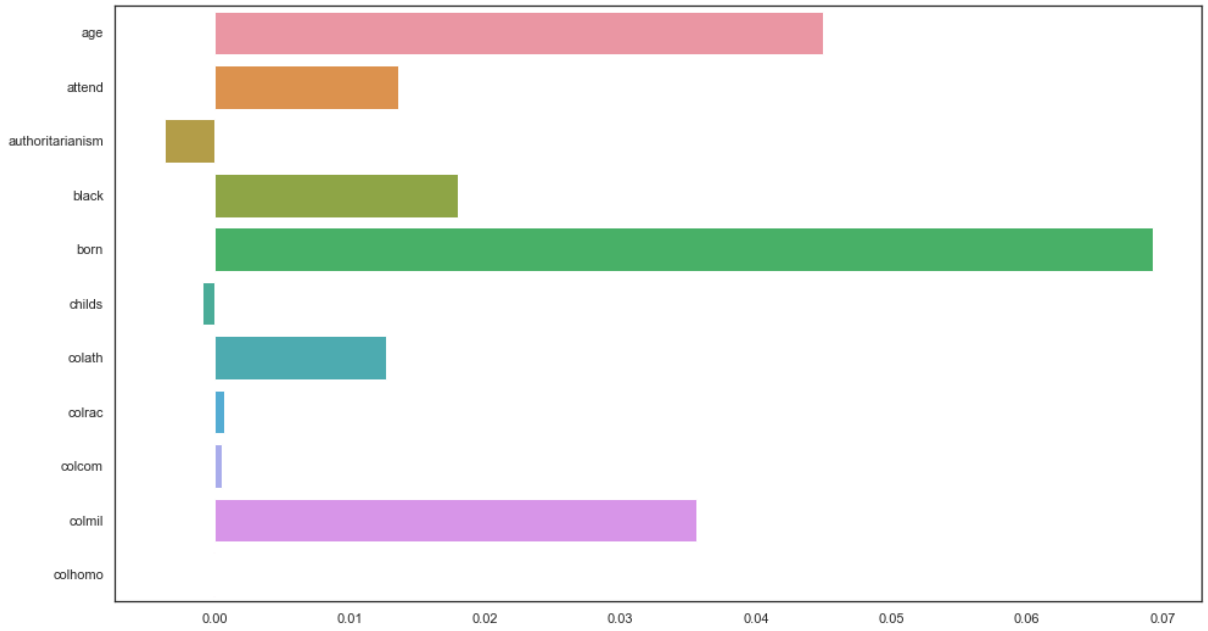
In [274]:
```
def plot_(model):
    imp_vals, _ = feature_importance_permutation(
        predict_method=model.predict,X=x_test, y=y_test, metric='r2',nu
m_rounds=10)
    ls_col = []
    ls_imp = []
    for i in range(x_test.shape[1]):
        ls_col.append(gss_test.columns[i])
        ls_imp.append(imp_vals[i])
    sns_barplot = sns.barplot(x=ls_imp, y=ls_col)
    return sns_barplot
```

In [275]:
```
lr_plot = plot_(LinearRegression().fit(x_train, y_train))
```



In [289]:
```
alpha = np.arange(0, 1.1, step=0.1)
elastic = ElasticNetCV(cv=10, alphas=alpha).fit(x_train, y_train)
print(elastic)
elastic_plot = plot_(elastic)
```

```
ElasticNetCV(alphas=array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8,
0.9, 1. ]),
             copy_X=True, cv=10, eps=0.001, fit_intercept=True, l1_rati
o=0.5,
             max_iter=1000, n_alphas=100, n_jobs=None, normalize=False,
             positive=False, precompute='auto', random_state=None,
             selection='cyclic', tol=0.0001, verbose=0)
```



We see that for pls and linear regression, there seems to exist interactio nbetween income and age, colmill, born, which is across almost all graphs, and it matches with our existing theory.