Hu_Chun_HW4

February 16, 2020

0.1 Non-linear regression

In this problem set, you are going to predict individual feelings towards egalitarianism. Specifically, egalit_scale is an additive index constructed from a series of questions designed to measure how egalitarian individuals are – that is, the extend to which they think economic opportunities should be distributed more equally in society. The variable ranges from 1 (low egalitarianism) to 35 (high egalitarianism).

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import statsmodels.api as sm
from patsy import dmatrix
from sklearn.linear_model import LinearRegression, ElasticNetCV
from sklearn.decomposition import PCA
from sklearn.preprocessing import PolynomialFeatures, scale, MinMaxScaler
from sklearn.cross_decomposition import PLSRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import KFold, cross_val_score

import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
gss_test = pd.read_csv("gss_test.csv")
gss_train = pd.read_csv("gss_train.csv")
```

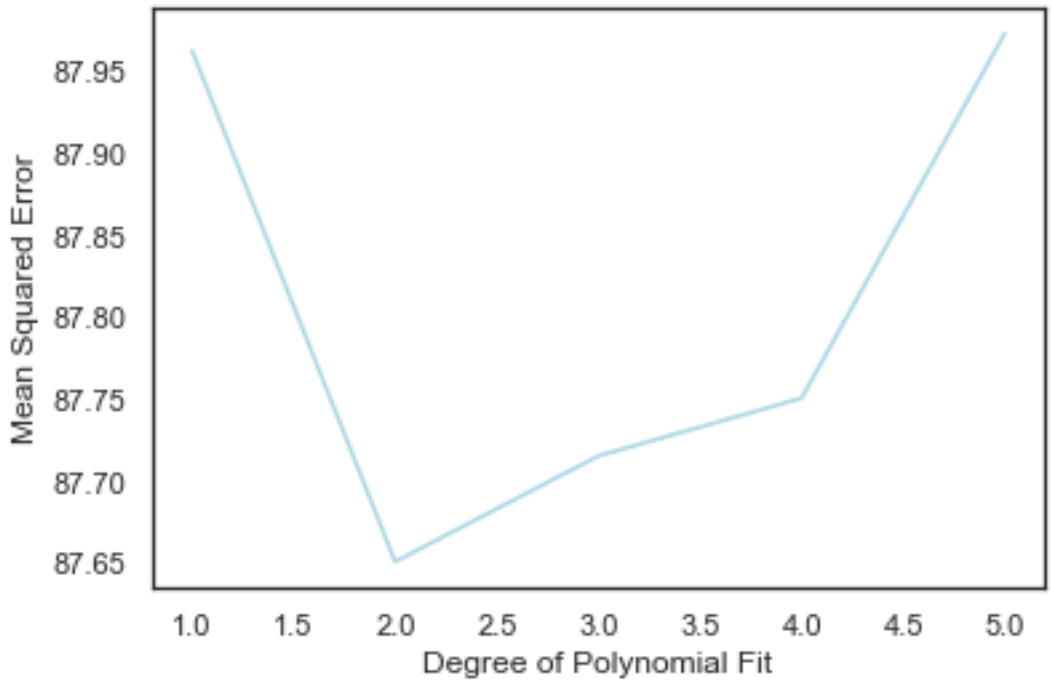
0.1.1 Egalitarianism and Income

1.(20 points) 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.

[3]:
```
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)
```

[4]:
```
crossvalidation = KFold(n_splits=10, random_state=1, shuffle=False)
lm = LinearRegression()
cv_dict = {}
for i in range(1,6):
    poly = PolynomialFeatures(degree=i)
    X_poly = poly.fit_transform(X_train)
    model = lm.fit(X_poly, y_train)
    scores = cross_val_score(model, X_poly, y_train,␣
 ↪scoring="neg_mean_squared_error", cv=crossvalidation)
    cv_dict[i] = np.mean(np.abs(scores))
```

[5]:
```
sns.set(rc={'figure.figsize':(6,4)})
sns.set_style("white")
ds = range(1, 6)
plt.plot(ds, list(cv_dict.values()), color = 'lightblue')
plt.xlabel('Degree of Polynomial Fit')
plt.ylabel('Mean Squared Error')
plt.show()
```

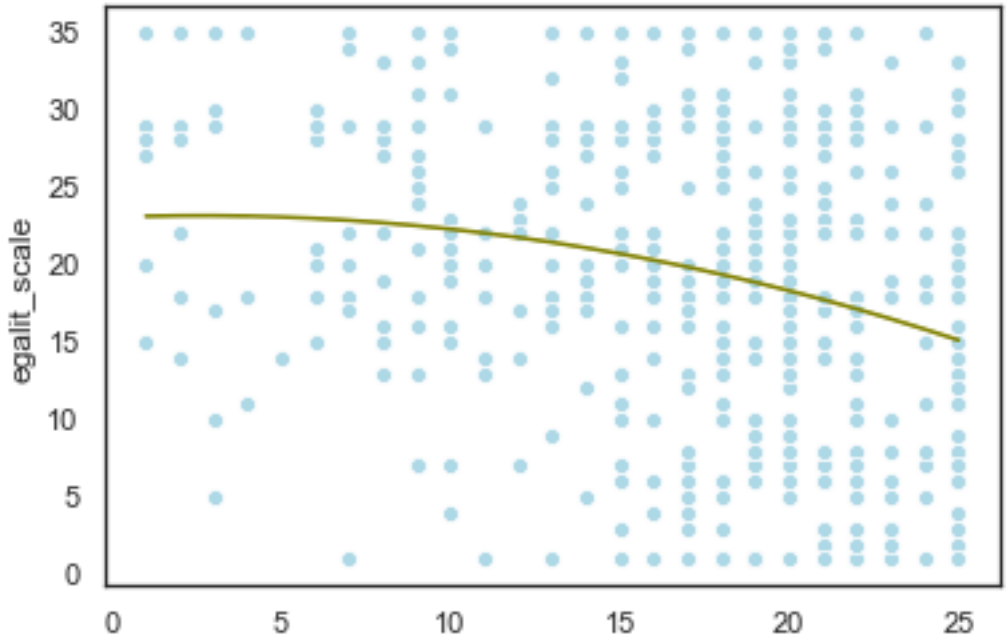


The optimal degree for the polynomial based on MSE is 2.

[6]:
```
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X_train)
model = lm.fit(X_poly, y_train)

sns.scatterplot(X_test.ravel(), y_test, color='lightblue')
```

```
sns.lineplot(X_test.ravel(), model.predict(poly.fit_transform(X_test)), color =␣
 ↪'olive')
```

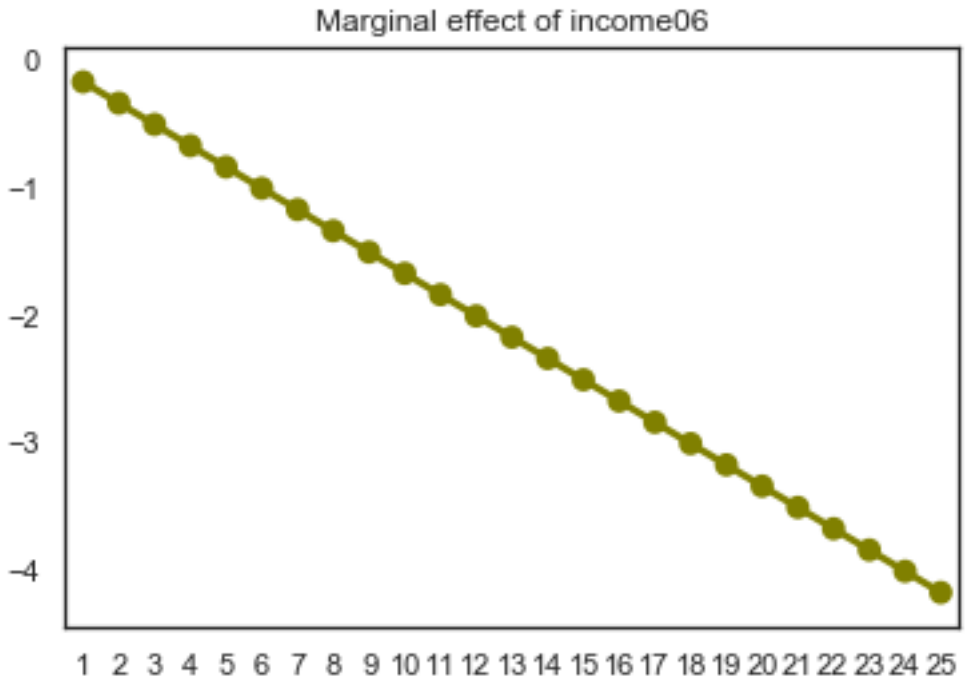
[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1035761d0>
```



[7]:
```
coefs = lm.coef_
def marginal_effect(x):
    y = 0
    for i in range(2):
        y += coefs[i] * (i+1) * (x**(i))
    return y

marg = []
for x in range(1,26):
    marg.append(-marginal_effect(x))
```

[8]:
```
income = [i for i in range(1,26)]
sns.pointplot(income, marg, color = 'olive').set_title("Marginal effect of␣
 ↪income06")
```

[8]:
```
Text(0.5, 1.0, 'Marginal effect of income06')
```

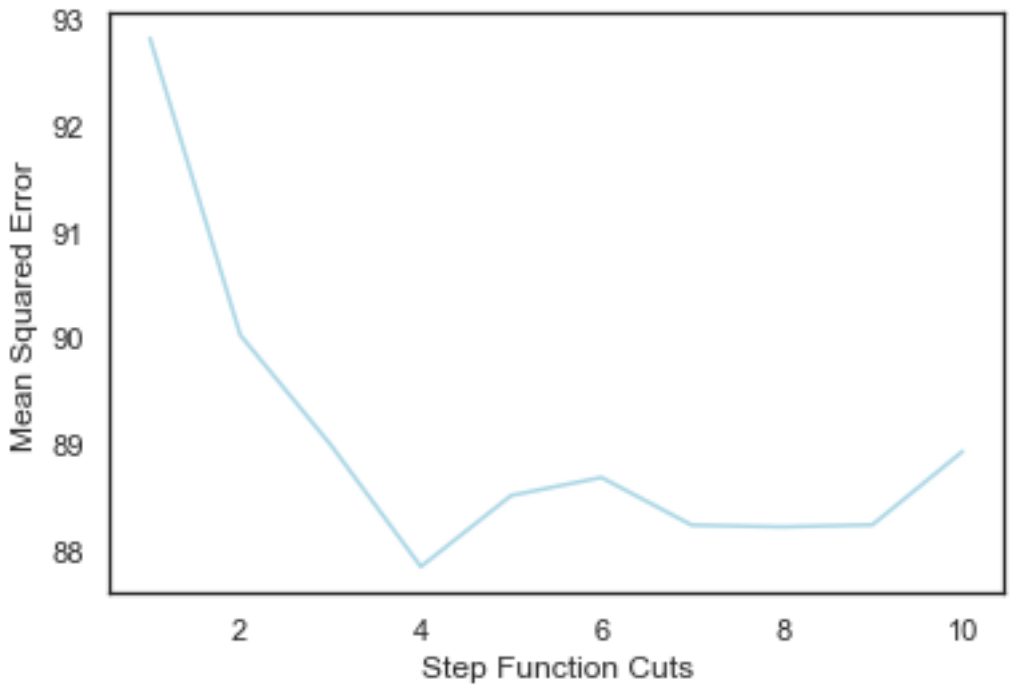


The plot for marginal effect showed a negative linear relationship, suggesting that as income06 increases, the marginal effect of income06 decreases, and the effect is in negative values. Our polynomial fit at degree=2 showed a curvilinear prediction line, slightly decreasing as income06 increases.

2.(20 points) 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.

[9]:
```
step_dict = {}
for i in range(1,11):
    df_cut, bins = pd.cut(X_train.ravel(), i, retbins = True, right = True)
    df_dummies = pd.get_dummies(df_cut)
    df_dummies = sm.add_constant(df_dummies)
    model = lm.fit(df_dummies, y_train)
    scores = cross_val_score(model, df_dummies, y_train,␣
 ↪scoring="neg_mean_squared_error", cv=crossvalidation)
    step_dict[i] = np.mean(np.abs(scores))
```

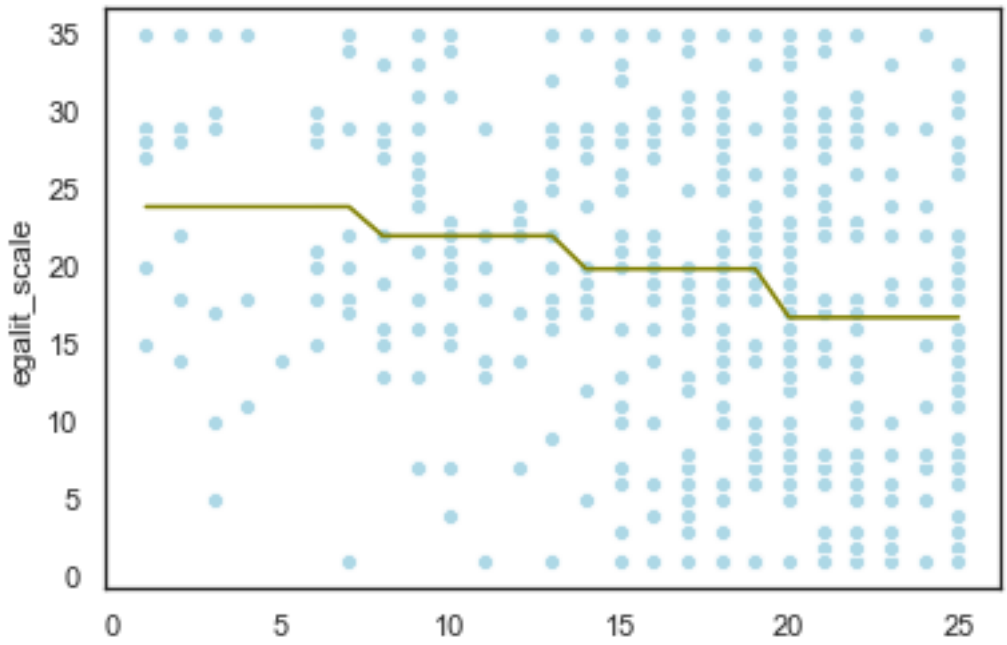
[10]:
```
mg = range(1, 11)
plt.plot(mg, list(step_dict.values()), color = 'lightblue')
plt.xlabel('Step Function Cuts')
plt.ylabel('Mean Squared Error')
plt.show()
```



The optimal number of cuts for step function is 4.

[11]:
```
df_cut, bins = pd.cut(X_train.ravel(), 4, retbins = True, right = True)
df_dummies = pd.get_dummies(df_cut)
df_dummies = sm.add_constant(df_dummies)
model = lm.fit(df_dummies, y_train)

bin_mapping = np.digitize(X_test.ravel(), bins, right = True)
test_dummies = pd.get_dummies(bin_mapping)
test_dummies = sm.add_constant(test_dummies)

sns.scatterplot(X_test.ravel(), y_test, color='lightblue')
sns.lineplot(X_test.ravel(), model.predict(test_dummies), color = 'olive')
```

[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c187ab978>
```

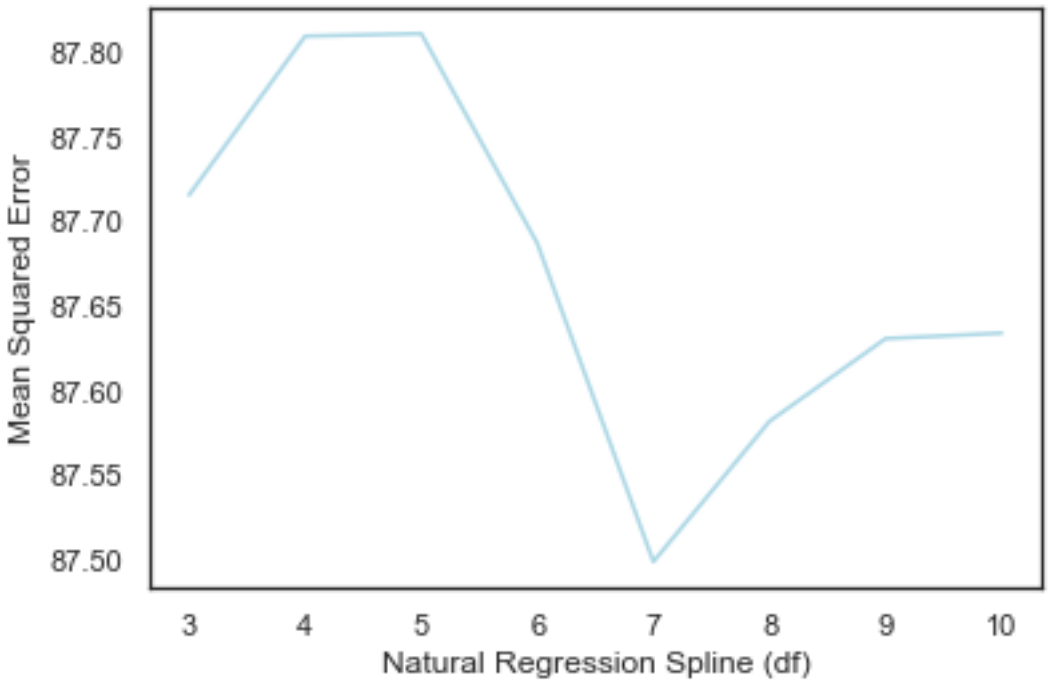


The plot showed the predicted value of egalit scale at 4 bins of income06. In general, the egalit scale decreases as income06 increases.

3.(20 points) 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.

[12]:
```
spline_dict = {}
for i in range(3,11):
    transformed_x = dmatrix("bs(train, df=i, degree=3)", {"train": X_train,␣
 ↪"df": i}, return_type='dataframe')
    model = lm.fit(transformed_x, y_train)
    scores = cross_val_score(model, transformed_x, y_train,␣
 ↪scoring="neg_mean_squared_error", cv=crossvalidation)
    spline_dict[i] = np.mean(np.abs(scores))
```

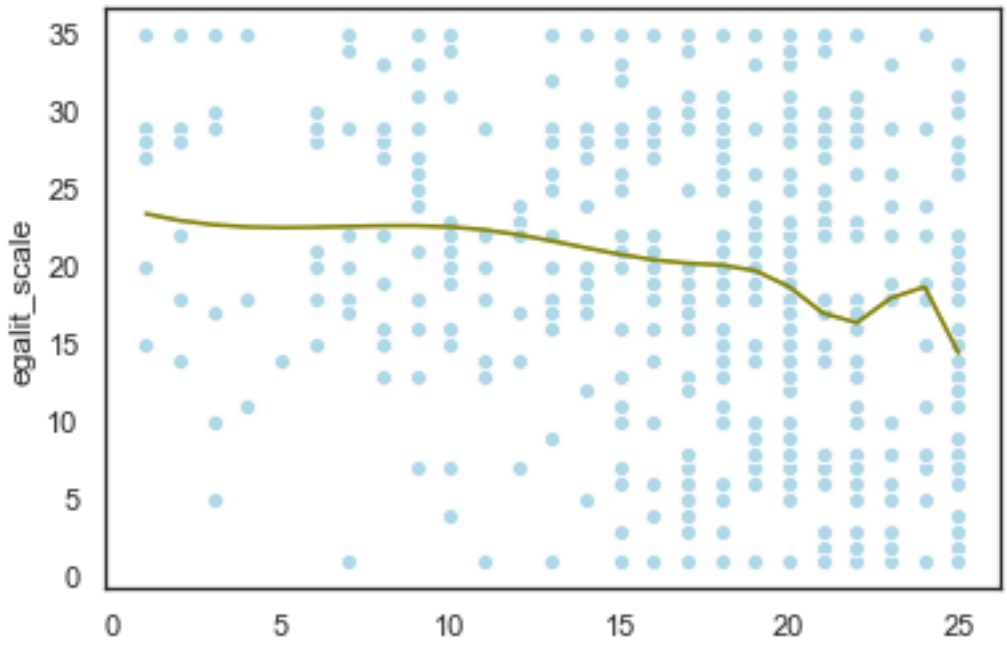
[13]:
```
sp = range(3, 11)
plt.plot(sp, list(spline_dict.values()), color = 'lightblue')
plt.xlabel('Natural Regression Spline (df)')
plt.ylabel('Mean Squared Error')
plt.show()
```



The optimal number of df for natural regression spline is 7.

[14]:
```
transformed_x = dmatrix("bs(train, df=7, degree=3)", {"train": X_train},␣
 ↪return_type='dataframe')
model = lm.fit(transformed_x, y_train)
test_spline = dmatrix("bs(test, df=7, degree=3)", {"test": X_test},␣
 ↪return_type='dataframe')

sns.scatterplot(X_test.ravel(), y_test, color='lightblue')
sns.lineplot(X_test.ravel(), model.predict(test_spline), color = 'olive')
```

[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c18878208>
```



Similar to the polynomial regression and the step function, the predicted value of egalit scale decreases as income06 increases. But natural regression spline produced a smoother fit than the other functions.

0.1.2 Egalitarianim and everything

4.(20 points total) 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):

[15]:
```
y_train = gss_train['egalit_scale']
y_test = gss_test['egalit_scale']
X_train = gss_train.drop('egalit_scale', axis=1)
X_test = gss_test.drop('egalit_scale', axis=1)
```

[16]:
```
scaler = MinMaxScaler(feature_range=(0, 1))

def scale_features(df):
    for column in df:
        if df[column].dtypes == object:
            df[column] = pd.get_dummies(df[column])
        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
```

[17]:
```
X_train = scale_features(X_train)
X_test = scale_features(X_test)

transform_ytr = y_train.values.reshape(-1,1)
scaler.fit(transform_ytr)
y_train = scaler.transform(transform_ytr)

transform_yte = y_test.values.reshape(-1,1)
scaler.fit(transform_yte)
y_test = scaler.transform(transform_yte)
```

a. (5 points) Linear regression

[18]:
```
lm = LinearRegression().fit(X_train, y_train)
scores = cross_val_score(lm, X_train, y_train,␣
 ↪scoring="neg_mean_squared_error", cv=crossvalidation)
lm_mse = np.mean(np.abs(scores))
print("Test MSE for Linear Regression: ", lm_mse)
```

```
Test MSE for Linear Regression:  0.055784762561532183
```

b. (5 points) Elastic net regression

[19]:
```
elasticnet = ElasticNetCV(l1_ratio=np.linspace(.1,1,11), alphas=np.linspace(.
 ↪001,.01,11)).fit(X_train, y_train)
print(elasticnet.alpha_)
print(elasticnet.l1_ratio_)
scores = cross_val_score(elasticnet, X_train, y_train,␣
 ↪scoring="neg_mean_squared_error", cv=crossvalidation)
elas_mse = np.mean(np.abs(scores))
print("Test MSE for Elastic Net: ", elas_mse)
```

```
0.0019000000000000002
0.82
Test MSE for Elastic Net:  0.055324700821125485
```

The regularization path chose alpha at 0.0019 and lambda at 0.82 for elastic net.

c. (5 points) Principal component regression

[20]:
```
pca = PCA(n_components=.95)
X_reduced = pca.fit_transform(X_train)
np.cumsum(np.round(pca.explained_variance_ratio_, decimals=4)*100)
```

[20]:
```
array([16.4 , 25.21, 32.22, 37.1 , 41.63, 45.94, 49.61, 53.07, 56.17,
       59.14, 61.81, 64.27, 66.69, 68.82, 70.94, 72.95, 74.91, 76.81,
       78.59, 80.31, 81.93, 83.52, 85.03, 86.46, 87.82, 89.14, 90.43,
```

```
       91.66, 92.78, 93.82, 94.77, 95.54])
```

PCA selected 32 principal components that explain 95% of the variance.

[21]:
```
reg = LinearRegression()
scores = cross_val_score(reg, X_reduced[:,:32], y_train,␣
 ↪scoring="neg_mean_squared_error", cv=crossvalidation)
pcr_mse = np.mean(np.abs(scores))
print("Test MSE for Principal Component Regression: ", pcr_mse)
```

```
Test MSE for Principal Component Regression:  0.056004354755577745
```

d. (5 points) Partial least squares regression

[22]:
```
min_comp = 20
min_score = 100
for i in np.arange(1, 21):
    pls = PLSRegression(n_components=i)
    scores = cross_val_score(pls, X_train, y_train,␣
 ↪scoring="neg_mean_squared_error", cv=crossvalidation)
    pls_mse = np.mean(np.abs(scores))
    if pls_mse < min_score:
        min_comp = i
        min_score = pls_mse

min_comp
```

[22]:
```
6
```

The optimal number of components for PLS regression is 6.

[23]:
```
pls = PLSRegression(n_components=6)
scores = cross_val_score(pls, X_train, y_train,␣
 ↪scoring="neg_mean_squared_error", cv=crossvalidation)
pls_mse = np.mean(np.abs(scores))
print("Test MSE for PLS Regression: ", pls_mse)
```

```
Test MSE for PLS Regression:  0.05572252235458728
```

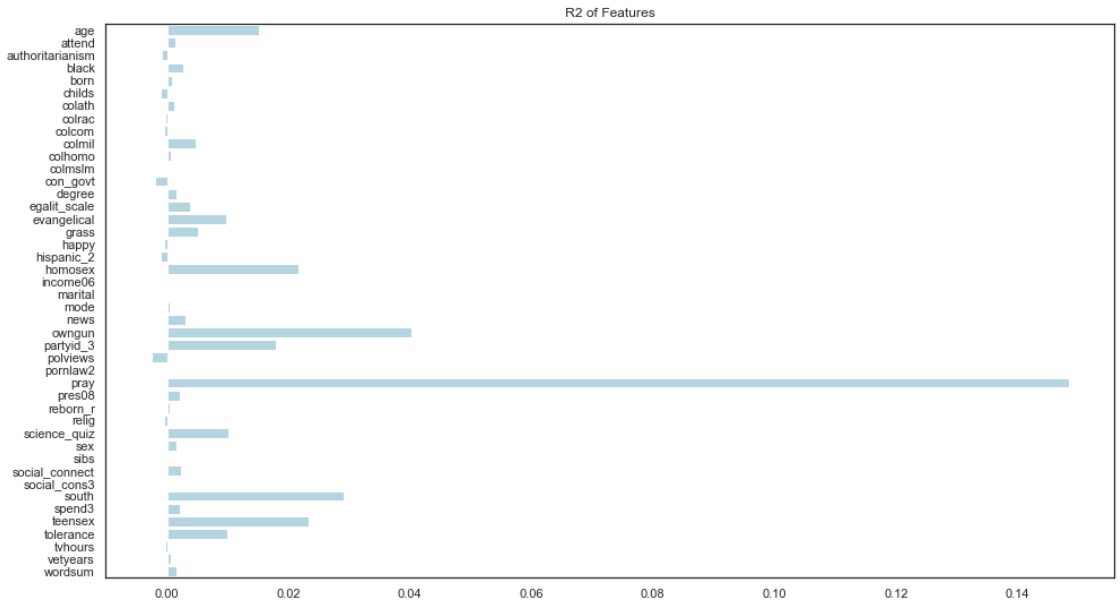
5.(20 points) For each final tuned version of each model fit, evaluate feature importance by generating feature interaction plots. Upon visual presentation, be sure to discuss the substantive results for these models and in comparison to each other (e.g., talk about feature importance, conditional effects, how these are ranked differently across different models, etc.).

[24]:
```
from mlxtend.evaluate import feature_importance_permutation
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values = np.nan, strategy = 'mean',verbose=0)
imputer = imputer.fit(X_test)
X_test = imputer.transform(X_test)

sns.set(rc={'figure.figsize':(16,9)})
```

Since Python was not able to product feature interaction plots, I will evaluate feature importance based on permutation importance by looking at the explained variance of each feature (using R2) for each model.

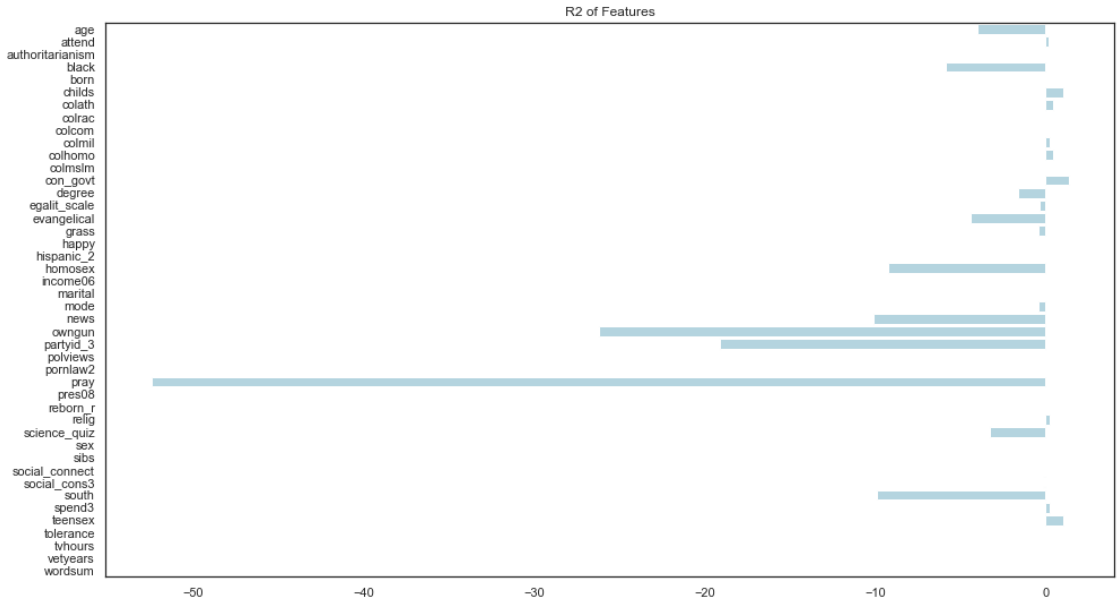
[25]:
```
def plot_imp(model):
    imp_vals, _ = feature_importance_permutation(
        predict_method=model.predict,
        X=X_test,
        y=y_test,
        metric='r2', # r2 metrics is recommended for regressors
        num_rounds=10)

    col = []
    imp = []
    for i in range(X_test.shape[1]):
        col.append(gss_test.columns[i])
        imp.append(imp_vals[i])

    sns.set_style("white")
    ax = sns.barplot(x=imp, y=col, color='lightblue').set_title("R2 of␣
 ↪Features")

    return ax
```

[26]:
```
lm_plot = plot_imp(lm)
```



For Linear Regression, the features that explained most of the variance of the model are "pray", "owngun", "age", and "south".

[27]:
```
elasticnet_plot = plot_imp(elasticnet)
```

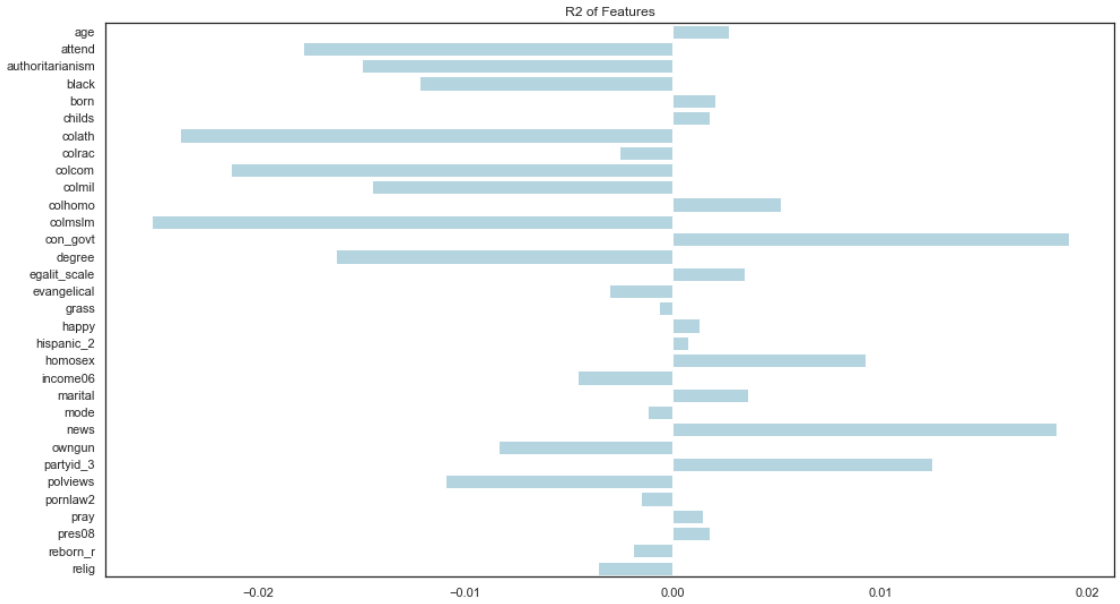


For Elastic Net Regression, most of the features did not work well to explain the variance of the model. A few features that obtained positive R2 values are "teensex", "childs", and "con_govt".

[28]:
```
reg.fit(X_reduced[:,:32], y_train)
```

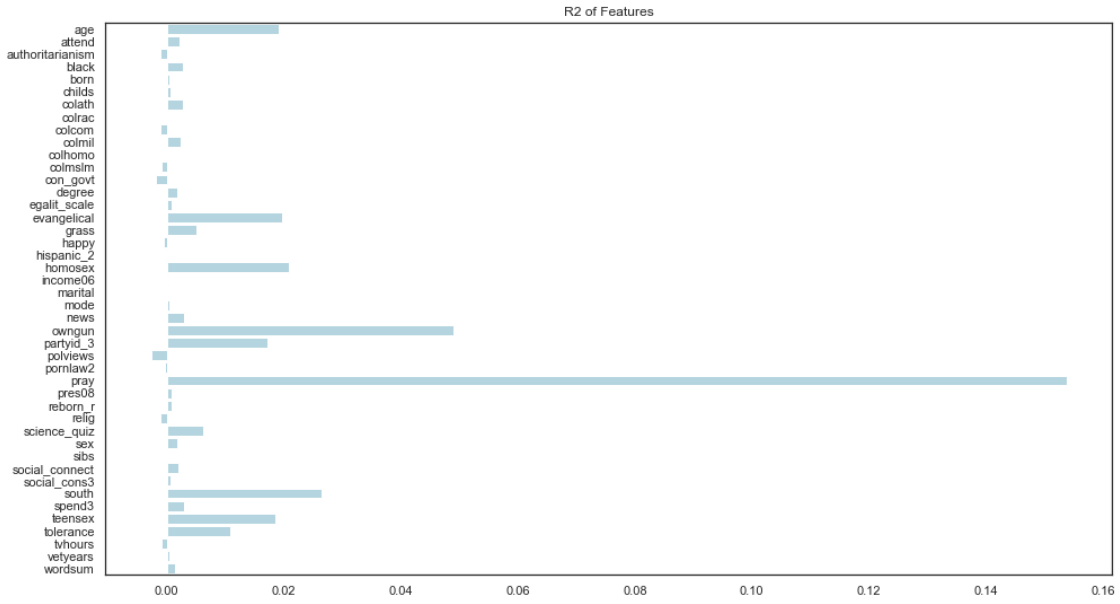
[28]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

[29]:
```
imp_vals, _ = feature_importance_permutation(
        predict_method=reg.predict,
        X=X_test[:,:32],
        y=y_test,
        metric='r2', # r2 metrics is recommended for regressors
        num_rounds=10)

col = []
imp = []
for i in range(32):
    col.append(gss_test.columns[i])
    imp.append(imp_vals[i])

sns.set_style("white")
ax = sns.barplot(x=imp, y=col, color='lightblue').set_title("R2 of Features")
```



Like Elastic Net Regression, many features did not work well to explain the variance of PCR model. The prominent features that obtained positive R2 include "news", "con_govt", and "partyid_3".

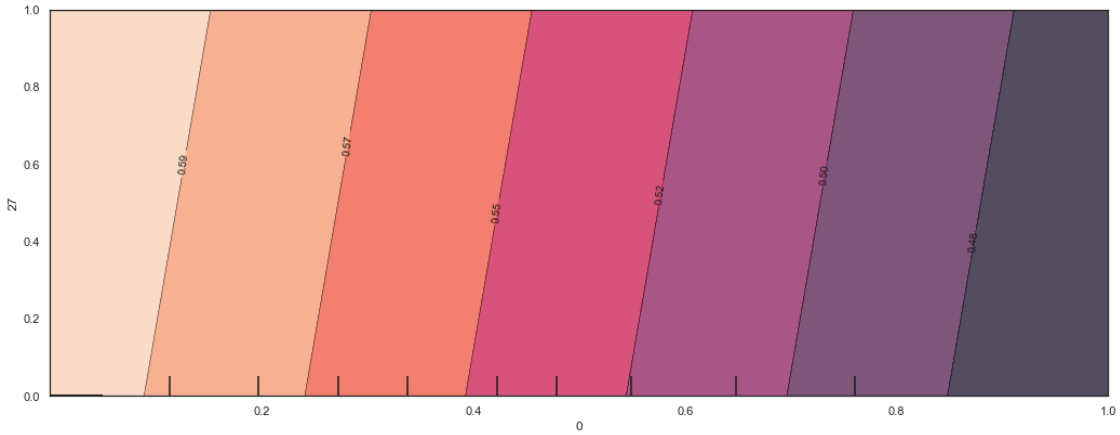
[30]:
```
pls.fit(X_train, y_train)
pls_plot = plot_imp(pls)
```

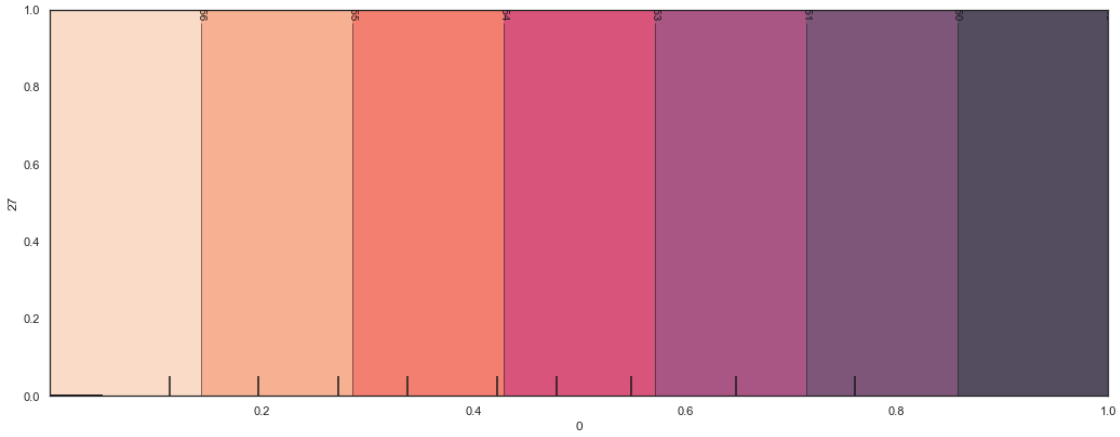


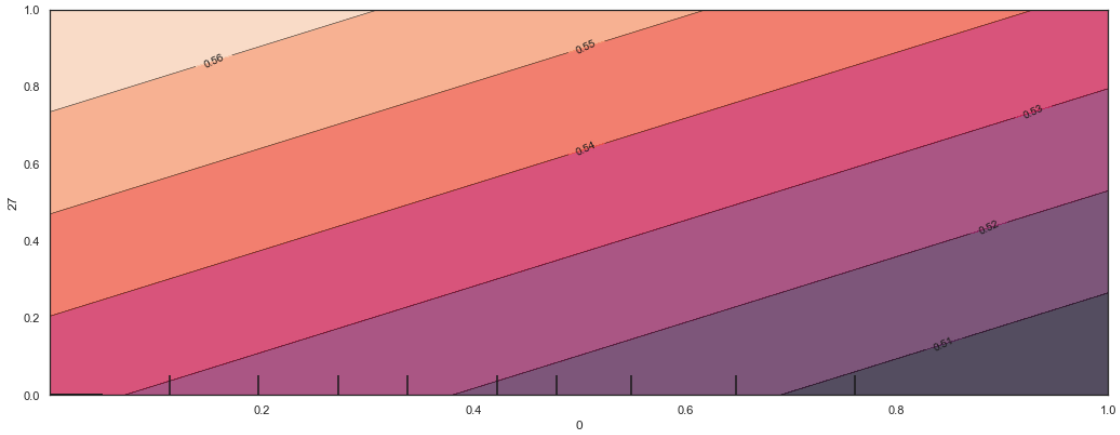
For PLS Regression, the features that explained most of the variance of the model are "pray", "owngun", "age", and "south". The R2 scores are similar to those of Linear Regression. In both models, "pray" is much more prominent than the other features.

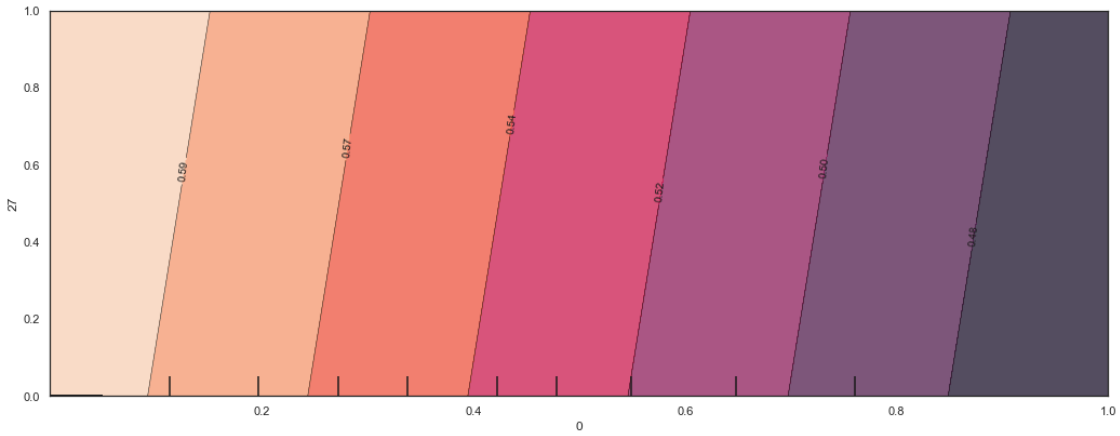
There are a few features that occured in all four models based on permutation importance – “age”, “pray”, “con-govt”, and “south”. These features explained most of the variance of the model, and may even produced interaction effects. Here, we will plot a partial dependence plot to look at the interaction effect of “age” and “pray”.

[31]:
```
from sklearn.inspection import plot_partial_dependence
# 0: age, 27: pray
plot_partial_dependence(lm, X_test, [(0,27)])
plot_partial_dependence(elasticnet, X_test, [(0,27)])
plot_partial_dependence(reg, X_test[:,:32], [(0,27)])
plot_partial_dependence(pls, X_test, [(0,27)])
```









We ploted the interaction among “pray”(27) and “age”(0) in four different models. From the plots, we can see that the interactions happened in very different ways. In Elastic net regression, “pray” is nearly independent of “age” at every level. This is consistent with their R2 values, in which both features did not explain well the variance in elastic model. The interactions in linear regression and PLS regression tend to be linear. This is consistent with our previous findings. The reason why the interaction is not strong might be due to the fact that “pray” is much more prominent than the other features so that changes in other features may not induce a strong effect on the levels of “pray”. The interaction among the two features is most obvious in PCR model, although the interaction still tends to be linear. The stronger effect might be due to the fact that the difference in R2 scores obtained by the two features is smaller than those in other models, and so the two features will be able to interact more.