Xu_Weijie_HW4

February 16, 2020

1 Egalitarianism and Income

1.1 Task 1

The best degree derived from cross validation is 2. Furthermore, as we can see from both the graph of the fitted model and the graph of the marginal effect, the score of egalitarianism is decreasing long with the increase of income. That is, rich people is more reluctant to the egalitarinisam compared with others. The average marginal effect of income on egalit_scale is -0.317

[1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import KFold, cross_val_score
from sklearn.base import BaseEstimator
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import ElasticNetCV
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import scale
from sklearn.decomposition import PCA
from sklearn.cross_decomposition import PLSRegression
from mlxtend.evaluate import feature_importance_permutation
from sklearn.impute import SimpleImputer
from sklearn.inspection import plot_partial_dependence
from patsy import dmatrix
```

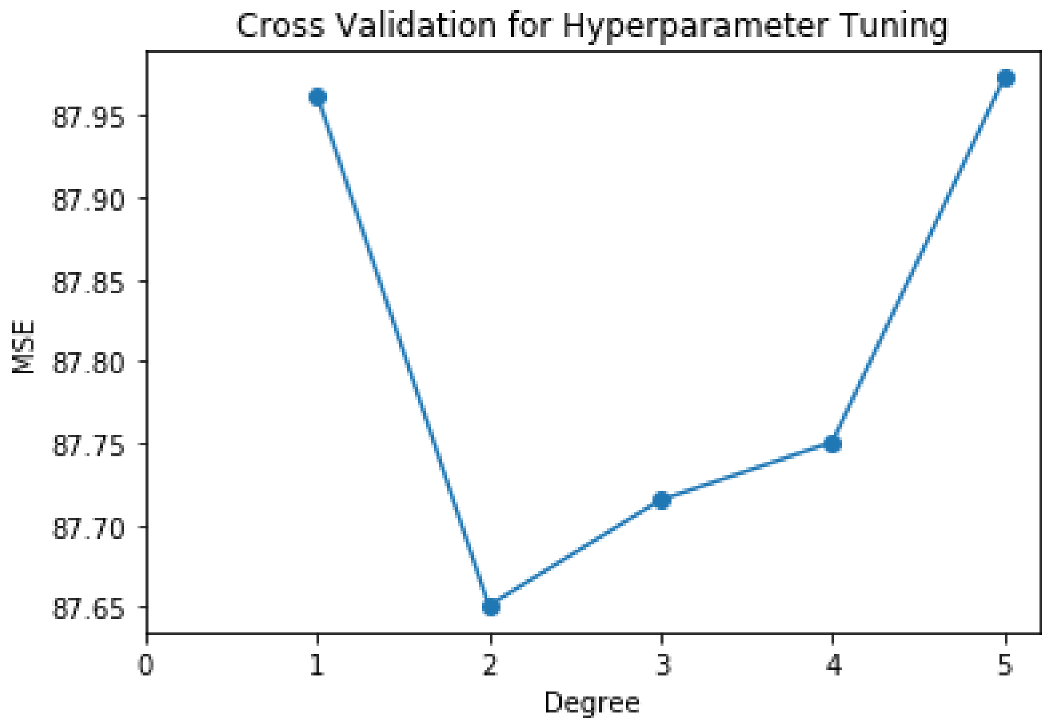
[2]:
```python
SEED = 19970608
```

[3]:
```python
df_train = pd.read_csv('data/gss_train.csv')
df_test = pd.read_csv('data/gss_test.csv')
```

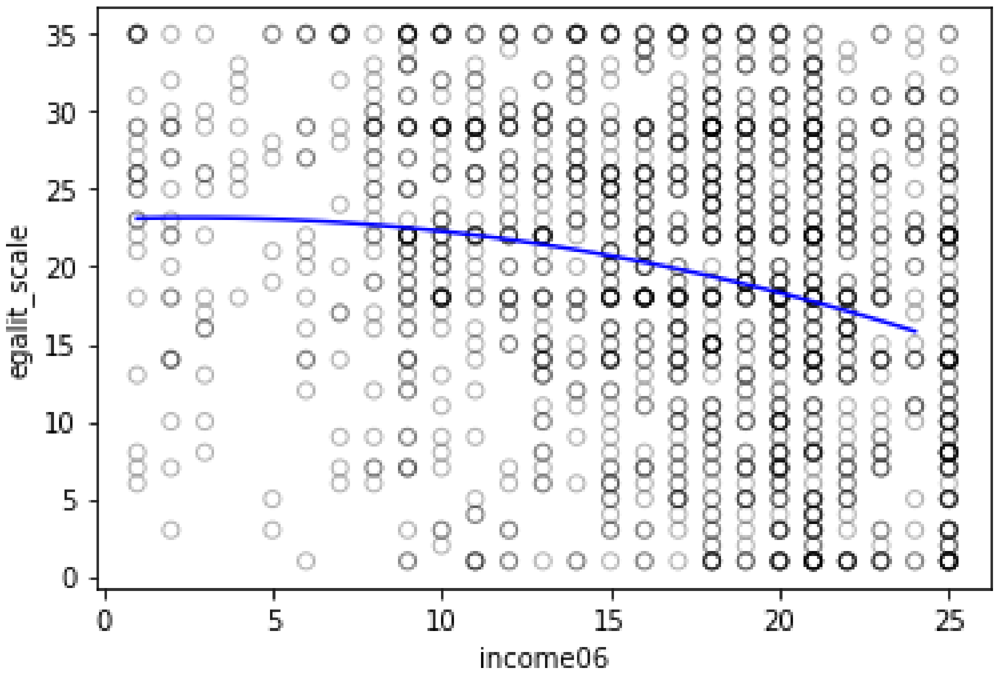
[4]:
```python
X_train, X_test = df_train['income06'], df_test['income06']
y_train, y_test = df_train['egalit_scale'], df_test['egalit_scale']
```

[5]:
```python
kf_10 = KFold(n_splits=10, random_state=SEED)

mse_poly = []
for degree in range(1, 6):
    lm_poly = LinearRegression()
    score = -1*cross_val_score(lm_poly,
                               np.vander(X_train, N=degree + 1), y_train,␣
 ↪cv=kf_10,
                               scoring='neg_mean_squared_error').mean()
    mse_poly.append(score)

plt.plot(list(range(1, 6)), mse_poly, '-o')
plt.xlabel('Degree')
plt.ylabel('MSE')
plt.title('Cross Validation for Hyperparameter Tuning')
plt.xlim(xmin=0)
```

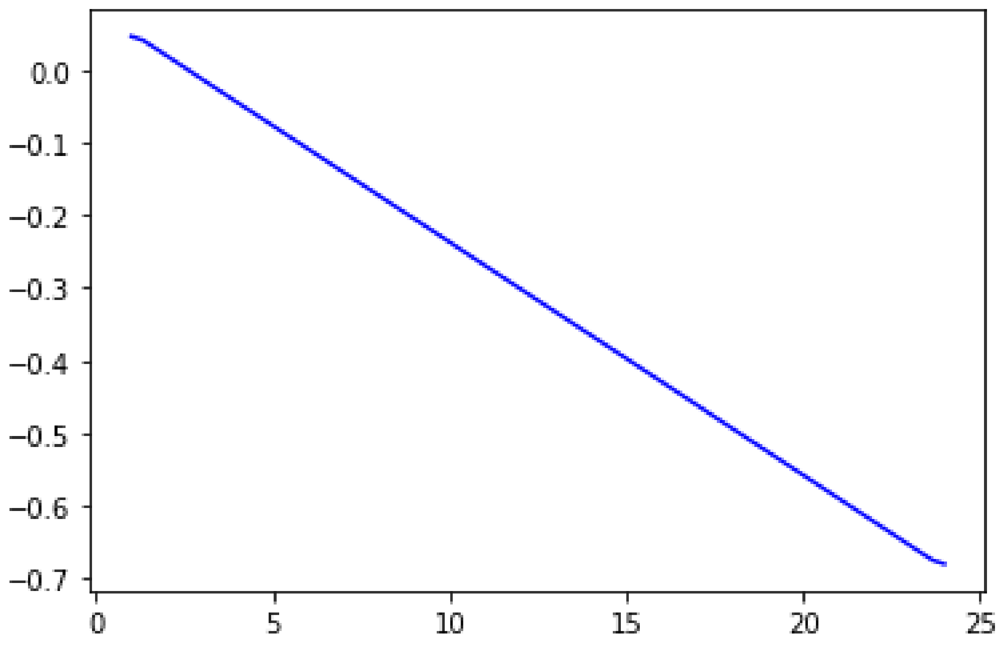
```
[5]: (0, 5.2)
```



```
[6]: xp = np.linspace(X_train.min(),X_train.max()-1,70)
     lm_poly.fit(np.vander(X_train, N=2 + 1), y_train)
     pred_poly = lm_poly.predict(np.vander(xp, N=3))

     plt.scatter(X_train, y_train, facecolor='None', edgecolor='k', alpha=0.3)
     plt.plot(xp, pred_poly, c='b')
     plt.xlabel('income06')
     plt.ylabel('egalit_scale')
plt.show()
```



[7]:
```
marginal = np.gradient(pred_poly, xp)
plt.plot(xp, marginal, c='b')
plt.show()
```



```
[8]: avg_marginal = marginal.mean()
     print(avg_marginal)
```

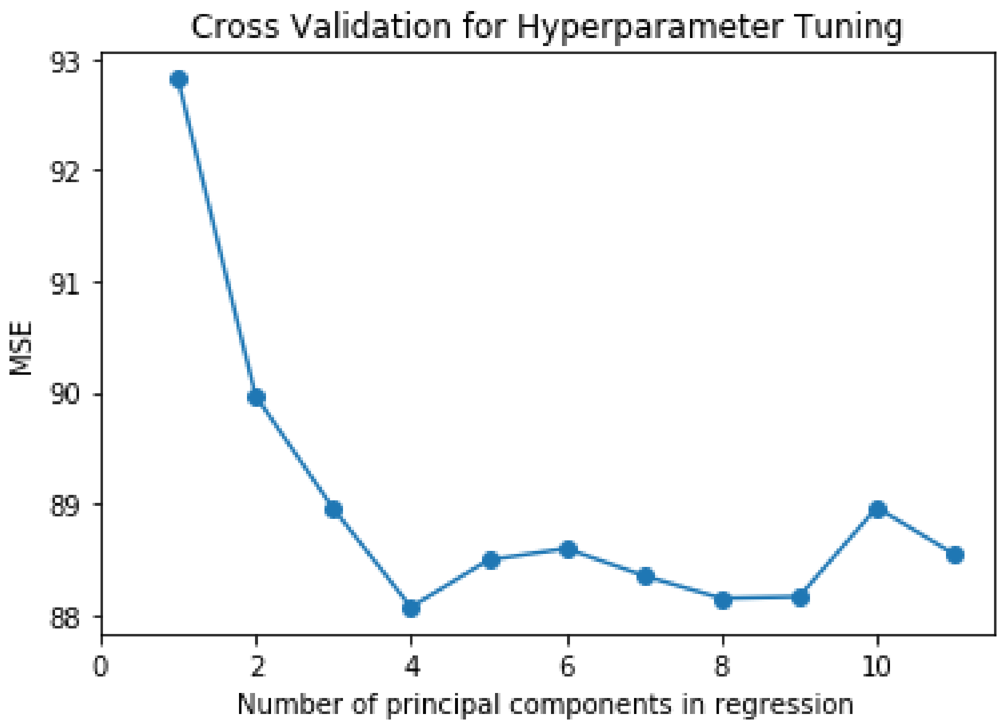
```
-0.3170024448385364
```

1.2 Task 2

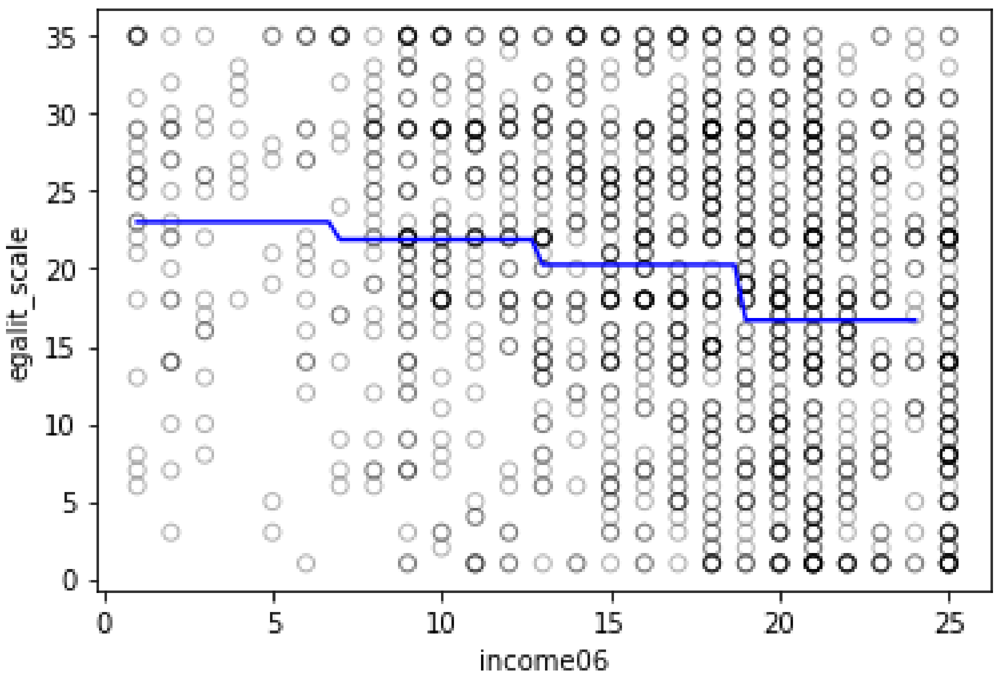
The best number of cuts derived from cross validation is 4, with each step having a decreasing trend of constant to represent. This decreasing trend is similar to what we get in Task 1, which could also lead to the claim that rich people are more reluctant to egalitarinism.

```
[9]: kf_10 = KFold(n_splits=10, random_state=SEED)

     mse_step = []
     for i in np.arange(1, 12):
         X_cut, bins = pd.cut(X_train, i, retbins=True, right=True)
         X_cut.value_counts(sort=False)
         X_steps_dummies = pd.get_dummies(X_cut)
         lm_step = LinearRegression()
         score = -1*cross_val_score(lm_step, X_steps_dummies, y_train, cv=kf_10,␣
      ↪scoring='neg_mean_squared_error').mean()
         mse_step.append(score)

     plt.plot(list(range(1, 12)), mse_step, '-o')
     plt.xlabel('Number of principal components in regression')
     plt.ylabel('MSE')
     plt.title('Cross Validation for Hyperparameter Tuning')
     plt.xlim(xmin=0)
```

```
[9]: (0, 11.5)
```



[10]:
```
X_cut_train, bins = pd.cut(X_train, 4, retbins=True, right=True)
X_cut_train.value_counts(sort=False)
X_steps_dummies_train = pd.get_dummies(X_cut_train)
lm_step = LinearRegression()
lm_step.fit(X_steps_dummies_train, y_train)

xp = np.linspace(X_train.min(),X_train.max()-1,70)
bin_mapping = np.digitize(xp, bins)
X_steps_plot = pd.get_dummies(bin_mapping)
pred_step = lm_step.predict(X_steps_plot)

plt.scatter(X_train, y_train, facecolor='None', edgecolor='k', alpha=0.3)
plt.plot(xp, pred_step, c='b')
plt.xlabel('income06')
plt.ylabel('egalit_scale')
plt.show()
```

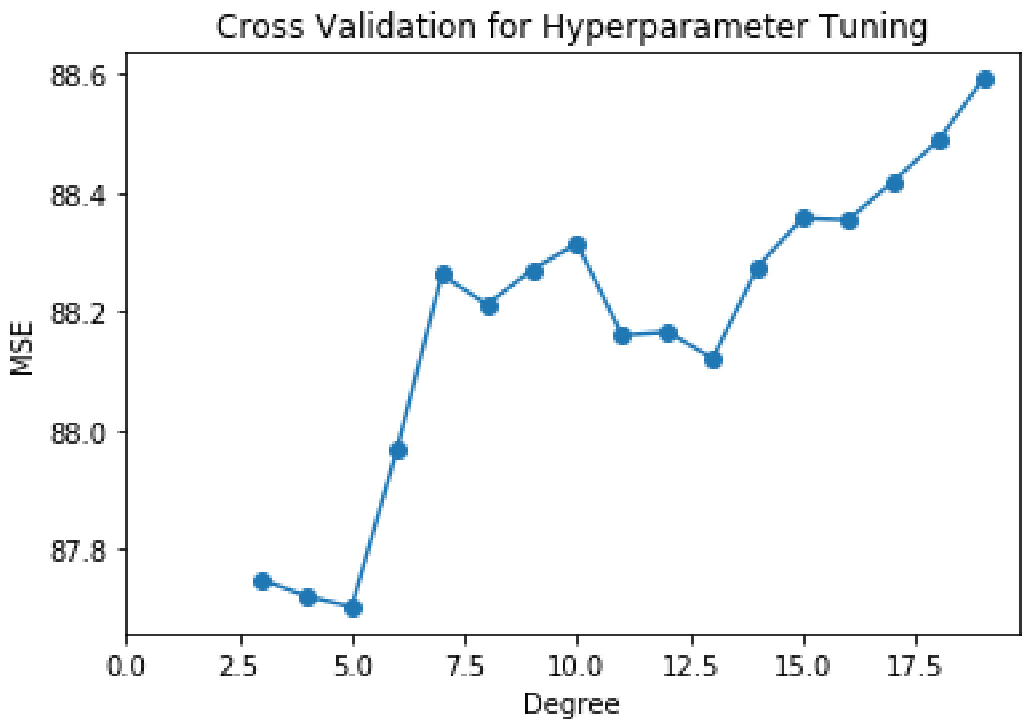


1.3 Task 3

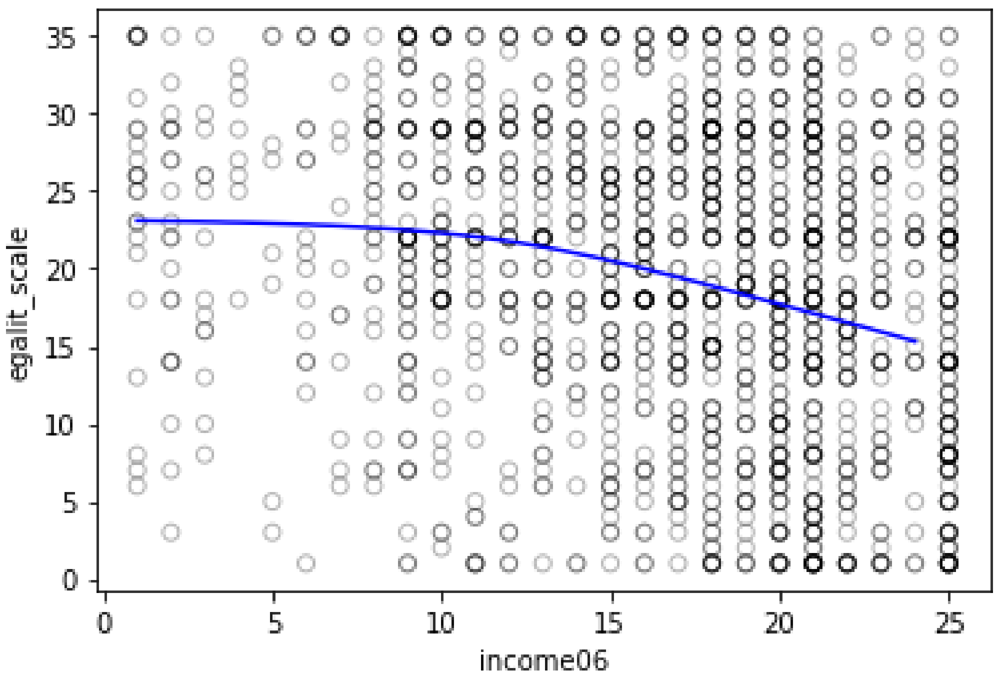
The optiminal degree of freedom derived from cross validation is 5. As we can see from the result of the fitted model, the score of egalitarinism and income is negatively related, which could lead to the conclusion that rich people are more reluctant to egalitarinism.

```
[11]: kf_10 = KFold(n_splits=10, random_state=SEED)

mse_spline = []

for degree in range(3, 20):
    X_train_trans = dmatrix("cr(X_train, df={})".format(degree), {"X_train":␣
 ↪X_train}, return_type='dataframe')
    lm_spline = LinearRegression()
    score = -1*cross_val_score(lm_spline, X_train_trans, y_train, cv=kf_10,␣
 ↪scoring='neg_mean_squared_error').mean()
    mse_spline.append(score)

plt.plot(list(range(3, 20)), mse_spline, '-o')
plt.xlabel('Degree')
plt.ylabel('MSE')
plt.title('Cross Validation for Hyperparameter Tuning')
plt.xlim(xmin=0)
```

```
[11]: (0, 19.8)
```



```
[12]: X_train_trans = dmatrix("cr(X_train, df=5)", {"X_train": X_train},␣
       ↪return_type='dataframe')
      lm_spline = LinearRegression()
      lm_spline.fit(X_train_trans, y_train)

      xp = np.linspace(X_train.min(),X_train.max()-1,70)
      xp_trans = dmatrix("cr(xp, df=5)", {"xp": xp}, return_type='dataframe')
      pred_spline = lm_spline.predict(xp_trans)

      plt.scatter(X_train, y_train, facecolor='None', edgecolor='k', alpha=0.3)
      plt.plot(xp, pred_spline, c='b')
      plt.xlabel('income06')
      plt.ylabel('egalit_scale')
      plt.show()
```



2 Egalitarianism and everything

2.1 Task 4

2.1.1 a.

The MSE derived from 10-fold cross-validation is 84.2

[13]:
```
new_df_train = df_train.select_dtypes(include=['float64', 'int'])
new_df_test = df_test.select_dtypes(include=['float64', 'int'])
X_train, X_test = new_df_train.drop('egalit_scale', axis=1), new_df_test.
 ↪drop('egalit_scale', axis=1)
y_train, y_test = new_df_train['egalit_scale'], new_df_test['egalit_scale']
```

[14]:
```
for col in X_train:
    X_train[col] = scale(X_train[col])
for col in X_test:
    X_test[col] = scale(X_test[col])
```

[15]:
```
lm = LinearRegression()
lm.fit(X_train, y_train)
```

[15]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
[16]: lm_cv_scores = cross_val_score(lm, X_train, y_train,
                                scoring='neg_mean_squared_error', cv=10)
      avg_accuracy_lm_cv = -lm_cv_scores.mean()
```

```
[17]: avg_accuracy_lm_cv
```

```
[17]: 84.20369084148085
```

2.1.2 b.

The best l1_ratio (alpha) is 0.5, the best lambda is 0.28 The MSE derived from 10-fold cross-validation is 84.17

```
[18]: lm_elastic = ElasticNetCV(cv=10)
      lm_elastic.fit(X_train, y_train)
      print('Alpha: {}; Lambda: {}'.format(lm_elastic.l1_ratio_, lm_elastic.alpha_))
```

```
Alpha: 0.5; Lambda: 0.09808702749002161
```

```
[19]: elastic_cv_scores = cross_val_score(lm_elastic, X_train, y_train,
                                scoring='neg_mean_squared_error', cv=10)
      avg_accuracy_elastic_cv = -elastic_cv_scores.mean()
```

```
[20]: avg_accuracy_elastic_cv
```

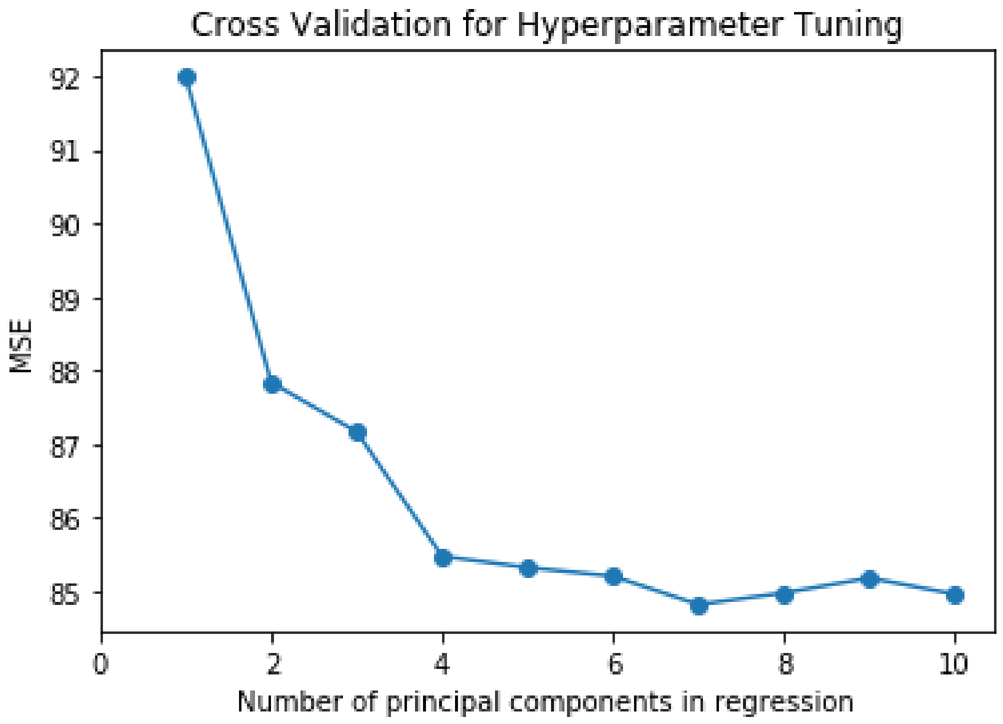
```
[20]: 84.17401759237762
```

2.1.3 c.

the best lambda is 7.
The MSE derived from 10-fold cross-validation is 84.2

```
[21]: pca = PCA()
      X_reduced_train = pca.fit_transform(X_train)
      n = len(X_reduced_train)

      lm_pcr = LinearRegression()

      kf_10 = KFold(n_splits=10, random_state=SEED)
      mse_pcr = []
      for i in np.arange(1, 11):
          score = -1*cross_val_score(lm_pcr, X_reduced_train[:,:i], y_train,␣
       ↪cv=kf_10, scoring='neg_mean_squared_error').mean()
          mse_pcr.append(score)

      plt.plot(list(range(1, 11)), mse_pcr, '-o')
      plt.xlabel('Number of principal components in regression')
      plt.ylabel('MSE')
      plt.title('Cross Validation for Hyperparameter Tuning')
      plt.xlim(xmin=0)
```

[21]:
```
(0, 10.45)
```



[22]:
```
lm_pcr = LinearRegression()
lm_pcr.fit(X_reduced_train[:,:8], y_train)
```

[22]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

[23]:
```
pcr_cv_scores = cross_val_score(lm_pcr, X_train, y_train,
                                scoring='neg_mean_squared_error', cv=10)
avg_accuracy_pcr_cv = -pcr_cv_scores.mean()
```

[24]:
```
avg_accuracy_pcr_cv
```

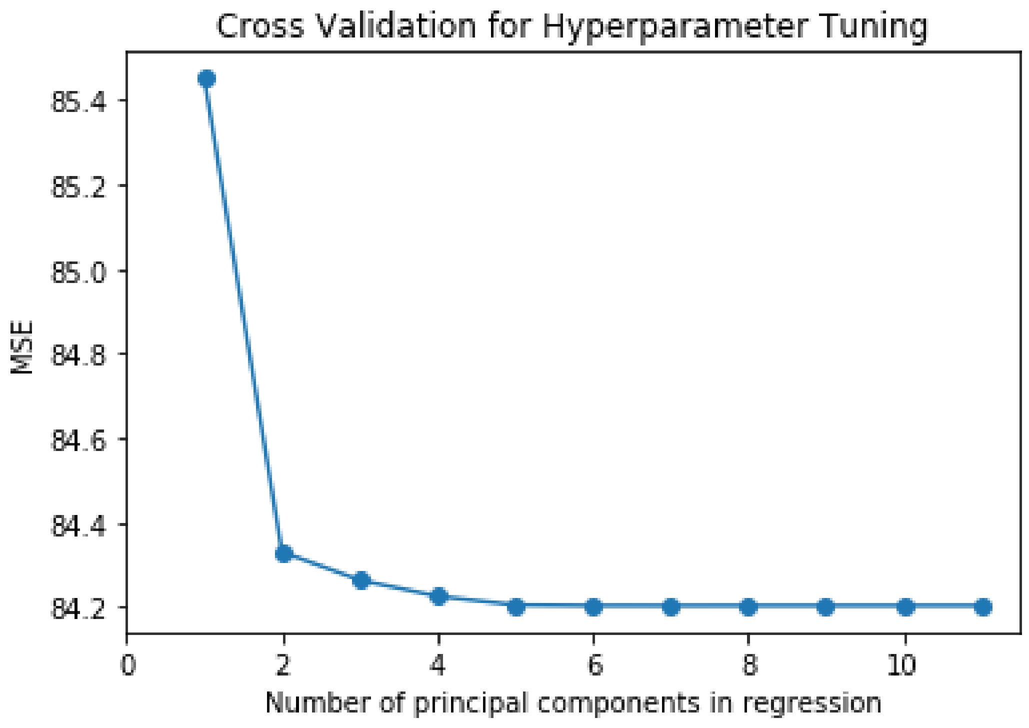
[24]:
```
84.20369084148085
```

2.1.4 d.

The model starts to land on its minimum MSE then the lambda is 5. The MSE derived from 10-fold cross-validation is 84.2

```
[25]: mse_pls = []
      kf_10 = KFold(n_splits=10, random_state=SEED)
      for i in range(1, 12):
          pls = PLSRegression(n_components=i)
          score = - cross_val_score(pls, scale(X_train), y_train, cv=kf_10,␣
       ↪scoring='neg_mean_squared_error').mean()
          mse_pls.append(score)

      plt.plot(list(range(1, 12)), mse_pls, '-o')
      plt.xlabel('Number of principal components in regression')
      plt.ylabel('MSE')
      plt.title('Cross Validation for Hyperparameter Tuning')
      plt.xlim(xmin=0)
```

```
[25]: (0, 11.5)
```



```
[26]: lm_pls = PLSRegression(n_components=5)
      lm_pls.fit(scale(X_train), y_train)
```

```
[26]: PLSRegression(copy=True, max_iter=500, n_components=5, scale=True, tol=1e-06)
```

```
[27]: pls_cv_scores = cross_val_score(lm_pls, X_train, y_train,
                                      scoring='neg_mean_squared_error', cv=10)
```

```
avg_accuracy_pls_cv = -pls_cv_scores.mean()
```

[28]:
```
avg_accuracy_pls_cv
```

[28]:
```
84.20542013298933
```

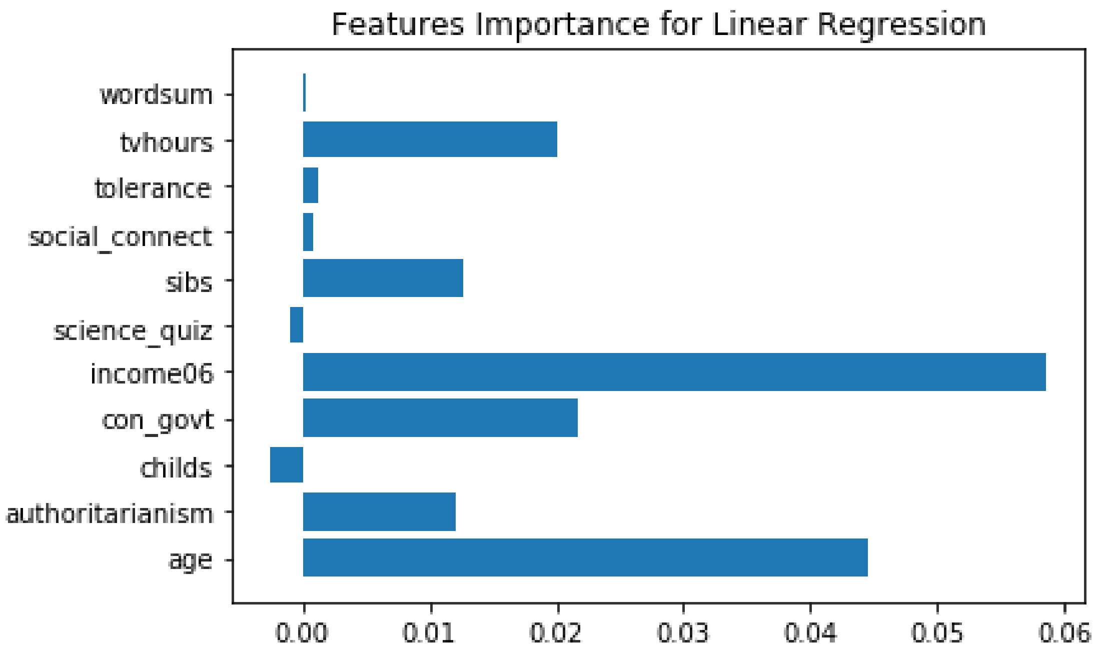
2.2 Task 5

The feature importance of linear regression and elastic net regression can be seen from the graphs below. For these models,“income06” always has the highest importance. Other features such as “age”, “con_govt”, “tvhours”, “sibs”, and “authoritarianism” also have obvious contribution to the model with decreasing order. For linear, elastic net, and PLS model, we also plot the partial dependency between “income06” and other important features mentioned above. As we can see in the graphs, for all the models, “income” has an obvious interaction with “tvhour” and “sibs”.

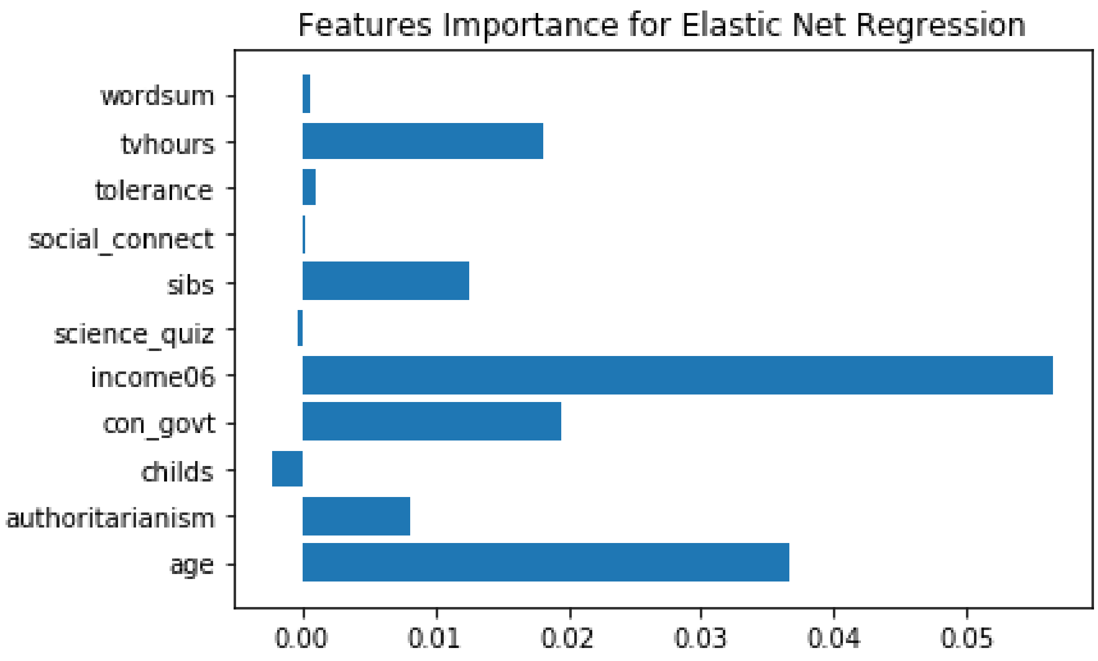
[29]:
```
imputer = SimpleImputer(missing_values = np.nan, strategy = 'mean',verbose=0)
imputer = imputer.fit(X_test)
X_test_imputed = imputer.transform(X_test)
```

[30]:
```
def plot_feature_importance(model, model_name):
    imp_vals, _ = feature_importance_permutation(predict_method=model.predict,␣
 ↪X=X_test_imputed, y=y_test, metric='r2', seed=SEED)
    plt.barh(X_test.columns, imp_vals, align='center')
    plt.title('Features Importance for {}'.format(model_name))
    plt.show()
```

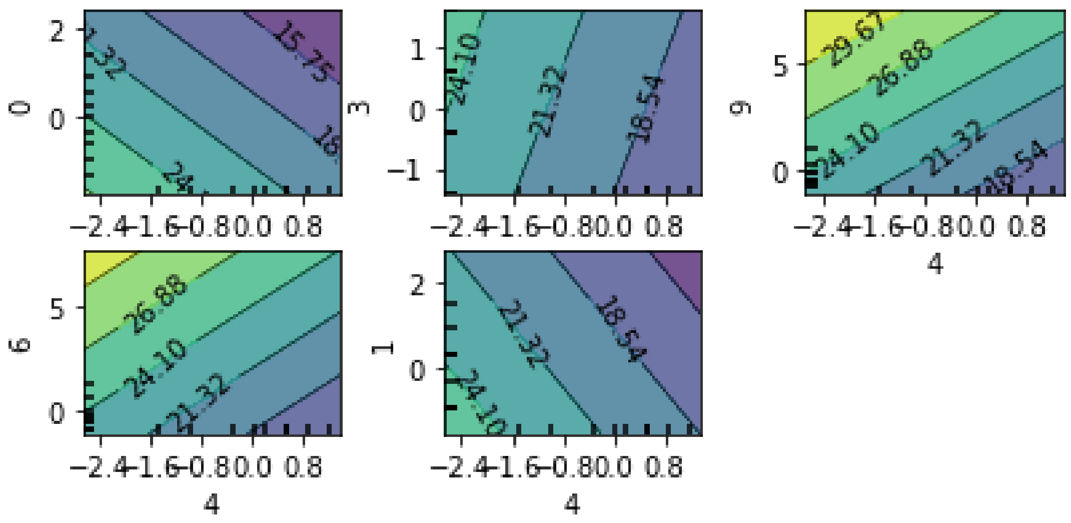
[31]:
```
plot_feature_importance(lm, 'Linear Regression')
```



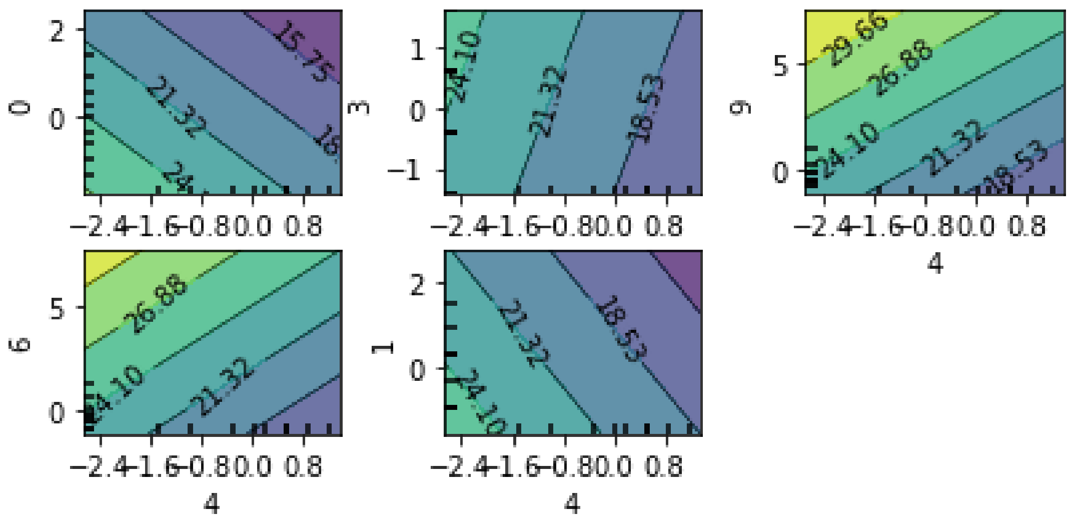
[32]:
```
plot_feature_importance(lm_elastic, 'Elastic Net Regression')
```



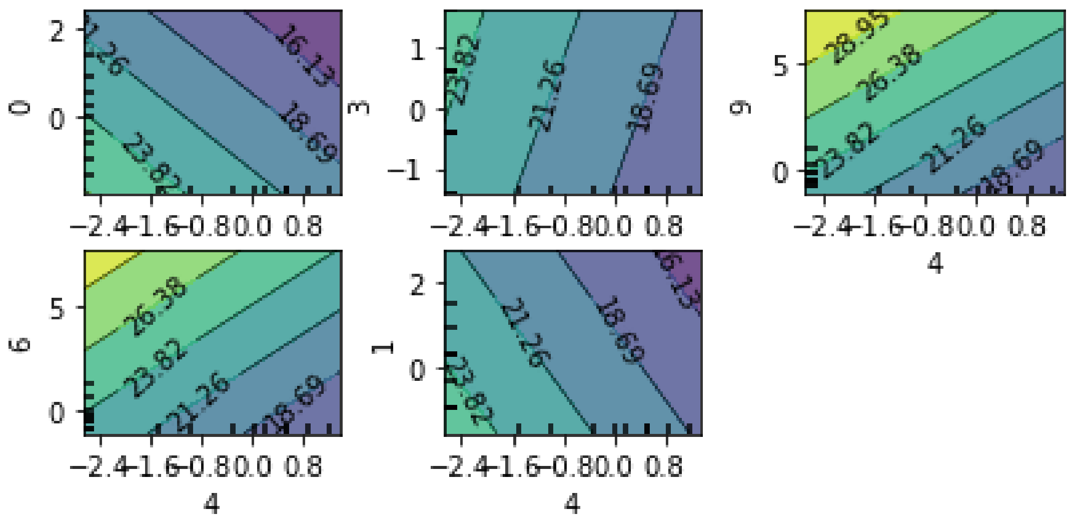
[33]:
```
features = [(4,0), (4,3), (4,9),(4,6),(4,1)]
plot_partial_dependence(lm_pls, X_test, features)
```



[34]:
```
plot_partial_dependence(lm, X_test, features)
```



[35]:
```
plot_partial_dependence(lm_elastic, X_test, features)
```



[ ]: