1.0-jo-initial-data-exploration

April 21, 2022

1 Gender Pay Gap

Team members: * Jordan Farrell * Julio Oliveira * Ashwath Ramesh * Satoshi Taniguchi * Junjun Tao * William Teodecki

[ ]:
```
import numpy as np
import pandas as pd

import shap
from fitter import Fitter
import statsmodels.api as sm
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LinearRegression

from sklearn.metrics import r2_score, mean_squared_error
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split


import seaborn as sns
import matplotlib.pyplot as plt
```

```
/home/julio/.local/share/virtualenvs/gender-pay-gap-VcKyyecI/lib/python3.8/site-
packages/tqdm/auto.py:22: TqdmWarning: IProgress not found. Please update
jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

[ ]:
```
columns = [
    'sex', 'age', 'annhrs', 'annlabinc', 'white', 'black', 'hisp', 'degree',
    'yrsexp', 'yrsftexp', 'yrsptexp', 'hrwage', 'northeast', 'northcentral',
    'south', 'west', 'manager', 'business', 'financialop', 'computer',
    'architect', 'scientist', 'socialworker', 'postseceduc', 'legaleduc',
    'artist', 'lawyerphysician', 'healthcare', 'healthsupport', 'protective',
    'foodcare', 'building', 'sales', 'officeadmin', 'constructextractinstall',
    'production', 'transport','farminc', 'labincbus'
```

```
]

psid = pd.read_csv('../data/external/PanelStudyIncomeDynamics.csv',
                   usecols=columns)
```

[ ]:
```
psid.head()
```

[ ]:
```
   sex  farminc  age  annhrs  annlabinc  white  black  hisp  degree  \
0    1      0.0   34    1600    10000.0      1      0     0     1.0
1    1      0.0   32     520     9095.0      0      1     0     0.0
2    1      0.0   64    2550    45200.0      0      1     0     0.0
3    1      0.0   50    3072    25000.0      1      0     0     0.0
4    1      0.0   26    2100    24500.0      1      0     0     0.0

   labincbus  …  healthcare  healthsupport  protective  foodcare  building  \
0          0  …           0              0           0         0         0
1          0  …           0              0           0         0         0
2          0  …           0              0           0         0         0
3          0  …           0              0           0         0         0
4          0  …           0              0           0         0         0

   sales  officeadmin  constructextractinstall  production  transport
0      1            0                        0           0          0
1      0            0                        0           1          0
2      0            0                        0           1          0
3      0            0                        0           0          1
4      0            0                        0           1          0

[5 rows x 39 columns]
```

1.0.1 Column Encoding

[ ]:
```
def get_category(row, columns):
    for col in columns:
        if row[col] == 1:
            return col


race_cols = ['white', 'black', 'hisp']
region_cols = ['northeast', 'northcentral', 'south', 'west']
job_cols = [
    'manager', 'business', 'financialop', 'computer', 'architect', 'scientist',
    'socialworker', 'postseceduc', 'legaleduc', 'artist', 'lawyerphysician',
    'healthcare', 'healthsupport', 'protective', 'foodcare', 'building',
    'sales', 'officeadmin', 'constructextractinstall', 'production',
    'transport'
]
```

```
psid['race'] = psid.apply(lambda x: get_category(x, race_cols), axis=1)
psid.drop(race_cols, axis=1, inplace=True)

psid['region'] = psid.apply(lambda x: get_category(x, region_cols), axis=1)
psid.drop(region_cols, axis=1, inplace=True)

psid['job'] = psid.apply(lambda x: get_category(x, job_cols), axis=1)
psid.drop(job_cols, axis=1, inplace=True)

psid['sex'] = psid.apply(lambda x: 'male' if x['sex'] == 1 else 'female',
                         axis=1)

degree_map = {0: 'no_college', 1: 'bachelors', 2: 'advanced_degree'}
psid['degree'] = psid.degree.map(degree_map)
```

[ ]:
```
psid.shape
```

[ ]:
```
(33398, 14)
```

[ ]:
```
psid.head()
```

[ ]:
```
    sex  farminc  age  annhrs  annlabinc      degree  labincbus  yrsexp  \
0  male      0.0   34    1600    10000.0   bachelors          0    12.0
1  male      0.0   32     520     9095.0  no_college          0    14.0
2  male      0.0   64    2550    45200.0  no_college          0    39.0
3  male      0.0   50    3072    25000.0  no_college          0    30.0
4  male      0.0   26    2100    24500.0  no_college          0     8.0

   yrsftexp  yrsptexp     hrwage   race        region         job
0      12.0       0.0   6.250000  white     northeast       sales
1      11.0       3.0  17.490385  black  northcentral  production
2      38.0       1.0  17.725491  black     northeast  production
3      30.0       0.0   8.138021  white  northcentral   transport
4       8.0       0.0  11.666667  white         south  production
```

1.0.2 Income Columns

[ ]:
```
income_cols = ['farminc','annlabinc','labincbus', 'annhrs','hrwage']
psid[income_cols].describe()
```

[ ]:
```
            farminc     annlabinc     labincbus        annhrs        hrwage
count  33398.000000  3.339800e+04  33398.000000  33398.000000  33398.000000
mean     104.879544  3.708689e+04    168.051859   1990.103449     18.418722
std     2662.001098  4.156487e+04   2257.977695    623.592732     19.462814
min    -5000.000000  3.000000e+01      0.000000     10.000000      0.891473
25%        0.000000  1.600000e+04      0.000000   1767.000000      8.823529
```

```
50%          0.000000  2.900000e+04      0.000000   2000.000000     14.423077
75%          0.000000  4.600000e+04      0.000000   2277.000000     22.373541
max     200000.000000  1.500000e+06  99999.000000   5840.000000   1000.000000
```

Looking into anomalous values in income we were able to find the farm incomes can be negative.

[ ]:
```
psid[psid['farminc'] != 0][income_cols].describe()
```

[ ]:
```
              farminc      annlabinc     labincbus       annhrs      hrwage
count      120.000000     120.000000    120.000000   120.000000  120.000000
mean     29189.725000   24491.858333    114.133333  1860.741667   12.721140
std      33654.453819   18135.461596    933.531183   668.718635    7.682096
min      -5000.000000     400.000000      0.000000   129.000000    2.334267
25%       7000.000000   10748.750000      0.000000  1470.000000    7.179206
50%      20000.000000   21000.000000      0.000000  2000.000000   10.971956
75%      40250.000000   33325.000000      0.000000  2173.250000   16.811773
max     200000.000000  101400.000000  10000.000000  3491.000000   41.666668
```

100% of the dataset has annual labor income greater than 0.

[ ]:
```
psid[psid.annlabinc > 0].shape[0] / psid.shape[0]
```

[ ]:
```
1.0
```

There is no considerable amount of missing values for the main attributes.

[ ]:
```
psid.isnull().sum()
```

[ ]:
```
sex            0
farminc        0
age            0
annhrs         0
annlabinc      0
degree        40
labincbus      0
yrsexp         0
yrsftexp       0
yrsptexp       0
hrwage         0
race         417
region         3
job          189
dtype: int64
```

1.0.3 Classes distribution

See dataset description for more details: https://github.com/jcalvesoliveira/gender-pay-gap/blob/master/docs/datasets/PanelStudyIncomeDynamics.names

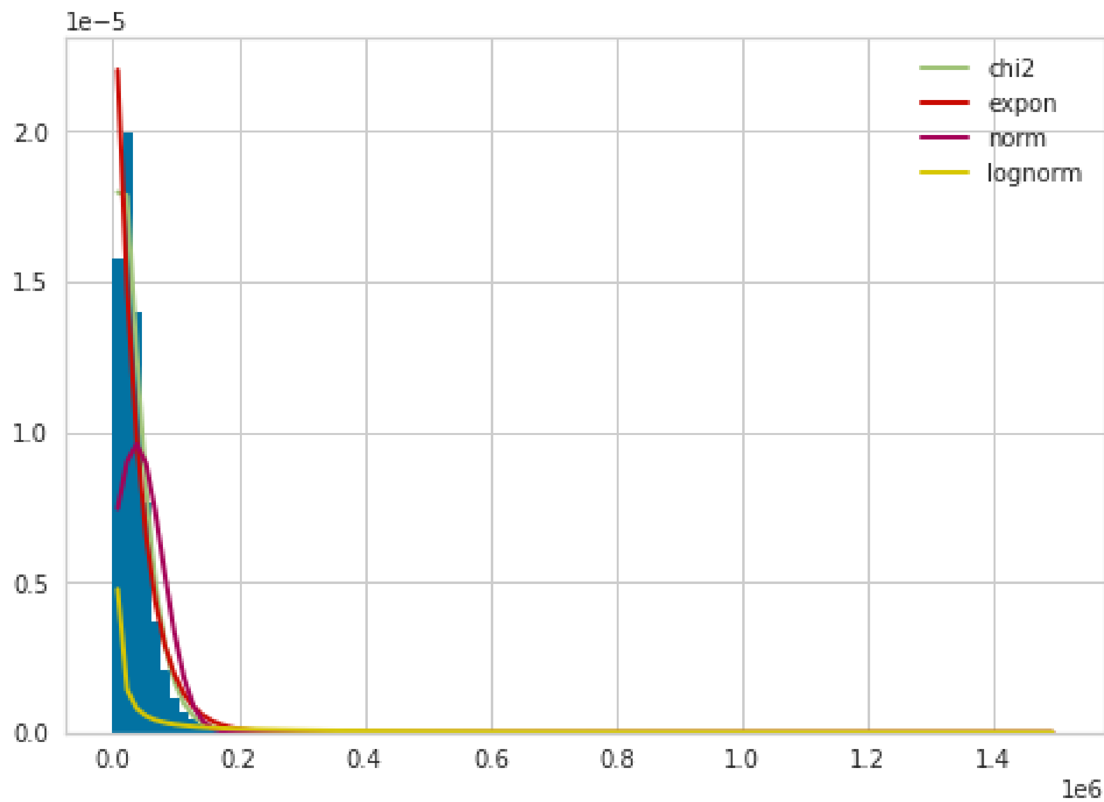
Distribution of labor income

[ ]:
```
f = Fitter(psid.annlabinc,
           distributions=['lognorm',
                          "norm", "expon","chi2"])
f.fit()
f.summary()
```

[ ]:

	sumsquare_error	aic	bic	kl_div
chi2	1.402789e-11	8022.927868	-1.182466e+06	inf
expon	8.829477e-11	6155.790778	-1.121036e+06	inf
norm	2.396454e-10	42587.165871	-1.087689e+06	inf
lognorm	7.046555e-10	3425.914448	-1.051657e+06	inf

```
WARNING:matplotlib.font_manager:findfont: Font family ['sans-serif'] not found.
Falling back to DejaVu Sans.
```



The labor income distribution is skewed to the right, meaning that a small amount of people will make a lot of money. In order to focus our analysis in the overall population we filter out the 10% highest incomes by selecting the 90th percentile.

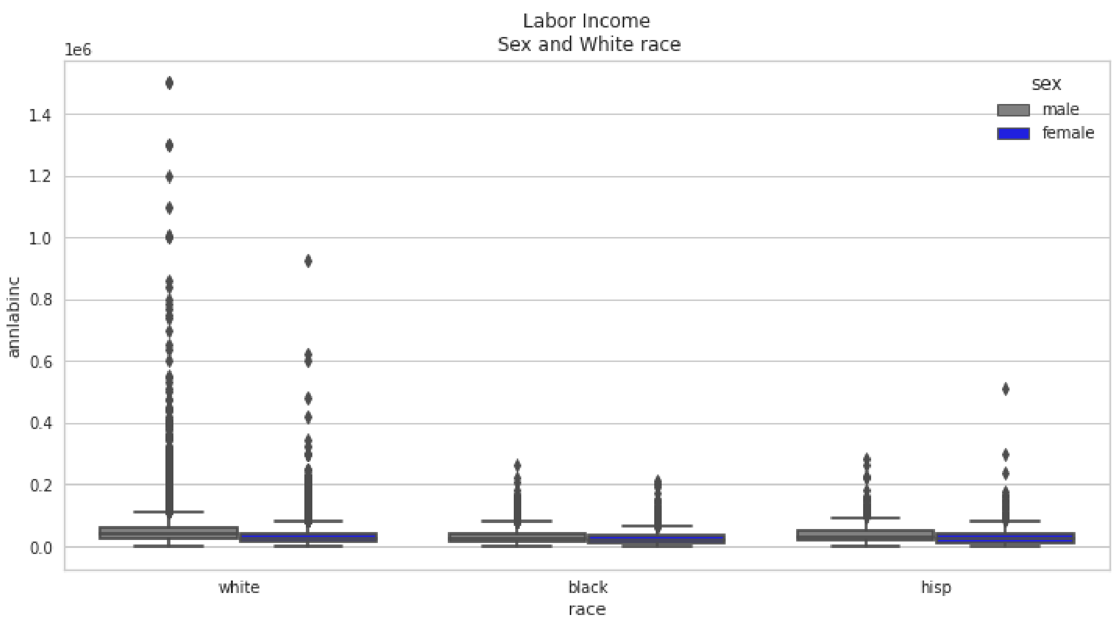
[ ]:
```
q_99 = psid.annlabinc.quantile(0.99)
q_99
```

[ ]: 171000.0

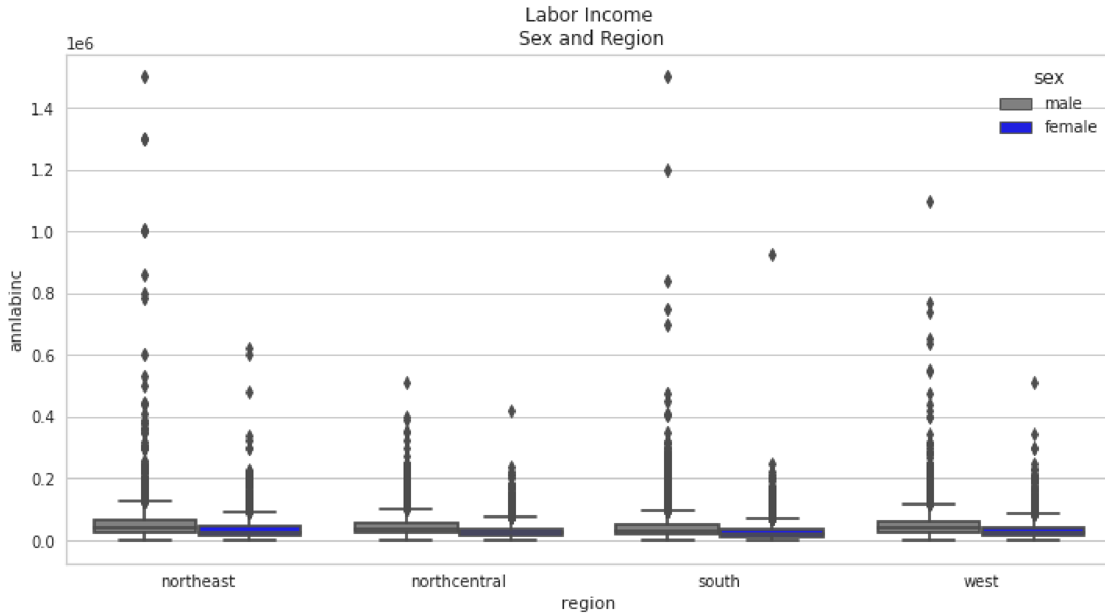
1.0.4 What categorical features may have an effect on income?

```
[ ]: fig, ax = plt.subplots(figsize=(12, 6))
ax.set(title="Labor Income\n Sex and White race")
ax = sns.boxplot(x="race", y="annlabinc", hue="sex",
                 data=psid,
                 dodge=True, ax = ax, palette=['gray','blue'])
```

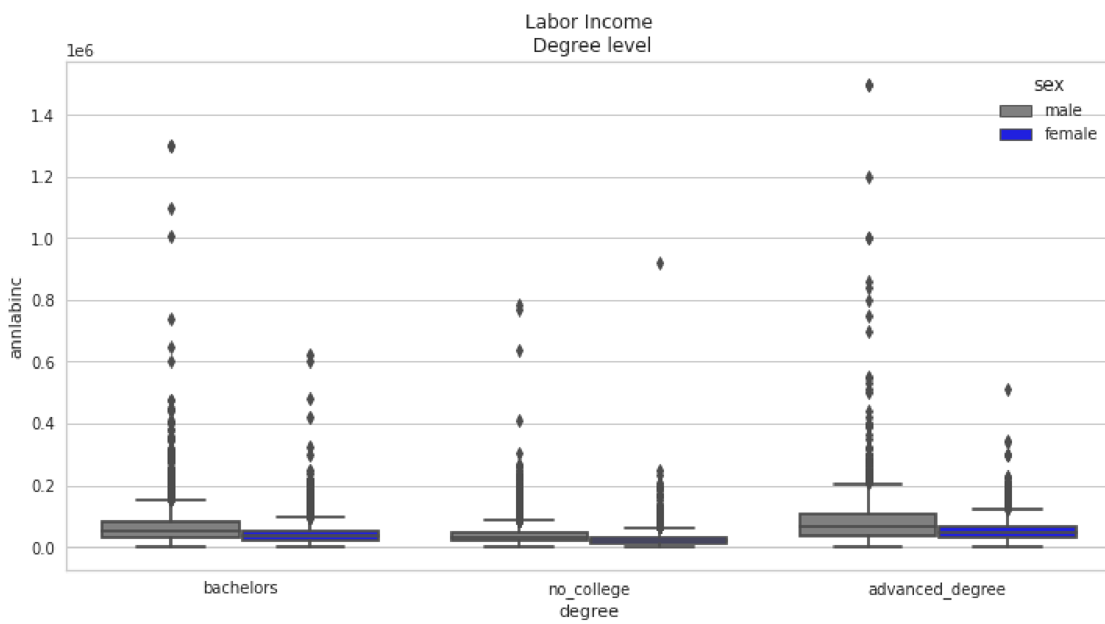
```
WARNING:matplotlib.font_manager:findfont: Font family ['sans-serif'] not found.
Falling back to DejaVu Sans.
WARNING:matplotlib.font_manager:findfont: Font family ['sans-serif'] not found.
Falling back to DejaVu Sans.
```




```
[ ]: fig, ax = plt.subplots(figsize=(12, 6))
ax.set(title="Labor Income\n Sex and Region")
ax = sns.boxplot(x="region", y="annlabinc", hue="sex",
                 data=psid,
                 dodge=True, ax = ax, palette=['gray','blue'])
```



[ ]:
```
fig, ax = plt.subplots(figsize=(12, 6))
ax.set(title="Labor Income\n Degree level")
ax = sns.boxplot(x="degree", y="annlabinc", hue="sex",
                 data=psid,
                 dodge=True, ax = ax, palette=['gray','blue'])
```



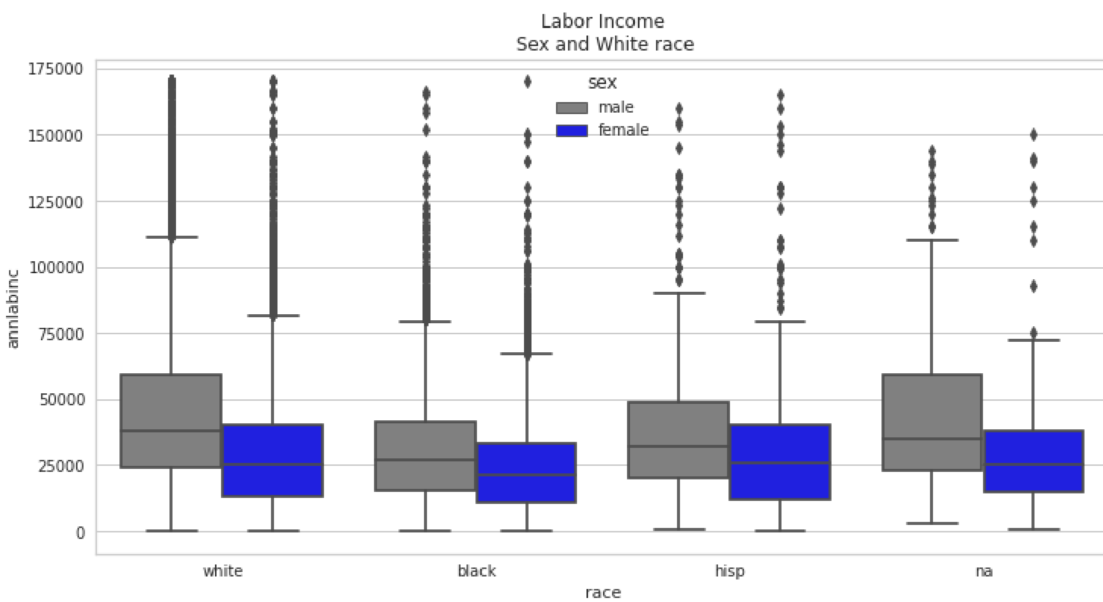
1.1 Data Cleaning

[ ]:
```
psid = psid[psid.annlabinc < q_99]
psid.degree.fillna('no_college', inplace=True)
psid.job.fillna('na', inplace=True)
psid.race.fillna('na', inplace=True)
psid = psid[psid['region'].notna()]
```

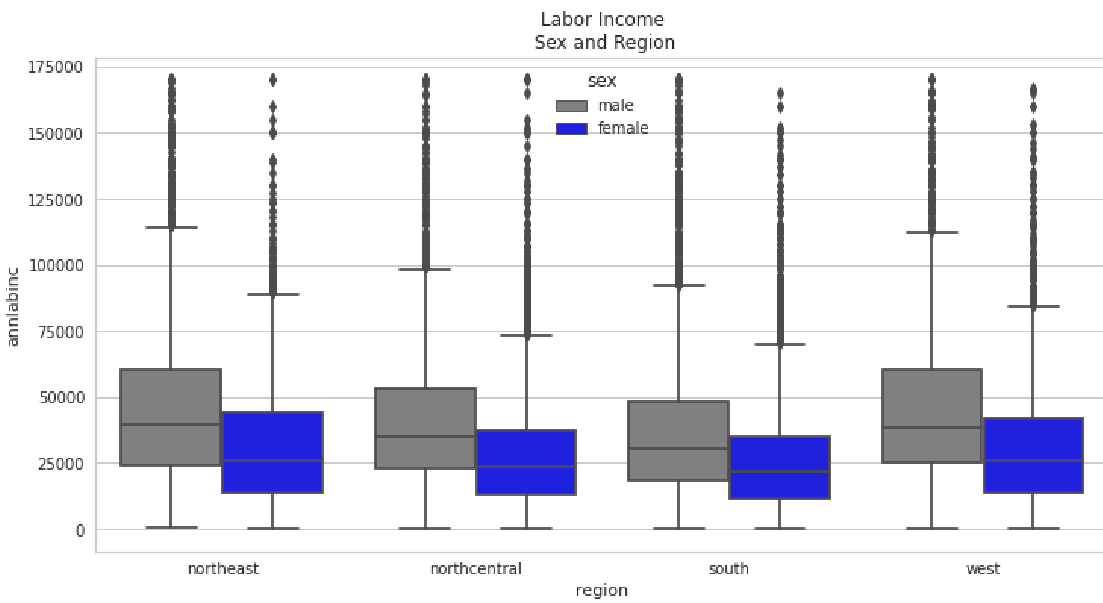
[ ]:
```
psid.to_csv('../data/processed/psid.csv')
```

1.1.1 What categorical features may have an effect on income?

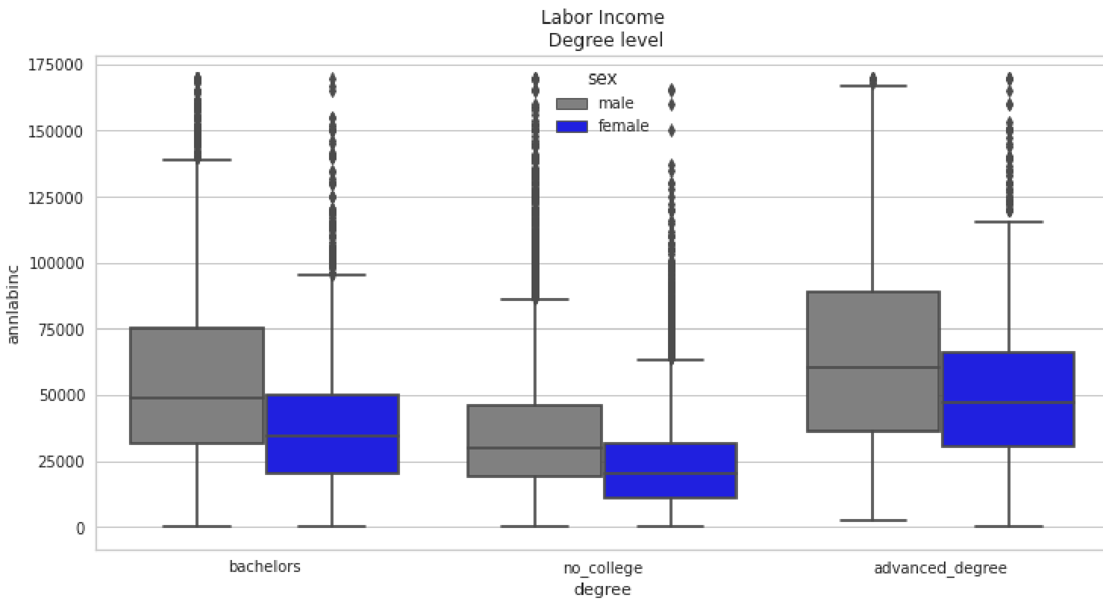
[ ]:
```
fig, ax = plt.subplots(figsize=(12, 6))
ax.set(title="Labor Income\n Sex and White race")
ax = sns.boxplot(x="race", y="annlabinc", hue="sex",
                 data=psid,
                 dodge=True, ax = ax, palette=['gray','blue'])
```



[ ]:
```
fig, ax = plt.subplots(figsize=(12, 6))
ax.set(title="Labor Income\n Sex and Region")
ax = sns.boxplot(x="region", y="annlabinc", hue="sex",
                 data=psid,
                 dodge=True, ax = ax, palette=['gray','blue'])
```



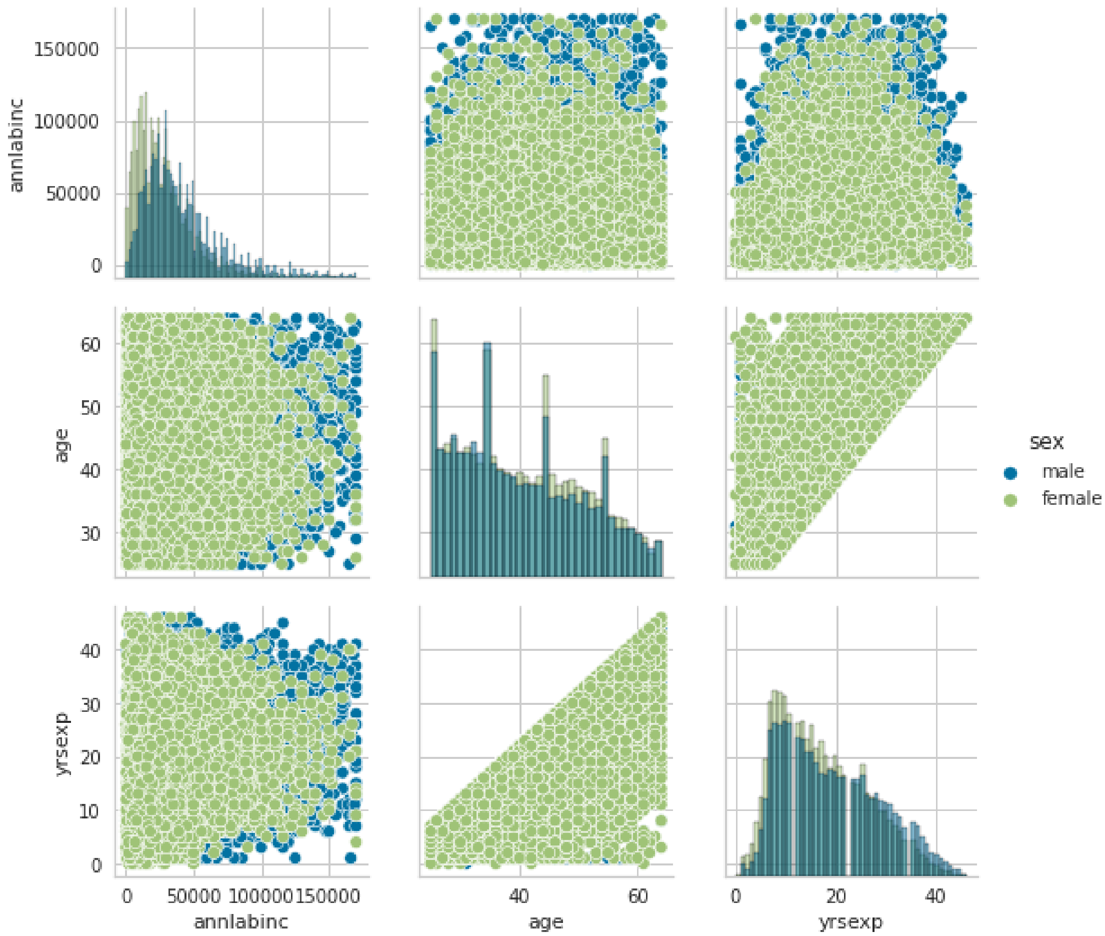

```
[ ]: fig, ax = plt.subplots(figsize=(12, 6))
     ax.set(title="Labor Income\n Degree level")
     ax = sns.boxplot(x="degree", y="annlabinc", hue="sex",
                      data=psid,
                      dodge=True, ax = ax, palette=['gray','blue'])
```



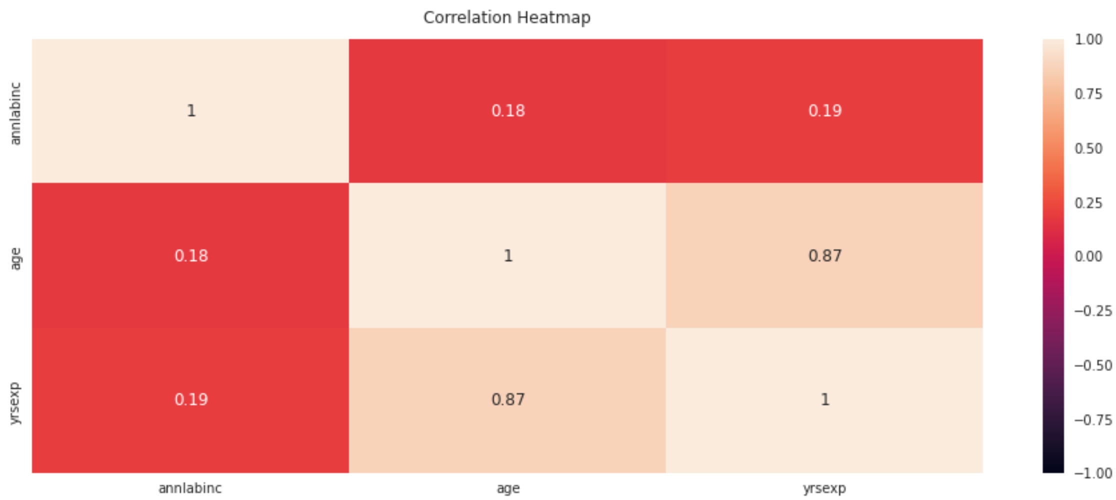
Looking into the plot below, we can observe that different from what we would expect we could not identify a clear relationship between age and years of experience with labor income.

```
[ ]: g = sns.PairGrid(psid[['annlabinc','age','sex','yrsexp']], hue="sex")
     g.map_diag(sns.histplot)
     g.map_offdiag(sns.scatterplot)
     g.add_legend()
```

```
[ ]: <seaborn.axisgrid.PairGrid at 0x7f99e2f51be0>
```




```
[ ]: plt.figure(figsize=(16, 6))
     heatmap = sns.heatmap(psid[['annlabinc','age','yrsexp']].corr(), vmin=-1,␣
      ↪vmax=1, annot=True)
     heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':12}, pad=12);
```



2 Data Transformation

[ ]:
```
psid.drop(['annlabinc','yrsftexp', 'yrsptexp','farminc','labincbus'], axis=1,␣
↪inplace=True)
```

[ ]:
```
psid.age.describe()
```

[ ]:
```
count    33060.000000
mean        40.411827
std         10.397327
min         25.000000
25%         31.000000
50%         39.000000
75%         49.000000
max         64.000000
Name: age, dtype: float64
```

[ ]:
```
psid.annhrs.describe()
```

[ ]:
```
count    33060.000000
mean      1985.416243
std        623.083807
min         10.000000
25%       1764.000000
50%       2000.000000
75%       2268.000000
max       5840.000000
Name: annhrs, dtype: float64
```

[ ]:
```
psid.yrsexp.describe()
```

[ ]:
```
count    33060.000000
mean        18.061162
std          9.383590
min          0.000000
25%         10.000000
50%         17.000000
75%         25.000000
max         46.000000
Name: yrsexp, dtype: float64
```

[ ]:
```
psid['annhrs'] = pd.cut(psid['annhrs'], bins=[0,1764, 2000,␣
↪2268,6000],labels=['0-1764','1764-2000','2000-2268','2268-6000'])
psid['age'] = pd.cut(psid['age'], bins=[0,31, 39,␣
↪49,64],labels=['0-31','31-39','39-49','49-64'])
psid['yrsexp'] = pd.cut(psid['yrsexp'], bins=[0,10, 17, 25,46],␣
↪labels=['0-10','10-17','17-25','25-46'])
```

[ ]:
```
X = psid.drop(['hrwage'], axis=1)
y = psid.hrwage.copy()
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2,␣
↪random_state=42)
print(f"X_train shape: {X_train.shape}, X_test shape: {X_test.shape},␣
↪proportion: {X_train.shape[0] / (X_train.shape[0] + X_test.shape[0])}")
```

```
X_train shape: (26448, 8), X_test shape: (6612, 8), proportion: 0.8
```

[ ]:
```
cat_cols = ['sex','degree','race', 'job', 'region','age','annhrs','yrsexp']
```

[ ]:
```
encoding_transf = ColumnTransformer(remainder='passthrough',
                       transformers=[('ohe', OneHotEncoder(drop='first',␣
↪sparse=False), cat_cols)],
                       verbose_feature_names_out=True)
norm_transf = ColumnTransformer(remainder='passthrough',
                       transformers=[('norm', MinMaxScaler())])

pipeline = Pipeline([
    ('encoding',encoding_transf),
    ('normalization', MinMaxScaler())
])
```

2.0.1 Modeling

[ ]:
```
X_train = pipeline.fit_transform(X_train,y_train)
X_test = pipeline.transform(X_test)
lr = LinearRegression()
lr.fit(X_train, y_train)
```

[ ]:
```
LinearRegression()
```

[ ]:
```
feature_names=list(pipeline.get_feature_names_out())
```

[ ]:
```
predicted = lr.predict(X_test)
predicted
```

[ ]:
```
array([ 6.86714946,  9.76743895,  9.52532006, …,  8.36140376,
       31.97124922, 16.48811779])
```

[ ]:
```
print(f"Hourly wage mean:{y_test.mean():.2f}, std:{y_test.std():.2f}, range:
  ↪{y_test.min():.2f}-{y_test.max():.2f}")
print(f"R2 score: {r2_score(y_test, predicted)}")
print(f"MSE: {mean_squared_error(y_test, predicted)}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, predicted))}")
```

```
Hourly wage mean:17.18, std:12.40, range:0.89-166.56
R2 score: 0.2645586985709846
MSE: 113.07469607048867
RMSE: 10.633658639926743
```

[ ]:
```
X2 = sm.add_constant(X_train)
ols = sm.OLS(y_train, X2).fit()
print(ols.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 hrwage   R-squared:                       0.209
Model:                            OLS   Adj. R-squared:                  0.208
Method:                 Least Squares   F-statistic:                     174.3
Date:                Thu, 21 Apr 2022   Prob (F-statistic):               0.00
Time:                        18:58:08   Log-Likelihood:            -1.0523e+05
No. Observations:               26448   AIC:                         2.105e+05
Df Residuals:                   26407   BIC:                         2.109e+05
Df Model:                          40
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         25.8702      0.701     36.892      0.000      24.496      27.245
x1             3.7306      0.199     18.767      0.000       3.341       4.120
```

```
x2           -4.5190      0.369    -12.248      0.000      -5.242      -3.796
x3          -11.7668      0.364    -32.314      0.000     -12.481     -11.053
x4            1.1109      0.489      2.274      0.023       0.153       2.069
x5           -0.4050      0.746     -0.543      0.587      -1.867       1.057
x6            0.9703      0.195      4.987      0.000       0.589       1.352
x7           -3.0902      0.907     -3.406      0.001      -4.868      -1.312
x8           -9.8649      0.691    -14.274      0.000     -11.220      -8.510
x9            0.3798      0.772      0.492      0.623      -1.133       1.892
x10           2.8755      0.763      3.770      0.000       1.381       4.370
x11          -4.1219      0.603     -6.840      0.000      -5.303      -2.941
x12           0.1491      0.811      0.184      0.854      -1.441       1.740
x13          -8.4714      0.641    -13.222      0.000      -9.727      -7.216
x14           0.4959      0.647      0.767      0.443      -0.772       1.764
x15          -6.5932      0.705     -9.356      0.000      -7.974      -5.212
x16           2.9501      1.114      2.648      0.008       0.766       5.134
x17          -6.1937      0.623     -9.941      0.000      -7.415      -4.973
x18          -0.0587      0.587     -0.100      0.920      -1.208       1.091
x19         -10.6502      1.165     -9.143      0.000     -12.933      -8.367
x20          -4.9092      0.577     -8.515      0.000      -6.039      -3.779
x21          -7.2659      1.052     -6.909      0.000      -9.327      -5.204
x22          -6.6368      0.591    -11.221      0.000      -7.796      -5.477
x23          -0.8042      0.715     -1.125      0.260      -2.205       0.597
x24          -2.9664      0.603     -4.923      0.000      -4.147      -1.785
x25          -1.4587      0.899     -1.623      0.105      -3.220       0.302
x26          -7.2479      0.779     -9.301      0.000      -8.775      -5.721
x27          -6.2271      0.611    -10.191      0.000      -7.425      -5.029
x28           1.3294      0.259      5.124      0.000       0.821       1.838
x29          -0.5995      0.208     -2.888      0.004      -1.006      -0.193
x30           1.6216      0.259      6.254      0.000       1.113       2.130
x31           2.2811      0.279      8.175      0.000       1.734       2.828
x32           4.1627      0.340     12.258      0.000       3.497       4.828
x33           5.4057      0.401     13.464      0.000       4.619       6.193
x34          -0.5791      0.231     -2.511      0.012      -1.031      -0.127
x35          -1.3697      0.237     -5.780      0.000      -1.834      -0.905
x36          -2.1075      0.245     -8.586      0.000      -2.589      -1.626
x37           0.5394      0.270      1.994      0.046       0.009       1.069
x38           1.2180      0.340      3.588      0.000       0.553       1.883
x39           1.3501      0.402      3.361      0.001       0.563       2.137
x40           0.0662      4.099      0.016      0.987      -7.969       8.101
==============================================================================
Omnibus:                    58511.321   Durbin-Watson:                   2.010
Prob(Omnibus):                  0.000   Jarque-Bera (JB):       1819866965.133
Skew:                          19.702   Prob(JB):                         0.00
Kurtosis:                    1287.472   Cond. No.                         92.0
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
```

specified.

2.1 Accesing model

[ ]:
```
shap.initjs()
```

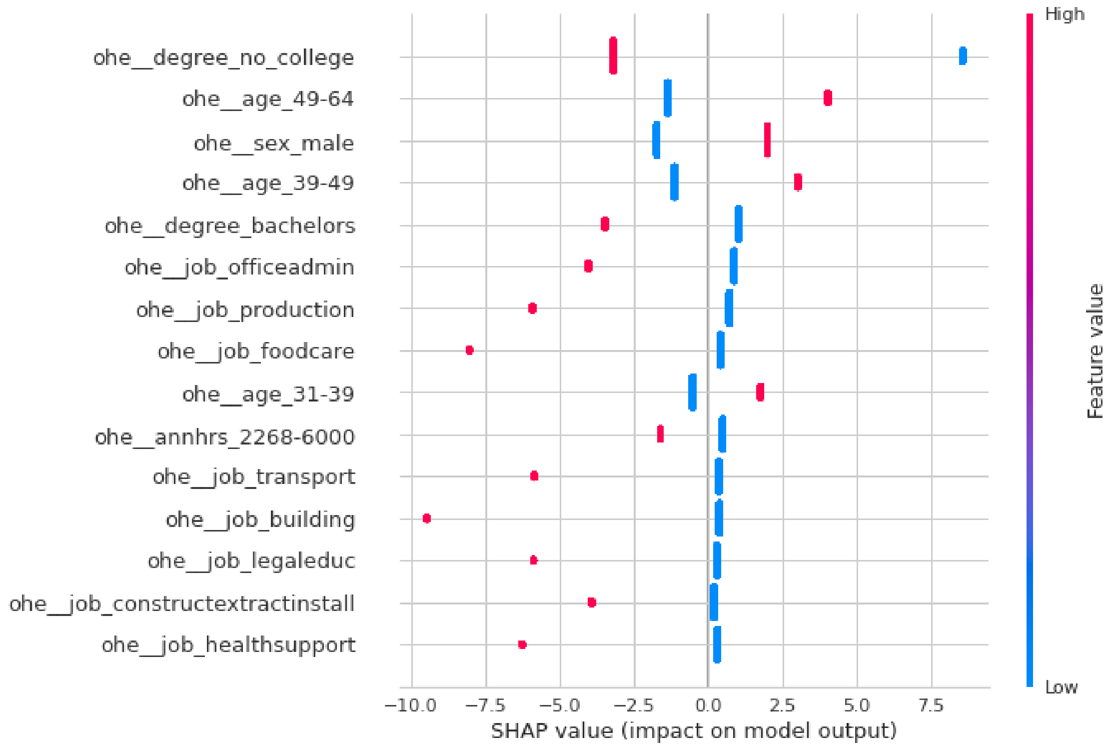
<IPython.core.display.HTML object>

[ ]:
```
X_train_df = pd.DataFrame(X_train, columns=pipeline.get_feature_names_out())
X_test_df = pd.DataFrame(X_test, columns=pipeline.get_feature_names_out())
```

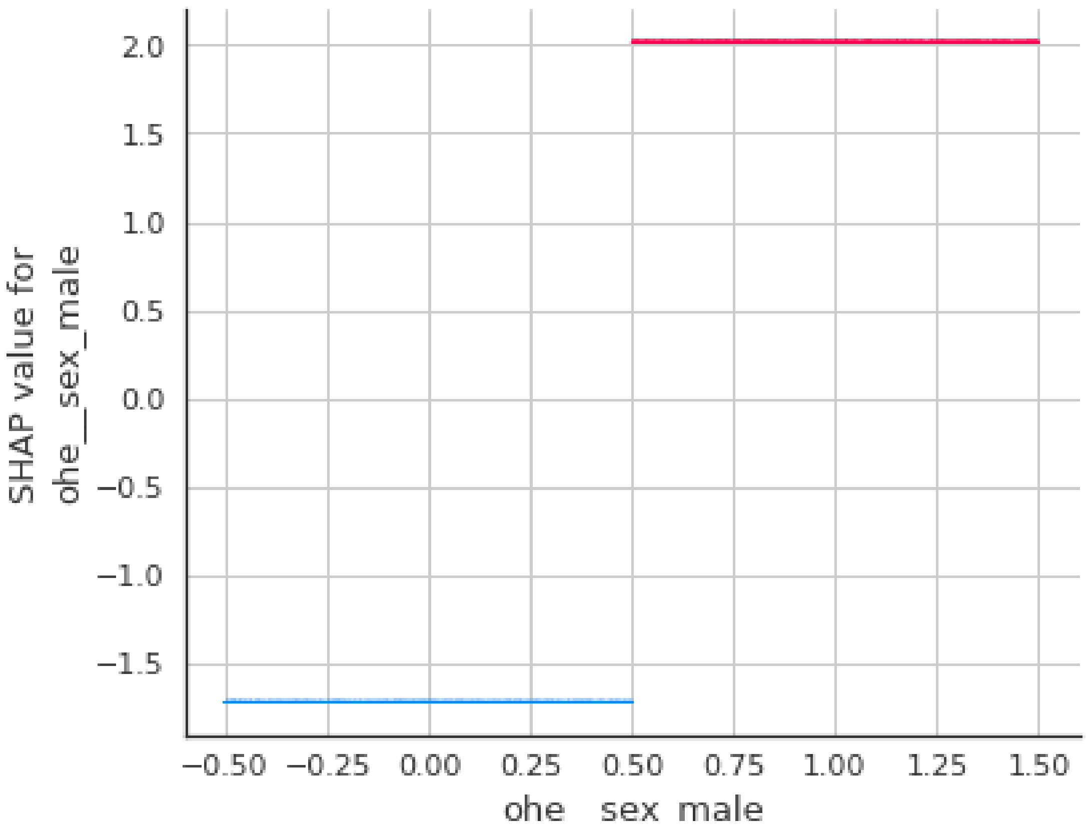
[ ]:
```
masker = shap.maskers.Independent(data = X_train_df)
explainer = shap.LinearExplainer(lr,␣
 ↪masker,feature_perturbation="correlation_dependent")
shap_values = explainer.shap_values(X_test_df)
```

The feature_perturbation option is now deprecated in favor of using the appropriate masker (maskers.Independent, or maskers.Impute)

[ ]:
```
shap.summary_plot(shap_values, X_test_df, max_display=15)
```



[ ]:
```
shap.dependence_plot("ohe__sex_male", shap_values, X_test_df, x_jitter=1,␣
 ↪dot_size=1, interaction_index="ohe__sex_male")
```

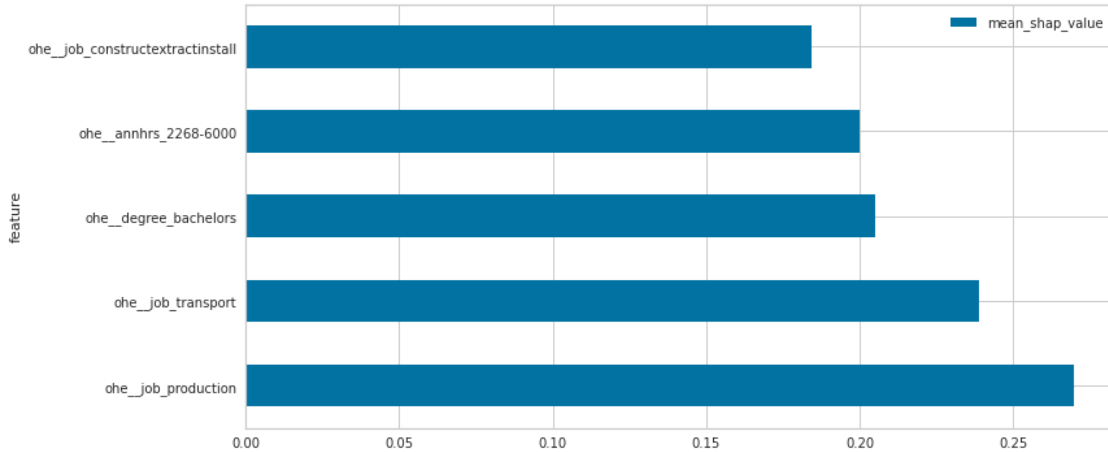


[ ]:
```
female = np.array([True if X_test[i,0] == 0 else False for i in range(X_test.
    ↪shape[0])])
```

[ ]:
```
female_importances = shap_values[female].mean(axis=0)
feature_importances = pd.DataFrame(female_importances,columns =␣
    ↪['mean_shap_value'])
feature_importances['feature'] = feature_names
feature_importances.sort_values('mean_shap_value',ascending=False, inplace=True)
```

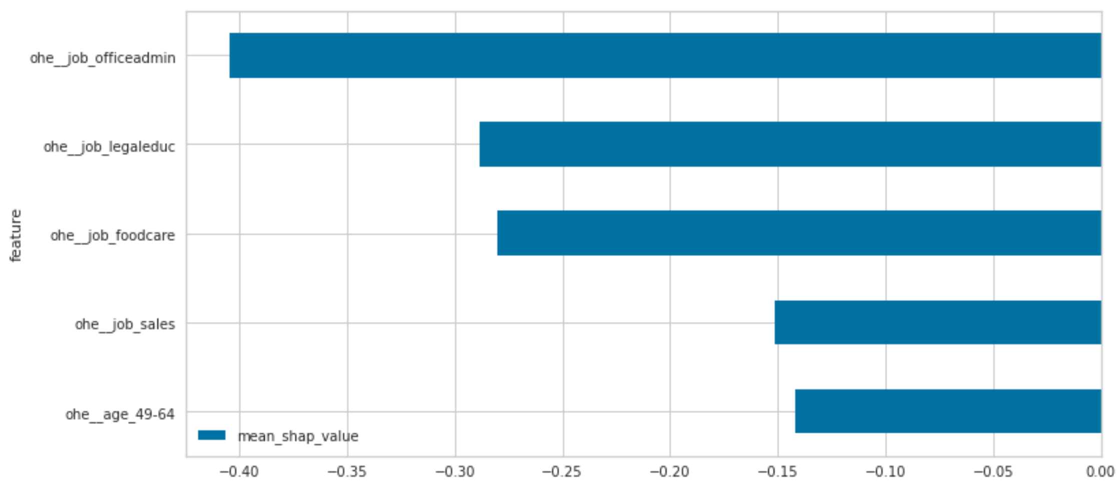
[ ]:
```
feature_importances[0:5].
    ↪plot(kind='barh',x='feature',y='mean_shap_value',figsize=(12,6))
```

[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f99c98e66d0>




```
[ ]: feature_importances[-6:-1].
      ↪plot(kind='barh',x='feature',y='mean_shap_value',figsize=(12,6))
```

```
[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f99c35e0a00>
```




```
[ ]:
```