Untitled

February 15, 2020

[1]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
import matplotlib.pyplot as plt
import wes
wes.set_palette('FantasticFox1')
```

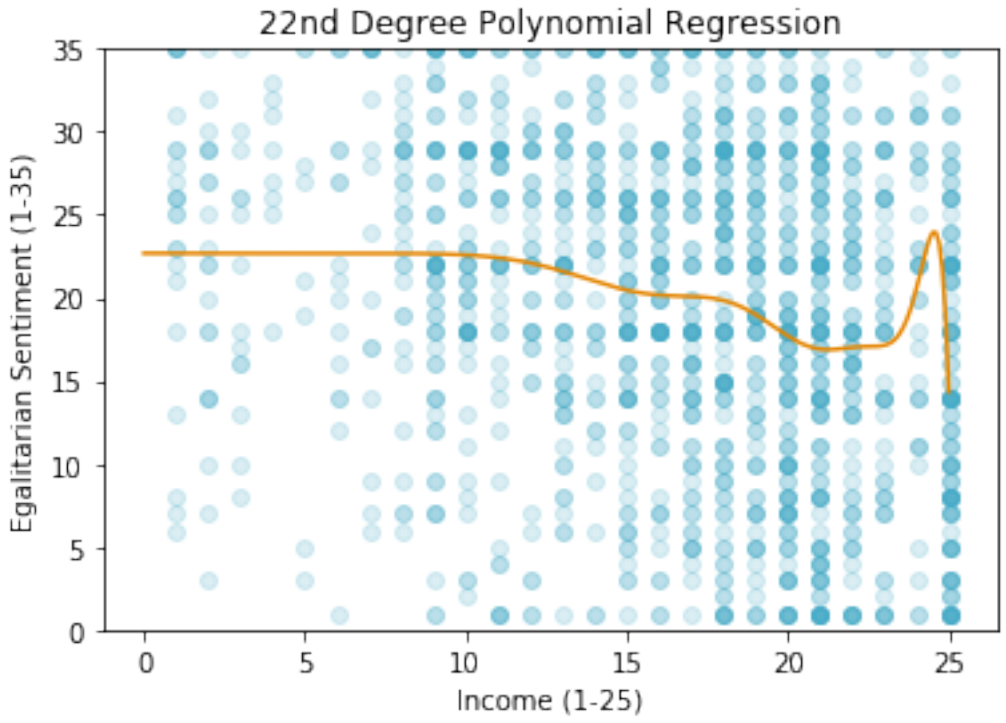
0.1 Egalitarianism and Income

0.1.1 1

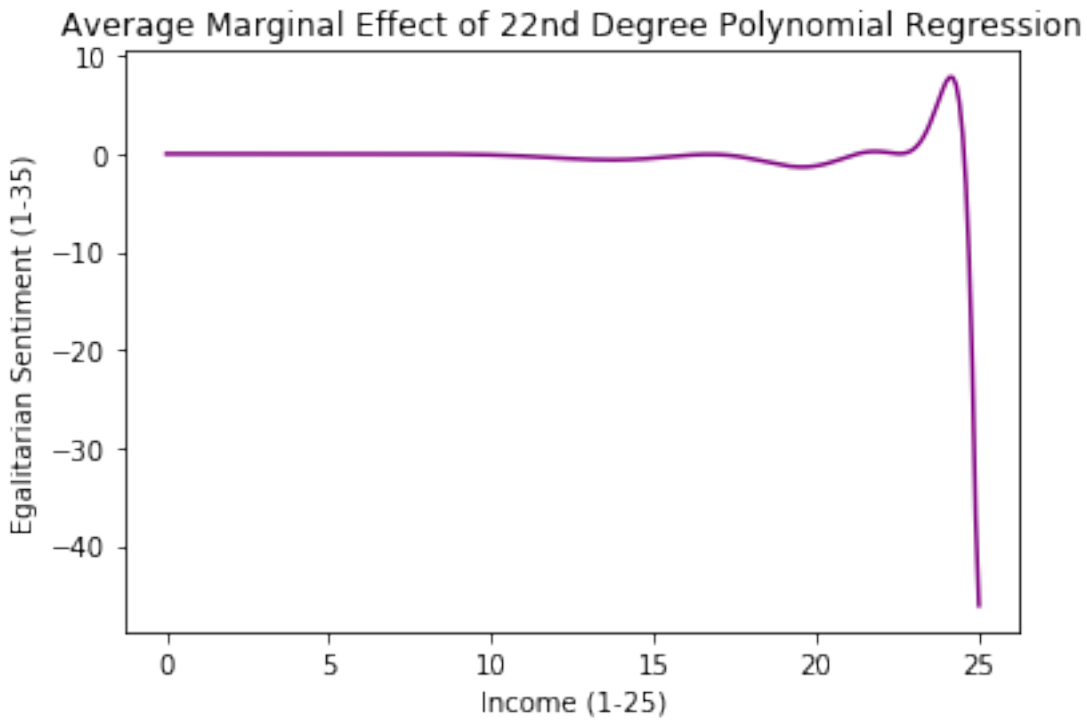
[2]:
```
np.random.seed(232)
```

[11]:
```
test = pd.read_csv('data/gss_test.csv')
train = pd.read_csv('data/gss_train.csv')

y_train = train['egalit_scale']
x_train = train['income06'].values.reshape(-1, 1)
y_test = test['egalit_scale']
x_test = test['income06'].values.reshape(-1, 1)

pipe = Pipeline([('poly', PolynomialFeatures()), ('linear',␣
 ↪LinearRegression())])
parameters = {'poly__degree': range(1, 101)}
griddy = GridSearchCV(pipe, parameters, n_jobs=-1,
                      scoring='neg_mean_squared_error', refit=True, cv=10)

griddy.fit(x_train, y_train)

scores = -1 * griddy.cv_results_['mean_test_score']
poly = griddy.best_estimator_
x_plot = np.linspace(0, 25, 250).reshape(-1, 1)
y_plot = poly.predict(x_plot)
```

```
plt.plot(x_plot, y_plot, c='C3')
plt.scatter(x_train, y_train, c='C2', alpha=.2)
plt.ylim(0, 35)
plt.xlabel('Income (1-25)')
plt.ylabel('Egalitarian Sentiment (1-35)')
plt.title('22nd Degree Polynomial Regression')
plt.show()
```



[12]:
```
x_ame = x_plot
y_ame = np.gradient(y_plot, .1)

plt.figure()
plt.plot(x_ame, y_ame, c='purple')
plt.title('Average Marginal Effect of 22nd Degree Polynomial Regression')
plt.xlabel('Income (1-25)')
plt.ylabel('Egalitarian Sentiment (1-35)')
plt.show()
```

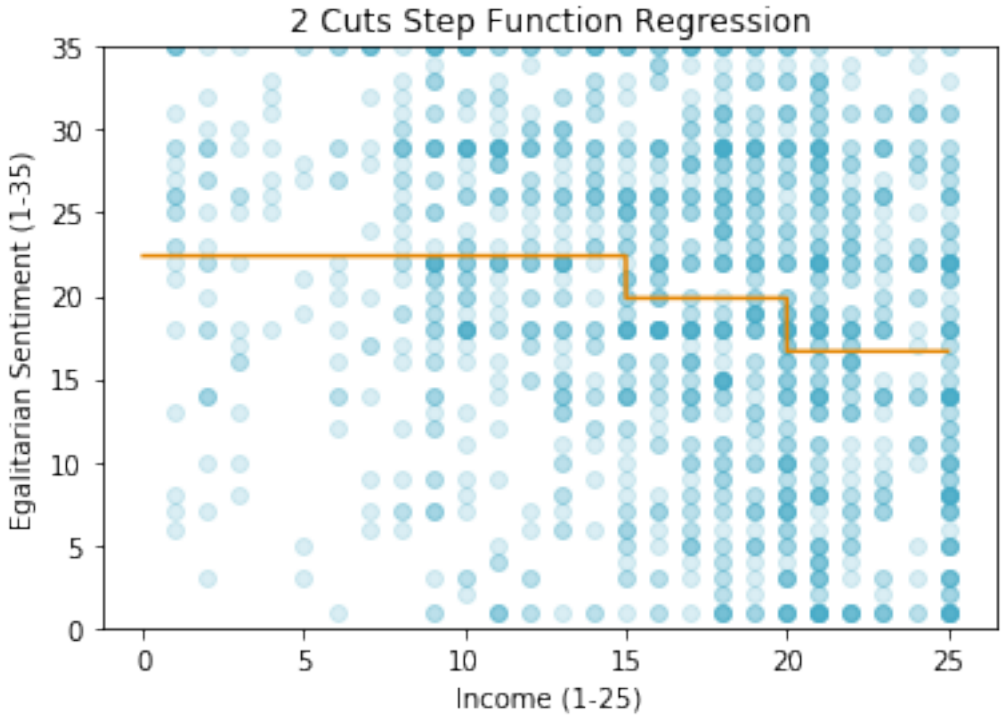


So This is basically no effect. The MSE for this was ~87, and the average marginal effect is essentially zero for all values. There's some weirdness on both ends, but that's an artifact of the function being an odd-degree polynomial. Also this is a high degree polynomial, which is impossible to interpret basically. There is a case to be made for high degree polynomials being Taylor Series behaviors, but I reran this analysis with the highest possible degree being 100, and it still optimized at 22. So we can assume that this is a comfortable analysis. There may be a sudden increase in marginal effect in the very high income regime, but this could easily be an anomaly, since it only occurs for the income brackets 24 and 25.

0.1.2 2

[13]:
```
from sklearn.preprocessing import KBinsDiscretizer

pipe = Pipeline([('binner', KBinsDiscretizer()), ('linear',␣
 ↪LinearRegression())])
parameters = {'binner__n_bins': range(1, 11)}
griddy = GridSearchCV(pipe, parameters, n_jobs=-1,
                      scoring='neg_mean_squared_error', refit=True, cv=10)


griddy.fit(x_train, y_train)

steppy = griddy.best_estimator_
x_plot = np.linspace(0, 25, 10000).reshape(-1, 1)
```

```
# I'm using a higher resolution here because stepwise prediction is␣
 ↪computationally easy and it
# makes the plot look better IMO.
y_plot = steppy.predict(x_plot)


plt.plot(x_plot, y_plot, c='C3')
plt.scatter(x_train, y_train, c='C2', alpha=.2)
plt.ylim(0, 35)
plt.xlabel('Income (1-25)')
plt.ylabel('Egalitarian Sentiment (1-35)')
plt.title('2 Cuts Step Function Regression')
plt.show()
```

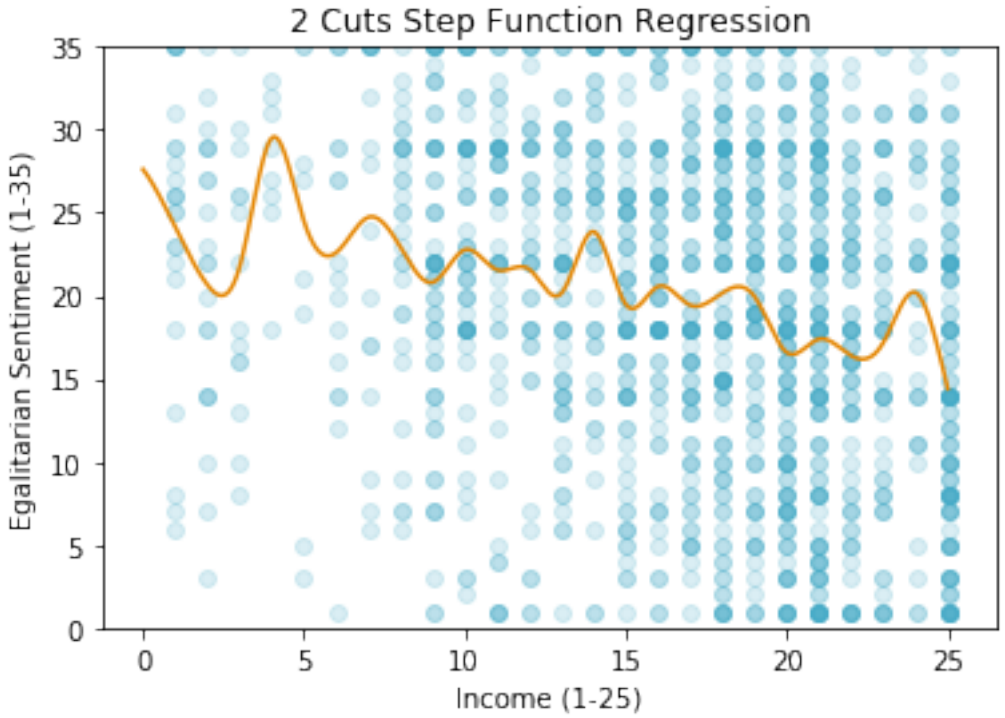


The optimal number of cuts is 2 (n_bins = 3). Again the MSE is pretty high. The best score for this CV was actually worse than the best polynomial score, coming in at about 88. This fit makes the claim that there are regions, low income, middle income, and high income, which have different average egalitarian sentiments. But these averages are also pretty close to each other, so it might not be all that meaningful.

[14]:
```
from scipy.interpolate import CubicSpline
from sklearn.model_selection import KFold
```

```
# See the documentation for this object. It implements the cubic spline which␣
 ↪can be forced to be natural.
# KFold is a builder function for GridSearchCV so I can implement CV for a␣
 ↪non-sklearn method.
# make_interp_spline() has controls for degrees of spline and knots.



record = {
    'mse': [],
    'model': [],
    'knots': []
}

kfold = KFold(n_splits=10)
splits = kfold.split(x_train, y_train)

for i_tr, i_te in splits:

    xtr = x_train[i_tr].flatten()
    ytr = y_train[i_tr].values.flatten()
    xva = x_train[i_te].flatten()
    yva = y_train[i_te].values.flatten()

    df = pd.DataFrame(data={'x':xtr, 'y':ytr}, index=pd.RangeIndex(len(xtr)))
    df = df.groupby('x', as_index=False)['y'].mean()
    xtrm = df['x'].values
    ytrm = df['y'].values

    sp = CubicSpline(xtrm, ytrm, bc_type='natural')

    y_pred = sp(xva)

    mse = np.mean((y_pred - yva) ** 2)

    record['mse'].append(mse)
    record['model'].append(sp)


best_mod = np.argmin(record['mse'])
best_mod = record['model'][best_mod]
x_plot = np.linspace(0, 25, 10000).reshape(-1, 1)
# I'm using a higher resolution here because stepwise prediction is␣
 ↪computationally easy and it
# makes the plot look better IMO.
y_plot = best_mod(x_plot)
```

```
plt.plot(x_plot, y_plot, c='C3')
plt.scatter(x_train, y_train, c='C2', alpha=.2)
plt.ylim(0, 35)
plt.xlabel('Income (1-25)')
plt.ylabel('Egalitarian Sentiment (1-35)')
plt.title('2 Cuts Step Function Regression')
plt.show()
```



Yeah this is unparsimonius as bonk, and it's barely better performing than the simpler models. It gets an MSE of about 75. The plot looks like gibberish too.

0.2 Egalitarianism and Everything

0.2.1 1

Note that I'm using sklearn's standard scalar which uses

$$\tilde{x}_{ij} = \frac{x_{ij} - \bar{x}_{ij}}{\sqrt{\frac{1}{N}\sum_i (x_{ij} - \bar{x}_{ij})}} \tag{1}$$

instead of

$$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{N}\sum_i (x_{ij} - \bar{x}_{ij})}} \tag{2}$$

The `with_mean=False` parameter sets the scalar to use the standardizer instead of a standard-normalizer. I believe that the former is the better choice generally, but the latter is what was mentioned in class, so we'll sticck to that.

[75]:
```
from sklearn.linear_model import ElasticNetCV
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.decomposition import PCA
from sklearn.cross_decomposition import PLSRegression
from sklearn.metrics import mean_squared_error as mse
from time import time

x_train = train.drop('egalit_scale', axis=1)
y_train = train['egalit_scale']
x_test = test.drop('egalit_scale', axis=1)
y_test = test['egalit_scale']

for col in x_train:
    if x_train[col].dtype == 'object':
        enc = LabelEncoder()
        x_train[col] = enc.fit_transform(x_train[col])
        x_test[col] = enc.transform(x_test[col])
```

[83]:
```
# a)

st = time()
pipea = Pipeline([('stand', StandardScaler(with_mean=False)),
         ('linreg', LinearRegression())])

grida = GridSearchCV(pipe, param_grid={}, cv=10, n_jobs=-1, refit=True)
grida.fit(x_train, y_train)

ma = mse(grida.predict(x_test), y_test)
mta = mse(grida.predict(x_train), y_train)
ti = time() - st
print(f'Linear Regression finished in {ti:.03f} seconds.\n Test MSE is {ma:.
 ↪03f}, Train MSE is {mta:.03f}.')
#b)

st = time()
pipeb = Pipeline([('stand', StandardScaler(with_mean=False)),
         ('enet', ElasticNetCV(alphas=[.1, .5, .7, .9, .95, .99, 1], cv=10))])

pipeb.fit(x_train, y_train)
mb = mse(pipeb.predict(x_test), y_test)
```

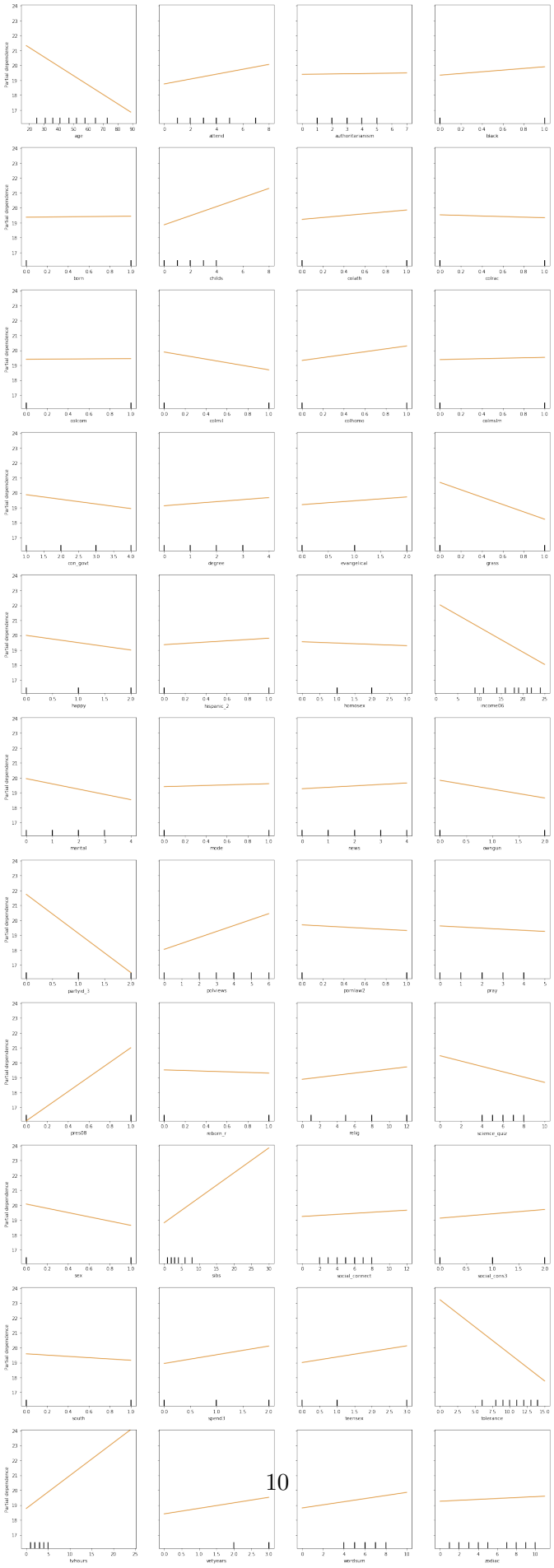
```
mtb = mse(pipeb.predict(x_train), y_train)
ti = time() - st
print(f'Elastic Net Regression finished in {ti:.03f} seconds.\n Test MSE is {mb:
 ↪.03f}, Train MSE is {mtb:.03f}.')

#c)
st = time()
pipec = Pipeline([('stand', StandardScaler(with_mean=False)),
                  ('dcomp', PCA()),
                  ('linreg', LinearRegression())])

gridc = GridSearchCV(pipec, param_grid={'dcomp__n_components': range(1, 45)},␣
 ↪cv=10, n_jobs=-1, refit=True)

gridc.fit(x_train, y_train)
mc = mse(gridc.predict(x_test), y_test)
mtc = mse(gridc.predict(x_train), y_train)
ti = time() - st
print(f'Principal Component Regression finished in {ti:.03f} seconds.\n Test␣
 ↪MSE is {mc:.03f}, Train MSE is {mtc:.03f}.')

#d)
st = time()
piped = Pipeline([('stand', StandardScaler(with_mean=False)),
                  ('pls', PLSRegression())])

gridd = GridSearchCV(piped, param_grid={'pls__n_components': range(1, 45)},␣
 ↪cv=10, n_jobs=-1, refit=True)

gridd.fit(x_train, y_train)
md = mse(gridd.predict(x_test), y_test)
mtd = mse(gridd.predict(x_train), y_train)
ti = time() - st
print(f'Partial Least Squares Regression finished in {ti:.03f} seconds.\n Test␣
 ↪MSE is {md:.03f}, Train MSE is {mtd:.03f}.')
```

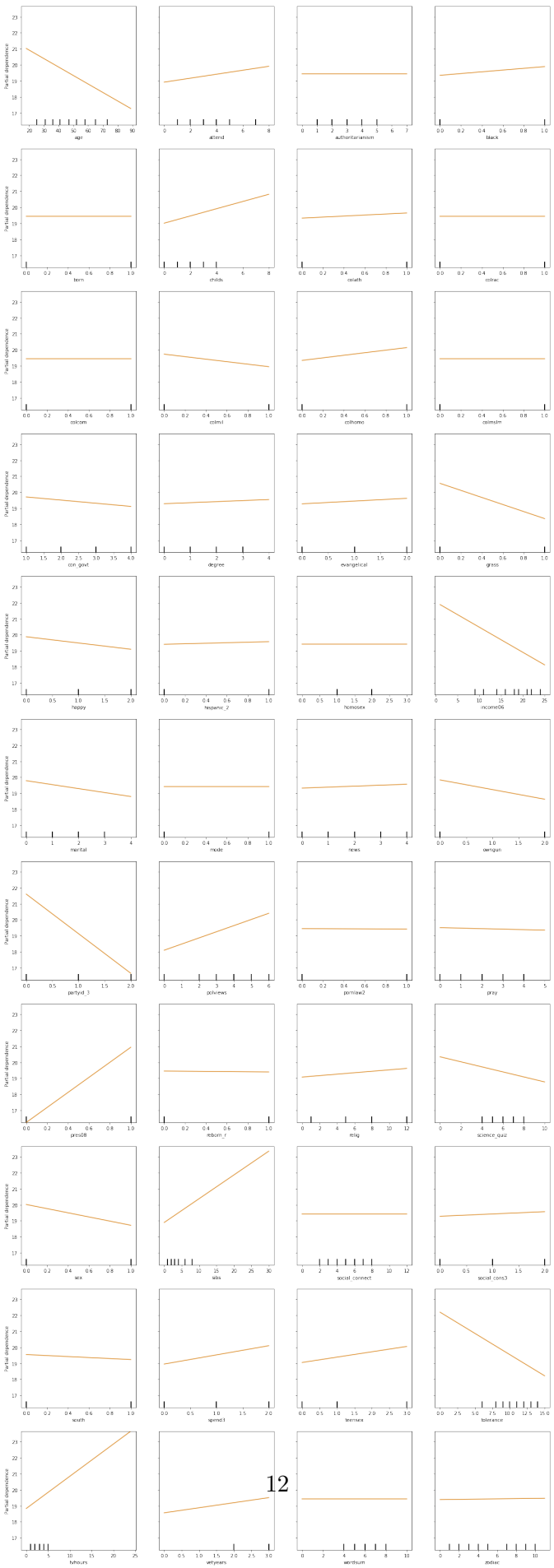
```
Linear Regression finished in 0.094 seconds.
 Test MSE is 64.010, Train MSE is 60.077.
Elastic Net Regression finished in 0.038 seconds.
 Test MSE is 62.784, Train MSE is 60.323.
Principal Component Regression finished in 2.544 seconds.
 Test MSE is 64.010, Train MSE is 60.077.
Partial Least Squares Regression finished in 3.963 seconds.
 Test MSE is 61.727, Train MSE is 62.253.
```

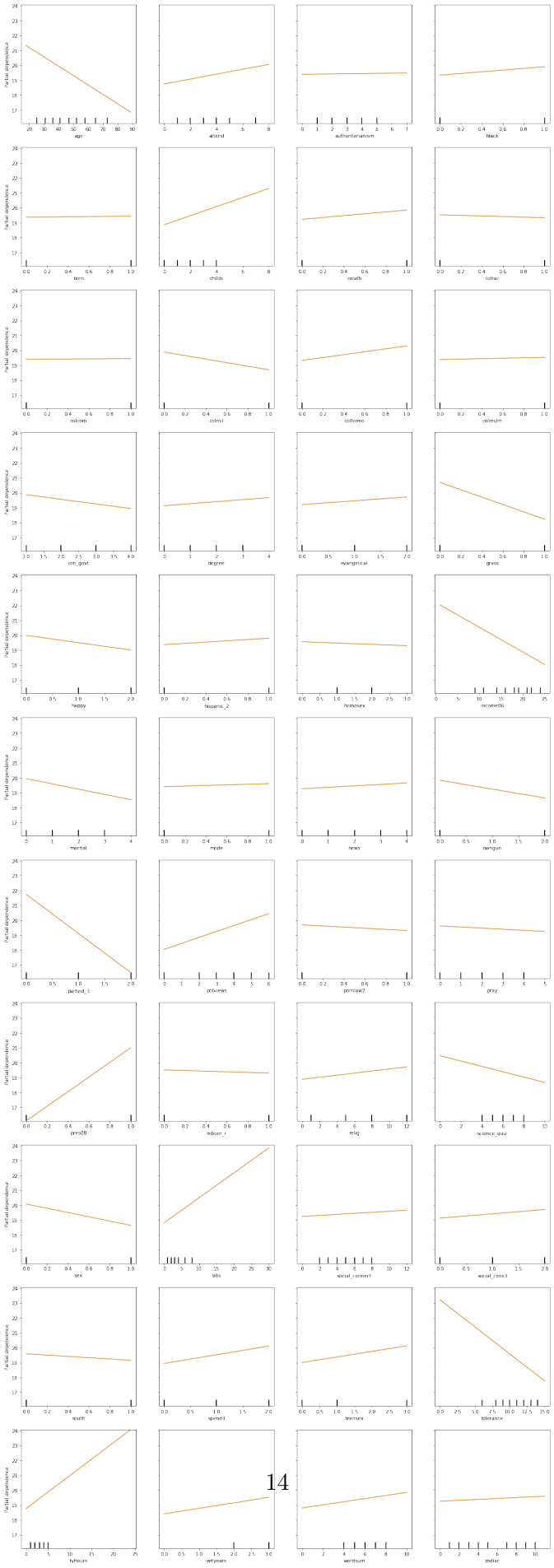
0.2.2 2

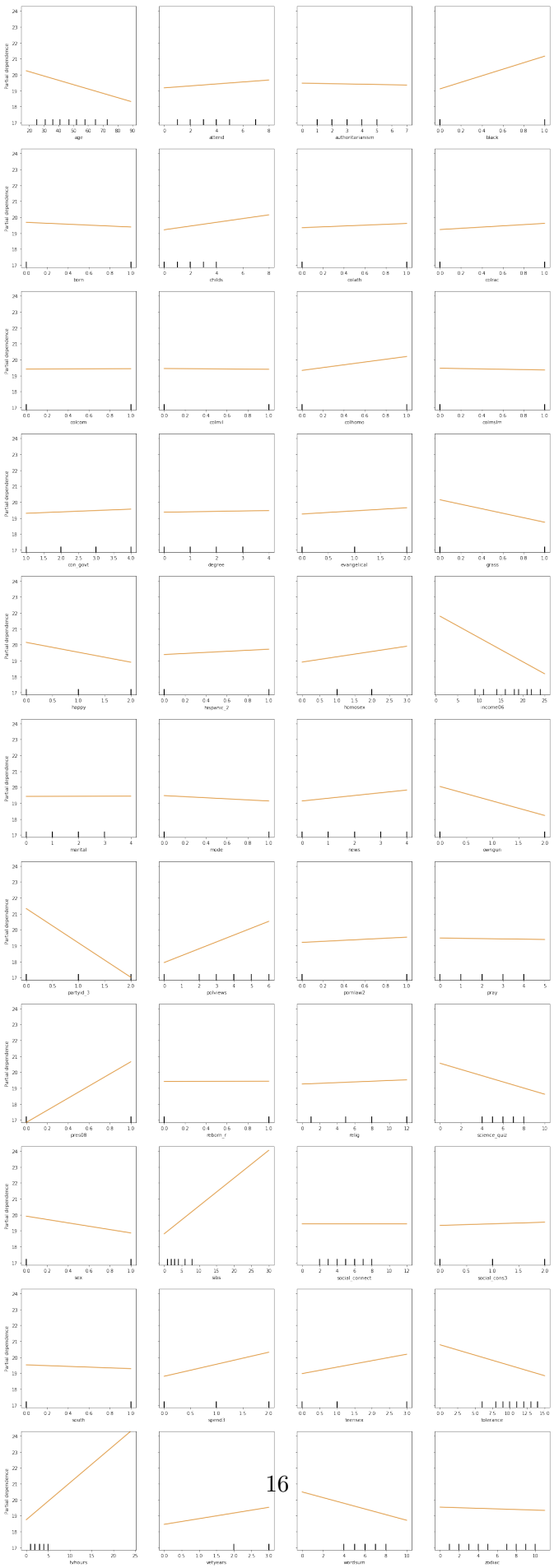
[113]:
```
%matplotlib inline

from sklearn.inspection import plot_partial_dependence
fig, ax = plt.subplots(11, 4, figsize=(20, 60))
plot_partial_dependence(grida, x_train, x_train.columns, n_jobs=-1, ax=ax)
```

[113]:
```
<sklearn.inspection._partial_dependence.PartialDependenceDisplay at
0x27b3bb20a08>
```


Partial dependence
age
attend
authoritarianism
black
born
childs
colath
colrac
colcom
colmil
colhomo
colmslm
con_govt
degree
evangelical
grass
happy
hispanic_2
homosex
income06
mental
mode
news
owngun
partyid_3
polviews
pornlaw2
pray
pres08
reborn_r
relig
science_quiz
sex
sibs
social_connect
social_cons3
south
spend1
teensex
tolerance
tvhours
wtyears
wordsum
zodiac

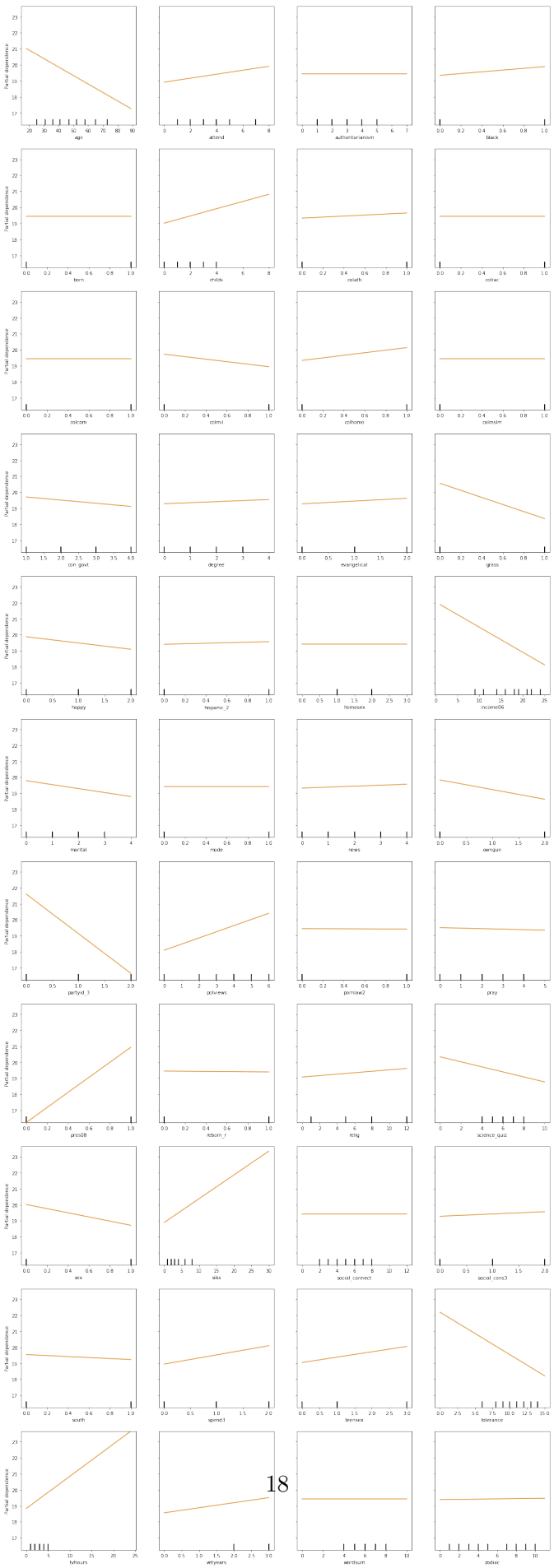

Partial dependence
age
attend
authoritarianism
black
born
childs
colath
colhec
colrelig
colmil
colhomo
colmslm
con_govt
degree
evangelical
grass
happy
hispanic_2
homosex
income06
marital
mode
news
owngun
partyid_3
polviews
pornlaw2
pray
pres08
reborn_r
relig
science_quiz
sex
sibs
social_connect
social_cons3
south
spend1
teensex
tolerance
tvhours
wtyears
wordsum
zodiac


Partial dependence
age
attend
authoritarianism
black
born
childs
colath
colrac
colcom
colmil
colhomo
colmslm
con_govt
degree
evangelical
grass
happy
hispanic_2
homosex
income06
mental
mode
news
owngun
partyid_3
polviews
pornlaw2
pray
pres08
reborn_r
relig
science_quiz
sex
sibs
social_connect
social_cons3
south
spend1
teensex
tolerance
tvhours
wtyears
wordsum
zodiac


Partial dependence
age
attend
authoritarianism
black
born
childs
colath
colrac
colcom
colmil
colhomo
colmslm
con_govt
degree
evangelical
grass
happy
hispanic_2
homosex
income06
marital
mode
news
owngun
partyid_3
polviews
pornlaw2
pray
pres08
reborn_r
relig
science_quiz
sex
sibs
social_connect
social_cons3
south
spend1
teensex
tolerance
tvhours
wtyears
wordsum
zodiac

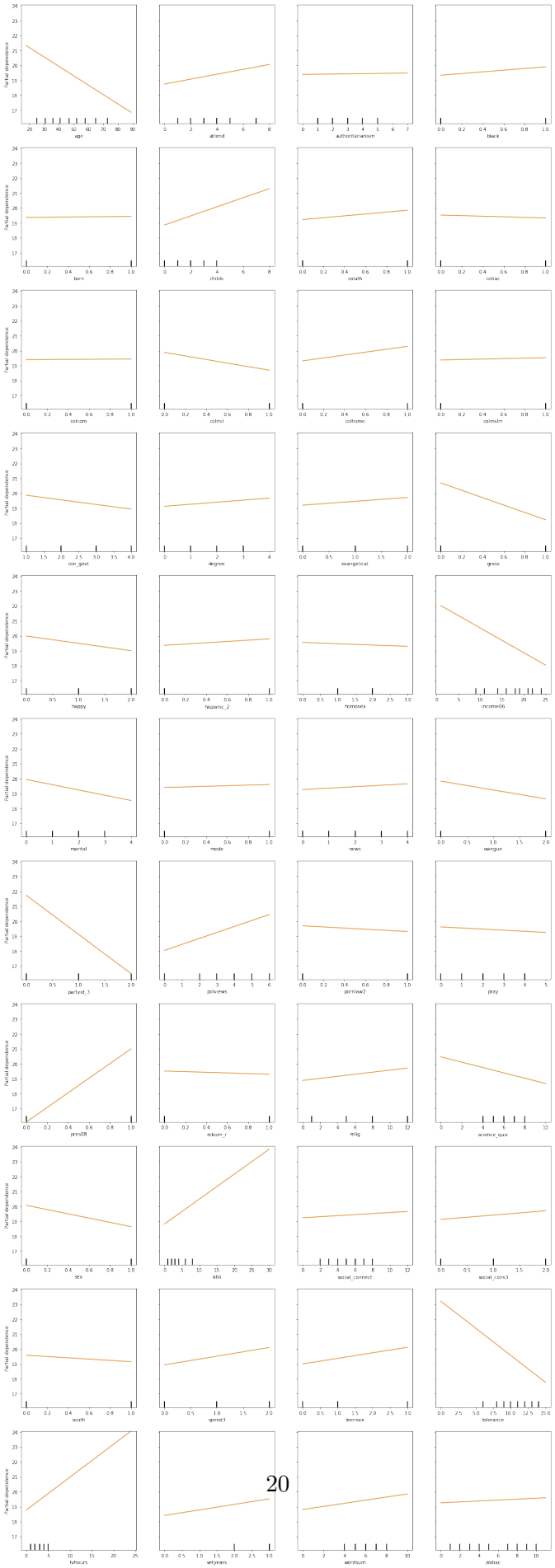
[114]:
```
fig, ax = plt.subplots(11, 4, figsize=(20, 60))
plot_partial_dependence(pipeb, x_train, x_train.columns, n_jobs=-1, ax=ax)
```

[114]:
```
<sklearn.inspection._partial_dependence.PartialDependenceDisplay at
0x27b3ea3cb08>
```


Partial dependence
age
attend
authoritarianism
black
born
childs
colath
colrac
colrelig
colmil
colhomo
colmslm
con_govt
degree
evangelical
grass
happy
hispanic_2
homosex
income06
marital
mode
news
owngun
partyid_3
polviews
pornlaw2
pray
pres08
reborn_r
relig
science_quiz
sex
sibs
social_connect
social_cons3
south
spend1
teensex
tolerance
tvhours
wtyears
wordsum
zodiac

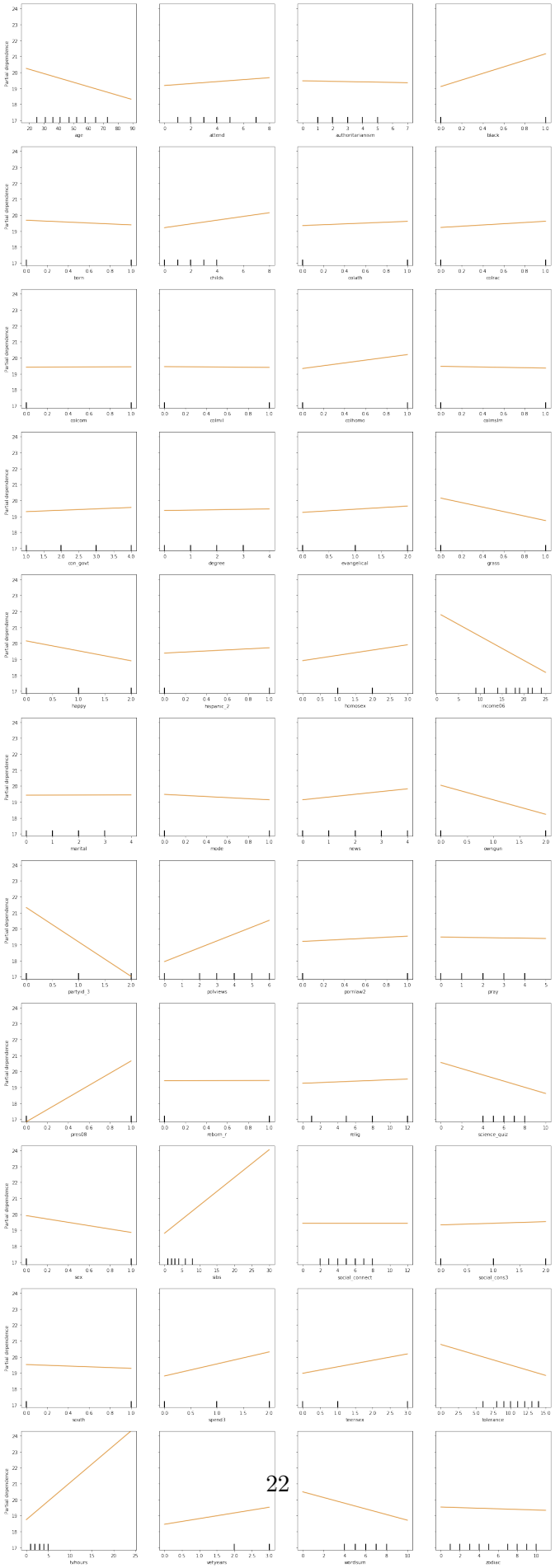
[115]:
```
fig, ax = plt.subplots(11, 4, figsize=(20, 60))
plot_partial_dependence(gridc, x_train, x_train.columns, n_jobs=-1, ax=ax)
```

[115]:
```
<sklearn.inspection._partial_dependence.PartialDependenceDisplay at
0x27b41f1fa88>
```


Partial dependence
age
attend
authoritarianism
black
born
childs
colath
colrac
colcom
colmil
colhomo
colmslm
con_govt
degree
evangelical
grass
happy
hispanic_2
homosex
income06
mental
mode
news
owngun
partyid_3
polviews
pornlaw2
pray
pres08
reborn_r
relig
science_quiz
sex
sibs
social_connect
social_cons3
south
spend1
teensex
tolerance
tvhours
wtyears
wordsum
zodiac

[116]:
```
fig, ax = plt.subplots(11, 4, figsize=(20, 60))
plot_partial_dependence(gridd, x_train, x_train.columns, n_jobs=-1, ax=ax)
```

[116]:
```
<sklearn.inspection._partial_dependence.PartialDependenceDisplay at
0x27b48818c88>
```


Partial dependence
age
attend
authoritarianism
black
born
childs
colath
colrac
colcom
colmil
colhomo
colmslm
con_govt
degree
evangelical
grass
happy
hispanic_2
homosex
income06
marital
mode
news
owngun
partyid_3
polviews
pornlaw2
pray
pres08
reborn_r
relig
science_quiz
sex
sibs
social_connect
social_cons3
south
spend1
teensex
tolerance
tvhours
wtyears
wordsum
zodiac

These methods all come to roughly similar conclusions about each feature. Most of the features have very little effect on egalitarian sentiments (even authoritarian sentiments, weirdly enough, but then again it seems like every political stance in the US is weirdly crypto-fascist so maybe I shouldn't be surprised.) The features that are related with reasonable effect sizes are the questions that directly address political beleifs. Things like `grass` which is opinion on cannabis legalization, or who they voted for in 2008, or party affiliation. There are a few red herrings, like `income06` which we discussed in part one, and `tvhours` and `sibs`. These all have really heavily tailed distributions though, which cause the effect size to be overstated in linear regressions. One of the things that is substantially different in each of these is that the Elastic Net has a lot more of the parameters just set to a flat zero. If you look at the MSE results above this is probably what causes the Elastic Net to outperform other methods on the test set.